TÜRK HAVA KURUMU ÜNİVERSİTESİ
FEN BİLİMLER ENSTİTÜSÜ

DERİN ÖĞRENME YÖNTEMLERİ İLE GÖĞÜS KANSERİ TEŞHİSİ

YÜKSEK LİSANS TEZİ

ALI MAHMOOD OGUR ANWER

Elektrik ve Bilgisayar Mühendisliği

Yüksek Lisans Programı

EYLÜL 2017

TÜRK HAVA KURUMU ÜNİVERSİTESİ
FEN BİLİMLER ENSTİTÜSÜ

DERİN ÖĞRENME YÖNTEMLERİ İLE GÖĞÜS KANSERİ TEŞHİSİ

YÜKSEK LİSANS TEZİ

ALI MAHMOOD OGUR ANWER

1403610004

Elektrik ve Bilgisayar Mühendisliği

Yüksek Lisans Programı

Tez Danışmanı: Yrd. Doç. Dr. Tansel DÖKEROĞLU

Türk Hava Kurumu Üniversitesi Fen Bilimler, Enstitüsü'nün 1403610004 numaralı Yüksek Lisans öğrencisi, Alı Mahmood Ogur ANWER ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı "Derin Öğrenme Yöntemleri ile göğüs Kanser Teşhisi" başlıklı tezini, aşağıda imzaları olan jüri önünde başarı ile sunmuştur.

Tez Danışmanı : Yrd. Doç. Dr. Tansel DÖKEROĞLU
Türk Hava Kurumu Üniversitesi

Jüri Üyeleri : Yrd. Doç. Dr. Meltem İMAMOĞLU
Türk Hava Kurumu Üniversitesi

: Yrd. Doç. Dr. Abdül Kadir GÖRÜR
Çankaya Üniversitesi

: Yrd. Doç. Dr. Tansel DÖKEROĞLU
Türk Hava Kurumu Üniversitesi

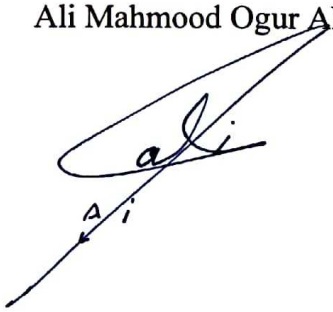
Tez Savunma Tarihi: 12 Eylül 2017

TÜRK HAVA KURUMU ÜNİVERSİTESİ
FEN BİLİMLER ENSTİTÜSÜ MÜDÜRLÜĞÜ'NE

Yüksek Lisans/Doktora Tezi olarak sunduğum, "Derin Öğrenme Yöntemleri ile göğüs Kanser Teşhisi" adlı çalışmamın, tarafımdan akademik etik ve kurallara aykırı düşecek bir yardıma başvurmaksızın yazıldığını ve yararlandığım kaynakların kaynakçada gösterilenlerden oluştuğunu, bunlara atıf yapılarak yararlanılmış olduğunu belirtir ve bunu onurumla doğrularım.

12.09.2017

Ali Mahmood Ogur ANWER



ÖNSÖZ

Yüksek Lisans tez çalışma sürecinde beni yönlendiren, karşılaştığım zorlukları bilgi ve tecrübesi ile aşmamda yardımcı olan tez danışmanım değerli Yrd. Doç. Dr. Tansel Dökeroğlu'na teşekkürlerimi sunarım.

Çalışmalarımın sürecinde her zaman yanımda olan babama, anneme, Aytekin abime, eşime ve tüm aileme teşekkürlerimi sunarım.

Eylül 2017 Ali Mahmood Ogur ANWER

İÇİNDEKİLER

TABLO LİSTESİ

ŞEKİL LİSTESİ

KISALTMALAR

API	:	Application Programming Interface
CNN	:	Convolution Neural Network
DBN	:	Deep Belief Networks
DCNN	:	Deep Convolution Neural Network
DN	:	Doğru Negatifler
DP	:	Doğru Pozitifler
FPR	:	Yanlış Pozitif Oran
FRE	:	Recursive Feature Elimination
GRU	:	Gated Recurrent Unit
LSTM	:	Long Short-Term Memory
MLPNN	:	Multilayer Perceptron Neural Network
NHG	:	Nesterov Hızlandırılmış Gradyan
PCA	:	Principal Component Analysis
PNN	:	Probabilistic Neural Network
RNN	:	Recurrent Neural Network
SGİ	:	Stokastik Gradyan İnişi
SRN	:	Simple Recurrent Network
SVM	:	Support Vector Machine
TPR	:	Doğru Pozitif Oran
WBCD	:	Wisconsin Breast Cancer Database
WHAVE	:	Weighted Hierarchical Adaptive Voting Ensemble
YN	:	Yanlış Negatifler
YP	:	Yanlış Pozitifler

ÖZET

DERİN ÖĞRENME YÖNTEMLERİ İLE GÖĞÜS KANSERİ TEŞHİSİ

ANWER, Ali Mahmood Ogur
Yüksek Lisans, Elektrik ve Bilgisayar Mühendisliği Bölümü
Tez Danışman: Yrd. Doç. Dr. Tansel DOKEROGLU
Eylül - 2017, 92 sayfa

Göğüs kanseri dünyada kadınların arasında en yaygın kanser türü olup, kadınlardaki kanser oranının %23'ünü kapsamaktadır. Normalde memenin hücreleri düzenlenmiş bir şekilde bölünürler. Eğer hücreler, yeni hücrelere ihtiyaç duyulmadığında bölünmeyi sürdürürlerse, bir doku kütlesi oluşur. Bu kütleye bir tümör denir. Bu tümör habis veya iyi huylu olabilir. Teşhisinin amacı habis ve iyi huylu hücreleri birbirinden ayırmaktır. Göğüs kanserinden kurtulmanın tek umudu erken teşhistir. Bu tezde, derin öğrenme tıp alanında kullanılmaktadır, göğüs kanseri teşhis için derin öğrenme tabanlı klinik destek sistemi önermektedir. Derin öğrenme, yapay zekâ problemlerini çözmek için makine öğrenmenin bir alt alanıdır. Bu çalışma, derin öğrenme yaklaşımlarının tıbbi alanda da başarılı sonuçlar üretip üretemeyeceğini araştırmayı amaçlamakta olup, göğüs kanseri türündeki çalışmaları gösterecektir. Göğüs kanserinin teşhisinde daha önce derin öğrenme yöntemleri Tekrarlayan sinir ağları ve Tam bağlantılı sinir ağları kullanılırken yeni sıra ilk defa farklı derin öğrenme tekniklerinden Konvolüsyon sinir ağlarının yeni mimarisi kullanılmıştır. Wisconsin UCI makine öğrenme deposundaki göğüs kanseri veri setleri farklı derin öğrenme yöntemlerinin yeteneğini test etmek için kullanılmaktadır. Sonuçlar, derin öğrenme yaklaşımlarının, tıbbi karar vermeyi destekleme yönünde umut verici bir yön gösterdiğini doğrulamaktadır.

Anahtar Kelimeler: Derin Öğrenme, Göğüs Kanseri, Tam bağlantılı yapay sinir ağları, Tekrarlayan Sinir Ağı, Konvolüsyon Sinir Ağı.

ABSTRACT

BREAST CANSER DIAGNOSIS USING DEEP LEARNING METHODS

ANWER, Ali Mahmood Ogur
Master, Department of Electrical and Computer Engineering
Thesis supervisor: Asst. Prof. Dr. Tansel DOKEROGLU
September - 2017, 92 pages

Breast cancer is the most well-known cancer of women over the world, involving 23% of every female cancer around the world. Normally, the cells of the breast divide in a regulated manner. If cells keep on dividing when new cells are not needed, a mass of tissue forms. This mass is called a tumor. This tumor can be malignant or benign. The goal of diagnosis is to distinguish between malignant and benign cells. The only hope of surviving from breast cancer is early detection. In this thesis, deep learning is used in medical filed, to propose deep learning based clinical support system for diagnosis breast cancer. Deep learning is a subfield of machine learning to solve problems of artificial intelligence. This study aims to investigate whether the deep learning approaches are also capable of producing successful results in the medical area. Many of deep learning algorithms have been used in the diagnosis of breast cancer, such as fully connected neural networks and recurrent neural network, the new order is the first time that the new architecture of convolution neural networks is used to diagnosis breast cancer. Wisconsin Breast Cancer datasets from the UCI Machine Learning repository is used to test the ability of deep learning different techniques. The experiments with deep learning approaches show a promising direction towards supporting medical decision making.

Keywords: Deep Learning, Breast Cancer, Fully Connected Neural Networks, Recurrent Neural Network, Convolution Neural Network.

BİRİNCİ BÖLÜM

GİRİŞ VE PROBLEM TANIMI

1.1 Araştırma Motivasyonu

Türkiye'de ve dünyada, insanlar sınırlı sayıdaki tıbbi kaynaklardan ve medikal malzemelerden yararlanmak için uzun süre beklemekten acı çekiyorlar. Dünya Sağlık Örgütü'ne (WHO) göre, Türkiye sağlık sistemi alanında 190 ülkeden 70'inci olarak sıralanmıştır [1].

Türkiye'nin artan nüfusu, yaşlanan nüfus, modern yaşam tarzı, iklim değişikliği ve ortaya çıkan yeni hastalıklar. Türkiye sağlık kuruluşlarını ve eyalet hükümetlerini yeni prosedürler koymak ve planlamalara yöneltti. Bu kuruluşlar mevcut tıbbi kaynaklar, altyapı ve sağlık personelindeki eksikliklere rağmen halka iyi bir sağlık hizmeti sunmaktır. Buna ek olarak, tıbbi hizmetler tüm bireyler için vazgeçilmezdir ve tüm sakinler ve vatandaşlar için tıbbi altyapı ve hizmetleri geliştirmek ve korumak ulusun sorumluluğundadır. Tıbbi personeldeki ve teknolojideki eksikliklere ek olarak, reçetelendirme hatası olayları, hastalar için küçük sorunları daha büyük sorunlar haline getirmektedir.

Dünya sağlık Örgütü’ne (WHO) göre, göğüs kanseri şu anda dünyadaki tüm kadınlar arasında en yüksek kanser türüdür ve göğüs kanseri kadınlarda ikinci en yüksek ölüm nedenidir. Türkiye’de göğüs kanseri, kanser türleri arasında üçüncü yaygın kanser türüdür ve genel olarak dokuzuncu ölüm nedeni olarak bilinir (Bakınız Tablo 1.1, 1.2) [2].

Tablo 1.1: Türüne göre Türkiye'de kanser sıralaması [2]

Türüne göre Türkiye'de kanser sıralaması					
Sıra	Kanser	Dünya Sıra.	Sıra	Kanser	Dünya Sıra.
1	Lung Cancers	17	9	Pancreas Cancer	66
2	Prostate Cancer	38	10	Ovary Cancer	90
3	Breast Cancer	113	11	Oesophagus Cancer	69
4	Stomach Cancer	32	12	Liver Cancer	149
5	Colon-Rectum Cancers	66	13	Uterine Cancer	63
6	Lymphomas	14	14	Cervical Cancer	159
7	Bladder Cancer	4	15	Oral Cancer	137
8	Leukemia	45	16	Skin Cancers	116

Tablo 1.2: Türkiye'de en yüksek ölüm nedenleri [2]

Türkiye'de en yüksek ölüm nedenleri					
Sıra	Ölüm nedeni	Dünya Sıra.	Sıra	Ölüm nedeni	Dünya Sıra.
1	Coronary Heart Disease	37	13	Asthma	39
2	Stroke	82	14	Colon-Rectum Cancers	66
3	Lung Cancers	17	15	Congenital Anomalies	67
4	Lung Disease	42	16	Other Injuries	107
5	Prostate Cancer	38	17	Suicide	94
6	Rheumatic Heart Disease	4	18	Lymphomas	14
7	Hypertension	93	19	Meningitis	47
8	Influenza and Pneumonia	139	20	Bladder Cancer	4
9	Breast Cancer	113	21	Low Birth Weight	99
10	Diabetes Mellitus	132	22	Inflammatory/Heart	124
11	Stomach Cancer	32	23	Leukemia	45
12	Road Traffic Accidents	118	24	Liver Disease	158

Irak'ta göğüs kanseri, kanser türleri arasında birinci yaygın kanser türüdür ve genel olarak dokuzuncu ölüm nedeni olarak bilinir (Bakınız Tablo 1.3, 1.4) [2].

Tablo 1.3: Türüne göre Irak'ta kanser sıralaması [2]

Türüne göre Irak'ta kanser sıralaması					
Sıra	Kanser	Dünya Sıra.	Sıra	Kanser	Dünya Sıra.
1	Breast Cancer	44	9	Liver Cancer	116
2	Lung Cancers	81	10	Ovary Cancer	105
3	Lymphomas	13	11	Pancreas Cancer	96
4	Bladder Cancer	3	12	Oral Cancer	103
5	Leukemia	1	13	Cervical Cancer	162
6	Prostate Cancer	144	14	Oesophagus Cancer	138
7	Colon-Rectum Cancers	116	15	Skin Cancers	143
8	Stomach Cancer	96	16	Uterin Cancer	164

Tablo 1.4: Irak'ta en yüksek ölüm nedenleri [2]

Irak'ta en yüksek ölüm nedenleri					
Sıra	Ölüm nedeni	Dünya Sıra.	Sıra	Ölüm nedeni	Dünya Sıra.
1	Coronary Heart Disease	22	13	Low Birth Weight	58
2	Stroke	57	14	Congenital Anomalies	38
3	Road Traffic Accidents	2	15	Rheumatic Heart Disease	17
4	Diabetes Mellitus	47	16	Violence	52
5	War	4	17	Inflammatory/Heart	61
6	Kidney Disease	5	18	Birth Trauma	60
7	Other Injuries	13	19	Asthma	55
8	Hypertension	28	20	Other Neoplasms	4
9	Influenza and Pneumonia	101	21	Lymphomas	13
10	Breast Cancer	44	22	Bladder Cancer	3
11	Lung Disease	114	23	Endocrine Disorders	73
12	Lung Cancers	81	24	Leukemia	1

Etkili ve koruyucu bir önlem yaygınlaşıncaya kadar, erken tanı ve ardından etkili bir tedavi, göğüs kanserinden ölüm sonucunu azaltmanın tek yoludur. Göğüs kanseri, hasta tarafından göğüste yumru kitleleri olarak algılanır. Bu kitleler genel olarak iyi huyludur (kanser değil), bu nedenle göğüs kanserini teşhis etmek doktorun sorumluluğundadır. Teşhisin amacı malign (kanserli) ya da benign göğüs kitlesini ayırt etmektir. Göğüs kanseri, hasta üzerinde yapılan çeşitli testlerin sonuçlarını analiz ederek hastalık teşhisi gerçekleştirilir. Testler yapıldıktan sonra, nihai teşhis

tıbbi bir uzman için bile zor olabilir. Son yıllarda, doktorlar test sonuçlarının verilerini paylaşmaya başladılar ve bu paylaşım bilgisayarlı tanısal sistemlerin geliştirmesine yol açtı.

Bu araştırma motivasyonunun amacı, testlerden elde edilen sonuçları sınıflandırmak için derin sinir ağ tabanlı bir araç yaratmaktır. Bu araştırmada önerilen araç tamamen klinik destek sistemidir. Bu sistem teşhis için tıbbi veri analizini daha kısa sürede sağlamakta ve deneyimsizlik veya yorgunluk nedeniyle oluşan insan hatalarını engellemektedir. Derin sinir ağlarının kullanımı, yöntemlerin çoğunun doğruluğunu arttırır ve bu süreç içerisinde ihtiyaç duyulan uzman ihtiyacını azaltır.

1.2 Araştırma Hedefleri

Makine öğrenme araçları, doktorların erken evrelerde hastalıkların teşhis edilmesinde ve öngörülmesinde yardımcı olabilecek uzman sistemler geliştirerek, sağlık bakım problemlerinin çözümünde önemli ölçüde yardımcı olabilir. Bu sistemler; maliyeti, bekleme süresini ve serbest insan uzmanlarının daha fazla araştırma süresini azaltabilir ve aynı zamanda tıbbi personel tarafından yapılan yanlışlıkları ve hataları da azaltabilir [3]. Bilgisayar destekli tasarım (CAD) ve tıbbi uzman sistemleri ve araçları, medikal tanı alanında en önde gelen araştırma alanlarından biri haline gelmiştir. Bilgisayar destekli tasarımın(CAD) amacı, daha doğru teşhis konusunu etkili bir şekilde sağlamak için insan uzmanlığını ve teknoloji bilgisini birleştiren bir uzman sistem tasarlamaktır [3]. CAD, doktorları hastalıkların teşhisi ve tahmini konusunda yardımcı olmak için kullanılabilir. Bunun sonucunda, doktorlar, ölüm olasılığı dâhil olmak üzere can kaybını önlemek için en kısa sürede gerekli tedaviyi sağlayabilirler.

Bu çalışmanın amacı, derin öğrenme algoritmalarını gösteren bir sistem tasarlamaktır. Bu sistem veri setindeki çok sayıda özellik ile çalışabilir ve bu sistem göğüs kanser teşhisini de yapabilir.

1.3 Giriş

Göğüs kanseri, kadınlar arasında kanser ölümlerinin başta gelen ölüm sebebidir. Göğüs kanseri kadınlarda genel olarak 40 yaşından sonra görülür, ancak yüksek risk özelliklerine sahip bazı kadınlar genç yaşta meme kanseri olabilir. Göğüs

kanseri insanlarda ve diğer Memeli hayvanlarda görülür. Göğüs kanseri, genel olarak kadınlarda olur, ancak erkeklerde de olabilir [4]. Kanser, hücrelerin anormal hale geldiği bir hastalıktır. Kanser, hücrelerin anormal hale geldiği ve kontrolsüz bir şekilde daha fazla hücre oluşturduğu bir hastalıktır. Kanser göğüsleri oluşturan dokularda başlar. Göğüs; loblar, lobüller ve kanallar ile birbirine bağlanan haznelerden oluşur. Göğüs ayrıca kan ve lenf damarları da içerir. Bu lenf damarları, lenf düğümleri adı verilen yapılara sebep olur. Kol altında, köprücük kemiğinin üstünde, göğüste ve vücudun diğer bölümlerinde lenf düğümleri kümeleri bulunur. Bunlarla birlikte, lenf damarları, lenf düğümleri ve lenf sistemini oluşturur ki bu vücutta lenf denilen sıvıyı dolaştırır. Lenf, enfeksiyon ve hastalıklarla mücadelede yardımcı olan hücreleri içerir. Normalde göğüs hücreleri düzenlenmiş bir şekilde bölünürler. Eğer hücreler, yeni hücrelere ihtiyaç duyulmadığında da bölünmeyi sürdürürlerse, orada bir doku kütlesi oluşur.

Bir tümör iyi huylu ya da kötü huylu olabilir. İyi huylu bir tümör kanser değildir ve vücudun diğer bölümlerine yayılmaz. Kötü huylu tümörler hücreleri bölünürler ve etrafındaki dokulara zarar verirler. Göğüs kanseri memenin dışında yayılırken, kanser hücreleri çoğunlukla lenf düğümleri kol altında bulunur. Herhangi bir durumda, kanser lenf bezlerine ulaştıysa, kanser hücreleri lenfatik sistem yoluyla veya kan dolaşımı yoluyla vücudun diğer bölgelerine de yayılabilir.

1.4 Göğüs Kanseri Belirtileri

Göğüs kanserinin ilk belirgin işareti tipik olarak göğüs dokusunun geri kalanından farklı hissettiren bir yumru şeklindedir. Kadın göğsünde bir yumru hissettiği zaman göğüs kanseri vakalarının %80'inden daha fazlası keşfedilir. Koltukaltı bölgelerinde bulunan lenf düğümlerinde bulunan şişlikler meme kanseri olarak da gösterilebilir [5]. Bir yumru dışındaki göstergeler diğer göğüs dokusundan farklı kalınlaşma içerebilir, bir göğüs daha büyük veya daha küçük olur. Meme başı pozisyonu veya şekli değişir ya da ters çevrilir, cilt kırışıklığı veya çukurlaşma, meme başında veya etrafında döküntü, meme başında veya başlarında boşalma memenin veya koltuk altının bir kısmında sürekli ağrı ve koltuk altında ya da köprücük kemiği çevresinde şişlik. İnflamatuvar göğüs kanseri önemli bir tanı koyma gücü oluşturabilen bir göğüs kanseridir. Semptomlar göğüs iltihaplarına benzeyebilir ve göğüste kaşıntı, ağrı, şişme, meme başı dönmesi, ateş, kızarıklık ve ciltte portakal

kabuğu gibi doku içerebilir. Göğüs kanserinden beliren diğer bir belirti kompleksi ise memenin paget hastalığıdır. Bu sendrom, kızarıklık ve meme ucu cildinin yumuşak pullanması gibi egzama töz cilt değişiklikleri gösterebilir. Paget'in ilerlemesi gibi semptomlar karıncalanma, kaşınma, aşırı hassasiyet, yanma ve ağrı içerebilir. Ayrıca meme ucundan boşaltma olabilir. Paget teşhis konan kadınların yaklaşık yarısında göğüste bir yumru da bulunur.

1.5 Göğüs kanser teşhisi

Göğüs kanseri hasta tarafından kitle olarak algılanır. Tanı amacı malign (kanserli) ve iyi huylu meme yumrularını ayırt etmektir. Göğüs kanseri teşhisi için sık kullanılan yöntemler mamografi, ince iğne aspirasyonu (FNA) ve cerrahi biyopsi [3]. Mamografi %68 ve %79 arasında teşhis hassasiyeti göstermektedir [6]. İnce bir iğne aspiratı alarak mikroskop altında görsel olarak incelemektir, %65 ile %98 arasında değişen bir duyarlılığa sahiptir [7].

En riskli ve masraflı yöntem cerrahi biyopsidir ve %100'e yakın hassasiyeti vardır. Fakat, mamografi duyarlılığı azdır, FNA duyarlılığı bazen iyi bazen kötüdür, Cerrahi biyopsi, doğru olmasına rağmen zaman alıcı ve masraflıdır [3]. Araştırmamızın tanısal yönünün amacı, FNA'dan elde edilen Wisconsin Meme kanseri veri tabanı yardımıyla meme kanseri teşhisi yapan bir derin sinir ağı sistemi geliştirmektir.

Birinci bölümde ilk olarak bu çalışmasının motivasyonunu ve hedeflerini tanımlamaktadır. Daha sonra göğüs kanseri hastalığını, semptom ve teşhisini ayrıntılı olarak açıklamaktadır. İkinci bölümde göğüs kanseri teşhisi için önceden kullanılan farklı makine öğrenme ve derin öğrenme yöntemleri açıklamaktadır. Üçüncü bölümde ilk olarak yapay sinir ağlarını ayrıntılı olarak anlatılmıştır. Daha sonra derin öğrenme tanımını ve derin öğrenme algoritmalarını ayrıntılı olarak açıklamaktadır. Dördüncü bölümde ilk olarak derin öğrenme testlerini yapa bilmek için kullandığımız materyalleri açıklamaktayız. Daha sonra kullandığımız WBCD veri setini analiz edip ve temizliyoruz ve son olarak değişik derin öğrenme ve makine öğrenme yöntemlerini kullanarak göğüs kanserini teşhis etmekteyiz. Son bölümde çalışmamızda kullanılan sonuçları kontrol edip ve değerlendirme yapmaktayız.

İKİNCİ BÖLÜM

ÖNCEKİ ÇALIŞMALAR

Göğüs kanseri teşhisi için çok sayıda makine öğrenme tekniği uygulanmış ve sonuçları sunulmuştur. Birçoğunun doğruluğu, UCI makine öğrenme havuzundan alınan veri kümesini kullanılarak değerlendirildi. Bu tekniklerin doğruluğu %99,8'e kadar çıkmaktadır.

Übeyli, çalışmasında göğüs kanseri teşhisi için farklı sınıflandırma makine öğrenme yöntemleri kullandı [8]. Bu çalışmada WBCD veri seti test için kullanıldı. Multilayer Perceptron Neural Network (MLPNN) yöntemin bulunan sınıflandırma doğruluğu %91,92, Probabilistic Neural Network (PNN) yöntemini kullanarak bulunan sınıflandırma doğruluk oranı %98,15, Recurrent Neural Network (RNN) yöntemini kullanarak bulunan sınıflandırma doğruluk oranı ise %98,61.

Bin Mohd Azmi & Che Cob, çalışmalarında göğüs kanserini teşhisi için Feed-forward Backpropagation algoritması ile sinir ağı yöntemini kullanarak bir sistem sundular Çalışmada WBCD veri seti test için kullanıldı, bu yaklaşımı kullanarak %98,22 doğruluk oranı bulundu [9].

Khosravi, Addeh, & Ganjipour, çalışmalarında üç ana modülü içeren göğüs kanseri tümörlerini tanımak için bir melez sistem sunmaktadır [10]. Bu modüller, öznitelik bulma modülü, eğitim modülü ve sınıflandırma modülüdür. Öznitelik bulma modülünde, bulanık öznitelik etkili bir sınıflandırıcı girişi olarak kullanılmıştır. Eğitim modülünde sınıflandırıcıyı eğitmek için hybrid beis algoritma (BA) ve back-propagation (BP) algoritmaları kullanılmıştır. Sınıflandırma modülünde ise, çok katmanlı algılayıcı (MLP) sinir ağı kullanılır. Çalışmada WBCD veri seti test için kullanıldı. Bu yaklaşımı kullanarak %98,22 doğruluk oranı bulundu.

Yang, Peng, & Shi, çalışmalarında isometric feature mapping (Isomap) yöntemine dayalı olan multiple kernels yöntemiyle SVM yöntemini kullanarak göğüs kanseri teşhisi için etkin bir yaklaşım sundular [11]. Bu çalışmada WBCD veri seti test

için kullanıldı. İlk adım olarak, Isomap yöntemi yüksek boyutlu göğüs kanseri verilerini daha düşük boyutlu bir alanına yansıttı. İkinci adımda, multiple kernels yöntemiyle SVM yöntemi daha düşük boyutlu göğüs kanseri verilerinin sınıflandırmasını yapıldı. Bu yaklaşımı kullanarak %98.22 doğruluk oranı bulundu.

Phetlasy, Ohzahata, Wu, & Kato, çalışmalarında iki farklı sınıflandırma algoritması birleştirerek göğüs kanseri teşhisi için melez yöntem sundular [12]. İlk sınıflandırma algoritması yanlış negatifi azaltmaya yanıt verir iken, ikinci sınıflandırma algoritması yanlış pozitifliği azaltmaya yanıt verdi. Bu çalışmada WBCD veri seti test için kullanıldı. Bu yaklaşımı kullanarak %99,13 doğruluk oranı bulundu.

Deng & Perkowski, çalışmalarında göğüs kanseri teşhisi için yeni makine öğrenme yöntemi tasarladılar, Weighted Hierarchical Adaptive Voting Ensemble (WHAVE) [13]. Üçüncü yöntem de ayrı makine öğrenme yöntemi kullanılarak inşa edilmiştir. Decision Tree, Naive Bayes classifier ve SVM. Bu çalışmada WBCD veri seti test için kullanıldı. WHAVE yöntemini kullanarak bulunan sınıflandırma doğruluk oranı %99,80' dir.

Yin, Fei, Yang, & Chen, çalışmalarında SVM, RFE ve PCA makine öğrenme yöntemlerini birleştirerek, göğüs kanseri teşhisi için bir yöntem kullandılar [14]. SVM, RFE ve PCA özelliklerin boyutunu etkili bir şekilde azalttı ve en ilgili özellikleri çıkardılar. Onlara göre, sınıflandırma doğruluğu ve hesaplama hızı iyileştirildi. Bu çalışmada, WBCD veri seti test için kullanıldı. SVM, RFE ve PCA yöntemini kullanarak bulunan sınıflandırma doğruluk oranı %99,58.

Bazazeh & Shubair, çalışmalarında göğüs kanseri tespiti ve teşhisi için kullanılan en popüler üç makine öğrenme tekniklerini karşılaştırdı [15]. SVM yöntemini kullanarak bulunan sınıflandırma doğruluk oranı %96,60, Random Forest (RF) yöntemini kullanarak bulunan sınıflandırma doğruluk oranı %99,90 ve Bayesian Networks (BN) yöntemini kullanarak bulunan sınıflandırma doğruluk oranı %99,10. Bu çalışmada WBCD veri seti test için kullanıldı.

Singh, Sanwal, & Praveen, çalışmalarında göğüs kanseri teşhisi için bir yaklaşım sundular. İki genetik algoritma kullanarak, Inter-Genetic Algorithm ve Intra-Genetic Algorithm, bu yaklaşımı kullanarak %99,90 sınıflandırma doğruluk oranı bulundu. Bu çalışmada WBCD veri seti test için kullanıldı [16].

Jhajharia, Varshney, Verma, & Kumar, çalışmalarında Çok değişkenli istatistik yaklaşımını yapay zekâya dayalı öğrenme tekniğiyle birleştirerek göğüs kanseri

teşhisi için bir tahmin uygulaması sundular [17]. Yapay sinir ağı eğitmek için PCA veriyi ön işleme koyar ve bu özellikleri en ilgili biçimde çıkarır ve bu, eğitmek yeni örneklerin sınıflandırılması verilerindeki modelleri çıkarır. Bu yaklaşım kullanılarak %98,39 sınıflandırma doğruluk oranı bulundu. Bu çalışmada WBCD veri seti test için kullanıldı.

Biyomedikal ve mikroarray verileri üzerine derin öğrenme yaklaşımlarının uygulanmasını kullanarak yapılan çalışmalar son yıllarda çok arttı ve çok büyük başarılar göstermektedir.

Dhungel, Carneiro, & P. Bradley, çalışmalarında derin öğrenme yöntemlerini ve random forecast sınıflandırma yöntemini kullanarak mamografide kitleleri tespit etmek için bir yaklaşım sundular [18]. Birinci aşamalı sınıflandırıcı, iki seviyeli derin CNN kasko tarafından işlenmek üzere şüpheli bölgeleri seçen çok ölçekli bir DBN'den oluşur. Bu derin öğrenme analizinden sonra kalan bölgeler, daha sonra, seçilen bölgelerden elde edilen morfolojik ve doku özelliklerini kullanan iki seviyeli random forecast sınıflandırıcıları tarafından işlenir. Bu yaklaşım, yüksek doğru pozitif algılama oranını korurken, sahte pozitif bölgelerin azaltılmasında etkilediğini göstermektedir.

Chen, Xu, Wong, Wong, & Liu, çalışmalarında derin öğrenme tekniği CCN yöntemini kullanarak otomatik glokom tanısı için bir model sundular [19]. Önerilen model mimarisi, altı adet öğrenilmiş kat içermektedir ve bunlar: dört CNN katmanı ve iki tam bağlantılı katman. ORIGIN ve SCES veri setleri üzerinde test yapılmıştır. Bu yöntemi kullanarak doğruluk oranı %88 olarak bulunmuştur.

Khademi & Nedialkov, çalışmalarında Göğüs kanseri tahmini ve tanısı için probabilistic graphical modelini (PGM) önerdiler. Gen ekspresyonu özelliklerinin boyutsallarını azaltmak ve örnekleri sınıflandırmak için manifold learning ve DBN yöntemlerini uyguladılar [20]. Farklı veri setleri kullanmak, tümörlerin sınıflandırılması ve rekürrens ve metastaz gibi olayların tahmininde umut verici sonuçlar vermektedir.

Suzuki, ve diğerleri, çalışmalarında derin öğrenme yöntemlerini kullanarak mamografide kitleleri tespit etmek için farklı bir yaklaşım sundular [21]. Çalışmada, 5 konvolüasyon katmanı ve 3 tam bağlantılı katmanı içeren ağırlıklı 8 kattan oluşan bir DCNN mimarisini denediler. İlk aşamada 1.000 sınıf sınıflandırması için 1,2 milyon doğal görüntü kullanarak DCNN'yi eğitiler. Ardından, iki sınıf tanımlama

için: kitle ve normal, 1,656 mamografi görüntüsünde ilgi bölgeleri kullanarak DCNN yeniden eğitildi. Algılama testi 99 kitle görüntüsü ve 99 normal görüntü ve toplamda 198 mamografi görüntüleri üzerinde gerçekleştirilmiştir. Bu çalışmada, kitle sınıflandırma doğruluğu %89,90 bulundu. Bu sonuçlar, DCNN'nin mamografi kitle algılama bilgisayar destekli tanı sistemleri için anahtar bir yaklaşım olma potansiyeline sahip olduğunu göstermiştir.

Sarraf & Tofighi, çalışmalarında, beyin hastalıklarının teşhisinde ve klinik karar vermede derin öğrenmenin avantajlarını göstermek için bir örnek olarak Alzheimer Hastalığı (AH) kullanılmıştır [22]. Alzheimer beynini normal ve sağlıklı bir beyinden ayırt etmek için CNN yöntemini kullandılar. Bu tür tıbbi verilerin sınıflandırılmasının önemi, normal kişilerle karşılaştırıldığında Alzheimer hastalığının semptomlarını tanımak ve hastalığın evrelerini tahmin etmek için bir tahmini model veya sistem geliştirme potansiyeli yatmaktadır. Alzheimer hastalığı gibi klinik verilerin sınıflandırılması her zaman zor olmuştur ve en sorunlu yönü güçlü ayrımcılık özelliklerini seçmek olmuştur. Bu yöntemi kullanarak sınıflandırma doğruluk oranı %96,85 bulunmaktadır.

Al-Fatlawi, Jabardi, & Ling, çalışmalarında, Parkinson hastalığını teşhis etmek için etkili bir teknik olarak DBN yöntemi kullanmaktadırlar [23]. Bu teşhis hastaların konuşma sinyaline dayanarak kurulmuştur. Konuşma sinyalinin ayırt edilmesi ve analizi yoluyla, DBN Parkinson hastalığını teşhis etme kabiliyetine sahiptir. DBN kullanarak Parkinson hastalığının teşhisini gerçekleştirmek için önerilen sistem hasta ve sağlıklı insanların sesleriyle eğitilmiş ve test edilmiştir. Önerilen DBN, iki Restricted Boltzmann Machines (RBM) ve bir çıktı tabakası içermektedir. Bu yöntemi kullanarak doğruluk oranı %94 bulunmuştur.

M. Abdel-Zaher & M. Eldeib, çalışmalarında göğüs kanserini tespit etmek için DBN yöntemine dayalı olarak bir otomatik tanı sistemi sundular [24]. Bu çalışmada WBCD veri seti test için kullanıldı. Bu yaklaşımı kullanarak %98,68 sınıflandırma doğruluk oranı bulundu.

Syafiandini, Wasito, Yazid, Fitriawan, & Amien, çalışmalarında Kolorektal karsinom kanserinin iki alt türünü tanımlamak için derin öğrenme yaklaşımını kullanan bir yaklaşım sundular [25]. Bu tanımlama işlemi, gen ekspresyonu ve klinik verilerini entegrasyonunu yaparak kullanılır. Birinci derin öğrenme mimarisi, Deep Boltzmann Machines (DBM) veri entegrasyon işlemi için kullanılır. Gen

ekspresyonu ve klinik verileri daha sonra Restricted Boltzmann Makineleri (RBM) girişi olarak kanser alt tipi tanımlaması için kullanılır. Kaplan Meier survival analysis tanımlama sonucunu değerlendirmek için kullanılır. Önerilen yaklaşım, veri setinde iki kanser alt tipi olan karsinom kolorektal olduğunu kanıtlamaktadır.

Shimizu, çalışmasında Derin öğrenmeye dayalı bir akciğer kanseri teşhis sistemi kurmayı sundular. Bu sistem Gas Chromatography Mass Spectrometer (GC-MS) tarafından oluşturuldu ve sistem giriş verisi olarak insan idrarı kullanıldı [26]. Sistem hastanın akciğer kanseri olup olmadığını algılama doğruluk oranı 90% olarak bulundu. Bu çalışmada, derin öğrenme insan hayatı ile ilgili verilerin analizinde de etkili olduğunu kanıtladı ve herhangi bir özel tıbbi bilgi olmadan ön tanı yapabileceği bilinmektedir.

Nasr-Esfahani, çalışmasında Melunuma lezyonlarını tespit etmek için derin öğrenme yöntemlerini kullanarak bir uygulama önermektedirler [27]. Deri hastalıkları uzmanları, deri kanserinin erken teşhisinde yardımcı olabilecek bu uygulamayı ve klinik görüntülerini kullanır. Bu uygulamada derin öğrenme modellerinin bir üyesi olan CNN kullanıldı. CNN, malign ve benign olguları arasında ayrım yapmaktadır. Deneysel sonuçlar, önerilen bu yöntemin çok başarılı olduğunu göstermektedir.

Silva, Neto, Silva, Paiva, & Gattass, çalışmalarında akciğer nodüllerinin şekil ve doku özelliklerini hesaplamadan, akciğer nodüllerinin malign veya benign olarak sınıflandırmak için genetik algoritmayla birlikte derin öğrenme tekniğinin bir üyesi olan CNN tekniğini kullanarak bir metot sundular [28]. Bu çalışmada, Lung Image Database Consortium and Image Database Resource Initiative (LIDC-IDRI) veri setinden alınan bilgisayarlı tomografi (BT) görüntülerini kullanarak test edildi. Bu çalışmada, sınıflandırma doğruluğu %94.78 olarak bulundu.

ÜÇÜNCÜ BÖLÜM

DERİN ÖĞRENME ALGORİTMALARI

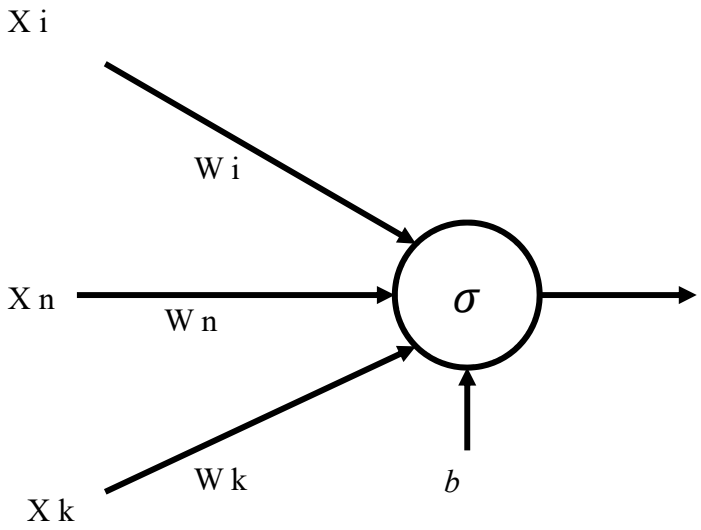
3.1 Yapay sinir Ağınlarına temel giriş

3.1.1 Yapay Sinir Ağları

Yapay Sinir Ağları, yapay zekâ tekniklerinden biridir. İnsandaki ve hayvandaki biyolojik sinir ağlarıyla bazı özellikleri paylaşan bir hesaplama modelidir. Pek çok basit ünite, merkezi kontrol ünitesi olmadan paralel olarak çalışmaktadır. Yapay sinir ağlarının davranışı, ağ mimarisi tarafından şekillendirilmiştir. Nöron sayısı, katman sayısı ve katmanlar arasındaki bağlantı türleri ağ mimarisini tanımlar.

3.1.2 Yapay Nöron

İnsan beyninin temel birimi nörondur. Beynin pirinç tanesi büyüklüğündeki bir parçası, yaklaşık olarak 10.000'den fazla nöron içermektedir. Bu nöronların her biri diğer nöronlar ile ortalama 6,000 bağlantı oluşturmaktadır. Bu büyük biyolojik ağ, çevremizdeki her şeyi anlamamızı sağlamaktadır. Yapay bir nöron; girdi, bias ve çıktıdan oluşur. Çıkış fonksiyonu herhangi bir fonksiyon olabilir ancak genelde sigmoid output veya ReLU tercih edilir. Şekil 3.1 yapay bir nöronu ve bu nöronun bağlantılarını göstermektedir.

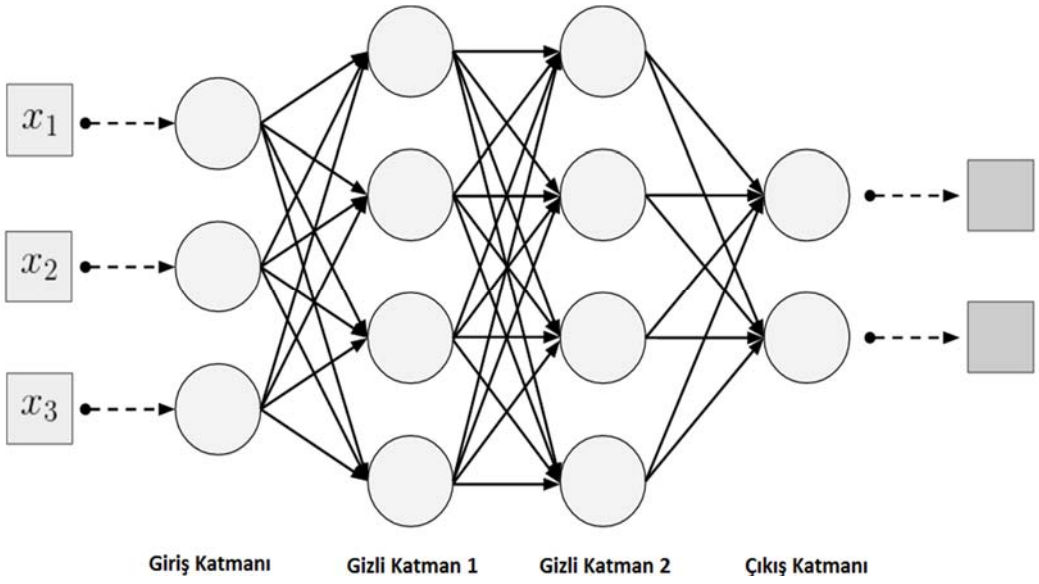


Şekil 3.1: Yapay Nöron

Bir nöronun çıktısı, girdilerin ağırlıkları ve bias toplamının bir fonksiyonu olarak hesaplanır. Eğer ağırlıklı toplamı $z = \sum jwjxj + b$ olarak ve nöron değerlendirme fonksiyonu σ olarak ele alınırsa, o zaman nöron çıkışı $a = \sigma(z)$ olarak hesaplanır.

3.1.3 Çok Katmanlı Yapay Sinir Ağları

İnsan beynindeki nöronlar katmanlar halinde düzenlenir. Aslında, insan beyni korteksi (insanın zekasına sorumlu olan yapı) altı katmandan oluşur. Bilgi bir tabakadan diğerine aktarılır ve duyusal girdi kavramsal anlayışa dönüştürülür. Örnek olarak, görsel korteksin en alt tabakası gözlerden işlenmemiş görüntü verisini alır. Bu bilgi her katman tarafından işlenir ve bir sonraki katmana aktarılır. Altıncı katmanda, bir kediye mi, bir soda kutusuna mı, yoksa bir uçağa mı baktığımıza karar verilir. Bu kavramları kullanarak bir yapay sinir ağı kurabiliriz. Bu ağdaki en sol katmana giriş katmanı denir ve bu tabakadaki nöronlara giriş nöronları denir. En sağdaki katmana çıkış katmanı denir. Ortadaki katmana gizli katman denir.



Şekil 3.2: Çok Katmanlı Yapay Sinir Ağları

3.1.4 Öğrenme süreci

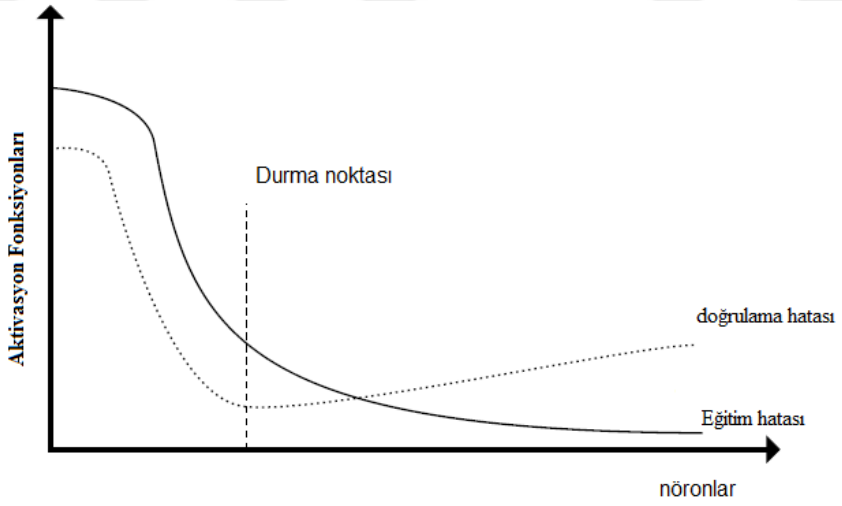
Öğrenme kabiliyeti, sinir ağlarının ana kavramını oluşturmaktadır. Sürecin amacı, verilen görevi çözmek için ağın optimum parametrelerini (ve yapısını) bulmaktır. Öğrenme süreci başlamadan önce, ağ parametrelerinin başlatılması gerekmektedir. Başlangıç değerleri genellikle rasgele seçilir, ancak bazı buluşsal yöntemlerin kullanılması optimal değerler açısından parametre ayarlamasının daha hızlı bir şekilde yapılmasına yol açabilir. Daha sonra öğrenme, eğitim verilerini ağ üzerinden besleyerek eğitim seti üzerinde gerçekleştirilir. Bu, eğitim kümesindeki her bir girdi üzerinde üretilen çıktıların analiz edildiği ve ağın daha iyi sonuçlar elde etmek için tekrar tekrar ayarlandığı gösteren tekrarlayan bir süreçtir. Ağın eğitim verileri üzerindeki hedef performansa ulaşmasından sonra, ağın eğitilmiş olduğu kabul edilir. Ağ performansını değerlendirmek için birçok farklı ölçüt bulunmaktadır. Bu tezde, etiketli bir veri kümesinden oluşan öğrenmeye odaklanacağım. Bu tür veri setleri, ağ için karşılık gelen etiketleriyle birlikte girdi kalıplarından oluşur. Etiketli veri kümesinden öğrenme süreci, danışmanlı öğrenme olarak adlandırılır. Veri etiketleri mevcut değilse, özel algoritmalar kullanılarak danışmansız öğrenmenin kullanılması mümkündür, bu özel algoritmalar burada tartışılmayacaktır.

Öğrenme sürecinde karşılaşılan yaygın bir sorun da aşırı uyma/uyum göstermektir. Genellikle öğrenme çok uzun süre gerçekleştirildiğinde ve özellikle

eğitim seti olası ağ girişlerinin alanındaki tüm kalıp tiplerini eşit şekilde temsil etmek için çok küçük olduğunda ortaya çıkar. Böyle bir durumda öğrenme, ağ eğitim verilerinde bulunan rasgele özelliklere göre ayarlayabilir. Aşırı uyum gösterim öğrenme sürecinde, ağın öngörücü performansı eğitim setinde gelişirken, diğer bir yandan daha önce görülmeyen test verilerinde kötüye gittiği zaman gözlemlenir.

Bu sorun ile mücadele etmek için etiketli veriler bir eğitim ve bir doğrulama setine bölünür. Doğrulama kümesini kullanmamızın başlıca nedeni, üzerinde çalıştığımız veriden bağımsız olarak verilerin hata oranlarını göstermesidir. Guyon tarafından yapılan bir araştırmada, eğitim ile doğrulama veri setleri arasındaki ideal oranın, tanınmış sınıfların sayısına ve sınıf özelliklerinin karmaşıklığına bağlı olduğu ileri sürülmektedir [29].

Bu oranın belirlenmesinde iyi bir başlangıç noktası, mevcut verilerin %80'ini eğitim setine, %20'sini ise doğrulama setine koymaktır. Daha fazla deneme, ideal orana daha yakın hareket etmeye yardımcı olabilir. Öğrenme gerçekleşirken ANN 'nin performansı doğrulama veri seti üzerinde düzenli olarak incelenir. Doğrulama verilerinde alınan hatalar durma noktasına ulaştığında, öğrenme süreci durdurulur ve ağ eğitilmiş olarak değerlendirilir.



Şekil 3.3: Eğitim hatasının ve doğrulama hatasının gelişimini karşılaştıran grafik

3.1.5 İleri Beslemeli Yapay Sinir Ağlar

İleri beslemeli ağları, çok katmanlı yapay sinir ağlarının bir türüdür. İleri beslemeli sinir ağları tek yönlü girdi sinyal akışı için izin verir. Giriş katman, gizli

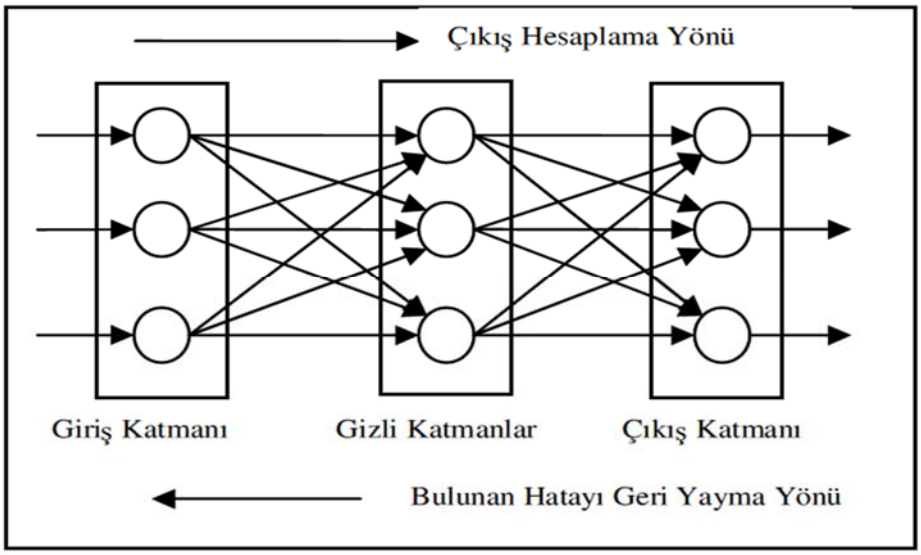
katmanlar yoluyla, çıkış katmanına hareket eder ve bilgiler her zaman ön katmana iletilir. Giriş katman, dışarıdan aldığı bilgileri hiçbir değişik yapmadan gizli katmandaki nöronlara iletir. Bilgi, gizli ve çıkış katmanında işlenerek ağ çıkışı belirlenir. İleri beslemeli ağlar, doğrusal olmayan statik bir işlevi gerçekleştirir. İleri beslemeli 3 katmanlı yapay sinir ağının, gizli katmanında yeterli sayıda hücre olmak kaydıyla, herhangi bir sürekli fonksiyonu istenilen doğrulukta yaklaştırabileceği gösterilmiştir. En çok bilinen algoritma olan geriye yayılım öğrenme algoritması, bu tip yapay sinir ağların eğitiminde etkin olarak kullanılmaktadır. Ağa hem örnekler hem de örneklerden almamız gereken sonuçlar verilmektedir. Ağ kendisine verilen örneklerden problem uzayını temsil eden bir çözüm uzayı üretmektedir. Daha sonra, gösterilen benzer örnekler için bu çözüm uzayını kullanarak sonuçlar ve çözümler üretebilmektedir [30].

3.1.6 Geri Beslemeli Yapay Sinir Ağları

Geri yayılım algoritmasının amacı, ağın istenen çıktıları üretmesi için nöronların ağırlıklarını ayarlamaktır. Algoritma, eğitim sürecini (öğrenme olarak da adlandırılır) tanımlar. Bu algoritmanın sonucu, verilen problemi çözerken hatayı en aza indirmek üzere yapılandırılmış bir sinir ağıdır. Eğitim etiketli veriler üzerinde yapılmalı ve bu nedenle denetlenmelidir. Algoritma başlamadan önce ağırlıkların bazı değerlere getirilmesi gerekir. Farklı yaklaşımlar olabileceğinden, başlatma algoritma özelliğinin bir parçası değildir, en yaygın olarak kullanılanı ise rastgele başlatma olmaktadır. Daha sonra ise eğitim algoritması başlamaktadır.

Öğrenme, çoklu dönemlerde yapılır. Her dönemde eğitim kümesindeki tüm veriler işlenir. Bir dönemin süresi, eğitim kümesindeki her girdi kalıbı tam olarak yalnızca bir kez işlendiğinden, ağın boyutuna ve eğitim kümesinin boyutuna doğrudan bağlıdır. Bununla birlikte, dönemlerin sayısı sınırlı değildir. Bu nedenle öğrenme sürecinin önemli bir kararı, durma kriterinin belirlenmesidir. Geriye doğru hesaplamada, ağın ürettiği çıktı değeri, ağın beklenen çıktıları ile kıyaslanır. Bunların arasındaki fark, hata olarak kabul edilir. Çıktı katmanında oluşan toplam hatayı bulmak için, bütün hataların toplanması gereklidir. Bazı hata değerleri negatif olacağından, toplamın sıfır olmasını önlemek amacıyla ağırlıkların kareleri hesaplanarak sonucun karekökü alınır.

Bu hata bir kez artmaya başladığında, bazı (yerel veya belki de genel olarak) minimum düzeyleri başarmış olduğumuzdan öğrenme sürecini bu noktada durdurmak akıllıca olacaktır.



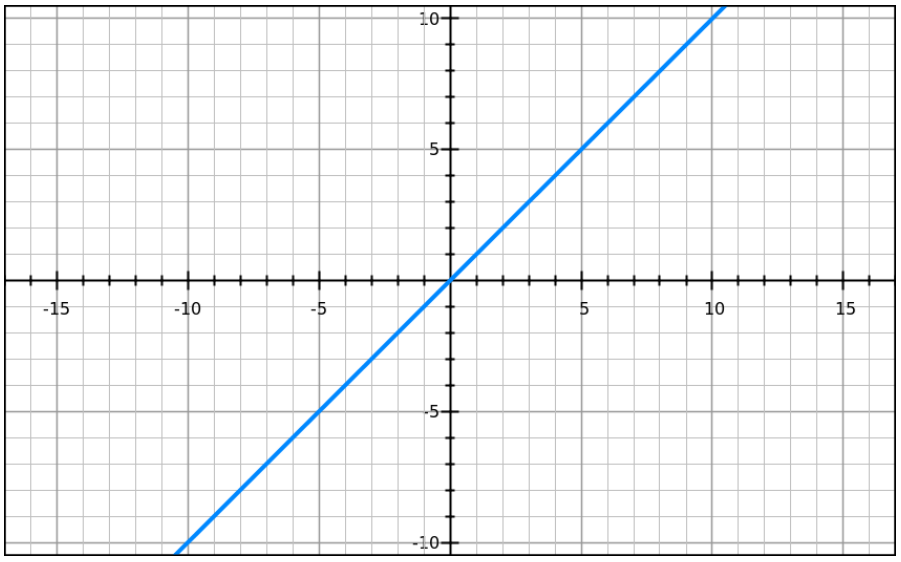
Şekil 3.4: Geri Yayılım Ağı *[31]*

3.1.7 Aktivasyon Fonksiyonları

Aktivasyon fonksiyonları, bir katmanın nöronlarının çıkışını bir sonraki katmana iletmek için kullanılır (çıkış katmanına kadar ve çıkış katmanı da dahil olmak üzere). Aktivasyon Fonksiyonları birçok türü vardır ve bu türlerden en çok kullanılan formları aşağıda açıklanmıştır.

3.1.7.1 Doğrusal aktivasyon fonksiyonu (Linear)

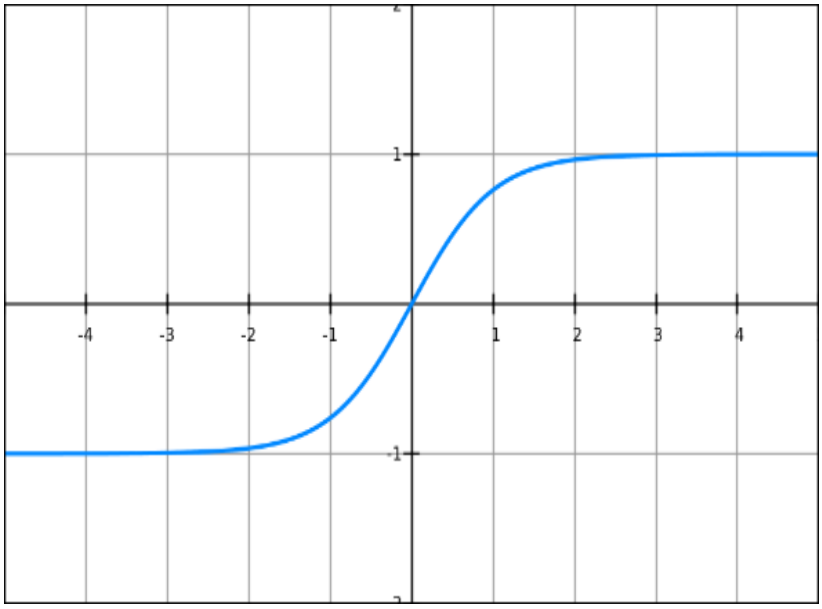
Doğrusal dönüşüm temelde özdeşlik fonksiyonudur, f(x) = Wx. Bağımlı değişken bağımsız değişkenle doğrudan orantılı bir ilişkiye sahiptir. Pratik anlamda, fonksiyonun sinyali değişmeden geçer. Bu Aktivasyon fonksiyonunu, sinir ağlarının giriş katmanında kullanılır.



Şekil 3.5: Doğrusal Aktivasyon Fonksiyonu *[31]*

3.1.7.2 Tanh aktivasyon fonksiyonu

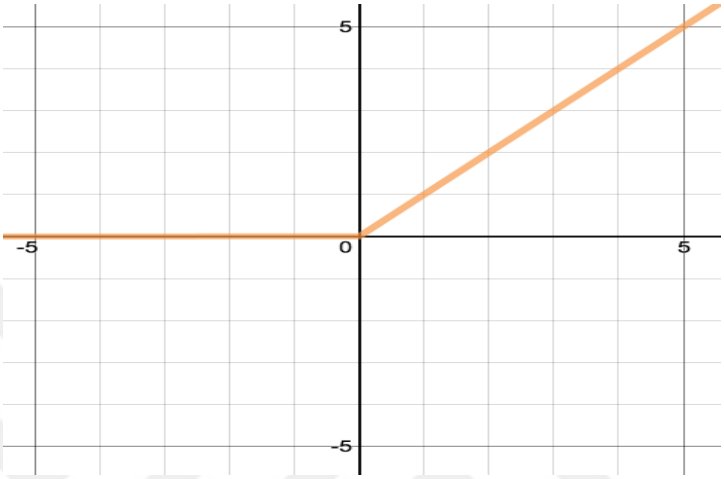
Tanh hiperbolik trigonometrik fonksiyondur. Tanjant bir dik üçgenin ters ve bitişik kenarları arasındaki bir oranı temsil ettiği gibi, tanh hiperbolik sinüsün hiperbolik kosinüse oranını temsil eder: tanh(x) = sinh(x) / cosh(x). Sigmoid aktivasyon fonksiyonun aksine, tanh'nın normaliz edilmiş aralığı -1 ile 1 arasındadır. Tanh'ın kullanılmasındaki asıl avantaj negatif sayılarla daha kolay anlaşabilmesidir.



Şekil 3.6: Tanh Aktivasyon Fonksiyonu *[31]*

3.1.7.3 Rectified linear aktivasyon fonksiyonu (ReLU)

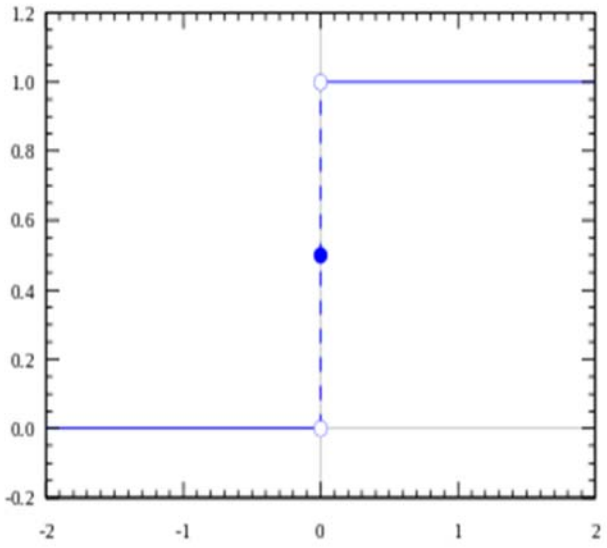
Bu fonksiyon, giriş değeri sadece belirli bir miktarın üzerinde olduğunda aktif olur. Girdi sıfırın altında iken, çıkış sıfırdır, ancak girdi belirli bir miktarın üzerinde olursa, bağımlı değişkenle doğrusal bir ilişkisi olur. Rectified Linear Fonksiyonu birçok farklı durumlarda performansını kanıtlamıştır.



Şekil 3.7: Rectified Linear Aktivasyon Fonksiyonu (ReLU) *[31]*

3.1.7.4 Adım (Step) aktivasyon fonksiyonu

Bu fonksiyon gelen girdinin belirlenen bir eşik değerin altında veya üstünde olması durumunda, nöron çıktısı 1 veya 0 değerini alır.



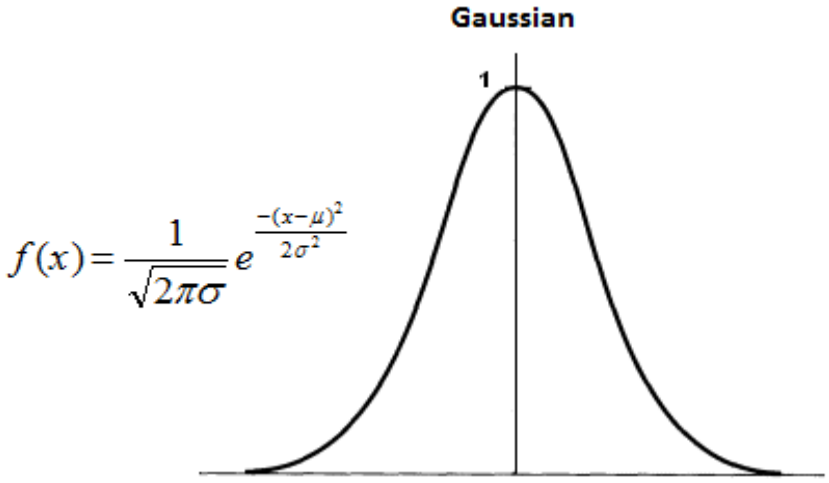
Şekil 3.8: Adım (Step) Aktivasyon Fonksiyonu *[31]*

3.1.7.5 Eşik değer aktivasyon fonksiyonu

Bu fonksiyon, nörona giren girdi 0'dan küçük eşit olduğunda nöronun çıktısı "0" olacaktır, nörona giren girdi 1'den büyük eşit olduğunda nöronun çıktısı "1" olacaktır, 0 ile 1 arasında olduğunda ise yine kendisini veren çıktılar üretilebilir. Bu fonksiyon aşağıdaki denklemde göstermektedir.

3.1.7.6 Gauss aktivasyon fonksiyonu

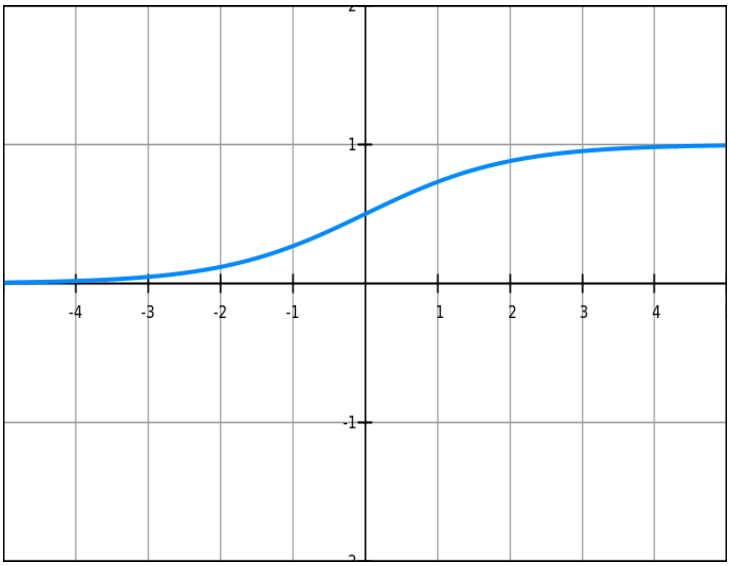
Gauss fonksiyonları sürekli çan şeklinde eğrilerdir. Net girdinin seçilen bir ortalama değerine ne kadar yakın olduğuna bağlı olarak nöron çıkışı (yüksek / alçak) sınıf üyeliği (1/0) açısından yorumlanır.



Şekil 3.9: Gauss Aktivasyon Fonksiyonu [31]

3.1.7.7 Sigmoid aktivasyon fonksiyonu

Tüm lojistik dönüşümleri gibi, sigmoid, verileri silmeden verilerin aşırı değerlerini azaltmanın bir yolunu temsil eder. Aşağıdaki şekilde dikey çizgi karar sınırıdır. Sigmoid fonksiyon, bağımsız değişkenleri 0 ile 1 arasında basit olasılıklara dönüştüren bir fonksiyondur ve çıkışının büyük kısmı 0 veya 1'e çok yakın olacaktır.



Şekil 3.10: Sigmoid Aktivasyon Fonksiyonu *[31]*

3.1.8 Hata Fonksiyonu

Bir sinir ağının eğitimindeki ana hedef ağın hata fonksiyonunu azaltmaktır. İyi eğitilen bir yapay sinir ağı minimum hataya sahiptir. Yapay sinir ağında bulunan hata, hata ölçüm fonksiyonları ile ölçülür. Bu hata fonksiyonları eğitim kümesindeki örneklerin hatalarının tümünü temsil eder. Problemin tipine ve tasarımcının seçimine göre farklı hata fonksiyonları kullanılabilir. Bu bölümde, sinir ağının çıkış hatasını hesaplamak için en yaygın kullanılan hata fonksiyonları açıklanmaktadır.

3.1.8.1 İkinci dereceden (kuadratik) hata fonksiyonu

Sinir ağları çıkış tabakasında yaygın olarak kullanılan hata fonksiyonlarından biridir. Aynı zamanda Mean Squared Error (MSE) olarak bilinir ve aşağıdaki denklem ile göstermektedir.

$$C(\mathcal{W}, b) \equiv \frac{1}{2n}\sum_x \| \mathcal{Y}(\mathcal{X}) - a \|^2 \qquad \text{(Denklem } \mathbf{3.1}\text{)}$$

Burada, $\mathbf{C}(\boldsymbol{\mathcal{W}}, \mathbf{b})$ ikinci dereceden (kuadratik) hata fonksiyonunu göstermektedir, $\boldsymbol{\mathcal{W}}$ ağdaki tüm ağırlıkları toplamını gösterir, b tüm baisların toplamıdır, $\boldsymbol{n}$ eğitim girdilerinin toplam sayısıdır, $\boldsymbol{\mathcal{Y}(\mathcal{X})}$ ağın istenen çıktısıdır, $\boldsymbol{a}$ ağ çıktıların vektörüdür, $\boldsymbol{\mathcal{X}}$ ağ girdisidir. Kuadratik hata fonksiyonunun formunu

inceleyerek, toplamdaki her terim negatif olmadığından, **C (w,b)**'nin negatif olmadığını görürüz. Bu fonksiyonda yavaşlama problemi bulunmaktadır.

3.1.8.2 Çapraz entropi hata fonksiyonu

Öğrenme yavaşlama problemi olan klasik kuadratik hata fonksiyonuna bir alternatiftir. Çapraz entropi hata fonksiyonunu anlamak için, birkaç giriş değişkeni olan x1, x2,..., bir nöronunu eğitmeye çalıştığımızı varsayacağız, ağırlıklar w1, w2,..., nöron çıktı formu, a = σ(z), burada, $z = \sum_j wjxj + b$ girdilerin ağırlıkları toplamıdır. Çapraz entropi hata fonksiyonu aşağıdaki denklem ile gösterilmektedir.

$$C = -\frac{1}{n}\sum_x [ylna + (1-y)\ln(1-a)] \qquad \text{(Denklem 3.2)}$$

Bu denklemde, n eğitim verilerinin toplam sayısıdır, x toplam tüm eğitim girdileri, y istenilen çıktıdır. Bu denklem öğrenme yavaşlama sorununu giderdiği kesin değildir ve negatif olmadığı gösterilebilir, $C \geq 0$. Dahası, nöronun gerçek çıktısı, verilen tüm girdiler için istenen çıktıya yakın olduğunda, çapraz entropi hata fonksiyonu sıfıra yakın olacaktır. Tek bir çıkış ağında, çapraz entropi hata fonksiyonunun kısmi türevi alınarak aşağıdaki denklemleri buluruz.

$$\frac{\partial C}{\partial wj} = \frac{1}{n}\sum_x xj(\sigma(z) - y) \qquad \text{(Denklem 3.3)}$$

$$\frac{\partial C}{\partial b} = \frac{1}{n}\sum_x \sigma(z) - y) \qquad \text{(Denklem 3.4)}$$

Bu denklemler ağırlığın öğrenme hızının, σ (z)- y değişkeni tarafından kontrol edildiğini gösterir. Hata ne kadar büyükse, nöron da o kadar hızlı öğrenir. Tek bir nöron için çapraz entropi inceledik. Fakat, çapraz entropi çok fazla sayıda nöronun çok katmanlı ağları ile genellemek kolaydır.

$$C = -\frac{1}{n}\sum_x \sum_j \left[y_j \ln a_j^L + \left(1 - y_j\right)\ln\left(1 - a_j^L\right)\right] \qquad \text{(Denklem 3.5)}$$

Bu denklemde, L ağdaki çıktı katmanıdır, $\boldsymbol{a_j^L}$ gerçek çıktı değeridir ve $\boldsymbol{y_j}$ çıkış nöronlarındaki istenen değerdir. Çapraz entropi hata fonksiyonu, ikinci dereceden (Kuadratik) hata fonksiyonundan daha hızlı çalışır fakat hata oranı daha fazladır.

3.1.8.3 Log-likelihood hata fonksiyonu

Nöronların softmax katmanlarına dayalı olan Log-likelihood hata fonksiyonu, öğrenme yavaşlama sorununa diğer bir yaklaşımdır. Softmax görüşü, sinir ağları için yeni bir çıktı katmanı türünü tanımlamaktadır. Softmax fonksiyonuna göre,a_j^L çıkış nöronunun aktivasyonu aşağıdaki denklem ile gösterilir.

$$\mathrm{A_j^L} = \frac{\mathrm{e}^{\mathrm{z_j^L}}}{\sum_{\mathrm{k}} \mathrm{e}^{\mathrm{z_k^L}}} \qquad \text{(Denklem } \mathbf{3.6}\text{)}$$

Ağa, bir eğitim girdisi göstermek için x'i kullanacağız, istenen çıktıyı belirtmek için ise y'yi kullanacağız. Log-likelihood hata fonksiyonu aşağıdaki denklem ile gösterilmektedir.

$$C \equiv -lna_y^L \qquad \text{(Denklem } \mathbf{3.7}\text{)}$$

3.1.9 Başlatma Yöntemleri

Sinir ağlarının iyi sonuç elde etmesi için başlatma oldukça önemlidir. Yakınsamayı hızlandırır ve zayıf bir yerel minimuma sıkışmaktan kaçınmaya yardımcı olur. Bu durum ağın sonucu için çok önemlidir; zayıf bir yerel minimuma sıkışmaktan kaçınmaya yardımcı olur ve ağın yakınsama hızı açısından oldukça önemlidir. Bu nedenle, birçok araştırmacı sinir ağı performansını iyileştirmek için başlatma teknikleri üzerinde çalışmıştır. Bunlardan bazıları ilerleyen kısımlarda gösterilecektir.

3.1.9.1 Gauss ve tekdüze başlatma

Gauss başlangıç işlemi, sinir ağında kullanılan yaygın bir başlatma yöntemidir. Sadece sıfır ortalama (simetriyi korumak etmek için) ve küçük varyansa $\left(N(0, \sigma^2)\right)$

sahip bir Gauss dağılımından ağırlıkları örnekler. Ağa verdiği simetri, yakınsama için oldukça önemlidir. Tekdüze başlatma, Gauss başlatmaya çok benzer şekilde kullanılır, fakat örnekler sıfır ortalamalı $v[-a, a]$ tekdüze dağılımdan alınır.

3.1.9.2 Glorot başlatma yöntemi

Rasgele tanımlanmış değişkene sahip tekdüze veya Gauss dağılımları kullanarak derin sinir ağlarını başlatmak, zorluğa neden olabilir ve eğitimi yavaşlatabilir. Zayıf başlatmanın ağ eğitimini nasıl yavaşlattığını ortaya çıkarmaya yönelik kapsamlı bir çalışma yaptılar [32]. Davranışlarını anlamak için bazı aktivasyon fonksiyonlarını analiz ettiler ve doğrusal bir nöron durumu için iyi bir varyans seçiminin aşağıdaki denklem tarafından verildiğini buldular.

$$\mathrm{Var(W)} = \frac{2}{\mathrm{fan_{in}+fan_{out}}} \qquad \text{(Denklem } \mathbf{3.8}\text{)}$$

Bu denklem, girdi ve çıktı arasındaki farkın bir ”1” olmasını sağlayarak elde edilmiştir. Düzgün bir dağılım ve doğrusal aktivasyon fonksiyonu kullanıldığı durumda aşağıdaki denklemle tanımlanan başlatmanın kullanılması iyi sonuçlar vermiştir.

$$\mathrm{W} \sim \mathrm{U}\,[-\mathrm{a}, \mathrm{a}],\ \mathrm{a} = \sqrt{\frac{6}{\mathbf{fan_{in}+fan_{out}}}} \qquad \text{(Denklem } \mathbf{3.9}\text{)}$$

Doğrusal bir aktivasyon işlevine sahip bir Gauss dağılımı durumunda, iyi bir başlatma aşağıdaki denklem ile verilmiştir.

$$\mathrm{W} \sim \mathrm{N}\,[0, \sigma^2],\ \mathrm{a} = \sqrt{\frac{2}{\mathbf{fan_{in}+fan_{out}}}} \qquad \text{(Denklem } \mathbf{3.10}\text{)}$$

Başlatma işlemleri, farklı aktivasyon fonksiyonları (doğrusal fonksiyonun farklı olması) için ayarlanmalıdır, kazanımlar aşağıdaki denklem ile elde edilenler ile aynıdır.

$$g = \begin{cases} 1 & \text{if activation is linear} \\ > 1 & \text{if activation is tanh} \\ \sqrt{2} & \text{if activation is ReLU} \end{cases}$$

Bu başlatma, pratikte çok iyi sonuçlar vermiştir ve çok derin ağların eğitilebilmesini de sağlayabilmiştir [33].

3.1.9.3 Ortogonal başlatma yöntemi

Ortogonal başlatma olarak bilinen yeni bir başlatma türünü ortaya koydular. Sinir ağının ağırlıklarını, bir g kazancı ile ölçeklendirilen rastgele ortogonal matrislerle başlatır [34]. Rastlantısal bir matris oluşturulur ve ardından rastgele bir ortogonal matris oluşturmak için tekil değer ayrıştırması (SVD) hesaplanır. Kazanç g, yukardaki denklemde gösterilen şekilde kullanılan aktivasyon fonksiyonuna bağlıdır. Tüm ağırlık matrislerinin sıfır olduğu eğer noktasının yakınına başladığı için küçük bir varyansa sahip gauss başlangıç durumunun zayıf olduğunu gösterdi [34]. Analiz ve matematik teorisi, yöntemlerinin iyi teorik sonuçlar gösterdiği derin doğrusal sinir ağları durumu için geliştirilmiştir. Daha sonra başlangıçta doğrusal durumlar için geliştirilen teoriyi kullanarak genel derin sinir ağları ile deneyler gerçekleştirdiler.

3.1.10 Optimizasyon Yöntemleri

Derin sinir ağı ile iyi sonuçlar veren birçok optimizasyon tekniği vardır ve bu yüzden bu teknikler büyük oranda kullanılmaktadırlar. Bu bölüm bu tekniklerin bazılarını gösterilecektir. Basit stokastik gradyan inişi bu tekniklerin hepsinin temelini oluşturmaktadır.

3.1.10.1 Stokastik Gradyan İnişi (SGİ) optimizasyon yöntemi

Geri yayılım algoritmasında, gradyan sinir ağının ağırlıklarını ayarlamak için birçok kez hesaplanmalıdır. Eğitim seti çok büyük olduğunda, genel olarak, tüm set için gradyanı hesaplamak çok pratik bir yöntem değildir, hafızaya sığması açısından hem çok yavaş hem de büyüktür. Bunu göz önünde bulundurarak, stokastik gradyan inişi kullanılmaktadır, bu da birkaç örnek üzerinde (tüm set yerine) hesaplanan

gradyandır. Genellikle stokastik gradyan inişini kullanmanın bir başka avantajı, yerel minimanın uzantı yeteneğidir [35]. Küçük-toplu işler yaygın olarak uygulanır, çünkü öğrenme varyansını düşürür ve bu nedenle daha istikrarlı bir yakınsama vardır. Başka bir nedeni ise, GPU'ların yüksek hesaplama gücüne sahip olması nedeniyle, küçük toplu işler çok hızlı işlenebilir, çünkü işlem kolaylıkla paralel hale getirilir. Aşağıdaki denklem stokastik gradyan inişinin güncelleme adımını göstermektedir.

$$\theta = \theta - \alpha * \sum_{k=i}^{i+m} \nabla_\theta J\left(\theta; x^{(k)}; y^{(k)}\right) \qquad \text{(Denklem } \mathbf{3.11}\text{)}$$

Burada θ güncellenecek parametredir ve α öğrenme oranıdır

3.1.10.2 Nesterov Hızlandırılmış Gradyan (NHG) optimizasyon yöntemi

Nesterov hızlandırılmış gradyanın (NHG), stokastik gradyan inişinden daha iyi olduğu gösterilmişti ve aynı stokastik gradyan inişi kadar uygulanması kolaydır [36]. Momentum yöntemleri, gradyan bilgilerini kullanarak maliyet fonksiyonunu azaltmak için iyi olan yönlerde hız biriktirir [37]. Aşağıdaki denklem, Nesterov hızlandırılmış gradyan güncelleme kuralını tanımlamaktadır.

$$\upsilon_t = \mu_{t-1}\upsilon_{t-1} - \epsilon_{t-1}\nabla f(\theta_{t-1} + \mu_{t-1}\upsilon_{t-1}) \qquad \text{(Denklem } \mathbf{3.12a}\text{)}$$

$$\theta_t = \theta_{t-1} + \mu_t \qquad \text{(Denklem } \mathbf{3.12b}\text{)}$$

Burada μ_t momenti ve ϵ_t öğrenme hızı kullanıcı tarafından seçilebilir (bunu yapmak için kullanılan en yaygın yol budur) ve genellikle bir momentum takvimi hızlı bir yakınsama elde etmek için kullanılır. NHG derin sinir ağlarında optimizasyon için en yaygın kullanılan yöntemlerden biridir.

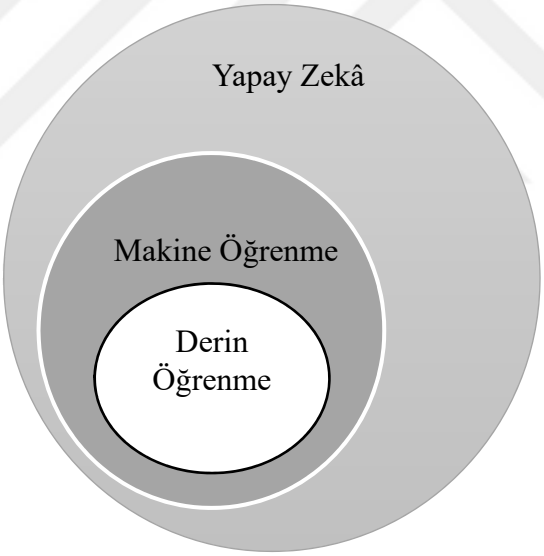
3.1.10.3 ADAM optimizasyon yöntemi

ADAM, KINGMA ve BA tarafından geliştirilen birinci derece gradyan tabanlı bir yöntemdir [38]. RMSProp optimizasyon yöntemi ve AdaGrad optimizasyon yöntemi iki optimizasyon yönteminden gelen fikirlerden ilham almaktadır. Pek çok problemde AdaGrad, RMSProp, SGİ ve NHG gibi diğer yöntemlerden daha üstün bir

performans gösterdiği ortaya konmuştur [38]. Genel olarak ADAM hızlıdır ve çok katı bir öğrenme programına ihtiyaç duymadığı için iyi bir seçimdir.

3.2 Derin Öğrenime giriş

Yapay zekâ, bilgisayarları insanların ve hayvanların sahip olduğu düşünce görevlerini yerine getirmeye muktedir olmakla ilgilidir [39]. Yapay zekânın bir dalı olan makine öğrenme ile ilgi çalışmalar son yıllarda oldukça arttı. Bilgisayar bilim programlarında, endüstri konferanslarında ve neredeyse her gün bilimsel dergilerde makine öğrenme tekniklerini görüyoruz. Temel olarak makine öğrenme, ham veriden bilgi çıkarmak ve bazı modellerde temsil etmek için algoritmalar kullanmaktadır [40]. Derin öğrenme, son on yılın büyük gelişme sayesinde, makine öğrenme yarışmalarını kazanarak ciddi bir rakip olarak ortaya çıktı. Derin öğrenme, yapay zekanın bir alt alanı olan makine öğrenme alanının bir alt bölümüdür.



Şekil 3.11: Yapay Zekâ ile Derin öğrenme arasındaki ilişki

Derin öğrenme modelleri çok caziptir, çünkü esnektirler (diğer bir deyişle, çok çeşitli problemlerde kullanılabilmektedirler). Derin mimariler son zamanlarda çok başarı göstermektedir, ancak geçmişte bu ölçüde başarılı olamamışlardır. Daha belirgin olarak, 2006 ve 2007 yıllarında araştırmacılar daha derin ağları eğitmeye başlamışlardır [41], [42], bunun öncesinde olan tek istisna LeCun tarafından kullanılan konvolüsyonel (kıvrımlı) sinir ağıdır [43]. Elde edilen yeni başarılar bu tekniklerin ilerlemesini engelleyen birçok sorunun üstesinden gelinmesiyle meydana

gelmiştir. Yakın geçmişteki başarının temel noktalarından bazıları, yeni optimizasyon teknikleri ve mimarileri oluşturulması ve (çok sayıda parametresi bulunan) derin ağları eğitmek için gerekli olan birçok alanda (resim, ses, metin vb.) erişilebilir olan büyük miktardaki verilerdir.

Derin öğrenmenin geniş ölçüde benimsenmesinin bir diğer nedeni, son GPU'ların hesaplama gücündeki ilerlemedir. Böylelikle (çok geniş bellek bant genişliği nedeniyle) büyük miktardaki verilerin analiz edilmesi, matris işlemleri çok hızlı yapılması ve mümkün olan eğitimin paralel eştirilmesi sağlanmıştır. Çok verimli sayısal hesaplama için GPU'ların nispeten daha düşük maliyetli olarak bulunabilirliğinden kaynaklanmaktadır.

Google, Microsoft, Amazon, Apple, Facebook ve daha birçoğu, büyük miktardaki verileri analiz etmek için bu derin öğrenme tekniklerini her gün kullanmaktadır. Bununla birlikte, bu tür uzmanlık saf akademik araştırmaların ve büyük sanayi şirketlerinin alanlarıyla sınırlı değildir. Ve bunlar modern yazılım üretiminin ayrılmaz bir parçası haline gelmiştir.

3.3 Derin Öğrenme tanımı

Derin öğrenmenin çeşitli tanımları ve yüksek seviyeli açıklamaları vardır:

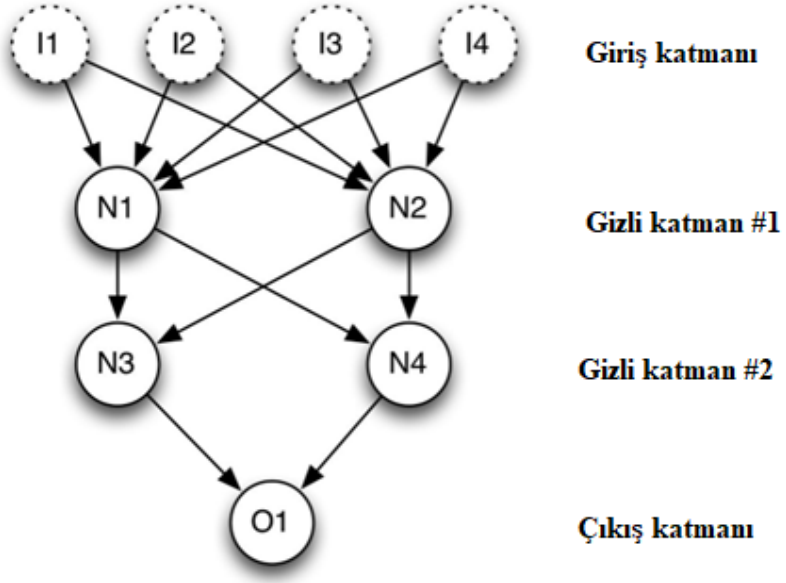
Derin öğrenme, hiyerarşik mimarilerdeki birçok katmandan oluşan bilgi işlem aşamalarının yapı sınıflandırması ve özellik veya temsil öğrenimi için kullanıldığı bir makine öğrenme teknikleri sınıfına atıfta bulunmaktadır [44].

Derin öğrenme, makine öğreniminin bir alanıdır ve yapay sinir ağlarının devrimci bir gelişimidir. Yapay zekâ, grafik modelleme, optimizasyon, model tanıma ve sinyal işleme kesişti noktada ortaya çıkmıştır. Derin öğrenme, çok katmanlı makine öğrenme modellerini kullanarak veriden gözetimli veya gözetimsiz öğrenme ile ilgilidir [45]. Çok gizli katmanlara sahip bir yapay sinir ağıdır. Sinir ağlarının nispeten basit bir kavramsal uzantısı olmasına rağmen, bu tür derin mimari, modellerin kapasitesi ve onları eğitmede yeni zorluklar açısından değerli ilerlemeler sağlar [46].

Derin öğrenme yapay sinir ağı programlamasında nispeten yeni bir gelişmedir ve derin yapay sinir ağları yetiştirmenin bir yolunu temsil eder. İki katmandan fazla olan herhangi bir yapay sinir ağı derindir [47].

3.4 Tam Bağlantılı Yapay Sinir Ağları (Dense)

Tam bağlantılı yapay sinir ağları bazen Dense adı olarak bilinir. Dense, yani bir katmandaki her bir nöron bir önceki katmandaki tüm nöronlara ve bir sonraki katmandaki tüm nöronlara bağlanır.



Şekil 3.12: Tam Bağlantılı Yapay Sinir Ağları (DENSE)

Tam bağlantılı sinir ağlarında, yukarıda gördeğimiz gibi, bu ağlarda her zaman bir giriş ve çıkış katmanı bulunmaktadır. Genel olarak derin ağlarda en az iki adet gizli katmana vardır. Tamamen bağlı ağlar, sinir ağlarının en basit modeli günümüzde sınıflandırma yapmak için kullanılır. Geleneksel tam bağlantılı bir sinir ağı ile sınıflandırma, görüntünün her girdi pikselini gizli nöron ünitelerinin tümüne atamanızı gerektirir. Tamamen bağlı ağların uzaklığa veya birbirine yakın olan girdi piksellerini aynı şekilde değerlendirir. Bu, ağ üzerinden ileri ve geri veri yayılımı yapıldığı için çok sayıda bağlantı ve ağırlık hesaplamaları anlamına gelmektedir. Tam bağlantılı yapay sinir ağları nöronlar arasındaki tam bağlantı nedeniyle boyutsallık lanetinden daha fazla etkindir.

3.5 Tekrarlayan Sinir Ağı (RNN)

Tekrarlayan sinir ağı (RNN), çıktının bazı zamanlarda ağın geçmiş durumuna bağlı olduğu bir sinir ağı türüdür, dolayısıyla bazı bağlantılar (beslemeli sinir

ağlarından farklı olan) yönlendirilmiş bir döngü oluşturur. Bu tür yapılandırmayla, ağ geçici bir davranış gösterebilir. Ağ aynı zamanda giriş dizilerini işleme yeteneği kazanan dahili bir bellek oluşturur. Bu gibi girdiler, metin, konuşma, zaman serileri ve dizideki bir öğenin kendisinden önce gelen diğer öğelere bağlı olduğu başka bir şey olabilir. Örneğin, köpek..... cümlesindeki bir sonraki kelimenin ‘’araba’’ değil de ‘’havlar’’ olması daha muhtemeldir, bu nedenle böyle bir dizilim göz önüne alındığında, bir RNN'nin ‘’havlar’’ kelimesini ‘’araba’’ kelimesinden daha çok tahmin etmesi olası bir durumdur. RNN'leri kullanmak için pek çok neden vardır, bunlardan bazıları aşağıda sıralanmıştır:

a. Tüm turing makineleri, sigmoid aktivasyon işlevleri olan, nöronlardan oluşmuş tam bağlantılı tekrarlayan ağlarla simüle edilebilir [48].
b. RNN'ler hâlâ, nasıl eğiteceğimiz konusunda ve mimarileri hakkında az şey bildiğimiz bir model türüdür, dolayısıyla RNN’ler önümüzdeki yıllarda büyük bir gelişme potansiyeli sahiptir.
c. Uçtan uca öğrenmeye imkân tanımaktadır ve bu da sisteme koymak için gerekli olan önceki bilginin miktarını en aza indirgemekte ve RNN ‘leri gerçek ölçekte benimsemek için önemli bir rol oynamaktadır.

Büyük bir potansiyele sahip olmasına rağmen, RNN'ler ve derin öğrenme ile ilgili en büyük sorun, genel olarak, RNN’lerin iyi sonuçlar elde edebilecek şekilde eğitilmesi için gerekli olan doğru mimariyi seçmektir. Geçmişte RNN'lerle ilgili iyi sonuçlar elde etmek neredeyse imkânsızdı çünkü hata yüzeyi birçok yerel minimal, plato ve uçurumlara sahipti. Karşılaşılan diğer problemler ise, eğitimde kaybolan ve patlayan Gradyan, uzun menzilli bağımlılıkların öğrenilmesindeki güçlük veya sinir ağı ayrışması yapmaktır. Bununla birlikte, günümüzde, yeni mimarilerin, optimizasyon tekniklerinin ve yeni GPU'ların keşfedilmesi ile birlikte birçok farklı sorun için çok iyi sonuçlar elde etmek mümkün olmuştur [49], [36].

3.5.1 RNN eğitim problemleri

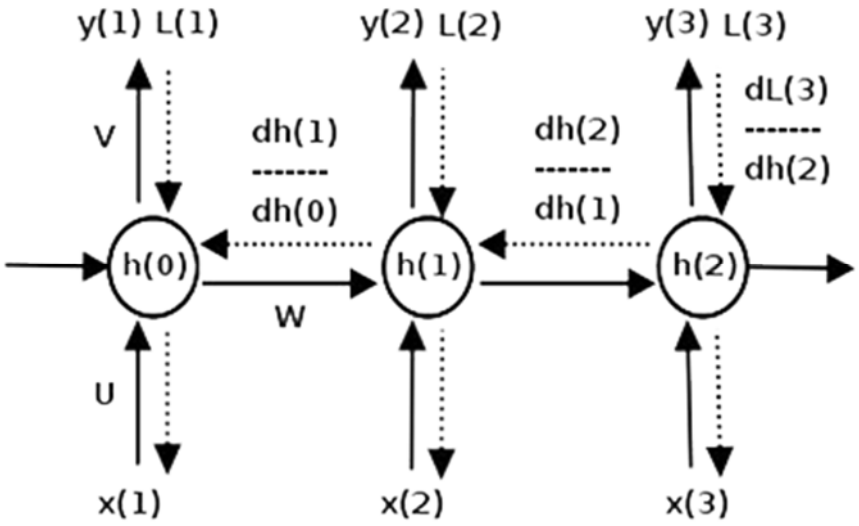
3.5.1.1 Uzun vadeli bağımlılık problemi

RNN'lerin avantajlarından birisi RNN’lerin önceki görev şuandaki göreve bağlayabilecekleri düşüncesidir. Örneğin önceki video karelerinin kullanılması şuandaki video karesinin anlaşılması hakkında bilgi sağlayabilir. Bazen, şuandaki

görevi yerine getirmek için yalnızca son bilgilere bakmamız gerekmektedir. “Bulutlar gökyüzünde" (The clouds are in the sky) ifadesinin son kelimesini tahmin etmeye çalışıyorsak, daha fazla içeriğe ve bağlama ihtiyaç duymayız- bir sonraki kelimenin gökyüzü (sky) olacağı açıktır. Alakalı bilgi ve ihtiyaç duyulduğu yer arasındaki boşluğun küçük olduğu bu gibi durumlarda, RNN‘ler geçmiş bilgileri kullanmayı öğrenebilirler. Ancak daha fazla içerik ve bağlama ihtiyaç duyduğumuz durumlar da söz konusudur. "Fransa'da büyüdüm... Akıcı Fransızca konuşuyorum" (I grew up in France… I speak fluent French.) cümlesinde geçen son kelimeyi tahmin etmeye çalıştığınızı düşünün. Yakın zamandaki bilgiler son sözcüğün muhtemelen bir dil adı olduğunu önermektedir (French-Fransızca). Ancak hangi dil olduğuna dair bilgileri daraltmak için, daha önceki bağlamdan (France-Fransa) içeriğine ihtiyacımız vardır. İlgili bilgi ile gerekli olduğu nokta arasındaki boşluğun oldukça büyük olması da tamamen mümkündür.

3.5.1.2 Kaybolan ve patlayan Gradyan problemi

Tıpkı geleneksel sinir ağları gibi, RNN’leri eğitmek te geri yayılım (Backpropagation) gerektirmektedir. Bu durumdaki fark şu şekilde açıklanabilir: Parametreler bütün zaman adımları tarafından paylaşıldığı için, her bir çıktıdaki Gradyan sadece mevcut zaman adımına değil aynı zamanda bir önceki zaman adımına da bağlıdır. Bu süreç zaman içerisinde geri yayılım (Backpropagation through time-BPTT) olarak adlandırılmaktadır.



Şekil 3.13: Üç katmanlı RNN *[50]*

Önceki diyagramda gösterilen küçük üç katmanlı RNN'yi ele alalım. İleri yayılım sırasında, ağ her zaman adımında Lt kaybını hesaplamak için etiketler ile karşılaştırılan tahminler üretir ve şekil 3.13'te düz çizgi ile gösterilmişti. Geri yayılım sırasında, U, V ve W parametreleri bakımından kaybın gardiyanları her bir zaman adımında hesaplanır ve parametreler gardiyanların toplamı ile güncellenir ve şekil 3.13'te noktalı çizgi ile gösterilmişti.

Aşağıdaki denklem, uzun vadeli bağımlılıklar için ağırlıkları kodlayan matris olan W'ye göre kaybın gardiyanını göstermektedir. Kaybolan ve patlayan Gradyan probleminin nedeni olduğu için, güncellemenin bu bölümüne odaklanıyoruz. U ve V matrislerine göre kaybın diğer iki gardiyanı da benzer şekilde tüm zaman adımlarında özetlenebilir:

$$\frac{\partial \mathrm{L}}{\partial \mathrm{W}} = \sum_{\mathrm{t}} \frac{\partial \mathrm{L_t}}{\partial \mathrm{W}} \qquad \text{(Denklem } \mathbf{3.13}\text{)}$$

Son zaman adımı (t=3)'te kaybın gardiyanına ne olduğunu ele alalım. Görebildiğiniz gibi, bu Gradyan zincir kuralı kullanılarak üç alt gardiyandan oluşan bir ürüne ayrılabilir. W'ye göre gizli durum h2'nin gardiyanı, her bir gizli durumun bir önceki duruma göre gardiyanının toplamı olarak ayrılabilir. Son olarak, gizli durumun önceki duruma göre olan her bir gardiyanı daha sonra mevcut gizli durumun bir önceki duruma karşı gardiyanlarının bir sonucu olarak ayırılabilir.

$$\frac{\partial \mathrm{L_3}}{\partial \mathrm{W}} = \frac{\partial \mathrm{L_3}}{\partial \mathrm{y_3}} . \frac{\partial \mathrm{y_3}}{\partial \mathrm{h_2}} . \frac{\partial \mathrm{h_2}}{\partial \mathrm{W}} \qquad \text{(Denklem } \mathbf{3.14}\text{)}$$

$$\frac{\partial \mathrm{L_3}}{\partial \mathrm{W}} = \sum_{\mathrm{t=0}}^{2} \frac{\partial \mathrm{L_3}}{\partial \mathrm{y_3}} . \frac{\partial \mathrm{y_3}}{\partial \mathrm{h_2}} . \frac{\partial \mathrm{h_2}}{\partial \mathrm{h_t}} . \frac{\partial \mathrm{h_2}}{\partial \mathrm{W}} \qquad \text{(Denklem } \mathbf{3.15}\text{)}$$

$$\frac{\partial \mathrm{L_3}}{\partial \mathrm{W}} = \sum_{\mathrm{t=0}}^{2} \frac{\partial \mathrm{L_3}}{\partial \mathrm{y_3}} . \frac{\partial \mathrm{y_3}}{\partial \mathrm{h_2}} . \left(\prod_{\mathrm{j=t+1}}^{2} \frac{\partial \mathrm{h_j}}{\partial \mathrm{h_{j-1}}} \right) . \frac{\partial \mathrm{h_2}}{\partial \mathrm{W}} \qquad \text{(Denklem } \mathbf{3.16}\text{)}$$

Benzer hesaplamalar W' ye göre (1 ve 2. Zaman adımlarında) L1 ve L2 kayıplarının gardiyanlarını hesaplamak için de kullanılır. Bu amaç doğrultusunda, yukarıda verilen denklemdeki gardiyanın son şekli RNN' lerin neden kaybolan ve patlayan Gradyan problemine sahip olduğunu göstermektedir. Bir önceki duruma göre

gizli durumun tek tek gardiyanlarının birden az olduğunu düşünelim. Birçok zaman adımında geri yayılım yaptığımızda, gardiyanların sonucu gittikçe küçülür ve kaybolan Gradyan problemine yol açar. Benzer bir şekilde; gardiyanların birden daha büyük olması durumunda, sonuç giderek büyür ve patlayan gardiyanlar sorununa yol açar.

Kaybolan gardiyanların etkisi şöyledir: Uzak adımlardan gelen gardiyanların öğrenme sürecinde hiçbir şeye katkısı yoktur, bu nedenle RNN' ler menzilli bağımlılıkları öğrenemezler. Kaybolan gardiyanlar geleneksel sinir ağları için de oluşabilir, fakat RNN' ler, üzerinde geri yayılımın gerçekleşmesi gereken daha fazla katmana (zaman adımlarına) sahip olma eğilimi gösterdiği için bu durum RNN'lerde daha belirgindir.

Tanh aktivasyon yerine ReLU kullanılması ve katmanların denetlenmeyen yöntemler kullanılarak ön eğitimden geçirilmesi gibi kaybolan gradyan sorununu en aza indirmek için çeşitli yaklaşımlar bulunsa da en popüler çözüm LSTM veya GRU mimarilerini kullanmaktır. Bu mimariler kaybolan Gradyan problemi ile başa çıkmak ve uzun vadeli bağımlılıkları daha etkili öğrenmek için tasarlanmıştır.

3.5.2 Basit Tekrarlayan Ağ (SRN)

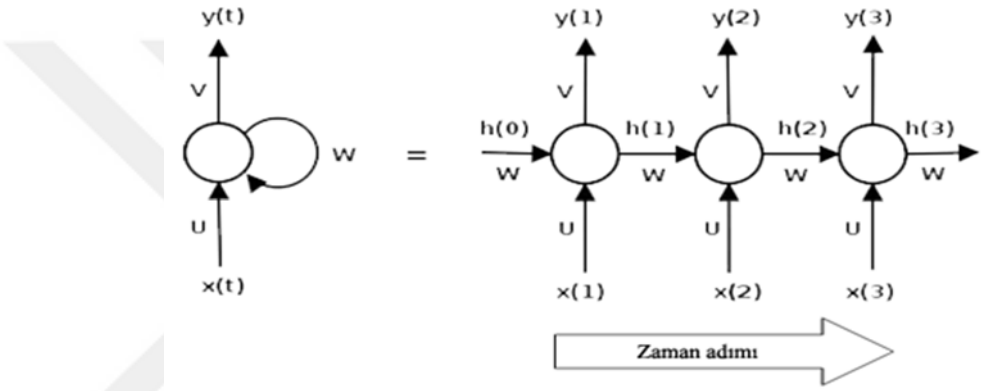
Geleneksel çok katmanlı sinir ağları, tüm girdilerin birbirinden bağımsız olduğunu kabul etmektedir. Bu varsayım, dizi verisi durumunda çökmektedir. Cümledeki ilk iki kelimenin üçüncü kelimeyi nasıl etkilediğini gösteren örneği önceki bölümde görmüştünüz. Aynı fikir konuşma için de geçerlidir- eğer gürültülü bir odada sohbet ediyorsak, şimdiye kadar duyduğumuz kelimelere dayanarak anlamadığımız bir kelime hakkında makul tahminler yürütebiliriz. Hava durumu gibi zaman serisi verileri, uzun vadeli eğilim olarak adlandırılan geçmiş verilere bağımlılık göstermektedir. RNN nöronları, bu bağımlılığı şimdiye kadar görülen şeyin özünü taşıyan gizli bir durum oluşturarak bünyesinde toplar. Gizli durumun zaman içindeki herhangi bir noktadaki değeri, önceki zaman adımında gizli durumun değerinin ve şu anki zaman adımında girdi değerinin bir fonksiyonudur, yani:

$$\mathrm{H_t} = \left(\mathrm{h_{t-1,}x_t}\right) \qquad \text{(Denklem } \mathbf{3.17}\text{)}$$

Burada h_t ve $h_{t-1,}$ sırasıyla t ve t-1 zaman adımlarındaki gizli durumların değerleri ve x_t, t zamanındaki girdi değeridir. Eşitliğin yinelemeli olduğuna dikkat

ediniz, yani, $h_{t-1,}$ değeri $h_{t-2,}$ ve xt-1 ve bu şekilde dizinin başlangıcına kadar giden değerler ile gösterilebilir. RNN'ler rasgele olarak uzun dizilerden gelen bilgileri kodlar ve birleştirir.

Ayrıca aşağıdaki sol kısımda bulunan şekil görüldüğü gibi RNN nöronu grafik olarak gösterebiliriz. T zamanda, nöron x(t) girdisine ve bir y(t) çıktısına sahiptir. y(t) çıktısının bir kısmı (ht gizli durumu) daha sonraki bir t + 1 aşamasında kullanılmak üzere nöron geri beslenir. Tıpkı bir geleneksel sinir ağının parametrelerinin ağırlık matrisinde bulunması gibi, RNN parametreleri sırasıyla girdi, çıktı ve gizli duruma karşılık gelen üç ağırlık matrisi U, V ve W ile tanımlanır:



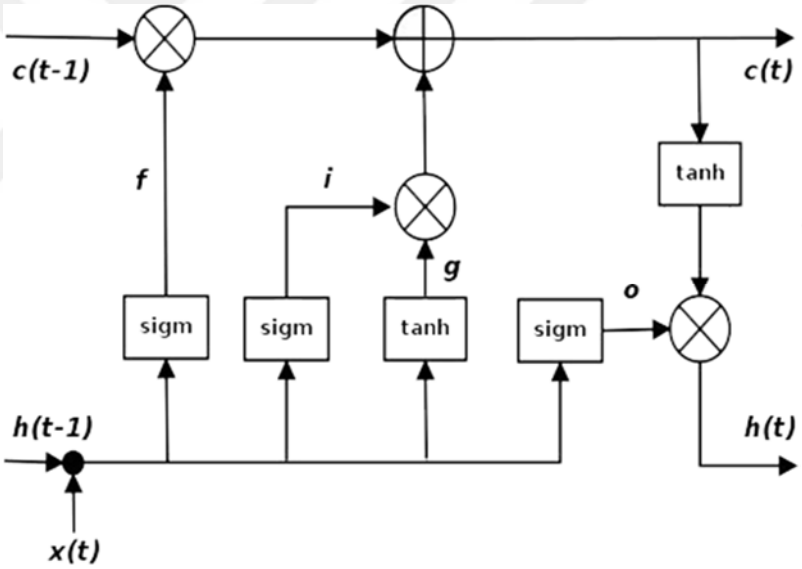
Şekil 3.14: RNN Yapay Sinir Ağı *[50]*

Bir RNN'yi açmak için kullanılacak başka bir yol sağ kısımda bulunan şekilde gösterildiği gibidir. Burada açmaktan kast edilen ağın dizinin tamamı için açılmasıdır. Burada gösterilen ağ, üç öğeden oluşan dizinin işlenmesi için uygun olan üç katmanlı bir RNN'dir. U, V ve W ağırlık matrislerinin basamaklar arasında paylaşıldığına dikkat ediniz. Bunun nedeni, aynı işlemi her bir zaman adımında farklı girdilere uyguluyor olmamızdır. Bu ağırlık vektörlerini tüm zaman adımlarında paylaşabilme yeteneği RNN'nin öğrenmesi gereken parametrelerin sayısını büyük ölçüde azaltmaktadır. Adından da anlaşılacağı gibi, basit yinelenen ağ (SRN), tekrarlayan sinir ağının basit bir mimarisidir ve vanilya RNN olarak da bilinir. SRN ‘de kaybolan ve patlayan gradyan problemi güçlü bir şekilde karşımıza çıkmaktadır. Pek çok çalışma bu sorunu mimariyi değiştirmeden çözmeye çalışmaktadır.

3.5.3 Uzun Kısa Dönem Ağları (LSTM)

Uzun Kısa Dönem ağları RNN'lerin özel bir türüdür, genellikle sadece ''LSTM'' olarak adlandırılan. Hochreiter & Schmidhubert tarafından ortaya çıkmış ve daha sonra yapılan çalışmalarda araştırmacılar tarafından geliştirilmiş ve popüler hale getirilmiştir [51]. LSTM ağları birçok geniş aralıktaki problemlere başarı ile uygulanmıştır örneğin, el yazısı tanıma [52], ses tanıma [53]. Bu tür bir mimari kaybolan gradyan problemi ve vanilya RNN'lerde yaygın olan uzun vadeli bağlılık problemini çözme amacı ile tasarlanmıştır.

RNN ağları gibi, LSTM'ler de aynı zamanda yinelemeyi benzer bir şekilde uygular, ancak tek tanh katmanı yerine birbirleriyle özel bir şekilde etkileşime giren dört katman vardır. Aşağıdaki diyagram t zaman adımında gizli duruma uygulanan dönüşümleri göstermektedir:



Şekil 3.15: LSTM Yapay Sinir Ağları *[50]*

Diyagramın üst kısmındaki çizgi c nöron durumudur ve birimin dahili belleğini temsil etmektedir. Alttaki çizgi gizli durumdur ve i, f, o ve g kapıları, LSTM'nin kaybolan Gradyan problemi etrafında çalıştığı mekanizmalardır. Eğitim sırasında, LSTM bu kapılar için parametreleri öğrenir.

Bu kapıların LSTM'nin gizli durumunu nasıl ayarladığını daha iyi anlayabilmek için, bir önceki zaman adımında h1 gizli durumundan, t zamanda ht gizli durumunu nasıl hesapladığını gösteren denklemleri ele alalım:

$$\text{İ} = \sigma(W_i h_{t-1} + U_i x_t) \quad \text{(Denklem } \mathbf{3.18}\text{)}$$

$$f = \sigma(W_f h_{t-1} + U_f x_t) \quad \text{(Denklem } \mathbf{3.19}\text{)}$$

$$o = \sigma(W_o h_{t-1} + U_o x_t) \quad \text{(Denklem } \mathbf{3.20}\text{)}$$

$$g = \tanh\left(W_g h_{t-1} + U_g x_t\right) \quad \text{(Denklem } \mathbf{3.21}\text{)}$$

$$c_t = (c_{t-1} \otimes f) \oplus (g \otimes i) \quad \text{(Denklem } \mathbf{3.22}\text{)}$$

$$h_t = \tanh(c_t) \bigotimes h\, o \quad \text{(Denklem } \mathbf{3.23}\text{)}$$

Burada i, f ve o girdi, unutma ve çıktı kapılarıdır. Aynı denklemler fakat farklı parametre matrisleri kullanılarak hesaplanmaktadırlar. Sigmoid fonksiyonu bu kapıların çıktısını sıfır ve bir arasında ayarlar, böylece üretilen çıktı vektörü ikinci vektörün ne kadarının ilk vektörden geçebileceğini tanımlamak için elemana göre başka bir vektör ile çarpılabilir.

Unutma kapısı önceki durum h_{t-1}'in ne kadarının geçmesine izin vermek istediğinizi tanımlar. Girdi kapısı yeni hesaplanan durumun x_t mevcut girdisi için ne kadarının geçmesine izin vermek istediğinizi tanımlar ve çıktı kapısı ise dahili durumun ne kadarının bir sonraki katmana geçmesine izin vermek istediğinizi tanımlar. Dahili gizli durum g, mevcut girdi x_t ve önceki gizli durum h_{t-1} 'e dayanılarak hesaplanır, g için olan denklemin Basit RNN hücresi için olan denkleme oldukça benzer olduğunu göz önünde bulunduralım; ancak bu durumda, çıktıyı girdi kapısı i'nin çıktısıyla ayarlayacağız.

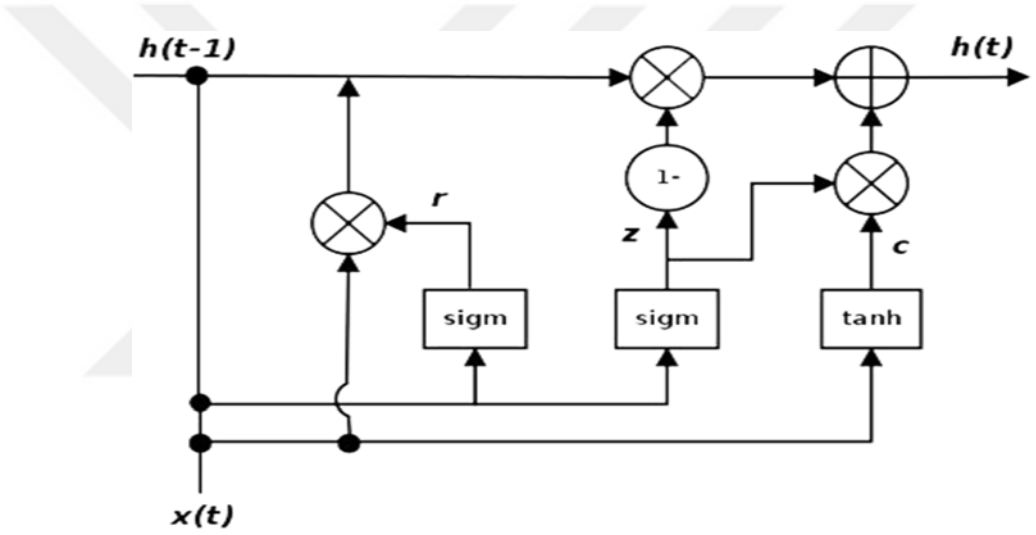
İ, f, o ve g verildiği için, t zamanda hücre durumu c_t'yi, unutma kapısı ve g durumu ile i girdi kapısı ile çarpılarak (t-1) zamanda ct-1 bakımından hesaplayabiliriz. Yani, bu durum temel olarak unutma kapısının 0 'a ayarlanması ile eski hafızanın ihmal edilmesi ve girdi kapısının 0'a ayarlanması ile yeni hesaplanan durumun ihmal edilmesi sayesinde eski hafıza ve yeni girdinin birleştirilmesinin bir yoludur. Son olarak, ct hafızasının çıktı kapısı ile çarpılması ile t zamanında gizli durum ht hesaplanır.

Dikkat edilmesi gereken bir nokta, bir LSTM'nin Basit RNN hücre hücresi yerine koyulabilir olmasıdır, tek fark LSTM'lerin kaybolan gradyan problemine dirençli olmasıdır. Herhangi bir yan etki hakkında endişe duymadan bir ağdaki bir

RNN hücresini bir LSTM ile değiştirebilirsiniz. Genel olarak daha uzun eğitim süreleri ile birlikte daha iyi sonuçlar alabilirsiniz.

3.5.4 Kapılı Yinelenen Birim (GRU)

İlk olarak Chung, Gulcehre, Cho, & Bengio, tarafından ortaya atılmıştır [54]. GRU (LSTM gibi) kaybolan Gradyan problemi ve vanilya RNN'lerde yaygın olan uzun vadeli bağlılık problemini çözme amacı ile tasarlanmıştır ancak iç yapısı daha basittir ve bu nedenle eğitmesi daha hızlıdır, çünkü gizli durumuna güncellemeler yapmak için daha az sayıda hesaplamalar yapmak gerekmektedir. GRU hücresi için kapılar aşağıdaki diyagramda gösterilmiştir:



Şekil 3.16: GRU YSA [50]

LSTM hücresindeki girdi, gizli, çıktı kapıları yerine, GRU hücresi iki kapıya sahiptir ve bu kapılar: güncelleme kapısı z ve reset kapısı r. Güncelleme kapısı önceki hafızanın ne kadar süre daha tutulacağını, resetleme kapısı ise yeni girdinin önceki hafıza ile nasıl birleştirileceğini tanımlamaktadır. LSTM'de olduğu gibi, gizli durumdan ayrı kalıcı hücre durumu yoktur. Aşağıdaki denklemler GRU'daki geçitlime mekanizmasını tanımlamaktadır:

$$z = \sigma(W_z h_{t-1} + U_z x_t) \qquad \text{(Denklem } \mathbf{3.24}\text{)}$$

$$r = \tanh(W_r h_{t-1} + U_r x_t) \qquad \text{(Denklem } \mathbf{3.25}\text{)}$$

$$c = (W_C(h_{t-1} \otimes r) + (U_c x_t) \qquad \text{(Denklem } \mathbf{3.26}\text{)}$$

$$h_t = (z \otimes c) \oplus ((1 - z) \otimes h_{t-1} \qquad \text{(Denklem } \mathbf{3.27}\text{)}$$

Çok sayıda bilimsel değerlendirmeye göre [55], GRU ve LSTM karşılaştırılabilir bir performansa sahiptir ve belirli özel bir görev için birini veya diğerini önermenin basit bir yolu yoktur. GRU'ların eğitilmek için daha hızlı olması ve genelleme yapmak için daha az veriye ihtiyaç duymasına rağmen, yeterli verinin bulunduğu durumlarda, bir LSTM'nin daha büyük dışa vurumcu gücü daha iyi sonuçlar verebilir.

3.6 Konvolüsyon Sinir Ağı (CNN)

CNN 2012 yılından sonra derin öğrenim alanında bir öncü olması sayesinde sınıflandırmanın kesinliğinde devasa iyileştirmeler yaşandıktan sonra popülerlik kazanmaya başladı. O zamandan beri, bazı ileri teknoloji şirketleri çeşitli hizmetler için CNN'i kullanmaktadır. Amazon CNN'i ürün tavsiyeleri için kullanır, Google fotoğraf araştırması için ve Facebook da esas olarak otomatik etiketleme algoritmaları için kullanır [50].

CNN, öğrenilebilir ağırlıklar ve önyargıları olan nöronlardan oluşan ileri beslemeli bir sinir ağı türüdür [56]. (CNN)" ifadesi networkun Konvolüsyon (evirişim) denilen bir matematik işlem kullanmasını ifade eder. CNN birçok alanda uygulanmıştır ancak, görüntü ve videolar gibi iki boyutlu verilerle kullanıldığında üstünlüğünü göstermiştir [57]. CNN spesifik olarak, iki boyutlu verilerdeki özellikleri tanımak için tasarlamıştır. CNN'lerin mimarisi [58] tarafından yapılmıştır, maymun görsel korteksindeki mikrobiyolojik sinyal işleme çalışmasından ilham almıştır. Tipik bir CNN uygulaması görüntülerdeki çeşitli nesnelerin tanınmasından oluşur. Ancak, Konvolüsyon networklar çeşitli görevler için de [59], [60], başarıyla kullanılmıştır.

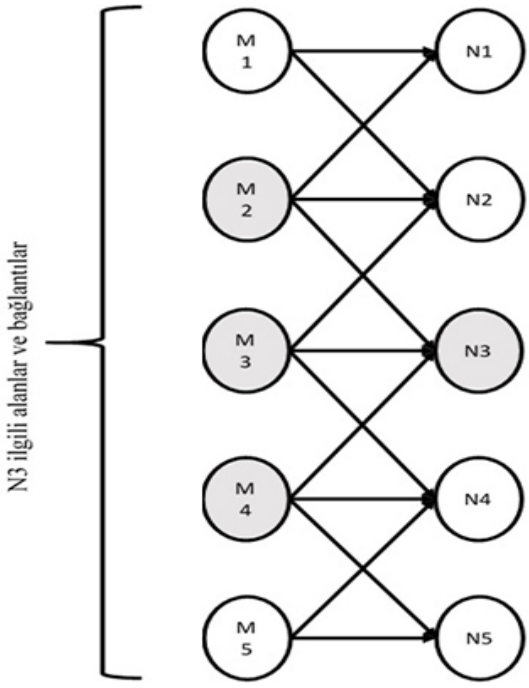
3.6.1 Konvolüsyon Ağların Özellikleri

Konvolüsyon ağların temel amacı, modelin belirli bir zamanda sınırlı sayıda girdi ile çalışmasına izin vermektir. Dahası, Konvolüsyon, derin bir öğrenme modelinin performansını artırmada önemli ölçüde yardımcı olan üç önemli özelliği desteklemektedir. Özellikler aşağıdaki gibi listelenmiştir: seyrek bağlantı, parametre

paylaşımı ve eşdeğerlik gösterimleri. Şimdi bu özelliklerin her birini sırayla açıklayacağız.

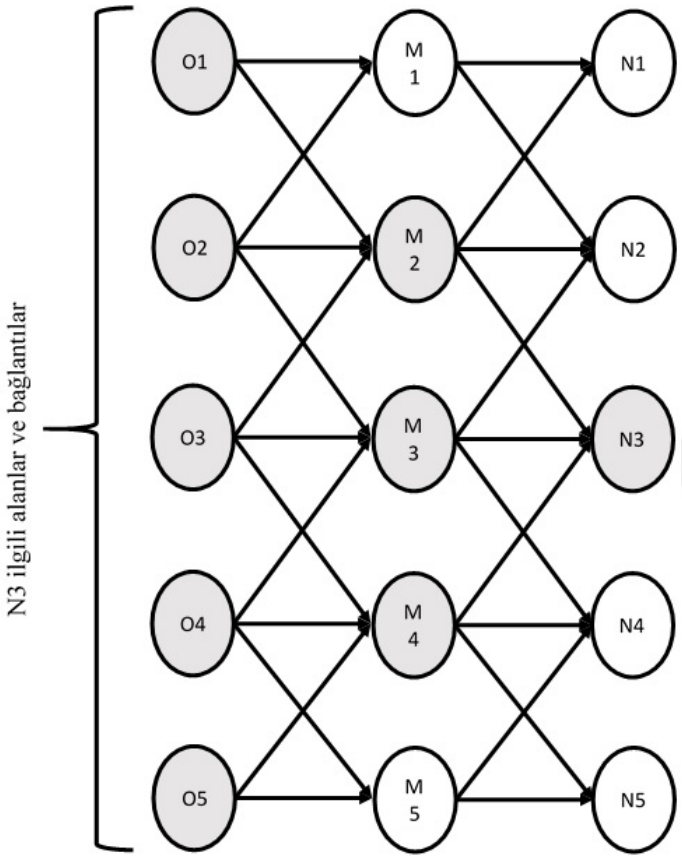
3.6.1.1 Seyrek bağlantı

Geleneksel ağ katmanları, matris çarpımını, her çıktı birimi ve her bir girdi birimi arasındaki etkileşimi tanımlayan farklı bir parametreden oluşan bir parametreler matrisi ile kullanır [50]. Diğer bir yandan, CNN'ler, bu amaçla bazen seyrek etkileşimler ya da seyrek ağırlık olarak adlandırılan seyrek bağlantı kullanıyor. Bu fikir, çekirdeğin boyutunu girişi daha küçük tutarak elde edilir, ki bu algoritmanın zaman karmaşıklığını azaltmaya yardımcı olur. Örnek olarak, büyük bir görüntü veri seti için, bu görüntü binlerce veya milyonlarca piksele sahip olabilir; buna ek olarak, çekirdeklerden yalnızca yüzlerce veya onlarca pikselin tümüne sahip olan kenarlar ve konturlar gibi resmin küçük, önemli özelliklerini belirleyebiliriz. Bu sebepten dolayı, yalnızca az sayıdaki parametreleri muhafaza etmemiz gerekiyor ve bu da bize modeller ve veri setlerinin hafıza gereksinimlerinin azaltılmasına yardımcı oluyor. Ayrıca bu fikir toplam bilgi işlem gücünü artıracak olan işlem sayısını hafifletir ve bunun sonucun, hesaplamanın çalışma süresinin karmaşıklığını büyük bir şekilde azaltır ve nihayetinde onun verimliliğini iyileştirir.



Şekil 3.17: Seyrek bağlantı yaklaşımı *[50]*

Şekil 3.17'te seyrek bağlantı yaklaşımı ile her bir nöron algılayıcı alan sayısını nasıl azaltabileceğini şematik olarak göstermektedir. Şekil 3.17 M giriş birimlerinin seyrek bağlantılı N3 çıkış birimini nasıl etkilediğini göstermektedir. Matris çarpımının aksine, seyrek bağlantı yaklaşımındaki alıcı alanların sayısı beşten üçe indirir (M2, M3 ve M4). Oklar da parametre paylaşım yaklaşımını gösteriyor. Bir nöronun bağlantıları model içerisindeki iki nöron ile paylaşılır. Bu nedenle, seyrek bağlantı yaklaşımı ile, her tabaka için alıcı alanlar, matris çarpım yaklaşımını kullanarak alıcı alanlardan daha küçüktür.



Şekil 3.18: Seyrek bağlantı yaklaşımı 2 *[50]*

Şekil 3.18 Konvolüsyon sinir ağlarının derin katmanları için seyrek bağlantıyı temsil etmektedir. Şekil 3.18 farklı olarak, N3 ünitesi üç alıcı alana sahip olduğu yerde, N3'ün alıcı alanları sayısı beşe yükselmiştir.

Seyrek bağlantı özelliğine sahip CNN katmanları zaman karmaşıklığını geliştirir. Örneğin, eğer bir katmanda p girişleri ve q çıkışları varsa, matris çarpımı parametrelerin (p * q) sayısını gerektirir. Algoritmanın çalışma süresi karmaşıklığı O (p * q) olacaktır. Seyrek olarak bağlı yaklaşımla, eğer her çıkışla ilgili üst limit

bağlantı sayısını n olarak sınırlarsak, o halde sadece n * q parametrelerine ihtiyaç duyulacak ve çalışma zamanı karmaşıklığı O (n * q) 'ya düşecektir

3.6.1.2 Parametre paylaşımı

Parametre paylaşımı, bir işlev için aynı parametrenin modeldeki çoklu işlevler için kullanılabileceği süreç olarak tanımlanabilir. Düzenli sinir ağlarında, katmanın çıktısını hesaplarken ağırlık matrisinin her bir elemanı tam olarak bir kez uygulanır. Ağırlık, girdinin bir elemanı ile çarpılır, fakat hiçbir zaman tekrar gözden geçirilmez. Parametre paylaşımı, bir girdi için kullanılan ağırlık değeri, diğer girdiler için kullanılan değere bağlı olduğundan, bağlı ağırlık olarak da ifade edilebilir. Şekil 3.18 parametre paylaşımı için bir örnek olarak da görülebilir. Örneğin, M2'den gelen bir parametre hem N1 hem de N3 ile kullanılır. Bu işlemin temel amacı Konvolüsyon katmanındaki serbest parametrelerin sayısını kontrol etmektir [50]. Bir CNN'de, çekirdeğin her bir öğesi, girdinin neredeyse her konumunda kullanılır. Bunun için mantıklı bir varsayım şu dur ki, eğer özelliklerden birinin bazı uzamsal konumda istenilen durum ise, o zaman diğer pozisyonların da hesaplanması gereklidir. Tek bir derinlik diliminin tüm elemanları aynı türden parametrizasyonu paylaştığından dolayı, Konvolüsyon katmanının her bir derinlik dilimindeki ileriye doğru geçişi, giriş hacminin Konvolüsyon olarak nöronların ağırlığı ile ölçülebilir. Bu konvolüsyonel sonucu, harita üzerinde bir aktivitedir. Bu aktivasyon haritalama koleksiyonları, çıktı hacminin sonuçlanması için derinlik boyutu ilişkilendirmesiyle birlikte yığılır. Parametre paylaşım yaklaşımı, CNN mimarisinin çeviri değişmezliğini arttırmasına rağmen, ileriye doğru yayılımın çalışma süresini artırmaz. Parametre paylaşımında, modelin çalışma zamanı O (n * q) olarak kalır. Bununla birlikte, modelin depolama gereksinimi parametrelerin n sayına azaltıldığında, genel alan karmaşıklığını önemli ölçüde azaltmaya yardımcı olur. P ve q genel olarak benzer boyutlara sahip olduklarından, n'nin değeri, p * q ile karşılaştırıldığında neredeyse ihmal edilebilecek kadar olur.

3.6.1.3 Eşdeğerlik gösterimleri

Konvolüsyon katmanında, parametre paylaşımı nedeniyle, katmanlar eşitlik tercümesi olarak adlandırılan bir özelliğe sahiptir. Bir eşitlik fonksiyonu, girdinin yaptığı gibi çıktının da değişen bir fonksiyon olarak tanımlanır.

Bu kavram bazı durumlarda yararlıdır; örneğin, iki farklı takımın kriket oyuncularından oluşan bir grup fotoğrafı düşünün. Bazı oyuncuları belirlemek için resimdeki formanın bazı ortak özelliklerini bulabilirsiniz. Şimdi, benzer özellik açık bir şekilde diğerlerinin tişörtlerinde de mevcut olacak. Bu nedenden dolayı, parametreyi tüm görüntü boyunca paylaşmak oldukça pratiktir.

3.6.2 Konvolüsyon Sinir Ağları için Parametreler Seçimi

Çıktı hacminin boyutunu kontrol etme yollarını tartışacağız. Başka bir deyişle, çıktı hacmindeki nöron sayısını ve bunların nasıl düzenlendiğini kontrol edilmesi. Temel olarak, Konvolüsyon katmanlarının dış hacminin boyutunu kontrol eden üç tane parametre vardır. Bunlar: derinlik, basamak ve sıfır doldurma. Bu parametreler kaç tane Konvolüsyon katmanlarında kullanmalıyız, filtrelerin boyutu ne olmalı ya da adım ve boşluk değerleri ile ilgili bize bilgi verir. Bunlar çok öznel sorular ve bunların çözümleri doğada hiçte önemsiz değildir. Hiçbir araştırmacı bu parametreleri seçmek için herhangi bir standart parametre oluşturmaz. Bir sinir ağı genellikle ve büyük ölçüde eğitim için kullanılan verilerin türüne bağlıdır. Bu veriler boyut olarak, giriş işlenmemiş görüntüsünün karmaşıklığı, görüntü işleme görevlerinin türü ve diğer birçok kritere bağlı olarak değişebilir. Büyük veri kümesine bakarak düşüncenin genel bir çizgisinde, doğru kümeyi çıkarmak için kombinasyonu karar vermek için parametrelerin nasıl seçileceğini düşünmek gerekir; ki bu, doğru ölçekte görüntülerin soyutlamalarını oluşturur.

3.6.2.1 Derinlik

Çıkış hacminde, derinlik önemli bir parametre olarak düşünülür. Derinlik, girişteki bazı değişiklikler üzerinde her bir öğrenme yinelemesini uygulamak istediğimiz filtre sayısına karşılık gelir. İlk konvolüsyonel katman, girdi olarak işlenmemiş görüntü alması durumunda, derinlik boyutu boyunca birden çok nöron,

çeşitli renk lekeleri veya farklı yönlendirilmiş kenarların varlığında etkinleşebilir. Aynı girdi bölgelerindeki nöron kümeleri derinlik sütunu olarak adlandırılır [50].

3.6.2.2 Adım (Stride)

Adım, mekânsal boyut (genişlik ve yükseklik) etrafında derinlik sütunlarının tahsisinin politikasını belirtir. Temel olarak, giriş hacminin etrafında filtrenin evriştirimini kontrol eder. Adım, evrişim sırasında filtrenin hangi değer ile kaydırıldığı miktarı biçimsel olarak tanımlanabilir. İdeal olarak, basamak değeri bir kesir değil, bir tam sayı olmalıdır. Kavramsal olarak, bu miktar, bir sonraki katmana geçmeden önce ne kadar veri görüntü girişinin girmek istediğini belirlemede yardımcı olur. Adım ne kadar çok olursa, bir sonraki tabaka için daha çok bilgi tutulacaktır. Örneğin, adım 1 iken, uzamsal konumlara ve ayrı bir boşluk birimine yeni bir derinlik sütunu atanır. Bu, sütunlar arasındaki ağır bir biçimde çakışan alıcı alanlarından dolayı büyük bir çıktı hacmi üretir. Öte yandan, eğer adım değeri artırılırsa, alıcı alanlar arasında daha az üst üste binme olacaktır ve bu da mekânsal olarak daha küçük boyutsal çıkış hacmi olarak sonuçlanacaktır [50].

3.6.2.3 Sıfır doldurma (Zero-padding)

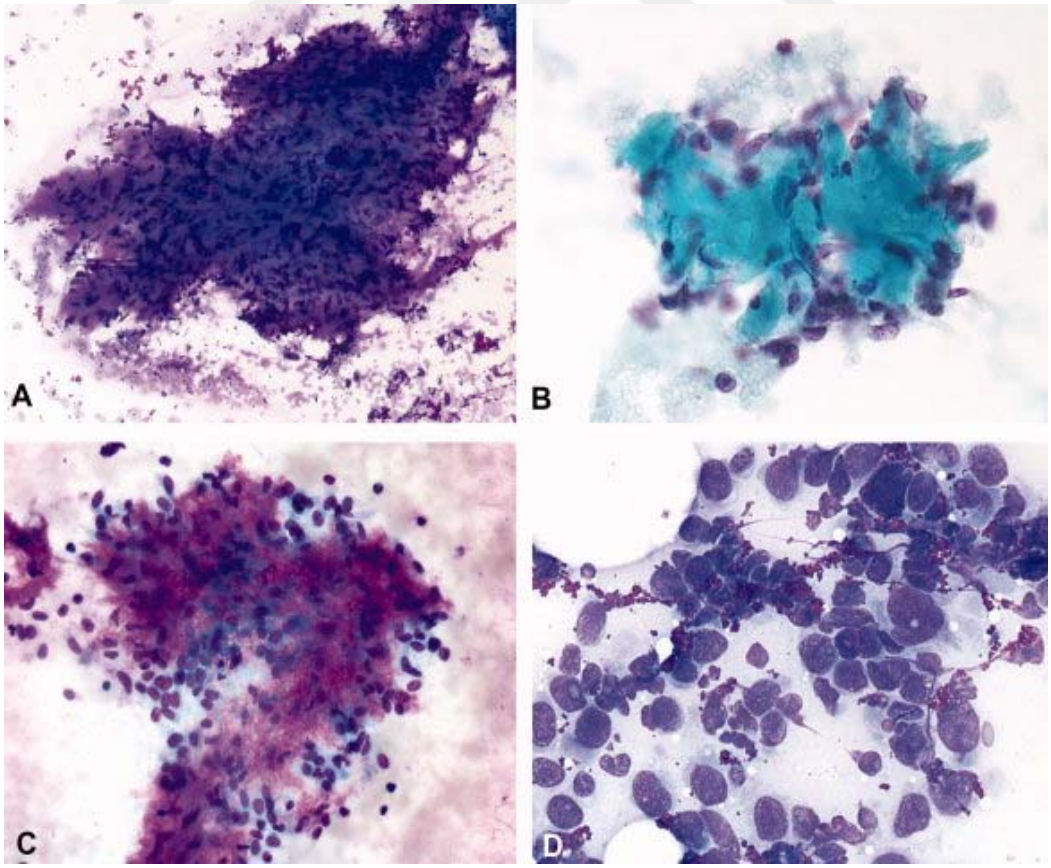
Giriş hacmine daha fazla kıvrım katmanı uyguladığımızda, çıkış hacminin boyutu daha da azalır. Bununla birlikte, bazı durumlarda, alt düzey özellikleri ayıklayabilmemiz için orijinal giriş hacmi ile ilgili neredeyse tüm bilgileri korumak isteyebiliriz. Bu tür senaryolarda, girdi hacmini, giriş hacminin sınırları çevresini sıfırlarla doldururuz. Bu sıfır dolgu boyutu bir hiper parametre olarak düşünülüyor. Giriş hacminin mekânsal boyutunu tam olarak korumak istediğimiz çıkış hacmi senaryoların mekânsal boyutunu doğrudan kontrol etmek için kullanılan bir parametre olarak tanımlanabilir.

DÖRDÜNCÜ BÖLÜM

MATERYAL VE YÖNTEMLER

4.1 Çalışmasında Kullanılan Veri Seti

Kullandığım bu veri seti, 1992 yılında Wisconsin Üniversitesi tarafından bir göğüs kütlesinin ince bir iğne aspirasyonunun (FNA) sayısal bir görüntüsünden alınan özellikleri kullanarak bağışta bulunan çok ünlü bir veri setidir. Makine öğrenimi ve klinik tanı konusunda bu veri setinden yola çıkarak, çok sayıda makale yayınlandı.



Şekil 4.1: Göğüs kütlesinin ince bir iğne aspirasyonunun (FNA) görüntü örneği [60]

Wisconsin göğüs kanseri teşhisi (WBCD) veri seti, UCI makine öğrenme havuzundan indirildi [61] ve metin dosyası olarak kaydedildi. Bu veri tabanı, 569 örnek ve 32 sınıf özniteliği içeren özniteliklere sahiptir. Özellik 2, sınıf niteliğidir. Örnekleri temsil etmek için başka nitelikler kullanılır. Diğer nitelikler örnekleri temsil etmek için kullanılır. Her örnek, kötü ve iyi huylu olmak üzere olası iki sınıftan birine sahiptir. Sınıf dağılımına göre: 357 iyi huylu ve 212 örnek ise kötü huyludur (Bakınız Tablo 4.1). WDBC veri kümesi özellik ile ilgili bilgileri vermektedir.

Tablo 4.1: WDBC veri kümesi

Özellik adı	Anlamı	Özellik Kimliği
Kimlik	Hastanın özgün kimliği	1
Teşhis	Teşhis (B- İyi huylu / M- Kötü huylu)	2
Yarıçapı	Çevrinin merkeze olan ortalama uzaklıkları	3, 13, 23
Doku	Gri ölçek değerlerinin standart sapması	4, 14, 24
Çevre	Hücre çekirdeğinin çevresi	5, 15, 25
Alan	Hücre çekirdeğinin alanı	6, 16, 26
Pürüzsüzlük	Yarı çap uzunluklarındaki yerel değişim	7, 17, 27
Yoğunluk	çevre^2 / alan	8, 18, 28
İçbükeydik	Şeklin iç bükey oranlarının yoğunluğu	9, 19, 29
Konik noktalar	Çevrenin konkav kısımlarının sayısı	10, 20,30
Simetri	Hücre çekirdeğinin simetri	11, 21, 31
Fraktal Boyut	Sahil şeridi yaklaşımı	12, 22, 32

Tablo 4.1'de yer alan WDBC veri kümesinde bulunan özniteliklerin ayrıntıları şunlardır: kimlik numarası, tanı (M=kötü huylu, B=iyi huylu) ve 10 adet gerçek değerli özellikler her bir hücre çekirdeği için hesaplandı: yarıçap, doku, çevre, alan, pürüzsüzlük, yoğunluk, konkavlık, konik noktalar, simetri ve fraktal boyut [62]. Tüm bu özellikler bir meme kütlesinin bir ince iğne aspiratının (FNA) sayısallaştırılmış bir görüntüsünden hesaplanır. Tek bir çekirdeğin yarıçapının neresi olduğu, aperatif ağırlık merkezi ve Bireysel aperatif noktaları tarafından belirlenen radyal çizgi parçalarının ortalama uzunluğu tarafından ölçülür. Ardışık aperatif noktaları arasındaki toplam mesafe çekirdek çevreyi oluşturur. Alan, aperatif içindeki piksel sayısını sayarak ve çevre üzerindeki piksellerin yarısını ekleyerek ölçülür. Çevresi ve alan, çevrenin karesinin alan bölünmesi (çevre^2/alan) formülünü kullanarak hücre

çekirdeğinin kompakt lığının bir ölçüsü vermek üzere birleştirilir. Düzgünlük, bir radyal çizginin uzunluğu ile onu çevreleyen çizgilerin ortalama uzunluğu arasındaki farkın ölçülmesiyle hesaplanır. Bu aperatif eğrilik enerjisi hesaplamasına benzemektedir. Konkavite, hücre çekirdeğinin sınırındaki girinti boyutunun (girintiler) ölçülmesi ile yakalanır. Bitişik olmayan aperatif noktaları arasındaki kirişler çizilir ve her kirişin içine doğru uzanan çekirdeğin gerçek sınırının ne derece olduğunu ölçülür. İçbükey Noktalar özelliği içbükeylik özelliğine benzer, fakat bu özellik sadece sınırın içbükey bölgeleri üzerindeki sınır noktası sayısını ile hesaplanır. Simetriyi ölçmek için, merkezin içinden geçen en büyük kiriş veya en uzun eksen bulunur. Daha sonra, her iki yöndeki nükleer sınırın ana eksenine dik olan çizgiler arasındaki uzunluk farkı ölçülür. Bir çekirdek sınırın fraktal boyutu Mandelbrot tarafından tanımlanan kıyı şeridi yaklaşımıyla hesaplanılır. Çekirdeğin çevresi giderek daha büyük cetveller kullanılarak ölçülür. Cetvel boyutu arttıkça, ölçüm hassaslığı azaltılırken, gözlenen çevre uzunluğu azalır. Cetvel boyutunun loğuna karşı gözlemlenen alanın loğunu çizmek ve aşağı eğimi ölçmek fraktal boyuta bir yaklaşımı (negatif) verir. Tüm şekil özellikleri ile daha yüksek bir değer, daha az düzenli bir kontura karşılık gelir ve böylece daha yüksek bir tehlike (kötü) olasılığına karşılık gelir. Hücre çekirdeğinin dokusu, bileşen piksellerindeki gri skala yoğunluğunun değişkenliğini bularak ölçülür.

4.2 Materyal

4.2.1 Anakonda

Anaconda, bir paket yöneticisi, bir çevre yöneticisi, bir Python dağıtımı ve 720'den fazla açık kaynak paketinin bir koleksiyonudur. Anaconda ücretsiz ve kurulumu oldukça kolaydır. Ayrıca ücretsiz topluluk desteği sunar. 4,5 milyonun üzerinde kullanıcısıyla Anaconda, dünyanın en popüler ve güvenilir veri bilimi eko sistemidir. Modern veri bilimin temelini oluşturan açık kaynak projeleri üzerinde öncü geliştirme ile yenilik yapmaya devam etmektedir. Ayrıca, işletmeler için Anaconda'nın desteklenmesine, yönetilmesine, ölçeklendirilmesine, temin edilmesine, özelleştirilmesine ve güvenli hale getirilmesine yardımcı olan ürünler ve hizmetler sunulmaktadır. Anaconda ile izole edilmiş ortamlarda Python'un birden

çok sürümünü çalıştırabilirsiniz, bu nedenle, varsayılan Python sürümünüz olacak gibi daha sık kullandığınız Python sürümüyle indirmeyi seçebilirsiniz.

4.2.2 Theano

Theano çerçevesi, bir optimizasyon derleyicinin avantajları ile bir bilgisayar cebir sisteminin avantajlarını bir araya getiren ilk çerçevelerden biridir. Bu yöntem Montreal Üniversitesinde araştırma grubu tarafından geliştirildi. Theano, çok boyutlu dizilere (tensörler) üzerine yoğun bir şekilde odaklanması ile matematiksel ifadeleri oldukça verimli bir şekilde uygulamak, derlemek ve değerlendirmek için geliştirildi. Theano CPU'lar üzerinde kod çalıştırmak için bir seçenekle birlikte gelir. Bununla birlikte, onun asıl gücü, büyük bellek bant genişliklerinin ve kayan nokta matematiği için büyük yeteneklerin avantajlarından yararlanmak için GPU'ları kullanmaktan kaynaklanmaktadır. Theano'yu kullanarak paylaşılan belleğin üzerinde de paralel olarak kolayca kod çalıştırabiliriz. 2010 yılında, Theano geliştiricileri kod CPU'da çalıştırıldığında NumPy'den 1,8 kat daha hızlı bir performans ile çalıştığını gösterdi ve Theano GPU'yu hedef alınırsa, NumPy'den bile 11 kat daha hızlıydı [63]. Theano, eğer biz bu kodu GPU üzerinde çalıştırmak istersek, C / C ++ veya CUDA / OpenCL kullanarak bizim için kodu çalıştırır ve optimize eder. Bizim için optimize edilmiş kodu üretmek için, Theano'nun sorunun kapsamını bilmesine ihtiyaç var; onu işlem ağacı (veya sembolik ifadelerin bir grafiği) olarak düşünün. Theano'nun halen aktif gelişme altında olduğunu ve birçok yeni özellik eklendi ve iyileştirmeler düzenli olarak yapıldı.

4.2.3 TensorFlow

TensorFlow, veri akış grafikleri kullanarak sayısal hesaplama için açık kaynaklı bir yazılım kütüphanesidir. Grafik kenarları aralarında iletilen çok boyutlu veri dizilerini (tensörler) temsil eder iken, grafikteki düğümler matematiksel işlemleri temsil eder. Esnek mimarisi, bir masaüstündeki, sunucudaki ya da bir tek API ile mobil cihazındaki bir ya da daha fazla CPU'ları ya da GPU'ları uygulamamıza olanak tanır. TensorFlow ilk olarak makine öğrenimi ve derin sinir ağları araştırması yürütmek amacıyla Google'ın makine zekâ araştırma organizasyonu içerisinde Google Brain ekibi üzerinde çalışan araştırmacılar ve

mühendisler tarafından geliştirildi, fakat bu sistem aynı zamanda diğer alanlarda da geniş bir yelpazede uygulanabilecek kadar geneldi.

4.2.4 Keras

Keras, Python'da yazılmış ve hem TensorFlow hem de Theano'nun üstünde çalışabilen üst düzey sinir ağları API'sıdır. Keras hızlı deney yapımına odaklanarak geliştirildi. Keras'ın en güçlü özelliği: kolay ve hızlı prototiple meye izin verir (kullanıcı dostu, modülerlik ve genişletilebilirdik), CPU'nun ve GPU'nun üzerinde sorunsuz bir şekilde çalışmasını ve ikisinin bağlanmasını desteklerken, aynı zamanda hem konvolüsyonel ağları hem de yinelenen ağları destekler.

4.2.5 Jupyter Notebook

Daha önce IPython olarak adlandırılan Jupyter, veri bilimcileri, araştırmacılar ve analistler tarafından yaygın şekilde benimsenmiştir. Jupyter'in dizüstü bilgisayar kullanıcı arabirimi, ekip iş birliği geliştirmek ve yeniden üretilebilir araştırma ve eğitim durumunu ilerletmek için anlatım metni, denklemler, etkileşimli görselleştirme ve görüntülerle yürütülebilir kodu karıştırmaya olanak tanır. Jupyter, Python ile başladı ve şimdi 50 farklı dil için çekirdeklere sahip ve IR Kernel, Jupyter için yerel R çekirdeği.

4.3 Veri Keşfetmek ve Hazırlamak

Bu proje için, meme kütlesi ince iğne aspiratı (FNA) olarak bilinen bir işlemin sayısallaştırılmış görüntülerinin tanımlayıcı özeliklerini içeren bir veri kümesi üzerinde bazı derin öğrenme tekniklerini uygulamak istedim. Her bir hücre çekirdeği için bir kimlik numarası ve teşhis (ikili gösterimlerine dönüştürülmüş: kötü huylu = 1, iyi huylu = 0) olmak üzere hesaplanan toplam da 30 özellik var.

4.3.1 Derin Öğrenme Ortamının Kurulması

a. Anaconda yükleyicisini indirdikten sonra [64], Windows'a Anaconda kuruyoruz.

b. Anaconda isteminden Theano'yu bu komutu kullanarak kurun [conda install -c conda-forge theano]
c. Anaconda isteminden Tensorflow'yu bu komutu kullanarak kurun [conda install -c conda-forge tensorflow]
d. Anaconda isteminden Kerası bu komutu kullanarak kurun [conda install -c conda-forge keras]

4.3.2 Modülleri Yüklemek

Modülleri Python ortamımıza yüklüyoruz. Benim durumumda, bir Windows 10 ortamında çalışırken Jupyter Notebook kullanıyorum.

```
import pandas as pd    # pandas is a dataframe library
import matplotlib.pyplot as plt      # matplotlib.pyplot plots data
import numpy as np
import theano
import keras
import tensorflow
%matplotlib inline
from sklearn import metrics
import matplotlib.pyplot as plt # side-stepping mpl backend
import matplotlib.gridspec as gridspec # subplots
import seaborn as sns # Danker visuals
from keras.models import Sequential
from keras.layers import LSTM
from keras.layers.core import Dropout, Flatten, Activation, Dense
from keras.layers.convolutional import Convolution2D, Convolution1D,MaxPooling1D,
#Import models from scikit learn module:
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.cross_validation import KFold #For K-fold cross validation
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier, export_graphviz
```

4.3.3 Veri Seti Yüklemesi

Bu bölüm için, read_csv işlevini kullanarak verileri bir Pandas veri çerçevesine yükleyeceğim.

```
df= pd.read_csv("/anaconda3//data/WBCD.csv",header = 0)
```

4.3.4 Keşfedici Veri Analizi

Derin öğrenme 'de önemli bir süreçtir, verileriniz için bir fikir edinmek için keşif analizi yapıyor. Birçok kişi tahmini modellemeye geçmeye çalışacaktır, ancak verilerinizin nasıl etkileşime girdiğini bilmiyorsanız, model seçerken. Analizin ilk adımı veri çerçevesi boyutlarını öğrenmektir. Burada, birinci çıktının satır uzunluğu ve ikinci çıktının sütun uzunluğu olduğu veri çerçevemizin uzunluklarını bize vermek için.shape fonksiyonunu kullanıyoruz. İkinci adım veri kümesindeki değişkenlerinizin veri türleridir. Bu veri seti için, tüm değişkenlerin ölçümlerden oluştuğunu biliyoruz (teşhis hariç) (Bakınız Tablo 4.2).

print("Veri setinin özellikleri ve veri türleri:\n"**,** df**.**dtypes**)**

Tablo 4.2: WBCD veri setinin nitelikleri

Sütun Adı	Veri tipi
Id	int64
Diagnosis	Object
radius_mean	float64
texture_mean	float64
perimeter_mean	float64
area_mean	float64
smoothness_mean	float64
compactness_mean	float64
concavity_mean	float64
concave points_meanpoints_mean	float64
symmetry_mean	float64
fractal_dimension_mean	float64
radius_se	float64
texture_se	float64
perimeter_se	float64
area_se	float64
smoothness_se	float64
compactness_se	float64
concavity_se	float64
concave points_se	float64
symmetry_se	float64
fractal_dimension_se	float64
radius_worst	float64
texture_worst	float64
perimeter_worst	float64
area_worst	float64
smoothness_worst	float64
compactness_worst	float64
concavity_worst	float64
concave points_worst	float64
symmetry_worst	float64
fractal_dimension_worst	float64

4.3.5 Veri Temizlemesi

Bir veri kümesinden bozuk veya yanlış kayıtları tespit etme ve düzeltme (veya kaldırma) işlemidir. Verilerin eksik, yanlış, hatalı veya alakasız kısımlarını belirlemek ve daha sonra kirli veya kaba verileri değiştirmek veya silmek anlamına gelir. Kimlik sütunu herhangi bir veri kümesi özellikleri göstermedi ve herhangi bir şekilde sonucu etkilemedi ve belki de analizde kötü sonuçlar doğurdu. Bu yüzden, bu sütun çıkararak algoritma daha iyi çalışır.

```
df.drop('id',axis=1,inplace=True)
df.drop('Unnamed: 32',axis=1,inplace=True)
```

4.3.6 Boş Değerleri Kontrol Etmek

Boş değerler birkaç soruna neden olur. Boş bir değer temelde tanımlanmamış bir değerdir.

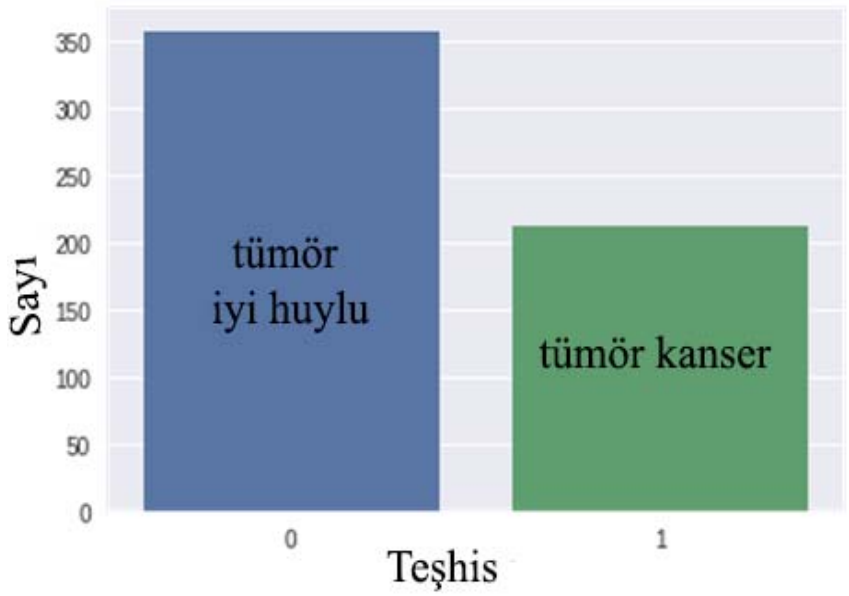
```
df.isnull().values.any(): false
```

Yukarıda mükemmel bir şekilde gösterildiği gibi, hiçbir boş değer bulamadık.

4.3.7 Değişken Türlerini Dönüştürmek

Pandas veri çerçevesini kullanarak, Analiz yaparken önemli olan, değişken türlerini uygun gösterime dönüştürmektir. Teşhis değişken değerlerini 1 = kötü ve 0 = iyi olarak belirleteceğiz.

```
df.diagnosis.unique()
df.describe()
plt.hist(df['diagnosis'])
plt.title('Diagnosis (M=1, B=0)')
plt.show()
```



Şekil 4.2: Kötü ve iyi sayılarını gösterecek

4.3.8 Sınıf Dengesizliği

Tanı konusundaki sayımız önemlidir, çünkü bu sayı makine öğreniminde ve derin öğrenme uygulamaları içindeki sınıf dengesizliği tartışmasını ortaya çıkarmaktadır. Sınıf dengesizliği, bir veri kümesindeki bir sınıfın diğer sınıf tarafından sayılamayacak kadar fazla olduğu anlamına gelir. Sınıf dengesizliği, bir sınıfın veri kümesinin yüzde 10-20'sini doldurması durumunda ortaya çıkar.

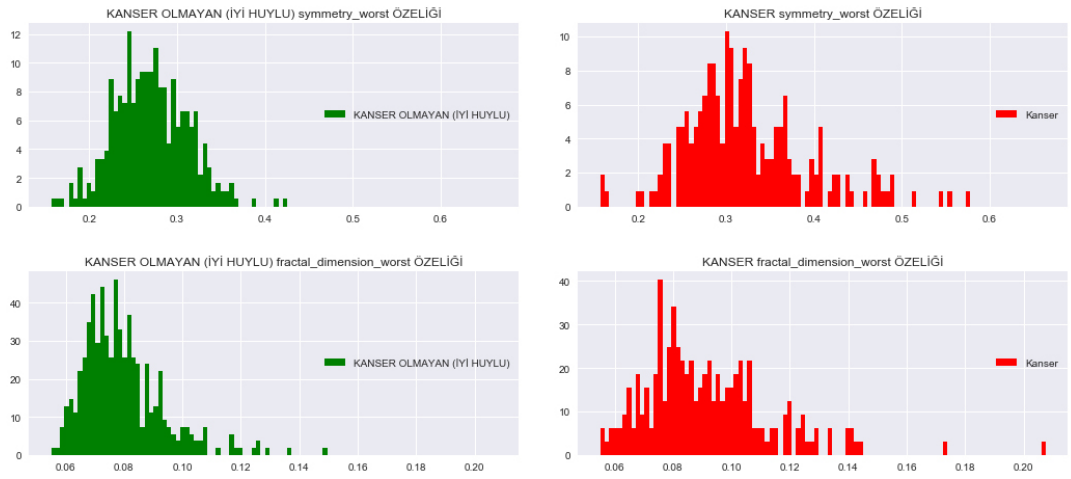
```
num_obs = len(df)
print (num_obs)
num_true = len(df.loc[df['diagnosis'] == 1])
num_false = len(df.loc[df['diagnosis'] == 0])
print("Malign vakaların sayısı:  {0} ({1:2.2f}%)".format(num_true, (float
(num_true)/num_obs) * 100))
print("İyi huylu vakaların sayısı: {0} ({1:2.2f}%)".format(num_false,
(float(num_false)/num_obs) * 100))

Malign vakaların sayısı:  212 (37.26%)
İyi huylu vakaların sayısı: 357 (62.74%)
```

Burada görebildiğimiz gibi, veri kümemiz sınıfsal dengesizlikten dolayı acı çekmiyor; bu nedenle analizimize devam edebiliyoruz.

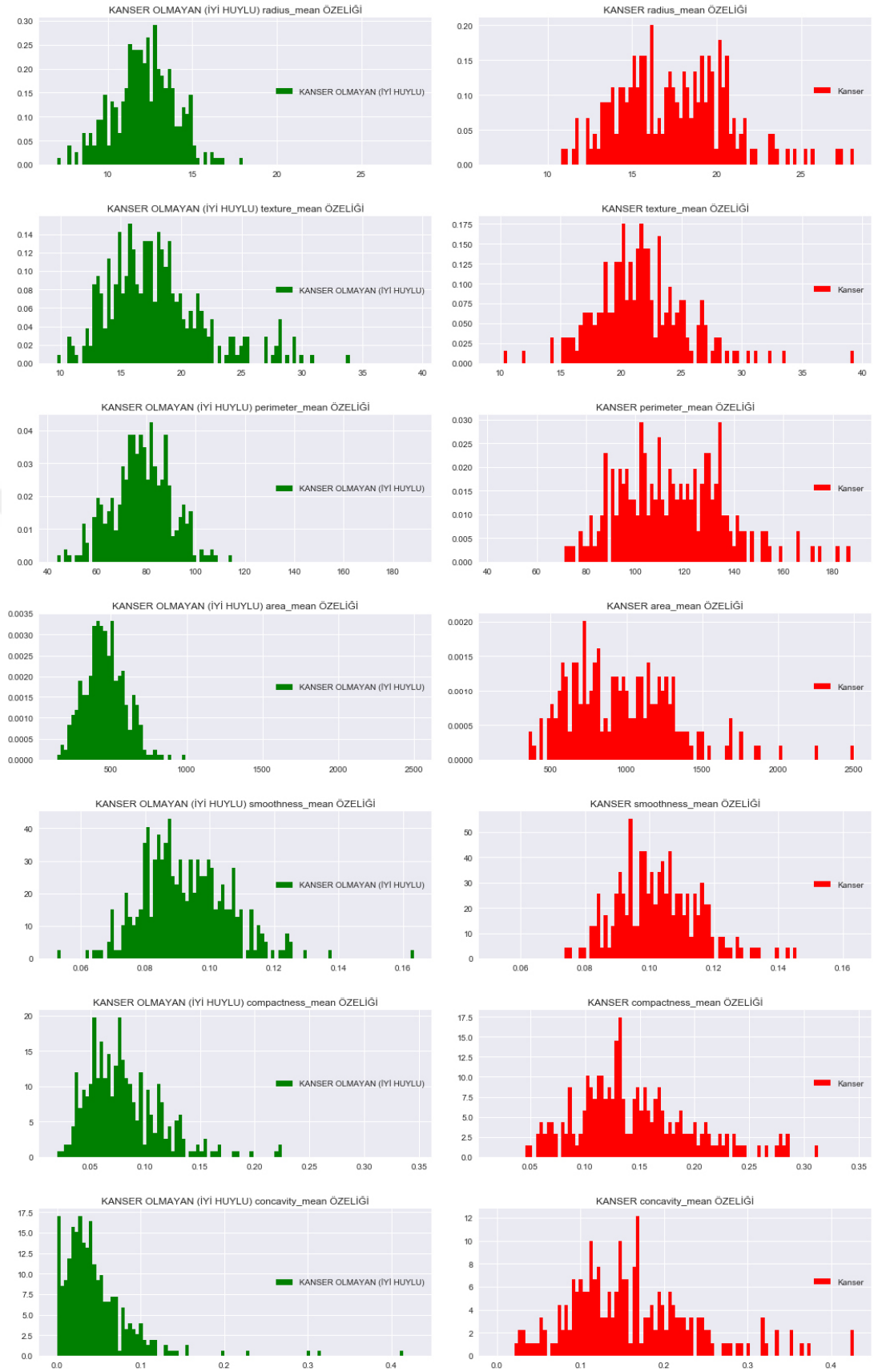
4.3.9 Görsel Keşif Analizi (Çekirdek Özellikleri ve Tanı)

Matplotlib kullanılarak üretilmiş olan çok güçlü istatistiksel grafikleri içeren modülden yararlanıyoruz. Her bir çizim içerisinde, biz tanının iki sınıfını renklendirebiliriz ki bu kötü (Malignant) ile iyi (Benign) arasındaki farkı kolayca ayırt edebildiğimiz açıktır. Bazı değişken yanı sıra, etkileşimler neredeyse doğrusal bir ilişkiye sahip.

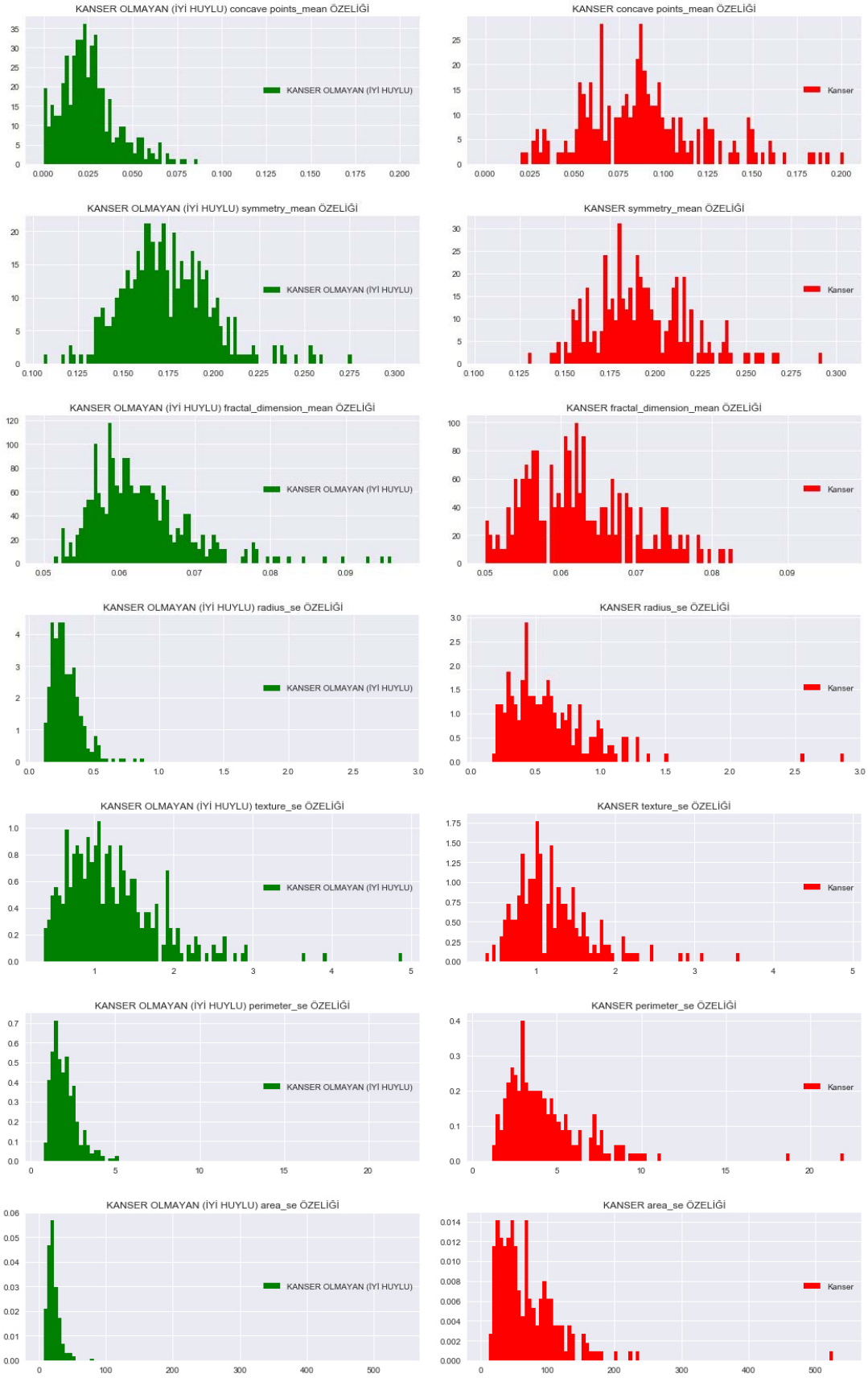
```
features_mean=list(df.columns[1:31])
dfM=df[df['diagnosis'] ==1]
dfB=df[df['diagnosis'] ==0]

plt.rcParams.update({'font.size': 28})
fig, axes = plt.subplots(nrows=30, ncols=1, figsize=(8,100))
x = axes.ravel()
for idx,ax in enumerate(x):
    ax.figure
    binwidth= (max(df[features_mean[idx]]) - min(df[features_mean[idx]]))/100
    ax.hist([dfM[features_mean[idx]]], bins=np.arange(min(df[features_mean[idx]]),
max(df[features_mean[idx]]) + binwidth, binwidth) ,stacked=True, normed = True,
label=['Kanser'],color=['R'])
    ax.legend(loc='right')
    ax.set_title("KANSER "+format(features_mean[idx])+" ÖZELİĞİ")
plt.tight_layout()
plt.show()
```



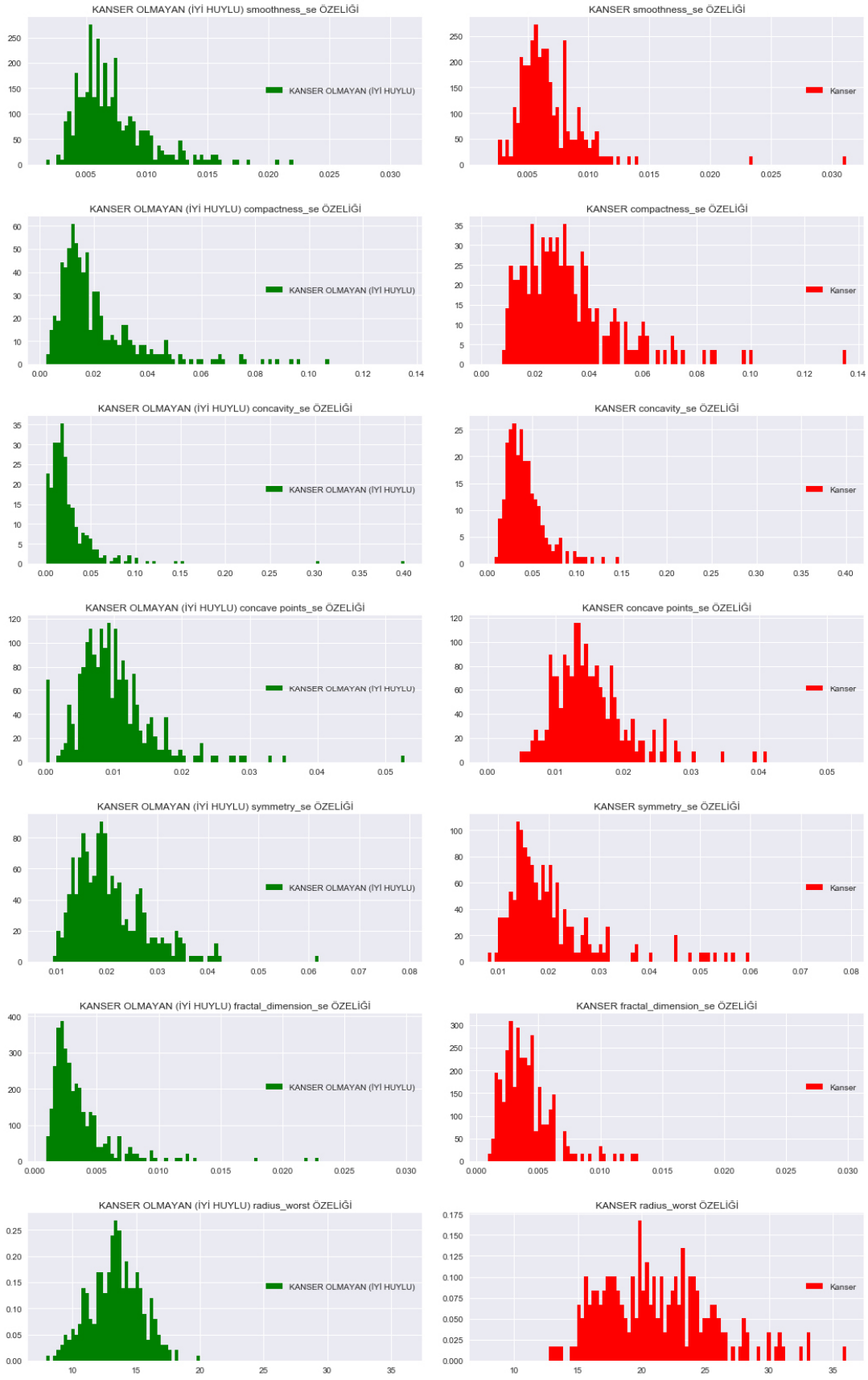
Şekil 4.3: Veri setinin görsel analizi 1



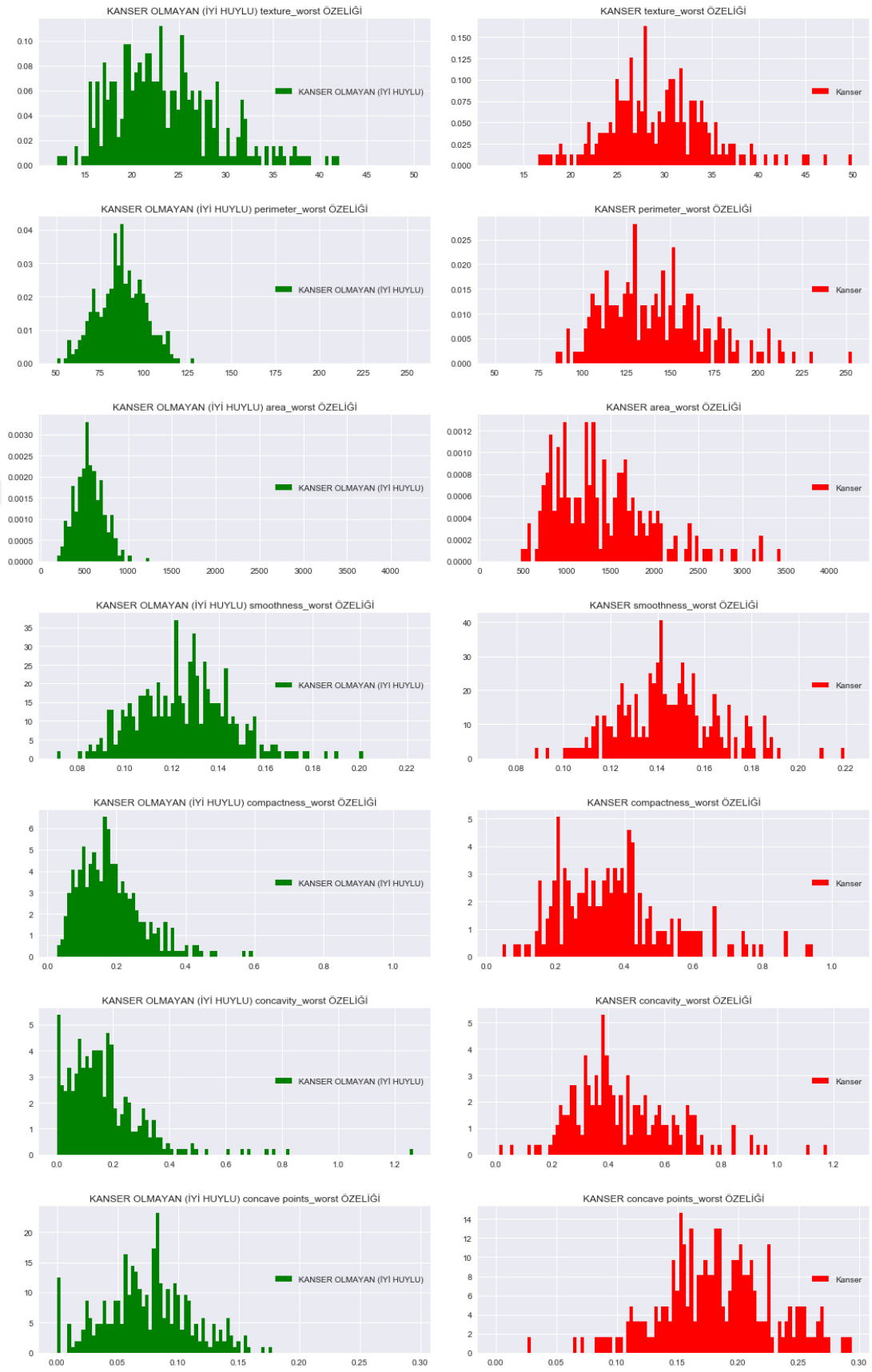
Şekil 4.4: Veri setinin görsel analizi 2



Şekil 4.5: Veri setinin görsel analizi 3



Şekil 4.6: Veri setinin görsel analizi 4



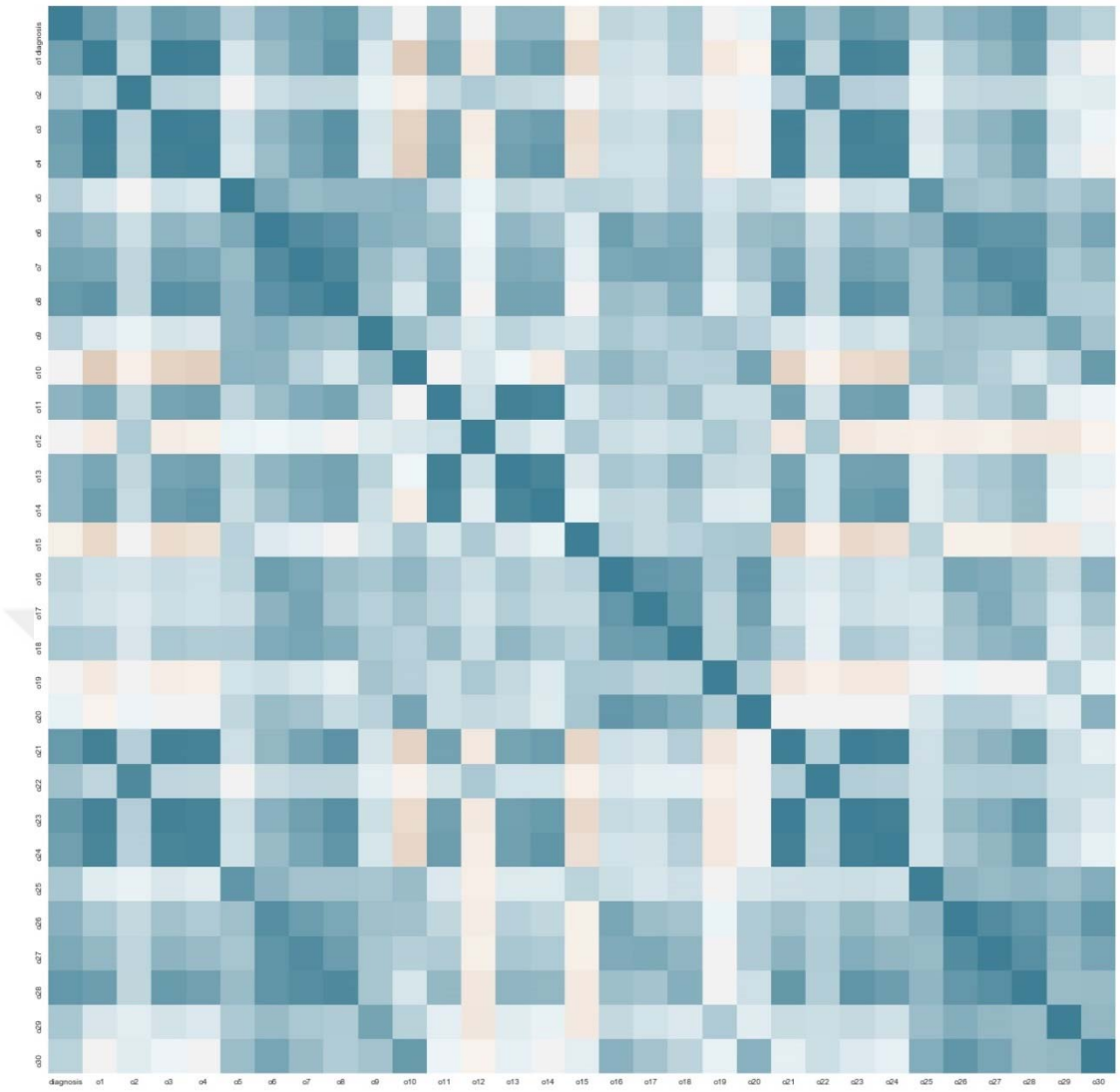
Şekil 4.7: Veri setinin görsel analizi 5

Bu grafiklerden anlaşılıyor ki, kanserin sınıflandırılmasında, pürüzsüzlük, simetri dokunun ortalama değeri ya da fraktal boyutun kullanıldığı, hücre yarıçapı, çevre, alan, kompaktlık, konkavite ve içbükey noktaların ortalama değerleri, bir tanının varlığını göstermez. Histogramların herhangi birinde, daha fazla temizleme garantisi veren belirgin bir belirsizlik yoktur.

4.3.10 Bağıntı (Correlation)

Bir Makine öğrenme konsepti içindeki değişken bağıntı, ancak bir veri kümesi çok fazla değişkene sahip olduğunda orada tehlikeli olabilir. Bir makine öğrenme ortamında iki özellik (veya daha fazlası) birbirine tam olarak bağlandığında, ardından birisi sürecinize ek bilgi ekleme potansiyeli bulunmadığından, bu yüzden özellik çıkarma, boyutsaldık lanetinde yardımcı olan boyutların azaltılmasına yardımcı olur. Hesaplama süresindeki azaltmanın ek olarak, eğer bir kez, özellik çıkarımının değişkenlerimiz içinde görsel olarak çok fazla korelasyon görmesi gerektiği sonucuna varırız.

```
corr = df.corr(method = 'pearson') # Bağıntı Matresi
plt.subplots(figsize=(50, 30))
# Renk haritası oluşturmak
cmap = sns.diverging_palette(4, 990900, as_cmap= True)
#Isı haritasını maskeyle çizimek
sns.heatmap(corr, cmap=cmap,square=True, xticklabels=True, yticklabels=True)
```



Şekil 4.8: Bağıntı (Correlation)

Veri setimizin çoğunlukla pozitif korelasyon içerdiğini görebiliriz ve ayrıca bizim özelleştirdiğimiz dağılım tablosu matrisinde yer alan beş bağımlı değişkenin güçlü bir korelasyona sahip olduğunu tekrarladığını görebiliriz.

4.3.11 Eğitim Seti ve Test Seti Oluşturma

Biz veri setini yüzde 70-30 sahip rastgele seçilecek olan test seti ve eğitim seti olmak üzere ayırdık.

```
from sklearn.cross_validation import train_test_split
feature_col_names = ['radius_mean', 'texture_mean', 'perimeter_mean', 'area_mean',
'smoothness_mean', 'compactness_mean', 'concavity_mean', 'concave points_mean',
```

```
'symmetry_mean', 'fractal_dimension_mean', 'radius_se', 'texture_se', 'perimeter_se', 'area_se',
'smoothness_se',
'compactness_se', 'concavity_se', 'concave points_se', 'symmetry_se', 'fractal_dimension_se',
'radius_worst', 'texture_worst', 'perimeter_worst', 'area_worst', 'smoothness_worst'
'compactness_worst', 'concavity_worst', 'concave points_worst',
'symmetry_worst', 'fractal_dimension_worst']
predicted_class_names = ['diagnosis']
X = df[feature_col_names].values
y = df[predicted_class_names].values
split_test_size = 0.30
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=split_test_size,
random_state=42)
# test_size = 0.3 is 30%, 42 is the answer to everything
print("{0:0.2f}% Eğitim setinde".format((float(len(X_train))/len(df.index)) * 100))
print("{0:0.2f}% Test setinde".format((float(len(X_test))/len(df.index)) * 100))
```

69.95% Eğitim setinde

30.05% Test setinde

Sonra, Eğitim ve test setlerinde sınıf dengesizliğini kontrol ederiz.

```
print("Eğitim setinde bulunan kanser : {0} ({1:0.2f}%)".format(len(y_train[y_train[:] == 1]),
(float(len(y_train[y_train[:] == 1]))/len(y_train) * 100.0)))
print("Test setinde bulunan iyi huylu: {0} ({1:0.2f}%)".format(len(y_train[y_train[:] == 0]),
(float(len(y_train[y_train[:] == 0]))/len(y_train) * 100.0)))
print("")
print("Test setinde bulunan kanser:{0}({1:0.2f}%)".format(len(y_test[y_test[:] == 1]),
(float(len(y_test[y_test[:] == 1]))/len(y_test) * 100.0)))
print("Test setinde bulunan iyi huylu:{0}({1:0.2f}%)".format(len(y_test[y_test[:]== 0]),
(float(len(y_test[y_test[:] == 0]))/len(y_test) * 100.0)))
```

Eğitim setinde bulunan kanser: 149 (37.44%)

Eğitim setinde bulunan iyi huylu: 249 (62.56%)

Test setinde bulunan kanser: 63 (36.84%)

Test setinde bulunan iyi huylu: 108 (63.16%)

Eğitim seti ve test seti oranlarının seçimi için belli bir kuralar olmadığından dolayı değişik oranları kullanacağız (test seti: %15, %20, %25, %30).

4.4 Modelleme ve Sonuçlar

4.4.1 Değerlendirme Metrikleri

Sınıflandırıcı sistemin dört muhtemel durumu vardır, doğru pozitif (DP), gerçek negatifler (DN), yanlış pozitifler (YP) ve yanlış negatifler (YN). Doğru Pozitif (DP), pozitif olarak tahmin edilen numunelerin sayısını ifade eder ve aslında pozitiftir. Yanlış Pozitif (YP), pozitif olarak tahmin edilen numunelerin sayısını ifade eder, ancak aslında negatiftir. Yanlış negatifleri (YN), negatif olarak tahmin edilen numunelerin sayısını ifade eder, ancak aslında pozitiftir. Doğru negatifler (DN), negatif olarak tahmin edilen örneklerin sayısını ifade eder ve aslında negatiftir. Bu olası durumlar, Tablo 4.3'de gösterildiği gibi bir karışıklık matrisinde yer alır.

Tablo 4.3: Doğru sınıflandırılmış örnekler DN ve DP tarafından temsil edilir, Yanlış sınıflandırılmış örnekler YP ve YN ile temsil edilir.

	Tahmin edilen Negatif	Tahmin edilen Pozitif
Gerçek Negatif	DN	YP
Gerçek Pozitif	YN	DP

Doğruluk, performans ölçüsüdür, doğru tahmin edilen gözlemin toplam gözlemlere göre oranıdır. Doğruluk ne kadar yüksekse, modelimiz iyi çalışıyor demektir. Doğruluk aşağıda gösterilen şekilde tanımlanır.

$$\text{Doğruluk} = (\text{DP}+\text{DN})/(\text{YP}+\text{YN}+\text{YN}+\text{YP}) \quad \text{(Denklem } \mathbf{4.1}\text{)}$$

Precision, gerçekten sınıflandırılmış pozitiflerin tüm tahmin edilen pozitiflere göre oranıdır.

$$\text{Precision} = \frac{DP}{DP + YP} \quad \text{(Denklem } \mathbf{4.2}\text{)}$$

Sensitivity (recall), TPR (Doğru Pozitif Oran) olarak da bilinir, doğru tahmin edilen pozitiflerin sayısıyla gerçek pozitiflerin toplamına oranıdır.

$$TPR = \frac{DP}{YN + DP} \qquad \text{(Denklem 4.3)}$$

F1 Score, Precision 'nin ve Recall 'nin ağırlıklı ortalamasıdır. Bu nedenle, F1 Score hem yanlış pozitif hem de yanlış negatifleri hesaba katıyor.

$$F1\ Score = \frac{2 * (\text{Recall } * \text{ Precision})}{(Recall + Precistion)} \qquad \text{(Denklem 4.4)}$$

4.4.2 Model Tanımı

Keras'ın ilk yapı taşı model tanımıdır. Tüm modelleri tasarladığımız zaman ilk adım, modeli tanımlamaktır. Bu çalışmada modelleri Sequential olarak tanımlarız. Sequential model, Sinir ağları katmanlarının doğrusal bir boru hattıdır (yığını).

```
model = Sequential()
```

4.4.3 Tam Bağlantılı Yapay Sinir Ağı (Dense Fully Connected Model)

İlk önce 512 yapay nöron ile 30 adet değişken giriş olarak beklenir (aynı zamanda özellikleri olarak da bilinir) ve aktivasyon fonksiyonu Sigmoid seçeriz.

```
model.add(Dense(596, input_dim=30))
model.add(Activation('sigmoid'))
```

İlk gelişme, ağımıza ek katmanlar eklemektir. Giriş katmanından sonra, 256 nöron ile ve Sigmoid aktivasyon fonksiyonuna sahip olan tam bağlı katman ekleriz.

```
model.add(Dense(30))
model.add(Activation('sigmoid'))
```

Çıkış katmanı 1 nöron tam bağlantılı katman olacaktır.

```
model.add(Dense(1))
```

Modeli tanımladıktan sonra, onu Keras tarafından yürütülebilecek şekilde tamamlamamız gerekmektedir. Derleme sırasında yapılacak birkaç seçenek var:

a. Modelimizi eğitirken ağırlıkları güncellemek için kullanılan özel algoritma olan optimizasyon yöntemini seçmeliyiz. İkinci Dereceden Hata Fonksiyonu (mean_squared_error) kullandık.
b. Optimizer tarafından ağırlıkların alanına gitmek için kullanılan hatta fonksiyonunu seçmeliyiz. ADAM Optimizasyon Yöntemini kullandık
c. Eğitimli modeli değerlendirmek gerekmektedir. Metrikler kullandığımız seçenek Doğruluk yöntemedir (Accuracy) bu seçenek Hedeflere göre doğru tahminlerin oranını bulur. Metrikler, kayıp fonksiyonlarına benzer, Tek fark, bir modeli eğitmek için değil, yalnızca bir modeli değerlendirmek için kullanılır.

```
model.compile(loss='mean_squared_error', optimizer='adam', metrics=['accuracy'])
```

son aşamada, Model daha sonra fit () fonksiyonuyla eğitilebilir ancak birkaç parametre belirlemek gerekmektedir.

a. epochs: Bu model eğitim kümesi maruz sayısıdır. Her yinelemede, optimizer ağırlıkları ayarlamaya çalışır, böylece kayıp fonksiyonu en aza indirgenir.
b. batch_size: Bu, optimizer bir ağırlık güncelleştirmesi gerçekleştirmeden önce gözlenen eğitim örneklerinin sayısıdır.
c. Verbose: Bilgi modelinin nasıl görüntüleneceğini gösterir.

Keras ‘ta bir modeli eğitmek çok basittir. 12000 yineleme bir model yapmak istediğimizi varsayalım.

```
from datetime import datetime
start_time = datetime.now()
model.fit(X_train, y_train, epochs=12000, batch_size=3000,
verbose=2,validation_data=(X_test, y_test))
end_time = datetime.now()
print('Duration: {}'.format(end_time - start_time))
```

Bu fonksiyonda modeli uygulamadan önce zaman hesaplamaya başlıyoruz, bu fonksiyonu kullanarak modeli eğitiyoruz ve eğitim süresini hesaplıyoruz.

Test seti = %30, Eğitim süresi: 0:09:06.031030

Test seti = %25, Eğitim süresi: 0:09:47.359295

Test seti = %20, Eğitim süresi: 0:07:35.127192

Test seti = %15, Eğitim süresi: 0:07:30.108921

Toplam eğitilen parametre sayısı aşağıda gösterilmiştir.

Layer (type)	Output Shape	Param #
dense_16 (Dense)	(None, 596)	18476
dense_17 (Dense)	(None, 30)	17910
dense_18 (Dense)	(None, 1)	31

Total params: 36,417
Trainable params: 36,417
Non-trainable params: 0

Değerlendirme metriklerini bulmamız ağın doğru bir şekilde değerlendirilmesinde yardımcı olur.

```
predictions = model.predict(X_test)
rounded = [np.round(X_test) for X_test in predictions]
from sklearn import metrics
print("Accuracy:{0:4f}".format(metrics.accuracy_score(y_test, rounded)))
```

Sunulan Python kodu modelin doğruluk oranını test etmektedir. Kullandığımız kod test setinin sonuçlarını tahmin ederek modelin doğruluğun gösterir. Sonuçlar aşağıda sunulmaktadır.

```
Test seti = %30, Doğruluk oranı: 0.9708 ⟹ %97,08
Test seti = %25, Doğruluk oranı: 0.9790 ⟹ %97,90
Test seti = %20, Doğruluk oranı: 0.9737 ⟹ %97,37
Test seti = %15, Doğruluk oranı: 0.9651 ⟹ %96,51
```

Gördüğümüz gibi sonucalar gayet iyi. Tam bağlantılı model kullanarak %98,25 kadar doğruluk oranına sahip bir model elde ettik. Modeli iyi değerlendirmemiz için diğer değerlendirme metrikleri bulmamız gösterilmiştir.

```
print("Değerlendirme metrikleri ")
print("{0}".format(metrics.confusion_matrix(y_test, rounded, labels=[1,0])))
print("")
print("Sınıflandırma Raporu")
print(metrics.classification_report(y_test, rounded, labels=[1,0]))
```

Yukardaki kod değerlendirme raporunu ve sınıflandırılma raporunu sunuar. Değerlendirme metrikleri 4.4.1. bölümünde belirtiğimiz gibi DP, DN, YP ve YN

sayılarını göstermektedir. Sınıflandırma raporu precision, recall, f1-score ve örnek sayısı (support) gösterilmiştir.

1. Eğitim seti %70 test seti %30

Değerlendirme metrikleri
[[59 3]
 [1 107]]

Sınıflandırma Raporu

	precision	recall	f1-score	support
kanser 1	0.98	0.94	0.96	63
iyi huylu 0	0.97	0.99	0.98	108
avg / total	0.98	0.97	0.97	171

2. Eğitim seti %75 test seti %25

Değerlendirme metrikleri
[[51 3]
 [0 89]]

Sınıflandırma Raporu

	precision	recall	f1-score	support
kanser 1	1.00	0.94	0.97	54
iyi huylu 0	0.97	1.00	0.98	89
avg / total	0.98	0.98	0.98	143

3. Eğitim seti %80 test seti %20

Değerlendirme metrikleri
[[40 3]
[0 71]]

Sınıflandırma Raporu

	precision	recall	f1-score	support
kanser 1	1.00	0.93	0.96	43
iyi huylu 0	0.96	1.00	0.98	71
avg / total	0.97	0.97	0.97	114

4. Eğitim seti %85 test seti %15

Değerlendirme metrikleri
[[29 3]
[0 54]]

Sınıflandırma Raporu

	precision	recall	f1-score	support
kanser 1	1.00	0.91	0.95	32
iyi huylu 0	0.95	1.00	0.97	54
avg / total	0.97	0.97	0.96	86

Değişik test seti oranı kullanarak modeli eğiterek sonuçlar tablo 4.4'de sunulmaktadır.

Tablo 4.4: Tam Bağlantılı Modelin eğitim sonuçları

	Test seti %30	Test seti %25	Test seti %20	Test seti %15
Doğruluk	%97,08	%97,90	%97,37	%96,51
Precision	%98	%98	%97	%97
TPR (Recall)	%97	%98	%97	%97
F1 Score	%97	%98	%97	%96
Hata	0.0315	0.0307	0.0191	0.0256
Parametre Sayısı	36,417	36,417	36,417	36,417
Süre	0:08:06	0:07:47	0:07:35	0:07:30

4.4.4 Konvolüsyon sinir ağı (CNN)

Her tasarladığımız modelde ilk katman giriş katmanıdır. Burada kullandığımız doğrusal (linear) katmanı giriş katmanı olarak kullanılacaktır.

```
model.add(Activation('linear', input_shape=(None, 30)))
```

İlk aşama ağımıza ek katmanlar eklemektir. Giriş katmanından sonra, bir Konvolüsyon katmanı ekleriz, Konvolüsyon katmanı, bir tenor çıktı üretmek için katman girdisi ile tek bir uzamsal (veya geçici) boyut üzerinde Konvolüsyon çekirdeği oluşturur. İlk gizli katman, kullandığımız Konvolüsyon katmanı 596 adet filtreye sahiptir, her filtre Kullanılacak Konvolüsyon çekirdeği sayısıdır (Örneğin, çıktılının boyutsal lığı). Çekirdek boyutu (kernel_size) sayısı üç kullanacağız, Çekirdek boyutu 1D Konvolüsyon penceresinin uzunluğunu belirtir. Padding = 'same', dolgu kullanıldığı anlamına gelir.

```
model.add(Activation('linear', input_shape=(None, 30)))
model.add(Convolution1D(450, 3, padding="same"))
```

İkinci gizli katman, Sigmoid aktivasyona sahip bir katmandır. Üçüncü gizli katman, Relu aktivasyona sahip bir katmandır. Dördüncü gizli katman, Sigmoid aktivasyona sahip bir katmandır. Beşinci gizli katman, 30 adet filtre ve çekirdek boyutu 3 olan bir Konvolüsyon katmanıdır. Altıncı gizli katman, Sigmoid

aktivasyona sahip bir katmandır. Çıktı katmanı, 1 adet filtre ve çekirdek boyutu 1 olan Konvolüsyon katmanıdır.

```
model.add(Activation('hard_sigmoid'))
model.add(Activation('Relu'))
model.add(Activation('hard_sigmoid'))
model.add(Convolution1D(30, 3, padding='same'))
model.add(Activation('hard_sigmoid'))
model.add(Convolution1D(1, 1, padding='same'))
```

Modeli tanımladıktan sonra, onu Keras tarafından yürütülebilecek şekilde tamamlarız gerekmektedir. 4.4.3. bölümünde belirtiğimiz gibi modeli tamamlarız.

```
model.compile(loss='mean_squared_error', optimizer='Adam', metrics=['accuracy'])
model.fit(X_train, y_train, epochs=2400, batch_size=3000,  verbose=2,
validation_data=(X_test, y_test))
```

Bu fonksiyonda modeli uygulamadan önce zaman hesaplamaya başlıyoruz, bu fonksiyonu kullanarak modeli eğitiyoruz ve eğitim süresini hesaplıyoruz.

Test seti = %30, Eğitim süresi: 0:01:33.374628

Test seti = %25, Eğitim süresi: 0:01:37.095042

Test seti = %20, Eğitim süresi: 0:01:39.721578

Test seti = %15, Eğitim süresi: 0:01:45.041809

Toplam eğitilen parametre sayısı aşağıda gösterilmiştir.

Layer (type)	Output Shape	Param #
activation_26 (Activation)	(None, None, 30)	0
conv1d_16 (Conv1D)	(None, None, 596)	54236
activation_29 (Activation)	(None, None, 596)	0
conv1d_17 (Conv1D)	(None, None, 30)	53670
activation_30 (Activation)	(None, None, 30)	0
conv1d_18 (Conv1D)	(None, None, 1)	31

Total params: 107,937
Trainable params: 107,937
Non-trainable params: 0

4.4.3. bölümünde belirtiğimiz gibi Kullandığımız kod test setinin sonuçlarını tahmin ederek modelin doğruluğun gösterir. Sonuçlar aşağıda sunulmaktadır.

Test seti = %30, Doğruluk oranı: 0.9825 ⟹ %98,25
Test seti = %25, Doğruluk oranı: 0.9930 ⟹ %99,30
Test seti = %20, Doğruluk oranı: 0.9737 ⟹ %97,37
Test seti = %15, Doğruluk oranı: 0.9651 ⟹ %96,51

Konvolüsyon model kullanarak %100 kadar doğruluk oranına sahip bir model elde ettik. Modeli iyi değerlendirmemiz için diğer değerlendirme metrikleri bulmamız gerekmektedir. 4.4.1. ve 4.4.3. bölümünde belirtiğimiz gibi değerlendirme metrikleri ve sınıflandırma raporu aşağıda sunulmaktadır.

1. Eğitim seti %70 test seti %30

Değerlendirme metrikleri
[[61 2]
[1 107]]
Sınıflandırma Raporu

		precision	recall	f1-score	support
kanser	1	0.98	0.97	0.98	63
iyi huylu	0	0.98	0.99	0.99	108
avg / total		0.98	0.98	0.98	171

2. Eğitim seti %75 test seti %25

Değerlendirme metrikleri
[[53 1]
[0 89]]
Sınıflandırma Raporu

		precision	recall	f1-score	support
kanser	1	1.00	0.98	0.99	54
iyi huylu	0	0.99	1.00	0.99	89
avg / total		0.99	0.99	0.99	143

3. Eğitim seti %80 test seti %20

Değerlendirme metrikleri
[[41 2]
[1 70]]
Sınıflandırma Raporu

		precision	recall	f1-score	support
kanser	1	0.98	0.95	0.96	43
iyi huylu	0	0.97	0.99	0.98	71
avg / total		0.97	0.97	0.97	114

4. Eğitim seti %85 test seti %15

Değerlendirme metrikleri
[[30 2]
[1 53]]

Sınıflandırma Raporu

	precision	recall	f1-score	support
kanser 1	0.97	0.94	0.95	32
iyi huylu 0	0.96	0.98	0.97	54
avg / total	0.97	0.97	0.96	86

Tablo 4.5: Konvolüsyon modelin eğitim sonuçları

	Test seti %30	Test seti %25	Test seti %20	Test seti %15
Doğruluk	%98,25	%99,30	%97,37	%96,51
Precision	%98	%99	%97	%97
TPR (Recall)	%98	%99	%97	%97
F1 Score	%98	%99	%97	%96
Hata	0.0231	0.0225	0.0431	0.0355
Parametre Sayısı	107,937	107,937	107,937	107,937
Süre	0:01:40	0:01:37	0:01:39	0:01:45

4.4.5 Tekrarlayan Sinir Ağı (RNN)

Önceki bölümlerde belirtiğimiz gibi (3.5.) RNN Ağları üç şekille bölünür. Her üç şekilde aynı model kriterlerini kullanacağız. Giriş katmanı tekrarlayan sinir ağlarının bir türü olacaktır. Giriş katmanından sonra, gizli katman tekrarlayan sinir ağlarının bir türü olacaktır. Çıkış katmanı tam bağlantılı sinir ağıdır. Modeli tanımladıktan sonra, onu Keras tarafından yürütülebilecek şekilde tamamlarız (4.4.3. bölümünde açıklamıştık).

4.4.5.1 Basit tekrarlayan ağ (SRN)

4.4.3. bölümünde belirtiğimiz gibi modeli tanımlarız.

```
model = Sequential()
model.add(SimpleRNN(596, input_shape=(None,30),return_sequences=True))
model.add(SimpleRNN(30, return_sequences=True))
model.add(Dense(1))
```

Modeli tanımladıktan sonra, onu Keras tarafından yürütülebilecek şekilde tamamlarız gerekmektedir. 4.4.3. bölümünde belirtiğimiz gibi modeli tamamlarız.

```
model.compile(loss='mean_squared_error', optimizer='adam', metrics=['accuracy'])
```

```
from datetime import datetime
start_time = datetime.now()
model.fit(X_train, y_train, epochs=600, batch_size=600,  verbose=2, validation_data=(X_test,
y_test))
end_time = datetime.now()
print('Duration: {}'.format(end_time - start_time))
```

Bu fonksiyonda modeli uygulamadan önce zaman hesaplamaya başlıyoruz, bu fonksiyonu kullanarak modeli eğitiyoruz ve eğitim süresini hesaplıyoruz.

Test seti = %30, Eğitim süresi: 0:01:31.536736

Test seti = %25, Eğitim süresi: 0:01:40.650430

Test seti = %20, Eğitim süresi: 0:01:49.818946

Test seti = %15, Eğitim süresi: 0:02:45.353185

Toplam eğitilen parametre sayısı aşağıda gösterilmiştir.

```
Layer (type)                 Output Shape              Param #
=================================================================
simple_rnn_1 (SimpleRNN)     (None, None, 596)         373692
_________________________________________________________________
simple_rnn_2 (SimpleRNN)     (None, None, 30)          18810
_________________________________________________________________
dense_1 (Dense)              (None, None, 1)           31
=================================================================
Total params: 392,533
Trainable params: 392,533
Non-trainable params: 0
```

4.4.3. bölümünde belirtiğimiz gibi Kullandığımız kod test setinin sonuçlarını tahmin ederek modelin doğruluğun gösterir. Sonuçlar aşağıda sunulmaktadır.

```
Test seti = %30, Doğruluk oranı: 0.9708 ⟹ %97,08
Test seti = %25, Doğruluk oranı: 0.9650 ⟹ %96,50
Test seti = %20, Doğruluk oranı: 0.9649 ⟹ %96,49
Test seti = %15, Doğruluk oranı: 0.9651 ⟹ %96,51
```

Konvolüsyon model kullanarak %98,25 kadar doğruluk oranına sahip bir model elde ettik. Modeli iyi değerlendirmemiz için diğer değerlendirme metrikleri bulmamız gerekmektedir. 4.4.1. ve 4.4.3. bölümünde belirtiğimiz gibi değerlendirme metrikleri ve sınıflandırma raporu aşağıda sunulmaktadır.

1. Eğitim seti %70 test seti %30

```
Değerlendirme metrikleri
 [[ 61   2]
 [  3 105]]
```

Sınıflandırma Raporu

		precision	recall	f1-score	support
kanser	1	0.95	0.97	0.96	63
iyi huylu	0	0.98	0.97	0.98	108
avg / total		0.97	0.97	0.97	171

2. Eğitim seti %75 test seti %25

Değerlendirme metrikleri
[[52 2]
[3 86]]
Sınıflandırma Raporu

		precision	recall	f1-score	support
kanser	1	0.95	0.96	0.95	54
iyi huylu	0	0.98	0.97	0.97	89
avg / total		0.97	0.97	0.97	143

3. Eğitim seti %80 test seti %20

Değerlendirme metrikleri
[[41 2]
[1 70]]

Sınıflandırma Raporu

		precision	recall	f1-score	support
kanser	1	0.95	0.95	0.95	43
iyi huylu	0	0.97	0.97	0.97	71
avg / total		0.96	0.96	0.96	114

4. Eğitim seti %85 test seti %15

Değerlendirme metrikleri
[[30 2]
[1 53]]

Sınıflandırma Raporu

		precision	recall	f1-score	support
kanser	1	0.97	0.94	0.95	32
iyi huylu	0	0.96	0.98	0.97	54
avg / total		0.97	0.97	0.96	86

Tablo 4.61: SRN modelin eğitim sonuçları

	Test seti %30	Test seti %25	Test seti %20	Test seti %15
Doğruluk	%97,08	%96,50	%96,49	%96,51
Precision	%97	%97	%96	%97
TPR (Recall)	%97	%97	%96	%97
F1 Score	%97	%97	%96	%96
Hata	0.0319	0.0320	0.0283	0.0355
Parametre Sayısı	392,533	392,533	392,533	392,533
Süre	0:01:31	0:01:40	0:01:49	0:02:45

4.4.5.2 Uzun Kısa Dönem ağları (LSTM)

4.4.3. bölümünde belirtiğimiz gibi modeli tanımlarız.

```
model = Sequential()
model.add(LSTM(596, input_shape=(None,30),return_sequences=True))
model.add(LSTM(30, return_sequences=True))
model.add(Dense(1))
```

Modeli tanımladıktan sonra, onu Keras tarafından yürütülebilecek şekilde tamamlarız gerekmektedir. 4.4.3. bölümünde belirtiğimiz gibi modeli tamamlarız.

```
model.compile(loss='mean_squared_error', optimizer='adam', metrics=['accuracy'])
from datetime import datetime
start_time = datetime.now()
model.fit(X_train, y_train, epochs=900, batch_size=900,  verbose=2, validation_data=(X_test, y_test))
end_time = datetime.now()
print('Duration: {}'.format(end_time - start_time))
```

Bu fonksiyonda modeli uygulamadan önce zaman hesaplamaya başlıyoruz, bu fonksiyonu kullanarak modeli eğitiyoruz ve eğitim süresini hesaplıyoruz.

Test seti = %30, Eğitim süresi: 0:41:47.495960

Test seti = %25, Eğitim süresi: 0:41:36.627618

Test seti = %20, Eğitim süresi: 0:48:05.734825

Test seti = %15, Eğitim süresi: 0:49:10.479739

Toplam eğitilen parametre sayısı aşağıda gösterilmiştir.

Layer (type)	Output Shape	Param #
lstm_3 (LSTM)	(None, None, 596)	1494768
lstm_4 (LSTM)	(None, None, 30)	75240
dense_2 (Dense)	(None, None, 1)	31

Total params: 1,570,039
Trainable params: 1,570,039
Non-trainable params: 0

4.4.3. bölümünde belirtiğimiz gibi Kullandığımız kod test setinin sonuçlarını tahmin ederek modelin doğruluğun gösterir. Sonuçlar aşağıda sunulmaktadır.

Test seti = %30, Doğruluk oranı: 0.9649 ⟹ %96,49
Test seti = %25, Doğruluk oranı: 0.9790 ⟹ %97,90
Test seti = %20, Doğruluk oranı: 0.9737 ⟹ %97,37
Test seti = %15, Doğruluk oranı: 0.9651 ⟹ %96,51

LSTM model kullanarak %98,25 kadar doğruluk oranına sahip bir model elde ettik. Modeli iyi değerlendirmemiz için diğer değerlendirme metrikleri bulmamız gerekmektedir. 4.4.1. ve 4.4.3. bölümünde belirtiğimiz gibi değerlendirme metrikleri ve sınıflandırma raporu aşağıda sunulmaktadır.

1. Eğitim seti %70 test seti %30

Değerlendirme metrikleri
[[58 5]
[1 107]]
Sınıflandırma Raporu

		precision	recall	f1-score	support
kanser	1	0.98	0.92	0.95	63
iyi huylu	0	0.96	0.99	0.97	108
avg / total		0.97	0.96	0.96	171

2. Eğitim seti %75 test seti %25

Değerlendirme metrikleri
[[52 2]
[1 88]]
Sınıflandırma Raporu

		precision	recall	f1-score	support
kanser	1	0.98	0.96	0.97	54
iyi huylu	0	0.98	0.99	0.98	89
avg / total		0.98	0.98	0.98	143

3. Eğitim seti %80 test seti %20

Değerlendirme metrikleri
[[40 3]
[0 71]]
Sınıflandırma Raporu

		precision	recall	f1-score	support
kanser	1	1.00	0.93	0.96	43
iyi huylu	0	0.96	1.00	0.98	71
avg / total		0.97	0.97	0.97	114

4. Eğitim seti %85 test seti %15

Değerlendirme metrikleri
[[30 2]
[1 53]]
Sınıflandırma Raporu

		precision	recall	f1-score	support
kanser	1	0.97	0.94	0.95	32
iyi huylu	0	0.96	0.98	0.97	54
avg / total		0.97	0.97	0.96	86

Tablo 4.72: LSTM modelin eğitim sonuçları

	Test seti %30	Test seti %25	Test seti %20	Test seti %15
Doğruluk	%96,49	%97,90	%97,37	%96,51
Precision	%97	%98	%97	%97
TPR (Recall)	%96	%98	%97	%97
F1 Score	%96	%98	%97	%96
Hata	0.0327	0.0283	0.0297	0.0365
Parametre Sayısı	1,570,039	1,570,039	1,570,039	1,570,039
Süre	0:41:47	0:41:36	0:48:05	0:49:10

4.4.5.3 Kapılı Yinelenen Birim (GRU)

4.4.3. bölümünde belirtiğimiz gibi modeli tanımlarız.

```
model = Sequential()
model.add(GRU(596, input_shape=(None,30),return_sequences=True))
model.add(GRU(30, return_sequences=True))
model.add(Dense(1))
```

Modeli tanımladıktan sonra, onu Keras tarafından yürütülebilecek şekilde tamamlarız gerekmektedir. 4.4.3. bölümünde belirtiğimiz gibi modeli tamamlarız.

```
model.compile(loss='mean_squared_error', optimizer='adam', metrics=['accuracy'])
from datetime import datetime
start_time = datetime.now()
model.fit(X_train, y_train, epochs=600, batch_size=600,  verbose=2, validation_data=(X_test,
y_test))
end_time = datetime.now()
print('Duration: {}'.format(end_time - start_time))
```

Bu fonksiyonda modeli uygulamadan önce zaman hesaplamaya başlıyoruz, bu fonksiyonu kullanarak modeli eğitiyoruz ve eğitim süresini hesaplıyoruz.

Test seti = %30, Eğitim süresi: 0:36:34.526736

Test seti = %25, Eğitim süresi: 0:37:25.560430

Test seti = %20, Eğitim süresi: 0:38:22.805946

Test seti = %15, Eğitim süresi: 0:36:45.456185

Toplam eğitilen parametre sayısı aşağıda göstermektedir.

Layer (type)	Output Shape	Param #
gru_1 (GRU)	(None, None, 596)	1121076
gru_2 (GRU)	(None, None, 30)	56430
dense_1 (Dense)	(None, None, 1)	31

Total params: 1,177,537
Trainable params: 1,177,537
Non-trainable params: 0

4.4.3. bölümünde belirtiğimiz gibi Kullandığımız kod test setinin sonuçlarını tahmin ederek modelin doğruluğun gösterir. Sonuçlar aşağıda sunulmaktadır.

Test seti = %30, Doğruluk oranı: 0.9766 ⟹ %97,66
Test seti = %25, Doğruluk oranı: 0.9860 ⟹ %98,60
Test seti = %20, Doğruluk oranı: 0.9737 ⟹ %97,37
Test seti = %15, Doğruluk oranı: 0.9651 ⟹ %96,51

Kapılı yinelenen birim model kullanarak %98,60 kadar doğruluk oranına sahip bir model elde ettik. Modeli iyi değerlendirmemiz için diğer değerlendirme metrikleri bulmamız gerekmektedir. 4.4.1. ve 4.4.3. bölümünde belirtiğimiz gibi değerlendirme metrikleri ve sınıflandırma raporu aşağıda sunulmaktadır.

1. Eğitim seti %70 test seti %30

Değerlendirme metrikleri
[[61 2]
[3 105]]
Sınıflandırma Raporu

	precision	recall	f1-score	support
kanser 1	1.00	0.94	0.97	63
iyi huylu 0	0.96	1.00	0.98	108
avg / total	0.98	0.98	0.98	171

2. Eğitim seti %75 test seti %25

Değerlendirme metrikleri
[[52 2]
[0 89]]
Sınıflandırma Raporu

	precision	recall	f1-score	support
kanser 1	0.96	1.00	0.98	54
iyi huylu 0	0.99	1.00	0.99	89
avg / total	0.98	1.00	0.98	143

3. Eğitim seti %80 test seti %20

Değerlendirme metrikleri

[[40 3]
[0 71]]
Sınıflandırma Raporu

		precision	recall	f1-score	support
kanser	1	1.00	0.93	0.96	43
iyi huylu	0	0.96	1.00	0.98	71
avg / total		0.97	0.97	0.97	114

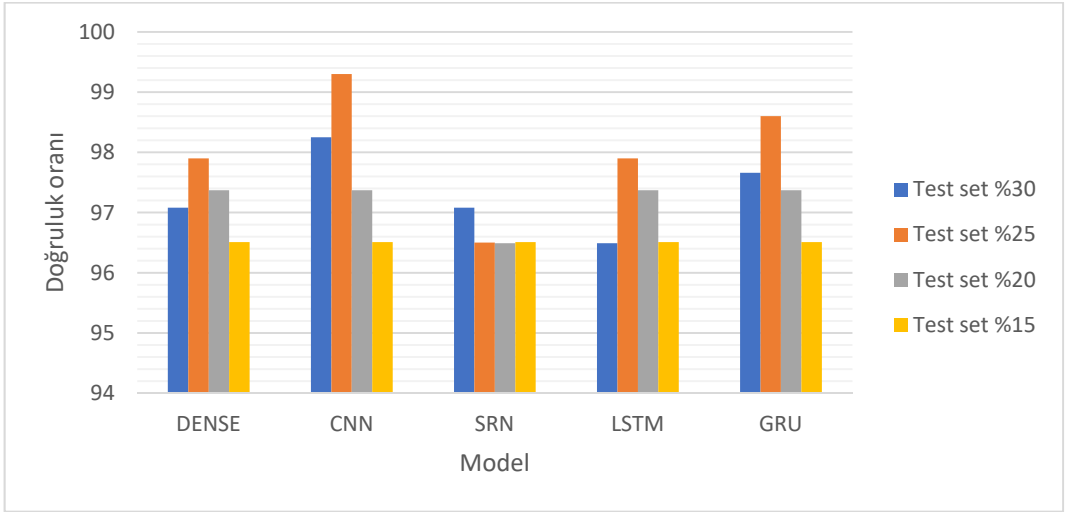
4. Eğitim seti %85 test seti %15

Değerlendirme metrikleri
[[30 2]
[1 53]]
Sınıflandırma Raporu

		precision	recall	f1-score	support
kanser	1	0.97	0.94	0.95	32
iyi huylu	0	0.96	0.98	0.97	54
avg / total		0.97	0.97	0.96	86

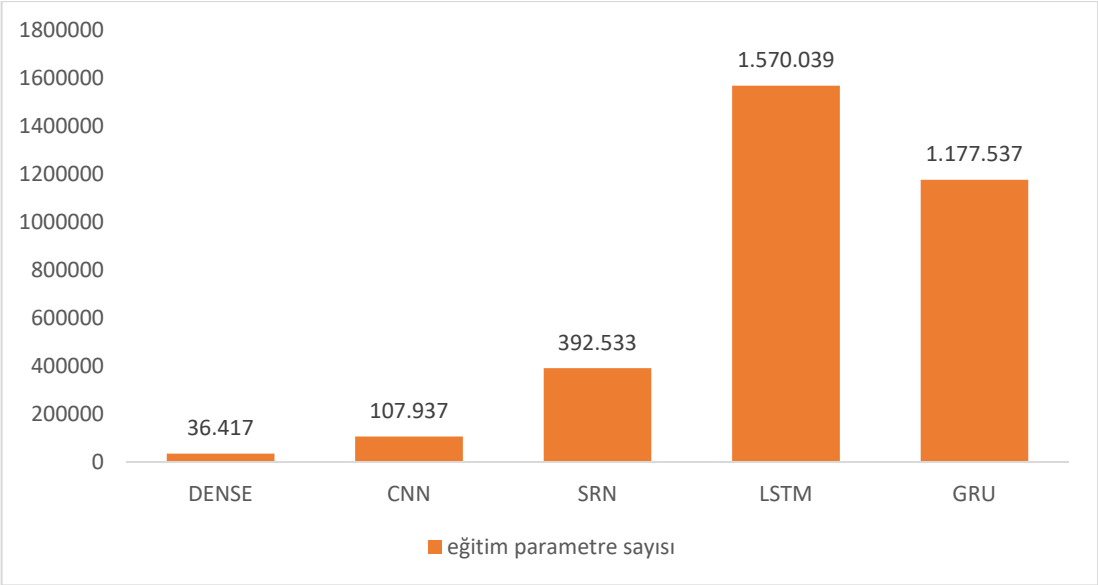
Tablo 4.8: GRU modelin eğitim sonuçları

	Test seti %30	Test seti %25	Test seti %20	Test seti %15
Doğruluk	%97,66	%98,60	%97,37	%96,51
Precision	%98	%98	%97	%97
TPR (Recall)	%98	%100	%97	%97
F1 Score	%98	%98	%97	%96
Hata	0.0365	0.0393	0.0307	0.0325
Parametre Sayısı	1,177,537	1,177,537	1,177,537	1,177,537
Süre	0:36:34	0:37:25	0:38:22	0:36:45



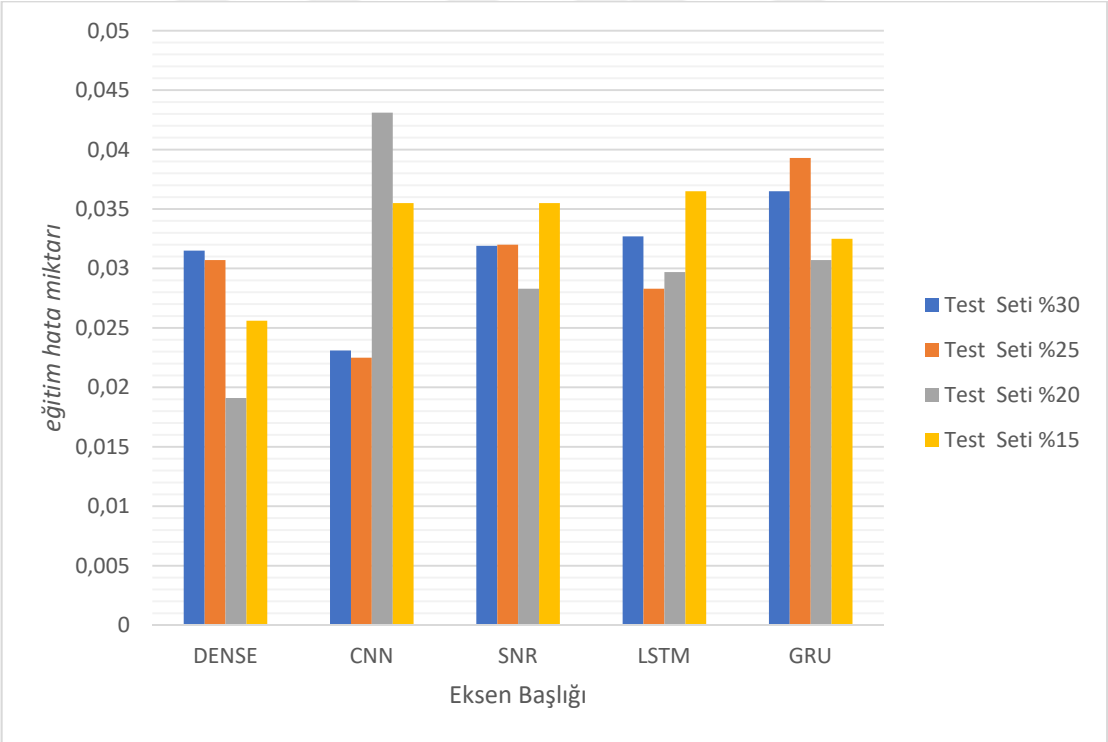
Şekil 4.9: Model doğruluk oranı

Şekil 4.9'da gösterildiği gibi göğüs kanser teşhisinde kullandığımız en iyi yöntem Konvolüsyon sinir ağlarıdır.



Şekil 4.10: Model eğitim parametre sayısı

Şekil 4.10'de gösterildiği gibi LSTM, GRU ve SRN yöntemlerinin eğitim parametresi çok yüksektir.



Şekil 4.11: Model eğitim hata miktarı

4.4.6 Diğer makine öğrenme yöntemleri

4.4.6.1 Naive Bayes

Naive Bayes sınıflandırma algoritması, adını Matematikçi Thomas Bayes'den alan bir sınıflandırma algoritmasıdır. Naive Bayes sınıflandırması olasılık ilkelerine göre tanımlanmış bir dizi hesaplama ile, sisteme sunulan verilerin sınıfını yani kategorisini tespit etmeyi amaçlar. 4.4.3 belirtiğimiz gibi veri setini kullanmak için hazırlıyoruz. Python sklearn kütüphanesini kullanarak ve test setinin oranı %25 kullanarak bu modeli test etmekteyiz.

```
from sklearn.naive_bayes import GaussianNB
model = GaussianNB()
model.fit(X_train, y_train)
```

Yukardaki Python kodında ilk adım Naive Bayes modeli sklearn kütüphanesinden kullanmak amacı ile çağırıyoruz, İkinci adım modeli tanımlıyoruz ve sonda modeli uyguluyoruz. Bulunan değerlendirme metrikleri aşağıda sunulmaktadır.

Eğitim seti %75 test seti %25

```
Değerlendirme metrikleri
 [[51  3]
 [ 3 86]]
Sınıflandırma Raporu
             precision    recall  f1-score   support
kanser   1       0.94      0.94      0.94       54
iyi huylu 0      0.97      0.97      0.97       89

avg / total      0.96      0.96      0.96      143
```

Tablo 4.9: Naive Bayes modelin eğitim sonuçları

	Test seti %25
Doğruluk	%95,80
Precision	%96
TPR (Recall)	%96
F1 Score	%96
Süre	0:00:00.075051

4.4.6.2 K-Neighbors

K en yakın komşuluk algoritması sorgu vektörünün en yakın k komşuluktaki vektör ile sınıflandırılmasının bir sonucu olan denetlemeli öğrenme algoritmasıdır. Bu algoritma ile yeni bir vektörü sınıflandırabilmek için doküman vektörü ve eğitim dokümanları vektörleri kullanılır. Bir sorgu örneği verilir, bu sorgu noktasına en yakın k tane eğitim noktası bulunur. Sınıflandırma ise bu k tane nesnenin en fazla olanı ile yapılır. K en yakın komşuluk uygulaması yeni sorgu örneğinin sınıflandırmak için kullanılan bir komşuluk sınıflandırma algoritmasıdır. 4.4.3 belirtiğimiz gibi veri setini kullanmak için hazırlıyoruz. Python sklearn kütüphanesini kullanarak ve test setinin oranı %25 kullanarak bu modeli test etmekteyiz.

```
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier(n_neighbors=3)
model.fit(X_train, y_train)
```

Yukardaki Python kodında ilk adım K-Neighbors modeli sklearn kütüphanesinden kullanmak amacı ile çağırıyoruz, İkinci adım modeli tanımlıyoruz ve sonda modeli uyguluyoruz. Bulunan değerlendirme metrikleri aşağıda sunulmaktadır.

Eğitim seti %75 test seti %25

Değerlendirme metrikleri
[[49 5]
[5 84]]
Sınıflandırma Raporu

	precision	recall	f1-score	support
kanser 1	0.91	0.91	0.91	54
iyi huylu 0	0.94	0.94	0.94	89
avg / total	0.93	0.93	0.93	143

Tablo 4.10: K-Neighbors modelin eğitim sonuçları

	Test seti %25
Doğruluk	%93
Precision	%93
TPR (Recall)	%93
F1 Score	%93
Süre	0:00:00.003503

4.4.6.3 Lojistik Regresyon

Lojistik regresyon, istatistik alanından makine öğrenmesinin ödünç aldığı bir başka tekniktir. Lojistik regresyon, bir sonucu belirleyen bir veya daha fazla bağımsız değişken bulunan bir veri kümesini analiz etmek için istatistiksel bir yöntemdir. Sonuç, iki yönlü bir değişkenle ölçülür (ki sadece iki olası sonuç vardır). 4.4.3 belirtiğimiz gibi veri setini kullanmak için hazırlıyoruz. Python sklearn kütüphanesini kullanarak ve test setinin oranı %25 kullanarak bu modeli test etmekteyiz.

```
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
model.fit(X_train, y_train)
```

Yukardaki Python kodında ilk adım Lojistik regresyon modeli sklearn kütüphanesinden kullanmak amacı ile çağırıyoruz, İkinci adım modeli tanımlıyoruz ve sonda modeli uyguluyoruz. Bulunan değerlendirme metrikleri aşağıda sunulmaktadır.

Eğitim seti %75 test seti %25

```
Değerlendirme metrikleri
 [[40  4]
 [ 2 87]]
Sınıflandırma Raporu
              precision    recall  f1-score   support
kanser   1       0.96      0.93      0.94        54
iyi huylu 0      0.96      0.98      0.97        89

avg / total      0.96      0.96      0.96       143
```

Tablo 4.11: Logistic Regression modelin eğitim sonuçları

	Test seti %25
Doğruluk	%95,80
Precision	%96
TPR (Recall)	%96
F1 Score	%96
Süre	0:00:00.021641

4.4.6.4 Karar Ağacı (Decision Tree)

Karar ağacı öğrenme yöntemi, makine öğrenmesi konularından birisidir. Literatürde karar ağacı öğrenmesinin alt yöntemleri olarak kabul edilebilecek sınıflandırma ağacı veya ilkelleştirme ağacı gibi uygulamaları vardır. Karar ağacı öğrenmesinde, bir ağaç yapısı oluşturularak ağacın yaprakları seviyesinde sınıf etiketleri ve bu yapraklara giden ve başlangıçtan çıkan kollar ile de özellikler üzerindeki işlemeler ifade edilmektedir. 4.4.3 belirtiğimiz gibi veri setini kullanmak için hazırlıyoruz. Python sklearn kütüphanesini kullanarak ve test setinin oranı %25 kullanarak bu modeli test etmekteyiz.

```
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
model.fit(X_train, y_train)
```

Yukardaki Python kodında ilk adım Lojistik regresyon modeli sklearn kütüphanesinden kullanmak amacı ile çağırıyoruz, İkinci adım modeli tanımlıyoruz ve sonda modeli uyguluyoruz. Bulunan değerlendirme metrikleri aşağıda sunulmaktadır.

Eğitim seti %75 test seti %25

```
Değerlendirme metrikleri
 [[50  4]
 [ 4 85]]
Sınıflandırma Raporu
              precision    recall  f1-score   support

kanser   1       0.93      0.94      0.94        54
iyi huylu 0      0.97      0.96      0.96        89

avg / total      0.95      0.95      0.95       143
```

Tablo 4.12: Decision Tree modelin eğitim sonuçları

	Test seti %25
Doğruluk	%95,10
Precision	%95
TPR (Recall)	%95
F1 Score	%95
Süre	0:00:00.007630

4.5 Tartışma

Derin öğrenme yöntemleri, sinir ağlarının bir alt alanıdır, bu nedenle sinir ağına etki eden tüm faktörler derin öğrenme yöntemlerine de etki edecektir. 3.1 bölümünde belirtiğimiz gibi aktivasyon fonksiyonları, hata fonksiyonları ve optimizasyon yöntemleri yapay sinir ağlarının en önemli faktörleridir. Bu faktörlerin değişik türlerini kullanarak Konvolüsyon sinir ağı üzerine uygulamaktayız, sonuçlar aşağıda sunulmaktadır.

Tablo 4.13: Değişik aktivasyon fonksiyon türlerini kullanarak Konvolüsyon sinir ağı doğruluk oranı

Aktivasyon fonksiyonu	Konvolüsyon sinir ağı test seti %25
Sigmoid	%99,30
Tanh	%94,41
Relu	%62,24

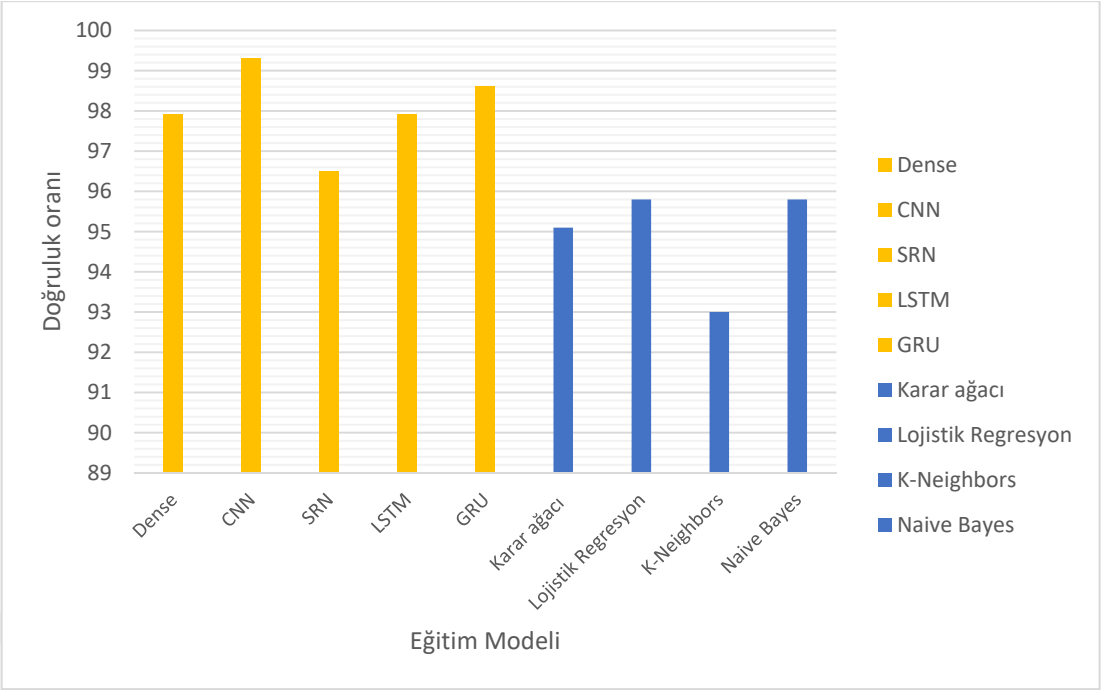
Tablo 4.14: Değişik hata fonksiyon türlerini kullanarak Konvolüsyon sinir ağı doğruluk oranı

Hata fonksiyonu	Konvolüsyon sinir ağı test seti %25
mean_squared_error	%99,30
Hinge	%00,00
Logcosh	%97,20

Tablo 4.15: Değişik optimizasyon fonksiyon türlerini kullanarak Konvolüsyon sinir ağı doğruluk oranı

Optimizasyon fonksiyonu	Konvolüsyon sinir ağı test seti %25
Adam	%99,30
Sgd	%90,21
RMSprop	%93,70

Tablo 4.12-4-14'de gösterildiği gibi Sigmoid aktivasyon fonksiyonu, İkinci Dereceden (Kuadratik) Hata Fonksiyonu ve ADAM optimizasyon yöntemi en iyi sonuçlar göstermiştir.



Şekil 4.12: %25 Test seti kullanarak değişik makine öğrenme teknikleri ve derin öğrenme yöntemleri doğruluk oranı.

Derin öğrenme yöntemleri diğer makine öğrenme yöntemlerine göre üstün çalışmaktadır. Yapay sinir ağlarını etkiliyken tüm faktörler aynı zamanda derin öğrenme yöntemlerini etkiler.

BEŞİNCİ BÖLÜM

SONUÇ VE DEĞERLENDİRME

Bu araştırmanın temel amacı, sağlık hizmetlerinin kalitesini arttırma çabalarına katılmaktır. Bu tez, pek çok ülkenin sağlık hizmetleri alanında karşılaştığı zorlukları sunmaktadır. Her sağlık sisteminin ilk aşaması, hastalığın tansıdır.

Bu çalışmada, derin öğrenme modeli eğitmek için yeterli veri varsa derin öğrenme yaklaşımlarının klinik karar verme sürecine başarıyla uygulanabileceği belirtilmiştir. Göğüs kanseri tanısında derin öğrenme teknikleri verimli bir şekille kullanılabilir. Derin öğrenme tekniklerinin kullanılması teşhis yöntemin doğruluğunu arttırır ve insan uzmanının ihtiyacını azaltır. Burada sunulan sonuçların ilginç olduğuna inanıyoruz ve derin öğrenme tekniğin diğer hastalıkların teşhisi için nasıl kullanılabileceği üzerine daha ileri araştırmalara yol açacaktır. Derin ağlar, ağa eğitilen verilere bağlı olarak çalışır. Derin ağlarda daha fazla veri eğitilirse, ağın daha doğru kararlar vermesine yardımcı olur. Tanı sonuçlarından, kadınların veya nadiren erkeklerin göğüste bulunan tümörler kanser olup olmadığını gösterir.

Deneylerde herkese açık gerçek hayatta biyomedikal veri kümesi kullandık. Kullandığım bu veri seti, 1992 yılında Wisconsin Üniversitesi tarafından bir göğüs kütlesinin ince bir iğne aspirasyonunun (FNA) sayısal bir görüntüsünden alınan özellikleri kullanarak bağışta bulunan çok ünlü bir veri setidir. Derin öğrenme tekniklerinin performansını değerlendirmek amacı ile göğüs tümörlerini teşhis etmek için başarılı bir şekilde uygulanan derin öğrenme algoritmaların (DENSE, CNN, SRN, LSTM ve GRU) sonuçları karşılaştırılarak analiz edilmiştir.

Tam bağlantılı yapay sinir ağları (DENSE) göğüs kanser teşhisinde, değişik oranlarda test seti kullanarak %96,51 ile %97,90 arası doğruluk oranı göstermektedir. Konvolüsyon yapay sinir ağları (CNN) göğüs kanser teşhisinde, değişik oranlarda test seti kullanarak %96,51 ile %99,30 arası doğruluk oranı göstermektedir.

Basit tekrarlayan yapay sinir ağları (SRN) göğüs kanser teşhisinde, değişik oranlarda test seti kullanarak %96,49 ile %97,08 arası doğruluk oranı göstermektedir. Uzun kısa dönem yapay sinir ağları (LSTM) göğüs kanser teşhisinde, değişik oranlarda test seti kullanarak %96,49 ile %97,90 arası doğruluk oranı göstermektedir. Kapılı yenilenen birim yapay sinir ağları (GRU) göğüs kanser teşhisinde, değişik oranlarda test seti kullanarak %96,51 ile %98,60 arası doğruluk oranı göstermektedir. Kullandığımız diğer makine yöntemleri, Naive Bayes, K-Neighbors, Lojistik Regresyon ve Karar Ağacı doğruluk oranları sıra ile %95,80, %93, 95,80 ve 95,10. Sonuçları karşılatarak derin öğrenme yöntemleri diğer makine öğrenme yöntemlerine göre saha üstün çalışmaktadır ancak derin öğrenme yöntemleri çok maliyetlidir ve öğrenme süresi diğer makine öğrenme yöntemlerine göre katlarca fazladır. Konvolüsyon yapay sinir ağları genellikle resim ve video sınıflandırmasında kullanılmaktadır, bu çalışmada tıbbi veriler üzerine kullanılmaktadır ve %99,30 doğruluk oranı göstermektedir.

Aktivasyon fonksiyonları, hata fonksiyonları ve optimizasyon yöntemleri yapay sinir ağlarının en önemli faktörleridir. Derin öğrenme yöntemleri, yapay sinir ağlarının bir türüdür. Aktivasyon fonksiyonları, hata fonksiyonları ve optimizasyon yöntemleri değişik türlerini seçerek sonuçlar ciddi bir şekil ile etkilenmektedir. Derin öğrenme mimarisini seçmek için belli bir kural olmadığından dolayı doğru mimari seçmek en zor noktadır. Bu çalışılmada kullandığımız mimariler deneyim üzerine istinaden seçilmektedir. Gizli katman sayısını aratılarak eğitim maliyeti fazla olduğuna rağmen daha iyi sonuçlar göstermek zorunda değildir (bazen daha iyi sonuçlar ve bazen daha kötü sonuçlar göstermektedir). Derin öğrenme mimarisi verinin tipine göre seçilir. Daha fazla veri bulunması halinde derin öğrenme yöntemleri daha iyi sonuçlar göstermektedir.

Mevcut araştırmanın değerlendirmesinde esas olarak sınıflandırma doğruluk oranı kullanılmaktadır. Bununla birlikte, gelecekteki çalışmalar sınıflandırma hızı ve hesaplama maliyeti gibi diğer kriterlere odaklanacaktır. Derin öğrenme yöntemleri, sadece diğer kanser türlerini teşhis etmekle sınırlanamaz, aynı zamanda farklı alanlarında araştırılmaya değerdir.

KAYNAKÇA

[1] World Health Organization, «World Health Organization Ranking; The World's Health Systems,» 2016. [Çevrimiçi]. Available: http://thepatientfactor.com/canadian-health-care-information/world-health-organizations-ranking-of-the-worlds-health-systems/.

[2] LeDuc Media, 2017. [Çevrimiçi]. Available: http://www.worldlifeexpectancy.com/country-health-profile/turkey.

[3] W. N. S. W. H. W. O.L. Mangasarian, «Breast cancer diagnosis and prognosis via linear programming.,» *Mathematical Programming Technical report,* pp. 1-9, 1994.

[4] V. Ozmen, «Breast Cancer in the World and Turkey,» *meme sağlık dergisi,* 2008.

[5] N. A. R. A. A. A. M. S. U. R. Muhammad Asim, «Merck Manual of Diagnosis and Therapy (2003) Breast Disorders: Breast Cancer. Retrieved 5 February 2008.,» *Open Access Library Journal,* 2003.

[6] S. W. Fletcher, W. Black, R. Harris, B. K. Rimer ve S. Shapiro, «Report of the international work shop on screening for breast cancer,» *Journal of the National Cancer Institute, 85,* pp. 1644-1656, 1993.

[7] R. W. M. Giard ve J. Hermans, «The value of aspiration cytologic examination of the breast,» *A statistical review of the medical literature.,* pp. 2104-2110, 1992.

[8] E. Übeyli, «Implementing automated diagnostic systems,» *Expert Systems with Applications,* p. 1054–1062, 2007.

[9] M. S. Bin Mohd Azmi ve Z. Che Cob, «Breast Cancer prediction based on Backpropagation Algorithm,» %1 içinde *IEEE Student Conference on Research and Development (SCOReD)*, Putrajaya, 2010.

[10] A. Khosravi, J. Addeh ve J. Ganjipour, «Breast Cancer Detection Using BA-BP Based Neural Networks and Efficient Features,» %1 içinde *7th Iranian Conference on Machine Vision and Image Processing*, 2011.

[11] X. Yang, H. Peng ve M. Shi, «SVM with multiple kernels based on manifold learning for Breast Cancer diagnosis,» %1 içinde *IEEE International Conference on Information and Automation (ICIA)*, 2013.

[12] S. Phetlasy, S. Ohzahata, C. Wu ve T. Kato, «Sequential Combination of Two Classifier Algorithms for Binary Classification to Improve the Accuracy,» %1 içinde *Third International Symposium on Computing and Networking (CANDAR)*, Sapporo, 2015.

[13] C. Deng ve M. Perkowski, «A Novel Weighted Hierarchical Adaptive Voting Ensemble Machine Learning Method for Breast Cancer Detection,» %1 içinde *IEEE International Symposium on Multiple-Valued Logic*, Ontario, 2015.

[14] Z. Yin, Z. F. Fei, C. Yang ve A. Chen, «A novel SVM-RFE based biomedical data processing approach: Basic and beyond,» %1 içinde *42nd Annual Conference of the IEEE Industrial Electronics Society*, IECON, 2016.

[15] D. Bazazeh ve R. Shubair, «Comparative Study of Machine Learning Algorithms for Breast Cancer Detection and Diagnosis,» %1 içinde *5th International Conference on Electronic Devices, Systems and Applications (ICEDSA)*, Ras Al Khaimah, 2016.

[16] I. Singh, K. Sanwal ve S. Praveen, «Breast cancer detection using two-fold genetic evolution of neural network ensembles,» %1 içinde *2016 International Conference on Data Science and Engineering (ICDSE)*, Cochin, 2016.

[17] S. Jhajharia, H. Varshney, S. Verma ve R. Kumar, «A neural network based breast cancer prognosis model with PCA processed features,» %1 içinde *2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, Jaipur, 2016.

[18] N. Dhungel, G. Carneiro ve A. P. Bradley, «Automated Mass Detection in Mammograms Using Cascaded Deep Learning and Random Forests,» %1 içinde *International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, 2015.

[19] X. Chen, Y. Xu, D. W. K. Wong, T. Y. Wong ve J. Liu, «Glaucoma detection based on deep convolutional neural network,» %1 içinde *37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 2015.

[20] M. Khademi ve N. Nedialkov, «Probabilistic Graphical Models and Deep Belief Networks for Prognosis of Breast Cancer,» %1 içinde *IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, 2015.

[21] S. Suzuki, X. Zhang, N. Homma, K. Ichiji, N. Sugita, Y. Kawasumi ve T. Ishibashi, «Mass detection using deep convolutional neural network for mammographic computer-aided diagnosis,» %1 içinde *55th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*, 2016.

[22] S. Sarraf ve G. Tofighi, «Deep learning-based pipeline to recognize Alzheimer's disease using fMRI data,» %1 içinde *Future Technologies Conference (FTC)*, 2016.

[23] A. H. Al-Fatlawi, M. H. Jabardi ve S. H. Ling, «Efficient diagnosis system for Parkinson's disease using deep belief network,» %1 içinde *IEEE Congress on Evolutionary Computation (CEC)*, 2016.

[24] A. M. Abdel-Zaher ve A. M. Eldeib, «Breast cancer classification using deep belief networks,» *Expert Systems with Applications,* p. 139–144, 2016.

[25] A. F. Syafiandini, I. Wasito, S. Yazid, A. Fitriawan ve M. Amien, «Cancer subtype identification using deep learning approach,» %1 içinde *International Conference on Computer, Control, Informatics and its Applications (IC3INA)*, 2016.

[26] R. Shimizu, S. Yanagawa, Y. Monde, H. Yamagishi, M. Hamada, T. Shimizu ve T. Kuroda, «Deep learning application trial to lung cancer diagnosis for medical sensor systems,» %1 içinde *International SoC Design Conference (ISOCC)*, 2016.

[27] E. Nasr-Esfahani, S. Samavi, N. Karimi, S. M. R. Soroushmehr, M. H. Jafari, K. Ward ve K. Najarian, «Melanoma detection by analysis of clinical images using convolutional neural network,» %1 içinde *38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 2016.

[28] G. L. F. d. Silva, O. P. d. S. Neto, A. C. Silva, A. C. d. Paiva ve M. Gattass, «Lung nodules diagnosis based on evolutionary convolutional neural network,» *Multimedia Tools and Applications,* pp. 1-17, 2017.

[29] I. Guyon, «A scaling law for the validation-set training-set size ratio,» AT&T Bell Laboratories, California, 1997.

[30] E. ÖZTEMEL, YAPAY SİNİR AĞLARI, PAPATYA YAYINCILIK EĞİTİM, 2006.

[31] S. N. Sivanandam ve S. N. Deepa, Introduction to Neural Networks Using Matlab 6.0, Tata McGraw-Hill Education, 2006.

[32] X. Glorot ve Y. Bengio, «Understanding the difficulty of training deep feedforward neural networks,» *Aistats,* pp. 249-256, 2015.

[33] K. He, X. Zhang, S. Ren ve J. Sun, «Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification,» %1 içinde *The IEEE International Conference on Computer Vision (ICCV)*, 2015.

[34] A. M. SAXE, MCCLELLAND ve G. S. J. L., «Exact solutions to the nonlinear dynamics of learning in deep linear neural networks,» 2015. [Çevrimiçi]. Available: https://arxiv.org/abs/1312.6120.

[35] L. BOTTOU ve O. BOUSQUET, «The Tradeoffs of Large Scale Learning,» %1 içinde *Advances in Neural Information Processing Systems, v. 20*, 2008.

[36] I. Sutskever, J. Martens, G. Dahl ve G. Hinton, «On the importance of initialization and momentum in deep learning,» %1 içinde *30th International Conference on Machine Learning*, 2013.

[37] B. T. POLYAK, «Some methods of speeding up the convergence of iteration methods,» *USSR Computational Mathematics and Mathematical Physics,* p. 1–17, 1964.

[38] J. Ba ve D. Kingma, «Adam: A Method for Stochastic Optimization,» arXiv preprint arXiv:1412.6980, 2014.

[39] I. MILLINGTON ve J. FUNGE, ARTIFICIAL INTELLIGENCE FOR GAMES, AMSTERDAM: Morgan Kaufmann Publishers is an imprint of Elsevier, 2009.

[40] A. Gibson ve J. Patterson, Deep Learning, O'Reilly Media, Inc., 2017.

[41] G. E. Hinton, S. Osindero ve Y.-W. Teh, « A Fast Learning Algorithm for Deep Belief Nets,» *Neural Computation,* pp. 1527 - 1554, 2006.

[42] B. Schölkopf, J. Platt ve T. Hofmann, «Greedy Layer-Wise Training of Deep Networks,» %1 içinde *Advances in Neural Information Processing Systems 19:Proceedings of the 2006 Conference*, 2007.

[43] Y. LECUN, L. BOTTOU, Y. BENGIO ve e. al, «Gradient-based learning applied to document recognition,» %1 içinde *Proceedings of the IEEE volume 86*, 1998.

[44] D. S. C. H. H. P. J. Z. Y. Z. a. J. L. JiWan, «Deep learning for content-based image retrieval: A comprehensive study,» %1 içinde *ACM International Conference on Multimedia*, 2014.

[45] N. Lewis, DEEP LEARNING MADE EASY WITH R: A Gentle Introduction for Data Science, Createspace Independent Publishing Platform, 2016.

[46] J. F. Wiley, R Deep Learning Essentials, Packt Publishing, 2016.

[47] J. Heaton, Deep Learning and Neural Networks, Heaton Research, 2015.

[48] H. T. Siegelmann ve E. D. Sontag, «Turing Computability With Neural,» *Applied Mathematics Letters,* pp. 77-80, 1991.

[49] M. Hermans ve B. Schrauwen, «Training and Analysing Deep Recurrent Neural Networks,» %1 içinde *Advances in Neural Information Processing Systems 26*, Curran Associates, 2013, p. 190–198.

[50] D. Dev, Deep Learning with Hadoop, Packt Publishing, 2017.

[51] S. Hochreiter ve J. Schmidhuber, «Long Short-Term Memory,» *Neural Computation,* pp. 1735 - 1780, 1997.

[52] P. Doetsch, M. Kozielski ve H. Ney, «Fast and Robust Training of Recurrent Neural Networks for Offline Handwriting Recognition,» %1 içinde *14th International Conference on Frontiers in Handwriting Recognition*, Heraklion, Greece, 2014.

[53] A. Graves, A.-r. Mohamed ve G. Hinton, «Speech recognition with deep recurrent neural networks,» %1 içinde *2013 IEEE International Conference on*

Acoustics, Speech and Signal Processing, Vancouver, BC, Canada, 2013.

[54] J. Chung, C. Gulcehre, K. Cho ve Y. Bengio, «Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling,» *arXiv e-prints,* 2014.

[55] R. Jozefowicz, W. Zaremba ve I. Sutskever, «An empirical exploration of recurrent network architectures,» *Journal of Machine Learning Research,* 2015.

[56] Y. LeCun ve Y. Bengio, «Convolutional Networks for Images, Speech, and Time,» %1 içinde *Neural Information Processing: Research and Development*, springer, 1995, pp. 276-279.

[57] I. Arel, D. C. Rose ve T. P. Karnowski, «Deep Machine Learning - A New Frontier in Artificial Intelligence Research [Research Frontier],» *IEEE Computational Intelligence Magazine,* pp. 13-18, 2010.

[58] D. H. HUBEL ve T. N. WIESEL, «Receptive fields and functional architecture of monkey striate cortex,» *Journal of Physiology,* pp. 215-243, 1968.

[59] C. Clark ve A. Storkey, «Teaching Deep Convolutional Neural Networks to Play Go,» *arvix.org,* 2014.

[60] R. Collobert ve J. Weston, «A unified architecture for natural language processing: deep neural networks with multitask learning,» %1 içinde *he 25th International Conference on Machine Learning*, NEW YORK, 2008.

[61] D. W. H. Wolberg, 1995. [Çevrimiçi]. Available: https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic).

[62] W. H. W.,. O. L. M. Nick Street, «Nuclear Feature Extraction For Breast Tumor Diagnosis,» *IS&T/ SPIE Int. Symp. on Electron. Imaging: Sci. and Technology,* pp. 861- 870, 1993.

[63] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley ve Y. Bengio, «Theano: A CPU and GPU math compiler in Python,» %1 içinde *9th Python in Science Conf*, 2010.

[64] Continuum, «Anaconda,» [Çevrimiçi]. Available: https://www.continuum.io/downloads.

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı	:	Ali Mahmood Ogur Anwer
Uyruğu	:	Irak
Doğum Yeri ve Tarihi	:	Kerkük 01.01.1985
Medeni Hali	:	Evli
Adres	:	Sancak mh. 538 sok. 1/7 Çankaya - Ankara
E-Posta Adresi	:	aliogor.ar@gmail.com
İletişim (Telefon)	:	00905319845396

EĞİTİM

Lise	:	Alresala lisesi / Irak - Musul
Lisans	:	Bilgisayar Mühendisliği / Irak - Musul
Yüksek Lisans	:	Türk Hava Kurumu Üniversitesi

MESLEKİ DENEYİM

2011-	:	Araştırma görevlisi /Irak Kuzey Üniversitesi

YABANCI DİL

Türkçe, İngilizce