T.C.
NUH NACİ YAZGAN ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
ELEKTRİK-ELEKTRONİK
MÜHENDİSLİĞİ
ANABİLİM DALI

GIDALARDAKİ KARBONHİDRAT MİKTARININ TESPİTİ İÇİN BÖLGE TABANLI EVRİŞİMSEL SİNİR AĞI TEMELLİ GÖRÜNTÜ İŞLEME MODELİ

(Yüksek Lisans Tezi)

Hazırlayan

Hüseyin HAKKOMAZ

Danışman

Doç. Dr. Zeki ORALHAN

MART 2022

KAYSERİ

T.C.
NUH NACİ YAZGAN ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
ELEKTRİK-ELEKTRONİK
MÜHENDİSLİĞİ
ANABİLİM DALI

GIDALARDAKİ KARBONHİDRAT MİKTARININ TESPİTİ İÇİN BÖLGE TABANLI EVRİŞİMSEL SİNİR AĞI TEMELLİ GÖRÜNTÜ İŞLEME MODELİ

(Yüksek Lisans Tezi)

Hazırlayan

Hüseyin HAKKOMAZ

Danışman

Doç. Dr. Zeki ORALHAN

MART 2022

KAYSERİ

BİLİMSEL ETİĞE UYGUNLUK

Bu çalışmadaki tüm bilgilerin, akademik ve etik kurallara uygun bir şekilde elde edildiğini beyan ederim. Aynı zamanda bu kural ve davranışların gerektirdiği gibi, bu çalışmanın özünde olmayan tüm materyal ve sonuçları tam olarak aktardığımı ve referans gösterdiğimi belirtirim.

HÜSEYİN HAKKOMAZ

YÖNERGEYE UYGUNLUK ONAYI

"Bilgisayarlı görü ve makine öğrenmesi yöntemleri kullanılarak tip 1 diyabet hastaları için yemeklerdeki karbonhidrat miktarını tahmin eden uygulama tasarımı" adlı Yüksek Lisans tezi, Nuh Naci Yazgan Üniversitesi Lisansüstü Tez Yazım Yönergesi'ne uygun olarak hazırlanmıştır.

Tezi Hazırlayan	Tez Danışmanı
Hüseyin HAKKOMAZ	Doç. Dr. Zeki ORALHAN

KABUL VE ONAY SAYFASI

Doç. Dr. Zeki ORALHAN danışmanlığında Hüseyin HAKKOMAZ tarafından hazırlanan "Bilgisayarlı görü ve makine öğrenmesi yöntemleri kullanılarak tip 1 diyabet hastaları için yemeklerdeki karbonhidrat miktarını tahmin eden uygulama tasarımı" adlı bu çalışma jürimiz tarafından Nuh Naci Yazgan Üniversitesi Fen Bilimleri Enstitüsü Elektrik Elektronik Mühendisliği Anabilim Dalında **yüksek lisans tezi** olarak kabul edilmiştir.

…….. /…… / ……..

Tez savunma sınav tarihi yazılacaktır.)

JÜRİ:

Danışman :……. …….

Üye :……. …….

Üye :……. …….

ONAY:

Bu tezin kabulü Enstitü Kurulunun ……...… tarih ve .....sayılı kararı ile onaylanmıştır.

……. /……../ ……

Enstitü Müdürü

TEŞEKKÜR

Bu tezi yazmamda yardımcı olan ve benden desteğini esirgemeyen annem Hacer HAKKOMAZ, rahmetli babam Halil HAKKOMAZ'a ve hayat arkadaşım Özge KALA HAKKOMAZ'a teşekkür ederim. (Kayseri, 2022).

Hüseyin Hakkomaz

Kayseri, Şubat 2022

GIDALARDAKİ KARBONHİDRAT MİKTARININ TESPİTİ İÇİN BÖLGE TABANLI EVRİŞİMSEL SİNİR AĞI TEMELLİ GÖRÜNTÜ İŞLEME MODELİ

Hüseyin HAKKOMAZ

Nuh Naci Yazgan Üniversitesi, Fen Bilimleri Enstitüsü

Yüksek Lisans Tezi, Mart 2022

Danışman: Doç. Dr. Zeki ORALHAN

ÖZET

Tip 1 Diabetes Mellitus (T1DM) tüm dünyadan her sene artış gösteren otoimmun bir hastalıktır. Beslenme yöntemlerini etkileyen ve karbonhidrat sayımının önemli olduğu bu hastalık bireylerin kendilerini sürekli takip etmesini ve beslenmelerini kayıt altında tutmasını gerektirmektedir. Bilgisayarlı görü yöntemleri ile kişilerin yemeklerinin içeriklerini ve besin değerlerini kolaylıkla hesaplayabilmesi son yıllarda yapılan çalışmalarla imkan kazanmıştır. Gerek 2 boyutlu görseller ile gerekse 3 boyutlu görseller ile yemeğin hacminin hesaplanması üzerine çalışmalar yapılmış ve yapılmaktadır. Yapay sinir ağları üzerindeki gelişmelerin son yıllarda hızlanması ile Yemek tanıma ve hacim tahmini üzerinde yapılan çalışmalar yüksek başarılar elde etmeye başlamıştır. Bu çalışmada yemeklerin çaplarının girdi alarak içerdiği karbonhidrat miktrarını tahmin ediyoruz. Sistemimiz yüzde 1 ile yüzde 15 arasında değişen hata paylarıyla ortalama yüzde 7 civarı hata ile taminde bulunmuştur. Hata oranlarının değişimi yemeğin servis biçimiyle alakalı olmakla birlikte tek görsel ile bu kadar yüksek başarı oranında tahminde bulunması da sistemimizin başarılı olduğunu göstermektedir. Sistem farklı açılardan elde edilen görseller ile 3 boyutlu hacim tahmini için geliştirilmeye elverişli olup bu haliyle daha çok sınıf ve eğitim görseli ile geliştirilerek kullanılmaya hazırdır.

Anahtar Kelimeler: makine öğrenmesi, yapay sinir ağları, bilgisayarlı görü, biyomedikal

REGION-BASED CONVOLUTİONAL NETWORK-BASED IMAGE PROCESSING MODEL FOR DETERMINING THE AMOUNT OF CARBOHYDRATES IN FOOD

Hüseyin HAKKOMAZ

Nuh Naci Yazgan University, Graduate School of Sciences

Master of Science(M.Sc.) Thesis, March 2022

Supervisor: Assoc. Prof. Zeki ORALHAN

ABSTRACT

Type 1 Diabetes Mellitus (T1DM) is an autoimmune disease that increases the number of patients every year around the world. This disease, which affects nutritional methods and carbohydrate counting is important, requires individuals to constantly monitor themselves and keep a record of their nutrition. The ability of people to easily calculate the ingredients and nutritional values of their meals with computer vision methods has been made possible by studies conducted in recent years. Studies on the calculation of the volume of the food with both 2-dimensional images and 3-dimensional images have been done and are being done. With the acceleration of developments on artificial neural networks in recent years, studies on food recognition and volume estimation have started to achieve high success. In this study, we estimate the amount of carbohydrate in the meals with the volume information we have obtained through calculations that we have made by inputting the diameters of the plates after training with the Detectron2 system. Our system was found to be accurate with an average error of around 7 percent, with an error margin ranging from 1 percent to 15 percent. Although the change in error rates is related to the way the food is served, the estimation of such a high success rate with a single image shows that our system is successful. The system is suitable to be developed for 3D volume estimation with visuals obtained from different angles, and is ready to be developed and used with more classes and training images.

Keywords: machine learning, artificial neural network, computer vision, biomedical

İÇİNDEKİLER

TABLO LİSTESİ

ŞEKİL LİSTESİ

SİMGELER ve KISALTMALAR

DM: **Diabetes Mellitus**

T1DM: Tip 1 Diabetes Mellitus

CHO: Karbonhidrat

YSA: Yapay Sinir Ağları

GPU: Grafik İşlem Birimi

CPU: Merkezi İşlem Birimi

AI: Yapay Zeka

RELU: Rectified Lineer Unit

RGB: Red-Green-Blue

RGB-D: Red-Green-Blue-Distance

SWM: Destek Vektör Makinesi

FAST R-CNN: Hızlı Bölgeye Dayalı Evrişimli Sinir Ağları

R-CNN: Bölgeye Dayalı Evrişimli Sinir Ağları

ROI: İlgi Bölgesi (Region Of İnterest)

RPN: Region Proposal Network

FPN: Feature Pyramid Network

API: Uygulama Programlama Arayüzü

GUI: Grafiksel Kullanıcı Arayüzü

IRF: Dürtü Yanıt İşlemi

1. BÖLÜM

GİRİŞ

Günümüz dünyasında gelişen teknoloji ile insan hayatı ve hastaların yaşam kalitesi artmaktadır. Yapay zeka yaşamın her alanına hızla giriş yapan ve neredeyse her yerde en ufak bir örneği ile karşımıza çıkan bir teknolojik ürün olmuştur. Bilgisayarlar insanların yapacağı işlemlerin yerini ilk çıktığı günden beri almaya başlamıştı. Ancak uzman görüşü gerektiren konularda bilgisayarlardan bir beklentimiz olmuyordu. Bilgisayarları işlem yapan büyük hesap makineleri olmaktan öteye götüren teknolojik atılım olan bilgisayarlı görü ve makine öğrenmesi bilgisayarların hayatımızdaki yerini apayrı bir noktaya taşımıştır. İnsanların hesaplamakta zorlandığı matematiksel işlemler, korelasyonlar ya da istatistiksel bilgilerden çıkarılacak ilişkiler gibi konularda yapay zeka çok iyi iş çıkarmaktadır. Bilgisayarlı görü gerek askeri gerek sivil konularda gösterdiği kolaylıklar ile geliştirilen algoritmalar sayesinde kameraların kullanım şekillerini değiştirdi demek yanlış olmaz. Makine öğrenmesi ile birçok işin takibi, yorumlanması ve ayırt edilmesi gibi konularda geliştirilen algoritmalar sayesinde çeşitli alanlarda hem ekonomik yönden hem de verimlilik açısından fayda sağlamıştır.

Makine öğrenimi yöntemlerinin gelişmesi ile birlikte derin öğrenme kavramıda gelişmiştir. Derin öğrenmenin zaman içinde faydalı gelişmeler ile birlikte bankacılık, savunma sanayi, tıp ve mühendislik alanlarında büyük gelişmeler sağlayacak uygulamalara altyapı oluşturmuştur.

Derin öğrenme algoritmaları sayesinde görüntülerden nesnenin tanımını yapmaya ve şeklini ayırt etme yöntemi çok farklı alanlarda kullanılmaktadır. Sinema sektöründe oyuncunun yüzünün yerine başka oyuncunun yüzünü yerleştirebildiğimiz deep fake algoritmasından, kanserli hücrelerin tespitini yapabildiğimiz tıbbı uygulamalara kadar çok geniş bir skalada kullanılan derin öğrenme algoritmaları bulunmaktadır.

Akıllı telefonların gelişimi ile birlikte işlerimizin çoğunu mobil uygulamalar yardımı ile gerçekleştiriyoruz. Gelişen akıllı telefon işlemcileri ve grafik işlem birimleri sayesinde neredeyse bilgisayarlar kadar işlem yapan bir cihazlar halini alan akıllı telefonlar

içlerinde yapay zeka yazılımı çalıştırmaya uygun hale gelmiştir. Hal böyle olunca gelişen telefon kameraları ile çalıştırılan ve yapay zeka kullanan uygulamaların sayısı gün geçtikçe artmaktadır.

1.1 Problemin tanımı

Günümüzün en önemli sağlık sorunlarından biri olan Diabetes Mellitus (DM), Uluslararası Diyabet Federasyonu'na göre acil eylem gerektiren 21. yüzyılın küresel bir salgınıdır [1,2]. Tip 1 diyabet (T1DM), insülin eksikliği ve ardından hiperglisemi gelişen kronik ve otoimmün bir hastalıktır. T1DM'de komplikasyonları önlemek ve metabolik kontrolü sağlamak için tıbbi beslenme tedavisi çok önemlidir. Ek olarak, uygun tıbbi beslenme tedavisi yoluyla, T1DM'li çocuklar sağlıklı bir şekilde büyüyüp gelişebilir ve yeterli ve dengeli beslenme alışkanlıkları geliştirebilir [3]. Küresel T1DM insidansı her yıl %3 oranında artmaktadır [4]. T1DM tedavisi, diyabetin kendi kendine yönetiminin önemli bir bileşeni olan iyi kan şekeri kontrolünü sürdürmeye dayanır. Karbonhidrat tüketimi (CHO) yemek sonrası kan şekerinin ana belirleyicisidir Güncel yönergeler, öğünlerde insülin hesaplanırken öğünlerdeki CHO miktarının dikkate alınması gerektiğini önermektedir [5]. CHO sayımı bir yemek planlama yöntemidir ve odak noktası, yemekten sonra kan şekeri tepkisini esas olarak etkileyen makro besinlerin içeriğini tahmin etmektir [6]. Akıllı telefon teknolojisinin ilerlemesi, kullanımı ve benimsenmesi ve tüm mobil sağlık ekosisteminin sürekli büyümesiyle, gelişmiş diyabet kontrolü için fırsatlar ortaya çıktı. Bununla birlikte, mevcut uygulamaların çoğu genellikle, kullanıcının yiyeceğin adını girmesi gereken ve bazı durumlarda porsiyon boyutunu da belirlemesi gereken işlemlere dayanır.

1.2 Tezin amacı

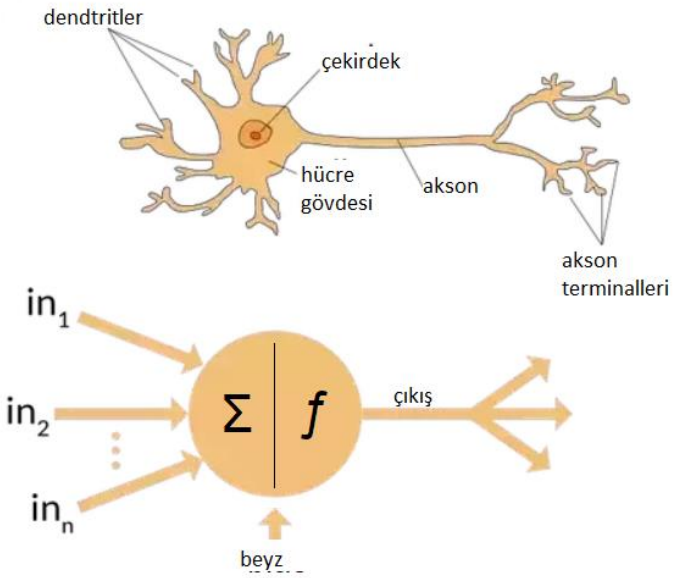
Bu kapsamda çalışmamızda telefon kamerası ile elde edilen yemek görüntüleri analiz edilecek. Elde edilen sonuç ile referans obje kullanılarak yemeğin ölçülerini hesaplanacak ve hacim bilgileri elde edilerek yemeğin karbonhidrat içeriği tahmin edilecek. Daha önceki benzer çalışmalar ile karşılaştırılacak ve avantaj ve dezavantajları belirlenecek. Ve sonuçlar incelenecektir.

Çalışmamız beş kısımdan oluşmaktadır. Birinci kısım giriş bölümü problemin tanımı ve tezin amacı anlatılacaktır. ikinci kısımda derin öğrenme detaylı olarak incelenecektir. Derin öğrenme kavramı, tarihi ve algoritmalarından bahsedilecektir. Derin öğrenmenin kullanım alanlarına değinilecektir. üçüncü bölümde araştırma gereç ve yöntemleri detaylı olarak anlatılacaktır. Kullanılan algoritmalar yardımcı yazılımlardan bahsedilecektir. dördüncü bölümde araştırma bulguları anlatılacak ve çalışmadan elde edilen veriler paylaşılacaktır. Son bölümde ise sonuçlar değerlendirilecek ve öneriler sunulacaktır.

2. BÖLÜM

DERİN ÖĞRENME

Zeka gibi, öğrenme de çok çeşitli süreçleri kapsar, bu nedenle kesin olarak tanımlamak zordur. Kelime tanımları, "bilgi, rehberlik veya deneyim edinmeyi öğrenme, bilgi veya becerileri anlama" ve "davranış değişikliği deneyimi" gibi ifadeleri içerir. Zoologlar ve psikologlar, hayvanların ve insanların öğrenmesini inceler. Hayvan öğrenimi ile makine öğrenimi arasında pek çok benzerlik vardır. Elbette, makine öğrenimindeki birçok teknik, psikologların hesaplamalı modeller aracılığıyla hayvan ve insan öğrenme teorilerini daha hassas hale getirme çabalarından kaynaklanmaktadır. Makine öğreniminde araştırmacılar tarafından keşfedilen kavramlar ve teknikler, biyolojik öğrenmenin belirli yönlerine ışık tutma potansiyeline sahip gibi görünüyor. Makine ile ilgili olarak, makine yapısını, programını veya verilerini değiştirdiği sürece, öğrenme yönteminin beklenen gelecekteki performansı iyileştirebileceği söylenebilir [1]. Şekil 2.1 de yapay sinir hücresi ile gerçek sinir hücresi çizimi gösterilmektedir.



Şekil 2. 1 Yapay sinir ağı ve gerçek sinir ağı

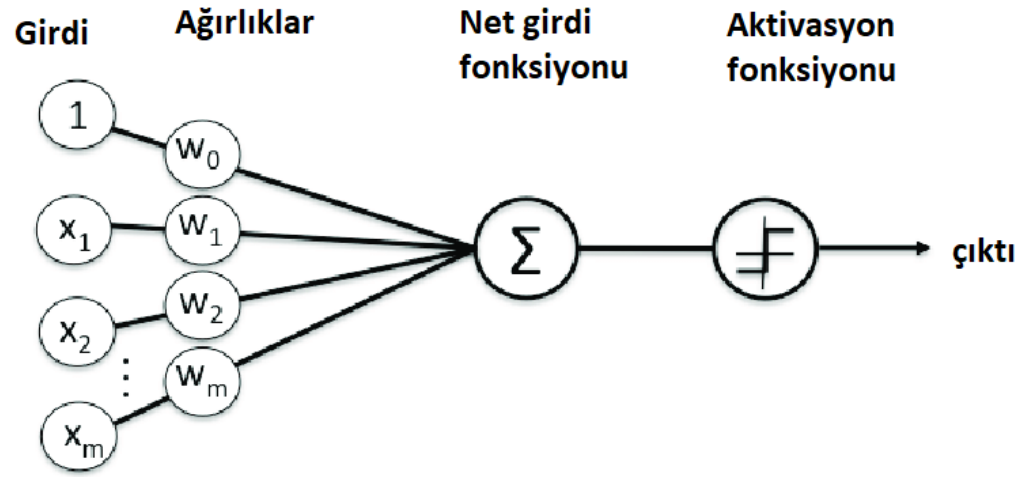
Makine öğrenimi, verilerden öğrenebilen ve programlama yapmadan zaman içinde doğruluğu artıran uygulamalar oluşturmaya odaklanan bir yapay zeka dalıdır. Veri biliminde, bir algoritma bir dizi istatistiksel işlem adımıdır. Makine öğreniminde, algoritmalar, yeni verilere dayalı kararlar ve tahminler yapmak için büyük miktarda verideki kalıpları ve özellikleri keşfetmek üzere eğitilir. Algoritma ne kadar iyi ve ne kadar çok veri işlerse, kararlar ve tahminler o kadar doğru olur. Derin öğrenme makine öğreniminin alt dalıdır. Tüm derin öğrenme işlemleri makine öğrenimidir ancak tüm makine öğrenimi işlemleri derin öğrenme değildir. Derin öğrenme algoritmaları insan beyninin öğrenme yöntemini öğrenmek için tasarlanmış yapay sinir ağlarıdır. Derin öğrenme modelleri sonuçları sürekli ayarlamak için birden çok katmandan büyük miktarda veri hesaplaması ve ardışık katmanlara ağırlık uygulaması yapar [2]Şekil 2.2'de derin öğrenmenin yapay zekanın bir alt dalı olduğu gösterilmektedir.

Şekil 2. 2 Derin öğrenme yapay zeka ilişkisi

2.1 Derin Öğrenme Tarihi

Derin öğrenmenin tarihi, Walter Pitts ve Warren McCulloch'un insan beyninin sinir ağına dayalı bir bilgisayar modeli oluşturduğu 1943 yılına kadar gider. Düşünme

sürecini taklit etmek için "eşik mantığı" adı verilen algoritmalar ve matematiksel yöntemler kombinasyonu kullandılar [3].

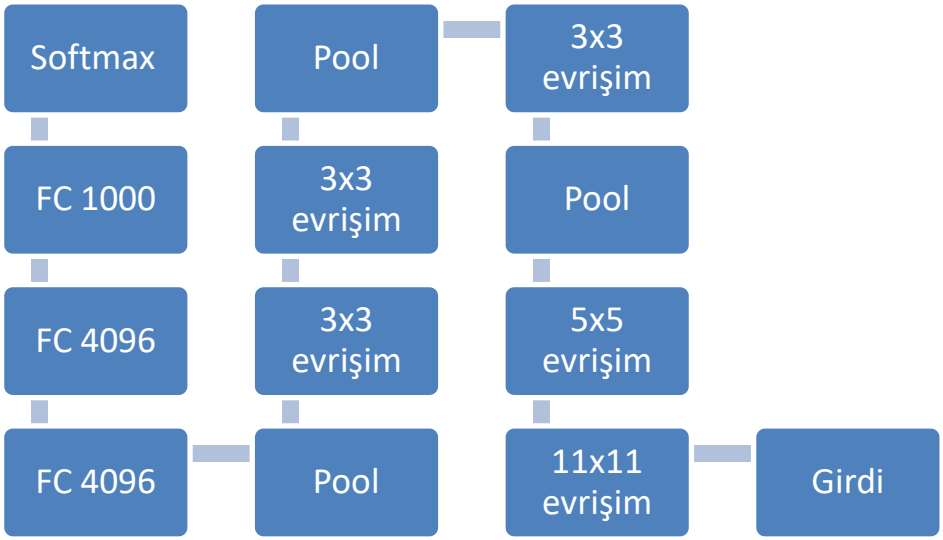


Şekil 2. 3 Perceptron modeli

Perceptron modeli Şekil 2.3'de gösterilmekte olan , 1958'de Frank Rosenblatt tarafından geliştirilmiştir [4]. Henry J. Kelley, 1960 yılında Back Propagation "Geri yayılım" algoritmasının temellerini geliştirdi [5]. 1962'de Stuart Dreyfus tarafından sadece zincir kuralına dayalı daha basit bir versiyon geliştirildi [6]. Geriye yayılma kavramı (eğitim amacıyla hataların geriye doğru yayılması) 1960'ların başlarında mevcutken verimsizdi ve 1985'e kadar işe yaramayacaktı. Derin öğrenme algoritmalarını geliştirmeye yönelik ilk çalışma, 1965 yılında Alexey Grigoryevich Ivakhnenko (geliştirilmiş veri işleme gruplama yöntemi) ve Valentin Grigor′evich Lapa (sibernetik ve tahmin teknolojisi yazarı) tarafından geliştirilmiştir. Polinom aktivasyon fonksiyonlarına sahip modelleri kullanırlar ve ardından istatistiksel analiz gerçekleştirirler. Ardından, her katmanda istatistiksel olarak seçilen en iyi özellikleri bir sonraki katmana geçirdiler [7]. Kunihiko Fukushima ilk "evrişimli sinir ağını" üretti. Fukushima, çoklu havuzlama ve evrişimli katmanlara sahip bir sinir ağı tasarladı. 1979'da, hiyerarşik çok katmanlı bir tasarım kullanan Neocognitron adlı yapay bir sinir ağı geliştirdi. Bu tasarım, bilgisayarın görsel modu öğrenmesini sağlıyordu. Bu ağlar modern versiyonlara benziyor, ancak takviye stratejilerinin tekrarlanan döngüsel aktivasyon eğitiminden sonra, ağlar zamanla geliştirildi. Ek olarak, Fukushima'nın tasarımı, belirli bağlantıların ağırlığını artırarak

önemli işlevlerin manuel olarak ayarlanmasına izin veriyor [8]. Neocognitron kavramlarının çoğu kullanılmaya devam ediyor. Yukarıdan aşağıya bağlantıların ve yeni öğrenme yöntemlerinin kullanılması, çeşitli sinir ağlarının gerçekleştirilmesine izin verdi. Aynı anda birden fazla model sunulduğunda, Seçici Dikkat Modeli "Selective Attention Model", dikkatini birinden diğerine kaydırarak bireysel kalıpları ayırabilir ve tanıyabilir. Modern bir Neocognitron, yalnızca eksik bilgi içeren desenleri tanımlayamaz, aynı zamanda eksik bilgileri ekleyerek görüntüyü tamamlayabilir. Bu, "çıkarım" olarak tanımlanabilir. Aynı yıllar Seppo Linnainmaa'nın geri yayılım için FORTRAN kodunu içeren yüksek lisans tezini yazdığı zamandır. Ne yazık ki, bu kavram 1985 yılına kadar sinir ağlarına uygulanmadı [9]. O sırada Rumelhart, Williams ve Hinton, sinir ağlarında geri yayılmanın "ilginç" dağıtım temsilleri sağlayabileceğini kanıtladılar [10]. Felsefi bir bakış açısıyla, bu keşif, bilişsel psikolojideki bir sorunu, yani insan anlayışının sembolik mantığa (hesaplama) mı yoksa dağıtılmış temsile mi (bağlantısallık) dayandığını ortaya çıkarır [11]. 1989'da Yann LeCun, Bell Laboratuvarlarında geri yayılımın ilk pratik tanıtımını gerçekleştirdi. El yazısıyla yazılmış sayıları okumak için evrişimli sinir ağlarını ve geri yayılımı birleştirdi [12]. 1995 yılında, Dana Cortes ve Vladimir Vapnik bir destek vektör makinesi geliştirdi (benzer verileri haritalamak ve tanımlamak için bir sistem) [13]. Sepp Hochreiter ve Juergen Schmidhuber, 1997'de tekrarlayan sinir ağları için LSTM'yi (Uzun Kısa Süreli Bellek) geliştirdi [14]. Derin öğrenme için bir sonraki evrimsel gelişim 1999 yılında grafik işlem birimlerinin çıkışı oldu. Bu GPU'lar ile birlikte resimleri işleyebilme hızları 10 yıl içinde 1000 kat arttı. Bu dönemde sinir ağları, destek vektör makineleri ile rekabet etmeye başladı. Sinir ağları, destek vektör makinelerinden daha yavaş olsa da, aynı verileri kullanan sinir ağları daha iyi sonuçlar sağlayabiliyordu. Sinir ağları, daha fazla eğitim verisi eklendikçe gelişme avantajına da sahiptir. 2000'li yıllarda kaybolan gradyan sorunu ortaya çıktı. Alt katmanlarda oluşan özelliklerin üst katmanlara aktarılamadığı görüldü. Bu sorun tüm sinir ağları için geçerli değil sadece gradyan tabanlı öğrenme yöntemlerine sahip olan ağlar için geçerli bir sorundu. Sorunun kaynağının bazı aktivasyon fonksiyonları olduğu keşfedildi.Bu sorunu çözmek için kullanılan iki çözüm, katman katman ön eğitim ve uzun kısa süreli belleğin geliştirilmesiydi [11]. 2009 yılında bir yapay zeka profesörü olan Fei-Fei Li, ImageNet'i çıkarttı. 14 milyondan fazla etiketli görüntü sunan bu veritabanını ücretsiz bir şekilde

ortaya çıkardı [15]. 2011 yılına gelindiğinde GPU'ların hızı önemli ölçüde arttı. Böylelikle katman katman ön eğitim olmadan erişimli sinir ağlarını eğitmek mümkün oldu. Buna örnek olarak 2011 ve 2012 de birçok uluslararası yarışmayı kazanan model olarak AlexNet'i verebiliriz [16].



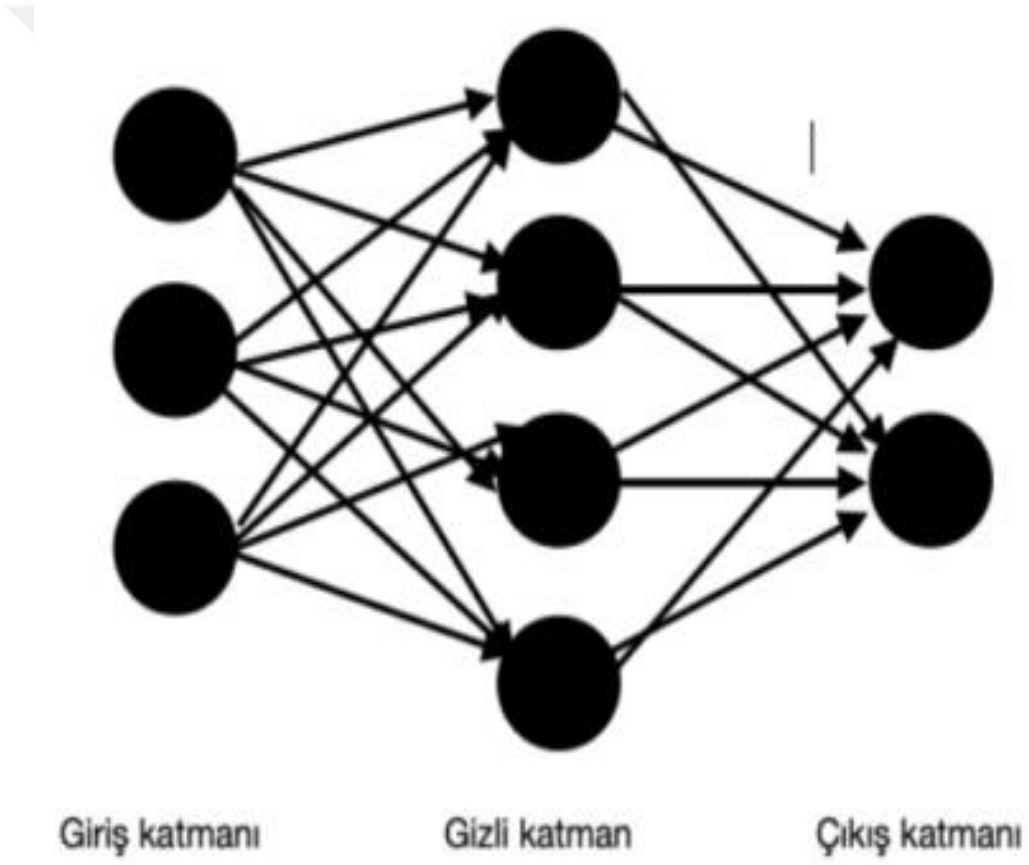
Şekil 2. 4 AlexNet mimarisi

2012 yılında Google Brain Cat Experiment isimli projenin sonuçlarını yayınladı.bu projenin hedefi denetimsiz öğrenmeyi deneyimlemekti. Normalde derin öğrenme denetimli öğrenme ile eğitilir. Örneğin ImageNet'ten elde edilen etiketli görseller ile eğitilmesi gibi. Bu projede ise 1000 bilgisayara dağılmış sinir ağında Youtube üzerinde alınan 10 milyon işaretlenmemiş görsel ile eğitim yapıldı. Eğitimin sonunda en üst katmandaki nöronların kedinin görseline güçlü bir tepki verdiği görüldü. Projenin kurucusu Andrew Ng, "Ayrıca insan yüzlerine çok güçlü yanıt veren bir nöron bulduk." dedi.

2.2 Yapay Sinir Ağları

Yapay Sinir Ağı (YSA), insan beyninin bilgiyi analiz etme ve işleme şeklini simüle etmek için tasarlanmış bir bilgi işleme sisteminin parçasıdır. Yapay zekanın (AI) temelidir ve insanlar veya istatistiksel standartlar tarafından kanıtlanamayan veya çözülmesi zor olan sorunları çözer. Yapay sinir ağlarının kendi kendine öğrenme işlevi vardır ve daha fazla veri mevcut olduğunda daha iyi sonuçlar üretebilirler. Biyolojik

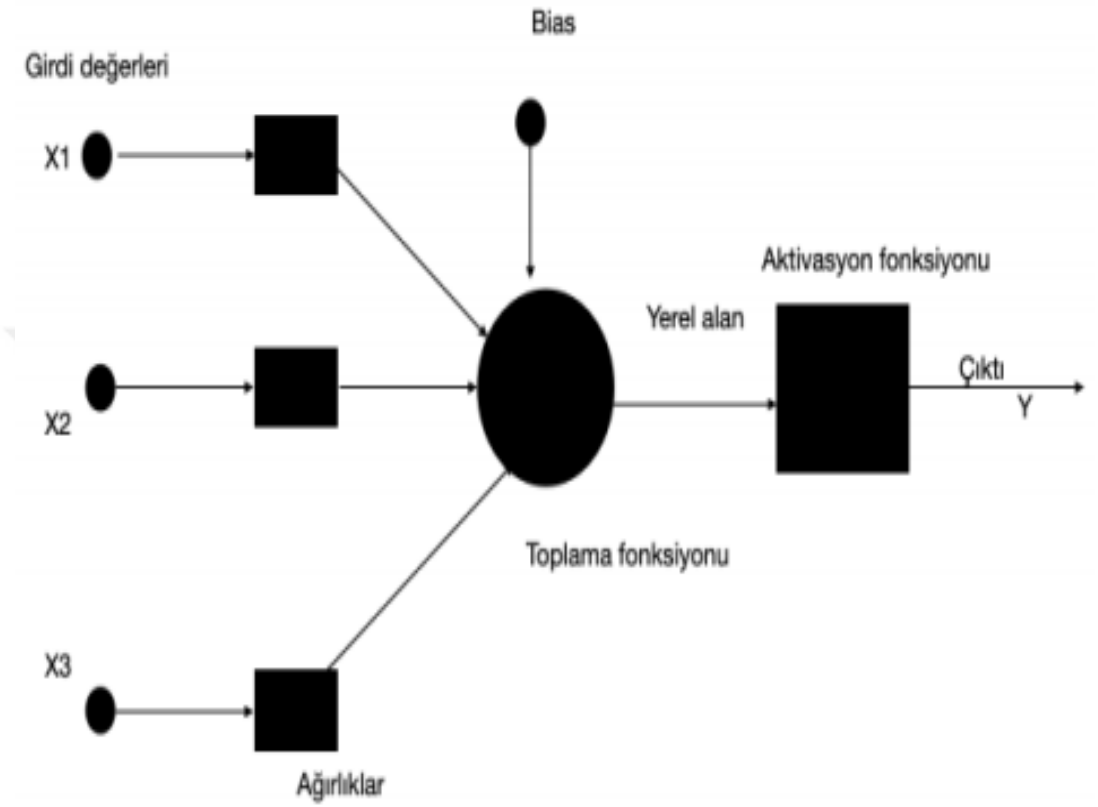
organizmalarda öğrenme işlemi nöronları birbirine bağlayan sinaptik bağlantılar sayesinde gerçekleşir. Yani insanlar doğduğu andan itibaren öğrenmeye başlarlar ve bu sinaptik bağlantılar gelişir zaman içerisinde yenileri oluşarak öğrenme devam eder. Bu durum yapay sinir ağları için de geçerlidir [17]. Yapay sinir ağları ağırlık verilmiş şekilde birbirine bağlı birçok nörondan oluşan matematiksel bir sistemdir. Transfer fonksiyonu olarak adlandırılan işlem birimi nöronlardan gelen sinyalleri birleştirir, dönüştürür ve ardından sayısal bir çıktı üretir. Genel anlamda bu işlem birimleri gerçek nöronlara karşılık gelirler ve sistem içerisinde birbirlerine bağlıdırlar. Bu yapı da yapay sinir ağları adını almaktadır [17].



Şekil 2. 5Yapay sinir ağı

Şekil 2.1 de görülen basit bir şekilde yapay sinir ağı temel olarak üç katmandan oluşmaktadır. Giriş katmanı, gizli katman ve çıkış katmanı olarak. Birbirlerine bağlı hücre

bağlantıları sinir sistemindeki sinapsları temsil etmektedir. Giriş katmanı sinir ağlarına veri girişi için kullanılır bu katmanda hesaplama yapılmamaktadır.



Şekil 2. 6 Yapay sinir ağı hücresi

Şekil 2.2'de bir yapay sinir hücresi gösterilmiştir. Bu sinir hücresinde n adet veri girişi olur. Ağırlıklar ile veriler çarpılır ve toplama fonksiyonunda toplanır üzerine bias değeri eklenir. Böylelikle net girdi elde edilmiş olur. Net girdi aktivasyon fonksiyonundan geçer ve çıktı elde edilir.

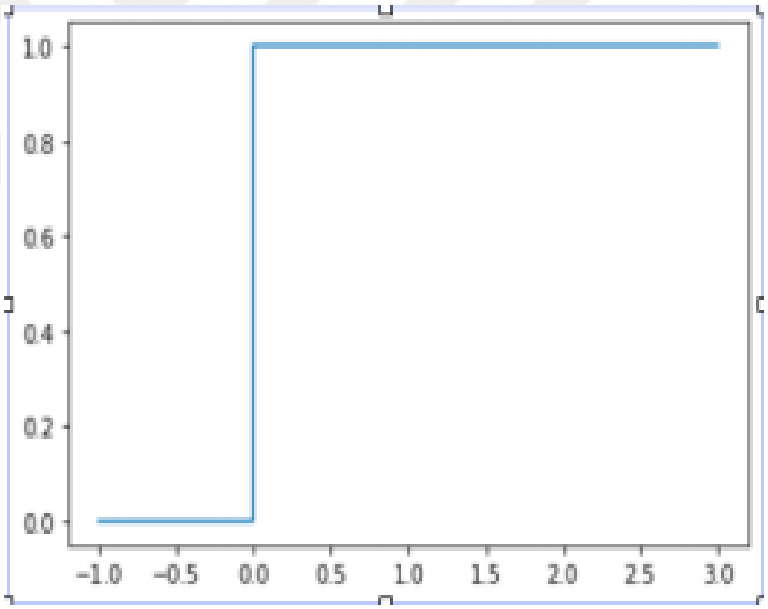
2.2.1 Aktivasyon fonksiyonu

Aktivasyon fonksiyonunda, yapay sinir hücreleri giriş verilerini işler ve karşılık gelen net çıktı sonucunu elde eder. Bu işlev genellikle doğrusal değildir. İşlevi doğru seçmek önemlidir. Çünkü sonuç performansı etkileyecektir. İşlev, tek kutuplu veya iki kutuplu

olabilir [18]. Aktivasyon fonksiyonu, sinir ağı modelinin çıktısını belirleyen matematiksel bir denklemdir. Aktivasyon fonksiyonunun, sinir ağının yakınsama yeteneği ve yakınsama hızı üzerinde de önemli bir etkisi vardır veya bazı durumlarda, aktivasyon fonksiyonu, sinir ağının ilk etapta yakınsamasını engelleyebilir. Aktivasyon işlevi ayrıca 1 ila -1 veya 0 ila 1 aralığındaki herhangi bir girişin çıkışını düzenlemeye yardımcı olur. Aktivasyon işlevi verimli olmalı ve hesaplama süresi kısaltılmalıdır, çünkü sinir ağları bazen milyonlarca veri noktası üzerinde eğitilir.

2.2.1.1 Basamak Fonksiyonu

İkili değer alan bu fonksiyon genellikle ikili sınıflayıcı olarak kullanılır. Bu yüzden genellikle çıkış katmanında kullanılırlar. Şekil 2.3 de basmak fonksiyonu gösterilmektedir.

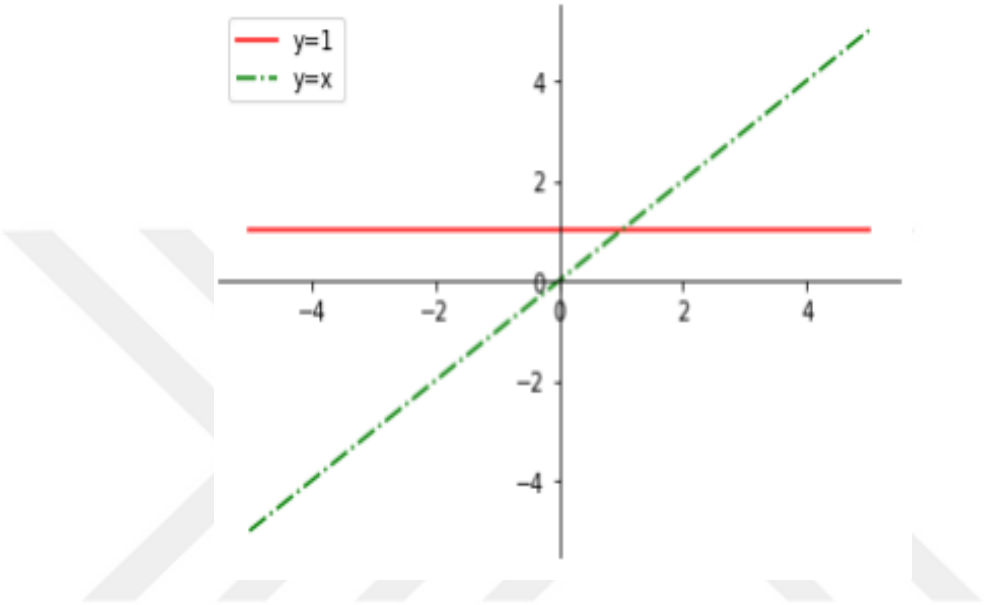


Şekil 2. 7 Basamak fonksiyonu

2.2.1.2 Doğrusal Fonksiyon

Çok sayıda aktivasyon değeri üretir ve basamak fonksiyonundaki gibi ikili değerler değildir. Birkaç fonksiyonu birbirine bağlamaya izin verir. Bu fonksiyonun problemi ise türevinin sabit olması. Geriye yayılım algoritmasında öğrenme işlemini

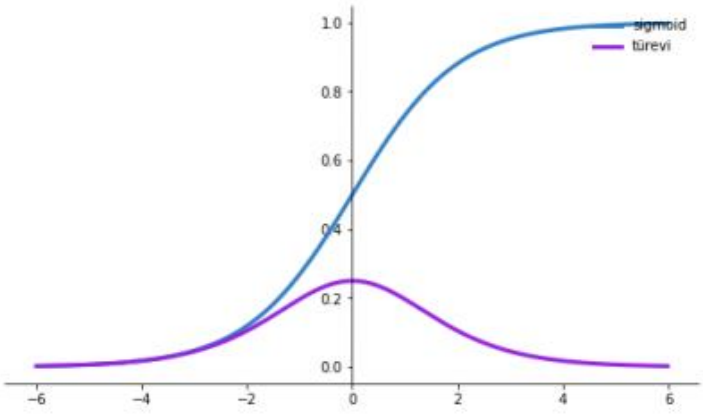
nöronlar için gerçekleştirmiş oluyoruz ve bu algoritma türev işlemi gerçekleştiriyor. Türev hep sabit bir değer çıkıyorsa öğrenme işlemini gerçekleştiremiyoruz. Ayrıca tüm katmanlarda doğrusal fonksiyon kullanıldığında giriş ileçıkış arasında doğrusal bir bağlantı elde ediyoruz. Durum böyle olunca ara katmanların amacı bir işe yaramamış oluyor. Şekil 2.4'te doğrusal fonksiyonun grafiğini görebiliriz.



Şekil 2. 8 Doğrusal fonksiyon

2.2.1.3 Sigmoid fonksiyonu

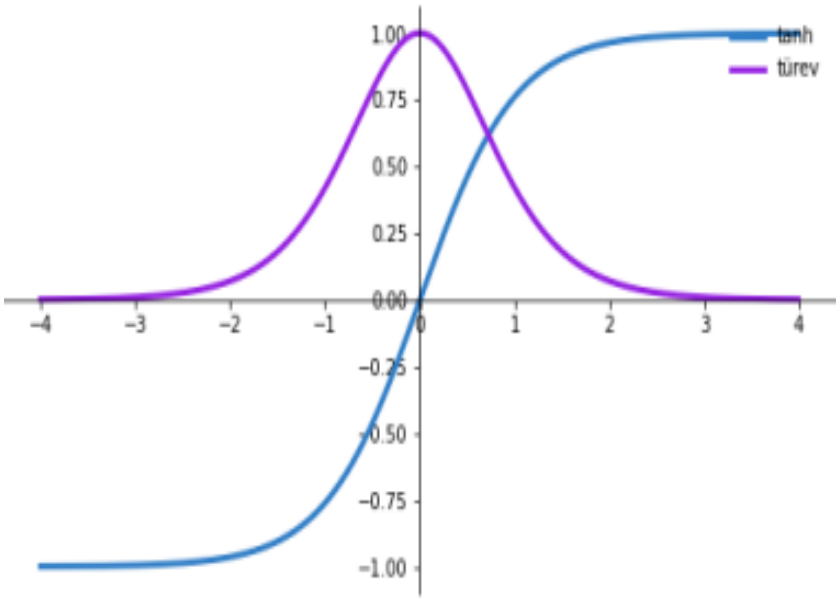
Aktivasyon fonksiyonları arasından en yaygın kullanılanlarından biridir. 0 ile 1 arasında çıktı veriri ara katmanlarda kullanılmasının önünde engel yoktur. Şekil 2.5'e bakarsak grafiğin uç noktalarına doğru y değerleri x değerine karşılık çok az tepki vermektedir. Bu şöyle bir soruna yol açmaktadır. Bu bölgedeki türevler çok küçük olur ve sıfıra yakınsar. Bu duruma gradyan kaybolması denir. Bu konuya giriş kısmında değinmiştik. Böyle olunca öğrenme olayı minimum düzeyde gerçekleşir 0 olursa gerçekleşmez. Sonuçta yapay sinir ağından alabileceğimiz maksimum performansı alamayız.



Şekil 2. 9 Sigmoid fonksiyonu ve türevi

2.2.1.4 Hiperbolik tanjant fonksiyonu

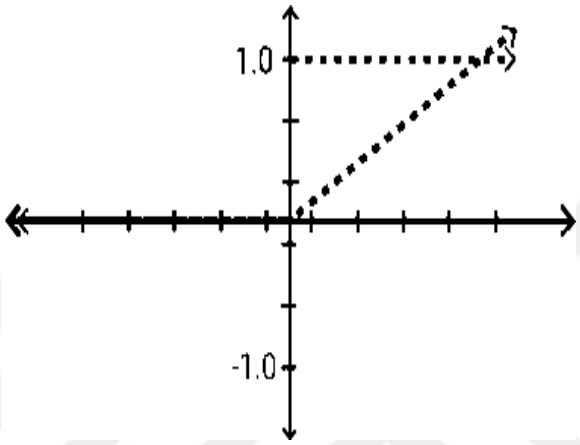
Sigmoid fonksiyonuna benzeyen bu fonksiyonun farklı -1 ile +1 aralığında gerçekleşmesidir. Avantajlı kısmı ise türevinin daha dik olması Şekil 2.6 da gördüğümüz gibi. Hızlı öğrenme ve sınıflama işlemleri için daha geniş aralığa sahip olacağından ötürü daha verimli bir fonksiyondur. Ancak sigmod'te olduğu gibi fonksiyonun uçlarında gradyan kaybolması sorunu devam etmektedir.



Şekil 2. 10 Hiperbolik tanjant fonksiyonu ve türevi

2.2.1.5 ReLU fonksiyonu

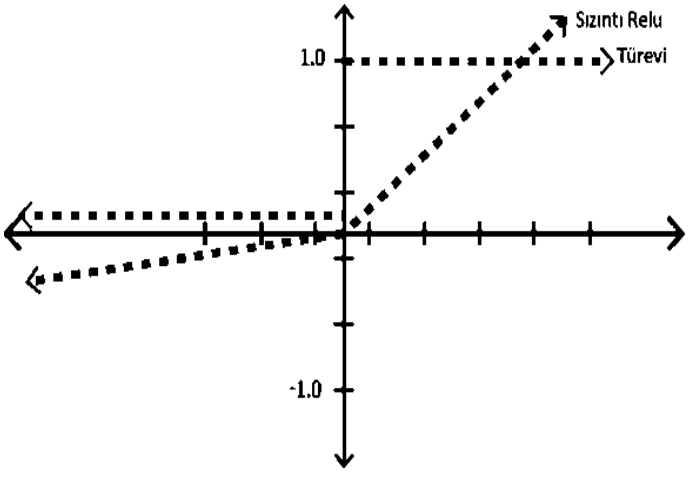
ReLU “Rectified Linear Unit” günümüzde en çok kullanılan aktivasyon fonksiyonlarından biridir. Alınan değer negatifse çıkışı sıfıra eşitlerken, pozitifse değer aynı kalır. Işarete yönelik basit bir fonksiyon olduğu için hızlı çalışmaktadır. Gradyan kaybolması gerçekleşmez. Dezavantajı ise negatif bölgenin sıfır olması nedeni ile türevininde sıfır olması sonucuyla bu bölgede öğrenme gerçekleşmez.



Şekil 2. 11 ReLu fonksiyonu ve türevi

2.2.1.6 Sızıntı (leaky) ReLU fonksiyonu

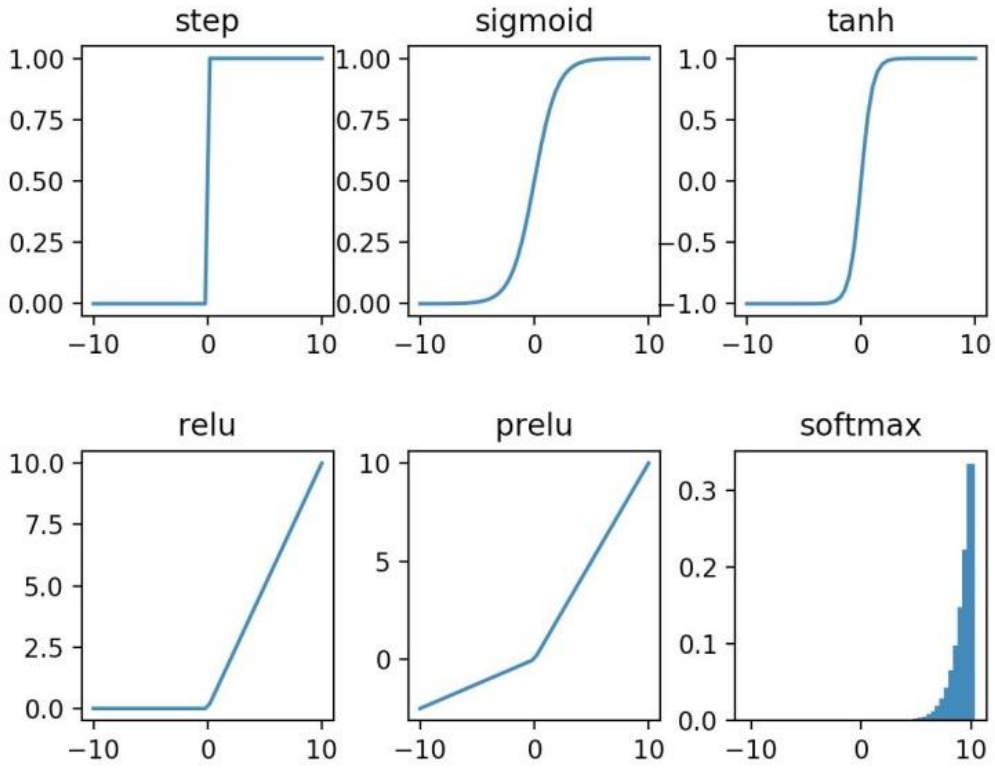
Bu fonksiyonun normal relu fonksyionuna göre farkı negatif kısımdaki sızıntıdır. Bu değer 0,01 olarak atanır. Bu sayede negatif bölgedeki ölü gradyanlardan kurtulma sağlanır ve negatif bölgede öğrenme gerçekleşir.



Şekil 2. 12 Sızıntı ReLu fonksiyonu ve türevi

2.2.1.7 Softmax fonksiyonu

Sigmoid fonksiyonuna bu yapıda benzemektedir. Sınıflandırıcı olarak kullanımı durumunda performansı daha da artmaktadır. Ikiden fazla sınıflama gereken durumlarda sigmoid fonksiyonu gibi çıkış katmanında tercih edilmektedir.



Şekil 2. 13 Aktivasyon fonksiyonları

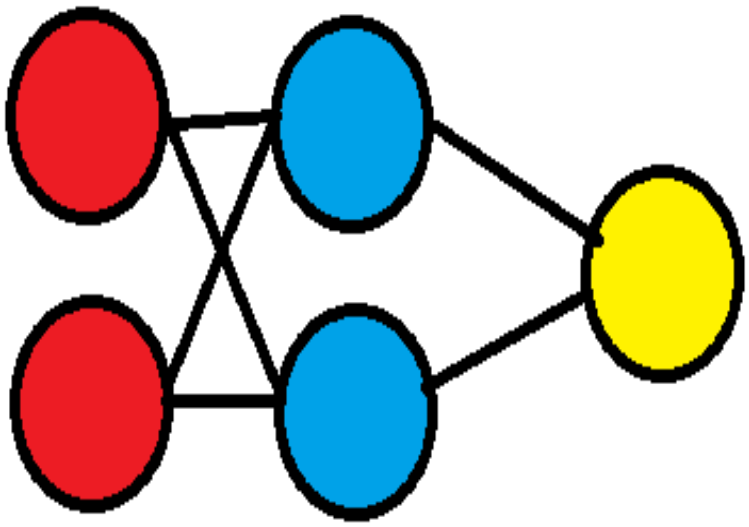
2.2.2 Yapay sinir ağları çeşitleri

Yapay sinir ağları birçok farklı grupta sınıflandırılabilirler. Bunlar ileri beslemeli, radyal temelli ağlar, yinelenen sinir ağları, evrişimli sinir ağları, modüler sinir ağları ve sıradan sıraya modelleri olarak sınıflandırılabilirler.

2.2.2.1 İleri beslemeli sinir ağları

En basit sinir ağlarından biri olan bu tipte veriler, çıkış düğümüne giderken farklı giriş düğümlerinden geçerler [19]. Bu sinir ağında veriler sadece tek yönde ilerler. Bu

sistem giriş düğümü iki adet gizli katman ve çıkış düğümünden oluşur. İleri beslemeli sinir ağları yüz tanıma ve bilgisayarlı görü teknolojilerinde kullanılmaktadır.



Şekil 2. 14 İleri beslemeli yapay sinir ağı

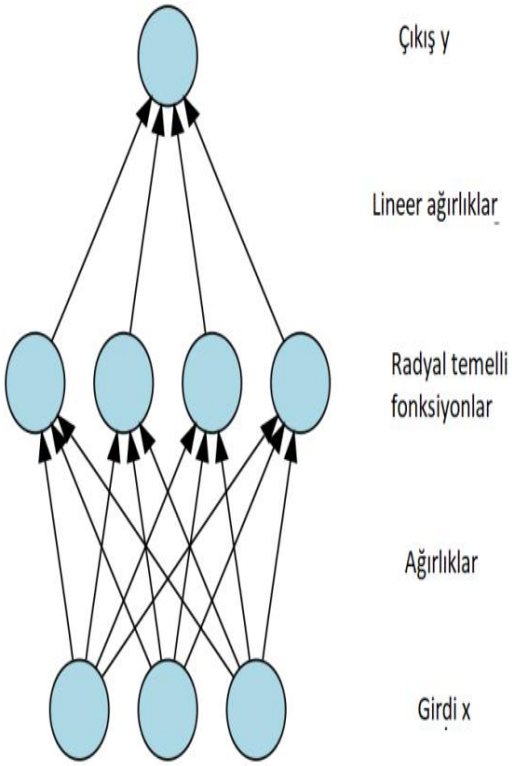
2.2.2.2 Radyal temelli ağlar

Radyal temel fonksiyonu yaklaşımı, köklerini, düzensiz olarak konumlandırılmış veri noktaları göz önüne alındığında, sinir ağlarında öğrenmeye alternatif bir araç olarak kullanımı özellikle çok değişkenli enterpolasyona uygun olan Powell (1987) 'nin çalışmasına dayandırır [20]. Radyal temelli ağlardan kasıt radyal temelli fonksiyonları aktivasyon fonksiyonları olarak kullanan ağlardır. Ağın çıktısı girdilerin ve nöron değerlerinin radyal temel fonksiyonlarının doğrusal kombinasyonlarıdır. Radyal temel fonksiyon ağları, zaman serisi tahmini, fonksiyon tahmini ve sınıflandırma gibi birçok alanda kullanıma sahiptir. Radyal temelli fonsiyonlar yapay sinir ağlarının özel bir türüdür [21]. Hesaplama olarak kolay bir ağ olarak kabul edilmektedir.

2.2.2.3 Yinelenen sinir ağları

Yinelenen sinir ağları bir önceki adımın çıktısını mevcut adıma girdi olarak verdiği bir sinir ağıdır. Geleneksel sinir ağlarındaki girdi ve çıktılar birbirinden bağımsız olsa da bir sonraki durumun tahmin edilmesi gereken durumlarda önceki sonuçlar

gereklidir. Bu sorunu gizli katman yardımıyla çözen yinelenen sinir ağları geliştirilmiştir. Bu sinir ağından veri ileri doğru iletilmektedir. Sistemin elde ettiği tahmin verisi doğru sonuç ile karşılaştırılarak hata oranı elde edilmiş olur



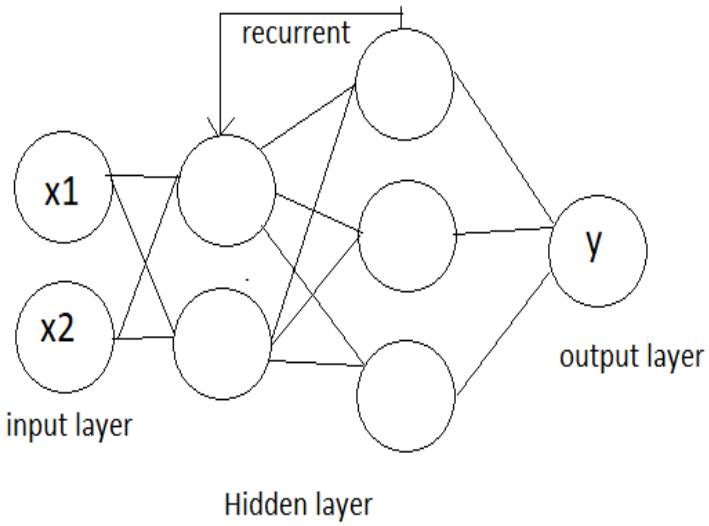
Şekil 2. 15 Radyal temelli ağ yapısı

. Ağda bulunan ağırlıklar hata değerleri yardımı ile yeniden ayarlanarak gerçek sonuçlara daha yakın sonuçlar elde edilmeye başlanmaktadır. Temel amaç hata payını düşürmektir. Yinelenen sinir ağının en büyük farkı o anki veri ile ilgilenmekten ziyade daha önceki veriler ile daha sonra yüklenecek verilerin birbirleri ile olan ilişkiyi sağlamaktır.

2.2.2.4 Evrişimli sinir ağları

Günümüzde derin öğrenme denilince akla ilk olarak evrişimli sinir ağları gelmektedir. Büyük veri ile uğraşılırken milyarlarca değerleir hesaplama için verilerden farklı bilgileri çıkarmak gerektiği düşünüldü. Klasik yapay sinir ağları ile nöronlar arası

katmanlar ve öğrenilen parametreler çok büyük hesaplama zorluklarına neden olmaktadır [22].

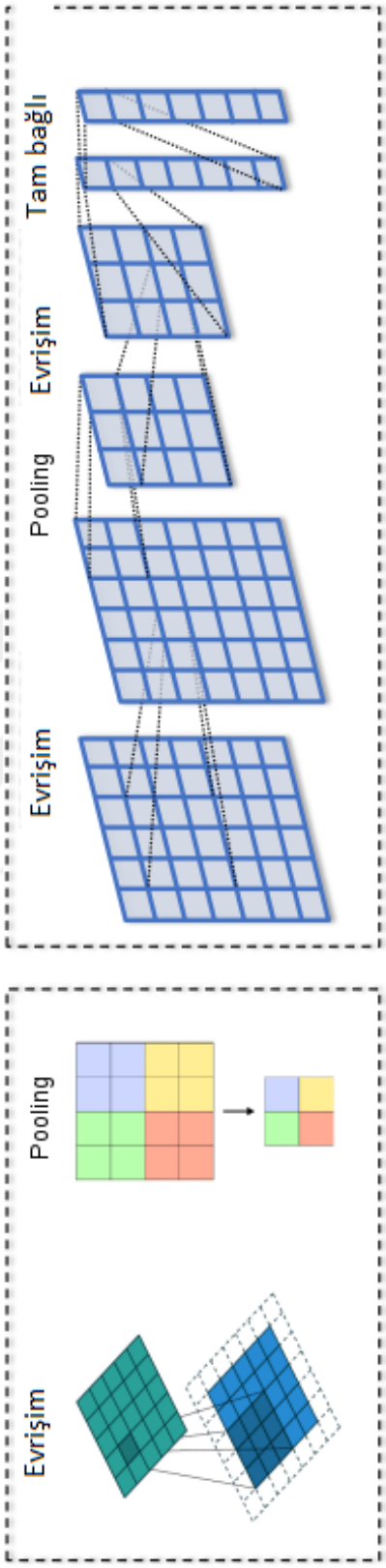


Şekil 2. 16 Yinelenen sinir ağı yapısı

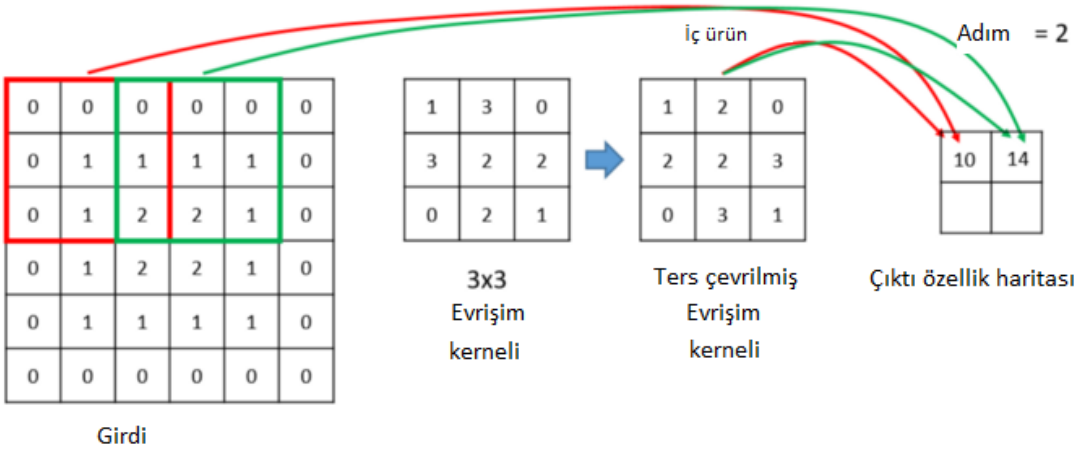
Klasik ağlara göre daha hızlı eğitilebilen ağlar olan evrişimli sinir ağları çeşitli katmanlar ile her katmanın çıktısında belirli değerlere sahip haritalar sunmaktadır. Bu haritalar kullanılan ağın özelliğine göre sınıflandırma, görüntü iyileştirme ve nesne bulma gibi sınıflandırılabilir.

2.2.2.4.1 İki boyutlu evrişim

İki boyutlu veriye uygulanacak fitrenin x ve y eksenine göre ayna görüntüsü elde edilir. Bütün veriler matriste tek tek çarpılır ve tüm verilerin toplam çıkış matrisindeki ilgili eleman olarak belirlenir. Bu yaptığımız işlemi çaprak korelasyon ilişkisi olaraktan isimlendirebiliriz. Giriş verisi tek katmanlı iken basit olarak bu işlem yapılabilir. Farklı formatlarda ve farklı kanal sayılarında da olabilen giriş verilerinde de yöntemimizi kullanabiliriz. Renk kısmının öğrenmede önemli olmadığı durumlarda tek katmanı olarak yapabileceğimiz işlemimizi renkli görüntülerde, kırmızı-yeşil-mavi(RGB) 3 katmandan meydana gelmektedir. Bu durumda evrişim işlemi 3 kanal için yapılırken çıkış işaretinin kanal sayısı ugulanan filte sayısı ile eşit hesaplanması gerekir [22].



Şekil 2. 17 Evrişimli sinir ağı yapısı



Şekil 2. 18 İki boyutlu evrişim işlemi [R.]

2.2.2.4.2 Kenar bulma

Kenar bulma işlemi görüntü özelliklerinden ilk aranan işlemlerden birisidir. Giriş verisini yüksek frekanslı kısımlarını veren bu özniteliği elde etmek için x ve y ekseninde olmak üzere iki filtre kullanılır. Filtreler görüntü içerisinde evrişim işlemi gerçekeleştirir. Elde edilen sonuç, görüntüye ait kenar bilgisini içerir. Genellikle kenar hespalamaları evrişimli sinir ağının ilk katmanlarında hesaplanmaktadır. Farklı filtreler ile açısal kenarlar, ışık geçişleri hesaplanır. Bu işlemler yapılmaktayken giriş ve çıkış boyular arasında farklılıklar meydana gelmektedir [22].

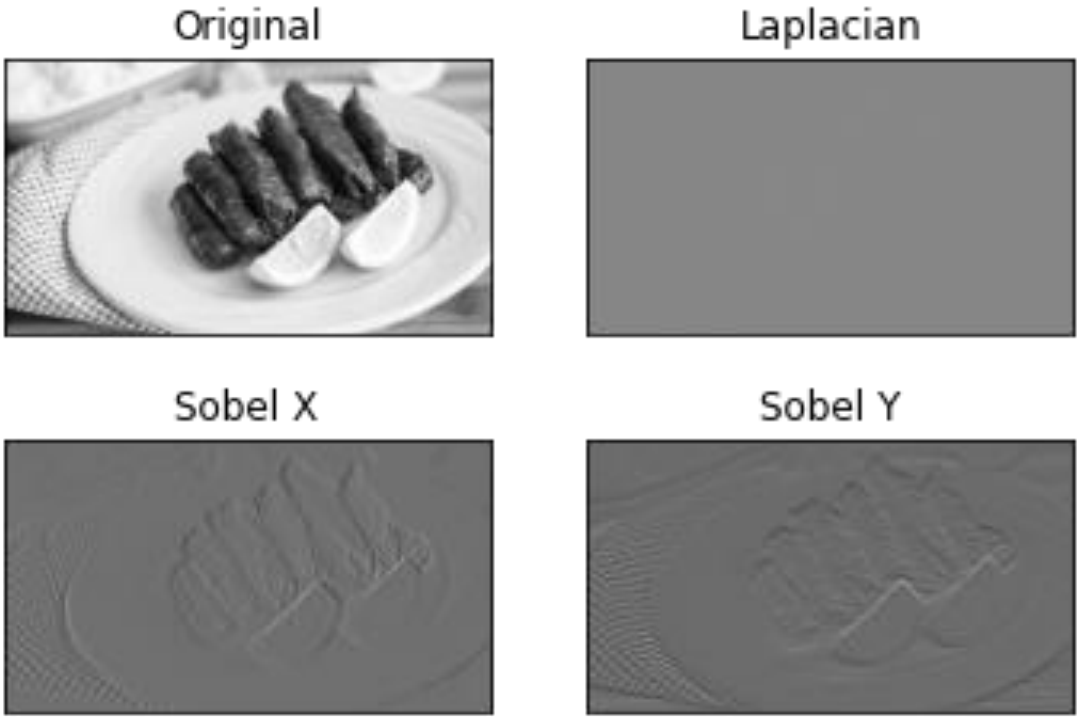
2.2.2.4.3 Piksel ekleme

Evrişim işleminin ardından giriş ve çıkış işaretlerinin arasındaki boyut farklılığını ayarlayabiliriz. Bu işlemi yapmanın yolu giriş matrisine piksel eklemektir. Piksel ekleme işlemine (padding) denir. Giriş matrisi nxn, filtre matrisi (fxf) olduğu sistemde çıkış matrisinin giriş ile aynı boyutta olması isteniyorsa 2.1 denklemi uygulanır [22].

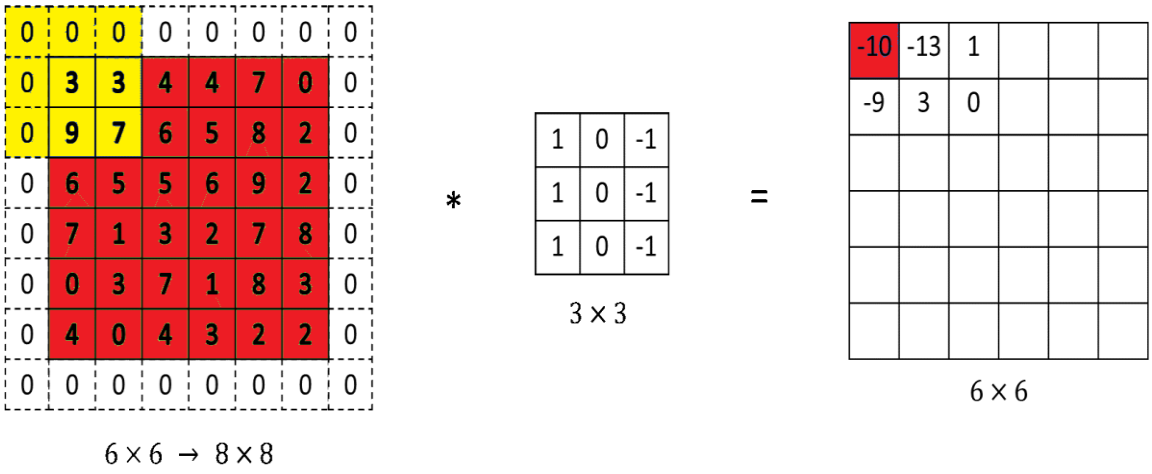
$$(n + 2p - f + 1)x(n + 2p - f + 1) \qquad 2.1$$

Formülde p değeri giriş matrisine eklenen piksel boyutunu ifade etmektedir. Bunu bulmak için 2.2 denklemi kullanılır [22].

$$p = (f - 1)/2 \qquad 2.2$$



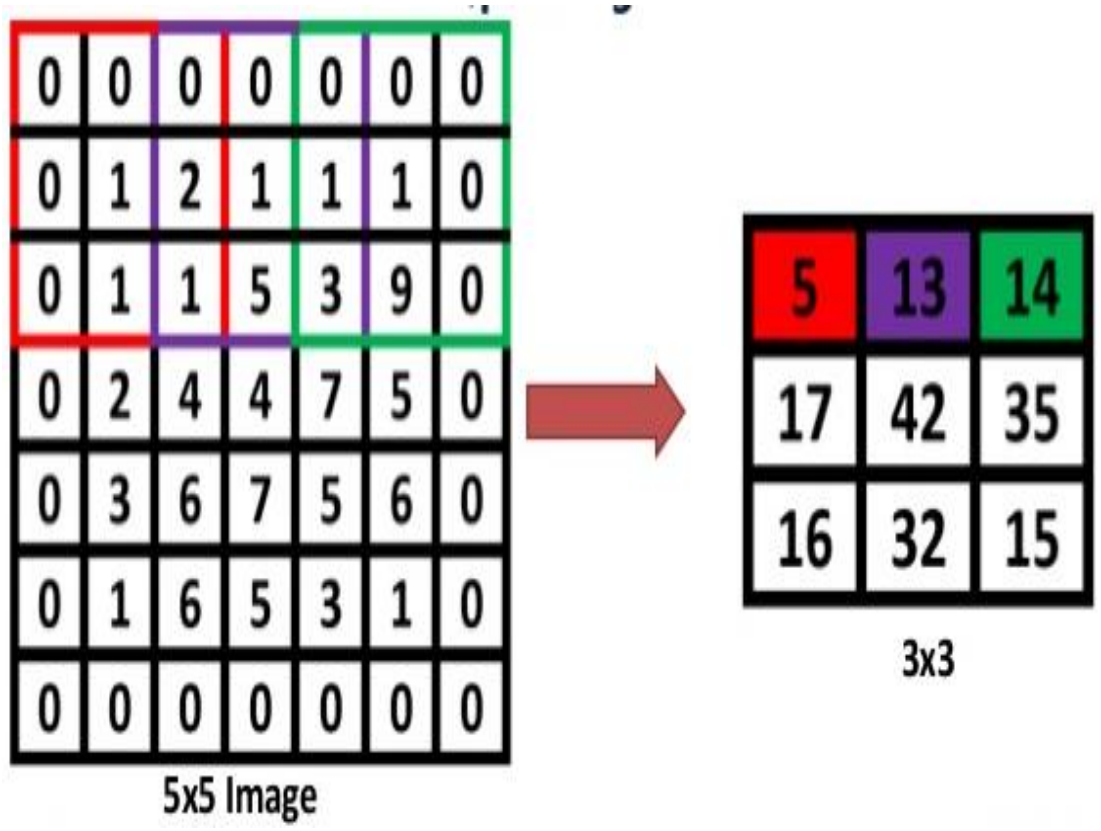
Şekil 2. 19 Sobel filtresi uygulanmış yaprak sarması görseli



Şekil 2. 20 Riksel ekleme işlemi

2.2.2.4.4 Kaydırma adımı (Stride)

Bu değer ile evrişim işlemde kullanmak için filtreyi birer piksellik veya daha büyük adımlarla kaydırma işlemi yapacağının blgisini vermektedir. Bu işlem çıkış boyutunu doğrudan etkilemektedir [22].



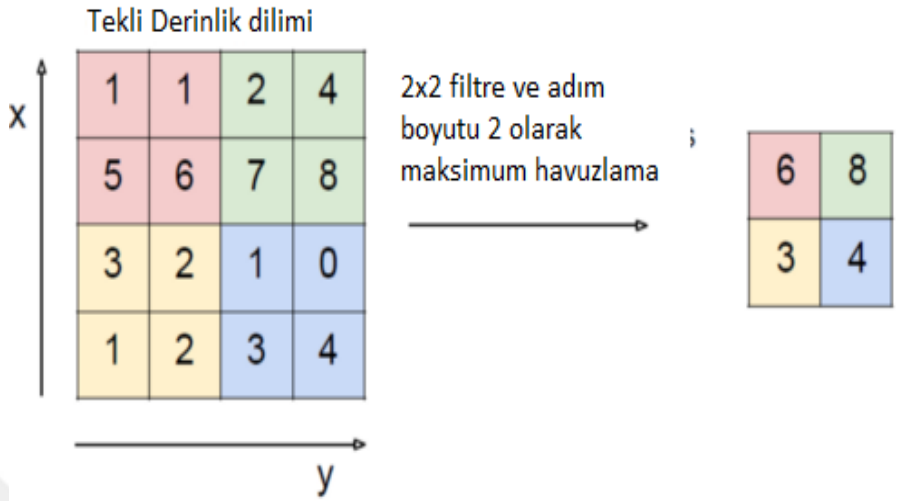
Şekil 2. 21 Kaydırma adımı

2.2.2.4.5 Ortaklama (Pooling)

Evrişimin bu katmanında maksimum seviyede ortaklama yöntemi kullanılmaktadır. Bu katmanda hiçbir öğrenim gerçekleşmez. Giriş matrisinin kanal sayısını sabit tutma yolu ile yükseklik ve genişlik bilgisini azaltır, bu sayede hesaplama karmaşıklığı azaltılmış olur [22].

2.2.2.5 Modüler sinir ağları

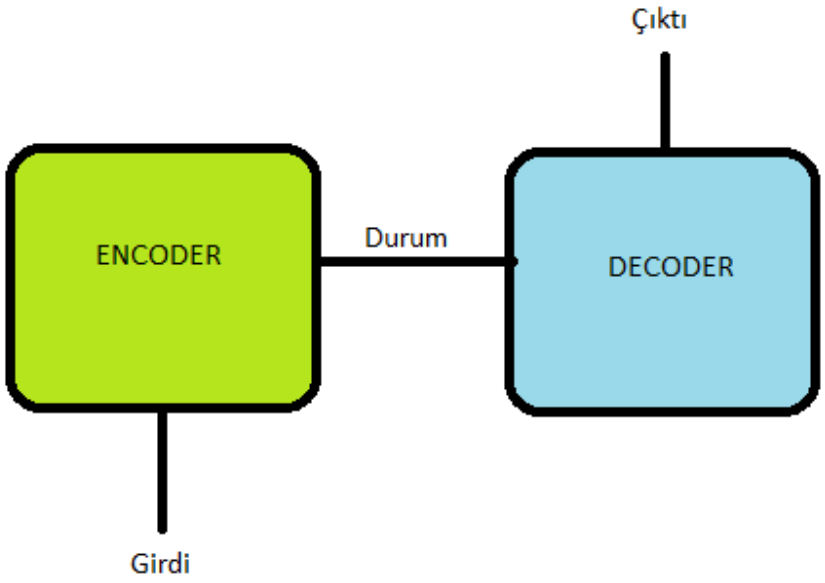
Modüler sinir ağları, birbirlerinde bağımsız sinir ağlarından oluşmaktadır. Her bir sinir ağına verilen girdiler ile alt görevleri tamamlamak için toplu halde çalışırlar. Sonuç olarak bireysel ağlardan gelen verileri toplayan bir araç tarafından çıktı alınır.



Şekil 2. 22 Ortaklama

2.2.2.6 Sıradan sıraya modeler

Sıradan sıraya modeler temel olarak iki adet evrişimli sinir ağından meydana gelir. Burada girişi işleyen bir kodlayıcı ve çıkışı işleyen bir kod çözücü var. Kodlayıcı ve kod çözücü aynı veya farklı parametreleri kullanabilir. Bu model, özellikle girdi verilerinin uzunluğunun çıktı verilerinin uzunluğu ile aynı olmadığı durumlarda geçerlidir.



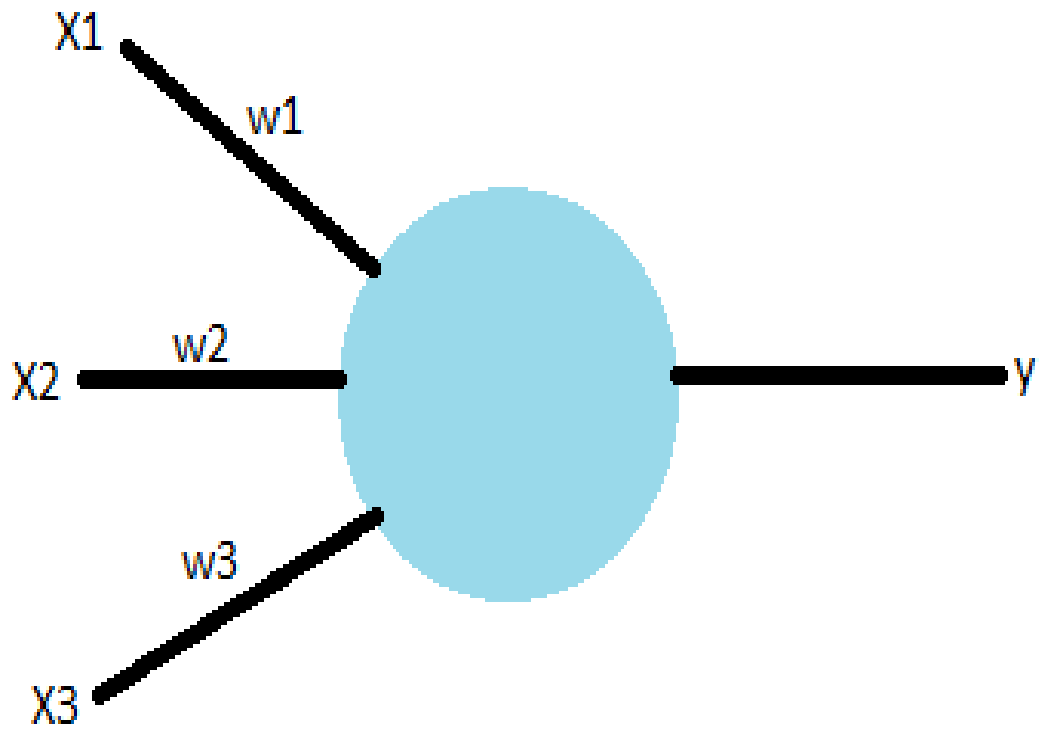
Şekil 2. 23 Sıradan sıraya model yapısı

2.2.2.7 Algılayıcı (Perceptron)

En basit yapay sinir ağı modeli olan perceptron modeli, 1957 yılında Frank RosenBlatt tarafından geliştirilmiştir. Tek bir nörona sahip olan bu sisteme giren bir veya daha fazla giriş değeri bir işlem ve tek bir çıkıştan oluşur.

2.2.2.8 Destek vektör makinesi (SVM)

Destek vektör makinesinin temel amacı özellik sayısının n olduğu uzayda veri noktalarının sınıflandırabilecek birçok boyutlu düzlem bulmaktır. Gözetimli öğrenme yöntemi olan bu algoritmada bir düzlem üzerindeki noktaları ayırma işlemini gerçekleştirmek için bir doğru çizilir.

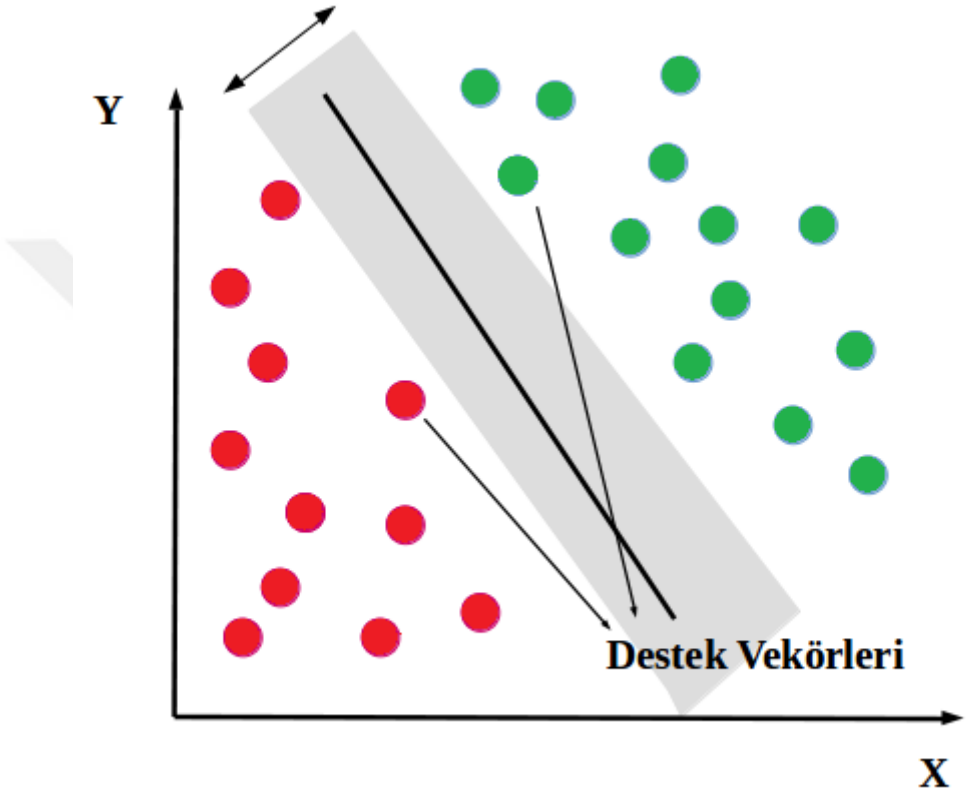


Şekil 2. 24 Perceptron modeli

2.2.2.8 Destek vektör makinesi (SVM)

Destek vektör makinesinin temel amacı özellik sayısının n olduğu uzayda veri noktalarının sınıflandırabilecek birçok boyutlu düzlem bulmaktır. Gözetimli öğrenme

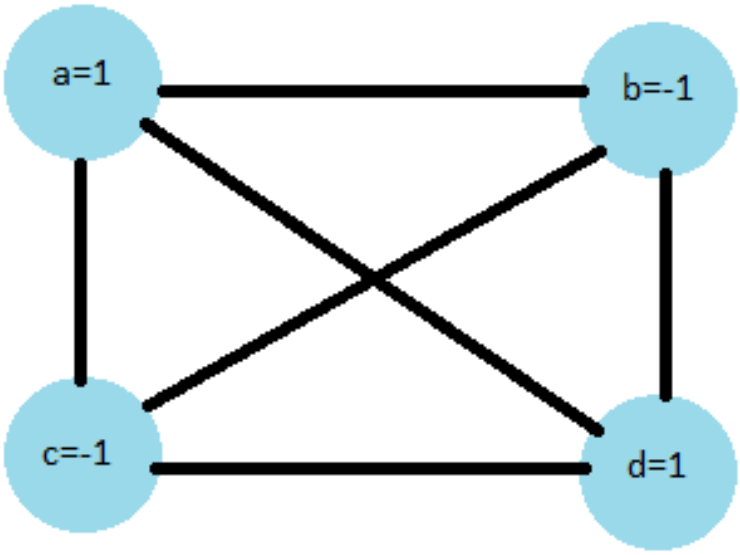
yöntemi olan bu algoritmada bir düzlem üzerindeki noktaları ayırma işlemini gerçekleştirmek için bir doğru çizilir. Bu doğrunun iki sınıf noktaları için maksimum uzaklıkta olması hedeflenmektedir. Küçük ve orta ölçekteki veri setleri için kullanışı bir algoritmadır [23].



Şekil 2. 25 Destek vektörleri

2.2.2.9 Hopfield ağı

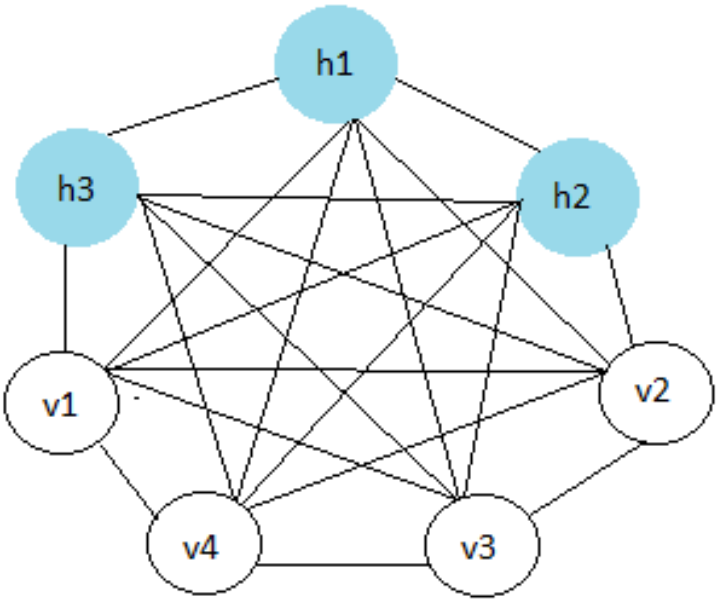
John Hopfield'ın 1982'de gündeme getirdiği ancak 1974 yılında Little'ın tanımladığı tekrarlayan yapay sinir ağı türüdür. Hopfield ağı ikili eşik bağları olan bellek sistemleri olarak çalışmaktadır. Bu değerlerin yerel mimumuma yakın olması hedeflenmektedir. Temel amaç bir veya daha fazla kalıp saklayarak bir parça girdiye dayanarak tüm kalıpları çağırmaktır. Bu ağın özelliklerinden birisi tüm düğümlerin hem giriş hem de çıkış olmasıdır [24].



Şekil 2. 26 Hopfield ağı

2.2.2.10 Boltzman makinesi

Boltzmann makinesi bir tür rastgele tekrarlayan sinir ağıdır. Hopfield ağının rastgele ve üretim karşılığı olarak görülebilir. İç temsilleri öğrenilebilen ilk sinir ağları diyebiliriz. Zor problemleri çözebilirler.



Şekil 2. 27 Boltzman makinesi

2.2.3 Nesne algılama

Nesne algılama algoritmasının doğası gereği üç farklı kestirim türü vardır. Bunlar görüntü sınıflandırma, hem sınfılandırma hem konumlama ve algılama. Görüntü sınıflandırma kısmı, bir görüntüyü sınıflandırır ve nesnenin olasılığını tahmin eder. Geleneksel CNN'ler bu işlemi yaparlar. Sınıflandırma ve lokalizasyon kestirimi ile görüntüdeki bir nesneyi tanır ve nesnenin bulunduğu yeri tahmin eder. Bu işlemi yapan algoritmalara basitleştirilmiş YOLO ve R-CNN diyebiliriz. Algılama kestirimi ise bir görüntüdeki birden fazla nesneyi algılar nessnelerin olasılıklarını ve nerede olduklarının tahmin eder. Bu işlemi yaptığımız algoritmalar ise YOLO ve R-CNN dir. Ayrıca bir diğer kestirim türü ise bir sonraki aşamaya geçerek nesneyi görüntüden ayırarak sadece nesne görüntüsünü alabileceğimiz maskeleme kestirimidir. Bu işlemi yapabileceğimiz algoritma türü ise Mask R-CNN dir [26].

2.2.3.1 Algılama

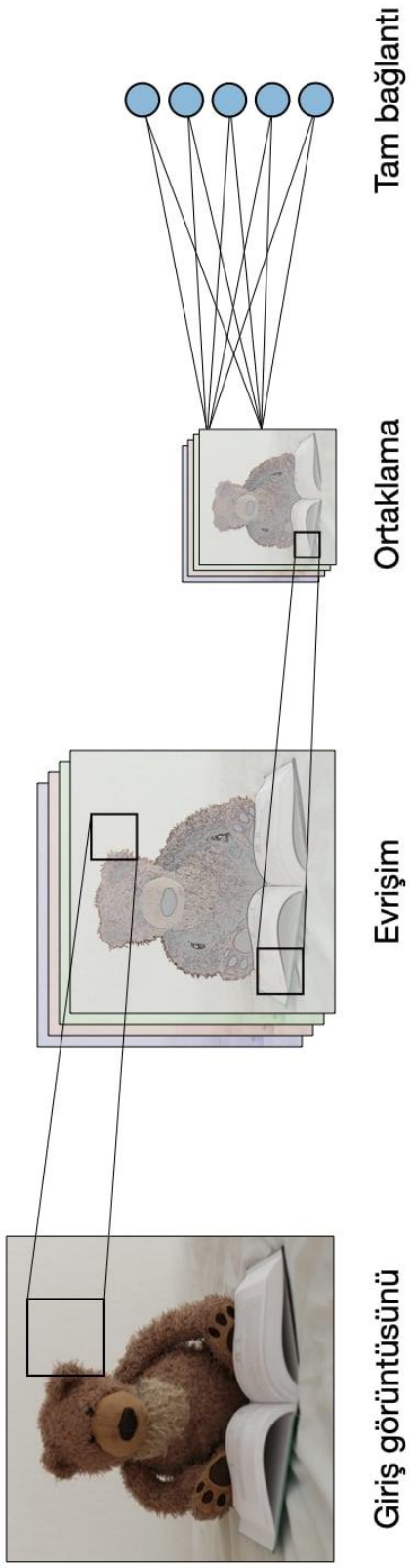
Nesne algılamada, nesneyi konumlandırma veya görüntüdeki daha karmaşık bir şekli tespit etmek işlemlerinden hangisini istediğimize göre farklı yöntemler kullanılmaktadır. Sınırlayıcı kutu ile tespit işleminde görüntüdeki nesnenin bulunduğu yeri belirlerken karakteristik nokta algılama işleminde bir nesnenin şeklini ve özelliklerini algılama işlemi yapılmaktadır [26].

2.2.3.2 Kesiştirilmiş bölgeler

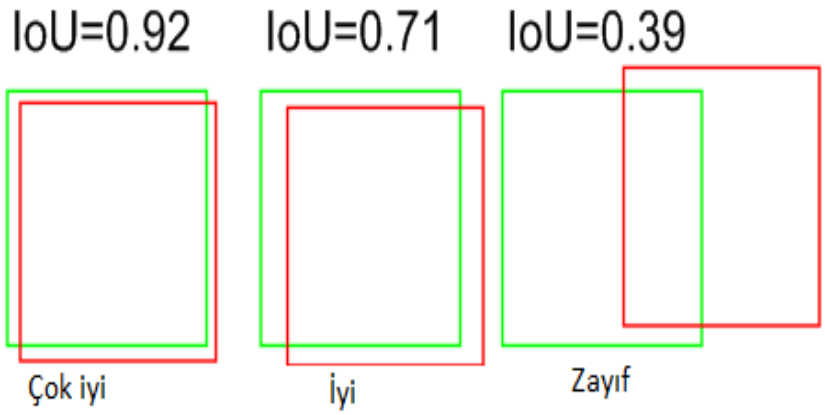
Kesiştrilmiş bölgeler, IoU(Intersection over Union) olarak isimlendirilir. Türkçesi birleştrilmiş sınırlama kutusu olarak bilinen IoU tahmin edilen kutu(B_p) ile gerçek sınırlama kutusu B_a nın ne kadar doğru olarak kesiştiğini ölçen bir fonksiyondur.(2.3)

$$IoU(Bp, Ba) = \frac{Bp \cap Ba}{Bp \cup Ba} \quad (2.3)$$

IoU yu her zaman 0 ile 1 arasında kabul ederiz. N,speten bu oranın 0,5 de büyük eşit olması durumunda sistemin iyi olduğunu kabul edebiliriz [26].



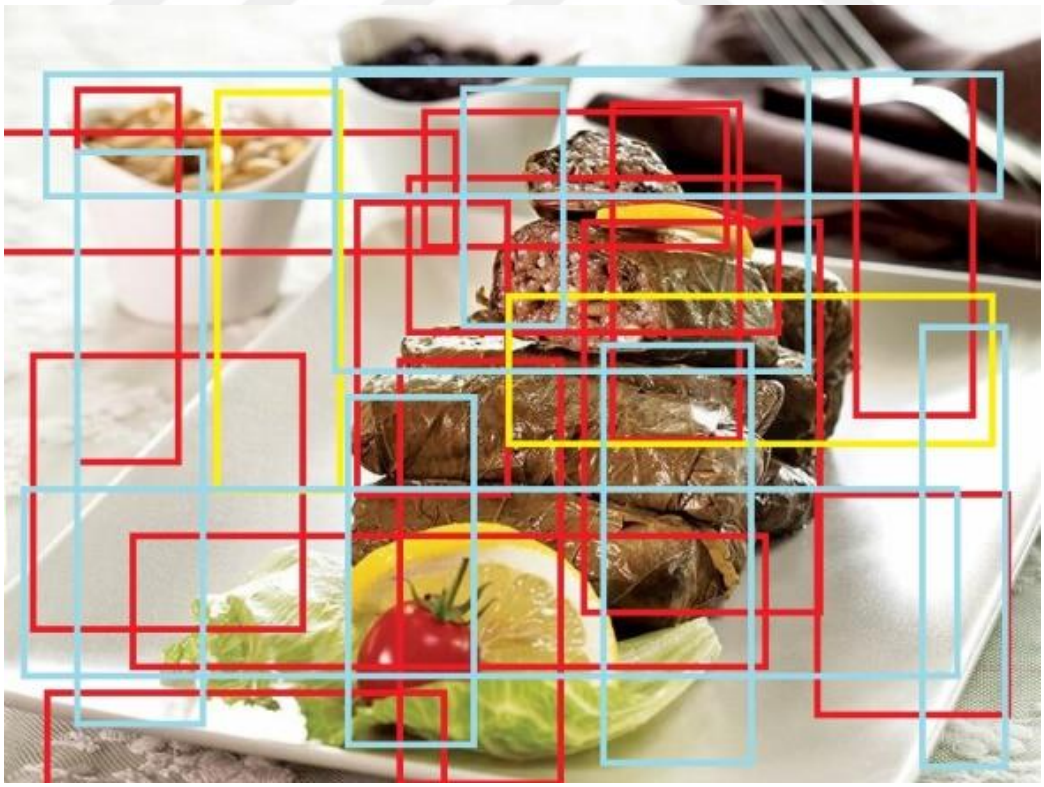
Şekil 2. 28 Temel nesne algılama algoritması [26]



Şekil 2. 29 Kesistirilmiş bölgeler

2.2.3.3 Öneri kutular

Öneri(Anchor) kutular, birbirleriyle örtüşen sınırlayıcı kutuları önceden tahmin için kullanılan yöntemdir. Sistemde yapay sinir ağının birden fazla kutuyu tahmin etmesine olanak sağlanır [26].



Şekil 2. 30 Öneri kutular

2.2.3.4 Maksimum olmayan bastırma

Maksimum olmayan bastırma tekniği, hedef için eşleşen kutuların içerisindeki en uygun olanları seçerek uygun olmayanları kaldırmayı hedefler. Olasılık tahmini 0,6'dan daha küçük sonuç veren kutuları çıkardıktan sonra kalan kutlular ile işlem tekrarlanmaktadır [26].

2.2.3.5 YOLO

You look only once (YOLO), CNN kullanarak nesne tespiti yapmamızı sağlayan bir algoritmadır. Türkçesi "Sadece Bir Kez Bak" olan bu algoritmanın adının böyle seçilme nedeni algoritma nesne tespiti işlemini çok hızlı bir şekilde yapabiliyor olmasıdır.

Şekil 2. 31 Maskimum olmayan bastırma

Sistem çalışmaya başladığında görüntüdeki nesneleri ve koordinatlarını aynı anda tespit etmektedir [27].

Adım 1: Giriş görüntüsü GxG hücrelerine bölünür.
Adım 2: Her bir hücre için aşağıdaki formda y'yi tahmin eden bir CNN çalıştırılır

$$Y= [p_c c_1, c_2, \ldots . c_p, \ldots]^T \epsilon R^{GxGxkx(5+p)}$$

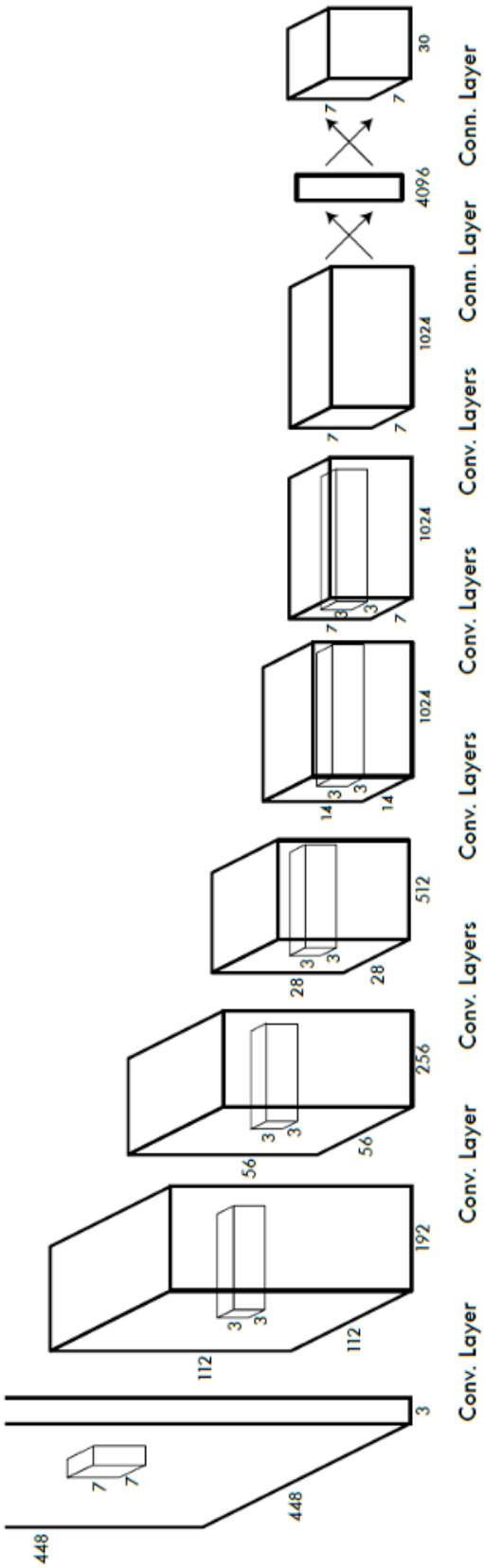
P_c'nin bir nesneyi algılama olasılığının mümkün olduğu durumlarda bx,by,bh,bw tespit edilen olası sınırlayıcı kutu özelliklerindendir.
Adım 3: Çakışan kutulardan kurtulmak için maksimum olmayan bastırma algoritması kullanılır [26].

2.2.3.6 R-CNN

Evrişimsel Sinir Ağı (R-CNN) aracılığıyla bölge araması, olası sınırlayıcı kutuları bulmak için görüntüyü bölümlere ayıran ve ardından sınırlayıcı kutudaki en olası nesneyi bulmak için algılama algoritmasını çalıştıran bir nesne algılama algoritmasıdır [26].
R-CNN mimarisi, görüntüdeki nesne kategorilerini ve bu nesnelere ait sınırlayıcı kutuları (sınırlayıcı kutular) tanımlamak için kullanılır. R-CNN mimarisi, CNN'i birden fazla nesne ile görsel efektlerde kolayca çalıştıramadığımız için geliştirildi. R-CNN'in ana fikri iki adımdan oluşur. Öncelikle seçici arama yoluyla, aday nesnelerin görsel nesneler olarak özellikleri belirlenir. Yaklaşık 2000 bölge belirlendikten sonra, her bölge CNN modeline girilir ve kategori ve sınırlayıcı kutu tahmin edilir.

2.2.3.6.1 Seçici arama

Seçmeli arama, bir görüntüde yakalanması gereken alanı belirlemek için kullanılan bir yöntemdir. Seçici aramada, ilk önce küçükten büyüğe hiyerarşik yapının mantığıyla çalışan küçük alan belirlenir. Ardından, iki benzer alan birleştirilir ve yeni ve daha büyük bir alan görüntülenir. Bu işlem tekrarlanır. Her yinelemede daha büyük alanlar belirir ve görsel nesneler bir dereceye kadar kümelenir. Seçici arama, R-CNN'nin belirli alanları için adayları belirler. Bu bölge adayları, girdi olarak farklı CNN ağlarına girilir.



Şekil 2. 32 Yolo ile obje tanıma [28]

Bölge adaylık sürecinin sonunda yaklaşık 2000 farklı alan ortaya çıkar. Bu 2000 bölge için 2000 CNN ağı kullanılır. CNN ağından gelen öznitelikler kullanılarak, nesnenin kategorisi SVM modelinde belirlenebilir ve sınırlayıcı kutu regresyon modelinde belirlenebilir. Verilen görsel efektlerden 2000 alan olmasına rağmen hepsi kullanılmamaktadır. R-CNN (maksimum olmayan sıkıştırma), doğru sınırlayıcı kutuyu belirlemek için kullanılır. Maksimum olmayan bastırma sınırlayıcı kutuyu kontrol eder, "Kesiştirilmiş bölgeler" (IoU) puanı 0,5'ten büyüktür ve diğer kesişimleri bastırır. Bir nesnenin 0,5'ten büyük birden çok sınırlayıcı kutusu varsa, aralarındaki en yüksek sınırlayıcı kutuyu kullanır ve kullanır [28].

2.2.3.7 Fast R-CNN

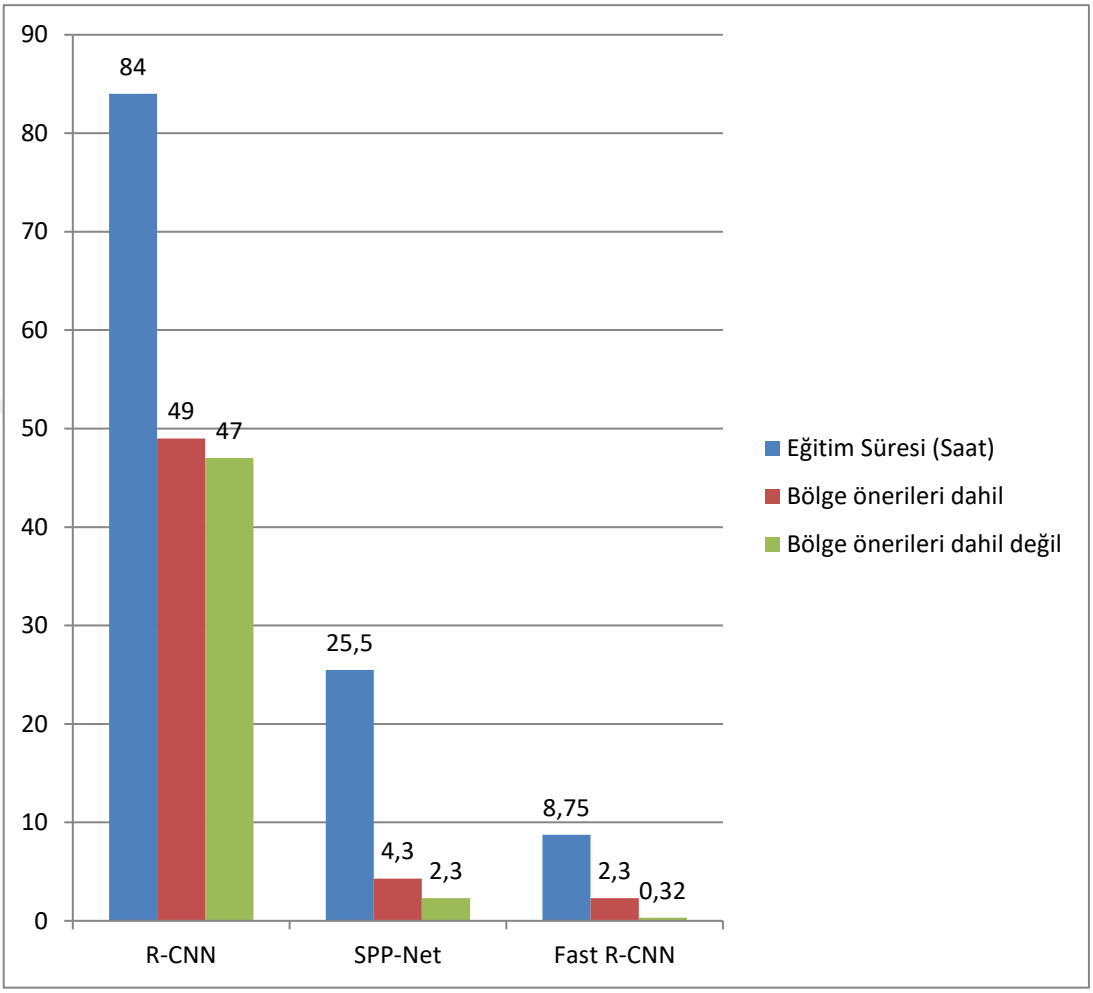
Önceki makalenin aynı yazarı (R-CNN), R-CNN'nin bazı eksikliklerini çözmüş ve Fast R-CNN adı verilen daha hızlı bir nesne algılama algoritması oluşturmuştur. Bu yöntem, R-CNN algoritmasına benzer. Ancak, CNN'ye bölge önerisi sağlamaz, ancak evrişimli özellik haritaları oluşturmak için giriş görüntüsünü CNN'ye sağlar. Evrişimsel özellik haritasından, önerilen bölgeleri tanımlarız ve bunları karelere bükeriz ve tamamen bağlı katmana beslenebilmeleri için bunları sabit bir boyuta yeniden şekillendirmek için ROI havuz katmanını kullanırız. Softmax fonksiyonu kullanrak RoI özellik vektöründen öneri alanının sınfını ve öneri kutusunun offset değerini tahmin ederiz. Her seferinde 2000 bölge önerisini sisteme girmemiz gerekmediğinden sistem daha hızlıdır. Bunun yerine, her görüntü üzerinde yalnızca bir evrişim işlemi gerçekleştirilir ve ondan bir özellik haritası oluşturulur.

Şekil 2.33’deki grafiklerden, Hızlı R-CNN'nin eğitim ve test oturumlarında R-CNN'e göre önemli ölçüde daha hızlı olduğu sonucuna varabilirsiniz. Hızlı R-CNN'in test süresi boyunca performansına baktığınızda, bölge önerileri dahil olmak üzere, bölge önerilerini kullanmama ile karşılaştırıldığında algoritmayı önemli ölçüde yavaşlatır. Bu nedenle, bölge önerileri, Fast R-CNN algoritmasında performansını etkileyen darboğazlar haline gelir [29].

2.2.3.8 Faster R-CNN

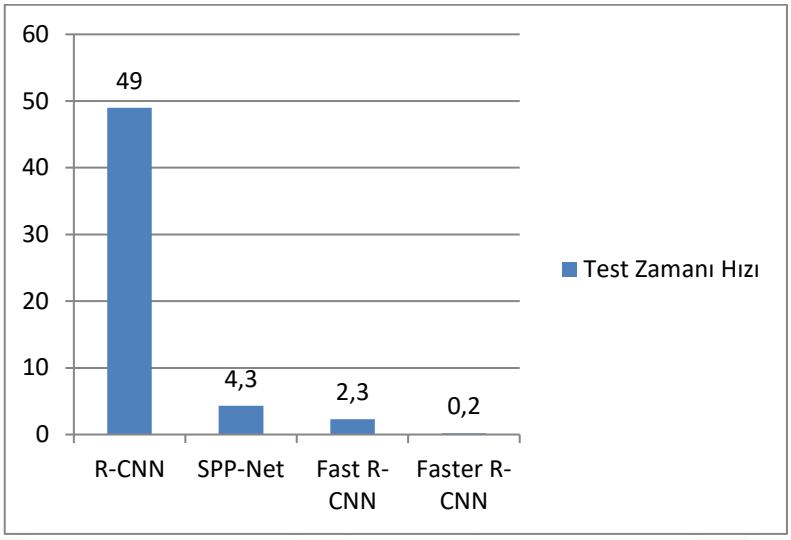
Yukarıda bahsedilen algoritmaların her ikisi de (R-CNN ve Fast R-CNN), alan önerilerini bulmak için seçici aramayı kullanır. Seçici arama, sistemin performansını

olumsuz etkileyen yavaş bir süreçtir.



Şekil 2. 33 R-CNN ve diğerleri karşılaştırması

Bu nedenle Ren Shaoqing ve diğerleri [30], seçici arama algoritmalarını kurtulmamızı sağlayan ve ağın alan önerilerini öğrenmesine olanak sağlayan bir nesne algılama algoritması keşfetti. Hızlı R-CNN'ye benzeyen bu sistemde görüntü evrşimli özellik haritaları veren evrişimli ağa sistem girdisi olarak sağlanır. Özellik haritalarında seçici arama algoritmalarını kullanmak yerine bölgesel önerileri tahmin etmek için farklı bir ağ kullanılır. Ardından, tahmin alanı teklifini yeniden şekillendirmek için RoI havuzlama katmanını kullanın ve ardından, önerilen alandaki görüntüyü sınıflandırmak ve sınırlayıcı kutunun ofset değerini tahmin etmek için bu katmanı kullanınılır.



Şekil 2. 34 R-CNN hız zamanı karşılaştırması

Yukarıdaki grafikten Daha Hızlı R-CNN'nin öncekilere göre hızlı olduğunu görebilirsiniz. Bu sebeple, gerçek zamanlı obje tespiti için kullanılması uygundur [29].

2.2.3.9 Mask R-CNN

Mask R-CNN genişletilmiş bir Faster R-CNN'dir. Peki Mask R-CNN'yi farklı kılan nedir dersek. Mask R-CNN, her bir İlgi Bölgesinde (ROI) pikselden piksele bir şekilde segmentasyon maskelerini tahmin etmek için ek bir dala sahiptir. Daha hızlı R-CNN, ağ girişleri ve çıkışları arasında pikselden piksele hizalama için tasarlanmamıştır. Daha hızlı R-CNN'nin iki çıkışı vardır. Her aday nesne için, bir sınıf etiketi ve bir sınırlayıcı kutu uzaklığı. Mask R-CNN'nin üç çıkışı vardır. Her aday nesne için, bir sınıf etiketi, bir sınırlayıcı kutu uzaklığı ve nesne maskesidir [31]. Mask R-CNN ve Faster R-CNN ortak özellikleri;

-Her ikisi de, sınıflandırma ve sınırlayıcı kutu regresyonu için bir dala sahiptir.

-Her ikisi de görüntüden özellikleri çıkarmak için ResNet 101 mimarisini kullanır.

-Her ikisi de İlgi Alanları Bölgesi'ni (ROI) oluşturmak için Bölge Teklif Ağı'nı (RPN) kullanır.

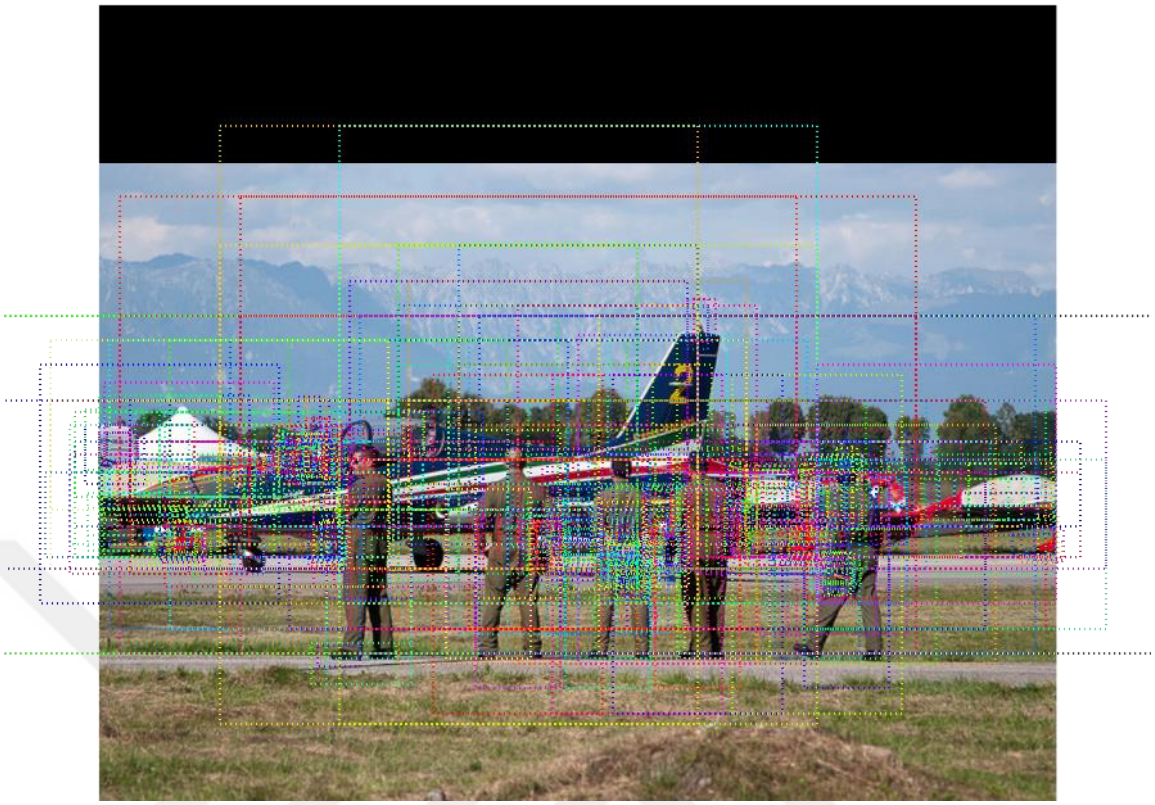
Mask R-CNN nasıl çalışır sorusunun cevabı ise şöyledir. Mask R-CNN modeli iki bölüme ayrılmıştır. Birincisi aday nesne sınırlayıcı kutuları önermek için Region proposal network (RPN) kullanması. İkincisi ise her sınıf için maske oluşturmak için binary maske sınıflandırıcı aşaması. Özellik haritalarını oluşturmak için görüntü CNN üzerinde

çalıştırılır. Görüntü, özellik haritaları oluşturmak için CNN aracılığıyla çalıştırılır. Bölge Teklif Ağı (RPN), CNN kullanarak birden çok hedef bölge (ROI) oluşturmak için hafif bir ikili sınıflandırıcı kullanır. Bunu yapmak için görüntünün üzerindeki 9 bağlantı kutusunu kullanır. Sınıflandırıcı, nesne / nesne yok puanını döndürür. Yüksek objektif puanlara sahip bağlantı noktalarına maksimum olmayan bastırma uygulanır. RoI Align ağı, belirlenen tek bir sınırlayıcı kutu yerine birden çok sınırlayıcı kutu çıkarır ve ardından bunları sabit bir boyuta deforme eder.

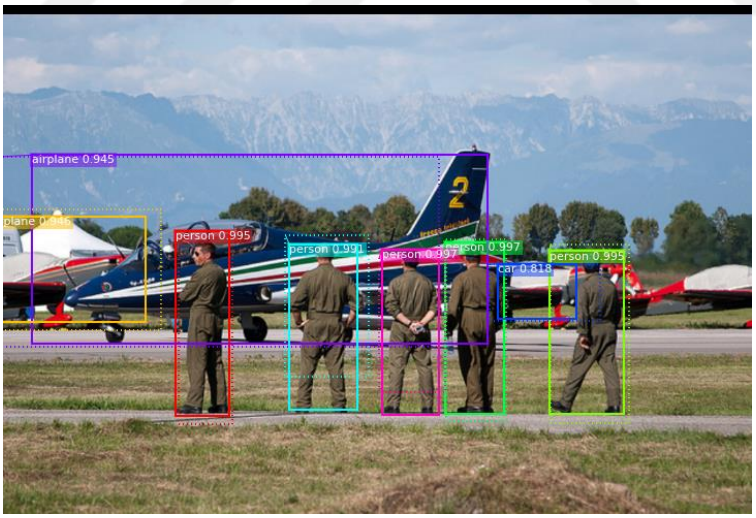
Ardından, sınıflandırma için softmax'ı kullanmak için deforme olmuş özellikleri tamamen bağlı katmana girilir ve sınırlayıcı kutu tahminini daha da iyileştirmek için regresyon modelini kullanınır. Bozulmuş fonksiyon ayrıca, iki CNN'den oluşan ve her RoI için bir ikili maske çıktısı verebilen Maske sınıflandırıcısına girilir. Maske sınıflandırıcı, ağın kategoriler arasında rekabet olmadan her kategori için bir maske oluşturmasına izin verir. R-CNN maskesi, görüntüdeki birden çok nesneyi, farklı ölçeklerdeki nesneleri ve üst üste binen nesneleri algılamak için konumlandırma çerçevelerini kullanır. Bu, nesne algılamanın hızını ve verimliliğini artırır. Bağlantı kutuları belirli yükseklik ve genişlik değerlerine sahip birden çok sınırlayıcı kutulara verilen isimdir. Bu kutular, tespit edilecek belirli nesne sınıfının ölçeğini ve en boy oranını yakalamak için tanımlanmıştır. Mask R-CNN, bir görüntüdeki birden çok nesneyi veya birden çok nesne örneğini tahmin etmek için binlerce tahmin gerçekleştirebilir. Arka plan sınıfına ait bağlantı kutuları silinerek nihai nesne tespiti tamamlanır ve kalan bağlantı kutuları güven puanlarına göre filtrelenir. 0,5'ten büyük IoU'ya sahip bağlantı kutuları buluyoruz. En yüksek güven puanına sahip bağlantı kutusunu seçmek için aşağıda açıklanan maksimum olmayan gizleme" seçeneğini kullanınılır [31].

2.2.3.10 AlexNet

2012'de yayınlanmasından bu yana, makale [33]. Alex Krizhevsky, Ilya Sutskever ve Geoffrey E. Hinton tarafından yazılmıştır ve 80.000'den fazla alıntı almıştır ve derin öğrenme topluluğundaki en etkili makale olarak bilinmektedir.



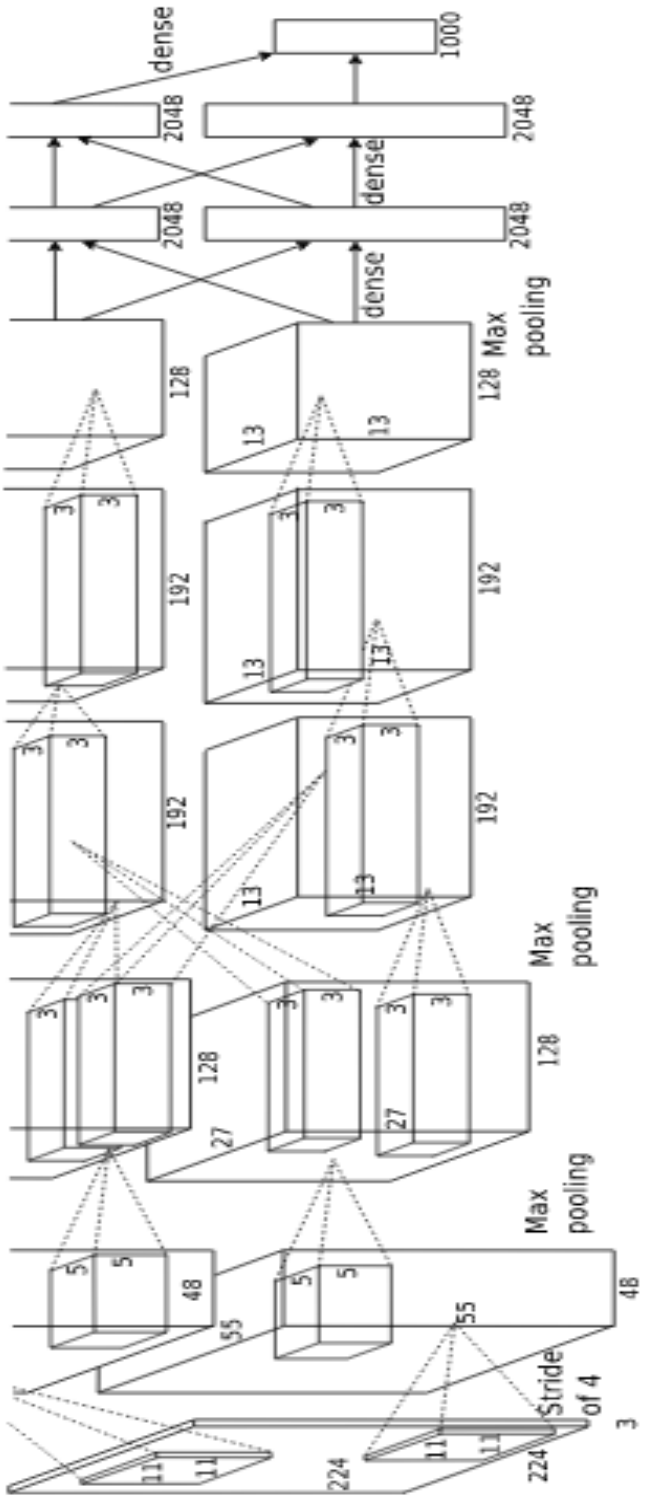
Şekil 2. 35 Mask R-CNN öneri kutular



Şekil 2. 36 Mask R-CNN maksimum olmayan bastırma

5 evrişimli katman ve 3 tam bağlantılı katman içerir. ReLU doymamış doğrusal olmayan f (x) = maks (0, x) 'i tanıttılar ve bu daha sonra aktivasyon fonksiyonunun altın standart seçeneği haline geldi. Tanh gibi doymuş doğrusal olmayan işlevleri kullanarak benzer doğruluğu elde etmek için gereken süre 6 kat daha hızlıdır.

Network mimarisi;

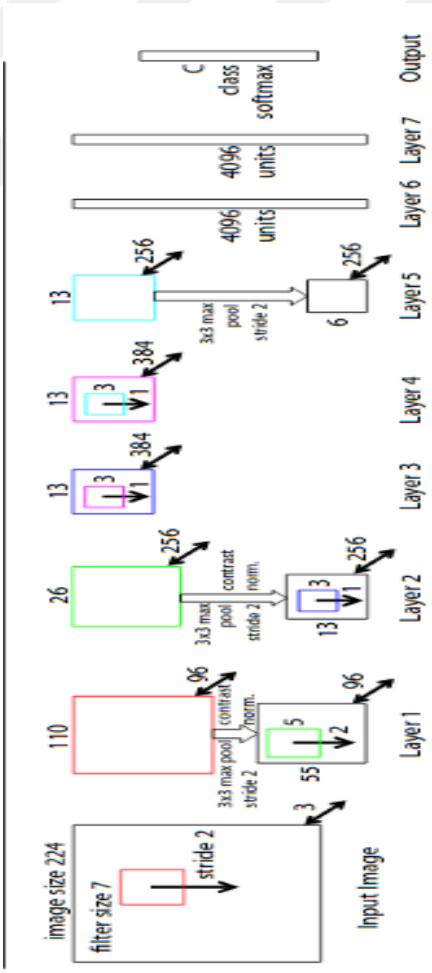


Şekil 2. 37 AlexNet mimarisi[34]

Ağın toplam 60 milyon parametresi var ve eğitim için kullanılan görüntülerde sadece 1,2 milyon görüntü var. Aşırı uyumu önlemek için, giriş görüntüsünün düşürme ve geliştirme işlevlerini kullandılar. Tam bağlantılı katmanlar arasında Dropout'u kullanınır. Dropout, yakınsama için gereken yineleme sayısını yaklaşık iki katına çıkarır [34].

2.2.3.11 ZFNet

AlexNet'in 2012 deki yarışmayı kazanmasının ardından yapılan yarışmalarda Matthew Zeiler ve Rob Fergus tarafından [35]. makalesi 2014 de yayınlanan ZFNet 2013 yılı ImageNet yarışması kazananı olmuştur. Hata oranının %11,2'ye indiği bu mimari AlexNet'in geliştirilmiş şeklidir.

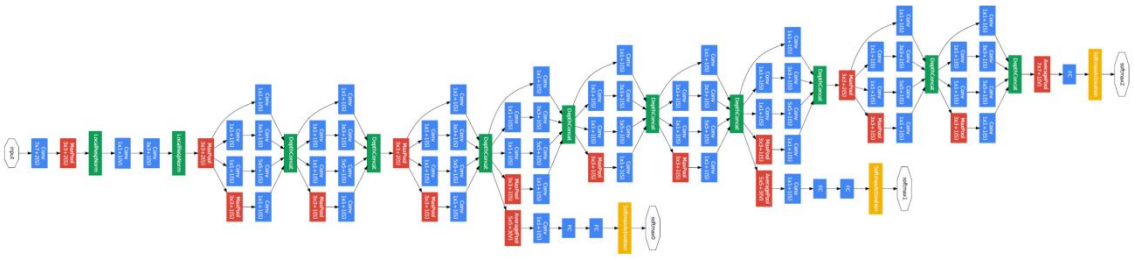


Şekil 2. 38 ZFNet mimarisi [35]

ZFNet'in ilk katmanında bulunan 7x7'lik filtreler ile pooling katmanında bulunan 2 adet kaydırma adımı kullanılmıştır. Bu işlemin ilk hedefi birinci evrişim katmanında bulunan daha küçük filtre boyutunun, girişteki birçok pikselin korunmasına yardımcı olmaktadır. Aktivasyon fonksiyonu seçiminde ReLu kullanılırken hata fonksiyonu için ise Cross-Entropy Loss ve eğitim için Stochastic Gradient Descent kullanılmıştur. Eğitim yapılırken ekran kartı olarak GTX 580 GPU üzerinde 12 gün eğitim yapılmıştır.

2.2.3.12 GoogleNet

GoogleNet [36] Google'dan Christian Szegedy ve diğer birçok araştırmacı, şimdiye kadar 30.000 alıntı alan bu makaleyi yayınladı. Bu, VGGNet ile paralel olarak geliştirilmiştir. Bu zamana kadar, insanlar daha büyük bir ağa sahip olmanın modelin performansını artıracağını fark ettiler. Boyut bazında derinliği (ağ seviyesi sayısı) veya genişliği (seviye başına birim sayısı) artırabiliriz. Derinlik, parametreleri büyük ölçüde artıracaktır ve eğitim seti sınırlıdır, bu da aşırı uyuma yol açacaktır. Genişlik, hesaplama gereksinimlerini artırır [filtre sayısındaki tek tip artış, hesaplamada ikinci bir artışa yol açar]. Basit bir dönüşüm katmanını başlangıç modülüyle değiştirdiler. İlk modül 3 * 3 dönüştürme, 5 * 5 dönüştürme, 3 * 3 maks_pooling ve 1 * 1 dönüştürme içerir. Her modül, üst katmanın özellik haritasında bağımsız olarak çalışır. Son olarak, tüm özellik haritalarını birbirine bağlanır. O zamanlar GPU belleği çok küçüktü, bu nedenle 512 derinlikli bir filtre haritasında 5 * 5 çekirdek çalıştırmak çok pahalıydı, bu nedenle derinliği azaltmak için 1 * 1 evrişim kullandılar [34].



Şekil 2. 39 GoogleNet mimarisi [36]

2.2.3.13 VGGNet

Makale, yayınlanmasından bu yana Karen Simonyan ve Andrew Zisserman

tarafından yazılmıştır [37] ve 55.000'den fazla alıntı almıştır. Bu makale, convnet mimarisi tasarım derinliğinin önemli yönüne odaklanmaktadır. Belirli bir görüntü bir grup dönüşüm katmanından geçer. Burada küçük alıcı alanı 3 * 3 olan bir filtre kullanırız ve dönüşüm aralığı 1'e sabitlenir. Uzay havuzu, en büyük 5 havuz katmanı tarafından yürütülür. Evrişimden sonra, görüntü doldurularak uzamsal çözünürlük korunur. Bir dizi conv katmanından sonra FC katmanları bulunur. Tüm gizli katmanlar ReLU ile donatılmıştır. Ağı farklı derinliklerde test ettiler [34].

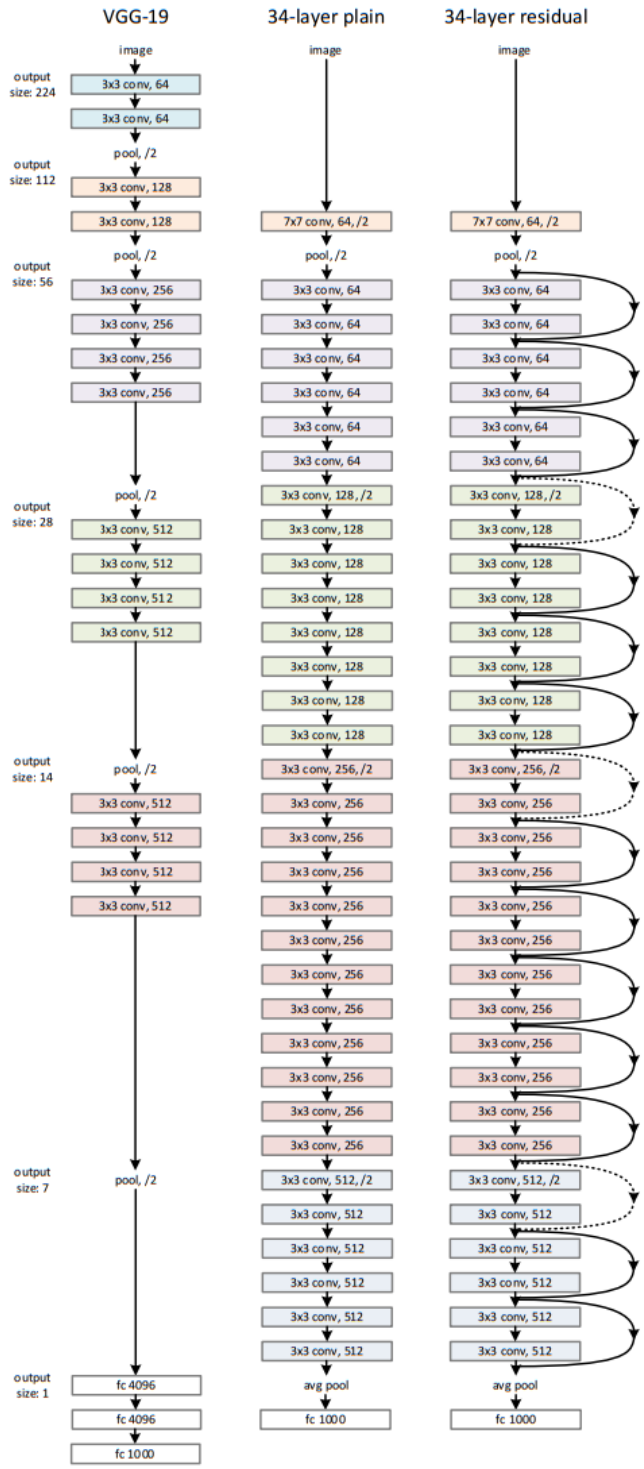
ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Şekil 2. 40 VGGNet mimarisi [37]

2.2.3.14 Resnet

Resnet [38] kendinden önceki tüm mimarilerden daha derin tasarlanmış bir mimaridir. 152 katmandan oluşmaktadır. Hata oranı insandan daha düşük seviyede olan bu mimarinin hata oranı %3,6'dır. İnsanlarda bu oran %5-10 arasındadır. Bu başarılı mimari ImageNet 2015 kazananı olmuştur. 34 katmanlı düz ağ mimarisinini bulalnan bu

ağ VGG-19 dan esinlenmiştir. Bu kısayol bağlantıları daha sonra mimariyi artık ağa dönüştürür. Şekilde gösterilmiştir.



Şekil 2. 41 ResNet mimarisi [38]

2.2.3.15 LeNet-5

1998'de Yann LeCun, Leon Bottou, Yoshua Bengio ve Patrick Haffner "Belge Tanıma Uygulanan Gradyan Tabanlı Öğrenme" araştırma makalesinde leNet'i tanıttı. Makalenin listelenen yazarlarının çoğu, derin öğrenme alanına bazı önemli akademik katkılar sağlamaya devam ediyor. Lenet-5 sayıları algılama için kullanılan eski bir yöntem olarak günümüze gelmiştir. Softmax algoritmasını kullanan bu sistem ortalama ortaklama "average pooling" işlemini kullanmaktadır.

2.2.4 Derin öğrenme kütüphaneleri

Derin öğrenme alanında birçok kütüphane bulunmasa bu kısımda sadece en çok kullanılanları anlatacağız.

2.2.4.1 Keras

Keras, Python ile yazılmış açık kaynak kodlu yapay sinir ağı kütüphanesidir. Keras bazı diğer kütüphaneler ile birlikte çalışabilir. Bunlardan en çok birlikte kullanılan tensorflow'dur.

- Kolay ve hızlı bir şekilde model oluşturmamıza olanak sağlar ve modelleri CPU ve GPU'da sorunsuz bir şekilde çalışır.
- Evrişimli sinir ağları ve yinelemeli sinir ağlarını destekler
- Kütüphaneyle alakalı internette çok fazla kaynak bulunmaktadır.

2.2.4.2 Caffe

Caffe, ifade, hız ve modülerliği dikkate alan derin bir öğrenme çerçevesidir. Berkeley Vision and Learning Center (BVLC) ve topluluk katkıda bulunanlar tarafından geliştirilmiştir. Google'ın DeepDream'i, Caffe çerçevesine dayanmaktadır. Çerçeve, bir Python arayüzüne sahip BSD lisanslı bir C ++ kitaplığıdır.

Etkileyici mimari, uygulamayı ve yeniliği teşvik eder. Modeller ve optimizasyonlar konfigürasyonla tanımlanır ve kodlanmaları gerekmez. Tek bir bayrak ayarlayarak GPU makinelerinde eğitim yapılır ve ardından CPU ile GPU arasında geçiş yapmak için ticari cihaz kümelerine veya mobil cihazlara dağıtılabilir.

Genişletilebilir kod, olumlu gelişimi destekler. Caffe, ilk yılında 1.000'den fazla geliştirici tarafından desteklendi ve birçok önemli değişiklik yapıldı. Bu katılımcılar

sayesinde çerçeve, kod ve modeller açısından en son teknolojileri takip ediyor.

Hız, Caffe'yi araştırma deneyleri ve endüstri dağıtımı için mükemmel bir seçim haline getirir. Caffe, günde 60 milyondan fazla görüntüyü işlemek için bir NVIDIA K40 GPU * kullanabilir. Bu, 1 milisaniye / görüntünün muhakeme yeteneği ve 4 milisaniye / görüntü öğrenme yeteneği ve daha yeni kitaplık sürümü ve donanım hızı daha hızlıdır.

2.2.4.3 Pytorch

PyTorch, geliştiricilerin kolayca derin öğrenme modelleri oluşturmasını sağlayan bir Python kitaplığıdır. Bir grafik işleme birimi kullanan PyTorch, sağladığı esneklik ve hız nedeniyle günümüzde de oldukça popülerdir.

PyTorch'un başarısının ana nedenlerinden biri, tamamen Pythonic olması ve kolayca sinir ağı modelleri oluşturabilmesidir. PyTorch, rakiplerine kıyasla pazarda nispeten genç bir konumda olmasına rağmen, geliştirme ivmesi iyidir.

- PyTorch tensörleri NumPy ndarray'lere benzer ve GPU'da çalıştırma seçeneğine sahiptir.
- GPU (grafik işleme birimi), derin öğrenme modellerini eğitme hızını büyük ölçüde artıran yüzlerce basit çekirdekten oluşur.
- Pytorch'un kullanımı basittir ve bu durum geliştiriciler için öğrenme eğrisinin daha kısa olmasını sağlar.
- Python ile derinlemesine entegre olduğu için python derleyicileri pytorch için de iyi bir hata ayıklayıcı olur.
- Veri paralelliği özelliği sayesinde hesaplamalar birden çok CPU veya GPU üzerine dağıtılabilir.
- Dinamik hesaplamalı grafikleri desteklediğinden ağ davranışı çalışma zamanı içinde değiştirilebilir olmaktadır.
- Hibrit uç ile istekli mod ve grafik modu olmak üzeri iki çalışma modu sağlar
- Geniş bir geliştirici topluluğu sayesinde sürekli gelişen bir kitaplıktır.
- ONNX desteği sayesinde geliştiriciler modellerini farklı araçlar arasında kolaylıkla değiştirebilir.
- Bulut desteği sayesinde büyük işlemleri bulut sunucular üzerinde kolaylıkla

yapabiliriz.

Tablo 2. 1 Performans tablosu Yüksek daha iyi

	AlexNet	VGG-19	ResNet-50
Tensorflow	1422±27	66±2	200±1
PyTorch	1547±316	119±1	212±2

2.2.4.4 TensorFlow

TensorFlow, makine öğrenimini daha hızlı ve daha kolay hale getirebilen, dijital bilgi işlem için Python dostu bir açık kaynak kitaplığıdır.

Makine öğrenimi karmaşık bir konudur. Bununla birlikte, makine öğrenimi çerçeveleri (Google'ın TensorFlow'u gibi) veri edinme, modelleri eğitme, tahminler sağlama ve gelecekteki sonuçları iyileştirme sürecini basitleştirdiği için, makine öğrenimi modellerini uygulamanın zorluğu ve zorluğu eskisinden çok daha az zordur.

Google Brain çalışma takımı tarafından ortaya çıkarılan TensorFlow, sayısal bilgi işlem ve büyük ölçekli makine öğrenimi için imkan sağlayan açık kaynaklı kitaplıktır. TensorFlow, çok sayıda makine öğrenimi ve derin öğrenme (sinir ağı olarak da adlandırılır) modellerini ve algoritmalarını kullanmamızı sağlayan ortak bir dil sağlar. Bu işlemleri yaparken programlama dili olarak C++'ı temel alır. Ayrıca çerçeveyi kullanarak uygulamalar oluşturmak için Python kullanır. TensorFlow, el yazısıyla yazılmış rakam sınıflandırması, görüntü tanıma, tekrarlayan sinir ağları, makine çevirisi için sıralı modeller, doğal dil işleme ve diferansiyel denklemlerin çözümü için derin sinir ağlarını eğitebilir ve çalıştırabilir. En önemlisi, TensorFlow, büyük ölçekli üretim tahminlerini desteklemek için eğitim için kullanılan aynı modeli kullanır. TensorFlow geliştiricilere verilerin bir grafik veya bir dizi işleme düğümündeki hareketlerini izlemeye imkan sağlayan akış grafikleri oluşturmasına izin verir. Grafikteki her düğüm gerçekte matematiksel bir bağlantıyı temsil ederken her bağlantı veya kenar çok boyutlu bir veri dizisi veya tensöre eşdeğerdir. TensorFlow, programcılara tüm bu işlevleri Python dili kullanarak sağlar. Python'un öğrenilmesi ve kullanılması kolaydır. TensorFlow'daki düğümler ve tensörler Python'a ait nesneler iken Tensorflow'un kendisi bir Python programıdır. Ancak, büyük matematiksel işlemler Python'da gerçekleştirilmez.

TensorFlow aracılığıyla kullanılabilen dönüştürme kitaplıkları, C ++ ikili dosyaları olarak kullanılmaktadır. Python sadece çeşitli parçalar arasındaki iletişimi sağlar ve bunları birbirine bağlamak için üst düzey programlama soyutlamaları sağlar. TensorFlow herhangi bir sistemde çalıştırılabilir: yerel bilgisayar, bulut sistemi, iOS ve Android ürünler, CPU veya GPU. Google'ın kendi bulut sistemini kullandığınızda daha hızlı çalışması için özel olarak geliştirilmiş TensorFlow işleme birimi (TPU) çipinde TensorFlow'u çalıştırabilirsiniz. Bununla birlikte, TensorFlow tarafından oluşturulan sonuç modeli, tahmin için kullanılacak çoğu cihazda dağıtılabilir.

TensorFlow 2.0 (Ekim 2019'da piyasaya sürüldü), kullanımı kolaylaştırmak (örneğin, model eğitimi için daha basit Keras API'sini kullanarak) ve daha yüksek performans sağlamak için çerçeveyi kullanıcılardan aldığı geribildirimlere dayalı olarak çeşitli yollarla geliştirdi. Yeni API ile, dağıtılmış eğitimin işlemi eskiye göre daha kolay yapılmaktadır ve TensorFlow Lite desteği sayesinde modellerin çeşitli platformlara uyum sağlamasını kolaylaştırmaktadır.. Bununla birlikte, TensorFlow 2.0'ın yeni özelliklerinden tam olarak yararlanmak için TensorFlow'un önceki sürümleri için yazılan kodun yeniden yazılması gerekir bazen sadece küçük değişiklikler, hatta bazen kapsamlı değişiklikler. TensorFlow'un makine öğrenimi geliştirmeye sağladığı en büyük fayda soyutlamadır. Geliştiriciler, algoritmayı uygulamanın belirli ayrıntılarıyla uğraşmak zorunda kalmadan ve bir işlevin çıkışını başka bir işlevin girişine bağlamanın doğru yolunu bulmak zorunda kalmadan uygulamanın geniş çerçevesine odaklanabilir. TensorFlow, perde arkasındaki ayrıntılardan sorumludur.

TensorFlow, TensorFlow uygulamalarında hata ayıklaması ve iç gözlem yapması gereken geliştiriciler için daha fazla kolaylık sağlar. İstenen yürütme modu, grafiği opak biçimde tek bir nesne olarak oluşturmak ve hemen değerlendirmek yerine, her grafik işlemini ayrı ayrı şeffaf bir şekilde değerlendirmenize ve değiştirmenize olanak tanır. TensorBoard görselleştirme paketi, grafiklerin web tabanlı etkileşimli bir gösterge panosu aracılığıyla nasıl çalıştırıldığını incelemenizi ve analiz etmenizi sağlar. TensorFlow ayrıca Google'ın A-list işletme organizasyonunun desteğinden de yararlanır. Google yalnızca projenin hızlı gelişimini teşvik etmekle kalmadı, aynı zamanda TensorFlow çevresinde dağıtımı ve kullanımı kolaylaştıran birçok önemli ürün geliştirdi [44].

2.2.4.5 Theano

LISA laboratuvarlarında makine öğrenimini verimli ve hızlı hale getirmek üzere geliştirilmiştir. İsmini bir yunan matematikçisinden almaktadır. Çok boyutlu matrisleri içeren matematiksel ifadeleri optimize etmek üzere çalışan açık kaynak kodlu Python kütüphanesidir.

2.2.4.6 Pylearn2

Theano ile aynı üniversite olan Montreal üniversitesinde geliştirilmiştir. Algoritmaların yanı sıra pythonda yazılmış derin öğrenme koleksiyonu sunar. Esnekliğe odaklanan bu kütüphanenin hedef kitlesi makine öğrenimi üzerine çalışmalar yapan kişilerdir.

2.3 Derin öğrenme kullanım alanları

Gerçek dünyadaki derin öğrenme uygulamaları günlük hayatımızın bir parçasıdır, ancak çoğu durumda, ürün ve hizmetlere iyi bir şekilde entegre edilmiştir, böylece kullanıcılar arka planda gerçekleştirilen karmaşık veri işlemeyi bilmezler. Bazı örnekler şunları içerir:

2.3.1 Kanun yaptırımı

Derin öğrenme algoritmaları, işlem verilerini analiz edebilir ve olası dolandırıcılık veya suç faaliyetlerini gösteren tehlikeli kalıpları belirlemek için bunlardan öğrenebilir. Konuşma tanıma, bilgisayar görüşü ve diğer derin öğrenme uygulamaları, kolluk kuvvetlerinin çok sayıda veriyi daha hızlı ve doğru bir şekilde analiz etmesine yardımcı olan ses ve video kayıtlarından, görüntülerden ve belgelerden kalıplar ve kanıtlar çıkararak araştırma ve analizin verimliliğini ve etkililiğini artırabilir [45].

2.3.2 Finansal hizmetler

Tahmine dayalı analitiği düzenli olarak hisse senetlerinin algoritmik ticaretini desteklemek, kredi onaylarında iş risklerini değerlendirmek, dolandırıcılığı tespit etmek ve müşterilerin kredi ve yatırım portföylerini yönetmeye yardımcı olmak için kullanınır [45].

2.3.3 Müşteri ilişkileri

Birçok kuruluş, derin öğrenme tekniklerini müşteri hizmetleri süreçlerine entegre eder. Chatbot'lar (çeşitli uygulamalarda, hizmetlerde ve müşteri hizmetleri portallarında kullanılır) doğrudan bir yapay zeka şeklidir. Geleneksel sohbet robotları, genellikle çağrı merkezlerine benzer menülerde bulunan doğal dili ve hatta görsel tanımayı kullanır. Bununla birlikte, daha karmaşık sohbet robotu çözümleri, belirsiz sorulara birden çok yanıt olup olmadığını belirlemeye çalışır. Sohbet botu daha sonra alınan yanıtlara göre bu soruları doğrudan yanıtlamaya çalışır veya konuşmayı bir insan kullanıcıya yönlendirir. Apple'ın Siri, Amazon Alexa veya Google Assistant gibi sanal asistanları, ses tanımayı etkinleştirerek sohbet robotu konseptini genişletiyor. Bu, kullanıcıları kişiselleştirilmiş bir şekilde çekmek için yeni bir yol yaratır [45].

2.3.4 Sağlık hizmeti

Hastane kayıtlarının ve görüntülerinin dijitalleştirilmesinden bu yana, sağlık sektörü derin öğrenme yeteneklerinden büyük ölçüde yararlandı. Görüntü tanıma uygulamaları, daha kısa sürede daha fazla görüntüyü analiz etmelerine ve değerlendirmelerine yardımcı olmak için tıbbi görüntüleme uzmanlarına ve radyologlara destek sağlayabilir. EEG sinyalleri ile çalışmalar kanserli hücrelerin görüntüleri üzerindeki çalışmalar ve benzeri birçok çalışma sürekli olarak yapılmaktadır.

3. BÖLÜM

GEREÇ VE YÖNTEM

3.1 Gereç

Tez çalışmasında kullanılan yemek görselleri daha önceden hazırlanmış herhangi bir veri setinde bulunmamaktaydı. Bu sebepten ötürü eğitim için kullanılan verileri kendimiz oluşturmamız gerekti. Eğitim için kullanılan görselleri internet üzerinden arama motorlarında anahtar kelime aratarak yazdığımız ve [46] numaralı kaynakta bulunan EK 1'de verilen python kodu ile 200'lü setler halinde bilgisayarımıza indirdik. Daha sonrasında veri kirliliğini önlemek amacıyla anahtar kelime ile alakasız olan görselleri veri setimizden kaldırdık. Bu aşamadan sonra eğitim için gerekli olan piksel boyutlarına [46] numaralı kaynakta bulunan EK 2'de verilen python kodu ile görsellerimizin tamamını 800 piksele 600 piksel boyutlarına indirdik. Ancak eğitim için daha hazır olmayan bu verilerimizi eğitime hazır hale getirmek için MacOs işletim sistemli bilgisayarımıza indirdiğimiz python programlama dili ile yazılmış olan LabelMe yazılımını kullanarak görsellerimizi etiketlememiz gerekmektedir. Bu yöntemi kullanarak görsellerimizin her birini tek tek etiketledik. Etiketlenmiş örnek görseller şekil 3.1-4'de gösterilmektedir.

Şekil 3. 1 Etiketlenmiş et görseli

Şekil 3. 2 Etiketlenmiş yemek görseli

Şekil 3. 3 Etiketlenmiş baklava görseli

Şekil 3. 4 Etiketlenmiş fasülye görseli

Etiketlenmiş görsellerin sınıflara ayrılmış sayısı Tablo 3.1’ de verilmiştir.

Tablo 3. 1 Etiket sayıları

Sınıf	Etiket sayısı
Baklava	95
Et sote	57
Pirinç pilavı	81
Menemen	40
Tavuk sote	65
Yaprak sarma	74
Barbunya yemeği	14
Bulgur pilavı	52
Mantı	66
Mercimek çorbası	72
Trileçe tatlısı	51

Taze fasülye	87
Çipura	60
Karnıyarık	93
Kuru fasülye	31
Nohut	46
Referans kart	16

Tablodaki 1000 adet etiket toplam 906 adet görselden elde edilmiştir.

3.2 Yöntem

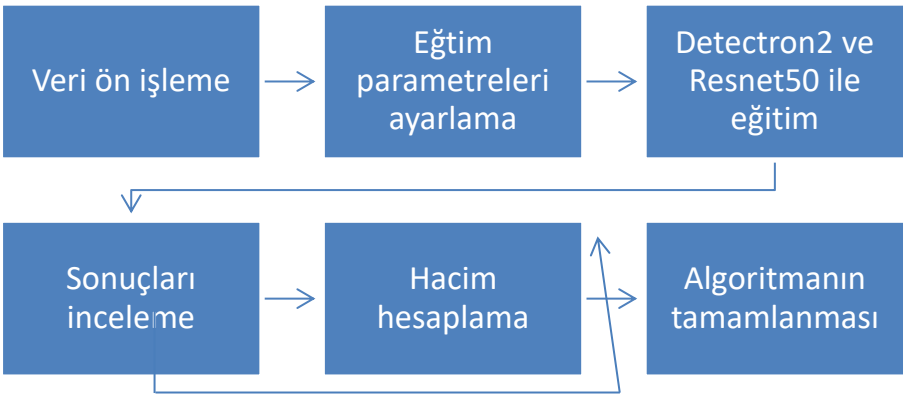
Şekil 3.5'te kabaca gösterilen akış şeması ile sistemin temel basamakları gösterilmektedir. İnternet üzerinden istenilen anahtar kelimeler kullanılarak indirilen görseller boyut sabitleme işlemi yapıldıktan sonra etiketleme yazılımı yardımı ile etiketlenir. Sistemimizde 17 sınıftan 906 görsel ile toplam 1000 adet etiket ile veri seti hazırlanmıştır. Bu bölümün diğer kısımlarında Detectron2 sistemi içinde eğitim algoritması olarak kullandığımız ResNet50 modelinin nasıl çalıştığını ardından eğittiğimiz model ile nasıl hacim hesaplaması yaptığımızı ve en sonunda nihai ürün olarak web aplikasyonu oluşturma kısmı anlatılacaktır.

3.2.1 Eğitim

Hızlı çalışan ve yüksek doğruluk ile çıktılar üreten sistem oluşturmak için birçok algoritma denenmiştir. Ancak nihai ürün üretme ve model çıktısını kolaylıkla diğer algoritmalarda da kullanabileceğimiz Detectron2 sistemi ile eğitimimizi gerçekleştirdik. 3.2.1.1'de bahsedeceğimiz Detectron2 sistemi bize nesne tanıma sistemleri oluştururken kolaylık sağlamaktadır. Bu kısımda ResNet50 modeli ve bu işlemleri yaparken bize GPU sağlayan Google Colab arayüzünün nasıl çalıştığı anlatılacaktır.

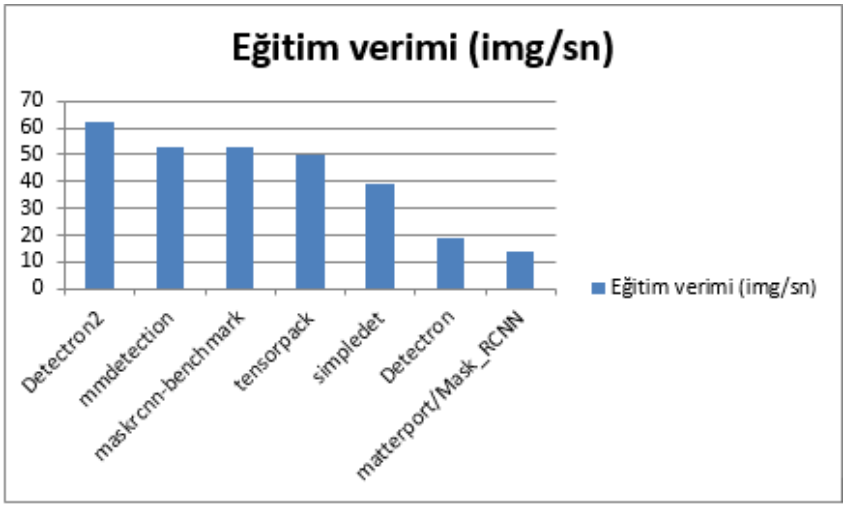
3.2.1.1 Detectron2 kullanımı

Facebook AI 2018 yılında Detectron isimli bir nesne algılama kütüphanesi geliştirdi [39]. Başarılı bir kütüphane olsa da kullanımı biraz zordu. Caffe2 tarafından destekleniyordu. Caffe2 ve Pytorch'u bir araya getiren birçok kod olsa da bu durum Detectron kullanımını güçleştiriyordu.



Şekil 3. 5 Sistem akış şeması

Bu durum üzerine Facebook AI Research grubu yeni bir versiyon geliştirdi. Kendi tanımlarıyla da Detectron2, Facebook AI Research'ın son teknoloji nesne algılama algoritmalarını uygulayan yeni nesil yazılım sistemidir. Bu versiyon ile yapay zeka algoritmalarının temelini oluşturan TensorFlow kütüphanesi ile nispeten daha çok rekabet edebilecek olan PyTorch ile geliştirildi. Kurulum talimatlarını kolay olan bu sistem puanlama sonuçlarını çıkarmak için çok kolay bir API'dir.



Şekil 3. 6 Nesne tanıma algoritmaları eğitim verimi karşılaştırması

Şekil 3.6'da gösterildiği üzere Detectron2 en güncel ve en hızlı sistemdir.

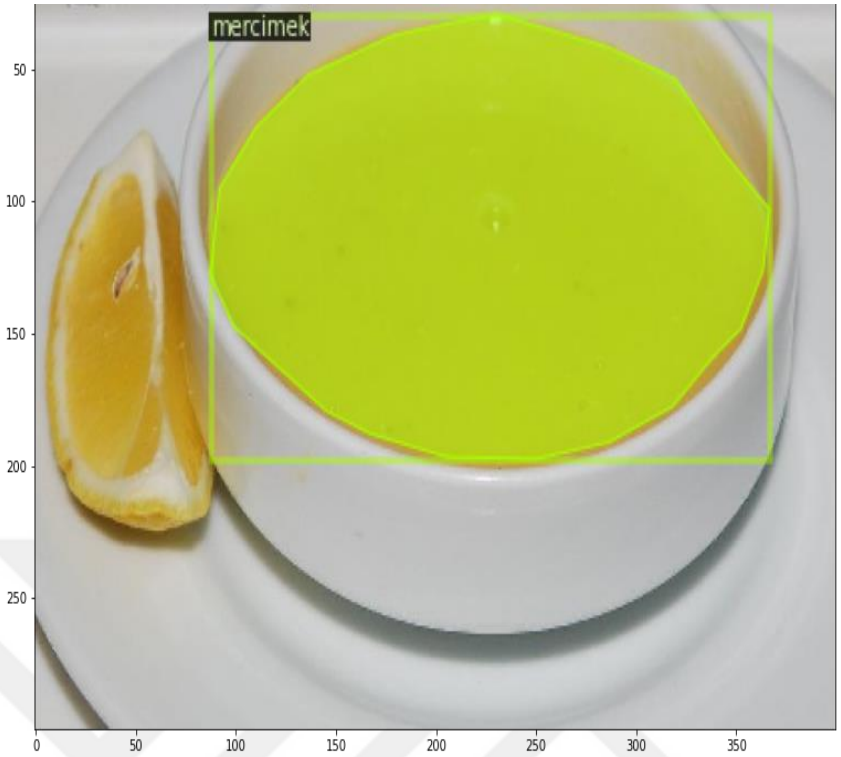
Detectron2'nin amacı nesne algılama araştırmacıları için yüksek kaliteli, yüksek performanslı bir kod tabanı oluşturmaktır. Yeni araştırmaların hızlı uygulanması ve değerlendirilmesini desteklemek için esnek olacak şekilde tasarlanmıştır. Detectron2 aşağıdaki nesne algılama algoritmalarının uygulamalarını içerir [39].

- Mask R-CNN
- RetinaNet
- Faster R-CNN
- RPN
- Fast R-CNN
- R-FCN

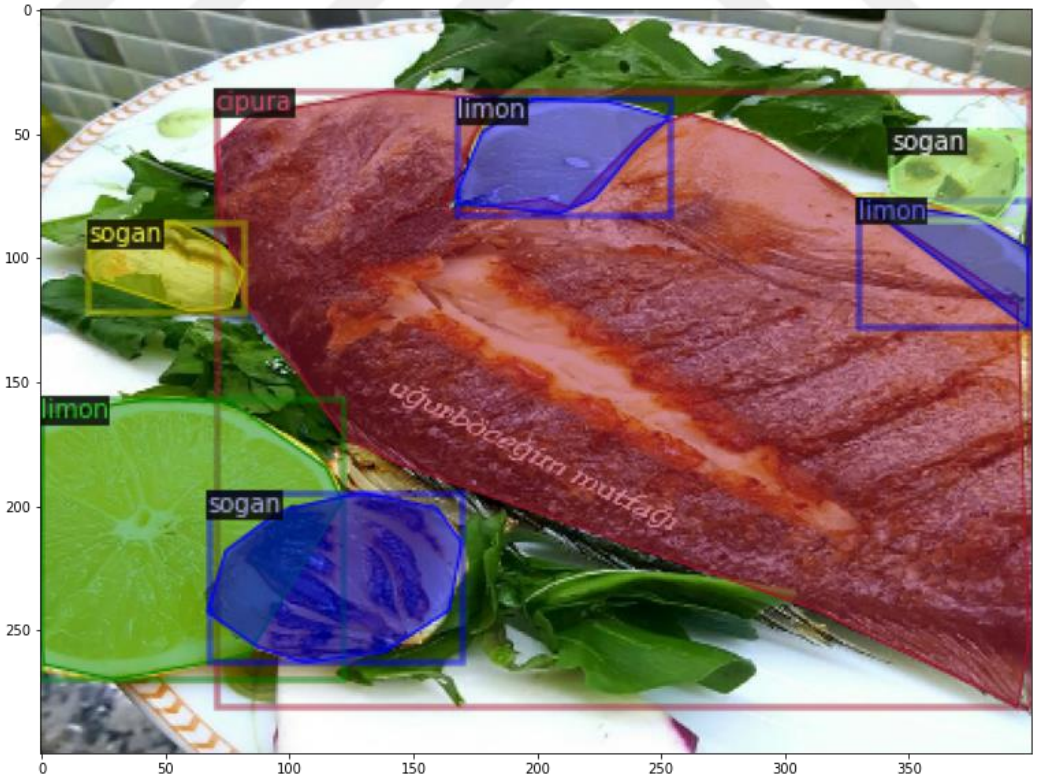
Bu işlemleri yaparken aşağıdaki omurga ağ mimarilerini kullanır.

- ResNeXt 50,101,152
- ResNet 50,101,152
- Özellik piramit ağlar (Feature Pyramid Networks) ile ResNet ve ResNeXt
- VGG16

Ayrıca ek omurga mimarilerini kolaylıkla uygulayabiliriz. [40]'ta verilen kod ile kolaylıkla örnek bir eğitimin nasıl gerçekleştiğini anlayabiliriz. Bunun için bölüm 3.2.1.3'de bahsedeceğimiz Goolge Colab arayüzünden yeni bir sayfa açtıktan sonra öncelikle kütüphane tanımlamalarımızı yapıyoruz. Sisteme Detectron2'yi yüklüyoruz. Örnekte gerçekleştirilen mevcut bir model kullanarak yeni bir veri setine sistemin uygulanmasıdır. Sisteme veri setimizi yüklüyoruz. Daha sonrasında eğitim için ayırdığımız dosya yolunu tanıtıyoruz. Sistemin tanıttığımız görselleri doğru algılayıp algılamadığını ölçmek için tanıtılan görselleri rastgele olarak etiketleri ile birlikte açıyoruz. Şekil 3.7'de kendi sistemimizden bir örnek bulunmaktadır. Sonraki kısımda veri setini önceden eğitilmiş R50-FPN Mask R-CNN modeline ince ayar yaparak tanıtıyoruz ve eğitimi başlatıyoruz. İnce ayar kısmında eğitime kör bir şekilde başlamak yerine önceden ImageNet görselleri kullanarak çok uzun süren ve yüksek kapasiteli bilgisayarlar ile eğitilmiş omurgaların ağırlıklarını kullanıyoruz.

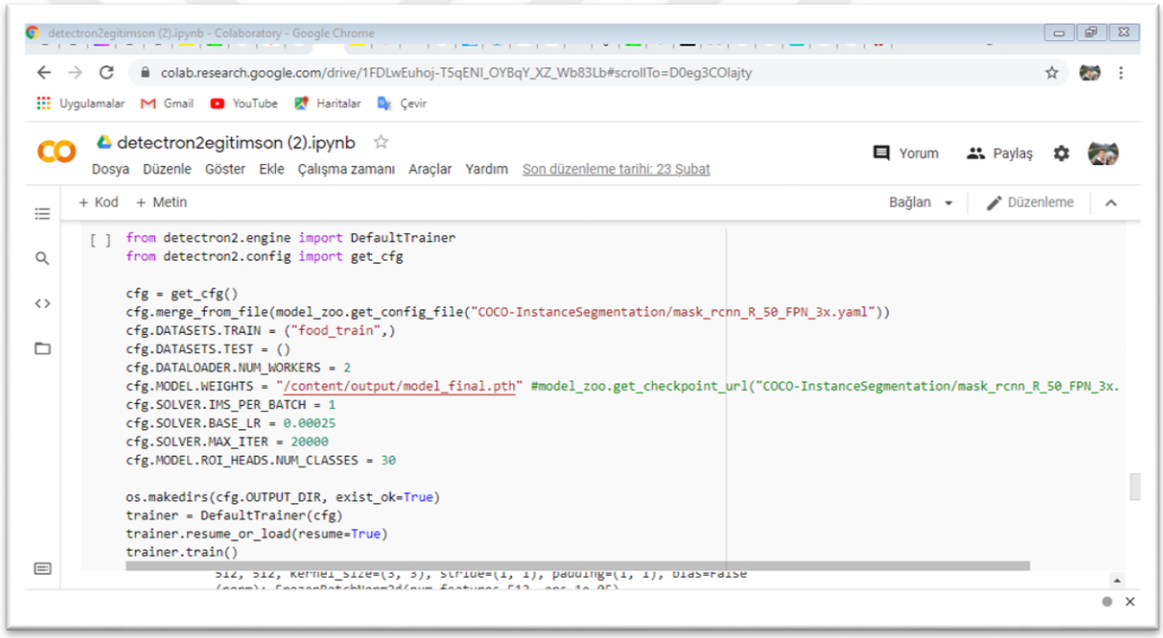


Şekil 3. 7 Kendi sistemimizden eğitim setinden bir görüntü



Şekil 3. 8 Eğitim setinin sisteme yüklenirken sistemin test görüntüsü

Tabi sıfırdan da başlayabilirdik. Sistem bu imkanı bize sunmaktadır ancak veri setimiz daha önceden görülmemiş bir veri olmadığı için örneğin karaciğerde bulunan kistlerin tespitini sağlayan bir uygulama olsaydı onun için sıfırdan başlayan bir eğitim gerçekleştirmemiz gerekirdi. Bizim sistemimiz yemek görselleri içerdiği için önceden eğitilmiş ağırlıklar ile başlamak bize eğitim süresi ve eğitim başarısı için daha uygun bir başlangıç olacaktır. Sistemde eğitim adım sayısı, her adımda kaç görsel kullanacağı, toplamda kaç sınıfı kullanılacağı ve iterasyon sayısı gibi parametreleri giriyoruz. Ve eğitimi başlatıyoruz. Sistem çıktı olarak öncelikle mimari yapıyı bizlere sunmakta ardından veri setinin özelliklerini listelemekte ve eğitime başlamaktadır.



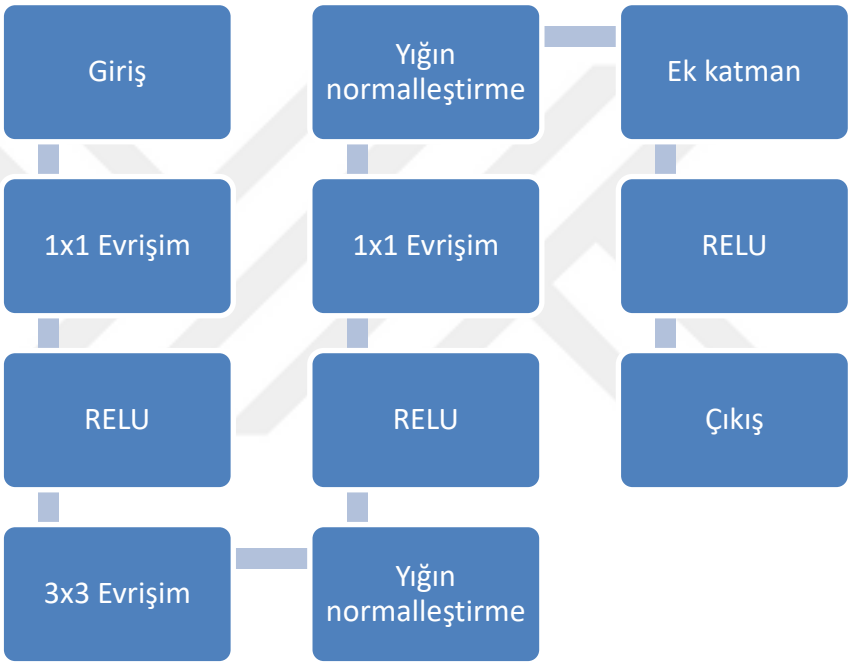
Şekil 3. 9 Detectron2 eğitim konfigürasyonları

Eğitim sırasında tahmini kalan süre, toplam kayıp, sınıf kaybı, maske kaybı ve kutu kaybı gibi verileri bizlere sunmaktadır. Eğitim tamamlandığında modeli kaydetmektedir. Ardından yazdığımız kod ile modelin daha önceden görmediği görseller ile nasıl başarı göstereceğini test edebiliriz.

3.2.1.2 ResNet50

Bu çalışmada açık kaynak CNN tabanlı ResNet-50 mimarisi sistem omurgası

olarak kullanılmıştır. ResNet mimarisi 2015'deki yarışmalarda birinci olmuş ve farklı veri setleri ile kullanımı oldukça kolaydır. ResNet-50 mimarisi 50 katmandan oluşmakta ve beş evrişimsel blok kullanılmıştır. Bunlar 1x1, 3x3 ve 1x1 evrişimsel katmanlarından meydana gelmektedir. 1x1 evrişimleri ile giriş görselleri daha küçük ölçülere indirilmiş ve 3x3 evrişimleri ile daha büyük ölçülerde filtreleme işlemi gerçekleştirilmiştir. Boyut küçültme amacıyla pooling (havuz) katmanı kullanılmıştır. Mimarinin tam bağlı katmanından Softmax aktivasyon fonksiyonu kullanılmıştır. Görsellerin sınıflandırılması için 1000 kategorilik bir çıkış verilmiştir. Şekil 3.8 de ResNet katmanları gösterilmiştir.



Şekil 3. 10 ResNet katmanı

3.2.1.2.1 Evrişim işleminin matematiksel açıklaması

Ortak katman davranışına girmeden önce ResNet'teki ilk adım, evrişim+ toplu normalleştirme + maksimum ortaklama ileminden oluşan bloktur. Bu bloğa evrişim 1 denir.

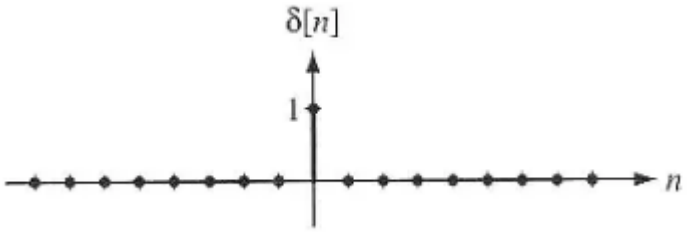
Evrişim ya da evrişim bir fonksiyonun şeklinin başka fonksiyon tarafından nasıl değiştirildiğini gösteren bir integral işlemidir. formülü aşağıda verilmiştir.

$$(f * g)(t) \triangleq \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau \tag{3.1}$$

Bu formülte f(t) girdi, g(t) ise dürtü yanıtı * işareti ise evrişim işlemini belirtmektedir.

Dürtü sinyali

Sinyal işlemede, dinamik sistemin dürtü yanıtı veya dürtü yanıt işlevi (IRF), kısa bir giriş sinyali (darbe olarak adlandırılır) olduğunda dinamik sistemin çıktısıdır. Daha genel olarak, dürtü tepkisi, herhangi bir dinamik sistemin bazı dış değişikliklere tepkisidir. Her iki durumda da, dürtü tepkisi sistemin tepkisini zamanın bir fonksiyonu (veya belki de sistemin dinamik davranışını parametrelendiren diğer bazı bağımsız değişkenlerin bir fonksiyonu) olarak tanımlar [41].



Şekil 3. 11 Dürtü sinyali

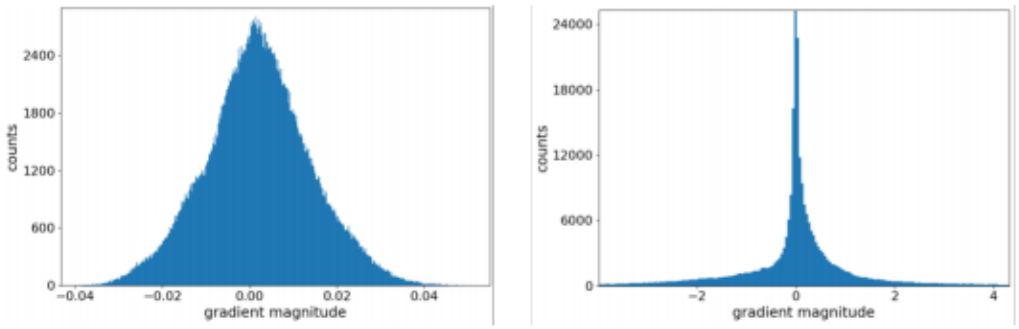
3.2.1.2.2 Batch Normalization (Yığın Normalleştirme)

2015 yılında [42] Batch Normalization ismli iki Google çalışanı tarafından yayınlandı o zamandan bu zamana 27 binden fazda atıf almıştır. Yığın normalleştirme methodu derin sinir ağlarındaki herhangi bir katmana 0'a ortalı ve 0 ile 1 arasında değerlere sahip veriler vermemizi sağlar [43].

- Mini yığındaki verilerin hepsi toplandıktan sonra toplam veri sayısına bölünerek ortalama değer bulunuyor.
- Mini yığındaki her veriden ortalama değer çıkarılıp karesi alınıyor ve hepsi toplanır. Ardından tekrar toplam girdi sayısına bölünür. Bu işleme min yığının varyansı denilir. Bir diğer deyişle verilerin hangi aralıklarda bulunduğunun bir ölçütüdür.
- Veriler normalize edilir.
- Normalize edilmiş veriler gamma ve beta parametreleri ile çarpılır.

Elimize geçen öğrenilebilir lineer denklem oluyor. Gamma ve Beta parametreleri

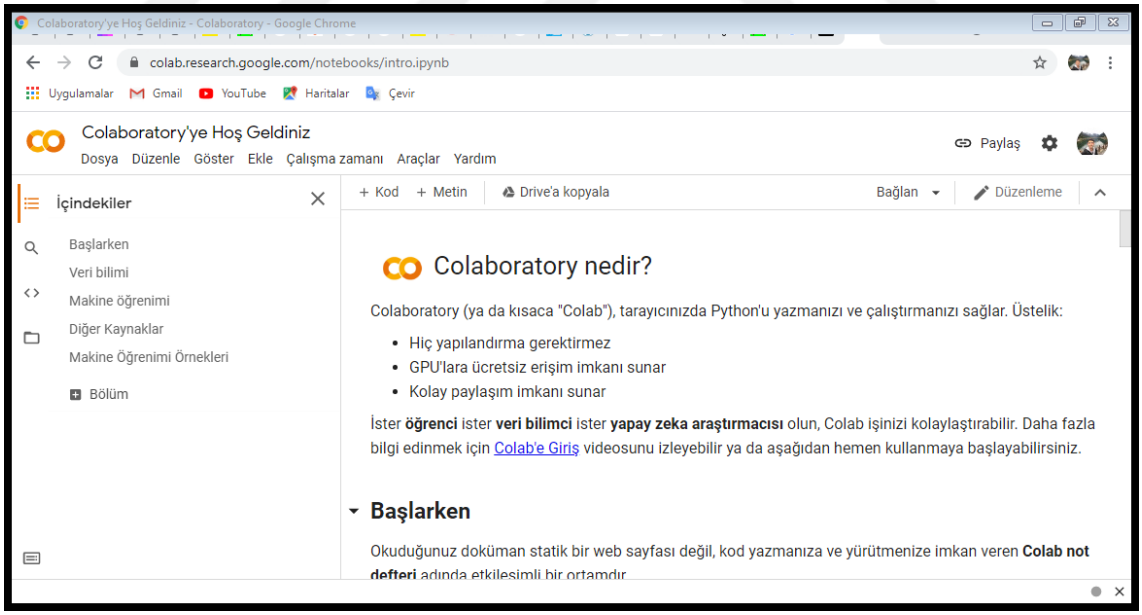
yığın normalleştirme katmanının veri üzerindek işlem yapma kapasitesini kontrol ediyor. Kısaca özetlersek bu katman eğitim verisi üzerinde ne kadar oynama yapacağına kendisi karar veriyor [43].



Şekil 3. 12 Yığın normalleştirme olmadan ve olduktan sonraki görüntü

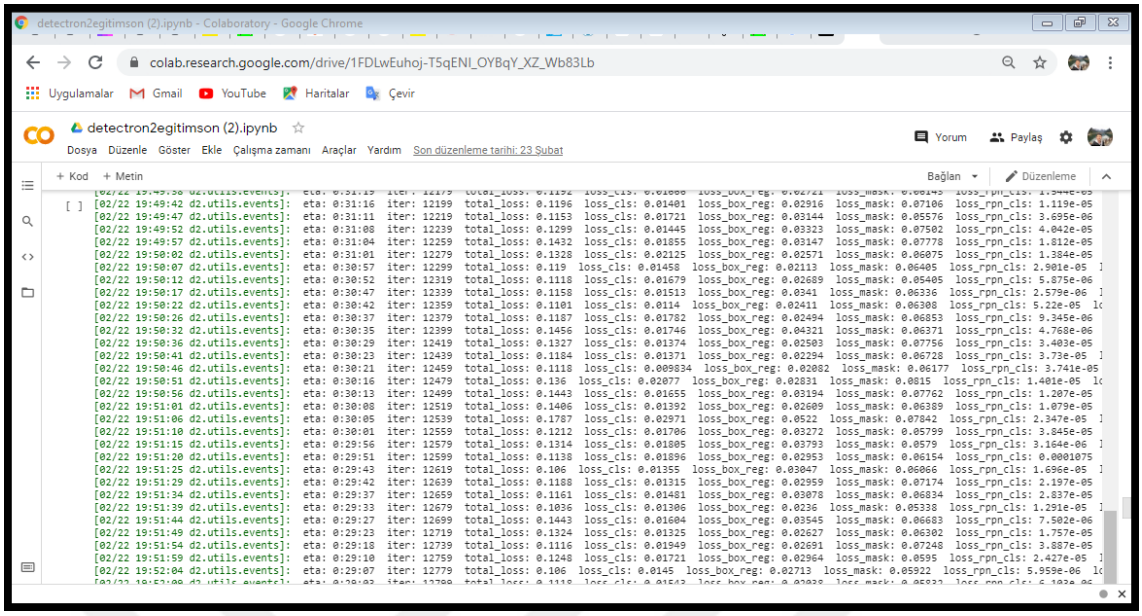
3.2.1.3 Google Colaboratory

Bu sistem Google'ın ücretsiz sunduğu tarayıcımız üzerinde Python'u yazmamızı sağlayan bir uygulamadır. Bizlere sunduğu Tesla K80 GPU üzerinde Keras, TensorFlow ve Pytorch kullanarak derin öğrenme uygulamaları geliştirebiliriz.



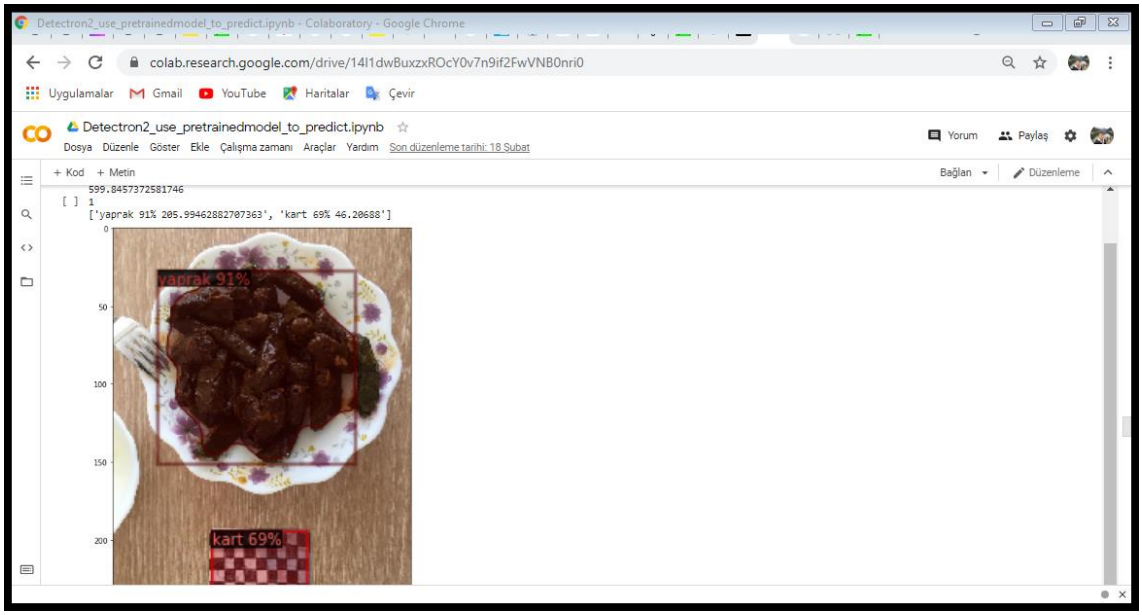
Şekil 3. 13 Google Colab Arayüzü

Şekil 3.10'da Google Colab'a ait giriş arayüzünü görmektesiniz şekil 3.11'de ise kendi eğitimizi yaparkenki ekran görüntüsünü görmektesiniz.



Şekil 3. 14 Eğitim sırasındaki Google Colab görüntüsü

Eğitim aşamasında ve daha sonraki test aşamalarının hepsinde Google Colab kullandık. Bu bize GPU sayesinde zaman kazandırdı. Eğitim bittikten sonraki son programımızı ise CPU ile çalışabilecek şekilde yazdık. Şekil 3.13'de eğitimden sonra gerçekleştirdiğimiz ve modeli test ettiğimiz ekranın görüntüsü bulunmaktadır.



Şekil 3. 15 Eğitimden sonra gerçekleştirilen test görüntüsü

3.2.2 Hacim tahmini

Eğittiğimiz modelleri yerel bilgisayarımıza kaydederek tarayıcı üzerindeki işlemlerimizi bitirdik. Bu aşamadan sonraki tüm işlemleri yerel bilgisayarımızda bulunan Spyder editörü üzerinden gerçekleştirdik. Hacim tahmini işlemi için birçok farklı yöntem bulunmaktadır. Bu yöntemlerden bazıları, 3 boyutlu görsel oluşturarak hesaplama, derinlik ölçümü yapan RGBD kameralar ile çekilen görüntüler ile hesaplama veya bizim kullandığımız hemen hemen her kamera ile elde edilebilecek görsel ile sorunsuz çalışacak olan referans obje yardımı ile öncelikle alan hesabı ardından daha önceden oluşturulan tablo ile hacim tahmini yapmak. Tabi bu işlemlerden en yüksek başarıda sonuç verecek olan yöntem RGBD kamera ile derinlik bilgisi yardımıyla hacim hesaplama işlemi ancak bu işlem için özel bir kamera gerekmektedir. Bundan sonraki yöntem ise birçok açıdan çekilen görüntüler yardımı ve bazı algoritmaların kullanılmasıyla oluşturulacak olan 3 boyutlu görsel üzerinden hacim tahmini yapmak. Bu işlem için yine ihtiyacımız olan bir adet referans nesnedir. Çoklu görsellerden yemek analizi yapan uygulamalar mevcuttur. Bizim sistemimizde ise bir adet tepen çekilmiş görsel yardımı ile ölçülerini daha önceden bildiğimiz referans nesnenin ölçülerinden faydalanarak yemeğimizin alanını ölçebiliyoruz. Bu işlemden sonra daha önceden değişik boyutlardaki çorba kaseleri ve yemek tabaklarının derinlik, çap ve hacim bilgilerini ölçtük. Aşağıdaki tablo 3.2'de verdik.

Tablo 3. 2 2 Ölçülen tabakların ölçüleri

Üst çap (cm)	**Alt çap (cm)**	**Derinlik (cm)**
21	12	3,5
16	7	5
20	11	4,5
20	12	3,8
16	7,5	5,5
21	10	4
20	11,5	4,8
12	7,5	5,2

20	10	4,5
20	12	3,8
19	10	3,5
11	6,5	3,5
20	12	4

$$r1 = \frac{cap}{2}$$

$$r2 = \frac{10}{2}$$

$$h = |2 * (r1 - r2)|$$

$$V = ((pi) * (h) * ((3) * (r12) + (3) * (r22) + (h2)))/6 \quad (3.2)$$

3.2'deki formülden tabakların hacimlerini hesapladık ancak bu hesaplamalar gerçek ölçümlerimiz ile tam olarak eşleşmedi bunun sebebi ise tabakların değişen daralma açıları ve kalınlıkları. Bunun için ölçümlerimiz ile hesaplamalar arasındaki farkı bazı katsayılar ile çarparak hata oranını düşürmeyi hedefledik.

Formülde "cap" modelimizin tahmin ettiği alanı dairenin alanı kabul ederek dairenin alanı formülünden yararlanarak çap bilgisi elde ediyoruz. Formüldeki "h" değerinin aralığına göre değişim gösterdiği bir if döngüsü ekledik kodumuza. Mercimek çorbası için farklı bir formül kullandık o formülde ise yarıçap olarak görüntüden elde ettiğimiz yarıçap ölçüsündeki yarım kürenin hacminden aynı şekilde "if" komutu ile yazdığımız kod ile bir alt yarıçap belirliyoruz, o alt yarıçapa sahip diğer yarım küreyi çıkararak kase şekli elde ediyoruz ve hacim bilgisi çıkarıyoruz. Diğer yandan sistemimizde tanımladığımız yemeklerimizin gerçekte kapladıkları hacimleri ve o hacimlere düşen ağırlıklarını ölçtük. Şekil 3.14-17 Bunu yaparken yemekleri 200 ml hacme sahip olan su bardaklarına doldurduk ve daha önceden darası alınmış hassas terazide tartarak 200 ml hacme denk gelen ağırlığı ölçtük. Yemeklerimizin bazılarını hazır olarak satın alırken bazılarını kendimiz hazırladık. Kendimiz hazırladığımız yemeklerin kalori bilgilerini internetteki farklı kaynaklardan doğrulayarak elde ettik. Hazır olarak kullandığımız yemeklerin ise paketleri üzerinde yazılı olan besin değerlerini esas aldık ve bir tablo oluşturduk. Tablo

3.3’de sulu yemekler için gr/ml üzerinden hesaplama yapılırken baklava ve trileçe için gr/cm^2 üzerinden hesaplama yapılıyor. Yazılımımız referans obje yardımı ile elde ettiği yüzey alanından, tabak yemekleri için yemeğin çapını tatlılar için üst yüzey alanını hesaplıyor ve tabak yemeklerinin tabak hacmini hesaplıyor. Hacmi hesaplarken pilav, et sote, tavuk sote ve yaprak gibi yemekler tabağa tepeleme doldurulabileceğinden kuşbakışı görüntüden bunu bizim modelimizin anlaması mümkün değildi. Bu yüzden yazılımı çalıştırırken bu yemekler için kullanıcıdan tercih yapmasını istiyoruz yemeğin üzerinin düz mü yoksa tepeleme mi olduğunu seçmesini ve ona göre biri diğerinin 1,3 katı olacak şekilde yazılımın hesaplama yapmasını sağlıyoruz.

Şekil 3. 16 200ml mercimek çorbası ağırlığı

Şekil 3. 17 200ml taze fasülye ağırlığı

Şekil 3. 18 200ml pirinç pilavı ağırlığı

Şekil 3. 19 200ml barbunya pilaki ağırlığı

Yazılımımızda mercimek çorbası için programın tahmin ettiği çapın belirli bir ölçüye kadar kase üzerinden hesaplama yapmasını daha üzeri ölçülerde tabak üzerinden hesaplama yapmasını sağladık. Kurduğumuz sistemin kullanıcı tarafından kolaylıkla kullanılabilmesi için Streamlit python paketi ile yazılımımız için kullanıcı arayüzü oluşturduk. Bu arayüz sayesinde herhangi bir internet sitesi üzerinden hem mobil görünümlü hem web görünümlü olarak uzak bilgisayarda çalışan bir sistem oluşturmuş olduk. Kullanıcı bu arayüze erişim sağlayarak ister yemeğin görüntüsünün web adresini ister yerel sisteminden yüklediği fotoğrafı sistemde çalıştırıp sonuç alabiliyor.

Tablo 3. 3 Yemek Ölçüleri tablosu

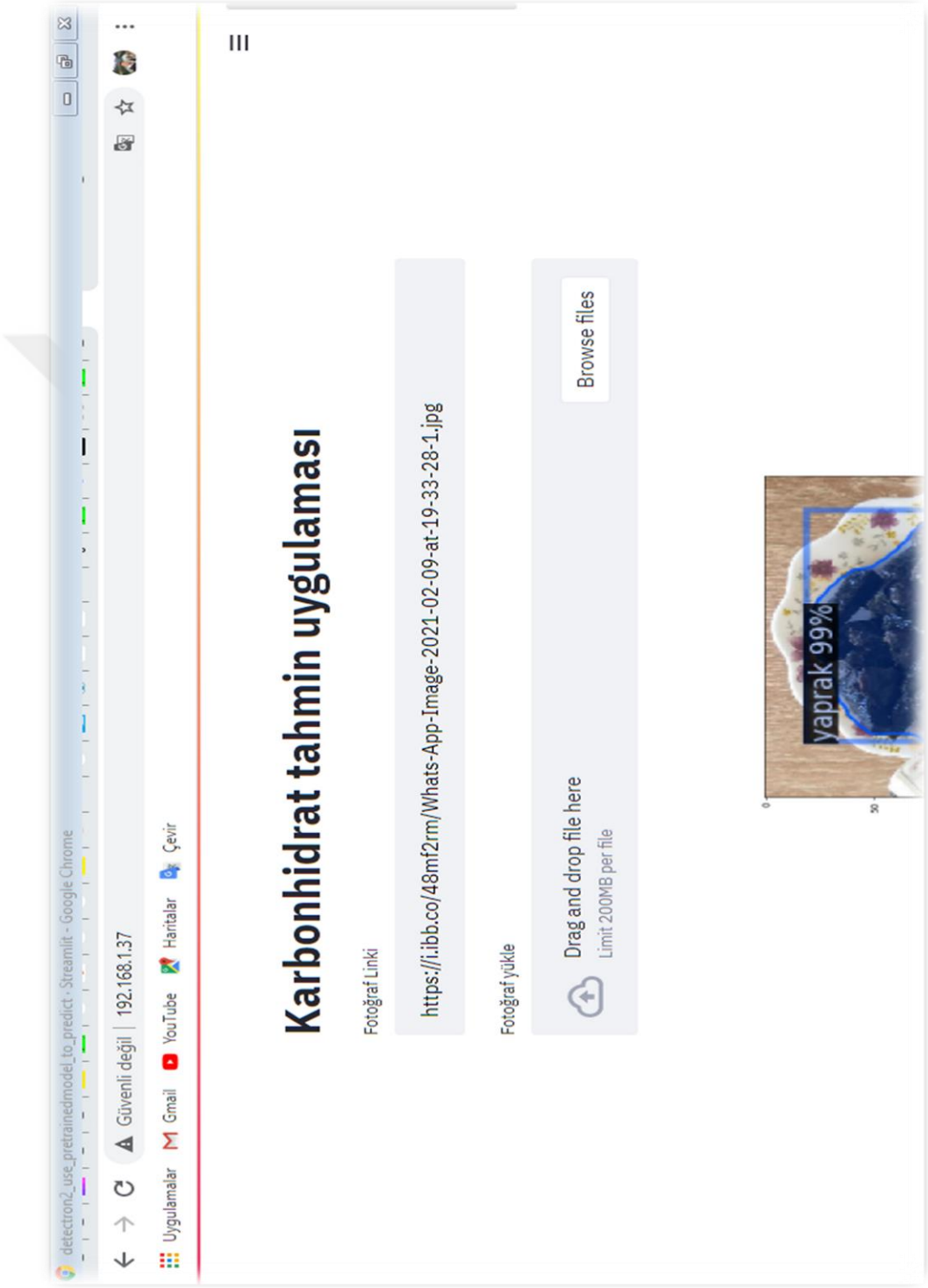
	Gr	Ml	Karbonhidrat(100g)	Karbonhidrat(100 ml)	Gr/ml	100(gr) kalori	Yüzey alanı cm^2	gram/yüzelalanı
mercimek	227	200	8,28	9,3978	1,135	55,9	0	
barbunya	226	200	24,46	27,6398	1,13	257	0	

cipura	200	100	0	0	2	96	0	
baklava	41	21	37,3	72,8238095238094	1,95238095238095	285	25	1,64
fasulye	205	200	7,6	7,79	1,025	85	0	
etsote	206	200	3,15	3,2445	1,03	107	0	
pilav	153	200	18,6	14,229	0,765	118	0	
karniyarik	390	300	5,33	6,929	1,3	52,84	0	
manti	10	20	29,71	14,855	0,5	170	0	
menemen	106	114	3,39	3,1521052631579	0,929824561403509	71	0	
nohut	376	385	20,59	20,1086753246753	0,976623376623377	164	0	
pirinc	153	200	28,59	21,87135	0,765	130	0	
tavuksote	159	200	4,51	3,58545	0,795	169	0	
trilece	200	243	21,05	17,3251028806584	0,823045267489712	163	64	3,125
yaprak	411	360	21,36	24,3860000000001	1,14166666666667	165,53	0	
kurufasulye	376	385	25,09	24,5034805194805	0,976623376623377	139		

3.2.3 Web uygulaması oluşturma

Streamlit hazırladığımız tüm sistem kodlarını değiştirmeden sadece bazı kullanıcı arayüzü ayarlamaları yaparak kullanmamızı sağlayan ve web üzerinden sistemimizin çalışmasına altyapı oluşturan kütüphanedir. Bu kütüphaneyi kullanarak [46] numaralı kaynakta bulunan EK 5'te ki kodlarımızı python editörü ile çalıştırdığımızda web uygulamamız aktif hale gelmektedir. Web uygulamasını üzerinde ister web üzerinden görüntünün adresini yazara ister yerel bilgisayardan yükleyerek sistemi çalıştırabiliriz. Sistem CPU üzerinde çalışmaktadır. Şekil 3.18 ve şekil 3.19 da bilgisayar üzerinden

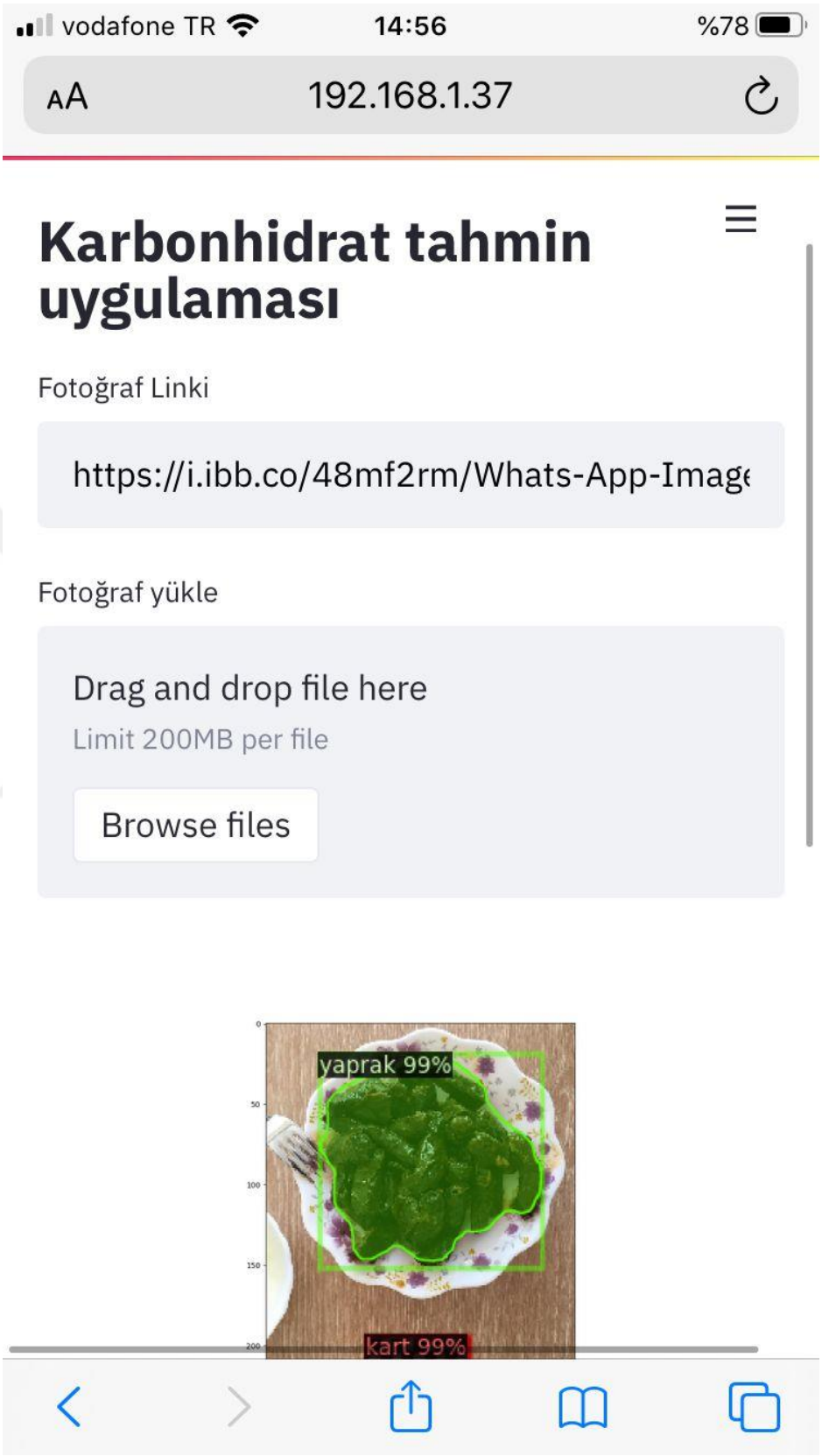
bağlandığımızda karşımıza çıkan görüntü bulunmaktadır. Şekil 3.20 ve şekil 3.21'de ise mobil arayüz gözükmektedir.



Şekil 3. 20 Web uygulaması ekran görüntüsü 1

Şekil 3. 21 Web uygulaması ekran görüntüsü 2

Şekil 3. 22 Uygulamanın mobil arayüzü 1



Şekil 3. 23 Uygulamanın mobil arayüzü 2

4. BÖLÜM

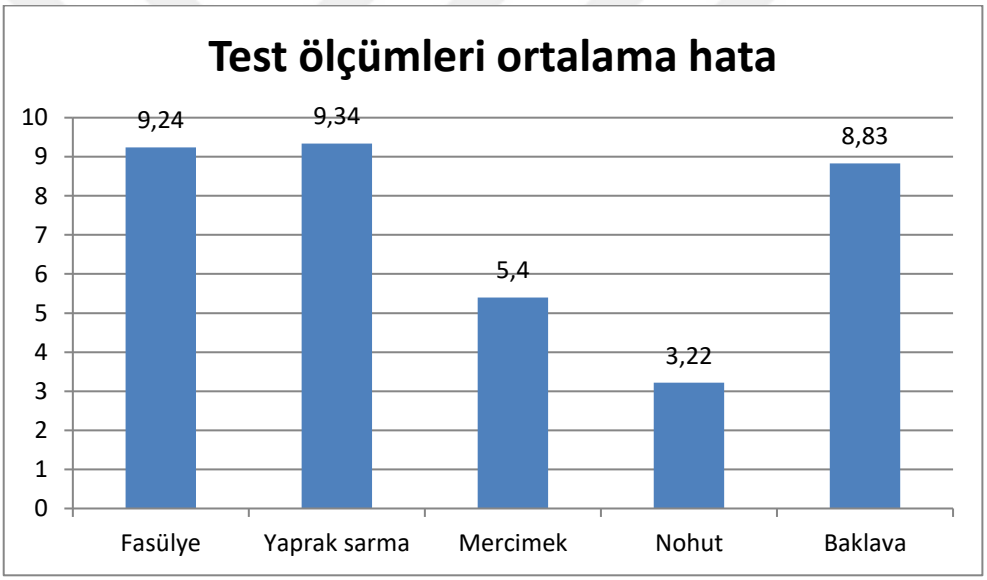
BULGULAR

Kurduğumuz arayüzün kullanıcıdan görüntü alabilmesi için gerekli kodlar yazılmıştır. Kullancı daha öncedende bahsettiğimiz farklı yollar ile görüntüsünü sisteme yükleyebilmektedir. Görüntü yüklenildikten sonra sistem baştan çalışarak girdi olarak kullanıcının yüklediği görseli çalıştırmakta ve görsel üzerinde referans obje ile birlikte yemeği tanımaya ve yemeğin alanını maskelemeye çalışmaktadır. Bulduğu alanlar ile metot kısmında bahsettiğimiz yöntem ile birlikte gerekli hesaplamaları yaparak yemek üzerinde hesapladığı karbonhidrat miktarını kullanıcıya sunmaktadır. Yemek görselleri yüklenmeden önce yemekler hassas mutfak terazisi ile tartılarak ağırlıkları kaydedilmiştir. Sistemimizin bulduğu sonuçlar ile gerçek sonuçlar ve hata oranları tablo 3.4'de verilmiştir. Sistemimiz tek bir görselden hacim hesaplaması yaptığından çıkan sonuçların yüksek başarıda olduğunu söyleyebiliriz. Daha başarılı sistemlerde genellikle birkaç farklı açıdan elde edilen görüntüler ile yemeğin 3 boyutlu görüntüsü elde edilerek hacim hesaplaması yapılmakta ve dolayısıyla daha başarılı sonuçlar elde edilmektedir. Ancak kullanıcının bir yemeğin birkaç farklı açıdan fotoğrafını yüklemesi çokta sürdürülebilir bir şey olamayabilir. Bu açıdan bizim sistemimiz genellikle çok kullanılan tabak yemeklerinde yüksek doğruluk oranları elde edebilmektedir.

Tablo 4. 1 Gerçek değerler ile tahmin edilen değerler ve hata yüzdeleri

Yemek ismi	Gerçek Hacim	Tahmin Edilen Hacim	Hata Yüzdesi	Gerçek Ağırlık	Tahmin Edilen Ağırlık	Hata Yüzdesi
Bulgur Pilavı	342	363,89	6,40058479532164			
Et Sote	200	206,66	3,33			
Baklava				41	46,90	14,390243902439
Baklava				66	70,44	6,72727272727273
Baklava				82	77,58	5,39024390243902
Menemen				114	102,47	10,1140350877193
Nohut	311	310,95	0,0160771704180064			
Nohut	385	354,54	7,91168831168831			

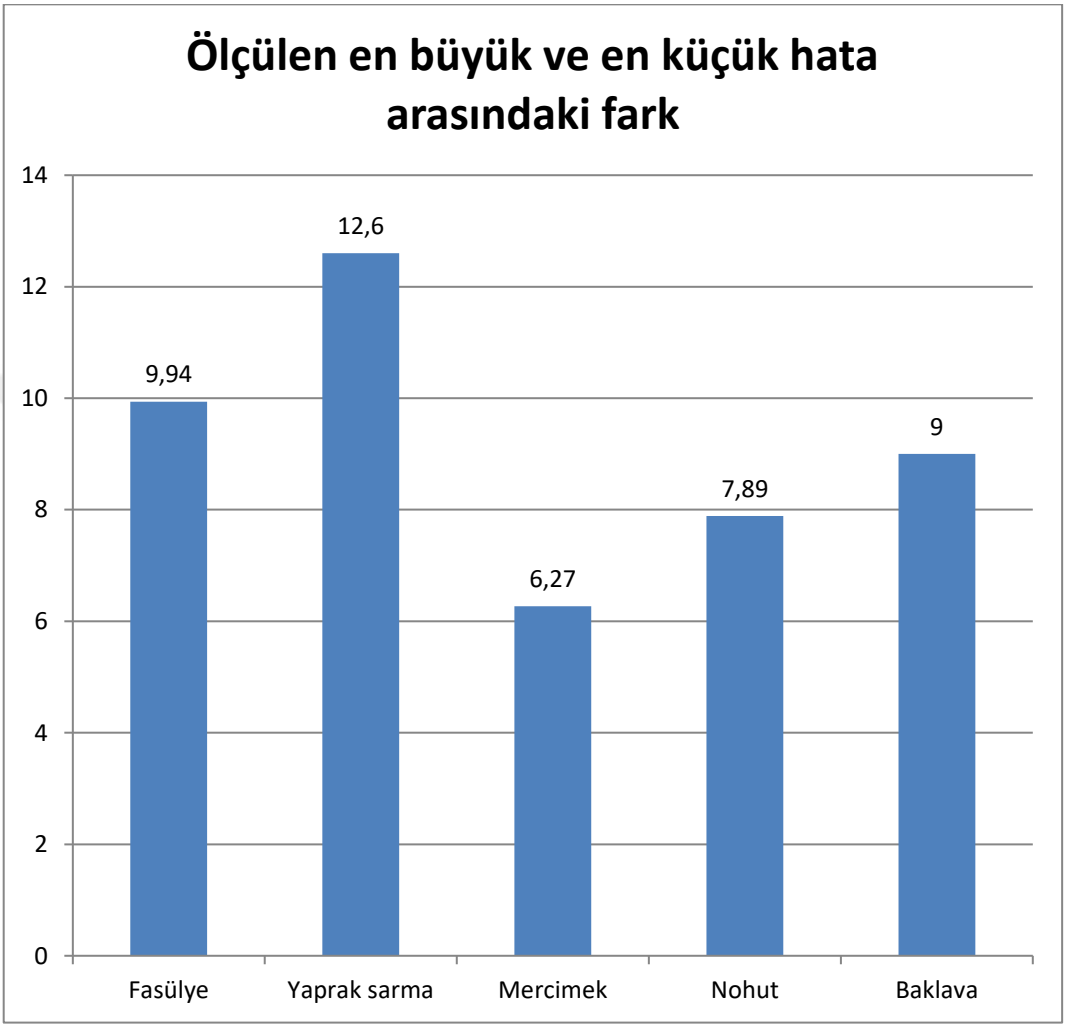
Nohut	385	378,27	1,74805194805195			
Pirinç Pilavı				345	342,1	0,840579710144928
Trileçe				400	428,59	7,1475
Mercimek	220	239,24	8,74545454545455			
Mercimek	220	230,98	4,99090909090909			
Mercimek	350	341,36	2,46857142857143			
Yaprak Sarma				274	316,88	15,6496350364964
Yaprak Sarma	385	396,72	3,04415584415584			
Taze Fasülye				122	128,04	4,95081967213115
Taze Fasülye				242	222,9	7,89256198347107
Taze Fasülye				242	205,95	14,896694214876
Tavuk Sote				276	236,77	14,213768115942



Şekil 4. 1 Test ölçümleri ortalama hata

Şekil 4.1'de gözüktüğü üzere Tablo 3.4'deki veriler ışığında 5 adet sınıfa ait ölçümler sonucunda çıkan ortalama hataları görmekteyiz. Bu sonuçlardan kolaylıkla anlayabiliriz ki katı şekilleri olan tabağın şeklini almayan yiyeceklerin hata oranları sulu yemeklere göre daha fazladır. Ancak her halükarda ortalama hata seviyeleri %10'un altında kalmaktadır. Nohut ve mercimek arasındaki fark ise birisinin kase üzerinde servis edilmesi diğerinin tabak üzerinde servis edilmesinden kaynaknaklanmaktadır. Kaseler tabaklara göre daha çeşitli alternatif ve derinliklere sahip olabilmektedir. Sistemimiz için ortalama bir hesaplama yapılmaktadır. Sistem görüntüdeki çapı tahmin ederek o çaptaki

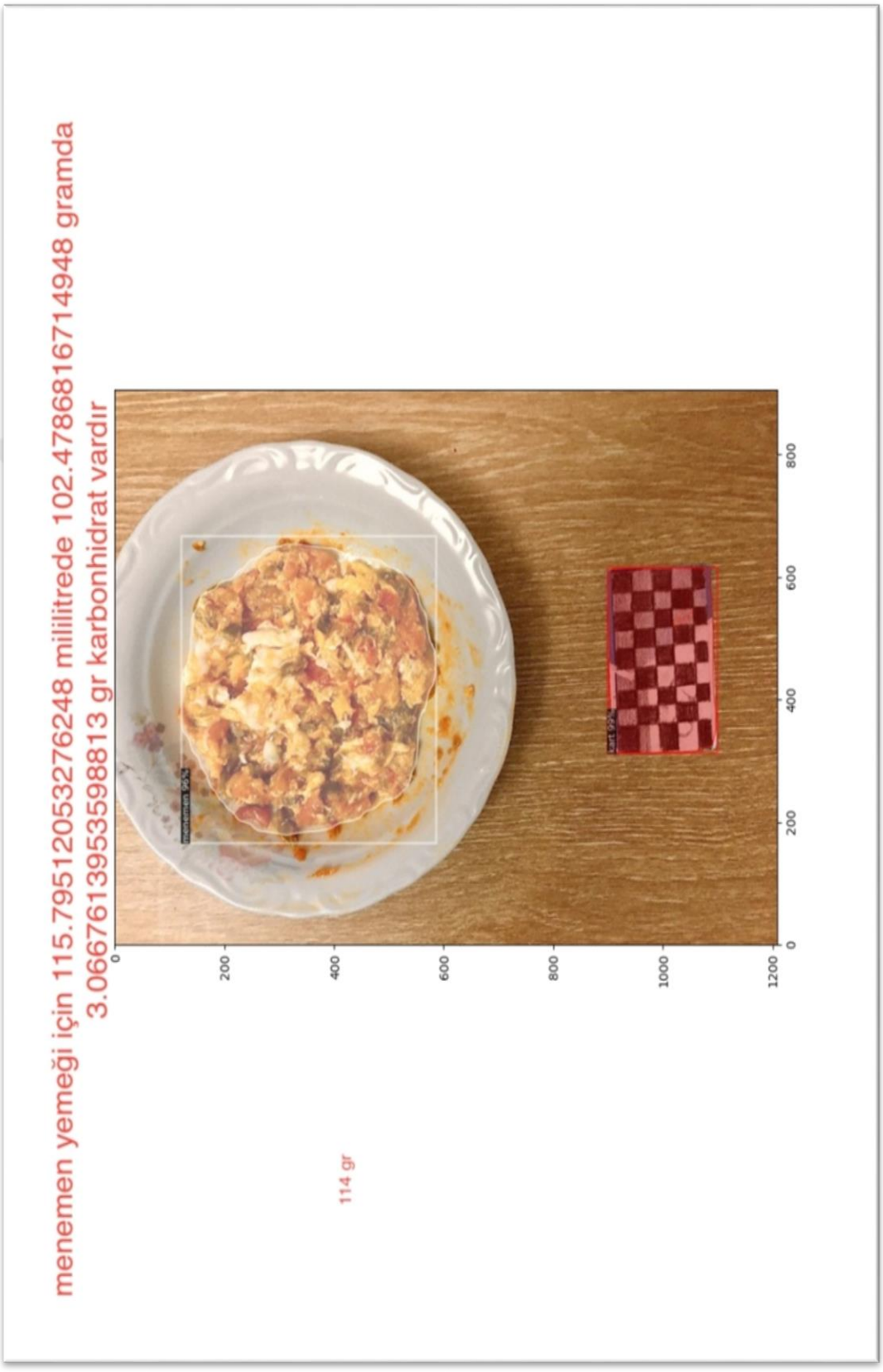
çorbanın kaç cm derinliğe sahip olduğunu kendi formülü ile hesaplamaktadır. Bu formül daha önce yapılan ölçümler neticesinde ortalama bir değer verecek şekilde ayarlanmıştır.



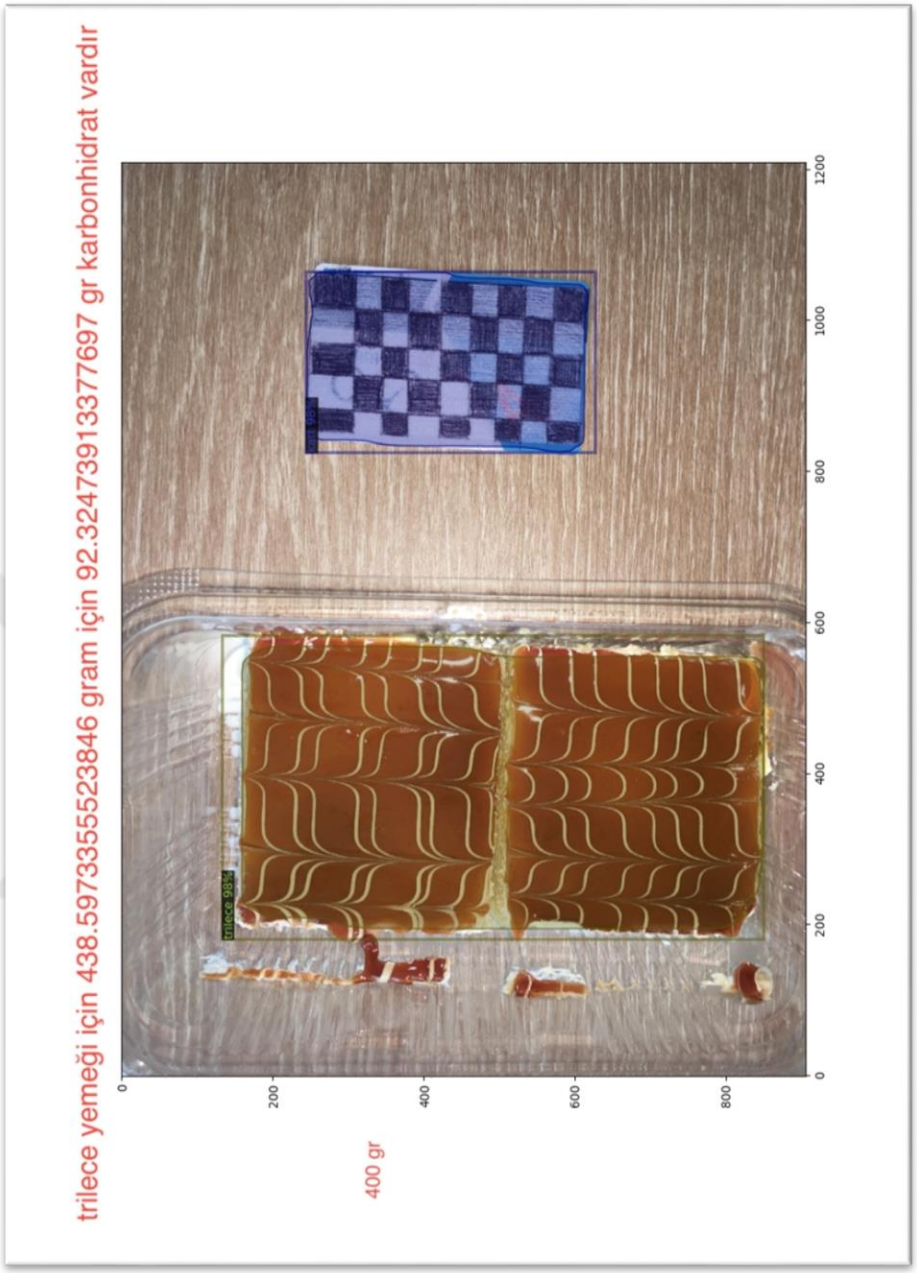
Şekil 4. 2 Ölçülen en büyük ve en küçük hata arasındaki fark

Şekil 4.2'de gördüğümüz ve Şekil 4.1'de bahsettiğimiz gibi sulu yemeklerde hata aralığı daha dar nispeten daha katı şekillere sahip yemeklerde daha fazladır. Sistemimizde en çok hatayı verebilecek olan yemek yaprak sarmadır. Yaprak sarma yemeği'nin servisi çok farklı şekiller alabildiğinden bu yöntem ile en doğru sonucu almak mümkün olmamaktadır. Önceden de bahsettiğimiz gibi sulu yemekler en doğru sonucu verirken tatlılarda ki hesaplama yöntemi farklıdır. Tatlıların üst yüzey alanı ile doğrudan gramaj ilişkisi bulunmaktadır. Ortalama bir baklavanın kalınlığı bilindiğinden direkt olarak yüzey alanı ile ilişkilendirebiliyoruz.

Şekil 4. 3 Tartılan ağırlığının 82 gr olan baklavanın tahmin edilen ağırlığı



Şekil 4. 4 Gerçek ağırlığı 114 gram olan menemen tahmini



Şekil 4. 5 Gerçek ağırlığı 400gr olan tatlı tahmini

Şekil 3.22-24 arası görseller gerçekte ağırlıkları bilinen yemeklerin uygulamadaki sonuçları gösterilmektedir. Uygulama tek görsel ile çalışmakta olduğundan yemeklerin tabağa yayılmış durumda mı yoksa tepeleme mi doldurulmuş olduğunu anlayamamaktadır. Bunun için böyle olabilecek yemeklerin görseli yüklendiğinde kullanıcıya bir soru sorulmakta ve seçim yapması istenmektedir. Bulgular kısmında bu işlem sonradan hata oranları düşürülmesi için eklenmiştir.

5. BÖLÜM

SONUÇLAR

T1DM'li bireylerin karbonhidrat sayımlarını düzenli olarak ve hızlı bir şekilde yüksek doğrulukta yapabilmeleri büyük önem arz etmektedir. Genellikle bu işlerini ya internet üzerinde porsiyon bilgileri girilerek elde edilen karbonhidrat miktarları ile ya da yemeklerini kendileri hazırlarken ürünleri tartarak ölçülü bir şekilde kullanarak hesaplama yapmaktadırlar. Bu işlem kendilerini yormaktadır. Bilgisayarlı görü yöntemleri sayesinde T1DM'li bireylerin yediği yemekler üzerinde daha az hesaplama yapmasını ve daha kaliteli vakitler geçirmesi hedeflenmektedir. Bu konuda çok fazla çalışma yapılmaktadır.

Önceki çalışmalarda kullanılan yöntemlerden farklı olarak sistemimiz Detectron2 ile geliştirilmiştir. Detectron2'nin hızı farklı çalışmalar ile kanıtlanmıştır. Bu açıdan bu alanda yapılan araştırmalardan Detectron2 kullanılımına karar verilmiştir. Bir diğer yönden elde ettiğimiz veri setini kendimiz internet üzerinden aldığımız görseller ile oluşturduğumuzdan ötürü çok fazla sayıda eğitim verisi elimizde bulunmamaktadır. Buna rağmen elde ettiğimiz verilerde gördüğümüz, sistemimiz yemekleri yüksek başarı oranları ile tanımakta ve maskelemektedir. Ayrıca veri setimizdeki yemeklerin görüntüleri bizim sistemimiz için özel olarak çekilmiş görüntüler olmadıklarından çoğunlukla eğik açıdan çekilmiş görüntülerdi. Ancak modelimiz tepeden çekilen görüntülerle başarılı bir şekilde çalışmaktadır. Bu da bir model başarısıdır.

Sistemimizin bulguları arasında da paylaştığımız gibi sistem sulu yemeklerde yüksek başarılı sonuçlar verirken katı yemeklerde, sulu yemekler kadar başarılı olamayabiliyor. Bunun önüne geçmek için katı yemekleri analiz ederken farklı yöntemler denenebilir. Örneğin yaprak sarması yemeği için görüntüden bir adet yaparak sarması tanındıktan sonra adet sayımı yapılarak daha doğru sonuçlar elde edebiliriz. Sistem sulu yemeklerdeki başarısı ise son kullanıcı tarafından rahatlıkla kullanılabilecek seviyededir. Referans obje eğitim görsellerinin az olmasından dolayı bazen tanımama problemleri ile karşılaşabilmekte ancak bu sorunun düzgün ışık ile önüne geçilmektedir.

Sistemimiz daha sonradan farklı açılar ile 3 boyutlu yemek analiz işlemleri için uygun olup o yönde geliştirilebilir. Kurduğumuz sistem internet erişimi olan herhangi bir telefon ile kolaylıkla kullanılabilineceğinden kolaylıkla tüm dünya üzerinden herhangi bir kullanıcının grafik işlem birimi ya da işlemcisini kullanmadan sunucu üzerinden alaniz yaptırması sağlanılabileceği gibi istenirse kolaylıkla Android veya IOS işletim sistemi için uygun mobil uygulamaya da dönüştürülebilir. Şu an için 17 sınıfın tanımlı olduğu uygulamada eğitim verileri ve yapılan eğitim iterasyonu ile alakalı tam anlamıyla düzgün bir şekilde çalışan 12 adet sınıf bulunmaktadır. Eğer sistem üzerinde gerekli geliştirmeler yapılırsa sınıf sayısı ve eğitim verilerinin kalitesi artırılırsa son kullanıcı tarafından kullanılmasının önünde hiçbir engel yoktur. Sistemimiz içerisindeki tanımlı sınıflar ile sorunsuz bir şekilde çalışmakta ve isteyen herkes rahatlıkla kullanabilmektedir.

KAYNAKÇA

[1] Nilson, N. J., 1998, Introductıon to machıne learnıng, Stanford University, USA 89 s

[2] By: IBM Cloud Education. (n.d.). 2021 what is machine learning? (Web sitesi: https://www.ibm.com/cloud/learn/machine-learning#toc-deep-learn-nOh7s5Rf) (Erişim Tarihi : Nisan 2021)

[3] McCulloch, W. S., & Pitts, W. 1943. a logical calculus of the ideas immanent in nervous activity.â the bulletin of mathematical biophysics. s115-133

[4]Rosenblatt, F. 1958. the perceptron: a probabilistic model for information storage and organization in the brain. Psychological review, 65(6), 386.

[5] Kelley, H. J. 1960. gradient theory of optimal flight paths. Ars Journal, 30(10), 947-954.

[6] Dreyfus, S. 1962. The numerical solution of variational problems. Journal of Mathematical Analysis and Applications, 5(1), 30-45.

[7] Ivakhnenko, A. G., & Lapa, V. G. 1965. Cybernetic predicting devices. Kiev.

[8] Fukushima, K. 1979. neural network model for a mechanism of pattern recognition unaffected by shift in position-Neocognitron. IEICE Technical Report, A, 62(10), 658-665.

[9] Linnainmaa, S. (1970). the representation of the cumulative rounding error of an algorithm as a taylor expansion of the local rounding errors. Yüksek Lisans Tezi (Fince), Univ. Helsinki, 6-7.

[10] Rumelhart, D. E., Hinton, G. E., & Williams, R. J. 1985. learning internal representations by error propagation. California Univ San Diego La Jolla Inst for Cognitive Science.

[11] Foote, K. 2017. a brief history of deep learning. (Erişim Tarihi: Nisan 03, 2021) (Web sitesi : https://www.dataversity.net/brief-history-deep-learning/#).

[12] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. 1989. backpropagation applied to handwritten zip code recognition. Neural computation, 1(4), 541-551.

[13] Cortes, C., & Vapnik, V. 1995. support-vector networks. Machine learning, 20(3), 273-297.

[14] Hochreiter, S., & Schmidhuber, J. 1997. long short-term memory. Neural computation, 9(8), 1735-1780.

[15] Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L. 2009. ımagenet: a large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition (s. 248-255).

[16] Krizhevsky, A., Sutskever, I., & Hinton, G. E. 2012. ımagenet classification with deep convolutional neural networks. Advances in neural information processing systems, 25, 1097-1105.

[17] Yapay Sinir ağları. 2021 (Web sitesi: https://tr.wikipedia.org/wiki/Yapay_sinir_a%C4%9Flar%C4%B1), (Erişim Tarihi : Nisan 04, 2021)

[18] Öztürk, K., & Şahin, M. E. 2018. yapay sinir ağları ve yapay zekâ'ya genel bir bakış. Takvim-i Vekayi, 6(2), 25-36.

[19] Mehta, A. 2020. a comprehensive guide to types of neural networks. digital vidya. (Web sitesi: https://www.digitalvidya.com/blog/types-of-neural-networks/) (Erişim Tarihi: Nisan 2021)

[20] Ababaei, B., Sohrabi, T. M., Mirzaei, F., Araghinejad, S., & Ahmadizadeh, M. (2012). suitability of different neural networks in daily reservoir inflow simulation. In The First International Conference on Dams and Hydropower in Iran, Tehran.

[21] Yarıçapsal Temelli Ağ (Radial Basis Network). 2017. (Web sitesi: https://veribilimcisi.com/2017/09/26/yaricapsal-temelli-ag-radial-basis-network/) (Erişim Tarihi: Nisan 2021)

[22] Kızrak, A. 2020. derine daha derine: Evrişimli Sinir Ağları - Ayyüce Kızrak. Medium.(Web sitesi : https://ayyucekizrak.medium.com/deri%CC%87ne-daha-deri%CC%87ne-evri%C5%9Fimli-sinir-a%C4%9Flar%C4%B1-2813a2c8b2a9) (Erişim Tarihi: Nisan 2021)

[23] Akca, M. F. 2020. Nedir Bu Destek Vektör Makineleri? (Makine Öğrenmesi Serisi-2). Medium. (Web sitesi: https://medium.com/deep-learning-turkiye/nedir-bu-destek-vekt%C3%B6r-makineleri-makine-%C3%B6%C4%9Frenmesi-serisi-2-94e576e4223e) (Erişim Tarihi: Nisan 2021)

[24] Hopfield Ağı (Hopfield Network). 2017. Veri Bilimcisi. (Web sitesi: https://veribilimcisi.com/2017/09/26/hopfield-agi-hopfield-network/)(Erişim Tarihi: Nisan 2021)

[25] Boltzmann Makinesi (Boltzmann Machine). 2017. Veri Bilimcisi. (Web sitesi : https://veribilimcisi.com/2017/09/26/boltzmann-makinesi-boltzmann-machine/) (Erişim Tarihi: Nisan 2021)

[26] Evrişimli Sinir Ağları el kitabı Star. CS 230 - Evrişimli Sinir Ağları El Kitabı. (n.d.). (Web sitesi: https://stanford.edu/~shervine/l/tr/teaching/cs-230/cheatsheet-convolutional-neural-networks#object-detection.) (Erişim Tarihi: Nisan 2021)

[27] YOLO Nedir ? (n.d.). (Web sitesi: http://gorselanaliz.com/yolo-nedir.)(Erişim Tarihi: Nisan 2021)

[28] Doğan, Ö. 2020. Nesne Tanıma Algoritmaları: R-CNN, Fast R-CNN ve Faster R-CNN Nedir? Teknoloji.org. (Web sitesi: https://teknoloji.org/nesne-tanima-algoritmalari-r-cnn-fast-r-cnn-ve-faster-r-cnn-nedir/#Region_Based_CNN_R-CNN). (Erişim Tarihi: Nisan 2021).

[29] Gandhi, R. 2018. R-CNN, Fast R-CNN, Faster R-CNN, YOLO — Object Detection Algorithms. Medium. (Web sitesi: https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e). (Erişim Tarihi: Nisan 2021)

[30] Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. arXiv preprint arXiv:1506.01497.

[31] Khandelwal, R. 2019. Computer Vision: Instance Segmentation with Mask R-CNN. Medium. (Web sitesi: https://towardsdatascience.com/computer-vision-instance-segmentation-with-mask-r-cnn-7983502fcad1). (Erişim Tarihi: Nisan 2021).

[32] Özkan, İ. N. İ. K., & Ülker, E. 2017. derin öğrenme ve görüntü analizinde kullanılan derin öğrenme modelleri. Gaziosmanpaşa Bilimsel Araştırma Dergisi, 6(3), 85-104.

[33] Krizhevsky, A., Sutskever, I., & Hinton, G. E. 2012. ımagenet classification with deep convolutional neural networks. Advances in neural information processing systems, 25, 1097-1105.

[34] Jay, P. (2018, July 21). ımage classification architectures review - prakash jay. medium. (Web sitesi: https://medium.com/@14prakash/image-classification-architectures-review-d8b95075998f). (Erişim Tarihi: Nisan 2021).

[35] Zeiler, M. D., & Fergus, R. 2014. visualizing and understanding convolutional networks. In European conference on computer vision (s. 818-833)

[36] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. 2015. going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (s. 1-9).

[37] Simonyan, K., & Zisserman, A. 2014. very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

[38] He, K., Zhang, X., Ren, S., & Sun, J. 2016. deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (s. 770-778).

[39] R.G.I.R.G.G.P.D. (2018). Detectron. GitHub. (Web sitesi : https://github.com/facebookresearch/Detectron) . (Erişim Tarihi: Nisan 2021).

[40] T. (n.d.). How to train Detectron2 with Custom COCO Datasets. Google. Retrieved May 5, 2021, from Web sitesi : (https://colab.research.google.com/github/Tony607/detectron2_instance_segmentation_demo/blob/master/Detectron2_custom_coco_data_segmentation.ipynb#scrollTo=7unkuuiqLdqd). (Erişim Tarihi: Nisan 2021).

[41] Wikipedia contributors. 2021. Impulse response. Wikipedia. (Web sitesi: https://en.wikipedia.org/wiki/Impulse_response#:%7E:text=In%20signal%20processing%2C%20the%20impulse,response%20to%20some%20external%20change) (Erişim Tarihi: Nisan 2021).

[42] Ioffe, S., & Szegedy, C. (2015, June). Batch normalization: accelerating deep network training by reducing internal covariate shift. In International conference on machine learning (s. 448-456). PMLR.

[43] Yığın Normalleştirme (Batch Normalization). 2018. Tolga Oğuz Kişisel Blog. (Web sitesi: http://tolgaoguz.net/2018/09/06/batchnormalization/) . (Erişim Tarihi: Nisan 2021).

[44] Yegulalp, S. (2019, June 18). What is TensorFlow? The machine learning library explained. InfoWorld.(Web sitesi: https://www.infoworld.com/article/3278008/what-is-tensorflow-the-machine-learning-library-explained.html). (Erişim Tarihi: Nisan 2021).

[45] Education, I. C. (2021, Nisan 30). Deep Learning. IBM. (Web sitesi : https://www.ibm.com/cloud/learn/deep-learning). (Erişim Tarihi: Nisan 2021).

[46] Hakkomaz, H. 2021. hhkkmz/19101010. GitHub. (Web sitesi:

https://github.com/hhkkmz/19101010.) . (Erişim Tarihi: Nisan 2021).

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı, Soyadı	: Hüseyin HAKKOMAZ
Uyruğu	: T.C.
Doğum Tarihi ve Yeri	: 1 Ocak 1996, Kayseri
Medeni Durumu	: Bekar
Telefon	: +90 506 136 95 45
Email	: hakkomazhuseyin@gmail.com
Yazışma adresi	: Sakarya mah Aybey sok. 4/2 Melikgazi/KAYSERİ

EĞİTİM

Derece	Kurum	Mezuniyet Tarihi
Lisans	Elektrik Elektronik Mühendisliği, Erciyes Üniversitesi, Kayseri	2019
Lise	Özel Hisarcıklıoğlu Fen Lisesi,Kayseri	2014

İŞ DENEYİMLERİ

Yıl	Kurum	Görev
2014- Halen	Akan İnşaat San ve Tic. Ltd. Şti	Müdür
2021- Halen	Alpün Yapı Denetim Ltd Şti	Elektrik Mühendisi

YABANCI DİL

İngilizce