



YILDIZ TEKNİK ÜNİVERSİTESİ
KİMYA – METALÜRJİ FAKÜLTESİ
MATEMATİK MÜHENDİSLİĞİ BÖLÜMÜ

MATEMATİK MÜHENDİSLİĞİ TASARIM
UYGULAMALARI

SİNİR AĞLARI İLE RAKAM TANIMA

Tez Yöneticisi: Dr. Mert BAL
Adı Soyadı: Hüsnü Mümtaz SANCAK
Öğrenci Numarası: 15052059

İstanbul, 2020

ÖNSÖZ

Bu proje süresince bana desteklerini esirgemeyen ve yardımcı olan değerli hocam Dr. Mert Bal'a teşekkürlerimi sunarım.

İÇİNDEKİLER

Kısaltma Listesi.....	v
Şekil Listesi.....	v - vi
Önsöz.....	vi
Özet.....	vii
Abstract.....	vii
1.Giriş.....	1
2. Yapay Zeka.....	2
2.1 Tanım.....	2
2.2 Yapay Zekanın Gelişimi.....	3
3. Derin Öğrenme.....	4
3.1 Tanım.....	4
3.2 Denetimli Öğrenme.....	4-5
3.3 Denetimsiz Öğrenme.....	5
4. Yapay Sinir Ağları.....	6
4.1 Tanım.....	6
4.2 Yapay Sinir Ağlarının Çalışma Prensipleri.....	6-7
4.3 Yapay Sinir Ağının Katmanları.....	8-9
4.4 Yapay Sinir Ağlarının Eğitilmesi.....	9-11
5. Görüntü İşleme.....	12
5.1 Tanım.....	12
5.2 Görüntü Türleri.....	13
5.3 Görüntünün Dijitale Dönüştürülmesi.....	13
5.3.1 Görüntü Örnekleme.....	13
5.3.2 Görüntü Niceleme.....	14
5.4 Gri Görüntü.....	14-15

6. MNIST Veri Seti.....	16
7. One Hot Encoding.....	17
8. Uygulama.....	18-28
9. Sonuçlar ve Öneriler.....	29
10. Kaynaklar.....	30-31
11. Özgeçmiş.....	32

Kısaltma Listesi

MNIST: Modified National Institute of Standards and Technology

YSA: Yapay Sinir Ağları

Şekil Listesi

Şekil 1.1	MNIST veri setinin görsel hali.	1
Şekil 2.1	Yapay zekânın diğer alanlarla ilişkisi	2
Şekil 2.2	Yapay zekânın zaman içindeki gelişimi.	3
Şekil 3.1	Derin öğrenme ve yapay sinir ağları.	4
Şekil 4.1	Yapay sinir ağlarının genel görünümü.	6
Şekil 4.2	Bir nöronun modeli.	8
Şekil 4.3	YSA modelinin katmanları	9
Şekil 5.1	$N \times M$ büyüklüğünde matrisin 2-B sayısal görünümünün temel yapısı.	14
Şekil 5.2	16x16'lık ızgara üzerinde 256 farklı gri seviyesinin gösterimi.	15
Şekil 5.3	İkili (Binary) görüntü.	15
Şekil 6.1	Veri normalizasyonun tahmindeki etkisi. (Soldaki normalize edilmemiş veri, sağdaki ise aynı verinin normalize edilmiş hali.)	16
Şekil 7.1	Label formatından One Hot Encoding e geçiş.	17
Şekil 8.1	Verinin yüklenmesi ve incelenmesi.	18
Şekil 8.2	Rakamların incelenmesi.	19
Şekil 8.3	Renklerin pikseller üzerinde dağılımı.	20
Şekil 8.4	Relu fonksiyonunun grafiği.	21
Şekil 8.5	YSA modeli.	22
Şekil 8.6	Normalizasyondan sonra veri.	22
Şekil 8.7	Modelin eğitime başlaması ve sürecin kaydının tutulması.	23
Şekil 8.8	Modelin eğitimi ve sonuçları.	24
Şekil 8.9	Modelin kaydı ve ilk değer için tahmini.	25
Şekil 8.10	Rastgele seçilen 6 değer tahminleri ve gerçek değerleri.	26
Şekil 8.11	Kayıp verilerin tespiti için hazırlık.	27

Şekil 8.12 Modelin yanlış tahmin ettiği rakamlar.	27
Şekil 8.13 Modelin görsel hali.	28

ÖZET

Bu çalışmada yapay sinir ağıları modelini kullanarak, 724 pikselden oluşan rakamlar ile bu model eğitilmiştir. Eğitilen modelin başarısı test verileriyle test edilmiştir. Kullanılan yöntemler açıklanmış, her aşaması ve sonucu analiz edilmiştir. Bu çalışma görüntü işlemeye giriş seviyesi olarak değerlendirilebilir.

ABSTRACT

In this study, using artificial neural network model, this model was trained with digits consisting of 724 pixels. The success of the trained model was tested with test data. The methods used were explained and each step and result was analyzed. This work can be considered as an basic level for image processing.

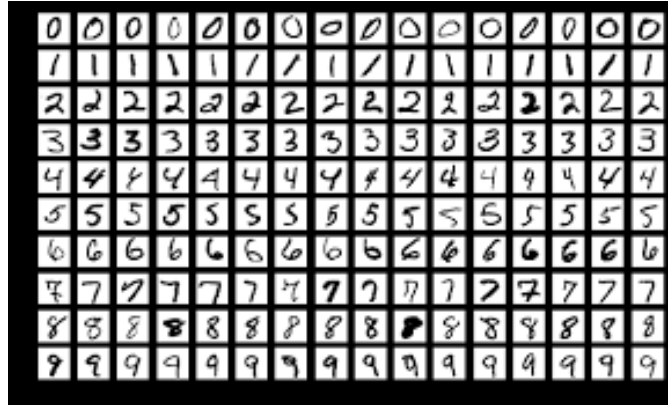
1.GİRİŞ

Bu tezde MNIST (“Modified National Institute of Standards and Technology”) veri tabanından çeşitli görüntü işleme sistemlerini eğitmek için yaygın olarak kullanılan el yazısı rakamlarından oluşan bir veri seti ile bir yapay sinir ağı modelini eğiterek bu modelin tahmin başarısını ve çalışma mantığını öğrenmek hedeflenmiştir. [1]

Bu tezin temelini oluşturan yapay zekâ, yapay sinir ağları gibi kavramların neler olduğu ve projede nasıl uygulandığı da bu bağlamda anlatılmıştır.

Tezde kullanılan veri seti açık kaynak olup çeşitli kaynaklardan temin edilebilmektedir. Projenin uygulama safhası Python programlama dilinde ve Jupyter Notebook ortamında yapılmıştır. Günümüzde açık kaynak olması ve bu konuda çok fazla kütüphaneye sahip olması sebebiyle Python programlama dili tercih edilmiştir. Sunuma ve anlatmaya daha elverişli olması sebebiyle Jupyter Notebook ortamı kullanılmıştır.

Veri seti eğitim ve test verileri olmak üzere iki parçadan oluşmaktadır. Eğitim veri seti modeli eğitmek için kullanılır. Daha sonra eğitilen bu modelin test verileriyle elde ettiği tahminlere bakılır.



Şekil 1.1 MNIST veri setinin görsel hali.

Bu tezin giriş kısmında tez hakkında kısa bir özet yapılmış tezin amacı, hedefi ve kullanılan teknolojilerden bahsedilmiştir. Sonraki bölümde tezin teorik arka planı ve uygulamanın kendisi açıklanmıştır.

2.YAPAY ZEKÂ

İnsan gibi öğrenmeyi ve davranmayı makine, bilgisayar ortamında gerçekleştirmeyi amaç edinen çalışma alanı.

2.1 Tanım:

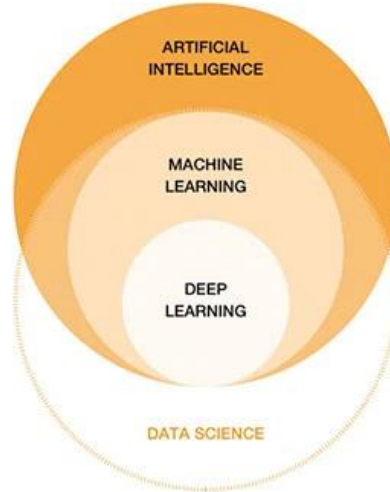
En basit ifadeyle yapay zekâ (AI), görevleri yerine getirmek için insan zekâsını taklit eden ve topladıkları bilgilere göre yinelemeli olarak kendilerini iyileştirebilen sistemler veya makineler anlamına gelir. Yapay zekâ pek çok biçimde kendini gösterir. Örneğin:

Sohbet robotları, müşterilerin sorunlarını daha hızlı bir şekilde anlamak ve daha verimli cevaplar vermek için yapay zekâdan yararlanır

Akıllı asistanlar, zamanlamayı iyileştirmek için büyük kullanıcı tanımlı veri kümelerinden kritik bilgileri çekmek için yapay zekâdan yararlanır

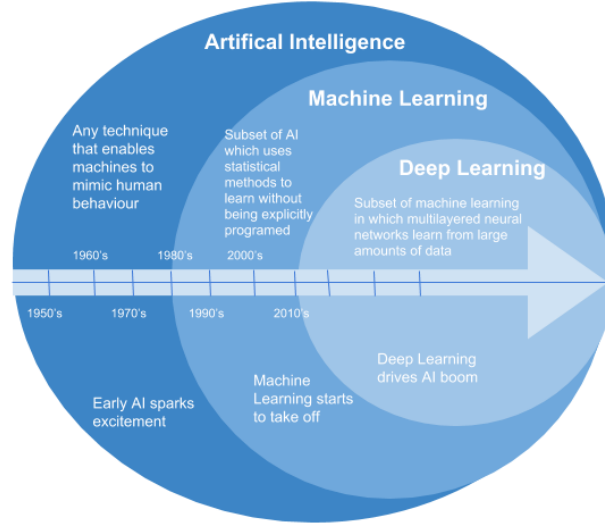
Öneri motorları, kullanıcıların izleme alışkanlıklarına göre TV programları için otomatik öneriler sunabilir

Yapay Zekâ, herhangi bir özel biçim veya işlevden ziyade süper güçlendirilmiş düşünce ve veri analizi yeteneği ve süreci ile ilgilidir. Yapay Zekâ dendiğinde zihinlerde dünyayı ele geçiren çok fonksiyonel, insan benzeri robotlar canlansa da yapay zekâ insanların yerine geçmek üzere tasarlanmamıştır. İnsan yeteneklerini ve katkılarını önemli ölçüde geliştirmek üzere tasarlanmıştır. Bu nedenle oldukça değerli bir ticari varlıktır.[2]



Şekil 2.1 Yapay zekânın diğer alanlarla ilişkisi.

2.2 Yapay Zekânın Gelişimi



Şekil 2.2 Yapay zekânın zaman içindeki gelişimi.

Yapay zekâ çalışmaları ilk başladığında bu alanda çalışan insanların nihai hedefi insan gibi düşünebilen hatta insandan daha üstün makineler üretebilmektir. Gerek fiziksel zorluklar gerekse dönemin teknolojik imkânlarıyla bu pek mümkün değildi. Bu noktada farklı çözüm arayışlarına giren bilim insanları istatistiksel yöntemlerle tahmin sonuçları elde ederek makine öğrenme alanını ortaya çıkardılar. Sahip olduğunuz veriye uygun makine öğrenme algoritması ile o verinin çeşitli parametreler altında gelecekteki ve ya geçmişteki durumunu gerçeğe yakın tahmin etmek mümkün oldu. Bu noktada insan beyninin çalışma ve öğrenme sistemine uzaklaşmış gibi görünse de günümüzde yapay sinir ağları ile makine öğrenme bir adım öteye taşınarak insan beynine benzer şekilde bir öğrenme biçimi makineler için mümkün hale geldi. Böylece çok büyük veri setleri ile çalışmak daha kolay ve hızlı hale gelmiş oldu.

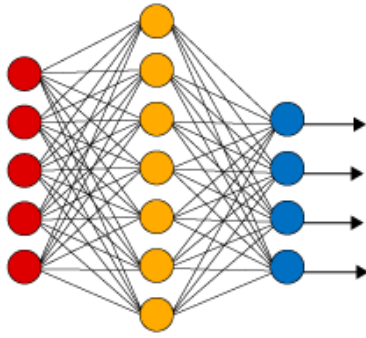
3.DERİN ÖĞRENME

Makine öğrenme yöntemlerinden biri olup yapay sinir ağları modelinin kullanıldığı bir metottur. Günümüzde gelişen teknoloji ve büyük veri ile popülerliği ve kullanımı artmıştır.

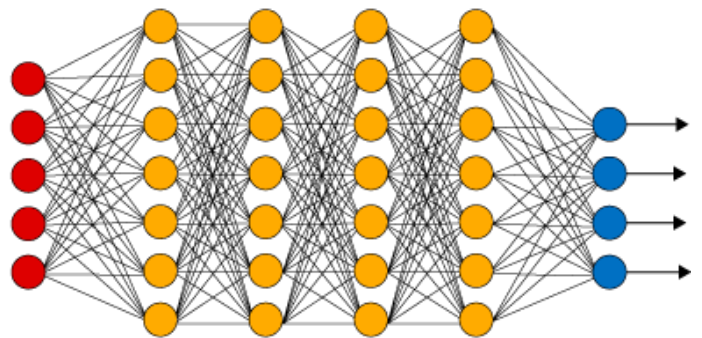
3.1 Tanım:

Derin Öğrenme bir makine öğrenme yöntemidir. Verilen bir veri kümesi ile çıktıları tahmin edecek yapay zekâyı eğitmeye olanak sağlar. Yapay zekâyı eğitmek için hem denetimli hem de denetimsiz öğrenme kullanılabilir. [3]

Simple Neural Network



Deep Learning Neural Network



● Input Layer ● Hidden Layer ● Output Layer

Şekil 3.1 Derin öğrenme ve yapay sinir ağları.

Derin öğrenmenin en önemli özelliği veriler arasındaki farkı kendi kendine öğrenebilmesidir. Bir başka deyişle iki veri arasında ki farka göre çalışan bir algorithmadan da öte bu iki veri arasındaki farkı bularak çalışmaktadır. Yani sınıflandırmayı nasıl yapacağını kendisi öğrenir.

Bunun dışında ileride bahsedeceğimiz yapay sinir ağlarına derin öğrenme denilebilmesi için genelde birden daha fazla gizli(ara) katman olmalıdır.

3.2 Denetimli Öğrenme

Denetimli öğrenmeyi kullanarak bir Yapay Zekâ eğitirken, ona bir girdi verir ve beklenen çıktıyı söylersiniz.

Yapay Zekâ tarafından üretilen çıktı yanlışsa, hesaplamalarını yeniden ayarlar. Bu işlem, Yapay Zekânın hata oranını en aza indirene kadar veri seti üzerinden tekrar tekrar yapılır.

Denetimli öğrenmeye örnek, hava durumu belirleyici Yapay Zekâdır. Geçmiş verilerini kullanarak hava durumunu tahmin etmeyi öğrenir. Bu eğitim verilerinde girdiler (basınç, nem, rüzgâr hızı) ve çıktılar (sıcaklık) bulunur.[4]

Bu tez çalışmasında modelimiz denetimli öğrenme metodu ile geliştirilecektir. Bunun daha detaylı açıklaması uygulama kısmında açıklanacaktır.

3.3 Denetimsiz Öğrenme

Denetlenmemiş öğrenmeyi kullanarak bir Yapay Zekâyı eğiterseniz, Yapay Zekâyı verilerin mantıksal sınıflandırmasını yapma izin verirsiniz.

Denetimsiz öğrenmenin bir örneği, bir e-ticaret web sitesi için tahmin yapan yapay zekâ örnek verilebilir. Çünkü burada etiketli bir girdi ve çıktı veri seti kullanılarak öğrenilmez.

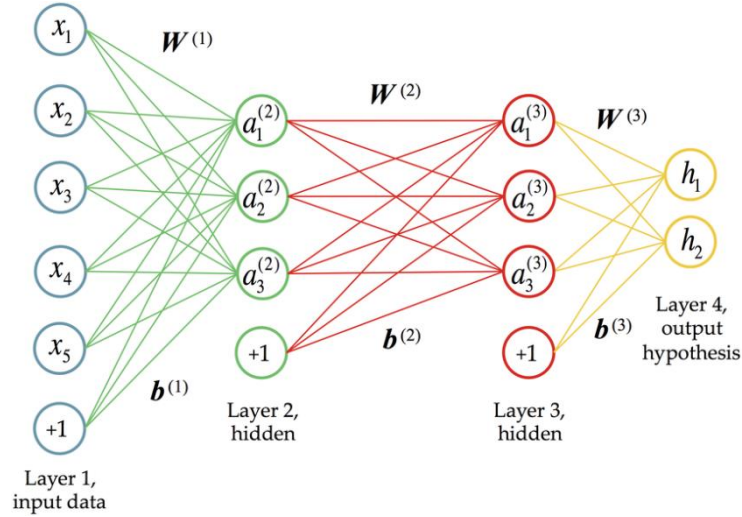
Bunun yerine girdi verileri kullanarak kendi sınıflandırmasını oluşturacaktır. Hangi tür kullanıcıların daha fazla farklı ürün alabileceklerini size söyleyecektir.[5]

4. YAPAY SİNİR AĞLARI

Yapay sinir ağları, insan beyninden esinlenerek geliştirilmiş, ağırlıklı bağlantılar aracılığıyla birbirine bağlanan ve her biri kendi belleğine sahip işlem elemanlarından oluşan paralel ve dağıtılmış bilgi işleme yapılarıdır. Yapay sinir ağları, bir başka deyişle, biyolojik sinir ağlarını taklit eden bilgisayar programlarıdır. Yapay sinir ağları zaman zaman bağlantılık (connectionism), paralel dağıtılmış işlem, sinirsel-işlem, doğal zekâ sistemleri ve makine öğrenme algoritmaları gibi isimlerle de anılmaktadır. [6]

4.1 Tanım

Yapay sinir ağları, insan beyninin bilgi işleme tekniğinden esinlenerek geliştirilmiş bir bilgiişlem teknolojisidir. YSA ile basit biyolojik sinir sisteminin çalışma şekli taklit edilir. Taklit edilen sinir hücreleri nöronlar içerirler ve bu nöronlar çeşitli şekillerde birbirlerine bağlanarak ağı oluştururlar.[7]



Şekil 4.1 Yapay sinir ağlarının genel görünümü.

YSA insanlardaki nöronlardan ilham alınarak, insan beynine benzer bir şekilde çalışmaktadır. Bu öğrenme sürecinin başarılı bir şekilde gerçekleşebilmesi temelde şu iki önemli noktaya bağlıdır. Veriye uygun modelin oluşturulması ve çok sayıda veriye sahip olunmasıdır.

4.2 Yapay Sinir Ağlarının Çalışma Prensibi

Yapay sinir hücreleri de biyolojik sinir hücrelerine benzer yapıdadır. Yapay nöronlar da aralarında bağ kurarak yapay sinir ağlarını oluştururlar. Aynı biyolojik nöronlarda olduğu gibi yapay nöronların da giriş sinyallerini aldıkları, bu sinyalleri toplayıp işledikleri ve çıktılarını ilettikleri bölümleri bulunmaktadır.

Bir yapay sinir hücresi beş bölümden oluşmaktadır;

- Girdiler
- Ağırlıklar
- Toplama Fonksiyonu (Birleştirme Fonksiyonu)
- Aktivasyon fonksiyonu
- Çıktılar

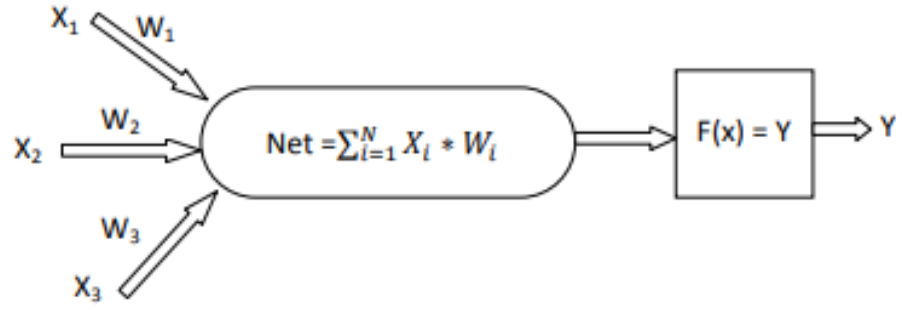
Girdiler: Girdiler nöronlara gelen verilerdir. Girdiler yapay sinir hücresine bir diğer hücreden gelebileceği gibi direk olarak dış dünyadan da gelebilir. Bu girdilerden gelen veriler biyolojik sinir hücrelerinde olduğu gibi toplanmak üzere nöron çekirdeğine gönderilir.

Ağırlıklar: Yapay sinir hücresine gelen bilgiler girdiler üzerinden çekirdeğe ulaşmadan önce geldikleri bağlantıların ağırlığıyla çarpılarak çekirdeğe iletilir. Bu sayede girdilerin üretilecek çıktı üzerindeki etkisi ayarlana bilinmektedir. Bu ağırlıkların değerleri pozitif, negatif veya sıfır olabilir. Ağırlığı sıfır olan girdilerin çıktı üzerinde herhangi bir etkisi olmamaktadır.

Toplama Fonksiyonu (Birleştirme Fonksiyonu): Toplama fonksiyonu bir yapay sinir hücresine ağırlıklarla çarpılarak gelen girdileri toplayarak o hücrenin net girdisini hesaplayan bir fonksiyondur.

Aktivasyon Fonksiyonu: Bu fonksiyon hücreye gelen net girdiyi işleyerek hücrenin bu girdiye karşılık üreteceği çıktıyı belirler. Aktivasyon fonksiyonu genellikle doğrusal olmayan bir fonksiyon seçilir. Yapay sinir ağlarının bir özelliği olan “doğrusal olmama” aktivasyon fonksiyonlarının doğrusal olmama özelliğinden gelmektedir. Aktivasyon fonksiyonu seçilirken dikkat edilmesi gereken bir diğer nokta ise fonksiyonun türevinin kolay hesaplanabilir olmasıdır. Geri beslemeli ağlarda aktivasyon fonksiyonunun türevi de kullanıldığı için hesaplamaların yavaşlamaması için türevi kolay hesaplanır bir fonksiyon seçilir. Günümüzde en yaygın olarak kullanılan “Çok katmanlı algılayıcı” modelinde genel olarak aktivasyon fonksiyonu olarak “Sigmoid fonksiyonu” kullanılır.

Hücrenin Çıktısı: Aktivasyon fonksiyonundan çıkan değer hücrenin çıktı değeri olmaktadır. Bu değer ister yapay sinir ağının çıktısı olarak dış dünyaya verilir isterse tekrardan ağın içinde kullanılabilir. Her hücrenin birden fazla girdisi olmasına rağmen bir tek çıktısı olmaktadır. Bu çıktı istenilen sayıda hücreye bağlanabilir.[8]



Şekil 4.2 Bir nöronun modeli.

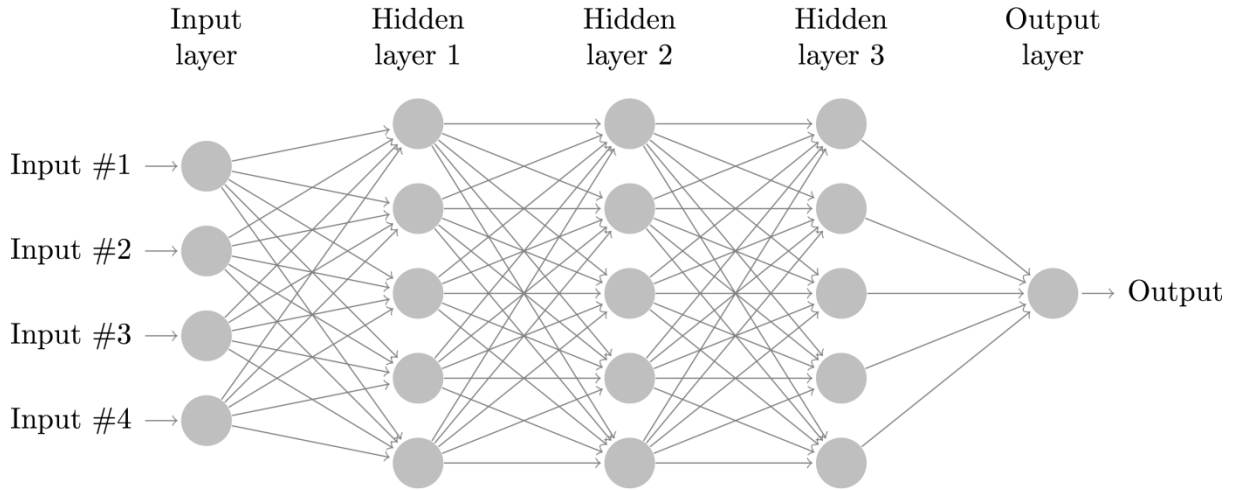
4.3 Yapay Sinir Ağının Katmanları

Yapay sinir ağları yapay sinir hücrelerinin birbirine bağlanmasıyla oluşan yapılardır. Yapay sinir ağları üç ana katmanda incelenir; Giriş Katmanı, Ara (Gizli) Katmanlar ve Çıkış Katmanı.

Giriş Katmanı: Yapay sinir ağına dış dünyadan girdilerin geldiği katmandır. Bu katmanda dış dünyadan gelecek giriş sayısı kadar hücrenin bulunmasına rağmen genelde girdiler herhangi bir işleme uğramadan alt katmanlara iletilmektedir.

Ara (Gizli) Katmanlar: Giriş katmanından çıkan bilgiler bu katmana gelir. Ara katman sayısı ağdan ağa değişebilir. Bazı yapay sinir ağlarında ara katman bulunmadığı gibi bazı yapay sinir ağlarında ise birden fazla ara katman bulunmaktadır. Ara katmanlardaki nöron sayıları giriş ve çıkış sayısından bağımsızdır. Birden fazla ara katman olan ağlarda ara katmanların kendi aralarındaki hücre sayıları da farklı olabilir. Ara katmanların ve bu katmanlardaki nöronların sayısının artması hesaplama karmaşıklığını ve süresini arttırmasına rağmen yapay sinir ağının daha karmaşık problemlerin çözümünde de kullanılabilmesini sağlar.

Çıkış Katmanı: Ara katmanlardan gelen bilgileri işleyerek ağın çıktılarını üreten katmandır. Bu katmanda üretilen çıktılar dış dünyaya gönderilir. Geri beslemeli ağlarda bu katmanda üretilen çıktı kullanılarak ağın yeni ağırlık değerleri hesaplanır.[9]



Şekil 4.3 YSA modelinin katmanları

Bu çalışmada oluşturulacak model bir giriş katmanından, iki gizli katmandan, iki silme katmanından ve bir çıkış katmanından oluşacaktır. Katmanların özellikleri uygulama kısmında anlatılacaktır.

Bu noktada belirtmek gerekir ki oluşturulacak modelin yapısı kullanacağınız veriye kesinlikle uygun olmalıdır. Katman miktarı, nöron sayısı, aktivasyon fonksiyonları vb. Uygun modeli oluşturmak içinse kesin bir yöntem bulunmamakla birlikte çoğu zaman deneme yanılma yolu ile en yüksek tutarlılığı sağlayan model tercih edilir. Yine de verinin türü ve sahip olduğu parametreler modelin inşasında faydalı olacaktır. Söz konusu bu çalışma için uygun olan model ileriki bölümlerde açıklanacaktır.

4.4 Yapay Sinir Ağlarının Eğitilmesi

İnsan beyni doğumdan sonraki gelişme sürecinde çevresinden duyu organlarıyla algıladığı davranışları yorumlar ve bu bilgileri diğer davranışlarında kullanır. Yaşadıkça beyin gelişir ve tecrübelenir. Artık olaylar karşısında nasıl tepki göstereceğini çoğu zaman bilmektedir. Fakat hiç karşılaşmadığı bir olay karşısında yine tecrübesiz kalabilir. Yapay sinir ağlarının öğrenme sürecinde de dış ortamdan girişler alınır, aktivasyon fonksiyonundan geçirilerek bir tepki çıkışı üretilir. Bu çıkış yine tecrübeyle verilen çıkışla karşılaştırılarak hata bulunur. Çeşitli öğrenme algoritmalarıyla hata azaltılıp gerçek çıkışa yaklaşılmaya çalışılır. Bu çalışma süresince yenilenen yapay sinir ağının ağırlıklarıdır. Ağırlıklar her bir çevrimde yenilenerek amaca ulaşılmaya çalışılır. Amaca ulaşmanın veya yaklaşmanın ölçüsü de yine dışarıdan verilen bir değerdir. Eğer yapay sinir ağı verilen giriş-çıkış çiftleriyle amaca ulaşmış ise ağırlık değerleri saklanır.

Ağırlıkların sürekli yenilenerek istenilen sonuca ulaşılan kadar geçen zamana öğrenme adı verilir. Yapay sinir ağı öğrendikten sonra daha önce verilmeyen girişler verilir, sinir ağı çıkışıyla gerçek çıkış yaklaşımı incelenir. Eğer yeni verilen örneklerle de doğru yaklaşıyorsa sinir ağı işi öğrenmiş demektir.

Sinir ağına verilen örnek sayısı uygun değer değerden fazla ise sinir ağı işi öğrenmemiş, ezberlemiş demektir. Genelde eldeki örneklerin %80 ağı verilip ağı eğitilir. Daha sonra kalan %20'lik kısım verilip ağın davranışı incelenir. Böylece ağı testi yapılmış olur.

1. Örneklerin toplanması: Ağı öğrenmesi istenilen olay için daha önce gerçekleşmiş örneklerin bulunması adımıdır. Ağı eğitilmesi için örnekler toplandığı gibi (eğitim seti) ağı test edilmesi için de örneklerin (test seti) toplanması gerekmektedir. Eğitim setindeki örnekler tek tek gösterilerek ağın olayı öğrenmesi sağlanır. Ağı olayı öğrendikten sonra test setindeki örnekler gösterilerek ağın performansı ölçülür. Hiç görmediği örnekler karşısındaki başarısı ağın iyi öğrenip öğrenmediğini ortaya koyar.

2. Ağı topolojik yapısının belirlenmesi: Öğrenilmesi istenen olay için oluşturulacak olan ağın topolojik yapısı belirlenir. Kaç tane girdi ünitesi, kaç tane ara katman, her ara katmanda kaç tane süreç eleman kaç tane çıktı eleman olması gerektiği bu adımda belirlenmektedir.

3. Öğrenme parametrelerinin belirlenmesi: Ağı öğrenme katsayısı, proses elemanlarının toplama ve aktivasyon fonksiyonları, momentum katsayısı gibi parametreler bu adımda belirlenmektedir.

4. Ağırlıkların başlangıç değerlerinin atanması: Proses elemanlarını birbirlerine bağlayan ağırlık değerlerinin ve eşik değer ünitesinin ağırlıklarının başlangıç değerlerinin atanması yapılır. Başlangıç genellikle rasgele değerler atanır. Daha sonra ağı uygun değerleri öğrenme sırasında kendisi belirler.

5. Öğrenme setinden örneklerin seçilmesi ve ağı gösterilmesi: Ağı öğrenmeye başlaması ve Öğrenme kuralına uygun olarak ağırlıkları değiştirmesi için ağı örnekler belirli bir düzeneğe göre gösterilir.

6. Öğrenme sırasında ileri hesaplamaların yapılması: Sunulan girdi için ağı çıktısı değerleri hesaplanır.

7. Gerçekleşen çıktının beklenen çıktı ile karşılaştırılması: Ağı ürettiği hata değerleri bu adımda hesaplanır.

8. Ağırlıkların değiştirilmesi: Geri hesaplama yöntemi uygulanarak üretilen hatanın azalması için ağırlıkların değiştirilmesi yapılır.

9. Öğrenmenin tamamlanması: İleri beslemeli sinir ağı öğrenmeyi tamamlayınca, yani gerçekleşen ile beklenen çıktılar arasındaki hatalar kabul edilir düzeye ininceye kadar devam eder.

Ağı kendisine gösterilen girdi örneği için beklenen çıktıyı üretmesini sağlayacak ağırlık değerleri başlangıçta rastgele atanmakta ve ağı örnekler gösterildikçe ağırlıklar değiştirilerek istenen değerlere ulaşması sağlanmaktadır. İstenen ağırlık değerlerinin ne olduğu bilinmemektedir.

Bu nedenle YSA' nın davranışlarını yorumlamak ve açıklamak mümkün olmaz. Bazı durumlarda ağı takıldığı yer hata düzeyinin üstünde kalabilir. Bu durumda ağı olayı öğrenmesi için bazı değişiklikler yapılarak yeniden eğitilmesi gerekir. Bunlar; Başlangıç değerlerinde değişiklik yapılabilir. Ağı topolojisinde değişiklikler yapılabilir.

Ağın parametrelerinde değişiklikler yapılabilir. Ağa sunulan verilerin gösterimi ve örneklerin formülasyonu değiştirilerek yeni örnek seti oluşturulabilir. Öğrenme setindeki örneklerin sayısı artırılabilir veya azaltılabilir. İleri beslemeli sinir ağının yerel sonuçlara takılıp kalmaması için momentum katsayısı geliştirilmiştir. Ağların eğitilmesinde diğer önemli bir sorun ise öğrenme süresinin çok uzun olmasıdır. Ağırlık değerleri başlangıçta büyük değerler olması durumunda ağın yerel sonuçlara düşmesi ve bir yerel sonuçtan diğerine sıçramasına sebep olmaktadır. Eğer ağırlıklar küçük aralıkta seçilirse o zaman da ağırlıkların doğru değerleri bulması uzun zamanlar almaktadır. Bazı problemlerin çözümü sadece 200 iterasyon sürerken bazıları 5-10 milyon iterasyon sürmektedir. Ağın öğrenmesinin gösterilmesinin en güzel yolu hata grafiğini çizmektir. Her iterasyonda oluşan hatanın grafiği çizilirse hatanın zaman içinde düştüğü gözlenebilir. Belirli bir iterasyondan sonra hatanın daha fazla azalmayacağı görülür. Bu ağın öğrenmesinin durduğu ve daha iyi bir sonuç bulunamayacağı anlamına gelir. Eğer elde edilen çözüm kabul edilemez ise o zaman ağ yerel bir çözüme takılmış demektir.[10]

5. GÖRÜNTÜ İŞLEME

Görüntü işleme bilgisayar ortamında görsel (resimsel) bilgilerin manipülasyonuna ve analizine denir. Mevcut görüntüyü işlemek, iyileştirmek, değiştirmek ve ya sınıflandırmak için kullanılır.

5.1 Tanım

Görüntü işleme, herhangi bir aygıt aracılığıyla alınan görüntüler üzerinde herhangi bir işlem yapabilmeyi sağlayan tekniğe verilen isimdir. Görüntü işleme; herhangi bir görüntünün netliğini artırma, görüntü üzerinde bulunan herhangi bir nesnenin elde edilebilmesi ya da nesnelerin tanımlanabilmesi gibi birçok amaçla kullanılmaktadır. Herhangi bir resmin yazılım aracılığıyla kullanılabilmesi için sayısallaştırılması gerekmektedir. Sayısallaştırma; resimde bulunan renklerin sayısal değerlerle ifade edilmesidir.

Bir görüntüden faydalı bir bilgi çıkarılarak yorumlanması gerektiğinde görüntü işleme tekniklerinden faydalanılmaktadır. İşlenecek görüntü, kameralar, optik tarayıcılar ve fotoğraf makineleri yardımıyla elde edilebilir. Bu dijital görüntülerin sayısallaştırılmasıyla üzerinde farklı işlemler uygulanarak anlamlı yorumlanabilir sonuçlar elde edilebilir. Tıp, Askeri, Endüstriyel ve Coğrafi Sistemler gibi birçok alanda kullanılan görüntü işleme teknikleri, güvenlik sistemleri alanında da yaygın olarak kullanılmaktadır. Parmak izi, iris ve yüz tanıma gibi uygulamalar güvenlik alanında görüntü işleme teknikleri kullanılarak yapılabilmektedir.

Tarayıcı, kamera ya da fotoğraf makinesi üzerinden alınan görüntülerin yorumlanabilmesi için belirli ön işlemlerden geçirilmesi gerekmektedir.[11]

- Image Enhancement (Görüntü İyileştirme)
- Image Restoration (Görüntü Onarma)
- Morphological Operations (Morfolojik İşlemler)
- Edge Detection (Kenar Belirleme)
- Segmentation (Bölümleme)
- Recognition (Tanıma)
- Object Tracking (Nesne İzleme, Nesne Takibi)
- Template Matching (Şablon Eşleme)
- Image Compression (Görüntü Sıkıştırma)

Görüntü işlemenin bazı konuları olmakla birlikte bu çalışmada Rakam Tanıma yapılacaktır.

5.2 Görüntü Türleri

- İkili Görüntü

- Sadece siyah ve beyaz piksellerden oluşur.
- 1 piksel, 1 bit yer kaplar [0,1]

- Gri Tonlamalı Görüntü

- Sadece grinin tonlarından oluşur.
- Genelde piksel başına 8 bit ayrılır. [0 - 255]

Bu çalışmada kullanılacak görüntü dosyaları gri tonlamaları görüntü formatındadır.

- Renkli Görüntü

- RGB renk modeli ve HSV, YUV, CIELab
- RGB üç renk katmanından oluşur. [12]

5.3 Görüntünün Dijitale Dönüştürülmesi

Analog görüntülerin bilgisayar ortamında işlenebilmesi için sayısallaştırılmaları gerekir. Sayısallaştırma için ilk olarak örnekleme daha sonra nicemleme (kuantalama) yapılır. Fonksiyonun bilgisayar ortamında işlenebilmesi için hem konumsal (uzaysal) olarak hem de genlikte (renk bilgisinde) sayısallaştırılması gerekir.

Görüntü fonksiyonuna ilişkin koordinatlarının sayısallaştırılması, görüntü örnekleme olarak adlandırılırken; genlik değerlerinin sayısallaştırılmasına görüntü nicemleme adı verilir.

5.3.1 Görüntü Örnekleme

Sayısal bir görüntü, sürekli bir görüntü fonksiyonu üzerinden eşit aralıklarla x-ekseni boyunca N adet örnek ve y-ekseni boyunca M adet örnek alınarak oluşturulabilir. Böylece, sürekli-zamanlı görüntü fonksiyonundan ayrık zamanlı görüntü fonksiyonuna geçiş gerçekleşmiş olur.

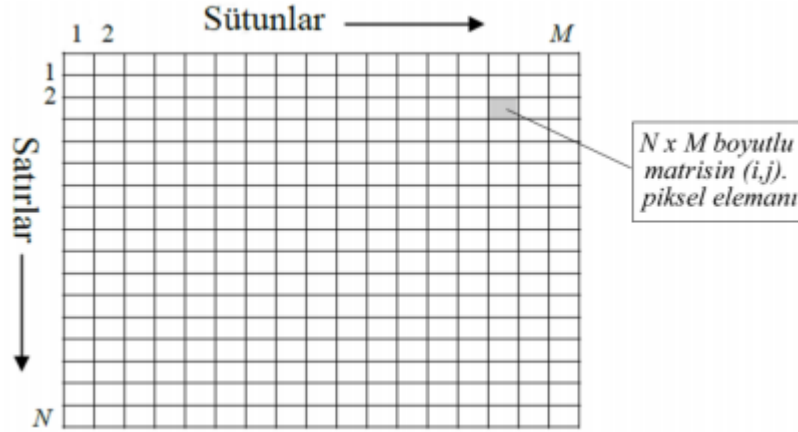
2-B ayrık-zamanda yatayda N ve düşeyde M örnekten oluşan toplam $N \times M$ sonlu örnek değeri ile analog bir görüntü yaklaşık olarak ifade edilebilir.

Bu işlemde, analog görüntü fonksiyonu düzgün örneklenmiş olur. Yani düzgün örnekleme, analog görüntüden hem yatay hem de düşey yönde eşit aralıklarla örnek alınarak oluşturulur. Oluşan dijital (sayısal) görüntü aslında N satır ve M sütundan oluşan bir matristir. Bu işlemde bilgi kaybı vardır.

5.3.2 Görüntü Nicemleme

Görüntünün her bir elemanın (pikselin) parlaklık şiddetini gösteren pozitif tamsayı değeri, nicemleme ile belirlenir. Görüntü elemanının en küçük ve en büyük genlik değerleri aralığı basamaklara ayrılarak, ilgili basamak değerine en yakın olan görüntü değerini almasıdır. Bu iki işlem sonucunda, bilgisayarlar tarafından işlenebilen sayısal bir görüntü elde edilmiş olur.

Analog bir görüntüyü bu iki işlemi de yaparak sayısallaştıran örnek cihazlardan birisi Tarayıcı (scanner) dır. Tarayıcıdan belli formatlarda elde edilen görüntüler sayısaldir ve bilgisayarda yazılımlarla işlenebilecek haldedir. Örnekleme ve nicemleme işleminden sonra elde edilen sayısal görüntü, bileşenleri pozitif tamsayı değerlerinden oluşmuş iki-boyutlu matris yapısındadır. Sayısal görüntüyü temsil eden matrisin her bir elemanı piksel olarak adlandırılır. Piksel, sayısal bir görüntüyü oluşturan en küçük eleman olup bir pikselin sahip olduğu değer ilgili görüntü elemanının parlaklık şiddetini belirtir. Parlaklık şiddeti ile ilgili olan pozitif tamsayı değeri, nicemleme ile belirlenir. [13]



Şekil 5.1 NxM büyüklüğünde matrisin 2-B sayısal görünümünün temel yapısı.

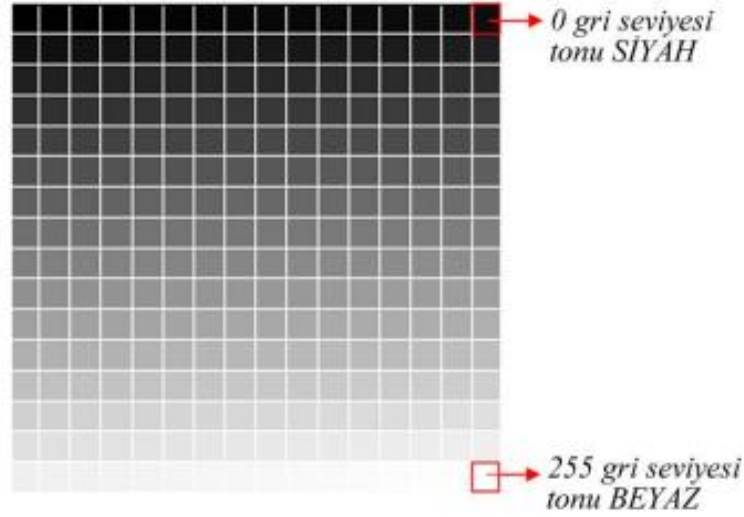
5.4 Gri Görüntü

Sayısallaştırma işleminde, görüntü boyutlarının ve her bir pikselin sahip olabileceği parlaklık değerinin belirlenmesi gerekir. Sayısal görüntünün her bir pikselinin sahip olduğu parlaklık değeri gri seviyeler olarak adlandırılır. Her bir pikseldeki parlaklık değerinin kodlandığı bit sayısına göre gri seviye aralığı belirlenir.

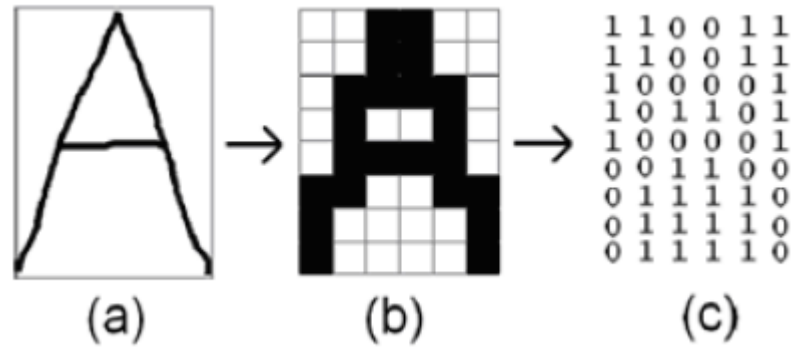
Gri seviyeni sınırlarında iki renk vardır, siyah ve beyaz. Bu ikisi arasında kodlanan görüntülere ise gri-ton (gray scale, monochromatic) görüntüler adı verilir. Uygulamada yaygın olarak kullanılan her bir piksel 8 bit ile kodlanmıştır.

Bu tip görüntülerde her bir piksel 28 = 256 farklı gri ton karşılığı (parlaklık seviyesi) değerlerinden oluşur ve gri değer aralığı $G = \{0, 1, 2, \dots, 255\}$ biçiminde ifade edilir.

Kural olarak; 0 gri seviyesi siyah renge, 255 gri seviyesi ise beyaz renge ve bu değerler arasındaki gri seviyeler ise gri tonlara karşılık gelir. Şekil de $N \times M = 16 \times 16$ 'lık bir ızgara üzerinde 256 farklı gri seviyenin gösterimi verilmiştir. [14]



Şekil 5.2 16x16'lık ızgara üzerinde 256 farklı gri seviyesinin gösterimi.



Şekil 5.3 İkili (Binary) görüntü.

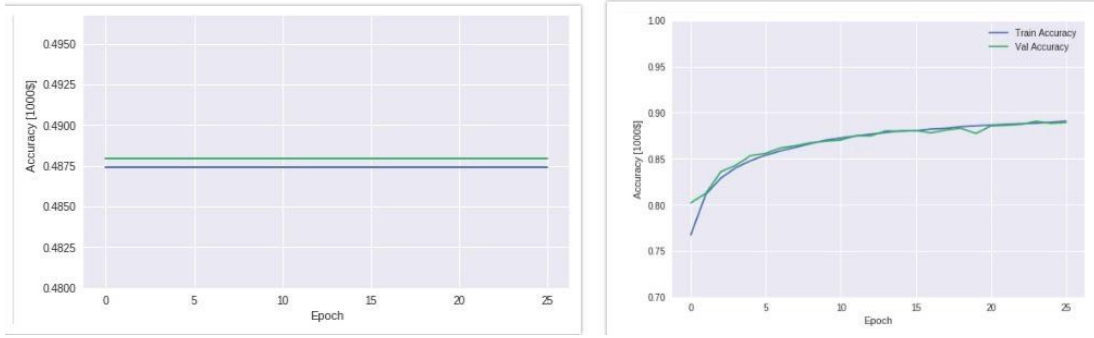
Yukarıdaki şekilde ikili görüntü örneği verilmiştir. Bu çalışmada kullanılan veriler ise 8 bitlik olup grinin 256 tonundan oluşan görüntüler kullanılmıştır. Yukarıdaki şekil iki farklı ton yani siyah ve beyaz renkten meydana gelmiştir.

6. MNIST VERİ SETİ

El yazısı rakamlardan oluşan MNIST veri tabanı, 60.000 satırlık eğitim seti ve 10.000 satırlık test setinden oluşmaktadır. MNIST' ten gelen orijinal siyah beyaz görüntülerin piksel değerleri normalizasyon ile 0 – 255 aralığından 0 – 1 aralığına indirildi. Normalizasyon sayesinde pikseller arasındaki farkı azalmıştır. Bu da modelin öğrenme süresini kısaltmasının yanında tutarlılığı da arttırmaktadır.

Left: Model Accuracy, without normalized data

Right: Model Accuracy with normalized data



Şekil 6.1 Veri normalizasyonun tahmindeki etkisi. (Soldaki normalize edilmemiş veri, sağdaki ise aynı verinin normalize edilmiş hali.)

MNIST veri setini işlemenin çeşitli yöntemleri vardır. Bu çalışmada one hot encoding yöntemi kullanılmıştır. Başarı oranı yüksek, hızlı, kolay anlaşılır olması sebebiyle tercih edilmiştir.

7. ONE HOT ENCODING

One Hot Encoding, kategorik deęişkenlerin ikili (binary) olarak temsil edilmesi anlamına gelmektedir. Bu işlem ilk önce kategorik deęerlerin tamsayı deęerleriyle eşlenmesini gerektirir. Daha sonra, her bir tamsayı deęeri, 1 ile işaretlenmiş tamsayı indeksi dışında ki tüm deęerleri sıfır olan bir ikili vektör olarak temsil edilir. Örneęin aşıağıda 3 kategoride veri vardır apple,chicken ve broccoli bu alanlar binary olarak ayrıştırıldığında Apple için ilk satır 1 iken dięerleri 0 oluyor Dięer veri içinde aynı şekilde sayısal veriye çevirme işlemi devam ediyor. [15]

Label Encoding			One Hot Encoding			
Food Name	Categorical #	Calories	Apple	Chicken	Broccoli	Calories
Apple	1	95	1	0	0	95
Chicken	2	231	0	1	0	231
Broccoli	3	50	0	0	1	50

Şekil 7.1 Label formatından One Hot Encoding e geçiş.

Bu çalışmada da one hot encoding kullanılmıştır. 0 dan 9 a 10 farklı rakam 10 farklı kategori anlamına gelmektedir. Yani çıkış vektörü 10 deęerden oluşmaktadır. Bir başka deyişle tahmin sonucu rakam 1 olduğunda çıkış katmanı [1,0,0,0,0,0,0,0,0,0] şeklinde bir vektör olacaktır. Vektörün 1 yazan deęeri en yüksek olasılığa sahiptir. Bu model her bir rakama ait pikselleri alarak olasılıkları hesaplar ve en yüksek olasılığa sahip olan deęeri tahmin sonucu olarak kabul eder. Eğitim sırasında hatalı bulduęu her sonuç için sinir ağılarındaki ağırlıkları güncelleyerek daha iyi bir tahmin yapmaya çalışır.

8. UYGULAMA

Uygulama Python program dilinde, Jupyter geliştirme ortamında hazırlanmıştır. Çeşitli Python kütüphaneleri kullanılmıştır. Bunların içinde Tensorflow'a ait keras yapay zeka uygulamalarında yapay sinir ağı modelini inşa etmek için kullanılan en önemli kütüphanelerden bir tanesidir. Hızlı, anlaşılır ve gelişmiş bir kütüphanedir. Pandas veri setlerini manipüle etmek için kullanılmıştır. Numpy gelişmiş dizi manipülasyonu için kullanılmıştır. Matplotlib ise görselleştirme için kullanılmıştır. Random ise rastgele sayılar elde etmek için kullanılmıştır.

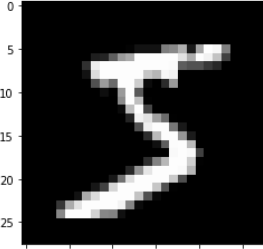
1. Adım

Verilere rahat bir şekilde uğraşmak için direkt olarak kerasın içinden MNIST veri seti data değişkenine yüklenmiştir. Daha sonra eğitim ve test veriler ayrılmıştır. X ile başlayanlar rakamların pikselleri iken, Y ile başlayanlar bu piksellerin oluşturduğu rakamın değeridir. Yani X içerisinde 28x28 toplam 784 piksel değeri vardır. Y de ise tüm bu piksellere karşılık gelen 0-9 arasındaki bir değerdir. Model eğitim verisinin X değerlerinde Y değerlerini bularak kendini eğitecektir. Daha sonra test verisinin X değerleri modele verildiğinde çıkan sonuçlar test verisinin Y değerleri ile karşılaştırılarak modelin başarısı ölçülecektir.

```
[2]: import pandas as pd
import tensorflow as tf
import matplotlib.pyplot as plt
import numpy as np
import random as rd
##tf.__version__
data = tf.keras.datasets.mnist #28x28 el yazısı rakamlar 0-9
(x_train, y_train), (x_test, y_test) = data.load_data()
print(x_train.shape) #eğitim verilerinin formatı

# ilk rakam
plt.imshow(x_train[0], cmap = "gray", vmin = 0, vmax = 255)
plt.show()
print(x_train[0][:10,:10]) #normalizasyondan önce
```

(60000, 28, 28)



[[0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 30 36]
 [0 0 0 0 0 0 0 0 49 238 253]
 [0 0 0 0 0 0 0 0 18 219 253]
 [0 0 0 0 0 0 0 0 80 156]]

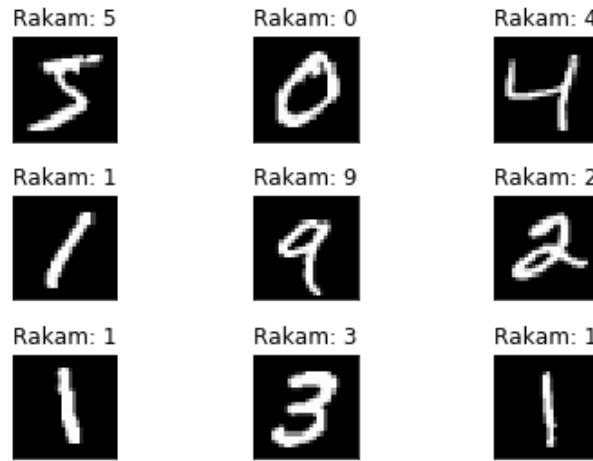
Şekil 8.1 Verinin yüklenmesi ve incelenmesi.

Eğitim verisinin ilk rakamı görselleştirilmiş ve daha sonra modelin giriş katmanına girecek piksellerin nasıl gözüktüğü 10x10 boyutunda gösterilmiştir. Değerler 0-256 arasındadır.

2. Adım

Bu aşamada eğitim setindeki ilk 9 rakamı görselleştiriyoruz. Görseller 28x28 boyutunda 0-256 gri renk aralığındadır. 0 siyah, 256 beyaz renktir.

```
[3]: fig = plt.figure()
for i in range(9):
    plt.subplot(3,3,i+1)
    plt.tight_layout()
    plt.imshow(x_train[i], cmap='gray', interpolation='none')
    plt.title("Rakam: {}".format(y_train[i]))
    plt.xticks([])
    plt.yticks([])
```



Şekil 8.2 Rakamların incelenmesi.

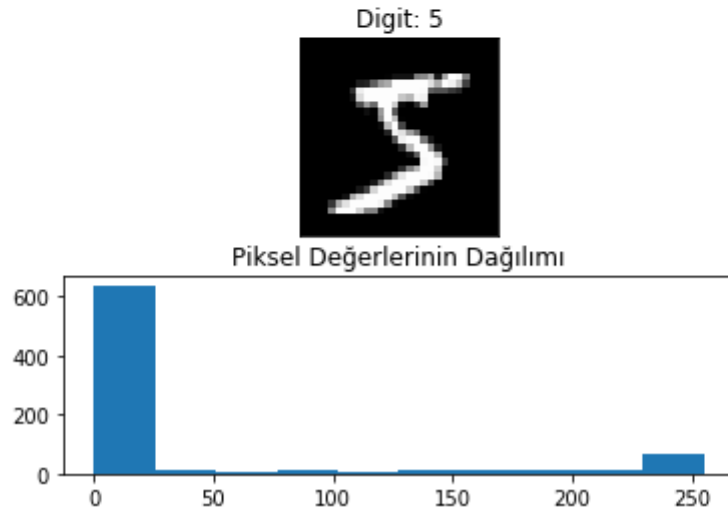
Bütün veri seti bu şekilde rakamlardan oluşmaktadır. Rakamlar el yazısı olduğu için bazı rakamların okunması burada görünenin aksine oldukça zor olabilir.

3. Adım

Görüntüler 28x28 yani 784 pikselden oluşmaktadır. Renk tonu ise 0-256 aralığındadır. Renk tonunun pikseller üzerindeki dağılımını görmek için aşağıdaki grafik oluşturulmuştur. Ortaya çıkan sonuç değerlerin düzgün dağılmadığını gösterir. Bu durum modelin eğitimi yavaşlatacağı gibi lokal ekstremum noktalarını da bulmayı zorlaştıracaktır. Bu yüzden ileriki aşamada veriler normalleştirilerek değerler 0-1 aralığına düşürülecektir.

```
[4]: # 784 pikselin 0-256 aralığındaki dağılımı
digit = 0
fig = plt.figure()
plt.subplot(2,1,1)
plt.imshow(x_train[digit], cmap='gray', interpolation='none')
plt.title("Digit: {}".format(y_train[digit]))
plt.xticks([])
plt.yticks([])
plt.subplot(2,1,2)
plt.hist(x_train[digit].reshape(784))
plt.title("Piksel Değerlerinin Dağılımı")
```

```
[4]: Text(0.5, 1.0, 'Piksel Değerlerinin Dağılımı')
```

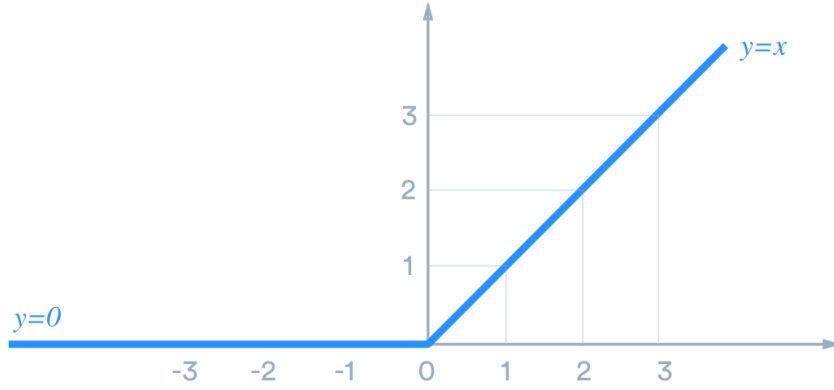


Şekil 8.3 Renklerin pikseller üzerinde dağılımı.

4. Adım

Yapay sinir ağı modelini bu aşamada inşa ediyoruz. Önce X eğitim ve test verilerini normalize edilmiştir. Daha sonra modelde kullan yapay sinir ağı modeli olan Sequential model koda eklenmiştir. Sequential model girdi ve çıktı değerlerinin farklı olması durumunda kullanılan gelişmiş ve kolay bir modeldir. Girdi verilerini encode ettikten sonra çıktı katmanından decode ederek sonuca ulaşılır.

Modelin ilk katmanı 28x28 matris formatını 784 satırlık tek sütunlu matris formatına getirmiştir. Böylece her bir piksel kendi başına birer input olmuştur. Daha sonra ilk sinir ağı katmanı oluşturulmuştur. Bu katman 512 nörona meydana gelmektedir ve aktivasyon fonksiyonu olarak relu tercih edilmiştir. Relu' nun açılımı rectified linear unit olup şuna benzemektedir.



Şekil 8.4 Relu fonksiyonunun grafiği.

Fonksiyon 0 ve 0'dan küçük değerler için 0' a eşittir. Böylece değeri sıfır olan piksellerin hesaplanmamıştır. Fonksiyonun sıfırdan büyük değerler için lineer olması daha çabuk yakınsamayı sağlar.

Bir sonraki katmanda eğitilmiş verilerin %2'sini siliyoruz. Bunun amacı overfitting denilen yani ezberleme durumunun önüne geçmek. Modelin eğitimi sırasında ezber durumu gerçekleşirse modelin tahmin yeteneği ciddi oranda düşer. Çünkü sadece eğitildiği veri için sonuç bulmayı öğrenmiştir. Bunu engellemek için öğrendiklerinin bir kısmını siliyoruz.

4.katman da 2.katmanın aynısını kullanılmıştır. Benzer şekilde verilerin yüzde ikisi silindikten sonra çıkış katmanından sonuca ulaşılır. Sonuç 10 değerden oluşan bir vektördür ve her bir değer girilen değer 0-9 arasındaki bir rakamın olma olasılığını verir. Bu olasılıklardan en büyüğü aranan cevaptır. Son katmandaki softmax fonksiyonu çıkış katmanındaki vektörde en büyük değeri bulur.

```
model = tf.keras.models.Sequential() #sequential model
```

Şekil 8.5 YSA modeli.

5. Adım

Bu adımda model eğitilmiştir. Batch_size ile modele aynı anda giren input sayısı belirlenir. Hızlı çalışması açısından 128 seçilmiştir. Epochs eğitimin kaç defa tekrar edileceğini ifade eden değerdir. Bu çalışmada model 20 defa eğitilecektir. Optimizer olarak adam kullanılmıştır. Optimizer modelin ortaya çıkan hata oranı kadar ağırlıkların düzeltilmesini sağlar. Adam hızlı ve az bellek kullandığı için tercih edilmiştir.

Geri kalan kodlar eğitim sürecinin grafiğini hazırlamak için yazılmıştır. Bu noktada modelin eğitim ve test verileri ile başarısı görülmektedir.

Şekil 8.6 Normalizasyondan sonra veri.

```
[47]: # modelin eğitimi ve sürecin kaydı
history = model.fit(x_train, y_train,
                    batch_size=128, epochs=20, verbose = 2,
                    validation_data=(x_test, y_test))

# eğitim sürecinin grafiksel yorumu
fig = plt.figure()
plt.subplot(2,1,1)
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('Modelin Başarısı')
plt.ylabel('Doğruluk')
plt.xlabel('Epoch')
plt.legend(['Eğitim', 'Test'], loc='lower right')

plt.subplot(2,1,2)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Modelin Kayıpları')
plt.ylabel('Kayıp')
plt.xlabel('Epoch')
plt.legend(['Eğitim', 'Test'], loc='upper right')

plt.tight_layout()
```

Şekil 8.7 Modelin eğitime başlaması ve sürecin kaydının tutulması.

Eğitimin sonunda modelin aşama-aşama başarısı ve sonuçları görülmektedir. Eğitim verileri ile %99,55 başarı elde edilmiş, test verileri ile %97,92 başarı elde edilmiştir. Eğitim verileri toplam 60.000 rakamdan, test verileri ise 10.000 rakamdan oluşmaktadır.

Daha fazla veri ile bu oranlar geliştirilebilir. Ayrıca kullanılan yöntem ve parametreler ile ince ayarlar yapılarak bu başarının artırılması mümkündür. İşlem gücü arttırılırsa daha derin bir yapay sinir ağı modeli oluşturulabilir.

Train on 60000 samples, validate on 10000 samples

Epoch 1/20
60000/60000 - 3s - loss: 0.2870 - acc: 0.9144 - val_loss: 0.1253 - val_acc: 0.9607

Epoch 2/20
60000/60000 - 3s - loss: 0.1107 - acc: 0.9660 - val_loss: 0.0935 - val_acc: 0.9698

Epoch 3/20
60000/60000 - 3s - loss: 0.0758 - acc: 0.9764 - val_loss: 0.0825 - val_acc: 0.9728

Epoch 4/20
60000/60000 - 3s - loss: 0.0567 - acc: 0.9819 - val_loss: 0.0794 - val_acc: 0.9765

Epoch 5/20
60000/60000 - 3s - loss: 0.0486 - acc: 0.9839 - val_loss: 0.0811 - val_acc: 0.9759

Epoch 6/20
60000/60000 - 3s - loss: 0.0380 - acc: 0.9872 - val_loss: 0.0810 - val_acc: 0.9772

Epoch 7/20
60000/60000 - 3s - loss: 0.0315 - acc: 0.9898 - val_loss: 0.0749 - val_acc: 0.9780

Epoch 8/20
60000/60000 - 3s - loss: 0.0266 - acc: 0.9909 - val_loss: 0.0779 - val_acc: 0.9792

Epoch 9/20
60000/60000 - 3s - loss: 0.0226 - acc: 0.9927 - val_loss: 0.0917 - val_acc: 0.9758

Epoch 10/20
60000/60000 - 3s - loss: 0.0225 - acc: 0.9923 - val_loss: 0.0872 - val_acc: 0.9793

Epoch 11/20
60000/60000 - 3s - loss: 0.0200 - acc: 0.9930 - val_loss: 0.0855 - val_acc: 0.9786

Epoch 12/20
60000/60000 - 3s - loss: 0.0191 - acc: 0.9933 - val_loss: 0.0925 - val_acc: 0.9789

Epoch 13/20
60000/60000 - 3s - loss: 0.0170 - acc: 0.9942 - val_loss: 0.0796 - val_acc: 0.9804

Epoch 14/20
60000/60000 - 3s - loss: 0.0161 - acc: 0.9944 - val_loss: 0.0969 - val_acc: 0.9782

Epoch 15/20
60000/60000 - 3s - loss: 0.0164 - acc: 0.9944 - val_loss: 0.0984 - val_acc: 0.9786

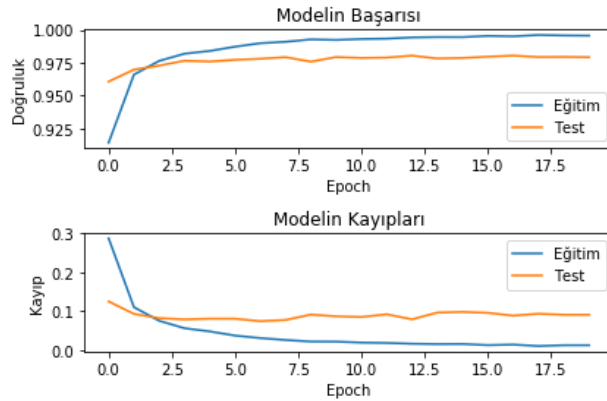
Epoch 16/20
60000/60000 - 3s - loss: 0.0135 - acc: 0.9953 - val_loss: 0.0961 - val_acc: 0.9796

Epoch 17/20
60000/60000 - 3s - loss: 0.0149 - acc: 0.9950 - val_loss: 0.0890 - val_acc: 0.9804

Epoch 18/20
60000/60000 - 3s - loss: 0.0112 - acc: 0.9960 - val_loss: 0.0938 - val_acc: 0.9793

Epoch 19/20
60000/60000 - 3s - loss: 0.0132 - acc: 0.9957 - val_loss: 0.0912 - val_acc: 0.9794

Epoch 20/20
60000/60000 - 3s - loss: 0.0132 - acc: 0.9955 - val_loss: 0.0912 - val_acc: 0.9792



Şekil 8.8 Modelin eğitimi ve sonuçları.

6. Adım

Model eğitildikten sonra yerel ortama modeli kaydederek eğitilmiş modele hazır olarak ulaşılır. Modelin eğitiminden sonra model üzerinde bazı analizler yapılmıştır. Modelin test verisinin ilk rakamı için yaptığı tahmin ve sonuç gösterilmiştir.

```
[14]: model.save("dig_rec.model")

[16]: new_model = tf.keras.models.load_model('dig_rec.model')

[17]: predictions = new_model.predict([x_test])

[18]: print("Tahmin Sonucu (Output Layer)")
print(predictions[0]) #düzenlenmediği için tahmin değeri dizi formatında, one hot encoding
print()
print("Tahmin Sonucu (Düzenlenmiş): ", np.argmax(predictions[0]))
print("Gerçek değer: ", y_test[0])

Tahmin Sonucu (Output Layer)
[2.4648114e-10 1.6019380e-11 3.8321399e-08 5.7601405e-06 3.5375323e-15
 9.5380726e-10 1.6831858e-16 9.9999404e-01 8.4090551e-10 5.8528077e-08]

Tahmin Sonucu (Düzenlenmiş): 7
Gerçek değer: 7
```

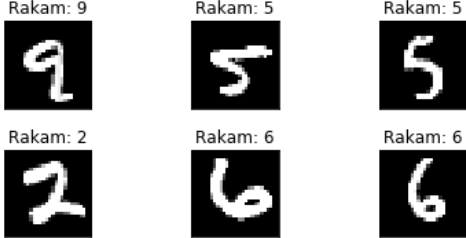
Şekil 8.9 Modelin kaydı ve ilk değer için tahmini.

Daha sonra benim tarafımdan yazılmış algoritma ile test verisinden seçilmiş rastgele 6 değer gerçek sonuçlarla karşılaştırılmıştır. Bunun için test verisindeki rastgele rakamların tahminleri yapılmış daha sonra bu tahminlerin gerçek değerleri ve görselleri hazırlanmıştır.


```
[19]: testSize = 6
random_digits = np.arange((testSize))
print("Modelin Tahminleri")
for i in range(testSize):
    random_digit = rd.randrange(0 , 10000)
    random_digits[i] = random_digit
    print(random_digit,"satırındaki rakam:",np.argmax(predictions[random_digit]))

Modelin Tahminleri
1714 satırındaki rakam: 9
935 satırındaki rakam: 5
2829 satırındaki rakam: 5
7074 satırındaki rakam: 2
937 satırındaki rakam: 6
5391 satırındaki rakam: 6

[20]: fig = plt.figure()
for j in range(testSize):
    plt.subplot(3,3,j+1)
    plt.tight_layout()
    plt.imshow(x_test[random_digits[j]], cmap = "gray", interpolation='none', vmin= 0, vmax = 0.25)
    plt.title("Rakam: {}".format(y_test[random_digits[j]]))
    plt.xticks([])
    plt.yticks([])
```



Şekil 8.10 Rastgele seçilen 6 değerin tahminleri ve gerçek değerleri.

Son olarak yine benim tarafımdan yazılan bir algoritma ile doğru tahmin edilemeyen veriler tespit edilip bunların için seçilen rastgele 6 rakam için modelin tahmini ve gerçek değerleri ile görselleri hazırlanmıştır. Bunun için test verisinin bütün tahmin sonuçları bir diziye toplanmıştır. Başarı oranı ile ne kadar verinin kaybedildiği hesaplanmış kaybedilen verilerin indeksleri bir başka diziye aktararak içinden seçilen rastgele indekslerin tahmin ve gerçek değerleri karşılaştırılmıştır.

```
[21]: pred_array = np.arange(10000)
      cnt = 0
      for k in predictions:
          pred_array[cnt] = np.argmax(k)
          cnt+= 1
      pred_array

[21]: array([7, 2, 1, ..., 4, 5, 6])

[25]: from sklearn.metrics import accuracy_score
      acc_Score = accuracy_score(y_test, pred_array)
      acc_Score

[25]: 0.9769

[47]: k = 0
      cnt = 0
      total_lost = int((100*(100 - acc_Score*100)))
      lost_digits = np.arange((total_lost))

      for i in pred_array:
          if y_test[cnt] != i:
              lost_digits[k] = cnt
              k+= 1
              cnt+=1
```

Şekil 8.11 Kayıp verilerin tespiti için hazırlık.

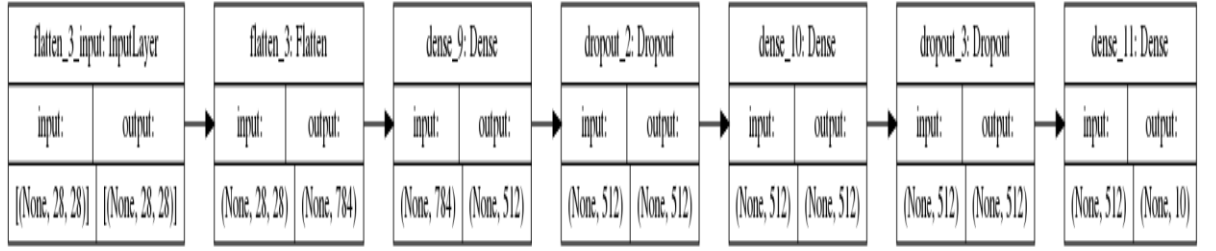
Bir sonraki görselde kayıp verilerin kıyası gösterilmiştir.

```
[57]: for i in range(testSize):
      random_digit = rd.randrange(0 , lost_digits.size)
      index = lost_digits[random_digit]
      #print("Gerçek değer:", y_test[index])
      plt.subplot(3,3,i+1)
      plt.tight_layout()
      plt.imshow(x_test[index], cmap = "gray", interpolation='none', vmin= 0, vmax = 0.25)
      plt.title("Tahmin Edilen: {}\n Gerçek değer: {}".format(pred_array[index],y_test[index]))
      plt.xticks([])
      plt.yticks([])
```



Şekil 8.12 Modelin yanlış tahmin ettiği rakamlar.

Bu uygulamada hazırlanan yapay sinir ağıları modelinin görselleştirilmiş hali.



Şekil 8.13 Modelin görsel hali.

9.SONUÇLAR VE ÖNERİLER

Yapay sinir ağıları kullanılarak rakam tanıma uygulaması on bin verilik sette %97 civarından bir başarıya ulaşmıştır. Bu oran içerisinde bir çok değişken parametre barındırmasından dolayı çalışma süresince yapılan gözlemler neticesinde %95 - %98 aralığında değişmektedir. Başarı oranının değişken olması yapay sinir ağı modellerinde sık karşılaşılan bir durumdur. Ancak başarısız tahmin sonuçları incelendiğinde bazı görsellerin kusuru çok az olmasına rağmen model tarafından yanlış tahmin edildiği fark edilmiştir. Bunun en önemli sebebi modeli eğitmek için kullanılan verilerin altmış bin rakamdan oluşmasıdır. Yapay sinir ağıları için bu rakam oldukça azdır. Yine de bu az sayıdaki veri ile daha stabil ve daha yüksek bir başarı oranı elde etmek mümkün olabilir. Çalışma süresince modelin parametreleri değiştirilerek başarı oranları incelenmiştir. Lakin tamamen aynı modelin başarı oranı bile değişiklik gösterebildiğinden bu sonuçların pek bir anlam ifade ettiği söylenemez. Sonuç olarak çalışmada hedeflenen rakam tanıma konsepti başarıya ulaşmıştır. Yapay sinir ağlarının nasıl çalıştığı, verilerin ne şekilde hazırlandığı ve görüntü işlemenin temeli bu çalışmada incelenmiştir.

10.KAYNAKLAR

Metin Kaynakları

- [1]: https://en.wikipedia.org/wiki/MNIST_database
- [2]: <https://www.oracle.com/tr/artificial-intelligence/what-is-artificial-intelligence.html>
- [3]: <https://medium.com/@nyilmazsimsek/derin-%C3%B6%C4%9Frenme-deep-learning-nedir-ve-nas%C4%B1l-%C3%A7al%C4%B1%C5%9F%C4%B1r-2d7f5850782>
- [4]: <https://medium.com/@nyilmazsimsek/derin-%C3%B6%C4%9Frenme-deep-learning-nedir-ve-nas%C4%B1l-%C3%A7al%C4%B1%C5%9F%C4%B1r-2d7f5850782>
- [5]: <https://medium.com/@nyilmazsimsek/derin-%C3%B6%C4%9Frenme-deep-learning-nedir-ve-nas%C4%B1l-%C3%A7al%C4%B1%C5%9F%C4%B1r-2d7f5850782>
- [6]: 355500-Fatih_ABA-2014-
Görüntü işleme ve yapay sinir ağları kullanarak mineral tanıma-
Yüksek Lisans Tezi
- [7]: https://tr.wikipedia.org/wiki/Yapay_sinir_a%C4%9Flar%C4%B1
- [8]:
<http://www.ibrahimcayiroglu.com/Dokumanlar/IleriAlgoritmaAnalizi/IleriAlgoritmaAnalizi-5.Hafta-YapaySinirAglari.pdf>
- [9]:
<http://www.ibrahimcayiroglu.com/Dokumanlar/IleriAlgoritmaAnalizi/IleriAlgoritmaAnalizi-5.Hafta-YapaySinirAglari.pdf>
- [10]:
<http://www.ibrahimcayiroglu.com/Dokumanlar/IleriAlgoritmaAnalizi/IleriAlgoritmaAnalizi-5.Hafta-YapaySinirAglari.pdf>
- [11]: <https://dergipark.org.tr/tr/download/article-file/391087>
- [12] http://yzgrafik.ege.edu.tr/~ugur/12_13_Spring/CI/ImageProcessing.pdf
- [13]:
http://www.ibrahimcayiroglu.com/Dokumanlar/GoruntuIsleme/Goruntu_Isleme_Ders_Notlari-1.Hafta.pdf

[14]:

http://www.ibrahimcayiroglu.com/Dokumanlar/GoruntuIsleme/Goruntu_Isleme_Ders_Notlari-1.Hafta.pdf

[15]: <https://womaneng.com/one-hot-encoding-nedir-nasil-yapilir/>

Şekil Kaynakları

şekil 1.1 https://cdn-images-1.medium.com/max/1000/1*Ft2rLuO82eItlvJn5HOi9A.png

şekil 2.1 <https://www.oracle.com/a/ocom/img/cc01-what-is-artificial-intelligence-diagram.jpg>

şekil 2.2 https://miro.medium.com/max/573/0*vHQxsXbMZlGE29KS

şekil 3.1 <https://i1.wp.com/yapayzeka.ai/wp-content/uploads/2017/07/1-5egrX-WuyrLA7gBEXdg5A.png?w=690&ssl=1>

şekil 4.1

<https://www.researchgate.net/publication/299474560/figure/fig6/AS:349583008911366@1460358492284/An-example-of-a-deep-neural-network-with-two-hidden-layers-The-first-layer-is-the-input.png>

şekil 4.2

<http://www.ibrahimcayiroglu.com/Dokumanlar/IleriAlgoritmaAnalizi/IleriAlgoritmaAnalizi-5.Hafta-YapaySinirAglari.pdf>

şekil 4.3 https://www.trzcacak.rs/myfile/full/251-2516654_4-layer-neural-network-four-layer-neural-network.png

şekil 5.1

http://www.ibrahimcayiroglu.com/Dokumanlar/GoruntuIsleme/Goruntu_Isleme_Ders_Notlari-1.Hafta.pdf

şekil 5.2

http://www.ibrahimcayiroglu.com/Dokumanlar/GoruntuIsleme/Goruntu_Isleme_Ders_Notlari-1.Hafta.pdf

şekil 5.3 355500-Fatih_ABA-2014-

Görüntü işleme_ve_yapay_sinir_ağları_kullanarak_mineral_tanıma-Yüksek_Lisans_Tezi

şekil 6.1 https://miro.medium.com/max/1006/1*6Jyg3NF-BsmSLrVGhLTasA.jpeg

şekil 7.1 <https://womaneng.com/wp-content/uploads/2018/09/onehotencoding.jpg>

şekil 8.4 https://miro.medium.com/max/1026/1*DfMRHwxYlgyyDmrIAd-gjQ.png

ÖZGEÇMİŞ

Ad Soyad	Hüsnü Mümtaz Sancak
Doğum Tarihi	05.11.1997
Doğum yeri	İstanbul
Lise	2013 – 2015 Kartal Anadolu Lisesi
Staj Yaptığı Yerler	Kanca Dövme Çelik Makine Sanayi AŞ- Kocaeli-(10 hafta) Schneider Electric - (4 hafta)