T.C.

AKDENİZ ÜNİVERSİTESİ

EL YAZISIYLA YAZILAN RAKAMLARIN ALGILANMASINDA FACEBOOK PROPHET FONKSİYONU ETKİSİNİN ARAŞTIRILMASI

Çağdaş KAPLAN

FEN BİLİMLERİ ENSTİTÜSÜ

ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ

ANABİLİM DALI

YÜKSEK LİSANS TEZİ

Şubat 2023

ANTALYA

T.C.

AKDENİZ ÜNİVERSİTESİ

EL YAZISIYLA YAZILAN RAKAMLARIN ALGILANMASINDA FACEBOOK PROPHET FONKSİYONU ETKİSİNİN ARAŞTIRILMASI

Çağdaş KAPLAN

FEN BİLİMLERİ ENSTİTÜSÜ

ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ

ANABİLİM DALI

YÜKSEK LİSANS TEZİ

Şubat 2023

ANTALYA

T.C.
AKDENİZ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

EL YAZISIYLA YAZILAN RAKAMLARIN ALGILANMASINDA FACEBOOK PROPHET FONKSİYONU ETKİSİNİN ARAŞTIRILMASI

Çağdaş KAPLAN

ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ

ANABİLİM DALI

YÜKSEK LİSANS TEZİ

Şubat 2023

ANTALYA

Evrak Tarih ve Sayısı: 10.02.2023-295

T.C.

AKDENİZ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

EL YAZISIYLA YAZILAN RAKAMLARIN ALGILANMASINDA FACEBOOK PROPHET FONKSİYONU ETKİSİNİN ARAŞTIRILMASI

Çağdaş KAPLAN

ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ

ANABİLİM DALI

YÜKSEK LİSANS TEZİ

Bu tez .../.../2023 tarihinde jüri tarafından Oybirliği / Oyçokluğu ile kabul edilmiştir.

Doç. Dr. Süleyman BİLGİN (Danışman)

Prof. Dr. Ömer Halil ÇOLAK

Dr. Öğr. Üyesi Ufuk ÖZKAYA

Bu belge, güvenli elektronik İmza ile imzalanmıştır.
Evrak sorgulaması https://turkiye.gov.tr/ebd?eK=5543&eD=BS9FP2N173&eS=295 adresinden yapılabilir. (PIN:45232)

ÖZET

EL YAZISIYLA YAZILAN RAKAMLARIN ALGILANMASINDA FACEBOOK PROPHET FONKSİYONU ETKİSİNİN ARAŞTIRILMASI

Çağdaş KAPLAN

Yüksek Lisans Tezi, Elektrik-Elektronik Mühendisliği Anabilim Dalı

Danışman: Doç. Dr. Süleyman BİLGİN

Şubat 2023; 57 sayfa

Yapay zekâ, çağımızda sürekli gelişen teknolojiyle birlikte tıp, mühendislik ve endüstri 4.0 uygulamalarında sıklıkla kullanılmaktadır. Aslında yapay zekâ teknolojilerinin temelini irdelersek derin öğrenme örüntü tanıma modellerinin yer aldığını görürüz. Özellikle bu disiplinin ilk yıllarında 1989 yılında Yann LeCUN el yazısıyla yazılan rakamların tanınması ile ilgili MNIST veri setiyle Evrişimsel Yapay Sinir kullanarak çığır açan bir çalışma yapmış ve örüntü tanımaya başka bir bakış açısı getirmiştir

Bu çalışmada, klasik derin öğrenme ve ESA ile tahmin edilen MNIST veri seti ağırlıkları derinlemesine incelenerek günümüzde etkin olarak iktisadi ve matematiksel tahmin analizlerinde kullanılan zaman seri fonksiyonlarından Facebook Prophet fonksiyonu yardımıyla derin öğrenme modellerindeki ağırlıkları, zaman serisi ile tekrar tahmin edilerek klasik ESA ile bir araştırma çerçevesinde karşılaştırma amaçlanmıştır. Bu karşılaştırma ile aslında klasik yapay zekâ ve derin öğrenme modellerinin ağırlıklarının eğitim girdisi olarak alınıp zaman serisi ile tahmin edilerek performans karşılaştırılması yapılmasıyla derin öğrenme disiplinlerine farklı bir bakış açışı getirecektir.

Sonuç olarak ESA modelinin tahmin performansının Facebook Prophet fonksiyonuna göre daha yüksek olduğu ağırlıklarının tahminin zor olduğu ağırlıklar bu görüntü verisi içerisindeki fark edilebilir ağırlıkları veya bunların gösterimi belirlemek için oldukça kolaylaştırıcı bir tekniğin temelini oluşturabilir. Aslında bu ağırlıklar ESA modelinin veya derin öğrenme modelinin tahmin edilme zorlukları açısından zayıflıkları olarak görülebilir. Prophet ve zaman serisi tahmin yöntemleri kullanılarak bahsedilen verilere veya bulunduğu katmanın o bölgesine veri zehirleme işlemi yapılarak nesne ve görüntünün tahmin edilmesi kolaylaştırılabilir.

ANAHTARKELİMELER: Bilgisayar Görüsü, Derin Öğrenme, Facebook Prophet, MNIST, Örüntü Tanıma, Veri Zehirlemesi, Yapay Zeka, Zaman Serileri

JÜRİ: Doç. Dr. Süleyman BİLGİN

Prof. Dr. Ömer Halil ÇOLAK

Dr. Öğr. Üyesi Ufuk ÖZKAYA

ABSTRACT

RESEARCH OF FACEBOOK PROPHET FUNCTION EFFECT FOR HANDWRITTEN DIGIT PREDICTION

Çağdaş KAPLAN

MSc Thesis in Electrical and Electronics Engineering

Supervisor: Assoc. Prof. Dr. Süleyman BİLGİN

February 2023; 57 pages

Artificial intelligence technologies are being used in medicine, engineering and industry 4.0 applications with the development of today's technologies. We can see that deep learning and pattern recognition models creates artificial intelligence science if we get the heart of this discipline. Especially in 1989 Yann LeCUN achieved a significant breakthrough in pattern recognition from a different view point with recognizing handwritten digits using Convolutional Neural Network(CNN) on MNIST dataset.

In this study, it is aimed that all predicted weights of classical deep learning models on MNIST datasets are re-predicted with Facebook Prophet function which is effectively used in economic forecasts and mathematical prediction analysis and compared with classical Convolutional Neural Network weight values.Indeed this comparison which uses classical CNN model weights as training input and predictes with time serie model Prophet and compares prediction performances that will give us a different viewpoint in this discipline.

As a result more classical CNN model predictions which has more powerful performances that Facebook Prophet function in layer and kernel numbers as are more difficult predict than other weights. It will be developed a new technic for specifying disentaglend representation with this weights.These weights can be seen as weakness of CNN or deep learning models. Finally with applying data poisoning to this weights also can be used as more easier method for predicting objects or images using time series as Prophet function.

KEYWORDS: Artificial Intelligence, Computer Vision, Data Poisoning, Deep Learning, Facebook Prophet, MNIST, Pattern Recognition, Time Series,

COMMITTEE: Assoc. Prof. Dr. Süleyman BİLGİN

Prof. Dr. Ömer Halil Çolak

Assistant Professor Ufuk ÖZKAYA

ÖNSÖZ

İlgili tez konusunun belirlenmesi, sonuçlanması gibi tüm süreçlerde bilgisi ve engin deneyimleriyle bana güç katan ve yolumu aydınlatan, yardımlarını ve değerli zamanını esirgemeyen danışman hocam Sayın Doç. Dr. Süleyman BİLGİN'e sonsuz teşekkürlerimi ve ana bilim dalımızdaki görevli tüm değerli akademisyen hocalarımıza saygılarımı sunarım.

Tez çalışmasının, sorgulama yöntemlerinin belirlenmesinde ve yöntem aşamasında desteklerini esirgemeyen, bilgileriyle yolumuzu aydınlatan ve değerli vakitlerini esirgemeyen değerli arkadaşım Sayın Doç. Dr. Cüneyt Gürcan AKÇORA'ya sonsuz teşekkürlerimi ve saygılarımı sunarım.

Maddi ve manevi destekleriyle yanımda olmasalar da daima yanımda hissettiğim sevgili eşime, kızıma ve aileme, sınıf arkadaşlarıma ve işyerinde değerli müdürümüz Sn. Şems TUĞLU'ya çok teşekkürlerimi bir borç bilirim.

İÇİNDEKİLER

AKADEMİK BEYAN

Yüksek Lisans Tezi olarak sunduğum "El Yazısıyla Yazılan Rakamların Algılanmasında Facebook Prophet Fonksiyonu Etkisinin Araştırılması" adlı bu çalışmanın, akademik kurallar ve etik değerlere uygun olarak yazıldığını belirtir, bu tez çalışmasında bana ait olmayan tüm bilgilerin kaynağını gösterdiğimi beyan ederim.

06/02/2023

Çağdaş KAPLAN

SİMGELER VE KISALTMALAR

Simgeler

A_j : j durumundaki gerçekleşen değeri

ε_t : Prophet fonksiyonunun hata değeri

P_j : j durumunda tahmin edilen değeri

$\tan h(x)$: Hiperbolik Tanjant Fonksiyonu

X_i : i durumundaki veri (standart sapma hesaplamasında)

$y^{(i)}$: En iyileme işleminde etiket, çıkış değeri

e_j : j durumundaki hata değeri

s_j : Softmax Fonksiyonunun j durumundaki değeri

$w_0, w_1 \dots w_i$: Ağırlık Değerleri

$x^{(i)}$: En iyileme işleminde eğitim örneklemi

$x_0, x_1 .. x_i$: Giriş Değerleri

y_t : Tahmin edilen değer (Zaman Serisi Çıktısı)

$\nabla_\theta J\left(\theta_j; x^{(i)}, y^{(i)}\right)$: Zıt yönde dereceli değişen(azalan) maliyet fonksiyonu

σ^2 : Varyans

σ_{xy} : Kovaryans

$*$: Evrişim işlemi

Ç : Zaman Serisi Çevresel Faktörler Bileşeni

d : Fark alma işlemi sayısı

D : Mevsimsel Fark alma işlemi sayısı

h(t) : Evrişim işleminde birim dürtü sinyali

h[n] : Evrişim işleminde ayrık birim dürtü sinyali

$h(t)$: Prophet fonksiyonunun tatil/etkinlik işlevi

P : Mevsimsel Oto regresyon modelinin derecesi

p : Oto regresyon(AR) modelinin derecesi

q : Hareketli ortalama (MA) modelinin derecesi

Q : Mevsimsel Hareketli ortalama (MA) modelinin derecesi

s : Periyot

x(t) : Evrişim işleminde giriş sinyali

y(t) : Evrişim işleminde çıkış sinyali

α : Düzeltme katsayısı(Otoregresif Hareketli Ortalama Modeli)

β : Trend Düzeltme katsayısı (Otoregresif Hareketli Ortalama Modeli)

$C(t)$: Büyüme işlevi zaman bağlı taşıma kapasitesi

$CCE(p,t)$: Aralıklı Çapraz Entropi

M : Zaman Serisi Mevsimsel Dalgalanma Bileşeni

R : Zaman Serisi Rassal Faktörler Bileşeni

T : Zaman Serisi Trend Bileşeni

$a \epsilon f(x) = ax$: Sızıntı doğrultulmuş doğrusal birim fonksiyonu eğimi

b : Bias değeri

f : Aktivasyon işlevi

$g(t)$: Prophet fonksiyonunun büyüme işlevi

n : öğrenci sayısı (standart sapma hesaplamasında)

$s(t)$: Prophet fonksiyonunun mevsimsellik işlevi

x[n] : Evrişim işleminde ayrık giriş sinyali

$x[i,j]$: i ve j durumlarındaki giriş sinyali

y[a, b] : Ayrık Zamanda iki sinyalin süperpozisyon değeri

y[n] : Evrişim işleminde ayrık çıkış sinyali

$y(t)$: Prophet fonksiyonunun tahmin değeri

η : (lokal minimum için adım sayısını belirleyen) öğrenme oranı

θ : En iyileme işleminde sürekli güncellenen model parametresi)

μ : İlgili verinin aritmetik ortalaması (standart sapma hesaplamasında)

σ : Standart Sapma

$\sigma(x)$: Sigmoid fonksiyonu

Kısaltmalar

BOHOM : Bütünleşik Otoregresif Hareketli Ortalama Model

BOHODDBM : Bütünleşik Otoregresif Hareketli Ortalama Dışsal Değişken Bağımlı Model

MBA : Microsoft Bilişsel Aracı

CWI : Centrum Wiskunde & Informatica

DTW : Dinamik Zaman Bükme -Dynamic Time Warping

ESA : Evrişimsel Sinir Ağı

ÇDZSSBM : Çok Değişkenli Zaman Serileri Saklı Birim Lojistik Modeli

KOKD : Kök Ortalama Karesel Değişim

KOKH : Kök Ortalama Karesel Hata

OMYH : Ortalama Mutlak Yüzde Hata

MNIST : Modified National Institute of Standards and Technology database

KOH : Karesel Ortalama Hata

DDB : Doğrultulmuş Doğrusal Birim

KOKH : Kök Ortalama Karesel Hata

MBOHOM : Mevsimsel Bütünleşik Otoregresif Hareketli Ortalama Model

MBOHODDM : Mevsimsel Bütünleşik Otoregresif Hareketli Ortalama Dışsal Değişken Bağımlı Model

ODA : Olasılıksal Dereceli Azalma

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

1. GİRİŞ

Yeni nesil teknolojilerde artık sıklıkla dile getirilen ve kullanılan yapay zekâ algoritmalarının gelecekte günlük yaşantımızın vazgeçilmezleri arasında yer alacağı rahatlıkla tezahür edilebilir. Özellikle insanın oluşturmuş olduğu yazma, konuşma günlük hareketleri, doğal hareketleri tespit ve tanıma teknolojileri bilgisayar görüsü biliminde daha da ön plana çıkacaktır. Tüm bunların temelinde yatan örüntü tanımada kullanılan evrişimsel sinir ağlarının ağırlıklarının tahmini ve bu tahminlerin farklı bir metotla

- Facebook Prophet,
- BOHODD,
- MBOHODDM

gibi zaman serisi tahmin fonksiyonları ile yapılabilir.

MNIST veri seti ile el yazısıyla yazılan rakamların tanınmasında kullanılan çok klasik ama çok kullanışlı veri setlerinden biridir.Derin öğrenme ile ESA modeli kullanılarak tanınan rakamların tahmin yöntemi şu anda yapay zekâ alanında en revaçta yöntemlerden biridir. Bu bağlamda el yazısıyla yazılan rakamların tanınması aslında bilgisayar görüsü disiplinin temelini oluşturuyor.

Otonom sistemlerde yapay zekâ disiplininde yer alan bilgisayar görüsü biliminde rakamları, resimleri trafik levhaları vb. nesneleri tanımada sıklıkla kullanıyoruz. Bahsedilen bu anıma işlemi sırasında kullanılan matematiksel ağırlık tahmin yöntemleri derin öğrenmede kullanılan ESA modelinde kullanılan çok bilinen dinamik türevsel ve çok işlevsel metotlardır. Tüm bu alışılagelmiş durumlara farklı bir bakış açısıyla ağırlıkların tahmin metotlarını zaman boyutunda irdeleyip verimli bir karşılaştırma yapmak çalışmanın ana konusunu oluşturmaktadır.

Tez çalışmasının bütününde, el yazısı rakamlarından oluşan MNIST veri seti yüklenmiş bir ESA modeli tabi tutularak bu ağdaki ağırlıkların tümü ve değişimi detaylı bir şekilde değinilecek olan python kodlarıyla kaydedilmiştir.

Yapay zekâ bilgisayar görüsü ve görüntü tahmin modellerinde ağırlık tahmininde zaman serisi tahmin yöntemini ilk kez uygulayacağımız çalışmamızda finansal tahmin modellemelerinde günümüzde halen en doğru tahmin işlemini yapan Facebook Prophet fonksiyonu kullanılmıştır. ESA modelinde ağırlıkların tahmini doğruluğu düşük olan ağırlıklara odaklanıp bu ağırlıkları zaman serisi yöntemiyle tahmin edip veride ayırt edici birkaç öznitelik yakalamak için faydalı bir algoritma gelişimine temel oluşturacak bir çalışma ortaya koyulmuştur.

Derin öğrenme modellerinin ağırlıkların tahmininin zaman serileriyle kayan pencere modelini kullanarak tahmin edeceğimiz ağırlıkların bulunduğu veri bölgeleriyle sınıflandırma problemlerinde veri ayırt edilebilirlik seviyesi arttırılması sağlanabilir.

Tez çalışması ana hatlarıyla 5 temel bölümden oluşturulmuştur. Birinci bölümde geniş perspektifte derinlemesine bilgilendirme yapılmış tezin öneminden, içeriğinden ve amacından bahsedilmiştir. İkinci bölümde, literatürde yer alan ve tez konusunu destekleyen benzer çalışmalardan bahsedilmiştir. Üçüncü bölümde ilgili tez çalışmasının metodunun yanı sıra derin öğrenme, çalışma ile alakalı ESA modeli, zaman serisi ve Prophet fonksiyonunun temel bilgileriyle birlikte matematiksel bilgilerine yer verilmiştir. Ayrıca bu bölümde ana çalışmanın tüm python kodları derinlemesine incelenmiştir. Dördüncü bölümde görsel ve matematiksel sonuçlar analiz edilip değerlendirilmiş ve yorumlanmıştır. Beşinci bölümde ise elde edilen analiz sonuçları ve veri zehirlemesi vb. sonraki çalışmalarda getirilebilecek yenilikler ifade edilmiştir.

2. KAYNAK TARAMASI

2.1. Literatür Özeti

Yapay zekâ disiplininde bilgisayar görüsü ve örüntü tanıma (el yazısı tanıma vs.) alanlarında ağırlıkların tahmini işlemini gerçekleştirmek için çeşitli zaman serileri tahmin yöntemlerinin kullanıldığı, karşılaştırmalar yapıldığı ve incelendiği araştırmaların yer aldığı çok fazla yayın olmamakla birlikte yapılmış çalışmalar mevcuttur. Bu doğrultuda literatür taramasında güncel ulusal çalışmalar bulunmamakla uluslararası hibrit ve benzeri gerçekleştirilen çalışmalara değinilmiştir.

Genelde literatürde yapılan çalışmalar zaman serisi öngörülerinin yapay zekâ teknikleri ile tahmin edilmesi üzerinedir. Ancak yaptığımız bu çalışma literatürde yapılan çalışmaları değişik bir yöne çevirerek Derin öğrenme tekniklerinde (örneğin: ESA ağırlıklarının zaman serisi ile tahmini) yapılan tahminlerin zaman seri fonksiyonları ile yapılmasıdır. Bu yüzden literatürde yaptığımız çalışmaya veya buna yakın örnek bulmak güçtür. Çalışmaların bazılarında da sınıflandırma problemlerinde kullanılan derin öğrenmesi tekniklerinin zaman serisi öngörü fonksiyonlarıyla ile birleştirip sınıflandırma doğruluğu arttırılmaya çalışılmıştır.

Yapay zekâ ve derin öğrenmede zaman serisi tahminlerinin birleştirilmiş olduğu çalışmalardan yakın konularda birkaçına değineceğiz. G. Peter Zhang 2003 yılında yapmış olduğu çalışmada otoregresif hareketli ortalamalar BOHOM ve hibrit bir yapay sinir ağı sistemin zaman serisi tahminlerinde kullanıldığı bir model kullanmıştır. Yapay Sinir ağları disiplininde sonuçta elde edilen doğruluğu arttırabilmek amacıyla genelde yapay sinir ağlarıyla yine yapay sinir ağları kombinasyonu yapılıyor. Ancak G. Peter Zhang yapmış olduğu çalışmada klasik lineer yöntemlerin de kombinasyona dahil edilerek daha etkili olacağını düşünmüştür. Literatürdeki çalışmalarda da belirtildiği üzere hibrit modelin sonuçlarının genelleştirilmiş varyans ve hatasının daha düşük olduğu görülmüştür. Bunlara ek olarak veri üzerinde olası stabil olmayan veya değişen örüntüler sebebiyle zaman serilerinde ve istatiksel anlamda modelin belirsizliğini azaltacaktır. BOHOM model yöntemi ile hiper parametrelerle oynayarak ve düzgün uydurma teknikleri ile yapay sinir ağlarındaki aşırı öğrenme problemi daha kolaylaştırılmış hale gelecektir. Sonuçlar gösteriyor ki kısa dönem bir aylık tahminlerde hibrit model ve yapay sinir ağları tahminlerindeki doğruluk rastgele yürüyüş modeli tahminlerinden daha fazladır. [1]

Mahlagha Afrasiabi ve çalışma arkadaşları tarafından 2019 yılında yapılmış olan çalışmada dinamik zaman bükme metodu yardımıyla videolarda ESA öznitelik çıkarımı kullanılarak zaman serisi tabanıyla insan etkileşimlerinin tahmin edilmesini içeren araştırmaya değinilmiştir. İlgili araştırmada kullanılan veri seti BIT-etkileşim veri seti, UT-etkileşim veri seti, TV-insan etkileşim veri setidir. Dinamik zaman bükme algoritması zaman serileri arasındaki benzerliği tespit etmek için oldukça verimli başarılı bir yöntemdir. Bu yöntemle -DTW- farklı çerçeve boyutlarında eğitim ve test imkânı sağlamıştır. Sonuçlara göre BIT-etkileşim veri seti, UT-etkileşim veri seti, TV-insan etkileşim veri setlerinde uygulanan metot iyi sonuçlar vermiştir. Öyle ki çalışmada alınan diğer 6 metoda karşılık istatistiksel olarak kayda değer bir doğruluk elde edilmiştir. [2]

Literatürde yer alan diğer çalışma ise, Laurens van der Maaten ve ekip arkadaşlarıyla birlikte yapmış olduğu 2018 yılında yayınlanan çok değişkenli zaman serileri saklı birim lojistik modelini geliştirmişlerdir. Bu yayında verilerdeki gizli yapıları modellemek ikili stokastik (rastgele) gizli birimler kullanılmıştır. Veri içindeki geçici bağımlılıkları modelleyen saklı birimler zincir yapısındadır. Önceki zaman serisi sınıflandırma modelleriyle karşılaştırıldığında model çok karmaşık karar sınırlarında gizli yapı durumları saklı birimle birlikte üstel olarak arttığı için daha kolay modelleme yapabilmektedir. Model, yüksek performansını gösterebilmek için çeşitli bilgisayar görüsü disiplininde yer alan el yazısı karakteri tanıma, konuşma ve ses tanıma-algılama, yüz ifadesi tanıma, harekete tanıma gibi görevler üzerinde denenmiştir. Çeşitli gerçek veri setlerinde yapılan deneylerde de gözlendiği gibi saklı birim lojistik modeli hareket ünitelerine uyarlandığı durumlarda ortalama %91 doğruluk %93 eğri altın kalan değeriyle çalışmanın yapıldığı zamanda çok iyi bir sonuç elde edilmiştir. [3]

Zaman serilerinin sınıflandırma probleminde kullanıldığı diğer bir benzer çalışma ise Jeffrey F. Cohn ve çalışma arkadaşlarının yaptığı duygusal ifadelerin zaman serisi kernel ile sınıflandırılması başlıklı çalışmadır. Mekân-zamansal çerçevede yüz ifadelerinin tahminin eğer hareketler 3 boyutlu uzayda ise kernel metotlarından yararlanılabilir. Çalışmada dinamik zaman bükme benzerlik ölçümlerinden elde edilen kernel verileri ile birlikte destek vektör sınıflandırmasını uygulanmıştır. Böylelikle sadece şekilde parametreleri ve hareket örüntüsü kullanılarak %99'un üzerinde bir doğruluk elde edilmiştir. Bütün hareket örüntülerinin sınıflandırılmasının yanı sıra birçok yüz ifadesi de en az 5-6 çerçeve içerisinde 200 milisaniyede %90'ın üzerinde doğrulukta tanıma sağlanmıştır. [4]

Literatürde benzer konularda ele alınan çalışmalarda da görüldüğü üzere nesne tanıma yüz ifadesi tanıma gibi problemlerde zaman serileri ve dinamik zaman bükme yöntemi kullanılarak sınıflandırma yapılmıştır. Ancak herhangi bir derin öğrenme veya ESA'nın ağırlık değişimleri birer birer incelenmemiştir. Yapılan bu çalışmada bu ağırlık değişimleri ayrı bir zaman seri tahmin yöntemiyle tekrar tahmin edilmiş olup sonuçlar karşılaştırılmıştır.

2.1.1. Yapay Zekâ ve Derin Öğrenme

Yapay zekâ ve derin öğrenme konuları günümüzde revaçta konular arasında yer almaktadır. Günlük yaşantımızda sıklıkta duyduğumuz 'Yapay Zekâ' ile ilgili temel bilgileri ve bu disiplinlerin altındaki en önemli yapısı olan 'Derin Öğrenme' terimini ifade etmeye çalışacağız.

Aslında en başta makinelerin insanı taklit edebileceği, düşünebileceği fikri Alan Turing 1950 yılında yazmış olduğu kitapta '**Can machines think?**' yani 'makineler düşünebilir mi?' sorgusuyla ortaya atılmıştır. [5]

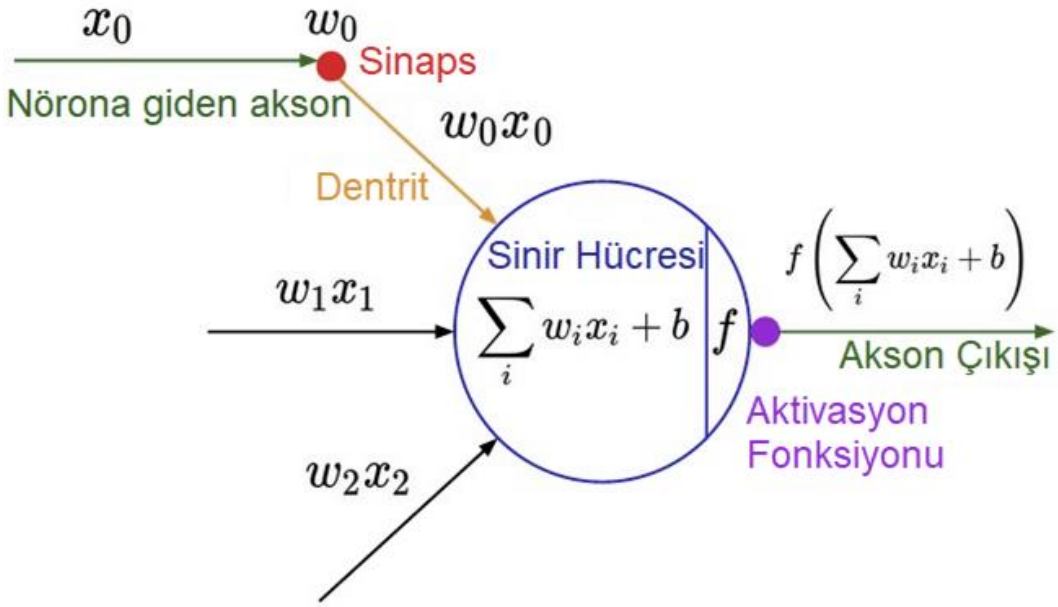
Yapay zekâ, bilgisayarın veya otonom bir yapının kontrolündeki bir robotun veya otonom aracın birden fazla karmaşık görev ve çeşitli işleri akıllı canlılara benzer biçimde taklit veya yerine getirebilme becerisidir.[6]

Yapay zekâ araştırmalarının ilk başladığında, bilim adamları belirli faaliyetleri ifa edebilmeleri için insan zekasını birebir taklit etmeye çalışıyorlardı. Buna en iyi örnek ünlü satranç oyuncusu Garry Kasparov ve tamamen bilgisayar kontrollü 'Deep Blue' adlı bilgisayarın arasında satranç karşılaşmalarıdır. Yine bu çalışmada bilgisayarın rakibiyle empati kurması gereken bir dizi kurallar koydular. Bilgisayar belirli olası eylemler ve hareketler listesine ve ilgili kurallara göre karar alıp gerçekleştirdi.

2.1.2. Derin Öğrenme

Derin Öğrenme, Var olan bir veri seti ile çıkışları tahmin edebilen yapay zekayı modelini eğitme işlemine olanak sağlayan bir yapay zekâ disiplinidir. Modelleri eğitmek için denetimli ve denetimsiz öğrenme çeşitlerinin ikisi de kullanılabilmektedir. Derin öğrenmeyi anlatabilmek için yapay sinir ağları ve evrişimsel sinir ağlarına detaylıca değinmemiz gerekir.[6], [7]

Şekil 2.1'de görüldüğü üzere Yapay sinir ağları, ilk tasarlanırken insan beynindeki nöronların iletişim ve haberleşme ağı örnek alınmış olup aynı insan beyninde olduğu gibi sinir hücrelerinden oluşur. Bu sinir hücreleri birbirine bağlıdır ve çıktıyı etkilemektedir.



Şekil 2.1. Yapay sinir hücresi(nöron) üzerinde yapılan matematiksel işlem

Girişler: $x_0, x_1 x_2 \dots x_i$

Ağırlıklar: $w_0, w_1 w_2 \dots w_i$

Bias: b

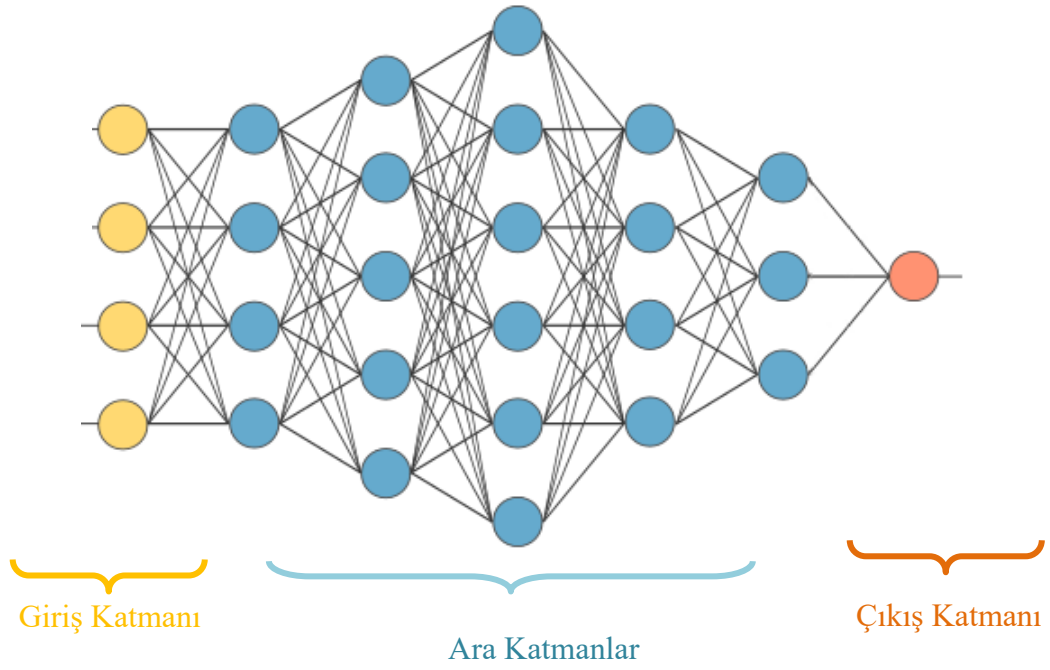
Aktivasyon fonksiyonu: f

Sinir hücresi akson çıkışı denklem 2.1'deki gibidir.

$$y = f(\sum_{i}(x_i w_i + b) \quad (2.1)$$

Yapay Sinir Hücreleri (Nöronlar) dört farklı katmandan oluşur:

- Giriş Katmanı
- Ara Katmanlar
- Çıkış Katmanı



Şekil 2.2. Yapay Sinir Ağları ve Katmanları

Şekil 2.2'de de belirtildiği gibi giriş katmanı giriş verilerini alır ve ilk gizli katmana verileri iletip bu katmanda çeşitli matematiksel hesaplamaları yapar. Yapay sinir ağlarının oluşturulmasındaki zorluklardan biri, her bir katman için sinir hücreleri sayısının yanı sıra gizli katmanların sayısına ve ekstra istatiksel hesaplamalara yarayan maliyet fonksiyonunu en aza indirgeyen parametrelere de -hiper parametrelere- karar vermektir. Bu modelin diğer bölümlerde de değineceğimiz hiper parametresi olarak sonucun doğruluğuna doğrudan katkı sunmaktadır. Derin Öğrenme ismindeki "derin" kavramı, aslında modelde birçok gizli katmanın bulunmasını ifade etmektedir. Çıkış katmanı ise yapılan hesaplamalarda elde edilen çıkış verilerini kullanıcıya veya sisteme bildirir.

Sinir hücrelerinin arasındaki her bağlantı bir katsayı yani bir ağırlık değeri ile ilintilidir. Bu ağırlık, girdi değerinin modeldeki karar verici etkinliğini belirler. Modelde ağırlıklar başlangıçta rastgele ayarlanır. Şekil 2.1'de de belirtildiği her nöron bir aktivasyon fonksiyonuna sahiptir (Fonksiyon çeşitlerinden bazıları çalışmamızda da yer aldığı için diğer bölümlerde detaylı bir biçimde açıklanacaktır). Aktivasyon fonksiyonunun işlevi bir sinir hücresinden elde edilen çıkışları tüm sistemlerin tanıyabileceği hale getirmektir. Bir veri seti yapay sinir ağının geçip çıkışta kullanıcıya bilgi verir hale gelir. Derin öğrenmen disiplininin en önemli problemlerinden biri yapay sinir ağını eğitme işidir. Çünkü büyük bir veri setlerine gereksinim vardır ve büyük boyutlu matematiksel işlem gücü gereklidir.

Yapay zekâ sistemlerini eğitebilmek adına, veri kümemizdeki girişleri vermek ve çıkışlarını veri kümemizdeki çıkışlarla karşılaştırmak gerekmektedir. Yapay zekâ sistemi hala eğitimsiz olduğundan, çıkışları yanlış olacaktır. Bütün veri seti incelendiğinde, yapay zekâ çıktılarının gerçek çıktılardan ne kadar yanlış olduğunu gösteren bir işlev oluşturabilir. Bu işleve maliyet işlevi denilir. Modelin ele alındığı bölümde değinilecektir. Modelin eğitimi esnasında isteğimiz maliyet işlevinin sıfır (0) veya sıfıra çok yakın bir değerde olmasıdır. Buradan çıkarılacak sonuç ise yapay zekâ çıkışları ile veri kümesindeki çıkışların aynı olduğu anlamına gelir.

Maliyet işlevini en aza indirebilmek adına, veri setinde birden fazla tekrarlama yapılması gerekir. Bu sebeple çok büyük ölçeklerde matematiksel işlem gücü yetisine gereksinim duyulur.

Veri setindeki tüm tekrarlamaların ardından, maliyet işlevi değerini azaltabilmek için yapay sinir hücreleri katsayıları çeşitli tekniklerle Stokastik gradyan iniş, Adam, Adadelta, Adaboost vs. gibi değişik teknikler ve algoritmalarla değiştirilir. Böylece maliyet işlevini azaltmak adına yapay sinir hücreleri bağlantılarındaki ağırlıklar bu tekniklerle ayarlanmış olur. Şekil 2.3.'den de anlaşılacağı üzere tüme varım yaklaşımla Derin Öğrenme ve Yapay Zekâ arasındaki ilişki anlaşılabilir. Yapay zekâ derin öğrenme disiplinini kapsar.[7]

Şekil 2.3. Yapay Zekâ ve Derin Öğrenme arasındaki Kapsama İlişkisi

2.2. Evrişimsel Sinir Ağları

Günümüzde örüntü tanıma, sınıflandırma ve görüntü tanıma problemlerimizde kullanılan yapaya zekâ içerisinde yer alan derin öğrenme algoritmalarının temelinde evrişim yani konvolüsyon işlemi vardır. Konvolüsyon genellikle sinyal işleme disiplininde matematiksel bir işlem olmasının yanı sıra ilgili işlemi tarif edebilmek için matematiksel birçok ifade kullanılmıştır.

$$y(t) = x(t) * h(t) \qquad (2.2)$$

Yukarıdaki formüldeki gibi konvolüsyon, birim dürtü yanıtı h(t) olarak bilinen bir sistemin, x(t) giriş işaretine karşılık üreteceği y(t) çıkış işaretini zaman domaininde bulmaya yarayan bir işlemdir. Yapay zekâ disiplinindeki ayrık konvolüsyon işlemidir.

$$\sum_{k=-\infty}^{k=\infty} x\,[n].\,h[n-k] = \mathrm{y[n]} = \mathrm{x[n]} * \mathrm{h[n]} \qquad (2.3)$$

Yukarıdaki denklemde konvolüsyon-evrişim işleminin ayrık bağıntısı görülmektedir. Bu denklem konvolüsyon integralinin de ayrık zamanda süper pozisyon yani toplam halidir.

$$\mathrm{y[a,b]} = \sum_{j}\sum_{i} x\,[i,j].\,h[a-i,b-i] \qquad (2.4)$$

Yukardaki ikili ayrık toplam ilgili denklemin uygulamalarda i ve j indisli 2 adet 3x3=9 boyutlu bir matriste örnek hesabıyla aşağıda adım adım açıklarsak;

1	**2**	**3**
4	**5**	**6**
7	**8**	**9**

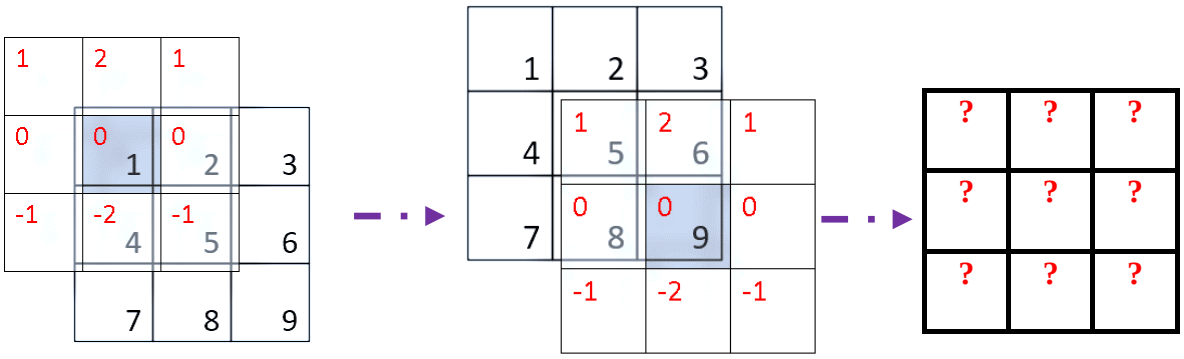
*

1	**2**	**1**
0	**0**	**0**
-1	**-2**	**1**

Şekil 2.4. İki matrisin konvolüsyon işlem temsili

Birinci işlemde denklem (2.4) 'te a =0 b=0 olması durumunda aşağıdaki formülle sırayla bir adım kaydırılarak yapılır. Örneğin birinci işlem için aşağıdaki gibi olacaktır.

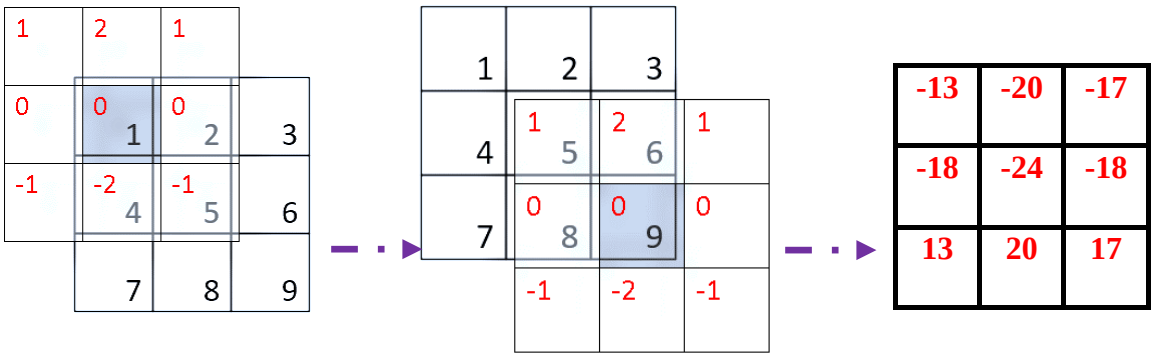
$$\mathrm{y[0,0]} = \sum_{j}\sum_{i} x\,[i,j].\,h[0-i,0-i] \qquad (2.5)$$



Birinci işlem Dokuzuncu İşlem Süperpozisyon Matrisi

Şekil 2.5. İki matrisin evrişim(konvolüsyon) işlemi ve muhtemel çıkış matrisi

Sonuç olarak konvolüsyon yukarıdaki işlem adımları tamamlanınca elde edilen matris; iki (2) matrisin 1 adım kaydırarak toplam 9 adımda yapılan işlemler sonucunda elde edilen süperpozisyon matrisidir.



1.işlem 9.İşlem Süperpozisyon Matrisi

Şekil 2.6. İki matrisin evrişim işlemi ve hesaplanan çıkış matrisi

Evrişim konvolüsyon işlevi matematiksel temel olarak Şekil 2.6'da belirtildiği gibidir.[8]

Evrişim işlemi uygulamada değişik boyutlardaki veriye göre ve değişik kaydırma boyutları ve filtre seçimine bağlı olarak değişiklik göstermektedir. Şimdi derin öğrenme yapısında filtre adım ve ortaklama işleminin olduğu bir yapıyı el yazısıyla rakamları tanıma verisi MNIST üzerinde inceleyeceğiz. Daha öncesinde ESA modelindeki üzerindeki çalışmamızda da kullanacağımız matematiksel işlemlerden ortaklama işlemi, kaydırma işlemi başlıklarına değinmekte fayda olacaktır.

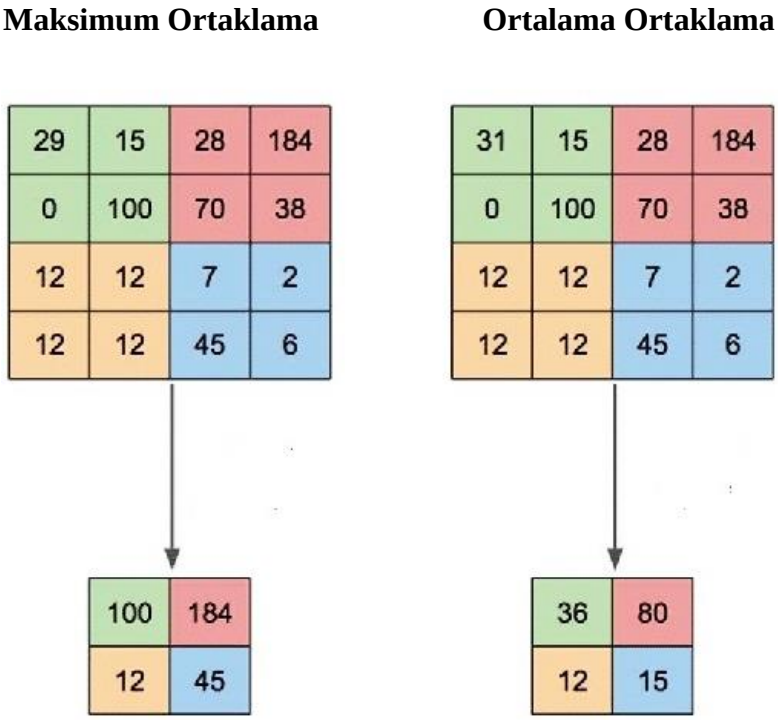
2.2.1. ESA Modelinde Ortaklama İşlemi

Ortaklama (Pool) katmanı, genellikle mekânsal olarak değişken olan evrişim katmanından sonra, modele başka hesaplama tekniği olarak katılan bir çeşit örnekleme işlemi biçimidir.

Çizelge 2.1. Maksimum ve Ortalama Ortaklama Özellikleri [10]

Tip	Maksimum ortaklama	Ortalama ortaklama
Amaç	Her ortaklama işlemi, ilgili matrisin maksimum değerini seçer ve hesaba katar	Her ortaklama işlemi, ilgili matrisin değerlerinin ortalamasını hesaba katar
Açıklama	• Algılanan özellikleri korur • En çok kullanılanlardandır	• Boyut azaltarak örneklenmiştelik öznitelik haritası

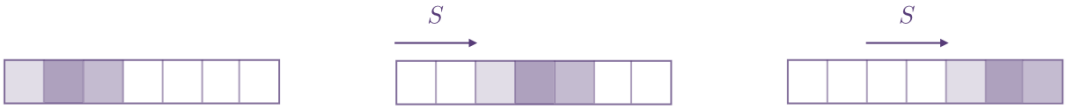
Çizelge 2.1'de görüldüğü üzere özellikle örneklem penceresi sınırlarında derin sinir ağlarında sıklıkla kullanılan sırasıyla ilgili matrisin maksimum ve ortalama değerin alındığı ve matematiksel işleminin Şekil 2.7' de açıklandığı özel ortaklama türlerindendir.[10]



Şekil 2.7. Maksimum ve Ortalama Ortaklama Matematiksel işlem örneği [10]

2.2.2. ESA Modelinde Kaydırma İşlemi

Şekil 2.8'de görüldüğü üzere S ifadesi adım aralığını gösterir aslında her matematiksel işleminin ardından pencerenin ne kadar hareket edeceği piksel sayısını ifade eder.[11]

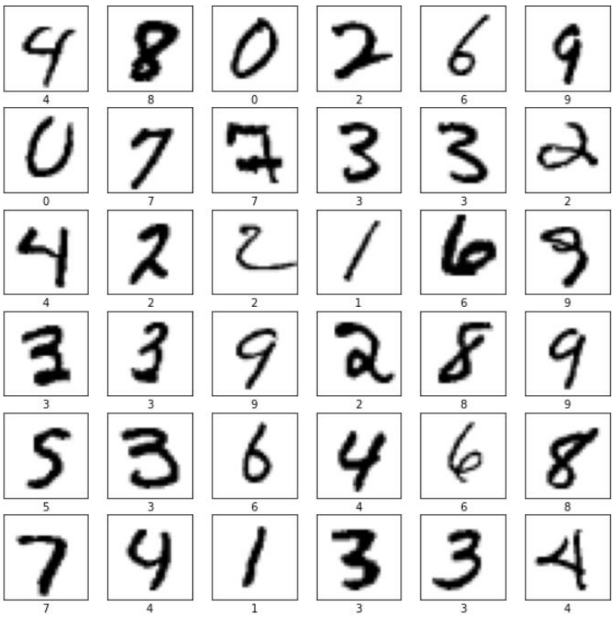


Şekil 2.8 Adım kaydırma işlemi örneği [11]

2.2.3. El yazısıyla yazılan rakamların tanınması, MNIST veri kümesinin incelenmesi

Yann LeCun ve birlikte çalıştığı bilim insanlarının 1998 yılında Amerika Birleşik Devletleri AT&T Bell laboratuvarlarında el yazısı rakamlarının tanınması ve sınıflandırması ilgili yaptıkları çalışma MNIST veri kümesi kullanılmış ve ESA modeliyle sınıflandırma yapılmıştır. [12]

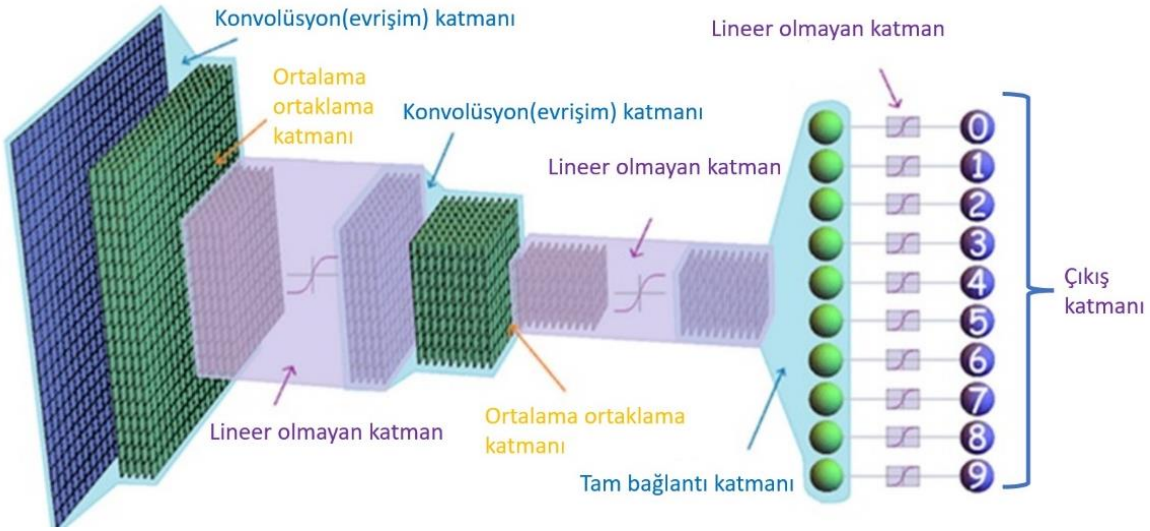
MNIST "Modified National Institute of Standards and Technology Database" açılımının kelimelerinin baş harflerinden oluşan bir kısaltmadır, çeşitli yapay zeka sistemlerini ve ESA modellerini eğitme sürecinde çok yaygın bir şekilde kullanılmakta olan el yazısı rakamlarının oluşturulmuş çok geniş bir veri tabanıdır. Bu veri seti aslında Amerika'da sayım bürosu çalışanlarından 60.000 eğitim görüntüsü ve lise öğrencilerinin el yazılarından elde edilmiş 10.000 test görüntüsünü içermektedir. Ayrıca Şekil 2.9'da örneği görülen bu veri setinde; derin öğrenme, yapay zekâ modellerinde test ve eğitim girişleri olarak yararlanılmaktadır.[13]



Şekil 2.9. MNIST veri kümesinden görüntü ve sınıflandırma örnekleri

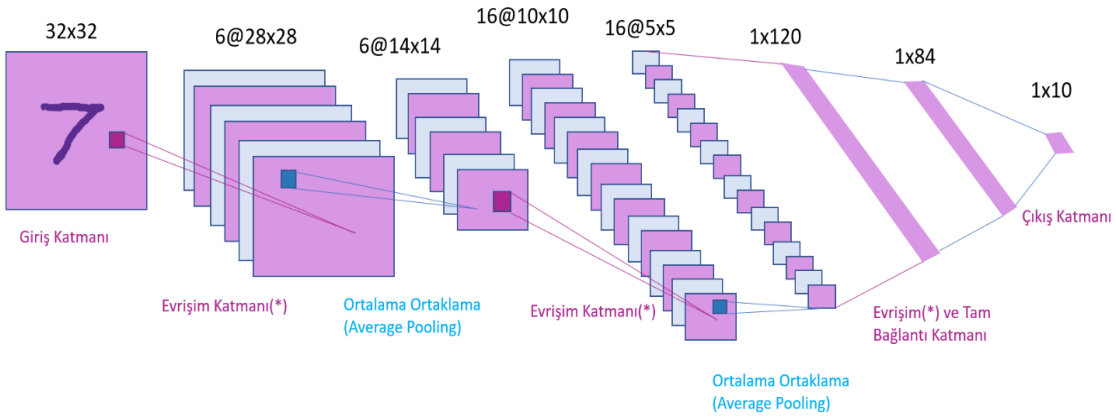
2.3. Le-Net 5; MNIST Veri Setinin ESA modeli ile Sınıflandırılması

Yapay zekâ ile ilgili yapılan çalışmalarda MNIST veri kümesi insan hareketini taklit etme için biçilmiş bir kaftandır. Çünkü yapılan araştırmalarda insanın el yazısı onun en doğal hareketlerinden biri olarak bilinmektedir. MNIST veri kümesindeki rakamları daha iyi ayırt edebilmek ve sınıflandırabilmek için derin öğrenme disiplininde alan ESA modelini uygulayacağız.



Şekil 2.10. MNIST veri kümesi için ESA Modeli

Şekil 2.10'da görüldüğü üzere bu bilgiler ışığında bir MNIST verisine çalışmamızın temelinden kullanacağımız kendimize ait bir ESA modelini uygulayacağız. El yazısıyla yazılan rakamları tanınması problemini Yann LeCun ve birlikte çalıştığı bilim insanlarının 1998 yılında Amerika Birleşik Devletleri AT&T Bell laboratuvarlarında Le-Net 5 modelini uygulayarak çözmüştür. Bu çalışma insan hareketlerini tahmin etme konusunda çığır açmış ve tüm yapay zekâ destekli araştırmaların temelinde kullanılan bir çalışma olarak tarihe geçmiştir.[14]



Şekil 2.11. Le-Net 5 Modeli

Le-Net 5 modelindeki Şekil 2.11'de çizmiş olduğumuz katmanları sıralarsak;

1) Giriş Katman (32x32)

2) Evrişim katmanı(6@28x28)

3) Ortalama Ortaklama katmanı (6@28x28)

4) Evrişim katmanı (6@28x28)

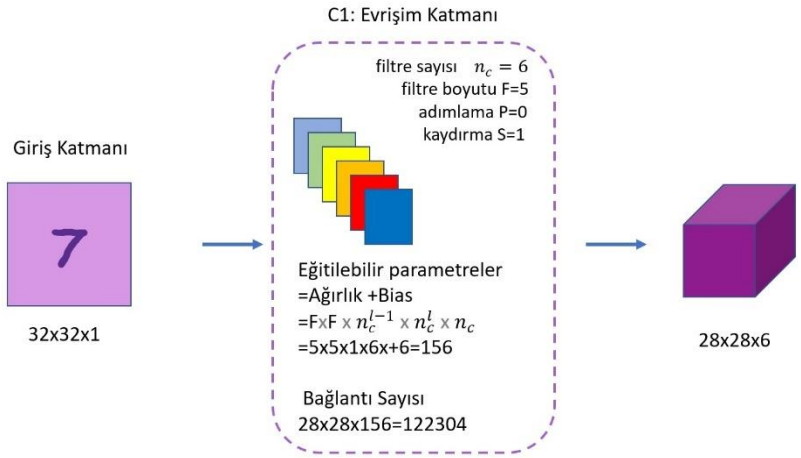
5) Ortalama Ortaklama katmanı (6@28x28)

6) Evrişim ve Tam Bağlantılı Katman (1x120)

7) Tam Bağlantılı Katman(1x84)

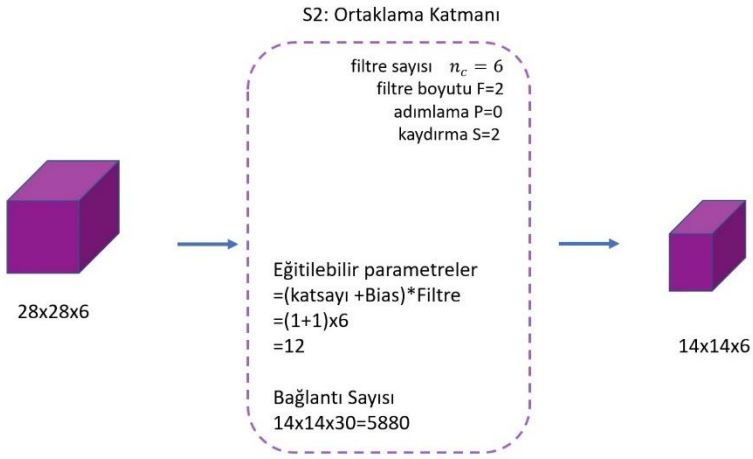
8) Çıkış Katmanı (0-9 rakamlar)

Modeldeki @ işareti derinlik anlamına gelmektedir. Örneğin 2. Evrişim katmanı 6 adet 28x28 boyutundaki ifade etmektedir.



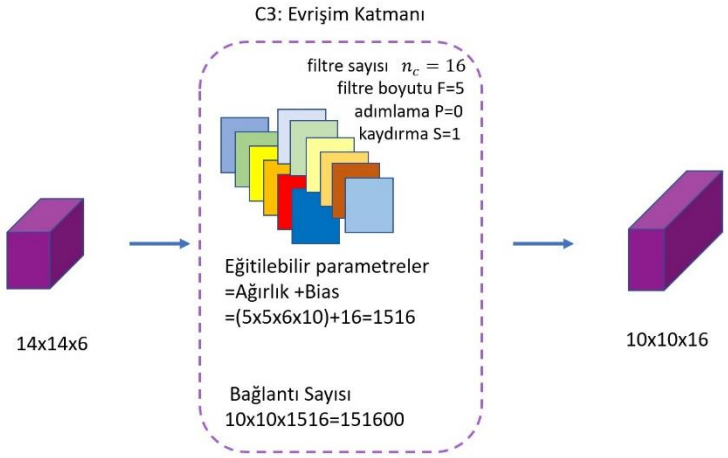
Şekil 2.12. Le-Net 5 Modeli C1: Evrişim Katmanı

Şekil 2.12'de görüldüğü üzere C1: Evrişim Katmanında filtre sayısı 6 birim ,filtre boyutu 5 birim, adımlama sayısı 0 ve kaydırma sayısı 1 seçilmiştir. Evrişim işlemi sonucu 32x32x1 çıkışta 28x28x6 boyutlu bir matris elde edilmiştir.



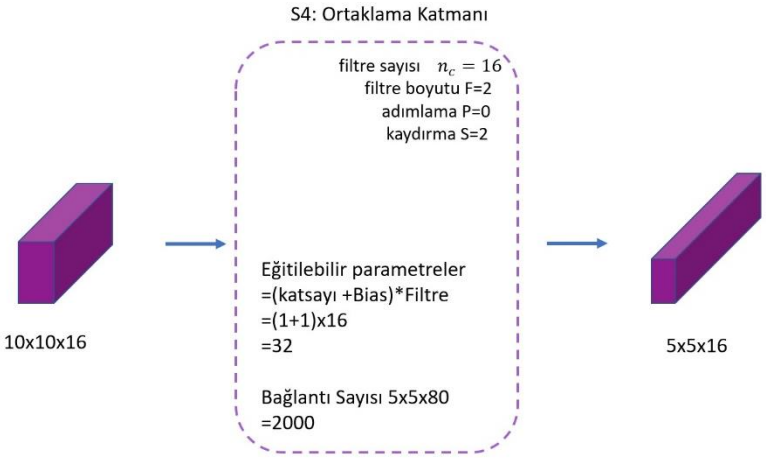
Şekil 2.13. Le-Net 5 Modeli S2: Ortaklama Katmanı

Şekil 2.13'te görüldüğü üzere S2: Ortaklama katmanında filtre sayısı 6 birim ,filtre boyutu 2 birim, adımlama sayısı 0 ve kaydırma sayısı 1 seçilmiştir. Ortaklama işlemi sonucu 28x28x6 çıkışta 14x14x6 boyutlu bir matris elde edilmiştir.



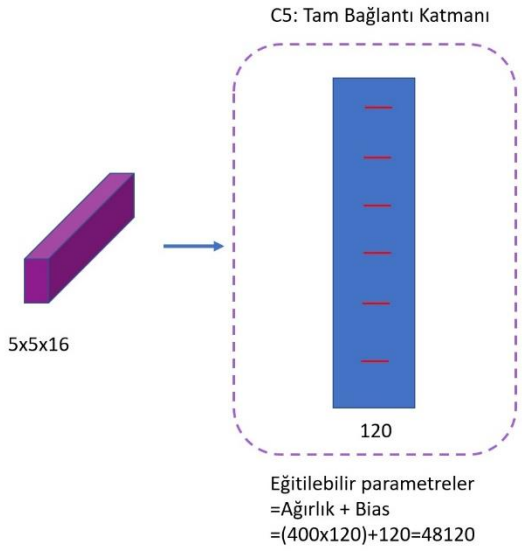
Şekil 2.14. Le-Net 5 Modeli C3: Evrişim Katmanı

Şekil 2.14'de görüldüğü üzere C3: Evrişim katmanında filtre sayısı 16 birim ,filtre boyutu 5 birim, adımlama sayısı 0 ve kaydırma sayısı 1 seçilmiştir. Evrişim işlemi sonucu 14x14x6 çıkışta 10x10x16 boyutlu bir matris elde edilmiştir.



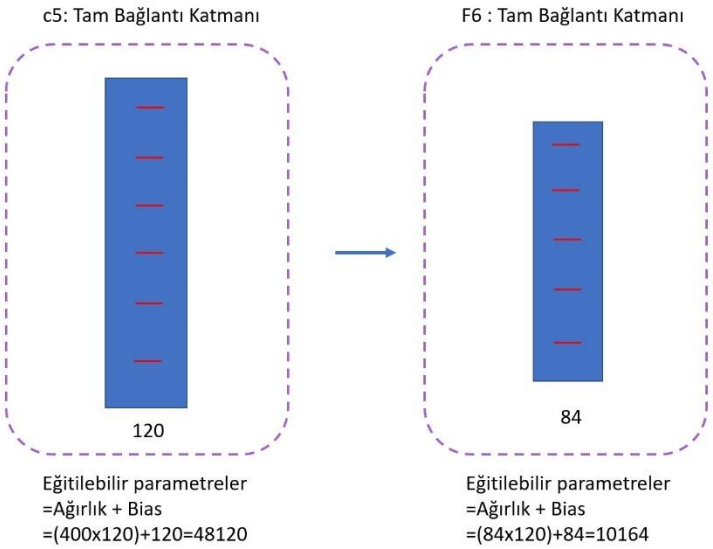
Şekil 2.15. Le-Net 5 Modeli S4: Ortaklama Katmanı

Şekil 2.15'te görüldüğü üzere S4: Ortaklama katmanında filtre sayısı 16 birim, filtre boyutu 2 birim, adımlama sayısı 0 ve kaydırma sayısı 2 seçilmiştir. Ortaklama işlemi sonucu 14x14x6 çıkışta 10x10x16 boyutlu bir matris elde edilmiştir.



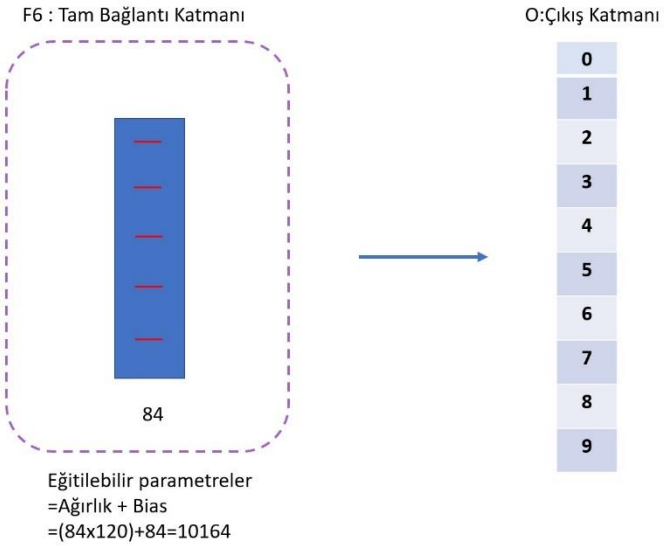
Şekil 2.16. Le-Net 5 Modeli C5: Tam Bağlantı Katmanı

Şekil 2.16'da görüldüğü üzere C5: Tam bağlantı katmanında işlemi sonucu 5x5x16 çıkışta 1x120 boyutlu bir matris 48120 eğitilebilir parametre elde edilmiştir.



Şekil 2.17. Le-Net 5 Modeli F6: Tam Bağlantı Katmanı

Şekil 2.17'de görüldüğü üzere tam bağlantı katmanında 1x84 boyutlu 10164 eğitilebilir parametre edilmiştir.



Şekil 2.18. Le-Net 5 Modeli Çıkış Katmanı

Şekil 2.18'de de görüldüğü üzere eğitilebilir parametre sayılarının da hesaplandığı MNIST veri kümesine ESA modeli uyarlandığında çıkışta 0 ve 9 arasındaki rakamları tahmin edilmesi planlamaktadır, bu yüzden tahmin edileceği üzere çıkış katmanı sayısı 1x10 boyutunda olacaktır. Tüm bu verilerden yola çıkarak aşağıdaki çizelgeyi oluşturulabilir. [Ek-1]

Çizelge 2. 2. Le-Net 5 Modeli ve Tüm Katmanlarının Parametreleri

Katman		Filtre Sayısı	Boyut	Kernel Boyutu	Kaydırma Sayısı (S)	Aktivasyon Fonksiyonu
Giriş	Görüntü	**1**	**32x32**	**-**	**-**	**-**
C1	Evrişim	**6**	**28x28**	**5x5**	**1**	**tanh**
S2	Ortalama Ortaklama	**6**	**14x14**	**2x2**	**2**	**tanh**
C3	Evrişim	**16**	**10x10**	**5x5**	**1**	**tanh**
S4	Ortalama Ortaklama	**16**	**5x5**	**2x2**	**2**	**tanh**
C5	Evrişim	**120**	**1x1**	**5x5**	**1**	**tanh**
F6	Tam Bağlantı	**-**	**84**	**-**	**-**	**tanh**
Çıkış (O)	Tam Bağlantı	**-**	**10**	**-**	**-**	**softmax**

Le-Net 5 modeline ait tüm katmanlardaki parametre sayıları şekiller yardımıyla hesaplarsak Çizelge 2.2 elde edilecektir. Çalışmamızda yer alan Çizelge 2.2'de de görüldüğü üzere asıl modelimizde kullanılan aktivasyon fonksiyon tipleri belirtilmiştir. Modelimizde kullanılan aktivasyon fonksiyonu softmax ve tanh yani hiperbolik tanjant fonksiyonlarıdır.

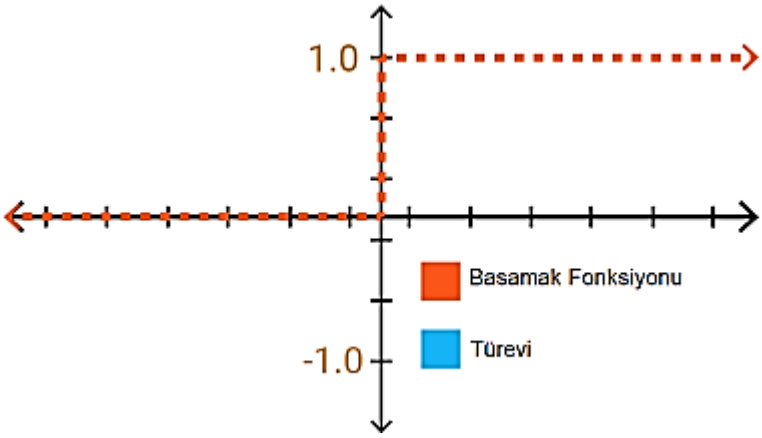
2.3.1. ESA Aktivasyon İşlevleri

Doğrusal olmayan gerçek dünya ve fiziksel koşulların özelliklerini yapay sinir ağlarına tanıtabilmek için çeşitli problemin durumuna göre aktivasyon işlevlerine gerek duyulabilir. En basit yapıdaki yapay sinir ağında x simgesi girişleri, w simgesi ağırlıkları ifade eder ve yapay sinir ağının çıkışına aktarılmış olan değere f(x); aktivasyon işlevi uygulanır.

Derin öğrenme modellemelerinde sıklıkla kullanılan belli başlı aktivasyon işlevleri aşağıdaki gibidir:

2.3.1.1. Basamak İşlevi

Şekil 2.19'da grafiği görüldüğü üzere derin öğrenme modellemelerinde geriye yayılım algoritmalarında türevi temsil edilebilir bir değer olmadığı için pek kullanılmayan iki değer alabilen bir işlevdir ve ikili sınıflandırıcılardan biridir. Bu sebeple yaygın olarak modelin çıktı katmanında kullanılmaktadır. Modelin gizli katmanlarında kullanılması işlevin türevi öğrenme katsayısı değerini tam olarak ifade etmediği için faydalı olmayacaktır.

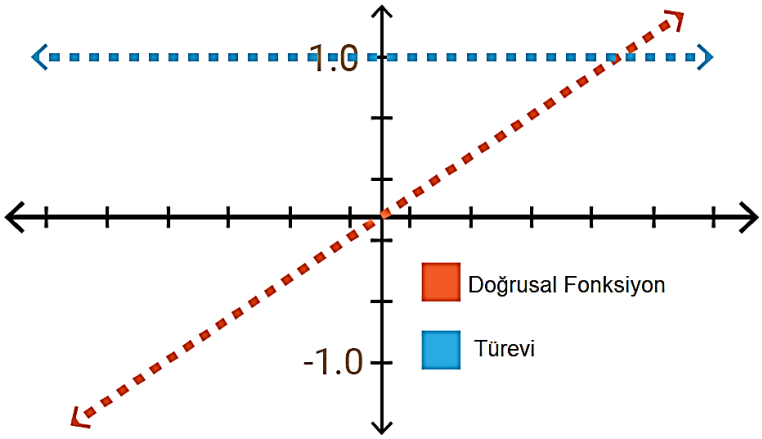


Şekil 2.19. Basamak Fonksiyonu[15]

2.3.1.2. Doğrusal (Lineer) İşlevi

Şekil 2.20'de grafiği görüldüğü üzere yapay sinir ağında basamak işlevinin aksine ikili değer olmayan bir seri değer üretir. Ancak denklem türevi sürekli sabit olduğu için geriye yayılım algoritmalarında kullanılması tercih edilmez. Diğer bir dezavantajı ise tüm katmanlarda doğrusal işlev kullanıldığında modelin giriş katmanı ile çıkış katmanı arasındaki sonuç sürekli aynı doğrusallıkta olacaktır.Yani fiziksel ortamdaki girişin değişken veri olması durumuna göre bir çıkışta yine doğrusal bir sonuca ulaşılacağı için sınıflandırma doğruluğu düşecektir.

$$y = mx \tag{2.15}$$

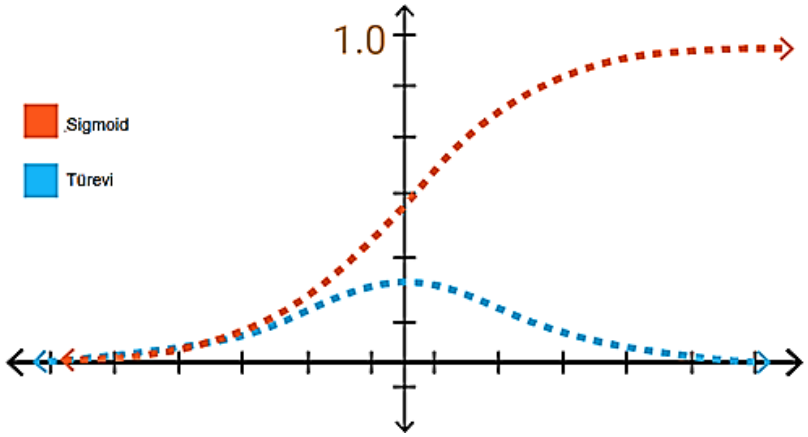


Şekil 2.20. Doğrusal Fonksiyon [15]

2.3.1.3.Sigmoid İşlevi

Sigmoid işlevi basitçe aşağıdaki denklemden yola çıkarak hesaplanır. Aşağıdaki Şekil 2.21'de ifade edildiği gibi kartezyen koordinat sisteminde S harfi (sigmoid) şeklinde olduğu için sigmoid olarak adlandırılır.

$$sigmoid(x) = \frac{1}{1 + e^{-x}} \tag{2.16}$$



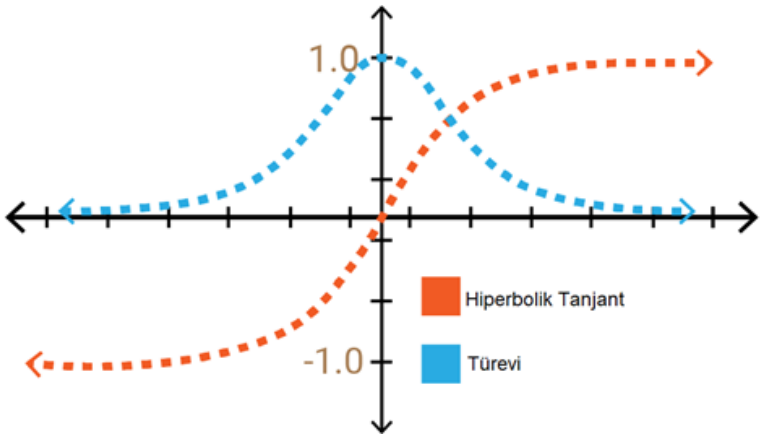
Şekil 2.21. Sigmoid İşlevi [15]

2.3.1.4. Hiperbolik Tanjant İşlevi

Şekil 2.22'de grafiği görüldüğü üzere trigonometrik bir işlev olup sigmoid işlevinin kaydırılmış halidir. Temel olarak aşağıdaki 2.17 denkleminden yola çıkılarak hesaplanır.

$$\tan h(x) = \frac{2}{1 - e^{-2x}} - 1 \qquad (2.17)$$

Derin öğrenme modellerinde kullanılması yüksek sınıflandırma ve çabuk öğrenme işlemleri için daha fazla aralığa sahip olacağından verimlilik sağlayacaktır.



Şekil 2.22. Hiperbolik Tanjant İşlevi [15]

Sigmoid işlevine çok benzer yapıya sahip olmakla birlikte türevinin daha dik ve kapsayıcı alanda olduğu için kullanmak avantajlıdır. Dezavantajı ise fonksiyonun uç noktalarında azalış değerlerini tespit edilememe problemidir.

2.3.1.5.Softmax İşlevi

Sigmoid işleviyle oldukça yakın yapıda sahip olmakla birlikte genellikle ikiden fazla sınıflandırma yapabilen derin öğrenme modellerinin çıkış katmanında kullanılır. Sınıflandırma için kullanılırsa oldukça verimli sonuçlar elde edilebilir. Softmax işlevi [0,1] aralığında gerçek değerler üreten N boyutlu bir vektör olup aşağıdaki denklem yardımıyla hesaplanır.

$$S(a) = \begin{bmatrix} a_1 \\ a_2 \\ \dots \\ a_N \end{bmatrix} \longrightarrow \begin{bmatrix} S_1 \\ S_2 \\ \dots \\ S_N \end{bmatrix}$$

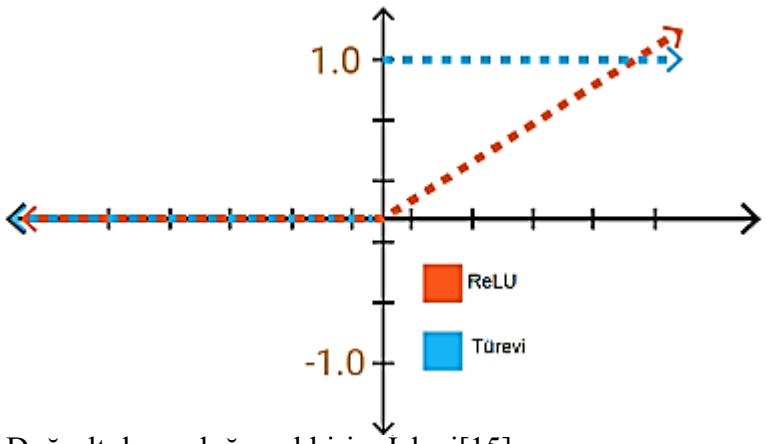
$$s_j = \frac{e^{a_j}}{\sum_{k=1}^{N} e^{a_k}} \qquad (2.18)$$

Softmax çok verimli bir aktivasyon işlevi, sadece çıktımızı bir [0,1] aralığına eşlemekle kalmayıp aynı zamanda her çıkış toplamı 1 olacak şekilde eşler. Softmax fonksiyonu çıkışı bu nedenle bir olasılık dağılımıdır.

2.3.1.6. Doğrultulmuş Doğrusal Birim İşlevi

Şekil 2.23'te grafiği görüldüğü üzere DDB günümüzde oldukça çok kullanılan aktivasyon işlevidir. DDB aktivasyon işlevine sahip bir sinir hücresi, giriş olarak herhangi bir gerçek değeri alır, fakat sadece bu girişler 0'dan büyük olduğunda aktif hale gelir. DDB aktivasyon işlevinin denklem 2.19'da ve grafiği de aşağıdaki gibidir:

$$f(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases} \qquad (2.19)$$

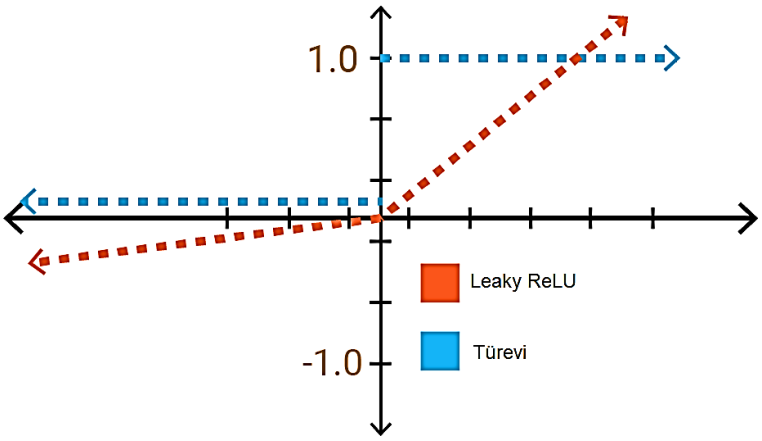


Şekil 2.23. Doğrultulmuş doğrusal birim İşlevi[15]

2.3.1.7. Sızıntı Doğrultulmuş Doğrusal Birim İşlevi

Şekil 2.24'de grafiği görüldüğü üzere DDB aktivasyon işlevinin yok olan 0' dan küçük değişken değerleri için denklem 2.20'de görüldüğü gibi bir çıkış üretir. SDDB işlevinin tanım aralığı -∞,+∞ aralığındadır. (a =0.01)

$$f(x) = \begin{cases} ax, & x < 0 \\ x, & x \geq 0 \end{cases} \tag{2.20}$$



Şekil 2. 24. Sızıntı Doğrultulmuş Doğrusal Birim İşlevi [15]

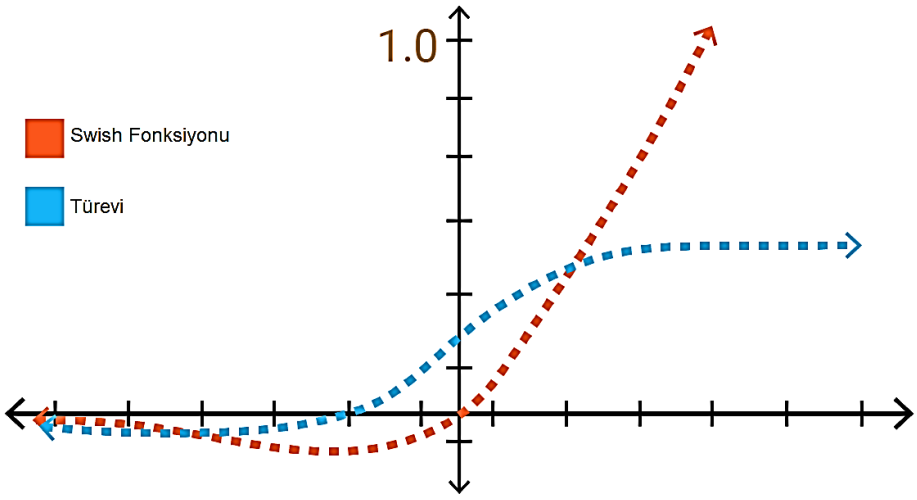
2.3.1.8. Swish (kendinden geçitli) İşlevi

Şekil 2.25'te grafiği görüldüğü üzere Doğrultulmuş Doğrusal Birim (ReLU) aktivasyon işlevinin yok olan 0' dan küçük değişken değerleri için sızıntı doğrultulmuş doğrusal birim (ReLU) aktivasyon işlevinin aksine denklem 2.21'de sabit eğimi olmayan yani negatif bölgede doğrusal olmayan bir çıkış üretir.

$$f(x) = 2 \times \sigma(\beta x) = \begin{cases} \beta = 0, f(x) = x \\ \beta = \infty, f(x) = 2\max(0, x) \end{cases} \tag{2.21}$$

$$sigmoid\ işlevi \qquad \sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.22}$$

Denklem 2.21'de yer alan $\sigma(x)$ terimi aslında daha önce de bahsedilmiş olan denklem 2.22'deki sigmoid işlevi.



Şekil 2.25. Swish İşlevi [15]

3. MATERYAL VE METOT

3.1. Python Programlama Dili ve Phycharm Ortamı

1990'lı yılların başında Python programlama dili, ilk olarak bir araştırma merkezinde çalışan Guido Van Rossum tarafından geliştirilmeye başlanmış olup aslında python tam anlamıyla programlama dili olma aşamasında, diğer yaygın kullanım için hazırlanmış olan ABC programlama dilinden esinlenilmiştir.Van Rossum, Centrum Wiskunde & Informatica (CWI) adlı bilimsel araştırma kurumunda, ilgili çalışmış olduğu ABC programlama dilin basit ve esnek olmayışı nedeniyle verimli bulmamış ve geliştirmeye açık olmadığı düşüncesiyle ABC programlama dilinin etkin ve efektif yönleri kullanarak python programlama dilini ortaya çıkarmıştır. Python programlama dili, en baştan beri revaçta ve kolay erişilebilirliği sayesinde oldukça fazla yazılımcı sayısına ulaşan, dinamik kodlanabilen ve nesne tabanlı programlama dilidir. Bu programlama dili, öncelikle sadelik ve kolaylık özellikle de söz dizimi yapısındaki esneklik sebebiyle dikkat çekmektedir. Diğer popüler dillere göre mevzu bahis programlama dilinde çok daha az kod satırı ile çok daha işlevsel ve anlaşılır kod yazılabilmektedir. [16]

Üstelik, python programlama dilinde kod yazım işlemi yapılırken satırlardaki girintiler çok önemlidir. Bu durum kodun anlaşılabilirlik ve düzenlilik durumunu göstermektedir. Python programlama dili nesne tabanlıdır. Sözcük biçiminde değişkenleri tanımlayan karakter ve dizge veri tipleri, listeler, tüpler, dosyalar ve sözlükler ,sayı biçiminde değişkenler tanımlayabildiğimiz tamsayı ve reel sayı gibi belli başlı veri tipleri Python dilinde algoritmadaki değişkenleri tanımlamak için kullanılan veri tipleridir. Bu veri tiplerinin her biri aslında birer sınıftır. Dolayısıyla python programlama dilinde tanımlanan herhangi bir değişken aslında yukarıda örneklerini verdiğimiz veri tiplerine ait birer nesneyi temsil eder.

Python programlama dili için oluşturulmuş kapsamlı kütüphaneler birden çok ana bilim disiplininde yazılım oluşturma ve yürütmeye olanak sağlamaktadır. Bilimsel araştırma yapılmak istenilen dala ait kütüphane, terminalden pip install komutu ile çapraz platforma daha sonra da projeye “”import modül_ismi” kod satırı şeklinde eklenebilmektedir. Python programlama dilinin aşağıdaki konularda uygulama alanları bulunmaktadır:

- Hesaplamalı Biyoloji ve Biyoenformatik Bilimlerinde Python Kullanımı
- Grafiksel kullanıcı arayüzü ve 3 boyutlu oyun algoritmaları
- Ağ Modelleme
- Veri Madenciliği
- Bilimsel Hesaplamalar
- İnternet tabanlı uygulamalar
- Derin Öğrenme vs. [17]

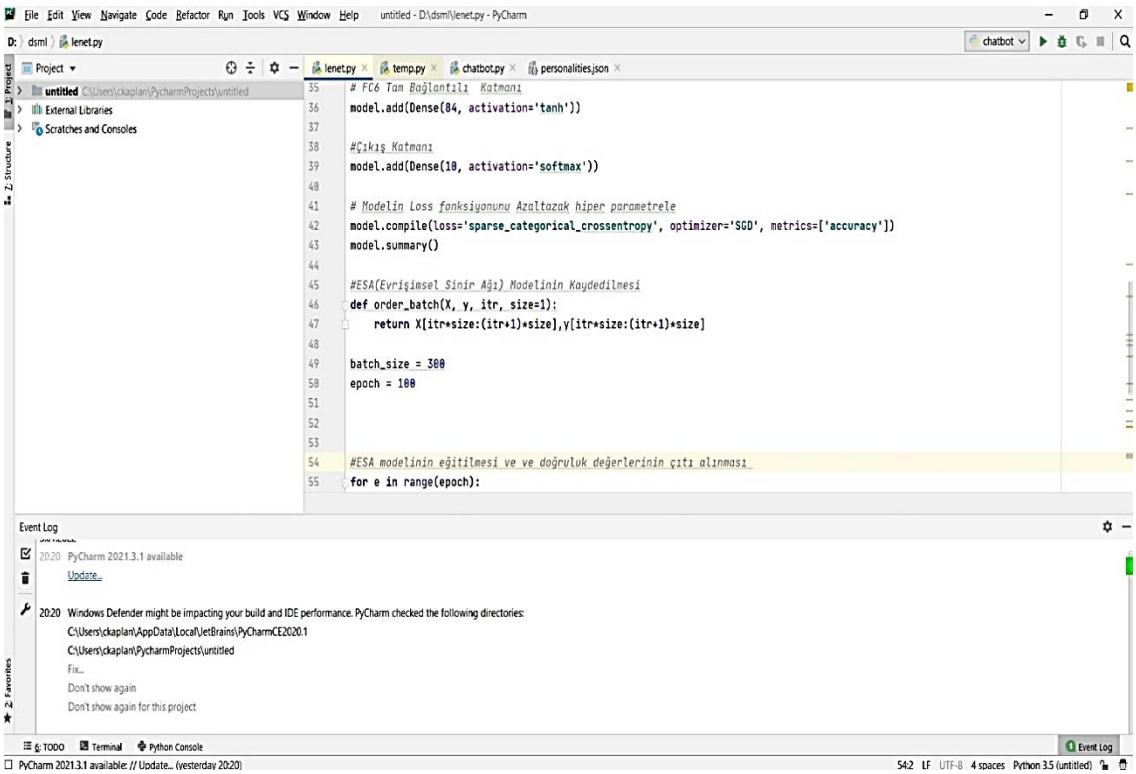
Tez çalışmamızın içerisinde de kullanacağımız bilimsel hesaplamalar derin öğrenme ve veri madenciliği başlıklarına değinilecek olursa: Python programlama dili içerisinde bulunan temel matematiksel fonksiyonları NumPy (Sayısal Python) ve IPython (interaktif Python) SciPy (Bilimsel Python) modülleri ile işlem yapılmasını sağlar, ayrıca temel lineer cebir ,seriler (çok boyutlu) ,ayrık Fourier dönüşümleri

asitmetrik matrislerin aralıklı matrislerin öz değer analizleri, seçme ve dizilim işlemleri, ana istatistiki işlemler gibi daha sayamadığımız bilimsel ve matematiksel hesaplamalar yapılabilmektedir. Plotly ve Matplotlib kütüphaneleri ile de bu hesaplamaların sonuçlarının verileri görselleştirilerek; gelişmiş grafiklerle gösterilip ve yayınlanabilir kalitede çizimler yapılabilmektedir.

Makina öğrenmesi alanında denetimli ve denetimsiz öğrenme araştırmaları, değişik sınıflandırma,tahmin ve kümeleme işlemlerini gerçekleştirebilmek amacıyla scikit-learn kütüphanesi oluşturulmuştur.Tensorflow kütüphanesi ise yapay zekâ disiplininde grafik işlemci arabirimini kullanan özellikle derin öğrenme modellerine yönelik çalışmalar yapan kullanıcılar için tasarlanmış bir modüldür.

Aynı zamanda Orange yazılımının veri tabanında Python programlama dili bulunmaktadır. Bu programla farklı dosya biçimlerinde kolaylıkla yüklenen veri setleri çeşitli veri madenciliği yazılımlarıyla kullanıcıya görsel kolaylıklar sunabilmektedir.

Python programlama dilini daha verimli ve etkin kullanabilmek için çeşitli kütüphane ve modülleri kullanıcılar tarafından yaratılan çevre üzerinde pip hizmeti ile yüklenmiş olduğu çeşitli ortamlar kullanılır.Şekil 3.1'de de görüldüğü üzere bu çapraz platformlardan birisi de PhyCharm adlı araçtır. Phycharm uygulaması bir Python algoritması ve kod yazma ortamıdır. Görsel tabanlı hata ayıklayıcısı, yazılım sürümü kontrolü sistemine entegre ve Django veri tabanı yardımıyla Python internet tabanlı geliştirmeler yapılmaktadır. Neredeyse tüm işletim sistemleri üzerinde çalışmasıyla çok verimli bir araçtır.[18]



Şekil 3.1. Phycharm arayüzü / platformu

3.2. Zaman Serileri ve Tahmin Fonksiyonları

Birden fazla türdeki değişkenin tarihsel sıralamaya göre değerleriyle oluşturulmuş dizi zaman serisi diye tanımlanmaktadır (t=n,....3,2,1 ve $y_1 y_2, y_3 \dots y_n)$ ve y_t $n$otasyonlarıyla gösterilmektedir.[19]

Zaman serileri ile ilgili detaylı bilgiler verilmeden önce çeşitli terimlerin tanımını yapmada fayda olacaktır.

Standard sapma ($\boldsymbol{\sigma}$) : Verilerin aritmetik ortalama sapma karelerinin aritmetik ortalamasının kare köküne eşit olan değerdir.

$$\sigma = \sqrt{\frac{\sum(X_i - \mu_x)^2}{n}} \tag{3.1}$$

σ : standart sapma

X_i: i durumdaki veri

μ: ilgili verinin aritmetik ortalama değeri

n: öğrenci sayısı

Varyans ($\boldsymbol{\sigma^2}$) : verilerin aritmetik ortalamadan sapmalarının karelerinin toplamına eşit olan değerdir. Yani standart sapmanın karekök alınmamış halidir.

$$varyans, \sigma^2 = \frac{\sum(X_i - \mu_x)^2}{n} \tag{3.2}$$

Kovaryans ($\boldsymbol{\sigma_{xy}}$) : Kovaryans iki değişken arasındaki doğrusal ilişkinin değişkenliğini ölçen bir kavramdır [20],[21].

$$\sigma_{xy} = \frac{1}{N}\sum_{i=1}^{N}(X_i - \mu_x) * \left(Y_i - \mu_y\right) \tag{3.3}$$

Zaman serileri kaydedilme biçimine göre zamanla sürekli elde edilebilen verileri kapsayan serilere sürekli zaman serisi, yalnız belirli dönem aralıklarında elde edilebilen verileri kapsayan serilere de kesikli zaman serileri denilmektedir. Sinyaller, gerilim değeri, elektriksel akım değeri, seslerin titreşimi, yeraltı hareketlerinin fiziksel değerleri gibi mühendislik disiplinine ait seriler sürekli zaman serileri kategorisine girerken, ekonomik verileri içeren seriler kesikli zaman serileri tanımına girmektedir. Tüm zaman serileri gözlemleme frekansına göre kategorilere ayrılır. [21]

Tüm zaman serileri birden fazla sosyal iktisadi ve doğal değişimlerin etkisinde olduklarından ;eğilim (E), mevsimsellik etkisi (M), çevresel etki (Ç) ve düzensizlik durumu etkisi (D) olmak üzere dört bileşeni bulunmaktadır. Bu 4 faktörün zaman serileri değeri iki işlemsel; toplamsal ve çarpımsal olduğu etkisi bilinmektedir. Bir zaman serisi

Toplamsal formülü;

y_t=E+M+Ç+D t= n,.....3,2,1 biçiminde, (3.4)

çarpımsal formülü ise;

y_t=E.M.Ç.D t = n,.....3,2,1 şeklinde yazılabilir. [16] (3.5)

Değerleri zamana bağlı olarak değişen serilere zaman serisi olarak tanımlamıştık. Örneklendirirsek borsada halka arz edilmiş işlem hacmi olan firmalara ait hisse senetlerinin zamanla değişen değerleridir. Eğer bu hisselerin değerleri önceki durumların etkisi olmasaydı zaman serisi kategorisinde olmayacaktı. [22]

Zaman serilerine ait aşağıdaki bileşenler bulunur.

1-Eğilim bileşeni,

2-Mevsimsellik bileşeni,

3-Düzensizlik durumu etkisi,

4-Çevresel bileşenler

Eğilim Bileşeni, zaman serilerinin değerlerinin uzun vadeli dönemde bulunduğu eğilim yönü olarak tarif edilebilir. Örnek olarak borsada yer alan hisse senedi fiyatlarının yükselme eğilimi verilebilir.

Mevsimsellik Bileşeni, mevsimsellik bileşeni yılın aynı döneminde seride gözlemlenen benzer ve aynı tip değer değişimleri olarak ifade edilebilir.

Çevresel Bileşenler, eğilim bileşeninin oluşturduğu doğru ya da eğrinin yakınsandığı değerlerde uzun vadeli değişimlere çevresel değişimler veya bileşenler denir. Ekonomide resesyon ve gelişme yükselme dönemleri vs. çevresel değişimler şeklinde isimlendirilebilir.

Düzensizlik Durumu Bileşeni, belirli olmayan, dönemsel olarak ortaya çıkabilen, genel olarak hata olarak adlandırılan rastlantısal ve belirli bir düzen içerisinde olmayan olaylardır. Pandemi, afetler, savaşlar gibi beklenmedik olayların meydana getirdiği bir zaman serisi bileşenidir. Olayların evvelden tahmin edilmesi imkansızdır.

Durağanlık; zaman serilerinde gecikmeli zaman periyodu değişkeninin (zaman serisinin varyansın sabit, ortalamasının sabit) kovaryansının zamandan bağımsız olduğu durumudur.

Verilerde tahmin inceleme işlemlerinden önce kontrol edilmesi gereken durum, verinin durağanlığın durumunun tespiti elzemdir. Zaman serisindeki değerlerin ortalamasının ve varyansının sabit olması durumudur.

Bir zaman serisi modeli oluşturulmak istenildiğinde, üretilen tesadüfi değişkenlerin var olduğu modelin zamandan bağımlı veya bağımsız değişkene sahip işlev olup olmadığı sorgulanmalıdır. Zamana bağlı bir işlev ise seri durağan değildir. Zaman serileri durağan özellik göstermiyorsa bilimsel biçimlerde ifade edilmesi beklentisi pek mümkün değildir.

Veri setini durağan biçime getirmek için çeşitli yöntemler vardır. Durağanlık durumu tespitinde yapılacak ilk analiz birim kök incelemesi olacaktır. Bu incelemede istatistiksel p değeri anlamlandırılabilir bir değerdeyse veri seti durağan görünümde değildir, dolayısıyla modelde birim kök değeri vardır. Veri setini ya da modeli durağan bir yapıya getirebilmek amacıyla 1. derece farklar alınır ve durağan yapıya sahip veri seti elde edilir. Bu hesaplamalardan sonra yine bir birim kök incelemesi yapılır, aynı durum devam ediyorsa 2. derece farklar alınır burada amaç aslında daha yüksek öz ilinti söz konusu ise, öz ilinti etkisini en aza indirgemektir.

Veriler veri setleri aslında bizlerle konuşur gibi iletişim halinde bizlere söylemek istediklerini göz ardı etmemiz gerekir. Örneğin serinin durağanlığını incelerken hemen 1. derece farkları almamız gerekmez. Bu farklar alınmadan önce göz ardı edilmemesi gereken bir durum da, eğilim mevsimsellik, durağanlık gibi bileşenler veri setinin karakterini yansıtır. Modelin veya veri setinin söylemek istediklerini göz ardı etmeden, karakterini tespit etmeye çalışmalıyız. Tüm eğilim-mevsimsellik-durağanlık sorgulama işlemlerinden emin olduktan sonra devam ediyorsa yapılacak olan farkların alınması yöntemine gidilmelidir. Çizelge 3.1'de de görüleceği üzere eğilim-mevsimsellik-durağanlık sorgulamasından sonra Box-Jenkins modellerinden hangi model biçiminin yakın ve daha uyarlanabilir olduğu tespit edilir.[23]

Çizelge 3.1. Durağan zaman serisi modelleri ve öz-İlinti işlevi özellikleri-Durağan modellerde ana kütle kısmi öz-ilinti

Model	**Öz-İlinti İşlevi**	**Kısmi Öz-İlinti İşlevi**
OM(p)	Sinüzoidal ya da üssel şekilde azalır	p gecikme durumu bilimsel olarak anlamlı kabul edilemez
OM(q)	q gecikmesinden sonra istatistiksel olarak anlamlı kabul edilmez	Sinüzoidal veya üssel şekilde azalır
OOM(p,q)	q-p gecikmesinden sinüzoidal ve7veya üssel biçimde azalır.	p-q gecikmesinden sinüzoidal ve/veya üssel biçimde azalır

Mevsimsel Olmayan Box-Jenkins Modellerine ait parametreler aşağıdaki gibidir:

BOHOM (p,d,q)

p: oto regresyon(AR) modeli derecesi,

d: fark alma işlemi sayısı

q: hareketli ortalama (MA) modelinin derecesi

Mevsimsel Box-Jenkins Modellerine ait parametreler aşağıdaki gibidir:

BOHOM (p,d,q) (P,D,Q),s

P: mevsimsel oto regresyon(SAR) modelinin derecesi,

D: mevsimsel fark alma işlemi sayısı,

Q: mevsimsel hareketli ortalama (SMA) modelinin derecesi, s: periyot

Çizelge 3.2. Zaman Serisi Model Parametreleri ve Matematiksel gösterimleri [24]

Model	**Gösterimi**	**Modelin Matematiksel Gösterimi**	**Parametreler**
Otoregresif Model	OM(p)	$X_t = \sum_{i=1}^{p} \alpha_i X_{t-i} + \varepsilon_i$	$\alpha_i, i = 1,2,\dots,p$
Hareketli Ortalama Modeli	HOM(p)	$X_t = \sum_{i=0}^{q} \beta_j \varepsilon_{t-j} \quad \beta_0 = 1$	$\beta_j, j = 1,2,\dots,q$
Otoregresif Ortalama Modeli	OOM (p,q)	$X_n - \sum_{i=0}^{q} \alpha_i X_{t-i} = \varepsilon_i - \sum_{i=0}^{q} \beta_j \varepsilon_{t-j}$	$\alpha_i, i = 1,2,\dots,p$ $\beta_j, j = 1,2,\dots,q$
Bütünleşik Otoregresif Hareketli Ortalama Modeli	BOHOM (p,d,q)	$\phi_p(B)(1-B)^d X_t = \theta(B)\varepsilon_t$	

Not: α: $d\ddot{u}zeltme\ katsayısı, \beta$: $trend\ d\ddot{u}zeltme\ katsayısı$

Zaman serisi modeli belirlendikten yani Çizelge 3.2'de belirtilen işlemler tamamlandıktan sonra, modelin uyarlanabilirliği belirlenmelidir. Bu işlem ilk olarak parametrelere ait olan değerler yerine koyularak tahmin ettiği değerler tespit edilir. Ardından bu tespit edilen değerler için hata değerleri analiz edilir.

Çizelge 3.3. Zaman Serileri karakteristiği ve tahmin modelleri yöntemleri [25]

Zaman Karakteristiği	**Tahmin Aralığı**	**Model(ler)**
Trend ve mevsimsel etki mevcut değilse	Kısa	Hareketli Ortalama Yöntemi
Trend mevcutsa (mevsimsel etki mevcut değilse)	Uzun	Trend Analizi
Trend mevcutsa (mevsimsel etki mevcut değilse)	Kısa	Çift Üssel Düzeltme Yöntemi
Trend ve mevsimsel etki mevcutsa	Uzun	Parçalara ayırma yöntemi
Trend ve mevsimsel etki mevcutsa	Uzun	Holt-Winters düzeltme yöntemi

Bilimsel anlamda kabul görmüş, önerilen tahmin aralığı analiz (kısa-uzun) -veri seti ilişkisi uygulanması Çizelge 3.3'te belirtilmiştir. Çizelge 3.3'te ifade edildiği üzere veri setinin zaman karakteristiğine göre tahmin yöntemi belirlenir. Bu yöntemlerden tez çalışmasının kapsamında kullanılacak olan analize ilgili bölüm 4'te değinilecektir. Model seçim ölçütleri için hata değerlerinin kullanılacağından bahsetmiştik. İstatistiksel çalışmalarda genelde karesel ortalama hata, kök ortalama karesel hata ve ortalama mutlak yüzde hata gibi kriterlere bakılarak seçim yapılır ve en uygun model belirlenir.

3.2.1 Zaman Serileri ve Tahmin Fonksiyonları Performans Analizi için hesaplanması gereken değerler

İstatistikte, veri biliminde ve yapaya zekâ modellerinin performans değerlendirmesinde çalışmamızda kullanılan bazı hesaplamalara değinecek olursak;

$$A_j = \text{gerçek değerler}, P_j = \text{tahmin edilen değerler}$$

$$e_j = A_j - P_j \tag{3.6}$$

3.2.2 KOH (Karesel Ortalama Hata)

Temelde 3.8 denkleminde de ifade edildiği üzere gerçek değerlerle tahmin arasındaki hataların karesel ortalamasını hesaplayan istatistiksel işlemdir. Karesel ortalama Hata, bir derin öğrenme yapısının, tahmin işlevinin performansını ölçer, sürekli sıfırdan büyüktür yani pozitiftir ve bu değer sıfır değerine ne kadar yakınsa ilgili tahmin işlevinin performans değeri yüksek kabul edilir.[26]

$$\text{KOH} = \frac{1}{n}\sum_{j=1}^{n} e_j^2 \tag{3.7}$$

3.2.3. KOKH (Kök Ortalama Karesel Hata)

Bir yapay zekâ modelinin, tahmin işlevinin tahmin edilen ile gerçekte olmuş değerler arasındaki fark olarak kabul edilen hata değerini elde eden bir ölçüttür. Kök ortalama karesel hata değeri aslında modelin tahmin işlevinin hatalarının standart sapma değerini ifade eder. Kök ortalama karesel hata değeri [0 ∞) aralığındadır. Kök ortalama karesel hata değeri kabul edilebilir hassasiyette 0 ise modelin tahmin işlevinin yüksek performansta olduğu ve modelde sıfır hata olduğu kabul edilir. Kök ortalama karesel hata değeri, büyük değerdeki hataları daha fazla fark edilebilme durumuna sahiptir ki bu durum veri zehirlemesi ve çekişmeli yapay sinir işe yarayabilir.[26]

$$\text{OMYH} = \sqrt{\frac{1}{n}\sum_{j=1}^{n} e_j^2} \tag{3.8}$$

3.2.4. OMYH (Ortalama Mutlak Yüzde Hata)

Genelde zaman serileri ve regresyon tahmin işlevinin performansını belirlemek adına ortalama mutlak yüzde hata değeri bir metrik olarak kullanılır. Gerçek değerlerin içerisinde 0 (sıfır) değeri veya değerleri olmaması gerekir, bunun aksi bir durum oluşursa yani 0 (sıfır) değeri ile bölünme durumu olursa işlev tanımsız ifade içereceğinden istatistiksel olarak anlamlı olmaz. [26]

$$\text{OMYH}=\frac{100}{n}\sum_{j}^{n}\frac{|e_j|}{|A_j|} \tag{3.9}$$

Yapay zekâ ve derin öğrenme modellerinde kullanılan belli başlı zaman serisi tahmin yöntemlerinin bazıları aşağıdaki gibidir:

- OOM,
- BOHODDM,
- MBOHODDM,
- Prophet

Bu işlevlerin içinde daha çok finansal analizde kullanılan Prophet fonksiyonu Meta şirketinin yazılımcılarının 2017 yılında oluşturduğu bir zaman serisi tahmin fonksiyonudur. Tez çalışmamızda MNIST veri setinin sınıflandırılması için oluşturulan ESA modelinin en fazla değişim gösteren ağırlıklarının geçmiş 10 adımlık pencerelerde eğitim verisi olarak alıp kullanılarak yeni tahmin verileri elde eden ana fonksiyon olarak kullanılacağı için detaylı bir matematiksel inceleme yapılacaktır.

3.2.5 Prophet Zaman Serisi Tahmin Fonksiyonu

Zaman serisi verileriyle çalışmak zor, sabır gerektiren işlerdir ve modelleri oluşturan çeşitli algoritmalar oldukça titiz ve ayarlanması zor olabilir. Bu özellikle birden çok mevsime sahip verilerle çalışıyorsanız karşılaşılabilecek bir durumdur. Bütün bunlara Ek olarak, BOHOM ve MBOHODDM gibi geleneksel zaman serisi modelleri, durağanlık ve eşit aralıklı değerler gibi birçok katı veri gereksinimine sahiptir. Uzun-Kısa Süreli Belleğe Sahip Yinelenen Sinir Ağları gibi diğer zaman serisi modelleri, sinir ağı mimarisi hakkında önemli bilgi altyapısına, oldukça karmaşık ve çalışmak zor olabilir. Bu yüzden, ortalama bir veri analisti için, zaman serisi analizine girişte yüksek bir engel vardır. Bu nedenle 2017'de, Facebook veri bilimcisi olan birkaç araştırmacı, Facebook Prophet fonksiyonunun açık kaynaklı projesini tanıtan ve her yerdeki veri analistlerine ve veri bilimcilere hızlı, güçlü ve erişilebilir zaman serisi modellemesi sağlayan "Ölçekte Tahmin" adlı bir makale yayınladı.[27]

Facebook Prophet, bazı yeni kavramlarla birkaç eski fikri kullanan zaman serisi modelleri oluşturmak için açık kaynaklı bir algoritmadır. Birden çok mevsime sahip olan ve diğer algoritmaların yukarıdaki dezavantajlarından bazılarıyla karşılaşmayan zaman serilerini modellemede özellikle iyidir. Aslında temel olarak zamanın üç işlevi artı bir hata teriminin toplamı ile ifade edilir:

$$y(t) = g(t) + h(t) + s(t) + \varepsilon_t \quad (3.10)$$

Büyüme $g(t)$ ile,

Mevsimsellik $s(t)$ ile,

Tatiller $h(t)$ ile,

Hata ε_t ile ifade edilir.

3.2.5.1. Büyüme Fonksiyonu ve değişim noktaları

Büyüme fonksiyonu, verilerin genel eğilimini modeller. Eski fikirler, doğrusal ve lojistik işlevler hakkında temel bilgiye sahip olan herkese aşina olmalıdır. Prophet fonksiyonuna dahil edilen yeni fikir, büyüme trendinin verilerin her noktasında mevcut olabileceği veya Prophet fonksiyonunun "değişim noktaları" dediği şekilde değiştirilebileceğidir. Değişim noktaları, verilerin yön değiştirdiği anlardır. Üstelik, büyüme işlevini değiştirirken değişim noktalarının sahip olduğu gücü ve otomatik değişiklik noktası algılamasında dikkate alınan veri miktarını da ayarlayabilirsiniz.

Büyüme işlevinin üç ana seçeneği vardır:

- **Doğrusal Büyüme:** Bu, Prophet için varsayılan ayardır. Değişim noktaları arasında farklı eğimlere sahip bir dizi parçalı doğrusal denklem kullanır. Doğrusal büyüme kullanıldığında, büyüme terimi y = mx + b klasiğine benzer görünecektir, ancak eğim (m) ve ofset (b) değişken olup her değişim noktasında değeri değiştirecektir.
- **Lojistik Büyüme:** Bu ayar, zaman serinizin, modellemekte olduğunuz değerlerin doygun hale geldiği ve maksimum veya minimum değeri geçemediği (taşıma kapasitesini düşünün) bir üst sınırı veya tabanı varsa yararlıdır. Lojistik büyüme kullanıldığında, büyüme terimi, lojistik eğri için tipik bir denkleme benzer görünecektir (aşağıya bakınız), ancak taşıma kapasitesi (C) zamanın bir fonksiyonu olarak ve büyüme oranı (k) ve dengeleme (m) değişkendir ve her değişim noktasında değeri değiştirecektir.

$$g(t) = \frac{C(t)}{1 + x^{-k(t-m)}} \quad (3.11)$$

- **Düz(lineer):** Son olarak, zaman içinde büyüme olmadığında sabit bir eğilim seçebilirsiniz (ancak yine de mevsimsellik olabilir). Düz olarak ayarlanırsa, büyüme işlevi sabit bir değer olacaktır. [27],[28]

3.2.5.2. Mevsimsellik Fonksiyonu

Mevsimsellik fonksiyonu, zamanın bir işlevi olarak aslında basit bir Fourier Serisidir. Fourier Serisine aşina değilseniz, onu düşünmenin kolay bir yolu, birbirini izleyen birçok sinüs ve kosinüsün toplamıdır. Her sinüs ve kosinüs terimi bir katsayı ile çarpılır.

$$s(t) = \sum_{n=1}^{\mathbb{N}} a_n \cos\left(\frac{2\pi nt}{P}\right) + b_n \sin\left(\frac{2\pi nt}{P}\right)) \quad (3.12)$$

Denklem 3.12'de t=zaman, n=iterasyon, N=döngü sonu değeri, P=periyot a_n ve b_n keyfi olarak belirlenen sabitler dizisini ifade eder.

Bu toplam, hemen hemen her eğriye veya Prophet fonksiyonu kullanılması, verilerimizdeki mevsimselliğe (döngüsel model) yaklaşabilir. Aşağıdaki formülde mevsimselliği hesaplayabiliriz:

3.2.5.3. Tatil/Etkinlik Fonksiyonu

Tatil işlevi, bir tatil veya büyük bir olay (pandemi, savaş vb.) tahmini değiştirdiğinde Prophet fonksiyonun tahmini ayarlamasını sağlar. Tarihlerin listesini baz alır (Amerika Birleşik Devletleri'nin tatillerinin belirli tarihleri vardır veya kullanıcı kendi tarihlerinin tanımlayabilir) ve tahminde her tarih mevcut olduğunda, tahmindeki geçmiş verilere dayalı olarak büyüme ve mevsimsellik terimlerinden değer ekler veya tahminden değer çıkarır. Özellikle borsalarda tatil günleri ve ulusal para birimlerinin değerlerinin tahminleri bu etkinlik işleviyle ilgili tahminin doğruluğunu arttırabilecek ayarlamalar yapabilir.

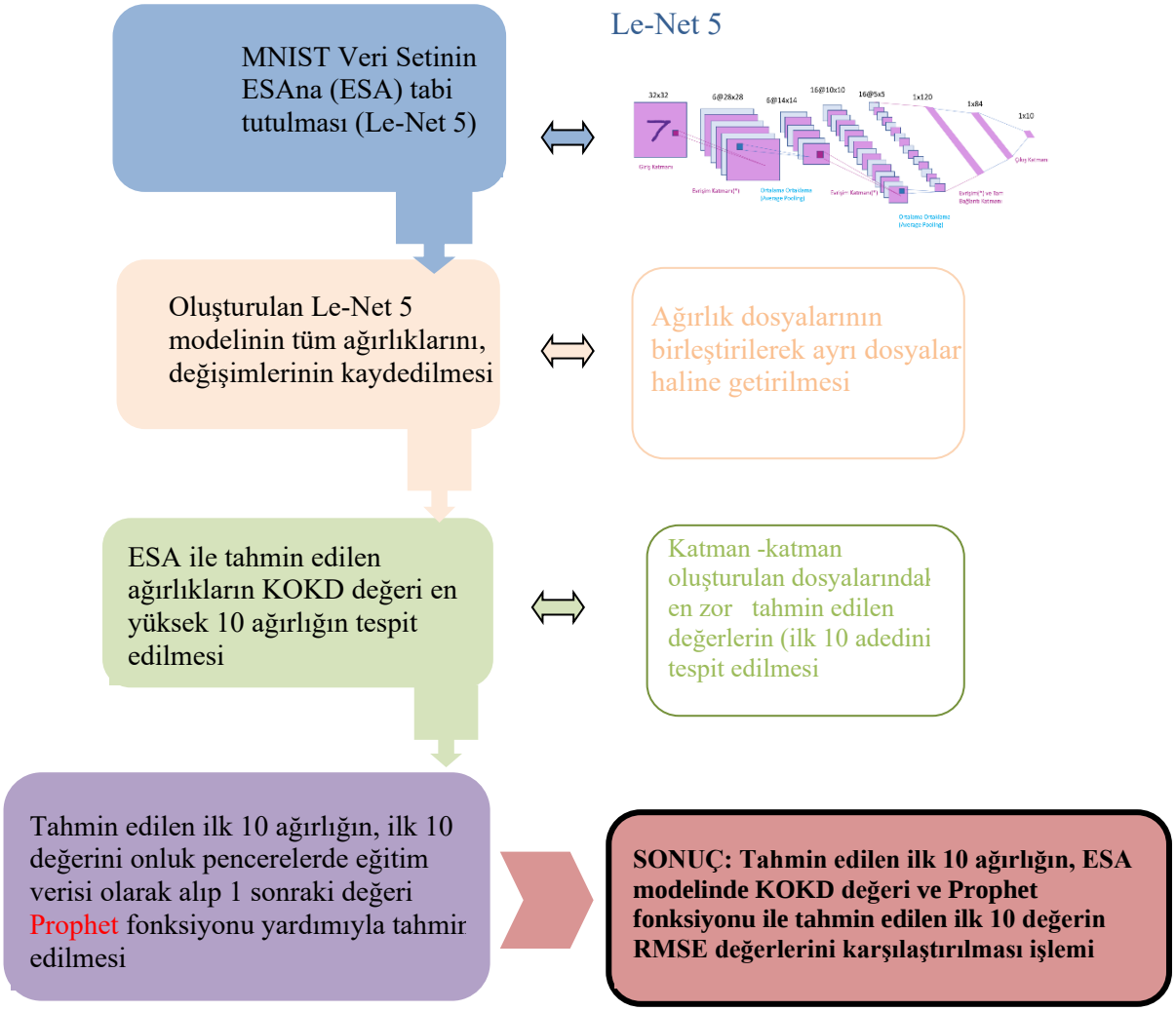
Facebook Prophet fonksiyonunun daha fazla derinliklerini keşfetmek için, Python üzerinde tez çalışmamızda yer alan uygulamamızı inceleyeceğiz.[28]

3.3. MNIST Veri Setinin Sınıflandırılması için Oluşturan ESA Modelinin Tüm Ağırlıklarının Kaydedilmesi

Çalışmanın ana süreci aşağıdaki Şekil 3.2'de belirtilmiştir. El yazısıyla yazılan rakamların tanıma işlemi için MNIST veri seti yukarıda oluşturduğumuz Bölüm 2 içerisinde literatür taramasında yer alan Le-Net 5 ağına tabi tutularak tüm katmanların ağırlıkları kaydedilecektir.

İlgili tüm algoritma kodlarına https://github.com/cagdaskaplan/DSML adresinden erişilebilir. Her katman için 100 evre)200 betik-yığın için toplamda 20.000 adet ağırlık değeri olacağı açıktır. Modeli oluşturacağımız algoritma Le-Net-5.py adlı dosya içeriğinde olacaktır.

Ekler kısmında belirtilmiş olan Lenet-5.py ismiyle belirtilen python algoritmasıyla MNIST verilerini bir ESA modelinden geçirip bütün ağırlıkları yığın (batch) başına düşen ağırlıkları ve modeli kaydetmiş oluyoruz. Bu kaydetme işlemi Python programlama dilinde hazırlanmış bir kod ile yapılmıştır.



Şekil 3. 2. El Yazısıyla Yazılan Rakamların Algılanmasında Facebook Prophet Fonksiyonu Etkisinin Araştırılması Süreci

Kaydedilen dosyalardan tüm ağırlıkları teker teker ayıklamak ve katman katman dosyalara hem bias ve kernel ağırlıkları ayıklamak için aşağıdaki kod parçası gerekecektir. Ancak bu işlemi yapabilmek veri çerçevelerinin için tüm bias ve kernel katmanlarının boyutlarını bilmek gerekli olacaktır. Bu boyutlar Ek-1 içerisinde açıklanmış olup aynı zamanda aşağıdaki Çizelge 3.4'teki gibidir:

Çizelge 3.4. Lenet-5 modelindeki katmanların bias-kernel boyutları

Katman Adı	Boyutu
conv2d kernel	**20000,150**
conv2d bias	**20000,6**
conv2d_1 kernel	**20000,2400**
conv2d_1 bias	**20000,16**
conv2d_2 kernel	**20000,48000**
conv2d_2 bias	**20000,120**
dense kernel	**20000,10080**
dense bias	**20000,84**
dense_1 kernel	**20000,840**
dense_1 bias	**20000,10**

3.3.1 ESA Modelinde Aralıklı Kategorik Çapraz Entropi Yöntemiyle Maliyet Fonksiyonu Değeri Azaltma ve Olasılıksal Dereceli Azalma Yöntemiyle En İyileme İşlemi (Optimizasyon)

Sistemde ağırlıklarını kaydettiğimiz ana ESA modelimizin maliyet(modelde loss olarak belirtilmiş) fonksiyonunu azaltmak için seçilen metot aralıklı kategorik çapraz entropi yöntemi ve en iyileme işlemi için seçilmiş olan yöntem olasılıksal(stokastik) dereceli azalma geriye yayılım yöntemidir. Aşağıdaki kod satırında SGD ile belirtilmiş olup modeli en iyi durma getirme yöntemlerinden biridir.

```
model.compile(loss='sparse_categorical_crossentropy', optimizer='SGD',
```

$$CCE(p,t) = -\sum_{c=1}^{c} t_{o,c} \log(p_o, c) \qquad (3.13)$$

Denklem 3.13'te görüldüğü üzere modelin maliyet fonksiyonunu değerini en aza indirmek için kategorik çapraz entropi hesaplamasının temel formülü verilmiştir. Bu hesaplama aslında vektörel bir işlemi ifade eder. Hesaplamadaki tahmin sınıf hedefleri tam sayı olduğunda parse terimi eklenir ve aralıklı kategorik çapraz entropi ismini alır. p: olayın gerçekleşme olasılığını ifade eder.

Dereceli azalma bir işlevin minimum değerini bulmak için 1. dereceden tekrarlamalı en iyileme algoritmasıdır. Eğimin iniş yönünü kullanıp yerelde en düşük değerdeki noktayı tespit amacıyla, kabul edilen noktada işlevin inişinin negatif yönüyle orantılı adımlarla işlem yapılır. Bu en iyileme algoritması en dik iniş şeklinde de ifade edilebilir.[29]

Dereceli azalma, derin öğrenme disiplininde revaçta bir metot olmakla birlikte derin öğrenmenin asıl hedeflerinden biri eğitim için kullandığı veri dikkate alınırsa ,en yüksek performansta doğruluğu yüksek, en az hata oranına sahip bir model elde etmektir. Bu metot maliyet işlevinin değerini en düşük değerde tutarak en az hata oranını elde etmek için kullanılır.[30]

ESA modelinin en iyileme işlemi için seçilen olasılıksal dereceli azalma geriye yayılım yöntemidir. Seçilmiş olan yöntemin hesaplama tekniği denklem 3.14'ten elde edilir.

$$\theta = \theta - \eta \nabla_\theta J\left(\theta_j; x^{(i)}, y^{(i)}\right) \tag{3.14}$$

θ: model parametresi (sürekli güncellenen)

η: (lokal minimuma erişebilmek içni adım sayısını belirleyen) öğrenme oranı

$\nabla_\theta J\left(\theta_j; x^{(i)}, y^{(i)}\right)$: zıt yönde dereceli değişen(azalan) maliyet fonksiyonu

$x^{(i)}$: eğitim örneklemi

$y^{(i)}$: $etiket, çıkış\ değeri$

İlgili algoritmanın avantajları ve dezavantajları aşağıda sıralanmıştır:

Olasılıksal dereceli azalma algoritmasının avantajları:

- Verimlilik
- Uygulamadaki kolaylığı

Olasılıksal dereceli azalma algoritmasının dezavantajları ise:

- Olasılıksal dereceli azalma düzenli hale getirmek için kullanılan parametresi ve tekrarlama sayısı vb. hiper parametrelerin kullanıcı tarafından girilmesi gerekebilir.
- Olasılıksal dereceli azalma nitelik ölçekleme hususunda hassastır.[30]

ODA ile ilgili önemli bir tespit, veri setini modele sunuş gösteriş biçimi çok önemlidir. Veriler belirli bir düzende, anlamlandırabilen düzeyde sunulursa metodun derecesini bozabilir. Bu durum zayıf büyük sayılar kanuna göre yakınsamaya sebep olur. Bu durumu engellemek için her eğitim süreci öncesinde rassal karıştırmalar yapılmalıdır. [30]

3.3.2. ESA Modelinde Elde Edilen Ağırlıkların Dosyalara Katman Katman Kaydedilmesi

Ekte belirtilen conv2d_bias.py adlı algoritmayı 2 boyutlu evrişim katmanının bias ağırlıklarının değişimlerini bir dosyaya kaydetmek için aşağıdaki kod hazırlanmıştır.

Bu kodu sırayla içeriğindeki layers = ["conv2d"] parametresini değişken kabul edip katman ismine göre sürekli değiştirerek her bir katmanı için sırayla hem bias hem kernel ağırlıkları teker teker çalıştırmamız gerekmektedir. Kod çalıştırıldığında sistemde bir önceki kodla oluşturduğumuz dosyalardan ağırlıkların seçimi yaparak bu ağırlıkların her batch ve epoch için nasıl değiştiğini ardı sıra ekleyerek kaydetmektedir.

Bu kodu çalıştırırken dikkat edeceğimiz en önemli husus katmanların boyutlarını ilgili satıra doğru yazmak olacaktır. Yoksa koda çalıştırılırken ilgili ağırlığın bulunduğu katmanın boyutunu doğru algılamayıp kodun çalıştırılmasını hata vererek durduracaktır.

Kodu sırayla layers = ["conv2d"], layers = ["conv2d_1"], layers = ["conv2d_2"], layers = ["dense"], layers = ["dense_1"] değerleri için

Ayrıca Lay=sample.get(layer).get(layer).get("bias:0") satırı değiştirip bir de kernel için birer birer çalıştırarak tüm ağırlıkların değişim değerlerini bir dosya içerisine kaydedilebilir.

3.3.3. ESA Modelinde elde edilen ağırlıkların en fazla değişen ilk 10 ağırlığın Kök Ortalama Karesel değişim (KOKD) ile tespiti

Matematiksel olarak kök ortalama karesel değişim**,** kök ortalama karesel hatadan hiçbir farkı yoktur. Denklem 3.9'dan tek farkı kullanılan son terimin 'hata' yerine 'hata' olarak kullanılmasıdır. Zaman serilerinde KOKH değeri,2 boyutlu ESA modellerinde ise KOKD değeri performans değerlendirilmesinde kullanılır. Kullanıldığı yere göre ismi değişir. Burada bir ağırlık değişimi olduğu için değişim tabiri kullanılmıştır. Bu performans belirleme metodunu kullanmamızın sebebi ESA modeli hangi ağırlığı tahmin ederken zorlanıyor ve acaba bu tahminler başka metotlarla nasıl tahmin ediliyor? (Tıpkı yapılan çalışmada da değineceğimiz Prophet fonksiyonuyla bu zor tahmin edilen ağrılıklar nasıl tahmin ediliyor?)

$$\text{KOKD}= \sqrt{\frac{1}{n}\sum_{j=1}^{n} e_j^2} \tag{3.15}$$

Ekte içeriği verilen Conv2d_rmsd.py adlı algoritma parçasıyla tüm ağırlıkların katman katman ayrı ayrı kaydedildiği dosyaların içerisinde yukarıdaki formüldeki hesabı yapıp outrmsd adlı dosyaya kaydedip bize bu dosya içerisindeki denklem 3.15'teki ifade edilen matematiksel işlemi kod içerisinde yaparak tahmini zor olan ağırlığı bulmamızı sağlayacaktır.

Tüm bu işlemler yapıldıktan sonra MNIST veri setinin yukarıda oluşturulan ESA modeliyle sınıflandırılmasına ait ağırlıkların KOKD değişimleri aşağıdaki Çizelge 3.5'te verilmiştir:

Çizelge 3.5. MNIST Veri Seti ESA Modeli ile Sınıflandırılmasında KOKD değeri en fazla olan ilk 10 ağırlık bilgileri

ESA Katmanı Ağırlık Numarası	Ağırlık Batch Numarası	Ağırlık Epoch Numarası	ESA KOKD değerleri
Conv2d:Kernel:149	13762	69	**0.150626049**
Conv2d:Kernel:143	8721	44	**0.148121276**
Conv2d:Kernel:147	13762	69	**0.143861903**
Conv2d:Kernel:146	9311	47	**0.119593173**
Conv2d:Kernel:137	4084	21	**0.118192931**
Dense_1:Kernel:342	9311	47	**0.117505488**
Conv2d:Kernel:141	8721	44	**0.115435119**
Conv2d:Kernel:117	4723	24	**0.110841657**
Conv2d:Kernel:119	4084	21	**0.108069505**
Conv2d:Kernel:50	9311	47	**0.106033022**

3.4. KOKD değeri en fazla olan ilk 10 ağırlık değerinin ağırlık değişimleri eğitim girdisi kullanılarak ağırlıkların yeniden Prophet fonksiyonuyla tahmin edilmesi

Çalışmamızın ana hattını oluşturan Facebook Prophet fonksiyonuyla elde ettiğimiz ilk 10 ağırlık değerinin tahmini bölümüdür. Bu metot için ESA modeline uygun parametreler seçtik. Ayrıca olup Prophet fonksiyonun modele ilişkin seçilmiş parametreleri de değinilecektir.

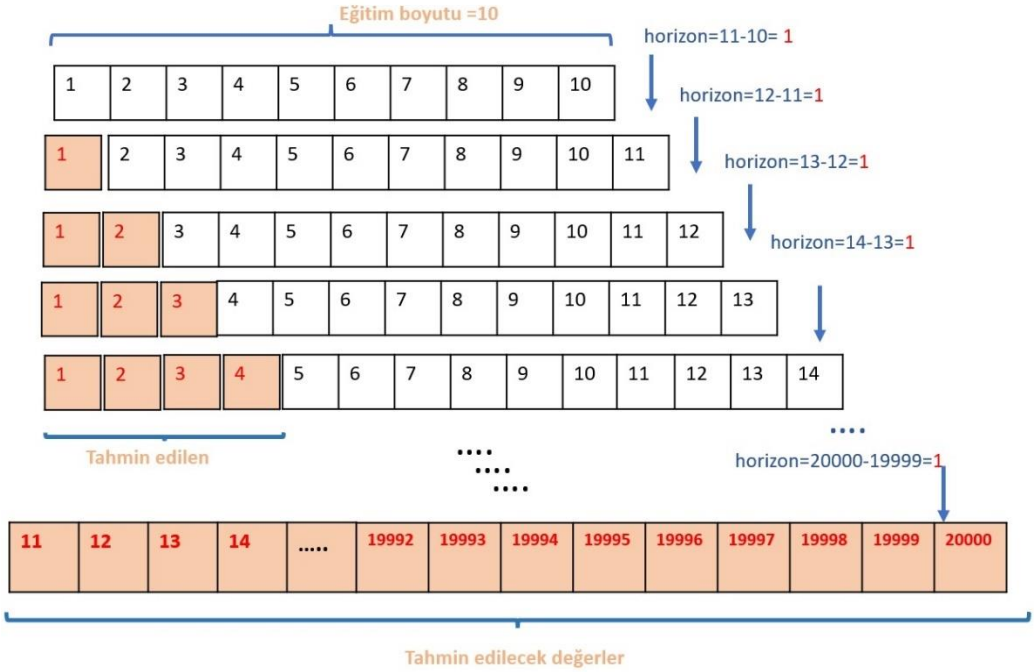
Daha öncesinde elde ettiğimiz 10 adet veri dosyasını Prophet Fonksiyonunu anlayabileceği formata getirilmiş olması gerekiyor. Prophet fonksiyonun veri giriş dosyası 1 adlı dosyasında 10 adlı dosyasına kadar tüm dosyalarda tarih ve sütun formatlarda Çizelge 3.6'daki gibi olmalıdır. Aksi takdirde Prophet fonksiyonu veriyi algılayamayacaktır. [31]

Çizelge 3.6. Prophet Fonksiyonu standart giriş verisi dosyası

İterasyon	ds(zaman)	y(mevcut Değer)
0	2007-12-10	9.590761
1	2007-12-11	8.519590
2	2007-12-12	8.183677
3	2007-12-13	8.072467
4	2007-12-14	7.893572
...	.....	.....
19999	.....	.....

Böylelikle ekler bölümünde içeriği bulunan Prophet_1.py python kodu her ağırlık için her dosya için ayrı tahmin yapmaktadır. Toplamda 10 adet çıkış dosyamız yani out adlı dosyamız olacaktır. Prophet Fonksiyonu ilgili ağırlık tahmin ederken belli parametreler ışığında işlevinin yerine getirmektedir. Bunlardan ilki ESA modelindeki ağırlığın geçmişte kaç adet değerinin eğitime alınacağı ve hangi metodun kullanılacağı hususudur. Ağırlık ilk 10 değeri Prophet.py adlı python kodu içerisindeki data_to_fit eşitlik denkleminde Şekil 3.3'te de görülen kayan pencere metodu ile belirlenmiştir.

Böylece 20.000 değişim değeri olan ağırlık değişiminin 20000-10=19990 adet tahmin yapılması aralığın tqdm kütüphanesinin kullanıldığı for döngüsünde belirtilmiştir. Bu algoritmanın işleyiş biçimini bir çizelge ile belirttiğimize daha anlaşılır hale gelmiş olacaktır.



Şekil 3.3. Prophet Fonksiyonunun Kayan Pencere Metoduyla Kullanılması

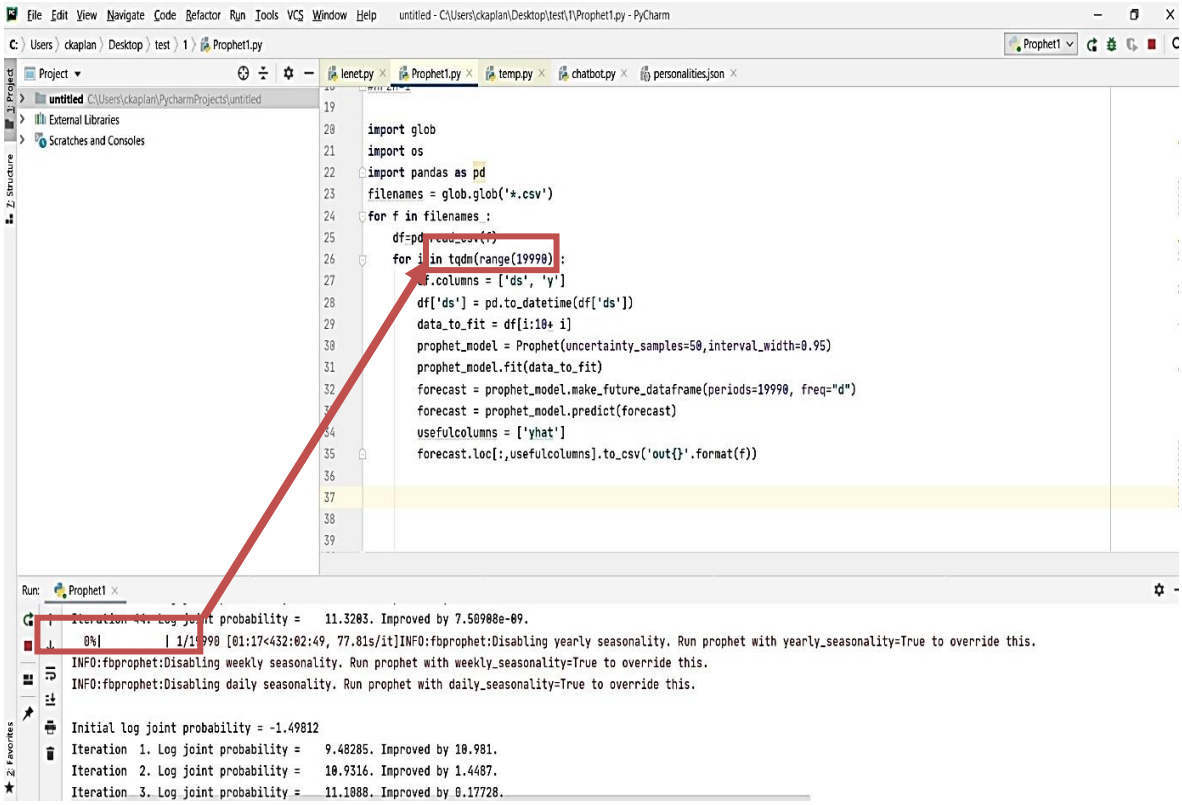
Yapay zekâ modelinin verimliğini arttıran ve metottan metoda değişen yüksek doğrulukta çalışmasını sağlayan bazı parametreler vardır.Prophet_1.py algoritmada belirtilen Prophet fonksiyonunda yer alan hiper parametreler fonksiyonun verimliliği ve doğruluğunu arttıran parametrelerdir.

Prophet(uncertainty_samples=50,interval_width=0.95) (3.16)

Bunlardan ilki uncertainty_samples=50 parametresi belirsizlik örnekleme sayısı aşırı öğrenmeden problemini en aza indiren ve geçmişte eğitilen verilerin hızlıca tahmin yapmasını gerçekleştiren bir işlevsel hiper parametredir.50 değeri ise modele göre değer verilerek en yüksek doğrulukta olan değer seçilebilir.

Diğer bir parametre ise interval_width=0.95 parametresi Türkçesi tahmin aralığı genişliğidir. Sistemin veya modelin tahminin doğruluğunu ve büyük veri tahminlerinde zaman kazanımına yarayan bir işlevsel hiper parametredir. 0.95 değeri varsayılan değerdir. Modele göre bu değer değiştirilebilir.

Kod içerisinde yüklenmiş ir modül olan tqdm modülü de tahmin sürecinin yüzdesel olarak hangi durumda görüntülemek için kullanılmıştır. Şekil 3.4'te görüldüğü üzere phycharm ekran görüntüsünde % 0 [ ] 1/19990 şeklindedir.



Şekil 3.4. Prophet fonksiyonu çalıştırıldığında elde edilen Phycharm ekranı (tqdm)

TensorFlow, bilimsel araştırma geliştirme amaçlı daha çok büyük ölçekli veri ve grafik işlemci arabiriminin kullanıldığı derin öğrenme modellerinde kullanılan python kütüphanesidir. İnternet ağı ve internet tabanlı sanallaştırma alanlarında çalışmak isteyen yazılımcılar için uygulama programlama ara yüzleri sunmaktadır. Derin öğrenme modelleri tasarımında bilimsel araştırmalar için Google firmasının geliştirmiş olduğu bir esnek bir sisteme sahip modüldür.

Görüntü tespiti, doğal dil işleme, bilgisayar görüsü, otonom, bilgi alma, coğrafi bilgi sistemleri, tıbbi tanılar ve farmakoloji dallarını da kapsayan birden çok bilişim disiplini ve ilgili dallarda halen yaygın biçimde geliştirilmekte ve kullanılmaktadır.

Keras python programlama dili tabanlı açık kaynak kodlu en yaygın kullanılan gelişmiş yapay sinir ağı kütüphanelerinden birisidir. Derin öğrenme modelleriyle ilgili çabuk sonuç alınabilmesi adına yazılmış bir algoritmalar modülüdür. Avantajı esnetilebilir, kullanım kolaylığı sağlamasıdır. Birçok alanda çalışan yazılım araştırmacılar tarafın ilk versiyonu 2015 yılı Mart ayı içerisinde yayınlanmıştır. Tensorflow Microsoft Bilişsel Araç(CNTK) ve Theano sistemleri üzerinde çalışabilmektedir.

4. BULGULAR ve TARTIŞMA

4.1. Prophet Fonksiyonu ile Tahmin Edilen Ağırlıkların Tahmin Sonuçları

Ekler bölümünde belirtilmiş olan Prophet.py adlı algoritmayla MNIST veri setindeki görüntülerin sınıflandırılması için oluşturmuş olduğumuz ESA modelinin ağrılıklarının en zor tahmin edilen ilk 10 tanesi seçilip Prophet zaman serisi tahmin fonksiyonuyla kayan pencere metoduyla ile tahmin edildi. Elde edilen sonuçlar Çizelge 4.1'de görüldüğü gibidir.

Çizelge 4. 1. Prophet Fonksiyonuyla tahmin edilen ağırlıkların Kök Ortalama Karesel Hata Değerleri

ESA Katmanı Ağırlık Numarası	Ağırlık Batch Numarası	Ağırlık Epoch Numarası	FB Prophet KOKH değerleri
Conv2d:Kernel:149	13762	69	0.015622065938781788
Conv2d:Kernel:143	8721	44	0.0033084461177362813
Conv2d:Kernel:147	13762	69	0.443781812824668
Conv2d:Kernel:146	9311	47	1.0642281583960937
Conv2d:Kernel:137	4084	21	2.420552960865531
Dense_1:Kernel:342	9311	47	0.011007323066514143
Conv2d:Kernel:141	8721	44	0.010140846014693325
Conv2d:Kernel:117	4723	24	0.010534252038338807
Conv2d:Kernel:119	4084	21	0.010022087659939326
Conv2d:Kernel:50	9311	47	0.009155961952817865

Çizelge 4. 1.'de görüldüğü üzere elde edilen kök ortalama karesel hata değerleri Conv2d katmanında 137.'nci ağırlık,146'ncı ,147'nci ağırlıklar oldukça yüksek tahmin edilmiştir.Çizelgede yer alan diğer satırlardaki ağırlık numaralarına ait tahminlerdeki hata değeri düşük çıkmıştır. Hata değeri yüksek çıkan değerler snıflandırmada ayırt edici noktalar olarak belirlenebilir.

4.2. Prophet Fonksiyonu ve Evrişimsel Sinir Ağları Kök Ortalama Karesel Hata (KOKH) Kök Ortalama Karesel Değişim (KOKD) değerlerinin karşılaştırılması

Çizelge 4.2'de de görüleceği üzere ESA Modeli ile Facebook Prophet KOKH &KOKD değerleri görülmektedir. Çoğu ağırlık değeri tahmininde Facebook Prophet fonksiyonun yüksek doğruluk oranından tespit ettiği görülmektedir. Zaman-verim ile ilgili olarak herhangi bir değerlendirme yapılmamıştır. Bu çalışmadaki karşılaştırma veri madenciliğine yön vermesi adına bir derin sinir ağının tahmin edilmiş ağırlıklarının zaman serisi fonksiyonuyla tahmin edildiğinde hangi tahmin yönteminin daha başarılı aşağıdaki yeşil renkte belirtilen sonuçlar küçük KOKH veya KOKD göre teorik anlamada uygulanabilirliği hakkında fikir veriyor.

Çizelge 4. 2. Prophet Fonksiyonu ve Evrişimsel Sinir Ağları KOKH&KOKD değerlerinin karşılaştırılması

ESA Katmanı Ağırlık Numarası	Ağırlık Batch Numarası	Ağırlık Epoch Numarası	ESA KOKD değerleri	FB Prophet KOKH değerleri
Conv2d:Kernel:149	13762	69	0.150626049	0.015622
Conv2d:Kernel:143	8721	44	0.148121276	0.00330
Conv2d:Kernel:147	13762	69	0.143861903	0.44378
Conv2d:Kernel:146	9311	47	0.119593173	1.06422
Conv2d:Kernel:137	4084	21	0.118192931	2.42055
Dense_1:Kernel:342	9311	47	0.117505488	0.01100
Conv2d:Kernel:141	8721	44	0.115435119	0.01014
Conv2d:Kernel:117	4723	24	0.110841657	0.01053
Conv2d:Kernel:119	4084	21	0.108069505	0.01002
Conv2d:Kernel:50	9311	47	0.106033022	0.00915

Tahmin sürecinde tamamen bir veri madenciliği süreci ile fonksiyonun tahmin gücü karşılaştırıldığı için tahmin zamanı-yani geçen kod çalıştırma süresi karşılaştırması yapılmamıştır. Çünkü Prophet fonksiyonu derinlemesine eğilim analizi yaptığı için modern bilgisayarlarda 20.000 satırlık bir tarih aralığı çok uzun bir çalışma süresi gerektirir.

4.3.Veri Zehirleme İşlemi

Tez çalışmasının başlangıç kısmında da ifade edilen bilgisayar görüsü alanında kullanılan veri zehirlemesi teriminin tanımlamasını yapacak olursak; elde edilmiş verileri sınıflandırma adına ilgili giriş veri setine yapay veriler ekleyerek tahmin edilecek değer yanıltacak bir veri eklemek olarak ifade edilebilir. Örneğin MNIST veri setinde 3 rakamını 8 gibi algılamasını sağlayarak ilgili ESA tahmin modelinin tahmin işleminde ilgili rakamın karakteristik ayırt edici ağırlıklarını tespit etmek için tahmin hatası yaptığı bölgeleri veya ağırlıkları tespit edebiliriz. Sınıflandırma problemlerinde ayırt edici özellikleri tespit etmek amacıyla ilgili veriye uygulanan verinin tanınmasının önleyen yapay bozma işlemini de ifade eder.

5. SONUÇLAR

Bu tez çalışmasında, MNIST veri setinin görüntülerinin bir ESA modeli olan Lenet-5 ile sınıflandırılmış olup bu sınıflandırılmadaki tüm ağırlıklar kaydedilmiştir. Kaydedilen bu ağırlıkların istatistiki bir yöntem olan KOKD değerleri baz alınarak tahmin edilmesi zor olan ilk 10 ağırlık tespit edilmiştir. Tespit edilen ilk 10 ağırlık gelişmiş bir zaman serisi tahmin yöntemi olan açık kaynak kodlu Facebook Prophet ile tahmin edilmiştir.

Aslında klasik yapay zekâ ve derin öğrenme modellerinin ağırlıklarının eğitim girdisi olarak alıp zaman serisi ile tahmin edip performans karşılaştırılması yapmak derin öğrenme disiplinlerine farklı bir bakış açışı getirecektir. Çizelge 4.2'den de anlaşılacağı üzere ESA modelinin zorlandığı ağırlıkların tahmininde Prophet fonksiyonun tahminlerindeki teorik kök ortalama karesel hata değeri daha az ve doğruluğu daha fazladır.

ESA modelinin tahmin performansının daha yüksek olduğu ağırlıkların tahminin zor olduğu bu durumun görüntü verisi içerisindeki fark edilebilir ağırlıkların gösterimi belirlemek için oldukça kolaylaştırıcı bir tekniğin temelini oluşturabilir. Bu bahsedilen verilere veya bulunduğu katmanın o bölgesine veri zehirleme işlemi yapılabilir.

Özü itibari ile çalışma sonucunda çekişmeli derin öğrenme disiplininde yer alan veri zehirlemesi temeli için derin öğrenme tahminlerinde gelişmiş zaman serisi fonksiyonlarının tahmin performansı değerlendirmesi için çok önemli bir altyapı oluşturacaktır. Ayrıca çalışma yapay zekâ ve derin öğrenme modellerinin zayıf noktalarının tespitinde kullanılabilir.

Yapılan bu çalışmanın ileri görüntü sınıflandırma metotlarına yeni bir anlayış katacağını umuyoruz. Bu çalışma otonom sürüş yapan bir aracın kamerasının veri zehirlemesi teknikleriyle görüntüyü dış koşullardan etkilenen yoldaki trafik levhasını veri zehirlemesi uygulanmış gibi algılayıp tanımasını sağlamada kullanılabilir.

6.KAYNAKLAR

[1] ZHANG, G. Peter. Time series forecasting using a hybrid ARIMA and neural network model. Neurocomputing, 2003, 50: 159-175.

[2] Afrasiabi, M., khotanlou, H. & Mansoorizadeh, M. DTW-CNN: time series-based human interaction prediction in videos using CNN-extracted features. Vis Comput 36, 1127–1139 (2020). https://doi.org/10.1007/s00371-019-01722-6

[3] Multivariate Time-Series Classification Using the Hidden-Unit Logistic Model Pei, Wenjie; Dibeklioglu, Hamdi; Tax, David M.J.; van der Maaten, Laurens DOI 10.1109/TNNLS.2017.2651018 Publication date 2018

[4] A. Lorincz, L. A. Jeni, Z. Szabó, J. F. Cohn and T. Kanade, "Emotional Expression Classification Using Time-Series Kernels," *2013 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2013, pp. 889-895, doi: 10.1109/CVPRW.2013.131.

[5] Alan. M. Turing, I.—Computing Machinery And Intelligence, *Mind*, Volume LIX, Issue 236, October 1950, Pages 433–460, https://doi.org/10.1093/mind/LIX.236.433

[6] Radu Raicea,''Want to know how Deep Learning works?'' https://medium.com/ [Çevrimiçi] Available at: https://medium.com/free-code-camp/want-to-know-how-deep-learning-works-heres-a-quick-guide-for-everyone-1aedeca88076 [Erişim: 03-Ocak-2022].

[7] I. Goodfellow, Y. Bengio, ve A. Courville, "Deep Learning". MIT Press, 2016

[8] Song Ho Ahn (안성호),"Convolution'' https://songho.ca/ [Çevrimiçi] Available at:https://songho.ca/dsp/convolution/convolution2d_example.html [Erişim:03-Oca-2021].

[9] CS231n: Convolutional Neural Networks for Visual Recognition http://www.stanford.edu/ [Çevrimiçi] Available at:http://cs231n.stanford.edu [Erişim: 21-Ara-2020].

[10] F. Beşer, M. A. Kizrak, B. Bolat and T. Yildirim, "Recognition of sign language using capsule networks," 2018 26th Signal Processing and Communications Applications Conference (SIU), 2018, pp. 1-4, doi: 10.1109/SIU.2018.8404385.

[11] CS230n: Evrişimsel Sinir Ağları el Kitabı http://www.stanford.edu/ [Çevrimiçi] Available at: https://stanford.edu/~shervine/l/tr/teaching/cs-230/cheatsheet-convolutional-neural-networks#layer [Erişim: 18-Aralık-2021].

[12] LECUN, Yann, et al. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 1998, 86.11: 2278-2324.

[13] C. Cortes, C. J. Burges ve Y. LeCun, "MNIST handwritten digit database",[Çevrimiçi]:http://yann.lecun.com/exdb/mnist/.[Erişim:15-Ağu-2018].

[14] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, november 1998

[15] Ayyüce Kızrak, "Comparison of Activation Functions for Deep Neural Networks", towardsdatascience.com[Çevrimiçi].Available at: https://towardsdatascience.com/comparison-of-activation-functions-for-deep-neural-networks-.[Erişim: 27-Tem-2020].

[16] Dierbach, C.,"Introduction to Computer Science Using Python:A Computational Problem-solving Focus", Wiley Publishing, (2012).

[17] Uğur Ayvaz, Adil Çoban, Hüseyin Gürüler, Musa Peker Python Dilinin Öznitelikleri, Programlama Eğitiminde ve Yazılım dünyasındaki Yeri AB akademik Bilişim Konferansı,2016

[18] Pycharm,Wikipedia [Çevrimiçi].Availableat:https://wikipedia.org/Pycharm/[Erişim: 02-Dec-2020].

[19] Cem Kadılar, SPSS Uygulamalı Zaman Serileri Analizine Giriş,2005

[20] Spiegel, Murray R. "Larry J." Stephens, İstatistik, (çev. Ed. Alptekin Esin ve Salih Çelebioğlu) Nobel Yayın Dağıtım, Ankara (1999)

[21] Erkan Şirin ," Varyans, Kovaryans ve Standart Sapma Nedir?", veribilimiokulu.com [Çevrimiçi]. Available at:https://www.veribilimiokulu.com/varyans-kovaryans-ve-standart-sapma-nedir-orneklerle-aciklama[Erişim: 23-May-2020].

[22] Neyran Orhunbilge, Zaman Serileri Tahmin ve Fiyat Endeksleri,1999

[23] Utku Kubilay ÇINAR ,"R ile Derinlemesine Zaman Serileri", veribilimiokulu.com [Çevrimiçi].Available at: https://www.veribilimiokulu.com/zaman-serileri-geleneksel-yontemler-ile-yapay-sinir-aglarinin-karsilastirilmasi/[Erişim: 1-Haz-2020].

[24] Özek, T., "Zaman Serisi Modelleri Üzerine Bir Simülasyon Çalışması" (2010).

[25] Özmen, A., "Zaman Serisi Analizinde Box-Jenkins Yöntemi ve Banka Mevduat Tahmininde Uygulama Denemesi" (1986).

[26] Meyer, R., Krueger, D., (2005), "A Minitab Guide to Statistics", s..379. Pearson Yayınları, London, UK.

[27] MSE, RMSE, MAE, MAPE ve Diğer Metrikler, veribilimiokulu.com [Çevrimiçi]. Available at: https://veribilimcisi.com/2017/07/14/mse-rmse-mae-mape-metrikleri-nedir/[Erişim: 30-May-2020].

[28] Taylor SJ, Letham B. 2017. Forecasting at scale. PeerJ Preprints 5:e3190v2 https://doi.org/10.7287/peerj.preprints.3190v2

[29] Dereceli Azalma nedir?Gradient Desecent [Çevrimiçi].Available at https://veribilimcisi.com/2017/07/18/dereceli-azalma-nedir-gradient-descent/ /[Erişim: 30-May-2020].

[30] Olasılıksal Dereceli Azalma nedir?Stochastic Gradient Descent [Çevrimiçi]Available at https://veribilimcisi.com/2017/07/19/olasiliksal-dereceli-azalma-nedir-stochastic-gradient-descent/ /[Erişim: 01-June-2020].

[31] Facebook Prophet [Çevrimiçi] Available at https://facebook.github.io/prophet/docs/quick_start.html#python-api/ [Erişim: 30-May-2020].

7.EKLER

EK-1.MNIST Veri Setinin ESA (ESA) Modeli LeNet5.py model yapısı kod çıktısı

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 28, 28, 6)	156
average_pooling2d (AveragePo	(None, 14, 14, 6)	0
conv2d_1 (Conv2D)	(None, 10, 10, 16)	2416
average_pooling2d_1 (Average	(None, 5, 5, 16)	0
conv2d_2 (Conv2D)	(None, 1, 1, 120)	48120
flatten (Flatten)	(None, 120)	0
dense (Dense)	(None, 84)	10164
dense_1 (Dense)	(None, 10)	850

Total params: 61,706

Trainable params: 61,706

Non-trainable params: 0

EK-2. MNIST Veri Setinin ESA (ESA) Modeli LeNet5.py kod çıktısı (Evre-evre maliyet fonksiyon değeri ve doğruluk değişimi)

epoch : 0001 , loss : 1.423724 , acc : 0.766667

epoch : 0002 , loss : 0.637336 , acc : 0.893333

epoch : 0003 , loss : 0.487066 , acc : 0.896667

epoch : 0004 , loss : 0.431202 , acc : 0.903333

epoch : 0005 , loss : 0.403007 , acc : 0.906667

epoch : 0006 , loss : 0.385673 , acc : 0.916667

epoch : 0007 , loss : 0.373155 , acc : 0.923333

epoch : 0008 , loss : 0.362993 , acc : 0.926667

epoch : 0009 , loss : 0.354112 , acc : 0.936667

epoch : 0010 , loss : 0.346013 , acc : 0.936667

epoch : 0011 , loss : 0.338451 , acc : 0.936667

epoch : 0012 , loss : 0.331303 , acc : 0.940000

epoch : 0013 , loss : 0.324505 , acc : 0.943333

epoch : 0014 , loss : 0.318025 , acc : 0.943333

epoch : 0015 , loss : 0.311846 , acc : 0.943333

epoch : 0016 , loss : 0.305956 , acc : 0.943333

epoch : 0017 , loss : 0.300346 , acc : 0.943333

epoch : 0018 , loss : 0.295009 , acc : 0.943333

epoch : 0019 , loss : 0.289935 , acc : 0.946667

epoch : 0020 , loss : 0.285112 , acc : 0.946667

epoch : 0021 , loss : 0.280526 , acc : 0.946667

epoch : 0022 , loss : 0.276165 , acc : 0.946667

epoch : 0023 , loss : 0.272012 , acc : 0.946667

epoch : 0024 , loss : 0.268055 , acc : 0.946667

epoch : 0025 , loss : 0.264280 , acc : 0.946667

epoch : 0026 , loss : 0.260672 , acc : 0.946667

epoch : 0027 , loss : 0.257220 , acc : 0.950000

epoch : 0028 , loss : 0.253913 , acc : 0.950000

epoch : 0029 , loss : 0.250739 , acc : 0.950000

epoch : 0030 , loss : 0.247689 , acc : 0.953333

epoch : 0031 , loss : 0.244755 , acc : 0.953333

epoch : 0032 , loss : 0.241927 , acc : 0.953333

epoch : 0033 , loss : 0.239200 , acc : 0.953333

epoch : 0034 , loss : 0.236565 , acc : 0.953333

epoch : 0035 , loss : 0.234019 , acc : 0.953333

epoch : 0036 , loss : 0.231555 , acc : 0.953333

epoch : 0037 , loss : 0.229168 , acc : 0.956667

epoch : 0038 , loss : 0.226855 , acc : 0.956667

epoch : 0039 , loss : 0.224611 , acc : 0.956667

epoch : 0040 , loss : 0.222433 , acc : 0.960000

epoch : 0041 , loss : 0.220318 , acc : 0.960000

epoch : 0042 , loss : 0.218263 , acc : 0.960000

epoch : 0043 , loss : 0.216264 , acc : 0.960000

epoch : 0044 , loss : 0.214319 , acc : 0.960000

epoch : 0045 , loss : 0.212425 , acc : 0.960000

epoch : 0046 , loss : 0.210579 , acc : 0.960000

epoch : 0047 , loss : 0.208779 , acc : 0.960000

epoch : 0048 , loss : 0.207022 , acc : 0.960000

epoch : 0049 , loss : 0.205305 , acc : 0.963333

epoch : 0050 , loss : 0.203628 , acc : 0.966667

epoch : 0051 , loss : 0.201986 , acc : 0.966667

epoch : 0052 , loss : 0.200378 , acc : 0.966667

epoch : 0053 , loss : 0.198802 , acc : 0.966667

epoch : 0054 , loss : 0.197256 , acc : 0.966667

epoch : 0055 , loss : 0.195738 , acc : 0.970000

epoch : 0056 , loss : 0.194246 , acc : 0.970000

epoch : 0057 , loss : 0.192779 , acc : 0.970000

epoch : 0058 , loss : 0.191335 , acc : 0.970000

epoch : 0059 , loss : 0.189913 , acc : 0.970000

epoch : 0060 , loss : 0.188512 , acc : 0.970000

epoch : 0061 , loss : 0.187130 , acc : 0.970000

epoch : 0062 , loss : 0.185766 , acc : 0.970000

epoch : 0063 , loss : 0.184420 , acc : 0.970000

epoch : 0064 , loss : 0.183091 , acc : 0.970000

epoch : 0065 , loss : 0.181777 , acc : 0.970000

epoch : 0066 , loss : 0.180480 , acc : 0.970000

epoch : 0067 , loss : 0.179197 , acc : 0.970000

epoch : 0068 , loss : 0.177928 , acc : 0.970000

epoch : 0069 , loss : 0.176674 , acc : 0.970000

epoch : 0070 , loss : 0.175433 , acc : 0.970000

epoch : 0071 , loss : 0.174206 , acc : 0.970000

epoch : 0072 , loss : 0.172992 , acc : 0.970000

epoch : 0073 , loss : 0.171791 , acc : 0.970000

epoch : 0074 , loss : 0.170604 , acc : 0.970000

epoch : 0075 , loss : 0.169429 , acc : 0.970000

epoch : 0076 , loss : 0.168267 , acc : 0.970000

epoch : 0077 , loss : 0.167118 , acc : 0.970000

epoch : 0078 , loss : 0.165982 , acc : 0.970000

epoch : 0079 , loss : 0.164859 , acc : 0.970000

epoch : 0080 , loss : 0.163749 , acc : 0.973333

epoch : 0081 , loss : 0.162651 , acc : 0.973333

epoch : 0082 , loss : 0.161566 , acc : 0.973333

epoch : 0083 , loss : 0.160494 , acc : 0.973333

epoch : 0084 , loss : 0.159434 , acc : 0.973333

epoch : 0085 , loss : 0.158388 , acc : 0.973333

epoch : 0086 , loss : 0.157353 , acc : 0.973333

epoch : 0087 , loss : 0.156332 , acc : 0.973333

epoch : 0088 , loss : 0.155323 , acc : 0.973333

epoch : 0089 , loss : 0.154327 , acc : 0.973333

epoch : 0090 , loss : 0.153343 , acc : 0.973333

epoch : 0091 , loss : 0.152371 , acc : 0.973333

epoch : 0092 , loss : 0.151412 , acc : 0.973333

epoch : 0093 , loss : 0.150465 , acc : 0.973333

epoch : 0094 , loss : 0.149530 , acc : 0.973333

epoch : 0095 , loss : 0.148607 , acc : 0.976667

epoch : 0096 , loss : 0.147696 , acc : 0.976667

epoch : 0097 , loss : 0.146797 , acc : 0.976667

epoch : 0098 , loss : 0.145909 , acc : 0.976667

epoch : 0099 , loss : 0.145032 , acc : 0.976667

epoch : 0100 , loss : 0.144167 , acc : 0.976667

EK-3 Le-Net 5.py Algoritması

```
author :ckaplan
#   Modüllerin Yüklenmesi
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D,Dense,FlattenA-veragePooling2D
import os
#   Mnist Veri setinin indirilmesi
mnist = tf.keras.datasets.mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train.reshape(-1,28,28,1) / 255.0, x_test.reshape(-l,28,28,l) / 255.0
#   Modelin oluşturulması
model = Sequential()
#   C1 Evrişim Katmani
model.add(Conv2D(6, kernel_size=(5, 5), strides=(l, 1), activation='tanh',
input_shape=(28,28,l), padding='same|))
#   S2 Ortaklama Katmanı
model.add(AveragePooling2D(pool_size=(2, 2), strides=(2,2), padding='valid'))
#   C3 Evrişim Katmani
model.add(Conv2D(16, kernel_size=(5, 5), strides=(l, 1), activation='tanh',
padding='valid'))
#   S4 Ortaklama Katmanı
model.add(AveragePooling2D(pool_size=(2, 2), strides=(2,2), padding='valid'))
#   C5 Tam Bağlantı Evrişim Katmanı
model.add(Conv2D(120, kemel_size=(5, 5), strides=(l, 1), activation='tanh':
padding='valid'))
model.add(Flatten())
#   FC6 Tam Bağlantı Katmanı
model.add(Dense(84, activation='tanh'))
#Çikiş Katmani
model.add(Dense(10, activation='softmax'))
# Modelin maliyet fonksiyonunu Azaltacak hiper parametreler
model.compile(loss='sparse_categorical_crossentropy', optimizer='SGD',
metrics=['accuracy'])
model.summaryO
#ESA(ESA) Modelinin Kaydedilmesi
def order_batch(X, y, itr, size=l);
  return X[itr * size: (itr+1) * size] ,y [itr * size: (itr+1) * size]
batch_size = 300
epoeh =100
#ESA modelinin egitilmesi ve dogruluk değerlerinin ahnmasi agirliklann kaydedilmesi
for e in range(epoeh):
  for b in range(int(len(x_train)/batch_size)):
    X_batch, y_batch = order_batch(x_train,y_train,b,batch_size)
    loss, aee = model.train_on_batch(X_batch, y_batch)
    model.save_weights("model%04d%06d.h5" % (e+l,b+l))
  print("epoch : %04d , loss : %f, aee : %f" % (e+l,loss,acc))
```

EK-4 Prophet_1.py algoritması

```
author :ckaplan
import tqdm
 from numpy import eumsum, log, polyfit, sqrt, std, subtraet
from tqdm import tqdm
import pystan
import tensorflow as tf
import numpy as np
import fbprophet from fbprophet
import Prophet
import pandas as pd
from pandas import DataFrame
import time
import glob
import os
import pandas as pd filenames = glob.glob('*.csv')
for f in filenames :
        df=pd.read_csv(f)
        for i in tqdm(range(19990)) :
        df.columns = ['ds', 'y']
        df['ds'] =pd.to_datetime(df['ds']) data_to_fit - df[i:10+ i]
        prophet_model =Prophet(uncertainty_samples=50,interval_width=0.95)
        prophet_model.fit(data_to_fit)
        forecast=prophet_model.make_future_dataframe(perrods«19990, freq-"d")
        forecast = prophet_model.predict(forecast) usefulcolumns = ['yhat']
        forecast.loc[:,usefulcolumns].to_csv('out{}'.format(f))
```

EK-5 Conv2d_bias.py algoritması

```
author :ckaplan
import glob
import h5py
import itertools
import numpy as np
import pandas as pd
from tqdm import tqdm
model_list = glob.glob('*.h5')
print(len(model_list))
layers = ["conv2d"]
sample = h5py.File(model_list[0],'r+')
df_list = []
for layer in layers:
    lay = sample.get(layer).get(layer).get("bias:0")
    length = np.prod(lay.shape)
    data = np.zeros((len(model_list),length))

    columns = []
    for idx, value in np.ndenumerate(lay):
        columns.append(layer + str(idx))

    i = 0
    for model in tqdm(model_list, desc= layer):
        h = h5py.File(model,'r+')
        h5 = h.get(layer).get(layer).get("bias:0")
        j = 0
        for idx, value in np.ndenumerate(h5):
            data[i,j] = value
            j = j + 1
        i = i + 1

    df_list.append(pd.DataFrame(data[:],columns=columns))
datax=data[:,:]
ax=datax.reshape(20000,10)
af=[]
af.append(pd.DataFrame(ax))
df=pd.concat(af, axis=0, sort=False)
df.to_csv('biasconv2d.csv')

sample.close()
h.close()
```

EK-6 Conv2d_rmsd.py

```
author :ckaplan
import csv
import numpy as np
import os
import pandas as pd
window=19999
df= pd.read_csv("out.csv")
dfa=df.iloc[:,1:]
df_s=dfa.diff()**2
df_a=df_s.iloc[:-1,:].sum(0)
df_r=np.sqrt(df_a/window)
df_r.to_csv('outrmsd.csv')
```

ÖZGEÇMİŞ

ÇAĞDAŞ KAPLAN

cagdaskaplan@gmail.com

ÖĞRENİM BİLGİLERİ

Yüksek Lisans 2018-2023	Akdeniz Üniversitesi Fen Bilimleri Enstitüsü, Elektrik-Elektronik Mühendisliği Anabilim Dalı, Antalya
Lisans 2013-2017	Karadeniz Teknik Üniversitesi Mühendislik Fakültesi, Elektrik-Elektronik Mühendisliği Bölümü, Trabzon

İŞ DENEYİMİ BİLGİLERİ

Transmisyon-Bilişim Mühendisi 2007-2010	Türk Telekomünikasyon A.Ş. ANTENSAN GMBH. Ankara
Elektrik-Elektronik Mühendisi 2012-2022	Hürriyet Gazetecilik ve Matbaacılık A.Ş. Antalya