DERİN ÖĞRENME TEKNİKLERİ KULLANILARAK ANAYOL TRAFİK ANALİZİ

2019
YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ

MUHAMMET ESAT ÖZDAĞ

DERİN ÖĞRENME TEKNİKLERİ KULLANILARAK ANAYOL TRAFİK ANALİZİ

Muhammet Esat ÖZDAĞ

Karabük Üniversitesi

Lisansüstü Eğitim Enstitüsü

Bilgisayar Mühendisliği Anabilim Dalında

Yüksek Lisans Tezi

Olarak Hazırlanmıştır

KARABÜK

Aralık 2019

Muhammet Esat ÖZDAĞ tarafından hazırlanan "DERİN ÖĞRENME TEKNİKLERİ KULLANILARAK ANAYOL TRAFİK ANALİZİ" başlıklı bu tezin Yüksek Lisans Tezi olarak uygun olduğunu onaylarım.

Dr. Öğr. Üyesi Nesrin AYDIN ATASOY .....
Tez Danışmanı, Bilgisayar Mühendisliği Anabilim Dalı

Bu çalışma, jürimiz tarafından oy birliği ile Bilgisayar Mühendisliği Anabilim Dalında Yüksek Lisans tezi olarak kabul edilmiştir. 27/12/2019

Ünvanı, Adı SOYADI (Kurumu)	İmzası
Başkan : Dr. Öğr. Üyesi Emel SOYLU (SAMÜ)	.....
Üye : Dr. Öğr. Üyesi Burhan SELÇUK (KBÜ)	.....
Üye : Dr. Öğr. Üyesi Nesrin AYDIN ATASOY (KBÜ)	.....

...../...../2019

KBÜ Lisansüstü Eğitim Enstitüsü Yönetim Kurulu, bu tez ile, Yüksek Lisans derecesini onamıştır.

Prof. Dr. Hasan SOLMAZ .....
Lisansüstü Eğitim Enstitüsü Müdürü

"Bu tezdeki tüm bilgilerin akademik kurallara ve etik ilkelere uygun olarak elde edildiğini ve sunulduğunu; ayrıca bu kuralların ve ilkelerin gerektirdiği şekilde, bu çalışmadan kaynaklanmayan bütün atıfları yaptığımı beyan ederim."

Muhammet Esat ÖZDAĞ

ÖZET

Yüksek Lisans Tezi

DERİN ÖĞRENME TEKNİKLERİ KULLANILARAK ANAYOL TRAFİK ANALİZİ

Muhammet Esat ÖZDAĞ

Karabük Üniversitesi
Lisansüstü Eğitim Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Tez Dnışmanı:
Dr. Öğr. Üyesi Nesrin AYDIN ATASOY
Aralık 2019, 83 sayfa

Başarılı bir akıllı ulaşım sisteminin tasarlanmasında, trafik akış tahmini önemli bir yer edinmektedir. Tahminin başarısı, akış verisinin doğruluğu ve zamanında elde edilmesi ile ilişkilidir. Veri sayısındaki yetersizlik, şimdiye kadar gerçekleşen trafik tahmin modellerinde sığ mimarilerin kullanılmasına ya da üretilmiş yapay ölçüm verileri ile modeller tasarlanmasına sebep olmuştur. Bu modeller yeterli başarıya sahip tahmin sonuçları üretememiştir. Büyük veri çağına girdiğimiz günümüzde, trafik yoğunluğundaki artışa paralel olarak, toplanan trafik verilerin çeşitliliği ve büyüklüğünde gözle görülür bir artış gerçekleşmiştir. Bu veri artışı, çalışmamızdaki temel motivasyonu oluşturmaktadır. Çalışmada, bağlantı yollarına sahip bir otoyolun çıkışındaki trafik yoğunluğunun önceden tahmin edilmesi hedeflenmiştir. Çalışmada önerilen tahmin modelleri, büyük veriler ile eğitilerek anlamlı tahmin sonuçları üretebileceği genel kabul gören Derin Öğrenme teknikleri kullanılarak tasarlanmıştır.

Çalışmada kullanılan bu teknikler sırası ile Yinelenen Sinir Ağları (RNN), Uzun Kısa Süreli Hafıza (LSTM), Yığınlı Uzun Kısa Süreli Hafıza (S-LSTM), Çiftyönlü Uzun Kısa Süreli Hafıza (B-LSTM) ve Geçitli Yinelenen Birim (GRU) sinir ağlarıdır. Çalışmada kullanılan veri seti, otoyol üzerinde 6 farklı noktaya yerleştirilmiş döngü sensörleri ile toplanmış 929.640 ölçüm verisini içermektedir. Tüm verinin %90, %80 ve %70'ini içerecek ve sıralı bir şekilde bölünmesi şartı ile 3 farklı eğitim veri seti oluşturulmuştur. Oluşturulan veri setlerinden geri kalan kısımlar test veri seti olarak kullanılmıştır. Geliştirilen modellerin test veri seti üzerindeki tahmin başarıları Ortalama Karesel Hata (MSE) ve Mutlak Ortalama Hata (MAE) değerleri hesaplanarak kaydedilmiştir. Ayrıca tüm modeller, farklı iterasyon sayıları ile çalıştırılmış ve öğrenme üzerine eğitim seti büyüklüğü ve iterasyon sayısının etkisi araştırılmıştır. Düşük MSE ve MAE değerleri ile trafik akış tahmininde Derin Öğrenme tekniklerinin başarılı sonuçlar üretmiş olması, trafik akış problemlerinde bu modellerinde kullanılabileceğini göstermektedir. Seçilen Derin Öğrenme teknikleri ile tasarlanan modellere ait sonuçlar MSE değerleri bakımından karşılaştırıldığında, en düşük MSE değeri olan 36.60 ile B-LSTM'in en iyi tahmin performansı gösterdiği bulunmuştur. Sonuçlar MAE değerleri bakımından değerlendirildiğinde, B-LSTM'e göre daha az mimari karmaşıklığa sahip ve daha az hesap yüküne sahip GRU tekniğinin 0.18 MAE değeri ile en iyi teknik olduğu sonucuna ulaşılmıştır.

Anahtar Sözcükler : Derin Öğrenme, RNN, GRU, LSTM, trafik akış tahmini.
Bilim Kodu **:** 92432

ABSTRACT

M. Sc. Thesis

ANALYSIS OF HIGHWAY TRAFFIC USING DEEP LEARNING TECHNIQUES

Muhammet Esat ÖZDAĞ

Karabuk University
Institute of Graduate Programs
Department of Computer Engineering

Thesis Advisor:
Assist. Prof. Dr. Nesrin AYDIN ATASOY
December 2019, 83 pages

Traffic flow forecasting has an important for designing a successful intelligent transportation system. The success of the prediction is related to the accuracy and timely measurement of the flow data. Lack in the number of data has led to the use of shallow architectures in the traffic prediction models realized so far or to design models with reproducted measurement data. These models failed to produce predictive results with sufficient success. Now days, as we enter big data era, there has been a significant increase in the diversity and size of the collected traffic data in parallel with the increase in traffic density. This increase in data constitutes the main motivation in our study. In this study, it is aimed to predict the traffic density at the exit of a motorway with connection roads. The prediction models proposed in the study are designed by using the generally accepted Deep Learning techniques in which they can produce meaningful prediction results by training with big data.

The techniques used in this study are Recurrent Neural Networks (RNN), Long Short Term Memory (LSTM), Stacked Long Short Term Memory (S-LSTM), Bidirectional Long Short Term Memory (B-LSTM) and Gate Recurrent Unit (GRU) nerve network are. The dataset used in the study consists of 929.640 measurement data collected by loop sensors located at 6 different points on the motorway. Three different training data sets were created on the condition that they would be 90%, 80% and 70% of all data and should be divided in order. The remaining parts of the dataset were used as test dataset. Forecasting results of the developed models on the test dataset were recorded by calculating the Mean Square Error (MSE) and Absolute Mean Error (MAE) values. In addition, all models were run with different epochs and the effect of the training set size and epoch size on learning was investigated. The fact that Deep Learning techniques have produced successful results in traffic flow estimation with low MSE and MAE values shows that these models can be used in traffic flow problems. When the results of the models designed with the selected Deep Learning techniques were compared in terms of MSE values, it was found that B-LSTM had the best predictive performance with the lowest MSE value of 36.60. When the results were evaluated in terms of MAE values, it was concluded that GRU technique with less architectural complexity and less computational load was the best technique with 0.18 MAE value compared to B-LSTM.

Key Words **:** Deep Learning, RNN, GRU, LSTM, traffic flow forecasting.
Science Code : 92432

TEŞEKKÜR

Bu tez çalışmasının planlanmasında, araştırılmasında, yürütülmesinde ve oluşumunda ilgi ve desteğini esirgemeyen, engin bilgi ve tecrübelerinden yararlandığım, yönlendirme ve bilgilendirmeleriyle çalışmamı bilimsel temeller ışığında şekillendiren sayın hocam Dr. Öğr. Üyesi Nesrin AYDIN ATASOY'a sonsuz teşekkürlerimi sunarım.

Eğitim hayatım boyunca maddi, manevi desteklerini hiç eksik etmeyen değerli aileme, akademik yaşantımda desteğini esirgemeyen Elçin'e tüm kalbimle minnettarım.

İÇİNDEKİLER

Sayfa

Sayfa

Sayfa

ŞEKİLLER DİZİNİ

Sayfa

Sayfa

ÇİZELGELER DİZİNİ

Sayfa

KISALTMALAR DİZİNİ

AENN : AutoEncoder Neural Network (Oto Kodlayıcı Sinir Ağı)

AI : Artificial Intelligence (Yapay Zeka)

ANN : Artificial Neural Networks (Yapay Sinir Ağları)

ARIMA : Autoregressive Integrated Moving Average (Bütünleşik Otoregresif Hareketli Ortalama)

ARMA : Autoregressive Moving Averages (Otoregresif Hareketli Ortalamalar)

B-LSTM : Bidirectional Long Short-Term Memory (Çiftyönlü Uzun Kısa Süreli Hafıza)

BPNN : Back Propagation Neural Network (Geri Yayılım Sinir Ağı)

CLI : Command Line Interface (Komut Satırı Arayüzü)

CNN : Convolutional Neural Network (Evrişimli Sinir Ağı)

CPU : Central Processing Unit (Merkezi İşlemci Birimi)

DBN : Deep Belief Networks (Derin İnanç Ağları)

DENSE : Dense Fully Connected Layer (Tam Bağlantılı Derin Sinir Ağı)

DNN : Deep Neural Networks (Derin Sinir Ağı)

DÖ : Derin Öğrenme

FFNN : Feedforward Neural Network (İleri Beslemeli Sinir Ağı)

GUI : Graphical User Interface (Grafiksel Kullanıcı Arayüzü)

GPS : Global Positioning System (Küresel Konum Belirleme Sistemi)

GPU : Graphics Processing Unit (Grafik İşleme Birimi)

GRU : Gated Recurrent Unit (Geçitli Yinelenen Birim)

INRIA : Inria Aerial Image Labeling Dataset (Inria Aerial Etiketli İmaj Veri Seti)

KNN : K-Nearest Neighbors (K-En Yakın Komşu)

LPDS : License Plate Dataset (Araç Plakası Veri Seti)

LReLU : Leaky Rectified Linear Activation (Zayıf Rectified Linear Aktivasyon)

LSTM : Long Short-Term Memory (Uzun Kısa Süreli Hafıza)

MAE : Mean Absolute Error (Ortalama Mutlak Hata)

MLP : Multilayer Perceptron (Çok Katmanlı Yapay Sinir Ağı)

MRE : Mean Relative Error (Ortalama Bağıl Hata)

MSE : Mean Square Error (Ortalama Karesel Hata)

MTL : Multi-Task Regression (Çoklu-Görev Regresyon)

NAG : Nesterov Accelerated Gradient (Nesterov Hızlandırılmış Gradyan)

NN : Neural Networks (Sinir Ağ)

DRL : Deep Reinforcement Learning (Pekiştirmeli Derin Öğrenme)

PeMS : Performance Measurement System (Verim Ölçüm Sistemi)

ReLU : Rectified Linear Units (Rectified Linear Aktivasyon Fonksiyonu)

RF : Random Forest (Rastgele Orman)

RNN : Recurrent Neural Network (Yinelenen Sinir Ağı)

SGD : Stochastic Gradient Descent (Stokastik Gradyan İnişi)

S-LSTM : Stacked Long Short-Term Memory (Yığınlı Uzun Kısa Süreli Hafıza)

sRNN : Simple Recurrent Neural Network (Basit Yinelenen Sinir Ağı)

SSD : Single Shot Detection (Tek Adım Belirleme)

STSD : Swedish Traffic Signs Dataset (İsveç Trafik İşaretleri Veri Kümesi)

SVM : Support Vector Machine (Destek Vektör Makinaları)

SAE : Stacked Auto-Encoder (Yığınlı Oto-Kodlayıcı)

SIFT : Scale Invariant Feature Transform (Ölçek Değişmez Özellik Dönüşümü)

BÖLÜM 1

GİRİŞ

Bireysel gezginler, iş gezginleri ve kamu kurumları için trafik akış bilgilerinin önceden bilinebilmesi hayati önem taşır [1]. Bu bilgi, seyehat konforunda iyileşme, trafik sıkışıklığında azalma, karbon emisyonunda azalma ve akıcı bir trafik ortaya çıkmasında önemli rol oynar. Son dönemde akıllı kent ve trafik sistemlerinin hızlı yayılması ve gelişmesi ile trafik akış tahmini kritik bir unsur olarak araştırmalar arasına girmiştir. Trafik akış tahmini, çeşitli sensör kaynaklarından gerçek zamanlı ya da geçmiş zamanlı olarak toplanan trafik verilerine bağlıdır. Teknolojik gelişmelere paralel olarak gelişen trafik sensör teknolojileri, ortaya çıkardığı büyük veri sayesinde tahmin doğruluğuna olumlu etkilerde bulunmaktadır.

Geleneksel Sinir Ağları [2], bayesian modelleri [3], bulanık mantık [4] ve istatiksel modeller [5][6] trafik akış tahmin çalışmalarında sıklıkla kullanılmakta olan yaklaşımlar arasındadır. Bu yaklaşımlar arasında özellikle NN'in, gelecek trafik akış tahmininde, veriler arasındaki gizli kuralların keşfedilmesinde ve tahmin edilmesinde başarısı kanıtlanmıştır. Büyük veri çağına giriş ile tahmin doğruluğunun arttırılması için ortaya fırsatlar çıkmıştır. Bu gelişmeler, çalışmada sığ modellerin aksine, veri üzerindeki özellik çıkarımının artması ve büyük verinin işlenmesindeki öğrenme başarısı nedeniyle Derin Sinir Ağlarını (DNN) kullanmayı motive etmiştir. DNN, geleneksel bir yapay sinir ağından daha çok katmana sahip, karmaşık bir mimariden faydalanır ve bu sayede daha iyi sonuçlar elde edebilir. Literatür tarandığında çok az çalışmada bir ulaşım ağının tamamı kullanılarak tahmin yapılmaya çalışıldığı gözlenmiştir. Bu çalışmada tek bir lokasyon yerine, bağlantılı birçok lokasyon verisi ile tahmin başarısı geliştirilmeye çalışılmıştır.

Bu tez çalışmasında, İngiltere'de bulunan ve 16 km uzunluğa sahip M57 (Liverpool çevre yolu) otoyoluna bağlanan 5 farklı kavşağa ait, araç türü, miktar

ve zaman verileri bir zaman serisi vektörü olarak kullanılarak, güney çıkışındaki araç sayısı 15 dakika sonrası için tahmin edilmeye çalışılmıştır.

Modelin eğitim ve test sürecinde, 2015 Nisan ayı ve günümüz 2019 yılı Ağustos ayları arasında, her 15 dakikada bir veri üreten ve otoyolun farklı 6 noktasına (1 çıkış, 5 bağlantı yolu) yerleştirilmiş döngü sensörlerinden elde edilen 929.640 farklı ölçüm verisi kullanılmıştır. Model, Yineleyen Sinir Ağlarının (RNN) varyasyonları ile ayrı ayrı tasarlanmış ve sonuçlar karşılaştırılmıştır.

1.1. TEZİN AMACI VE KAPSAMI

Trafik akış kontrolü, bazı zorluklar nedeniyle araştırmacılar için zor bir problem olmaya devam etmektedir. Bu zorlukların en önemli ikisi: modelleme ve optimizasyondur [7]. Bu problemin çözümü için başlıca iki tür yaklaşım gözlenmektedir. Bunlardan ilki, farklı trafik noktalarında ölçülen büyük ölçekli verinin incelenerek bir modelin tasarlanmasıdır. İkinci bir yaklaşım ise, Yapay Zeka (AI) veya simülasyon teknikleri kullanılarak gelecek trafik akışının modellenmesidir.

2006 yılından günümüze, DNN'ler artan bir şekilde çalışma alanı olarak karşımıza çıkmaktadır [8]. Buna karşın trafik akış problemlerinde kullanımı henüz sınırlı sayıdadır. Bilgisayar veri depolama ve işleme sistemlerindeki gelişme ve maliyet düşüşü, veri toplama ve biriktirme işlemlerinin daha kolay ve yaygın bir şekilde yapılabilmesi sonucunu doğurmuştur. Bu ortaya çıkan büyük verinin stokastik yapısı, doğrusal olmayan problemleri modeleyebilme gücü dikkate alındığında, DNN'leri avantajlı hale getirmektedir. Bu avantaj, tez çalışmasında DNN teknikleri kullanılmasının temel nedenidir. Diğer bir neden, geleneksel NN'lere göre DNN'in eğitim sürecinde, özellik çıkarımında, insan uzmanlığına gerek duymaksızın daha iyi performans göstermesidir [9].

Başarılı bir trafik akışının modellenebilmesi için ondan önce gelen en az dört kavşağın dikkate alınması gerekliliği sonucuna ulaşmıştır [10]. Tez çalışmasında İngiltere'deki M57 isimli otoyol ve bu yola bağlanan beş kavşağa ait trafik akış verisi kullanılarak, M57 otoyolunun çıkışındaki araç sayısı tahmin edilmeye çalışılmıştır.

Veri seti, 2015 yılı Nisan Ayı ve 2019 yılı Ağustos ayları arasında otoyolu ve bağlantı yollarını kullanan, 15'er dakikalık aralıklarla toplanmış, binek araç ve büyük ölçekli araç (kamyon vs.) sayıları ve takvim günü bilgilerinden oluşmaktadır. Elde edilmeye çalışılan, çıkış noktasındaki araç sayısıdır.

Tez çalışmasında temel olarak DNN'lere yoğunlaşılmış ve trafik akışının tahmininde başarısı incelenmiştir. Tez çalışmasında birincil hedef, trafik sorunlarında DNN'lerin kullanılmasını araştırmak ve literatür için katkılar sağlamakıtır. İkinci hedef olarak, stokastik yapısı ile tahmini güç olan karmaşık ve büyük trafik akış planlamasının çözümüne yardımcı olmaktır.

1.2. HİPOTEZ

DNN kullanılarak stokastik bir yapıda olan trafik akışını önceden tahmin etmek, bu tez çalışmasının temelini oluşturmaktadır. Çalışmada trafik akış yoğunluğunun tahmini, Basit Yinelenen Sinir Ağı (sRNN), Uzun Kısa Süreli Hafıza (LSTM), Yığınlı Uzun Kısa Süreli Hafıza (S-LSTM), Çiftyönlü Uzun Kısa Süreli Hafıza (B-LSTM) ve Geçitli Yinelenen Birim (GRU) gibi DNN teknikleri ile incelenmiştir. Deneysel çalışmada ilk olarak, stokastik yapıda olan trafik akış verilerinin DNN ile kullanılabileceği incelenmiştir. Tasarlanan ağların sonuçları gerçek hayat sonuçları ile karşılaştırılmış ve algoritmalar arasındaki performans incelenmiştir. Çalışmada DNN'lerin, trafik akış tahmin performanslarının karşılaştırılması analiz edilmesi. ve trafik akışında başarılı bir tahmin modeli oluşturulmasıdır. Sonuçlar test kümesi üzerinden elde edilerek, DNN'lerin tahminleme başarı sonuçları bulunmuştur.

Tez çalışmasında cevap aranan temel sorular şu şekilde sıralanabilir:

1. DNN'ler trafik akış kestirimi için kullanılabilir mi?
2. DNN'ler içinde hangi teknik daha avantajlıdır?
3. DNN ile önceden tahmin edilemeyen stokastik bir yapıda olan trafik akışı için önceden belirlenmiş bir strateji başarılı olabilir mi?

1.3. TEZİN BÖLÜMLERİ

Tez çalışmasının amaç, kapsam ve hipotez hakkında bilgilerin içerdiği bölümden sonra tez şu şekilde devam etmektedir:

2. Bölüm, trafik akış problemlerine DÖ teknikleri kullanılarak getirilen öneriler ve modellerin içerdiği çalışmaların özetlerini içeren kapsamlı bir literatür araştırması içermektedir. Bölüm 3'te, ANN, DÖ tanımı, Evrişimli Sinir Ağları (CNN) ve RNN hakkında kavramsal bilgiler sunulmaktadır. 4. Bölüm, çalışmada kullanılan veri seti hakkında tanımlayıcı veriler, deneysel çalışma ve sonuçlarına yer verilmiştir. Son bölüm olan 5. Bölüm'de, tez çalışmasının sonuçları tartışılmaktadır.

BÖLÜM 2

ÖNCEKİ ÇALIŞMALAR

Yuan vd. akıllı ulaşım sistemleri için önemli bir bileşen olan trafik işaretlerini tanımak için, video tabanlı, DÖ modelini geliştirdiler [11]. Bu model, CNN ile tasarlanmıştır. Çalışmada kullanılan veriler, araç içine entegre edilmiş kameralar sayesinde elde edilmiştir. Bu videolarda görüntüler kırpılarak, her işaret için 4 ya da 5 farklı uzaklıktan tekrarlı olarak elde edilen 6 bin işaret simgesi barındıran bir eğitim seti oluşturulmuştur. Çalışma sonucunda, mekansal ön bilginin eğitim sonuçlarının iyileşmesine katkısı olduğu gözlenmiştir. Çalışmada kullanılan CNN modeli sonuçlarının, aynı veri seti üzerinde uygulanan, Destek Vektör Makinaları (SVM) ve diğer DNN modellerine göre daha iyi sınıflandırma sonuçları verdiği bulunmuştur.

John vd. çalışmalarında, farklı aydınlatma koşulları için, DÖ ve Makine Öğrenmesi teknikleri kullanan, trafik ışıkları tanıma algoritması sunmuşturlar [12]. Çalışmada, görsel video görüntülerinden öznitelikleri çıkartmak ve tespit etmek için CNN kullanılmıştır. Tanıma doğruluğunun arttırmak için, trafik ışığının bulunduğu görsele bir Küresel Konum Belirleme Sistemi (GPS) verisi ilişkilendirilmiştir. Gerçek dünyada çeşitli otoyollardan elde edilmiş deney verileri, 2402 çerçeveden oluşan öğleden sonra grubu ve 934 çerçeveden oluşan akşam grubu olmak üzere iki ayrı zaman diliminde gruplandırılmıştır. Çalışma sonuçları, GPS verileri kullanılması durumunda başarı performansının arttığını göstermesine rağmen, aynı görsel çerçevede birden çok trafik ışığı olma durumunda tanıma hatalarının arttığı gözlenmiştir.

Yingying vd. trafik işaretlerini hızlı ve doğru bir şekilde tanımak ve sınıflandırmak için, CNN ve tam bağımlı konvolüsyonel sinir ağı (FCN) bileşenleri kullanan bir model geliştirmişlerdir [13]. Çalışmada önerilen bu modele, kamuya açık olan, İsveç

Trafik İşaretleri Veri Kümesi (STSD)'den elde edilen ve manuel olarak etiketlenen 2 bin fotoğraf karesine uygulanmıştır. Önerilen bu model, %98,67'lik kestirim başarısına ulaşmıştır.

Carlo vd., çalışmalarında düşük çözünürlüklü trafik video görüntülerinde araç tespiti için CNN mimarisinin kullanıldığı bir model önerisinde bulunmuşlardır [14]. Sonuçlar, önerilen sistemin %94.72 sınıflandırma başarısına ulaştığını göstermiştir. 2 GHz CPU'ya sahip bir makine ile sistem test edildiğinde, 22.59 ms'lik tepki süresi ile modelin gerçek zamanlı kullanılabileceği sonucuna varılmıştır.

Rongqiang vd. yaptıkları çalışmada, CNN kullanarak Çin otoyol trafik işaretlerini tespit ve tanıma için bir model geliştirmişlerdir [15]. Gerçekleştirilen model, farklı trafik işaretleri ve metinlerin sınıflandırılması için, CNN'e dayandırılmış, etkili öznitelik çıkarımı yapmak üzere eğitilmiştir. Çalışmada veri seti olarak, Araç Plaka Veri Seti (LPDS) kullanılmıştır. LPDS, farklı çözünürlük, aydınlatma ve bakış açısına sahip 410 görüntü içermektedir. Modelin sınıflandırma başarısı %95 olarak bulunmuştur.

Kartik vd.'in Temsil Öğrenme teknikleri kullanarak gerçekleştirdikleri çalışmada, uykulu sürücü tespit etmek için bir model önerilmiştir [16]. Önerilen model, insan yüzünün çeşitli gizli özellikleri ve doğrusal olmayan etkileşimlerini yakalamak için CNN'den yararlanmıştır. Elde edilen video görüntülerinden kareler çıkartılmış, Viola ve Jones Haar teknikleri ile yüz bölgesi tespiti yapılmış, kırpılan resim verisi üzerinde model uygulanmıştır. 30 farklı denekten görüntüler elde edilerek eğitilen model, test kümesi üzerinde %92.33 sınıflandırma doğruluğu ulaşmıştır. Aynı kişinin sürekli görüntüleri alınarak etiketlenen bir veri seti üzerinde %88 sınıflandırma başarı elde edilmiştir.

Di vd. çalışmalarında, Çin genelinde elde edilen otoyol trafik videoları üzerinde, araç plakalarının tanınması ve görsel bir uyarı sistemi oluşturulması için, DÖ yaklaşımına sahip bir model önermişlerdir [17]. Önerilen model, özellik çıkarımı için CNN yapılarını kullanmıştır, sınıflandırma için ise SVM kullanılmıştır. Çalışmada kullanılan veriler, otoyola kurulmuş 30 fps kare hızına sahip kameralar ile 24 saat

esası ile toplanmıştır. Modelin sınıflandırma doğruluk oranı, %98.9 olarak bulunmuştur.

Xiaoguang vd., çevrim içi trafik rota önerileri sunabilen, DÖ tabanlı bir model tasarlamışlardır [18]. Model, 30 bin taksiden alınan GPS verisi, Wuhan şehrine ait trafik kamera görüntüleri, kamuya açık çevresel bilgiler (nem, sıcaklık gibi) içeren bir veri seti üzerinde uygulanmıştır. Sonuçlar, önerilen modelin trafik ulaşımında, stokastik yapının DÖ mimarileri ile temsil edilebileceği ve dinamik olarak doğru rota önerisinde bulunabileceğini gösterilmiştir.

Yisheng vd. geleneksel kestirim algoritmalarının otoyol trafik akış tahmininde tatmin edici sonuçlar veremediği için, çalışmalarında zamansal ve mekansal korelasyonları dikkate alan yeni bir DÖ modeli önermişlerdir [1]. Yapılan bu çalışmada, genel trafik akış özelliklerinin öğrenilmesi için Oto-Kodlayıcı Sinir Ağı (AENN) modeli kullanılmıştır. Geliştirilen modele, PeMS veritabanından toplanan veriler, sayısal bir örnek olarak uygulanmıştır. Trafik verileri, Kaliforniya genelinde, eyalet çapında dağıtılmış olan, otoyol sistemine entegre, 15 bin dedektör tarafından her otuz saniyede bir toplanmıştır. Çalışmanın sonuçları, trafik verilerinin doğrusal bir yapıda olmaması sebebiyle DÖ tekniklerinin kullanılmasının, trafik akışının gizli özelliklerini keşfedilebileceğini ortaya çıkartmıştır. Modelin doğruluk oranı, % 93 olarak tespit edilmiştir.

Wenhao vd. yaptıkları çalışmada, Derin İnanç Ağları (DBN) ve Çoklu-Görev Regresyon (MTL) teknikleri kullanarak geliştirilen bir model ile otoyol trafik akışını tahmin etmeye çalışmıştır [19]. Çalışmada, PeMS ve highway system of China otoyol veri setleri kullanılmıştır. Tasarlanan model çıktısı, belirlenen bir gözlem noktasında, bir sonraki zaman için oluşacak trafik yoğunluğu olarak tanımlanmıştır. Yapılan çalışma sonuçları, Bütünleşik Otoregresif Hareketli Ortalama (ARIMA), Bayes modelleri, SVM ve NN modelleri ile karşılaştırıldığında %3 daha iyi sonuç elde edildiği gözlenmiştir.

Xiaolei vd. çalışmalarında, otoyollarda ortaya çıkabilecek trafik sıkışıklığını önceden tahmin etmek ve büyük ölçekli ulaştırma ağlarını analiz etmek için bir DÖ modeli

önermişlerdir [20]. GPS verilerine dayanan bu DNN modelinin tasarımı, Boltzmann Makinesi ve RNN ile gerçekleştirilmiştir. Modelin etkinlik ve verimliliğini doğrulamak için Çin'in Ningbo kentinde toplanan veriler ile sayısal bir çalışma yapılmıştır. Çalışma sonuçlar, Grafik İşleme Birimi (GPU) tabanlı bir paralel hesaplama ortamında, 6 dakikadan daha az sürede %88 doğruluk oranına ulaşabileceğini göstermektedir.

Nicholas vd. çalışmalarında DNN kullanarak kısa vadeli otoyol trafik akış yoğunluğunu tahmin etmeye çalışmışlardır [21]. Trafik akışının öngörülmesi, arızalar, tıkanıklık ve düzelme arasında doğrusallık olmaması sebebiyle oldukça zordur. Çalışmada, DÖ mimarisi ile doğrusal olmayan mekansal ve zamansal etkilerin yakalanabileceği gösterilmiştir. Modelde kullanılan veri seti, Amerika Birleşik Devletleri İllinois Ulaştırma Bakanlığı'ndan tarafından, eyalet geneline yerleştirilmiş dokuz yüz trafik yoğunluk sensörü kullanılarak elde edilmiştir. Model, I-55 isimli yolun trafik yoğunluğunu iki özel etkinlik sırasında (Chicago Bears futbol müsabakası ve aşırı kar fırtınası) oluşacak ani trafik yükünü tahmin etmek için kullanılmıştır. Çalışma sonuçları, DÖ tekniklerinin kısa vadeli trafik akış tahmininde başarılı olduğunu ortaya çıkarmıştır.

Xiaolei vd. çalışmalarında, büyük ölçekli bir ulaşım ağında hız tahmini için CNN mimarisi ile tasarlanmış bir NN modeli önermişlerdir [22]. Çalışmada kullanılan veriler, Çin'in başkenti ve dünyanın en büyük şehirlerinden biri olan Pekin'de, yaklaşık 10 bin taksinin GPS cihazları ile donatılması ile sağlanmıştır. 37 gün boyunca her iki dakikada bir, araç konumları, araç hızları gibi bilgiler kaydedilmiştir. Yapılan çalışmada sonuçlar, K-En Yakın Komşu (KNN), ANN, Rastgele Orman (RF), RNN, LSTM ve AENN modelleri ile ayrı ayrı karşılaştırılmış ve önerilen sistemin ortalama %42.91 oranında daha iyi bir iyileşme sağladığı ortaya çıkarılmıştır.

Zheng vd. çalışmalarında LSTM mimarisi kullanarak, kısa vadeli trafik akış tahmini yapabilen bir yaklaşım sunmaktadırlar [23]. Çalışmada önerilen model, Pekin Trafik Yönetim Bürosu tarafından 500'den fazla gözlem istasyonundan toplanmış olan verilere uygulanmıştır. Her gözlem istasyonundan 6 ay süresince, araç yoğunluk

değerleri, şerit doluluk oranları ve ortalama hız gibi niteliklere sahip 25 milyon veri elde edilmiş ve kullanılmıştır. Trafik tahmini 15 dakika olduğunda, tahmin sonuçlarının orjinal veriye yakın olarak elde edildiği gözlenmiştir. Bu çalışmada Ortalama Bağıl Hata (MRE), %6.41'dir.

Trafik akış çalışmalarında veri yetersizliği problerinde simülasyon çok sık başvurulan bir yöntemdir [24]. Li vd. çalışmalarında, Pekiştirmeli Derin Öğrenme (DRL) tekniği kullanarak otoyol trafik ışık sinyalizasyonu zamanlaması için bir model önermişlerdir [7]. Hazırlanan simülasyon yazılımı sayesinde üretilen veri, model üzerinde uygulanmıştır. Bu model, kontrol eylemlerini ve sistem durumlarının değişimini dolaylı olarak modelleyerek trafik sistemlerinin dinamiklerini ve optimum kontrol planını aynı anda öğrenmiştir. Sonuçlar hazırlanan modelin, geleneksel yöntemleri geride bıraktığını göstermiştir.

Arief vd. çalışmalarında ulaşım sistemini etkileyen, hava şartları ve kazalar arasındaki korelasyonu ispatlamış ve bu ilişkiyi kullanan, trafik akış tahmin doğruluğunu arttırmayı hedefleyen bir mimari önermişlerdir [25]. Bu çalışmada mimari için, DBN kullanılmıştır. Çalışmada kullanılmakta olan trafik verileri PeMS'den, hava durumu verileride Utah (Amerika Birleşik Devletleri) Üniversitesi, MesoWest projesinden elde edilmiştir. Verinin bütünlüğü ve güvenirliğinin artırılması için farklı kaynaklardan gelen bu verilere, veri füzyonu yaklaşımı uygulanmıştır. Çalışma sonuçları, veriye dayalı trafik akış sistemi tahmininin, diğer gelişmiş tekniklerden daha iyi performans gösterdiğini ortaya çıkarmıştır.

Jingyuan vd., çalışmalarında trafik akış hızı tespiti ve tıkanıklık kaynağını keşfeden bir DNN mimarisi önermişlerdir [26]. Bu mimari, hata geri beslemeli yinelenen bir CNN tekniği içermektedir. Çalışmada kullanılmakta olan hacim veri seti, Pekin şehrine ait ve ortalama günlük 200 bin araç tarafından kullanılan bir otoyoldan elde edilmiştir. Çalışmada otoyola ait hız verisi, ticari taksilere monte edilmiş GPS cihazları vasıtasıyla elde edilmiştir. Veri seti, 5 dakikada aralıklar ile, 25 gün boyunca toplanmış ve kaydedilmiştir. Çalışma sonuçlarında model, kısa ve uzun mesafe olmak üzere iki farklı senaryo ile test edildiğinde, hız kestirim sonuçlarında, SVM ve ARIMA modellerine göre daha küçük Ortalama Mutlak Hata (MAE)

değerine sahip olduğu bulunmuştur. Uzun mesafe senaryolarında hata oranının daha da az olduğu tespit edilmiştir.

Rui vd. çalışmalarında LSTM ve GRU tekniklerini kullanarak kısa vadeli trafik akış tahmini yapan bir model önermişlerdir [27]. Hazırlanan sinir ağ modeline, PeMS trafik veri seti uygulanmıştır. Çalışma sonuçları Otoregresif Hareketli Ortalamalar (ARMA), ARIMA gibi mevcut modeller ile yapılan çalışma sonuçları ile karşılaştırıldığında, trafik akışının doğrusal olmayan ve stokastik doğasını tanımlama gücüne sahip DNN modelinin daha başarılı bir sonuç verdiği gözlenmiştir.

Hao-Fan vd., çalışmalarında trafik akışının verimliliği ve tıkanıklık problemlerini optimize etmek için DÖ yaklaşımlarını kullanan bir model önermişlerdir [28]. Çalışmada optimizasyon başarısı sağlamak amacıyla yığınlı Oto-Kodlayıcı modeli kullanılmıştır. Modelin eğitiminde, denetimsiz bir öğrenme algoritması ile trafik akış özelliklerini modele öğretmek için Taguchi yöntemi kullanılmıştır. Çalışmada kullanılan veri seti, İngiltere'deki M6 otoyolunda 4 kavşağa yerleştirilmiş sensörler yardımı ile toplanan 1 aylık iş günü verileri kullanılmıştır. Önerilen model %90 tahmin başarısı ile diğer geleneksel iki modele göre, trafik akış optimizasyonu için başarımı çok yüksek ölçüde iyileştirdiğini göstermiştir. Stokastik verilere sahip tahmin problemleri için en uygun yaklaşım olduğu gözlenmiştir.

Zhenhua vd., çalışmalarında DÖ teknikleri kullanarak sosyal medya verileri ile trafik kazalarının tespit edilmesi için bir model önermişlerdir [29]. Modelin mimarisi, DBN ve LSTM teknikleri ile gerçekleştirilmiştir. Tasarlanan modele, Amerika Birleşik Devletleri'nde iki büyük metropol olan, Northern Virginia ve New York City bölgelerinde toplanmış 3 milyon tweet uygulanmıştır. Modelin başarısı %85 doğruluk oranı olarak elde edilmiştir. Sonuçlar, SVM ve gözetimli Latent Dirichlet Tahsisi (sLDA) tekniklerinden daha iyi sınıflandırma sonucu üretmiştir. Model sonuçlarının doğrulanması için, trafik kaza tutanakları ve bölgedeki 15 bin yol bilgi sensörü verileri karşılaştırılmıştır. Karşılaştırma, kazaların %66'sının kaza kütükleri ile örtüştüğünü ortaya çıkartmıştır.

Yuhan vd. çalışmalarında, trafik akış hızı tahmini için DNN temelli bir model geliştirmişlerdir [30]. Model mimarisi, trafik akış özelliklerinin stokastik yapısı düşünülerek, DBN kullanılarak tasarlanmıştır. Çin'in Pekin şehrinde, bir otoyol arterinden toplanan trafik akış hızı verilerine dayanarak, model farklı zaman aralıkları için eğitilmiş ve test edilmiştir. Sonuçlar, DBN'nin tüm zaman dilimleri için, Geri Yayılım Sinir Ağı (BPNN) ve ARIMA modellerine göre daha iyi performans gösterdiği ortaya çıkartmıştır.

Yuankai vd., çalışmalarında trafik akış tahmini için, stokastik özelliklere sahip, mekansal ve zamansal özelliklerin göz ardı edilmediği bir hibrid DÖ modeli tasarlamıştır [31]. Model mimarisi, CNN tekniği kullanılarak tasarlanmıştır. Model, PeMS veritabanından toplanan veriler üzerine uygulanmıştır. Çalışma sonuçları, ayrı ayrı, LASSO, Yığınlı Oto-Kodlayıcı (SAE), RNN model sonuçları ile karşılaştırılmış ve önerilen mimarinin en iyi hata performansını gösterdiği bulunmuştur.

Yanjie vd., ulaştırma sistemlerinde kullanılan loop dedektörler ve diğer cihazlardan toplanan verilerde ortaya çıkan ve çalışmaları etkileyen, eksik verileri etiketlemek için derin sinir ağları ile bir yaklaşım önermiştirler [32]. Önerilen bu mimari, SAE'den oluşmaktadır. Bu önerilen DÖ tabanlı yaklaşım, PeMS tarafından sağlanan veri seti üzerinde denenmiştir. Seçilen veri seti üzerinde rastgele tahrip işlemi uygulanarak veri hazır hale getirilmiştir. Önerilen modelin test sonuçları, geleneksel sinir ağlarından daha MAE değerine sahip olduğu göstermiştir.

Xiaoguang vd. çalışmalarında, taksiler üzerine montaj edilmiş GPS cihaz verileri ile trafik akış tahmini yapan bir yaklaşım önermişlerdir [33]. Model mimarisi, DÖ ağları yaklaşımı ile tasarlanmıştır. Çalışmada Wuhan şehrine ait verilerden oluşan bir veri seti kullanılmıştır. Şehirde hizmet veren 30 bin taksiye ait GPS verisi, şehre ait otoyollar ile eşleştirilerek önerilen model üzerinde uygulanmıştır. Önerilen model, geleneksel modellerin sonuçları ile karşılaştırıldığında, %5 daha iyi performans gösterdiği bulunmuştur.

Hongsuk vd., trafik akış tahmini için, gerçek zamanlı veriler kullanan DÖ temelli bir yaklaşım önermiştir [34]. Önerilen sistem, TensorFlow kütüphanesi kullanılarak

gerçekleştirilmiş ve test edilmiştir. Trafik verileri, yarım milyon araca yerleşik bir navigasyon cihazından elde edilmiştir. Bu cihaz GPS tabanlı ve gerçek zamanlı olarak, bir araca ait ortalama hız verisini oluşturmaktadır. Sonuçlar, önerilen modelin doğruluk oranının %99'a ulaştığını göstermiştir.

Rose vd. çalışmalarında, trafikte ortaya çıkan sıradışı durumları tahmin etmek için DÖ temelli bir yaklaşım önermişlerdir [35]. Modelde trafik verisi, Los Angeles genelinde 2 bin 18 loop dedektörü ile, kaza verileri ise, LA Ulaştırma Bakanlığı tarafından sağlanmıştır. Önerilen mimari, LSTM kullanılarak tasarlanmıştır. Model, bazal seviyeye göre %30 ile %50 arasında bir iyileşme ortaya çıkarmıştır.

Tayara vd. çalışmalarında, havadan çekilmiş görüntüleri kullanarak araç algılama ve sayma sistemi geliştirmişlerdir [36]. Sistem mimarisi, özellik çıkarımı için CNN tekniği, sınıflandırma işlemleri için SVM kullanılarak tasarlanmıştır. Alman Havacılık ve Uzay Merkezinin Uzaktan Algılama Teknolojisi Enstitüsü ve Tepegöz Görüntü Araştırma Merkezi Veri Seti (OIRDS) tarafından sağlanan, iki genel veri seti üzerinde sistem değerlendirilmiştir. Değerlendirme sonuçlarına göre, 351 araç içeren 385 görüntü üzerinde %95.09 doğruluk oranı elde edilmiştir.

Chun vd. akıllı otoyol trafik sistemleri için popüler bir araştırma konusu olan, araç logosu tanıma modeli tanıtmışlardır [37]. Öznitelik çıkarımı için, CNN ve Ölçek Değişmez Özellik Dönüşümü (SIFT) ile ayrı ayrı tasarlanmış iki ayrı model geliştirilmiştir. Sınıflandırma prosedürü için her iki modelde SVM kullanmıştır. Çalışmada kullanılan veriler, JiangSu eyaletinde trafikten toplanmıştır. Veriler, 16 farklı araç üreticisinden 26 farklı sınıfta ve her sınıfa ait 200 görüntü örneğine sahip olacak şekilde kategorize edilmiştir. Çalışma sonuçlarında, CNN ve SIFT mimarisine sahip modeller sırası ile %99.23 ve %90.22 doğruluk oranına ulaşmıştır. CNN mimarisine sahip modelin daha iyi performans gösterdiği sonucuna ulaşılmıştır.

Xin vd. çalışmalarında, Bölgesel Bazlı Evrişimli Sinir Ağı (R-CNN) ile tasarlanmış bir yaya algılama algoritması önermişlerdir [38]. Önerilen algoritma, eş zamanlı bir uyarı sistemine sahip olarak uygulanmıştır. Çalışmada veri seti olarak, kamuya açık renkli yaya resimlerinden oluşan Inria Aerial Etiketli İmaj Veri Seti (INRIA)

kullanılmıştır. Deney sonuçları, modelin %84 doğruluk oranına sahip olduğunu göstermiştir.

Mingqi vd. çalışmalarında, trafik kazalarına sebep olan sürücü hatalı davranışlarının önlenebilmesi için bir model önermiştirler [39]. Araç kokpitinin karmaşık arka planı ve ışık farklılıkları, tespitin zorluğundan daha öte önemli zorluklar getirmiştir. Modelin mimarisi, araç içine yerleştirilmiş kamera tarafından oluşturulan RGB görüntülerine uygulanmış, R-CNN tekniğine dayalı bir algoritma ile gerçekleştirilmiştir. Çalışmada kullanılmak üzere, araç kokpitinde 62 farklı kişinin çeşitli davranışlarından, 16 bin 750 görsel içeren bir veri seti oluşturulmuştur. Sonuçlar, önerilen modelin etkinliğini göstermiştir.

Peppa vd. çalışmalarında, çeşitli mevsim şartlarında, düşük kaliteli kapalı devre kamera sistemi görüntülerini kullanarak, CNN temelli, trafik akış düzenini analiz eden bir model önermiştir [40]. Sonuçlar %98.2 doğruluk oranı ile CNN ağlarının aşırı hava koşullarında, trafik düzenin bozukluğunu tespit etmekte kullanılabileceğini göstermiştir.

Alex vd. çeşitli hava koşullarında gerçek zamanlı çalışabilen, şehir ve otoyollarda kentsel nesnelerin tespit edilmesi ve lokasyonların güvenilir bir şekilde bulunması sağlayan bir yaklaşım sunmuşlardır [41]. Çalışmada, gerçek trafik verilerinden oluşan yeni bir veri seti oluşturulmuş ve temel iki gruba bölünmüştür. İlk grup veri seti, otomobil, motosiklet, insan, trafik ışığı, bisiklet, trafik levhası ve otobüs olmak üzere 7 sınıfta etiketlenmiştir. İkinci veri seti grubu ise yollarda en çok kullanılmakta olan 43 temel trafik işaretlerini içermektedir. Çalışmada sunulan DÖ modeli, ilk grup tespiti için R-CNN, ikinci trafik işaretleri grubu için CNN mimarisi kullanılarak tasarlanmıştır. Model, nesne tanımada %74.2, trafik işaret tanımada ise %98.1'e varan doğruluk oranına erişmiştir.

Lin vd. yaptıkları çalışmada, trafik sıkışıklığının otomatik olarak algılanmasını sağlayan, bu sayede, çevresel ve ekonomik kazanımlar elde edilmesinin planlandığı bir model önermişlerdir [42]. Çalışmada kullanılan veri seti, Londra SCOOT trafik merkezinden elde edilmiş, önceden planlanmamış 139 olayı içermektedir. Çalışmada

önerilen model, CNN mimarisi kullanılarak tasarlanmıştır. Sonuçlar, geleneksel sinir ağları ve makine öğrenme tekniklerine göre CNN ağlarının daha iyi sonuç verdiğini göstermiştir.

Julian vd., çalışmalarında Tek Adım Belirleme (SSD) algoritması ile trafik ışığı tespitinde ortaya çıkan çok küçük nesnelerin tespit edilememesi problemine çözüm olarak, entegre bir DÖ yaklaşımı sunmuşlardır [43] ve CNN mimarisi ile tasarlanmıştır. Çalışmada 230 binden daha fazla elle etiketlenmiş trafik ışığı içeren bir veri seti kullanılmıştır. Model, Nvidia Titan Xp üzerinde saniyede on kare ile eş zamanlı çalışabildiği görülmüştür. Çalışma sonuçları, daha çok eğitim verisinin başarı yüzdesini arttıracağını göstermektedir. Model, büyük nesneler için %98, küçük nesneler için %95 varan başarı oranları göstermiştir.

Bensedik vd. çalışmalarımalarında, derin sinir ağları kullanarak araç türlerinin sınıflandırılması için bir model geliştirmişlerdir [44]. Geliştirilen model, CNN mimarisi kullanılarak tasarlanmıştır. Çalışmada kullanılan veri seti, dört kategoriye ayrılmış (okul otobüsü, ambulans, polis ve taşıma araçları) 2400 örnek içermektedir ve el yordamı ile toplanmıştır. Model performansı, i5- 4200M CPU ve 8GB RAM sahip bir ortamda test edilmiştir. Sonuçlar, %90'a varan oranlarda tespit kesinliğine ulaşıldığını göstermiştir.

Zhao vd. çalışmalarında, ImageNet veri seti içinde bulunan araç kümesi üzerinde çalışan ve araç türlerinin tespiti gerçekleştiren Faster R-CNN mimarisi tabanlı bir model geliştirmiştir [45]. Veri seti içerisindeki tüm araçlar yeniden etiketlenmiş ve sırası ile 6:4 oranında eğitim ve test verisi olarak rastgele ayrıştırılmıştır. Geliştirilen model R-CNN ile tasarlanan bir diğer model ile karşılaştırıldığında, Faster R-CNN mimarisinin araç tespitinde doğruluk ve tespit verimliliği açısından belirgin avantajlara sahip olduğunu göstermiştir. Suhao vd. derin sinir ağları kullanarak MIT ve Caltech veri setleri üzerinde araç türlerini tespit eden bir çalışma gerçekleştirmiş ve araç türlerinin sınıflandırılmasında DÖ tekniklerinin verimliliğinin yüksekliğini ispat etmişlerdir [46].

Sommer vd. çalışmalarında büyüyen şehirler ve buna paralel ortaya çıkan trafik yoğunluna çözüm üretmekte kullanılacak araç tespiti için, havadan alınmış görüntüler kullanarak araçların semantik olarak etiketlenmesi için R-CNN mimarisi kullanan bir model önermişlerdir [47]. Çalışma, ImageNet kütüphanesinden faydalanarak önceden eğitilmiş VGG-16 ağını kullanmıştır. Model, kamuya açık bir veri seti ile çeşitli zeminlerde test edilmiştir. Sonuçlar, hatalı tespitlerin sayısının azaltılarak tespit performansının arttığını göstermektedir. Çalışmada ayrıca küçük çözünürlüklü fotoğraflara sahip VGG-16 gibi veri setlerinin kullanılması ile eğitilen ağın, yüksek çözünürlüklü hava görüntülerinden oluşan örnekler üzerinde tespit hatalarının yüksek olduğu sonucuna ulaşılmıştır. Sommer vd. bu sorunun çözümü için Deconvolutional Faster R-CNN mimarisi kullanan yeni bir model önermişlerdir. [48].

Chen vd. çalışmalarında, akıllı ulaşım sistemlerinde önemli bir yeri olan, sürücü emniyet kemeri tespit problemi için CNN ve SVM tabanlı iki model önermiştir [49]. Mevcut emniyet kemer tespit çalışmalarında kullanılan kenar tespit algoritmaları ve Hough dönüşümlerinin, bulanık ve karışık zemin üzerinde hata performansları yüksektir. Çalışmada ilk olarak, CNN ağ yapısı ile geliştirilen algılama modelinin eğitimi için ön cam ve emniyet kemeri ile özellik çıkarımı yapılmış ve görseller etiketlenmiştir. Ardından test görüntülerinde ön cam algılanmış ve kemer tespiti yapılmıştır. Tespit sonuçları daha sonra SVM kullanan bir model ile sınıflandırılmıştır. Sonuçlar, bu önerilen modelin trafik akışında kemer tespitinde daha iyi bir performans gösterdiğini ortaya çıkarmıştır.

Wang vd. çalışmalarında, son yıllarda görüntü sınıflandırma ve nesne tespitinde başarı oranı artan, R-CNN ve CNN ağları ile tasarlanmış gerçek zamanlı araç türü tespit edebilen bir sınıflandırma modeli önermişlerdir [50]. Çalışma sonuçları otomobil ve kamyonlarda, sırası ile %90.65 ve %90.51 doğruluk oranı ile modelin geleneksel makine öğrenme yöntemlerine göre sınıflandırmada büyük bir fark ile üstün geldiğini göstermektedir.

BÖLÜM 3

DERİN ÖĞRENME ALGORİTMALARI

DÖ, ardışık gizli katman sayılarının arttırılarak sinir ağlarına yeni bir bakış açısı kazandıran, makine öğrenmenin bir alt araştırma alanıdır [51]. DÖ kavramında geçen derin kelimesi, ağın başarımındaki artış değil, modelin katmansal derinliğini ifade etmektedir. ANN çoğu zaman bir veya iki temsil katmanının öğrenmesine odaklanır ve bu sebeple çoğu zaman sığ öğrenme olarak adlandırılır. Buna karşın DÖ, çoğu zaman yüzlerce, binlerce ardışık gizli katmana sahip olabilir. Bu bölümde, geleneksel NN yapılarının temelleri, DNN ve önde gelen algoritmalar açıklanmaktadır.

3.1. ANN TEMELLERİ

Sinir ağları, birçok basit birimin merkezi kontrol ünitesi ile paralel olarak çalıştığı, insan beyni ile benzer özelliklere sahip hesaplamalı bir modeldir [52]. Bu bölümde NN bileşenleri ve eğitim süreçleri hakkında bilgi verilmektedir.

3.1.1. ANN

Yaygın olarak "sinir ağları" olarak adlandırılmakta olan ANN üzerine çalışmalar, insan beyninin geleneksel bilgisayarlardan tamamen farklı bir şekilde işlediğinin kabul edilmesi ile hız kazanmıştır [53].

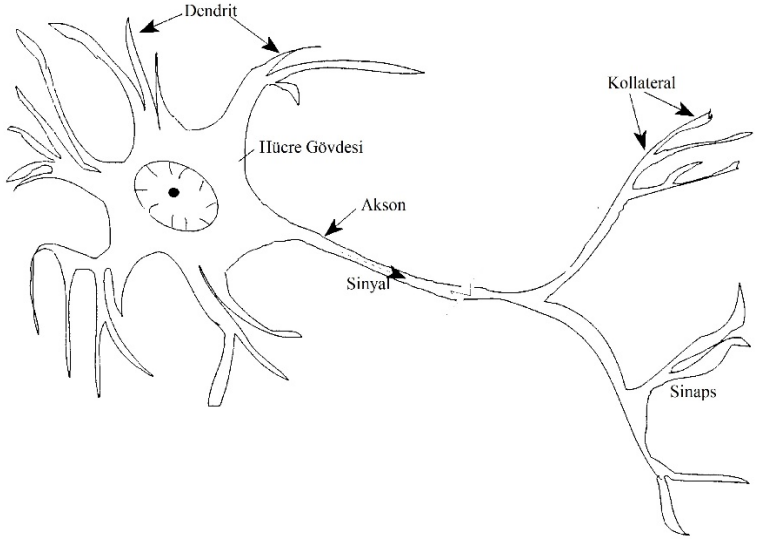
İnsan beyni, oldukça karmaşık, doğrusal olmayan ve paralel işlemler yapabilen bir bilgi işlem sistemidir. Günümüzde var olan en hızlı bilgisayarlardan çok daha hızlı bir şekilde, belirli hesaplamaları gerçekleştirmek için, nöron adı verilen yapısal bileşenleri düzenleme yeteneğine sahiptir. Doğumunda insan beyninin, dikkate değer bir yapısı ve "deneyim" olarak adlandırılan kavram sayesinde kendi davranış kurallarını geliştirme yeteneği vardır. En genel şekliyle NN, insan beyninin belirli bir

görevi veya işlevi yerine getirme şeklini modellemek için tasarlanmış bir makinedir [53]. Ağ genellikle elektronik parçalar kullanılarak uygulanır veya bilgisayar yazılımı ile simüle edilmektedir.

NN, deneyimsel bilgiyi saklamak için doğal bir eğilime sahip olan ve bu bilgiyi kullanılabilir kılan, büyük ölçüde paralel dağıtılmış bir işlemci olarak tanımlanmaktadır [53].

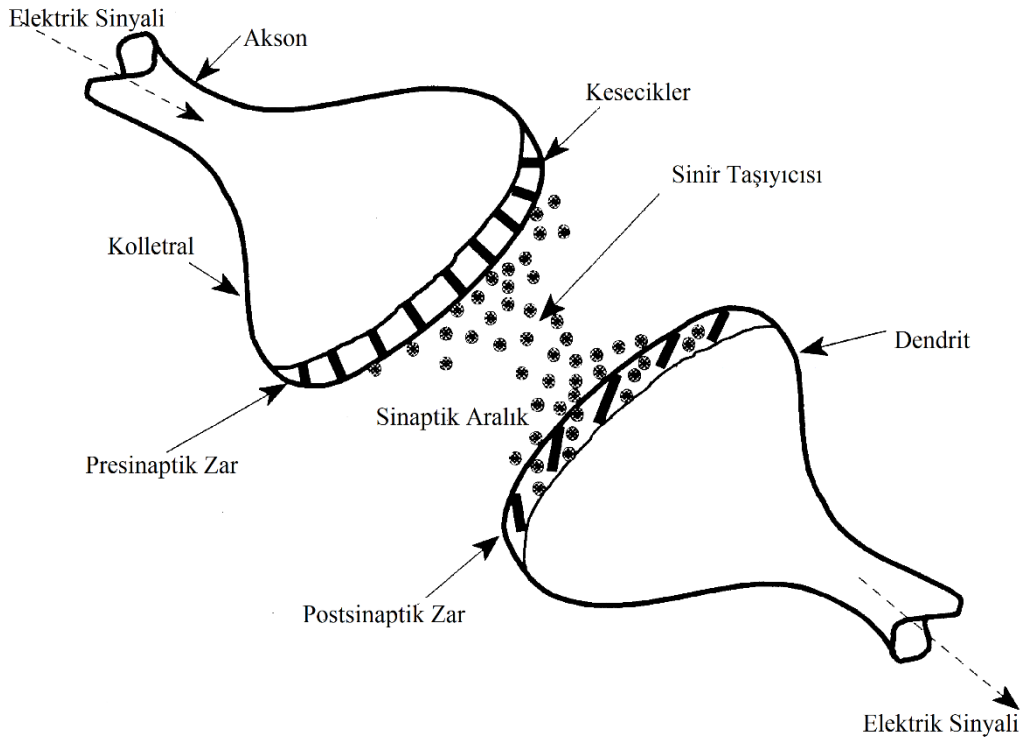
3.1.2. Biyolojik Nöron

İnsan sinir sistemi, vücuttaki konumlarına göre değişen tip ve uzunlukta milyonlarca nörondan oluşmaktadır. Şekil 3.1'de, üç ana fonksiyonel birim olan dendrit, hücre gövdesi ve akson ile basitleştirilmiş bir biyolojik nöronun yapısını göstermektedir [54]. Hücre gövdesi, kalıtım özellikleri hakkında bilgi içeren bir çekirdeğe ve nöronun ihtiyaç duyduğu materyali üretmek için kullanılan moleküler ekipmanı tutan bir plazmaya sahiptir. Dendritler, diğer nöronlardan sinyal almakta ve bunları hücre gövdesine iletmektedir.



Şekil 3.1. Biyolojik nöronun şekli [54].

Tipik bir nöronun dendritlerinin sinyal alma alanları toplamı yaklaşık olarak 0.25 mm^2'dir [55]. Kolonlara uzanan akson, hücre gövdesinden sinyalleri almakta ve komşu nöronların dendritlerine sinaps yoluyla taşımaktadır.

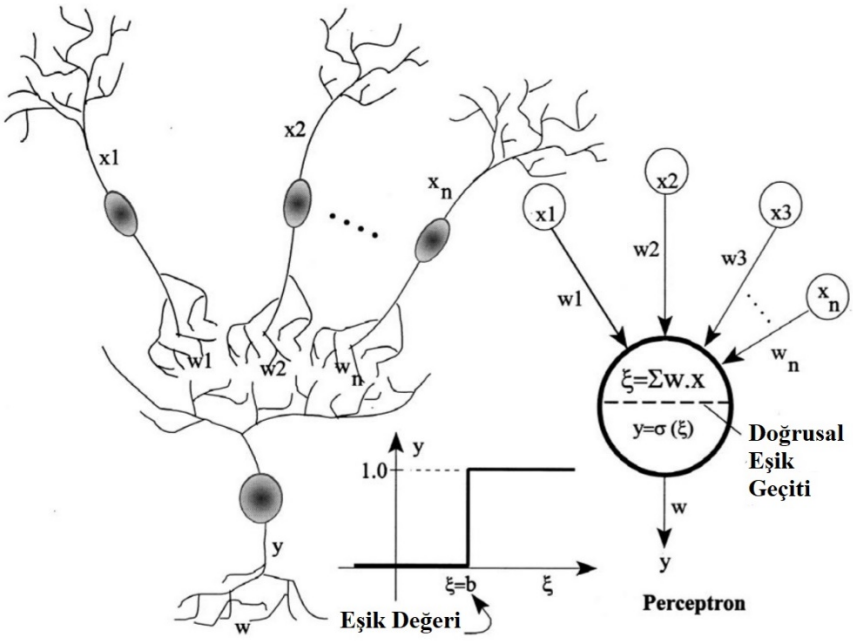


Şekil 3.2. İki biyolojik nöron arasındaki elektriksel sinyal transferi [54].

Bir alıcı nörondan geçen sinyalin miktarı, her bir beslenme nöronundan çıkan sinyalin yoğunluğuna, sinaptik kuvvetlerine ve alıcı nöronun eşik değerine bağlıdır. Şekil 3.2'de biyolojik iki nöron arasındaki sinyal transferi temsilen gösterilmektedir. Bir nöron çok sayıda dendrit ve sinapse sahiptir. Bu sayede nöron, aynı anda birçok sinyali alma ve aktarabilme becerisi kazanmaktadır. Bu sinyaller ya nöronun ateşlenmesine (nöronun aktif olma durumu) sağlamakta ya da engellemektedir. Bu basitleştirilmiş sinyal aktarım mekanizması, ANN'in temel birimlerinin yapısını ve işleyişinin temel adımlarını oluşturmuştur [54].

3.1.3. Yapay Nöron

1958'de Rosenblatt, karakter tanıma alanındaki problemleri çözmek için ve tek bir nörondan oluşan "Perceptron" adını verdiği yapıyı tanıttı [54]. Biyolojik nöronların mekanizmalarından elde edilen temel bulgular, basit yapay nöronların çalışmasını modellemek için ilk araştırmalara olanak sağlamıştır.



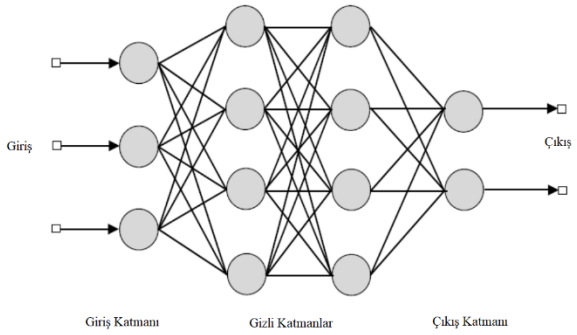
Şekil 3.3. Tek katmanlı perceptron içeren yapay bir nöronda sinyal toplama ve *n* adet nöron ile sinyal etkileşim benzeşimi [54].

Yapay bir nöron işlemcisi çevreden uyarıcı olarak girişler almaktadır ve bu girişleri özel bir yol ile birleştirmektedir. Şekil 3.3'te gösterildiği şekilde, yapay nöron bu sinyali, belirli bir eşik değerinin üstünde ise bir başka nörona ya da çevreye taşımaktadır. Birleştirilen değer sadece belirli bir eşik değerini aştığında nöron aktif hale gelmektedir (aynı zamanda ateşlemek olarak isimlendirilmektedir). Giriş değeri, nöron *(x)* ve ağırlıkların *(w)* çarpılması ile hesaplanmaktadır. *N* adet nöron için, hesaplama işlemi Eşitlik 3.1'de gösterilmektedir.

$$y = \begin{cases} 1, & eğer \sum_{i=1}^{n} w_i x_i \geq b \\ 0, & eğer \sum_{i=1}^{n} w_i x_i < b \end{cases} \tag{3.1}$$

3.1.4. Çok Katmanlı Yapay Sinir Ağları

Çok katmanlı sinir ağları (MLP) mimarisinde, nöronlar, Şekil 3.4'te gösterildiği gibi eş girdi verilerini paylaşan ve paralel bir düzene sahip olan çoklu katmanlara bölünmektedir [56].



Şekil 3.4. Nöronlar, aynı giriş verilerini paylaşan paralel nöron düzenine sahip çoklu katmanlar [56].

Bu mimariye gösterilebilecek en güzel örnekler, radyal temelli fonksiyonlar ve çok katmanlı algılayıcılar olarak bilinmektedir [56]. Bu mimariye sahip ağlar, gerçek verileri özel bir fonksiyon ile temsil etmek için yararlıdır.

3.1.5. Öğrenme Süreci

Yapay bir sistemde öğrenme, özel bir görev yerine getirmek için, içsel sürecin güncellenerek, dışsal uyarılara yanıt verme süreci olarak tanımlanmaktadır [54]. Bu sürece ayrıca, "eğitim" adı verilmektedir, çünkü öğrenme, ANN'de bulunan bağlantı ağırlıklarının iteratif olarak ayarlanması işlemidir [57]. Bu işlem, bağlantıların ağırlıklarının ayarlanması, bazı bağlantıların koparılması ya da oluşturulması, nöronların eşik değerlerinin değiştirilmesi gibi ağ mimarisindeki değişiklikleri içermektedir. Ağ, insan deneyimlerinin öğrenme sürecine etkisine benzer bir şekilde, bir başka deyişle, yenilemeli bir şekilde ANN öğrenimini gerçekleştirmektedir. Bu işleyiş göz önünde bulundurulursa, öğrenmenin olasılıksal olduğu ve öğrenilen görevden yola çıkılarak genelleme yapabildiği söylenebilir [54].

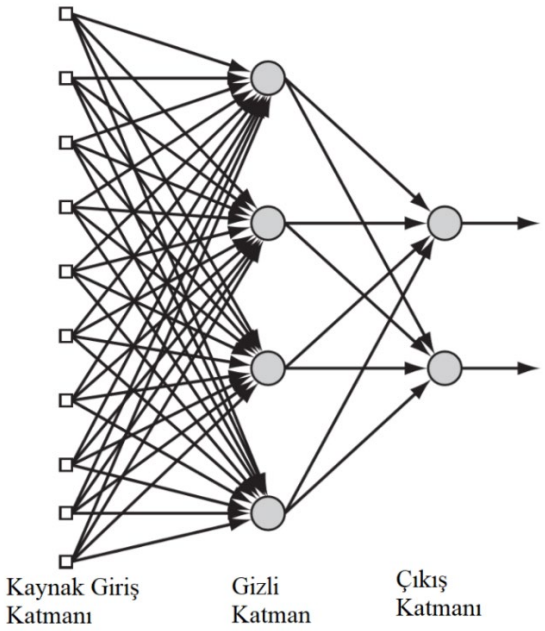
ANN'de öğrenme;

1. Denetimli
2. Denetimsiz
3. Takviyeli

olarak üçe ayrılmaktadır [57]. Denetimli öğrenme, bir NN'in hali hazırdaki çıktısı ile beklenen çıkış değeri arasındaki karşılaştırmaya dayanmaktadır. Genellikle tüm veriler üzerinde, oluşan çıkış değeri ile mevcut beklenen çıkış değeri arasındaki Ortalama Karesel Hata (MSE) fonksiyonunu değeri toplamının en aza indirilmesi olarak formüle edilmektedir. Denetimsiz öğrenme, sadece giriş verileri arasındaki ilişkiye dayanmaktadır. Öğrenme sonuçlarında, "doğru çıktı" hakkında bilgi bulunmamaktadır. Takviyeli öğrenme, istenilen çıktının tam olarak bilinmediği denetimli öğrenimin özel bir durumudur. Sadece gerçek çıktının doğru olup olmadığı bilgisine dayanmaktadır.

3.1.6. İleri Beslemeli Sinir Ağları

İleri Beslemeli Sinir Ağları (FFNN), bir veya daha gizli sigmoid nöron katmanına, son olarak bir doğrusal nöron çıkış katmanına sahip ağ olarak tanımlanmaktadır [58]. NN çıkışlarının kısıtlanılması istenildiğinde (0 ya da 1 gibi), çıkış katmanında sigmoid aktivasyon fonksiyonları kullanılabilir. Doğrusal olmayan aktivasyon fonksiyonlarına sahip çoklu nöron katmanları, ağın, giriş ve çıkış vektörleri arasındaki olasılıksal ilişkileri öğrenmesini sağlamaktadır [53].



Şekil 3.5. Bir gizli katman ve bir çıkış katmanıyla tam bağlı ileri beslemeli sinir ağı yapısı [53].

Ağın giriş katmanında bulunan düğümler (kaynak düğümleri), ikinci katmandaki (yani ilk gizli katman) nöronlara (hesaplama düğümleri) uygulanan giriş sinyalini (giriş vektörü) sağlamaktadır. İkinci katmanın çıkış sinyalleri, ağın üçüncü katmanına girdi olarak kullanılmaktadır. Benzer bir şekilde, ağın her katmanındaki nöronlar, giriş olarak yalnızca önceki katmanın çıkış sinyallerini içermektedir. Ağın çıkış (son) katmanındaki bulunan nöronlarda oluşan çıkış sinyal seti, ağın giriş (birinci) katındaki kaynak düğümler tarafından sağlanan aktivasyon düzeninin genel tepkisini temsil etmektedir.

m kaynak düğüm sayısı, h_1 birinci gizli katmandaki nöron sayısı, h_2 ikinci gizli katmandaki nöron sayısı ve q çıkış katmanındaki nöron sayısı olan bir FFNN, m-h_1-h_2-q ağı olarak adlandırılmaktadır [53].

Şekil 3.5'te gösterilerilen ağın, her katmanındaki her düğümün, bitişik ön katmandaki tüm nöronlarla bağlı olma durumuna "tam bağlı" ağ ismi verilmektedir. Bitişik ön katmandaki her düğüm ile bağlı olmama durumuna ise, "kısmi bağlı" ağ ismi verilmektedir.

3.1.7. BPNN

BPNN, en az bir geri besleme döngüsüne sahip olması ile kendisini FFNN'den ayırmaktadır [53]. Geriye doğru yayılma algoritması, ortalama hatalar karesi öğrenme algoritmasına benzerdir ve gradyan azalmaya dayanmaktadır, yani ağırlıklar hata ölçüsünün negatif gradyanı yönünde güncellenmektedir [59]. Geri yayılım algoritması, NN alanında önemli bir etkiye sahiptir ve birçok disiplinde yaygın olarak kullanılmaktadır [60]. Geri yayılım, sınıflandırma, fonksiyon yaklaşımı ve kestirim gibi çeşitli uygulamalar için kullanılmaktadır [59].

3.1.8. Aktivasyon Fonksiyonları

Bir nöron tarafından hesaplanan ağırlıklı toplam, artı sonsuz ve eksi sonsuz arasında değerler alabilmektedir [61]. Nöronun ürettiği bu toplam değerinin sınırlarının

bilinmemesi hangi durumlarda ateş edip etmeyeceği (aktif olup olmama durumu) problemini ortaya çıkarmaktadır [62].

Nöronun çıkışını "evet" ya da "hayır" gibi kararlaştırmak için transfer fonksiyonu adıyla da bilinen aktivasyon fonksiyonları kullanılmaktadır. Kullanılan fonksiyona bağlı olarak elde edilen toplam değer, 0 ile 1 ya da -1 ile 1 arasında eşlenir [56].

Aktivasyon fonksiyonları tür olarak, doğrusal ve doğrusal olmayan aktivasyon fonksiyonları olarak ikiye ayrılmaktadır. Doğrusal olmayan aktivasyon fonksiyonları birden çok katman istiflenmesini mümkün kılar ve türevlenebilir yapıları sayesinde ağın eğitimi sırasında gradyan temelli optimizasyon yöntemlerinin kullanılmasına olanak tanırlar [63]. Günümüzde kullanılmakta olan birçok aktivasyon fonksiyonu bulunmaktadır. Kullanım sıklığı dikkate alınarak bazıları aşağıda açıklanmaktadır.

3.1.8.1. Adım Aktivasyon Fonksiyonu

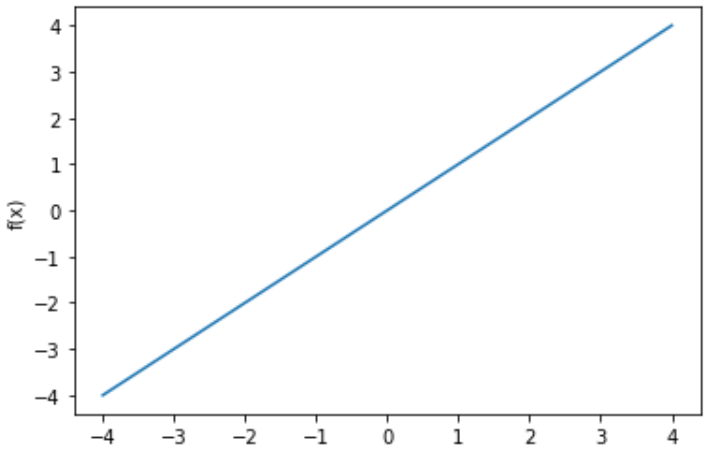
Nöronun belirlenen bir eşik değerinin üzerinde aktif olmasını, aksi tüm durumlarda deaktif olmasını sağlayan bir fonksiyon türüdür [64]. İkili bir sınıflandırma probleminde "evet" ya da "hayır" sonucu vererek çözüm üretmektedir. Birden fazla çıktı nöronunun olduğu sınıflandırma problemlerinde olasılıksal sonuçlar (%20, %30 gibi) üretemediği için kullanılması zordur. Eşitlik 3.2'de Adım fonksiyonu matematiksel olarak gösterilmektedir.

$$f(x) = \begin{cases} 1, & eğer\ x \geq 1 \\ 0, & eğer\ x < 0 \end{cases} \tag{3.2}$$

3.1.8.2. Doğrusal Aktivasyon Fonksiyonu

Aktivasyon fonksiyonu çıkışlarının, girişleri ile doğru orantılı olduğu ($y = xw$ gibi) düz bir çizgi fonksiyonudur. Tasarlanan ağ ne kadar çok istiflenmiş katmana sahip olsada, transfer fonksiyonu olarak doğrusal fonksiyonlar seçilmesi durumunda, son katmanda bulunan aktivasyon fonksiyonu giriş katmanın doğrusal bir fonksiyonu olmaktadır. Bu durum tüm ağın doğrusal katmanlı bir tek katmana denk olmasını

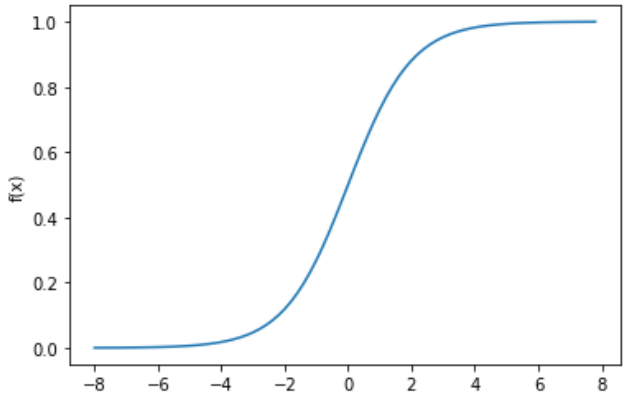
ortaya çıkartır. Doğrusal fonksiyonlar sadece çıktı katmanında kullanılırlar. Şekil 3.6'da, *F(x)* doğrusal fonksiyonuna ait grafik gösterilmektedir.



Şekil 3.6. Doğrusal aktivasyon grafiği [56].

3.1.8.3. Sigmoid Aktivasyon Fonksiyonu

Sigmoid düz ve türevlenebilen bir fonksiyondur [53]. Adım ve doğrusal fonksiyonlar üzerine en büyük avantajı doğrusal olmamasıdır. Aktivasyon fonksiyonu olarak sigmoid kullanan bir nöronun, çıkış değerleri doğrusal değildir [61]. Sigmoid fonksiyonu *0* ve *1* aralığında değerler üretmektedir. Şekil 3.7'de, 1 ve 0 arasında değer üreten Sigmoid grafiği gösterilmektedir. Sigmoid fonksiyonun sorunu, *-3* ve *+3* bölgelerinin ötesinde düz olmalarıdır. Bu düzleşme, o bölgelerde eğimin küçük olduğu anlamına gelmektedir.



Şekil 3.7. Sigmoid aktivasyon fonksiyonun grafiği [56].

3.1.8.4. Tanh Aktivasyon Fonksiyonu

Ölçeklendirilmiş sigmoid fonksiyonu olarak bilinmektedir [53]. Doğrusal olmaması sebebiyle ağın katmanlar halinde çalışmasını mümkün kılmaktadır. Tanh aktivasyon fonksiyonunun ürettiği çıktı değerleri *-1* ile *1* aralığındadır [66]. Tanh fonksiyonun eğimi, sigmoid fonksiyonun eğimine göre daha diktir. Tanh fonksiyonun avantajı, negatif girişlerin negatif eşlenmesidir.

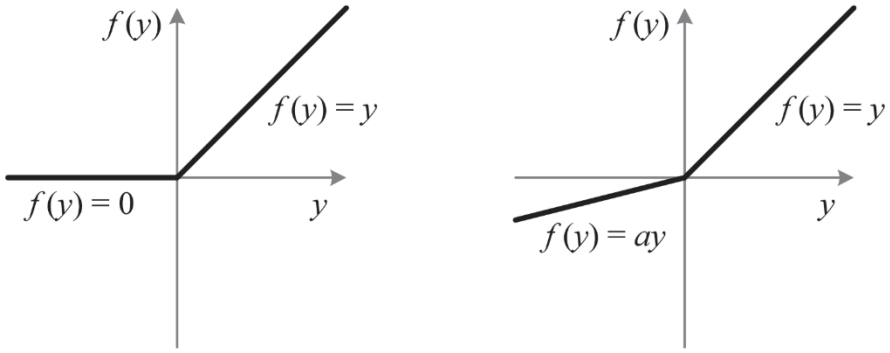
3.1.8.5. Rectified Linear Aktivasyon Fonksiyonu

Rectified Linear Aktivasyon Fonksiyonu (ReLU) fonksiyonu $f(y) = \max(0, y)$ olarak tanımlanmaktadır. ReLU, *0* ile artı sonsuz aralığında değerler üretmektedir [67]. ReLU fonksiyonundaki temel sorun, negatif değerli girişler için ANN modelinde çıkışların sıfır olarak üretilmesidir. Bu, ağın eğitim gücünü kısıtlamaktadır.

DNN, yapıları gereği çok fazla katmana sahiptirler [68]. Aktivasyon fonksiyonunun daha çok kullanımına neden olan bu yapı, daha basit matematik işlemler içermekte olan ReLU fonksiyonun kullanımını, sigmoid ve tanh fonksiyonlarına göre daha çok arttırmaktadır [69].

3.1.8.6. Leaky Rectified Linear Aktivasyon Fonksiyonu

Leaky Rectified Linear Aktivasyon Fonksiyonu (LReLU), ReLU fonksiyonun genişletilmiş bir sürümüdür [70]. ReLU işlevinde girdi değeri olan *x*'lerin sıfır olması durumda eğimin sıfır olduğu görülür ve nöronların ölü olduğu kabul edilir. Bu sorunun çözümü için LReLU fonksiyonu önerilmektedir. LReLU, *x*'in sıfırdan küçük olduğu değerleri sıfır tanımlamak yerine, *x*'in küçük bir doğrusal bileşeni olarak tanımlamaktadır [70].



Şekil 3.8. Relu ve LReLU [69].

Yatay çizginin kaldırılmasındaki temel avantaj, eğimin sıfır olmasının engellenmesidir. Bu, grafiğin sol tarafındaki eğimin sıfır olmasını engelleyerek, ölü nöronları ortadan kaldırmaktadır. Şekil 3.8'te Relu ve LReLU fonksiyonlarına ait grafikler gösterilmektedir. LReLU için negatif kısmın katsayısı sabit değildir ve uyarlanabilir bir şekilde öğrenilir.

3.1.8.7. Softmax Aktivasyon Fonksiyonu

Softmax bir sigmoid fonksiyonudur ve sınıflandırma problemlerinin çözümünde kullanılmaktadır [58]. Sigmoid fonksiyonu sadece iki sınıfı ortaya çıkarabilir. İkiden fazla sınıfı ele almamız gereken sınıflandırma problemlerinde sigmoid yeterli olmayacaktır [64]. Softmax fonksiyonu, her çıktı değerini *0* ile *1* arasında sıkıştır ve çıktılar toplamına böler. Bu, her girdinin belirli bir sınıfta olma olasılığı vermektedir [58].

3.1.9. Hata Fonksiyonları

Sinir ağlarının kullanımını sınırlayan temel faktörlerden biri, eğitilmiş bir ağın gerçek hayat kullanımında güvenilir sonuçlar üretmeye devam edeceğini göstermenin zorluğudur [71]. Bir ağın, eğitim sırasında kullanılan verilere benzer veriler sunulduğunda güvenilir sonuçlar verdiği, yeni veriler sunulduğunda ise, beklenen değer ve çıkış değeri arasında ciddi farklılık olduğu gözlenmektedir [71].

Hata fonksiyonları, ağın tutarsızlıklarını ölçmek için kullanılan önemli bir parçasıdır. Ağ modelinin sağlamlığı, hata fonksiyonunu sonucu ile ters orantılı olarak değişmekte olan ve negatif olmayan değer almaktadır. Ağın çözüm aradığı probleme göre seçimleri değişmekte olan ve en yaygın kullanılan hata fonksiyonları aşağıda açıklanmaktadır.

3.1.9.1. MSE

Bir NN eğitilmesi, bağlantı ağırlık değerlerinin ayarlanmasını içermektedir [58]. Ağın beklenen çıktı değeri ile elde edilen çıktı değeri arasındaki farkın kareleri toplanarak, ağın performansı gözlenmektedir. Bu yöntem, MSE olarak adlandırılır [58] ve Eşitlik 3.3'teki formül ile tanımlanmaktadır:

$$f = mse = \frac{1}{N}\sum_{i=1}^{N} {e_i}^2 = \frac{1}{N}\sum_{i=1}^{N} (t_i - a_i)^2 \tag{3.3}$$

Eşitlik 3.3'te [58], *t* beklenen değeri, *a* ise elde edilen çıkış değerini temsil etmektedir. MSE, fark değerlerinin karesini alarak negatif sonuç elde edilmesini engellemektedir. MSE'nin temel hedefi, beklenen değer ve elde edilen çıkış değerlerinin farklarının toplamını en aza indirgemektir [72].

3.1.9.2. Çapraz Entropi Hata Fonksiyonu (Cross-Entropy)

Sinir ağlarının eğitim süreci oldukça yavaş aşamalardan geçmektedir. Bu aşamalarda ortaya çıkan zaman kaybının engellenmesi için çapraz entropi hata fonksiyonun kullanılması önerilmektedir [73]. Cross-Entropy ikili sınıflandırma problemlerinde kullanılan bir hata fonksiyonudur [64].

BPNN'in hızlandırılması, çıkış birimlerinden beklenen ve elde edilen çıkış değerlerinin farklarının karelerini en aza indirgemek yerine, ağın tüm durumları için, Eşitlik 3.4'te verilen çapraz hata fonksiyonu minimize edilerek gerçekleştirilmektedir [73].

$$E_m = \frac{1}{m}\sum_{k=1}^{m}(t_{k-}y_k)^2 \tag{3.4}$$

Eşitlik 3.5'te E_m, hata miktarını en aza indirgemek için, her W_{jk} ağırlığı, ağırlıkları oranında E_m kısmi türevi ile güncellenmektedir.

$$\frac{\partial E_m}{\partial w_{jk}} = \partial.(y_k - t_k)y_k(1 - y_k).z_j \tag{3.5}$$

Çapraz entropi hata fonksiyonu kullanılarak, E_m'in w_{jk}'e göre kısmi türevi Eşitlik 3.5'te elde edilmiştir.

$$\frac{\partial E_m}{\partial w_{jk}} = \partial.(y_k - t_k).z_j \tag{3.6}$$

Bu yöntem, ağırlıkların güncellenmesi için geçen zamanı, beklenen değer ve gerçekleşen değer arasındaki farkla doğru orantılı hale getirerek, ağın performansını daha iyi bir hale getirmektedir [73]. Eşitlik 3.6'da beklenen değer ve gerçekleşen değer farkı matematiksel olarak gösterilmektedir.

3.1.9.3. MAE

Eğitilmiş bir NN'in başarısının ölçülebilmesinde öngörülen sonucun gerçek sonuç ile karşılaştırılması gerekmektedir [26]. Eşitlik 3.7'de, t beklenen değeri, a ise gerçek ölçüm değerini temsil etmektedir. Test veri setinde bulunan her ölçüm değeri için, gerçek ve önerilen değerler arasındaki farkın multak değerleri toplanarak, verisetinde bulunan ölçm sayısına bölünerşek MAE değeri elde hesaplanmaktadır.

$$f = mae = \frac{1}{N}\sum_{i=1}^{N}|t_{i-}a_i| \tag{3.7}$$

3.1.10. Başlatma Yöntemleri

Sinir ağlarında ağırlıkların başlangıç değerinin belirlenmesi, ağın eğitim hızını en fazla etkileyen yaklaşımlardan biri olarak kabul edilmektedir [74]. Optimal ağırlıkların belirlenmesi, ağın eğitim sürecinde gerçekleştirdiği iterasyon sayısını önemli ölçüde azaltacaktır.

3.1.10.1. Gauss ve Tekdüze Başlatma

NN'leri oluşturan nöron ağırlıkları simetrik bir şekilde küçük bir değer verilerek başlatılmaktadır. Temel fikir, nöronların başlangıç değerlerinin rastgele ve benzersiz olduğu, farklı günceleme adımları sonrasında yeniden hesaplanacakları ve ağın tam ağırlığını oluşturan bir ağırlık değeri bütününün bir parçası olacağıdır.

Bunun için, *ortalama = 0 ve sapma farkı = 0.1* olan (*-1,1*) değerleri arasında değişen Gaussian dağılımı kullanırız. Bu formülasyon ile her nöronun ağırlık vektörü çok boyutlu bir Gaussian'dan elde edilen rastgele bir vektör olarak başlatılır ve böylece nöronlar giriş alanında rastgele yönde işaret ederler.

Bu yaklaşımın temel problemi, çıktıların rastgele başlatılmış nöronlardan dağılımının, girdi sayısı ile birlikte artan bir varyansa sahip olmasıdır. Bu, Glorot başlatma yöntemi kullanılarak çözülmektedir.

3.1.10.2. Glorot Başlatma Yöntemi

Ağırlık ve bias değerlerini başlatmanın en yalın yolu, nöronlara tekdüze ve rastgele bir şekilde -0.01 ve 0.01 aralığında değerler vermektir. Bu yaklaşım özellikle ReLU aktivasyon fonksiyonu kullanan ağlarda iyi sonuçlar üretmemektedir [75]. Glorot başlatma yöntemi, DNN'lerde her bir ağırlığın $mean = 0.0$ değerli küçük bir Gaussian değeri ile başlaması ve varyans giriş ve çıkış yelpazasine bağlı kalması fikrine dayanmaktadır.

Glorot başlatma, sınırlandırılmış değerler arasında rastgele tekdüze dağıtımdan katman ağırlıklarının seçilmesi ile gerçekleştirilir.

$$\pm \frac{\sqrt{6}}{\sqrt{n_i + n_{i+1}}} \tag{3.8}$$

Eşitlik 3.8'de, n_i, ağa gelen bağlantıları veya katmana giriş yelpazesini, n_{i+1} ise bu katmandan çıkış yelpazesini yani giden ağ bağlantılarını temsil etmektedir.

3.1.10.3. Ortogonal Başlatma Yöntemi

Saxe vd. derin sinir ağlarında ağırlıkların nasıl başlatılacağı sorunu üzerine yaptıkları çalışmalarında, ağırlık matrisinin rastgele ortogonal bir matris olarak seçilmesini, yani $W^tW = I$ için bir kare matris olan W seçilmesini öne sürmüşlerdir [76].
Ortogonal matrislerin derin sinir ağları eğitimi için iki önemli özelliği vardır. Bunlar;

1. Normların korunmasıdır, yani $\|w_x\|_2 = \|x\|_2$,
2. Sütunlar (ve satırlar) birbirinin ortonormalidir, yani $w_i^T w_i = \delta_{ij}$, burada w_i, W'in i. sutununa atıfta bulunur.

Bu iki önemli özellikten birincisi, eğitimin başlangıcında, ağın genelinde normun sabit kalmasına yardım ederek patlayan/ kaybolan gradyan sorunun aşılmasına yardımcı olabilir. Benzer şekilde ikinci özellik, sezgisel bir şekilde ağırlık vektörlerine sahip olması sebebiyle, farklı girdi özelliklerinin öğrenmesini teşvik eder [77].

3.1.11 Optimizasyon Yöntemleri

DÖ çok büyük sayıda iterasyon içeren bir sürece sahiptir. Ağın hangi ağırlık kombinasyonunda başarılı olduğunu bulabilmek için farklı permütasyonları denememiz gerekmektedir [78]. Bu kabul, modelin eğitim zaman maliyetinin düşürülmesini önemli kılmaktadır.

3.1.11.1. Stokastik Gradyan İnişi Optimizasyon Yöntemi

Stokastik Gradyan İnişi (SGD), fonksiyonun minimumunu bulmak için bir optimizasyon yöntemidir. Sinir ağlarında, nöronların ağırlığını geri yayılım yolu ile bulmak için kullanılmaktadır. SGD, uygulanmasındaki basitlik ve büyük sayıda eğitim örneğine sahip ağlardaki hızı sebebiyle makine öğrenme problemlerinde en çok kullanılan optimizasyon yöntemidir [78]. SDG'ler için en büyük dezavantajlar ise, öğrenme oranı, yakınsama kriterleri gibi optimizasyon parametrelerinin manuel olarak ayarlanması gerekmesidir. SGD'in iterasyonlarda ağırlığı yeniden hesaplaması için kullnılan hesaplama Eşitlik 3.8'de gösterilmektedir.

$$w_{t+1} = w_t - \alpha \frac{\partial L}{\partial w_t} \tag{3.9}$$

Eşitlik 3.9'de, *t*, zaman adımı, *w*, güncellenmek istenen ağırlık, α, öğrenme oranı, $\partial L / \partial w$, kayıp fonksiyonu L'in eğimini temsil etmektedir.

3.1.11.2. Nesterov Hızlandırılmış Gradyan Yöntemi

Nesterov Hızlandırılmış Gradyan (NAG) yöntemi, momentum yöntemine birçok yönden benzerlik gösteren gradyan azaltımından ibaret gibi görünsede, klasik momentum yöntemi ile tamamen aynı değildir [79]. Bu farkı yaratan güncelleme, öngörülen gradyanların hesaplamasında üstsel hareketli ortalama olan V'nin kullanılmasıdır.

$$w_{t+1} = w_t - \alpha V_t \tag{3.10}$$

$$V_t = \beta V_{t-1} + (1 - \beta) \frac{\partial L}{\partial w_*} \tag{3.11}$$

V'in başlangıç değeri 0 olarak başlatılmaktadır. Eşitlik 3.10'daki V_t, tahmini gradyandır. Eşitlik 3.11'da *V* değerinin hesaplanması gösterilmektedir. Bu değer, önceki *V* değerini kullanarak bir adım önceden Eşitlik 3.12'de görüldüğü gibi elde

edilebilir. Bunun anlamı, *t* zaman adımı için geri yayılım gerçekleştirmeden önce ileri yayılım gerçekleştirmek zorunda olduğumuz anlamına gelir.

$$w^* = w_t - \alpha V_{t-1} \tag{3.12}$$

Bu işlem şu şekilde gerçekleştirilmektedir;

1. Önceki hız değeri kullanılarak, mevcut *w* ağırlığı, yeni bir w^* değeri ile güncellenmektedir.
2. Tahmini ağırlık kullanılarak ileri yayılım gerçekleştirilir.
3. $\partial L/\partial w_*$ işlemi ile tahmini gradyan bulunur.
4. Elde edilen değerlere göre, *V* ve *w* hesaplanır [80].

Genellikle *β,* 0.9 varsayılan değerini almaktadır.

3.1.11.3. ADAM Optimizasyon Yöntemi

Adam, King ve Ba tarafından 2014 yılında önerilmiş momentum ve RMSprop'un iki temel hesap farklılığı gösteren bir kombinasyonudur [81]. Bunlardan ilki, Momentumda olduğu gibi, gradyanın üstsel hareketli ortalaması *V*, gradyanın tamamlayıcısıdır. İkincisi, RMSprop'da olduğu gibi, uydurulan gradyanın üstsel hareketli ortalaması, *S*'in kare kökü, öğrenme oranını, α'i böler.

$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{\hat{S}_t + \epsilon}}.\hat{V}_t \tag{3.13}$$

$$\hat{V}_t = \frac{V_t}{1 - \beta_1^t} \tag{3.14}$$

$$\hat{S}_t = \frac{S_t}{1 - \beta_2^t} \tag{3.15}$$

Eşitlik 3.13, bias düzeltmeleridir, $\widehat{V_t \; ve \; S_t}$ değerlerinin hesaplamaları Eşitlik 3.14, Eşitlik 3.15, Eşitlik 3.16 ve Eşitlik 3.17'da gösterilmektedir ve

$$V_t = \beta_1 V_{t-1} + (1 - \beta_1) \frac{\partial L}{\partial w_t} \tag{3.16}$$

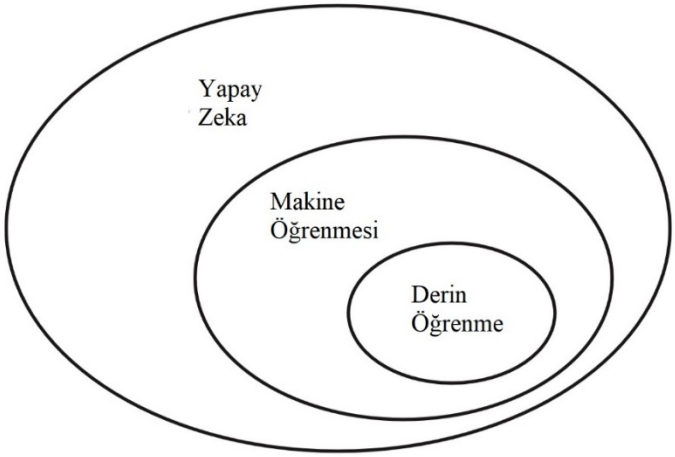
$$S_t = \beta_2 S_{t-1} + (1 - \beta_2) \left[\frac{\partial L}{\partial w_t}\right]^2 \tag{3.17}$$

V ve *S*'in başlangıç değeri, 0'dır. Yazarlar tarafından varsayılan değerler, $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 10^{-8}$'dir [81].

3.2. DERİN ÖĞRENMEYE GİRİŞ

Son birkaç yıl içerisinde AI yoğun ilgi gören çalışma konularından biri haline gelmiştir [82]. AI, DÖ ve makine öğrenmesi arasında sıkı bir ilişki bulunmaktadır. AI, makine öğrenmesi ve DÖ'yi kapsayan genel bir alandır, aynı zamanda herhangi bir öğrenmeyi içermeyen daha birçok yaklaşımı da içermektedir. Şekil 3.9'da bu ilişkilerin kapsayıcılıkları gösterilmektedir.

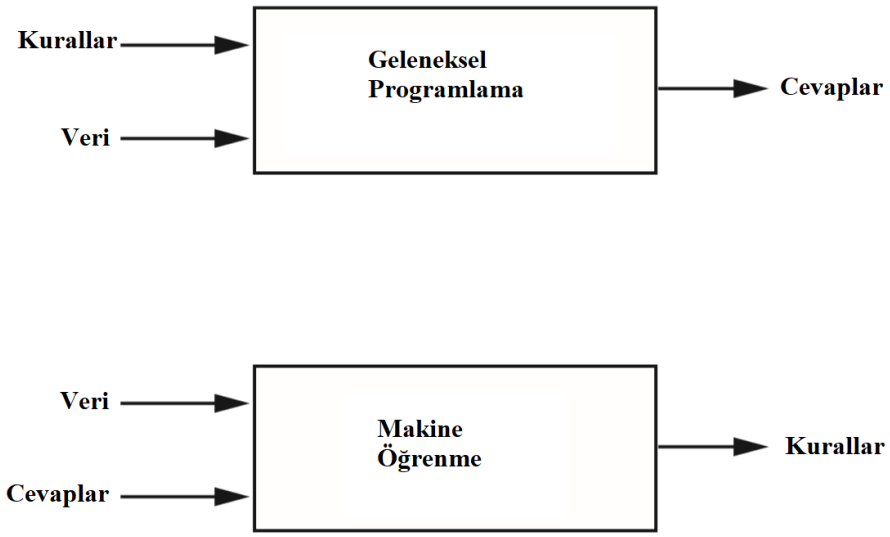
AI, insanlar tarafından gerçekleştirilen görevleri otomatikleştirme çabası olarak tanımlanmaktadır [51]. 1950'li yıllarda üzerine çalışmalar başlamıştır. Oldukça uzun bir süre boyunca, pek çok araştırmacı tarafından, insan düzeyindeki düşünebilen bir AI'in, programcıların bilgiyi manipüle etmek için yeterince geniş bir açık kurallar setine sahip olmasıyla sağlanabileceğine inanmıştır. Bu yaklaşım, sembolik yapay zeka olarak bilinmektedir ve 1950 ile 1980 yılları arasında baskın paradigma olmuştur [82]. Bu yaklaşım, görüntü tanıma, dil işleme, konuşma tanıma gibi problemlerde yetersiz olması sonucunda, ortaya yeni bir yaklaşım olan makine öğrenmesini çıkartmıştır [51].



Şekil 3.9. Yapay Zeka, Makine Öğrenmesi ve DÖ İlişkisi [82].

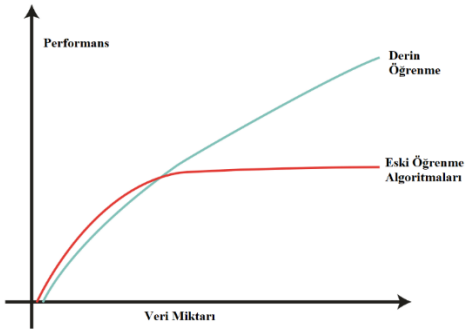
Makine öğrenmesi geçtiğimiz son on yıl içinde büyük bir popülerlik kazanmıştır. Temel olarak makine öğrenmesi, ham verilerden çeşitli algoritmalar kullanarak bilgiler almak ve bir model oluşturmak olarak tanımlanmaktadır [52]. Bu ortaya çıkan model, henüz modellemediğimiz diğer veriler hakkında bir çıkarım yapmak için kullanılmaktadır. Bu yeni paradigma Şekil 3.10'da temsili olarak gösterilmektedir.

Makine öğrenmesi teknolojisi, web aramalarından sosyal ağlarda içerik filtrelemeye, e-ticaret web sitelerinde kullanıcılara içerik önerilerinden kamera ve akıllı telefonlar gibi tüketici ürünlerine kadar hayatımızın birçok alanında giderek daha fazla yer almaktadır [83]. Makine öğrenim sistemleri, görüntülerdeki nesneleri tanımlamak, konuşmayı metne dönüştürmek, haber öğelerini, yayınları veya ürünleri kullanıcıların ilgi alanlarına göre eşleştirmek ve ilgili arama sonuçlarını seçmek için kullanılabilmektedir. Giderek artan şekilde, bu uygulamalar DÖ denilen bir teknikten faydalanmaya başlamıştır.



Şekil 3.10. Makine öğrenme yaklaşımı [82].

DÖ, metin, imge ve konuşma alanlarında son derece faydalı olduğu kanıtlanmış geniş bir makine öğrenme alt alanıdır [84]. DÖ altında uygulanmakta olan algoritmalar, insan beynindeki nöronlar ve algılayıcılar arasındaki ilişki ile benzerliklere sahiptir. DÖ yaklaşımlarının, bilgisayarla görme, doğal dil işleme, konuşma tanıma, görüntü oluşturma gibi birçok uygulaması bulunmaktadır.



Şekil 3.11. Veri miktarı ile veri bilimi tekniklerinin başarı değişimi [84].

DÖ algoritmalarının çoğu, ANN kavramına dayanmaktadır ve bu tür algoritmaların günümüz dünyasında eğitimi, bol miktarda veri ve yeterli hesaplama kaynaklarının bulunmasıyla daha kolay hale getirilmiştir [84]. Eğitim kümesinde kullanılan verinin büyümesi ile DÖ modellerinin performansı gelişmeye devam etmektedir. Şekil

3.11'de veri artışının, veri biliminde performansa etkisi temsili olarak gösterilmektedir.

3.3. DERİN ÖĞRENME TANIMI

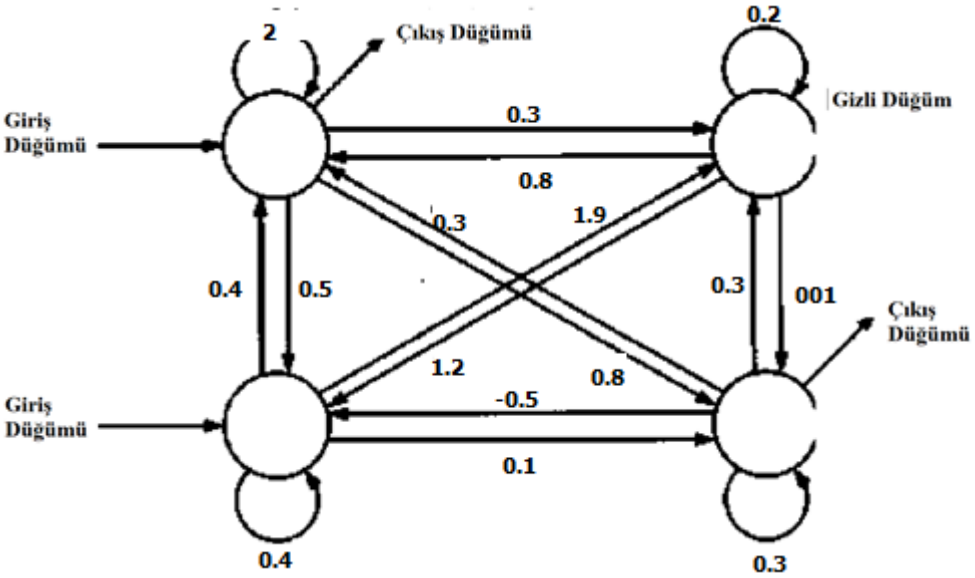
DÖ, makine öğrenmenin belirli bir alt alanıdır: sayıca fazla ardışık katmanın, belirli iterasyonda ve git gide daha iyi öğrenmesi üzerine odaklanan, verilerden bir öğrenme temsili elde etmeyi amaçlayan yeni bir bakış olarak tanımlanmaktadır [51].

DNN birden çok doğrusal olmayan gizli katman içermektedir ve bu mimari sayesinde ağ, girişleri ve çıkışları arasında çok karmaşık ilişkileri öğrenebilen bir beceri elde eder [85]. Bu katmanlar, bir biri üzerine yığılmış ANN adı verilen katmanlar ile yapılandırılmıştır [82]. NN terimi nörobiyolojiye atıfta bulunmaktadır, ancak DÖ'deki merkezi kavramların bir kısmı insan beyninden ilham alarak gelişmiş olsa da, DÖ modelleri bire bir beyin modeli değildir [82].

Otomatik özellik çıkarımı, DÖ'nin geleneksel makine öğrenimi algoritmalarına karşı sahip olduğu büyük avantajlardan bir tanesidir [52]. Özellik çıkarımı, ağın, bir veri kümesinin sahip olduğu özellikler içerisinden hangilerinin bu verileri güvenilir bir şekilde etiketlemek için gösterge olarak kullanılacağına karar verme sürecidir [52].

3.4. TAM BAĞLANTILI DERİN AĞLAR

Tam bağlantılı sinir ağları, "dense" ismiylede bilinmektedir. Her düğümün, kendinden sonra bulunan katmandaki tüm düğümler ile bağlantılı olduğu ağ mimarisi olarak tanımlanmaktadır [59]. Bağlantı ağırlıkları, pozitif (uyarı rolünde), negatif (engelleyici rolünde) ya da sıfır değerlerine sahip olabilirler. Şekil 3.12'te farklı ağırlık değerlerine sahip tam bağlantılı bir ağ gösterilmektedir.



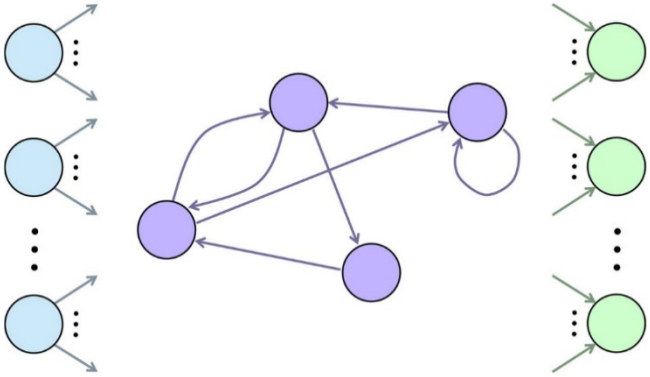
Şekil 3.12. Tam bağlantılı asimetrik sinir Ağı [59].

Bilinen diğer ağ mimarileri, ağırlık değerleri sıfıra ayarlanarak tam bağlantılı bir ağdan dönüştürülen özel durumlar olarak görülmektedir. Bu ağ mimarisi, genelliği ve kavramsal basitliğine rağmen, çok sayıda parametre içermesinden dolayı nadiren kullanılmaktadır [59]. Tam bağlı ağlar biyolojik olarak makul görülmemektedir, çünkü sinir hücreler kendileri ile uzaktaki nöronlarla çok az sinaps bağlantısına sahiptir.

3.5. RNN

Yineleyen sinir ağları ilk kez 1980'li yıllarda tanıtılmıştır, bu ağlar yakın dönem içinde gelişen donanım yenilikleri sayesinde yeniden popülerliğini kazanmıştır [86]. Yinelenen sinir ağları, en az bir geri besleme döngüsüne sahip olma şartı ile kendini FFNN'dan ayırmaktadır [53]. Şekil 3.13'de gösterilmekte olan tekrarlayan bir NN modelinde, bir nöronun birden çok bağlantıya sahip olabildiği, aynı zamanda nöronun kendi ile aynı katmandaki diğer nöronlar ile bağlantılı olabileceği görülmektedir.

Tamamen Yineleyen bir NN, r adet nörondan oluştuğu varsayıldığında, r^2 kadar bağlantıya sahip olduğu kabul edilir [86].



Şekil 3.13. Yineleyen ağlarda nöronların bağlantıları [86].

RNN, sıralı bilgilerden yararlanmak ve bu bilgilerde varolan desenleri öğrenmek için geliştirilmiş bir NN modelidir [87]. Bu ağ mimarisinin, bir diziye ait tüm elemanlar için aynı görevi gerçekleştirmesi beklenmektedir. Herhangi bir konuşma dilinde, kelimelerin sıra bağımlılıklarından dolayı RNN'ler, dil işleme problemlerinde çok kullanışlı olmuştur [87].

3.5.1. Yineleyen Sinir Ağları Eğitim Problemleri

Sinir ağlarında belirli bir aktivasyon fonksiyonu kullanan daha çok katman eklediğinde, kayıp fonksiyonlarında gradyan sıfıra yaklaşmakta ve ağın eğitimi zorlaşmaktadır. Sigmoid fonksiyonu gibi bazı aktivasyon fonksiyonları, 0 ve 1 arasında dar bir giriş alanında aldıkları değerler sonucunda, çıkışta büyük değişikliklere neden olabilir. Bu sebeple türevleri küçük olmaktadır. Sıfıra yaklaşan türev eğitimin gerçek başarıya ulaşmasını engellemektedir.

3.5.1.1 Kaybolan ve Patlayan Gradyan Problemi

Geri yayılım algoritmasından faydalanmakta amaç, beklenen çıktı ve modelin ürettiği çıktı değerleri arasındaki farkın tespit edilerek, hatanın hesapedilebilesidir. Hesaplanan hata kullanılarak, ağırlık değerlerinde güncellemeler gerçekleştirilir. Ağırlıkların hesaplanabilmesi için geri yayılma sürecinde, kısmi türev yardımı ile gradyan değerleri hesaplanırken, kaybolan gradyanlar ortaya çıkmaktadır (Vanishing Gradients) [84]. Aktivasyon fonksiyonu olarak sigmoid kullanan bir ağ, türev sonucu

bir değerinden az olacağı için, zamanla türev güncellenmeyecek bir küçük değere ulaşacak ve ağ öğrenmeyi durduracaktır.

Bu problem, derin sinir ağları ve çok tekrar eden sinir ağlarında, fazla sayıda geri yayılım işlemi gerektirdiği için sıklıkla görülmektedir [82].

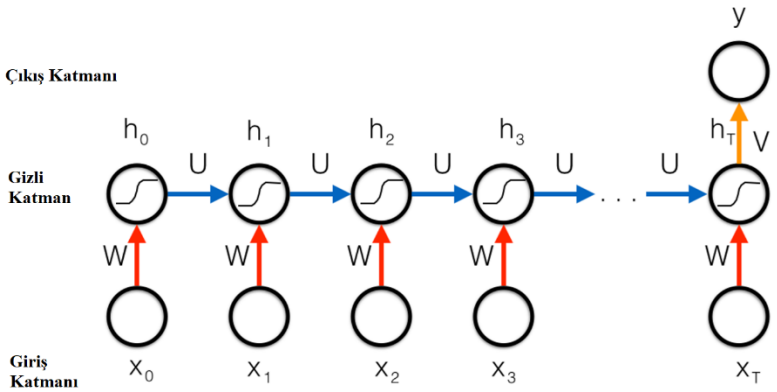
ReLU aktivasyon fonksiyonunun, sigmoid aktivasyon fonksiyonu yerine kullanılması kaybolan gradyan problemini aza indiren bir yaklaşım olsada, LSTM veya GRU mimarileri en popüler çözümler olarak kullanılmaktadır [86].

3.5.2. Basit Yineleyen Ağ

Basit Yineleyen bir ağda (SRN), nöronlar kendileri, aynı katmandaki diğer nöronlar veya önceki katmanlardaki nöronlar ile bağlantı içerisindedir. Bu bağlantılar, hem ileri hem de geri yönde bilgi akışını sağlamaktadır. Böylece ağ dinamik bir hafıza kazanmaktadır [54].

FFNN'de girdiler, aralarında hiçbir ilişkiye bakılmaksızın bağımsız olarak işlenir [51]. Bu tür FFNN'lerde, bir dizi işlenmesi için, bir kerede, yani bir vektöre dönüştürülerek ağa gösterilmelidir. Buna karşın, gerçek hayat örneğinde bir cümle okunurken, daha önce gelen kelimeler hafızada tutularak yeni gelen kelime işlenmektedir. Bu, cümlede aktarmak istediğimiz anlamı akıcı bir şekilde sunabilmemizi sağlamaktadır. Biyolojik zeka, geçmiş bilgilerden inşa edilmiş ve işlediklerinin içsel bir modelini korurken, sürekli olarak yeni bilgiler ile onu güncelleyerek, bilgiyi adım adım işlemektedir. RNN, basitleştirilmiş bir sürüm olarak bu çalışma prensibini benimsemektedir. Dizi elemanlarını her iterasyonda girdi olarak kabul ederek, yeni bilgiyi ortaya çıkartan bir model sunmaktadır.

Bir teknoloji şirketine ait hisse fiyatlarının tahmin edilmesi problemi ile tekrarlayan sinir ağları açıklanmaya çalışılacaktır. Böyle bir problemde, x yani giriş değerleri [x_0, x_1, ... x_t], geçmiş tarihli günlük hisse fiyatları, y yani tahmin ise, bugünkü hisse fiyatı olarak kabul edilmektedir.



Şekil 3.14. Tahmin problemi için yineleyen sinir ağı yapısı [88].

Şekil 3.14'de, x_0, x_1,...x_t bugüne kadar şirkete ait hisse fiyatlarını temsil etmektedir. h_0, h_1, ...h_t, tekrarlayan ağa ait gizli durumlardır ve her çember, bir katmanı temsil etmektedir. RNN'de temel olarak üç parametre seti bulunmaktadır [88]: giriş katmanından gizli katmana ağırlıklar (w), gizli katmandan gizli katmana ağırlıklar (u), ve gizli katmandan çıkış katmanına ağırlıklar (v). Bu üç parametre değeri paylaşılmaktadır ve bu paylaşma özelliği, ağımızı değişken boyutlu girdiler için uyumlu bir hale getirmektedir. Özyinelemeli bir şekilde hesaplama matematiği Eşitlik 3.18'de gösterilmektedir. Eşitlik 3.19, gizli katmanların hesaplanma matematiğini göstermektedir. h_0 için, önce katman olmamasından dolayı hesaplama başlangıçta Eşitlik 3.20'ta verildiği gibi gerçekleştirilir.

$$f(x) = Vh_T \tag{3.18}$$

$$h_t = \sigma(Uh_{t-1} + Wx_t), for\ t = T, \dots 1 \tag{3.19}$$

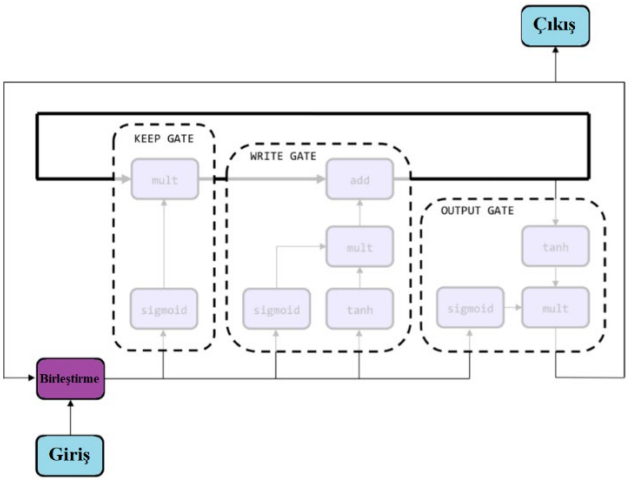
$$h_0 = \sigma(Wx_0) \tag{3.20}$$

Uygun ağırlıkları elde etmek için maliyet fonksiyonunu $(y - f(x))^2$ en aza indirilmelidir. RNN'de gradyanını hesaplamak için tekrar geri yayılım kullanabiliriz. Bu durumda algoritma, özel bir isim alarak, "Zaman içinde geri yayılım (BPTT)" olarak adlandırılmaktadır [88].

3.5.3. LSTM

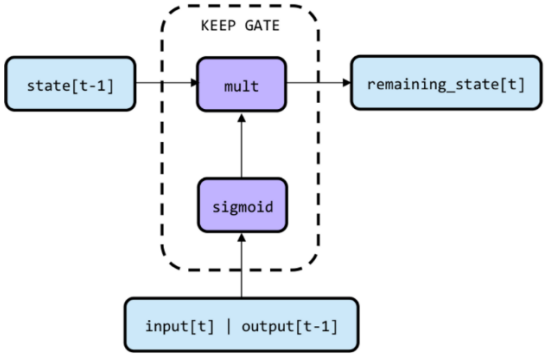
Sepp Hochreiter ve Jürgen Schmidhuber, kaybolan gradyanlar problemine çözüm olarak uzun kısa dönem ağ mimarisini ortaya çıkartmıştır [89]. Bu ağ mimarisinin arkasındaki temel ilke, ağın önemli bilgileri çok sayıda iterasyon içerisinde geleceğe güvenilir bir şekilde iletmesine dayanmaktadır [86].

Şekil 3.15'te görüldüğü gibi, LSTM ünitesi birkaç ana bileşenden oluşmaktadır. LSTM mimarisinin temel bileşenlerinden biri, şekil ortasındaki kalın çerçeve ile temsil edilen, tensör olarakta bilinen, bellek hücresidir. Şekil 3.15'te tensöre ait mimari gösterilmektedir. Birçok iterasyon boyunca, faydalı bilgileri saklamak üzere tasarlanmış ağa ait bu bellek hücresi, zaman içinde öğrendiği kritik bilgileri saklar [86].



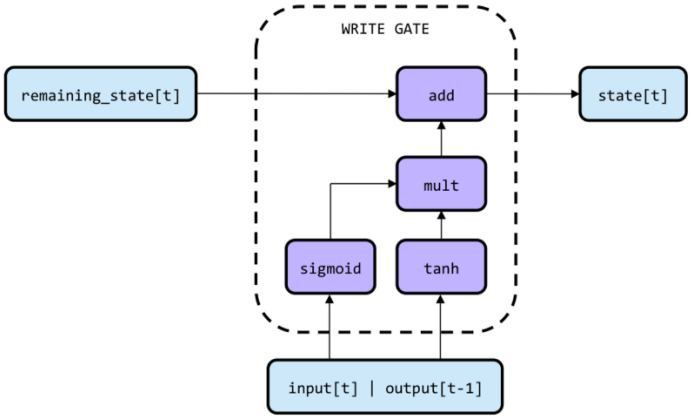
Şekil 3.15. Tensör ve işlem düzeyinde gösterilen LSTM birimine ait mimari [86].

Her zaman adımında, LSTM ünitesi yeni bilgiler ile üç farklı aşamada tensör hücresini değiştirmektedir. İlk olarak, ünite bir önceki hafızanın ne kadarını tutacağını belirlemektedir. Bu, "keep gate" olarak tanımlanmaktadır.



Şekil 3.16. Bir LSTM birimine ait Keep Gate mimarisi [86].

Keep gate'in temel fikri basittir ve ilkel mimari Şekil 3.16'da gösterilmektedir. Önceki zaman adımındaki hafıza durum tensörü bilgi bakımından zengindir, ancak bu bilgilerin bir kısmı eskimiş olabilir ve bu nedenle silinmeleri gerekebilir. Bit sensöründe konum bir ise, bellek içindeki konumun halen geçerli olduğu ve saklanması gerektiği anlamına gelmektedir. Konum sıfır olması durumu, tensör hücresindeki konumun artık alakalı olmadığı ve değiştirilmesi gerektiği anlamına gelmektedir. Bit tensörü, bu zaman adımı girişi ve LSTM ünitesinin bir önceki zaman adımının çıktısını çizer ve elde edilen tensörü bir sigmoid katmana getiririz.

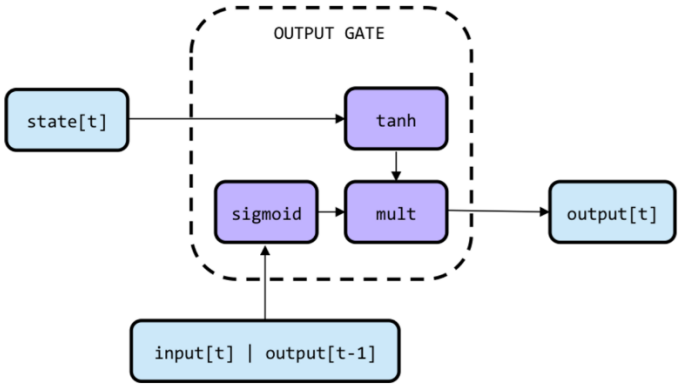


Şekil 3.17. Bir LSTM birimine ait Write Gate mimarisi [86].

Bir sigmoidal nöron, sıfıra çok yakın ya da bire çok yakın bir değer üretmektedir. Bu kural ölçüsünde, sigmoidal katmanın çıktısı, bit tensörün çalışma yaklaşımına yakındır. Bu sebeple, keep gate tamamlamak için bu fonksiyon kullanılmaktadır [86].

LSTM ünitesinin bilgi yazma alanı “write gate” ismi ile bilinmektedir. Şekil 3.17’de mimarisi verilen bu ünite, iki ana bileşenden oluşmaktadır. Bu bileşenlerden ilki, hangi bilgilerin tensöre yazılacağının bulunmasıdır. Bu, tanh katmanı tarafından hesaplanmaktadır. İkinci bileşen, hangi bilgilerin yazılmadan atılacağının bulunmasıdır. Bu seçimi keep gate’de kullandığımız aynı strateji ile gerçekleştiririz. Bit vektörü ara tensör ile çarpılır ve LSTM için yeni durum vektörü oluşturulur.

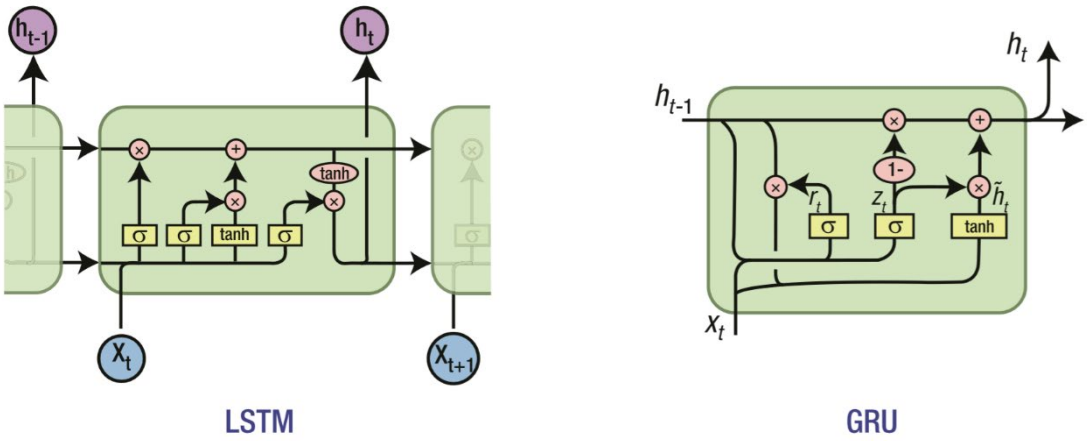
Output Gate mimarisi Şekil 3.18’de gösterilmektedir. Write Gate olarakta benzer bir yapı kullanılmaktadır. Tanh katmanı, durum vektöründen bir ara tensör oluşturur. Sigmoid katmanı, mevcut giriş ve önceki çıkışı kullanarak bit tensör maskesi oluşturur. Bu ara tensör, son çıktının elde edilmesi için bit tensörü ile çarpılır.



Şekil 3.18. Bir LSTM birimine ait Output Gate mimarisi [86].

3.5.4. GRU

Günümüzde kullanılan çok sayıda LSTM varyasyonu vardır. Bunların en sık kullanılanı kapı yenileyen birim yani GRU’dur [84]. GRU ve LSTM mimarileri Şekil 3.19’da karşılaştırılmaktadır. GRU, unutma ve giriş kapılarını birleştirerek bir güncelleme kapısı oluşturur ve çıktının üretilme biçimnde değişiklik yapmar. Standart LSTM modellerine kıyasla daha az karmaşıklığa sahiptir [90].
LSTM’nin daha büyük veri setleri için daha iyi, GRU’nun daha küçük veri setleri için daha iyi çalıştığı gözlenmiştir [84].



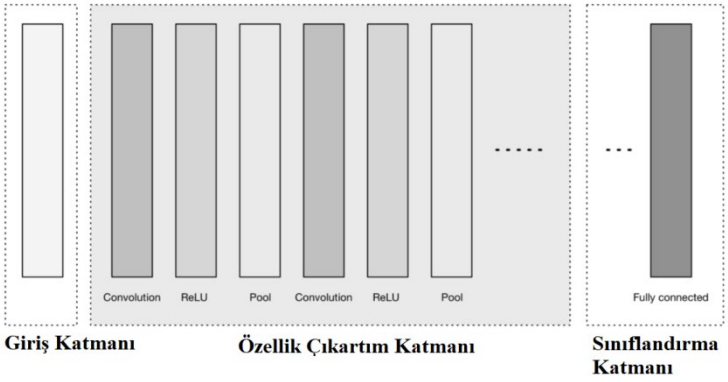
Şekil 3.19. LSTM ve GRU mimarileri [84].

3.6. CNN

Bir CNN'in amacı, verilerdeki daha yüksek dereceli özellikleri konvolüsyonlarla öğrenmektir. Resimlerle nesne tanımaya ve resim sınıflandırma problemlerine çok uygundurlar [52]. Yüzleri, bireyleri, sokak işaretlerini ve görsel verinin diğer birçok yönünü tanımlayabilme yeteneğine sahiptirler. Mimari açıdan, CNN'ler çok katmanlı algılayıcılardan ilham almaktadır. CNN, bitişik ağ katmanların nöronları arasında yerel bağlantı kısıtlamaları uygulayarak, yerel mekansal korelasyondan yararlanmaktadır [87]. CNN'in temel odağı, konvolüsyon operasyonu yoluyla verilerin işlenmesidir. Herhangi bir sinyalin başka bir sinyale çevrilmesi ile sinyal hakkında orijinal sinyalin kendisi hakkında sunduğu bilgiden daha fazlasını açığa çıkaran üçüncü bir sinyal üretilmektedir [91].

3.6.1. CNN Mimarisi

CNN'ler, girdi katmanından verilen veriyi, tüm bağlı katmanlardan geçirerek, çıktı katmanı tarafından verilen bir sınıf puanına dönüştürmektedir [52]. CNN mimarisi birçok varyasyona sahip olsada, temel olarak Şekil 3.20'e dayanmaktadır.



Şekil 3.20. Genel bir CNN mimarisi [52].

Şekil 3.20 üç ana grubu tasvir etmektedir:

1. Giriş katmanı (Input layer)
2. Öznitelik bulma katmanı (Feature-extraction (learning) layers)
3. Sınıflandırma katmanları (Classification layers)

Giriş katmanı, genel olarak görüntünün boyutlarından oluşan (*genişlik x yükseklik*) üç boyutlu girişi kabul eder ve renk kanallarını temsil eden bir derinliğe (genellikle RGB renk kanalları için üç) sahiptir [91].

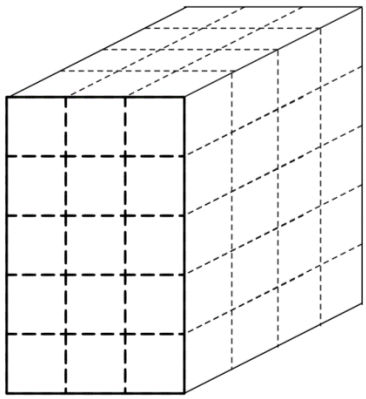
Öznitelik bulma katmanları, dizinin genel bir yinelenen düzenine sahiptir:

1. Evrişim katmanı: ReLU aktivasyon fonksiyonu, konvolüsyonel katman olarak tanımlanmaktadır.
2. Havuz katmanı: Bu katmanlar, görüntülerde bir takım öznitelikler çıkartır ve aşamalı olarak ileri seviyeli öznitelikler oluşturur. Bu, gözetimli eğitim yerine, özniteliklerin otomatik olarak öğrenildiği DÖ tanımına karşılık gelir [83].

Sınıflandırma katmanı, kendinden önceki tüm nöronlarla tam bağlantıya sahiptir. Bu katman, [$b \times N$] boyutlarına sahip iki boyutlu bir sonuç çıkarmaktadır. Burada b, örnek sayısı, N ise beklenen sınıf sayısını ifade etmektedir [52].

3.6.1.1. Giriş Katmanı

Giriş katmanları, ağın işlem yapabilmesi için görüntünün ham giriş verilerini yüklediğimiz ve sakladığımız yerdir. Bu giriş verileri, kanalların genişliğini, yüksekliğini ve sayısını belirler. Her piksel için RGB bir grafik için kanal sayısı üçtür. Şekil 3.21'de bu yapı temsil edilmektedir.



Şekil 3.21. 3d boyutlu giriş katmanı [52].

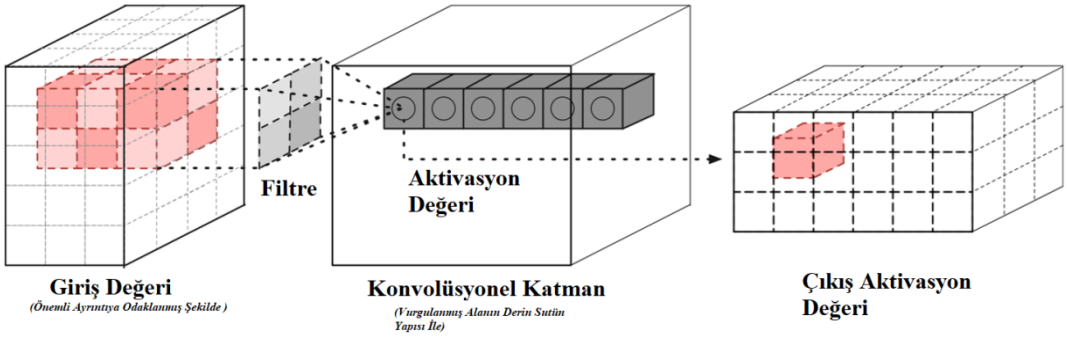
3.6.1.2. Özellik Çıkartma Katmanları

Özellik çıkartma, ham verinin anlamlı küçük boyutlu veriler haline indirilmesi aşamasıdır. Bu bölümde katmanlar açıklanmaktadır.

3.6.1.2.1. Evrişim Katman

Evrişim katmanları, CNN mimarisinin temel yapı taşları olarak kabul edilmektedir [52]. Şekil 3.22'de gösterildiği gibi, konvolüsyon katmanları, önceki katmandan lokal olarak bağlı nöronların bir parçasını kullanarak giriş verisini dönüştürür.

Bu katman, giriş katmanındaki nöronların bölgesi ile çıkış katmanında lokal olarak bağlandıkları ağırlıklar arasında yeni bir piksel ürünü hesaplamaktadır.



Şekil 3.22. Giriş ve çıkış değerlerine sahip konvolüsyonel katman [52].

Ortaya çıkan çıktı, genellikle aynı uzamsal boyutlara (veya daha küçük uzamsal boyutlara) sahiptir, ancak bazen çıktının derinlik boyutundaki öğelerin sayısı artmaktadır.

3.6.1.2.2. Havuzlama Katmanı

Havuzlama katmanları, art arda sıralanmış Evrişim katmanları arasına yerleştirilmektedir. Verinin uzamsal boyutunu (*genişlik ve yükseklik*) aşamalı olarak azaltmaktadır. Havuz katmanları bu azaltma sayesinde, aşırı uyumu kontrol altına alır [52]. Havuzlama katmanı, girişin her derinlik diliminde bağımsız olarak çalışmaktadır.

Havuzlama katmanı, girdi verilerini uzamsal olarak yeniden boyutlandırmak için max() işlemini kullanmaktadır. Bu işlem, maksimum havuzlama olarak isimlendirilir [52]. *2 × 2* boyutlu br filtre ile *max()* işlemi, filtre alanındaki dört sayının en büyüğünü alır. Bu işlem derinlik boyutunu değiştirmemektedir.

Giriş görüntüsü 32 piksel genişliğinde ve 32 piksel yüksekliğinde olan bir örnek üzerinde yapılan havuzlama işlemi sonrasında çıkış görüntüsü, genişlik ve yükseklik olarak daha küçük oluşmaktadır (16 piksel genişliğinde ve 16 piksel yüksekliğinde elde edilir). Bu küçülme, aktivasyon işlemlerinde %75 bir azalma ortaya çıkartır.

3.6.1.2.3. Sınıflandırma Katmanı

Bu katman ağın çıktısı olarak kullanılan sınıf puanlarının hesaplanması için kullanılmaktadır. Çıktı vektörü, sınıf sayısı boyutunda sahip bir vektördür. Bu katman genel olarak, aktivasyon fonksiyonu olarak sınıflandırma problemlerinde softmax fonksiyonu kullanılmaktadır [82]. Bu katmanda bulunan tüm nöronlar, önceki katmanda bulunan her nöron arasında bağlantı bulunmaktadır.

On sınıfa sahip olduğumuz bir problem için örnek çıktı, [*0 .1 .1 .75 0 0 0 0 0 .05*] değerli vektör olduğu düşünülsün. Bu vektör değerleri ağın, 2. Örnek olma olasılığını %10, 4. Örnek olma olasılığını %75 gibi kestirimde bulunduğu şeklinde yorumlanmaktadır.

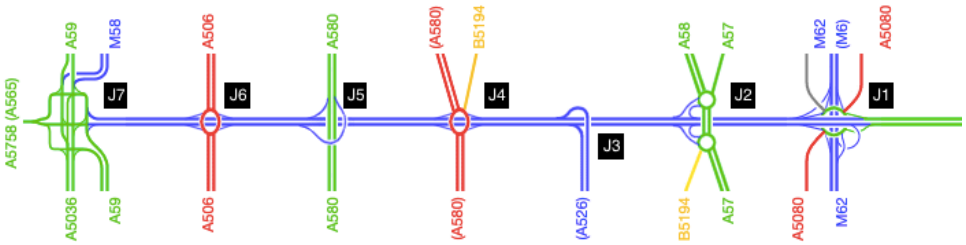
BÖLÜM 4

MATERYAL VE YÖNTEM

Bu bölümün amacı, Bölüm 3'te anlatılan DÖ yöntemleri kullanılarak gerçekleştirilen, M57 otoyoluna ait trafik akış yoğunluğu tahminleyen modellerin başarı performanslarını araştırmaktır.

4.1. VERİ SETİ

Trafik akış verileri, İngiltere'deki M57 otoyoluna bağlantılı birkaç kavşak (J1, J2, J3, J4, J5, J6, J7) ve otoyol çıkışında konumlanan döngü sensörleri vasıtasıyla, 1 Nisan 2015 ve 31 Ağustos 2019 yılları arasında her 15 dakikada bir toplanmış 929 bin 640 ölçüm değerinden oluşmaktadır. Veriler kamuya açık olarak erişilebilecek *tris.highwaysengland.co.uk/detail/trafficflowdata* [92] isimli web sitesinden elde edilmiştir.



Şekil 4.1. M57 yolu ve kavşak bağlantıları [93].

Liverpool çevre yolu olarak bilinmekte olan M57 otoyolu, 16 km uzunluğa sahip, gidiş ve geliş olarak toplam 6 şerite sahiptir. Çalışmada sadece güney yönüne hareket eden 3 şerit verileri dikkate alınmıştır. Çalışmada, otoyol başlangıcı olan J7 kavşağı ile J6 kavşağı arasındaki araç sayısı J_{76w}, J6 ile J5 arası araç sayılarını J_{65w}, J4 ile J3 arası araç sayılarını J_{43w}, J3 ile J2 arası araç sayılarını J_{32w}, J2 ile J1 arası araç sayılarını J_{21w} ve son olarak otoyol çıkışını noktası ve tahmin edilen nokta olan J1

kavşağı araç sayıları ise J_{Exit} olarak isimlendirilmiştir. Şekil 4.1'de M57 otoyolu üzerine kavşakların konumları işaret edilmiştir. Güney hareket yolu J7'den J1 kavşağına doğru akmaktadır. Şekil 4.1'de otoyola ait bağlantılar gösterilmiştir.

Şekil 4.2. M57 otoyoluna ait görsel ve ulaşım ağı.

Şekil 4.2'de M57 yoluna bağlantılı diğer anayollar ve otoyola ait bir görsel verilmiştir.

Çizelge 4.1. Veri setini oluşturan özellikler ve açıklamaları.

Özellik Adı:	**Açıklama:**
localTime	Ölçümün yapıldığı gün, ay, yıl, saat, dakika ve saniye bilgilerini içerir. Ölçümler 15 dakika aralıklarla elde edilmiştir. Ağın eğitiminde bu alan kullanılmaz.
year	Ölçümün yıl bilgisini tutar. Ağın eğitiminde bu alan kullanılmaz, sadece veri hakkında istatistiklerde kullanılmıştır.
month	Trafik yoğunluğu ve mevsim ilişkisini temsil eden ölçüm ayına ait değer.
dayOfWeek	Trafik ölçümüne hafta içi ve hafta sonu etkisini temsil eden değişkendir. İş günleri için 1, tatil günleri için 0 değeri almıştır.
$\mathbf{J_{Exit}}$	M57 yoluna ait güney yönünde en son noktası, çıkış kavşağından çıkan tüm türlere ait araçların toplam sayısını temsil eder.
$\mathbf{J_{76w}}$	J7 ve J6 arasındaki araç yoğunluğunu temsil eder. *W*, dört farklı araç tipinin ayrı ayrı sayıları ve toplam sayılarını temsil eden 5 farklı değişken ortaya çıkartır.
$\mathbf{J_{65w}}$	J6 ve J5 arasındaki araç yoğunluğunu temsil eder. *W*, dört farklı araç tipinin ayrı ayrı sayıları ve toplam sayılarını temsil eden 5 farklı değişken ortaya çıkartır.
$\mathbf{J_{43w}}$	J4 ve J3 arasındaki araç yoğunluğunu temsil eder. *W*, dört farklı araç tipinin ayrı ayrı sayıları ve toplam sayılarını temsil eden 5 farklı değişken ortaya çıkartır.
$\mathbf{J_{32w}}$	J3 ve J2 arasındaki araç yoğunluğunu temsil eder. *W*, dört farklı araç tipinin ayrı ayrı sayıları ve toplam sayılarını temsil eden 5 farklı değişken ortaya çıkartır.
$\mathbf{J_{21w}}$	J2 ve J1 arasındaki araç yoğunluğunu temsil eder. *W*, dört farklı araç tipinin ayrı ayrı sayıları ve toplam sayılarını temsil eden 5 farklı değişken ortaya çıkartır.

Çizelge 4.1'de, *w* ile belirtilen parametre, 0, 1, 2, 3 ve 4 değerlerini alarak her 2 kavşak noktası arasındaki toplam araç sayısı ve bir birinden uzunlukları farklı 4 araç türüne ait araç sayılarının ayrı ayrı toplamını verir. Örneğin veri setinde, J7 ve J6 kavşakları arasında, $J_{760,}$ J_{761}, J_{762}, J_{763}, J_{764} olmak üzere 5 değişken değer tutmaktadır. 0, 1, 2, 3 ve 4 sırası ile farklı boyutlu araçların toplam sayısı, 5.2 metrenden daha az uzunlukta, 5.21 ve 6.6 metre arasındaki uzunlukta, 6.61 ve 11.6 metre arasındaki uzunlukta ve 11.6 metreden daha büyük uzunluktaki araç türlerinin sayısını temsil etmektedir. W'in 0 olduğu durumlar ile alınan toplam araç sayısı modelin eğitimine girmez, sadece istatiksel çalışmalar için kullanılmıştır.

4.2. MATERYAL

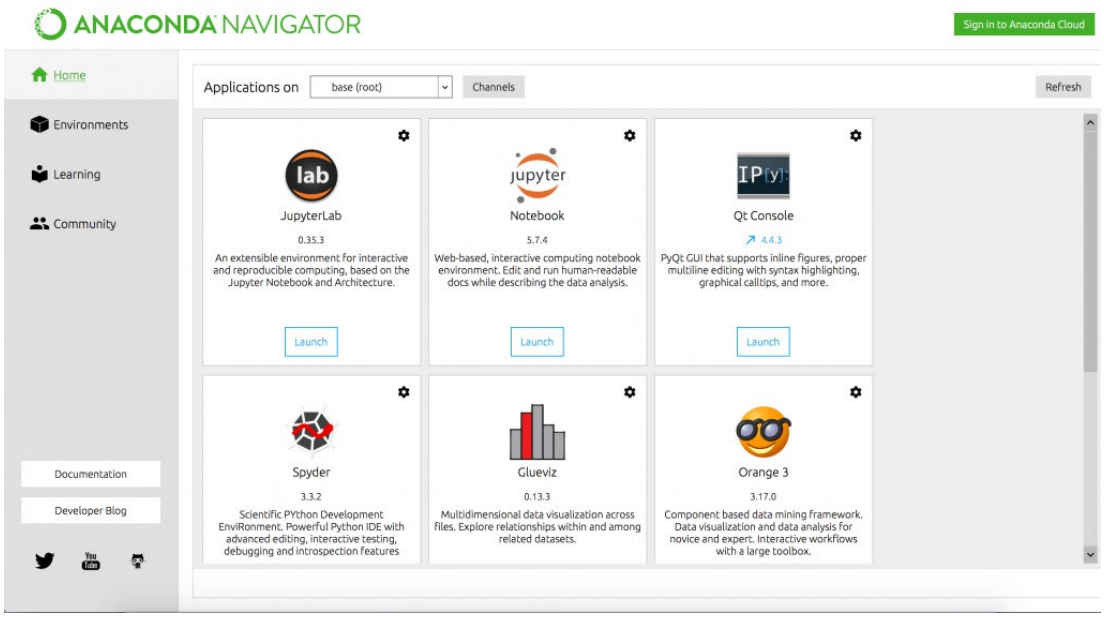
DÖ modellerinin karmaşıklıkları ve kodlama zorlukları nedeniyle, matematiksel ve bilimsel yönleri güçlü birçok kütüphane bağımsız geliştiriciler tarafından hazırlanmıştır. Bu çalışmada, esneklik ve kullanım kolaylığı gibi sebepler dikkate alarak seçilmiş temel kütüphaneler ve geliştirme araçları hakkında bilgi verilmektedir.

4.2.1. Anaconda

Anaconda, paket yönetimi ve dağıtımını kolaylaştırmayı amaçlayan, bilimsel hesaplamalar için Python ve R programlama dillerinin ücretsiz ve açık kaynak kodlu bir dağıtımıdır. Anaconda dağıtımı, 15 milyondan fazla kullanıcı tarafından kullanılmaktadır ve Windows, Linux ve Mac OS platformları için uygun 1500'den fazla veri bilim paketi içermektedir.

Anaconda dağıtımı, 1500'den fazla paketin yanı sıra conda paketi ve sanal çevre yöneticisi ile birlikte gelmektedir. Anaconda, bir komut satırı arayüzüne (CLI) ve grafiksel bir alternatif olarak bir Grafiksel Kullanıcı Arayüzü (GUI), Anaconda Navigator içermektedir. Conda ve pip paket yöneticisi arasındaki temel farklılık paket bağımlılıklarının yönetilmesi ile ilgilidir. Pip bir paket yüklediğinde, daha önce kurulmuş paketlerle çakışıp çakışmadığını kontrol etmeden bağımlı tüm Python paketlerini otomatik olarak yükler. Bu nedenle, örneğin Google Tensorflow'un

çalışan bir yüklemesi olan bir kullanıcı, bu kütüphane ile bağımlı numpy kitaplığının Tensorflow tarafından gerekli bir sürümünden farklı sürümü yüklemesinden dolayı çalışmasının durduğunu görebilir. Anaconda2'nin varsayılan kurulumunda Python 2.7, Anaconda3 kurulumu ise Python 3.7'yi içermektedir. Bu tez kapsamında Anaconda3 kullanılmıştır. Şekil 4.3'te programa ait ana ekran görüntüsü verilmiştir.



Şekil 4.3. Anaconda ekran görüntüsü.

4.2.1. Theano

Theano, Yoshua Bengio öncülüğünde Université de Montréal'da geliştirilmiş açık kaynak kodlu bir projedir ve ismini Yunan bir matematikçiden almıştır. Sözdizimi olarak NumPy kütüphanesine benzemektedir ve Python için sayısal bir hesaplama kütüphanesidir [84]. Çok boyutlu dizilerle karmaşık matematiksel ifadeleri gerçekleştirmede etkilidir. Birçok makine öğrenme uygulamasının temelinde, karmaşık bir matematiksel ifadenin tekrardan hesaplanması yatmaktadır [94]. Bu ifadelerin doğrudan hesaplanması, tüm ifadenin hesaplanması için en iyi strateji olmasa bile, basit bir şekilde matris veya tensör işlemleri ile yazılabilir. Bu karmaşık ifadeler üzerindeki yeteneği, sinir ağlarının eğitilmesi için çok önemli avantaj sağlamaktadır.

Theano, Python'da matematiksel ifadeler için bir derleyicidir. NumPy, BLAS gibi kod yapılarını alarak, Merkezi İşlemci Birimi (CPU) ve GPU üzerinde olabildiğince hızlı çalıştırmak için yerel bir dile (C++) dönüştürmekte ve optimize etmektedir. DÖ algoritmalarında kullanılan büyük NN algoritmaları için gerekli hesaplama türlerini işlemek için özel olarak tasarlanmıştır.

4.2.2. TensorFlow

TensorFlow, Google tarafından büyük ölçekli makine öğrenimi uygulamaları için kullanılan açık kaynaklı bir kütüphanedir [84]. Google Brain ekibi tarafından geliştirilen TensorFlow, bir takım makine öğrenme algoritmaları ve DÖ (sinir ağı) modelleri bir araya getirir ve ortak bir uygulama çerçevesi ortaya çıkartır. Bu uygulamaları yüksek performanslı C++ kodları ile uygulamaktadır fakat geliştiriciler için Python kullanan bir API desteği sağlamaktadır. TensorFlow, görüntü tanıma, doğal dil işleme, kısmi diferansiyel denklem tabanlı simülasyonlar, el yazısı sınıflandırma, sözcük yerleştirme, RNN ve makine çevirisi için DNN'leri eğitir ve çalıştırabilir.

TensorFlow, geliştiricilerin veri akış grafları oluşturmalarını sağlar. Graftaki her düğüm matematiksel bir işlemi temsil eder ve düğümler arasındaki her bağlantı çok boyutlu bir veri dizisi veya tensör ile ifade edilmektedir. Bu yapılar programcıya Python dili aracılığıyla sağlanmaktadır. TensorFlow'daki düğümler ve tensörler Python nesneleridir ve TensorFlow uygulamaları da Python uygulamalarıdır.

4.2.3. Keras

Keras, Theano veya TensorFlow'un üzerinde çalışabilen, yüksek modülerliğe sahip, Python programlama dili ile yazılmış bir NN kütüphanesidir [84]. Keras, hem CNN'leri hem de RNN'leri destekleyen kütüphanelerden biridir ve GPU ve CPU üzerinde kolayca çalıştırılabilir. Bir model, mümkün olduğu kadar az kısıtlama ile birbirine entegre olabilen bir dizi veya bağımsız bir graf olarak temsil edilir. Sinir katmanları, maliyet fonksiyonları, optimize ediciler, başlatma yöntemleri ve aktivasyon fonksiyonları, bir model oluşturmak için birleştirilebilecek bağımsız

modüllerdir. Şekil 4.4'te, projede kullanılan Keras kütüphanesi kodlarının bir örnek görüntüsü bulunmaktadır.

```
62
63
64 #Backendde tensorFlow çalıştıran Keras kütüphanesinin eklenilmesi
65 from keras import Sequential
66 from keras.layers import Dense, LSTM
67
68
69 model = Sequential()
70 model.add(LSTM(units=30, return_sequences= True, input_shape=(X.shape[1],28)))
71 model.add(LSTM(units=30, return_sequences=True))
72 model.add(LSTM(units=30))
73 model.add(Dense(units=1))
74 print(model.summary())
75
76
77
78
79 model.compile(optimizer='adam', loss='mean_squared_error')
80 model.fit(X, y, epochs=10, batch_size=32)
81
```

Şekil 4.4. Keras kod örneği.

4.3. VERİ ANALİZİ VE ÖN İŞLEMLER

Yaklaşık 5 yıl boyunca 1 milyona yakın farklı ölçüm değeri, M57 otoyolunda bulunan döngü sensörleri vasıtasıyla elde edilmiştir. Bu verilerin DÖ modelinde kullanılmadan önce veri seti üzerinde ön işlemlere ihtiyaç duyulmaktadır. Bunlar, kayıp verilerin tamamlanması, eğitim matrisinin oluşturulması ve gereksiz alanların yok edilmesi ve sayısal olmayan içeriklerin sayısal karşılıklara dönüştürülmesidir. Ayrıca bu başlık altında verinin keşfedilebilmesi için çeşitli testler uygulanmış ve görsel olarak gösterilmiştir.

4.3.1. Kayıp Veriler

Eğitim kümesinde kullanılan ve bir zaman serisi üzerinde, ölçüm değeri eksik olan veriler, modelin tahmin performansını oldukça azaltmaktadır [95]. Çalışmamızda, 929 bin 640 farklı ölçüm değerinin yaklaşık %6'i, yani 56 bin 375'i bilinmeyen sebeplerle elde edilememiştir. Bu kayıp verilerden 578'i, toplam geçen araç sayısı ve araç türüne göre segmente edilen sayıların toplamı bakımından uyuşmamaktadır. Bu durumda toplam geçiş sayısı olarak, segmente edilen araçların toplam sayısı dikkate alınmıştır. Ölçüm değeri olmayan verilerin büyük bir ağırlığının, 2015 yılına ait 9 ay boyunca olduğu tespit edilmiştir.

Eksik verilerin doldurulmasında, C# dilinde hazırlanmış bu çalışmaya özel bir program kullanılmıştır. Tamamlama yöntemi olarak, aynı gün ve saat dilimine ait, aynı ölçüm sensörlerinden alınan diğer 4 yılki ölçümlerin ortalama değerleri alınmış ve çıkan sonuç küçük tam sayı değerine yuvarlanarak o türe araç segment toplamı alanına yazılmıştır. Tamamlama yaklaşımını bir senaryo ile ifade edecek olursak, 25 Aralık 2017 08:29:00 tarihi için J7 ve J6 bağlantı yolunda konumlandırılan döngü sensörünün kayıt toplayamadığını kabullenelim. Hazırlanan tamamlama programımız, 2015, 2016, 2018 ve 2019 yıllarına ait 25 Aralık tarihi ve 08:29:00 zamanı için elde edilen ölçüm verilerin ortalamasını hesaplamakta daha sonra küçük tam sayı değerine yuvarlayarak eksik tarih verisi olarak kabul etmektedir.

localTime	year	month	dayOfWeek	JExit	J760	J761	J762	J763	J764	J650	J651	J652	J653	J654	J430	J431	J432	J433	J434	J320	J321	J322	J323	J324	J210	J211	J212	J213	J214
1.04.2015 01:14:00	2015	4	1	29	0	0	0	0	0	29	16	1	0	12	35	20	1	1	13	29	16	1	0	12	29	16	1	0	12
1.04.2015 01:29:00	2015	4	1	36	0	0	0	0	0	36	20	2	2	12	38	21	2	2	13	36	20	2	2	12	36	20	2	2	12
1.04.2015 01:44:00	2015	4	1	26	0	0	0	0	0	26	12	3	3	8	30	13	4	5	8	26	12	3	3	8	26	12	3	3	8
1.04.2015 01:59:00	2015	4	1	26	0	0	0	0	0	26	12	3	1	10	25	12	2	2	9	26	12	3	1	10	26	12	3	1	10
1.04.2015 02:14:00	2015	4	1	32	0	0	0	0	0	32	15	5	6	6	38	21	3	6	8	32	15	5	6	6	32	15	5	6	6
1.04.2015 02:29:00	2015	4	1	27	0	0	0	0	0	27	14	0	2	11	22	11	1	2	8	27	14	0	2	11	27	14	0	2	11
1.04.2015 02:44:00	2015	4	1	25	0	0	0	0	0	25	13	3	3	6	28	13	5	3	7	25	13	3	3	6	25	13	3	3	6
1.04.2015 02:59:00	2015	4	1	20	0	0	0	0	0	20	10	4	0	6	21	10	2	2	7	20	10	4	0	6	20	10	4	0	6
1.04.2015 03:14:00	2015	4	1	28	0	0	0	0	0	28	15	1	2	10	37	18	5	4	10	28	15	1	2	10	28	15	1	2	10
1.04.2015 03:29:00	2015	4	1	50	0	0	0	0	0	50	24	3	13	10	55	22	4	11	18	50	24	3	13	10	50	24	3	13	10
1.04.2015 03:44:00	2015	4	1	47	0	0	0	0	0	47	27	4	2	14	43	24	8	3	8	47	27	4	2	14	47	27	4	2	14

Şekil 4.5. Modele uygulanan veri setine ait görüntü (ilk 10 kayıt görünümü).

Şekil 4.5'te ilk 10 kayıt örneği görünen veri seti tüm işlemler sonrası [154940 X 30] boyutunda bir veri matrisine dönüştürülmüştür. Veri seti içerisinde bulunan ölçüm değerleri, 2015-01-04 01:14:00 ile 2019-12-08 23:59:00 tarihleri arasını kapsamaktadır.

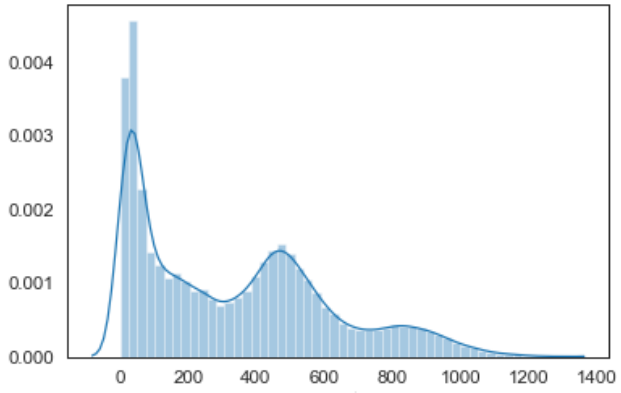
4.3.2. Tanımlayıcı İstatistikler ve Testler

D'Agostino's K-squared testi, dağılımın normal bir dağılıma sahip olup olmadığını belirlemeye yarayan bir testir. Basıklık ve çarpıklık dönüşümlerine dayanmaktadır. Eşitlik 4.1'de test, matematiksel olarak ifade edilmektedir. Eşitlikte, $Z_1(g_1)^2$ çarpıklık, $Z_1(g_1)^2$ ise basıklık değerini ifade etmektedir. Çizelge 4.2'de D'Agostino's K-squared normallik testi için p değeri 0.05'den daha küçüktür. Bu değer, M57 otoyol çıkış sayılarının normal dağılıma uygun olmadığını ve otoyol

çıkış rakamlarının diğer veri seti özellikleri ile arasında doğrusal olmayan bağımlılık içerebileceğini göstermektedir.

$$K^2 = Z_1(g_1)^2 + Z_2(g_2)^2 \tag{4.1}$$

Normal dağılım, veri seti üzerinde çarpıklık ve basıklık değerleri hesaplanarak da test edilmiştir. Çarpıklık değerinin, 0'dan farklı olması ve basıklık değerinin ise -1 ve +1 arasında olması verimizin gaussian bir dağılıma sahip olmadığını ispat etmektedir [65]. Ayrıca M57 otoyolu araç çıkış verisini tanımlamak için, ortalama, standart sapma v.d. değerleri Çizelge 4.2'de verilmiştir.



Şekil 4.6. M57 otoyolundan çıkış yapan araçların histogram dağılımı.

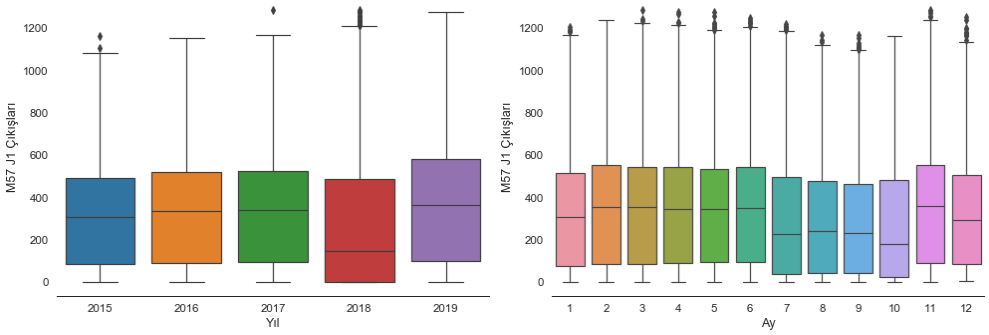
Şekil 4.6'te, Otoyoldan çıkış yapan araçların histogram dağılımı gösterilmiştir. Grafikte, X ekseni araç sayısını, Y ekseni ise 0-1 aralığında histogram basıklığını ifade etmektedir.

Çizelge 4.2. Otoyol çıkış noktasına ait tanımlayıcı değerler.

Ortalama	Min	Max	Standart Sapma	Çarpıklık	Basıklık	*p* değeri
333.914	0	1283	280.079	0.636	-0.494	0.000

Şekil 4.7'te yıllara göre araç geçiş sayılarının medyan yan yana kutu grafikte incelendiğinde, 2019 yılı değerinin en yüksek olduğu görünmektedir. Bu değer bize 2019 yılının sadece 9. aya kadar olan rakamlarına sahip olduğumuz için doğru sonuçları vermez. Yaz aylarının araç geçiş rakamlarının fazla olması ve ölçümün 2019 yılı için o ayları kapsaması bu sonucu doğurmuştur. Ay bazında medyan

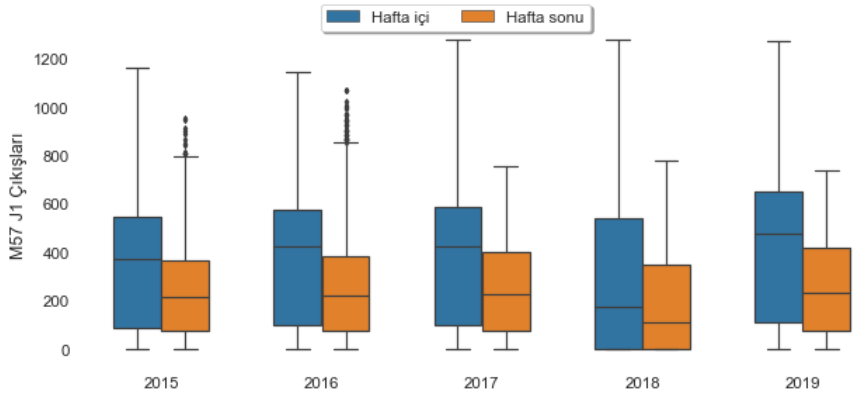
değerleri incelendiğinde, sonbahar ayının araç geçişi bakımından en az değerlere sahip olduğu görülmektedir. Grafikler kapsamlı yorumlanır ise, tahmin problemlerinde mevsimleri temsil eden değişkenlerin (veri seti içerinde month değer alanı) önemi daha açık ortaya çıkacaktır.



Şekil 4.7. M57 yoluna ait araç çıkışlarının medyan değerlerinin yıllık ve aylık dağılımı.

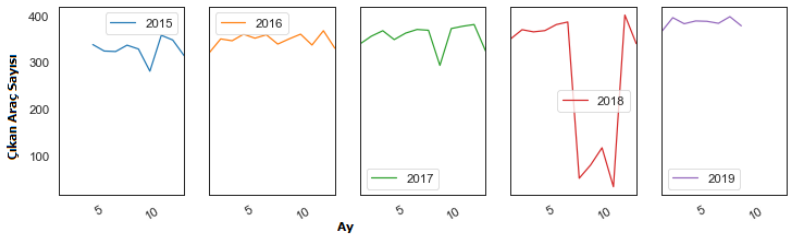
Şekil.4.8'de iş günü ve tatil günlerinin araç geçişi üzerinde etkisi incelenmeye çalışılmaktadır. Grafik incelendiğinde hafta içi geçişlerin tüm zamanlarda gözle görülür bir şekilde arttığı görünmektedir. Veri setinde bu alanın temsili için dayOfWeek isimli alan kullanılmıştır.

Şekil 4.9'da ölçüm alınan 5 yıl süresince araç çıkışlarının aylara göre dağılım grafiği verilmiştir. Grafik incelendiğinde, geçiş rakamlarındaki azalış ve artış değerlerinin yıl içinde tarihlere göre benzerlik gösterdiği görülmektedir.



Şekil 4.8. M57 yoluna ait araç çıkışlarının medyan değerlerinin hafta içi ve hafta sonu dağılımı.

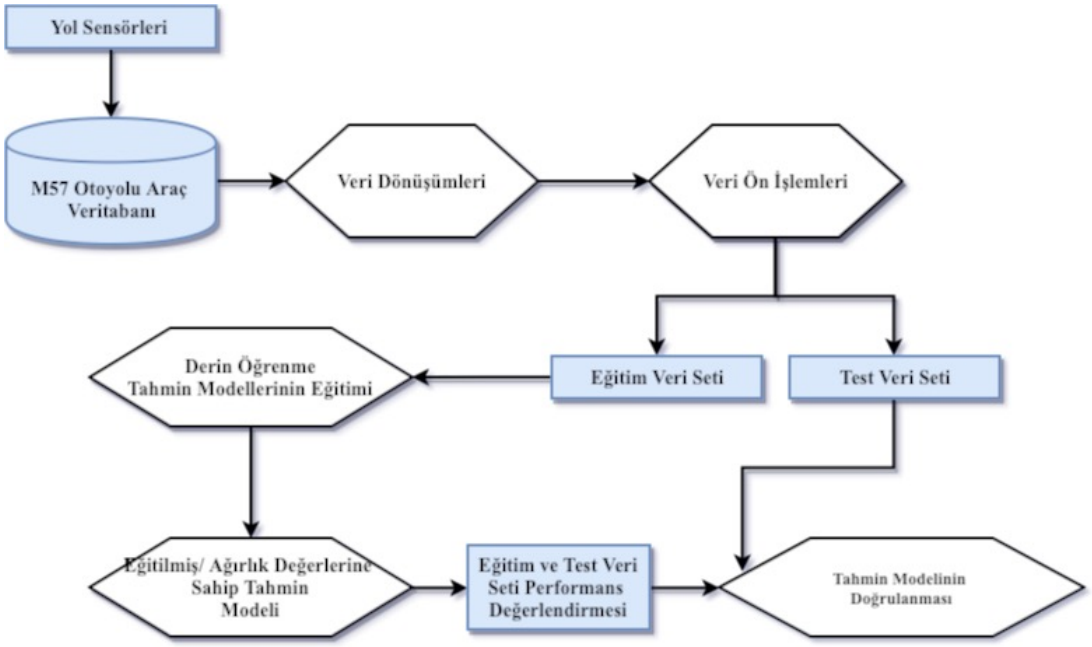
2018 yılı ilkbahar döneminde yolun kullanımı, bakım ve onarım ve diğer nedenler ile orantısal olarak azaldığı ve önceden tahmin edilemeyen bir oynaklığın ortaya çıktığı görülmüştür. Bu beklenmedik yani olasılıksal değişimlerin, sinir ağlarının karmaşık öğrenme gücü sayesinde matematiksel bir ifadeye dönüştürülmüştür.



Şekil 4.9. M57 otoyolu araç çıkışlarının yıl içinde aylara göre dağılımı.

4.4. MODELLER VE SONUÇLAR

Tez çalışmasında otoyola ait trafik çıkış yoğunluğunun 15 dakika önceden tahmin modelinin geliştirilmesi için 5 farklı algoritma kullanılarak 44 farklı model elde edilmiştir. Algoritmaların belirlenmesinde literatür üzerinde yapılan taramalardan faydalanılmıştır. Çalışmada kullanılan algoritmalar sırası ile RNN, LSTM, S-LSTM, B-LSTM ve GRU NN'leridir. RNN'ler zaman serisi problemlerinde geleneksel NN'lere göre daha üstündür. RNN'lerin tahminleme problemlerinde başarısının arttırılması için bellek birimi eklenerek LSTM ağlar geliştirilmiştir. Literatür incelendiğinde LSTM'in en yaygın varyasyonları olan, GRU, S-LSTM ve B-LSTM yöntemlerinin trafik akış tahminlemesinde kullanılmasına rastlanmamıştır. Trafik tahmin problemlerindeki başarılarının değerlendirilebilmesi ve literatüre kazanım sağlanması için bu algoritmalar kullanılan yöntemler arasına eklenmiştir. Her bir algoritma, eğitim iterasyon sayısı (epoch), eğitim ve test veri seti büyüklüğü, gizli katman sayısı ve eğitime aynı anda alınan veri sayısı büyüklüğü (batch size) gibi hiper parametreler değiştirilerek ayrı ayrı çalıştırılmış ve sonuçlar kaydedilmiştir. Hiper parametrelerin belirlenmesinde, kullanılan kütüphanenin varsayılan değerleri ve rastgele seçimler yöntemi benimsenmiştir.



Şekil 4.10. Trafik akış tahmin modeline ait akış diyagramı

Çalışmada kullanılan veri setinde bulunan veri kullanılarak, 3 farklı veri seti ortamı oluşturulmuştur. 3 ortamda veri seti, toplamında eşit miktarda veri olmak üzere, %90, %80 ve %70'i eğitim, kalan kısmı (%10, %20, %30) test verisi olacak şekilde bölünmüştür. Veri seti bölme işleminde, sıralı yapı dikkate alınmıştır. Rastgele seçimden kaçınarak bu yöntemin benimsenme nedeni, sonraki bir zamana dair tahminin ondan hemen önceki durumdan etkilenmesidir. Farklı hiper parametreler ile tasarlanan modeller için bu 3 veri seti grubu ayrı ayrı denenmiş ve sonuçlar kaydedilmiştir.

Şekil 4.10'da, trafik akışının tahmin edilmesinde kulamnılan veri ve veri üzerinde işlem yapan DÖ modellerinin işlem adımları gösterilmektedir. Modellerin tasarlandığı ve test edildiği fiziksel ve yazılımsal bilgisayar ortamına ait bilgiler Çizelge 4.3'te verilmiştir.

Bu bölümde, sonuçların değerlendirilmesi için MSE ve MAE hata fonksiyonuna dayanan sonuç bulguları, model çalışma süreleri ve hiper parametreler arasında ilişki, eğitim sonuçları ile veri seti büyüklüğü, eğitim iterasyon sayısı ve gizli katman sayısı arasındaki ilişkiler verilmektedir.

Çizelge 4.3. Modellerin çalıştırıldığı bilgisayara ait bilgiler.

İşletim Sistemi:	OS X El Captain
CPU:	2.8 GHz Intel Core i5
GPU:	Intel Iris 1516 MB
Bellek:	8 GB 1600 MHz DDR3
Keras:	Sürüm 2.2.4
TensorFlow:	Sürüm 1.13.1
Theano:	Sürüm 1.0.4
Anaconda Navigator:	1.9.7

4.4.1. RNN ile Tasarlanan Model ve Bulgular

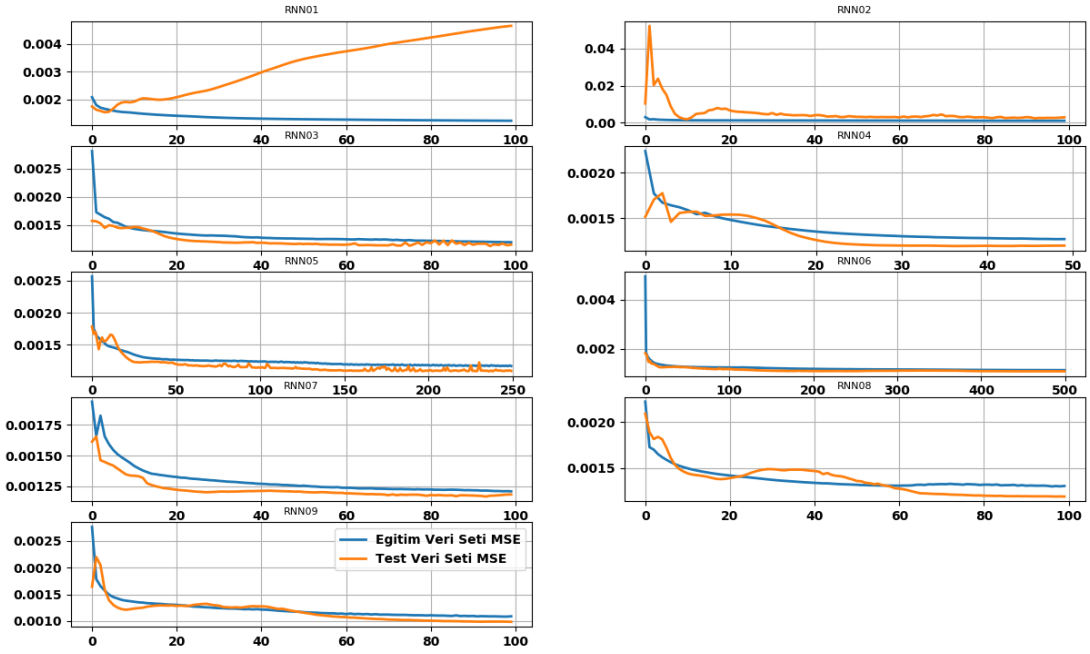
Tez çalışmasında RNN ile kurulan model, veri seti büyüklüğü, eğitim iterasyon sayısı, girdi olarak giriş katmanı düğümüne bir defada giriş yapan veri grubu büyüklüğü, gizli katman, kullanılan ağırlık sayısı (param) ve düğüm miktarı (node) bakımından değiştirilerek ayrı ayrı test edilmiştir. Oluşturulan modellere ait değerler Çizelge 4.4'te verildiği şekilde etiketlenmiştir. Tasarlanan bu model için tüm varyasyonlarda aktivasyon fonksiyonu olarak hiperbolik tanjant fonksiyonu kullanılmıştır. Hata fonksiyonu (loss function) için MSE, optimizasyon yöntemi (optimizer) olarak ADAM kullanılmıştır ve Öğrenme Katsayısı (Learning Rate) 0.001 olarak sabit tutulmuştur.

Çizelge 4.4. Kullanılan RNN modellerine ait parametre bilgileri ve bulgular.

Model Adı	epoch	Batch size	Gizli Katman Sayısı	Düğüm Sayısı	Ağırlık Sayısı	Eğitim Veri Seti	Eğitim Süresi	MSE	MAE
RNN01	100	72	1	51	4001	%70	0:09:56	87.51	12.62
RNN02	100	72	1	51	4001	%80	0:15:28	69.67	4.87
RNN03	100	72	1	51	4001	%90	0:11:16	43.63	6.86
RNN04	50	72	1	51	4001	%90	0:05:36	44.47	5.97
RNN05	250	72	1	51	4001	%90	0:27:59	42.43	4.69
RNN06	500	72	1	51	4001	%90	0:52:15	42.14	2.81
RNN07	100	36	1	51	4001	%90	0:21:53	44.14	2.13
RNN08	100	144	1	51	4001	%90	0:06:04	44.16	6.47
RNN09	100	72	2	101	9051	%90	0:17:19	40.26	10.79

Çizelge 4.4'te modellerin özellikleri incelendiğinde, eğitim veri seti büyüklük olarak artması durumunda modelin MSE değeri dikkate değer bir oranda azalmakta ve

tahmin başarısı yükselmektedir. MAE değerlerinin düşük iterasyonlarda başarılı sonuçlar ürettiği gözlenmektedir.

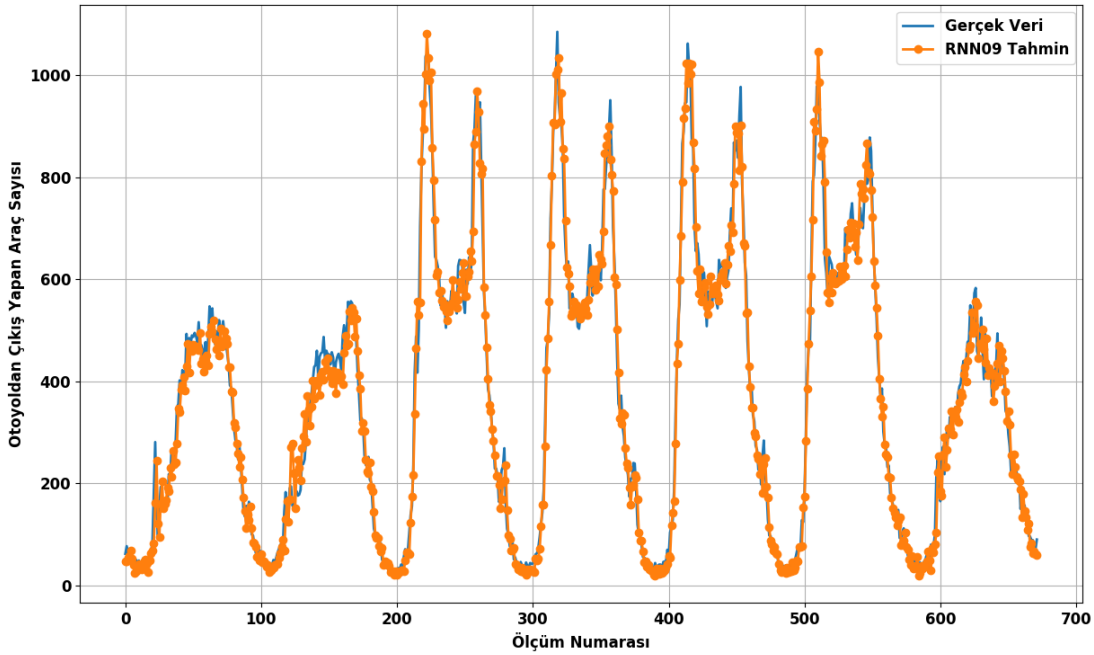


Şekil 4.11. RNN modelleri için, eğitim ve test veri setleri üzerinde hata fonksiyonların her iterasyon için sonuçları.

Modellerin eğitim iterasyon sayılarının artması durumunda MSE değerleri küçülmekte ancak eğitim için harcanan süre doğrusal bir şekilde yükselmektedir. MSE sonuçları bakımından, modele başlangıçta verilen girdi gruplarının büyüklükleri ile zaman arasında ters bir orantı tespit edilmesine rağmen, modelin başarısı ile doğrusal bir ilişki tespit edilememiştir. Çizelge 4.4'te RNN modelleri için, eğitim ve test veri setleri üzerinde hata fonksiyonların her iterasyon için sonuçları gösterilmektedir. Grafikte X ekseni iterasyon numarasını, Y ekseni o iterasyona ait MSE değerini göstermektedir.

RNN model sonuçları MSE değerleri dikkate alınarak incelendiğinde, en iyi tahmin sonucu üreten modelin, 40.26 MSE değerine sahip, gizli katman sayısı arttırılmış olarak tasarlanan RNN09 mimarisine ait olduğu görülmüştür. MAE değerleri dikkate alındığında RNN07 mimarisi, ağın girişlerine uygulanan veri büyüklüğü bakımından en iyi olarak bulunmuştur. RNN09 mimarisine ait gerçek ve tahmin sonuçlarının grafiksel karşılaştırması Şekil 4.12'de gösterilmektedir. İterasyon sayısının

arttırılması ise öğrenmenin artmasına, bir başka değişle tahmin doğruluğunun artmasına neden olduğu bulunmuştur.



Şekil 4.12. RNN09 modeli ile 25 Ağustos 2019 ile 31 Ağustos 2019 tarihleri arasında 15 dakikalık aralıklarla otoyol araç çıkış sayısının gerçek ve tahmin değerlerinin grafiksel gösterimi.

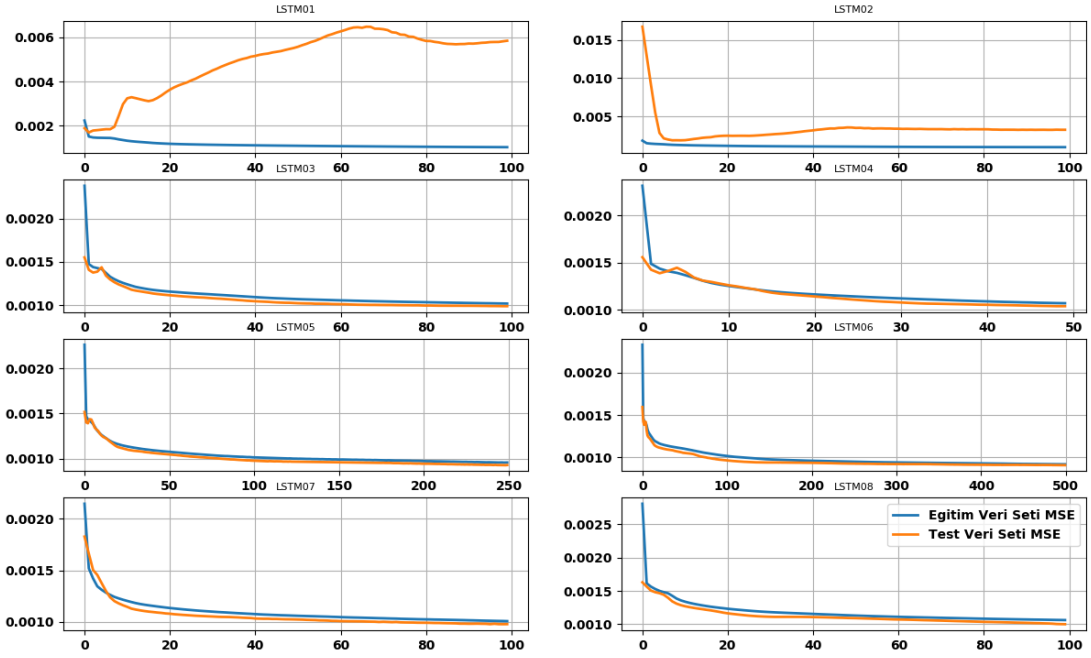
4.4.2. LSTM ile Tasarlanan Model ve Bulgular

Tez çalışmasında LSTM ile kurulan model, veri seti büyüklüğü, eğitim iterasyon sayısı (epoch), girdi olarak giriş katmanı düğümüne bir defada giriş yapan veri grubu büyüklüğü (batch size), kullanılan ağırlık sayısı (param) ve düğüm miktarı (node) bakımından değiştirilerek ayrı ayrı test edilmiştir. Çok katmanlı LSTM için S-LSTM kullanıldığı için, gizli katman sayısı LSTM deneylerinde sabit tutulmuştur. Oluşturulan modellere ait değerler Çizelge 4.5'te verildiği şekilde etiketlenmiştir.

Tasarlanan bu model için tüm varyasyonlarda aktivasyon fonksiyonu olarak ReLU fonksiyonu kullanılmıştır. Hata fonksiyonu (loss function) için MSE, optimizasyon yöntemi (optimizer) olarak ADAM kullanılmıştır ve Öğrenme Katsayısı (Learning Rate) 0.001 olarak sabit tutulmuştur.

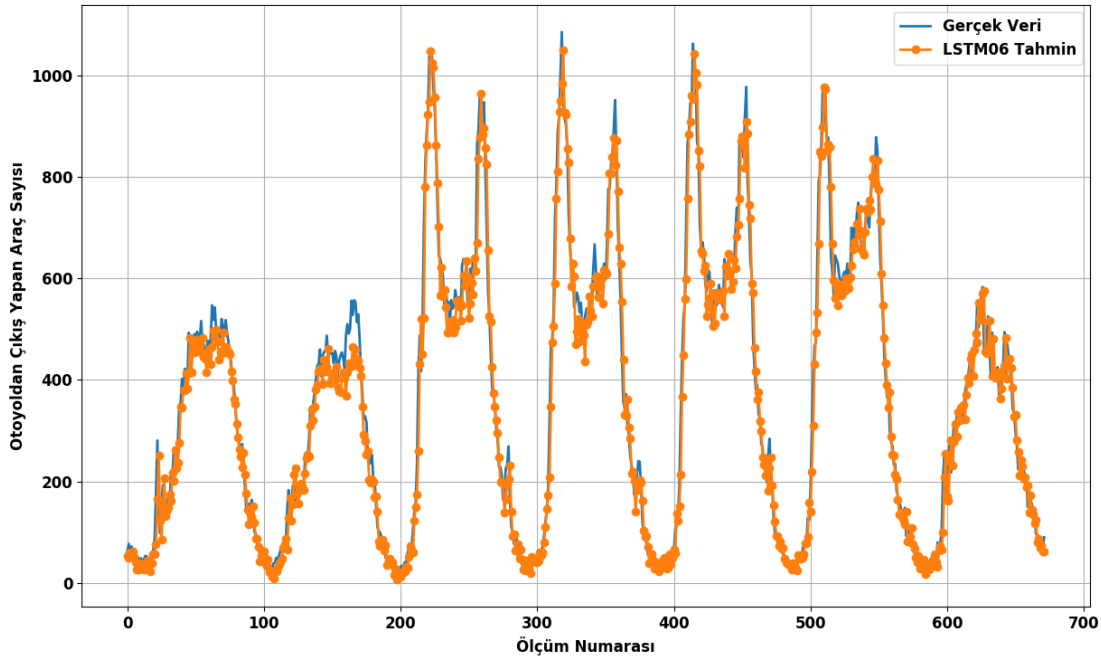
Çizelge 4.5. Kullanılan LSTM modellerine ait parametre bilgileri ve bulgular.

Model Adı	epoch	Batch size	Gizli Katman Sayısı	Düğüm Sayısı	Ağırlık Sayısı	Eğitim Veri Seti	Eğitim Süresi	MSE	MAE
LSTM01	100	72	1	51	15851	%70	0:21:06	98.06	20.26
LSTM02	100	72	1	51	15851	%80	0:27:44	73.41	2.11
LSTM03	100	72	1	51	15851	%90	0:24:22	40.32	2.61
LSTM04	50	72	1	51	15851	%90	0:13:07	41:38	2.02
LSTM05	250	72	1	51	15851	%90	1:02:07	39:11	3.03
LSTM06	500	72	1	51	15851	%90	1:59:59	38:66	2.10
LSTM07	100	36	1	51	15851	%90	0:45:45	40.13	3.04
LSTM08	100	144	1	51	15851	%90	0:13:50	40.59	4.39



Şekil 4.13. LSTM modelleri için, eğitim ve test veri setleri üzerinde hata fonksiyonların her iterasyon için sonuçları.

Şekil 4.13 ve Çizelge 4.5'te gösterilen MSE değerleri incelendiğinde, artan eğitim iterasyon sayısının modelin eğitim başarısını arttırdığı gözlenmiştir. LSTM06 mimarisi en iyi MSE sonuçlarına sahiptir. MAE sonuçları ele alındığında, LSTM04 en iyi mimari olarak gözlenmiştir. Ağın iterasyon sayısındaki azalma MAE değerlerini küçülterek ağın başarısını arttırmaktadır. Şekil 4.13'de X ekseni iterasyon numarasını, Y ekseni o iterasyona ait MSE değerini göstermektedir.



Şekil 4.14. LSTM06 modeline ait gerçekleşen ve elde edilen sonuçların karşılaştırılması.

Şekil 4.14'de, LSTM06 modeli ile 25 Ağustos 2019 ile 31 Ağustos 2019 tarihleri arasında 15 dakikalık aralıklarla otoyol araç çıkış sayısının gerçek ve tahmin değerlerinin grafiksel olarak karşılaştırılması yapılmıştır.

4.4.3. S-LSTM ile Tasarlanan Model ve Bulgular

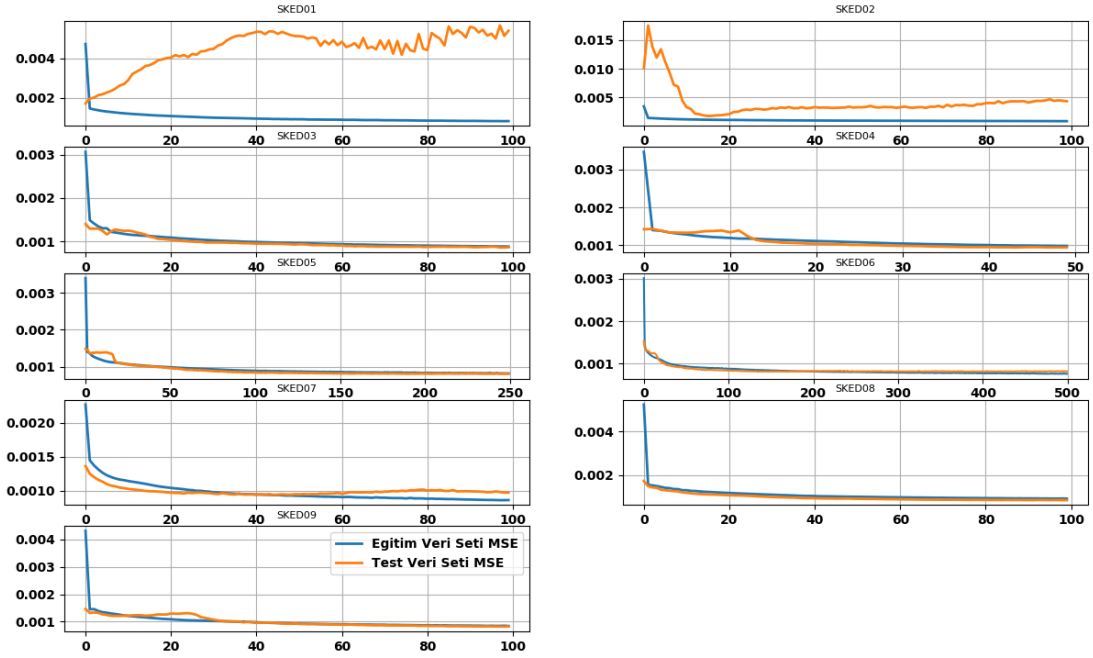
Tez çalışmasında S-LSTM ile kurulan model, veri seti büyüklüğü, eğitim iterasyon sayısı (epoch), girdi olarak giriş katmanı düğümüne bir defada giriş yapan veri grubu büyüklüğü (batch size), kullanılan ağırlık sayısı (param), gizli katman sayısı ve düğüm miktarı (node) bakımından değiştirilerek ayrı ayrı test edilmiştir. Oluşturulan modellere ait değerler Çizelge 4.6'da verildiği şekilde etiketlenmiştir. Tasarlanan bu model için tüm varyasyonlarda aktivasyon fonksiyonu olarak hiperbolik tanjant fonksiyonu kullanılmıştır.

Hata fonksiyonu (loss function) için MSE, optimizasyon yöntemi (optimizer) olarak ADAM kullanılmıştır ve Öğrenme Katsayısı (Learning Rate) 0.001 olarak sabit tutulmuştur.

Çizelge 4.6. Kullanılan S-LSTM modellerine ait parametre bilgileri ve bulgular.

Model Adı	epoch	Batch size	Gizli Katman Sayısı	Düğüm Sayısı	Ağırlık Sayısı	Eğitim Veri Seti	Eğitim Süresi	MSE	MAE
SKED01	100	72	2	101	36051	%70	0:35:56	94.22	22.96
SKED02	100	72	2	101	36051	%80	0:38:36	84.68	2.36
SKED03	100	72	2	101	36051	%90	0:43:07	37.88	2.35
SKED04	50	72	2	101	36051	%90	0:21:34	39.26	21.97
SKED05	250	72	2	101	36051	%90	1:44:16	36.68	3.70
SKED06	500	72	2	101	36051	%90	3:21:44	36.74	2.17
SKED07	100	36	2	101	36051	%90	1:19:30	40.05	1.51
SKED08	100	144	2	101	36051	%90	0:23:45	37.40	2.62
SKED09	100	72	4	201	76451	%90	1:17:44	36.88	2.19

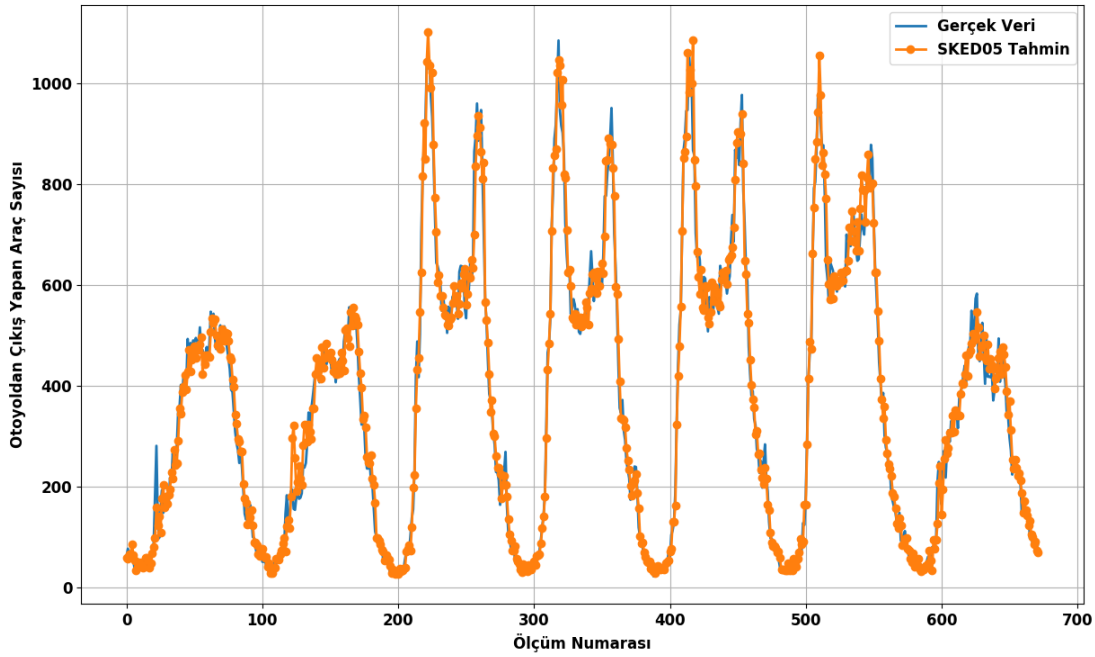
Şekil 4.15'de S-LSTM modelleri için, eğitim ve test veri setleri üzerinde hata fonksiyonların her iterasyon için sonuçları gösterilmektedir.



Şekil 4.15. S-LSTM modelleri için, eğitim ve test veri setleri üzerinde hata fonksiyonların her iterasyon için sonuçları.

Çizelge 4.6'da, X ekseni iterasyon numarasını, Y ekseni o iterasyona ait MSE değerini göstermektedir. Sonuçlar incelendiğinde, S-LSTM modelleri içerisinde en iyi MSE değeri iterasyon sayısı ve gizli katman sayısının artışı ile ilişkili olduğu bulunmuştur. Batch Size değerinin küçülmesi ağın MAE değerleri bakımından

tahmin başarısını arttırmaktadır. S-LSTM, LSTM mimarisinde bulunan gizli katman sayısının arttırılması ile elde edilen bir LSTM modelidir. Deneylerde elde edilem MSE sonuçlarından yola çıkarak en iyi tahmin sonuçlarına sahip olan mimari SKED05 olarak belirlenmiştir. MAE açısından, daha küçük veri girdi boyutuna sahip SKED07 en iyi mimari olarak görülmektedir.



Şekil 4.16. SKED05 modeline ait gerçekleşen ve elde edilen sonuçların karşılaştırılması.

Şekil 4.16'da, SKED05 modeli ile 25 Ağustos 2019 ile 31 Ağustos 2019 tarihleri arasında 15 dakikalık aralıklarla otoyol araç çıkış sayısının gerçek ve tahmin değerlerinin karşılaştırılması gösterilmiştir.

4.4.4. B-LSTM ile Tasarlanan Model ve Bulgular

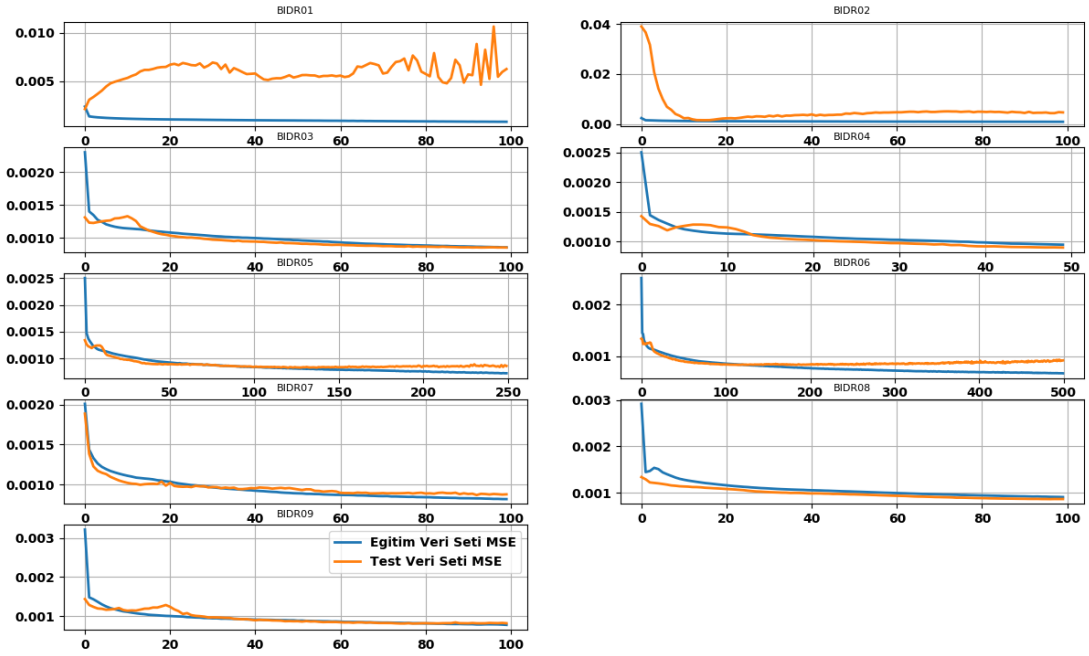
Tez çalışmasında B-LSTM ile kurulan model, veri seti büyüklüğü, eğitim iterasyon sayısı (epoch), girdi olarak giriş katmanı düğümüne bir defada giriş yapan veri grubu büyüklüğü (batch size), kullanılan ağırlık sayısı (param), gizli katman sayısı ve düğüm miktarı (node) bakımından değiştirilerek ayrı ayrı test edilmiştir. Oluşturulan modellere ait değerler Çizelge 4.7'de verildiği şekilde etiketlenmiştir. Tasarlanan bu

model için tüm varyasyonlarda aktivasyon fonksiyonu olarak ReLU fonksiyonu kullanılmıştır.

Çizelge 4.7. Kullanılan B-LSTM modellerine ait parametre bilgileri ve bulgular.

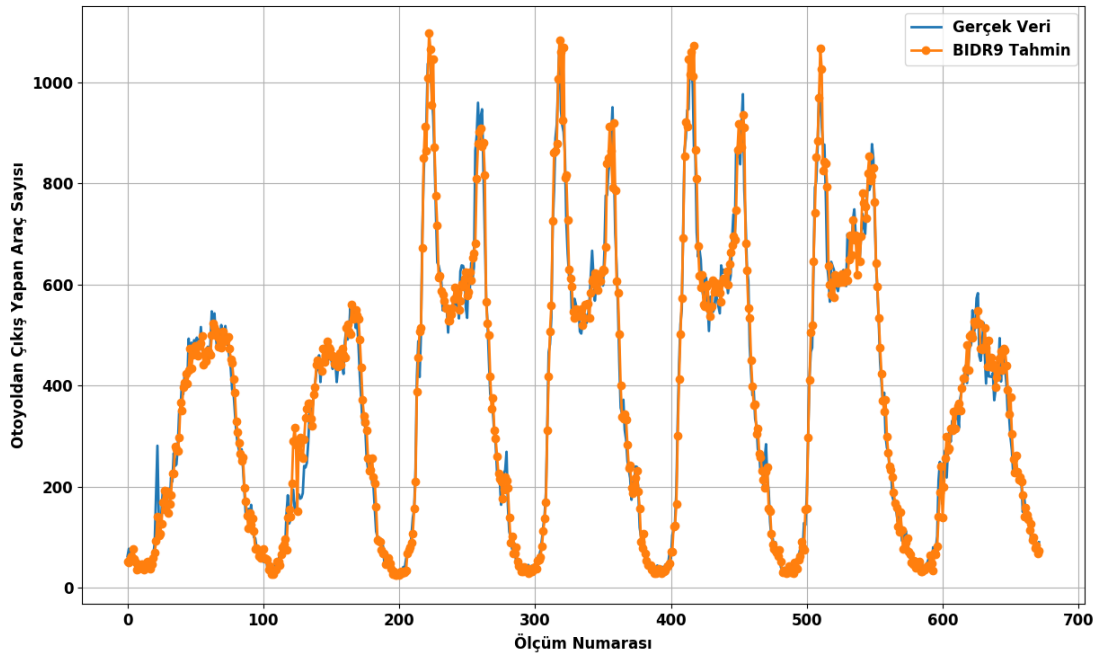
Model Adı	epoch	Batch size	Gizli Katman Sayısı	Düğüm Sayısı	Ağırlık Sayısı	Eğitim Veri Seti	Eğitim Süresi	MSE	MAE
BIDR01	100	72	2	101	92101	%70	1:10:01	101.48	22.40
BIDR02	100	72	2	101	92101	%80	1:12:30	87.18	1.19
BIDR03	100	72	2	101	92101	%90	1:19:17	37.46	3.20
BIDR04	50	72	2	101	92101	%90	0:39:09	38.46	4.03
BIDR05	250	72	2	101	92101	%90	3:19:04	37.67	4.39
BIDR06	500	72	2	101	92101	%90	6:22:14	38.85	3.15
BIDR07	100	36	2	101	92101	%90	2:20:29	37.98	1.75
BIDR08	100	144	2	101	92101	%90	0:46:54	37.71	4.04
BIDR09	100	72	4	201	212901	%90	2:41:46	36.60	3.01

Hata fonksiyonu (loss function) için MSE, optimizasyon yöntemi (optimizer) olarak ADAM kullanılmıştır ve Öğrenme Katsayısı (Learning Rate) 0.001 olarak sabit tutulmuştur.



Şekil 4.17. B-LSTM modelleri için, eğitim ve test veri setleri üzerinde hata fonksiyonların her iterasyon için sonuçları.

B-LSTM, bir gizli katmanda bulunan düğüme ondan iki önceki ve önceki düğüm çıkışlarını bağlayan bir LSTM türevidir. Şekil 4.17 ve Çizelge 4.7'de bulunan veriler incelendiğinde, B-LSTM modelleri arasında en başarılı en başarılı MSE değerinin, tahmin sonuçlarının gizli katman sayısının arttırılması ile elde edilen BIDR09 mimarisi olduğu bulunmuştur. MAE değerleri incelendiğinde, iterasyon sayısının büyümesi MAE değerinin büyütmektedir. En başarılı MAE değerine sahip mimari BIDR02'dir. Şekil 4.17'de X ekseni iterasyon numarasını, Y ekseni o iterasyona ait MSE değerini göstermektedir. Giriş katmanındaki verilerin grup büyüklüklerin arttırılması eğitim performansını artırmakta ve eğitim zamanını ciddi oranda düşürmektedir. Şekil 4.18'de BIDR09 mimarisine ait tahmin performansı gösterilmiştir.



Şekil 4.18. BIDR09 modeli ile 25 Ağustos 2019 ile 31 Ağustos 2019 tarihleri arasında 15 dakikalık aralıklarla otoyol araç çıkış sayısının gerçek ve tahmin değerlerinin grafiksel gösterimi.

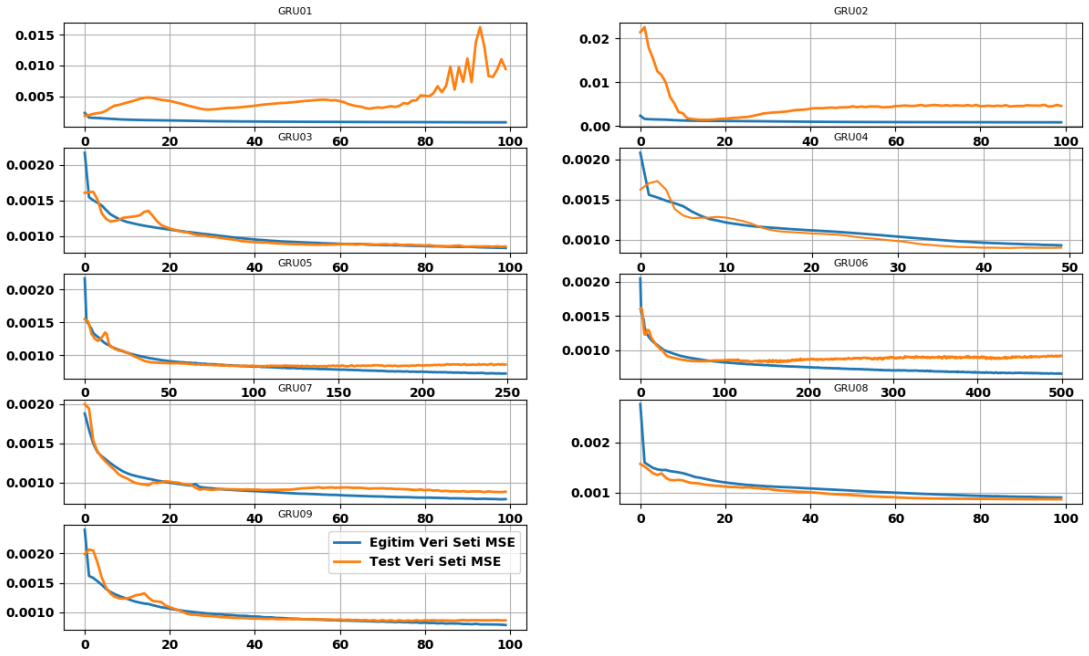
4.4.5. GRU ile Tasarlanan Model ve Bulgular

Tez çalışmasında GRU ile kurulan model, veri seti büyüklüğü, eğitim iterasyon sayısı (epoch), girdi olarak giriş katmanı düğümüne bir defada giriş yapan veri grubu

büyüklüğü (batch size), kullanılan ağırlık sayısı (param), gizli katman sayısı ve düğüm miktarı (node) bakımından değiştirilerek ayrı ayrı test edilmiştir.

Çizelge 4.8. Kullanılan GRU modellerine ait parametre bilgileri ve bulgular.

Model Adı	epoch	Batch size	Gizli Katman Sayısı	Düğüm Sayısı	Ağırlık Sayısı	Eğitim Veri Seti	Eğitim Süresi	MSE	MAE
GRU01	100	72	3	151	42201	%70	0:44:05	124.68	21.68
GRU02	100	72	3	151	42201	%80	0:46:44	87.01	0.28
GRU03	100	72	3	151	42201	%90	0:51:13	37.58	4.04
GRU04	50	72	3	151	42201	%90	0:25:27	38.53	3.73
GRU05	250	72	3	151	42201	%90	2:07:02	37.62	2.35
GRU06	500	72	3	151	42201	%90	4:06:14	38.95	3.08
GRU07	100	36	3	151	42201	%90	1:18:56	38.08	1.55
GRU08	100	144	3	151	42201	%90	0:30:13	37.75	2.05
GRU09	100	72	5	251	72501	%90	1:17:48	37.71	6.65

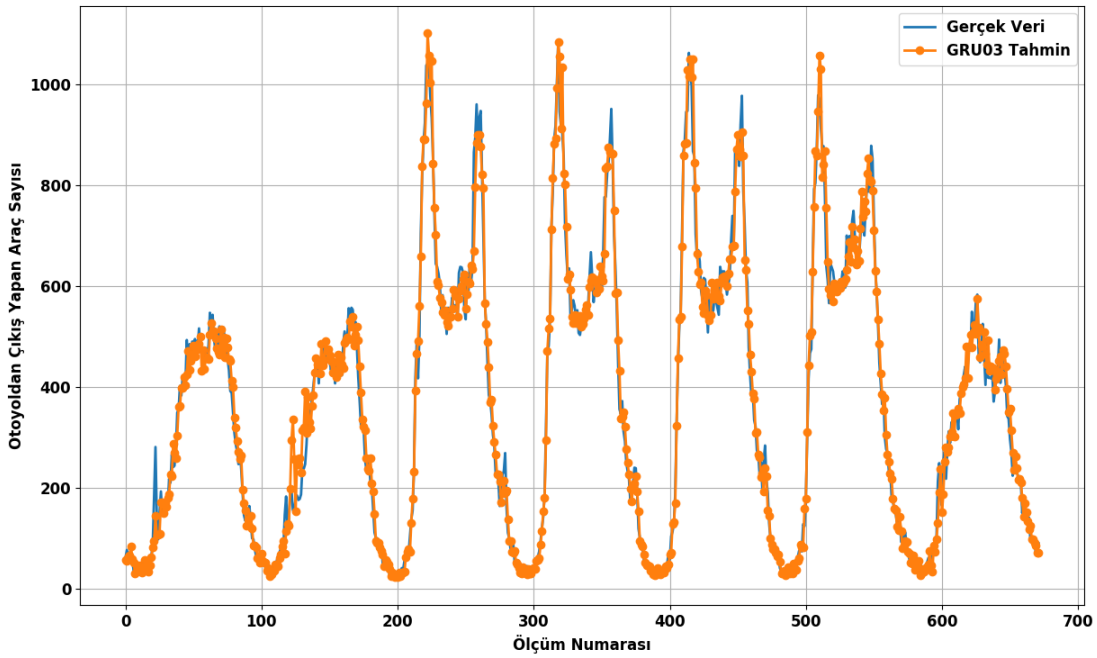


Şekil 4.19. GRU modelleri için, eğitim ve test veri setleri üzerinde hata fonksiyonların her iterasyon için sonuçları.

Oluşturulan modellere ait değerler Çizelge 4.8'de verildiği şekilde etiketlenmiştir. Şekjil 4.19'da modellere ait MSE hata değerlerinin iterasyonlar sırasındaki değişimleri gösterilmektedir. Tasarlanan bu model için tüm varyasyonlarda aktivasyon fonksiyonu olarak hiperbolik tanjant fonksiyonu kullanılmıştır. Hata

fonksiyonu için MSE, optimizasyon yöntemi olarak ADAM kullanılmıştır ve Öğrenme Katsayısı (Learning Rate olarak bilinir) 0.001 olarak sabit tutulmuştur.

GRU moelleri Çizelge 4.8'de X ekseni iterasyon numarasını, Y ekseni o iterasyona ait MSE değerini göstermektedir. Model sonuçları MSE değerleri bakımından incelendiğinde, modelin eğitim iterasyon sayısı artışı ile tahmin başarısının arttığını göstermektedir. MSE değerleri göz önünde bulundurulduğunda GRU03'ün en iyi tahmin sonucunu verdiği gözlenmiştir. MAE değerleri göz önünde bulundurulduğunda GRU02 en iyi model olarak tespit edilmiştir. Şekil 4.20'de GRU03 modeli ile 25 Ağustos 2019 ile 31 Ağustos 2019 tarihleri arasında 15 dakikalık aralıklarla otoyol araç çıkış sayısının gerçek ve tahmin değerlerinin grafiksel olarak gösterilmektedir.



Şekil 4.20. GRU03 modeline ait gerçekleşen ve elde edilen sonuçların karşılaştırılması.

BÖLÜM 5

SONUÇLAR

DÖ tekniklerinin trafik akış tahmin problemlerinde kullanımı, büyük veri çağına girmemiz, veriye kolay erişebilmemiz sayesinde sıklıkla başvurulan bir yöntem haline gelmiştir. Bu tez çalışması, çok bağlantıya sahip otoyollar için oluşacak taşıt sayısının tahmin edilmesini sağlamak, bu sayede insanlar için trafikte geçirecekleri sürelerde kazanç sağlayacakları alternatif yol seçimlerinin oluşmasını sağlamak, taşıtların trafikte çok fazla kalmaları ile oluşabilecek çevre kirliliğine neden olan emisyon salınımının azaltımı sağlamak ve trafik tahmininin önemli bir unsuru olduğu akıllı ulaşım sistemlerinin başarısının arttırılmasını amaçlanmaktadır.

Çizelge 5.1. Beş temel teknik için en iyi MSE değerli modeller.

Kullanılan Teknik	epoch	Batch size	Gizli Katman Sayısı	Düğüm Sayısı	Ağırlık Sayısı	Eğitim Veri Seti	Eğitim Süresi	MSE
GRU	100	72	3	151	42201	%90	0:51:13	37.58
B-LSTM	100	72	4	201	212901	%90	2:41:46	36.60
S-LSTM	250	72	2	101	36051	%90	1:44:16	36.68
LSTM	500	72	1	51	15851	%90	1:59:59	38:66
RNN	100	72	2	101	9051	%90	0:17:19	40.26

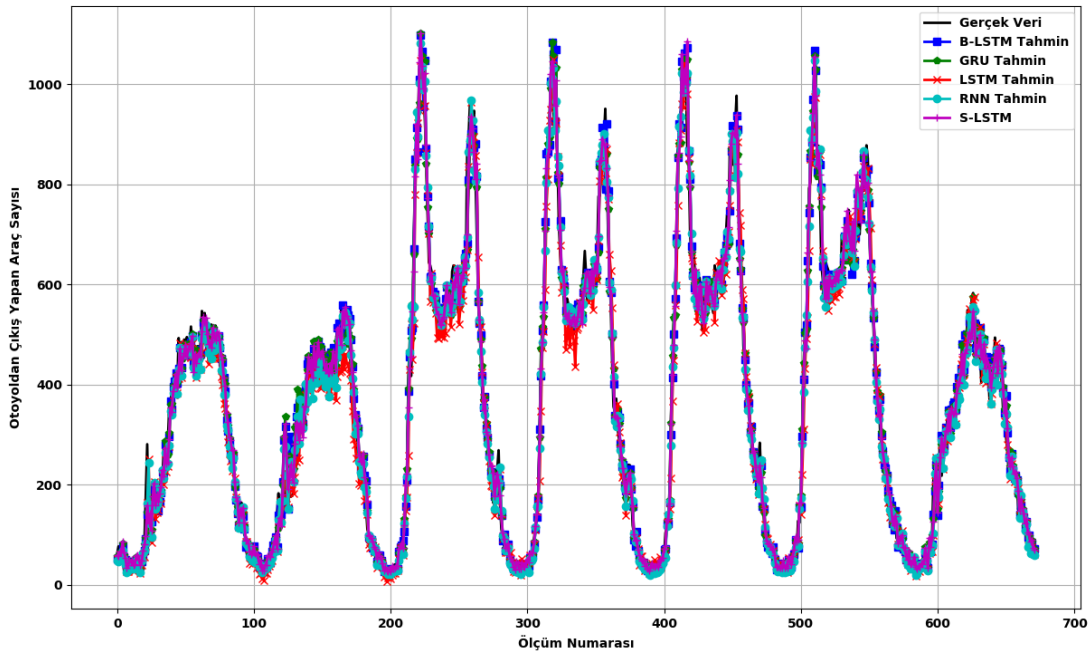
Çalışmada, DÖ teknikleri arasında bulunan, LSTM, S-LSTM, B-LSTM, GRU ve RNN teknikleri ile oluşturulmuş 44 farklı model tasarlanmış, tasarlanan modeller kullanarak otoyolun 15 dakika sonraki çıkış trafik yoğunluğu tahmin edilmiş ve elde edilen sonuçların başarıları karşılaştırılmıştır. Çizelge 5.1'de seçilen 5 temel teknik için en iyi MSE sonuçlarını veren model parametreleri verilmiştir. Çizelge 5.2'de MAE değerleri dikkate alınarak teknikler için en iyi parametrik değerler ve tahmin sonuçları gösterilmektedir.

Çizelge 5.2. Beş temel teknik için en iyi MAE değerli modeller.

Kullanılan Teknik	epoch	Batch size	Gizli Katman Sayısı	Düğüm Sayısı	Ağırlık Sayısı	Eğitim Veri Seti	Eğitim Süresi	MAE
GRU	100	72	3	151	42201	%80	0:46:44	0.28
B-LSTM	100	72	2	101	92101	%80	1:12:30	1.19
S-LSTM	100	36	2	101	36051	%90	1:19:30	1.51
LSTM	50	72	1	51	15851	%90	0:13:07	2.02
RNN	100	36	1	51	4001	%90	0:21:53	2.13

Çizelge 5.1'de verilen sonuçlar dikkate alındığında, otoyol trafik yoğunluk tahmini için en düşük MSE değerini, yani en iyi tahmin performansını gösteren DÖ tekniği B-LSTM olarak tespit edilmiştir. B-LSTM tekniği eğitim süresi açısından, diğer LSTM türevlerine göre daha fazla süre elde ederek zaman performansı olarak en kötü performansı sunmuştur. MSE değerleri açısından B-LSTM ve S-LSTM arasında çok az farklılık olduğu gözlenmiştir. Buna küçük farklılığa rağmen, eğitim süresi olarak S-LSTM çok daha iyi bir süre performansı elde etmiştir. Eğitim zamanında kısıtlama olduğu durumlarda RNN MSE değerinde en iyi olmasada tercih edilebilecek bir teknik olarak karşımıza çıkmaktadır. Eğitim sürelerin göz önüne alındığında MSE değeri en iyiler arasında olsada B-LSTM ağlarının eğitim süreleri oldukça fazladır. Şekil 5.1'de eğitim iterasyonları sırasında, test ve eğitim veri setleri için MSE değerleri tek bir grafikte verilmiştir.

MAE sonuçları değerleri incelendiğinde, GRU tekniği en iyi tahmin sonucunu üreten DÖ tekniği olarak bulunmuştur. Gizli katman sayısının eğitim süresinde neden olduğu artış, ağırlık sayısı sebebiyle oluşan karmaşıklık açısından daha iyi olan GRU sonuç değeri olarak %76 daha az hata sonuçu ile MSE değerleri bakımından en yi olan B-LSTM karşısında daha üstün görünmektedir.



Şekil 5.1. Beş teknik için en iyi modeller için 25 Ağustos 2019 ile 31 Ağustos 2019 tarihleri arasında 15 dakikalık aralıklarla otoyol araç çıkış sayısının gerçek ve tahmin değerlerinin grafiksel gösterimi.

Elde edilen MSE değerleri ve grafikler incelendiğinde, DÖ tekniklerinin trafik akış tahmini çalışmalarında başarılı bir şekilde kullanılabileceği sonucuna ulaşılmıştır. Modelin tahmin başarısı, veri seti büyüklüğü, gizli katman sayısı ve eğitim iterasyonun artışı ile pozitif yönlü bir ilişki içinde olduğu ancak buna karşın modelin eğitim süresinden büyük oranda negatif bir ilişki içinde olduğu sonucuna ulaşılmıştır. Çalışmada kullanılan teknikler açısından ayrı ayrı sonuçlar incelendiğinde aşağıdaki gibi özetlenebilir:

1. RNN ile geliştirilen modeller için tahmin performansı, gizli katman sayısının artışı ile gözle görülür bir başarı elde etmektedir. Eğitim süresi bakımından tüm teknik içinde en iyi zaman değerine sahip olan RNN'ler aynı sayısa gizli katman ve düğüme sahip olması durumunda eğitim iterasyon sayısının artması sonucunda eğitim zaman kaybı yaşamakta ancak MSE değerlerinde ciddi bir artış vermemektedir.

2. LSTM ağlarında, eğitim iterasyon sayısının arttırılması modelin tahmin başarısını yükselttiği sonucuna varılmıştır ancak karşılığında eğitim zamanından büyük

oranda kayıp yaşanmaktadır. LSTM başarısının arttırılması için kullanılan katman sayısının arttırılması gerektiği sonucuna varılmıştır ve bu S-LSTM ağları ile geliştirilen model sonuçları ile gözlenmiştir. LSTM ağlarda girdi katmanında bir defada eğitime verilen verinin büyüklüğünün arttırılması eğitimin başarısını arttırmamakta ancak eğitim süresinden dikkate değer oranda kazanç elde edilmesini sağlamaktadır.

3. S-LSTM ağları için, tahmin başarısının gizli katman sayısının ve eğitim iterasyon sayısının artışı ile yükseldiği sonucuna ulaşılmıştır. Ağın girdi katmanında bir defada eğitime verilen verinin büyüklüğünün arttırılması modelin tahmin başarısında olumsuz etki göstermesine rağmen, eğitim zamanın büyük oranda azalmasına neden olmaktadır. S-LSTM tahmin performansı bakımından en iyi modele en yakın değeri elde eden ikinci model olmasına karşın, süre önemli durumlar için en iyi model olarak tespit edilmiştir.

4. B-LSTM ağları, trafik akış tahmini için en iyi MSE değerini veren teknik olarak tespit edilmiştir. B-LSTM ağlarının başarısın daha çok artması için, kullanılan gizli katman sayısının artırılması gerektiği gözlenmiştir. B-LSTM ağlarının en olumsuz yönleri eğitim sürelerindeki aşırı büyüklüktür. Harcanan bu eğitim süresi, girdi katmanına alınan aynı anda veri büyüklüğünün artırılması ile azaltılabilmektedir.

5. GRU modellerinin tahmin başarıların artırılması için, eğitim iterasyon sayısı ve eğitim veri setinin büyüklüğünün arttırılmasının gerektiği sonucuna ulaşılmıştır. Ağın girdi katmanında aynı anda verilen veri büyüklüğünün azaltılması, diğer LSTM modellerinin aksine GRU modellerinde tahmin başarısını artırmamaktadır.

Trafik yoğunluğunu, mevsim şartları, trafik kazaları, etkinlikler ve yol çalışmaları gibi birçok etmen etkilemektedir. Bu etmenlerin tasarlanan model içinde temsil edilmesi, tahmin performansında artışa neden olacaktır. Gelecekteki çalışmalarda elde dilen trafik taşıt sayısı veri setine eklemeler yapılarak elde edilen MSE değeri azaltılmaya çalışılacaktır.

KAYNAKLAR

1. Lv, Y., Duan, Y., Kang, W., et al., "Traffic flow prediction with big data: A deep learning approach", ***IEEE Trans. Intell. Transp. Syst.***, 16, (2), 865–873 (2015).

2. Smith, B.L. and Demetsky, M.J., "Short-term traffic flow prediction: neural network approach", ***Transportation Research Record 1453***, 98–104 (1994).

3. Sun, S., Zhang, C. and Guoqiang, Y., "A Bayesian network approach to traffic flow forecasting", ***IEEE Intell. Transp. Syst. Mag.***, 1-7, 124–132 (2006).

4. Yin, H., Wong, S. C., Xu, J. and Wong, C. K., "Urban traffic flow prediction using a fuzzy-neural approach", ***Transportation Res.***, (2002).

5. Yu, G. and Zhang, C., "Switching ARIMA model based forecasting for traffic flow", ***IEEE international conference on acoustics, speech, and signal processing (ICASSP '04)***, (2004).

6. VanderVoort, V., Dougherty, M. and Watson, S., "Combining Kohonen maps with ARIMA time series models to forecast traffic flow", ***Transp. Res. C, Emerging Technol.***, vol. 4-5, 307–318 (1996).

7. Li, L., Lv, Y. and Wang, F.Y., "Traffic signal timing via deep reinforcement learning", ***IEEE/CAA J. Autom. Sin.***, 3, 247–254 (2016).

8. Bengio, Y., "Learning deep architectures for AI", ***Universite´ de Montreal***, Canada, (2007).

9. Schmidhuber, J., "Deep learning in neural networks: An overview", ***Neural Networks***, 61:85–117 (2015).

10. Wang, C., Li, X., Wang, A., Yang, F. and Zhou, X., "An Intelligent Transportation System Using RFI Based Sensors", ***In Proceedings of the IEEE 10th International Conference on High Performance Computing and Communications & Embedded and Ubiquitous Computing***, China, 337–344 (2013).

11. Yuan, Y., Xiong, Z. and Wang, Q., "An incremental framework for video-based traffic sign detection, tracking, and recognition", ***IEEE Trans. Intell. Transp. Syst***, 18(7), 1918–1929 (2017).

12. John, V., Yoneda, K., Qi, B., Liu, Z., & Mita, S., "Traffic light recognition invarying illumination using deep learning and saliency map", ***Proceedings Intelligent Transportation Systems Conference***, Qingdao, China, 2286–2291 (2014).

13. Zhu, Y., Zhang, C., Zhou, D., Wang, X., Bai, X. and Liu, W., "Traffic sign detection and recognition using fully convolutional network guided proposals", ***Neurocomputing***, 214, 758–766 (2016).

14. Bautista, C.M., Dy, C.A., Mañalac, M.I., Orbe, R.A. and Cordel, M., "Convolutional neural network for vehicle detection in low resolution traffic videos", ***IEEE Region 10 Symposium (TENSYMP)***, 277–281 (2016).

15. Qian, R.Q., Zhang, B.L. and Yue, Y., "Robust Chinese traffic sign detection and recognition with deep convolutional neural network", ***Proceedings of 11th International Conference on Natural Computation***, Zhangjiajie, China, 791–796 (2015).

16. Dwivedi, K., Biswaranjan, K. and Sethi, A., "Drowsy driver detection using representation learning", ***IEEE International Advance Computing Conference (IACC), Gurgaon***, India (2014).

17. Zang, D., Chai, Z.L., Zhang, J.Q., et al., "Vehicle license plate recognition using visual attention model and deep learning", ***J. Electron. Imaging***, (2015).

18. Niu, X., Zhu, Y., Cao, Q., Zhang, X., Xie, W. and Zheng, K., "An online-traffic-prediction based route finding mechanism for smart city", ***International Journal of Distributed Sensor Networks***, 18 (2015).

19. Huang, W., Song, G., Hong, H. and Xie, K., "Deep Architecture for Traffic Flow Prediction: Deep Belief Networks With Multitask Learning", ***IEEE Trans. Intell. Transp. Syst.***, 15, 2191–2201 (2014).

20. Ma, X., Yu, H., Wang, Y. and Wang, Y., "Large-scale transportation network congestion evolution prediction using deep learning theory", ***PLoS ONE*** (2015).

21. Polson, N.G. and Sokolov, V.O., "Deep learning for short-term traffic flow prediction", ***Transportation Research Part C: Emerging Technologies***, 79 1–17 (2017).

22. Ma, X., Dai, Z., He, Z., Ma, J., Wang, Y. and Wang, Y., "Learning traffic as images: A deep convolutional neural network for large-scale transportation network speed prediction", ***Sensors***, 17, 818 (2017).

23. Zhao, Z., Chen, W., Wu, X., Chen, P.C. and Liu, J., "LSTM network: A deep learning approach for short-term traffic forecast", ***IET Intell. Transp. Syst.***, 11, 68–75 (2017).

24. Fellendorf, M. and Vortisch, P., "Microscopic Traffic Flow Simulator VISSIM", ***Fundamentals of Traffic Simulation***, 63-94 (2010).

25. Koesdwiady, A., Soua R. and Karray F., "Improving traffic flow prediction with weather information in connected cars: a deep learning approach", ***IEEE T Veh Technol***, 65: 9508–9517 (2016).

26. Wang, J., Gu, Q., Wu, J., Liu, G., and Xiong, Z., "Traffic speed prediction and congestion source exploration: A deep learning method", ***Data Mining (ICDM), IEEE 16th International Conference***, 499–508 (2016).

27. Fu, R., Zhang, Z. and Li, L., "Using LSTM and GRU neural network methods for traffic flow prediction", ***Proceedings of the Youth Academic Annual Conference of Chinese Association of Automation (YAC)***, Wuhan, China, 324–328 (2016).

28. Yang, H.F., Dillon, T.S. and Chen, Y.P., "Optimized Structure of the Traffic Flow Forecasting Model with a Deep Learning Approach", ***IEEE Trans. Neural Netw. Learn. Syst***., 99, 1–11 (2016).

29. Zhang, Z., He, Q., Gao, J. and Ni, M., "A deep learning approach for detecting traffic accidents from social media data", ***Transport. Res. Part C: Emerg. Technol***, 86, 580–596 (2017).

30. Yuhan, J., Jianping, W. and Yiman D., "Traffic speed prediction using deep learning method", ***ITSC***, 1217–1222 (2016).

31. Wu, Y., Tan, H., Qin, L., Ran, B. and Jiang, Z., "A hybrid deep learning based traffic flow prediction method and its understanding", ***Transportation Research Part C: Emerging Technologies***, 90, 166–180 (2018).

32. Duan, Y., Lv, Y., Kang, W. and Zhao, Y., "A deep learning based approach for traffic data imputation", ***Intell. Transp. Syst***, 912–917 (2014).

33. Niu, X., Zhu, Y., and Zhang, X., "DeepSense: A novel learning mechanism for traffic prediction with taxi GPS traces", ***IEEE Global Communications Conference***, (2014).

34. Yi, H., Jung, H. and Bae, S., "Deep Neural Networks for traffic flow prediction", ***IEEE International Conference on Big Data and Smart Computing (BigComp)***, Jeju, 328-331 (2017).

35. Yu, R., Li Y., Shahabi, C., Demiryurek, U., and Liu, Y., "Deep learning: A generic approach for extreme condition traffic forecasting", ***SIAM International Conference on Data Mining (SDM)***, (2017).

36. Tayara, H., Soo, K. G. and Chong, K. T., "Vehicle detection and counting in high-resolution aerial images using convolutional regression neural network", ***IEEE***, 6, 2220–2230 (2018).

37. Pan, C., Yan, Z., Xu, X., et al., "Vehicle logo recognition based on deep learning architecture in video surveillance for intelligent traffic system, Smart and Sustainable City", ***IET International Conference***, IET, 123-126 (2013).

38. He, X. and Zeng D., "Real-time pedestrian warning system on highway using deep learning methods", ***International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS),*** 701–706, IEEE (2017).

39. Lu, M., Hu, Y., and Lu, X., "Driver Detection Based on Deep Learning", ***Journal of Physics: Conference Series***, 1069(1), 012118 (2018).

40. Peppa, M. V., Bell, D., Komar, T. and Xiao, W., "Urban traffic flow analysis based on deep learning car detection from CCTV image series", ***Int. Arch. Photogramm. Remote Sens***, Spatial Inf. Sci., 499–506 (2018).

41. Dominguez-Sanchez, A., Cazorla, M. and Orts-Escolano, S., "A New Dataset and Performance Evaluation of a Region-Based CNN for Urban Object Detection", ***Electronics***, 7, 301 (2018).

42. Zhu, L., Guo, F., Krishnan, R. and Polak, J.W., "A deep learning approach for traffic incident detection in urban networks", ***21st International Conference on Intelligent Transportation Systems (ITSC)***, IEEE, 1011–1016 (2018).

43. Muller, J. and Dietmayer, K., "Detecting traffic lights by single shot detection", ***21st Int. Conf. Intelligent Transportation Systems***, IEEE, 2016, 42–348 (2016).

44. Hicham, B., Ahmed, A. and Mohammed, M., "Vehicle type classification using convolutional neural network", ***Colloquium in Information Science and Technology, CIST***, 8596500, 313-316 (2018).

45. Zhao, D. and Li, H., "Forward vehicle detection based on deep convolution neural network", ***AIP Conference Proceedings***, 2073, 5090761 (2019).

46. Suhao, L., Jinzhao, L., Guoquan, L., Tong, B., Huiqian, W., and Yu, P., "Vehicle type detection based on deep learning in traffic scene.", ***Proc. Computer Science***, 131, 564-572, (2018).

47. Sommer, L., Nie, K., Schumann, A., Schuchert, T. and Beyerer, J., "Semantic labeling for improved vehicle detection in aerial imagery", ***IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)***, 1–6, (2017).

48. Sommer, L., Schumann, A., Schuchert, T. and Beyerer, J., "Multi feature deconvolutional faster R-CNN for precise vehicle detection in aerial imagery", ***WACV***, IEEE (2018).

49. Chen, Y., Tao, G., Ren, H., Lin, X. and Zhang, L., "Accurate seat belt detection in road surveillance images based on CNN and SVM," ***Neurocomputing***, 274, 80–87 (2018).

50. Wang, X., Zhang, W., Wu, X., Xiao, L., Qian, Y. and Fang, Z., "Real-time vehicle type classification with deep convolutional neural networks", ***Journal of Real-Time Image Processing*** **(**2017).

51. Chollet, F., and Joseph, J. A., "Deep Learning with R", ***Manning Publications***, Shelter Island, 1-112 (2018).

52. Patterson, J. and Gibson, A., "Deep Learning: A practitioner's Approach", ***O'Reilly Press***, ISBN:978-1-491-91425-0, 41-114, 125-165 (2017).

53. Haykin, S., "Neural Networks and Learning Machines (3d Edition)", ***Prentice Hall***, ISBN-13:978-0-13-147139-9, 1-68 (2009).

54. Basheer, I.A. and Hajmeer, M., "Artificial neural networks: fundamentals,computing, design, and application", ***Journal of Microbiological Methods,*** 43 3–31 (2000).

55. Zupan, J. and Gasteiger, J., "Neural networks: a new method for solving chemical problems or just a passing phase?", ***Anal Chim. Acta,*** 248 1–30 (1991).

56. Souza, A.M. and Soares, F.M, "Neural Network Programming with Java", ***Packt Publishing Ltd***, Birmingham, 1-79 (2016).

57. Yao, X., "Evolving artificial neural networks", ***Proceedings of the IEEE***, 87(9):1423–1447 (1999).

58. Demuth, H., Beale, M. and Hagan, M., "User Guide, Neural Networks Toolbox™ 6", ***The MathWorks, Inc***., ISBN:0-9717321-0-8, 2-25, 3-5, 8-6 (2009).

59. Kishan, M., Chilukuri, K.M. and Sanjay, R., "Elements of Artificial Neural Networks", ***ISBN13:0-262-13328-8***, 16-22, 70-80 (1996).

60. Hecht-Nielsen, R., "Theory of the back propagation neural network", ***In Proceedings of the International Joint Conference on Neural Networks***, San Diego, 593-606 (l989).

61. Specht, D.F., "Probabilistic neural networks for classification, mapping, or associative memory", ***IEEE Int. Conf. Neural Nerworks***, 1, 525-532 (1988).

62. Rabunal, J.R. and Dorrado, J., "Artificial Neural Networks in Real-Life Applications", Hershey, ***Idea Group Pub***, 1-47 (2006).

63. Tavanaei, A., Ghodrati, M., Kheradpisheh, S. R., Masquelier, T. and Maida, A. S., "Deep learning in spiking neural networks", ***Neural Networks*** (2018).

64. Bishop, C.M., "Pattern Recognition and Machine Learning", ***Springer***, New York, ISBN13: 978-0387-31073-2, 225-284 (2006).

65. Bai, J. and Ng, S., "Tests for skewness, kurtosis, and normality for time series data", ***Journal of Business and Economic Statistics***, 23: 49–60 (2005).

66. Ozkan, C. and Erbek F. S., "The comparison of activation functions for multispectral Landsat TM image classification," ***Photogrammetric Engineering and Remote Sensing***, 6911, 1225–1234 (2003).

67. Jin, X., Xu, C., Feng, J., Wei, Y., Xiong, J., and Yan, S., "Deep learning with s-shaped rectified linear activation units", ***Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)*** (2015).

68. LeCun Y., Bengio Y. & Hinton G., "Deep learning", ***Nature*** 521, 436–444 (2015).

69. Krizhevsky, A., Sutskever, I. and Hinton, GE., "Imagenet classification with deep convolutional neural networks", ***Advances in neural information processing systems*** (2012).

70. Maas, A. L., Hannun, A. Y. and Ng, A. Y., "Rectifier nonlinearities improve neural network acoustic models", ***ICML*** (2013).

71. Bishop, C., "Novelty detection and neural network validation", ***Proceedings of the IEEE Conference on Vision, Image and Signal Processing***, 141, IET, 217–222 (1994).

72. Hagan, M.T., Demuth, H.B. and Beale, M., "Neural Network Design", Boston, ***PWS Publishing Co.***, 10-04 (1996).

73. Nasr, G. E., Badr, E. A., and Joun, C., "Cross Entropy Error Function In Neural Networks: Forecasting Gasoline Demand", ***FLAIRS-02 Proceedings of the AAAI*** (2002).

74. Yam, J.Y.F., Chow, T.W.S., "A weight initialization method for improving training speed in feedforward neural network", ***Neurocomput***, 30:219–232 (2000).

75. Glorot, X. and Bengio, Y., "Understanding the difficulty of training deep feedforward neural networks", ***AISTATS*** (2010).

76. Saxe, A. M., McClelland, J. L., and Ganguli, S., "Exact solutions to the nonlinear dynamics of learning in deep linear neural networks", ***CoRR*** (2013).

77. İnternet : Github, "Orthogonal Initialization in Convolutional Layers", **https://hjweide.github.io/orthogonal-initialization-in-convolutional-layers** (2019).

78. Le, Q. V., Ngiam, J., Coates, A., Lahiri, A., Prochnow, B. and Ng, A. Y., "On optimization methods for deep learning"***, Proc. 28th Int. Conf. Machine Learning***, 265–272 (2011).

79. Sutskever, I., Martens, J., Dahl, G. E. and Hinton, G. E., "On the importance of initialization and momentum in deep learning", ***In ICML (3)***, 28 of JMLR Proceedings, 1139–1147 (2013).

80. İnternet : Towarsd Data Science, "10 Gradient Descent Optimisation Algorithms + Cheat Sheet", **https://towardsdatascience.com/10-gradient-descent-optimisation-algorithms-86989510b5e9** (2019).

81. Kingma, D. P. and Ba J. L., "Adam: A method for stochastic optimization", ***arXiv preprint,*** arXiv:1412.6980 (2014).

82. Chollet, F., "Deep Learning with Python", ***Manning Publications***, 1-56, 196-207 (2017).

83. LeCun, Y., Bengio, Y. and Hinton, G., "Deep learning", ***Nature***, 521 436–444 (2015).

84. Goyal, P., Pandey, S., and Jain, K., "Deep Learning for Natural Language Processing", ***Apress***, 35 -75, 119-168 (2018).

85. Srivastava, N., Hinton G., Krizhevsky, A., Sutskever, I. and Salakhutdinov R., "Dropout: A simple way to prevent neural networks from overfitting", ***The Journal of Machine Learning Research***, 1929–1958 (2014).

86. Buduma, N. and Locascio, N., "Fundamentals of Deep Learning. Designing Next-Generation Machine Intelligence Algorithms", ***O'Reilly Media***, 172-217 (2017).

87. Pattanayak, S., "Pro Deep Learning with TensorFlow", ***Apress***, New York, ISBN 978-1-4842-3095-4, 153-278 (2017).

88. Le, Q.V., "A tutorial on deep learning part 2: Autoencoders, convolutional neural networks and recurrent neural networks", ***Google Brain***, 1–20 (2015).

89. Hochreiter, S. and Schmidhuber J., "Long short-term memory", ***Neural computation***, 9(8):1735–1780 (1997).

90. Chung, J., Gülçehre, Ç., Cho, K., and Bengio, Y., "Empirical evaluation of gated recurrent neural networks on sequence modeling", ***CoRR***, abs/1412.3555 (2014).

91. Krizhevsky, A., Sutskever, I. and Hinton, G., "Imagenet classification with deep convolutional neural networks", ***NIPS*** (2012).

92. İnternet : Highways England, "Highways England - Open Data", **http://tris.highwaysengland.co.uk/detail/trafficflowdata** (2019).

93. İnternet : Roads, M57 and A5300 | Roads.org.uk, **https://www.roads.org.uk/motorway/m57/** (2019).

94. Bandari, R., Asur, S. and Huberman, B. A., "The pulse of news in social media: Forecasting popularity", **ICWSM'12**, pages 26–33, (2012).

95. Zhengping, C., Sanjay, P., Kyunghyun, C., David, S. and Yan, L., "Recurrent neural networks for multivariate time series with missing values", ***Scientific Reports***, 8(1):6085 (2018).

ÖZGEÇMİŞ

Muhammet Esat Özdağ Konya'da doğdu; ilk, orta ve lise eğitimini yine aynı şehirde tamamladı. 2001 yılında Konya Selçuk Üniversitesi'nde Bilgisayar Mühendisliği Bölümü'nde lisans eğitimine başladı. 2017 yılında Karabük Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı'nda yüksek lisans eğitimine başladı. 2003 ve 2015 yılları arasında bilişim sektöünrde birçok kuruluşta yazılım geliştirici, analist ve ekip lideri görevlerinde bulundu. 2015 Yılında Tokat Gaziosmanpaşa Üniversitesinde Öğretim Görevlisi olarak göreve başladı ve halen bu görevi sürdürmektedir.

ADRES BİLGİLERİ

Adres: Tokat Gaziosmanpaşa Üniversitesi, Taşçiftlik/ TOKAT

Tel: 0 (356) 252 16 16
E-Posta: muhammetesat.ozdag@gop.edu.tr