SİVAS CUMHURİYET ÜNİVERSİTESİ
Sosyal Bilimler Enstitüsü
Yönetim Bilişim Sistemleri Ana Bilim Dalı

DERİN ÖĞRENME KULLANILARAK PERAKENDE ÜRÜN TESPİTİ

Yüksek Lisans Tezi

İsmail KÖSE

Sivas
Mayıs 2022

SİVAS CUMHURİYET ÜNİVERSİTESİ
Sosyal Bilimler Enstitüsü
Yönetim Bilişim Sistemleri Ana Bilim Dalı

DERİN ÖĞRENME KULLANILARAK PERAKENDE ÜRÜN TESPİTİ

Yüksek Lisans Tezi

İsmail KÖSE

Tez Danışmanı
Prof. Dr. Oğuz KAYNAR

Sivas
Mayıs 2022

ETİK İLKELERE UYGUNLUK BEYANI

Sivas Cumhuriyet Üniversitesi Sosyal Bilimler Enstitüsü bünyesinde hazırladığım bu Yüksek Lisans Tezinin bizzat tarafımdan ve kendi sözcüklerimle yazılmış orijinal bir çalışma olduğunu ve bu tezde;

1- Çeşitli yazarların çalışmalarından faydalandığımda bu çalışmaların ilgili bölümlerini doğru ve net biçimde göstererek yazarlara açık biçimde atıfta bulunduğumu;
2- Yazdığım metinlerin tamamı ya da sadece bir kısmı, daha önce herhangi bir yerde yayımlanmışsa bunu da açıkça ifade ederek gösterdiğimi;
3- Başkalarına ait alıntılanan tüm verileri (tablo, grafik, şekil vb. de dâhil olmak üzere) atıflarla belirttiğimi;
4- Başka yazarların kendi kelimeleriyle alıntıladığım metinlerini, tırnak içerisinde veya farklı dizerek verdiğim yine başka yazarlara ait olup fakat kendi sözcüklerimle ifade ettiğim hususları da istisnasız olarak kaynak göstererek belirttiğimi,

beyan ve bu etik ilkeleri ihlal etmiş olmam halinde bütün sonuçlarına katlanacağımı kabul ederim.

27./05/2022
İsmail KÖSE

ÖNSÖZ

Yüksek Lisans eğitimim boyunca fikirleri ve desteğiyle beni doğru bir şekilde yönlendiren ve desteğini esirgemeyen kıymetli danışman hocam Prof. Dr. Oğuz KAYNAR'a; hayatımın her aşamasında yanımda olan çok sevdiğim ailem, eşim ve çocuklarıma teşekkürlerimi bir borç bilirim.

İsmail KÖSE

İÇİNDEKİLER

KISALTMALAR

CNN : Evrişimsel Sinir Ağları

CPU : Merkezi İşlem Birimi

DWM : Dinamik Pencere Yönetim

ESA : Evrişimli Sinir Ağları

GAN : Çekişmeli Üretici Ağlar

GPU : Grafik İşlem Birimi

LSTM : Uzun Kısa Süreli Hafıza

LSTM : Uzun Kısa Süreli Hafıza

NLP : Doğal Dil İşleme

RBM : Sınırlı Boltzman Makineleri

R-CNN : Bölge Bazlı Evrişimsel Sinir Ağı

RNN : Tekrarlayan Sinir Ağları

RPN : Bölge Öneri Ağı

SSD : Tek Atış Çoklu Kutu Algılama

YOLO : Sadece Bir Defa Bakarsın

YSA : Yapay Sinir Ağı

TABLO LİSTESİ

ŞEKİL LİSTESİ

ÖZET

Derin öğrenme ve bu konudaki gelişmeler günümüzde insan yaşamındaki zor ve karmaşık olan birçok problemin çözümünde oldukça önemli bir yere sahip olmuşlardır. Özellikle bilgisayarlı görü konusu ; müşteriler için mağaza içi alışveriş yardımı, Google Lens tarafından kullanılan görsel ürün arama, mağazalarda kasaların ve kasiyerlerin kullanılmadığı mağazalar (Amazon Go), mağaza yönetimleri için gerçek zamanlı stok takibi ve mağazaların raf yerleşimlerini belirleyebilen uygulamaların geliştirilmesiyle perakende süreçlerinin otomasyonunda oldukça önemli ve işlevsel bir araç haline gelmiştir. Geliştirilen bu akıllı uygulamaların temelinde, nesne tanıma uygulamalarının aksine çeşitli yeni zorluklar içeren ürün tanıma sorununa çözüm bulma çabaları yatmaktadır.

Ürün tanıma sorunu birçok ayrıntıya sahip benzer ürünleri tespit etmeyi amaçlayan bir sınıflandırma problemi olarak kabul edilebilir. Herhangi bir süpermarketteki aynı markaya ait ürünlerde bile şekil, ambalaj, paket boyutu gibi küçük görsel farklılıklar olması nedeniyle ayırt edilmeleri oldukça zor olan binlerce farklı paketlenmiş ürün olduğu düşünüldüğünde ürün tanıma işleminde ürün görsellerinin belirlenmesi oldukça büyük bir zorluk barındırmaktadır. Karşılaşılan diğer bir zorluk ise, eğitim setleri için elde edilen görsellerin stüdyo koşullarında ve her ürün için birkaç eğitim görseline sahip sınırlı sayıda veri kümesi olmasıdır. Bunun sonucu olarak araştırmacılar veri kümelerini genişletmek için gerçek bir perakende ortamında raftan ürün görüntüleri alınarak elde etmeyi tercih etmektedirler. Bu durumda ise elde edilen görüntülerde bulanıklık, düşük çözünürlük, kapanma, beklenmedik arka planlar vb. sorunları beraberinde getirmektedir.

Günümüzde nesne tanıma işleminin çözümü için çok katmanlı sinir ağlarının geliştirilmiş hali olan Evrişimsel Sinir Ağları (CNN-Convolutional Neural Network) kullanılmaktadır. Yapılan çalışmalarda CNN'lerin, görüntü üzerindeki nesnenin tespit edilmesi ve sınıflandırılması problemlerinde oldukça iyi sonuçlar verdiği görülmüştür.

Derin öğrenme tabanlı nesne tanıma yöntemleri kullanılarak geliştirilecek otomatik ürün tanıma otomasyonları, insan merkezli sistemler düşünüldüğünde gerek zaman tasarrufu gerekse güvenilirlik bakımından hem daha ekonomik hem de sosyal

ilerleme için büyük öneme sahiptir. Görüntüleme sistemleri kullanılarak elde edilen görüntüler ile ürün tanıma, bilgisayarlı görü konusunda oldukça zorlu bir iştir. Ancak nesne tanıma kullanılarak geliştirilecek otomasyonlar sayesinde otomatik ödeme, stok takibi, planogram uyumluluğu ve görme engelli yardımı gibi birçok konu çok hızlı ve yüksek başarı oranıyla çözüme kavuşturulabilecektir.

Bu çalışmada, literatürdeki nesne tanıma işleminin çözümü için kullanılan derin öğrenme yöntemleri ve nesne tanımada karşılaşılabilecek sorunlar incelenmiş, nesne tanıma işlemi, özel bir konu olan süpermarket raflarındaki perakende ürünlerin tespitinde kullanılmıştır. Yapılan bu çalışma ile süpermarket raf resimleri kullanılarak raflardaki ürünlerin etrafına sınırlayıcı kutular çizilerek ve ürünlerin tespit edilmesi amaçlanmıştır.

Bu çalışmada popüler derin öğrenme yöntemlerinden YOLOv5 ve Faster R-CNN kullanılmış, bu yöntemlerin başarımları ve karşılaşılabilecek problemler ile ilgili değerlendirmeler yapılmıştır.

Anahtar Kelimler: Derin Öğrenme, Nesne Tanıma, Derin Öğrenme Modelleri, Evrişimsel Sinir Ağı, YOLO, Faster R-CNN

ABSTRACT

Deep learning and developments in this field have a very important place in solving many difficult and complex problems in human life today. Especially the subject of computer vision; It has become a very important and functional tool in the automation of retail processes with the development of in-store shopping assistance for customers, visual product search used by Google Lens, stores where cashiers and cashiers are not used in stores (Amazon Go), real-time inventory tracking for store management and the development of applications that can determine the shelf placement of stores. has arrived. On the basis of these developed smart applications, there are efforts to find solutions to the problem of product recognition, which includes various new challenges, unlike object recognition applications.

The product identification problem can be regarded as a classification problem that aims to identify similar products with many details. Considering that there are thousands of different packaged products that are difficult to distinguish due to small visual differences such as shape, packaging, package size, even in products of the same brand in any supermarket, identifying product images in the product recognition process poses a great challenge. Another challenge is that the images obtained for the training sets are limited in studio conditions and with a few training images for each product. As a result, researchers prefer to obtain off-the-shelf product images in a real retail environment to expand their dataset. In this case, blur, low resolution, closure, unexpected backgrounds, etc. in the images obtained. brings with it problems.

Today, Convolutional Neural Networks (CNN - Convolutional Neural Network), which is the developed version of multi-layered neural networks, are used for the solution of object recognition. In the studies, it has been seen that CNNs give very good results in the problems of detecting and classifying the object on the image.

Automatic product recognition automations, which will be developed using deep learning-based object recognition methods, are of great importance for both economic and social progress in terms of both time saving and reliability when human-centered systems are considered. Product recognition with images obtained using imaging systems is a very challenging task in computer vision. However, thanks to the

automations to be developed using object recognition, many issues such as automatic payment, stock tracking, planogram compatibility and assistance for the visually impaired will be solved very quickly and with a high success rate.

In this study, deep learning methods used for the solution of the object recognition process in the literature and the problems that may be encountered in object recognition were examined, and the object recognition process was used in the detection of retail products on the supermarket shelves, which is a special subject. With this study, it is aimed to identify the products by drawing bounding boxes around the products on the shelves using supermarket shelf pictures.

In this study, YOLOv5 and Faster R-CNN, which are popular deep learning methods, were used, and the performances of these methods and the problems that could be encountered were evaluated.

Keywords: Deep Learning, Object Recognition, Deep Learning Models, Convolutional Neural Network, YOLO, Faster R-CNN

GİRİŞ

Bilgisayarlı görü konusunun alt çalışma alanı olarak kabul edilen nesne tanıma problemi gelişen teknolojiyle birlikte günümüzde gerçek hayatta karşılaşılan birçok problemin çözümünde etkin bir şekilde kullanılmaya başlanmıştır. Sürücüsüz otonom araçlar için geliştirilen nesne tanıma yöntemleri sayesinde araçların yol üzerindeki ve çevredeki nesnelerin tanınmasından yaya tespitine, trafik işaretlerinin algılanmasından yol çizgilerinin belirlenerek şerit takibine kadar birçok problemin çözümünde başarıyla kullanıldığı görülmektedir. Bununla birlikte kamu hizmetleri için geliştirilen akıllı trafik uygulamaları sayesinde plaka tespiti ve aracın takibi, yoldaki araç sayısının tespit edilerek trafik yoğunluğunun belirlenmesi veya trafik ışıklarının otomatik kontrolünde, kural ihlali yapan araçların belirlenmesinde nesne tanıma ve tespitinden faydalanılmaktadır. Özellikle güvenlik amaçlı yüz tanıma, şüpheli kişilerin, paketlerin ve olayların tespit ve takibi de nesne tanıma uygulamaları arasında sayılmaktadır.

Nesne tanımanın kullanıldığı diğer bir güncel alan ise elde edilen uydu görüntülerini kullanarak uzaktan nesne algılamadır. Elde edilen uydu görüntülerini kullanarak tarımsal alandaki ekili ve boş arazilerin tespit edilebildiği gibi ormanlık alanlardaki yangınların tespitine, şehir planlama amaçlı binaların cadde ve arsaların tespitine hatta enerji nakil hatları üzerindeki yabancı cisimlerin belirlenmesine kadar birçok alanda nesne tanımanın etkili bir şekilde kullanıldığı görülmektedir. Bunun yanısıra tıp ve sağlık alanında elde edilen medikal görüntüler kullanılarak özellikle erken teşhis amaçlı kanser hücresi (Yancı 2019), beyin tümörü tespiti (Aslan 2022), sağlıklı ve sağlıksız sperm hücrelerinin tespiti (Alhajalabdulla 2020) gibi alanlarda nesne tespitinden yararlanılmaktadır.

Bilgisayarlı görü ve nesne tespiti uygulamalarının sıklıkla kullanıldığı sektörlerden birisini de perakende sektörü ve süper marketler oluşturmaktadır. Gündelik yaşantımızda neredeyse hepimizin doğrudan etkileşim içerisinde olduğu süpermarket ya da perakende ürün satışı yapılan mağazalar hem mağaza yönetimi için hem de müşteri etkileşimi açısından raf yönetimi, envanter analizi, müşteri analizi, ürün ödemeleri gibi birçok zorluğun olduğu çalışma alanlarıdır. Bu zorlukların çözümü için hali hazırda sürekli tekrarlanan, önemli ölçüde personel ve zaman

ihtiyacının ortaya çıktığı ve insan merkezli yöntemler kullanılmaktadır. Bununla beraber özellikle teknolojideki hızlı gelişmeler perakende ürün sektörünün de yönetimsel ya da müşteri odaklı problemlerin çözümleri için yeni arayışlara girmesine sebep olmuştur. Perakende sektörü büyük bir hızla gelişirken, işletmeler perakende sektörünün ekolojisini yeniden şekillendirmek, çevrimiçi ve çevrimdışı deneyimleri entegre etmek için yapay zekâ teknolojisinin nasıl kullanılacağına giderek daha fazla odaklanmaktadırlar (Grewal vd, 2017). Juniper Research tarafından yapılan araştırmaya göre, perakendecilerin yapay zekâ hizmetlerine yaptığı küresel harcama, 2019'da 3,6 milyar dolardan 2023'te 12 milyar dolara %300'ün üzerinde artış göstereceği öngörülmektedir (Hampshire 2019). Kısaca gelecekte yeni inovatif perakendecilik tamamen yapay zekâ teknolojisi ile gerçekleştirilebilir.

Her şeyin çok hızlı ve dinamik şekilde değiştiği günümüzde süper marketlerdeki alışveriş şekilleri, müşteri deneyim ve beklentileri de değişmektedir. Müşteriler marketlerde geçirdikleri süreleri daha etkin ve verimli kullanmak istemekte, aradığı ürünlere kısa sürede erişip, kasada sıra beklemek yerine otomatik ödeme sistemleriyle alışverişlerini hızlıca tamamlamak istemektedirler. Süper marketler ise personel ve zaman gerektiren birçok rutin işi otomatikleştirerek maliyetlerini düşürmeyi, girişte müşteriyi tanıyarak ilgilenebileceği kampanya ve ürün önerileriyle daha iyi bir alışveriş deneyimi sunarak müşteri memnuniyetini sağlamayı ve karlığını artırmayı hedeflemektedirler.

Süper market raflarından elde edilen görselleri kullanarak geliştirilecek olan bir otomatik ürün tanıma uygulamasının perakende sektörü üzerinde önemli bir etkisi olacağı düşünülmektedir. Geliştirilecek uygulama sayesinde ilk olarak raftaki ürünlerin planogram uyumunun kontrolü etkin bir şekilde yapılabilir. Bu sayede uygulama son derece hızlı ve etkili bir şekilde market raflarındaki eksik ürünlerin otomatik tespitini yaparak ilgili mağaza personeline ürünleri hemen yenilemeleri gerektiği konusunda bilgi verebilir. Optimize edilmiş bir planogram %100 eşleştirildiğinde satışların %7,8, kârın ise %8,1 oranında artacağı görülmüştür (Shapiro 2019). Bununla beraber perakende ürün maliyetlerini azaltmak ve müşteri deneyimini geliştirmek amacıyla 2014-2019 yılları arasında artan sayıda küresel self-checoutların (SCO) kurulmasıyla birlikte müşteri deneyimini optimize etmek için müşterilerin aldıkları ürünler için yapacakları ödemelerde, otomatik ödeme

sistemlerine görüntü tabanlı ürün tanımlamanın uygulanabilmesi etkili sonuçlar ortaya koyacağı göz ardı edilememelidir. Araştırmacılar Orel, Wu,Van Riel ve Morimura yapmış oldukları çalışmada, müşterilerin ödeme işlemleri için bekleme sürelerinin, alışveriş tatminleri üzerinde olumsuz bir etkisi olduğunu, yani SCO'larda bilgisayarlı görüye dayalı bir ürün tanıma uygulamasının hem perakendecilere hem de müşterilere fayda sağladığını göstermişlerdir.

Nesne tanımanın kullanıldığı diğer bir uygulama alanında ise ürün tanıma teknolojisi, görme engelli kişilerin bağımsız olarak alışveriş yapmalarına yardımcı olabilir, bu da onların alışverişlerinde çeşitli kolaylıklar sağlayabilir (George ve Florkemeier 2014). Geleneksel alışveriş yöntemleri genellikle gören bir kişiden yardım gerektirir, çünkü görme engelli bir kişinin ürünleri görsel özellikleriyle (örneğin, fiyat, marka ve son tarih) tanımlaması zor olabilir ve satın alma kararlarını zorlaştırabilir (Lopez vd, 2011). Bir diğer husus ise müşterinin sadakat kartı ya da üye kartı taşımaksızın bilgisayarlı görü yardımıyla otomatik olarak tanınması müşteriye özgü indirim kampanyalardan müşterinin haberdar edilmesi ve müşterinin daha önceki alışveriş deneyimlerine göre ürün öneren bir sistem sayesinde müşteri bağlılığı ve sadakatinin artırılması hedeflenebilir. Müşteriler ürün tanıma teknolojisini kullanan bir mobil uygulama yazılımı sayesinde ürüne ait bilgilere çok daha kolay ulaşılabilecektir. Bu bilgiler kullanılarak ürünün alış fiyatı, satış fiyatı, üretim tarihi, son kullanma tarihi, stoktaki miktarı gibi bilgiler yanında üretildiği yer, içindekiler, eğer bir tekstil ürünüyse renk ve beden seçenekleri, benzer ürünler, kampanyalar gibi bilgilere erişmek mümkün olacaktır. Aynı zamanda sepet üzerinde yerleştirecek bir ürün tanıma sistemiyle müşteriye ait sepekteki alışveriş tutarı kasaya gelmeden hesaplanabilecektir.

Ayrıca yaşam standartlarının yükselmesiyle birlikte süpermarket çalışanları ve müşterileri sayısız perakende ürünle karşı karşıya kalmaktadırlar. Böyle bir senaryoda, klasik yöntemlerle ürün yönetimini etkin bir şekilde gerçekleştirmek için büyük miktarda insan emeği ve iş yükü gerekmektedir. Ayrıca, fotoğraf çekmek için kullanılan çeşitli elektronik cihazların yardımıyla, ürünlerin ait dijital resimler her geçen gün hızla artmaktadır. Bu nedenle, muazzam miktarda görüntü verisi için, bunların nasıl etkili bir şekilde analiz edilip işleneceği, bu görüntülerden ürünleri tanıyıp sınıflandırılması, nesne tanıma alanında önemli bir araştırma konusu haline

gelmiştir. Tüm bunları gerçekleştirmek için market içerinde yer alan bazı iş ve süreçlerin akıllı hale getirilerek otomatikleştirilmesi gerekmektedir. Bunun en güzel örneği, sadece belirli bir mağaza için geliştirilen Amazon Go projesidir. Bu projede mağazanın her tarafı kameralar ve sensörlerle donatılmıştır. Müşteriler mağaza içerisine girdiği andan itibaren herhangi bir personel olmadan alışverişlerini yaparak kasiyersiz ödeme sistemlerinden faydalanabilmektedir. Bununla beraber tüm mağazayı sensörler ve kameralarla donatmak ciddi bir maliyet oluşturacağından küçük bütçeli işletmeler için uygulanması şimdilik zor gözüksede ileride maliyetlerin düşmesiyle birlikte sistemin daha çok yaygınlaşacağı öngörülmektedir. Akıllı market uygulamalarının diğer bir örneği ise Walmart's Intelligent Retail Lab projesidir.

Ürün tanıma, bilgisayarların ürünleri manuel olarak tanımlama ve sınıflandırma sürecinin yerini alabilmesi için temel olarak bilgisayarlı görme yöntemlerine dayanan teknolojinin kullanımını ifade eder. Süpermarketlerde kullanılan akıllı uygulamaların birçoğu raflarda ve reyonlarda sergilenen ürünün tespiti ve tanınmasına dayanmaktadır. Ürünlerin tanınması amacıyla yaygın olarak her ne kadar barkod sistemi kullanılsada, bir barkod okuyucuya ihtiyaç duyması, barkodların ürünlerde farklı yerlerde olmasından dolayı barkodun bulunması için belirli bir zaman harcanmasını, bazı barkodların silik basılmasından dolayı okunamaması gibi problemlerle karşılaşılmaktadır. Bu soruna çözüm olması amacıyla RFID taglar önerilmiştir. RFID etiketlerinin yerleştirilmesi geleneksel yöntemler düşünüldüğünde çalınma, ödeme kolaylığı, ürün tanımanın otomatikleştirilmesi ve envanter takibi açısından oldukça sağlam, hızlı ve güvenli bir yöntem olarak düşünülsede radyo frekansından etkilenme ve özellikle ürün maliyetlerinin düşük olduğu işletmeler için ekonomik yükler getirmek gibi dez avantajlar içermektedir. Son yıllarda bilgisayarlı görü ve nesne tanımadaki başarılar bu yöntemin ürün tanımada barkod ve RFID kullanıma alternatif hale getirmiştir.

Son yıllarda artan veri sayısı, işlemci gücünün yanına, öğrenme algoritmalarındaki gelişmelerle birlikte derin öğrenme yöntemleri olarak bir gurup yeni sinir ağı geliştirilmiştir. Özelikle Derin evrişimsel sinir ağlarının ortaya çıkmasıyla birlikte klasik yöntemlere oranla görüntü sınıflandırma ve nesne tanımada başarılı sonuçlar elde edilmiş, ardından bu ağlarındaki gelişmelerle birlikte derin öğrenme yöntemleri görüntü sınıflama ve nesne tanımada temel çözümler haline

gelmiştir. Nesne tanıma işlemi için en çok kullanılan derin öğrenme algoritması, Evrişimsel Sinir Ağı (CNN) kabul edilir (Angelova vd, 2015; Hinton vd, 2012). Evrişimsel sinir ağı giriş olarak verilen görüntüdeki, nesneleri sınıflandırabilen bir derin öğrenme algoritmasıdır. Nesne tanıma probleminde kullanılan bir evrişimsel sinir ağı, evrişim katmanı, havuzlama katmanı, aktivasyon fonksiyonu, dropout katmanı, tam bağlı katman ve sınıflandırma katmanından oluşmaktadır (Guo vd, 2016).

Derin öğrenme ve geleneksel görüntü tanıma yöntemleri arasındaki temel fark, ilkinin manuel olarak resimden elde edilen öznitelikleri kullanmak yerine doğrudan görüntü verilerinden öz nitelikleri öğrenebilmesidir. Derin öğrenmenin güçlü yeteneğinin bir başka nedeni de geleneksel sinir ağlarından daha iyi özellikler çıkarabilen daha derin katmanlara sahip olmasıdır. Nesne tanıma problemi, nesnelerin algılanmasını, yerelleştirilmesini ve sınıflandırılmasını gerektiren bir bilgisayarlı görü görevidir. Bu görevde, elde edilen görüntüde ilişkili nesne olup olmadığını anlamak için ilk başta bir makine öğrenimi modeline ihtiyacımız vardır. Eğer görüntüde bir nesne tespit edilmişse nesnenin çevresine bir sınırlayıcı kutu çizilmeli ve modelimizin sınırlayıcı kutunun temsil ettiği nesneyi sınıflandırması beklenmektedir.

Perakende ürünlerin otomatik tanınması konusunda daha öncen yapılmış olan çalışmalar incelendiğinde araştırmacıların çalışmalarında farklı modellerin başarımlarını sık sık test etmeye çalıştıkları görülmüştür.

Chong T. vd 2016 yılında yapmış oldukları çalışmalarında önceden eğitilmiş bir CNN görüntü sınıflandırıcısı olan ve ImageNet'den alınan verilerle eğitilmiş olan Inception V3 ile transfer öğrenme gerçekleştirilmiştir. Çalışmalarında üç ayrı model eğitmişlerdir. Bu modellerden M1 olarak isimlendirdikleri ilk modeli sadece belirli bir mağazadan elde ettikleri ürün görsellerle, M2 olarak isimlendirdikleri ikinci modeli belirlenen sekiz farklı ürün grubu için internetten elde ettikleri görsellerle ve M3 olarak isimlendirdikleri üçüncü modeli ise %50 belirli bir mağazadan elde ettikleri %50 internetten elde ettikleri görselleri kullanarak eğitmişlerdir. Bu sayede hem modelin aşırı öğrenmesinin önüne geçmeyi hem de modelin daha önce hiç görmediği ürünler üzerindeki test başarısını arttırmayı amaçlamışlardır. Bu modeller her ürün grubu için 50 görselden oluşan (25'i eğitim için kullanılan görsellerden, 25'i daha önce hiç

kullanılmayan görsellerden) test kümesi üzerinde test edilmiş ve modellerin ortalama başarımlarının M1 modeli için %87,5, M2 modeli için %75 ve M3 modeli için %89,5 doğruluk oranına sahip olduğu görülmüştür (Chongvd, 2016).

Geng W. vd. 2018 yılında yapmış oldukları çalışmalarında, 2 farklı marketten topladıkları 102 sınıf içeren gerçek ürün görsellerini kullanarak ürün tanıma senaryosu için hibrit bir yöntem önermişlerdir. Önerdikleri yöntem görsellerdeki ürünleri iki adımda sınıflandırmayı amaçlamıştır. Bu yöntem birinci adımda ürünlerin kaba hatlarını belirlemeyi ikinci adımda ise belirlenen bu hatların ayırt edici özellikleri (ROI) belirlenerek diğer ürünlerden farklı bir sınıfa yerleştirilmesini amaçlamıştır. Yöntem ürün tanıma senaryosunda ürün örneklerini tanımak için özellik tanımlayıcısı olarak VGG-16'yı kullanmışlar ve görüntülerdeki ürünleri %75 ortalama tahmin doğruluğuyla tespit etmişlerdir (Geng vd, 2018).

Karlinsky L,vd. 2017 yılında yapmış oldukları çalışmada üçü kendileri tarafından oluşturulan (Point Clouds Pose Estimation veri seti, video oyunları veri seti (3.7K kategori) ve perakende ürünler veri seti (121 kategori)) veri setleri ve üçü hazır olarak elde edilen (Grozi-120, Grozi 3.2K ve FlickrLogos-32) veri setleri olmak üzere toplam altı veri setini kullanmış ve çalışmalarını üç aşamalı olarak modellemişlerdir. Önerdikleri model birinci adımda görseldeki ürünleri hızlı bir şekilde belirlemeyi, ikinci adımda belirlenen sınıflandırmaları daha da iyileştirmeyi ve son adımda ise zamansal entegrasyon ve iyileştirmeyi amaçlamıştır. Yapmış oldukları çalışmada çok geçişli nesne tanıma algoritmalarından Faster R-CNN'i kullanmış ve CNN özellik tanımlayıcılı ImageNET üzerinde eğitilmiş toplam 3.235 ürün arasından her bir ürün kategorisini %52,16 ortalama tahmin doğruluğu ile tanımak için bir yöntem sunmuşlardır (Karlinsky vd, 2017).

Redmon J. ve Farhadi A. 2017 yılında yapmış oldukları çalışmalarında popüler nesne algılayıcı YOLO9000 ve DarkNET kullanarak 9000 nesne sınıfını algılayabilen bir yöntem önermişlerdir. Çalışmalarında kullandıkları YOLO9000 algılayıcısının milyonlarca görüntüyle eğitilmiş olmasının getirdiği işlem yükü bir tarafa bırakılacak olunursa YOLO9000'in sınıflandırma başarısının R-CNN'e yaklaşmış olması önemli bir gelişme olarak ifade edilebilir (Redmon ve Farhadi 2017).

Yi W. vd 2019 yılında yapmış oldukları çalışmalarında hazır olarak elde edilen 324 kategori ve her kategori için 5000 görselden oluşan ürün görsellerini kullanarak ResNeT50 temeline dayanan transfer öğrenme kullanmış, Faster R-CNN modelini eğitmişlerdir. Eğittikleri bu modeli kullanarak raflarda hazır olarak dizilmiş ürünler ve konveyördeki ürünler için bir dizi endüstriyel kamera (toplamda 9 kamera, yukarıdan aşağıya görünüm için 6, yan görünüm için 3 kamera) kullanmışlar ve elde ettikleri görüntülerle modeli test etmişlerdir. Çalışmada endüstriyel kamerayla çekilen görüntülerde %86 smart kamerayla çekilen görüntülerde %83 ortalama tahmin doğruluğuna ulaşılmıştır (Yi vd, 2019).

Tonioni A. vd 2019 yılında yapmış oldukları çalışmalarında market raflarında bulunan ürünlerin tanınması sorununa çözüm getirecek hızlı ve etkili bir yaklaşım önermişlerdir. Önerilen yaklaşım ürün tanıma sorununu üç adımda ele almıştır. Birinci adımda ürünleri tanımlamak için belirsiz nesne algılama, ikinci adımda K-NN benzerlik araması yoluyla tanıma ve üçüncü adımda ise performansı arttırmak için bir iyileştirmeyi kapsamaktadır. Hazırlanmış olan çalışmada 8400'den fazla raf ürünü içeren ve kullanıma açık olan sunulan Grocery Products veri kümesini kullanmışlardır. Önerdikleri modeli test etmek için daha önceden eğitilmiş bir YOLOv2 mimarisi kullanmışlar %76,93 ortalama tahmin doğruluğuyla ürün tespit işlemini gerçekleştirmişlerdir (Tonioni vd 2019).

Goldman E.vd 2019 yılında yapmış oldukları çalışmalarında süper market raflarındaki ürünlerin tespiti için Trax trafından kullanıma sunulan SKU110K veri kümesini kullanmış ve çalışmalarında ürün tespiti için iki yeni yaklaşım sunmuşlardır. Bu yaklaşımlardan birincisi, tahmin edilen nesneleri belirleyen sınırlayıcı kutular arasındaki örtüşmeyi tahmin etmek için bir Soft-IoU katmanıdır. İkincisi ise bu örtüşmelerin yaygın olduğu sıkıca paketlenmiş sahnelerde bile, sınırlayıcı kutu örtüşme belirsizliklerini çözmek için EM (Expectation Maximization) tabanlı bir birimdir. Yapmış oldukları çalışmada Faster R-CNN nesne algılama modelini kullanmış ve %83,4'lük ortalama tahmin doğruluğu sonucuna ulaşmışlardır (Goldman vd, 2019)

Kozlov A.'nın 2020 yılında Retail VISION tarafında organize edilen CVPR 2020 yarışması için hazırlamış olduğu çalışmasında süper market raflarında yoğun şekilde yerleştirilmiş raf görüntülerindeki ürünleri tespit etmek için trax tarafından kullanıma sunulan SKU110k veri kümesini kullanmıştır. Bu çalışmada yazar Faster R-CNN ve RetinaNET modellerini kullanmış ve %92,8'lik bir ortalama tahmin doğruluğuna ulaşmıştır (Kozlov 2020).

Yapılan bu çalışmanın amacı, literatürde kullanılan güncel nesne tespit yöntemlerinden YOLOv5 ve Faster R-CNN'i kullanarak yoğun şekilde yerleştirilmiş mağaza vitrinlerindeki ürünlerin tanınmasını otomatikleştirmek için seçilen model başarımlarının incelenmesi ve değerlendirilmesidir. Bu amaçla öncelikle ayrıntılı literatür araştırması yapılmış, incelenen yöntemler arasında öne çıkan bazı yöntemler uygulamalı olarak daha detaylı analiz edilmek üzere seçilmiştir. Çalışmanın temel dayanak noktası, güncel derin öğrenme tabanlı yöntemlerin sağladığı avantajlar ile birlikte nesne tespit çalışmalarının güncel problemlerin çözümü ile ilgili geldiği noktanın ortaya konulmasıdır. Bu problemin çözümü için Trax tarafından Amerika Birleşik Devletleri, Avrupa ve Doğu Asya'daki birçok süpermarket raflarından elde edilen toplam 11.762 raf görüntüsünden oluşan SKU110K veri kümesi kullanılmıştır.

Derin öğrenme tabanlı Perakende Ürün Tespit konusunda yapılan bu tez çalışması 4 kısımdan oluşmaktadır. Bu bölümlerin başlıkları ve içerikleri şu şekildedir.

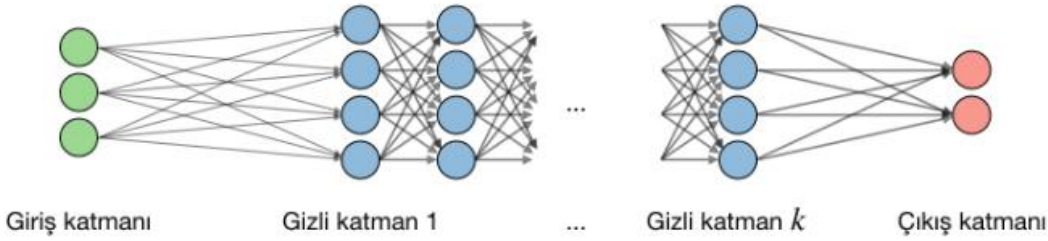
Çalışmanın birinci bölümünde yapay sinir ağları ve derin öğrenmenin temelleri ve gelişimi hakkında bilgi verilmektedir. İkinci bölümünde, Evrişimsel Sinir Ağlarının temel yapıları ve Evrişimsel Sinir Ağlarını oluşturan katmanlar ayrıntılı olarak anlatılmaktadır. Üçüncü bölümde, derin öğrenme tabanlı nesne tanıma anlatılmaktadır. Dördüncü ve son bölümde ise yapılan çalışmanın sonuçları ve başarım değerleri seçilen yönteminin avantaj ve dezavantajları ifade edilip, seçilen yöntemlerde ne gibi değişiklikler yapılarak yöntemlerin başarımlarının arttırılabileceği anlatılmıştır. Nesne tanıma modellerinden YOLOv5 ve Faster R-CNN kullanılarak Perakende Ürün Tespiti konusunda ihtiyaç duyulan kavramlara değinilmiş, çalışma süreci içerisinde yapılan model tasarım, optimizasyon, regularizasyon ve geliştirme süreçleri öncesinde ihtiyaç duyulan teorik altyapı kazanması sağlanmıştır.

BİRİNCİ BÖLÜM

1. YAPAY SİNİR AĞLARI VE DERİN ÖĞRENME

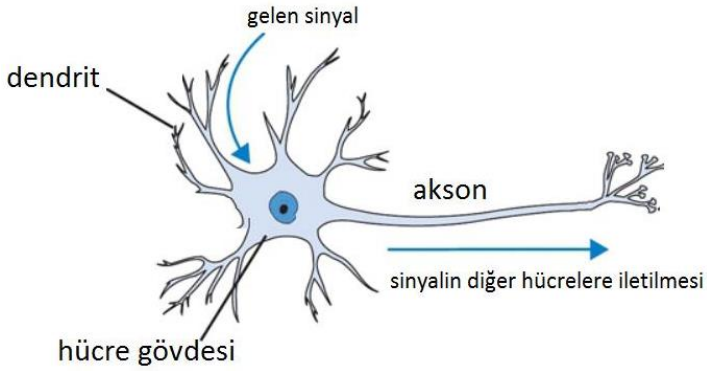
1.1. Yapay Sinir Ağları

Şekil 1'de basit olarak ifade edilen Yapay Sinir Ağları (YSA) giriş katmanı, gizli katman ve çıkış katmanı olarak isimlendirilen üç temel katmandan oluşan bir matematiksel algoritmadır.



Şekil 1. YSA'nın Yapısı (Amidi ve Amidi 2019)

Giriş katmanı, temel olarak girdi vektörleri olarak düşünülebilen sistem girdilerinden oluşur. Gizli katman nöron adı verilen birçok birimden oluşan, matematiksel işlemleri kullanarak nöronlardaki girdileri işler ve istenilen çıkışları üretir. İnsan beynine benzer bir yapıyla bu katmandaki nöron kendisinden önceki katmandan aldığı değeri işler ve bir sonraki katmana gönderir. Katmandaki nörona gelen ve nörondan gönderilen değerler nörona gelen değeri taşıyan kanalın ağırlık değerine bağlıdır. Ağırlık değeri ile önceki nörondan gelen değer çarpılarak kanalın ağırlık değeri bulunur. Şekil 2'de insan beyninde bulunan temel biyolojik sinir hücresinin yapısı gösterilmiştir.

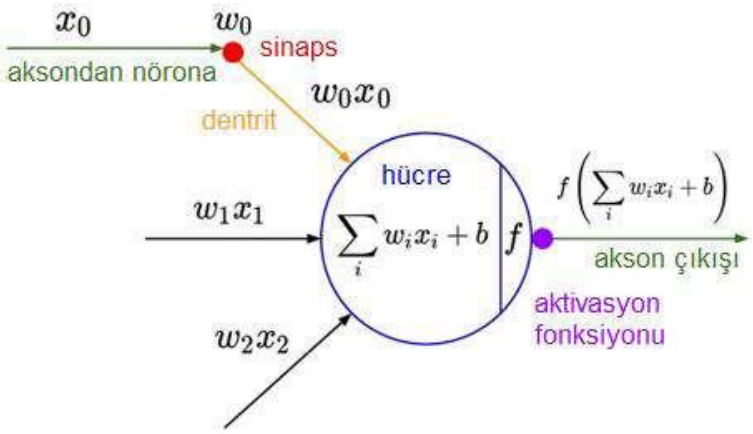


Şekil 1. YSA'nın Biyolojik Yapısı (Kızrak 2018)

Şekil 2'de görülen *dendrit'ler* aracılığı ile sinir hücresine önceki sinir hücrelerinden gelen girdiler hücrenin gövde kısmına iletilir. Sinir hücresinde kararlılık halinin bozulmasıyla hücre içerisinde kimyasal bir süreç başlamış olur. Bu süreçle beraber aksonlar ile girdiler bir sonraki sinir hücresine iletilir. Akson ucu ile bir sonraki hücrenin *dendrit'i* arasındaki bölüm sinaptik boşluk olarak ifade edilir. Bilginin iletilmesini sağlayan iletken madde sinaptik'e dolar. Böylece bilgi bir sinir hücresinden diğerine geçmiş olur. İşte öğrenme süreci sinir hücreleri arasındaki sinaptik ilişki yerine yeni bir ilişki kurulması olarak ifade edilir.

YSA'nın matematiksel modeline bakacak olursak YSA biyolojik sinir hücresine benzer şekilde çalışmaktadır. Şekil 3'te gösterilmiş olan, w_1x_1 , w_2x_2 ... w_nx_n , kendisinden önceki sinir hücresinden gelen veriyi temsil etmektedir. Yapay sinir hücresi aldığı veriyi seçilmiş bir aktivasyon fonksiyonuna tabii tutup çıktı hesaplamakta ve bu sonucu kendisinden sonraki sinir hücrelerine iletmektedir. Yapay sinir hücreleri biyolojik sinir hücrelerinin birebir kopyası değildir. Yapay sinir hücreleri biyolojik sinir hücrelerinden ilham alınarak tasarlanmış yapılardır.

Şekil 3'te gösterilen yapıda w_ix_i 'nin bir matris çarpımı olduğu unutulmamalıdır. Elde edilen bu çarpım sonucuna Bias değeri eklenerek elde edilen yeni sonuç önceden belirlenen aktivasyon fonksiyonuna tabii tutulur.



Şekil 2. Yapay Sinir Ağı'nın Matematiksel Modeli (Kızrak 2018)

Buradan elde edilen hesaplama sonucuna göre hangi yapay sinir hücresi daha baskınsa model o sinir hücresine göre tahminde bulunur. Buraya kadarki işlemler ileri besleme (forward- propagation) işlemi olarak ifade edilir. Modelin eğitimi yapılırken bu işlem sürekli tekrar edilerek en uygun değere ulaşılmaya çalışılır. İleri besleme işlemi tersine çevrilerek katmandaki nöronlara ait ağırlık değerleri geriye doğru beslenerek güncellenir. Buradaki güncelleme sırasında geriye doğru (back-propagation) türev işlemi ileriye doğru matris çarpımı işlemi yapılmaktadır.

YSA'nın i. sırasındaki katmana i ve katmanın j. sırasındaki gizli birime j dersek, elimizde:

Denklem 1 Nöronun sonraki katmana etkisi (Amidi A. ve Amidi, S. 2019)

$$z_j^{[i]} = w_j^{[i]^T} x + b_j^{[i]} \tag{1}$$

Matematiksel denklemi olacaktır. Bu denklemde *w* nöronun ağırlığını, *b* bias değerini (eğilim) ve *z* elde edilen sonucu ifade etmektedir. Her bir nöron için *z* değerleri hesaplanarak:

Denklem 2 Aktivasyon Fonksiyonu (Amidi A. ve Amidi, S. 2019)

$$\alpha = \sigma(z) \tag{2}$$

Denklem 2'de belirtilen aktivasyon fonksiyonuna tabii tutulur. Ve çıkış katmanına iletilir. En son çıkış katmanına iletilmeden önce hesaplanan *a* değeri:

Denklem 3 Kayıp Fonksiyonu (Amidi A. ve Amidi, S. 2019)

$$\mathcal{L}(\alpha, \gamma) \tag{3}$$

İfadesindeki *y* değeriyle karşılaştırılır. Buradaki *y* değeri ulaşılmak istenen değer *a* ise hesaplanan değerdir. İşlem sonucunda *a* değerinin *y* değerine maximum yakınlıkta olması beklenir. Eğer kayıp fonksiyonu yeterince küçük değilse bu defa geriye yayılım işlemiyle tüm nöronlardaki ağırlıklar yeniden güncellenir.

İleri yönlü işlemleri matematiksel olarak ifade edecek olursak her nöron için;

Denklem 4 İleri yönlü yayılım işlemi (Amidi A. ve Amidi, S. 2019)

$$z = w^T x + b \rightarrow \alpha = \sigma(z) \rightarrow \mathcal{L}(\alpha, \gamma) \tag{4}$$

denklem ifadesiyle özetleyebiliriz.

Geri yayılım YSA'daki ağırlıkları güncellemek için kullanılan ve bunu yaparken de asıl sonuç ile istenilen sonucu hesaba katan bir yöntemdir. Ağırlık *w* değerine göre türev, zincir kuralı kullanılarak hesaplanır. Bu hesaplama;

Denklem 5 Geriye Yayılım İşlemi - Zincir Kuralı (Amidi A. ve Amidi, S. 2019)

$$\frac{\partial L(\mathcal{Z}, \mathcal{Y})}{\partial w} = \frac{\partial L(\mathcal{Z}, \mathcal{Y})}{\partial a} \times \frac{\partial a}{\partial z} \times \frac{\partial z}{\partial w} \tag{5}$$

Matematiksel denklemi kullanılarak yapılır. Bu işlem sonucunda ağırlıkların güncellenmesi Denklem 6'daki gibi yapılır.

Denklem 6 Ağırlıkların Güncellenmesi (Amidi A. ve Amidi, S. 2019)

$$w \leftarrow w - \alpha \frac{\partial L(\mathcal{Z}, \mathcal{Y})}{\partial w} \tag{6}$$

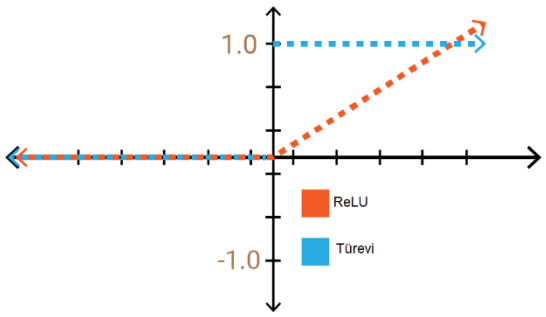
Yapılacak işlemleri özetleyecek olursak;

1. Adım: Eğitim veri kümesi belirlenir.
2. Adım: Aktivasyon fonksiyonu belirlenir.
3. Adım: Kayıp fonksiyonunu elde etmek için, ileri yayılım gerçekleştirilir.
4. Adım: Gradyanları elde etmek için hesaplanan kayba geri yayılım işlemi uygulanır.
5. Adım: YSA'nın ağırlıklarını güncellemek için gradyanlar kullanılır.

1.1.1. Aktivasyon Fonksiyonları

Aktivasyon fonksiyonu YSA'lar eğitilirken kullanılan ve modele lineer olmayan karmaşıklıklar eklemek için kullanılan yapılardır. Aktivasyon fonksiyonlarının amacı, *w* (ağırlık) ve *b* (Bias) değerlerini ayarlayarak, nörona gelen giriş sinyalini işleme tabi tutarak sonraki nöronda kullanılacak çıkış sinyali haline getirmektir. Elde edilen bu çıkış sinyali aynı zamanda bir sonraki nöronun giriş sinyalidir. Uygulamada en sık kullanılan aktivasyon fonksiyonları, RELU, Leaky RELU, Sigmoid, TanH, SoftMax, SWISH tir.

Relu: Şekil 4'te gösterilen **ReLU (*Doğrultulmuş doğrusal üniteler*) fonksiyonu** ilk olarak 2000'li yıllarda Hahnloser ve arkadaşları tarafından biyolojik kavramların matematiksel ispatları için önerilmiştir. 2010 yılından sonra Nair ve Hinton tarafından sınırlı Boltzmann makinasında kullanmasıyla bu fonksiyon popülerleşmiştir. Fonksiyonun genel formülü ***g(z)=max(0,z)***'dir. Fonksiyon, hesaplanan değer pozitif mi negatif mi diye bakar. Eğer değer negatifse çıkışa 0 değerini aktarır. Eğer değer pozitifse, değer fonksiyondan olduğu gibi geçer. Hesaplama yükünün sigmoid ve hiperbolik tanjant fonksiyonlarına göre az olması çok katmanlı ağlarda daha çok tercih edilmesine sebep olmuştur. İşlem hızı fala olmasına rağmen en büyük dezavantajı sıfır değer bölgesinin türevinin de sıfır olmasıdır yani öğrenmenin o bölgede gerçekleşmemesidir.



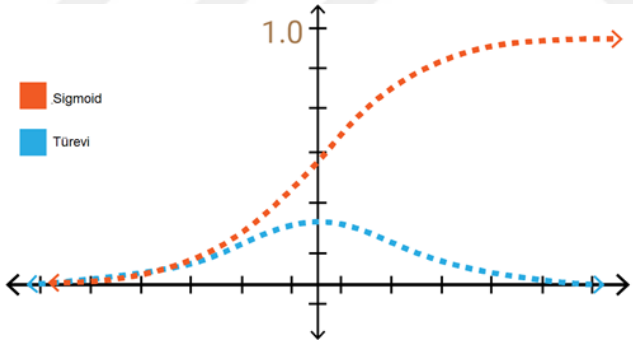
Şekil 3. RELU Aktivasyon Fonksiyonu (Kızrak 2019)

Sigmoid: Şekil 5'te gösterilen **Sigmoid fonksiyonu** sigmoid eğrisine sahip matematiksel bir fonksiyondur. Bu fonksiyon, 0 ile 1 arasında tanımlı olan, ***g(z)=1/1+e−z*** genel formülüyle ifade edilen ve 'S' şeklinde bir eğriye sahip olan fonksiyondur. Şekil 5'i inceleyecek olursak x, -2 ile +2 arasında iken y değerleri hızlı

şekilde değişir. X'te yapılan küçük değişimler y'de büyük olacaktır. Bu iyi bir sınıflayıcı olarak kullanılabileceği anlamına gelir. Bu fonksiyonun bir diğer avantajı da doğrusal fonksiyonda olduğu gibi (-sonsuz, +sonsuz) ile karşılaşıldığında her zaman (0,1) aralığında değer üretir. Yani aktivasyon değeri kaybolmaz. Sigmoid fonksiyonu anlamak ve uygulamak bakımında en sık kullanılan aktivasyon fonksiyonu olmakla beraber bazı dezavantajları vardır.

Bunlardan ilki Fonksiyonun uçlarına doğru grafiğe dikkatlice bakacak olursak, **Vanishing** (*gradyanların ölmesi/kaybolması*) problemi, yani y değerleri x'teki değişikliklere çok az tepki vermektedir. Bu bölgelerde türev değerleri çok küçük olur ve 0'a yakınsar ve öğrenme olayı minimum düzeyde gerçekleşir. Yavaş bir öğrenme olayı gerçekleştiğinde hatayı minimize eden optimizasyon algoritması yerel (lokal) minimum değerlere takılabilir ve yapay sinir ağı modelinden alınabilecek maksimum performans alınamaz

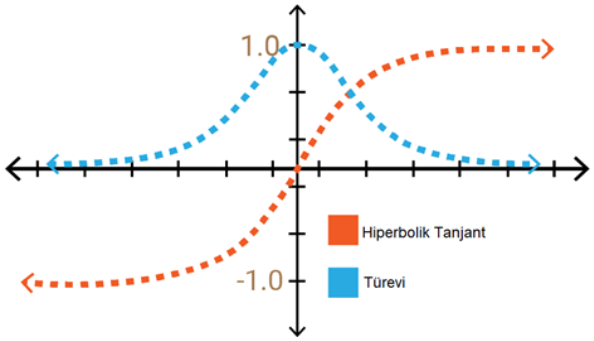
İkincisi ise Sigmoid çıkışları sıfır merkezli değildir. YSA'da daha katmanlarındaki nöronların sıfır merkezli olmayan verileri alabileceğinden sonuçta istenmeyen bir durum ortaya çıkabilir.



Şekil 4. Sigmoid Aktivasyon Fonksiyonu (Kızrak 2019)

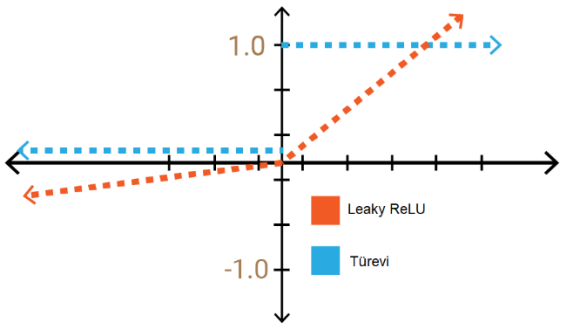
Şekil 6'da gösterilen **TanH fonksiyonu:** hiperbolik tanjant fonksiyonu olarak bilinir. Fonksiyon $g(z)=e^{z}-e^{-z}/e^{z}+e^{-z}$ genel formülüyle ifade edilir. Hiperbolik tanjant fonksiyonu Sigmoid fonksiyonuna çok benzer bir yapıya sahiptir. Ancak fonksiyonun aralığı (-1,+1) olarak tanımlanmaktadır. Sigmoid fonksiyonuna göre en önemli avantajı türevinin daha dik olması yani daha çok değer alabilmesidir. Daha hızlı öğrenme ve sınıflama işlemi için daha geniş aralığa sahip olması sigmoide göre daha

verimli olacağı anlamına gelmektedir. Bununla birlikte fonksiyon uçlarındaki vanishing problemi devam etmektedir.



Şekil 5. Tanh Aktivasyon Fonksiyonu (Kızrak 2019)

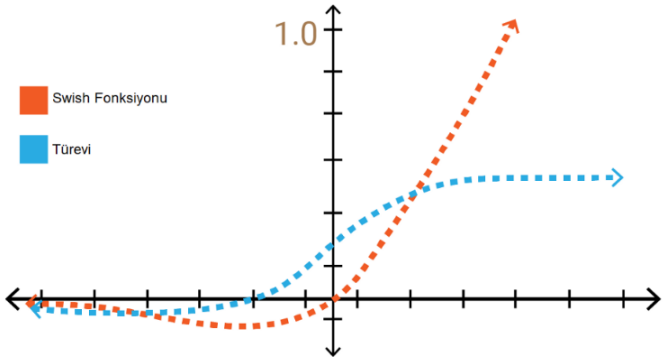
Şekil 7'da gösterilen **Sızıntı (Leaky) ReLU Fonksiyonu** RELU fonksiyonundaki ölü gradyanların yaşatılması prensibine göre geliştirilmiş bir fonksiyondur. Sızıntı Relu **g(z) = max (ϵz,z) , with ϵ≪1** genel formülüyle ifade edilir. Buradaki sızıntı negatif bölgede ve 0,01 olarak verilmelidir. Sızdırılan ReLU'nun tanım aralığı eksi sonsuza doğru devam etmektedir. Buradaki 0'a yakın ama 0 olmayan değer sayesinde ReLU'daki kayıp gradyanlar kazanılmış yani öğrenme negatif bölgedeki değerler için de sağlanmış olunur.



Şekil 6. Leaky ReLU Aktivasyon Fonksiyonu (Kızrak 2019)

SoftMax fonksiyonu, Sigmoid fonksiyonuna çok benzer bir yapıya sahiptir. En önemli farkı sigmoid fonksiyonu gibi ikiden fazla sınıflamak gereken durumlarda özellikle derin öğrenme modellerinin çıkış katmanında tercih edilmektedir. Fonksiyon $g(z)_j = e^{z_j} / \sum_{k=1}^{K} e^{zk}$ genel formülüyle ifade edilir. Aynı Sigmoid'te olduğu gibi sınıflayıcı olarak kullanıldığında oldukça iyi bir performans sergiler. Olasılıksal bir yorumlama gerçekleştirerek, girdinin belirli sınıfa ait olma olasılığını 0–1 aralığında değerler üreterek belirlenmesini sağlamaktadır.

Şekil 8'de gösterilen **Swish (A Self-Gated/Kendinden Geçitli) Fonksiyonu** ReLU aktivasyon fonksiyonuna oldukça benzer bir yapıdadır. Fonksiyon $g(x)=x*1/1+e^{-x}$ formülüyle ifade edilir.



Şekil 7. Swish Aktivasyon Fonksiyonu (Kızrak 2019)

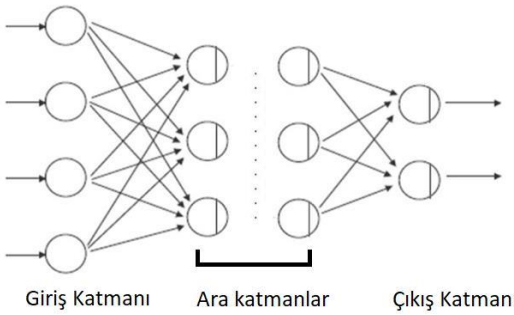
ReLU'dan en önemli farkı negatif bölgede değer alabiliyor olmasıdır. ReLU'da olduğu gibi 0'dan büyük değerler doğrudan çıkışa aktarılırken, 0'dan küçük değerlerde sıfırdan farklı ama sıfıra çok yakın değerler almaktadır. Pascanu ve ark. (2012) göre kaybolan eğim kayıp gradient problemini önlemektedir. ImageNET mimarisinde doğruluk ve hızda kazanç sağlaması sebebiyle Nwankpa ve ark. (2018) RELU ya alternatif olarak swish aktivasyon fonksiyonunu önermiştir.

1.1.2. Kayıp Fonksiyonu (Loss Function)

Kayıp fonksiyonu bilinen diğer adıyla maliyet fonksiyonu modelimizin tahmin ettiği değer ile gerçekte olması gereken değer arasındaki farkı hesaplayarak hatanın büyüklüğünü gösteren fonksiyondur. Tasarlanan modelin eğitim sonucunda kayıp değerinin sıfıra yaklaşması amaçlanır. Kayıp fonksiyonu, eğitilmiş modele verilen girdi sonucunda modelin yaptığı tahminin ne gerçeğe ne kadar uzak olduğunun matematiksel olarak hesaplanmasıdır. Sonuçta kayıp fonksiyonu sayesinde tasarımcı tarafından model hatası görülür ve optimizasyon (iyileştirme) ile model başarımı arttırılmaya çalışılır.

1.1.3. İleri Yayılım ve Geri Yayılım Algoritması

İleri yayılım algoritması (forward- propagation) temelde, bilginin sinir ağı boyunca, ilk olarak ağın giriş düğümlerine, sonra gizli katman / katmanlara ve son olarak da çıkış düğümlerine ilerleme mimarisi üzerine kurulmuştur. Şekil 9'da çok katmanlı ileri beslemeli bir sinir ağı örneği gösterilmiştir.

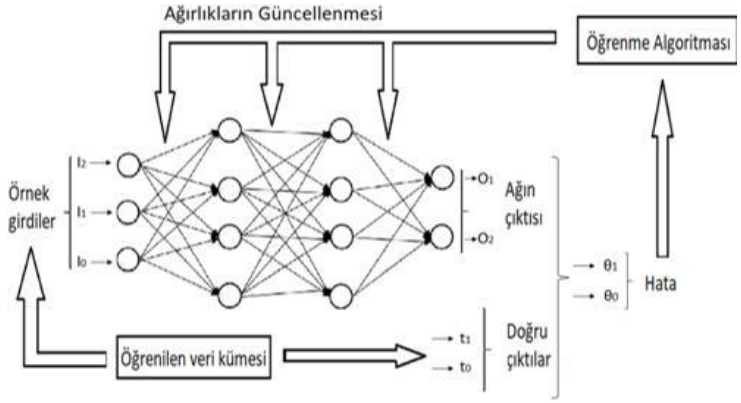


Şekil 8. İleri Beslemeli YSA (Mete 2019)

Algoritma temelde iki adımda çalışır: Birinci adımda ağa verilen girdiler için ağırlıklar toplamını hesaplar ve ikinci adımda hesaplanan toplama göre normalleştirme yapmak için bir aktivasyon işlemini gerçekleştirir. İleri beslemeli bir sinir ağının sadece tek yönlü bilgi akışına izin verdiği unutulmamalıdır.

İleri beslemeli olarak modellenmiş bir YSA'da, katmanlardaki çıkışlar kendilerinde sonra gelen katmanlara hesaplanan ağırlıklar üzerinden girdi olarak aktarılır. Modele verilen giriş verisi, gizli katmanlarda ve çıkış katmanında işlenerek sonuca ulaşılmaya çalışılır.

Geri yayılımda ise temel amaç YSA'daki her nöronun kayıp fonksiyonuna ne kadar etkisinin olduğunu hesaplamaktır. Geri yayılım işleminde her katmandaki ağırlık ve Bias değerleri yeniden düzenlenir. Modelin eğitimi sırasında her nöronun kayıp fonksiyonuna etkisi minimize edilene kadar bu işlem tekrar edilir. Şekil 10'da üç katmanlı, elde edilen çıkışların giriş katmanına geri besleme yapıldığı bir YSA'nın yapısı görülmektedir.



Şekil 9. Geri Beslemeli Sinir Ağı (Mete 2019)

1.2. Derin Öğrenmenin Tanım ve Tarihçe

Derin öğrenme, sistematik olarak yapay zekâ ve makine öğrenmesinin en son basamağında yer almaktadır ve yapay zekâ uygulamalarında en popüler olan yaklaşımdır. Derin öğrenme, yapay sinir ağları ve insan beyninin işlevlerini taklit eden hesaplama sistemleri kavramına dayanır.

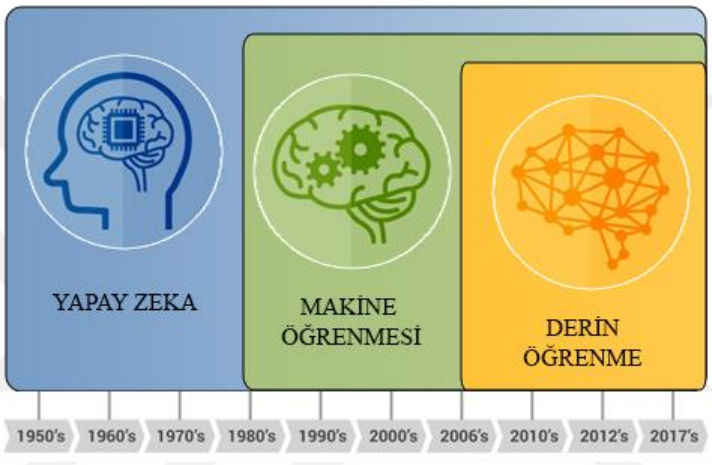
Derin öğrenme kavramını kronolojik olarak inceleyecek olursak, Warren McCulloch ve Walter Pitts'in 1943 yılında matematik ve sinir ağı mantığına dayanan algoritmalarla beynin düşünme aşamalarını taklit eden yapıyı oluşturmaları başlangıç noktası olarak söylenebilir (McCulloch ve Pitts 1943). Frank Rosenblatt 1958 yılında, basit toplama ve çıkarma işlemleri kullanılarak iki katmanlı bir bilgisayar yapay sinir ağına dayalı, denetimli öğrenmeli bir desen tanıma algoritması olan perceptron oluşturuldu (Haykin 2009). 1965 yılında Ivakhnenko ve Lapa, Sibernetik Tahmin Araçları isimli çalışmalarında, karmaşık denklemlerin aktivasyon fonksiyonlarına sahip modelleri kullanmışlardır (Ivakhnenko ve Lapa 1965).

1980'li yıllara gelindiğinde Kunihiko Fukushima tarafından, Neocognitron önerilmiştir. Neocognitron el yazısı ve diğer desenleri tanıma problemleri için kullanılan çok katmanlı ve hiyerarşik yapay sinir ağı yapısıdır. Bu modelde bilgisayarların öğrenme işlemini, görsel örüntüler kullanılarak yapabileceği önerilmiştir (Fukushima 1988). Juyang Weng ve arkadaşları 1992 yılında, karma sahnelerden otomatik olarak 3 boyutlu nesne tanıma işlemini gerçekleştiren Cresceptron yöntemini yayımladılar (Weng, Cohan ve Herniou 1992). 1997'de Hochreiter ve Schmidhuber tarafından LSTM (Uzun Kısa Süreli Hafıza) uzun vadeli bağımlılık problemlerinin çözümü için tekrarlayan bir sinir ağı önerilmiştir (Hochreiter ve Schmidhuber 1997). 2000'li yıllara gelindiğinde, Hinton ve Salakhutdinov'un yaptıkları bir çalışma sonrasında "Derin Öğrenme" kavramı araştırmacılar için daha da ilgi çekici bir konu olmaya başlamıştır. Bununla beraber çok katmanlı bir sinir ağının, bir kerede bir katmanının nasıl önceden eğitebileceği gösterilmiştir (Hinton ve Salakhutdinov 2006).

2014 yılında Facebook tarafından geliştirilen ve DeepFace adı verilen kullanıcıların resimler üzerinde otomatik olarak etiketlenmesini sağlayan bir derin öğrenme teknolojisi kullanıma açılmıştır. Google firması tarafından geliştirilen ve

DeepMind algoritması olan AlphaGo 2016 yılında Go oyununda, milyonlarca simülasyon kullanılarak eğitilmiş ve profesyonel Go oyuncusu Lee Sedol'u turnuvada 4-1 yenmiştir (Guardian 2016).

1950 yılının başlarında gelişmeye başlayan yapay zekâ teknolojisi, 1980 yılının başında yerini makine öğrenmesine bırakmış ve 2010 yılının başında ise derin öğrenme ile makine öğrenmesinin eksiklikleri giderilmeye çalışılmıştır. Yapay zekâ ile başlayan öğrenme ve yorumlama algoritmalarının kronolojik değişimi Şekil 11'de verilmiştir.



Şekil 10. Yapay Zekâ İle Başlayan Öğrenme ve Yorumlama Algoritmalarının Kronolojik Değişimi (Kayaalp ve Süzen 2018)

Şu ana kadar yapılan çalışmalarda derin öğrenme farklı kaynaklarda değişik şekillerde tanımlanmıştır. Gu X ve arkadaşları derin öğrenmeyi; bilgisayarların, deneyimlerden öğrenmelerini ve dünyayı kavramların hiyerarşisi açısından anlamalarını sağlayan bir makine öğrenimi olarak tanımlamıştır (Gu, Zhang ve Kim 2016).

Başka bir çalışmada Deng L. Yu derin öğrenmeyi; denetimli veya denetimsiz özellik çıkarma, dönüştürme, desen analizi ve sınıflandırma için birçok doğrusal olmayan gizli katmandan yararlanan bir makine öğrenme teknikleri sınıfı olarak tanımlamıştır (Deng ve Yu 2014).

Başka bir çalışmada Najafabadi ve arkadaşları derin öğrenme; insan beyninin son derece karmaşık problemler için gözlemleme, analiz etme, öğrenme ve karar verme yeteneğini taklit etmeyi amaçlayan, büyük miktarda denetimsiz veri kullanan

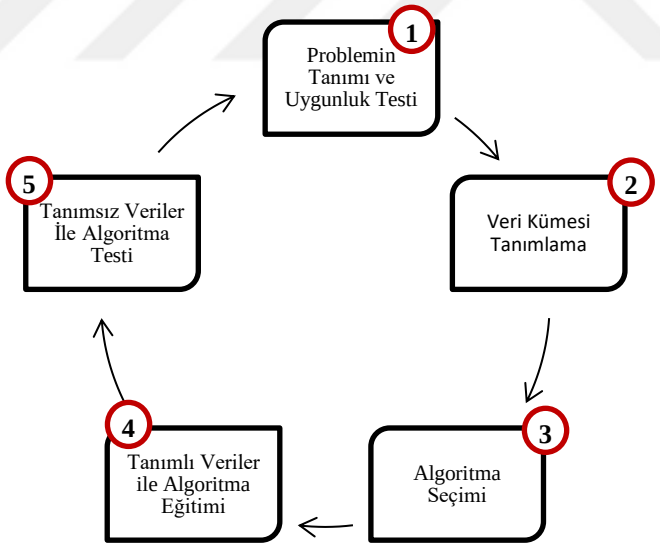
bir makine öğrenmesi olarak tanımlanmıştır (Najafabadi, Villanustre, Khoshgoftaar, Seliya, Wald ve Muharemagic 2015).

Farklı kaynaklardan alınan derin öğrenme tanımlarından yola çıkarak derin öğrenme; insan beyninin problemleri çözebilmek için ihtiyaç duyduğu gözlem yapma, analiz etme, öğrenme ve karar yeteneklerini taklit ederek, denetimli yâda denetimsiz biçimde öznitelik çıkarma, sınıflandırma ve dönüştürme işlemlerini yeterli miktardaki verileri kullanarak yapabilen bir makine öğrenmesi tekniği olarak ifade edilebilir.

Araştırmacılar tarafından Yapay zekâ uygulamalarının en çok ilgi duyulan yaklaşımı olan derin öğrenme sistematikte makine öğrenmesinin en alt seviyesinde yer almaktadır.

1.3. Derin Öğrenme Süreci

Literatürde derin öğrenme algoritmalarının uygulamanın karmaşıklığına göre değişken bir modele sahip olması gerekir. Sonuçta elde edilmek istenen başarı oranı en yüksek seviyede gerçekleşene kadar derin öğrenme süreci tekrar eder.



Şekil 11. Derin Öğrenme Süreci (Kayaalp ve Süzen 2018)

Şekil 12 ‘de süreç tamamlanana kadar verilerin geçmesi gereken genel adımlar gösterilmiştir. Şekil 12‘de de görüldüğü üzere derin öğrenme süreci beş temel adımda gerçekleşmektedir. Sürecin ilk adımı problemin tanımı ve uygunluk testidir. Bu adımda çözümü amaçlanan problem doğru bir şekilde tanımlanmalı ve derin öğrenme

teknikleri kullanılarak çözülüp çözülemeyeceği net bir biçimde ifade edilmektedir. Sürecin ikinci adımında tanımlanmış olan problemin çözümü ile ilgili ihitiyaç duyulan veri kümesi oluşturulmaktadır. Tanımlanmış ve veri kümesi oluşturulmuş problemin derin öğrenme yöntemleri kullanılarak çözülebilmesi için hangi algoritmanın kullanılacağının seçildiği adım sürecin üçüncü adımında gerçekleşmektedir. Sürecin dördüncü adımında daha önceden hazırlanmış veri kümesinin belirli bir miktarının, problemin çözümü için belirlenmiş algoritma kullanılarak derin öğrenme modelinin eğitimi için kullanıldığı adımdır. Öğrenme sürecinin son adımı olan beşinci adımında ise seçilen algoritmanın problemin çözümü için test edildiği adım olarak ifade edilir. Bu adımda test için kullanılan veri kümesinin eğitim sırasında ağa gösterilmemiş olması ağın başarımını ölçmek için son derece önemlidir.

1.4. Derin Öğrenme Mimarileri

YSA'da katman sayısı arttırılarak / azaltılarak, katmanlarda kullanılan neuron yapısı değiştirilerek farklı türde derin öğrenme mimarileri oluşturulabilmektedir. Araştırmalarda sıklıkla kullanılan derin öğrenme mimarileri şu şekilde ifade edilebilir;

1.4.1. Çok Katmanlı Derin Sinir Ağları

Derin öğrenme süreçlerinde araştırmacılar, insan beyninin bilgi işleme yönteminden faydalanılarak geliştirilen YSA'nın çok katmanlı halini kullanmışlardır. Doğrusal problemlerin çözümünde sıklıkla kullanılan tek katmanlı algılayıcılar (single layer perceptron) araştırmacıların doğrusal olmayan problemlerin çözümü için yetersiz kalmıştır. Bunun üzerine Fernandez ve ark. tarafından 2006 yılında çok katmanlı algılayıcılar (multi-layerperceptron) geliştirilmiştir (Fernandez vd, 2006). Temelde çok katmanlı bir yapay sinir ağı karmaşık problemlerin çözümü içim ; girdi katmanı, gizli (hidden) katmanlar ve çıktı katmanlarından oluşmaktadır.

1.4.2. Konvolüsyonel Sinir Ağları (Convolutional Neural Network-CNN)

İnsanların görme sistemini örnek alan Konvolüsyonel sinir ağları ile yapay sistemlerde, nesnelerin algılanması, tanımlanması ve sınıflandırılması amaçlanmıştır (Le vd. 2009). Konvolüsyonel sinir ağları kullanılarak nesne tanımlama, görüntü sınıflandırma ve segmentasyonu gibi işlemlerde başarılı sonuçlar elde edilmektedir.

1.4.3. Tekrarlayan Sinir Ağları RNN LSTM ve GRU

RNN'ler prensipte sıralı bir şekilde gelişen olayların birbiriyle anlamlandırarak gizli katmandan çıkan değeri yine aynı gizli katmana giriş olarak kullanabilen, bir çeşit rekürsif (kendi kendini çağıran) fonksiyon gibi çalışabilen bir derin öğrenme mimarisidir. Özellikle doğal dil işleme konusunda başarılı sonuçlar elde eden bir mimaridir.

RNN'nin gelişmiş bir türü olarak ifade edilen Uzun Kısa Süreli Hafıza Ağları (LSTM) ise 1997 yılında Hochreiter ve Schmidhuber tarafından literatüre kazandırılmıştır (Hochreiter ve Schmidhuber 1997). Özellikle derin öğrenme kullanılarak yapılan görüntü işleme konusunda zaman ve olayların değişkenlik gösterdiği uygulamalarda sıklıkla kullanılmaktadır. Resimlerden otomatik başlık çıkarma, ilişkili metinlerden kelime üretme, el yazısının tespit edilmesi sıklıkla kullanıldığı alanlardır.

Uzun Kısa Süreli Belleğin (LSTM) biraz daha basitleştirilmiş hali olarak kabul edilen Geçitli Tekrarlayan Birimler (GRU), tekrarlayan sinir ağlarında (RNN) bir geçiş mekanizması olarak kabul edilmektedir. GRU, tekrarlayan sinir ağlarında ortaya çıkan kaybolan gradyan problemini ele alan bir mimariye sahiptir. GRU temelde, güncelleme kapısı (LTSM'deki unutma ve giriş kapılarını birleştirilmiş hali) ve ilave olarak bir sıfırlama kapısından oluşmaktadır. GRU, yapı olarak LSTM modellerine benzese de değindiğimiz gibi daha basittir ve giderek daha popüler hale gelmektedir (Rana 2016).

1.4.4. Sınırlı Boltzman Makineleri (Restricted Boltzmann Machines - RBM)

Sınırlı Boltzman Makineleri (RBM) iki katmanlı, kullanılan veri setindeki olasılık dağılımlarını öğrenebilen rastlantısal bir YSA'dır. RBM mimarisindeki ilk katman giriş katmanı ikinci katman ise gizli katman olarak isimlendirilir. RBM'ler, kümeleme (Larochelle ve Bengio 2008), özellik öğrenimi (Coates, Ng ve Lee 2011), boyut indirgeme (Hinton ve Salakhutdinov 2006), işbirlikçi filtreleme (Salakhutdinov, Mnih ve Hinton 2007) ve konu modelleme (Hinton ve Salakhutdinov 2009) gibi farklı uygulamlarda kullanılan derin öğrenme mimarisidir.

1.4.5. Çekişmeli Üretici Ağlar (GAN)

Gerçekçi foto görüntü üretme, metinden görüntü üretme (Reed vd, 2016), çözünürlük artırma, video tamamlama (Mathieu, Couprie ve Lecun 2015) gibi görüntü işleme uygulamalarında sıklıkla kullanılan Çekişmeli Üretici Ağları ilk olarak 2014 yılından Goodfellow ve arkadaşları tarafından NIPS konferansında tanıtılmıştır (Goodfellow 2014). Temelde GAN'lar birbiriyle rekabet eden iki yapay sinir ağından oluşmaktadır. Bunlar Üretici (Generator - G) ve Ayırt Edici (Discriminator - D) ağ olarak adlandırılır. Üretici ağ, gerçeğe benzeyen yeni veriler (resimler, sesler, modeller vb.) üretirken Ayırt Edici ağ ise sahte ve gerçek verileri birbirinden ayırt etmeye çalışmaktadır.

1.5. Derin Öğrenmede Kulanılan Kavramlar

Tüm sürecin model tasarımcısı tarafından yönetildiği Hiper-parametre kavramı modelde kullanılacak veri seti boyutu, katman sayısı, nöron sayısı, aktivasyon fonksiyonu, optimizasyon fonksiyonu, öğrenme katsayısı gibi tercihlere verilen genel isimdir.

1.5.1. Veri Setinin Boyutu

Veri setinin boyutu ve çeşitliliği derin öğrenme için en temel faktördür. Çünkü Veri seti boyut ve çeşitlilik bakımından büyüdükçe öğrenme oranın da doğru orantılı olarak arttığı gözlemlenmiştir. Özellikle eğitimin sürekli yapılmadığı ve depolama sorunu olmayan uygulamalarda veri seti boyutu önemsenmez. Model eğitiminin sürekli artması demek sonsuza kadar süreceği anlamına gelmez. Bir süre sonra eğitim oranı çok küçük artışlar gösterir. Bu durum eğitimin tamamlanması olarak kabul edilebilir.

1.5.2. Katman Sayısı

Derin öğrenmenin YSA'ya göre başarımının artmasını sağlayan özellik derinlik kavramını ortaya çıkaran katman sayısıdır. Katman sayısının artması her ne kadar öğrenmeyi arttırsa da belli katmandan sonra geri besleme etkisi ilk katmanlara daha az ulaştığından, bu katmanlardan sonraki katmanlar eğitim başarımına etki etmemektedir.

1.5.3. Nöron Sayısı

Derin öğrenme uygulamalarında kullanılan her bir nöron hafızada tutulacak bilgi sayısını ifade eder. Bu sayının fazla olması model için fazladan depolama alanı ve hesaplama zamanına, az olması ise modelin yetersiz uyumuna sebep olmaktadır.

1.5.4. Optimizasyon Algoritması Seçimi

Derin öğrenmede öğrenme işlemi temelde bir optimizasyon problemi mantığını barındırır. Optimizasyon algoritmaları, yitim fonksiyonunu (loss function) minimize etmek için nöronlardaki ağırlıkları güncelleyen algoritmalardır. En çok kullanılan optimizasyon (eniyileme) algoritmaları, Rasgele Gradyan İnişi (Stochastic Gradient Descent - SGD), Uyarlamalı Gradyan (Adaptive Gradient-Adagrad), Ortalama Karekökü Yayılımı (Root Mean Square Propagation - Rmsprop), Uyarlamalı Delta (Adaptive Delta - Adadelta) ve Uyarlamalı Momentum (Adaptive Moment - Adam) olarak ifade edilebilir.

1.5.5. Eğitim Tur (Epoch) Sayısı

Modelin eğitimi sırasındaki gerçekleşen her adıma epoch ismi verilir. Tasarımcı tarafından belirlenen Epoch değeri çözülecek probleme göre farklılık gösterir. Epoch sayısı yükseldikçe modelin eğitim başarım oranı da artmaktadır. Eğtim sırasında belirli bir epoch değerine ulaşıldığında öğrenme başarımında küçük artışlar gözlenmeye başlanır ve bu aşamadan sonra doğru olan eğitimin sonlandırılmasıdır.

1.6. Derin Öğrenme Kütüphaneleri

Araştırmacıların derin öğrenme algoritmalarında kullanabilecekleri, Caffe, Torch, Theano, TensorFlow, Keras gibi birçok popüler yazılım ve kütüphane mevcuttur. Bu kütüphanelere temel çerçevede bakacak olursak ;

Theano : Theano kütüphanesi, matematiksel ifadeleri etkin bir biçimde tanımlayan, değerlendiren ve optimize eden bir açık kaynak kodlu Python kütüphanesidir. Bu kütüphane yine Python’a ait olan NumPy kütüphanesiyle entegre biçimde çalıştığı için yoğun matematiksel işlemleri kolay ve hızlı bir şekilde gerçekleştirebilmektedir.

Tensorflow: Goggle tarafından derin sinir ağları ve makine öğrenmesi konularında araştırma yapmak için geliştirilen açık kaynak kodlu bir derin öğrenme kütüphanesi olan Tensorflow, sayısal hesaplamaları yaparken veri akış grafiklerini kullanmaktadır. Esnek bir mimariye sahip olan TensorFlow, yapılmak istenen hesaplamayı tek bir API (Uygulama Programlama Arabirimi) ile farklı cihazlardaki bir veya daha fazla CPU'ya veya GPU'ya dağıtılmaya imkân sağlar.

Caffe: Caffe derin öğrenme çatısı Yangqing Jia tarafından oluşturulmuş olup, bir insanın beyin aktivitelerinde olduğu gibi hızlı ve modüler bir yapıya sahiptir. Verileri işlemedeki hızından dolayı endüstriyel ve akademik araştırma uygulamalarında araştırmacılar tarafından tercih edilir. Caffe kütüphanesi, tek bir NVIDIA K40 GPU kullanılarak bir günde 60 milyondan fazla görüntü işleyebilmektedir. Caffe mevcut en hızlı ESA uygulamaları arasındadır (BVLC Caffe 2020)

Keras: Keras, TensorFlow ve Theano derin öğrenme kütüphanelerine dayanan modüler bir Python kütüphanesidir (Keras 2020). TensorFlow ve Theano kütüphaneleri temelde GPU veya CPU üzerinde çalışacak şekilde oluşturulmuştur. Keras kütüphanesi, araştırmacılar tarafından görüntü işleme alanında oldukça yaygın bir kullanım oranına sahiptir.

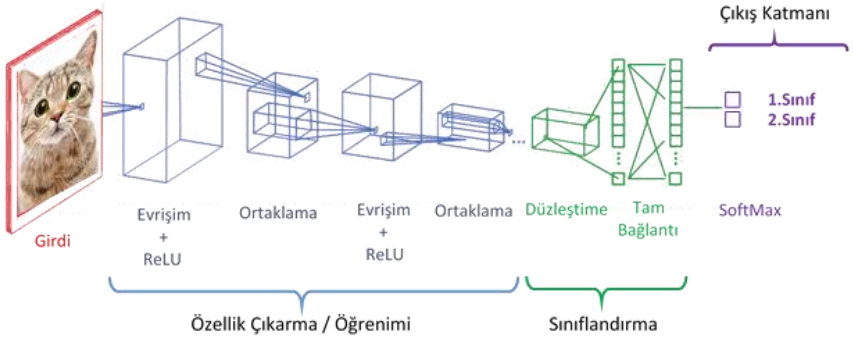
Torch: Makine öğrenmesi algoritmalarına oldukça fazla destek sağlayan Torch kütüphanesi LuaJIT programlama dili kullanılarak yazılan ve C/CUDA uygulama mimarisini kullanan, açık kaynaklı bir yapıya sahip olan ve GPU desteği sağlayan bilimsel bir bilgi işlem yapısı olarak ifade edilebilir. Torch kütüphanesi sayısal optimizasyon yöntemlerini kullanabilmesinden dolayı, çeşitli sinir ağları ve enerji tabanlı modelleri içinde barındırmaktadır. Torch sürekli geliştirilmekte ve Facebook, Google ve Twitter gibi çeşitli şirketler tarafından kullanılmaktadır (Torch.ch 2020).

İKİNCİ BÖLÜM

2. EVRİŞİMSEL SİNİR AĞLARI (CONVOLUTİONAL NEURAL NETWORK-CNN)

Derin öğrenme kullanılarak görüntü analiz, sınıflandırma, tespit ve takip konularında yapılacak çalışmalarda en çok kullanılan YSA sınıfı Evrişimsel Sinir Ağı (CNN)'dır. CNN'in çalışma şeklini anlamak için ilk başta konvolusyon işleminin anlaşılması gereklidir. Konvolusyon işlemi temelde iki fonksiyon arasında tanımlanan bir işlemdir ve tanımlanan bu iki fonksiyondan üçüncü bir fonksiyon türetilir. Türetilen üçüncü fonksiyon, ikinci fonksiyonun birinci fonksiyonu nasıl etkilediğini ifade eder. Evrişim işlemi asterix veya yıldız sembolüyle ifade edilen, olasılık, bilgisayarlı görü, istatistik, doğal dil işleme, görüntü ve sinyal işleme, diferansiyel denklemler ve mühendislik konuları gibi birçok problemin çözümünde kullanılan bir işlemdir. Bu işlem $(f * g)(t)$ şeklinde ifade edilir. Krizhevsky ve ark. 2012 yılında imagenet veri setini kullanarak yaptıkları bir çalışmada CNN'i nesne sınıflarını bulmada gözetimli öğrenme metodları ile birlikte en çok kullanılan yöntem olarak ifade etmişlerdir (Krizhevsky vd, 2012). Yapılan bu çalışmada da CNN mimarisi kullanılmıştır

CNN temel olarak evrişim (konvolüsyon), ortaklama (pooling) ve tam bağlantı katmanlarından oluşmaktadır. Şekil 13'te bu katmanlar gösterilmiştir.

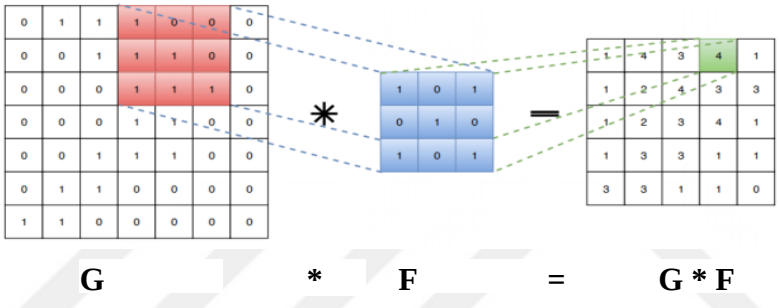


Şekil 12. CNN Katmanları (Mete 2019)

2.1. Evrişim (Konvolüsyon) Katmanı

Evrişim katmanı girdi olarak verilen görüntüdeki özellikleri belirlemek için kullanılan ve özellik çıkarıcı katman olarak ta isimlendirilen katmandır. Burada bahsedilen özellik görüntüdeki basit kenar, eğim ya da daha karmaşık ağız, burun veya gözün bulunduğu nokta olabilir.

Özellik belirleme işlemi basit olarak tasarımcı tarafından belirlenen bir filtreyle giriş matrisinin çarpımından ibarettir. Bu sayede ağ üzerinde verinin önemli bölgeleri tespit edilmiş olunur. Bu çarpma işlemi tüm girdi boyunca yapılarak konvolüsyon işlemi tamamlanmış ve eğitimin yapılacağı veri elde edilmiş olunur. Şekil 14'te evrişim işlemi basit olarak gösterilmiştir.



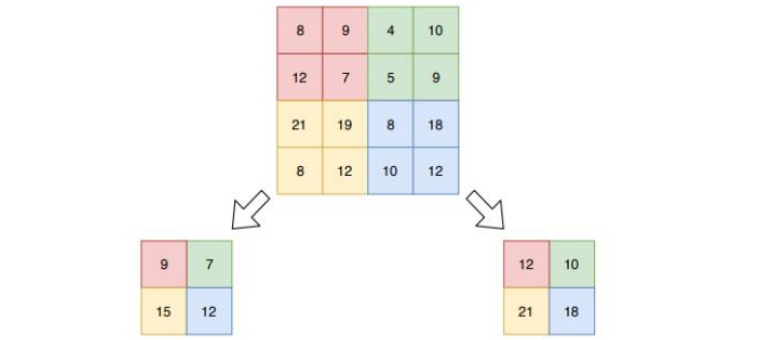
Şekil 13. Temel Evrişim İşlemi

Ayrıca konvolüsyon işlemi sonucunda elde edilen çıktıdaki negatif değerlerin sıfır yapılması için ReLU aktivasyon fonksiyonu kullanılır. Bu yüzden evrişim katmanı ReLU aktivasyon fonksiyonunu da içinde barındırmalıdır.

2.2. Ortaklama (Pooling) Katmanı

Ortaklama katmanı genellikle bir evrişim katmanından sonra gelir ve temel görevi kendisinden sonra gelen evrişim katmanı için girdi boyutunu küçültmektir. Ortaklama katmanından sonra veri boyutu küçüldüğünden bir sonraki evrişim katmanında daha fazla hesap yoğunluğu oluşur. Aynı zamanda veriye ait bazı değerler kaybolacağından sistemin ezberlemesinin de önüne geçilmiş olunur. Ortaklama katmanında belirlenen fitre görüntü üzerinde gezdirilir ve piksellere ait en büyük (maksimum) ya da ortalama (avarage) değerler alınarak evrişim katmanına iletilir. Uygulamalarda daha yüksek başarım değerleri verdiği için genelde maximum

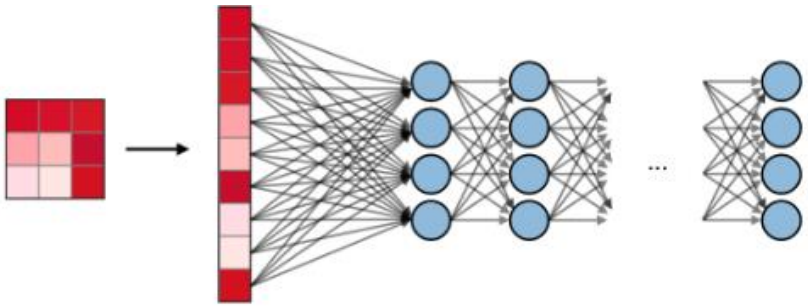
havuzlama tercih edilir. Şekil 15'te maksimum havuzlama işleminin uygulanması gösterilmiştir.



Şekil 14. 2x2 için Maksimum ve Ortalama Havuzlama (Güney 2019)

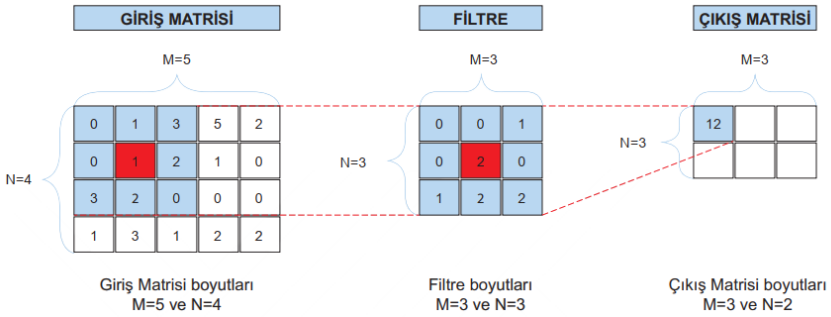
2.3. Tam Bağlantı (Fully Connected) Katmanı

Evrişimsel sinir ağlarında Evrişim+ ReLu ve ortaklama katmanlarında sonra eğer varsa tam bağlantılı (FC) katman bulunur ve sınıf skorları gibi hedefleri optimize etmek için kullanılır. Tam bağlı katman (FC), her girişin tüm nöronlara bağlı olduğu bir giriş üzerinde çalışır (A.Amidi vd,2020). Şekil 16'da tam bağlantı katmanının yapısı gösterilmiştir.



Şekil 15. Tam Bağlantı Katmanı (Amidi vd, 2020)

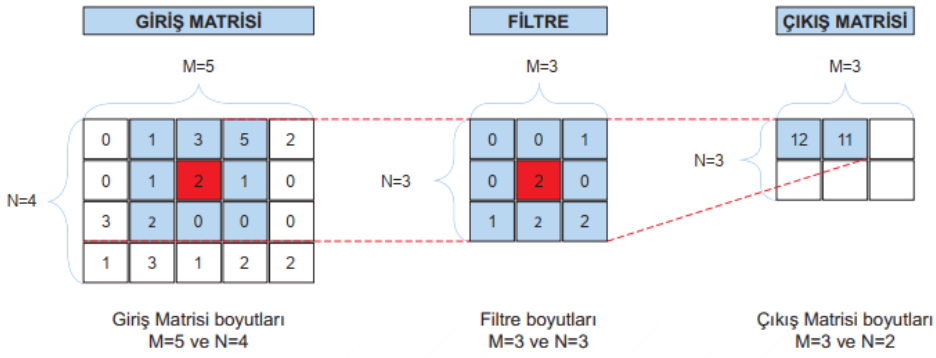
Evrişim işleminin daha iyi anlaşılabilmesi için 2 boyutlu evrişim işlemini adım adım inceleyecek olursak; Şekil 17 incelendiğinde örnek uygulamada eni ve boyu M × N olan üç adet matris bulunmaktadır. Bu matrislerden ilki giriş matrisi, ikinci matris, giriş matrisi üzerinde işlem yapmak için kullanacağımız filtre matrisi (3 × 3) ve üçüncü matris ise evrişim işlemi sonucunda elde edilen çıkış matrisidir.



Şekil 16. Evrişim İşlemi Adım-1 (Başarır 2019)

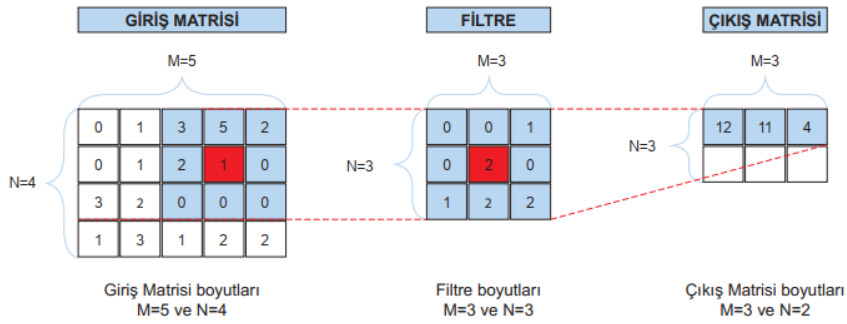
Tasarımcı tarafından belirlenen parametrelere göre (*giriş matrisi boyutu, filtre boyutu, kaydırma ve doldurma işlemlerinin miktarı*) çıkış matrisinin boyutu otomatik olarak hesaplanacaktır. Evriştirme işlemi sırasında filtre matrisinin merkezinde bulunan ve kırmızı ile işaretlenmiş olan kısım giriş matrisindeki aynı renkte işaretlenmiş olan alanının üstüne yerleştirilir. Bu aşamadan sonra matris çarpım işlemi gerçekleştirilerek $\boldsymbol{0 \times 0 + 1 \times 0 + 3 \times 1 + 0 \times 0 + 1 \times 2 + 2 \times 0 + 3 \times 1 + 2 \times 2 + 0 \times 2 = 12}$ sonucu elde edilir ve çıkış matrisinin ilk hücresine yerleştirilir.

Bu işlem filtre matrisi giriş matrisinin tüm değerleriyle evriştirilene kadar filtre bir birim sola kaydırılarak tekrar edilir. Bir sonraki adımda çıkış matrisinin sıradaki hücre değeri yine matris çarpımıyla hesaplanır ve Şekil 18'daki sonuç elde edilir.



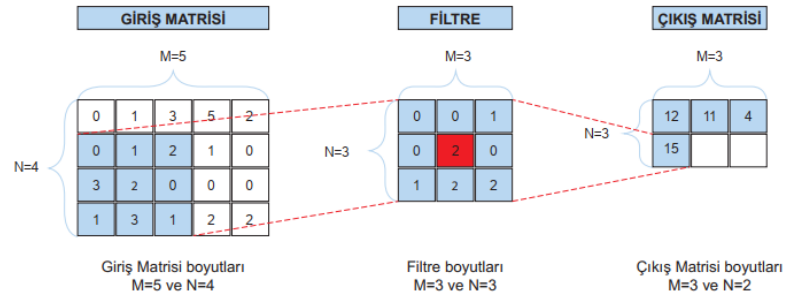
Şekil 17. Evrişim işlemi Adım-2 (Başarır 2019)

Filtre giriş matrisinin üzerinde bir birim sola kaydırılır ve çıkış matrisinin sıradaki hücre değeri şekil 19'deki gibi hesaplanır.



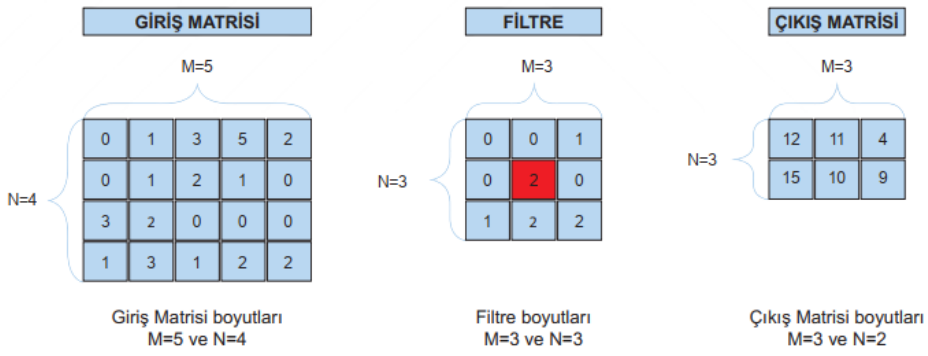
Şekil 18. Evrişim işlemi Adım-3 (Başarır 2019)

Şekil 19'da görüldüğü üzere filtre artık giriş matrisinin ilgili satırında en sola dayanmıştır. Artık sola kaydırma yapılamayacağı için filtre giriş matrisinde en sağa ve bir birim aşağıya kaydırılarak çıkış matrisinin sıradaki hücre değeri Şekil 20'deki gibi hesaplanır.



Şekil 19. Evrişim işlemi Adım-4 (Başarır 2019)

Sıradaki adımda Filtre giriş matrisinin üzerinde bir birim sola kaydırılır ve çıkış matrisinin sıradaki hücre değeri şekil 21'daki gibi hesaplanır.



Şekil 20. Evrişim işlemi Adım-5 (Başarır 2019)

Tasarımcının belirlediği filtre ile giriş matrisi Şekil 21'de görüldüğü gibi tamamen evrişim işlemine tabi tutulmuş ve çıkış matrisi elde edilmiştir. Evrişim işlemi sonucunda farklı iki fonksiyondan anlamlı bir birleşim fonksiyonu elde edilmiş olunur. Elde edilen çıkış matrisi gerçekte, filtrenin giriş matrisine uygulanması ile giriş matrisinin bazı özelliklerinin arttırılması, bazı özelliklerinin ise azaltılması anlamını taşır.

2.4. Evrişim İşleminde Parametre Kavramı

Evrişim işleminde çıkış matrisinin boyutu tasarımcının belirlediği bazı parametrelere bağlı olarak değiştiği ifade edilmişti. Belirlenen bu parametrelere kısaca değinelim.

2.4.1. Filtre Kaydırma (Stride)

İncelenen örnekteki evrişim işlemine bakacak olursak modelimiz için belirlediğimi filtremiz giriş matrisinde yalnız bir birimlik sağa ve aşağıya kaydırılarak çıkış matrisi oluşturulmuştur. Yapılan bu işleme kaydırma (stride) denir. Kaydırma ne kadar çok olursa çıkış matrisinin boyutu o kadar küçülecektir. Kaydırma işlemi günümüzde, maksimum örnekleme (maxpooling) denilen ve giriş matrisinde belirlenen alandaki değerlerin en yüksek olanlarının alınarak yeni bir matris oluşturulması şeklinde tarif edilebilecek işlemin yerine daha yaygın olarak kullanılan işlemdir. Kaydırma işleminin tercih edilmesindeki en önemli etken maxpooling'e göre hem daha hızlı çalışması hem de giriş matrisinin çıkışa etkisinin daha iyi belirgin olmasıdır.

2.4.2. Sıfır Ekleme (Zero-Padding)

Çıkış matrisinin boyutu ile giriş matrisinin boyutunun eşit olması gereken durumlarda giriş matrisinin çevresine yeni satır veya sütunlar ekleyerek giriş matrisinin boyutunun büyütülmesi ve burada oluşan yeni hücrelere sıfır verisi girilmesi işlemi olarak ifade edilebilir.

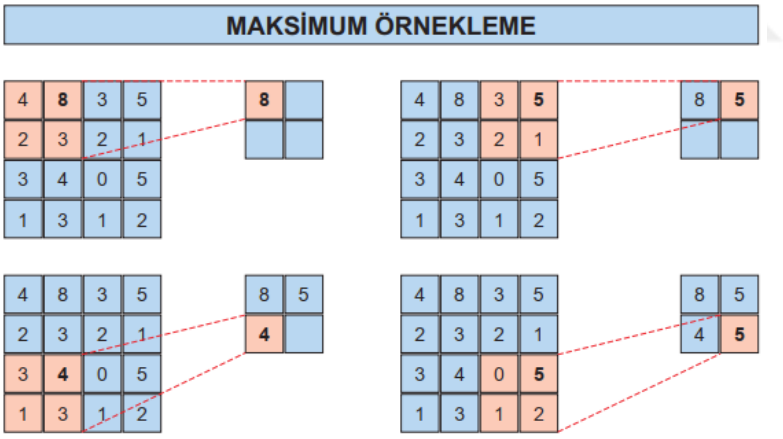
Yapılacak işlemi matematiksel olarak ifade edecek olursak çıkış matrisinin boyutu Denklem 7'deki gibi bulunur. *(Çıkış matrisinin boyutu= ÇB, Giriş matrisinin boyutu= GB, Filtre boyutu =FB, Kaydırma sayısı= K, Dolgulama sayısı= D)*

Denklem 7 Çıkış Matrisinin Boyutu (Başarır B,2019)

$$ÇB = \frac{GB - FB + 2 \times D}{K} + 1 \quad (7)$$

2.4.3. Maksimum Örnekleme (Maxpooling)

CNN'de evrişim uygulanan katmanı daha anlamlı olacak şekilde küçültmek ve işlem hızını arttırmak için maksimum örnekleme katmanı kullanılır. Evrişim işlemi sonucunda hesaplanan özellik haritaları yeniden evrişim işlemine tabi tutulur bu sayede yeni filtreler ve yeni özellik haritaları elde edilmiş olunur. Yeniden evrişim işlemi sonucunda elde edilen özellik haritalarından belirli bir bölge seçilir ve bu bölgedeki belirlenen en büyük değer kullanılmak üzere alınır. Maksimum örnekleme işlemi ile özellik haritasından özet bir yapı çıkarılmış olunur. Bununla birlikte alternatif olarak geliştirilen filtre kaydırma miktarını 2 veya daha fazla olacak şekilde ayarlamak maksimum örnekleme katmanı maksimum örnekleme katmanı kullanmaya göre daha başarılı sonuçlar verdiği kabul edilmektedir. Kısaca maksimum örnekleme şekil 22'teki gibi yapılır;



Şekil 21. Maksimum Örnekleme Adımları (Başarır 2019)

Şekil 22 incelendiğinde 4x4 lük giriş matrisi maksimum örnekleme işlemi sonucunda 2x2'lik bir yapıya dönüştürülmüş olunur. Bu dönüşüm işlemi sonucunda işlem hızı etkin bir oranda artmış olacaktır.

2.4.4. İyileştirici (Optimizer)

CNN'de iyileştirme, yapılan işlem sonucunda hesaplanan kayıp değerinin hangi metot kullanılarak en düşük seviyeye çekileceğinin belirlenmesidir. 2012 yılında

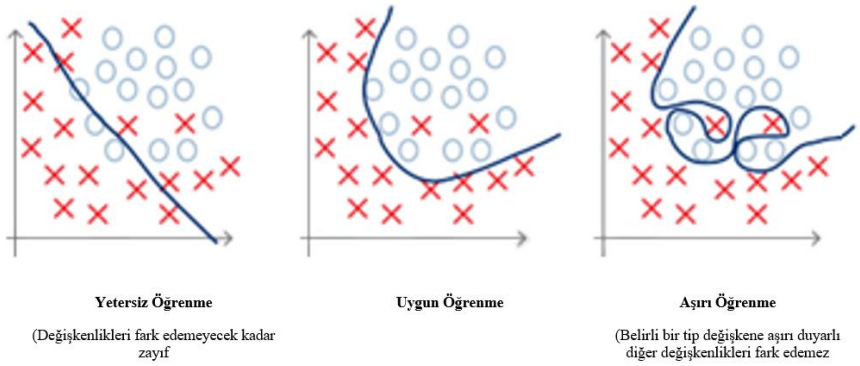
Byrd ve ark. yaptıkları bir çalışmada seçilen filtrelerin istenilen nesneleri tespit edip, gerekli özellikleri çıkarıp ve doğru bir tahmin sonucuna ulaşmak için ağda nasıl bir yol izlenmelidir sorusuna cevap aramışlardır (Byrd vd, 2012). CNN konusunda araştırmacılar tarafından en çok kullanılan optimizasyon metotlarının, Kademeli İniş Metodu (Gradient Descent), Ağırlık Düşürme Katsayısı (Momentum), RMSprop ve Adam olduğu söylenebilir.

2.5. CNN Eğitim Aşaması

YSA eğitirken tasarımcının dikkat etmesi gereken en önemli faktörlün eğitim kümesi olduğu söylenebilir. Tasarımcı hangi modeli belirlemiş olursa olsun seçtiği eğitim kümesinin boyutu ve çeşitliliği yeterli değilse eğitim sonucunun istenilen oranda bir başarıya ulaşması mümkün değildir. Çünkü yetersiz eğitim kümesinden dolayı model yeterince eğitilemediğinden ya kısıtlı bir veri kümesine duyarlı olur ya da tüm veri kümesine duyarsız olur.

Bir veri seti oluşturulurken verinin bir kısmı eğitim için, bir kısmı test için ve bir kısmı da seçilen mimarinin doğru çalışıp çalışmadığını gözlemleyecek doğrulama veri seti olarak ayrılmalıdır. Bu ayrımlar yapılırken eğitim ve doğrulama veri kümelerinin etiketlenmiş kümeler olduğu unutulmamalıdır.

Tasarımcının belirlediği modele ve eğitim kümesine göre ağ eğitildikten sonra eğitim sonucunun test edilebilmesi için etiketlenmemiş verilerden oluşan test veri kümesi kullanılır. Eğitilmiş modelin başarısı hiç görmediği bir veriyi hangi oranda doğru bir şekilde etiketlemesiyle belirlenir. Şekil 23'te eğitilmiş bir modelin farklı öğrenme durumları gösterilmişti.



Şekil 22. Aşırı öğrenme, Normal öğrenme ve Yetersiz öğrenme (tinyurl.com/y2ljr34t 2020)

2004 yılında Hawkins yaptığı çalışmada eğitim setinde hesaplanan hatanın düşük, doğrulama setinde ise hesaplanın hatanın yüksek olduğu bir durumu ifade etmiştir. Bu duruma aşırı öğrenme (overfitting) adı verilir. Şekil 23'te görüldüğü gibi aşırı öğrenme aslında overfitting, modelin verilen veri setini öğrenmek yerine ezberlemesi olarak ifade edilebilir. Eğer modelimiz öğrendiği veri seti üzerinde çalıştırıldığında verdiği sonuçlarla çok iyi, görmediği bir veri setinde sonuçlar yetersiz ise bu durum overfiting durumudur. Yine Şekil 23'e bakacak olursak az öğrenme (underfitting) denilen bir kavramdan da bahsetmemiz gereklidir. Underfittingde ise modelimiz veri setini ezberlemek yerine anlamaya çalışmakta, fakat yeterince anlayamadan eğitim bitmektedir. Bu durumda model ile underfitting yapmışsak modelimiz hem öğrendiği veri setinde hem de görmediği test veri setinde kötü sonuçlar verecektir. Overfiting ve underfiting kavramları modelin esnekliğiyle doğrudan ilgilidir. Sonuç olarak, modelimiz ne kadar esnek ise overfitting olma ihtimali o kadar yüksek ve modelimiz ne kadar "esnek değil" ise underfitting olma ihtimali o kadar fazla olacaktır. Eğitim sonucunda istenilen öğrenme Şekil 23'teki uygun öğrenme ile belirtilen grafikteki gibi olmalıdır. Tasarımcının belirlediği mimaride değişiklik yaparak ve düzenlileştirme, seyreltme, yığın normalleştirme gibi yöntemleri kullanarak, ya da eğitim daha erken kesilerek overfiting veya underfting problemlerini çözmeye çalışabilir. Bu yöntemlere kısaca bakacak olursak;

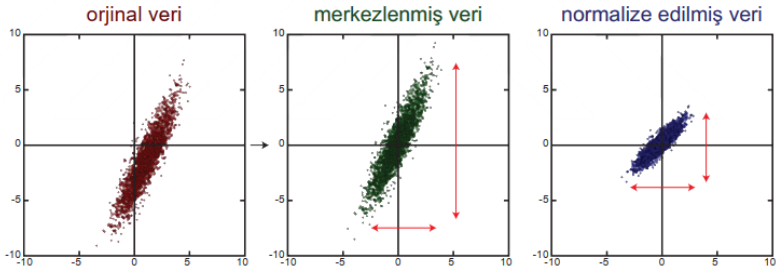
2.5.1. Seyreltme (Dropout)

2016 yılında Hinton ve ark. yaptıkları çalışmada tam bağlantılı YSA'da overfiting probleminin çözümü için YSA'daki bazı sinir hücrelerinin belirlenen oranda

kapatılması olarak ifade etmişlerdir. Bu reguralizasyon yöntemi yalnız aşırı öğrenme riski olan yerlerde ve yalnız CNN'nin genellikle sonunda yer alan tam bağlantı katmanlarına uygulanabilir.

2.5.2. Yığın Normalleştirme (Batch Normalization)

Ioffe ve Szegedy 2015 yılında yaptıkları bir çalışmada bu regularizasyon yönteminin CNN'i kaybolan eğime (vanishing gradiente) dayanıklı hale getirerek modele daha iyi performans ve daha kısa eğitim zamanına sahip olmasını sağladığını ifade etmişlerdir. Yığın normalleştirmenin seyreltmeye göre regularizasyon işlemini daha iyi başardığı için dolayı modern mimarilerde sıklıkla tercih edilmektedir.



Şekil 23. Normalizasyon İşlemi (tinyurl.com/y4gmmb4m 2020)

Şekil 24'te verinin orijinal hali, merkezlenmiş hali ve normalizasyon yapılmış hali gösterilmiştir. Bu işlemler sonucundan seçilen model daha az iş yüküyle boğuşmak zorunda kalacak ve bu sayede daha hızlı çalışacaktır.

2.6. Evrişimsel Sinir Ağı Modelleri

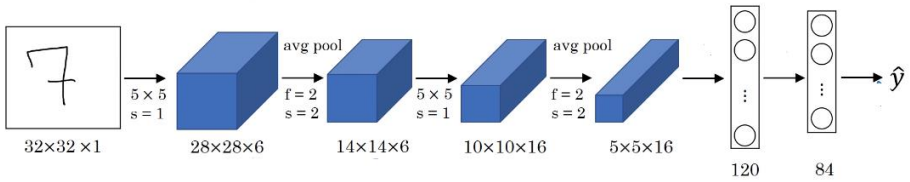
Tipik bir CNN mimarisi, birbiri ardına gelen ReLU katmanı ve havuzlama katmanı ile birlikte bir araya getirilmiş evrişimsel katmandan oluşur. Girdi olarak verilen görüntü ağ boyunca ilerledikçe boyut olarak küçülür, ancak özellik haritalarındaki artış nedeniyle tipik olarak daha da derinleşecektir. İleri beslemeli sinir ağı yığının üstüne eklenir ve son katmanda sınıf tahminlerini (softmax katmanı) verir.

Yıllar boyunca araştırmacılar bu temel mimarinin türevlerini geliştirerek çalışmalarında kullanmışlardır. Bu mimariler araştırmacılar tarafından özellikle nesne tanıma, nesne tespit, görüntü sınıflandırma gibi konularda sıklıkla tercih edilmektedir.

Bu bölümde, en yaygın kullanılan CNN mimarileri yeni yaklaşımlarıyla tanıtılmaktadır.

2.6.1. LeNet-5

1998 yılında LeCun ve ekibi tarafından yayınlanmış ve ilk olarak başarılı sonuç elde edilen ESA modeli olarak kabul edilmiştir. Yann LeCun ve ekibi tarafından MNIST (Modified National Institute of Standards and Technology) veri seti kullanılarak posta numaraları, banka çekleri üzerindeki sayıların okunması amacıyla deneyler yapılmıştır. LeNet modeli kendisinden sonra geliştirilen diğer modellerden farklı olarak en büyük ortaklama yerine ortalama ortaklama yöntemiyle boyut azaltma adımlarını gerçekleştirmektedir. Ayrıca LeNet mimarisi sigmoid ve hiperbolik tanjant aktivasyon fonksiyonlarını kullanmaktadır. Lecun ve ekibi yaptıkları çalışmada ESA'nın tam bağlantı katmanına giren parametre sayısını 5x5x16=400 ve çıkışında 10 sınıflı eşikleyici bulunduğunu ifade etmektedirler. Ayrıca LeNet modeli 60 bin parametre hesaplamaktadır (Lecun vd, 1998). LeNet mimarisinin yapısı Şekil 25'te görülmektedir.

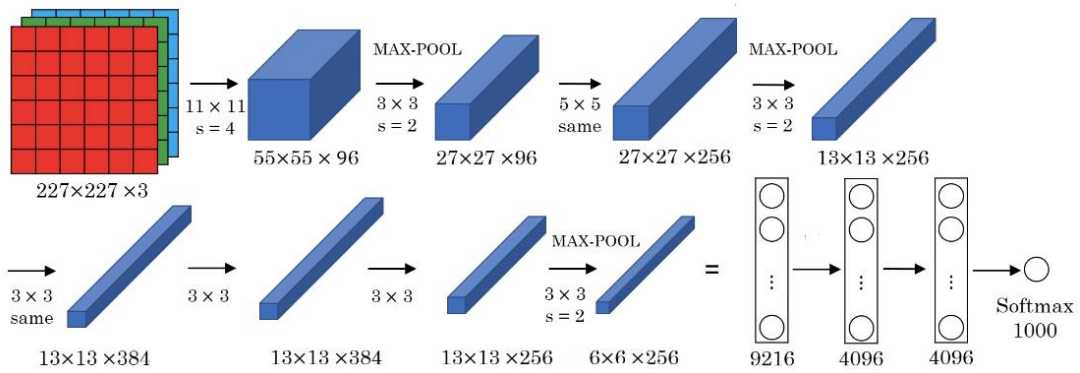


Şekil 24. LeNet Mimarisi (Lecun vd, 1989)

2.6.2. AlexNet

Araştırmacılar için 2012 yılında yapılan ImageNet yarışmasıyla ESA ve derin öğrenme konuları yeniden popüler hale gelmiştir. Alex Krizhevsky, Ilya Sutskever ve Geoffrey Hinton tarafından geliştirilen AlexNet mimarisi 2012'de bu yarışmayı kazanmıştır. AlexNet mimarisi, artarda gelen evrişim ve pooling katmanlarından dolayı LeNet modeline oldukça benzemektedir. LeNet modelinden farklı olarak ReLU (Rectified Linear Unit) aktivasyon fonksiyonunu ve ortaklama katmanlarında ise en büyük ortaklama yöntemini kullanılmaktadır. AlexNet 60 milyona yakın parametrenin kullanıldığı paralel çift GPU üzerinde çalışan ilk model olarak kabul edilmektedir. AlexNet modelinin 2012 yılında yapılan ImageNet ILSVRC yarışmasında sınıflandırma doğruluk oranını %74,3'ten %83,6'ya yükseltmesi görüntü

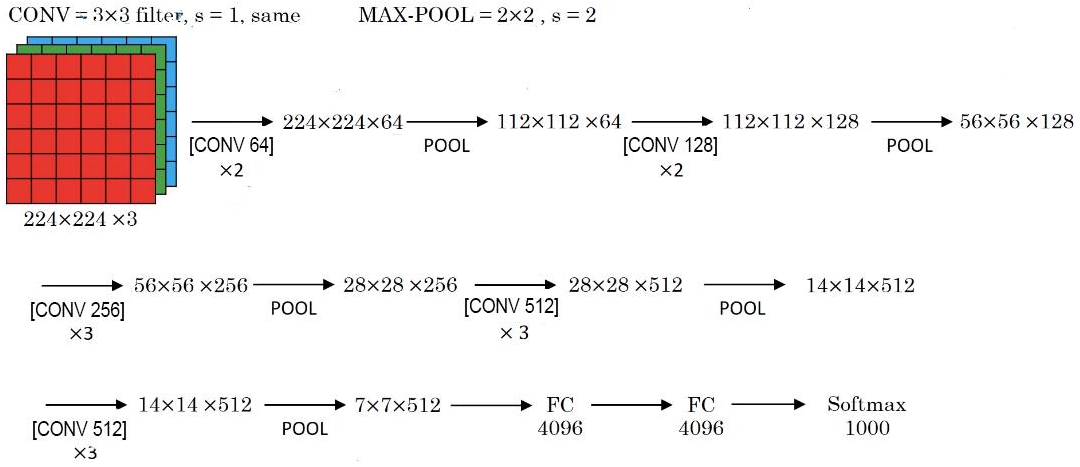
sınıflandırma probleminde bir kırılma noktası olarak kabul edilmektedir (Krizhevsky vd, 2012) . Şekil 26'da AlexNet mimari yapısı gösterilmiştir.



Şekil 25. AlexNet Mimarisi (Kızrak 2018)

2.6.3. VGG-16

VGG-16 yapısal anlamda basit bir ağ modeli olarak kabul edilir ve 2'li veya 3'lü evrişim katmanı kullanması diğer modellerden VGG-16 modelini ayıran en temel özelliğidir. Bu modelin tam bağlantı katmanında 7x7x512=4096 nöronlu bir öznitelik vektörü oluşturulur. Kullanılan iki tam bağlantı katmanı çıkışında 1000 sınıf için sofmax başarım değeri hesaplanabilir.Modelde yaklaşık 138 milyon parametre hesabı yapılmaktadır (Zeiler ve Fergus 2013). Farklı modellerde de olduğu gibi matrislerin boyutları (yükseklik,genşlik) girişten çıkışa doğru azalırken, kanal sayısı (derinlik) girişten çıkışa doğru artmaktadır. Şekil 27'de VGG-16 mimari yapısı gösterilmiştir.



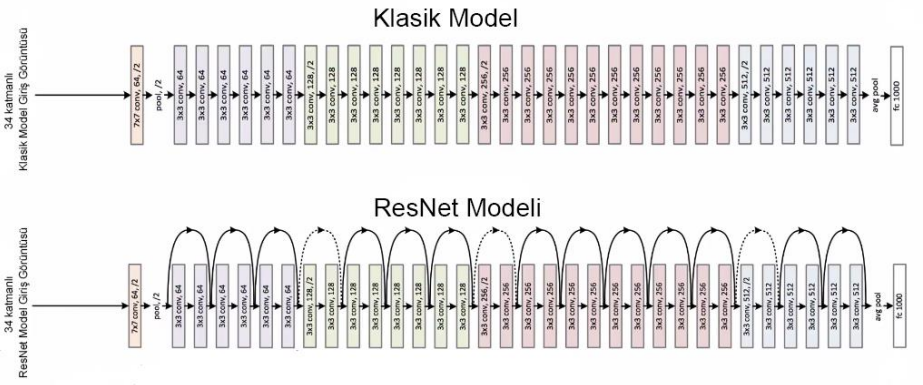
Şekil 26. VGG-16 Mimarisi (Kızrak 2018)

2.6.4. ResNet

Modellerin gerçek anlamda derinleşmeye başlamasıyla birlikte ResNet daha önceki modellere göre daha derin bir yapıyla geliştirilmiştir. ResNet bilinen klasik modellerden ayıran en önemli özelliği ise modele artık değerlerin (residual value) kendisinden sonraki katmanlara besleyen blokların (residual block) modele eklenmesiyle oluşturulmasıdır.

Doğrusal ve ReLU katmanları arasında her iki katmanda bir eklenen bu değer önceden gelen a[l] değeri, a[l+2] hesabına eklenmiş olacağından sistemdeki hesabı da değiştirmektedir. Teorik olarak modelin katman sayısının arttırılması model başarımının da artacağı anlamına gelir, ancak pratikte başarım sonucuna bakıldığında durumun böyle olmadığı görülmüştür. Böylece $w[l+2]=0$ olduğu durumda bu değerin türevinin de sıfır değeri üretmesi sorunu (vanishing gradient) ortaya çıkaracağından istenmeyen bir durum olarak ifade edilir (Zhang vd, 2015).

Yeni teoride bu durum artık değer beslemesinin yeni çıkışı iki önceki katmandan gelen $a[l]$ değeri için o an ki ağırlık sıfır bile olsa öğrenme hatasını optimize eder ve modeldeki ağın daha hızlı eğitilmesini sağlar. Şekil 28'de Klasik ve ResNet model mimarileri gösterilmektedir.



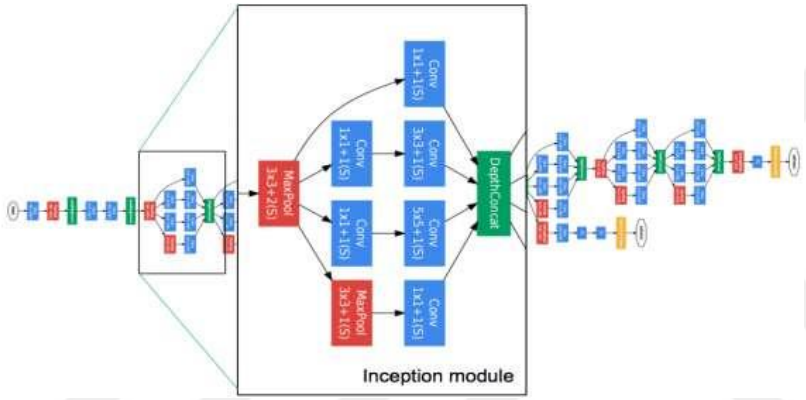
Şekil 27. Klasik ve ResNet Model Mimarileri (Kızrak 2018)

2.6.5. GoogLeNet

GoogLeNet Inception modeliyle yalnız katman sayısını artırmanın başarımı artırmak için yeterli olamayacağı, ağın katmansal anlamda derinleşirken genişlemesinin de gerekliliği ortaya çıkmıştır. Çalışmada şimdiye kadar anlatılan ağ modellerine göre daha farklı ve anlaşılması daha zor olan model GoogLeNet

modelidir. Bununla beraber modelin hızını ve başarımını hesaplama karmaşıklığına ve büyüklüğüne bulunan çözüm yöntemi sağlamaktadır. Inception ağ modeli ezberleme ihtimalini azaltmak için farklı boyutlardaki evrişim ve en büyük ortaklama işlemlerinden meydana gelen modüllerden oluşmaktadır. Her bir modüle 'inception' adı verilmektedir.

Toplam 9 inception bloğundan oluşan model, derin öğrenme konusunda başlangıç kabul edilen LeNet modeline atıf yaparak GoogLeNet ismi verilmiştir. GoogLeNet 5 milyon parametre ile AlexNet'e göre 12 kat daha az işlem yüküne sahiptir (Szegedy vd, 2016). Şekil 29'da GoogleNet mimarisi gösterilmektedir.



Şekil 28. GoogLeNet Mimarisi (Szegedy 2015)

2.7. Transfer Öğrenme (Transfer Learning)

Belirli bir problemin çözümü için kullanılacak bir modelin eğitimi, modelin karmaşık bir yapıya sahip olması veya modelin eğitimi için çok büyük veri kümelerine ihtiyaç duyulması gibi sebeplerden dolayı normal bilgisayarlarla çok uzun zaman alabilmektedir. Yani yüksek başarım oranına sahip bir modelin eğitimi için değişken özellikleri barındıran büyük bir veri kümesi, yeterli sayıda test yapabilmek için zaman ve güçlü donanım özelliklerine ihtiyacımız vardır.

Araştırmacılar için problemin çözümünde gerekli miktarda veriyi toplamak, toplanan bu verileri işleyecek donanım ve zaman sorununu çözmek oldukça ciddi bir zorluk olarak kabul edilmektedir. Araştırmacılar bu sorunların üstesinden gelebilmek için literatürde "Transfer Learning- Transfer Öğrenme" diye adlandırılan ve daha önceden eğitilmiş CNN ağlarını kullanmayı sağlayan yöntemi tercih etmişlerdir. Transfer Öğrenme ile araştırmacılar daha önceden ImageNet ve Ms-Coco gibi büyük

veri kümeleri kullanılarak eğitilmiş olan Alexnet, Resnet, VGG gibi CNN modellerinin sadece çıkış katmanlarında değişiklik yaparak özel ağlar oluşturulabilir bu sayede zamandan ve yüksek donanım ihtiyacından tasarruf etmiş olurlar.

Bir diğer sorun ise araştırmacıların belirli bir probleme çözüm üretmek için eğittikleri CNN ağında milyonlarca "weight" hesaplanması ve "bias" katsayılarının belirlenmesi sorunudur. Transfer öğrenme sayesinde çok fazla zaman olacak bu hesaplamaların sadece gerekli olanlarının yapılarak büyük ölçüde zaman ve donanımdan tasarruf etmeleri sağlanmıştır

Transfer öğrenme, daha önceden belirli bir görevi gerçekleştirmek için eğitilmiş bir modelin başka bir ilgili görev için yeniden tasarlandığı bir makine öğrenimi tekniği olarak ifade edilebilir.

Araştırmacı transfer öğrenme kullanarak yeni bir CNN ağı geliştirmek istiyorsa;

1. Kullanacağı ağın bütün konvolüsyon katmanındaki sabitleri dondurmalı,

2. Problemin çözümü için kullanılacak sınıflandırıcının sınıf sayısı ile kullanacağı ağın çıkış katmanındaki aktivasyon fonksiyonu ve aktive olan çıkış nöronlarının sayısını eşit olacak şekilde değiştirmeli,

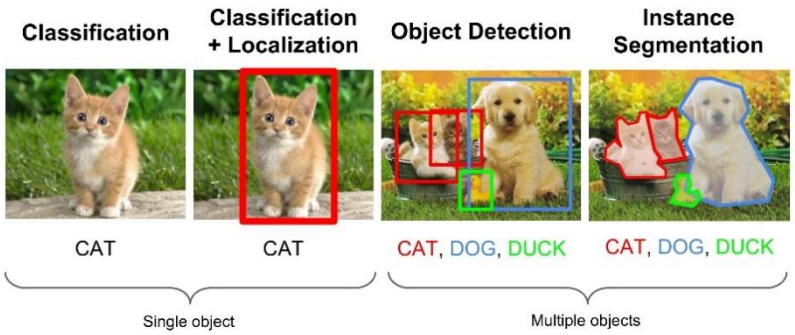
3. Tamamen bağlı katmanlara (Fully Connected Layer) ait ağ katsayıları başlangıç değerleri rastgele olacak şekilde mevcut veriler kullanılarak tekrar eğitilmelidir.

ÜÇÜNCÜ BÖLÜM

3. NESNE TANIMA

Nesne tespit işlemi filtre olarak isimlendirilen ve boyutu tasarımcı tarafından belirlenmiş pencere adı verilen matrisin giriş görüntüsü yüzeyinde kaydırılmasıyla, pencerenin üzerinde bulunduğu alanda ilişkili sınıfa ait veri olup olmadığının araştırılması işidir. Bu işlemi görüntüden pencerenin ebatları değiştirilerek nesneye ait sınıfların bulunması şeklinde de yapılabilir.

Ancak bu yaklaşım çok fazla zaman ihtiyacı doğurabilir veya güvenilir olmayabilir. Çünkü işlenen görüntü üzerindeki nesneler farklı boyutta, farklı konumda hatta üst üste binmiş olma ihtimalinden dolayı nesne tespiti için fazla sayıda ve boyutta pencere oluşturulması gerekebilir. Bu yaklaşım yerine etkin bir tespit işlemi, giriş görüntüsünün ilgili alanlarından nesnelere ait sınıf özetlerinin tasarımcı tarafın boyut ve özellikleri iyi belirlenmiş filtre ile çıkarılması şeklinde ifade edilebilir.



Şekil 29. Nesne Tanıma ve Tespit (Ouaknine 2018)

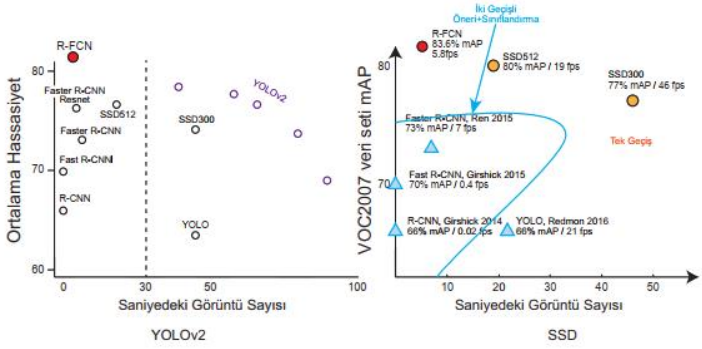
Şekil 30'da görülen nesne tespiti dört faklı biçimde yapılabilir. İlk resimde nesnenin yalnız sınıflandırma işlemi yapılarak nenenin ne olduğu ifade edilmiştir. Bu işlemde nesnenin görüntü üzerindeki konumu önemsizdir. İkinci resimde nesnenin görüntü üzerinde konumu ve sınıfı tespit edilmeye çalışılmıştır. Bu işlemde nesne bir sınırlayıcı kutu yardımıyla konumlandırılmış ve sınıf etiketi belirlenmiştir. Üçüncü fotoğraf da birden fazla nesnenin görüntü üzerindeki konumları farklı sınırlayıcı kutular yardımıyla belirtilmiş ve sınıf etiketleri belirlenmiştir. Dördüncü ve son fotoğrafta ise görüntü üzerindeki nesnelerin konumları sınırlayıcı kutular yerine

çerçeve konturları yardımıyla belirlenmeye çalışılmıştır ki bu işlem nesne bölütleme (segmantasyon) işlemi olarak ifade edilir. Bu tez çalışmasında üçüncü resim ile ifade edilen işlem gerçekleştirilmeye çalışılacaktır. Çalışmada derin öğrenme kullanılarak nesne tespit yöntemlerinden olan Faster R-CNN ve YOLOv5 yöntemi kullanılarak giriş far görüntülerindeki nesnelerin konumlarının belirlenmeye çalışılması amacıyla gerekli yazılım geliştirilmeye çalışılacak ve kullanılacak modellerin avantaj ve dezavantajları ifade edilmeye çalışılacaktır.

3.1. Güncel Nesne Tespit Yöntemleri

Bu tez çalışmasında kullanılacak nesne tespit yöntemlerinin temel amacı Şekil 31'de gösterilen üçüncü resmindeki gibi giriş görüntüsü üzerindeki nesnelere farklı sınırlandırma kutuları çizerek, nesnenin sınıfının ve görüntüdeki konumunun, nesne için çizilmiş olan sınırlayıcı kutunun belirlenen noktasına yazarak ifade etmektir.

Literatürde nesne tespit algoritmaları temelde, tüm işlemleri tek geçişte gerçekleştiren ve iki geçişte (Bölge bazlı) gerçekleştiren olmak üzere iki bölümde incelenir. Görüntü üzerindeki küçük nesnelerin tespitinde iki geçişli detektörlerin başarımlarının tek geçişli detektörlere göre daha iyi olduğu söylenebilir. İki geçişli detektörlerin en çok kullanılanı R-CNN mimari ailesidir. Bu aile R-CNN (Dai vd, 2016), Fast-RCNN (Girshick 2015) ve Faster R-CNN (Ren vd, 2015) olarak ifade edilir. Tek geçişli dedektörler olarak ifade edilen diğer nesne tespit yaklaşımında ise tespitin ilk adımında gerçekleştirilen bölgesel alan önerme aşaması elemine edilerek doğrudan nesne tespit işlemini özellik haritasındaki nesnenin olabileceği yoğun bölgelerde gerçekleştirmeye çalışmaktır. Tek geçişli detektörler iki geçişli detektörlere göre daha hızlı ve basit olmasına rağmen, nesne sınıflandırmadaki güvenilirlikleri ve görüntüdeki küçük nesnelerin tespitinde bazı dezavantajları vardır. Tek geçişli algoritmalara örnek olarak da YOLO (Redmon vd, 2016)) ve SSD (Liu vd, 2016) mimarileri verilebilir.



Şekil 30. R-CNN, YOLO ve SSD karşılaştırılması (Chen 2017)

Şekil 31'de de ifade edildiği gibi nesne tanıma konusunda tek geçişli detektörler iki geçişli detektörlere göre daha iyi başarıma sahiptirler. Daha öncede ifade edildiği gibi R-CNN mimari ailesi nesneleri daha iyi tespit ediyor olsa da yavaş olması bir dezavantajdır. Bu yüzden YOLO ve SSD algoritmalı özellikle gerçek zamanlı nesne tespitinde daha fazla kullanılmaktadır.

3.2. Bölge Bazlı Dedektörler

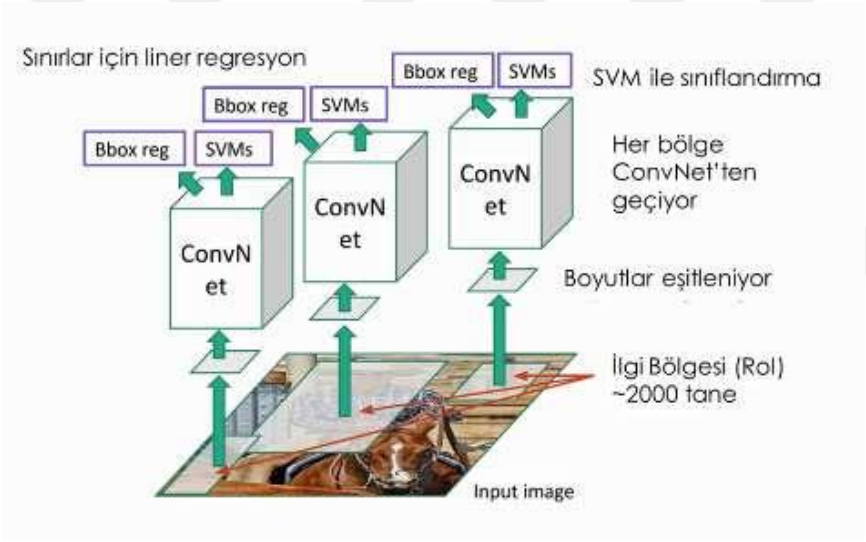
Bölge bazlı detektörler, dijital resimler veya videolardaki farklı boyutlara sahip nesnelerin belirlenmesinde oldukça önemli bir role sahiptir. Bölge önerisi yaklaşımında algoritmaya girdi olarak verilen görüntü üzerinden yaklaşık 2000 (iki bin) farklı bölge önerisi belirlenir. Önerilen her bir bölge ESA'ya girdi olarak verilir ve ağın özellik çıkarma yeteneği kullanılarak her bir bölge için otomatik özellik çıkarımı yapılır. Elde edilen bu özellikler doğrusal destek vektör makineleri yöntemi ile sınıflandırma işlemine sokularak görüntü üzerinde hangi nesnelerin bulunduğu tespit edilmeye çalışılır. Araştırmacıların çalışmalarında sıklıkla kullandığı bölge bazlı detektörler aşağıda verilmiştir.

3.2.1. R-CNN, Fast R-CNN, Faster R-CNN, Mask R-CNN

Nesne tespit işlemi sırasında bütün giriş görüntüsünü pencerelerle gezmektense bölge önerisi metoduyla belirlenen yaklaşık 2000 pencere içerisinde nesne aramak çok daha hızlı ve kolay olacaktır. Tamda yapılmak istenen bu işlem R-CNN'de (Region-based Convolutional Neural Network) (Girshick vd, 2015) gerçekleştirilmektedir. R-CNN' de değişik boyutlarda bölge öneri pencereleri oluşturulur. Önerilen pencereler tek tek evrişimli sinin ağından geçirileceğinden ve nesne sınıflandırma yapan evrişimli

sinir ağı belirli boyuttaki pencereleri kabul ettiğinden ancak önerilen pencerelerin boyutları eşitlenince evrişimli sinir ağından geçirmek mümkün olacaktır.

Evrişimli sinir ağı sonucunda belirlenen bölgede sınıflandırma işlemini yapabilmek için DVM (Destek Vektör Makineleri-Support Vector Machines) kullanılmaktadır (Vapnik 1999). DVM, makine öğrenmesinde lineer regresyon yöntemi ile de nesnenin etrafına dörtgen çizerek nesnenin sınırları belirleyen ve gözetimli öğrenmede kullanılan bir yöntemdir. Özetle R-CNN önerilen bölgede nesneyi arayacak ve nesneyi bulduğunda nesneye ait sınıfı, sınıf bilgisiyle birlikte nesnenin görüntüdeki konumunu ifade eden 4 değer döndürecektir. Döndürülen değerler nesnenin etrafına çizilecek olan dörtgenin görüntü üzerindeki koordinatlarını ifade etmektedir (Girshick vd, 2014). R-CNN mimari yapısı şekil 32'de görülmektedir.

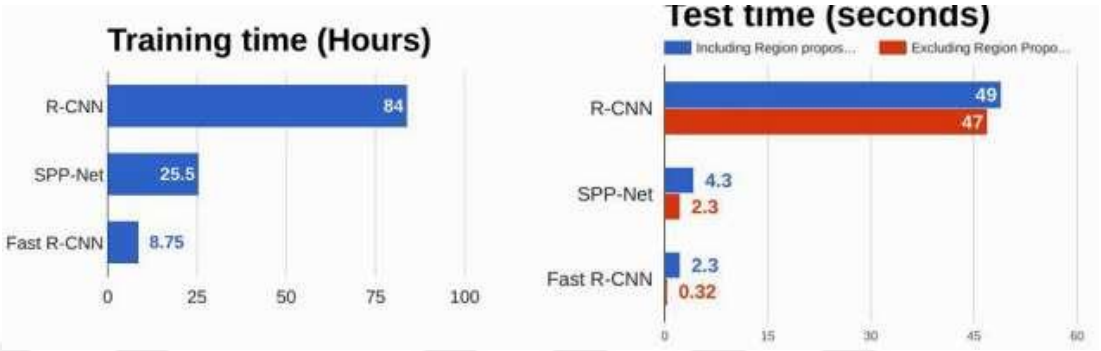


Şekil 31. R-CNN Mimari Yapısı (Girshick vd, 2014)

R-CNN mimarisinin uygulanması sırasında karşılaşılan sorunlara çözüm bulmak amacıyla Fast R-CNN (Girshick 2015) ve Faster R-CNN (Ren vd, 2017) geliştirilmiştir. R-CNN'de önerilen bölgelerin her biri evrişimli sinir ağından geçmek zorunda olduğundan bu durum R-CNN'in çalışmasını oldukça yavaşlatmaktadır. Ağdaki bu yavaşlamayı ortadan kaldırmak için Fast R-CNN geliştirilmiştir (Girshick 2015).

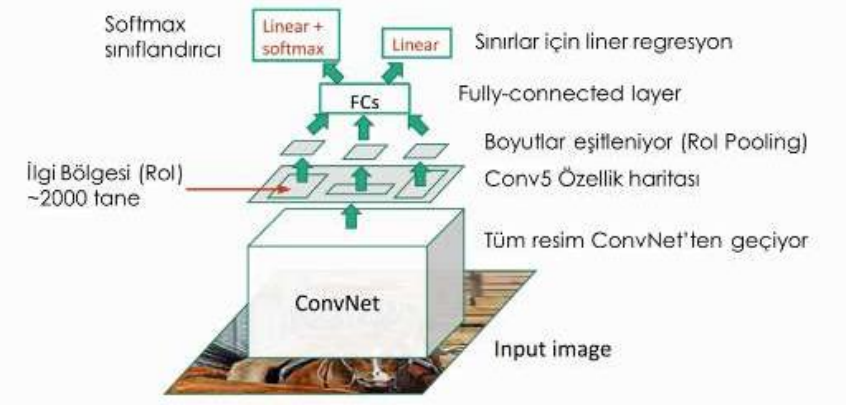
Fast R-CNN mimari olarak R-CNN ailesinin bir üyesi olduğu unutulmamalıdır. R-CNN'de her bir bölge önerisi ayrı ayrı ESA'dan geçirilirken Fast R-CNN'de giriş görüntüsü doğrudan ESA'dan geçirilerek, yüksek çözünürlüklü özellik haritaları

çıkarılmış ve bölge önerileri bu özellik haritalarından oluşturulmuştur. Bu sayede orijinal görüntüden bölge haritası çıkarıp ESA'da geçen süre kaybı ortadan kaldırılıp bölge önerileri özellik haritalarından çıkarılarak ortadan kaldırılmış olunur. Şekil 33'te Fast R-CNN ve R-CNN arasındaki hız farkı gösterilmiştir.



Şekil 32. Fast R-CNN ve R-CNN hız karşılaştırması (Vapnik 1999)

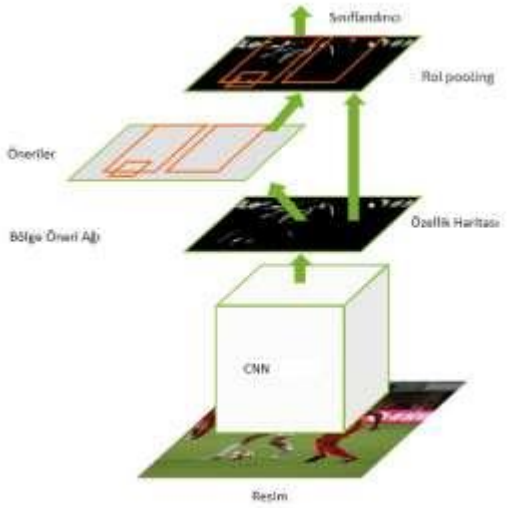
Bu adımdan sonra özellik haritasından oluşturulan bölge önerilerinin boyutları eşitlenerek tam bağlantılı katmana (FC) aktarılır. Son adımda ise nesne sınıflandırma yapılır ve tespit edilen nesnelerin görüntü üzerindeki sınırları belirlenmiş olunur. Diğer bir farklılık ise R-CNN nesne sınıflandırma için DVM kullanırken Fast R-CNN Softmax katmanını kullanılmaktadır (Girshick R ,2015). Şekil 34'te Fast R-CNN mimari yapısı görülmektedir.



Şekil 33. Fast R-CNN Mimari yapısı (Vapnik 1999)

Geliştirilen Fast-RCNN mimarisi modelin eğitim aşamasında yeterince hızlı çalışmaktadır. Buna rağmen Fast R-CNN oluşturulan modelin test aşamasında çoğunu bölge önerisi yapmak için çok fazla zamana ihtiyacı vardır. Bu sorunun ortadan kaldırılması için Faster R-CNN mimarisi geliştirilmiştir. Faster R-CNN hız kazanmak

için, R-CNN den farklı olarak bölge önerisi işlemini ağ içerisinde yapmaktadır. Şekil 35'te Faster R-CNN in Mimari yapısı görülmektedir.



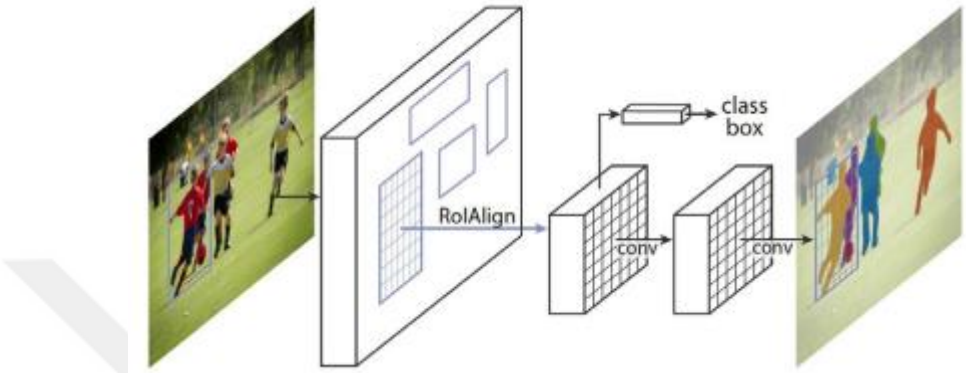
Şekil 34. Faster R-CNN in Mimari yapısı (Ren vd, 2017)

Faster R-CNN de giriş görüntüsüne ESA uygulanır ve özellik haritası çıkarılır. Bu adımdan sonra bölge önerileri R-CNN'de kullanılan DWM yerine RPN (Bölge Öneri Ağı-Region Proposal Network) kullanılarak yapılır. RPN önerilen bölgeleri tespit ettikten sonra geri kalan işlemler R-CNN ile aynı şekilde yapılır.

Nesne tanıma uygulamalarındaki örnek segmantasyon problemini çözmek amacıyla geliştirilmiş olan Mask R-CNN mimarisi ilk olarak 2017 yılında Kaiming H. ve arkadaşları tarafından geliştirilmiştir (He vd, 2017). R-CNN ailesinin dördüncü üyesi olarak kabul edilen Mask R-CNN , Faster R-CNN'de tanımlanan her bir nesne için sınıf etiketi ve sınırlayıcı kutu çizimi olmak üzere iki adet çıktı varken bu iki çıktıya ek olarak nesnenin maskesini çıktı olarak veren üçüncü bir dal eklenmiş hali olarak ifade edilebilir.

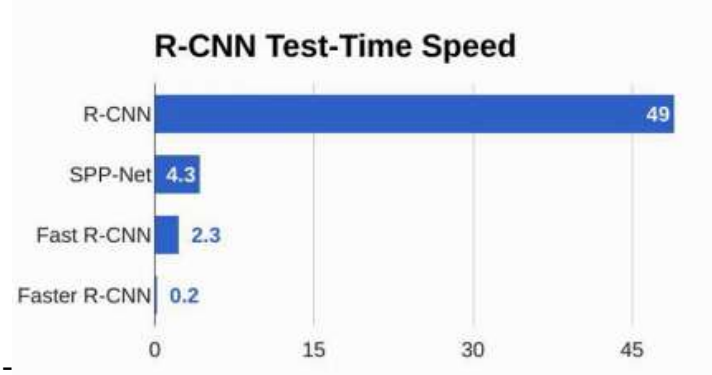
Mask R-CNN tıpkı Faster R-CNN'de olduğu gibi iki aşamalı bir prosedürü ele almaktadır. Faster R-CNN'de, aday nesneler için sınırlandırma kutularının önerildiği birinci aşama (Bölge Teklif Ağı (Region Proposal Network-RPN)) ve her bir aday sınırlandırma kutusu için ROIPool kullanarak özellik çıkarımıdır. Bu çıkarım ile sınıflandırma ve sınırlama kutusu regresyonu gerçekleştirilir. Mask R-CNN'de ise Faster R-CNN tarafından ikinci aşamada gerçekleştirilen işlemlere ek olarak her bölge için ikili bir maske oluşturulur.

Şekil 36'da Mask R-CNN temel mimari yapısı görülmektedir. Mask R-CNN'nin mimari yapısı oluşturulurken Faster R-CNN'de de yaygın olarak kullanılan ResNet ağının 50 veya 101 katmanlı ağ mimarisinin kullanılmıştır. Bu sebeple ağ mimarisi seçiminde ResNet ağ mimarilerinin kullanımı daha başarılı sonuçlar sunmaktadır (He vd, 2017).



Şekil 35. Mask R-CNN Mimari Yapısı (He vd, 2017)

Bu bölüme kadar anlatılan farklı R-CNN mimari modelleri Everigham ve arkadaşları tarafından PASCAL VOC 2007 (Everignham vd, 2007) veri seti üzerinde test edilmiş ve Şekil 37'deki sonuçlar elde edilmiştir.

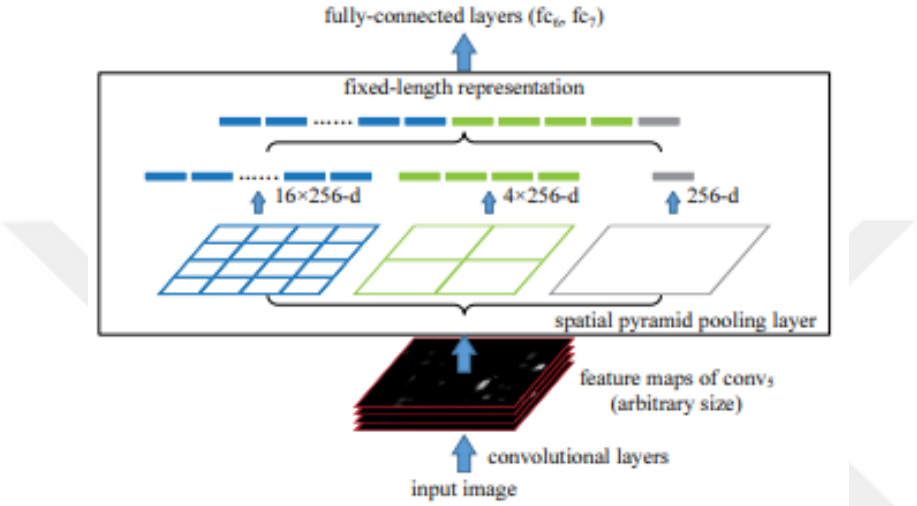


Şekil 36. PASCAL VOC 2007 veri seti üzerinde R-CNN, Fast R-CNN ve Faster R-CNN ortalama test hızları (Everignham vd, 2007)

3.2.2. SPP-Net

SPP-net ve Mekansal Piramit Havuzlama kavramı, 2014 yılında He ve arkadaşları tarafından geleneksel evrişimli sinir ağlarının bir optimizasyonu olarak önerilmiştir (He vd, 2014). Önerilen yaklaşım, görüntü boyutu ve ölçeğinden bağımsız

olarak sabit uzunluklu bir temsil üreten evrişimli sinir ağlarının sabit boyutlu girdi kısıtlamasını ortadan kaldırmayı öngörmektedir. Sadece tam bağlı katmanlarda kullanılabilen sabit boyutlu girdi kısıtlaması, evrişim katmanlarında uygulanmaz. Şekil 38'de görülen SPP CNN'leri herhangi bir boyuttaki görüntülere uyarlamak için, SPP katmanı, özellik haritalarının çıktısını yerel uzamsal kutularda havuzlayan tam bağlantılı katmanlara bir ara yüz olarak tanıtılmıştır.



Şekil 37. Mekansal piramit havuzlama katmanına sahip bir ağ yapısı (He vd, 2015).

Özellikle, R-CNN'yi test zamanında 10x'ten 100x'e ve eğitim zamanında 3x hızlandıran bir optimizasyon önerilerek, nesne algılama alanındaki SPP-net'in kullanışlılığına işaret edilmiştir. R-CNN, evrişimli bölge-öneri ileri geçişleri arasında hesaplamaları paylaşmadığından yavaş olduğundan, SPP yöntemi, paylaşılan bir evrişimli özellik haritasından çıkarılan bir özellik vektörü kullanarak her bölge önerisini sınıflandırır. Birden çok bölge teklifi çıktı özelliği boyutları havuzlanır ve bir Uzamsal Piramit Havuzlama katmanı aracılığıyla sabit boyutlu bir çıktıda birleştirilir.

3.3. Tek Aşamalı Dedektörler

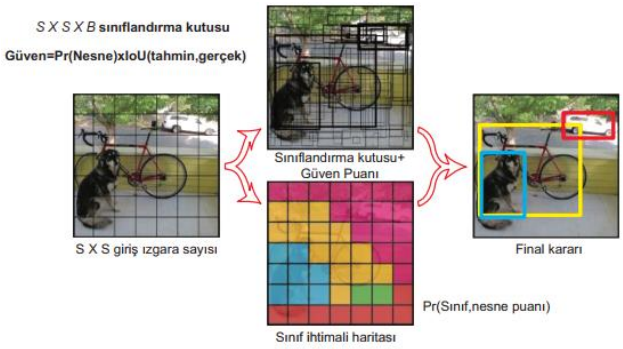
Bölge bazlı detektörler her ne kadar nesne tespiti konusundaki çalışmaların temelini oluşturmuş olsa da çok fazla kaynak ihtiyaçları ve yavaş çalışmaları sebebiyle yeni bir yaklaşımın ortaya çıkmasına sebep olmuştur. Bu yaklaşımda, bölgesel öneri aşaması olan birinci aşamayı ortadan kaldırarak doğrudan tespit aşamasını, özellik haritasındaki nesnenin olabileceği yoğun bölgelerde tespit etmeye çalışılmaktır. Bu

yaklaşımı benimseyen detektörler tek aşamalı detektörler olarak isimlendirilir. Bu detektörler bölge önerisini benimseyen yöntemlere göre daha hızlı ve basit olmasına rağmen özellikle küçük cisimlerin tespitinde bazı güçlüklere sahiptirler. Fakat nesne tespiti konusunda yine de yeterli güvene sahiptirler. Araştırmacıların çalışmalarında sıklıkla kullandığı tek aşamalı detektörler aşağıda verilmiştir.

3.3.1. YOLO (You Only Look Once)

Derin öğrenme tabanlı nesne tespit yöntemlerinin başlangıcı olarak kabul edilen YOLO (You only look once), 2016 yılından Redmod ve arkadaşları tarafından geliştirilmiştir. YOLO hızını aynı geçiş sırasında nesneye ait hem konumu hem de nesnenin sınıf bilgisini oluşturmasıyla sağlar. Prensipte YOLO, nesne tespit işlemini tek bir regresyon problemi olarak kabul eder. Bunun için bölgesel öneri adımı yerine belirli sayıdaki sınırlama kutu tahminleri yapar. Geliştirilen birinci YOLO mimari önerisinden sonra güvenilirlik ve hız konusunda ilerleme sağlamak için YOLOv1, YOLOv2, YOLOv3,YOLOv4 ve son olarak ta YOLOv5 mimarileri geliştirilmiştir.

Önerilen YOLOv5 mimarisi Perakende Ürün Tespit problemi için Ms-COCO veri seti yapısındaki SKU110K veri setini kullanmıştır. Temelde YOLO mimarisi çalışırken: ilk olarak giriş görüntüsü SxS'lik ızgaralara bölünür. Görüntü üzerindeki nesnenin merkez noktası herhangi bir ızgaranın içinde bulunuyorsa, bu ızgara nesnenin konumu ve sınıfını belirlemek için sorumlu ızgara olur. Her ızgara sınırlayıcı kutunun tahmini, güven puanın belirlenmesi ve bir nesnenin varlığına bağlı olarak nesneye ait sınıf bilgisinin bulmakla görevlidir. Nesnenin tespit edilemesiyle birlikte belirlenecek sınırlayıcı kutunun giriş görüntüsündeki konumu için kutunun nesneye göre x, y koordinatı, genişlik ve yüksekliği olmak üzere 4 parametre tanımlanır. Izgaranın içerisinde herhangi bir nesne olup olmadığı güven puanıyla belirlenir. Bu aşamalar Şekil 39'de gösterilmiştir. Eğitilmiş model, sınırlayıcı kutu sayısından bağımsız olarak, her bir ızgara için sadece bir sınıf belirlemesi yapar. En son katmanda ise evrişim katmanı tensör haline çevrilir.



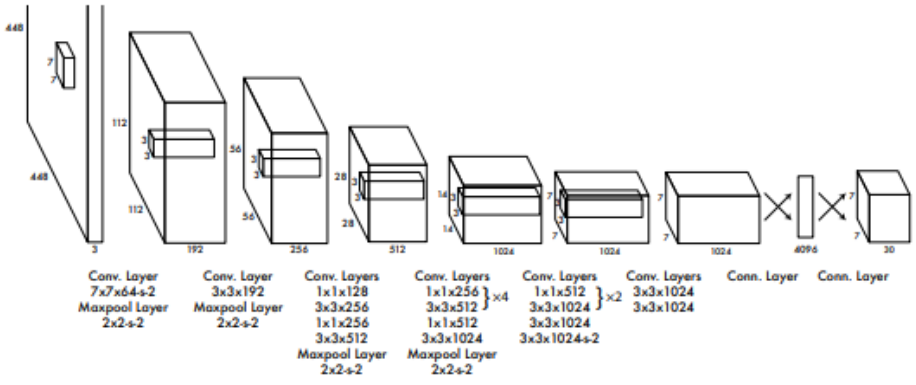
Şekil 38. YOLO İşlem Aşamaları (Redmon vd, 2016)

YOLO mimarisi belirlenen her bir ızgara için tahmin vektörleri oluşturmaktadır. Her bir tahmin vektöründe;

Güven Skoru: Bu skor modelin geçerli ızgara içinde nesne bulunup bulunmadığından ne kadar emin olduğunu gösteren,

Bağlı Sınıf Olasılığı: Modelimiz için farklı sınıf sayısı kadar tahmin değerini ifade eden, değerler bulunur.

Ağ Mimarisi: Şekil 40'ta gösterilen YOLO modeli temelde GoogleNet'e benzer yapıda olmasına rağmen ilk katman 1x1 ve 3×3 evrişim katmanları ile değiştirilmiştir. Mimarinin son bölümünde ise, birbirine bağlı iki katman tarafından evrişim özellik haritası kullanılarak nesne tespit işlemi gerçekleştirilir.



Şekil 39. Yolo Mimarisi (Mesci 2019)

Hata Fonksiyonu: YOLO mimarisinde hata oranı tespit edilen nesne için, nesnenin ne kadar doğru sınıflandırıldığının tespit edildiği sınıflandırma kaybı, tespit edilen nesnenin etrafına çizilen sınırlandırma kutusunun konumunun ne kadar doğru

olduğunu gösteren konum kaybı ve belirlenen ızgaraların içinde nesne olup olmadığını gösteren güven kaybı olmak üzere 3 temel başlıkta incelenebilir.

Sınıflandırma kaybı ızgara içerisinde nesnenin var olduğu varsayılarak Denklem-8 kullanılarak hesaplanır.

Denklem 8 Sınıflandırma Kaybı (Mesci Y,2019)

$$\sum_{i=0}^{s^2} \mathbf{1}_i^{obj} \sum_{c \in classes} (p_i(c) - \hat{p}_i(c))^2 \tag{8}$$

Konum kaybı ızgara içerisinde nesne var olduğu varsayılarak Denklem 9'daki gibi hesaplanır.

Denklem 9 Konum Kaybı (Mesci Y,2019)

$$\lambda_{coord} \sum_{i=1}^{S^2} \sum_{j=0}^{B} 1_{i\,j}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2]$$
$$+\lambda_{coord} \sum_{i=1}^{S^2} \sum_{j=0}^{B} 1_{i\,j}^{obj} \left[\left(\sqrt{w}_i - \sqrt{\hat{w}_i}\right)^2 + \left(\sqrt{h}_i - \sqrt{\hat{h}_i}\right)^2 \right] \tag{9}$$

Güven kaybı ise ızgaranın içerisinde nesne var olduğu kabul edilerek Denklem 10'daki gibi hesaplanır.

Denklem 10 Güven Kaybı - Nesne Varsa (Mesci Y,2019)

$$\sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{i\,j}^{obj} \left(C_i - \hat{C}_i\right)^2 \tag{10}$$

Güven kaybı ise ızgaranın içerisinde nesne yok olduğu kabul edilerek Denklem 11'deki gibi hesaplanır.

Denklem 11 Güven Kaybı Nesne Yoksa (Mesci Y,2019)

$$\lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{i\,j}^{noobj} \left(C_i - \hat{C}_i\right)^2 \tag{11}$$

Yukarıda anlatılmış olan üç hata fonksiyonu Denklem 12'deki gibi toplanarak genel hata fonksiyonu olarak ifade edilen fonksiyona ulaşılmış olunur.

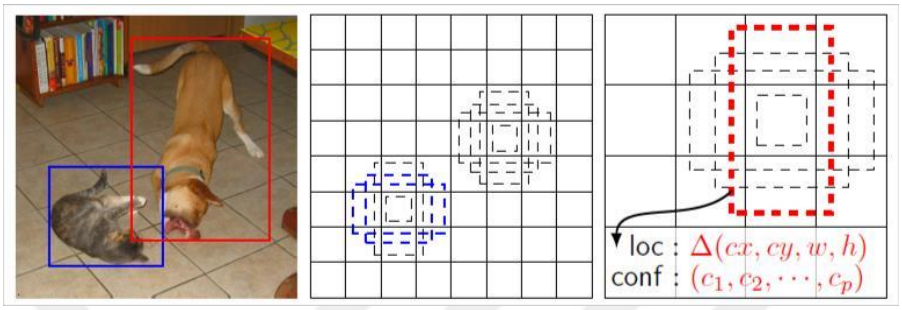
Denklem 12 Genel Hata Fonksiyonu (Mesci 2019)

$$\begin{aligned}
&\lambda_{coord} \sum_{i=1}^{S^2} \sum_{j=0}^{B} 1_{i\,j}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] \\
&+\lambda_{coord} \sum_{i=1}^{S^2} \sum_{j=0}^{B} 1_{i\,j}^{obj} \left[\left(\sqrt{w}_i - \sqrt{\hat{w}_i}\right)^2 + \left(\sqrt{h}_i - \sqrt{\hat{h}_i}\right)^2 \right] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{i\,j}^{obj} \left(C_i - \hat{C}_i\right)^2 \\
&+\lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{i\,j}^{noobj} \left(C_i - \hat{C}_i\right)^2 \\
&+ \sum_{i=0}^{S^2} \mathbf{1}_i^{obj} \sum_{\boldsymbol{c} \in \boldsymbol{classes}} (\boldsymbol{p_i}(\boldsymbol{c}) - \hat{\boldsymbol{p}}_{\boldsymbol{i}}(\boldsymbol{c}))^2
\end{aligned} \tag{12}$$

3.3.2. SSD (Single Shot MultiBox Detector)

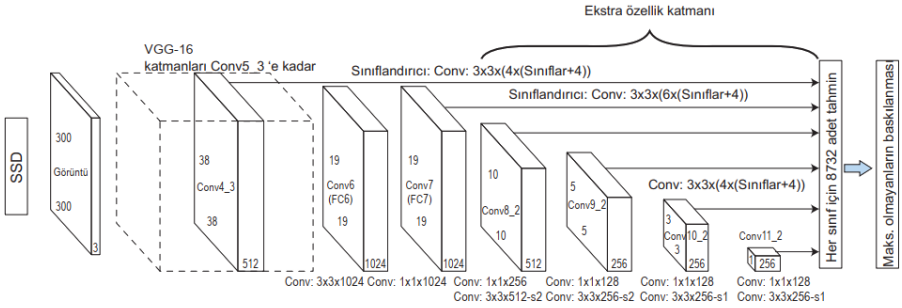
SSD tek aşamalı bir nesne tespit yöntemidir ve temelleri Multibox'a dayanmaktadır (Erhan vd, 2014). SSD nesne tespit işlemini tek geçişte tamamlamaya çalışan ve giriş görüntüsü üzerinden nesnelerin sınıflarını temel bir sınıflandırma mimarisi kullanan bir yapıdır. SSD giriş görüntüsünde nesneleri tanımlamak için, ilk olarak Şekil 41'deki gibi farklı ölçülerde öznitelik haritaları oluşturur. İkinci adımda çıkardığı öznitelik haritalarındaki her noktaya belirli boyutlarda sınırlayıcı kutular

(Default Box) yerleştirir. En son adımda ise yerleştirdiği her bir kutu için ayrı ayrı işlem yaparak nesne tanımlama işlemini gerçekleştirir. SSD'nin önerilen orijinal versiyonunda CNN mimari olarak VGG-16 kullanılmasına rağmen VGG-16 yerine ResNet ve ResNet + FPN kullanan versiyonları da araştırmacılar tarafından kullanılmıştır



Şekil 40. SSD'nin Nesne Tanıma Yapısı (Liu vd, 2016)

Aşağıda görülen Şekil 42'de SSD'nin mimari yapısı görülmektedir. VGG16 mimarisinde giriş katmanı 256 x 256 iken Şekil 41'de görülen mimari incelendiğinde SSD mimarisinde giriş katmanı 300×300 olarak seçilmiş olmasına rağmen temel olarak VGG16 mimarisine oldukça benzemektedir. SSD tarafından VGG16'ya ait olan 38 × 38 × 512 ebatlarındaki Conv4_3 katmanından sonra 19 × 19 × 1024 olan katmanlar ilave edilen katmanlardır. Bu katmanlardan sonra nesnenin sınıf ve konumuna ait karar vermemizi sağlayan ek evrişim katmanları ile mimari sonuçlandırılmıştır.



Şekil 41. SSD algoritmasının mimari yapısı (Liu vd, 2016)

SSD mimarisinin kendisine test olarak verilen görüntüdeki nesnelerin konum ve sınıf bilgilerini bilmesi mümkün değildir. Bunun için eğitim sırasında belirlenin filtreler oldukça önemlidir. Çünkü SSD ancak eğitim sırasında kazandığı yeteneklerle sonuca gidebilecektir. Genelde SSD mimarisi kullanılarak bir modelin eğitimi ve bu eğitim için kullanılacak verilerin hazırlanması oldukça fazla zaman alacağı için

araştırmacılar çalışmalarında kullanmak için ellerinde yeterli donanım ve veri seti olan teknoloji şirketlerinin hazırladığı ücretsiz eğitim verilerini kullanmaktadır. Ancak burada dikkat edilmesi gereken nokta hazır olarak elde edilen bu verilerdeki sınıfların çalışmamızda kullanacağımız sınıflarla ilişkili olmasıdır. SSD'nin orijinal referans sürümü Pascal-Voc veri setiyle hazırlandığından dolayı hazırlanan veri setinin de Pascal-Voc veri seti şablonunda olması önemlidir. Eğer bu şablona uyulmazsa eğitim işlemi gerçekleşmeyecektir.

DÖRDÜNCÜ BÖLÜM

4. DENEYSEL BULGULAR (UYGULAMA)

Perakende ürün algılama senaryosu için daha önceden hazırlanmış ve araştırmacılar için ücretsiz olarak kullanıma açılmış olan SKU110k veri kümesi tercih edilmiştir. Bu veri kümesi çalışmaya uygun olacak şekilde Roboflow ara yüzü kullanılarak ön işlemden geçirilmiş bozuk olan görüntüler veri kümesinden çıkarılmıştır. Veri kümesi daha önceden belirlemiş olduğumuz YOLOv5 ve Fast R-CNN nesne tespit yöntemleri için yeniden düzenlenmiştir. Çalışmanın donanım ihtiyacını karşılamak için ColabPro kullanılmıştır. Her iki nesne tespit yöntemi için yapılmış olan tüm işlemler aşağıda ayrıntılı olarak verilmiştir.

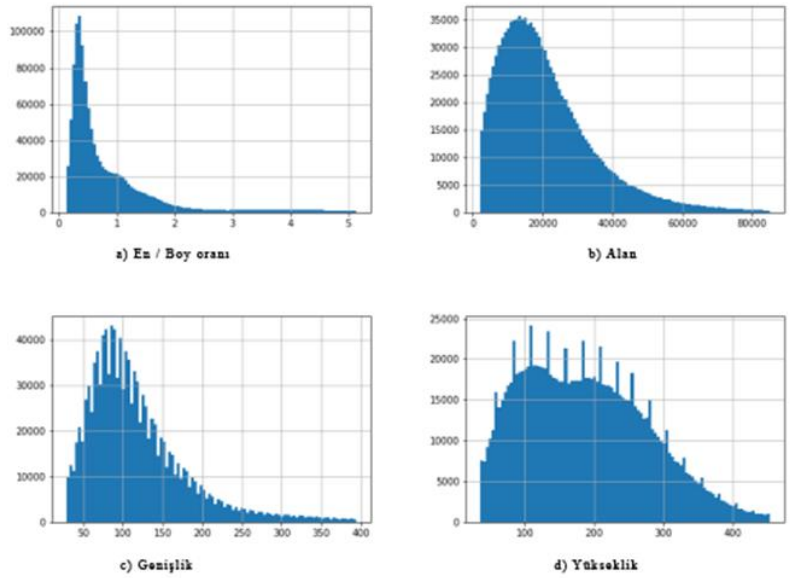
4.1. Veri Setinin Hazırlanması

Perakende ürün algılama senaryosu, milyonlarca olası özellik ve görüntü başına yüzlerce nesnenin bulunduğu oldukça zor bir durumun varlığını görmemizi sağlamıştır. Bu zor ve karmaşık durumun üstesinden gelmek için içerisinde neredeyse dünyanın her yerindeki birçok süpermarket rafına ait 11.762 raf görüntüsü barındıran ve yalnızca ticari olmayan akademik çalışmalar için kullanıma sunulan SKU110k veri kümesi kullanılmıştır.

SKU110k veri kümesi, market raflarına yerleştirilmiş perakende nesnelerin görüntülerinden oluşmaktadır. Hazırlanmış olan bu veri seti içerisinde, eğitim, doğrulama ve test seti görüntülerinin yanı sıra görüntülerdeki tüm nesnelerin sınırlayıcı kutu konumlarına ilişkin bilgileri içeren bir ".csv" dosyası da temin edilebilmektedir.

SKU110k veri setinde, test için 2347 görsel, eğitim için 8232 görsel ve doğrulama içinde 587 görsel bulunmaktadır. Her görüntü farklı sayıda nesneyi barındırabileceğinden görsellerdeki sınırlayıcı kutu sayılarının da farklı olacağı unutulmamalıdır.

Trax tarafından kullanıma sunulan SKU110K veri seti incelendiğinde görüntü boyutlarının MS-COCO ve Pascal-VOC veri setlerinden farklı olduğu görülmüştür. Bu durum sebebiyle orijinal veri kümesindeki görüntülerin boyutları yeniden düzenlenmiştir. Şekil 43'te orijinal görüntülere ait temel bilgiler verilmiştir.



Şekil 42. SKU110K veri seti görüntü boyutları (Kozlov 2020)

Çalışmada kullanılacak olan veri setini hazırlamak için Roboflow platformu kullanılmıştır. Roboflow platformuna uygulamada kaynak olarak kullanılacak SKU110k veri seti ve veri setindeki görüntülerin sınırlayıcı koordinatlarını içeren ".csv" dosyası yüklenmiştir. Bu işlem adımından sonra yüklenen görüntüler Roboflow tarafından ön işleme tabi tutularak 416x416x3 boyutlarında yeni görüntüler elde edilmiştir. Ön işleme tabi tutulan görüntü örneği Şekil 44'teki gibidir.

Şekil 43. Roboflow Tarafından İşlenmiş Görüntü

Roboflow platformunda yeniden oluşturulan veri seti için yüklenmiş olan SKU110k veri seti %70 eğitim (8232 resim), %20 test (2347 resim) ve %10 (587 resim) doğrulama kümesi olarak ayarlanmıştır. Çalışmada seçilen ilk derin öğrenme yöntemi YOLOv5 için yeni veri seti Ms-COCO veri kümesi yapısında yeniden hazırlanmıştır. Aynı şekilde çalışmada seçilen ikinci derin öğrenme yöntemi Faster R-CNN için veri seti Keras Retinanet veri kümesi yapısına yeniden düzenlenmiştir.

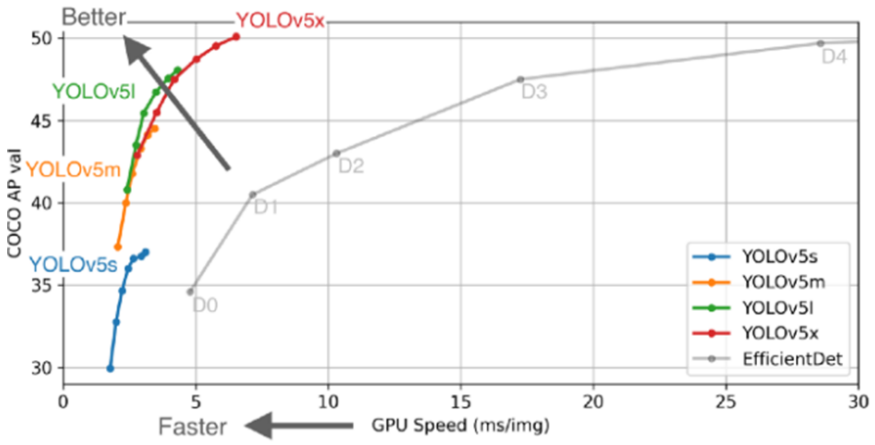
4.2. YOLOv5 Modelinin Uygulanması

4.2.1. Çalışma Ortamı

Derin öğrenme kullanılarak perakende ürün tespit uygulaması için çalışma ortamı olarak Google Colab Pro bulut hizmetinden faydalanılmıştır. Yapay Zekâ (AI) ve Derin Öğrenme alanlarındaki araştırma ve öğrenmeyi desteklemek için Google'ın sunduğu Google Colab Pro (Google Collaboratory) bir Entegre Geliştirme Ortamıdır (IDE). Google Colab, Grafik İşlem Birimi (GPU) ve Tensör İşlem Birimi (TPU) kullanmaya olanak sağlayan ve Jupyter Notebook yapısında bir kod ortamı sağlamaktadır. Google Colab; derin öğrenme araştırmalarında sıklıkla kullanılan, PyTorch, TensorFlow, Keras ve OpenCV gibi kütüphaneleri önceden yükleyerek çalışma ortamına dâhil etmektedir. Bu çalışmada GPU ve TPU desteğinden tam olarak faydalanabilmek ve notebook çalışma zamanını etkili bir biçimde kullanabilmek için Google Colab'ın ücretsiz sürümü yerine aylık ücretli sürümü kullanılmıştır.

Google Colab Pro 24 saate kadar çalışma süresi, 25GB ram,150GB saklama alanı ve 16GB GPU (Tesla P100) destekli bulut hizmeti ile çalışmadaki modelin eğitimi ve test edilmesi sırasında ihtiyaç duyulan donanım ortamını sağlamıştır.

YOLO modelleri, esas olarak Alexey Bochkovsky tarafından C dilinde yazılmış özel bir çerçeve üzerinde geliştirilmiştir. YOLOv4 yayınlandıktan kısa biri süre sonra araştırmacı Glenn JOCHER ve ekibi, YOLO ailesinin YOLOv5 adlı yeni bir versiyonunu yayınlamıştır. (Jocher G, 2020). Perakende ürün tespit uygulamasında tek seferde nesne tespiti yapan YOLO modellerinin en son geliştirilen YOLOv5 sürümü tercih edilmiştir. Ultralytics tarafından geliştirilen YOLOv5 dört alt sürümden oluşmaktadır (s,m,l,x) Ultralystic, YOLO'nun önceki sürümlerini derin öğrenme alanındaki en ünlü çerçevelerden biri olan Python dilinde yazılmış PyTorch'a dönüştüren şirkettir. YOLOv5 diğer sürümlerin aksine C dili yerine Python diliyle yazılmış ve PyTorch desteği sayesinde gelişmeye açıktır. Şekil 45'te Ultralytic tarafından geliştirilen YOLOv5'in COCO dataset üzerindeki başarımları görülmektedir.



Şekil 44. YOLOv5'in Ms-COCO Dataset Üzerindeki Başarımları

YOLOv5 modeli ile ilgili Jocher'in hazırladığı repo incelendiğinde Jocher'in modele ait yapı kodunu bir yaml dosyasında tuttuğu görülecektir. Bu yaml dostasından YOLOv5 modeli aşağıdaki gibi özetlenebilir (Jocher 2020):

- Model Backnone: Odak yapısı, CSP ağı
- Model Neck: SPP bloğu, PANet
- Model Head: GIoU-loss kullanan YOLOv3 head

Model Backbone (Model omurgası) esas olarak verilen girdi görüntüsünden önemli özellikleri çıkarmak için kullanılır. YOLOv5, bir girdi görüntüsünden özellik çıkarmak için CSP'yi (Çapraz Aşamalı Kısmi Ağlar) bir omurga olarak kullanmaktadır.

Model Neck (Model Boynu) esas olarak özellik piramitleri oluşturmak için kullanılır. Özellik piramitleri, modelin aynı nesnenin farklı boyut ve ölçeklerdeki görüntülerini tanımasına yardımcı olur. Özellik piramitleri ve modellerin görünmeyen veriler üzerinde iyi performans göstermesi açısından oldukça kullanışlıdırlar. Literatürde FPN, BiFPN, PANET gibi özellik piramit tekniklerinin farklı türlerini kullanan diğer modeller vardır (Hui 2018). YOLOv5'te piramit özelliklerinin elde edilmesi için PANet tekniği, kullanılmaktadır.

Model Head (Model Kafası), esas olarak son algılama bölümünü gerçekleştirmek için kullanılmaktadır. Model tarafından belirlenen özellikler için sınıf olasılıkları, nesnellik puanları ve sınırlayıcı kutularla nihai çıktı vektörleri modelin bu bölümünde gerçekleştirilmektedir. YOLOv5'te modelin head bölümü, önceki YOLOv3 ve YOLOv4 sürümleriyle aynı yapıda kullanılmıştır.

Glenn Jocher çalışmasında yaptığı en temel değişiklik, sınırlayıcı kutu seçim sürecini YOLOv5'e entegre etmeyi önermiş olmasıdır. Bu sayede, sinir ağı veri kümelerinin hiçbirini girdi olarak kullanmak zorunda kalmayacak, veri kümesi için en iyi sınırlayıcı kutuları otomatik olarak öğrenerek bunları eğitim sırasında kullanabilecektir (Solawetz 2020).

Çalışmada seçilen YOLOv5' modeli içi YOLOv5 yazarlarının seçmiş oldukları Leaky ReLU ve Sigmoid aktivasyon fonksiyonları aynen kullanılmıştır. YOLOv5'te Leaky ReLU aktivasyon fonksiyonu orta / gizli katmanlarda kullanılırken, sigmoid aktivasyon fonksiyonu son tespit katmanında kullanılmaktadır (Rajput 2020).

YOLOv5 yazarları araştırmacılara optimizasyon işlevi için SGD ve ADAM olmak üzere iki optimizasyon seçeneği sunmuşlardır. Yapılmış olan bu çalışmada YOLOv5 modelinin eğitimi için SGD optimizasyon işlevi tercih edilmiştir.

Şu ana kadar geliştirilen tüm YOLO modellerinde, nesnellik puanı, sınıf olasılık puanı ve sınırlayıcı kutu regresyon puanına göre hesaplanan bileşik bir kayıp olduğu kabul edilmektedir (Rajput 2020).

Ultralytics, sınıf olasılığının ve nesne puanının kayıp hesaplaması için PyTorch'un Logits Loss işleviyle İkili Çapraz Entropi'yi kullanmıştır. Yapılan bu çalışmada sınıf olasılığı ve nesne puanı için aynı hesaplama yöntemleri kullanılmıştır.

4.2.2. İlişkilerin Kurulması ve kütüphanelerin Yüklenmesi

Çalışmada YOLOv5 modeli kullanarak perakende ürün tespit uygulaması Colab Notebook Pro ortamında geliştirilirken Tablo 1'de belirtilen kütüphanelerin ve Ultralytics tarafından sağlanan reponun Colab Notobook Pro'ya dahil edilmesi gerekmektedir.

Tablo 1. YOLOv5 için Gerekli Kütüphaneler ve Repo

Kütüphane adı	Repo
matplotlib>=3.2.2	https://github.com/ultralytics/yolov5
numpy>=1.18.5	
opencv-python>=4.1.2	
Pillow	
PyYAML>=5.3.1	
scipy>=1.4.1	
torch>=1.7.0	
torchvision>=0.8.1	
tqdm>=4.41.0	
tensorboard>=2.4.1	
# wandb	
seaborn>=0.11.0	
Pandas	
# coremltools>=4.1	
# onnx>=1.8.1	
# scikit-learn==0.19.2	
Thop	
pycocotools>=2.	

Çalışmada kullanılacak orijinal repo ve kütüphaneler Şekil 46'daki komutlar kullanılarak Colab Notebook Pro'ya dâhil edilmiştir.

```
!git clone https://github.com/ultralytics/yolov5 # clone repo
!pip install -r yolov5/requirements.txt  # install dependencies
%cd yolov5

import torch
from IPython.display import Image, clear_output  # to display images
from utils.google_utils import gdrive_download  # to download models/datasets

clear_output()
print('Setup complete. Using torch %s %s' % (torch.__version__, torch.cuda.get_de

Setup complete. Using torch 1.8.1+cu101 _CudaDeviceProperties(name='Tesla V100-SX
```

Şekil 45. Gerekli Kütüphane ve Repo Yükleme

4.2.3. Veri Setinin Yüklenmesi

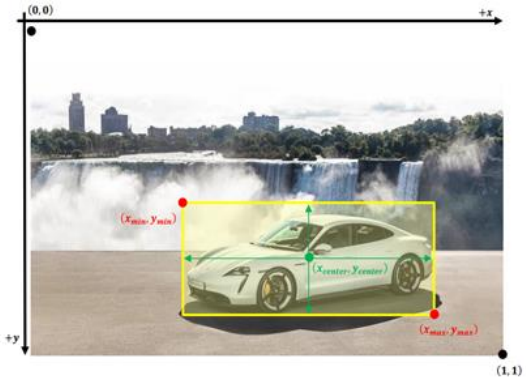
Çalışmanın bu bölümünde Roboflow platformu kullanılarak daha önceden hazırlanmış olan veri seti (Train, Test, Val) ve bu veri setindeki görüntülere ait açıklamaların bulunduğu *".csv"* uzantılı dosyaların notebookun çalışma zamanına dâhil edilmesi gerekmektedir. Bu işlem için Şekil 47'deki komutlar kullanılmıştır.

```
# Export code snippet and paste here
%cd /content
robo_flow_data_download_key = "ENTER YOUR DATA DOWNLOAD LINK HERE IN QUOTES"
!curl -L "https://app.roboflow.com/ds/CKL0opANMQ?key=NNP2HO5y8O" > roboflow.zip;
```

Şekil 46. İşlenmiş SKU110K Veri Setinin Yüklenmesi

Çalışmada kullanılan veri setindeki görüntülere ait açıklama bilgileri *".csv"* uzantılı bir dosyada tutulmaktadır. Bu açıklama dosyasında her görüntüdeki her bir nesne için sınırlayıcı kutu verileri $[class, xcenter, ycenter, width, height]$ formatında saklanmaktadır.

YOLO modellerinin tamamı için örnek sınırlayıcı kutu biçimi Şekil 48'deki gibidir.



Şekil 47. YOLO Modelleri için Sınırlayıcı Kutu Yapısı

YOLOv5 modeli Pytorch kullanarak verilere erişmek ve yapay sinir ağına girdi olarak kullanabilmek için kullanılacak veri setine ait özet bilgileri içeren bir .yaml dosyasına ihtiyaç duymaktadır. YOLO modelinde kullanılan data.yaml dosyası aşağıdaki yapıya sahiptir:

1. Train: "eğitim seti dizin yolu"

2. Val: "doğrulama seti dizin yolu"

3. Nc: "sınıf sayısı"

4. Name: "nesnelerin adı"

Çalışmada kullanılacak veri seti YOLOv5 modeli için bir *".yaml"* dosyası barındırmadığından Şekil 49'daki komutlar yardımıyla çalışma zamanında bir "*data.yaml*" dosyası oluşturulmuştur.

```
[ ]  # this is the YAML file Roboflow wrote for us that we're
     %cat data.yaml

     train: ../train/images
     val: ../valid/images

     nc: 1
     names: ['object']
```

Şekil 48. Data.yaml Dosyası

4.2.4. Eğitim Modeli İçin Mimari Hazırlanması

Çalışmada kullanılacak olan YOLOv5 modeli için PyTorch farklı nesne algılama sorunlarını ortadan kaldırmak için .yaml uzantılı bir model mimarisine ihtiyaç duymaktadır. Araştırmacı Glenn JOCHER ve ekibi daha önceki YOLO modellerine benzer bir mimariyi çalışmalarında kullanıma sunmuşlardır. Ultralystick tarafından kullanıma sunulan model mimarisi Şekil 50 'de görülmektedir.

```
[43] %cat /content/yolov5/models/yolov5s.yaml

     # parameters
     nc: 80  # number of classes
     depth_multiple: 0.33  # model depth multiple
     width_multiple: 0.50  # layer channel multiple

     # anchors
     anchors:
       - [10,13, 16,30, 33,23]  # P3/8
       - [30,61, 62,45, 59,119]  # P4/16
       - [116,90, 156,198, 373,326]  # P5/32

     # YOLOv5 backbone
     backbone:
       # [from, number, module, args]
       [[-1, 1, Focus, [64, 3]],  # 0-P1/2
        [-1, 1, Conv, [128, 3, 2]],  # 1-P2/4
        [-1, 3, C3, [128]],
        [-1, 1, Conv, [256, 3, 2]],  # 3-P3/8
        [-1, 9, C3, [256]],
        [-1, 1, Conv, [512, 3, 2]],  # 5-P4/16
        [-1, 9, C3, [512]],
        [-1, 1, Conv, [1024, 3, 2]],  # 7-P5/32
        [-1, 1, SPP, [1024, [5, 9, 13]]],
        [-1, 3, C3, [1024, False]],  # 9
       ]

     # YOLOv5 head
     head:
       [[-1, 1, Conv, [512, 1, 1]],
        [-1, 1, nn.Upsample, [None, 2, 'nearest']],
        [[-1, 6], 1, Concat, [1]],  # cat backbone P4
        [-1, 3, C3, [512, False]],  # 13
```

Şekil 49. Örnek YOLO Mimarisi

Hazırladığımız Perakende ürün tespit uygulamasında YOLOv5 (s-m-l-x) modelinin performansını değerlendirmek istediğimizden çalışma ortamımıza modelimize ait mimari belirlenirken Ultralystick tarafından sağlanan mimariye bağlı kalınmıştır. Bu mimariyi çalışma zamanına dâhil etmek için Şekil 51'deki komutlar kullanılmıştır.

```
[ ]  #customize iPython writefile so we can write variables
     from IPython.core.magic import register_line_cell_magic

     @register_line_cell_magic
     def writetemplate(line, cell):
         with open(line, 'w') as f:
             f.write(cell.format(**globals()))
```

```
[ ]  %%writetemplate /content/yolov5/models/custom_yolov5s.yaml

     # parameters
     nc: {num_classes}  # number of classes
     depth_multiple: 0.33  # model depth multiple
     width_multiple: 0.50  # layer channel multiple

     # anchors
     anchors:
       - [10,13, 16,30, 33,23]  # P3/8
       - [30,61, 62,45, 59,119]  # P4/16
       - [116,90, 156,198, 373,326]  # P5/32
```

Şekil 50. YOLO Mimarisini Çalışma Zamanına Dâhil Edilmesi

4.2.5. YOLOv5 Modelin Eğitimi

Hazırlamış olduğumuz Perakende Ürün Tespit uygulamasının bu bölümünde YOLOv5 modelinin ürün tespit performansını karşılaştırarak YOLOv5 modellerinden hangisinin daha iyi sonuç vereceği araştırılmaya çalışılmaktadır. Bunun için YOLOv5 modelinin s (small), m (midium), l (large) ve x (extendet) versiyonları ayrı ayrı eğitilmiş ve sonuçları karşılaştırılmıştır. Modellerin eğitim sonuçlarını görselleştirmek için akademik çalışmalar için ücretsiz hizmet veren wand platformu çalışma zamanına dâhil edilerek sonuçların görsel açıdan ifade edilmesi sağlanmıştır.

YOLOv5 modeline ait farklı versiyonları için eğitim kodları Şekil 52'deki gibidir.

Şekil 51. YOLOv5 (s-m-l-x) Model Eğitimleri

Yapılan çalışmada YOLOv5 modelinin tüm versiyonlarını karşılaştırabilmek için eğitimin aynı tur sayısında yapılmasına dikkat edilmiştir. Eğitim sırasında kullanılan komutların yapısı tüm YOLOv5 versiyonları için aynıdır. Bu komutlardaki parametrelere bakacak olursak;

img: Eğitilecek model için giriş görüntü boyutları bu parametreyle belirlenir. Bilgisayarlı görü problemi konusunda çalışan birçok araştırmacı giriş görüntüsünü üzerinde çok fazla ayrıntı kaybetmeden kullanabilmek için 416x416 görüntü boyutunun ideal olduğunu düşünmektedir. Bu çalışmada da görüntü boyutu olarak 416 belirlenmiştir.

batch: Eğitilecek model için parti boyutu bu parametrede belirlenmektedir. Eğitim sırasında tüm görüntülerin aynı anda sinir ağına iletilmesi modelin öğrendiği ağırlıkların çok fazla olmasına sebep olacaktır. Bu sebeple eğitim sırasında görüntüler belirli gruplara ayrılmalıdır. Her grup sırasıyla sinir ağına iletilerek ayrı ayrı ağırlıklar hesaplanır ve tüm gruplar işlendikten sonra ağırlıklar toplanarak sonuç elde edilir. Parti boyutu belirlenirken büyük parti boyutlarının çok fazla bellek tüketeceği unutulmamalıdır. Yapılmış olan bu çalışmada eğitim seti için 8232 görüntü ayrılmış ve parti boyutu 32 olarak belirlenmiştir. Sonuç olarak çalışmadaki toplam grup sayısı YOLOv5s için 8232 /64 = 128, YOLOv5m için 8232/48= 171, YOLOv5l için 8232/32=257 ve YOLOv5x için 8232/16=514 olduğu görülmektedir.

epochs: Eğitilecek model için eğitim tur sayısının tanımlandığı parametredir. Bir tur, modelin bir parti boyutu için tüm grup girdilerini eğitmekle sorumlu yapı olarak ifade edilebilir. Tur sayısı modelin tüm girdileri çalıştırma sayısını temsil eder ve kesin referans etiketlerine yaklaşmak için ağırlıkları günceller. Yapılan çalışmada tüm sürümler için 300 tur sayısı belirlenmiştir.

data: Eğitilecek model için daha önceden hazırlanmış veri setinin özetini içeren *data.yaml* dosyasının yolu bu parametreyle belirlenir.

cfg: Eğitilecek model için model mimarisini içeren yapılandırma dosyasının yolunun belirtildiği parametredir. Eğitim sırasında giriş görüntüleri bu mimariye uygun olarak derlenir ve eğitime dâhil edilir.

weights: Eğitilecek model için eğitim süresinden tasarruf etmek için önceden eğitilmiş bir ağırlık kullanılmasını sağlayan parametredir. Bu parametre boş bırakılırsa, model eğitim için rastgele ağırlıklar belirleyeceğinden eğitim süresi artacaktır.

name: Eğitim sonucunun kaydedileceği klasör adının belirlendiği parametredir. Model, eğitim sırasında gerçekleştirilen tüm sonuçları içeren bir dizin oluşturacak ve tüm sonuçları buraya kaydedecektir.

cache: Eğitilecek modelde eğitim hızını arttırmak için giriş görüntülerinin ön belleğe alınıp alınmayacağının belirlendiği parametredir.

device: Model eğitiminin hangi aygıt kullanarak yapılacağının belirlendiği parametredir. Modelin eğitimi sırasında CPU kullanılacaksa parametre "CPU" değerini, GPU kullanılacaksa "0" değerini almalıdır.

Perakende ürün tespit uygulaması için daha önceden hazırlan veri seti tüm YOLOv5 versiyonları için ayrı ayrı Colab Pro Tesla P100 çalışma ortamında aynı tur sayıyla eğitilmiş ve eğitim sonucunda eğitilen modele ait ağırlıklar ayrı ayrı kaydedilmiştir. Bu ağırlıklar ve eğitim sürelerini içeren bilgiler Tablo 2'de gösterilmiştir.

Tablo 2. YOLOv5 Modelleri Eğitilmiş Ağırlıklar ve Eğitim Süreleri

Model	Eğitilmiş Model Boyutu	Eğitim Süresi
YOLOv5s / Ms COCO	14,02MB	6s 51d 37sn
YOLOv5m / Ms COCO	41,45MB	11s 9d 51sn
YOLOv5l / Ms COCO	91,5MB	17s 30d 15sn
YOLOv5x / Ms COCO	170,94MB	23s 36d 47sn

Tablo 2 incelendiğinde eğitim süresi ve eğitilmiş ağırlıkların dosya boyutu en az olan YOLOv5s modeli en çok olanınsa YOLOv5x modeli olduğu görülmektedir.

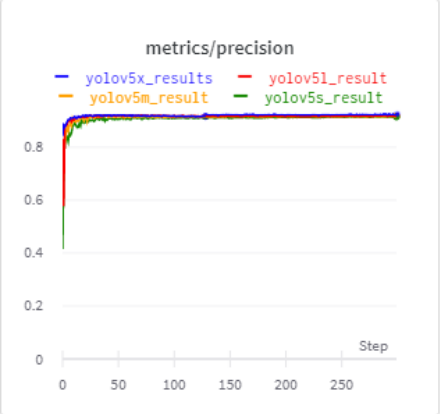
Perakende ürün tespit uygulaması için SKU110k veri seti üzerinde seçilmiş YOLOv5 modelinin tüm versiyonları için yapılan model eğitim sonuçları Tablo 3'te verilmiştir. Tablo 3'te, **P:** Precission (Hassasiyet) ,**R :** Recall (Geri Çağırma) **.mAP:** Ortalama doğruluk oranı olarak belirtilmiştir.

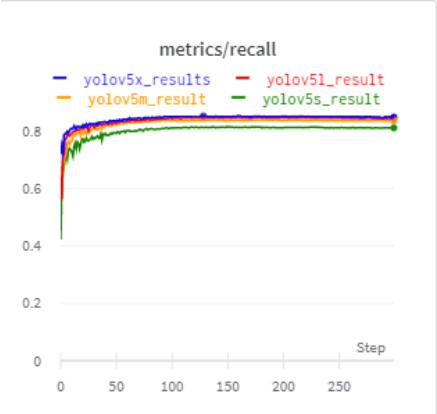
Tablo 3. YOLOv5 Model Eğitim Sonuçları

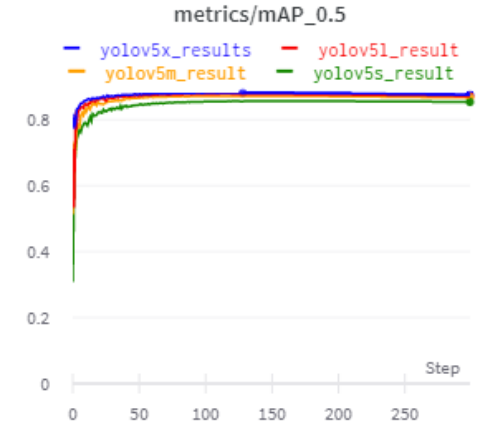
Model	P	R	mAP 0.5	mAP 0.5:0.95	Train Box loss	Train Obj loss	Val Box loss	Val Obj loss
YOLOv5s / Ms COCO	**0.913**	**0.812**	**0.855**	**0.497**	**0.053**	**0.321**	**0.051**	**0.279**
YOLOv5m / Ms COCO	0.914	0.838	0.867	0.517	0.050	0.301	0.050	0.280
YOLOv5l / Ms COCO	0.917	0.846	0.872	0.528	0.048	0.291	0.049	0.279
YOLOv5x / Ms COCO	**0.918**	**0.849**	**0.875**	**0.532**	**0.047**	**0.285**	**0.049**	**0.279**

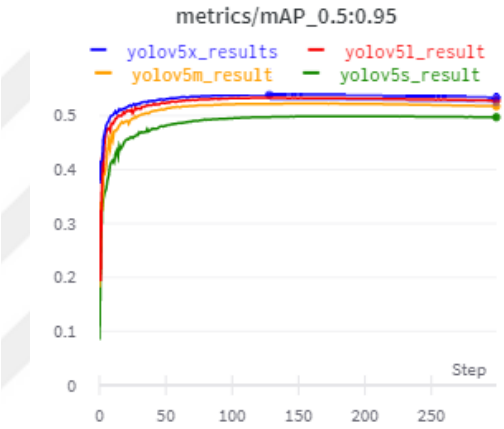
Tablo 3 incelendiğinde 0.5'lik sınırlayıcı kutu eşiği için, YOLOv5s modeli 8232 girdi görüntüsü, 64 parti boyutu ve 300 tur sayısı için %85.5 doğruluk oranıyla, YOLOv5m modeli 8232 girdi görüntüsü, 48 parti boyutu ve 300 tur sayısı için %86.7 doğruluk oranıyla, YOLOv5l modeli 8232 girdi görüntüsü, 32 parti boyutu ve 300 tur sayısı için %87.2 doğruluk oranıyla, YOLOv5x modeli 8232 girdi görüntüsü, 16 parti boyutu ve 300 tur sayısı için %87.5 doğruluk oranıyla görüntü üzerindeki nesneleri tespit edebildiği görülmektedir.

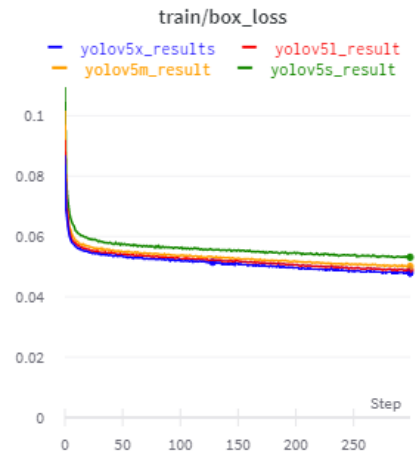
Elde edilen sonuçlar göz önünde bulundurulduğunda YOLOv5x modeli eğitim süresi uzun olması ve eğitilmiş ağırlıkların boyutu yüksek olmasına rağmen en yüksek hassasiyette sonuç verdiği görülmüştür. Elde edilen tüm sonuçlar grafiksel olarak Şekil 53'te gösterilmiştir.

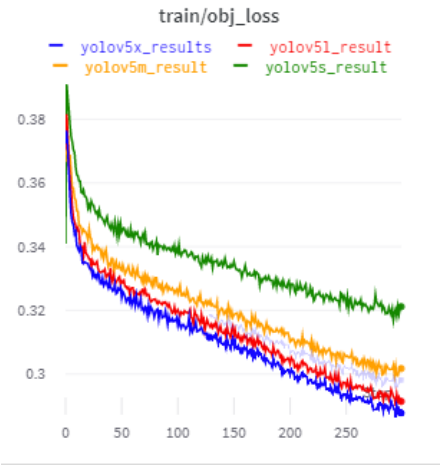

metrics/precision
yolov5x_results
yolov51_result
yolov5m_result
yolov5s_result
0.8
0.6
0.4
0.2
0
Step
0
50
100
150
200
250


metrics/recall
yolov5x_results
yolov51_result
yolov5m_result
yolov5s_result
0.8
0.6
0.4
0.2
0
Step
0
50
100
150
200
250


metrics/mAP_0.5
yolov5x_results
yolov51_result
yolov5m_result
yolov5s_result
0.8
0.6
0.4
0.2
0
Step
0
50
100
150
200
250


metrics/mAP_0.5:0.95
yolov5x_results
yolov51_result
yolov5m_result
yolov5s_result
0.5
0.4
0.3
0.2
0.1
0
Step
0
50
100
150
200
250


train/box_loss
yolov5x_results
yolov51_result
yolov5m_result
yolov5s_result
0.1
0.08
0.06
0.04
0.02
0
Step
0
50
100
150
200
250


train/obj_loss
yolov5x_results
yolov51_result
yolov5m_result
yolov5s_result
0.38
0.36
0.34
0.32
0.3
0
50
100
150
200
250

Şekil 52. YOLOv5 Eğitim Sonuçlarının Grafik Gösterimi

Eğitim sonucunda elde edilen grafikler incelendiğinde YOLOv5s modelinin 0.5'lik sınırlayıcı kutu eşiğinde geri çağırma değeri (R) 0.812 ve otalama doğruluk oranı değeri (mAP0.5) 0.855, YOLOv5m modelinin 0.5'lik sınırlayıcı kutu eşiğinde geri çağırma değeri (R) 0.838 ve otalama doğruluk oranı değeri (mAP0.5) 0.867, YOLOv5l modelinin 0.5'lik sınırlayıcı kutu eşiğinde geri çağırma değeri (R) 0.846 ve otalama doğruluk oranı değeri (mAP0.5) 0.872 ve YOLOv5x modelinin 0.5'lik sınırlayıcı kutu eşiğinde geri çağırma değeri (R) 0.849 ve otalama doğruluk oranı değeri (mAP0.5) 0.875 olduğu görülmüştür.

4.2.6. Seçilen Modellerin Test Edilmesi

Yapılan Perakende ürün tespit uygulaması için seçilen YOLOv5s ve YOLOv5x modelleri aynı donanımsal özelliklere sahip Colab Pro Notebook ortamında ve aynı test veri üzerinde Şekil 54'teki komutlar yardımıyla test edilmiştir.

```
#yolov5s Result (0,0,255)
%cd /content/yolov5/
!python detect.py --weights /content/yolov5/runs/train/yolov5s_results/weights/last.pt --img 416 --conf 0.4
--source ../test/images
```

```
#yolov5x Result (255,0,255)
%cd /content/yolov5/
!python detect.py --weights /content/yolov5/runs/train/yolov5x_results/weights/last.pt --img 416 --conf 0.4
--source ../test/images
```

Şekil 53. YOLOv5s ve YOLOv5x Modellerinin Test Edilmesi

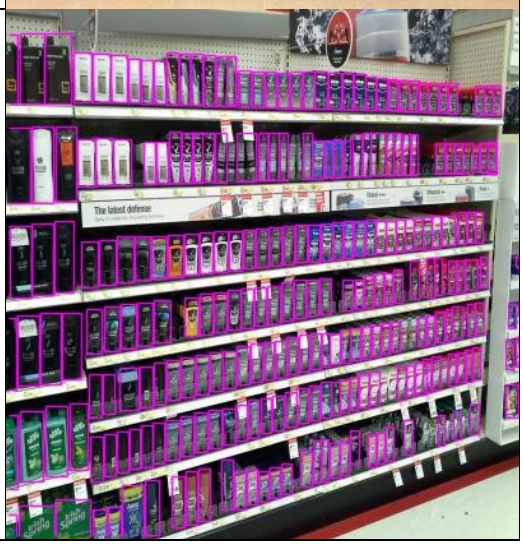
Eğitimi tamamlanan tüm YOLOv5 modelleri içerisinde ortalama hassasiyet değeri en düşük olan YOLOv5s modeli ve ortalama hassasiyet değeri en yüksek olan YOLOv5x modeli SKU110K veri setinde test için ayrılmış olan 2347 raf görüntüsü üzerinde test edilmiş ve Tablo 4'te YOLOv5s sonuçları, Tablo 5'te YOLOv5x sonuçları gösterilmiştir.

Yapılan çalışmada aynı test veri seti ve aynı donanımlar kullanıldığı göz önünde bulundurulduğunda YOLOv5s modeli tüm test veri seti için toplam 65sn, YOLOv5x modeli toplam 84sn işlem süresinde nesne belirleme işlemini gerçekleştirebilmiştir.

Tablo 4. YOLOv5s Test Sonuçları

Orijinal Test Görüntüsü	YOLOv5s tarafından çizilen sınırlayıcı kutular
	
	

Tablo 5. YOLOv5x Test Sonuçları

Orijinal Test Görüntüsü	YOLOv5x tarafından çizilen sınırlayıcı kutular
	
	

Tablo 4 ve Tablo 5‘deki raf görüntüleri incelendiğinde;

1. YOLOv5s modeli (a) ile gösterilen 416x416 boyutundaki raf görüntüsünde 0.008sn’lik işlem süresinde 176 nesneyi işaretlemiş, YOLOv5x modeli ise aynı raf görüntüsü için 0.015sn’lik işlem süresinde 173 nesneyi işaretlemiştir.

2. YOLOv5s modeli (b) ile gösterilen 416x416 boyutundaki raf görüntüsünde 0.008sn’lik işlem süresinde 242 nesneyi işaretlemiş, YOLOv5x modeli ise aynı raf görüntüsü için 0.015sn’lik işlem süresinde 247 nesneyi işaretlemiştir.

3. YOLOv5s modeli (c) ile gösterilen 416x416 boyutundaki raf görüntüsünde 0.007sn’lik işlem süresinde 192 nesneyi işaretlemiş, YOLOv5x modeli ise aynı raf görüntüsü için 0.015sn’lik işlem süresinde 194 nesneyi işaretlemiştir.

Yapılan çalışma sonucunda YOLOv5 modelinin seçilen YOLOv5s versiyonu ve YOLOv5x versiyonu karşılaştırıldığında işlem süresi bakımından YOLOv5s daha hızlı olmasına rağmen YOLOv5x nispeten daha fazla nesneyi işaretleyebilmiştir.

4.3. Faster R-CNN Modelinin Uygulanması

Perakende ürün tespit uygulaması için seçilen ikinci model R-CNN ailesine ait olan Faster R-CNN modelidir. Faster R-CNN modelinin bu çalışmada seçilmesinin nedeni nesne tanıma konusunda yavaş olmasına rağmen yüksek doğrulukta sonuçlar vermesidir.

4.3.1. Çalışma Ortamı

Derin öğrenme kullanılarak perakende ürün tespit uygulaması için çalışmada seçilen ikinci model olan Faster R-CNN modelinde de daha önceden seçilmiş olan YOLOv5 modelindeki aynı çalışma ortamı Google Colab Pro hizmetinden faydalanılmıştır. Colab Pro çalışma ortamına yine PyTorch, TensorFlow, Keras ve OpenCV gibi kütüphaneleri önceden yükleyerek çalışma ortamına dâhil edilmiştir.

4.3.2. İlişkilerin Kurulması ve Kütüphanelerin Yüklenmesi

Çalışmada Faster R-CNN modeli kullanarak perakende ürün tespit uygulaması Colab Notebook Pro ortamında geliştirilirken Tablo 6'da belirtilen kütüphanelerin ve eg4000 tarafından sağlanan reponun Colab Notebook Pro'ya dâhil edilmesi gerekmektedir.

Tablo 6. Faster R-CNN için Gerekli Kütüphane ve Repo

Kütüphane adı	Repo
Keras==2.2.5	https://github.com/eg4000/SKU110K_CVPR19
keras-retinanet==0.5.1	
tensorflow-gpu==1.15.4	
numpy==1.19.2	
opencv-python==3.1.0.5	
tqdm==4.50.2	
pandas==0.23.4	
Deeplabcut==2.1.6.3	

Çalışmada kullanılacak orijinal repo ve kütüphaneler Şekil 55'teki komutlar kullanılarak Colab Notebook Pro'ya dâhil edilmiştir.

```
!git clone https://github.com/eg4000/SKU110K_CVPR19
!pip install keras==2.2.4
!python object_detector_retinanet/setup.py install
!pip install keras_resnet
!pip install tensorflow
!pip install "tensorflow_gpu==1.14.0"
!pip install "tensorflow==1.14.0"
!pip install opencv-python==3.4.9.31
!pip install opencv-python-headless==3.4.9.31
!pip install deeplabcut==2.1.6.3
!pip install pandas==0.23.4
```

Şekil 54. Faster R-CNN için Gerekli Kütüphane ve Reponun Dahil Edilmesi

4.3.3. Veri Setinin Yüklenmesi

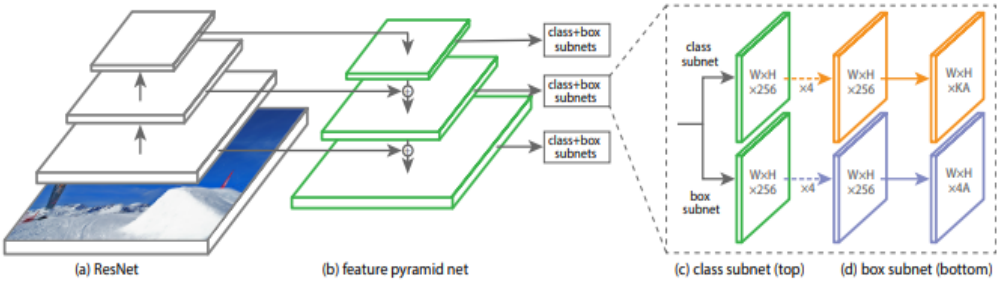
Çalışmanın bu bölümünde Roboflow platformu kullanılarak daha önceden hazırlanmış olan veri seti (Train, Test, Val) ve bu veri setindeki görüntülere ait açıklamaların bulunduğu *.csv* uzantılı dosyaların notebookun çalışma zamanına dahil edilmesi gerekmektedir. Bu işlem için Şekil 56'daki komutlar kullanılmıştır.

```
[ ] %cd /content/tensorflow-object-detection-faster-rcnn/data

    /content/tensorflow-object-detection-faster-rcnn/data

[ ] # UPDATE THIS LINK - get our data from Roboflow
    !curl -L https://app.roboflow.com/ds/5VN2uhm0yj?key=mvm8BOuNLA > roboflow.zip; unzip r

      % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                     Dload  Upload   Total   Spent    Left  Speed
    100   891  100   891    0     0   1354      0 --:--:-- --:--:-- --:--:--  1352
    100  550M  100  550M    0     0  96.4M      0  0:00:05  0:00:05 --:--:--  118M
```

Şekil 55. Veri Setinin Çalışma Ortamına Yüklenmesi

Çalışmada kullanılan veri setindeki görüntülere ait açıklama bilgileri .csv uzantılı bir dosyada tutulmaktadır. Bu açıklama dosyasında her görüntüdeki her bir nesne için sınırlayıcı kutu verileri $[class, xcenter, ycenter, width, height]$ formatında saklanmaktadır.

Perakende ürün tespit uygulaması için seçilen model Faster R-CNN Keras RetinaNET mimarisini kullanmıştır. Şekil 57'de Keras RetinaNET modelinin mimari yapısı görülmektedir.



Şekil 56. Keras RetinaNET Mimari Yapısı

Şekil 3. Tek aşamalı RetinaNet ağ mimarisi, zengin, çok ölçekli bir evrişimli özellik piramidi (b) oluşturmak için ileri beslemeli ResNet mimarisinin (a) üstünde bir Özellik Piramit Ağı (FPN) omurgası kullanmaktadır. RetinaNet modeli bu omurgaya

bağlantı kutularını (c) sınıflandırmak ve bağlantı kutularından kesin gerçek nesne kutularına (d) gerilemek için iki alt ağ eklemektedir. (Lin vd, 2017)

Yapılmış olan bu çalışmada yeni modelin eğitiminde 2019 yılında Goldman ve arkadaşları tarafından yapılmış olan *Precise Detection in Densely Packed Scenes* isimli çalışmadaki model temel olarak kullanılmıştır.

4.3.4. Modelin Eğitimi

Hazırlamış olduğumuz Perakende Ürün Tespit uygulamasının bu bölümünde Faster R-CNN modelinin ürün tespit performansı araştırılmaya çalışılmaktadır. Modelin eğitimi için YOLOv5 modelinde kullanılan aynı veri seti ve aynı donanım özellikleri kullanılmıştır. Modelin eğitimi için Şekil 58'deki komutlar kullanılmıştır.

```
[ ]  !python -u object_detector_retinanet/keras_retinanet/bin/train_iou.py --gpu 0
     --weights "/content/drive/MyDrive/SKU110K/snapshot/Fri_Apr__9_13_51_27_2021/resnet50_csv_50.h5"
     csv | tee train_iou_sku110k.log
```

Şekil 57. Faster R-CNN modelinin eğitimi

Komut penceresindeki weights parametresiyle Keras tarafından ücretsiz olarak dağıtılan resnet50_csv_50.h5 dosyası başlangıç ağırlıklarını oluşturmak için temin edilmiş ve kendi modelimizin eğitiminde kullanılmıştır. Eğitim sonucunda elde edilen ağırlıklar ve eğitim süreleri Tablo 7'de gösterilmiştir.

Tablo 7. Faster R-CNN Eğitim Sonucunda Elde Edilen Ağırlıklar ve Eğitim Süresi

Algoritma	Eğitilmiş Model Boyutu	Eğitim Süresi
Faster R-CNN / Keras RetinaNet	148.7MB	34s 26d 17sn

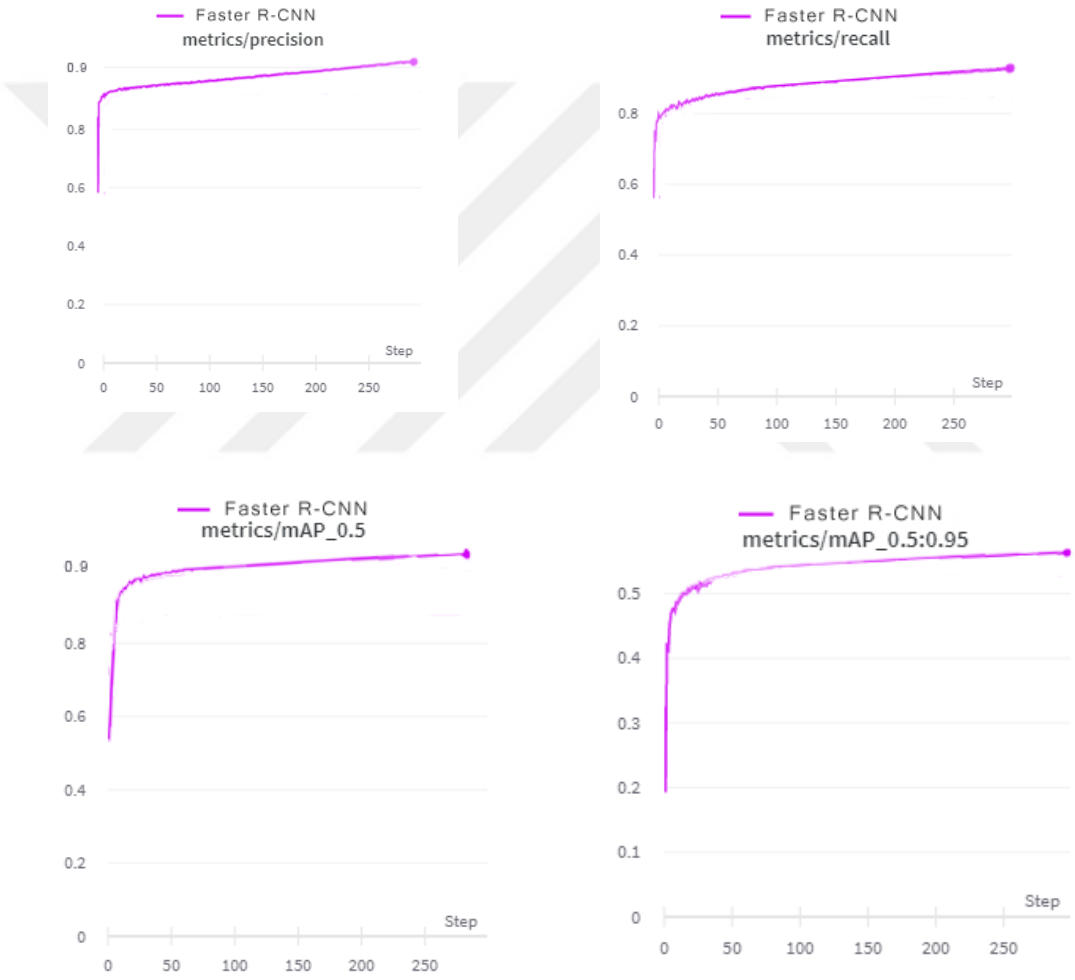
Çalışmada seçilen ikinci model olan Faster R-CNN eğitimi için 300 tur sayısı için (300x1000) 34 saat 26 dakika 17 saniye ve eğitilmiş modelin boyutu da 148.7MB olarak tespit edilmiştir.

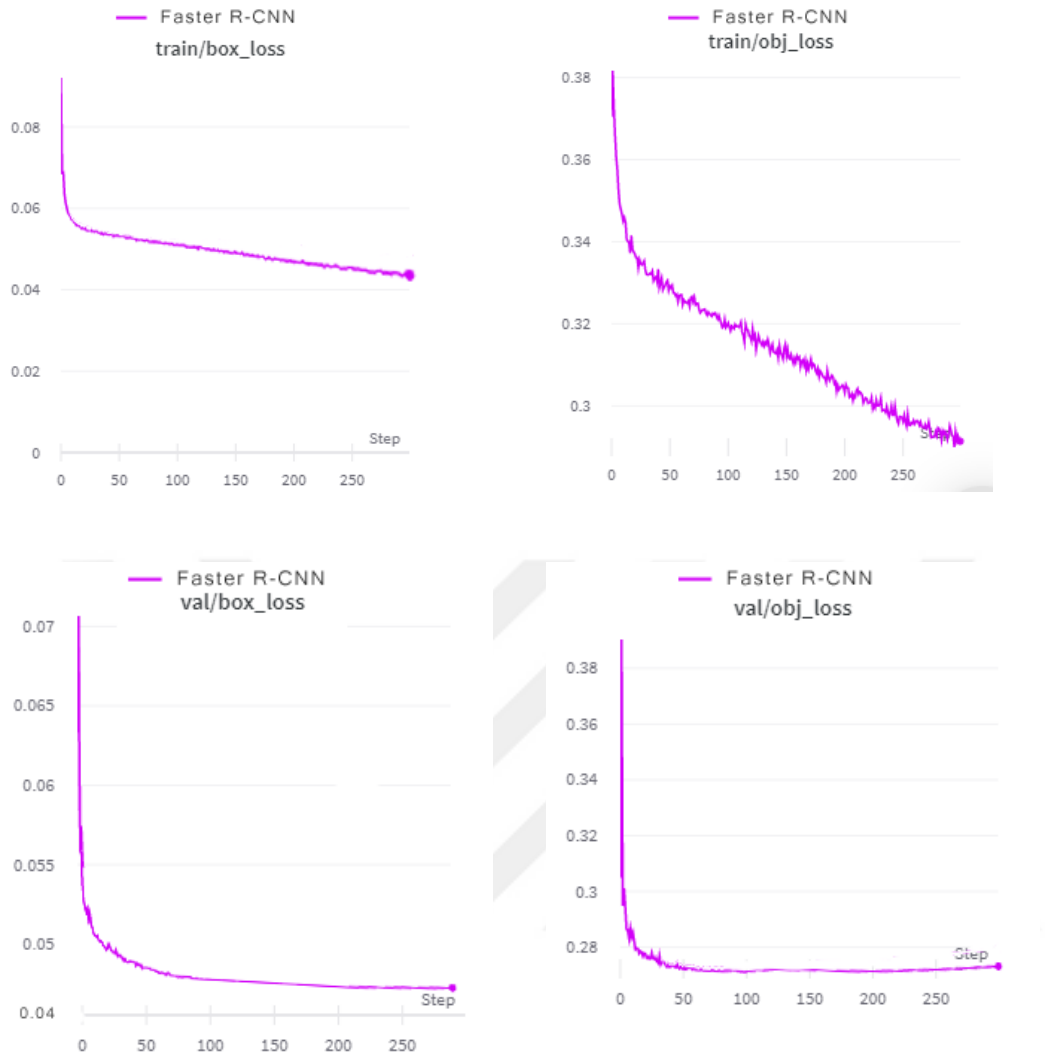
Tablo 8. Faster R-CNN Eğitim Sonuçları

Model	P	R	mAP 0.5	mAP 0.5:0.95	Train Box loss	Train Obj loss	Val Box loss	Val Obj loss
Faster RCNN-r50-fpn	0.902	0.855	0.892	0.550	0.042	0.276	0.044	0.276

Perakende ürün tespit uygulaması için SKU110k veri seti üzerinde seçilmiş Faster R-CNN modeli için yapılan model eğitim sonuçları Tablo 8'de verilmiştir. Tablo 8'de, **P:** Precission (Hassasiyet), **R:** Recall (Geri Çağırma), .**mAP:** Ortalama doğruluk oranı olarak belirtilmiştir.

Tablo 8 incelendiğinde 0.5'lik sınırlayıcı kutu eşiği için, Faster R-CNN modeli 8232 girdi görüntüsü ve 300 tur sayısı için %89,2 doğruluk oranıyla görüntü üzerindeki nesneleri tespit edebildiği görülmektedir. Eğitim sonucunda elde edilen değerler Şekil 59'da gösterilmiştir.





Şekil 59. Faster R-CNN Eğitim Sonucunun Grafiksel Gösterimi

Eğitim sonucunda elde edilen grafik incelendiğinde Faster R-CNN modelinin 0.5'lik sınırlayıcı kutu eşiğinde geri çağırma değeri (R) 0.855 ve otalama doğruluk oranı değeri (mAP0.5) 0.892 olduğu görülmüştür.

4.3.5. Seçilen Modelin Test Edilmesi

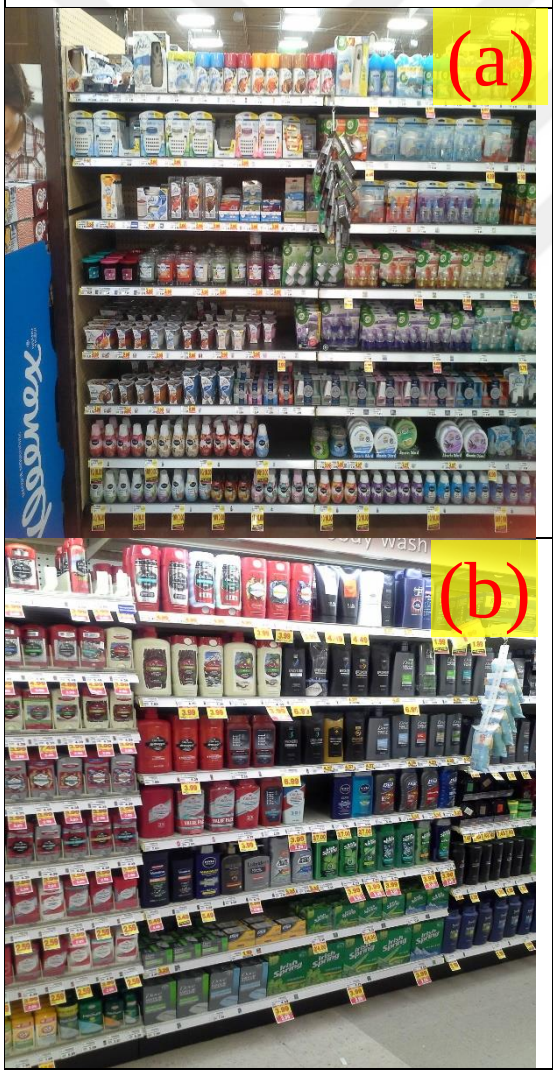
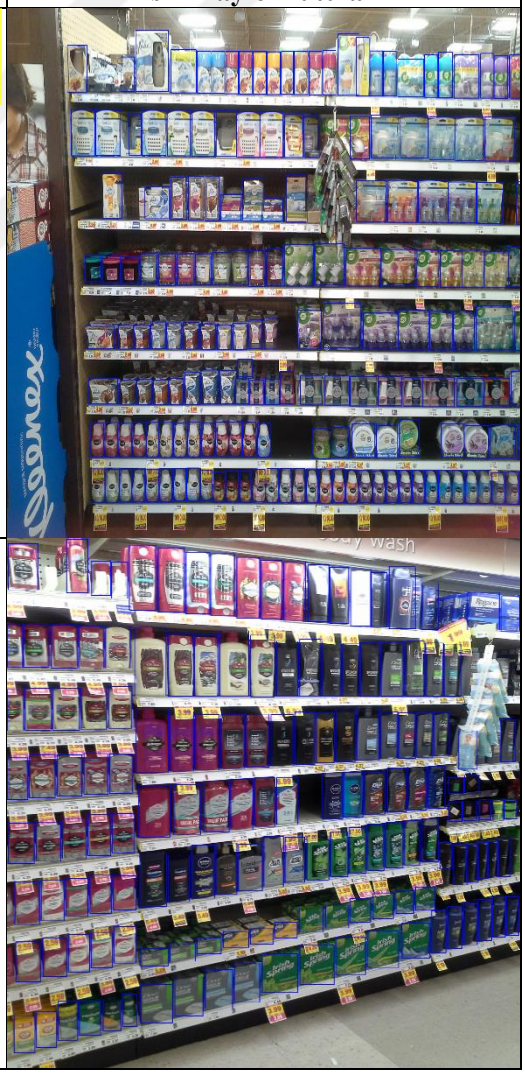
Yapılan Perakende ürün tespit uygulaması için seçilen Faster R-CNN modeli YOLOv5 modeli ile aynı donanımsal özelliklere sahip Colab Notebook Pro ortamında ve aynı test veri üzerinde Şekil 60'daki komutlar yardımıyla test edilmiştir.

```
!python -u object_detector_retinanet/keras_retinanet/bin/predict.py --gpu 3 csv
"/content/gdrive/MyDrive/SKU110K/snapshot/Fri_Apr__9_13_51_27_2021/ozel_resnet50_csv_27.h5"
--hard_score_rate=0.5 | tee predict_sku110k.log
```

Şekil 58. Faster R-CNN Modelinin Test Edilmesi

Eğitimi tamamlanan Faster R-CNN modeli SKU110K data setinde test için ayrılmış olan 2347 raf görüntüsü üzerinde test edilmiş ve Tablo 9'da sonuçları gösterilmiştir. Yapılan çalışmada aynı test veri seti ve aynı donanımlar kullanıldığı göz önünde bulundurulduğunda Faster R-CNN modeli tüm test seti için 900sn işlem süresinde nesne belirleme işlemini gerçekleştirebilmiştir.

Tablo 9. Faster R-CNN modelinin test sonuçları

Orijinal Test Görüntüsü	Faster R-CNN tarafından çizilen sınırlayıcı kutular
	

Tablo 9 ‘daki raf görüntüleri incelendiğinde;

1. Faster R-CNN modeli (a) ile gösterilen 416x416 boyutundaki raf görüntüsünde 0.3sn’lik işlem süresinde 160 nesneyi işaretlemiştir.

2. Faster R-CNN modeli (b) ile gösterilen 416x416 boyutundaki raf görüntüsünde 0.33sn’lik işlem süresinde 187 nesneyi işaretlemiştir.

3. Faster R-CNN modeli (c) ile gösterilen 416x416 boyutundaki raf görüntüsünde 0.37sn’lik işlem süresinde 198 nesneyi işaretlemiştir.

SONUÇ

Bilgisayarlı görü konusunun en önemli problemlerinden biri olan nesne tespiti derin öğrenme tabanlı modellerle oldukça başarılı sonuçlarla çözüme kavuşturulabilmiştir. Yapılan deneysel çalışmalar sonucunda derin öğrenme tabanlı nesne tespit uygulamalarında kullanılacak veri setinin boyutu ve çeşitliliği, model eğitiminin tur sayısı ve öğrenme oranı gibi hiper parametreler başarımın arttırılmasında oldukça büyük önem arz etmektedir.

Yapılmış olan bu tez çalışmasında günlük yaşamda sürekli karşılaştığımız market raflarına ait görüntülerdeki ürünleri tespit edebilmek için farklı derin öğrenme mimarileri araştırılmış, analiz edilmiş ve araştırmacılar için yayınlanmış olan SKU110K veri kümesi ön işlemden geçirilerek seçilmiş modeller üzerinde eğitimleri gerçekleştirilmiştir. Çalışmada nesne tanıma konusunda sıklıkla kullanılan iki derin öğrenme modeli (Faster R-CNN ResNet50 ve YOLOv5) seçilmiş ve seçilen mimariler için literatürde belirlenmiş hiper parametrelere bağlı kalınarak model eğitimleri transfer öğrenme yöntemiyle gerçekleştirilmiş, modelin eğitim ve test sonuçları kaydedilmiştir. Yapılmış olan tez çalışmasının katkıları ve sonuçları şu şekilde ifade edilebilir;

Perakende ürün tespit uygulaması için seçilen ilk model YOLO ailesinin en yeni üyesi YOLOv5 olarak belirlenmiştir. YOLOv5 modeli eğitim süresi, eğitilmiş model ağırlıklarını içeren dosya boyutu, modelin test süresi ve modelin çalışma süresi gibi ayırt edici unsurlar bakımında S (Small), M (Medium), L (Large) ve X(Extended) olmak üzere 4 alt sürümle ifade edilmektedir. Bu alt sürümlerden;

- YOLOv5s modeli 64 parti boyutu ve 300 tur sayısı için 6 saat 51 dakika 37 saniye eğitim süresi, 14.02 MB eğitilmiş model boyutu,%85,5 doğruluk oranı, 0.053 loss değeri ve 2347 raf görüntüsü için toplam 65.34 saniyelik test süresine sahip olduğu,
- YOLOv5m modeli 48 parti boyutu ve 300 tur sayısı için 11 saat 9 dakika 51 saniye eğitim süresi, 41.45 MB eğitilmiş model boyutu,%86,7 doğruluk oranı, 0.050 loss değeri ve 2347 raf görüntüsü için toplam 73.06 saniyelik test süresine sahip olduğu,

- YOLOv5l modeli 32 parti boyutu ve 300 tur sayısı için 17 saat 30 dakika 15 saniye eğitim süresi, 91.5 MB eğitilmiş model boyutu,%87,2 doğruluk oranı, 0.048 loss değeri ve 2347 raf görüntüsü için toplam 77.98 saniyelik test süresine sahip olduğu,
- YOLOv5x modeli 16 parti boyutu ve 300 tur sayısı için 23 saat 36 dakika 15 saniye eğitim süresi, 170.94 MB eğitilmiş model boyutu,%87,5 doğruluk oranı, 0.047 loss değeri ve 2347 raf görüntüsü için toplam 83.95 saniyelik test süresine sahip olduğu görülmüştür.

Tez çalışmasında seçilen YOLOv5 modellerinde aynı donanım, veri kümesi ve tur sayısıyla yapılan değerlendirme sonucunda, **eğitim ve test süresi bakımından** en hızlı çalışan model YOLOv5s, **en yüksek doğruluk oranına** sahip model ise YOLOv5x olduğu görülmüştür.

Eğitim sonucundan Seçilen YOLOv5 modellerinde oluşan loss fonksiyonu grafikleri incelendiğinde kayıp değeri 0'a olabildiğince yaklaşmış ve eğitimin başarısı test sonuçlarına da yansımıştır. Seçilmiş olan YOLOv5 modellerinden YOLOv5s modelinin loss değeri 0.053, YOLOv5m modelinin loss değeri 0.050, YOLOv5l modelinin loss değeri 0.048 ve YOLOv5x modelinin loss değeri 0.047 olarak gözlemlenmiş ve sonuçlar eğitimin başarısı olarak yansımıştır. Sonuç olarak YOLOv5x modelinin en yüksek doğruluk oranına sahip olduğu söylenebilir.

Perakende ürün tespit uygulaması için seçilen ikinci model R-CNN ailesinin en hızlı üyesi Faster R-CNN olarak belirlenmiştir. Faster R-CNN modeli 300 tur sayısı için (300x1000) 34 saat 26 dakika 17 saniye eğitim süresi, 148.7MB eğitilmiş model boyutu, %89,2 doğruluk oranı,0.042 loss değeri ve 2347 raf görüntüsü için toplam 900 saniyelik test süresine sahip olduğu görülmüştür.

Yapılmış olan Perakende Ürün Tespiti tez çalışması sonucunda iki ayrı derin öğrenme tabanlı nesne tespit modeli (YOLOv5-s/m/l/x ve Faster R-CNN) aynı donanım, veri kümesi ve tur sayısıyla işlenmiş sonuçlar yukarıda belirtilmiştir. Bu sonuçlar bakımından **eğitim süresi ve test süresi bakımından** YOLOv5 modelinin tüm sürümlerinin Faster R-CNN'den başarılı olduğu görülmüştür. Bununla beraber Faster R-CNN **doğruluk oranı bakımından** YOLOv5 modellerinin hepsinden daha başarılı sonuç vermiştir.

Tez çalışması sırasında seçilmiş olan her iki modelinde test başarısının düştüğü raf görüntüleri incelendiğinde bu görüntülerdeki düşük çözünürlük ve dikey olmayan görüntü çekim açılarının olduğu gözlemlenmiştir.

Yapılmış olan Perakende Ürün Tespiti isimli tez çalışması ile derin öğrenme tabanlı teknikler kullanılarak, insanlar tarafından yoğun şekilde ve farklı ürün gruplarıyla yerleştirilmiş market raflarındaki ürünlerin tespitinde başarılı sonuçlar verdiği görülmüştür. Yapılmış olan bu tez çalışması temelde müşterilerin alışveriş deneyimlerine katkı sağlamak için yapılmış olsa da bir diğer kazanımı da market yönetimlerindeki otomatik sistemlerin geliştirilmesidir. Geliştirilen bu sistemler sayesinde yöneticiler, raf yerleşimleri, ürün stok takipleri veya müşteri cari işlemleri için hem daha az personel hem de daha az zaman harcayacaklardır. Buda diğer yönetim işlemlerindeki performans ve kalitenin artmasına sebep olacaktır.

Yapılmış olan bu çalışmada karşılaşılan güçlüklerin en başında elde edilen görüntülerdeki düşük çözünürlük, görüntülerin çekim açılarındaki hatalar, her bir ürün sınıfı için yeterli görüntünün olmaması ve ürünlerin raf yerleşiminde düzensiz yerleşim veya üst üste gelen ürünler olduğu görülmüştür. Bundan sonra yapılması düşünülen ürün tespit çalışmalarında bu güçlüğün ortadan kaldırılmasının ürün tespit sisteminin başarımını arttıracağı düşünülmektedir. Yapılması düşünülen sonraki çalışmalarda elde edilen ESA'ya ait ağırlıklar farklı platformlara uyarlanarak mobil cihazlarla da kullanılabilir olması amaçlanmaktadır. Bu sayede mobil cihazlardaki görüntü birimine ek olarak ses birimi de kullanılarak özellikle görme engelli bireylerin alışveriş deneyimlerinin daha verimli olması amaçlanmaktadır.

KAYNAKÇA

Adam Coates, Honglak Lee and Andrew Niki (2011), "An Analysis of Single-Layer Networks in Unsupervised Feature Learning", *in PMLR, pp.* 215–223.

Afshine Amidi and Shervine Amidi (2019). *Derin Öğrenme El Kitabı,* (Çeviri; Ekrem, Ç. ve Omer, B.,) (Stanford CS 229)- Cheatsheet-Deep-Learning.

Afshine Amidi and Shervine Amidi (2019). *Evrişimli Sinir Ağları El Kitabı,* https://stanford.edu/~shervine/l/tr/teaching/cs-230/cheatsheet-convolutional-neuralnetworks. Erişim Tarihi:19/05/2020-16:00.

Afshine Amidi and Shervine Amidi (2019). *Yapay Zekâ El Kitabı* (Çeviri; Ayyüce, K., Başak, B., Yavuz, K., Cemal, G.,) (Stanford CS 221)-Super-Cheatsheet-Artificial-İntelligence.

Aktan, Ertuğrul (2018). "Büyük Veri: Uygulama Alanları, Analitiği ve Güvenlik Boyutu". *Bilgi Yönetimi Dergisi,* Cilt: 1 Sayı: 1, s. 1-22.

Alhajalabdulla, Ahmad (2020), *Koç, Boğa ve Aygır Sperm Hücrelerinin Takibi İçin Yeni Bir Algorıtmanın Modellenmesi.* Atatürk Üniversitesi, Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi, Erzurum.

Angelova Anelia, Krizhevsky Alex and Vanhoucke Vincent (2015), "Pedestrian Detection With a Large-Field-Of-View Deep Network". *Proceedings - IEEE International Conference on Robotics and Automation,* June(June), 704–711.

Aslan, Muzaffer (2022), "Derin Öğrenme Tabanlı Otomatik Beyin Tümör Tespiti", *Fırat Üniversitesi Müh. Bil. Dergisi Araştırma Makalesi* 34(1), 399-407.

Başarır, Bilen (2019), Deep Learning Based Object Detection-Derin Öğrenme Tabanlı Nesne Takibi, Yayınlanmamış yüksek lisans tezi. Bursa Uludağ Üniversitesi Fen Bilimleri Enstitüsü, Bursa.

Berk Güney and Sertel Elif (2019). *Semantıc Land Cover And Land Use Classıficatıon Usıng Deep Convolutıonal Neural Networks-Derin Evrişimsel Sinir Ağları ile Arazi Kullanımı ve Arazi Örtüsünün Anlamsal Sınıflandırılması,* İstanbul Teknik Üniversitesi, Yüksek Lisans Tezi, İstanbul.

BVLC (2016). “Caffe” http://caffe.berkeleyvision.org/, Erişim Tarihi: 14.05.2020-22:00.

Byrd Richard, Chin Gillian, Nocedal Jorge and Wu Yuchen (2012), “Sample Size Selection in Optimization Methods for Machine Learning”, *Mathematical Programming* 134.1, 127–155.

Chen, Wenjing (2017), “Deep Learning for Object Detection”, Erişim Adresi: https://www.slideshare.net/WenjingChen7/deep-learning-for-object-detection, Erişim Tarihi : 23/05/2020-01:06.

Chong Timothy, Bustan Idawati and Wee Mervyn (2016). “Deep Learning Approach to Planogram Compliance in Retail Stores, Stanford University”, *Stanford,* CA, USA.

Ciaburro Giuseppe and Venkateswaran Balaji (2017), *Neural network with R.* In Packt (Vol. 91). Packt Publishing, ISBN: 9781788397872.

Dai Jifeng, Li Yi, He Kaiming and Sun Jian (2016). *R-Fcn: Object Detection via Region-Based Fullyconvolutional Networks*, Advances in Neural İnformation Processing Systems, 379-387.

Daş Resul, Polat Berna and Tuna Gürkan (2019). “Derin Öğrenme ile Resim ve Videolarda Nesnelerin Tanınması ve Takibi”. *Fırat Üniversitesi Mühendislik Bilimleri Dergisi*, 31(2), 571–581.

Demirci Orel Fatma ve Kara Ali (2014). "Supermarket Self-Checkout Service Quality, Customer Satisfaction, and Loyalty: Empirical Evidence From an Emerging Market," *Journal of Retailing and Consumer Services, Elsevier,* vol. 21(2), pages 118-129.

Deng Li and Yu Dong (2014), “Deep Learning: Methods and Applications”. *Foundations and Trends in Signal Processing,* 7(3–4), 197–387.

Dumitru Erhan, Szegedy Christian, Toshev Alexander and Dragomir Anguelov (2014). “Scalable Object Detection using Deep Neural Networks.” *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* (pp. 2147-2154). Columbus, Ohio, ABD: 24-27.

Everingham Mark, Eslami Ali, Gool Van Luc, Williams Christopher, Winn John and Zisserman Andrew (2012), "The pascal visual object classes challenge 2012 (voc2012) results", Erişim Adresi: http://www. pascal-network. org/ challenges/ VOC/voc2011/workshop/index. html, 2011, Erişim Tarihi: 22/06/2020-17:00.

Everingham Mark, Gool Van Luc, Williams Christopher, Winn John and Zisserman Andrew (2007), "The PASCAL Visual Object Classes Challenge", Results http://host.robots.ox.ac.uk/pascal/VOC/voc2007/, Erişim Tarihi: 23/05/2020-03:18.

Fernandez Cristin, Soria Emilo, Martin Jack and Serrano-López Antonio (2006), "Neural networks for animal scienceapplications: Two case studies", *Expert Systems With Applications.* 31, 444-450.

Fumikazu Morimura and Kenichi Nishioka (2016). "Waiting in exit-stage operations: expectation for self checkout systems and overall satisfaction", *Journal of Marketing Channels,* vol. 23, no. 4, pp. 241–254.

Geng Weidong, Han Feilin, Lin Jiangke, Zhu Liuyi, Bai Jieming, Wang Suzhen, He Lin, Xiao Qiang and Lai Zhangjiong (2018), *Fine-grained grocery product recognition by one-shot learning, in Proceedings of the 2018 ACM Multimedia Conference on Multimedia Conference,* Seoul, Republic of Korea.

George Marian and Floerkemeier Christian (2014). "Recognizing products: a per-exemplar multi label image classification approach," in *Proceedings of the 2014 European Conference on Computer Vision,* pp. 440–455, Zurich, Switzerland.

Girshick Ross, Donahue Jeff, Darrell Trevor and Malik Jitendra (2014). "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation", The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 580-587.

Girshick, Ross (2015). "Fast R-CNN, Proceedings of the IEEE international conference oncomputer Visio"n, *Submitted* on 30 Apr 2015 (v1), last revised 27.

Goldman Eran, Herzig Roei, Eisenschtat Aviv, Ratzon Oria, Levi Itsik, Goldberger, Jacob and Hassner Tal (2019). “Precise Detection İn Densely Packed Scenes”, Work done while at the University of Southern California.

Goodfellow Ian, Pouget-Abadie Jean, Mirza Mehdi, Xu Bing, Warde-Farley David, Ozair Sherjil, Courville Aaron, Bengio Yoshua, (2014). *Generative adversarial nets.* In Advances in neural information processing systems (pp. 2672-2680).

Grewala Dhruv, Roggeveena Anne and Nordfältb Jens (2017). “The Future of Retailing”, *Journal of Retailing,* vol. 93, no. 1, pp. 1–6.

Gu Xiaodong, Zhang Hongyu, Zhang Dongmei and Kim Sunghun (2016),” Deep API Learning”. *In Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering* (pp. 631-642). ACM.

Guo Yanming, Liu Yu, Oerlemans Ard, Lao Songyang, Wu Song and Lew Michael (2016), “Deep learning for visual understanding: A review”. *Neurocomputing* 187:27–48.

Hampshire (2019). “AI Spending By Retailers To Reach $12 Billion By 2023, Driven By The Promise Of İmproved Margins”, https://www. juniperresearch. com/ press/ press-releases/ai-spending-by-retailers-reach-12-billion-2023.

Hawkins, Douglas (2004). “The Problem of Overfitting”, *Journal of Chemical Information and Computer Sciences* 44.1, 1–12.

Haykin, Simon (2009), *Rosenblatt’s Perceptron.* Neural Networks and Learning Machines-3rd ed, 47–67, ISBN-13: 978-0-13-147139-9.

He Kaiming, Zhang Xiangyu, Ren Shaoqing and Sun Jian (2014). *Spatial pyramid pooling in deep convolutional networks for visual recognition.* In: European conference on computer vision. 13th European Conference, Zurich, Switzerland, September 6-12

He Kaiming, Zhang Xiangyu, Ren Shaoqing and Sun Jian (2015), “Deep Residual Learning for Image Recognition”, In IEEE Conference on Computer Vision and Pattern Recognition, pp. 770-778.

He Kaiming, Zhang Xiangyu, Ren Shaoqing and Sun Jian (2015), “Spatial pyramid pooling in deep convolutional networks for visual recognition”. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI).* pp. 346–361.

He Kaiming, Zhang Xiangyu, Ren Shaoqing and Sun Jian (2017) “Mask r-cnn”. In: Computer Vision (ICCV), IEEE International Conference on. IEEE. 2017, pp. 2980–2988.

Hinton Geoffrey and Ruslan Salakhutdinov (2006), “Reducing the Dimensionality of Data with Neural. Networks.” *Science 313,* no. 5786, 504–507.

Hinton Geoffrey and Ruslan Salakhutdinov (2009). “Replicated Softmax: an Undirected Topic Model”, *in Advances in Neural Information Processing* Systems 22, pp. 1607–1614.

Hinton Geoffrey, Alexander Krizhevsky, Ilya Sutskever and Nitish Srivastava (2016), *System and Method for Addressing Overfitting in A Neural Network.* US Patent 9,406,017.

Hinton, Geoffrey (2012). “A Practical Guide To Training Restricted Boltzmann Machines”. Lecture Notes in Computer Science *(Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics),* 7700 LECTU, 599–619.

Hochreiter Sepp and Schmidhuber Jürgen (1997), “Long Short-Term Memory”. *Neural Computation,* 9(8), 1735–1780.

Hochreiter Sepp, Heusel Martin, Obermayer Klaus and Notes Author (2007), “Fast Model-based Protein Homology Detection Without Alignment”, *Bioinformatics,* vol. 23, no. 14, pp. 1728– 1736.

https://tinyurl.com/y2ljr34t, Erişim Tarihi: 18.05.2020- 00:10.

Hugo Larochelle and Bengio Yoshua (2008). “Classification Using Discriminative Restricted Boltzmann Machines”, *in Proceedings of the 25th international conference on Machine learning- ICML,* C.P. 6128, Montreal, Qc, H3C 3J7, Canada.

Hui, Jonathan (2018), “Understanding Feature Pyramid Networks for object detection (FPN)”, Erişim Adres: https://jonathan-hui.medium.com/understanding-feature-pyramid-networks-for-object-detection-fpn-45b227b9106c,Erişim Tarihi: 02/05/2021-02: 00.

Ioffe Sergey and Szegedy Christian (2015), “Batch normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, ICML'15”, *Proceedings of the 32nd International Conference on International Conference on Machine Learning-Volume* 37 July 2015 Pages 448-456.

Ivakhnenko Grigor′evich Alekseĭ and Lapa Grigorévich Valentin (1965). *Cybernetic Predicting Devices.* N.Y. CCM Information Corp.

Jocher, Glenn (2020). “YOLOv5. (GitHub)”, https://github.com/ultralytics/yolov5 Erişim Tarihi:02/05/2021-00:30.

Karlinsky Leonid, Shtok Joseph, Tzur Yochay and Tzadok Asaf (2017). *Fine-grained Recognition of Thousands of Object Categories With Singleexample Training,* in Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition, pp. 4113–4122, Honolulu, HI, USA.

Keras (2020). “Deep Learning Library for Theano and Tensor Flow”, Erişim Adresi: https://keras.io/, Erişim Tarihi: 14.05.2020.

Kızrak, Ayyüce (2018). “Şu Kara Kutuyu Açalım: Yapay Sinir Ağları”, https://ayyucekizrak.medium.com/şu-kara-kutuyu-açalim-yapay-sinir-ağları, Erişim tarihi: 21/05/2020-04:13.

Kızrak, Ayyüce (2019), “Comparison of Activation Functions for Deep Neural Networks”, Erişim Adresi:https://towardsdatascience.com/comparison-of-activation-functions-for-deep-neural-networks-706ac4284c8a, Erişim tarihi: 22/05/2020-01:00.

Kızrak, Ayyüce (2020), “Derine Daha Derine: Evrişimli Sinir Ağları”, https://medium.com/@ayyucekizrak/derine-daha-derine-evrişimli-sinir-ağları, Erişim tarihi: 21/05/2020-01:13.

Kozlov, Artem (2020). “2nd Place Solutıon To Product Detectıon In Densely Packed Scenes Technıcal Report, Cvpr”, https://trax-geometry.s3.amazonaws .com/ cvpr_challenge/detection_challenge_technical_reports/2nd_Working_with_scale__2nd_place_solution_to_Product_Detection_in_Densely_Packed_Scenes.pdf, Erişim Tarihi:19/05/2021-15:40.

Krizhevsky Alex, Sutskever Ilya and Hinton Geoffrey (2012), *ImageNet Classification with Deep Convolutional Neural Networks,* NIPS'12 Proceedings of the 25th International Conference on Neural Information Processing Systems, cilt 1, pp. 1097-1105.

Kunihiko, Fukushima (1988), “Neocognitron: A hierarchical neural network capable of visual pattern recognition”. *Neural Networks,* 1(2), 119–130.

Kurzweil, Raymond (1992), *The age of intelligent machines,* Cambridge, MA: MIT Press, ISBN: 9780262610797.

LeCun Yann, Bottou Leon, Bengio Yoshua and Haffner Patrick (1989), “Handwritten Digit Recognition: Applications of Neural Network Chips and Automatic learning”, *IEEE Commun. Mag,* cilt 27, no. 11, pp. 41-46,

Lecun Yann, Ieee Member, Bottou Leon, Bengıo Yoshua and Haffner Patrıck (1998), “Gradient-based Learning Applied Todocument Recognition”, *Proceeding of the IEEE,* cilt 86, no. 11, pp. 2278-2324.

Lee Honglak, Grosse Roger, Ranganath Rajesh and Andrew Ng (2009). “Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations”, *In Proceedings of the 26th Annual International Conference on Machine Learning,* pp. 609–616.

Li Peixia, Wang Dong, Wang Lijun and Lu Huchuan (2018). “Deep visual tracking: Review and Experimental Comparison”. *Pattern Recognition,* 76, 323–338.

Lin Tsung-Yi, Dollar Piotr, Girshick Ross, He Kaiming, Hariharan Bharath and Belongie Serge (2017), “Feature Pyramid Networks for Object Detection”. *In Published 9 December.*

Liu Lei, Pan Zongxu and Lei Bin (2017), "Learning a Rotation Invariant Detector with Rotatable Bounding Box". http://arxiv.org/abs/1711.09405, Erişim Tarihi: 23/04/2020.

Liu Wei, Anguelov Dragomir, Erhan Dumitru and Szegedy Christian (2016), "Ssd: Single Shot Multibox Detector", European conference on computer vision, Springer, 21–37.

López-de-Ipiña Diego, Lorido Tania and López Unai (2011), "Indoor navigation and product recognition for blind people assisted shopping, in Proceedings of the 2011", *International Workshop on Ambient Assisted Living,* pp. 33–40, Torremolinos, Spain.

Mathieu Michael, Couprie Camille ve LeCun Yann (2015). "Deep Multi-scale Video Prediction Beyond Mean Square Error". arXiv preprint arXiv:1511.05440.

McCarthy John, Minsky Marvin, Rochester Nathaniel and Shannon Claude (2006), "A Proposal Forthe Dartmouth Summer Research Project On Artificial İntelligence", *AI Magazine Vol.*27 (4).

Mcculloch Warren and Pitts Walter (1943), "A Logical Calculus of the Idea Immanent in Nervous Activity". *Bulletin of Mothemnticnl Biology* Vol. 52, No. l/2. pp. 99-115. 1990.

Mesci, Yiğit (2019), YOLO "Algoritmasını Anlamak", Erişim Adresi: https://medium.com/deep-learning-turkiye/yolo-algoritmasını-anlamak-290 f215280 8f, Erişim Tarihi: 03/06/2020-14:41.

Mihir, Rajput (2020). "YOLO V5-Model Architecture and Technical Details Explanation", Erişim Adresi: https://pub.towardsai.net/yolo-v5-explained-and-demysti fied-4e4719891d69, Erişim Tarihi: 03/05/2021-19:00.

Nair Vinod and Hinton Geoffrey (2010), *Rectified linear units improve restricted boltzmannmachines,* Proceedings of the 27th international conference on machine learning (ICML-10), 807–814.

Najafabadi, Maryam, Villanustre Flavio, Khoshgoftaar Taghi, Seliya Naeem, Wald Randall and Muharemagic Edin (2015), "Deep learning applications and challenges in big data analytics". *Journal of Big Data,* 2, 1–21.

Narrative Science (2018). “Artificial Intelligence (AI) Adoption Grew Over 60% in the Last Year”, https://narrativescience.com/companyannoucements/artificial-intelligence-ai-adoption-grew-over-60-in-the-last-year/, Erişim Tarihi: 17/ 06/ 2020.

Nwankpa Chigozie, Ijomah Winifred, Gachagan Anthony and Marshall Stephen (2018), *Activation functions: comparison of Trends in Practice and Research for Deep Learning,* arXiv preprintarXiv:1811.03378.

Ouaknine, Arthur (2018). “Review of Deep Learning Algorithms for Object Detection”, Erişim Adresi: https://medium.com/zylapp/review-of-deep-learning-algorithms-for-object-detection-c1f3d437b852, Erişim Tarihi: 22/ 05/ 2020-01:40.

Pascanu Razvan, Mikolov Tomas and Bengio Yoshua (2012). “Understanding the Exploding Gradient Problem”, Erişim Adresi: https://arxiv.org/ /1211.5063v1. pdf, Erişim Tarihi: 17/07/2020-13:15.

Ponti, Moacir, Ribeiro Leonardo, Nazare Tiago, Bui Tu and Collomosse John (2017), “Everything You Wanted to Know about Deep Learning for Computer Vision but Were Afraid to Ask”, 2017 30th SIBGRAPI Conference on Graphics, *Patterns and Images Tutorials (SIBGRAPI-T)* 17–41.

Rana, Rajib (2016). “Gated Recurrent Unit (GRU) for Emotion Classification From NoisySpeech”, *arXiv preprint arXiv,* 1612.07778.

Redmon Joseph and Farhadi Ali (2017). “YOLO9000: Better, Faster, Stronger, İn Proceedings Of The 2017” ,IEEE Conference on Computer Vision and Pattern Recognition.

Redmon Joseph, Divvala Santosh, Girshick Ross and Farhadi Ali (2016). “You Only Look Once: Unified, real-time object detection”, *University of Washington, Allen Institute for AI, Facebook AI Research,* 779-788.

Reed Scott, Akata Zeynep, Yan Xinchen, Logeswaran Lajanugen, Schiele Bernt and Lee Honglak (2016). “Generative adversarial text to image synthesis”. arXiv preprint arXiv:1605.05396.

Ren Shaoqing, He Kaiming, Girshick Ross and Sun Jian (2015), “Faster R-CNN: Towards Real-time Object Detection With Region Proposal Networks”, *Advances in Neural Information Processing Systems,* 91–99.

Ren Shaoqing, He Kaiming, Girshick Ross and Sun Jian (2017), “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, *IEEE Transactıons on Pattern Analysıs and Machıne Intellıgence, cilt* 39, no. 6, pp. 1137-1149.

Riel van Allard, Semeijn Janjaap, and Ribbink Dina (2012). “Waiting for service at the checkout: negative emotional responses, store image and overall satisfaction,” *Journal of Service Management,* vol. 23, no. 2, pp. 144–169.

Rümeysa, Büşra (2019), *Image Classification With Deep Learning-Derin Öğrenme İle Görüntü Sınıflandırma,* Bursa Uludağ Üniversitesi, Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi, Bursa.

Salakhutdinov Ruslan, Mnih Andriy and Hinton Geoffrey (2007), “Restricted Boltzmann Machines for Collaborative Filtering”, in Proceedings of the 24th İnternational Conference on Machine Learning-ICML, pp. 791-798.

Shapiro, Marcos (2009), *Executing the Best Planogram,* Vol. 1, Professional Candy Buyer, Norwalk, CT, USA.

Solawetz, Jacob (2020). “Breaking Down YOLOv4 Roboflow”. Erişim Adresi: https://blog.roboflow.com/a-thorough-breakdown-of-yolov4/, Erişim Tarihi: 02/05/2021-01:30.

Süzen Ali Ahmet ve Kayaalp Kıyas (2018). *Derin Öğrenme ve Türkiye’deki Uygulamaları,* Iksad Internatıonal Publıshıng Houseısbn: 978-605-7510-53-2.

Szegedy Christian, Ioffe Sergey, Vanhoucke Vincent and Alemi Alex (2016), “Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning”, *Proceedings of the ThirtyFirst AAAI Conference on Artificial Intelligence,* pp. 4278-4284.

Szegedy Christian, Liu Wei, Jia Yangqing, Sermanet Pierre, Reed Scott, Anguelov Dragomir, Erhan Dumitru, Vanhoucke Vincent and Rabinovich Andrew (2015). *Going Deeper with Convolutions,* Computer Vision and Pattern Recognition.

Szegedy Christian, Toshev Alexander and Dumitru Erhan (2013). *Deep Neural Networks for object detection.* Advances in Neural Information Processing Systems.

Tonioni Alessio, Serra Eugenio and Stefano Di Luigi (2019). *A Deep Learning Pipeline For Product Recognition On Store Shelves,* 2018 IEEE International Conference on Image Processing, Applications and Systems (IPAS).

Vladimir, Vapnik (1999). "An Overview of Statistical Learning Theory", *IEEE Transactions on Neural Networks,* cilt 10, no. 5, pp. 988-999.

Weng Juyang, Cohen Paul and Herniou Marc (1992). "Camera Calibration With Distortion Models and Accuracy Evaluation"., *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Volume: 14, Issue: 10.

What is Torch?, Erişim Adresi: http://torch.ch/ Erişim Tarihi: 14.05.2020-16:00.

Wu Bing-Fei, Tseng Ju Wan, Chen Shin Yung, Yao Jhe Shih and Chang Ju Po (2016), "An İntelligent Self-Checkout System For Smart Retail", in 18 Computational Intelligence and Neuroscience Proceedings of the 2016 *International Conference on System Science and Engineering* (ICSSE), pp. 1–4, Puli, Taiwan,.

Yanci, Mustafa (2019), *Derin Öğrenme Yöntemleri İle Medikal Görüntülerde Kanserli Doku Tespiti,* İstanbul.

Yi Wei, Sun Yaoran, Ding Tao and He Sailing (2019), Detecting retail products in situ using CNN without human effort labeling - Sailing He Center for Optical and Electromagnetic Research.

Zeiler, Matthew and Fergus Rob (2013), "Visualizing and Understanding Convolutional Networks", European Conference on Computer Vision, pp. 818-833.

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı : İsmail KÖSE

Uyruğu : T.C.

EĞİTİM

Derece	Kurum	Mezuniyet Yılı

İŞ TECRÜBESİ

Tarih	Kurum	Görev