T.C.
YOZGAT BOZOK ÜNİVERSİTESİ
MÜHENDİSLİK MİMARLIK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

PNEUMONIA HASTALIĞI TEŞHİSİNDE MAKİNE ÖĞRENİMİ TEKNİKLERİ

BİTİRME ÖDEVİ

ESRA YÜCE

2020-2021 GÜZ/BAHAR DÖNEMİ

T.C.
YOZGAT BOZOK ÜNİVERSİTESİ
MÜHENDİSLİK MİMARLIK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

PNEUMONIA HASTALIĞI TEŞHİSİNDE MAKİNE ÖĞRENİMİ TEKNİKLERİ

BİTİRME ÖDEVİ

ESRA YÜCE

2020-2021 GÜZ/BAHAR DÖNEMİ

ÖNSÖZ

Bu çalışma, göğüs röntgeni görüntü örneklerinden oluşan bir veri setinden pneumonia'nın varlığını sınıflandırmak ve tespit etmek için sıfırdan eğitilmiş bir evrişimli sinir ağı modeli geliştirilmiştir.

Çalışma konusunun belirlenmesinde ve çalışmanın hazırlanma sürecinin her aşamasında bilgilerini, tecrübelerini ve değerli zamanlarını esirgemeyerek bana her fırsatta yardımcı olan değerli hocam Sayın Dr. Öğretim Üyesi Emre ÖLMEZ'e teşekkürü bir borç bilirim. Ayrıca çalışmalarım boyunca maddi manevi destekleriyle beni hiçbir zaman yalnız bırakmayan aileme ve arkadaşlarıma da sonsuz teşekkürler ederim.

İçindekiler

ÖNSÖZ i

ŞEKİL LİSTESİ iv

TABLO LİSTESİ v

KISALTMA LİSTESİ v

ÖZET vi

SUMMARY vii

1. GİRİŞ 1

2. LİTERATÜR TARAMASI 3

3. VERİ SETİ 4

4. YÖNTEM 4

5. DERİN ÖĞRENME 5

5.1. EVRİŞİMSEL SİNİR AĞLARI (CONVOLUTIONAL NEURAL NETWORKS) 5

5.1.1. Giriş Katmanı (Input Layer) 6

5.1.2. Evrişimli Katman (Convolutıon Layer) 6

5.1.3. Aktivasyon (ReLu) Katmanı (Activation (ReLu) Layer) 7

5.1.4. Havuzlama Katmanı (Pooling Layer) 7

5.1.5. Tam Bağlantılı Katman (Fully Connected Layer) 8

5.2. EVRİŞİMSEL SİNİR AĞI UYGULAMALARI 8

5.3. DERİN ÖĞRENME MİMARİLERİ 9

5.3.1. LeNet Mimarisi 10

5.3.2. AlexNet Mimarisi 10

5.3.3. ZFNet Mimarisi 11

5.3.4. GoogLeNet Mimarisi 12

5.3.5. VGGNet Mimarisi 13

5.3.6. Microsoft ResNet Mimarisi 14
5.3.7. MobileNet 15
6. CNN İLE EĞİTİM MODELİ 16
6.1. MODEL 1 17
6.1.1. Yöntem 1 19
6.1.2. Yöntem 2 21
6.2. MODEL 2 22
6.2.1. Yöntem 1 24
6.2.2. Yöntem 2 26
7. SONUÇ VE TARTIŞMA 28
8. KAYNAKLAR 30
9. EKLER 34
10. ÖZGEÇMİŞ 41

ŞEKİL LİSTESİ

Şekil 1. Evrişimsel Sinir Ağı Yapısı 6
Şekil 2. Çekirdekte kıvrımlı girdi matrisinin Evrişim İşlemi 7
Şekil 3. Havuzlama 8
Şekil 4. LeNet'te veri akışı. Giriş el yazısı bir rakamdır, çıktı ise 10 olası sonucun üzerinde bir olasılıktır. 10
Şekil 5. AlexNet Mimarisi 11
Şekil 6. ZFNet Mimarisi 12
Şekil 7. GoogleNet Mimarisi 13
Şekil 8. VGGNet Mimarisi 13
Şekil 9. Microsoft ResNet Mimarisi 14
Şekil 10. Residual blok 15
Şekil 11. MobileNet Mimarisi 15
Şekil 12. (a) sağlıklı bir kişinin ve (b) pneumonia hastalığı taşıyan bir kişinin göğüs röntgeni. 16
Şekil 13. Model 1 katmanları. 18
Şekil 14. Modelin eğitim işlemi sonucu 19
Şekil 15. Modele ait doğruluk oranı 19
Şekil 16. Modele ait kayıp oranı 20
Şekil 17. Modele ait özet bilgiler 20
Şekil 18. Modelin eğitim işlemi sonucu 21
Şekil 19. Modele ait doğruluk oranı 21
Şekil 20. Modele ait kayıp oranı 22
Şekil 21. Modele ait özet bilgiler 22
Şekil 22. Model 2 katmanları 23
Şekil 23. Modelin eğitim işlemi sonucu 24
Şekil 24. Modele ait doğruluk oranı 24
Şekil 25. Modele ait kayıp oranı 25
Şekil 26. Modele ait özet bilgiler 25
Şekil 27. Modelin eğitim işlemi sonucu 26
Şekil 28. Modele ait doğruluk oranı 26

Şekil 29. Modele ait kayıp oranı 27
Şekil 30. Modele ait özet bilgiler 27

TABLO LİSTESİ

Tablo 1. Mimarilerin Karşılaştırlması.....16
Tablo 2. Model karşılaştırması.....29

KISALTMA LİSTESİ

CT (BT) : Computed Tomography (Bilgisayarlı Tomografi)

MRG - MRI: Magnetic Resonance İmaging (Manyetik Rezonans Görüntüsü)

CNN: Convolutional Neural Network (Evrişimsel Sinir Ağları)

F-RCNN: Fast Region Based Convolutional Neural Network (Hızlı Bölge Tabanlı Evrişimli Sinir Ağı)

FCN: Fully Convolutional Network (Tam Evrişimli Ağ)

R-CNN: Region Based Convolutional Neural Network (Bölge Tabanlı Evrişimli Sinir Ağı)

MNIST: Modified National Institute of Standards and Technology (Modifiye Ulusal Standartlar ve Teknoloji Enstitüsü)

ILSVRC: ImageNet Large Scale Visual Recognition Challenge (ImageNet Büyük Ölçekli Görsel Tanıma Mücadelesi)

GPU: Graphics Processor Unit (Grafik İşlemci Birimi)

RGB: Red Green Blue (Kırmızı Yeşil Mavi)

LRN: Local Response Normalization (Yerel Yanıt Normalleştirme)

CONV: Convolutional (Evrişimli)

ÖZET

Pneumonia, her yıl yaklaşık olarak 700.000 çocuğun ölümüne neden olur ve dünya nüfusunun % 7'sini etkiler. Tıbbi olarak pneumonia teşhisini kesin olarak konulabilmesi için akciğer röntgen görüntülerinin bir radyolog tarafından incelenmesi gereklidir. Bununla birlikte, eğitimli bir radyolog için bile göğüs röntgenlerini incelemek zor bir görevdir. Bu çalışmada, radyologlara karar verme süreçlerinde yardımcı olabilecek pneumonia tespiti için etkili bir model önerilmiştir. Hastalığın teşhisinde geliştirilen tanıma sistemi için erişime açık olan akciğer röntgen görüntülerinden faydalanılmıştır.

Anahtar Kelimeler: pneumonia, göğüs röntgeni görüntüleri, evrişim sinir ağı (CNN), derin öğrenme, transfer öğrenmesi, bilgisayar destekli teşhis

SUMMARY

Pneumonia causes the death of approximately 700,000 children each year and affects 7% of the world's population. In order to make a definite medical diagnosis of pneumonia, lung x-ray images should be examined by a radiologist. However, examining chest X-rays is a difficult task, even for a trained radiologist. In this study, an effective model for pneumonia detection is proposed that can assist radiologists in their decision making process. For the diagnosis system developed in the diagnosis of the disease, accessible lung x-ray images were used.

Keywords: pneumonia, chest X-ray images, convolution neural network (CNN), deep learning, transfer learning, computer-aided diagnostics

1. GİRİŞ

Görüntü işleme, insan gözünün yaptığı işlevleri bilgisayar ortamında çeşitli ara yüz yazılımlarıyla hızlı sonuçlar elde eden bir teknolojidir [1]. Bu teknolojide çeşitli modeller geliştirilmiştir. Geliştirilen bu modeller ile yapılan bilimsel çalışmalar da katkıda bulunulmuştur [2]. Son yıllarda, yapılan bu analiz sonuçlarında en çok tercih edilen model, makine öğrenmesinin bir parçası olan derin öğrenme modelidir. Makine öğrenmesi yöntemine kıyasla çok katmanlı bir yapıya sahip olan derin öğrenme, insan beyninin işleyişinden esinlenerek, son zamanlarda oldukça artan bir ilgi görmektedir [3]. Derin öğrenme modellerinin, görüntü işleme üzerinde yoğunlaştığı alanlardan biri de biyomedikal uygulamalardır. Özellikle bu alanlarda elde edilen biyomedikal imgeler üzerinde, derin öğrenme modellerinin uygulanması sonucu oldukça yüksek başarılar elde edilmiştir [4,5].

Geçmişten günümüze enfeksiyon hastalıkları, insan sağlığını tehdit eden en önemli unsurlardan biridir. Enfeksiyon hastalıklarının ilk sırasında ise latince tıbbi terim olarak pneumonia adı verilen zatürre hastalığı gelmektedir [6]. Zatürre, virüs ve bakteri gibi canlıların mikroskobik hava keselerini etkilemesi sonucu oluşan akciğer iltihaplanması olarak tanımlanır [7,9]. Esas olarak iki tür pneumonia vardır: bakteriyel ve viral. Genel olarak bakteriyel pneumonianın daha akut semptomlara neden olduğu görülmektedir. Bakteriyel ve viral pneumonia arasındaki en önemli fark tedavidir. Bakteriyel pneumonia tedavisi antibiyotik tedavisi kullanılarak yapılırken viral pneumonia genellikle kendi kendine iyileşir.

Her yıl dünya nüfusunun yaklaşık % 7'si zatürre hastalığından etkilenir ve etkilenen hastaların 4 milyon kadarı ölümle sonuçlanır [8]. Bu tip hastalıklarda erken tanı önemlidir [10]. Tipik belirtiler arasında göğüs ağrısı, nefes darlığı, öksürük vs. yer alır. Tanı araçları arasında balgam kültürü ve göğüs röntgeni görüntüleri bulunmaktadır [11].

Pneumonia tüm dünyada yaygın bir hastalıktır. Başlıca nedeni yüksek düzeyde kirliliği içerir. Pneumonia, Amerika Birleşik Devletleri'ndeki ilk 10 ölüm nedeni listesinde sekizinci sırada yer almaktadır [3]. Pneumonia nedeniyle, Hindistan'da her yıl 3,7 bin çocuk ölmektedir ve bu, Hindistan'da meydana gelen pneumonia ölümlerinin toplam yüzde ellisini oluşturmaktadır. Hastalık, özellikle yaşlı hastalarda ölümcül bir noktaya gelene kadar sıklıkla gözden kaçar ve tedavi edilmez. Dünya çapında çocuklarda (özellikle beş yaş altı) tek en büyük ölüm nedenidir [4]. Dünya Sağlık Örgütü'ne göre, "Her yıl, beş yaşın altındaki tahmini 1,4 milyon çocuğu öldürüyor ve bu,

dünya çapında beş yaşın altındaki çocukların tüm ölümlerinin % 18'ini oluşturuyor. Pneumonia, çocukları ve aileleri her yerde etkiler ancak en çok Güney Asya ve Sahra altı Afrika'da görülür. Çocuklar zatürreden korunabilir. Basit müdahalelerle önlenebilir ve düşük maliyetli, düşük teknolojili ilaç ve bakım ile tedavi edilebilir" [2]. Bu nedenle, özellikle çocuklarda pneumoniaya bağlı ölümlerin azaltılabilmesi için bilgisayar destekli tanı konusunda önemli işlere imza atılabilir.

Pneumonia tanısı için yandaki testler yapılabilir: Göğüs röntgeni, Akciğer BT'si, Göğüs ultrasonu, Akciğer iğne biyopsisi Göğüs MRG'si [5]. Günümüzde akciğer röntgeni pneumonianın saptanması için en iyi yöntemlerden biridir [6]. CT görüntülemesi tipik olarak X-Ray görüntülemeden önemli ölçüde daha fazla zaman aldığından ve pek çok gelişmemiş bölgede yeterli yüksek kaliteli CT tarayıcıları bulunmadığından, CT görüntülemeye göre X-Ray görüntüleme tercih edilir. Aksine, X-ışınları en yaygın olarak bulunan tanısal görüntüleme tekniğidir. Klinik bakım ve epidemiyolojik çalışmalarda çok önemli bir rol oynamaktadır. Dünyada, bu tür hastalıklarla ilgili öngörüleri büyük ölçüde önemli olan pratisyen sağlık çalışanları ve radyologların az olduğu birkaç bölge vardır. Yapay zeka tabanlı çözümler kullanan bilgisayar destekli teşhis, günümüzde giderek daha popüler hale geliyor. Bu teşhis, minimum maliyetle büyük bir nüfus için kullanılabilir hale getirilebilir. Bu hastalıkla ilgili bir başka sorun da, bazen hastalığın varlığını tanımlayan özelliklerin sıklıkla diğer hastalıklarla karışması ve bu nedenle radyologların bu hastalığı teşhis etmeyi zor bulmasıdır. Tıp alanında doktorların göğüs röntgenlerine bakıp teşhis koyması zaman alan bir süreçtir. Mevcut teknolojik araçlar ve yazılımlardan yararlanılarak teşhisin yapılması zaman ve maliyet açısından son derece olumlu bir gelişmedir. Derin öğrenme modelleri, pneumonia hastalarından elde edilen göğüs röntgeni görüntüleriyle geleneksel yöntemlere göre daha verimli sonuçlar sağlamıştır.

Bu çalışmada, hastanın pneumonia olup olmadığını sınıflandırabilen derin öğrenme ve evrişimli sinir ağları(CNN) uygulamalarına dayalı bir model sunulacaktır. Önerilen metodoloji, hastalığın varlığını tanımlayan ve bunun bir pneumonia vakası olup olmadığını raporlayan X-Ray görüntüsünden özellikleri çıkaran derin transfer öğrenme algoritması kullanır.

2. LİTERATÜR TARAMASI

Roth vd. [12], derin evrişimli sinir ağının (CNN) klinik tanı görevinde lenf düğümünü saptama gücünü göstermiş ve bilgisayarlı tomografiden elde edilen düşük kontrastlı çevre yapılarının varlığında bile iyi sonuçlar elde etmiştir.

Başka bir çalışmada Shin ve ark. [13], derin CNN kullanarak torako-abdominal lenf tespiti ve interstisyel akciğer hastalığı sınıflandırması sorunlarını ele aldı. Farklı CNN mimarileri geliştirdi ve hasta başına üç yanlış pozitifte % 85 hassasiyetle umut verici sonuçlar elde etti.

Ronneburger vd. [14], veri artırmanın kullanımıyla bir CNN yaklaşımı geliştirdi. İletilen ışık mikroskobundan elde edilen görüntü verilerinin küçük örnekleri üzerinde bile eğitilmiş olduklarını öne sürdü; geliştirilen model yüksek doğruluk yakalamayı başardı.

Jamaludin vd. [15], spinal kereste manyetik rezonans görüntülemesinden (MRI) elde edilen verileri analiz etmek için CNN mimarisini uyguladı. Omurga kereste MRI'larının radyolojik derecelendirmesini oluşturmak için etkili bir CNN modeli geliştirdi.

Verilerin boyutunun birkaç yüz hasta örneğiyle sınırlı olması dışında, tüm bu çalışmalar radyolojik veriler üzerinde iyi performans göstermiştir. Bu nedenle, doğru ve güvenilir tahminlere ulaşmak için binlerce hasta örneğinin üzerinde derin öğrenmenin gücünü kullanmak için detaylı bir çalışma gereklidir.

Kallianos vd. [16], göğüs röntgeni görüntü sınıflandırma ve analizinde yapay zekanın önemini belirten son teknoloji bir inceleme sundu. Wang vd. [17] bu sorunu ele aldı ve 32.717 benzersiz hastanın 108.948 önden görünüm X-Ray görüntülerini içeren yeni bir veritabanı ChestX-ray8 hazırladı. Röntgen görüntülerinin her biri birden fazla etikete sahip olabilir. Bu verilerdeki sonuçları doğrulamak için derin evrişimli sinir ağları kullandılar ve umut verici sonuçlar elde ettiler. ChestX-ray8 veritabanının daha fazla hastalık sınıfı dâhil edilerek genişletilebileceğini ve diğer araştırma çalışmaları için faydalı olacağını belirttiler.

Rajpurkar vd. [18], 121 katmanlı derin evrişimli katman ağ ChestX-ray14 veri seti geliştirdi. Bu veri seti, 14 hastalık etiketli 0,1 milyondan fazla önden görünüm X-Ray görüntüsü ile halka açıktır. Algoritmalarının 14 hastalık kategorisinin tamamını yüksek verimlilikle tahmin edebildiğini belirttiler. Irvin vd. [19] büyük etiketli veri setinin tahmin ve sınıflandırma görevleri

için başarının anahtarı olduğunu belirtti. 65.240 hastanın 224.316 göğüs radyografik görüntüsünden oluşan devasa bir veri seti sundular. Bu veri setini CheXpert olarak adlandırdılar. Daha sonra, model tarafından atanan olasılığa dayalı olarak etiketler atamak için evrişimli sinir ağlarını kullandılar. Model, her bir gözlemin olasılıklarının çıktısını almak için ön ve yan radyografları kullandı. Ayrıca, veri kümesini bir kıyaslama veri kümesi olarak yayınladılar. Büyük bir veri setinin mevcudiyetinin yanı sıra, görüntüdeki her nesnenin dikkatlice tespit edilmesi ve her bir örneğin segmentasyonunun hassas bir şekilde yapılması son derece arzu edilir. Bu nedenle, hem örnek segmentasyonu hem de nesne algılamayı işlemek için farklı bir yaklaşım gereklidir. Bu tür güçlü yöntemler daha hızlı bölge tabanlı CNN'dir (F-RCNN)[20] ve FCN (Tam Evrişimli Ağ) [21].

Ayrıca, F-RCNN, sınıflandırma görevi için mevcut dalların yanı sıra ilgili her bölgede bölümleme maskesi tahmini için ek bir dal ile genişletilebilir. Bu genişletilmiş ağa Mask R-CNN adı verilir ve verimlilik ve doğruluk açısından F-RCNN'den daha iyidir. Kaiming He vd. [22], nesne örnek segmentasyonu için Maske R-CNN yaklaşımını sundu. Elde ettikleri sonuçları COCO 2016'daki en iyi modellerle [23, 24] karşılaştırdılar. Luc vd. [25], evrişimli özellikleri tahmin ederek bir eşgörünüm seviyesinde bölümleme sunarak yaklaşımlarını genişletti.

3. VERİ SETİ

Veri seti, test ve train(eğitim) olmak üzere iki klasör halindedir. Eğitim seti klasörü, pneumonia vakaları için bir görüntü klasörü ve normal vakalar için bir görüntü klasörü içerir. Eğitim seti toplam 5216 görüntüden oluşmaktadır. Test seti klasörü, pneumonia vakaları için bir görüntü klasörü ve normal vakalar için bir görüntü klasörü içerir. Test seti, toplam görüntü setinin yaklaşık % 10,68'i olmak üzere toplam 624 resimden oluşmaktadır.

Veri seti adresi: https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia

4. YÖNTEM

Elde edilen veri kümesinden öznitelik çıkarımı için derin öğrenme modellerinden Evrişimsel Sinir Ağı (CNN) kullanılmıştır. Hastalığın teşhisi için elde edilen öznitelikler farklı sınıflandırıcılar kullanılarak başarım karşılaştırmaları yapılacaktır.

5. DERİN ÖĞRENME

Derin öğrenme bir makine öğrenmesi sınıfıdır. Derin öğrenme, özellik çıkarma ve dönüştürme için birçok doğrusal olmayan işlem birimi katmanını kullanır. Her ardışık katman, önceki katmandaki çıktıyı girdi olarak alır [26]. Algoritmalar denetimli (sınıflandırma gibi) veya denetimsiz (desen analizi gibi) olabilir. Derin öğrenmede, verilerin birden fazla özellik seviyesinin veya temsillerinin öğrenilmesine dayanan bir yapı söz konusudur. Üst düzey özellikler, alt düzey özelliklerden türetilerek hiyerarşik bir temsil oluşturur. Bu temsil, soyutlamanın farklı seviyelerine karşılık gelen birden çok temsil seviyesini öğrenmektedir [27]. Derin öğrenme temel olarak verinin temsilinden öğrenmeye dayalıdır. Bir görüntü için temsil denildiğinde; piksel başına yoğunluk değerlerinin bir vektörü veya kenar kümeleri, özel şekiller gibi özellikler düşünülebilir. Bu özelliklerin içinden bazıları veriyi daha iyi temsil etmektedir. Bu aşamada yine bir avantaj olarak, derin öğrenme yöntemleri, elle çıkarılan özellikler (handcrafted features) yerine veriyi en iyi temsil eden hiyerarşik özellik çıkarımı için etkin algoritmalar kullanmaktadır [28].

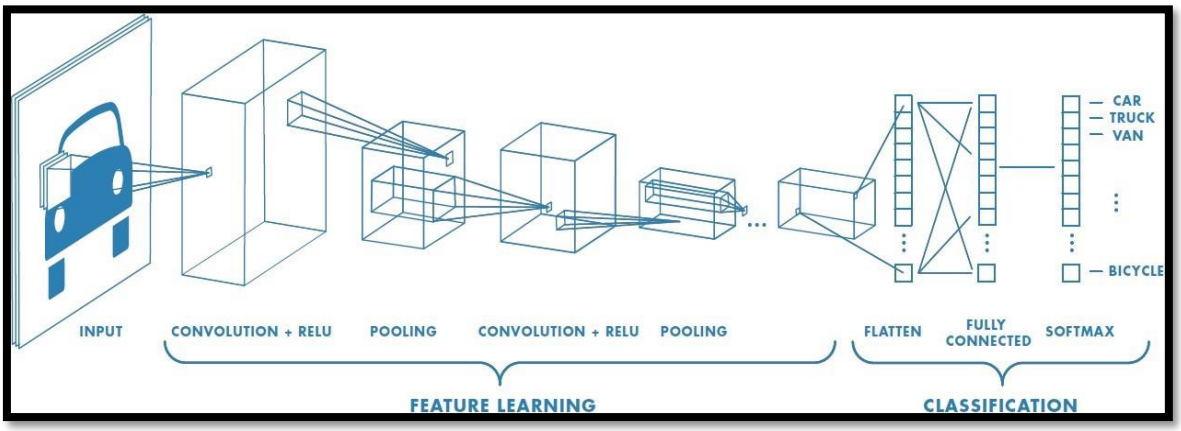
5.1. EVRİŞİMSEL SİNİR AĞLARI (CONVOLUTIONAL NEURAL NETWORKS)

Evrişimsel sinir ağları kısaca bir görüntünün analizini yapmamızı sağlar. Görüntüdeki çeşitli obje veya nesneleri birbirinden ayrıştırmamıza yardımcı olan derin öğrenme algoritmasıdır. İleri yönlü bir sinir ağı olan CNN, hayvanların görme merkezinden esinlenilerek ortaya çıkmıştır. Evrişimsel işlemler, bir nöronun kendi uyarı alanından uyaranlara verdiği cevap olarak düşünülebilir [29]. Basit olarak ifade etmemiz gerekirse evrişim olayı matematik, fizik veya mühendislik uygulamalarında karmaşık işlemleri basitleştirmek için kullanılan matematiksel işlemlerdir. Çalışmaların başında görüntü ve ses işleme, doğal dil işleme ve biyomedikal gibi alanlarda kullanılmıştır fakat en iyi sonuçları görüntü işleme alanında vermiştir.

Konvolüsyonel Sinir Ağları, bir veya daha fazla konvolüsyonel katmanlardan, altörnekleme katmanından ve standart çok katmanlı bir sinir ağı gibi bir veya daha fazla bağlı katmanlardan oluşur. CNN'lerin avantajı ise, aynı miktarda gizli birimle tamamen bağlı ağlardan daha az sayıda eğitime ve daha az parametreye gerek duymalarıdır.

CNN mimarileri oluşturmak için belirli katman türleri vardır. Bunlar:

- Giriş Katmanı (Input Layer)
- Evrişimli Katman (Convolutıon Layer)
- Aktivasyon (ReLu) Katmanı (Activation (ReLu) Layer)
- Havuzlama Katmanı (Pooling Layer)
- Tam Bağlantılı Katman (Fully Connected Layer)



Şekil 1. Evrişimsel Sinir Ağı Yapısı

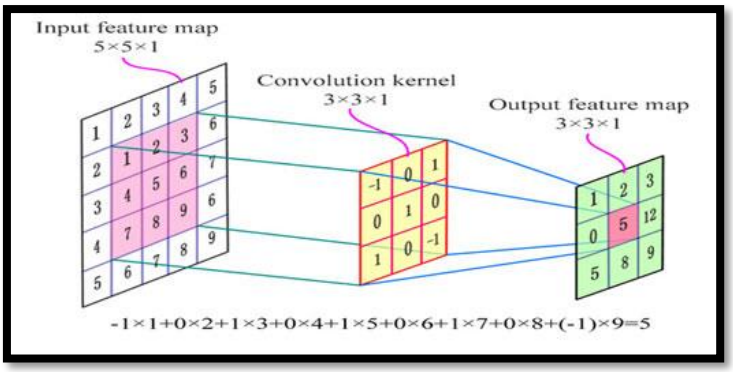
5.1.1. Giriş Katmanı (Input Layer)

Bu katman isminden de anlaşılacağı üzere ESA'ın ilk katmanını oluşturmaktadır. Bu katmanda veri ham olarak ağa verilmektedir. Tasarlanacak modelin başarımı için bu katmandaki verinin boyutu önem kazanmaktadır. Giriş görüntü boyutunun yüksek seçilmesi hem yüksek bellek ihtiyacını hem eğitim süresini hem de görüntü başına düşen test süresini uzatabilir. Bunun yanında ağ başarısını arttırabilir. Giriş görüntü boyutunun düşük seçilmesi bellek ihtiyacını azaltır ve eğitim süresini kısaltır. Fakat kurulacak ağın derinliği azalır ve performansı düşük olabilir. Görüntü analizinde hem ağ derinliği hem donanımsal hesaplama maliyeti hem de ağ başarısı için uygun bir giriş görüntü boyutu seçilmelidir.

5.1.2. Evrişimli Katman (Convolutıon Layer)

CNN'nin ilk adımı, her bir girişin bir öğrenme sürecinden geçtiği evrişim katmanıdır. Bu katman esas olarak çekirdekler olarak bilinen seri filtreden oluşur. Kernal, genellikle ağırlıklar olarak bilinen değerler kümesinden oluşan dikdörtgen matris olarak ifade edilir. Bu ağırlıklar giriş verilerinden [27] öğrenilir ve bir hata geri yayılma algoritması [28] ile otomatik olarak

hesaplanır. Şekil 2'de gösterildiği gibi, her filtre, giriş verileri ve bunların özellik haritası üreterek dönüştürülür. Bir Dönüşüm katmanının amacı, giriş hacminin özelliklerini çıkarmaktır.



Şekil 2. Çekirdekte kıvrımlı girdi matrisinin Evrişim İşlemi

5.1.3. Aktivasyon (ReLu) Katmanı (Activation (ReLu) Layer)

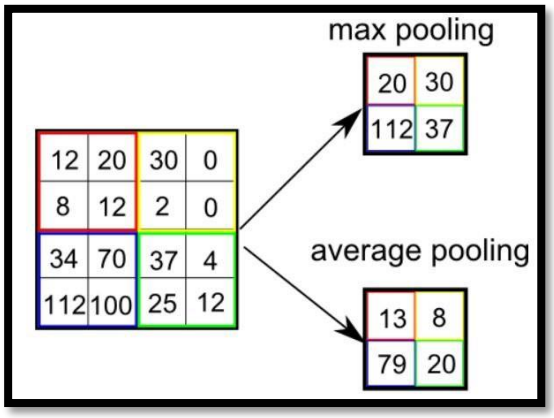
Evrişim katmanının ardından Relu katmanı gelir. Bu katmanda, evrişim katmanının çıktısına doğrusal olmayan bir etkinleştirme işlevi uygulanır. Relu aktivasyonu, genellikle derin sinir ağında ortaya çıkan gradyan kaybolma problemini ve aşırı uyum problemini önlemede de yardımcı olur.

5.1.4. Havuzlama Katmanı (Pooling Layer)

Havuzlama katmanı, özellik haritasının boyutsallığını altörnekleme veya azaltmak için kullanılır. Havuzlamada kullanılan, maksimum havuzlama veya ortalama havuzlama olan iki tür yöntem vardır.

Maksimum havuzlama, filtrenin kapsadığı özellik haritasının bölgesinden maksimum öğeyi seçen bir havuzlama işlemidir. Dolayısıyla, maksimum havuz katmanından sonraki çıktı, önceki özellik haritasının en belirgin özelliklerini içeren bir özellik haritası olacaktır.

Ortalama havuzlama, filtrenin kapsadığı özellik haritası bölgesinde bulunan öğelerin ortalamasını hesaplar. Bu nedenle, maksimum havuz oluşturma, özellik haritasının belirli bir yamasında en belirgin özelliği verirken, ortalama havuzlama, bir yamada bulunan özelliklerin ortalamasını verir.



Şekil 3. Havuzlama

5.1.5. Tam Bağlantılı Katman (Fully Connected Layer)

CNN'in son aşaması Tam bağlantı katmanıdır, bu katman son öğrenme aşaması olarak kullanılır. Normalde bu katman, CNN'nin çıktı verilerinin özelliklerini çıkarmak ve ayrıca özellik çıkarma aşamalarını Softmax sınıflandırıcıyla bağlamak için kullanılır. Tamamen bağlı bir CNN normalde 2 ila 3 katmandan oluşur ve ileri beslemeli bir sinir ağına bağlıdır. Tamamen bağlı katmanın son çıktısı 2 boyutlu diziden 1 boyutlu diziye dönüştürülür. Bu 1-D dizinin her bir öğesi, bir girdi görüntüsünün sınıf türünün belirlenmesi için kullanılan parametredir.

5.2. EVRİŞİMSEL SİNİR AĞI UYGULAMALARI

CNN uygulamaları daha çok görüntü işleme üzerine yapılmaktadır. Fakat ses algılama gibi doğal dil işleme işlemleriyle birlikte biyomedikal birçok farklı alanda çalışılmaktadır. Özellikle görüntü işleme alanında state-of-art sonuçlar elde edilmiştir. MNIST veri kümesi üzerinde, Cireşan yaptığı çalışmada, CNN ile hata oranı % 2'lere kadar düşürmeyi başarmışlardır. 2015'te çok katmanlı bir CNN, ters yüzler de dâhil olmak üzere geniş açı aralıklarındaki yüzleri yakalayabilme yeteneğini göstermiştir. CNN algoritmaları ilaç keşfinde de kullanılmıştır. 2015 yılında Atomwise şirketinin geliştirdiği AtomNet, ilaç tasarımı için geliştirilen ilk derin sinir ağı olmuştur. Kısaca özetlemek gerekirse CNN Uygulamaları daha çok şu şekilde yapılmaktadır:

- Yüz Tanımlama
- Belgeleri Analiz Etme
- İklimi Tanımlama
- Gri Alanlar

- Reklamlar
- Tarihi Eserlerin Ayrıştırılması

5.3. DERİN ÖĞRENME MİMARİLERİ

Evrişimsel sinir ağlarından sonraki aşama model oluşturma tarafıdır. Tarihsel sıraya göre birden fazla şekilde bu mimariler oluşturulmuştur. Derin Öğrenme Mimarileri şu şekilde sıralanabilir:

- LeNet
- AlexNet
- ZFNet
- GoogLeNet
- VGGNet
- Microsoft ResNet
- MobileNet

ImageNet projesi, görsel nesne tanıma yazılım araştırmalarında kullanılmak üzere tasarlanmış büyük bir görsel veritabanıdır. 14 milyondan fazla görüntü, hangi nesnelerin resmedildiğini göstermek için proje insanlar tarafından elle açıklanmıştır ve görüntülerin en az bir milyonunda sınırlayıcı kutular da bulunmaktadır.

Yapay zeka araştırmacısı Fei-Fei Li , 2006 yılında ImageNet fikri üzerinde çalışmaya başladı. Yapay zeka araştırmalarının çoğunun modellere ve algoritmalara odaklandığı bir zamanda Li, yapay zeka algoritmalarını eğitmek için mevcut verileri genişletmek ve iyileştirmek istedi. 2007 yılında, Li Princeton profesörü ile WordNet projesini görüşmek üzere bir araya geldi. Bu toplantının bir sonucu olarak Li, WordNet'in kelime veritabanından başlayarak ve birçok özelliğini kullanarak ImageNet'i oluşturmaya devam etti.

ILSVRC, 2005 yılında kurulan ve sadece yaklaşık 20.000 görüntü ve yirmi nesne sınıfını içeren daha küçük ölçekli PASCAL VOC mücadelesinin ayak izlerini takip etmeyi amaçlamaktadır. ImageNet'i demokratikleştirmek için Fei-Fei Li, PASCAL VOC ekibine 2010'da başlayarak, araştırma ekiplerinin algoritmalarını verilen veri setinde değerlendirecekleri ve çeşitli görsel tanıma görevlerinde daha yüksek doğruluk elde etmek için rekabet edecekleri bir işbirliği önerdi.

Ortaya çıkan yıllık yarışma artık ImageNet Büyük Ölçekli Görsel Tanıma Mücadelesi (ILSVRC) olarak biliniyor.

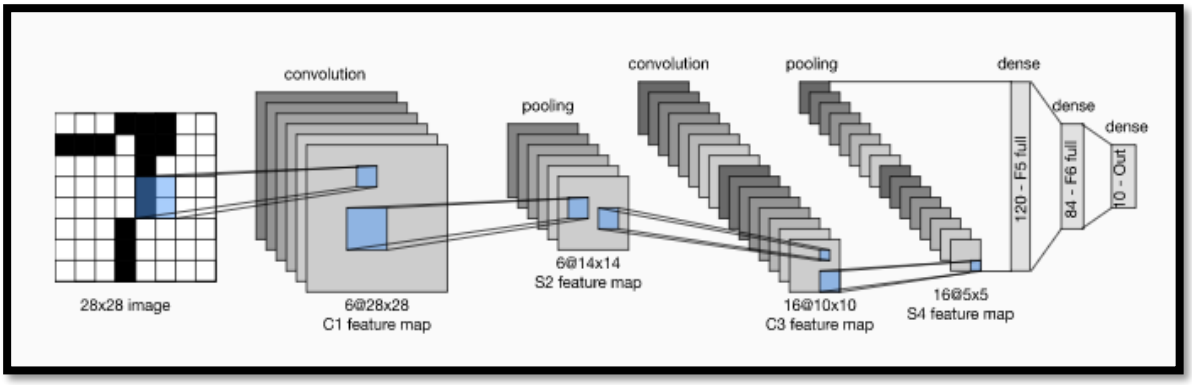
5.3.1. LeNet Mimarisi

LeNet , Yann LeCun ve diğerleri tarafından önerilen evrişimli bir sinir ağı yapısıdır. Genel olarak LeNet, LeNet-5'i ifade eder ve basit bir evrişimli sinir ağıdır . Evrişimsel sinir ağları, yapay nöronların kapsama alanındaki çevreleyen hücrelerin bir kısmına yanıt verebilen ve büyük ölçekli görüntü işlemede iyi performans gösteren bir tür ileri beslemeli sinir ağıdır [30].

Yüksek düzeyde, LeNet (LeNet-5) iki bölümden oluşur:

- İki evrişimli katmandan oluşan bir evrişimli kodlayıcı
- Üç tam bağlantılı katmandan oluşan yoğun bir blok

1998'de LeCun ve diğerleri tarafından rakamları sınıflandıran 7 seviyeli evrişimli bir ağ olan LeNet-5, 32x32 piksel gri tonlamalı girdi görüntülerinde sayısallaştırılmış çeklerdeki (çekler) elle yazılmış numaraları tanımak için birkaç banka tarafından uygulanmıştır. Daha yüksek çözünürlüklü görüntüleri işleme yeteneği, daha büyük ve daha fazla evrişimli katmanlar gerektirir, bu nedenle bu teknik, bilgi işlem kaynaklarının kullanılabilirliği ile sınırlıdır [31].

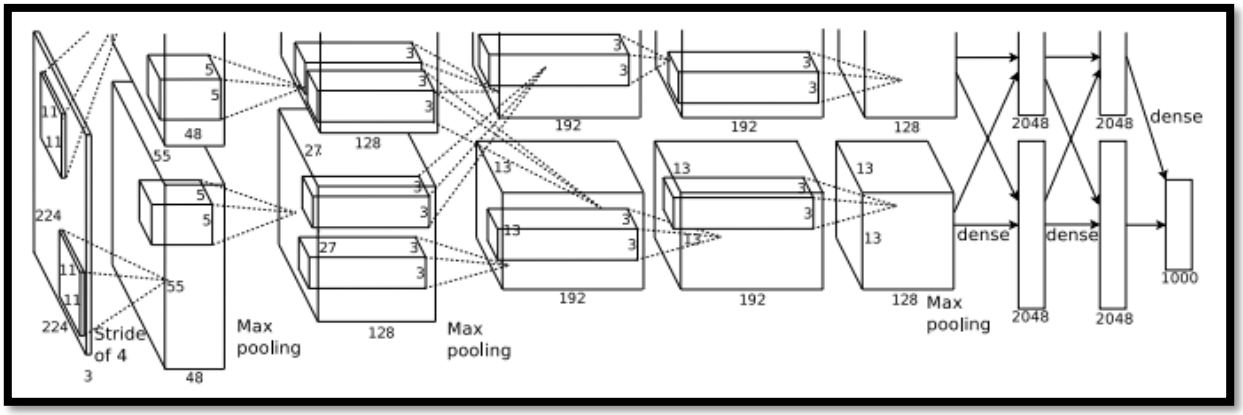


Şekil 4. LeNet'te veri akışı. Giriş el yazısı bir rakamdır, çıktı ise 10 olası sonucun üzerinde bir olasılıktır.

5.3.2. AlexNet Mimarisi

Her ne kadar derin öğrenmenin ilk olarak 1998 yılında Yann LeCun'nun yayınlamış olduğu makale(Lecun, Bottou et al. 1998) ile ilk uygulama yapıldığı söylense de, dünya çapında duyulması 2012 yılında olmuştur. O yıl gerçekleştirilen İmageNet yarışmasını, derin öğrenme mimarisi ile tasarlanan AlexNet modeli kazanmıştır. Yapılan çalışma "ImageNet Classification with Deep

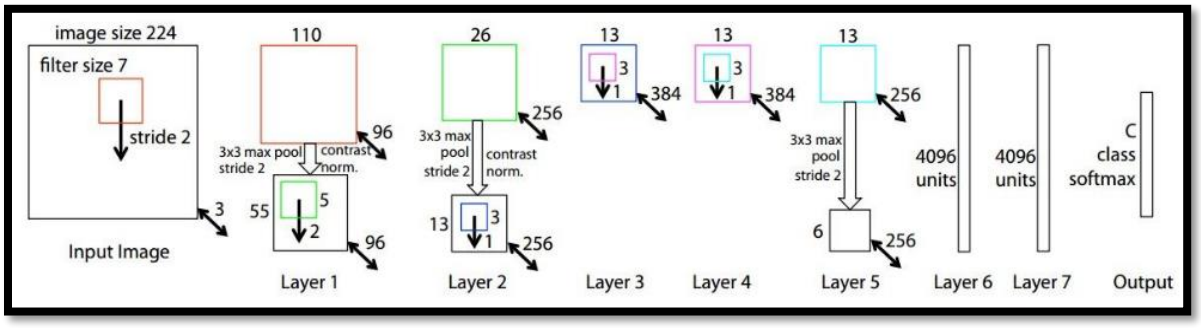
Convolutional Networks"(Krizhevsky, Sutskever et al. 2012) isimli makale ile yayınlanmış ve Ekim 2017 tarihi itibari ile 16227 alıntı yapılmıştır. Bu mimari ile bilgisayarlı nesne tanımlama hata oranı % 26,2'den % 15,4'de düşürülmüştür. Şekil 5'te verilen mimari 5 konvolüsyon katmanı, havuzlama katmanı ve 3 tam bağlantılı katmandan oluşmaktadır. Mimari 1000 nesneyi sınıflandıracak şekilde tasarlanmıştır. Filtreler 11x11 boyutunda ve adım kayma sayısı 4 olarak belirlenmiştir. AlexNet genel olarak 60 milyon parametreye sahiptir [32].



Şekil 5. AlexNet Mimarisi

5.3.3. ZFNet Mimarisi

ZFNet, klasik bir evrişimli sinir ağıdır. Tasarım, ara özellik katmanlarının görselleştirilmesi ve sınıflandırıcının çalışmasıyla motive edildi. 2012 yılında AlexNet'in ImageNet yarışmasını kazanmasının akabinde yapılan yarışmalarda derin öğrenme modelleri kullanılmaya başlandı. Matthew Zeiler ve Rob Fergus tasarlamış olduğu ZFNet(Zeiler and Fergus 2014) 2013 yılında ImageNet yarışmasının kazananı olmuştur. Bu model ile nesne tanımada hata oranı % 11,2'ye indirilmiştir. Bu mimari AlexNet mimarisinin geliştirilmiş halidir.

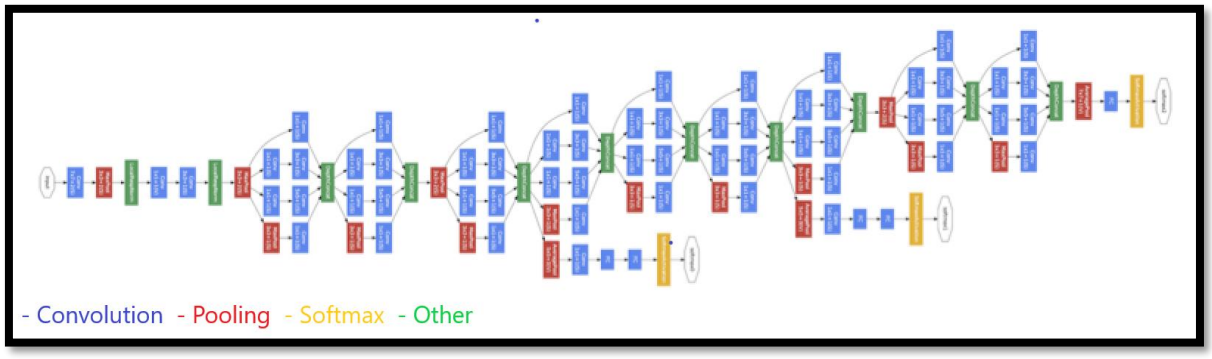


Şekil 6. ZFNet Mimarisi

ZF Net modelinde, ilk katmanda AlexNet'in uyguladığı 11x11 boyutlu filtreler kullanmak yerine, 7x7 boyutundaki filtreleri ve havuzlama katmanında 2 adım kayma miktarı kullanılmıştır. Bu değişikliğin arkasındaki mantık, birinci konvolüsyon katmanındaki daha küçük bir filtre boyutunun, giriş boyutundaki birçok orijinal piksel bilgisinin korunmasına yardımcı olmasıdır. Aktivasyon fonksiyonu için ReLu, hata fonksiyonu için Cross-Entropy Loss ve eğitim için Stochastic Gradient Descent kullanılmıştır. Ekran kartı olarak bir GTX 580 GPU üzerinde on iki gün boyunca eğitimi sürmüştür.

5.3.4. GoogLeNet Mimarisi

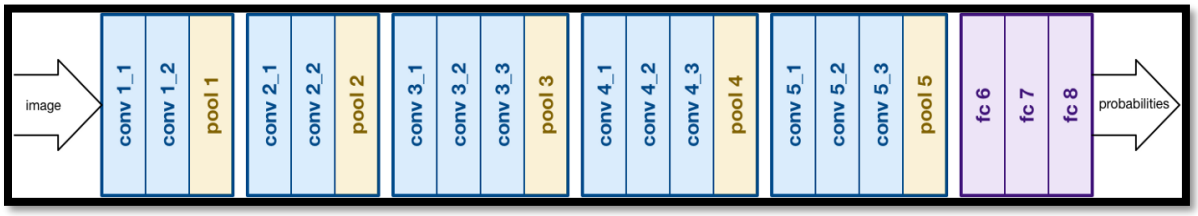
GoogLeNet(Szegedy, Liu et al. 2015) yapısındaki Inception modüllerinden dolayı karmaşık bir mimaridir. GoogLeNet 22 katmanlı ve % 5,7 hata oranı ile ImageNet 2014 yarışmasının kazananı olmuştur. Bu mimari genel olarak, ardışık bir yapıda konvolüsyon ve havuzlama katmanlarını üst üste istiflemekten uzaklaşan ilk CNN mimarilerinden birisidir. Ayrıca bu yeni model bellek ve güç kullanımı üzerinde önemli bir yere sahiptir. Çünkü katmanların hepsini yığınlamak ve çok sayıda filtre eklemek, bir hesaplama ve bellek maliyeti getirir ve ezberleme olasılığını artırır. GoogLeNet bu durumun üstesinden gelmek için paralel olarak birbirine bağlı modüller kullanılmıştır.



Şekil 7. GoogleNet Mimarisi

5.3.5. VGGNet Mimarisi

ILSVRC 2014 yarışmasının ikincisi, topluluk tarafından VGGNet olarak adlandırıldı ve Simonyan ve Zisserman tarafından geliştirildi. VGGNet, 16 evrişimli katmandan oluşur ve tekdüze mimarisi nedeniyle çok çekicidir. AlexNet'e benzer, sadece 3x3 evrişim, ancak çok sayıda filtre. 2–3 hafta boyunca 4 GPU üzerinde eğitim aldı. Görüntülerden özellikleri çıkarmak için şu anda toplulukta en çok tercih edilen seçimdir. VGGNet'in ağırlık konfigürasyonu halka açıktır ve diğer birçok uygulamada ve zorluklarda temel özellik çıkarıcı olarak kullanılmıştır. Ancak VGGNet, idare edilmesi biraz zor olabilen 138 milyon parametreden oluşur.



Şekil 8. VGGNet Mimarisi

AlexNet'in önceki türevleri, ilk evrişimli katmanda daha küçük pencere boyutlarına ve adımlara odaklanırken, VGG, CNN'lerin çok önemli başka bir yönünü ele alıyor: derinlik. VGG'nin mimarisine bakalım:

Giriş

VGG, 224x224 piksellik bir RGB görüntüsü alır. ImageNet yarışması için yazarlar, girdi görüntü boyutunu tutarlı tutmak için her görüntüdeki merkez 224x224 yamayı kırptılar.

Evrişimli Katmanlar

VGG'deki evrişimli katmanlar çok küçük bir alıcı alan kullanır (3x3, sol / sağ ve yukarı / aşağı hala yakalayan olası en küçük boyut). Ayrıca, girdinin doğrusal dönüşümü olarak hareket eden 1x1 evrişim filtreleri de vardır ve bunu bir ReLU birimi takip eder. Evrişim adımı, evrişimden sonra uzamsal çözünürlük korunacak şekilde 1 piksele sabitlenir.

Tamamen Bağlı Katmanlar

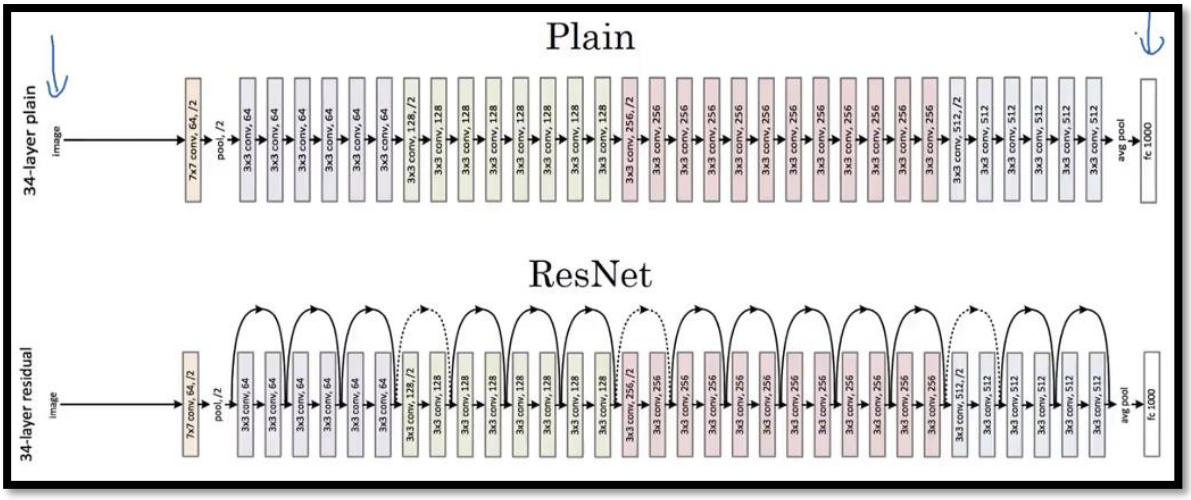
VGG'nin tamamen bağlı üç katmanı vardır: ilk ikisinde 4096 kanal ve üçüncüsünde her sınıf için 1 adet olmak üzere 1000 kanal bulunur.

Gizli Katmanlar

VGG'nin tüm gizli katmanları ReLU (AlexNet'ten eğitim süresini kısaltan büyük bir yenilik) kullanıyor. LRN, doğrulukta belirli bir artış olmaksızın bellek tüketimini ve eğitim süresini artırdığından, VGG genellikle Yerel Yanıt Normalleştirme (LRN) kullanmaz.

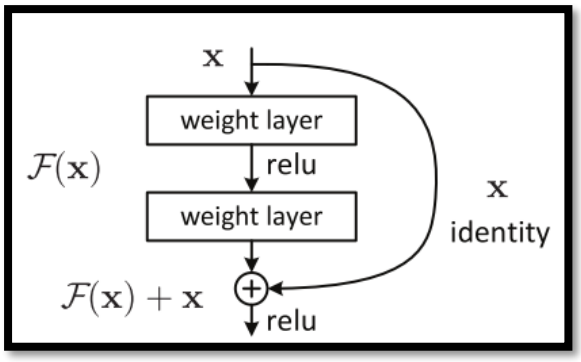
5.3.6. Microsoft ResNet Mimarisi

ResNet(He, Zhang et al. 2016) şuana kadarki tüm mimarilerden daha derin olarak tasarlanan bir mimaridir. 152 katmandan oluşmaktadır. Aynı zamanda % 3,6 (Becerileri ve uzmanlıklarına bağlı olarak, insanlar genelde % 5-10 hata oranına sahipler) hata oranı ile ImageNet 2015 yarışmasının kazananı olmuştur. Microsoft RestNet ilk 34 katmanlı ağ mimarisi Şekil 9'da gösterilmiştir.



Şekil 9. Microsoft ResNet Mimarisi

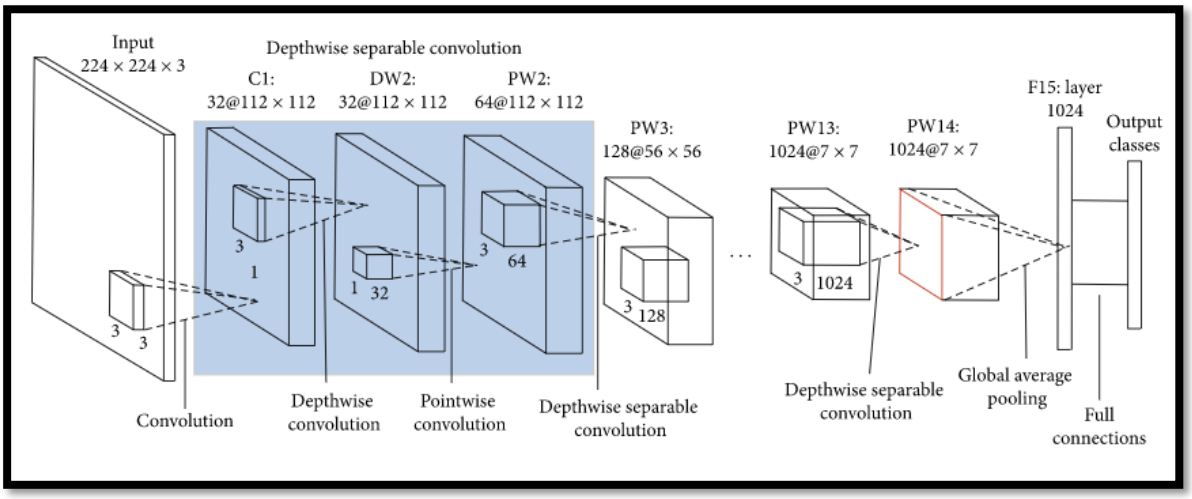
Bu mimari Residual bloklardan oluşmaktadır. Residual blokta, x girişinin konvolüsyonReLu-konvolüsyon serisinden sonra bir F (x) sonucu vermektedir. Bu sonuç daha sonra orijinal x girişine eklenir ve H (x) = F (x) + x olarak ifade edilir (Şekil 10).



Şekil 10. Residual blok

5.3.7. MobileNet

MobileNet, hafif, derin evrişimli sinir ağları oluşturmak için derinlemesine ayrılabilir evrişimler kullanan ve mobil ve gömülü görüntü uygulamaları için verimli bir model sağlayan modern bir mimaridir. MobileNet'in yapısı, Şekil 11'de gösterildiği gibi derinlemesine ayrılabilir filtrelere dayanmaktadır.



Şekil 11. MobileNet Mimarisi

Derinlemesine ayrılabilir evrişim filtreleri, derinlemesine evrişim filtreleri ve nokta evrişim filtrelerinden oluşur. Derinlemesine evrişim filtresi, her bir giriş kanalında tek bir evrişim

gerçekleştirir ve nokta evrişim filtresi derinlemesine evrişim çıktısını doğrusal olarak 1 - 1 kıvrımlarla birleştirir.

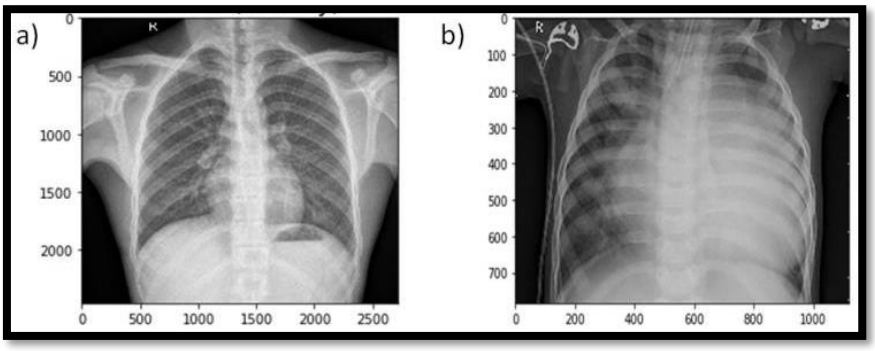
Year	CNN	Developed by	Place	Top-5 error rate	No. of parameters
1998	LeNet(8)	Yann LeCun et al			60 thousand
2012	AlexNet(7)	Alex Krizhevsky, Geoffrey Hinton, Ilya Sutskever	1st	15.3%	60 million
2013	ZFNet()	Matthew Zeiler and Rob Fergus	1st	14.8%	
2014	GoogLeNet(19)	Google	1st	6.67%	4 million
2014	VGG Net(16)	Simonyan, Zisserman	2nd	7.3%	138 million

Tablo 1. Mimarilerin Karşılaştırlması

6. CNN İLE EĞİTİM MODELİ

Pneumonia hastalığının teşhisi için CNN modeli oluşturularak ‘normal’ veya ‘pneumonia’ şeklinde sınıflandırma işlemi gerçekleştirildi.

Veri seti pneumonia ve normal olmak üzere iki etiketten oluşmaktadır.



Şekil 12. (a) sağlıklı bir kişinin ve (b) pneumonia hastalığı taşıyan bir kişinin göğüs röntgeni.

Eğitim için Google Colaboratory ortamı kullanıldı. Google Colaboratory, makine öğrenimi araştırmalarına ve çalışmalarına yardımcı olmak için oluşturulan bir projedir. Kurulum gerektirmeyen ve tamamen bulutta çalışan bir Jupyter notebook ortamıdır.

Colab, derin öğrenme çalışmalarında yardımcı olacak pek çok seçenek ve alternatif sunmaktadır. Colab ile popüler Python kitaplıklarının tüm avantajlarından yararlanarak veriler analiz edilip görselleştirilebilir.

Veri seti google drive ortamına normal hastalara ait X-Ray görüntüleri 'NORMAL' adlı klasörün içerisine ve pneumonia hastalarına ait X-Ray görüntüleri 'PNEUMONIA' adlı klasörün içerisinde olacak şekilde eklendi.

Veri Setinin Alınıp Modele Uygun Hale Getirilmesi

Google drive ortamında bulunan veri setinin uzantısı eklenerek veri setine ulaşım sağlandı. Veri setinde NORMAL ve PNEUMONIA klasörlerindeki X-Ray görüntüleri alınıp etiketleme işlemi yapıldı.

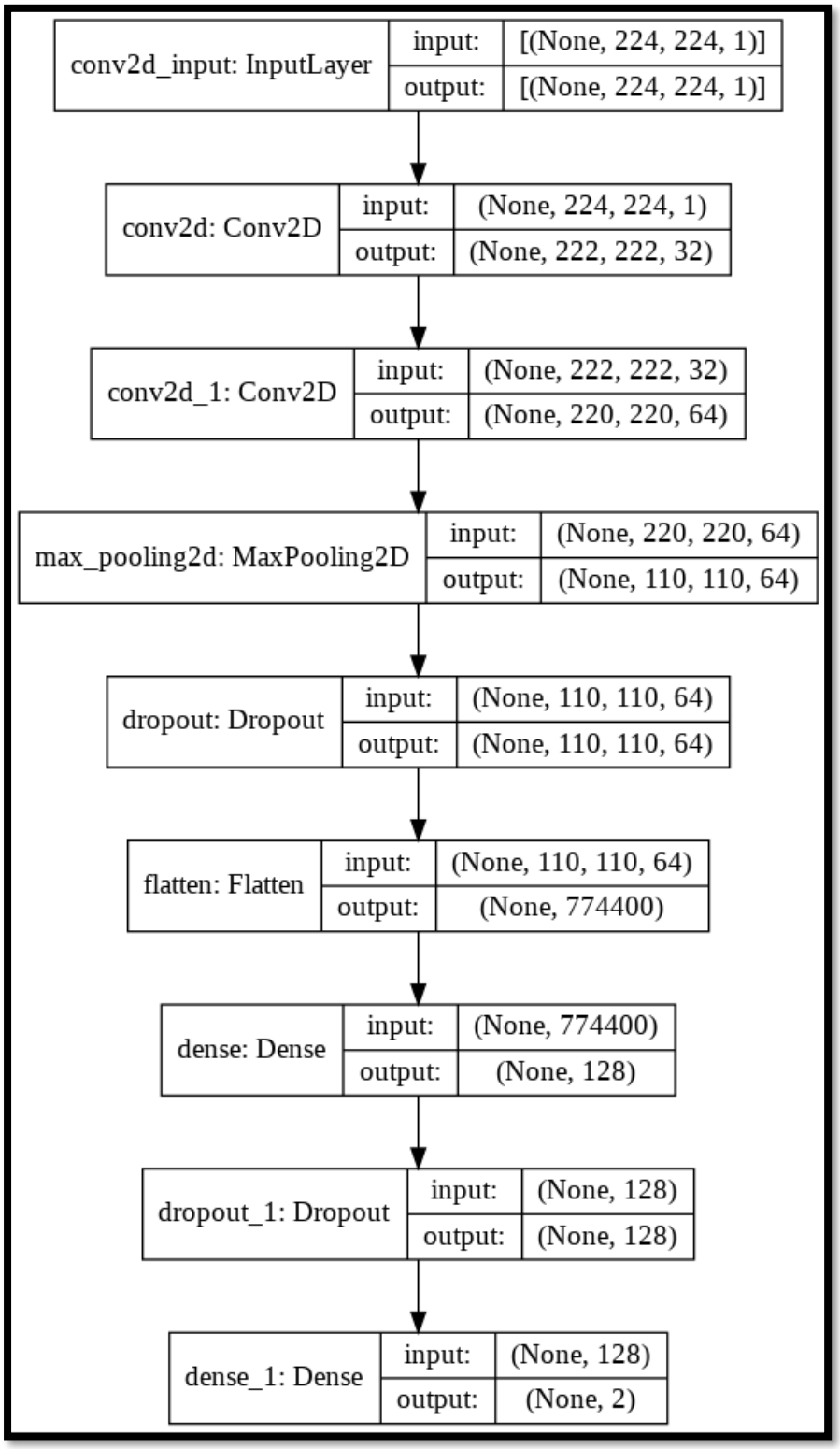
Veri setinin % 80'i eğitim için, % 20'si ise test için kullanılacak şekilde bölümleme yapıldı.

6.1. MODEL 1

Ağ Mimarisi

Ağ mimarisi aşağıdaki bileşenlerden oluşmaktadır.

- 3x3 boyutunda 32 adet filtreden oluşan ReLU aktivasyonlu CONV katmanı.
- 3x3 boyutunda 64 adet filtreden oluşan ReLU aktivasyonlu CONV katmanı.
- 2x2 boyutlu çerçeveden oluşan MAXPOOL katmanı.
- Her seferinde nöronların % 25'inin atılma işlemi (drop).
- Tam bağlantılı (fully connected) katmanına geçiş olacağı için düzleştirme işlemi.
- 128 nörondan oluşan ReLU aktivasyonu FC katmanı.
- Her seferinde nöronların % 50'sinin atılma işlemi (drop).
- Çıkış katmanına sınıf sayısı kadar Softmax aktivasyonlu nöron eklenmesi.



Şekil 13. Model 1 katmanları.

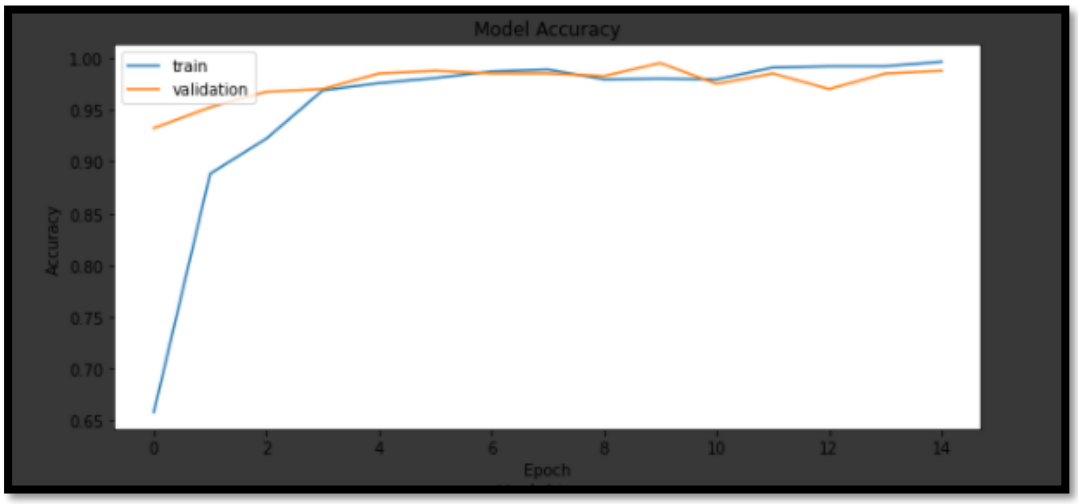
6.1.1. Yöntem 1

- Adadelta optimizasyon yöntemi kullananıldı.

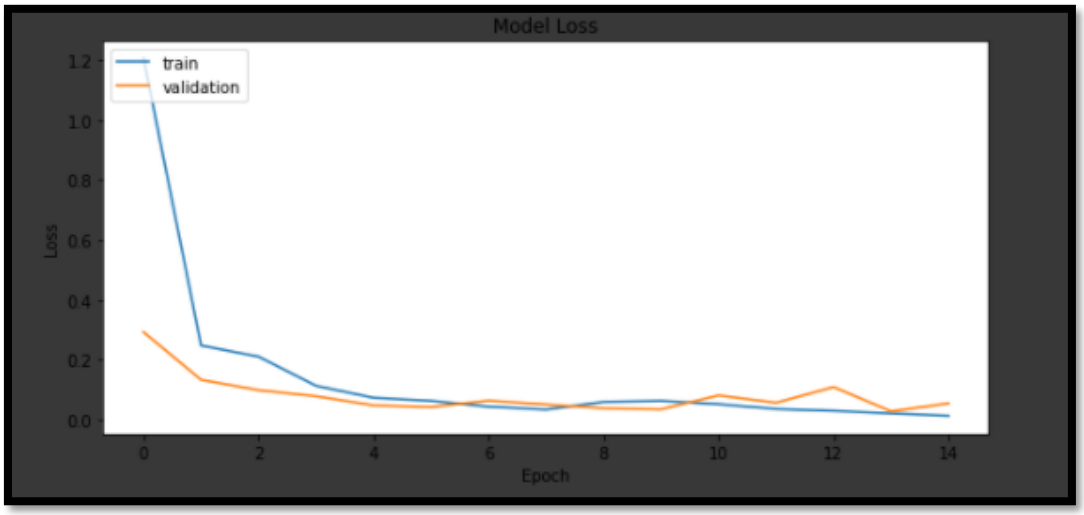
Eğitim Çıktıları:

```
Epoch 1/15
40/40 [==============================] - 6s 150ms/step - loss: 0.6118 - accuracy: 0.6660 - val_loss: 0.5368 - val_accuracy: 0.6784
Epoch 2/15
40/40 [==============================] - 6s 148ms/step - loss: 0.5518 - accuracy: 0.7132 - val_loss: 0.4697 - val_accuracy: 0.7337
Epoch 3/15
40/40 [==============================] - 6s 149ms/step - loss: 0.4869 - accuracy: 0.7571 - val_loss: 0.4154 - val_accuracy: 0.8191
Epoch 4/15
40/40 [==============================] - 6s 150ms/step - loss: 0.4356 - accuracy: 0.8188 - val_loss: 0.3747 - val_accuracy: 0.8668
Epoch 5/15
40/40 [==============================] - 6s 151ms/step - loss: 0.3910 - accuracy: 0.8447 - val_loss: 0.3432 - val_accuracy: 0.8869
Epoch 6/15
40/40 [==============================] - 6s 152ms/step - loss: 0.3811 - accuracy: 0.8630 - val_loss: 0.3159 - val_accuracy: 0.8945
Epoch 7/15
40/40 [==============================] - 6s 151ms/step - loss: 0.3437 - accuracy: 0.8741 - val_loss: 0.2881 - val_accuracy: 0.9347
Epoch 8/15
40/40 [==============================] - 6s 150ms/step - loss: 0.3226 - accuracy: 0.8916 - val_loss: 0.2691 - val_accuracy: 0.9347
Epoch 9/15
40/40 [==============================] - 6s 150ms/step - loss: 0.3001 - accuracy: 0.9098 - val_loss: 0.2570 - val_accuracy: 0.9422
Epoch 10/15
40/40 [==============================] - 6s 149ms/step - loss: 0.2972 - accuracy: 0.9128 - val_loss: 0.2371 - val_accuracy: 0.9397
Epoch 11/15
40/40 [==============================] - 6s 149ms/step - loss: 0.2827 - accuracy: 0.9134 - val_loss: 0.2342 - val_accuracy: 0.9548
Epoch 12/15
40/40 [==============================] - 6s 148ms/step - loss: 0.2556 - accuracy: 0.9314 - val_loss: 0.2168 - val_accuracy: 0.9447
Epoch 13/15
40/40 [==============================] - 6s 148ms/step - loss: 0.2750 - accuracy: 0.9118 - val_loss: 0.2147 - val_accuracy: 0.9598
Epoch 14/15
40/40 [==============================] - 6s 148ms/step - loss: 0.2496 - accuracy: 0.9138 - val_loss: 0.1954 - val_accuracy: 0.9397
Epoch 15/15
40/40 [==============================] - 6s 149ms/step - loss: 0.2296 - accuracy: 0.9348 - val_loss: 0.1862 - val_accuracy: 0.9397
```

Şekil 14. Modelin eğitim işlemi sonucu



Şekil 15. Modele ait doğruluk oranı



Şekil 16. Modele ait kayıp oranı

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_2 (Conv2D)            (None, 222, 222, 32)      320
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 220, 220, 64)      18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 110, 110, 64)      0
_________________________________________________________________
dropout_2 (Dropout)          (None, 110, 110, 64)      0
_________________________________________________________________
flatten_1 (Flatten)          (None, 774400)            0
_________________________________________________________________
dense_2 (Dense)              (None, 128)               99123328
_________________________________________________________________
dropout_3 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_3 (Dense)              (None, 2)                 258
=================================================================
Total params: 99,142,402
Trainable params: 99,142,402
Non-trainable params: 0
_________________________________________________________________
```

Şekil 17. Modele ait özet bilgiler

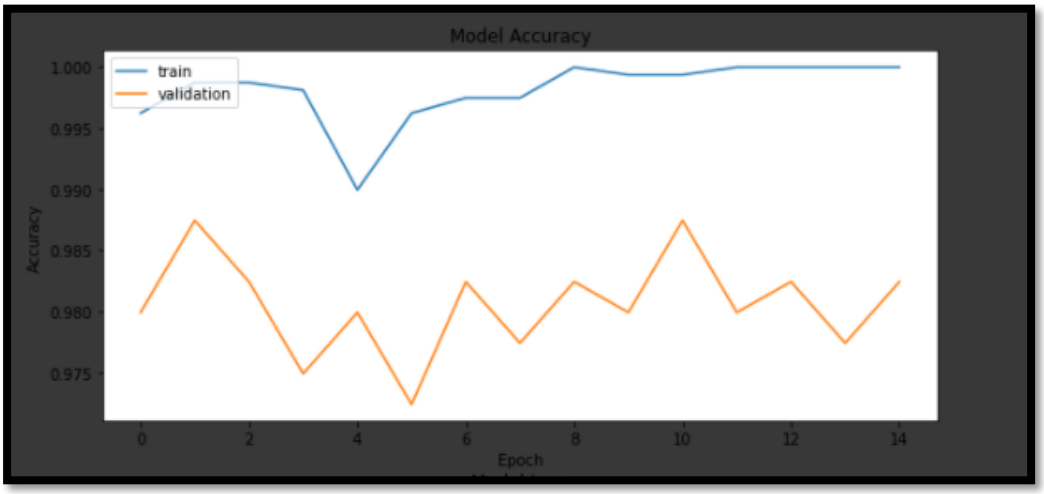
6.1.2. Yöntem 2

- Adam optimizasyon yöntemi kullananıldı.

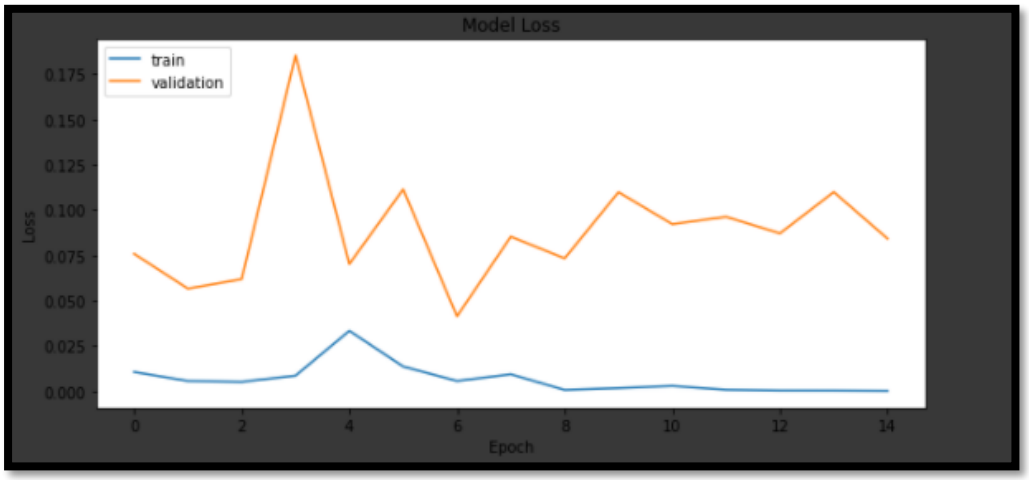
Eğitim Çıktıları:

```
Epoch 1/15
40/40 [==============================] - 6s 147ms/step - loss: 5.7800 - accuracy: 0.6698 - val_loss: 0.1157 - val_accuracy: 0.9422
Epoch 2/15
40/40 [==============================] - 6s 144ms/step - loss: 0.1140 - accuracy: 0.9590 - val_loss: 0.0928 - val_accuracy: 0.9623
Epoch 3/15
40/40 [==============================] - 6s 145ms/step - loss: 0.0798 - accuracy: 0.9725 - val_loss: 0.0621 - val_accuracy: 0.9799
Epoch 4/15
40/40 [==============================] - 6s 143ms/step - loss: 0.0267 - accuracy: 0.9935 - val_loss: 0.0502 - val_accuracy: 0.9849
Epoch 5/15
40/40 [==============================] - 6s 143ms/step - loss: 0.0235 - accuracy: 0.9943 - val_loss: 0.0600 - val_accuracy: 0.9849
Epoch 6/15
40/40 [==============================] - 6s 142ms/step - loss: 0.0076 - accuracy: 0.9977 - val_loss: 0.0517 - val_accuracy: 0.9899
Epoch 7/15
40/40 [==============================] - 6s 142ms/step - loss: 0.0038 - accuracy: 0.9988 - val_loss: 0.0647 - val_accuracy: 0.9849
Epoch 8/15
40/40 [==============================] - 6s 141ms/step - loss: 0.0040 - accuracy: 0.9986 - val_loss: 0.0315 - val_accuracy: 0.9874
Epoch 9/15
40/40 [==============================] - 6s 145ms/step - loss: 0.0034 - accuracy: 0.9991 - val_loss: 0.0491 - val_accuracy: 0.9849
Epoch 10/15
40/40 [==============================] - 6s 141ms/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.0320 - val_accuracy: 0.9899
Epoch 11/15
40/40 [==============================] - 6s 142ms/step - loss: 8.6012e-04 - accuracy: 1.0000 - val_loss: 0.0269 - val_accuracy: 0.9874
Epoch 12/15
40/40 [==============================] - 6s 142ms/step - loss: 4.4109e-04 - accuracy: 1.0000 - val_loss: 0.0226 - val_accuracy: 0.9874
Epoch 13/15
40/40 [==============================] - 6s 142ms/step - loss: 1.7858e-04 - accuracy: 1.0000 - val_loss: 0.0207 - val_accuracy: 0.9874
Epoch 14/15
40/40 [==============================] - 6s 142ms/step - loss: 4.4707e-04 - accuracy: 1.0000 - val_loss: 0.0487 - val_accuracy: 0.9874
Epoch 15/15
40/40 [==============================] - 6s 142ms/step - loss: 3.1332e-04 - accuracy: 1.0000 - val_loss: 0.0259 - val_accuracy: 0.9899
```

Şekil 18. Modelin eğitim işlemi sonucu



Şekil 19. Modele ait doğruluk oranı



Şekil 20. Modele ait kayıp oranı

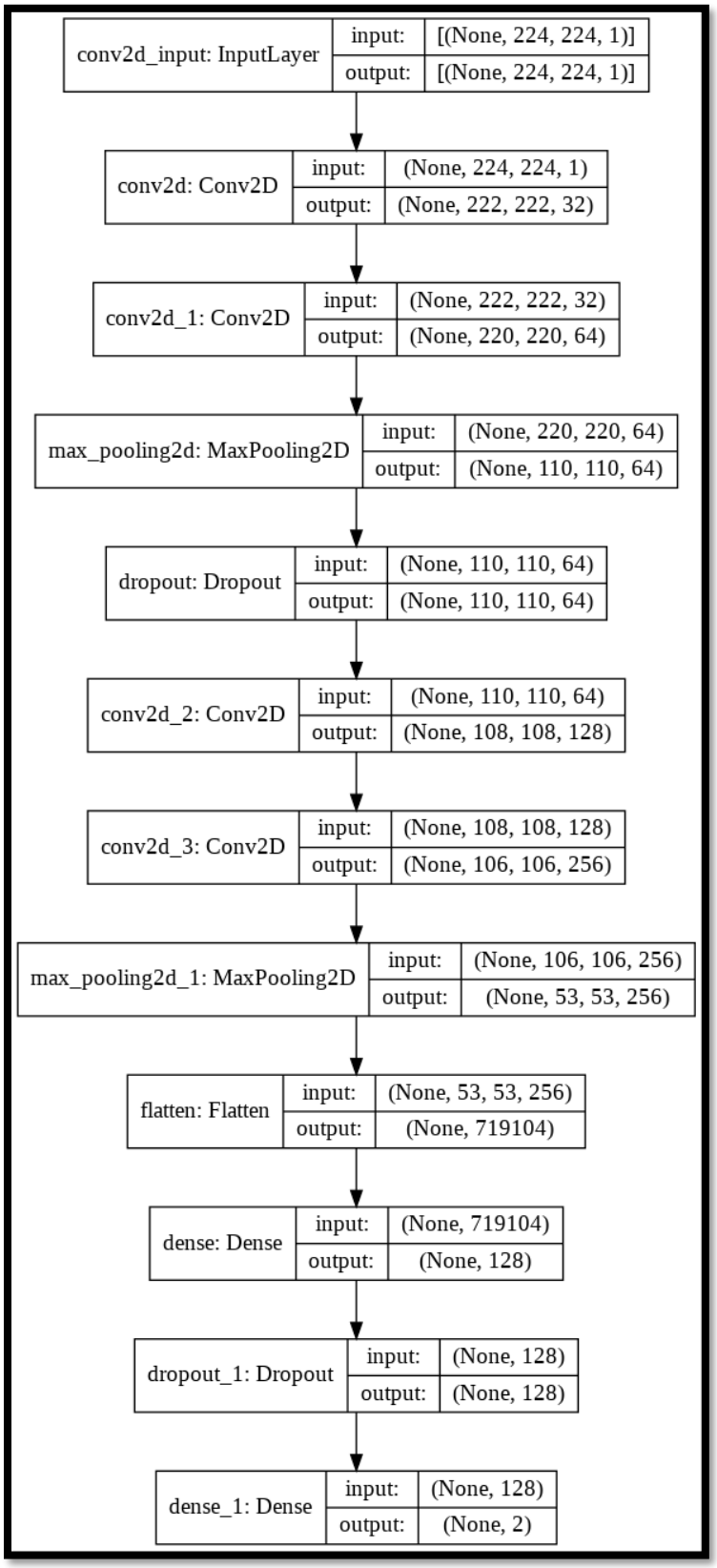
```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_4 (Conv2D)            (None, 222, 222, 32)      320
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 220, 220, 64)      18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 110, 110, 64)      0
_________________________________________________________________
dropout_4 (Dropout)          (None, 110, 110, 64)      0
_________________________________________________________________
flatten_2 (Flatten)          (None, 774400)            0
_________________________________________________________________
dense_4 (Dense)              (None, 128)               99123328
_________________________________________________________________
dropout_5 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_5 (Dense)              (None, 2)                 258
=================================================================
Total params: 99,142,402
Trainable params: 99,142,402
Non-trainable params: 0
```

Şekil 21. Modele ait özet bilgiler

6.2. MODEL 2

Model 1'e ek olarak aşağıdaki özellikler eklendi.

- 3x3 boyutlu 128 adet filtreden oluşan ReLU aktivasyonlu CONV katmanı eklendi.
- 3x3 boyutlu 256 adet filtreden oluşan ReLU aktivasyonlu CONV katmanı eklendi.
- 2x2 boyutlu MaxPooling katmanı eklendi.



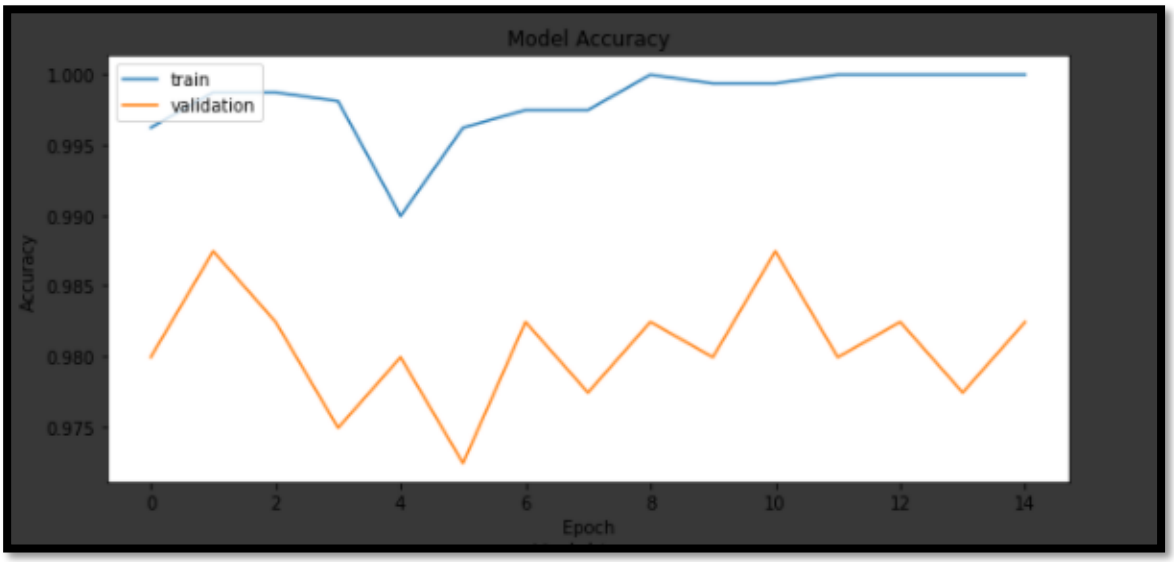
Şekil 22. Model 2 katmanları

6.2.1. Yöntem 1

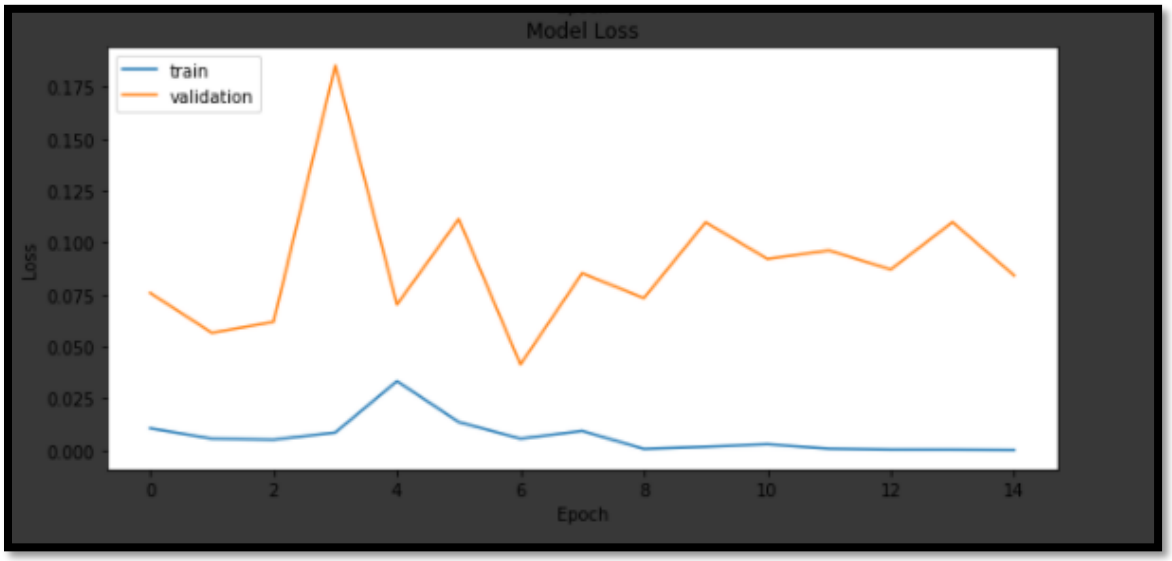
- Adadelta optimizasyon yöntemi kullanıldı.

```
Epoch 1/15
40/40 [==============================] - 13s 317ms/step - loss: 0.3301 - accuracy: 0.9038 - val_loss: 0.3526 - val_accuracy: 0.9397
Epoch 2/15
40/40 [==============================] - 13s 319ms/step - loss: 0.3090 - accuracy: 0.9120 - val_loss: 0.3234 - val_accuracy: 0.9397
Epoch 3/15
40/40 [==============================] - 13s 314ms/step - loss: 0.2955 - accuracy: 0.9007 - val_loss: 0.3030 - val_accuracy: 0.9397
Epoch 4/15
40/40 [==============================] - 12s 311ms/step - loss: 0.2806 - accuracy: 0.9158 - val_loss: 0.2891 - val_accuracy: 0.9397
Epoch 5/15
40/40 [==============================] - 12s 310ms/step - loss: 0.2663 - accuracy: 0.9158 - val_loss: 0.2752 - val_accuracy: 0.9422
Epoch 6/15
40/40 [==============================] - 12s 310ms/step - loss: 0.2557 - accuracy: 0.9183 - val_loss: 0.2614 - val_accuracy: 0.9422
Epoch 7/15
40/40 [==============================] - 12s 313ms/step - loss: 0.2481 - accuracy: 0.9208 - val_loss: 0.2447 - val_accuracy: 0.9397
Epoch 8/15
40/40 [==============================] - 13s 314ms/step - loss: 0.2374 - accuracy: 0.9252 - val_loss: 0.2385 - val_accuracy: 0.9497
Epoch 9/15
40/40 [==============================] - 13s 315ms/step - loss: 0.2290 - accuracy: 0.9309 - val_loss: 0.2210 - val_accuracy: 0.9397
Epoch 10/15
40/40 [==============================] - 12s 313ms/step - loss: 0.2254 - accuracy: 0.9252 - val_loss: 0.2340 - val_accuracy: 0.9623
Epoch 11/15
40/40 [==============================] - 12s 312ms/step - loss: 0.2171 - accuracy: 0.9265 - val_loss: 0.2099 - val_accuracy: 0.9548
Epoch 12/15
40/40 [==============================] - 12s 311ms/step - loss: 0.2080 - accuracy: 0.9283 - val_loss: 0.1999 - val_accuracy: 0.9573
Epoch 13/15
40/40 [==============================] - 12s 312ms/step - loss: 0.1959 - accuracy: 0.9346 - val_loss: 0.1881 - val_accuracy: 0.9472
Epoch 14/15
40/40 [==============================] - 12s 313ms/step - loss: 0.1878 - accuracy: 0.9359 - val_loss: 0.1981 - val_accuracy: 0.9598
Epoch 15/15
40/40 [==============================] - 12s 313ms/step - loss: 0.1873 - accuracy: 0.9327 - val_loss: 0.1787 - val_accuracy: 0.9648
```

Şekil 23. Modelin eğitim işlemi sonucu



Şekil 24. Modele ait doğruluk oranı



Şekil 25. Modele ait kayıp oranı

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_10 (Conv2D)           (None, 222, 222, 32)      320
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 220, 220, 64)      18496
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 110, 110, 64)      0
_________________________________________________________________
dropout_7 (Dropout)          (None, 110, 110, 64)      0
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 108, 108, 128)     73856
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 106, 106, 256)     295168
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 53, 53, 256)       0
_________________________________________________________________
flatten_3 (Flatten)          (None, 719104)            0
_________________________________________________________________
dense_6 (Dense)              (None, 128)               92045440
_________________________________________________________________
dropout_8 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_7 (Dense)              (None, 2)                 258
=================================================================
Total params: 92,433,538
Trainable params: 92,433,538
Non-trainable params: 0
_________________________________________________________________
```

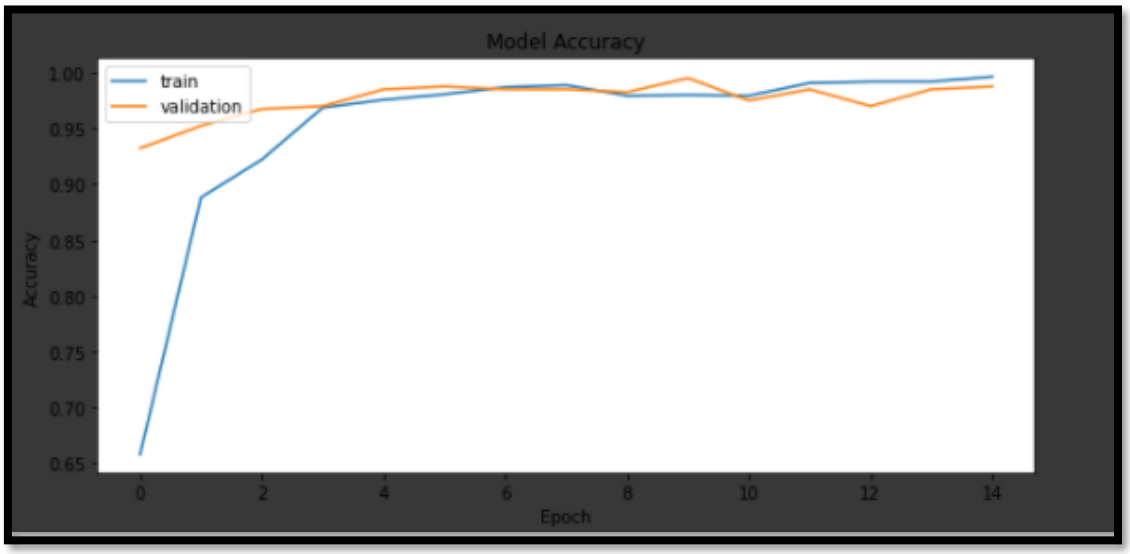
Şekil 26. Modele ait özet bilgiler

6.2.2. Yöntem 2

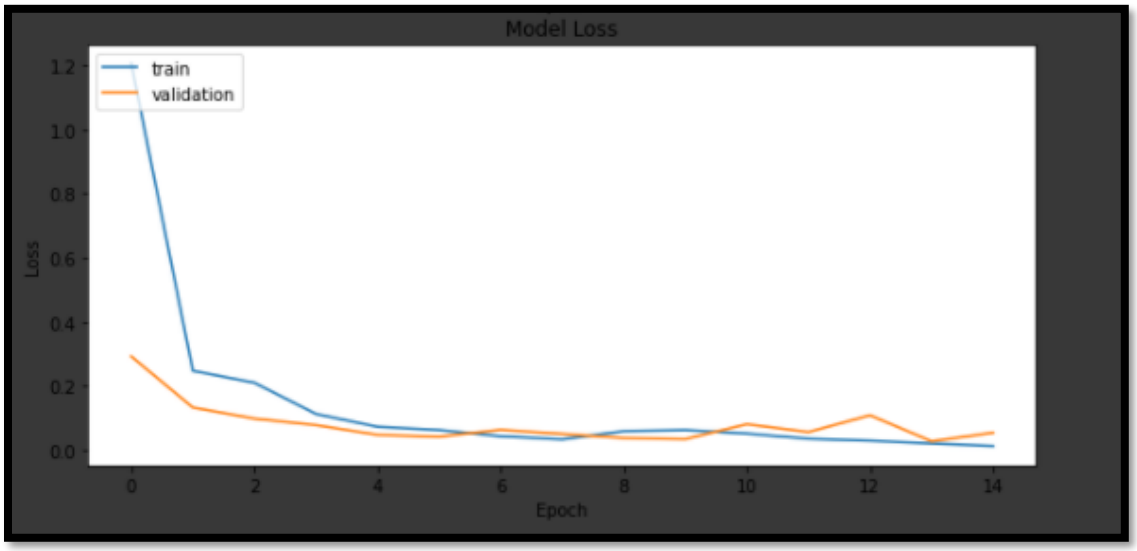
- Adam optimizasyon yöntemi kullanıldı.

```
Epoch 1/15
40/40 [==============================] - 13s 309ms/step - loss: 2.5645 - accuracy: 0.5996 - val_loss: 0.2924 - val_accuracy: 0.9322
Epoch 2/15
40/40 [==============================] - 12s 305ms/step - loss: 0.2903 - accuracy: 0.8558 - val_loss: 0.1330 - val_accuracy: 0.9523
Epoch 3/15
40/40 [==============================] - 12s 302ms/step - loss: 0.2817 - accuracy: 0.9093 - val_loss: 0.0985 - val_accuracy: 0.9673
Epoch 4/15
40/40 [==============================] - 12s 300ms/step - loss: 0.1149 - accuracy: 0.9677 - val_loss: 0.0781 - val_accuracy: 0.9698
Epoch 5/15
40/40 [==============================] - 12s 298ms/step - loss: 0.0619 - accuracy: 0.9809 - val_loss: 0.0477 - val_accuracy: 0.9849
Epoch 6/15
40/40 [==============================] - 12s 299ms/step - loss: 0.0524 - accuracy: 0.9848 - val_loss: 0.0419 - val_accuracy: 0.9874
Epoch 7/15
40/40 [==============================] - 12s 301ms/step - loss: 0.0499 - accuracy: 0.9813 - val_loss: 0.0634 - val_accuracy: 0.9849
Epoch 8/15
40/40 [==============================] - 12s 305ms/step - loss: 0.0463 - accuracy: 0.9834 - val_loss: 0.0505 - val_accuracy: 0.9849
Epoch 9/15
40/40 [==============================] - 12s 302ms/step - loss: 0.0370 - accuracy: 0.9858 - val_loss: 0.0378 - val_accuracy: 0.9824
Epoch 10/15
40/40 [==============================] - 12s 301ms/step - loss: 0.0579 - accuracy: 0.9812 - val_loss: 0.0345 - val_accuracy: 0.9950
Epoch 11/15
40/40 [==============================] - 12s 300ms/step - loss: 0.0591 - accuracy: 0.9727 - val_loss: 0.0809 - val_accuracy: 0.9749
Epoch 12/15
40/40 [==============================] - 12s 299ms/step - loss: 0.0321 - accuracy: 0.9888 - val_loss: 0.0564 - val_accuracy: 0.9849
Epoch 13/15
40/40 [==============================] - 12s 298ms/step - loss: 0.0404 - accuracy: 0.9913 - val_loss: 0.1089 - val_accuracy: 0.9698
Epoch 14/15
40/40 [==============================] - 12s 300ms/step - loss: 0.0230 - accuracy: 0.9929 - val_loss: 0.0277 - val_accuracy: 0.9849
Epoch 15/15
40/40 [==============================] - 12s 300ms/step - loss: 0.0141 - accuracy: 0.9955 - val_loss: 0.0543 - val_accuracy: 0.9874
```

Şekil 27. Modelin eğitim işlemi sonucu



Şekil 28. Modele ait doğruluk oranı



Şekil 29. Modele ait kayıp oranı

```
Model: "sequential_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_14 (Conv2D)           (None, 222, 222, 32)      320
_________________________________________________________________
conv2d_15 (Conv2D)           (None, 220, 220, 64)      18496
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 110, 110, 64)      0
_________________________________________________________________
dropout_9 (Dropout)          (None, 110, 110, 64)      0
_________________________________________________________________
conv2d_16 (Conv2D)           (None, 108, 108, 128)     73856
_________________________________________________________________
conv2d_17 (Conv2D)           (None, 106, 106, 256)     295168
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 53, 53, 256)       0
_________________________________________________________________
flatten_4 (Flatten)          (None, 719104)            0
_________________________________________________________________
dense_8 (Dense)              (None, 128)               92045440
_________________________________________________________________
dropout_10 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_9 (Dense)              (None, 2)                 258
=================================================================
Total params: 92,433,538
Trainable params: 92,433,538
Non-trainable params: 0
```

Şekil 30. Modele ait özet bilgiler

7. SONUÇ VE TARTIŞMA

Bir veri kümesinin tamamı, sinir ağı üzerinden tam olarak bir kez ileri ve geri aktarıldığında bir epoch sona erer. Veri kümesinin tamamı algoritmaya aynı anda aktarılamıyorsa, mini gruplara bölünmesi gerekir. Batch size, tek bir dakikalık grupta bulunan eğitim numunelerinin toplam sayısıdır. Bir iterasyon eğitim sırasında tek bir gradyan güncellemesidir (model ağırlıkların güncellemesi).

Ağ üzerinden her epoch'ta ağırlıklar güncellenir. Epoch sayısını seçmek için bir kural yoktur. Bu, eğitim başlamadan önce belirlenmesi gereken bir hiper parametredir.

Epoch sayısı gibi, batch size da bir kural içermeyen bir hiper parametredir. Çok küçük bir seri boyutunun seçilmesi, küçük bir numunenin tüm veri setinin iyi bir temsili olma olasılığı düşük olduğundan, her parti içinde yüksek derecede bir varyans (gürültü) ortaya çıkaracaktır. Tersine, bir batch size çok büyükse, eğitim için kullanılan hesaplama örneğinin belleğine sığmayabilir ve veriyi fazla sığdırma eğiliminde olacaktır. Batch size'ın öğrenme hızı gibi diğer hiper parametreleri etkilediğini unutmamak önemlidir, bu nedenle bu hiper parametrelerin kombinasyonu, batch size'ın kendisi kadar önemlidir.

Önerilen yaklaşımın etkinliğini değerlendirmek ve doğrulamak için deneyler sırasıyla her biri 15 kez gerçekleştirildi. Modelin performansını artırmak için parametre ve hiper parametreler yoğun bir şekilde çevrildi. Farklı sonuçlar elde edildi, ancak bu çalışmada yalnızca en geçerli olanı rapor ediliyor.

Kullanılan katman sayısı, filtre sayısı ve optimizasyon yöntemlerine bağlı olarak modelde iyileştirmeler yapıldı. İlk modelin ağ mimarisine ek olarak katman derinleştirme, filtre sayısını artırma ve çeşitli optimizasyon yöntemleri denenerek iyileştirme işlemi gerçekleştrilmiştir. Hepsi ayrı ayrı denenerek başarı oranları gözlemlenmiştir.

MODEL 1'in Adadelta optimizasyon yöntemi kullanılan Yöntem 1'de % 93 doğruluk oranı elde edilmişken Adam optimizasyon yöntemi kullanılan Yöntem 2'de % 100 doğruluk oranına ulaşılmıştır.

MODEL 2'de eklenen CONV katmanları ve havuzlama katmanı eğitim için daha doğru başarı oranları elde edilmesini sağlamıştır. MODEL 1'de olduğu gibi ağ mimarilerinin yanı sıra optimizasyon yöntemleri üzerinde değişiklik yaparak başarı oranları gözlemlenmiştir.

Model iyileştirmeleri sonucunda doğruluk oranları karşılaştırıldı ve özet şeklinde aşağıdaki tabloda belirtildi.

Model Adı	**Model Özellikleri**	**Optimizasyon Yöntemi**	**Doğruluk Değeri(%)**
MODEL 1	• 3x3 32 adet filtreli CONV(ReLU) • 3x3 64 adet filtreli CONV(ReLU) • 2x2 MaxPooling	Adadelta	% 93
		Adam	% 100
MODEL 2	• 3x3 32 adet filtreli CONV(ReLU) • 3x3 64 adet filtreli CONV(ReLU) • 2x2 MaxPooling • 3x3 128 adet filtreli CONV(ReLU) • 3x3 256 adet filtreli CONV(ReLU) • 2x2 MaxPooling	Adadelta	% 93
		Adam	% 99

Tablo 2. Model karşılaştırması

Bu proje tamamlanmaktan uzak olsa da, derin öğrenmenin başarısını bu kadar çeşitli gerçek dünya problemlerinde görmek dikkat çekicidir. Bir dizi X-Ray görüntülerinden pozitif ve negatif pneumonia verilerinin nasıl sınıflandırılacağı gösterildi.

8. KAYNAKLAR

[1]. Johns Hopkins Medicine pneumonia. [(erişim tarihi 31 Aralık 2019)];Çevrimiçi erişilebilir: https://www.hopkinsmedicine.org/health/conditions-and-diseases/pneumonia

[2]. Johnson S., Wells D., Healthline Viral pneumonia: Semptomlar, Risk Faktörleri ve Daha Fazlası. [(erişim tarihi 31 Aralık 2019)];Çevrimiçi mevcut: https://www.healthline.com/health/viral-pneumonia

[3]. Sağlık Hizmetleri, Utah Üniversitesi. pneumonia, İlk 10 Ölüm Nedeni İçin Liste Yapıyor. [(erişim tarihi 31 Aralık 2019)]; 2016 Çevrimiçi erişilebilir: https://healthcare.utah.edu/the-scope/shows.php?shows=0_riw4wti7

[4]. Rudan I., Tomaskovic L., Boschi-Pinto C., Campbell H. Beş yaşın altındaki çocuklar arasında klinik pneumonia insidansının global tahmini. Boğa. Dünya Sağlık Örgütü. 2004; 82 : 895–903. [PMC ücretsiz makale] [PubMed] [Google Scholar]

[5]. Zatürree. [(erişim tarihi 31 Aralık 2019)];Çevrimiçi erişilebilir: https://www.radiologyinfo.org/en/info.cfm?pg=pneumonia

[6]. Dünya Sağlık Örgütü. Çocuklarda pneumonia Tanısına Yönelik Göğüs Radyografilerinin Yorumlanmasının Standardizasyonu. Dünya Sağlık Örgütü; Cenevre, İsviçre: 2001. Teknik Rapor. [Google Scholar]

[7]. McLuckie, A. (editor), Respiratory disease and its management. New York, Springer, p. 51, ISBN 978-1-84882-094-4, 2009.

[8]. Osler, William (1901). Principles and Practice of Medicine, 4th Edition. New York: D. Appleton and Company. s. 108.

[9]. Jeffrey C. Pommerville (2010) . Alcamo's Fundamentals of Microbiology (9. bas.). Sudbury MA: Jones & Bartlett. s. 323. ISBN 0-7637-6258-X.

[10]. J. M. Qu and H. Summah, "Biomarkers: A definite plus in pneumonia," Mediators Inflamm., vol. 2009, 2009.

[11]. D. Berliner, N. Schneider, T. Welte, and J. Bauersachs, “The differential diagnosis of dyspnoea,” Dtsch. Arztebl. Int., vol. 113, no. 49, pp. 834–844, 2016.

[12]. H.R. Roth, L. Lu, A. Seff, K.M. Cherry, J. Hoffman, S. Wang, J. Liu, E. Turkbey, R.M. S ummersA new 2.5 d representation for lymph node detection using random sets of deep convolutional neural network observations

International Conference on Medical Image Computing and Computer-assisted Intervention, Springer (2014), pp. 520-527

CrossRefView Record in Scopus Google Scholar

[13]. H.C. Shin, H.R. Roth, M. Gao, L. Lu, Z. Xu, I. Nogues, J. Yao, D. Mollura, R.M. Summe rsDeep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning

IEEE Trans. Med. Imaging, 35 (5) (2016) 1285–computing1298 Google Scholar

[14]. O. Ronneberger, P. Fischer, T. BroxU-net: convolutional networks for biomedical image segmentation

International Conference on Medical Image Computing and Computer-assisted Intervention, Springer (2015), pp. 234-241

CrossRefView Record in Scopus Google Scholar

[15]. A. Jamaludin, T. Kadir, A. ZissermanSpinenet: automatically pinpointing classification evidence in spinal mris

International Conference on Medical Image Computing and Computer-Assisted Intervention, Springer (2016), pp. 166-175

CrossRefView Record in Scopus Google Scholar

[16]. K. Kallianos, J. Mongan, S. Antani, T. Henry, A. Taylor, J. Abuya, M. Kohli, How far have we come? artificial intelligence for chest radiograph interpretation, Clin. Radiol.

Google Scholar

[17]. X. Wang, Y. Peng, L. Lu, Z. Lu, M. Bagheri, R.M. SummersChestx-ray8: hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases
Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2017), pp. 2097-2106 View Record in Scopus Google Scholar

[18]. P. Rajpurkar, J. Irvin, K. Zhu, B. Yang, H. Mehta, T. Duan, D. Ding, A. Bagul, C. Langlotz, K. Shpanskaya, et al., Chexnet: radiologist-level pneumonia detection on chest x-rays with deep learning. arXiv:1711.05225. Google Scholar

[19]. J. Irvin, P. Rajpurkar, M. Ko, Y. Yu, S. Ciurea-Ilcus, C. Chute, H. Marklund, B. Haghgoo, R. Ball, K. Shpanskaya, et al., Chexpert: a large chest radiograph dataset with uncertainty labels and expert comparison. arXiv:1901.07031. Google Scholar

[20]. S. Ren, K. He, R. Girshick, J. SunFaster r-cnn: towards real-time object detection with region proposal networks Adv. Neural Inf. Process. Syst. (2015), pp. 91-99
View Record in ScopusGoogle Scholar

[21]. J. Long, E. Shelhamer, T. DarrellFully convolutional networks for semantic segmentation Proceedings of the IEEE conference on computer vision and pattern recognition (2015), pp. 3431-3440 CrossRefView Record in Scopus Google Scholar

[22]. K. He, G. Gkioxari, P. Dollár, R. GirshickMask r-cnn
Proceedings of the IEEE International Conference on Computer Vision (2017), pp. 2961-2969
View Record in Scopus Google Scholar

[23]. J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. S ong, S. Guadarrama, *et al.*Speed/accuracy trade-offs for modern convolutional object detectors Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2017), pp. 7310-7311 View Record in Scopus Google Scholar

[24]. A. Shrivastava, R. Sukthankar, J. Malik, A. Gupta, Beyond skip connections: top-down modulation for object detection. arXiv:1612.06851. Google Scholar

[25]. P. Luc, C. Couprie, Y. Lecun, J. VerbeekPredicting future instance segmentation by forecasting convolutional features Proceedings of the European Conference on Computer Vision (ECCV) (2018), pp. 584-599 View Record in Scopus Google Scholar

[26]. L. Deng and D. Yu, "Deep Learning: Methods and Applications," Found. Trends® Signal Process., vol. 7, no. 3–4, pp. 197–387, 2014.

[27]. Y. Bengio, "Learning Deep Architectures for AI," Found. trends® Mach. Learn., vol. 2, no. 1, pp. 1–127, 2009.

[28]. H. A. Song and S.-Y. Lee, "Hierarchical Representation Using NMF," in International Conference on Neural Information Processing., 2013, pp. 466–473.

[29]. T. Dettmers, «Özetle Derin Öğrenme: Tarihçe ve Paralel Anlatım,» 2015 .. [Çevrimiçi]. **https://devblogs.nvidia.com/parallelforall/deep-learning-nutshell-history-training/.**

[30]. .LeNet, *Wikipedia Blog* https://en.wikipedia.org/wiki/LeNet

[31]. CNN Mimarileri, *Analytic Vidhya Blog* https://medium.com/analytics-vidhya/cnns-architectures-lenet-alexnet-vgg-googlenet resnet-and-more-666091488df5

[32]. AlexNet Mimarisi, *Great Learning Blog* https://www.mygreatlearning.com/blog/alexnet-the-first-cnn-to-win-image-net/

9. EKLER

EK 1 Veri Setinin Alınıp Modele Uygun Hale Getirilmesi ve Etiketlenmesi

```
import numpy as np
import os
from tensorflow.keras.preprocessing.image import img_to_array, load_img
DIRECTORY = r"/content/drive/MyDrive/Proje/"
CATEGORIES = ["NORMAL", "PNEUMONIA"]
```

```
data = []
labels = []
for category in CATEGORIES:
    path = os.path.join(DIRECTORY, category)
    for img in os.listdir(path):
      img_path = os.path.join(path, img)
      image = img_to_array(load_img(img_path, color_mode="grayscale" , tar
get_size=(224, 224)))/255
      data.append(image)
      labels.append(category)
```

```
from tensorflow.keras.utils import to_categorical
from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split
```

```
lb = LabelBinarizer()
labels = lb.fit_transform(labels)
labels = to_categorical(labels)
```

```
veri = np.array(data, dtype="float32")
etiket = np.array(labels)
```

```
(trainX, testX, trainY, testY) = train_test_split(veri, etiket,
  test_size=0.20, stratify=labels, random_state=42)
```

```
import keras
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
```

```
batchSize = 40
classes = 2
epoch = 15
dataSize = trainX.shape
imgRow, imgCol, channel = dataSize[1], dataSize[2], dataSize[3]
```

EK 2 Model 1

Yöntem 1

```
model = Sequential()

model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=(imgRow,imgCol,1)))

model.add(Conv2D(64, (3, 3), activation='relu'))

model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Dropout(0.25))

model.add(Flatten())

model.add(Dense(128, activation='relu'))

model.add(Dropout(0.5))

model.add(Dense(classes, activation='softmax'))
```

```
model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adadelta(),
              metrics=['accuracy'])
```

```
CNN_AdadeltaModel = model.fit(trainX, trainY,
          batch_size=batchSize,
          epochs=epoch,
          verbose=1,
          validation_data=(testX, testY),
```

```
          callbacks=[kerasboard])
```

```
plt.figure(1, figsize = (10, 10))
plt.subplot(211)
plt.plot(CNN_AdamModel.history['accuracy'])
plt.plot(CNN_AdamModel.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['train', 'validation'], loc='upper left')

plt.subplot(212)
plt.plot(CNN_AdamModel.history['loss'])
plt.plot(CNN_AdamModel.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['train', 'validation'], loc='upper left')
plt.show()
```

```
score = model.evaluate(testX, testY, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

Yöntem 2

```
model = Sequential()

model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
                 input_shape=(imgRow,imgCol,1)))

model.add(Conv2D(64, (3, 3), activation='relu'))

model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Dropout(0.25))

model.add(Flatten())

model.add(Dense(128, activation='relu'))

model.add(Dropout(0.5))
```

```
model.add(Dense(classes, activation='softmax'))
```

```
model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adam(),
              metrics=['accuracy'])
CNN_AdamModel = model.fit(trainX, trainY,
          batch_size=batchSize,
          epochs=epoch,
          verbose=1,
          validation_data=(testX, testY))
```

```
plt.figure(1, figsize = (10, 10))
plt.subplot(211)
plt.plot(CNN_AdamModel.history['accuracy'])
plt.plot(CNN_AdamModel.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['train', 'validation'], loc='upper left')

plt.subplot(212)
plt.plot(CNN_AdamModel.history['loss'])
plt.plot(CNN_AdamModel.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['train', 'validation'], loc='upper left')
plt.show()
```

EK 3 Model 2

Yöntem 1

```
model = Sequential()

model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=(imgRow,imgCol,1)))

model.add(Conv2D(64, (3, 3), activation='relu'))
```

```
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Dropout(0.25))
model.add(Conv2D(128, (3,3),activation='relu'))
model.add(Conv2D(256, (3,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Flatten())

model.add(Dense(128, activation='relu'))

model.add(Dropout(0.5))

model.add(Dense(classes, activation='softmax'))
```

```
model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adadelta(),
              metrics=['accuracy'])
```

```
CNN_AdadeltaModel = model.fit(trainX, trainY,
          batch_size=batchSize,
          epochs=epoch,
          verbose=1,
          validation_data=(testX, testY))
```

```
plt.figure(1, figsize = (10, 10))
plt.subplot(211)
plt.plot(CNN_AdamModel.history['accuracy'])
plt.plot(CNN_AdamModel.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['train', 'validation'], loc='upper left')

plt.subplot(212)
plt.plot(CNN_AdamModel.history['loss'])
plt.plot(CNN_AdamModel.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['train', 'validation'], loc='upper left')
plt.show()
```

```
score = model.evaluate(testX, testY, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

Yöntem 2

```
model = Sequential()

model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=(imgRow,imgCol,1)))

model.add(Conv2D(64, (3, 3), activation='relu'))

model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Dropout(0.25))
model.add(Conv2D(128, (3,3),activation='relu'))
model.add(Conv2D(256, (3,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Flatten())

model.add(Dense(128, activation='relu'))

model.add(Dropout(0.5))

model.add(Dense(classes, activation='softmax'))
```

```
model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adam(),
              metrics=['accuracy'])
CNN_AdamModel = model.fit(trainX, trainY,
          batch_size=batchSize,
          epochs=epoch,
          verbose=1,
          validation_data=(testX, testY))
```

```
plt.figure(1, figsize = (10, 10))
plt.subplot(211)
```

```
plt.plot(CNN_AdamModel.history['accuracy'])
plt.plot(CNN_AdamModel.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['train', 'validation'], loc='upper left')

plt.subplot(212)
plt.plot(CNN_AdamModel.history['loss'])
plt.plot(CNN_AdamModel.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['train', 'validation'], loc='upper left')
plt.show()
```

```
score = model.evaluate(testX, testY, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

10. ÖZGEÇMİŞ

Adı Soyadı: Esra YÜCE

Okul Numarası: 16008117051

Doğum Yeri: HAKKÂRİ

Doğum Tarihi: 01.04.2000

Yabancı Dili: İngilizce

Eğitim Durumu:

Lise: İMKB Derecik Çok Programlı Lisesi (2013 - 2017)

Lisans: Yozgat Bozok Üniversitesi, Bilgisayar Mühendisliği Bölümü (2017 - 2021)