DERİN SİNİR AĞLARININ FÜZYONU
İLE YÜZ İMGELERİNDEN YAŞ GRUBU
VE CİNSİYET SINIFLANDIRMA

Eren KARAKAŞ

Yüksek Lisans Tezi
Elektrik-Elektronik Mühendisliği Anabilim Dalı
Haberleşme Bilim Dalı
Doç. Dr. İbrahim Yücel ÖZBEK
2019

ATATÜRK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YÜKSEK LİSANS TEZİ

DERİN SİNİR AĞLARININ FÜZYONU İLE YÜZ İMGELERİNDEN YAŞ GRUBU VE CİNSİYET SINIFLANDIRMA

Eren KARAKAŞ

ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI
Haberleşme Bilim Dalı

ERZURUM
2019

T.C.
ATATÜRK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

TEZ ONAY FORMU

DERİN SİNİR AĞLARININ FÜZYONU İLE YÜZ İMGELERİNDEN YAŞ GRUBU VE CİNSİYET SINIFLANDIRMA

Doç. Dr. İbrahim Yücel ÖZBEK danışmanlığında, Eren KARAKAŞ tarafından hazırlanan bu çalışma 27/08/2019 tarihinde aşağıdaki jüri tarafından Elektrik-Elektronik Mühendisliği Anabilim Dalı – Haberleşme Bilim Dalı'nda Yüksek Lisans Tezi olarak **oybirliği/~~oy çokluğu~~ (.../...)** ile kabul edilmiştir.

Başkan : Doç. Dr. Bülent ÇAVUŞOĞLU *İmza* :

Üye : Doç. Dr. Ahmet DUMLU *İmza* :

Üye : Doç. Dr. İbrahim Yücel ÖZBEK *İmza* :

Yukarıdaki sonuç;

Enstitü Yönetim Kurulunun 29/08/2019 tarih ve 34/37 nolu kararı ile onaylanmıştır.

Prof. Dr. Mehmet KARAKAN
Enstitü Müdürü

Not: Bu tezde kullanılan özgün ve başka kaynaklardan yapılan bildirişlerin, çizelge, şekil ve fotoğrafların kaynak olarak kullanımı, 5846 sayılı Fikir ve Sanat Eserleri Kanunundaki hükümlere tabidir.

ÖZET

Yüksek Lisans Tezi

DERİN SİNİR AĞLARININ FÜZYONU İLE YÜZ İMGELERİNDEN YAŞ GRUBU VE CİNSİYET SINIFLANDIRMA

Eren KARAKAŞ

Atatürk Üniversitesi
Fen Bilimleri Enstitüsü
Elektrik-Elektronik Mühendisliği Anabilim Dalı
Haberleşme Bilim Dalı

Danışman: Doç. Dr. İbrahim Yücel ÖZBEK

Günümüzde akıllı uygulamaların her geçen gün daha fazla artması, sistemlerin bazı özelliklerden otomatik olarak çıkarım yapmalarına zemin hazırlamış ve araştırmaları da bu alana kaydırmıştır. İnsan yüzü bu açıdan önem arz etmektedir. Çünkü içerisinden duygu, ifade, yaş ve cinsiyet gibi birçok özellik elde edilebilir. Bu çalışmamızda yüz imgelerinden yaş ve cinsiyet sınıflandırma üzerine çalışmalar yaptık. Sınıflandırma işlemi için Adience veri setini kullanıldı ve çeşitli derin sinir ağlarını oy çokluğu ile füzyon edildi. Veri setinde filtreleme yapılmamış gerçek hayattaki fotoğraflardan oluşturulmuş ve bu sebeple de içerisinde görüntü kalitesi düşük, gürültü seviyeleri yüksek imgeler bulunmaktadır. Bu zorluklara rağmen yüksek başarımlı sonuçlara ulaşılmıştır. Oy çokluğu yöntemi ile test edilen yöntemler füzyon edilip bu füzyon çıktısı gerçek değerlerle karşılaştırılmıştır. Cinsiyet sınıflandırmada %92.29, yaş grubu sınıflandırmada %60,26 başarı elde edilmiştir.

2019, 68 sayfa

Anahtar Kelimeler: Derin Öğrenme, Cinsiyet Sınıflandırma, Yaş Sınıflandırma, Adience, CNN, Majority Voting

ABSTRACT

MS Thesis

AGE GROUP AND GENDER CLASSIFICATION WITH FACIAL IMAGES BASED ON DEEP NEURAL NETWORK FUSION

Eren KARAKAŞ

Ataturk University
Graduate School of Natural and Applied Sciences
Department of Electrical and Electronics Engineering
Communication Science

Supervisor: Assoc. Prof. Dr. İbrahim Yücel ÖZBEK

Nowadays, the increasing number of smart applications has prepared the ground for the systems to make some inferences automatically and the research has been shifted to this field. The human face is important in this respect. Because many features such as emotion, expression, age and gender can be obtained. In this study, we conducted studies on age and gender classification from face images. Adience data set was used for classification and various deep neural networks were fused by majority. The data set is composed of real-life photos without filtering and therefore contains images with low image quality and high noise levels. Despite these difficulties, high performance results have been achieved. The methods tested by majority vote were fused and this fusion output was compared with the actual values. 92.29% success was achieved in gender classification and 60.26% success in age group classification.

2019, 68 pages

Keywords: Deep Learning, Gender Classification, Age Classification, Adience, CNN, Majority Voting

TEŞEKKÜR

Yüksek lisans eğitimim süresince bilgi, birikim ve tecrübelerinden faydalanmama fırsat tanıyan, bana yol gösteren, sabır ve hoşgörüsünü eksik etmeyen danışman hocam Sayın Doç. Dr. İbrahim Yücel ÖZBEK'e, sahip olduğum değer yargılarımın mimarı, beni ben yapan, bugünüme getiren aileme, hayatımın her alanında daima bana destek olup güç veren, yardımını esirgemeyen kıymetli eşime ve en çok da onun için çok değerli olan oyun saatlerimizden bile fedakârlık edip çalışmama imkân veren minik kızım Defne'me sevgi, saygı ve teşekkürlerimi sunarım.

Eren KARAKAŞ
Ağustos, 2019

İÇİNDEKİLER

SİMGELER ve KISALTMALAR DİZİNİ

Kısaltmalar

ADAM	Derived from Adaptive Moment Estimation
BN	Batch Normalization
CNN	Convolution Neural Network
CPU	Central Processing Unit
DE	Doğru Erkek
DK	Doğru Kadın
ESA	Evrişimli Sinir Ağı
FC	Fully Connected
GPU	Graphics Processing Unit
RELU	Rectified Linear Units
ResNet	Residual Network
RMSprop	Root Mean Square Propagation
RoR	Residual of Residual
SGDM	Stochastic Gradient Descent with Momentum
SVM	Support Vector Machine
YE	Yanlış Erkek
YK	Yanlış Kadın

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

1. GİRİŞ

Makine öğrenmesinin alt dalı olan derin öğrenme uygulamalarının giderek yaygınlaşması, büyük veri oluşumunun ve işlemcilerin hız ve güç performanslarındaki yükselişin doğal bir sonucudur. Konuşma tanıma, doğal dil işleme, nesne bulma, nesne takip etme, görüntü sınıflandırma derin öğrenmenin çalışma alanlarından bazılarıdır.

Akıllı telefonlardaki uygulamalar, ekranlarda veya billboardlarda karşımıza çıkan reklamlar, hasta takibinden ilaç üretimi, insansız hava araçları, bilgisayar oyunları derin öğrenmenin çözüm bulmaya veya geliştirmeye çalıştığı problemlerden sadece birkaçıdır.

Bu tezde yaş ve cinsiyet sınıflandırmada başarı oranını arttırmak, hata payını azaltmak, süre ve işlem gücünü minimum seviyeye düşürmek hedeflenmiştir. Çalışmada derin sinir ağlarından birkaç farklı metot kullanılmış ve sonuçlar birbirleri ile karşılaştırılmıştır. En iyi sonuca ulaşabilmek için bazı parametreler değiştirilerek testler tekrar yapılmıştır. Son olarak bu derin sinir ağları oy çokluğu ile füzyon edilerek başarı oranı arttırılmıştır.

Çalışmalarımızda, yaş ve cinsiyet sınıflandırma alanında sıkça kullanılan ve 2014 yılında Eran Eidinger, Roee Enbar ve Tal Hassner tarafından oluşturulan Adience (Eidinger *et al.* 2014) datasetini kullandık. Adience datasetinde, verilerin gerçek hayattaki görüntüleme koşullarının zorluklarına mümkün olduğu kadar yakın olması hedeflenmiştir. Veriler, önceden herhangi bir filtreleme olmadan Iphone 5 veya sonraki model akıllı telefonlardan otomatik olarak Flickr'a[1] yüklenen resimlerden oluşur. Filtreleme yapılmamasından dolayı poz, aydınlatma şartları ve görüntü kalitesi açısından büyük değişimler mevcuttur. Dataset, yüzler (faces) ve hizalanmış yüzler (aligned faces) olmak üzere 2 gruptan oluşur. 2284 kişinin 26850 adet fotoğrafı

[1] Fotoğraf saklama, düzenleme ve paylaşım platformu

bulunmaktadır. Cinsiyet sınıflandırma için 2 sınıf (kadın / erkek), yaş sınıflandırma için 8 sınıf (0-2, 4-6, 8-13, 15-20, 25-32, 38-43, 48-50, 60-) vardır.

Beş-katlı çapraz doğrulama sayesinde aynı yöntemi 5 farklı eğitim ve test kümelerinde çalıştırmış oluyoruz. Bu yöntemde genel başarı hesaplanırken elde edilen 5 sonucun ortalaması alınır. Beş-katlı çapraz doğrulama yöntemi ile veri kümesi parçalara ayrılır ve her parçanın hem eğitim hem de test için kullanılabilmesi sağlanmış olur.

Çalışmalarımızı Matlab programının Deep Learning Toolbox'ını kullanarak yaptık.

2. KAYNAK ÖZETLERİ

(Eidinger *et al.* 2014) bu makalelerinde yaş ve cinsiyet tanıma çalışmalarını kolaylaştırmak için kendilerinin oluşturduğu Adience veri setini sunmuşlardır. Veri seti oluşturulurken önce fotoğraflar Viola ve Jones yüz dedektöründe (Jones and Viola 2001) çalıştırılmış ve ardından başka bir kodun değiştirilmiş versiyonu kullanılarak yüz özellik noktaları tespit edilmiştir. Muhtemelen son "selfie" eğiliminden dolayı, bu albümlerdeki birçok yüz farklı yuvarlanma açılarına sahiptir. Bu yüzleri kaçırmamak için, yüz algılama işlemi her görüntüye 5° artışlarla 360° derece döndürülmüş şekilde uygulandı. Yüz özelliği saptama adımının başarısız olduğu yüzler çok gürültülü veya küçük olarak kabul edildi ve atıldı. Aynı kod kullanılarak tahmin edilen poz ayrıca, 0°'den 45°'ye kadar daha fazla açılı olan yüzleri atmak için de kullanıldı. Son olarak, hem görüntüler hem de mevcut bağlamsal bilgiler (resim etiketleri ve ilgili metin, aynı albümdeki ek fotoğraflar vb.) kullanılarak yaş, cinsiyet ve kimlik için elle etiketlendi.

(Levi and Hassner 2015) çalışmalarında otomatik yüz tanıma özellikleri ile yaş ve cinsiyet tahmini yöntemleri arasındaki boşluğu kapatmaya çalışmışlardır. Ağlarını, filtre uygulanmamış yüz görüntülerinin yaş ve cinsiyet sınıflandırması için o dönem yeni çıkan Adience data setiyle test etmişlerdir. Yalın bir ağ mimarisi kullanmalarının dışında, overfitting riskini daha da sınırlamak için iki ek yöntem uygulamışlardır. İlk olarak dropout uygulamışlardır. Dropout ile ağdaki bazı bağlantılar rastgele olarak kapatılmıştır. İkinci olarak dataaugmentation (veri büyütme) uygulamışlar. Dataaugmentation 256 × 256 giriş görüntüsünden rastgele 227 × 227 piksel bir ürün alarak veri artırımı işlemidir.

(Wolfshaar *et al.* 2015) yaptıkları çalışma ile farklı bir bakış açısı getirerek mevcut makine öğrenme algoritmalarının cinsiyet sınıflandırmalarına karşı uygulanabilirliğini araştırmak için bir hibrit makine öğrenme sistemi tasarlamışlardır. Önerilen yaklaşım, önceden tanımlanmış bir evrişimsel sinir ağının (CNN) bir lineer destek vektör makinesi (SVM) ile birleştirilmesine dayanmaktadır. CNN'leri cinsiyet tanıma için kullanmanın en büyük motivasyonu, yüz tanıma ve doğrulama için gösterdikleri başarılarıdır.

Genellikle SVM sınıflandırıcıları çeşitli görüntü tanımlayıcılarıyla beslenerek görüntüler üzerinde cinsiyet tahmini yapmak için kullanılırlar. Sorunun zorlu doğasına rağmen, en son sınıflandırma oranlarının nispeten kısa eğitim süreleri kullanılarak elde edilebileceğini göstermişlerdir. Her iki veri kümesinde de en iyi sonuçlar ince ayarlı ağlar kullanıldığında elde edilmiştir. Nihai sınıflandırıcıların puanlarının ortalaması alınarak yapılan aşırı örneklemenin, tüm durumlarda sınıflandırma oranlarını iyileştirdiği gösterilmiştir.

(Ekmekji 2016) çalışmasında (Levi and Hassner 2015) in yukarıda bahsettiğimiz çalışmasından yararlanarak bu mimarileri geliştirip genel performansı artıracak bir sistem oluşturmayı hedeflemiştir. Proje için veri setini 6 özel katlama katmanına bölmüş, ancak daha sonra bu kıvrımların her birini erkeklere, kadınlara ve 8 yaş grubunun her birine bölmüştür. Bu, eğitim alan sınıflayıcı türlerine göre, orijinal 6 katın her birinin 11 gruba bölündüğü toplam 66 “alt kat” ile sonuçlandı. 6. orijinal kattan gelen alt katların tümü, asla eğitilmemesi için test verileri olarak ayrıldı. Sonra kalan 5 kat ve alt katları eğitim ve çapraz doğrulama için kullanıldı. Bu projenin en zor kısmı, verileri uygun bir şekilde katlara bölmek, her sınıflandırıcıyı eğitmek, çapraz onaylamak ve sonuçta sınıflandırıcıları teste hazır bir sınıflandırıcıda birleştirmek için eğitim altyapısını kurmaktı.

(Rodríguez *et al.* 2017)‘nin modeli VGG-16 CNN'e dayanmaktadır ve Tensorflow ile uygulanmaktadır. Imagenet gibi genel görevlerde ön eğitimden daha iyi performans gösterdiği kanıtlandığından, CNN ağırlıklarını başlatmak için etki alanına özgü ön eğitim kullanmışlardır.

(Liu *et al.* 2017) 22 katman içeren klasik GoogLeNet'i eğitim ağı olarak kullanmışlardır. Transfer öğrenmesine dayanarak eğitilmiş modeli, bir eğitim öncesi model olarak ele alıyorlar ve farklı ırkların tanınma doğruluğunu arttırmak için CAS-PEAL-R1 data seti ile adım adım eğitiyorlar. Deneysel sonuçlar, yaş ve cinsiyet sınıflandırmasının hassasiyetinin ve performansının, transfer öğrenmeye ve artımlı eğitime dayanan

geleneksel çekirdek ön hazırlık yöntemiyle geliştirildiğini göstermektedir. Öte yandan, çoklu GPU'lar modeli bir CPU'dan daha hızlı eğitebileceğini göstermiş oldular.

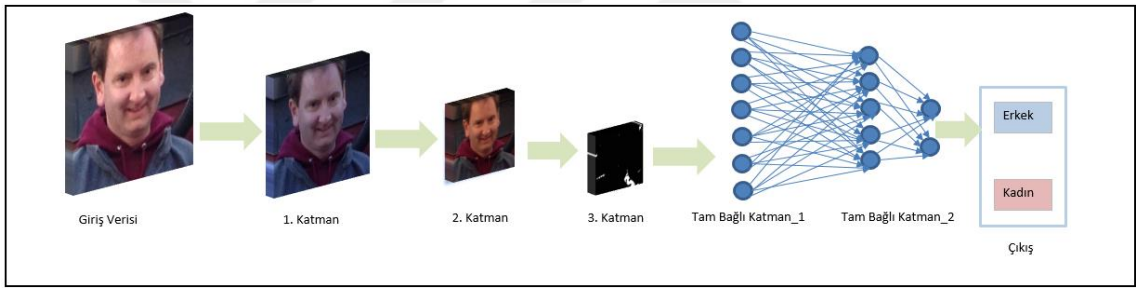
(Akbulut vd 2017) yaptıkları çalışmada derin öğrenmenin 2 farklı algoritması olan Evrişimsel Sinir Ağları ve Yerel Alıcı Alanları kullanarak cinsiyet sınıflandırma yapılmış ve bu iki algoritmanın başarımları karşılaştırılmıştır. Çalışmada Adience data setinin hizalanmış yüz imgeleri kullanılmıştır. Neticede iki algoritmanın da %80 üzerinde başarı oranına sahip olmasına karşın evrişimsel sinir ağlarının diğer algoritmaya kıyasla daha iyi başarı oranına sahip olduğunu belirtmişlerdir.

(Zhang *et al.* 2017) bu makalelerinde yüksek çözünürlüklü yüz görüntülerinin doğal yaşamdaki yaş ve cinsiyet sınıflandırması için yeni bir Residual of Residual ağı (RoR) mimarisi önermektedir. ImageNet'te önceden eğitim yapmak, over-fitting'i azaltmak için kullanılır. IMDB-WIKI-101'de daha fazla ince ayar yapmak, yüz görüntülerinin özelliklerini öğrenmek içindir. İki mekanizma içeren RoR veya Pre-RoR ile doğal yaşamdaki yaş ve cinsiyet sınıflandırması için yeni bir modern performans elde etmişlerdir. RoR ve Pre-RoR arasındaki tek fark Pre-RoR uygulamasında Batch Normalization ve ReLu katmanlarının konvolüsyon katmanından önce gelmesidir. RoR modeli ile yüksek başarı oranları elde edilmiştir.

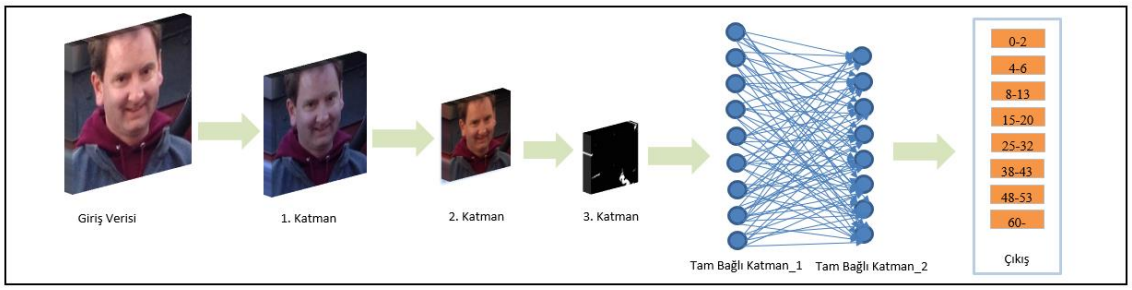
3. MATERYAL ve YÖNTEM

3.1. Problem Tanımı

İnsan yüzü duygu, ifade, cinsiyet, kimlik ve yaş gibi bir çok özelliği içinde barındırmaktadır. Akıllı uygulamaların günümüzde hızla büyümesi, yüz özelliklerinin otomatik olarak çıkarılmasının önemini gün yüzüne çıkarmıştır. Yaş tahmini erişim kontrolü, insan-bilgisayar etkileşimi, yetkilendirme ve kolluk kuvvetleri tarafından gözetleme, arama, takip etme gibi alanlarda kullanılmaya başlanmıştır (Tivive and Bouzerdoum 2006).



Şekil 3.1. Cinsiyet sınıflandırması örnek CNN ağı



Şekil 3.2. Yaş sınıflandırması örnek CNN ağı

Bu tezde yüz imgelerinden faydalanarak yaş tahmini ve cinsiyet sınıflandırması sorununa yüksek başarımlı çözümler bulmaya çalıştık. Buradaki ana zorluk, farklı özellik çıkarıcılarını kullanarak, hatta onları birleştirip birlikte kullanarak yüksek doğruluk oranına sahip sistemlerin tasarımı ve uygulamasıdır. Birçok engel ve gürültü

yanlış sonuçlara yol açabilir. Fotoğraflardan yüz özelliklerini doğru bir şekilde çıkarabilmek sınıflandırmanın da aynı oranda doğruluğunu arttıracaktır. Ancak günlük hayatta karşılaşılabilecek fotoğraflar herhangi bir filtreleme yapılmamış veya arka plana, poza ve ışıklandırmaya dikkat edilmemiş, görüntü kalitesi düşük gürültü seviyesi yüksek olabilecek fotoğraflardır. Bu fotoğraflarla test yapmak hata oranını arttırabilir.

Bu nedenle verilerini günlük hayattan filtreleme yapılmamış fotoğraflardan oluşturan Adience data setini kullanmayı tercih ettik ki elde edeceğimiz sonuçlar doğal yaşama ve gerçekliğe daha yakın sonuçlar olsun.

Bu çalışmamızda cinsiyet sınıflandırma için hizalanmış yüzler grubundaki verileri kullandık. Bu gruptan cinsiyeti belirsiz olarak etiketleme yapılan kişilere ait imgeler çıkarılarak toplam 17492 adet imge ile çalışma yapılmıştır. Aynı şekilde yaş sınıflandırma için de hizalanmış yüzler grubundaki 17523 adet imgeyi kullandık. Çizelge 3.1 ve Çizelge 3.2' de yaş ve cinsiyet sınıflandırma için kullandığımız verilerin katlara göre dağılımı verilmiştir.

Çizelge 3.1. Beş-kat çapraz doğrulama cinsiyete göre imge dağılımı

Cinsiyet	1.Kat	2.Kat	3.Kat	4.Kat	5.Kat	Toplam
Kadın	1948	1998	1774	1804	1848	9372
Erkek	2047	1611	1363	1502	1597	8120
Toplam	3995	3609	3137	3306	3345	17492

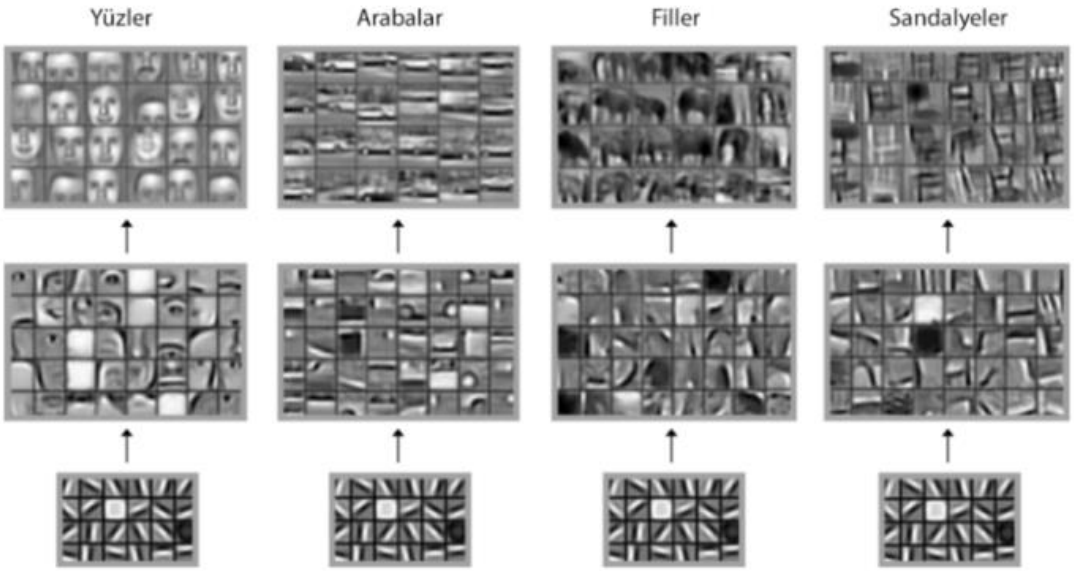
Çizelge 3.2. Beş-kat çapraz doğrulama yaşa göre imge dağılımı

Yaş	1.Kat	2.Kat	3.Kat	4.Kat	5.Kat	Toplam
0-2	960	84	810	151	483	2488
4-6	494	480	358	238	570	2140
8-13	213	600	475	497	340	2125
15-20	152	525	270	468	227	1642
25-32	1646	635	774	970	1056	5081
38-43	555	485	276	522	507	2345
48-53	219	146	120	104	241	830
60-	139	156	202	118	257	872
Toplam	4378	3111	3285	3068	3681	17523

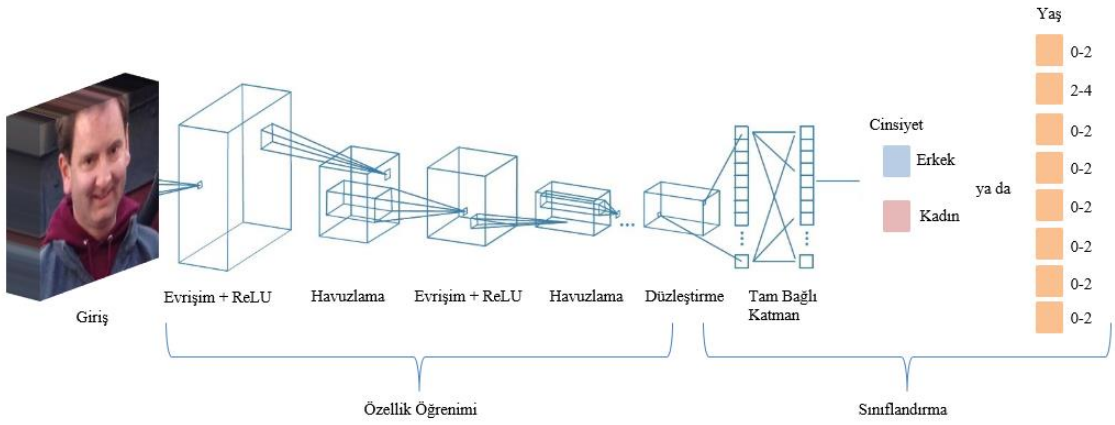
3.2. Önerilen Metotlar

3.2.1. Evrişimsel sinir ağı (CNN)

Evrişim, iki boyutlu giriş matrisinden (genellikle imge) öz nitelik çıkarmak için matris üzerinde filtre gezdirme işlemidir (Kutlu 2018). Evrişimsel sinir ağı derin öğrenmenin en yaygın kullanılan mimarilerinden bir tanesidir. ESA'lar görüntü sınıflandırma, nesne tanımlama, görüntü segmentasyon v.b. problemlerde başarılı bir şekilde uygulanmaktadır. Şekil 3.3'de örnek bir ESA yapısı gösterilmiştir. Bu yapıya göre ilk birkaç katman konvolüsyon (Convulation) ve havuzlama (Pooling) katmanlarından oluşur. Son katman ise tam bağlı katmandan oluşur ve ardından sınıflandırma katmanı gelir. Kısacası ESA'lar ard arda yerleştirilmiş birden fazla eğitilebilir katmanlardan oluşur. Devamında eğitici bir sınıflandırıcı ile devam edilir. Girişte veriler alındıktan sonra katmanlardan sırayla işlemler yapılarak eğitim gerçekleştirilmiş olur. En sonunda olması gereken sonuç ile karşılaştırma yapmak için bir final çıktısı verilir. Tahmin edilen ile gerçek değerin farkı kadar bir hata oluşur. Bu hatanın bütün ağırlıklara aktarılması için geriye yayılım algoritması kullanılır. Her bir iterasyonla ağırlıkların güncellenmesi yapılarak hatanın azaltılması sağlanır. ESA'larda görüntü sınıflandırma işlemlerinde, Şekil 3.4'te görüldüğü gibi pikseller, kenar kombinasyonundan oluşan motifleri, bu motifler birleşerek nesne parçalarını ve nesne parçaları birleşerek nesneleri oluşturur (LeCun *et al.* 2015).



Şekil 3.3. Giriş katmanlarından çıkış katmanlarına doğru gidildikçe oluşan özellikler

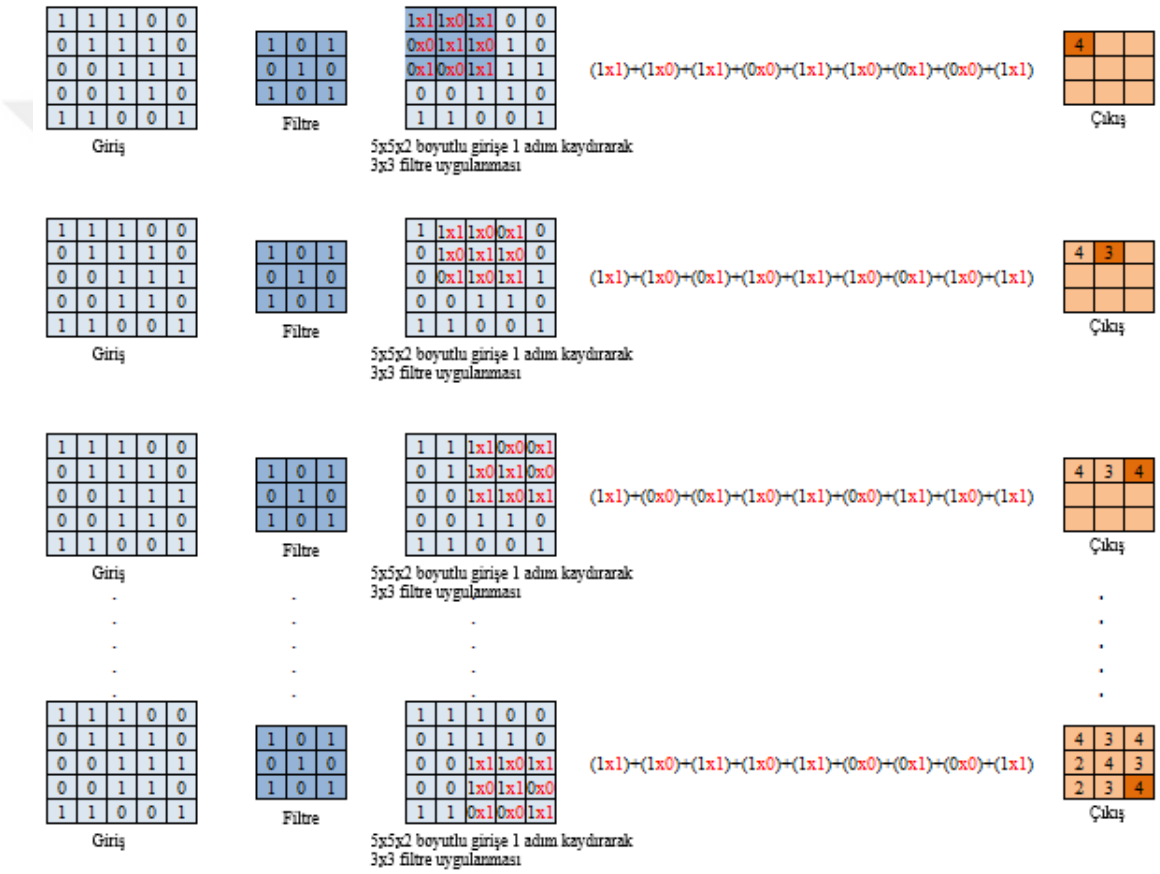


Şekil 3.4. Evrişimsel sinir ağı (ESA) genel mimarisi

3.2.1.a. Evrişim katmanı

Evrişim katmanı ile öznitelikler çıkarılır ve çoğaltılır. Evrişim basitçe filtreleme işlemidir. Filtreler imge boyunca kaydırılır. Kaydırma sırasında filtrenin üzerinde bulunduğu imge piksel değerleri ile filtredeki değerler çarpılır ve elde edilen değerler toplanır ve net sonuç bulunur. Bu işlemi tüm görsele uyguladığımızda yeni bir imge elde etmiş oluruz. Elde edilen imgeye evrişim özniteliği (Convolved Feature) denir. Her

bir filtre aslında yapay sinir ağlarına ait bir nörondur. Bu nöronun ağırlıkları ise filtre içerisindeki değerlerdir. Filtre boyutu büyüdükçe çıkış matrisi küçülecektir. Bu küçülme giriş matrisindeki bilgileri kaybetme anlamına gelmektedir. Dolayısıyla genellikle 3x3, 5x5, 7x7 boyutunda filtreler giriş matirisi üzerinde gezdirilmektedir. Şekil 3.5'te 5x5 boyutundaki girişe 3x3 boyutundaki filtrenin gezdirimi ve 3x3 boyutunda evrişim matrisinin oluşumu gösterilmiştir (Kutlu 2018).



Şekil 3.5. 5x5 boyutunda iki boyutlu girişe, 1x1 kaydırma(stride) ile 3x3'lük bir filtre uygulanarak konvolüsyon özniteliğinin oluşması

Evrişim katmanından sonraki çıkış görüntüsü ile giriş görüntüsü arasında aşağıdaki gibi bir bağlantı vardır (Goodfellow *et al.* 2016).

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a) \qquad (3.1)$$

Denklem (3.1)'de w filtre, x girdi, t zaman ve s sonuç olarak ifade edilmiştir.

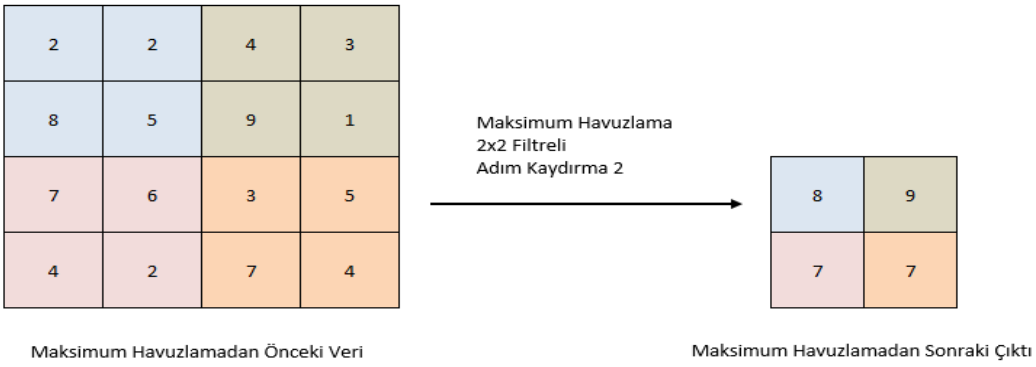
Girdi olarak resim gibi iki boyutlu bir girdi kullanıldığında evrişim işlemi Denklem (3.2)'de ki gibi olur (Goodfellow *et al.* 2016).

$$S(i,j) = (I * K)(i,j) = \sum_{m}\sum_{n} I(i,j)K(i-m, j-n) \tag{3.2}$$

Yukarıdaki denklemde belirtilen i ve j terimleri evrişim işleminden sonra ki yeni matrisin boyutlarını, m ve n terimleri ise filtrenin boyutlarını göstermektedir.

3.2.1.b. Adım kaydırma (Stride)

S adımı, evrişim veya havuzlama işlemi için her işlemden sonra pencerenin hareket ettiği piksel sayısını belirtir.

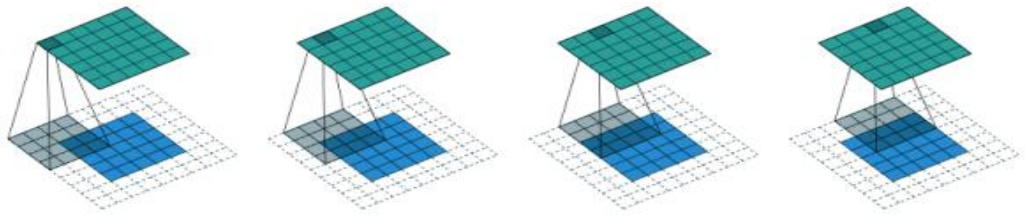


Şekil 3.6. Maksimum havuzlama 2 adım kaydırma

3.2.1.c. Doldurma (padding)

Dolgu, girişteki matrisin etrafına sıfır eklenmesi işlemidir. Evrişim işlemi sonrasında çıktının boyutunun kontrol edilmek istenmesi durumunda dolgu kullanılır. Şekil 3.7'de dolgu uygulanmış evrişim örneğini görebilirsiniz. Eğer girişteki boyutun çıkışta

değişmesini istemiyorsanız dolguyu “1” seçmeniz gerekmektedir. Bu duruma “yarım dolgu” (half-padding) adı verilmektedir.



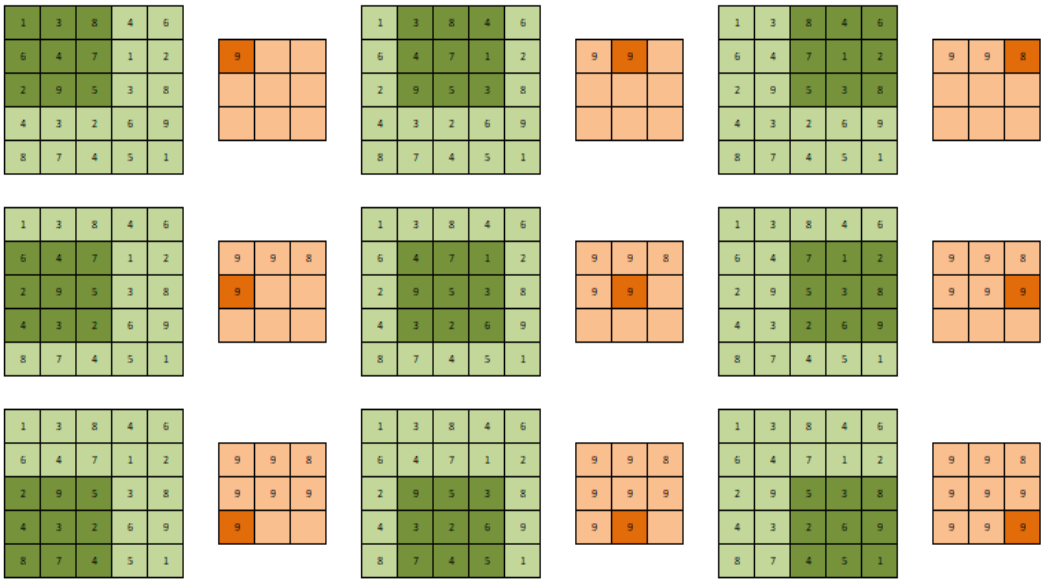
Şekil 3.7. Padding uygulanmış evrişim örneği

En genel haliyle ifade edecek olursak padding, stride, filtre, giriş boyutu ve çıkış boyutu arasında Denklem (3.3)’te ki bağıntıyı verebiliriz. Girdi boyutu (i), filtre boyutu (k), dolgu değeri (p) ve adım sayısı (s) olarak belirtilmiştir.

$$o = \left[\frac{i + 2p - k}{s}\right] + 1 \tag{3.3}$$

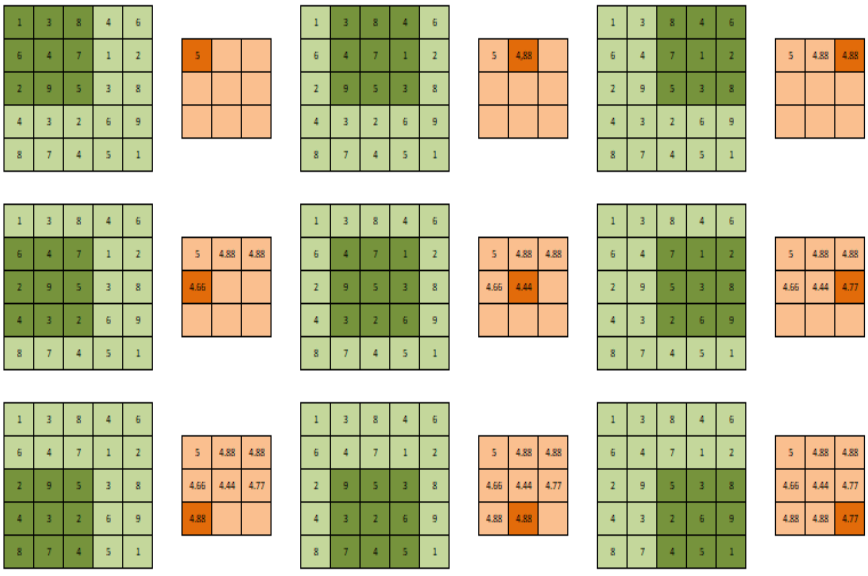
3.2.1.d. Havuzlama katmanı

Havuzlama işlemleri, ortalama veya maksimum değeri alma gibi alt bölgeleri özetlemek için bazı işlevleri kullanarak özellik haritalarının boyutunu azaltır. Girdide havuzlama işlemi kayan pencere yöntemiyle uygulanmaktadır. Kayan pencere her seferinde belirlenen havuzlama yöntemine göre denk geldiği girdi alanından bir değer oluşturur ve bunu çıktı katmanına ekler (Dumoulin and Visin 2018). Havuzlama katmanı hesaplamaları, evrişim katmanı hesaplamalarına göre daha az maliyetlidir. Havuzlama türlerinden en yaygın olarak kullanılan maksimum havuzlama, girdinin parçalara ayrılıp her bir parçadaki en yüksek değerin alındığı havuzlama türüdür. 3x3'lük maksimum havuzlama işleminin 5x5'lik girdi üzerinde 1x1'lik adım sayısı ile hesaplanması Şekil 3.8’de gösterilmiştir (Dumoulin and Visin 2018).



Şekil 3.8. Maksimum havuzlama

Havuzlama türlerinden bir diğeri ortalama havuzlamada da, girdi aynı maksimum havuzlamadaki gibi parçalara ayrılır ancak bu sefer her bir parçadaki değerlerin ortalaması alınır. 3x3'lük ortalama havuzlama işleminin 5x5'lik girdi üzerinde 1x1'lik adım sayısı ile hesaplanması Şekil 3.9'da gösterilmiştir (Dumoulin and Visin 2018).



Şekil 3.9. Ortalama havuzlama

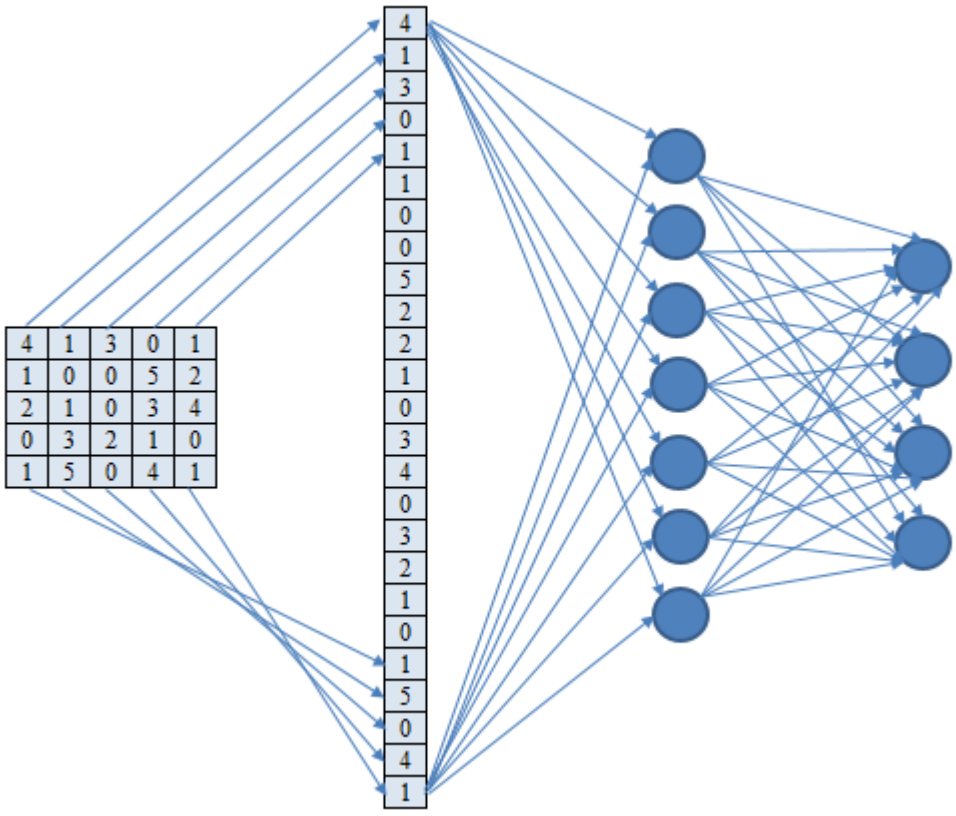
Havuzlama katmanında işlem gördükten sonra oluşan matrisin boyutu (o), girdi boyutu (i), işlemin gerçekleştirileceği pencere (filtre) boyutu (k) ve adım sayısı (s) ile gösterilsin.

Bu durumda oluşan bağıntı Denklem (3.4)'tür.

$$o = \left[\frac{i-k}{s}\right]+1 \tag{3.4}$$

3.2.1.e. Tam bağlantılı katman

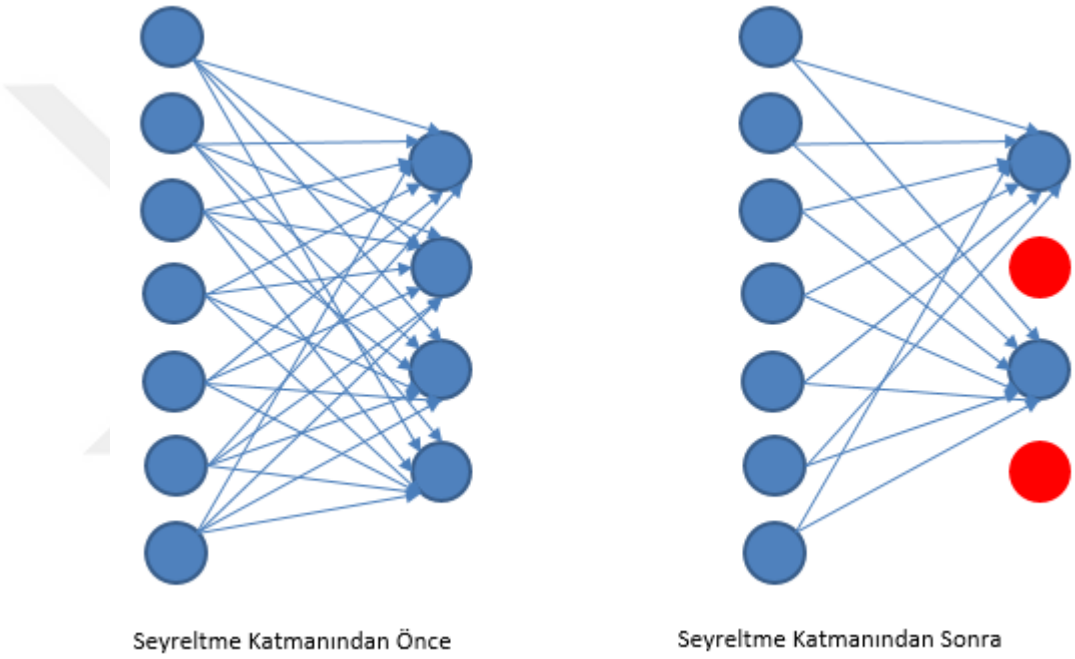
Tam bağlı katman (FC), her girişin tüm nöronlara bağlı olduğu bir giriş üzerinde çalışır. Eğer varsa, FC katmanları genellikle CNN mimarisinin sonuna doğru bulunur ve sınıf skorları gibi hedefleri optimize etmek için kullanılabilir (Amidi and Amidi 2018).



Şekil 3.10. Tam bağlantılı katman

3.2.1.f. Seyreltme katmanı (Dropout)

Seyreltme katmanı eğitim esnasında ağdan bazı düğümlerin atılması işlemidir. Yani her eğitim aşamasında, tek tek düğümler ya 1-p olasılıkla ağdan çıkarılır ya da p olasılıkla tutulur, böylece azaltılmış bir ağ kalır. Ayrılan bir düğüme gelen ve giden kenarlar da kaldırılır (Srivastava *et al.* 2014).



Şekil 3.11. Seyreltme katmanı

Seyreltme katmanı, aşırı öğrenmenin önüne geçmek için kullanılır. Eğer eğitim süresi çok uzun olursa, eğitim yapılacak veri sayısı çoksa veyahut eğitim yapılacak ağ çok derinse aşırı öğrenme gerçekleşebilir.

3.2.1.g. Yığın normalleştirme katmanı (Batch Normalization)

Yığın normalleştirme katmanı, her giriş kanalını mini-yığın boyunca normalleştirir. Konvolüsyonel sinir ağlarının eğitimini hızlandırmak ve ağ başlatma işlemine

duyarlılığı azaltmak için, konvolüsyon tabakaları ile ReLU tabakaları gibi doğrusal olmayan katmanlar arasında yığın normalleştirme kullanıyoruz.

Katman ilk önce mini-yığın ortalamasını çıkartarak ve mini-yığın standart sapmasına bölerek her kanalın aktivasyonunu normalleştirir. Daha sonra girişi öğrenilebilir bir dengeleyici β ile kaydırır ve öğrenilebilir bir ölçek faktörü γ ile ölçeklendirir. Yığın normalleştirme katmanları, sinir ağları boyunca yayılan aktivasyonları ve eğimleri normalleştirir ve ağ eğitimini daha kolay bir optimizasyon problemi haline getirir. Bu durumdan tam olarak yararlanmak için, öğrenme oranını arttırmayı deneyebilirsiniz. Optimizasyon problemi daha kolay olduğu için parametre güncellemeleri daha büyük olabilir ve ağ daha hızlı öğrenebilir.

Yığın normalleştirme, girişleri x_i'yi ilk önce bir mini-yığın üzerinden ve her giriş kanalı üzerinden ortalama μ_β ve σ_β^2 varyansını hesaplayarak normalleştirir. Sonra normalize hesaplar.

$$\hat{x}_i = \frac{x_i - \mu_\beta}{\sqrt{\sigma_\beta^2 + \varepsilon}} \tag{3.5}$$

Burada ε, mini-yığın değişkeni, çok küçük olduğunda sayısal dengeyi iyileştirir. Sıfır ortalama ve birim varyanslı girişlerin yığın normalleştirme katmanını izleyen katman için optimal olmadığından emin olmak için yığın normalleştirme katmanı aktivasyonları daha da kaydırır ve ölçeklendirir.

$$y_i = \gamma \hat{x}_i + \beta \tag{3.6}$$

β ve γ, ağ eğitimi sırasında güncellenen, kendi kendilerine öğrenebilen parametrelerdir.

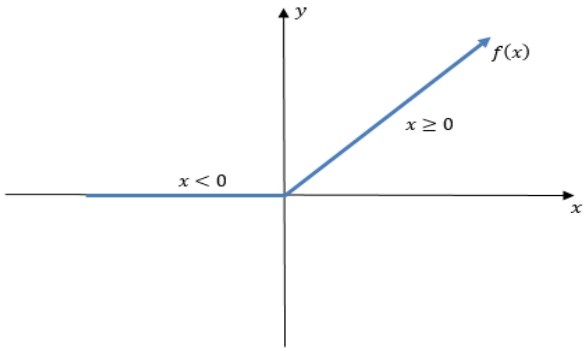
Ağ eğitimi sona erdiğinde, yığın normalleştirme katmanı tüm eğitim setindeki ortalamayı ve varyansı hesaplar. Matlab bunları TrainedMean ve TrainedVariance

dosyalarında saklar. Yeni görüntüler üzerinde tahminlerde bulunmak için eğitimli bir ağ kullandığınızda, katman, mini-yığın ortalama ve varyans yerine, eğitimli ortalama ve varyansı aktivasyonları normalleştirmek için kullanır (Matlab Documentation 2019).

3.2.1.h. Düzleştirilmiş doğrusal birim katman (ReLU)

Düzleştirilmiş doğrusal birim katmanında negatif olan giriş değeri sıfıra eşitlenir. Denklem (3.7)'de fonksiyon tanımı verilmiştir. Bu katman sayesinde ağ daha hızlı eğitilmektedir (Altuntaş and Naneli 2017).

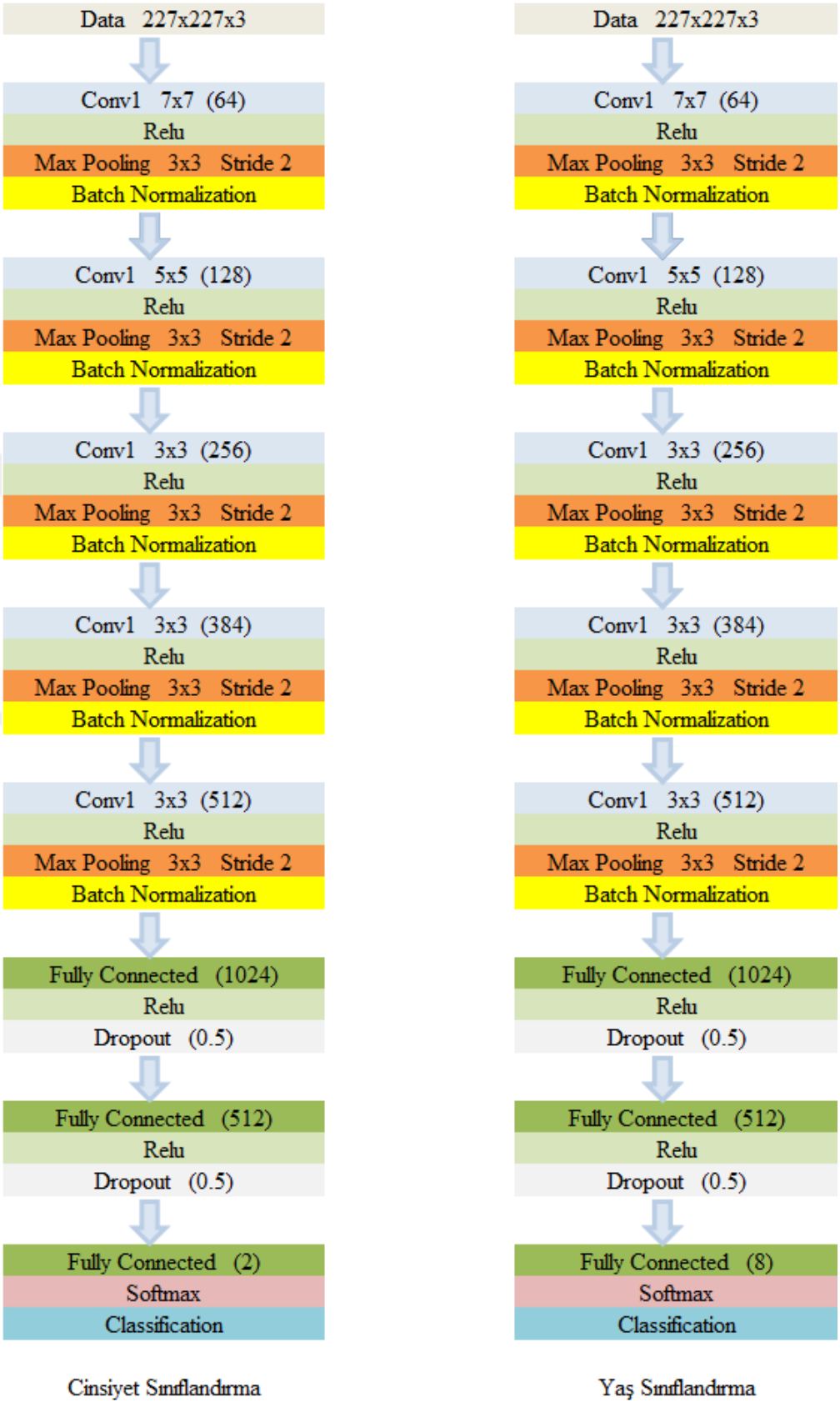
$$f(x) = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases} \tag{3.7}$$



Şekil 3.12. ReLU fonksiyonu

3.2.1.ı. Çalışmada kullandığımız CNN yapısı

Tezimizde üzerinde çalıştığımız CNN yapısını oluştururken (Levi and Hassner 2015)'nin kendi çalışmalarında kullandığı yapıyı temel alarak katman sayılarını arttırdık ve hiper parametrelerde değişlikler yaptık. Şekil 3.13' te hem cinsiyet sınıflandırma hem de yaş sınıflandırma için CNN ağımızın şeması verilmiştir.



Şekil 3.13. Cinsiyet ve yaş sınıflandırma için önerilen CNN ağı

Çalışma için hazırladığımız CNN ağında yaş ve cinsiyet sınıflandırmadaki ağ hemen hemen birbiriyle aynı. Sadece yaş sınıflandırmada son fully connected katmanı 8 iken cinsiyet sınıflandırmada bu katmanın değeri 2'dir. Bu sebeple ağlardan sadece birini anlatmak diğeri içinde yeterli olacaktır.

Ağımız 5 evrişimli katman ve 3 tam bağlantılı katmandan oluşmakta. Bu katmanlar arasında da maksimum havuzlama, düzleştirilmiş doğrusal birim katmanı, yığın normalleştirme gibi katmanlar kullanıldı. İşlem sırası ve katmanların özellikleri aşağıda belirtilmiştir:

- Girişte 256x256x3 boyutundaki resimlerden 227x227x3 boyutunda rastgele resimler kesilmiştir.
- İlk evrişimli katmanda 227x227x3 boyutuna çevrilmiş olan resimlere 64 tane 7x7x3 filtre uygulandı. Sonrasında ReLU katmanından geçen imgeler 2 adım kaydırmalı 3x3 boyutunda maksimum havuzlama katmanında işlem görmüştür. Ve daha sonra yığın normalleştirme ile maksimum havuzlamadan çıkan veriler normalize edilmiştir.
- İkinci evrişimli katmanda giriş verisinin boyutu 106x106x64 oldu artık. Bu veriye 5x5x64 boyutunda 128 tane filtre uygulanıp yine ReLU, 2 adım kaydırmalı 3x3 boyutunda maksimum havuzlama ve yığın normalleştirme katmanları uygulanır.
- Üçüncü evrişimli katmana gelen veriler artık 53x53x128 boyutundadır. 3x3x128 boyutunda 256 tane filtre uygulanıp yine ReLU, 2 adım kaydırmalı 3x3 boyutunda maksimum havuzlama ve yığın normalleştirme katmanları uygulanır.
- Dördüncü evrişimli katmana ulaşan veriler 25x25x256 boyutuna indirgenmiş oldu. 3x3x256 boyutunda 384 tane filtre uygulanıp yine ReLU, 2 adım kaydırmalı 3x3 boyutunda maksimum havuzlama ve yığın normalleştirme katmanları uygulanır.
- Beşinci evrişimli katmanda artık veriler 11x11x384 boyutuna ulaşmış oldu. 3x3x384 boyutunda 512 tane filtre uygulanıp yine ReLU, 2 adım kaydırmalı 3x3 boyutunda maksimum havuzlama ve yığın normalleştirme katmanları uygulanır.
- Beşinci katman sonunda 4x4x512 boyutundaki resimden 8192 tane parametre tam bağlantılı katmana girmektedir. Daha sonra düzleştirilmiş doğrusal birim katmanında ve 0,5 oranında seyreltme yapılan seyreltme katmanında veriler işlem görür.

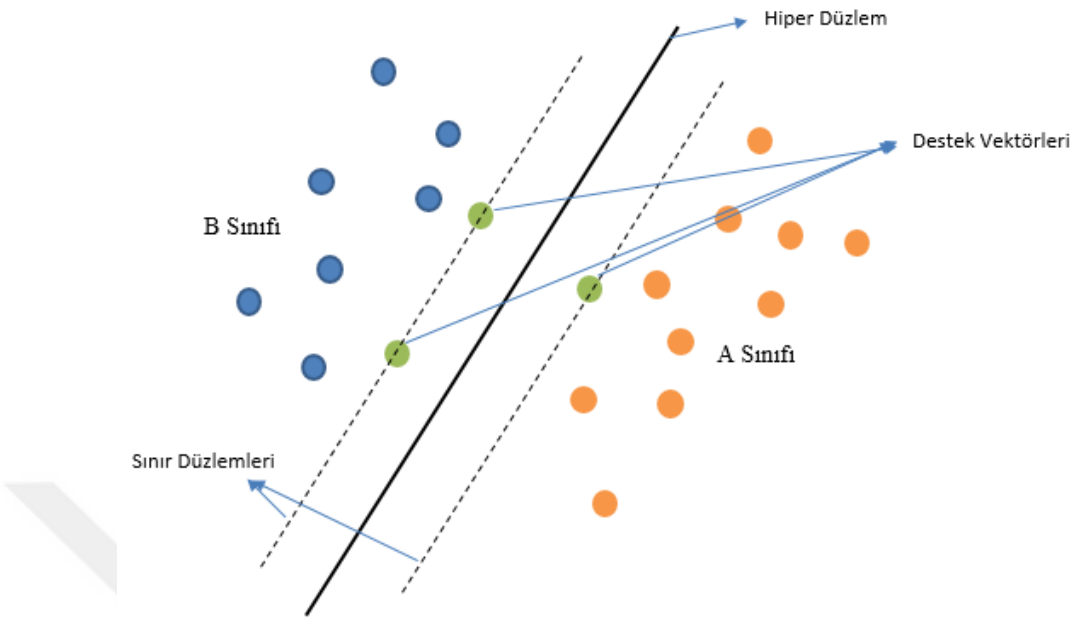
- Tam bağlantılı katmanın aynısını tekrarladıktan sonra bu kez bağlantı sayısı 2 olan (cinsiyet için) tam bağlantılı katmanın çıkışına softmax ve onunda çıkışına sınıflandırma katmanları bağlanarak eğitim işlemi için gerekli ağ hazırlanmış oldu.

Eğitimde kullanılan parametrelerden bizim için önemli olanları öğrenme katsayısı 0,001 mini-yığın boyutu 32 ve maksimum epoch sayısı da 150 olarak seçilmiştir. Eğitim ve doğrulama için kullanılan resimlerde boyut kesme işlemi rastgele olacak şekilde iken test için merkez noktasına göre kesme yapılmıştır. Sınıflandırma, katlardan biri test kalan dört katın da %20'si doğrulama, %80'i de eğitim olacak şekilde veri seti bölünmüştür. Her kat için ayrı ayrı testler yapılmış ve sonuçların ortalaması alınmıştır.

3.2.2. Destek vektör makineleri

Destek vektör makineleri imge tanıma ve sınıflandırma problemlerine çözüm olarak Vapnik tarafından geliştirilmiştir (Vapnik and Cortes 1995). Destek vektör makineleri, birçok sınıflandırma probleminin çözümünde başarıya ulaşmış, genelleme performansı yüksek ve etkin makine öğrenimi algoritmalarından biri olarak literatürdeki yerini almıştır. Destek vektör makineleri'nin en önemli avantajı, sınıflandırma problemini kareli optimizasyon problemine dönüştürüp çözmesidir. Böylece problemin çözümüne ilişkin öğrenme aşamasında işlem sayısı azalmakta ve diğer teknik/algoritmalara göre daha hızlı çözüme ulaşılmaktadır (Osowski *et al.* 2004). Bu özelliğinden dolayı, büyük hacimli data setlerinde büyük avantaj sağlamaktadır.

Destek vektör makinelerinde amaç, sınıfları birbirinden ayıracak optimal ayırma hiper düzleminin elde edilmesidir. Bir başka deyişle, farklı sınıflara ait destek vektörleri arasındaki uzaklığı maksimize etmektir. İki sınıflı ve çok sınıflı sınıflandırma probleminin çözümü için geliştirilmiş makine öğrenmesi algoritmalarıdır. Genelde iki sınıflı problemlerin çözümünde kullanılmaktadırlar.



Şekil 3.14. İki sınıflı sınıflandırma için destek vektör makinesi düzlemi

Kesikli çizgilerle gösterilmiş ve destek vektörlerinin üzerinde bulunduğu düzlemlere sınır düzlemleri denir. Sınır düzlemlerinin tam ortasından geçen ve her iki düzleme de eşit uzaklıkta bulunan düzlem ise hiper düzlem olarak ifade edilir (Ayhan ve Erdoğmuş 2014).

Destek vektör makineleri, veriyi daha yüksek bir boyuta dönüştürerek oluşturacağı bir hiper-düzlem ile iki sınıfı birbirinden ayırma prensibini esas alır. Yüksek boyuta dönüşüm aşamasında değişik özelliklere sahip fonksiyonlar kullanılır. Bu fonksiyonlara kernel fonksiyonları denir. Kernel fonksiyonlarının kullanımı için bu fonksiyonların matematiksel ifadesinde bulunan bazı parametrelerin kullanıcı tarafından belirlenmesi gerekir.

Polinom kerneli Denklem (3.8)'deki gibi ifade edilir. "d" polinom derecesidir.

$$K(x, y) = ((x.y) + 1)^d \tag{3.8}$$

Normalleştirme polinom kerneli Denklem (3.9)'da verilmiştir. "d" polinom derecesidir.

$$K(x,y) = \frac{((x.y)+1)^d}{\sqrt{((x.x)+1)^d((y.y)+1)^d}} \tag{3.9}$$

Radyal tabanlı fonksiyon kerneli denklemi Denklem (3.10)'da gösterildiği gibidir. "γ" kernel boyutudur.

$$K(x,y) = e^{-\gamma\|(x-x_i)\|^2} \tag{3.10}$$

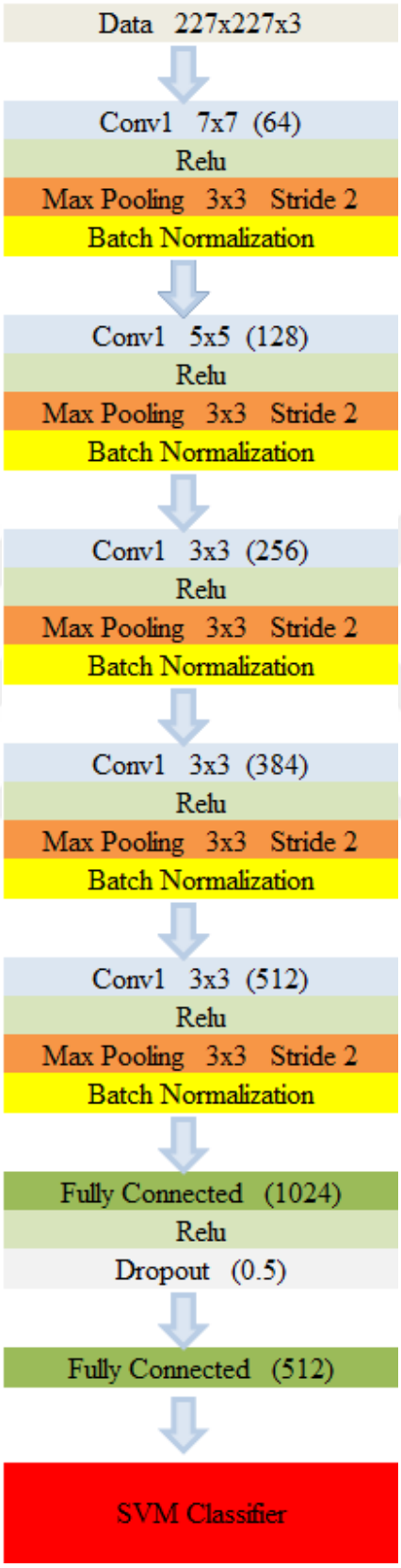
Radyal tabanlı fonksiyonlarda kernel boyutunun (γ) değişimlerinin sınıflandırma doğruluk oranına etkisi daha az iken polinomun fonksiyonlardaki kernel boyutunun artışı hem işlem süresinin artmasına neden olurken hem de bir noktadan sonra sınıflandırma performansını düşürmektedir (Hsu *et al.* 2016). Normalleştirilmiş polinomda fonksiyonu veri setinin normalleştirilmesi değil de polinom kernele ait matematiksel ifadenin normalleştirilmesi amaçlanmıştır (Arnulf and Borer 2001).

Evrişimli sinir ağı (CNN) + Destek vektör makineleri (SVM)

Evrişimli sinir ağlarından Bölüm 3.2.1'de destek vektör makinelerinden de Bölüm 3.2.2'de bahsetmiştik. Bu bölümde destek vektör makineleri ile evrişimli sinir ağlarını nasıl birleştirip birlikte çalıştırdığımızı anlatacağız.

3.2.2.a. Çalışmada kullandığımız CNN +SVM yapısı

Destek vektör makinelerinden kısaca bahsettikten sonra biz çalışmamızın neresinde ve nasıl kullandık gelelim bu soruları cevaplamaya. Tasarladığımız evrişimli sinir ağının sınıflandırma kısmını classification katmanında yapmıştık ve derin öğrenme kullanmıştık. Şekil 3.15'ten CNN+SVM ağımızın mimarisini görebilirsiniz.

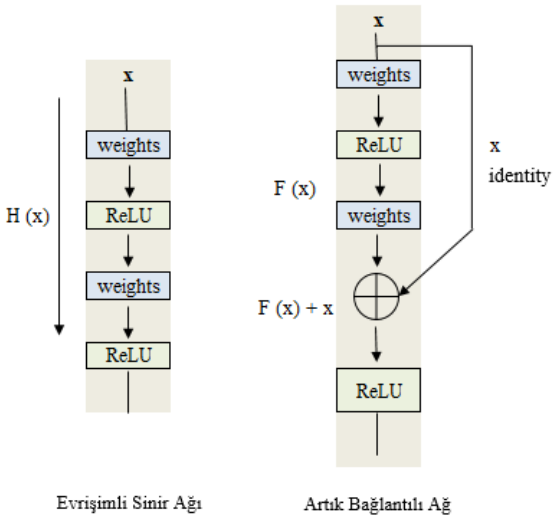


Şekil 3.15. Cinsiyet ve yaş sınıflandırma için önerilen CNN + SVM ağı

Bu çalışmada sınıflandırma işini destek vektör makinelerine bırakıyoruz. Yani ağımız CNN ile eğitiliyor ve fc_2 katmanına kadar derin öğrenme kullanılıyor. fc_2 katmanından sonra destek vektör makineleri devreye girerek özellik çıkarımı ve ardından sınıflandırma yapmaktadır. fc_2'ye kadar olan derin öğrenme mimarisi düz CNN mimarimiz ile aynıdır. Bu sebeple ayrıyeten bahsedilmeyecektir. SVM tarafı için yaptıklarımızdan bahsedecek olursak; SVM sınıflandırıcı için bazı temel kernel fonksiyonlar mevcut. Bizim ağımızda kernel fonksiyon olarak radyal temelli kernel fonksiyon seçilmiştir. Kernel ölçek parametresi "auto" olarak girilmiştir. Eğitim ve test sırasında kullanılacak verilerin özellik çıkarımlarında, mini-yığın boyutu da 32 olarak testler yapılmıştır.

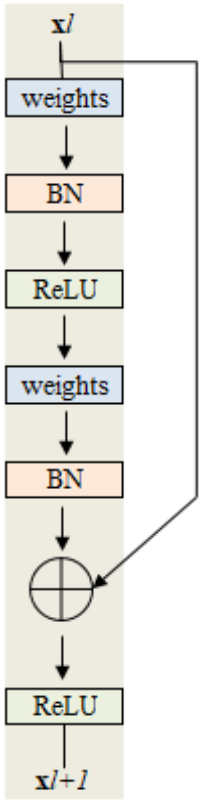
3.2.3. Residual of Residual (RoR) Network

Standart bir CNN yapısında girişten çıkışa kadar olan kısmı nonlinear bir H(x) fonksiyonu ile haritalandırabiliyoruz. Residual network mimarisinde bu yol, H(x) yerine F(x) = H(x) — x olarak tanımlanan başka bir nonlinear fonksiyon ile haritalandırılıyor. Buna ek olarak da girişten çıkışa bir atlama bağlantısı yapılarak, x (giriş) değeri F(x) fonksiyonuna aritmetik olarak ekleniyor. Daha sonra F(x)+x fonksiyonu ReLU katmanından geçiriliyor. Dolayısıyla 2. katmanın sonuna giriş değeri eklenerek, geçmiş katmanlardaki değerlerin ileriki katmanlara daha güçlü bir şekilde iletilmesi hedefleniyor.



Şekil 3.16. Residual network yapısı

Derin artık ağlar (ResNets) kümelenmiş birçok "Artık Birim" den oluşur. Şekil 3.17'de gösterilen artık birimlerin en genel ifadesi Denklem (3.11)'dedir (He *et al.* 2016).

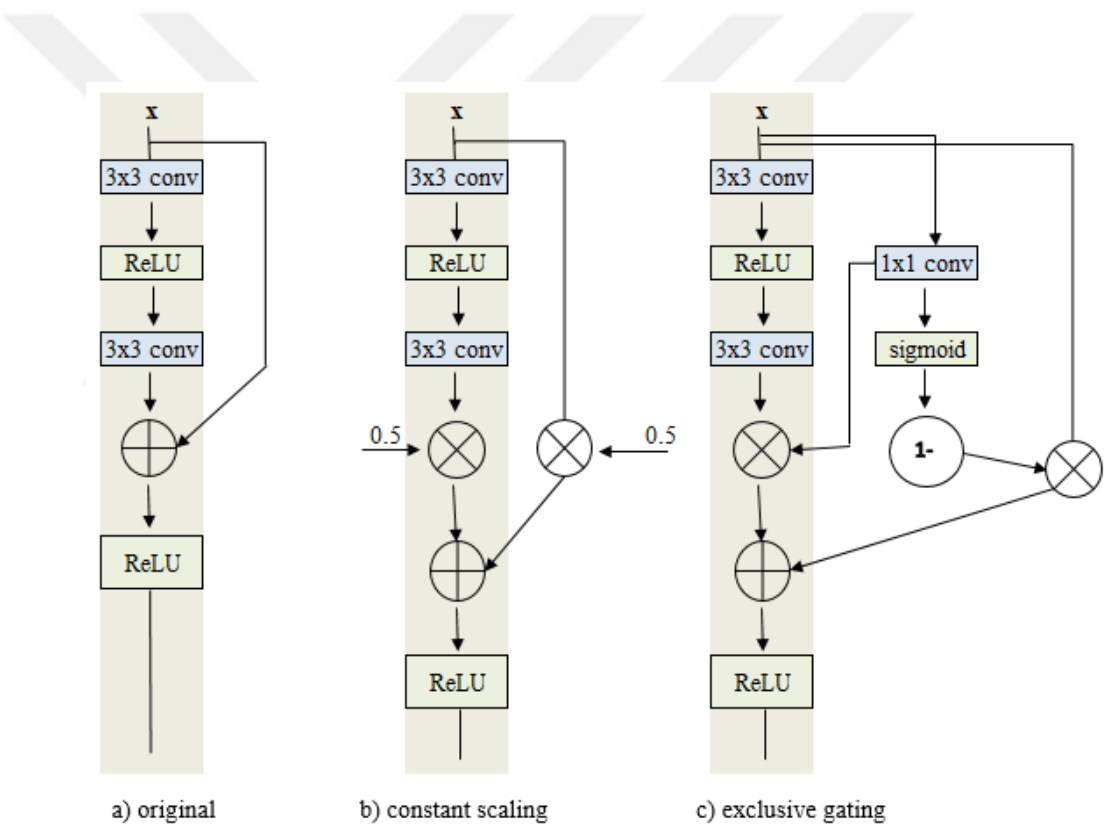


Şekil 3.17. Artık birim genel görünümü

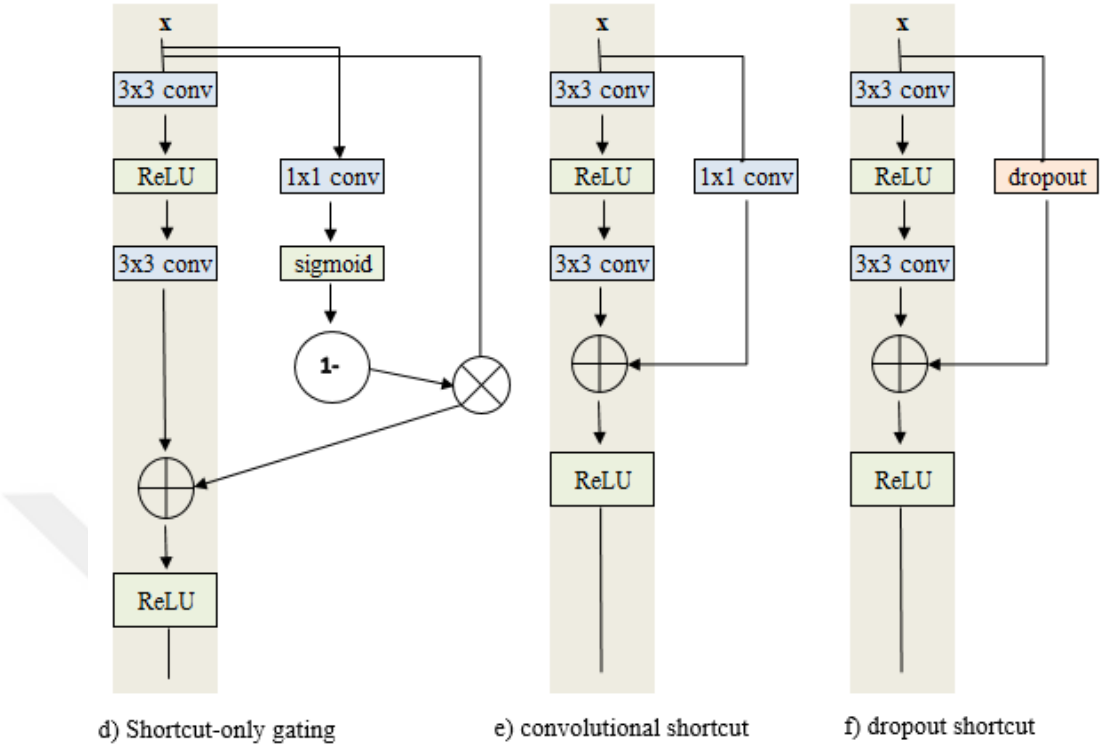
$$y_l = h(x_l) + F(x_l, w_l)$$
$$x_{l+1} = f(y_l) \tag{3.11}$$

x_l : l'ninci birimin girişi , x_{l+1} : l'ninci birimin çıkışı, F : Artık fonksiyon

Artık ağda giriş değerini fonksiyona sokup çıkış değerine ekleme işleminin atlama bağlantıları ile gerçekleştirildiğinden bahsetmiştik. Şimdi bu atlama bağlantı çeşitlerine ve ne işe yaradıklarına bir bakalım. Şekil 3.18 ve Şekil 3.19'da atlama bağlantı metotları gösterilmiştir.



Şekil 3.18. Atlama bağlantı metotları-1



Şekil 3.19. Atlama bağlantı metotları-2

a) Orijinal: Atlama bağlantılarının orijinal hali girişin herhangi bir işlem uygulanmadan çıkışa aktarılmasıdır.

b) Sabit Ölçeklendirme (Constant scaling): λ değeri kadar ölçeklendirip atlama bağlantı sinyalini azaltarak girişi çıkışa ekleme işlemidir. Bu işlem optimizasyonun zorluk yaşamasına neden olmaktadır.

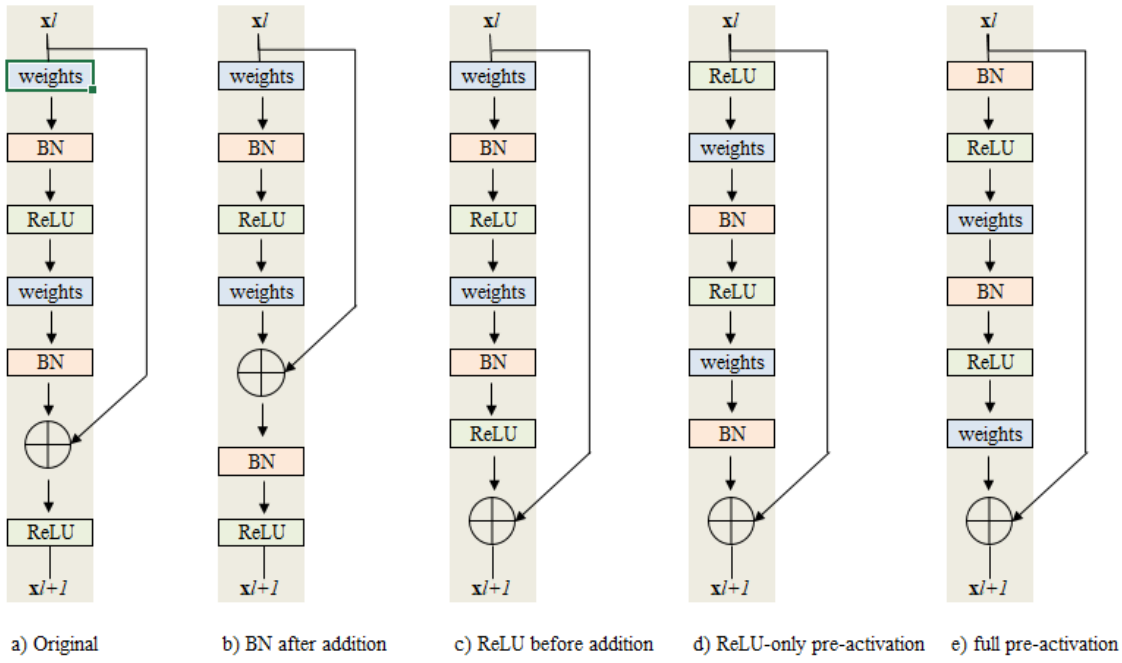
c) Özel Sinyal Ayırıcı (Exclusive gating): Özel sinyal ayırıcıda F (artık fonksiyon) g(x) ile atlama bağlantısı da 1-g(x) ile ölçeklendirilir. 1-g(x) 1'e yaklaştığında, sinyal ayırıcılı atlama bağlantıları, bilginin yayılmasına yardımcı olan kimliğe daha yakındır; ancak bu durumda g(x) 0'a yaklaşır ve F (artık) fonksiyonunu bastırır. Sinyal ayırıcılı fonksiyonların sadece atlama yolu üzerindeki etkilerini izole etmek için, özel olmayan bir sinyal ayırıcı yöntemi geliştirilmiştir.

d) Sadece Kısayol Sinyal Ayırıcısı (Shortcut-only gating): Bu yapıda F (artık) fonksiyonu ölçeklendirilmez. Sadece atlama bağlantısı 1-g(x) ile ölçeklendirilir.

e) Evrişimli Kısayol (convolutional shortcut): Bu atlama bağlantısı yöntemi daha çok giriş boyutu ile çıkış boyutu birbirinden farklı olduğu durumlarda giriş boyutunu çıkışa ekleyebilmek için boyut değiştirme işlemlerinde kullanılır.

f) Dropout Kısayolu (Dropout shortcut): Atlama bağlantısı girişteki verileri λ değeri oranı kadar atar ve kalan verileri çıkışa ekler. Ancak bu atlama bağlantı yöntemi çok başarılı sonuçlar vermemektedir.

Artık ağlarda, seçtiğimiz atlama bağlantı türü kadar önemli bir diğer konuda aktivasyon fonksiyonlarının farklı kullanım şekilleridir. Şekil 3.20'de aktivasyon fonksiyonlarının farklı kullanımları gösterilmektedir.



Şekil 3.20. Aktivasyon fonksiyonlarının farklı kullanım şekilleri

a) Orijinal: Aktivasyon fonksiyonların kullanımının orijinal hali yukarıdaki şekilde de görüldüğü üzere weight – BN – ReLU – weight – BN – addition - ReLU şeklinde gitmekte ve son BN den sonra toplama işlemi yapılmaktadır.

b) Toplamdan Sonra BN (BN after addition): Sondaki batch normalization işleminin toplamdan sonra yapıldığı şekildir. Bu yöntemde eğitim başlangıcındaki

eğitim kaybını azaltmadaki zorlukların bir yansıması olarak, BN katmanı, atlama bağlantısından geçen ve bilgi yayılımını engelleyen sinyali değiştirir.

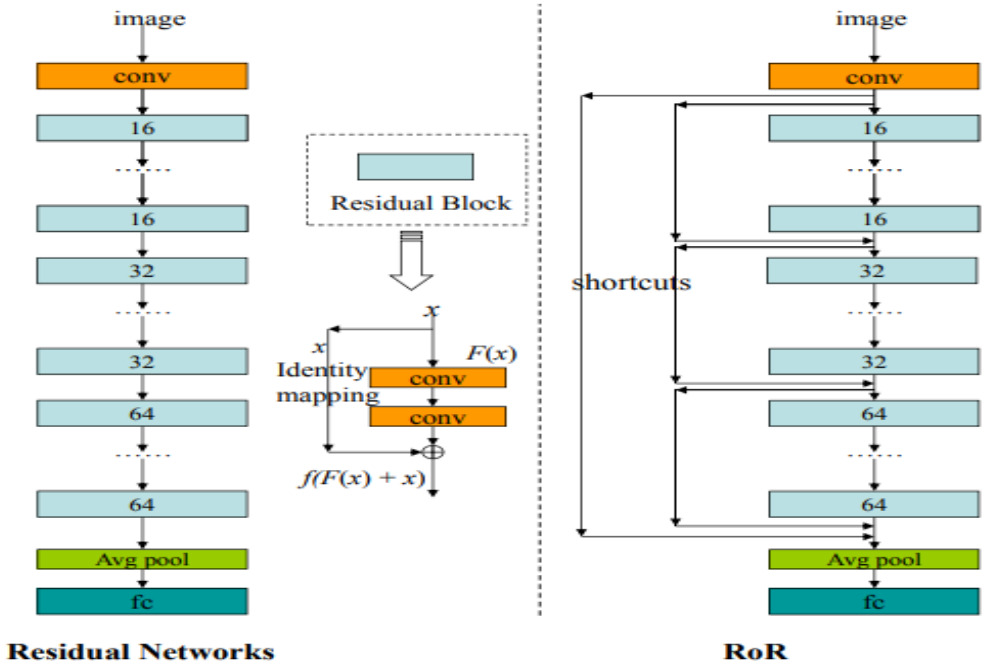
c) Toplamdan Önce ReLU (ReLU before addition): Orijinal kullanımda toplamdan sonraki ReLU katmanının toplamdan önce işleme koyulduğu yöntemdir. Bu yöntemde ileri yayılan sinyal monoton bir şekilde artmaktadır.

d) Sadece ReLU Ön Aktivasyon (ReLU-only pre-activation): Orijinal kullanımda toplamdan sonraki ReLU katmanının ilk weight katmanından önce işleme sokulmasıdır.

e) Tam Ön Aktivasyon (Full pre-activation): Tam ön aktivasyonda ise hem ReLU hem de batch normalization katmanları weight katmanından önce işleme tabi olur.

RoR, artık ağların optimizasyon yeteneğini derinleştirmek için, artık haritalamanın artık haritalamasını optimize etmeyi hedefler. Bu yüzden, artık ağlara dayalı RoR oluşturmak için kısayolları seviye seviye eklemişlerdir (Zhang *et al.* 2016).

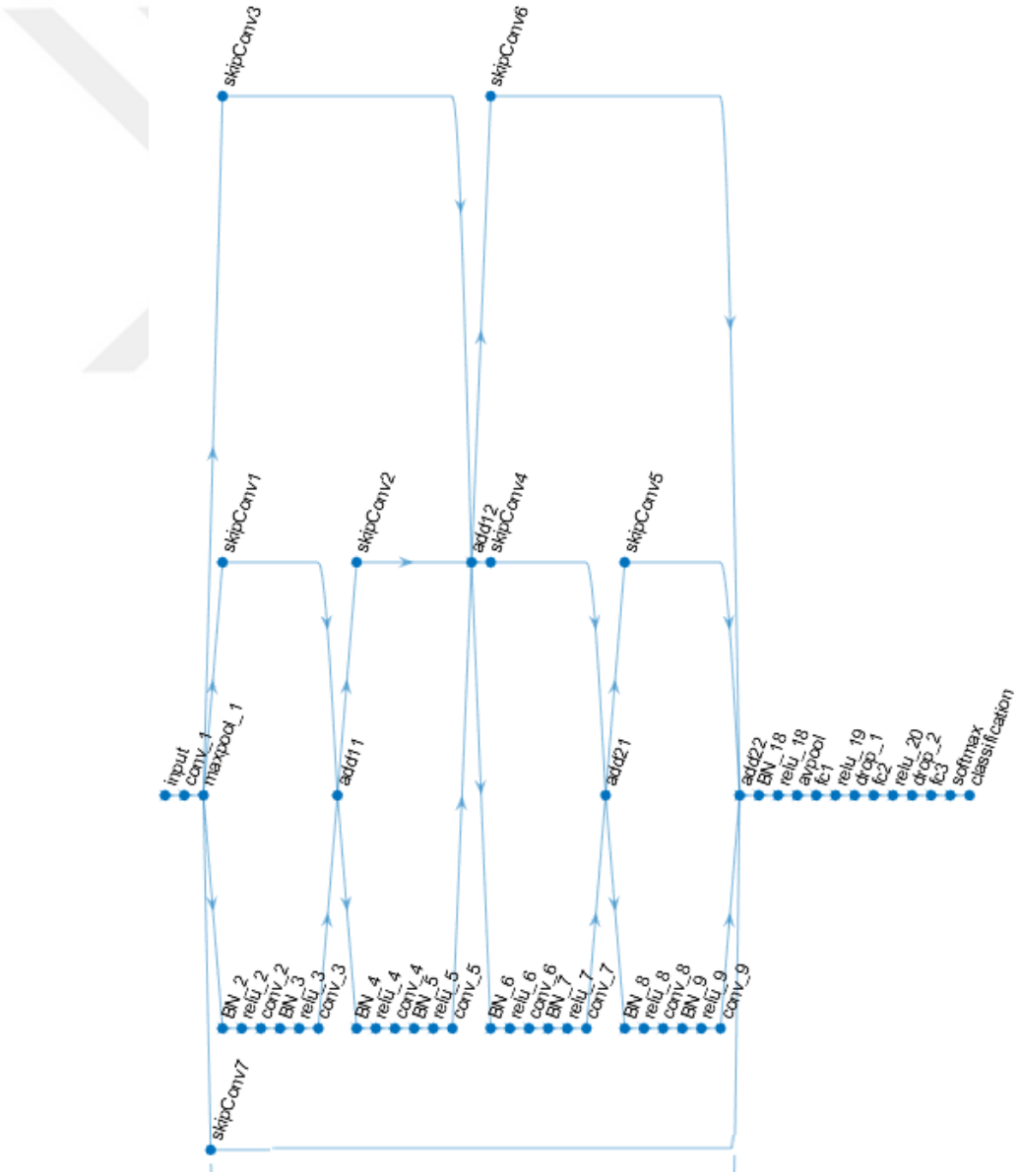
Residual network ile RoR arasındaki ilişkiyi Şekil 3.21'den görebilirsiniz.



Şekil 3.21. Residual Network ve RoR arasındaki ilişki (Zhang *et al.* 2016)

3.2.3.a. Çalışmada kullandığımız Residual of Residual ağ yapısı

Çalışmamızda kullandığımız Pre-RoR ağ yapısının temeli (ZHANG, ve diğerleri, 2017) tarafından hazırlanan makalede belirtilen ağ yapısıdır. Biz bazı hiper parametrelerde değişiklik yaparak Matlab ortamında Pre-RoR ağını oluşturduk. Bu ağ Matlab'da bulunan hazır ağlar içerisinde bulunmamaktadır. Şekil 3.22'de oluşturduğumuz ağ verilmiştir.



Şekil 3.22. Önerilen Pre-RoR ağ yapısı

RoR ağ mimarimizde 4 seviyemiz vardır. Şekil 3.22'de belirgin olarak gözükebilsin diye sadece 2 seviye gösterilmiştir. Bu seviyelerin her birinin içerisinde de 4'er adet conv katmanı bulunmaktadır. Bu convlar önce ikili şekilde seviyelerin içerisinde toplanıyor daha sonra da seviye girişleri toplamlara ekleniyor ve en sonunda da girişe uygulanan conv_1 - pool_1 – BN_1 – relu_1 katmanının çıkışı toplama eklenmiştir. Genel olarak yapılan işlem anlatıldıktan sonra katman katman neler yapılmış onlara değinelim.

- Girişe 227x227x3 boyutunda veriler verilmiştir.
- 227x227x3 boyutundaki imgelere ilk olarak 3x3 boyutunda, 1 adım kaydırmalı eklemesi olmayan 16 adet filtre uygulandı ve yeni boyutu 225x225x16 yapıldı. Devamında 3x3 boyutunda 2 adım kaydırmalı havuzlama katmanı uygulanmıştır. Havuzlamanın çıkışında resmin boyutu 112x112x16 olmuştur. Bu noktaya pool_1 adını verelim. İleride toplama katmanında kullanacağız.
- pool_1 den sonra yığın normalleştirme (BN), düzleştirilmiş doğrusal birim katmanı (ReLU)ve 7x7 boyutunda 2 adım kaydırmalı 16 filtreli evrişim katmanı uygulanmıştır. Evrişim katmanı padding same [2] olarak seçilmiştir. Bu durumda çıkıştaki imge 56x56x16 boyutunda olacaktır. Bu evrişim katmanına da conv_2 adını verdik.
- conv_2 çıkışına BN, ReLU ve conv_3 (7x7, 16 adet, Stride 1, padding same) uyguladık. conv_3 çıkışında boyut 56x56x16 oldu..
- add11 adını verdiğimiz toplama katmanında conv_3 ve pool_1'i toplayacağız. Toplama işlemini yapabilmemiz için boyutların aynı olması gerekiyor. conv_3 56x56x16, pool_1 ise 112x112x16 boyutunda. Bu haliyle toplama işlemini yapamayız. İşte bu noktada "skipconv" adını verdiğimiz atlama evrişimleri yardımıyla boyutları eşitleyebiliriz. Skipconv da aslında bir evrişim işlemidir. İşlemsel olarak diğer evrişimlerden farkı yoktur. pool_1'in çıkışına skipconv_1 (1x1, 16 adet, stride 2, padding same) uygulayarak boyut 56x56x16'ya düşürülmüştür. Böylece add11'de conv_3 ve skipconv_1 yardımıyla pool_1'in toplama işlemi yapılabilmiştir. Bir

[2] Matlab'da "padding same" seçilince girişteki resmin boyutu 1 adım kaydırmalı evrişim için çıkışa değişmeden aktarılır. Adım kaydırma 2 seçilirse $\text{Çıkış } Boyutu = \frac{\text{Giriş } Boyutu}{2}$ olur.

katmanda yapılan işlemleri detaylı bir şekilde anlattıktan sonra bütün mimaride uygulanan işlemleri şöyle özetleyebiliriz.

Çizelge 3.3. Önerilen Pre-RoR ağ katmanları (Matlab Komutları)

Giriş Boyutu	Yapılan İşlem	Çıkış Boyutu
227x227x3	convolution2dLayer(3,16,'Stride',1,'Name','conv_1')	225x225x16
225x225x16	maxPooling2dLayer(3,'Stride',2,'Name','maxpool_1')	112x112x16
	Level 1	
	batchNormalizationLayer('Name','BN_2')	
	reluLayer('Name','relu_2')	
112x112x16	convolution2dLayer(7,16,'Padding','same','Stride',2,'Name','conv_2')	56x56x16
	batchNormalizationLayer('Name','BN_3')	
	reluLayer('Name','relu_3')	
56x56x16	convolution2dLayer(7,16,'Padding','same','Stride',1,'Name','conv_3')	56x56x16
56x56x16	additionLayer(2,'Name','add11')	56x56x16
	batchNormalizationLayer('Name','BN_4')	
	reluLayer('Name','relu_4')	
56x56x16	convolution2dLayer(7,16,'Padding','same','Stride',1,'Name','conv_4')	56x56x16
	batchNormalizationLayer('Name','BN_5')	
	reluLayer('Name','relu_5')	
56x56x16	convolution2dLayer(7,16,'Padding','same','Stride',1,'Name','conv_5')	56x56x16
56x56x16	additionLayer(3,'Name','add12')	56x56x16
	Level 2	
	batchNormalizationLayer('Name','BN_6')	
	reluLayer('Name','relu_6')	
56x56x16	convolution2dLayer(5,32,'Padding','same','Stride',2,'Name','conv_6')	28x28x32
	batchNormalizationLayer('Name','BN_7')	
	reluLayer('Name','relu_7')	
28x28x32	convolution2dLayer(5,32,'Padding','same','Stride',1,'Name','conv_7')	28x28x32
28x28x32	additionLayer(2,'Name','add21')	28x28x32
	batchNormalizationLayer('Name','BN_8')	
	reluLayer('Name','relu_8')	
28x28x32	convolution2dLayer(5,32,'Padding','same','Stride',1,'Name','conv_8')	28x28x32
	batchNormalizationLayer('Name','BN_9')	
	reluLayer('Name','relu_9')	
28x28x32	convolution2dLayer(5,32,'Padding','same','Stride',1,'Name','conv_9')	28x28x32
28x28x32	additionLayer(3,'Name','add22')	28x28x32
	Level 3	
	batchNormalizationLayer('Name','BN_10')	
	reluLayer('Name','relu_10')	
28x28x32	convolution2dLayer(3,64,'Padding','same','Stride',2,'Name','conv_10')	14x14x64
	batchNormalizationLayer('Name','BN_11')	
	reluLayer('Name','relu_11')	
14x14x64	convolution2dLayer(3,64,'Padding','same','Stride',1,'Name','conv_11')	14x14x64
14x14x64	additionLayer(2,'Name','add31')	14x14x64

Çizelge 3.3. (devam)

	batchNormalizationLayer('Name','BN_12')	
	reluLayer('Name','relu_12')	
14x14x64	convolution2dLayer(3,64,'Padding','same','Stride',1,'Name','conv_12')	14x14x64
	batchNormalizationLayer('Name','BN_13')	
	reluLayer('Name','relu_13')	
14x14x64	convolution2dLayer(3,64,'Padding','same','Stride',1,'Name','conv_13')	14x14x64
14x14x64	additionLayer(3,'Name','add32')	14x14x64
	Level 4	
	batchNormalizationLayer('Name','BN_14')	
	reluLayer('Name','relu_14')	
14x14x64	convolution2dLayer(3,128,'Padding','same','Stride',2,'Name','conv_14')	7x7x128
	batchNormalizationLayer('Name','BN_15')	
	reluLayer('Name','relu_15')	
7x7x128	convolution2dLayer(3,128,'Padding','same','Stride',1,'Name','conv_15')	7x7x128
7x7x128	additionLayer(2,'Name','add41')	7x7x128
	batchNormalizationLayer('Name','BN_16')	
	reluLayer('Name','relu_16')	
7x7x128	convolution2dLayer(3,128,'Padding','same','Stride',1,'Name','conv_16')	7x7x128
	batchNormalizationLayer('Name','BN_17')	
	reluLayer('Name','relu_17')	
7x7x128	convolution2dLayer(3,128,'Padding','same','Stride',1,'Name','conv_17')	7x7x128
7x7x128	additionLayer(4,'Name','add42')	7x7x128
	batchNormalizationLayer('Name','BN_18')	
	reluLayer('Name','relu_18')	
7x7x128	averagePooling2dLayer(3,'Padding','same','Stride',1,'Name','avpool')	7x7x128
6272	fullyConnectedLayer(1024,'Name','fc1')	1024
	reluLayer('Name','relu_19')	
	dropoutLayer(0.5,'Name','drop_1')	
1024	fullyConnectedLayer(512,'Name','fc2')	512
	reluLayer('Name','relu_20')	
	dropoutLayer(0.5,'Name','drop_2')	
512	fullyConnectedLayer(2,'Name','fc3')	2
	softmaxLayer('Name','softmax')	
	classificationLayer('Name','classification')];	

Çizelge 3.3'te resmin katmana girmeden önceki boyutu ile katmandan sonraki boyutu verilmiştir. Önce yukarıdaki ağ yapısını oluşturduk. Bu ağdaki katmanları aşağıdaki komut ile lgraph'a[3] dönüştürdük.

```
lgraph = layerGraph(layers)
```

Daha sonra girişteki boyutları farklı olan imgeleri toplama katmanında işleme dahil edebilmek için boyutlarını değiştirmede kullandığımız skipconv evrişimlerini oluşturduk.

Çizelge 3.4. Skipconv yapısı (Matlab Komutları)

Giriş Boyutu	Yapılan İşlem	Çıkış Boyutu
112x112x16	skipConv1 = convolution2dLayer(1,16,'Padding','same','Stride',2,'Name','skipConv1');	56x56x16
56x56x16	skipConv2 = convolution2dLayer(1,16,'Padding','same','Stride',1,'Name','skipConv2');	56x56x16
112x112x16	skipConv3 = convolution2dLayer(1,16,'Padding','same','Stride',2,'Name','skipConv3');	56x56x16
56x56x16	skipConv4 = convolution2dLayer(1,32,'Padding','same','Stride',2,'Name','skipConv4');	28x28x32
28x28x32	skipConv5 = convolution2dLayer(1,32,'Padding','same','Stride',1,'Name','skipConv5');	28x28x32
56x56x16	skipConv6 = convolution2dLayer(1,32,'Padding','same','Stride',2,'Name','skipConv6');	28x28x32
28x28x32	skipConv7 = convolution2dLayer(1,64,'Padding','same','Stride',2,'Name','skipConv7');	14x14x64
14x14x64	skipConv8 = convolution2dLayer(1,64,'Padding','same','Stride',1,'Name','skipConv8');	14x14x64
28x28x32	skipConv9 = convolution2dLayer(1,64,'Padding','same','Stride',2,'Name','skipConv9');	14x14x64
14x14x64	skipConv10 = convolution2dLayer(1,128,'Padding','same','Stride',2,'Name','skipConv10');	7x7x128
7x7x128	skipConv11 = convolution2dLayer(1,128,'Padding','same','Stride',1,'Name','skipConv11');	7x7x128
14x14x64	skipConv12 = convolution2dLayer(1,128,'Padding','same','Stride',2,'Name','skipConv12');	7x7x128
112x112x16	skipConv13 = convolution2dLayer(1,128,'Padding','same','Stride',16,'Name','skipConv13');	7x7x128

[3] lgraph, katmanları eklemek, çıkarmak, birleştirmek, bağlantıları kesmek için kullandığımız karmaşık bir katman grafik yapısına sahip olan derin öğrenme ağının mimarisini belirtir

Tanımladığımız skipconv'ları oluşturduğumuz lgraph içine ekliyoruz.

```
lgraph = addLayers(lgraph,skipConv1);
lgraph = addLayers(lgraph,skipConv2);
lgraph = addLayers(lgraph,skipConv3);
lgraph = addLayers(lgraph,skipConv4);
lgraph = addLayers(lgraph,skipConv5);
lgraph = addLayers(lgraph,skipConv6);
lgraph = addLayers(lgraph,skipConv7);
lgraph = addLayers(lgraph,skipConv8);
lgraph = addLayers(lgraph,skipConv9);
lgraph = addLayers(lgraph,skipConv10);
lgraph = addLayers(lgraph,skipConv11);
lgraph = addLayers(lgraph,skipConv12);
lgraph = addLayers(lgraph,skipConv13);
```

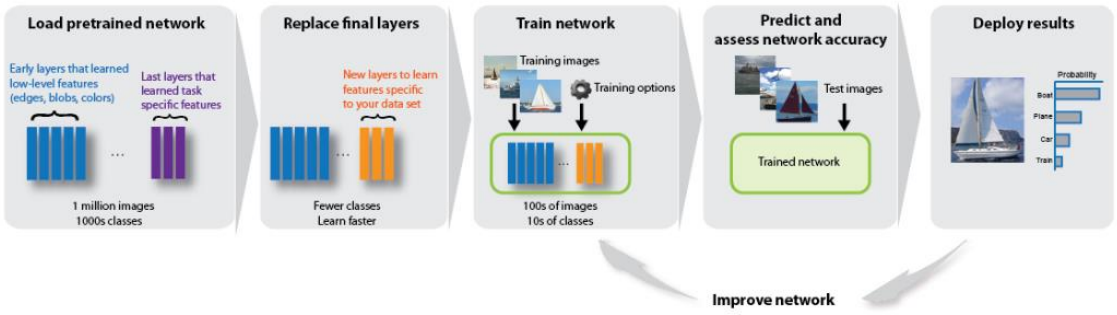
Sonrasında skipconvlar ile diğer katmanlar arasında bağlantıları oluşturuyoruz ve toplama katmanlarında toplama dahil ediyoruz.

```
Level 1 connections
lgraph = connectLayers(lgraph,'maxpool_1','skipConv1');
lgraph = connectLayers(lgraph,'skipConv1','add11/in2');
lgraph = connectLayers(lgraph,'add11','skipConv2');
lgraph = connectLayers(lgraph,'skipConv2','add12/in2');
lgraph = connectLayers(lgraph,'maxpool_1','skipConv3');
lgraph = connectLayers(lgraph,'skipConv3','add12/in3');

Level 2 connections
lgraph = connectLayers(lgraph,'add12','skipConv4');
lgraph = connectLayers(lgraph,'skipConv4','add21/in2');
lgraph = connectLayers(lgraph,'add21','skipConv5');
lgraph = connectLayers(lgraph,'skipConv5','add22/in2');
lgraph = connectLayers(lgraph,'add12','skipConv6');
lgraph = connectLayers(lgraph,'skipConv6','add22/in3');

Level 3 connections
lgraph = connectLayers(lgraph,'add22','skipConv7');
lgraph = connectLayers(lgraph,'skipConv7','add31/in2');
lgraph = connectLayers(lgraph,'add31','skipConv8');
```

```
lgraph = connectLayers(lgraph,'skipConv8','add32/in2');
lgraph = connectLayers(lgraph,'add22','skipConv9');
lgraph = connectLayers(lgraph,'skipConv9','add32/in3');

Level 4 connections
lgraph = connectLayers(lgraph,'add32','skipConv10');
lgraph = connectLayers(lgraph,'skipConv10','add41/in2');
lgraph = connectLayers(lgraph,'add41','skipConv11');
lgraph = connectLayers(lgraph,'skipConv11','add42/in2');
lgraph = connectLayers(lgraph,'add32','skipConv12');
lgraph = connectLayers(lgraph,'skipConv12','add42/in3');
lgraph = connectLayers(lgraph,'maxpool_1','skipConv13');
lgraph = connectLayers(lgraph,'skipConv13','add42/in4');
```

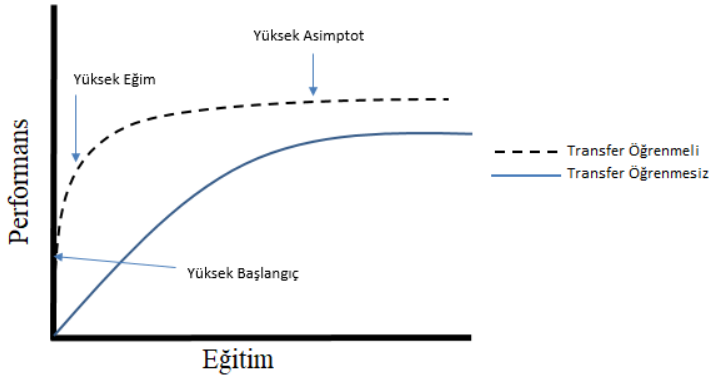
3.2.4. Transfer öğrenme (Transfer Learning)

Yaş tahmini ve cinsiyet sınıflandırma için önerdiğimiz yöntemlerden bir diğeri de transfer öğrenmesidir. Transfer öğrenmesi, daha önceden eğitilmiş bir ağın yeni çalışmalarda başlangıç noktası olarak kullanılmasıdır. Bir ağa transfer öğrenmesi ile ince ayar yapmak ağırlıkları sıfırdan rastgele olarak başlatılan bir ağın eğitimini yapmaktan daha hızlı ve daha kolaydır. Daha az sayıda eğitim görüntüsü kullanarak öğrenilmiş özellikler hızla yeni bir işe aktarılabilir.



Şekil 3.23. Transfer öğrenmesi (Beale *et al.* 2018)

Transfer öğrenmesi daha yüksek başlangıç seviyesi, daha yüksek eğim ve daha yüksek asimptot sağlar (Torrey and Shavlik 2009).



Şekil 3.24. Transfer öğrenmesinin avantajları

Matlab da transfer öğrenmesinde kullanılabilecek bazı önceden eğitilmiş modeller aşağıdaki gibidir. Bu modeller kullanılarak transfer öğrenmesi yapılabilir.

- NASNet-Large : ImageNet veritabanından bir milyondan fazla resim ile eğitilmiş bir evrişimli sinir ağıdır. Ağ, klavye, fare, kurşun kalem ve birçok hayvan gibi 1000 nesne kategorisinde görüntüleri sınıflandırabilir. 331x331 giriş boyutuna sahip görüntülerden oluşur.
- ShuffleNet : ImageNet veritabanı ile eğitilmiştir. 224x224 giriş boyutuna sahip görüntülerden oluşur.
- Places365GoogleNet : 22 katman derinliğe sahip önceden tanımlanmış bir evrişimsel sinir ağıdır. ImageNet veya Places365 veri setlerinde eğitilmiş bir ağ yükleyebilirsiniz. Places365'te eğitilen ağ, ImageNet'te eğitilen ağa benzer, ancak görüntüleri alan, park, pist ve lobi gibi 365 farklı yer kategorisinde sınıflandırır. Bu ağlar, çok çeşitli görüntüler için farklı özellik gösterimleri öğrendi. Ağların her ikisinde de görüntü 224x224 giriş boyutuna sahiptir.
- MobileNet-v2 : ImageNet veritabanı ile eğitilmiştir. 224x224 giriş boyutuna sahip görüntülerden oluşur ve ağ 54 katman derinliğe sahiptir.
- Xception : ImageNet veritabanı ile eğitilmiştir. 299x299 giriş boyutuna sahip görüntülerden oluşur ve ağ 71 katman derinliğe sahiptir.
- DenseNet-201 : ImageNet veritabanı ile eğitilmiştir. 224x224 giriş boyutuna sahip görüntülerden oluşur ve ağ 201 katman derinliğe sahiptir.

- SqueezeNet : ImageNet veritabanı ile eğitilmiştir. 227x227 giriş boyutuna sahip görüntülerden oluşur ve ağ 18 katman derinliğe sahiptir.
- ResNet-18 : ImageNet veritabanı ile eğitilmiştir. 224x224 giriş boyutuna sahip görüntülerden oluşur ve ağ 18 katman derinliğe sahiptir.
- ResNet-50 : ImageNet veritabanı ile eğitilmiştir. 224x224 giriş boyutuna sahip görüntülerden oluşur ve ağ 50 katman derinliğe sahiptir.
- ResNet-101 : ImageNet veritabanı ile eğitilmiştir. 224x224 giriş boyutuna sahip görüntülerden oluşur ve ağ 101 katman derinliğe sahiptir.
- Inception-v3 : ImageNet veritabanı ile eğitilmiştir. 299x299 giriş boyutuna sahip görüntülerden oluşur ve ağ 48 katman derinliğe sahiptir.
- Inception-ResNet-v2 : ImageNet veritabanı ile eğitilmiştir. 299x299 giriş boyutuna sahip görüntülerden oluşur ve ağ 164 katman derinliğe sahiptir.
- VGG-16 : ImageNet veritabanı ile eğitilmiştir. 224x224 giriş boyutuna sahip görüntülerden oluşur ve ağ 16 katman derinliğe sahiptir.
- VGG-19 : ImageNet veritabanı ile eğitilmiştir. 224x224 giriş boyutuna sahip görüntülerden oluşur ve ağ 19 katman derinliğe sahiptir.

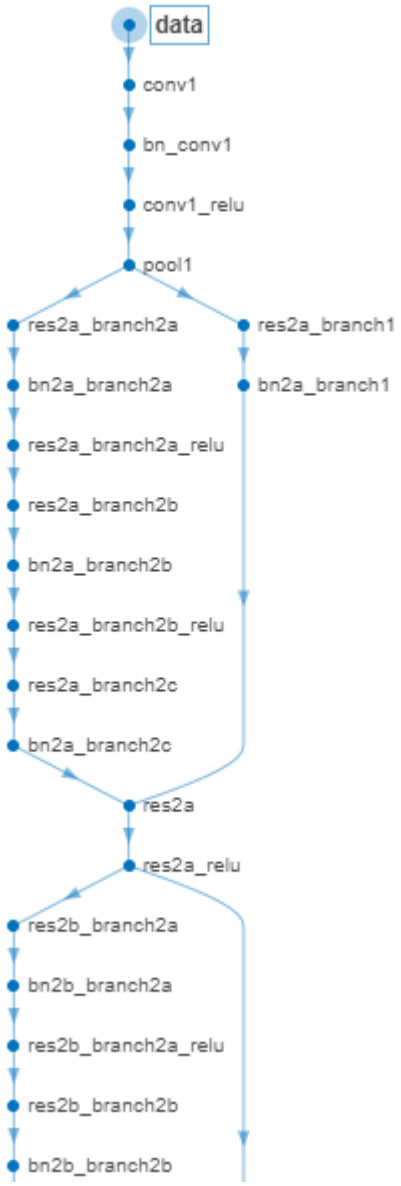
3.2.3.a. Çalışmada kullandığımız transfer öğrenmesi ağ yapıları

ResNet-101

Transfer öğrenmesi olarak kullandığımız ağ yapısı ResNet-101'dir. 101 katman derinliğinden oluşan ağ, yaş sınıflandırma için 50 epoch, cinsiyet sınıflandırma için 100 epoch, 0,001 öğrenme katsayısı ve 64 mini yığın boyutu parametreleri ile çalıştırıldı.

layers(1:5) = freezeWeights(layers(1:5)); komutu ile ağdaki ilk 5 katmanın ağırlık değerleri dondurulmuştur. Diğer katmanların ağırlık değerleri eğitim esnasında ağın özelliklerine göre değiştirilerek iyileştirilmeye çalışılmıştır.

224x224x3 boyutundaki imgeler Şekil 3.25'te gösterildiği gibi evrişim, toplama işlemlerinden geçirilir. Ortalama havuzlama katmanına kadar gelen imgeler 7x7x2048 boyutuna dönüştürülür.



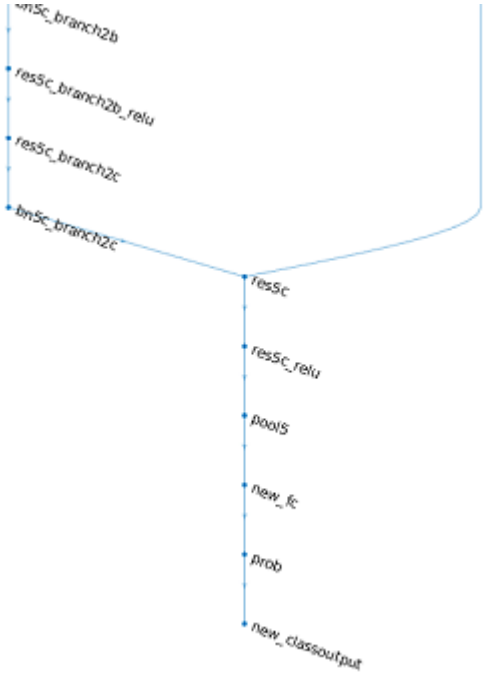
Şekil 3.25. ResNet-101 ilk katmanları

Ortalama havuzlama katmanında uygulanan filtre ile 1x1x2048 boyutuna dönüştürülen imgeler Şekil 3.26'da gösterildiği gibi tam bağlantılı katmanda 1000 sınıflı eğitim seti kullanıldığı için 1000x2048 bağlantı oluşturulur. Bizim ağımızda tam bağlantılı katman

çıkışında cinsiyet sınıflandırma için 2x2048, yaş sınıflandırma için 8x2048 bağlantı oluşturulur.

Şekil 3.26. ResNet-101 son katmanları

ResNet-101'in orijinal şeklindeki son 3 katmanı atarak yerine kendi modelimizin 3 katmanını ekledik. Yaş sınıflandırma için sınıflandırma katmanı 8 sınıf iken cinsiyet sınıflandırma da 2 sınıftır. Şekil 3.27'de gösterilmiştir.



Şekil 3.27. ResNet-101 değiştirilen katmanlar

ResNet-50 ve ResNet-18 modellerini de ResNet-101 ile aynı şekilde transfer öğrenmesinde kullandık. ResNet-50 ve ResNet-18 sadece katman sayısı olarak birbirinden farklıdır.

Inception-v3

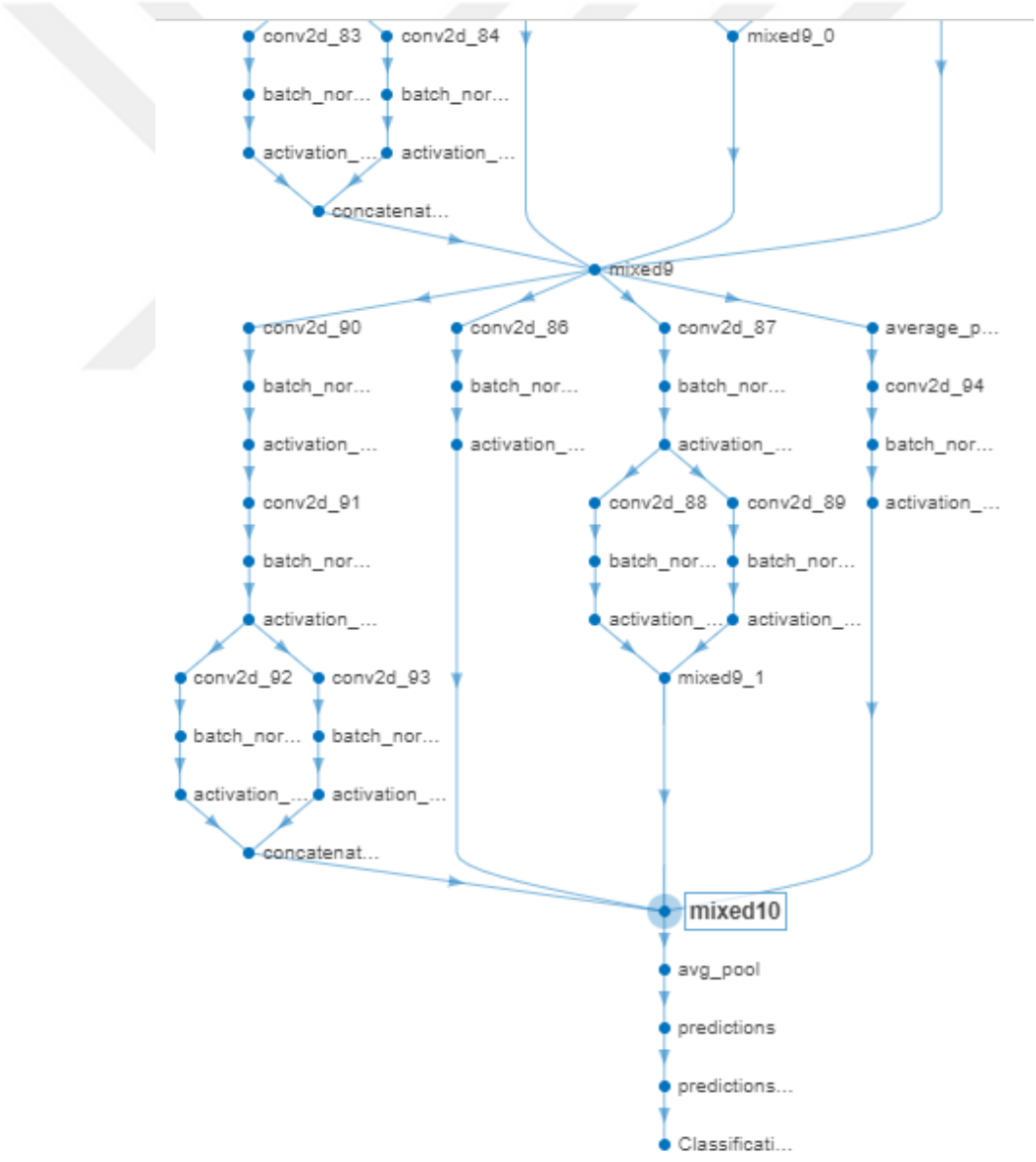
Transfer öğrenmesinde kullandığımız ön eğitimli ağlardan bir diğeri de inception-v3'tür. Burada da yaş sınıflandırma için 50 epoch, cinsiyet sınıflandırma için 100 epoch, 0,001 öğrenme katsayısı ve 64 mini yığın boyutu parametreleri ile çalıştırıldı. İlk 5 katmanın ağırlık değerleri dondurulmuştur.

299x299x3 boyutundaki imgeler Şekil 3.28'de gösterildiği gibi evrişim, derinlik bağlama işlemlerinden geçirilir. Ortalama havuzlama katmanına kadar gelen imgeler 8x8x2048 boyutuna dönüştürülür.

Ortalama havuzlama katmanında uygulanan filtre ile 1x1x2048 boyutuna dönüştürülen imgeler Şekil 3.28'de gösterildiği gibi tam bağlantılı katmanda 1000 sınıflı eğitim seti

kullanıldığı için 1000x2048 bağlantı oluşturulur. Bizim ağımızda tam bağlantılı katman çıkışında cinsiyet sınıflandırma için 2x2048, yaş sınıflandırma için 8x2048 bağlantı oluşturulur.

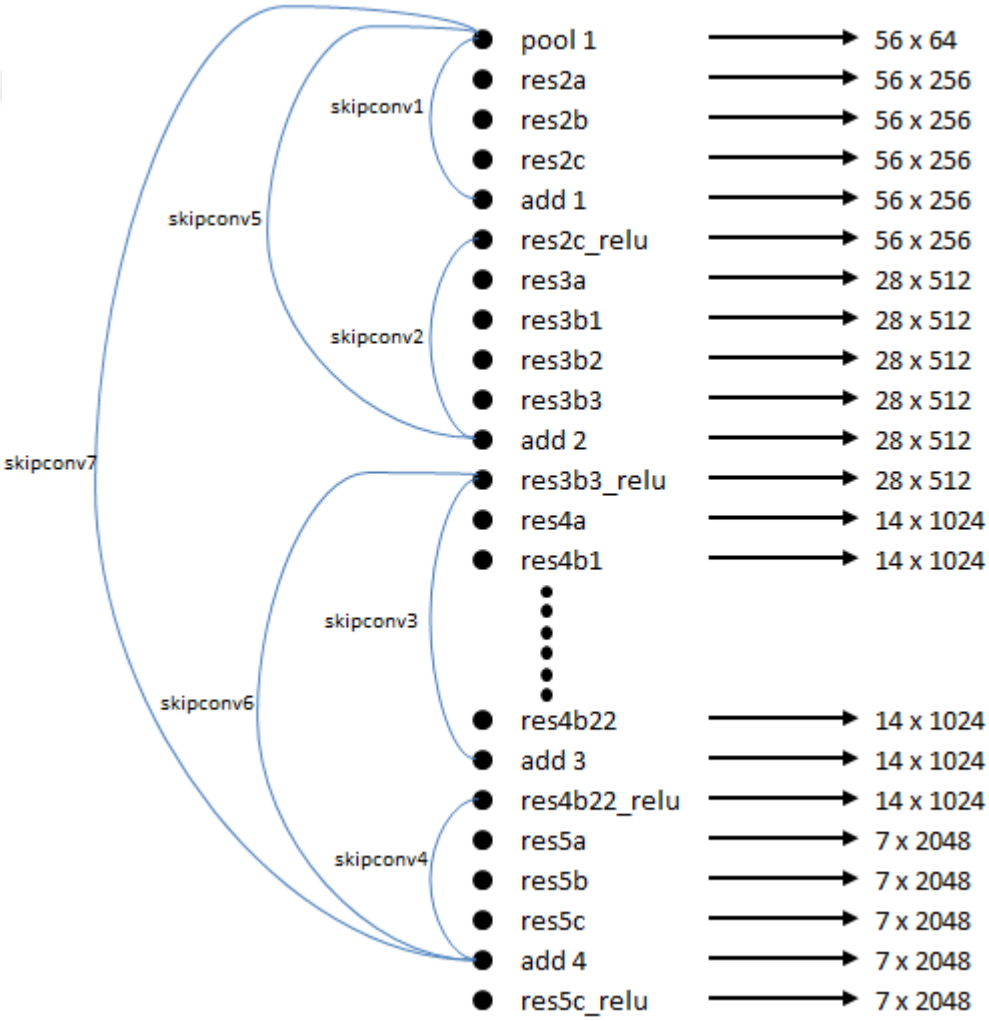
Inception modeli tasarım olarak ResNet'e benzese de aralarındaki en büyük fark toplama ve derinlik bağlama işlemleridir. ResNet'te aynı boyuttaki matrisler birbirleriyle üst üste toplanırken, inception modelinde ard arda birbirine bağlanarak derinlik arttırılmış olur.



Şekil 3.28. Inception-v3 modelinin son katmanları

ResNet-101-RoR

ResNet-101 modelinin transfer learning ile eğitiminden sonra ResNet-101’ in içine atlama konvolüsyonlarını ekleyerek ResNet-101-RoR modelini oluşturduk. Bu model ile ResNet-101’ de önceden eğitilerek elde edilen ağırlık değerleri başlangıç parametresi seçilerek bu parametreler iyileştirilmeye çalışılmıştır.



Şekil 3.29. ResNet-101-RoR atlama konvolüsyon bağlantıları

Şekil 3.29’ da ResNet-101-RoR modelinde kullandığımız atlama konvolüsyonları ve eklediğimiz toplama katmanları görülmektedir. Katman isimleri Matlab’ da ResNet-101’ in network analizörde bize verdiği isimlerdir.

3.3. Deneysel Kurulum

3.3.1. Hiper parametreler

Ne seçilmesi gerektiği, ağı tasarlayan kişiye bırakılmış, veri seti ve probleme göre değişiklik gösteren parametreler, hiper parametreler olarak ifade edilmektedir (Çarkacı 2019). Bu parametrelerin ne olacağı belli değildir.

Çalışmalarımızı Matlab'da yaptığımız için bu bölümde anlatacağımız hiper parametlerde Matlab'da var olan parametrelerden olacaktır. Burada bahsedeceklerimizin dışında da birçok parametre vardır.

3.3.1.a. Öğrenme katsayısı (LearnRate)

Eğim düşüm algoritmalarında kullanılan bu katsayı, hata düzeltme katsayısı olarak da adlandırılmaktadır. Derin öğrenmede ağırlık değerlerinin güncellenmesi geriye yayılım işlemi ile yapılmaktadır. Geriye yayılım işleminde geriye doğru türev alınır ve aradaki fark hesaplanarak öğrenme katsayısı ile çarpılır. Çıkan sonuç önceki ağırlık değerlerinden çıkarılarak yeni ağırlık değeri hesaplanmış olur.

Öğrenme katsayısının büyük olması salınıma neden olacaktır ve hedefe ulaşılamayabilecektir. Katsayının küçük olması da öğrenme işinin küçük adımlarla gerçekleşmesine neden olacak ve öğrenim çok fazla zaman alacaktır. Bu sebeple öğrenme katsayısını doğru seçmek önemlidir. Uygulama yaparken katsayıyı büyük seçip ihtiyaç halinde küçülterek işlem yapılmasında fayda vardır.

3.3.1.b. Devir (Epoch)

Ağ eğilirken verilerin hepsi aynı anda eğitime giremez. Kümeler halinde işlem yapılır. Birinci parça eğitildikten sonra geri yayılım ile ağırlıklar hesaplanır güncellenir. Daha

sonra başka bir küme eğitime dahil olur tekrar geri yayılım ile ağırlıklar güncellenir. Bu işlemler verilerin hepsi eğitime girip en uygun ağırlık değerleri bulununcaya kadar her bir adımda tekrar eder. İşte bu adımların her birine devir denir. Devir sayısının fazla seçilmesi elde edilecek başarı oranının da artacağı anlamına gelmiyor.

3.3.1.c. Mini-parça boyutu (Mini-batch Size)

Mini-parça, kayıp fonksiyonunun eğimini değerlendirmek ve ağırlıkları güncellemek için kullanılan eğitim setinin bir alt kümesidir.

Mini-parça boyutu, ileri ve geri yönde yayılım için veri setinden alınan veri miktarıdır. Boyut büyüdükçe, veriyi taşıyabilmek için daha fazla hafıza gerekir. Yüksek paket boyutu seçilmesi her iterasyon için kullanılacak eğitim örneğinin de büyümesi demektir.

3.3.1.d. Karıştırma (Shuffle)

Aşağıda seçebileceğiniz seçeneklere göre verilerin karıştırılma işlemidir. Matlab'da bu seçenekler şöyle gözükür.

once – Eğitimden önce bir kez eğitim ve doğrulama verilerini karıştırır.
never – Asla verileri karıştırmaz.
every-epoch – Her eğitim devrinden önce eğitim verilerini ve her ağ doğrulama işleminden önce doğrulama verilerini karıştırır.

3.3.1.e. Eğim düşüm algoritmaları (Gradient Descent Algorithms)

Eğim düşüm için Matlab'da üç tane algoritma var. Bunların dışında da farklı algoritmalar mevcut ancak biz sadece üç tanesinden bahsedeceğiz.

1) Momentumlu stokastik eğim düşüm (Stochastic Gradient Descent with Momentum)

Stokastik eğim düşüm algoritması, en dik eğim yolu boyunca optimum seviyeye salınım yapabilir. Parametre güncellemesine momentum terimi eklemek bu salınımı azaltmanın bir yoludur. Momentum güncellemesi ile stokastik eğim düşümü Denklem (3.12)'de belirtildiği gibidir.

$$\theta_{l+1} = \theta_l - \alpha \nabla E(\theta_l) + \gamma(\theta_l - \theta_{l-1}) \tag{3.12}$$

γ önceki düşüm adımının geçerli yinelemeye katkısını belirler. Matlab'da γ değerini 'Momentum' isimli bağımsız değişkeni kullanarak belirliyoruz. Yine Matlab'da bir sinir ağını eğitmek için stokastik eğim düşümünü momentumla kullanmak istediğimizde, solverName 'sgdm' olarak belirtmek gerekiyor. α öğrenme katsayısının başlangıç değerini belirlemek için 'InitialLearnRate' isimli bağımsız değişkeni kullanıyoruz. (Matlab Documentation 2019).

2) Adaptif Moment Tahmin (Derived from Adaptive Moment Estimation - Adam)

Adam algoritmasında önceki eğimlerin karelerinin üssel olarak azalan ortalamalarının (m_t) depolanmasının yanında, momentumdaki geçmiş eğimlerin üssel olarak azalan ortalamalarını (v_t) da tutar (Ruder 2016).

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1) g_t \\ v_t &= \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \end{aligned} \tag{3.13}$$

β_1 ve β_2, hesaplanacak ortalama sayısını tanımlamaya yardımcı olur. g_t, eğim düşümünden gelen değerdir. $m_t$ve v_t'nin başlangıç değerleri sıfıra eşittir.

Sapma doğrulaması için m_t ve m_t değerleri ayarlanmalıdır. Bu işlem öğrenmenin ilk aşamalarında daha iyi bir tahmin sonucu elde edilmesine yardımcı olur. $\hat{m}_t$ve $\hat{v}_t$ düzeltilmiş değerleri gösterir.

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}$$
$$\hat{v}_t = \frac{v_t}{1-\beta_2^t} \quad (3.14)$$

Denklem (3.15), Adam algoritmasının ana formülünü gösterir. ε hataların sıfıra bölünmesini önler. α, 15 öğrenme katsayısı değerini ifade eder (Kingma and Ba 2015). β_1 için 0,9, $\beta_2$için 0,999 ve ε için 10−8 değerlerini önermektedir.

$$\theta_{t+1} = \theta_t - \frac{\alpha \hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon} \quad (3.15)$$

Genel olarak diğer uyarlanabilir öğrenme yöntemi algoritmalarından daha iyi performans gösterir.

3) Kök ortalama kare yayılımı (RMSprop)

Momentumlu stokastik gradyan inişi, tüm parametreler için tek bir öğrenme oranı kullanır. Diğer optimizasyon algoritmaları, farklı parametreler için farklı olan ve optimize edilen kayıp fonksiyonuna otomatik olarak adapte olabilen öğrenme oranlarını kullanarak ağ eğitimini iyileştirmeye çalışır. Kök ortalama kare yayılımı bu tür bir algoritmadır.

$$v_l = \beta_2 v_{l-1} + (1-\beta_2)[\nabla E(\theta_l)]^2 \quad (3.16)$$

Kök ortalama kare yayılımı algoritması, her bir parametrenin güncellemelerini ayrı ayrı normalleştirmek için bu hareketli ortalamayı kullanır.

$$\theta_{l+1} = \theta_l - \frac{\alpha \nabla E(\theta_l)}{\sqrt{v_l} + \varepsilon} \tag{3.17}$$

3.3.2. Performans ölçütleri

3.3.2.a. Karışıklık matrisi (Confusion Matrix)

Karışıklık matrisi, bir sınıflandırma problemine ait tahminlerin sonuçlarının özetidir. Makine öğreniminde ve özellikle istatistiksel sınıflandırma problemlerinde hata matrisi olarak da bilinir.

Bir karışıklık matrisi, gerçek değerlerin bilindiği bir dizi test verisi üzerinde sınıflandırma modelinin performansını tanımlamak için sıkça kullanılan bir tablodur. Algoritmanın performansının görselleştirilmesine izin verir.

Örneğin, sınıflar arasındaki karışıklığın kolayca tanımlanmasını sağlar. Bir sınıf genellikle diğer sınıf olarak yanlış etiketlenir. Çoğu sistem performans ölçütü olarak, karışıklık matrisinden faydalanır (Sharma).

Karışıklık matrisinin boyutu B x B boyutunda olabilir. Burada B, sınıflandırma algoritmasındaki etiket sayısıdır. Cinsiyet sınıflandırma gibi iki sınıflı bir sınıflandırmada karışıklık matrisi Şekil 3.30'da ki gibidir.

	Tahmin Edilen 1.Sınıf (Kadın)	Tahmin Edilen 2.Sınıf (Erkek)
Gerçekte Olan 1.Sınıf (Kadın)	DK	YK
Gerçekte Olan 2.Sınıf (Erkek)	YE	DE

Şekil 3.30. Cinsiyet sınıflandırma karışıklık matrisi

Doğru Kadın (DK) : Gerçekte kadın sınıfına ait verinin, kadın diye tahmin edilmesi.
Yanlış Kadın (YK) : Gerçekte kadın sınıfına ait verinin, erkek diye tahmin edilmesi.
Doğru Erkek (DE) : Gerçekte erkek sınıfına ait verinin, erkek diye tahmin edilmesi.
Yanlış Erkek (YE) : Gerçekte erkek sınıfına ait verinin, kadın diye tahmin edilmesi.

Sınıflandırma doğruluğu Denklem (3.18)'de verilmiştir.

$$Doğruluk = \frac{DK + DE}{DK + YK + DE + YE} \tag{3.18}$$

Doğru Erkek oranı (hassaslık) Denklem (3.19)'da verilmiştir.

$$Hassaslık = \frac{DE}{DE + YE} \tag{3.19}$$

Doğru Kadın oranı (Özgüllük) Denklem (3.20)'de verilmiştir.

$$Özgüllük = \frac{DK}{DK + YK} \tag{3.20}$$

Yanlış Kadın oranı Denklem (3.21)'de verilmiştir.

$$Yanlış\ Kadın\ Oranı = \frac{YK}{DK + YK} \tag{3.21}$$

Yanlış Erkek oranı Denklem (3.22)'de verilmiştir.

$$Yanlış\ Erkek\ Oranı = \frac{YE}{DE + YE} \tag{3.22}$$

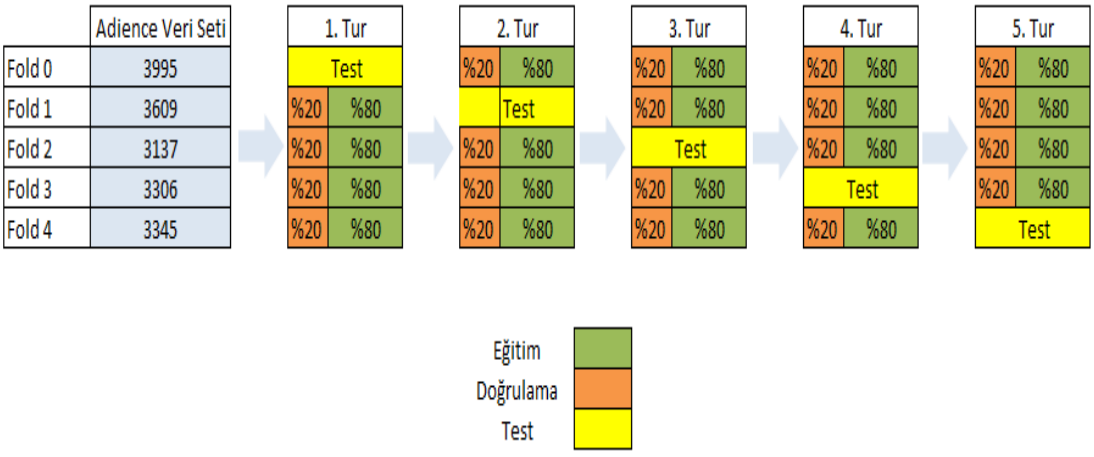
Kesinlik değerini elde etmek için, doğru sınıflandırılmış erkek örneklerin toplam sayısı, öngörülen erkek örneklerin toplam sayısına bölünür. Denklem (3.23)'te verilmiştir.

$$Kesinlik = \frac{DE}{DE + YK} \tag{3.23}$$

Yüksek kesinlik için YK değerinin düşük olması gerekmektedir (Kohavı and Provost 1998).

3.2.3.b. K-Katlamalı çapraz doğrulama (K-Fold Cross Validation)

Çapraz Doğrulama, verileri bölümlere ayırarak öğrenme algoritmalarını değerlendirmek ve karşılaştırmak için kullanılan istatistiksel bir yöntemdir. K-katlamalı çapraz doğrulamada data set önce hemen hemen eşit büyüklükte k tane parçaya ayrılır. k-1 parça eğitim ve doğrulama için kullanılırken 1 parçada test seti için kullanılır ve k defa bu işlem tekrar eder. Her turda elde edilen sonuçlar toplanır ve ortalaması alınır. Ortalama değeri bize ağın performansını verir (Refaeılzadeh *et al.* 2009). Şekil 3.30'da bizim çalışmalarımızda kullandığımız Adience data setinin K-katlamalı çapraz doğrulama yöntemine göre katlara göre verilerin dağılımı gösterilmiştir. Her turda bir kat test için kullanılmıştır. Kalan 4 katın %20 si doğrulama için, %80'i ise eğitim için kullanılmıştır. Bu durum farklı turlarda farklı katların test için kullanılması şeklinde devam eder.



Şekil 3.31. K-Katlamalı çapraz doğrulama

3.2.3.c. Oy çokluğu ile ağların füzyonu

Bütün testler tamamlandıktan sonra sonuçlar oy çokluğu ile birbirleri arasında karşılaştırıldı. Test aşamasında, her bir foldda yapılan tahminler diğer modellerdeki sonuçlar ile karşılaştırılarak sayıca en fazla olan tahmini test sonucu olarak kabul eden bir yöntem uygulandı. Örneğin fold0 klasöründeki bir resmin, CNN modelinde '0-2' yaş grubu, ResNet-101-RoR modelinde '8-13' yaş grubu, ResNet-101 modelinde '8-13' yaş grubu, ResNet-50 modelinde '0-2' yaş grubu, InceptionV3 modelinde '0-2' yaş grubu, Pre-RoR modelinde '4-6' yaş grubu, ResNet-18 modelinde ise '0-2' yaş grubu olarak tahmin edildiğini varsayalım. Bu modellere oy çokluğu uygulandığında test sonucu '0-2' olarak kabul edilir. Çünkü tahminler içerisinde '0-2' yaş grubu sayıca diğerlerinden daha fazla. Çizelge 3.5'de örnek hesaplama verilmiştir.

Oy çokluğu yönteminde hesaplamaya dahil edilen modeller CNN, ResNet-101, ResNet-50, ResNet-18, Pre-RoR, ResNet-101-RoR ve InceptionV3' tür. Karşılaştırma sonucu daha doğru olabilsin diye hesaplamaya katılacak modellerin toplam sayısı tek sayı olacak şekilde modeller belirlendi. Bu sebeple CNN-SVM modelini hesaplamaya katmadık.

Çizelge 3.5. Örnek oy çokluğu hesaplama

CNN	Resnet-101-RoR	Resnet-101	Resnet-50	InceptionV3	Pre-RoR	Resnet-18	Oy Çokluğu
'0-2'	'0-2'	'0-2'	'0-2'	'0-2'	'4-6'	'0-2'	'0-2'
'0-2'	'0-2'	'0-2'	'0-2'	'0-2'	'0-2'	'0-2'	'0-2'
'0-2'	'4-6'	'4-6'	'0-2'	'0-2'	'4-6'	'0-2'	'0-2'
'0-2'	'4-6'	'4-6'	'0-2'	'0-2'	'8-13'	'4-6'	'0-2'
'0-2'	'0-2'	'0-2'	'0-2'	'0-2'	'0-2'	'0-2'	'0-2'
'0-2'	'4-6'	'4-6'	'0-2'	'0-2'	'4-6'	'0-2'	'0-2'
'0-2'	'0-2'	'0-2'	'0-2'	'0-2'	'4-6'	'0-2'	'0-2'
'0-2'	'0-2'	'0-2'	'0-2'	'0-2'	'4-6'	'0-2'	'0-2'
'0-2'	'0-2'	'0-2'	'0-2'	'0-2'	'4-6'	'0-2'	'0-2'
'4-6'	'0-2'	'0-2'	'0-2'	'0-2'	'0-2'	'0-2'	'0-2'
'4-6'	'0-2'	'0-2'	'0-2'	'0-2'	'8-13'	'38-43'	'0-2'
'0-2'	'4-6'	'4-6'	'0-2'	'0-2'	'4-6'	'0-2'	'0-2'
'0-2'	'8-13'	'8-13'	'0-2'	'0-2'	'4-6'	'0-2'	'0-2'
'0-2'	'0-2'	'0-2'	'0-2'	'0-2'	'4-6'	'0-2'	'0-2'
'0-2'	'0-2'	'0-2'	'0-2'	'0-2'	'4-6'	'0-2'	'0-2'
'0-2'	'0-2'	'0-2'	'0-2'	'0-2'	'0-2'	'0-2'	'0-2'
'0-2'	'0-2'	'0-2'	'0-2'	'0-2'	'4-6'	'0-2'	'0-2'
'0-2'	'0-2'	'0-2'	'0-2'	'0-2'	'0-2'	'0-2'	'0-2'
'0-2'	'0-2'	'0-2'	'0-2'	'0-2'	'4-6'	'0-2'	'0-2'

4. ARAŞTIRMA BULGULAR

Yaptığımız testlerde kullandığımız modellerin detaylarından bahsetmiştik. Bu bölümde testlerde elde ettiğimiz başarımları tablolar halinde sunacağız. Testler 5 fold'un hepsi için ayrı ayrı yapıldı ve 5 testin ortalaması alınarak başarı oranı hesaplandı.

Testler üzerinde 3 adet 12GB GeForce RTX 2080 Ti bulunan 128 GB ram li bilgisayarda Matlab programı kullanılarak yapıldı.

CNN ağını kullanarak yaptığımız test sonuçları Çizelge 4.1, Çizelge 4.2 ve Çizelge 4.3'te verilmiştir.

Çizelge 4.1. CNN cinsiyet sınıflandırma 32 mini-batch

	Validation	Test	Süre
Fold 0	92,19	86,71	88 dakika
Fold 1	92,33	85,76	91 dakika
Fold 2	93,1	84,25	97 dakika
Fold 3	93,83	90,05	92 dakika
Fold 4	93,13	82,87	92 dakika
Ortalama	92,92	85,93	

Standart Sapma	± 2,44

Çizelge 4.2. CNN cinsiyet sınıflandırma 64 mini-batch

	Validation	Test	Süre
Fold 0	92,7	85,88	81 dakika
Fold 1	92,96	85,76	82 dakika
Fold 2	93,93	82,37	86 dakika
Fold 3	92,78	88,29	85 dakika
Fold 4	92,99	81,94	84 dakika
Ortalama	93,07	84,85	

Standart Sapma	± 2,38

Çizelge 4.3. CNN yaş sınıflandırma 32 mini-batch

	Validation	Test	Süre
Fold 0	77,14	60,51	90 dakika
Fold 1	80,36	47,16	95 dakika
Fold 2	78,4	53,49	94 dakika
Fold 3	78,52	47,26	95 dakika
Fold 4	80,53	53,55	92 dakika
Ortalama	78,99	52,39	

Standart Sapma	± 4,94

CNN modelinden sonra CNN+SVM modelinin testleri yapıldı. Test sonuçları Çizelge 4.4 ve Çizelge 4.5'te verilmiştir.

Çizelge 4.4. CNN + SVM cinsiyet sınıflandırma

	CNN	CNN+SVM
Fold 0	86,71	86,76
Fold 1	85,76	85,52
Fold 2	84,25	84,04
Fold 3	90,05	89,92
Fold 4	82,87	82,53
Ortalama	85,93	85,75

Standart Sapma	± 2,44	± 2,52

Çizelge 4.5. CNN + SVM yaş sınıflandırma

	CNN	CNN+SVM
Fold 0	60,51	49,43
Fold 1	47,16	40,9
Fold 2	53,49	44,96
Fold 3	47,26	45,11
Fold 4	53,55	45,54
Ortalama	52,39	45,19

Standart Sapma	± 4,94	± 2,70

CNN+SVM modelinden sonra ResNet-101 modelini transfer learning ile veri setimize uyguladık. Test sonuçları Çizelge 4.6 ve Çizelge 4.7'de verilmiştir.

Çizelge 4.6. ResNet-101 transfer learning cinsiyet sınıflandırma

	Validation	Test	Süre
Fold 0	95,07	92,04	446 dakika
Fold 1	94,74	89,97	481 dakika
Fold 2	94,53	88,94	516 dakika
Fold 3	95,07	91,95	452 dakika
Fold 4	95,34	88,56	466 dakika
Ortalama	94,95	90,29	

Standart Sapma	± 1,47

Çizelge 4.7. ResNet-101 transfer learning yaş sınıflandırma

	Validation	Test	Süre
Fold 0	80,64	60,37	206 dakika
Fold 1	80,95	51,14	474 dakika
Fold 2	82,44	56,83	455 dakika
Fold 3	81,15	48,34	476 dakika
Fold 4	82,3	55,28	481 dakika
Ortalama	81,50	54,39	

Standart Sapma	± 4,23

ResNet-50 ve ResNet-18 modellerini transfer learning ile veri setimize uyguladık. Test sonuçları Çizelge 4.8, Çizelge 4.9, Çizelge 4.10 ve Çizelge 4.11'de verilmiştir.

Çizelge 4.8. ResNet-50 transfer learning cinsiyet sınıflandırma

	Validation	Test	Süre
Fold 0	93,74	89,54	95 dakika
Fold 1	94,53	87,34	95 dakika
Fold 2	94,98	89,42	103 dakika
Fold 3	95,42	91,74	100 dakika
Fold 4	95,66	87,58	98 dakika
Ortalama	94,87	89,12	

Standart Sapma	± 1,59

Çizelge 4.9. ResNet-50 transfer learning yaş sınıflandırma

	Validation	Test	Süre
Fold 0	81,78	63,11	89 dakika
Fold 1	82,27	46,06	104 dakika
Fold 2	80,01	58,84	100 dakika
Fold 3	80,73	52,05	98 dakika
Fold 4	81,55	55,75	94 dakika
Ortalama	81,27	55,16	

Standart Sapma	± 5,82

Çizelge 4.10. ResNet-18 transfer learning cinsiyet sınıflandırma

	Validation	Test	Süre
Fold 0	94,74	91,89	66 dakika
Fold 1	97,52	89,39	69 dakika
Fold 2	95,86	84,22	67 dakika
Fold 3	95,45	91,95	65 dakika
Fold 4	95,02	88,39	65 dakika
Ortalama	95,72	89,17	

Standart Sapma	± 2,84

Çizelge 4.11. ResNet-18 transfer learning yaş sınıflandırma

	Validation	Test	Süre
Fold 0	83,26	64,14	60 dakika
Fold 1	84,32	49,18	67 dakika
Fold 2	82,16	57,05	66 dakika
Fold 3	82,67	50,42	68 dakika
Fold 4	84,9	53,44	65 dakika
Ortalama	83,46	54,85	

Standart Sapma	± 5,38

Matlab' da oluşturduğumuz Pre-RoR modelin test sonuçları Çizelge 4.12 ve Çizelge 4.13' te verilmiştir.

Çizelge 4.12. Pre-RoR model ile cinsiyet sınıflandırma

	Validation	Test	Süre
Fold 0	92,7	87,88	64 dakika
Fold 1	92,37	85,31	64 dakika
Fold 2	92,02	84,12	65 dakika
Fold 3	92,11	88,51	66 dakika
Fold 4	93,99	85,69	65 dakika
Ortalama	92,64	86,30	

Standart Sapma	± 1,64

Çizelge 4.13. Pre-RoR model ile yaş sınıflandırma

	Validation	Test	Süre
Fold 0	39,44	29,47	72 dakika
Fold 1	60,58	38,35	80 dakika
Fold 2	53,39	47,46	75 dakika
Fold 3	59,25	51,11	77 dakika
Fold 4	40,41	30,73	75 dakika
Ortalama	50,61	39,42	

Standart Sapma	± 8,68

ResNet-101-RoR modeli ile yaptığımız testlerin sonuçları Çizelge 4.14 ve Çizelge 4.15' te verilmiştir.

Çizelge 4.14. ResNet-101-RoR modeli ile cinsiyet sınıflandırma

	Validation	Test	Süre
Fold 0	93,81	90,86	470 dakika
Fold 1	94,31	88,31	502 dakika
Fold 2	94,78	88,3	502 dakika
Fold 3	95,21	90,05	514 dakika
Fold 4	94,8	86,3	535 dakika
Ortalama	94,58	88,76	

Standart Sapma	± 1,58

Çizelge 4.15. ResNet-101-RoR modeli ile yaş sınıflandırma

	Validation	Test	Süre
Fold 0	78,01	62,24	481 dakika
Fold 1	81,64	49,08	508 dakika
Fold 2	81,07	56,13	497 dakika
Fold 3	80,94	51,08	505 dakika
Fold 4	81,33	53,68	487 dakika
Ortalama	80,60	54,44	

Standart Sapma	± 4,57

InceptionV3 modelinde transfer learning yöntemi ile yaptığımız testlerin sonuçları Çizelge 4.16 ve Çizelge 4.17' de verilmiştir.

Çizelge 4.16. InceptionV3 modeli ile cinsiyet sınıflandırma

	Validation	Test	Süre
Fold 0	95,63	91,06	378 dakika
Fold 1	95,68	90,58	399 dakika
Fold 2	95,86	88,62	408 dakika
Fold 3	95,03	91,98	404 dakika
Fold 4	96,16	88,65	434 dakika
Ortalama	95,67	90,18	

Standart Sapma	± 1,34

Çizelge 4.17. InceptionV3 modeli ile yaş sınıflandırma

	Validation	Test	Süre
Fold 0	82,54	63,5	370 dakika
Fold 1	85,53	49,24	402 dakika
Fold 2	83,6	58,63	401 dakika
Fold 3	83,64	49,51	404 dakika
Fold 4	82,59	55,5	390 dakika
Ortalama	83,58	55,28	

Standart Sapma	± 5,45

Yaptığımız tüm testler ve fold'lara göre sonuçların ortalama değerleri cinsiyet sınıflandırma için Çizelge 4.18' de yaş sınıflandırma için Çizelge 4.19' da verilmiştir.

Çizelge 4.18. Bütün test sonuçlarının cinsiyet sınıflandırma başarı oranı

Model	Başarı
CNN	% 85,93
CNN+SVM	% 85,75
ResNet-101	% 90,29
ResNet-50	% 89,12
ResNet-18	% 89,17
Pre-RoR	% 86,30
ResNet-101-RoR	% 88,76
InceptionV3	% 90,18

Çizelge 4.19. Bütün test sonuçlarının yaş sınıflandırma başarı oranı

Model	Başarı
CNN	% 52,39
CNN+SVM	% 45,19
ResNet-101	% 54,39
ResNet-50	% 55,16
ResNet-18	% 54,85
Pre-RoR	% 39,42
ResNet-101-RoR	% 54,44
InceptionV3	% 55,28

Oy çokluğu yöntemi ile elde edilen sonuçlar Çizelge 4.20' de verilmiştir.

Çizelge 4.20. Oy Çokluğu yöntemi ile yaş ve cinsiyet sınıflandırma

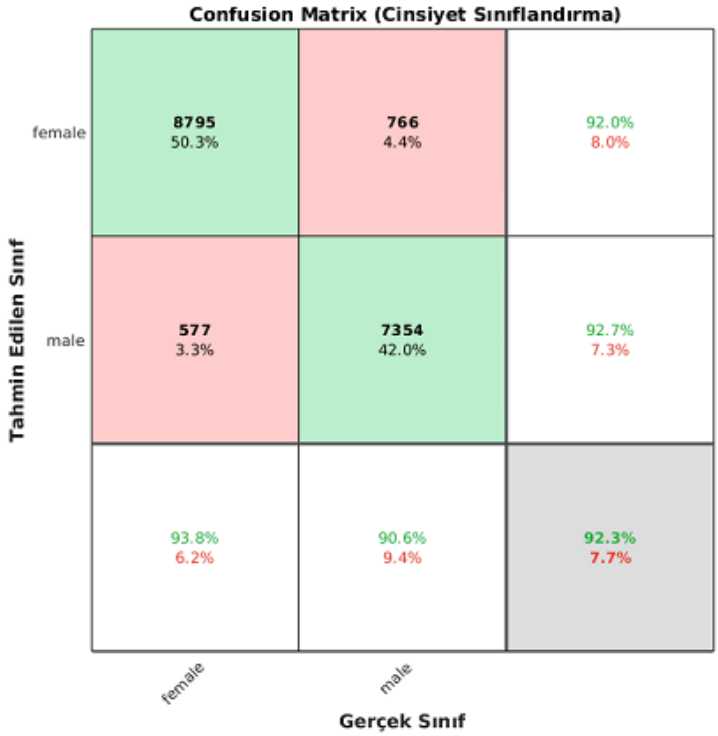
Oy Çokluğu	Yaş Sınıflandırma	Cinsiyet Sınıflandırma
Fold 0	66,54	93,17
Fold 1	54,5	92,35
Fold 2	63,1	91,3
Fold 3	57,7	94,22
Fold 4	59,47	90,42
Ortalama	60,26	92,29
Standart Sapma	± 4,19	± 1,34

Oy çokluğu yöntemi ile diğer yöntemlerin sonuçlarının karşılaştırılması Çizelde 4.21' de verilmiştir.

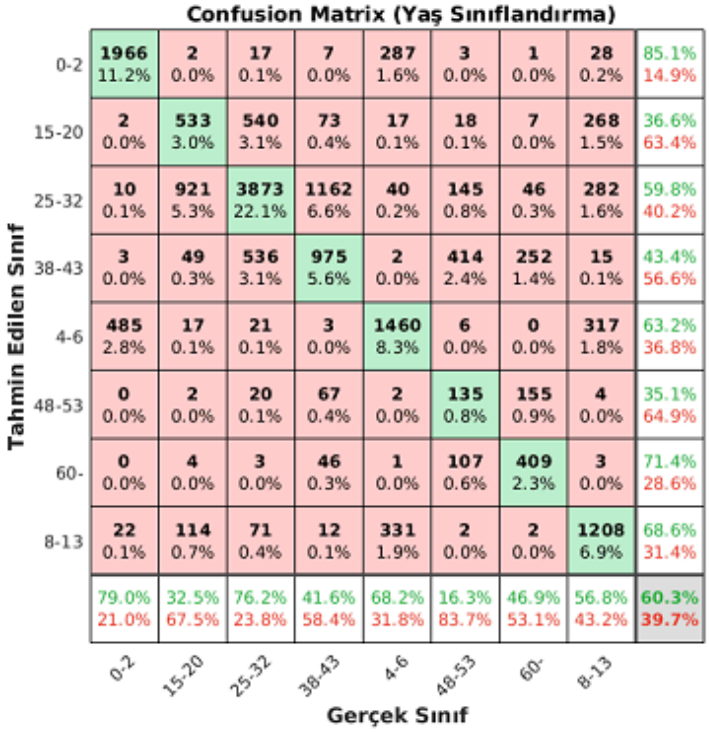
Çizelge 4.21. Test sonuçlarının karşılaştırılması

Model	Başarı	Başarı
CNN	% 85,93	% 52,39
CNN+SVM	% 85,75	% 45,19
ResNet-101	% 90,29	% 54,39
ResNet-50	% 89,12	% 55,16
ResNet-18	% 89,17	% 54,85
Pre-RoR	% 86,30	% 39,42
ResNet-101-RoR	% 88,76	% 54,44
InceptionV3	% 90,18	% 55,28
Önerilen Oy Çokluğu Yöntemi	**%92,29**	**% 60,26**

Oy çokluğu ile füzyon yaptığımız modelin tahmin değerleri ile gerçek değerlerin confusion matrisleri Şekil 4.1 ve Şekil 4.2' den görülebilir. Çapraz inen hücreler doğru sınıflandırılmış tahminleri göstermektedir. Her bir hücrede hem gözlem sayısı hem de toplam gözlem sayısının yüzdesi gösterilmektedir. Grafiğin en sağındaki sütun, doğru ve yanlış sınıflandırılmış her bir sınıfa ait olduğu tahmin edilen tüm örneklerin yüzdesini gösterir. Grafiğin altındaki satır, her sınıfa ait tüm örneklerin yüzdelerini doğru ve yanlış sınıflandırılmış olarak gösterir. Grafiğin sağ alt köşesindeki hücre ise genel doğruluğu gösterir.



Şekil 4.1. Confusion matris (Cinsiyet Sınıflandırma)



Şekil 4.2. Confusion matris (Yaş Sınıflandırma)

Literatürde önceden yapılan çalışmaların başarı oranları Çizelge 4.22' de verilmiştir.

Çizelge 4.22. Literatürde önceden yapılan çalışmaların başarı oranları

Çalışma	Yaş Sınıflandırma	Cinsiyet Sınıflandırma
Levi and Hassner 2015	% 50,7	%86,8
Wolfshaar et al. 2015	Çalışma yok	%86,2
Ekmekji 2016	% 50,2	%80,8
Rodríguez et al. 2017	% 57,8	%92,4
Hosseini et al. 2018	% 61,3	%88,9
Akbulut vd 2017	Çalışma yok	%87,13
Zhang et al. 2017	% 67,34	%93,24
Önerilen Oy Çokluğu Yöntemi	**% 60,26**	**%92,29**

Çizelge 4.22' den görülebileceği üzere önerdiğimiz çalışma en yüksek başarı veren yöntemlerin içindedir. (Zhang *et al.* 2017) çalışmalarında ağlarını ön eğitim ve ince ayardan geçirdikleri için başarı oranları daha yüksektir. Bizim önerdiğimiz yöntemde herhangi bir ön eğitim ve ince ayar işlemleri yapılmadığından başarı oranı biraz daha düşüktür.

5. TARTIŞMA ve SONUÇ

Bu tezde evrişimli sinir ağlarının farklı modelleri kullanılarak yaş ve cinsiyet sınıflandırma uygulamaları yapılmıştır. Kullandığımız veri setindeki resimler gerçek hayattan alınan filtresiz, doğal resimler olduğundan test sonuçlarının doğruluğu da bir o kadar gerçek hayata yakındır.

Yaptığımız testler, en iyi sonuçlara ulaşabilmek için mini-batch parametrelerinin seçiminin ağ model yapısına göre farklı farklı olduğunu göstermektedir. Mini-batch değerini mümkün oldukça küçük tutmak öğrenme esnasında paketlerin azar azar işlenmesini sağlamaktadır. Bu durum çoğu zaman eğitim için faydalı olsa da mini-batch değerinin çok küçük seçilmesi aşırı öğrenmeye neden olmaktadır. Küçük mini-batch değeri ayrıca süre ve işlem hacmi olarak da olumsuz etki göstermektedir.

Önceden eğitilmiş hazır ağların yardımıyla transfer öğrenmesi yapmak, uzun süre eğitim yapılması gereken ağların daha kısa sürede iyi sonuçlar vermesini sağlamaktadır. Transfer öğrenmesinde en iyi sonuçları alabilmek için önceden eğitilmiş hazır ağda kullanılan veri setine benzer sınıflar içeren veri setleri ile çalışma yapılmalıdır. Eğer benzer veri seti kullanılacaksa bütün ağırlıklar dondurulabilir. Eğer içerik olarak farklı bir veri seti kullanılacaksa ağırlıkların sadece küçük bir bölümü dondurulur. Kalan ağırlıklar eğitim esnasında güncelleştirilerek iyileştirilmesi sağlanabilir.

Bir ağın çok derin olması çok daha iyi başarım elde edilebileceği anlamına gelmiyor. Çok derin katmanlı ağlar da aşırı öğrenmeye neden olmaktadır. Yaş sınıflandırmada bunun örneğini verebiliriz. ResNet-101 katman olarak ResNet-50'den daha derin olmasına rağmen, ResNet-50'nin başarı ortalaması ResNet-101'e göre daha iyidir.

Veri artırımı (data augmentation) yaparak başarı oranının arttırılması hedeflenmiştir. Biz veri artırımını, resimleri (-30 30) dereceler arasında çevirerek ve x – eksenine göre simetriğini alarak gerçekleştirdik. Ayrıca transfer öğrenmesi dışındaki modellerimizde

yaptığımız testlerde 256x256x3 boyutundaki giriş resimlerimizi rastgele olarak 227x227x3 boyutunda kırpıyoruz. Bunu her epochta tekrarlıyoruz. Böylece her epochta farklı bir resim eğitime dahil edilmiş oldu.

Oy çokluğu yöntemi, bir modelde yanlış tahmin edilen imgenin başka modellerde doğru tahmin edilmiş olmasıyla yanlış tahminin düzeltilebilmesini sağlıyor. Böylece yanlış tahmin edilen imgelerin sayısı azalıyor. Sonuçlar karşılaştırıldığında oy çokluğu yönteminin başarı oranı tüm testlerdeki başarı oranlarından daha yüksek.

Yaş ve cinsiyet sınıflandırma bir biri ile karşılaştırıldığında cinsiyet sınıflandırmanın yaş sınıflandırmaya oranla daha yüksek başarım oranına sahip olduğu gözlenmektedir. Ancak testler foldlara göre incelendiğinde bebek ve küçük çocuk resimlerinin veri setinin içinde var olması cinsiyet sınıflandırmayı zorlaştıran en önemli faktördür. Bu gruplarda cinsiyet sınıflandırma için özellik çıkarımı oldukça zordur. Yaş sınıflandırmada ise kadınların yaş grubunu tahmin etmek erkeklere göre daha zordur. Makyaj, saç gibi etmenler kadınların yaş tahminini zorlaştırmaktadır.

KAYNAKLAR

Akbulut, Y., Şengür, A., & Ekici, S. (2017). Gender recognition from face images with deep learning. *2017 International Artificial Intelligence and Data Processing Symposium (IDAP).*

Amidi, A., & Amidi, S. (2018). Stanford.edu: https://stanford.edu/~shervine/l/tr/teaching/cs-230/cheatsheet-convolutional-neural-networks#filter adresinden alındı

Arnulf , B., & Borer, S. (2001). Normalization in Support Vector Machines. *Lecture Notes in Computer Science (LNCS)*, 277–282.

Altuntaş, E., & Naneli, İ. (2017). Beyaz ve siyah kinoa tohumlarının geometrik,hacimsel ve sürtünme özellikleri. *Gaziosmanpaşa Bilimsel Araştırma Dergisi*, 1-8.

Ayhan, S., & Erdoğmuş, Ş. (2014). Destek Vektör Makineleriyle Sınıflandırma Problemlerinin Çözümü İçin Çekirdek Fonksiyonu Seçimi. *Eskişehir Osmangazi Üniversitesi İİBF Dergisi*, 175- 198.

Beale, M., Hagan, M., & Demuth, H. (2018). *Deep Learning Toolbox™ User's Guide.* MathWorks.

Çarkacı , N. (2019, Temmuz 11). *Derin Öğrenme Uygulamalarında En Sık kullanılan Hiper-parametreler.* Medium: https://medium.com/deep-learning-turkiye/derin-ogrenme-uygulamalarinda-en-sik-kullanilan-hiper-parametreler-ece8e9125c4 adresinden alındı

Dumoulin, V., & Visin, F. (2018). *A guide to convolution arithmetic for deep learning.*

Eidinger , E., Enbar , R., & Hassner, T. (2014). Age and Gender Estimation of Unfiltered Faces. *IEEE Transactions on Information Forensics and Security*, 2170 - 2179.

Ekmekji, A. (2016). Convolutional Neural Networks for Age and Gender Classification. *Stanford University*.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning (Adaptive Computation and Machine Learning Series).* MIT Press.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Identity Mappings in Deep Residual Networks. *ECCV 2016*.

Hsu, C.-W., Chang, C.-C., & Lin, C.-J. (2016). *A Practical Guide to Support Vector Classification.* Taipei: National Taiwan University.

Hosseini, S., Lee, S. H., Kwon, H. J., Koo, H. I., & Cho, N. I. (2018). Age and Gender Classification Using Wide Convolutional Neural Network and Gabor Filter. *2018 International Workshop on Advanced Image Technology.* Chiang Mai: IEEE.

İnik, Ö., & Ülker, E. (2017). Derin Öğrenme ve Görüntü Analizinde Kullanılan Derin Öğrenme Modelleri. *Gaziosmanpaşa Bilimsel Araştırma Dergisi, 6*(3), 85-104.

Jones, M., & Viola , P. (2001). Robust real-time face detection. *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV*.

Kavzoğlu, T., & Çölkesen, İ. (2010). (Investigation of the Effects of Kernel Functions in Satellite Image Classification Using Support Vector Machines. *Harita Dergisi*.

Kingma, D., & Ba, J. (2015). Adam: A Method for Stochastıc Optımızatıon. *International Conference on Learning Representations ICLR.* San Diego.

Kohavı , R., & Provost, F. (1998). Confusion matrix. *Machine Learning*, 271–274.

Kutlu, H. (2018). *Biyoistatik Temelli Bilimsel Araştırmalarda Derin Öğrenme Uygulamaları.* Lefkoşa: Yakın Doğu Üniversitesi Sağlık Bilimleri Enstitüsü.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep Learning. *Nature*, 436-444.

Levi, G., & Hassner, T. (2015). Age and gender classification using convolutional neural networks. *2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*.

Liu, X., Li , J., Pan, J.-S., & Hu, C. (2017). Deep convolutional neural networks-based age and gender classification with facial images. *2017 First International Conference on Electronics Instrumentation & Information Systems (EIIS)*.

Matlab Documentation. (2019). MathWorks.

Osowski, S., Siwek, K., & Markiewicz, T. (2004). MLP and SVM Networks – a Comparative Study. *Proceedings of the 6th Nordic Signal Processing Symposium - NORSIG 2004.* Espoo.

Refaeılzadeh, , P., Tang, L., & Lıu, H. (2009). Cross-Validation. *Encyclopedia of Database Systems* (s. 532-538). içinde

Rodríguez, P., Cucurull , G., Gonfaus, J., Roca, F., & Gonzàlez, J. (2017). Age and gender recognition in the wild with deep attention. *Pattern Recognition*, 563-571.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *ArXiv e-prints*.

Sharma, A. (tarih yok). *Confusion Matrix in Machine Learning.* GeeksforGeeks: https://www.geeksforgeeks.org/confusion-matrix-machine-learning/ adresinden alındı

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever , I., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research* , 1929-1958.

Tivive , F., & Bouzerdoum, A. (2006). A Gender Recognition System using Shunting Inhibitory Convolutional Neural Networks. *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, 5336-5341.

Torrey, L., & Shavlik, J. (2009). *Transfer Learning.* University of Wisconsin.

Vapnik, V., & Cortes, C. (1995). *Support Vector Networks.*

Wolfshaar, J., Karaaba, M., & Wiering, M. (2015). Deep Convolutional Neural Networks and Support Vector Machines for Gender Recognition. *2015 IEEE Symposium Series on Computational Intelligence.* Cape Town.

Zhang, K., Gao, C., Guo, L., Sun, M., Yuan, X., Han, T., . . . LI, B. (2017). Age Group and Gender Estimation in the Wild With Deep RoR Architecture. *IEEE Access* , 22492 - 22503.

Zhang, K., Sun, M., Han, T., Yuan, X., Guo, L., & Liu, T. (2016). Residual Networks of Residual Networks: Multilevel Residual Networks. *IEEE Transactıons on Latex Class Fıles*.

ÖZGEÇMİŞ

1987 yılında Erzurum'da dünyaya geldi. İlkokul, ortaokul, lise ve üniversiteyi Erzurum'da okudu. 2006 yılında Atatürk Üniversitesi Mühendislik Fakültesi Elektrik-Elektronik Mühendisliği bölümüne başladı. 2010 yılında üniversiteden mezun oldu. 2011 yılında iş hayatına başladı. Sırası ile TEK-DAĞ+MAG+BİL-GÜN İş Ortaklığı, Alcatel-Lucent ve Nokia firmalarında bölge destek mühendisi olarak çalıştı. Haziran 2019'dan beri Ericsson firmasında bölge destek mühendisi olarak çalışmakta. Evli ve 1 kız çocuğu babasıdır.