

**Tensorflow Kütüphanesi Kullanarak
Yapay Sinir Ağları İle Rakam Tanıma**

**Number Recognition With Artificial Networks
Using Tensorflow Library**

Eren Sayar

Beykent Üniversitesi, Bilgisayar Mühendisliği Öğrencisi, 34398, İstanbul, Türkiye
erensayar@yandex.com

Nisan 2020

Özet

Bu çalışmada amaç, insan eliyle çizilmiş olan rakamların bilgisayar ortamında tanınmasını sağlamaktır. Bu problemin çözümü için kullanılacak modelde tahmin, tanıma ve sınıflandırma problemlerinde günümüz şartlarında iyi bir yöntem olarak kabul edilen yapay sinir ağları yöntemi kullanılacaktır. Uygulama geliştirilmesi için yapay zeka, makine öğrenmesi, derin öğrenme çalışmaları için gerekli metotları içeren Tensorflow kütüphanesi kullanılmıştır.

Anahtar Kelimeler: OCR, Rakam Tanıma, Yapay Zeka, Makine Öğrenmesi, Derin Öğrenme, Yapay Sinir Ağları, Çok Katmanlı Yapay Sinir Ağları, Tensorflow.

Abstract

The purpose of this study is to recognize the numbers drawn by human on computer. For the solution to succeed of this problem used artificial neural networks. Artificial neural Networks is considered a good method these days for estimation, recognition and classification etc. problems. Tensorflow library is used for the development of the application in this study. Tensorflow Library contains the methods required for artificial intelligence, machine learning, deep learning

Keywords: OCR, Number Recognition, Artificial Intelligence, Machine Learning, Deep Learning, Artificial Neural Networks, Multilayer Artificial Neural Networks, Tensorflow.

İçindekiler

1.Giriş	4
1.1.Yapay Sinir Ağları Nedir?	4
1.2.Tensorflow Nedir?	4
1.3.MNIST Nedir?	4
1.4.CPU Yerine Neden GPU Kullanılır?	5
1.5.CUDA (Compute Unified Device Architecture) Nedir?	5
1.6.Literatür Araştırması.....	5
2.Yöntem	6
2.1. Çok Katmanlı Yapay Sinir Ağları	6
2.1.1.Giriş Katmanı	6
2.1.2.Ara Katman (Gizli Katman)	6
2.1.3.Çıkış Katmanı	7
3.Uygulama.....	7
3.1.Ön Hazırlık	7
3.1.1.Python İçin Tensorflow Kütüphanesinin Sisteme Yükenmesi	7
3.1.2.GPU ile Hesaplama Yapabilmek İçin Ön Hazırlık	7
3.2.Yazılım İncelemesi	8
3.2.1.Çok Katmanlı YSA'nın Kullanılması	9
3.2.2.Kodun İncelenmesi	11
3.3.Yazılımın Çalıştırılması	12
4.Sonuç Ve Öneriler	13
5.Kaynaklar	14

1.Giriş

İnsanlık yazı ve rakam adı verilen anlam yüklediği belirli şekiller dizisinin kullanımını yüz yıllardır sürdürmektedir. İnsan algısı yazı ve rakamlara anlam vermede ve tanımlamada zorlanmamaktadır. Yazılan rakam ve yazılar her zaman birebir aynı değildir ancak insan algısı okumada başarı göstermektedir.

Bilgisayarların bu yazı ve rakamları tanımasını sağlamak, dijital ortama text olarak bu rakam ve yazıları aktarabilmek birçok amaç için kullanılabilir. Plaka tanıma, el yazılarını dijital ortamda text olarak depolama ve bazı otomatize sistemlerin çalışması gibi birçok amaç için gereklidir.

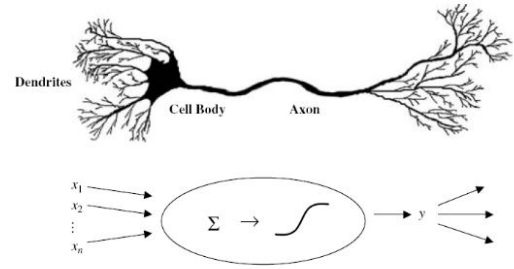
Bilgisayarların yapısı itibarıyla bilgisayarlar için, el yazısı olan rakam ve yazıları tanıma işleminde akla gelen ilk husus rakamların, yazıların bire bir aynı olmadığıdır. Bu farklılıklardan ötürü bilgisayarın farklı ama benzer bir formda olan şekilleri insanlar gibi algılaması için yapay sinir ağı yöntemi kullanılmıştır.

Bir öğrenme modeli olan yapay sinir ağı insanların öğrenme yolundan esinlenerek ortaya çıkmıştır.

Bu projede de amaç insan eliyle yazılan rakamların bilgisayar ortamında tanınmasıdır. Bilgisayarın rakamları tanıyabilmesi için de yapay sinir ağı yöntemi kullanılmıştır.

1.1.Yapay Sinir Ağları Nedir?

Yapay sinir ağları (YSA), insan beyninin öğrenme yolunu taklit ederek beynin öğrenme, hatırlama, genelleme yapma yolu ile topladığı verilerden yeni veri üretebilme gibi temel işlevlerin gerçekleştirildiği bilgisayar yazılımlarıdır. Yapay sinir ağları; insan beyninden esinlenerek, öğrenme sürecinin matematiksel olarak modellenmesi uğraşı sonucu ortaya çıkmıştır.[1]



Şekil 1: Biyolojik sinir hücresi ve yapay sinir ağı

1.2.Tensorflow Nedir?

TensorFlow, özellikle yapay zeka, makine öğrenmesi, derin öğrenme için kullanılan ücretsiz ve açık kaynaklı bir yazılım kütüphanesidir. TensorFlow, Google Brain ekibi tarafından Google projelerinde kullanımı için geliştirilmiştir.9 Kasım 2015'te Apache Lisansı 2.0 altında yayınlanmıştır, Google'da hem araştırma hem de üretim için kullanılmaktadır. Tensorflow şu an itibarıyla C++, Python, Javascript ve Java dilleri için hazır kütüphane olarak kullanılabilir.

1.3.MNIST Nedir?

MNIST bir dataset. Çeşitli görüntü işleme sistemlerinin eğitimi için yaygın olarak kullanılan insanlar tarafından yazılan yazı ve rakamları barındıran veri setidir. MNIST dataseti NIST datasetinin bir alt kümesidir. NIST datasetinde olan veriler MNIST datasetinde olan verileri kapsar. Kullanılan dataset; Corinna Cortes (Google Labs, New York) ve Christopher JC Burges (Microsoft Araştırma, Redmond) tarafından normalizasyon ve çeşitli işlemler yapılarak oluşturulan datasetin Yann LeCun (Courant Enstitüsü, NYU) tarafından özelleştirilmiş versiyonudur. Bu dataset 60.000'i eğitim için 10.000'i ise test için kullanılacak olan veri setine sahiptir. Resimler 28x28 piksel oranına sahip 784 piksel değerine sahiptir. Bu resimler elle çizilen rakamları içermektedir.[2]

1.4.CPU Yerine Neden GPU Kullanılır?

GPU yapısal olarak paralel hesaplamalar için uygundur. Sebebi GPU'lar CPU'lara göre çok daha fazla çekirdeğe sahiptir. CPU'nun GPU'ya göre üstünlüğü ise tek bir çekirdeğin GPU'daki bir çekirdeğe oranla çok daha yüksek saat hızına sahip olmasıdır. Günümüz şartlarında CPU'lar genel kullanım için 2, 4 veya 8 çekirdekliken, server, iş istasyonu ve benzeri çok işlem gücü gerektiren sistemlerde kullanımlar için üretilen CPU'lar bile bugün için en fazla 64 fiziksel çekirdeğe sahip. GPU'lar ise çok daha fazla sayıda çekirdeğe sahiptir. Bu günün şartlarında en iyi ekran kartlarından olan, Nvidia Titan V 5120 çekirdeğe sahipken, gene bugün için epey güçlü bir ekran kartı olarak kabul edilen Nvidia GTX 1080Ti ise 2560 çekirdeğe sahiptir. Bu çalışmada projenin geliştirilmesi için kullanılan bilgisayarda bulunan GPU (Nvidia MX 250) ise 384 çekirdeğe sahiptir. Çekirdek sayısı paralel işlemlerde önemli bir faktördür.

Tensorflow bugünün şartlarında yalnızca Nvidia GPU'lara destek vermektedir. Bu GPU'lar CUDA teknolojisi ile NVIDIA tarafından desteklenmektedir. Bu projede de GPU ile hesaplama gerçekleştirilecektir.

1.5.CUDA (Compute Unified Device Architecture) Nedir?

Bir yazılım. CUDA, GPU için NVIDIA'nın sunduğu C programlama dili üzerinde eklenti olarak kullanıma sunulan bir mimari ve teknolojidir. PathScale tabanlı bir C derleyicisi ve C ile yazılmış algoritmaların GPU üzerinde çalışmasını sağlayan geliştirme araçları kümesidir.

1.6.Literatür Araştırması

Yapılan literatür araştırması sonucu karakter tanıma çalışmalarının çokça yapıldığı birçok

araştırma yürütüldüğü gözlemlenmiştir. Bu yapılan çalışmalarda öğrenme işleminin en çok yapay sinir ağları kullanılarak yapıldığı da gözlemlenmiştir. İncelenen çalışmaların bir kısmı şu şekildedir;

•*Gradient Based Learning Applied To Document Recognition – [Yann LeCun, Yashua Bengio, Patrrick Haffner]*

1998 yılında Proceedings of the IEEE akademik dergisinde yayınlanmıştır. Önemli bir referans çalışmadır. Bunun sebeplerinden biri kullanılan MNIST datasetini oluşturan Yann Lecun'un makaleye katkısının olması ve MNIST datasetinin çalışma içerisinde kullanılmış olmasıdır. Ayrıca makale konuyla alakalıdır. OCR işlemleri üzerinde durulmuştur ve birçok yöntem de bu çalışma sırasında geliştirilmiştir. Önemli bir çalışmadır. Çok katmanlı yapay sinir ağları çalışma içerisinde kullanılmıştır.

•*Algorithmic And Mathematical Principles Of Automatic Plate Recognition Systems - Ondrej Martinsky - (BRNO University Of Technology Faculty Of Information Technology - Department Of Intelligent System)*

JavaANPR adlı çalışma. Kendini duyurmayı başaran bu çalışma Java dili ile Çekya'da bulunan BRNO University bünyesinde Ondrej Martinsky tarafından geliştirilmiştir. Bu çalışmada yapay zeka bir plaka tanıma sistemi geliştirilmiştir. Karakter tanımda yapay sinir ağları kullanılmıştır. Gerekli matematiksel ilkeler ve algoritmalar geliştirilmiştir.

•*Multi-digit Number Recognition from Street View Imagery using Deep Convolutional Neural Networks - [Ian J. Goodfellow, Yaroslav Bulatov, Julian Ibarz, Sacha Arnoud, Vinay Shet] - April2014*

Google şirketi bünyesinde Google çalışanları tarafından 2014 yılında yürütülmüş olan bu çalışma sokak görünümü projelerinde kapı numaralarını veya daha başka rastgele çok

karakterli metinleri bilgisayar ortamında tanıma amacıyla yürütülmüş bir çalışmadır.

•*Poker-DVS and MNIST-DVS. Their History, How They Were Made, and Other Details - [TeresaSerrano-Gotarredonaand Bernabé Linares - Barranco] – December 2015*

Frontiers açık bilim platformunda 2015 yılında yayınlanan çalışma. Bu çalışmada iki dataset üretilmiştir ikisi de poker kartlarını içerir ancak kartların okuma şekli farklıdır. İlk dataset için bir dinamik görme sensörü (DVS) kullanarak 20-30 ms boyunca kartlar okunarak ilk dataset oluşturulmuştur. Datasetin adı Poker-DVS, bu dataset ilk datasettir. İkinci dataset ise LCD bir ekrandan 2-3 saniye boyunca okunarak kart dataseti elde edilmiştir. Bu datasete ise MNIST-DVS ismi verilmiştir. Bu çalışma daha çok dataset oluşturulmasını incelemeye yöneliktir, iki datasetin yönlerini, nasıl çalıştıklarını, ayrıntılarını incelemektedir.

•*Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms – [Han Xiao, Kashif Rasul, Roland Vollgraf] - September 2017*

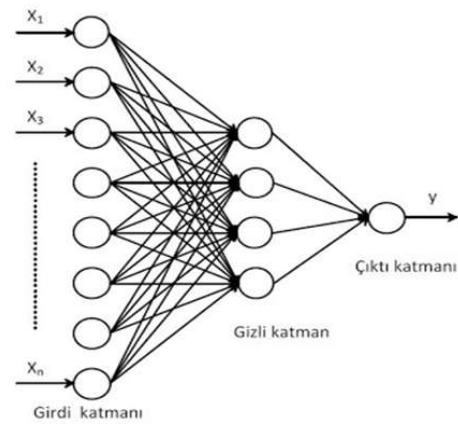
Bu çalışma ismi MNIST-Fashion. Şu anda yapılan çalışmayla yönrtem olarak fazla benzerlik gösteren bir çalışmadır. Rakam okuma yerine 10 farklı türden oluşan (10 farklı rakam gibi) giysilerin tanınması çalışmasıdır. Ancak bu çalışmada asıl üzerinde durulan nokta sınıflandırma modellerinin karşılaştırılması işlemidir. Ölçümlenen bazı algoritmalar şu şekildedir: Extra Tree Classifier, Gaussian NB, Gradient Boosting Classifier, Logistic Regression, Perceptron...

2.Yöntem

İstenen işlemin gerçekleşmesi için çok katmanlı yapay sinir ağları yöntemi kullanılmıştır. Bu yapı Tensorflow içinde metodlar aracılığıyla kullanıma sunulmuştur.

2.1. Çok Katmanlı Yapay Sinir Ağları

Belirsiz, gürültülü ve eksik bilgilerin işlenmesinde yapay sinir ağları başarıyla kullanılmaktadır. YSA'lar, ağırlıklandırılmış şekilde birbirlerine bağlanmış birçok işlem biriminden (nöronlar) oluşan matematiksel sistemlerdir. İşlem birimi, transfer fonksiyonu olarak anılan bir denklemdir. Bu işlem birimi, diğer nöronlardan sinyalleri alır; bunları birleştirir, dönüştürür ve sayısal bir sonuç ortaya çıkartır. Genelde, işlem birimleri kabaca gerçek nöronlara karşılık gelirler ve bir ağ içinde birbirlerine bağlanırlar; bu yapı da sinir ağlarını oluşturmaktadır.[1]



Şekil 2: İleri Beslemeli Ve Çok Katmanlı Yapay Sinir Ağı

2.1.1.Giriş Katmanı

Bu katmandaki işlem elemanları, dışarıdan bilgileri alarak bir sonraki katman olan ara katmanlara transfer ederler. Giriş katmanındaki nöron sayısı bir sınıflandırma çalışması için sınıf sayısı kadar olmalıdır.[1]

2.1.2.Ara Katman (Gizli Katman)

Giriş katmanından gelen bilgiler işlenerek çıkış katmanına gönderilirler. 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 nöron 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.[1]

2.1.3.Çıkış Katmanı

Çıkış katmandaki işlem elemanları ara katmandan gelen bilgileri işleyerek ağı girdi katmanından sunulan girdi seti için üretilmesi gereken çıktıyı üreten katmandır ve üretilen çıktı dışarıya gönderilir. Geri beslemeli ağlarda bu katmanda üretilen çıktı kullanılarak ağı yeni ağırlık değerleri hesaplanır. [1]

*Uygulama içinde nasıl kullanıldığı uygulama başlığı altında olan 3.2.1 başlığında detaylandırılmıştır.

3.Uygulama

Programlama dili olarak Python seçilmiştir. Python projesine tensorflow kütüphanesi import edilerek yapay sinir ağı metodları kullanılıp uygulama gerçekleştirilmiştir. Yapay sinir ağları yönteminde CPU yerine GPU kullanılması çok daha hızlı sonuç vereceği için GPU kullanılarak rakam tanıma yapılmıştır.

3.1.Ön Hazırlık

Sistemde GPU ile hesaplama yapmak için ön hazırlıklar gerçekleştirilmiştir. Tensorflow, python paket yöneticisi (PIP) ile sisteme yüklenmiş, çalışma klasörü içinde python dosyası açılmış ve Tensorflow import edilmiştir.

3.1.1.Python İçin Tensorflow Kütüphanesinin Sisteme Yüklenmesi

Python paket yöneticisi PIP ile tensorflowu sisteme yüklenilmektedir. Tensorflow kendi sitesinde yükleme işlemlerini detaylı olarak

install dizini altında açıklamıştır. Bu projede install dizini altında pip ile yükleme işlemi tercih edilmiştir. Bazı ön şartlar vardır bunlar tamamlanır ve tensorflowun son sürümünün yüklenmesi için terminal açılıp şu komut girilir;

```
pip3 install --user --upgrade tensorflow
```

Bu komut ile tensorflow sisteme yüklenmiş olacaktır. Python'da sistem interpreter'ı kullanılırsa tensorflow kütüphanesini import ederken sıkıntı yaşanmaz.

Bu projede sisteme yüklenmesi yerine virtual envoirement içine kurulmuştur zira işlemler oradan yürütülecektir, bu pek önemli olmayan küçük bir ayrıntıdır. Ayrıca Tensorflow 1.15 sürümü kullanılmıştır. Gpu desteği için önceki sürümlerde gpu parametresi eklenmelidir; "tensorflow-gpu==1.15". Sırayla izlenen yollar şu şekildedir;

- Virtual envoirement oluşturulmuştur:
>**virtualenv -p python3 ./venv**
- Venv içindeyken 1.15 sürümü yüklenmiştir:
>**pip install tensorflow-gpu==1.15**

3.1.2.GPU İle Hesaplama Yapabilmek İçin Ön Hazırlık

İlk olarak Tensorflow resmi sitesinde Install kısmından GPU Support kısmına girilir ve yönergeler takip edilerek işlem sonlanır. Not olarak Tensorflow resmi sitesinde GPU desteğinin yalnızca Ubuntu dağıtımı ve Windows işletim sistemi için kullanılabilir olduğu belirtilmiştir. Yönergeler tensorflow yüklendikten sonra sırayla gerçekleştirilir.

Zorunlu yazılım kurulumları gerçekleştirilmesi gerekmektedir sonrasında ise işletim sistemine özgü son işlemler gerçekleştirilir ve PATH değişkenleri ayarlanır. Yazılımlar sırayla şu şekildedir:

- NVIDIA GPU Drivers
- CUDA Toolkit
- CUPTI
- cuDNN SDK
- TensorRT 6.0 (İsteğe Bağlı)

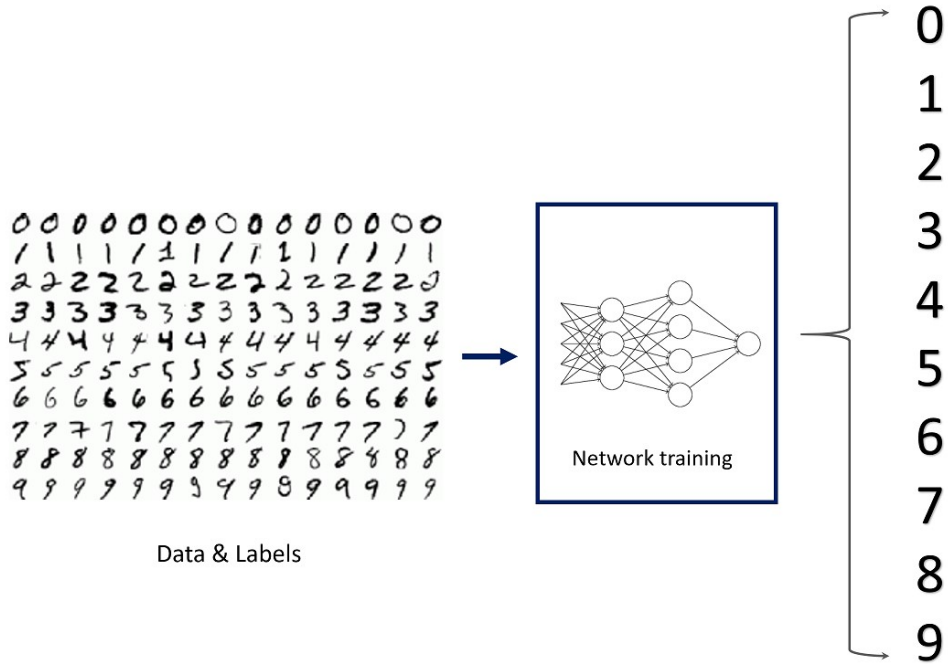
(Şu anki tarihte 10.2 çıkmıştır ancak tensorflow 10.1 desteği vermekte) (Tensorflow 1.15 sürümü içinse 10.0 yüklenmeli)

Ayarların doğru yapıldığından emin olmak için son olarak bir komut çalıştırılabilir. Çıktının başında “Successfully opened dynamic library cudart64_101.dll”(100, 101, 102.dll: dll dosyalarının isimleri sürüme göre değişir) mesajı ve çıktı sonunda GPU sayısı verirse başarılı demektir. Kod şu şekildedir;

```
import tensorflow as tf
print("Num GPUs Available: ", len(tf.config.experimental.list_physical_devices('GPU')))
```

3.2.Yazılım İncelemesi

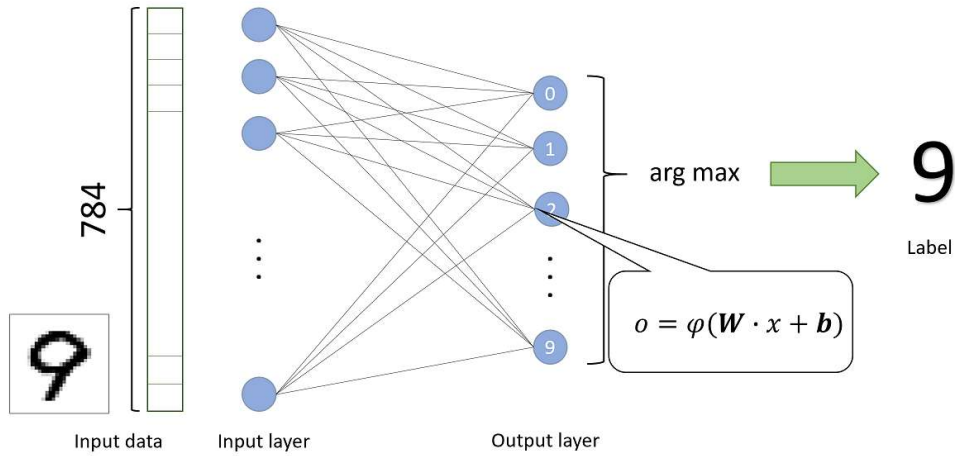
MNIST dataseti 28x28 genişliğinde toplam 784 pixel boyutunda olan el ile çizilmiş rakamlar barındırır. Bu projede 60.000 veri eğitim verisi olarak kullanılmıştır. Bu işlem ideal ağırlık kat sayısının ve sapmanın bulunmasıdır. Sonrasında 10.000 veri ise test verisi olarak kullanılmıştır. Bu işlemin sonucunda accuracy oranı elde edilir ve var olan verileri yazılıma verip bir sonuç elde edilebilir.



Şekil 3: Yazılım Çalışma Şekli

3.2.1.Çok Katmanlı YSA'nın Kullanılması

Uygulama içerisinde kullanılan yapay sinir ağını ve kullanım şeklini detaylandırmak gerekirse;

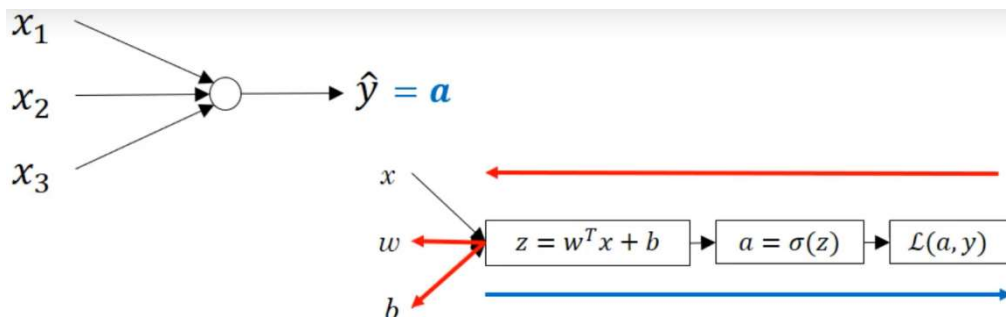


Şekil 4: YSA'nın uygulamada kullanım şeması

28x28 genişliğinde toplam 784 piksel array olarak alınıp, sıra sıra düğümlere(nöronlara) aktarılır. Input layer'da 784 düğüm mevcuttur. En son çıkış katmanında ise 10 düğüm vardır. Çıkış katmanında işaret edilen düğüm(0,1,2,.....,9) sonucu verir.

Yukarıdaki formülde; W(Ağırlık) ile X vektörü çarpılır bias eklenir sonra probleme uygun seçilmiş aktivasyon fonksiyonu ile çarpılır.(φ) Sonrasında ise loss fonksiyonuna (maliyet fonksiyonu) aktarılır.

a.Öğrenme İşlemi: Bu işlem sonrasında loss(veya maliyet) fonksiyonundan bir sayı değeri döndürülür amaç bu sayının olabildiği kadar küçülmesidir. Sayı küçüldükçe öğrenme başarısı artmaktadır ve küçüldükçe öğrenme gerçekleşiyor demektir. Geriye doğru döndürmek için işlemler tersten yapılır. Örneğin aktivasyon fonksiyonunun türevi alınır, bir sonraki aşamada ters işlem yapılır. En son geriye döndüğümüzde yeni ağırlık değerleri elde edilir ve işlem tekrar baştan yapılır ve tekrar loss fonksiyonu bir değer döndürür amaç bu değerin azaltılmasıdır.[7]



Şekil 5: Yapay Sinir Ağları Eğitim Yöntemi

b.Loss Fonksiyonu: Matematiksel optimizasyon ve karar teorisinde, bir kayıp fonksiyonu veya maliyet fonksiyonu. Bir veya daha fazla değişkenin bir olayını veya değerlerini, sezgisel olarak olayla ilişkili bir "maliyeti" temsil eden gerçek bir sayıya eşleyen bir fonksiyondur. [7] Yani hata oranı gibi bir sayı döner, amaç bu sayıyı düşürmektir. Birçok loss değeri hesaplayan fonksiyon mevcuttur bu projede softmax ve çapraz entropi kullanılmıştır.

c. Geri yayılım Algoritması: Geri yayılım sinir ağındaki ağırlıkları güncellemek için kullanılan ve bunu yaparken de asıl sonuç ile istenilen sonucu hesaba katan bir yöntemdir.[7] Ağırlık w değerine göre türev, zincir kuralı kullanılarak hesaplanır ve aşağıdaki şekildedir;

$$\frac{\partial L(a, y)}{\partial w} = \frac{\partial L(a, y)}{\partial u} \times \frac{\partial u}{\partial a} \times \frac{\partial a}{\partial w}$$

Sonuç olarak, ağırlık güncellenmesinin son hali şu şekildedir:

$$\text{Yeni } W = w - \alpha \frac{\partial L(a, y)}{\partial w}$$

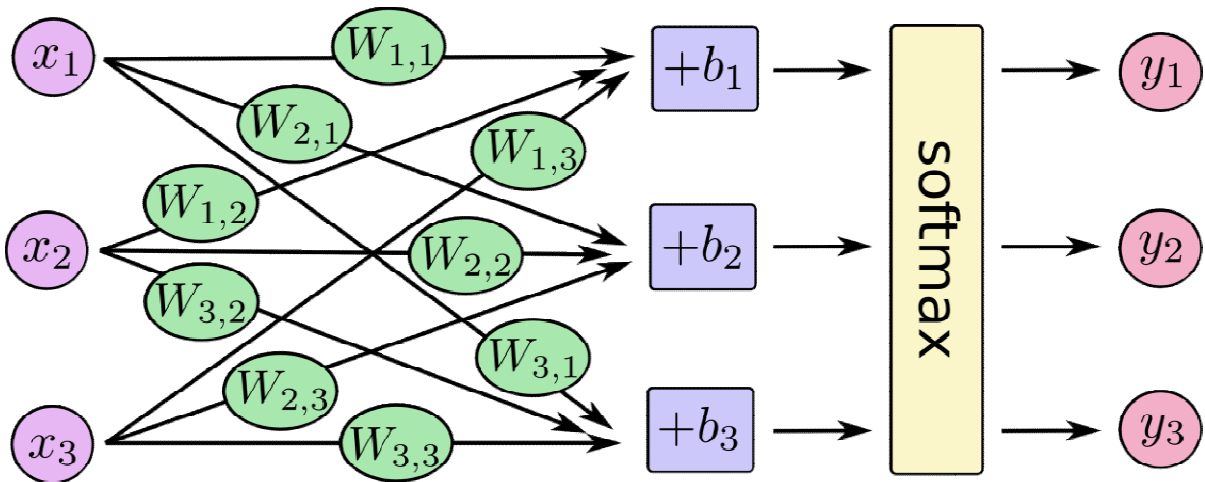
Ağırlıkları güncelleme — Sinir ağında ağırlıklar, aşağıdaki gibi güncellenir:

- 1. Adım: Bir eğitim verisi kümesi alınır.
- 2. Adım: Denk gelen kaybı elde etmek için, ileri yayılım gerçekleştirilir.
- 3. Adım: Gradyanları elde etmek için kayba geri yayılım uygulanır.
- 4. Adım: Ağın ağırlıklarını güncellemek için gradyanlar kullanılır.

d.Çapraz Entropi: Çapraz-entropi kaybı. Sinir ağıları içeriğinde, çapraz-entropi kaybı $L(a, y)$ sık olarak kullanılır bir loss fonksiyonudur. Çapraz entropi kaybı veya log kaybı(log loss), çıktısı 0 ile 1 arasında bir olasılık değeri olan bir sınıflandırma modelinin performansını ölçer. Tahmin edilen olasılık asıl değerden uzaklaştıkça çapraz entropi kaybı artar. Aşağıdaki gibi tanımlanır:

$$L(a, y) = -[y \log(a) + (1 - y) \log(1 - a)]$$

e. Softmax: Softmax fonksiyonu yapay sinir ağı tarafından üretilen skor değerlerini kullanarak olasılık temelli loss değeri üretmektedir. Softmax sonucunda test girdisinin her bir sınıfa ait benzerliği için olasılık değeri üretilir. Yapay sinir ağı modelinin çıktı olarak verdiği skor değerler normalize edilmemiş değerlerdir. Softmax bu değerleri normalize ederek olasılık değerlerine dönüştürmektedir. Özetle Softmax fonksiyonunda amaç test verisinin doğru sınıfı için log likelihood değerini maksimize etmektir. Loss fonksiyonunda amaç ise negative likelihood minimize etmektir; yani en az farka sahip, en çok benzerliği sahip sınıfı bulmaktır.



Şekil 6: Softmax

3.2.2.Kodun İncelenmesi

```
#Kendi çizimimizi okuyabilmek için OpenCV kütüphanesi import edilmiştir.
import cv2
#Array işlemleri için numpy import edilmiştir.
import numpy as np
import tensorflow as tf

#Dataset tensorflow paketinden import edilir.
from tensorflow.examples.tutorials.mnist import input_data
#Dataset okunur
mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)

#Placeholder'lar tanımlanır
#Eğitim sürecinde karşılaştırmalar yapılacaktır.
#Bu yüzden placeholder eklendi, geçici değerler gibi düşünülebilir.
x = tf.placeholder(tf.float32, [None, 784]) #Softmax regresyonu için
y_ = tf.placeholder(tf.float32, [None, 10]) #Çapraz entropi için

#Ağırlıklar ve sapmaların ayarlanması için değişkenler.
W = tf.Variable(tf.zeros([784, 10]))
b = tf.Variable(tf.zeros([10]))

#Çıktı katmanı ağı oluşturulur
#Softmax regresyonu kullanılır
y = tf.nn.softmax(tf.matmul(x, W) + b)

#Çapraz Entropi algoritması.[Döküman içerisinde olan L(a,y) loss fonksiyonu.]
cross_entropy = tf.reduce_mean(-tf.reduce_sum(y_ * tf.log(y), reduction_indices=[1]))

#Geri yayılım algoritması ve gradyan inişi(0,5 oranı ile) gerçekleşir(Ağırlıklar optimize edilir)
train_step = tf.train.GradientDescentOptimizer(0.5).minimize(cross_entropy)

#Model başlatılır
sess = tf.InteractiveSession()
tf.global_variables_initializer().run()

#Eğitim işlemi gerçekleşir.(1000 kez)
for _ in range(1000):
    batch_xs, batch_ys = mnist.train.next_batch(100)
    sess.run(train_step, feed_dict={x: batch_xs, y_: batch_ys})

#Modelin ne kadar iyi olduğu değerlendirilir.
#Accuracy hesaplanır
correct_prediction = tf.equal(tf.argmax(y,1), tf.argmax(y_,1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))

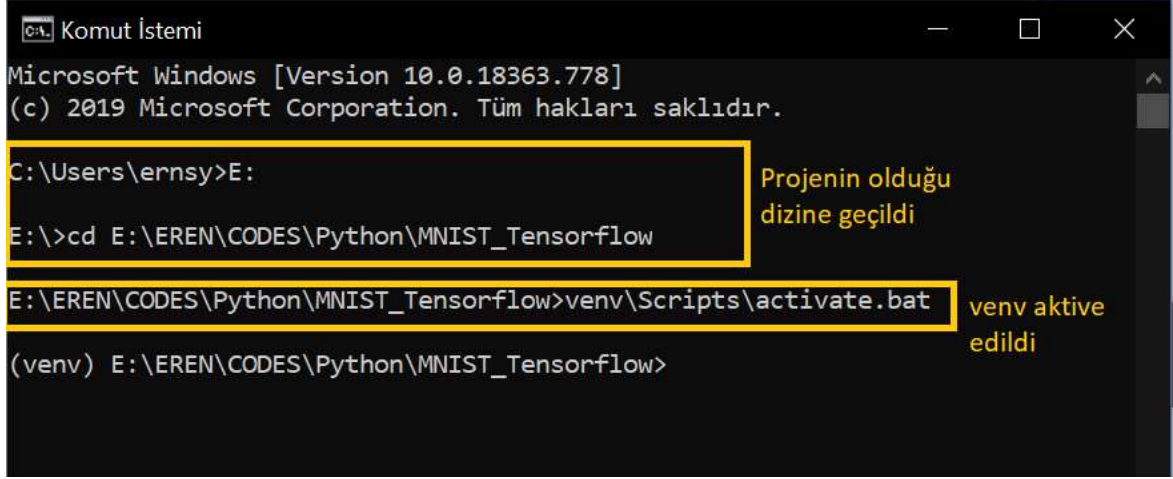
#Accuracy oranı çıktısı verilir.
print(sess.run(accuracy, feed_dict={x: mnist.test.images, y_: mnist.test.labels}))

#Kendi çizimim OpenCV aracılığıyla gri renk düzeyinde okunur.
#Tahmin gerçekleştirilir, sonuç çıktısı verilir.
cizim = np.vectorize(lambda x: 255 - x) (np.ndarray.flatten
(cv2.imread('E:\EREN\CODES\Python\MNIST_Tensorflow\Rakam1.png', cv2.IMREAD_GRAYSCALE)))
sonuc = sess.run(tf.argmax(y, 1), feed_dict={x: [cizim]})
print(sonuc)
```

3.3.Yazılımın Çalıştırılması

Yazılımın çalıştırılması için python dosyasının bulunduğu dizinde oluşturulan virtual envoirement aktif edilir. Sonrasında >python MNIST-Tensorflow.py yazarak kod çalıştırılır. (MNIST-Tensorflow dosya ismi) Kod içerisinde resmin ismi ve hangi dizinde olduğu belirtilmiştir. Dosya okunur ve terminal üzerinde sonuç elde edilir. Sırayla işlemler şu şekildedir;

- **Virtual Envoirement Aktif Edilir:** İlk olarak proje dizinine girilir sonrasında venv/Scripts/activate.bat aktif edilir. Linux içinse activate dosyası çalıştırılır. Bu işlem zorunlu değildir ancak daha sağlıklıdır.



```
Komut İstemi
Microsoft Windows [Version 10.0.18363.778]
(c) 2019 Microsoft Corporation. Tüm hakları saklıdır.

C:\Users\ernsy>E:

E:\>cd E:\EREN\CODES\Python\MNIST_Tensorflow

E:\EREN\CODES\Python\MNIST_Tensorflow>venv\Scripts\activate.bat

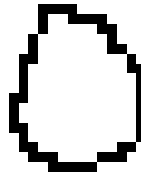
(venv) E:\EREN\CODES\Python\MNIST_Tensorflow>
```

Projenin olduğu dizine geçildi

venv aktive edildi

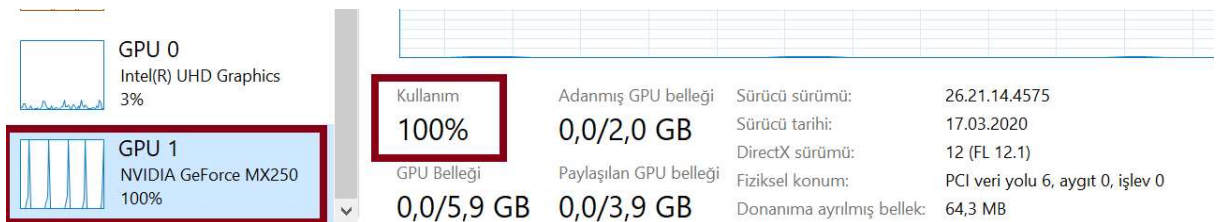
Şekil 7: Virtual Envoirement Aktivasyon, Bu işlem ile virtual envoirement içinde bulunan kütüphaneler kullanılabilir.

- **Proje Çalıştırılır:** python <python_dosyasi_ismi.py> komutuyla çalıştırılır. Kodda 0 rakamının tahmini gerçekleştirilmiştir. Bunun için Rakam0.png isimli kendi çizdiğim 28x28 piksellik 0 rakamı okunmuştur.



Şekil 8: 28x28 piksel boyutlu deneme amaçlı çizilen 0 rakamı

Ayrıca yazılan program çalıştırılırken GPU'ya da bir yük bindiği gözlemlenmektedir. GPU'nun yaptığı işlemin görülmesi amacıyla 5 kere çalıştırılmıştır ve grafikte de 5 kere artış olduğu görülmektedir.



Şekil 9: Görev Yöneticisinden GPU Kaynakları

Çıktıda önemli kısımlar şu şekildedir:

```
(venv) E:\EREN\CODES\Python\MNIST_Tensorflow>python MNIST-Tensorflow.py
2020-04-19 16:06:45.642116: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library cudart64_100.dll
```

Şekil 10: CUDA başarılı şekilde çalışıyor. Çalışması gereken dll dosyası çalıştı.

```
2020-04-19 16:06:49.469024: I tensorflow/core/platform/cpu_feature_guard.cc:142] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX2
2020-04-19 16:06:49.473536: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library nvcuda.dll
2020-04-19 16:06:50.080850: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1618] Found device 0 with properties:
  name: GeForce MX250 major: 6 minor: 1 memoryClockRate(GHz): 1.0375
  pciBusID: 0000:06:00.0
2020-04-19 16:06:50.086179: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library cudart64_100.dll
2020-04-19 16:06:50.090862: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library cublas64_100.dll
2020-04-19 16:06:50.095544: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library cufft64_100.dll
2020-04-19 16:06:50.098794: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library curand64_100.dll
2020-04-19 16:06:50.106950: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library cusolver64_100.dll
2020-04-19 16:06:50.111614: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library cusparse64_100.dll
2020-04-19 16:06:50.124066: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library cudnn64_7.dll
2020-04-19 16:06:50.126329: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1746] Adding visible gpu devices: 0
2020-04-19 16:06:50.594075: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1159] Device interconnect StreamExecutor with strength 1 edge matrix:
2020-04-19 16:06:50.596201: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1165] 0
2020-04-19 16:06:50.597716: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1178] 0: N
2020-04-19 16:06:50.599681: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1304] Created TensorFlow device (/job:localhost/replica:0/task:0/device:GPU:0 with 1356 MB memory) -> physical GPU (device: 0, name: GeForce MX250, pci bus id: 0000:06:00.0, compute capability: 6.1)
WARNING:tensorflow:From MNIST-Tensorflow.py:34: The name tf.global_variables_initializer is deprecated. Please use tf.compat.v1.global_variables_initializer instead.
2020-04-19 16:06:50.711024: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library cublas64_100.dll
0.9181 Accuracy GPU ile ilgili
[0] Yazılımın Rakam Tahmini CUDA'dan
dönen mesajlar
```

Şekil 11: Rakam Tahmini Başarıyla Gerçekleşmiştir

4.Sonuç Ve Öneriler

Proje Tensorflow kütüphanesi kullanılarak yapay sinir ağları yöntemiyle rakam tanıma işlemini %92 başarı oranıyla gerçekleştirmiştir. %92 başarı oranı iyi gibi görünse de pek de iyi bir sonuç değildir. En iyi modeller %99.7 oranlarına çıkabilmektedir. Bu projede ise bazı küçük değişiklikler ile %97 oranına çıkabilmek mümkündür. Modelin tekrar gözden geçirilip yeniden modellenmesi ile çalışma bu yönde ilerletilebilir.

Çalışma, rakam okuma işlemlerinin gerektiği projelerde kullanılabilir. Plaka okuma, yazı okuma (OCR) işlemlerine de uyarlanabilir bir model olmuştur dolayısıyla yazı okuma için de uygun veri setleri ile benzer bir proje oluşturulabilir.

5.Kaynaklar

- [1]YAPAY SİNİR AĞLARI PAPATYA YAYINCILIK - Prof.Dr. Ercan ÖZTEMEL - ISBN: 978-975-6797-39-6
- [2]Gradient Based Learning Applied To Document Recognition – Yann LeCun, Yashua Bengio, Patrrick Haffner
- [3]Algorithmic And Mathematical Principles Of Automatic Plate Recognition Systems - Ondrej Martinsky (BRNO University Of Technology Faculty Of Information Technology - Department Of Intelligent System)
- [4] Multi-digit Number Recognition from Street View Imagery using Deep Convolutional Neural Networks - [Ian J. Goodfellow, Yaroslav Bulatov, Julian Ibarz, Sacha Arnoud, Vinay Shet] - 14April2014
- [5]Poker-DVS and MNIST-DVS. Their History, How They Were Made, and Other Details - [TeresaSerrano-Gotarredonaand Bernabé Linares - Barranco] – December 2015
- [6]Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms – [Han Xiao, Kashif Rasul, Roland Vollgraf] - September 2017
- [7]Stanford University Makine Öğrenmesi Dersi CS-229, Derin Öğrenme El Kitabı, Sinir Ağları
- [8] ERDEM, O. Ayhan, and U. Z. U. N. Emre. "YAPAY SİNİR AĞLARI İLE TÜRKÇE TIMES NEW ROMAN, ARIAL VE ELYAZISI KARAKTERLERİ TANIMA." Gazi Üniversitesi Mühendislik-Mimarlık Fakültesi Dergisi 20.1 (2005).
- [9] Yapay Sinir Ağları ve Tanıma Sistemleri [Halit Ergezer, Mehmet Dikmen ve Erkan Özdemir] (P I V O L K A, Yıl:2 Sayı:6, Sayfa:14)
- [10] T. Boulton, R.A. Melter, F. Skorina, and I. Stojmenovic, "G-neighbors", Proceedings of the SPIE Conference on Vision Geometry II, pages 96-109, 1993.
- [11] B-LIGHT: A Reading aid for the Blind People using OCR and OpenCV [Mallapa D.Gurav , Shruti S. Salimath , Shruti B. Hatti, Vijayalaxmi I. Byakod , Shivaleela Kanade] (International Journal of Scientific Research Engineering & Technology (IJSRET), ISSN 2278 – 0882 Volume 6, Issue 5, May 2017)