T.C.
YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ARAÇ PARKLARININ UYGUNLUK DURUMUNUN DERİN ÖĞRENME İLE TESPİT EDİLMESİ

Ayşe Betül BAKTIR

YÜKSEK LİSANS TEZİ
Elektronik ve Haberleşme Mühendisliği Anabilim Dalı
Haberleşme Programı

Danışman
Doç. Dr. Bülent BOLAT

Temmuz, 2020

T.C.
YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ARAÇ PARKLARININ UYGUNLUK DURUMUNUN DERİN ÖĞRENME İLE TESPİT EDİLMESİ

Ayşe Betül BAKTIR tarafından hazırlanan tez çalışması 22.07.2020 tarihinde aşağıdaki jüri tarafından Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü Elektronik ve Haberleşme Mühendisliği Anabilim Dalı Haberleşme Programı **YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Doç. Dr. Bülent BOLAT
Yıldız Teknik Üniversitesi
Danışman

Jüri Üyeleri

Doç. Dr. Bülent BOLAT, Danışman
Yıldız Teknik Üniversitesi _____

Doç. Dr. Umut Engin AYTEN, Üye
Yıldız Teknik Üniversitesi _____

Prof. Dr. Mehmet SAĞBAŞ, Üye
İstanbul Yeni Yüzyıl Üniversitesi _____

Danışmanım Doç. Dr. Bülent BOLAT sorumluluğunda tarafımca hazırlanan Araç Parklarının Uygunluk Durumunun Derin Öğrenme ile Tespit Edilmesi başlıklı çalışmada veri toplama ve veri kullanımında gerekli yasal izinleri aldığımı, diğer kaynaklardan aldığım bilgileri ana metin ve referanslarda eksiksiz gösterdiğimi, araştırma verilerine ve sonuçlarına ilişkin çarpıtma ve/veya sahtecilik yapmadığımı, çalışmam süresince bilimsel araştırma ve etik ilkelerine uygun davrandığımı beyan ederim. Beyanımın aksinin ispatı halinde her türlü yasal sonucu kabul ederim.

Ayşe Betül BAKTIR

İmza

TEŞEKKÜR

Öncelikle değerli danışman hocam Doç. Dr. Bülent BOLAT'a, lisans hayatımdan bugüne kadar akademik bilgisi ve tecrübesiyle bana yol gösterip yanımda olduğu ve ne zaman ihtiyacım olsa bana desteğini sunduğu için teşekkür ederim.

Beni yüksek lisans yapmam konusunda teşvik eden ve ilk gününden bugüne kadar desteğini benden esirgemeyen sevgili babama, beni her zaman her konuda dinleyen, manevi desteğini bana karşılıksız sunan anneme teşekkür ederim.

Bu çalışma boyunca benden desteklerini esirgemeyen, her aşamasında bana yeterli vakit ve imkan sağlayan sevgili eşim Turgut Gencay BAKTIR'a teşekkür ederim.

Ayşe Betül BAKTIR

İÇİNDEKİLER

SİMGE LİSTESİ

∇	Gradyan
Θ	Model Parametreleri
β	Moment tahmini için üstel bozulma oranı
ε	Öğrenme Katsayısı
$\mathbb{R}$	Reel Sayılar Kümesi

KISALTMA LİSTESİ

AI	Artificial Intelligence
CIFAR	Canadian Institute for Advanced Research
CNN	Convolutuonal Neural Network
DoG	Difference of Gaussian
GPS	Global Positioning System
GPU	Graphical Processor Unit
ILSVRC	ImageNet Large Scale Visual Recognition Competition
MNIST	Modified National Institue of Standarts and Technology
PSO	Particle Swarm Optimization
Relu	Rectifier Lineer Unit
RFID	Radio Frequency Identificaiton
RMS	Root Mean Square
SGD	Stochastic Gradient Descent
STN	Spatial Transform Network
SVM	Support Vector Machines
VGG	Visual Geometry Group
YSA	Yapay Sinir Ağları

ŞEKİL LİSTESİ

TABLO LİSTESİ

ÖZET

Araç Parklarının Uygunluk Durumunun Derin Öğrenme ile Tespit Edilmesi

Ayşe Betül BAKTIR

Elektronik ve Haberleşme Mühendisliği Anabilim Dalı
Yüksek Lisans Tezi

Danışman: Doç. Dr. Bülent BOLAT

Büyük şehirlerde yoğun trafik koşullarında bir aracın park edilmesi, uzun bir sürüş süresi ve beraberinde büyük zaman kayıplarına, trafik akışının bozulmasına ve dolayısıyla boş alan ararken çevre kirliliğine yol açar. Bir otoparka henüz girmeden önce içeride boş yer olup olmadığı bilgisine erişebilmek, bu tarz problemlerin önüne geçecektir. Kapalı otoparklarda bu sorun çeşitli sensör sistemleri ile çözülebilir, ancak bu çözümün açık otoparklara uygulanabilirliği konusunda gerek maliyet gerekse sürdürülebilirlik açısından problemler olmaktadır. Hali hazırda kameralarla donatılmış sokaklar ve caddeler, görüntü temelli çözümlere büyük bir kapı açmıştır. Bu çözümlerden son yıllarda yüksek doğruluklu sonuçlar vererek kendini kanıtlamış derin öğrenme metodu seçilmiş ve açık otoparklardaki park yerlerinin doluluk durumunu sınıflandıran bir derin öğrenme uygulaması geliştirilmiştir. Transfer öğreniminin ResNet modeli kullanılarak yapıldığı bu uygulamada yüksek doğruluk oranları elde edilmiştir.

Anahtar Kelimeler: Derin Öğrenme, park yeri, PKLot, CNRPark, ResNet

YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ABSTRACT

Determining the Occupancy of Vehicle Parking Areas by Deep Learning

Ayşe Betül BAKTIR

Department of Electronics and Communication Engineering
Master of Science Thesis

Advisor: Assoc. Prof. Dr. Bülent BOLAT

Parking a vehicle in heavy traffic conditions in large cities leads to a long driving time, as well as large time losses, traffic flow deterioration, and therefore environmental pollution when looking for free space. Having access to the information whether there is a free space before entering a parking lot will prevent such problems. In indoor parking lots, this problem can be solved with various sensor systems, but there are problems in terms of both cost and sustainability regarding the applicability of this solution to outdoor parking lots. Streets and parking areas already equipped with cameras have opened a huge door to image-based solutions. Among these solutions, the deep learning method, which has proven itself by giving high accuracy results in recent years has been chosen and a deep learning application has been developed that classifies the occupancy of parking spaces in outdoor car parks. In this application, where transfer learning is carried out using the ResNet model, high accuracy rates are obtained.

Keywords: Deep Learning, parking lot, PKLot, CNRPark, ResNet

YILDIZ TECHNICAL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

1
GİRİŞ

1.1 Literatür Özeti

Son yıllarda, otopark alanlarının doluluk oranını belirlemek için çalışmalar artmıştır. Paidi [1] ve arkadaşları bu çalışmaları karşılaştırmıştır. Park alanı doluluk tespiti konusunda infrared,ultrasonik ve RFID sensörler kullanılmaktadır. Ancak bu yöntemlerin açık alanlara uygulanması konusunda çevresel koşulların değişken olması sebebiyle uygulama zorluğu vardır. Açık alanlara uygulanabilir çözümler arasında ise mikrodalga radar ve GPS sayılabilir. Bu yöntemler ise uygulamada ve sürdürülebilirlik konusunda maliyetli olduğu için tercih edilmemektedir.

Genel olarak, büyük ölçekli açık otoparklarda donanımsal uygulamalar zordur. Buna rağmen kablosuz sensör ağlarını kullanarak Jihoon [2], bu sorun üzerine çalışmış ve Google Haritalar'ı kullanarak bir prototip geliştirmiş ve iyi sonuçlar elde etmiştir. Ancak, hem sokaklarda hem de açık otoparklarda zaten güvenlik kameraları bulunduğundan dolayı, bilgisayarlı görü ve görüntü işlemeyi kullanmak hem daha uygun maliyetli hem de daha kolay uygulanabilir bir yaklaşımdır. Bilgisayarlı görü yaklaşımıyla bir çözüme giden Ichihashi [3] parçacık sürü optimizasyonu (PSO) ile bulanık c-ortalama kümeleme(fuzzy c-means clustering) ve hiperparametre ayarı kullanarak bir sınıflandırıcı yapmıştır. Sistemleri, açık otoparklar için% 99,6'lık bir tespit oranına ulaşmıştır. Tschentscher, otoparkları sınıflandırmak için Destek Vektör Makineleri(SVM) ve DoG özelliklerini kullanmış ve park alanının kameraya olan mesafesine bağlı olarak % 92,33'ten % 99,96'ya ulaşan sonuçlar elde etmiştir [4]. Son yıllarda, Kabak ve Turgut, hava görüntüleri kullanarak boş park alanlarını tespit etmek için SVM tabanlı sınıflandırma yöntemini kullanmışlardır [5]. Raj da bu sorunu çözmek için görüntü işleme tekniğini kullanmıştır. Belirli bir slottaki görüntüden özellikleri çıkarmak için Canny Edge dedektörü ve LUV bazlı renk değişim algılayıcı kullanmıştır. Özelliklerin çıkarılmasından sonra, her bir slotu boş veya dolu olarak sınıflandırmak için bu özellikler Random Forest sınıflandırıcısına verilmiş ve % 98,31 oranında doğruluk elde edilmiştir [6].

Derin öğrenme hem açık alanlara uygulanabilirlik hem de sürdürülebilirlik ve maliyet açısından geliştirilebilecek bir çözümdür. Ayrıca, bu yöntemler geleneksel yöntemlerden daha yüksek doğrulukta sonuçlar verir. Evrişimli Sinir Ağları'nı kullanan Hoang, kendi sinir ağlarını geliştirmiştir. İlk olarak, yerel görüntü alanını araç boyutuna ve park yer değiştirmesine göre uyarlanabilir şekilde kırpmak için evrişimli bir uzamsal transformatör ağı (STN) entegre etmiştir. Bundan sonra nesneler arası oklüzyon problemlerini çözmek için 3 komşu alanı bir birim olarak gruplandırmıştır. Bu yöntemleri uygulayarak % 99,25 doğruluğa ulaşmıştır [7].

Evrişimsel Sinir Ağlarını kullanmanın başka bir yolu da önceden eğitilmiş ağları kullanmaktır. Bunun üzerinde çalışan Guiseppe, mLeNet ve mAlexNet kullanarak % 98 doğrulukta sonuçlar elde etmiştir [8]. Mauro ve meslektaşları VGG16 model kullanarak % 97.58, AlexNet model kullanarak % 97.27 ve GoogleNet model kullanarak % 99.52 oranlarında doğru bir şekilde sınıflandırma yapmışlardır [9]. Akıncı, MobileNet modelini kullanarak % 99,9 doğrulukla park yerlerini sınıflandırmıştır [10]. Valipour [11] da VGGNet mimarisini kullanmış ve % 99,9 doğruluk elde etmiştir ve buna ek olarak kullanıcılara boş park yerlerini bildiren bir mobil uygulama gerliştirmiştir. Ng ve arkadaşları [12] ise doğruluk yüzdelerinden ziyade uygulamaya odaklanmıştır. Doğruluk oranlarından bahsetmemiş ancak bu tür bir uygulamanın yararlı olup olmadığı konusunda bir anket yapmışlardır. Ayrıca Guiseppe'nin CNN eğitim yöntemini kullanmışlardır. Bu çalışmalara bakıldığında, önceden eğitilmiş modellerle yapılan sınıflandırmalarda yüksek derecede doğruluk elde edildiği görülmektedir.

1.2 Tezin Amacı

Günlük hayatta hemen hemen herkesin özel aracıyla bir yerden bir yere gitmesi yaygınlaşmıştır. Günümüzde kapalı otoparklarda sensörler ve tabelalar yardımıyla dolu-boş tespiti yapılmaktadır. Otoparkta hangi katta kaç tane boş yer olduğunu gösteren tabelalar da kullanılmaya başlanmıştır. Bu durum herkesin hayatını kolaylaştırmıştır. Açık alanlarda da aynı konforu ve teknolojiyi sağlayarak günlük hayattaki konforun arttırılması amaçlanmıştır.

1.3 Hipotez

Literatürde, bu çalışmadaki veri kümelerini kullanarak yapılan sınıflandırıcılarda henüz kullanılmamış olan ResNet modeliyle bir model oluşturarak bir sınıflandırıcı uygulaması öne sürülmüştür. ResNet modelinin de bu gibi problemlerin çözümünde başarılı sonuçlar vereceği gösterilmiştir. Veri kümelerindeki sayısal dengesizliği

çözmek adına, kullanılan veri kümesinden bir tanesi daha küçük bir gruba ayrılarak kullanılmıştır. Çalışmanın sonunda bir veri kümesiyle eğitilen model, diğer veri kümesiyle test edilerek çapraz bir test uygulanması ön görülmüştür. Bu sayede model bir kere eğitildikten sonra hangi kameradan hangi şartlarda görüntü alınırsa alınsın doğru bir sınıflandırma yapacaktır varsayımı yapılmış ve bu şekilde test edilmiştir.

2
YÖNTEM

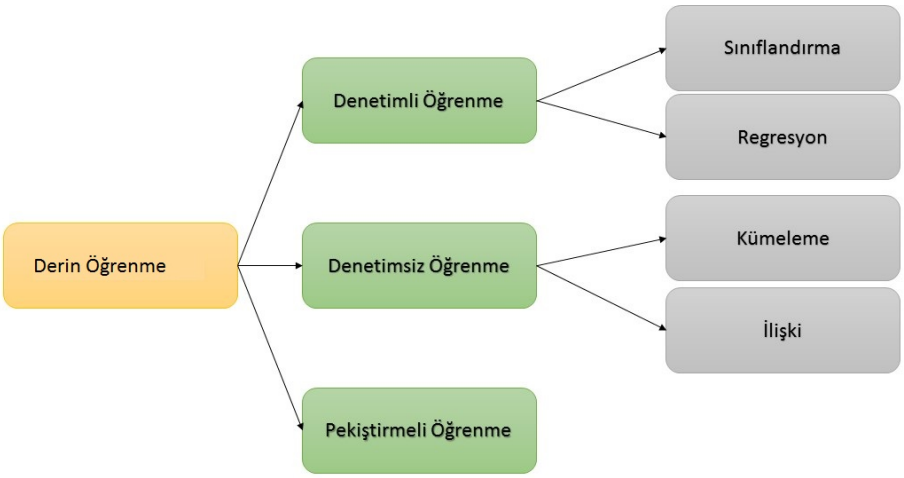
2.1 Derin Öğrenme

Günümüzde en popüler makina öğrenmesi olan derin öğrenme yapay zeka alt başlığında incelenebilir. Yapay Zeka, bilgisayarların bazı etkinlikleri insanlara benzer şekilde uygulama yeteneğidir [13]. Yapay Zeka, makinaların problem çözme yeteneklerinin insan kadar gelişmiş olması olarak açıklanabilir. Temelleri 1950'lere dayanan bu terim daha derin incelenecek olursa, basit bir yapay zekanın kendisine öğretilen çözümleri uygulayabildiği, daha yetenekli yapay zekaların ise kendi kendilerine öğrenebilen sistemler oldukları gözlenebilir. Bu öğrenmeyi de çeşitli algoritmaları kullanarak hataları optimize ederek çözdükleri söylenebilir.

Makine Öğrenmesi'nin tarihi ise 1980'lere dayanır. Bu yıllarda popüler olması ise veri madenciliğinin kullanımının artmasıyla açıklanabilir çünkü bu terim, büyük veri setlerini kullanarak öğrenmeyi ifade eder. Öyle ki bu öğrenme türü, insan hatasını minimize edebilecek kadar güvenilir sonuçlar sunmaktadır. Günümüzde bazı hastalıkların tespitinde insandan daha yüksek doğrulukta tespitler yapması, büyük verilerden kendi kendini eğitmesiyle açıklanabilir.

Derin öğrenme, iç içe birden fazla katmandan oluşan bir makine öğrenmesi sistemidir. Makine öğrenmesine ilave olarak, birçok katmandan oluşması ve geri yayılım yöntemleriyle parametreleri güncellemesiyle, bu alanda çok daha başarılı sonuçlar elde edilmektedir.

Derin öğrenme de kendi içinde üç kategoriye ayrılır, bunlar denetimli öğrenme(supervised learning), denetimsiz öğrenme(unsupervised learning) ve pekiştirmeli öğrenmedir(reinforcement learning). Şekil 2.1'de derin öğrenme kategorileri gösterilmiştir.



Şekil 2.1 Derin öğrenme yöntemleri

2.1.1 Denetimli Öğrenme

Denetimli Öğrenme, verilerin önceden bilinen şekilde etiketlendiği derin öğrenme çeşididir. Genel olarak, denetimli öğrenme, bir sisteme nasıl eşleştirildiklerini öğrenmek amacıyla giriş ve çıkış değişkenleri verildiğinde gerçekleşir. Amaç, yeni girdi verildiğinde algoritmanın çıktıyı tahmin edebileceği kadar doğru bir haritalama işlevi üretmektir. Bu yinelemeli bir süreçtir ve algoritma her tahmin yaptığında, kabul edilebilir bir performans seviyesine ulaşana kadar düzeltilir veya geri bildirim verilir. Denetimli öğrenmeye yönelik eğitim verileri, eşleştirilmiş girdi verilerine ve istenen çıktıya sahip bir dizi örnek içerir. Örneğin, görüntü işleme için denetimli öğrenmenin uygulanmasında, araba veya kamyon gibi kategorilerdeki araçların etiketli resimlerini içeren bir sistem sağlanabilir. Yeterli miktarda gözlem yapıldıktan sonra, sistemin etiketlenmemiş görüntüleri ayırt edebilmesi ve kategorilere ayırabilmesi beklenir.

Denetimli öğrenmenin uygulamaları genellikle iki kategoriye ayrılır, sınıflandırma ve regresyon. Bir regresyon problemi, çıktının fiyat veya ağırlık gibi gerçek, hesaplanmış bir değer olması durumudur.

- **Sınıflandırma (Classification):** Sınıflandırma, çıktı değerinin bir kategori olması olarak açıklanabilir. Örneğin otomobil ve kamyon veya kırmızı ve mavi. Buradaki amaç, belirli bir sınıfa ait ayrık değerleri tahmin etmek ve doğruluk temelinde değerlendirmektir.İkili veya çoklu sınıflandırma olabilir. İkili sınıflandırmada, model 0 ya da 1'i tahmin eder; ancak çoklu sınıflandırma

durumunda, model birden fazla sınıfı öngörmektedir.

- **Regresyon (Regression):** Çıktının sürekli bir değere sahip olduğu bir denetimli Öğrenme çeşididir. Büyüklük, fiyat vb. gibi özellikleri göz önüne alındığında, bir evin fiyatlarının tahmin edilmesi regresyona örnek olarak verilebilir.Buradaki amaç, modelin olabildiğince gerçek çıktı değerine daha yakın bir değeri tahmin etmesidir ve daha sonra değerlendirme hata değerini hesaplayarak yapılır. Hata ne kadar küçükse, regresyon modelinin doğruluğu o kadar yüksek olur.

2.1.2 Denetimsiz Öğrenme

Denetimsiz öğrenme, bir algoritmaya eğitim seti olarak karşılık gelen çıktı değerleri olmadan yalnızca girdi verisi verildiğinde gerçekleşir. Denetimli öğrenmenin aksine, doğru çıktı değerleri ya da öğretmenler yoktur. Bunun yerine, veriler hakkında daha fazla bilgi edinmek ve ilginç bulgular sunmak için algoritmalar serbestçe çalışabilir. Denetimsiz öğrenme, kümelenme uygulamalarında veya veri içindeki grupları bulma ve ilişkilendirme ya da verileri tanımlayan kuralları öngörme eyleminde popülerdir. Denetimsiz öğrenme uygulamalarında, veri analizi ve tahminleri arasında ilişkilendirme yapılır. Analizi yapılan verilere ait olan en iyi tahmin sonucuna ulaştırırlar.

- **Kümeleme (Clustering):** Mevcut etiketlenmemiş verilerdeki doğal gruplamaları bulmak için kullanılan bir yöntemdir. İyi bir kümelenme için kriter yoktur. Kullanıcıya, ihtiyaçlarını karşılayan kullanabilecekleri kriterlerin ne olduğuna bağlıdır.
- **İlişki (Association):** İlişkilendirme yöntemi, X'i alan insanların Y'yi de alma eğiliminde olduğu gibi kuralları keşfetmek için kullanılır.

2.1.3 Pekiştirmeli Öğrenme

Pekiştirmeli öğrenme problemleri, sayısal bir ödül sinyalini en üst düzeye çıkarmak için ne yapılması gerektiğini, durumları eylemlerle nasıl eşleştireceğini öğrenmeyi içerir. Temel bir yolla, döngüsel sorunları çözerler, çünkü öğrenme sisteminin davranışları daha sonraki girdileri etkilemektedir. Ayrıca, makinaya, birçok makine öğrenim biçiminde olduğu gibi hangi eylemlerin gerçekleştirileceği söylenmez, bunun yerine, hangi eylemlerin onları deneyerek en fazla ödülü sağladığını keşfetmesi gerekir. En ilgi çekici ve zorlayıcı vakalarda, eylemler yalnızca ani ödüllendirmeyi değil, bir sonraki durumu ve bundan sonraki tüm ödülleri etkileyebilir. Pekiştirmeli öğrenmenin aşağıdaki üç özelliği;

- Temel bir şekilde kapalı döngü olması,
- Hangi eylemlerin gerçekleştirileceği konusunda doğrudan engellerin olmaması,
- Ödül sinyalleri

dahil olmak üzere eylemlerin sonuçlarının uzun süreler boyunca ortaya çıkması pekiştirici öğrenme problemlerinin en önemli ayırt edici özellikleridir [14].

2.1.4 Konvolüsyonel Sinir Ağları

Konvolüsyonel Sinir Ağları, görüntü işlemeden ses işlemeye, pek çok alanda etkili sonuçlar vermektedir. Geleneksel YSA'lardaki parametre sayısını azaltması dolayısıyla bu ağlar, araştırmacıları daha büyük modellerle problem çözmeye itmiştir. Daha az parametreyle daha karmaşık problemleri çözmesindeki yaklaşım, CNN'lerin uzamsal niteliklere bağımlı olmamalarıdır. Bir yüz tanıma algoritması düşünüldüğünde, yüzün resimdeki uzamsal niteliği, yani resmin neresinde konumlandığının önemi yoktur. Yalnızca resimde bir yüz olup olmadığını bulmak gerekir. Bu ağların en önemli diğer bir yönü ise katmanlar derinleştiğinde soyut özellikler elde edilir. Örnek vermek gerekirse, görüntü sınıflandırma işleminde, ilk katmanlarda kenarlar algılanır daha sonra ikinci katmanlarda daha basit şekiller algulanır. Sonraki katmanlardaki yüz gibi daha yüksek seviye özellikleri ortaya çıkarılır [15].

Konvolüsyon doğrusal işlemlerin özel bir türü olarak düşünüldüğünde, konvolüsyonel sinir ağları en az bir katmanında konvolüsyon işlemi yapılan bir sinir ağı türü olarak tanımlanabilir. CNN'lerin katmanları; konvolüsyonel katman, doğrusal olmayan katman, havuzlama katmanı ve tam bağlı katman olarak sayılabilir. Konvolüsyonel ve tam bağlantılı katmanlar parametrelere sahiptir, ancak havuzlama ve doğrusal olmayan katmanlar parametrelere sahip değildir. Resimlerden nitelik çıkarma işlemi konvolüsyon katmanlarında, sınıflandırma işlemi tam bağlı katmanlarda, ne kadar olasılıkla eşleştiği ise doğrusal olmayan katmanda gerçekleşir.

2.1.4.1 Konvolüsyon

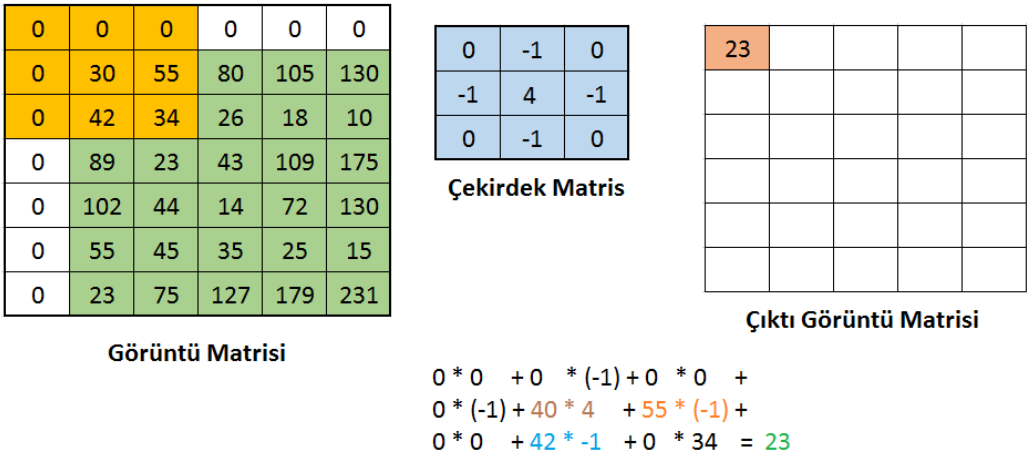
Konvolüsyon katmanı, yerel bağlantılara ve paylaşılan özelliklerin ağırlıklarına sahip olan konvolüsyonel sinir ağının ana parçasıdır. Konvolüsyon katmanının amacı, görüntünün özelliklerini öğrenmektir. Konvolüsyon katmanı birkaç özellik haritasından oluşur. Aynı özellik haritasındaki her bir nöron, önceki katmandaki farklı konumların yerel özelliklerini çıkarmak için kullanılır [16].

Konvolüsyon işleminde bir çekirdek matrisi, görüntü matrisinin üzerinde birer eleman kaydırılarak matematiksel çarpım ve toplam işlemleriyle elde edilen sonuçları, matrisin ilgili elemanına yazılarak işlenir ve sonucunda yeni bir görüntü matrisi elde edilir. Bu sayede elde edilen matris, özellik haritası olarak adlandırılır. Çünkü bu işlem sonucunda kenarlar, köşeler gibi özelliklere sahip bir görüntü matrisi ortaya çıkar. Bu sonuçlar, lineer olmayan bir aktivasyon fonksiyonuna girdi olarak verilir. Denklem 2.1 ve 2.2'de f girdisiyle h çekirdeğinin konvolüsyon işlemi anlatılmıştır. Şekil 2.2'de konvolüsyon işlemi görülebilir.

$$f * h = \sum_k \sum_l f(k,l)h(i-k, j-l) \tag{2.1}$$

$$f = \begin{pmatrix} f_1 & f_2 & f_3 \\ f_4 & f_5 & f_6 \\ f_7 & f_8 & f_9 \end{pmatrix} h = \begin{pmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{pmatrix}$$

$$f * h = f_1h_9 + f_2h_8 + f_3h_7 + f_4h_6 + f_5h_5 + f_6h_4 + f_7h_3 + f_8h_2 + f_9h_1 \tag{2.2}$$

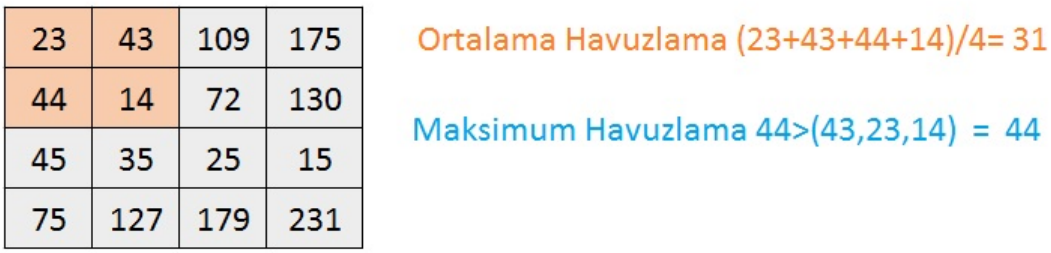


Şekil 2.2 Konvolüsyon işlemi

2.1.4.2 Havuzlama(Pooling)

Havuzlama katmanı, önceki konvolüsyon katında hesaplanan özellik haritalarının uzamsal boyutunu kademeli olarak azaltır. Havuzlama katmanında genellikle maksimum havuzlama veya ortalama havuzlama kullanır [17].Bu katmanın ardındaki yaklaşım, belirli bir özelliğin orijinal girdi hacminde olduğunu bildiğimizde, kesin konumunun diğer özelliklere göre göreceli konumu kadar önemli olmadığıdır. Tahmin edeilebileceği gibi, bu katman girdi hacminin uzamsal boyutunu (uzunluk ve genişlik

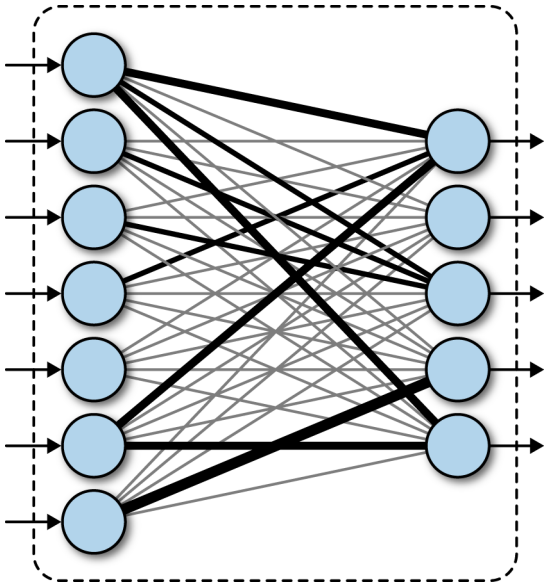
değişir, ancak derinlik değişmez) büyük ölçüde azaltır. Bu iki ana amaca hizmet eder. Birincisi, parametre veya ağırlık miktarının % 75 oranında azaltılması, böylece hesaplama maliyetinin azaltılmasıdır. İkincisi, ezberlemeyi(overfit) kontrol etmesidir. Bu terim genel olarak bir sınıflandırıcının veriyi genelleştirme amacından uzaklaşarak veriyi ezberlemesine verilen teknik isimdir. Ezberlemenin bir belirtisi, eğitim setinde % 100 veya % 99, ancak test verilerinde sadece % 50 alan bir modele sahip olmasıdır. Bu işlem temelde bir filtreyi (normalde 2x2 boyutunda) giriş hacmine uygular ve filtrenin etrafında dolaştığı her alt bölgeye havuzlama türüne göre bir matematiksel işlem uygulayarak bir çıktı verir. Şekil 2.3'te görülebileceği gibi, maksimum havuzlama işlemi 2x2 lik bir matristeki maksimum sayıyı, ortalama havuzlama ise o matristeki sayıların ortalamasını çıktı olarak verir.



Şekil 2.3 Maksimum havuzlama ve ortalama havuzlama işlemleri

2.1.4.3 Tam Bağlı Katman

Konvolüsyonel Sinir Ağlarının buraya kadar anlatılan sürecinin sonucu, nihai sınıflandırma kararını yönlendiren tamamen bağlantılı bir sinir ağı yapısına beslenir. Tamamen bağlı bir katman, bir ağırlık matrisine sahip olunan tek bir sinir ağı katmanı gibi davranır. Çünkü bu katmandan önce girişine beslenecek verileri hazırlayan düzleştirme(flatten katmanı) vardır. Bu katman kendisine gelen girdiyi tek boyutlu bir hale indirger. Konvolüsyonel sinir ağlarının girişleri genllikle tek boyutlu dizilerdir. Konvolüsyonel ve havuzlama katmanlarının sonuç matrislerinin tek boyutlu diziye çevrilmesiyle bu işlem gerçekleştirilir. Tam Bağlantılı katman, Şekil 2.4'te gösterildiği gibi bütün katmanların birbirine bağlanması olarak düşünülebilir. Konvolüsyonel ve havuzlayıcı katmanlardan öğrenilen özelliklerin çoğu iyi olabilir, ancak bu özelliklerin birleşimleri daha iyi sonuçlar verebilir. Tam bağlı katmanın amacı doğrusal olmayan bu birleşimleri öğrenmektir.

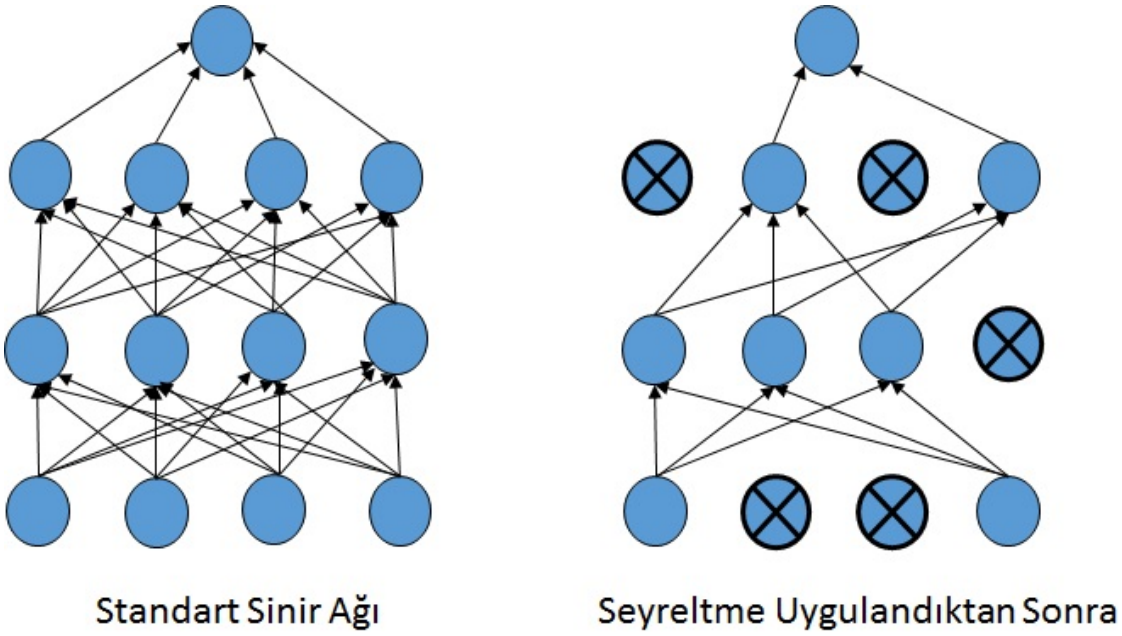


Şekil 2.4 Tam bağlı katman

2.1.4.4 Seyreltme(Dropout) Katmanı

Büyük bir ileri beslemeli sinir ağı küçük bir veri setiyle eğitildiğinde, genellikle test verisinde kötü performans gösterir. Bu ezberlemeden kaynaklıdır ve her eğitim anında özellik dedektörlerinin yarısı ihmal edilerek azaltılabilir. Buna seyreltme denir. Algoritmanın arkasındaki ana motivasyon, nöronları sağlam olmaya zorlayarak ve diğer spesifik ünitelerin aktivitelerine değil, popülasyon davranışına dayanarak, özellik dedektörlerinin birlikte adapte olmasını veya ezberlemesini önlemektir [18]. Şekil 2.5'de bir sinir ağına seyreltme uygulandıktan sonraki hali verilmiştir.

- Seyreltme değeri 0 ile 1 arasında seçilen bir değerdir. '1' seçilirse hiç seyreltme yok demektir. '0' ise katmandan çıkış yok demektir.
- En yaygın kullanılan seyreltme değeri 0.5'tir, bu da ağın yarı yarıya seyreltilmesi demektir.
- Her bir katmanda farklı seyreltme değeri atanabileceği gibi, tek bir değer bütün ağ için kullanılabilir.



Şekil 2.5 Seyreltme işlemi

2.1.4.5 Hiper Parametre Seçimleri

Makine Öğrenimi modelleri iki farklı parametre türünden oluşur:

- **Hiperparametreler:** Eğitime başlamadan önce kullanıcı tarafından keyfi olarak ayarlanabilecek tüm parametrelerdir. Modelin ilk başta nasıl yapılandırıldığını belirler.
- **Model parametreleri:** İstenen çıktıyı elde etmek için girdi verilerinin nasıl kullanılacağını tanımlar ve eğitim zamanında öğrenilir.

Hiperparametrelerin çoğu makine öğrenme algoritmaları için ayarlanması gereken ve doğru seçilmeleri önem arz eden parametrelerdir. Hiperparametre ayarı sıklıkla bir algoritmanın performansının vasat mı yoksa son teknoloji mi olduğuna karar verir, bu nedenle, hiperparametre ayarının görevi yeni algoritmalar geliştirmek kadar önemlidir. Hiperparametre performansı veri setine bağlıdır, yani bir veri setinde harika olan hiperparametre konfigürasyonları diğeri için düşük performans sağlayabilir [19].

Hiperparametre seçimi, bir çeşit optimizasyon problemi ve çözümü olarak düşünülebilir. Hatayı minimize edecek parametreleri ayarlamak manuel olabileceği gibi, son yıllarda bu sorunun çözümü için de algoritmalar geliştirilmeye başlanmıştır. Manuel de olsa otomatik de olsa yapılan işlemler sonucunda birden fazla parametre seti elde edilebilir. Bu parametrelerin her birinin modelin başarımında istenen etkiyi

elde edip etmediği incelenmeli ve optimum sonuca ulaşılmalıdır. Bir model ve bir sorunun çözümü için optimum olan bir parametre seti, benzer bir soruna uymayabilir. Bunun için farklı parametreler denenmelidir.

2.1.4.6 Aktivasyon Fonksiyonları

Aktivasyon fonksiyonları, sinir ağlarının çıktısını belirleyen matematiksel denklemlerdir. Fonksiyon, ağdaki her bir nörona bağlanır ve her bir nöronun girişinin modelin öngörüsü ile ilgili olup olmadığına bağlı olarak etkinleştirilmesinin yapılıp yapılmayacağını belirler. Aktivasyon fonksiyonları ayrıca her bir nöronun çıkışını 1 ile 0 arasında veya -1 ile 1 arasında bir değere normalleştirmeye yardımcı olur.

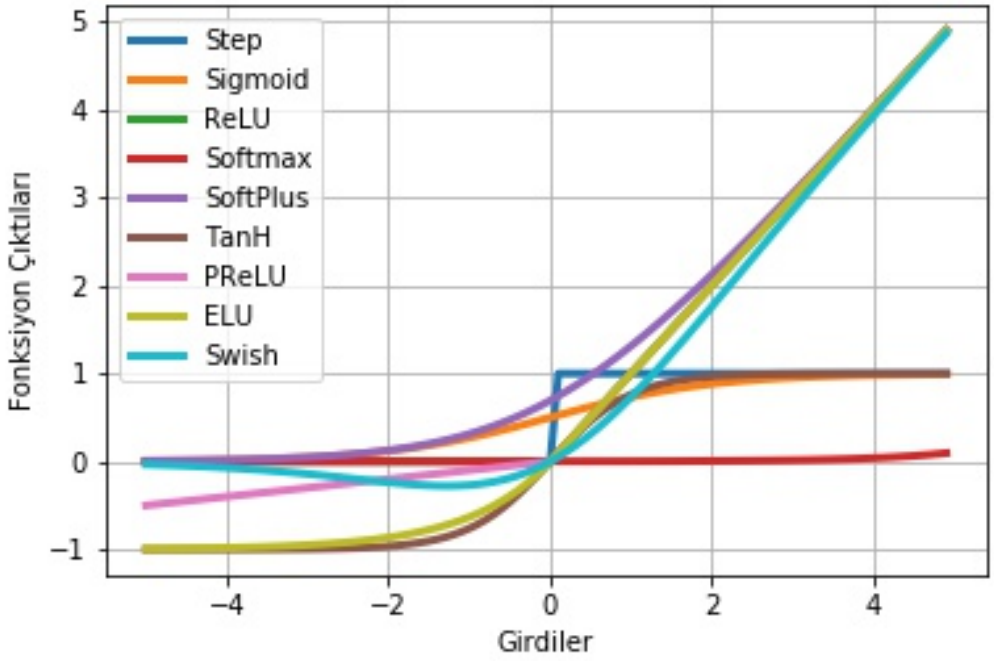
Aktivasyon fonksiyonlarının ek bir yönü de, her bir veri örneği için binlerce hatta milyonlarca nöronda hesaplanmaları nedeniyle hesaplama açısından verimli olmaları gerektiğidir. Modern sinir ağları geri yayılım olarak adlandırılan bir işlev gerçekleşitirir, bu işlev de nöronun aktivasyon fonksiyonuna ve onun türev fonksiyonuna bağlıdır. Hata oranını azaltmak için gradyan iniş veya başka bir optimizasyon tekniğini kullanarak ağırlıkları optimize etmek için bu teknik kullanılmaktadır.

Aktivasyon fonksiyonları doğrusal olmayan ya da bir başka deyişle non-lineer problemlerde etkili bir rol oynar. Doğrusal olmayan fonksiyonlar türevleri alınabilir oldukları için derin öğrenme problemlerinde etkili sonuçlar verir. Eğer seçilen fonksiyon lineer bir fonksiyon olsaydı, türevi alındığında sabit bir değer vereceği ve girişle korele olmayacağı için geri yayılım katmanında ağırlıkların güncellenmesine olanak sağlamazdı.

Geri Yayılım(Backpropagation): Bir sinir ağı, girdi verilerinin ağırlıklarını, hesaplama boyunca ileri doğru yönlendirir ve daha sonra hata hakkındaki bilgiyi ağ üzerinden tersine geçirerek parametreleri değiştirebilir. Bu işem şu şekilde olur:

- Ağ, parametrelerini kullanarak veriler hakkında bir tahmin yapar.
- Ağın kayıp fonksiyonu ölçülür.
- Yanlış yönlendirilmiş parametreleri ayarlamak için geri yayılır [20].

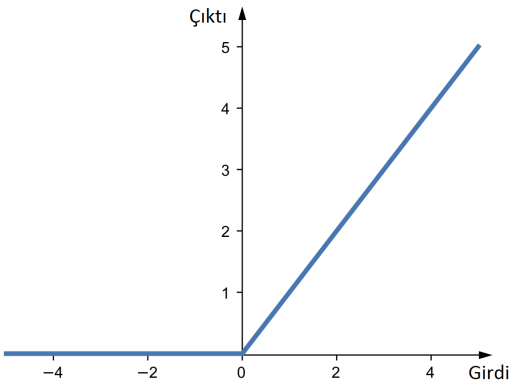
En çok kullanılan aktivasyon fonksiyonları: Sigmoid, ReLU, Softmax'tir. Bu ve diğer aktivasyon fonksiyonların grafikleri Şekil 2.6'da görülebilir.



Şekil 2.6 Aktivasyon fonksiyonlarının karşılaştırmalı grafiği [21]

- **ReLu(Rectified Linear Unit):** Aktivasyon fonksiyonlarından en yaygın kullanılanı (Rectified Linear Unit-ReLU)'dur. Aşağıda belirtilen Denklem 2.3'de görülebileceği gibi, fonksiyon girişine negatif değerler verildiğinde 0 çıktısını verirken, pozitif bir x değeri verildiğinde x çıktısını verir. ReLU fonksiyon grafiği Şekil 2.7'de gösterilmiştir.

$$\begin{aligned} f(x) &= x \qquad \forall x > 0, \\ f(x) &= 0 \qquad \forall x \leq 0 \end{aligned} \tag{2.3}$$



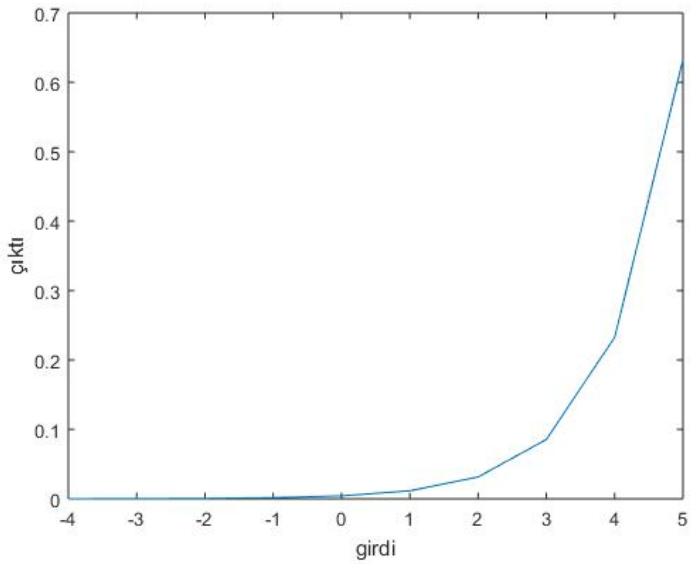
Şekil 2.7 ReLu fonksiyonu grafiği

- **Softmax:**Çok kategorili sınıflandırma için softmax katmanı kullanılır. Softmax işlevi, N boyutlu bir vektörü alır, [0, 1] arağındaki gerçek değerlere sahip N

boyutlu vektör üretir. Softmax, N boyutlu vektörü, bir sınıfa ait olma olasılığını gösteren bir olasılık dağılımına dönüştürür [22]. Softmax fonksiyonun denklemi, Denklem 2.4'de verilmiştir.

$$S_i = \frac{e^{a_i}}{\sum_{j=1}^{N} e^{a_j}} \tag{2.4}$$

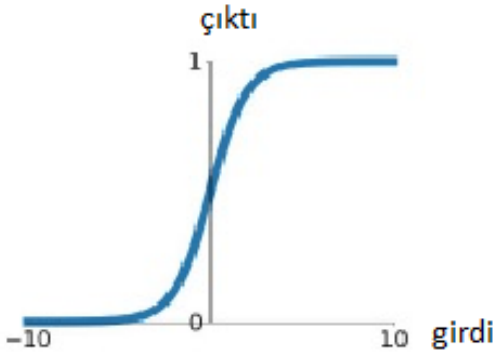
Burada, giriş vektörü [a1, a2, ... aN] olmak üzere Si, N büyüklüğündeki giriş vektöründe i'inci değer için Softmax çıkışıdır. Aşağıda Şekil 2.8'de softmax fonksiyonunun grafiği görülebilir.



Şekil 2.8 Softmax fonksiyonu grafiği

- **Sigmoid:** Sigmoid, türev alınabilir ve sürekli bir fonksiyondur. Sigmoid fonkisyonunun ismi fonksiyonun andırdığı S harfinden (sigma) gelmektedir. Denklem 2.5'te görüleceği üzere, sigmoid fonksiyonu 0 ile 1 arasında pozitif sayılar üretir bu nedenle genellikle ikili sınıflandırma işlemlerinde kullanılır. Son katmanda kullanılması yaygındır. Sigmoid fonksiyon grafiği Şekil 2.9'da gösterilmiştir.

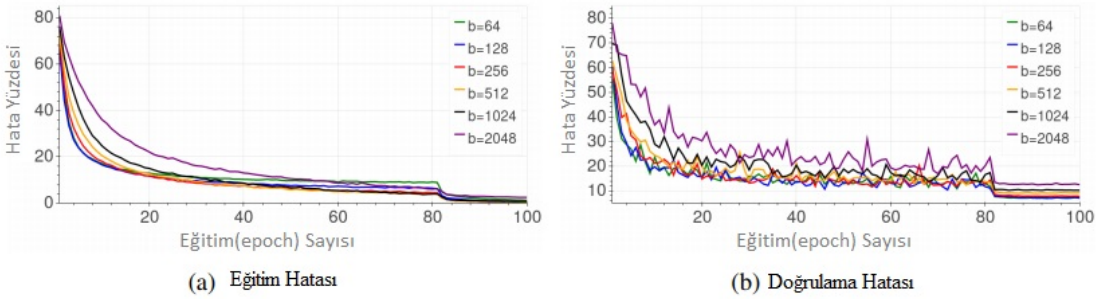
$$\frac{1}{1+e^{-z}} \tag{2.5}$$



Şekil 2.9 Sigmoid fonksiyonu grafiği

2.1.4.7 Mini Yığın Kümesi Boyutu(Mini-batch)

Mini-yığın kümesi, modelin tek seferde kaç veriyi işleyeceği anlamına gelen bir parametredir. Bir derin öğrenme modelinin işlediği veri sayısı ne kadar fazla ise aldığı süre de o kadar fazladır. Bu sebeple işlenen veri küçük gruplara ayrılarak sinir ağlarına verilir. Eğer büyük yığın kümesi kullanılacak olursa, "genelleme aralığı" olarak bilinen, genelleme performansında kalıcı bir bozulma olduğu görülmüştür. Bu boşluğun kökeninin belirlenmesi ve kapatılması açık bir sorun olarak kalmıştır [23]. Şekil 2.10'daki grafikte mini-yığın kümesi uygulamasının eğitim ve doğrulama hatalarına etkileri görülebilir. Doğrulama hatasındaki dalgalanmalar ise her bir eğitim adımında farklı kümeler işlendiği için bazı parametrelerin o kümeye uygun olup olmayışı olarak açıklanabilir.



Şekil 2.10 Mini-yığın kümesi sayısının eğitim hatası ve doğrulama hatasına etki grafikleri [23]

Mini-yığın kümesi boyutu en küçük 1 en büyük ise eğitim kümesindeki tüm verilerin sayısı olarak seçilir. Bu değer 1 olarak seçilirse her eğitim döngüsünde tek bir veri üzerinde işlem yapılır. Bu durumun olası sonuçları ise şöyledir;

- Modelin öğrenmemesi gereken bir özelliği ya da gürültüyü öğrenebilir. Bu da grafikteki zikzakların çoğalmasıyla anlaşılabilir.
- Çözüm kümesi lokal optimumda takılı kalabilir.
- Her seferinde tek bir veri işlemek uzun süren bir işlemdir.

Mini-yığın kümesi boyutu, alabileceği maksimum değeri aldığında; eğitim kümesindeki tüm veriler eğitime girecektir bu sayede Şekil 2.10'da soldaki grafiğe benzeyecektir. Bu durumda;

- Model gürültüye daha dayanıklı olur.
- Aynı anda çok fazla veriyle çalışacağı için işlem uzun sürecektir.
- Optimizasyon işleminde büyük adımlarla ilerleme olacaktır [24].

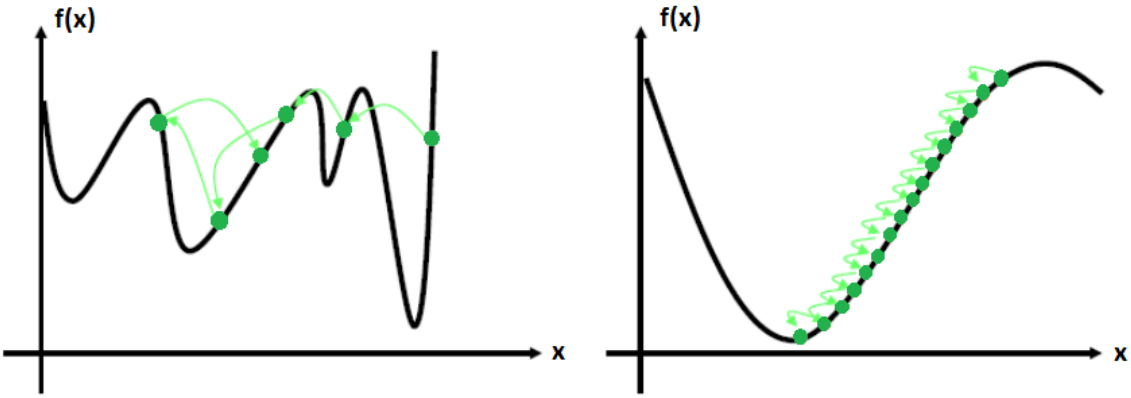
Bu değer ne çok büyük ne çok küçük bir değer seçilerek, kullanılan modele uygun bir şekilde kullanılabilir.

2.1.4.8 Öğrenme Katsayısı(Learning Rate)

Eğitim sırasında ağırlıkların güncellenen miktarına, adım büyüklüğü veya öğrenme katsayısı denir. Öğrenme katsayısı, her bir eğitim adımında ağırlıklar güncellendiğinde elde edilen hataya göre modelin ne kadar güncelleneceğini kontrol eden bir hiperparametredir. Öğrenme katsayısını seçmek zorlu bir görevdir, çünkü çok küçük bir değer uzun bir eğitim süreci ile sonuçlanabilir, buna karşın çok büyük bir değer düşük hızda veya dengesiz bir eğitim süreciyle sonuçlanabilir. Öğrenme katsayısının etkileri Şekil 2.11'de görülebilir. Şeklin sol tarafında görüleceği üzere eğer öğrenme katsayısı gereğinden fazla büyük seçilirse çözüme ulaşmak için ratgele arama yapılacaktır. Ancak sağdaki gibi çok küçük bir değer seçilirse de çözüme ulaşmak daha fazla vakit alacaktır.

Sinir ağları yapılandırırken öğrenme katsayısı en önemli hiperparametrelerden biri olabilir. Bu nedenle, öğrenme katsayısının model performansı üzerindeki etkilerini araştırmanın ve öğrenme hızının model davranışı üzerindeki dinamikleri hakkında bir sezginin oluşturulmasının hayati öneme sahiptir. Öğrenme kaysayısı için bir başlangıç değeri seçmek, sorunun sadece bir kısmıdır. Eğitim sırasında öğrenme oranının nasıl değiştirileceği de başka bir sorundur. Geçmişten gelen bilgi birikimi bize, öğrenme oranının zaman içinde düşmesi gerektiğini söyler. Bu sebepten adaptif öğrenme katasyısı yöntemleri geliştirilmeye başlanmıştır. Bunlardan bazıları, kaybın

iyileştirilmesi durduğunda tavlama adım öğrenme oranı, üstel öğrenme hızı azalması, kosinüs tavlaması vb.dir.



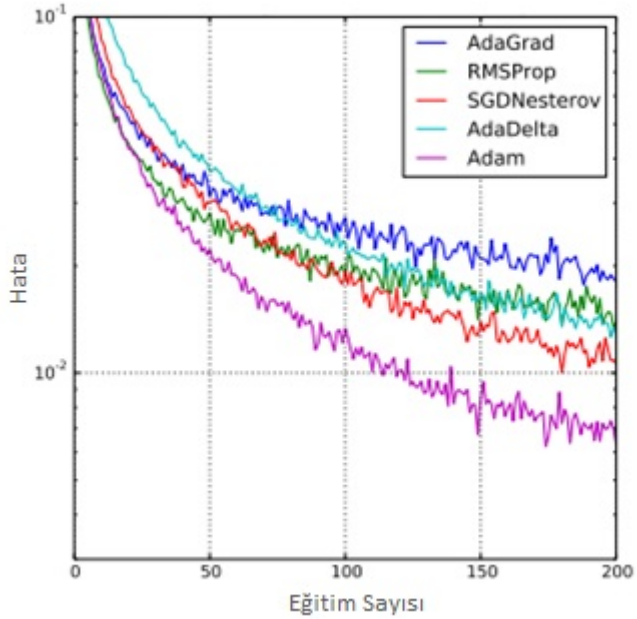
Şekil 2.11 Öğrenme katsayısı

2.1.4.9 Eğitim (Epoch) Sayısı

Eğitim sayısı ya da bilinen ismiyle epoch, algoritmanın tüm eğitim veri kümesi boyunca çalışacağı sayıyı tanımlayan bir hiperparametredir. Bir veri setinin eğitilmesi aşamasında geçtiği aşamalar, doğruluk ve kayıp fonksiyonlarının hesaplanması, elde edilen bu değerlere göre geri yayılım gerçekleştirilerek parametrelerin yani ağırlıkların güncellenmesi olarak sayılabilir. Bu işlem birden fazla kere yapılmalıdır. Kaç kez yapılacağı, model için optimize edilmesi gereken bir sayı olup, eğitim sayısı (epoch) olarak tanımlanır. Bu sayı deneme yanılma yoluyla bulunabileceği gibi, fazla ya da az oluşunun da sonuca etkileri kesin olarak bilinememektedir. Ancak çok büyük bir sayı seçilmesi ezberlemeye yol açabilir.

2.1.4.10 Optimizasyon Algoritmasının Seçimi

Modelin sınıflandırma sürecinde daha iyi çalışmasını sağlamak için, tüm ağ katmanlarının değişkenleri bir şekilde değiştirmelidir. İlk olarak, modelin öngörülen çıktısını istenen çıktıyla karşılaştırarak model performansının ne kadar iyi olduğunu bilmek önemlidir. En önemli performans ölçütlerinden biri çapraz entropidir. Çapraz entropi, daima pozitif olan sürekli bir fonksiyondur ve modelin öngörülen çıktısının istenen çıktı ile tam olarak eşleşmesi durumunda sıfır olur. Sonuç olarak, optimizasyonun amacı, çapraz entropiyi sıfıra mümkün olduğunca yakın en aza indirmektir [25]. Algoritmaların optimizasyonunda rol oynayan farklı yöntemler vardır. Sistemlerin performansı her optimizasyon tekniğine göre önemli ölçüde değişir. Aşağıdaki bölümde farklı optimizasyon algoritmaları ele alınmaktadır. Şekil 2.12'de bu algoritmaların karşılaştırmalı grafiği verilmiştir.



Şekil 2.12 Optimizasyon algoritmalarının karşılaştırılması [26]

- **Gradyan İnişi(Gradient Descent):**
  Ağırlık gradyanını sürekli değerlendirme ve ardından parametre güncellemesi yapma işlemi gradyan inişi olarak adlandırılır. Herhagi bir fonksiyonun sözgelimi bir noktadaki gradyanı, o fonksiyonun maksimum artış gösterdiği yönü göstermektedir. Bu sebeple gradyan boyunca gidildiğinde fonksiyonun değeri artar, zıt yönde gidildiğinde ise fonksiyon azalır. Gradyan inişi, nesnel bir J (Θ) fonksiyonunu azaltmak için bir yöntemdir. Buradan hareketle J (Θ) fonksiyonunun eğiminin tersi yönde parametreler güncellenerek bir modelin parametreleri belirlenir. Öğrenme katsayısı, yerel bir minimum seviyeye ulaşmak için atılan adımların boyutunu kontrol eder. Nesnel fonksiyon tarafından oluşturulan yüzeyin eğim yönü minimuma ulaşılana kadar yokuş aşağı izlenir [27].

- **AdaGrad:**
  Adaptif gradyan veya AdaGrad, daha önceden tüm tekniklerin sabit bir öğrenme oranına sahip olmasının aksine, parametrelerin her biri için öğrenme oranını değiştirir ve yaygın olarak ortaya çıkan özelliklerle ilgili parametreler için küçük güncellemeler (yani düşük öğrenme oranı) ve nadiren ortaya çıkan özelliklere sahip parametreler için büyük güncellemeler yapar.
  Denklem 2.6, 2.7 ve 2.8'de görülebileceği gibi S, şu anki ve geçmişteki kare eğim normlarının toplamı olmak üzere, bu toplamı öğrenme oranını karekköüne

bölerek öğrenme oranı bileşeni üzerinde çalışır.

$$g = \frac{1}{m}\nabla_\theta \sum_i L\left(f\left(x^{(i)};\theta\right), y^{(i)}\right) \tag{2.6}$$

$$s = s + g^T g \tag{2.7}$$

$$\theta = \theta - \varepsilon_k \times g/\sqrt{s + eps} \tag{2.8}$$

Burada, Θ model parametreleri olduğunda, g gradyan, m mini veri grubunun boyutu ve ε_k öğrenme oranıdır. Sinir ağı, f (x(i); Θ) ile temsil edilir; burada x(i), eğitim verileri, y(i) eğitim etiketleridir. L kaybının gradyanı, Θ model parametrelerine göre hesaplanır [27].

- **AdaDelta:**
  AdaDelta [28] adaptif delta'nın kısaltması olararak düşünüldüğünde delta burada, mevcut ağırlık ile yeni güncellenen ağırlık arasındaki farktır. Adadelta, öğrenme oranlarını, geçmiş tüm degradeleri biriktirmek yerine, degrade güncellemelerinin hareketli bir penceresine göre uyarlayan Adagrad'ın daha gürbüz bir halidir. Bu şekilde Adadelta, güncelleme sayısı arttığında bile öğrenmeye devam eder. Bir öğrenme oranının manuel olarak ayarlanmasını gerektirmez ve değişken gradyan bilgisine, farklı model tasarım kararlarına, farklı veri yöntemlerine ve hiperparametrelerinin belirlenmesine karşı dayanıklı davranır.

- **Adam:**
  Adam [26], Adaptif Moment Tahmini anlamına gelir. Adam, gradyanların eğitiminde yer alan her bir parametre için uyarlanabilir öğrenme katsayısını belirleyen bir tekniktir. Adam, stokastik optimizasyon için az bellek gereksinimi olan, birinci dereceden gradyanlar içeren, basit ve hesaplama açısından verimli bir yöntemdir. Bu teknik iki popüler yöntemin faydalarını birleştirir: birincisi, seyrek degradelerle uyumlu çalışan AdaGrad; ikincisi ise durağan olmayan ve çevrimiçi ayarlara uyan RMSProp'tur [29]. Denklem 2.9 ve 2.10'da Adam algoritmasının denlemleri verilmiştir.

$$m_t = \beta_1 m_t - 1 + (1-\beta_1) g_t \tag{2.9}$$

$$v_t = \beta_2 v_t - 1 + (1-\beta_2) g_t^2 \tag{2.10}$$

Burada β_1 ilk moment tahminleri için üstel bozulma oranı, β_2 ikinci moment tahminleri için üstel bozulma oranıdır. g, bulunulan mini batch'in gradienti olmak üzere, m momentumdaki geçmiş eğimlerin üssel olarak hareketli ortalaması(moving average), v ise geçmiş eğimlerin karelerinin üssel olarak hareketli ortalamasıdır. Hiperparametre β_1 genellikle 0.9 civarında tutulurken, β_2 0.99'da tutulur.

- **RMSProp:**
  RMSprop'un ana fikri, her ağırlık için kare gradyanların hareketli ortalamasını tutmaktır. Ve sonra gradyan ortalama karenin karekökü ile bölünür. Bu yüzden RMSprop (kök ortalama kare) olarak adlandırılır.

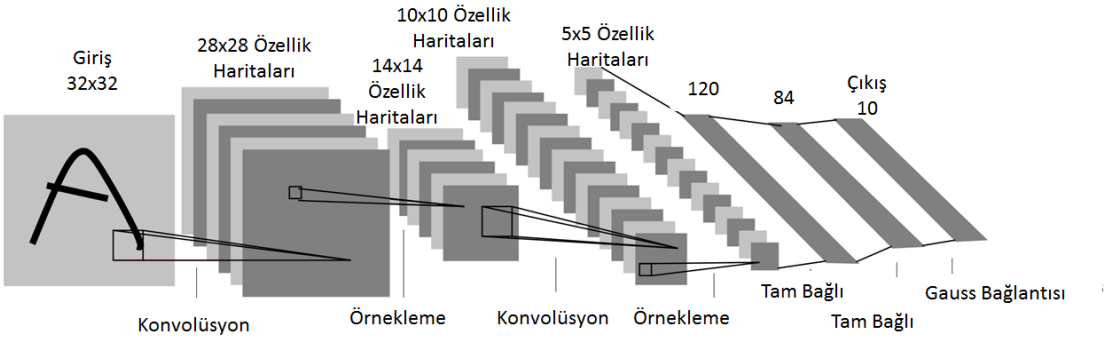
2.1.4.11 Momentum Katsayısı

Derin öğrenme yöntemlerinde kullanılan hiperparametreler, bir optimizasyon problemi gibi düşünüldüğünde, sonucun lokal optimuma takılı kalmasını önlemeyi amaçlar. Momentum katsayısı da bu problemi çözmek adına ağırlık değişim değerinin belirli bir oranda sonraki değişime eklenmesine olanak verir. Bu değer çok küçük seçilirse lokal değerden kurtulmak zorlaşır, çok büyük seçilirse tek bir çözüme ulaşmak zorlaşabilir. En yaygın kullanılan değer 0.9 olup deneysel sonuçlar bu değerin 0.6-0.8 arasında en doğru sonuçlar vereceğini göstermiştir.

2.1.5 Derin Öğrenme Ağ Yapıları

2.1.5.1 LeNet

LeNet [30] Yann LeCun ve ekibi tarafından icat edilmiş konvolüsyonel sinir ağı modelinin ilk örneğidir. 1998 yılında posta numaraları ve banka çekleri üzerindeki sayıların okunması için geliştirilmiştir. MNIST (Modified National Institute of Standards and Technology) veri seti üzerinde çalışılmıştır. Modern ağlarla kıyaslandığında LeNet çok basit bir ağdır. Sadece 7 katmandan oluşmakta olup, aralarında üç adet evrişimsel katman, iki adet havuzlama katmanı ve bir adet tam bağlı katman bulunmaktadır. Bu mimari Şekil 2.13'te görülebilir. Bu modelde diğer modellerden farklı olarak boyut azalma adımlarında maksimum havuzlama yerine ortalama havuzlama işlemi yapılmaktadır. Sigmoid ve hiperbolik tanjant kullandığı aktivasyon fonksiyonlarıdır. Her konvolüsyon katmanı 5 x 5 çekirdek kullanır. Çıkışında 0–9 arasındaki rakamları sınıflandırdığı için 10 sınıflı softmax bulunmaktadır.



Şekil 2.13 LeNet mimarisi [30]

2.1.5.2 AlexNet

AlexNet [31], 2012 yılında SuperVision grubu tarafından tasarlanmıştır. Grubun üyeleri Alex Krizhevsky, Geoffrey Hinton ve Ilya Sutskever'dir. ImageNet Büyük Ölçekli Görsel Tanıma Yarışması'da (ILSVRC) önceden elde edilen %74,3 oranındaki doğruluk oranını %83,6'ya yükselterek görüntü sınıflandırma konusunda önemli bir değişim yaratmıştır. Ard arda gelen konvolüsyon ve havuzlama katmanlarına sahip olması dolayısıyla LeNet modeline benzemektedir. Lineer olmayan katmanda ReLU fonksiyonu, havuzlama katmanlarında da maksimum-havuzlama kullanılmaktadır. Ağın mimarisi Şekil 2.14'te özetlenmiştir. Beş konvolüsyonel ve üçü tam bağlı sekiz katman içermektedir. Mimari 1000 nesneyi sınıflandıracak şekilde tasarlanmıştır. Bazı özellikleri şu şekilde açıklanabilir,

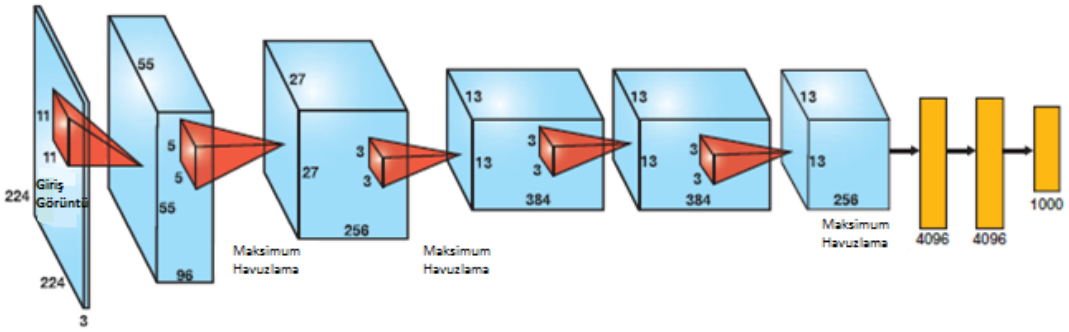
- AlexNet'in girişi 256 x 256 boyutunda bir RGB görüntüsüdür. Bu, eğitim setindeki tüm görüntülerin ve tüm test görüntülerinin 256 x 256 boyutunda olması gerektiği anlamına gelir. Giriş görüntüsü bu boyutlarda değilse, ağı eğitmek için kullanmadan önce dönüştürülmesi gerekir.

- AlexNet'in ilk katmanında evrişim penceresi boyutu 11x11'dir. ImageNet'teki görüntülerin çoğu, MNIST görüntülerinden on kat daha yüksek ve daha geniş olduğundan, ImageNet verilerindeki nesneler daha fazla piksel işgal etme eğilimindedir. Sonuç olarak, nesneyi yakalamak için daha büyük bir evrişim penceresi gerekir. İkinci katmandaki evrişim penceresi boyutu 5x5'e, ardından 3x3'e düşürülür. Ek olarak, birinci, ikinci ve beşinci evrişimli tabakalardan sonra ağ, 3x3'lük bir pencere şeklinde ve ikişer adımda maksimum havuzlama katmanları ekler.

- Son evrişim katmanından sonra, 4096 çıkışa sahip iki tam bağlı katman vardır. Bu iki büyük tamamen bağlı katman, yaklaşık 1 GB'lık model parametreleri üretir. İlk GPU'lardaki sınırlı bellek nedeniyle, orijinal AlexNet çift veri akışı

tasarımı kullanmıştır, böylece iki GPU'nun her biri modelin sadece yarısını depolamaktan ve hesaplamaktan sorumlu olacaktır.

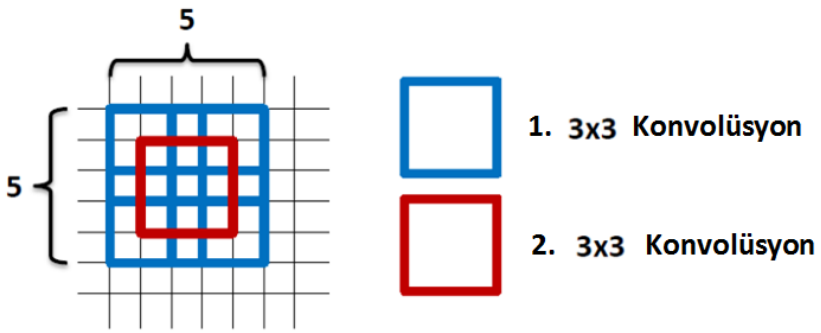
- AlexNet, LeNette kullanılan sigmoid aktivasyon fonksiyonu yerine ReLU aktivasyon fonksiyonunu kullanır. ReLU aktivasyon fonksiyonunun hesaplanması daha basittir. Örneğin, sigmoid aktivasyon fonksiyonunda bulunan üstelleştirme işlemine sahip değildir. ReLU fonksiyonu, farklı parametre başlatma yöntemleri kullanıldığında model eğitimini kolaylaştırır. Bunun nedeni, sigmoid aktivasyon fonksiyonunun çıktısının 0 veya 1'e çok yakın olması durumunda, bu bölgelerin gradyanı neredeyse 0'dır, böylece geri yayılım bazı model parametrelerini güncellemeye devam edemez. Aksine, ReLU aktivasyon fonksiyonunun pozitif aralıktaki gradyanı her zaman 1'dir. Bu nedenle, model parametreleri uygun şekilde başlatılmazsa, sigmoid fonksiyonu pozitif aralıkta neredeyse 0 derecesinde bir gradyan elde edebilir. Böylece model etkili bir şekilde eğitilemez.



Şekil 2.14 AlexNet mimarisi [32]

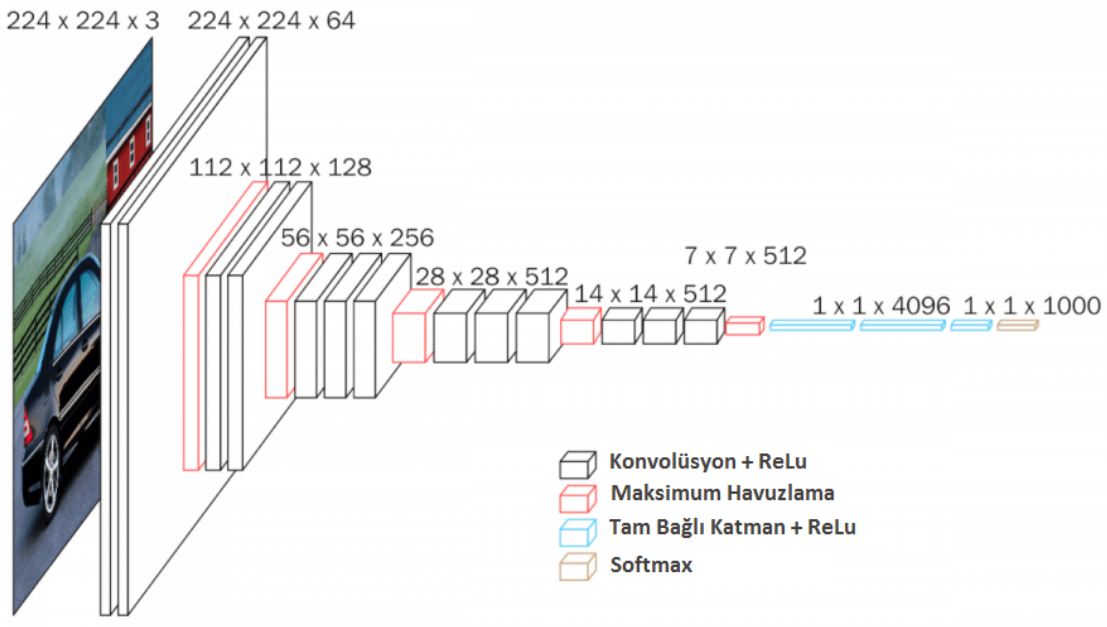
2.1.5.3 VGGNet

VGGNet [33], Karen Simonyan ve Andrew Zisserman oluşan Visual Geometry Group tarafından tasarlanmıştır. VGG ağı, üst üste istiflenmiş yalnızca 3x3 evrişimli katmanlar kullanan basitliği ile karakterize edilir. Şekil 2.15'te bu evrişimli katmanlar görülebilir. Hacim büyüklüğünün azaltılması maksimum havuzlama ile gerçekleştirilir. Her biri 4,096 düğümü olan iki tam bağlı katman, ardından softmaksimum sınıflandırıcısını takip eder.



Şekil 2.15 VGGNet filtresi [34]

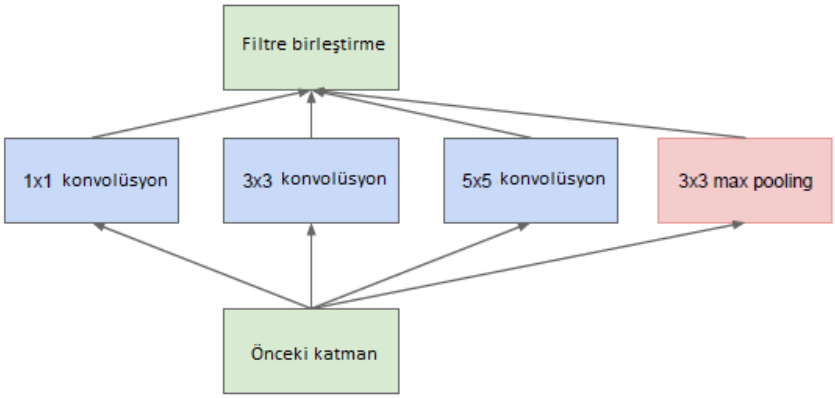
İki kat 3x3 filtre kullanarak, aslında yukarıdaki şekilde olduğu gibi 5x5 alanı kaplamıştır. 3x3 filtreden oluşan 3 kat kullanılarak, aslında 7x7 etkili alanı kapsıyor. VGGNet kendisinden önceki AlexNet gibi modellerdeki filtrenin boyutunu bu şekilde azaltarak kullanılan parametre sayısını azaltmıştır. Öğrenilmesi gereken daha az parametre olduğunda, daha hızlı sonuçlar elde edilir ve ezberleme problemi giderilebir. Model mimarisi Şekil 2.16'daki gibidir. İsimlerinden de anlaşılacağı gibi VGG-16'da 16 katman, VGG-19'da 19 katman olmak üzere iki yapıda model bulunmaktadır. VGG-16, ImageNet veri kümesi, VGG-19 ise CIFAR-10 veri kümesi için kullanılır.



Şekil 2.16 VGGNet mimarisi [35]

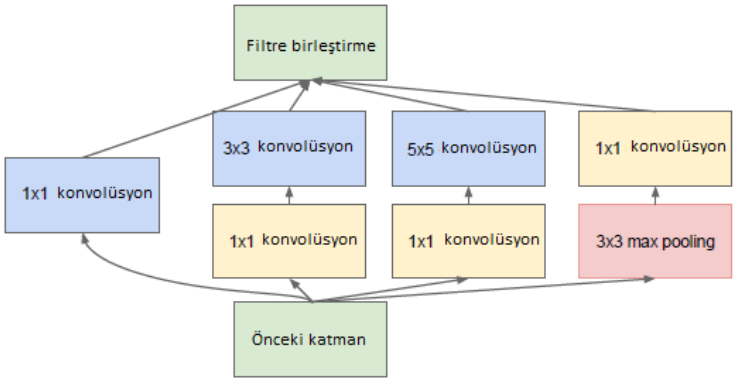
2.1.5.4 Inception(GoogleNet)

Inception(GoogleNet) [36], 2014 yılında Google'daki araştırmacılar tarafından geliştirilmiş ve 2014 ImageNet yarışmasında birincilik kazanmıştır. Ağ esas olarak "daha derinden" ziyade biraz "daha geniş"tir. Önceden, AlexNet ve VGGNet gibi ağlarda, konvolüsyonel boyut her katman için sabitken, inception ağındaki genişleme etkisi, konvolüsyonel katmanlardaki 1x1, 3x3, 5x5 filtrelerin ve 3x3 maksimum ortalama işleminin paralel olarak gerçekleştirilmesiyle sağlanmaktadır. Bu yapı "Naif Inception Modülü (Naive Inception Module)" olarak isimlendirilir. Şekil 2.17'de bu yapı görülebilir.



Şekil 2.17 Naif Inception modülü

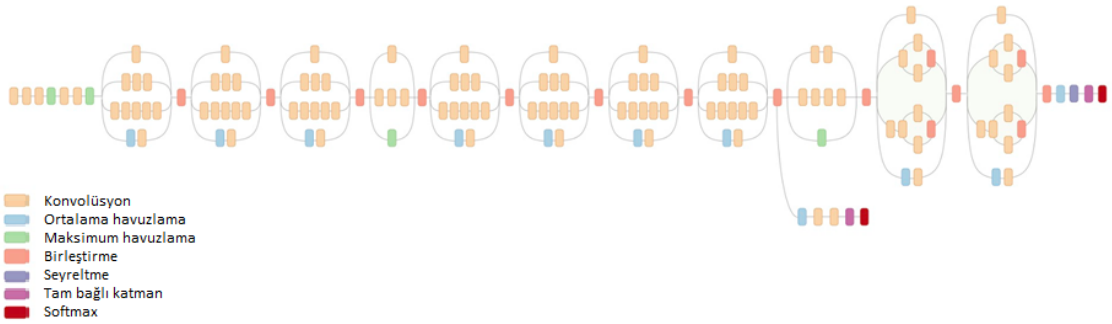
Bu modülün kullanmanın getirdiği problemlerden biri çıkış boyutunun büyük olmasıdır. Paralel işlemlerin getirdiği işlem yükü ve parametre sayısının çokluğu da çözülmesi gereken problemlerden biridir. Bunu sağlamak adına her konvolüsyon katmanından önce boyut küçültme için başlangıç modülüne 1x1 evrişim yerleştirilirek işlem yükünü optimize etmeye odaklanılmıştır. Şekil 2.18'de Boyut azaltma modülü görülebilir.



Şekil 2.18 Inception modülü

Modüllerin her biri inception olarak adlandırılır. Toplamda 9 adet inception modülü içerir. İsimlendirme konusunda ise Google ismiyle ilk derin öğrenme modeli olan

LeNet ismi birleştirilerek GoogLeNet olarak adlandırılmıştır. Şekil 2.19'da ağın mimarisi verilmiştir. Inception ismini ise bir sinema yapıtı olan ve iç içe katmanlardan oluşma temasına sahip Christopher Nolan'ın Inception filminden almıştır.



Şekil 2.19 GoogleNet mimarisi [37]

Inception modülünün çalışma mantığını açıklanacak olursa, derin öğrenme modelindeki bir katmanın, yüzün bireysel kısımlarına odaklanmayı öğrendiğini varsayalım. Ağın bir sonraki katmanı, muhtemelen orada bulunan farklı nesneleri tanımlamak için görüntüdeki genel yüze odaklanacaktır. Şimdi bunu yapmak için, katman, farklı nesneleri tespit etmek için uygun filtre boyutlarına sahip olmalıdır.

Burası inception modülünün öne çıktığı yerdir. İç katmanların, gerekli bilgileri öğrenmek için hangi filtre boyutunun uygun olacağını seçmesine olanak tanır. Dolayısıyla görüntüdeki yüzün boyutu farklı olsa bile, katman yüzün tanınması için buna göre çalışır.

2.1.5.5 ResNet

Resnet [38], artık değerli yapay sinir ağların (residual neural network) bir kısaltmasıdır. 2016 yılında Kaiming He ve arkadaşları tarafından tasarlanmış ve aynı yıl ILSVRC'de sınıflandırma alanında 1.lik kazanmıştır. Model yarışmada %3.57 oranında bir hata vermiştir. 152 katmanlı ResNet o ana dek ImageNet üzerinde koşturulan en derin ağ özelliğini taşımaktadır. Katmansal olarak kıyaslandığında VGGNet ResNet'in sekizde biri katmandan oluşmasına rağmen, ResNet daha az parametreden oluşur. Bu da daha eğitim hızını arttıran bir etken olmuştur. Resnet, geleneksel CNN modelinde derinlik için bir üst eşik olduğunu göstermektedir. Derin ağlar birleşmeye başladığında, bir bozulma sorunu ortaya çıkar: ağ derinliği arttıkça, doğruluk doygunluğa ulaşır ve sonra hızla düşer. Bu model, ağ derinleştikçe artan eğitim hatasını gidermektedir.
ResNet'in çözdüğü sorunlardan biri de kaybolan türev (gradyen) problemidir.

Bunun nedeni, ağın çok derin olması durumunda, birkaç uygulamadan sonra kayıp fonksiyonunun kolayca hesaplandığı yerdeki gradyanların sıfıra çekilmesidir. Bu sonuç, ağırlıkları hiçbir zaman değerlerini güncellemez ve bu nedenle hiçbir öğrenme gerçekleştirilmez. Bu sorunu da şu şekilde çözer, kendisinden önceki ReLU aktivasyon fonksiyonu çıkışıyla şu anki aktivasyon fonksiyonunun girişiyle toplayıp aktive eder, böylece bulunduğu katmanın lineer dönüşüm sonucu 0 olsa dahi bir değer üretir. Bu da evrişim çıkışlarında öğrenmenin durduğu koşullarda bile çıkışa bir önceki aktivasyon değerini aktarma üzerine kurulmuş bir yapıdır.

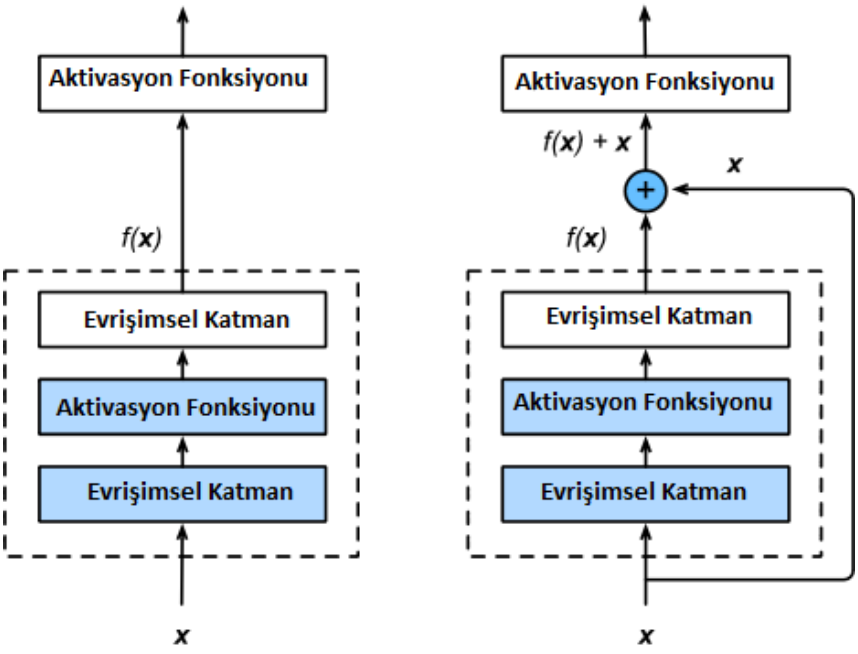
Yapay sinir ağları iyi fonksiyon belirleyicileri olduklarından, bir fonksiyonun çıktısının, girişin kendisi olduğu tanımlama fonksiyonunu kolayca çözebilmelidirler. x, giriş, f(x), bir yapay sinir ağı fonksiyonu olmak üzere Denklem 2.11'de bu durum gösterilmiştir.

$$f(x) = x \tag{2.11}$$

Aynı mantıktan sonra, ağ, kendisine eklenmiş girdi ile daha önce hangi işlevi öğrendiğini tahmin edebilmelidir. Bu durum Denklem 2.12'de görülebilir.

$$f(x) + x = h(x) \tag{2.12}$$

Sezgi, f (x) = 0 olduğunda öğrenmenin ağ için kolay olması gerektiğidir. Şekil 2.20'de Resnet modeli verilmiştir.



Şekil 2.20 ResNet modeli [39]

Yukarıda gösterildiği gibi yapay sinir ağında girdi x olmak üzere, öğrenerek elde etmek istenen ideal haritalamanın aktivasyon fonksiyonuna girdi olarak kullanılmak üzere

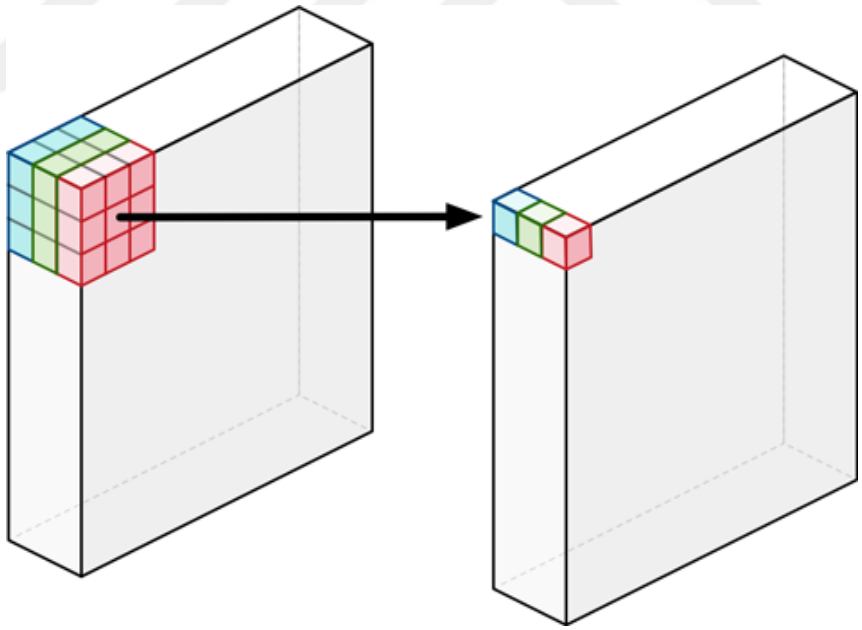
f(x) olduğu varsayılabilir. Sol tarafta noktalı çizgiyle çevrelenmiş kısım doğrudan f(x) eşlemesine uymalıdır. Sağ tarafta ise ağın eğitiminde gereksiz olan ağırlıkların f (x) = 0 olarak ayarlanarak eğitime katılmamasına olanak sağlar. Sağ taraftaki ağ yapısındaki bağlantı atlamalı bağlantı olarak isimlendirilmektedir.

Resnet-X, notasyonudaki X, ResNet'in kaç katmanlı olduğunu ifade eder. Örneğin ResNet-50 ImageNet veri kümesi üzerinde eğitilmiş 50 katmanlı bir ağdır.

2.1.5.6 MobileNet

MobileNet [40] Google'da bir grup araştırmacı tarafından tasarlanmıştır ve bu tür sinir ağlarının mobil cihazlarda çok verimli bir şekilde çalıştığını ortaya sürmüşlerdir. Bu yeni yaklaşım evrişim katmanında 3x3 çekirdekler için yaklaşık 9 kat daha hızlı ve daha etkilidir.

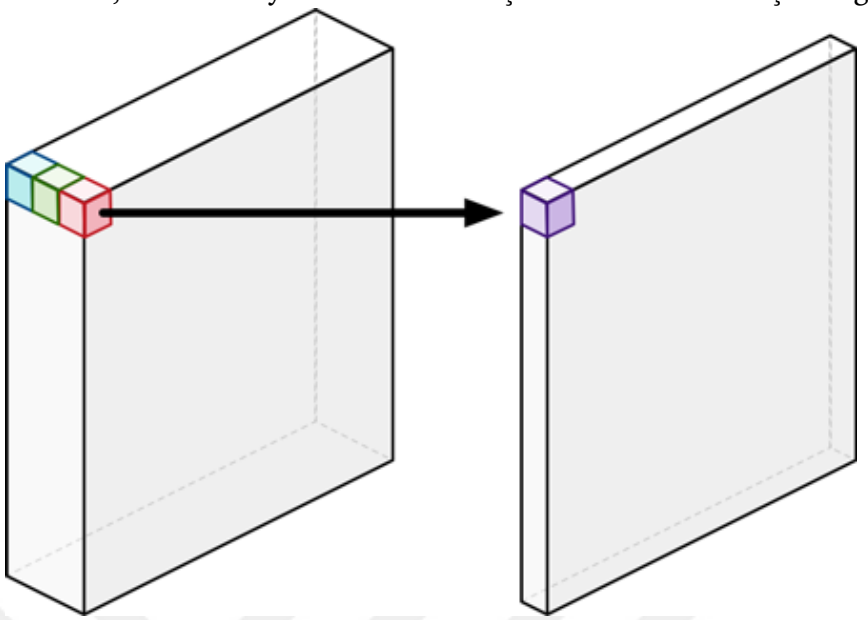
Mobilenet temel olarak her renk kanalında tek bir konvolüsyon işlemi gerçekleştirdiği anlamına gelen uzamsal ayrılabilir konvolüsyon kullanır. Uzamnsal oluşu, yükseklik ve uzunluk boyutlarını kasteder, derinlik yani renk kanalları ise diğer bir boyuttur. Şekil 2.21'de bu konvolüsyon işleminin nasıl olduğu görülebilir.



Şekil 2.21 Derinlik tabanlı konvolüsyon [41]

Konvolüsyonel katmanın işi iki alt göreve ayrılır: bunlardan ilki derinlik tabanlı ayrılabilir konvolüsyon, diğeri ise noktasal tabanlı ayrılabilir konvolüsyondur.Yukarıdaki şekilde görüldüğü gibi her bir kanal kendi içinde konvolüsyon işlemine tabi tutulduktan sonra yeni özellikler oluşturmak için bu filtreli değerleri birleştiren 1x1'lik bir konvolüsyon katmanı vardır. Bu

katmana noktasal konvolüsyon denir. 1x1'lik bir filtre yani her bir noktada itere edilen filtre, kanal sayısını azaltır. Şekil 2.22'de bu işlem görülebilir.



Şekil 2.22 Noktasal konvolüsyon [41]

İki işlem beraber çalıştığında "derinlemesine ayrılabilir"(depthwise seperable) bir konvolüsyonel blok oluşturur. Standart bir konvolüsyon işleminden tek farkı çok daha hızlı olmasıdır.

3
UYGULAMA

3.1 Kullanılan Veri Setleri

Bu çalışmada kullanılan veri setleri PKLot [42] ve CNRParktır [8, 43]. PKLot veri kümesi, güneşli,bulutlu ve yağmurlu hava şartlarında farklı park yerlerinden çekilmiş 700.000'den fazla park yeri görüntüsünden oluşmaktadır. CNRPark veri kümesinde ise, farklı gün ve hava koşullarında iki farklı kamera ile çekilmiş yaklaşık 13.000 adet park yeri görüntüsü vardır. Her iki veri kümesi de boş veya dolu olacak şekilde etiketlenmiş görüntülerden oluşmaktadır. Bu çalışmada PKLot veri seti büyüklük olarak CNRPark'tan çok daha fazla sayıda görüntüden oluşması dolayısıyla kıyaslanabilirlik açısından eşitlenmek istenmiş ve rastgele görüntüler seçilerek bir grup oluşturulmuştur. CNRPark seti ise bölümlemeden kullanılmıştır. Çalışmaya başlanmadan önce görüntüler eğitim, doğrulama ve test olmak üzere 3 parçaya ayrılmıştır. Bu ayrım sonucunda üzerinde çalışılan görüntü sayıları Tablo 3.1'de verilmiştir.

Tablo 3.1 Veri setlerindeki görüntü sayıları

	Eğitim	**Doğrulama**	**Test**
PKLot	17160	5040	2244
CNRPark	10066	3240	2518

Aşağıda Şekil 3.1'de veri setindeki dolu, Şekil 3.2'de boş park alanı görüntüleri verilmiştir.

Şekil 3.1 Dolu park alanı görüntüleri

Şekil 3.2 Boş park alanı görüntüleri

3.2 Transfer Öğrenmesi

Transfer öğrenmesi, daha önceden herhangi bir problemi çözmek için kullanılan yöntemi başka bir alanda kullanmaya denilir. Derin öğrenme uygulamalarında bu terimin kullanılması ise, önceden eğitilmiş modellerin başka problemlerin çözümü için kullanılmasına denir. Derin öğrenme uygulamaları, büyük veri kümeleriyle çalışıldığı ve bazı modellerin fazla işlem yükü gerektirmesi dolayısıyla bilgisayar işlemci birimlerinde gerçekleştirilememektedir. Bu uygulamalar için daha güçlü ve daha hızlı işlem yapan grafik işleme birimlerine ihtiyaç vardır. Bu birimlerde yapılan eğitimler sonucu elde edilen modeller farklı problemlerin çözümü için kullanılabilir. Transfer öğrenmesi üç şekilde gerçekleştirilebilir. Bunlar,

- Tüm modeli olduğu gibi kullanıp yalnızca softmax çıkışını problemin sınıf sayısına eşitleyerek,
- Modelin yalnızca son katmanlarını değiştirerek,
- Tüm ağı yeni bir veri setiyle eğiterek yapılabilir [44].

3.3 Derin Öğrenme Uygulamalarında Bulut Platformların Kullanılması

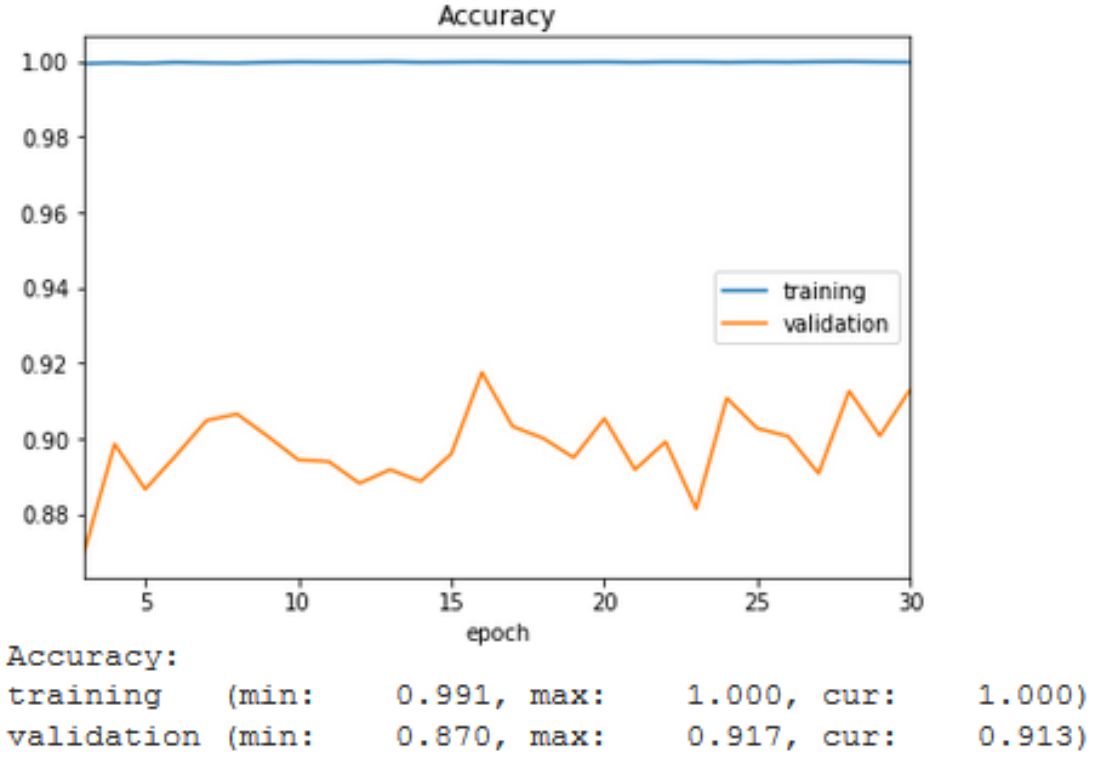
3.3.1 Google Colab

Derin Öğrenme uygulamalarında işlem hızı ve kapasitesi için GPU(Graphics processing unit) kullanmaya ihtiyaç vardır. Ancak günümüzde kendimize ait bir GPU satın almanın, maliyet, bakım ve sürdürülebilirlik açısından tartışılması gerekir. Üzerinde çalışabilecek bir GPU yok ve alabilecek bütçe de yok ise bulut platformları burada devreye giriyor. Bulut platformları, bir sunucu üzerinden uzaktan GPU'lara erişim sağlayan platformlardır. Google Colaboratory (Colab), makine öğrenmesi çalışmalarını yaymak için Jupyter Notdefterini temel alan bir bulut servisidir. Güçlü bir GPU'ya derin öğrenme ve ücretsiz erişim için tamamen yapılandırılmış bir çalışma zamanı(runtime) sağlar. Bu bulut hizmeti tarafından sağlanan çalışma zamanı, TensorFlow, Matplotlib ve Keras gibi önde gelen yapay zeka kütüphaneleriyle önceden yapılandırılmış Python 2 ve 3 çalışma zamanlarını sağlar ve ayrıca güçlü bir GPU sunar. Bu Google hizmeti bir Google Drive hesabına bağlı olarak çalışır ve ücretsizdir.
Google Colaboratory'yi tanıtmadan önce, Colaboratory'nin dayandığı teknoloji olan Jupyter Notebook'lardan bahsedilecek olursa, Jupyter, çeşitli yazılım dillerini ve kütüphanelerini birleştiren açık kaynaklı ve tarayıcı tabanlı bir araçtır. Bir Jupyter notebook yerel olarak veya bulut üzerinde çalışabilir. Her belge, her bir hücrenin komut dosyası dili veya işaretleme kodu içerdiği ve çıktıların belgeye gömülü olduğu birden çok hücreden oluşur. Tipik çıktılar metin, tablolar, çizelgeler ve grafikleri içerir. Bu teknolojiyi kullanmak, deneyler ve sonuçlar bağımsız bir şekilde sunulduğundan, bilimsel eserlerin paylaşılmasını ve çoğaltılmasını kolaylaştırır [45].

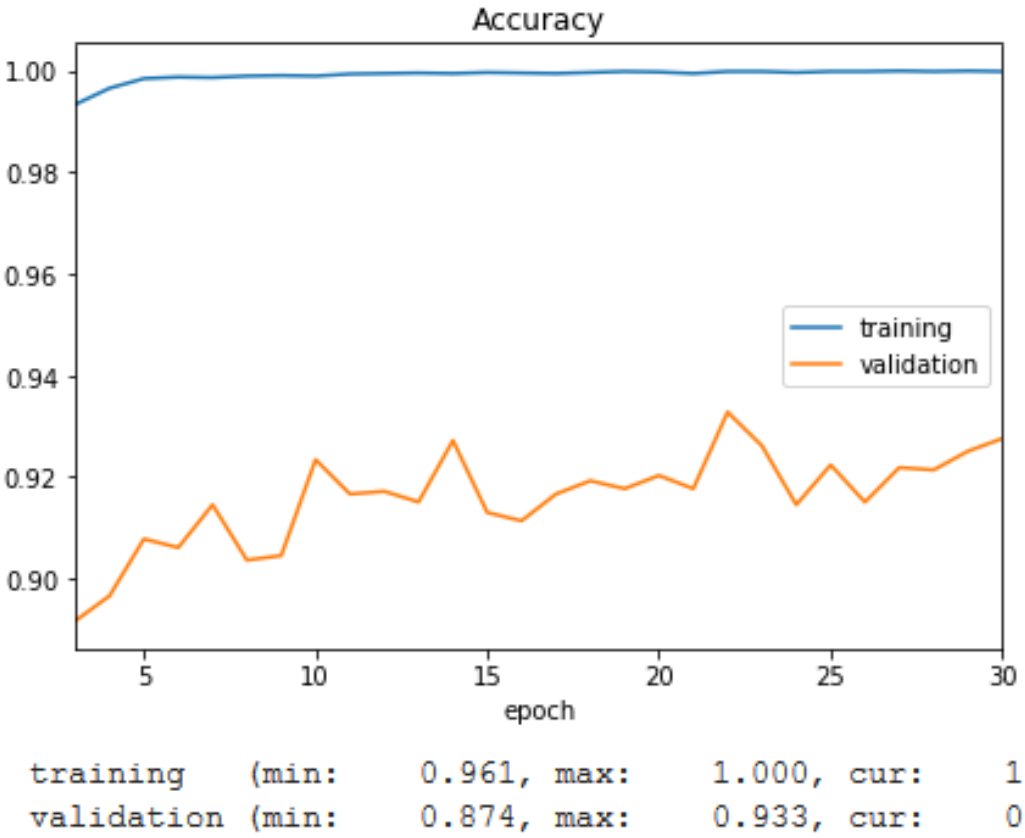
Colab üzerinden ulaşılan donanım, iki çekirdeği @ 2,3 GHz ve 13 GB RAM'e sahip bir Intel Xeon işlemcisinden oluşur. NVIDIA Tesla K80 (GK210 yonga seti), 12 GB RAM, 560 MHz'de 2496 CUDA çekirdeği ile donatılmıştır.

4
UYGULAMA SONUÇLARI

CNRPark ve PKLot datasetlerinden alınan görüntüler ResNet50 modeli kullanılarak eğitilmiştir. ResNet modelinin giriş katmanı 224x224 boyutunda görüntüler aldığı için veri kümelerinden görüntüler bu boyuta çevrilmiştir. Son katmanı Ortalama Havuzlama katmanı ve sınıflandırma için 2 kategorili softmax katmanı eklenmiştir. Stokastik Gradyen İnişi optimizasyon algoritması seçilmiş ve öğrenme katsayısı azalan bir fonksiyon olarak seçilmiştir. Uygulamada Keras kütüphanesi kullanılmıştır. Şekil 4.1'de PKLot verisetiyle yapılan eğitim grafiği, Şekil 4.2'de ise CNRPark verisetiyle yapılan eğitim grafiği verilmiştir.



Şekil 4.1 PKLot veri seti kullanarak yapılan eğitim grafiği



Şekil 4.2 CNRPark veri seti kullanarak yapılan eğitim grafiği

Eğitilen modeller çapraz test yöntemi ile test edilmiştir. Bu yöntemde ağ önce bir veri seti ile eğitilip sonra diğeri ile test edilmiştir. Sonuçlar Tablo 4.1'dedir.

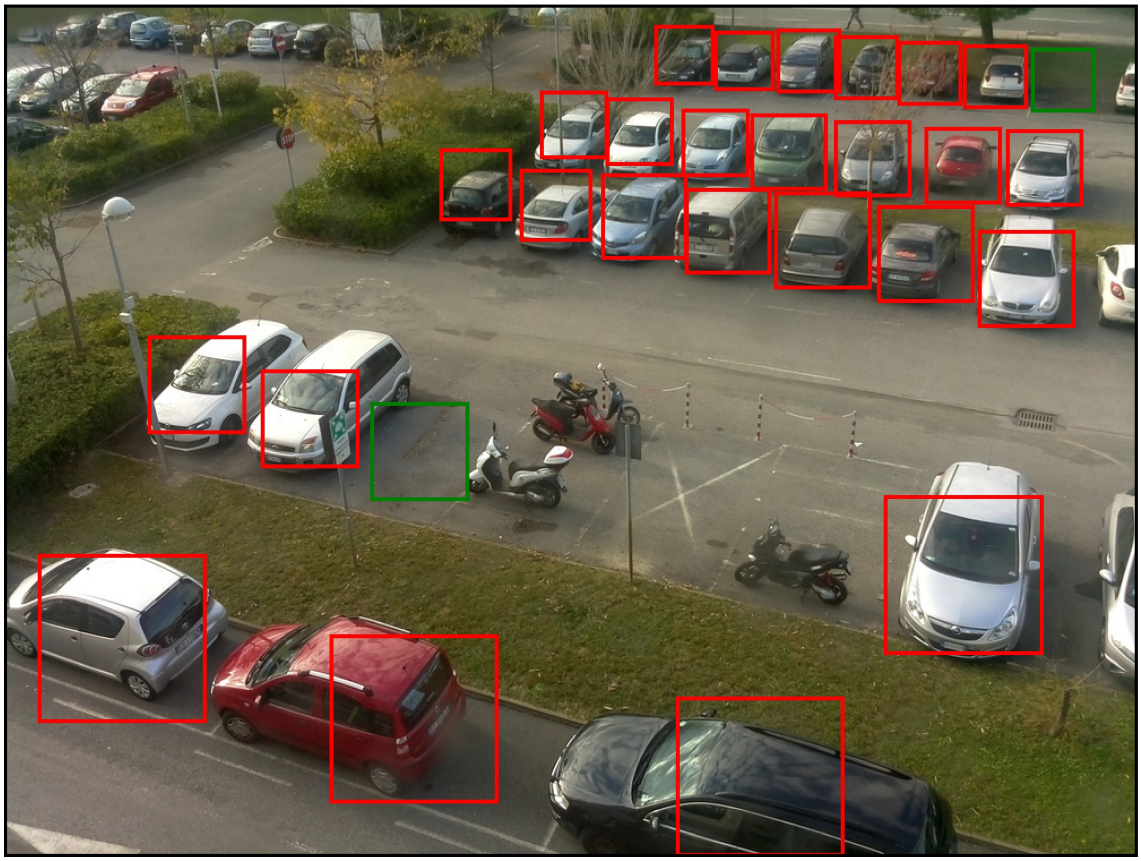
Tablo 4.1 Test doğruluk oranları

Eğitim Seti	**Test Seti**	
	CNRPark	**PKLot**
PKLot	%97,36	%99,82
CNRPark	%99,52	%99,28

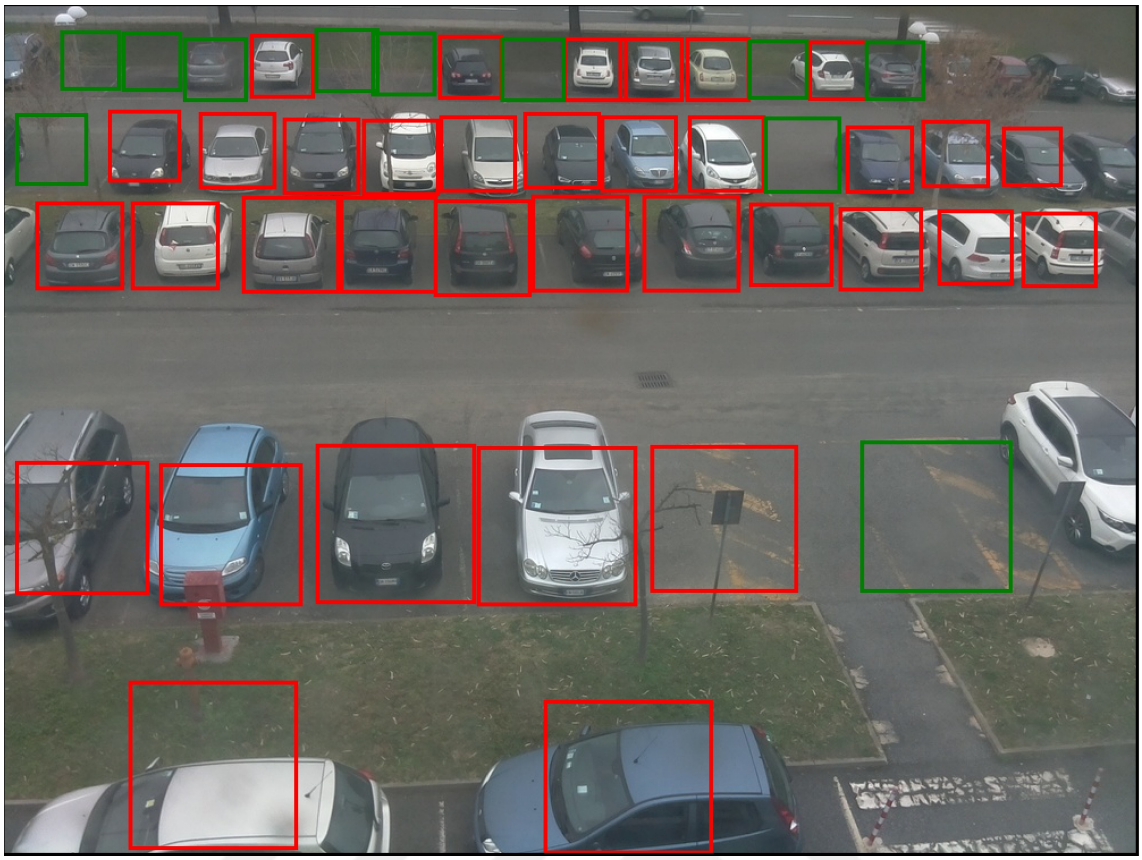
CNRPark datasetinde slotlara ayrılmış görüntüler eğitim işlemi için kullılmıştır. Bunun dışında farklı kameralardan alınan ve bütün bir park alanını gören görüntüler de veri kümesinde paylaşılmıştır. Her bir slotun görüntüdeki konum koordinatları [x,y] ve uzunluk ve genişlik bilgileri [h,w] de bir csv dosyası olarak paylaşılmıştır. Bu koordinatlar kullanılarak büyük görüntüden kısımlar kırpılarak test edilmiş ve sonucuna göre dolu ya da boş olarak sınıflandırılmıştır. Bu sonuca göre dolu olduğu tespit edilen slotlar kırmızı ile, boş olduğu tespit edilen slotlar yeşil ile kare içine alınmıştır. Sonuçlar aşağıdaki şekillerde görülebilir.

Şekil 4.3 ve Şekil 4.7 'deki sonuçlarda modelin hatasız tespit yaptığı gözlenebilir.

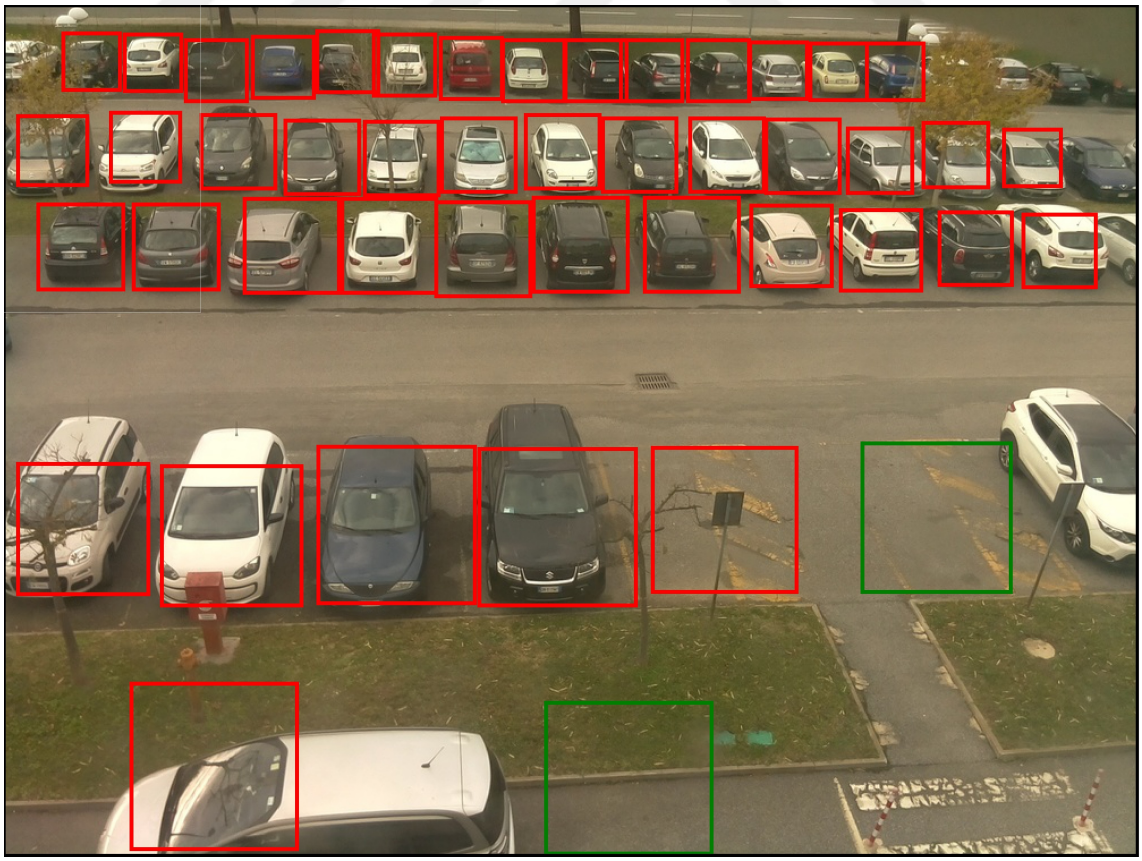
Ancak Şekil 4.4 ve Şekil 4.5'te önden ikinci sırada direğin olduğu park yeri boş olduğu halde dolu olarak işaretlenmiş, Şekil 4.6'da ise bu park yeri boş olarak doğru bir değerlendirme sonucu vermiştir. Bu gibi sonuçlar, modelin eğitimi sırasında yanlış bir özelliği öğrenmiş olmasından kaynaklanıyor olabilir. Bunun sebebi de veri kümesinde yeterli direk içeren görüntü olmaması olabilir. İlerleyen çalışmalarda daha fazla verisetiyle eğitim yapılabilir. Benzeri bir sonuç Şekil 4.8'deki ağaç olan yerde de mevcuttur. Ağacın özellikleri, rengi ya da köşeleri modeli yanıltmış olabilir. Bahsedilen ağacın arkasına park etmiş bir araç dahi olsa görüntüde büyük alanı kaplayan şey ağaç olduğundan dolayı, modelin buradaki ayrımı yapmakta zorlanması şaşırtıcı değildir. Ancak model gelişime açıktır ve bu da ilerleyen çalışmalara bırakılmıştır.



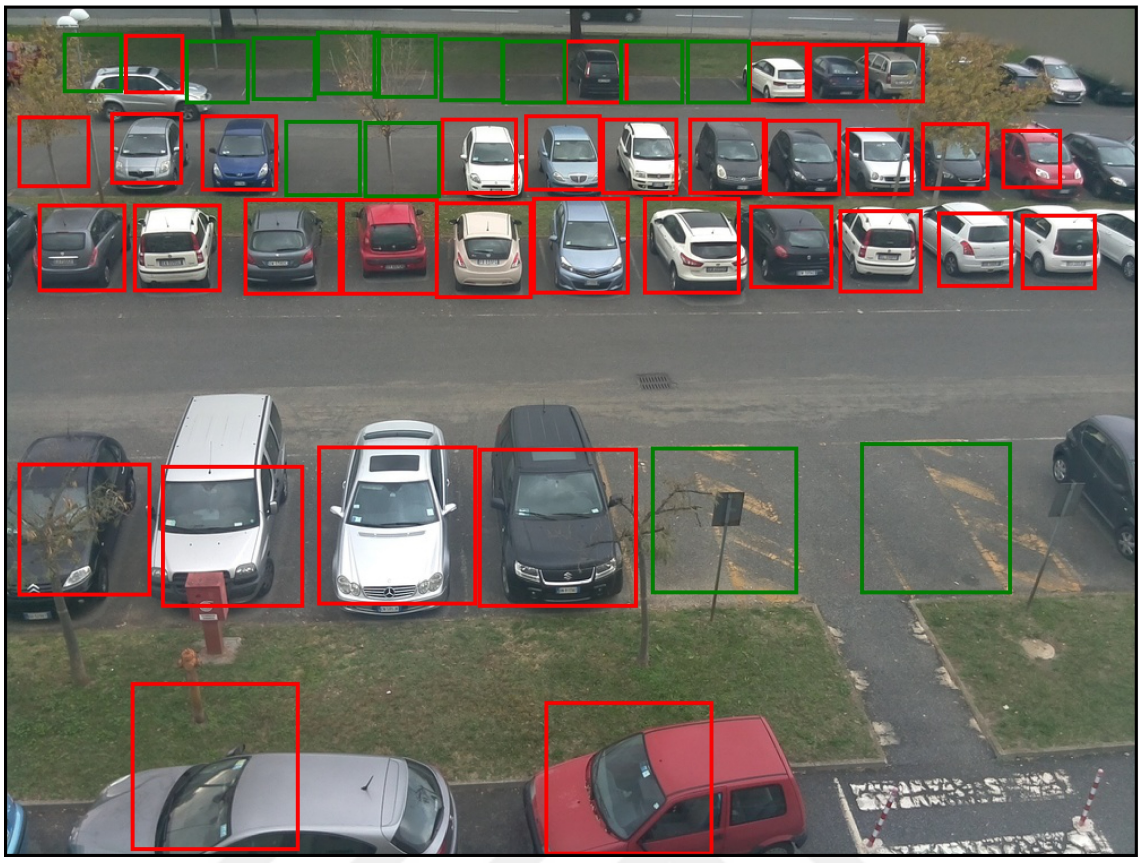
Şekil 4.3 Test görüntüsü örnek 1



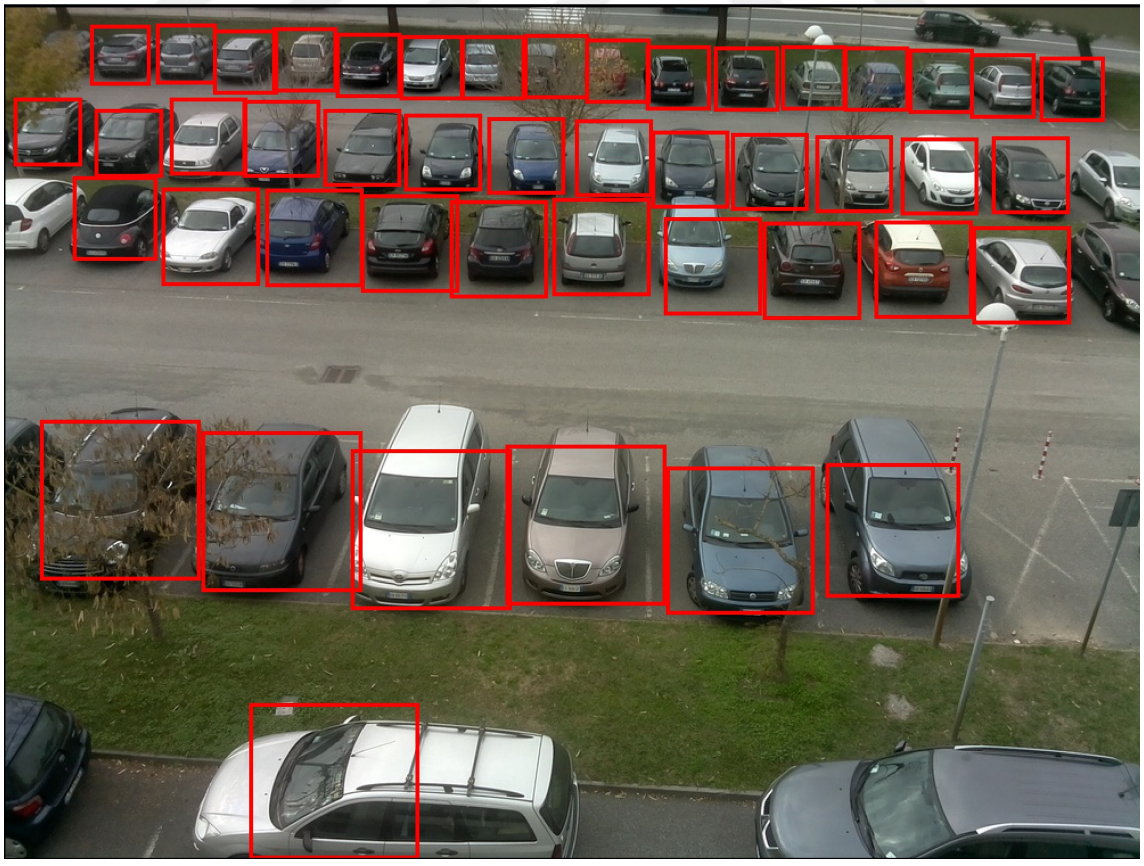
Şekil 4.4 Test görüntüsü örnek 2



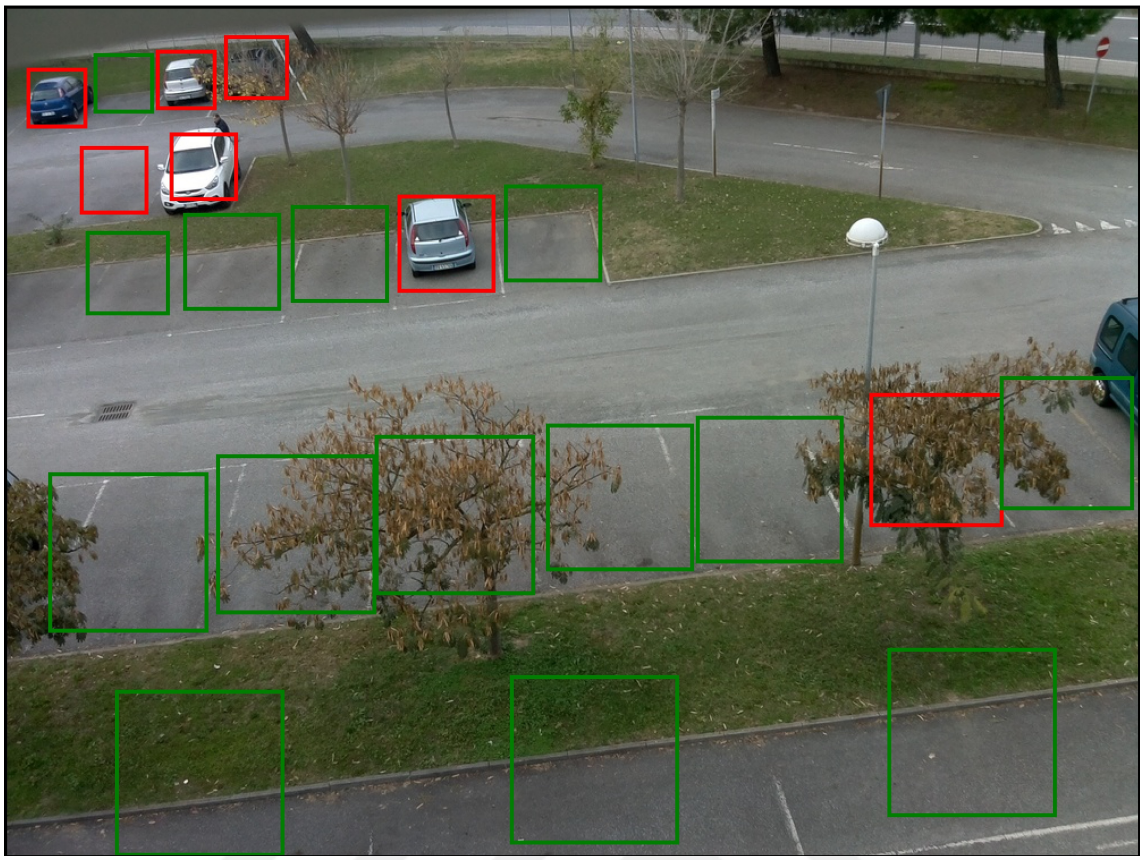
Şekil 4.5 Test görüntüsü örnek 3



Şekil 4.6 Test görüntüsü örnek 4



Şekil 4.7 Test görüntüsü örnek 5



Şekil 4.8 Test görüntüsü örnek 6

Uygulama sonuçlarından da görülebileceği gibi literatürle karşılaştırıldığında daha küçük eğitim seti kullanıldığı ve daha az eğitim adımı uygulandığı halde doğruluk oranları elde edilmiştir. Karşılaştırma sonuçları Tablo 4.2'de görülebilir.

Tablo 4.2 Uygulanan yöntemin literatürle karşılaştırması

Literatürde kullanılmış ağlar	**Eğitim adım sayısı**		**Kullanılan veri kümesi büyüklüğü**		**Doğruluk oranı**	
	CNRPark	**PKLot**	**CNRPark**	**PKLot**	**CNRPark**	**PKLot**
LeNet	30	30	12.584	695.899	%99,3	-
AlexNet	30	30	12.584	695.899	%99,6	%98,1
VGGNet	-	3000	-	695,899	-	%99,9
GoogleNet	500	500	12.584	63.562	%99.7	%99,8
MobileNet	500	500	12.584	63.562	%99,8	%99,9
Resnet	**30**	**30**	**15.824**	**24.444**	**%99,52**	**%99,82**

5
SONUÇ VE ÖNERİLER

Günlük yaşantıda hemen hemen her gün karşılaşılan bir sorun olan park yeri sorununun açık otopark yerlerine yerleştirilecek kameralarla çözülmesine yönelik bir çalışma yapılmıştır.Bu çalışmada bugüne kadar bu konuda derin öğrenme yöntemini kullanan çalışmalardan farklı olarak ResNet modeliyle eğitim yapılmıştır. ResNet modelini kullanarak önceki çalışmalarda elde edilen doğruluk oranı yakalanmıştır. Derin evrişimsel sinir ağlarında, ağ derinliği arttıkça, doğruluk doygunluğa ulaşır ve sonra hızla düşer. Bu model, ağ derinleştikçe artan öğrenme hatasını optimize ederek daha hızlı eğitilir. Bu da diğer modellere göre avantaj sağlar.

Gelecekte planlanan çalışmalarda gerçek zamanlı video görüntülerinden boş park yerlerini kontrol ederek sürücülere bildiren bir mobil uygulama düşünülmektedir.

KAYNAKÇA

[1] V. Paidi, H. Fleyeh, J. Håkansson, R. G. Nyberg, "Smart parking sensors, technologies and applications for open parking lots: A review," *IET Intelligent Transport Systems*, vol. 12, no. 8, pp. 735–741, 2018.

[2] J. Yang, J. Portilla, T. Riesgo, "Smart parking service based on wireless sensor networks," in *IECON 2012-38th Annual Conference on IEEE Industrial Electronics Society*, IEEE, 2012, pp. 6029–6034.

[3] H. Ichihashi, T. Katada, M. Fujiyoshi, A. Notsu, K. Honda, "Improvement in the performance of camera based vehicle detector for parking lot," in *International Conference on Fuzzy Systems*, IEEE, 2010, pp. 1–7.

[4] M. Tschentscher, C. Koch, M. König, J. Salmen, M. Schlipsing, "Scalable real-time parking lot classification: An evaluation of image features and supervised learning algorithms," in *2015 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2015, pp. 1–8.

[5] M. O. Kabak, O. Turgut, "Parking spot detection from aerial images," *Stanford University, Final Project Autumn 2010, Machine Learning class*, 2010.

[6] S. U. Raj, M. V. Manikanta, P. S. S. Harsitha, M. J. Leo, "Vacant parking lot detection system using random forest classification," in *2019 3rd International Conference on Computing Methodologies and Communication (ICCMC)*, IEEE, 2019, pp. 454–458.

[7] H. T. Vu, C.-C. Huang, "A multi-task convolutional neural network with spatial transform for parking space detection," in *2017 IEEE International Conference on Image Processing (ICIP)*, IEEE, 2017, pp. 1762–1766.

[8] G. Amato, F. Carrara, F. Falchi, C. Gennaro, C. Vairo, "Car parking occupancy detection using smart camera networks and deep learning," in *2016 IEEE Symposium on Computers and Communication (ISCC)*, IEEE, 2016, pp. 1212–1217.

[9] D. D. Mauro, M. Moltisanti, G. Patane, S. Battiato, G. M. Farinella, "Park smart," in *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, IEEE, Aug. 2017.

[10] M. Karakaya, F. C. Akıncı, "Parking space occupancy detection using deep learning methods," in *2018 26th Signal Processing and Communications Applications Conference (SIU)*, IEEE, 2018, pp. 1–4.

[11] S. Valipour, M. Siam, E. Stroulia, M. Jagersand, "Parking-stall vacancy indicator system, based on deep convolutional neural networks," in *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*, IEEE, 2016, pp. 655–660.

[12] C.-K. Ng, S.-N. Cheong, E. Hajimohammadhosseinmemar, W.-J. Yap, "Mobile outdoor parking space detection application," in *2017 IEEE 8th Control and System Graduate Research Colloquium (ICSGRC)*, IEEE, 2017, pp. 81–86.

[13] N. Y. Şimşek, *Derin Öğrenme(Deep Learning) Nedir ve Nasıl Çalışır?* https://medium.com/@nyilmazsimsek/derin-ogrenme-deep-learning-nedir-ve-nasil-calisir-2d7f5850782, 2019.

[14] R. S. Sutton, A. G. Barto, *Reinforcement Learning:An Introduction*. The MIT Press, 2014, 2015.

[15] S. Albawi, T. A. Mohammed, S. Al-Zawi, "Understanding of a convolutional neural network," in *2017 International Conference on Engineering and Technology (ICET)*, IEEE, 2017, pp. 1–6.

[16] T. Guo, J. Dong, H. Li, Y. Gao, "Simple convolutional neural network on image classification," in *2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA)(*, IEEE, 2017, pp. 721–724.

[17] A. Kasagi, T. Tabaru, H. Tamura, "Fast algorithm using summed area tables with unified layer performing convolution and average pooling," in *2017 IEEE 27th International Workshop on Machine Learning for Signal Processing (MLSP)*, IEEE, 2017, pp. 1–6.

[18] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

[19] M. Wistuba, N. Schilling, L. Schmidt-Thieme, "Sequential model-free hyperparameter tuning," in *2015 IEEE international conference on data mining*, IEEE, 2015, pp. 1033–1038.

[20] C. Nicholson, *A Beginner's Guide to Backpropagation in Neural Networks*, https://skymind.ai/wiki/backpropagation, 2019.

[21] B. A. Karakuş, *Derin Sinir Ağları için Aktivasyon Fonksiyonları*, http://buyukveri.firat.edu.tr/2018/04/17/derin-sinir-aglari-icin-aktivasyon-fonksiyonlari/, 2018.

[22] A. A. Mohammed, V. Umaashankar, "Effectiveness of hierarchical softmax in large scale classification tasks," in *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, IEEE, 2018, pp. 1090–1094.

[23] E. Hoffer, I. Hubara, D. Soudry, "Train longer, generalize better: Closing the generalization gap in large batch training of neural networks," in *Advances in Neural Information Processing Systems*, 2017, pp. 1731–1741.

[24] N. Çarkacı, *Derin Öğrenme Uygulamalarında En Sık kullanılan Hiperparametreler*, https://medium.com/deep-learning-turkiye/derin-ogrenme-uygulamalarinda-en-sik-kullanilan-hiper-parametreler-ece8e9125c4, 2018.

[25] A. M. Taqi, A. Awad, F. Al-Azzo, M. Milanova, "The impact of multi-optimizers and data augmentation on tensorflow convolutional neural network performance," in *2018 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, IEEE, 2018, pp. 140–145.

[26] D. P. Kingma, J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[27] G. Suresh, V. Gnanaprakash, R. Santhiya, "Performance analysis of different cnn architecture with different optimisers for plant disease classification," in *2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS)*, IEEE, 2019, pp. 916–921.

[28] M. D. Zeiler, "Adadelta: An adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.

[29] A. Shaf, T. Ali, W. Farooq, S. Javaid, U. Draz, S. Yasin, "Two classes classification using different optimizers in convolutional neural network," in *2018 IEEE 21st International Multi-Topic Conference (INMIC)*, IEEE, 2018, pp. 1–6.

[30] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[31] A. Krizhevsky, I. Sutskever, G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[32] A. Sachan, *Quick complete Tensorflow tutorial to understand and run Alexnet, VGG, Inceptionv3, Resnet and squeezeNet networks*, `https://cv-tricks.com/tensorflow-tutorial/understanding-alexnet-resnet-squeezenetand-running-on-tensorflow/`, 2017.

[33] K. Simonyan, A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[34] A. Z. Karen Simonyan, *Very Deep ConvNetsfor Large-Scale Image Recognition*, `http://www.robots.ox.ac.uk/~karen/pdf/ILSVRC_2014.pdf`, 2014.

[35] M. ul Hassan, *VGG16 – Convolutional Network for Classification and Detection*, `https://neurohive.io/en/popular-networks/vgg16/`, 2018.

[36] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[37] Intel, *Hands-On AI Part 16: Modern Deep Neural Network Architectures for Image Classification*, `https://software.intel.com/content/www/us/en/develop/articles/hands-on-ai-part-16-modern-deep-neural-network-architectures-for-image-classification.html`, 2017.

[38] K. He, X. Zhang, S. Ren, J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[39] A. Zhang, Z. C. Lipton, M. Li, A. J. Smola, *Dive into Deep Learning*. 2020, `https://d2l.ai`.

[40] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[41] M. Hollemans, *Google's MobileNets on the iPhone*, https://machinethink.net/blog/googles-mobile-net-architecture-on-iphone/, 2017.

[42] P. R. De Almeida, L. S. Oliveira, A. S. Britto Jr, E. J. Silva Jr, A. L. Koerich, "Pklot–a robust dataset for parking lot classification," *Expert Systems with Applications*, vol. 42, no. 11, pp. 4937–4949, 2015.

[43] G. Amato, F. Carrara, F. Falchi, C. Gennaro, C. Meghini, C. Vairo, "Deep learning for decentralized parking lot occupancy detection," *Expert Systems with Applications*, vol. 72, pp. 327–334, 2017.

[44] M. A. Kızrak, *DERİNE DAHA DERİNE: Evrişimli Sinir Ağları*, https://medium.com/@ayyucekizrak/derine-daha-derine-evrisimli-sinir-aglari, 2018.

[45] T. Carneiro, R. V. M. Da Nóbrega, T. Nepomuceno, G.-B. Bian, V. H. C. De Albuquerque, P. P. Reboucas Filho, "Performance analysis of google colaboratory as a tool for accelerating deep learning applications," *IEEE Access*, vol. 6, pp. 61 677–61 685, 2018.

TEZDEN ÜRETİLMİŞ YAYINLAR

İletişim Bilgileri: aysebetulebren@gmail.com

Konferans Bildirisi

1. A.B.Baktir, B.Bolat, "Determing the occupancy of vehicle parking areas by deep learning," in 2nd International Conference on Electrical, Communication and Computer Engineering(ICECCE), IEEE, 2020.