T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

DERİN ÖĞRENME AĞLARI KULLANILARAK 3B TIBBİ GÖRÜNTÜ TANIMLANMASI

YÜKSEK LİSANS TEZİ

Rouba OMAR ALAHMAD ALOSMAN

Bilişim Sistemleri Mühendisliği Anabilim Dalı

MAYIS 2023

T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

DERİN ÖĞRENME AĞLARI KULLANILARAK 3B TIBBİ GÖRÜNTÜ TANIMLANMASI

YÜKSEK LİSANS TEZİ

Rouba OMAR ALAHMAD ALOSMAN

Bilişim Sistemleri Mühendisliği Anabilim Dalı

Tez Danışmanı: Prof. Dr. İsmail Hakkı CEDİMOĞLU

MAYIS 2023

Rouba OMAR ALAHMAD ALOSMAN tarafından hazırlanan "Derin Öğrenme Ağları Kullanılarak 3b Tıbbi Görüntü Tanımlanması" adlı tez çalışması 05.05.2023 tarihinde aşağıdaki jüri tarafından oy birliği/oy çokluğu ile Sakarya Üniversitesi Fen Bilimleri Enstitüsü Bilişim Sistemleri Mühendisliği Anabilim Dalı Yüksek Lisans tezi olarak kabul edilmiştir.

Tez Jürisi

Jüri Başkanı : **Prof. Dr. İsmail Hakkı CEDİMOĞLU** (Danışman) ……
Sakarya Üniversitesi

Jüri Üyesi : **Dr. Öğr. Üyesi Ahmet KARACA**
Sakarya Uygulamalı Bilimler Üniversitesi .....

Jüri Üyesi : **Dr.Öğr.Üyesi Muhammed KOTAN** .....
Sakarya Üniversitesi

ETİK İLKE VE KURALLARA UYGUNLUK BEYANNAMESİ

Sakarya Üniversitesi Fen Bilimleri Enstitüsü Lisansüstü Eğitim-Öğretim Yönetmeliğine ve Yükseköğretim Kurumları Bilimsel Araştırma ve Yayın Etiği Yönergesine uygun olarak hazırlamış olduğum "DERİN ÖĞRENME AĞLARI KULLANILARAK 3B TIBBİ GÖRÜNTÜ TANIMLANMASI" başlıklı tezin bana ait, özgün bir çalışma olduğunu; çalışmamın tüm aşamalarında yukarıda belirtilen yönetmelik ve yönergeye uygun davrandığımı, tezin içerdiği yenilik ve sonuçları başka bir yerden almadığımı, tezde kullandığım eserleri usulüne göre kaynak olarak gösterdiğimi, bu tezi başka bir bilim kuruluna akademik amaç ve unvan almak amacıyla vermediğimi ve 20.04.2016 tarihli Resmi Gazete'de yayımlanan Lisansüstü Eğitim ve Öğretim Yönetmeliğinin 9/2 ve 22/2 maddeleri gereğince Sakarya Üniversitesi'nin abonesi olduğu intihal yazılım programı kullanılarak Enstitü tarafından belirlenmiş ölçütlere uygun rapor alındığını, çalışmamla ilgili yaptığım bu beyana aykırı bir durumun ortaya çıkması halinde doğabilecek her türlü hukuki sorumluluğu kabul ettiğimi beyan ederim.

(29/12/2022).

(imza)

Rouba OMAR ALAHMAD ALOSMAN

TEŞEKKÜR

Yıllar boyunca ciddi bir emek ve özveri hazırladığım yüksek lisans tezim tamamlamanın heyecanın ve gururunu yaşıyorum. Bu bölümü, bendenize yardım ve teşvik eden insanlara teşekkür etmek için bir fırsat olarak kullanacağım.

Öncelikle danışmanlığımı üstlenen, tüm çabalarını ve sabrını son derece takdir ediyorum, tez süreci boyunca en iyi sonucu almam için beni sınırlamayıp özgür bırakan Prof. Dr. İsmail Hakkı CEDİMOĞLU'na teşekkür ediyorum.

Bana olan destekleri ve her zaman yanımda oldukları için eşime, çocuklarıma ve aileme çok teşekkür ediyorum.

Rouba OMAR ALAHMAD ALOSMAN

İÇİNDEKİLER

Sayfa

KISALTMALAR

ACC : Kesinlik

AUC : Eğri Altındaki Alan

CT-Scan : Bilgisayarlı Tomografi Taraması

Conv3D : Evrişim 3D

CM : Karışıklık Matrisi

CNN : Evrişimli Sinir Ağları

DL : Derin Öğrenme

DNN : Derin Sinir Ağı

FC : Tamamen Bağlı

FRR : Yanlış Kabul Oranı

FN : Yanlış Negatifler

FP : Yanlış Pozitifler

GPU : Grafik İşlem Birimi

HU : Ev Saha Birimleri

Keras : Hızlı derin öğrenme deneyi için tasarlanmış bir tensorflow arayüz kitaplığı

Logging : Mesajları görüntülemek için kullanılan Abseil kitaplığının işlevi (hata, uyarı, hata ayıklama, bilgi, önemli).

Loss : Kayıp Fonksiyonu

MRI : Manyetik Rezonans Görüntüleme

MSE : Ortalama Kare Hatası

Nibabel : Kütüphane PyNIfTI'nin halefidir. Bu kitaplık, bazı yaygın beyin görüntüleme dosya formatlarında (Okuma, yazma) işlemleri için kullanılır.

Ndimage : Scipy paketi, bir dizi görüntü işleme işlemi sağlar. Çok boyutlu diziler için kullanılabilirler

Numpy : Dizi işlemleri için kullanılan kütüphane.

OS : Kitaplık, birleştirme yolları gibi işletim sistemi işlevleri için kullanılır.

RGB : Kırmızı Yeşil Mavi

ResNet : Artık Ağlar

ROI : İlgi Alanı (Projemizde Akciğer bölgesi)

ROC **:** Alıcı işletim karakteristiği

Random **:** Kütüphane rastgele sayılar üretmek için kullanılır

Tf **:** Tensorflow kitaplığı

tensorflow_hub: Bir dizi ilgili makine öğrenimi modeli.

TP **:** Gerçek Pozitifler

TN **:** Gerçek Negatifler

TPR **:** Gerçek Pozitif Oranı

TNR **:** Gerçek Negatif Oranı

Zipfile **:** Sıkıştırılmış dosyalarla uğraşmak için kitaplık

2D/2B **:** İki Boyutlu Görüntü

3D/3D **:** Üç Boyutlu Görüntü

TABLO LİSTESİ

Sayfa

ŞEKİL LİSTESİ

Sayfa

DERİN ÖĞRENME AĞLARI KULLANILARAK 3B TIBBİ GÖRÜNTÜ TANIMLANMASI

ÖZET

Günümüzde, derin öğrenme ağlarının kullanımı, birden fazla alandaki performansından ve sürekli gelişen yeteneklerinden yararlanmak için, günümüzde bu tür ağlara güveni artıran en önemli konulardan biridir. Ancak, klasik etkileşim derin öğrenme ağları ve kamera veya videoya dayalı sıradan görüntüler arasında belirgin olabilir. Bazı ciddi hastalıkların tespitinde çok fazla doğruluk gerektiğinden ve örneğin trafik gibi ihlallerin baş edilmesi ile ilgili kullanımlar doğru ve doğru olmayı gerektirdiğinden, iki boyutluluk bu alanda ilerlemek için gereken istek ve gelişmelerin gerisinde kalmaktadır. Tüm bunlar, sistemin sağlam bir şekilde görülmesi, iki boyutlu perspektifte olmayan faydaları nedeniyle üç boyutlu bir perspektifle temsil edilen görüntünün yeni ve doğru yüzü ile ele alınması için çalışmayı gerektirmiştir. Sinir ağını, tanımlama yeteneğinde daha fazla doğruluk ve performansta doğruluk sağlayan özelliklerle zenginleştirmek gerekmektedir.

Tanıma alanında iki boyutlu görüntülerin sunduğu geniş seçeneklere rağmen, yine de istenen hedefe ulaşmakta yetersiz kalmaktadır ve bu, en son görüntü teknolojilerine güvenmek, bunları derin öğrenme yoluyla işlemek ve ele almak için sürekli gelişmeyi gerektirmektedir. Ağlar, derin öğrenme ağları, (kenarlar, çizgiler, eğriler, açılar, noktalar vb.) dahil olmak üzere temel bilgileri çıkarmak için çok sayıda 2B görüntüyü manipüle edebilen ağlardır.

Konvolüsyonel sinir ağları, sırayla değişen ağırlıklara ve yer değiştirmelere sahip nöronlardan meydana gelmesi bakımından normal sinir ağlarına çok benzemektedir. Her nöron bazı girdi verilerini alır, standart ürünü hesaplar ve isteğe bağlı olarak doğrusal olmayan bir aktivasyon fonksiyonu kullanır. Ağın tamamı, daha önce olduğu gibi, tek ayırt edilebilir değerlendirme işlevidir: bir uçta ilk piksel kümesinden (görüntü), diğer uçta belirli bir sınıfa ait olma olasılık dağılımına kadar.

Bu ağlar, son (tamamen bağlı) katmanda hala bir kayıp fonksiyonuna (örneğin, SVM/Softmax) sahiptir ve normal sinir ağlarıyla ilgili olarak verilen tüm tavsiyeler, evrişimli sinir ağlarıyla da ilgilidir.

Evrişimli sinir ağlarının mimarisi, girdi verilerinin bazı özelliklerini, ağ mimarisinin kendisinde dikkate almamıza izin veren girdide açıkça görüntü edinmeyi içerir. Bu özellikler, doğrudan dağıtım işlevinin daha verimli uygulanmasına ve ağdaki toplam parametre sayısını önemli ölçüde azaltılmasına imkan tanır. Gerçekleştirilen bu araştırmada, 3D evrişimli sinir ağı olan 3DCNN'ye dayalı olarak derin öğrenme kapsamında tam bir 3B tıbbi görüntü tanıma modeli tasarlanarak uygulanmıştır. Sistemi meydana getirmek amacıyla, 3D görüntü veri seti "MosMedData" kullanılmıştır. Kullanılan veri setinde her gibi 12800 kısımdan oluşan ve 64 bölüm içeren 100 normal ve 100 anormal 3D görüntüden yararlanılmıştır. Kullanılan veri seti, %70 eğitim veri seti ve %30 doğrulama veri olmak üzere ayrıştırılmıştır.

Önerilen derin öğrenme modelini oluşturmak için dört temel adım yer almaktadır. İlki, ön işleme aşaması, ardından veri çoğaltma süreci, ardından derin öğrenme modeli yapılandırması ve son aşama ise eğitim ve değerlendirme kısmından oluşmaktadır. Ön işleme adımı için, akciğer dokularını elde etmek için yeniden boyutlandırma, yeniden ölçeklendirme ve normalizasyon gibi birçok ön işleme işlemi kullanılmaktadır. İkinci adım, eğitimin sahte eğitim sorununu ortadan kaldırabilmesi için aynı görüntünün döndürülmüş, çevrilmiş ve kontrastı değiştirilmiş kopyalarını kullanarak eğitim görüntülerinin sayısını artırmak için uygulanan veri büyütme ve veri artırmadır. Karıştırma işlemi, eğitim örneğinin eğitim sürecine karıştırılma sırasına göre verilmesi için de uygulanmaktadır. Eğitim modeli için 3B CNN ağının değiştirilmiş bir versiyonu önerilmektedir. Tamamen bağlı katmanı küresel ortalama havuzlama katmanıyla değiştirerek ve son iki yoğun katman arasına bir bırakma katmanı ekleyerek mimari değiştirilmiştir.

Deneyler, farklı eğitim dönemleri, farklı veri artırma durumları ve tüm 3D görüntülerin veya yalnızca akciğer dokularının kullanıldığı birçok senaryoda uygulanmaktadır. Gerçekleştirilen deneylerde, en iyi dönem 25 olduğunu ve en iyi durumun tüm akciğer 3D görüntüsünü kullanmak yerine sadece akciğer dokularını kullanmak olduğunu göstermektedir. Eğitim için %70 doğruluk ve %71 doğrulama doğruluğu elde ediyoruz.

Gelecekteki çalışmalar için, 3D CNN mimarisi aşağıdaki özelliklerle değiştirilebilir:

- Verileri daha iyi ölçeklendirmek için standardizasyon yöntemini uygulamak.
- Resnet101 gibi transfer öğrenme yöntemini kullanarak verileri eğitmek veya InceptionV3 derin sinir ağları.
- Verilerin artırılması.

3D MEDICAL IMAGE RECOGNITION USING DEEP LEARNING NETWORKS

SUMMARY

At the present time, the use of deep learning networks is one of the most important issues that have increased reliance on them at the present time in order to take advantage of its capabilities in multiple fields and its constantly evolving capabilities, but the classic interaction between deep learning networks and between ordinary images based on camera or video can be apparent.

The two-dimensionality falls short of the aspirations and developments required to advance in this field, as some serious diseases need a lot of accuracy in identifying them, and the uses related to dealing with violations, such as traffic, for example, require accurate and sound vision of the site, all of this necessitated work to deal with the view with its new and accurate face Which is represented by a three-dimensional perspective because of its benefits that may not be available in the two-dimensional perspective, in addition to enriching the neural network with features, which gives more accuracy in its ability to identify and accuracy in performance.

Despite the wide options offered by two-dimensional images in the field of recognition, they still fell short of achieving the desired goal, and this called for continuous development in order to rely on the latest vision technologies, processing and dealing with them through deep learning networks. Deep learning networks are networks capable of manipulating 2D images in large numbers in order to extract basic information including (edges, lines, points, curves, angles, points, etc.).

Convolutional neural networks are very similar to regular neural networks in that they consist of neurons, which in turn have variable weights and displacements. Each neuron receives some input data, computes the standard product, and optionally uses a nonlinear activation function. The entire network, as before, is the only distinguishable evaluation function: from the initial set of pixels (the image) at one end to the probability distribution of belonging to a particular class at the other end.

These networks still have a loss function (e.g., SVM/Softmax) on the last (fully connected) layer, and all the advice and recommendations that have been given regarding normal neural networks are also relevant to convolutional neural networks.

The architecture of convolutional neural networks explicitly includes image acquisition at the input, which allows us to consider some properties of the input data in the network architecture itself. These characteristics allow you to implement the direct distribution function more efficiently and significantly reduce the total number of parameters in the network.

Convolutional Neural Networks (CNNs) in deep learning use two-dimensional images in their matrix form in order to process them and extract features from them, and these matrices must be processed and search for the most important locations of the two-dimensional image. Resulting in more accurate results than the results of the human

contestants. Convolutional neural networks use the fact that the input data are images, so they form a structure that is more sensitive to this type of data unlike ordinary neural networks, layers in a convolutional neural network arrange neurons in 3 dimensions - width, length, and depth. So, the neurons in one layer must be connected to a small number of neurons in the previous layer, instead of connecting to all the previous neurons in the layer. Moreover, the image output layer will have a dimension of 1×1×10, because when we approach the end of the neural network, we will reduce the image size to the vector class estimates located along the depth

In this research, a full 3D medical image recognition model based on deep learning have been designed and implemented based on 3D convolutional neural network 3DCNN. 3D image dataset "MosMedData" is used for building the system. The used dataset consists of 100 normal and 100 abnormal 3D images each of which includes 64 slices resulting in 12800 slices. The used dataset is decomposed into 70% as a training dataset and 30%for validation dataset.

Four basic steps are involved to build the proposed deep learning model. The first is preprocessing step, then the data augmentation process, after that there is the deep learning model configuration and the last step is the training and evaluation part. Forthe preprocessing step, rotate the volumes by random degrees, to get different image rotation results so that the model will be robust orientation variations.

Scale HU values to be between 0 and 1(Convert pixel density values into a home field (CT number) by calculating pixel density values in different CT images and in order to compare these measurements statistically, the grayscale value is from 0-255 and we need to convert the values to HU (-100 to 1000)).

Change the size, width, height, and depth (This change requires cropping in order to obtain the same image in several consistent or inconsistent sizes, as well as making it cropped sometimes and at other times while maintaining the size, in order to increase the number of the same perspective of the image).

Resizing, rescaling and normalizationare used to get the lung tissues. The second step isthe data augmentation and data increase which are applied in order to increase the number of training images by using a rotated, flipped and contrast-modified copies of the same image so that the training will avoid the problem of fake training. The shuffling operation is also applied in order to provide training sample in a shuffle order to the training process.

For the training model, we suggest using a modified version of 3D CNN network. We modified the architecture by replacing the fully-connected layer by the global average pooling layer and insert a dropout layer between the last two dense layers.

After that, the dataset is splitted into 70% for training, and 30% for validation. The experiments are applied in many scenarios using different training epochs, different data augmentation cases and using either the entire 3D images or the lung-tissues only.

Experiments shows that the best epochs number is 25 and the best case is using the lung tissues only rather than using the entire lung 3D image. We get 70% accuracy for training and 71% validation accuracy.

For future work, 3D CNN architecture can be modified by the following features:

- Implementing the standardization method to better scale the data.

- Training data using transfer learning method such as Resnet101 or augmenting data with InceptionV3 deep neural networks.

- Increase the volume of data.

1. GİRİŞ

1.1. Derin Öğrenmeye Giriş

Derin öğrenme ağlarının kullanımı, çoklu alanlardaki yeteneklerinden ve sürekli gelişim gösteren becerilerden yararlanmak için günümüzde güven oranını artıran en önemli konulardan biri olarak kabul edilmektedir. Bazı ciddi hastalıkların tanımlanmasında çok fazla doğruluk gerektiğinden iki boyutluluk, ilgili gelişmelerin gerisinde kalmaktadır. İki boyutlu perspektif içerisinde yer almayan faydalar, üç boyutlu bir perspektifi zorunlu hale getirmiştir [1].

Tanıma alanında iki boyutlu görüntülerin sunduğu geniş seçeneklere rağmen, yine de istenen hedefe ulaşmakta yetersiz kaldığı bir gerçektir ve derin öğrenme yolu ile son görme teknolojilerinin işlenmesi de devamlı bir gelişim gerektirmektedir [2].

1.2. Derin Öğrenme Ağlarının Çalıştığı Görüntüler

Bu ağların son (tamamen bağlı) katmanda hala bir kayıp işlevi (örneğin, SVM/Softmax) vardır ve normal sinir ağlarıyla ilgili olarak verilen tüm tavsiyeler aynı zamanda evrişimli sinir ağlarıyla da ilgilidir.

Evrişimli sinir ağlarının mimarisi, giriş verilerinin bazı özelliklerini ağ mimarisinin kendisinde hesaba katmamıza izin veren görüntü alımını açıkça içermektedir. Bu özellikler, doğrudan dağıtım işlevini daha verimli bir şekilde uygulamanıza ve ağdaki toplam parametre sayısını önemli ölçüde azaltmanıza olanak tanımaktadır [3].

Derin öğrenmede Evrişimsel Sinir Ağları (CNN'ler), işlemek ve onlardan özellik çıkarmak için matris biçiminde iki boyutlu görüntüleri kullanmakta ve bu matrisler işlenerek iki boyutlu görüntünün en önemli konumlarını aramaktadır. İnsan kaynaklı sonuçlardan daha doğru sonuçlarla sonuçlanmaktadır.

Derin öğrenme konusunda eğitilmiş araçlar ve aktüatörler artık bize, geniş bir genetik sendrom veritabanıyla ilişkili insan malformasyon vakalarını analiz etmek için kullanılan olağandışı yüz morfolojisi analizi (FDNA) olan 360 derecelik bir kamera görüntüsü ve önizlemesi sunmaktadır.

Görüntü tanımadaki gelişmeler ve derin öğrenme tekniklerinin çeşitli görsel sanatlar uygulamalarına uygulanmasının her geçen gün artması ile yakından ilgili olan DNN'ler, bir resimde ve belirli bir çağda kullanılan stili belirleme ve ayrıca sinirsel şekil silme- kalıbı yakalama yeteneklerini kanıtlamıştır. Rastgele görsel girdi alanlarına dayalı göz alıcı görüntüler oluşturmanın yanı sıra derin öğrenme teknikleri ile rastgele bir görüntü veya videonun görsel açıdan tatmin edici bir şekilde belirtilmesi ve uygulanması mümkün hale gelmektedir [4].

Ancak tüm bu gelişmeler ışığında, hayatımızın önemli bir alanına giren ve günlük hayatımızda güvenebileceğimiz üç boyutlu perspektifin temsil ettiği görsel tekniğinin de derin sinir ağları ile yazışma alanında gelişme ihtiyacı doğmuştur DNN [5].

Evrişimli sinir ağları, giriş verilerinin görüntü olduğu gerçeğini kullanmaktadır. Bu nedenle bu tür verilere daha duyarlı bir yapı oluşturmaktadır. Özellikle sıradan sinir ağlarının aksine, bir evrişimli sinir ağındaki katmanlar, nöronları 3 boyutta düzenleyerek genişlik, uzunluk ve derinlik katmaktadır.

Bu nedenle, bir katmandaki nöronlar, katmandaki önceki tüm nöronlara bağlanmak yerine, önceki katmandaki az sayıda nörona bağlı olmalıdır. Ayrıca görüntü çıktı katmanı 1×1×10 boyutunda olmalıdır. Çünkü sinir ağının sonuna yaklaştığımızda görüntü boyutunu derinlik boyunca yer alan vektör sınıfı tahminlerine (üçüncü boyut) düşürmek gerekmektedir [6].

1.3. 3D Görüntü İşleme İçin Derin Öğrenme

3B veriler veya 3B görüntüler, geleneksel 2B verilerden ziyade sağladıkları faydalar ve bilgiler nedeniyle son zamanlarda çok fazla ilgi görmektedir. Örneğin tıbbi 3B veriler, tümör tespiti, segmentasyon ve sınıflandırma gibi birçok temel görevi yerine getirmek için analiz edilebilen ve işlenebilen 3B verileri arasında yer almaktadır [7]. Kural tabanlı algoritmalar ve manuel özellik çıkarma yöntemleri gibi geleneksel segmentasyon ve sınıflandırma yöntemlerinin zaman alıcı, düşük performans gibi bazı sınırlamaları vardır ve çok daha fazla bilgi gerektirmektedir [7]. Bununla birlikte, örneğin CNN'ler gibi derin öğrenme teknikleri, evrişim katmanlarının bir parçası olarak nitelendirilen hiyerarşik özelliği ortaya çıkardıkları için manuel özellik ortaya çıkarma gereksinimi duymamaktadır. Bu tür derin öğrenme mimarisinin bir başka yararıda GPU yardımcı programlarının etkin kullanımıdır, böylece 3B devasa verilerle

başa çıkma yeteneği, yüksek hesaplama yetenekleri sayesinde daha kolay ve daha az zaman alıcı hale gelmektedir.

3D tıbbi veriler, X-ray, CT taraması, MRI, Ultrason ve diğer birçok olası kaynak gibi farklı kaynaklardan elde edilmektedir [8]. Bu tür 3D görüntüler, vücut dokularının ayrıntılı 3D görünümünü içermekte, böylece doktorlar daha etkili bir şekilde kararlarını tespit edebilmekte ve türetebilmektedir. Bir 3D görüntünün tek bir veri dosyasında, herhangi bir görüntü işleme algoritmasının her bir adımında K-derinlikli çerçeveler işlenmesi gerekmektedir (yani, 64 derinlikli 3D dosyası, bu tekliyi uygulamak için her adımda işlenmesi gereken 64 görüntü içerecektir).

Bu hesaplama zamanı, geleneksel bilgisayar bilimi algoritması ile ele alınamamaktadır. Bu nedenle kesinlikle bazı derin öğrenme GPU destekli metodolojilere ihtiyaç duyulmaktadır.

1.4. Sistem Amaçları

Önerilen sistem amaçlarını aşağıdaki şekilde sıralamak mümkündür;

- Derin öğrenme ağlarını (CNN) kullanarak 3B görüntülerle başa çıkmak için yeni bir sistem geliştirmek.
- Evrişimli sinir ağının performansını artırmak ve üç boyutlu görüntülerle uğraşma yollarını geliştirmek.
- Derin öğrenme ağları kullanılarak 3 boyutlu görüntülere dayalı veri tabanları ile tanıma performansının iyileştirilmesini sağlamak.

1.5. Sistemin Önemi

İki boyutlu görüntüleri tanımlama sürecinde sinir ağlarının yeteneğinden yararlanma süreçleri, genellikle deneyim eksikliğinden veya doğruluk eksikliğinden ve düşük performanstan ve kullanımında ortaya çıkan teknik ve gelişmelerin gerektirdiği ihtiyaçtan müzdarip olmuştur. Sistemlerin geliştirilmesine dahil olan iki boyutlu muadilleri yerine üç boyutlu perspektiflerin tercih edilmesinin en önemli nedenlerinden biri de bu olmuştur. Küresel görselliğin daha gerçek hale getirilmesi gibi gelişmeler, medeni teknik gelişme yollarının ayrılmaz bir parçası haline gelen derin öğrenme ağlarının gerçekliğinin ve buna yapay zekanın katkılarının geliştirilmesini gerekli hale getirmiştir.

Bu bağlamda çalışmamız, üç boyutlu görüntülerle uğraşma sürecini ve özellikle Evrişimsel sinir ağı olmak üzere derin öğrenme sinir ağları ile uyumluluk yollarının kesin olarak belirlenmesini amaçlamaktadır.

Araştırma ayrıca görüntülerin iki boyutlu veya üç boyutlu olup olmadığını bunun için yapılan testlere güvenerek karşılaştıracaktır.

Ağı, örneğin tümör gibi, bunları tespit etmek için 3D CT'ye veya rezonatör görüntülere bağlı hastalıklarla başa çıkmak için eğitilmesinin vurgulandığı bu çalışmada örneğin, bu bireylerden alınan örneklerin üç boyutlu görüntüleri gibi veritabanında bulunanlara göre. Hastalık ve daha sonra iki boyutlu ve üç boyutlu vakalar üzerinde test etme, sonuçları gösterme ve analiz etme hedeflenmiştir.

1.6. Sistem Sorunu

3B derin öğrenme metodolojileri, 3B verilerin türetilmiş temsillerini kullanmaktan doğrudan ham verileri kullanmaya geçmiştir. Yöntemler açısından, 2B evrişimli sinir ağlarının 3B verilere uygulanması, nesne sınıflandırma ve anlamsal segmentasyon gibi görevlerin performansını büyük ölçüde artıran sistemin 3B sahnelere uyarlanmış bir yönteme dönüştürülmesini sağlamıştır.

Bu çalışmada sistemin 3B nesne sınıflandırmasına, segmentasyonuna odaklanmak ve 3B perspektiflerle çalışmanın yollarını belirlemek amaçlanmıştır.

3B verilerin büyük hacimleri nedeniyle işlenmesi için daha fazla hesaplama süresi gerekmektedir, bu nedenle bu tür verilerle çok zaman verimli ve yüksek performanslı bir şekilde başa çıkabilen bir derin öğrenme 3B mimarisine ihtiyaç duyulmaktadır.

1.7. Sistem Hipotezleri

- Önerilen sistem Hipotezleri şu şekilde sıralanabilmektedir;
- 3B veriler, işleme, segmentasyon ve sınıflandırma için 3B derin öğrenme mimarisi gerektirir.
- 3D-CNN gibi derin ağları 3D verilerle kullanmak performansı artıracak ve zamanı kısaltacaktır.
- 3D veri büyütmeyi 3D veriye uygulamak performansı artıracaktır.
- 3D görüntüden akciğer olmayan dilimlerin çıkarılması, derin öğrenme modelinin performansını artıracaktır.

- Çıktı katmanının çıkış katmanından önce eklenmesi ve tam bağlantılı katman yerine global ortalama havuzlamanın kullanılması modelin performansını iyileştirecektir.

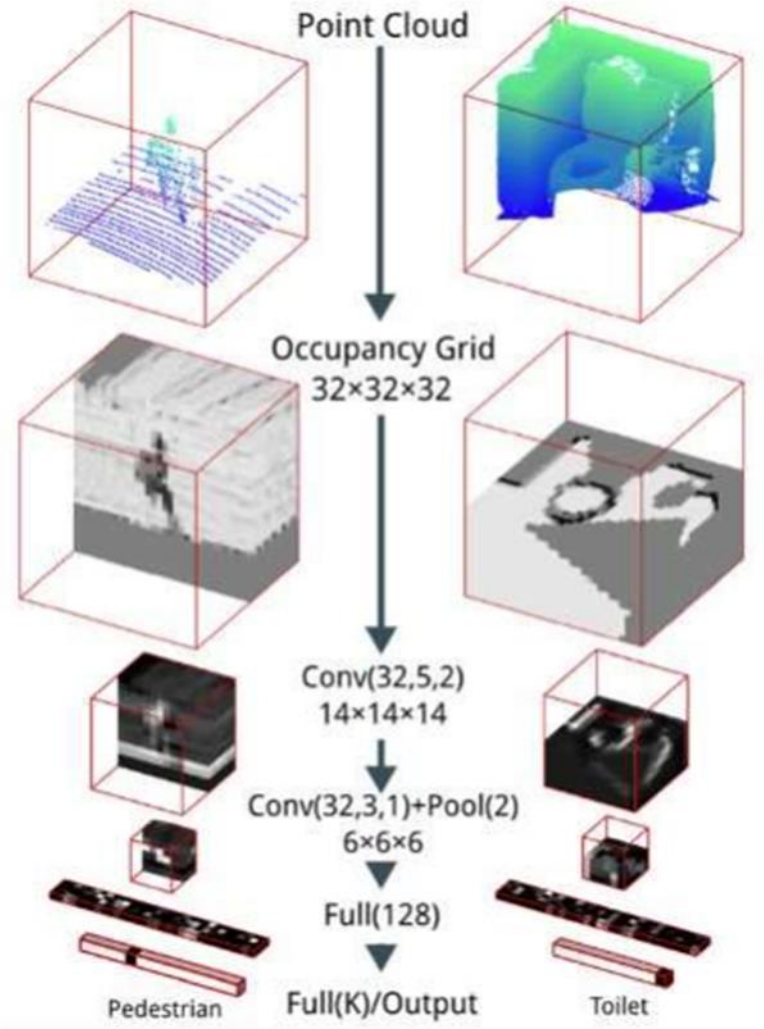
1.8. Malzemeler ve Yöntemler

Bu bölümde önerilen yazılım, veri seti ve metodolojiler listelenecek, açıklanacak ve tartışılacaktır. Ama önce, önerilen sistemimin tanımlaması için ana metodijilerden bahsetmek uygun görülmüştür.

1.8.1. Önceki derin tanıma tekniklerinde 3b görüntüleri kullanma yöntemleri

Maturana ve Scherer, 2015 yılında önerilen VoxNet [18], belirli bir voksel ızgarası ile nesne sınıflandırma görevlerinde mükemmel performans elde eden ilk kişi olmuştur. VoxNet, her vokselin uzayda bulunma olasılığını içerdiği bir doluluk ızgarası kullanmaktadır. Bunun bir avantajı, ağın serbest olduğu bilinen vokselleri ve dolulukları bilinmeyen vokselleri ayırt etmesine izin vermesidir.

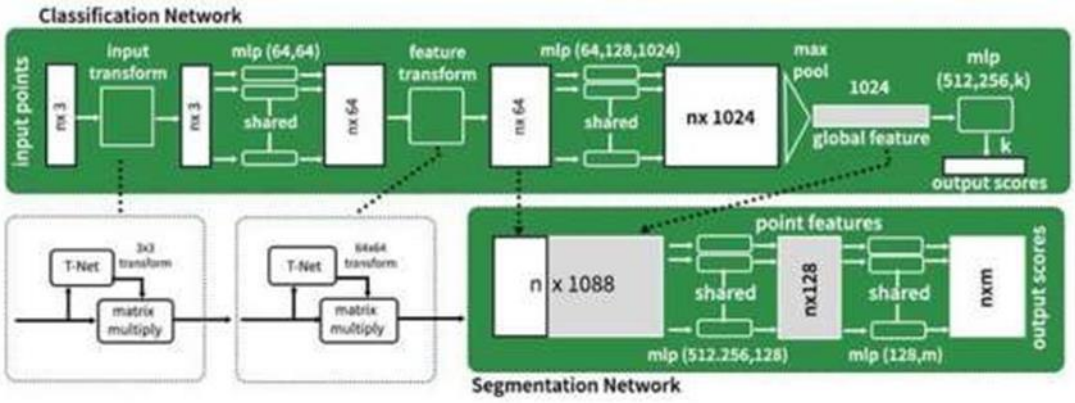
Mimari, iki evrişim katmanından ve bir havuzlama katmanından oluşmakta ve çıktı kategorisi puan vektörünü hesaplamak için iki tam bağlantılı katman kullanılmaktadır. VoxNet, gerçek 3Böğrenmeye doğru büyük bir adımıtemsiletmektedir. Ancak voksel ızgaralarının hala bazı eksiklikleri vardır. İlk olarak, nokta bulutlarına kıyasla çözünürlüğü kaybetmektedir. Çünkü çok yakınlarsa farklı noktalar aynı vokselde kümelenecektir. Bazen voksel ızgaraları gereksiz yere yüksek bellek kullanımına neden olabilmekte ve bunun nedeni, boş ve bilinmeyen alanı temsil etmek için bellek tüketmeleri olarak gösterilmektedir. Öte yandan, nokta bulutu yalnızca bilinen noktaları içerecektir. Aşağıda yer alan Şekil **1.1.**VoxNet 3D ağının mimarisini göstermektedir.



Şekil 1.1. Voxnet Derin Ağ Mimarisi

Voksel tabanlı yöntemleri kullanmanın sorunları göz önünde bulundurulduğunda, son çalışmalar doğrudan ham nokta bulutu verileri üzerinde çalışan mimarilere odaklanmıştır [19].

Qi ve diğerleri tarafından önerilen PointNet. 2016'da [20] bu düzensiz 3B verileri işlemek için en erken yöntem olarak kabul edilmiştir. Nokta bulutunda bize N noktası verildiğinde, ağın bu N giriş noktasının tam düzenine göre sabit olan benzersiz bir özellik öğrenmesi gerekmekte, çünkü bu girişler Sinir Ağlarına verilmektedir. Noktaların sırası temeldeki geometriyi etkilememektedir. Şekil 1.2. PointNet mimarisini içermektedir.



Şekil 1.2. VoxNet Derin Ağ Mimarisi [21]

Ayrıca ağ, nokta bulutu döndürme, dönüştürme ve diğer dönüşümler için sağlam olmalıdır ve ölçekleme işlemleri tahmin sonuçlarını etkilememelidir.

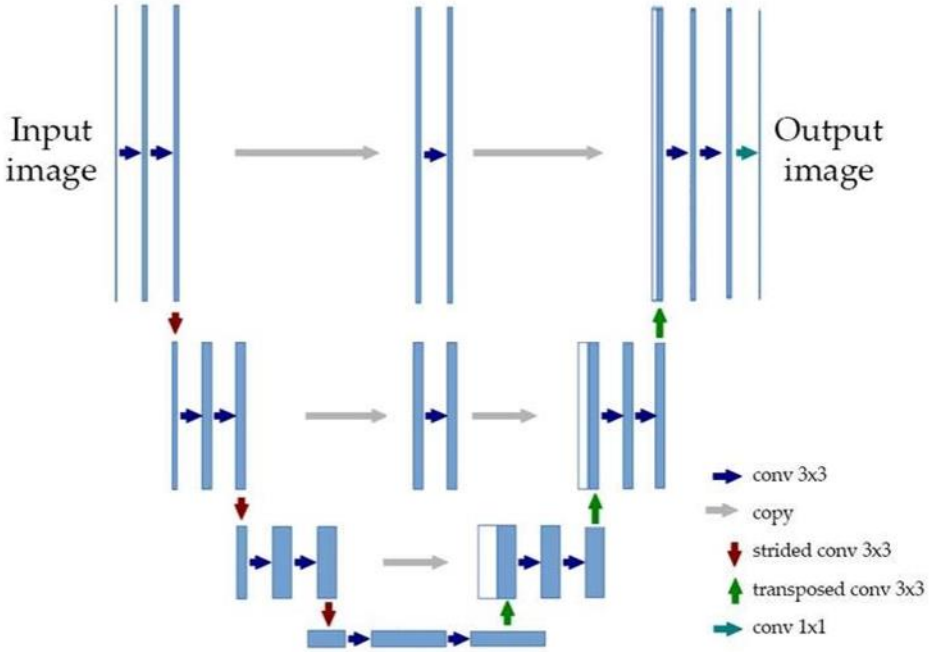
PointNet, bir nokta bulutunun geometrik dönüşümüne değişmez olan bir temsili öğrenme zorluğunu karşılamak için, giriş noktası bulutuna afin dönüşüm uygulayan T-Net adlı küçük bir ağ kullanılmaktadır. Afin dönüşüm, aynı görüntülerin çok daha fazla kopyasını elde etmek için görüntülerin dönüştürüldüğü (döndürüldüğü, ölçeklendirildiği, çevrildiği vb.) çok güçlü bir mekanizmadır.

Bu mekanizma, görüntü sayısını artırmakta ve eğitilmiş ağı tüm bu dönüşümlere karşı dayanıklı hale getirmektedir. Bu kavram uzamsal dönüşüm ağına benzemekte, ancak çok daha basit olarak ifade edilmektedir. Çünkü yeni bir katman türü tanımlamaya gerek yoktur.

T-Net (Şekil 1.3.) öğrenilebilir bir parametre bileşimidir, bu parametre PointNet'in giriş noktası bulutunu sabit ve standartlaştırılmış bir alana dönüştürmesini ve böylece tüm ağın en uygun değişikliklere bile dayanıklı olmasını sağlamaktadır.

Genel PointNet mimarisi, en temel yöntemleri ve T-Net ve çok katmanlı katmanları devralır, nokta bulutları için özellik temsilleri oluştururlar.

Ancak, nesne sınıflandırmasına ek olarak PointNet, nesnelerin ve sahnelerin semantik bölümlemesini de destekler.



Şekil 1.3. VoxNet Derin Ağ Mimarisi [22]

2. LİTERATÜR ÖZETİ

2.1. İlgili Çalışmalar ve Önceki Çalışmalar

3B görüntü bölütleme ve sınıflandırma alanında birçok çalışma yapılmıştır. Xiaojuan Qi et al. [9] tarafından RGB görüntüleri için olağan semantik bölütleme görevlerinden farklı olan anlamsal RGBD bölütleme gerçekleştirilmiştir. Burada ayrıca D kanalından daha ayrıntılı bilgileri de düşünebilmek mümkündür. Dolayısıyla bu tür bir görev için 2B görünümü eklem mantığının 3B geometrik bilgileriyle birleştirilmesi gerekmektedir. RGBD görüntülerinin nasıl bölüneceği ile ilgili olarak, önceki çalışma esas olarak aşağıdaki yöntemleri içermektedir.

Derinlik haritası doğrudan başka bir giriş görüntüsü olarak kullanılmaktadır. İki CNN mimarisi, özellikleri çıkarmak için renk ve derinlik bilgilerini işlemek için kullanılmaktadır. Bu, çift sayım ve depolama tüketimi ile sonuçlanmaktadır. Bu ayar, belirli hataların oluşmasına neden olabilecek gerçek konumundan ayrılmış bir 2B görüntü pikseli ve bir 3B alan olacaktır. 2B uzaydaki pikseller ayrık olarak kabul edildiğinden, onlarla elde edilen komşuluk ilişkisi, uzaydaki iyi komşuluk ilişkisine karşılık gelmeyebilir [9].

Diğer bir seçenek olan vokselleştirilme bir 3B uzayda ve bir de 3B CNN ‘de kullanmaktır. Bu tür bir yöntem, bağlam bilgisini üç boyutlu uzayda daha iyi kullanışlı olmaktadır. Bununla birlikte, 3D voksel ağlarının yüksek hesaplama maliyeti nedeniyle, peyzaj ölçeklemenin yüksek çözünürlüğü ve karmaşıklığı nedeniyle bazı zorluklar ortaya çıkarmaktadır. Ek olarak, bir 3B alan tanımlamak da ek hatalara neden olabilmektedir.

2018 yılında Chen ve ark. [10] tarafından, MRI tıbbi görüntülerinin özelliklerini çıkarmak için bir 3D Yoğun Bağlantılı Süper Çözünürlüklü Ağları (DCSRN) tanıtılmıştır. Yüksek çözünürlüklü 3D MRI görüntülerinin restorasyonu amacıyla 1113 görüntüden oluşan bir veri seti kullanılmıştır. Sonuçlar, DCSRN ağlarının 3DFSRCNN'den 4 kat daha hızlı eğitildiğini göstermiştir.

2019 yılında Chen ve ark. [11] tarafından toplanan 3Dseg-8 veri seti ile ilgilenmek için ResNet'e dayalı Med3D adında bir 3D ağ oluşturulmuştur. Akciğer

segmentasyonu amacıyla transfer öğrenimininde kullanılan bu ağlarla yapılan deneyler (80) görüntü üzerinde uygulanmış ve %87.16 ile %93.82 arasında bölütleme oranları elde edilmiştir. 2020 yılında Satya P. Singh ve ark. [7] tarafından, tıbbi görüntüleri analiz etmek için 3D CNN'leri kullanılmış, makine öğreniminin köklerinden 3D CNN'lerin nasıl geliştirildiğinin tarihi takip edilmiştir. 3D CNN'lerin özet bir matematiksel tanımını sağlanmış ve daha önce tıbbi görüntüler için uygulanması gereken ön işleme aşamaları sunulmuştur. 3D CNN'ler ile beslenmişler ve sınıflandırma, segmentasyon, tespit ve lokalizasyon gibi çeşitli tıbbi alanlarda 3D CNN'ler (ve varyantları) kullanılarak 3D tıbbi görüntüleme analizi alanındaki önemli araştırmalar, ilgili zorluklar tartışılarak gözden geçirilmiştir.

Üç boyutlu (3D) CNN'ler, tıbbi görüntüleri analiz etmek, makine öğreniminin köklerinden 3D CNN'lerin nasıl geliştirildiğinin tarihini izlemek, 3D CNN'nin kısa bir matematiksel tanımını sağlamak ve ayarlama için gerekli ön işleme adımlarını sunmak veya tıbbi görüntüleri 3D CNN'lere yerleştirmeden önce iyileştirmek için kullanılmıştır. Sınıflandırma, segmentasyon, tespit ve lokalizasyon gibi çeşitli tıbbi alanlarda 3B CNN'ler (ve varyantları) kullanılarak 3B tıbbi görüntüleme analizi alanındaki önemli araştırmalar gözden geçirilmiştir. Tıbbi görüntüleme (ve genel olarak derin öğrenme modellerinin kullanımı) alanında 3D CNN'lerin kullanımıyla ilgili zorlukların ve bu alandaki potansiyel geleceğin eğilimleri tartışılmıştır [12].

L. Shi ve diğerleri [13] tarafından, "Kırmızı" renk modeli kullanılarak tek bir stereo görüntüden gerçek zamanlı olarak gerçekçi, renkli, üç boyutlu bir hologram üretebilen derin öğrenme teknolojisine dayalı bir dizi bilgisayar hologram işleme operasyonu gerçekleştirilmiştir. "Yeşil Mavi". Araştırmacıların evrişimli sinir ağı (CNN) bellek açısından oldukça verimli bulunmuştur (620 kilobayttan daha az bellek kapasitesi tüketir) ve kişisel kullanıma yönelik bir GPU üzerinde çalıştırıldığında 1920 x 1080 piksel çözünürlük elde etmek için 60 Hz frekansında çalıştığı tespit edilmiştir [13]. Yakın zamanda Alalwan ve ark. [14] tarafından, karaciğer ve tümör segmentasyonu amacıyla derin öğrenme DenseNet-569 kullanılarak etkili bir 3D semantik segmentasyon önerilmiştir. Deneylerini standart LiTS veri seti üzerinde uygulamışlar ve DenseNet modelinin karaciğer ve tümör segmentasyonu için sırasıyla %96,2 ve %69 doğrulukla etkili olduğunu bulmuşlardır.

Serte ve Demirel [15] tarafından, 3D akciğer BT görüntülerinde COVID-19 varlığını tahmin etmek için bir derin öğrenme sistemi tasarlanmıştır. ResNet50 ağlarını

kullanmışlar ve MosMed veri setinde (%70 ila %84) doğruluk elde etmişlerdir. Yaklaşımlarının ana sorunu, birlikte çalışması gereken çok sayıda ResNet ağı nedeniyle yüksek hesaplama süresi olmuştur 2021'de Chen ve arkadaşları [16] tarafından, 3D görüntüler kullanılarak 3D AlexNet'e dayalı bir prostat kanseri segmentasyon modeli önerilmiştir. Modellerini değerlendirmek için 500 MRI görüntüsünden oluşan bir veri seti kullanılmıştır. Yöntemleri karşılaştırma amacıyla ResNet50 ve Inception-v4 derin ağlarını kullanmayı içermiştir. %92,1 doğruluk ve %89,6 özgüllük elde edilmiştir. 3D AlexNet'in ResNet ve Inception-V4'e kıyasla daha güçlü ve daha yüksek performansa sahip olduğu sonucuna varmışlardır. Ancak bu tür derin ağların hesaplama süresi 3D-CNN'den çok daha fazla olduğu bilinmeketedir.

Karimiet al. [17] tarafından, hesaplama süresini azaltmak için bitişik görüntü yamaları arasındaki öz-dikkat temelinde, evrişimsiz bir tıbbi görüntü bölütleme modeli tantılmıştır. Yaklaşımlarının ana fikri, her 3 boyutlu görüntüyü parçalara bölmek ve her biri için 1 boyutlu yerleştirme oluşturmaktır. Deneyleri Beyin kortikal plakasına uygulamışlar ve doğruluk için 87.9 puan almışlardır.

2021'de Jefferson ve Shanmugasundaram, özellik çıkarma için 3D-CNN ve sınıflandırma için KNN kullanarak 3D-MRI görüntülerinin tümör sınıflandırması için bir derin öğrenme sistemini tanıtmıştır. Yaklaşımlarını 300 MRI görüntüsünden oluşan bir veri kümesine uygulamışlardır. Gerçek pozitif, gerçek negatif, yanlış pozitif ve yanlış negatif oranları için %62,99, %33,75, %1,17 ve %2,09 elde etmişlerdir.

3. YÖNTEMLER

3.1. Genel Sistem Şeması

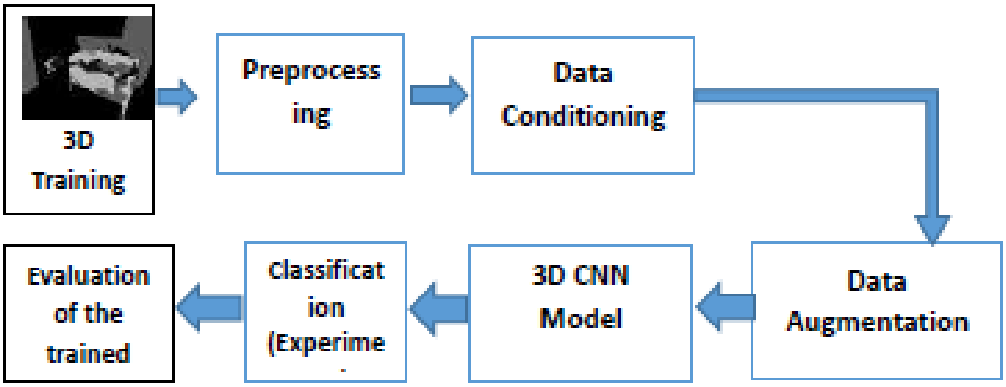
Bu çalışmada, aşağıdaki adımların gerçekleştirilmesi önerilmektedir;

- 3B-CNN'leri kullanarak bir 3B nesne sınıfı tanıma sisteminin uygulanması gerekmektedir.
- Hem eğitim hem de sınıflandırma için GPU'ları kullanarak bu sistemin hızlandırılması gerekmektedir. Derin öğrenme, özellikle 3B verilerde (dev veriler) eğitim için çok daha fazla zaman gerektirdiğinden, hesaplama süresini azaltmak için GPU ile çalışmaya geçmek çok önemlidir. Bunun için donanımım üzerinden kodlama yapılması durumunda bazı DL kitaplıkların kurulması gerekmektedir ancak Google Colab kullanılması durumunda "GPU" seçeneğini seçerek GPU Runtime ortamını çok kolay bir şekilde etkinleştirebilmek mümkündür. Böylece notebook dosyası bulutta sağlanan ücretsiz GPU'yu kullanacaktır.
- Sistem girdileri, nesneleri içeren nokta bulutlarının parçalı bölgelerinden oluşmakta, bu nokta bulutları, düşük maliyetli RGB (RGB-D) derinlik sensörleri tarafından oluşturulmuş gibi yapay olarak simüle edilmektedir. Bu öneriyi gerçekleştirmek için aşağıdakilerin sağlanması gerekmektedir.

1) CNN ağlarının sistem uygulamasını sağlamak için teorik arka planının analizinin gerçekleştirilmesi gerekmektedir.
2) 3B verilerin hacimsel temsillerinin oluşturulması gerekmektedir.
3) 3D CNN’nin tasarımı, uygulaması ve test edilmesi için bir ara adım olarak sunucuların derlenmesi ve yapılandırılması gerekmektedir.

3.1.1. Sistem blok şeması

Önceki çalışmalar (3B nesne tanıma alanı) ve önceki teklif bölümleri açısından, metodolojinin tanımlanması bu bölümde gerçekleştirilmiştir. Aşağıdaki şekil sistemin genel aşamalarını göstermektedir



Şekil 3.1. Derin Öğrenme Ağlarını Kullanarak 3B Görüntü İşlemenin Blok Şeması

3.2. Önerilen Malzemeler

3.2.1. Veri kümesi

MosMedData: Üç boyutlu görüntülerin bulunduğu bu veriler, bu amaçla kullanılabilecek verilere göre birçok bölüme ayrılmakta olup, kişilerin tomografi, manyetik rezonans veya diğerleri gibi çeşitli tıbbi tekniklerle muayene edilmesi için kullanılan bir veri setidir. Bununla başa çıkmanın amacı, bu veritabanına dayalı derin öğrenme ağlarını kullanarak üç boyutlu görüntülerle başa çıkmak için tahmine dayalı bir sınıflandırıcı oluşturmaktır [14]. Veri kümesine bağlantı https://mosmed.ai/en/

3.2.2. Programlama araçları

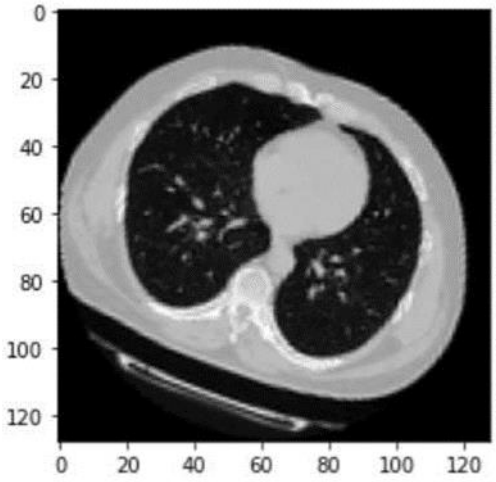
Aşağıdaki yazılımı kullanmanızı öneririz:

- Derin öğrenme aracı ve google sürücü bağlantısı ve görüntü işleme araçları dahil Google Colab.
- CNN, Keras, AlexNet ve PyTorch gibi önceden eğitilmiş derin öğrenme açık kaynak paketleri.

3.3. Önerilen Yöntem

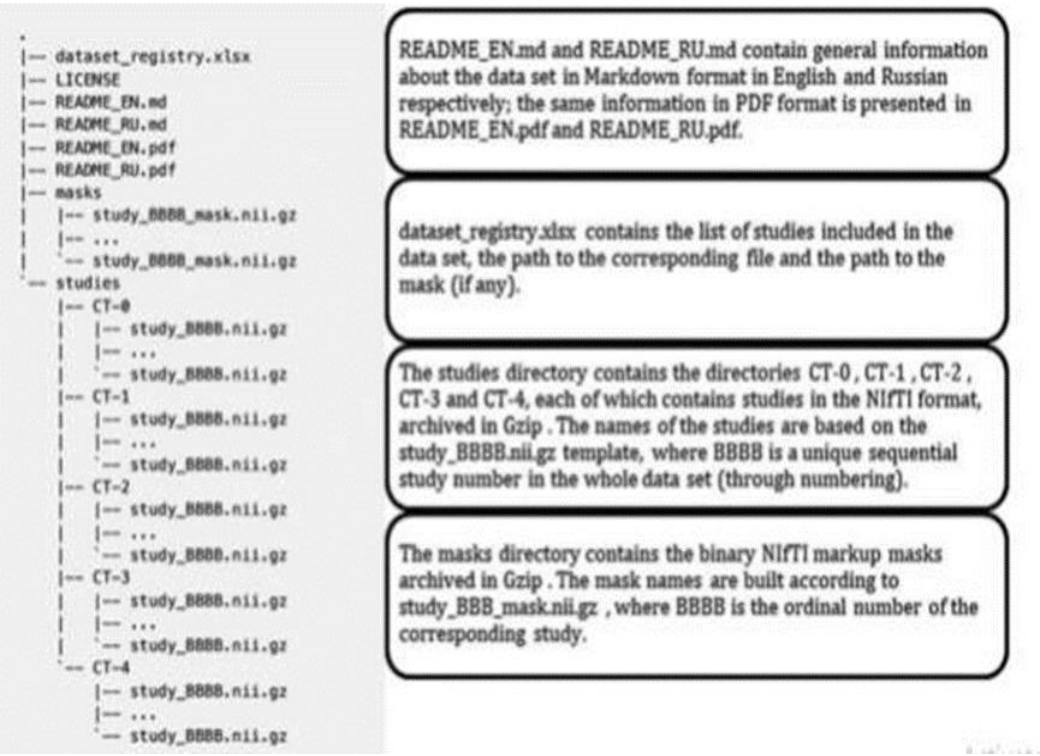
3.3.1. Veri kümesinin alınması

Önerilen 3B veri seti, tıbbi alanın farklı değerli verilerini içermektedir. Aşağıdaki bağlantıdan elde edilen COVID-19 hastalığının özel bir veri seti olan "MosMedData" veri seti elde edilmiştir [24]. https://mosmed.ai/en/datasets/covid19_1110/ Şekil **3.2.** bu veri kümesinin bir örneğini içermektedir.



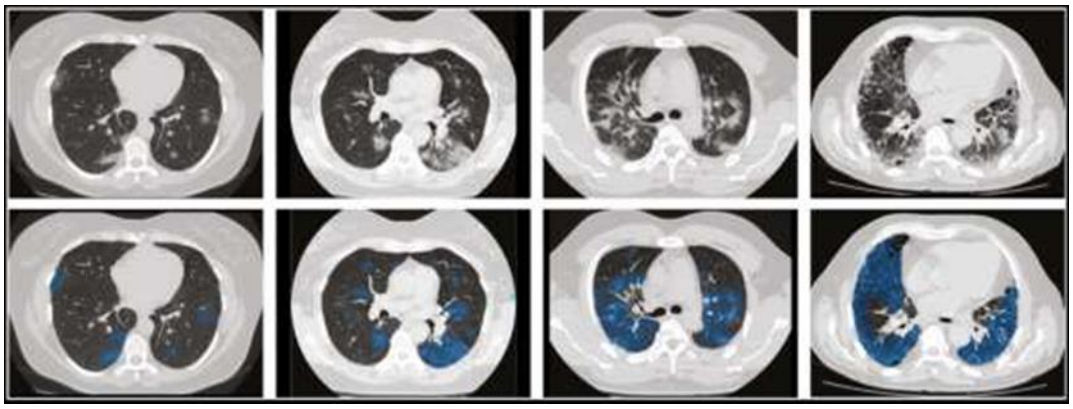
Şekil 3.2. Kullanılan veri kümesinin bir örneğinin bir dilimi örneği

Şekil 3.3**.** MosMed veri kümesinin mimarisini göstermektedir. Veri setinde CT-0, CT-1, CT-3, CT4 ve CT-5 olmak üzere her biri .ZIP dosyalarında arşivlenmiş NIFTI formatında vaka veya vaka görüntülerini içeren 5 klasör bulunmaktadır. Her zip dosyası 64 CT çerçevesine sahip bir hastayı temsil etmektedir [25].



Şekil 3.3. Kullanılan veri kümesinin hanedan görünümü

İşte farklı hastalık durumlarına sahip COVID-19 hastalarının görüntülerinden bazı örnekler aşağıda sunulmuştur. COVID-19 hafiften (CT-1) kritik (CT-4) şiddetine. Soldan sağa doğru sınıflandırılmaktadır [26].



Şekil 3.4. Farklı hastalık durumu. COVID-19 hafiften (CT-1) kritik (CT-4) şiddetine. Soldan sağa

3.3.2. Ön işleme adımları

3B görüntü sınıflandırmasında en iyi performansı elde etmek için gerçekleştirilmesi gereken birçok ön işleme adımı vardır. Bu ön işleme adımları aşağıdaki şekilde sıralanmaktadır;

- (KERAS, PYTORCH, tensorflow, NUMPY, scipy, matplotlib) gibi bu seçilen veri kümesiyle (resimler üzerindeki tüm hesaplama işlemleri, matris işlemleri, derin 3D görüntü ağları oluşturma ve eğitme vb. dahil) ilgilenebilecek gerekli kitaplıkların edinilmesi gerekmektedir.
- Derin sinir ağları modellerini yazmak ve yürütmek için kullanılabilecek ve yapay zeka projelerini gerçekleştirmek için bir platform temsil eden Google Colab ile ilgilenilmesi gerekmektedir.
- 3B veri görüntülerini derin ağa beslemeden önce ilk aşama olarak ön işleme işleminin uygulanılması gerekmektedir.

Bu işlemlerden bazıları şunlardır:

- Girdi veri görüntüleri Nifti formatındadır, bu yüzden bu tiple uğraşılması gerekmektedir.
- Bu görüntülerle başa çıkmak için gerekli paketin kurulması gerekmektedir.
- Verilerin derin ağ içinde kullanılması için HU değerlerinin 0-1 arasında ölçeklendirilmesi gerekmektedir.
- Her bir CT tarama görüntüsünün derinliğini, yüksekliğini ve genişliğinin değiştirilmesi gerekmektedir. Böylece her görüntü için derinlik_yeniden boyutlandırma_faktörü, yükseklik_yeniden boyutlandırma_faktörü ve genişlik_yeniden_boyutlandırma faktörü hesaplanmalıdır.

deep_resize_factor=gerekli derinlik/mevcut derinlik

height_resize_factor= gerekli yükseklik / mevcut yükseklik

width_resize_factor= gerekli genişlik / mevcut genişlik

- Bundan sonra, her boyutun yeni boyutlarını elde etmek için her yeniden boyutlandırma faktörü, yeniden boyutlandırma işlevi için girdi olarak kullanılacaktır.
- Ön işlemedeki diğer bir son adım, her görüntünün 90 derece döndürüldüğü ve böylece oryantasyonun sabitlendiği oryantasyon prosesi kullanılacaktır.
- Artık veriler sonraki adımlar için hazır hale gelmiştir. (eğitim ve doğrulama setleri hazırlanıyor).

3.3.3. Veri koşullandırma

CNN'ler gibi derin sinir ağlarının eğitimi, iyi genelleme yetenekleri elde etmek için çok sayıda örnek gerektirmektedir.

Bu nedenle, gerçek dünya nesneleri için büyük ölçekli 3B veri kümelerinin eksikliği göz önünde bulundurulduğunda, çok sayıda vaka dosyası içeren sentetik veri kümelerinin kullanılması haklıdır. Ancak, bu veri setleri, gerçek dünya nesnelerini tarayansensörlerden gelen sisteme beklenen girdiye benzeyecek şekilde uyarlanmalıdır [27].

Seçtiğimiz veri setimizde, CT tarama görüntüleri, -1024 ile 2000 arasında değişen ev alanı birimlerinde (HU) ham voksel yoğunluğunu depolamaktadır. Sorun şu ki, 400'ün üzerindeki değerler kemik alanlarını temsil etmekte, bu nedenle -1000 ile 400 arasında bir aralığı seçmenin doğru olacağı düşünülmektedir. CT tarama değerlerinin yalnızca istenen ROI'lerinin alınması gerekmektedir (doku ROI'lerini temsil eder).

3.3.4. Veri büyütme

Veri büyütme, dönüştürme, yakınlaştırma, çevirme, döndürme, afin dönüşümler vb. gibi bazı işlemleri kullanarak veri örneklerini artırma işlemidir. Bütçeyi ve veritabanındaki görüntü sayısını artırmak veri kümesi boyutunu artıracaktır.

Artış miktarına artış da denilebilir ve aslında öğrenme miktarının iyileştirilmesine katkıda bulunur. Bu nedenle, veri büyütmenin ana fikri, ağı herhangi bir gürültülü veriye veya veri örneklerindeki diğer deformasyonlara karşı daha sağlam hale getirmek için aynı veri örneklerinin farklı şekil, boyut ve formlarında ağı öğrenmektir.

Bu iyileştirme, ağın performansını artıracak ve sahte eğitim durumunu önleyecektir [28].

3.3.4.1. Veri artışı

Veritabanındaki veriler, yeni veriler üretmek için farklı rastgele açılarla döndürme gibi birden çok işlem uygulanarak artırılacaktır. Rank 3 tensörler verileri depolamak için kullanılır ve aşağıdaki yapıya (örnekler, yükseklik, genişlik, derinlik) sahip olup, veri artışını kullanarak, veriler üzerinde 3B evrişimler gerçekleştirmek için eksen 4'e yeni bir boyut-1 boyutu eklenebilmektedir. Yeni veri yapısı (örnekler, yükseklik, genişlik, derinlik, 1) olacaktır. Orada farklı ön işleme ve güçlendirme teknikleri vardır.

Tren ve doğrulama veri yükleyicisi seçilirken, boyutu rastgele farklı açılarda döndüren eğitim verileri ve artırma işlevi iletilerek eğitim ve doğrulama verileri 0 ile 1 arasında değerlere sahip olacak şekilde zaten yeniden değerlendirilebilmektedir.

3.3.4.2. Veri ayırma

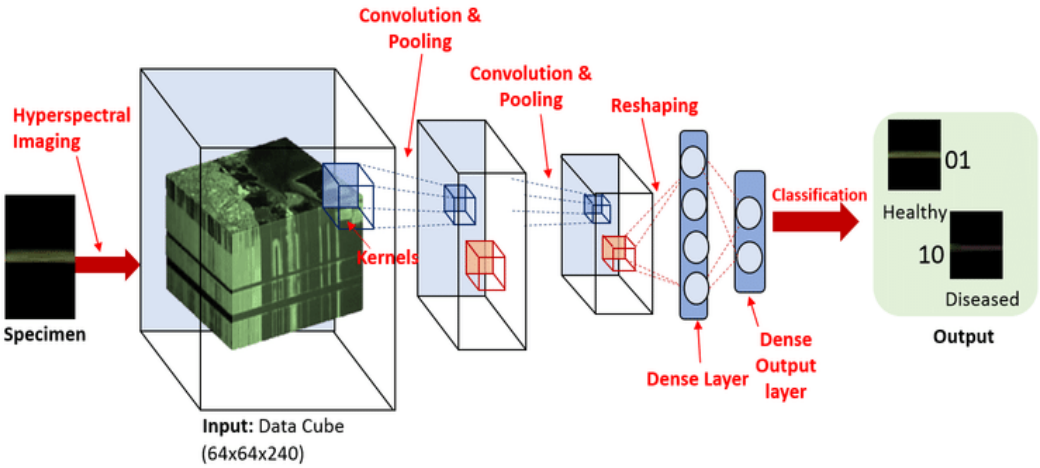
Bu projede, veriler sırasıyla 0.7, 0.2, 0.1 oranlarıyla Eğitim veri seti, Test veri seti ve Doğrulama veri seti olarak ayrılmıştır.

- Eğitim veri setindeki örnek sayısı, normal sınıfta 70 örnek ve anormal sınıfta 70 örnek olmak üzere 140 örnektir.
- Test veri setindeki örnek sayısı, her sınıf için 20 örnek olmak üzere 40 örnektir.
- Doğrulama veri setindeki örnek sayısı sadece 20 örnektir, her sınıf için 10 örnek.

3.4. 3D CNN Modeli

3B Tahmin Evrişimli Sinir Ağı (CNN) Oluşturma Adımları, 2B CNN'ler RGB (3 kanallı) görüntüleri işlemek için yaygın olarak kullanılmaktadır. 3B CNN basitçe 3B eşdeğeridir: 3B hacim girişi veya bir dizi 2B çerçeve (örneğin bilgisayarlı tomografideki dilimler) olarak alınmaktadır. 3B CNN'ler hacimsel veri temsillerini öğrenmek için güçlü bir modeldir.

Şekil 3.5. aşağıdaki bölümlerden oluşan 3D CNN mimarisini içermektedir.



Şekil 3.5. Sınıflandırma için 3B Evrişimli Sinir Ağı Mimarisi

Giriş katmanı:

Bu katman, ağın girişini temsil etmekte ve bizim durumumuzda giriş, veri örneklerini temsil eden 3B görüntülerini içermektedir (3B kutu).

Evrişim katmanları:

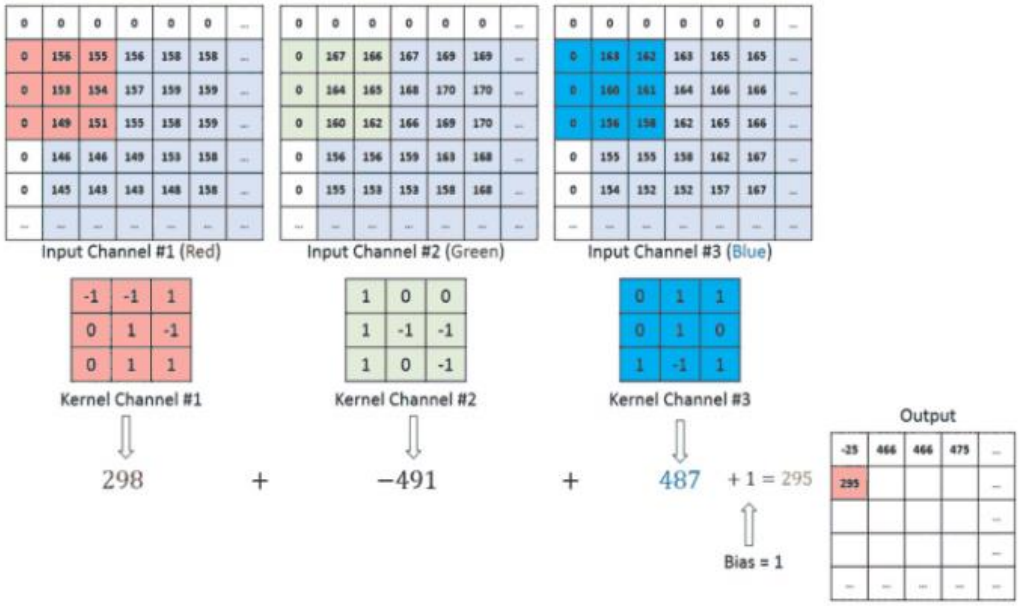
Evrişim işlemi, bir grup filtre (örneğin 64 filtre) kullanılarak görüntü üzerinde evrişim işleminin gerçekleştirildiği CNN'nin temel parçasıdır.

Evrişimde, filtre ve görüntü arasındaki ürünlerin toplamı hesaplanmakta ve her piksel (x,y,z) için tekrarlamaktadır. Evrişim, dolgu seçeneği kullanılarak yapılmakta; bu, görüntünün, yatılı olanlar da dahil olmak üzere tüm piksellere evrişim uygulanmasının hesaba katılması için genişletildiği anlamına gelmektedir. Dolgu, aşağıda yer alan denkleme göre yapılmaktadır.

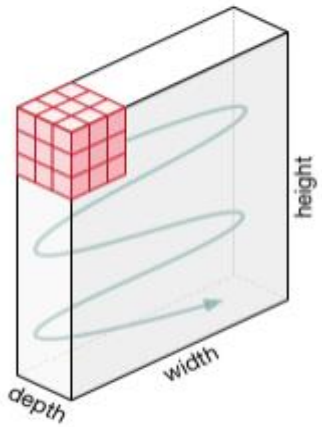
NewImgSize=Floor(k/2)+1

Burada k çekirdek boyutudur. Yani örneğin resim boyutu 128*128*64 ve filtre boyutu 3*3 ise, dolgudan sonraki yeni boyut 130*130*66 olacaktır.

Konvolüsyon kısmı, görüntüden kenarlar, çizgiler, noktalar, eğriler, köşeler vb. dahil olmak üzere üst düzey öznitelikleri çıkarmak için çok önemlidir. Böylece hiyerarşik özellikler, bu küçük özniteliklerin parçalarından başlayarak nihai bileşene ulaşana kadar inşa edilecektir. Şekil 3.6. evrişim sürecini gösterirken, Şekil 3.7 filtrenin görüntü boyunca nasıl hareket ettiğini gösterir [30].



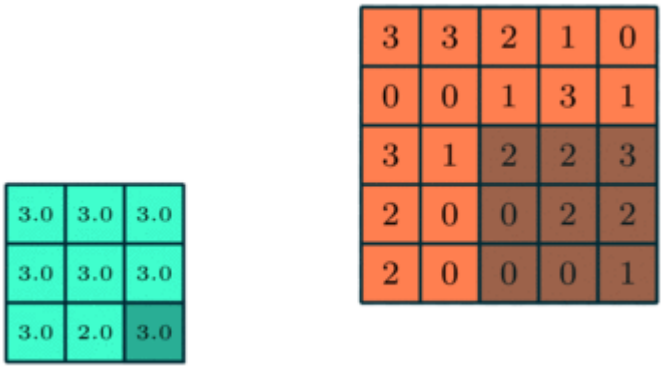
Şekil 3.6. Sınıflandırma için 3B Evrişimli Sinir Ağı Mimarisi



Şekil 3.7. Filtrenin hareketi (Çekirdek)

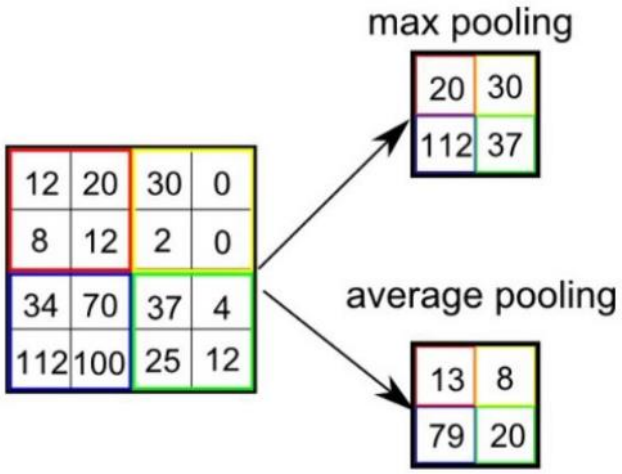
Havuz Katmanları

Giriş görüntüsüne evrişim uygulandıktan sonra, bu sonuç katlanmış veriler havuzlama mekanizması kullanılarak boyut küçültme ile azaltılmaktadır. Şekil 3.8. maksimum havuzlama yöntemini [30] göstermektedir.



Şekil 3.8. Maksimum Havuzlama Yöntemi

Havuzlama katmanında, görüntü tüm boyutlarında 2 küçültülür, böylece görüntü evrişim kısmından sonra 130*130*66 boyutundaysa, havuzlama işleminden sonraki yeni boyut 115*115*33 olacaktır. Diğer bir bakış açısından, yeni görüntü enterpolasyonlu değerler, orijinal görüntünün 4 pikselinin maksimumu veya ortalaması kullanılarak yapılandırılacaktır. Şekil 3.9. maksimum ve ortalama havuzlama arasındaki farkı gösterir.

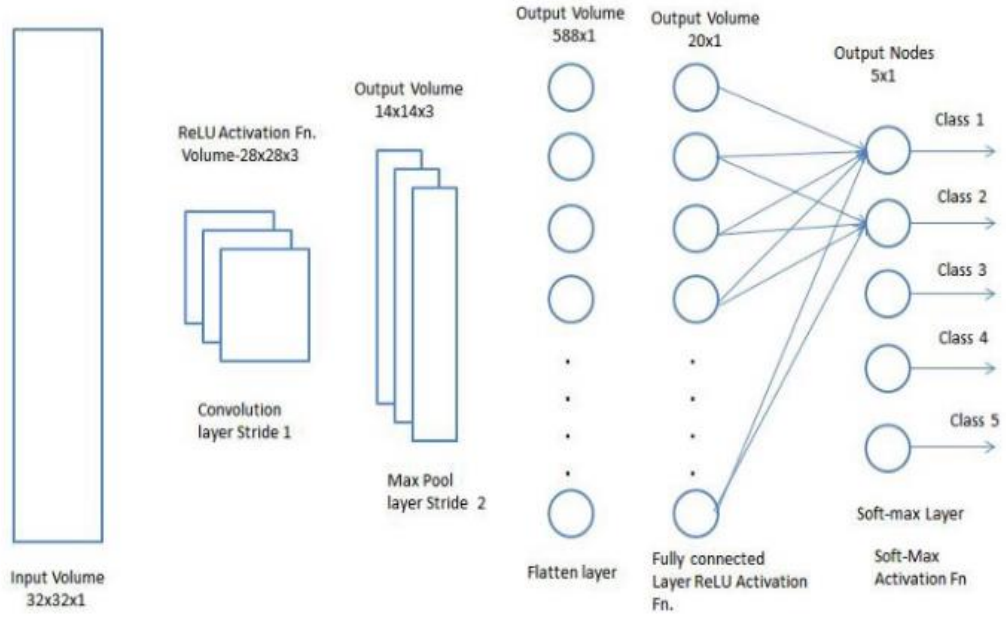


Şekil 3.9. Maksimum ve Ortalama Havuzlama Arasındaki Farkı

Tam Bağlantılı Katman FC sınıflandırması

Sınıflandırma katmanı, evrişim ve havuzlama katmanlarından elde edilen üst düzey özelliklerin doğrusal olmayan kombinasyonlarının öğrenilmesini gerçekleştiren 3D CNN'deki tam bağlantılı katmandır. Bir sınıflandırıcı olarak CNN kullanılması durumunda, FC katmanını, sınıflandırmadan sorumlu olan bir sigmoid aktivasyon

katmanı takip etmektedir (problemdeki her sınıfın bir olasılığını atama, böylece daha yüksek olasılığa sahip sınıf kazanacaktır). Şekil 3.10**.** FC ve sigmoid aktivasyon katmanlarını göstermektedir.



Şekil 3.10. Sonunda FC katmanı ve sigmoid aktivasyonu ile CNN

3.5. 3D CNN Modelindeki Modifikasyonumuz

3D CNN'nin en iyi performansını elde etmek için aşağıdaki mimari modifikasyonlar önerilmektedir.

3.5.1. Toplu normalleştirme katmanı

Bu katman, önceki katmandan alınan özellik değerlerinin bir sonrakine gönderilmeden önce normalleştirilmesi için gereklidir. Bu katmanın işlemi, ağı daha hızlı ve daha kararlı hale getiren katmanların çıktılarını yeniden oluşturmak ve yeniden ölçeklendirmektir. Toplu normalleştirme, katman [31] [32] çıktısını normalleştirmek için denklem aşağıdaki denklem kullanılmıştır.

$Z^n=(Z-M_z)/S_z$

Mz, nöronların çıktısının ortalaması iken Sz, bunun standart sapmasıdır.

3.5.2. Küresel ortalama havuzlama katmanı

Tam bağlantılı katmanlar kullanmak yerine, modelin son tahminini elde etmek için tüm girdi görüntüsünün bir özellik vektörünü elde etmek için özellik haritasında aşağı

örnekleme işlemini uygulayan ortalama havuzlama katmanının kullanılması gerekmektedir. Küresel havuzlama, keras kitaplığı tarafından desteklenmektedir

(bir Python arabirimi sağlar) [33] [34].

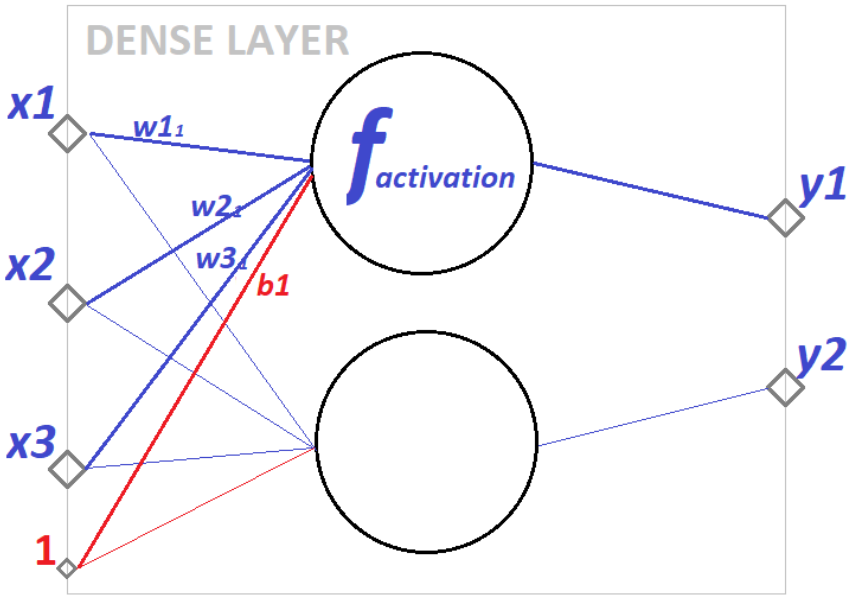
3.5.3. Yoğun katman

Yoğun katmanlar, indirgeme katmanı veya sınıflandırıcı olarak kullanılabilmektedir. Bizim durumumuzda, her iki durumda da kullanılması uygun olacaktır. İlk olarak, bir önceki katmanın çıktısını yoğunlaştırmak için 512 birim boyutunda yoğun bir katman kullanılacak, ikinci yoğun katman ise global normalizasyon katmanı tarafından elde edilen özellik vektörünü sınıflandırmak için kullanılacaktır. Bu katmanın matematiksel formülü aşağıda belirtilmiştir.

$$Y_j = sigmoid\left(\sum_{i=1}^{n} W_{i,j} \times X_i + b_j\right)$$

W_i,j önceki katmanın nöronlarından jth çıktı nöronuna olan ağırlıklar olduğunda, Xi yoğun katmanın girişi veya önceki katmanın çıktısı, bj ise önyargıdır.

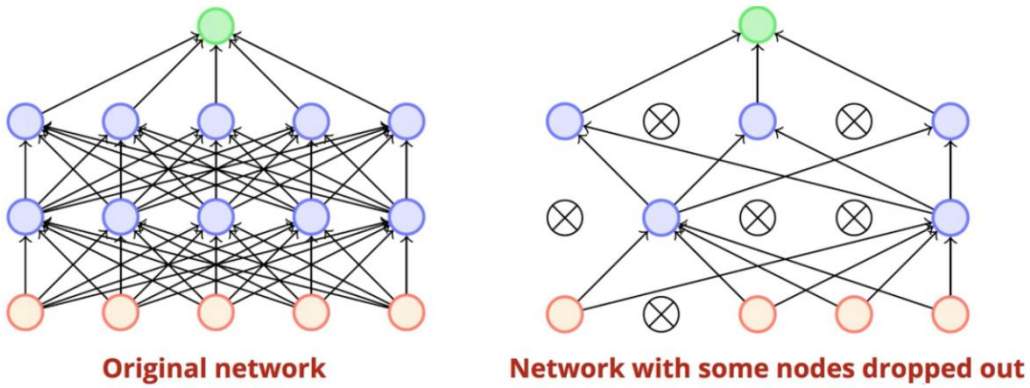
Sigmoid, yoğun katmanın aktivasyon işlevidir. Önceki tüm katmanların "ReLU" aktivasyon fonksiyonu vardır, ancak son sınıflandırma katmanında [0-1] aralığında çıktı veren bir fonksiyona ihtiyaç duyulmaktadır. Bu yüzden sigmoid fonksiyonunun seçilmesi gerekmektedir. Şekil 3.11**.** Yoğun katmanın mimarisini açıklanmaktadır [35].



Şekil 3.11. Yoğun Katman

3.5.4. Bırakma katmanı

Bu katman, (yoğun katmandan elde edilen) bazı öznitelik vektör değerlerini sıfıra ayarlayarak ağın aşırı uyum sorununu önlemek için kullanılır. Bu mekanizma, ağı aşırı uyumdan koruyacak olan son özellik vektörünün bazı değerlerini ortadan kaldıracaktır. Bu katman, yukarıda açıklanan iki yoğun katman arasına yerleştirilecektir. Şekil 3.12. bırakma katmanı topolojisini göstermektedir.

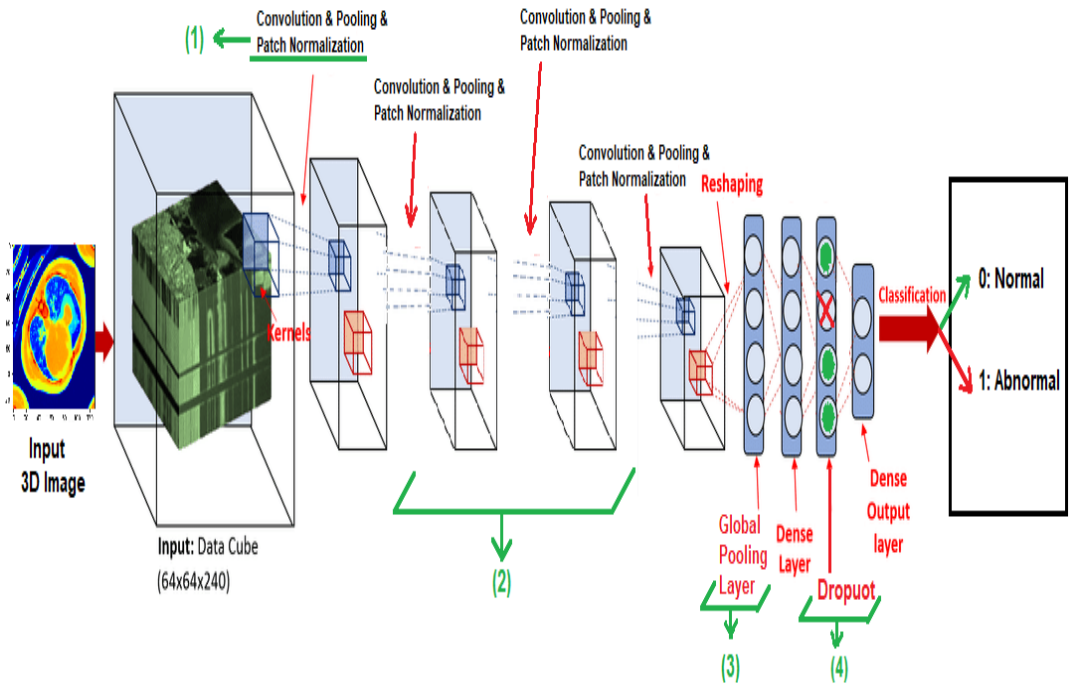


Şekil 3.12. Bırakma Katmanı

Bırakma hiperparametresi, bir katmandaki belirli bir düğümü eğitme olasılığıdır; burada 1.0, bırakma olmadığını gösterirken, 0.0, katmandan hiçbir çıktı olmadığını göstermektedir [37].

Bırakma değeri 0,3'e atanmıştır, bu, fazla uydurmayı önlemek amacıyla giriş birimlerinin %30'unun rastgele bırakıldığı anlamına gelir.

Şimdi, 3B CNN'nin modifikasyonunu gösterdikten sonra, 3B CNN'nin mimarisi Şekil 3.13' te gösterildiği gibi olacaktır.



Şekil 3.13. Önerilen 3D CNN modeli

3D CNN'deki değişiklikleri şu şekilde sıralamak mümkünür;

- Öğrenme sürecini iyileştirmek ve öğrenme süresini azaltmak için evrişimli katmanların özellik haritasını normalleştirmek için yama normalleştirme katmanının eklenmesi.
- Daha üst düzey özellikler elde etmek için modele iki yama (Evrişim + havuzlama + yama normalleştirme) katmanı eklenmesi.
- Öğrenmenin daha hızlı olması için parametre sayısını en aza indirmek için tam bağlı katman FC'yi global havuz katmanıyla değiştirilmesi.
- Aşırı sığmayı önlemek için son iki yoğun katman arasına bırakma katmanının eklenmesi.
- - Fazla uydurmayı önlemek için ikinci, üçüncü ve son yoğun katmanlardan sonra bırakma katmanı ekleyin. İlk üç bırakma katmanı 0.1'e, son katman ise 0.2 oranına ayarlanmıştır.

3.6. Performans Metrikleri

Sınıflandırma sorunlarımızı değerlendirmek için aşağıdaki ölçümleri kullanmanızı öneririz. [38] [39] [40] [41]

Bu problem ne tanıma ne de bölütleme problemidir. Bu problem Sınıflandırma problemidir.

- - Karışıklık Matrisi: Karışıklık Matrisi, doğruluk, Kesinlik, Geri Çağırma ve Skor ölçümlerinin oranını şu esaslara göre hesaplayan istatistiksel bir değerlendirme konseptidir: Gerçek Pozitif, Yanlış Negatif, Gerçek Negatif, Yanlış Pozitif değerler.
- (Gerçek pozitif: TP): gerçek pozitif tahmin değeri; Modelin bir test görüntüsüne dayalı olarak doğru bireyi başarıyla tanımladığı tahminlerinin sayısı, örneğin bir test görüntüsü birey 1 ile ilişkilidir ve model bunu doğru bir şekilde tahmin eder.
- (Yanlış pozitif FP): yanlış pozitif tahmin değeri; Modelin bir test görüntüsüne dayalı olarak doğru bireyi tanımada başarısız olduğu tahminlerin sayısı, örneğin, eğitilmiş bireylerin hiçbiriyle ilişkili olmayan bir test görüntüsü, ancak tahmin edilen model bazı bireylerle ilişkilidir.
- (Gerçek negatif TN): doğru negatif tahmin değeri; Modelin hatalı test görüntülerini belirlemede başarılı olduğu tahminlerin sayısı, eğitilmiş bireylerin hiçbiriyle ilişkili olmayan bir test görüntüsü ve model bunu doğru bir şekilde öngördü.
- (Yanlış negatif FN): bir yanlış negatif tahmin değeri. Modelin yanlış test görüntülerini tanıyamadığı tahminlerin sayısı, örneğin bir test görüntüsü eğitilmiş veri kümesinden bir kişiyle ilişkilendirilir, ancak model bunun herhangi bir kişiyle ilgili olmadığını tahmin eder.
- Gerçek Kabul Oranı (TAR): İlgili tüm test örneklerinden doğru olarak kabul edilen örneklerin oranı.
- Yanlış Kabul Oranı (FAR): Tüm alakasız test örneklerinden yanlış kabul edilen örneklerin yüzdesi.
- Gerçek reddetme oranı (TRR): Tüm alakasız test örneklerinden doğru olarak reddedilen örneklerin oranı.

- Yanlış ret oranı (FRR): İlgili tüm test örneklerinden hatalı olarak reddedilen örneklerin oranı.

Tanımlama sürecinin doğruluğunu ifade eden sınıflandırma doğruluğu.

Doğruluk (ACC) TP, TN, FP ve FN kullanılarak aşağıdaki gibi hesaplanabilir:

ACC = (TP + TN) / (TP + TN + FP + FN) %

- ROC ve AUC Eğrileri: ROC'nin AUC'si hesaplanabilir, AUC işlevi hem test setinden gerçek sonuçları (0,1) hem de birinci sınıfın beklenen olasılıklarını alır, ROC bir sinyal grafiğidir (gerçek pozitif oranı) gürültüye karşı (yanlış pozitif oran). Model performansını belirlemek için AUC olarak kısaltılan ROC altındaki alanı kullanabiliriz. Mümkün olan en iyi AUC 1'dir, en kötüsü ise 0,5'tir (rastgele çizgi 45 derece).
- Kesinlik: Kesinlik, doğru tahmin edilen pozitif gözlemlerin toplam tahmin edilen pozitif gözlemlere oranıdır. Bu metrik cevabın hayatta kaldı olarak etiketlenen tüm yolcular için saptanmıştır. Yüksek hassasiyet, düşük yanlış pozitif oranı ile ilgilidir.
- Kesinlik = TP/TP+FP
- Geri Çağırma: Doğru tahmin edilen pozitif gözlemlerin gerçek sınıftaki tüm gözlemlere oranıdır: Gerçekten hayatta kalan tüm yolculardan kaç tanesini etiketledik? Sorusuna cevap aramaktadır.

Geri Çağırma= TP/TP+FN

- F1 Puanı: Kesinlik ve Geri Çağırmanın ağırlıklı ortalamasıdır. Bu nedenle, bu puan hem yanlış pozitifleri hem de yanlış negatifleri hesaba katar. Sezgisel olarak, doğruluk kadar anlaşılması kolay değildir, ancak F1 genellikle, özellikle eşit olmayan bir sınıf dağılımınız varsa, doğruluktan daha kullanışlıdır. Doğruluk, yanlış pozitifler ve yanlış negatifler benzer maliyete sahipse en iyi sonucu verir. Yanlış pozitiflerin ve yanlış negatiflerin maliyeti çok farklıysa, hem Kesinlik hem de Geri Çağırmaya bakmak daha iyidir.

4. DENEYSEL SONUÇLAR

Bu bölümde, modelin uygulama kısmı tanımlanmıştır. Bunun yanında bazı deneysel senaryolar uygulanmış ve sonuçları listelenerek detaylı olarak tartışılmaktadır.

4.1. Uygulama

Önceki bölümde gösterilen 3D CNN'nin önerilen tasarımının şimdi uygulanması gerekmektedir. Tüm uygulamalar için Python programlama dili ve Google Colab kullanılmaktadır. Amacımız, 3 boyutlu akciğer görüntüsünü girdi olarak alan ve ardından onu normal dokular veya anormal olanlar olarak sınıflandıran bir model oluşturmaktır. Toplam görüntü sayısı 12800 dilim olacak şekilde 100 normal 3D görüntü ve 100 anormal 3D görüntü (her biri 64 görüntü içerir) kullanılmıştır.

Sınıf dizinlerinden taranan verileri okuyun ve her birine numara atayın (normal için 0 ve anormal için 1 etiketleri).

HU değerlerinin normalleştirilmiş işlenmiş görüntülerini [0-1] aralığına getirmek için taramalara ön işleme adımını uygulayın.

Derin öğrenme modelini oluşturun.

4.1.1. Ön İşleme adımının uygulanması

4.1.1.1. Adım 1: işlemler ve 3b görevler için gerekli kitaplıkların içeri aktarılması

Aşağıdaki kitaplıklar yüklenecek ve içe aktarılacaktır:

Açıklama:

Imageio: Animasyonlu görüntüler, video, hacimsel veriler dahil olmak üzere çok çeşitli görüntü verilerini okumak ve yazmak için kullanılan kitaplık.

IPython kitaplığının görüntüleme seçeneği:

Görüntüleme seçenekleri için kullanılan kitaplık ipython kitaplığının işlev gösterimi.

URL'lerin (çoğunlukla HTTP) açılmasına yardımcı olmak için urllib kitaplığının istek işlevi kullanılacaktır.

4.1.1.2. Adım 2: gerekli araçların yüklenmesi

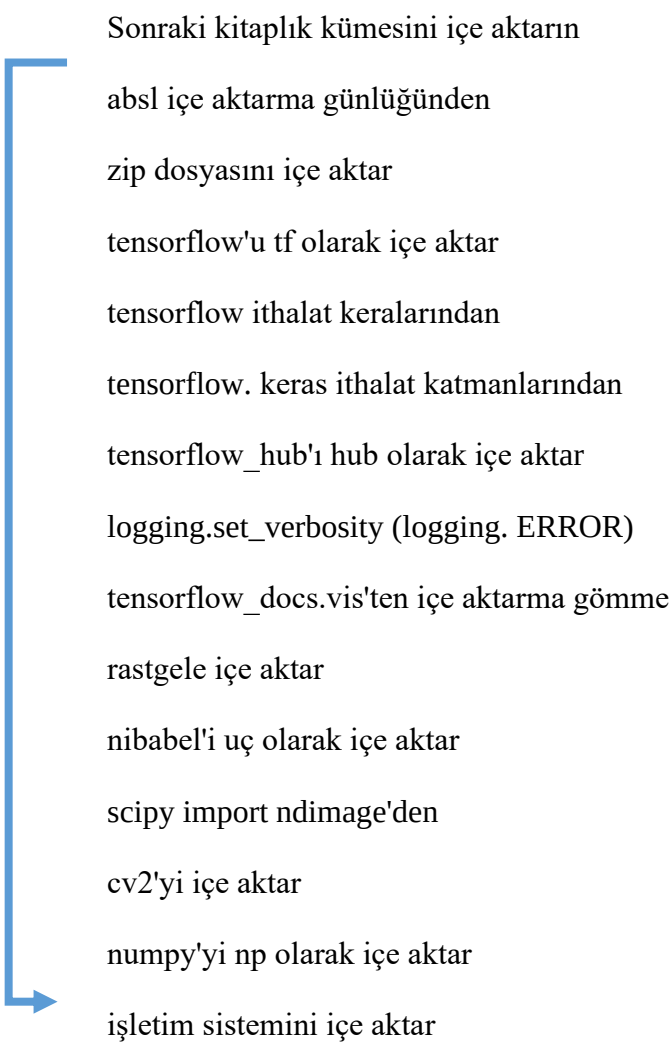
!pip install -q git+https://github.com/tensorflow/docs

Açıklama

OpenCV kütüphanesi, görüntü işleme işlemleri için bir kütüphanedir.

Tensör akışı:

Makine öğrenimi işlemleri için açık kaynaklı bir platform. Bu platform, bir sonraki adımda içe aktarılacak ve kullanılacak birçok kitaplık içerir.



Açıklama:

Mesajları (hata, uyarı, hata ayıklama, bilgi, önemli) görüntülemek için kullanılan Abseil kitaplığının günlük işlevi.

- Sıkıştırılmış dosyalarla uğraşmak için Zipfile kitaplığı.
- Tf, tensorflow kitaplığıdır.
- Tensorflow içe aktarılan kitaplıklar:
- Keras: Hızlı derin öğrenme deneyi için tasarlanmış bir tensorflow arayüz kitaplığı.
- Keras katmanları, ağ katmanlarını oluşturmak ve bunlarla uğraşmak için kullanılır.
- Katmanlar, Keras'taki sinir ağlarının temel yapı taşlarıdır. Bir katman, bir tensör içeri tensör dışarı hesaplama işlevinden (katmanın çağrı yöntemi) ve TensorFlow değişkenlerinde (katmanın ağırlıkları) tutulan bazı durumlardan oluşmaktadır.
- tensorflow_hub, ilgili makine öğrenimi modelleri kümesidir.

Rastgele sayı üretmek için rastgele kitaplık kullanılır. nibabel library PyNIfTI'nin halefidir. Bu kitaplık, bazı yaygın beyin görüntüleme dosya formatlarında (Okuma, yazma) işlemleri için kullanılır. ndimage scipy paketi, bir dizi görüntü işleme işlemi sağlar. Çok boyutlu diziler için kullanılabilirler. Dizi işlemleri için Numpy kitaplığı kullanılır. OS kitaplığı, birleştirme yolları gibi işletim sistemi işlevleri için kullanılır.

4.1.1.3. Adım 3: normal veri kümesi örneklerini indirme (ct-0.zip)

- PASS = "https://github.com/hasibzunair/3D-image-classification-tutorial/ releases/ download/v0.2/CT-0.zip"
- f_NAME = os.path.join(os.getcwd(), "CT-0.zip")
- keras.utils.get_file(f_NAME, PASS)
- with zipfile.ZipFile("CT-0.zip", "r") as z_fp:
- z_fp.extractall("./MosMedData/")

Kodun açıklaması: Geçiş, veri kümesi bağlantısını tutan bir değişkendir.

- .path.join işlevi, geçerli çalışma dizinine (os.getcwd) CT-0.zip (veri kümesinin adı).

Bu yol adının tamamı f_name değişkeninde saklanacaktır.

Dosyaları (keras.utils.get_file) işlevini kullanarak alma, sıkıştırılmış CT-0.zip dosyasını ("r" okuma modunu kullanarak) indirecek ve z_fp dosyasında saklayacaktır. z_fp.extractall("./MosMedData/") ifadesi, sıkıştırılmış dosyanın görüntü dosyalarını çıkaracak ve bunları MosMedData'da saklayacaktır.

Result of execution:

```
Downloading data from https://github.com/hasibzunair/3D-image-classification-tutorial/releases/download/v0.2/CT-0.zip
1065476096/1065471431 [==============================] - 21s 0us/step
1065484288/1065471431 [==============================] - 21s 0us/step
```

4.1.1.4. Adım 4: normal örnekler üzerinden ön işleme işlemleri için gerekli işlevleri oluşturma

Adım 4.1: read_n_f_işlevi

Nifti 3D görüntü sürecinin okunması gerekmektedir.

1) Nib.load fonksiyonu 3D görüntünün F_path'ını okumak için kullanılır, sonuç "tarama" değişkeninde saklanması gerekmektedir.
2) scan.get.fdata() fonksiyonunu kullanarak ham veriyi alın ve sonucu scan değişkeninde saklanması gerekmektedir.
3) Taranan güzel 3D görüntüyü temsil eden taramayı döndürmek gerekmektedir.

Kod:

```
- def read_n_f(F_PATH):
- """Read and load C_heighteighteightume"""
- # Read file
- scan = nib.load(F_PATH)
- # Get raw data
- scan = scan.get_fdata()
- return scan
```

Adım 4.2: resize_C_heigh function

Bu prosedür, iki işlemi içeren yeniden boyutlandırma sürecini temsil etmektedir:

1) Yönlendirmenin sabitlenmesi için her görüntünün yönlendirmesinin 90 derece döndürüldüğü yönlendirme işlemi.
2) Her bir CT tarama görüntüsünün derinliğini, yüksekliğini ve genişliğini değiştirin, böylece her görüntü için derinlik_boyut_boyut_ faktörü, yükseklik_yeniden boyut_faktörü ve genişlik_yeniden boyut_faktörünü aşağıdaki gibi hesaplanması gerekmektedir;

- deep_resize_factor=gerekli derinlik/mevcut derinlik
- height_resize_factor= gerekli yükseklik / mevcut yükseklik
- width_resize_factor= gerekli genişlik / mevcut genişlik

Döndürdükten ve yeniden boyutlandırdıktan sonra 3D görüntüyü temsil eden fotoğrafların döndürülmesi gerekmektedir.

Yeniden boyutlandırma işlemi ndimage kütüphanesinin "zoom" fonksiyonu kullanılarak, döndürme işlemi ise "döndür" fonksiyonu kullanılarak yapılacaktır.

Kod:

```
def resize_C_heig(PHOTOS):
    """Resize across z-axis"""
    # Set the desired depth
    D_depth = 64
    D_width = 128
    D_height = 128
    # Get current depth
    C_depth = PHOTOS.shape[-1]
    C_width = PHOTOS.shape[0]
    C_height = PHOTOS.shape[1]
    # Compute depth factor
    depth = C_depth / D_depth
    width = C_width / D_width
    height = C_height / D_height
    depth_f = 1 / depth
    width_f = 1 / width
    height_f = 1 / height
    # Rotate
    PHOTOS = ndimage.rotate(PHOTOS, 90, reshape=False)
    # Resize across z-axis
    PHOTOS = ndimage.zoom(PHOTOS, (width_f, height_f, depth_f), order=1)
    return PHOTOS
```

Adım 4.3: norm function

Normalleştirme işlevi

Seçilen veri setinde CT tarama görüntüleri, -1024 ile 2000 arasında değişen ev alanı birimlerinde (HU) ham voksel yoğunluğunu depolamaktadır. Sorun şu ki, 400'ün üzerindeki değerler kemik alanlarını temsil etmekte, bu nedenle -1000 ile 400 arasında

bir aralığın seçilmesi gerekmiştir. CT tarama değerlerinin yalnızca istenen ROI'lerini alınması gerekmektedir (doku ROI'lerini temsil eder).

Normalleştirme adımları

1) Min ve max değerlerin tanımlanması (-1000, 400).
2) Her değeri 400'den 400'e çevrilmesi.
3) Her değeri -1000'den -1000'e çevrilmesi.
4) Verilerin derin ağ içinde kullanılabilmesi için HU değerlerini şimdi min-max yaklaşımını kullanarak 0 ile 1 arasında ölçeklendirilmesi.
5) Yeni 3D görüntüyü döndürülmesi.

Kod:

```
- def norm (C_heig):
- """norm the C_heig"""
- min = -1000
- max = 400
- C_heig[C_heig < min] = min
- C_heig[C_heig > max] = max
- C_heig = (C_heig - min) / (max - min)
- C_heig = C_heig.astype("float32")
- return C_heig
```

Bu prosedür, tüm 3D görüntü verilerini okumak ve önceden işlemek için ana prosedürdür.

1) Şık 3D görüntünün okunması ve C_heig değişkeninde saklanması.
2) Resmin normalleştirilmesi.
3) Resmin yeniden boyutlandırılması.
4) Normalleştirilmiş - yeniden boyutlandırılmış - döndürülmüş görüntünün döndürülmesi.

Kod:

```
- def p_scan(path):
- """Read and resize C_heig"""
- # Read scan
- C_heig = read_n_f(path)
```

- # norm
- C_heig = norm(C_heig)
- # Resize width, height and depth
- C_heig = resize_C_heig(C_heig)
- return C_heig

4.1.1.5. Adım 5: normal örnekler klasörünün hazırlanması

"MosMedData/CT-0" normal örnekler klasörünün hazırlanması gerekmektedir. "CT-0" klasörü, normal akciğer dokusuna sahip BT taramalarından oluşmaktadır. VAL_X değişkeni, doğrulama setindeki her bir 3D nifti görüntüsünün adını ve yolunu tutacaktır, bu nedenle normal_S_p tüm normal görüntülerin adlarını içerecektir.

Print ifadesi, normal akciğer 3D görüntülerinin sayısını yazdırmak için kullanılmaktadır.

Kod:

- normal_s_p = [
- os.path.join(os.getcwd(), "MosMedData/CT-0", VAL_X)
- for VAL_X in os.listdir("MosMedData/CT-0")]
- print("CT scans with normal lung tissue: " + str(len(normal_s_p)))

Yürütme sonucu:

```
CT scans with normal lung tissue: 100
```

n_scans, tüm normal görüntüleri diziler olarak içeren bir diziyi temsil etmektedir. (tüm 100 normal görüntünün verileri).

n_labels, karşılık gelen etiketlerini içermektedir. [0-100]

Kod:

- n_scans = np.array([p_scan(path) for path in normal_s_p])
- n_labels = np.array([0 for _ in range(len(n_scans))])

4.1.1.6. Adım 6: normal veri kümesi örneklerinin indirilmesi (ct-23.zip)

- PASS11 = "https://github.com/hasibzunair/3D-image-classification-tutorial/releases/download/v0.2/CT-23.zip"
- f_NAME1 = os.path.join(os.getcwd(), "CT-23.zip")
- keras.utils.get_file(f_NAME1, PASS11)

- with zipfile.ZipFile("CT-23.zip", "r") as z_fp:
- z_fp.extractall("./MosMedData/")-

Adım 3 ile aynı adımlar ancak burada anormal örnekler indirilmektedir, sıkıştırılmış dosyanın bağlantısı PASS11 değişkeninde bulunmaktadır.

Yürütme Sonucu:

```
Downloading data from https://github.com/hasibzunair/3D-image-classification-tutorial/releases/download/v0.2/CT-23.zip
1045168128/1045162547 [==============================] - 13s 0us/step
1045176320/1045162547 [==============================] - 13s 0us/step
```

4.1.1.7. Adım 7: anormal numuneler üzerinde ön işleme işlemleri için gerekli işlevlerin oluşturulması

Adım 4-1, adım 4-2 ve adım 4-3'teki adımların aynısı (okuma, yeniden boyutlandırma ve norm işlevleri).

4.1.1.8. Adım 8: Anormal Örnekler Klasörünün Hazırlanması

abnormal_s_p = [

- os.path.join(os.getcwd(), "MosMedData/CT-23", VAL_X)
- for VAL_X in os.listdir("MosMedData/CT-23")
-]

print("CT scans with abnormal lung tissue: " + str(len(abnormal_s_p)))

```
CT scans with normal lung tissue: 100
```

Anormal örnekler klasörünün hazırlanması gerekmektedir "MosMedData/CT-23"

"CT-23" klasörü, anormal akciğer dokusuna sahip BT taramalarından oluşmaktadır.

VAL_X değişkeni, doğrulama kümesindeki her bir 3D nifti görüntüsünün adını ve yolunu tutacaktır, bu nedenle anormal_s_p, tüm anormal görüntülerin adlarını içerecektir.

Print ifadesi, anormal akciğer 3D görüntülerinin sayısını yazdırmak için kullanılmaktadır.

4.1.1.9. Adım 9: eğitim ve doğrulama setlerinin oluşturulması

Adım 9.1: Eğitim ve Doğrulama Setlerini Oluşturma

- n_scans, tüm normal görüntüleri diziler olarak içeren bir diziyi temsil etmektedir (tüm 100 normal görüntünün verileri).
- n_labels, karşılık gelen etiketlerini içerir [0]
- abn_scans diziler olarak tüm anormal görüntüleri içeren bir diziyi temsil etmektedir. (tüm 100 anormal görüntünün verileri)
- abn_labels ilgili etiketlerini içermektedir. [1]

Kod:

- n_scans = np.array([p_scan(path) for path in normal_s_p])
- n_labels = np.array([0 for _ in range(len(n_scans))])
- abn_scans = np.array([p_scan(path) for path in abnormal_s_p])
- abn_labels = np.array([1 for _ in range(len(abn_scans))])

Adım 9.2: Veri Kümesini Eğitim İçin %70, test için %20 ve normal ve anormal görüntüler için doğrulama için %10'a bölme.

Kod:

- x_train = np.concatenate((a_scans[:70], n_scans[:70]), axis=0)
- y_train = np.concatenate((a_labels[:70], n_labels[:70]), axis=0)
- x_val = np.concatenate((a_scans[:70], n_scans[70:]), axis=0)
- y_val = np.concatenate((a_labels[70:], n_labels[70:]), axis=0)
- print(
- "Number of samples in train is: %d. Number of samples in Validation is %d."
- % (x_train.shape[0], x_val.shape[0])

```
Number of samples in train and validation are 160 and 40.
```

Burada eğitim için 160 örnek ve doğrulama için 40 örnek alınmıştır, ancak her örnek 64 dilimden oluşmultur. Toplam eğitim görüntüleri: 160*64=10240 görüntü, doğrulama görüntüleri ise 2560 görüntü olarak elde edilmiştir.

4.1.2. Veri büyütme uygulaması

Bu adımda, eğitimin daha etkili olması için görüntü sayısını artırmak için kullanılacak veri büyütme fonksiyonunun uygulanması gerekmektedir.

Kod:

- data_augmentation = tf.keras.Sequential([
- layers.experimental.preprocessing.RandomFlip("horizontal_and_vertical"),
- layers.experimental.preprocessing.RandomRotation(0.2),
- layers.experimental.preprocessing.RandomContrast([0.5, 1.0]),
- #layers.experimental.preprocessing.RandomZoom(.2, .2)
-])

Keras kütüphanesi, veri büyütme sürecini [42] oluşturmak için bize çok verimli bir yol sağlamaktadır. Döndürme, kontrast iyileştirme, yakınlaştırma, çevirme vb. gibi farklı işlem türlerini içeren "ön işleme" paketini kullanıldığı bu model treni aynı görüntünün farklı şekilleri üzerinde gerçekleşmektedir.

Veri büyütmenin uygulanması:

1) Eğitim ve doğrulama verilerinin hazırlanması:

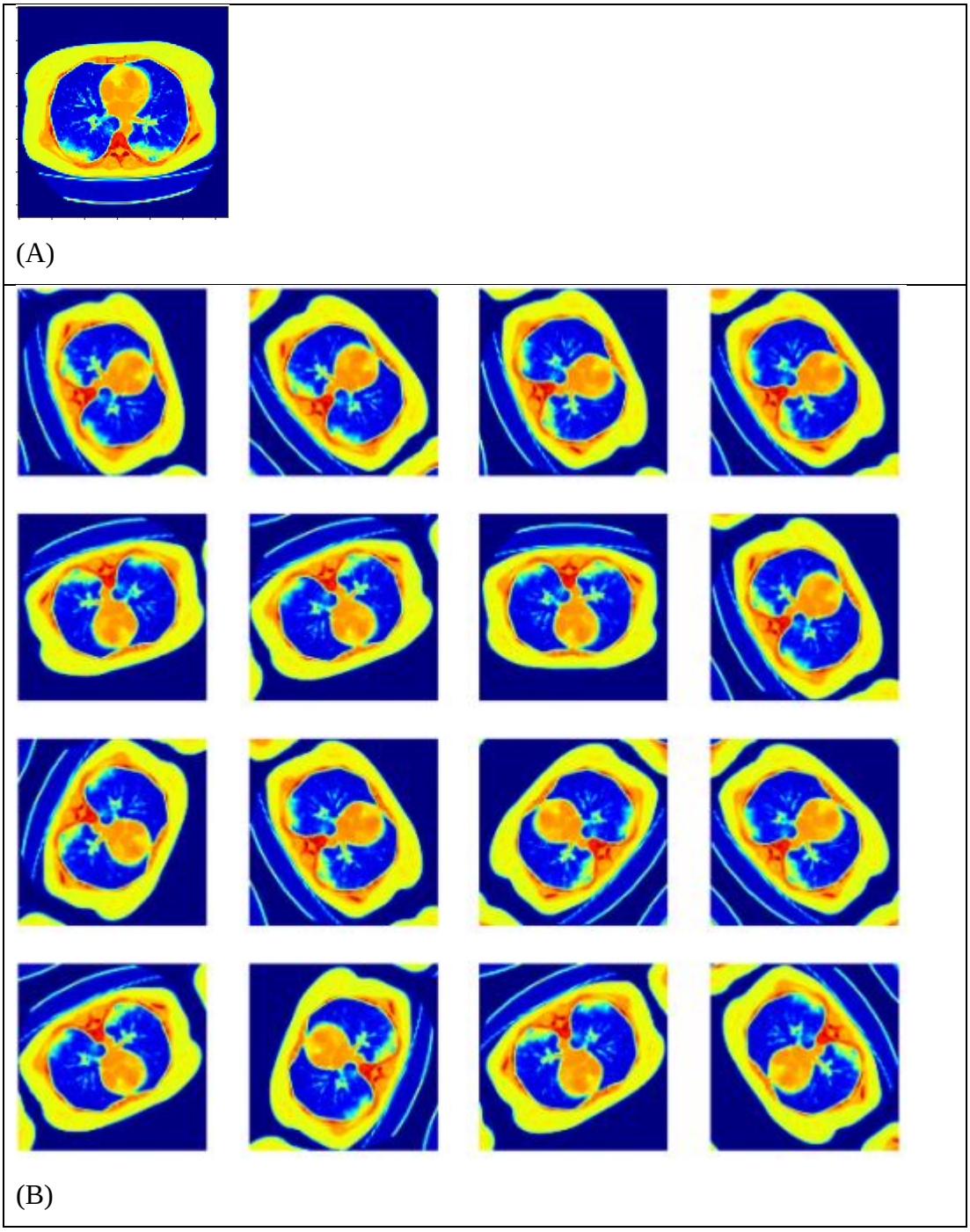
```
# Prepare dataset (before augmentation)
train_loader = tf.data.Dataset.from_tensor_slices((x_t, y_t))
validation_loader = tf.data.Dataset.from_tensor_slices((x_v, y
_v))
```

2) Veri büyütme işlevinin uygulanması:

```
batch_size = 4
AUTOTUNE = tf.data.AUTOTUNE

def prepare(ds, shuffle=False, augment=False):
  # Resize and rescale all datasets
 # ds = ds.map(lambda x, y: (resize_and_rescale(x), y),
  #            num_parallel_calls=AUTOTUNE)

  if shuffle:
    ds = ds.shuffle(100)
  # Batch all datasets
  ds = ds.batch(batch_size)

  # Use data augmentation only on the training set
  if augment:
    ds = ds.map(lambda x, y: (data_augmentation(x, training=True
), y),
                num_parallel_calls=AUTOTUNE)
  # Use buffered prefecting on all datasets
  return ds.prefetch(buffer_size=2)
```

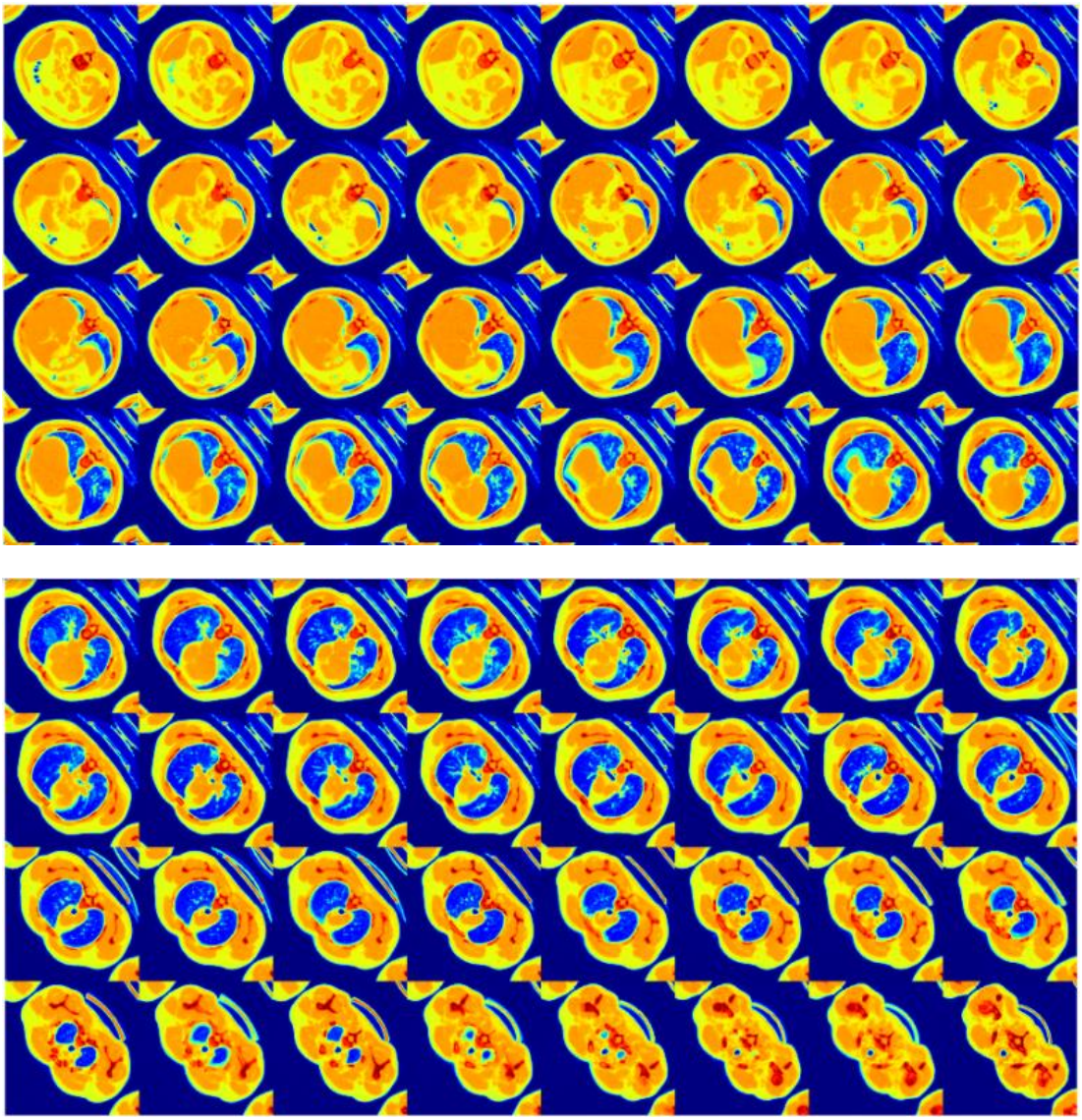
Şekil 4.1. Veri kümesinin bir örneğine veri artırma uygulama örneğini göstermektedir.

Süreci kontrol etmek için bazı büyütme sonuçlarının görüntülenmesi gerekmektedir.



Şekil 4.2. Veri Kümesinin Bir Örneği Üzerindeki Veri Büyütme Etkisi: A) Orijinal Görüntü, B) Aynı Görüntünün 16 Farklı Veri Büyütmesi

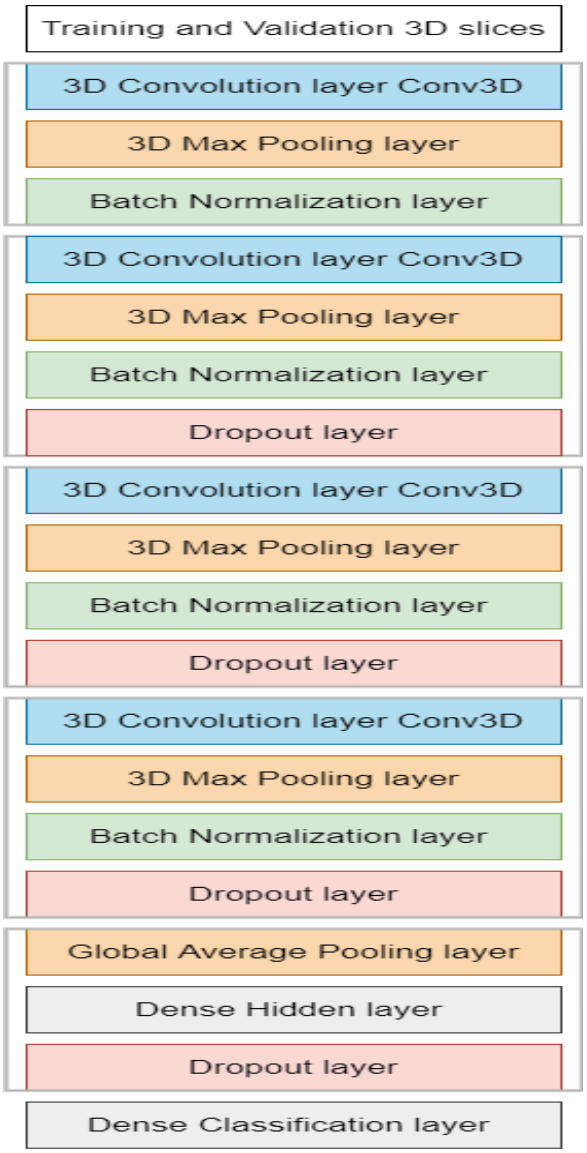
Şekil 4.2. bir 3D görüntü örneğini ve veri büyütme uygulandıktan sonraki 64 dilimini içermektedir.



Şekil 4.3. 3D Görüntü Örneği ve 64 Dilimi

4.1.3. 3D CNN modifiye modelinin oluşturulması

Modelin oluşturulması için tensorflow [43] tarafından sağlanan keras katmanlarını kullanılmaktadır. Şekil **4.3.** modelin mimarisini göstermektedir.



Şekil 4.4. Önerilen 3D CNN'nin Mimarisi

- from keras.models import Sequential
- model = keras.models.Sequential()
- from tensorflow.keras.layers import Conv3D
- from keras.layers import Activation, Dense
- def get_model(width=128, height=128, depth=64):
- """Build a 3D conC_heightutional neural network model."""

Ağın giriş şeklinin tanımlanması:

- inputs = keras.Input((width, height, depth, 1))

16 filtre ve ReLU etkinleştirme işleviyle evrişim katmanları, havuzlama katmanları ve toplu normalleştirme katmanlarının ilk kombinasyonunun tanımlanması:

- VAL_X = layers.Conv3D(filters=16, kernel_size=3, activation="relu") (inputs)
- PARA = layers.MaxPool3D(pool_size=2)(PARA)
- PARA = layers.BatchNormalization()(PARA)

32 filtreli evrişim katmanları, havuzlama katmanları, yığın normalleştirme katmanları ve 0.1 ile ReLU aktivasyon fonksiyonu ve bırakma katmanının ikinci kombinasyonunu tanımlanması;

- PARA = layers.Conv3D(filters=32, kernel_size=3, activation="relu")(PARA)
- PARA = layers.MaxPool3D(pool_size=2)(PARA)
- PARA = layers.BatchNormalization()(PARA)
- PARA = layers.Dropout(0.1)(PARA)

64 filtreli evrişim katmanları, havuzlama katmanları ve toplu normalleştirme katmanlarının üçüncü kombinasyonunu ve 0.1 ile ReLU aktivasyon fonksiyonu ve bırakma katmanı tanımlanması;

- PARA = layers.Conv3D (filters=64, kernel_size=3, activation="relu") (PARA)
- PARA = layers.MaxPool3D(pool_size=2)(PARA)
- PARA = layers.BatchNormalization()(PARA)
- PARA = layers.Dropout(0.1)(PARA)

64 filtreli evrişim katmanları, havuzlama katmanları ve toplu normalleştirme katmanlarının dördüncü kombinasyonunu ve 0.1 ile ReLU aktivasyon fonksiyonu ve bırakma katmanı tanımlanması;

- PARA = layers.Conv3D(filters=64, kernel_size=3, activation="relu") (PARA)
- PARA = layers.MaxPool3D(pool_size=2)(PARA)
- PARA = layers.BatchNormalization()(PARA)
- PARA = layers.Dropout(0.1)(PARA)

Küresel ortalama havuzlama katmanını, ReLU aktivasyon fonksiyonu ile yoğun 512 birim katmanlarının tanımlanması;

- PARA = layers.GlobalAveragePooling3D()(PARA)
- PARA = layers.Dense(units=512, activation="relu")(PARA)

1 birim ve sigmoid aktivasyon fonksiyonu ile bırakma katmanını tanımlanması ve yoğun katmanı sınıflandırılması;

- PARA = layers.Dropout(0.2)(PARA)
- outputs = layers.Dense(units=1, activation="sigmoid")(PARA)

Modelin 3D CNN adıyla tanımlanması ve giriş boyutuyla oluşturulması (128,128,64)

- # Modelin Tanımlanması:
- model = keras.Model(inputs, outputs, name="3dcnn")
- return model
- # Modelin İnşa Edilmesi:
- model = get_model(width=128, height=128, depth=64)
- model.summary ()

4.1.4. 3D CNN modelinin özet ve sonuçları

Aşağıdaki sonuçlar model özetini ve eğitilen parametre sayısını göstermektedir.

```
Model: "3dcnn"
 Layer (type)                Output Shape              Param #
=================================================================
 input_7 (InputLayer)        [(None, 128, 128, 64, 1)  0
                             ]
conv3d_24 (Conv3D)          (None, 126, 126, 62, 16)  448

 max_pooling3d_24 (MaxPoolin  (None, 63, 63, 31, 16)   0
 g3D)

 batch_normalization_24 (Bat  (None, 63, 63, 31, 16)   64
 chNormalization)

 conv3d_25 (Conv3D)          (None, 61, 61, 29, 32)    13856

 max_pooling3d_25 (MaxPoolin  (None, 30, 30, 14, 32)   0
 g3D)

 batch_normalization_25 (Bat  (None, 30, 30, 14, 32)   128
 chNormalization)

 dropout_20 (Dropout)        (None, 30, 30, 14, 32)    0

 conv3d_26 (Conv3D)          (None, 28, 28, 12, 64)    55360

 max_pooling3d_26 (MaxPoolin  (None, 14, 14, 6, 64)    0
 g3D)

 batch_normalization_26 (Bat  (None, 14, 14, 6, 64)    256
 chNormalization)

 dropout_21 (Dropout)        (None, 14, 14, 6, 64)     0

 conv3d_27 (Conv3D)          (None, 12, 12, 4, 64)     110656

 max_pooling3d_27 (MaxPoolin  (None, 6, 6, 2, 64)      0
 g3D)

 batch_normalization_27 (Bat  (None, 6, 6, 2, 64)      256
 chNormalization)

 dropout_22 (Dropout)        (None, 6, 6, 2, 64)       0

 global_average_pooling3d_6   (None, 64)               0
 (GlobalAveragePooling3D)

 dense_12 (Dense)            (None, 512)               33280

 dropout_23 (Dropout)        (None, 512)               0

 dense_13 (Dense)            (None, 1)                 513

=================================================================
Total params: 214,817
Trainable params: 214,465
Non-trainable params: 352
_________________________________________________________________
```

Şekil 4.5. 3D CNN model özetini ve eğitilen parametre sayısı

4.2. Eğitim Modeli

4.2.1. Eğitim parametrelerinin seçilmesi

Öğrenme hızı: Ağın sahte bir eğitime girmesi için değeri yüksek olmayacak, öğrenmenin çok yavaş olması için ne küçük ne de küçük olacak şekilde seçilmektedir. Derin öğrenme ağının ilk yinelemeden sonra öğrenme oranını değiştirilmesi gerekmektedir. Yani tanımlanan öğrenme oranı, ilk öğrenme oranıdır.

```
- initial_learning_rate = 0.1
- lr_schedule = keras.optimizers.schedules.Exponential
  Decay(
- initial_learning_rate, decay_steps=100000, decay_rat
  e=0.96, staircase=True
- )
```

4.2.2. Derleme parametrelerinin tanımlanması

Derleme işlemi, kayıp işlevi, optimize edici öğrenme oranı yöntemi ve performans ölçütleri olmak üzere üç bölümden oluşmaktadır. Optimize edici, öğrenme oranının ağın girişine göre değiştirildiği eğitim sürecinin sürücüsü olarak görev yapan bir sınıftır. Seçilen optimize edici, ikinci türevlerin (gerçek çıktı ile tahmin edilen arasındaki fark) uyarlanabilir tahminini kullanan gradyan iniş algoritmasını kullanan bir optimizasyon algoritmasıdır.

Kayıp fonksiyonu, gerçek sınıf ile tahmin edilen sınıf arasındaki farkı hesaplamak için kullanılmaktadır. Birçok kayıp fonksiyonu vardır ve model için seçilen kayıp fonksiyonu ikili çapraz entropidir.

```
- model.compile(
- loss="binary_crossentropy",
- optimizer=keras.optimizers.SGD(learning_rate=lr_sche
  dule),
- Momentum=[0.9 ],
- )
```

4.2.3. Geri aramaların tanımlanması

Bu adımda eğitim parametrelerinin içinde tutulacağı dosyayı tanımlanmaktadır. Ayrıca, mağaza sürecinin yalnızca en iyi sonuçları koruyacağı ve diğerlerini bırakacağı ifade edilmektedir.

Bu adımda, belirli bir durumda ağın bu koşul nedeniyle eğitimi durdurması için durdurma kriterlerini ele alınmaktadır. Modelde durum, doğrulama doğruluğu ile ilgilidir, bu nedenle ağ önceki kontrol noktasından daha yüksek doğrulama doğruluğu

aldığında eğitim durdurulacaktır. Bu yöntem, öğrenciyi güncelleyerek eğitim sürecinin her yinelemesinde eğitim verilerine daha iyi uymasını sağlayacaktır [44].

```
- # Define callbacks.
- checkpoint_cb = keras.callbacks.ModelCheckpoint(
- "3d_image_classification.h5", save_best_only=True
- )
- early_stopping_cb = keras.callbacks.EarlyStopping(mo
  nitor="val_acc", patience=15)
```

4.3. Eğitim

Eğitim sürecinde, dönem sayısını, eğitim veri kümesini ve doğrulama veri kümesini tanımlamak doğru olacaktır. Ayrıca eğitim örneklerinin her eğitim döneminde karıştırılacağını da belirtmek de fayda vardır. Verbose=1, zaman, ölçümler, eğitimin ilerlemesi gibi tüm eğitim ayrıntılarını görüntüleme seçeneğidir.

Yeni dönem 25'tir (aşırı uydurmayı önlemek için)

```
- # Train the model, doing validation at the end of ea
  ch epoch
- epochs = 25
- history=model.fit(
- train_ds1,
- validation_data=val_ds1,
- epochs=epochs,
- shuffle=True,
- verbose=1,
- callbacks=[checkpoint_cb, early_stopping_cb],
- )
```

Şekil 4.4.'de bir eğitim süreci örneğini göstermektedir.

```
                                Epoch 1/25

10/10 [==============================] - 6s 168ms/step - loss: 0.8276 - acc: 0.5250 -
         MSE: 0.3017 - val_loss: 2.1464 - val_acc: 0.5000 - val_MSE: 0.4765

                                Epoch 2/25

10/10 [==============================] - 2s 149ms/step - loss: 0.7093 - acc: 0.5250 -
         MSE: 0.2580 - val_loss: 3.1710 - val_acc: 0.5000 - val_MSE: 0.4966

                                Epoch 3/25

10/10 [==============================] - 3s 270ms/step - loss: 0.7212 - acc: 0.4750 -
         MSE: 0.2640 - val_loss: 3.3971 - val_acc: 0.5000 - val_MSE: 0.4979

                                Epoch 4/25

10/10 [==============================] - 2s 156ms/step - loss: 0.7101 - acc: 0.5750 -
         MSE: 0.2552 - val_loss: 3.4094 - val_acc: 0.5000 - val_MSE: 0.4982

                                Epoch 5/25
```

```
10/10 [==============================] - 2s 154ms/step - loss: 0.6962 - acc: 0.5750 -
        MSE: 0.2517 - val_loss: 2.9813 - val_acc: 0.5000 - val_MSE: 0.4950

                                    Epoch 6/25

10/10 [==============================] - 2s 171ms/step - loss: 0.6658 - acc: 0.5750 -
        MSE: 0.2387 - val_loss: 2.3597 - val_acc: 0.5000 - val_MSE: 0.4861

                                    Epoch 7/25

10/10 [==============================] - 2s 167ms/step - loss: 0.6353 - acc: 0.5500 -
        MSE: 0.2236 - val_loss: 2.0089 - val_acc: 0.5000 - val_MSE: 0.4757

                                    Epoch 8/25

10/10 [==============================] - 2s 168ms/step - loss: 0.7133 - acc: 0.4750 -
        MSE: 0.2584 - val_loss: 1.5629 - val_acc: 0.5000 - val_MSE: 0.4454

                                    Epoch 9/25

10/10 [==============================] - 3s 262ms/step - loss: 0.6793 - acc: 0.6250 -
        MSE: 0.2423 - val_loss: 1.2556 - val_acc: 0.5000 - val_MSE: 0.4048

                                    Epoch 10/25

10/10 [==============================] - 2s 168ms/step - loss: 0.6285 - acc: 0.6500 -
        MSE: 0.2203 - val_loss: 0.9747 - val_acc: 0.5000 - val_MSE: 0.3445

                                    Epoch 11/25

10/10 [==============================] - 2s 168ms/step - loss: 0.7090 - acc: 0.5000 -
        MSE: 0.2578 - val_loss: 0.7653 - val_acc: 0.5000 - val_MSE: 0.2785

                                    Epoch 12/25

10/10 [==============================] - 2s 172ms/step - loss: 0.6460 - acc: 0.6250 -
        MSE: 0.2273 - val_loss: 0.7127 - val_acc: 0.5000 - val_MSE: 0.2587

                                    Epoch 13/25

10/10 [==============================] - 2s 206ms/step - loss: 0.6640 - acc: 0.6000 -
        MSE: 0.2359 - val_loss: 0.6643 - val_acc: 0.5500 - val_MSE: 0.2374

                                    Epoch 14/25

10/10 [==============================] - 2s 196ms/step - loss: 0.6566 - acc: 0.5250 -
        MSE: 0.2322 - val_loss: 0.6310 - val_acc: 0.6500 - val_MSE: 0.2209

                                    Epoch 15/25

10/10 [==============================] - 2s 165ms/step - loss: 0.6849 - acc: 0.6000 -
        MSE: 0.2455 - val_loss: 0.5862 - val_acc: 0.7000 - val_MSE: 0.1990

                                    Epoch 16/25

10/10 [==============================] - 2s 171ms/step - loss: 0.6197 - acc: 0.5750 -
        MSE: 0.2158 - val_loss: 0.5603 - val_acc: 0.6500 - val_MSE: 0.1883

                                    Epoch 17/25

10/10 [==============================] - 2s 169ms/step - loss: 0.7042 - acc: 0.5750 -
        MSE: 0.2496 - val_loss: 0.5510 - val_acc: 0.7000 - val_MSE: 0.1850

                                    Epoch 18/25

10/10 [==============================] - 2s 166ms/step - loss: 0.6571 - acc: 0.5750 -
        MSE: 0.2328 - val_loss: 0.5499 - val_acc: 0.6500 - val_MSE: 0.1842

                                    Epoch 19/25

10/10 [==============================] - 2s 201ms/step - loss: 0.7001 - acc: 0.5250 -
        MSE: 0.2505 - val_loss: 0.5608 - val_acc: 0.6500 - val_MSE: 0.1877

                                    Epoch 20/25
```

```
10/10 [==============================] - 2s 182ms/step - loss: 0.6069 - acc: 0.6750 -
         MSE: 0.2085 - val_loss: 0.5528 - val_acc: 0.6500 - val_MSE: 0.1841

                                    Epoch 21/25

10/10 [==============================] - 2s 167ms/step - loss: 0.6414 - acc: 0.6750 -
         MSE: 0.2248 - val_loss: 0.5325 - val_acc: 0.7500 - val_MSE: 0.1757

                                    Epoch 22/25

10/10 [==============================] - 2s 167ms/step - loss: 0.6606 - acc: 0.6000 -
         MSE: 0.2344 - val_loss: 0.5263 - val_acc: 0.8000 - val_MSE: 0.1733

                                    Epoch 23/25

10/10 [==============================] - 2s 159ms/step - loss: 0.6442 - acc: 0.6500 -
         MSE: 0.2252 - val_loss: 0.5271 - val_acc: 0.8000 - val_MSE: 0.1737

                                    Epoch 24/25

10/10 [==============================] - 2s 211ms/step - loss: 0.6239 - acc: 0.6750 -
         MSE: 0.2168 - val_loss: 0.5225 - val_acc: 0.8000 - val_MSE: 0.1727

                                    Epoch 25/25

10/10 [==============================] - 2s 189ms/step - loss: 0.6239 - acc: 0.6500 -
         MSE: 0.2166 - val_loss: 0.5177 - val_acc: 0.8000 - val_MSE: 0.1718
```

4.4. Test Senaryoları

Modelin performansını kontrol etmek için birçok test senaryosu uygulanmıştır. Bu senaryolar şunlardır

- 3D görüntülerin tüm dilimleri ve sadece 10 dönem kullanılması durumunda performans değerlendirmesi.
- 3D görüntülerin tüm dilimleri ve sadece 15 dönem kullanılması durumunda performans değerlendirmesi.
- 3D görüntülerin tüm dilimleri ve sadece 25 dönem kullanılması durumunda performans değerlendirmesi.
- 3D görüntülerin tüm dilimleri ve herhangi bir veri artırma olmaksızın yalnızca 5 dönem kullanılması durumunda performans değerlendirmesi (bu senaryo, önerilen veri artırma işleminin etkisini kontrol etmek için oldukça kullanışlıdır).

4.5. Model Değerlendirmesi

Bu adım, modelin genel değerlendirmelerini sunduğu ve inşa edildiği amaçları karşılayıp karşılamadığını kontrol ettiği için araştırmanın en önemli adımı olarak kabul edilmektedir.

Model değerlendirmesi için, karışıklık matrisi, eğitim doğruluğu, doğrulama doğruluğu, ortalama kare hatası (MSE), eğitim süresi, gerçek pozitifler TP, gerçek negatifler TN, yanlış pozitifler FP, yanlış negatifler FN, kesinlik, geri çağırma, F1 gibi birçok metrik kullanılmıştır.

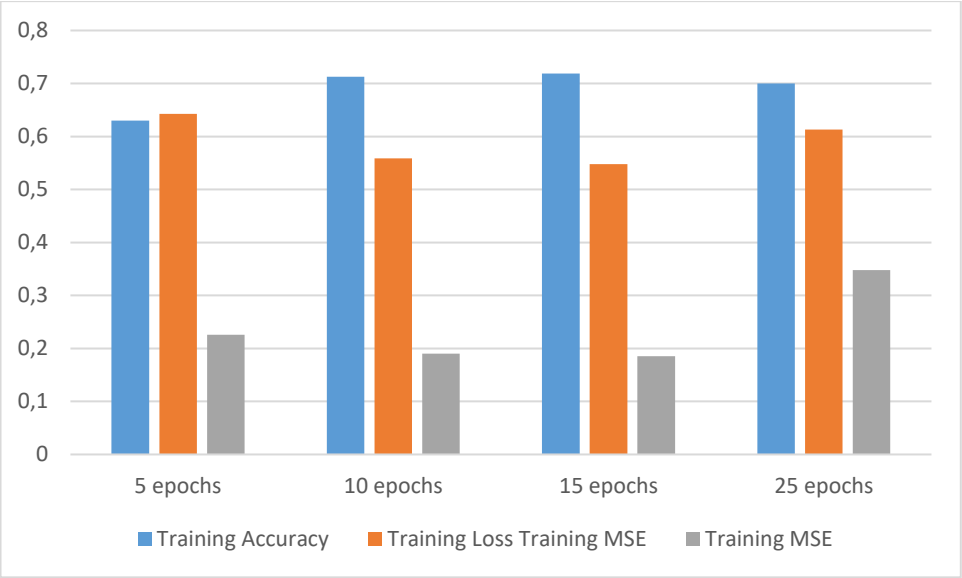
Tablo 4.1. 3D CNN modelinin performansı üzerindeki bu etkiyi görmek için farklı eğitim dönemlerinin kullanılmasının ayrıntılı bir karşılaştırmasını içermektedir. Karşılaştırmalar, eğitim doğruluğu, zaman, eğitim kaybı ve eğitim MSE'sini kapsamaktadır.

Tablo 4.1.eğitim dönemleri için en iyi seçimin, eğitim doğruluğu, doğrulama kaybı ve doğrulama MSE'nin diğer tüm durumların en iyi kombinasyonu olduğu 25 olduğu sonucuna varmaktadır.

Tablo 4.1. Veri Büyütme Kullanılarak Eğitim Dönemlerinin Değiştirilmesinin Karşılaştırmalı Eğitim Sonuçları

vaka	Eğitim Doğruluğu	Dönem Başına Eğitim Süresi	Eğitim Kaybı	MSE Eğitimi
5 epochs	0.63	37.6	0.6429	0.2257
10 epochs	0.7125	37.6	0.5589	0.1904
15 epochs	0.7188	38.66	0.5478	0.1851
25 epochs	0.71	2	0.6132	0.1985

Aşağıda yer alan Şekil 4.6. eğitim doğruluğu, eğitim kaybı ve eğitim MSE durumunda Tablo 4.1.'in karşılaştırmasını göstermektedir



Şekil 4.6. Veri Büyütmeyi Kullanarak Eğitim Dönemlerini Değiştirmenin Karşılaştırmalı Eğitim Sonuçları

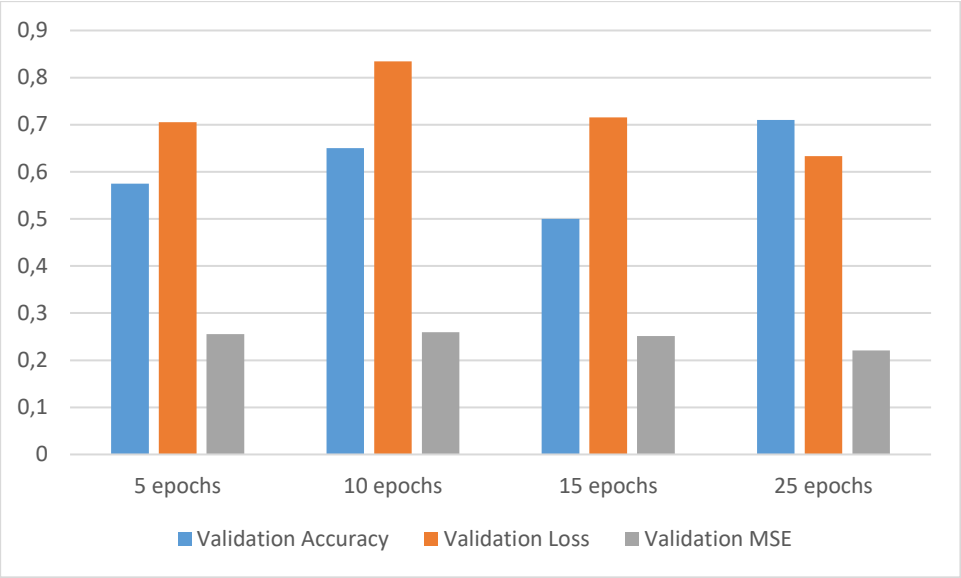
Tablo 4.2 ve Şekil 4.7. ise aynı karşılaştırmanın içinde doğrulama setini içermektedir. En iyi durumun, diğer tüm durumların en iyisi olan doğrulama doğruluğunun 65.5olduğu "25 dönem" olduğu sonucunu göstermektedir.

Bu sonucun temel sorunu, bu durumda doğruluk daha iyi olmasına rağmen, MSE ve Loss'un diğer durumlara göre daha yüksek değerlere sahip olmasıdır. Bunun nedeni, modelimizin bazı veri gruplarında iyi, diğerlerinde ise kötü olmasıdır.

Bu sonuç nedeniyle, veriler araştırılmakta ve her 3D görüntünün 64 diliminde derinlere inilmektedir. Modelin geliştirilmesi için ilk 12 ve son 8 dilimin akciğer dokusu olmadığı sonucuna ulaşılmış, bu yüzden onların çıkartılması uygun görülmüştür.

Tablo 4.2. Veri Büyütmeyi Kullanarak Eğitim Dönemlerini Değiştirmenin Karşılaştırmalı Doğrulama Sonuçları

Vaka	Doğrulama Doğruluğu	Doğrulama Kaybı	MSe Doğrulaması
5 epochs	0.575	0.7050	0.2552
10 epochs	0.65	0.8343	0.2593
15 epochs	0.500	0.7154	0.2514
25 epochs	0.71	0.6331	0.2206

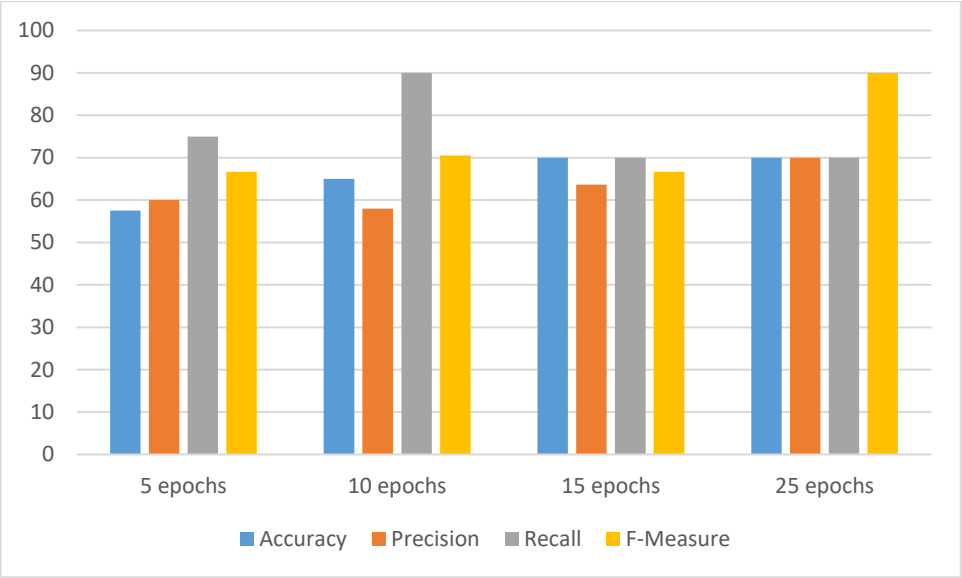


Şekil 4.7. Veri Büyütmeyi Kullanarak Eğitim Dönemlerini Değiştirmenin Karşılaştırmalı Doğrulama Sonuçları

Diğer performans metrikleri için, Kesinlik, Geri Çağırma ve F1-ölçümü, Tablo **4.3.**ve Şekil 4.8**.** en iyi durumun aynı zamanda kesinliğin %70.00 doğrulamada kesinlik %70.00 ve F1-ölçümünün %90.00 olduğu 25 dönem olduğunu göstermektedir.

Tablo 4.3. Veri Büyütme Kullanılarak Eğitim Dönemlerinin Değiştirilmesinin Karşılaştırmalı Performans Değerlendirme Sonuçları

Vaka	Doğrulama Doğruluğu	Doğrulamada Kesinlik	Doğrulamada Geri Çağırma	F-Ölçü Doğrulaması
5 epochs	57.5	60	75	66.67
10 epochs	65	58	90	70.54
15 epochs	70	63.64	70	66.67
25 epochs	70	64.63	70.00	66.67

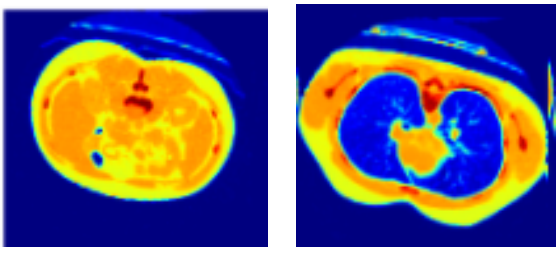


Şekil 4.8. Veri Büyütmeyi Kullanarak Eğitim Dönemlerini Değiştirmenin Karşılaştırmalı Performans Değerlendirme Sonuçları

4.5.1. Akciğer dışı doku görüntülerinin çıkarılmasının etkisinin incelenmesi

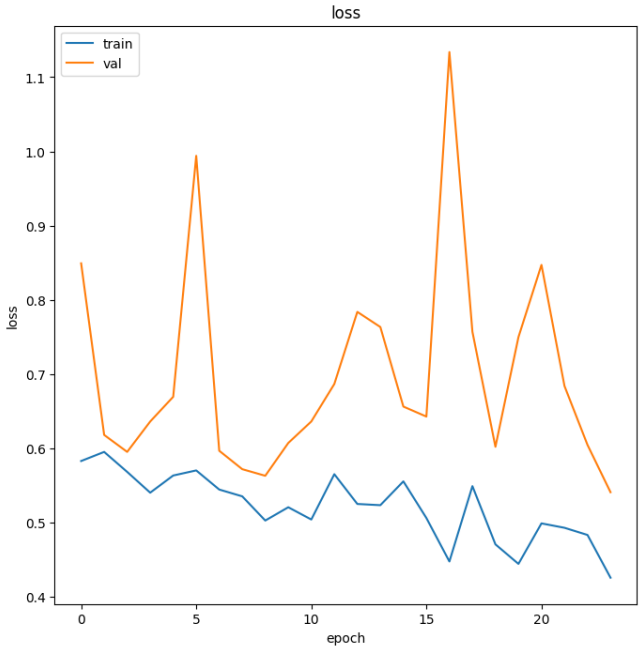
Bu deneyde, her 3B görüntünün ilk 12 dilimi, aynı görüntünün ikinci 12 dilimiyle değiştirilerek kaldırılmıştır. Ayrıca son 8 dilimi kaldırılarak onlardan önceki 8 dilimle değiştirilmiştir.

Bu işlem, eğitimi çok iyi etkileyebilecek akciğer dışı dokuları ortadan kaldıracak ve bu dokuların değiştirilmesi için önerilen 3D CNN modelinin performansı üzerindeki etkisini artıracaktır. (Şekil **4.9.** akciğer dışı sorun dilim şeklini göstermektedir).

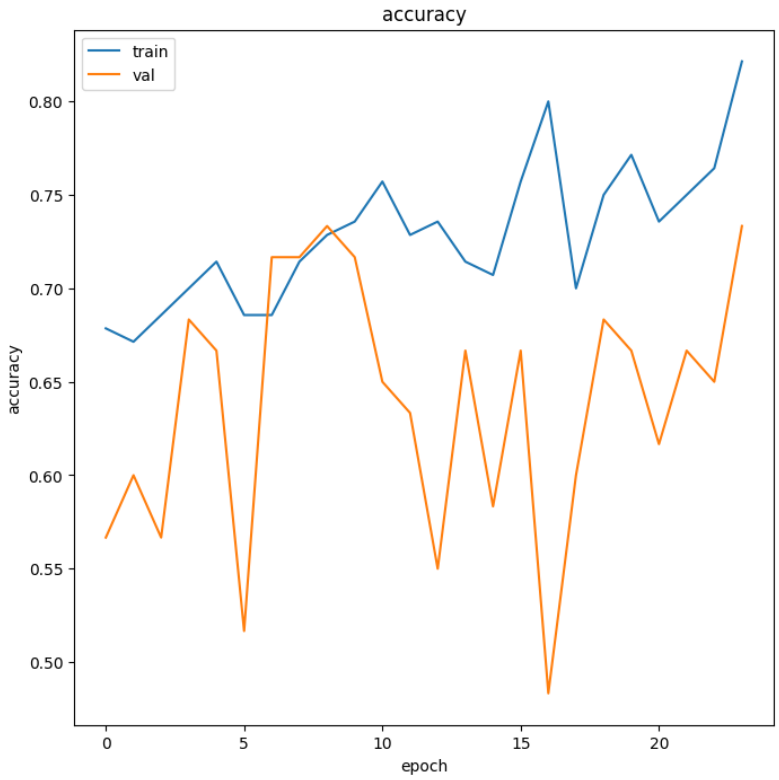


Şekil 4.9. Görüntünün Biri Akciğer Dokusu Olmayan Diğeri İse Mükemmel Akciğer

Şekil 4.10. aynı modelin eğitim ve doğrulama doğruluğunu gösterir.



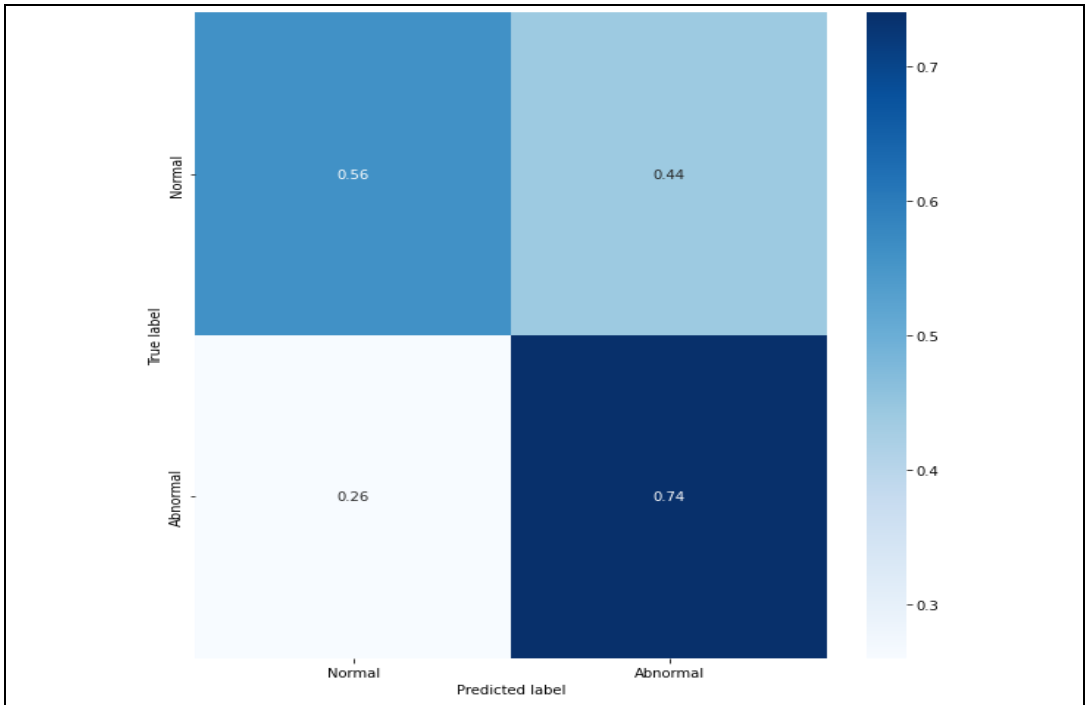
Şekil 4.9. En İyi Modelin Yinelemelerine Karşı Kayıp



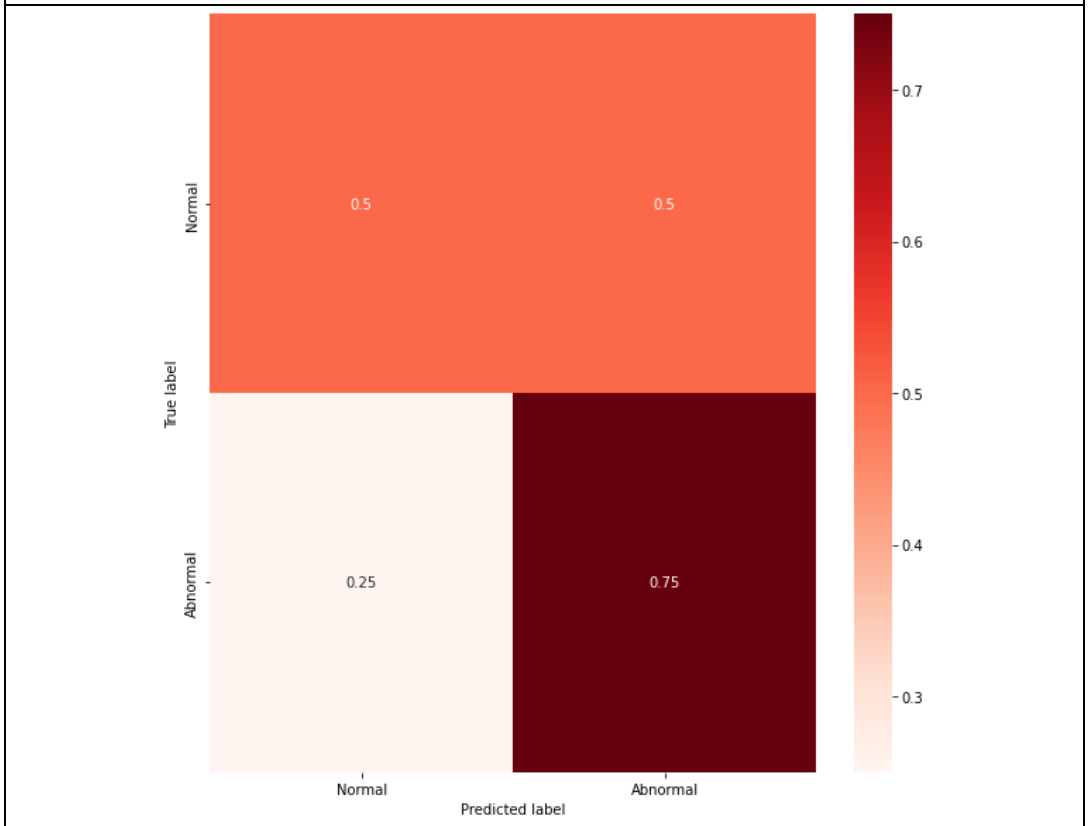
Şekil 4.10. Doğruluk (Eğitim Ve Doğrulama) ve En Iyi Modelin Yinelemeleri.

En iyi model, eğitim ve doğrulama için sırasıyla %70 ve %71 doğruluk elde eder.

4.5.2. Karışıklık matrisi CM sonuçları

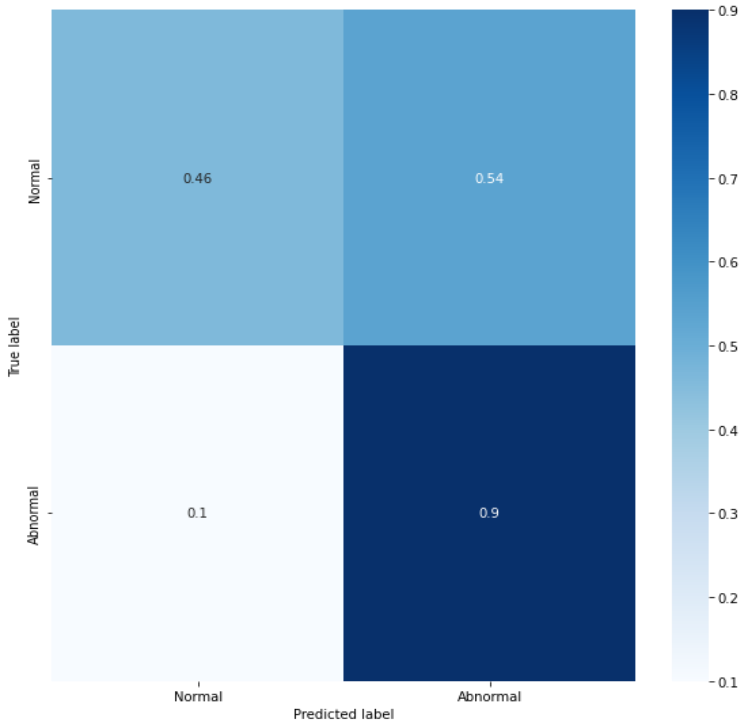


Eğitimin CM'si (5 dönem)

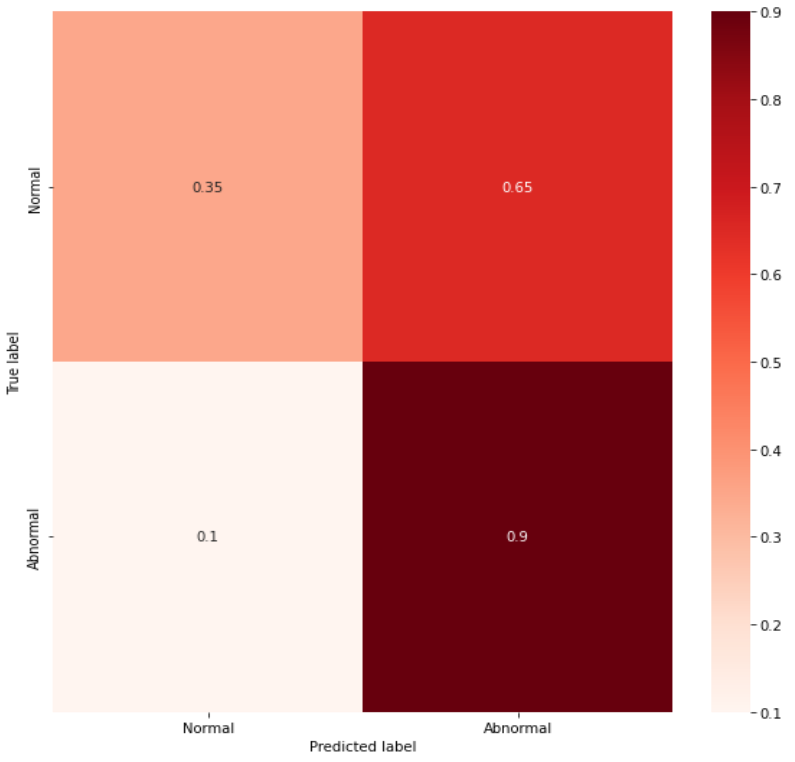


TESTİN CM'si (5 dönem)

Şekil 4.11. Veri Büyütmeyi Kullanarak Eğitim Dönemlerini Değiştirmenin Karşılaştırmalı Performans Değerlendirme Sonuçları

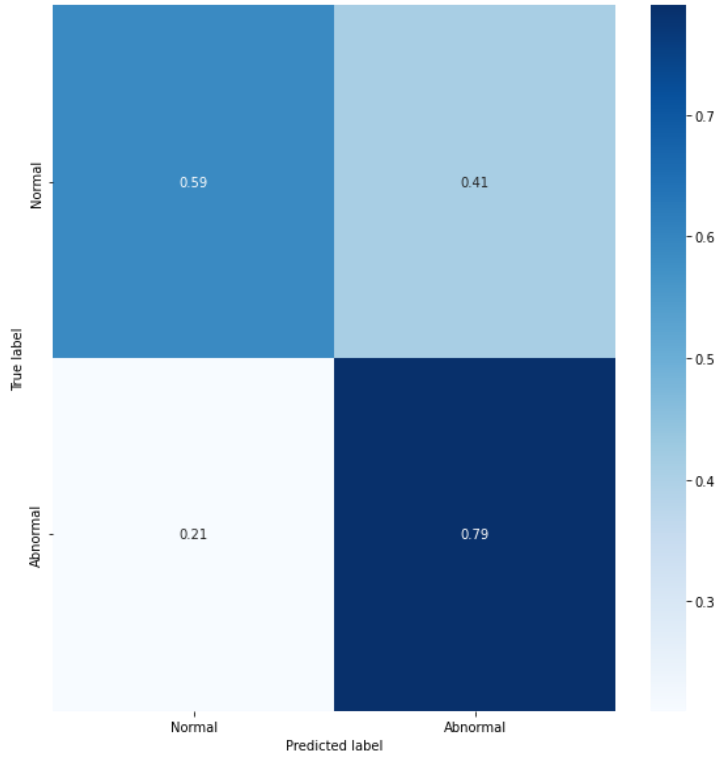


Eğitimin CM'si (10 dönem)

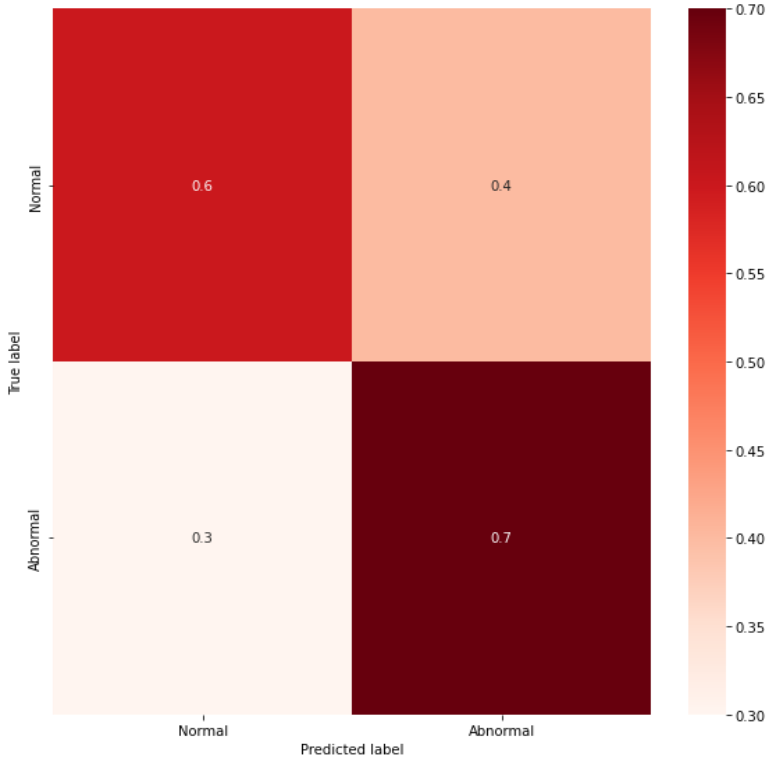


TESTİN CM'si (10 dönem)

Şekil 4.11. (Devamı) Veri Büyütmeyi Kullanarak Eğitim Dönemlerini Değiştirmenin Karşılaştırmalı Performans Değerlendirme Sonuçları

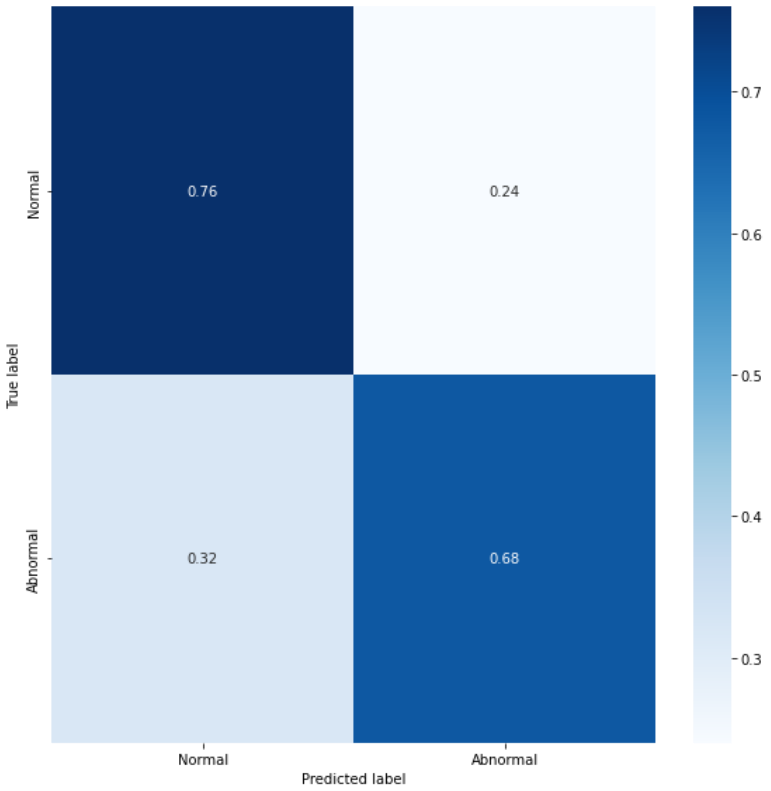


Eğitimin CM'si (15 dönem)

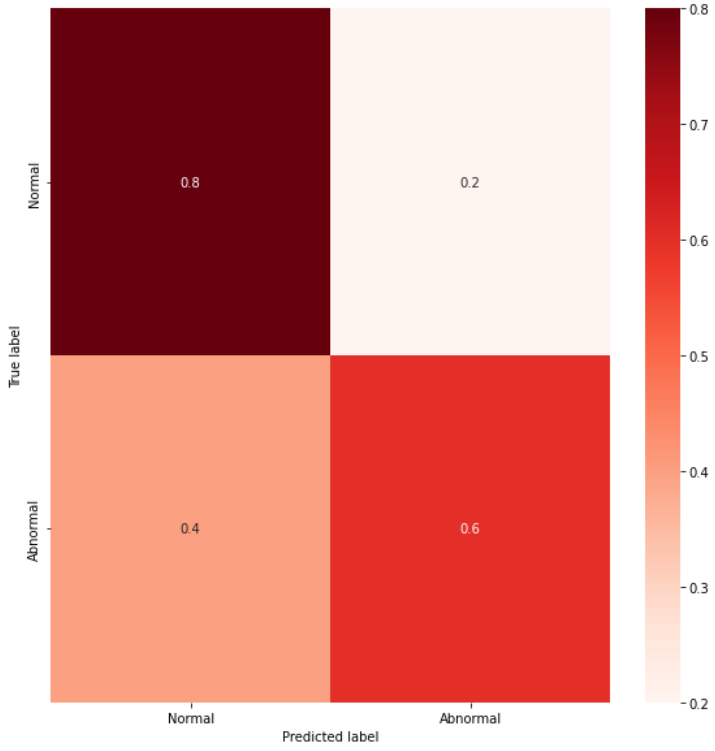


TESTİN CM'si (15 dönem)

Şekil 4.11. (Devamı) Veri Büyütmeyi Kullanarak Eğitim Dönemlerini Değiştirmenin Karşılaştırmalı Performans Değerlendirme Sonuçları



Eğitimin CM'si (25 dönem)



TESTİN CM'si (25 dönem)

Şekil 4.11. (Devamı) Veri Büyütmeyi Kullanarak Eğitim Dönemlerini Değiştirmenin Karşılaştırmalı Performans Değerlendirme Sonuçları

CM sonuçları (Şekil 4.13), önceki performans ölçümlerinden elde edilen sonuçların aynısını onaylar; en iyi model, akciğer dışı dokuların çıkarıldığı ve veri artırma ile 25 dönem kullanan modeldir. Akciğer dışı dokuların çıkarılması geliştirilmiştir.

5. SONUÇLAR VE TARTIŞMA

Gerçekleştirilen bu araştırmada derin öğrenme 3B CNN ağına dayalı tam bir 3B tıbbi görüntü tanıma modeli tasarlanmış ve uygulanmıştır. İlk olarak "MosMedData" adlı 3B görüntü veri seti kullanılmakta ve toplam 12800 dilim kullanılmaktadır. *(Normal akciğer dokuları için 100 3B dosya ve 100 anormal akciğer dokusu için).*

Ön işleme aşaması için akciğer dokularını elde etmek için boyutlandırma, yeniden ölçeklendirme ve diğer normalleştirme işlemleri gibi birçok işlem uygulanmıştır. İkinci adım, eğitimin sahte eğitim probleminden kaçınması için aynı görüntünün farklı çoklu kopyalarını elde etmek amacıyla görüntülerin döndürüldüğü, çevrildiği ve kontrastın değiştirildiği veri büyütme olarak ifade edilmektedir.

Eğitim modeli için 3D CNN ağının değiştirilmiş bir versiyonunun kullamanın uygun olacağı düşünülmektedir. Tam bağlantılı katmanı küresel ortalama havuz katmanıyla değiştirerek mimarinin değiştirildiği bu çalışmada son iki yoğun katman arasına bir bırakma katmanı eklenmiştir.

Daha sonra veri seti eğitim için %70 ,%20 test için ve %10 doğrulama olarak bölünmektedir. Çalışmada yapılan deneyler farklı eğitim dönemleri, farklı veri artırma durumları veya tüm 3D görüntülerin ya da yalnızca akciğer dokularının kullanıldığı birçok senaryoda uygulanabilmektedir.

Gerçekleştirilen çalışmada elde edilen sonuçlar en iyi dönem sayısının 25 olduğuna ulaşılırken en iyi durum için tüm akciğer 3D görüntüsü kullanmak yerine sadee akciğer dokularını kullanmanın doğru olduğunu ortaya çıkarmıştır.

Gelecekteki çalışmalar için, 3D CNN mimarisi aşağıdaki özelliklerle değiştirilebilir:

- Verileri daha iyi ölçeklendirmek için standardizasyon yöntemini uygulamak.
- Resnet101 gibi transfer öğrenme yöntemini kullanarak verileri eğitmek veya InceptionV3 derin sinir ağları.
- Verileri artırmak.

KAYNAKLAR

[1] Xie, Q., Tu, S., Wang, G., Lian, Y., & Xu, L. (2019). Feature Enrichment Based Convolutional Neural Network for Heartbeat Classification From Electrocardiogram. IEEE Access, 7, 153751–153760. https://doi.org/10.1109/access.2019.2948857.

[2] Moro, S., Cortez, P., & Rita, P. (2016). A framework for increasing the value of predictive data-driven models by enriching problem domain characterization with novel features. Neural Computing and Applications, 28(6), 1515–1523. https://doi.org/10.1007/s00521-015-2157-8.

[3] O'Shea, A., Lightbody, G., Boylan, G., & Temko, A. (2020). Neonatal seizure detection from raw multi-channel EEG using a fully convolutional architecture. Neural Networks, 123, 12–25. https://doi.org/10.1016/j.neunet.2019.11.023.

[4] Kyriakides, G., & Margaritis, K. (2021). Evolving graph convolutional networks for neural architecture search. Neural Computing and Applications. Published. https://doi.org/10.1007/s00521-021-05979-8.

[5] HAMOUDA, M., & BOUHLEL, M. S. (2021). Modified Convolutional Neural Networks Architecture for Hyperspectral Image Classification (Extra-Convolutional Neural Networks). IET Image Processing. Published. https://doi.org/10.1049/ipr2.12169.

[6] Audhkhasi, K., Osoba, O., & Kosko, B. (2016). Noise-enhanced convolutional neural networks. Neural Networks, 78, 15–23. https://doi.org/10.1016/j.neunet.2015.09.014.

[7] Singh, S., Wang, L., Gupta, S., Goli, H., Padmanabhan P., Gulyas, B . 3D Deep Learning on Medical Images: A Review. Sensors Journal., 20(18):1-24, 2020.

[8] Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet classification with deep convolutional neural networks.

[9] Commun. ACM 2017, 60, 84–90.

[10] Qi, X., Liao, R., Jia, J., Fidler, S., & Urtasun, R. 3d graph neural networks for rgbd semantic segmentation. In Proceedings of the IEEE International Conference on Computer Vision, :5199-5208, 2017.

[11] Chen, Y., Xie, Y., Zhou, Z., Shi, F., Christodoulou, A. G., & Li, D. (2018). Brain MRI super resolution using 3D deep densely connected neural networks. 2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018). doi:10.1109/isbi.2018.8363679

[12] Chen, S., Ma, K. and Zheng, Y., 2019. Med3d: Transfer learning for 3d medical image analysis. arXiv preprint arXiv:1904.00625, 2019.

[13] https://medium.com/@nabil.madali/introduction-to-3d-deep-learning-740c199b100c, Access Date: 10. 10.2021.

[14] Shi, L., Li, B., Kim, C., Kellnhofer, P., & Matusik, W. Towards real-time photorealistic 3D holography with deep neural networks. Nature, 591(7849), 234-239, 2021.

[15] Alalwan, N., Abozeid, A., ElHabshy, A. A., & Alzahrani, A. Efficient 3d deep learning model for medical image semantic segmentation. Alexandria Engineering Journal, 60(1):1231-1239, 2021.

[16] Serte, S., & Demirel, H. Deep learning for diagnosis of COVID-19 using 3D CT scans. Computers in biology and medicine, 132, 104306, 2021.

[17] Karimi, D., Vasylechko, S., & Gholipour, A. (2021). Convolution-Free Medical Image Segmentation using Transformers. arXiv preprint arXiv:2102.13645.

[18] B. Jefferson and R. S. Shanmugasundaram, Brain Tumor Classification in 3D-MRI using Features from Radiomics and 3D-CNN Combined with KNN Classifier, International Journal of Electrical Engineering and Technology (IJEET), 12(2):185-198, 2021.

[19] Maturana, Daniel, and Sebastian Scherer. "Voxnet: A 3d convolutional neural network for real-time object recognition." 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2015.

[20] Girod, L., Allen, M., Collier, T., Blumstein, D. T., Estrin, D., & Taylor, C. (2008). Experience with VoxNet: a rapidly-deployable acoustic monitoring system for bio-acoustic studies. The Journal of the Acoustical Society of America, 123(5), 2008. https://doi.org/10.1121/1.2932968.

[21] Qi, Charles R., Hao Su, Kaichun Mo, and Leonidas J. Guibas. "Pointnet: Deep learning on point sets for 3d classification and segmentation." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 652-660. 2017.

[22] https://www.geeksforgeeks.org/pointnet-deep-learning. Access Date: 10. 9.2021.

[23] Pimkin, A., Makarchuk, G., Kondratenko, V., Pisov, M., Krivov, E. and Belyaev, M., 2018, June. Ensembling neural networks for digital pathology images classification and segmentation. In International Conference Image Analysis and Recognition (pp. 877-886). Springer, Cham.

[24] Pavlov, N. A., Andreychenko, A. E., Vladzymyrskyy, A. V., Revazyan, A. A., Kirpichev, Y. S., & Morozov, S. P. (2021). Reference medical datasets (MosMedData) for independent external evaluation of algorithms based on artificial intelligence in diagnostics. Digital Diagnostics, 2(1), 49–66. https://doi.org/10.17816/dd60635.

[25] https://mosmed.ai/en/datasets/covid19_1110/ Access Date: 01.06.2021.

[26] https://arxiv.org/abs/2005.06465 Access Date: 01.06.2021.

[27] Morozov S.P., Andreychenko A.E., Blokhin I.A., et al. MosMedData: data set of 1110 chest CT scans performed during the COVID-19 epidemic // Digital Diagnostics. - 2020. - Vol. 1. - N. 1. - P. 49-59. doi: 10.17816/DD46826.

[28] Hosseini, K., & Sigloch, K. (2017). ObspyDMT: a Python toolbox for retrieving and processing large seismological data sets. Solid Earth, 8(5), 1047–1070. https://doi.org/10.5194/se-8-1047-2017.

[29] Snow, D. (2020). DeltaPy: A Framework for Tabular Data Augmentation in Python. SSRN Electronic Journal. Published. https://doi.org/10.2139/ssrn.3582219.

[30] Atila, O., & ŞEngür, A. (2021). Attention guided 3D CNN-LSTM model for accurate speech based emotion recognition. Applied Acoustics, 182, 108260. https://doi.org/10.1016/j.apacoust.2021.108260.

[31] https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 Access Date: 01.09.2021.

[32] Riva M, Batch Normalization in Convolutional Neural Networks, baeldung website, https://www.baeldung.com/cs/batch-normalization-cnn Access Date: 11.08.2021.

[33] Sledevic, Tomyslav. "Adaptation of convolution and batch normalization layer for CNN implementation on FPGA." In 2019 Open Conference of Electrical, Electronic and Information Sciences (eStream), pp. 1-4. IEEE, 2019.

[34] Brownlee J, A Gentle Introduction to Pooling Layers for Convolutional Neural Networks, Deep Learning for Computer Vision, 2019, https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks/.

[35] Gholamalinezhad, H., & Khosravi, H. (2020). Pooling Methods in Deep Neural Networks, a Review. arXiv preprint arXiv:2009.07485, 2020.

[36] Liukis A, The Basic Building Block of Neural Networks, towardsdatascience, 2020. https://towardsdatascience.com/the-basic-building-block-of-neural-networks-a9b2e8f5c056 Access Date: 11.08.2021.

[37] Kumar N, Batch Normalization and Dropout in Neural Networks with Pytorch, towardsdatascience, 2019, https://towardsdatascience.com/batch-normalization-and-dropout-in-neural-networks-explained-with-pytorch-47d7a8459bcd Access Date: 11.08.2021.

[38] Brownlee J, A Gentle Introduction to Dropout for Regularizing Deep Neural Networks, machinelearningmastery, 2019, https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/, Access Date: 11.08.2021.

[39] Sochat, V. (2020). GridTest: testing and metrics collection for Python. Journal of Open Source Software, 5(51), 2284. https://doi.org/10.21105/joss.02284.

[40] Sage. (2020, June 11). Performance Metrics Examples and Definition. Sage Advice US. https://www.sage.com/en-us/blog/glossary/what-are-performance-metrics/

[41] Metrics and scoring: quantifying the quality of predictions — scikit-learn 0.24.2 documentation. (2020). Metrics and Scoring: Quantifying the Quality of Predictions. https://scikit-learn.org/stable/modules/model_evaluation.html.

[42] https://blog.exsilio.com/all/accuracy-precision-recall-f1-score-interpretation-of-performance-measures/ Access Date: 01.08.2021.

[43] https://www.tensorflow.org/tutorials/images/data_augmentation Access Date: 01.08.2021.

[44] https://www.tensorflow.org/api_docs/python/tf/keras/layers/ Access Date: 01.08.2021.

[45] Raskutti, G., Wainwright, M. J., & Yu, B. (2014). Early stopping and non-parametric regression: an optimal data-dependent stopping rule. The Journal of Machine Learning Research, 15(1), 335-366.

EKLER

EK A - Performans Metriklerinin Hesaplanması

Case of 5 epochs

Validation:

[[10 10]

[5 15]]

	Normal	Abnormal
Normal	TN	FP
Abnormal	FN	TP

	Normal	Abnormal
Normal	0.50	0.50
Abnormal	0.25	0.75

Precision =TruePositives / (TruePositives + FalsePositives)

Precision = TP/TP+FP=0.75/(0.75+0.5)=0.6=60%

Recall = TruePositives / (TruePositives + FalseNegatives)

Recall = TP/TP+FN=0.75/(0.75+0.25)=0.75=75%

F-Measure = (2 * Precision * Recall) / (Precision + Recall)=66.67%

Acc= TP+TN/(TP+TN+FP+FN)= (0.75+0.5)/(0.75+0.5+0.25+0.5)=0.625=57.5%

Case of 10 epochs

Validation set

[[7 13]

[2 18]]

	Normal	Abnormal
Normal	0.35	0.65

Abnormal 0.10 0.90

Precision = TP/TP+FP=0.9/(0.9+0.65)=0.58=58%

Recall = TP/TP+FN=0.9/(0.9+0.1)=0.9=90%

F-Measure = (2 * Precision * Recall) / (Precision + Recall)=70.54%

Acc= TP+TN/(TP+TN+FP+FN)= (0.9+0.35)/(0.9+0.35+0.65+0.1)=0.65=65.0%

Case of 15 epochs

Validation set

[[12 8]

[6 14]]

Normal Abnormal

Normal 0.6 0.4

Abnormal 0.3 0.7

Precision = TP/TP+FP=0.7/(0.7+0.4)=0.6364=63.64%

Recall = TP/TP+FN=0.7/(0.7+0.3)=0.7=70%

F-Measure = (2 * Precision * Recall) / (Precision + Recall)=66.67%

EK - B

Code

```
!pip install -q imageio

!pip install -q opencv-python

!pip install -q git+https://github.com/tensorflow/docs

import imageio
from IPython import display
from urllib import request
import zipfile
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from absl import logging
import tensorflow_hub as hub
import random
import nibabel as nib
from scipy import ndimage
import cv2
import numpy as np
import os
import tempfile
import tensorflow_hub as hub
from keras.models import Sequential
from tensorflow.keras.layers import Conv3D
from keras.layers import Activation, Dense
#from __future__ import absolute_import, division, print_function, unicode_literals
import matplotlib.pylab as plt
import tensorflow as tf
```

```
import tensorflow_hub as hub

import pandas as pd


logging.set_verbosity(logging.ERROR)


#from google.colab import drive

#drive.mount('/content/drive')


url = "https://github.com/hasibzunair/3D-image-classification-
tutorial/releases/download/v0.2/CT-0.zip"

#url = "/content/drive/MyDrive/CT-0.zip"

url1 = "https://github.com/hasibzunair/3D-image-classification-
tutorial/releases/download/v0.2/CT-23.zip"

#url1 = "/content/drive/MyDrive/CT-0.zip"



filename = os.path.join(os.getcwd(), "CT-0.zip")

filename1 = os.path.join(os.getcwd(), "CT-23.zip")

keras.utils.get_file(filename, url)

keras.utils.get_file(filename1, url1)

with zipfile.ZipFile("CT-0.zip", "r") as z_fp:

    z_fp.extractall("./MosMedData/")

with zipfile.ZipFile("CT-23.zip", "r") as z_fp:

    z_fp.extractall("./MosMedData/")



normal_scan_paths = [

    os.path.join(os.getcwd(), "MosMedData/CT-0", x)

    for x in os.listdir("MosMedData/CT-0")

]

abnormal_scan_paths = [

    os.path.join(os.getcwd(), "MosMedData/CT-23", x)
```

```
    for x in os.listdir("MosMedData/CT-23")
]
print("CT scans with normal lung tissue: " + str(len(normal_scan_paths)))
print("CT scans with abnormal lung tissue: " + str(len(abnormal_scan_paths)))

def read_nifti_file(filepath):
    scan = nib.load(filepath)
    scan = scan.get_fdata()
    return scan

def resize_volume(img):
    desired_depth = 64
    desired_width = 128
    desired_height = 128
    current_depth = img.shape[-1]
    current_width = img.shape[0]
    current_height = img.shape[1]
    depth = current_depth / desired_depth
    width = current_width / desired_width
    height = current_height / desired_height
    depth_factor = 1 / depth
    width_factor = 1 / width
    height_factor = 1 / height
    img = ndimage.rotate(img, 90, reshape=False)
    img = ndimage.zoom(img, (width_factor, height_factor, depth_factor), order=1)
    return img

def normalize(volume):
    min = -1000
    max = 400
    volume[volume < min] = min
```

```
    volume[volume > max] = max
    volume = (volume - min) / (max - min)
    volume = volume.astype("float32")
    return volume

def process_scan(path):
    volume = read_nifti_file(path)
    volume = normalize(volume)
    volume = resize_volume(volume)
    return volume

#normal scans
n_scans1 = np.array([process_scan(path) for path in normal_scan_paths])
#abnormal scans
a_scans1 = np.array([process_scan(path) for path in abnormal_scan_paths])

#n_scans=n_scans1[:,:,:,12:56]
#a_scans=a_scans1[:,:,:,12:56]
n_scans=n_scans1
a_scans=a_scans1
print(n_scans.shape)

n_labels = np.array([0 for _ in range(len(n_scans))])
a_labels = np.array([1 for _ in range(len(a_scans))])

x_train = np.concatenate((a_scans[:70], n_scans[:70]), axis=0)
y_train = np.concatenate((a_labels[:70], n_labels[:70]), axis=0)
x_val = np.concatenate((a_scans[90:], n_scans[90:]), axis=0)
y_val = np.concatenate((a_labels[90:], n_labels[90:]), axis=0)

print(
```

```
    "Number of samples in train is:  %d.  Number of samples in Validation is %d."
    % (x_train.shape[0], x_test.shape[0], x_val.shape[0])
)

IMG_SIZE = 128

resize_and_rescale = tf.keras.Sequential([
  layers.experimental.preprocessing.Resizing(IMG_SIZE, IMG_SIZE),

  #layers.experimental.preprocessing.Rescaling(1./255)
])

data_augmentation = tf.keras.Sequential([
  layers.experimental.preprocessing.RandomFlip("horizontal_and_vertical"),
  layers.experimental.preprocessing.RandomRotation(0.5),
  layers.experimental.preprocessing.RandomContrast([0.5, 1.0]),
 # layers.experimental.preprocessing.RandomZoom(.2, .2)
])



# Prepare dataset (before augmentation)
train_loader = tf.data.Dataset.from_tensor_slices((x_test, y_test))
validation_loader = tf.data.Dataset.from_tensor_slices((x_val, y_val))
test_loader = tf.data.Dataset.from_tensor_slices((x_test, y_test))

# Plot a sample of the dataset before augmentation process
data = train_loader.take(1)
image, label = list(data)[0]
print("Dimension of the CT scan is:", image.shape)
plt.imshow((image[:, :, 30]), cmap="jet")
#
```

```
# Plotting 64 slices of a sample before the data augmentation
def plot_slices(num_rows, num_columns, width, height, data):
    """Plot a montage of all 64 CT slices"""
    data = np.rot90(np.array(data))
    data = np.transpose(data)
    data = np.reshape(data, (num_rows, num_columns, width, height))
    rows_data, columns_data = data.shape[0], data.shape[1]
    heights = [slc[0].shape[0] for slc in data]
    widths = [slc.shape[1] for slc in data[0]]
    fig_width = 12.0
    fig_height = fig_width * sum(heights) / sum(widths)
    f, axarr = plt.subplots(
        rows_data,
        columns_data,
        figsize=(fig_width, fig_height),
        gridspec_kw={"height_ratios": heights},
    )
    for i in range(rows_data):
        for j in range(columns_data):
            axarr[i, j].imshow(data[i][j], cmap="jet")
            axarr[i, j].axis("off")
    plt.subplots_adjust(wspace=0, hspace=0, left=0, right=1, bottom=0, top=1)
    plt.show()
plot_slices(8, 8, 128, 128, image[:, :, :64])

# reference source https://www.tensorflow.org/tutorials/images/data_augmentation
batch_size = 4
AUTOTUNE = tf.data.AUTOTUNE

def prepare(ds, shuffle=False, augment=False):
```

```
 # Resize and rescale all datasets
# ds = ds.map(lambda x, y: (resize_and_rescale(x), y),
 #           num_parallel_calls=AUTOTUNE)

 if shuffle:
   ds = ds.shuffle(100)

 # Batch all datasets
 ds = ds.batch(batch_size)

 # Use data augmentation only on the training set
 if augment:
   ds = ds.map(lambda x, y: (data_augmentation(x, training=True), y),
               num_parallel_calls=AUTOTUNE)

 # Use buffered prefecting on all datasets
 return ds.prefetch(buffer_size=2)

# See how the data augmentation process affect the datasets
data = train_loader.take(1)
image, label = list(data)[0]

print("Dimension of the CT scan is:", image.shape)

plt.figure(figsize=(10, 10))
for i in range(16):
  augmented_image = data_augmentation(image)
  ax = plt.subplot(4, 4, i + 1)
  plt.imshow((augmented_image[:, :, 30]), cmap="jet")
  plt.axis("off")
a=(augmented_image);
```

```
print(a.shape)



# show sequense of slices of the same 3D image
from IPython.display import clear_output
from numpy.random import randn
from time import sleep

for i in range(20):
  clear_output()
  #plt.cla()
  #plt.axis([0, 128, 0, 128])
  plt.imshow(np.squeeze(image[:, :, i]), cmap="jet")
  plt.show()
  plt.pause(1.5)
  #sleep(1.5)

# Apply the data augmentation on the training and validation datasets
def train_preprocessing(volume, label):
    """Process training data by rotating and adding a channel."""
    # Rotate volume
    volume = tf.expand_dims(volume, axis=3)
    return volume, label



def validation_preprocessing(volume, label):
    """Process validation data by only adding a channel."""
    volume = tf.expand_dims(volume, axis=3)
    return volume, label
train_ds1 = prepare(train_loader, shuffle=True, augment=True)
```

```
train_ds1.map(train_preprocessing)
train_ds1.batch(4);
train_ds1.prefetch(2);

val_ds1 = prepare(validation_loader, shuffle=True, augment=False)
val_ds1.map(validation_preprocessing)
val_ds1.batch(4);
val_ds1.prefetch(2);

test_ds1 = prepare(test_loader, shuffle=False, augment=False)

print(train_ds1.snapshot)
print(val_ds1.snapshot)
print(train_ds1.snapshot)



# plotting an example of the dataset samples after applying the data augmentation
data = train_ds1.take(1)
images, labels = list(data)[0]
images = images.numpy()
image = images[0]
print("Dimension of the CT scan is:", image.shape)
plt.imshow(np.squeeze(image[:, :, 30]), cmap="jet")
plot_slices(8, 8, 128, 128, image[:, :, :64])

model = keras.models.Sequential()

def get_model(width=128, height=128, depth=64):
    """Build a 3D convolutional neural network model."""

    inputs = keras.Input((width, height, depth, 1))
```

```
    mx = layers.Conv3D(filters=64, kernel_size=3, activation="relu")(inputs)
    mx = layers.MaxPool3D(pool_size=2)(mx)
    mx = layers.BatchNormalization()(mx)

    mx = layers.Conv3D(filters=64, kernel_size=3, activation="relu")(mx)
    mx = layers.MaxPool3D(pool_size=2)(mx)
    mx = layers.BatchNormalization()(mx)

    mx = layers.Conv3D(filters=128, kernel_size=3, activation="relu")(mx)
    mx = layers.MaxPool3D(pool_size=2)(mx)
    mx = layers.BatchNormalization()(mx)

    mx = layers.Conv3D(filters=256, kernel_size=3, activation="relu")(mx)
    mx = layers.MaxPool3D(pool_size=2)(mx)
    mx = layers.BatchNormalization()(mx)

    mx = layers.GlobalAveragePooling3D()(mx)
    mx = layers.Dense(units=512, activation="relu")(mx)
    mx = layers.Dropout(0.3)(mx)

    outputs = layers.Dense(units=1, activation="sigmoid")(mx)

    # Define the model.
    model = keras.Model(inputs, outputs, name="3dcnn")
    return model


# Build model.
model = get_model(width=128, height=128, depth=64)

model.summary()
```

```
# Compile model.
initial_learning_rate = 0.001
lr_schedule = keras.optimizers.schedules.ExponentialDecay(
    initial_learning_rate, decay_steps=100000, decay_rate=0.96, staircase=True
)



model.compile(
    loss="binary_crossentropy",
    optimizer=keras.optimizers.SGD(learning_rate=lr_schedule),
    momentum=[0.9 ],
)

# Define callbacks.
checkpoint_cb = keras.callbacks.ModelCheckpoint(
    "3d_image_classification.h5", save_best_only=True
)
early_stopping_cb = keras.callbacks.EarlyStopping(monitor="val_acc", patience=15)



# Train the model, doing validation at the end of each epoch
epochs = 25
history=model.fit(
    train_ds1,
    validation_data=val_ds1,
    epochs=epochs,
    shuffle=True,
    validation_steps=2,
```

```
    verbose=1,
    callbacks=[checkpoint_cb, early_stopping_cb],
)

model.load_weights("3d_image_classification.h5")
prediction = model.predict(np.expand_dims(x_test[1], axis=0))[0]
print(prediction)
scores = [1 - prediction[0], prediction[0]]

class_names = ["noraml", "abnormal"]
for score, name in zip(scores, class_names):
    print(
        "The model  %.2f percent confident that scan is: %s"
        % ((100 * score), name)
    )

# Confusion matrix computations
from sklearn.metrics import confusion_matrix
def CM(y_true, y_pred):
  CM1=confusion_matrix(y_true, y_pred)
  return CM1

# evaluation of the training samples
predictions=np.zeros(len(x_train), dtype = int)
for i in range(len(x_train)):
  predictions[i] = np.round(model.predict(np.expand_dims(x_train[i], axis=0))[0])
  #print(predictions)
print(CM(y_train, predictions))

# Show confusion matrix of training samples as percentge
y_pred=predictions
```

```
con_mat = tf.math.confusion_matrix(labels=y_train, predictions=y_pred).numpy()
con_mat_norm = np.around(con_mat.astype('float') / con_mat.sum(axis=1)[:, np.newaxis], decimals=2)
classes=["Normal","Abnormal"]
con_mat_df = pd.DataFrame(con_mat_norm,
                 index = classes,
                 columns = classes)
print(con_mat_df)

# Plot the confusion matrix of the training set
import seaborn as sns
figure = plt.figure(figsize=(8, 8))
sns.heatmap(con_mat_df, annot=True,cmap=plt.cm.Blues)
plt.tight_layout()
plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.show()

# evaluation of the test dataset
predictions=np.zeros(len(x_test), dtype = int)
for i in range(len(x_test)):
  predictions[i] = np.round(model.predict(np.expand_dims(x_test[i], axis=0))[0])
  #print(predictions)
print(CM(y_test, predictions))

# Show confusion matrix of the test set as percentge
y_pred=predictions
con_mat = tf.math.confusion_matrix(labels=y_test, predictions=y_pred).numpy()
con_mat_norm = (con_mat.astype('float') / con_mat.sum(axis=1)[:, np.newaxis])
classes=["Normal","Abnormal"]
con_mat_df = pd.DataFrame(con_mat_norm,
```

```
                index = classes,
                columns = classes)
print(con_mat_df)

# Plot the confusion matrix of the validation set
import seaborn as sns
figure = plt.figure(figsize=(8, 8))
sns.heatmap(con_mat_df, annot=True,cmap=plt.cm.Reds)
plt.tight_layout()
plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.show()

import sys
from keras import metrics
import numpy as K
# def recall(y_true, y_pred):
#     # y_true = K.ones_like(y_true)
#     true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
#     all_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
#     recall = true_positives / (all_positives)
#     return recall

# def precision(y_true, y_pred):
#     # y_true = K.ones_like(y_true)
#     true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))

#     predicted_positives = K.sum(K.round(K.clip(y_pred, 0, 1)))
#     precision = true_positives / (predicted_positives)
#     return precision
```

```
# def f1_score(y_true, y_pred):
#     precision1 = precision(y_true, y_pred)
#     recall1 = recall(y_true, y_pred)
#     return 2*((precision1*recall1)/(precision1+recall1))

from sklearn.metrics import precision_score,recall_score,f1_score
y_true=y_test
y_pred=predictions
recall_test=recall_score(y_true, y_pred)
print("The recall is: %.2f" % (100*recall_test)+"%")
precision_test=precision_score(y_true, y_pred)
print("The precision is: %.2f" % (100*precision_test)+"%")
f1_score_test=f1_score(y_true, y_pred)
print("The f1_score is: %.2f" % (100*f1_score_test)+"%")



#saving the  model to be used later
fer_json = model.to_json()
with open("f1.json", "w") as json_file:
   json_file.write(fer_json)
model.save_weights("f1.h5")
print("Saved")

pd.set_option("display.precision", 8)

val_image_batch, val_label_batch = next(iter(train_ds1))
true_label_ids = np.argmax(val_label_batch, axis=-1)
```

```
print("Validation batch shape:", val_image_batch.shape)


tf_model_predictions = model.predict(val_image_batch)
tf_pred_dataframe = pd.DataFrame(tf_model_predictions)
tf_pred_dataframe.columns = ["normal"]
print("Prediction results for the first elements")
tf_pred_dataframe.head()


print(history.history)
# summarize history for loss
fig = plt.figure(figsize=(8,8))
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
# summarize history for accuracy
fig = plt.figure(figsize=(8,8))
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```

ÖZGEÇMİŞ

Ad-Soyad : Rouba OMAR ALAHMAD ALOSMAN

ÖĞRENİM DURUMU:

- **Lisans :** 2016-2017, Arab Open university, Fen Bilimleri, Bilişim teknolojisi ve Bilgisayar
- **Yükseklisans :** 2022-2023,Sakarya Üniversitesi, Bilişim sistemleri Mühendisliği, Bilişim sistemleri Mühendisliği.

MESLEKİ DENEYİM VE ÖDÜLLER:

- 2019yılında Sakarya Üniversitesi Tömer'de Başarı Ödülü'nü kazandı.
- 2016-2017 Arab Gazal Generel Şirket / Muhasebe sektörü

TEZDEN TÜRETİLEN ESERLER:

- **Yayın kınferans :** International Conference on Cyber Security and Digital Forensics (ICONSEC'21), June 4-5, 2021, Yalova, TURKEY, A Model for Header Compression Context Transfer in Cellular IP