



**PARKİNSON HASTALIĞININ TEŞHİSİNDE DERİN
ÖĞRENME YÖNTEMİ İLE SPECT GÖRÜNTÜ
ANALİZİ**

Okan ALKAN

**Yüksek Lisans Tezi
Matematik Ana Bilim
Bilgisayar Bilimleri Bilim Dalı
Doç. Dr. Aytürk KELEŞ
AĞRI-2019
(Her hakkı saklıdır.)**

T.C.
AĞRI İBRAHİM ÇEÇEN ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
MATEMATİK ANABİLİM DALI

Okan ALKAN

PARKİNSON HASTALIĞININ TEŞHİSİNDE DERİN ÖĞRENME YÖNTEMİ İLE
SPECT GÖRÜNTÜ ANALİZİ
YÜKSEK LİSANS TEZİ

TEZ YÖNETİCİSİ
Doç. Dr. Aytürk KELEŞ

AĞRI-2019

TEZ ONAY FORMU

PARKİNSON HASTALIĞININ TEŞHİSİNDE DERİN ÖĞRENME YÖNTEMİ İLE SPECT
GÖRÜNTÜ ANALİZİ

Doç. Dr. Aytürk KELEŞ danışmanlığında, Okan ALKAN tarafından hazırlanan bu çalışma, 19/07/2019 tarihinde aşağıdaki jüri tarafından Matematik Anabilim Dalı Bilgisayar Bilimleri Bilim Dalı'nda Yüksek Lisans tezi olarak **öybirliği / oy çokluğu (.../...)** ile kabul edilmiştir.

Başkan : Prof. Dr. Uğur YAVUZ

Üye : Doç. Dr. Aytürk KELEŞ

Üye : Dr. Öğr. Üyesi Ali KELEŞ

İmza :

İmza :

İmza :

Yukarıdaki sonuç;

Enstitü Yönetim Kurulu .../.../201.. tarih ve / nolu kararı ile onaylanmıştır.

Prof. Dr. İbrahim HAN
Enstitü Müdürü

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

ÖZET
YÜKSEK LİSANS TEZİ
PARKİNSON HASTALIĞININ TEŞHİSİNDE DERİN ÖĞRENME
YÖNTEMİ İLE SPECT GÖRÜNTÜ ANALİZİ

Tez Danışmanı: Doç. Dr. Aytürk KELEŞ

Sinir hücreleri tahribatı sonucunda gelişen Parkinson hastalığı, Alzheimer hastalığının ardından en sık teşhisi konulan hastalıktır. Dünya üzerinde 55-60 yaş üzerindeki bireylerde %1 oranında görülmektedir. Bu oran düşük görünmesine rağmen, yaşam sürelerinin uzaması göz önüne alındığında Parkinson hastalığına yakalanan bireylerde artışa neden olmaktadır. Parkinson hastalığının erken teşhisinde bugün geleneksel yöntemler kullanılmaktadır. Ancak araştırmacılar Yapay Zekâ teknolojilerini kullanarak bu hastalığı daha erken ve doğru teşhis edilebilmesi için çalışmaktadır.

Bu çalışmada SPECT görüntüleri kullanılarak derin öğrenme teknolojisi ile Parkinson hastalığının daha erken ve daha doğru teşhis edilebilmesine katkı sağlanmaya çalışılmıştır. SPECT görüntüleri analiz etmek için evrişimsel sinir ağı yapısı kullanılarak bir model oluşturulmuş ve Keras kütüphanesi içerisinde yer alan optimizasyon seçenekleri ile test edilmiştir. Her optimizasyon değeri için farklı aktivasyon fonksiyonları ve eğitim sayıları denenmiş ve en başarılı doğruluk oranına ulaşmaya çalışılmıştır. Hastalık teşhisi için başarı kriteri olarak doğruluk oranının yanında duyarlılık ve özgüllük değerleri de hesaplanarak tablo olarak sunulmuştur. Üç farklı sınıfta yer alan SPECT görüntü verileri üzerinde uygulanan derin öğrenme işlemi sonucunda en başarılı sınıflandırma Adamax optimizasyon yöntemi ile doğruluk %94.15, duyarlılık %91.03 ve özgüllük %95.71 değerleri ile elde edilmiştir.

Bu tez çalışması genel olarak 5 bölümden oluşmaktadır. İlk bölümünde Parkinson hastalığına ve derin öğrenme yöntemlerine yer verilerek konuyla ilgili alanyazındaki mevcut çalışmalar sunulmuştur. İkinci bölümde yapay zekâ ve onun alt alanında yeni gelişen teknoloji derin öğrenme yer almaktadır. Bu bölümde derin öğrenme metotları ve kullanım alanları da örneklendirilerek açıklanmıştır. Üçüncü bölümde kullanılan veri seti hakkında ayrıntılı bilgi verilmiş ve uygulamada kullanılan yapılar tanıtılmıştır. Dördüncü bölümde kullanılan optimizasyon yöntemlerine göre elde edilen doğruluk, duyarlılık ve özgüllük değerleri görsel olarak değerlendirilmiştir. Son bölümde ise yapılan teşhis uygulamasından elde edilen sonuçlar alan yazında benzer veya yakın çalışmalar ile karşılaştırılarak bir değerlendirme yapılmış ve bu çalışmanın ışığında ileriye yönelik bazı önerilerde bulunulmuştur.

2019, 115 sayfa

Anahtar sözcükler: Derin öğrenme, yapay zekâ, parkinson hastalığı, hastalık teşhisi, tensorflow, keras, python, spect, dicom.

ABSTRACT
MASTER THESIS
SPECT IMAGE ANALYSIS WITH DEEP LEARNING METHOD FOR
DIAGNOSIS OF PARKINSON'S DISEASE

Advisor: Associate Professor Aytürk KELEŞ

Parkinson's disease which develops as a result of nerve cells devastation, is the most commonly diagnosed disease after Alzheimer's disease. It is seen in 1 percent of individuals over the age of 55-60 years. Although this ratio seems low, it causes an increase in individuals with Parkinson's disease considering the prolongation of life expectancy. Today, traditional methods are used in the early diagnosis of Parkinson's disease. However, using artificial intelligence technologies, researchers are studying to be detected this disease earlier and more accurately.

In this study, it has been tried to contribute to the early and more accurate diagnosis of Parkinson's disease by using SPECT images with deep learning technology. In order to analyze SPECT images, a model was created using convolutional neural network structure and tested with optimization options which is within the Keras library. For each optimization value, different activation functions and epoch have been tested and tried to reach the most successful accuracy rate. Sensitivity and specificity values as well as accuracy rate as the criterion of success for disease diagnosis are calculated and presented as a table. As a result of the deep learning process applied on SPECT image data, which is in the three different classes, the most successful classification method is obtained with Adamax optimization method 94.15% accuracy, 91.03% sensitivity and 95.71% specificity.

This thesis consists of 5 chapters in general. In the first part, Parkinson's disease and deep learning methods are mentioned and the current studies regarding topic in the related literature are presented. Artificial intelligence and the deep learning, newly developing technology in its subfield, take place in the second part. In this part, deep learning methods and areas of usage are explained with examples. In the third section, detailed information is given about the data set and the structures used in the application are introduced. In the fourth chapter, the accuracy, sensitivity and specificity values obtained according to the optimization methods used were evaluated visually. In the last section, the results obtained from the diagnostic application have been compared with similar or connected studies in the literature and an evaluation has been made and some prospective suggestions have been made in the light of this study.

2019, Page: 115

Keywords: Deep learnig, artifacal intelligence, Parkinson disease, disease diagnosis, tensorflow, keras, python, spect, dicom.

TEŞEKKÜR

Yüksek Lisans eğitimim boyunca, benden bilgi ve deneyimlerini esirgemeyen, çalışmalarımın tamamlanabilmesi için her türlü şartı sağlayan ve bana her zaman her türlü desteği sunan çok değerli danışman hocam Sayın Doç. Dr. Aytürk KELEŞ'e ve Dr. Öğretim Üyesi Ali KELEŞ'e teşekkürlerimi sunarım.

Atatürk Üniversitesi Uzaktan Eğitim Uygulama ve Araştırma Merkezinde beraber çalışma fırsatı bulduğum, bilgi ve deneyimlerinden yararlandığım Sayın Prof. Dr. Selçuk KARAMAN, Doç. Dr. Engin KURŞUN, Dr. Öğretim Üyesi Ömer ARPACIK, Dr. Öğretim Üyesi M. Fatih ALAEDDİNOĞLU'na teşekkürlerimi sunarım.

Tez çalışmasında ve eğitim hayatımda görüşlerinden faydalandığım beni daima destekleyen eşime teşekkür ediyorum. Hayatımın her anında yanımda olan, her türlü desteği ile elimden tutan, evlatları olmaktan onur duyduğum annem ve babama teşekkürlerimi borç bilirim.

Bu tez çalışmasında tıbbi bilgi desteğinden yararlandığım Atatürk Üniversitesi Nöroloji Anabilim Dalı Dr. Öğr. Üyesi Mustafa CEYLAN'a teşekkürlerimi sunarım.

19/07/2019 Okan ALKAN

İÇİNDEKİLER

ÖZET.....	ii
ABSTRACT.....	iii
TEŞEKKÜR.....	iv
SİMGELER ve KISALTMALAR DİZİNİ.....	vii
ŞEKİLLER VE ÇİZELGELER DİZİNİ.....	viii
1. GİRİŞ	1
2. KURAMSAL TEMELLER.....	3
2.1. Yapay Zekâ Nedir?	3
2.2. Derin Öğrenme Nedir?.....	6
2.2.1. Derin Öğrenme Metotları.....	9
2.2.2. Derin Öğrenmede Kullanılan Araçlar ve Diller	12
3. MATERYAL ve YÖNTEM.....	20
3.1. Çalışma Veri Seti	20
3.2. Geliştirme Ortamları ve Platformları	20
3.2.1. CPU, GPU ve TPU.....	20
3.2.2. CUDA Kütüphanesi	24
3.2.3. Anaconda.....	25
3.2.4. Sypder	26
3.2.5. Tensorflow	29
3.2.6. Keras	31
3.2.7. Derin Öğrenme Ortamı Kurulumu	41
3.2.8. Kütüphanelerin Kurulumu	49
3.3. Veri Ön İşleme	50
3.3.1. Veri Seti Kurulumu.....	50
3.3.2. Veri Seti İnceleme.....	55
3.3.3. Veri Setini Uyumlu Hale Getirme.....	62
3.3.4. Eğitim ve Test Seti Oluşturma	66
3.4. Modelleme ve Sonuçlar.....	68
3.4.1. Model Katmanları Oluşturma	69
3.4.2. Model Derleme Kriterleri.....	71
3.4.3. Model Eğitim İşlemi	72
3.4.4. Parametre Ayarlamaları	72
3.4.5. Eğitimi Görselleştirme	73
3.4.6. Eğitilmiş Modeli Test Etme	75

4. ARAŞTIRMA BULGULARI	77
4.1. SGD Başarı Oranı	77
4.2. RMSprop Başarı Oranı.....	79
4.3. Adagrad Başarı Oranı.....	81
4.4. Adadelta Başarı Oranı	83
4.5. Adam Başarı Oranı.....	85
4.6. Adamax Başarı Oranı	87
4.7. Nadam Başarı Oranı.....	90
5. SONUÇ, TARTIŞMA ve ÖNERİLER	93
5.1. Sonuç.....	93
5.2. Tartışma.....	95
5.3. Öneriler	97
KAYNAKLAR	98
ÖZGEÇMİŞ	103

SİMGELER VE KISALTMALAR DİZİNİ

YZ	Yapay Zekâ
PH	Parkinson Hastalığı
PD	Parkinson Disease
CPU	Central Processing Unit – Merkezi İşlem Birimi
GPU	Graphics Processing Unit – Grafik İşleme Birimi
TPU	Tensor Processing Unit – Tensör İşleme Birimi
CUDA	Compute Unified Device Architecture
cuDNN	CUDA Deep Neural Network
PPMI	Parkinson’s Progression Markers Initiative
	Parkinson İlerleme Belirtileri Girişimi
DICOM	Digital Imaging and Communications in Medicine
	Tıpta Dijital Görüntüleme ve İletişim
DCM	DICOM dosya uzantısı
SPECT	Single Photon Emission Computerized Tomography
	Tek Foton Emisyonlu Bilgisayar Tomografisi
SWEDD	Scans Without Evidence for Dopaminergic Deficit
	Dopaminerjik için Açık Kanıt Bulunmayan Taramalar

ŞEKİL VE ÇİZELGELER DİZİNİ

Şekil 1. Yapay Zekâ Katmanları	4
Şekil 2. Alan Turing Testi Görseli	5
Şekil 3. Yapay Zekâ, Makine Öğrenmesi ve Derin Öğrenme İlişkisi	6
Şekil 4. Derin öğrenme kullanılarak üretilen resim örneği	8
Şekil 5. Sınıflandırma Metodu Gösterimi	10
Şekil 6. Synapse Technology firmasına ait nesne tespit görüntüsü	21
Şekil 7. CPU ve GPU performans seviyeleri	22
Şekil 8. Google TPU v2 görüntüsü	23
Şekil 9. CPU, GPU ve TPU hesaplama birim göstergeleri	23
Şekil 10. Farklı alanlara özgü CUDA kütüphaneleri	24
Şekil 11. Anaconda platformu açılış ekranı	26
Şekil 12. Spyder kod editörü görüntüsü	27
Şekil 13. Spyder değişken görüntüleme ekranı	28
Şekil 14. Spyder konsol ekranı	28
Şekil 15. Spyder yardım ekranı	29
Şekil 16. Tensorflow kullanan firmalar	31
Şekil 17. 5x5 boyutundaki resim üzerine 3x3 çekirdek boyutu uygulama örneği	34
Şekil 18. Sinir ağı katmanında doldurma uygulama örneği	35
Şekil 19. Relu ve Sigmoid fonksiyonlarının grafiksel gösterimi	36
Şekil 20. Aktivasyon fonksiyonlarının grafiksel gösterimi	36
Şekil 21. Maximum Pooling (Havuzlama) işlemi çalışma mantığı	37
Şekil 22. Dropout uygulanmış sinir ağı örneği	38
Şekil 23. Nvidia Tesla P100 performans grafiği	42
Şekil 24. GPU desteği verilen GeForce ürünlerinin listesi	42
Şekil 25. Anaconda yazılımını indirme ekranı	43
Şekil 26. Anaconda yüklenen klasörün yol ayarı	43
Şekil 27. Python sistem değişkenlerine ekleme ayarı	44
Şekil 28. Nvidia ekran kartı sürücü yükleme ekranı	44
Şekil 29. CUDA Toolkit sistem seçim ekranı	45
Şekil 30. CUDA Toolkit indirme bağlantısı	45
Şekil 31. cuDNN Archive yazılımı indirme ekranı	46
Şekil 32. Tensorflow GPU doğru yüklendiğini gösteren ekran	48
Şekil 33. Spyder python interpreter ayar ekranı	49
Şekil 34. Veri seti için belirlenen kriterlerin seçimi	52
Şekil 35. PPMI veri seti arama sonuçları	52
Şekil 36. Veri arttırma yöntemi uygulanmış resim örneği	53
Şekil 37. Kontrol grubundan bir bireyin görüntülediği ekran	54
Şekil 38. PPMI web sayfasında kullanıcı koleksiyon ekranı	55
Şekil 39. Dicom dosyasından alınan metinsel veriler	57
Şekil 40. Dicom dosyasından alınan 41. kesit görüntüsü	58
Şekil 41. Dicom dosyasından alınan 91 adet kesitin görüntüsü	60
Şekil 42. Control grubu dicom dosyaları klasör görüntüsü	61
Şekil 43. Okunan resimlerin Numpy dizilerine dönüştürülmesi	64
Şekil 44. Resimlerden okunan piksel değerlerinin saklandığı X dizisi	64
Şekil 45. Resimlerin karşılığı olan değerlerin saklandığı Y dizisi	65
Şekil 46. Değişkenleri şekillendirme ve kategorik dönüşüm sonrası görüntüleri	66

Şekil 47. Oluşturulan modelin eğitim verileri için katman ve çıktı verileri	70
Şekil 48. Eğitim görselleştirme grafikleri	75
Şekil 49. SGD optimizer, Relu-Softmax activation, 100 epoch, 128 batch size ile doğruluk (92,46) grafiği	78
Şekil 50. SGD optimizer, Relu-Softmax activation, 50 epoch, 128 batch size ile doğruluk (92,33) grafiği	78
Şekil 51. SGD optimizer, Relu-Softmax activation, 50 epoch, 64 batch size ile doğruluk (92,27) grafiği	79
Şekil 52. RMSprop optimizer, Tanh-Softmax activation, 50 epoch, 128 batch size ile doğruluk (93,63) grafiği	80
Şekil 53. RMSprop optimizer, Relu-Sigmoid activation, 20 epoch, 64 batch size ile doğruluk (93,18) grafiği	80
Şekil 54. RMSprop optimizer, Relu-Softmax activation, 100 epoch, 128 batch size ile doğruluk (93,05) grafiği	81
Şekil 55. RMSprop optimizer, Relu-Softmax activation, 20 epoch, 64 batch size ile doğruluk (93,05) grafiği	81
Şekil 56. Adagrad optimizer, Relu-Sigmoid activation, 100 epoch, 128 batch size ile doğruluk (92,72) grafiği	82
Şekil 57. Adadelat optimizer, Elu-Softmax activation, 100 epoch, 64 batch size ile doğruluk (93,63) grafiği	84
Şekil 58. Adadelat optimizer, Relu-Sigmoid activation, 50 epoch, 64 batch size ile doğruluk (93,63) grafiği	84
Şekil 59. Adadelat optimizer, Elu-Softmax activation, 50 epoch, 128 batch size ile doğruluk (93,57) grafiği	85
Şekil 60. Adadelat optimizer, Tanh-Sigmoid activation, 100 epoch, 64 batch size ile doğruluk (93,44) grafiği	85
Şekil 61. Adam optimizer, Elu-Sigmoid activation, 20 epoch, 64 batch size ile doğruluk (93,89) grafiği	86
Şekil 62. Adam optimizer, Selu-Softmax activation, 100 epoch, 128 batch size ile doğruluk (93,57) grafiği	87
Şekil 63. Adam optimizer, Relu-Sigmoid activation, 50 epoch, 128 batch size ile doğruluk (93,50) grafiği	87
Şekil 64. Adamax optimizer, Elu-Softmax activation, 50 epoch, 64 batch size ile doğruluk (94,15) grafiği	88
Şekil 65. Adamax optimizer, Tanh-Sigmoid activation, 100 epoch, 64 batch size ile doğruluk (93,63) grafiği	89
Şekil 66. Adamax optimizer, Selu-Sigmoid activation, 50 epoch, 64 batch size ile doğruluk (93,44) grafiği	89
Şekil 67. Adamax optimizer, Tanh-Softmax activation, 100 epoch, 64 batch size ile doğruluk (93,44) grafiği	90
Şekil 68. Nadam optimizer, Tanh-Softmax activation, 20 epoch, 64 batch size ile doğruluk (93,70) grafiği	91
Şekil 69. Nadam optimizer, Relu-Softmax activation, 50 epoch, 128 batch size ile doğruluk (93,31) grafiği	91
Şekil 70. Nadam optimizer, Relu-Sigmoid activation, 20 epoch, 64 batch size ile doğruluk (93,24) grafiği	92
Çizelge 1. Kullanılan veri seti içeriği	20
Çizelge 2. Veri saklama ortamlarının boyut örnekleri	30

Çizelge 3. Keras ile oluşturulmuş örnek model	32
Çizelge 4. Tahmini yapılan değerin çıktı verileri	39
Çizelge 5. Veri seti için belirlenen kriter ve değerleri	51
Çizelge 6. Veri seti araştırma grubu cinsiyet dağılımları	56
Çizelge 7. Veri seti araştırma grubu yaş aralığı dağılımları	56
Çizelge 8. Dicom görüntüsünü incelemek için gerekli kodlar.....	57
Çizelge 9. Dicom dosyasından tüm kesitleri görüntülemek için gerekli kodlar	59
Çizelge 10. Dicom dosyasından ilgili kesitlerin resim olarak ayrılması	62
Çizelge 11. Resim dosyalarını Numpy dizisine dönüştüren kodlar	63
Çizelge 12. Veri seti üzerinde değişken örneği	66
Çizelge 13. Eğitim ve Test değişkenlerinin isimlendirilmesi	67
Çizelge 14. Verileri eğitim ve test olarak ayırma ve kaydetme	68
Çizelge 15. Çalışma için oluşturulan örnek model yapısı	70
Çizelge 16. Model derleme ayarları kod örneği.....	71
Çizelge 17. Keras kütüphanesi optimizier değerleri	71
Çizelge 18. Model eğitimi kod bloğu	72
Çizelge 19. Eğitim işlemi için kullanılacak epoch ve batch size değerleri.....	72
Çizelge 20. Kullanılacak aktivasyon fonksiyonları	73
Çizelge 21. Eğitim görselleştirme kod örneği.....	73
Çizelge 22. Eğitilmiş modeli bir resim üzerinden test etmek için gerekli kodlar.....	76
Çizelge 23. SGD optimizier ile test edilmiş eğitim doğruluk, duyarlılık ve özgüllük oranları	77
Çizelge 24. RMSprop optimizier ile test edilmiş eğitim doğruluk, duyarlılık ve özgüllük oranları	79
Çizelge 25. Adagrad optimizier ile test edilmiş eğitim doğruluk, duyarlılık ve özgüllük oranları	81
Çizelge 26. Adadelat optimizier ile test edilmiş eğitim doğruluk, duyarlılık ve özgüllük oranları	83
Çizelge 27. Adam optimizier ile test edilmiş eğitim doğruluk, duyarlılık ve özgüllük oranları	85
Çizelge 28. Adamax optimizier ile test edilmiş eğitim doğruluk, duyarlılık ve özgüllük oranları	87
Çizelge 29. Nadam optimizier ile test edilmiş eğitim doğruluk, duyarlılık ve özgüllük oranları	90
Çizelge 30. Hastalık teşhisi başarı durum sıralaması.....	94
Çizelge 31. Hastalık teşhisi en iyi 3 başarı durum sıralaması.....	95

1. GİRİŞ

Parkinson hastalığı (PH), beynin alt kısmında oluşan hasar ile dopamin salgılayan hücrelerin tahrip olması sonucu ortaya çıktığı bilinmektedir (Akbayır, 2017). Dopamin salgısının eksikliği zamanla; hareket hızında azalma, denge problemi, titreme gibi sorunları ortaya çıkarmaktadır (Güçlü Gündüz, 2009). Parkinson hastalığı; hasta ile yakınları arasında, sosyal çevrede ve sağlık açısından birçok olumsuzluğa neden olan önemli bir hastalıktır. Hastalığın erken tanısı ile uygun ilaç ve doğru bir uygulama yapılarak tedavi başarısı artırılabilir (Özyiğit, 2016).

Parkinson hastalığında erken teşhis tedavinin başarısı için büyük öneme sahiptir. Bu sebeple çalışmamızda derin öğrenme yöntemleri ile SPECT (Single Photon Emission Computed Tomography - Tek Foton Emisyonlu Bilgisayar Tomografisi) görüntüleri üzerinden hastalık teşhisi tahmini yapılmıştır. Çalışmamızda yüksek doğruluk oranı ile hastalık teşhisi yapılması amaçlanmaktadır. Ayrıca, tahmin için kullanılan yöntemlerin performans durumları değerlendirilmiştir.

SPECT beyindeki kan akışı hakkında bilgi almak amaçlı 3 boyutlu olarak kaydedilen nükleer tıp görüntüleme yöntemidir. Hastalık teşhisinde beyin yapısı hakkında bilgileri bilgisayarlı tomografi ve manyetik rezonans görüntüleme ile elde etmek mümkündür. Ancak SPECT görüntülemeye beyin kan akışı ile diğer yöntemlerde görüntülenemeyecek bilgiler sağlanmaktadır (Kargın, 2007).

Derin öğrenme, Makine öğrenmesi teknikleri arasında yer alan bir teknolojidir. Derin öğrenme terimi ile gündelik yaşantımızda insanoğlunun öğrenme yetisi bilgisayarlara aktararak veriler üzerinden istenilen işlemlerin gerçekleştirilmesi amaçlanmaktadır. Derin öğrenme, son yıllarda popülaritesi artmakla beraber pek çok alanda uygulamaları bulunmaktadır. Güvenlik sistemleri, insansız hava taşıtları, hastalık teşhisi, veri güvenliği, mali işlemler, doğal dil işleme ve online arama gibi pek çok teknoloji üretiminde veriler üzerinden çıkarım yapmak son derece önemlidir. Bu verilerden istenilen çıkarımların yapılmasında derin öğrenme büyük rol oynamaktadır.

Alandaki çalışmalara bakıldığında parkinson hastalıkları üzerine farklı veri setleri ve derin öğrenme yöntemleri kullanılarak çeşitli çalışmalar yapılmıştır. Prashanth et al. (2016) görüntü üzerinden özellik çıkarımı yöntemi ile belirli nitelikler çıkarmışlardır. Bu niteliklerde Parkinson hastalarındaki değerlerin, normal insanlara göre daha aşağıda olduğu tespit edilmiştir. Daha sonra bu özelliklere bağlı olarak bilgisayar üzerinden öğrenme algoritması uygulanmış ve sonuçları SVM (Support Vector Machine), Boosted Trees, Random Forest ve Naive Bayes sınıflandırma yöntemleri ile değerlendirilmiştir.

Zhang and Kagen (2017) yaptıkları çalışmada sağlık alanında kullanılmak üzere makine öğrenmesi arayüzü tasarlamışlardır. Yapay sinir ağı kullanılarak beyin görüntüleri üzerinden hastalığın erken teşhisini amaçlamışlardır. Yapılan çalışmalar sonucunda hastalık teşhisinde %93,8 başarı sağlamışlardır.

Rana et al. (2015) MRI görüntülerini kullanarak Parkinson hastaları üzerinde grafik-teori tabanlı “spectral” görüntüleri incelemiştir. Görüntüler üzerinden VV-only ve VV-GBSFS sınıflandırma metotları kullanılarak bilgisayar destekli tanı işlemi gerçekleştirilmiş ve %86,67 başarı sağlanmıştır. Oliveira and Branco (2015) SVM ve Voxel-as-Features Approach sınıflandırma yöntemleri ile görüntüler üzerinden bilgisayar destekli Parkinson hastalığı tespiti üzerine çalışmışlardır.

Prashanth and Roy (2018) MDS-UPDRS (Hareket Bozukluğu Derneği – Birleşik Parkinson Hastalığı Derecelendirme Ölçeği) ve hastalık evresi çıkarımı için Hoehn ve Yahr (HY) ölçeği ile Parkinson hastalığı için yeni ve geliştirilmiş aşama tahmini önermişlerdir. Görüntüler üzerinden 59 adet özellik çıkarımı yapılmış ve MATLAB Machine Learning Toolkit ve “libsvm” kütüphaneleri kullanılarak çalışmalar gerçekleştirilmiştir.

Parkinson hastalığında hasta verilerinin hızlı değerlendirilmesi ve doğru teşhis tedavi sürecinde hayati öneme sahiptir. Derin öğrenme teknikleri doğru ve hızlı veri analizi amacıyla günümüzde yaygın olarak kullanılmaktadır. Bu tez çalışmasında derin öğrenme yöntemleri kullanılarak SPECT görüntüler üzerinden hastalık teşhisi tahmini yapılmıştır. Parkinson hastalığının teşhisine yönelik yapılan bu tez çalışmasında Atatürk Üniversitesi Nöroloji Anabilim Dalı Dr. Öğr. Üyesi Mustafa CEYLAN’dan tıbbi bilgi desteği alınmıştır.

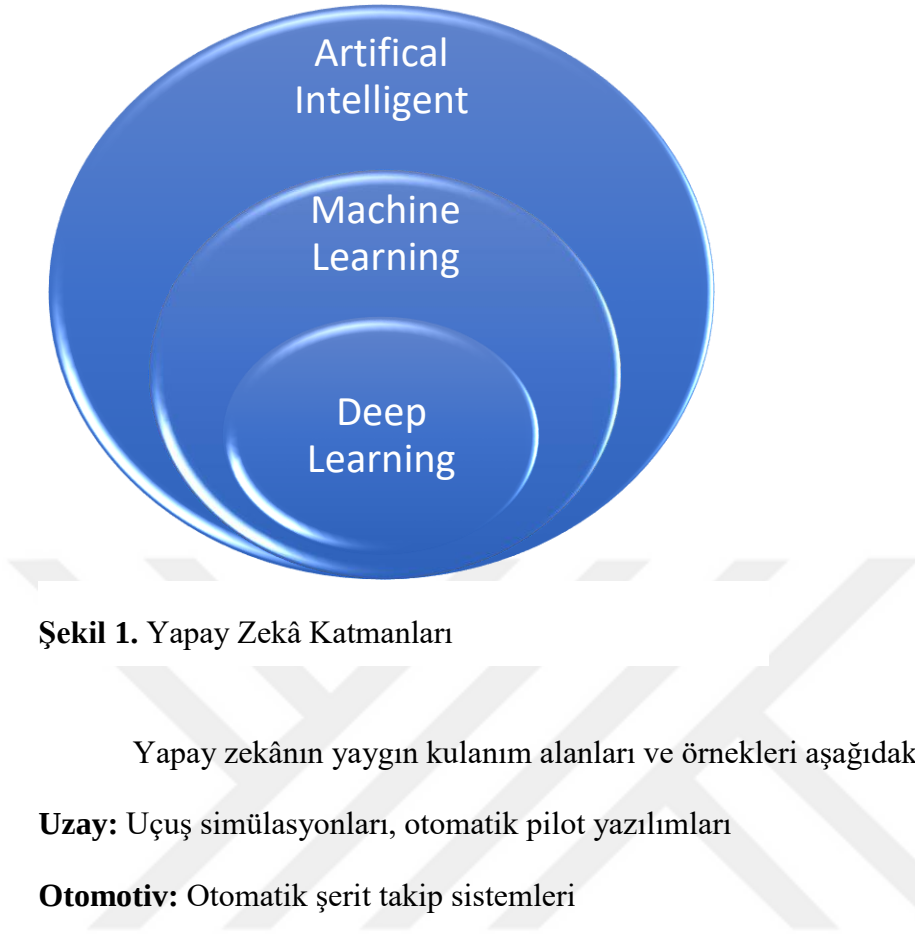
2. KURAMSAL TEMELLER

2.1. Yapay Zekâ Nedir?

Uygulama alanı oldukça geniş olan Yapay Zekâ (YZ) kavramı için uzman kişiler tarafından pek çok tanım yapılmıştır. Yapay Zekânın amacı, insan zekâsıyla yapılabilecek işlerin makineler tarafından yapılmasını sağlamaktır. Yapay Zekâ araştırmalarının amacı, zekâ gerektiren durumları bilgisayarların anlayabilmeleri için gerekli alt yapıyı hazırlamaktır. Bu işlemin gerçekleşmesi için ilk önce insan düşünce yapısının anlaşılması gerekir. Daha sonra bu yapı bilgisayar ortamında uygulanabilmelidir. Yapay Zekâ, insanoğlunun hayatını kolaylaştırmak adına makineleri daha akıllı ve faydalı duruma getirmek amacıyla kullanılmaktadır. İnsan hayatında birçok davranış türü zekâ belirtisi olarak kabul edilmektedir. (Yılmaz, 2017). Bu davranışlar:

- Deneyim yoluyla öğrenme,
- Yeni bir duruma adaptasyon,
- Karmaşık yapılardan anlam çıkarma,
- Bilgiyi alma ve aktarma,
- Düşünme, tahmin ve karar verme,

Yapay Zekâ, insan zekâsının çalışma yapısını bilgisayarlara aktararak (insan gibi algılama, akıl yürütme, anlama, sonuç çıkarma, genelleme yapabilme vb.) bu özelliklere sahip bilgisayar sistemleri yapmayı hedefleyen bilim dalıdır. Makine öğrenmesi; yapay zekânın bir alt koludur ve problemleri çözmek için belli yöntemleri (algoritmaları) kullanmaktadır. Yapay Zekâyı sağlama yaklaşımıdır. Derin öğrenme; insan beyninin algılama ve karar verme becerisini bilgisayara kazandırmayı amaçlar. Diğer bir deyişle makine öğrenmesini gerçekleştiren teknikler (Koç, 2018; Nahiye, 2016).



Şekil 1. Yapay Zekâ Katmanları

Yapay zekânın yaygın kullanım alanları ve örnekleri aşağıdaki gibidir:

Uzay: Uçuş simülasyonları, otomatik pilot yazılımları

Otomotiv: Otomatik şerit takip sistemleri

Bankacılık: Müşteri analizi, akıllı kampanya yönetimi

Savunma: Radar, Sonar sistemler, İnsansız hava aracı, Silahlı insansız hava aracı

Finans: Pazar analizi, Bütçe, Borsa

Sağlık: Kanser teşhisi, hastalık yönetimi, akıllı sağlık uygulamaları

Robotik: Yörünge kontrolü, Uzaktan kumandalı sistemler

Dil: Sözcük tanıma, yazı-ses veya ses-yazı çevirimi, tercüme

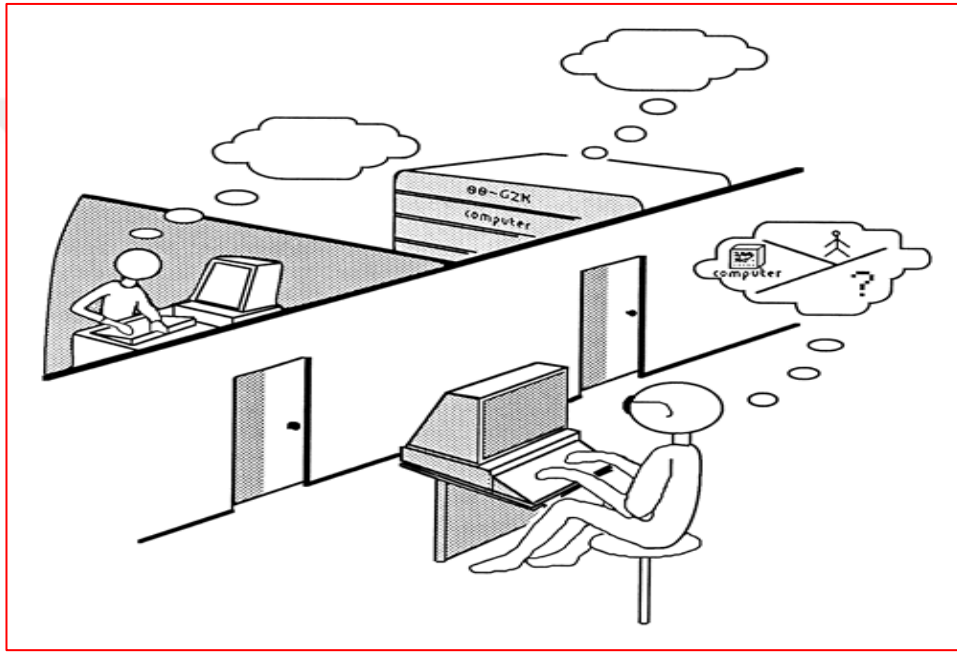
Telekomünikasyon: Veri karşılaştırma, filtreleme işlemleri, ses işleme, görüntü işleme, trafik yoğunluk kontrolü

Güvenlik: Göz tanıma, Ses tanıma, Yüz tanıma, Parmak izi tanıması (Yılmaz, 2017)

Yapay Zekânın tarihinin çok eskilere dayandığı bilinmektedir. 1884 yılında Charles Babbage, zeki davranışlarda bulunabilecek mekanik makineler üzerinde

çalışmış fakat başarılı olamamıştır. 1950’li yıllarda Shanon, bilgisayar üzerinde satranç oyunu oynanabileceği fikrini sundu. 1965-1970 yılları arasında yapay zekâ üzerinde karanlık bir dönem olarak karşımıza çıksa da 1970 yılından sonra yapay zekâ çalışmaları büyük hız kazandı. Kronolojik olarak Yapay Zekâ tarihçesine bakılacak olursa:

- 1950– Turing Testi: “Mind” adlı felsefe dergisinde “Computing Machinery & Intelligence” adıyla Turing testinden ve zekâ kavramından bahsedildi. (Turing A. M. 1950)



Şekil 2. Alan Turing Testi Görseli¹

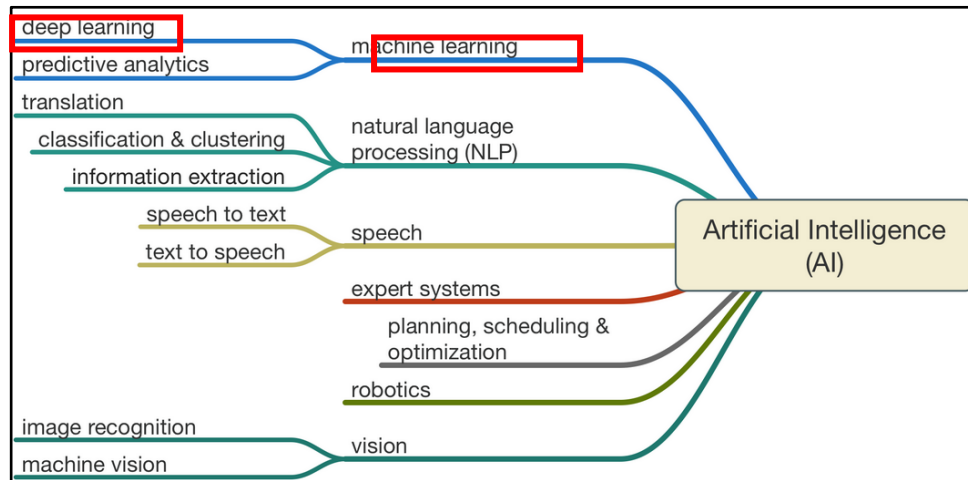
- 1956- Dartmouth Konferansında “Yapay Zekâ” ismi ilk kez ortaya atıldı. 8 hafta süren bu konferansda yapay sinir ağları, sinyal işleme, mantık teorisi ve bilişsel bilim üzerine konuşuldu (McCarthy J., Minsky M. L., Rochester N., Shannon C. E. 1956) (Koç, 2018).
- 1952-1969- IBM tarafından satranç oynayabilen ilk programı yazıldı (Koç, 2018).

¹ http://www.alanturing.net/turing_archive/pages/Reference%20Articles/TheTuringTest.html adresinden alınmıştır.

- 1958- MIT'den McCarty tarafından LISP (List Processing Language) dili geliştirildi.
- 1980- Yapay Zekâ Endüstri haline geldi.
- 1986- Yapay sinir ağları tekrar popüler oldu.
- 1987- Yapay Zekâ bilim haline geldi.
- 1997- Deep Blue isimli YZ süper bilgisayar Kasparov'u yendi.
- 1998- İnternetin yaygınlaşması ile YZ tabanlı birçok program geniş kitlelere ulaştı.
- 2000-2005- Robot oyuncaklar piyasaya sürüldü. Bugün birçok elektronik cihazda YZ uygulamaları kullanılmaktadır.
- 2005- YZ'ya sahip, insan yeteneği ve becerisine en yakın sayılan ASIMO adındaki robot tanıtıldı.

2.2. Derin Öğrenme Nedir?

Derin öğrenme, Makine öğrenmesi teknikleri arasında yer alan bir teknolojidir. Derin öğrenme terimi ile gündelik yaşantımızda insanoğlunun öğrenme yetisi bilgisayarlara aktarılarak veriler üzerinden istenilen işlemlerin gerçekleştirilmesi amaçlanmaktadır (Alpaydın, 2018).



Şekil 3. Yapay Zekâ, Makine Öğrenmesi ve Derin Öğrenme İlişkisi²

² <https://www.legaltechnology.com/latest-news/artificial-intelligence-in-law-the-state-of-play-in-2015/> adresinden alınmıştır.

Derin öğrenme konusunu irdelemeden önce veri kavramına detaylı bir şekilde bakılmalıdır. İnsan öğrenmesinde olduğu gibi derin öğrenme teknolojisinde de veriler üzerinden öğrenme gerçekleşir. Her gün bilgisayar ve mobil cihazların kullanımının da yaygınlaşmasıyla ciddi miktarda veri üretimi yapılmaktadır. Bu kullanım sonucunda da büyük veri kavramı ortaya çıkmıştır.

Büyük veri; sosyal medya, blog, fotoğraf, video ve loglardan toplanan verilerin anlamlı ve işlenebilir şekilde halidir. Büyük veri kavramında büyük kelimesinin ne kadar büyük olduğu ifadesine bakmak gerekirse; bugün dünyanın en büyük kütüphanesi Amerika'dadır. Bu kütüphanedeki kaynaklar dijital ortama aktarıldığında 462 Terabyte'lık veri elde edilir. Günümüzde 1 yılda 8 Zettabyte veri üretimi yapılmaktadır (Gürsaka, 2017).

Büyük veri doğru analiz ile şirketlerin stratejik kararlarını doğru biçimde almalarına, risklerini daha iyi yönetmelerine ve inovasyon yapmalarına imkân sağlar. Örnek verilecek olursa havayolu şirketi A ve B şehirleri arasında uçuş seferi koyma konusunda GSM şirketleri ile anlaşılabilir. Bununla beraber aynı gün içinde her iki şehirde cep telefonundan sinyal alınan kişilerin sayısı ile havayolu şirketi stratejik bir karar verebilir.

Firmalar kendi veri tabanlarında (ya da dış kaynaklardan ulaşabildikleri) verileri doğru biçimde işleyebilirse rekabet avantajına sahip olabilirler. Veriye erişimden sonra en önemli kısım veriyi işlerken ve bilgiyi açığa çıkarırken kullanılan algoritmalarıdır. Bu algoritma örnekleri aşağıdaki gibidir (Köse, 2018).

- Karar Ağaçları
- En Yakın Komşu Bulma
- K-means Kümeleme
- Sınıflama
- Kural Çıkarımı

Verinin işlenebilecek şekilde saklanabilmesi ve veri üzerinde sorguların yapılabilmesiyle veri madenciliği terimi ortaya çıktı. Veri madenciliği terimi bu

üretileen verinin en iyi şekilde nasıl kullanılabileceğı yönünde yapılan akademik çalışmalar sonucunda doğmuştur (Akpınar, 2017).

Verilerin arasında en çok alışveriş yapmış müşteri ve en karlı müşteri gibi sorulara cevap bulunabilir. Ancak rekabet ortamında daha iyi noktalara gelmek amacıyla yeni üretilen ürünü alma olasılığı en yüksek müşteri, pazarlama kampanyalarına dönüş yapma ihtimali yüksek müşteri gibi soruların cevapları veri madenciliğı kavramlarının kullanımı ile elde edilir. Verilerin hangi iş problemlerinde ne şekilde kullanılabileceğı ise derin öğrenme konusunun en temel unsurudur. Aşağıdaki resimde Andrej Karpathy ve Li Fei-Fei tarafından derin öğrenme metodu kullanılarak eğitimi yapılmış bir makinenin vermiş olduğı sonuç görüntülenmektedir. Görüntü üzerinde nesne tanımlamanın yanı sıra nesneler arasındaki ilişki düzeyinin tespiti de ayrıca yapılmıştır (Karpathy and Fei-Fei, 2015).



Şekil 4. Derin öğrenme kullanılarak üretilen resim örneğı

Kaya (2017) tez çalışmasında derin öğrenme alt yapısı, Python ve Objective-C programlama dilleri kullanılarak anaokulu öğrencilerine rakam öğretme uygulaması geliştirilmiştir. Rakam Öğrenme Sistemi ile derin öğrenme ve yapay zekânın eğitim sektörü ile iç içe geçebileceğı ve faydalı olabileceğı gösterilmiştir.

Elitez (2015)'in tez çalışmasında el yazısı olarak yazılmış belgeler üzerinde çalışmıştır. Derin öğrenme içerisinde yapay sinir ağına öğretme işlemi gerçekleştirilmiş ve rakam sınıflandırma işlemleri için DBN (Deep Belief Networks) ve SVM (Support Vector Machines) yöntemleri kullanılarak sonuçlar karşılaştırılmıştır.

Ağ saldırı tespiti sistemleri her geçen gün daha da popülerleşmektedir. Aygün (2017) tez çalışmasında makine öğrenmesi yöntemlerine göre daha başarılı sonuçlar alınan derin öğrenme yöntemlerinden yapay sinir ağlarını kullanarak, NSL-KDD veri seti üzerinde anomali tespiti tabanlı ağ saldırı tespiti yapmıştır.

Anwer (2017) tez çalışmasında, göğüs kanseri teşhisi için uygulanan derin öğrenme algoritmalarının (Tam Bağlantılı Yapay Sinir Ağı, Konvolüsyonel Sinir Ağı, Basit Tekrarlayan Ağ, Uzun Kısa Dönem Ağları ve Kapılı Yinelenen Birim) sonuçlarının karşılaştırmalı analizini yapmış ve başarılı sonuçlar elde etmiştir.

2.2.1. Derin Öğrenme Metotları

Derin öğrenmede temel olarak aşağıdaki metotlar uygulanır.

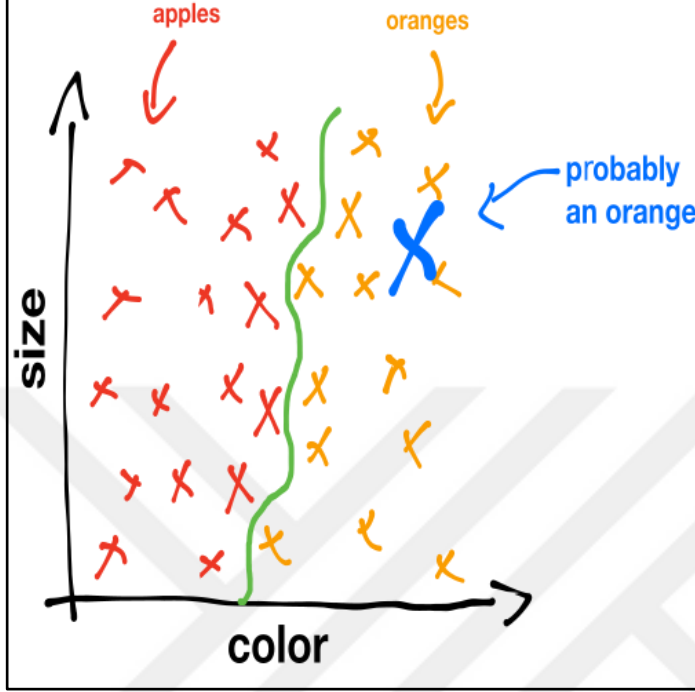
- Sınıflandırma (Classification)
- Kümeleme (Clustering)
- Eğri Uydurma (Regresyon)
- Birlikte Kural Çıkarımı (Association Rule Mining)
- Transkripsiyon (Transcription)
- Makine Çevirisi (Machine Translation)
- Anormal Durumların Farkındalığı (Anomaly Detection)
- Gürültü Azaltma (Noise Reduction)
- Metin Özetleme (Text Summarization)

Sınıflandırma (Classification)

Sınıflandırma yönteminde giriş ve çıkış parametreleri vardır. Giriş parametrelerine bakarak çıkış parametresi belirlenir (Akpınar, 2017). Yani geçmiş bilgilerin hangi sınıftan olduğu biliniyorsa, yeni gelen verinin hangi sınıfa dâhil olacağını bulmasıdır (Bilgin, 2018).

Elimizde bulunan meyvelerin renklerine göre sınıflandırılması problemi çözülmek istenebilir. Bu problemi çözmek için önce bir model oluşturulur ve

verilerle bu model eğitilerek cevabı aranan sınıflandırma işlemleri tahmin edilmeye çalışılır. Bu problemi çözmek için renk ve büyüklük değerlerine bakılarak meyvenin hangi sınıfta yer aldığı belirlenebilir (Gürsaka1, 2017).



Şekil 5. Sınıflandırma Metodu Gösterimi³

Kümeleme (Clustering)

Kümeleme yönteminde ise veri seti içerisinde yer alan değişkenler arasındaki ilişkiler ortaya konur. Geçmişteki verilerin sınıfları/etiketleri verilmediği/bilinmediği durumlarda verilerin birbirlerine yakın benzerliklerinin yer aldığı kümelerin bulunmasıdır (Akpınar, 2017; Gürsaka1, 2017).

Eğri Uydurma (Regresyon)

Bu yöntemle örneğin aylık satış miktarlarının bulunduğu veri setinden yararlanarak gelecek aylar için satış tahminlerinde bulunulabilir. Training (eğitim) sürecinden sonra bilgisayara gelecek aylara ait satışları tahmin edebilecek bir x y eğrisi oluşturulabilir (Bilgin, 2018).

³ <https://makineogrenimi.wordpress.com/2017/05/26/siniflandirma-classification/> adresinden alınmıştır.

Birliktelik Kural Çıkarımı (Association Rule Mining)

Bir süpermarkette, “x” ürününü alan müşterilerin %80’i “y” ürününü de alıyorsa, “x” ürünü alıp, “y” ürününü almayan müşteriler, “y” ürününün potansiyel müşterileridir. Müşterilerin sepet bilgilerinin bulunduğu bir veri tabanından potansiyel “y” müşterilerini bulma işlemi türündeki problemler birliktelik kuramı ile çözülebilir (Köse, 2018).

Transkripsiyon (Transcription)

Yapılandırılmamış veriden text formatında veri elde edilmesi işlemidir. Örneğin Speed Recognition (Konuşma Tanıma) uygulamasında uygulanmaktadır. Apple telefonlarında bulunan “SİRİ” uygulaması buna örnek olarak gösterilebilir (Nilsson, 2018).

Makine Çevirisi (Machine Translation)

Giriş verileri incelenerek ona karşılık gelecek çıkış verilerine çevrilir. Türkçe-İngilizce veya İngilizce-Fransızca çeviri işlemlerinde derin öğrenme kullanılır (Nilsson, 2018).

Anormal Durumların Farkındalığı (Anomaly Detection)

Bir grup olaydan veya nesneden olağandışı olan verileri bulmak için kullanılır. Örneğin bu yöntemle geçmiş kayıtlar kullanılarak kredi kartı dolandırıcılığı tespit edilebilir.

Gürültü Azaltma (Noise Reduction)

Gürültülü yani yapılacak işte kullanılmayacak fazla verinin temizlenmesi için kullanılan yöntemdir.

Metin Özetleme (Text Summarization)

Büyük metinlerin özetlerinin çıkarılmasında kullanılan derin öğrenme metodudur.

Uygulama alanı ne olursa olsun, çok miktardaki verinin analiz edilerek gelecek ile ilgili tahminlerde bulunma ve karar verme konusunda yardımcı olma açısından derin öğrenme yöntemleri her geçen gün kat ve kat önem kazanmaktadır (Köse, 2018).

2.2.2. Derin Öğrenmede Kullanılan Araçlar ve Diller

Yapay zekâ, makine öğrenmesi ve derin öğrenme konuları üzerinde kullanılabilecek ücretli, ücretsiz ve açık kaynak kodlu pek çok platform mevcuttur. Bu platformların hangisinin seçilmesi gerektiği konusunda birçok ölçüt sıralanabilir. Bu alanda kullanılabilecek doğru platformu seçmek verileri işleme aşamasından değerlendirme ve sonuç çıkarma aşamasına kadar büyük öneme sahiptir. Derin öğrenme konusunda yapılacak çalışmalarda; veri seti içerisindeki hangi verilerin kullanılacağı, veriler arasındaki ilişkilerin belirlenmesi ve verilerin hangileri kullanılarak en iyi sonuca varılabileceği sorularına yanıt bulunmalıdır. Bu sorulara yanıt bulurken çok sayıda çalışma yapılması gerekebilir. Dolayısıyla bu platformlar; kullanılan araçtan en hızlı verim alma, çıkarılan sonucu en iyi şekilde görselleştirebilme, veri seti içerisindeki dosya formatı üzerinde işlem yapabilme gibi pek çok özellik açısından birbirinden üstün olabilirler. Bu üstünlükler belirlenirken çok fazla seçenek olduğu için en iyi platformu belirleme mümkün değildir. Bu yüzden yapılmak istenen çalışmaya en iyi yanıt verebilecek programı seçmek; araştırmacılara zaman, kolay erişim, doğru sonuç alma ve maliyet açısından avantaj sağlayacaktır (Doğan, 2017).

Yapay zekâ alanında en çok tercih edilen araçlar ve diller; Matlab, Python, Weka, OpenCV, C++, Java ve C# olarak listelenebilir.

Matlab

Matlab, MathWorks şirketi tarafından geliştirilmiştir. Lineer, istatistik, optimizasyon çalışmaları, analiz işlemleri gibi çok karmaşık matematiksel hesaplamaları bile hızlı bir şekilde yapabilme kapasitesine sahiptir. Polinom, parabol, sinüs ve cosinüs gibi matematiksel hesaplamaların 2 ve 3 boyutlu çizimleri kolaylıkla oluşturulabilir. Bu güçlü özellikleri ile Matlab günümüzde yapay zekâ, sinyal işleme,

görüntü işleme, sayısal analiz gibi alanlarda sıklıkla kullanılan bir programdır (Kubat, 2017).

Nagi et al. (2008) yaptıkları çalışmada 5 farklı yüz ifadesine sahip toplamda 25 görüntüyü, MATLAB Neural Network Toolbox aracı kullanarak 850 eğitim turundan (epoch) geçirmişler ve %81,36 oranında başarı elde etmişlerdir. Bu çalışmada oldukça küçük bir veri seti kullanılmasına rağmen Matlab içerisindeki yapay sinir ağları aracı ile iyi sonuçlar elde edilmiştir.

Kolokotsa et al. (2006) kurdukları sistemde Matlab üzerindeki “Fuzzy Logic Toolbox” aracılığıyla analiz edilen veriler doğrultusunda iç mekân içerisindeki enerjinin verimli kullanılmasını sağlamak amacıyla çalışmışlardır. Bu çalışmada gündelik yaşamda, bina tesisatlarındaki enerjinin daha verimli kullanılması sağlanmaya çalışılmıştır.

Elektronik devreler hazırlanmadan önce, devrenin davranışının test edilmesi hayati öneme sahiptir. Bu test işleminin yapılabilmesi için matematiksel modeller kullanılmaktadır. Bu nedenle devre simülasyonlarının gerçekleştirilmesi günler hatta haftalar boyunca devam edebilmektedir. Salah (2017) yaptığı çalışmada Matlab programında genetik algoritmalar, bulanık mantık ve yapay sinir ağları ile elektronik devrelerin davranışlarını simüle etmeye çalışmıştır. Bu simülasyonlar yapılan çalışma ile hızlı ve doğru bir biçimde gerçekleştirilmiştir.

Yukarıda sunulan örnek çalışmalar görüntü işleme, enerji verimliliği hesaplama ve elektronik alanında Matlab programının başarılı bir şekilde kullanıldığını göstermektedir. Bu çalışmalarda yapay zekâ alanında en çok kullanılan ve tercih edilen yöntemler genellikle genetik algoritmalar, bulanık mantık ve yapay sinir ağlarıdır.

Python

Python Hollanda kökenli bilgisayar programcısı Guido van Rossum tarafından geliştirildi. Python dili tanıtıldıktan sonra kısa sürede pek çok araştırmacı tarafından kullanılarak popüler hale geldi. Bu popülerliğin sebebi arasında aşağıdaki başlıklar gösterilebilir:

- Açık kaynak olarak yayınlanması
- Öğrenilmesi kolay
- Tüm platformlarda çalışabilme
- Geniş topluluk desteği
- Modüler sistem

Python programı ücretsiz olarak yayınlanmaktadır. Yazılan programlar için herhangi bir ücret ödenmesi gerekmemektedir. Diğer programlama dillerine göre okunabilmesi ve öğrenilmesi daha kolaydır. Aynı işi yapan kod bloklarına bakıldığında en az kod satırı Python'da bulunmaktadır. Az kod ile çok iş yapılabilmesi çok fazla alanda tercih edilme sebebi olmuştur.

Programlama dilleri yapısal olarak derleyici ve yorumlayıcı olarak iki gruba ayrılır. Derleyici yazılan programın bilgisayarın anlayabileceği ve çalıştırabileceği makine koduna çeviren programdır. Örnek olarak C# veya Java ile yazılmış kod blokları program üzerinde yer alan derleyici tarafından öncelikle derlenir. Bu derlenme işleminde yazılan tüm kodlar test işleminden geçirilerek kod bloklarında herhangi bir hata olup olmadığı kontrol edilir. Yorumlayıcı ise derlenme işlemi gerektirmeyen, yüklendiği sistem üzerinde doğrudan çalıştırılan programdır. Örnek olarak Php ve Python yorumlayıcı kategorisindeki programlama dillerindendir. Bu programlar ile yazılan kodlar satır satır çalıştırılır. Çalışması gerekmeyen kod blokları gerek duyulmadıkça çalıştırılmaz (Kaya, 2018).

Python ile masaüstü, web, online, offline, mobil gibi farklı platformlarda çalıştırılabilen programlar yazılabilir. Son zamanlarda kullanım oranı artan Raspberry Pi gibi elektronik kartlar üzerinde çalıştırılan programlar da yazılabilmektedir. Yazılan program Unix, Linux, Mac, Windows, Symbian gibi birçok işletim sistemi üzerinde çalışabilmektedir. Bu özellikleri nedeniyle geniş bir topluluk tarafından desteklenen Python son yılların popüler programlama dilleri arasında ilk sıraya yerleşmiştir (Aksoy, 2017).

Python üzerinde işlem yapılacağı zaman çok fazla kod yazılmasına gerek yoktur. Çünkü Python içerisinde standart olan kütüphaneler zaman içerisinde çok fazla geliştirilmiştir. Ayrıca Python için geliştirilmiş olan Pandas, Mathplot ve Numpy gibi işleri kolaylaştıracak pek çok kütüphane mevcuttur. Bu kütüphaneler

diziler üzerinde daha kolay işlem yapmayı mümkün kılar ve verileri daha iyi görselleştirmeyi sağlar. Bu sayede daha az kod ile daha hızlı işlem yapılabilir (Arslan, 2019).

Son yıllarda yapay zekâ alanındaki gelişmelerin artması sonucu Python üzerinde de bu konu ile alakalı kütüphaneler oluşturulmuştur. Bunlardan bazıları bu tez çalışmasında da kullanılan “Tensorflow”, “Keras” ve “Scikit Learn” kütüphaneleridir.

Weka

WEKA, yapay zekâ alt dalı olan makine öğrenmesi alanında sıklıkla kullanılan programlardan biridir. Veri düzenleme, sınıflandırma, regresyon ve kümeleme gibi birçok alanda kullanılan araçları içermektedir. WEKA, GNU (Genel Kamu Lisansı) ve açık kaynak olarak servis edilmektedir. Waikato üniversitesi içerisinde geliştirilen program **Waikato Environment for Knowledge Analysis** (Bilgi Analizi için Waikato Ortamı) kelimelerinin ilk harfleri yazılarak WEKA olarak adlandırılmıştır.

Program içerisinde istatistik ve makine öğrenmesi alanında kullanılan kütüphaneler mevcuttur. Veri işleme, regresyon gibi işlemler dışında veri görselleştirme, özellik çıkarımı gibi işlemler de yapılmaktadır.

Işık ve Elektron mikroskobundan alınan görüntüler üzerinde değerlendirmeler genellikle büyük zaman almaktadır. Bu sebeple işlemlerde tıkanıklığa sebebiyet vermektedir. Arganda-Carreras et al. (2017) aldıkları görüntüleri WEKA programındaki TWS sınıflandırma aracını kullanarak sınıflandırdılar ve otomatik olarak bölümlere ayırdılar.

Dener vd. (2009) yaptıkları çalışmada Gazi Üniversitesinde yüksek lisans yapan öğrencilerinin, mezun oldukları lisans programları ile okudukları yüksek lisans programlarının aynı veya farklı olması durumuna göre başarı seviyelerini ölçmüşlerdir. Bunun için araştırmacılar okuldan aldıkları verileri WEKA aracında yer alan Naive Bayes, Kstar ve Rddor algoritmaları ile değerlendirmişlerdir.

OpenCV

Gelişen teknoloji çağında görüntü algılayıcı donanımların kolay erişilebilir olması görüntü üzerine çalışmaların artmasını sağlamaktadır. Güvenlik, trafik kontrol ve cep telefonlarının kameralarından ulaşılan görüntülerin çözülmesi, içerisindeki bilgilerin açığa çıkarılması büyük önem kazanmıştır. Bina çevresini kuşatan kameralardan gelen görüntüler güvenlik amacıyla kaydedilmektedir. Fakat görüntüleri kaydetmek yerine anlık olarak güvenlik ihlali tespiti yapılması gerekmektedir. Bu noktada görüntü işleme üzerine geliştirilmiş olan OpenCv, yukarıda bahsedilen durumları yapmak için bir araç olarak kullanılmaktadır (Kuyumcu, 2016).

OpenCv açık kaynak kodlu, BSD lisansına sahip görüntü işleme kütüphanesidir. Son sürümlerinde C++ ile geliştirilen bu kütüphane birçok programlama dili içerisinde kullanılabilir. Kütüphane olarak çalıştırılan bu araç Python, Matlab, C# gibi diller üzerinden programlanıp kullanılmaktadır.

Son zamanlarda artan araç sayıları ile bunların kontrolü gün geçtikçe zorlaşmaktadır. Araç tespiti, takibi, sınıflandırılması, plaka tespiti ve araç yoğunluğu kontrolü büyük önem taşımaktadır. Uke ve Thool (2013) bu sorunlardan yola çıkarak OpenCv kütüphanesi ile gerçek zamanlı araç algılama ve sayma işlemi yapmışlardır. Deneysel çalışmalarında tek kameradan alınan video görüntülerinden hafif araç, ağır araç ve motosiklet olarak hareketli araçların tespiti gerçekleştirilmiştir.

C++

1979 yılında Bjarne Stroustrup tarafından C dili kullanılarak geliştirilmeye başlanan bir programlama dilidir. Bjarne Stroustrup kullandığı programlama dillerinin yeterli seviyede işlevli ve yüksek performanslı olmadığı için C++ dilini programlamaya başlamıştır. C++ günümüzde pek çok platformda çalışabilecek sistem, uygulama ve sürücü yazılımları üretmek için kullanılmaktadır. C#, Python, Php ve Java gibi programlama dillerinin oluşturulmasında etkili olan C++ programlama dili günümüzde en çok kullanılanlar arasında yerini korumaktadır. Yapay zekâ çalışmalarında da kullanılan C++ birçok araştırmacı tarafından çalışmalarında kullanılmaktadır.

Trafikte uygulanan ışık süreleri genellikle sabit yapıdadır. Her araç kavşakta kendisine ait olan ışıktaki belirli süreyi bekler. Fakat bu bekleme esnasında trafik yoğunluğu dikkate alınmaz. Bu nedenle bazen bir ışıktaki trafik yoğunluğu oluşabilmektedir. Erdem (2007) yaptığı çalışmada C++ programlama dilindeki bilgisayarlı görü araçlarını kullanarak, kameradan alınan görüntüye göre yoğunluk yaşanan yol kısmındaki trafik akışını rahatlatmaya yönelik uygulama geliştirmiştir.

Bilgisayarlı görü çalışmaları psikolojik değerlendirmelerde de kullanılmaya çalışılmaktadır. Tenekeci vd. (2014) yaptıkları çalışmada C++ programlama dili üzerinde OpenCv kütüphanesini kullanarak, hareketli ve sabit görüntüler üzerinde yüz görüntülerinin mutlu, üzgün ve doğal olarak üç kategoride sınıflandırılmasını gerçekleştirmişlerdir.

Java

Java, Sun Microsystems firması tarafından 1995 yılında çıkarılan bir programlama dilidir. Java dili web içerikleri, mobil programlar, masaüstü programları, kurumsal yazılımlar ve yapay zekâ gibi alanlarda kullanılır. Java ile kodlanan programların çalışması için özel bir platform olmasına gerek yoktur. Windows, MacOS, Linux, Unix gibi işletim sistemlerinin tamamında Java rahatlıkla kullanılabilir. Bu durum Java'nın tercih edilmesini arttıran bir etkidir.

Son zamanlarda karşımıza çıkan yapay zekâ tabanlı, aranan bilgiye en kısa yoldan ulaşılmasını sağlayan web sayfaları bulunmaktadır. Bu sayfaların alt yapısında web sayfalarını kategorize eden yazılımlar mevcuttur. Bu yazılım ile aramak istediğimiz veriye en uygun siteler filtrelenerek kişilere sunulmaktadır. Bu amaçla Uğur ve Kınacı (2006) Java platformu üzerinde JOONE kütüphanesini kullanarak, Yapay Sinir Ağları ile web sayfalarının sınıflandırılması üzerine çalışma yapmışlardır.

Kimlik doğrulama çalışmaları üzerine uzun zamandır çalışılmaktadır. Şifre kontrolü, kart denetimi, parmak izi okuma ve yüz tanıma sistemleri gibi alternatif birçok yöntem geliştirilmiştir. Şifrelerin unutulma ve kartların kaybedilme riski yüksek olduğundan parmak izi okuma ve yüz tanıma sistemleri gibi biyometrik tanıma yöntemlerinin önemi artmıştır (Kakıcı, 2008). Yüz tanıma sistemi, görüntü

veya video dosyası üzerinden yüzün özelliklerini çıkararak tanıma işlemlerinin gerçekleşmesini sağlayan yazılımlardır. Ötkün ve Karlık (2013) ORL veri tabanı üzerinde bulunan toplamda 40 kişinin 400 adet resim dosyalarından Java platformu üzerinde yapay sinir ağları yardımıyla kimlik tespiti çalışması gerçekleştirmişlerdir.

C#

C# Microsoft firması tarafından geliştirilmiş, öğrenim aşamasında ücretsiz olarak sunulan, fakat ticari olarak ücretli sunulan bir programlama dilidir. C ve C++ dillerinin etkileşiminde türetilen nesne yönelimli bir dildir. Masaüstü, sunucu, gömülü sistemler, sunucu işlemler, web programcılığı gibi yazılımlar geliştirilmektedir. Son zamanlarda gelişen teknoloji ile görüntü işleme, yapay zekâ gibi alanlarda da kullanılmaktadır (Özel, 2018).

Küçüksille ve Tokmak (2011) yaptıkları çalışmada, çalıştıkları birimde ders programının otomatik oluşturulması için yapay zekâ algoritmalarından yapay arı koloni sistemi ile C# programlama dilini birlikte kullanarak yazılım geliştirdiler. Dersi verecek öğretim elemanları, dersin hangi sınıfta işlenmesi gerektiği, öğretim elemanlarının günlük verecekleri ders sayısı gibi sınırlandırmaları tanımlayarak; oluşturdukları model üzerinden başarılı bir ders programı oluşturmayı başardılar.

Çevik ve Dandıl (2012) C# programlama dilini kullanarak her sayı için ayrı ayrı kaydedilen görüntüler üzerinden yapay sinir ağları ile sayı tahmini yapılabilecek bir yazılım geliştirmişlerdir. İstenilen verilerin seçilebileceği, gerekli kriterlerin belirlenebileceği, eğitim için gerekli ayarların yapılabileceği yazılımı geliştirerek, sayı tahmininde kullanılacak bir platform oluşturmuşlardır.

Yukarıda bahsedilen platformların veriler ve bilgiler dahilinde yapay zekâ alanında birçok problemin çözümünde kullanıldığı görülmektedir. Bankacılık, elektronik, eğitim, meteoroloji, güvenlik, internet teknolojileri, trafik kontrolü, sosyal medya üzerinde analiz gibi alanlarda gerçek problemleri çözmek amacıyla sistemler geliştirilmiştir.

Teknolojinin gelişmesiyle artan bilgisayar, mobil araçlar ve sosyal medyanın yüksek kullanımının sonucunda çok büyük miktarlarda veri oluşmaktadır. Bu

rakamlara istinaden elde edilen verilerden istenilen bilgilerin çıkarılması ihtiyacı doğmuştur. Artan verilerin incelenmesi konusunda ise insanların bilgi çıkarımı elbette yavaş kalacaktır. İstenilen bilgilerin çıkarılması konusunda bilgisayarlardan yardım alma ihtiyacı gün geçtikçe artmaktadır ve bu alandaki teknolojik gelişmeler devam etmektedir.



3. MATERYAL ve YÖNTEM

3.1. Çalışma Veri Seti

Araştırmada kullanılan SPECT görüntü verileri; 213 sağlıklı, 1363 parkinson hastası ve 134 swedd grubundan oluşmaktadır. Kullanılan SPECT veri seti Parkinson çalışmaları üzerine oluşturulmuş olan, Parkinson's Progression Markers Initiative (<https://www.ppmi-info.org>) web sayfasından güncel formuyla 4 Mart 2019 tarihinde alınmıştır.

Çizelge 1. Kullanılan veri seti içeriği

Veri Türü	Adet
Sağlıklı	213
Parkinson Hastası	1363
Swedd	134
Toplam	1710

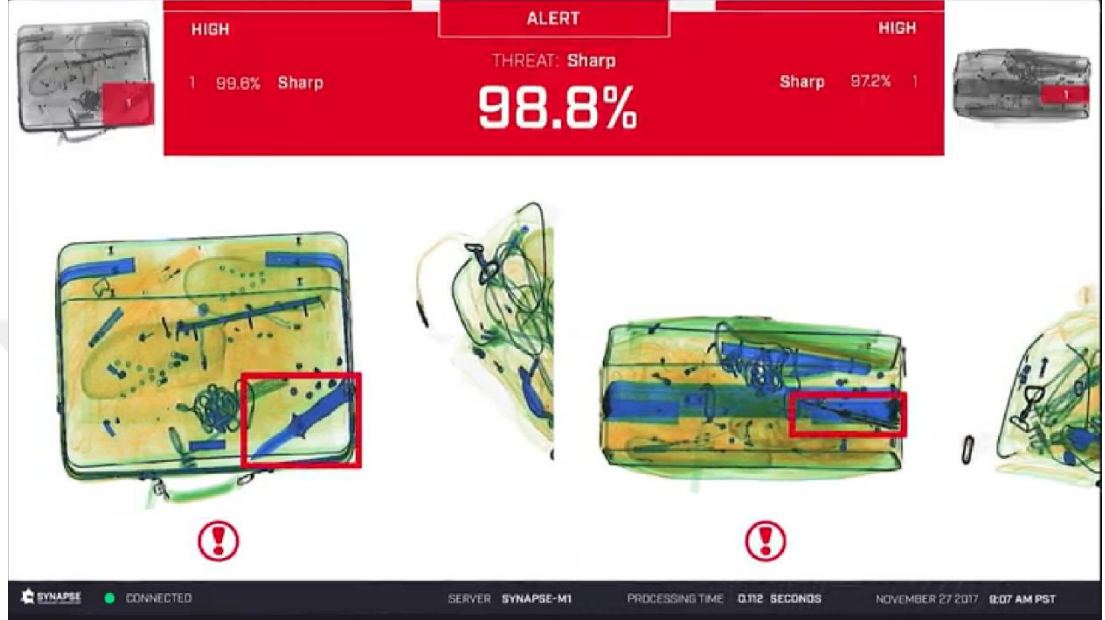
3.2. Geliştirme Ortamları ve Platformlar

3.2.1. CPU, GPU ve TPU

Yapay zekâ çalışmalarında büyük boyutlu veriler üzerinde işlem yapıldığından en önemli etkenlerden birisi uygulamanın yapıldığı cihazın hızıdır. Yapılan uygulamada kullanılan veri, görüntü üzerinden çok fazla matematiksel işlemler döngüsü çalıştığı için bunu karşılayacak hızda bir bilgisayar çalışmanın durumunu değiştirecektir. Ayrıca gerçek zamanlı uygulamalarda anlık verilerin işlenip gerekli çıktıların alınması önemli bir husustur.

Bugün pek çok binada, havalimanlarında veya alışveriş merkezlerinde x-ray cihazları kullanılmaktadır. Cihazların başında bekleyen bir personel tarafından cihazdan geçen malzemelerin içerisinde güvenlik riski oluşturacak parçaların bulunup bulunmadığı insan gözüyle kontrol edilmektedir. İnsanın olduğu her noktada hataların olabileceği düşünüldüğünde bu kontrol noktalarından gözden kaçabilecek parçaların geçmesi muhtemeldir. Bu güvenlik riskini ortadan kaldıracak, Synapse Technology isimli firma tarafından bir proje sunuldu. Bu projede x-ray

cihazlarındaki kontrol insanlardan alınıp, yapay zekâ teknolojisi ile geliştirilmiş sistemlere aktarılmaktadır. Bu sistemler aracılığıyla görüntü üzerinden riskli nesne tespiti yapılmakta ve görüntüdeki parçalar hakkında bilgi alınabilmektedir.



Şekil 6. Synapse Technology firmasına ait nesne tespit görüntüsü⁴

Yukarıda bahsedilen türde bir uygulamanın çalıştırılmasında cihazların çalışma hızı büyük öneme sahip olacaktır. Algılanan görüntü üzerinde hızlıca işlem yapıp dönüt alınmalıdır. Bu noktada bilgisayarlarda kullanılan hızı etkileyecek en önemli parça CPU (Central Processing Unit – Merkezi İşlem Birimi)’dur.

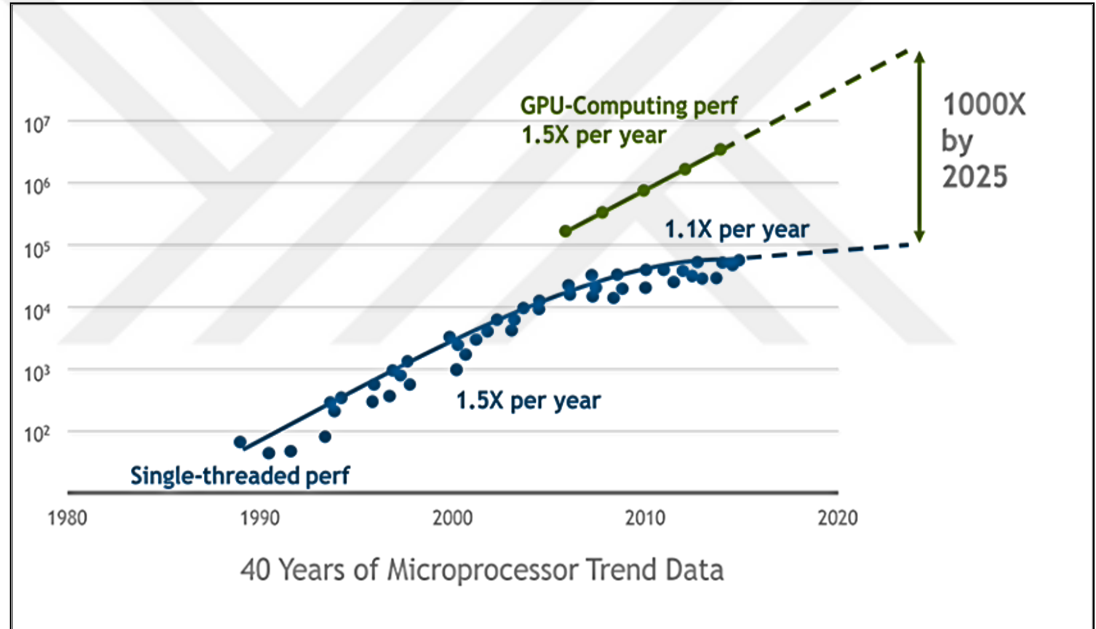
CPU bilgisayarın beyni olarak nitelendirilir. Türkçe karşılığı işlemci olarak bilinir. Bilgisayardaki temel aritmetik, mantıksal, kontrol işlemleri işlemciler üzerinde gerçekleştirilerek ilgili birime yönlendirilmesi sağlanır. CPU genel programlama problemleri için en uygun şekilde tasarlanmıştır.

Gelişen teknolojiler ile oyun sektörü de grafiksel olarak çok ileri seviyelere ulaştı. CPU grafiksel işlemler için yetersiz kaldı. Bu noktada ekran kartları üzerinde yerleşik olarak gelen grafiksel işlemler için kullanılacak işlemciler geliştirildi. Hızlı çözülmesi gereken işlemlerde bilgisayarın işlemcisi yerine ekran kartı üzerindeki

⁴ <https://www.synapsetechnology.com/product> adresinden alınmıştır.

işlemci kullanılarak sonuçlar daha hızlı elde edilmektedir. Bu teknolojinin adı GPU (Graphics Processing Unit – Grafik İşleme Birimi)’dur. GPU üzerinde bir seferde on binlerce işlem gerçekleştirilebilir. Genel olarak grafiklerin oluşturulmasını hızlandırmak amacıyla kullanılmaktadır. Ayrıca makine öğreniminde ve derin öğrenmede model eğitimi ve çıkarımların oluşturulmasında kullanılmaktadır.

Şekil 7’de yıl bazlı CPU ve GPU işlemcilerinin yıllara göre performansları ve performans artışları gösterilmiştir. Nvidia firmasının verilerine göre önümüzdeki yıllarda CPU performansları %10 civarında artarken, GPU performansı her yıl %50 oranında artacağı ön görülmektedir.

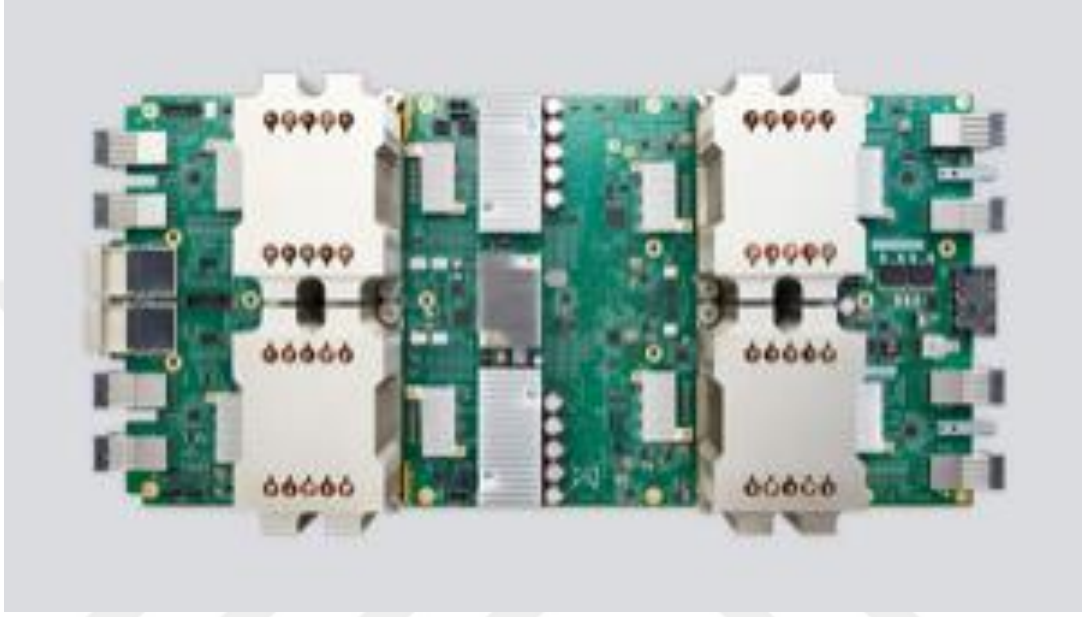


Şekil 7. CPU ve GPU performans seviyeleri⁵

Son zamanlarda kullanımı oldukça artan derin öğrenme uygulamaları için GPU’lar büyük kolaylık sağlıyordu. Fakat Google bu konuda GPU teknolojisini daha ileri boyutlara taşıyarak TPU (Tensor Processing Unit – Tensör İşleme Birimi) kavramını ortaya çıkardı. TPU, makine öğrenmesi ve derin öğrenme uygulamalarını eğitmek ve sonucunu almak için kullanılmaktadır.

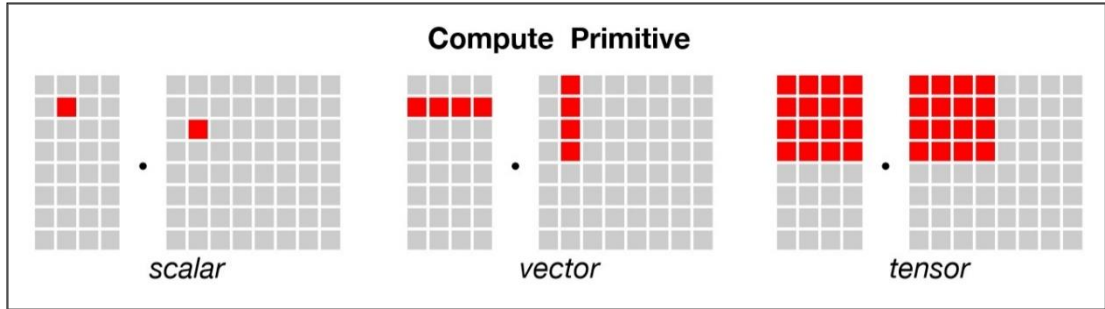
⁵ <https://www.nvidia.com/tr-tr/about-nvidia/ai-computing/> adresinden alınmıştır.

Bir makine öğrenmesi modelini eğitmek için, yüzlerce kez eğitim işleminin tekrar edilmesi doğru sonuca ulaşmak için çok önemlidir. Fakat bu işlemler hız ve maliyet gibi sorunları ortaya çıkarmaktadır. TPU işlemcileri eğitim işlemlerinin hızlı ve düşük maliyetli kalması için derin öğrenme uygulamaları geliştirenlere ücretsiz olarak sunulan bir teknolojidir.



Şekil 8. Google TPU v2 görüntüsü⁶

TPU'nun diğer işlemcilerden farkı, makine öğrenmesinin en önemli özelliği olan lineer cebir (matris) işlemlerinin paralel ve çok boyutlu olarak yürütülmesi için oluşturulmuş bir donanım olmasıdır (Kızrak, 2018). Aşağıdaki şekilde CPU, GPU ve TPU arasındaki hesaplama birim göstergeleri sunulmuştur.



Şekil 9. CPU, GPU ve TPU hesaplama birim göstergeleri⁷

⁶ <https://cloud.google.com/tpu/> adresinden alınmıştır.

⁷ <https://iq.opengenus.org/cpu-vs-gpu-vs-tpu/> adresinden alınmıştır.

Google, e-posta hesabı ile Colab üzerinden ücretsiz derin öğrenme uygulamaları geliştirmeye izin vermektedir. Açılan hesap ile sisteme derin öğrenme kodları yüklenip, hangi işlemci üzerinde (CPU, GPU ve TPU) çalıştırılacağını seçerek işlemler gerçekleştirilmektedir. Hazırladığımız tezde, çalışmalarımızın uygulamaları Nvidia GeForce GTX 1050 GPU işlemcisi olan yerel bir bilgisayar üzerinde gerçekleştirilmiştir.

3.2.2. CUDA Kütüphanesi

CUDA, Nvidia firması tarafından GPU ile yapılan işlemler üzerinde en iyi performansı almak için oluşturulmuş bir yapıdır. CUDA kütüphaneleri ve GPU işlemcileri gömülü sistemler, masaüstü ve bulut tabanlı platformlarda kullanılmaktadır. CUDA ile doğrusal cebir, görüntü işleme, derin öğrenme gibi alanlarda üstün performans alınması sağlanabilir. Aşağıda farklı çalışma alanları için oluşturulan CUDA kütüphaneleri gösterilmiştir. Çalışma alanlarının kendisine özgü işlem yapıları olduğundan her alan için farklı farklı kütüphaneler oluşturulmuştur.



Şekil 10. Farklı alanlara özgü CUDA kütüphaneleri⁸

⁸ <https://developer.nvidia.com/cuda-toolkit> adresinden alınmıştır.

Nvidia GPU'ları ile dünya üzerinde masaüstü, notebook, iş istasyonları ve süper bilgisayarlarda kullanılmak üzere araştırmacıların yapacakları işlemleri en üst seviyelere çıkarmak için kullanılmaktadır. CUDA kütüphaneleri sadece Nvidia firmasına ait olan GPU ürünleri grubu üzerinde kullanılmaktadır. Herhangi bir dil ile geliştirilen uygulamalar GPU'lar üzerinde hızlıca çalıştırılıp test edilebilmektedir.

Nvidia firmasının GPU ürünleri sunduğu 4 farklı grup bulunmaktadır. Bu farklı gruplar; masaüstü ürünleri için Quadro, sunucular için Tesla, laptop ürünleri için GeForce ve mobil ürünler için Tegra olarak karşımıza çıkmaktadır. Bu tez çalışmasında kullanılan GPU özelliğine sahip GTX 1050 isimli donanım birimi 768 CUDA çekirdeğine sahiptir. Standart hız (Mhz) seviyesi 1290, arttırılmış hız (Mhz) seviyesi ise 1392 olarak belirtilmiştir. Çekirdek sayısı CPU'lara göre oldukça fazladır. Derin öğrenme uygulamalarında veriler üzerinde çok fazla işlem yapılması gerektiği düşünüldüğünde, GPU'ların sahip olduğu çekirdekler üzerinde paralel olarak binlerce işlem yürütülebilmektedir.

3.2.3. Anaconda

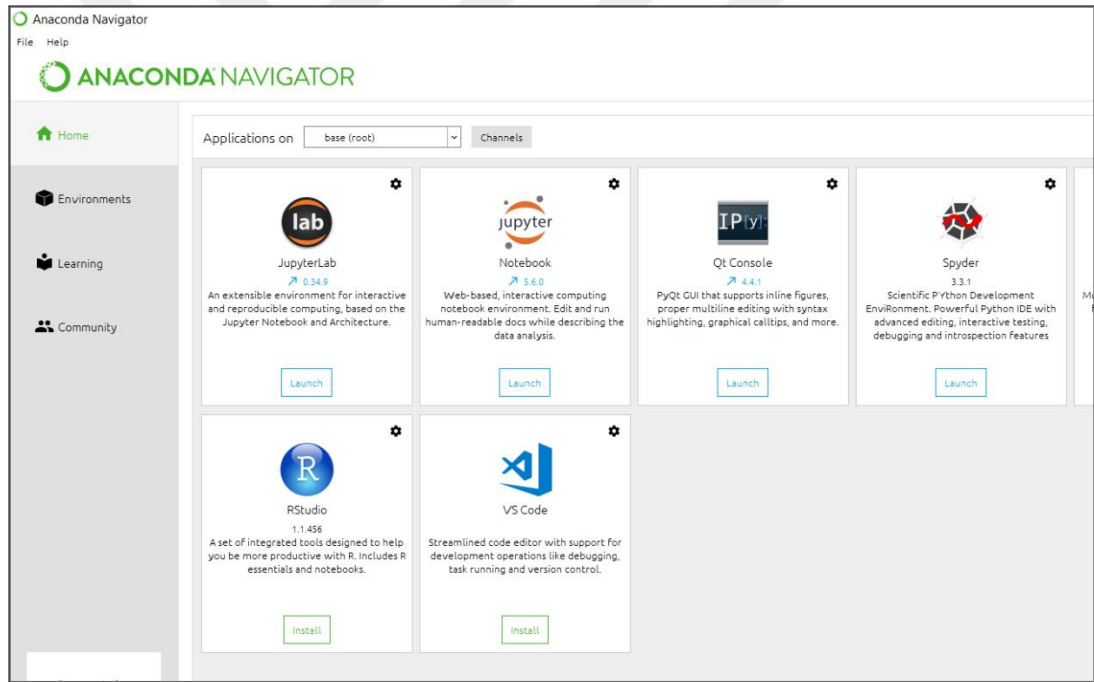
Araştırmacılar tek bir platform yerine birden fazla platform üzerinde çalışma yapabilirler. Belirli bir konuda anket sonuçları değerlendirilirken SPSS gibi platformlar üzerinde analizler gerçekleştirilir. Bunun yanında AMOS programı ile karmaşık değişken yapılarındaki ilişkilerin test edilmesi de gerçekleştirilir.

İlhan ve Çetin (2014) yaptıkları çalışmada LISREL ve AMOS programlarını kullanmışlardır. Bu programlar üzerinde yapısal eşitlik modeli uygulamasının sonuçlarını karşılaştırmışlardır. Çalışmada tek bir platform yerine birden fazla platformun kullanıldığı görülmektedir.

Bu tez çalışmasında kullanılan Python programlama dilinde kod yazımı için etkin bir geliştirme ortamı kullanılması, geliştirme sürecinde büyük fayda sağlayacaktır. Python programlama dili için gerekli yazılımlar yüklendiğinde, kara ekran olarak tabir edilen bir ekran üzerinde kodlar yazılmaktadır. Farklı bir geliştirme ortamı yüklendiğinde ise bu program ile kurulu olan Python programının ilişkilendirilmesi gerekmektedir.

Bu noktada Anaconda platformu araştırmacılar için kolaylıklar sunan bir platform olarak karşımıza çıkmaktadır. Anaconda, Python ve R dilleri ile makine

öğrenmesi, veri madenciliği gibi alanlarda kullanılmak üzere Windows, Linux ve Mac işletim sistemlerinde kullanılabilir açık kaynak olarak sunulan yazılım platformudur. Anaconda platformu yüklendiğinde hem Python programlama dili hem de geliştirme ortamı yüklenir ve aralarındaki bağlantı sağlanmış olur. Bunların yanında Anaconda ile Python için en çok kullanılan kütüphaneler de (Numpy, Pandas, Mathplot) hazır gelmektedir. Aşağıdaki ekranda görüldüğü gibi Anaconda platformunun açılış ekranında kullanılmaya hazır Python editörleri bulunmaktadır. Yüklenmek istenen seçenek altında bulunan “Install” düğmesi tıklanarak kurulumu yapılabilir. Yükleme yapılan platformlarda ise Spyder seçeneğinde olduğu gibi “Launch” düğmesi görüntülenmektedir. Launch düğmesi ile istenilen uygulama çalıştırılabilir.

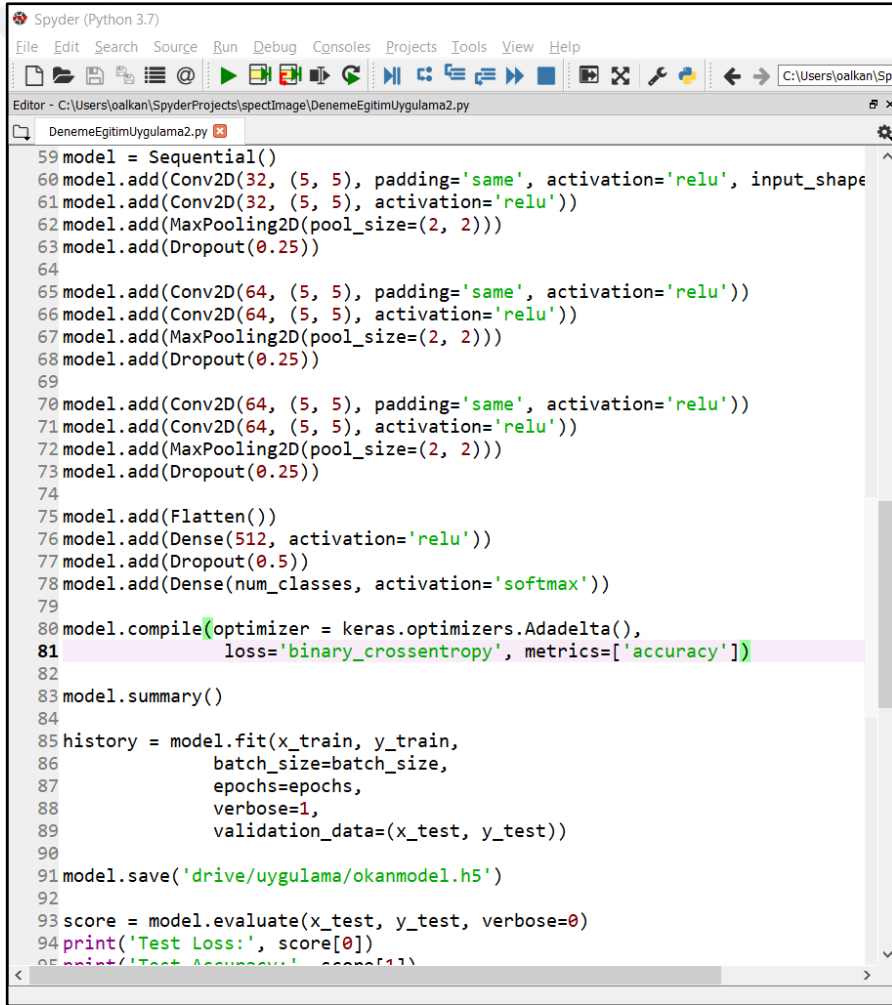


Şekil 11. Anaconda platformu açılış ekranı

3.2.4. Spyder

Programlama dillerinde kod yazmak için editörler kullanılır. Python dili için kullanılabilecek birçok editör mevcuttur. Bu tez çalışmasında Anaconda içinde yerleşik olarak gelen Spyder editörü kullanılmıştır. Öne çıkan bazı özellikleri nedeniyle bu editör seçilmiştir.

Öncelikle bu editör ücretsiz ve açık kaynak olarak dağıtılmaktadır. Kullanılabilecek birçok kütüphane dosyası editörde yüklü olarak gelir. Ara yüz üzerinde Python dilinin uzantısı olan ".py" dosyaları düzenlenebilir. Python için console kısmında kodların denemesi yapılabilir. Dosyaların olduğu klasör yapısının gösterildiği ekran üzerinde rahatlıkla dosyalar arası geçiş ve dosya yönetimi yapılabilir. Yazılan kodlara bağlı olarak kullanılan değişken yapıları ve aldığı değerler değişken ekranı aracılığıyla görüntülenir. Kod bloğunun ne için kullanıldığı, parametre olarak hangi değerleri alacağı gibi sorulara cevap bulunabilecek yardım menüsü bulunmaktadır. Tüm bu faydalı özelliklerin bir arada olması Spyder editörünün tercih edilmesine sebep olmuştur.



```
59 model = Sequential()
60 model.add(Conv2D(32, (5, 5), padding='same', activation='relu', input_shape=(28, 28, 1)))
61 model.add(Conv2D(32, (5, 5), activation='relu'))
62 model.add(MaxPooling2D(pool_size=(2, 2)))
63 model.add(Dropout(0.25))
64
65 model.add(Conv2D(64, (5, 5), padding='same', activation='relu'))
66 model.add(Conv2D(64, (5, 5), activation='relu'))
67 model.add(MaxPooling2D(pool_size=(2, 2)))
68 model.add(Dropout(0.25))
69
70 model.add(Conv2D(64, (5, 5), padding='same', activation='relu'))
71 model.add(Conv2D(64, (5, 5), activation='relu'))
72 model.add(MaxPooling2D(pool_size=(2, 2)))
73 model.add(Dropout(0.25))
74
75 model.add(Flatten())
76 model.add(Dense(512, activation='relu'))
77 model.add(Dropout(0.5))
78 model.add(Dense(num_classes, activation='softmax'))
79
80 model.compile(optimizer = keras.optimizers.Adadelta(),
81             loss='binary_crossentropy', metrics=['accuracy'])
82
83 model.summary()
84
85 history = model.fit(x_train, y_train,
86                   batch_size=batch_size,
87                   epochs=epochs,
88                   verbose=1,
89                   validation_data=(x_test, y_test))
90
91 model.save('drive/uygulama/okanmodel.h5')
92
93 score = model.evaluate(x_test, y_test, verbose=0)
94 print('Test Loss:', score[0])
95 print('Test Accuracy:', score[1])
```

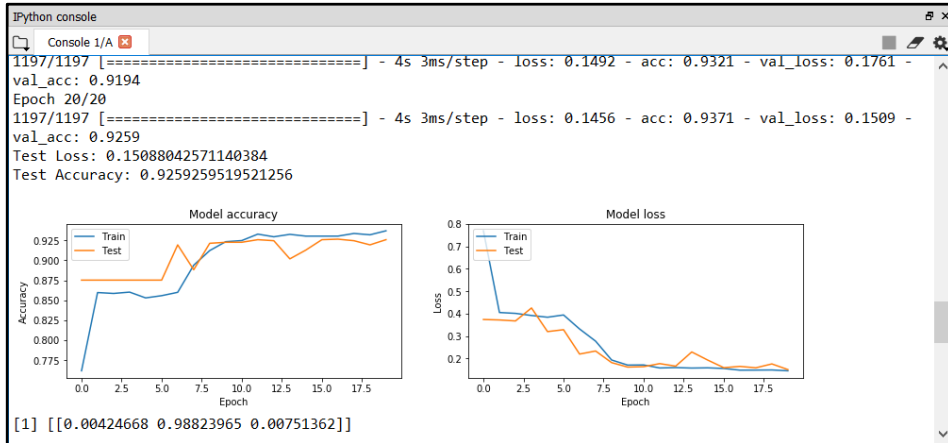
Şekil 12. Spyder kod editörü görüntüsü

Yukarıdaki şekilde görüldüğü üzere kodlar üzerindeki renklendirmeler, dosyasının hangi klasör içerisinde bulunduğu, kodları yazarken kod tamamlama özelliği Spyder programının editör kısmını ön plana çıkarmaktadır.

Variable explorer			
Name	Type	Size	Value
batch_size	int	1	64
device_name	str	1	/device:GPU:0
epochs	int	1	20
img_cols	int	1	109
img_rows	int	1	91
input_shape	tuple	3	(91, 109, 1)
num_classes	int	1	3
pre	float32	(1, 3)	[[0.00424668 0.98823965 0.00751362]]
preds	int64	(1,)	[1]
Variable explorer File explorer Help			

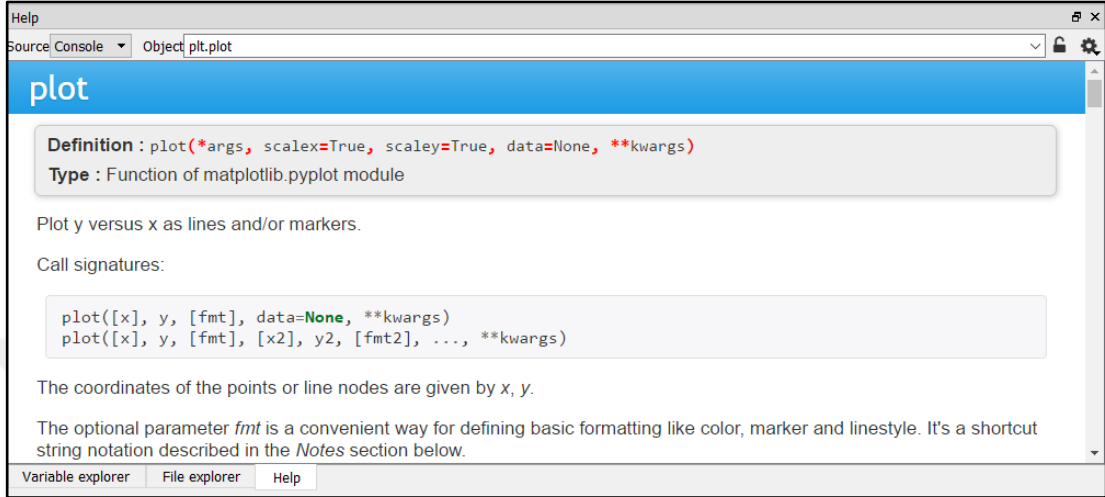
Şekil 13. Spyder değişken görüntüleme ekranı

Yukarıdaki şekilde Spyder ortamının değişken kontrol ekranı gösterilmektedir. Değişken isimleri, türleri, boyutları ve içerisindeki değerler rahatlıkla görüntülenebilmektedir. Diğer editörlerde bu verilere ulaşmak için konsol kısmında kod yazmak gerekmektedir. Fakat Spyder üzerinde bulunan değişken ekranıyla bu işlemlere gerek duyulmadan istenilen verilere ulaşılabilir.



Şekil 14. Spyder konsol ekranı

Yukarıdaki şekilde Spyder ortamının konsol ekranı gösterilmektedir. Yazılan kodlar çalıştırıldığında işlem sonuç değerleri, verilerin görselleştirilmiş hali gibi birçok yapı kolaylıkla görüntülenebilmektedir.



Şekil 15. Spyder yardım ekranı

Yardım penceresinden (Şekil 15) kullanılmak istenen komut (örn: “plot”) hakkında detaylı bilgiye ulaşılabilir. Bu pencere aracılığıyla komutların sözdizimine, alabileceği parametrelere ve kullanım örneklerine rahatlıkla ulaşılabilir.

3.2.5. Tensorflow

2015 tarihinde Google tarafından oluşturulan makine öğrenme uygulaması Tensorflow platformunu açık kaynak olarak servis edilir. Güçlü kütüphane desteği ile makine öğrenmesi uygulamaları geliştirmek için araştırmalara katkı sağlamaktadır. Yapay sinir ağı modelleri oluşturarak spor ayakkabı ve gömlek gibi pek çok resmin sınıflandırmasında kullanılmıştır. İçerisinde barındırdığı Keras kütüphanesi ile el yazısı rakamlar dahi sınıflandırılabilir (Elitez, 2015). Google Translate alt yapısında olduğu gibi Tensorflow ile diller arası çeviri uygulamaları da yapılabilir.

Tensorflow’un çalışma yapısında verileri saklama ortamı olarak “tensör” kelimesi karşımıza çıkmaktadır. Veri saklama ortamları boyutlarına göre değişiklik göstermektedir. Çizelge 2’de kullanılabilecek veri saklama ortamları gösterilmiştir.

Çizelge 2. Veri saklama ortamlarının boyut örnekleri

Boyut	Nesne	Örnek
0	Sayısal	3
1	Vektör	[1, 2, 3]
2	Matris	[[1, 2, 3], [4, 5, 6]]
>=3	Tensör	[[[1, 2, 3]], [[7, 8, 9]]]

Sayısal değerin boyutsal karşılığı 0 (Sıfır) olarak gösterilir ve içerisinde tek bir sayısal veri saklar. Vektör ise 3 boyutlu bir ortamda x, y ve z eksenlerine göre iz düşüm verilerinin saklandığı veri saklama ortamıdır. Birkaç vektör grubunun bir araya getirilmesi ile matris adını verdiğimiz yapılar ortaya çıkmaktadır. Bu yapılarında bir araya gelmesi ile Çizelge 2’de örneği verilen [2, 1, 3] boyutunda tensör veri saklama ortamı oluşmaktadır. Tensorflow yapısında ilk önce veriler tensör adı verilen veri yapılarına dönüştürülür. Ardından işlemler GPU veya TPU üzerinde hızlıca değerlendirilerek sonuca ulaşılmaya çalışılır.

Tensorflow 2015 yılında piyasaya sürülmesine karşın birçok büyük firma tarafından kullanılmaktadır. Airbnb dünya genelinde kullanılan ev kiralama şirketidir. Web sayfası üzerinden evler kiraya verilebilir ya da gidilecek yerden ev kiralama hizmeti alınabilir. Site içerisinde yapılan ev ilanları aramalarında fotoğrafların kategorilere ayrılması, her fotoğrafın ilandaki evin fotoğrafı olup olmadığını anlamak zor bir durumdur. Yao et al. (2018) yaptıkları çalışmada Tensorflow ile aynı mekânlara ait resimleri sınıflandırmayı başarmışlar ve yüklenen fotoğrafların aynı evin fotoğrafları olduğunu göstermişlerdir.

Coca-Cola şirketi adı altında satılan ürünlerin kapak altlarında bulunan ürün kodlarını müşteriler internet üzerinden göndererek ödül için çekiliş kampanyalarına katılmaktadırlar. Sistem müşterinin çektiği 14 karakterlik kodu içeren resim üzerinden sayıları ve harfleri tanıyarak bu bilgiyi anında kaydeder. Daha önce mevcut kullanılan OCR (Optik Karakter Tanıma) yazılımı ürünlerin üzerindeki kodları okumakta yüksek performans sağlayamamıştır. Bu yüzden şirket kodların basımı için yeniden yazı karakteri oluşturmuş ve milyonlarca ürün için tekrar basım yapmak zorunluluğu ortaya çıkmıştır. Fakat Tensorflow uygulaması ile mevcut kodların okunmasında doğruluk oranı %99 seviyelerine ulaştığı için şirket büyük bir

basım maliyetinden kurtulmuş ve bu iş için Tensorflow kütüphanesini kullanmaya başlamıştır (Gürsakar,2017; Brandt, 2017).



Şekil 16. Tensorflow kullanan firmalar⁹

3.2.6. Keras

Derin öğrenme, yapay zekâ alanında ön plana çıkan konulardan biri haline geldi. Alpha Go adlı programın Go oyununun şampiyonuna karşı elde ettiği galibiyet ile derin öğrenme ilgi görmeye başladı. 2015 yılında Tensorflow ile resim sınıflandırma alanında elde edilen üstün başarı ile derin öğrenme büyük önem kazandı (Koç, 2018).

Keras kütüphanesi, Tensorflow tabanında derin öğrenme modellerinin tanımlanmasını, aşamalarının belirlenmesini ve eğitim işlemlerini araştırmacılar için kolaylaştırmaktadır. Yapılması planlanan yapay öğrenme ortamı için gerekli tüm özelliklerin belirlenmesini sağlamaktadır.

⁹ <https://www.tensorflow.org/> adresinden alınmıştır.

Çizelge 3. Keras ile oluşturulmuş örnek model

<code>model = Sequential()</code>
<code>model.add(Conv2D(32, (5, 5), padding='same', activation='relu', input_shape=(31,30,1)))</code>
<code>model.add(MaxPooling2D(pool_size=(2, 2)))</code>
<code>model.add(Dropout(0.25))</code>
<code>model.add(Flatten())</code>
<code>model.add(Dense(3, activation='softmax'))</code>
<code>model.summary()</code>
<code>model.compile(optimizer = keras.optimizers.Adam(), loss = keras.losses.binary_crossentropy, metrics = ['accuracy'])</code>
<code>model.fit(x_train, y_train, batch_size=64, epochs=20, verbose=1, validation_data=(x_test, y_test))</code>
<code>model.save('uygulama/kerasmodel.h5')</code>
<code>model.evaluate(x_test, y_test, verbose=0)</code>
<code>model_test = load_model('uygulama/ kerasmodel.h5')</code>
<code>model_test.predict(test_data, batch_size=1)</code>

Çizelge 3, Keras kütüphanesini kullanarak model katmanlarını oluşturma, eğitim için gerekli yöntemi belirleme, eğitim işlemini uygulama, yapılan eğitimi bilgisayar ortamına kaydetme, eğitimin başarı oranını belirleme ve eğitilmiş olan model üzerinden bilgisayarın tahmin yapması aşamalarını bir kod bloğu olarak göstermektedir.

Keras kütüphanesi kullanıcı dostu bir yapı olup sıralı modeller oluşturulmasına izin verir. Sıralı model olarak isimlendirilmesindeki sebep; her

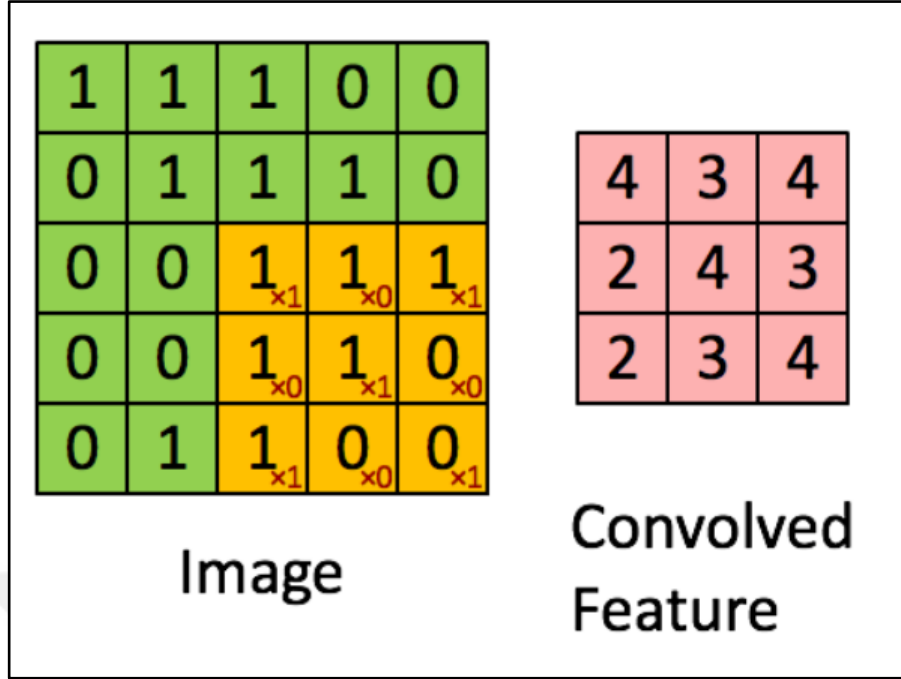
katmanın çıktısı bir sonraki katmanın girdisi olacak şekilde kullanıma sunulmasından kaynaklanmaktadır. Bu yapı sayesinde, işlenen veriden elde edilen sonuç sıradaki katmanda ilgili işlemden geçerek istenilen sonuçlara daha hızlı ulaşılacak hedeflenmektedir. Sıralı model oluşturma için “Sequential” kodu kullanılmaktadır. “Sequential” olarak oluşturulan modele sıralı katmanlar “add” komutu ile eklenmektedir.

Tek katmanlı veya çift katmanlı evrimsel/evrimsel sinir ağı modellere eklenebilmektedir. Conv1D ve Conv2D olarak veri setine uygun olan ağı katmanı seçilebilir. Görüntüler 2 boyutlu verilerden oluştuğu için görüntü işleme üzerine yapılacak çalışmalarda Conv2D katmanı seçilmektedir.

Conv2D katmanı için önemli argümanlar bulunur. Bu argümanlardan bir tanesi resim boyutunun verildiği “input_shape” değeridir. Bu değer resim genişliği, yüksekliği ve 3. boyut değeridir. Örneğin (30,30,1) boyutunda resimlerden oluşan veri setimiz var. Bu resimlerin genişliği ve yüksekliği 30 piksel değerindedir. 1 sayısı ise bu resim dosyasının gri olduğunu göstermektedir. Eğer resim dosyamız renkli ise bu değer 3 olacaktır. Renkli resimlerde RGB renk uzayı kullanılmaktadır. Bunlar kırmızı, yeşil ve mavi renk kodlarından oluşan katmalardır (Kılınç, 2018).

Katmandaki diğer bir değer filtre sayısıdır. Filtre sayısı görüntü hakkında alınmak istenen özellikler için görüntü boyutunu genişletmeye yarar. Kod örneğinde verilen (30,30,1) boyutlarındaki görüntü dosyası bu filtre sayılı katmandan geçtikten sonra (30,30,32) boyutlarında olacaktır. Bu katman sayısının artması resim dosyasından öğrenilecek olan değerlerin artmasını sağlayacaktır. Ve çıktı katmanından alınan değer başka bir katmana giriş verisi olarak iletilerek süreç devam ettirilecektir.

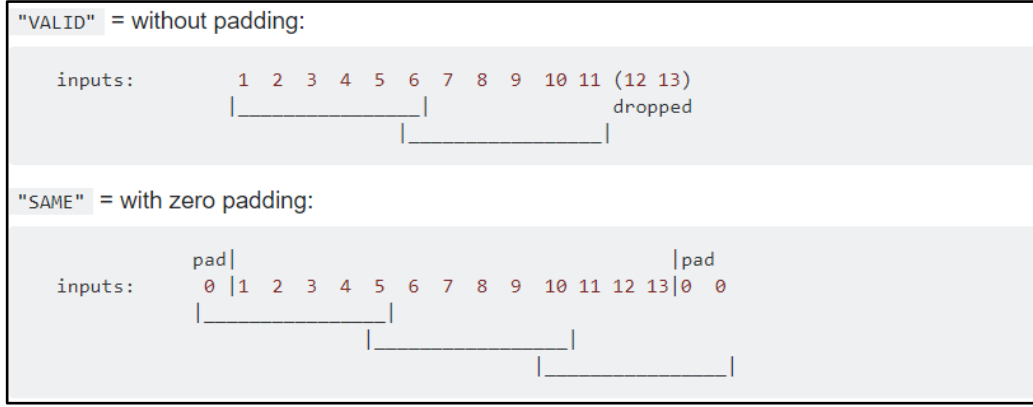
Diğer argüman kernel size (çekirdek boyutu) değeridir. Bu değer resim üzerine uygulanacak olan matris değeridir. Aşağıdaki resimde 5x5 boyutlarında veri setine 3x3 boyutlarında tanımlanan çekirdek boyutu değerinin uygulanma durumu gösterilmiştir. 5x5 boyutlarındaki veriler için sol köşeden başlayarak sağ alt köşeye kadar verilen boyutlar kullanılarak değer çıkarımı yapılır. Bu adımlar birçok kez uygulandığında resimler üzerinden aranılan değerlerin verilerine sayısal olarak ulaşmak mümkündür.



Şekil 17. 5x5 boyutundaki resim üzerine 3x3 çekirdek boyutu uygulama örneği¹⁰

Başka bir argüman değeri padding (doldurma) değeridir. Bu değer verilere uygulanan çekirdek boyut değerinin yetmediği eksik kaldığı durumlarda veri sütunlarının doldurulması için kullanılan gerekli bir argümandır. Katman oluşturulurken çekirdek boyut değerinin kaç adımda bir uygulanacağı belirtilebilir. Adım değerinin belirtilmediği durumlarda 1 er sütun ve satır kaydırılarak filtre uygulanır. Örnek incelenecek olursa 6 boyutlu bir filtre 5 adım değeri ile uygulanmaktadır. Eğer doldurma değeri resimdeki gibi “valid” olarak belirlenmiş ise sondan iki sütun devre dışı kalacaktır. Bu durumda bu sütunlardan alınabilecek bilgiler göz ardı edilmiş olur. Eğer herhangi bir kayıp yaşamadan tüm verilerin işlenmesi isteniyorsa doldurma değeri “same” olarak girilebilir. Bu durumda eksik kalan sütunlar veya satırlar için doldurma işlemi kullanılarak, olası kayıpların önüne geçilmeye çalışılır.

¹⁰ http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution adresinden alınmıştır.



Şekil 18. Sinir ağı katmanında doldurma uygulama örneği¹¹

Sinir ağı içerisine girdi olarak alınan değerler hesaplanır ve çıktı olarak bir değer oluşturulur. Oluşturulan bu değerlerin kullanılıp kullanılmayacağı durumu aktivasyon (activation) fonksiyonu ile belirlenir. Aşağıdaki resimde kullanılabilecek fonksiyonlar gösterilmiştir. Veriler için farklı fonksiyonların kullanıldığı yapılar vardır. Şekil 20’de gösterilen fonksiyonlardan step ve swish değerleri keras içerisinde kullanılamaz sunulmamıştır. Diğer fonksiyonlar keras içerisinde kullanılabilir değerlerdir (Karakuş, 2018). Bu fonksiyonlardan en çok kullanılanların özellikleri aşağıda sıralanmıştır.

Softmax: Belirlenen bir eksen etrafında normalizasyon işlemi yapılan fonksiyondur.

Sigmoid: 0 ve 1 arasında değerler üzerinden işlem yapılan fonksiyondur.

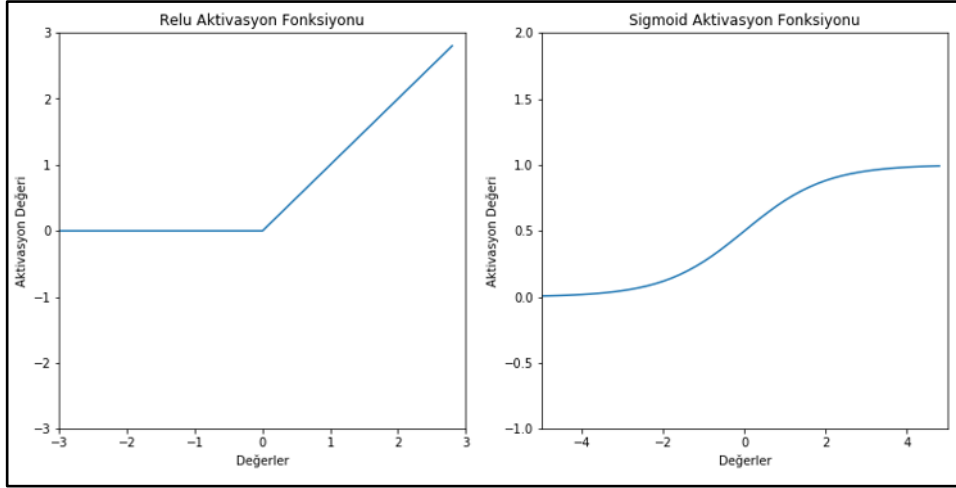
Tanh: -1 ve 1 arasında değerler üzerinden işlem yapılan fonksiyondur.

Relu: Negatif değerler için 0, diğer değerler için aynı değeri koruyan fonksiyondur.

Elu: Negatif değerler için değeri kullanıcılar tarafından belirlenebilen, yani negatif kısmın eğiminin belirlenebildiği fonksiyondur.

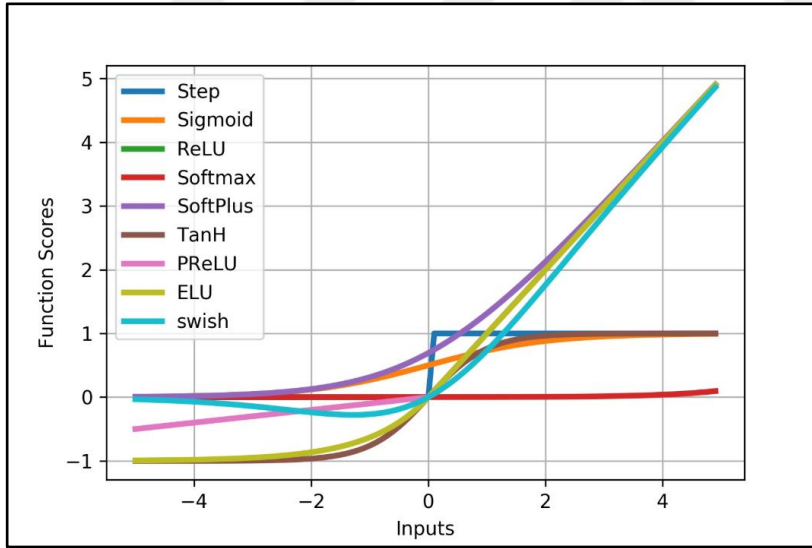
Selu: Veriler üzerinde kendiliğinden değerleri normalleştiren bir fonksiyondur.

¹¹ <https://stackoverflow.com/questions/37674306/what-is-the-difference-between-same-and-valid-padding-in-tf-nn-max-pool-of-t> adresinden alınmıştır.



Şekil 19. Relu ve Sigmoid fonksiyonlarının grafiksel gösterimi¹²

Aktivasyon fonksiyonları veri setlerine göre en iyi çıktıya ulaşmak için kullanılan sinir ağı katmanı argümanlarındandır.



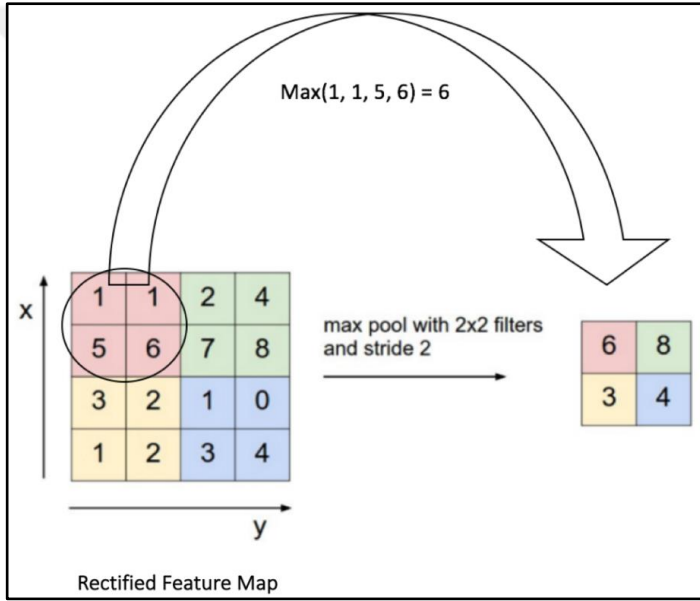
Şekil 20. Aktivasyon fonksiyonlarının grafiksel gösterimi¹³

Sinir ağlarının çıktıları üzerinde de pooling (havuzlama) işlemleri kullanılmaktadır. Havuzlama işlemi uygulanırken boyut değeri belirlenebilir.

¹² <http://www.veridefteri.com/2019/02/06/keras-ile-derin-ogrenmeye-giris/> adresinden alınmıştır.

¹³ https://github.com/betulays/activation_functions adresinden alınmıştır.

Örneğin (4,4) boyutundaki veri kümesine (2,2) boyutunda havuzlama yapıldığında, yeni veri boyutu (2,2) olarak düzenlenecektir. Havuzlama değeri için 3 farklı yöntem kullanılmaktadır. Bunlar maksimum havuzlama, minimum havuzlama ve ortalama havuzlama işlemleridir. Belirtilen filtre boyutları uygulandığında; maksimum için en büyük değer, minimum için en küçük değer, ortalama için değerlerin ortalaması alınarak veri kümesi belirlenmektedir. Aşağıdaki resimde bu durumu gösteren resim örneği bulunmaktadır. Resimde maksimum havuzlama kullanıldığı için filtre boyutu içinde olan verilerin en yüksek olanı alınarak veri oluşturulmaktadır. En çok kullanılan havuzlama yöntemi maksimum havuzlama yöntemidir (Çarkacı, 2018).

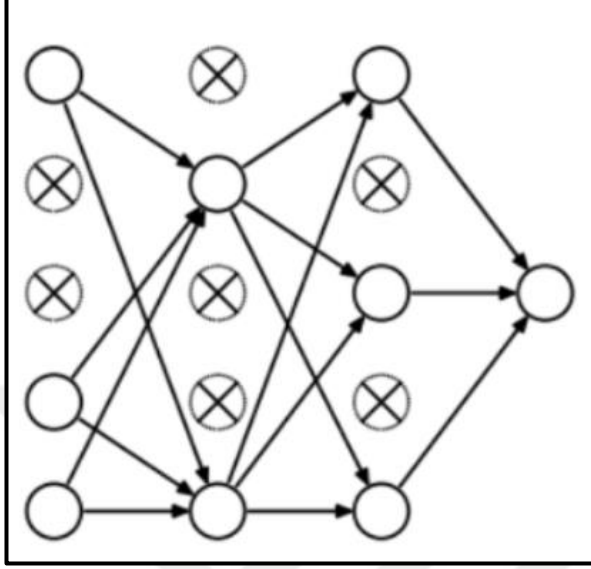


Şekil 21. Maximum Pooling (Havuzlama) işlemi çalışma mantığı¹⁴

Derin öğrenme uygulamaları yaparken oluşabilecek problemlerden bazıları ezberleme durumudur. Veri setinin az olduğu durumlarda daha çok karşılaşılan ezberleme işleminin önüne geçmek için Dropout (Bırakma) katmanı eklenmelidir (Budhiraja, 2016). Bırakma katmanı oluşturulan sinir ağında yapılan işlem bazı bağlantıların kaldırılmasından ibarettir. 0 ile 1 arasında değer girilebilir. Eğitim yapılırken girilen değer kadar rastgele bağlantı görmezden gelinir. Bu işlem

¹⁴ <https://blog.algorithmia.com/convolutional-neural-nets-in-pytorch/> adresinden alınmıştır.

uygulandığında eğitim işlemlerinde daha başarılı sonuçlar elde edilmiştir (Jain, 2018).



Şekil 22. Dropout uygulanmış sinir ağı örneği¹⁵

Yapılan uygulamalarda kullanılan veri setleri çok boyutlu yapılardan oluşmaktadır. Eğer bu çok boyutlu katmanlar tek satırlı bir dizi şekline getirilmek istenirse Flatten (Düzleştirme) katmanı kullanılabilir (Chen et al., 2014). Örneğin oluşturulan modelde katmandan alınan veri boyutları (7, 10, 128) değerlerinde olduğu düşünüldüğünde, düzleştirme katmanı eklendiğinde verilerin yeni boyutu $7 \times 10 \times 128 = 8960$ olacaktır.

Sinir ağında her katmanın girdi ve çıktı olarak iki grupta veri değerleri olur. Katmana gelen veri sayısı işlemten sonra çıktı olarak verilir. Çalışmalarda çıktı olarak istenilen bir değer varsa bu “Dense” argümanı ile belirlenebilir. Genelde çalışmaların son katmanlarında kullanılır. Örneğin 10 adet rakamdan oluşan bir veri setimiz var. Bu veri seti eğitildikten sonra tahmin edilmesi istenen değer için 10 adet sonuç çıktısı almak istenebilir. Bu durum aşağıdaki Çizelge 4’de gösterilmiştir. Tahmin edilecek değer 3 olduğu için 3 değerine karşılık gelen index numarasındaki

¹⁵ <https://www.analyticsvidhya.com/blog/2018/04/fundamentals-deep-learning-regularization-techniques/> adresinden alınmıştır.

kayıt 1, diğer kayıtlar 0 değerini göstermektedir. Bu verilere bakılarak tahmin edilen değer hangisi olduğu anlaşılmaktadır.

Çizelge 4. Tahmini yapılan değer çıktı verileri

Tahmin Edilecek Değer	Çıktı Katmanından Alınan Değer
3	(0, 0, 0, 1, 0, 0, 0, 0, 0, 0)
4	(0, 0, 0, 0, 1, 0, 0, 0, 0, 0)

Oluşturulan model katmanlarını ekranda göstermek için ise summary fonksiyonu kullanılır. Görülen katman yapılarına ek olarak veri setimizdeki çıktı parametreleri hakkında bilgiler de yer almaktadır. Model oluşturmak için gerekli olan ayarlamalardan sonra öğrenme sürecinin yapılandırılması gerekmektedir. Bu yapılandırma işlemi “Compile” fonksiyonu ile belirlenir. İçerisine 3 argüman yer almaktadır.

Birinci argüman değeri optimizasyon yöntemidir. Bu değer eğitimin prosedürünü belirlemek için kullanılmaktadır. Çalışmada ulaşılabilecek sonuç için istenilen en uygun değer bulunmasında optimizasyon yöntemleri kullanılmaktadır (Karaboğa, 2017). Kullanılabilecek optimizasyon yöntemleri aşağıdaki gibidir:

- SGD
- RMSprop
- Adagrad
- Adadelta
- Adam
- Adamax

İkinci argüman loss fonksiyonudur. Bu değer tahmini yapılacak olan gerçek değer ile tahmin edilen değer arasındaki ortaya çıkan hata oranıdır. Eğitim esnasında bu değer düşük olması eğitimin başarısının arttığını göstermektedir. Kullanılabilecek loss fonksiyonları aşağıdaki gibidir:

- binary_crossentropy
- categorical_crossentropy
- mean_squared_error

Üçüncü argüman ise metric'tir. Metric değeri yapılan eğitimin performansını belirlemek amacıyla kullanılır. Eğitimin başarısı verilen doğru tahminler olarak gösterilebileceği gibi, tahminler üzerinden yapılan hataların az olması da gösterilebilir. Kullanılabilecek metric değerleri aşağıdaki gibidir:

- accuracy: doğruluk
- sensitivity: duyarlılık
- specificity: özgüllük
- mae: ortalama mutlak hata
- msc: ortalamaların karesi

Compile işleminin ardından eğitim işleminin yapılması gerekmektedir. Bu işlem için fit fonksiyonu kullanılmaktadır. Fit fonksiyonu içerisinde eğitim işleminin yapılabilmesi için belirlenmesi gereken argümanlar vardır. İlk iki argüman veri setinden eğitim için kullanılan değişkenlerdir. Verilen kod örneğinde x_train ve y_train değerleri bu eğitim değişkenlerini göstermektedir. x_train dosyasında elimizdeki veri grupları, y_train içerisinde ise bu verilen karşılığındaki etiket değerleri bulunmaktadır. Örneğin bir görüntü dosyasının sınıflandırması yapılmak istendiğinde; görüntü dosyasının içerdiği veriler x_train dosyasında, görüntülerin karşılığı olan etiket verileri ise y_train dosyasında saklanmalıdır.

“Batch_size” değeri eğitilecek olan verilerden küçük gruplar almak için kullanılır. Tüm veriler bir anda eğitime katıldığında çok büyük veri işleme kapasitesi gerektirdiği için batch_size değeri ile veri setinden istenilen kadar parça alınıp eğitim bu şekilde gerçekleştirilir. Örneğin 1000 adet örnekten oluşan bir yapı için batch_size değeri 25 olarak belirlenirse veri seti ($1000/25=40$) 40'ar örnek parçalar halinde işlenerek eğitilecektir.

“Epoch” değeri ise eğitim işleminin kaç kez tekrar edeceğini ifade eder. Tüm veri seti baştan sona kaç kez eğitim işleminden geçirilmek isteniyorsa bu değer üzerinden değişiklik yapılabilir. Örnek verilecek olursa bu değer bir öğrencinin bir ders kitabını kaç kez okuyacağını (epoch değerini) göstermektedir.

“Verbose” argümanı ise eğitimin yapıldığı esnada kullanıcıya eğitimin ne kadarının tamamlandığı, her epoch sonucundaki doğruluk oranı gibi değerlerin gösterilip gösterilmeyeceğini belirlemek için kullanılır. 3 değer alır. Bunlar 0, 1 ve 2

değerleridir. 0 değeri verilir ise eğitim işlemi herhangi bir şekilde kullanıcıya gösterilmeden tamamlanır. 1 değeri verilir ise her adım ilerleme çubuğu şeklinde kullanıcıya gösterilir. 2 değeri verilir ise her epoch sonucunda sadece verilen başarı metriği sonucunu gösterir.

“validation_data” argümanı ise eğitim işlemi yapılırken, eğitimin doğruluğunu test etmek için kullanılacak olan değişkenleri belirlemek için kullanılır. Daha önce eğitim için kullanılacak olan değişkenler x_train ve y_train olarak belirlenmiştir. Yapılan eğitimi test etmek için eğitimde kullanılan örneklerden farklı olarak oluşturulan x_test ve y_test değişkenleri verilmelidir. Veri seti genel uygulama olarak iki gruba ayrılır. İlk grup veri setinin %70 oranını kapsayan train bölümü, ikinci kısım olarak ise %30 luk kısmı sağlayan test bölümüdür.

Yapılan derin öğrenme uygulamasını farklı dosyalarda daha sonradan da kullanmak için model HDF5 formatında “.h5” uzantısı ile kayıt altında alınabilir. Kaydetmek işlemi için “Save” fonksiyonu kullanılmaktadır. Fonksiyon içerisine dosyanın kaydedileceği dizin ve dosya adı verilmelidir. Kaydedilen model dosyasında modelin katmanları, eğitim yapılandırma ayarları gibi tüm bilgiler yer almaktadır. Kaydedilmiş bir model kullanılmak için “load_model” fonksiyonu ile çağrılabilir.

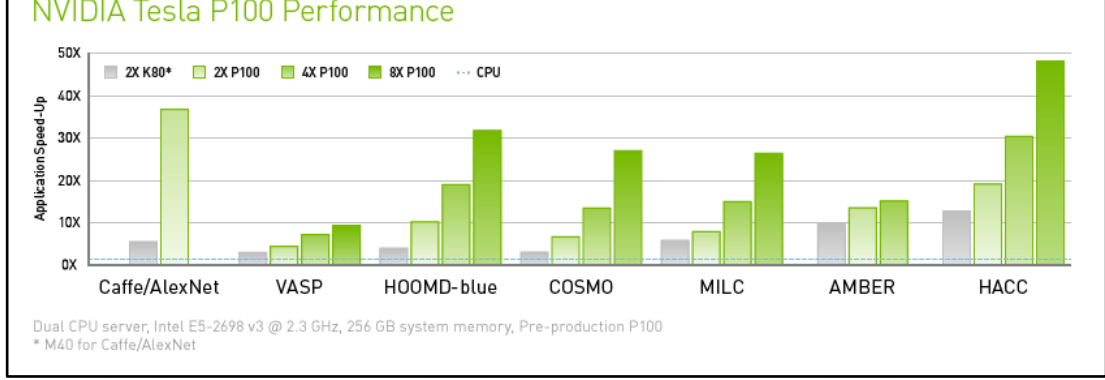
Eğitim işlemi bitirildikten sonra “evaluate” fonksiyonu ile eğitim işleminin “loss”, “accuracy”, “sensitivity” ve “specificity” değerleri hesaplanabilmektedir. Doğruluk hesaplaması için ayrılan x_test ve y_test değişkenleri verilerek hesaplama yapılabilir.

Tüm işlemler bitirildikten sonra tek bir dosya için tahmin yapılmak istendiğinde “predict” fonksiyonu kullanılabilir. Fonksiyon içerisinde tahmini yapılmak istenen veri dosyası yüklenerek işlem yapılabilir.

3.2.7. Derin Öğrenme Ortamı Kurulumu

Çalışmada kullanılan yazılımların kurulumlarında dikkat edilmesi gereken ayarlar bulunmaktadır. Derin öğrenme uygulaması yapılırken CPU ve GPU için ayrı ayrı kurulum ayarları mevcuttur. Kullanılan cihazda GPU desteği varken

kurulumların eksik yapılması halinde CPU üzerinden çalışmalar yürütülür. Bu da zaman açısından büyük problemlere yol açabilir.



Şekil 23. Nvidia Tesla P100 performans grafiği¹⁶

Şekilde de gösterildiği gibi 8x P100 CPU hızı ile 2x K80 GPU üzerinde zaman açısından büyük farklılıklar oluşmaktadır. Bu fark yapılacak eğitimin hızında ve uygulanmasında zamansal olarak büyük kayıplara neden olmaktadır.

Nvidia GPU desteği verdiği ekran kartlarının listesini <https://developer.nvidia.com/cuda-gpus> adresinden araştırmacılara sunmaktadır. Hangi ekran kartının GPU platformunu desteklediğini öğrenmek için ilgili sayfa incelenebilir.



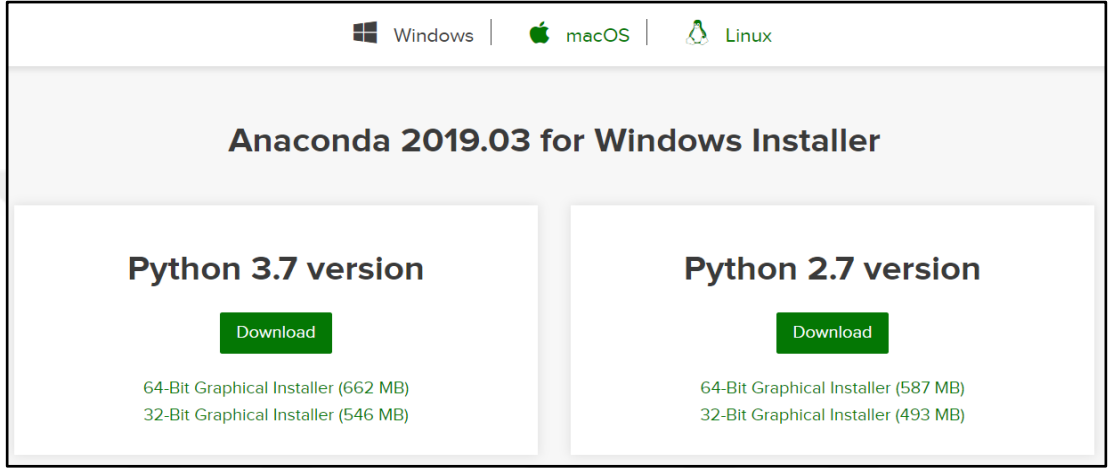
CUDA-Enabled GeForce Products

GeForce Desktop Products		GeForce Notebook Products	
GPU	Compute Capability	GPU	Compute Capability
NVIDIA TITAN RTX	7.5	Geforce RTX 2080	7.5
Geforce RTX 2080 Ti	7.5	Geforce RTX 2070	7.5
Geforce RTX 2080	7.5	Geforce RTX 2060	7.5
Geforce RTX 2070	7.5	GeForce GTX 1080	6.1
Geforce RTX 2060	7.5	GeForce GTX 1070	6.1
NVIDIA TITAN V	7.0	GeForce GTX 1060	6.1

Şekil 24. GPU desteği verilen GeForce ürünlerinin listesi

¹⁶ <https://www.nvidia.com/en-gb/data-center/tesla-p100/> adresinden alınmıştır.

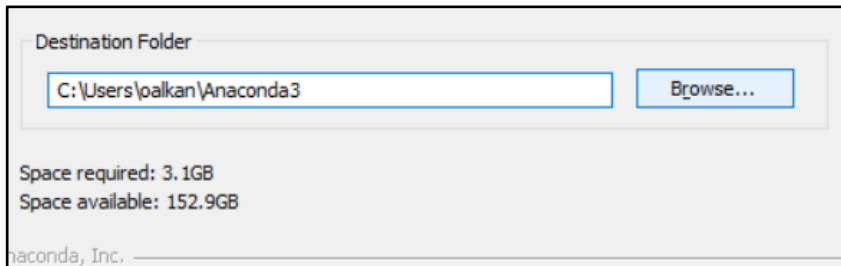
Kurulumu yapılacak ilk platform, içerisinde hem Python programlama dilini hem de kullanılan editörü barındığı için seçilen Anaconda Navigator programıdır. Kurulum dosyasını indirmek için ilk önce <https://www.anaconda.com/distribution/> adresi açılır. Açılan sayfa Windows, Mac ve Linux için ayrı ayrı kurulum dosyaları mevcuttur. Programlama dili için Python 2.7 ve Python 3.7 paketleri mevcuttur. Python 2 versiyonu en son 2.7 sürümünde bırakıldı. Bunun için güncel ve desteği devam eden 3.7 versiyonunun kurulumu seçilmelidir.



Şekil 25. Anaconda yazılımını indirme ekranı

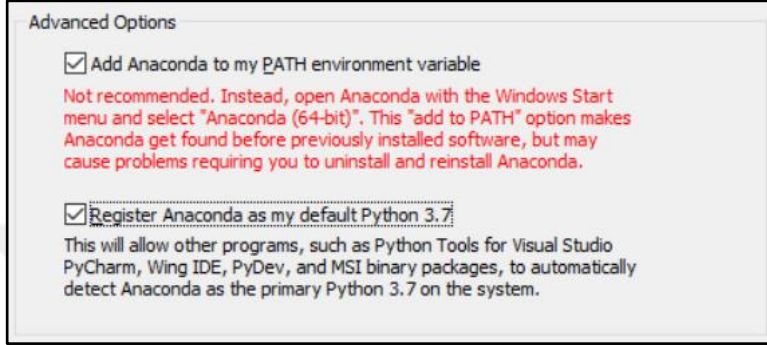
Şekilde gösterilen bölümden ilk önce kurulum yapılacak platform olan Windows bölümü seçilir. Daha sonra alt bölümde yer alan Python 3.7 version altında “64-Bit Graphical Installer” seçeneği tıklanarak indirme işlemi başlatılır.

Anaconda kurulum dosyası indirildikten sonra dosya çalıştırılarak kurulum başlatılır. Kurulum esnasında dikkat edilmesi gerek iki unsur bulunmaktadır. Birinci unsur kurulumun yapılacağı adres yolunun belirlenmesidir. Python kurulumu yine bu klasör içerisine yapılacağından bu adresin unutulmaması gerekmektedir.



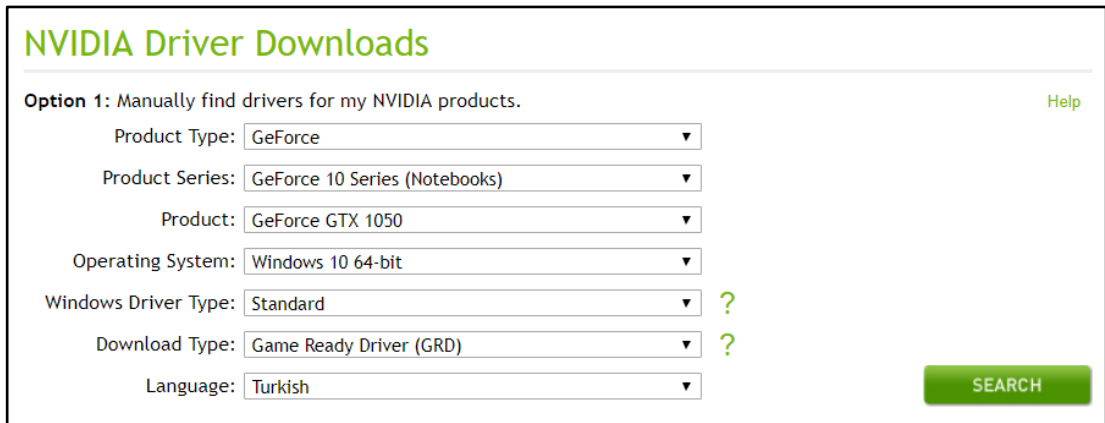
Şekil 26. Anaconda yüklenen klasörün yol ayarı

İkinci unsur ise kurulum adımında bulunan “Add Anaconda to my PATH environment variable” seçeneğinin seçilmesidir. Bu ayar Python bilgisini sistem değişkenlerine eklemeyi sağlar. “Register Anaconda as my default Python 3.7” ayarı ile Python dilini kullanan diğer programlar için kurulumu yapılan klasörü varsayılan olarak ayarlamaktadır. Bu ayar aracılığıyla Python yazılımına erişim kolaylaşmaktadır.



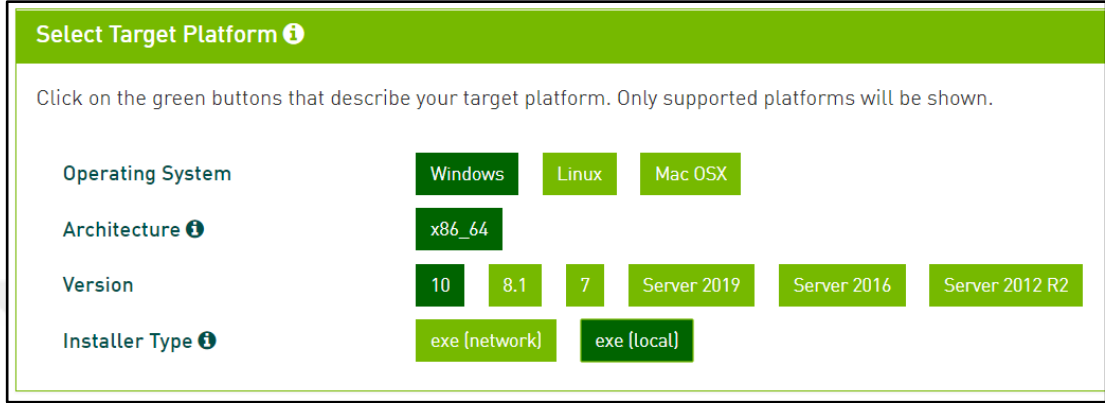
Şekil 27. Python sistem değişkenlerine ekleme ayarı

Anaconda kurulumu tamamlandıktan sonra GPU kurulumu için gerekli programların yüklenmesi gerekmektedir. İlk kurulum programı olarak GPU destekli ekran kartının sürücüsünün güncellenmesi gerekmektedir. Tarayıcı üzerinden <https://www.nvidia.com/Download/index.aspx?lang=en-us> adresine bağlanarak sistem özellikleri seçilir ve “Search” butonuna tıklanarak sürücü araması yapılır. Güncel bir sürücü versiyonu mevcut ise indirme ve yükleme işlemleri gerçekleştirilir.



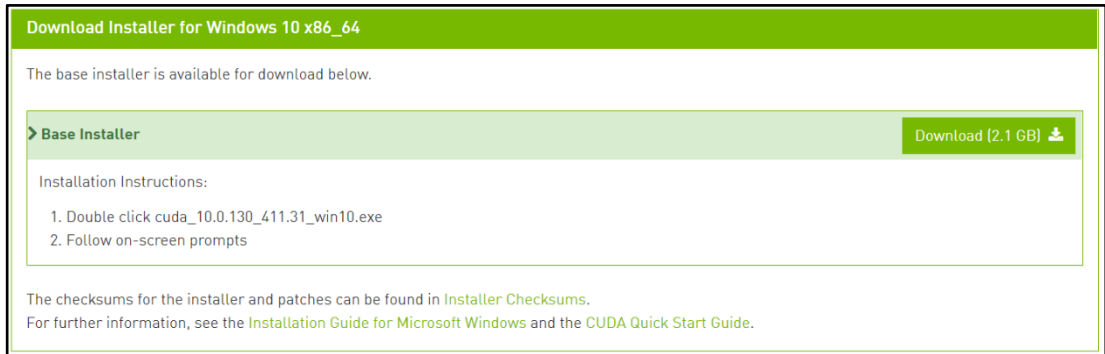
Şekil 28. Nvidia ekran kartı sürücü yükleme ekranı

Sürücü dosyası yüklendikten sonra yüklenmesi gereken yazılım CUDA Toolkit aracıdır. Bu aracı indirmek için <https://developer.nvidia.com/cuda-downloads> adresi açılır. Açılan sayfadan aşağıdaki resimde gösterildiği gibi sistem bilgileri seçilir.



Şekil 29. CUDA Toolkit sistem seçim ekranı

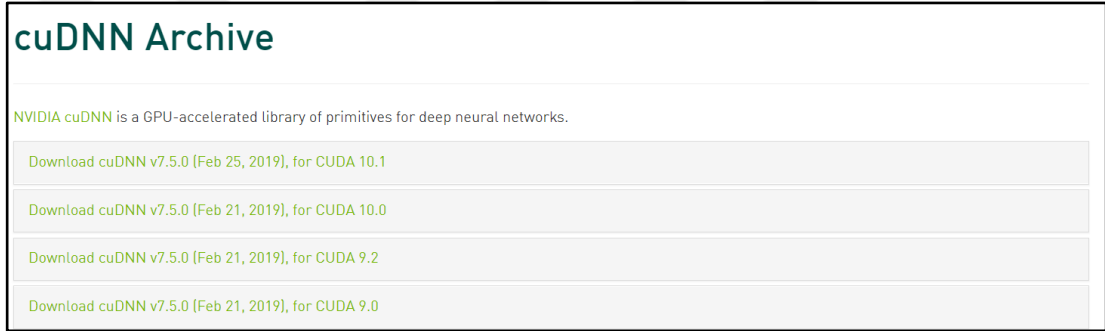
Sistem ayarları seçildikten sonra aşağıdaki ekranda CUDA Toolkit yazılımının en son sürümü olan 10.1 versiyonuna ait indirme bağlantısı bulunmaktadır. <https://www.tensorflow.org/install/gpu> adresinde Tensorflow desteği bulunan sürümün 10.0 olduğu gösterilmektedir. Bu neden ekranda bulunan “Legacy Release” seçeneğine tıklanarak açılan eski sürümler listesinden 10.0 seçeneği seçilir. Açılan ekranda bulunan indirme bağlantısı tıklanarak CUDA Toolkit 10.0 yazılımının indirilmesi başlatılır.



Şekil 30. CUDA Toolkit indirme bağlantısı

İndirilen CUDA Toolkit yüklendikten sonra kurulum klasörü varsayılan olarak “C:\Program Files\NVIDIA GPU Computing Toolkit” şeklinde olacaktır. Bu klasör sistem değişkenlerine ekleneceği için adres kontrol edilmelidir.

CUDA Toolkit yazılımının ardından cuDNN Archive yazılımının indirilmesi gerekmektedir. Yazılımı indirmek için <https://developer.nvidia.com/rdp/cudnn-archive> adresine gidilir. Burada dikkat edilmesi gereken nokta yüklenen CUDA Toolkit versiyonuna uygun olan cuDNN yazılımının indirilmesidir. Uyumsuz sürümler yüklendiğinde çalıştırma aşamasında problemlerle karşılaşılabılır. Bunun için indirilen sürüm olan CUDA Toolkit 10.0 versiyonu için uyumlu olan “Download cuDNN v7.5.1 (April 22, 2019), for CUDA 10.0” seçeneği seçilmelidir.



Şekil 31. cuDNN Archive yazılımı indirme ekranı

İlgili bölüm seçildikten sonra Windows 10 işletim sistemi kullanıldığı için “cuDNN Library for Windows 10” seçeneği seçilir. İndirme bağlantısına tıklandığında üye girişi istemektedir. Sistem üzerinden hesap oluşturularak indirme işleminin bu şekilde yapılması gerekmektedir.

cuDNN yazılımı indirme dosyası içerisinde “cuda” isimli bir klasör bulunan arşiv dosyasıdır. İndirme gerçekleştikten sonra “cuda” isimli klasör kolay erişilebilecek bir dizine çıkarılmalıdır. Bunun için “C://tools” isimli bir dizin oluşturularak indirilen dosya bu dizine çıkarılır. Tensorflow yükleme işlemi “C://tools//bin” dizininde bulunan “cudnn64_7.dll” dosyası olmadan yüklenmeyecektir. Bunun için “cudnn64_7.dll” dosyasını CUDA Toolkit aracının yüklendiği “C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v10.0\bin” dizinine kopyalanması gerekmektedir.

Bu işlemlerden sonra sistemin doğru bir şekilde çalışması için sistem değişkenlerinde bazı güncellemeler yapılması gerekmektedir. Bunun için “Bilgisayar” simgesine sağ tıklayıp Özellikler seçeneği açılır. Sol bölümden “Gelişmiş sistem ayarları” seçeneği seçilir. Gelen ekrandan “Ortam Değişkenleri” butonu tıklanır. Gelen pencerenin alt bölümünde sistem değişkenleri kısmında “Path” seçeneği seçilip “Düzenle” butonuna tıklanır. Gelen pencere üzerinden aşağıdaki her dizin için ayrı ayrı “Yeni” butonuna tıklayarak ekleme işlemleri yapılır.

C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v10.0\bin

C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v10.0\libnvvp

C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v10.0\extras\CUPTI\libx64

C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v10.0\include

C:\tools\cuda\bin

Yükleme işlemlerinin ardından sistemin etkin olması için sistemin yeniden başlatılması gerekmektedir. Yeniden başlatma işleminden sonra Windows komut istemi (cmd) programı çalıştırılır. Gelen ekran üzerinden aşağıdaki kodlar sırasıyla çalıştırılır.

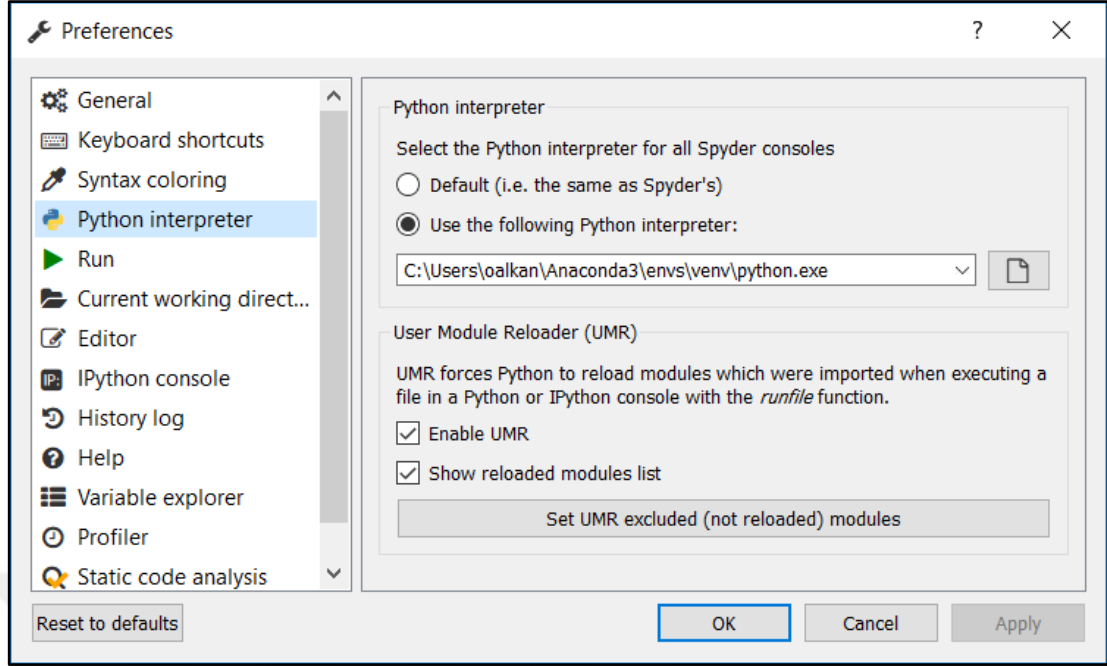
- conda create -n venv pip python=3.6
 - Python 3.7 versiyonunda kod yapılarından bulunan eksiklikler sebebiyle 3.6 versiyonu tercih edildi. Bu kod ile belirlenen Python versiyonu için paket kurulumlarını sistemden izole etmek için sanal ortam oluşturulur. Sanal ortam ismi olarak “venv” belirlenmiştir.
- conda activate venv
 - Oluşturulan sanal ortam üzerinde kurulumlar yapmak için sanal ortam ismi aktif edilir. Daha sonra herhangi bir paket kurulumuna ihtiyaç duyulduğunda, komut isteme (cmd) ekranı açılarak bu kod çalıştırılır ve istenilen paket kurulumları gerçekleştirilebilir.
- pip install tensorflow-gpu
 - Bu kod ile GPU üzerinde tensorflow’un çalışması için gerekli yazılımlar internet üzerinden indirilerek yüklenir.

- `pip install spyder-kernels==0.*`
 - Python'un Spyder üzerinde doğru bir şekilde çalışması için Spyder-kernel kütüphanesinin yüklenmesi gerekmektedir.
- `python -c "import tensorflow as tf; tf.enable_eager_execution(); print(tf.reduce_sum(tf.random_normal([1000, 1000])))"`
 - Bu kod Tensorflow GPU kurulumunun doğru bir şekilde yüklendiğini anlamak için çalıştırılabilir.

```
(venv) C:\Users\oalkan>python -c "import tensorflow as tf; tf.enable_eager_execution(); print(tf.reduce_sum(tf.random_normal([1000, 1000])))"
2019-04-24 23:09:17.317474: I tensorflow/core/platform/cpu_feature_guard.cc:141] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX2
2019-04-24 23:09:18.409777: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1433] Found device 0 with properties:
name: GeForce GTX 1050 major: 6 minor: 1 memoryClockRate(GHz): 1.493
pciBusID: 0000:01:00:0
totalMemory: 4.00GiB freeMemory: 3.30GiB
2019-04-24 23:09:18.423658: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1512] Adding visible gpu devices: 0
2019-04-24 23:09:19.337611: I tensorflow/core/common_runtime/gpu/gpu_device.cc:984] Device interconnect StreamExecutor with strength 1 edge matrix:
2019-04-24 23:09:19.351336: I tensorflow/core/common_runtime/gpu/gpu_device.cc:990] 0
2019-04-24 23:09:19.359031: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1003] 0: N
2019-04-24 23:09:19.374232: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1115] Created TensorFlow device (/job:localhost/replica:0/task:0/device:GPU:0 with 3011 MB memory) -> physical GPU (device: 0, name: GeForce GTX 1050, pci bus id: 0000:01:00:0, compute capability: 6.1)
tf.Tensor(-1179.9304, shape=(), dtype=float32)
```

Şekil 32. Tensorflow GPU doğru yüklendiğini gösteren ekran

Kurulumlar tamamlandıktan sonra Anaconda Navigator programı çalıştırılır. Gelen ekrandan Python geliştirme ortamı Spyder altında bulunan “Launch” seçeneği tıklanarak platform çalıştırılır. Anaconda ile gelen Python programını Spyder ile ilişkilendirmek için “interpreter” ayarlanması gerekmektedir. Bunun için Spyder ekranında Tools menüsünün altında bulunan “Preferences” seçeneği tıklanır. Gelen pencereden sol bölümde bulunan “Python interpreter” seçeneği seçilir. Aşağıdaki ekranda görüldüğü gibi “Use the following Python interpreter” kutucuğu seçilir. Bu bölüme daha önce yüklenen “venv” isimli sanal ortamdaki Python dosyasının seçilmesi gerekmektedir. Adres çubuğunun sağ tarafında bulunan “Browse” butonuna tıklayarak “C:\Users\oalkan\Anaconda3\envs\venv” dizini altındaki “Python.exe” dosyası seçilir ve Apply butonu ile işlem tamamlanır.



Şekil 33. Spyder python interpreter ayar ekranı

Python interpreter ayarını yaptıktan Tensorflow GPU çalıştırmak için gerekli olan tüm programların kurulumu ve ayarlamaları işlemi tamamlanmış oldu.

3.2.8. Kütüphanelerin Kurulumu

Bu tez çalışmasında yapılmak istenen işlemler için gerekli olan kütüphaneler teker teker yüklenmiştir. Aşağıda bu çalışmada kullanılan kütüphaneler ve hangi amaç için kullanıldıkları listelenmiştir.

- pip install keras
 - Derin öğrenme işlemleri için tensorflow altında oluşturulmuş olan keras kütüphanesidir.
- pip install matplotlib
 - Ekranda herhangi bir görsel değer veya grafik benzeri yapılar oluşturmak için kullanılan kütüphanedir.
- pip install dicom
 - Bu tez çalışmasında kullanılan SPECT görüntülerinin dosya türü dicom'dur. Bu dosyalar üzerinde işlem yapmak için "dicom" kütüphanesi kullanılır.

- pip install os
 - Python ile klasör izinleri üzerinde işlemler yapmak için kullanılan kütüphanedir.
- pip install imageio
 - Dicom görüntülerinden alınan verilerin resim dosyası olarak kaydedilmesi için imageio kütüphanesi kullanılmıştır.
- pip install scikit-image
 - Dosyalar üzerinde dönüşüm işlemlerinin yapılması için kullanılan kütüphanedir.
- pip install scikit-learn
 - Makine öğrenmesi uygulamaları geliştirmek için kullanılan bir kütüphanedir.
- pip install numpy
 - Anaconda programıyla beraber “Numpy” kütüphanesi yüklü olarak gelmektedir. Numpy matematiksel hesaplamaların hızlı bir şekilde yapılmasını sağlar. Temelinde dizi yapıları kullanılır. Python üzerindeki dizi yapıları yerine Numpy içerisindeki dizi yapılarını kullanmak daha kullanışlıdır. Bu nedenle verilerin saklanması ve yönetilmesi için Numpy kütüphanesi kullanılmıştır.

3.3. Veri Ön İşleme

3.3.1. Veri Seti Kurulumu

PPMI (Parkinson’s Progression Markers Initiative – Parkinson İlerleme Belirtileri Girişimi), Parkinson hastalığının ilerleme belirtilerini belirlemek amacıyla oluşturulmuş bir platformdur. Bu platformda parkinson araştırma vakfı, kalıtsal hastalıklar ve nöroloji alanlarında uzman bilim insanları bir arada çalışmaktadır. Bu alanda yapılan çalışmalar için kurumsal firmalar ve özel şahıslar tarafından fon desteği sağlanmaktadır. İtalya, Norveç, Almanya, Avusturalya ve Amerika üzerinde birçok klinik ile atanan koordinatörler vasıtasıyla iletişim kurulmaktadır. Parkinson hastalığının teşhis ve tedavisinde birçok uzmanlık alanı iç içe çalışması gerekmektedir. Bu nedenle PPMI platformu içinde Uyku, Genetik, Biyoloji, Görüntüleme ve Bilişsel alanlarda çalışma grupları oluşturulmuştur.

Bu tez çalışmasında kullanılan veriler Parkinson Progression Markers Initiative (PPMI) veri tabanından (www.ppmi-info.org/data) elde edilmiştir. Hastalık ve verilerle ilgili güncel bilgilere www.ppmi-info.org adresinden ulaşılabilir.

Verileri kullanmak isteyen araştırmacılar PPMI platformundan yapacakları araştırma çalışması için izin almak zorundadır. Bu izin bir sözleşme çerçevesinde gerçekleşir. Sözleşme onaylandıktan sonra hesap açılmış olur.

Sisteme giriş yapıldıktan sonra menü üzerinden “Download” bölümü altındaki “Image Collection” sayfasına gidilir. Gelen sayfa üzerinden “Advance Search” bölümü açılır. Bu sayfa üzerinden istenilen görüntü verilerinin arama işlemleri gerçekleştirilebilir.

Arama ekranı üzerinde birçok kriter bulunmaktadır. Yaş, cinsiyet, ağırlık, araştırma grubu, görüntüleme yöntemi, görüntü tipi gibi değerlerden istenilen ölçütler girilerek istenilen verilere kolayca ulaşılabilir. Tez çalışmasında kullanılan kriterler Çizelge 5’de verilmiştir.

Çizelge 5. Veri seti için belirlenen kriter ve değerleri

Kriter	Belirlenen Değer
Araştırma Grubu	Control
	Pd
	Swedd
Görüntü Yöntemi	Spect
Görüntü Türü	Pre-Processed

Search Advanced Search (beta) Advanced Search Results Data Collections

Search Options

☐ Exclude Previous Downloads

SEARCH SECTION

☒ Project/Phase

☐ Subject

☐ Subject Specific Information

☐ Assessments

☐ Study/Visit

☒ Image

☐ Imaging Protocol

☐ Image Status

☐ Image Processing

IMAGE TYPES

☐ Original

☒ Pre-processed

☐ Post-processed

Search Criteria

PROJECT/PHASE Specify selection criteria using the checkboxes on the left. Wild cards (*) are permitted in fields marked with a star below. For example, "rest*" returns results that begin with "rest."

PROJECTS ☒ PPMI

SUBJECT

Subject ID * Separate multiple Subject ID's by commas

Age (years) Equals

Sex Both

Weight (kgs) Equals

Research Group ☒ Control ☐ GenCohort PD ☐ GenReg PD ☐ GenReg Unaff ☐ Phantom ☐ Volunteer ☒ PD ☒ SWEDD ☐ Prodromal ☐ GenCohort Unaff

IMAGE

Image Description *

Image ID Separate multiple Image ID's by commas (eg. I123,I456,... or 123,456,...)

Modality ☐ CT ☐ DTI ☐ MRI ☐ PET ☒ SPECT ☐ fMRI ☐ OR ☐ AND Subject has at least one

IMAGING PROTOCOL

(SPECT) **Manufacturer** ☐ 0000000 ☐ ADAC ☐ GE MEDICAL SYSTEMS ☐ GE MEDICAL SYSTEMS, NUCLEAR

Şekil 34. Veri seti için belirlenen kriterlerin seçimi

Çalışmada kullanılan görüntüleme yöntemi Spect'tir. Görüntü türü olarak pre-processed seçeneği zorunlu olarak seçilmiştir. Çünkü Spect görüntü yönteminin orijinal olarak veri kaydı bulunmamaktadır. Spect görüntüleri için varsayılan tür olarak pre-processed değeri verilmektedir. Araştırma grubu olarak Control, Pd ve Swedd değerleri bulunmaktadır. Kriterler belirlendikten sonra "Search" butonu ile bu kriterlere uyan hasta kayıtları listelenmektedir.

Search (beta) Advanced Search Results Data Collections

Displaying Results 1-20 of 212 0 images selected

Subject ID ▼	Sex	STUDY	Age	IMAGE	Description
3000	F	Select ☐	69.1	Select ☐	VIEW Reconstructed DaTSCAN ⇌ Raw Data
3004	M	Select ☐	59.8	Select ☐	VIEW Reconstructed DaTSCAN ⇌ Raw DataR2S
3008	F	Select ☐	82.2	Select ☐	VIEW Reconstructed DaTSCAN ⇌ Raw DataR2
3009	F	Select ☐	84.0	Select ☐	VIEW Reconstructed DaTSCAN ⇌ Raw DataR1S
3011	M	Select ☐	31.9	Select ☐	VIEW Reconstructed DaTSCAN ⇌ I123ADD
3013	F	Select ☐	79.3	Select ☐	VIEW Reconstructed DaTSCAN ⇌ PPMI BRAIN B
3016	M	Select ☐	57.4	Select ☐	VIEW Reconstructed DaTSCAN ⇌ I-123 SPECT
3029	M	Select ☐	66.3	Select ☐	VIEW Reconstructed DaTSCAN ⇌ I-123 SPECT
3053	M	Select ☐	68.5	Select ☐	VIEW Reconstructed DaTSCAN ⇌ SPECT I123
3055	F	Select ☐	68.8	Select ☐	VIEW Reconstructed DaTSCAN ⇌ SPECT I123

Şekil 35. PPMI veri seti arama sonuçları

Gelen liste üzerinden birçok farklı cihazdan alınmış görüntüler mevcuttur. Bu görüntü çeşitliliği çalışmanın gerçek zamanlı uygulamalarda hata oranını en aza indirmektedir (Zhang, 2017). Yapılan derin öğrenme çalışmalarında binlerce veriden oluşan veri setleri kullanılmaktadır. Araştırmacılar MNIST web sayfası üzerinden yayınladıkları veri tabanında toplam 70 bin adet el yazısını bir araya getirdiler. Veri

setinde bulunan resimlerden 60 bin adeti eğitim ve 10 bin adeti test işlemleri için ayrılmıştır. Bu tür çalışmalarda veri seti az olduğunda veri arttırma yöntemlerine başvurulmaktadır. Veri sayısı az olan durumlarda veri setinde çeşitli bozma, dönüşüm, renk değişiklikleri ve kesit alma işlemleri ile başarı arttırma çalışmaları veri arttırma işlemleri olarak adlandırılmaktadır (Kızrak, 2018). Veri arttırma işlemi doğruluğu yüksek eğitim yapmak ve öğrenme işleminde ezberleme olmamasını sağlamak adına yapılmaktadır. Veri arttırma kullanmadan yapılan çalışmalarda eğitim sırasında yüksek doğruluk oranları elde edilirken, gerçek hayatta kullanıldığında doğruluk oranı düşmektedir (Wang ve Perez, 2017).

Aşağıdaki resimde veri arttırma yöntemi ile çoğaltılmış resim örnekleri bulunmaktadır. Veri setinde sadece ön taraftan alınmış kedi resimleri ile eğitim işlemi gerçekleştirilebilir. Bu durumda gerçek hayatta yandan veya yakından kedi resmi algılandığında tespit işlemi doğru yapılamayacaktır.

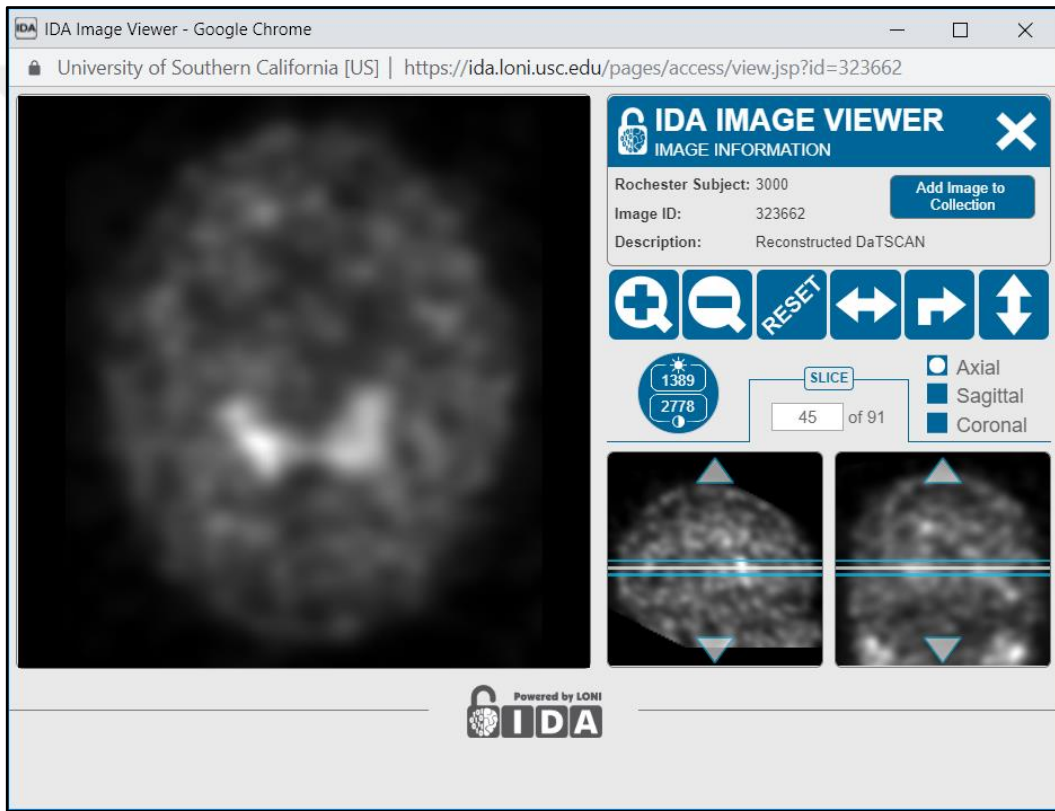
Bu çalışmada ise Spect görüntüleri üzerinde aşağıda yer alan resimdeki gibi yakınlaştırma, belli bir kesit alma gibi yöntemler uygulanmamıştır. Çünkü görüntü üzerindeki her bir nokta/piksel hastalık teşhisi için gerekli değerleri içerebilmektedir. Fakat bu noktada tek bir cihazdan alınan görüntülerden eğitim yapıldığı düşünüldüğünde gerçek hayatta başarısız sonuçlara ulaşılabilir. Bu nedenle sınıflandırmada başarı oranının gerçek hayatta olumsuz sonuçlar doğurmaması için farklı cihazlardan alınan resim dosyaları kullanılmıştır.



Şekil 36. Veri arttırma yöntemi uygulanmış resim örneği¹⁷

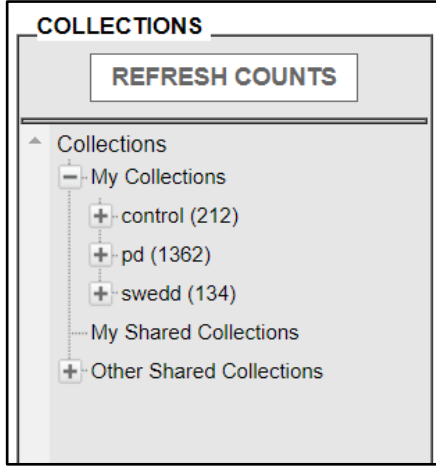
¹⁷ <https://github.com/xkumiyu/numpy-data-augmentation> adresinden alınmıştır.

Arama sonuçlarında her hasta için tanımlanmış ID değeri, cinsiyet, yaş ve farklı üretici firmaların kayıtları listelenmektedir. Listelenen kayıtlardan görüntüler incelenebilir. Bunun için kayıt sırasında bulunan “View” butonuna tıklayarak görüntüler incelenebilir. View butonuna tıklandıktan sonra aşağıdaki gibi bir ekran gelmektedir. Gelen ekran üzerinden görüntünün önden veya yandan görüntüleri incelenebilir. Yakınlaştırma ve uzaklaştırma ayarları ile detaylı incelemeler yapılabilir. Ayrıca 91 kesitten oluşan Spect görüntünün her bir kesiti “Slice” bölümünden ayarlama yapılarak görüntülenebilir.



Şekil 37. Kontrol grubundan bir bireyin görüntülendiği ekran

Listelenen kayıtlar seçildikten sonra “CSV Download” butonu ile “.csv” dosya formatında bilgiler indirilebilir. “Add To Collection” butonu ile listelenen kayıtları daha sonra tekrar arama yapmamak adına giriş yapan kullanıcının koleksiyon hesabına aktarılabilir.



Şekil 38. PPMI web sayfasında kullanıcı koleksiyon ekranı

Koleksiyon üzerinden herhangi bölüm seçildikten sonra daha önce arama yapılmış kayıtlar listelenebilir. Gelen listeden “All” butonu ile tüm kayıtlar seçilip “Click Download” butonuna tıklandığında zip dosyası olarak indirme ekranı açılacaktır. Bu ekran üzerinden seçilen kayıtların “dcm” uzantılı DICOM türündeki görüntü dosyaları kullanıcı bilgisayarına indirilebilir. Bu tez çalışmasında kullanılan veri seti 03.04.2019 tarihinde <https://www.ppmi-info.org> adresinden indirilmiştir.

3.3.2. Veri Seti İnceleme

Veri seti indirilirken kullanılan araştırma grubu kriterleri 3 farklı gruptan oluşmaktadır. Control grubu görüntüleme tekniği ile görüntü kayıtları alınan sağlıklı bireylerdir. Bu kayıtların hiçbirine Parkinson hastalığı teşhisi konulmamıştır. Pd olarak adlandırılan kayıtlar ise Parkinson hastalığı teşhisi konulan bireylerden oluşmaktadır. Swedd olarak adlandırılan kayıtlarda ise parkinson hastalığı için belirteç olan dopamin eksikliği kanıtı olmayan taramalar yer almaktadır.

Veri seti yaş grubu olarak 25-50, 50-75 ve 75-100 yaş aralığında erkek ve kadın cinsiyetine sahip kişilerin kayıtların oluşmaktadır. İlgili kayıtların veri setindeki dağılımları Çizelge 6 ve 7’de gösterilmiştir. Kayıtlarda Erkek grubundan 1111 adet, kadın grubundan ise 599 adet veri yer almaktadır. Yaş gruplarına bakıldığında ise en çok veriye sahip olan kayıtlar 1357 adet veri ile 50-75 yaş grubundaki kişilere ait olduğu görülmektedir. Bunu 186 adet veri ile 25-50 yaş

aralığı izlemektedir. En az veri ise 75-100 yaş aralığındaki bireylerin kayıtlarından oluşmaktadır.

Çizelge 6. Veri seti araştırma grubu cinsiyet dağılımları

Araştırma Grubu	Erkek	Kadın	Veri Sayısı
Control	140	73	213
Pd	888	475	1363
Swedd	83	51	134
Toplam	1111	599	1710

Çizelge 7. Veri seti araştırma grubu yaş aralığı dağılımları

Araştırma Grubu	25-50	50-75	75-100
Control	33	160	20
Pd	131	1100	132
Swedd	22	97	15
Toplam	186	1357	167

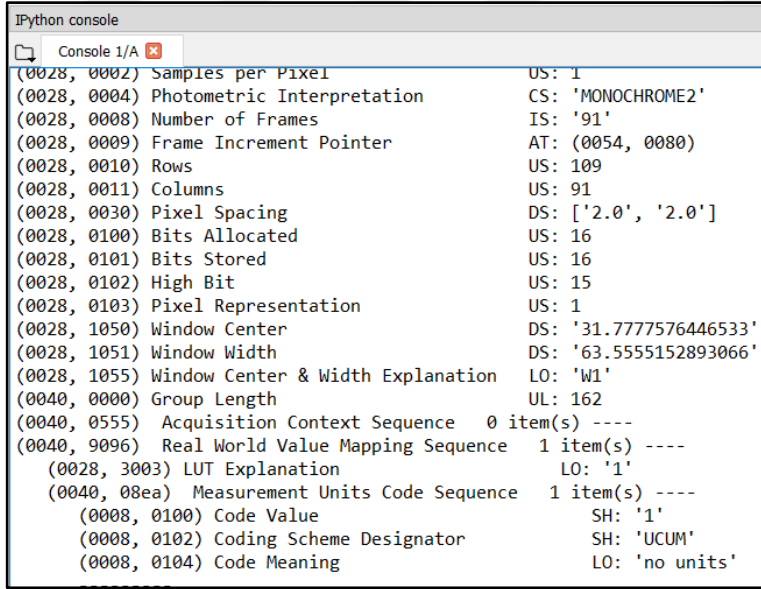
Her bireye ait 91 kesit görüntü PPMI web sayfası üzerinden görüntülenebildiği gibi, indirilen “dcm” uzantılı dicom dosya formatı üzerinden farklı programlar aracılığıyla erişilebilmektedir. Bu çalışmada Python programlama dili üzerinden bu işlemleri gerçekleştirmek için daha önce bahsedilen dicom kütüphanesi yüklenmiştir. Ayrıca 91 kesitten oluşan her bir resmi görüntülemek için matplotlib kütüphanesi yüklenmiştir.

Kütüphaneler yüklendikten sonra Spyder programı üzerinden yeni bir dosya oluşturup dicom dosyası üzerinden verileri kontrol edilebilir. Dicom dosyası görüntünün alındığı cihaz, oluşturulma tarihi, hasta id bilgisi, cinsiyet, kesit sayısı ve her kesite ait görüntü vb. pek çok bilgiyi barındırmaktadır.

Çizelge 8. Dicom görüntüsünü incelemek için gerekli kodlar

import dicom
import matplotlib.pyplot as plt
deger = dicom.read_file("3008_control.dcm")
deger
deger.dir("patient")
deger.PatientSex
img = deger.pixel_array[40]
plt.imshow(img)

Dicom dosyasındaki kesit görüntülerine ve bilgilere ulaşmak için gerekli kütüphaneler import edilmelidir. “dicom.read_file” komutu ile dcm uzantılı dosyasının adres bilgisi verilerek Python üzerinde oluşturulan “deger” isimli değişkene aktarılmaktadır. Bundan sonraki işlemlerde “deger” değişkeni kullanılacaktır. “deger” değişkeni tek satır olarak çalıştırdığında aşağıdaki dosya bilgileri görüntülenebilir.



IPython console	
Console 1/A	
(0028, 0002) Samples per Pixel	US: 1
(0028, 0004) Photometric Interpretation	CS: 'MONOCHROME2'
(0028, 0008) Number of Frames	IS: '91'
(0028, 0009) Frame Increment Pointer	AT: (0054, 0080)
(0028, 0010) Rows	US: 109
(0028, 0011) Columns	US: 91
(0028, 0030) Pixel Spacing	DS: ['2.0', '2.0']
(0028, 0100) Bits Allocated	US: 16
(0028, 0101) Bits Stored	US: 16
(0028, 0102) High Bit	US: 15
(0028, 0103) Pixel Representation	US: 1
(0028, 1050) Window Center	DS: '31.7777576446533'
(0028, 1051) Window Width	DS: '63.5555152893066'
(0028, 1055) Window Center & Width Explanation	LO: 'W1'
(0040, 0000) Group Length	UL: 162
(0040, 0555) Acquisition Context Sequence	0 item(s) ----
(0040, 9096) Real World Value Mapping Sequence	1 item(s) ----
(0028, 3003) LUT Explanation	LO: '1'
(0040, 08ea) Measurement Units Code Sequence	1 item(s) ----
(0008, 0100) Code Value	SH: '1'
(0008, 0102) Coding Scheme Designator	SH: 'UCUM'
(0008, 0104) Code Meaning	LO: 'no units'

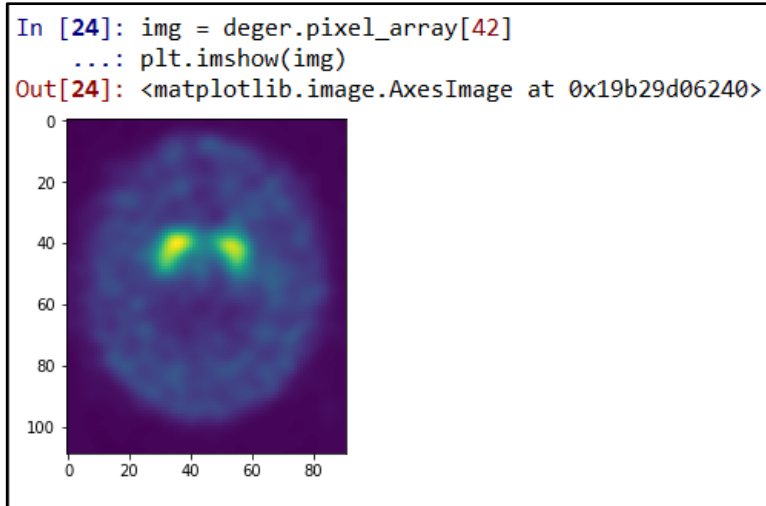
Şekil 39. Dicom dosyasından alınan metinsel veriler

“deger.dir” komutu içerisinde verilen bölüm ismi ile o bölümde hangi verilerin olduğu bilgisine ulaşılabilir. Kod içerisinde verilen patient komutu ile hasta hakkında hangi bilgilere ulaşılacağı listelenir. Aşağıdaki listede patient bölümünden

alınabilecek bilgiler yer almaktadır. Örnek olarak listede bulunan cinsiyet bilgisinin değerini görmek için “deger.PatientSex” komutu kullanılmaktadır.

```
['PatientBirthDate',  
'PatientGantryRelationshipCodeSequence',  
'PatientID',  
'PatientName',  
'PatientOrientation',  
'PatientOrientationCodeSequence',  
'PatientSex',  
'PatientSize',  
'PatientWeight']
```

“deger.pixel_array” komutu ile dosya içerisinde bulunan kesitlerin istenilen index numarasına ait görüntü verisi alınabilmektedir. Örnek olarak 40 index numarasına ait 41. kesit görüntü verisi değişkene alınmaktadır. Değişkene alınan bu bilgi “plt.imshow” komutu ile ekranda görüntülenmesi sağlanmaktadır. Komut çalıştırıldıktan sonra çıktısı aşağıdaki gibidir.



Şekil 40. Dicom dosyasından alınan 41. kesit görüntüsü

Görüntü dosyaları tek tek alınabildiği gibi Python dili ile döngü yapısı kurularak tüm kesitlerin görüntülenmesi de sağlanabilmektedir. Çizelge 9’da kullanılan kod yapısı ile tüm kesitler tek bir grafik şeklinde görüntülenebilir. “figure” komutu ekrana yansıtılacak görüntünün büyüklüğünü belirlemek için kullanılmaktadır. “figsize” komutu ile inch cinsinden genişlik ve yükseklik değerleri belirlenmektedir. Tüm görüntülerin bir arada daha iyi görüntülenmesi için satır ve sütunlar kullanılmıştır. Sütun sayısı 9, satır sayısı ise 11 olarak belirlenmiştir. “range” komutu oluşturulacak döngünün başlangıç ve bitiş değerlerini belirlemek için kullanılmaktadır. Index numaraları sıfırdan başladığı için range içerisinde 0, toplam 91 kesit olduğu için 91 değerleri girilmiştir. Range komutunda bitiş değeri olan 91 değeri kullanılmayacak, döngü 90 değerinde tamamlanacaktır. “add_subplot” komutu ile ilk sıradan başlayarak alınan kesit görüntüleri grafik üzerine eklenmektedir. En son “plt.show()” komutu ile oluşturulan grafik ekranda gösterilmektedir. Şekil 41’de dicom dosyasından alınan kesitlerin gösterildiği grafik yer almaktadır.

Çizelge 9. Dicom dosyasından tüm kesitleri görüntülemek için gerekli kodlar

```
fig=plt.figure(figsize=(15, 15))

columns = 9

rows = 11

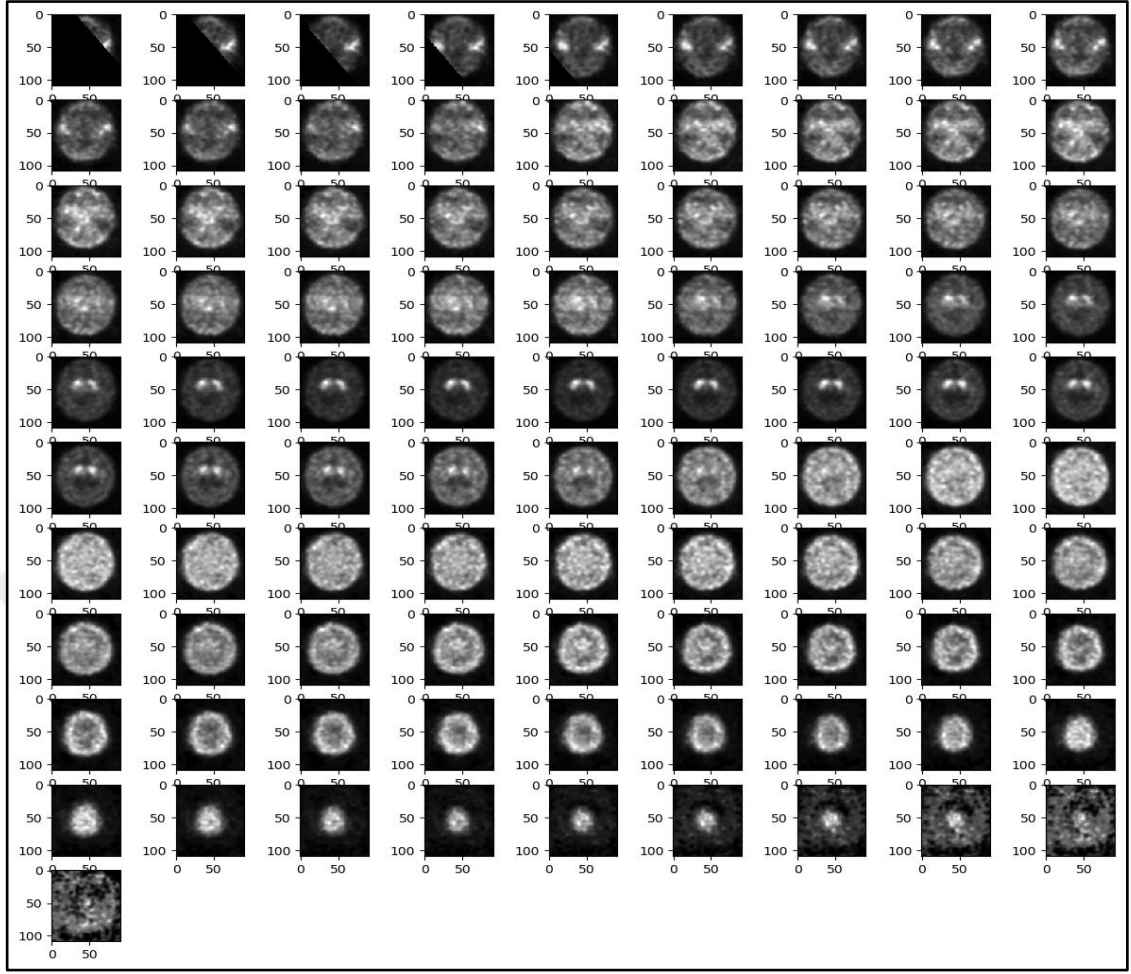
for i in range(0, 91):

    img = ds1.pixel_array[i]

    fig.add_subplot(rows, columns, i+1)

    plt.imshow(img, cmap=plt.cm.gray)

plt.show()
```



Şekil 41. Dicom dosyasından alınan 91 adet kesitin görüntüsü

Spect görüntüler üzerinden hastalık teşhisi için daha önce yapılmış birçok çalışmada 41. kesit görüntüsü kullanılmıştır. Zhang ve Kagen (2017) 40, 41 ve 42. kesitler üzerinde çalıştılar. Prashanth et al. (2017) yaptıkları çalışmada 35 ile 48. kesitleri kullandılar ve en iyi sonuçları 41. kesitte aldılar.

Bu nedenle tez çalışmamızda en iyi sonuçlar alınan 41. kesit üzerinde çalışılarak model geliştirilmiş ve test edilmiştir. Bunun için dicom veri dosyalarından 41. kesitlerin seçilmesi gerekmektedir. Dcm uzantılı dosyalar control, pd ve swedd olarak ayrı ayrı klasörlere kaydedilmiştir. Aşağıda (Şekil 42) control grubundan olan kayıtların klasör görüntüsü verilmiştir.

oalkan > SpyderProjects > spectImage > img > control			Ara: co
Ad	Değiştirme tarihi	Tür	
PPMI_3000_NM_Reconstructed_DaTSCAN_Br_20120814154829508_1_S117534_I323662.dcm	4.03.2019 02:55	DCM Dosyası	
PPMI_3004_NM_Reconstructed_DaTSCAN_Br_20121019114136675_1_S124286_I341194.dcm	4.03.2019 03:10	DCM Dosyası	
PPMI_3008_NM_Reconstructed_DaTSCAN_Br_20121019114149366_1_S124288_I341195.dcm	4.03.2019 03:11	DCM Dosyası	
PPMI_3009_NM_Reconstructed_DaTSCAN_Br_20121019114204864_1_S127540_I341196.dcm	4.03.2019 03:11	DCM Dosyası	
PPMI_3011_NM_Reconstructed_DaTSCAN_Br_20121019114236829_1_S117537_I341198.dcm	4.03.2019 03:11	DCM Dosyası	
PPMI_3013_NM_Reconstructed_DaTSCAN_Br_20121019114322662_1_S131267_I341200.dcm	4.03.2019 03:11	DCM Dosyası	
PPMI_3016_NM_Reconstructed_DaTSCAN_Br_20121019114502002_1_S142196_I341202.dcm	4.03.2019 03:11	DCM Dosyası	
PPMI_3029_NM_Reconstructed_DaTSCAN_Br_20130909134214102_1_S190842_I388468.dcm	4.03.2019 03:16	DCM Dosyası	
PPMI_3053_NM_Reconstructed_DaTSCAN_Br_20121019114713706_1_S117541_I341207.dcm	4.03.2019 03:11	DCM Dosyası	
PPMI_3055_NM_Reconstructed_DaTSCAN_Br_20121019114740043_1_S117543_I341209.dcm	4.03.2019 03:11	DCM Dosyası	
PPMI_3057_NM_Reconstructed_DaTSCAN_Br_20121019114802195_1_S117545_I341211.dcm	4.03.2019 03:11	DCM Dosyası	
PPMI_3064_NM_Reconstructed_DaTSCAN_Br_20121019114919650_1_S124289_I341217.dcm	4.03.2019 03:11	DCM Dosyası	
PPMI_3069_NM_Reconstructed_DaTSCAN_Br_20121019115137949_1_S117551_I341221.dcm	4.03.2019 03:12	DCM Dosyası	
PPMI_3070_NM_Reconstructed_DaTSCAN_Br_20121019115215225_1_S124291_I341222.dcm	4.03.2019 03:12	DCM Dosyası	
PPMI_3071_NM_Reconstructed_DaTSCAN_Br_20121019115226621_1_S131269_I341223.dcm	4.03.2019 03:12	DCM Dosyası	
PPMI_3072_NM_Reconstructed_DaTSCAN_Br_20121019115242665_1_S124292_I341224.dcm	4.03.2019 03:12	DCM Dosyası	
PPMI_3073_NM_Reconstructed_DaTSCAN_Br_20121019115253890_1_S127542_I341225.dcm	4.03.2019 03:12	DCM Dosyası	
PPMI_3074_NM_Reconstructed_DaTSCAN_Br_20121019115307771_1_S131270_I341226.dcm	4.03.2019 03:12	DCM Dosyası	

Şekil 42. Control grubu dicom dosyaları klasör görüntüsü

Spect görüntüsü 91 kesitten oluşan 16 bitlik gri tonlama ile kodlanmış 91x109 piksel değerlerinden oluşur (Zhang, 2017). 16 bitlik bir görüntü dosyası 0 ile 65536 sayısal değer aralığında bulunur. 16 bitlik görüntü ile kayıtlı resimlerde renk geçişleri daha kaliteli olarak karşımıza çıkmaktadır. Çizelge 10’da yer alan Python kod blokları ile dcm uzantılı dosya üzerinden istenilen kesitler alınarak resim olarak kaydetme işlemi gerçekleştirilmiştir. “ResimOlustur” olarak tanımlanmış fonksiyonun iki argüman değeri vardır. Birinci değer resim olarak ayrıştırılmak istenen grup bilgisini, ikincisi ise hangi kesit ayrılmak isteniyorsa o kesitin karşılığı olan index numarasını ifade eder. Klasör içerisinde her bir dosya için işlem yapabilmek adına operating system (işletim sistemi) kısaltması olarak kullanılan “os” kütüphanesi kullanılmıştır.

Alınan her bir resimde piksel değerleri 0 ile 65536 arasında olduğundan bu değerler ile eğitim yapmak zor olabilir. Bu durumu kolaylaştırmak için alınmak istenen kesit piksel değeri “skimage” kütüphanesinde bulunan “img_as_ubyte” fonksiyonu kullanılarak resim 0 ile 255 arasında değerlere dönüştürülür. Dönüştürme işleminden sonra resimler isimli klasör altında ilgili araştırma grubu adına açılan klasör içine resim dosyası olarak kaydedilir.

Çizelge 10. Dicom dosyasından ilgili kesitlerin resim olarak ayrılması

```
import os
import dicom
import imageio
from skimage import img_as_ubyte

def dosyaAl(path):
    dirList = os.listdir(path)
    dirList.sort()
    fnames = []
    for fname in dirList:
        if os.path.isfile(path + fname):
            fnames.append(fname)
    return fnames

def ResimOlustur(etiket, kesit):
    i=0
    for item in dosyaAl("img\\" + etiket + "\\"):
        imageio.imwrite("resimler\\" + etiket + "\\" + item[5:-68] + "_" + str(i) + ".jpg",
img_as_ubyte(dicom.read_file("img\\" + etiket + "\\" + item).pixel_array[kesit]))
        i+=1
    ResimOlustur("control", 40)
    ResimOlustur("pd", 40)
    ResimOlustur("swedd", 40)
```

3.3.3. Veri Setini Uyumlu Hale Getirme

Resim dosyaları üzerinden okunan değerlerin derin öğrenme uygulamasında kullanılacak hale getirilmesi gerekmektedir. Bunun için “Numpy” kütüphanesi kullanılacaktır. Python içerisinde dizi işlemleri yapılabilmektedir. Fakat “Numpy” kütüphanesi daha hızlı ve fonksiyonel olduğu için sıklıkla tercih edilmektedir.

Çizelge 11. Resim dosyalarını Numpy dizisine dönüştüren kodlar

```
import numpy as np
from os import listdir
from skimage.transform import resize
from keras.utils import to_categorical
import imageio

def resmi_al(path):
    resim = imageio.imread(path, as_gray= True)
    resim.flatten();
    resim = resize(resim, (91, 109, 1))
    return resim

etiketler = listdir("resimler")
X, Y = [], []

for i, etiket in enumerate(etiketler):
    etiket_klasoru = 'resimler/'+etiket

    for resim_adi in listdir(etiket_klasoru):
        resim = resmi_al(etiket_klasoru+'/'+resim_adi)
        X.append(resim)
        Y.append(i)

X = np.array(X).astype('float32')/255.
X = X.reshape(X.shape[0], 91, 109, 1)
Y = np.array(Y).astype('float32')
Y = to_categorical(Y, 3)
```

Çizelge 11’de yer alan kod blokları ile daha önce Dcm uzantılı dosyalardan alıp resim olarak kaydedilen bilgiler derin öğrenme işlemlerinde kullanılmak için daha işlevsel olan Numpy dizilerine dönüştürülmektedir.

İlk önce numpy, os, skimage.transform ve keras.utils kütüphaneleri import ediliyor. Numpy kütüphanesi ile resimlerden alınan piksel bilgileri dizi olarak saklanır ve içerisindeki değerler istenilen değerlere dönüştürülebilir. “os” kütüphanesi resimler klasörü içerisindeki klasör isimlerini almak için kullanılmaktadır. etiketler = listdir("resimler") komutu ile resimler klasöründe bulunan control, pd ve swedd isimleri otomatik olarak etiketler değişkenine aktarılmaktadır. Kod içerisinde kullanılan for döngüsü ile resimler klasöründeki tüm alt klasörlerdeki resimlerin değerleri X değişkenine, bu bilgilere karşılık gelen

control, pd ve swedd değerleri için sırasıyla 0,1 ve 2 değerleri olarak Y değişkenine kaydedilir.

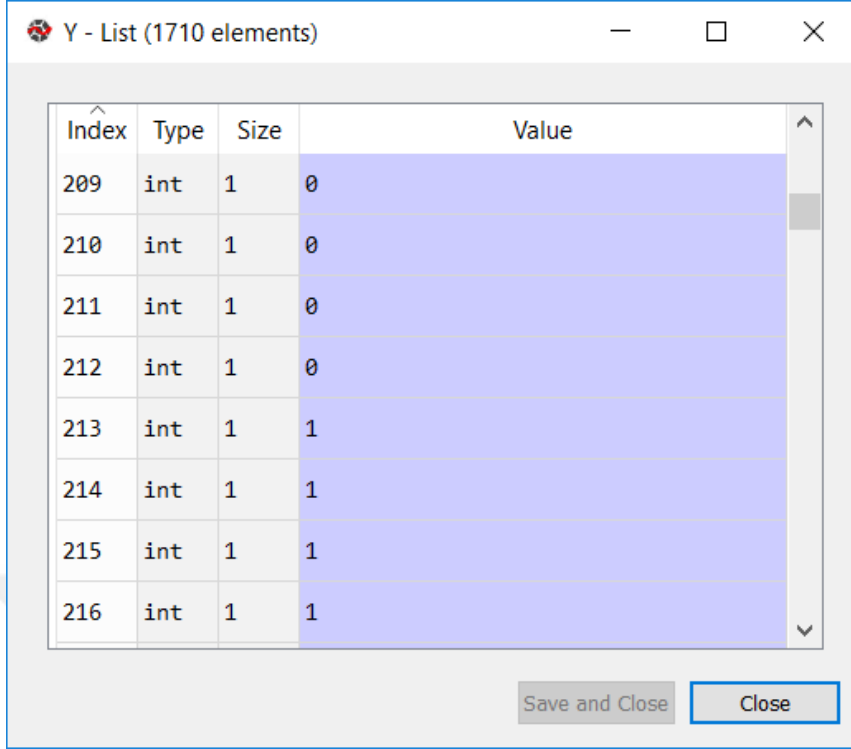
Variable explorer			
Name	Type	Size	Value
X	list	1710	[Numpy array, Numpy array, Numpy array, Numpy array, Numpy array,
Y	list	1710	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...]
etiketler	list	3	['control', 'pd', 'swedd']

Şekil 43. Okunan resimlerin Numpy dizilerine dönüştürülmesi

Yukarıdaki şekilde for döngüsü ile otomatik olarak resimlerden okunan verilerin aktarıldığı değişkenler gösterilmiştir. Etiketler değişkeni içerisindeki veriler listdir komutu ile otomatik çekilmiştir. X, 1710 adet olan resimlerden okunan (91,109,1) boyutlarındaki piksel değerlerinin saklandığı float64 tipinde tanımlanmış list türünde değişkendir. Y ise control resimlerinin karşılığı için 0, pd için 1 ve swedd değerleri için 2 sayısal verileri bulunduran list türünde değişkendir. Aşağıdaki şekillerde X ve Y değişkenlerindeki veriler gösterilmiştir.

X - List (1710 elements)			
Index	Type	Size	Value
0	float64	(91, 109, 1)	[[[0.] [0.]
1	float64	(91, 109, 1)	[[[3.] [3.]
2	float64	(91, 109, 1)	[[[0.] [0.]
3	float64	(91, 109, 1)	[[[0.] [0.]
4	float64	(91, 109, 1)	[[[5.] [5.]
5	float64	(91, 109, 1)	[[[2.] [2.]
6	float64	(91, 109, 1)	[[[5.] [5.]
7	float64	(91, 109, 1)	[[[3.] [3.]
8	float64	(91, 109, 1)	[[[4.08256881] [4.75229358] [4.75229358]

Şekil 44. Resimlerden okunan piksel değerlerinin saklandığı X dizisi



Index	Type	Size	Value
209	int	1	0
210	int	1	0
211	int	1	0
212	int	1	0
213	int	1	1
214	int	1	1
215	int	1	1
216	int	1	1

Şekil 45. Resimlerin karşılığı olan değerlerin saklandığı Y dizisi

Oluşturulan X ve Y değişkenleri list türündeki değişkenlerdir. Bun değişkenlerin numpy dizisine dönüştürülmeleri gerekmektedir. Bunun için np.array ve astype kodları kullanılmaktadır. “reshape” komut ile X değişkeni (1710,91,109,1) boyutlarında veri saklayan bir değişken haline getirilir. Y değişkeninde bulunan değerlerin kullanılabilmesi için kategorik olarak kodlanması gerekmektedir. Bunun için “to_categorical” fonksiyonu kullanılmıştır. Y değişkeninde değerler ilk olarak tek boyutlu halde 0,1 ve 2 şeklinde bulunmaktaydı. Kategorik dönüşüm işleminden sonra değerler aşağıdaki şekle dönüşür.

- 0 için (1,0,0)
- 1 için (0,1,0)
- 2 için (0,0,1)

Variable explorer			
Name	Type	Size	
X	float32	(1710, 91, 109, 1)	[[[0.] [0.]
Y	float32	(1710, 3)	[[1. 0. 0.] [1. 0. 0.]
etiketler	list	3	['control', 'pd', 'swedd']

Şekil 46. Değişkenleri şekillendirme ve kategorik dönüşüm sonrası görüntüleri

3.3.4. Eğitim ve Test Seti Oluşturma

Kullanılacak verilerin tamamı X ve Y isimli değişkenler üzerinde saklanmaktadır. Peki bu verilerin iki grup (X ve Y) olarak saklanmasıdaki amaç nedir? Çizelge 12’de metin madenciliği derslerinde sıklıkla kullanılan nem, rüzgâr ve hava durumuna göre tenis oyunu oynanma durumlarını içeren veri seti örneği bulunmaktadır. Bu veri setinde nem, rüzgâr ve hava değerleri X değişkeninde, tenis oyunu oynanma durumu ise Y değişkeninde saklanmaktadır. X değişkeninin içerisinde durumlar olduğunda, Y değişkeninde oyun oynanıp oynanmama durumu meydana gelmektedir. Bilgisayara eğitim işlemi yaptırılırken X üzerinde veriler oluştuğunda Y değişkeninde karşılığı meydana getir” ifadesi iletilmektedir. Yani Y değişkeninde tahmini yapılacak olan veri grubu bulunmaktadır.

Çizelge 12. Veri seti üzerinde değişken örneği

X			Y
Nem	Rüzgâr	Hava	Oyun Oynanma Durumu
Az Nemli	Rüzgârlı	Güneşli	Evet
Çok Nemli	Aşırı Rüzgârlı	Aşırı Güneşli	Hayır
Çok Nemli	Rüzgâr Yok	Kapalı	Evet
Az Nemli	Rüzgâr Yok	Kapalı	Evet

Bu tez çalışmasında tahmini yapılacak olan değer görüntünün hangi gruba (control, pd ve swedd) ait olduğunu gösteren Y değişkeninde saklanmaktadır. Derin öğrenme uygulamasının yapılması için X ve Y verilerinin eğitim ve test olarak iki gruba ayrılması gerekmektedir. Eğitim işleminde eğitim için ayrılan küme, daha sonra bu eğitimin doğruluğunu test etmek için test kümesi kullanılacaktır. Değişken grubu X ve Y olarak iki gruptan oluştuğu için eğitim ve test kümeleri de iki gruptan oluşacaktır. Eğitim ve test grupları Çizelge 13’de verildiği şekilde isimlendirilmiştir.

Çizelge 13. Eğitim ve Test değişkenlerinin isimlendirilmesi

Veri Değişkenleri	Eğitim Değişkenleri	Test Değişkenleri
X	x_train	x_test
Y	y_train	y_test

Değişken verileri tanımlandıktan sonra her eğitim işleminde resimlerden veri okunması ve değişkenler üzerindeki işlemlerin yapılmaması için bu işlemlerin kayıt altına alınması çalışma verimliliğini arttıracaktır. Çizelge 14’de verilen kod blokları ile X ve Y değişkenlerin eğitim ve test grubu olarak ayrılması ve daha sonra kullanılabilirlikte kaydedilmesi işlemlerini gerçekleştirmektedir.

Değişkenler “train_test_split” fonksiyonu ile x_train, y_train, x_test ve y_test olarak 4 gruba rastgele ayrılmaktadır. Bu fonksiyon içerisine 4 adet veri girişi yapılmalıdır. İki tanesi kullanılacak değişken grupları olan X ve Y değişkenleridir diğeri ise test_size değeri yani toplam verinin ne kadarının test kümesinde bulunacağını gösteren orandır. Yapılan çalışmalarda genelde Eğitim oranı %70, test oranı %30 olarak ayarlanmaktadır. “test_size” değeri 0.3 olarak girildiğinde toplam verinin %30’u test grubuna aktarılacaktır. Bu işlemin ardından eğitim grubu 1197, test grubu ise 513 olarak ayrılmış olur.

Veriler değişkenlere aktarılırken ilk önce control grubu, sonra pd grubu ve en son olarak swedd grubu aktarılmıştır. Fakat eğitim işlemine geçmeden bu verilerin rastgele olarak karıştırılması gerekir. Çalışmada kullanılan veriler dikkate alındığında eğitim için ayrılan veri sayısı 1197 adettir. Eğer rastgele karışım yapılmadan eğitim

ve test gruplarını oluşturulursa eğitim grubuna swedd grubundan hiçbir değer girmeyecektir. Bu durumda yapılan eğitim işlemi test edildiğinde başarısız bir durum ortaya çıkaracaktır.

“train_test_split” fonksiyonunda kullanılan dördüncü argüman “random_state” değeridir. Rastgele sayı oluşturmak için kullanılır. Girilmesi zorunlu değildir. Sabit bir değer girilmesi durumunda kod çalıştırıldığında her defasında aynı şekilde rastgele bölünme meydana gelecektir. Bu değerın sabit yapılması için iki neden gösterilebilir. Birinci neden olarak her eğitim işleminde aynı sonuçların elde edilmesi sağlanmış olacaktır. İkinci neden ise oluşturulan model üzerinde veya parametrelerde yapılan değişikliklerin etkisi daha rahat görüntülenebilir.

Çizelge 14. Verileri eğitim ve test olarak ayırma ve kaydetme

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(X,Y,test_size=0.3, random_state=42)
np.save('veriseti/x_train.npy', x_train)
np.save('veriseti/x_test.npy', x_test)
np.save('veriseti/y_train.npy', y_train)
np.save('veriseti/y_test.npy', y_test)
```

Eğitim ve test grubu değişkenleri oluşturulduktan sonra her defasında bu işlemlerin yapılmaması için numpy dizilerine kaydedilebilir. Bu işlem için save fonksiyonu kullanılmaktadır. Fonksiyon içerisine kaydedilecek dizin ve değişken ismi verilerek kayıt işlemi gerçekleştirilir. x_train, x_test, y_train ve y_test olarak isimlendirilen değişkenler aynı isimlerde “veriseti” isimli klasör içerisine kaydedilmiştir.

3.4. Modelleme ve Sonuçlar

Araştırmada amaç contol, pd ve swedd kategorilerindeki görüntüleri kullanarak ele alınan spect görüntünün hangi kategoriye ait olduğunu bulmaktır. Yapay sinir ağında görüntü katmanlardan geçirilerek hangi kategoriye ait olduğu sonucu bulunmaya çalışılacaktır. Yapay sinir ağları convolutional katman, pooling,

flatten ve dense katmanlarının harmanlanmasından oluşmaktadır. Bu parametrelerin oluşturulması ve parametrelerin neden seçildikleri sırasıyla verilecektir.

3.4.1. Model Katmanları Oluşturma

Model katmanları oluşturulurken katman sayısının çokluğu verilerin işlenmesi ve gerekli bilgi çıkarımı için önemli bir husustur. Çalışma görüntü işleme üzerine olduğu için katmanlarda Conv2D Evrişimsel sinir ağı yapısı tercih edilmiştir. Evrişimsel sinir ağı için genelde 32, 64 ve 128 değerleri kullanılmaktadır. İlk önce 32 değerinden başlayıp iki sayısının katları olarak artarak eklenebilir. Katlar arttıkça daha çok işlem yapılacağından derin öğrenme uygulamasının yapılacağı cihaz seçimi önem kazanmaktadır.

Görüntü işleme üzerine çalışmalarda kullanılacak görüntüler 128x128 piksel değerinden yüksek boyutlarda oluştuğunda uygulanacak kernel size (çekirdek boyutu) 5x5 ve 7x7 tercih edilmektedir. Fakat 128x128 piksel değerinden düşük olan resim öğelerinden bilgi çıkarımı yapılırken 1x1 veya 3x3 değerleri daha iyi sonuçlar vermektedir. Bunun nedeni düşük çekirdek boyutu ile düşük piksellik resimlerden daha iyi özellik çıkarımı yapılabilmesidir. Düşük boyutlu resimlerden 5x5 veya 7x7 çekirdek boyutları kullanıldığında resimden alınması gereken bilgilerde kayıplar gerçekleşebilir. Bundan dolayı çalışmamızda kernel size (çekirdek boyutu) değeri olarak 3x3 tercih edilmiştir.

Çalışmamızdaki resim boyutu değeri 91x109 piksel değerlerine sahip olduğu için padding değeri olarak same tercih edilmiştir. Same değeri kullanıldığında kernel size (çekirdek boyutu) resim üzerine uygulandığında eksik kalan sütun veya satırlar için resim boyutlarında doldurmalar ile bilgi kayıpları önlenmektedir.

Aktivasyon fonksiyonu olarak son katmanda “sigmoid” ve “softmax”, model içerisinde ise “relu”, “elu”, “selu” ve “tanh” aktivasyon fonksiyonları kullanılmıştır. Eğitim esnasında hız önemli bir değer olduğunda Relu tercih edilen fonksiyonlar arasında birinci sırayı almaktadır.

Model üzerine belli bir aşamadan sonra MaxPooling2D katmanı eklenmiştir. Bu katman model yapıları arasında daha doğru sonuca ulaşmak için verileri belli oranlarda azaltmak için kullanılmaktadır. Aynı şekilde model içerisinde belli

noktalara dropout katmanı eklenerek aşırı uydurma (ezberleme) olayı ortadan kaldırılmıştır. Modelin son katmanları içerisine ise flatten (düzleştirme) katmanı eklenerek tahmin edilecek değer için veriler tek vektör yapıya dönüştürülmüştür. Tahmin edilecek 3 sınıf olduğu için en son katmanda bulunan Dense ile katman yapıları sonlandırılmıştır.

Çizelge 15. Çalışma için oluşturulan örnek model yapısı

model = Sequential()
model.add(Conv2D(32, (3, 3),padding='same',activation='relu',input_shape= (91,109,1)))
model.add(Conv2D(32, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Conv2D(64, (3, 3), padding='same', activation='relu'))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Conv2D(128, (3, 3), padding='same', activation='relu'))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(3, activation='softmax'))

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 91, 109, 32)	320
conv2d_2 (Conv2D)	(None, 89, 107, 32)	9248
max_pooling2d_1 (MaxPooling2D)	(None, 44, 53, 32)	0
dropout_1 (Dropout)	(None, 44, 53, 32)	0
conv2d_3 (Conv2D)	(None, 44, 53, 64)	18496
conv2d_4 (Conv2D)	(None, 42, 51, 64)	36928
max_pooling2d_2 (MaxPooling2D)	(None, 21, 25, 64)	0
dropout_2 (Dropout)	(None, 21, 25, 64)	0
conv2d_5 (Conv2D)	(None, 21, 25, 128)	73856
conv2d_6 (Conv2D)	(None, 19, 23, 128)	147584
max_pooling2d_3 (MaxPooling2D)	(None, 9, 11, 128)	0
dropout_3 (Dropout)	(None, 9, 11, 128)	0
flatten_1 (Flatten)	(None, 12672)	0
dense_1 (Dense)	(None, 3)	38019
Total params: 324,451		
Trainable params: 324,451		
Non-trainable params: 0		

Şekil 47. Oluşturulan modelin eğitim verileri için katman ve çıktı verileri

3.4.2. Model Derleme Kriterleri

Model derleme kriteri olarak 3 argüman bulunmaktadır. Bunlardan ilki tahmin edilen değer ile doğru değer arasındaki hata oranını gösteren “loss” değeridir. İkincisi eğitimin prosedürünün ayarlandığı “optimizer” değeri, üçüncüsü ise eğitim işleminin başarısını hangi bilgiye göre gösterileceğini belirleyen metric ayarlarıdır. Çalışmada metric ayarı olarak “accuracy” (doğruluk), “sensitivity” (duyarlılık), “specificity” (özgüllük) değerleri belirlenmiştir. Yapılan tahminlerin doğru olma durumu eğitimin başarısını göstermektedir. Çalışmada sınıflandırma etiketleri az sayıda olduğundan “loss” değeri olarak “binary_crossentropy” tercih edilmiştir.

Çizelge 16. Model derleme ayarları kod örneği

```
model.compile( loss = keras.losses.binary_crossentropy,
               optimizer = keras.optimizers.Adam(),
               metrics=['accuracy', 'sensitivity', 'specificity'])
```

Çalışmada eğitim prosedürünün ayarlandığı optimizer yöntemleri için keras kütüphanesi altında bulunan 7 adet yöntem için ayrı ayrı hesaplama yapılmış ve sonuçları değerlendirilmiştir. Kütüphane altında bulunan yöntemler Çizelge 17’de gösterilmiştir.

Çizelge 17. Keras kütüphanesi optimizer değerleri

Optimizer Yöntemleri
SGD
RMSprop
Adagrad
Adadelata
Adam
Adamax
Nadam

3.4.3. Model Eğitim İşlemi

Model katmanlarını oluşturma ve derleme kriterleri belirlendikten sonra eğitim işlemi gerçekleştirilebilir. Eğitim işlemini gerçekleştirmek için Çizelge 18’de yer alan kod bloğundaki fit fonksiyonu kullanılmalıdır. Fonksiyonun içerisinde eğitim için kullanılan veri değişkenleri “x_train, y_train”, eğitimi test etmek için “validation_data” değişkenleri “x_test, y_test”, eğitimin kaç kez tekrar edilmesi gerektiğini ayarlayan “epoch” ve eğitim esnasında eğitim verilerinden kaç parça veri alınacağını ayarlayan “batch_size” değerleri belirlenmelidir.

Çizelge 18. Model eğitimi kod bloğu

```
model.fit( x_train, y_train,  
          batch_size=64,  
          epochs=20,  
          verbose=1,  
          validation_data=(x_test, y_test))
```

Çalışmamızda fit fonksiyonu için belirlenecek epoch ve batch_size değişkenlerinden epoch için 20-50-100, batch size için 64-128 değerleri kullanılarak eğitim işlemi yapılacaktır. Yapılan bu farklı değerler için eğitimin başarısı üzerinde doğruluk olarak artı katkısı olup olmadığı değerlendirilecektir.

Çizelge 19. Eğitim işlemi için kullanılacak epoch ve batch size değerleri

Kullanılacak Değişken	Kullanılacak Değerler
Epoch (Eğitim Tur Sayısı)	20, 50, 100
Batch_Size (Eğitim Parça Sayısı)	64, 128

3.4.4. Parametre Ayarlamaları

Tez çalışmasında eğitim işlemi yapılırken parametreler değiştirilerek eğitimin başarısı üzerinde testler gerçekleştirilmiştir. Model derleme bölümünde bahsedilen optimizasyon değerleri, model eğitim işlemi bölümünde bahsedilen epoch ve batch size değerlerine ek olarak aktivasyon fonksiyonları üzerinde de farklı

kombinasyonlar kullanılarak eğitimin başarısına etkisi incelenmiştir. Bu parametreler Çizelge 20’de gösterilmiştir.

Çizelge 20. Kullanılacak aktivasyon fonksiyonları

Kullanılacak Aktivasyon Türü	Kullanılacak Değerler
Model içerisinde kullanılan aktivasyon	relu, elu, selu, tanh
Çıkış aktivasyon	sigmoid, softmax

3.4.5. Eğitimi Görselleştirme

Görselleştirme ve tablolarla yapılan işlemleri daha iyi yorumlayabilmek, sonuçları daha rahat görebilmek adına kullanılmaktadır. Çalışmada eğitim işlemleri sonrasında accuracy (doğruluk), loss (kayıp), sensitivity (duyarlılık) ve specificity (özgüllük) değerlerinin eğitim boyunca değişimleri grafiksel olarak gösterilmiştir. Bu tablolar incelendiğinde eğitimin ilerleyiş süreci rahatlıkla görülebilir. Çizelge 21’de grafik oluşturmak için kullanılan kod bloğu verilmiştir.

Çizelge 21. Eğitim görselleştirme kod örneği

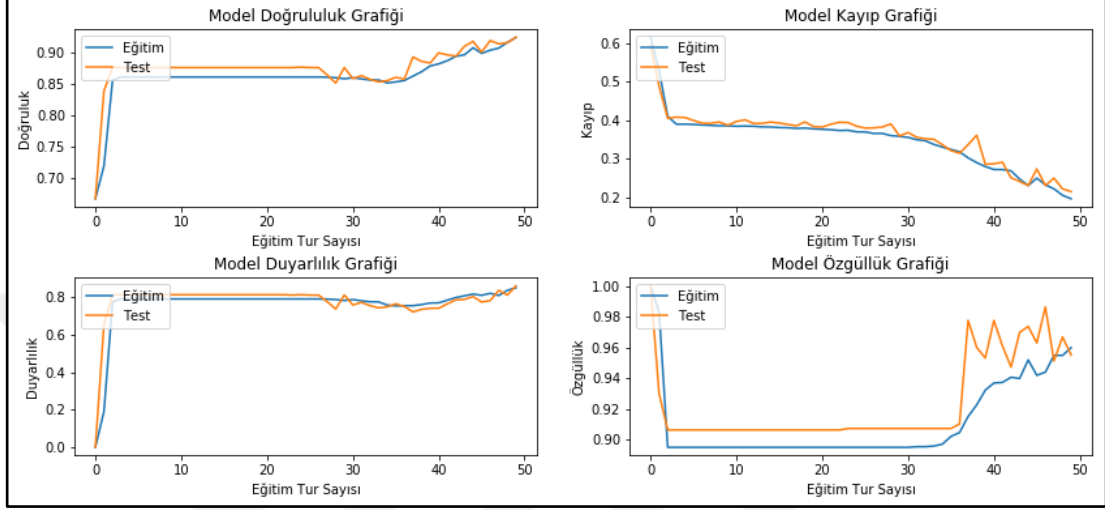
```
history = model.fit(x_train, y_train,
                    batch_size=64,
                    epochs=20,
                    verbose=1,
                    validation_data=(x_test, y_test))
from matplotlib import pyplot as plt

plt.figure(figsize=(14.,6.))
plt.subplots_adjust(hspace=0.4)
plt.subplot(2, 2, 1)
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('Model Doğruluk Grafiği')
plt.ylabel('Doğruluk')
```

```
plt.xlabel('Eğitim Tur Sayısı')
plt.legend(['Eğitim', 'Test'], loc='upper left')
plt.subplot(2, 2, 2)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Kayıp Grafiği')
plt.ylabel('Kayıp')
plt.xlabel('Eğitim Tur Sayısı')
plt.legend(['Eğitim', 'Test'], loc='upper left')
plt.subplot(2, 2, 3)
plt.plot(history.history['sensitivity'])
plt.plot(history.history['val_sensitivity'])
plt.title('Model Duyarlılık Grafiği')
plt.ylabel('Duyarlılık')
plt.xlabel('Eğitim Tur Sayısı')
plt.legend(['Eğitim', 'Test'], loc='upper left')
plt.subplot(2, 2, 4)
plt.plot(history.history['specificity'])
plt.plot(history.history['val_specificity'])
plt.title('Model Özgüllük Grafiği')
plt.ylabel('Özgüllük')
plt.xlabel('Eğitim Tur Sayısı')
plt.legend(['Eğitim', 'Test'], loc='upper left')
plt.show()
```

Yukarıdaki kod örneğinde görüldüğü gibi grafik çizimi için “matplotlib” kütüphanesi kullanılmıştır. 14x3 boyutunda 4 grafik oluşturulmuştur. Birinci grafikte modelin doğruluk oranındaki değişim, ikinci grafikte modelin loss (kayıp) değerindeki azalış durumu, üçüncü grafikte modelin duyarlılık oranı ve dördüncü grafikte modelin özgüllük değerindeki değişim gösterilmiştir. Eğitim işlemi verileri history isimli bir değişkene alınarak grafik çiziminde kullanıldığı görülmektedir. Doğruluk oranında “acc” ve “val_acc” isimlerinde iki değer bulunmaktadır. “acc”

değeri modelin eğitim işlemindeki artış oranıdır ve grafikte mavi ile çizilmiştir. “val_acc” değeri ise model eğitildikten sonra test verileri üzerindeki tahmin işlemi sonucu alınan doğruluk değeridir ve grafikte turuncu ile çizilmiştir. Aynı işlem diğer grafiklerde gösterilen değerler için de çalıştırılmıştır.



Şekil 48. Eğitim görselleştirme grafikleri

3.4.6. Eğitilmiş Modeli Test Etme

Eğitimi yapılmış bir model daha sonra farklı bir dosya üzerinde test edilebilmesi için kayıt altına alınabilir. H5 dosyası olarak kayıt altında alınan model daha sonra resimler üzerinde tahmin için kullanılabilir. “load_model” komutu ile daha önce kayıt altına alınan model kullanılmak üzere değişkene aktarılır. “test_resim” dosyasına test için ayrılan kümeden rastgele bir değer aktararak bu resim için tahmin işlemi gerçekleştirilir. “reshape” komutu ile veri dosyası (1,91,109,1) şekline dönüştürülür. İlk değer tahmin edilecek verinin 1 adet olduğunu, ikinci değer resim dosyasının genişliğini, üçüncü değer resim dosyasının yüksekliğini ve son değer ise resim dosyasının gri formatta olduğunu göstermektedir. Resim şekilsel olarak ayarlandıktan sonra “predict” komutu ile seçilen resim okutulan modele göre tahmin edilmektedir. Tahmin işleminden sonra “predict_classes” komutu ile resmin hangi sınıfta olduğu bilgisi alınabilmektedir. “predict_proba” komutu ile tahmin edilen resim dosyasının her sınıf için uyumluluk değerleri alınabilmektedir.

Çizelge 22. Eğitilmiş modeli bir resim üzerinden test etmek için gerekli kodlar

```
model_test = load_model('drive/uygulama/okanmodel.h5')
test_resim = x_test[178]
test_veri = test_resim.reshape(1,91,109,1)
model_test.predict(test_veri, batch_size=1)
preds = model_test.predict_classes(test_veri)
prob = model_test.predict_proba(test_veri)
print(preds, prob)
```

4. ARAŞTIRMA BULGULARI

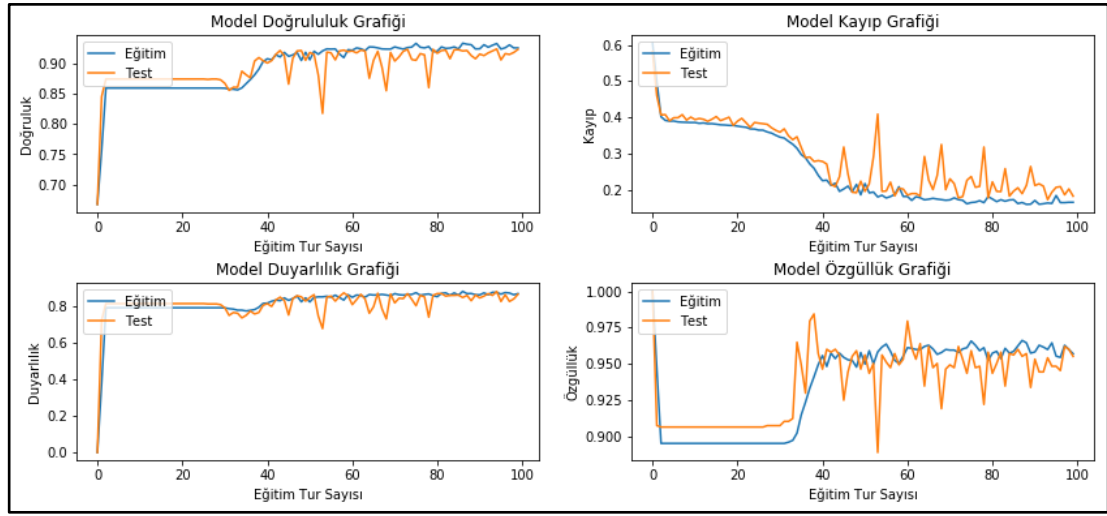
Bu bölümde Keras kütüphanesi içerisinde kullanılabilecek 7 farklı optimizasyon yöntemi 20-50-100 epoch ve 64-128 batch size değerleri kullanılarak doğruluk değerleri hesaplanmıştır. Doğruluk oranları arasında en iyi sonucu veren 3 değer için doğruluk, kayıp, duyarlılık ve özgüllük oranları çizelgelerde (Çizelge 23,24,25,26,27,28,29) listelenmiştir.

4.1. SGD Başarı Oranı

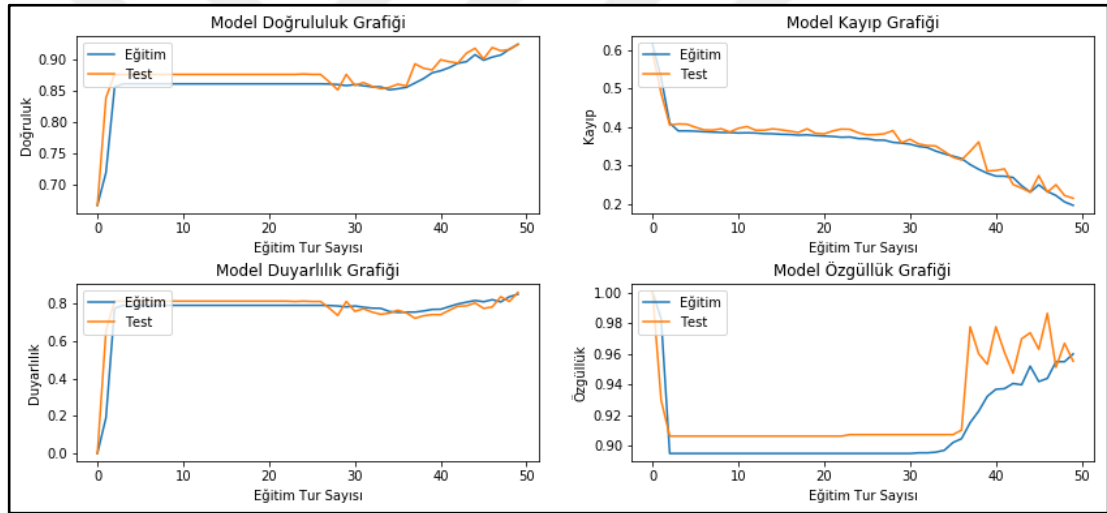
Çizelge 23’de SGD optimizasyon yöntemi kullanılarak elde edilmiş doğruluk, duyarlılık ve özgüllük değerleri yer almaktadır.

Çizelge 23. SGD optimizasyon yöntemi ile test edilmiş eğitim doğruluk, duyarlılık ve özgüllük oranları

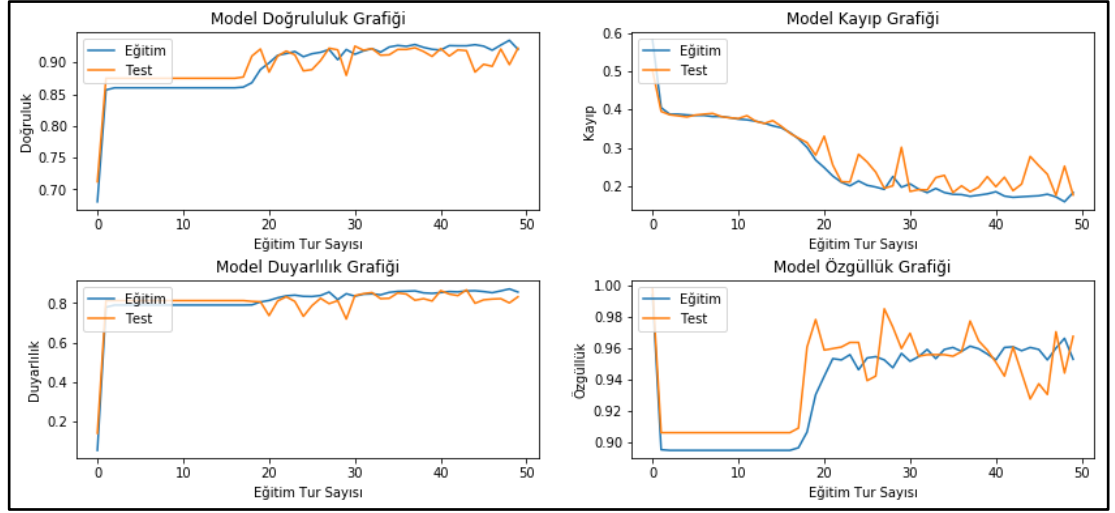
Aktivasyon	Epoch	Batch Size 64			Batch Size 128		
		Doğruluk	Duyarlılık	Özgüllük	Doğruluk	Duyarlılık	Özgüllük
Relu-Sigmoid	20	87,52	81,29	90,64	87,52	81,29	90,64
	50	92,14	85,77	95,32	87,52	81,29	90,64
	100	92,14	86,16	95,13	91,94	82,26	96,78
Relu-Softmax	20	87,13	79,14	91,13	87,52	81,29	90,64
	50	92,27	83,24	96,78	92,33	85,96	95,52
	100	91,81	84,60	95,42	92,46	86,35	95,52
Elu-Sigmoid	20	87,52	81,09	90,74	87,52	81,29	90,64
	50	80,31	73,68	83,63	89,08	72,51	97,37
	100	91,29	88,30	92,79	90,58	85,96	92,88
Elu-Softmax	20	79,21	54,78	91,42	85,38	74,66	90,74
	50	85,96	76,22	90,84	90,38	81,48	94,83
	100	90,19	83,24	93,66	90,06	83,24	93,47
Selu-Sigmoid	20	76,41	68,23	80,51	70,70	59,26	76,41
	50	77,00	73,88	78,56	78,88	73,88	81,38
	100	81,61	75,05	84,89	71,86	68,03	73,78
Selu-Softmax	20	75,11	61,40	81,97	65,95	41,33	78,27
	50	77,84	62,96	85,28	81,48	68,62	87,91
	100	84,83	76,02	89,38	76,15	61,40	83,53
Tanh-Sigmoid	20	87,52	81,29	90,64	87,52	81,29	90,64
	50	80,64	69,01	86,45	87,52	70,76	95,91
	100	87,26	81,48	90,16	78,23	73,49	80,60
Tanh-Softmax	20	86,22	69,79	94,44	84,93	73,29	90,74
	50	82,39	67,06	90,06	76,02	58,48	84,80
	100	90,77	83,24	94,54	84,86	70,37	92,11



Şekil 49. SGD optimizier, Relu-Softmax activation, 100 epoch, 128 batch size ile doğruluk (92,46) grafiği



Şekil 50. SGD optimizier, Relu-Softmax activation, 50 epoch, 128 batch size ile doğruluk (92,33) grafiği



Şekil 51. SGD optimizier, Relu-Softmax activation, 50 epoch, 64 batch size ile doğruluk (92,27) grafiği

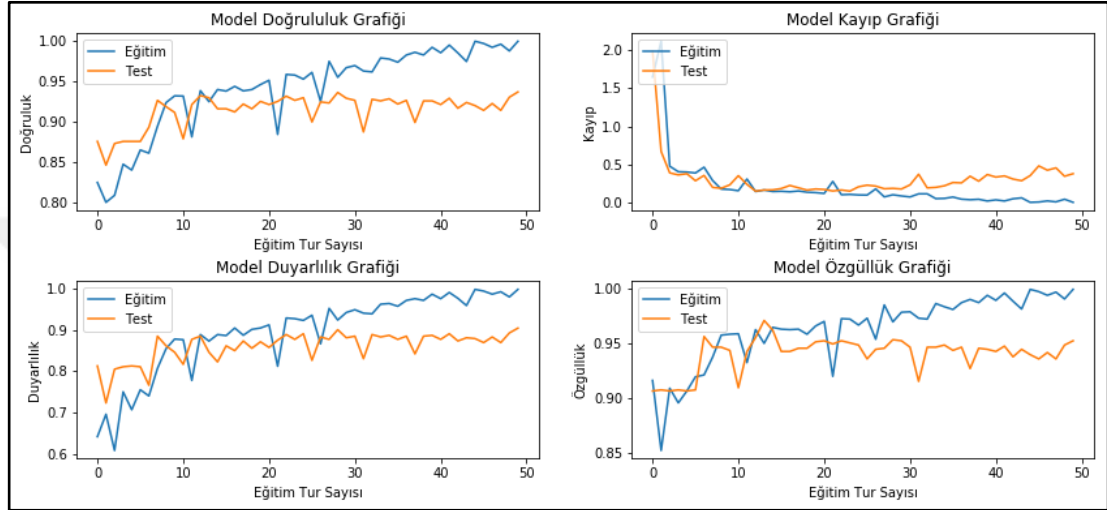
4.2. RMSprop Başarı Oranı

Çizelge 24’de RMSprop optimizasyon yöntemi kullanılarak elde edilmiş doğruluk, duyarlılık ve özgüllük değerleri yer almaktadır.

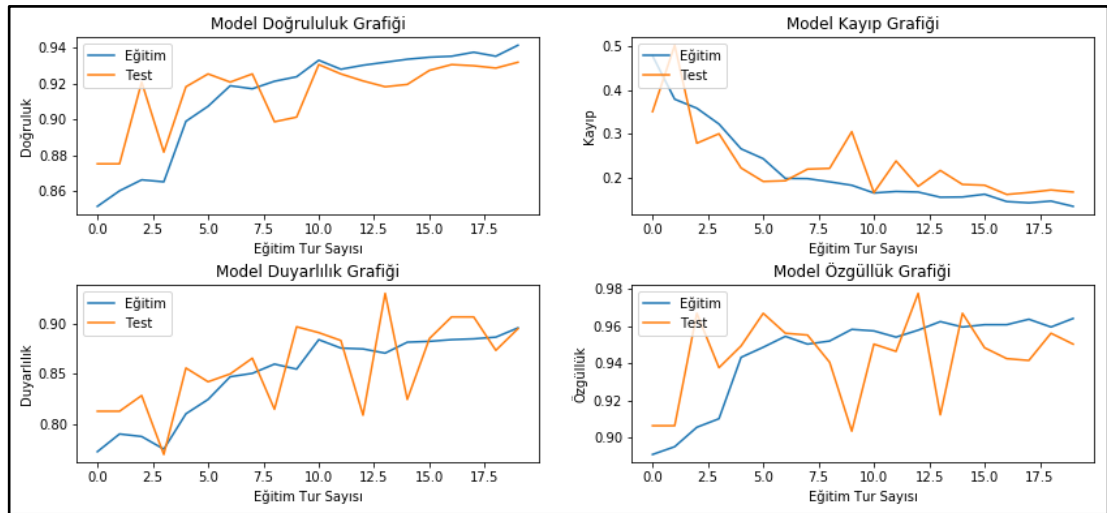
Çizelge 24. RMSprop optimizier ile test edilmiş eğitim doğruluk, duyarlılık ve özgüllük oranları

Aktivasyon	Epoch	Batch Size 64			Batch Size 128		
		Doğruluk	Duyarlılık	Özgüllük	Doğruluk	Duyarlılık	Özgüllük
Relu-Sigmoid	20	93,18	89,47	95,03	92,59	86,74	95,52
	50	92,85	83,82	97,37	92,40	80,70	98,25
	100	92,85	89,08	94,74	92,07	88,88	93,66
Relu-Softmax	20	93,05	85,38	96,88	91,55	86,74	93,96
	50	92,20	87,52	94,54	92,98	88,50	95,22
	100	92,98	89,28	94,83	93,05	89,47	94,83
Elu-Sigmoid	20	91,88	87,72	93,96	92,98	87,72	95,61
	50	90,45	88,50	91,42	90,45	85,77	92,79
	100	92,07	86,35	94,93	91,81	87,52	93,96
Elu-Softmax	20	92,38	88,11	94,54	92,85	88,30	95,13
	50	87,52	81,29	90,64	92,98	89,28	94,83
	100	91,88	87,72	93,96	92,53	88,69	94,44
Selu-Sigmoid	20	87,52	81,29	90,64	87,52	81,29	90,64
	50	87,52	81,29	90,64	87,52	81,29	90,64
	100	87,52	81,29	90,64	87,52	81,29	90,64
Selu-Softmax	20	87,52	81,29	90,64	87,52	81,29	90,64
	50	87,52	81,29	90,64	87,52	81,29	90,64

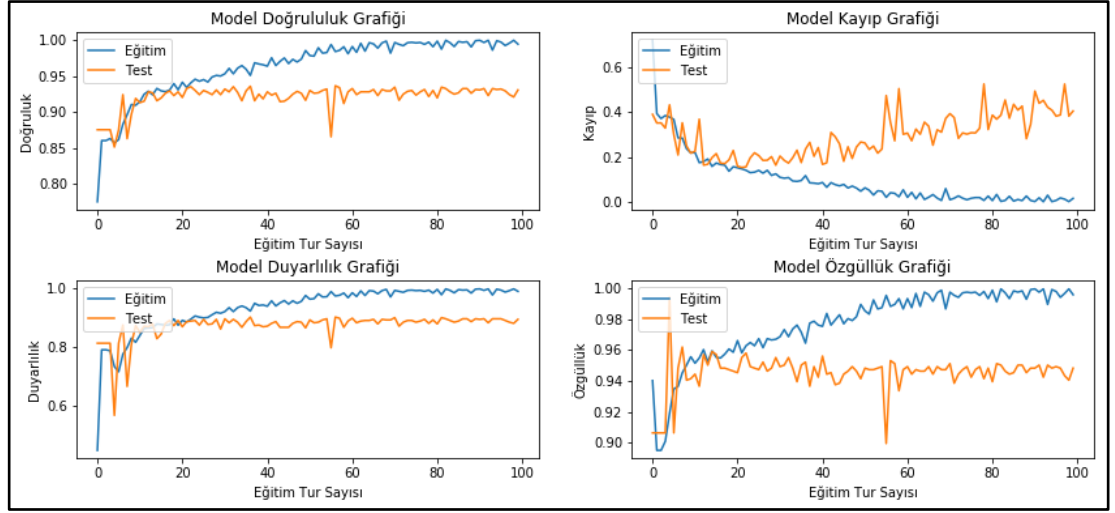
	100	87,52	81,29	90,64	87,52	81,29	90,64
Tanh-Sigmoid	20	91,49	95,38	94,54	91,49	85,38	95,54
	50	92,01	85,58	95,22	92,07	84,99	95,61
	100	91,75	87,33	93,96	91,75	87,33	93,96
Tanh-Softmax	20	89,60	84,02	92,40	92,85	88,11	95,22
	50	91,69	87,52	93,76	93,63	90,45	95,22
	100	91,16	86,74	93,37	89,73	84,60	92,30



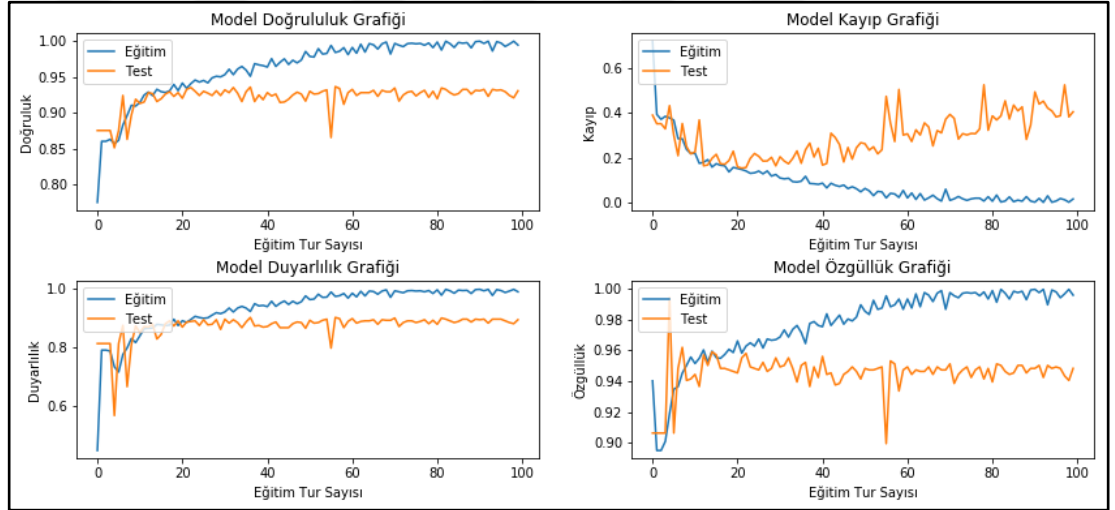
Şekil 52. RMSprop optimizier, Tanh-Softmax activation, 50 epoch, 128 batch size ile doğruluk (93,63) grafiği



Şekil 53. RMSprop optimizier, Relu-Sigmoid activation, 20 epoch, 64 batch size ile doğruluk (93,18) grafiği



Şekil 54. RMSprop optimizer, Relu-Softmax activation, 100 epoch, 128 batch size ile doğruluk (93,05) grafiği



Şekil 55. RMSprop optimizer, Relu-Softmax activation, 20 epoch, 64 batch size ile doğruluk (93,05) grafiği

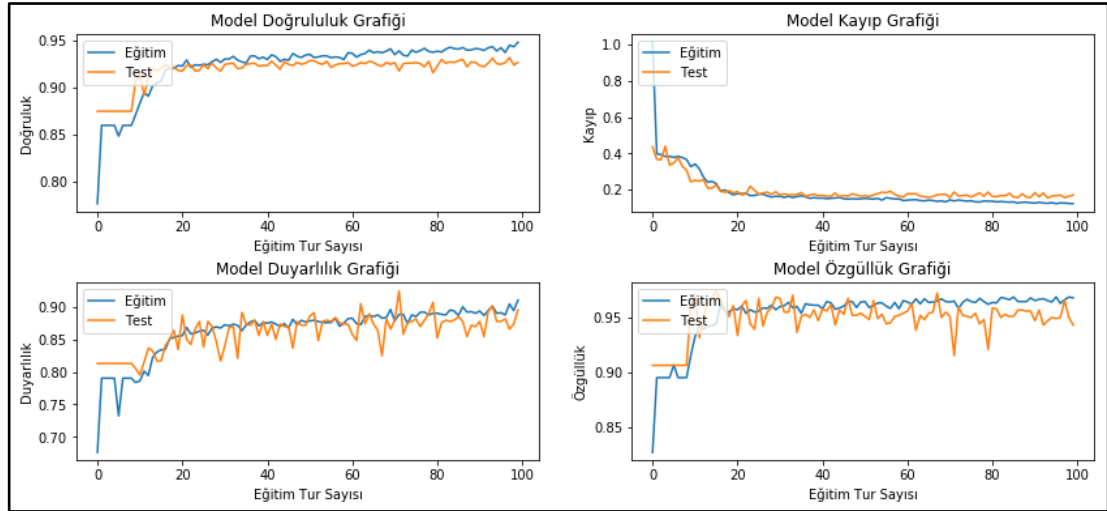
4.3. Adagrad Başarı Oranı

Çizelge 25’de Adagrad optimizasyon yöntemi kullanılarak elde edilmiş doğruluk, duyarlılık ve özgüllük değerleri yer almaktadır.

Çizelge 25. Adagrad optimizer ile test edilmiş eğitim doğruluk, duyarlılık ve özgüllük oranları

Aktivasyon	Epoch	Batch Size 64			Batch Size 128		
		Doğruluk	Duyarlılık	Özgüllük	Doğruluk	Duyarlılık	Özgüllük
Relu-Sigmoid	20	87,52	81,29	90,64	87,52	81,29	90,64
	50	87,52	81,29	90,64	87,52	81,29	90,64

	100	87,52	81,29	90,64	92,72	89,47	94,95
Relu-Softmax	20	87,52	81,29	90,64	87,52	81,29	90,64
	50	87,52	81,29	90,64	87,52	81,29	90,64
	100	87,52	81,29	90,64	87,52	81,29	90,64
Elu-Sigmoid	20	87,52	81,29	90,64	87,52	81,29	90,64
	50	87,52	81,29	90,64	87,52	81,29	90,64
	100	87,52	81,29	90,64	87,52	81,29	90,64
Elu-Softmax	20	87,52	81,29	90,64	87,52	81,29	90,64
	50	87,52	81,29	90,64	87,52	81,29	90,64
	100	87,52	81,29	90,64	87,52	81,29	90,64
Selu-Sigmoid	20	87,52	81,29	90,64	87,52	81,29	90,64
	50	87,52	81,29	90,64	87,52	81,29	90,64
	100	87,52	81,29	90,64	87,52	81,29	90,64
Selu-Softmax	20	87,52	81,29	90,64	87,52	81,29	90,64
	50	87,52	81,29	90,64	67,06	48,93	76,12
	100	87,52	81,29	90,64	87,52	81,29	90,64
Tanh-Sigmoid	20	87,52	81,29	90,64	87,52	81,29	90,64
	50	87,52	81,29	90,64	12,47	18,71	9,36
	100	87,52	81,29	90,64	12,47	18,71	9,36
Tanh-Softmax	20	40,09	10,14	55,07	87,52	81,29	90,64
	50	39,05	8,58	54,29	87,52	81,29	90,64
	100	39,05	8,58	54,29	87,52	81,29	90,64



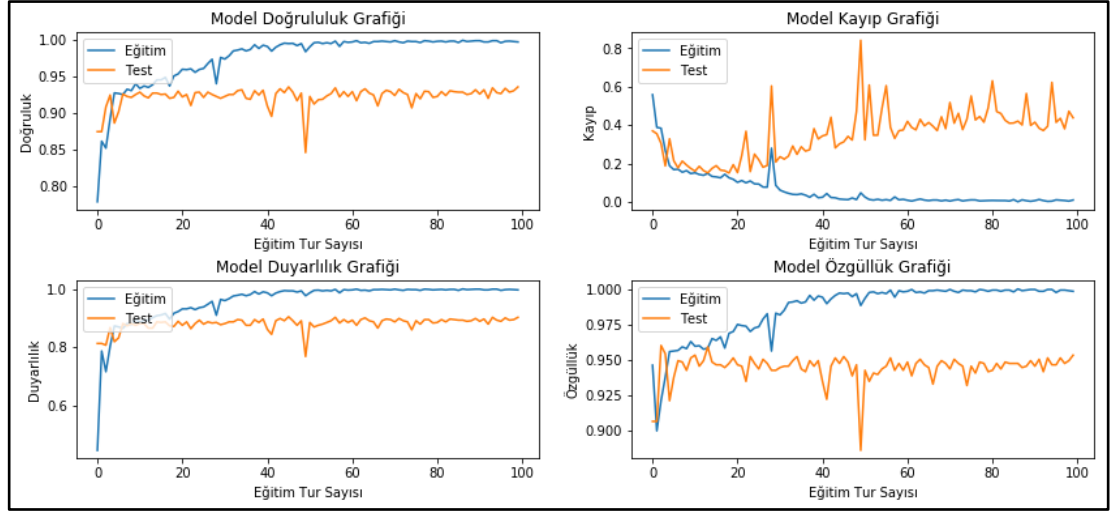
Şekil 56. Adagrad optimizier, Relu-Sigmoid activation, 100 epoch, 128 batch size ile doğruluk (92,72) grafiği

4.4. Adadelata Başarı Oranı

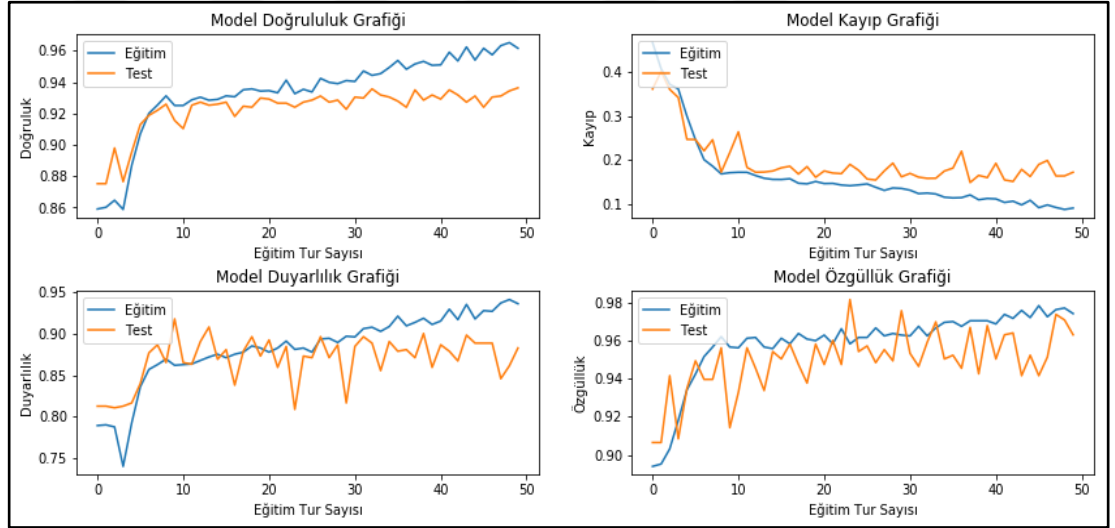
Çizelge 26’da Adadelata optimizasyon yöntemi kullanılarak elde edilmiş doğruluk, duyarlılık ve özgüllük değerleri yer almaktadır.

Çizelge 26. Adadelata optimizasyon ile test edilmiş eğitim doğruluk, duyarlılık ve özgüllük oranları

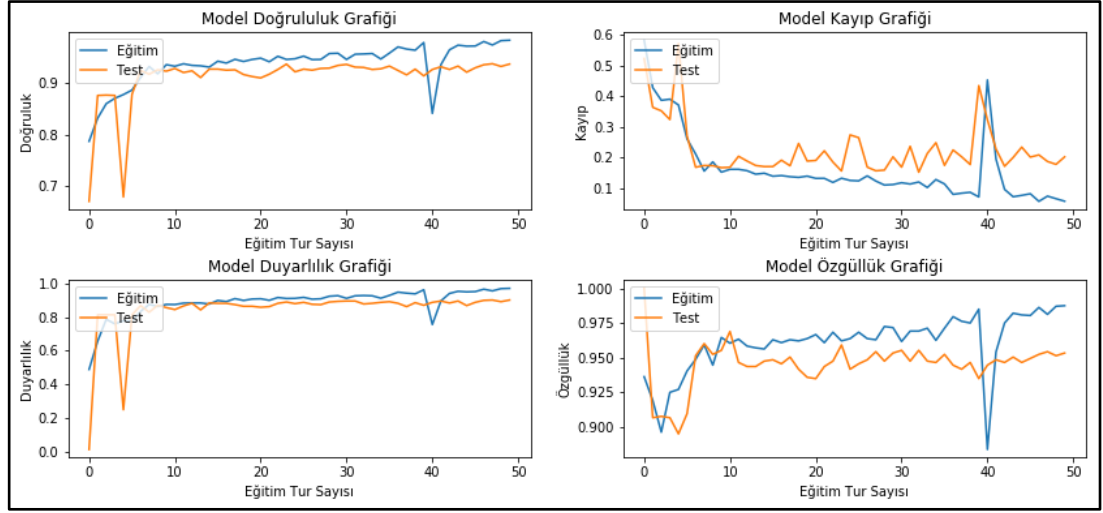
Aktivasyon	Epoch	Batch Size 64			Batch Size 128		
		Doğruluk	Duyarlılık	Özgüllük	Doğruluk	Duyarlılık	Özgüllük
Relu-Sigmoid	20	92,79	85,38	96,49	92,46	87,33	95,03
	50	93,63	88,30	96,30	93,11	87,52	95,91
	100	92,98	87,33	95,81	91,62	87,91	93,47
Relu-Softmax	20	92,92	85,77	96,49	90,97	86,16	93,37
	50	92,72	88,89	94,64	92,85	88,30	95,13
	100	92,66	88,89	94,54	92,85	88,69	94,93
Elu-Sigmoid	20	92,85	89,28	94,64	92,27	89,86	93,47
	50	92,01	85,77	95,13	92,53	85,77	95,91
	100	92,40	87,72	94,74	91,68	89,47	92,79
Elu-Softmax	20	91,81	86,74	94,35	92,66	88,30	94,83
	50	92,40	88,50	94,35	93,57	90,06	95,32
	100	93,63	90,25	95,32	92,85	89,28	94,64
Selu-Sigmoid	20	91,68	88,69	93,17	91,49	93,18	90,64
	50	92,92	89,47	94,64	92,33	87,91	94,54
	100	87,52	81,29	90,64	87,52	81,29	90,64
Selu-Softmax	20	87,52	81,29	90,64	87,52	81,29	90,64
	50	87,52	81,29	90,64	87,52	81,29	90,64
	100	87,52	81,29	90,64	39,05	8,58	54,29
Tanh-Sigmoid	20	93,18	85,96	96,78	92,92	88,11	95,32
	50	92,33	86,35	95,32	92,01	88,69	93,66
	100	93,44	89,08	95,61	92,66	87,52	95,22
Tanh-Softmax	20	92,85	88,89	94,83	92,66	88,50	94,74
	50	92,59	88,89	94,44	92,27	88,30	94,25
	100	91,94	87,72	94,05	92,14	88,11	94,15



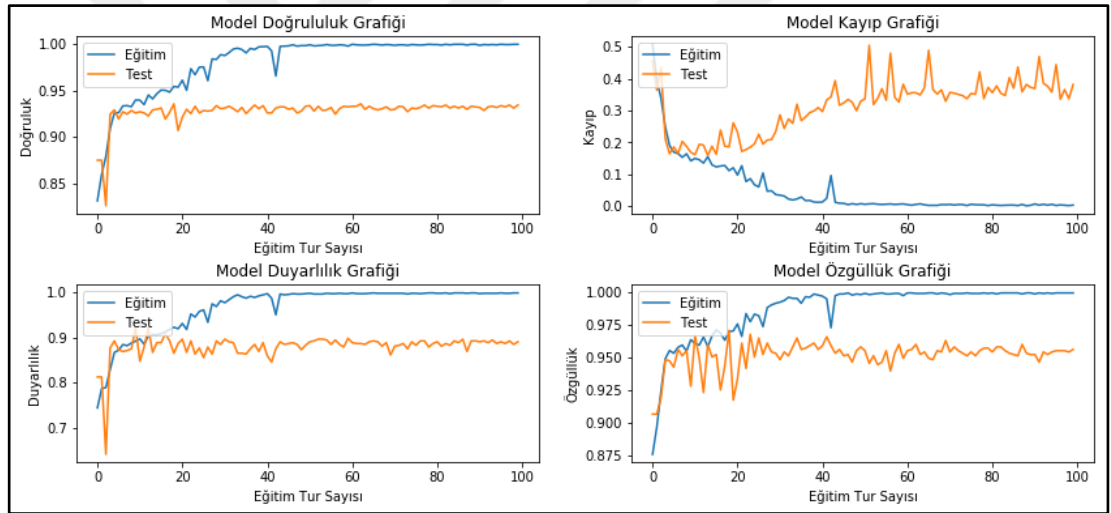
Şekil 57. Adadelata optimizier, Elu-Softmax activation, 100 epoch, 64 batch size ile doğruluk (93,63) grafiği



Şekil 58. Adadelata optimizier, Relu-Sigmoid activation, 50 epoch, 64 batch size ile doğruluk (93,63) grafiği



Şekil 59. Adadelta optimizier, Elu-Sigmoid activation, 50 epoch, 128 batch size ile doğruluk (93,57) grafiği



Şekil 60. Adadelta optimizier, Tanh-Sigmoid activation, 100 epoch, 64 batch size ile doğruluk (93,44) grafiği

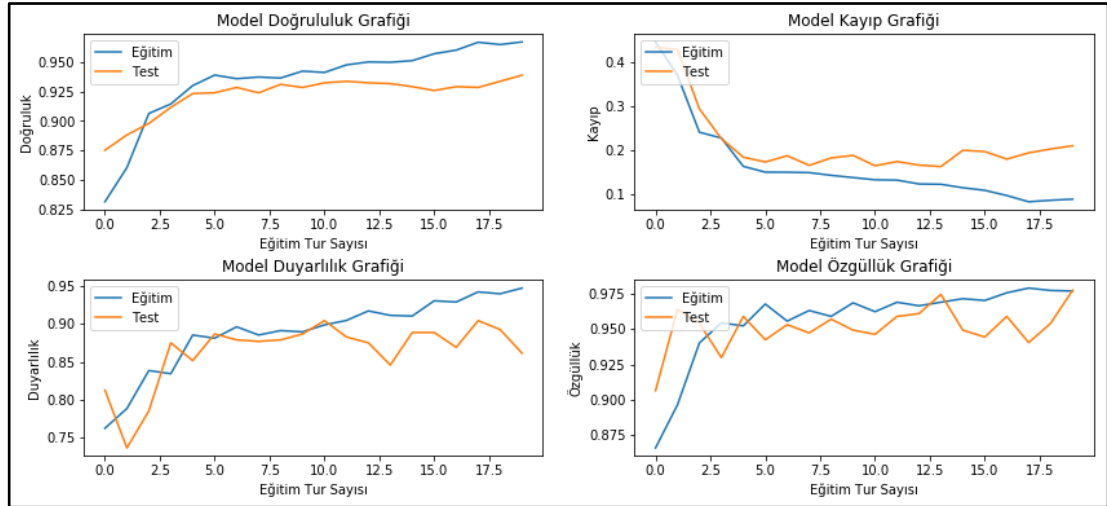
4.5. Adam Başarı Oranı

Çizelge 27’de Adam optimizasyon yöntemi kullanılarak elde edilmiş doğruluk, duyarlılık ve özgüllük değerleri yer almaktadır.

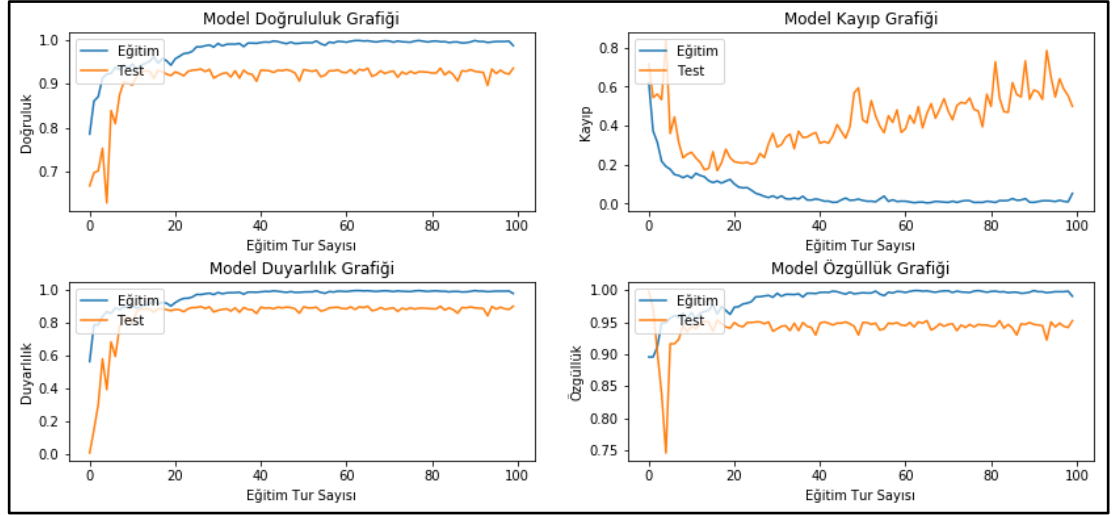
Çizelge 27. Adam optimizier ile test edilmiş eğitim doğruluk, duyarlılık ve özgüllük oranları

Aktivasyon	Epoch	Batch Size 64			Batch Size 128		
		Doğruluk	Duyarlılık	Özgüllük	Doğruluk	Duyarlılık	Özgüllük
Relu-Sigmoid	20	93,05	90,84	94,15	92,59	87,52	95,13
	50	92,27	89,67	93,58	93,50	91,62	94,44

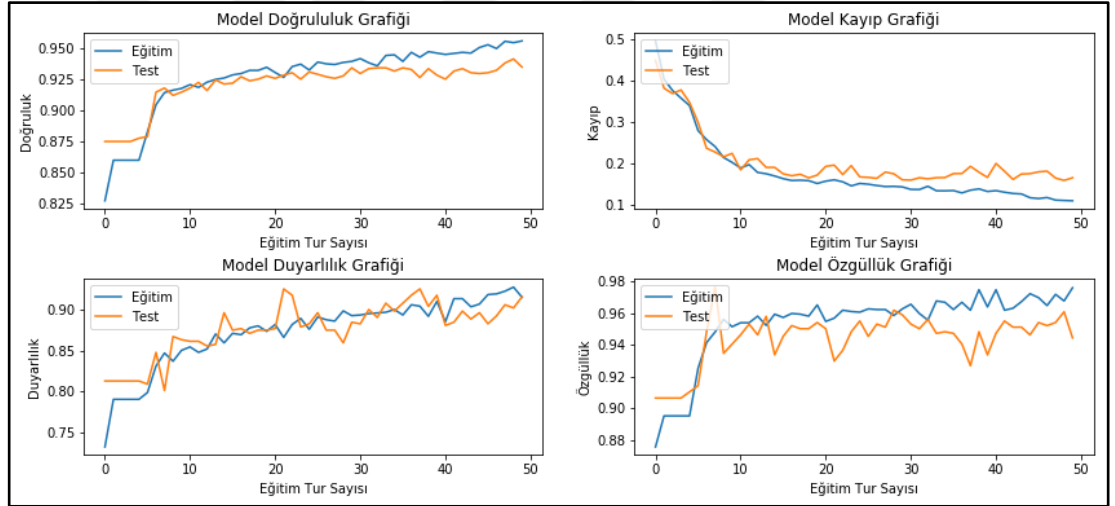
	100	92,72	87,33	95,42	93,24	89,28	95,22
Relu-Softmax	20	92,72	88,69	94,74	92,46	86,74	95,32
	50	92,92	88,50	95,13	92,79	88,50	94,93
	100	92,66	88,69	94,64	92,27	88,30	94,25
Elu-Sigmoid	20	93,89	86,16	97,76	92,46	87,72	94,83
	50	92,85	85,58	96,49	92,79	86,94	95,71
	100	92,27	85,77	95,52	92,59	86,55	95,61
Elu-Softmax	20	92,40	88,30	94,44	93,24	89,08	95,32
	50	93,11	89,67	94,83	92,27	88,30	94,25
	100	92,98	89,28	94,83	91,62	87,33	93,76
Selu-Sigmoid	20	92,66	88,89	94,54	93,18	92,20	93,66
	50	91,42	88,50	92,88	91,94	87,52	94,15
	100	92,66	88,50	94,74	92,85	86,16	96,20
Selu-Softmax	20	92,59	88,89	94,44	93,24	89,47	95,13
	50	90,58	85,77	92,98	93,24	89,67	95,03
	100	87,52	81,29	90,64	93,57	90,25	95,22
Tanh-Sigmoid	20	92,79	88,30	95,03	92,92	90,06	94,35
	50	92,85	87,91	95,32	92,92	88,30	95,22
	100	92,07	84,60	95,81	92,33	86,55	95,22
Tanh-Softmax	20	93,18	89,47	95,03	92,40	88,50	94,35
	50	93,05	89,47	94,83	92,33	88,50	94,25
	100	92,46	88,69	94,35	91,81	87,52	93,96



Şekil 61. Adam optimizier, Elu-Sigmoid activation, 20 epoch, 64 batch size ile doğruluk (93,89) grafiği



Şekil 62. Adam optimizer, Selu-Softmax activation, 100 epoch, 128 batch size ile doğruluk (93,57) grafiği



Şekil 63. Adam optimizer, Relu-Sigmoid activation, 50 epoch, 128 batch size ile doğruluk (93,50) grafiği

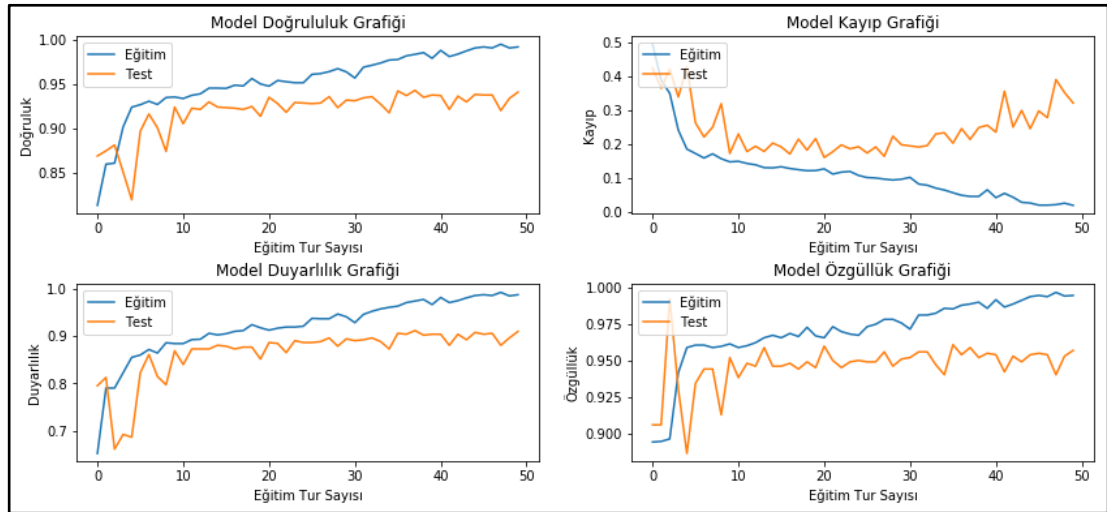
4.6. Adamax Başarı Oranı

Çizelge 28’de Adamax optimizasyon yöntemi kullanılarak elde edilmiş doğruluk, duyarlılık ve özgüllük değerleri yer almaktadır.

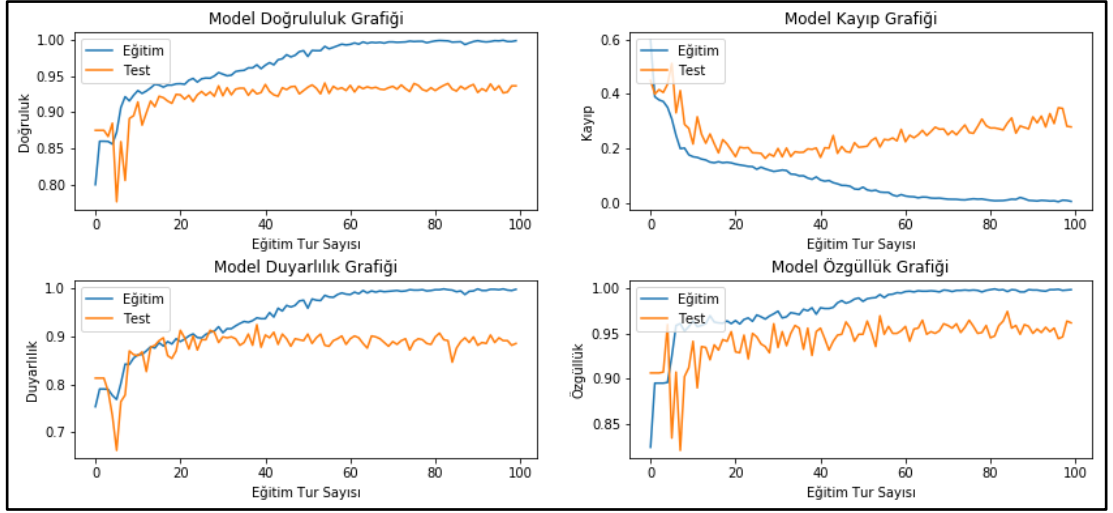
Çizelge 28. Adamax optimizer ile test edilmiş eğitim doğruluk, duyarlılık ve özgüllük oranları

Aktivasyon	Epoch	Batch Size 64			Batch Size 128		
		Doğruluk	Duyarlılık	Özgüllük	Doğruluk	Duyarlılık	Özgüllük
Relu-Sigmoid	20	92,07	87,33	94,44	92,33	87,13	94,93

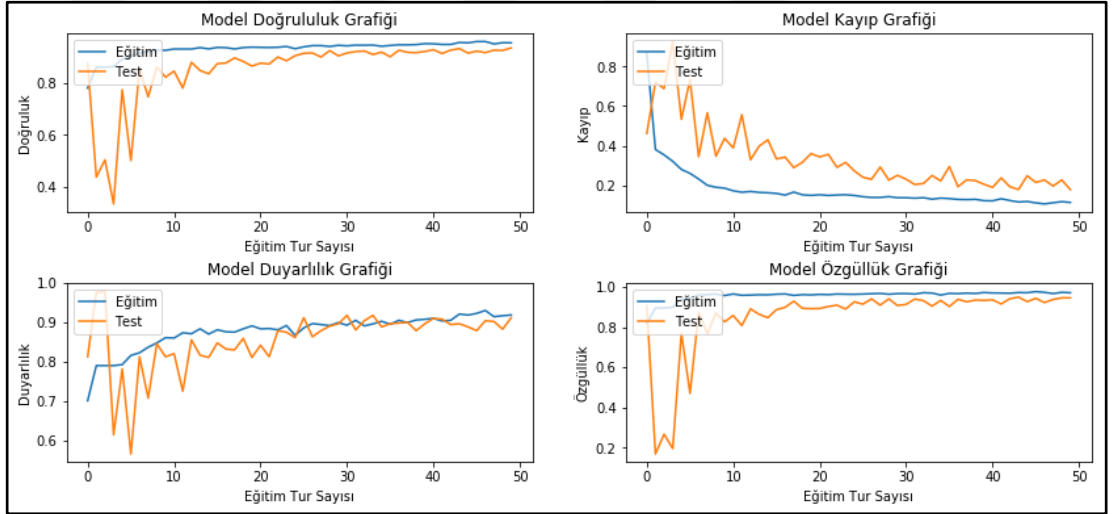
	50	92,53	88,69	94,44	92,79	91,42	93,47
	100	93,31	88,89	95,52	92,66	88,30	94,83
Relu-Softmax	20	92,14	87,33	94,54	90,06	82,46	93,86
	50	93,37	88,69	95,71	93,37	88,50	95,81
	100	93,24	89,86	94,93	92,40	88,30	94,44
Elu-Sigmoid	20	92,40	87,52	94,83	92,40	87,91	94,64
	50	92,72	89,29	94,44	92,40	90,25	93,47
	100	92,14	88,30	94,05	92,59	88,30	94,74
Elu-Softmax	20	91,55	86,35	94,15	90,38	84,41	93,37
	50	94,15	91,03	95,71	93,24	89,28	95,22
	100	92,27	88,30	94,25	92,46	88,50	94,44
Selu-Sigmoid	20	89,93	89,08	90,35	70,57	75,05	68,32
	50	93,44	91,23	94,54	89,02	91,23	87,91
	100	92,85	88,30	95,13	92,79	90,06	94,15
Selu-Softmax	20	87,72	78,17	92,50	87,52	81,29	90,64
	50	87,52	81,29	90,64	87,52	81,29	90,64
	100	87,52	81,29	90,64	92,20	88,11	94,25
Tanh-Sigmoid	20	92,66	91,23	93,37	91,36	85,38	94,35
	50	93,11	88,30	95,52	93,18	88,69	95,42
	100	93,63	88,50	96,20	93,11	88,69	95,32
Tanh-Softmax	20	92,72	88,11	95,03	89,47	81,68	93,37
	50	92,85	89,08	94,74	93,11	89,47	94,93
	100	93,44	90,06	95,13	91,62	87,33	93,77



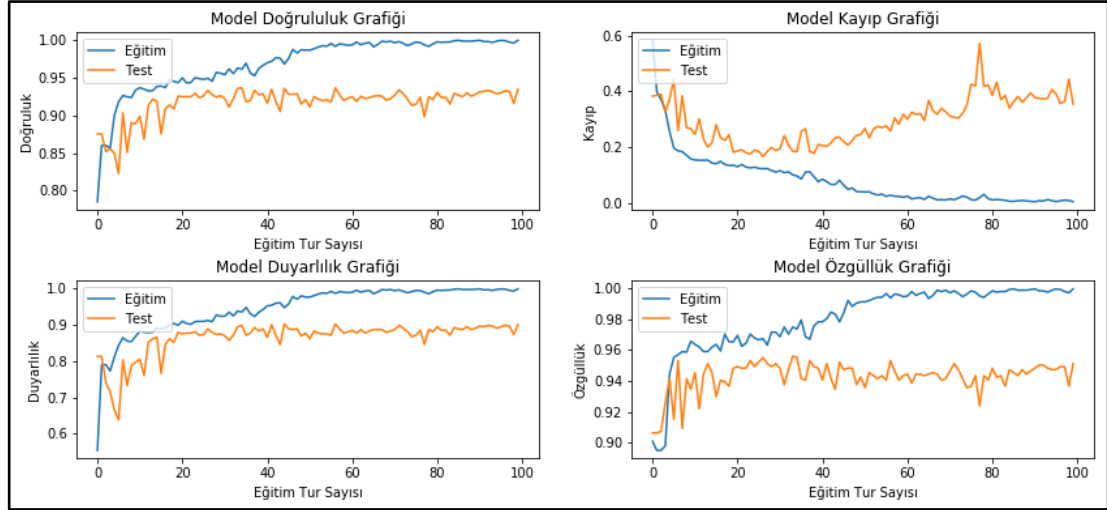
Şekil 64. Adamax optimizier, Elu-Softmax activation, 50 epoch, 64 batch size ile doğruluk (94,15) grafiği



Şekil 65. Adamax optimizier, Tanh-Sigmoid activation, 100 epoch, 64 batch size ile doğruluk (93,63) grafiği



Şekil 66. Adamax optimizier, Selu-Sigmoid activation, 50 epoch, 64 batch size ile doğruluk (93,44) grafiği



Şekil 67. Adamax optimizier, Tanh-Softmax activation, 100 epoch, 64 batch size ile doğruluk (93,44) grafiği

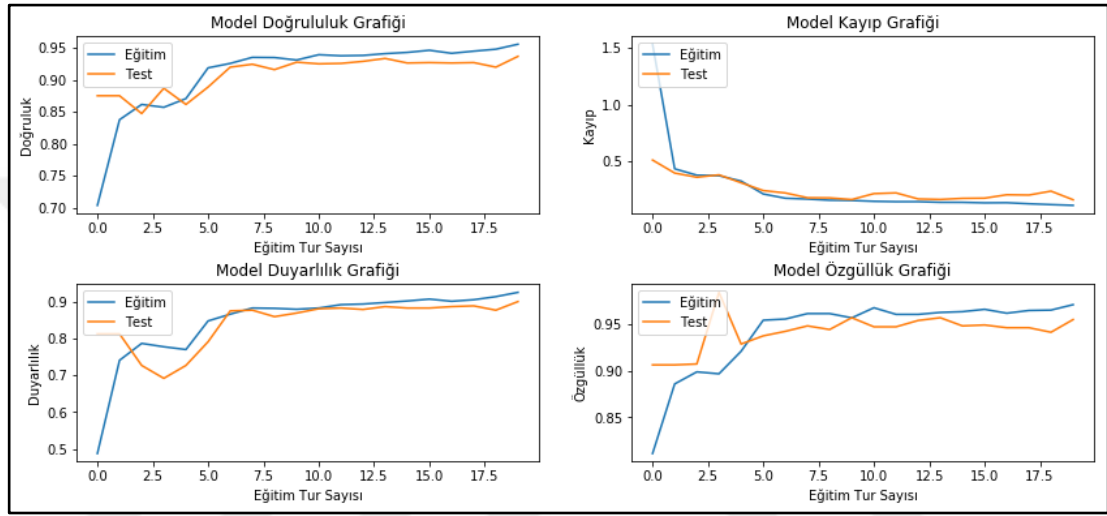
4.7. Nadam Başarı Oranı

Çizelge 29’da Nadam optimizasyon yöntemi kullanılarak elde edilmiş doğruluk, duyarlılık ve özgüllük değerleri yer almaktadır.

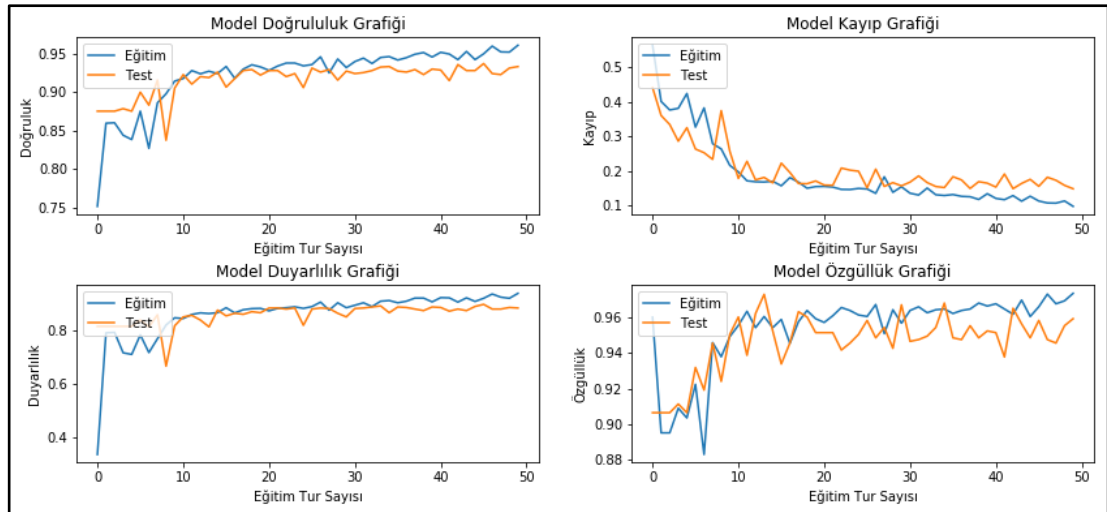
Çizelge 29. Nadam optimizier ile test edilmiş eğitim doğruluk, duyarlılık ve özgüllük oranları

Aktivasyon	Epoch	Batch Size 64			Batch Size 128		
		Doğruluk	Duyarlılık	Özgüllük	Doğruluk	Duyarlılık	Özgüllük
Relu-Sigmoid	20	93,24	91,03	94,35	91,62	90,25	92,30
	50	92,98	87,52	95,71	92,79	88,50	94,93
	100	91,62	86,16	94,35	92,72	89,67	94,25
Relu-Softmax	20	91,55	85,19	94,74	92,53	85,96	95,81
	50	92,72	88,89	94,64	93,31	88,11	95,91
	100	92,46	88,30	94,54	93,05	89,47	94,83
Elu-Sigmoid	20	87,52	81,29	90,64	87,52	81,29	90,64
	50	87,52	81,29	90,64	87,52	81,29	90,64
	100	87,52	81,29	90,64	87,52	81,29	90,64
Elu-Softmax	20	87,52	81,29	90,64	87,52	81,29	90,64
	50	87,52	81,29	90,64	87,52	81,29	90,64
	100	87,52	81,29	90,64	87,52	81,29	90,64
Selu-Sigmoid	20	87,52	81,29	90,64	87,52	81,29	90,64
	50	87,52	81,29	90,64	87,52	81,29	90,64
	100	87,52	81,29	90,64	87,52	81,29	90,64
Selu-Softmax	20	87,52	81,29	90,64	87,52	81,29	90,64
	50	87,52	81,29	90,64	87,52	81,29	90,64
	100	87,52	81,29	90,64	87,52	81,29	90,64

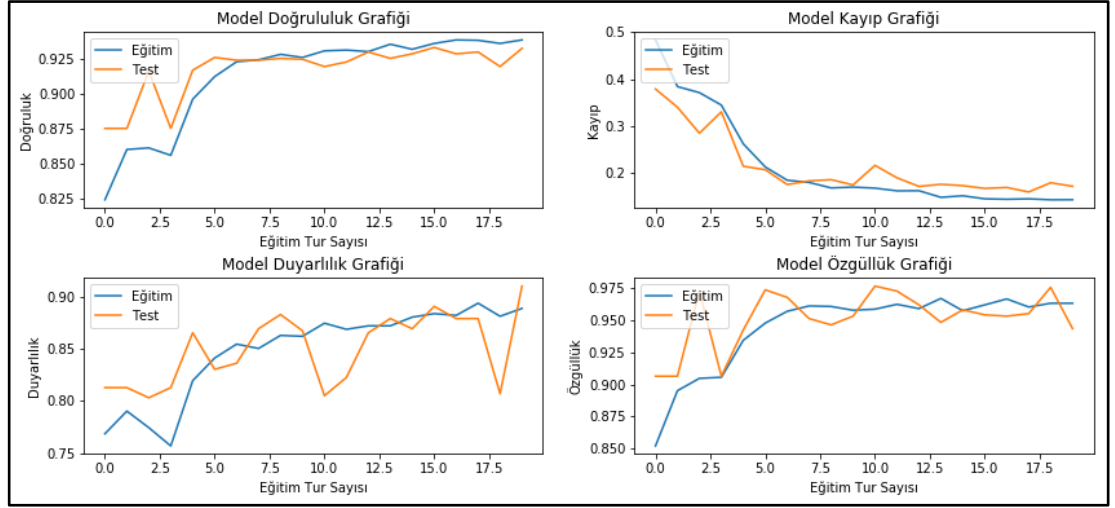
	50	87,52	81,29	90,64	87,52	81,29	90,64
	100	87,52	81,29	90,64	87,52	81,29	90,64
Tanh-Sigmoid	20	60,95	91,42	45,71	92,79	92,01	93,18
	50	87,52	81,29	90,64	87,52	81,29	90,64
	100	87,52	81,29	90,64	87,52	81,29	90,64
Tanh-Softmax	20	93,70	90,06	95,52	87,52	81,29	90,64
	50	87,52	81,29	90,64	87,52	81,29	90,64
	100	40,09	10,14	55,07	39,05	8,58	54,29



Şekil 68. Nadam optimizier, Tanh-Softmax activation, 20 epoch, 64 batch size ile doğruluk (93,70) grafiği



Şekil 69. Nadam optimizier, Relu-Softmax activation, 50 epoch, 128 batch size ile doğruluk (93,31) grafiği



Şekil 70. Nadam optimizier, Relu-Sigmoid activation, 20 epoch, 64 batch size ile doğruluk (93,24) grafiği

5. SONUÇ, TARTIŞMA ve ÖNERİLER

Bu bölümde araştırma bulguları genel olarak özetlenerek bir sonuca bağlanmıştır. Ayrıca alan yazında yapılmış çalışmaların sonuçları ve bu tez çalışmasında elde edilen sonuçlar tartışılmıştır.

5.1. Sonuç

SGD optimizasyon yöntemi sonuçları incelendiğinde yapılan eğitim işlemlerinde en iyi sonuçları elde etmede Relu-Softmax aktivasyon fonksiyonlarının başarılı olduğu görülmektedir. Epoch sayısı olarak 50 ve 100 değerleri, Batch size sayısı olarak ise 128 kullanıldığında daha başarılı sonuçlar elde edildiği gözlemlenmiştir. En iyi sonuç Relu-Softmax aktivasyon yönteminde 100 epoch ve 128 batch size değerleri kullanıldığında %92,46 doğruluk, %86,35 duyarlılık ve %95,52 özgüllük oranı ile elde edilmiştir.

RMSprop optimizasyon yöntemi sonuçları incelendiğinde yapılan eğitim işlemlerinde Relu aktivasyon fonksiyonu ile yüksek doğruluk değerleri elde edilmesine rağmen en iyi sonucu Tanh-Softmax aktivasyon yöntemi ile alındığı görülmektedir. En iyi sonuç Tanh-Softmax aktivasyon yönteminde 50 epoch ve 128 batch size değerleri kullanıldığında %93,63 doğruluk, %90,45 duyarlılık ve %95,22 özgüllük oranı ile elde edilmiştir.

Adagrad yöntemi görüntü işlemede tercih edilen bir yöntem olmamakla birlikte yapılan değerlendirme işlemlerinde ezberleme durumunun çok fazla olduğu görülmüştür. En iyi sonuç Relu-Sigmoid aktivasyon yönteminde 100 epoch ve 128 batch size değerleri kullanıldığında %92,72 doğruluk, %89,47 duyarlılık ve %94,95 özgüllük oranı ile elde edilmiştir.

Adadelat optimizasyon yöntemi incelendiğinde Elu-Softmax aktivasyon fonksiyonları ile daha başarılı sonuçlar alındığı görülmüştür. Epoch sayısı olarak 50 ve 100 değerlerinin, Batch size sayısı olarak ise 64 kullanıldığında daha başarılı sonuçlar elde edildiği gözlemlenmiştir. En iyi sonuçlar Elu-Softmax aktivasyon fonksiyonunda 100 epoch ve 64 batch size değerleri kullanıldığında %93,63 doğruluk, %90,25 duyarlılık ve %95,32 özgüllük oranı ile ayrıca Relu-Sigmoid aktivasyon fonksiyonunda 50 epoch ve 64 batch size değerleri kullanıldığında %93,63 doğruluk, %88,30 duyarlılık ve %96,30 özgüllük oranı ile elde edilmiştir.

Adam optimizasyon yöntemi incelendiğinde Elu, Selu ve Relu aktivasyon fonksiyonlarında başarılı sonuçlar elde edildiği görülmüştür. Batch size değeri olarak 128 kullanıldığında daha başarılı sonuçlar elde edildiği gözlemlenmiştir. En iyi sonuç Elu-Sigmoid aktivasyon fonksiyonunda 20 epoch ve 64 batch size değerleri kullanıldığında %93,89 doğruluk, %86,16 duyarlılık ve %97,76 özgüllük oranı ile elde edilmiştir.

Adamax optimizasyon yöntemi incelendiğinde Elu, Tanh ve Selu aktivasyon fonksiyonları ile daha başarılı sonuçlar alındığı görülmüştür. Epoch sayısı olarak 50 ve 100 değerlerinin, batch size sayısı olarak ise 64 kullanıldığında daha başarılı sonuçlar elde edildiği gözlemlenmiştir. En iyi sonuç Elu-Softmax aktivasyon fonksiyonunda 50 epoch ve 64 batch size değerleri kullanıldığında %94,15 doğruluk, %91,03 duyarlılık ve %95,71 özgüllük oranı ile elde edilmiştir.

Nadam optimizasyon yöntemi incelendiğinde Relu ve Tanh aktivasyon fonksiyonlarında daha başarılı sonuçlar alındığı görülmüştür. Epoch sayısı olarak 20 ve 50 değerlerinin, batch size sayısı olarak ise 64 kullanıldığında daha başarılı sonuçlar elde edildiği gözlemlenmiştir. En iyi sonuç Tanh-Softmax aktivasyon fonksiyonunda 20 epoch ve 64 batch size değerleri kullanıldığında %93,70 doğruluk, %90,06 duyarlılık ve %95,52 özgüllük oranı ile elde edilmiştir.

Çizelge 30. Hastalık teşhisi başarı durum sıralaması

Optimizer	Aktivasyon	Epoch	Batch Size	Doğruluk (%)
SGD	Relu-Softmax	100	128	92,46
	Relu-Softmax	50	128	92,33
	Relu-Softmax	50	64	92,27
RMSprop	Tanh-Softmax	50	128	93,63
	Relu-Sigmoid	20	64	93,18
	Relu-Softmax	100	128	93,05
	Relu-Softmax	20	64	93,05
Adagrad	Relu-Sigmoid	100	128	92,72
Adadelat	Elu-Softmax	100	64	93,63
	Relu-Sigmoid	50	64	93,63
	Elu-Softmax	50	128	93,57

	Tanh-Sigmoid	100	64	93,44
Adam	Elu-Sigmoid	20	64	93,89
	Selu-Softmax	100	128	93,57
	Relu-Sigmoid	50	128	93,50
Adamax	Elu-Softmax	50	64	94,15
	Tanh-Sigmoid	100	64	93,63
	Selu-Sigmoid	50	64	93,44
	Tanh-Softmax	100	64	93,44
Nadam	Tanh-Softmax	20	64	93,70
	Relu-Softmax	50	128	93,31
	Relu-Sigmoid	20	64	93,24

Optimizasyon yöntemleri için ayrı ayrı başarı durumlarının geneline bakıldığında ise en yüksek doğruluk oranına sahip 3 yöntemin değerleri Çizelge 31’de gösterilmiştir. Genel durumun verildiği Çizelge 30’da %90 doğruluk oranı üstündeki değerler kontrol edildiğinde Relu aktivasyon fonksiyonu ile iyi sonuçlar alınmasına rağmen en iyi 3 başarı durumu sıralamasında Relu aktivasyon fonksiyonu yer almamaktadır. Ayrıca batch size değeri olarak en iyi 3 sonucun 64 değeri ile alındığı görülmektedir. Epoch sayısı olarak bakıldığında ise en iyi sonuçlar 20 ve 50 epoch değerleri kullanılarak elde edilmiştir.

Çizelge 31. Hastalık teşhisi en iyi 3 başarı durum sıralaması

Optimizier	Aktivasyon	Epoch	Batch Size	Doğruluk (%)
Adamax	Elu-Softmax	50	64	94,15
Adam	Elu-Sigmoid	20	64	93,89
Nadam	Tanh-Softmax	20	64	93,70

5.2. Tartışma

Alan yazın incelendiğinde Parkinson hastalığı teşhisi üzerine yapılmış pek çok makale karşımıza çıkmaktadır. Bu makalelerden bazılarında görüntü işleme temelli, bazılarında ise hastalık teşhisinde kullanılabilecek farklı veri grupları üzerinden hastalık teşhisi yapıldığı görülmektedir.

Zhang and Kagen (2017), PPMI veri tabanından 3 grup (PD, Control, SWEDD) görüntü dosyasından control ve swedd gruplarını sağlıklı birey olarak gruplandırmış ve sonuçta PD ve Control olarak iki grup altında, SPECT görüntülerden 41. kesiti kullanarak makine öğrenmesi uygulaması ile hastalık teşhisi yapmışlardır. Gruplandırma için Gradient descent, adagrad optimizasyon algoritmalarını ve 1000 epoch değeri kullanılarak %93,8 doğruluk oranında doğru teşhis yapmışlardır. Bu tez çalışmasında PD, Control ve Swedd görüntülerin her biri farklı bir grup olarak ele alınarak Keras kütüphanesi altından kullanıma sunulan 7 farklı optimizasyon yöntemi kullanılarak hastalık teşhisi yapılmıştır.

Prashanth et al. (2016) yaptıkları çalışmada ise UPSIT – Koku Tanımlama Testi, Uyku Davranış Bozukluğu Tarama Anketi ve Beyin omurilik sıvısı değerlerinden oluşan veri kümelerinden hastalık teşhisi yapmışlardır. Sınıflandırma yöntemi olarak SVM, boosted trees, random forest ve naive bayes olarak isimlendirilen makine öğrenmesi algoritmalarını kullanarak 100 epoch değeri ile %96 doğruluk oranına ulaşmışlardır.

Rana et al. (2015) yaptıkları çalışmada Parkinson hastası ve sağlıklı bireylerden elde edilen MRI görüntüleri üzerinden hastalık teşhisi yapmışlardır. Sınıflandırma işlemi yapılırken VV-only ve VV-GBSFS metotlarını kullanmışlardır. Yaptıkları çalışma sonucu %86,67 oranında başarı sağlamışlardır.

Oliveira and Branco (2015) ise PPMI veri tabanından alınan Parkinson hastası ve sağlıklı bireylere ait Spect görüntüler üzerinden, makine öğrenmesi algoritmalarından SVM ile bilgisayar destekli tanı işlemi yapılarak %97,86 oranında başarı elde ettiler.

SPECT görüntülerin kullanıldığı çalışmalarda genel olarak Parkinson hastası ve sağlıklı bireylerin kayıtları tercih edildiği görülmektedir. Sınıflandırma yöntemleri olarak makine öğrenmesi algoritmaları öne çıkmaktadır. Bu tez çalışmasında ise PPMI veri tabanında bulunan Parkinson hastası, sağlıklı birey ve Swedd değerlerinin hepsi alınarak derin öğrenme çalışmaları gerçekleştirilmiştir. Ayrıca 7 farklı optimizasyon yöntemi ile detaylı bir çalışma yapılmıştır. Alan yazında 100 ve 1000 gibi epoch değerleri ile çalışmalar yapılmıştır oysa bu çalışmada daha az epoch değerleri üzerinde testler gerçekleştirilmiş ve sonuçlar aktarılmıştır.

5.3. Öneriler

Yapay zekâ ve onun alt alanı olan derin öğrenme teknolojileri büyük bir hızla gelişmeye devam etmektedir. Özellikle derin öğrenme konusunda son yıllarda pek çok başarılı çalışma gerçekleştirilmiştir. Bu tez çalışmasında kullanılan Tensorflow kütüphanesi gibi öğrenme yapılarının başarımlarının daha yukarılara çıkacağı ön görülmektedir. Bu teknoloji benzer semptomlar ve belirsizliklerin hâkim olduğu zor hastalıkların teşhisinde başarılı bir şekilde kullanılabilir. Bu nedenle hastalıklarla ilgili veri setleri üzerinde yeni derin öğrenme kütüphaneleri ve metotları kullanılarak daha yüksek doğruluk oranları ile bu hastalıklar erken teşhis edilebilir.

Bu tez çalışmasında elde edilen en başarılı model yapısının katmanları ilerde geliştirilerek Tensorflow kütüphanesi üzerinden doğruluk katsayıları daha da yüksek seviyelere çıkarılabilir.

Veri sayısını artırma adına hastanelerde bulunan Nükleer Tıp Görüntüleme merkezlerinden alından farklı veri setleri PPMI üzerinden indirilen veri setine entegre edilerek daha başarılı sınıflandırma oranlarına ulaşılabilir.

Doktorlar, bugün hastane bilgi yönetim sistemleri aracılığıyla hastalarının tomografi, MR ve SPECT gibi görüntü verilerine kolayca ulaşabilmektedir. Bu tez çalışması kapsamında derin öğrenme ile elde edilen en başarılı sınıflama modelleri bu bilgi sistemlerine entegre edilerek SPECT görüntülerin başarılı bir şekilde sınıflandırılması mümkün olabilir. Bu sayede hekimlerin Parkinson hastalığını daha kolay ve doğru bir şekilde teşhis edebilmelerine yardımcı olarak, onların kararlarını destekleyecek akıllı araçlar sisteme dâhil edilebilir.

Ayrıca bu akıllı sistem yeni gelen vakalar ve doktorun teşhis koyduğu hastalar ile öğrenmesini pekiştirerek daha da mükemmel bir seviyeye ulaşabilir. Bunun için hastaların SPECT görüntüleri sisteme dahil edilerek doğruluk katsayısının artırılmasını mümkün kılan yapı hazırlanabilir ve görüntülerin daha yüksek oranlarda sınıflandırılması sağlanabilir.

KAYNAKLAR

- Akbayır, E., Şen, M., Ay, U., Şenyer, S., Tüzün, E., ve Küçükali, C. İ. (2017). Parkinson Hastalığının Etyopatogenezi. Deneysel Tıp Araştırma Enstitüsü Dergisi, 7(13), 1-23.
- Akpınar, H. (2017), *Data Veri Madenciliği Veri Analizi*, İstanbul: Papatya Yayıncılık.
- Aksoy, F. (2017), *Yeni Başlayanlar İçin Python*, İstanbul: Abaküs Yayıncılık.
- Alpaydın, E. (2018), *Yapay Öğrenme*, İstanbul: Boğaziçi Üniversitesi Yayınevi.
- Anwer, A. M. O. (2017). *Derin Öğrenme Yöntemleri İle Göğüs Kanseri Teşhisi*, Yüksek Lisans Tezi, Türk Hava Kurumu Üniversitesi, Fen Bilimleri Enstitüsü, Ankara.
- Arganda-Carreras, I., Kaynig, V., Rueden, C., Eliceiri, K. W., Schindelin, J., Cardona, A., and Sebastian Seung, H. (2017). Trainable Weka Segmentation: a machine learning tool for microscopy pixel classification. Bioinformatics, 33(15): 2424-2426.
- Arslan, İ. (2019), *Python ile Veri Bilimi*, İstanbul: Pusula Yayıncılık.
- Aygün, R. C. (2017). *Derin Öğrenme Yöntemleri İle Bilgisayar Ağlarında Güvenliğe Yönelik Anormallik Tespiti*, Yüksek Lisans Tezi, Yıldız Teknik Üniversitesi, Fen Bilimleri Enstitüsü, İstanbul.
- Bilgin, M. (2018), *Makine Öğrenmesi*, İstanbul: Papatya Yayıncılık.
- Brandt, P. (2017). How Machine Learning with TensorFlow Enabled Mobile Proof-Of-Purchase at Coca-Cola. <https://developers.googleblog.com/2017/09/how-machine-learning-with-tensorflow.html> (18.04.2019).
- Budhiraja, A. (2016). Dropout in (Deep) Machine learning. <https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5> (20.04.2019).
- Chen, Y., Lin, Z., Zhao, X., Wang, G., and Gu, Y. (2014). Deep learning-based classification of hyperspectral data. IEEE Journal of Selected topics in applied earth observations and remote sensing, 7(6): 2094-2107.
- Çarkacı, N., 2018, Derin Öğrenme Uygulamalarında En Sık kullanılan Hiper-parametreler. <https://medium.com/deep-learning-turkiye/derin-ogrenme-uygulamalarinda-en-sik-kullanilan-hiper-parametreler-ece8e9125c4> (19.04.2019).

- Çevik, K. K., ve Dandıl, E. (2012). Yapay Sinir Ağları İçin. NET Platformunda Görsel Bir Eğitim Yazılımının Geliştirilmesi. Bilişim Teknolojileri Dergisi, 5(1): 19-28.
- Dener, M., Dörterler, M., ve Orman, A. (2009). Açık kaynak kodlu veri madenciliği programları: WEKA’da örnek uygulama. Akademik Bilişim, 9: 11-13.
- Doğan, O. (2017). Ücretsiz Veri Madenciliği Araçları ve Türkiye’de Bilinirlikleri Üzerine Bir Araştırma. Ege Stratejik Araştırmalar Dergisi. 8(1): 77-93.
- Erdem, O. (2007). Kavşak Trafik Sinyalizasyon Sistemi İçin Bulanık Mantık Tabanlı Gerçek Zamanlı Denetleyici Tasarımı ve Uygulaması. Engineering Sciences, 2(4): 241-255.
- Elitez O. (2015). *Derin Öğrenme Yöntemi Kullanılarak El Yazısı Rakam Dizileri Bölütleme ve Tanıma*, Yüksek Lisans Tezi, Orta Doğu Teknik Üniversitesi, Doğal ve Uygulamalı Bilimleri Enstitüsü, İstanbul.
- Güçlü Gündüz, A., Otman, A.S., Köse, N., Bilgin, S., ve Elibol, B. (2009), Parkinson hastalığında farklı denge ölççeklerinin karşılaştırılması, Fizyoterapi Rehabilitasyon, 20(1):17-24.
- Gürsakal, N. (2017), *Makine Öğrenmesi ve Derin Öğrenme*, Bursa: Dora Yayıncılık.
- İlhan, M., ve Çetin, B. (2014). LISREL ve AMOS programları kullanılarak gerçekleştirilen yapısal eşitlik modeli (yem) analizlerine ilişkin sonuçların karşılaştırılması. Eğitimde ve Psikolojide Ölçme ve Değerlendirme Dergisi, 5(2): 26-42.
- Jain, S. (2018). An Overview of Regularization Techniques in Deep Learning (with Python code). <https://www.analyticsvidhya.com/blog/2018/04/fundamentals-deep-learning-regularization-techniques/> (20.04.2019).
- Kakıcı, A. (2008). Biyometrik Tanıma Sistemleri. <http://www.ahmetkakici.com/genel/biyometriktanimasistemleri/> (13.04.2019).
- Karaboğa, D. (2017), *Yapay Zekâ Optimizasyon Algoritmaları*, Ankara: Nobel Yayıncılık.
- Karakuş, B. A. (2018). Derin Sinir Ağları için Aktivasyon Fonksiyonları. <http://buyukveri.firat.edu.tr/2018/04/17/derin-sinir-aglari-icin-aktivasyon-fonksiyonlari/> (18.04.2019).
- Karpathy, A., & Fei-Fei, L. (2015). Deep visual-semantic alignments for generating image descriptions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 3128-3137).

- Kargın, S. (2007). *Yapay Sinir Ağları İle Beyin Tümörü Tomografi Görüntülerinin Sınıflandırılması*, Yüksek Lisans Tezi, İstanbul Üniversitesi, Fen Bilimleri Enstitüsü, İstanbul.
- Kaya, A. (2017). *Derin Öğrenme ile Rakam Öğretme*, Yüksek Lisans Tezi, Beykent Üniversitesi, Fen Bilimleri Enstitüsü, İstanbul.
- Kaya, Y. (2018), *Uygulamalı Örneklerle Python Programlama Dili*, İstanbul: Papatya Yayıncılık.
- Kılınç, D., ve Başegmez, N. (2018), *Uygulamalarla Veri Bilimi*, İstanbul: Abaküs Yayıncılık.
- Kızrak, A. (2018), Adım Adım Google Colab Ücretsiz TPU Kullanımı. <https://medium.com/deep-learning-turkiye/ad%C4%B1m-ad%C4%B1m-google-colab-%C3%BCcretsiz-tpu-kullan%C4%B1m%C4%B1-621dc6e5487d> (18.04.2019).
- Kızrak, A. (2018). Derine Daha Derine: Evrişimli Sinir Ağları. <https://medium.com/deep-learning-turkiye/deri%CC%87ne-daha-deri%CC%87ne-evri%C5%9Fimli-sinir-a%C4%9Flar%C4%B1-2813a2c8b2a9> (18.04.2019).
- Koç, O. (2018), *Daha İyi Bir Dünya İçin Yapay Zekâ*, İstanbul: Doğan Kitap.
- Kolokotsa, D., Saridakis, G., Pouliezios, A., and Stavrakakis, G. S. (2006). Design and installation of an advanced EIB™ fuzzy indoor comfort controller using Matlab™. *Energy and buildings*, 38(9): 1084-1092.
- Köse, İ. (2018), *Veri Madenciliği*, İstanbul: Papatya Yayıncılık.
- Kubat, C. (2017), *Matlab Yapay Zekâ ve Mühendislik Uygulamaları*, İstanbul: Abaküs Yayıncılık.
- Kuyumcu, B. (2016), *OpenCv Görüntü İşleme ve Yapay Öğrenme*, İstanbul: Level Kitap.
- Küçüksille, E. U., ve Tokmak, M. (2011). Yapay arı kolonisi algoritması kullanarak otomatik ders çizelgeleme. *Süleyman Demirel Üniversitesi Fen Bilimleri Enstitüsü Dergisi*, 15(3): 203-210.
- Lecun, Y., Cortes, C., and Burges, C. MNIST Veri Tabanı. <http://yann.lecun.com/exdb/mnist/> (18.04.2019).
- Nabiyev, V.V. (2016), *Yapay Zekâ*, Ankara: Seçkin Yayıncılık.

- Nagi, J., Ahmed, S. K., and Nagi, F. (2008). A MATLAB based face recognition system using image processing and neural networks. In 4th International Colloquium on Signal Processing and its Applications 2: 83-8.
- Nilsson, N.J. (2018), *Yapay Zekâ Geçmişi ve Geleceği*, İstanbul: Boğaziçi Üniversitesi Yayınevi.
- Oliveira, F.P.M., and Branco, M.C. (2015), Computer-aided diagnosis of Parkinson's disease based on [123I]FP-CIT SPECT binding potential images, using the voxels-as-features approach and support vector machines, *Journal of Neural Engineering*, 12(2):026008.
- Ötkün, A. ve Karlık, B. (2013). YSA ve Pencere Ortalamaları Kullanılarak Yüz Tanıma Sistemi. Otomatik Kontrol Ulusal Toplantısı, İnönü Üniversitesi, 26-28 Eylül, Malatya, Türkiye.
- Özel, R. (2018). C# nedir? C# ile Neler Yapılabilir?. <https://medium.com/@resulzel/c-nedir-c-ile-neler-yap%C4%B1labilir-199e8b337615> (15.04.2019).
- Özyiğit, F., Canbaz Kabay, S., ve Arık, Ö. (2016), Akılcı İlaç Kullanımı ve Parkinson Hastalığı, *Journal Of Contemporary Medicine*, 6(2): 104-109.
- Prashanth, R., Roy, S. D., Ghosh, S., and Mandal, P. K. (2013). Shape features as biomarkers in early Parkinson's disease. In 2013 6th International IEEE/EMBS Conference on Neural Engineering (NER) (pp. 517-520).
- Prashanth, R., Roy, S.D., Mandal, P.K., and Ghosh, S. (2016), High-Accuracy Detection of Early Parkinson's Disease through Multimodal Features and Machine Learning, *International Journal of Medical Informatics*, 90:13-21.
- Prashanth, R., Roy, S. D., Mandal, P. K., and Ghosh, S. (2017). High-accuracy classification of parkinson's disease through shape analysis and surface fitting in 123I-Ioflupane SPECT imaging. *IEEE journal of biomedical and health informatics*, 21(3): 794-802.
- Prashanth, R., and Roy, S.D. (2018), Novel and improved stage estimation in Parkinson's disease using clinical scales and machine learning, Elsevier, 305:78-103.
- Rana, B., Juneja, A., Saxena, M., Gudwani, S., Kumaran, S.S., Behari, M., and Agrawal, R.K. (2015), Graph-Theory-Based Spectral Feature Selection for Computer Aided Diagnosis of Parkinson's Disease Using T1-Weighted MRI, *International Journal of Imaging Systems and Technology*, 25:245-255.
- Salah, K. (2017). A novel model order reduction technique based on artificial intelligence. *Microelectronics journal*, 65: 58-71.

Tenekeci, M. E., Gümüşçü, A., ve Aslan, M. (2014). Görüntüden OpenCV ile Duygu Analizi. Akademik Bilişim 14-XVI. Akademik Bilişim Konferansı Bildirileri, Mersin Üniversitesi, 5-7 Şubat, Mersin, Türkiye.

Uğur, A., ve Kınacı, A. C. (2006). Yapay Zekâ Teknikleri Ve Yapay Sinir Ağları Kullanılarak Web Sayfalarının Sınıflandırılması. XI. Türkiye'de İnternet Konferansı Bildirileri, TOBB Ekonomi ve Teknoloji Üniversitesi, 21-23 Aralık, Ankara, Türkiye.

Uke, N., and Thool, R. (2013). Moving Vehicle Detection For Measuring Traffic Count Using Opencv. Journal of Automation and Control Engineering, 1(4): 349-352.

Wang, J., and Perez, L. (2017). The effectiveness of data augmentation in image classification using deep learning. Convolutional Neural Networks Vis. Recognit.

Yao, S., Zhu, Q., and Siclait, P. (2018). Categorizing Listing Photos at Airbnb. <https://medium.com/airbnb-engineering/categorizing-listing-photos-at-airbnb-f9483f3ab7e3> (18.04.2019).

Yılmaz, A. (2017), *Yapay Zekâ*, İstanbul: Kodlab Yayıncılık.

Yüceoğlu, B. (2019). Keras ile Derin Öğrenmeye Giriş. <http://www.veridefteri.com/2019/02/06/keras-ile-derin-ogrenmeye-giris/> (18.04.2019).

Zhang, Y. C., and Kagen, A. C. (2017). Machine learning interface for medical image analysis. Journal of digital imaging, 30(5): 615-621.

<https://www.tensorflow.org/install/gpu> (Erişim Tarihi: 18.04.2019)

<https://www.tensorflow.org/install/pip> (Erişim Tarihi: 18.04.2019)

ÖZGEÇMİŞ

Kişisel Bilgiler	
Adı Soyadı	Okan ALKAN
Doğum Yeri ve Tarihi	YUSUFELİ / 29.01.1991
Eğitim Durumu	
Lisans Öğrenimi	Atatürk Üniversitesi Kazım Karabekir Eğitim Fakültesi Bilgisayar ve Öğretim Teknolojileri Eğitimi
Bildiği Yabancı Diller	İngilizce
İş Deneyimi	
Çalıştığı Kurumlar	Atatürk Üniversitesi Uzaktan Eğitim Uygulama ve Araştırma Merkezi (2013-2016) Erzincan Binali Yıldırım Üniversitesi Meslek Yüksekokulu (2016-Halen)
İletişim	
E-posta Adresi	okanalkan011@gmail.com
Mezuniyet Tarihi	
19.07.2019	