T.C.

BİLECİK ŞEYH EDEBALİ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

DERİN ÖĞRENME KULLANILARAK BİLECİK İLİ RÜZGAR HIZI TAHMİNİ

YÜKSEK LİSANS TEZİ

BERKAY ÖZKAY

TEZ DANIŞMANI

DR. ÖĞR. ÜYESİ SALİM CEYHAN

BİLECİK, 2021

10407128

T.C.

BİLECİK ŞEYH EDEBALİ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

DERİN ÖĞRENME KULLANILARAK BİLECİK İLİ RÜZGAR HIZI TAHMİNİ

YÜKSEK LİSANS TEZİ

BERKAY ÖZKAY

TEZ DANIŞMANI

DR. ÖĞR. ÜYESİ SALİM CEYHAN

BİLECİK, 2021

10407128

BEYAN

Derin Öğrenme Kullanılarak Bilecik İli Rüzgâr Hızı Tahmini adlı yüksek lisans tezi hazırlık ve yazımı sırasında bilimsel ahlak kurallarına uyduğumu, başkalarının eserlerinden yararlandığım bölümlerde bilimsel kurallara uygun olarak atıfta bulunduğumu, kullandığım verilerde herhangi bir tahrifat yapmadığımı, tezin herhangi bir kısmının Bilecik Şeyh Edebali Üniversitesi veya başka bir üniversitede başka bir tez çalışması olarak sunulmadığını beyan ederim.

<table>
<tr><td colspan="4">Bu çalışmanın,
Bilimsel Araştırmalar Projeleri (BAP), TÜBİTAK veya benzeri kuruluşlarca desteklenmesi durumunda; projenin ve destekleyen kurumun adı proje numarası ile birlikte beyan edilmelidir.</td></tr>
<tr><td colspan="2">DESTEK ALINMIŞTIR</td><td></td><td>DESTEK ALINMAMIŞTIR</td><td></td></tr>
<tr><td colspan="4">Destek alındı ise;</td></tr>
<tr><td colspan="4">Destekleyen Kurum:</td></tr>
<tr><td colspan="3">Desteğin Türü</td><td>Proje Numarası</td></tr>
<tr><td colspan="2">1- BAP (Bilimsel Araştırma Projesi)</td><td></td><td rowspan="3"></td></tr>
<tr><td colspan="2">2- TÜBİTAK</td><td></td></tr>
<tr><td colspan="3">Diğer; .....</td></tr>
</table>

Berkay ÖZKAY

Tarih

İmza

…….....

ÖN SÖZ

Bu tez yeterlik çalışmasının yazılmasında, çalışmamı sahiplenerek takip eden danışmanım Sayın Dr. Öğr. Üyesi Salim CEYHAN'a değerli katkı ve emekleri için teşekkürlerimi ve saygılarımı sunarım.

Savunma sınavı sunumu sırasında değerli jüri üyeleri Sayın Doç. Dr. Emre DANDIL'a ve Sayın Doç. Dr. Mehmet KOÇ'a çalışmamın son haline gelmesindeki değerli katkıları adına teşekkürlerimi ve saygılarımı sunarım.

Son olarak bu günlere ulaşmamdaki emekleri adına değerli aileme teşekkür ederim.

Berkay ÖZKAY

ÖZET

DERİN ÖĞRENME KULLANILARAK BİLECİK İLİ RÜZGAR HIZI TAHMİNİ

Son zamanlarda yapılan çalışmalarda, rüzgâr hızının kısa vadeli tahmininde derin öğrenme tabanlı modeller yaygın olarak kullanılmaktadır. Bu tez kapsamında rüzgâr hızları zaman serisi tipinde olduğundan buna en uygun derin öğrenme yöntemlerinden biri olan LSTM (Long Short-Term Memory) modeli kullanılmıştır. LSTM sinir ağı, derin öğrenme RNN ağının tüm avantajlarına sahip olmakla birlikte aynı zamanda RNN ağının kaybolan gradyan sorununun üstesinden geldiğnden güçlü doğrusal olmayan işlem yapma yeteneğine sahiptir ve sabit olmayan rüzgar hızı tahmini için uygundur.

Çalışmada ilk olarak, Bilecik Meteoroloji Müdürlüğü'nden alınan saatlik rüzgar hızı ham verilerindeki eksik rüzgar hızı verileri ön işlemeden geçirilerek tamamlanmıştır. Daha sonra, Bilecik Meteoroloji Müdürlüğü'nden alınan 10 yıllık (2010-2019) işlenmiş saatlik rüzgâr hızı verileri çeşitli giriş büyüklükleri için tasarlanan derin öğrenme modelimizde eğitime sokularak belirli zaman aralıklarındaki rüzgar hızlarının en iyi doğruluklarda tahminini gerçekleştirilmiştir. Bunun için 10 yıllık saatlik rüzgar hızı verileri, oluşturduğumuz 3 katmanlı LSTM modelinde verilerin %67'si eğitim ve %33'ü test için kullanılarak eğitilmiş daha sonra 2020 yılının aralık ayına ait modelin hiç görmediği saatlik rüzgar hızı verileri test olarak alınarak iki tip LSTM modelinde Ocak ayının 1 gün, 2 gün, 1 hafta ve 1 aylık tahminleri yapılmıştır. Daha iyi tahmin değerleri elde etmek amacıyla, yapı olarak farklı iki tip LSTM modeli, tek değişkenli çift yönlü (Univarate Bidirectional) ve çok adımlı vektör çıkışlı (Multi-Step Vector Output) kullanılmış ve belirli aralıklarda saatlik rüzgar hızı tahmininde hangisinin daha uygun olduğu araştırılmıştır. Çalışmanın diğer bir sonucu ise LSTM modellerinde kullanılan farklı büyüklükteki 1 ve 10 yıllık saatlik rüzgar hızları verilerinin modellerin doğruluğuna etkisi incelenmiş ve sonuçlar karşılaştırılmıştır.

Anahtar Kelimeler: Derin Öğrenme, Rüzgar Hızı Tahmini, Uzun Kısa-Süreli Hafıza (LSTM), Zaman Serileri.

ABSTRACT

WIND SPEED FORECAST IN BILECIK PROVINCE USING DEEP LEARNING

In recent studies, deep learning-based models are widely used for short-term prediction of wind speed. Within the scope of this thesis, LSTM (Long Short-Term Memory) model, which is one of the most suitable deep learning methods, was used since wind speeds are time series type. The LSTM neural network has all the advantages of a deep learning RNN network, but also has strong nonlinear processing capability as it overcomes the vanishing gradient problem of the RNN network and is suitable for unsteady wind speed prediction.

In the study, the missing wind speed data in the hourly wind speed raw data obtained from the Bilecik Meteorology Directorate were completed by preprocessing. Then, 10 years (2010-2019) processed hourly wind speed data obtained from Bilecik Meteorology Directorate were put into training in our deep learning model designed for various input sizes, and the best accurate prediction of wind speeds at certain time intervals was realized. For this, 10 years of hourly wind speed data was trained by using 67% of the data for training and 33% for testing in the 3-layer LSTM model we created. Forecasts for 1 day, 2 days, 1 week and 1 month of January were made. In order to obtain better forecast values, two different types of LSTM models, Univarate Bidirectional and Multi-Step Vector Output, were used and it was investigated which one is more suitable for hourly wind speed forecasting at certain intervals. Another result of the study is the effect of different sizes of 1-year and 10-year hourly wind speed data used in LSTM models on the accuracy of the models and the results are compared.

Keywords: Deep Learning, Wind Speed Prediction, Long Short-Term Memory (LSTM), Time Series.

İÇİNDEKİLER

Sayfa No

TABLOLAR LİSTESİ

Sayfa No

ŞEKİLLER LİSTESİ

Sayfa No

SİMGELER VE KISALTMALAR LİSTESİ

LSTM : Long-Short Term Memory

M.Ö : Milattan Önce

IEA : International Energy Agency (Uluslararası Enerji Topluluğu)

EMO : Elektrik Mühendisleri Odası

GWH : GigaWatt Hours (GigaWatt Saat)

GPU : Graphics Processing Unit (Grafik İşlemci Birimi)

CPU : Central Process Unit (Merkezi İşlem Birimi)

XOR : eXclusive OR (Özel Veya)

IEEE : Institute of Electrical and Electronics Engineers (Elektrik ve Elektronik Mühendisleri Enstitüsü)

LVQ : Learning Vector Quantization (Doğrusal Vektör Nicelemesi)

CNN : Convolutional Neural Network (Evrişimli Sinir Ağları)

D-FFNN : Deep Feed Forward Neural Network (Derin İleri Beslemeli Sinir Ağları)

RNN : Recurrent Neural Network (Yinelenene Sinir Ağları)

SVM : Support Vector Machine (Destek Vektör Makinesi)

ILSVRC : ImageNet Large Scale Visual Recognition Challenge (ImageNet Büyük Ölçekli Görsel Tanıma Yarışması)

MLP : Multi Layer Percecptron (Çok Katmanlı Algılayıcı)

NLP : Nature Language Processing (Doğal Dil İşleme)

BM : Boltzman Machine (Boltzman Makinesi)

RBM : Restricted Boltzman Machine (Kısıtlamalı Boltzman Makinesi)

$\boldsymbol{\sigma}$ **:** Sigma (Sigmoid)

$\boldsymbol{\theta}$ **:** Theta (Ağırlık)

$\boldsymbol{\nabla}$ **:** Nabla (Kısmi Türev)

$\boldsymbol{\eta}$ **:** Eta (Öğrenme Katsayısı)

$\boldsymbol{\beta}$ **:** Beta (Hiperparametre)

Batch_size : Yığın_boyutu

1. GİRİŞ

Bu tezin ana konusu olan "rüzgar hızı tahmini" işlemlerinin ana kullanım alanları rüzgar tribünleridir. Rüzgar hızına bağlı olarak ilgili rüzgar tribünün ileri ki zamanlarda üreteceği elektrik enerjisi hesaplanır. Ani rüzgar hızları rüzgar tribünlerinde ciddi hasarlara yol açabilmektedir. Bu tarz maddi kayıpların yaşanmaması için rüzgar hızı tahmini sonucunda tribüne zarar verebilecek bir aşırılık fark edilirse rüzgar tribünleri kapatılır ve tribünler bu şekilde korunur. Potansiyel rüzgar tribünü kurulacak olan alanların belirlenmesinde de rüzgar hızı tahmini verileri kullanılabilinir.

Rüzgar hızı tahmini literatürde farklı parametrelerin (nem, basınç, sıcaklık vb.,) işlenmesi veya rüzgar hızlarını bir zaman serisine dönüştürüp işlenmesi şeklinde 2 farklı türde yapılabilmektedir. Bu tezde, Bilecik Meteoroloji Müdürlüğünden alınan Bilecik Merkez İlçesinin geçmiş yıllara ait saat başlarında okunan rüzgâr hızı verilerinden zaman serisi oluşturulmuş, derin öğrenme modelleri kullanılarak gelecek zamanlara ait saatlik rüzgâr hızları tahmin (forecast) edilmiş ve derin öğrenme modeline göre işlem sonuçlarının doğruluk değerleri belirlenmiştir.

Bilecik ili için daha önceden yapılan rüzgar hızı tahmini ile ilgili çalışmalar; M. Recep Minaz ve ark., *"Uyarlanır Sinir Bulanık Sistemi (ANFIS) ve doğrusal çoklu regresyon analizi yöntemleri kullanılarak rüzgar hızı, basınç ve sıcaklık tahminleri"* ve E. Dokur ve ark., "*Rüzgar hızı dağılımının modellenmesinde kullanılan Weibull dağılımı için parametre tahmini yaparak Bilecik ili için rüzgar hızı modeli elde edildi.*" şeklindedir.

Çalışmamızı yukarıda verilen iki çalışmadan faklı kılan özellikler aşağıda verilmiştir:

- Bugüne kadar ki, Bilecik ili rüzgâr hızı tahmin çalışmalarında ya yapay sinir ağları ya da Weibull dağılımına ait parametre tahminleri kullanılarak yapılmıştır. Bu yönüyle çalışmamız, Bilecik ilinin rüzgâr hızı tahmininde derin öğrenme modeli kullanan ilk çalışma olmaktadır.
- Bu çalışmamızda, derin öğrenme modeli (LSTM) yapısının iki farklı türü kullanılarak hangi yapının Bilecik ili için rüzgâr hızı tahmininde daha iyi sonuç vereceğinin araştırması yapılmıştır.
- Eğitimde veri sayısı büyüklüklerinin derin öğrenme modeli sonuçlarına etkisinin ne olduğunun görülmesi amacıyla 1 yıllık ve 10 yıllık kapsamlı veriler bu iki derin öğrenme modeline uygulanmıştır. Yapılan bu uygulamayla modelin eğitimi için

kullanılan veri seti büyüklüğünün rüzgar hızı tahmin sonuçlarındaki etkisi incelenmiştir.

1.1. Enerji

'Enerji' kavramı, insanlığın var olduğu tarihten itibaren her zaman önemli bir kavram olarak ön plana çıkmıştır. Fizik biliminde kısaca 'bir cismin veya bir sistemin iş yapabilme yeteneği' olarak tanımlanan 'enerji', yıllar geçtikçe ve insanlığın teknolojiyi geliştirmesi ile birlikte var olan önemi ve kullanım büyüklüğü gün geçtikçe artarak günümüze kadar gelmiştir. Nitekim dünya nüfusu 1500'lü yıllardan günümüze kadar yaklaşık 14 kat artarken, 'enerji tüketimindeki artış' 115 kat olmuştur. Bu sonuç da bize enerji tüketimindeki artışın nüfustaki artıştan çok daha fazla olup 8 katını geçtiği, görülmektedir (Harari, 2012: 247). Enerjinin bu derece önemli olması dünya devletlerinin yenilenebilir temiz enerjiler üzerinde yatırım ve araştırma yapmalarına neden olmuştur. Bu nedenle enerji konusunda yapılan akademik çalışmaların giderek ivme ve önem kazanması bizi bu araştırmayı yapmaya sevk etmiştir.
Enerji kaynakları kullanım alanları ve dönüştürüldüğü türe göre iki ana gruba ayrılmıştır (Kaya & Koç, 2015: 37).

ENERJİ KAYNAKLARI

Kullanılışlarına Göre Enerji Kaynakları
A) Yenilenemez (Tükenen)
a) Fosil Kaynaklı
-Kömür
-Petrol
-Doğal gaz
b) Çekirdek Kaynaklı
-Uranyum
-Toryum
B) Yenilenebilir (Tükenmeyen) (Alternatif Enerji)
-Hidrolik
-Günes
-Biyokütle
-Rüzgar
-Jeotermal
-Dalga, Gel-git
-Hidrojen

Dönüştürülebilirliklerine Göre Enerji Kaynakları
A) Birincil (Primer)
-Kömür
-Petrol
-Doğalgaz
-Nükleer
-Biyokütle
-Hidrolik
-Günes
-Rüzgar
-Dalga, Gel-git
B) İkincil (Sekonder)
-Elektrik, Benzin, Mazot, Motorin
-İkincil kömür
-Kok, Petrokok
-Hava gazı
-Sıvılaştırılmış petrol gazı (LPG)

Şekil 1.1. Enerji kaynakları ve gruplandırılmaları

Kaynak: (Kaya & Koç, 2015: 37)

1.1.1. Alternatif enerji ve rüzgâr enerjisi

Rüzgar enerjisi literatürde alternatif enerji kaynaklarının içinde bulunmaktadır. Dünya'da alternatif enerji kaynaklarına yönelimin başlangıç noktası olarak İngiltere'de buharlı makinelerin icadı ile başlayan Sanayi Devrimi'ni sayabiliriz. Sanayi Devrimi ile birlikte üretimin ve enerji ihtiyacının artması nedeniyle gerekli olan kömür, petrol ve doğal gaz gibi fosil kaynaklı klasik enerji kaynaklarının kullanım oranı da artmıştır. Fakat bu yapı sürdürülebilir bir yapı değildir. Bunun nedenlerini sıralayacak olursak:

- Bu kaynaklar tükenebilir bir yapıdadırlar.
- Yukarıdaki duruma bağlı olarak kaynaklar azaldıkça bunları elde etmek için gereken maddi harcamalar da artmaktadır.
- Araştırma, çıkarma, iletme gibi çeşitli farklı maliyet kalemleri bulunmaktadır.
- Fosil bazlı kaynaklar her yerde homojen bulunabilir bir yapıda değildir. Bazı ülkelerde bu enerji kaynaklarına sahip olma oranı çok yüksek olduğundan dolayı ortaya çıkan eşitsizlik bu tarz kaynaklara sahip olmayan diğer ülkeler için büyük bir dezavantaj yaratmaktadır (Çıtak & Kılınç Pala, 2016: 82).

Fosil bazlı klasik enerji kaynaklarındaki sıkıntı durumunun gözlendiği en büyük olaylardan birisi 1970'lerde patlak veren petrol krizidir. Bu kriz sonucunda Dünya üzerindeki ülkeler buna benzer bir sıkıntı durumunda kriz yaşamamak için alternatif enerji kavramına yönelmişlerdir. Alternatif enerjinin artı yönleri:

- Fosil bazlı kaynaklardaki gibi bir coğrafi konumdan dolayı büyük bir avantaja sahip olma durumu alternatif enerji kaynakları açısından önemli bir sorun teşkil etmemektedir. Çünkü her ülke az ya da çok alternatif enerji kaynaklarından biri ya da birkaçına sahip olabilmektedir.
- Fosil kaynakların kullanımı sonrası ortaya çıkan atık maddeler doğaya zarar vermektedirler. Alternatif enerji kaynaklarının kullanımı sonrası ortaya herhangi bir atık çıkmadığından dolayı doğaya herhangi bir zarar verme durumundan söz edilememektedir.
- Fosil bazlı kaynakların en büyük sorunlarından biri ise tüketime bağlı olarak ilerleyen zamanlardaki tükenme olasılığı alternatif enerji kaynakları açısından neredeyse yok gibidir (İnançlı & İnal, 2018: 105).

Rüzgar, Dünya'nın engebeli yeryüzü yapısı sebebiyle Güneş'ten gelen ışınların farklı konumlarda bulunan havayı farklı şekillerde ısıtması ve bunun sonucunda ortaya çıkan farklı

sıcaklık ve basınçlara sahip olan havanın sirkülasyon işlemi gerçekleştirmesine denmektedir. Ayrıca karaların ısınma ve soğuma hızı denizlerden daha hızlı olduğu için buralar arasında da rüzgar oluşumu oldukça fazla olmaktadır.

Rüzgar enerjisinin tarihçesine bakıldığı zaman tarihteki ilk kullanımlarından biri, M.Ö 5500 yıllarda Akdeniz'de bulunan yelkenli gemilerin hareket ettirilmesi olmuştur. M.Ö 2500'lerde Hindistan civarında ilk değirmen örnekleri olduğuna dair bulgular günümüzde hala tartışılmaktadır. M.Ö 250 yılında İran'da tahılları öğütmek amacıyla rüzgar çarkları kullanılmıştır. İçinde 'yel değirmenleri' kavramı geçen kaynakların ilk örneklerine bakıldığı zaman ise tarih 12. Yüzyılı göstermektedir. Haçlı seferleri ile birlikte rüzgar enerjisi ile alakalı tüm bu bilgi ve birikimler Avrupa'ya yayılmıştır**.**

Rüzgâr enerjisininim kullanımının modern ilk örnekleri 1890'da Danimarka'da olmuştur. 1930'lara gelindiğinde ise Amerika'da rüzgâr enerjisine yönelim artmış ve 1940'lara kadar yoğun bir şekilde rüzgâr tribünü inşa etme çalışmaları gerçekleştirilmiştir. Avrupa'da da rüzgar enerjisine yapılan yatırımlar Amerika'dakine benzer bir şekilde gerçekleşmektedir.1980 sonrası ise gerek Amerika'da gerekse Avrupa'da rüzgar enerjisine çok büyük yatırımlar yapılmıştır (İlkılıç, 2016: 2) (Çakır vd., 2016: 1) (Sørensen, 1991: 8).

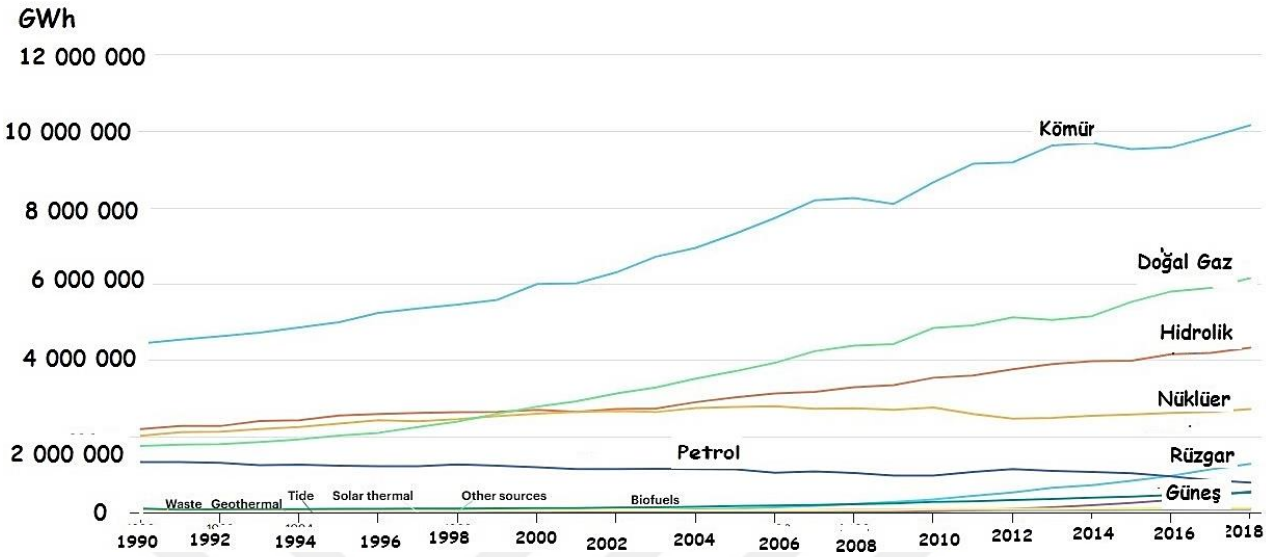
1.1.2. Dünya ve Türkiye'de rüzgâr enerjisinin durumu

Elektrik enerjisinin diğer enerji türlerine (ısı, ışık, kimyasal, mekanik, vb.) çok kolay dönüştürülebilir olması, enerji türü dönüşümlerinde ve enerji nakillerinde enerji kaybının küçük oluşu, istenilen büyüklükte parçalara kolay bölünerek satılabilir olması nedeniyle genel olarak enerji kaynakları, elektrik enerjis*i* türüne dönüştürülerek kullanıma sunulur. Bu uygulamayla enerjiler aynı ölçü birimiyle ifade edildiklerinden, değişik enerji kaynaklarından elde edilip de kullanıma sunulan enerjilerin büyüklüklerin karşılaştırmasını, kolaylaştırmaktadır.

Şekil 1.2 ve Şekil 1.3 grafikleri bu anlayışa göre oluşturulmuş grafikler olup Şekil 1.2'de: dünyada, 1990 – 2018 yılları arasında, enerji kaynaklarına göre üretilen elektrik enerji miktarları (IEA, 2020) gösterilirken; Şekil 1.3 ise: Türkiye'de, 2020 yılında, enerji kaynaklarına göre üretilen elektrik enerji miktarları (EMO, 2021) gösterilmektedir.

IEA (Uluslararası Enerji Topluluğu)'nın sitesinde yayınladığı Şekil 1.2 den, Dünya 'da, rüzgârdan üretilen elektrik enerjisi büyüklük durumu yönünden incelenirse, bu enerjinin geçmişten bu güne bir yükseliş eğilimi içinde olduğu; tüm enerji kaynakları içinde üretilen elektrik enerjisi büyüklük sıralamasında (5.); alternatif enerji kaynakları grubu dediğimiz

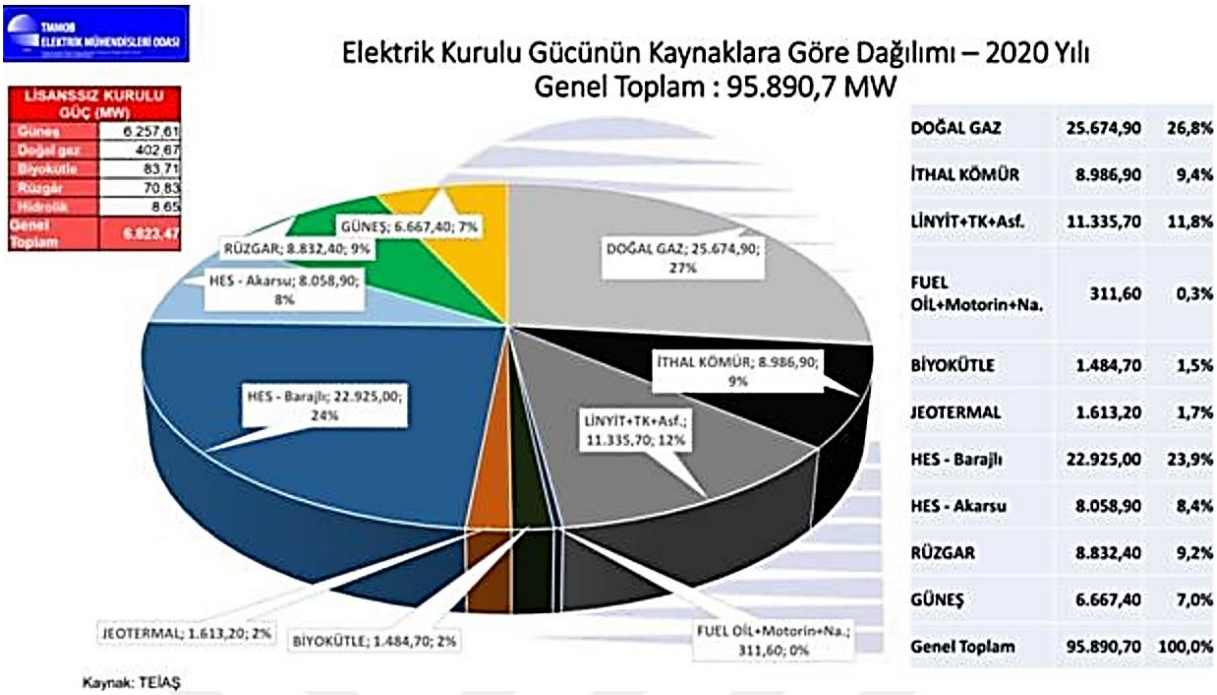
(hidrolik, güneş, rüzgâr, biokütle, dalga, gel-git, jeotermal, vb.) grubun içinde, hidrolikten sonra, (2.) sırada yer aldığı, görülmektedir.



Şekil 1.2. Dünyada, 1990–2018 yılları arasında, enerji kaynaklarına göre elektrik enerjisi üretimi

Kaynak: (IEA, 2020)

Şekil 1.3'den, Türkiye' de, rüzgârdan üretilen elektrik enerjisi büyüklük durumu yönünden incelendiğinde Türkiye'deki durumunun Dünya'daki duruma çok yakın olduğu, görülmektedir. Enerji kaynakları içinde rüzgâr enerjisinden üretilen elektrik enerjisinin büyüklüğü 'Dünya' ölçeğinde (5.) sıradayken 'Türkiye' ölçeğinde, nükleer enerji kullanımı hazırlık aşamasında olduğu için, (4.) sıradadır. Alternatif enerji kaynaklarının içerisinde, HES'lerden sonra, 'Dünya' da olduğu gibi, (2.) sıradadır.



Şekil 1.3. Türkiye' de 2020 yılında enerji kaynaklarına göre elektrik üretimi.

Kaynak: (EMO, 2021)

2. MATERYAL ve METOT

Bu bölümde Python dili, Python kütüphaneleri, kodların derlenmesi için kullanılan ortam olan Colab, yapay zeka ve ilişkili konular ve son olarak da literatürde yapılmış benzer çalışmalar verilmiştir.

2.1. Python

Python, Hollanda'da da bulunan Ulusal Araştırma Enstitü'sünde çalışan Guido van Rossum önderliğinde 1990'ların başında geliştirilmeye başlanmıştır. Bu yazılım dilinin isim babası olan Rossum'un bu ismi verirken ki esinlenme kaynağı ise İngiliz komedi grubu olan Monty Python'dır. Nesne tabanlı betik bir dil olan Python, o dönemler Hollanda menşeili bir başka dil olan ABC'nin yerine geçeceği düşünülmüştür. 2000 yılında 2.0 versiyonu ve 2008 yılında ise 3.0 olan versiyonu yayımlanmıştır. C#, Java gibi nesne tabanlı diğer dillerde de bulunan 'Garbage Collector' yapısı Python dilinde de bulunmaktadır.

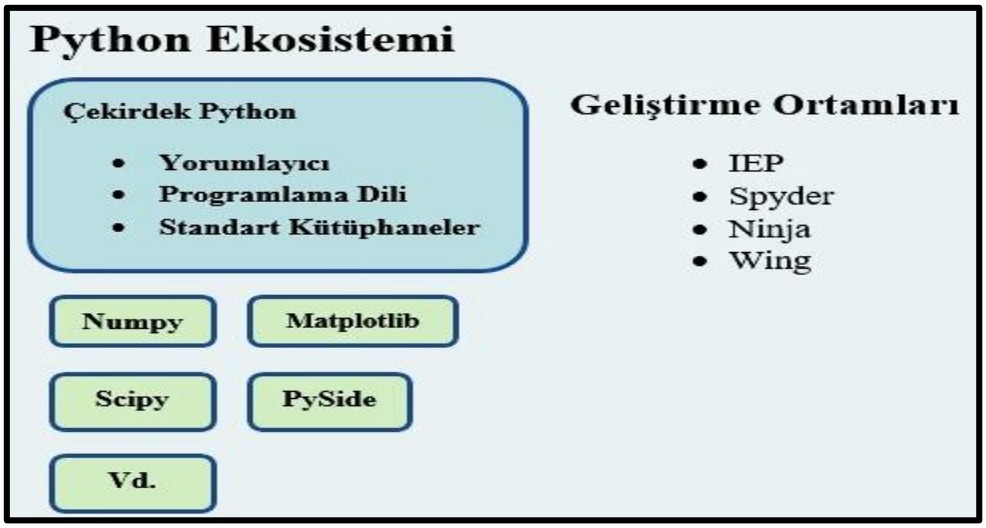
Python dilinin en önemli özellikleri:

- Öğrenmesi ve okuması kolay bir dildir.
- Farklı platformlarda da kullanılabilme özelliği bulunmaktadır.
- Çok farklı tool'lara sahip olması sebebiyle çeşitli farklı işlemler için kullanılabilmektedir.
- Açık kaynaklı olduğu için bu dilin geliştirilmesi işlemine tüm internet topluluğu katılabilir.

En büyük eksisi sayılabilecek durum ise çalıştırma hızının düşüklüğüdür.

2.2. Python Kütüphaneleri

200.000'den fazla pakete sahip olan Python otomasyon, veri analizi, makine öğrenmesi, web programlama vb., birçok farklı alanda kullanılmaktadır (Perl vd., 2017: 1) (Halterman, 2018: 669).



Şekil 2.1. Python Şeması

Kaynak: (Ozgur vd., 2017: 356)

Python hakkında bulunan genel bilgiler bu şekildedir. Bu projede kullanılan çeşitli Python kütüphaneleri ve kullanım amaçları aşağıda bahsedilmiştir.

2.2.1. Pandas

2008 yılında çıkartılan bu kütüphanenin Python dilindeki öncelikli amacı finansal veri analizi işlemlerini yapmaktır. Fakat Python'nın gittikçe popüler olması ile birlikte endüstri ve akademi alanında da Python dili çokça kullanılmaya başlanmıştır. Pandas kütüphanesi 'dataframe' ve 'series' olarak iki ana yapının üstüne kurulmuştur. Genellikle Numpy ile beraber kullanılmaktadır. Sql veri tabanı yapısı ile beraber bir ilişki kurabilmesi önemli avantajlarından bir tanesidir (Mckinney, 2011: 2) (Bloice & Holzinger, 2016: 450) .

Pandas kütüphanesinin önemli özelliklerinden bir tanesi de veri setinde bulunan parametrelerin kendi aralarında bir korelasyon bulunup bulunmadığıdır. Özellikle makine öğrenmesi işlemlerinde parametreler arasındaki korelasyonu bulabilmek önemlidir. Pandas ile parametreler arası korelasyon bulma işlemi 'corr()' fonksiyonu kullanılarak kolayca elde edilmektedir (Uğuz, 2019: 36).

2.2.2. Numpy

'Numerik Python' kelimelerinin baş harflerinden oluşturulan Numpy kütüphanesi sayısal işlemlerin yapılması amacıyla geliştirilmiştir. Benzer tiplerden oluşan 'ndarray' isimli bir matris bu kütüphanenin ana kısmını oluşturmaktadır. Bu şekilde normal bir diziye oranla daha hızlı şekilde işlemler yapılabilmektedir. Numpy'ın en önemli özelliklerinden biri

algoritma ve kitaplıklar arasında veri aktarımı yapabilme özelliğidir. Farklı dillerde yazılan kodlarla da Numpy dizisindeki veriler işlenebilir (Charan, 2020: 2) (Mckinney, 2017: 4).

2.2.3. Matplotlib

John D. Hunter tarafından oluşturulan bu kütüphanenin amacı verilerin görselleştirme işini yapmaktır. Veri noktaları çizmek, farklı grafikler oluşturmak ve bunları çoklu şekilde göstermek, histogram oluşturma vb., çoğu işlemi matplotlib kütüphanesi ile yapılabilmektedir (Charan, 2020: 2) (Mckinney, 2017:5).

2.2.4. Scikit-learn

2010 yılında geliştirilmeye başlanan kütüphane kümeleme, sınıflandırma, regresyon vb., türden işlemler yapılmasına olanak sağlamaktadır. Ayrıca yüklenen veri setini istenene oranlarda test ve eğitim kümelerine ayırabilir ve ayrılan bu veriler ile modeli eğiterek doğruluk oranını da ortaya çıkarabilir (Charan, 2020: 2) (Mckinney, 2017:7).

2.3. Colab

Derin öğrenme sayesinde günümüzde farklı alanlarda kullanılmak üzere çeşitli işlemler yapılabilmektedir. Arama motoru oluşturma, doğal dil işleme, e-ticaret önerileri sunmak vb., arttırılabilecek şekilde farklı alanlardan farklı örnekler sıralanabilir. Bu işlemlerin düzgün bir şekilde yapılabilmesi için kurulacak modelin iyi bir şekilde eğitilmesi gerekmektedir. İyi bir eğitim yapılabilmesinin en önemli unsurlarından biri de nicelik olarak çok sayıda eğitim verisine sahip olmaktır. Bu sayıca aşırı fazla olan verinin işlenmesi için de hızlı bir işlemciye ihtiyaç duyulmaktadır. GPU'lar, CPU'lara göre çok daha fazla hızlı işlem yapabilme yeteneklerine sahip olduklarından dolayı derin öğrenme işlemlerinde çokça kullanılmaktadır. GPU'lar ile verilerin hızlı bir şekilde işlenmesi pozitif bir durumken bu işlemin negatif sonuçları ise;

- Yüksek miktarlarda enerji harcamaları.
- Bakımı nedeniyle ortaya çıkan maliyet masrafları.
- Sahip donanımların pahalı olması.

Yukarıdaki nedenlerden dolayı insanlar derin öğrenme ve makine öğrenmesi gibi işlerini Google, Amazon, Intel gibi markaların oluşturdukları bulut teknolojisinden yararlanarak halletmektedirler. Google'ın diğer platformlardan ayrılan en büyük özelliği ücretsiz oluşudur. Tek gereken şart ise bir Google hesabına sahip olmaktır.

Google Colaboratory veya kısaca Colab, Jupyter Notebooks benzeri bir yapıya sahiptir. Python'ın 2.0 ve 3.0 olan iki sürümünü de desteklemektedir. Ayrıca sadece derin öğrenme benzeri işler değil bilimsel hesaplama gibi çok fazla işlem gerektiren işler de bu platformda yapılabilmektedir (Carneiro vd., 2018: 2).

2.4. Yapay Zeka

Yapay zeka kavramı "*insanın düşünme, ilişkilendirme, anlamlandırma yetenekleri ile makinelerin bellek gücü ve işlem hızı yeteneklerinin birleştirilmiş hali*" olarak tasvir edilebilir. Yapay zeka kavramının insandan alınan özelliği olan 'zeka' olgusunun literatürde kesin bir tanımı bulunmamaktadır. Çeşitli psikologların zeka ile ilgili farklı tanımları bulunmaktadır bazıları şunlardır:

- Jean Piaget, "*zekayı insanın çevresi ile uyumlu bir halde olması, eylemleri ve düşüncelerinin tutarlılığı vb., uyumlu davranış örneğidir*" diyerek tanımlar (Clark, 1994: 206).
- Aflred Binet ise "*insanın sahip olduğu dikkat, bellek, yargılama, akıl yürütme, sorgulama benzeri yetiler topluluğu*" olarak tanımlar.
- David Wechsler, "bireyin amaçlı bir şekilde hareket edebilme, mantıklı düşünebilme ve çevresine uyum gösterme yetilerinin tamamıdır" der.
- Pierre Oleron, "*bir amacın gerçekleşmesi için araçların duruma uygun kılınmasıdır*" şeklinde açıklar (Nabiyev, 2016: 26).

Akıl kavramının bu tanımın içinde bulunmama sebebi ise aklın stabil bir yapıda olmayışı ve aklı etkileyen parametrelerin önemli bir kısmının toplum, çevre vb., dış etmenlere bağlı olmasıdır. Zeka ise genetik tabanlı ve geliştirilmesi daha kısıtlı olmasına rağmen analiz, anlama, sorun çözme gibi özellikleri makine ve bilgisayarlara entegre edilebilecek bir yapı olduğundan dolayı akıl yerine zeka kavramı kullanılmıştır (Yılmaz, 2020: 1).

Yapay zekanın kurucusu İngiliz matematikçi Alan Mathison Turing'dir. 'Mind' isimli dergide yayınladığı makalede "*can machines think (makineler düşünebilir mi)?*" sorusunu ortaya atarak makinelerin de insanlar gibi düşünebilmesi ihtimalini gündeme getirmiştir. "Turing Testi" makinelerin düşünce konusunda insanlara ne kadar yaklaştığının test edilmesi üzerine tasarlanan bir yapıdır.

Turing testi, ikisi insan biri makine olmak üzere üç kişiliktir. 'Savcı' olarak adlandırılan gerçek kişi elindeki soruları haberleşme kanalı üzerinden makine ve insana sorar. Gelen cevaplar üzerine 'savcı' bu iki kişiden hangisinin insan ve makine olduğunun ayrımını yapar.

Günümüzde hala Turning Testi makinelerin yapay zekasını ölçmek için kullanılan belirleyici bir testtir.

Yapay zeka konusu ile ilgili Turing testine benzer olarak T. Winogard'ın 'SHRDLU' ve J. Wezenbaum'un 'ELIZA' testleri yapılmıştır. Bu testlerin hepsinin ana fikri makinelerin eğitilerek insan gibi düşünebileceğini temel almasıdır. Bu mantığa karşı olan John Searle'nın ortaya attığı fikir ise "Çin Odası" testiydi. Bu testin ana fikri, Çince bilmeyen bir kişi kapalı odada bulunmaktadır. Yanında ise Çince yazılan tabelalar ile bunları açıklayan bir kılavuz verilmektedir. Kendisine Çince soru tabelaları gelmekte olan bu kişi kural kitabına bakarak ilgili cevap tabelasını karşı tarafa geri vermektedir. Dışarıdan biri bu olayı gözlendiğinde içeride bulunana kişinin Çince biliyor olduğu düşünebilir fakat bu soruların cevapları, anlayarak veya öğrenilerek değil kural kitabından bakılarak verilmiştir. Bu görüş öğrenmenin sadece cevap verme ile alakasının olmadığını savunur (Güç & Ceyhan, 2016: 36) (Nabiyev, 2016: 52).

Tarihsel olarak yapay zekâ'nın gelişimine bakıldığında temelinin 1884 yılında atıldığı ifade edilebilir. Charles Babbage isimli kişi insan gibi zeki davranabilecek bir makine tasarlama çalışmaları gerçekleştirmişti. Bundan sonra yapay zeka çalışmaları 1950'lerin ortalarına kadar oldukça yavaş bir şekilde devam etmiştir.

Yapay zeka kavramının ilk defa sözel olarak tanımlanması 1956 yılında Dartmouth Koleji'nde bir konferansta olmuştur. Bu dönemde geliştirilen yapay zeka uygulamaları; mantıksal teorem uygulaması ve satranç oyunu programıdır. Bu programların yanı sıra asıl yapay zeka kavramına güveni arttıran olay ise geometrik şekilleri ayırt edebilen bir programın ortaya çıkmasıdır. Bu tarihlerin dışında yapay zeka alanında yapılan bazı işler şunlardır:

- 1923 yılında Çek oyun yazarı Karel Capek'in yazdığı tiyatro oyununda ilk kez 'robot' ismi geçmiştir.
- 1936 yılında 64K hafızaya sahip 'Z1' isimli programlanabilir bir bilgisayar Konrad Zuse tarafından geliştirildi.
- 1946 yılında 'ENIAC' isimli ilk bilgisayar kullanılmaya başlanmıştır.
- 1956 yılında ilk yapay zeka sistemi örneği sayılan "Logic Theorist" isimli program Neweell, Shaw, Simon tarafından geliştirilmiştir.
- 1957 yılında John McCarth'nin yapay zeka için geliştirdiği bir yazılım dil olan LISP (List Processing Language) ortaya çıkmıştır.
- 1962 yılında Unimaton, Endüstri alanında kullanılmak üzere robot üreten ilk firma olmuştur.

- 1965 yılında ELIZA programı yazılmıştır.
- 1966 yılında Stanford Üniversitesi tarafından ilk haraketli robot ‘Shakey’ üretilmiştir.
- 1974 yılında Cerf ve Kahn tarafından ‘İnternet’ terimi dile getirilmiştir.
- 1978 yılında yapay zeka alanında yaptığı çalışmalardan dolayı Simon Nobel ödülünün sahibi olmuştur.
- 1981 yılında kişisel bilgisayarların (PC) üretimine IBM tarafından başlanılmıştır.
- 1993 yılında ‘Cog’ isimli insansı görünümlü robot MIT tarafından tasarlanmıştır.
- 1997 yılında o dönemin satranç şampiyonu olan Garri Kasparov’u “Deep Blue” isimli bilgisayar yenmiştir.
- 1998 yılında ‘Furby’ isimli ilk yapay zeka oyuncağı piyasaya sürülmüştür.
- 2000 yılında ‘Kismet’ isimli jest ve mimikleri olan robot üretilmiştir.
- 2005 yılında beceri ve şekil olarak insana en çok benzeyen robot olan ‘Asimo’ üretilmiştir.
- 2010 yılında Asimov’un beyin gücüyle hareket etmesi sağlanmıştır

  (Pirim, 2006: 82) (Yılmaz, 2020: 15).

Yapay Zekâ; Makine Öğrenmesi, Derin öğrenme gibi konuların daha kapsayıcı bir halidir. Yani bu diğer konular yapay zeka’nın alt kümeleri olarak düşünülebilir (Karakuş & Kaya, 2019: 34).

Şekil 2.2. Yapay Zeka ve diğer konuların ilişkisinin görseli.

Kaynak: (Karakuş & Kaya, 2019: 34)

2.4.1. Yapay sinir ağları

Makineler ile insanlar arasında büyük bir fark bulunmaktadır. Makineler matematiksel işlemler ve hafıza olarak insanlardan oldukça üstünken insanlar ise özellikle sözel ve görsel materyaller üzerinde yapılan çeşitli işlemleri makinelerden daha iyi yapabilmektedirler. Makinelerde olan bu eksikliği tamamlayabilmek için bilim adamları, insanın biyolojik yapısının benzerinin iz düşümünü makineler üstünde yapmak istemişlerdir. Bunun yapılabilmesi için gereken çözümün beyin ve sinir sisteminde olduğunu düşünmüşlerdir.

Yapay sinir ağları, modellenmesi açısından insan beyni ve sinir sistemini örnek olarak almıştır. Bu örnek alma birebir bir şekilde değildir. Çünkü insanın beyin yapısı oldukça karmaşık bir sistemdir. Günümüzde bile beynin çalışma işleyişi tam olarak açıklanamamaktadır. McCulloch ve Pitts gerçek nöron hücresi yapısının bir modellemesini hayata geçirerek ilk yapay sinir ağı modelini tasarlamışlardır.

Yapay sinir ağlarını kabaca tanımlayacak olursak; insanda bulunan sinir sisteminin bir benzerinin makineye uygulanmasıyla makinelerin insanlar gibi öğrenme, analiz, sorun çözme gibi işlemleri yapabilecek hale getirmesini sağlayan yapıya denmektedir. Yapay sinir ağı örneklerden elde edilen ağırlık, bias gibi elemanlardan ve duruma göre 2 veya 3 katmandan oluşan bir ağ yapısıdır.

Kullanıldığı alanlara örnek olarak; sınıflandırma, örüntü tanıma, sinyal filtrelenmesi, veri sıkıştırma ve optimizasyon gibi çokça alanda kullanılıp başarılı sonuçlar elde edilmiştir. Hayatın içinden örneklere bakılacak olunduğunda; parmak izi tanıma, malzeme analizi, iş çizelgesi, rota belirleme, kalite kontrol gibi gündelik yaşamımızda sıkça karşılaştığımız işlerde de yapay sinir ağları etkin olarak kullanılmaktadır (Öztemel, 2012: 29) (Nabiyev, 2016: 580).

Yapay sinir ağının tarihçesine bakıldığı zaman:

- 1943 yılında McCulloch ve Pitts tarafından ilk yapay sinir ağ modeli tasarlanmıştır.
- 1949 yılında Hebb yayınladığı kitap ile öğrenme ile alakalı temel teoriyi açıkladı. Bu kitapta Hebb sinir ağı bağlantısı ile öğrenme arasında bir bağ olduğundan bahsediyordu. Bu tanımlamaya “Hebb Kuralı” denmektedir.
- 1957 yılında F. Rosenblatt ilk perceptron yapısını modellemiştir.
- 1959 yılında Widrow ve Hoff adaline ve madeline ağlarını modellediler.

- 1969 yılında Minsky ve Papert XOR problemini çözen bir yapay sinir ağı modellemişlerdir.
- 1982 yılında Hopfield çeşitli çalışmalarda bulunmuştur.
- 1984 yılında Kohonen "gözetimsiz öğrenme" kavramını geliştirmiştir.
- 1987 yılında IEEE (Elektrik Elektronik Mühendisliği Enstitüsü) sinir ağlarını konu alan geniş kapsamlı uluslararası bir konferans düzenlendi (Elmas, 2018: 49).

Her yapı gibi yapay zekanın da artı ve eksi yanları bulunmaktadır. Artı yanları:

- Hücrelerden oluşan yapay sinir ağlarında herhangi bir hücre düzgün işlem yapamaz hale gelse bile sistemde sorunsuz çalışmasına devam eder.
- Yapay sinir ağları eğitim verileri sayesinde sadece ana değil yan konular hakkında da bilgi üretebilir.
- Bilgi işleme yeteneği olarak klasik programlama yapısından farklı bir yapıya sahip olduğu için klasik programlamanın eksik yönlerini barındırmaz.
- Klasik programlamada bilgiler dosya ve veri tabanında tutulmaktadır. Yapay sinir ağlarında ise veri tüm sisteme yayılmıştır. Bu yapısı sayesinde ağ, olası bir bilgi kaybı durumundan çok fazla etkilenmez.
- Yapay sinir ağlarına eksik bilgi içeren örüntüler verilmesi sistemde bir soruna yol açmaz. Ağ bu eksik örüntüleri tamamlayabilir.

Yapay sinir ağlarının eksi yönleri ise şunlardır:

- Paralel işlem yapabilme yeteneği olan yapay sinir ağları gücünü güçlü donanımlardan almaktadır. Ancak iyi bir işlem gücüne sahip olan donanım ile optimum performans ortaya çıkabilir.
- Katman sayısı, nöron sayısı, aktivasyon fonksiyonu vb., parametreler seçilirken herhangi bir kural yoktur. Bu sebepten dolayı iş daha çok tecrübeye dayanmaktadır.
- Yapay sinir ağları sözel veri işleyemez bundan dolayı verilerin sayısal hale getirilmesi gerekmektedir. Tüm verilerin sayısal hale getirilmesi işlemi ağın performansını etkiler.

- Eğitimin bitirilmesi için kesin bir kural yoktur. Bitirme işlemi için belli bir hata oranının altına düşmesi beklenmekte veya tekrar sayısı verilmektedir.
- Yapay sinir ağları adeta bir kara kutu gibidirler. Yani ağın çalışma prensibi, işleyişi dışarıdan tarif edilemez.
- Çoğu yapay sinir ağları kararlılık analizi yapamazlar (Yılmaz, 2020: 67).

2.4.2. Makine öğrenmesi

Makine öğrenmesinin başlangıcı olarak 1949 yılında Donal Hebb'in yazdığı "*The organization of behavior*" isimli kitabı sayılabilir. Yapay sinir ağının 'öğrenme' işlemini içinde olduğu sistemdeki ağırlıkların eğitim ile güncellenmesinden anlaşılmaktadır. Hebb yayınladığı kitapta iki sinir hücresi arasındaki ağırlığın, sinirlerin aynı anda etkinleşmesiyle olacağını ileri sürer. Hebb'in ileri sürdüğü bu kural uzun zaman kullanılmış ve çeşitli öğrenme metodolojileri ortaya çıkmıştır. Bu bilgiye dayanarak makine öğrenmesinin tanımını yapacak olursak makine öğrenmesi, makinelerin insanlar gibi düşünebilen hale getiren yapay sinir ağları yapısının makinelere nasıl öğretileceğini açıklayan bir metodolojidir (Elmas, 2018: 97).

Makine öğrenmesini eldeki verilerden yola çıkarak ileriyi tahmin etme veya çeşitli nedenlerle belirsiz olan bir konu için karar verme gayesiyle kullanılan yöntemler bütünü olarak adlandırabiliriz. Yapay zekanın 1980'lerde gelişmesiyle birlikte makine öğrenmesi kavramı da bu zamanlarda ortaya çıkmıştır. Kullanıldığı alanlar yapay sinir ağlarına benzer şekilde kümeleme, tahmin, sınıflandırmadır. 3 tip öğrenme türü bulunur:

- Gözetimli Öğrenme
- Gözetimsiz Öğrenme
- Takviyeli Öğrenme.

Gözetimli öğrenme: Sisteme eğitim verileri sokulur ve tahmin değerleri elde edilir. Elde edilen bu tahmin değerleri, gerçek değerler ile karşılaştırılarak modelin doğru çalışma oranı bulunur. Eğitim kısmındaki önemli kriterlerden biri eğitim veri setinin çeşitliliğinin geniş bir skalada olmasıdır. Ayrıca test ve eğitim veri setlerinin de ayrı bir şekilde değerlendirmeye alınması gereklidir.

Tahmin ve gerçek değerlerin farkı hata değerini göstermektedir. Hata değeri ne kadar küçük olursa sistem o kadar efektif bir şekilde çalışmaktadır. Eğitimle amaçlanan bu hata değerini minimum seviyeye indirmektir. Modeldeki hata güncelleme tipine göre bu öğrenme çeşidi ikiye ayrılmaktadır. Modeldeki toplam hataya göre ağırlıklar güncellenirse buna "küme

tipi öğrenme" denir. Modeldeki ilgili giriş-çıkış ile alakalı hata güncellenirse o zaman buna "örüntü tipi öğrenme" denmektedir. Lineer ve lojistik regresyon, karar ağacı gibi algoritmalar gözetimli öğrenmenin içerisinde bulunmaktadır. Bu algoritmalar genellikle daha çok tahmin işlemleri için kullanılır (Ayturan, 2019: 16) (Elmas, 2018: 98).

Gözetimsiz öğrenme: Bu öğrenme tipinde ise modelin tahmin işlemini gerçekleştirdikten sonra karşılaştırılacağı bir gerçek değerler kümesi yoktur. Öğrenme işleminin dışarıdan herhangi bir etki almadan kendi içinde gerçekleştirir. Hebbian, Grossberg, Kohonen gibi gözetimsiz öğrenmenin çeşitli örnekleri de bulunmaktadır. Kohonen öğrenme tipi için "yarışmacı öğrenme" denilebilir. Modelde bulunan sinir ağları en iyi sonucu verebilmek için birbirleriyle yarışır ve en iyi sonucu veren komşularının ağırlıklarını güncellemesine izin verir.

Gözetimsiz öğrenmenin alt iki dalı bulunmaktadır bunlar; "çevrim içi" ve "çevrim dışı"öğrenmedir. Bunlar gözetimli öğrenmenin çeşitleri olan küme tipi ve örüntü tipine benzer yapılardır. Güncelleme işlemi eğer eğitimden sonra yapılırsa buna çevrim dışı, giriş – çıkış işleminden sonra hata güncellemesi yapılırsa çevrim içi hata denmektedir (Elmas, 2018: 121).

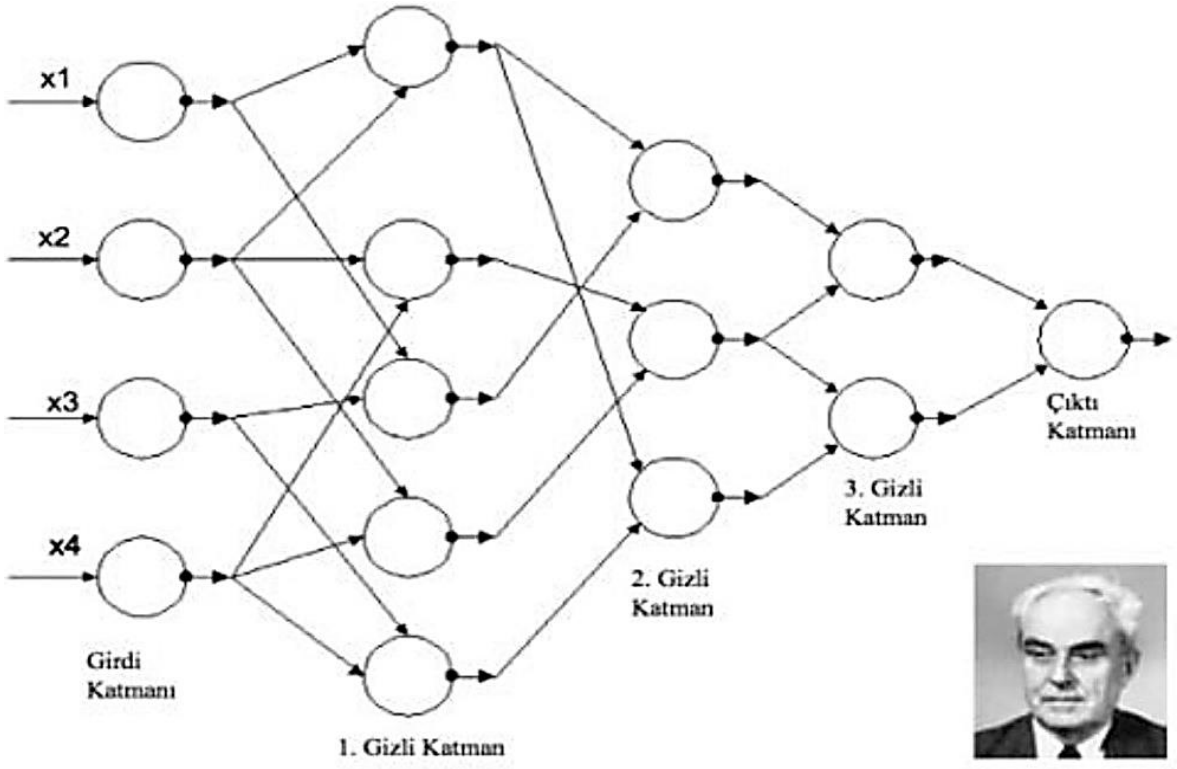
Takviyeli öğrenme: Bu öğrenme tipi hem gözetimli hem de gözetimsiz öğrenmeye ait yapılar barındırmaktadır. Gözetimsiz öğrenmedeki gibi sadece girişler verilip sonuçlar verilmemektedir. Gözetimli öğrenme de olduğu gibi bir gözetmen vardır fakat bu gözetmen doğruluk karşılaştırması yapmaz sadece sistemin çıktısının doğru ya da yanlış olduğunu belirtir.

Takviyeli öğrenme tiplerinden biri de LVQ (Doğrusal Vektör Nicelemesi)'dir. Genellikle sınıflandırma amaçlı kullanılan bu yapının çalışma şekli çoklu boyutlu matrisin çeşitli vektör tiplerine ayrılması ile başlar. Bu ayrılan vektörlerin hangisinin başlangıç olacağı sistemin öğrenmesi ile alakalıdır. Ağ çıktısı 1 ve 0 şeklinde olmaktadır. Kazanan her şeyi alır mantalitesinde olan bu yapıda yalnızca 1 değer çıktısı veren vektörün değerleri değiştirilir.

2.5. Derin Öğrenme

Derin öğrenme, giriş kısmında bahsedildiği gibi makine öğrenmesinin içinde bulunan bir konudur. Denetimli ve denetimsiz olmak üzeri iki çeşit öğrenme algoritmasına sahiptir. Derin öğrenmenin asıl ana noktası, makine öğrenmesindeki gibi birincil özelliklerin belirlenmesine gerek duymaz. Sistem otomatikman bunu kendisi belirlemektedir. Duygu analizi, doğal dil işleme vb., çokça alanda kullanılmaktadır (Sezer vd., 2019: 5) (Şeker vd., 2017: 48).

1965 yılında bu konuyla alakalı ilk algoritmalardan sayılan Ivakhenko ve Lapa tarafından geliştirilen "*denetimli derin beslemeli çok katmanlı perceptronlar*" yayınlanmıştır. Katmandaki en iyi özellikler matematiksel işlemlerle belirlenerek bir sonraki katmana aktarılmaktadır. Ağın daha iyi eğitilmesi için geriye yayılım işlemi kullanılmıştır.



Şekil 2.3. İlk derin ağ mimarisi (Ivankhnenko A.)

Kaynak: (Şeker vd., 2017: 48)

1979 yılında 'Neokognitron' isimli ilk derin öğrenme mimarisi ortaya çıktı. Denetimsiz öğrenme metodunu kullanan bu yapı Fukushima tarafından omurgalı canlıların görsel sinir sistemleri baz alınarak modellenmiştir. Bu çalışma evrişimli sinir ağlarını (CNN) tanıtmıştır ve günümüz modern denetimli ileri beslemeli sinir ağlarının (D-FFNN) temelini oluşturmuştur.

Çok katmanlı derin mimarilerde hataların geri yayılımının öğrenmeyi zorlaştırdığı bilinmektedir. Önceki yıllarda bu tip algoritmalar yayınlansa da ilk başarılı örneği 1989 yılına gelindiğinde Yann LeCun ve arkadaşları tarafından posta kutuları üzerindeki yazılarda gerçekleştirilmiştir. 3 gün sürmüş olan eğitim sonunda algoritma başarılı olmuş fakat pratikte uygun görülmemiştir. Daha sonraları (1990 sonları ve 2000 başlarında) el ile yazılmış çeklerin kontrolü için Amerika'da kullanılmıştır.

1995 yılına gelindiğinde Brendan Frey, Peter Dayan ve Geoffrey Hinton “uyanık-uyku (wake-sleep)” ismini verdikleri 6’sı tamamen bağlı yüzlerce gizli katman içermekte bir ağ oluşturdular.

1997 yılında Hochrieter ve Schmidhuber 1000’den fazla katmanı olan tekrarlayan sinir ağları (RNN) ve uzun kısa vadeli bellek (long-short term memory) gibi olgular ortaya çıkmıştır. Yapay sinir ağlarının bu dönemde çok tercih edilmemesinin sebebi hesaplama maliyetidir.

1990 ve 2000’ler arası genellikle destek vektör makineleri (SVM) kullanılmıştır.

Teknolojinin gelişmesi ile beraber bilgisayarların işlem hızı da artmış ve matematiksel işlemlerin hızlı bir şekilde çözülmesi için grafik işlemci birimlerinin (GPU) kullanılmasının yararlı olduğu fark edilmiştir. Böylece eskiye nazaran işlem hızı tam 1000 kat artmıştır.

2000’lerin başında yapay sinir ağları içinde “Derin Öğrenme” kavramını Igor Aizenberg ve arkadaşları tanıtmıştır.

2006 yılında yayınlanan bir makalede Geoffrey Hilton, çok katmanlı bir ağın nasıl her katmanının efektif bir şekilde eğitilmesi gerektiğinden bahsetmiştir. Böylece derin öğrenme kavramı gittikçe popüler olmuştur.

2009 yılına gelindiğinde 167 ülkeden 50 bin çevrimiçi çalışan, 22 bin kategori ve 15 milyon görüntüden oluşan kümeyi ImageNet ile görüntülerin kümelenmesinde kullanmıştır.

2010 yılında yapılan resimlerin görsel tanımlama yarışmasında (ILSVRC) büyük veriler ile çalışma konusu tekrar ortaya atılmıştır.

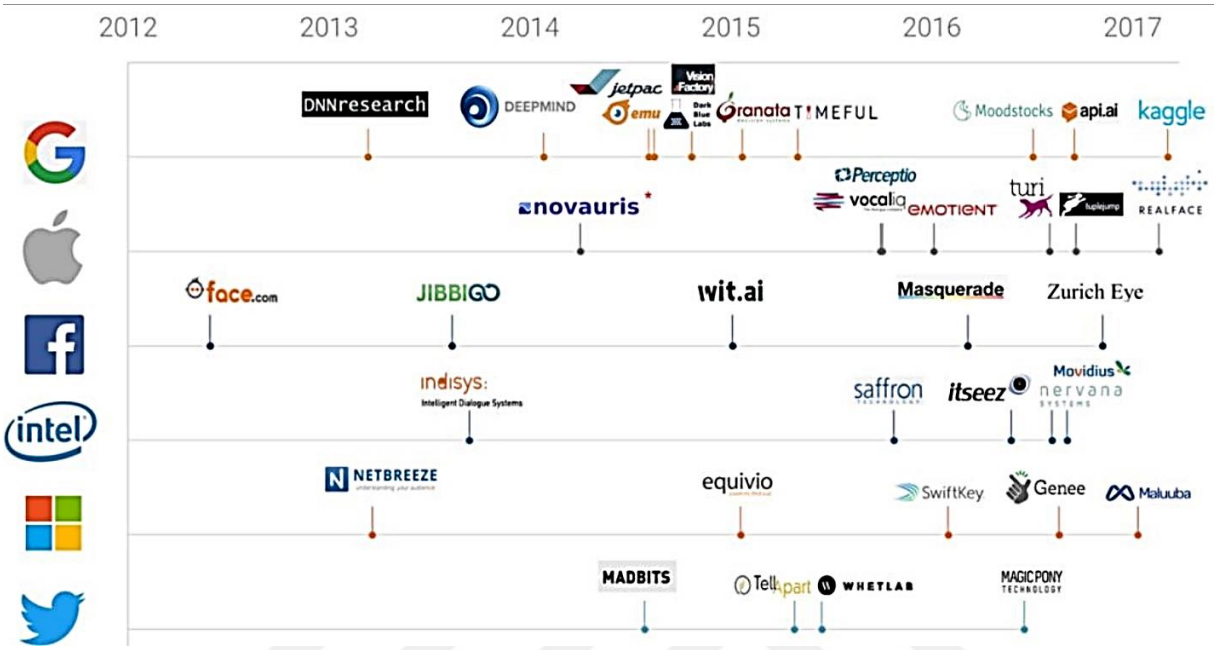
2012 yılında Krizhevsky, Sutskever ve Hilton, GPU kullanarak yaptığı çalışmalarda sistemin ezberleme işleme yapmaması için “Dropout” isimli normalleştirme işlemini yapmıştır. İlk 5 katmanın hata oranı %16.4‘e kadar düşürülmüştür.

2014 yılında “üretken çekişmeli ağ“ ortaya çıkmıştır. Buradaki amaç, iki ağın birbiri ile rekabet etmesiyle birlikte ağların öğrenmesini sağlamaktır. Ayrıca GoogLeNet modeli ile daha derin katmanlar oluşturulmuş ve başarılı sonuçlar elde edilmiştir.

2016 yılı Microsoft insanlarla konuşarak öğrenen ‘TAY’ isimli bir sohbet robotu geliştirdi. Fakat kendisine yazılan ırkçı yazıları öğrenip bunları insanlara karşı kullanınca iptal edildi.

2017 yılında Google kompleks bir oyun olan ‘Go’ isimli oyun için geliştirdiği AlphaGo yazılımı en iyi Go oyuncusunu yendi.

2019'da Yoshua Bengio, Geoffrey Hinton ve Yann LeCun, derin sinir ağlarını bilgi işlemin kritik bir bileşeni haline getiren kavramsal ve mühendislik buluşları için Turing Ödülü'ne layık görüldü (Şeker vd., 2017: 49) (Emmert - Streib vd., 2020: 3) (Kızrak & Bolat, 2018: 264).



Şekil 2.4. Şirketlerin yıllara göre Derin Öğrenme ile yaptıkları işler.

Kaynak: (Şeker vd., 2017: 49).

2.5.1. Derin öğrenmenin kullanıldığı çalışmalar

Derin öğrenmenin genellikle kullanıldığı alanlar; görüntü – ses analizi, robotik, otonom araçlar, tıbbi alanlar (gen analizi, kanser teşhisi vb.) ve sanal gerçekliktir. Siyah beyaz renklerin renklendirilmesinde de derin öğrenme kullanılmaktadır. Sessiz olan videolarda ise insanların dudak hareketlerini okuyarak bu videoların seslendirilmesi yapılmıştır. Bu dudak okuma tahminlerinde insanların başarası %52'lerde iken derin öğrenmenin başarısı %93'lere varmaktadır. Bu gibi tahmin işlerini aynı zamanda çizim konusunda (harita taslağından gerçek haritanın elde edilmesi) ve ayrıca insan hareketlerinin tahmininde kullanılmaktadır. Facebook ve Google firmaları etiketleme, obje tanınması gibi işlemlerini derin öğrenme ile yapmaktadır. İşler biraz daha ileriye götürüldüğünde ise derin öğrenme ile oyun oynayan makineler yapılmaktadır (İnik & Ülker, 2017: 88).

Yukarıda bahsedildiği gibi derin öğrenmenin yararlı amaçlar için kullanıldığı çokça alan bulunmaktadır. Fakat bazı kişiler derin öğrenmeyi zararlı amaçlar için de kullanabilmektedir. "Deep Fake" ile video ve ses kayıtlarında manipülasyon yapılabilir ve bu şekilde insanlar,

söylemedikleri şeyleri söylemiş gibi yaptırılarak olay yaratılabilir. Yüz değiştirme, ifade değiştirme, yüz yaratma gibi işlemler de yapılabilmektedir (Berk, 2020: 1512).

2.5.2. Derin öğrenme kütüphaneleri

Java, C#, C++, Python gibi daha çok nesne tabanlı olan programlama dillerinde derin öğrenme kütüphaneleri vardır. Çok katmanlı sinir ağları (MLP), tekrarlanabilen sinir ağları (RNN) gibi birçok modeli destekler. Bu diller arasında derin öğrenme açısından en uygun olanı bünyesinde çok sayıda kütüphane barındırmasından dolayı Python kabul edebilir. Keras ve TensorFlow en çok bilinen ve kullanılan kütüphanelerin başında gelmektedir.

Keras: Google tarafından Python dili ile geliştirilmiş bir kütüphanedir. Tensorflow ve Thenano gibi kütüphanelerin üzerinde de çalışabilmektedir.

TensorFlow: Keras gibi Google tarafından Python dili ile geliştirilmiştir. Ana kullanım alanları görüntü işleme ve sayısal hesaplamalardır.

Theano: MILA Lab tarafından Python dili ile geliştirilen bir kütüphanedir. Genellikle matematiksel konularda kullanılmaktadır.

Caffe: BVLC tarafından Python dili ile geliştirilirmiştir. Matlab'ta da kullanılabilmektedir. En önemli avantajı hızlı ve modüler olmasıdır. Kolayca modelleme ve optimizasyon işlemleri de yapabilmektedir.

Mxnet: Amazon tarafından Python ile geliştirilmiştir. En önemli avantajları çok dilli bir yapıya sahip olmasından dolayı farklı dillerde (R, Python, Scala, Julia) de işlemler yapılabilir. Ayrıca dağıtılmış bilgi işlemini destekler (Gündüz & Cedimoğlu, 2019: 12) (Toğaçar & Ergen, 2019: 111).

Ctnk: Mircosoft tarafından geliştirilmiş olan bir derin öğrenme yapısıdır.

PyTorch: Numpy kütüphanesi kullanılarak matematiksel işlemlerin çözümü için kullanılan Python temelli derin öğrenme kütüphanesidir. GPU'ya sahip olması ve tensor hesaplama yapabilmesi en önemli iki özelliğidir.

Chainer: Yapay sinir ağlarının eğitilmesi ve oluşturulması üzerine çalışan Python tabanlı bir derin öğrenme yapısıdır.

PaddlePaddle: CNN, RNN gibi modellerin kolayca yapılandırmasını sağlayan derin öğrenme yapısıdır.

2.5.3. Derin öğrenme modelleri

Alex Net: 2012 yılında ImageNet yarışmasındaki birincilikle adını duyuran bu model, 25 katmandan oluşmaktadır. GPU'lar tarafından paralel olarak çalışması için Nvidia tarafından geliştirilmiştir.

Le Net: 1998'de yayınlanan ilk başarılı modellerden biridir. Yazı karakteri analizi için kullanılmıştır.

ZF Net: 2013 yılında ImageNet yarışmasında birinci olan bu model Alex Net modelini baz alınarak tasarlanmıştır ve ondan daha iyi sonuçlar vermiştir. 7 katmandan oluşan bu model nesne tanıma konusundaki hata oranını %4.2 azaltarak genel hata oranını %11.2'ye çekmiştir.

Google Net: 2012 ve 2013 yılında ImageNet yarışmasına katılan model 2014'te birinci olmuştur.22 Katmandan oluşan bu model veri kümeleme konusunda başarılıdır.

Microsoft Rest Net: 152 katmandan oluşan bu model ImageNet yarışmasını 2015 yılında birinci tamamlamıştır.

ResNeXT: "Bölme, dönüştürme, birleştirme" mantığıyla resim sınıflandırması yapan bu model Facebook tarafından geliştirilmiştir (Gündüz & Cedimoğlu, 2019: 13) (Toğaçar & Ergen, 2019: 112) (Shrestha vd, 2019: 5).

2.5.4. Derin öğrenme mimarileri

Çok çeşitli derin öğrenme mimarileri mevcuttur. Bu mimarilerin birden fazla olması sebebi ise kullanım alanlarında gösterdikleri performanstır. Bazı mimariler görüntü işleme uygulamalarında etkili olurken bazıları ise tahmin konusunda daha iyi sonuçlar vermektedir. Bu mimariler şu şekildedir:

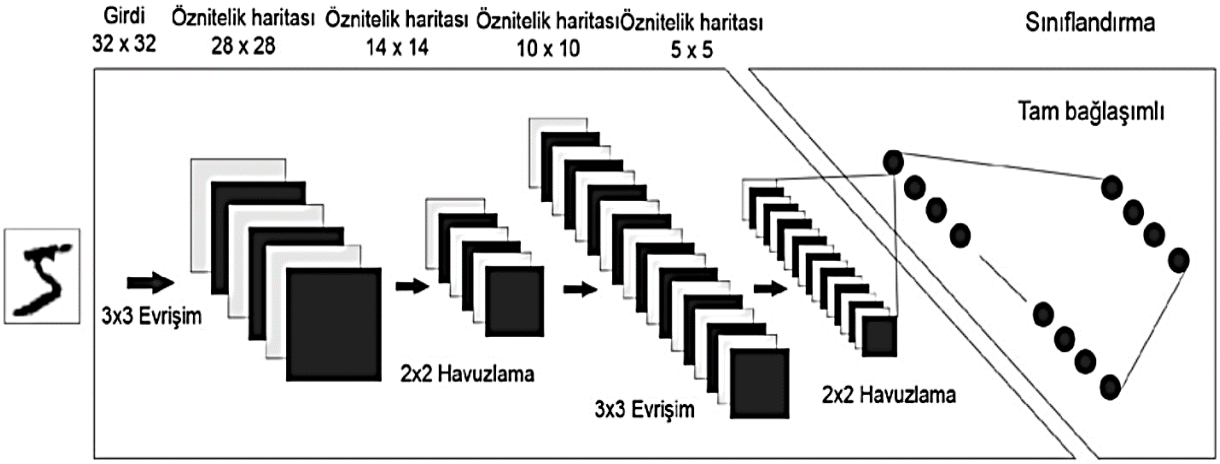
Evrişimli sinir ağları (Convolutional Neural Networks)

Evrişim, kelime anlamı olarak katlanma, sarmalama demektir. Evrişimin mühendislik, matematik, fizik gibi alanlarda ise kompleks yapıdaki işlemlerin sadeleştirmesi amacıyla kullanılmaktadır.

$$f(t) * g(t) \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} f(\tau) g(t-\tau) d\tau \qquad (2.1)$$

CNN ağlarında ki evrişimin kullanım amacı ise görüntünün istenilen forma sokulması için filtrelenmesidir. MLP (Çok Katmanlı Perceptron), bu derin öğrenme mimarisinin temel yapısıdır. Doğadan baz alınan model insan ve hayvanlardaki görme mekanizmasıdır. Bu mimarinin kullanıldığı alanlara örnek olarak; NLP (Doğal Dil İşleme), biyomedikal,

sınıflandırma, görüntü ve ses işleme vb., verilebilir. Ayrıca bu mimarilerin daha efektif bir şekilde kullanılması işlemci türü olarak CPU yerine GPU kullanılmasıyla olmuştur. GPU kullanıldığı zaman CPU'ya oranla hem daha doğru sonuçlar ortaya çıkmış ve aynı zamanda öğrenme hızında 10-60 kat bir artış meydana gelmiştir. Ayrıca bu mimaride doğru sonuçların alınabilmesi için geriye yayılım algoritması da eklenerek kullanılmış ve sonuçlar daha iyi bir şekle sokulmuştur (Tüfekçi & Karpat, 2019: 20).



Şekil 2.5. Görüntü işlemede kullanılan CNN yapısı.

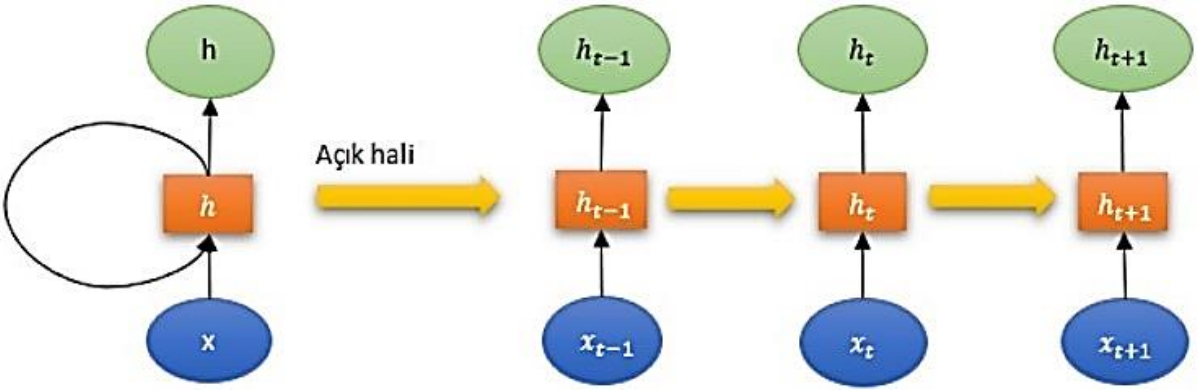
Kaynak: (Kurt & Efe, 2018: 3)

Tekrarlayan Sinir Ağı (Recurrent Neural Network)

Bu modelin kaynağı olarak Elman Ağı gösterilebilir. Genel kullanım alanı; zaman serileri ve ardışık verilerdir. Karakter tanıma konusunda çok iyi olan bu model el yazısı ve konuşma tanıma gibi işlemlerde de kullanılır.

Diğer modellerin işleyişinde çıkışlara bağlı olan girdiler birbirinden bağımsız şekildedir, buradaki yapıda ise bir bağımsızlık yoktur önceki katmanlardan alınan geri bildirimler sonraki katmanı etkilemektedirler. Geriye yayılım ve Jordan-Elman ağı yapıları RNN'in içinde bulunmaktadır. "*Geriye yayılım*" yapısı ortaya çıktığı zaman ilk kullanılan ağlardan biri RNN olmuştur.

RNN'in iki farklı çeşidi bulunmaktadır. Bunlar; derin RNN ve iki yönlü RNN'dir. Derin RNN'ler daha uzun zaman serilerinde kullanılmaktadır (Lecun vd., 2015: 436) (Doğan & Türkoğlu, 2019: 414).

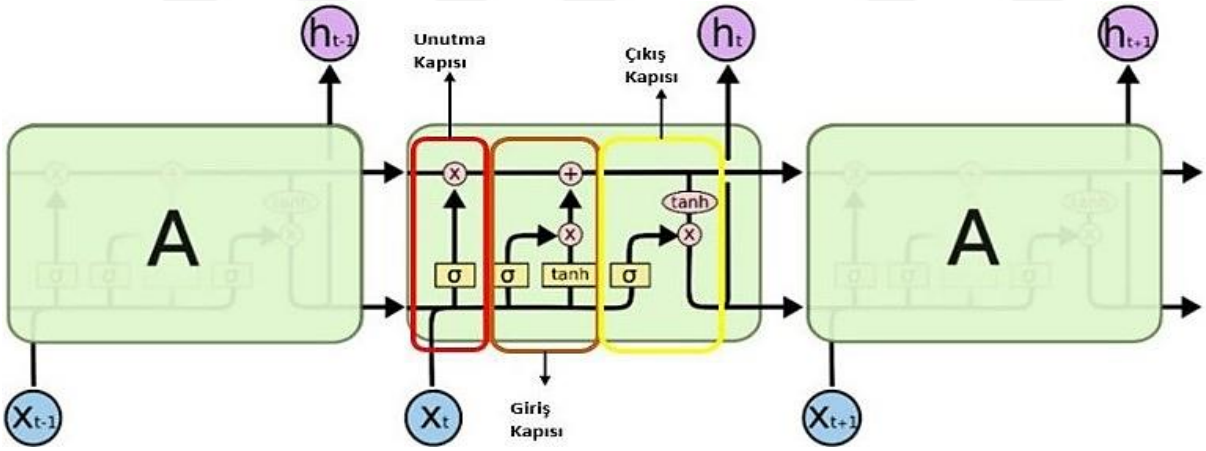


Şekil 2.6. RNN yapısı.

Kaynak: (Kalkavan, 2020: 247)

Uzun-kısa süreli hafıza (LSTM-Long Short-Term Memory)

Derin RNN modelinin geliştirilirmiş hali olan LSTM modeli aynı zamanda RNN ağının kaybolan gradyan sorununun üstesinden gelmektedir. Hochreitear ve Schmidhuber 1997 yılında bu modeli ortaya çıkarmalarının nedeni, aralarında boşluklar olan zaman serileri RNN yapısı için zor bir yapıya sahiptirler. Bu nedenden dolayı RNN yapılarına hafıza eklenerek verilerin gerekliliğine bakılması ve veri gereksiz görülürse bu verilerin hafızadan silinmesi işlemi yapılmaktadır (Doğan & Türkoğlu, 2019: 414) (Kalkavan, 2020: 247).



Şekil 2.7. LSTM yapısı

Kaynak: (Kalkavan, 2020: 248)

Unutma kapısının kullandığı formül aşağıdaki gibidir.

$$f_i^{(t)} = \sigma\left(b_i^f + \sum_j U_{i,j}^f x_j^{(t)} + \sum_j W_{i,j}^f h_j^{(t-1)}\right) \quad (2.2)$$

Yukarıda gösterilen formülde b^f, u^f ve W^f 'nin gösterdiği değerler; unutma geçidi ön yargı değerleri, girdi ağırlıkları ve yineleme ağırlıklarını göstermektedir.

$$s_i^{(t)} = f_i^{(t)} s_i^{(t-1)} + g_i^{(t)} \sigma\left(b_i + \sum_j U_{i,j} x_j^{(t)} + \sum_j W_{i,j} h_j^{(t-1)}\right) \quad (2.3)$$

$S_i^{(t)}$ durum birimini göstermektedir. Yukarıdaki formül $f_i^{(t)}$ kullanılarak sisitemin nasıl güncellendiğini göstermektedir.

$$g_i^{(t)} = \sigma\left(b_i^g + \sum_j U_{i,j}^g x_j^{(t)} + \sum_j W_{i,j}^g h_j^{(t-1)}\right) \quad (2.4)$$

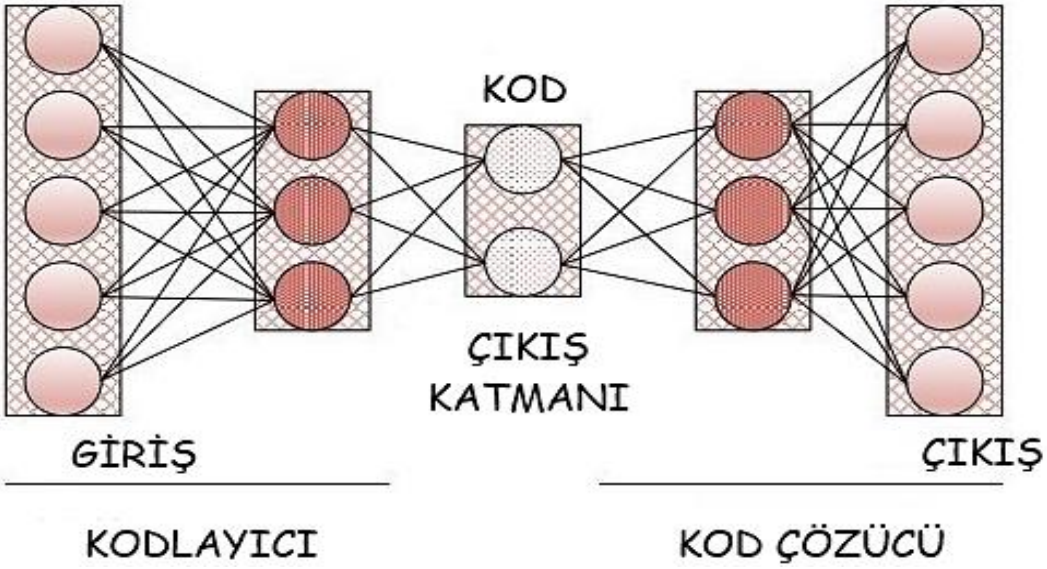
$g_i^{(t)}$ yapısı "harici girdi geçit birimini" temsil etmektedir. Unutma geçidine benzemesine karşın hesaplanma kısmında kendi parametreleri ile hesaplanır.

$$\begin{aligned} h_i^{(t)} &= \tanh\left(s_i^{(t)}\right) q_i^{(t)} \\ q_i^{(t)} &= \sigma\left(b_i^o + \sum_j U_{i,j}^o x_j^{(t)} + \sum_j W_{i,j}^o h_j^{(t-1)}\right) \end{aligned} \quad (2.5)$$

$h_i^{(t)}$, LSTM yapısının hücre çıktısını, $q_i^{(t)}$ ise çıktı geçidini göstermektedir. LSTM yapısının bir alt elemanı sayılabilecek GRU (Kapılı Tekrarlayan Birim) yapıları vardır. Bu yapıların LSTM'den farkı unutma işlemi ve durum güncelleme işlemini tek bir birimin yapmasıdır (Goodfellow vd., 2018: 411).

Oto-kodlayıcılar (Auto-Encoder)

Denetimsiz bir öğrenme algoritmasına sahip bu mimari 3 katmandan (Giriş, Gizli ve çıkış) oluşmaktadır. Giriş katmanından gizli katmanda bulunan 'Code' kısmına kadar bir 'Encode' yani kodlama işlemi yapılmaktadır. Burada boyutsal olarak bir azaltma işlemi yapılmıştır. Daha sonrasında ise bu olayların tam tersi yapılarak 'Decode' yapılması yani kod çözme işlemi yapılarak modelin çıktı verileri oluşturulur. Bu yapının kullanım alanları olarak; gürültülü resimlerin sınıflandırılması, enerji tahmini, siber güvenlik, bankacılık gibi alanlar örnek verilebilir. (Doğan & Türkoğlu, 2019: 416) (Mosavi vd., 2019: 8).

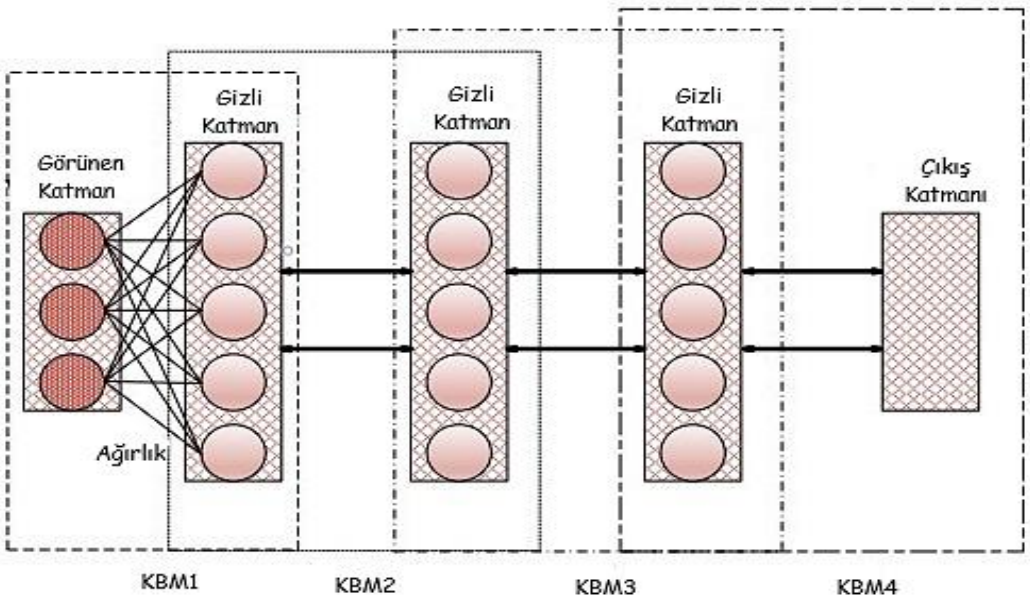


Şekil 2.8. Autocoder yapısı

Kaynak: (Mosavi vd., 2019)

Derin inanç ağları (Deep Belief Networks)

Derin İnanç Ağları, KBM (Kısıtlamalı Boltzman Makinesi) temelli çalışan çok katmanlı karışık (hibrit) yapılı bir mimaridir. Her katmandaki KBM'ler kendinden önceki katmanlar ile iletişim halindedirler. Öncelikle bir ön çalışma yapılarak ağırlıklar rasgele atanmış olmaktan kurtulup belirlenir sonrasında ise ileri - beslemeli bir şekilde ağ çalışmaya başlar (Altan, 2019: 324) (Mosavi vd., 2019: 9).

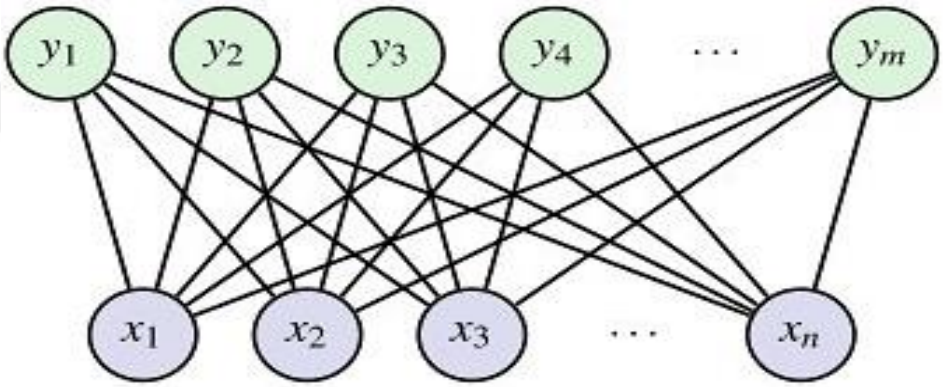


Şekil 2.9. DBN yapısı

Kaynak: (Mosavi vd., 2019: 10)

Kısıtlamalı Boltzman makinesi (Redistrict Boltzman Machine)

Kısıtlamalı Boltzman Makine'leri, Boltzman Makineleri'nin (BM) özelleştirilirmiş halidir. BM'ler Hoptfield Ağı'nın stokastik bir versiyonudur ve gözetimsiz öğrenme tipindedir. Günümüzde RBM'ler sınıflandırma ve derin öğrenme mimarisi oluşturma alanlarında kullanılmaktadır. RBM'lerin katmanları görünür ve gizli olmak üzere iki sınıfa ayrılmaktadır. Görünür katmanı giriş katmanı olarak düşünebilir. Her katmandaki eleman kendi katmanındaki elemanlar hariç diğer katmandaki elemanlarla iletişim halindedir. BM'ler ile arasındaki fark da buradan gelmektedir. Aynı katmanlar arasında bir iletişim bulunmamaktadır (Karasulu, 2018: 224) (Fischer & Igel, 2012: 23).



Şekil 2.10. RBM yapısı.

Kaynak: (Montúfar, 2018: 6)

2.6. Derin Öğrenmede Kullanılan Önemli Parametreler

Derin öğrenme, yapay sinir ağları tabanlı bir yapı olduğu için yapay sinir ağları tarafından kullanılan tüm kavramlar derin öğrenme için de geçerlidir. Bunlar;

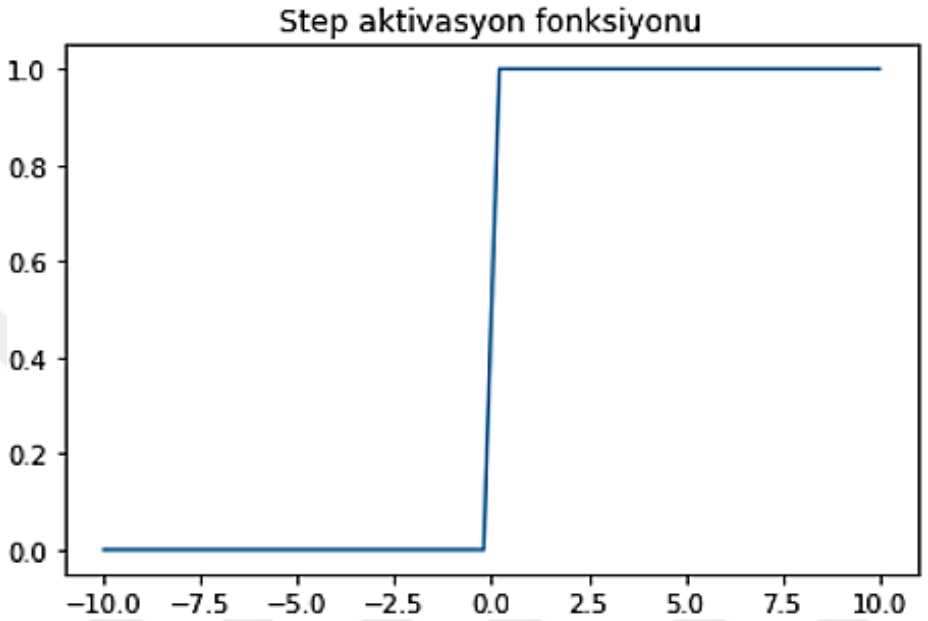
- Aktivasyon fonksiyonları
- Optimizasyon fonksiyonları
- Maliyet (Kayıp) fonksiyonları

2.6.1. Aktivasyon fonksiyonları

Aktivasyon fonksiyonlarının derin öğrenmedeki önemi şu şekildedir. Aktivasyon fonksiyonları ağda kullanılmadığı zaman katmanlar arası iletilen bilgiler lineer şekilde olmaktadırlar. Fakat yapılmak istenen işlem lineer olmayan bir yapıda ve karmaşıklığı yüksek olduğu zaman (ses, görüntü, video gibi) bunun işlenmesi çok zor olacak ve ağın öğrenmesi

sınırlı kalacaktır. Ağın bu sınırlandırılmadan kurtulması için ağırlık ve girişlerin çarpımlarının toplamları bir aktivasyon fonksiyonuna alınması gerekmektedir.

Step (Basamak) fonksiyonu: Bu fonksiyon 0 ve 1 gibi iki değere sahip olduğundan genellikle doğrusal ayrılabilir işlemler için kullanılır. Yapı olarak sınırlı olduğu için kullanım alanı kısıtlıdır.

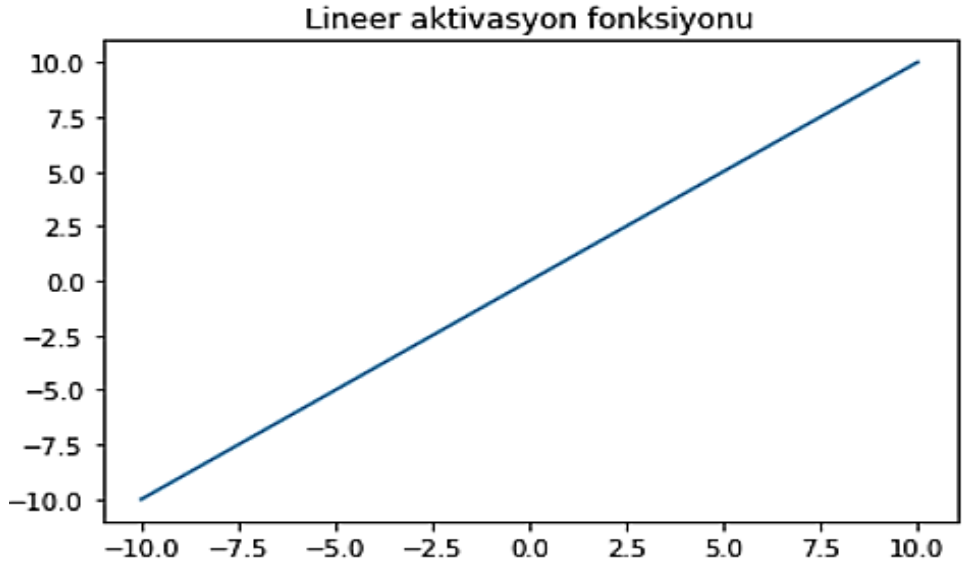


Şekil 2.11. Step aktivasyon fonksiyonu şekli.

En büyük dezavantajlarından biri türevinin sıfır olmasından dolayı geriye yayılımda hataları güncelleyememesidir. Şekil 2.11'de formül halinde görüldüğü üzere 0 değerine eşit veya altındaki tüm değerleri 0 olarak çıktı verir, 0'ın üstündeki tüm değerleri ise 1 olarak döndürür (Szandala, 2021: 3).

$$f(x) = \begin{cases} 0, & x \leq T \\ 1, & x > T \end{cases} \tag{2.6}$$

Lineer (Doğrusal) fonksiyon: Step fonksiyonuna oranla daha işlevsel bir yapıdadır. Step fonksiyonlarında görülen türevi 0 olma ve geriye doğru hata güncellememe durumları lineer fonksiyonlar için gözükmez.

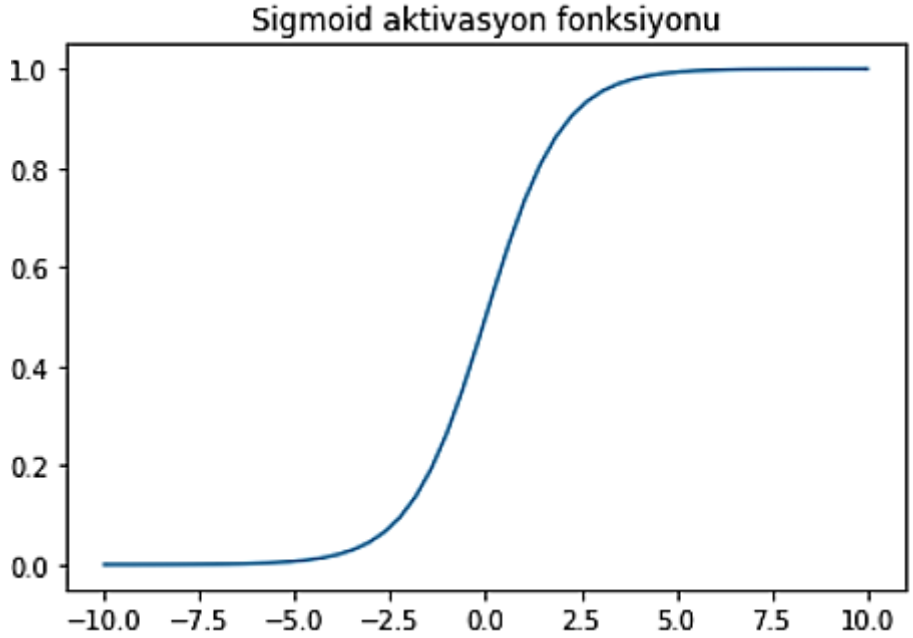


Şekil 2.12. Lineer aktivasyon fonksiyonu şekli.

Lineer fonksiyonun sorunu ise her türev durumunda değeri değişen sabit bir sayı döndürmesidir. Değişkenden bağımsız bir durum olması iniş gradyen inişlerinde sorun yaratmaktadır. Çok katmanlı bir derin öğrenme ağında kullanım oranı çok düşüktür (Szandala, 2021: 4).

$$f(x) = a * x, a \in R \tag{2.7}$$

Sigmoid fonksiyon: Bundan önceki diğer fonksiyonlardan farkı lineer bir şekilde olmamasıdır. ‘S’ Harfi şeklinde bir görüntü çizen bu fonksiyonun sınırları 1 ile 0 arasındadır.

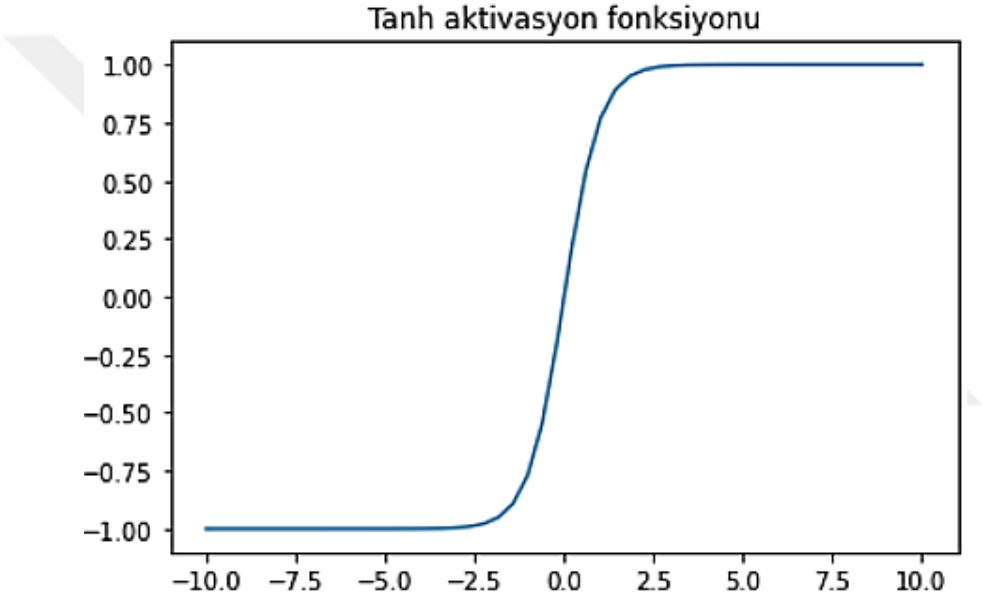


Şekil 2.13. Sigmoid aktivasyon fonksiyonu şekli.

Bu aktivasyon fonksiyonu sığ ağlar için uygundur. 0 ve 1 arasında olması değer olarak kısıtlama yaratsa da çıkış değerlerinin patlama yapmasını önlediği için çokça kullanılmaktadır. Bu fonksiyonun önemli bir eksikliği ise X düzlemindeki değişim kadar Y düzleminde bir değişim olmadığı için "kaybolan gradyen" durumunun ortaya çıkmasıdır. Bunun sonucu olarak ağdaki öğrenme, zaman olarak daha geç olmaktadır (Szandala, 2021: 5).

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{2.8}$$

Hiperbolik tanjant ($tanh$) fonksiyonu: Sigmoid fonksiyonu ile benzer bir yapıya sahip olan bu fonksiyonun görünürde sigmoid fonksiyonundan farkı, Y değerlerinin 0 ile 1 arasında değil 1 ile -1 arasında olmasıdır.

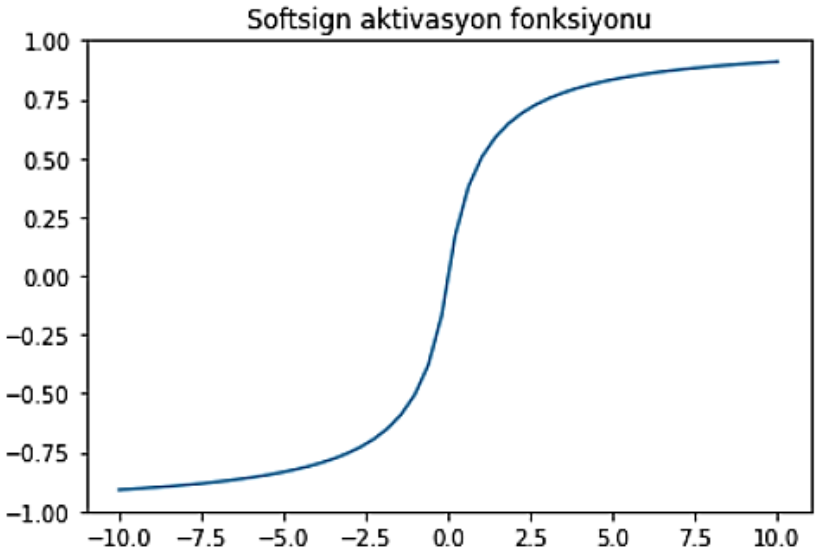


Şekil 2.14. $tanh$ aktivasyon fonksiyonu şekli.

Y değerlerinin farklı olmasının yanında sigmoid fonksiyondan bir diğer farkı ise negatif bir alanı kapsadığı için giriş değerleri negatif olduğu zaman negatif çıkış verecek ve ağda bir takılma durumu olmayacaktır. Ayrıca türevi sigmoid fonksiyonundan daha büyük olduğu için sigmoid fonksiyonundaki öğrenme zaman gecikmesi $tanh$ fonksiyonunda olmamaktadır. Ayrıca sigmoid fonksiyonundaki zikzak çizme sorunu da görülmemektedir (Szandala, 2021: 6).

$$f(x) = \tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \tag{2.9}$$

Softsign fonksiyonu: Bu fonksiyon da sigmoid ve $tanh$ gibi görsel olarak 'S' harfine benzemektedir. Büyümesi üstel olarak değil polinom şeklinde olarak gerçekleşir.

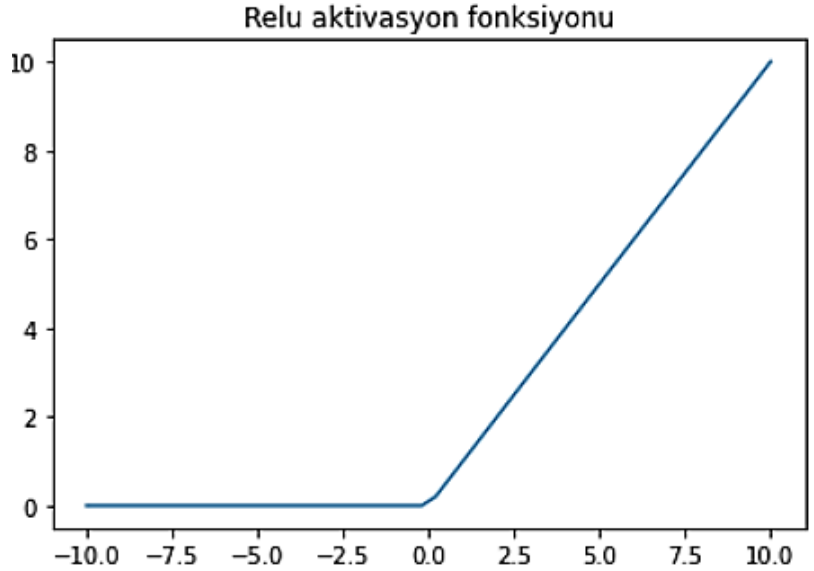


Şekil 2.15. Softsign aktivasyon fonksiyonu şekli.

Gradyan kaybolması durumu bu fonksiyonda azaltılmıştır ve böylece öğrenme durumu ve hızı artmıştır. Y-ekseninde aralık -1 ile 1 arasındadır fakat sigmoid ve $tanh$ fonksiyonlarındaki gibi bir doyma durumu olmaması eğitimi pozitif yönde etkiler. Eksik söylenebilecek yanı ise türevlerinin daha karmaşık bir yapıda olması gradyanı hesaplarken az ya da çok aşırılığa kaçabilmektedir (Szandala, 2021: 7).

$$f(x) = \frac{x}{1+|x|} \tag{2.10}$$

Relu fonksiyonu: Diğer bir adı ‘rampa fonksiyonu’ dur. Elektrik mühendisliğindeki yarım dalga doğrultma işlemine benzemektedir. İlk kullanıldığı model biyolojik bazlı, matematiksel gerekçelere sahip dinamik bir ağdır.



Şekil 2.16. Relu aktivasyon fonksiyonu şekli.

2011'de yapılan bir çalışmayla birlikte diğer aktivasyon fonksiyonlarından daha etkili olduğu kanıtlanmıştır. 2018'de yapılan bir çalışmaya göre ise en popüler aktivasyon fonksiyonu olduğu açıklanmıştır. Hız açısından diğer bilinen aktivasyon fonksiyonlarından yaklaşık 6 kat daha hızlıdır, ileri ve geri yayılım hızı yüksektir. Özellikle CNN yapılarında kullanım oranı fazladır (Szandala, 2021: 9).

$$f(x) = x^{+} = \max(0, x) \tag{2.11}$$

2.6.2. Optimizasyon algoritmaları

Derin öğrenmede sistemin başarısı hata değerinin minimalize edilmesi ile doğru orantılıdır. Optimizasyon algoritmalarının yapılardaki amacı ise bu hata değerlerini en az seviyeye indirmektir. Gradyan bilgisi açısından optimizasyon yöntemleri üçe ayrılı. Bunlar; birinci mertebe (first – order), yüksek mertebe (high – order) ve türevsiz (derivative – free) yöntemleridir. Yüksek mertebe yöntemlerin birinci mertebe yöntemlere karşı üstünlüğü, yakınsama işlemini daha hızlı bir şekilde yapmasıdır. Bu yöntemin zorluğu ise Hessian matrisinin ters matrisinin çalıştırılması ve depolanması gerekmektedir. Türevsiz yönteminin ana kullanım yerleri ise fonksiyonun türevinin hesaplanmasının zor ya da hiç olmadığı yerlerdir. Derin öğrenme yapısında kullanılan SGD, Adam, Adagrad, RMSprop gibi yöntemleri birinci mertebe yöntemlerin içinde bulunmaktadır (Sun vd., 2020: 4).

SGD (Stochastik Gradient Descent): Toplu gradyan inişi (batch gradient descent) yönteminin geliştirilmiş hali olarak tanımlanabilir. Toplu grandyen inişi yapısı, çevrimiçi güncellemeye izin vermez ayrıca büyük ölçekli veriler karşısında yaptığı her yineleme işleminde yüksek hesaplama karmaşıklığı ortaya çıkar. SGD'nin ana fikri, gradyan değerini doğrudan hesaplamak yerine yineleme başına güncelleyerek işlem yapmasına dayanmaktadır.

SGD yapısı dik yüzey eğrilerinde bulunan eğim nedeniyle global optimum değerine ulaşmada sıkıntılıdır. Gürültülü ve küçük eğimler de SGD için sorun teşkil edebilecek yapılardır. Bu durumlardaki öğrenme güçlüğünü aşmak amacıyla SGD yapıları "momentum" ile modifiye edilmiştir (Soydaner, 2020: 3).

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta J\left(\theta_t; x^{(i)}; y^{(i)}\right) \tag{2.12}$$

Adagrad: SGD yapısındaki öğrenme katsayısı (learning rate) manuel olarak ayarlandığı için optimum değeri bulmak zor bir iştir. SGD'ye ait bu problemi çözmek amacıyla kullanılan bir diğer optimizasyon algoritması ise Adagrad'dır. Çalışma şekli tüm kare gradyanların

toplanılması sonrası parametrelerin öğrenme katsayıları, bu toplamın karekköküne ters orantılı olarak hesaplanır.

İkinci dereceden problemler için iyi bir performans veren bu algoritma, sinir ağları eğitiminde erken durma problemi yaşatmaktadır. Bunun sebebi öğrenme oranının aşırı küçük bir değere sahip olmasıdır. Ayrıca derin sinir ağlarda kare gradyen birikmesi sonucu öğrenme katsayısında aşırı düşüşler gözlemlenir (Sun vd., 2020: 6) (Soydaner, 2020: 5).

$$g_{t,i} = \nabla_{\theta_t} J(\theta_{t,i})$$
$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,u}+\epsilon}} g_{t,i} \quad (2.13)$$

RMSprop: Adagrad algoritmasında bulunan öğrenme katsayısının küçülerek sıfıra gitme sorununu çözme amacıyla geliştirilen bir algoritmadır. Durağan ve dış bükey olmayan sorunları optimize etmek için uygun bir yapıdır. Dezavantajı ise güncelleme işlemini yaparken yerel minimum değerler etrafında sıkışabilmesi ihtimalidir (Soydaner, 2020: 6) (Sun vd., 2020: 6).

$$E[g^2]_t = 0.9E[g^2]_{t-1} + 0.1g_t^2$$
$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t+\epsilon}} g_t \quad (2.14)$$
$$g_t = \nabla_{\theta_t} J(\theta_t)$$

Adam: En çok kullanılan optimizasyon algoritmalarından biri olan Adam algoritması, Adagrad algoritmasının çevrimiçi çalışma özelliği ile RMSprop'un sabit olmayan ortamlarda iyi çalışabilmesi özelliğini almıştır. Adam algoritmasının bazı durumlarda yakınsamama problemi yaşaması dışında çok sayıda avantajı bulunmaktadır. Bunlar;

- Kolay uygulanabilmesi.
- Bellekte yer kapladığı alanın azlığı.
- Hesaplama verimliliği.
- Durağan olmayan ve aşırı gürültülü problemler için de düzgün sonuçlar verebilmesidir (Soydaner, 2020: 7) (Sun vd., 2020: 7).

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$$
$$m_t' = \frac{m_t}{1-\beta_1^t}, v_t' = \frac{v_t}{1-\beta_2^t} \quad (2.15)$$
$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{v_t'+\epsilon}} m_t'$$

2.6.3. Maliyet (Kayıp) fonksiyonları

Maliyet fonksiyonlarının amacı sistemin oluşturduğu hata değerine göre çalışma performansını ölçmeyi sağlamaktır. Çıkış katmanında konuşlanan bu fonksiyonlar 0 değerine ne kadar yakın olursa sistemin performansı o kadar iyi, 1 değerine ne kadar yakınlaşırsa o kadar kötüdür (Tan & Ziya, 2019: 42).

Derin öğrenme yapısı içinde regresyon işlemlerinde kullanılan çeşitli maliyet fonksiyonlarının aşağıda isimleri ve tanımlamaları verilmiştir.

Ortalama Mutlak Hata (MAE): Hataların mutlak değerlerinin toplamının toplam eleman sayısına bölümü ile bulunur. Mutlak hatanın ana fikri işaretleri farklı olan hata değerlerinin birbirlerini nötralize etmesini önlemektir. Gradyen ve hassasiyet hesaplaması zor olabilen bu fonksiyon düzgün dağılmış hataları tanımlamak için daha uygundur (Chai & Draxler, 2014: 1248) (Botchkarev, 2019: 59).

$$MAE = \frac{\sum_{i=1}^{n} |E_i|}{n} \quad \textbf{(}2.16\text{)}$$

Ortalama Mutlak Yüzde Hata (MAPE): MAE yapısına benzeyen bu fonksiyon mutlak hatanın yüzdeliğini alarak hata durumunu hesaplar. Üst sınırı çok yüksek olan tahminlerde bulunmamaktadır (Naser & Alavi, 2020: 5).

$$MAPE = \frac{100}{n} \sum_{i=1}^{n} |E_i|/|A_i| \quad \textbf{(}2.17\text{)}$$

Ortalama Kare Hata (MSE): Hataların karelerinin toplamının eleman sayısına bölünmesiyle bulunmaktadır. Hataların karesinin alınmasının nedeni zıt işaretli hata değerlerinin birbirlerini sönümlemesini önlemektir. Hata değerlerinin karesi alındığı için büyük hatalar daha çok vurgulanır (Botchkarev, 2019: 59).

$$MSE = \frac{\sum_{i=1}^{n} E_i^2}{n} \quad \textbf{(}2.18\text{)}$$

Karekök Ortalama Hata (RMSE): MSE değerlerinin karesinin alınarak bulunan bir hata metrik parametresidir. Yapısal olarak MAE'den farklı olduğu için model hatalarının normal dağılım durumunda olduğunda model iyileştirilmesinde ondan daha iyi sonuçlar verebilmektedir (Botchkarev, 2019: 59) (Chai & Draxler, 2014: 1248).

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} E_i^2}{n}} \quad \textbf{(}2.19\text{)}$$

2.7. Literatürde Yapılan Çalışmalar

Bilecik İlinde rüzgâr hızının tahminiyle ilgili yapılmış çalışmalar araştırıldığında (Yeşilnacar & Gün, 2011)'ın yayımladıkları *"Bilecik İlinin Yapay Sinir Ağları ile Rüzgâr Hızı, Basınç ve Sıcaklık Tahmini"* isimli yüksek lisans tezi çalışmasının olduğu görülmektedir. Bu tez çalışmasında kullanılan ortam MATLAB olarak seçilmiştir. Giriş parametresi olarak sadece rüzgar hızı değil ortalama basınç ve ortalama sıcaklık gibi farklı parametreler alınmış ve bunlar işlenerek ortalama rüzgar hızı tahmini yapılıştır. Yapılan bu tezde saatlik rüzgar hızları kullanılmamıştır. Bunun yerine günlük ortalama rüzgar hızları bulunarak 2010 yılının tüm aylarına ait günlük ortalama rüzgar hızları bulunmuştur. Bunu yanında tek ve çift günler ayrılarak her 2010 yılının tek ve çift günlerinin hataları ortaya çıkarılmıştır. En iyi sonucu ise Kasım ayında 0.0010 MSE değeri ile almıştır.

Bilecik ili rüzgar hızı tahmini ile alakalı diğer bir tez ise (Dokur vd., 2016)'un yaptığı *"Wınd Speed Modellıng Usıng Inverse Weıbull Dıstrubıtıon: A Case Study For Bilecik, Turkey"* adlı tezdir. Bu tezde "Weibull Dağılımı" ve "Ters Weibull Dağılımı" kullanılarak 2014 yılının aylık ortalama rüzgar hızı tahmini yapılmıştır. En iyi sonucu Haziran ayında Ters Weibull Dağılımı yöntemiyle 0.0640 RMSE değeriyle bulmuştur.

Bilecik ili dışında yapılmış rüzgâr hızı tahmin çalışmalarına bakıldığında (Altan & Karasu, 2020) *"Ayrıştırma Yöntemlerinin Derin Öğrenme Algoritması ile Tanımlanan Rüzgâr Hızı Tahmin Modeli Başarımına Etkisinin İncelenmesi"-'* adlı tezdir. Tekli modellerin belirli başarımla aşamadığından dolayı tezlerinde hibrid yapılar kullanmışlardır. Rüzgar tahmin çalışmalarında çokça kullanılan yöntemlerden biri olan "ayrıştırma yöntemleri" kullanmışlardır. Bu tezde "ampirik kip ayrıştırma", "topluluk ampirik kip ayrıştırma" ve "ampirik dalgacık dönüşümü" yöntemlerini LSTM derin öğrenme modeli ile birleştirerek hibrid yapılar ortaya çıkarmışlardır. Tezde işlem yapılan rüzgar hızlarının lokasyonları ise Bandırma, Bozcaada ve İpsala'dır. Bu bölgerin seçilmesinin amacı Türkiye'deki rüzgar hızı en potansiyeli en yüksek alanlar olmasıdır. Kullanılan rüzgar verileri saatlik değil 10 saatlik verinin ortalanması alınarak işlem yapılmışıtr. Seçilen 3 bölgenin tez sonuçlarına bakıldığında sade LSTM yapılarının MSE değerleri 1.8892, 3.2929, 1.1318 gibi değerler çıkmıştır. Hibrid LSTM yapıları içindeki en iyi sonuçları ampirik dalgacık dönüşümlü LSTM yapıları vermiştir. MSE sonuçları 0.0046, 0.0080, 0.0082 çıkmıştır.

Bu konuda hakkında yazılan bir diğer tez de (Liu vd., 2020)'ya aittir. *"Wind Speed Forecasting Using Deep Neural Network With Feature Selection"* isimli tezinde Kuzeybatı Çin'de bulunan 'Neimenggu' şehrinini rüzgar hızlarını zaman serisine dönüştürerek rüzgar

hızının tahmin işlemi yapılmıştır. Test bölümünde 4 farklı kategoriye ayırıp yapılan bu çalışmada rüzgar hızı tahmini LSTM, autoencoder, LSTM-Autocoder hibrit ve çok katmanlı perceptron modelleri ile yapılıp, doğruluk değerlerinin karşılaştırması yapılmıştır. Çıkan test RMSE değerlerine bakıldığında MLP (çok katmanlı perceptron) en iyi RMSE değeri 0.9262 çıkmıştır. Autocoder yapısının en iyi RMSE sonucu ise 0.5982 bulunmuştur. LSTM yapısının elde ettiği en iyi RMSE sonucu ise 0.5302 olmuştur. Hibrit yapının sahip olduğu en iyi RMSE değeri ise 0.3863 olarak belirlenmiştir.

Bir başka tez de (Kumar vd., 2019) ”*Deep Learning based Wind Speed Forecasting- A Review”* adlı yaptığı tez olup Dünya’da yapılan farklı rüzgar hızı tahmini tezlerini tek bir makalede toplamıştır. 2004 – 2018 arası yaklaşık 19 tane çeşitli yöntemlerle yapılmış olan rüzgar hızı tahmini ile alakalı tezlerin kullandığı modeller, kullandığı parametreler, avantajları, dezavantajları hakkında çeşitli bilgiler verilmiştir.

(Sergio & Ludermir, 2015) ise ”*Deep Learning for Wind Speed Forecasting in Northeastern Region of Brazil”* adlı tezinde derin öğrenme ile Brezilya’nın güney doğusundaki rüzgar hızı tahminini yapmıştır. Rüzgar hızları kullanılarak zaman seirisinden rüzgar hızı tahmini yapılan bu tezde veriler “DBN (Derin İnanç Ağı)”, “MLP (Çok Katmanlı Perceptron)” ve “SDAE (Yığılmış Gürültü Giderici Otomatik Kodlayıcılar)” modellerine entegre edilerek çeşitli tahminler elde edilmiştir. Modellere göre en iyi MSE değerleri yazıldığında DBN için 0.6449, MLP için 0.5992, SDAE için ise 0.7119 sonuçları ortaya çıkmıştır.

(Khouloud vd., 2021) ise Fourier dönüşümü ile çeşitli derin öğrenme modellerini birleştirerek Güneybatı Cezayir’in rüzgâr hızı ölçümünü yapmıştır. Rüzgar hızlarının zaman serilerine dönüştürerek 1 ve 3 saatlik rüzgar hızı tahmini yapmıştır. Kullandığı modeller ise MLP, LSTM, Fourier dönüşümlü MLP, Fourier dönüşümlü LSTM, Autocoder-LSTM ve Fourier dönüşümlü Autocoder-LSTM gibi hibrit yapılar kullanmıştır. 1 saatlik sonuçlara bakıldığında en iyi sonucu Fourier dönüşümlü autocoder-LSTM yapısı vermiştir RMSE değeri 0.035’tir. 3 Saatlik sonuçlarda da aynı şekilde en iyi sonucu Fourier dönüşümlü autocoder-LSTM yapısı vermiştir RMSE değeri 0.4870’tir.

3. DENEYSEL ÇALIŞMALAR

Rüzgar hızı tahmininde iki farklı yöntem kullanılmaktadır. Birincisi sıcaklık, basınç, nem gibi rüzgar hızıyla ilişkili parametreleri kullanarak, ikincisi ise sadece rüzgar hızı verilerini ele alarak yapılan tahmindir. Bu çalışma kapsamında Bilecik Meteoroloji Müdürlüğünden alınan 10 yıllık satlik rüzgar hızı verileri zaman serisi haline getirilerek bir derin öğrenme modeli olan LSTM modelinde işlenmiştir. Bu çalışma genel olarak 3 kısımdan oluşmaktadır:

- Veri Toplanılması
- Veri Ön İşlenmesi
- Derin Öğrenme Modeli Oluşturma

Veri Toplanılması:

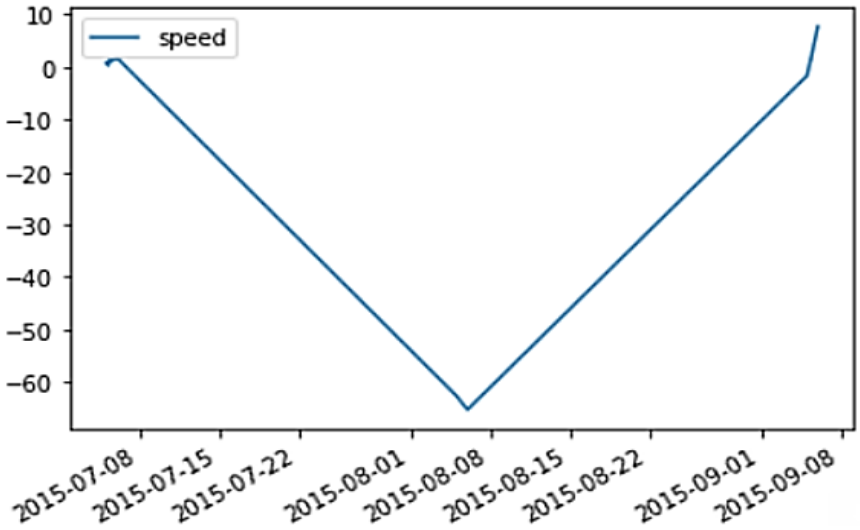
Rüzgar hızlarının temini için Bilecik Meteoroloji Müdürlüğü'nden 10 yıllı kapsayacak şekilde saatlik hız verileri alınmıştır. 10 yıllık verilerin alınmasının sebebi eğitim verilerinin sayısını arttırarak kurulacak olan modelin test tahmin hata oranını minimize etmektir.

Veri Ön İşlenmesi: 10 Yıllık saatlik verilerin alınması ile beraber yapılması gereken iki işlem vardır. Bunlardan bir tanesi alınan eksik verilerin giderilmesi diğeri ise derin öğrenme modeline uygun şekilde alınan verilerin biçimini hazırlamaktır.

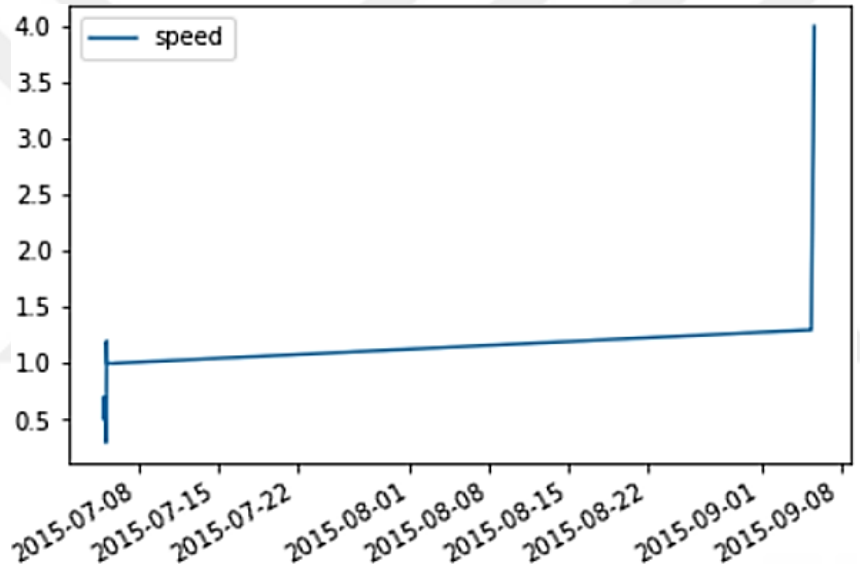
Veri eksikliği sorununun çözümü de iki farklı şekilde olmaktadır.

- 2, 3, 6 saat vb., kısa süreli veri eksikliklerinde eksik saat aralığının başlangıç ve bitim zamanlarında değeri bulunan verilerin aritmetik ortalaması alınarak bu eksik veri yerine azalan ya da çoğalan şekilde değerlerin yerleştirilmesi işlemi yapılmıştır.
- 12 saat ve üzerinde olan saatlik boşluklarında ise interpolasyon yöntemiyle bu boşluklar doldurulmuştur.

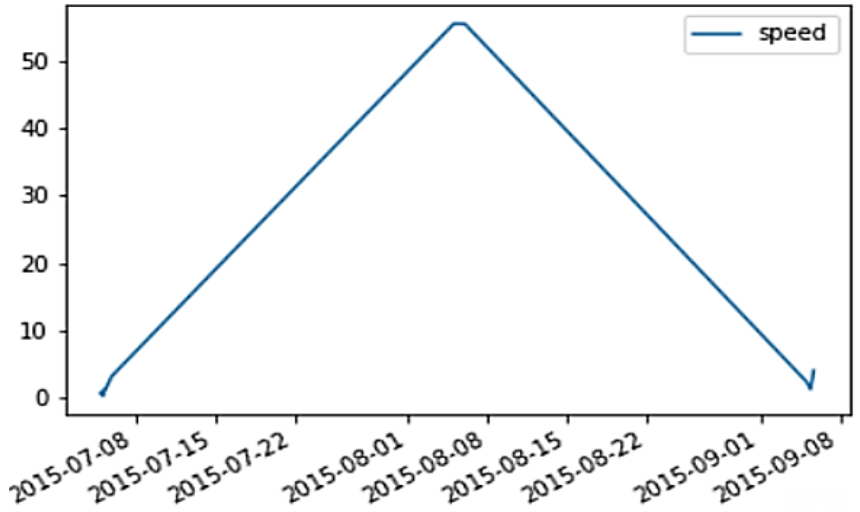
Aşağıda ki şekillerde 07.08.2015 - 09.08.2015 tarihleri arasında 2 günlük saatlik rüzgar hızlarının eksik olmasından dolayı farklı interpolasyon yöntemleri ile doldurulması gösterilmiştir. Burada bulunan en iyi yöntem diğer boşluk bulunan zaman aralıklarında da uygulanarak devam edilmiştir.



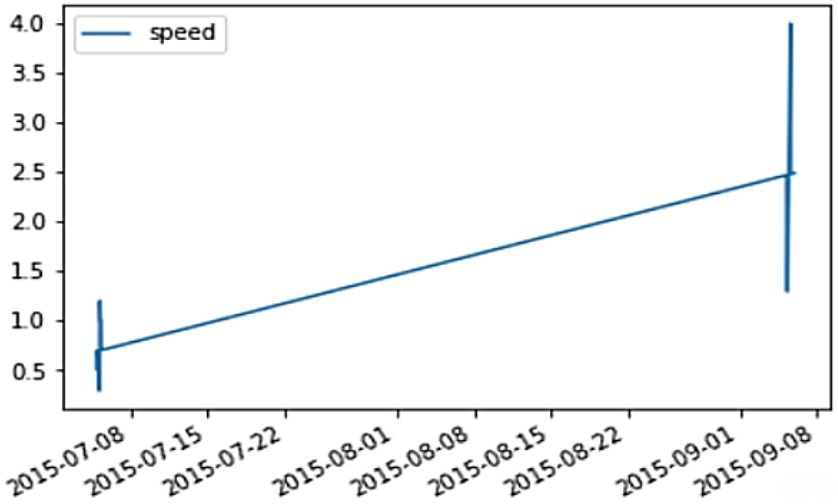
Şekil 3.1. Spline interpolasyonu 3.derece



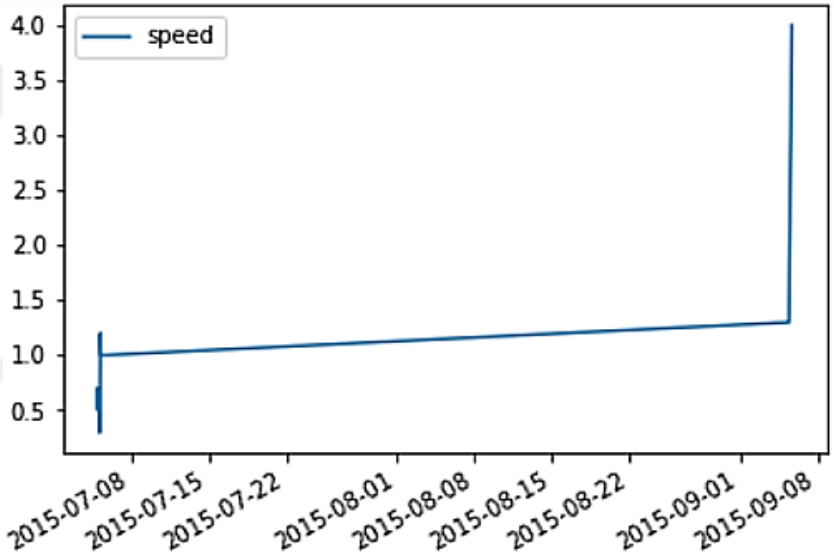
Şekil 3.2. Polinomal interpolasyonu 1.derece



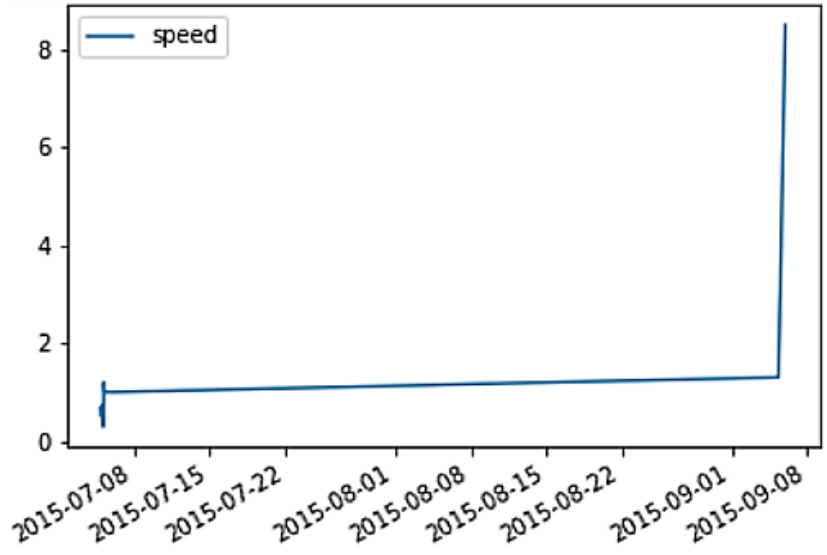
Şekil 3.3. Polinomal interpolasyonu 3.derece



Şekil 3.4. Spline interpolasyonu 1.derece



Şekil 3.5. Akima interpolasyonu.



Şekil 3.6. Kübik Hermite Spline interpolasyonu

Sonuçlara bakıldığında kısa süreli rüzgar hızı değişikliklerinin minimum olması gerekliliğine uyan en iyi sonucun ‘Spline interpolasyonu 1.derece’ olduğu görülmüştür. Tablolardaki diğer uzun süreli eksik rüzgar hızlarıda interpolasyon yöntemi kullanılarak doldurulmuştur.

Veri eksikliği yukarıda anlatılan iki çeşit yöntem ile giderildikten sonra yapılması gereken diğer bir işlem verilerin LSTM modeline girebilecek bir şekle getirilmesini sağlamaktır. Bilecik Meteoroloji Müdürlüğü’nden alınan veriler aşağıdaki tablodaki gibidir.

T.C.
Tarım ve Orman Bakanlığı
Meteoroloji Genel Müdürlüğü

Yıl/Ay: 2020/1 İstasyon Adı/No: BİLECİK/17120

Gün/Saat	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
1	0,00	0,20	0,10	0,00	0,00	0,00	0,50	0,00	1,50	1,50	1,50	1,50	1,50	2,60	2,10	0,50	1,00	0,00	1,00	1,40	1,60	0,50	1,30	0,30
2	0,50	0,20	0,00	0,50	1,00	0,00	0,00	0,50	0,50	0,00	0,00	0,50	0,00	0,50	0,00	0,50	0,00	0,50	0,00	0,70	0,00	0,50	0,00	0,00
3	0,00	0,80	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,50	0,00	0,00	0,00	0,50	0,00	0,00	0,00	0,00	0,00	0,00
4	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	1,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
5	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,50	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
6	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,50	0,50	0,00	0,00	0,00	0,00	0,00	1,00	0,50	0,00	0,00	0,00	0,50	0,00
7	1,00	0,00	0,50	1,00	0,00	0,00	0,00	1,00	0,50	1,00	0,50	0,00	1,00	2,10	2,60	2,10	2,10	1,50	2,10	3,10	2,60	2,10	2,10	2,60
8	3,10	1,50	2,60	3,10	2,10	3,10	1,50	2,60	2,10	2,60	2,10	1,00	0,50	2,10	0,50	0,50	1,50	1,00	0,00	0,00	0,50	0,00	1,50	1,50
9	0,00	0,00	0,00	0,50	0,50	0,00	0,50	0,50	0,00	0,50	0,50	0,50	0,00	0,50	0,50	0,50	0,00	0,00	0,00	0,30	0,50	0,00	0,00	0,00
10	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,50	0,50	1,50	1,00	1,00	0,50	1,00	1,50	0,50	0,50	0,20	1,10	0,50	0,70	1,00
11	1,00	0,40	0,50	0,50	0,70	1,20	1,00	0,50	1,00	2,60	2,60	2,10	2,10	2,60	2,60	2,60	2,10	1,50	1,00	0,70	2,90	0,00	0,00	0,00
12	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
13	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,50	1,00	1,00	0,50	1,00	0,00	0,00	0,00	0,00	0,00	0,00	0,50	0,50	0,50	0,00
14	0,00	0,00	0,50	0,50	1,00	0,00	1,00	0,50	1,00	1,50	0,50	0,50	0,00	0,50	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
15	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,50	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
16	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,50	0,00	0,00	0,00	0,50	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,50	0,00
17	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,50	0,50	0,00	0,50	0,00	0,00	0,00	0,50	0,50	0,00	0,50	0,00	0,50	0,50
18	0,50	0,50	0,50	0,00	0,50	0,00	0,00	0,00	0,50	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,50	0,00	0,50	0,50	0,00	0,00	0,00	0,00
19	0,50	0,50	0,00	0,00	0,00	0,00	0,00	0,50	0,00	0,50	0,50	1,00	0,50	0,50	0,50	0,50	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,50
20	0,00	0,00	2,10	1,50	1,00	1,00	1,00	1,00	0,00	0,50	0,00	0,00	0,00	2,60	1,50	1,50	2,10	0,50	0,50	1,00	0,50	0,00	0,50	1,00
21	1,00	1,00	1,50	1,50	1,50	2,10	1,50	1,50	1,50	0,50	0,50	0,00	0,00	0,50	1,50	0,00	0,00	1,00	0,00	0,00	0,00	0,00	0,00	0,00
22	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,50	0,50	0,00	0,00	0,50	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
23	0,00	0,00	0,00	0,00	0,00	0,00	0,00	1,50	2,10	2,60	3,10	2,10	2,60	3,10	0,00	1,00	1,50	1,00	1,00	0,50	0,50	0,50	0,50	1,00
24	1,00	0,50	0,50	0,00	0,00	1,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,50	0,00	0,00	0,00	0,00	0,00	0,00	0,00	1,00	0,50	0,00
25	0,50	0,00	0,50	1,50	0,50	1,50	1,50	1,50	1,00	1,50	3,10	2,60	2,10	2,60	2,60	0,50	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
26	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,50	0,00	0,50	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
27	0,00	0,00	0,00	0,00	0,00	0,50	0,00	0,00	0,50	0,50	0,00	1,00	0,00	0,50	0,50	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
28	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,50	0,50	0,50	1,50	1,00	1,00	0,50	0,50	0,00	0,00	0,00	0,00	0,00	0,00
29	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	3,10	1,50	2,60	0,50	0,50	1,50	1,00	2,10	1,00	0,50	1,00	0,00	0,50

Şekil 3.7. Bilecik Meteoroloji Müdürlüğü’nden alınan rüzgar hızı tablosu

Bu verilerin işlenerek LSTM modeline uygun hale getirilmesi gerekmektedir. Python dilinde derin öğrenme modeline dışardan veri aktarımı yapılabilmektedir. Yani bir Excel tablosunu istenen hale getirerek bu derin öğrenme modeline entegre edilebilir. En çok kullanılan yöntem ise Excel’in ‘.csv’ tipine döndürüp derin öğrenme modeline entegre etmektir. Bunun için kodu ‘.csv’ formatıyla çalışabilecek hale çevirmemiz ve verilerinde bu formata dönüştürülmesi gerekmektedir.

Açılımı ‘comma seperated values (virgül ile ayrılmış değerler)’ olan bu Excel tipinde parametreler ile değerler virgülle ayrılır ve tüm değerler alt alta yazılır.

datetime,speed
2019-01-01 00:00,4.4
2019-01-01 01:00,2.8
2019-01-01 02:00,3.3
2019-01-01 03:00,3.7
2019-01-01 04:00,3.3
2019-01-01 05:00,3.1
2019-01-01 06:00,3.6
2019-01-01 07:00,3.4
2019-01-01 08:00,3.5
2019-01-01 09:00,2.7
2019-01-01 10:00,3
2019-01-01 11:00,2.9
2019-01-01 12:00.3.3

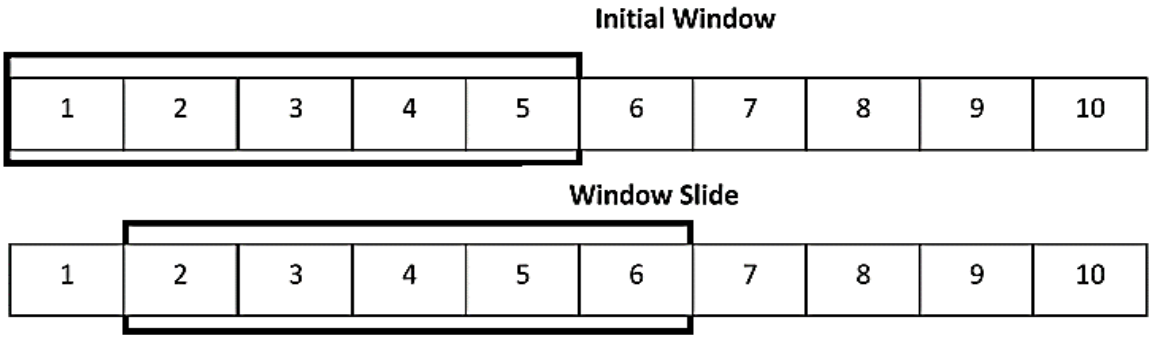
Şekil 3.8. Örnek 12 saatlik veri değer gösterimi

Derin öğrenme modeline sokulacak şekilde veriler ayarlandıktan sonra yapılacak son işlem ise derin öğrenme modelini oluşturmaktır.

Derin Öğrenme Modeli Oluşturma: Rüzgar hızı tahmininde iki farklı LSTM yapısı oluşturup bunların hangisinin daha etkin olduğu çalışmanın amaçlarından biridir. İlk olarak zaman serisi ve kullanılan modellerden tek değişkenli çift yönlü (univarate bidirectional) LSTM yapısının ana prensiplerinden biri olan kayan pencereler kavramlarının bu konu için ne anlama geldiğini belirtmek gerekmektedir.

Zaman serisi, zamana bağlı parametrelerin oluşturduğu veri topluluğun adıdır. Sıcaklık, rüzgar hızı, finans veriler vb., her zaman diliminde ilgili parametrenin bir verisi bulunmaktadır. Zaman değiştikçe bu veriler de değişmektedir (Güdelek & Özbayoğlu, 2013: 5).

Zaman serisinde kullanılan öğrenme metodu ‘gözetimli öğrenme’ metodudur. Yani makine veya derin öğrenme modelinin oluşturduğu sonuçlar gerçek veriler ile karşılaştırılarak sistemin doğruluk oranı hesaplanır. Zaman serisindeki veriler kullanılarak bir matris oluşturulur. Oluşturulan bu matrisin satır sayısı, tahmin edilmesi istenen büyüklük bazında çıktı sayısına eşittir. Bir alttaki satırın başlangıç zamanı bir öncekinin başlangıç zamanından 1 fazladır. Bu şekilde matris derin öğrenme modeline sokulabilecek bir şekle girer. Zaman serisinde yapılan bu kaydırma işlemine ‘kayan pencereler’ yöntemi denmektedir.



Şekil 3.9. Kayan pencereler yöntemi örneği

Kaynak: (Hota vd., 2017: 1148)

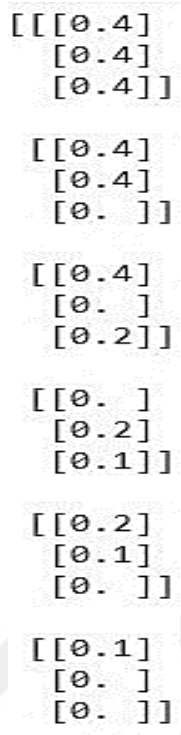
Yukarıdaki şekilde gösterildiği gibi genellikle 1 adımlık bir kaydırma işlemi yapılarak zaman serisinde kullandığımız pencerenin içindeki değerlerin bir matrise atılma işlemi gerçekleştirilir.

$$X = \begin{bmatrix} y_{N-1} & y_{N-2} & \cdots & y_{N-n-1} \\ y_{N-2} & y_{N-3} & \cdots & y_{N-n-2} \\ \vdots & \vdots & \vdots & \vdots \\ y_n & y_{n-1} & \cdots & y_1 \end{bmatrix}$$

Şekil 3.10. Örnek matris oluşturma işlemi

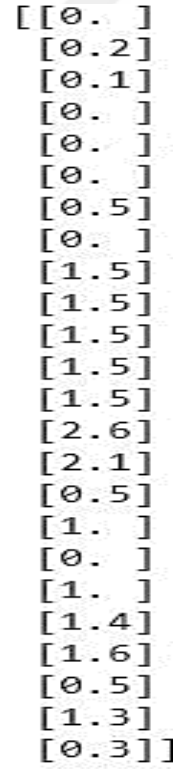
Kaynak: (Bontempi vd., 2013: 65)

Örneğin 3 saat girişli ve 24 saat tahminli bir yapı oluşturmak istiyorsak test kısmında kullanacağımız formül "test veri sayısı = giriş veri saati + tahmin saati" olacaktır. Bu örnek için test sayısı toplamda 27 saatlik bir genişlikte olmaktadır. Bu yapıyı 3'erli matrisler haline getirdiğimizde 24 tane 3'erli matris grupları oluşmaktadır. Bu matris grubunun her birini önceden eğittiğimiz modele sokarak 24 saatlik bir tahmini yapmasını sağlayıp bunu gerçek değerler ile karşılaştırarak sistemin doğruluk oranını tahmin edebilmekteyiz.



Şekil 3.11. 3'erli gruplandırılmış test veri seti.

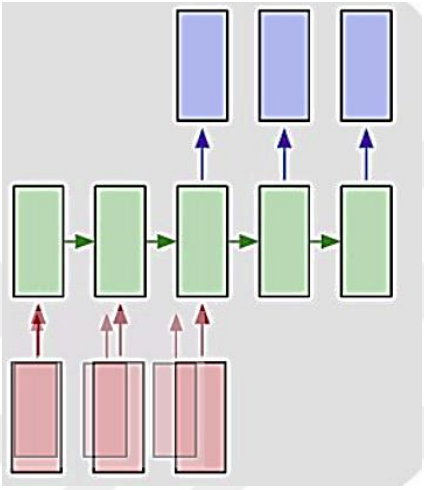
Yukarıda ki ana matrisin büyüklüğü $24x3x1$ şeklindedir. Yani 24 tane $3x1$'lik matrislerden oluşan büyük bir matris bulunmaktadır.



Şekil 3.12. 24 saatlik test veri seti.

Derin öğrenme modeline her 3x1'lik matris sokulduktan sonra her biri için 1 saatlik tahmin değeri bulunur. 24 tane olan bu tahmin değerleri ile gerçek test verileri karşılaştırarak sistemin doğruluk oranını bulunur.

Diğer çok adımlı vektör çıkışlı (multi–step bidirectional vector output) LSTM modelinin çalışma prensibi ise kayan pencere tarzında değil sabit bloklar halindedir. Yani test kısmında belirlenen sayıda giriş veri sayısını direkt işleyerek istenen çıktı sayısında tahmin yapmaktadır. 48 saat girişli tek bir blok alınarak bu derin öğrenmeye sokulur ve buradan 24 saatlik tahmin yapması beklenir.



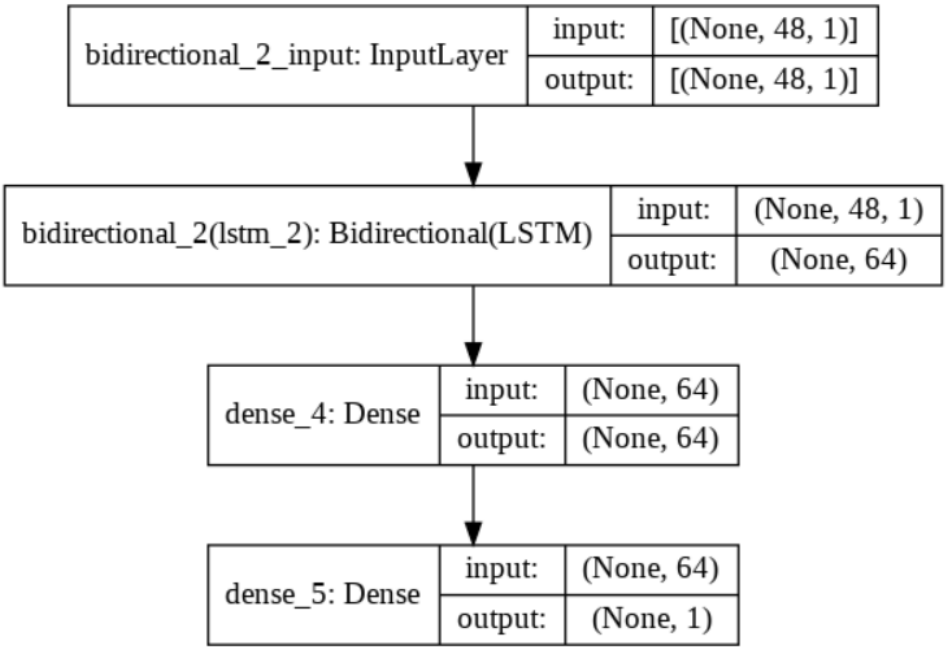
Şekil 3.13. Vektör yapılı LSTM modelinin çalışma şekli.

LSTM modeli oluşturulurken baz alınan kaynak (Brownlee, 2018: 130) olmuştur. İki farklı LSTM modeli alınarak bunların rüzgar hızı tahmininde hangisinin daha etkili olduğunun araştırılması yapılmaktadır. Seçilen modeller tek değişkenli çift yönlü (univarate bidirectional) ve çok adımlı vektör çıkışlı (multi–step bidirectional vector output) tipleridir. Bu iki modeli daha efektif hale getirmek için bu iki LSTM modelini de 'Bidirectional (iki yönlü)' olarak ayarlanmıştır. Bidirectional yapının normal yapıya göre avantajı geriye yayılım işlemi yaparak çıkacak hata oranlarını düşürmektir.

Model oluşturulurken katı bir kural silsilesi bulunmamaktadır. Model oluşturma, biraz 'deneme – yanılma' yapılarak parametreler ve katman yapılarıyla oynanarak yapılmaktadır. Bu şekilde yapılan çalışmada oluşturulan en etkili model ve parametreleri aşağıdaki şekildedir.

- 3 Katmanlı
- Learning Rate = 0.004 (Default hali 0.001)
- Batch_size =128
- Epoch = 20
- ‘Adam’ Optimizasyon Fonksiyonu

Sistemin katman sayısı belirlenirken giriş, gizli ve çıkış katmanı olmak üzere 3 katmandan başlanarak gittikçe arttırılması işlemi yapılmıştır. Yapılan denemeler sonucunda gizli katman sayısı arttırılıp toplam katman sayısı 4, 5, 6’ya kadar çıkarılsa dahi öğrenme işleminde bir artış gözükmemekte hatta düşüşler gözükmektedir. En optimum katman sayısı 3 aşağıdaki LSTM modeli olarak gözlemlenmiştir.



Şekil 3.14. LSTM modelinin .png çıktı halindeki yapısı.

LSTM modelin özeti Şekil 3.15’te verilmiştir.

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
bidirectional_2 (Bidirection (None, 64)                8704
_________________________________________________________________
dense_4 (Dense)              (None, 64)                4160
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 65
```

Şekil 3.15. LSTM yapısının özeti.

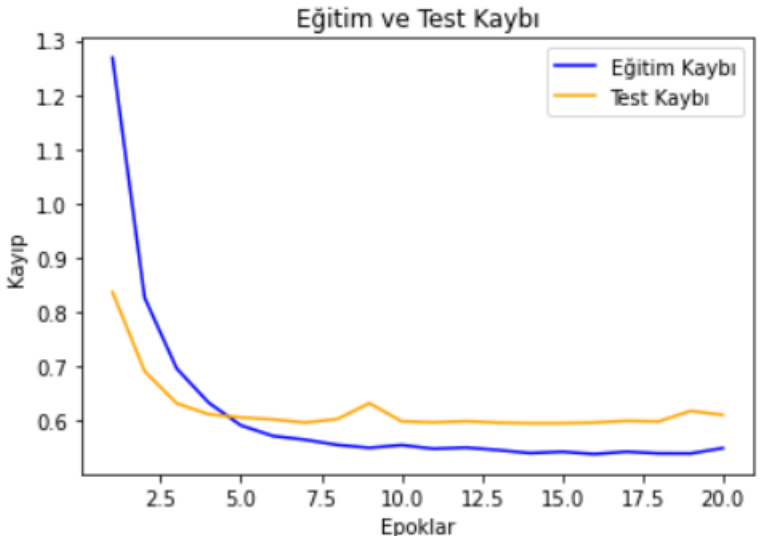
SGD, Adam, Adagrad, RMSprop gibi optimizasyon algoritmalarının hepsi katman sayısı belirlendikten sonra denenmiş ve optimum çalışanı belirlenmeye çalışılmıştır. Yapılan denemeler sonucunda en iyi değerlerin Adam algoritmasında alındığı görülmüştür. Algoritmaya ait öğrenme katsayısı değiştirilerek sistemin optimize edilme işlemi biraz daha genişletilmiştir.

```
tf.keras.optimizers.Adam(
    learning_rate=0.001,
    beta_1=0.9,
    beta_2=0.999,
    epsilon=1e-07,
    amsgrad=False,
    name="Adam",
    **kwargs
)
```

Şekil 3.16. Keras Adam optimizasyon algoritması default değerleri.

Keras yapısında bulunan Adam algoritmasının öğrenme katsayısı 0.001 olarak gözükmektedir. Bu değer arttırılıp azaltılarak sonuca olan etkileri araştırılmıştır. Yapılan çalışmalar sonucu en uygun seviyenin 0.004 olduğu kararına varılmıştır.

Seçilen epoch sayısı da öğrenmenin durumuna göre ayarlanmıştır. 20'den sonra sistemin eğitim için hata değerlerinin düşüş eğilimi göstermemesi ve 10, 15 değerleri de kimi durumlarda yeteri kadar eğitim için uygun zaman yaratmaması sebebiyle epoch için en uygun değer olarak 20 seçilmiştir.



Şekil 3.17. Eğitim verilerinin kayıp-epoch grafiğine ait bir örnek.

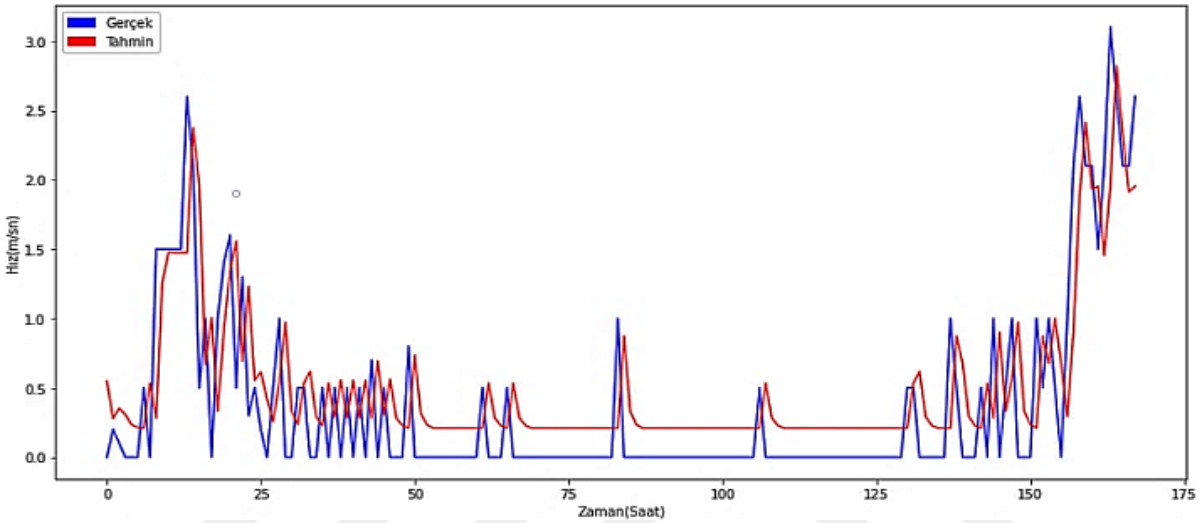
Şekil 3.16'da eğitim değerleri tablosunun içinde test değerlerinin bulunma sebebi; eğitim verilerinin kendi içerisinde de %67-%33 oranında eğitim ve test diye verilerin ayrılmasıdır.

Batch_size modelin belirlenen değer kadar geçmesinden sonra güncellenmesini sağlar. Eğer bu değer çok büyük olursa çıkan hata oranlarında hızlı bir atlama eğer çok küçük olursa sistemin eğitim süresinde uzama meydana gelecektir. Bu sebeple alınacak batch_size değeri bu istenmeyen iki durumu yaşatmayacak bir değerde olmalıdır. Çeşitli denemeler sonucunda en uygun batch_size değeri 128 olarak belirlenmiştir.

4. BULGULAR

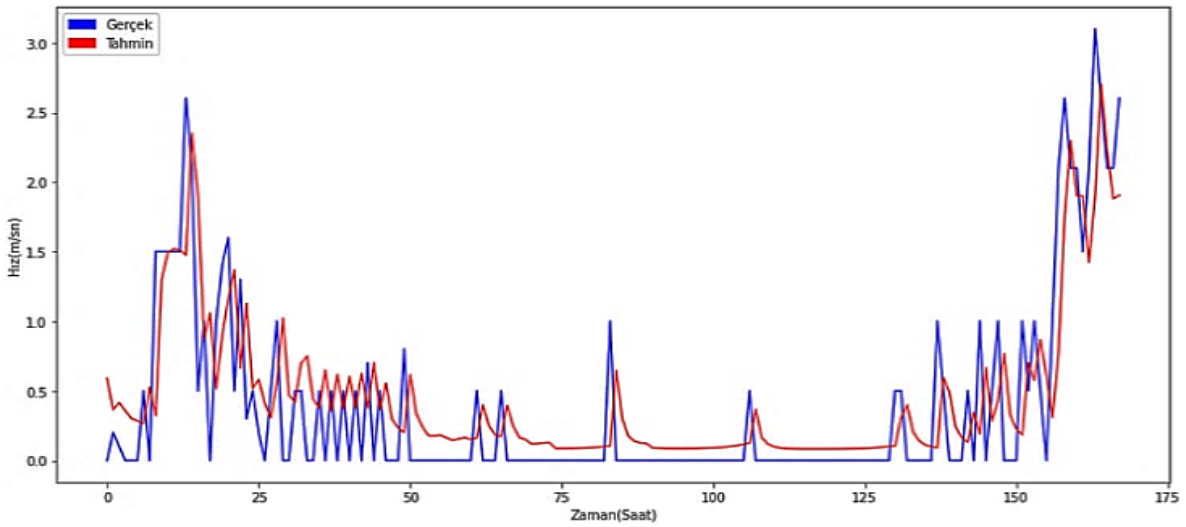
Kullanılan iki farklı LSTM modeli ve çıktı büyüklüklerine göre MSE hata değerleri aşağıdaki grafiklerde verilmiştir.

1 Yıllık (2019) eğitim verileri kullanılarak oluşturlan modelde 2020 yılının Ocak ayı için yapılan tahminlerde en iyi sonucu giriş büyüklüğünü 3 saat olarak alan ve 1 haftalık (168 saatlik) saatlik rüzgar hızını bulan tek değişkenli çift yönlü LSTM modeli yapmıştır. MSE sonucu **0.1989**'dur.



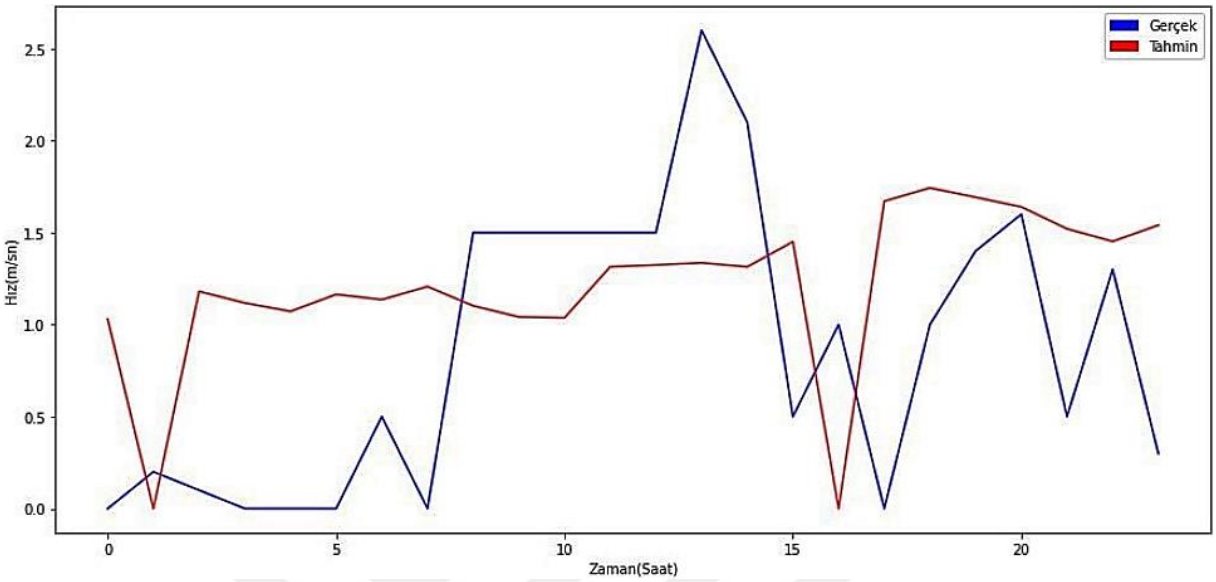
Şekil 4.1. 1 yılık eğitim setli tek değişkenli çift yönlü LSTM en iyi saatlik tahmini.

10 Yıllık (2010-2019) veriler kullanılarak oluşturlan modelde 2020 yılının Ocak ayı için yapılan tahminlerde en iyi sonucu giriş büyüklüğünü 24 saat olarak alan ve 1 haftalık (168 saatlik) saatlik rüzgar hızını bulan tek değişkenli çift yönlü LSTM modeli yapmıştır. MSE sonucu **0.1781**'dir.



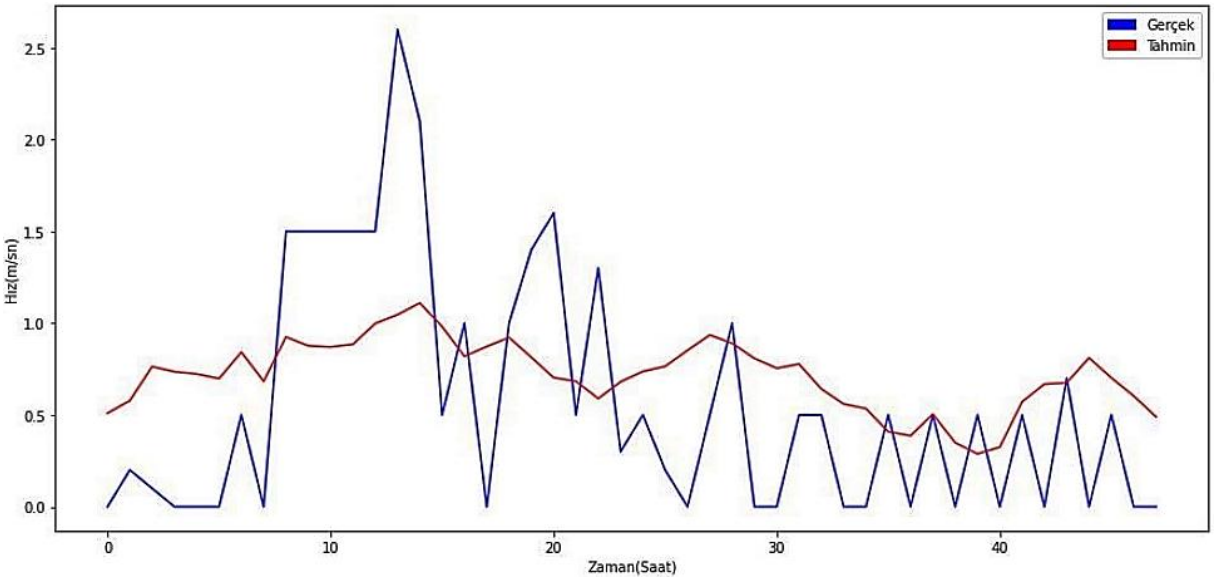
Şekil 4.2. 10 yılık eğitim setli tek değişkenli çift yönlü LSTM en iyi saatlik tahmini.

1 Yıllık veriler kullanılarak oluşturlan modelde 2020 yılının Ocak ayı için yapılan tahminlerde en iyi sonucu giriş büyüklüğünü 744 saat olarak alan ve 1 günlük (24 saatlik) saatlik rüzgar hızını bulan LSTM modeli yapmıştır. MSE sonucu **0.7785**'dir.



Şekil 4.3. 1 yılık eğitim setli çok adımlı vektör çıkışlı LSTM en iyi saatlik tahmini.

10 Yıllık veriler kullanılarak oluşturlan modelde 2020 yılının Ocak ayı için yapılan tahminlerde en iyi sonucu giriş büyüklüğünü 48 saat olarak alan ve 2 günlük (48 saatlik) saatlik rüzgar hızını bulan LSTM modeli yapmıştır. MSE sonucu **0.3414**'dir.



Şekil 4.4. 10 yılık eğitim setli çok adımlı vektör çıkışlı LSTM en iyi saatlik tahmini.

Tüm parametreler ve hata değerlerinin bulunduğu tablo aşağıda verilmiştir.

Tablo 4.1. Tüm hata (MSE) değerleri tablosu.

Eğitim Veri Adet	Giriş Adedi	1 Gün MSE	2 Gün MSE	1 Hafta MSE	1 Ay MSE	LSTM TİP
10 Yıllık Veri	3 Giriş	0.4433	0.3287	0.2298	0.2865	Tek Değişkenli
	24 Giriş	0.3887	0.2890	0.1781	0.2568	
	48 Giriş	0.4105	0.2837	0.1785	0.2525	
	168 Giriş	0.4186	0.2947	0.1865	0.2586	
1 Yıllık Veri	3 Giriş	0.4228	0.3317	0.1989	0.2763	
	24 Giriş	0.4023	0.3072	0.1994	0.2673	
	48 Giriş	0.4189	0.3253	0.2425	0.2597	
	168 Giriş	0.4211	0.3122	0.2005	0.2727	
10 Yıllık Veri	24 Giriş	0.6268	0.9014	1.1830	1.9795	Çok Adımlı
	48 Giriş	0.5839	0.3414	1.3572	1.8884	
	168 Giriş	0.7843	0.4597	0.8149	1.6388	
	744 Giriş	0.4294	0.4541	1.6540	2.6654	
1 Yıllık Veri	24 Giriş	1.9036	1.4501	2.2507	1.9805	
	48 Giriş	1.0557	1.6741	2.4795	2.4896	
	168 Giriş	0.7810	2.0033	2.0569	2.1413	
	744 Giriş	0.7785	1.5555	1.8469	1.8805	

Tabloda her girdi değerine karşılık tahminler için en iyi değerler renkli yazılmıştır.

Genel tablodan elde edilen çıkarımlar şu şekildedir:

- LSTM modelleri ile uzun vadeli (1 Hafta) saatlik tahminler daha doğru çıkarken çok çıktılı vektör tipli LSTM'lerde ise kısa vadeli (1 ve 2 gün) tahminlerde doğruluk oranı artmaktadır.
- Eğitim verisi sayısı fazla oldukça sistem daha doğru sonuçlar vermektedir.
- 1 ve 10 yıllık verilerin mevsimsel verileri alınarak, eğitilmesiyle oluşturulacak bir modelle tahminler yapılıp sonuçlar incelenebilir.

6. SONUÇLAR ve ÖNERİLER

Bu çalışmada Bilecik İli için Bilecik Merkez Meteoroloji Müdrülüğünden alınan 2010-2019 yıllarına ait ham saatlik rüzgar hızı verileri kullanılarak gelecek zaman dilimlerindeki saatlik rüzgar hızı tahminleri derin öğrenme RNN ağının gelişmiş bir modeli olan LSTM modeli kullanılarak yapılmıştır. Çalışmanın hedeflerinden biri derin öğrenme modelinin saatlik rüzgar hızı tahmininde gerçekleyebileceği başarımları ölçmek ve bu yolda yapılacak daha ileri çalışmalarımıza temel oluşturmaktır. Deneysel çalışmalardan elde edilen sonuçlara göre 10 yıllık veriler için oluşturulmuş tek değişkenli çift yönlü LSTM modeli için en iyi tahmin sonucunun 24 girişli 1 haftalık (168 saatlik) tahmin olduğu görülmüştür. Diğer bir hedef ise 1 yıllık veya 10 yıllık verilerden hangisinin Derin Öğrenme modeli için daha uygun olduğunu belirlemektir. LSTM modelinde en iyi sonuçların daha büyük veri kümelerinde daha verimli rüzgar hızı tahminleri olduğu görülmüştür.

Çalışma kapsamında saatlik rügar hızı verileri zaman serisi olarak desen hatırlama özelliğine sahip olan LSTM modelinde 1 yıllık ve 10 yıllık veriler kullanılarak eğitilmiştir. Bu eğitim sonucunda oluşturulan yine karşılaştırma yapılabilmek amacıyla tek değişkenli çift yönlü ve çok adımlı vektör çıkışlı iki farklı LSTM tipinde çeşitli zaman aralıkları için tahminler elde edilmiştir. Bu tahminlerin sonucunda modelin elde ettiği en iyi sonuçlar tezin bulgular bölümünde incelenmiştir.

İleriki çalışmalarda, LSTM modelinin zaman serileri üzerindeki tahmin başarımlarını artırmak için diğer makine öğrenmesi yöntemleriyle beraber hibrid bir modelle yaklaşımlar denenebilir. Sıcaklık, basınç gibi başka parametrelerin de modele dahil edilmesiyle tek parametreli tahminin mi? yoksa çok parametreli tahminin mi? doğruluk oranının daha yüksek olduğu araştırılabilir.

KAYNAKÇA

Altan, A., & Karasu, S. (2020). Ayrıştırma Yöntemlerinin Derin Öğrenme Algoritması ile Tanımlanan Rüzgâr Hızı Tahmin Modeli Başarımına Etkisinin İncelenmesi. *European Journal of Science and Technology*, *20*, 844–853.

Altan, G. (2019). DeepGraphNet: Grafiklerin Sınıflandırılmasında Derin Öğrenme Modelleri. *European Journal of Science and Technology*, 319–327.

Arslan, İ. (2020). *Python ile Veri Bilimi*. Pusula Yayınları. İstanbul.

Ayturan, Y.A. (2019). *Derin Öğrenme İle Havadaki Partikül Madde Konsantrasyonu Tahmini*. *23*(3), 57.

Berk, M. E. (2020). Dijital Çağın Yeni Tehlikesi "Deepfake." *OPUS Uluslararası Toplum Araştırmaları Dergisi*, *16*(28), 1–1.

Bloice, M. D., & Holzinger, A. (2016). A tutorial on machine learning and data science tools with python. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Vol. 9605 LNCS* (pp. 435–480).

Bontempi, G., Ben Taieb, S., & Le Borgne, Y. A. (2013). Machine learning strategies for time series forecasting. *Lecture Notes in Business Information Processing*, *138 LNBIP*, 62–77.

Botchkarev, A. (2019). A new typology design of performance metrics to measure errors in machine learning regression algorithms. *Interdisciplinary Journal of Information, Knowledge, and Management, 14*, 45–76.

Brownlee, J. (2018). *Deep Learning for Time Series*. 557.

Carneiro, T., Da Nobrega, R. V. M., Nepomuceno, T., Bian, G. Bin, De Albuquerque, V. H. C., & Filho, P. P. R. (2018). Performance Analysis of Google Colaboratory as a Tool for Accelerating Deep Learning Applications. *IEEE Access*, *6*, 61677–61685.

Chai, T., & Draxler, R. R. (2014). Root mean square error (RMSE) or mean absolute error (MAE)? -Arguments against avoiding RMSE in the literature. *Geoscientific Model Development*, *7*(3), 1247–1250.

Charan, M. S. (2020). *Python And Its libraries in Data Science and Related*. *December*.

Clark, F. (1994). Zekâ. *Ankara Universitesi Egitim Bilimleri Fakultesi Dergisi*, *5*(3), 001–006.

Çakır, B. & Helvacı, E. & Demir, A. (2016). *Rüzgar Türbini Kanat Tasarımı Ve Analizi*. (Yayınlanmamış Lisans Tezi). Karabük Üniversitesi, Mühendislik Fakültesi, Karabük.

Çıtak, E., & Kılınç Pala, P. B. (2016). Yenilenebilir Enerjinin Enerji Güvenliğine Etkisi. *Süleyman Demirel Üniversitesi Sosyal Bilimler Enstitüsü Dergisi*, *3*(25), 79–102.

Doğan, F., & Türkoğlu, İ. (2019). Derin Öğrenme Modelleri ve Uygulama Alanlarına İlişkin Bir Derleme. *DÜMF Mühendislik Dergisi*, *10*(2), 409–445.

Dokur, E., Kurban, M., & Ceyhan, S. (2016). Wınd Speed Modellıng Usıng Inverse Weıbull Dıstrubıtıon: A Case Study For Bilecik, Turkey. *International Journal of Energy Applications and Technologies*, *3*(2), 55–59.

Elektrik Mühendisleri Odası (EMO) (2021)**.** Türkiye Elektrik İstatistikleri Şubat 2021. [Erişim:18.04.2021, https://www.emo.org.tr/genel/bizden_detay.php?kod=88369]

Elmas, Ç. (2018). *Yapay Zeka Uygulamaları.* Seçkin Yayınları

Emmert-Streib, F., Yang, Z., Feng, H., Tripathi, S., & Dehmer, M. (2020). An Introductory Review of Deep Learning for Prediction Models With Big Data. *Frontiers in Artificial Intelligence*, *3*, 4.

Fischer, A., & Igel, C. (2012). An introduction to restricted Boltzmann machines. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *7441 LNCS*, 14–36.

Goodfellow, I., Bengio, Y., & Courville, A. (2018). *Deep learning*. (Çev.) Fatoş Yarman Vural, (Çev.) Ramazan Gökberk Cinbiş, (Çev.) Sinan Kalkan, Buzdağı Yayıncılık, İstanbul.

Güç, R. & Ceyhan, S. (2016). *Bilecik İli İçin Güneş Enerjisi Analizi Ve Yapay Sinir Ağları İle Hava Sıcaklığı Tahmini*. (Yüksek Lisans Tezi) Bilecik Şeyh Edebali Üniversitesi, Fen Bilimleri Enstitüsü. Bilecik.

Güdelek, M. U., & Özbayoğlu, A. M. (2013). Zaman Serisi Analizi Ve Tahmin: Derin Öğrenme Yaklaşımı. *Journal of Chemical Information and Modeling*, *53*(9), 1689–1699.

Gündüz, G., & Cedimoğlu, İ. H. (2019). Derin Öğrenme Algoritmalarını Kullanarak Görüntüden Cinsiyet Tahmini. *Sakarya University Journal of Computer and Information Sciences*, *2*(1), 9–17.

Halterman, R. L. (2018). *Fundamental of Python Programming*. *1*(1), 669.

Harari, N. H. (2012). *Sapiens: İnsan Türünün Kısa Bir Tarihi*, (Çev.) Ertuğrul Genç, Kolektif Yayınları, İstanbul.

Hota, H. S., Handa, R., & Shrivas, A. K. (2017). Time Series Data Prediction Using Sliding Window Based RBF Neural Network. *International Journal of Computational Intelligence Research, 13*(5).

İlkılıç, Z. (2016). Türkiye'de Rüzgar Enerjisi ve Rüzgar Enerji Sistemlerinin Gelişimi. *Batman Üniversitesi Yaşam Bilimleri Dergisi*, *6*(2/2), 1–13.

İnançlı, S., & İnal, V. (2018). Türkiye' De Alternatif Enerji Üretimi İle Ekonomik Büyüme Arasındaki İlişkinin Saklı Eşbütünleşme Testi İle Analizi. *Kastamonu Üniversitesi İktisadi ve İdari Bilimler Fakültesi Dergisi*, *20*(4), 102–116.

İnik, Ö., & Ülker, E. (2017). Derin öğrenme ve görüntü analizinde kullanılan derin öğrenme modelleri deep learning and deep learning models used in image analysis. *Gaziosmanpasa Journal of Scientific Research*, *ISSN*, 85–104.

Kalkavan, Y. (2020)**.** Lstm Derin Öğrenme Yöntemi İle Arıma Zaman Seri Leri Yönteminin Araç Kiralama Sektör Verileri Aracılığıyla Karşılaş Tırılması. *International Marmara Science Congres*, s. 244-250.

Karakuş, S. & Kaya, M. (2019). *Derin öğrenme yöntemleri kullanarak dijital deliller üzerinde adli bilişim incelemesi*. (Yüksek Lisans Tezi). Fırat Üniversitesi, Fen Bilimleri Enstitüsü, Elazığ.

Karasulu, B. (2018). Kısıtlanmış Boltzmann Makinesi Ve Farklı Sınıflandırıcılarla Oluşturulan Sınıflandırma İş Hatlarının Başarımının Değerlendirilmesi. *Bilişim Teknolojileri Dergisi*, *11*(3), 223–233.

Kaya, K., & Koç, E. (2015). Enerji Kaynakları -Yenilenebilir Enerji Durumu. *Mühendis ve Makina*, *56*(668), 36–47.

Khouloud, Z., Salim, G., & Saber, R. M. (2021). Hourly Wind Speed Forecasting Using FFT-Encoder-Decoder-LSTM in South West of Algeria (Adrar). In *International Journal of Informatics and Applied Mathematics* (Vol. 4, Issue 1).

Kızrak, M. A., & Bolat, B. (2018). Derin Öğrenme ile Kalabalık Analizi Üzerine Detaylı Bir Araştırma. *Bilişim Teknolojileri Dergisi*, *11*(3), 263–286.

Kumar, A., Gangwar, S., & Bali, V. (2019). Deep Learning based Wind Speed Forecasting-A Review. *2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, 426–431.

Kurt, F., & Efe, M. Ö. (2018). *Evrişimli Sinir Ağlarında Hiper Parametrelerin Etkisinin İncelenmesi Analysıs Of The Effects Of Hyperparameters In Convolutıonal Neural Networks*.(Yayımlanmış Yüksek Lisans Tezi). Hacattepe Üniversitesi, Fen Bilimleri Enstitüsü, Ankara, s. 144

Lecun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. In *Nature* (Vol. 521, Issue 7553, pp. 436–444). Nature Publishing Group.

Liu, X., Zhang, H., Kong, X., & Lee, K. Y. (2020). Neurocomputing Wind speed forecasting using deep neural network with feature selection. *Neurocomputing*, *397*, 393–403.

McKinney, W. (2011). pandas: a Foundational Python Library for Data Analysis and Statistics. *Python for High Performance and Scientific Computing*, 1–9.

Mckinney, W. (2017)**.** *Python for Data Analysis*. O'Reilly Medya

Montúfar, G. (2018). Restricted boltzmann machines: Introduction and review. *Springer Proceedings in Mathematics and Statistics*, *252*(October), 75–115.

Mosavi, A., Ardabili, S., & Várkonyi-Kóczy, A. R. (2020). List of Deep Learning Models. In *Lecture Notes in Networks and Systems* (Vol. 101, pp. 202–214).

Nabiyev, N. N. (2016). *Yapay Zeka.* Seçkin Yayınları.

Naser, M. Z., & Alavi, A. (2020). *Insights into Performance Fitness and Error Metrics for Machine Learning*.

Ozgur, C., Colliau, T., Rogers, G., Hughes, Z., & Myer-Tyson, E. B. (2017). MatLab vs. Python vs. R. *Journal of Data Science*, *15*, 355–372.

Öztemel, E. (2012). *Yapay Sinir Ağları* (Ç. Rıfat (ed.); 3rd ed.). Papatya Yayıncılık.

Perl, L., General, G. N. U., License, P., Programming, C., Point, T., Point, T., & Point, T. (n.d.). *Python Programming Language*.

Pirim, H. (2006). YAPAY ZEKA. *Journal of Yaşar University*, *1*(1), 81–93.

Ser, G., & Bati, C. T. (2019). Determining the best model with deep neural networks: Keras application on mushroom data. *Yuzuncu Yil University Journal of Agricultural Sciences*, *29*(3), 406–417.

Sergio, A. T., & Ludermir, T. B. (2015). *Deep Learning for Wind Speed Forecasting in Northeastern Region of Brazil*. *Dl*, 322–327.

Sezer, O. B., Gudelek, M. U., & Ozbayoglu, A. M. (2020). Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. *Applied Soft Computing Journal*, *90*.

Shrestha, A., Mahmood, A., & Member, S. (2019). Review of Deep Learning Algorithms and Architectures. *IEEE Access*, *PP*(c), 1.

Sørensen, B. (1991). A history of renewable energy technology. *Energy Policy*, *19*(1), 8–12.

Soydaner, D. (2020). A Comparison of Optimization Algorithms for Deep Learning. *International Journal of Pattern Recognition and Artificial Intelligence*, *34*(13).

Sun, S., Cao, Z., Zhu, H., & Zhao, J. (2020). A Survey of Optimization Methods from a Machine Learning Perspective. *IEEE Transactions on Cybernetics*, *50*(8), 3668–3681.

Szandała, T. (2021). Review and comparison of commonly used activation functions for deep neural networks. *Studies in Computational Intelligence*, *903*, 203–224.

Şeker, A., Diri, B., & Balık, H. (2017). Derin Öğrenme Yöntemleri ve Uygulamaları Hakkında Bir İnceleme. *Gazi Journal of Engineering Sciences*, *3*(3), 47–64.

Tan, Z., & Aydın, G. (2019). *Derin Öğrenme Yardımıyla Araç Sınıflandırma*. (Yayımlanmış Yüksek Lisans Tezi). Fırat Üniversitesi, Fen Bilimleri Enstitüsü, Elazığ.

Toğaçar, M., & Ergen, B. (2019). *Biyomedi kal Görüntülerde Derin Öğrenme ile Mevcut Yöntemlerin Kıyaslanması Comparison of Deep Learning and Existing Methods in Biomedical Imagery*. *31*(1), 109–121.

Tüfekçi, M., & Karpat, D. F. (2019). Derin Öğrenme Mimarilerinden Konvolüsyonel Sinir Ağları (CNN) Üzerinde Görüntü İşleme - Sınıflandırma Kabiliyetininin Arttırılmasına Yönelik Yapılan Çalışmaların İncelenmesi A Review for Investigation Studies That are Done for Improving Image Processing. *International Conference on Human-Computer Interaction, Optimization and Robotic Applications.* 28–31.

Uğuz, S. (2019). Makine Öğrenmesi Teorik Yönleri ve Pyhton Uygulamaları ile Bir Yapay Zeka Ekolü. Nobel Yayınları.

Uluslararası Enerji Ajansı (International Energy Agency) (2020). 1990 - 2018 Yılları Arası Enerji Kaynaklarına Göre Elektrik Üretimi Kapasite İlaveleri.[Erişim:18.04.2021,

https://www.iea.org/data-and-statistics].

Yeşilnacar, Y. O., & Gün, A. (2011). *Bilecik İlinin Yapay Sinir Ağları İle Rüzgar Hızı, Basınç Ve Sıcaklık Tahmini*. (Yayınlanmamış Yüksek Lisans Tezi). Bilecik Şeyh Edebali Üniversitesi, Fen Bilimleri Enstitüsü, Bilecik.

Yılmaz, A. (2020). *Yapay Zeka.* İnkilap Yayınları.