T.C.

YILDIZ TEKNİK ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

DERİN ÖĞRENME İLE GÜNLÜK AKTİVİTE VE SEYAHAT TÜRLERİNİ BİRLİKTE TANIMA

Jemshit ISKANDEROV

YÜKSEK LİSANS TEZİ

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Programı

Danışman

Dr. Öğr. Üyesi M. Amaç GÜVENSAN

Nisan, 2019

T.C.

YILDIZ TEKNİK ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

DERİN ÖĞRENME İLE GÜNLÜK AKTİVİTE VE SEYAHAT TÜRLERİNİ BİRLİKTE TANIMA

Jemshit ISKANDEROV tarafından hazırlanan tez çalışması çalışması 07.05.2019 tarihinde aşağıdaki jüri tarafından Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı, Bilgisayar Mühendisliği Programı **YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Dr. Öğr. Üyesi M. Amaç GÜVENSAN

Yıldız Teknik Üniversitesi

Danışman

Jüri Üyeleri

Dr. Öğr. Üyesi M. Amaç GÜVENSAN, Danışman
Yıldız Teknik Üniversitesi _____

Prof. Dr. Banu DİRİ, Üye
Yıldız Teknik Üniversitesi _____

Dr. Öğr. Üyesi Gökhan İNCE, Üye
İstanbul Teknik Üniversitesi _____

Danışmanım Dr. Öğr. Üyesi M. Amaç GÜVENSAN sorumluluğunda tarafımca hazırlanan Derin Öğrenme ile Günlük Aktivite ve Seyahat Türlerini Birlikte Tanıma başlıklı çalışmada veri toplama ve veri kullanımında gerekli yasal izinleri aldığımı, diğer kaynaklardan aldığım bilgileri ana metin ve referanslarda eksiksiz gösterdiğimi, araştırma verilerine ve sonuçlarına ilişkin çarpıtma ve/veya sahtecilik yapmadığımı, çalışmam süresince bilimsel araştırma ve etik ilkelerine uygun davrandığımı beyan ederim. Beyanımın aksinin ispatı halinde her türlü yasal sonucu kabul ederim

Jemshit ISKANDEROV

İmza

Bu çalışma, Yıldız Teknik Üniversitesi Bilimsel Araştırma Proje Koordinatörlüğü' nün FYL-2018-3171 numaralı projesi ile desteklenmiştir.

Sevgili Eşime

TEŞEKKÜR

Yüksek lisans tez çalışması, benim için yüksek lisans öğrenim sürecimdeki en çok verimli ve öğretici proje olmuştur. Bu proje ile Makine Öğrenmesi alanına sıfırdan giriş yapmış oldum ve bu konuda neler yapılabilir, hangi sorunlar çözülebilir öğrenmiş oldum. Tez süreci uzun ve zor bir süreç oldu, bu süreçte baştan sona bana çok destek olan danışman hocam M.Amaç Güvensana çok teşekkür ediyorum.

Jemshit ISKANDEROV

İÇİNDEKİLER

KISALTMA LİSTESİ IX

ŞEKİL LİSTESİ XI

TABLO LİSTESİ XIV

ÖZET XV

ABSTRACT XVII

1 GİRİŞ 1

1.1 Giriş 1

1.2 Literatür Özeti 2

1.3 Tezin Amacı ve Kapsamı 10

1.4 Hipotez 11

2 AKTİVİTE VE SEYAHAT TÜRÜ TANIMA 13

2.1 Aktivite ve Seyahat Türü Tanıma 13

2.1.1 Sensörler 15

2.1.2 Aktiviteler ve Seyahat Türleri 16

2.1.3 Uygulama Alanları 17

2.2 Aktivite ve Seyahat Türü Tanıma Adımları, Yöntemleri 18

2.2.1 Veri Toplama 19

2.2.2 Veri Düzenleme 20

2.2.3 Veri Süzme 20

2.2.4 Özellik Çıkartma ve Özellik Seçme 21

2.2.5 Eğitim 26

2.2.6 İyileştirme 44

3 KULLANILAN VERİ SETİ, ARAÇLAR VE YÖNTEMLER 45

3.1 Veri Seti ve Ek Veri Toplama 45

3.1.1 Veri Seti, Aktiviteler ve Sensörler 45

3.1.2 Ek Veri Toplama 46
3.2 Veri Hazırlama, Ön İşleme ve Pencerelere Ayırma 49
3.2.1 Veri Hazırlama 49
3.2.2 Veri Sayısı Dengeleme 51
3.2.3 Veriyi Ön İşleme 52
3.2.4 Veriyi Pencerelere Ayırma 53
3.2.5 Eğitim, Doğrulama ve Test Veri Seti Oluşturma 54
3.3 Zaman Alanında Özellik Çıkartma 56
3.4 Kullanılan Yöntemler 59
3.4.1 Oto Kodlayıcı 59
3.4.2 Tekrarlayan Sinir Ağı (RNN) 59
3.4.3 Kayıp Fonksiyonu 59
3.4.4 Aktivasyon Fonksiyonu 59
3.4.5 Eniyileştirici (Optimizer) 60
3.5 Yazılım Geliştirme Ortamı ve Kullanılan Araçlar 60
4 DENEYLER 63
4.1 Sınıflandırma ve Karşılaştırmalı Sonuçlar 63
4.1.2 LSTM ve Zaman Alanı Özellikleri 65
4.1.2 LSTM ve Ham Veri Özellikleri 90
4.1.3 LSTM ve Oto Kodlayıcı Özellikleri 94
4.1.4 Makine Öğrenmesi Yöntemleri 96
4.2 Sınıflandırma Sonucunu Etkileyen Etkenler 96
4.2.1 Veri Ön İşleme 96
4.2.2 Pencere Boyutu 99
4.2.3 Veriyi Eğitim, Doğrulama, Test Setlerine Ayırma Yöntemi 100
4.3 Sınıflandırma Başarısını İyileştirme 102
4.3.1 Bölüm Esaslı İyileştirme 102
4.3.2 Tren ve Hızlı Tren Sınıflarını Birleştirme 105
5 SONUÇ VE ÖNERİLER 106
5.1 Sonuç 106
5.2 Normalleştirme ve Düzleştirme 106

5.3 Zaman Adımı Sayısı 107
5.4 Özellik Çıkartma 108
5.5 Pencere Boyutu 108
5.6 Karıştırılan Aktiviteler 109
5.7 İyileştirme 111
5.8 Model Boyutu ve Sınıflandırma Süresi 113
KAYNAKÇA 116
TEZDEN ÜRETILMIŞ YAYINLAR 125

KISALTMA LİSTESİ

ACO	Ant Colony Optimization
Adadelta	Adaptive Learning Rate Method
Adam	Adaptive Moment Estimation
CNN	Convolutional Neural Network
CNTK	Microsoft Cognitive Toolkit
DHMM	Discrete Hidden Markov Model
DNN	Deep Neural Network
DT	Decision Tree
FFT	Fast Fourier Transform
GIS	Geographic Information System
GPS	Global Positioning System
GRU	Gated Recurrent Unit
GSM	Global System for Mobile
HSR	High Speed Rail
IDE	Integrated Development Environment
k-NN	K-Nearest Neighbors
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
MCU	Micro Computer Unit
MLP	Multi-Layer Perceptron
mRMR	Minimum-Redundancy Maximum-Relevancy
MSE	Mean Squared Error

OHE	One-Hot Encoding
QDA	Quadratic Discriminant Analysis
RBM	Restricted Boltzmann Machine
ReLU	Rectified Linear Units
RNN	Recurrent Neural Network
RF	Random Forest
SGD	Stochastic Gradient Descent
SVM	Support Vector Machine
VGP	Vanishing Gradient Problem
YSA	Yapay Sinir Ağları

ŞEKİL LİSTESİ

Şekil 2.1 Aktivite tanıma yaklaşımları 14
Şekil 2.2 Seyahat türü tanıma yaklaşımları 14
Şekil 2.3 Aktivite ve seyahat türü tanıma adımları 19
Şekil 2.4 Sade Oto Kodlayıcı 24
Şekil 2.5 Gürültü azaltan Oto Kodlayıcı 24
Şekil 2.6 Seyrek Oto Kodlayıcı 25
Şekil 2.7 Derin Oto Kodlayıcı 26
Şekil 2.8 k-NN algoritması 28
Şekil 2.9 Karar Ağacı algoritması 30
Şekil 2.10 Rasgele Orman algoritması 31
Şekil 2.11 Solda 2 boyutlu alanda 1 boyutlu çizgi olan hiper düzlem. Sağda 3 boyutlu alanda 2 boyutlu hiper düzlem [86] 32
Şekil 2.12 Basit Tekrarlayan Sinir Ağı [89] 33
Şekil 2.13 LSTM ağı kapıları [89] 34
Şekil 2.14 GRU ağı kapıları [89] 35
Şekil 2.15 Konvolüsyonel Sinir Ağı katmanları [92] 35
Şekil 2.16 Sigmoid aktivasyon fonksiyonu [93] 37
Şekil 2.17 Tanh aktivasyon fonksiyonu [93] 37
Şekil 2.18 ReLU aktivasyon fonksiyonu [93] 38
Şekil 2.19 WEKA aracı 42
Şekil 2.20 Scikit-learn aracı 43
Şekil 2.21 Sınıflandırma sonucu üzerinde iyileştirme yapmak 44
Şekil 3.1 Veri dosyasının içeriği 46
Şekil 3.2 Android Studio projesi 47
Şekil 3.3 Android veri toplama uygulaması, sensör listesi ve detayı 48
Şekil 3.4 Android veri toplama uygulaması, yapılan aktivitenin ve toplayan kişinin bilgilerini girme ekranı. 48
Şekil 3.5 Android veri toplama uygulamasının çalışma akışı 49
Şekil 3.6 Veri seti klasör yapısı 50
Şekil 3.7 Veri dosya düzeni 50
Şekil 3.8 Veri hazırlama işleminin çalışma akışı 51

Şekil 3.9 Veriye eksen bazında filtre uygulanması 53
Şekil 3.10 Pencere oluşturma (16 saniyelik) 54
Şekil 3.11 Pencereleri 3-1-1 yoluyla setlere ayırmak 55
Şekil 3.12 Pencereleri %60-%20-%20 yoluyla setlere ayırmak 55
Şekil 3.13 Zaman alanında özellik çıkartma 56
Şekil 4.1 LSTM sınıflandırma ağ mimarisi 65
Şekil 4.2 Eğitim hızının doğruluğa etkisi 67
Şekil 4.3 Eğitim hızının kayıp değerine etkisi 68
Şekil 4.4 Eğitim hızının test seti üzerinde sonuçları 69
Şekil 4.5 Dropout eklenmesinin doğruluğa etkisi 70
Şekil 4.6 Dropout eklenmesinin kayıp değerine etkisi 71
Şekil 4.7 Dropout değerlerinin test seti üzerinde sonuçları 72
Şekil 4.8 Ek katmanın doğrulama setinde sonuca etkileri 74
Şekil 4.9 Ek katmanın test seti üzerinde sonuçları 75
Şekil 4.10 LSTM birim sayısının doğruluğa etkisi 76
Şekil 4.11 LSTM birim sayısının kayıp değerine etkisi 77
Şekil 4.12 LSTM birim sayısının test seti üzerinde sonuçları 78
Şekil 4.13 Batch size parametresinin doğrulama setinde sonuca etkileri 79
Şekil 4.14 LSTM batch size test seti üzerinde sonuçları 80
Şekil 4.15 (a) LSTM hücre sayısının doğruluğa etkisi. (b) LSTM hücre sayısının test seti üzerinde sonuçları 81
Şekil 4.16 (a) Eniyileştirici algoritmasının doğruluğa etkisi. (b) Eniyileştirici algoritmasının test seti üzerinde sonuçları 82
Şekil 4.17 (a) Eğitim ve doğrulama setinde doğruluk (b) Eğitim ve doğrulama setinde kayıp 84
Şekil 4.18 (a) L1, L2 düzenleme tekniğinin doğruluğa etkisi. (b) L1, L2 düzenleme tekniğinin test seti üzerinde sonucu 85
Şekil 4.19 (a) Normalleştirme yöntemlerinin doğruluğa etkisi. (b) Normalleştirme yöntemlerinin test seti üzerinde sonuçları 86
Şekil 4.20 (a) Tek ve çift yönlü LSTM'in doğruluğa etkisi. (b) Tek ve çift yönlü LSTM'in test seti üzerinde sonuçları 87
Şekil 4.21 Girdinin farklı boyutlarda bölümlere ayrılmasının doğruluğa etkileri 88

Şekil 4.22 Girdinin farklı boyutlarda bölümlere ayrılmasının test seti üzerinde sonuca etkileri..... 89

Şekil 4.23 LSTM sınıflandırma ağı mimarisi 90

Şekil 4.24 Ham verinin farklı boyutlarda bölümlere ayrılarak LSTM ağında kullanılmasının doğruluğa (a) ve kayıp değerine (b) etkileri..... 91

Şekil 4.25 Ham verinin farklı boyutlarda bölümlere ayrılarak LSTM ağında kullanılmasının test setinde sonuca etkileri 92

Şekil 4.26 (a) Normalleştirme yöntemlerinin ham veride doğruluğa etkisi. (b) Normalleştirme yöntemlerinin ham veride test seti üzerinde sonuçları 93

Şekil 4.27 Derin Oto Kodayıcı mimarisi 94

Şekil 4.28 Derin Oto Kodayıcıda dropout parametresinin LSTM sınıflandırma sonucuna etkileri 95

Şekil 4.29 Eğitim setinde ivme ölçer x ekseni verileri çubuk grafiği (Solda) Ön işleme yapılmadan önce (Sağda) Aritmetik ortalama Düzleştirme yapıldıktan sonra 97

Şekil 4.30 (a) Doğrulama setinde Düzleştirmenin doğruluğa etkisi (b) Test setinde Düzleştirmenin sonuca etkileri 98

Şekil 4.31 Pencere boyutlarına göre farklı yöntemler ile elde edilen doğruluk sonuçları 99

Şekil 4.32 Veri setini sırayla 3-1-1 şeklinde setlere ayırmak 100

Şekil 4.33 Veri setini aktiviteye göre %60-%20-%20 oranında setlere ayırmak 101

Şekil 4.34 Veri setini aktiviteye ve cihaz konumuna göre %60-%20-%20 oranında setlere ayırmak 102

Şekil 4.35 Bölüm esaslı iyileştirme..... 103

Şekil 5.1 LSTM ve zaman alanı özellikleriyle elde edilen sınıflandırma sonucunun karışıklık matrisi 111

Şekil 5.2 İyileştirme adımından sonra sınıflandırma sonucunun karışıklık matrisi..... 112

Şekil 5.3 Sınıflandırma Model Boyutu 113

Şekil 5.4 Modelin Tek Örneği Sınıflandırma Süresi 114

TABLO LİSTESİ

Tablo 3.1 Aktiviteler ve seyahat türleri, cihaz konumları ve örnek sayısı..... 52
Tablo 3.2 Zaman alanında çıkartılan özellikler..... 57
Tablo 3.3 Projenin yazılım gereksinimleri..... 61
Tablo 3.4 Projenin donanım gereksinimleri..... 62
Tablo 4.1 Karışıklık matrisi..... 64
Tablo 4.2 Araçla seyahat sınıflarında iyileştirme yapmanın metriklere etkileri 103
Tablo 4.3 Aktivite ve seyahat türlerine göre elde edilen sonuç metrikleri (iyileştirmeden önce/iyileştirmeden sonra) 104

ÖZET

Derin Öğrenme ile Günlük Aktivite ve Seyahat Türlerini Birlikte Tanıma

Jemshit ISKANDEROV

Bilgisayar Mühendisliği Anabilim Dalı

Yüksek Lisans Tezi

Danışman: Dr. Öğr. Üyesi M. Amaç GÜVENSAN

Aktivite tanıma ve seyahat türü tanıma sağlık, akıllı ortam, ulaşım planlama, trafik yönetimi gibi geniş uygulama alanına sahip önem arz eden bir konudur. Kişilerin aktivitelerinin takip edilmesi akıllı telefonların ve akıllı saatlerin yaygınlaşması ile daha kolay hale gelmiştir. Aktivite ve seyahat türü tanıma literatürde ayrı ayrı ele alınmış problemler olsa da doğası gereği bu problemleri birlikte değerlendirmek daha fazla uygulama alanına çıktı üretmeyi sağlamaktadır. Bu çalışmada bu iki konu tek bir problem olarak ele alınmıştır. Tez kapsamında akıllı telefonlarda bulunan ivme ölçer, jiroskop ve mıknatıs ölçer sensörleri kullanılarak 13 adet aktivite ve seyahat türü cihazın kişi üzerindeki konumundan bağımsız sınıflandırılmıştır. Bu aktivite ve seyahat türleri şunlardır: *sabit durma, yürüme, koşma, merdiven çıkma, merdiven inme, bisiklet, motosiklet, araba, metro, tren, hızlı tren, tramvay, metrobüs*. Bu tezde ham veri, istatistiksel özellikler ve Oto Kodlayıcı ile çıkartılan özellikler kullanılarak LSTM derin öğrenme yöntemi ile sınıflandırma yapılmıştır. LSTM (Long Short-Term Memory) ağıyla 16 saniyelik pencerede istatistiksel özelliklerle %86.4, ham verilerle %83.3, Oto Kodlayıcı özellikleriyle %79.3 doğruluk edilmiştir. Bu sonuçlar makine öğrenmesi yöntemleri olan Naive

Bayes, Karar Ağacı, Destek Vektör Makineleri ve Rasgele Orman ile elde edilen sonuçlar ile karşılaştırılmıştır. Makine öğrenmesi yöntemleriyle sırasıyla %56.9, %71.2, %81.6 ve %83.8 doğruluk sonuçları elde edilmiştir. LSTM ile en iyi elde edilen sonuç üzerinde iyileştirme yapılarak %93.1 doğruluk elde edilmiştir.

Anahtar Kelimeler: Aktivite Tanıma, Seyahat Türü Tanıma, Derin Öğrenme, İstatistiksel Özellikler

ABSTRACT

Activity Recognition and Transport Mode Detection Using Deep Learning

Jemshit ISKANDEROV

Department of Computer Engineering

Master of Science Thesis

Advisor: Assist. Prof. Dr. M. Amaç GÜVENSAN

Activity recognition and travel mode detection is important topic with application areas in health, smart environment, transportation planning and traffic management domains. The recognition of human activities has become easier due to the widespread use of smartphones and smart watches. While activity recognition and travel mode detection are treated as separate problems in the literature, evaluating them together will provide solutions to wider application domains. In this paper activity recognition and travel mode detection are treated as single problem. In the scope of thesis 13 activities and travel modes are classified using accelerometer, gyroscope and magnetometer sensors that are built in smartphones. Classified activities and travel modes are *still, walk, run, upstairs, downstairs, bicycle, motor, car, metro, train, HSR, tram and metrobus*. In this thesis, LSTM (Long Short-Term Memory) deep learning method is trained with raw data, with 27 statistical time domain features and with features extracted by Auto Encoders. Using 16 seconds windows and LSTM, we have obtained 86.4%

accuracy with time domain features, 83.3% accuracy with raw data, 79.3% accuracy with features obtained from Auto Encoder. Those results are compared with results obtained from Naïve Bayes, Decision Tree, Support Vector Machine and Random Forest machine learning methods. Accuracy results obtained from machine learning methods are %56.9, %71.2, %81.6 and %83.8 in order. 93.1% accuracy is obtained after healing of best result obtained using LSTM.

Keywords: Activity Recognition, Transport Mode Detection, Deep Learning, Statistical Features

1
GİRİŞ

1.1 Giriş

Aktivite tanıma uzun zamandır üzerinde çalışmalar yapılan bir konudur. Aktivite tanıma çalışmaları kişinin yürüme, koşma, oturma gibi basit aktivitelerini ve/veya yemek yapma, ders anlatma, bitki sulama gibi karışık aktivitelerini tanıyabilmeyi (sınıflandırabilmeyi) amaçlamaktadır. Kişisel form gelişimi (kalori ölçme, aktivite takip), hastaların hareket takibi [1] [2], akıllı ortamlar [3], kişiye özel önerilerde bulunmak için mobil uygulamalar gibi geniş uygulama alanına sahip olan aktivite tanımayı günümüzde hayata geçirmek, daha önceki telefon veya giyilebilir cihaz kullanmayan aktivite tanıma yaklaşımlarına kıyasla, çok daha elverişli ve uygun hale gelmiştir. Aktivite tanımanın kolay uygulanabilir hale gelmesinin en büyük sebeplerinden birisi akıllı telefonlar ve giyilebilir cihazlardır. 2018 yılında 3.3 milyar akıllı telefon kullanıcısı vardı [4] ve günlük akıllı telefon kullanımı 2017'de 3.3 saat etmekteydi [5]. Akıllı saat ve akıllı bileklik gibi giyilebilir cihazların 2018 senesinde kullanıcı sayısı 51.9 milyona ulaşmıştır [6]. Herkes tarafından her gün kullanılan bu cihazlarda bulunan ivme ölçer, jiroskop, mıknatıs ölçer, basınç ölçer vb. sensörler aktivite tanıma yapmak için uygun olmaktadır. Birçok çalışma basit aktivite [7] [8] ve karışık aktivite tanıma [9] konusunu ayrı ele almışlardır. Bazı çalışmalar bu iki tür aktiviteleri beraber değerlendirmiştir ve ikisini birlikte tanımaya çalışmıştır [10]. Çalışmaların çoğu ivme ölçer ve jiroskop sensörleri yardımıyla basit aktivite tanıma konusunda iyi sonuçlar elde etmişlerdir. Karışık aktivite tanıma için ise bu sensörlerin yeterli olmadığı ve GPS vb. ortam hakkında bilgi içeren verilerin olması gerektiğini vurgulamışlardır [11].

Seyahat türü tanıma hava kirliliği ölçümü/takibi [12] [13], ulaşım planlama ve trafik yönetimi [14] gibi uygulama alanlarına sahip önemli bir konudur. Kişinin araba, metro, tramvay, otobüs ile seyahat etme aktivitelerini birbirinden ayırt

etmek seyahat türü tanımaya örnektir. Seyahat türü tanımanın uygulama alanlarından olan ulaşım planlama ve trafik yönetimi için anket doldurarak veya telefon yoluyla bilgiler toplayarak çözümler uygulanmıştır. Daha sonra çalışmalar akıllı telefonlarda bulunan GPS, ivme ölçer, jiroskop vb. sensörleri kullanarak seyahat türü tanıma işlemini daha uygun şekilde yapmışlardır. Birçok çalışma GPS ve ulaşım ağı bilgilerini kullanarak seyahat türü tanıması yapmıştır [15] [16] ve iyi sonuçlar elde etmişlerdir. Diğer çalışmalar GPS kullanmadan ivme ölçer, jiroskop ve/veya mıknatıs ölçer ile bu probleme çözüm önermişlerdir [17] [18].

1.2 Literatür Özeti

Literatürde aktivite tanıma ve seyahat türü tanıma ayrı ayrı ele alınmış konulardır ve üzerinde çok çalışmalar yapılmıştır. Aktivite tanıma üzerinde çalışma yapan makaleler basit aktivite ve karışık aktivite tanıma olmak üzere konuyu ikiye ayırarak probleme yaklaşmışlardır. Üzerinde en çok çalışma yapılmış olan basit aktiviteler, belirli aksiyonlardan oluşan ve çok kısa sürede yapılabilen aktivitelerdir. Yürüme, koşma, oturma, zıplama gibi aktiviteler basit aktivitelere örneklerdir. Karışık aktiviteler ise daha uzun sürede ve birkaç tane basit aktivite ve aksiyonların birleşiminden oluşan aktivitelerdir. Yemek pişirme, ev temizliği yapma, alışveriş yapma, ders anlatma karışık aktivitelere örnektir.

Aktivite tanıma alanında birçok makale basit aktiviteler üzerinde çalışmışlardır. Bu çalışmaların önemli bir kısmı sadece ivme ölçer sensörlerinden toplanan verileri kullanmışlardır. Basit aktiviteleri tanımak için küçük boyutlu pencereler yeterli görülmüştür ve 16 saniyenin altında pencere boyutları tercih edilmiştir [19] [20] [21]. Makale [22] aktivite tanımada enerji tüketimi üzerinde çalışmıştır ve özellik sayısını artırmanın, verinin örnekleme frekansını artırmanın enerji tüketimini olumsuz etkilediğini yazmıştır.

Makale [23] MLP (Multi-Layer Perceptron) ağı kullanarak 6 tane aktiviteyi (sabit durma, yürüme, tempolu koşma, merdiven çıkma, merdiven inme, oturma) sınıflandırmıştır. %91 doğruluk elde eden bu çalışma merdiven inme ve merdiven çıkma aktivitelerinin birbiriyle karıştırıldığını yazmıştır. Benzer aktiviteler üzerinde ivme ölçer sensörüyle aktivite tanıma yapan makale [24] Karar Ağacı ile

%85 doğruluk elde etmiştir. Makale [24] eğitim verisinin çok olmasının doğruluğu olumlu etkilediğini yazmıştır. İvme ölçer kullanan bir diğer çalışma [19] sabit durma, yürüme, koşma, yatma, zıplama ve oturma aktivitelerini sınıflandırmıştır. Makale [19] hem cihaz oryantasyonundan hem de cihaz konumundan bağımsız basit aktivite tanıması yapmıştır. iPhone cihazlarıyla toplanan veriler üzerinde %86 doğruluk sonucunu elde etmişlerdir.

Basit aktivite tanımada sadece ivme ölçer kullanarak en iyi sonuçlardan birini elde eden makale [7] 5 tane aktivite üzerinde çalışmıştır. İvme ölçer eksenlerinin büyüklüğünü (karelerinin toplamının karekökü) kullanarak cihaz oryantasyonundan bağımsız aktivite tanıması yapmıştır. Sabit durma, yürüme, koşma, bisiklet sürme, araba sürme aktivitelerine ait veriler QDA (Quadratic Discriminant Analysis) algoritması ile eğitilmiş ve %95 doğruluk sonucunu elde edilmiştir. k-NN (K-Nearest Neighbors) algoritması kullanıldığında ise %93 doğruluk sonucunu almışlardır. [25] makalesi ivme ölçer ile yürüme, tempolu koşma, koşma, merdiven çıkma, merdiven inme ve bisiklet sürme aktiviteleri üzerinde çalışmıştır. Bu çalışmada SVM (Support Vector Machines) makine öğrenmesi yöntemiyle %80 civarı doğruluk elde etmişlerdir. İvmeölçer sensörü kullanarak en iyi sonuçlardan birini elde eden makale [2] 8 basit aktivite üzerinde çalışmıştır. %95 doğrulukla sınıflandırılan bu aktiviteler şunlardır: *sabit durma, yürüme, koşma, düşme, araba sürme, merdiven çıkma, merdiven inme, bisiklet sürme*. Makale [2] sınıflandırma modelini çevrimdışı eğittikten sonra çevrimiçi olarak bu modeli güncellemiştir ve kişiye özel sınıflandırma modeli oluşturmuştur.

Makale [26] ivme ölçer ve jiroskop sensörleri kullanarak 5 basit aktiviteyi (sabit durma, yürüme, yatma, merdiven çıkma, merdiven inme) %89 kesinlikle sınıflandırmıştır. Benzer şekilde ivme ölçer ve jiroskop sensörlerini kullanan makale [27] 6 basit aktiviteyi %94 doğrulukla sınıflandırmıştır. Makale [8] ivme ölçer, jiroskop ve mıknatıs ölçer kullanarak basit aktivite tanıma yapmıştır. Sınıflandırılan aktiviteler şunlardır: *sabit durma, yürüme, koşma, oturma, bisiklet*. Bu 5 aktiviteyi %97 doğrulukla sınıflandırabilen bu makale en iyi sonuç elde eden çalışmalardan biri olsa da, kısıtlı aktiviteler üzerinde çalışmıştır. Bir diğer birden

fazla sensör kullanan çalışma [28] ivme ölçer, G sensör, ışık sensörü, mıknatıs ölçer, jiroskop ve yakınlık ölçer sensörleri kullanmıştır. Bu sensörlerden toplanan verilerden hem zaman alanında hem de frekans alanında özellikler çıkartılmıştır. Makale [28] frekans alanında özellik çıkartmanın daha karışık olduğunu yazmıştır. Bu makale cihaz oryantasyon ve konumundan bağımsız aktivite tanıma yapan çalışmalardan biridir. Çıkartılan özelliklerle Karar Ağacı eğitilmiştir ve 5 basit aktivite %94 doğrulukla sınıflandırılmıştır.

Aktivite tanımada derin öğrenme yöntemleri kullanan ve iyi başarılar elde eden çalışmalar mevcuttur. Makale [29] ivme ölçer ve jiroskop sensör verilerinden sinyal resmi oluşturmuştur ve daha sonra bu resimlerden Konvolüsyonel Sinir Ağı (CNN) kullanarak özellik çıkartmıştır. UCI veri setinde[1] (6 basit aktivite) %97 doğruluk elde etmiştir. Benzer bir yaklaşımı makale [30] kullanmıştır ve WISDM[2] (6 basit aktivite) veri setinde %98 doğruluk elde etmiştir. Çalışma [31] Kısıtlı Boltzmann Makinesi (RBM) derin öğrenme yöntemi kullanarak özellikler çıkartmıştır. UCI veri seti üzerinde Rasgele Orman (RF) makine öğrenmesi yöntemiyle %75 doğruluk elde ederken önerdikleri MM-RBM yöntemiyle %81 doğruluk elde etmişlerdir. Makale [32] Tekrarlayan Sinir Ağı (RNN) çeşidi olan LSTM derin öğrenme yöntemi kullanarak WISDM veri seti üzerinde %92 doğruluk elde etmiştir.

Aktivite tanımanın yanında bir diğer konu olan seyahat türü tanıma, genel olarak nokta esaslı veya bölüm esaslı seyahat türü tanıma olarak ikiye ayrılabilir. Bu iki yaklaşım sınıflandırma hızı (gerçek zamanlı veya değil), kullanılan sensörler (GPS kullanmak veya kullanmamak) ve çözülen problemin amacı açılarından birbirinden farklıdırlar. Bölüm esaslı sınıflandırmanın nokta esaslı sınıflandırmadan en önemli farkı tahmin edilen sınıf değişmediği sürece tahmin tek bir sonuç olarak değerlendirilir. Özellikle kişinin yolculuklarını seyahat türü bazlı bölümlere ayırmak için kullanılmaktadır. Bu yüzden bölüm esaslı seyahat türü tanıma yapan çalışmaların bir kısmı sadece GPS sensörü kullanmıştır [16]

[1] https://archive.ics.uci.edu/ml/datasets/Heterogeneity+Activity+Recognition
[2] http://www.cis.fordham.edu/wisdm/dataset.php#activityprediction

[33] [34]. Araç veya yolculuk değişikliğini anlamak için GPS sensöründeki ani yön değişimi vb. farklı yaklaşımlar kullanılmıştır. Nokta esaslı seyahat türü tanımadaki amaç ise seyahati yolculuklara ayırmak değildir. Bu yaklaşımda kişinin seyahat türünü gerçek zaman yakın sürede tahmin etmektir. Eğer GPS sensörü kullanılmışsa, sensörden her veri toplandığında gerçek zamanlı sınıflandırma tahmini yapılmaya çalışılır. Eğer ivme ölçer gibi hızlı veri toplanabilen sensörler kullanılmışsa, belirli boyuttaki pencere aralıklarıyla sınıflandırma yapılır ve gerçek zamana yakın bir sürede sonuç üretilmeye çalışılır. Seyahat türü tanıma yapan çalışmaların yarısından fazlası 20 saniye ve altında pencere boyutları kullanmışlardır. Pencere boyutunu büyütmenin doğruluğu çok olumlu etkilemediği [18] ve [35] makaleleri tarafından belirtilmiştir.

Makale [18], bu projede bir bölümü kullanılan veri setinin tamamını kullanarak seyahat türü tanıma çalışması yapmıştır. Sınıflandırılan 5 seyahat türleri şunlardır: *yürüme, bisiklet, sabit durma, koşma, araçta (hızlı tren, tren, metro, otobüs, araba, motosiklet)*. 30Hz frekansta toplanan verilerden zaman ve frekans alanında özellikler çıkartılmıştır. 17 saniyelik pencereler kullanan bu çalışma SVM makine öğrenmesi yöntemiyle %92.5 doğruluk elde etmiştir. Çalışmanın asıl odak noktası enerji tüketimini azaltmaktır ve jiroskop sensör verisini yazılımsal olarak elde etmişlerdir. Bu yüzden test aşamasında sadece ivme ölçer ve mıknatıs ölçer sensörlerini kullanmışlardır. Aynı zamanda sınıflandırma aşamasında özel donanım kullanmışlardır (CPU işlemci yerine MCU). Enerji tüketimini daha azaltmak için önce ivme ölçer ile sabit durma aktivitesi tespitini, sonra diğer bütün hareketli türleri ayırt etmek için iki aşamalı sınıflandırma yapmışlardır. Bu çalışmada araçla seyahat türlerinin hepsi tek sınıf altında sınıflandırılmış olduğu için çok kapsamlı çalışma değildir. Makale [35] bu 5 sınıfı ivme ölçer, jiroskop, mıknatıs ölçer sensörleri kullanarak ve Derin Sinir Ağı (DNN) ile %95 doğrulukla sınıflandırabilmiştir. Makale [36] bu projede kullanılan veri setini kullanarak seyahat türü tanıma çalışması yapmıştır. İvme ölçer, jiroskop ve mıknatıs ölçer sensörlerinden toplanmış verilerden zaman ve frekans alanında özellikler çıkartılarak 17 saniyelik pencereler ile sınıflandırma yapılmıştır. Bu çalışma seyahat türü ve araç türü olmak üzere iki farklı ve bağımsız sınıflandırma deneyi

yapmıştır. Sınıflandırılan gurubun birincisi sabit durma, yürüme, koşma, bisiklet ve araç sınıflarından oluşmuştur. SVM makine öğrenme yöntemi ile bu 5 seyahat türlerini %86 doğruluk ile sınıflandırmışlardır. Çalışmanın diğer deneyi olan ve metro, tren, hızlı tren, araba, motosiklet, otobüs araç türlerinden oluşan deneyde k-NN yöntemiyle %83 doğruluk elde etmişlerdir. Bu deneyler birbirinden bağımsız ele alınmıştır ve iki aşamalı sınıflandırma değildir. Makale [13] benzer bir yaklaşımla seyahat türlerini birleştirerek sınıflandırma yapmıştır. Bu makale 7 tane etiket arasında sınıflandırma yapmıştır: *sabit durma, yürüme, koşma, bisiklet, yol, ray, uçak*. Bu çalışma araba, otobüs, motosiklet ve elektrikli araçları yol etiketi altında tek sınıf olarak değerlendirmiştir. Tramvay, tren ve metro seyahat türlerini ray etiketi altında tek sınıf olarak kullanmışlardır. Veriler 100Hz frekansta toplanmıştır ve ivme ölçer, jiroskop, GPS sensörleri kullanılarak %94-%96 doğruluk sonucunu elde etmişlerdir.

Yürüme, tempolu koşma, bisiklet, metro, araba ve otobüs olmak üzere 6 tür arasında sınıflandırma yapan çalışma [37], SVM ile çevrimiçi öğrenme yapmıştır. Çevrimiçi öğrenme yaparak kişiye özel sınıflandırma modeli oluşturulabilir. Bu çalışma diğer çalışmalardan farklı olarak ivme ölçer, jiroskop, mıknatıs ölçer sensörlerinin yanında yer çekimi ve basınç ölçer sensörlerini de kullanmıştır. Bu 5 sensörden toplanan veriler üzerinde Normalleştirme (Normalization), Düzleştirme (Smoothing) yapılmıştır ve 6 sınıf arasında %97 doğruluk ile sınıflandırma yapılmıştır. Çalışma [38] ivme ölçer, jiroskop ve mıknatıs ölçer sensörleri kullanarak 8 tane seyahat türünü sınıflandırmıştır. Bu seyahat türleri sabit durma, yürüme, metro, vapur, tren, tramvay, araba ve otobüstür. Rasgele Orman makine öğrenmesi yöntemi 1 dakikalık pencereler ile eğitilmiştir ve sınıflandırma aşamasından sonra iyileştirme algoritması uygulanarak %91 anımsama elde edilmiştir.

GPS sensörü kullanarak seyahat türü tanıma yapmak doğruluğu artırmaktadır [39]. Ama fazla enerji tüketimi [39] ve sinyal olmayan mekanlar [16] gibi problemleri birlikte getirmektedir. Makale [40] ivme ölçer ve GPS sensör verilerinden çıkartılan özellikleri kullanarak sabit durma, yürüme, koşma, bisiklet

ve araç olmak üzere 5 ayrı etiketi sınıflandırmıştır. Araç etiketi araba ve otobüs verilerinden oluşmaktadır. 20 saniyelik pencereler kullanarak Rasgele Orman makine öğrenmesi yöntemiyle eğitim yapılmıştır. Bu çalışma sınıflandırma aşamasından sonra bölüm esaslı iyileştirme algoritması uygulamıştır ve sonucu %4 iyileştirerek %93.8 doğruluk elde etmiştir. Bir diğer GPS sensörü kullanan makale [39] sabit durma, yürüme, bisiklet, araba, otobüs, tren ve tramvay olmak üzere 7 seyahat türünü sınıflandırmıştır. Bu çalışma cihaz oryantasyonundan bağımsız seyahat türü tanıma yapmak için ivme ölçer eksenlerinin büyüklüğünü (karelerinin toplamının karekökü) kullanmıştır. Rasgele Orman makine öğrenmesi yöntemi kullanarak ivme ölçer, jiroskop sensörleri ile %86 civarı doğruluk elde ederken ek olarak GPS sensörü kullanınca %95 doğruluk oranına ulaşabilmişlerdir. Makale [39] daha fazla sensör kullanmanın doğruluğu olumlu etkilediğini yazmıştır. GPS sensörü ile seyahat türü tanımaya çalışan makale [41] yürüme, bisiklet, motosiklet, araba ve otobüs olmak üzere 7 sınıf üzerinde çalışmıştır. Karar Ağacı makine öğrenmesi yöntemi 90 saniyelik pencereler ile eğitilmiştir ve iki aşamalı sınıflandırmadan sonra %79 kesinlik sonucuna ulaşmıştır. Makale [41] araba, otobüs ve bisiklet seyahat türlerinin benzer desenler içerdiği için birbiriyle karıştırıldığını belirtmiştir. Makale [33] yürüme, bisiklet, otobüs, araba sınıflarını GPS sensör verilerinden çıkartılmış özellikleri kullanarak sınıflandırmıştır. Bölüm esaslı seyahat türü tanıma yapan bu çalışma yolculukları GPS nokta kümeleri ve GPS yönündeki ani değişimlere bakarak ayırmaya çalışmıştır. Makale [33] diğer çalışmalarda sık kullanılmayan Sinir Ağlarını kullanmayı tercih etmiştir ve %94 doğruluk sonucunu elde etmiştir. Çalışma [34] GPS sensör verileri kullanarak yürüme, bisiklet, e-bisiklet, araba ve otobüs seyahat türleri üzerinde çalışma yapmıştır. Makale [33] gibi bölüm esaslı seyahat türü tanıma yapmıştır ve yolculukları benzer yöntemlerle birbirinden ayırt etmiştir. Eğitim yöntemi olarak Bayesian Ağları kullanılmış ve %92 doğruluk elde edilmiştir.

Makale [42] ivme ölçer ve GPS sensör verileri üzerinde Kalman filtre uyguladıktan sonra özellik çıkartmıştır. Cihaz oryantasyonundan bağımsız seyahat türü tanıma için ivme ölçer eksenlerinin büyüklüğünü kullanmıştır. Bu çalışma DHMM

(Discrete Hidden Markov Model) yöntemiyle 9 seyahat türünü sınıflandırmıştır: *sabit durma, yürüme, bisiklet, motosiklet, araba, otobüs, tren, tramvay, metro*. Bu seyahat türlerini sınıflandırmada %75 F1 sonucunu elde etmişlerdir. GPS ve ivme ölçer sensörleriyle birlikte GSM kullanan makale [43] 4 tane seyahat türünü sınıflandırmıştır. Sınıflandırma iki aşamalı olup önce sabit durma aktivitesi tespit edilmiştir. Kişinin harekette olup olmadığına karar vermek için 10 saniyelik pencere süresi içerisindeki kayıt olduğu GSM kulelerinin sayısına bakılmıştır. Sabit durma, yürüme, araba ve tren sınıflarını %85.2 doğrulukla sınıflandırmışlar ve sınıflandırma yöntemi olarak Karar Ağacını seçmişlerdir.

GIS (Geographic Information System) gerçek zamanlı ulaşım türleri rotası, sefer bilgileri gibi verileri bulunduran sistemdir. GPS sensör ve GIS verileriyle kişinin bulunduğu rotanın hangi ulaşım türüne ait olduğu tespit edilebilmektedir. Seyahat türü tanımada GIS verilerini kullanmak doğruluğu olumlu etkilemektedir [43]. Ama GIS verilerine erişmek sürekli internet bağlantısı gerektirmektedir. Ayrıca GIS verileri her şehir için mevcut değildir. GPS ve GIS verilerini kullanan çalışmalardan biri olan makale [15] sabit durma, yürüme, bisiklet, araba, otobüs ve tren seyahat türlerini sınıflandırmıştır. Çalışma sadece GPS verisi ile %75 doğruluk elde ederken GPS ve GIS verilerini birlikte kullanarak doğruluğu %93'e yükseltmiştir. Makale [16] GPS ve GIS verileri ile kapsamlı çalışma yapmıştır. 10 tane seyahat türünü sınıflandıran bu çalışma makine öğrenmesi yöntemi kullanmamıştır. Bulanık mantık sistemleri (fuzzy logic systems) kullanan bu çalışma bölüm esaslı seyahat türü tanıması yapmıştır. Yolculukları bölümlere ayırmak için ise durma veya sinyal eksikliği anlarını kullanmıştır. Sınıflandırılan seyahat türleri şunlardır: *yürüme, bisiklet, araba, otobüs, tren, metro, tramvay, vapur, yelkenli tekne ve uçak*. Sınıflandırma 3 katman ile yapılmıştır. Birinci katmanda aktivitenin kara, su veya havada yapılan seyahat türü olduğu tespit edilmiştir. Diğer katmanlarda ise detaylı seyahat türü tespiti yapılmıştır. Diğer çalışmalardan daha fazla seyahat türünü %91 başarıyla sınıflandırsa da hem GPS sensörü hem de GIS verileri kullanmak uygulanabilirlik, pil tüketimi açısından problem oluşturabilmektedir.

Sadece ivme ölçer kullanarak seyahat türü tanıma yapan çalışmalar mevcuttur. Sadece ivme ölçer kullanmak daha az enerji tüketmekte, internet bağlantısına ihtiyaç duymamakta ve kapalı mekanlarda da çalışmaktadır. İvme ölçer sensörü kullanan makale [44] yürüme, bisiklet, araba, tren sınıflarını tanımaya çalışmıştır. Bu 4 seyahat türünü Rasgele Orman yöntemi kullanarak %99 doğrulukla sınıflandırmıştır. Benzer bir şekilde yürüme, araba, otobüs, tramvay/tren türlerinden oluşan 4 sınıfı sadece ivme ölçer kullanarak ayrıt etmeye çalışan makale [45] %82 doğruluk elde etmiştir. Makale [45] bölüm esaslı sınıflandırma yapmıştır ve yolculukları yürüme aktivitesiyle birbirinden ayırmıştır. Sadece ivme ölçer sensör verileri kullanan bir diğer çalışma olan [46], Karar Ağacı kullanarak şu seyahat türlerini %70 doğrulukla sınıflandırmıştır: *sabit durma, yürüme, bisiklet, metro, araba, otobüs*.

Diğer çalışmalardan farklı olarak ivme ölçer ve jiroskop sensörleriyle birlikte dönüş sensörü kullanan makale [47] yürüme, koşma, bisiklet, otobüs ve arabadan oluşan 5 seyahat türü üzerinde çalışma yapmıştır. Dönüş sensörü kullanarak cihaz oryantasyonundan bağımsız seyahat türü tanıması yapmıştır. Sensörlerden özellikler çıkartıldıktan sonra mRMR yöntemiyle bu özellikler arasında özellik seçme işlemini gerçekleştirmiş ve %95 doğruluk elde etmişlerdir.

Seyahat türü tanıma alanında derin öğrenme yöntemleri ile çalışma yapmış çok az çalışma mevcuttur. Makale [48] sadece ivme ölçerden elde edilen veriler ile 12 saniyelik pencereler oluşturmuş ve özellik çıkartma yapmadan Tekrarlayan Sinir Ağına (RNN) girdi olarak vermiştir. Sabit durma, yürüme, koşma, bisiklet ve araç olmak üzere 5 sınıfı %94 doğrulukla birbirinden ayırt etmişlerdir. Bir diğer çalışma olan [17] Konvolüsyonel Sinir Ağı (CNN) derin öğrenme yöntemi kullanmıştır. İvme ölçerden toplanan veriler üzerinde Normalleştirme ve Düzleştirme yapıldıktan sonra ağa girdi olarak verilmiştir. Bu makale sabit durma, yürüme, bisiklet, metro, tren, araba, otobüs türlerini sınıflandırmıştır ve %94 doğruluk elde etmiştir.

1.3 Tezin Amacı ve Kapsamı

Bu projede kişilerin günlük aktivite ve seyahat türlerinin birlikte tanınması amaçlanmaktadır. Sınıflandırılan 6 adet aktivite ve 7 adet seyahat türü şunlardır: *sabit durma, yürüme, koşma, merdiven çıkma, merdiven inme, bisiklet, motosiklet, araba, metro, tren, hızlı tren, metrobüs ve tramvay*.

Aktivite ve seyahat türlerini sınıflandırmak için ivme ölçer, jiroskop ve mıknatıs ölçer sensör verileri kullanılmıştır. Projenin amaçlarından biri 13 adet aktivite ve seyahat türünü GPS, GIS, GSM gibi fazla enerji tüketen sensörleri ve verileri kullanmadan iyi bir başarıyla sınıflandırabilmektir. Bu şekilde her mekânda ve internet erişimine ihtiyaç duymadan, daha az enerji tüketimi ile sınıflandırma yapılabilmektedir. Projenin bir diğer amacı, sensörlerle veri toplarken ve sınıflandırma yaparken, cihazın kişinin vücudu üzerindeki konumundan (cep, bel, el, çanta) ve cihaz oryantasyonundan (dikey, yatay) bağımsız aktivite ve seyahat türü tanıması yapmaktır.

Aktivite tanıma için çalışmalarda küçük pencere boyutları (1-5 saniye) kullanılmıştır. Yürüme, merdiven çıkma, koşma gibi günlük aktiviteler kısa pencereler kullanılarak iyi başarıyla sınıflandırılabilmektedir. Diğer yandan seyahat türlerinin yolculuk desenlerini çıkartmak ve iyi başarıyla bunları ayırt etmek için daha uzun pencereler (5-90 saniye) üzerinde sınıflandırma yapmak gerekmektedir. Bu tezde aktivite ve seyahat türlerini birlikte iyi başarıyla tanıyabilmek için ideal pencere boyutunu bulmak amaçlanmaktadır.

Projede farklı yöntemlerle veriden özellikler çıkartılarak yapılan sınıflandırma sonuçları birbirleriyle karşılaştırılmıştır:

- Ham veriyi kullanmak
- Oto Kodlayıcı (derin öğrenme) yöntemi ile özellik çıkartmak
- Bilirkişi yardımıyla zaman alanında özellik çıkartmak

Sınıflandırma yöntemi olarak LSTM derin öğrenme yöntemi tercih edilmiştir. LSTM derin öğrenme yönteminin tercih edilmesinin sebebi, periyodik tekrarlayan aktivitelerin zamanla oluşturduğu desenleri anlayabilmek, bellekte tutabilmek ve

geçmiş zaman bağlaçlarını kullanarak daha iyi sınıflandırma yapabilmektir. LSTM ile elde edilen sonuçlar Naive Bayes, Karar Ağacı, SVM ve Rasgele Orman makine öğrenmesi yöntemleri ile karşılaştırılmıştır. Derin öğrenme ile büyük veri üzerinde çalışabilmek ve çok sınıfı daha iyi başarıyla sınıflandırabilmek amaçlanmaktadır. Bunların yanında derin öğrenmedeki eğitim parametrelerinin, kullanılan pencere boyutlarının, veriye ön işleme uygulamanın aktivite ve seyahat türü tanıma sonucuna etkilerini bulmak amaçlanmaktadır.

Projede basit aktiviteleri sınıflandırmak amaçlanmaktadır, yemek yeme gibi karışık aktiviteler kapsam dışıdır. Seyahat türleri olarak açıkhava (outdoor) seyahat araçları üzerinde çalışılmıştır. Küçük motosiklet (scooter), bina içi mini otobüs, yatak taşıyıcı gibi bina içi seyahat araç türlerini sınıflandırmak bu projenin kapsamı dışındadır.

1.4 Hipotez

Bu projede hipotezler şunlardır:

- Günlük aktivite ve seyahat türleri aynı (tek) problem olarak ele alınabilir ve birlikte derin öğrenme yöntemiyle sınıflandırılarak iyi sonuç elde edilebilir
- Bilirkişi yardımıyla çıkartılan özellikleri ve/veya Oto Kodlayıcı derin öğrenme yöntemiyle çıkartılan özellikleri kullanmak, ham veri kullanarak sınıflandırma yapmaktan daha iyi sonuç üretir
- Sensörlerden elde edilen verilerin büyüklüğü (eksenlerin karelerinin toplamının karekökü) hesaba katılarak cihaz konumundan ve oryantasyonundan bağımsız aktivite ve seyahat türü tanıması yapılabilir

Makalenin ikinci bölümünde aktivite ve seyahat türlerini tanımak, uygulama alanları, hangi sensörler ile veriler toplandığı, sınıflandırma yapmanın adımları detaylı şekilde anlatılmıştır. Bu konuda çalışma yapmış makalelerin tercih ettikleri sınıflandırma yöntemleri, algoritmalar ve fonksiyonlar anlatılmıştır. Daha sonra üçüncü bölümde aktivite ve seyahat türü tanıma adımlarının bu projede nasıl uygulandığı anlatılmaktadır. Veri toplama, veri ön işleme, sınıflandırma adımları

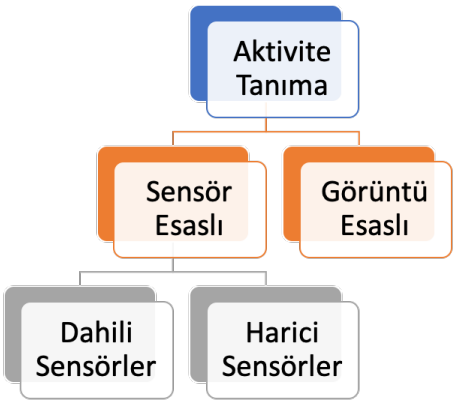
ve tercih edilen yöntemler, algoritmalar ve araçlar anlatılmıştır. Dördüncü bölümde bu projede elde edilen sonuçlar sunulmuştur ve beşinci bölümde bu sonuçlar değerlendirilmiştir ve tartışılmıştır.

2
AKTİVİTE VE SEYAHAT TÜRÜ TANIMA

Bu bölümde aktivite ve seyahat türü tanıma için kullanılan sensörler, uygulama alanları ve sınıflandırma için izlenen genel adımlar anlatılmıştır.

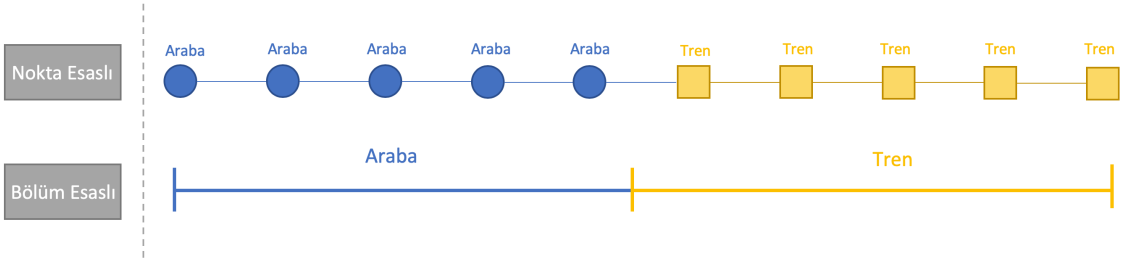
2.1 Aktivite ve Seyahat Türü Tanıma

Aktivite tanıma yürüme, koşma, merdiven çıkma gibi kişinin günlük yaptığı aktiviteleri sınıflandırmaktır. Seyahat türü tanıma ise kişinin metro, tramvay, araba gibi araçlarla yaptığı seyahat türlerini sınıflandırmaktır. Aktivite tanıma yöntemleri görüntü esaslı ve sensör esaslı diye ikiye ayrılabilir (Şekil 2.1). Görüntü esaslı [49] aktivite tanıma için ortamda sabit kameraların olması gerekmektedir. Bu şekilde kişinin (veya kişilerin) yaptığı hareketler sürekli kayıt edilir ve daha sonra bu veriler kullanılarak aktivite tanıma yapılır. Bu yöntem taşınabilirlik, kullanıcı mahremiyeti, fiyat ve uygulanabilirlik acısından cazip değildir. Sensörlerin kullanılması aktivite tanıma için kamera kullanımının meydana getirdiği dezavantajları azaltmaktadır. Sensör esaslı aktivite tanıma iki başlık altında toplanabilir (Şekil 2.1): harici sensörler ve dahili sensörler. Harici sensörlerin [50] kameralar gibi ortama sabit monte edilmesi gerekmektedir. Harici sensörlere örnek pasif kızıl ötesi sensör, mikrofon ve termal kameralar olabilir. Bu yöntemin kullanım alanları için akıllı evler, akıllı ofisler vb. genellikle kapalı alanlar [3] [50] örnek olarak verilebilir. Dahili sensörler kişilerin günlük kullandıkları cihazların içindeki sensörlerdir. Akıllı telefon ve giyilebilir cihazlarda (akıllı saat veya akıllı bileklik) bulunan ivme ölçer, jiroskop, GPS, yakınlık ölçer vb. sensörler dahili sensörlere örnektir. Kişiler akıllı telefonları, akıllı saatleri ve giyilebilir cihazları hep yanlarında taşıdıkları için, bu cihazlardaki dahili sensörleri aktivite tanıma için kullanmak çok uygun hale gelmektedir. Bu projede dahili sensörler olan ivme ölçer, jiroskop, mıknatıs ölçer kullanılarak aktivite ve seyahat türü tanıma gerçeklenmiştir.



Şekil 2.1 Aktivite tanıma yaklaşımları

Seyahat türü tanıma için Şekil 2.2'de gösterilen iki önemli yaklaşımdan bahsedilebilir [51] [52]: nokta (konum) esaslı (point/location based) ve bölüm esaslı (segment based).



Şekil 2.2 Seyahat türü tanıma yaklaşımları

Nokta esaslı yaklaşımda [15] gerçek zamanlı (veya yakın) her sensör okuması için bir seyahat türü sınıflandırması yapılır. Bu yaklaşım için genel olarak GPS, ivme ölçer, jiroskop sensörleri kullanılır. GPS sensör kullanılmadığı zaman sınıflandırma pencereler (sliding window) kullanılarak yapılır [36] [39]. Bölüm esaslı seyahat türü tanımada [33] [37] kişinin seyahati belirli kurallara göre bölümlere ayrılır [40] ve bu bölümler için seyahat türü tanıması yapılır [51]. Çalışmalar seyahati bölümlere ayırmak için GPS nokta kümeleme [33] [34], ani yön değişimi tespiti [33] [34], yürüme aktivitesi ayıracı [45] veya uzun süre bekleme/durma tespiti [16] gibi yaklaşımları esas almışlardır. Bölüm esaslı seyahat türü tanımada GPS, ivme ölçer, jiroskop ve/veya diğer sensörler kullanılır ve ardışık aynı seyahat türleri tek bölüm olarak birleştirilir. Bu projede kayan pencereler kullanılarak (nokta esaslı) aktivite ve seyahat türü tanıma gerçeklenmiştir.

2.1.1 Sensörler

İncelenen makalelerde aktivite tanıma ve seyahat türü tanımada kullanılan sensörleri şu şekilde sıralayabiliriz: *ivme ölçer (accelerometer)* [32], *jiroskop (gyroscope)* [53], *mıknatıs ölçer (magnetometer)* [54], *basınçölçer (barometer)* [37], *yerçekimi sensörü (gravity sensor)* [28], *ışık sensörü (light sensor)* [28], *dönüş sensörü (rotation)* [47], *yakınlık ölçer (proximity sensor)* [28], *GPS* [55]. Bu sensörlerin yanı sıra yazılım ile oluşturulan oryantasyon/yön [44] ve jiroskop [18] sensör verilerini kullanan çalışmalar vardır. Makale [43] seyahat türü tanıma yapmadan önce kişinin hareket edip etmediğini anlamak için GSM verisi kullanmıştır.

İncelenen çalışmalarda en çok kullanılan sensör ivme ölçerdir. İvme ölçer x, y ve z olmak üzere 3 eksende cihazın (dolayısıyla kişinin) ivmesini ölçer. Jiroskop x, y ve z eksenleri etrafında telefonun dönüş hızını ölçer. İvme ölçer kullanmanın getirdiği zorluklardan birisi cihaz oryantasyon ve konumundan bağımsız aktivite ve seyahat türü tanıma yapmaktır. Yapılan çalışmalarda cihaz oryantasyon ve konumundan bağımsız aktivite veya seyahat türü tanıma için birkaç yöntem tercih edilmiştir. Birisi cihaz oryantasyonundan bağımsız ivme ölçer sensör veri özelliklerini (ör. eksenlerin büyüklüğü) kullanmak [7] [19] [40] [45]. Diğer yaklaşım ivme ölçer ile jiroskop ve mıknatıs ölçeri beraber kullanmaktır [11] [20]. Jiroskop sensörü kullanarak telefonun oryantasyonunu da hesaba katmak mümkün hale gelmektedir. Makale [47] ise bu problemi çözmek için dönüş sensörü kullanmayı tercih etmiştir. Makale [44] ivme ölçer ve mıknatıs ölçer sensör verilerinden oryantasyon/yön sensör verisini yazılımsal olarak oluşturmuş ve cihaz oryantasyonundan bağımsız seyahat türü tanıması yapmıştır.

GPS sensörü kullanıcının konumunu tespit eder ve neredeyse tüm akıllı telefonlarda mevcuttur. GPS sensörü yardımı ile kullanıcının çevresi, konumu hesaba katıldığında yapılabilecek aktiviteler [10] ve seyahat türleri [41] sınırlı hale geldiği için daha iyi tanıma başarısı elde etmek mümkün hale gelmektedir. GPS sensör verisine ek olarak otobüs durakları, tramvay yolları, seyahat rotaları

gibi GIS verilerini kullanarak [15] [16] iyi sonuçlar elde eden çalışmalar vardır. Bazı çalışmalar ise GPS sensör kullanımını isteğe bağlı yapmıştır [13] [42].

Bu projede ivme ölçer, jiroskop ve mıknatıs ölçer verileri kullanılmıştır. Seyahat türü tanıma için sık kullanılan GPS veya GIS verileri bu projede kullanılmamıştır. İvme ölçer kullanmak GPS sensörüne kıyasla çok az enerji tüketmektedir [46]. Bu 3 sensör ile veri toplamaya gecikmesiz başlanabilir ve sürekli veri toplanabilir [46]. GPS sensörü seyahat türü tanıma için doğruluğu artırsa da [39] önemli dezavantajları vardır. GPS kullanımında güç tüketiminin fazla olması [39], kapalı alanlarda veya denizde sinyal olmaması [16], hatalı konum verebilmesi [47] gibi problemler bulunmaktadır. GPS kullanan çalışmaların bazıları ulaşım rota bilgilerini içeren GIS verileri kullanmıştır. Çalışma [43] GIS verileri kullanarak seyahat tanımada %10 daha iyi sonuç elde edebilmiştir. GIS verilerini kullanabilmek için uzaktan sunucudan sürekli verilerin alınması gerekmektedir. Bu dış bir sisteme bağlılık ve sürekli internet bağlantısı gerektiren zorlukları getirmektedir. Diğer problem ise her şehrin GIS veri tabanı yoktur ve seyahat rotaları değiştiğinde bu verilerin güncel tutulması gerekmektedir.

Jiroskop ve mıknatıs ölçer gibi ek sensör kullanımının, sadece ivme ölçer sensörü kullanarak elde edilen tanıma başarılarından daha iyi sonuç verdiği çalışmalarda gözlemlenmiştir [28]. Makale [39] daha fazla sensör verisi kullanmanın sonuca olumlu etkisinin olduğunu yazmıştır. Makale [18] ivme ölçer ve mıknatıs ölçer sensör verilerinden jiroskop verisini yazılımsal olarak oluşturmuştur ve bu şekilde kaynak tüketimini azaltmıştır.

2.1.2 Aktiviteler ve Seyahat Türleri

Günlük aktiviteler basit ve karışık aktiviteler olmak üzere ikiye ayrılabilir. Bunları aksiyon ve aktivite diye ayıran çalışmalar da vardır [11]. Basit aktiviteler yürüme, oturma, koşma gibi belirli aksiyonlardan oluşmaktadır. Karışık aktiviteler ise birçok basit aktivitenin bileşiminden oluşur. Örneğin yemek pişirme karışık aktivitesi yürüme, ayakta durma, eğilme gibi basit aktivitelerden oluşmaktadır. Karışık aktivite tanıma bu makalenin kapsamı dışındadır.

İncelenen çalışmalarda tanımlanan aktiviteler şunlardır: *sabit durma* [23], *yürüme* [40], *tempolu koşma (jogging)* [56], *merdiven çıkma* [27], *merdiven inme* [57], *düşme* [2], *oturma* [24], *koşma* [21], *yatma* [19], *zıplama* [19], *yavaş yürüme* [58], *oturarak rahatlama (sit relax)* [22], *yürüyen merdiven ile çıkma* [22], *yürüyen merdiven ile inme* [22], *asansör ile çıkma* [22], *asansör ile inme* [22], *telefon kişide değil* [20], *gezinme (strolling)* [23], *yukarı tırmanma (climb up)* [23], *aşağı inme (climb down)* [23].

İncelenen makalelerde tanımlanan seyahat türleri şunlardır: *bisiklet sürme* [48], *araba sürmek/araba ile seyahat etmek* [42], *otobüs* [37], *tren* [17], *metro* [38], *tramvay* [39], *uçak* [16], *motosiklet* [41], *e-bisiklet* [34], *e-otobüs* [59], *e-yatak taşıyıcı (e-bedpusher)* [59], *küçük motosiklet (scooter)* [59], *vapur (ferry)* [38], *yelkenli tekne (sailing boat)* [16].

Bu projede aktivite ve seyahat türü tanıma birlikte yapılmıştır. Tanınan aktivite ve seyahat türleri şunlardır: *sabit durma, yürüme, merdiven çıkma, merdiven inme, koşma, bisiklet sürme, motosiklet, araba, metro, tren, hızlı tren, tramvay, metrobüs.*

2.1.3 Uygulama Alanları

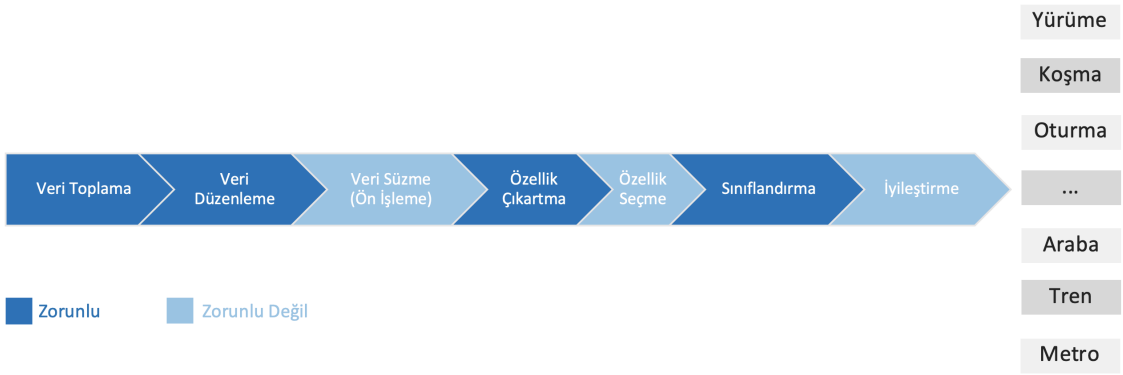
Aktivite tanımanın farklı alanlarda ve senaryolarda birçok uygulama alanı vardır. Kişisel form gelişimi (kalori ölçme, aktivite takip), hastaların hareket takibi [1] [2], akıllı ortamlar [3] [50], kişiye özel önerilerde bulunmak için mobil uygulamalar bunlardan bazılarıdır. Kişinin yaptığı aktivitelere göre harcanan kalori ölçülebilir ve form (fitness) takibi yapılabilir. Kişinin koşma veya yemek yeme gibi yaptığı aktivitesine uygun olarak akıllı telefon uygulaması müzik önerisinde bulunabilir. Aktivite tanıma sağlık alanında [1] [2] aktivite izleme (activity tracking), yardımlı yaşam (assistive living) [60] ve sağlık denetleme (well-being monitored) [61] uygulamalarında kullanılabilir. Bu alanda başka kullanım senaryoları evde yaşlının/çocuğun düşme tespiti veya uzun süre hareketsiz oturulması durumunda hatırlatma yapılması olabilir. Issız bölgelerde meydana gelebilecek araç çarpmaları, bayılmalar kişinin yere düşmesine [62] ve zamanında yardım gelmediği için kalıcı hasarlara veya ölümüne yol açabilir. Bu

nedenle akıllı telefon yardımı ile kişinin düştüğü ve hareketsiz kaldığı anlaşılarak ilgili otoritelere ve/veya kişinin yakınlarına haber verilmesi can kaybını önlemeye yardımcı olabilir.

Seyahat türü tanıma hava kirliliği ölçümü/takibi [12] [13], ulaşım planlama [16] ve trafik yönetimi [14] için çözümler sunabilir. Büyük şehirlerde şehrin hareketliliği, kişilerin ulaşım türü tercihleri incelenerek yeni ulaşım güzergahları oluşturulabilir, kalabalık/ıssız bölgelerde güvenlik önlemlerinin alınması sağlanabilir. Şehirde kullanılan seyahat araçlarının kullanım oranı belirlenerek seyahat türlerinin şehir hava kirliliğine etkileri hesaplanabilir ve önlemler alınabilir. Kişinin seyahat türüne uygun olarak akıllı telefon uygulamaları özel öneri veya reklamlar sunabilir. Örneğin araba süren kişi için en yakın yakıt istasyonlarının listesi gösterilebilir [39] veya otobüs, tramvay kullananlar için etrafta gezilebilecek turistik mekanlar önerilebilir.

2.2 Aktivite ve Seyahat Türü Tanıma Adımları, Yöntemleri

Aktivite ve seyahat türü tanıma yapmak için izlenen adımlar aşağıda Şekil 2.3'de gösterilmiştir. Önce sensörlerden ham veriler toplanır ve yerelde veya sunucuda kaydedilir. Daha sonra bu toplanan verilerden gerekli sensör verileri çıkartılıp belirli dosya formatına çevrilir. Sonra isteğe bağlı olarak veri üzerinde filtreler uygulanabilir ve uç değerler (outlier) elenebilir. Bu tekniğe Düzleştirme de (Smoothing) denmektedir. Daha sonra hazır veri üzerinden özellikler çıkartılır. Bilirkişi tarafından istatistiksel özellikler çıkartılabilir veya derin öğrenme ağı ile özellik çıkartma yapılabilir (CNN [17]). Özellikler çıkartıldıktan sonra girdinin boyutu büyük olursa isteğe bağlı olarak özellik seçme yapılabilir ve farklı yöntemlerle özellikler kümesi seçilir. Bu özellikler eğitim ağına girdi olarak verilir ve sınıflandırma yapılır. Bazı çalışmalar sınıflandırma adımından sonra sonuçlar üzerinde farklı yaklaşımlar kullanarak iyileştirme yapmıştır [13] [40].



Şekil 2.3 Aktivite ve seyahat türü tanıma adımları

2.2.1 Veri Toplama

Birçok çalışmada veri toplama işlemi için [10] [20] Android işletim sistemi yüklü olan akıllı telefonlar tercih edilmiştir. iOS işletim sistemine sahip telefonlar ile de veri toplayan çalışmalar mevcuttur [19]. Çalışma [44] özel donanım ile veriler toplamıştır ve makalede [9] giyilebilir cihazlar kullanılmıştır. Android cihazların daha çok tercih edilmesinin sebepleri arasında ucuz olmaları, piyasada yaygın bulunmaları, kolay programlanabilir olmaları ve uygulamaların arka planda çalıştırılabilmeleri gösterilebilir [23]. Akıllı telefon ile veri toplamak için genellikle uygulama yazılmaktadır. Bu uygulama ara yüzünden hangi sensörlerin kullanılacağı, örnekleme frekansı, hangi aktivitenin yapılacağı seçilir ve veri toplanmaya başlanır [25] [63].

Veriler ilgili cihazın üzerinde belirlenen sensörlerden alındıktan sonra eğer aktivite tanıma modeli telefonun kendisinde çevrimiçi eğitilecekse (online training) veriler yerel hafızada tutulabilir [63]. Eğer aktivite tanıma modeli bilgisayar, sunucu gibi ortamda çevrimdışı eğitilecekse (offline training) bilgisayar ortamına internet [23] veya kablo vb. aracılığıyla aktarılır.

Veri toplanırken bir diğer önemli seçim örnekleme frekansıdır. Örnekleme frekansı verinin saniyede toplanma hızıdır. İncelenen makalelerin çoğu sabit örnekleme frekansı seçmiştir [55] ve çok azı ise [20] [22] dinamik olarak örnekleme frekansı kullanmayı önermiştir. Örnekleme hızının yüksek olmasının doğruluğu olumlu etkilediğini [44] ama enerji tüketimini olumsuz etkilediğini [22] belirtmişlerdir.

Çalışmalar ivme ölçer, jiroskop, mıknatıs ölçer gibi sensörler ile 10Hz-200Hz arası frekansta veri toplamışlardır [39] [59]. GPS sensörü kullananlar 0.5Hz, 1Hz frekansta veri toplamışlardır [42] [59]. Aktivite ve seyahat türü tanıma çalışmalarının yarısından fazlasında 30Hz ve üstü örnekleme frekansı kullanılmıştır.

Toplanan verilerin sayısının ve çeşitliliğinin de eğitim için önemi vardır. Eğitim verisinin çok olması ve çeşitliliğinin fazla olması doğruluğu olumlu yönde etkiler [24] [37] [63].

2.2.2 Veri Düzenleme

Veriler toplandıktan sonra bunlar üzerinde işlem yapmak için verinin belirli dosya düzenine çevrilmesi gerekmektedir. Aynı zaman damgasında toplanan farklı sensörlere ait veriler tek dosyada birleştirilebilir.

Veride sınıf sayı dengesizliği varsa, veri düzenlemeden önce veya sonra belirli yöntemlerle sınıf sayısı dengelenmeye çalışılmalıdır. Aşırı dengesiz sayıda sınıf verileri üzerinde eğitim yapıldığında model sayıca fazla olan sınıfa daha eğilimli olmaktadır [64]. Sınıf sayısını dengelemek için farklı yöntemler bulunmaktadır [65] [66]. İncelenen çalışmaların çoğunluğu bu konuya değinmemiştir. Makale [55] SMOTE tekniği [67] kullanarak verilerin sayısını dengelemeye çalışmıştır.

Veriler düzenlendikten ve sınıf sayıları dengelendikten sonra verinin eğitim, doğrulama, test setlerine ayrılması gerekmektedir. Eğitim seti ile eğitim yapılır ve doğrulama seti kullanılarak model (parametreleri) iyileştirilmeye çalışılır. Modelde en uygun parametreler bulunduktan sonra test seti ile sınıflandırma yapılır ve modelin sonucu belirlenir. Çalışmaların çoğu %80-%20 [17], %60-%40 [36] gibi çeşitli yüzdeler ile veriyi eğitim ve test setlerine ayırmıştır.

2.2.3 Veri Süzme

Veri toplanırken sensörden ve/veya veri toplayan kişiden kaynaklı veriye gürültü eklenebilir. Verideki gürültüleri azaltmak için belirli filtreler uygulanabilir.

Makalede [37] Winsorization[1] tekniği kullanılarak gürültü azaltma yapılmıştır. Bu teknik ile verideki en büyük ve en küçük değerlerin belirli yüzdesi silinir ve yerlerine yeni değerler (yeni en büyük/en küçük vb.) girilir.

Makale [17] ve [44] Kayan Ortalama ile Düzleştirme (Moving Average Smoothing) tekniği uygulamıştır. Bu yöntem ile veri üzerinden belirli pencere boyutu ile gezilir ve penceredeki verinin ortalaması alınarak pencerenin ortasındaki değer güncellenir. Bu şekilde uç değerler pencere içindeki diğer değerlerin ortalaması ile güncellenebilir. Güncelleme işlemi ortalama yerine medyan kullanılarak ta yapılabilir.

Bunlar dışında çalışma [42] Kalman Filtreleme [68], çalışma [37] Gaussian Düzleştirme[2] tekniği kullanmıştır.

2.2.4 Özellik Çıkartma ve Özellik Seçme

Veri toplandıktan ve üzerinde işlemler yapılıp hazırlandıktan sonra ham verinin tamamını sınıflandırma için kullanmak yerine, önemli/ayırt edici özelliklerini çıkartıp onun üzerinde işlem yapmak hız ve enerji tüketimi açısından önemlidir. Özellik sayısını azaltmanın enerji kullanımını [22] ve algoritmanın çalışma süresini azalttığı [10], özellik sayısını artırmanın ise doğruluğu artırdığı [21] [55] incelenen makalelerde belirtilmiştir.

Özelliklerin çıkartılması için veriyi belirli süreli pencerelere ayırmak gerekmektedir. Bu yöntem literatürde kayan pencere tekniği (sliding window technique) [69] olarak adlandırılmaktadır. Pencere boyutu aktivite ve seyahat türlerinin ayırt edici özelliklerini/desenlerini içinde bulundurabilecek kadar büyük olmalıdır. Eğer pencere boyutu çok büyük olursa hem sınıflandırma gecikmesi artar hem de pencerenin birkaç farklı aktiviteyi içinde bulundurabileceği için sonucu olumsuz etkileyebilmektedir. Basit aktivite tanıma yapan çalışmalar 1-60 saniye arası pencere boyutu kullanmışlar ve bu çalışmaların yarısından fazlası 3 saniye ve daha küçük pencereleri tercih etmişlerdir [24].

[1] https://www.statisticshowto.datasciencecentral.com/winsorize/
[2] https://homepages.inf.ed.ac.uk/rbf/HIPR2/gsmooth.htm

Seyahat türü tanıma yapan makaleler 5 saniye ile 10 dakika arası pencere boyutları kullanmış olsalar da bunların çoğu 17 saniye ve daha küçük pencere boyutlarını tercih etmişlerdir [18]. Pencere boyutunu büyütmenin sonuca çok büyük katkıda bulunmadığı makalelerde belirtilmiştir [18] [36]. Pencereler genelde belirli oranda örtüştürülerek kullanılmaktadır. Bu şekilde hem sınıflandırma gecikmesi azalır hem de aktivitenin bulunduğu pencerenin kaçırılma riski azaltılmış olur. Çoğu çalışma %50 oranda pencereleri örtüşürse de makale [28] bunun doğruluğa çok etkisinin olmadığını yazmaktadır. Pencereleri örtüştürmeden kullanan çalışmalar da [13] [39] mevcuttur.

2.2.4.1 Bilirkişi Yardımıyla Özellik Çıkartma

Hangi özelliklerin problemi çözmeye daha etkili olduğunu bilmek konu hakkında detaylı bilgi gerektirmektedir. Bu yüzden veriden önemli, ayırt edici özelliklerin çıkartılması için problem alanını bilen kişiye ihtiyaç duyulmaktadır. Özellikler temel olarak zaman alanında (time domain) ve/veya frekans alanında (frequency domain) elde edilmektedir. Çalışmalarda kullanılan zaman alanındaki özelliklerin bazıları şunlardır: *ortalama (average/mean), standart sapma (standard deviation), ortalama mutlak fark (average absolute difference), değişiklik (variance), standart değişiklik (standard variance), eğiklik (skew), basıklık (kurtosis), dağıntı (entropy), ortak değişiklik (covariance), minimum değer, maksimum değer, mutlak minimum değer, mutlak maksimum değer, kare ortalamalarının kökü (root mean square)*. Frekans alanında kullanılabilecek bazı özellikler ise şu şekilde sıralanabilir: *hızlı Fourier dönüşümü (Fast Fourier Transform) katsayıları, FFT katsayıları ortalaması, FFT katsayıları toplamı, ilk FFT katsayı değeri, FFT katsayıları enerjisi, FFT katsayıları bilgi değeri, spectral roll-off, spectral flux*. Bazı çalışmalar hem zaman hem frekans alanında özellikler çıkarıp kullansa da [24] [36], çalışmaların çoğu zaman alanındaki özellikleri kullanmıştır [27]. Frekans alanında özellikleri hesaplamanın sayısal anlamda daha karışık olduğu [25] ve [28] makalelerinde vurgulanmıştır.

Özelliklerin boyutunu azaltmak için bazı çalışmalar özellik seçme yapmışlardır. Makale [47] mRMR tekniği [70] kullanmıştır. Bu teknik ile özellikler sıralanmıştır

ve en üst 80 tanesi seçilmiştir. Makale [15] benzer sıralama yöntemiyle özellikleri seçmiştir. Makale [55] ACO [71] özellik seçme yöntemi ile çıkartılan özelliklerin boyutunu düşürmüştür.

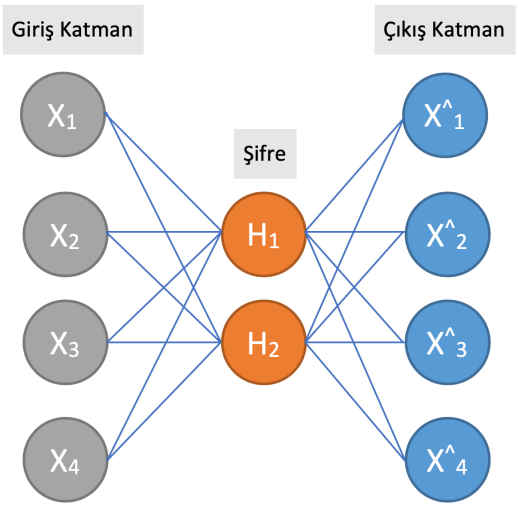
2.2.4.2 Derin Öğrenme ile Özellik Çıkartma

Derin öğrenme yöntemleri kullanılarak bilirkişiye ihtiyaç duymadan, verideki ayırt edici özellikler çıkartılabilir. Oto Kodlayıcılar, CNN gibi ağlar özellik çıkartma ve sınıflandırma için uygun derin öğrenme yöntemleridir.

Oto Kodlayıcı kullanılarak veriden özellikler çıkartılabilir ve herhangi sınıflandırma ağına girdi olarak verilebilmektedir. Oto Kodlayıcı bir sinir ağıdır (Neural Network) ve gözetimsiz öğrenme (unsupervised learning) yöntemidir. Oto Kodlayıcı ağında çıktı olarak girdinin aynısı üretilmeye çalışılmaktadır. Bunu yaparken amaç ağa bazı kısıtlamalar eklemek ve verinin önemli özelliklerini ortaya çıkartmaktır [72]. Makale [73] ve [31] RBM Oto Kodlayıcı [74] kullanarak özellikler çıkartmıştır ve bu özellikleri kullanarak sınıflandırma yapmıştır. Oto Kodlayıcı ağına, veriden önemli özellikleri öğrenmesini zorlamak için, aşağıda bahsedilen birkaç kısıtlama eklenebilir.

2.2.4.2.1 Sade Oto Kodlayıcı

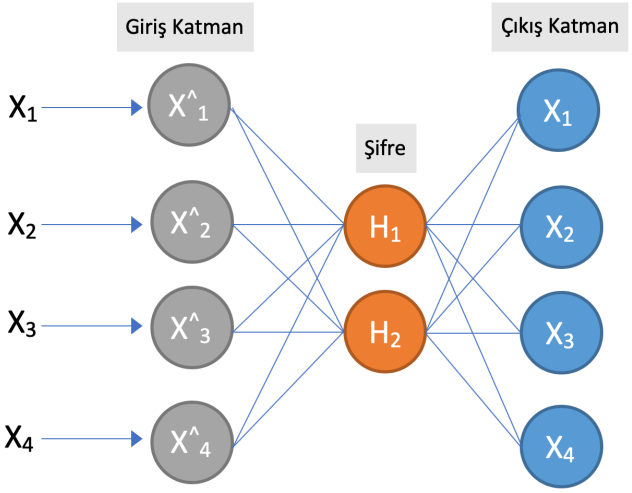
Sade Oto Kodlayıcı (Vanilla Autoencoder) Şekil 2.4'te görüldüğü gibi 3 katmandan oluşmaktadır [75]. Giriş katmanına girdiler verilir ve gizli katmanda bu girdilerin boyutu küçültülür. Çıktı olarak aynı girdi verisi beklenir ve hata ölçülür. Oto Kodlayıcı, ağın girdiyi ezberleyip çıktı olarak vermemesi için gizli katmanın boyutu küçük tutulur ve ağ veriden önemli özellikleri çıkartmaya mecbur bırakılır.



Şekil 2.4 Sade Oto Kodlayıcı

2.2.4.2.2 Gürültü Azaltan Oto Kodlayıcı

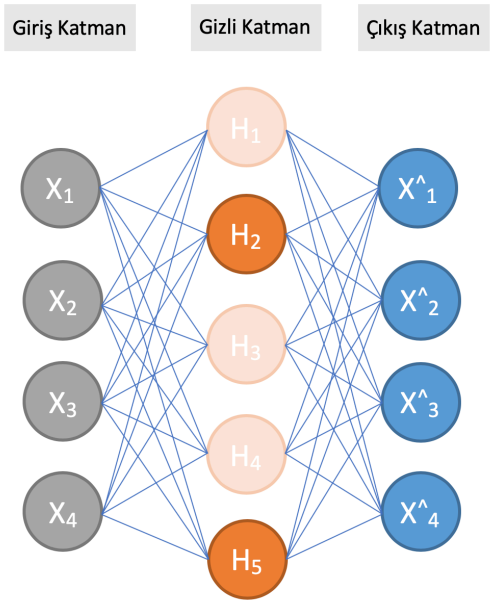
Gürültü Azaltan Oto Kodlayıcı (Denoising Autoencoder) ağı Şekil 2.5'te gösterilen yapıdadır. Sade Oto Kodlayıcının girdinin ezberlenmesini engellemek için kullandığı kısıtlama gizli katman boyutunun küçük tutulmasıdır. Bu kısıtlama ağın ezberlemesini engellemek için yeterli olmayabilir [76]. Gürültü Azaltan Oto Kodlayıcıya rasgele gürültü eklenmiş girdi verilir ve çıktı olarak gürültüsüz veri istenmektedir. Bu şekilde Oto Kodlayıcı gürültülü veriden asıl girdiyi öğrenmeye çalışır ve ağın ezberlemesi zorlaştırılır.



Şekil 2.5 Gürültü azaltan Oto Kodlayıcı

2.2.4.2.3 Seyrek Oto Kodlayıcı

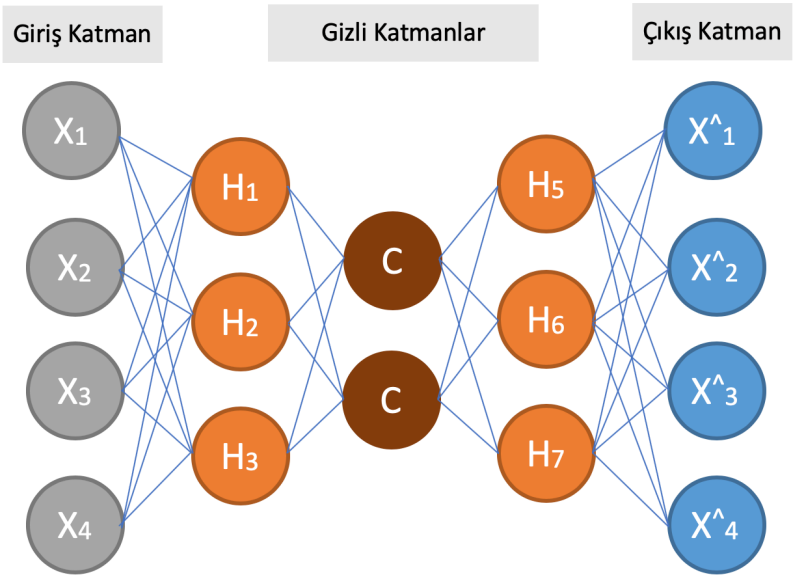
Seyrek Oto Kodlayıcı (Sparse Autoencoder) 3 ve daha fazla katmandan oluşmaktadır. Ağın girdiyi ezberlemesini önlemek için kullanılan kısıtlama gizli katmanlardaki düğümlerin (node) bazılarının aktif olması ve diğerlerinin ise sıfır olmasıdır (Şekil 2.6). Düğümlerin belirli yüzdesi aktif olacağından gizli katmanın boyutu giriş katman boyutuyla aynı veya büyük tutulabilmektedir [77]. Her eğitim döngüsünde aktif olan düğümler rasgele değişir ve Oto Kodlayıcının girdiyi ezberleyip çıktı olarak vermesi engellenmeye çalışılmaktadır.



Şekil 2.6 Seyrek Oto Kodlayıcı

2.2.4.2.4 Derin Oto Kodlayıcı

Derin Oto Kodlayıcı (Deep Autoencoder) birden fazla gizli katmandan oluşmaktadır. Giriş katmanından itibaren gizli katman boyutu azalarak devam etmektedir (Şekil 2.7). Şifre katmanında (en ortadaki özelliğin çıkartıldığı katman) özellikler çıkartılmış ve girdinin boyutu küçültülmüş olmaktadır. Şifre katmanından çıkış katmanına doğru gizli katmanların boyutu artarak devam etmektedir [78] [79].



Şekil 2.7 Derin Oto Kodlayıcı

Oto Kodlayıcılar dışında CNN ve RNN ağları ile özellik çıkartılarak sınıflandırma yapılabilmektedir. Bu ağlara ham veriler girdi olarak verilir ve ağların özellik çıkartması beklenmektedir. CNN kullanan [17] çalışması ham veriyi CNN ağına girdi olarak vermiş ve ağın sonuna sınıflandırma katmanı ekleyerek sınıflandırma modeli oluşturmuştur. [29] ve [30] çalışmaları ise ham veriyi resme çevirerek (spectogram) ağa girdi olarak vermiştir. Benzer bir yaklaşımla ham veriyi RNN [48], LSTM [32] ağına girdi olarak vermiş olan ve ağın çıkarttığı özellikleri kullanarak sınıflandırma yapan çalışmalar vardır.

2.2.5 Eğitim

Aktivite ve seyahat türü tanımada en önemli adımlardan birisi olan eğitim aşamasında sınıflandırma modeli oluşturulmaktadır. Daha önceki adımlarda çıkartılan özellikler pencereler halinde girdi olarak eğitim algoritmasına verilir ve eğitim yapılır. Eğitim yapmak için çevrimiçi ve çevrimdışı olmak üzere iki yaklaşım kullanılmaktadır.

2.2.5.1 Çevrimiçi, Çevrimdışı Eğitim

Çevrimiçi eğitim [37] [63] modelin bir veri seti ile eğitildikten sonra, tekrar yeni ek veriler ile eğitilmeye ve güncellenmeye devam edilebilmesidir. Aktivite ve

seyahat türü tanıma alanında çevrimiçi eğitim akıllı telefon ve giyilebilir cihazların üzerinde yapılmaktadır. Daha önceden eğitilmiş model cihaz üzerinde kişinin yapmakta olduğu aktiviteler ve seyahat türleri ile sürekli eğitilir ve güncel tutulur. Yeni veriler ile eğitim yapılabilmesi için yapılan aktivitelerin kişi tarafından etiketlenmesi gerekmektedir. Bu şekilde eğitim yapıldığında belirli tekrarlamadan sonra kişiselleştirilmiş model ortaya çıkmaktadır. Kişiselleştirilmiş model ile o kişiye ait aktiviteler ve seyahat türleri daha iyi doğruluk oranı ile sınıflandırılabilir.

Çevrimdışı eğitimde [19] [23] sınıflandırma modeli bilgisayar ortamında çevrimdışı eğitilir, eğitilmiş model telefona yüklenir. Akıllı telefonda veya giyilebilir cihazda gerçek zamanlı olarak sensör verileri okunur, işlenir, özellik çıkartılır ve sınıflandırma yerelde (in-situ processing) yapılır.

2.2.5.2 Makine Öğrenmesi Yöntemleri

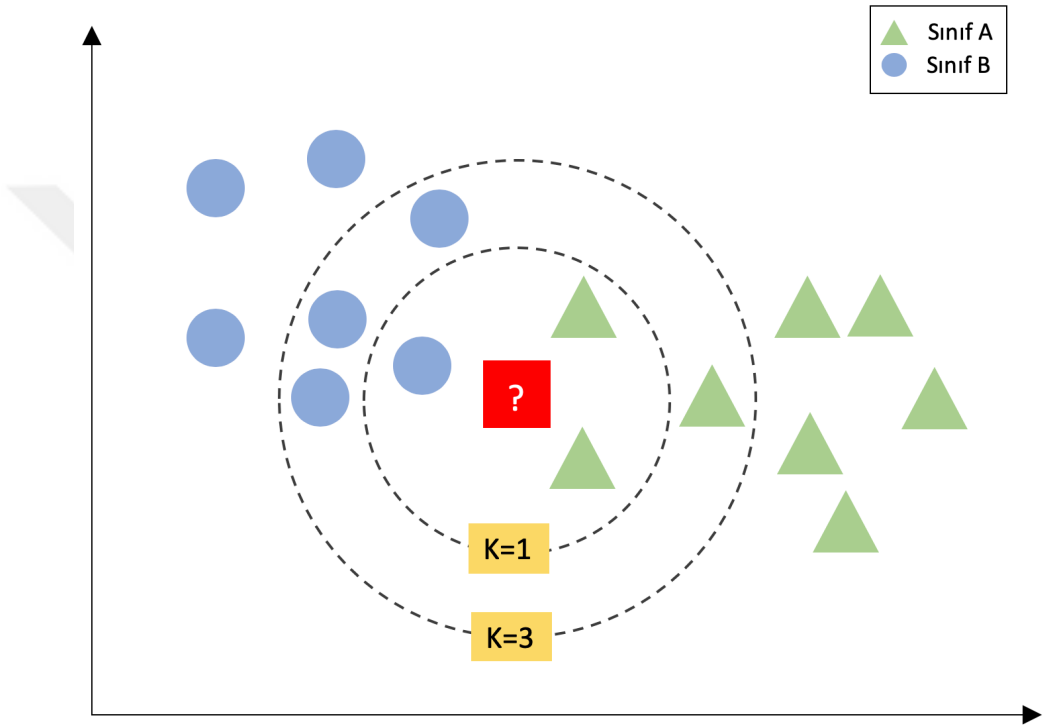
Eğitim aşamasında en önemli seçim probleme uygun sınıflandırma yöntemi seçmektir. Sınıflandırma yöntemleri makine öğrenmesi ve derin öğrenme başlıkları altında incelenebilir. Aktivite ve seyahat türü tanıma çalışmaları en çok makine öğrenmesi yöntemlerini kullanmayı tercih etmişlerdir.

Makine öğrenmesi yöntemleri aktivite ve seyahat türü tanıma alanında çok kullanılmaktadır. Bu yöntemler ile eğitim yapmak için, derin ağlarda olduğu gibi, çok güçlü donanıma ihtiyaç duyulmaz. Hazır araçlar kullanılarak farklı makine öğrenmesi yöntemleri ile hızlı eğitimler yapılabilir ve sonuçlar karşılaştırılabilir. Aktivite ve seyahat türü tanıma konusunda çalışma yapan makaleler en çok şu makine öğrenmesi yöntemlerini kullanmışlardır: *SVM, Rasgele Orman (RF), Karar Ağacı (DT), k-NN, Naive Bayes*. Rasgele Orman yöntemi aktivite tanıma için çok az kullanılmış olsa da seyahat türü tanıma çalışmalarının çoğu [13] [39] tarafından kullanılmıştır ve bu yöntemle iyi sonuçlar elde edilmiştir.

2.2.5.2.1 k-NN

k-NN modeli diğer makine öğrenmesi yöntemlerinden farklı olarak bütün veri seti içinde bulundurmaktadır. Bu yüzden k-NN algoritması eğitim yaparak genelleme (generalization) yapmaz [80]. İki boyutlu ortamda düşünülürse, veri setindeki her

örnek bir nokta ile ifade edilebilir. k-NN benzer örneklerin aynı yerde birbirine yakın olacağını varsayar [81]. Yeni gelen örneği sınıflandırmak için bütün veri set taranır ve K tane benzer/yakın örnekler seçilir (Şekil 2.8). K benzer/yakın örneği bulmak için Öklid Uzaklığı kullanılabilir. Bu K örnek arasında belirli mantığa göre hesaplama/oylama yapılır ve girdinin etiketi belirlenir. K örnek arasında hesaplama mantığı, problemine göre, aritmetik ortalama almak, en çok sayıda bulunan sınıf etiketini saymak vb. olabilir.



Şekil 2.8 k-NN algoritması

Aktivite tanıma alanında [7] [8] [56] gibi birçok makale k-NN kullanmıştır. Seyahat türü tanıma için [36] [47] [59] gibi çalışmalar k-NN kullanmıştır. k-NN modeli bütün veri seti içinde barındırdığı için, veri seti boyutuna göre çok büyük yer tutabilmektedir. Diğer yandan, yeni örneklerle veri seti ve k-NN modeli sürekli güncel tutulabilmektedir. Verideki özelliklerin boyutu çok büyük ise, k-NN performansı düşebilmektedir. k-NN algoritması verideki uç değerlere (outlier) duyarlıdır, eğer uç değerler fazlaysa ve veri süzme ile silinmediyse, k-NN yanlış sonuçlar üretebilmektedir.

2.2.5.2.2 Naive Bayes

Naive Bayes modeli şu olasılık değerlerinden oluşmaktadır: *her sınıfın yüzdesi; her sınıfın verilen her özellik için koşullu olasılığı*. Bu olasılıklar hesaplandıktan sonra, Bayes Teoremi[1] kullanılarak yeni örnek verinin sınıfı tahmin edilebilir (**2.1**).

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)} \quad \textbf{(2.1)}$$

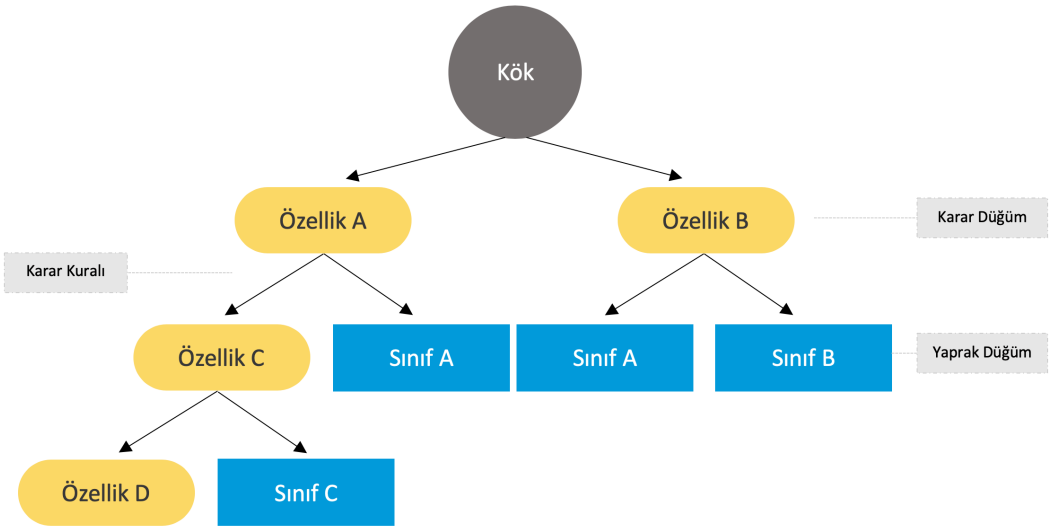
Naive Bayes algoritmasında her girdinin diğerlerinden bağımsız olduğu varsayılmaktadır [82]. Bir diğer varsayım ise girdilerin değerlerinin çan eğrisi (bell curve[2]) şeklinde olmasıdır. Gerçek problemlerde bu varsayımlar doğru olmayabilir ve Naive Bayes iyi sonuç üretmeyebilir.

Aktivite tanıma alanında [10] [11] [20] gibi birçok makale Naive Bayes kullanmıştır. Seyahat türü tanıma için [15] [17] [38] gibi çalışmalar Naive Bayes kullanmıştır.

2.2.5.2.3 Karar Ağacı (DT)

Karar ağacı algoritması (DT), İkili Ağaç gibi (Binary Tree[3]), birden fazla dallı düğümlerden oluşmaktadır. Her karar düğümlerinde (decision node) özellik bulunmakta ve hangi dala gidileceğine kural ile karar verilmektedir. Hangi özelliği kullanarak karar düğümü oluşturulacağına kayıp fonksiyonu ile karar verilmektedir. Karar düğümleri için en iyi özellikler seçildikten sonra yaprak düğümlerinde (leaf node) ağaç sonuçlanmakta ve sınıf etiketi bulunmaktadır (Şekil 2.9).

[1] https://en.wikipedia.org/wiki/Bayes%27_theorem
[2] https://en.wikipedia.org/wiki/The_Bell_Curve
[3] https://en.wikipedia.org/wiki/Binary_tree

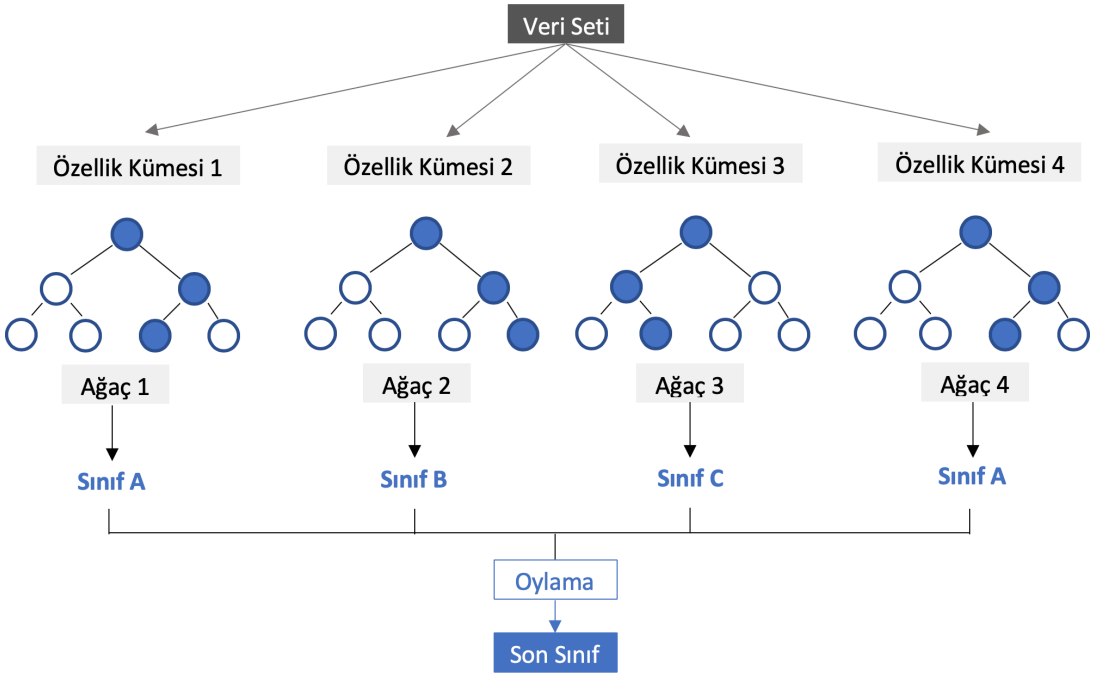


Şekil 2.9 Karar Ağacı algoritması

Aktivite tanıma alanında [22] [23] [24] gibi birçok makale Karar Ağacı kullanmıştır. Seyahat türü tanıma için [41] [43] [44] gibi çalışmalar Karar Ağacı kullanmıştır. Karar Ağacı çok hızlı eğitilen ve çok hızlı sonuç üreten bir algoritmadır. Eğer çok fazla karar düğümleri oluşturulursa ezberlemeye (overfitting) yol açabilir [83].

2.2.5.2.4 Rasgele Orman (RF)

Rasgele Orman bir Bagging algoritma çeşididir. Rasgele Orman algoritması birçok Karar Ağacından oluşmaktadır. Karar Ağacında düğümleri oluşturulurken bütün özellikler arasından en iyi özellikleri seçmek yerine, rasgele özellik kümesi oluşturulur ve her özellik kümesi için ayrı Karar Ağacı oluşturulmaktadır [84]. Aktivite tanıma alanında sadece [57] makalesi Rasgele Orman kullanmıştır. Seyahat türü tanıma için [13] [39] [59] gibi çalışmalar Rasgele Orman kullanmıştır.

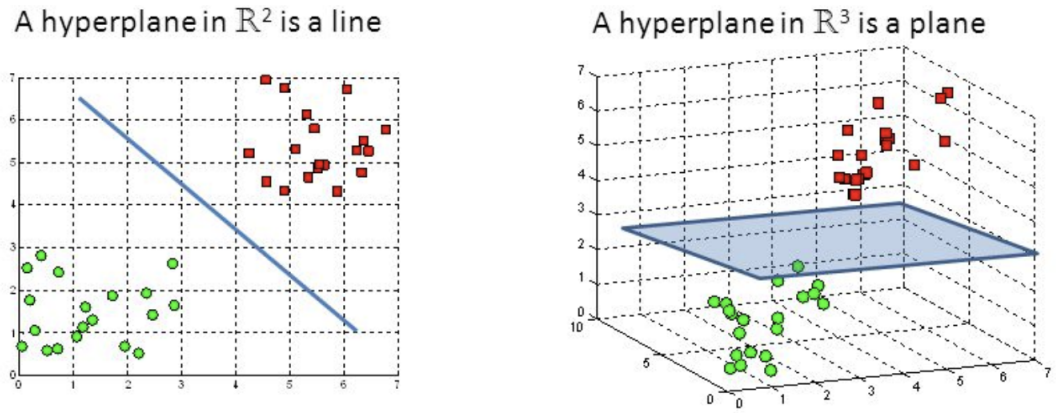


Şekil 2.10 Rasgele Orman algoritması

Yeni örnek için tahmin yapıldığında, her Karar Ağacı ile ara sonuç hesaplanmakta ve bu tahminlerden mod vb. hesap ile sonuç üretilmektedir (Şekil 2.10). Eğer bir problemde Karar Ağacı güzel sonuç üretiyorsa genellikle Rasgele Orman ile daha iyi sonuç üretilebilmektedir [85].

2.2.5.2.5 Destek Vektör Makineleri (SVM)

Destek Vektör Makineleri (SVM) algoritması farklı sınıflara ait verileri çok boyutlu hiper düzlem ile birbirinden ayırmaktadır (Şekil 2.11). Eğitim yapılarak destek vektörler (support vectors) hesaplanmakta ve bu vektörler hiper düzlemi oluşturmaktadır. Hiper düzlem ile ona en yakın verilerin (vektörlerin) arasındaki boşluk ne kadar büyükse SVM o kadar iyi sonuç üretmektedir. Aktivite tanıma alanında [24] [31] [57] gibi birçok makale SVM kullanmıştır. Seyahat türü tanıma için [33] [37] [55] gibi çalışmalar SVM kullanmıştır.



Şekil 2.11 Solda 2 boyutlu alanda 1 boyutlu çizgi olan hiper düzlem. Sağda 3 boyutlu alanda 2 boyutlu hiper düzlem [86]

2.2.5.3 Derin Öğrenme Yöntemleri

Derin öğrenme yöntemleri veya derin ağlar, Yapay Sinir Ağlarının çeşidi olup karmaşık problemleri çözebilme kapasitesine sahip ağlardır. Klasik yapay sinir ağları birkaç katmandan (giriş katman, gizli katmanlar, çıkış katman) oluşmaktadır. Her katmanda girdi hakkında bir şeyler öğrenilmektedir ve bir sonraki katmanda girdinin daha karışık özellikleri çıkartılmaya çalışılmaktadır. Katmanlar ve katmanlardaki nöron sayısı arttıkça ağın eğitilmesi çok uzun sürebilirken derin ağlarının eğitilmesini engelleyen VGP[1] problemi ortaya çıkabilmektedir. Bu sebeplerden dolayı Yapay Sinir Ağları son on beş seneye kadar sınırlı problem alanlarında kullanılıyordu. Bilgisayarlardaki GPU teknolojisinin gelişmesi ağ eğitim süresi problemini çözmüştür ve bazı derin öğrenme yöntemlerindeki ilerleme VGP problemine çözüm getirmiştir.

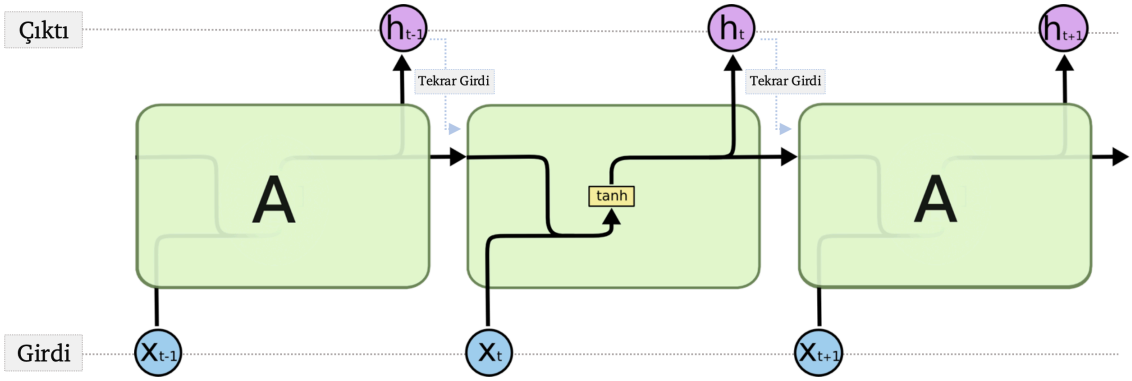
Derin ağların eğitim süresi çok uzun süreler alabildiği için süreyi kısaltmak adına güçlü donanımların kullanımına ihtiyaç duyulmaktadır. Dolayısıyla derin öğrenme ile eğitimin şu anki mobil cihaz donanımları üzerinde yapılması mümkün değildir. Sadece bilgisayar ortamında eğitilmiş hazır modeller telefon üzerinde gerçek zamanlı aktivite sınıflandırmak için kullanılabilmektedir.

[1] https://en.wikipedia.org/wiki/Vanishing_gradient_problem

Aktivite tanımada derin öğrenme yöntemleri belirli çoğunlukta kullanılmış olsa da [32] [54] [87] seyahat tanıma alanında çok az kullanılmıştır [17] [48]. Derin öğrenme yöntemlerinin her birisinin farklı çalışma prensipleri vardır. Çalışmaların en çok kullandığı RNN ve CNN ağları aşağıda anlatılmıştır.

2.2.5.3.1 Tekrarlayan Sinir Ağları (RNN)

Tekrarlayan Sinir Ağları [88], geri beslemeli sinir ağı, zamana bağlı değişen verileri olan problemleri çözmek için kullanılan ağdır. Şekil 2.12'de gösterildiği gibi bir önceki zaman adımındaki çıktı bir sonraki zaman adımında tekrar girdi ile beraber ağı beslemekte ve bellek oluşturmaktadır. Çok zaman adımı öncesini ezberleyebilecek belleğe sahip RNN ağı oluşturulabilir ama bu VGP problemine yol açmaktadır (Backpropagation Through Time[1]). VGP problemini çözmek için GRU veya LSTM gibi kapılı RNN ağları kullanılmaktadır.

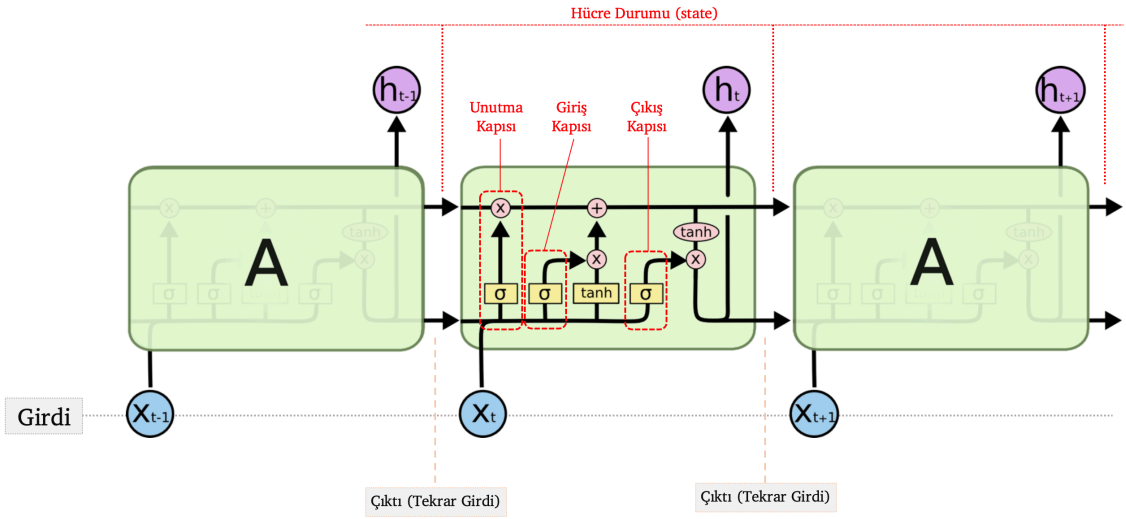


Şekil 2.12 Basit Tekrarlayan Sinir Ağı [89]

LSTM ağı RNN ağının bir çeşididir. Girdiyi alarak bellek oluşturur, çıktı üretir ve çıktı tekrar bir sonraki zaman adımında girdi ile beraber ağı beslemektedir. LSTM bu şekilde her zaman adımında iki girdi ile ağ belleğini sürekli güncellemektedir. Belleğe yeni verinin ekleneceğine veya eski verilerin unutulacağına LSTM kapıları karar vermektedir. LSTM ağında 3 farklı kapı vardır: *giriş kapısı, unutma kapısı, çıkış kapısı* (Şekil 2.13). Giriş kapısı yeni girdinin ne kadarının hafızaya ekleneceğine, unutma kapısı mevcut hafızanın ne kadarının unutulacağına ve çıkış kapısı da yeni hafızanın ne kadarının bir sonraki hücreye aktarılacağına karar

[1] https://en.wikipedia.org/wiki/Backpropagation_through_time

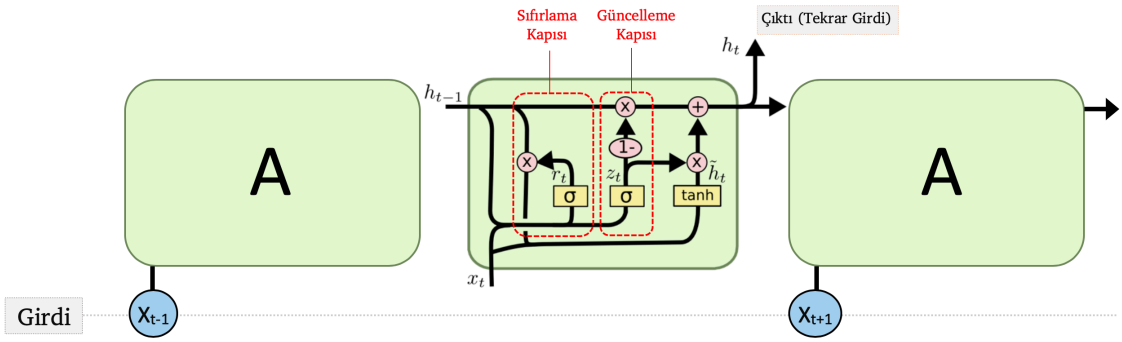
vermektedir [90]. Bu kapılar ile bellek güncel tutulmakta ve VGP problemi çözülmektedir.



Şekil 2.13 LSTM ağı kapıları [89]

LSTM ağı girdi olarak 3 boyutlu vektör almaktadır. Girdi sayısının yanında (1.boyut) her örnek verinin kaç zaman adımından oluştuğunu (Timesteps, 2.boyut) ve her zaman adımında kaç tane özelliğin bulunduğunu (Features, 3.boyut) ifade eden parametrelerin girilmesi gerekmektedir.

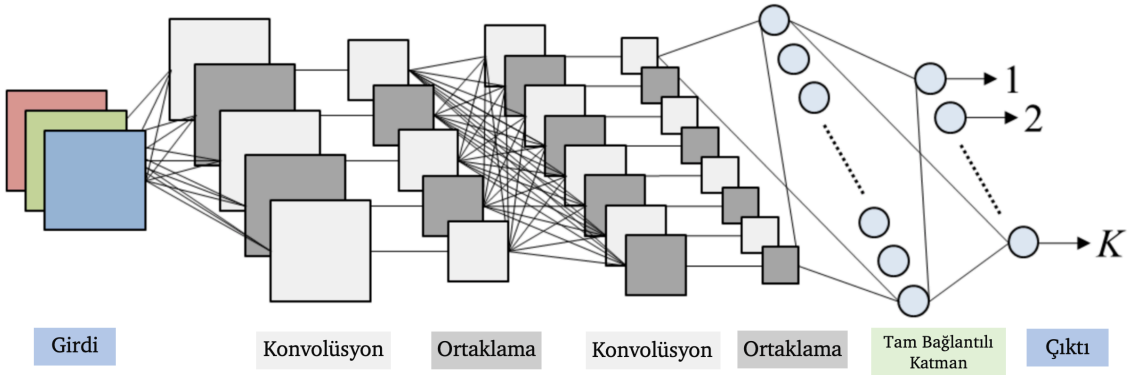
LSTM ile GRU arasındaki önemli fark kullanılan kapılardır. LSTM'de giriş, unutma ve çıkış kapıları vardır. GRU ağında LSTM ağındaki giriş ve unutma kapıları yoktur ve bunun yerine güncelleme kapısı mevcuttur. LSTM ağ hücresinde bellek ve sonraki adım için çıktı olmak üzere iki ayrı veri çıkışı varken GRU'da bunlar birleşiktir (Şekil 2.14). Bu sebepten LSTM ağında belleğin bir bölümü bir sonraki zaman adımına girdi olarak verilirken GRU ağında belleğin tamamı bir sonraki zaman adımında girdi olarak ağı beslemektedir [89].



Şekil 2.14 GRU ağı kapıları [89]

2.2.5.3.2 Konvolüsyonel Sinir Ağları (CNN)

Yann Lecun et al [91] tarafından ortaya atılan Konvolüsyonel Sinir Ağlarında birçok filtre ile veri üzerinde farklı desenler aranmaktadır. CNN ağı birçok Konvolüsyon (Convolution) ve Havuzlama (Pooling) katmanlarından oluşabilmektedir (Şekil 2.15). Konvolüsyon katmanında veri üzerinde bulunan desenler (özellikler) çok büyük boyutta olacağından daha sonra Havuzlama katmanında bu özelliklerin boyutu küçültülmektedir. CNN eğitim yapmak için Backpropagation[1] algoritması kullanmakta ve VGP problemi mevcuttur. Bu problemi çözmek için ReLU[2] aktivasyon fonksiyonu kullanılmaktadır.



Şekil 2.15 Konvolüsyonel Sinir Ağı katmanları [92]

[1] https://en.wikipedia.org/wiki/Backpropagation
[2] https://en.wikipedia.org/wiki/Rectifier_(neural_networks)

Aktivite ve seyahat türü tanıma çalışmaları CNN ağını özellik çıkartmak için kullanmışlardır [17] [54] [87]. Ağın sonunda ise sınıflandırma yapmak için tam bağlantılı katman (fully connected layer) eklenmektedir.

2.2.5.3.3 Aktivasyon Fonksiyonları

Yapay Sinir Ağları, CNN, RNN gibi ağlarda her katmandaki nöronların çıktısını almak için aktivasyon fonksiyonları kullanılır. Aktivasyon fonksiyonları girdi almakta ve algoritmasına bağlı olarak çıktı üretmektedir. Bu fonksiyonlar doğrusal (linear) ve doğrusal olmayan (non linear) olmak üzere ikiye ayrılırlar. Doğrusal fonksiyonlar basittir, sıfır veya bir bilinmeyenden oluşur (**2.2**). Girdi ve çıktı grafik olarak çizildiğinde düz çizgi ile ifade edilir[1]. Fonksiyonun türevi sabit sayıdır.

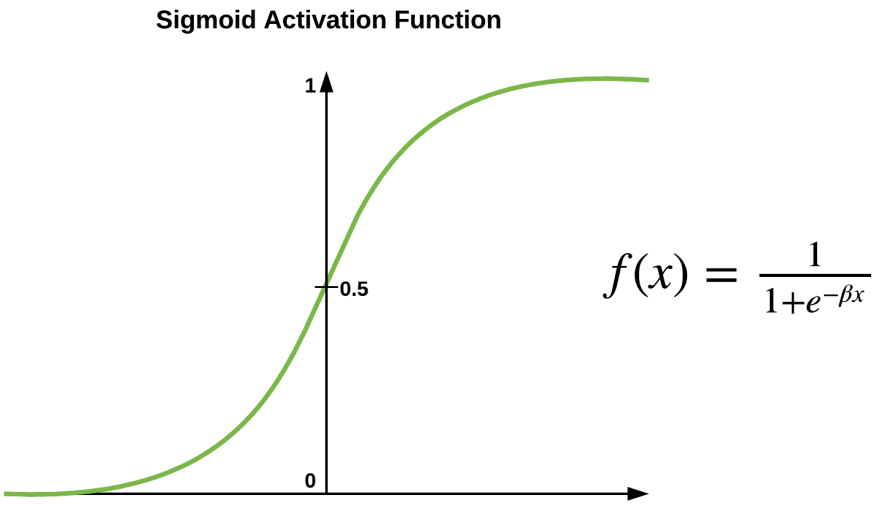
$$f(x) = ax + b \qquad \textbf{(2.2)}$$

Doğrusal olmayan fonksiyonlar en az bir bilinmeyenden oluşurlar. Fonksiyonun türevi sabit değildir, bilinmeyen değere bağlı olarak belli aralıkta değişmektedir[2]. Fonksiyon iki boyutlu düzleme çizildiğinde eğri bir çizgi oluşturmaktadır. Doğrusal olmayan fonksiyonlar kullanılarak daha karmaşık problemler çözülebilmektedir. Sinir ağlarında, özellikle gizli katmanlarda, doğrusal olmayan fonksiyonlar kullanılmaktadır.

Aktivite ve seyahat türü tanıma için sınıflandırma ağları oluşturulurken en çok kullanılan aktivasyon fonksiyonları şunlardır: *Sigmoid, Tanh, ReLU, Softmax.*

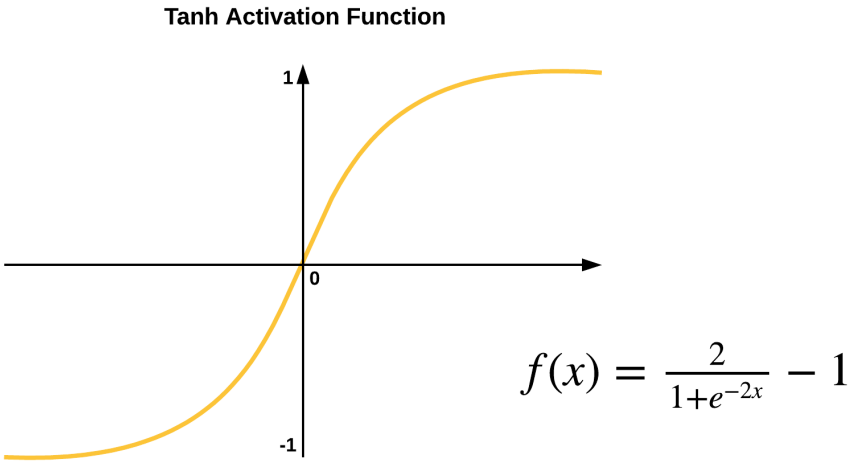
Sigmoid aktivasyon fonksiyonu[3] Şekil 2.16'da görüldüğü gibi 0 ve 1 değerleri arasında çıktı üretmektedir. Fonksiyonun türevini hesaplamak basit ve maliyetsizdir. Sigmoid aktivasyon fonksiyonu sinir ağlarda gizli katmanlarda en çok kullanılan fonksiyonlardan biridir.

[1] http://www.columbia.edu/itc/sipa/math/linear.html
[2] https://study.com/academy/lesson/nonlinear-function-definition-examples.html
[3] https://en.wikipedia.org/wiki/Sigmoid_function



Şekil 2.16 Sigmoid aktivasyon fonksiyonu [93]

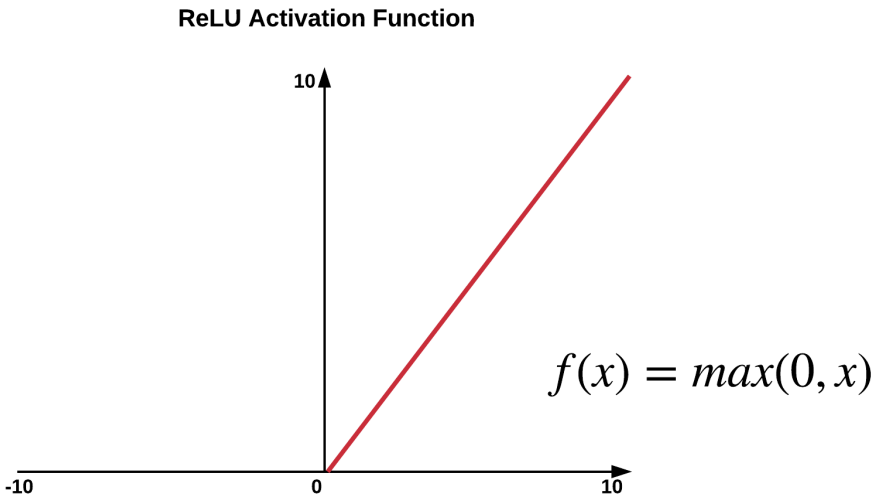
Tanh aktivasyon fonksiyonu[1] Şekil 2.17'de görüldüğü gibi çıktıyı -1 ve 1 değerleri arasına sıkıştırmaktadır. Fonksiyonun türevini hesaplamak basit ve maliyetsizdir. Sinir ağlarda gizli katmanlarda en çok kullanılan aktivasyon fonksiyonlardan diğeri de Tanh fonksiyonudur.



Şekil 2.17 Tanh aktivasyon fonksiyonu [93]

[1] https://en.wikipedia.org/wiki/Hyperbolic_function

ReLU aktivasyon fonksiyonu sıfırdan küçük değerler için 0, diğer değerler için ise girdinin aynısını çıktı olarak vermektedir (Şekil 2.18). Dolayısıyla fonksiyonun türevi 0 veya 1 olmaktadır. ReLU fonksiyonu ve türevi hesaplama açısından maliyetsizdir. ReLU aktivasyon fonksiyonu özellikle CNN ağında VGP problemini çözmek için kullanılmaktadır. ReLU VGP problemini çözse de "Dying ReLU" [94] problemine yol açabilmektedir. Bu problemi çözmek için Leaky ReLU[1] aktivasyon fonksiyonu kullanılabilir.



Şekil 2.18 ReLU aktivasyon fonksiyonu [93]

Softmax aktivasyon fonksiyonu[2] çıktı olarak 0 ve 1 arasında değer üretmektedir. Sigmoid aktivasyon fonksiyonundan farklı olarak çıktıların toplamı 1'e eşit olmaktadır. Bu sebepten Softmax aktivasyon fonksiyonu sınıflandırma ağlarının çıkış katmanında kullanılmaktadır. Ağın çıkış katmanında her sınıfın yüzdesi bu fonksiyonu kullanarak ifade edilebilir.

2.2.5.3.4 Kayıp Fonksiyonları

Kayıp fonksiyonları eğitilmiş modelin ne kadar hatalı sonuç ürettiğini ölçmek için kullanılmaktadır. Her eğitimden sonra kayıp fonksiyonu ile hata hesaplanmakta

[1] https://en.wikipedia.org/wiki/Rectifier_(neural_networks)
[2] https://en.wikipedia.org/wiki/Softmax_function

ve sinir ağındaki ağırlıklar bu hataya göre güncellenmektedir. Ortalama Karesel Hata (MSE), Ortalama Mutlak Hata (MAE), Categorical Cross-Entropy en çok tercih edilen kayıp fonksiyonlarından bazılarıdır.

MSE kayıp fonksiyonu[1] iki değer arasındaki hatayı (**2.3**)'de gösterildiği gibi hesaplamaktadır. Modelin ürettiği çıktı değeri ile asıl beklenen çıktı değeri kullanılarak hata hesaplanır. Eğer çıktı ile girdi benzer şekilde sayısal değer ise, regresyon problemleri gibi, MSE kayıp fonksiyonu kullanılabilir. MSE'de hatanın yönünün önemi yoktur. Örneğin, beklenen değer 0.5 iken çıktının 0.4 veya 0.6 olması aynı oranda hata olarak değerlendirilmektedir. MSE kayıp fonksiyonunda hatanın karesi alındığından bu fonksiyon verideki uçdeğerlere duyarlıdır. Eğer veri setinde uçdeğerler fazla ise MSE iyi sonuç üretmeyebilir.

$$\frac{1}{n} \times \sum_{j=1}^{n} (y_j - o_j)^2 \tag{2.3}$$

MAE kayıp fonksiyonu[2] iki değer arasındaki hatayı MSE fonksiyonuna benzer şekilde hesaplamaktadır. Çıktı olarak üretilen değer ile beklenen değer arasındaki fark hesaplanır ve mutlak değeri alınır (**2.4**). MAE kayıp fonksiyonunda hatanın karesi alınmadığı için MSE fonksiyonu gibi verideki uçdeğerlere çok duyarlı değildir.

$$\frac{1}{n} \times \sum_{j=1}^{n} |y_j - o_j| \tag{2.4}$$

Categorical Cross-Entropy kayıp fonksiyonu [95] sınıflandırma problemlerinde Softmax aktivasyon fonksiyonu ile birlikte kullanılmaktadır (**2.5**). MSE ve MAE fonksiyonlarından farklı olarak hatanın yönü önemlidir. Örneğin, beklenen değer 0.5 iken 0.4 veya 0.6 çıktıları için farklı oranda hata değeri üretmektedir.

$$-\log\left(f(s)_i\right) \tag{2.5}$$

[1] https://en.wikipedia.org/wiki/Mean_squared_error
[2] https://en.wikipedia.org/wiki/Mean_absolute_error

2.2.5.3.5 Eniyileştirici Algoritmaları

Modelde hatayı azaltmak için ve ağırlıkları bu doğrultuda güncellemek için eniyileştirici (optimizer) algoritmaları kullanılmaktadır. Eniyileştirici algoritması arama tekniği olup, küresel (global) minimum hatayı bulmaya çalışır. Sinir ağları genellikle Backpropagation algoritması ile eğitilmektedir. Backpropagation ile her katmandaki nöronun hataya katkısı (gradient), aktivasyon fonksiyonunun türevi kullanılarak hesaplanmaktadır. Eniyileştirici algoritması bu gradyanları kullanarak her katmandaki nöronların ağırlıklarını doğru yönde güncellemeye çalışır. Aktivite ve seyahat türü tanımada en çok kullanılan eniyileştirici algoritmaları SGD, Adadelta ve Adam'dır.

SGD (Stochastic Gradient Descent) algoritması[1] birinci dereceden türev kullanarak ağın ağırlıklarını güncellemektedir. Ağırlıkları güncellemek için gradyan, öğrenme hızı (learning rate), ivme (momentum) değerlerini kullanmaktadır. SGD algoritmasında en önemli parametre öğrenme hızıdır. Öğrenme hızı büyük olduğu zaman ağ hızlı eğitilmekte ama küresel minimum hatayı kaçırabilme olasılığı mevcuttur. Diğer yandan çok küçük öğrenme hızı eğitim süresini uzatmakta ve yerel minimumda takılıp kalabilmektedir. İvme parametresi hatadaki dalgalanmaları azaltarak küresel minimuma hızlı ulaşmak için kullanılabilir. SGD eniyileştirici algoritması her eğitim örneği için ağın ağırlıklarını güncelleyebilir ama bu hata değerinin sürekli çok dalgalanmasına sebep olabilmektedir. Bu yüzden küresel minimum hatayı bulamayabilir. Bunun yerine her yığından (batch) sonra ağ ağırlıklarını güncellemek tercih edilmektedir. Yığın boyutu (batch size) olarak genelde 32-256 arası değerler kullanılmaktadır.

Adadelta eniyileştirici algoritması [96] SGD gibi tek öğrenme hızı kullanmak yerine her özelliğe ait parametreler için farklı öğrenme hızı kullanmaktadır. Bu şekilde sık kullanılan özelliğe ait parametrelere daha küçük güncellemeler yapılmakta ve az kullanılan özelliklere ait parametrelere büyük güncellemeler yapılmaktadır. Adadelta özellikle seyrek veriler (sparse data) için uygun bir

[1] https://en.wikipedia.org/wiki/Stochastic_gradient_descent

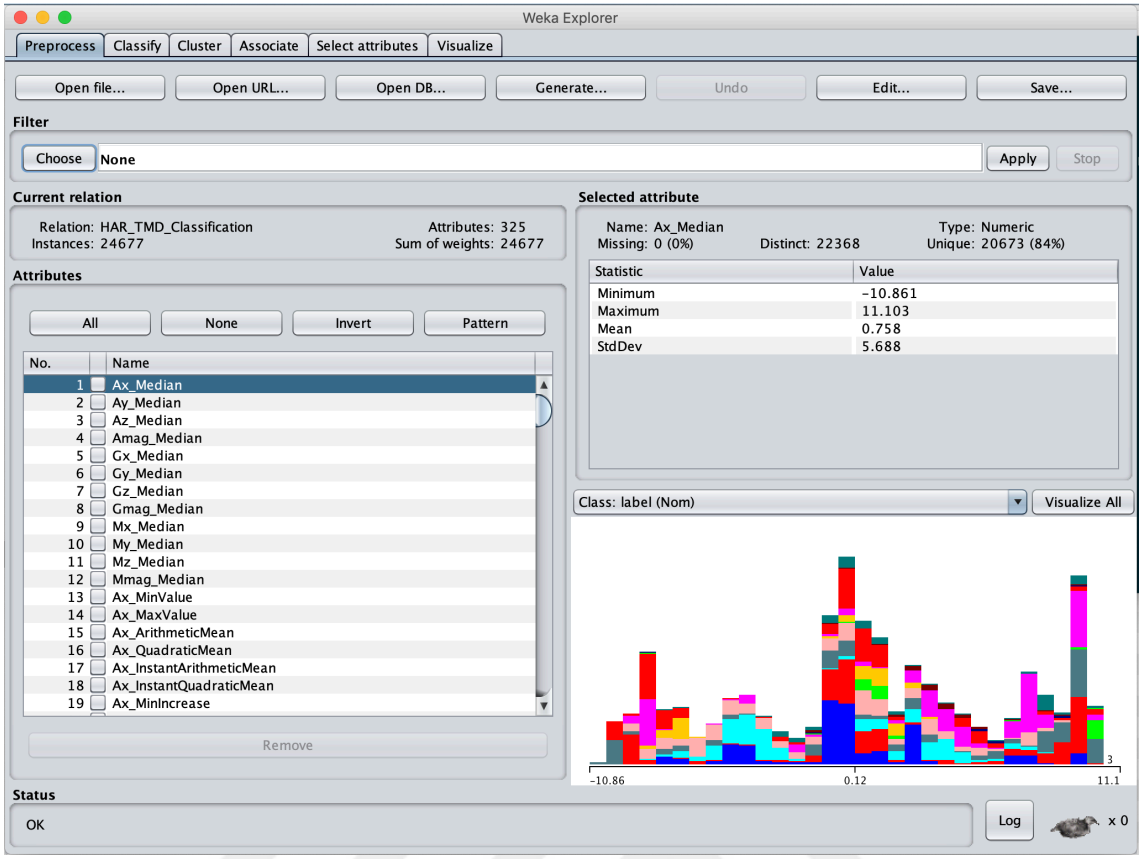
algoritmadır. Adadelta eniyileştirici algoritması kullanılırken parametreleri varsayılan değerleriyle kullanmak tercih edilmektedir.

Adam eniyileştirici algoritması Adadelta gibi uyarlanabilir öğrenme hızı parametresi seçmektedir. Buna ek olarak benzer mantıkla uyarlanabilir ivme kullanmaktadır. Bu şekilde her özelliğe ait parametrelerin ayrı öğrenme hızı, ivmesi olabilmekte ve birbirinden bağımsız güncellenebilmektedir. Adam eniyileştirici algoritması kullanılırken parametreleri varsayılan değerleriyle kullanmak tercih edilmektedir.

2.2.5.4 Araçlar

İncelenen çoğu çalışma modeli eğitim yapmak için hazır araçlardan yararlanmışlardır. Hazır araçlar ile veri dağılımı, özelliklerin grafikleri, uçdeğerler eğitimden önce gözlemlenebilmektedir. Veri süzme ve/veya Normalleştirme gibi işlemler de yapılabilir. Makine öğrenmesi veya derin öğrenme yöntemleriyle eğitim yapılabilmekte ve sonuçlar alınabilmektedir. Zaman kazanmak ve çalışmada önerilen yöntemin diğer yöntemler ile hızlı karşılaştırılabilmesi için hazır araçlar önemli göreve sahiptirler.

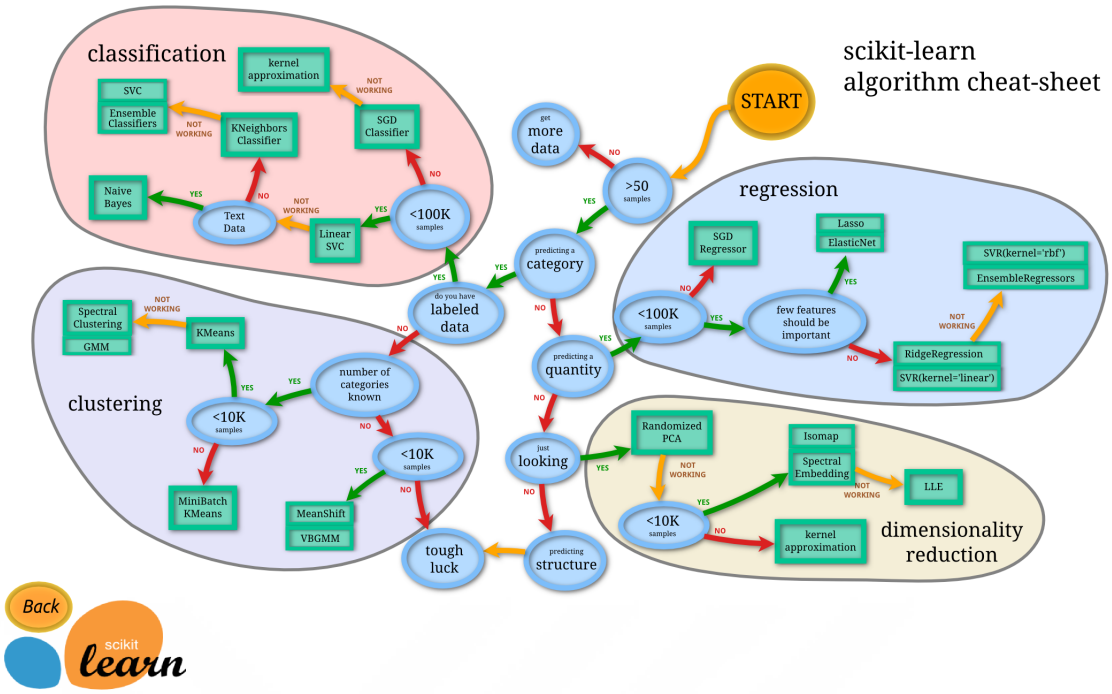
Aktivite tanıma alanında çalışma yapanlar en çok WEKA aracını kullanmışlardır [10] [23]. Seyahat türü tanıma çalışmalarında da en çok tercih edilen araç WEKA'dır [13] [39] [41]. WEKA [97] aracı ile veri özelliklerinin histogram grafiği çıkartılabilmekte ve bazı istatistiklerine (minimum, maksimum, standart sapma, ortalama) göz atılabilmektedir (Şekil 2.19). Veri üzerinde Normalleştirme ve veri süzme işlemleri yapılabilmektedir. En önemlisi birçok makine öğrenmesi yöntemleri ile kod yazmadan eğitim yapılabilmektedir.



Şekil 2.19 WEKA aracı

Scikit-learn[1] Python programlama dili kütüphanesi makine öğrenmesi yöntemlerini kod yazarak uygulamak için kullanılabilecek bir başka araçtır. Veri ön işleme, sınıflandırma, kümeleme, boyut küçültme gibi işlemler kütüphane yardımıyla kolay ve hızlı yapılabilmektedir (Şekil 2.20).

[1] https://scikit-learn.org/stable/index.html



Şekil 2.20 Scikit-learn aracı[1]

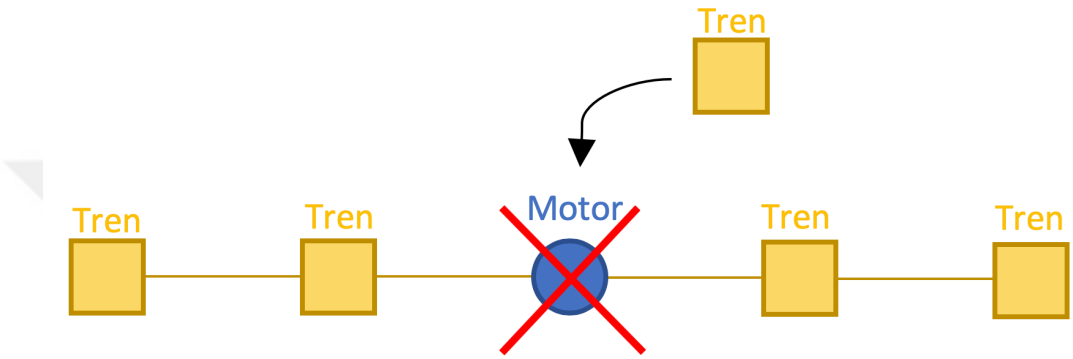
Derin öğrenme yöntemi uygulamak için makale [17] Tensorflow aracını kullanmıştır. Tensorflow[2] Google tarafından geliştirilmiş, özellikle derin öğrenme yöntemlerini uygulamak için kullanılabilecek araçtır. Kurulumu basit olup Python, C++ dilleri ile Tensorflow kütüphane kodları kullanılabilmektedir. Eğitimler GPU kullanılarak daha hızlı şekilde yapılabilmektedir. Eğitim yapıldıktan sonra Tensorflow modelleri Android, iOS, Raspberry Pi cihazları üzerinde de çalıştırılabilir.

Keras [98] Python programlama dili ile yazılmış kullanımı kolay bir araçtır. Keras arka planda Tensorflow, Theano veya CNTK araçlarını kullanarak kullanıcıya daha kolay kullanım ara yüzü sunmaktadır. Keras ile yapay sinir ağı, CNN, RNN gibi ağları oluşturmak kolaydır. Keras hazır kayıp fonksiyonlar, eniyileştirici algoritmalar, değerlendirme metrikler, çeşitli katmanlar (dropout, gürültü vb.) gibi kullanıcıya birçok hazır özellikler sunmakta ve parametreler ile bu kullanılan özellikler ihtiyaç/deneye göre özelleştirilebilmektedir.

[1] https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html
[2] https://www.tensorflow.org

2.2.6 İyileştirme

Sınıflandırma aşamasından sonra ardışık sonuçlar üzerinde iyileştirme/düzeltme yapılabilir. Örneğin Şekil 2.21'de görüldüğü gibi, beş tane tren etiketi içerisinde ortadaki sınıf motosiklet olarak tahmin edildiyse, iki tren aktivitesi arasında (pencere boyutuna bağlı olarak) motosiklet aktivitesi olamayacağı için, yanlış tahmin edilmiş demektir. İyileştirme tekniği kullanılarak bu yanlış tahmin düzeltilebilir ve yerine doğru etiket yazılabilir.



Şekil 2.21 Sınıflandırma sonucu üzerinde iyileştirme yapmak

Bazı seyahat türü tanıma makaleleri farklı yaklaşımlar kullanarak sonuç üzerinde iyileştirme yapmıştır [13] [18] [38] [40]. Makale [18] oylama (voting) yaparak iyileştirme yapmıştır. Makale [13] DHMM[1] yöntemi kullanarak iyileştirme uygulamıştır. Sınıflandırma aşamasından sonra iyileştirme uygulayan çalışmalar bunun doğruluğa olumlu etkisi olduğunu belirtmişlerdir [18] [38] [40].

[1] https://en.wikipedia.org/wiki/Hidden_Markov_model

3
KULLANILAN VERİ SETİ, ARAÇLAR VE YÖNTEMLER

Bu bölümde aktivite ve seyahat türü tanıma için izlenen adımların bu projede nasıl uygulandığı anlatılmıştır.

3.1 Veri Seti ve Ek Veri Toplama

Projede kullanılan veri setinin detayı, sensörler ve ek veri toplamak için geliştirilen Android uygulaması bu bölümde sırasıyla detaylı anlatılmıştır.

3.1.1 Veri Seti, Aktiviteler ve Sensörler

Bu projede veri seti olarak HTC [99] tarafından hazırlanan [18], toplamda 100GB boyutunda olan veri setinin halka açık 17GB boyutundaki verisi kullanılmıştır. Veri seti toplamda 150 öğrenci, 74 çalışan ve stajyerlerin katılımıyla toplanmıştır [18]. Veri setini toplayan kişilerin cinsiyeti %60 erkek ve %40 kadındır, kişilerin yaş aralığı 20-63 yaştır. Bu veri setinde bulunan günlük aktiviteler ve seyahat türleri şunlardır: *Sabit Durma (Still), Yürüme (Walk), Koşma (Run), Bisiklet Sürme (Bicycle), Motosiklet Sürme (Riding Motorcycle), Araba (Car), Otobüs (Bus), Metro (Metro), Tren (Train), ve Hızlı Tren (High Speed Rail)*. Bu aktivite verileri şu cihaz konumlarında toplanmıştır*: Cep (Pocket), Çanta (Bag), El (Hand), Göğüs (Chest), Bel (Waist), Asılı Çanta (Bag Hang On), Sabit (Dock), Diğer/Bilinmeyen (Other)*.

İlgili makalede [18] verilerin 30Hz frekansta Android uygulama kullanılarak toplandığı belirtilmektedir. Fakat verilerin zaman damgalarına bakıldığında çoğunluğunun 50Hz frekansta toplanmış olduğu hesaplanmıştır ve bu projede veri toplama frekansı 50Hz olarak kabul edilmiştir. Kullanılan veri setinde otobüs seyahat türüne ait 50Hz frekansta toplanan veriler mevcut olmadığından otobüs seyahat türü verileri kullanılmamıştır.

Veri setindeki ham veriler ivme ölçer (accelerometer), jiroskop (gyroscope), mıknatıs ölçer (magnetometer), basınç ölçer (barometer), yakınlık ölçer

(proximity), çizgisel ivme ölçer (linear acceleration), dönüş sensörü (rotation), yerçekimi sensörü (gravity), ışık sensörü (light) ve konum sensörleri (location) kullanılarak toplanmıştır. Bunlardan en çok veri toplanan sensörler ivme ölçer, jiroskop ve mıknatıs ölçerdir. İvme ölçer, jiroskop, mıknatıs ölçer, çizgisel ivme ölçer, dönüş sensörü, yerçekimi sensörü verileri 3 eksende toplanmıştır. Basınç ölçer, yakınlık ölçer, ışık sensörü verileri 1 eksende toplanmıştır. Konum sensörü verisinde ise enlem (latitude), boylam (longitude) ve yükseklik (altitude) toplanmıştır. Sensör verilerinin yanında verinin toplandığı zaman milisaniye ve nanosaniye cinsinden kaydedilmiştir. Veri dosyasının içeriği Şekil 3.1'de gösterildiği gibidir:

```
1363166805883 62675126277993 G: -0.042827092 0.07019427 0.05911472
1363166805883 62675126277993 A: 9.940529 -2.8918364 -4.7749205
1363166805889 62675126277993 M: -9.899994 33.899994 4.199997
1363166805893 62675160762000 PR: -2.4414063E-4
1363166805894 62675165607993 G: 0.20231822 0.4496372 0.171029
1363166805895 62675165607993 A: 13.640684 -2.5758436 -2.7399023
1363166805902 62675165607993 M: -9.899994 33.899994 4.199997
1363166805903 62675200688000 PR: -2.4414063E-4
1363166805904 62675214725993 G: 0.27373007 0.46988815 0.1347903
1363166805905 62675214725993 A: 6.1574454 -2.5945547 -2.961741
1363166805912 62675214725993 M: -10.199997 33.899994 3.2999878
1363166805913 62675242869000 PR: -2.4414063E-4
1363166805914 62675250243992 G: 0.024321053 0.46882236 0.08149779
1363166805916 62675250243992 A: 9.433383 -2.8846514 -3.9963894
1363166805921 62675250243992 M: -10.199997 33.899994 3.2999878
1363166806045 62675280758000 PR: -2.4414063E-4
1363166806047 62675296523992 G: 0.06269214 0.23113762 -0.22333589
1363166806047 62675296523992 A: 9.770931 -2.7274783 -3.7061431
1363166806051 62675296523992 M: -11.099991 34.199997 2.399994
1363166806052 62675320704000 PR: -2.4414063E-4
1363166806053 62675327675992 G: -0.0406955 0.45283458 -5.326322E-7
1363166806054 62675327675992 A: 9.315279 -3.9131625 -2.397265
1363166806055 62675327675992 M: -11.099991 34.199997 2.399994
```

Şekil 3.1 Veri dosyasının içeriği

3.1.2 Ek Veri Toplama

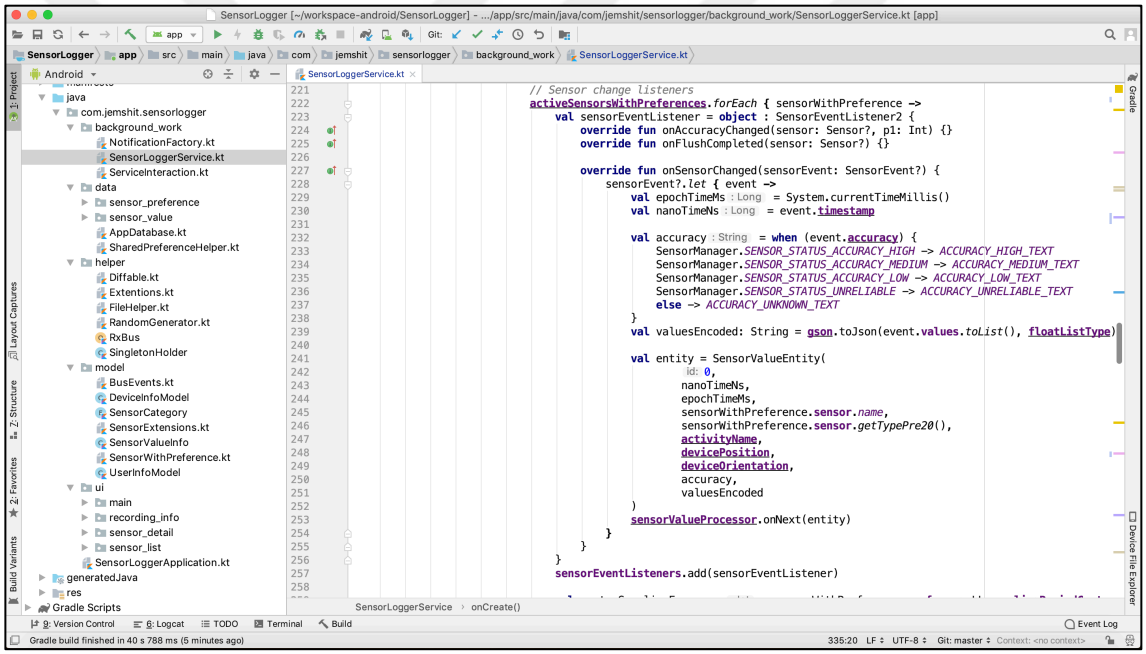
HTC veri setine ek olarak 700MB boyutunda 4 aktivite için veri toplanmıştır. Bunlar Metrobüs (Metrobus), Tramvay (Tram), Merdiven Çıkma (Stair Up) ve Merdiven İnmedir (Stair Down). Veriler ortalama 25 yaşında olan kişiler tarafından toplanmıştır ve %50 erkek, %50 kadından oluşmaktadır. Veriler HTC U11 ve Samsung S5 cihazlarında Android uygulama kullanılarak 50Hz frekansta

toplanmıştır. Veri toplamada kullanılan sensörler ivme ölçer, jiroskop ve mıknatıs ölçerdir. Veri dosyasının içerik düzeni Şekil 3.1'de gösterildiği gibidir.

3.1.2.1 Veri Toplama Android Uygulaması

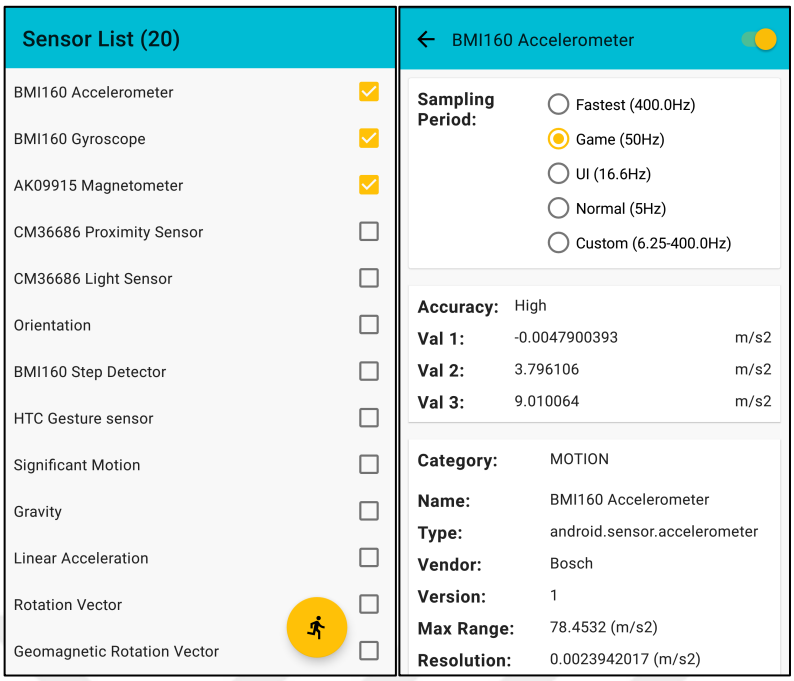
Ek veri toplama amaçlı Android cihazlar için uygulama yazılmıştır. Uygulamayı geliştirmek için kullanılan yazılım ve donanım gereksinimleri şunlardır:

- Android Studio IDE[1] (Şekil 3.2)
- Kotlin programlama dili[2]
- Android SDK 28
- Gradle Build Tool[3]
- Git Sürüm Kontrolü (Github[4], GitKraken[5] kullanıldı)

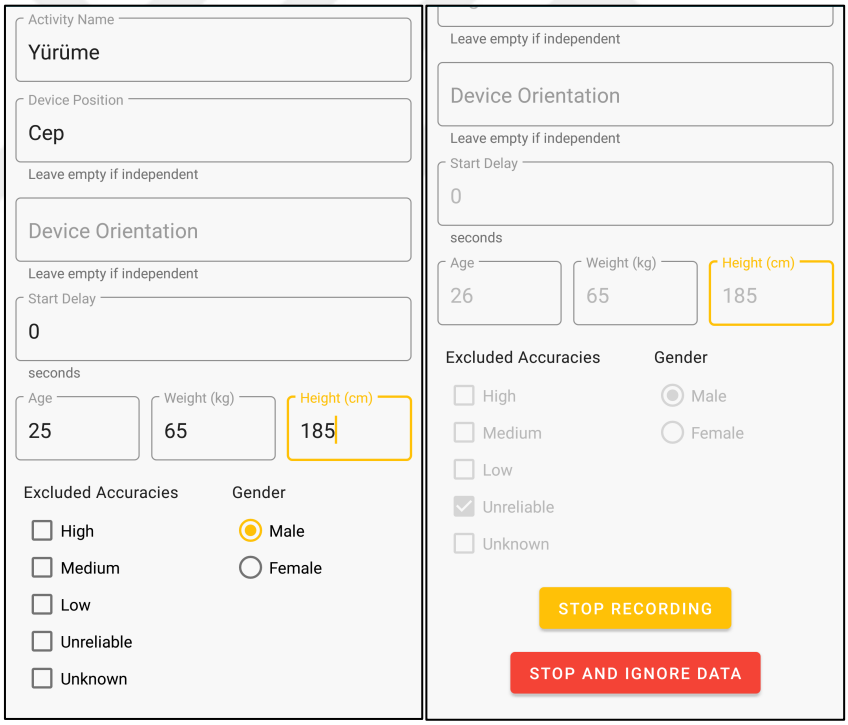


Şekil 3.2 Android Studio projesi

[1] https://developer.android.com/studio
[2] https://kotlinlang.org
[3] https://gradle.org
[4] https://www.github.com
[5] https://www.gitkraken.com



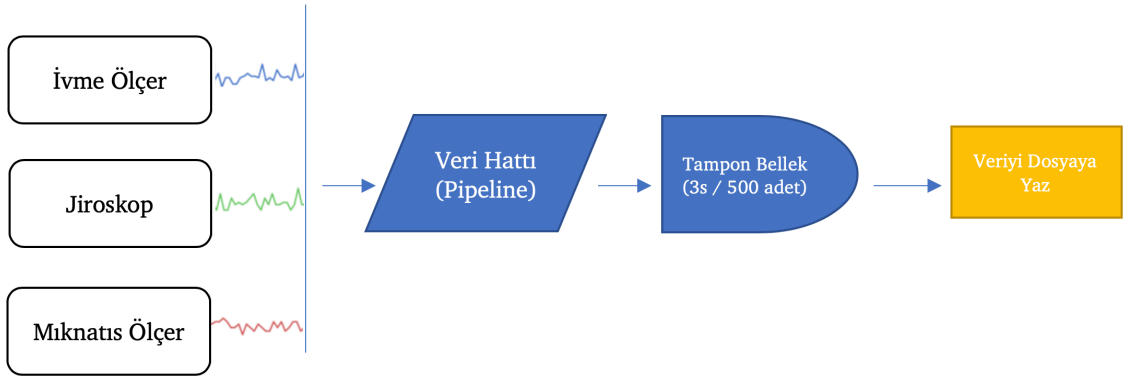
Şekil 3.3 Android veri toplama uygulaması, sensör listesi ve detayı



Şekil 3.4 Android veri toplama uygulaması, yapılan aktivitenin ve toplayan kişinin bilgilerini girme ekranı.

Veri toplama için geliştirilen Android uygulamasının ekranları Şekil 3.3 ve Şekil 3.4'te gösterilmiştir. Bu uygulamada önce ana ekranda (Şekil 3.3 solda) kullanılacak sensörler seçilir. Daha sonra sensör detay sayfasına girilerek (Şekil

3.3 sağ) verinin toplanma frekansı seçilir. Daha sonra Şekil 3.4'te gösterilen ekranda, yapılan aktivitenin ve veriyi toplayan kişinin bilgileri girilerek veri toplama işlemi başlatılır. Veriler uygulama arka plandayken veya ön plandayken toplanabilmektedir. Uygulama Şekil 3.5'te gösterilen mantıkta çalışmaktadır. 3 sensörden gelen veriler 3 saniyelik arayla veya 500 adet veri sınırına ulaştığında dosyaya yazılmaktadır. Bu şekilde veriyi dosyaya kaydetme işlemi 1 saniyede 150 (3 sensör 50Hz frekans) kere yapılmak yerine daha etkili olması açısından 3 saniyede 1 kere yapılmıştır.



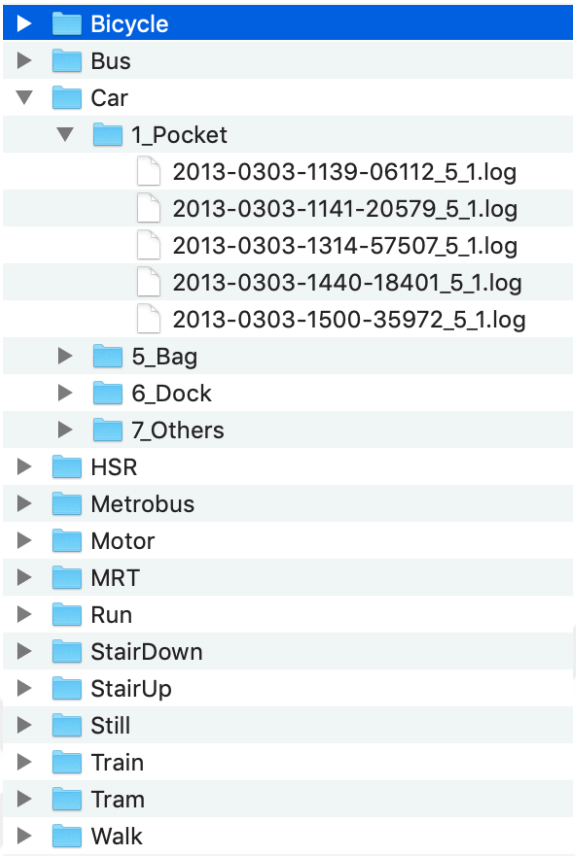
Şekil 3.5 Android veri toplama uygulamasının çalışma akışı

3.2 Veri Hazırlama, Ön İşleme ve Pencerelere Ayırma

Bu bölümde veri üzerinde yapılan işlemler, verilerin pencerelere nasıl ayrıldığı ve verinin eğitim, doğrulama, test setlerine nasıl ayrıldığı detaylı anlatılmıştır.

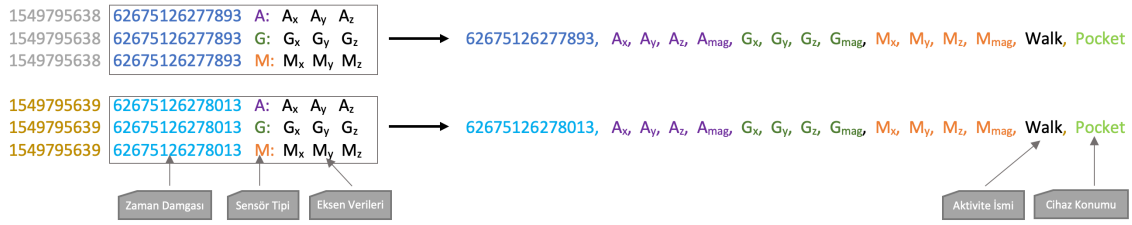
3.2.1 Veri Hazırlama

Ham veriler Şekil 3.6'da gösterilen klasör yapısında saklanmaktadır. En üst klasör ismi aktivite adıdır, alt klasör isimleri ise cihaz konumlarıdır.



Şekil 3.6 Veri seti klasör yapısı

'.log' uzantılı veri dosyasının her satırında farklı sensörlere ait veriler bulunmaktadır (Şekil 3.1). Bu sensörlerden bazıları aynı frekansta ve bazıları ise farklı frekansta toplanmıştır. Aynı frekansta toplanan sensörlerin verileri aynı zaman damgasına sahiptir. Şekil 3.7'de gösterildiği gibi aynı zaman damgasında toplanan ivme ölçer, jiroskop ve mıknatıs ölçer verileri tek satırda birleştirilerek yeni dosyaya kaydedilmiştir.



Şekil 3.7 Veri dosya düzeni

Her sensörün eksen verilerine ek olarak eksenlerin karelerinin toplamlarının karekökü de veri olarak eklenmiştir (**3.1**). Eksen büyüklüğü kullanılarak cihaz

konumundan ve oryantasyonundan bağımsız aktivite ve seyahat türü tanıma yapmak hedeflenmektedir.

$$A_{magnitude} = \sqrt[2]{x^2 + y^2 + z^2} \tag{3.1}$$

Ham veri dosyaları okunup yeni düzene çevrilirken veri toplama frekansına göre yeni dosyaya kaydedilmiştir. Bu sayede 20Hz frekansta toplanan yürüme aktivite verisiyle 50Hz frekansta toplanan yürüme aktivite verisi birbirinden ayırt edilmiştir. Ham verinin okunup yeni düzene çevrilmesi ve veri toplama frekansına göre yeni dosyaya kaydedilmesi Şekil 3.8'te gösterildiği gibi yapılmaktadır.



Şekil 3.8 Veri hazırlama işleminin çalışma akışı

Veriler yeni dosyaya kaydedilirken, her yolculuğa ait verileri birbirinden ayırt edebilmek için, her dosya işlendikten sonra araya özel satır eklenmiştir.

3.2.2 Veri Sayısı Dengeleme

Veri setinde yer alan her aktivite için toplam veri sayısı ve cihaz konumları Tablo 3.1'de verilmiştir. Dengesiz sayıda sınıf verilerinin getirebileceği problemlerden en önemlisi, modelin sınıflandırma yaparken veri sayısı çok olan sınıfa eğilimli olmasıdır [64] [65] [66]. Bu potansiyel problem riskini azaltmak için, sayısı çok olan aktivite veri sayıları azaltılarak ~6 milyon adete indirgenmiştir. Sayı adedini azaltmak için adedi en fazla olan cihaz konumlarından veriler silinmiştir.

Veriler 50Hz frekansta toplandığı için 6 milyon adet veri yaklaşık 33.3 saatlik yolculuğa karşılık gelmektedir. Veri boyutu en az olan aktivite ise 114 bin veri adediyle merdiven inmedir ve 38 dakikadır. Seyahat türlerinden en az veriye sahip olan tramvay aktivitesine ait ~4 saatlik yolculuk verisi mevcuttur.

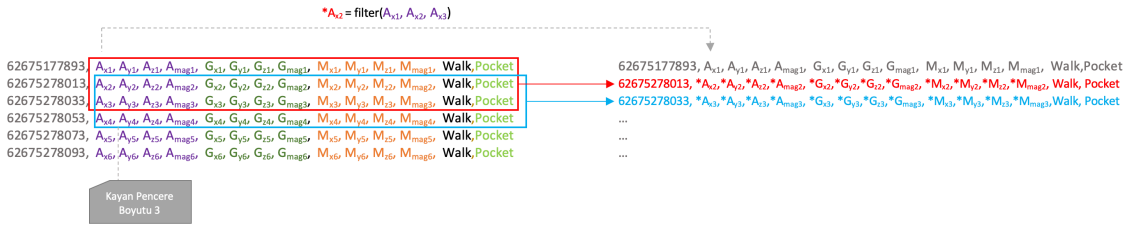
Tablo 3.1 Aktiviteler ve seyahat türleri, cihaz konumları ve örnek sayısı

Aktivite İsmi	Cihaz Konumları	Örnek Sayısı
Sabit Durma	El, cep, göğüs, bel, çanta, sabit	10.3 milyon
Yürüme	El, cep, göğüs, bel, çanta	10.6 milyon
Koşma	El, cep, göğüs, kol, bel, çanta, diğer/bilinmeyen	4.5 milyon
Merdiven İnme	El, cep	114 bin
Merdiven Çıkma	El, cep	124 bin
Bisiklet Sürme	Cep, göğüs, bel, çanta, asılı çanta, sabit	8.2 milyon
Motosiklet Sürme	Cep, göğüs, bel, çanta, sabit	7.5 milyon
Araba	Cep, çanta, sabit, diğer/bilinmeyen	1.3 milyon
Metro	El, cep, göğüs, bel, çanta	2.1 milyon
Tren	El, cep, göğüs, kol, bel, çanta, sabit	7.2 milyon
Hızlı Tren	El, cep, göğüs, kol, bel, çanta, sabit, diğer/bilinmeyen	8.6 milyon
Metrobüs	El, cep, çanta	2 milyon
Tramvay	El, cep, çanta	714 bin

3.2.3 Veriyi Ön İşleme

Ham verinin gürültüsünü azaltmanın sonuca etkilerini araştırmak için, düzenlenmiş ve veri adedi dengelenmiş veri seti üzerinde Aritmetik Ortalama

[100] Alçak-Geçirgen Süzgeç (Low-Pass Filter[1]) filtresi uygulanmıştır. Filtreli ve filtresiz veriler üzerinde ayrı deneyler yapılmıştır. Aritmetik Ortalama fitresi kayan pencereler ile eksen bazında işlemler yapılarak uygulanmıştır. Her pencerede 11 veri örneği, 12 eksen mevcuttur. Her eksen için filtre işlemi yapılır ve elde edilen sonuç kayan pencerenin ortasındaki veri örneği yerine yeni dosyaya kaydedilir. Filtrelerin nasıl uygulandığı Şekil 3.9'da gösterilmektedir.

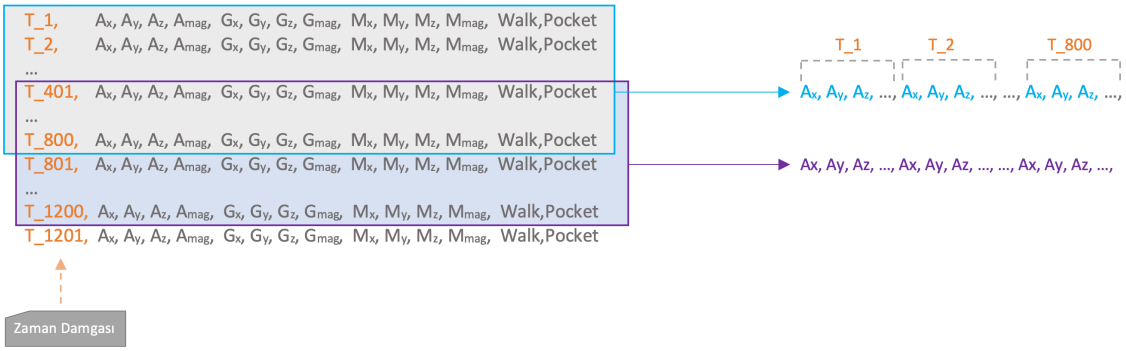


Şekil 3.9 Veriye eksen bazında filtre uygulanması

3.2.4 Veriyi Pencerelere Ayırma

Veri seti, veriyi hazırlama aşamasında, her satırda 3 sensöre ait düzenlenmiş veri olacak şekilde tek dosyaya kaydedilmiştir (Şekil 3.7). Bunları sınıflandırmaya uygun hale getirmek için kayan pencere tekniği (sliding window technique) [69] [101] kullanılmıştır. Verilerden 4, 8, 16 ve 32 saniyelik, %50 örtüşmeli pencereler oluşturulmuştur. Pencere oluşturmak için her saniyede toplanan verileri art arda birleştirmek gerekmektedir. Her pencereye ait veri tek satırda olacak şekilde yeni dosyaya kaydedilmiştir. Veriyi pencereye ayırma işlemi Şekil 3.10'da gösterildiği gibi yapılmıştır.

[1] https://en.wikipedia.org/wiki/Low-pass_filter



Şekil 3.10 Pencere oluşturma (16 saniyelik)

Farklı yolculukları ayırt etmek için hazır veri dosyasına yolculuklar arası özel satır eklenmiştir. Pencereler oluşturulurken her pencerede aynı yolculuğa ait verilerin sıralı şekilde koyulmasına dikkat edilmiştir. İki farklı yolculuğa ait veriler aynı pencere içerisine yerleştirilmemiştir. Pencereler oluşturulurken her pencerenin sınıf etiketi OHE (One-Hot Encoding) formatında farklı dosyaya kaydedilmiştir.

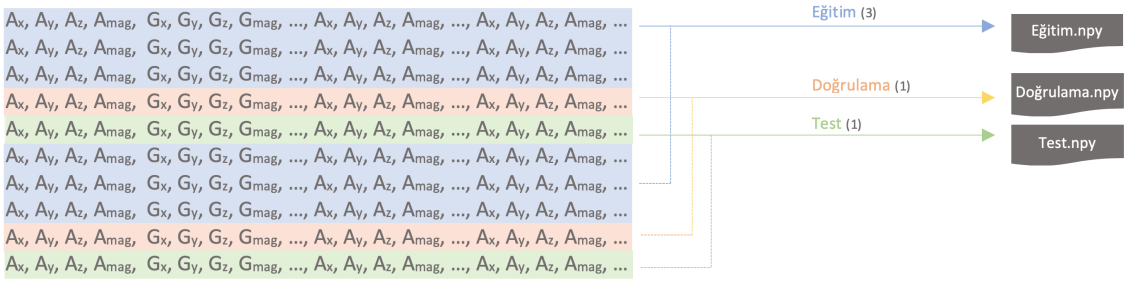
3.2.5 Eğitim, Doğrulama ve Test Veri Seti Oluşturma

Veri seti %60, %20, %20 oranında eğitim, doğrulama ve test setlerine ayrılmıştır. Eğitim setiyle sınıflandırma modelleri eğitilmiştir. Bu modeller için uygun parametreleri seçmek için ve modellerin veriyi ezberlemediğini teyit etmek için doğrulama seti kullanılmıştır. Eğitim yapılırken sınıflandırma modelinin doğrulama seti üzerindeki performansı modelin nasıl genelleme yaptığını da göstermektedir. Parametreler seçildikten sonra ve model eğitildikten sonra test seti üzerinde sınıflandırma yapılmıştır ve nihai sonuç olarak bunlar kabul edilmiştir. Bu oranda veri setini bölmek için 3 farklı yol kullanılmıştır ve daha sonra sonuçlara etkisi ortaya konmuştur.

3.2.5.1 Veriyi 3-1-1 Şeklinde Setlere Ayırmak

Pencereler oluşturulduktan sonra (Şekil 3.10) her satırda bir pencere olacak şekilde dosyaya kaydedilmiştir. Bu pencereler dosyaya kaydedilirken, ilk üç pencere eğitim setine, sonraki bir pencere doğrulama setine, sonra ki bir pencere de test setine kaydedilmiştir. Sonra tekrar üç pencere eğitim setine olacak şekilde aynı döngüyle pencereler setlere ayrılmıştır (Şekil 3.11). Bu yöntem sayesinde

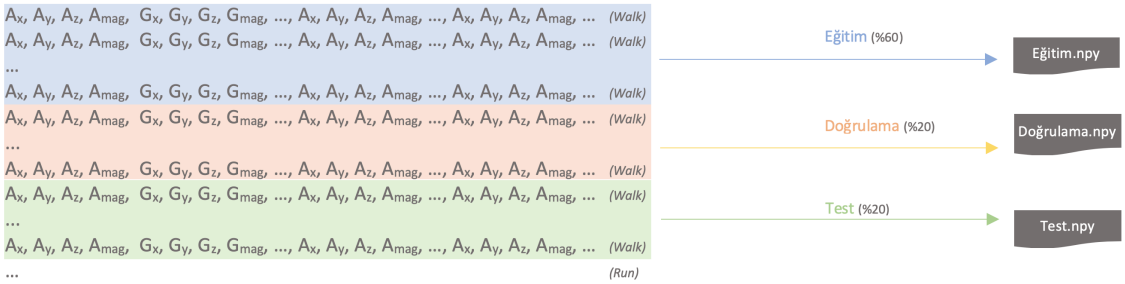
veriler tam olarak %60, %20, %20 oranında ayrılmış olur ve her sette her aktivite için her cihaz konumuna ait veriler mevcut olmaktadır. Veri bu yöntemle ayrıldığında aynı yolculuğun parçaları her sette bulunmaktadır.



Şekil 3.11 Pencereleri 3-1-1 yoluyla setlere ayırmak

3.2.5.2 Veriyi %60-%20-%20 Oranında Setlere Ayırmak

Bu yöntemle aktivitenin ilk %60'lık pencereleri eğitim setine, sonraki %20'lik pencereleri doğrulama setine, en son %20'lik pencereleri ise test setine kaydedilmiştir. Her aktivite bu şekilde döngüyle setlere ayrıldı. Bu yöntem sayesinde veriler tam olarak %60, %20, %20 oranında ayrılmış olur (Şekil 3.12). Ama her sette aynı aktivitenin bütün cihaz konumları için veriler mevcut olmamaktadır. Aktivite pencerelerinin ilk %60'lık bölümündeki cihaz konumları eğitim setine, sonraki %20'lik pencerelerdeki cihaz konumları sadece doğrulama setine ve en son %20'lik pencerelerdeki cihaz konumları ise sadece test setine eklenmektedir. Veri bu yöntemle ayrıldığında her sette farklı yolculuklar bulunmaktadır.



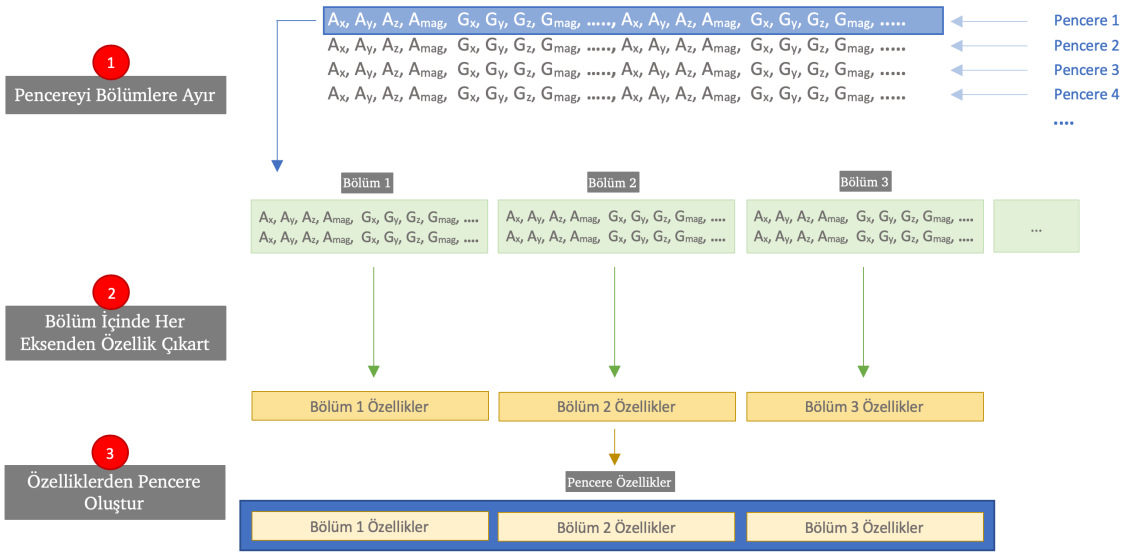
Şekil 3.12 Pencereleri %60-%20-%20 yoluyla setlere ayırmak

3.2.5.3 Veriyi Dosya Boyutuna Göre Setlere Ayırmak

Bu yöntemde ham veriler en başta dosya boyutlarına göre %60, %20, %20 oranda olacak şekilde 3 sete ayrılmıştır. Her sette her aktiviteye ait bütün cihaz konumları mevcuttur. Veri bu yöntemle ayrıldığında her sette farklı yolculuklar bulunmaktadır.

3.3 Zaman Alanında Özellik Çıkartma

Veriden pencereler oluşturulduktan sonra zaman alanında [38] makalesinde kullanılan özellikler çıkartılmıştır. Pencerelerdeki örnekler ivme ölçer, jiroskop, mıknatıs ölçer sensörlerine ait 3'er eksen verisi ve her sensörün büyüklüğü olmak üzere toplam 12 eksenden oluşmaktadır. Bu 12 eksenin her biri için 27 adet özellik çıkartılmıştır. LSTM sınıflandırma ağı için, ağın doğası gereği girdiler bölümlere (segment/chunk) ayrılmıştır. Dolayısıyla her pencereden özellik çıkartmak yerine pencere içindeki her bölümden özellikler çıkartılmıştır. LSTM ağının girdileri pencerelerdir ve her pencere Şekil 3.13 üzerinde gösterildiği gibi belirli sayıda bölümlere ayrılmıştır. Daha sonra her bölümde eksen bazlı toplam 324 özellik çıkartılmıştır. Bölümlerden çıkartılan bu özelliklerden tekrar pencereler oluşturulmuştur.



Şekil 3.13 Zaman alanında özellik çıkartma

Çıkartılan özellikler Tablo 3.2'de gösterilmiştir. Özellikler eksen bazında çıkartıldığı için, N belirli eksendeki değerlerin sayısını ve s_i o değerlerin i sırasındaki değerini ifade etmektedir.

Tablo 3.2 Zaman alanında çıkartılan özellikler

Özellik İsmi	**Denklem**
Minimum Reduction	$\forall_i \in \{1, \ldots, N-1\},\ min(s_{i+1}-s_i),\ if\ (s_{i+1}-s_i) < 0$
Maximum Reduction	$\forall_i \in \{1, \ldots, N-1\},\ max(s_{i+1}-s_i),\ if\ (s_{i+1}-s_i) < 0$
Minimum Increase	$\forall_i \in \{1, \ldots, N-1\},\ min(s_{i+1}-s_i),\ if\ (s_{i+1}-s_i) > 0$
Maximum Increase	$\forall_i \in \{1, \ldots, N-1\},\ max(s_{i+1}-s_i),\ if\ (s_{i+1}-s_i) > 0$
Minimum Value	$\forall_i \in \{1, \ldots, N\},\ min(s_i)$
Maximum Value	$\forall_i \in \{1, \ldots, N\},\ max(s_i)$
Range	*Maximum Value – Minimum Value*
Arithmetic Mean	$\left(\sum_{i=1}^{N} s_i\right)/N$
Quadratic Mean	$\sqrt{\frac{\sum_{i=1}^{N} s_i^2}{N}}$
Median	$Sort\ N \begin{cases} s_{\frac{N}{2}},\ if(N\%2\ !=0) \\ \frac{s_{\frac{N}{2}}+s_{\frac{N+1}{2}}}{2},\ if(N\%2==0) \end{cases}$
Variance	$\left(\sum_{i=1}^{N}(s_i - Arithmetic\ Mean)^2\right)/(N-1)$
Standard Deviation	$\sqrt{Variance}$
Arithmetic Mean of Instant Exchange	$\left(\sum_{i=1}^{N-1}(s_{i+1}-s_i)\right)/N$

Quadratic Mean of Instant Exchange	$\sqrt{\left(\sum_{i=1}^{N-1}(s_{i+1}-s_i)^2\right)/N}$
Covariance	Standard Deviation / Arithmetic Mean
Frequency Above Median	$\forall_i \in \{1, ..., N\},\ freq(s_i)\ \ if\ (s_i) > median + \varepsilon$
Frequency Below Median	$\forall_i \in \{1, ..., N\},\ freq(s_i)\ \ if\ (s_i) < median - \varepsilon$
Frequency Between Median	$\forall_i \in \{1, ..., N\},\ freq(s_i)\ \ median + \varepsilon > if\ (s_i) > median - \varepsilon$
Frequency Above Mean	$\forall_i \in \{1, ..., N\},\ freq(s_i)\ \ if\ (s_i) > mean + \varepsilon$
Frequency Below Mean	$\forall_i \in \{1, ..., N\},\ freq(s_i)\ \ if\ (s_i) < mean - \varepsilon$
Frequency Between Mean	$\forall_i \in \{1, ..., N\},\ freq(s_i)\ \ mean + \varepsilon > if\ (s_i) >$ mean $- \varepsilon$
Maximum Consecutive Above Median	$\forall_i \in \{1, ..., N\},\ max(m)\ \ if\ \forall_j \in \{1, ..., m\},\ s_{i+j} > median + \varepsilon$
Maximum Consecutive Below Median	$\forall_i \in \{1, ..., N\},\ max(m)\ \ if\ \forall_j \in \{1, ..., m\},\ s_{i+j} < median - \varepsilon$
Maximum Consecutive Between Median	$\forall_i \in \{1, ..., N\},\ max(m)\ \ if\ \forall_j \in \{1, ..., m\},\ median + \varepsilon > s_{i+j} > median - \varepsilon$
Maximum Consecutive Above Mean	$\forall_i \in \{1, ..., N\},\ max(m)\ \ if\ \forall_j \in \{1, ..., m\},\ s_{i+j} > mean + \varepsilon$
Maximum Consecutive Below Mean	$\forall_i \in \{1, ..., N\},\ max(m)\ \ if\ \forall_j \in \{1, ..., m\},\ s_{i+j} < mean - \varepsilon$
Maximum Consecutive Between Mean	$\forall_i \in \{1, ..., N\},\ max(m)\ \ if\ \forall_j \in \{1, ..., m\},\ mean + \varepsilon > s_{i+j} > mean - \varepsilon$

3.4 Kullanılan Yöntemler

3.4.1 Oto Kodlayıcı

İstatistiksel özellikler dışında Oto Kodlayıcı kullanılarak özellikler çıkartılmıştır. Oto Kodlayıcılar veri seti ve problem alanı hakkında bilgi sahibi olmaksızın özellik çıkartmak için kullanılan derin öğrenme yöntemidir. Projede, bir önceki bölümde anlatılan, birkaç gizli katmanlardan oluşan Derin Oto Kodlayıcı kullanılmıştır. Ağın veriyi ezberlemesini zorlaştırmak için giriş katmandan sonra Dropout katmanı eklenmiştir.

3.4.2 Tekrarlayan Sinir Ağı (RNN)

Günlük aktivite ve seyahat türü verileri zamana bağlı değişen verilerdir ve zamanla bir desen oluştururlar. LSTM bu problemi çözmek için uygun derin öğrenme ağı olarak görülmüştür. Projede veri pencereleri belirli zaman adımlarına bölünerek LSTM ağına girdi olarak verilmiştir. LSTM ağı bu zaman adımlarından bir bellek oluşturur. Bu şekilde sınıflandırma yapılırken daha önceki zaman adımlarının oluşturduğu hafızaya bakarak sınıflandırma yapılması amaçlanmıştır.

3.4.3 Kayıp Fonksiyonu

Oto Kodlayıcı ağının performansını değerlendirmek için kayıp (loss) metriği kullanılmıştır. Oto Kodlayıcı ağında sınıflandırma yapılmadığı için doğruluk (accuracy) metriğinin kullanılması uygun görülmemiştir. Kayıp fonksiyonu olarak MAE kullanılmıştır. MSE fonksiyonu MAE fonksiyonuna göre daha büyük değer ürettiğinden uç değerlere (outlier) daha duyarlıdır. Bu sebepten deneylerde MAE kayıp fonksiyonu tercih edilmiştir.

3.4.4 Aktivasyon Fonksiyonu

Oto Kodlayıcı ağında gizli katmanlarda Tanh aktivasyon fonksiyonu tercih edilmiştir. Çıkış katmanında ise, veri 0-1 arasına normalleştirildiğinde çıktı 0-1 arası değer alabilmesi için Sigmoid fonksiyonu kullanılmıştır. ReLU aktivasyon fonksiyonu sıfır altındaki değerleri sıfıra eşitlemektedir. Bu yüzden Oto

Kodlayıcının şifre çözücü (decoder) katmanlarında veri kaybı oluşmaması için ReLU kullanılmamıştır.

LSTM ağında aktivasyon fonksiyonu olarak varsayılan Tanh fonksiyonu kullanılmıştır.

3.4.5 Eniyileştirici (Optimizer)

Oto Kodlayıcı ağında ve LSTM ağında eniyileştirici olarak Adam kullanılmıştır. Uyarlanabilir öğrenme hızı kullanan Adadelta ve Adam algoritmaları LSTM sınıflandırma ağında karşılaştırılmıştır ve Adam daha iyi sonuç ürettiği için tercih edilmiştir.

3.5 Yazılım Geliştirme Ortamı ve Kullanılan Araçlar

Makine öğrenmesi alanında kütüphanelerin ve araçların fazla olması nedeniyle bütün yazılımlar Python[1] programla dili kullanılarak yazılmıştır. Geliştirme ortamı olarak PyCharm IDE[2] tercih edilmiştir. Numpy[3] Python kütüphanesi verileri dizi olarak ifade etmek, yeniden boyutlandırmak ve diziyi dosyaya kaydetmek için kullanılmıştır. Pandas[4] Python kütüphanesi .csv uzantılı dosyaları okumak için kullanılmıştır. Sonuçları grafik halinde çizdirmek için Matplotlib ve Seaborn[5] kütüphaneleri kullanılmıştır. Projede kullanılan bütün yazılımlar ve sürümleri Tablo 3.3'te gösterilmiştir.

Klasik Makine Öğrenmesi yaklaşımlarıyla eğitim yapmak için WEKA[6] [102] kullanılmıştır. WEKA aracında veri ön işleme, sınıflandırma ve kümeleme yapmak için birçok yöntem mevcuttur ve verileri özel düzende verilerek bu işlemler yapılabilmektedir.

Derin öğrenme, Oto Kodlayıcı (Autoencoder) modelleri oluşturmak için Keras[7] [98] kullanılmıştır. Keras aracı Tensorflow, Theano ve CNTK araçlarını arka

[1] https://www.python.org
[2] https://www.jetbrains.com/pycharm
[3] http://www.numpy.org
[4] https://pandas.pydata.org
[5] https://seaborn.pydata.org/#
[6] https://www.cs.waikato.ac.nz/ml/weka
[7] https://keras.io

planda kullanarak çalışmakta ve derin öğrenmeyi hızlı uygulayabilmek için kolay arayüz sağlamaktadır. Keras, en çok kullanılan aktivasyon fonksiyonlarını (Sigmoid, Tanh, Softmax), eniyileştirici algoritmalarını (Adam, SGD, Adadelta), kayıp fonksiyonlarını (MSE, MAE), değerlendirme metriklerini (Doğruluk, Kayıp) ve mimarileri (CNN, RNN) hazır sunmaktadır. Bunlara ek olarak Dropout ekleme, düzenleme (Regularization), gürültü ekleme (Noise), eğitimi erken durdurmak (Early Stop) gibi ek araçların kolayca kullanılmasına imkân sağlamaktadır.

Her yazılım için ayrı Python Sanal Ortam (Virtual Environment) oluşturulmuştur ve bu şekilde her yazılımın Python sürümü, kullanılan kütüphaneleri ve sürümleri birbirinden bağımsız hale getirilmiştir. Kodların sürüm kontrolü için Git, GitKraken ve Github kullanılmıştır.

Tablo 3.3 Projenin yazılım gereksinimleri

Yazılım/Gereksinim İsmi	**Kullanılan Sürüm**
PyCharm IDE	2018.3
Python	2.7 ve 3.6
Virtualenv	16.2.0
Keras	2.2.4
Tensorflow	1.12.0
Numpy	1.16.0
Pandas	0.24.2
Matplotlib	3.0.3
Seaborn	0.9.0
Scikit-learn	0.19.2
WEKA	3.8

Donanım olarak belirli veri seti işlemleri için 8/16GB RAM, i7 İntel işlemcili dizüstü bilgisayar kullanılmıştır. Eğitim, özellik çıkartma ve belirli veri seti işlemleri için GTX 1080 ekran kartlı, Intel i5 işlemcili 16GB/23GB RAM bulunduran (Ubuntu işletim sistemi kurulu) Linux makinesi kullanılmıştır. Bu makineyi soğutmak için 2 tane fan takılmıştır ve güç kaynağı olarak 650W Cooler Master kullanılmıştır. Uzaktan makineye bağlantı kurmak için SSH ve dosya aktarımı için SFTP (FileZilla[1]) kullanılmıştır. Projenin donanım gereksinimleri ve kapasiteleri Tablo 3.4'te verilmiştir.

Tablo 3.4 Projenin donanım gereksinimleri

Donanım/Gereksinim İsmi	**Donanım Kapasitesi**
Bellek (RAM)	16GB-23GB
Ekran Kartı (GPU)	GTX 1080, 8-12GB
İşlemci (CPU)	i5-i7
Güç Kaynağı (Power Supply)	650W+
Fan	2+

[1] https://filezilla-project.org

4
DENEYLER

Bu bölümde LSTM derin öğrenme ağı kullanılarak yapılan sınıflandırma deneyleri ve sonuçları anlatılmıştır.

4.1 Sınıflandırma ve Karşılaştırmalı Sonuçlar

LSTM ağının parametrelerini seçebilmek için, zaman alanında çıkartılan özellikler kullanılarak deneyler yapılmıştır. Bu deneylerin sonucunda en uygun LSTM ağı parametreleri seçilmiştir. Daha sonra ham veri özellikler, zaman alanında çıkartılan özellikler ve Oto Kodlayıcı ağıyla çıkartılan özellikler kullanılarak sınıflandırma yapmanın sonuçları ortaya konmuştur. Bu sonuçlar daha sonra makine öğrenmesi yöntemleriyle elde edilen sonuçlar ile karşılaştırılmıştır.

4.1.1 Değerlendirme Metrikleri

Sınıflandırma başarısını ölçmek için farklı metrikler kullanılabilir. Aktivite tanıma çalışmaları en çok doğruluk (accuracy), kesinlik (precision) ve anımsama (recall) metriklerini kullanmışlardır. Seyahat türü tanıma makalelerinde ise kesinlik, anımsama ve karışıklık matrisi (confusion matrix) daha çok kullanılmıştır.

Doğruluk metriği ile modelin, girdilerin yüzde kaçını doğru sınıflandırdığı ölçülebilir. Doğruluk (**4.1**)'de gösterildiği gibi hesaplanmaktadır.

$$\frac{Doğru\ Pozitif + Doğru\ Negatif}{Doğru\ Pozitif + Doğru\ Negatif + Yanlış\ Pozitif + Yanlış\ Negatif} \quad \textbf{(4.1)}$$

Doğruluk metriği dengesiz veri seti olan problemlerde yetersiz kalabilmektedir. Özellikle yanlış sonuçların getirdiği maliyet büyük ise kesinlik ve anımsama metrikleri kullanılabilir. Örneğin, verilerin 98 tanesi A sınıfına, 2 tanesi B sınıfa ait ise, model bütün verileri A sınıfı olarak etiketleyebilir ve %98 doğruluk elde edilebilir. Ama sağlık, bankacılık gibi alanlarda 2 örneği yanlış etiketlemenin

maliyeti yüksek olabilmektedir. Bunlara benzer, Yanlış Negatif sınıflandırmanın maliyeti yüksek olan problemlerde anımsama metriği kullanılmalıdır (**4.2**).

$$\frac{Doğru\ Pozitif}{Doğru\ Pozitif + Yanlış\ Negatif} \quad \textbf{(4.2)}$$

Kesinlik metriği Yanlış Pozitif sınıflandırmanın maliyetinin yüksek olduğu problemlerde kullanılmalıdır. Önemli epostayı istenmeyen eposta olarak etiketlemek buna bir örnektir. Kesinlik metriği (**4.3**)'te gösterildiği gibi hesaplanmaktadır.

$$\frac{Doğru\ Pozitif}{Doğru\ Pozitif + Yanlış\ Pozitif} \quad \textbf{(4.3)}$$

F1 metriği ile kesinlik ve anımsama arasında denge kurulabilmektedir (**4.4**).

$$2 \times \frac{Kesinlik \times Anımsama}{Kesinlik + Anımsama} \quad \textbf{(4.4)}$$

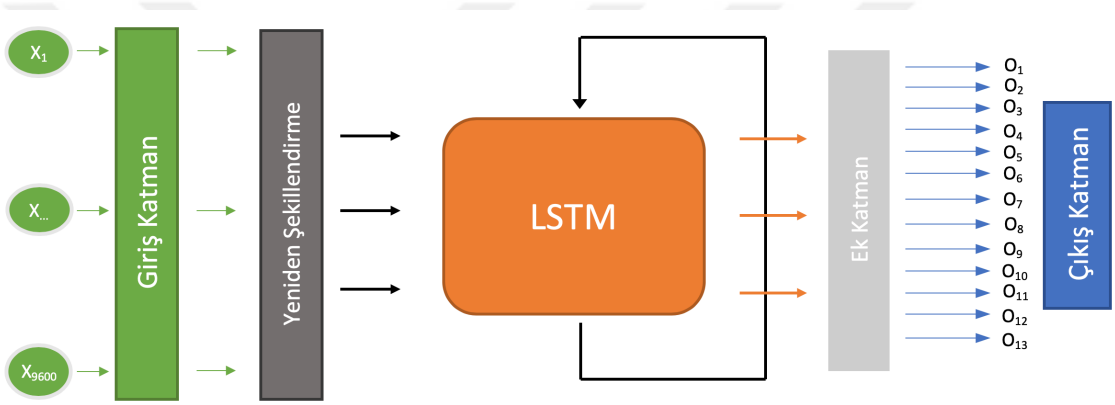
Sınıflandırma yapıldıktan sonra karışıklık matrisi ile hangi sınıfın hangi sınıf ile karıştırıldığı tespit edilebilmektedir. Özellikle seyahat türü tanımada hangi araçların birbiriyle karıştırılıp yanlış tahmine sebep olduğu anlaşılabilir. Karışıklık matrisi Tablo 4.1'de görüldüğü gibi ifade edilmektedir. Tablonun bir tarafında tahmin edilen etiketler, diğer tarafında ise beklenen sınıf etiketleri yer almaktadır. Beklenen sınıf etiketi ile tahmin edilen sınıf etiket sayısı eşit/yakın ise (Doğru Pozitif) yüksek başarı elde edilmiş demektir.

Tablo 4.1 Karışıklık matrisi

		Tahmin	
		Pozitif	**Negatif**
Gerçek	**Pozitif**	*Doğru Pozitif*	*Yanlış Negatif*
	Negatif	*Yanlış Pozitif*	*Doğru Negatif*

4.1.2 LSTM ve Zaman Alanı Özellikleri

LSTM ağı bir derin öğrenme yöntemi olup 3 boyutlu veriyi girdi olarak almaktadır. Veriyi ağa girdi olarak verebilmek için önce şu şekilde boyutlandırılması gerekmektedir: *örnek sayısı, zaman adımı sayısı, özellik sayısı*. Örnek sayısı LSTM ağına toplam kaç örneğin girdi olarak verileceğini ifade etmektedir. Bu deneyde her pencere (ör. 16 saniyelik) bir örnek anlamına gelmektedir. Zaman adımı sayısı her örneğin kaç zaman adımından oluştuğunu ifade etmektedir. Eğer pencere (örnek) 16 saniye uzunluğunda ise, bu pencere 1 saniyelik bölümlere ayrıldığında 16 zaman adımı etmektedir. Girdinin üçüncü boyutu olan özellik sayısı her bölümde kaç özellik bulunduğunu ifade etmektedir.



Şekil 4.1 LSTM sınıflandırma ağ mimarisi

LSTM sınıflandırma ağının mimarisi Şekil 4.1 üzerinde gösterilmiştir. Veriler 50Hz frekansta toplandığı için her saniyede 3 sensörden 50'şer örnek toplanmıştır. Her sensörde 3 eksen (x,y,z) vardır ve bunlara ek olarak eksenlerin büyüklüğü (karelerinin toplamlarının karekökü) bunlara ek olarak kullanılmıştır. Dolayısıyla her sensörde 4 değer bulunmaktadır. Veri 50Hz frekansta toplandığında her saniyede 3 sensörden toplamda 600 tane değer toplanmaktadır. 16 saniyelik pencere içinde ise 9600 tane değer bulunmaktadır. Bu pencereler LSTM ağına Şekil 4.1'de gösterildiği gibi girdi olarak verilmektedir. Daha sonra girdi LSTM hücresine uygun olacak şekilde boyutlandırılmak için Şekillendirme (Reshape) katmanından geçirilmiştir. Bu katmanda 9600 uzunluğundaki girdi bölümlere ayrılır ve zaman adımı, özellik sayısı olmak üzere iki boyutlu hale getirilir. Daha

sonra bu veri LSTM katmanından geçer. LSTM katmanının çıktısı ek katmana veya direk çıkış katmanına verilebilir. Çıkış katmanında, 13 adet sınıf olduğu için, 13 nöron bulunmaktadır ve aktivasyon fonksiyonu olarak Softmax kullanılmıştır.

Eğitimler bir önceki bölümde anlatıldığı gibi eğitim setinde yapılmıştır ve her epoch'ta sonuçlar doğrulama seti ile test edilmiştir. Parametreler doğrulama setinde çıkan sonuçlara göre seçilmiştir. Doğrulama setinde çok yakın sonuçlar çıktığında ise parametreler modelin test veri seti üzerindeki performansına göre seçilmiştir.

4.1.1.1 LSTM Parametrelerinin Sonuca Etkileri

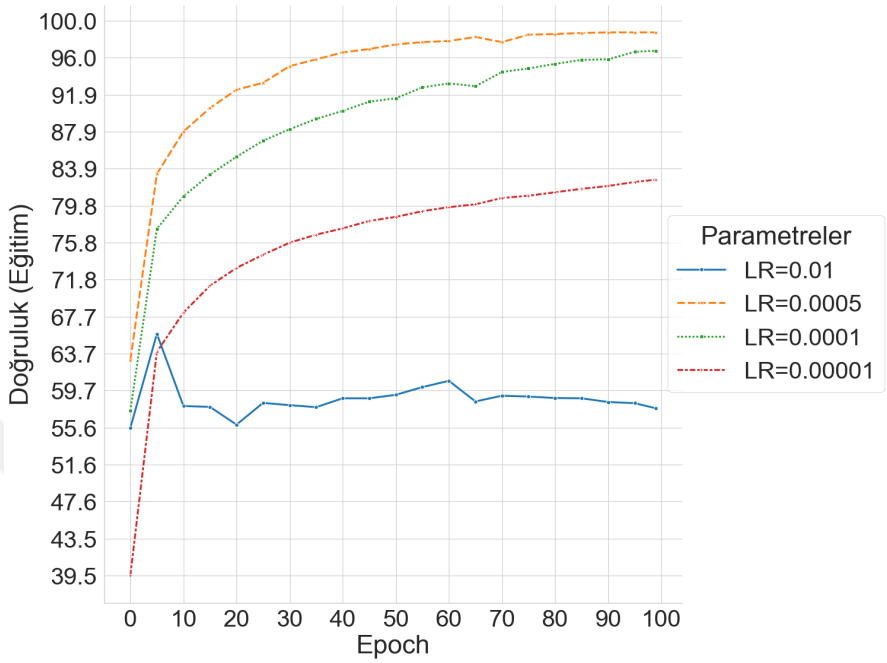
LSTM ağını oluşturmak için Keras aracı kullanılmıştır. Keras ile LSTM ağı parametreleri kolay değiştirilebilir ve sonuçlar karşılaştırılabilir. LSTM parametrelerini şu şekilde sıralayabiliriz: *öğrenme hızı (learning rate), dropout, LSTM birim sayısı (unit size), batch size, LSTM hücre sayısı (LSTM cells), eniyileştirici (optimizer), epoch, düzenleme (regularization), çift yönlü LSTM (Bidirectional LSTM)*.

4.1.1.1.1 Öğrenme Hızı (Learning Rate)

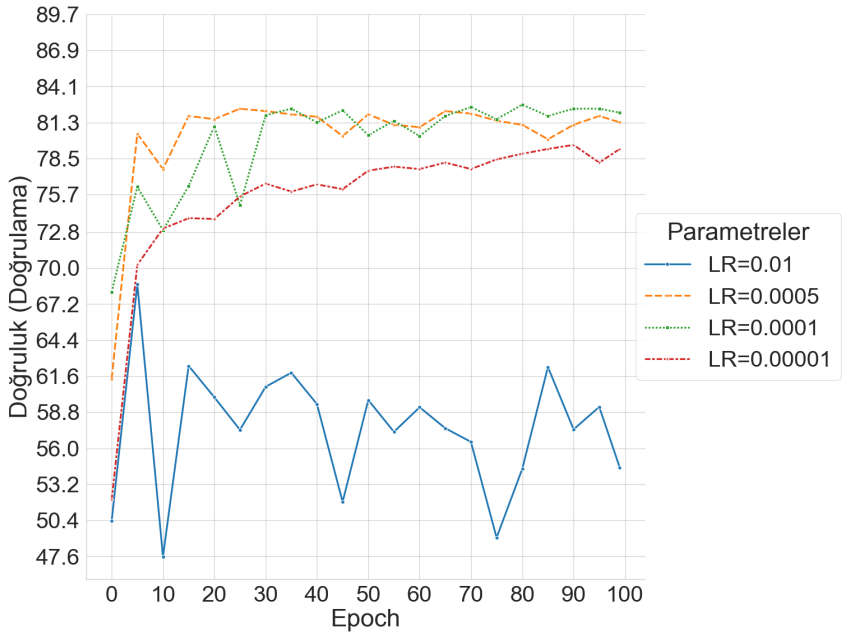
Öğrenme hızı ağın ağırlıklarını hangi hızda, ölçekte güncelleyeceğini belirtmektedir. Öğrenme hızı aynı zamanda eğitim süresini de etkilemektedir. Büyük öğrenme hızı eğitimin hızlı ve kısa sürmesine, küçük öğrenme hızı eğitimin uzun sürmesine neden olmaktadır. Büyük öğrenme hızı ile eğitim yapılırsa ağın küresel minimum hatayı (uygun ağırlıkları) kaçırma ihtimali vardır. Büyük öğrenme hızı aynı zamanda doğruluk grafiğinde yüksek dalgalanmaya neden olmaktadır. Diğer yandan küçük öğrenme hızı ile daha yavaş eğitim yapılabilir ve küresel minimum hata aranabilir. Küçük öğrenme hızı kullanmanın yerel minimum hataya takılıp kalma riski bulunmaktadır. Bu yüzden bazı eniyileştirici algoritmaları ivme (momentum) parametresi ile bu problemi çözmektedir.

Uyarlanabilir (dinamik) öğrenme hızı kullanan Adam algoritması ile deneyler yapılmıştır. Adam eniyileştirici algoritması, özelliklere göre farklı öğrenme hızı kullanır ve bu şekilde her özelliğe ait parametreleri farklı ölçekte

güncelleyebilmektedir. Buna ek olarak Adam algoritması uyarlanabilir ivme kullanmaktadır. Eğitim başlamadan Adam algoritmasına başlangıç öğrenme hızı parametre olarak belirtilir. Çeşitli başlangıç öğrenme hızları kullanılarak yapılan deneylere ait sonuçlar Şekil 4.2 ve Şekil 4.3'te gösterilmiştir.

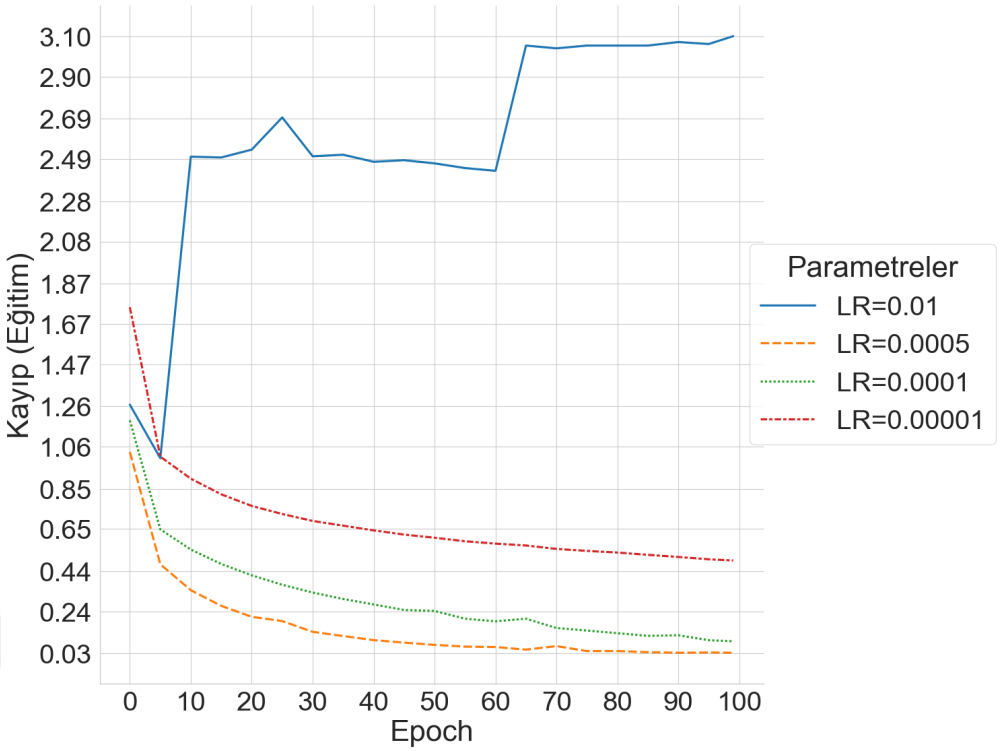


(a) Eğitim seti



(b) Doğrulama seti

Şekil 4.2 Eğitim hızının doğruluğa etkisi



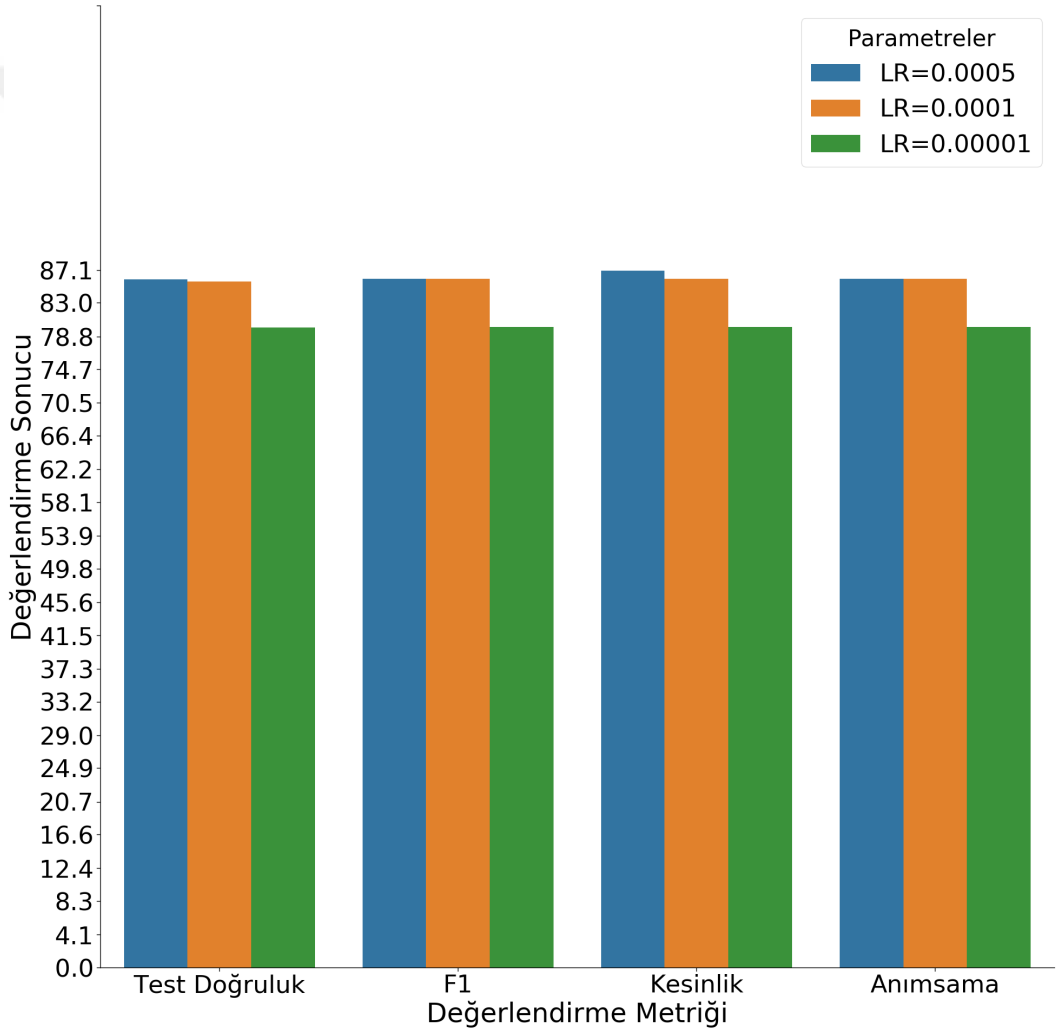
(a) Eğitim seti



(b) Doğrulama seti

Şekil 4.3 Eğitim hızının kayıp değerine etkisi

Eğitim hızı olarak 0.01 gibi çok büyük değer seçmek düşük doğruluk sonucu üretmektedir ve doğruluk grafiğinde çok dalgalanmaya sebep olmaktadır. Aynı zamanda kayıp değerleri çok yüksek çıkmıştır. Bu eğitim hızı ile küresel minimum hata bulunamamıştır. 0.00001 gibi çok küçük değer seçildiğinde, diğer daha büyük değerlere göre doğruluğun biraz daha geç (80. epoch gibi) doyuma ulaştığı görülebilir. 0.0005 ve 0.0001 eğitim hızı değerleri bu sonuçlar arasında en iyi doğruluğa ulaşan sonuçlardır. Bu iki eğitim hızının doğrulama setindeki doğruluk değerleri çok yakın olduğundan test seti kullanılarak sonuçlar hesaplanmıştır ve F1, kesinlik, anımsama değerleri karşılaştırılmıştır (Şekil 4.4).

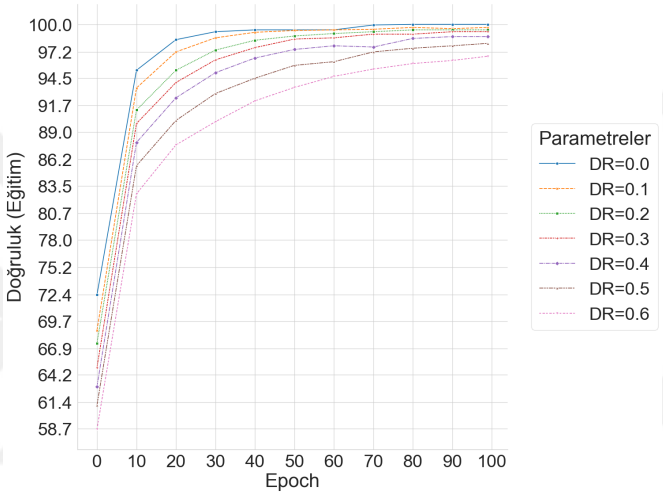


Şekil 4.4 Eğitim hızının test seti üzerinde sonuçları

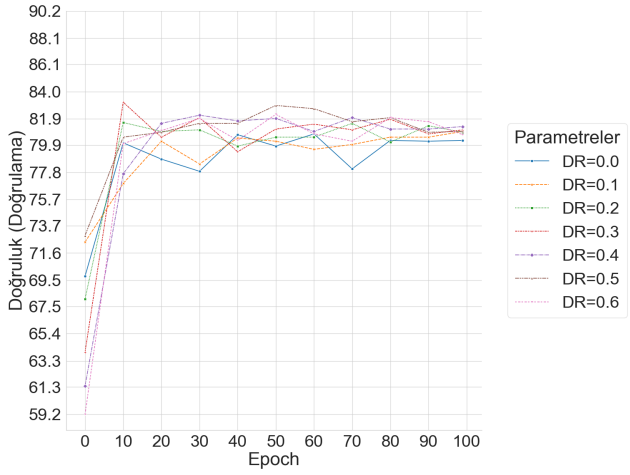
Şekil 4.4'te görüldüğü gibi 0.0005 öğrenme hızı 0.0001 değerinden biraz daha iyi doğruluk (%0.3) ve kesinlik (%1) sonucuna ulaşmıştır. Bundan sonraki deneylerde öğrenme hızı olarak 0.0005 değeri kullanılmıştır.

4.1.1.1.2 Dropout

Dropout parametresi ağdaki nöronların bir kısmının çıktısının (aktivasyon) sıfır olmasını sağlamaktadır. Dropout eklenerek ağın girdiyi ezberlemesini engellemek amaçlanmaktadır. LSTM ağına 0.0 ile 0.6 değerleri arasında (oranında) dropout eklenerek sonuçlar gözlemlenmiştir.

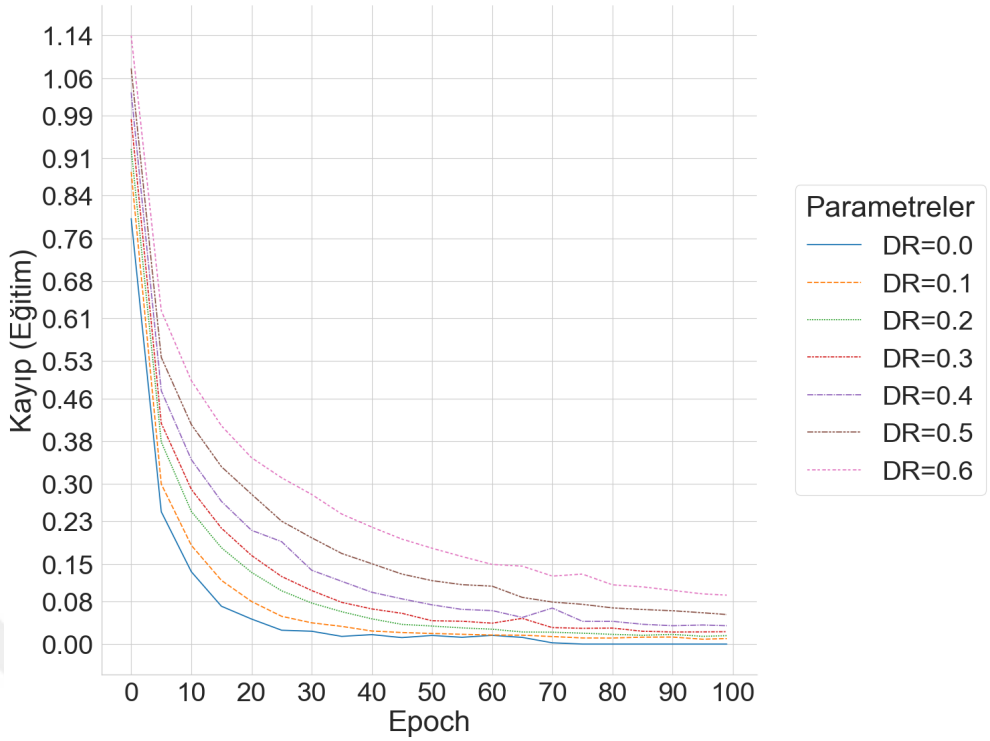


(a) Eğitim seti

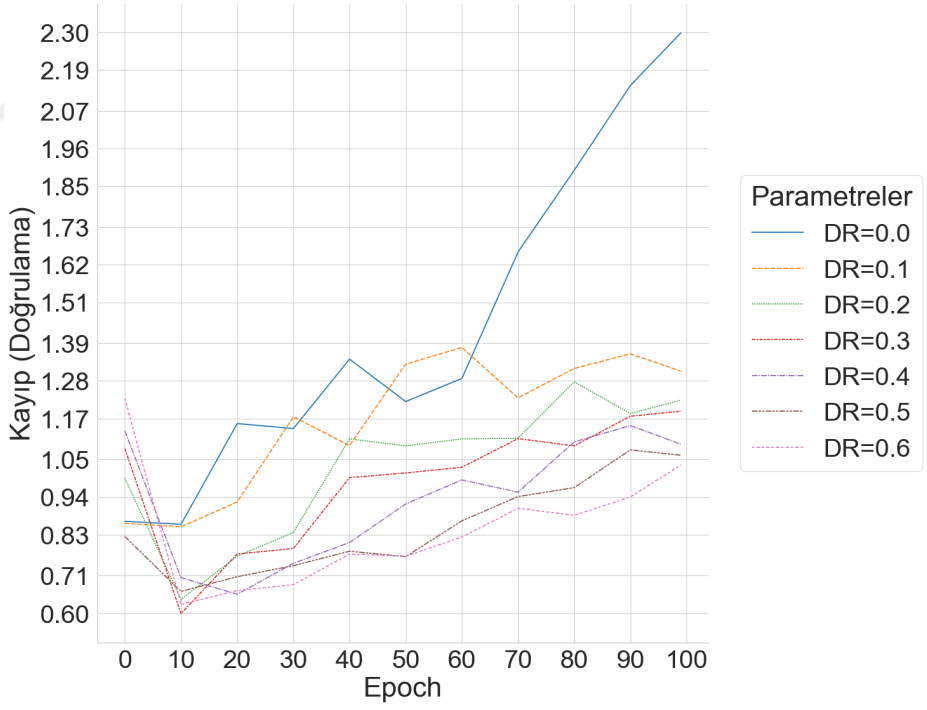


(b) Doğrulama seti

Şekil 4.5 Dropout eklenmesinin doğruluğa etkisi



(a) Eğitim seti

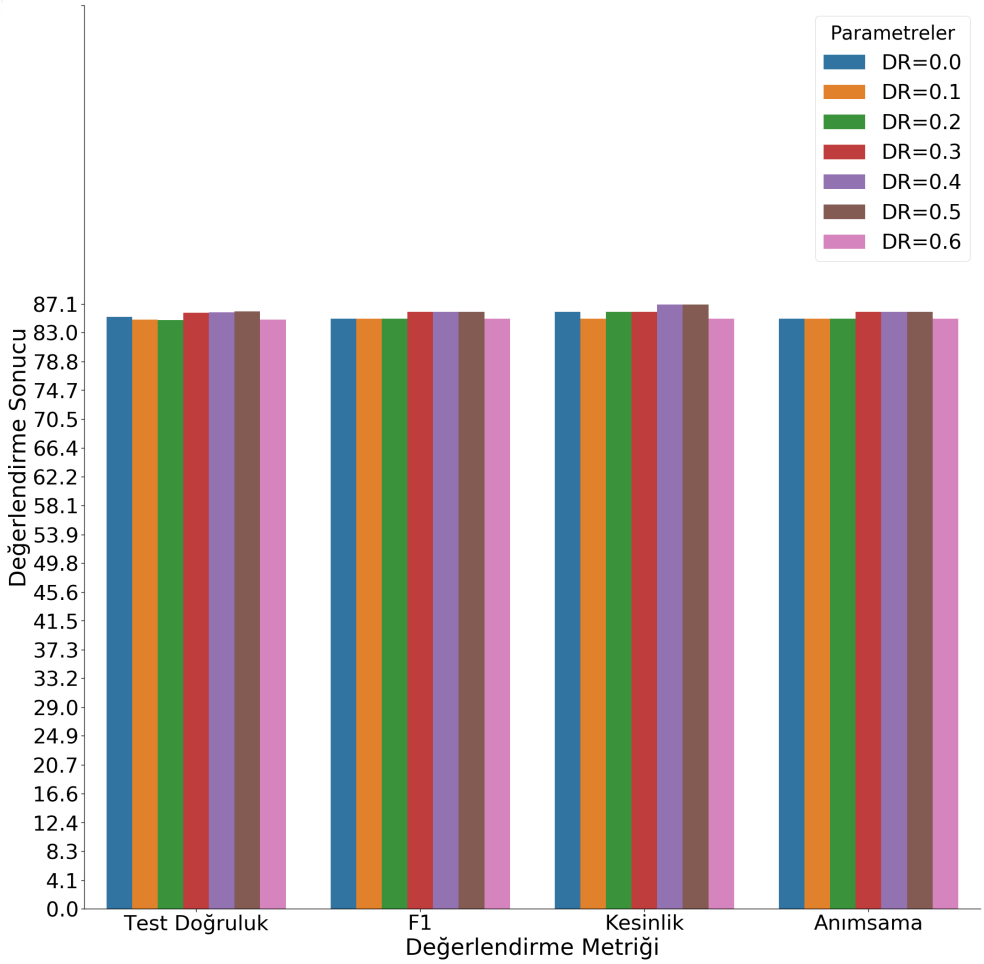


(b) Doğrulama seti

Şekil 4.6 Dropout eklenmesinin kayıp değerine etkisi

Dropout olmadığı (0.0) veya düşük olduğu (0.1, 0.2) durumlarda doğrulama setinde doğruluğun biraz daha düşük olduğu Şekil 4.5 üzerinde görülebilir. Doğrulama setinde 0.3, 0.4, 0.5 dropout oranları için doğruluk değerleri birbirine yakındır. Bu yüzden bu modellerin test seti üzerindeki sonuçları da karşılaştırılmıştır (Şekil 4.7).

Benzer şekilde dropout olmadığı (0.0) veya düşük olduğu (0.1, 0.2, 0.3) durumlarda eğitim setinde kayıp değerleri çok düşük iken doğrulama setinde kayıp oranları yüksektir (Şekil 4.6-a). Dropout 0.4-0.6 değerleri arasındaki iken kayıp değerleri doğrulama setinde daha düşük çıkmıştır (Şekil 4.6-b). Dropout kullanılarak ağın ezberlemesinin biraz önüne geçildiği Şekil 4.6 üzerinde görülebilir.



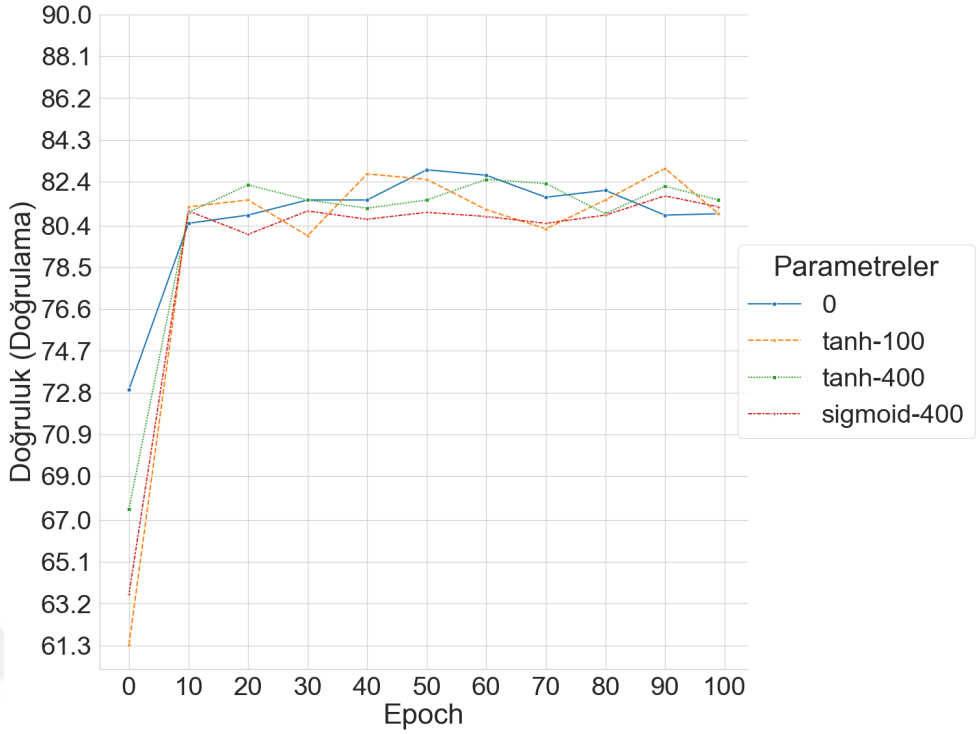
Şekil 4.7 Dropout değerlerinin test seti üzerinde sonuçları

Şekil 4.7 üzerinde gösterildiği gibi test seti üzerinde çeşitli dropout oranları kullanıldığında en iyi sonucu 0.4 ve 0.5 (0.4 dropout sonucundan %0.1 daha iyi) değerleri vermektedir. Bundan sonraki deneylerde dropout oranı olarak 0.5 değeri kullanılmıştır.

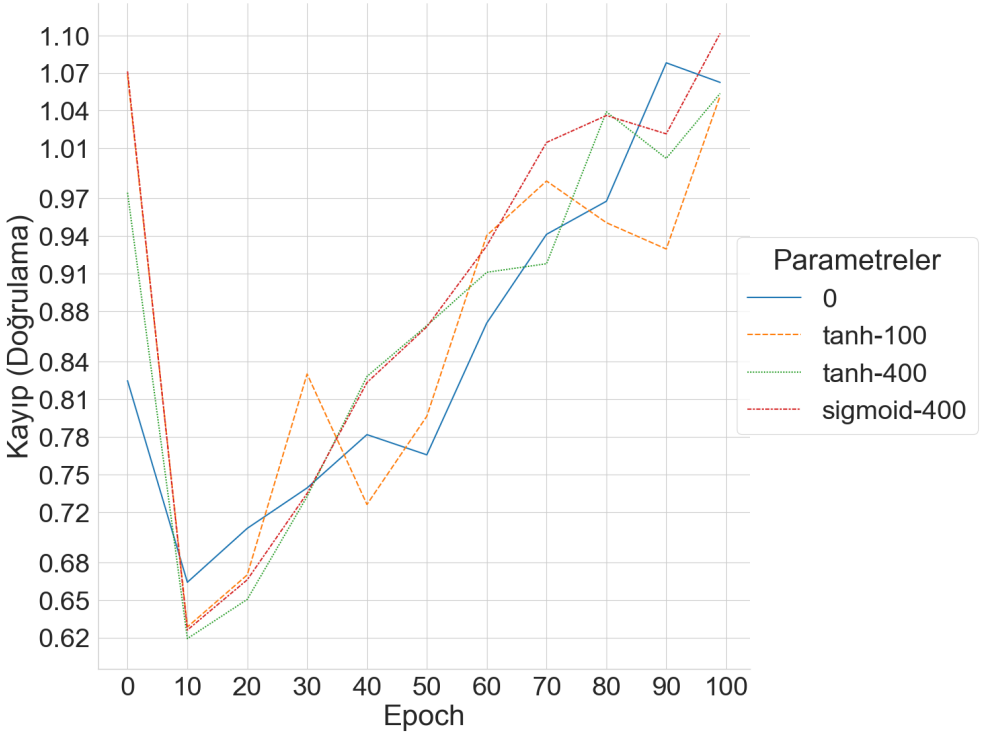
4.1.1.1.3 LSTM Sonrası Ek Katman

Şekil 4.1'de gösterilen mimaride LSTM katmanı ile çıkış katmanı arasına ek katman koyularak bunun sonuca etkisi araştırılmıştır. LSTM katmanının çıktı boyutu LSTM birim sayısı (unit size) kadardır. LSTM katmanından sonra 0, 100 ve 400 nöronlu ek katman eklenerek deneyler yapılmıştır. Bu katmanın aktivasyon fonksiyonu olarak Tanh ve Sigmoid fonksiyonları denenmiştir.

Ek katmanı eklemenin eğitim setinde doğruluğa veya kayıp değerine etkisi olmamıştır. Ek katman olmamasının doğruluk ve kayıp değerine olumlu etkisi Şekil 4.8'de gösterilmiştir.



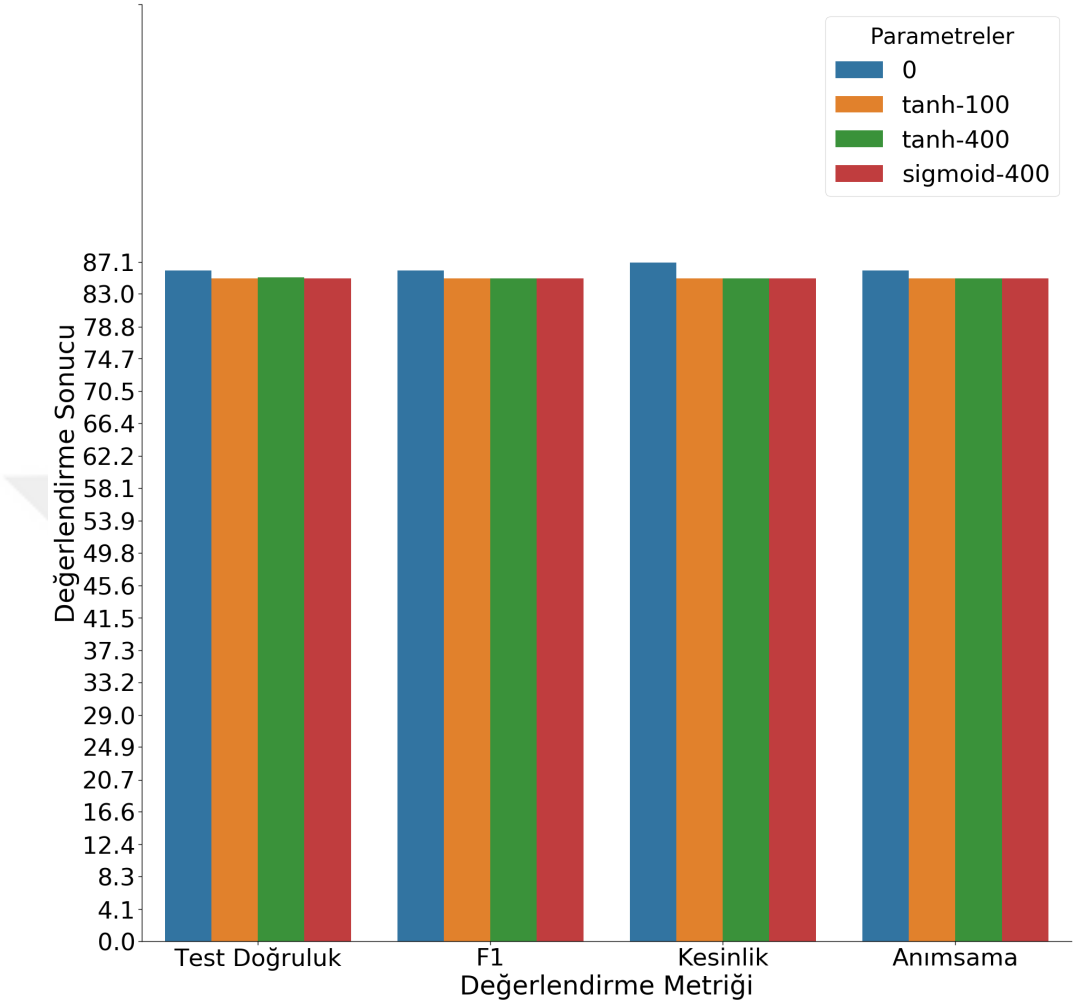
(a) Doğruluk



(b) Kayıp

Şekil 4.8 Ek katmanın doğrulama setinde sonuca etkileri

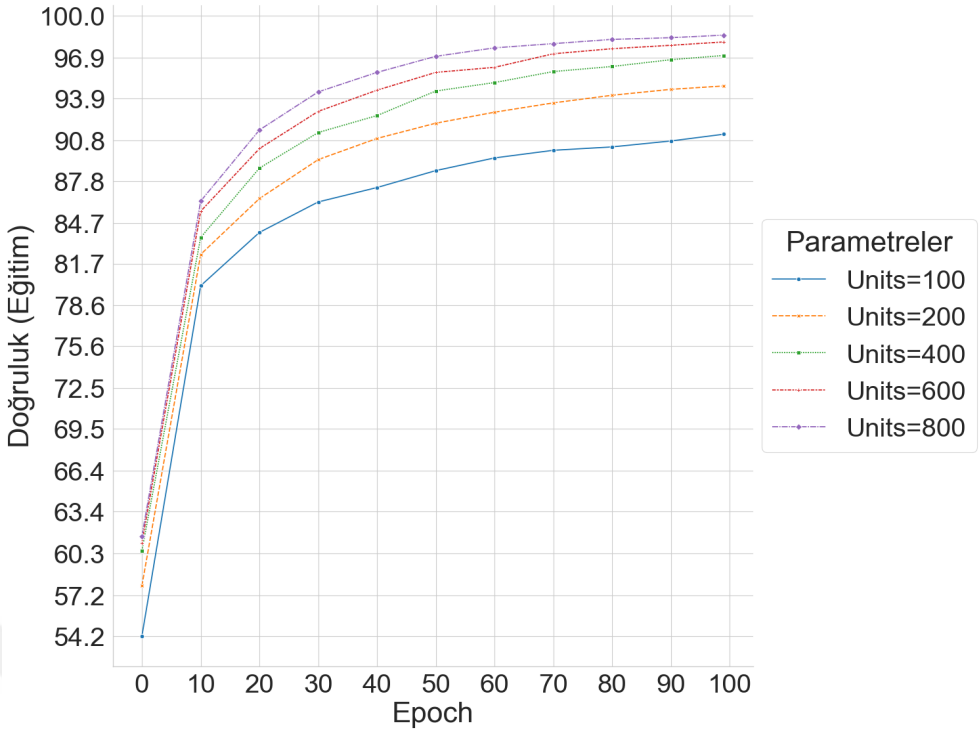
Test seti üzerinde deney yapıldığında ise ek katman kullanmamak %1 daha iyi doğruluk ve %2 daha iyi kesinlik üretmiştir (Şekil 4.9).



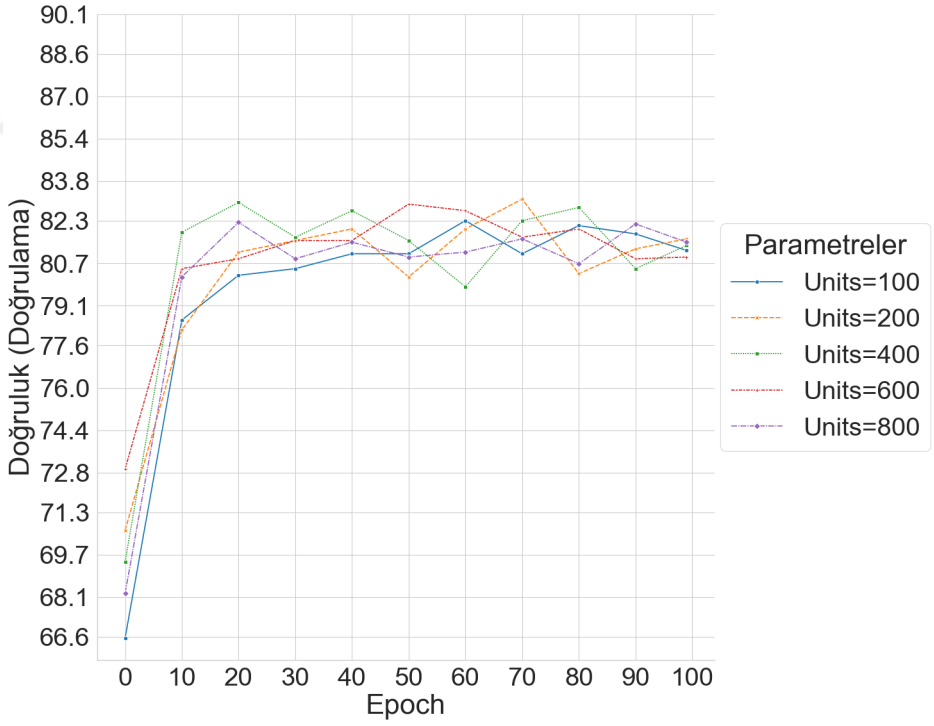
Şekil 4.9 Ek katmanın test seti üzerinde sonuçları

4.1.1.1.4 LSTM Birim Sayısı (Unit Size)

LSTM birim sayısı, LSTM hücresindeki nöron sayılarını ve aynı zamanda LSTM katmanının çıktı boyutunu belirten parametredir. 100, 200, 400, 600 ve 800 olmak üzere beş farklı birim sayısı ile deneyler yapılmıştır.



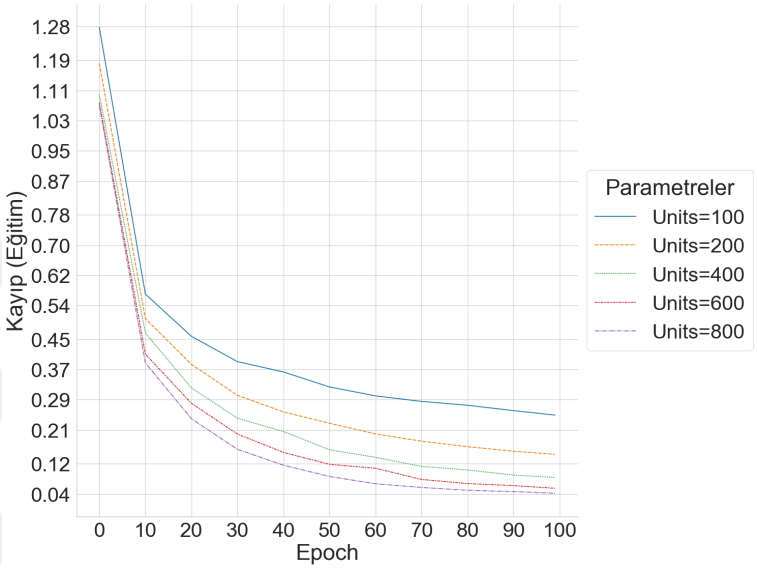
(a) Eğitim seti



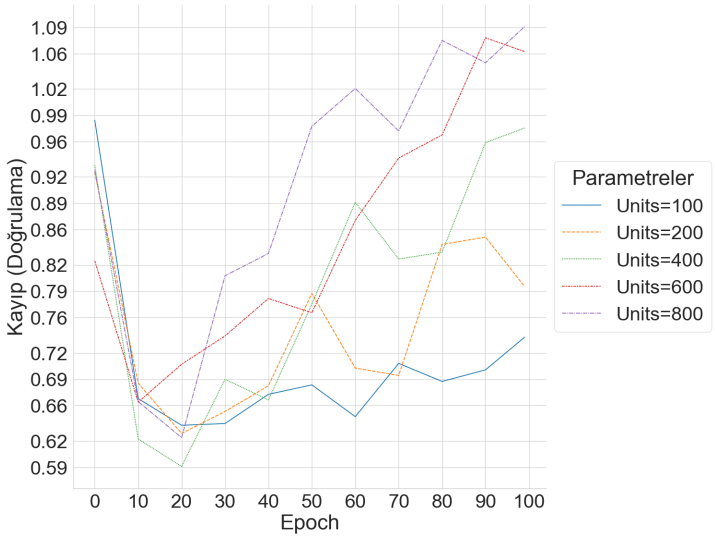
(b) Doğrulama seti

Şekil 4.10 LSTM birim sayısının doğruluğa etkisi

Şekil 4.10 üzerinde görüldüğü gibi eğitim setinde birim sayısı arttıkça doğruluk artmaktadır. Benzer oranda kayıp değeri birim sayısı arttıkça düşmektedir (Şekil 4.11). Doğrulama setinde sınıflandırma sonucuna bakıldığında farklı birim sayıları için doğruluğun yakın olduğu görülebilir. Doğrulama setinde kayıp değerlere bakıldığında ise birim sayısı arttıkça kayıp değerinin de yükseldiği söylenebilir (Şekil 4.11).



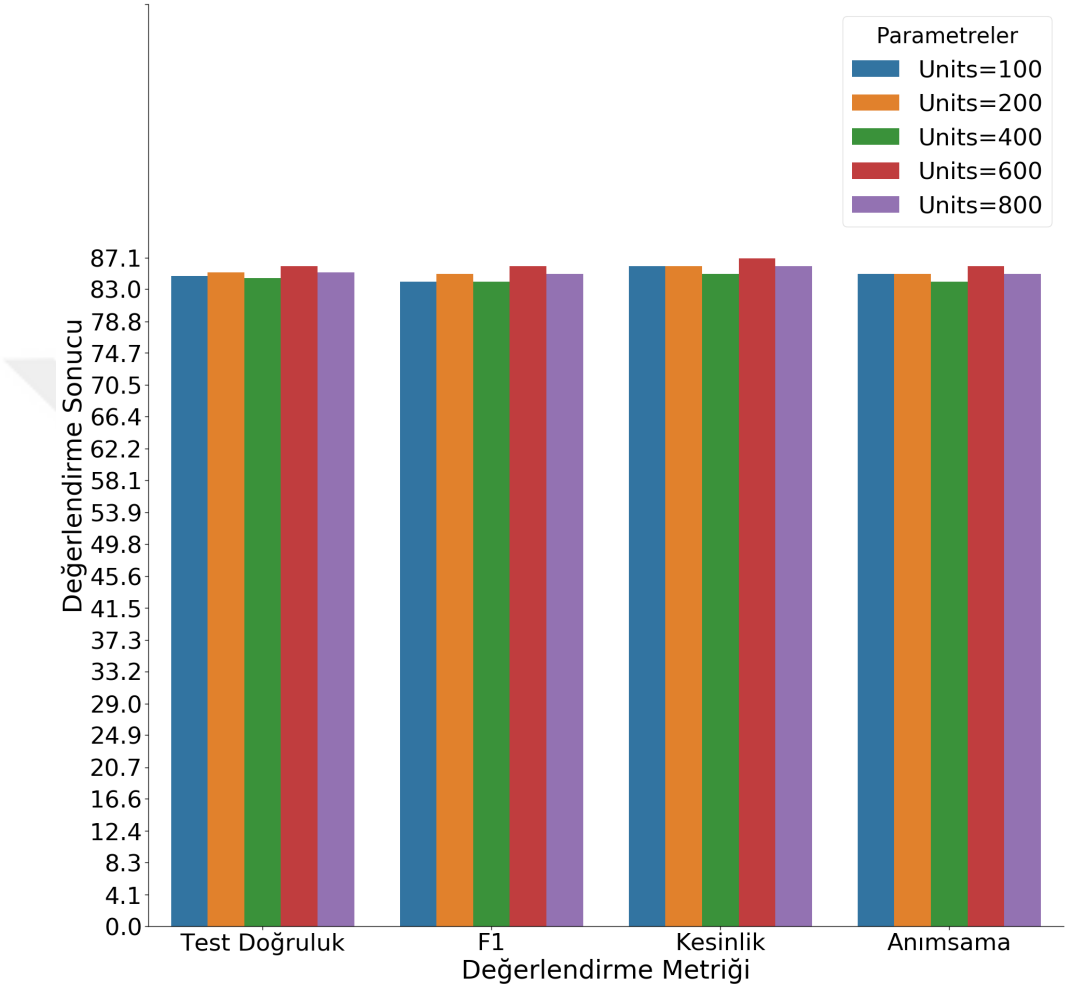
(a) Eğitim seti



(b) Doğrulama seti

Şekil 4.11 LSTM birim sayısının kayıp değerine etkisi

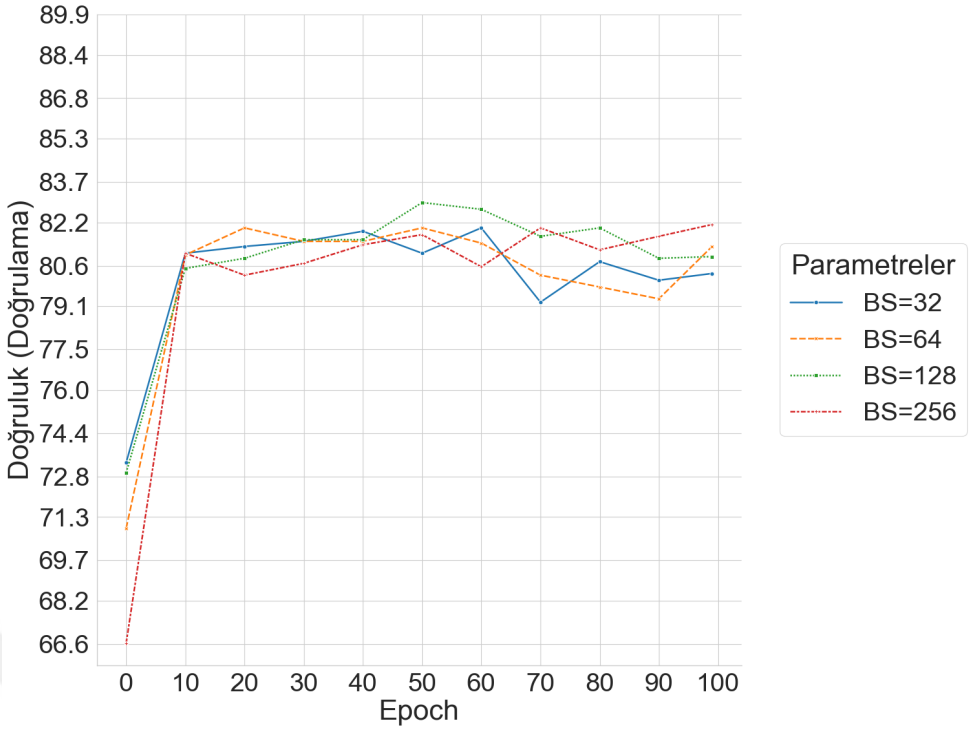
Doğruluk oranını karşılaştırmak için test seti üzerinde de sınıflandırma sonuçları alınmıştır. Test seti üzerinde yapılan deneylerde 600 birim sayısı kullanmak doğruluk, F1, kesinlik ve anımsama metriklerinde en az %1 daha iyi sonuç vermiştir (Şekil 4.12).



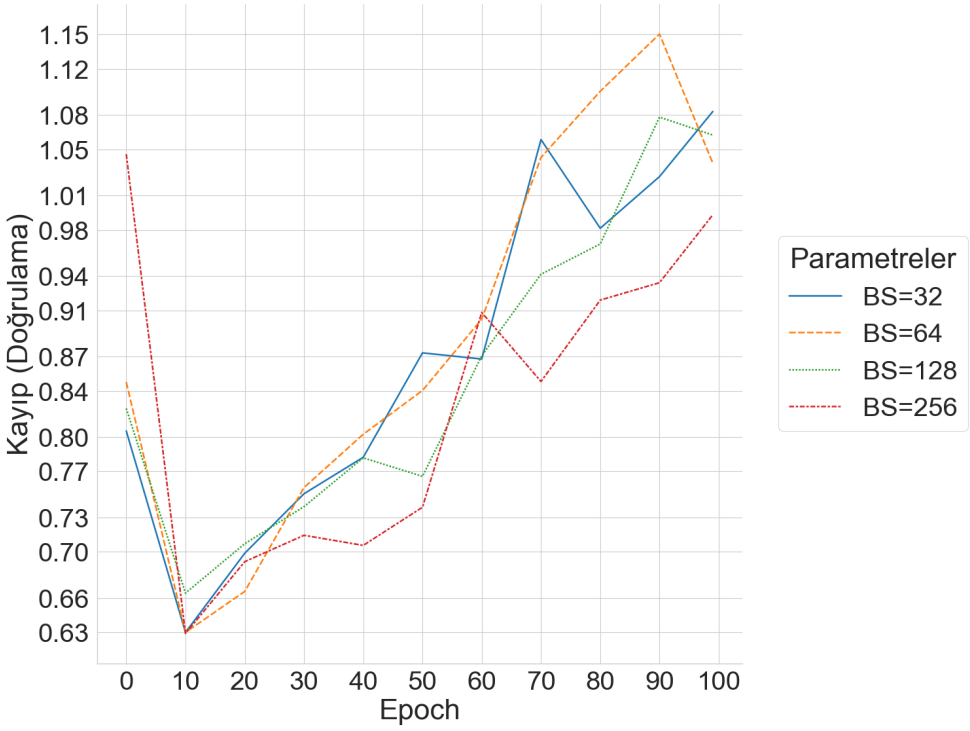
Şekil 4.12 LSTM birim sayısının test seti üzerinde sonuçları

4.1.1.1.5 Batch size

Batch size parametresi her eğitim epoch'unda ne kadar örnek ile eğitim yapıldıktan sonra ağ ağırlıklarının güncelleneceğini belirtmektedir. Batch size genelde 32'nin katları oranında seçilmektedir. Batch size ne kadar büyük olursa eğitim o kadar hızlı yapılmaktadır. Aynı zamanda batch size büyük olması model ile tahmin etme süresini azaltmaktadır. 32, 64, 128 ve 256 batch size değerleriyle deneyler yapılmıştır.



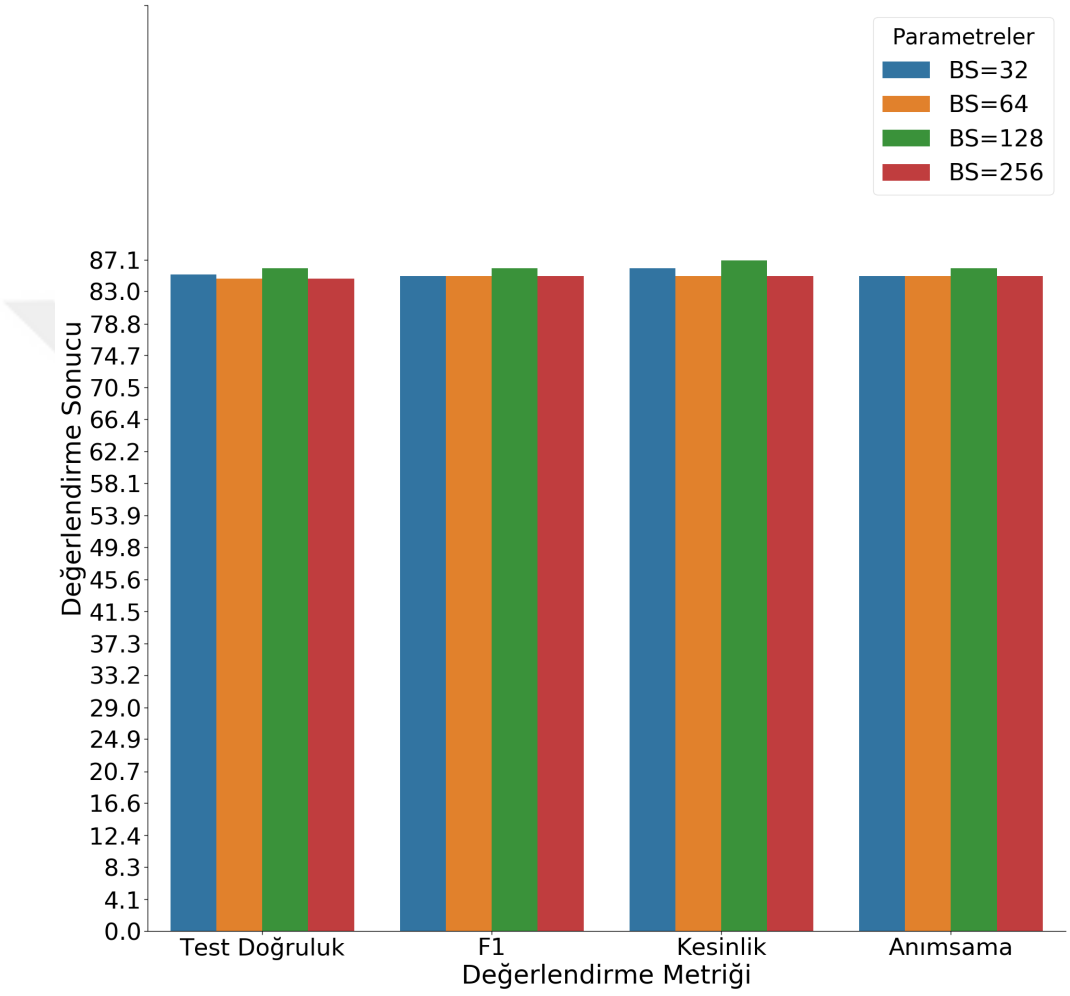
(a) Doğruluk



(b) Kayıp

Şekil 4.13 Batch size parametresinin doğrulama setinde sonuca etkileri

Eğitim setinde batch size parametresinin doğruluğa ve kayıp değerine büyük etkisi olmamıştır. Doğrulama setinde 128 ve 256 değerlerinin daha iyi sonuç verdiği Şekil 4.13'te görülebilir. Test seti üzerinde yapılan deneylerde batch size parametresini 128 yapmak doğruluk, F1, kesinlik ve anımsama metriklerinde en az %1 daha iyi sonuç vermiştir (Şekil 4.14).

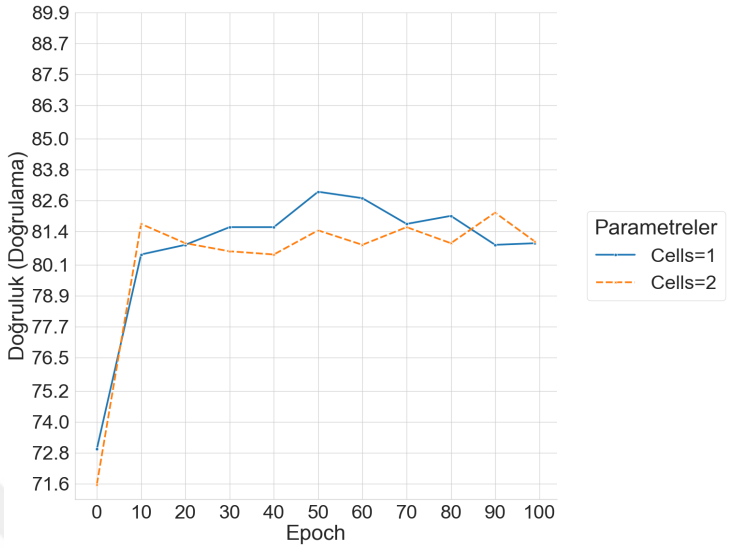


Şekil 4.14 LSTM batch size test seti üzerinde sonuçları

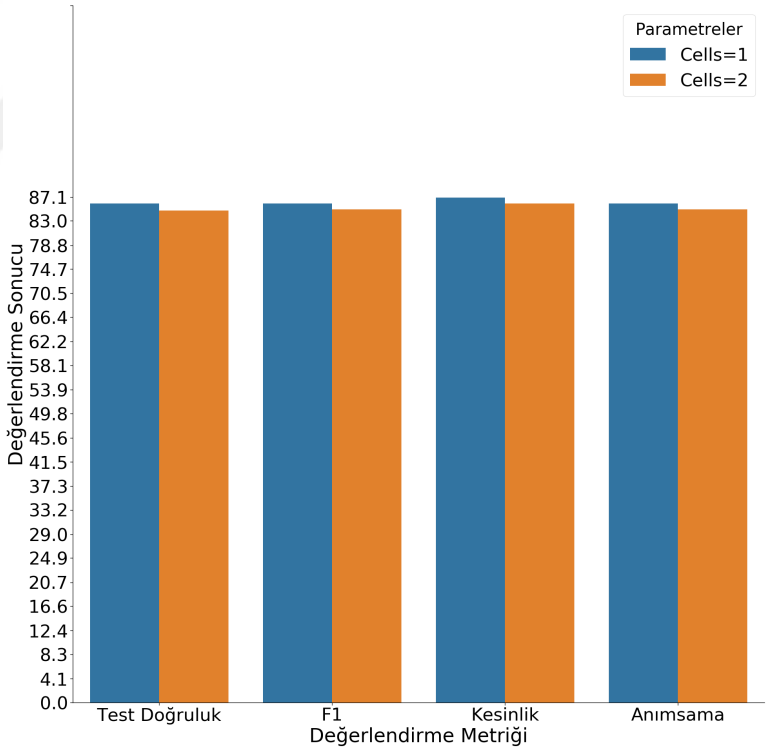
4.1.1.1.6 LSTM Hücre Sayısı (LSTM Cells)

Sınıflandırma ağında tek LSTM hücre (katman) kullanmak yerine birden fazla LSTM hücresi kullanılabilir. Bunun sonuca etkisini öğrenmek için iki hücreli LSTM ağı ile deney yapılmıştır. Bir ve iki hücreli LSTM deneylerinde eğitim seti üzerinde doğruluk ve kayıp değerleri çok yakındır. Doğruluk ve test setinde ise bir hücreli

LSTM daha iyi doğruluk, F1, kesinlik, anımsama ve kayıp değerleri üretmiştir (Şekil 4.15).



(a)

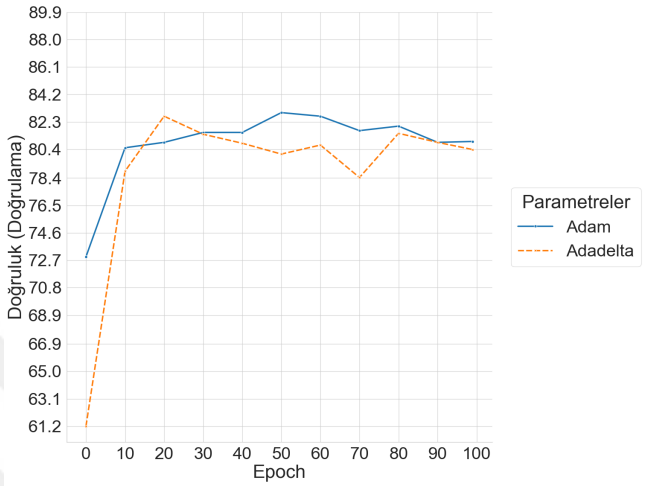


(b)

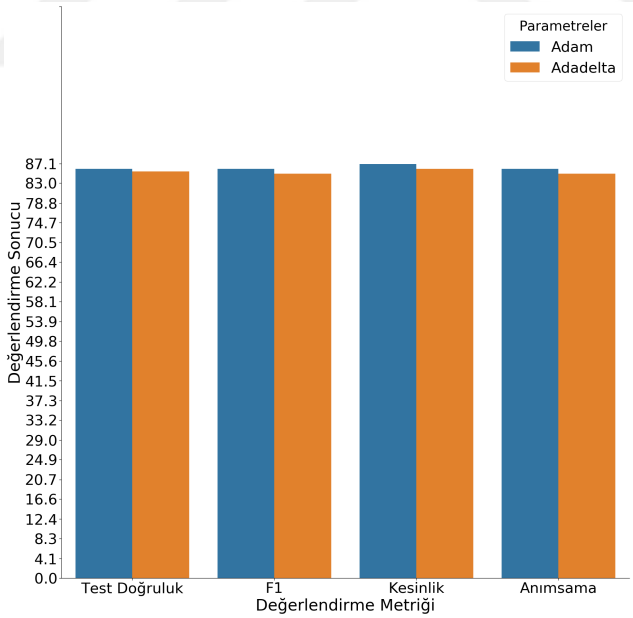
Şekil 4.15 (a) LSTM hücre sayısının doğruluğa etkisi. (b) LSTM hücre sayısının test seti üzerinde sonuçları

4.1.1.1.7 Eniyileştirici Algoritması

Eğitimde en önemli seçimlerden biri eniyileştirici algoritmasıdır. Modelde öğrenme hızını, ivmeyi ayarlar, ağırlıkların güncellenmesini sağlar ve hatayı minimuma indirgemeye çalışır. Bu çalışmada uyarlanabilir öğrenme hızı ile eğitim yapabilen Adam ve Adadelta eniyileştirici algoritmaları karşılaştırılmıştır.



(a)



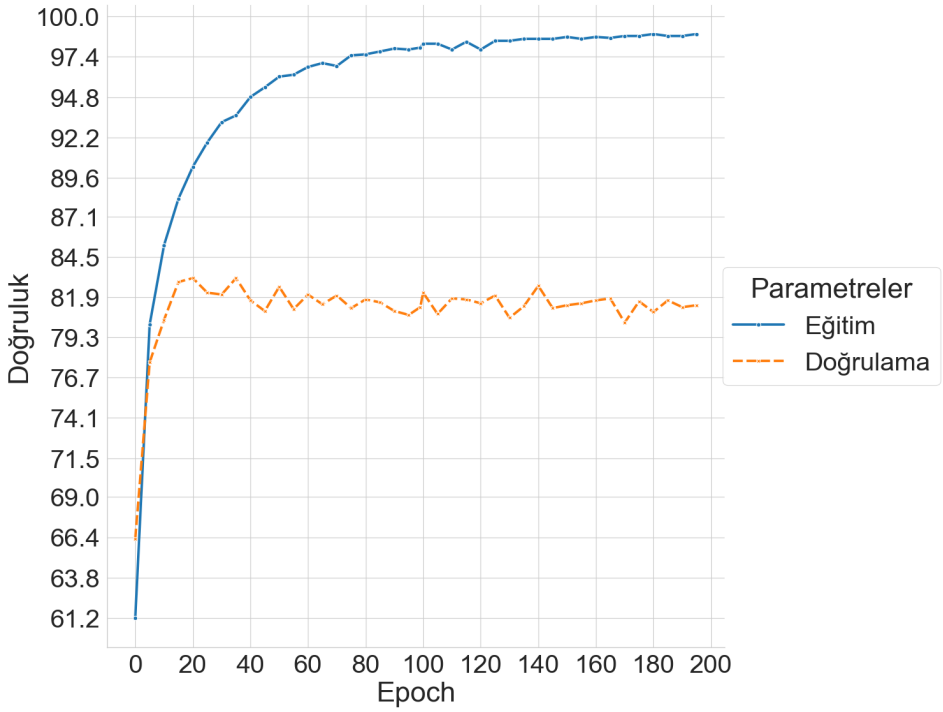
(b)

Şekil 4.16 (a) Eniyileştirici algoritmasının doğruluğa etkisi. (b) Eniyileştirici algoritmasının test seti üzerinde sonuçları

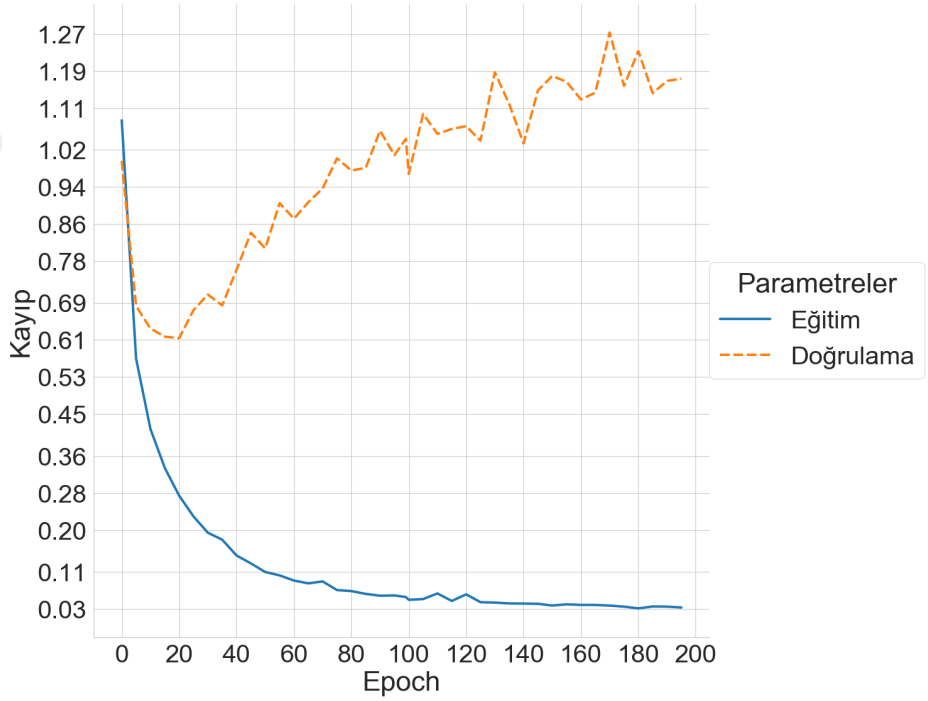
Doğruluk, F1, kesinlik, anımsama ve kayıp metriklerinde Adam algoritmasının daha iyi sonuç ürettiği Şekil 4.16 üzerinde görülebilir.

4.1.1.1.8 Epoch

Eğitimde bütün örneklerin üzerinden bir kere geçilmesine epoch denmektedir. Epoch sayısı eğitimin ne zaman duracağını da belirtmektedir. Eğitimde belirli epoch sayısından sonra ağ doyuma ulaşmaktadır ve daha fazla eğitim yapmak ağın girdiyi ezberlemesine neden olabilmektedir. Şekil 4.17 üzerinde görüldüğü gibi 100 epoch bir eğitim için yeterli olmaktadır. Bütün deneylerde her 2, 5 veya 10 epoch'tan sonra ağ modeli kayıt edilmiştir ve daha sonra en iyi sonucun bulunduğu epoch'a ait model ile testler gerçeklenmiştir.



(a)

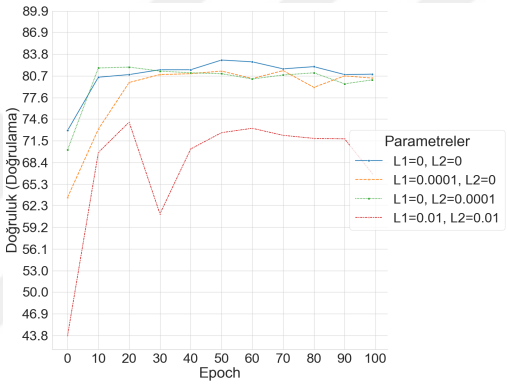


(b)

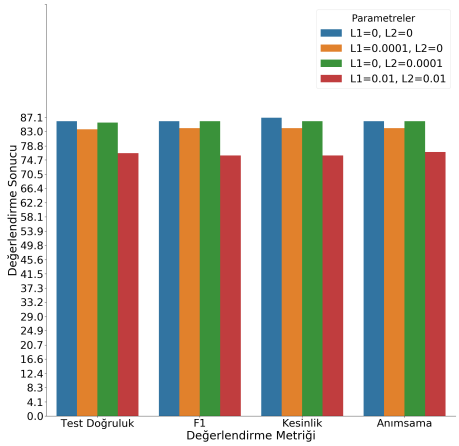
Şekil 4.17 (a) Eğitim ve doğrulama setinde doğruluk (b) Eğitim ve doğrulama setinde kayıp

4.1.1.1.9 Düzenleme (Regularization)

Sınıflandırma ağının ezberlemesini önlemek için kullanılabilecek bir diğer yöntem L1, L2 düzenlemelerini kullanmaktır. Şekil 4.18 üzerinde görülebildiği gibi L1 ve L2 düzenleme tekniklerini birlikte varsayılan değerleri ile kullanıldığında doğruluk azalmakta ve kayıp artmaktadır. L2 ve L1 teknikleri karşılaştırıldığında L2 düzenleme kullanmak daha iyi doğruluk ve kesinlik sonucu vermektedir. Eğitim setinde doğruluk ve kayıp değerine düzenleme tekniğinin çok büyük etkisi olmamıştır. Hiç düzenleme tekniği kullanılmadığında ise L2 tekniği kullanarak test seti üzerinde elde edilen doğruluktan %0.4, kesinlikten %1 daha iyi sonuç elde edilmiştir.



(a)

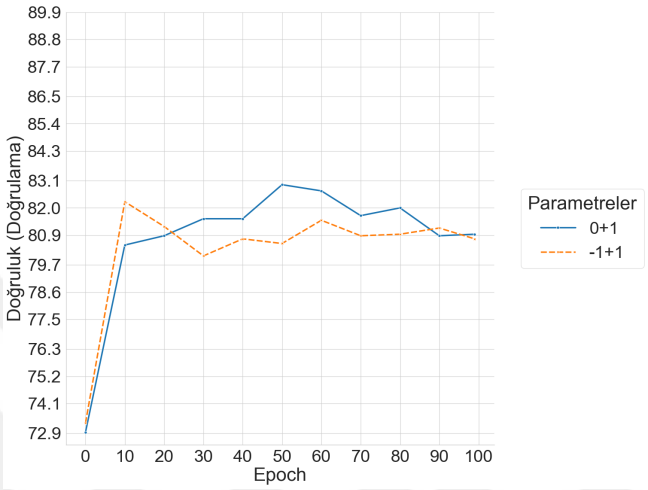


(b)

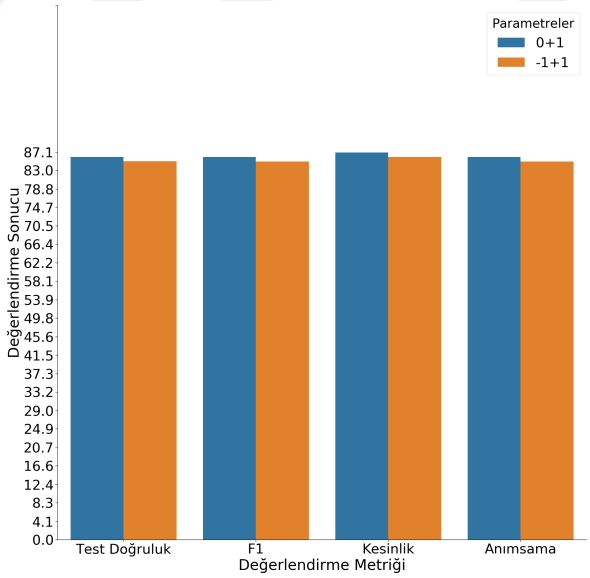
Şekil 4.18 (a) L1, L2 düzenleme tekniğinin doğruluğa etkisi. (b) L1, L2 düzenleme tekniğinin test seti üzerinde sonucu

4.1.1.1.10 Normalleştirme (Normalization)

Veriler LSTM ağına girdi olarak verilmeden önce 0+1 değerleri arasına veya -1+1 değerleri arasına sıkıştırılarak normalleştirilmiştir. Normalleştirilme yapılmadığında farklı değerler arasında olan özellikler, ağdaki ağırlıkları farklı şekilde etkileyebilmektedir. Normalleştirme yapılmasının sebebi her özelliğin eşit ağırlıkta olmasını ve ağın ağırlıklarını eşit şekilde güncellemesini sağlamaktır.



(a)



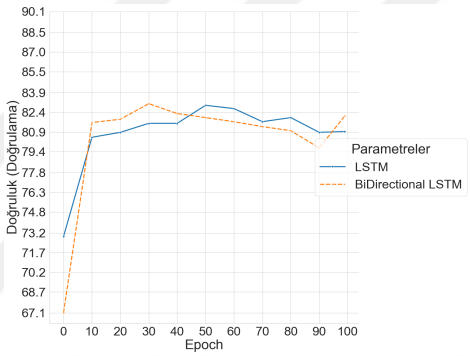
(b)

Şekil 4.19 (a) Normalleştirme yöntemlerinin doğruluğa etkisi. (b) Normalleştirme yöntemlerinin test seti üzerinde sonuçları

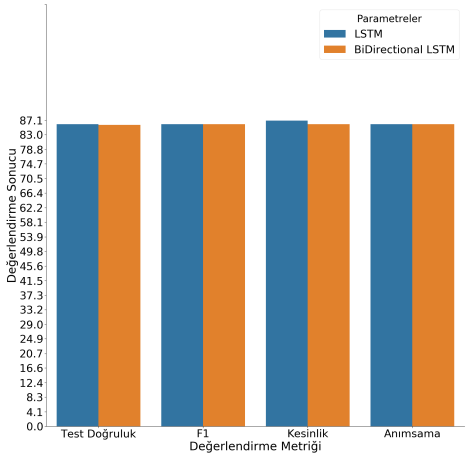
Şekil 4.19'da görüldüğü gibi 0+1 değerleri arasında veriyi Normalleştirmek doğruluk, F1, kesinlik ve anımsama metriklerinde %1 daha iyi sonuç üretmiştir.

4.1.1.1.11 Çift Yönlü LSTM

LSTM hücresi tek yönlü veya çift yönlü olabilmektedir. Tek yönlü LSTM hücresi, ileriye (zaman) doğru hareket etmektedir ve geçmişe ait bağlam bilgisini bellekte tutmaktadır. Çift yönlü LSTM hücresinde hem ileri zamana doğru hareket edilir ve geçmiş bağlam tutulur, hem de girdiler tersten verilerek ileriden geriye doğru hareket edilir ve gelecek bağlam tutulur. Bu şekilde çift yönlü LSTM'de daha geniş bağlam bilgisi tutulabilir. Ama çift yönlü LSTM her problem alanına uygun değildir.



(a)



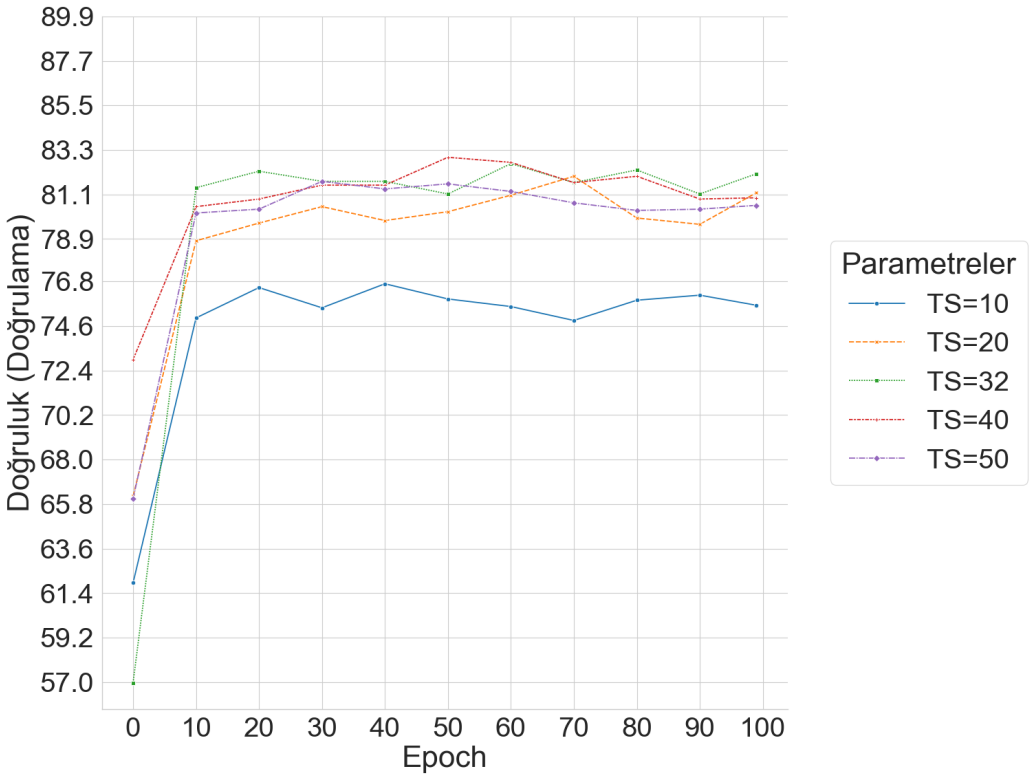
(b)

Şekil 4.20 (a) Tek ve çift yönlü LSTM'in doğruluğa etkisi. (b) Tek ve çift yönlü LSTM'in test seti üzerinde sonuçları

Yapılan deneylerde tek ve çift yönlü LSTM sonuçları yakın olup, tek yönlü LSTM kesinlik olarak %1 daha iyidir (Şekil 4.20).

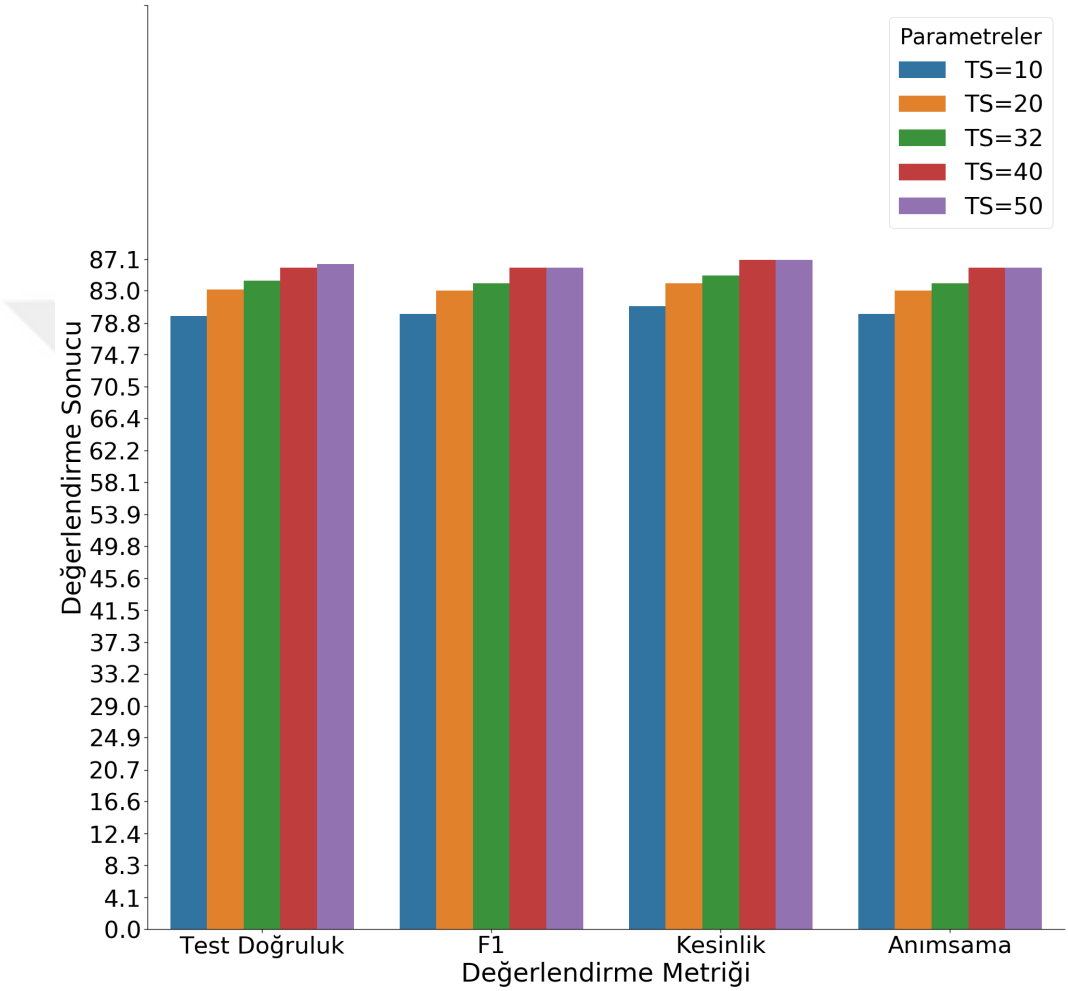
4.1.1.1.12 Zaman Adım Sayısı

LSTM uygun parametreleri seçildikten sonra girdi olarak verilen pencereler farklı boyutlarda bölümlere ayrılmıştır ve eğitim sonuçları karşılaştırılmıştır. Şekil 4.21 üzerinde görüldüğü gibi girdi daha az (10) bölümlere ayrılınca doğruluk düşük olmaktadır. 32, 40 ve 50 tane bölüm halinde girdileri vermek ise daha iyi ve yakın sonuçlar üretmiştir. Özellikler bölümler içinde eksen bazında çıkartıldığı için bölümlerdeki değerlerin boyutu 12'ye (12 eksen var) bölünebilir olmalıdır. Eğer girdi 16 saniyelik pencere ise ve 9600 uzunluğunda veri ise, girdiyi 32 bölüme bölmek her bölümde 300 tane değer olduğu anlamına gelmektedir. 300 12'ye tam bölünmektedir ve eksen bazında özellikler çıkartılabilir. Eğer girdi 30 parçaya bölünürse her bölüm 320 değerden oluşur ve 12'ye tam bölünemez, eksen bazında özellikler çıkartılmaz.



Şekil 4.21 Girdinin farklı boyutlarda bölümlere ayrılmasının doğruluğa etkileri

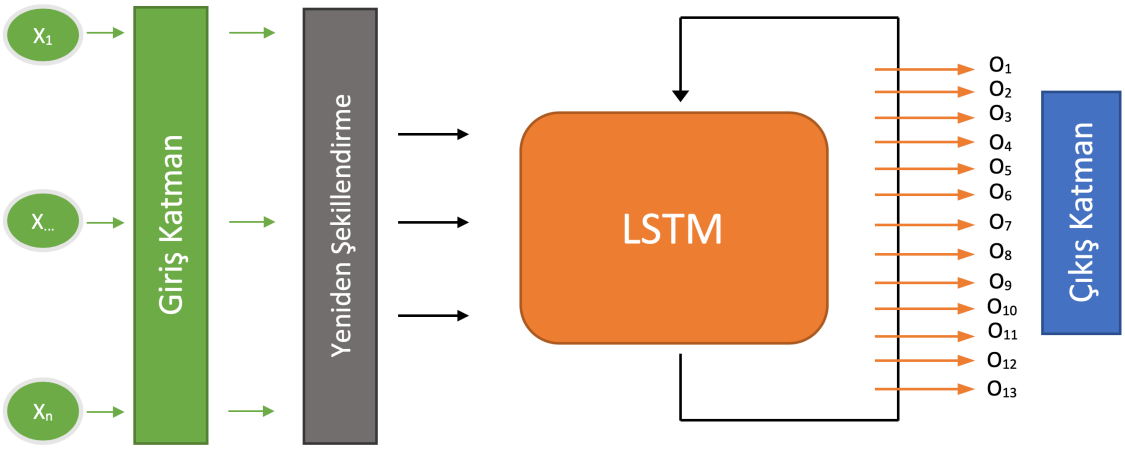
Şekil 4.22'de görülebildiği gibi, bölüm sayısı arttıkça bütün metrikler (doğruluk, F1, kesinlik, anımsama) daha iyi başarı vermektedir. Donanım kısıtlamasından dolayı daha fazla bölümler üzerinde deneyler yapılamamıştır. Pencereleri 50 bölüme ayırarak ağa girdi olarak vermek %86.4 doğruluk (40 bölümde %86) ile en iyi sonucu üretmiştir.



Şekil 4.22 Girdinin farklı boyutlarda bölümlere ayrılmasının test seti üzerinde sonuca etkileri

4.1.1.2 LSTM Mimarisi ve Sonuç

Uygun parametreler seçildikten sonra LSTM mimarisi Şekil 4.23'te gösterildiği gibi tek hücreli, tek yönlü LSTM'den oluşmaktadır.



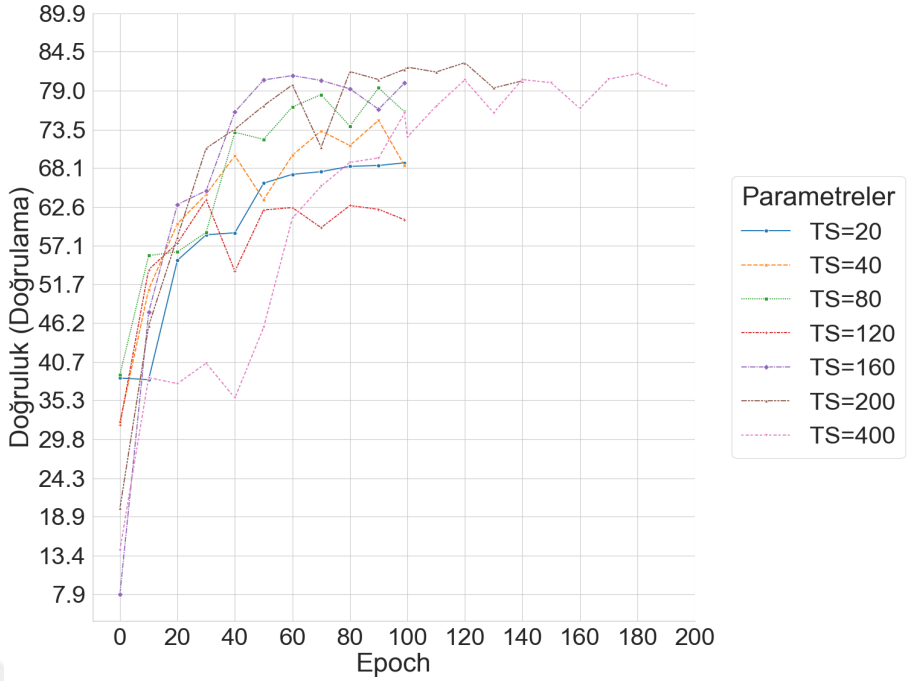
Şekil 4.23 LSTM sınıflandırma ağı mimarisi

Bu mimari ve seçilen uygun parametreler ile test seti üzerinde elde edilen en iyi sonuç %86.4 doğruluk, %86 F1, %87 kesinlik, %86 anımsamadır.

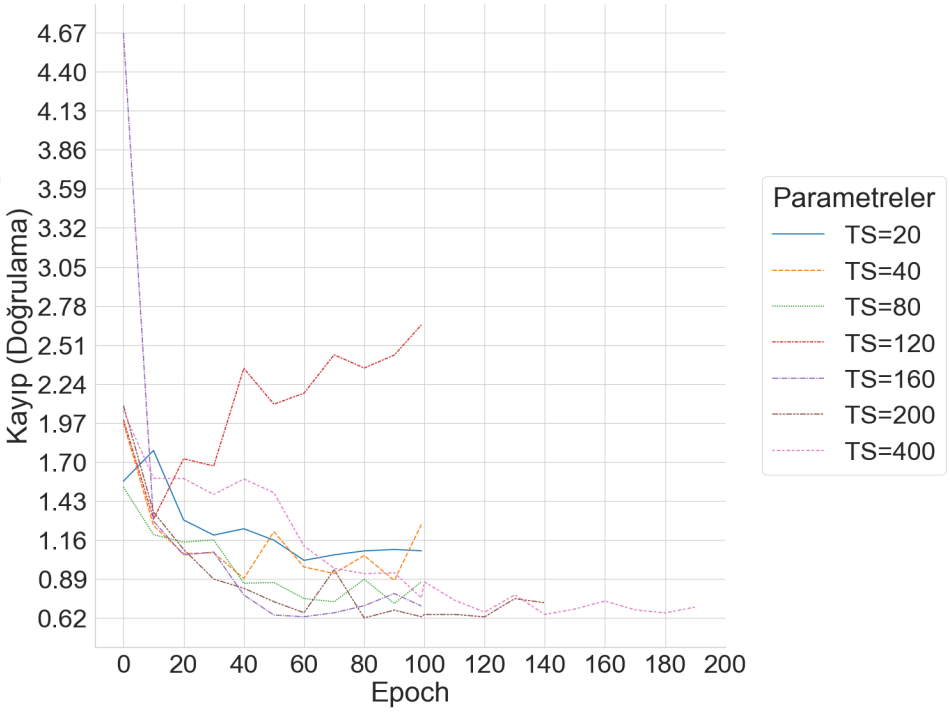
4.1.2 LSTM ve Ham Veri Özellikleri

Herhangi ayırt edici özellikler çıkartılmadan, sensörlerden toplanan ham veriler kullanılarak LSTM ağı ile eğitim yapılmıştır. Ham veriler, pencereler oluşturulduktan ve değerler 0+1 arasına Normalleştirme yapıldıktan sonra LSTM ağına girdi olarak verilmiştir. Buradan elde edilen sonuçlar zaman alanında özellikler çıkartılarak eğitilen model sonuçları ile karşılaştırılmıştır.

Çeşitli sayıda bölümlere ayrılan veriler ile yapılan sınıflandırma deney sonuçları Şekil 4.24'te gösterilmiştir. Girdiler 20, 40, 80, 120, 160, 200 ve 400 bölüme ayrılarak LSTM ağına verilmiştir. Bölüm sayısı arttıkça doğruluk artmaktadır (Şekil 4.24). Veri 160, 200 bölüme ayrıldığında ise sonuçlar doyuma ulaşmıştır. Girdi 400 bölüme ayrılınca doğruluk çok az düşmeye başlamıştır. 20 bölümden 200 bölüme kadar doğruluk artarken veri 120 bölüme ayrıldığında çok düşük doğruluk sonucu vermiştir (Şekil 4.24). Bunun sebebi 9600 uzunluğundaki girdi 120 bölüme ayrıldığında her bölümün uzunluğu 12'ye tam bölünmemektedir ve her bölüm 12 eksene ait verileri eksiksiz bulundurmamaktadır.



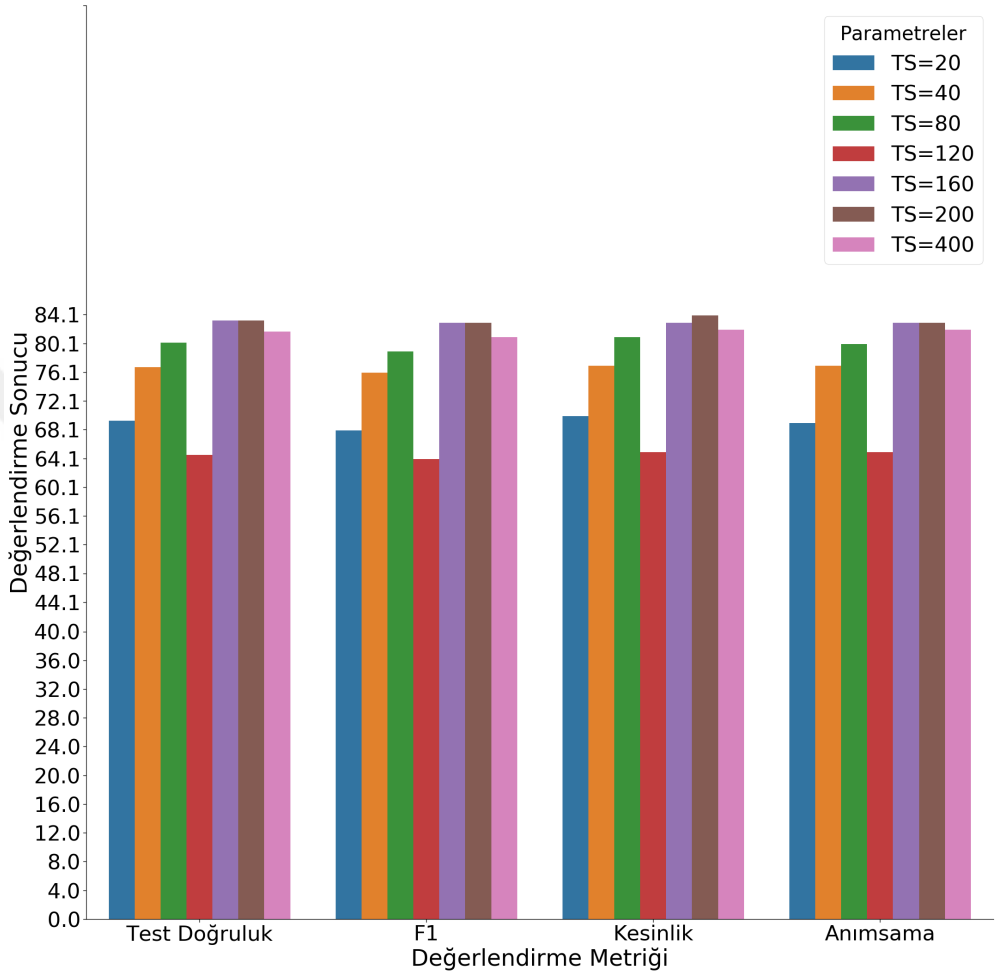
(a)



(b)

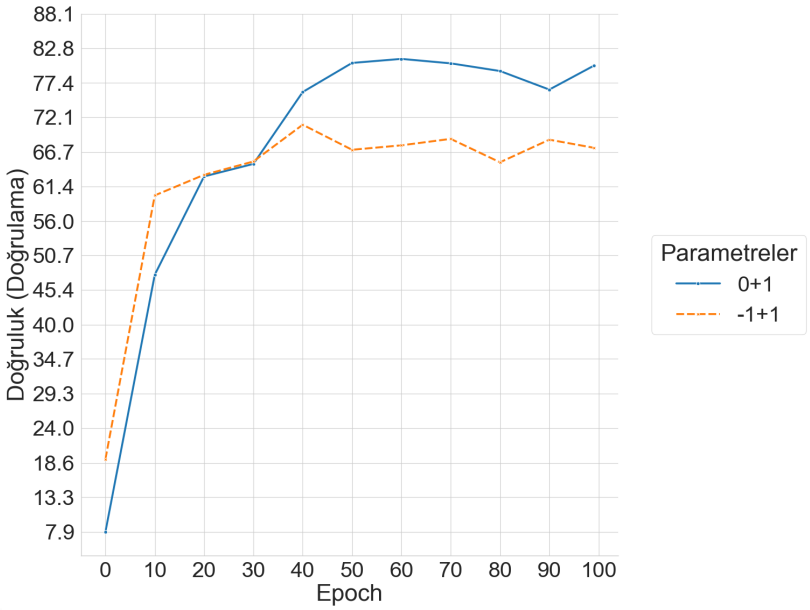
Şekil 4.24 Ham verinin farklı boyutlarda bölümlere ayrılarak LSTM ağında kullanılmasının doğruluğa (a) ve kayıp değerine (b) etkileri

Doğrulama setinde 160 ve 200 bölüme ait sonuçlar en iyi sonuçları vermiştir ve doğruluk grafiğinde sonuçlar yakındır. Test seti üzerinde yapılan deney sonuçları Şekil 4.25'te gösterilmiştir. Ham veriler 200 bölümler halinde girdi olarak verildiğinde en iyi sonuç olan %83.3 doğruluk ve %84 kesinlik elde edilmiştir.

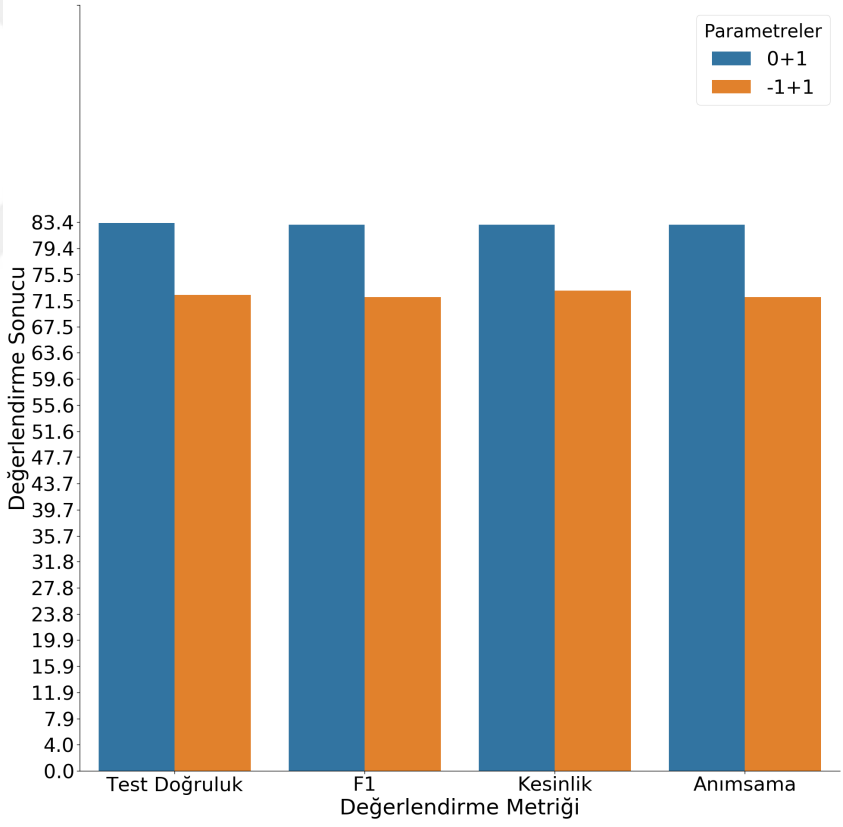


Şekil 4.25 Ham verinin farklı boyutlarda bölümlere ayrılarak LSTM ağında kullanılmasının test setinde sonuca etkileri

Ham veriyi -1+1 değerleri ve 0+1 değerleri arasına Normalleştirmenin sonuca etkileri Şekil 4.26'da gösterilmiştir. Zaman alanında çıkartılmış özellikler kullanıldığında Normalleştirme yöntemleri arasında %1 sonuç farkı vardı (Şekil 4.19). Ham veride 0+1 değerleri arasına Normalleştirme yapmak, -1+1 değerleri arasına Normalleştirme yapmaktan %11 daha iyi doğruluk, %10 daha iyi kesinlik sonucu üretmiştir (Şekil 4.26).



(a)



(b)

Şekil 4.26 (a) Normalleştirme yöntemlerinin ham veride doğruluğa etkisi. (b) Normalleştirme yöntemlerinin ham veride test seti üzerinde sonuçları

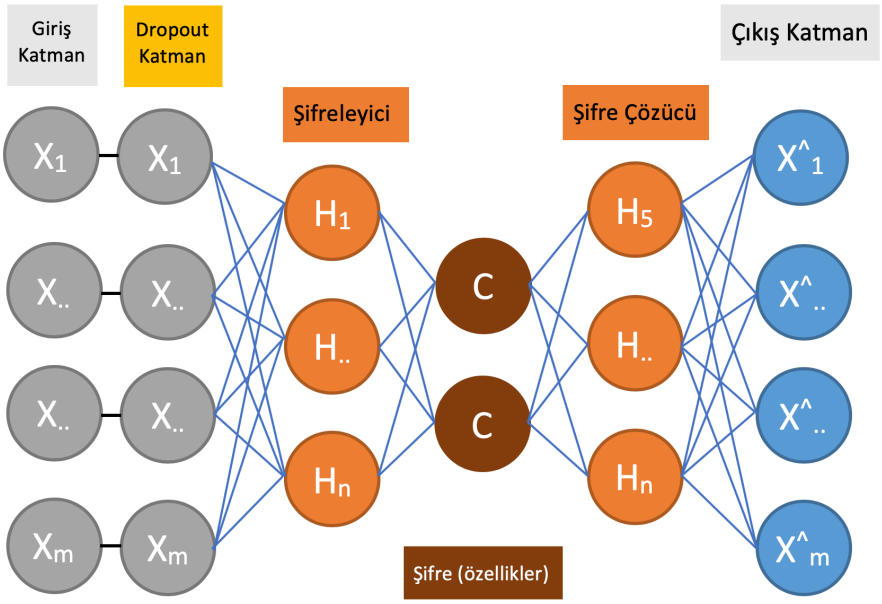
Herhangi özellik çıkartmaksızın, ham veriler ile LSTM ağı eğitilerek elde edilen en iyi sonuç %83.3 doğruluk, %84 kesinlik, %83 F1, %83 kesinliktir.

4.1.3 LSTM ve Oto Kodlayıcı Özellikleri

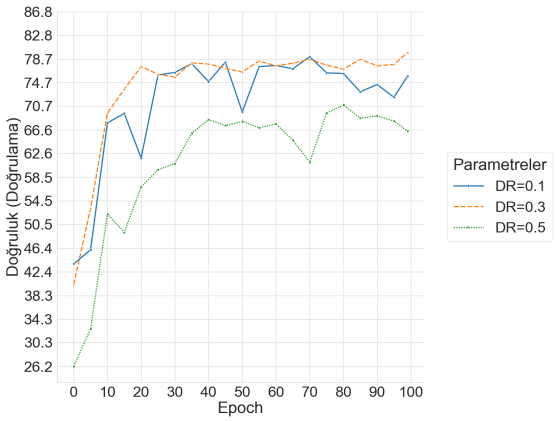
Derin öğrenmede bilirkişiye ihtiyaç duymadan Oto Kodlayıcılar kullanılarak veriden ayırt edici özellikler çıkartılabilmektedir. Bölüm 2'de bahsedilen birkaç farklı Oto Kodlayıcı türlerinden Derin Oto Kodlayıcı ile özellik çıkartılmıştır ve LSTM ağına girdi olarak verilmiştir. Sınıflandırma sonuçları daha sonra, ham veri ve zaman alanındaki özellikler kullanılarak elde edilen sonuçlar ile karşılaştırılmıştır.

4.1.3.1 Oto Kodlayıcı Parametrelerinin Sonuca Etkileri

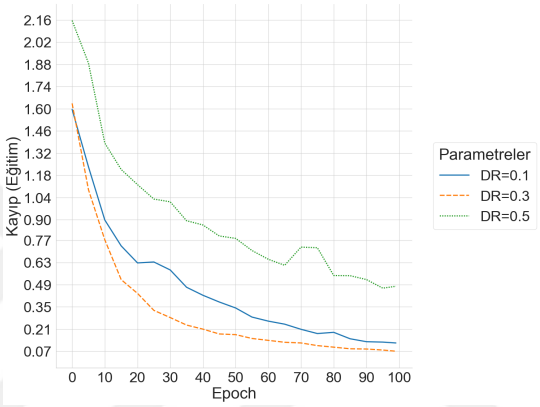
Deneyler giriş (input), şifreleyici (encoder), şifre (code), şifre çözücü (decoder) ve çıkış (output) katmanlarından oluşan Derin Oto Kodlayıcı üzerinde yapılmıştır (Şekil 4.27). 16 saniyelik pencerelerde giriş katmanı ve çıkış katmanı 9600 boyutundadır. Şifre katmanına kadar girdi %25 oranında küçültmekte (7200) ve daha sonra şifre çözücü katmanında girdi tekrardan oluşturulmaya çalışılmaktadır.



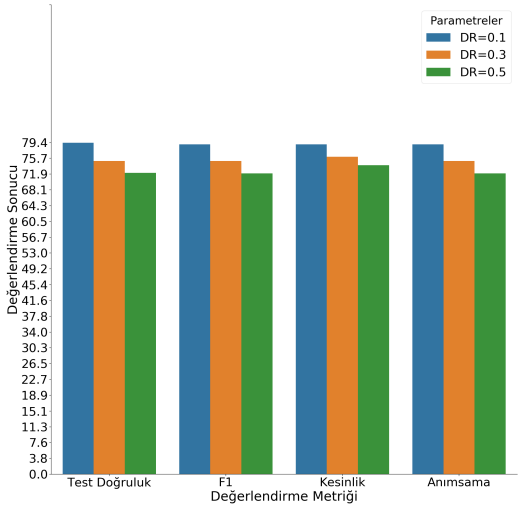
Şekil 4.27 Derin Oto Kodayıcı mimarisi



(a) Doğrulama setinde doğruluk değeri



(b) Doğrulama setinde kayıp değeri



(c) Test seti üzerinde sonuçlar

Şekil 4.28 Derin Oto Kodayıcıda dropout parametresinin LSTM sınıflandırma sonucuna etkileri

Oto Kodlayıcı ağında ezberlemeyi engellemek için giriş katmanından sonra dropout katmanı eklenmiştir. 0.1-0.5 dropout değerleri kullanılarak Oto Kodlayıcıdan elde edilen özellikler kullanılarak LSTM ağında sınıflandırma yapmanın sonuçları Şekil 4.28'de gösterilmiştir.

Şekil 4.28 üzerinde görüldüğü gibi dropout değerini 0.1 kullanmak, test setinde bütün metriklerde daha iyi sonuç üretmiştir. Oto Kodlayıcı kullanılarak çıkartılan özelliklerle en iyi elde edilen sınıflandırma sonucu %79.3 doğruluk, %79 F1, %79 kesinlik ve %79 anımsamadır.

4.1.4 Makine Öğrenmesi Yöntemleri

Derin öğrenmeyle elde edilen sonuçları karşılaştırmak için makine öğrenmesi yöntemleri ile sınıflandırma yapılmıştır. Makine öğrenmesi yöntemleri olarak, bölüm 2'de anlatılan, bu alandaki çalışmalarda en çok tercih edilen Naive Bayes, Karar Ağacı, SVM ve Rasgele Orman kullanılmıştır. Bu yöntemler ile eğitim yapmak ve model oluşturmak için WEKA aracı kullanılmıştır. Bütün yöntemler WEKA'da bulunan varsayılan ayarlar ve parametreler ile uygulanmıştır.

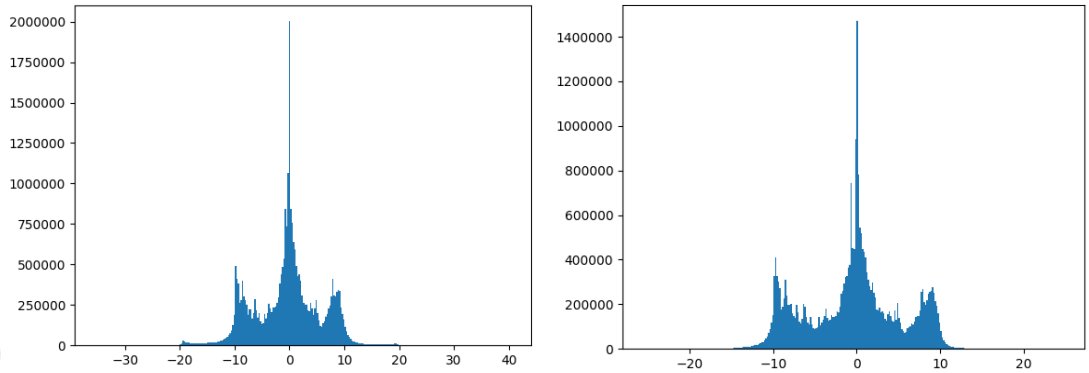
Makine öğrenmesi yöntemlerine pencerelerden zaman alanında özellikler çıkartılarak girdi olarak verilmiştir. Naive Bayes ile yapılan deneyde %56 doğruluk elde edilmiştir. Naive Bayes karşılaştırılmak için çok kullanılan yöntem olsa da aktivite ve seyahat türü tanımada düşük başarı üreten yöntem olmuştur. Karar Ağacı (J48) ile yapılan deneyde %71 doğruluk elde edilmiştir. SVM ile daha iyi sonuç olan %81 doğruluk elde edilmiştir. En iyi sonuç ise diğer çalışmalarda da iyi sonuç üreten Rasgele Orman ile %83.8 doğruluk elde edilmiştir.

4.2 Sınıflandırma Sonucunu Etkileyen Etkenler

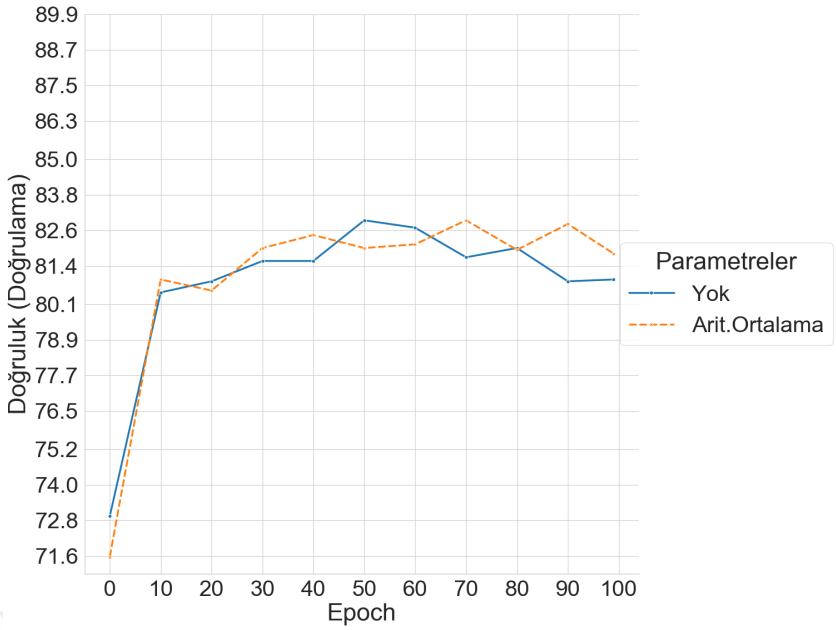
4.2.1 Veri Ön İşleme

Veriler toplanırken veri toplayan kişiden, cihazdan veya sensörlerden kaynaklı hatalar oluşabilmektedir. Bu hatalar veri setine uç değerleri (çok büyük veya çok küçük) eklemektedir. Şekil 4.29'da solda, ön işleme yapılmadan önce ivme ölçer sensörünün x ekseninin çubuk grafiği verilmiştir. Daha sonra veri üzerinde

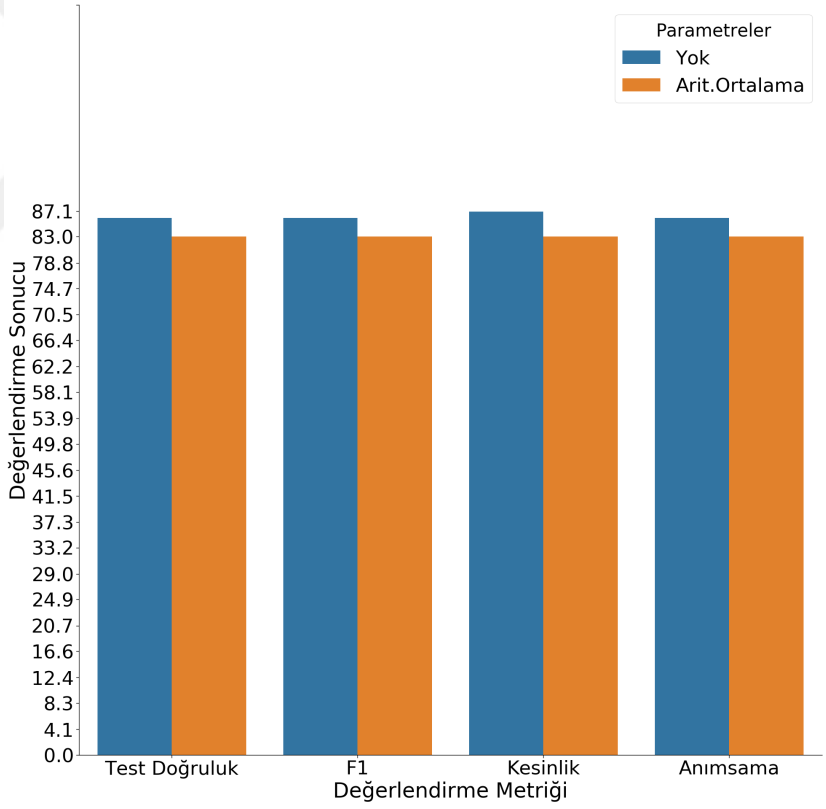
aritmetik ortalama ile Düzleştirme (Arithmetic Mean Smoothing) yapıldıktan sonra, Şekil 4.29'da sağda bazı uç değerlerin (-30, +40 gibi) düzleştiği görülebilmektedir. Düzleştirme için 11 örnekten oluşan pencere boyutu kullanılmıştır.



Şekil 4.29 Eğitim setinde ivme ölçer x ekseni verileri çubuk grafiği (Solda) Ön işleme yapılmadan önce (Sağda) Aritmetik ortalama Düzleştirme yapıldıktan sonra



(a)



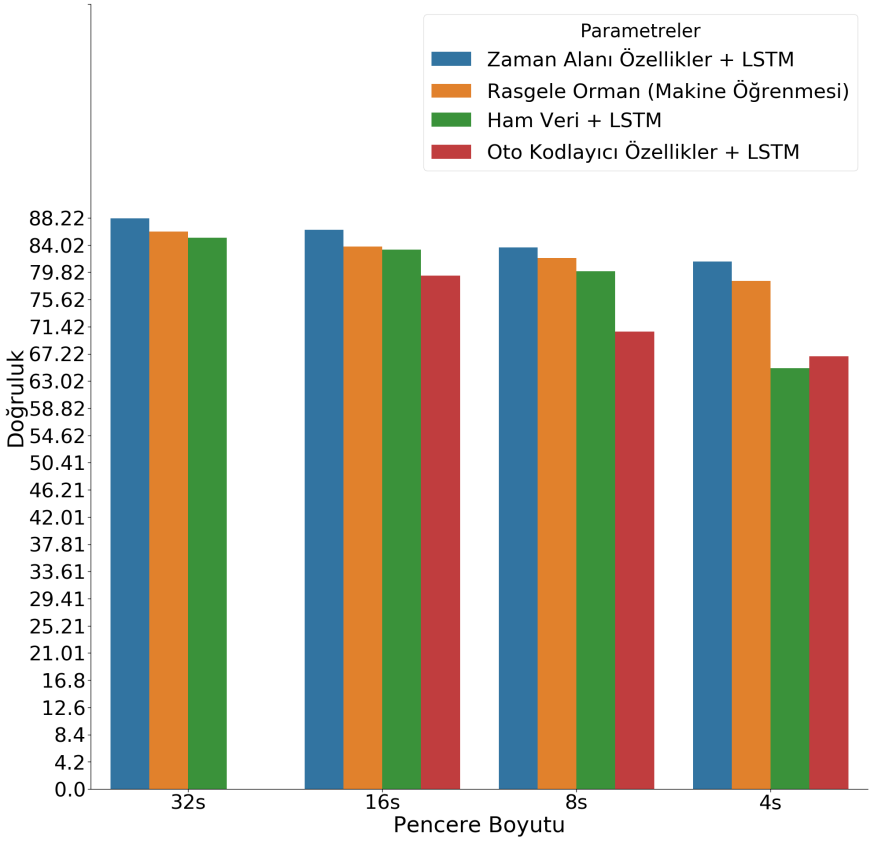
(b)

Şekil 4.30 (a) Doğrulama setinde Düzleştirmenin doğruluğa etkisi (b) Test setinde Düzleştirmenin sonuca etkileri

Şekil 4.30'da görüldüğü gibi, Düzleştirme yapmanın doğruluk setinde doğruluğa çok az etkisi olsa da test setinde sonuçlara olumsuz etki yapmıştır.

4.2.2 Pencere Boyutu

Bütün deneyler 16 saniyelik pencereler ile yapıldıktan ve uygun parametreler belirlendikten sonra 4, 8 ve 32 saniyelik pencereler ile deneyler yapılmıştır. 32 saniyelik pencerede 80 zaman adımı ve 600 LSTM birim sayısı kullanılmıştır. 8 saniyelik pencerede 40 zaman adımı ve 300 LSTM birim sayısı kullanılmıştır. 4 saniyelik pencerede ise 20 zaman adımı ve 300 LSTM birim sayısı kullanılmıştır.



Şekil 4.31 Pencere boyutlarına göre farklı yöntemler ile elde edilen doğruluk sonuçları

32 saniyelik pencere yerine 16 saniyelik pencere kullanıldığında, LSTM ve zaman alanı özellikleriyle %1.7, Rasgele Orman makine öğrenmesi yöntemiyle %2.3, LSTM ve ham veri özellikleriyle %1.9 daha düşük doğruluk elde edilmiştir (Şekil 4.31). Pencere boyutunu 16 saniyeden 32 saniyeye çıkartmak sonuçlar üzerinde olumlu olsa da büyük iyileştirme yapmamıştır.

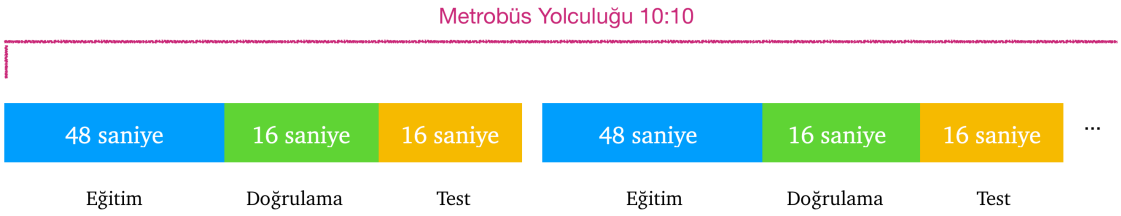
16 saniyelik pencere yerine 8 saniyelik pencere kullanıldığında, LSTM ve zaman alanı özellikleriyle %2.7, Rasgele Orman makine öğrenmesi yöntemiyle %1.8, LSTM ve ham veri özellikleriyle %3.3, LSTM ve Oto Kodlayıcı özellikleriyle %8.6 daha düşük doğruluk elde edilmiştir (Şekil 4.31). Bu deney sonucunda görüldüğü gibi Oto Kodlayıcı özellikleri kullanırken 16 saniyeden küçük pencere boyutu kullanmak sonucu çok olumsuz etkilemektedir.

Pencere boyutu 8 saniyeden 4 saniyeye indirildiğinde ise, LSTM ve zaman alanı özellikleriyle %2.2, Rasgele Orman makine öğrenmesi yöntemiyle %3.5, LSTM ve ham veri özellikleriyle %15, LSTM ve Oto Kodlayıcı özellikleriyle %3.8 daha düşük doğruluk elde edilmiştir. Bu sonuçta görüldüğü gibi 8 saniyelik pencere ile ham veri özellikleri kullanarak %80 doğruluk elde edilirken 4 saniyelik pencere kullanıldığında doğruluk %65'e düşmüştür. Ham veri özellikleri kullanırken 8 saniyeden küçük pencere boyutu kullanmak sonucu çok olumsuz etkilemektedir.

32 saniyelik pencere ile en iyi sonuç, %88.1 doğruluk, elde edilse de 16 saniyelik pencere kullanmak tercih edilmiştir. 32 saniyelik pencereler kullanıldığında ve pencereler %50 örtüştürüldüğünde her 16 saniyede bir tahmin yapılmaktadır. 16 saniyelik pencere kullanıldığında ise, 2 kat daha hızlı, 8 saniyede bir tahmin yapılabilmektedir.

4.2.3 Veriyi Eğitim, Doğrulama, Test Setlerine Ayırma Yöntemi

Veriler bölüm 3.2.5'te anlatıldığı gibi 3 farklı yöntemle eğitim, doğrulama ve test setlerine ayrılmıştır.



Şekil 4.32 Veri setini sırayla 3-1-1 şeklinde setlere ayırmak

İlk denenmiş olan yöntem, pencereleri sırasıyla 3 tanesini eğitim setine, 1 tanesini doğrulama setine, 1 tanesini de test setine ekleyerek setlere ayırmaktır (Şekil

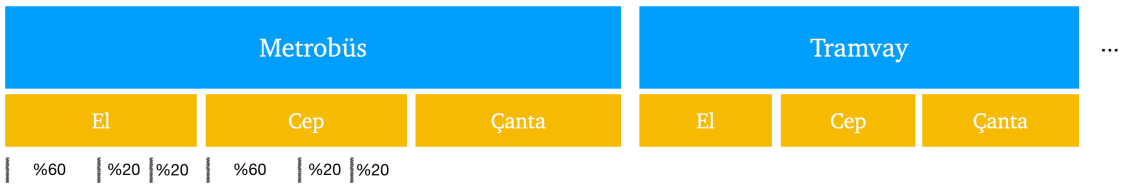
4.32). Bu şekilde veriler %60 eğitim setine, %20 doğrulama setine ve %20 test setine ayrılmış olmaktadır. Bu yöntem kullanıldığında farklı cihaz konumlarında toplanan aktivite verileri de eşit şekilde setlere dağılmış olmaktadır. Bu yöntem kullanılarak setlere ayrılan veri ile Oto Kodlayıcı özellikleri çıkartılarak yapılan LSTM sınıflandırma sonucunda, 16 saniyelik pencere ile, %97 doğruluk elde edilmiştir. Bu yöntemle elde edilen sonuç adil olmamaktadır. Veriler setlere ayrıldığında, aynı yolculuğun 48 saniyesi eğitim setine (3 pencere), devamı olan 16 saniye doğrulama setine, devamı olan 16 saniye de (1 pencere) test setine eklenmektedir. Dolayısıyla aynı yolculuğun art arda gelen ve çok benzer desenler içeren bölümleri bütün setlere de eklenmiş olmaktadır. Bu şekilde eğitim yapıldığında ağ ezberleme yapmaktadır ama doğrulama seti veya test seti ile bu algılanamaz durumdadır. Ağın ezberlemesini doğrulama setinde fark edilemez olmasının sebebi doğrulama setinde de eğitim setindeki verilerin çok benzerinin bulunmasıdır.



Şekil 4.33 Veri setini aktiviteye göre %60-%20-%20 oranında setlere ayırmak

Bir diğer veriyi setlere ayırma yöntemi, Şekil 4.33'te gösterildiği gibi, her aktivitenin ilk %60'lık bölümünü eğitim setine, sonraki %20'lik bölümünü doğrulama setine, sonraki %20'lik bölümünü de test setine eklemektir. Bu yöntemle ayrılan veri setlerinden özellikler çıkartılarak LSTM ile sınıflandırma sonucunda ~%6 daha düşük doğruluk (%80) elde edilmiştir. Bu yöntemle veri setini ayırarak sınıflandırma yapmak adil sonuç vermemektedir. Veriler bu şekilde setlere ayrıldığında aktiviteye ait cihaz konumları dengesiz şekilde dağılmış olmaktadır. Aktivite verisinin ilk %60'lık bölümünde, eğitim setinde, olan cihaz konumları doğrulama setinde ve test setinde eksik olmaktadır. Aynı şekilde test

setindeki cihaz konumları eğitim setinde mevcut değildir. Bu şekilde eğitim setindeki belirli cihaz konumu ile eğitim yapılırken, test sırasında eğitimde hiç görülmemiş cihaz konumu ile karşılaşılmış olmaktadır ve modelin başarısı yanlış ölçülmüş olmaktadır.



Şekil 4.34 Veri setini aktiviteye ve cihaz konumuna göre %60-%20-%20 oranında setlere ayırmak

Veriyi eğitim, doğrulama, test setlerine ayırmak için tercih edilen yöntem ise yolculuk esaslı dosyalar ile ayırmaktır. Her aktiviteye ait klasörler içerisindeki her cihaz konumunun %60'lık bölümü eğitim setine, %20'lik bölümü doğrulama setine, %20'lik bölümü de test setine eklenmiştir. Hesaplamalar dosya boyutuna göre yapılmıştır. Bu şekilde her aktivite doğru oranda setlere ayrılmış olmakta ve aktivitelere ait cihaz konumları doğru oranda bütün setlere dağıtılmış olmaktadır.

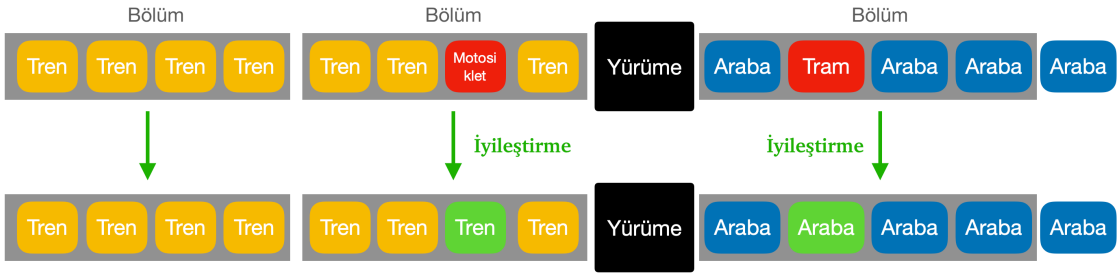
4.3 Sınıflandırma Başarısını İyileştirme

Bu bölümde sınıflandırma aşamasından sonra başarıyı daha iyileştirmek için uygulanan adımlar anlatılmaktadır.

4.3.1 Bölüm Esaslı İyileştirme

Sınıflandırma aşamasından sonra seyahat türleri üzerinde iyileştirilme yapılmıştır. İyileştirme yapmak için yolculuklar 5'er dakikalık bölümlere ayrılmıştır. Kişi seyahat ederken iki seyahat aracı arasında yürüme aktivitesi olacağı için her iki farklı seyahat türü etiketi arasına yürüme aktivitesi etiketi eklenmiştir (Şekil 4.35). Bu sayede her 5 dakikalık yolculuktan sonra veya araçla seyahat ettikten sonra yürüme aktivitesi yapıldığında sonuç üzerinde iyileştirme yapılmış olacaktır. İki yürüme aktivitesi arasında sadece tek bir araçla seyahat edilebileceği düşünülerek, bir bölüm içindeki en çok tahmin edilen sınıf hangisi ise, o bölümdeki diğer tahmin sonuçları buna göre düzeltilmiştir (Şekil 4.35). Üzerinde

iyileştirilme yapılan seyahat türleri şunlardır: *metrobüs, tramvay, hızlı tren, tren, metro, araba, motosiklet, bisiklet.*



Şekil 4.35 Bölüm esaslı iyileştirme

Araçla seyahatlarda bölüm esaslı iyileştirme algoritması uygulamak doğruluk üzerinde %4.7, F1 metriğinde %5, kesinlik metriğinde %4, anımsama metriğinde %5 iyileştirme yapmıştır. İyileştirme sonucunda %91.1 doğruluk elde edilmiştir. İyileştirme algoritması Rasgele Orman makine öğrenmesi yönteminin sonuçları üzerinde uygulandığında %2.7 iyileştirilerek %86.5 doğruluk elde edilmiştir.

Tablo 4.2 Araçla seyahat sınıflarında iyileştirme yapmanın metriklere etkileri

Araçla Seyahat Türü	**F1 İyileştirme**	**Kesinlik İyileştirme**	**Anımsama İyileştirme**
Tramvay	%19	%13	%25
Metro	%15	%17	%11
Araba	%14	%19	%7
Tren	%9	%7	%10
Hızlı Tren	%6	%4	%10
Bisiklet	%5	%3	%7
Metrobüs	%4	%3	%7
Motosiklet	%4	%6	%1

Tablo 4.2'de görüldüğü gibi iyileştirme yapmak en çok sırasıyla tramvay, metro, araba ve tren araç türleri üzerinde olumlu etki yapmıştır.

Tablo 4.3 Aktivite ve seyahat türlerine göre elde edilen sonuç metrikleri (iyileştirmeden önce/iyileştirmeden sonra)

Aktivite/Seyahat Türü	**Doğruluk (%)**	**F1(%)**	**Kesinlik(%)**	**Anımsama(%)**
Metrobüs	91.1/97.9	90/94	88/91	91/98
Tramvay	65.3/90.4	69/88	73/86	65/90
Merdiven İnme	97.0/97.0	90/90	83/83	97/97
Merdiven Çıkma	56.0/56.0	70/72	93/100	56/56
Hızlı Tren	82.1/91.8	76/82	70/74	82/92
Tren	65.0/75.0	75/84	88/95	65/75
Metro	86.8/98.1	78/93	71/88	87/98
Araba	89.3/96.3	83/97	78/97	89/96
Motosiklet	95.5/97.4	93/97	90/96	96/97
Bisiklet	92.4/99.1	94/99	96/99	92/99
Koşma	96.7/96.7	97/97	97/97	97/97
Yürüme	97.2/97.2	94/94	91/91	97/97
Sabit Durma	81.7/80.1	83/85	85/91	82/80
Ortalama	**91.1**	**91**	**91**	**91**

Tablo 4.3'de elde edilen en iyi sonuçlar aktivite ve seyahat türlerinin her biri için ayrı ayrı verilmiştir. Sonuçlar, iyileştirme yapılmadan önce ve iyileştirmeden sonraki oranları olmak üzere karşılaştırmalı verilmiştir.

4.3.2 Tren ve Hızlı Tren Sınıflarını Birleştirme

Sınıflandırma sonucunda en çok karıştırılan seyahat araçlarının hızlı tren, tren ve metro olduğu tespit edilmiştir. Bu üç seyahat türleri raylı sistem üzerinde çalışan seyahat türleridir ve benzer hareket desenleri içerebilmektedir. Özellikle hızlı tren ve tren seyahat türleri belirli rotalarda çok benzer yolculuklar yapabilmektedirler. Bu yüzden tren ve hızlı tren aktiviteleri tek sınıf olarak değerlendirildiğinde, 12 sınıf arasında aktivite ve seyahat türü tanımanın doğruluk başarısı %91.1'den %93.1'e yükseldiği görülmektedir.

Tren ve hızlı tren aktivitelerinin bir kısmı sabit durma aktivitesi olarak tahmin edilmiştir. Sabit durma olarak tahmin edilen bu aktiviteler gerçekte trenin durma anlarında toplanmış ve tren olarak etiketlenmiş veriler olabilmektedir. Bu aktiviteler doğru olarak kabul edildiğinde doğruluk başarısı %93.1'den %94.1'e yükselmektedir.

5
SONUÇ VE ÖNERİLER

Bu bölümde aktivite ve seyahat türü tanımada elde edilen sonuçlar ve sonuca etki eden etkenlerin sebepleri anlatılmıştır.

5.1 Sonuç

Bu tezde LSTM derin öğrenme yöntemi kullanılarak aktivite ve seyahat türü tanıması yapılmıştır. Veriler ivme ölçer, jiroskop ve mıknatıs ölçer sensörlerinden toplanmıştır. Cihaz oryantasyonundan ve konumundan bağımsız 13 aktivite ve seyahat türü birlikte sınıflandırılmıştır. Bu türler *sabit durma, yürüme, koşma, merdiven çıkma, merdiven inme, bisiklet, motosiklet, araba, metro, tramvay, tren, hızlı tren ve metrobüstür*.

LSTM derin öğrenme ağında aktivite ve seyahat türü tanımada uygun parametreleri bulmak için deneyler yapılmıştır. Bu deneyler sonucunda Adam eniyileştirici algoritması, 0.0005 öğrenme hızı, %50 dropout, 600 LSTM birim sayısı, 1 hücreli LSTM, tek yönlü LSTM parametreleri ile en iyi sonuç elde edilmiştir.

5.2 Normalleştirme ve Düzleştirme

Veriyi ön işleme adımlarının sınıflandırma sonucu üzerindeki etkileri araştırılmıştır. Zaman alanında çıkartılan özellikler pencere içinde bölüm esaslı normalleştirilmiştir. Her bölümdeki her eksenin en büyük ve en küçük değerleri hesaplanmıştır ve Normalleştirme yapılmıştır. Zaman alanı özelliklerinde 0+1 veya -1+1 değerleri arası Normalleştirme yapmanın sonuçları arasında çok büyük fark görülmese de ham veriler üzerinde 0+1 değerleri arasına Normalleştirme yapmak %11 daha iyi doğruluk vermiştir. Ham verilerde Normalleştirme işlemi bütün veri setindeki her eksenin en büyük ve en küçük değerleri hesaplandıktan sonra eksenler üzerinde yapılmıştır.

Veriler üzerinde Düzleştirme (Smoothing) yapmanın verideki uç değerleri çıkarttığı gözlemlenmiştir (Şekil 4.29). Ama bunun sonuca olumlu etkisinin olmadığı belirlenmiştir. Düzleştirmenin sonucu olumlu etkilememiş olmasının sebeplerinden birisi eğitim yapılırken uç değerlere duyarlı olmayan MAE kayıp fonksiyonunun kullanılması olabilir. Bir başka sebep olarak veride eğitimi etkileyen çok fazla uç değerin olmadığı sonucu çıkartılabilir.

5.3 Zaman Adımı Sayısı

Ham veriler 20-400 arası zaman adımlarına ayrılarak sınıflandırma deneyleri yapılmıştır. Ham veriler ile eğitim yapıldığında, girdi ne kadar fazla zaman adımına ayrılırsa sonucun da o kadar arttığı gözlemlenmiştir (Şekil 4.25). 200 zaman adımında sonuç doyuma ulaşmıştır ve en iyi doğruluk değerini üretmiştir. 16 saniyelik pencerenin 200 bölüme ayrılması her bölümde 80 milisaniyelik veri olduğu anlamına gelmektedir. 80 milisaniyede 3 sensörden 48 tane değer toplanmaktadır. Bu 48 ham veri özelliği LSTM ağı belleği için yeterli kadar bilgiye sahip bölümleri oluşturduğu düşünülmüştür. Girdi 400 bölüme ayrıldığında ise başarı biraz düşmüştür. Bunun sebebi her bölümde LSTM ağının eğitimi için yeterli kadar özellik, bilgiye sahip olmaması olarak yorumlanmıştır.

Zaman alanında özellikler çıkartıldığında girdiler 10-50 arası zaman adımına ayrılarak deneyler yapılmıştır. Her bölüm için bütün istatistiksel özellikler çıkartıldığı için bölüm sayısı arttıkça verinin boyutu da büyümektedir. Donanım kısıtlamasından dolayı girdiyi daha fazla bölümlere ayırarak deneyler yapılamamıştır. Şekil 4.22'de görüldüğü gibi zaman adımı sayısı arttıkça başarı da artmaktadır. 40 ve 50 zaman adımı arasında çok az doğruluk farkı mevcuttur ve girdi 50 bölüme ayrıldığında en iyi başarıyı üretmiştir. Her bölümde 320 milisaniyede sensörlerden toplanmış her eksen için 16 tane değer mevcuttur (toplam 192). İstatistiksel özellikler çıkartmak için 16 tane değerin uygun, yeterli olduğu ve iyi sonuç ürettiği belirlenmiştir.

5.4 Özellik Çıkartma

LSTM parametreleri seçildikten sonra önce ham veriler ile eğitilerek sınıflandırma sonucu elde edilmiştir. Daha sonra verilerden 27 adet zaman alanında istatistiksel özellikler çıkartılmıştır ve LSTM ile sınıflandırma yapılmıştır. Son olarak Oto Kodlayıcı derin öğrenme yöntemi ile elde edilen özelliklerle LSTM ağı eğitilmiştir ve sınıflandırma yapılmıştır.

Ham veri üzerinde herhangi ayırt edici özellik çıkartmadan veriyi LSTM ağına girdi olarak vererek %83.3 doğruluk elde edilmiştir. Diğer yandan makine öğrenmesi yöntemiyle, istatistiksel özellikler kullanılarak en iyi elde edilen sonuç %83.8'dir. Bu sonuç LSTM ağının girdiyi bölümler halinde alarak ve bellek oluşturarak, belirli ölçüde özellik çıkartabildiği ve istatistiksel özellikler kullanan makine öğrenmesi yöntemleri kadar iyi sınıflandırma yapabildiğini göstermektedir. Bunun sebebi LSTM yapısının zamana bağlı problemlere uygun olması ve girdi bölümlerinden bellek oluşturabilmesi olarak yorumlanabilir.

Oto Kodlayıcı ile ham veriden özellikler çıkartılarak %79.3 doğruluk elde edilmiştir. Bu sonuç ham veri kullanılarak elde edilen sonuçtan %4 daha düşüktür. Oto Kodlayıcının zamana bağlı verilerdeki desenleri, herhangi bellek yapısı olmadığı için, çok iyi çıkartamadığı sonucuna varılabilir. Zaman alanında 27 adet istatistiksel özellik çıkartılarak LSTM ağında kullanıldığında %86.4 doğruluk elde edilmiştir. Sürekli tekrar eden aktivite ve seyahat türü desenleri için istatistiksel özellikleri kullanmanın daha iyi sonuç ürettiği sonucuna varılabilir. Derin öğrenme yöntemi olan LSTM ağı makine öğrenmesi yöntemlerinden daha iyi sonuç üretmiştir.

5.5 Pencere Boyutu

LSTM ve zaman alanı özellikleriyle 16 saniyelik pencerelerle %86.4 doğruluk elde edilmişken 32 saniyelik pencereler ile sınıflandırma yapıldığında %88.1 doğruluk elde edilmiştir. Pencere boyutunu iki katına çıkarmak sonuç üzerinde %1.7 iyileştirme yapmıştır. Pencere boyutunu büyütmek, makale [18] ve [35] içinde belirtildiği gibi doğruluk üzerinde büyük etki yapmamıştır. Bu sonuç 16 saniyelik

pencereden çıkartılan istatistiksel özelliklerin 13 adet aktivite ve seyahat türlerini yüksek başarıyla sınıflandırmak için yeterli olduğunu göstermektedir.

Makine öğrenmesi yöntemleri, ham veri ve Oto Kodlayıcı özellikleriyle 4 saniyelik pencerede %80'den düşük doğruluk elde edilmişken, istatistiksel özellikler ve LSTM ağıyla %81.5 doğruluk elde edilmiştir. 4 saniyelik %50 örtüşmeli pencereler kullanıldığında her 2 saniyede sınıflandırma yapılmaktadır. LSTM derin öğrenme yönteminin ve zaman alanındaki özelliklerin, gerçek zamanlı aktivite ve seyahat türü tanıma uygulamalarında kullanılabilecek düzeyde iyi sonuç verdiğini göstermektedir.

Oto Kodlayıcı özellikleriyle 16 saniyelik pencerede %79.3 doğruluk elde edilse de 8 ve 4 saniyelik pencerelerde sırasıyla %70.7 ve %66.9 doğruluk elde edilmiştir. Bu sonuçlar Derin Oto Kodlayıcı ağının, tekrar eden aktivite verisi desenlerinden özellik çıkartmada istatistiksel özellikler kadar iyi olmadığını göstermektedir. İstatistiksel özelliklerin aktivitedeki tekrarlayan desenleri daha iyi ifade ettiği ortaya konmuştur.

16 saniyelik pencerede LSTM ve ham veri özellikleriyle elde edilen doğruluk ile Rasgele Orman makine öğrenmesi yöntemiyle elde edilen doğruluk arasında %0.5 fark vardır. Herhangi ayırt edici özellik çıkartmadan ham veri kullanarak derin öğrenme yöntemiyle sınıflandırma yapmak, istatistiksel özellikler kullanan makine öğrenmesi yöntemi kadar iyi sonuç verebildiği ortaya konmuştur.

5.6 Karıştırılan Aktiviteler

Şekil 5.1 üzerinde sınıflandırma sonucunun karışıklık matrisi verilmiştir. Bu grafikte, tramvay ve metrobüs seyahat türlerinin belirli miktarda karıştırıldığı görülmektedir. Bu iki seyahat türü belirli yollarda benzer şekilde yolculuk yapmaktadır ve hızları veya yokuşları benzer olabilmektedir. Bu benzer desenler içeren veriler, az miktarda bu iki seyahat türünün karıştırılmasına sebep olmuş olabilir.

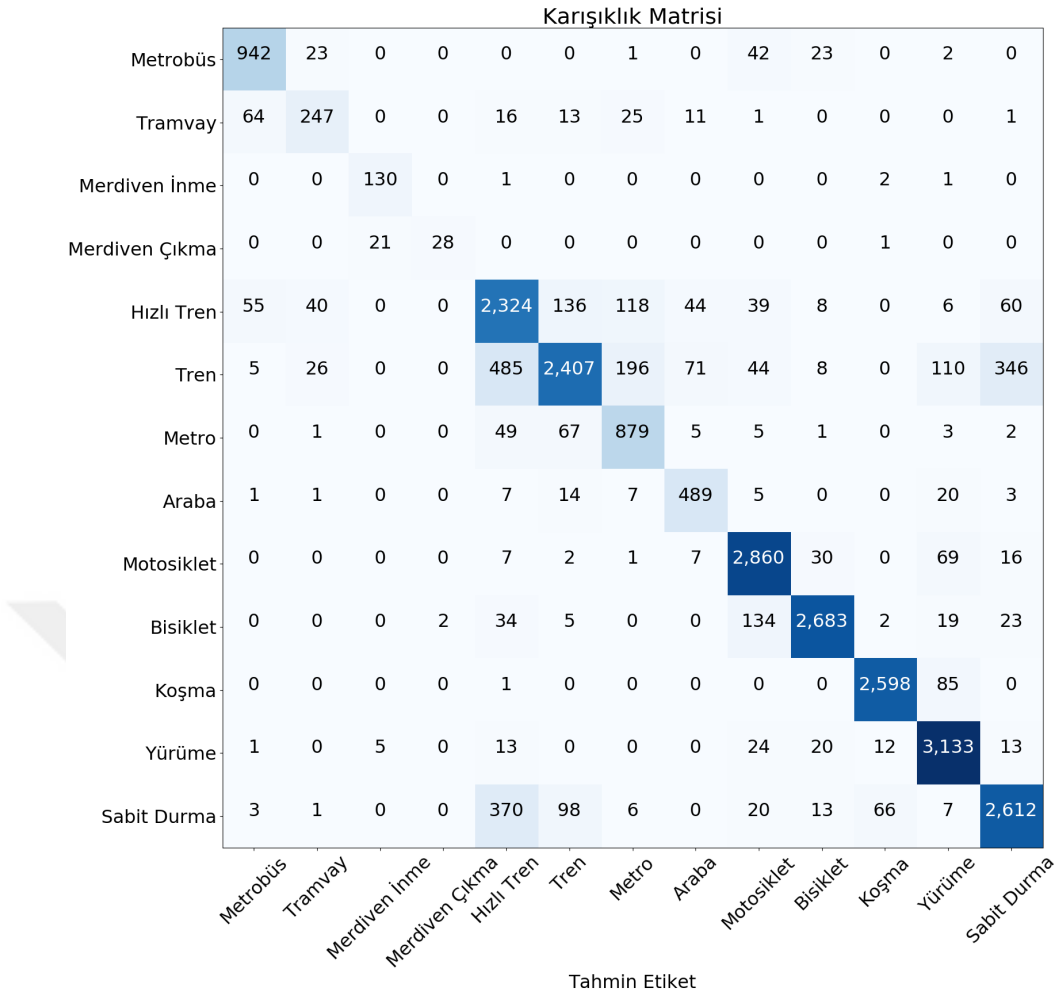
Merdiven çıkma aktivitesi merdiven inme aktivitesiyle karıştırılmıştır. Benzer problem makale [23]'de de bahsedilmiştir. Merdiven inme ve merdiven çıkma

aktiviteleri benzer ayak hareketleriyle yapılmaktadır. Bu benzer desenler karışıklığa neden olmuş olabilir. Diğer neden ise bu iki aktivite verilerinin diğerler verilere kıyasla çok az olması olabilir. Makale [24] eğitim verisinin doğruluğa etkisinin olduğunu belirtmiştir. Merdiven çıkma ve merdiven inme aktivitelerinin eğitim verisinin çok az olması bu aktivitelerin iyi başarıyla sınıflandırılmasına engel olmuş olabilir.

Çok karıştırılan seyahat türleri ise hızlı tren, tren ve metro'dur. Bunlardan en çok tren ve hızlı tren seyahat türleri birbiriyle karıştırılmıştır. Tren ve hızlı tren benzer yollar üzerinde seyahat etmektedir. Hızlı tren uzun yolda trenden daha hızlı seyahat etse de bazı yerlerde bu iki aracın hızları ve yolculukları benzer olabilmektedir. Metro ise hızlı tren ve tren gibi raylar üzerinde seyahat gerçekleştiren toplu taşıma aracıdır. Bu üç raylı sistem seyahat araçlarının karıştırılma sebebi belirli yerlerde benzer yolculuklar yapılabilmesidir ve beklenen karışıklıklardan biridir.

Motosiklet seyahat türünün yürüme ile az oranda karıştırıldığı görülmektedir. Motosikletteki kişinin ayak hareketleri yürüyen kişininkiyle aynı olmasa da trafikte yavaş hareket eden motosikletin desenleri yürüme hareketi ile karıştırılmış olabilir. Diğer yandan koşma hareketinin yürüme aktivitesiyle karıştırılması ise doğaldır. Bisiklet sürme motosiklet sürme ile karıştırılmıştır. Bu iki binilen araç birbirine benzer olup, kişinin oturma şekli de benzemektedir. Bisiklette pedal çevrilse de bisikletin hareketi motosiklet hareketine benzeyebilmektedir.

Bazı araç türleri sabit durma aktivitesi olarak tahmin edilmiştir. Buna sebep olabilecek nedenlerden birisi araçlarda seyahat ederken durma anlarının da seyahat türü olarak etiketlenmesidir. Dolayısıyla bu sabit durma olarak tahmin edilen araçlar gerçekte de sabit duran araçları ifade ediyor olabilir.



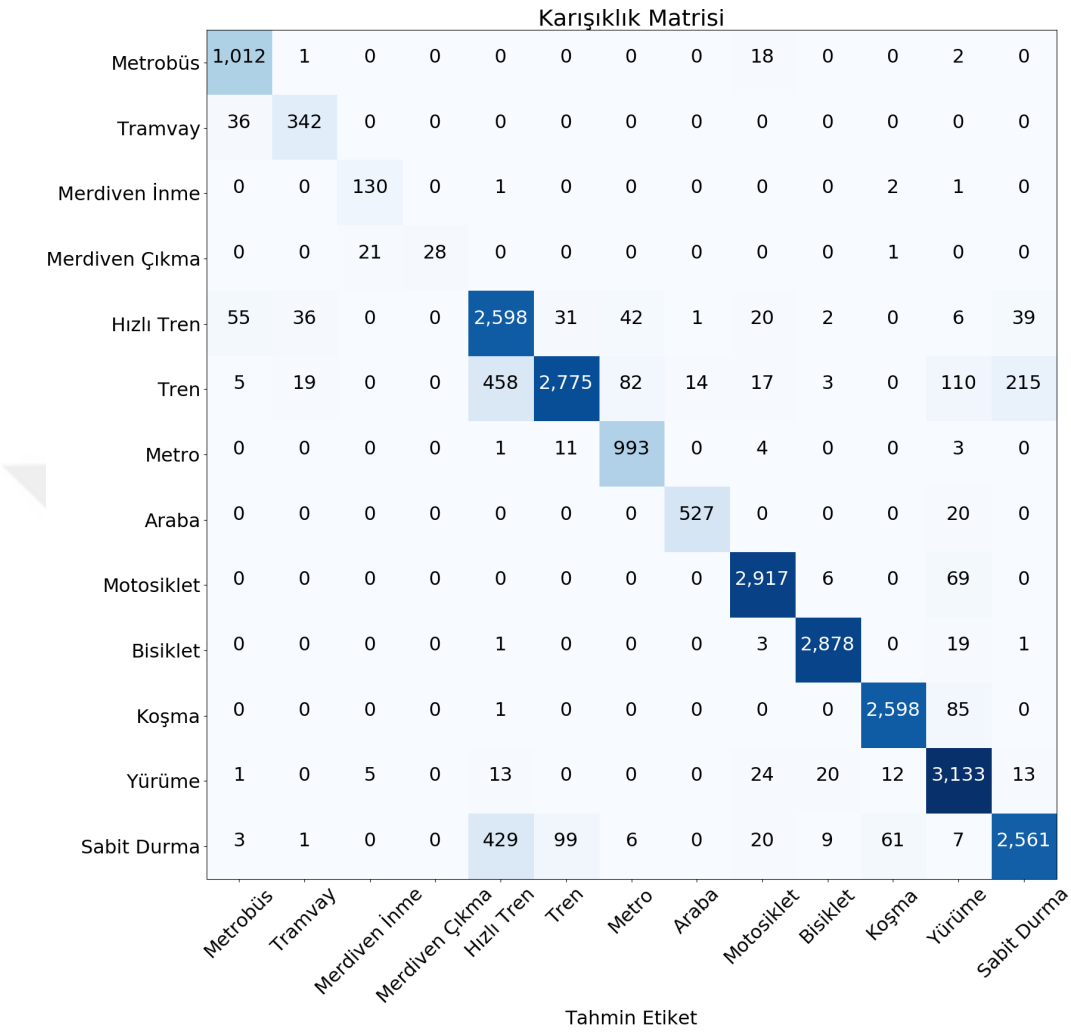
Şekil 5.1 LSTM ve zaman alanı özellikleriyle elde edilen sınıflandırma sonucunun karışıklık matrisi

Çalışma [41] araba ve bisiklet araçlarının birbiriyle karıştırıldığını yazsa da yapılan deneyde bu iki araç birbiriyle karıştırılmamıştır.

5.7 İyileştirme

Sınıflandırma yapıldıktan sonra 5 dakikalık araçla yolculuklar üzerinde iyileştirme yapılmıştır ve doğruluk %4.7, F1 %5, kesinlik %4 ve anımsama %5 iyileştirilmiştir. Araçla seyahat türüne binerken ve araçtan inerken yürüme aktivitesi yapılmaktadır. İyileştirme algoritması yürüme aktivitesine kadar olan araçla seyahat üzerinde iyileştirme yapmaktadır. İyileştirme gecikmesini azaltmak için, yürüme aktivitesine ulaşıncaya kadar yolculuklar 5 dakikalık bölümlere ayrılarak

iyileştirme uygulanmıştır. İyileştirme sonucunda elde edilen karışıklık matrisi Şekil 5.2'de verilmiştir.



Şekil 5.2 İyileştirme adımından sonra sınıflandırma sonucunun karışıklık matrisi

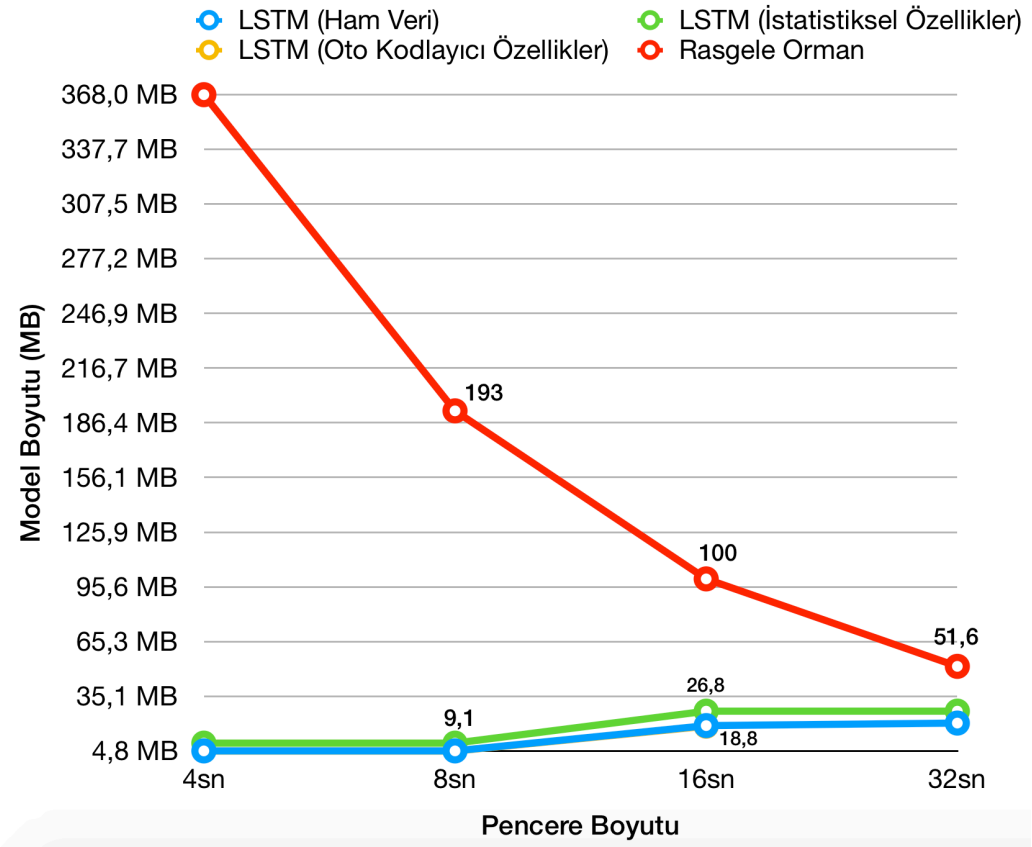
İyileştirme adımından sonra tramvay seyahat türünde F1 metriği %19 iyileştirilmiştir. Birbiriyle karıştırılan metro seyahat türünde %15 F1, tren seyahat türünde %9 F1 metriği iyileştirilmiştir. Motosiklet ile karıştırılan bisiklet aracında ise iyileştirme sonucunda neredeyse karıştırma olmamıştır. Sabit durma olarak tahmin edilen araç türlerinin sayısı azalsa da belirli ölçüde hala karıştırılmaktadır.

Şekil 5.2'de sabit durma aktivitesinin araç türleri olarak etiketlendiği görülmektedir. Bunun sebebi kişinin sabit duran telefonu bir anda eline alırken

oluşan hareket desenlerinin araçla seyahat desenlerine benzetilmesi olarak yorumlanmıştır.

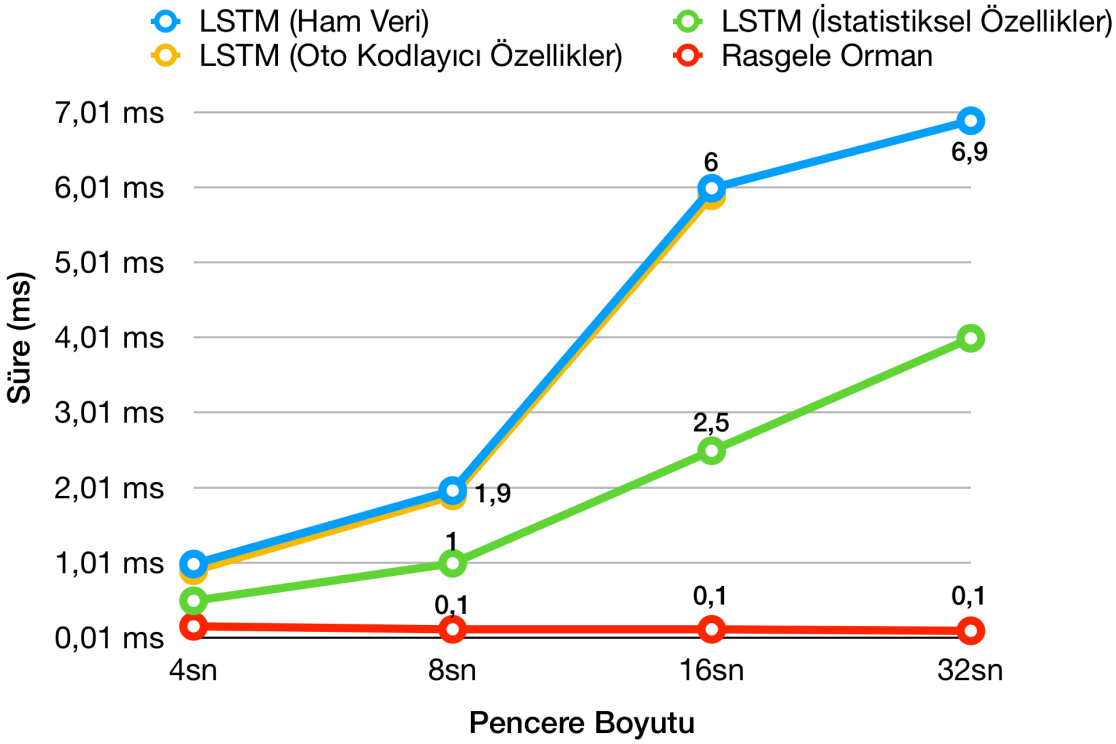
5.8 Model Boyutu ve Sınıflandırma Süresi

LSTM ve Rasgele Orman sınıflandırma modellerinin boyutları Şekil 5.3'te verilmiştir. 4 ve 8 saniyelik pencereleri kullanan LSTM modelinin boyutu 4.8 ve 9.1 megabayttır. 16 saniyelik pencereler ile ham veri kullanan LSTM modeli 18.8 megabayt iken istatistiksel özellikler kullanan LSTM model boyutu 26.8 megabayta çıkmıştır. Diğer yandan Rasgele Orman makine öğrenmesi yöntemi modelinde pencere boyutu arttıkça model boyutu küçüldüğü görülmektedir. 4 saniyelik pencere kullanıldığında 368 megabayt olan model boyutu 16 saniyelik pencere kullanıldığında 100 megabayta düşmüştür. İstatistiksel özellikler kullanan LSTM modeli boyutu Rasgele Orman modelinden 4 kat daha küçüktür ve akıllı telefonlarda kullanılmaya uygundur.



Şekil 5.3 Sınıflandırma Model Boyutu

Modellerin sınıflandırma süreleri karşılaştırmalı olarak Şekil 5.4'te gösterilmiştir.



Şekil 5.4 Modelin Tek Örneği Sınıflandırma Süresi

Rasgele Orman sınıflandırma modeli, pencere boyutundan bağımsız her örneği 0.1 milisaniyede sınıflandırmıştır. LSTM modelleri arasında ise istatistiksel özellikler kullanan model diğerlerinden daha hızlı sınıflandırma yapmıştır. Ham veri ve Oto Kodlayıcı özellikleri kullanan LSTM modeli 16 saniyelik veri örneğini 6 milisaniyede sınıflandırmıştır. Oto Kodlayıcının veriden özellik çıkartması 0.7 milisaniye sürmektedir ve toplam süre 6.7 milisaniye etmektedir. 16 saniyelik veri örneğinden 27 adet istatistiksel özellik çıkartmak 810 milisaniye sürmektedir. Bu istatistiksel özellikler kullanılarak 16 saniyelik veri örneğini sınıflandırmak 2.5 milisaniye sürmektedir.

Literatürde bu tez projesinde kullanılan veri setin daha büyüğünü kullanan çalışma [18] ivme ölçer, jiroskop ve mıknatıs ölçer ile 5 adet aktiviteyi %92.5 doğrulukla sınıflandırmıştır. Çalışma [18] seyahat türlerini tek sınıf olarak değerlendirmiştir. Bu projede ise 12 aktivite ve seyahat türü sınıflandırmada %93.1 doğruluk başarısına ulaşılmıştır. Diğer çalışmalarda bu projede kullanılan veri setinden farklı veri setleri kullanılmıştır ve sonuçların birebir karşılaştırılması

mümkün değildir. Derin öğrenme kullanan makale [17] 7 adet aktivite ve seyahat türünü 10 saniyelik pencereler kullanarak %94.4 doğrulukla sınıflandırmıştır. Bu tezde LSTM derin öğrenme yöntemiyle 8 saniyelik pencereler kullanılarak 12 adet sınıf %90.1 doğrulukla sınıflandırılmıştır. 16 saniyelik pencereler kullanılarak %93.1 doğruluk elde edilmiştir. Çalışma [16] içinde uçak ve feribot araçlarının bulunduğu 10 adet seyahat türünü sınıflandırmıştır. GPS ve GIS kullanan bu çalışma %91.6 doğruluk elde etmiştir. Seyahat türü tanıma yolculuk esaslı yapılmıştır ve sabit pencere boyutu yoktur. Bu tez projesinde ise 13 aktivite ve seyahat türü, çok enerji tüketen GPS ve GIS kullanmadan, %91.1 doğruluk ile sınıflandırılmıştır. Makale [16]'dan farklı olarak bu projede aktivite ve seyahat türleri tanıma hem cihaz oryantasyonundan hem de cihazın kişinin üzerindeki konumundan bağımsız yapılmıştır. Aktivite tanıma üzerinde çalışan makale [2] ise 8 adet aktiviteyi %95 doğrulukla tanımlayabilmiştir.

Bu tez projesinde 13 aktivite ve seyahat türü birlikte değerlendirilmiş ve %91.1 doğrulukla sınıflandırılmıştır. Hızlı tren ve tren raylı araç türleri birleştirilerek başarı %93.1'e yükseltilmiştir. Sınıflandırma ivme ölçer, jiroskop ve mıknatıs ölçer sensör verileriyle, cihaz konumundan ve oryantasyonundan bağımsız yapılmıştır. Derin öğrenme yöntemi olan LSTM ağının ham verilerle, istatistiksel özelliklerle ve Oto Kodlayıcı özellikleriyle nasıl sonuçlar ürettiği deneyler ile bulunmuştur. Derin öğrenme ile elde edilen sonuçlar makine öğrenmesi yöntemleriyle elde edilen sonuçlar ile karşılaştırılmış ve derin öğrenmenin daha iyi sonuç ürettiği ortaya konmuştur. Zaman alanında çıkartılan ayırt edici özellikler ve LSTM ağında sonucu etkileyen önemli hiperparametreler listelenmiştir. Kullanılan pencere boyutları, model boyutları, modelin sınıflandırma süresi karşılaştırmalı olarak verilmiştir. Araçlı seyahat türleri üzerinde uygulanan iyileştirme algoritmasının sonucu olumlu etkilediği ortaya konmuştur.

Kaynakça

[1] M. V. Albert, S. Toledo, M. Shapiro and K. Kording, "Using mobile phones for activity recognition in Parkinson's patients," *Frontiers in Neurology,* vol. 3, no. 3, p. 158, 2012.

[2] M. Á. Á. d. l. Concepción, L. M. S. Morillo, J. A. Á. García and L. González-Abril, "Mobile activity recognition and fall detection system for elderly people using Ameva algorithm," *Pervasive and Mobile Computing,* vol. 34, no. C, pp. 3-13, 2017.

[3] S. Yan, Y. Liao, X. Feng and Y. Liu, "Real time activity recognition on streaming sensor data for smart environments," in *2016 International Conference on Progress in Informatics and Computing (PIC)*, Shanghai, 2016.

[4] J. Kooistra, "Newzoo's 2018 Global Mobile Market Report: Insights into the World's 3 Billion Smartphone Users," Newzoo, 2018.

[5] M. Meeker, "Internet Trends Report 2018," KLEINER PERKINS, 2018.

[6] Y. Wurmser, "Wearables 2019 Advanced Wearables Pick Up Pace as Fitness Trackers Slow," eMarketer, 2019.

[7] P. Siirtola and J. Röning, "Recognizing human activities user-independently on smartphones based on accelerometer data," *International Journal of Interactive Multimedia and Artificial Intelligence,* vol. 1, no. 5, pp. 38-45, 2012.

[8] Y. E. Ustev, O. D. Incel and C. Ersoy, "User, device and orientation independent human activity recognition on mobile phones: challenges and a proposal," in *UbiComp '13 Adjunct*, Zurich, 2013.

[9] M. Borazio and K. V. Laerhoven, "Using time use with mobile sensor data: a road to practical mobile activity recognition?," in *Proceedings of the 12th International Conference on Mobile and Ubiquitous Multimedia*, Luleå, 2013.

[10] D. Riboni and C. Bettini, "COSAR: hybrid reasoning for context-aware activity recognition," *Personal and Ubiquitous Computing,* vol. 15, no. 3, pp. 271-289, 2011.

[11] Y.-S. Lee and S.-B. Cho, "Activity Recognition Using Hierarchical Hidden Markov Models on a Smartphone with 3D Accelerometer," in *International Conference on Hybrid Artificial Intelligence Systems*, Wroclaw, 2011.

[12] N. Maisonneuve, M. Stevens and B. Ochab, "Participatory noise pollution monitoring using mobile phones," *Information Polity,* vol. 15, no. 1, pp. 51-71, 2010.

[13] O. Lorintiu and A. Vassilev, "Transportation mode recognition based on smartphone embedded sensors for carbon footprint estimation," in *International Conference on Intelligent Transportation Systems (ITSC)*, Rio de Janeiro, 2016.

[14] R. Bhoraskar, N. Vankadhara, B. Raman and P. Kulkarni, "Wolverine: Traffic and road condition estimation using smartphone sensors," in *International Conference on Communication Systems and Networks*, Bangalore, 2012.

[15] L. Stenneth, O. Wolfson, P. S. Yu and B. Xu, "Transportation Mode Detection using Mobile Phones and GIS Information," in *ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems*, Chicago, 2011.

[16] F. Biljecki, H. Ledoux and P. v. Oosterom, "Transportation mode-based segmentation and classification of movement trajectories," *International Journal of Geographical Information Science,* vol. 27, no. 2, pp. 385-407, 2013.

[17] X. Liang and G. Wang, "A Convolutional Neural Network for Transportation Mode Detection Based on Smartphone Platform," in *2017 IEEE 14th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, Orlando, 2017.

[18] M.-C. Yu, T. Yu, S.-C. Wang, C.-J. Lin and E. Y. Chang, "Big data small footprint: the design of a low-power classifier for detecting transportation modes," *Proceedings of the VLDB Endowment,* vol. 7, no. 13, pp. 1429-1440, 2014.

[19] A. Henpraserttae, S. Thiemjarus and S. Marukatat, "Accurate Activity Recognition Using a Mobile Phone Regardless of Device Orientation and Location," in *International Conference on Body Sensor Networks*, Dallas, 2011.

[20] S. Dernbach, B. Das, N. C. Krishnan, B. L. Thomas and D. J. Cook, "Simple and Complex Activity Recognition through Smart Phones," in *International Conference on Intelligent Environments*, Guanajuato, 2012.

[21] J. Lee and J. Kim, "Energy-Efficient Real-Time Human Activity Recognition on Smart Mobile Devices," *Mobile Information Systems,* vol. 2016, 2016.

[22] Z. Yan, V. Subbaraju, D. Chakraborty, A. Misra and K. Aberer, "Energy-Efficient Continuous Activity Recognition on Mobile Phones: An Activity-

Adaptive Approach," in *16th International Symposium on Wearable Computers*, Newcastle, 2012.

[23] J. R. Kwapisz, G. M. Weiss and S. A. Moore, "Activity recognition using cell phone accelerometers," *ACM SIGKDD Explorations Newsletter,* vol. 12, no. 2, pp. 74-82, 2010.

[24] Z. Zhao, Y. Chen, J. Liu, Z. Shen and M. Liu, "Cross-people mobile-phone based activity recognition," in *IJCAI'11 Proceedings of the Twenty-Second İnternational Joint Conference on Artificial Intelligence*, Barcelona, 2011.

[25] Q. V. Vo, M. T. Hoang and D. Choi, "Personalization in Mobile Activity Recognition System Using K-Medoids Clustering Algorithm," *International Journal of Distributed Sensor Networks,* vol. 9, no. 7, 2013.

[26] D. Anguita, A. Ghio, L. Oneto, X. Parra and J. L. Reyes-Ortiz, "Human Activity Recognition on Smartphones Using a Multiclass Hardware-Friendly Support Vector Machine," in *IWAAL 2012: Ambient Assisted Living and Home Care*, Vitoria-Gasteiz, 2012.

[27] D. Acharjee, A. Mukherjee, J. K. Mandal and N. Mukherjee, "Activity recognition system using inbuilt sensors of smart mobile phone and minimizing feature vectors," *Microsystem Technologies,* vol. 22, no. 11, p. 2715–2722, 2016.

[28] H. Martín, A. M. Bernardos, J. Iglesias and J. R. Casar, "Activity logging using lightweight classification techniques in mobile devices," *Personal and Ubiquitous Computing,* vol. 17, no. 4, pp. 675-695, 2013.

[29] W. Jiang and Z. Yin, "Human Activity Recognition Using Wearable Sensors by Deep Convolutional Neural Networks," in *Proceedings of the 23rd ACM international conference on Multimedia*, Brisbane, 2015.

[30] D. Ravi, C. Wong, B. Lo and G.-Z. Yang, "Deep learning for human activity recognition: A resource efficient implementation on low-power devices," in *IEEE 13th International Conference on Wearable and Implantable Body Sensor Networks (BSN)*, San Francisco, 2016.

[31] V. Radu, N. D. Lane, S. Bhattacharya, C. Mascolo, M. K. Marina and F. Kawsar, "Towards multimodal deep learning for activity recognition on mobile devices," in *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct*, Heidelberg, 2016.

[32] Y. Chen, K. Zhong, J. Zhang, Q. Sun and X. Zhao, "LSTM Networks for Mobile Human Activity Recognition," in *Proceedings of the 2016 International Conference on Artificial Intelligence: Technologies and Applications*, Bangkok, 2016.

[33] G. Xiao, Z. Juan and J. Gao, "Travel Mode Detection Based on Neural Networks and Particle Swarm Optimization," *Information,* vol. 6, no. 3, pp. 522-535, 2015.

[34] G. Xiao, Z. Juan and C. Zhang, "Travel mode detection based on GPS track data and Bayesian networks," *Computers, Environment and Urban Systems,* vol. 54, pp. 14-22, 2015.

[35] S.-H. Fang, Y.-X. Fei, Z. Xu and Y. Tsao, "Learning Transportation Modes From Smartphone Sensors Based on Deep Neural Network," *IEEE Sensors Journal,* vol. 17, no. 18, pp. 6111-6118, 2017.

[36] S.-H. Fang, H.-H. Liao, Y.-X. Fei, K.-H. Chen, J.-W. Huang, Y.-D. Lu and Y. Tsao, "Transportation Modes Classification Using Sensors on Smartphones," *Sensors,* vol. 16, no. 8, p. 1324, 2016.

[37] X. Su, H. Caceres, H. Tong and Q. He, "Online Travel Mode Identification Using Smartphones With Battery Saving Considerations," *IEEE Transactions on Intelligent Transportation Systems,* vol. 17, no. 10, pp. 2921-2934, 2016.

[38] M. A. Guvensan, B. Dusun, B. Can and H. I. Turkmen, "A Novel Segment-Based Approach for Improving Classification Performance of Transport Mode Detection," *Sensors,* vol. 18, no. 1, p. 87, 2018.

[39] L. Bedogni, M. D. Felice and L. Bononi, "Context-aware Android applications through transportation mode detection techniques," *Wireless Communications And Mobile Computing,* vol. 16, no. 16, pp. 2523-2541, 2016.

[40] X. Zhou, W. Yu and W. C.Sullivan, "Making pervasive sensing possible: Effective travel mode sensing based on smartphones," *Computers, Environment and Urban Systems,* vol. 58, pp. 52-59, 2016.

[41] C. A. d. M. S. Quintella, L. C. V. Andrade and C. A. V. Campos, "Detecting the transportation mode for context-aware systems using smartphones," in *International Conference on Intelligent Transportation Systems (ITSC)*, Rio de Janeiro, 2016.

[42] P. Nitsche, P. Widhalm, S. Breuss, N. Brändle and P. Maurer, "Supporting large-scale travel surveys with smartphones – A practical approach," *Transportation Research Part C: Emerging Technologies,* vol. 43, no. 2, pp. 212-221, 2014.

[43] A. Bloch, R. Erdin, S. Meyer, T. Keller and A. d. Spindler, "Battery-Efficient Transportation Mode Detection on Mobile Devices," in *International Conference on Mobile Data Management*, Pittsburgh, 2015.

[44] M. A. Shafique and E. Hato, "Travel Mode Detection with Varying Smartphone Data Collection Frequencies," *Sensors,* vol. 16, no. 5, p. 716, 2016.

[45] D. Shin, D. Aliaga, B. Tunçer, S. M. Arisona, S. Kim, D. Zünd and G. Schmitt, "Urban sensing: Using smartphones for transportation mode classification," *Computers, Environment and Urban Systems,* vol. 53, pp. 76-86, 2015.

[46] S. Wang, C. Chen and J. Ma, "Accelerometer Based Transportation Mode Recognition on Mobile Phones," in *2010 Asia-Pacific Conference on Wearable Computing Systems*, Shenzhen, 2010.

[47] A. Jahangiri and H. A. Rakha, "Applying Machine Learning Techniques to Transportation Mode Recognition Using Mobile Phone Sensor Data," *IEEE Transactions on Intelligent Transportation Systems,* vol. 16, no. 5, pp. 2406-2417, 2015.

[48] T. H. Vu, L. Dung and J.-C. Wang, "Transportation Mode Detection on Mobile Devices Using Recurrent Nets," in *Proceedings of the 24th ACM international conference on Multimedia*, Amsterdam, 2016.

[49] S. Zhang, Z. Wei, J. Nie, L. Huang, S. Wang and Z. Li, "A Review on Human Activity Recognition Using Vision-Based Method," *Journal of Healthcare Engineering,* vol. 2017, 2017.

[50] R.-S. Huang and B.-C. Chien, "Activity recognition on multi-sensor data streams using distinguishing sequential patterns," in *JSAI2013*, Toyama, 2013.

[51] A. C. Prelipcean, G. Gidófalvi and Y. O. Susilo, "Measures of transport mode segmentation of trajectories," *International Journal of Geographical Information Science,* vol. 30, no. 9, pp. 1-22, 2016.

[52] A. C. Prelipcean, G. Gidófalvi and Y. O. Susilo, "Transportation mode detection – an in-depth review of applicability and reliability," *Transport Reviews,* vol. 37, no. 4, pp. 442-464, 2017.

[53] C. A. Ronao and S.-B. Cho, "Evaluation of Deep Convolutional Neural Network Architectures for Human Activity Recognition with Smartphone Sensors," in *Proceedings of the Korean Information Science Society*, 2015.

[54] F. J. Ordóñez and D. Roggen, "Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition," *Sensors,* vol. 16, no. 1, p. 115, 2016.

[55] H. Xia, Y. Qiao, J. Jian and Y. Chang, "Using Smart Phone Sensors to Detect Transportation Modes," *Sensors,* vol. 14, no. 11, pp. 20843-20865, 2014.

[56] A. D. Ignatov and V. V. Strijov, "Human activity recognition using quasiperiodic time series collected from a single tri-axial accelerometer," *Multimedia Tools and Applications,* vol. 75, no. 12, p. 7257–7270, 2016.

[57] T. Sztyler, H. Stuckenschmidt and W. Petrich, "Position-aware activity recognition with wearable devices," *Pervasive and Mobile Computing,* vol. 38, no. 2, pp. 281-295, 2017.

[58] M. A. Ayu, S. A. Ismail, A. F. A. Matin and T. Mantoro, "A Comparison Study of Classifier Algorithms for Mobile-phone's Accelerometer Based Activity Recognition," *Procedia Engineering,* vol. 41, pp. 224-229, 2012.

[59] T. S. Prento, H. Blunck, M. B. Kjærgaard and A. Stisen, "Towards Indoor Transportation Mode Detection Using Mobile Sensing," in *International Conference on Mobile Computing, Applications, and Services*, Berlin, 2015.

[60] G. Singla, D. J. Cook and M. Schmitter-Edgecombe, "Recognizing independent and joint activities among multiple residents in smart environments," *Journal of Ambient Intelligence and Humanized Computing,* vol. 1, no. 1, p. 57–63, 2010.

[61] Q. Ni, A. B. G. Hernando and I. P. D. l. Cruz, "The Elderly's Independent Living in Smart Homes: A Characterization of Activities and Sensing Infrastructure Survey to Facilitate Services Development," *Sensors,* vol. 15, no. 5, pp. 11312-11362, 2015.

[62] E. Casilari, R. Luque and M.-J. Morón, "Analysis of Android Device-Based Solutions for Fall Detection," *Sensors,* vol. 18, no. 8, pp. 17827-17894, 2015.

[63] J. B. Gomes, S. Krishnaswamy, M. M. Gaber, P. A. Sousa and E. Menasalvas, "MARS: A Personalised Mobile Activity Recognition System," in *IEEE 13th International Conference on Mobile Data Management*, Bengaluru, 2012.

[64] B. X. Wang and N. Japkowicz, "Boosting support vector machines for imbalanced data sets," *Knowledge and Information Systems,* vol. 25, no. 1, pp. 1-20, 2010.

[65] H. Minh, "How to Handle Imbalanced Data in Classification Problems," 11 10 2018. [Online]. Available: https://medium.com/james-blogs/handling-imbalanced-data-in-classification-problems-7de598c1059f. [Accessed 10 02 2019].

[66] G. Wu and E. Y. Chang, "KBA: kernel boundary alignment considering imbalanced data distribution," *IEEE Transactions on Knowledge and Data Engineering,* vol. 17, no. 6, pp. 786-795, 2005.

[67] N. V. Chawla, K. W. Bowyer, L. O. Hall and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," *Journal of Artificial Intelligence Research,* vol. 16, p. 321–357, 2002.

[68] G. Welch and G. Bishop, "An Introduction to the Kalman Filter," Chapel Hill, 1997.

[69] O. Banos, J.-M. Galvez, M. Damas, H. Pomares and I. Rojas, "Window Size Impact in Human Activity Recognition," *Sensors,* vol. 14, no. 4, pp. 6474-6499, 2014.

[70] H. Peng, F. Long and C. Ding, "Feature Selection Based on Mutual Information: Criteria of Max-Dependency, Max-Relevance, and Min-Redundancy," *IEEE Transactions on Pattern Analysis & Machine Intelligence,* vol. 27, pp. 1226-1238, 2005.

[71] M. H. Aghdam, N. Ghasem-Aghaee and M. E. Basiri, "Text feature selection using ant colony optimization," *Expert Systems with Applications,* vol. 3, no. 2, pp. 6843-6853, 2009.

[72] "Autoencoders," Stanford University, [Online]. Available: http://ufldl.stanford.edu/tutorial/unsupervised/Autoencoders/. [Accessed 16 02 2019].

[73] S. Bhattacharya and N. D. Lane, "From smart to deep: Robust activity recognition on smartwatches using deep learning," in *IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*, Sydney, 2016.

[74] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science,* vol. 313, no. 5786, pp. 504-507, 2006.

[75] N. Hubens, "Deep inside: Autoencoders," Towards Data Science, 25 02 2018. [Online]. Available: https://towardsdatascience.com/deep-inside-autoencoders-7e41f319999f. [Accessed 16 02 2019].

[76] J. Jordan, "Introduction to autoencoders.," 19 03 2018. [Online]. Available: https://www.jeremyjordan.me/autoencoders/. [Accessed 16 02 2019].

[77] A. Dertat, "Applied Deep Learning - Part 3: Autoencoders," Towards Data Science, 03 10 2017. [Online]. Available: https://towardsdatascience.com/applied-deep-learning-part-3-autoencoders-1c083af4d798. [Accessed 16 02 2019].

[78] M. Deshpande, "All About Autoencoders," 30 10 2017. [Online]. Available: https://pythonmachinelearning.pro/all-about-autoencoders/. [Accessed 19 02 2019].

[79] F. Chollet, "Building Autoencoders in Keras," Keras, 14 05 2016. [Online]. Available: https://blog.keras.io/building-autoencoders-in-keras.html. [Accessed 16 02 2019].

[80] A. Bronshtein, "A Quick Introduction to K-Nearest Neighbors Algorithm," 12 04 2017. [Online]. Available: https://medium.com/@adi.bronshtein/a-quick-introduction-to-k-nearest-neighbors-algorithm-62214cea29c7. [Accessed 03 03 2019].

[81] O. Harrison, "Machine Learning Basics with the K-Nearest Neighbors Algorithm," Towards Data Science, 10 09 2018. [Online]. Available: https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761. [Accessed 03 03 2019].

[82] G. Chauhan, "All about Naive Bayes," Towards Data Science, 08 10 2018. [Online]. Available: https://towardsdatascience.com/all-about-naive-bayes-8e13cef044cf. [Accessed 03 03 2019].

[83] G. Seif, "A Guide to Decision Trees for Machine Learning and Data Science," Towards Data Science, 30 11 2018. [Online]. Available: https://towardsdatascience.com/a-guide-to-decision-trees-for-machine-learning-and-data-science-fe2607241956. [Accessed 03 03 2019].

[84] N. Donges, "The Random Forest Algorithm," Towards Data Science, 22 02 2018. [Online]. Available: https://towardsdatascience.com/the-random-forest-algorithm-d457d499ffcd. [Accessed 03 03 2019].

[85] J. Le, "A Tour of The Top 10 Algorithms for Machine Learning Newbies," Cracking The Data Science Interview , 11 01 2018. [Online]. Available: https://medium.com/cracking-the-data-science-interview/a-tour-of-the-top-10-algorithms-for-machine-learning-newbies-7228aa8ef541. [Accessed 03 03 2019].

[86] R. Gandhi, "Support Vector Machine—Introduction to Machine Learning Algorithms," Towards Data Science , 07 06 2018. [Online]. Available: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47. [Accessed 03 030 2019].

[87] M. Zeng, L. T. Nguyen, B. Yu, O. J. Mengshoel, J. Zhu, P. Wu and J. Zhang, "Convolutional Neural Networks for human activity recognition using mobile sensors," in *6th International Conference on Mobile Computing, Applications and Services*, Austin, 2014.

[88] Z. C. Lipton, J. Berkowitz and C. Elkan, "A Critical Review of Recurrent Neural Networks for Sequence Learning," eprint arXiv:1506.00019v4, 2015.

[89] C. Olah, "Understanding LSTM Networks," 27 08 2015. [Online]. Available: http://colah.github.io/posts/2015-08-Understanding-LSTMs/. [Accessed 17 02 2019].

[90] T. Shen, "GRUs vs. LSTMs," Paper Club, 11 08 2017. [Online]. Available: https://medium.com/paper-club/grus-vs-lstms-e9d8e2484848. [Accessed 17 02 2019].

[91] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE,* vol. 96, no. 11, pp. 2278 - 2324, 1998.

[92] A. Hidaka and T. Kurita, "Consecutive Dimensionality Reduction by Canonical Correlation Analysis for Visualization of Convolutional Neural Networks," in *Proceedings of the ISCIE International Symposium on Stochastic Systems Theory and its Applications*, Fukuoka, 2017.

[93] E. Bisong, "Deep Learning," [Online]. Available: https://ekababisong.org/ieee-ompi-workshop/deep_learning/. [Accessed 09 03 2019].

[94] S. C. Douglas and J. Yu, "Why ReLU Units Sometimes Die: Analysis of Single-Unit Error Backpropagation in Neural Networks," in *52nd Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, 2018.

[95] R. G. Bruballa, "Understanding Categorical Cross-Entropy Loss, Binary Cross-Entropy Loss, Softmax Loss, Logistic Loss, Focal Loss and all those confusing names," 23 05 2018. [Online]. Available: https://gombru.github.io/2018/05/23/cross_entropy_loss/. [Accessed 09 03 2019].

[96] M. D. Zeiler, "ADADELTA: An Adaptive Learning Rate Method," *arXiv:1212.5701*, 2012.

[97] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann and I. H. Witten, "The WEKA Data Mining Software: An Update.," *SIGKDD Explorations*, vol. 11, no. 1, pp. 10-18, 2009.

[98] F. Chollet, "Keras," https://keras.io, 2015.

[99] HTC, "HTC Research," HTC, [Online]. Available: https://research.htc.com. [Accessed 07 02 2019].

[100] J. Yang, "Toward physical activity diary: motion recognition using simple acceleration features with mobile phones," in *IMCE '09 Interactive multimedia for consumer electronics*, Beijing, 2009.

[101] J. O. Laguna, A. G. Olaya and D. Borrajo, "A Dynamic Sliding Window Approach for Activity Recognition," in *UMAP 2011: User Modeling, Adaption and Personalization*, Girona, 2011.

[102] E. Frank, M. Hall, G. Holmes, R. Kirkby, B. Pfahringer, I. H. Witten and L. Trigg, "Weka-A Machine Learning Workbench for Data Mining," in *Data Mining and Knowledge Discovery Handbook*, Boston, Springer, 2009, pp. 1269-1277.

[103] B. Pratool, "Context-based Human Activity Recognition Using Multimodal Wearable Sensors," USF Scholar Commans, Tampa, 2017.

Tezden Üretilmiş Yayınlar

İletişim Bilgisi: jemshit@mail.com

Makaleler

J. Iskanderov ve M.A. Güvensan, "Akıllı telefon ve giyilebilir cihazlarla aktivite tanıma: Klasik yaklaşımlar, Yeni çözümler," *Pamukkale Mühendislik Bilimleri Dergisi*, vol. 25, no. 2, pp. 223-239, 2019