Derin Öğrenme Teknikleriyle Tomografi Görüntülerinde Karaciğer Bölütlemesi

Furkan Avcı

YÜKSEK LİSANS TEZİ

Bilgisayar Mühendisliği Anabilim Dalı

Ağustos 2019

Liver Segmentation On Tomography Images With Deep Learning Techniques

Furkan Avcı

MASTER OF SCIENCE THESIS

Department of Computer Engineering

August 2019

Derin Öğrenme Teknikleriyle Tomografi Görüntülerinde Karaciğer Bölütlemesi

Furkan Avcı

Eskişehir Osmangazi Üniversitesi
Fen Bilimleri Enstitüsü
Lisansüstü Yönetmeliği Uyarınca
Bilgisayar Mühendisliği Anabilim Dalı
Bilgisayar Yazılımı Bilim Dalında
YÜKSEK LİSANS TEZİ
Olarak Hazırlanmıştır

Danışman: Doç. Dr. Eyyüp Gülbandılar

Ağustos 2019

ONAY

Bilgisayar Mühendisliği Anabilim Dalı Yüksek Lisans öğrencisi Furkan AVCI'nın YÜKSEK LİSANS tezi olarak hazırladığı "Derin Öğrenme Teknikleriyle Tomografi Görüntülerinde Karaciğer Bölütlemesi" başlıklı bu çalışma, jürimizce lisansüstü yönetmeliğin ilgili maddeleri uyarınca değerlendirilerek oybirliği ile kabul edilmiştir.

Danışman **:** Doç.Dr. Eyyüp Gülbandılar

İkinci Danışman **:** -

Yüksek Lisans Tez Savunma Jürisi:

Üye : Doç.Dr. Eyyüp Gülbandılar

Üye : Doç. Dr. Kemal Özkan

Üye : Dr. Öğr. Üyesi Muammer Akçay

Fen Bilimleri Enstitüsü Yönetim Kurulu'nun ..... tarih ve ..... sayılı kararıyla onaylanmıştır.

Prof. Dr. Hürriyet Erşahan
Enstitü Müdürü

ETİK BEYAN

Eskişehir Osmangazi Üniversitesi Fen Bilimleri Enstitüsü tez yazım kılavuzuna göre, Doç.Dr. Eyyüp Gülbandılar danışmanlığında hazırlamış olduğum "Derin Öğrenme Teknikleriyle Tomografi Görüntülerinde Karaciğer Bölütlemesi" başlıklı YÜKSEK LİSANS tezimin özgün bir çalışma olduğunu; tez çalışmamın tüm aşamalarında bilimsel etik ilke ve kurallara uygun davrandığımı; tezimde verdiğim bilgileri, verileri akademik ve bilimsel etik ilke ve kurallara uygun olarak elde ettiğimi; tez çalışmamda yararlandığım eserlerin tümüne atıf yaptığımı ve kaynak gösterdiğimi ve bilgi, belge ve sonuçları bilimsel etik ilke ve kurallara göre sunduğumu beyan ederim. 22/08/2019

Furkan Avcı
İmza

ÖZET

Günümüzde, yapay zeka teknolojilerindeki gelişmeler ile birlikte yeniden önem kazanan derin öğrenme teknikleri ile birçok bilimsel ve akademik çalışmalar yapılmaktadır. Bununla birlikte gelişen medikal görüntüleme teknikleriyle çok detaylı teşhise imkan sağlayan kaliteli görüntüler üretilmektedir. Vücudumuzdaki önemli organlardan biri olan karaciğerin rahatsızlıklarında önemli bir yer tutan karaciğer nakli veya cerrahi operasyonları öncesinde, radyolojik görüntülerden elde edilen bulguların değerlendirilmesi çok önemli bir yer tutmaktadır. Bu görüntülerin 3 boyutlu incelenmesinin ön adımı ise çekilmiş görüntülerden karaciğer dokusunun ayrılması aşamasıdır. Bu tez kapsamında derin öğrenme yapılarından biri olan evrişimli sinir ağlarıyla; çeşitli girdi boyutları ve aktivasyon fonksiyonlarıyla oluşturulmuş modellerin örnek veri seti ile başarımları ölçülerek, sonuçları değerlendirilmiştir. Deneyler sonucunda U-NET yapısı ile 256x256 ve 512x512 görüntüler üzerinden 97.58% ve 94.97% F-measure değerleri elde edilmiştir. Basit ESA yapısı ve Sigmoid ve Linear aktivasyon fonksiyonuna sahip U-NET modelleri (ortak eşik değer için) başarım gösterememiştir.

Anahtar kelimeler: Derin öğrenme, yapay zeka, evrişimli sinir ağları, karaciğer bölütleme, tomografi

SUMMARY

Nowadays, many scientific and academic studies are carried out with deep learning techniques, which have gained importance once again with the developments in artificial intelligence technologies. On the other hand, high quality images are produced which enable very detailed diagnosis with developing medical imaging techniques. The evaluation of the findings obtained from radiological images prior to liver transplantation or surgical operations, which play an important role in liver disorders, is one of the most important organs in our body. The preliminary step of the 3-D examination of these images is the separation of liver tissue from the images taken. Within the scope of this thesis, the performance of the models created with various input dimensions and activation functions measured with the sample data set and the results evaluated. After empirical evaluations, the 97.58% and 94.97% F-measure scores were obtained when applying U-NET architecture on 256x256 and 512x512 image samples. Simple CNN and U-NET models with Sigmoid and Linear activation function (for common threshold) failed.

Keywords: Deep learning, artificial intelligence, convolutional neural networks, liver segmentation, tomography

TEŞEKKÜR

Bu tez çalışmasında, konusunun belirlenmesinde, tamamlanmasında ve çalışmaya konu olan kaynaklara erişmemde, karşılaştığım problemlerin çözümünde daima bana yol gösterici olan çok değerli tez danışmanım sayın Doç.Dr. Eyyüp GÜLBANDILAR'a teşekkür ederim.

Tez konusu ile ilgili yaptığım çalışmalarda benden yardımlarını esirgemeyen ve daima yanımda olan çok değerli arkadaşım sayın Araş.Gör. Yusuf KARTAL'a teşekkür ederim.

Tez konusu ile ilgili yaptığım çalışmalarda değerli deneyimlerini benim ile paylaşan ve yol göstericiliği ile sonuca ulaşmamda ışık tutan sayın hocam Araş.Gör. Dr. Şahin IŞIK'a teşekkür ederim.

Bu yüksek lisans programına başlamamda, başladıktan sonra tamamlamanda her zaman olduğu gibi tüm desteğiyle yanımda olan, sevgili eşim Semra GÜNAL AVCI'ya teşekkür ederim.

İÇİNDEKİLER

Sayfa

ÖZET vi
SUMMARY vii
TEŞEKKÜR viii
İÇİNDEKİLER ix
ŞEKİLLER DİZİNİ xii
ÇİZELGELER DİZİNİ xiii
SİMGELER VE KISALTMALAR DİZİNİ xiv
1. GİRİŞ VE AMAÇ 1
2. LİTERATÜR ARAŞTIRMASI 3
3. GENEL BİLGİLER 5
3.1. Karaciğer Anatomisi 5
3.1.1. Karaciğer bölütsel anatomisi 5
3.1.2. Karaciğer vasküler anatomisi 7
3.2. Yapay Sinir Ağları 9
3.2.1. İleri beslemeli YSA 12
3.2.2. Geri beslemeli YSA 13
3.2.3. Öğrenme algoritması 13
3.2.4. Aktivasyon fonksiyonu 14
3.2.4.1. Basamak (Step) fonksiyonu 14
3.2.4.2. Doğrusal (Linear) fonksiyon 15
3.2.4.3. Sigmoid fonksiyonu 16
3.2.4.4. Hiperbolik tanjant fonksiyonu 17
3.2.4.5. ReLU (Rectified Linear Unit) fonksiyonu 17
3.2.4.6. Sızıntı (Leaky) ReLU fonksiyonu 18
3.3. Evrişimli Sinir Ağları (ESA) 19
3.3.1. ESA yapısı 19

İÇİNDEKİLER (devam)

Sayfa

3.3.2. Convolutional layer 20
3.3.3. Padding 21
3.3.4. Non-Linearity layer 22
3.3.5. Pooling layer 22
3.3.6. Flattening layer 23
3.3.7. Fully-Connected layer 23
3.4. ESA Türleri 24
3.4.1. LeNet-5 24
3.4.2. AlexNet 24
3.4.3. VGG-16 25
3.4.4. ResNet 26
3.4.5. U-Net 26
4. MATERYAL VE YÖNTEM 28
4.1. Kaynak Veri Seti 28
4.1.1. Veri seti özellikleri 28
4.1.2. Veri Arttırma 30
4.2. Uygulama Ortamı, U-Net Hiper-parametreler 31
4.3. Kullanılan Metrikler 32
4.4. Karşılaştırılan ESA Çeşitleri 33
4.4.1. Basit ESA 33
4.4.2. U-Net 256x256 34
4.4.3. U-Net 512x512 35
4.4.4. U-Net 512x512 (Sigmoid) 37
4.4.5. U-Net 512x512 (Linear) 37

İÇİNDEKİLER (devam)

Sayfa

5. BULGULAR VE TARTIŞMA 38
6. SONUÇ VE ÖNERİLER 43
KAYNAKLAR DİZİNİ 44

ŞEKİLLER DİZİNİ

Şekil **Sayfa**

3.1. Karaciğer'in subsegmental anatomisi 6
3.2. Aksiyal kesit CT görüntülerinde karaciğer segmentlerinin görünümü. 7
3.3. İnsan Sinir Hücresi Yapısı 9
3.4. Sinir Hücresinin Matematiksel Modeli 10
3.5. Tek ve Çok katmanlı Sinir Ağ Yapısı 11
3.6. İleri beslemeli Yapay Sinir Ağı 12
3.7. Geri beslemeli Yapay Sinir Ağı 13
3.8. Basamak Fonksiyonu ve Türevi 15
3.9. Doğrusal Fonksiyon ve Türevi 16
3.10. Sigmoid Fonksiyon Ve Türevi 16
3.11. Hiperbolik Tanjant Fonksiyonu ve Türevi 17
3.12. ReLU Fonksiyonu ve Türevi 18
3.13. Sızıntı ReLU Fonksiyonu ve Türevi 18
3.15. Evrişimli Sinir Ağının Temel gösterimi 20
3.16. Convolution Adımları 21
3.17. ReLu fonksiyonun uygulanmış hali 22
3.18. MaxPooling işlemi 23
3.19. Flatting İşlemi 23
3.20. LeNet-5 CNN gösterimi 24
3.21. AlexNet Gösterimi 25
3.22. VGG-16 Temel Yapısı 25
3.23. Artık değer Bloğu 26
3.24. U-Net Mimarisi 27

ÇİZELGELER DİZİNİ

Çizelge **Sayfa**

3.1. Couinand'a göre karaciğerin segmentleri 7
3.2. Biyolojik Sinir Hücresi ile Yapay Sinir Ağı karşılıkları 10
4.1. Örnek görüntüler ve Etiketler 29
4.2. Veri Arttırma Yöntemleri 30
4.3. Arttırılmış Veri Örneği 30
4.4. U-Net Hiper parametreler 32
5.1. Eğitim süreci elde edilen değerler 38
5.2. Görsel Sonuçlar 39
5.3. Nesnel Sonuçlar 39

SİMGELER VE KISALTMALAR DİZİNİ

Kısaltmalar	Açıklama
CNN	Convolutional Neural Network
CT	Computed Tomography
ESA	Evrişimli Sinir Ağı
HU	Hounsfield Unit
YSA	Yapay Sinir Ağı
NLST	National Lung Screening Trial

1. GİRİŞ VE AMAÇ

Karaciğer; insan vücudunda dolaşan kanın üretilmesinden, temizlenmesinden, sindirilmiş gıdaların işlenmesinden ve buna benzer yaklaşık beş yüz farklı işlevin yerine getirilmesinden sorumlu hayati bir organdır. Karaciğer sağlığının bozulması ile birlikte diğer organlarında buna bağlı olarak işlevlerini yerine getirmesinde ciddi sıkıntılar yaşanmaktadır.

Karaciğer; diğer organlardan farklı olarak toplam hacminin belirli bir oranında kaybedilmesi halinde bile kendini yenileyebilen ve onarabilen bir organ olarak ayrılmaktadır. Bu özelliği bakımından karaciğerin malign (kanser) bozukluklarında hasarlı kısım cerrahi operasyon ile çıkarılıp tedavi sürecine katkı sağlanmaktadır. Bu operasyonun başarısı açısından bazı önemli parametreler bulunmaktadır. Bu parametrelerden özellikle karaciğerin kan akışını sağlayan hepatik arter ile portal ven damarları ve karaciğerin toplam hacmi, operasyon sonrası karaciğerin yaşamsal faaliyetleri sürdürebilmesi açısından çok önemlidir.

Medikal görüntüleme teknikleri bir çok hastalığın tespitinde vazgeçilmez bir araç haline gelmiştir. Bilgisayarlı Tomografi (BT) gibi dokuları birbirinden ayırt etmeye yarayan Hounsfield Unit (HU) değerini kullanan görüntüleme teknikleriyle birçok organ detaylı olarak incelenebilmektedir. Üretilen bu dijital medikal görüntüler çeşitli görüntü işleme ve yapay zekâ teknikleriyle incelenerek birçok yeni yaklaşımın ortaya konulmasında önemli rol oynamaktadır.

Günümüzde karaciğer rahatsızlıklarının tedavisinde gelişmiş yöntemler kullanılsa da, özellikle karaciğerin tümörsel bozukluklarında, tümöral bölgenin uzaklaştırılmasında yapılacak hesaplamalar, tedavinin başarısını, hastanın yaşam süresinin uzatılması gibi sonuçlara direk etki etmektedir.

Bu bağlamda operasyon öncesi ve sonrası kalan hacim, hepatik ve portal damarların uzunluğu gibi parametrelerin hesaplanması çok önemlidir. Günümüzdeki ileri görüntüleme teknikleri (BT) dokuları yoğunluğuna göre ayırabilecek kabiliyettedir (HU ile). Bu cihazların ürettiği görüntülerden 3 boyutlu gerçek modeller çıkarılabilmektedir.

Bu tez çalışmasında BT görüntülerinden karaciğer dokusunun sınırlarının derin öğrenme teknikleriyle belirlenmesi (bölütlenmesi) amacıyla bazı ağ modelleri incelenmiş ve sonuçlar ortaya konulmuştur. Karaciğer dokusunun otomatik olarak bölütlenmesi buradan üretilecek çıktılarla geliştirilecek 3 boyutlu görüntüleyici uygulamalarda kullanılabilecektir. Böylece elde edilen görüntülerden daha hassas hesaplama sonuçların ortaya koyacak modellerin geliştirilmesine imkân verilmiş olacaktır.

2. LİTERATÜR ARAŞTIRMASI

Medikal görüntülerin uygulamaya bağlı olarak işlenmesinde görüntü üzerinde anlam ifade eden bir bölümünün tespit edilmesi tanı aşamasının önemli bir bölümüdür. Literatürde "bölütleme" olarak ifade edilmektedir. Bölütleme, görüntünün sınıflanmasında ve tanımlanmasında temel adımdır. Görüntünün bölütlenmesi sonucunda elde edilen görüntü parçasına "ilgili bölge" denir (Wismüller, 2000).

Bölütleme ile ilgili yapılan çalışmalarda çeşitli yöntemler kullanılmış olmakla beraber bunlardan Ginger (1994) lezyonları sınıflandırmak için çoklu gri seviye eşikleme ve kurala dayalı yaklaşım yöntemini kullanmıştır. Genel olarak yapılan çalışmalar incelendiğinde Yapay Zekâ tekniklerinin kullanımının oldukça yaygın olduğu görülmektedir.

Ahmed ve Farad (1997) BT ve MR görüntüleri için iki ayrı katmandan oluşan yapay sinir ağı temelli bir bölütleme yöntemi uygulamışlardır. Yöntemin birinci adımında özellik çıkarım kullanılmıştır. İkinci adımda ise kendinden organizeli özellik haritalama kullanılmıştır. BT görüntüleri üzerinde çalıştırılan bölütleme yöntemiyle %97.21 doğruluk ile bölütleme başarımı elde edilmiştir.

Saxena, H. (2018) BT görüntüleri üzerindeki çekim masasını görüntüden kaldırmak için yaptığı çalışmada oluşturduğu Evrişimli Sinir Ağı (ESA) sistemiyle; 500 görüntü üzerinde yaptığı testlerde %99 başarı oranı sağlamıştır. Oluşturulan ESA farklı vücut bölgelerini içeren CT görüntülerinde de %99 başarı oranına erişmiştir.

KILIKÇIER, Ç. ve YILMAZ, E. (2018), yaptıkları çalışmada BT görüntülerden akciğerlerin tespiti için süper pikseller ve YSA üzerinde temellendirilmiş bir yöntem önerilmiştir. Önerilen yöntemin başarımı National Lung Screening Trial (NLST) veri tabanında yer alan akciğer görüntüleri üzerinde incelenmiştir. Deneyler sonucunda önerilen yöntemin %92,66 sınıflandırma doğruluğuna sahip olduğu görülmüştür.

İnik, Ö. ve Ülker, E. (2017) yaptıkları çalışmada; derin öğrenme tekniklerinin temelini oluşturan ESA mimarisi ve bu mimaride kullanılmış olan katmanlar ile ilgili

bilgiler vermiştir. Bu katmanların alt yapısında gerçekleştirilen işlemler açıklanarak konvolüsyon, ReLu ve havuz katmanlarında uygulanan işlemlerin çıktı görüntülerini ortaya koymuşlardır. Her katmanın modelin oluşturulmasına olan katısı ifade edildikten sonra, ESA modelinin eğitim süreci basamaklar halinde gösterilmiştir. Derin öğrenmenin tekrar popüler olmasına sebep olan AlexNet ve sonrasında temel alınan birkaç ESA modellerinin yapıları anlatılmıştır. Derin Öğrenme ile ilgili yapılan çalışmalar hakkında bilgiler sunulmuştur.

3. GENEL BİLGİLER

3.1. Karaciğer Anatomisi

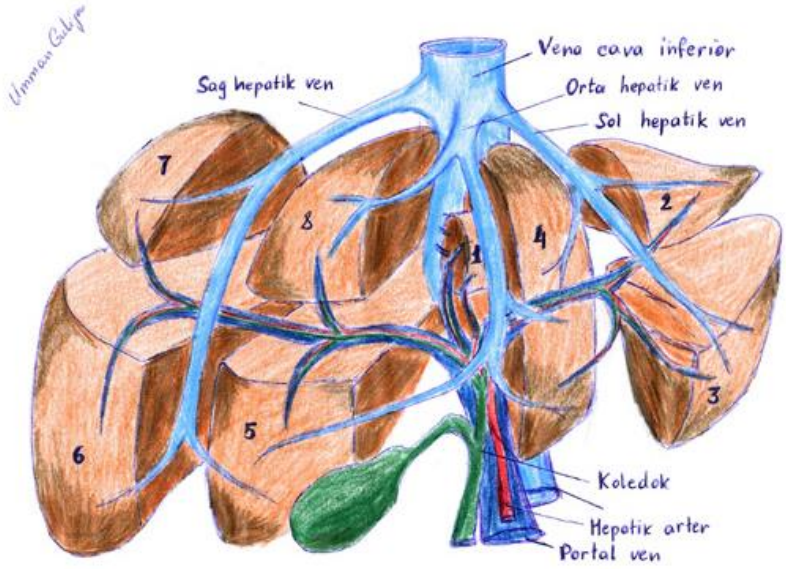
Vücudumuzun en büyük organlarından biri olan karaciğer, karın bölgesinin sağ üst kısmında diyaframın hemen altında bulunmaktadır. Karaciğer çift kan akımına sahip parankimatöz tek organdır. Portal ven karaciğere taşınan kanın yaklaşık %75'ini sağlarken, hepatik ven ise yaklaşık %25'ini sağlamaktadır.

Karaciğerin şekilsel ve fonksiyonel anatomisi aşağıda bahsedileceği gibi çeşitli farklılıklar göstermektedir. Morfolojik açıdan karaciğer, falsiform ligament ile sağ ve sol olmak üzere iki loba ayrılır. Riedel lobu ve sol karaciğer lobunun diyaframın sol altına doğru uzanması en çok görülen varyasyonlardandır. Sağ karaciğer lobundan başlayan Riedel lobu genellikle inferiora doğru uzanmaktadır.

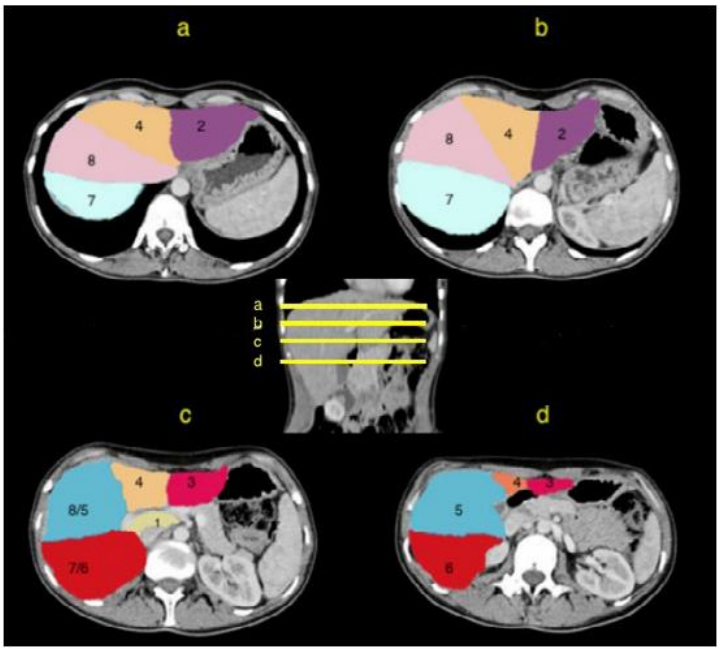
3.1.1. Karaciğer bölütsel anatomisi

Başak, M. ve Akan, D., (2015) yayınladıkları makaleye göre; ilk defa fonksiyonel karaciğer anatomisi kavramı Cantlie tarafından 1898 senesinde ortaya atılmıştır. Goldsmith ve Woodburne (1957) hepatik venlerin dağılımına göre karaciğeri üç loba ayırmıştır: Bunlar; orta hepatik ven ile oluşturulmuş sol ve sağ lob ile kaudat lobdur. Ayrıca sağ lob ise kendi içinde hepatik ven tarafından; posterior ve anterior segmentlere, sol lob ise medial ve lateral segment olmak üzere sol hepatik ven tarafından ikiye ayrılır. 1957 yılında Fransız cerrah Couinaud (1957) tarafından ilk kez karaciğerin kendi vasküler ve biliyer drenajına sahip bağımsız fonksiyonel segmentlerden oluştuğu öne sürülmüştür ve karaciğer 8 segmente ayrılmıştır. Couinaud sınıflamasının günümüzde halen önemini korumasının sebebi, belirtilen segmentlerin birbirinden bağımsız olarak rezeke edilebilmesidir. Couinaud'a göre karaciğer segmental anatomisine göre, orta hepatik ven ile sol ve sağ lob olmak üzere ikiye ayrılmaktadır. Kaudat lob ise karaciğerin posteroinferiorunda, sağ ve sol lob arasında ayrı bir lob olarak izlenmekte ve segment 1 olarak adlandırılmaktadır. Goldsmith ve Woodburne tarafından tanımlandığı gibi, sağ lob sağ hepatik ven tarafından

posterior ve anterior segmentlere, sol lob sol hepatik ven tarafından medial ve lateral segmentlere ayrılmaktadır. Sağ ve sol ana portal venlere paralel çizilen hayali bir transvers hatla ise, kaudat lob ve sol lob medial segment dışındaki segmentler subsegmentlere bölünmektedir. Couinaud sınıflamasına (Çizelge-3.1.) göre karaciğere ventral yüzden bakıldığında; kaudat lob haricindeki segmentler saat yönünde segment 2' den segment 8' e kadar isimlendirilmektedir (Şekil-3.1. ve Şekil-3.2.).



Şekil 3.1. - Karaciğer'in subsegmental anatomisi (Başak M., Akan D., 2015)



Şekil 3.2. - Aksiyal kesit CT görüntülerinde karaciğer segmentlerinin görünümü. (Başak M., Akan D., 2015)

Çizelge 3.1. - Couinand'a göre karaciğerin segmentleri (Başak M., Akan D.'den 2015)

Couinaud(1957)	**Fonksiyonel anatomi**
I	Kaudat lob
II	Sol lateral segment superior
III	Sol lateral segment inferior
IV	Sol medial segment
V	Sağ anterior segment inferior
VI	Sağ posterior segment inferior
VII	Sağ posterior segment superior
VIII	Sağ anterior segment superior

3.1.2. Karaciğer vasküler anatomisi

Karaciğerin beslenmesi portal ven ve hepatik arter tarafından sağlanırken, venöz drenajı hepatik venler tarafından gerçekleştirilir. Karaciğer parankiminin portal ven ve hepatik arter tarafından sağlanan dual kanlanma özelliği, karaciğer enfarktlarının oldukça nadir görülmesine neden olur. Günümüzde karaciğer vaskülarizasyonunun preoperatif radyolojik değerlendirmesine en sık ihtiyaç duyulan alan, metastazlar nedeniyle yapılan

hepatik tümör rezeksiyonudur (metastazektomi). Karaciğer metastazlarının büyük kısmı, kolorektal karsinom kökenlidir.

Multidedektör bilgisayarlı tomografi ve manyetik görüntüleme teknolojisinde kaydedilen gelişmeler sayesinde, hepatik tümör rezeksiyonu öncesinde yapılan radyolojik değerlendirmenin, nonterapötik laparoskopi oranını önemli ölçüde azalttığı gösterilmiştir. Ayrıca tümör rezeksiyonu sonrası geride bırakılacak sağlam karaciğer dokusu hacmi ve rezeksiyon sonrasında yeterli vaskülarizasyon ve biliyer drenajın sağlanabileceğinden de preoperatif dönemde emin olunmalıdır.

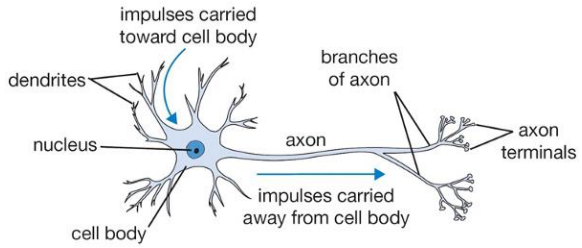
Karaciğerin vasküler anatomisinin preoperatif radyolojik değerlendirmesinin vazgeçilmez olduğu diğer bir konu, karaciğer transplantasyonudur. Özellikle verici karaciğerinde seçilecek olan hemihepatektomi planında, orta hepatik venin yaklaşık 1 cm sağından geçen ve safra kesesi yatağı ile inferior vena kava doğrultusunda uzanan, nispeten avasküler karakterdeki hattın (Cantlie hattı) tercih edilmesi büyük önem taşır.

Girişimsel radyolojide elde edilen gelişmeler neticesinde, karaciğerin vasküler özellikle arteryel anatomisinin detaylı bir şekilde değerlendirilmesinin öneminin arttığı alanlardan biri de, hepatik intraarteryel kemoterapi ve transarteryel kemoembolizasyon-radyoembolizasyon uygulamalarıdır. Bu tip uygulamalar öncesinde; kateteri, uygulanacak ajanın ekstrahepatik dokuya reflüsünü engelleyecek ve aynı zamanda karaciğer parankimine homojen dağılacak şekilde yerleştirebilmek çok önemlidir. Bu da ancak preoperatif değerlendirme ve uygun hasta seçimi sayesinde mümkündür.

Yine günümüzde girişimsel radyoloji klinikleri tarafından ultrasonografi veya bilgisayarlı tomografi eşliğinde uygulanan, radyofrekans ablasyon tedavisi veya mikrodalga ablasyon tedavileri öncesinde de karaciğer anatomisi titizlikle değerlendirilmelidir. Majör safra yolu veya damar invazyonu varlığında, karaciğer hilusuna veya çevre organlara yakın komşulukta bulunan tümörlerde ve mevcut karaciğer hacminin %40'ından fazla yer kaplayan tümörlerde ablasyon tedavisi uygulamaktan kaçınılmalıdır. Perkütanöz yolla ulaşımın riskli olacağı tümörlerde ise, ablasyon tedavilerinin açık cerrahi ile uygulanması tercih edilmelidir.

3.2. Yapay Sinir Ağları

Sibernetik; canlıların davranışlarını inceleyen ve bu davranışların matematiksel modelini çıkarıp, bu yapıya uygun yapay modellerin üretilmesidir. Eğitilebilir, dinamik ve kendi kendine öğrenebilen ve karar verebilen yapay sinir ağları ile insan beyninin çalışma yapısı ve öğrenme mekanizması modellenmeye çalışılmaktadır. İnsandaki gibi yapay sinir ağlarıyla; makinelerin eğitilmesi, öğrenmesi ve karar vermesi amaçlanmaktadır.



Şekil 3.3. - İnsan Sinir Hücresi Yapısı (Kızrak, A., 2018)

İnsandaki bir sinir hücresinin (nöron) yapısı Şekil 3.3. gibidir:

- Akson (Axon): elektriksel çıkış sinyallerinin üretildiği aktif gövdedir, elektrik sinyallerinin iletimi tek yönlüdür. Bu kısım sistemin çıkışını ifade etmektedir.
- Dentritler (Dendrites): İlişkili hücrelerden gelen elektriksel sinyalleri alan pasif uzantılardır. Sistemin elektriksel girişini ifade etmektedir.
- Sinaps (Synapse): Aksonlar ile dentritler arasındaki bağlantıları sağlar.
- Miyelin Tabaka (Myelin Sheath): Elektriksel iletim hızını kontrol eden yalıtım maddesidir.
- Çekirdek (Nucleus): Aksonlar boyunca elektriksel sinyallerin periyodik olarak tekrardan üretilmesini sağlar.

Aksonda iletilen elektriksel sinyal sinapslara kimyasal taşıyıcılar ile taşınır. Stoplazma -85mV ile polarizedir. -40mV (Na+ içeri): uyarma (+) akımı oluşturur. -90mV (K+ dışarı): bastırma (-) akımı oluşturur. Belirli eşik gerilim değerinin üstünde hücre uyarılır, diğer durumlarda hücre bastırılır. Sinirsel hesaplama Σ bu durumla ilişkili sinyali üretilmesidir.

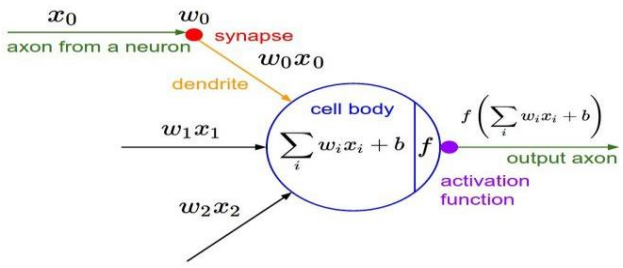
Sinir hücresine diğer sinir hücrelerinden gelen uyarımlar, dentritler aracılığıyla hücre gövdesine taşınır ve hücre içi aktivasyonun/kararlılık halinin bozulmasıyla oluşan bir

kimyasal süreç içerisinde diğer hücrelere aksonlarla iletilir. Uyarımların diğer sinir hücrelerine taşınabilmesinde akson uçları ile dentritler arasındaki sinaptik boşluklar (sinaps) rol oynar. Sinaptik boşluk içinde yer alan "sinaptik kesecikler", gelen uyarımların diğer hücrelere dendritler aracılığıyla geçmesini koşullayan elemanlardır. Sinaptik boşluğa, "sinaptik kesecikler" tarafından sağlanan nöro-iletken maddenin dolması uyarımların diğer hücrelere geçişini koşullar. Hücrelere gelen uyarımlarla uyumlu olarak hücreler arasındaki mevcut sinaptik ilişkilerin değişimi veya hücreler arasında yeni sinaptik ilişkilerin kurulması "öğrenme" sürecine karşılık gelir. Çizelge 3.2.'de karşılıklı olarak verilmiştir.

Çizelge 3.2. - Biyolojik Sinir Hücresi ile Yapay Sinir Ağı karşılıkları (Şengül N.'den 2017)

Biyolojik Sinir Hücresi	**Yapay Sinir Ağı**
Nöron	İşlemci eleman
Dentrit	Toplama fonksiyonu
Hücre gövdesi	Transfer fonksiyonu
Aksonlar	Yapay nöron çıkışı
Sinapslar	Ağırlıklar
Nöron	İşlemci eleman

İnsan sinir hücresinin matematiksel olarak ifade edildiği model ise Şekil-3.4. gibi gösterilebilir.



Şekil 3.4. - Sinir Hücresinin Matematiksel Modeli (Kızrak, A., 2018)

Perseptron (Perceptron): Bir Yapay Sinir Ağında (YSA) bulunan perseptron, (3.1) denkleminde ki gibi lineer bir fonksiyon olarak tanımlanmaktadır ve 1957 yılında Frank Rosenblatt tarafından ilk kez tanımlanmıştır.

$$\mathbf{y} = W\ x\ X + b \qquad (3.1)$$

y: Yapay sinir hücresinin çıktısını ifade eder, X giriş değerlerine bağlıdır ve bağımlı bir değişkendir.

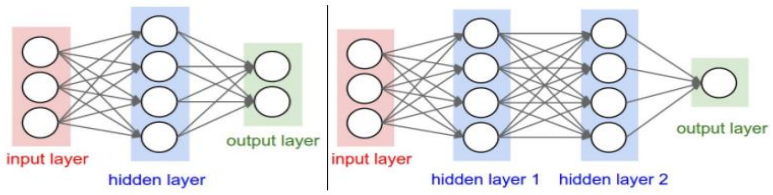
X: Dış dünyadan gelen değerlerdir, ağın öğrenmesi istenilen örneklere göre belirlenir.

W: Hücreye gelen verinin önem ve hücre üzerindeki etkisini ifade eder. Ağırlıkların yüksek/düşük olması önem derecesini ifade etmez. Herhangi bir ağırlık değerinin sıfır olması ağ için en belirleyici özellik olabilir.

b: Bias değeri, aktivasyon fonksiyonunun sağa/sola kaydırılmasını (shift) gerçekleştirir.

Giriş verilerinin toplam değeri sıfır olursa öğrenme gerçekleşmez. Sinir hücresinin çıktı değerleri hep 1 olan bias nöronları, nöronların giriş değerlerinin sıfır olmamasını sağlar. Öğrenme sürecinin hızlandırır ve lokal optimum değerde sabitlenmeyi önler. Bias değeri bunun yanında, nöronun tepki eşiğini belirler.

Şekil-3.5.'de gösterilen çok katmanlı bir yapay sinir ağında veya Derin Öğrenme modelinde ana hedef; modelin en iyi skorları üreteceği W ve b parametrelerinin hesaplamaktır.

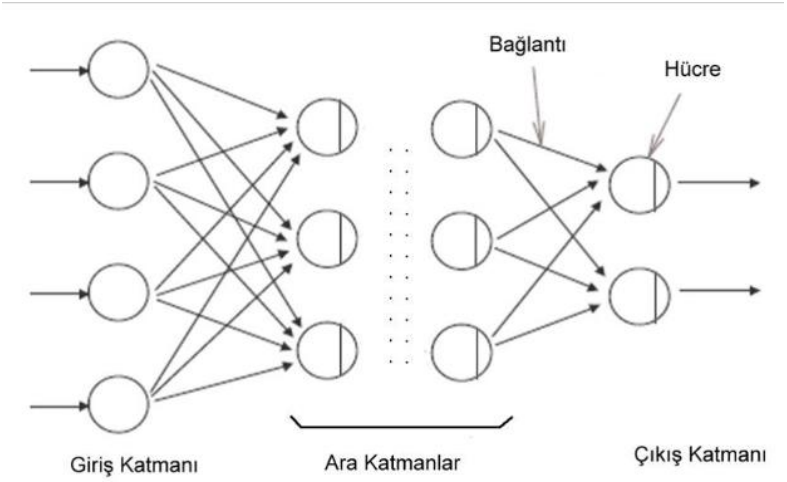


Şekil 3.5. - Tek ve Çok katmanlı Sinir Ağ Yapısı (Kızrak, A., 2018)

- Soldaki tek katmanlı modelde; gizli katmanda 4 ve çıkış katmanında 2 olmak üzere toplam 6 nöron bulunmaktadır (giriş katmanı haricinde), gizli katmanda 3x4 ve çıkış katmanında 4x2 (3x4 + 4x2 = 20) olmak üzere toplam 20 ağırlık ve gizli katmanda 4 ve çıkış katmanında 2 olmak üzere toplam 6 bias değeri ile birlikte toplamda tüm ağ için 26 adet öğrenilmesi gereken parametre vardır.
- Sağdaki iki gizli katmanlı modelde ise birinic gizli katmanda 4, ikinci gizli katmanda 4 ve çıkış katmanında 1 olmak üzere toplamda 9 nöron, birinci gizli

katmanda 3x4, ikinci gizli katmanda 4x4 ve çıkış katmanında 4x1 olmak üzere 32 ağırlık ve birinici gizli katmanda 4, ikinci gizli katmanda 4 ve çıkış katmanında 1 olmak üzere toplam 9 bias değeri ile birlikte toplamda tüm ağ için 41 adet öğrenilmesi gereken parametre vardır.

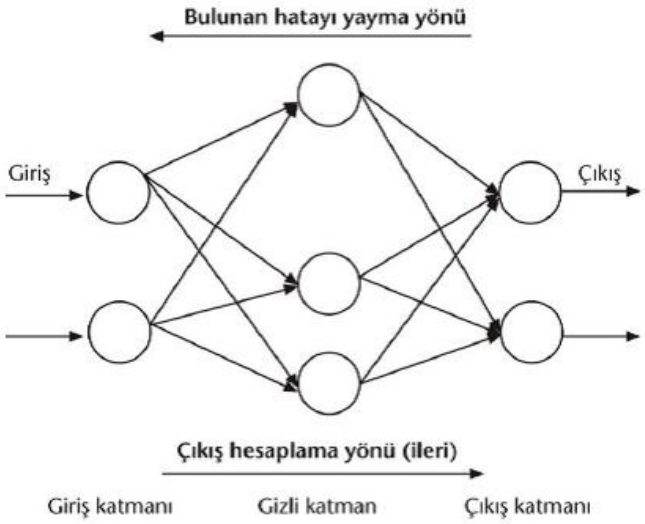
3.2.1. İleri beslemeli YSA



Şekil 3.6. - İleri beslemeli Yapay Sinir Ağı (Şengül N., 2017)

Şekil 3.6. da gösterilen İleri beslemeli YSA da bilgi akışı sadece tek bir yöne doğrudur. Girdi katmanından alınan bilgiler içerideki gizli katmana gönderilir. Gizli ve çıktı katmanlarında girdi bilgileri işlenerek çıkış değeri üretilir.

3.2.2. Geri beslemeli YSA



Şekil 3.7. - Geri beslemeli Yapay Sinir Ağı (Şengül N., 2017)

Geri beslemeli YSA, çıkış katmanındaki ve ara katmanlardaki çıkış değerlerinin, giriş katmanına veya bir önceki ara katmana doğru geri beslendiği Şekil-3.7.'de ki gibi bir mimariye sahiptir. Bu şekilde, giriş değerleri ileri yönde aktarıldığı gibi geriye doğruda aktarılmış olur. Bu tip YSA'lar dinamik bir hafızaya sahiptir ve herhangi bir ***t*** anındaki çıkış değeri hem o andaki hem de önceki giriş değerlerini yansıtır. Bu sebeple, özellikle ön tahmin uygulamaları için başarılıdırlar. Geri beslemeli ağlar bazı zaman-serilerinin tahmin edilmesinde yüksek başarı göstermişlerdir . Örnek olarak Hopfield, Elman, SOM (Self Organizing Map) ve Jordan ağları verilebilir.

3.2.3. Öğrenme algoritması

Bir veri setinden öğrenebilme kabiliyeti YSA'nın en önemli özelliğidir. Yapay sinir ağlarında öğrenilen bilgi, ağ içindeki sinirler arasındaki bağlantılardaki ağırlıklarda tutulmaktadır. Bu sebeple ağırlıkların nasıl tespit edileceği önem kazanmaktadır. Bilgiyi ifade eden değerler ağın tamamına yayılmış bir şekilde saklandığı için sadece bir düğümün tutuğu ağırlık değeri tek başına bir anlam içermemektedir. Ağın tamamındaki ağırlıkların optimal değer alması gereklidir. Bu ağırlıkları elde etmek için yapılan işleme ise "ağın eğitilmesi" denir. Bu kurama göre ağın eğitilebilir bir ağ olabilmesinin şartı; ağırlık değerlerinin bir kurala göre dinamik bir şekilde değiştirilebilir olmasıdır.

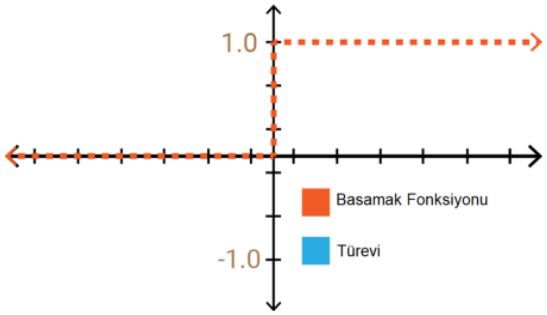
Basit bir ifade ile; “Eğitim - öğrenme işlemi, bu ağırlıkların en optimal değerlerinin bulunması” olarak tanımlanabilir. YSA’da ağırlıklar (filtreler), rastgele, Gauss dağılımlı veya düzgün dağılımlı olarak seçilebilir. YSA’da batch sayısı aynı anda network modeline kaç tane örnek gönderildiği anlamına gelmektedir. Batch sayısının 1 olması seçilen modele örnekler sırayla tek tek gönderilmektedir. YSA ‘da 1 Epoch durumunda tüm eğitim verisi bir kez networkün ileri ve geri beslemesinden geçirilmektedir. Örneğin elimizde 100 tane eğitim verisi var olduğunu varsayalım. Epoch = 1 ve Batch = 1 durumunda seçilen model 100 iterasyon ile çalışacaktır. Epoch = 1 ve Batch = 5 durumunda seçilen model 100/5=20 iterasyon sürecektir. Epoch = 20 ve Batch = 5 durumunda seçilen model (20*100)/5=400 iterasyon sürecektir.

3.2.4. Aktivasyon fonksiyonu

Bu fonksiyon hücreye gelen net girdiyi işleyerek hücrenin bu girdiye karşılık üreteceği çıktıyı belirler. Aktivasyon fonksiyonu genellikle doğrusal olmayan bir fonksiyon seçilir. Yapay sinir ağlarının bir özelliği olan “doğrusal olmama” aktivasyon fonksiyonlarının doğrusal olmama özelliğinden gelmektedir. Aktivasyon fonksiyonu seçilirken dikkat edilmesi gereken bir diğer nokta ise fonksiyonun türevinin kolay hesaplanabilir olmasıdır. Geri beslemeli ağlarda aktivasyon fonksiyonunun türevi de kullanıldığı için hesaplamanın yavaşlamaması için türevi kolay hesaplanır bir fonksiyon seçilir. Günümüzde en yaygın olarak kullanılan aktivasyon fonksiyonu “Sigmoid fonksiyonudur.

3.2.4.1. Basamak (Step) fonksiyonu

İkili değer alan bir fonksiyondur ve tabiatı gereği ikili sınıflayıcı olarak kullanılır. Bu yüzden genellikle çıkış katmanlarında tercih edilir. Gizli katmanlarda türevi öğrenme değeri temsil etmediği için kullanılması tavsiye edilmez ve zaten karşınıza da çıkmayacaktır (Şekil-3.8.). Basamak fonksiyonunu denklemi (3.2) de verilmiştir.



Şekil 3.8. - Basamak Fonksiyonu ve Türevi (Kızrak, A., 2019)

$$\boldsymbol{f(x) = x} \tag{3.2}$$

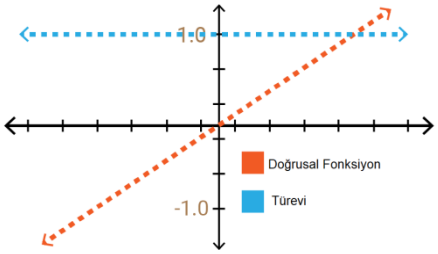
3.2.4.2. Doğrusal (Linear) fonksiyon

Bir dizi aktivasyon değeri üretir ve bunlar basamak fonksiyonundaki gibi ikili değerler değildir. Kesinlikle bir kaç nöronu (sinir hücresi) birbirine bağlamaya izin verir. Bu fonksiyonda ki en önemli sorun; türevinin sabit olmasıdır. Türev hep sabit bir değer çıktığından öğrenme fonksiyonu gerçekleşmez. Çünkü geriye yayılım algoritması türev alan bir sistemden oluşmaktadır.

$$\boldsymbol{A = c.x} \tag{3.3}$$

x'e göre türevi alındığında sabit bir c değerini alır.. Böylece çıkış fonksiyonun x ile bir ilişkisinin kalmadığı anlamına gelir.

Başka bir sorunda; tüm katmanlarda doğrusal fonksiyon kullanıldığında giriş katmanı ile çıkış katmanı arasında hep aynı doğrusal sonuca ulaşılır. Doğrusal fonksiyonların doğrusal bir şekilde birleşimi yine bir başka doğrusal fonksiyondur. Bu en başta birbirine bağlayabiliriz dediğimiz nöronların yani ara katmanların işlevsiz kaldığı anlamına gelir (Şekil 3.9.). Doğrusal fonksiyon denklemi (3.4) de verildiği gibidir.

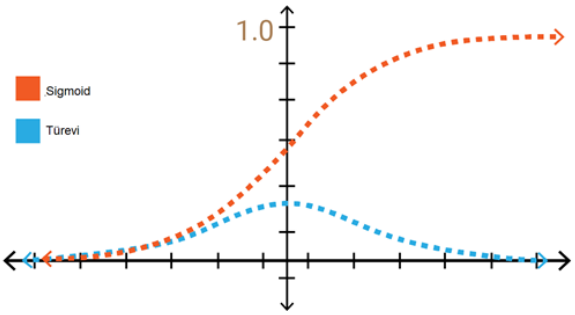


Şekil 3.9. - Doğrusal Fonksiyon ve Türevi (Kızrak, A., 2019)

$$f(x) = \begin{cases} 0 \text{ için } x < 0 \\ 1 \text{ için } x \geq 0 \end{cases} \quad (3.4)$$

3.2.4.3. Sigmoid fonksiyonu

Doğadaki çoğu problem doğrusal değildir ve sigmoid fonksiyonunun kombinasyonları da doğrusal değildir. Bu sebeple katmanlar arası bilgi transferi için sigmoid fonksiyon uygun bir seçim olarak görünmektedir.



Şekil 3.10. - Sigmoid Fonksiyon Ve Türevi (Kızrak, A., 2019)

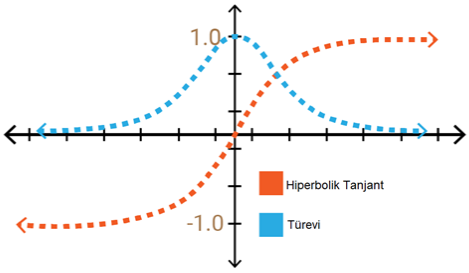
Şekil-3.10.'da ki grafiği incelersek x, -1 ile +1 arasında iken y değerleri hızlı şekilde değişir. x'te yapılan küçük değişimler y'de büyük olacaktır. Bu iyi bir sınıflayıcı olarak ullanılabileceği anlamına gelir. Bu fonksiyonun bir diğer avantajı da doğrusal fonksiyonda olduğu gibi (-sonsuz, +sonsuz) ile karşılaşıldığında her zaman (0,1) aralığında değer üretir. Yani aktivasyon değeri çok uç noktalara gitmez.

Fonksiyonun uçlarına doğru grafiğe dikkatlice bakarsak, y değerleri x'teki değişikliklere çok az tepki vermektedir. Bu bölgelerde türev değerleri çok küçük olur ve 0'a yakınsar. Buna gradyanların ölmesi/kaybolması (vanishing gradient) denir ve öğrenme

olayı minimum düzeyde gerçekleşir. Sıfır olursa gerçekleşmez. Yavaş bir öğrenme olayı gerçekleştiğinde hatayı minimize eden optimizasyon algoritması yerel (lokal) minimum değerlere takılabilir ve yapay sinir ağı modelinden alınabilecek maksimum performans alınamaz. Denklemi ise aşağıdaki gibidir.

$$\boldsymbol{f(x) = \partial(x) = \frac{1}{1+e^{-x}}} \tag{3.5}$$

3.2.4.4. Hiperbolik tanjant fonksiyonu



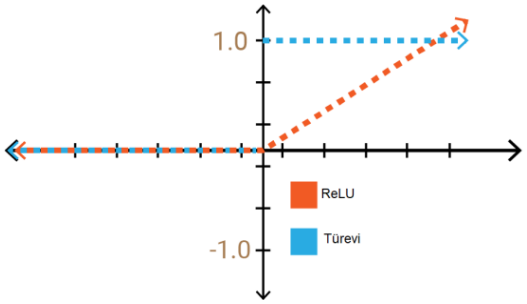
Şekil 3.11. - Hiperbolik Tanjant Fonksiyonu ve Türevi (Kızrak, A., 2019)

Sigmoid fonksiyonuna çok benzer bir yapıya (Şekil-3.11.) sahiptir. Ancak fonksiyonun aralığı bu kez (-1,+1) olarak tanımlanmaktadır. Sigmoid fonksiyonuna göre avantajı ise türevinin daha dik olması yani daha çok değer alabilmesidir. Bu daha hızlı öğrenme ve sınıflama işlemi için daha geniş aralığa sahip olmasından dolayı daha verimli olacağı anlamına gelmektedir. Ama yine fonksiyonun uçlarında gradyanların ölmesi problemi devam etmektedir. Denklemi aşağıdaki gibidir.

$$\boldsymbol{f(x) = \tanh(x) = \frac{(e^{x} - e^{-x})}{(e^{x} + e^{-x})}} \tag{3.6}$$

3.2.4.5. ReLU (Rectified Linear Unit) fonksiyonu

Şekil-3.12. ye ilk bakışta pozitif eksende doğrusal fonksiyon ile aynı özelliklere sahip gibi görünecektir. Ama her şeyden önce ReLU doğada doğrusal değildir. Aslına bakılırsa iyi bir tahmin edicidir. ReLU'nun kombinasyonları ile herhangi başka bir fonksiyona da yakınsamak mümkündür.

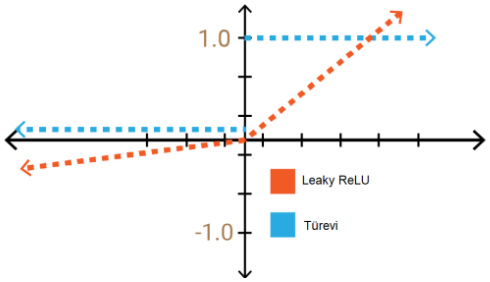


Şekil 3.12. - ReLU Fonksiyonu ve Türevi (Kızrak, A., 2019)

ReLU [0, +∞) aralığında değer almaktadır. Çok fazla nöronlu büyük bir sinir ağında; Sigmoid ve Hiperbolik Tanjant neredeyse tüm nöronların aynı şekilde ateşlenmesine/aktive olmasına sebep olmaktadır. Bu aktivasyon yoğun yani çok işlem gerektiriyor demektir. Ağdaki bazı nöronların aktif olup, aktivasyon seyrek yani verimli bir hesaplama yükü olsun istenmektedir. ReLU ile bu sağlamış olur. Negatif eksende 0 değerlerini alması ağın daha hızlı çalışacağı anlamına da gelmektedir. Hesaplama yükünün sigmoid ve hiperbolik tanjant fonksiyonlarına göre az olması çok katmanlı ağlarda daha çok tercih edilmesine sebep olmuştur. ReLU fonksiyonunda da küçük bir sorun bulunmaktadır. İşlem hızı kazandıran bu sıfır değer bölgesinin türevinin de sıfır olması bir problemdir. Yani öğrenme o bölgede gerçekleşmemektedir. Denklemi ise aşağıdaki gibidir.

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

3.2.4.6. Sızıntı (Leaky) ReLU fonksiyonu



Şekil 3.13. - Sızıntı ReLU Fonksiyonu ve Türevi (Kızrak, A., 2019)

Bu Fonksiyonda negatif düzlemde bir sızıntı değeri bulunmaktadır (Şekil-3.13.). Bu sızıntı değeri 0,01 olarak verilir eğer sıfıra yakın farklı bir değer verilirse fonksiyonun adı rastgele Leaky ReLU olarak olarak değişmektedir. Sızdırılan ReLU'nun tanım aralığı eksi

sonsuza doğru devam etmektedir. Bu 0'a yakın ama 0 olmayan değer sayesinde ReLU'daki ölen gradyanları yaşatılmış yani öğrenme negatif bölgedeki değerler için de sağlanmış olur. Denklemi ise aşağıdaki gibidir.

$$f(x) = \begin{cases} 0,01 \text{ için } x < 0 \\ x \text{ için } x \geq 0 \end{cases} \quad (3.8)$$

3.3. Evrişimli Sinir Ağları (ESA)

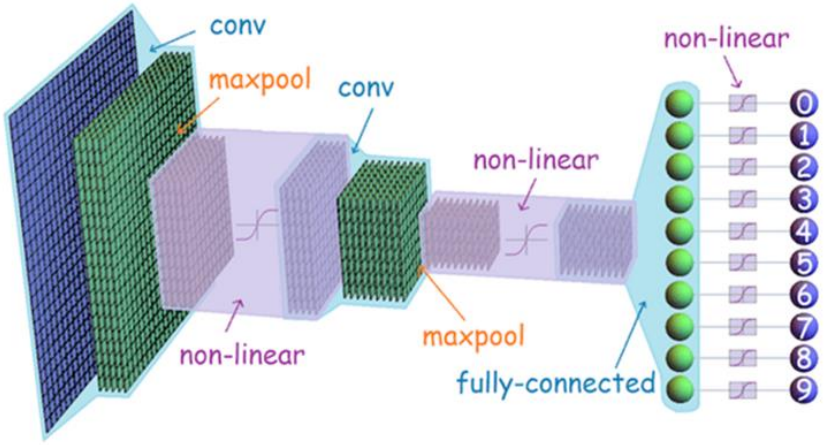
Derin öğrenme alanında en yaygın olarak kullanılan yaklaşım ESA (CNN) dır. Günümüzde onlarca parametreyi hesaplayabilmek için eldeki veriden çeşitli özelliklerin çıkarılması gerektiği ortaya koyulmuştur. Obje bulma (Object Detection), Obje takibi (Object Tracking), Görüntü Sınıflandırma (Image Classification), Doğal Dil İşleme (Natural Language Processing) gibi problemleri çözmek için gerekli sistemler yapay sinir ağları temelinde kurulmuştur.

Evrişimli Sinir Ağları, görüntüyü veya objeyi ayırt etmek için hedef nesneyi benzersiz olarak ifade eden özellikleri kullanmaktadır. İnsan beynininde temel çalışma mekanizması bu şekilde işlemektedir. İnsan beyni de örneğin bir arabayı tanımlarken, tekerlekleri, plakayı ve diğer araçlara has özellikleri tespit eder ve daha soyut özelliklere kadar detayları tespit etmeye çalışır.

3.3.1. ESA yapısı

Evrişimli sinir ağları görüntüyü işleyebilmek için birden fazla katmanın bir araya getirilmesiyle çalışır. Şekil 3.14.'de görüldüğü üzere, genel olarak bu katmanlar şu şekilde sıralanabilir:

- Convolutional Layer : Özelliklerin çıkarımı yapılır
- Non-Linearity Layer : Doğrusal olmayanlığın sisteme tanıtılması
- Pooling (Downsampling) Layer : Ağırılık sayısının azaltılması ve uygunluğun kontrol edilmesi
- Flattening Layer : Temel Sisnir Ağı için verilerin hazırlanmasını gerçekleştirir.
- Fully-Connected Layer : Temel Sinir Ağı



Şekil 3.14. - Evrişimli Sinir Ağının Temel gösterimi (Çarkacı, N., 2018)

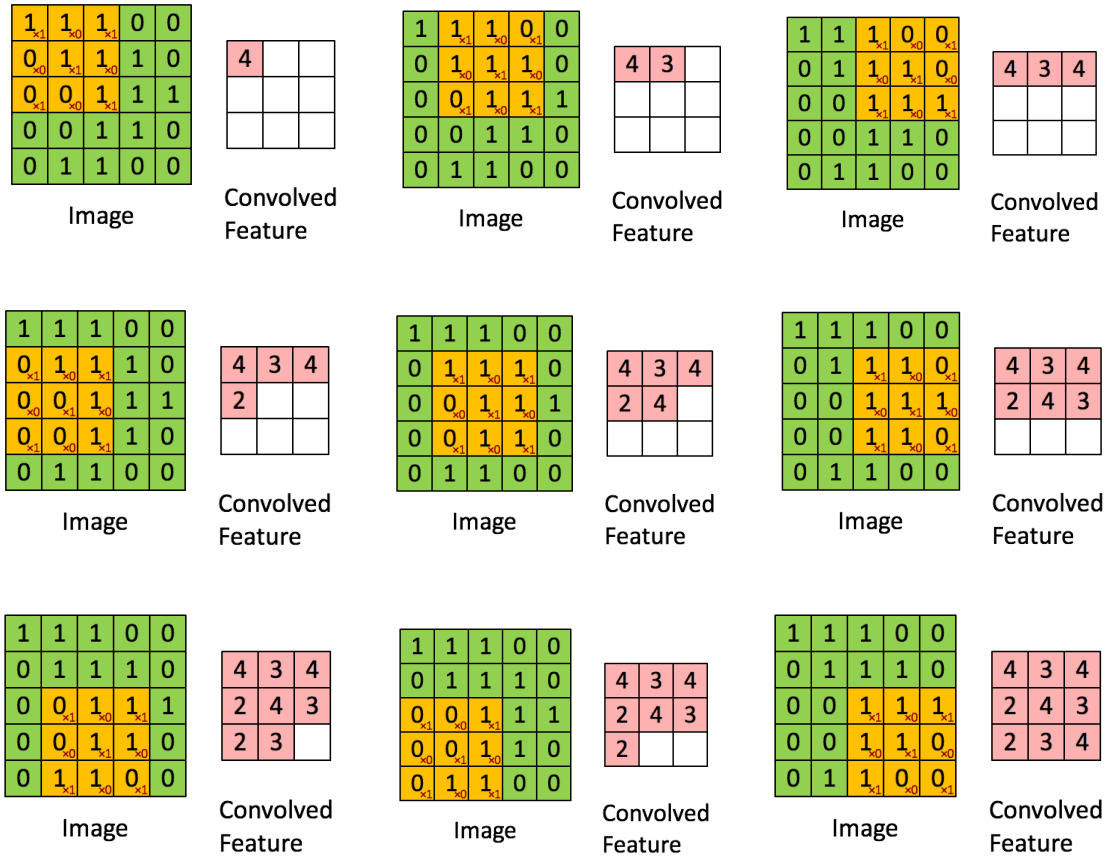
3.3.2. Convolutional layer

Görüntünün özelliklerinin çıkartıldığı bu katman CNN'in temel katmanıdır. Görüntünün düşük ve yüksek seviyeli özelliklerinin çıkartılması için çeşitli ön tanımlı filtreleri uygular. Bu filtreler kenar çıkarımı yapacak bir filtre olabilir. Filtreler çok boyutlu ve piksel katsayıları içerirler.

Filtrenin uygulanması sırasında, seçilen filtre görüntü piksel matrisinin sol üst köşesine konumlandırılır. Görüntü pixel matrisi ile uygulanacak filtre matrisi karşılıklı olarak çarpılır ve sonucun toplamı çıktı matrisine yazılır. Filtre bir adım sağa kaydırılarak tekrar işlem yapılır. İlk satır için işlem tamamlandıktan sonra sonraki satıra geçilir. Tüm görüntüye filtre tamamen uygulanıncaya kadar döngü tekrarlanır. W x H boyutunda bir görüntüye M x N boyutunda bir filtre uygulandığında çıktı matrsinin boyutu (W-M+1) x (H-N+1) boyutunda bir matris olur. Bu matris "özellik haritası" olarak da adlandırılır. Örnek olarak Filtre işleminin adımlarını göstermek gerekirse. Filtre matrisi olarak:

$$A = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

Seçilmiş olsun. 5x5 bir görüntü üzerine bu filtrenin uygulanması sırasıyla Şekil-3.15. gibi olmaktadır.



Şekil 3.15. - Convolution Adımları (Bayramlı B., 2015)

3.3.3. Padding

Bu katmanda görüntünün özellikleri çıkarılırken uygulanan filtrenin boyutuna göre çıktı matrisi ana görüntüye göre küçülmektedir. Bu görüntünün sonraki aşamalarda kullanılacak bazı özelliklerin kaybolmasına sebep olabilmektedir. Bunun önüne geçmek için oluşturulan "Özellik Haritası" matrisine 0 değerli satır ve sutunlar eklenerek görüntünün boyutunun korunması sağlanır.

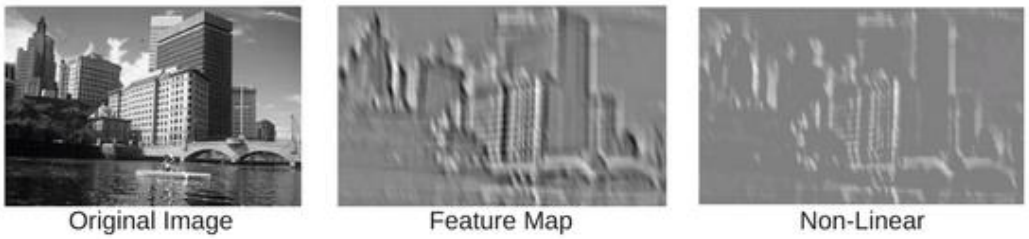
3.3.4. Non-Linearity layer

Convolutional katman(lar) dan sonra genel olarak Non-Linearity (doğrusal olmayan) katmanı gelir. Görüntüdeki doğrusallık bir problem oluşturabilir. Buradaki temel problem bu doğrusallığın; ağın tamamının bir perception gibi davranmasına sebep olabilmesidir. Çıktı ; girdinin lineer bir foksiyonu gibi hesaplanabilir.

Bu katmanın diğer bir adıda aktivasyon katmanıdır. Bu aşamada aktivasyon fonksiyonlarından biri kullanılır. Önceki çalışmalarda Sigmoid, Tahn vs gibi non-lineer fonksiyonlar kullanılmıştır. Eğitim sürecinin zamanını kısaltması bakımından en iyi sonucu Rectifier (ReLu) fonksiyonu (3.9) vermektedir ve günümüzde sıklıkla kullanılmaya başlamıştır.

ReLu Fonksiyonu f(x) = max (0,x) (3.9)

ReLu fonksiyonu Özellik Haritasına uygulandığında şu Şekil 3.16.'da gösterildiği gibi bir sonuç ortaya çıkmaktadır. Özellik haritasındaki siyah değerler ReLu fonksiyonu ile 0 ile değiştirilir.

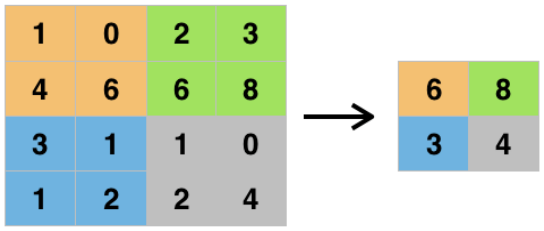


Şekil 3.16. - ReLu fonksiyonun uygulanmış hali (Bayramlı B., 2015)

3.3.5. Pooling layer

Bu katman, CNN de birbirini takip eden Convolutional katmanların arasına eklenen bir katmandır. Gösterimin kayma boyutunu ve ağdaki parametreleri ve bu parametrelere bağlı hesaplama adımlarını azaltma görevini üstlenir. Ağdaki uyumsuzlukların bir nevi

denetlenmesi görevini üstlenir. Pooling mekanizemasının çeşitli türleri olmakla birlikte sıklıkla kullanılan türü Max Poolling dir. Benzer türde L2-norm ve Avarage Pooling türleride vardır. 4x4 bir görüntü üzerine 2x2 max pooling uygulanmış hali aşağıdaki gibidir (Şekil-3.17.).

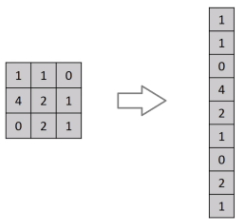


Şekil 3.17. - MaxPooling işlemi (Bayramlı B., 2015)

Bu sayede ağın karar mekanizmasının doğru çalışabilmesi için yeteri kadar bilgi barındıran daha küçük bir örneği alınmış olur.

3.3.6. Flattening layer

Bu katmanda (Şekil-3.18.) temel olarak Fully Connected Layer için gerekli giriş veri setinin hazırlanması işlemi yapılır. Sinir ağları genelde veri setini tek boyutlu dizi olarak almaktadır. Önceki katmanlarda hazırlanmış veri matrisleri bu katmanda tek boyutlu diziye dönüştürülmektedir.



Şekil 3.18. - Flatting İşlemi (Bayramlı B., 2015)

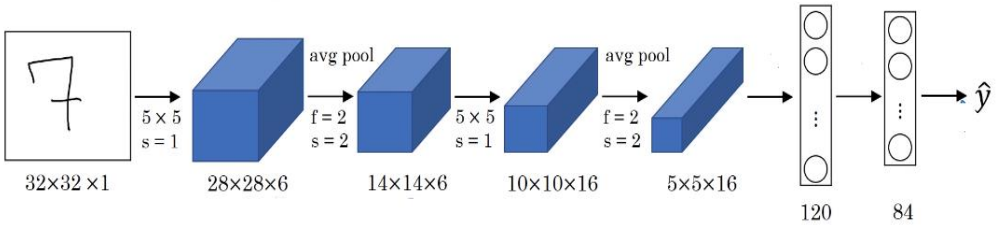
3.3.7. Fully-Connected layer

Klasik Yapay Sinir Ağı katmanlarının bulunduğu katmandır. Bu katmandaki her bir gizli katman kendinden önceki katmanın tüm alanlarına direk bağlıdır. Bu sebeple bu katman tam bağlantılı (Full-Connected) katman olarak adlandırılmıştır.

3.4. ESA Türleri

3.4.1. LeNet-5

Yan LeCun ve arkadaşları tarafından 1998 yılında yayınlanmış ve doğruluk oranı yüksek olup başarılı kabul edilen ilk ESA modelidir (Şekil-3.19.). Posta numaraları, banka çekleri üzerindeki sayıları okumak için geliştirilmiştir. Bu modelde Pooling katmanında max yerine avarage pooling uygulanmıştır. Aktivasyon fonksiyonu olarak ise Sigmoid ve Hiperbolik Tanjant kullanılmıştır.

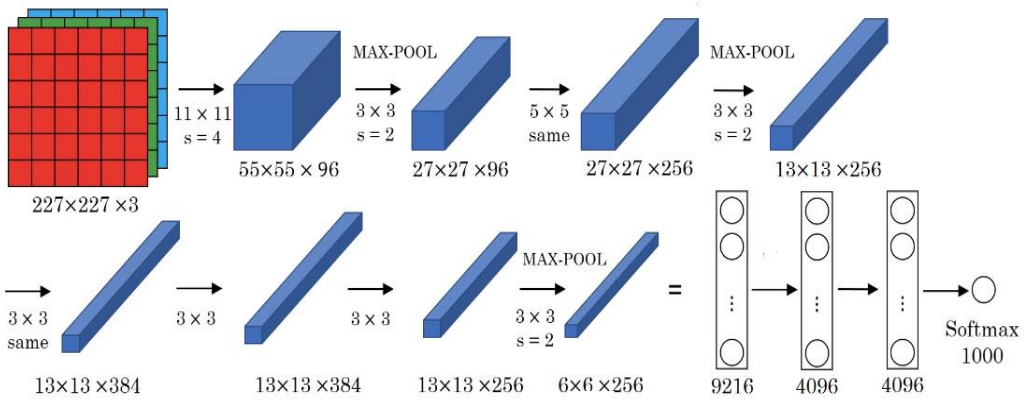


Şekil 3.19. - LeNet-5 CNN gösterimi (Kızrak, A., 2019)

Tam bağlantılı (Fully-Connected) katmanına giriş 5x5x16 = 400 adet ve çıkış katmanında ise 0-9 arasındaki rakamları sınıflandırmak için 10 adet sınıflı softmax bulunmaktadır. Bu modelde hsaplanan toplam parametre sayısı 60 bindir. Ağ boyunca derinlik artarken genişlik düşürülmektedir.

3.4.2. AlexNet

Alex Krizhevsky, Geoffrey Hinton ve Ilya Sutskever tarafından 2012 yılında ortaya konulmuş çalışmadır. LeNet modelindeki gibi birbiri ardına sıralanmış evrişim ve pooling katmanlarından oluşmaktadır. Kullanılan aktivasyon fonksiyonu ReLU dur. Genel yapısı Şekil-3.20.'de ki gibidir.

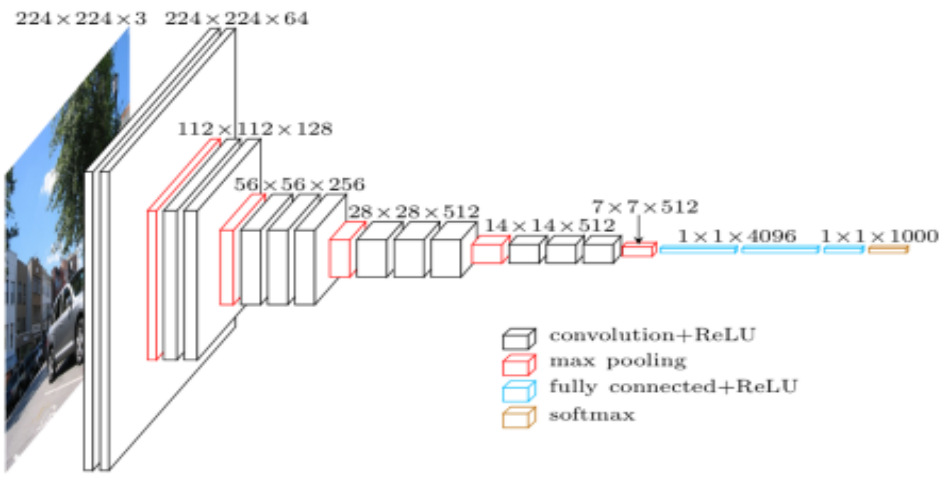


Şekil 3.20. - AlexNet Gösterimi (Kızrak, A., 2019)

Büyük ve derinliği daha fazla olan AlexNet (Şekil-3.21) de hesaplanan parametre sayısı 60 milyondur. ImageNet ILSVRC de sınıflandırma doğruluk oranını %74.3 den %83.6 ya dramtik bir şekilde iyileştirmiştir.

3.4.3. VGG-16

Temelde çok karmaşık olmayan bir CNN modeli olup diğer modellerden 2'li veya 3'lü evrişim katmanı kullanmasıyla ayrılmaktadır. Tam bağlantılı (FC) katmanda 7x7x512 = 4096 nöronlu öznitelik vektörlerini kullanır. İki FC katmanının çıkışında ise 1000 sınıflı softmax başarımının hesaplandığı bu modelde 138 milyon parametre hesaplanmaktadır. Girişten çıkışa doğru genişlik azalırken derinlik artmaktadır (Şekil-3.21.).

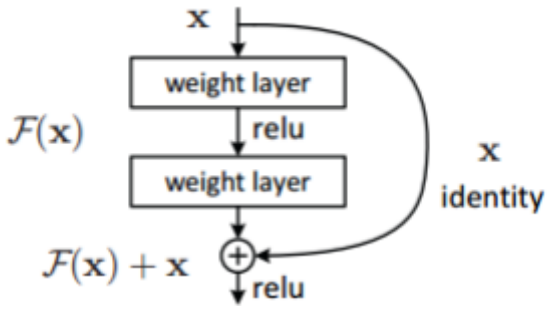


Şekil 3.20. - VGG-16 Temel Yapısı (https://neurohive.io/en/popular-networks/vgg16/, erişim tarihi: 19.08.2019)

3.4.4. ResNet

Klasik Evrişimli ağlarda ortaya çıkan degradasyon problemini çözmeye yönelik ResNet (Residiual Network) ağ modeli ortaya çıkmıştır. Degradasyon problemi ile ağın verimliliği doyuma ulaştıktan sonra daha fazla katman eklemek ağın verimini azaltmaktadır. ResNet modeli bu problemi çözmek için faklı bir yaklaşım benimsemektedir. ResNet modelinde Şekil-3.22.'de ki gibi bloklar bulunmaktadır. Bu blokların yapısında iki katman öncesinden gelen artık değerler blok sonunda eklenmektedir.

Bu sayede katmanlar arasındaki referans geçişleri sunarak optimizasyona yardımcı olmaktadır.



Şekil 3.21. - Artık değer Bloğu (https://www.deeplearningitalia.com/an-intuitive-guide-to-deep-network-architectures/, erişim tarihi: 19.08.2019)

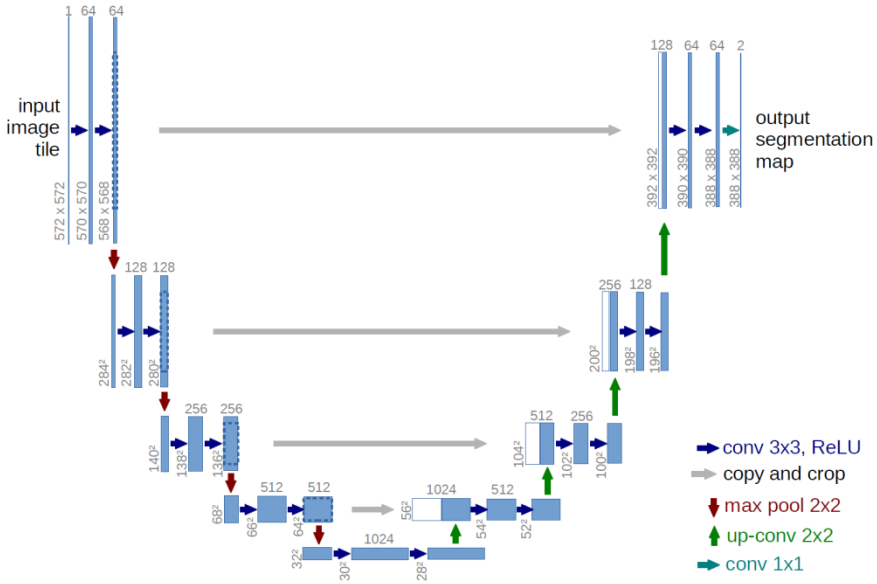
3.4.5. U-Net

U-Net mimarisini diğerlerinden farklı olarak ortaya çıkmasını sağlayan en önemli özellik, ağın içerik bilgilerini daha yüksek çözünürlük katmanlarına yayılmasına izin veren çok sayıda filtreleme kanalı olmasıdır. U-Net mimarinde enkoder aşamasında girdi imgesi farklı konvolusyonlardan geçirilerek 1024 öznitelik boyutuna düşürülmekte ve dekoder yapısında konvolusyonun tersi filtreler tercih edilerek tekrar istenilen çıktı imgesinin boyutu kadar filtrelemelerden geçirilmektedir.

Denetimli öğrenme (supervised learning) metodolojileri kullanılarak geliştirilen modeller denetimsiz öğrenme (unsupervised learning) yöntemlerine göre birçok avantaj bulunmaktadır. Denetimsiz öğrenme metodolojileri ile öğrenme işlemlerini model kendisi

yapıyor olsa da görüntünün alındığı kamera veya görüntüde yer alan objelerde ufak blur veya kamera gürültüsü olması durumlarında doğru sonuçlar elde edilememektedir. Bu durumda denetimli öğrenme mekanizmaları ile çalışmalar gerçekleştirerek bahsi geçen ve sonuçları olumsuz etkileyen durumların etkisini ortadan kaldırmak amaçlanmıştır.

Şekil 3-23.'de temel U-Net mimarisi ve her bir katman arasındaki geçiş aşamaları gösterilmiştir.



Şekil 3.22. - U-Net Mimarisi (Kızrak, A., 2019)

4. MATERYAL VE YÖNTEM

Bu çalışmada karaciğer bölütleme işlemi denetimli öğrenme yoluyla gerçekleştirilmiştir. Bunun için uzmanlar tarafından etiketlenmiş karaciğer bölgeleri referans (groundtruth) olarak kabul edilmiştir. Her bir referans imgenin piksel aralığı 0-255 arasına normalize edilmiştir. U-Net mimarisin en son katmanında oluşan imge ile referans imge arasındaki hatalar geri besleme yoluyla minimize edilmektedir. En az hataya ulaşmak için veri artırma tekniği devreye sokulmuştur.

4.1. Kaynak Veri Seti

Bu tez çalışmasında https://chaos.grand-challenge.org/ adresinde sunulan örnek karaciğer veri setleri kullanılmıştır. Veri kullanımı için ilgili sağlayıcıdan kullanıcı alınarak verilere erişilmiştir.

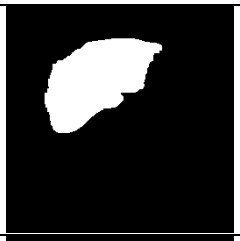
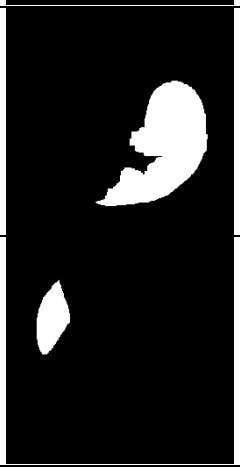
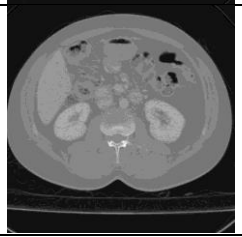
4.1.1. Veri seti özellikleri

Tez çalışmasında kullanılan CT görüntüleri 17 farklı (anonimleştirilmiş) hastadan alınmıştır. Bu hastalar; sağlıklı (tümör, lezyon veya herhangi bir hastalığa sahip olmayan) potansiyel karaciğer donörü olan hastalardır. Verilerin alındığı kaynakta da belirtildiği üzere: CT görüntüleri üst abdomen bölgesinden portal venöz fazda kontrast madde enjekte edilerek alınmıştır. Portal venöz faz; kontrast madde enjeksiyonundan sonra 70-80 saniye veya bolustracking işleminden 50-60 saniye sonra elde edilen fazdır. Bu aşamada karaciğer parankimi, kan damarı yoluyla portal ven tarafından maksimal olarak artar. Bu nedenle, bu faz ameliyattan önce karaciğer ve damar segmentasyonu için yaygın olarak kullanılır. Görüntü çıktı özellikleri aynı olan üç farklı modaliteden (spiral CT seçeneğine sahip) elde edilen görüntülerde, 16 dedektörlü Philips SecuraCT ve 64 dedektörlü bir Philips Mx8000 CT ve 320 dedektörlü Toshiba AquilionOne kullanılmıştır. Hasta oryantasyonu ve hizalaması tüm veri kümeleri için aynıdır.

Her veri seti 512x512, xy aralığı 0,7-0,8 mm arasında değişen ve 3 ila 3,2 mm arası dilim arası mesafeye sahip 16 bit DICOM görüntülerinden oluşur. Bu, veri seti başına ortalama 90 dilime tekabül eder (yani minimum 77, maksimum 105 dilim). Eğitim seti toplamda 2579 adet abdominal CT görüntüsünü içermektedir. Her bir görüntüye karşılık gelen groundtruth görüntüsü; mevcut görüntüdeki karaciğerin bulunduğu bölge için 1 (bir), bulunmadığı bölge için 0 (sıfır) piksel değerlerini içeren görüntülerden oluşmaktadır. Bu görüntülerden 102 tanesi test amaçlı olarak eğitim yapılmadan önce eğitim setinden çıkarılmıştır.

Bu veri seti içerisinde bulunan orijinal görüntü ve karşılığı groundtruth görüntülerin bazıları aşağıdaki Çizelge 4.1.'de verilmiştir. Çizelgenin ilk sütununda orijinal BT görüntüsü bulunurken, bu görüntü üzerinde karaciğer olan bölgeler uzman hekimler tarafından işaretlenerek ikinci sütundaki GroundTruth görüntüleri oluşturulmuştur.

Çizelge 4.1. - Örnek görüntüler ve Etiketler

Görüntü	**GroundTruth**
	
	
	

4.1.2. Veri Arttırma

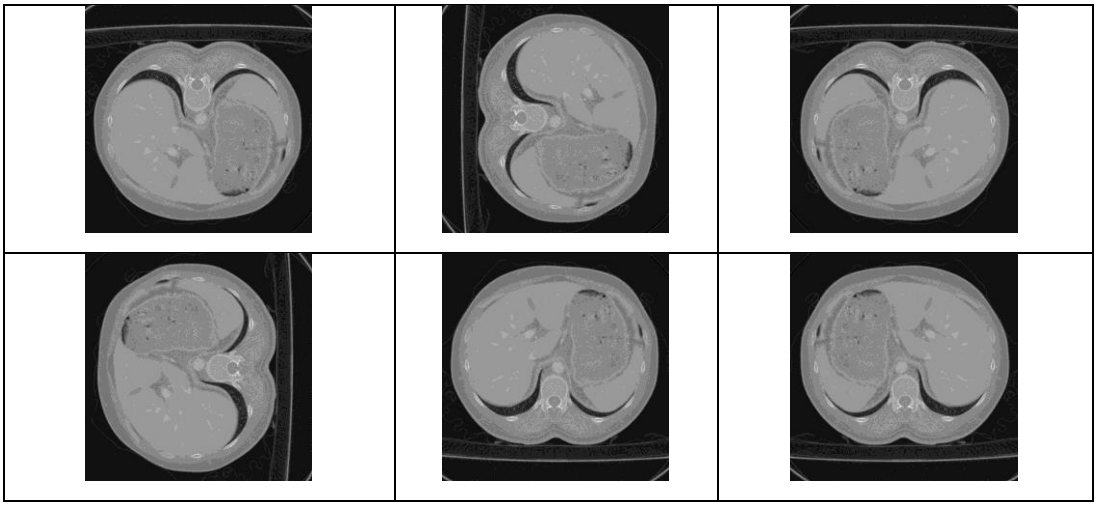
Veri artırma, ESA tabanlı derin öğrenme tabanlı çalışmalarda birçok avantaj sağlamaktadır. Bunlardan en önemlisi overfitting denilen filtrelerin öğrenmesini olumsuz etkileyen etkenin ortadan kaldırılmasıdır. Kısıtlı veri setlerinde modelin ezberlenmiş değerleri öğrenmesi yerine verinin çeşitli varyasyonları oluşturuluarak bu sorunun üstesinden gelinmeye çalışılır. Çizelge 4.2.'de verilen veri arttırma teknikleri tercih edilmiştir.

Çizelge 4.2. - Veri Arttırma Yöntemleri

rotation_range(rotasyon aralığı)	0.2
width_shift_range (yatay kaydırma aralığı)	0.05
height_shift_range (dikey kaydırma aralığı)	0.05
shear_range (shear kaydırma aralığı)	0.05
zoom_range (zoom aralığı)	0.05
horizontal_flip (yatay aynalama)	True
fill_mode (döndürme sonrası boşlukları kapatma yöntemi)	'nearest'

Çizelge 4.3.'de veri arttırma işlemleri ile elde edilmiş orijinal görüntü ve aynı görüntülerin farklı açılarda döndürülmesiyle elde edilen görüntülerinden bazıları gösterilmiştir.

Çizelge 4.3. - Arttırılmış Veri Örneği



4.2. Uygulama Ortamı, U-Net Hiper-parametreler

Verilerin işlenmesi, evrişimli sinir ağının oluşturulup eğitim aşamasının gerçekleştirilmesi ve sonuçların değerlendirilmesi için kullanılan yazılımsal ve donanımsal kaynaklar şu şekildedir;

Yazılmsal Kaynaklar:

- Python 3.6
- Anaconda 2019.3
- Tensorflow 1.13
- Cuda Toolkit 10.0.130
- NVIDIA Driver 418

Donanımsal Kaynaklar:

- Intel i7-8700K CPU
- 32 GB 2666 Mhz DDR4L RAM
- NVIDIA GTX 1080, 8 GB GDDR5 RAM, Grafik kartı

Tensorflow : Açık kaynak kodlu derin öğrenme kütüphanesi olan Tensorflow Esnek yapısıyla, platform bağımsız hesaplamaları, tekli veya çoklu CPU, GPU kullanarak kullanmamıza olanak sağlayan bir API kütüphanesidir. Python kullanılarak geliştirilmiş olan bu kütüphane, Python'ın yanısıra C++, Java, C#, Javascript ve R gibi birçok dili desteklemektedir.

NVidia GTX 1080 : bünyesinde barındırdığı 2560 bağımsız CUDA çekirdeği ile aritmetik hesaplamaları çok hızlı bir şekilde yapabilmesi sebebiyle özelikle yapay zeka ve benzeri işlemlerde sıklıkla kullanılmaktadır.

CUDA Toolkit : NVidia ekran kartlarındaki CUDA çekirdeklerini doğrudan kullanabilmek için NVidia tarafından geliştirilmiş olan bu araç kullanılmalıdır.

Bir ESA algoritmasının öğrenmesini sağlayan en önemli etken hiper-parametrelerdir. Bu parametreler epoch, batch size ve learning rate olarak verilebilir. Bu parametreler sezgisel olarak bulunamadığından deneysel olarak karar verilmek zorundadır. Bu çalışmada Çizelge 4.4.'de verilen hiper-parametreler tercih edilmiştir. Bu

parametrelerin seçilirken elde bulunan eğitim setinde ki örnek sayısii verinin türü ve litaratürde sıklıkla kullanılan değerler baz alınmıştır. Deney sürecinde her 1 epochtaki oluşan modelin performansı göz önüne alınarak, en yüksek doğruluk oranı veren model kaydedilmiştir.

Çizelge 4.4. - U-Net Hiper parametreler (Kızrak, A., 2018)

epoch	20
Steps per epoch	300
Batch Size	2
Verbose (Model Kaydetme Periyodu)	1
Loss Function	Binary crossentropy
Optimization Function	Adam
Learning Rate	1e-4

Bu çalışmada Anaconda aracı kullanılarak Python programlama dili ile kodlama yapılmıştır. Anaconda, veri bilimi veya görüntü işleme gibi uygulamalarda python kullanımına olanak sağlayan tümleşik bir python dağıtımıdır. Aynı zamanda çalışma kapsamında tercih edilen birçok kütüphaneyi yapısında barındırmaktadır. Anaconda ile en uygun python versiyonuna göre environment oluşturulduktan sonra Tensorflow kütüphanesi indirilmiştir. Tüm deneyler Python ortamında gerçekleştirilmiştir.

4.3. Kullanılan Metrikler

Bu çalışmada literatürde ikili formatındaki imgeler için en çok tercih edilen karşılaştırma metrikleri kullanılmıştır. Bunun için referans ve çıktı imgesi karşılaştırılıarak, True Positive (TP), False Positive (FP), False Negative (FN) ve True Negative (TN) sayıları bulunmuştur. Daha sonra dengeli bir değerlendirme olması açısıdan elde edilen TP, FP, TN, FN değerleri ile Precision, Recall, PBC, specificity, FPR, FNR, FMeasure sonuçları bulunmuştur. Referans imgedeki piksel GT(x,y) olsun ve elde edilen çıktı imgedeki piksel I(x,y) olmak üzere TP, FP, TN, FN elde etmek için aşağıdaki kurullar kullanılmıştır.

- **TP**: GT(x,y)=255 ve I(x,y)=255

- **TN**: GT(x,y)=0 ve I(x,y)=0
- **FP**: GT(x,y)= 0 ve I(x,y)=255
- **FN**: GT(x,y)=255 ve I(x,y)=0

Elde edilen bu değerler ile aşağıda formülleri verilen metrikler hesaplanmıştır. (Ghoneim, S, 2019)

$$\textbf{recall} = TP / (TP + FN) \quad (4.1)$$

$$\textbf{specficitiy} = TN / (TN + FP) \quad (4.2)$$

$$\textbf{FPR} = FP / (TN + FP) \quad (4.3)$$

$$\textbf{FNR} = FN / (TN + FP) \quad (4.4)$$

$$\textbf{PBC} = 100.0 * (FN + FP) / (TP + FP + FN + TN) \quad (4.5)$$

$$\textbf{precision} = TP / (TP + FP) \quad (4.6)$$

$$\textbf{FMeasure} = 2.0 * (recall * precision) / (recall + precision) \quad (4.7)$$

4.4. Karşılaştırılan ESA Çeşitleri

Farklı U-net ağ yapılarının başarımının ölçülmesi ve karşılaştırılması için bu tez çalışmasında aşağıda detayları verilen farklı ağ mimarileri aynı eğitim sistemiyle aynı hiper parametreler ile eğitime tabi tutulmuş ve aynı test veri seti ile başarımı ölçülmüştür.

4.4.1. Basit ESA

Sınırlı sayıda girdi ve çıktı katmanıyla en temel yapısıyla basit bir ESA modeli kurulmuştur. Bu modelde eğitim setindeki 512x512 boyutlu imgeler eğitim için verilirken 256x256 ya küçültülmüş ve eğitim bu boyutlar ile tamamlanmıştır. Başarım ölçülürken elde edilen 256x256 çıktı imgesi 512x512 ye tekrar ölçeklenmiş ve referans görüntüler ile karşılaştırılmıştır. Modelin Python kodlarındaki karşılığı aşağıdaki gibidir:

```
inputs = Input(input_size)
    conv1 = Conv2D(64, 3, activation = 'relu', padding = 'same',
                   kernel_initializer = 'he_normal')(inputs)
    conv2 = Conv2D(64, 3, activation = 'relu', padding = 'same',
                   kernel_initializer = 'he_normal')(conv1)
    conv3 = Conv2D(64, 3, activation = 'relu', padding = 'same',
                   kernel_initializer = 'he_normal')(conv2)
```

```
    conv4 = Conv2D(64, 3, activation = 'relu', padding = 'same',
                           kernel_initializer = 'he_normal')(conv3)
    conv5 = Conv2D(64, 3, activation = 'relu', padding = 'same',
                           kernel_initializer = 'he_normal')(conv4)
    conv6 = Conv2D(1, 1, activation = 'sigmoid')(conv5)

    model = Model(input = inputs, output = conv6)

    model.compile(optimizer = Adam(lr = 1e-4),
                           loss = 'binary_crossentropy', metrics = ['accuracy']
```

4.4.2. U-Net 256x256

Bu modelde temel U-net yapısı kullanılmıştır. Aktivasyon fonksiyonu RELU dur. Ayrıca eğitim setindeki 512x512 boyutlu imgeler eğitim için verilirken 256x256 ya küçültülmüş ve eğitim bu boyutlar ile tamamlanmıştır. Başarım ölçülürken elde edilen 256x256 çıktı imgesi 512x512 ye tekrar ölçeklenmiş ve referans görüntüler ile karşılaştırılmıştır. Modelin Python kodlarındaki karşılığı aşağıdaki gibidir.

```
def unet(pretrained_weights = None,input_size = (256,256,1)):
    inputs = Input(input_size)
    conv1 = Conv2D(64, 3, activation = 'relu', padding = 'same',
                           kernel_initializer = 'he_normal')(inputs)
    conv1 = Conv2D(64, 3, activation = 'relu', padding = 'same',
                           kernel_initializer = 'he_normal')(conv1)
    pool1 = MaxPooling2D(pool_size=(2, 2))(conv1)
    conv2 = Conv2D(128, 3, activation = 'relu', padding = 'same',
                           kernel_initializer = 'he_normal')(pool1)
    conv2 = Conv2D(128, 3, activation = 'relu', padding = 'same',
                           kernel_initializer = 'he_normal')(conv2)
    pool2 = MaxPooling2D(pool_size=(2, 2))(conv2)
    conv3 = Conv2D(256, 3, activation = 'relu', padding = 'same',
                           kernel_initializer = 'he_normal')(pool2)
    conv3 = Conv2D(256, 3, activation = 'relu', padding = 'same',
                           kernel_initializer = 'he_normal')(conv3)
    pool3 = MaxPooling2D(pool_size=(2, 2))(conv3)
    conv4 = Conv2D(512, 3, activation = 'relu', padding = 'same',
                           kernel_initializer = 'he_normal')(pool3)
    conv4 = Conv2D(512, 3, activation = 'relu', padding = 'same',
                           kernel_initializer = 'he_normal')(conv4)
    drop4 = Dropout(0.5)(conv4)
    pool4 = MaxPooling2D(pool_size=(2, 2))(drop4)

    conv5 = Conv2D(1024, 3, activation = 'relu', padding = 'same',
                           kernel_initializer = 'he_normal')(pool4)
    conv5 = Conv2D(1024, 3, activation = 'relu', padding = 'same',
                           kernel_initializer = 'he_normal')(conv5)
    drop5 = Dropout(0.5)(conv5)

    up6 = Conv2D(512, 2, activation = 'relu', padding = 'same',
           kernel_initializer = 'he_normal')(UpSampling2D(size = (2,2))(drop5))
    #merge6 = merge([drop4,up6], mode = 'concat', concat_axis = 3)
    merge6 = concatenate([drop4, up6], axis=3)
    conv6 = Conv2D(512, 3, activation = 'relu', padding = 'same',
```

```
                          kernel_initializer = 'he_normal')(merge6)
    conv6 = Conv2D(512, 3, activation = 'relu', padding = 'same',
                          kernel_initializer = 'he_normal')(conv6)

    up7 = Conv2D(256, 2, activation = 'relu', padding = 'same',
kernel_initializer = 'he_normal')(UpSampling2D(size = (2,2))(conv6))
    #merge7 = merge([conv3,up7], mode = 'concat', concat_axis = 3)
    merge7 = concatenate([conv3, up7], axis=3)
    conv7 = Conv2D(256, 3, activation = 'relu', padding = 'same',
                          kernel_initializer = 'he_normal')(merge7)
    conv7 = Conv2D(256, 3, activation = 'relu', padding = 'same',
                          kernel_initializer = 'he_normal')(conv7)

    up8 = Conv2D(128, 2, activation = 'relu', padding = 'same',
kernel_initializer = 'he_normal')(UpSampling2D(size = (2,2))(conv7))
    #merge8 = merge([conv2,up8], mode = 'concat', concat_axis = 3)
    merge8 = concatenate([conv2, up8], axis=3)
    conv8 = Conv2D(128, 3, activation = 'relu', padding = 'same',
                          kernel_initializer = 'he_normal')(merge8)
    conv8 = Conv2D(128, 3, activation = 'relu', padding = 'same',
                          kernel_initializer = 'he_normal')(conv8)

    up9 = Conv2D(64, 2, activation = 'relu', padding = 'same',
kernel_initializer = 'he_normal')(UpSampling2D(size = (2,2))(conv8))
    #merge9 = merge([conv1,up9], mode = 'concat', concat_axis = 3)
    merge9 = concatenate([conv1, up9], axis=3)
    conv9 = Conv2D(64, 3, activation = 'relu', padding = 'same',
                          kernel_initializer = 'he_normal')(merge9)
    conv9 = Conv2D(64, 3, activation = 'relu', padding = 'same',
                          kernel_initializer = 'he_normal')(conv9)
    conv9 = Conv2D(2, 3, activation = 'relu', padding = 'same',
                          kernel_initializer = 'he_normal')(conv9)
    conv10 = Conv2D(1, 1, activation = 'sigmoid')(conv9)

    model = Model(input = inputs, output = conv10)

    model.compile(optimizer =
         Adam(lr = 1e-4), loss = 'binary_crossentropy', metrics = ['accuracy'])

    if(pretrained_weights):
        model.load_weights(pretrained_weights)

    return model
```

4.4.3. U-Net 512x512

Bu modelde temel U-net yapısı kullanılmıştır. Aktivasyon fonksiyonu RELU dur. Ayrıca eğitim setindeki 512x512 boyutlu imgeler eğitim için doğrudan kullanılmış ve başarım ölçülürken elde edilen 512x512 çıktı imgesi referans görüntüler ile karşılaştırılmıştır. Modeli dinamik olarak oluşturan Python kodu aşağıdaki gibidir. Oluşturulan bu kod ile ***act_func*** parametresi ile ağın aktivasyon fonksiyonu dinamik olarak değiştirilebilmektedir.

```
def unet(pretrained_weights = None,input_size = (512,512,1), act_func = 'relu'):

    print(act_func)
    inputs = Input(input_size)
    conv1 = Conv2D(64, 3, activation = act_func, padding = 'same',
           kernel_initializer = 'he_normal')(inputs)
    conv1 = Conv2D(64, 3, activation = act_func, padding = 'same',
           kernel_initializer = 'he_normal')(conv1)
    pool1 = MaxPooling2D(pool_size=(2, 2))(conv1)
    conv2 = Conv2D(128, 3, activation = act_func, padding = 'same',
           kernel_initializer = 'he_normal')(pool1)
    conv2 = Conv2D(128, 3, activation = act_func, padding = 'same',
           kernel_initializer = 'he_normal')(conv2)
    pool2 = MaxPooling2D(pool_size=(2, 2))(conv2)
    conv3 = Conv2D(256, 3, activation = act_func, padding = 'same',
           kernel_initializer = 'he_normal')(pool2)
    conv3 = Conv2D(256, 3, activation = act_func, padding = 'same',
           kernel_initializer = 'he_normal')(conv3)
    pool3 = MaxPooling2D(pool_size=(2, 2))(conv3)
    conv4 = Conv2D(512, 3, activation = act_func, padding = 'same',
           kernel_initializer = 'he_normal')(pool3)
    conv4 = Conv2D(512, 3, activation = act_func, padding = 'same',
           kernel_initializer = 'he_normal')(conv4)
    drop4 = Dropout(0.5)(conv4)
    pool4 = MaxPooling2D(pool_size=(2, 2))(drop4)

    conv5 = Conv2D(1024, 3, activation = act_func, padding = 'same',
            kernel_initializer = 'he_normal')(pool4)
    conv5 = Conv2D(1024, 3, activation = act_func, padding = 'same',
           kernel_initializer = 'he_normal')(conv5)
    drop5 = Dropout(0.5)(conv5)

    up6 = Conv2D(512, 2, activation = act_func, padding = 'same',
           kernel_initializer = 'he_normal')(UpSampling2D(size = (2,2))(drop5))
    #merge6 = merge([drop4,up6], mode = 'concat', concat_axis = 3)
    merge6 = concatenate([drop4, up6], axis=3)
    conv6 = Conv2D(512, 3, activation = act_func, padding = 'same',
            kernel_initializer = 'he_normal')(merge6)
    conv6 = Conv2D(512, 3, activation = act_func, padding = 'same',
           kernel_initializer = 'he_normal')(conv6)

    up7 = Conv2D(256, 2, activation = act_func, padding = 'same',
           kernel_initializer = 'he_normal')(UpSampling2D(size = (2,2))(conv6))
    #merge7 = merge([conv3,up7], mode = 'concat', concat_axis = 3)
    merge7 = concatenate([conv3, up7], axis=3)
    conv7 = Conv2D(256, 3, activation = act_func, padding = 'same',
           kernel_initializer = 'he_normal')(merge7)
    conv7 = Conv2D(256, 3, activation = act_func, padding = 'same',
           kernel_initializer = 'he_normal')(conv7)

    up8 = Conv2D(128, 2, activation = act_func, padding = 'same',
           kernel_initializer = 'he_normal')(UpSampling2D(size = (2,2))(conv7))
    #merge8 = merge([conv2,up8], mode = 'concat', concat_axis = 3)
    merge8 = concatenate([conv2, up8], axis=3)
    conv8 = Conv2D(128, 3, activation = act_func, padding = 'same',
           kernel_initializer = 'he_normal')(merge8)
    conv8 = Conv2D(128, 3, activation = act_func, padding = 'same',
           kernel_initializer = 'he_normal')(conv8)

    up9 = Conv2D(64, 2, activation = act_func, padding = 'same',
           kernel_initializer = 'he_normal')(UpSampling2D(size = (2,2))(conv8))
    #merge9 = merge([conv1,up9], mode = 'concat', concat_axis = 3)
    merge9 = concatenate([conv1, up9], axis=3)
    conv9 = Conv2D(64, 3, activation = act_func, padding = 'same',
           kernel_initializer = 'he_normal')(merge9)
```

```
65.     conv9 = Conv2D(64, 3, activation = act_func, padding = 'same',
66.            kernel_initializer = 'he_normal')(conv9)
67.     conv9 = Conv2D(2, 3, activation = act_func, padding = 'same',
68.            kernel_initializer = 'he_normal')(conv9)
69.     conv10 = Conv2D(1, 1, activation = 'sigmoid')(conv9)
70.
71.     model = Model(input = inputs, output = conv10)
72.
73.     model.compile(optimizer = Adam(lr = 1e-
    4), loss = 'binary_crossentropy', metrics = ['accuracy'])
74.
75.
76.     if(pretrained_weights):
77.         model.load_weights(pretrained_weights)
78.
79.     return model
```

4.4.4. U-Net 512x512 (Sigmoid)

Bu modelde temel U-net yapısı kullanılmıştır. Ancak aktivasyon fonksiyonu değiştirilerek RELU yerine Sigmoid kullanılmıştır. Eğitim setindeki 512x512 boyutlu imgeler eğitim için doğrudan kullanılmış ve başarım ölçülürken elde edilen 512x512 çıktı imgesi referans görüntüler ile karşılaştırılmıştır. 4.4.3. de verilen model koduna ***act_func*** parametresi ***sigmoid*** gönderilmiştir.

4.4.5. U-Net 512x512 (Linear)

Bu modelde temel U-net yapısı kullanılmıştır. Ancak aktivasyon fonksiyonu değiştirilerek RELU yerine Linear kullanılmıştır. Eğitim setindeki 512x512 boyutlu imgeler eğitim için doğrudan kullanılmış ve başarım ölçülürken elde edilen 512x512 çıktı imgesi referans görüntüler ile karşılaştırılmıştır. 4.4.3. de verilen model koduna ***act_func*** parametresi ***linear*** gönderilmiştir.

5. BULGULAR VE TARTIŞMA

Bölüm 4.4. de verilen parametreler ile hazırlanan ağ modelleri, 20 epoch ve her epoch içerisinde 300 iterasyon içerek şekilde örnek veri seti ile eğitim sürecine tabi tutulmuştur. Modellerin eğitim sürecindeki Loss (kayıp) ve Accuracy (doğruluk) değerleri Çizelge 5.1.'de gösterildiği gibidir.

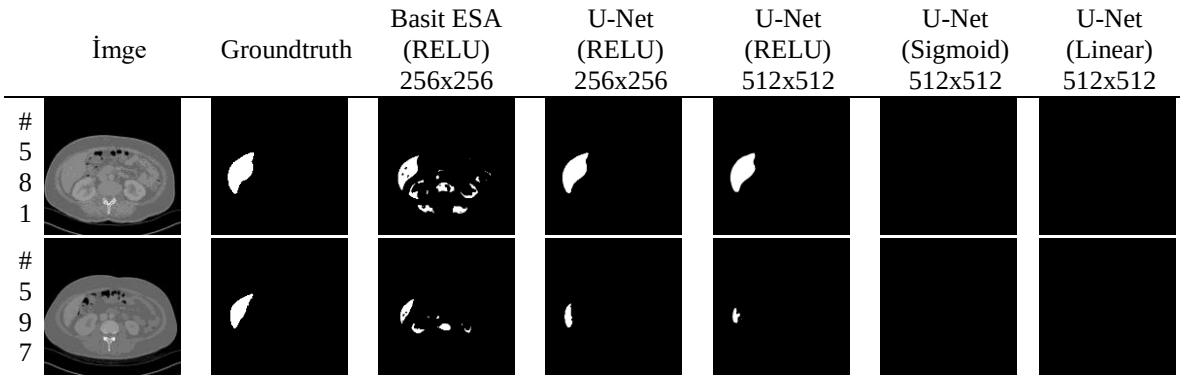
Çizelge 5.1. - Eğitim süreci elde edilen değerler

	Basit ESA (RELU) 256x256		**U-Net (RELU) 256x256**		**U-Net (RELU) 512x512**		**U-Net (Sigmoid) 512x512**		**U-Net (Linear) 512x512**	
Epoch	**Loss**	**Acc**	**Loss**	**Acc**	**Loss**	**Acc**	**Loss**	**Acc**	**Loss**	**Acc**
1	0.2708	0.9299	0.2131	0.9285	0.2046	0.9287	0.2980	0.9303	1.1355	0.9291
2	0.2053	0.9269	0.1641	0.9269	0.1707	0.9268	0.2911	0.9268	1.1792	0.9268
3	0.1820	0.9266	0.1378	0.9266	0.1504	0.9264	0.2861	0.9266	1.1832	0.9266
4	0.1779	0.9251	0.1482	0.9249	0.1235	0.9249	0.2848	0.9249	1.2111	0.9249
5	0.1681	0.9248	0.1215	0.9311	0.1268	0.9254	0.2803	0.9254	1.2026	0.9254
6	0.1578	0.9283	0.0950	0.9641	0.1041	0.9539	0.2718	0.9281	1.1584	0.9281
7	0.1480	0.9309	0.0846	0.9725	0.0842	0.9712	0.2682	0.9286	1.1512	0.9286
8	0.1629	0.9270	0.0909	0.9699	0.0790	0.9769	0.2689	0.9272	1.1738	0.9272
9	0.1530	0.9276	0.0788	0.9773	0.0768	0.9800	0.2758	0.9231	1.2387	0.9231
10	0.1491	0.9294	0.0732	0.9813	0.0766	0.9797	0.2685	0.9260	1.1927	0.9260
11	0.1513	0.9289	0.0669	0.9834	0.0823	0.9759	0.2664	0.9265	1.1851	0.9265
12	0.1456	0.9296	0.0656	0.9835	0.0691	0.9826	0.2631	0.9275	1.1679	0.9275
13	0.1367	0.9356	0.0984	0.9638	0.0670	0.9828	0.2619	0.9277	1.1649	0.9277
14	0.1441	0.9299	0.0663	0.9817	0.0687	0.9806	0.2625	0.9272	1.1733	0.9272
15	0.1380	0.9339	0.0601	0.9871	0.0654	0.9846	0.2673	0.9250	1.2087	0.9250
16	0.1381	0.9334	0.0585	0.9863	0.0676	0.9820	0.2618	0.9272	1.1733	0.9272
17	0.1354	0.9349	0.0589	0.9860	0.0798	0.9741	0.2663	0.9252	1.2052	0.9252
18	0.1303	0.9374	0.0553	0.9884	0.0599	0.9856	0.2558	0.9295	1.1364	0.9295
19	0.1356	0.9344	0.0527	0.9887	0.0484	0.9854	0.2578	0.9286	1.1510	0.9286
20	0.1306	0.9377	0.0561	0.9882	0.0322	0.9873	0.2668	0.9249	1.2111	0.9249

Eğitim sürecinde görüldüğü üzere 256x256 lık modellerde Basit ESA yapısı doğruluk oranını 0.9299 dan en iyi 0.9377 oranına kadar geliştirebilirken U-Net yapısı ise 0.9285 den en iyi 0.9887 oranına kadar geliştirmiştir. Basit ESA yapısının eğitim sürecinde de öğrenme seviyesinin çok verimli olmadığı görülmektedir. U-Net (RELU) 256x256 modeli ile U-Net (RELU) 512x512 modeli eğitim süreci değerlendirdiğinde ise çok yakın bir doğruluk ile başlayıp, eğitim sonunda benzer bir doğruluk oranı değerini yakladığı görülmektedir. Öte yandan U-Net 512x512 modelleri arasında Sigmoid ve Linear aktivasyon fonksiyonlarına sahip olan modellerin Basit ESA modelinde olduğu gibi öğrenme sürecinde verimli olmadığı ve kabul edilebilir ölçüde öğrenmeyi gerçekleştirmediği görülmektedir.

Eğitilmiş modelin son haline test için verilen girdilere karşılık gelen groundtruth ve modelin çıktısı Çizelge 5.2.'de gösterilmiştir.

Çizelge 5.2. - Görsel Sonuçlar



Çizelge 5.3 - Nesnel Sonuçlar

Metrik	Basit ESA 256x256	U-Net (Relu) 256x256	U-Net (Relu) 512x512	U-Net (Linear) 512x512	U-Net (Sigmoid) 512x512
TP	565488	878526	824976	0	0
FP	754596	16218	6324	0	0
FN	340476	27438	80988	905964	905964
TN	25078128	25816506	25826400	25832724	25832724
Recall (%)	62.42	96.97	91.06	0.00	0.00

Specificity (%)	97.08	99.94	99.98	100.00	100.00
FPR (%)	2.92	0.06	0.02	0.00	0.00
FNR (%)	37.58	3.03	8.94	100.00	100.00
PBC (%)	409.55	16.33	32.65	338.82	338.82
Precision (%)	42.84	98.19	99.24	0.00	0.00
FMeasure (%)	50.81	**97.58**	**94.97**	0.00	0.00

Çizelge 5.2.'de ki görsel sonuçlardan anlaşılacağı gibi önerilen U-Net (RELU) yapısı 256x256 ve 512x512 yapıları referans imgeye yakın sonuçlar üretebilmiştir. Diğer yandan Basit ESA yapısı çok kötü sonuçlar üretirken, U-Net in aktivasyon fonksiyonu değiştirildiğinde (Linear veya Sigmoid, 512x512) hiçbir sonuç üretememiştir. Basit ESA yapısındaki filtreler küçük sayıda olduklarından dolayı overfitting durumu oluşturmaktadır ve kötü sonuç üretilmesine sebep olmuştur. Görüldüğü gibi Basit ESA algoritması piksellerdeki yumuşak geçişleri öğrenememektedir. Diğer yandan U-Net (RELU) yapısı gözle dahi görülmeyen karaciğer alanlarını bölütleyebilmiştir. U-Net 512x512 için Linear ve Sigmoid fonksiyonlarının karaciğer bölgelerini öğrenemedikleri gözlenmiştir. Bunun nedeni karaciğer verisinin 0-255 arasında olduğundan RELU aktivasyon fonksiyonuyla daha uyumlu çalışmasından kaynaklanmaktadır. Ayrıca adil bir karşılaştırma olması için tüm modellerde üretilen çıktılar için eşik değeri 0.5 olarak ayarlanmıştır. RELU aktivasyon fonksiyonlu U-Net ler 0.5 eşik değerinde en iyi sonucu üretirken aynı eşk değerinde Sigmoid ve Linear aktivasyon fonksiyonlu U-Net ler bu eşik değerine göre başarı elde edememişlerdir.

Performans değerlendirmesi için 4.3. de verilen (4.1) - (4.7) denklemlerinde ki nesnel metrikler kullanılmıştır. Bu çalışmada kullanılan metriklerin performansları yüzdelik olarak sergilenmiştir. Elde ettiğimiz sonuçlar Çizelge 5.3'te verilmiştir. F-measure metriğini tercih etmemizin nedeni precision ve recall değerlerinin harmonik ortalamasını aldığından dolayı daha güvenilir nesnel sonuçlar vermektedir. Recall, Specificity, FPR, FNR, PBC, Precision ve F-measure metrikleri 0-1 arasında değer almaktadır. F-measure, Recall, Specificity ve Precision için en yüksek tanıma değerinde 1 değeri vermekte diğer türlü 0 değerine yaklaşmaktadır. Yine FPR, FNR ve PBC değerlerinde 0 yaklaşması durumunda iyi performans verdiği bilinmektedir.

Sonuçlar incelediğinde, 256x256 girdi boyutunda; U-net yapısı aşikar olarak Basit ESA yapısından daha başarılıdır. F-measure değeri referans alındığında, deneysel sonuçlara göre U-net yapısı 97.58% F-measure skoru verirken, Basit ESA yapısı sadece 50.81% F-measure skoru verebilmiştir. Yine Specificity değeri incelediğinde karaciğer olmayan bölgelerin Basit ESA tarafından daha fazla oranda karaciğer bölgesi olarak işaretlendiği görülmektedir. Precision sonuçları bize bir yöntemin karaciğer olması gereken yerlerin ne kadar oranda karaciğer olarak işaretlediği bilgisini vermektedir. Precision sonuçları temel alındığında U-Net 98.19% skor verirken, Basit ESA yapısı ancak 42.84% performans skoru sergilemiştir.

Basit ESA yapısının düşük performans vermesinin nedeni, sınırlı sayıda veri ile eğitildiğinde overfitting veya underfitting durumuyla karşılaşmasıdır. Yeterli sayıda filtrenin olmamasından dolayı Basit ESA yapısı doğru sonuçları üretememektedir. Diğer yandan U-net yapısı incelendiğinde, auto-encoder paradigmasının filtrelere taşınmasından dolayı daha iyi görsel çıktılar verebilmiştir. Eşik değerleri ile oynanarak U-Net yapısının sonuçları daha da iyileştirilebilinir. Fakat adil bir karşılaştırma olması açısından eşik değeri U-Net ve Basit ESA içinde 128 olarak alınmıştır.

Diğer yandan adil bir performans karşılaştırması gerçekleştirmek için karaciğer imgeleri 512x512 boyutlarında (orjinal) eğitim işlemi gerçekleştirilmiştir. Farklı aktivasyon fonksiyonlarıyla eğitim işlemi tekrarlanmıştır. Çizelge 5.3'te verilen sonuçlar göz önüne alındığında Relu aktivasyon fonksiyonu iyi sonuçlar üretirken Linear ve Sigmoid fonksiyonları karaciğer bölgelerini öğrenemedikleri gözlenmiştir. Bunun nedeni karaciğer verisinin 0-255 arasında olduğundan Relu aktivasyon fonksiyonuyla daha uyumlu çalışmasından kaynaklanmaktadır.

Karaciğer imgeleri 256x256 boyutlarında eğitim yapıldığında 97.58% F-measure değeri üretirken 512x512 boyutlarında eğitim yapıldığında performans 94.97% F-measure skoru vermiştir. Performans düşüklüğünün sebebi yüksek çözünürlükte karaciğerdeki gürültülü değerlerinin öğrenmeyi kötü etkilediği gözlenmiştir. Çünkü yüksek çözünürlükte daha detaylı bölgeler de eğitim sürecine dahil olmuştur. Sonuç olarak eğitim aşamasında

imgeleri 256x256 boyutunda eğitmek ve test aşamasında çıkan 256x256 lık sonuçları 512x512 boyutlarına yükseltmenin daha başarılı sonuçlar verdiği not edilmiştir.

6. SONUÇ VE ÖNERİLER

Bu çalışmada derin öğrenme tabanlı yöntemlerin karaciğer bölütlemesi üzerine performansı analiz edilmiştir. Bu amaç için en çok tercih edilen U-Net yapısı 256x256 ve 512x512 boyutlarındaki imgeler üzerine uygulanarak sonuçlar incelenmiştir. Ayrıca U-Net yapısı ile basit bir CNN yapısı arasındaki performansı analiz edilmiştir.

Kurulan derin öğrenme modelleri içinde 256x256 giriş ile eğitilmiş RELU aktivasyon fonksiyonu ile çalışan U-Net modeli beklenen başarımı göstermiştir. Eğitim veri seti 512x512 ye çıkarıldığında ise RELU ile çalışan U-Net modelinin başarımı kabul edilebilir seviyede olmasına rağmen bir miktar düşüş göstermiştir. Aktivasyon fonksiyonu Sigmoid veya Linear seçildiğinde ise 512x512 giriş ile çalışılan modeller sonuç üretememiştir.

Basit ESA modelinin ise 256x256 girişte bile kabul edilebilir sonuçlar üretememiştir. Günümüzde artık Tomografi cihazları git gide daha yüksek çözünürlüklerde görüntüler üretmeye başlamıştır. HR CT (High resolution CT) gibi görüntü üretebilen tomografi cihazlarının görüntüleri 1024x1024 çözünürlüğe ulaşmıştır.

Sonraki yapılacak çalışmalarda U-Net ağ yapısı RELU aktivasyon fonksiyonu ile değişken boyutlu girdilere (256x256, 384x384, 512x512, 768x768, 1024x1024 vs..) göre çıktı üretebilecek şekilde eğitilip dinamik bir model oluşturulması denenebilir ve başarım karşılaştırması yapılabilir. Son zamanlarda özellikle organ bölütlemesinde öne çıkmaya çalışan Seg-Net mimarisi ile U-Net mimarisinin karşılaştırılması yapılabilir.

KAYNAKLAR DİZİNİ

Başak, M., Akan, D., 2015, Karaciğerin ve Safra Yollarının Radyolojik Anatomisi, Türk Radyoloji Seminerleri s.336-348

Bayramlı, B., 2015, Bilgisayar Bilim, Yapay Zeka, https://burakbayramli.github.io/dersblog/algs/convnet/evrisimsel_aglar_derin_ogrenim__convolutional_nets_convnet_deep_learning_.html, erişim tarihi: 27.04.2019

Couinaud C., 1957, Etudes anatomiques et chirurgicales. Paris: Masson.

Çarkacı, N., 2018, Derin Öğrenme Uygulamalarında En Sık kullanılan Hiper-parametreler, https://medium.com/deep-learning-turkiye/derin-ogrenme-uygulamalarinda-en-sik-kullanilan-hiper-parametreler-ece8e9125c4, erişim tarihi: 27.04.2019

Ergin, T., 2018, Convolutional Neural Network (ConvNet yada CNN) nedir, nasıl çalışır?, https://medium.com/@tuncerergin/convolutional-neural-network-convnet-yada-cnn-nedir-nasil-calisir-97a0f5d34cad, erişim tarihi: 27.04.2019

Ginger, M. L., Giger, M.L., Bae, K.T., MacMahon, H., 1994, Computerized detection of pulmonary nodules in computed tomography images, Invest. Radiol. 29, s.459–465.

Ghoneim, S, 2019, Accuracy, Recall, Precision, F-Score, Specificity, which to optimize on?, https://towardsdatascience.com/accuracy-recall-precision-f-score-specificity-which-to-optimize-on-867d3f11124, erişim tarihi : 19.08.2019

Goldsmith N., Woodburne R., 1957, Surgical anatomy pertaining to liver resection. Surg Gynecol Obstetr , s.310-318.

İnik, Ö., Ülker, E., 2017, Derin Öğrenme ve Görüntü Analizinde Kullanılan Derin Öğrenme Modelleri, Gaziosmanpaşa Bilimsel Araştırma Dergisi (Gbad), s.85-104.

Kılıkçıer, Ç., Yılmaz, E., 2018, BT Görüntülerden Akciğerin Tespiti için Süper Piksel ve Yapay Sinir Ağı Tabanlı BirYöntem, Erzincan Üniversitesi Fen Bilimleri Enstitüsü Dergisi, s.223-230.

Kızrak, A., 2018, ŞU KARA KUTUYU AÇALIM: Yapay Sinir Ağları, https://medium.com/deep-learning-turkiye/%C5%9Fu-kara-kutuyu-a%C3%A7alim-yapay-sinir-a%C4%9Flar%C4%B1-7b65c6a5264a, erişim tarihi: 27.04.2019

Kızrak, A., 2018, Derin Öğrenme Uygulamalarında En Sık kullanılan Hiper-parametreler, https://medium.com/deep-learning-turkiye/derin-ogrenme-uygulamalarinda-en-sik-kullanilan-hiper-parametreler-ece8e9125c4 , erişim tarihi : 14.09.2019

KAYNAKLAR DİZİNİ (devam)

Kızrak, A., 2019, Derin Öğrenme İçin Aktivasyon Fonksiyonlarının Karşılaştırılması, https://medium.com/@ayyucekizrak/derin-%C3%B6%C4%9Frenme-i%C3%A7in-aktivasyon-fonksiyonlar%C4%B1n%C4%B1n-kar%C5%9F%C4%B1la%C5%9Ft%C4%B1r%C4%B1lmas%C4%B1-cee17fd1d9cd , erişim tarihi : 14.09.2019

Krizhevsky, A., Sutskever, I., Hinton, G.E., ImageNet Classification with Deep Convolutional Neural Networks, https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf, erişim tarihi: 27.04.2019

LeCun, Y., 1998, Gradient-Based Learning Applied to Document Recognition, http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf, erişim tarihi:27.04.2019

Saxena, H., 2015, DICOM Image Segmentation with CNNs in Tensorflow, https://github.com/harsh1795/CNN-DICOM-Segmentation, erişim tarihi: 27.04.2019.

Şengül N., 2017, Yapay Sinir Ağları http://www.derinogrenme.com/2017/03/04/yapay-sinir-aglari/ , erişim tarihi: 14.08.2019

Wismüller, A., F, Vietze., 2000, Segmentation with neural network, Academic Press.