T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

KONTROLLÜ GEÇİŞLER İÇİN DERİN ÖĞRENME TABANLI PLAKA TANIMA SİSTEMİ

YÜKSEK LİSANS TEZİ

İsmail DEMİRCİ

Enstitü Anabilim Dalı	**:**	**ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ**
Tez Danışmanı	**:**	**Dr. Öğr. Gökçen ÇETİNEL**

Haziran 2019

T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

KONTROLLÜ GEÇİŞLER İÇİN DERİN ÖĞRENME TABANLI PLAKA TANIMA SİSTEMİ

YÜKSEK LİSANS TEZİ

İsmail DEMİRCİ

Enstitü Anabilim Dalı : ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ

Bu tez 13.06.2019 tarihinde aşağıdaki jüri tarafından oybirliği ile kabul edilmiştir.

Jüri Başkanı	Üye	Üye
Dr. Öğretim Üyesi	Dr. Öğretim Üyesi	Dr. Öğretim Üyesi
Gökçen ÇETİNEL	Burhan BARAKLI	M. Zahid YILDIZ

BEYAN

Tez içindeki tüm verilerin akademik kurallar çerçevesinde tarafımdan elde edildiğini, görsel ve yazılı tüm bilgi ve sonuçların akademik ve etik kurallara uygun şekilde sunulduğunu, kullanılan verilerde herhangi bir tahrifat yapılmadığını, başkalarının eserlerinden yararlanılması durumunda bilimsel normlara uygun olarak atıfta bulunulduğunu, tezde yer alan verilerin bu üniversite veya başka bir üniversitede herhangi bir tez çalışmasında kullanılmadığını beyan ederim.

İsmail DEMİRCİ

09.05.2019

TEŞEKKÜR

Yüksek lisans eğitimim boyunca değerli bilgi ve deneyimlerinden yararlandığım, her konuda bilgi ve desteğini almaktan çekinmediğim, araştırmanın planlanmasından yazılmasına kadar tüm aşamalarında yardımlarını esirgemeyen, teşvik eden, aynı titizlikte beni yönlendiren değerli danışman hocam Dr. Öğretim Üyesi Gökçen ÇETİNEL'e teşekkürlerimi sunarım.

Projeyi gerçekleştirmem için teşvik eden ve proje datasetini sağlayan Mehmet ÖMERBEYOĞLU'na değerli desteklerinden dolayı teşekkürlerimi sunarım.

Proje boyunca beni teşvik eden Mehmet TAYGUN ve çalışma arkadaşlarıma teşekkürlerimi sunarım.

İÇİNDEKİLER

SİMGELER VE KISALTMALAR LİSTESİ

ALPR	: Otomatik Lisanslı Plaka Tanıma
API	: Uygulama Programlama Arayüzü
CPU	: Merkezi İşlem Ünitesi
CNN	: Konvolüsyonel Sinir Ağları
GPU	: Grafik İşlem Ünitesi
HOG	: Yönlendirilmiş Gradyanların Histogramı
ILSVRC	: ImageNet Büyük Ölçekli Görsel Tanıma Yarışması
MNIST	: Modified National Institu of Standards and Technology
NLP	: Doğal Dil İşleme
OCR	: Optik Karakter Tanıma
RAM	: Rasgele Erişim Belleği
RPN	: Bölge Teklif Ağı
R-CNN	: Bölge Merkezli Konvolüsyonel Sinir Ağı
SVM	: Destek Vektörel Makinesi
TDNN	: Zaman Gecikmeli Sinir Ağı
VOC	: Görsel Nesne Sınıfları

ŞEKİLLER LİSTESİ

TABLOLAR LİSTESİ

ÖZET

Anahtar kelimeler: Görüntü İşleme, Plaka Tanıma Sistemi, Derin Öğrenme, Yapay Sinir Ağları

Otomatik Plaka Tanıma, otoyollar, otoparklar, terminaller, kamu kurum ve kuruluşları gibi birçok yerde yaygın olarak ihtiyaç duyulan sistemlerdir. Bu çalışmada plaka tanıma sistemi gerçekleşmiştir. Plaka tanıma sistemleri üç ana aşamadan oluşmaktadır. Bu aşamalar: plaka yerinin tespiti, karakter bölütleme ve karakter tanıma işlemleridir. Çalışmada plaka ve karakter tespit için Tensorflow Nesne Algılama API'si kullanılmıştır. Bu üç ana işlem şu sırayla gerçekleşmektedir. İlk önce sistem giriş görüntüleri üzerinde plaka tespit işlemi gerçekleşmektedir. Ardından tespit edilen plaka yeniden boyutlandırma ve filtre işlemlerine tabi tutularak karakter tespit işlemine hazır hale getirilmektedir. İkinci aşamada plaka üzerindeki karakterler tespit işlemi gerçekleşmektedir. Üçüncü ve son aşamada tespit edilen karakter tanınma işlemleri gerçekleştirilmiştir.

Sistem çalışma süresi, i5-8250 CPU 1.80 GHz işlemci özelliklerine sahip bir bilgisayarda ortalama 5.25 saniye sürmektedir. Sistem başarımı üç ana aşama ayrı olarak incelendiğinde: plaka tespit işlem başarımı %99.06, karakter tespit başarımı %95.03 ve karakter tanıma başarımı %90.03 doğrulukla gerçekleştirilmiştir. Ancak modüler bazdaki bu başarımlar genel sistem başarımlarında elde edilememiştir. Genel sistem başarımının modüler bazdaki başarımlara oranla düşük olmasının başlıca sebepleri, plaka eğimi ve çevresel faktörlerdir. Gelecek çalışmalarda bu problemlerin giderilerek genel başarımın yükseltilmesi hedeflenmektedir.

DEEP LEARNING BASED LICENSE PLATE RECOGNITION SYSTEM FOR CONTROLLED TRANSITIONS

SUMMARY

Keywords: image processing, license plate recognition system, deep learning, artificial neural networks.

Automatic Lisence Plate Recognition is a widely needed system in many places such as highways, car parks, terminals, public institutions and organizations. In this study, plate recognition system is implemented. Plate recognition systems consist of three main stages. These stages are plate detection, character segmetation and character recognition. The Tensorflow Object Detection API is used for the plate and character detection process in the study. The multilayer convolution neural networks are used for character recognition. These three main operations take place in the following order: First, the plate detection process is performed on the input image of the system. Then, detected plate is subjected to re-sizing and filtering processes and made ready for character recognition. In the second stage, the characters on the plate are detected. In the third and last stage, the recognition process of the detected characters is performed.

The system runtime takes on average 5.25 seconds on a computer with the i5-8250 CPU 1.80 GHz processor. When the three main stages of system performance are analyzed separately: plate detection process performance is 99.06%, character detection performance is 95.03% and character recognition performance is 90.03%. But this modular performances can not been obtained for overall system performance. The main reasons of this performance decrease are plate slope and environmental factors. In the future works, it is aimed to incerase the performance of the system by overcoming this problems.

BÖLÜM 1. GİRİŞ

Bu bölümde otomatik plaka tanıma sistemleri hakkında genel bir bilgi verilerek tezin kapsamı ve katkıları belirtilmiştir. Ayrıca, plaka tanımada performansı düşüren önemli etmenlerden bahsedilmiştir.

1.1. Plaka Tanıma Sistemleri

Otomatik Plaka Tanıma (Automatic License Plate Recognition, ALPR), kurulan donanımlar ve geliştirilen yazılımlar vasıtasıyla araç plakalarını yüksek doğrulukla verimli bir şekilde tanımlamayı amaçlamaktadır. Son yıllarda otoyollar, otoparklar, terminaller, kamu kurum ve kuruluşları gibi birçok yerde yaygın olarak kullanılmaktadır. ALPR sistemleri sayesinde zaman kaybı yaşanmadan ve insan gücü olmadan araçlı giriş-çıkış işlemleri rahatlıkla sürdürülebilmektedir.

Sadece giriş-çıkış işlemleri için değil, başka birçok uygulamada otomatik plaka tanıma oldukça kullanışlıdır. Suça karışmış veya çalıntı araçları bulmak, yıllık ücret ödenip ödenmediğini tespit etmek, trafik kurallarını ihlal eden kişileri belirlemek gibi uygulamalar ALPR sistemlerinin kullanım alanlarına örnek olarak verilebilir.

Tipik bir plaka tanıma sistemi üç ana aşamaya ayrılabilir:

1. Plaka Tespiti: Görüntüde plakanın bulunduğu bölge tespit edilir.
2. Karakter Bölütleme: Plaka üzerindeki karakterleri tespit edilir.
3. Karakter Tanıma: Tespit edilen her bir karakteri tanıma işlemi gerçekleştirilir.

Bu üç adımı gerçekleştirmek için çeşitli görüntü işleme ve makine öğrenme teknikleri kullanılmaktadır. Bu tezde, plakaları otomatik olarak tanımak için derin öğrenme tabanlı bir yaklaşım sunulmuştur.

Derin öğrenme teknikleri, örüntü tanımada uygulanan öznitelik çıkarma adımını kendi mimarisinde gerçekleştirerek karakter tanıma işlemini yüksek doğrulukla yapabilme yeteneğine sahiptir. En güçlü derin öğrenme yöntemi Konvolüsyonel Sinir Ağları (Convolutional Neural Network, CNN)'dır. CNN mimarisi, çok büyük bir temsil kapasitesi ve yüksek öğrenme potansiyeli veren, seyrek bağlantılara ve ağırlık paylaşımına dayanmaktadır. CNN'lerin en büyük dezavantajları yüksek hesaplama maliyetlerinin olması ve eğitim aşamasında çok fazla sayıda veri gerektirmeleridir.

1.2. Genel Bakış

ALPR problemi birçok varyasyon içeren, çözümünde donanımsal ve yazılımsal işlemler gerektiren karmaşık bir problemdir. İlk aşamada görüntüler renkli, siyah beyaz veya kızılötesi kamerayla çekilebilir. Bazı ulusların plakalar için farklı arka plan renklerine veya plaka üzerinde dikkat dağıtıcı desenlere izin veren farklı standartlara sahip olması tanıma işlemini zorlaştırmaktadır. Harflerin sayısı, sayılar ve boşluklar, konumları ve kullanılan yazı tipleri uluslararası farklılık gösteren diğer değişkenlerdir.

Bu tezde, ülkemiz standartlarında kullanılan plakalar için otomatik bir tanıma sistemi tasarlanmıştır. Ülkemizdeki standart bir plaka sırasıyla illeri temsil eden iki adet rakam, maksimum üç adet İngilizce karakterde yazılan harf ve iki-dört arası rakamdan oluşmaktadır. Bir Türkiye plaka örneği Şekil 1.1.'de gösterilmektedir.

Şekil 1.1. Standart bir Türkiye plaka örneği.

Standart bir plakada diziliş XX_YYY_XXX şeklindedir. Burada X harfi, Y rakamı ve "_" işareti ise boşluğu temsil etmektedir. Ancak, römorklar, motosikletler ve diğer bazı özel araçlar için plakalar bu modelden farklıdır. Bazı plakalar kare olmalarından dolayı çift sıradan oluşur. Bu farklılıklar plaka tespit işlemi sırasında doğruluğu etkilemektir. Arka plan rengi, normal araçlarda beyaz olarak kullanılır. Ancak özel araçlarda bu durum değişebilmektedir. Yazı tipi, ülkemizdeki plakalarda genel olarak "arial bold" fontu kullanılmaktadır. Ancak şehirlere göre farklılıklar olabilmektedir.

Otomatik plaka tanıma sistemlerinde karşılaşılan en önemli problemler aşağıda verilmiştir:

1. Konum: Plaka konumu aracın ön ve arka kısmında tampon bölgesinin ortalarına konulmaktadır. Ancak bazı araçlarda, farklı yerlere de plakalar yerleştirilmektedir.
2. Boyut: Araç plakalarının en boy oranı sabittir. Ancak plakanın boyutu kamera uzaklığına veya yakınlaştırmaya bağlı olarak değişmektedir.
3. Eğim: Görüntü çekildiği açıya bağlı olarak plaka görüntüsünde eğim olabilmektedir.
4. Resim kalitesi: Bulanıklık ve düşük kamera kalitesi ile görüntü düşük kalitede olabilir.
5. Adet: Bir görüntüde birden fazla veya az plaka tespit edilebilir.

6. Arka plan: Giriş görüntüsünde plakaya benzer nesnelerin bulunması veya görüntüdeki herhangi bir yerde bulunan metin veya benzeri desenler plaka ile karıştırılabilir.
7. Aydınlatma: Çevre ışıkları, araç ışıkları veya kamera ışıkları tespit işlemini etkileyebilmektedir.
8. Gölge: Plaka yakınındaki nesneler plaka üzerinde gölgeler bırakabilir ve bu sebepler plaka algılamada problem oluşturabilir.
9. Diğer: Plakada vida, çerçeve gibi plaka tanıma sistemlerini yanıltacak malzemeler kullanılabilmektedir. Bu problemlerin önüne geçmek adına cezalar uygulansa da trafikte hala kullanılan araçlar mevcuttur.

Geçmiş yıllarda plaka tanıma sistemleri ile ilgili yapılmış bazı çalışmalar şu şekilde özetlenebilir. El-Adavi, Keshk ve Haragi, otomatik plaka tanıma sistemi çalışmalarını yapay sinir ağına dayalı olarak gerçekleştirmişlerdir. Geri yayılım algoritması kullanarak eğitilmiş olan plaka tanıma sisteminde, plaka tespit başarımı %89, karakterlerinin tanınma başarımı %93 olarak bildirilmiştir [1]. Juntanasub ve Surreerattanan iki sıralı Tayland plakaları üzerinde bir plaka tanıma sistemi çalışması yapmışlardır. Benzerlik ölçümü için çevrimdışı Hausdorff Distance tekniğini kullanmışlardır. Plaka tanıma başarımı %92 olarak bildirilmiştir [2]. Reus ve Kreft, plaka algılama ve karakter tanıma sistemi için sinir ağları tabanlı bir yaklaşımda bulunmuşlar ve başarımlarının klasik plaka tanıma sistemlerine göre üstün olduğunu iddia etmişlerdir. Ancak çalışmada başarım oranı belirtilmemiştir [3]. Sirithinaphong ve Chamnongthai, plaka tanıma sistemi için dört katmanlı geri yayılımlı yapay sinir ağı kullanmışlardır. Plaka bölgesinin tespit başarımını %84.29, karakter tanıma başarımını ise %80 olarak elde etmişlerdir [4]. Park ve ark. plaka algılama işlemi için sinir ağları tabanlı bir model geliştirmişlerdir. Bu modeli iki farklı veri seti üzerinde test etmişlerdir. Başarım oranları iki veri seti için sırasıyla %97,5 ve %99'dur [5]. Kim ve arkadaşları ise plaka tanına sistemi için iki kademeli bir yöntem izlemişlerdir. Plaka tespit işlemi için sinir ağı, karakter tanıma işlemi için Destek Vektör Makinesi (Support Vector Machine, SVM) kullanılmıştır. Plaka tespit işlem başarımı %97.5, karakter tanıma başarımı için %97.2 doğruluk elde edilmiştir [6]. Jianfeng, Shaofa ve Zhibin, Çin araç plakası sistemi üzerinde çalışmışlardır. Doğru

plaka çıkarma işleminde renk analizi yapılmıştır. Başarı oranı olarak %95.7 olarak bildirilmiştir [7]. Broumandia ve Fathy, Fars plaklarını tanımak için sinir ağları kullanmışlardır. Elde edilen doğruluk %95'dir [8]. Ganapathy ve Lui karakter tanımada ileri beslemeli geri yayılım sinir ağlarını tercih etmişlerdir. Başarım oranı benzer şekilde yaklaşık %95 olarak elde edilmiştir [9].

Angara, otomatik plaka tanıma sistemi için derin öğrenme yöntemini kullanmıştır. Bu çalışma, plaka tespit, karakter tespit ve karakter tanıma adımlarından oluşmaktadır. Çalışmada konvolüsyonel sinir ağları (Convolutional Neural Network, CNN) kullanılmıştır. Başarım oranı %81.1 olarak bildirilmiştir [10]. Rafique, Pedrycz ve Jeon, plaka tespit sistemi için bölge tabanlı evrişimli konvolüsyonel sinir ağları (Region Convolutional Neural Network, R-CNN) kullanmışlardır. Çalışmada üç temel işlemden oluşmaktadır. Bunlar, araç görselleri içerin video karelerindeki plakaların algılanması, kısmi plaka algılama, hareketli kamera ve hareketli taşıtlarda plakaların tespiti işlemleridir. Belirtilen problemlerin çözümü için bölgesel CNN (R-CNN), hızlı R-CNN (Fast R-CNN) ve daha hızlı R-CNN (Faster R-CNN) kullanılmıştır. Çalışmada, yapılan test ve karşılaştırmalarda geleneksel yöntemlere göre daha iyi sonuçlar elde edildiği bildirilmiştir [11]. Jorgensen, plaka tanıma sistemi için geliştirmiş olduğu sistemde konvolüsyonel sinir ağları (Convolutional Neural Networks, CNN) kullanmıştır. Bu çalışmada ayrı iki CNN kullanılmıştır. Birincisi plaka tespit işlemini, ikincisi ise karakter bölütleme ve tanıma işlemlerini gerçekleştirmiştir. Mimaride YOLOv2 kullanılmıştır. Bu çalışmanın genel başarımı %97.6 olarak bildirilmiştir. Sunulan yöntem ile plaka tespit işleminde %99.8, karakter tanımada ise %97.8 doğruluk elde edilmiştir [12]. Puarungroj ve Boonsirisumpun yapmış oldukları çalışmada Tayland motosiklet plakalarını ele almışlardır. Yaptıkları çalışmada üç satırdan oluşan plakaların tespit ve tanıma işlemleri için derin öğrenme yönteminlerini kullanmışlardır. Plaka tespit işlem başarımı %96.94, karakter tanıma başarımı %91.76 olarak elde edilmiştir [13].

Bu tezde amaç, derin öğrenme tekniklerini kullanarak gerçek araç görselleri üzerinden otomatik plaka tanıma işlemini gerçekleştirmektir. İşlemin ne kadar süre aldığı ve ne doğrulukta sonuçlar verdiği de tezde araştırılmıştır.

Genel olarak tezin özgün yönleri ve katkıları aşağıda özetlenmiştir.

1. Önerilen sistemin girdileri olan araç görüntüleri gerçek zamanlı çalışan bir sistem üzerinden alınmıştır. Toplamda 1219 araç görseli kullanılmıştır. Kullanılan görüntüler kontrollü geçiş sisteminden alınmış gerçek araç görüntülerinden oluşmaktadır.
2. Önerilen derin öğrenme tabanlı sistemde üç adet evrimsel sinir ağı kullanılmıştır. İlk CNN, görüntüdeki plaka konumunu algılarken, ikinci CNN, plakadaki karakterleri algılamakta ve üçüncü CNN ise karakterleri tanıma işlemini gerçekleştirmektedir.
3. Sistemin performansı hız ve doğruluk açısından değerlendirilmiştir. Sonuçlar önerilen sistemin gelecek vadeden bir sistem olduğunu doğrulamaktadır.

BÖLÜM 2. SİNİR AĞLARI VE DERİN ÖĞRENME

Canlıların davranışlarını inceleyip, matematiksel olarak modelleyip, benzer yapay modellerin üretilmesine sibernetik denir. Eğitilebilir, adaptif ve kendi kendine organize olup öğrenebilen ve değerlendirme yapabilen yapay sinir ağları ile insan beyninin öğrenme yapısı modellenmeye çalışılmaktadır. Aynı insanda olduğu gibi yapay sinir ağları vasıtasıyla makinelerin eğitilmesi, öğrenmesi ve karar vermesi amaçlanmaktadır. Bu bölümde tez çalışmasının daha iyi anlaşılması için genel olarak sinir ağları ve derin öğrenme mimarisinden bahsedilmiştir.

2.1. Derin Öğrenme

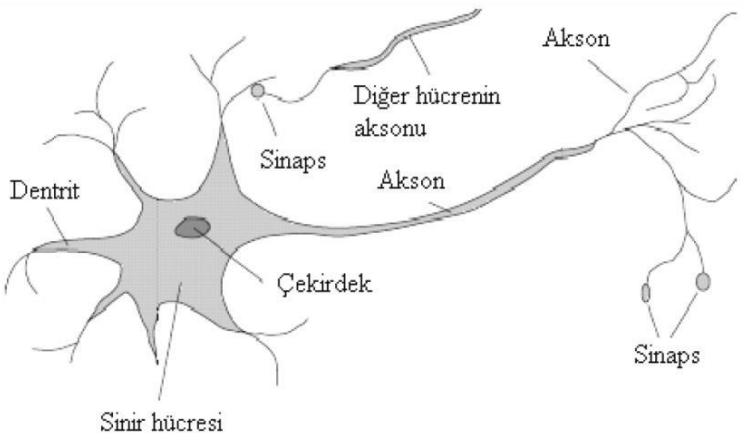
Derin öğrenme, bir veya daha fazla katmanlı yapay sinir ağları ve makine öğrenme algoritmaları kapsayan çalışma alanıdır. Bu yöntemler konuşma tanıma, görsel nesne tanıma, nesne algılama, ilaç keşfi ve genomik gibi birçok alanda çığır açacak gelişmelere olanak sağlamıştır.

Derin öğrenme, bir makinenin her katmandaki gösterimi önceki katmandaki gösterimden hesaplamak için kullanılan dahili parametrelerini nasıl değiştirmesi gerektiğini göstermek için geri yayılma algoritmasını kullanarak karmaşık veri kümelerinde karmaşık yapıyı keşfeder [14].

Derin öğrenme algoritmaları, görüntü, video ve ses işleme konularında sıklıkla kullanılırken, tekrarlayan ağlar metin ve konuşma gibi ardışık veriler üzerine de birçok çalışmaya da olanak sağlamaktadır.

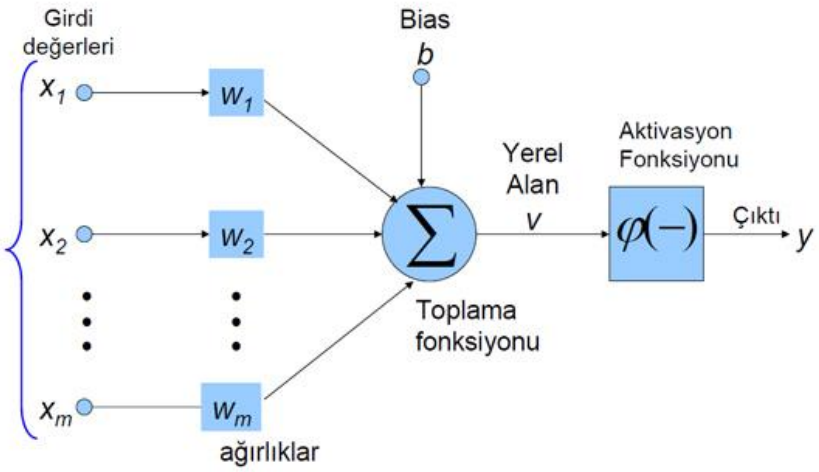
2.1.1. Yapay sinir ağlarına ve nöron yapısına giriş

Yapay sinir ağları, biyolojik sinir ağlarından esinlenerek ortaya atılmıştır. Şekil 2.1.'de bir insanın sinir sisteminde yer alan bir nöron yapısı gösterilmektedir. Şekil 2.2.'de ise matematiksel olarak bir yapay nöron yapısı verilmektedir. Nöronlar uçlarında bulunan dendritler ile sinyalleri alır ve bir çıktı üretir. Ürettiği çıktıyı aksonlar ile sinaps dediğimiz iletişim noktalarına taşıyarak diğer nöronlara iletir.



Şekil 2.1. Biyolojik Nöron Yapısı [15]

Bu şekilde nöronlar arasındaki iletişim elektriksel sinyallerle gerçekleşir. Yapay nöronlar da benzer şekillerde çalışmaktadır. Şekil 2.2.'de yapay nöron yapısı gösterilmiştir. Şekilde X_1W_1 diğer nörondan gelen veriyi temsil etmektedir. Nöron aldığı bu veriyi aktivasyon fonksiyonundan geçirerek elde ettiği çıktıyı diğer nöronlara iletmektedir. Yapay nöronlar biyolojik nöronlardan ilham alınarak tasarlanmıştır.



Şekil 2.2. Yapay Nöron Yapısı [16]

Nöronları birbirine bağlarken aşağıdaki eşitlik kullanılarak (Denklem 2.1) yerel alanlar hesaplanmaktadır.

$$V = \sum_{i=1}^{i=m} X_i . W_i \ + b \tag{2.1}$$

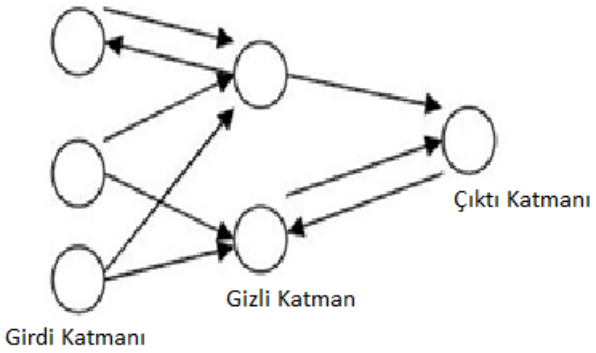
Bu eşitlikte V yerel alan değerini, X giriş değerini, W ağırlık ve b ise yönelim değerlerini temsil etmektedir. X giriş olarak verildiğinde eğitim esnasında model W ve b değerlerini güncelleyerek bir çıktı oluşturmaktadır. Yapay sinir ağlarının çalışmasını kısaca özetleyecek olursak; yukarıda belirtilen Denklem 2.1 kullanılarak verilen giriş için sonuç, aktivasyon fonksiyonundan geçirerek modelin aktif olduğu nöronları bulup tahmin etmesini sağlamaktadır.

Bu model ileri beslemeli yapay sinir ağı modeli olarak bilinir, modelin etkin sonuçlar üretmesi için ağırlık ve yönelim değerlerini güncellemesi gerekmektedir.

2.2. İleri Beslemeli Sinir Ağlar

İleri beslemeli yapay sinir ağları Şekil 2.3.'de gösterildiği gibi temelde 3 katmandan oluşmaktadır. Bunlar, giriş, gizli ve çıkış katmanlarıdır. Giriş katmanı, yapay sinir ağına giren verileri tutan katmandır. Gizli katman veya katmanlar, işlemlerin

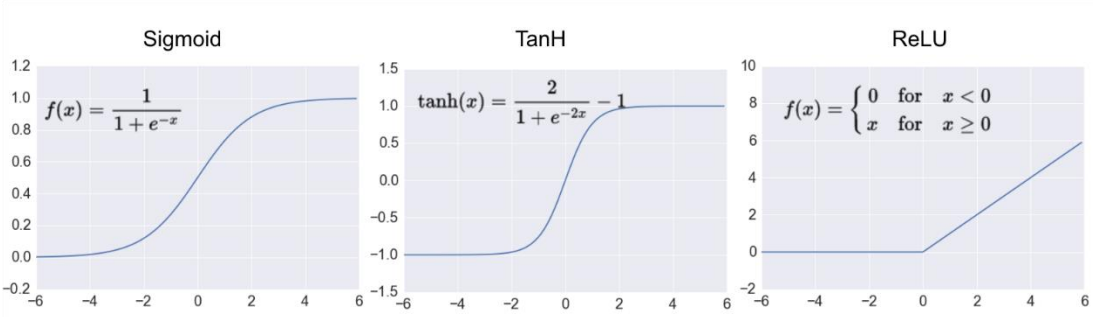
yapıldığı ve istenen sonuca göre ağın kendisi eğittiği katmanlardır. Çıkış katmanı ise eğitilen değerin gösterildiği katmanlardır.



Şekil 2.3. İleri Beslemeli Sinir Ağları [17]

2.2.1. Aktivasyon fonksiyonları

Yapay sinir ağlarını eğitirken aktivasyon fonksiyonları kullanılmaktadır. Aktivasyon fonksiyonunun amacı ağırlık ve yönelim değerlerini ayarlamaktır. Şekil 2.2.'de gösterilen yerel alan değeri bulunduktan sonra bu değer aktivasyon fonksiyonundan geçirilmektedir. Aktivasyon fonksiyonları doğrusal olmayan fonksiyonlardır. Sık kullanılan aktivasyon fonksiyonları Şekil 2.4.'de gösterilmektedir.



Şekil 2.4. Aktivasyon Fonksiyonları [12]

Sigmoid fonksiyonlar aşağıda (Denklem 2.2) verilen eşitlik ile temsil edilir.

$$f(x) = \frac{1}{1+ e^{-x}} \tag{2.2}$$

Görüldüğü üzere sigmoid fonksiyonu giriş değerlerini çıkışa 0 ile 1 arasında bir değer olarak vermektedir. Fonksiyon giriş değeri büyüdükçe 1'e yakın değerler verirken giriş değeri küçüldükçe 0'a yakın değerler üretmektedir. Bu aktivasyon fonsiyonun temel sorunu uç noktalarda gradyan kaybı olmasıdır. Bu ise geri yayılım algoritması parametrelerinin değişimi zorlaştırmakta ve sonucunda eğitim zaman maliyetini artırmaktadır.

TanH fonksiyonları aşağıda (Denklem 2.3) verilen eşitlik ile temsil edilmektedir.

$$f(x) = \frac{2}{1+ e^{-2x}} - 1 \qquad (2.3)$$

Görüldüğü üzere tanh fonksiyonu giriş değerlerini çıkışa -1 ile 1 arasında bir değer olarak vermektedir. TanH fonksiyonu genel anlamda sigmoid fonksiyonuyla benzer çalışmaktadır. Ancak TanH aktivasyon fonksiyonu sigmoid aktivasyonuna göre daha iyi çalışmaktadır. Ancak sigmoid fonksiyonunda karşılaşılan problemler burada da görülmektedir.

ReLU, Günümüzün en popüler aktivasyon fonksiyonlarındandır. Bu fonksiyon gelen değerlere negatif mi yoksa pozitif mi diye bakmaktadır. Eğer gelen değer negatif ise değeri sıfır yapmakta, pozitif ise değere sıkıştırma ya da değiştirme uygulanmamaktadır. ReLU aktivasyon fonksiyonu ile Sigmoid ve TanH aktivasyon fonksiyonlarına göre daha iyi sonuçlar elde edilmektedir. Bu aktivasyon fonksiyonunda Sigmoid ve TanH aktivasyon fonksiyonlarında gerçekleşen gradient kaybı gerçekleşmemektedir. Sigmoid ve TanH fonksiyonları karmaşık hesaplamalar yapmaktadır. Bu aktivasyon fonksiyonunda ise yalnızca negatif mi yoksa pozitif mi olduğuna baktığı için hesaplaması Sigmoid ve TanH'e göre çok daha hızlıdır. Ayrıca bu fonksiyonun biyolojik nöronlara daha yakın bir çalışma yapısı vardır.

Biyolojik nöronlar belli bir eşik değerinin altında kalan sinyalleri görmezden gelmektedir. ReLU fonksiyonu da bu durum göz önüne alınarak tasarlanmış ve tasarlandıktan sonra da popülerliği her geçen gün artmıştır. ReLU aktivasyon fonksiyonun en büyük dezavantajı orta noktası sıfır olmadığı için bazı nöronlarda

kayıpların gerçekleşmesidir. Bu dezavantajı ortadan kaldırmak amacıyla Leaky ReLU aktivasyon fonksiyonu tasarlanmıştır. Bu aktivasyon fonksiyonu negatif değerlere çok küçük bir değer vermekte ve bu sayede nöron kayıplarının önüne geçmektedir. Bu aktivasyon fonksiyonu ReLU aktivasyon fonksiyonuna göre daha yavaş çalışmaktadır. Ancak Sigmoid ve TanH aktivasyon fonksiyonlarına göre yine de çok hızlı çalışmaktadır. Bunun dışında "maxout" fonksiyonu ise, ReLU ve Leaky ReLU aktivasyon fonksiyonlarının genelleştirerek kullanıldığı bir aktivasyon fonksiyonudur. Bu aktivasyon fonksiyonunda nöron kayıpları gerçekleşmemektedir. Ancak parametre sayısı iki katına çıktığı için daha yavaş bir çalışma yapısına sahiptir.

2.2.2. Kayıp (loss) fonksiyonları

Hata fonksiyonu makine öğrenimi için önemli parametrelerden biridir. Hata fonksiyonu modelin tahmin ettiği değerin gerçek değerden ne kadar uzak olduğunu hesaplayarak hatanın büyüklüğünü göstermektedir. Model eğitilirken amaç, hata fonksiyonu tarafından hesaplanmış olan değeri sıfır'a yaklaştırmaktır. Kullanılan çeşitli hata fonksiyonlarından bazıları aşağıda belirtilmektedir.

L2 Kayıp fonksiyonu en sık kullanılan kayıp fonksiyonlarından biridir. Fonksiyona ilişkin eşitlik aşağıda (Denklem 2.4) verilmiştir.

$$S = \sum_i (y_i - f(x_i))^2 \tag{2.4}$$

Eşitlikte x_i giriş değerini y_i gerçek değeri $f(x_i)$ fonksiyonu belirtmektedir. Görüldüğü gibi S, y_i ile $f(x_i)$ arasındaki karesel hatayı verir. Bu hata fonksiyonunda kare alma işlemi hatadan uzaklaştıkça cezalandırmanın daha da büyümesini sağlamaktadır.

Cross Entropi kayıp fonksiyonu sinir ağlarının eğitiminde sıkça başvurulan hata fonksiyonlarından biridir**.** Şekil 2.5.'de verilen örnek yapay sinir ağı ele alındığında, yapay nöron çıktısı α ile temsil edilmektedir. Aşağıda (Denklem 2.5) nöron çıktısının eşitliği görülmektedir.

$$\alpha = \sigma(z) \tag{2.5}$$

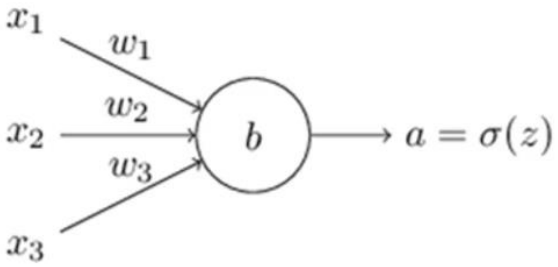
Bu eşitlikte z, sinir ağının çıktısını temsil etmektedir ve aşağıda (Denklem 2.6) verilmiştir.

$$z = \sum_j (Wj * Xj) + b \tag{2.6}$$

Fonksiyon giriş değeri X_j modelin optimize edeceği W_j değeri ile çarpılarak toplanmaktadır. Son olarak yine modelin optimize edeceği b değeri bu toplama eklenmektedir. Şekil 2.5.'de verilen örnek nöron ağındaki sigma ise sigmoid aktivasyon fonksiyonunu temsil etmektedir. Yani z değeri aktivasyon fonksiyonundan geçirilerek 0 ile 1 arasında bir çıktı elde edilmektedir. Şekil 2.5.'de belirtilen örnek yapay sinir ağının çapraz entropi kayıp fonksiyonu denklemi aşağıda (Denklem 2.7) belirtilmiştir.

$$C = -\frac{1}{n} \sum_x [y * ln\alpha + (1-y) * \ln(1-\alpha) \tag{2.7}$$

Burada n, eğitim veri sayısını ve y ise hedeflenen değeri ifade etmektedir. Cross Entropy hata fonsiyonu pozitif ve hedeflenen değere yaklaştıkça sıfıra yaklaşacaktır.



Şekil 2.5. Örnek Yapay Nöron [17]

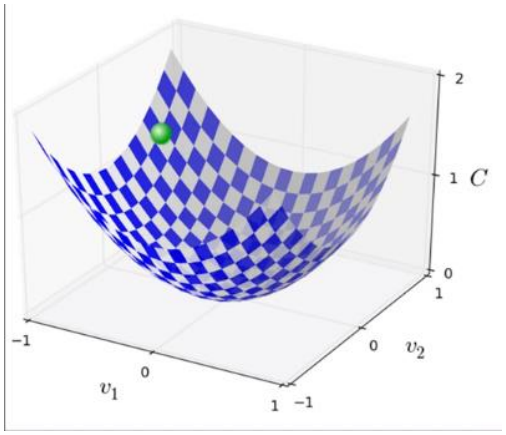
2.2.3. Optimizasyon algoritmaları

Optimizasyon, verilen bir fonksiyonun mümkünse en minimum ya da en maksimumunu değilse bile bu değerlere en yakın değeri bulmayı amaçlar. Burada optimizasyon algoritmalarının kullanılmasında amaç, kayıp fonksiyonunun minimum değerinin bulunmasıdır. Bu amaçla tezde gradyan iniş yönteminden bahsedilmiştir.

2.2.3.1. Gradyan iniş yöntemi

Model bir giriş bilgisi alıp, bu girişi sinir ağından geçirerek sonuç olarak bir çıkış olasılığı verecektir. Modelin vermiş olduğu olasılığın ne kadar doğru olduğu hesaplanıp, modele hata büyüklüğünü bildirir. Bildirilen hata büyüklüğüne göre model ağırlık(W) ve yönelim(b) değerleri güncellenir. Eğitim setine bağlı olarak model bu işlemi defalarca yapmaktadır. Tüm veri setinin yukarıda bahsedilen işlemleri bir defa gerçekleştirmesine epoch denmektedir.

Gradyan iniş, ağırlık değerini değiştirerek kayıp fonksiyonunu en aza indirmeyi amaçlamaktadır. Bu işlem yinelemeli olarak gradyan iniş yöntemi ile yapılır.



Şekil 2.6. Gradyan Alçalış [17]

Gradyan iniş Şekil 2.6.'da verildiği gibi hata değeri değişiminin minimum değere yönelimi için v1 yönünde Δv1 kadar ve v2 yönünde Δv2 kadar hareket ettirilmelidir. Bu işlem aşağıdaki (Denklem 2.8) eşitlik kullanarak gerçekleştirilir.

$$\Delta C \approx \frac{\partial C}{\partial v1}\Delta v1 + \frac{\partial C}{\partial v2}\Delta v2 \quad (2.8)$$

Hata değerinin sıfıra yakınlaştırılması amaçladığı için, ΔC'yi negatif yapacak Δv1 ve Δv2 değerleri seçilmesi gerekmektedir.

2.2.3.2. Stokastik gradyan iniş yöntemi

Parçalı eğitme metoduna stokastik gradyan iniş denmektedir. Yani bilgisayarın performansını artırmak için veriyi parçalara bölerek grafiği beslemek amaçlanmaktadır. Bu algoritma aşağıdaki (Denklem 2.9) gibi ifade edilir.

$$x = x + learning_{rate} * dx \quad (2.9)$$

Eşitlikte (Denklem 2.9) x, parametre vektörünü, dx ise türev işlemi temsil etmektedir. Parçalı eğitme yönteminin karşılaştığı bazı problemler bulunmaktadır. Bunlardan en sık karşılaşılanları, yerel minimum ve semer nokta problemleridir.

Yerel minimum problemi, hata değerinden daha fazla gradyan alçalışı yapamadığı için bulunduğu konumu en düşük hata değeri olarak algılama problemidir. Bu durumda global minimuma ulaşmadan iterasyon sonlandırılır.

Semer nokta problemi ise eğimin sıfır olduğu noktalarda parçalı eğitme yönteminin aşılamadığı problemlerdendir. Bu noktada eğim sıfır olduğu için daha fazla optimize yapılamama problemidir.

Stokastik Gradyan Alçalış optimizasyonuna momentum eklemek bu iki problemin ortadan kalması için yeterlidir. Momentum ile iniş eylemine hız eklenerek optimizasyon işlemi gerçekleşmektedir.

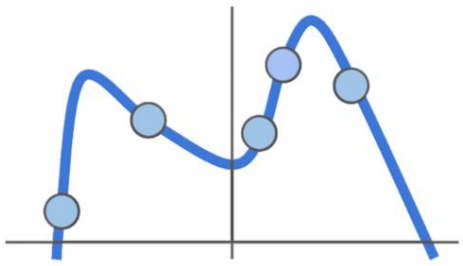
2.2.4. Geri yayılım (backpropagation)

Sinir ağı giriş değerini alıp, ileri yönde ilerleyerek bir çıkış değeri vermektedir. Model bu çıkışın hata değerlerinin ağırlık ve yönelim değerlerini güncelleyerek daha iyi sonuçlar vermesini sağlamaktadır. Modelin bu işlemi gerçekleştirmesinde geri yayılım olarak adlandırılan metodu kullanmaktadır.

Geri yayılım, yapay sinir ağlarında her nöronun hataya ne kadar katkısı olduğunu hesaplamak için kullanılır. Geri yayılım metodu türev işleminin zincir kuralı kullanarak gerçekleştirmektedir. Sinir ağımızdaki her bir nöron yalnızca bağlı bulunduğu düğümler tarafından etkilenmektedir. Her bir düğümün yerel girişi ve çıkışı mevcuttur. Buradan yerel türev hesaplanır ve bu işlem bütün düğümlerde geri yönelimli olarak gerçekleşir.

2.2.5. Aşırı uyum (overfitting) ve düzenlileştirme (regularization)

Aşırı uyum, makine öğreniminde sık karşılaşılan problemlerinden biridir. Bu problem modelin eğitim setindeki verileri tanımada başarılıyken hiç görmediği bir test setinde kötü sonuçlar vermesi olarak düşünülebilir. Problemin ortaya çıkma sebebi modelin veriyi genelleştirememesidir.

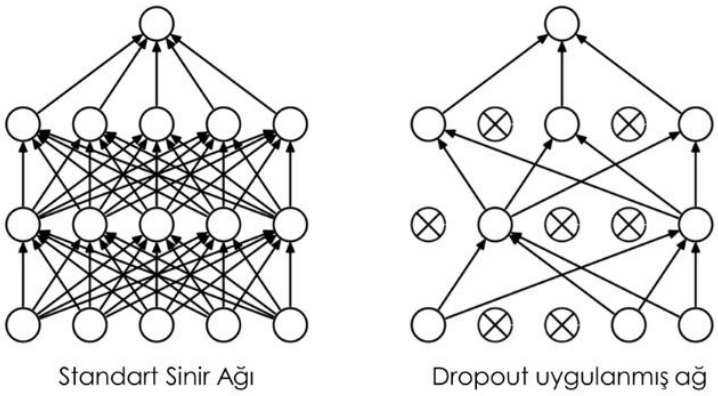


Şekil 2.7. Aşırı Uyum [17]

Şekil 2.7.'de görüldüğü gibi model, eğitim setindeki verileri çok iyi bir şekilde sınıflandırmak için data seti verilerini takip edip, genelleştirme yapamamaktadır. Bu da eğitim setindeki verilerin dışında bir veri ile karlışması durumunda modelin yanlış cevaplar vermesine neden olmaktadır.

Aşırı uyum problemini çözmek için kullanılan yöntemlerden bazıları aşağıdaki gibidir:

1. Veri setini genişletmek: Eğitim kümesinin mümkün olduğunca genelleştirilmesidir.
2. Veri artırma: Eğitim esnasında veri setinin değiştirilerek modelin daha iyi genelleştirme yapmasının sağlanmasıdır. Bu yöntem ile very sayısı yapay yollarla artırılır.
3. Çıkarma: Bu yöntemde sinir ağının her ileri gidişinde katmandaki bazı nöronlar devre dışı bıralır. Devre dışı bırakma işlemi rastgele yapılır. Eğitim bittiğinde ise devre dışı bırakılan böronlar aktifleştirilerek test işlemi yapılır. İşlem Şekil 2.8.'de gösterilmiştir.

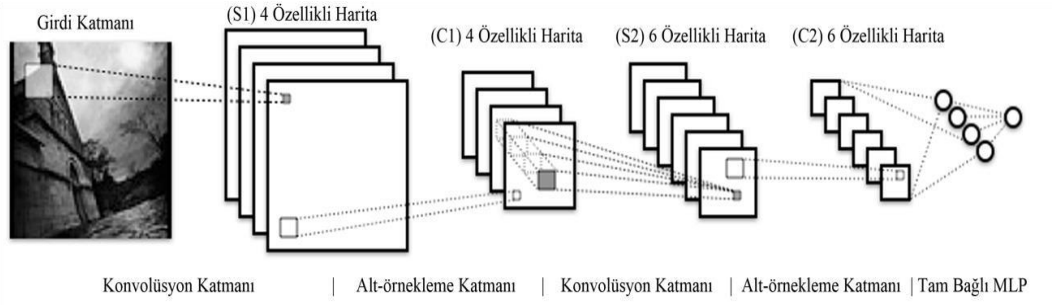


Şekil 2.8. Çıkarma [18]

2.3. Konvolüsyonel Sinir Ağları

Konvolüsyonel sinir ağları (CNN) biyolojik sinir ağlarından ilham alınarak LeCun [19] tarafından önerilen bir sinir ağıdır. Klasik bir yapay sinir ağı modelinde nöronlar ve katmanlar arasındaki bağlantılar ve öğretilen parametreler hesaplama zorluklarını ortadan kaldırmak için önerilen sinir ağıdır.

CNN, bir veya daha fazla konvolüsyonel katman, altörnekleme (subsampling) katmanı ve bunun ardından standart çok katmanlı bir sinir ağı gibi bir veya daha fazla tamamen bağlı katmandan oluşur.



Şekil 2.9. LeNet Ağı Mimarisi [19]

Şekil 2.9.'da verilen ilk CNN ağı 1988 yılında Yann LeCun tarafından ortaya atılan, 1998'lere kadar iyileştirmeleri devam eden LeNet isimli mimaridir [19]. LeNet ağında, alt katmanlar art arda yerleştirilmiş konvolüsyon ve maksimum havuzlama katmanlarından oluşur.

CNN algoritmaları görüntü ve ses işleme alanı başta olmak üzere doğal dil işleme (Natural Language Processing, NLP), biyomedikal gibi birçok farklı alanda uygulanmaktadır. Özellikle görüntü işleme alanında görülmüş en iyi (state of the art) sonuçlar CNN ile elde edilmiştir. Modifiye Ulusal Standartlar ve Teknoloji Enstitüsü (Modified National Institute of Standards and Technology, MNIST) veri kümesi üzerinde, Cireşan yaptığı çalışmada, CNN ile hata oranını %2'ye kadar düşürmeyi başarmıştır [20]. Yine Cireşan ve arkadaşları tarafından, MNIST ve New York University (NYU) Nesne Tanıma Karşılaştırma (NYU Object Recognition Benchmark, NORB) veri kümeleri üzerinde denenen başka bir çalışmada CNN ile, öğrenme sürecinin çok hızlı gerçekleştiği ve o zamana kadar önerilen yöntemlerin en başarılısı olduğu belirtilmiştir [21]. 2014 yılında, ImageNet Yarışması'nda, milyonlarca resim ve yüzlerce nesne sınıfı ile nesne sınıflandırması ve algılaması dalında en başarılı dereceleri alan ekiplerin hepsi CNN algoritmalarının modifikasyonlarını kullanmışlardır. 2015'te yapılan bir çalışmada CNN, ters yüzler de dahil olmak üzere geniş açı aralıklarındaki yüzleri yakalayabilme başarısını

göstermiştir. Bu ağ, çeşitli açılar ve yönlerde yüzleri içeren 200.000 görüntü ve yüzleri olmayan 20 milyon görüntü daha içeren bir veri tabanı üzerinde eğitilmiştir.

Aşağıda CNN mimarisine ait bileşenler kısaca açıklanmıştır.

2.3.1. Konvolüsyonel katman (convolutional layer)

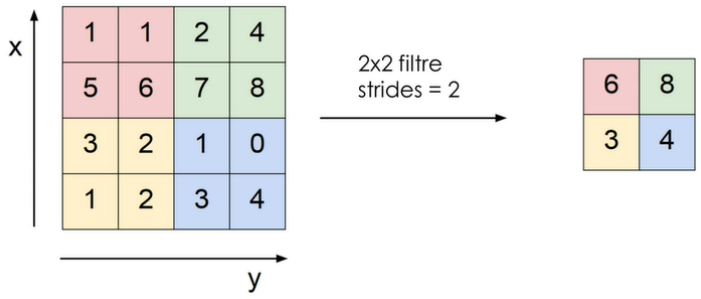
Konvolüsyonel katman bir CNN'nin çekirdek yapı bloğudur. Katmanın parametreleri, küçük bir alıcı alana sahip olan ancak girdi hacminin tam derinliği boyunca uzanan bir dizi öğrenilebilir filtre (veya çekirdek) içerir.

Bir konvolüsyonel katmanındaki girdi, m x m x r 'lik bir resimdir. Burada m değerleri sırasıyla görüntünün yüksekliği ve genişliği, r ise kanalların sayısıdır. Konvolüsyonel katman n x n x q boyutlarında k adet filtreden oluşur. Filtre için görüntünün boyutundan daha küçük olan n, ve r (kanal) değeriyle genellikle aynı seçilen q değeri seçilir. Bu filtreler ile, her biri m-n+1 m+n+1 boyutunda k adet birbirine yerel olarak bağlı aktivasyon haritaları üretilir. Sonuç olarak, ağ, girişte belirli bir özellik türünde belirli bir özellik türü algıladığında etkinleşen filtreleri öğrenir. Derinlik boyutu boyunca tüm filtreler için aktivasyon haritalarının depolanması, konvolüsyonel katmanının tam çıkış hacmini oluşturur. Böylece çıkış hacmindeki her giriş, girdideki küçük bir bölgeye bakan ve parametreleri aynı aktivasyon haritasında nöronlarla paylaşan bir nöronun çıktısı olarak da yorumlanabilir.

2.3.2. Havuzlama katmanı (pooling layer)

Havuzlama katmanın amacı modeli küçültmek ve parametre sayısını azaltmaktır. Genellikle havuzlama işlemi ile katmanların sayısı yarıya indirilir. En popüler havuzlama yöntemleri maksimum ve ortalama havuzlama yöntemleridir. Bu havuzlama yöntemlerinden en sık kullanılan havuzlama yöntemi maksimum havuzlama yöntemidir.

Şekil 2.10.'da belirtilen havuzlama Yöntemi, geçerli matrisin maksimum değerini seçer. Bu havuzlama yöntemi, algılanan özellikleri korumaktadır.



Şekil 2.10. Maksimum Havuzlama [22]

Havuzlama katmanı, girişin her derinlik diliminden bağımsız olarak çalışır ve uzamsal olarak yeniden boyutlandırılmaktadır. En yaygın biçim, aktivasyonların %75'ini atmak suretiyle girişteki her derinlik diliminde 2 aşağı örnek ile 2 genişlikteki bir adımla uygulanan 2 x 2 boyutunda filtreler içeren bir havuz katmanıdır. Bu durumda, her maksimum işlem 4 sayının üzerindedir. Derinlik boyutu değişmeden kalmaktadır.

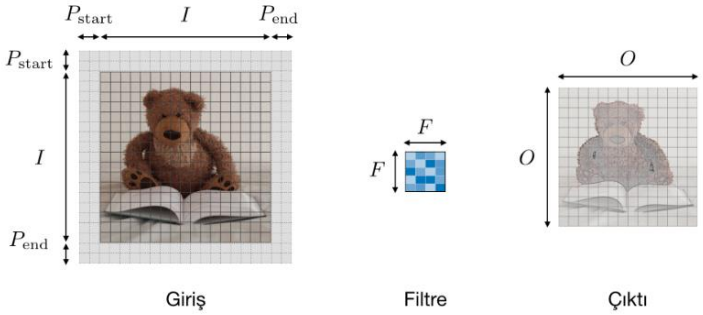
Ortalamalı havuzlama sisteminde, her havuzlama işlemi, geçerli matrisin değerinin ortalamasını almaktadır.

2.3.3. Tam bağlantılı katmanı (fully connected layer)

Tam bağlantı katmanı, her girişin tüm nöronlara bağlı olduğu bir giriş üzerinde çalışır. Tam bağlantı katmanları genellikle CNN mimarisinin sonunda bulunmaktadır. Tam bağlantı katmanları amaç fonksiyonunu optimize etmek için kullanılır.

2.3.4. Hiper parametrelerin filtrelenmesi

Konvolüsyonel katmanı, hiperparametrelerinin ardındaki anlamı bilmenin önemli olduğu filtreler içerir. Şekil 2.11.'da hiperparametrelerin filtreleme işlem yapısı gösterilmektedir.



Şekil 2.11. Hiperparametre Filtrelenmesi [22]

Eşitlikte (Denklem 2.10) aktivasyon haritasının çıkış büyüklüğü ifade edilmektedir.

$$O = \frac{I-F+Pstart+Pend}{S} + 1 \tag{2.10}$$

Burada I, girdinin hacim büyüklüğünü, F filtre uzunluğunu, P sıfır ekleme miktarını, S adım aralığını, O ise öz nitelik haritasının çıkış büyüklüğünü belirtmektedir. Adım aralığı (S), her işlemden sonra pencerenin hareket ettiği piksel sayısını belirtir.

Sıfır Ekleme/Doldurma, girişin sınırlarının her bir tarafına P adet sıfır ekleme işlemini belirtir. Bu değer el ile eklenebilir veya 3 ayrı modda otomatik olarak ayarlanabilir. Geçerli modda, P = 0 yani sıfır eklemesi yapılmaz. Boyut uyuşmuyor ise kalan piksel için konvolüsyon gerçekleşmez. Aynı modda, öznitelik harita büyüklüğüne sahip ekleme yapılır [I/S]. Çıktı boyutu matematiksel olarak uygundur. Yarım ekleme olarak da isimlendirilir. Tüm modda ise son konvolüsyonellerin giriş sınırlarına uygulandığı maksimum ekleme yöntemidir. Filtre girişi uçtan uca görür.

Bir filtrenin boyutları, C kanalları içeren bir girişe uygulanan F x F boyutunda bir filtre, I x I x C boyutundaki bir girişte konvolüsyon gerçekleştiren ve aynı zamanda bir çıkış özniteliği haritası üreten F aktivitesi (aktivasyon olarak da adlandırılır) O x O x 1 boyutunda olan bir harita çıkartır. Sonuç olarak çıkışımızın derinliği kullanılan filtre sayısı kadar olacaktır.

Konvolüsyonel sinir ağlarında her bir katmanda matris üzerinde işlem yapan çekirdekler kullanılmaktadır. Bunlar basit Gabor filtreleri gibidir. Bu çekirdeklerin boyutu öğrenme üzerinde çok etkilidir. Çünkü çekirdek boyutu ile ne kadar genişlikte verinin birbirini etkileyeceğine karar verilmektedir. Genelde 3x3, 5x5, 7x7 çekirdekler kullanılmaktadır. Büyük boyutlu çekirdek kullanılması konvolüsyon uygulandıktan sonra oluşacak resmin küçük olmasına neden olacaktır. Bu durumun bilgi kaybına yol açmasından dolayı genelde 3x3 gibi küçük boyutlu çekirdek kullanılmaktadır. Kenar bulma işleminde işlem yapılan pikselin sağına-soluna, üstüne-altına bakılabilmesi için genelde tek sayılardan oluşan merkezi olabilecek filtreler kullanılır.

BÖLÜM 3. MATERYAL VE YÖNTEM

Otomatik plaka tanıma sistemi genel olarak üç temel adımdan oluşmaktadır. Bunlar; plak tespiti karakter segmentasyonu ve karakter tanımadır. Bu bölümde bu üç temel adımdan ve literatürde yaygın olarak tercih edilen yaklaşımdan bahsedilmiştir.

3.1. Plaka Tespiti

Plaka tespit adımı, sonraki adımların doğruluğunu büyük ölçüde etkilemektedir. Giriş, bir veya birden fazla plaka içeren bir görüntüdür. Çıktı ise görüntünün yalnızca plakaları içeren bölümü olmalıdır. Bu zorlu problemi çözmek için son yıllarda birçok algoritma önerilmiştir. Ancak giriş görüntüsündeki araç rasgele bakış açısı, araç plaka konumlandırma, ortam aydınlatma koşulları gibi birçok faktör plaka tespit işlemi halen daha zorlaştırmaktadır.

Bir giriş görüntüsündeki her bir pikseli işleme tutup, bu işlemler ile plakanın tespitini gerçekleştirme yöntemi yüksek maliyet gerektirmektedir. Bu yöntem yerine kullanılan ortak yöntem plakada öne çıkan özellikleri baz alarak plaka tespit adımı gerçekleştirmek süre açısından önemli kayıpları önlemiştir. Mevcut algoritmalar plaka tespit yaklaşımlarına göre katagorize edilmiştir. Burada sunuculacak beş ana yaklaşım şunlardır: kenar tabanlı, renk bazlı, karakter bazlı ve iki veya daha fazla yaklaşım kombinasyonlarıdır. Her yaklaşım avantaj ve dezavantajlarının özeri: Tablo 3.1.' de verilmektedir.

3.1.1. Kenar tabanlı yaklaşımlar

Kenar tabanlı yaklaşımlar, plakaların bilinen en boy oranıyla dikdörtgen olması gerçeğinden yararlanarak plakaları algılar. Plaka ile arka plan arasındaki renk

geçişini kullanarak, plakanın sınırı bulunabilir. [23] ve [24] 'de, potansiyel kenarları tespit etmek için bir Sobel filtresi kullanılmıştır. Kenar yoğunluğu yüksek olan bölgeler potansiyel araç plakası olarak görülür ve aday bölgeleri önermek için hem yatay hem de dikey kenar algılama gerçekleştirebilmektedir.

Tablo 3.1. Plaka Tespit Yaklaşımları Karşılaştırması [12]

Yaklaşım	Avantajları	Dezavantajları
Kenar tabanlı	Basit ve hızlı	Görüntünün kenarları çok olduğunda, çok fazla aday döndürür ve görüntü bulanık olduğunda sorun çıkarır.
Renk tabanlı	Eğimli veya deforme olmuş plakalardan etkilenmez	Yalnız kullanıldığında sınırlı sonuç.
Doku tabanlı	Düzenli şekillere sahip plakalara bağlı değil	Giriş görüntüsünün birçok kenarı veya farklı aydınlatması olduğunda hesaplama açısından karmaşıktır.
Karakter tabanlı	Yüksek geri çağırma hızı ve dönüşe dayanıklı	Resimde başka metinler bulunduğunda sorun var.
İki veya daha fazla özelliği birleştirmek	Çok sağlam ve etkili	Çok hesaplamalı olarak karmaşık.

Zheng ve ark. [23] sadece dikey kenar algılaması kullanmaktadır, çünkü yatay kenar algılaması daha yüksek bir hata vermiştir. Araç plakalarının yatay kenarları araç tamponunun yatay kenarlarıyla kolayca karıştırılabilir [25]. Ayrıca, yalnızca en boy oranına sahip dikdörtgen plakalar ile aynı adayları yoksaydığı ve genel sonucu iyileştirdiği düşünülmektedir. Rapor edilen tespit oranı %99.7'dir. Hough dönüşümü, bir görüntüdeki plakaları düz çizgiler ile tespit etmek için de kullanılabilir [26]. Hough dönüşümünün avantajı, 30 derece eğime kadar düz çizgileri tespit edebilmesidir. Ancak, bu zaman ve hafıza tüketen bir süreçtir.

Genel olarak, kenar tabanlı yöntemler plaka tespitini gerçekleştirmenin en basit ve en hızlı yoludur. Ancak, giriş görüntüsü karmaşık olduğunda sorunla karşılaşırlar. Giriş

görüntüsü çok fazla kenar içeriyorsa çok fazla aday döndürebilir veya görüntü bulanıksa plakayı bulmakta sorun yaşayabilmektedir.

3.1.2. Renk tabanlı yaklaşımlar

Renk tabanlı yaklaşımlar, plakaları tespit etmek için görüntüdeki renkleri kullanmaktadır. Plaka genellikle arka plana göre farklı bir renge sahiptir ve renk farkı plakayı bulmak için kullanılabilmektedir. Plakanın ve karakterlerinin renk kombinasyonu, plaka bölgesinin dışında başka yerlerde nadiren bulunmaktadır. [27] 'de, bu renk kombinasyonlarının nasıl tanınacağını öğrenmek için genetik bir algoritma kullanılmıştır. Farklı aydınlatma koşullarında eğitim yapılarak algoritmanın hangi renkleri araması gerektiğini öğretilir. Çalışma her bölgenin ortalama parlaklığı ile plaka bölgelerini diğer bölgelerden ayırt edebilmektedir. Çalışma sonucunda %92.8 doğruluk elde edilmiştir. Ortalama kayma algoritması [28] 'de farklı renklere göre bölgeleri bölümlere ayırmak için kullanılmıştır. Bir aday bölgedeki plakayı bulmak için, kenar tabanlı yöntemler kullanılmıştır. Belirtilen plaka tespit oranı %97.6'dır. Yürütme süresi belirtilmemiştir. Renk bazlı yöntemler sıklıkla diğer yaklaşımlarla birlikte kullanılır, ancak tek başlarına da kullanılabilir. Eğimli veya deforme olmuş plakalardan etkilenmezler. Bununla birlikte, sadece aydınlatma koşulunu göz önüne aldıklarından, tek başına kullanıldığında renk bazlı yöntemlerin sınırlı sonuçları olabilmektedir.

3.1.3. Doku tabanlı yaklaşımlar

Doku tabanlı yaklaşımlar, bir görüntüdeki plakaları algılamak için plaka içindeki geleneksel piksel yoğunluğu dağılımını kullanır. Plakanın üzerindeki karakterler plakanın geri kalanını karşılaştırır ve farklı doku bazlı metotlarla tespit edilebilir.
[29]'da kayan eşmerkezli pencereler kullanan bir yöntem önerilmiştir. Plakalar görüntünün dokusunda düzensizlik olarak görülmektedir. Dokudaki ani değişiklikler potansiyel plakaları tespit etmek için kullanılır.

Kaushik ve ark. [30], sürgülü eşmerkezli pencereleri ve histogramları birleştirerek doğruluğunu artıran bir yöntem önermektedir. [31]'de hem genel istatistiksel özellikleri hem de yerel Haar benzeri özellikleri kullanan bir yöntem önerilmiştir. Global özellikler arka plan alanlarının çoğunu hariç tutarken, yerel Haar benzeri özellikler yöntemi parlaklık, renk, boyut ve plakaların pozisyonundaki değişikliklere karşı dayanıklı kılmaktadır.

Doku-temelli yaklaşımlar, düzenli şekillere sahip plakalara bağlı olmadığından ve plakaların sınırları deforme olduğunda bile işe yaradığından, renk ve kenar özelliklerinden daha fazlasını tespit edebilirler. Dokubazlı yöntemlerin dezavantajı, hesaplamalı olarak karmaşık olmalarıdır. Giriş görüntüsünün birçok kenarı veya farklı aydınlatması varsa, işlem süresi yüksek olabilmektedir.

3.1.4. Karakter tabanlı yaklaşımlar

Karakter tabanlı yaklaşımlar, plakaları karakter dizileri olarak işler. Bu yöntemler görüntüyü karakterlerin varlığı doğrultusunda incelemektedir. Karakterlerin bulunduğu bölgeler, bir plaka içeren potansiyel bölgeler olarak önerilmektedir. Görüntüdeki bölgeleri karakter benzeri bölgelere göre sınıflandırmak için bir sinir ağı kullanılır ve karakter benzeri bölgelerin doğrusal kombinasyonları bulunursa bir plaka tespit edilir. Li ve ark. [32], görüntüdeki tüm karakterleri 37-yollu bir CNN kullanarak tespit etmişlerdir ve daha sonra yanlış fazla önermeleri ortadan kaldırmak için yeni bir CNN daha kullanmışlardır. İşlem, görüntü başına yaklaşık 5 saniye olduğundan oldukça yavaştır, ancak karakter bölümlendirme ihtiyacını ortadan kaldırmaktadır.

[33]'de görüntü belirginliğine dayalı bir yöntem kullanılmaktadır. İlk olarak, yöntem yalnızca karakterleri algılar ve yüksek bir geri çağırma hızı veren bir yoğunluk belirginliği haritası kullanarak bunları bölümlere ayırır. Ardından bölümlenmiş karakterlerde kayan bir pencere kullanılır ve ilgili göze çarpan özelliklere göre plakalar tespit edilir.

Karakter tabanlı yaklaşımlar genellikle yüksek bir geri çağırma oranına sahiptir ve ayrıca döndürülmüş plakalara dayanıklıdır. Ancak, bu yaklaşımlar, resimde başka bir metin bulunduğunda yüksek bir hata vermektedir. Yöntemler ayrıca zaman açısından dezavantajlı olabilmektedir.

3.1.5. İki veya daha fazla özelliği birleştiren yaklaşımlar

Uygun olduğunda, birden fazla özelliği birleştirmek plakaları tespit etmenin etkili bir yoludur. Porikli ve ark. [34], plakadan çıkartılan istatistiksel ve uzamsal bilgileri içeren bir kovaryans matrisi kullanmaktadır. Bir sinir ağı, daha sonra kovaryans matrisi ve plaka dışı bölgeler üzerinde eğitim alarak plakalar tespit edilmiştir.

[35]'de iki zaman gecikmeli sinir ağı (time delay neural network, TDNN) kullanıldı. İlk TDNN, plakanın renklerini, ikincisi ise dokuyu analiz eder. İki bölgenin çıktısı, aday bölgeleri bulmak için birleştirilir. Bildirilen tespit oranı, ücretli kapılardan 1000 video dizisi içeren bir veri setinde %100'dür.

İki veya daha fazla özelliği birleştiren yaklaşımlar, genellikle tek bir özellik kullanan yaklaşımlardan daha güvenilir ve etkilidir. Bununla birlikte, yöntemler tek özellikli yaklaşımlara kıyasla daha fazla hesap yükü gerektirdiğinden karmaşıktır.

3.2. Karakter Bölütlendirme

Karakterleri tanımadan önce, karakterlerin bir bölütlendirme işlemine sıklıkla ihtiyaç duyulur. Genellikle plakanın dönmesi veya plaka boyunca parlaklıktaki değişikliklerden kaynaklanan problemleri önlemek için bu işlemden önce ön işlem yapılır. Öncelikle, mevcut algoritmalar iki ana kategoriye uygundur: projeksiyon tabanlı ve piksel bağlantı tabanlı. Her ikisi de giriş görüntüsünün binarize edilmesini gerektirir. Her bir yaklaşımın avantaj ve dezavantajlarının bir özeti Tablo 3.2.'de verilmektedir.

Tablo 3.2. Karakter Bölütlendirme yaklaşım karşılaştırılması [12]

Yöntem	Avantaj	Dezavantaj
Projeksiyon tabanlı	Basit ve rotasyona dayanıklı	Önceden bilgi gerektiren, yazı tipi değişikliklerine duyarlı ve gürültü bağışıklığı düşük.
Piksel Bağlantı Tabanlı	Basit ve rotasyona dayanıklı	Kırık ve birleştirilmiş karakterler büyük bir problemdir.

3.2.1. Projeksiyon tabanlı yaklaşım

Projeksiyon tabanlı yaklaşımlar, karakterlerin ve plakanın arka planının, görüntünün binarize edilmesinden sonra karşıt değerler veren farklı renklere sahip olduğu gerçeğini kullanır. [35]'de ikili görüntü ilk önce her karakterin başlangıç ve bitiş konumlarını bulmak için dikey olarak yansıtılır. Daha sonra görüntü, her karakteri kendi başına bölümlere ayırmak için yatay olarak yansıtılır. Bu yöntem ile %97.5'lik bir bölütleme başarımı elde edilmiştir.

Dikey ve yatay projeksiyon yöntemi basittir ve yaygın olarak kullanılır. Karakter pozisyonlarından bağımsız olarak çalışır ve plaka hafifçe döndürülmüş olsa bile çalışır. Ancak, yazı tipi değişikliklerine duyarlı olan yaklaşım bu yöntemde alfanümerik karakterlerin önceden bilinmesi gerekmektedir.

3.2.2. Piksel bağlantı tabanlı yaklaşımlar

Piksel bağlantı tabanlı yaklaşımda, piksel sayma ile piksellerin karaketer ilişki haritaları çıkartılarak bölütleme yapılmaktadır. Piksel bağlantı tabanlı karakter bölütleme yaklaşımı basit bir tekniktir, ancak karakterlerin bozuk veya üst üste gelmeleri durumunda bölütleme işleminde hatalar gerçekleşmektedir.

Piksel bağlantı tabanlı yaklaşımlar [36], ikili görüntüdeki tüm bağlı pikselleri etiketleyerek bölümlendirme gerçekleştirir. Karakterlerle aynı özelliklere sahip pikseller karakterin bir parçası olarak kabul edilir.

3.3. Karakter Tanıma

Son adım, optik karakter tanıma (OCR) olarak da bilinen her bölütlenmiş karakteri tanıma adımıdır. Bu, alfanümerik karakter başına bir sınıfa sahip görüntü sınıflandırma problemi olarak görülebilir.

Mevcut yöntemler iki kategoriye ayrılabilir: şablon eşleştirme tabanlı ve öğrenmeye dayalı yaklaşımlar. Her bir yaklaşımın avantaj ve dezavantajlarının bir özeti Tablo 3.3.'te verilmiştir.

Tablo 3.3. Karakter tanıma yaklaşımlarının karşılaştırılması [12]

Yaklaşım	Avantaj	Dezavantaj
Şablon Eşleştirme Tabanlı	Basit	Her yazı tipi için şablon gerektirir ve döndürülmüş olarak çalışmaz. Deforme olarak ayrışmış karakterlerde çalışmaz.
Öğrenme tabanlı	Güçlü	Hesap yükü yüksek.

3.3.1. Şablon eşleştirme yaklaşımı

Şablon eşleştirme yaklaşımları basittir ve karakter ile şablon arasındaki benzerliği karşılaştırarak çalışır. Karaktere en çok benzeyen şablon seçilir. [26] ve [35]'de, şablon metodu ikili görüntülerde kullanılmıştır. Benzerliği ölçmek için Mahalanobis mesafesi, Jaccard değeri [35] ve Hamming mesafesi de dahil olmak üzere çeşitli benzerlik ölçütleri önerilmiştir. [35] ile verilen çalışmada karakter tanıma oranı yaklaşık %97.2'dir.

Şablon eşleştirme yöntemi basit olmakla birlikte uygulama alanı sınırlıdır. Her yazı tipi için bir şablon gerektirir, sabit bir boyut alır ve döndürülmüş ya da bozuk karakterler üzerinde çalışmaz. Sınıflandırma görevi için önemli olmayan birçok pikseli işlediği ve her şablonla karşılaştırdığı için gürültüye karşı hassastır ve nispeten yavaş çalışır.

3.3.2. Öğrenme tabanlı yaklaşımlar

Öğrenme tabanlı yaklaşımlar, bir veya daha fazla özelliğe dayalı karakterleri ayırt etmek için makine öğrenme tekniklerini kullanır. Jiao ve ark. [37], görüntü yoğunluğuna göre ayrım yapmayı öğrenen bir sinir ağı kullanmaktadır. Bazı CNN mimarisi de bu yaklaşımda iyi çalışmaktadır. CNN'ler birden çok özellik kullanır ve özellikleri önceden tanımlamaları gerekmez. [32]'de sınırlayıcı kutular boyunca kayan önceden belirlenmiş bir dokuz katmanlı CNN modeli ile %97.6 oranında bir doğruluk elde edilmiştir.

Bu metod oldukça kullanışlıdır. Tanıma kısmı hızlıdır çünkü piksellerden çok özellik temelli çalışmaktadır. Bununla birlikte, özellik çıkarımı zaman alır ve uygun olmayan özellikler seçildiğinde başarım düşer.

3.4. Konvolüsyon Mimarileri ile Nesnelerin Sınıflandırması

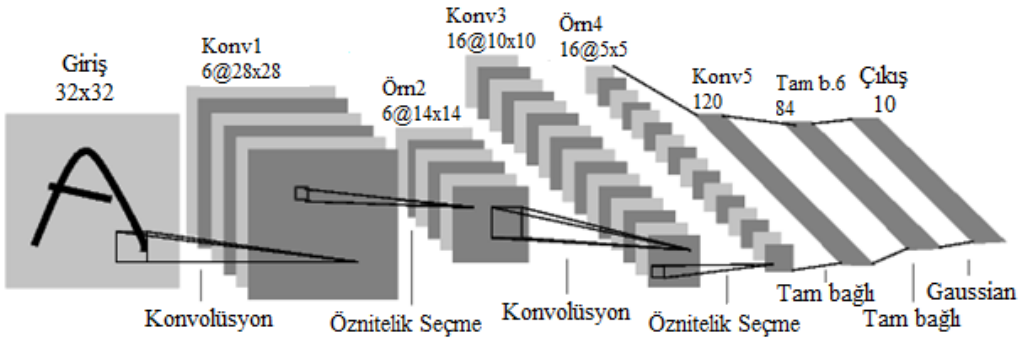
1990'lı yıllardan itibaren farklı konvolüsyonel mimariler geliştirilmiştir. Bu bölümde en dikkat çekici mimarilerden bazılarına değinilmiştir.

Göreceğimiz gibi, devrim yaratan mimarilerin çoğu ImageNet Büyük Ölçekli Görsel Tanıma Yarışması (ImageNet Large Scale Visual Recognition Competition, ILSVRC) için geliştirilmiştir [38]. Bu yarışma, hedefin on milyondan fazla resim ve 1000'in üzerinde sınıf içeren belirli bir veri setinde en düşük sınıflandırma hatasını elde etmek olduğu yıllık bir yarışmadır. Ortaya çıkan rekabet son yıllarda CNN'lerin daha da gelişmesi için itici bir faktör olmuştur.

Bir görüntüdeki içindeki nesneleri sınıflandırmak ve bu projenin ihtiyaçlarına uygun transfer öğrenimi için iyi adaylar tespit etmektedir. Bunlar aynı zamanda çoğu nesne algılama konvolüsyonel mimarilerinin önemli bir parçasıdır.

3.4.1. Geleneksel CNN'ler

Yann LeCun tarafından geliştirilen LeNet5 [19], derin öğrenme araştırmalarını hızlandıran öncü bir CNN'dir. Posta kodlarını ve rakamları tanımak için kullanılmıştır. Mimari daha sonra CNN'lere ilham vermiştir. Ancak temel olarak modern CNN'lerden farklıdır.



Şekil 3.1. Karakter tanımlama için LeNet5 Mimarisi. [19]

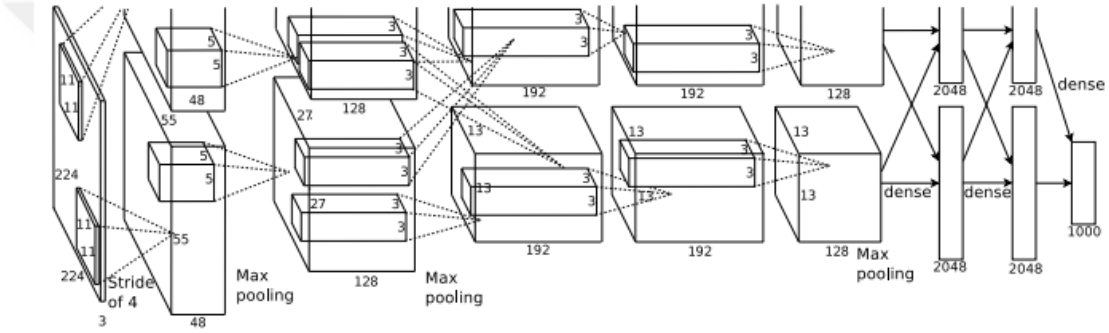
LeNet5 mimarisi Şekil 3.1.'de gösterilmektedir. Alternatif olarak, son sınıflandırıcı olarak tamamen bağlı katmanlara sahip evrişim ve birleştirme katmanları kullanılmıştır. Doğrusal olmayan aktivasyon fonksiyonu olarak, hiperbolik tanjant veya sigmoid kullanılmış ve havuzlama (pooling) fonksiyonu olarak ortalama havuzlama kullanılmıştır. Boyutların sayısını azaltmak için katmanlar arasında seyrek bağlantılar mevcuttur. Kayıp fonksiyonu olarak ise ortalama kare hatası tercih edilmiştir.

3.4.2. Derin CNN'ler

AlexNet [39], ILSVRC 2012'de yarışmış ve önemli bir başarı kazanmıştır. Temeli LeNet'e dayanır, ancak toplam 60 milyon parametre ile daha derin ve daha geniştir. Parametre sayısı nedeniyle, bellek gereksinimlerini karşılamak için birden fazla GPU kullanmak için uygulanmıştır. Çoklu GPU uygulaması Şekil 3.2.'de gösterilmektedir.

Alana katkıları:

1. Doğrultulmuş doğrusal birimin (ReLU) aktivasyon işlevi olarak kullanılması.
2. Havuzlama katmanından önce çoklu konvolüsyonel katman depolaması olması.
3. Ortalama havuzlama ile sorun yaşanmaması için maksimum havuzlama kullanılması.
4. Eğitim süresini azaltmak için GPU kullanılması.

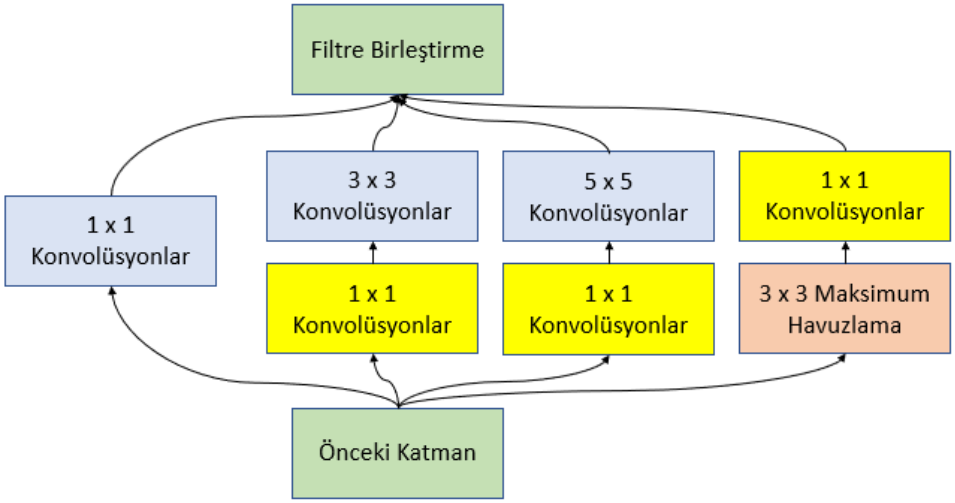


Şekil 3.2. AlexNet çoklu GPU uygulaması. [39]

3.4.3. Çok derin CNN'ler

GoogLeNet [40], ILSVRC 2014'ü kazanmıştır. Mimari, VggNet [41]'dan esinlenerek her evrişim katmanında 5x5 veya 7x7 filtreler yerine çok daha küçük 3x3 filtreler kullanmıştır. Bu fikir, katman sayısını artırma ve daha derin ağlar oluşturma imkânı sunmaktadır. Sağladığı katkılar, kullanılan parametre sayısını önemli ölçüde azaltmaktır. Bu, bir darboğaz olarak adlandırılan pahalı paralel bloklardan önce ucuz 1x1 filtreler kullanılarak gerçekleştirilmiştir.

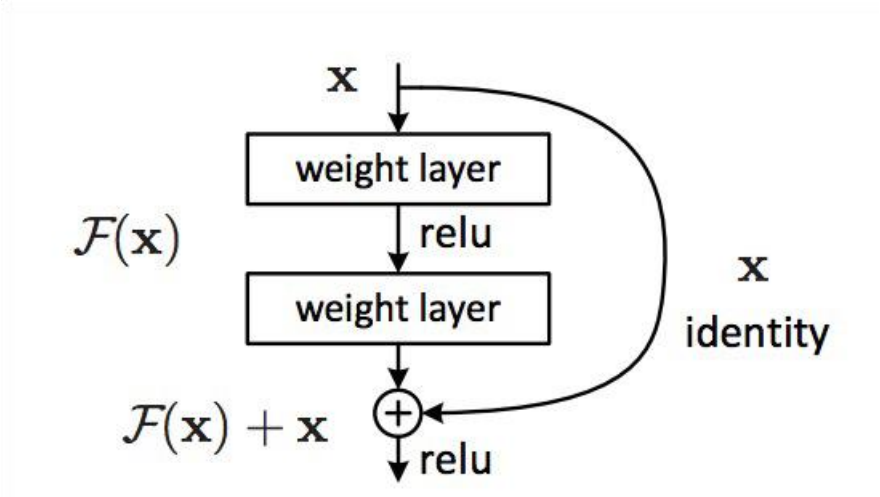
Şekil 3.3.'de bu tekniği kullanarak InceptionV1'deki bir modülü göstermektedir. Parametre sayısını daha da azaltmak için, ortalama havuzlama kullanılmıştır ve tam bağlantılı katmanlar yerine sınıflandırıcılar olarak kullanılmıştır.



Şekil 3.3. InceptionV1'deki bir modül. [40]

3.4.4. Artık CNN'ler (resual-CNNs)

ResNet [42], ILSVRC 2015'i kazanmıştır. Önceki mimarilere göre çok fazla katmandan oluşmaktadır. Katmanlar arasında Şekil 3.4.'te modellenen atlama bağlantıları kullanılmıştır. Bir atlama bağlantısı, giriş sinyali tarafından bir dizi katmanı atlamak için kullanılan bir bağlantıdır. Bu teknikle 1000'in üzerinde katmana sahip CNN'ler eğitilebilir. Yapının neden bu kadar iyi çalıştığına dair görüşler hala araştırılmaktadır. Ampirik araştırmalar, ResNet'in tüm ağın seri akışı yerine, paralel olarak hareket eden yaklaşık 20-30 kat derinlikte bloklarla çalıştığını göstermiştir [43]. Böylece, davranış nispeten sığ ağların toplulukları ile karşılaştırılabilir. Ek olarak, çıktı RNN'lerde olduğu gibi tekrar tekrar geri beslendiğinde görsel kortekste ventral akımın biyolojik olarak uygun bir modeli olarak görülebilir [44].



Şekil 3.4. ResNet bağlantı modeli. [42]

3.5. Konvolüsyon Mimarileri ile Nesne Tespiti

Nesne tespiti için kullanılan konvolüsyonel mimariler, aynı zamanda nesnenin yerini belirlemeye çalışan CNN sınıflandırmalarından farklıdır. Bu daha zor bir görevdir, ancak bir görüntüdeki nesneleri algılamak, bölümlere ayırmak ve sınıflandırmak için kullanılabildiğinden daha fazlasını başarır.

PASCAL Görsel Nesne Sınıfları (PASCAL Visual Object Classes, VOC) [45], nesne tespitinde her yıl düzenlenen bir rekabettir. Konum açıklamalarına sahip farklı sınıflara ait nesnelerin görüntülerini içeren yayınlanmış veri kümeleri, algılama ağlarını karşılaştırmak için sıklıkla kullanılır. Ortalama ortalama hassasiyet (mean Average Precision, mAP) standart bir değerlendirme ölçütü olarak kullanılır.

Son yıllara kadar, nesne algılama algoritmaları, nesnelerin doğru bir şekilde konumlandırılması ve sınıflandırılması düşünüldüğünde düşük bir hassasiyete sahiptir. Yönlendirilmiş Gradyanların Histogramı (Histogram of Oriented Gradients, HOG) özelliklerini kullanan destek vektör makineleri (Support Vector Machines, SVM) [46], kök ve parça şablonları kullanarak deforme olabilen parça modelleri en son teknolojiye sahiptir, ancak bu yöntemlerle yüksek doğruluk elde edilememiştir. En iyi HOG temelli SVM'lerden biri VOC 2007 veri setinde sadece %33.7'lik bir mAP skoruna ulaşmıştır [47].

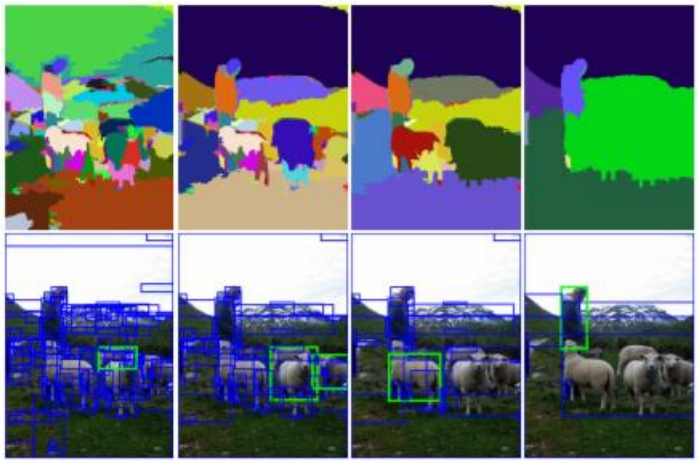
Bölge tabanlı CNN'lerin (R-CNN) keşfi ve R-CNN'lerin aşağıdaki iyileştirmeleri, doğruluğu çarpıcı biçimde artırarak nesne algılama alanında çığır açmıştır.

3.5.1. Bölge tabanlı CNN'ler

Girshick ve ark. [48] bir görüntüdeki nesneleri algılamak için 2014 yılında bölge tabanlı bir CNN (R-CNN) inşa etti. Mimari, VOC 2007 veri setinde önceki en iyi sonuçlara göre tespit oranını %50'den daha fazla geliştirmiştir ve 66'lık bir mAP skoruna ulaşmıştır. Hız oldukça yavaştır ve görüntü başına işlem süresi 20 saniyedir. Genel olarak, sistem üç modülden oluşmaktadır.

İlk modül, [49] 'de önerilen yaklaşımla sınırlayıcı kutular olarak da adlandırılan kategoriden bağımsız bölge önerileri oluşturmaktadır. Tıkanma sınırları, geleneksel kenar ve bölge temelli yaklaşımla 3D yüzey ve derinlik özelliklerini çıkaran yöntemlerle etiketlenmiştir [50]. Görüntünün 3D yorumu yeniden yapılandırılmıştır ve denetimli öğrenme ile kategoriden bağımsız bölgeler önerebilmektedir. Çok yönlü bir aramaya devam ederken ayrıntılı aramayı önlemek için seçici bir arama [51] gerçekleştirilmiştir. Yüksek düzeyde, bu seçici arama, görüntüye farklı pencere boyutlarıyla bakar ve her bir boyut, nesneleri tanımlamak için renk, yoğunluk ve dokularına göre bitişik pikselleri bir araya getirmeye çalışır. Şekil 3.5, seçici aramanın önerdiği piksel gruplarının örneklerini ve teklif edilen ilgili sınırlayıcı kutuları göstermektedir.

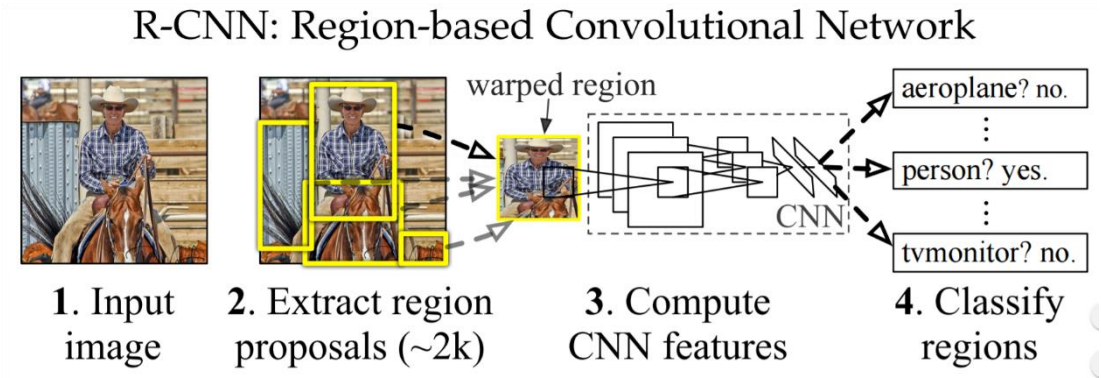
İkinci modül, sabit bir 4096 boyutlu özellik vektörü çıkaran büyük bir evrişimli ağdır. Daha önce tarif edilen GoogleNet ile aynı ağ mimarisini kullanan beş konvolüsyonel katman ve bunu takip eden iki tam bağlantı katmanı içerir [40]. Ağ yalnızca sabit boyutlu bir girişe izin verdiğinden, tüm görüntüler çarpıktır ve 227 x 227 vektörüne dönüştürülür.



Şekil 3.5. Alt görüntülerde sınırlayıcı kutular, üst görüntüler bitişik piksel grupları. [51]

Son modül, sabit özellik çıkarıcısını her bölgeyi önceden tanımlanmış kategorilere ayıran doğrusal bir SVM'e tabi tutulur. Şekil 3.6.'da giriş görüntüsünden her bölgeye sınıflandırılmış işlemi göstermektedir.

R-CNN'nin bölgeler sınıflandırıldıktan sonra son adımı, bulunan sınırlayıcı kutuların iyileştirilmesinin ve sıkılaştırılmasının mümkün olup olmadığını görmektir. Bu, nesnelere karşılık gelen alt bölgelerin girdi olarak alındığı ve alt bölgelerin yeni geliştirilmiş sınırlama kutusu koordinatlarının çıktı olarak verildiği basit bir doğrusal regresyon modeli ile gerçekleştirilir.



Şekil 3.6. R-CNN, 1. Giriş görüntüsü. 2. Bölge önerileri. 3. Özellik vektörü hesabı. 4. Bölge sınıflandırma. [48]

3.5.2. Hızlı R-CNN

Hızlı R-CNN Girshick ve ark. tarafından geliştirilen mevcut R-CNN modelinin geliştirilmiş ve basitleştirilmiş halidir. Bu gelişmeler sonucunda hız ve hassasiyette iyileşmeler olmuştur. Hızlı R-CNN [52], görüntü başına 2 saniyelik bir işlem süresiyle R-CNN'den 10 kat daha hızlıdır ve VOC 2007'de 70.0'lık bir mAP puanı aldı.

R-CNN'nin önemli bir zayıflığı, her bir bölge önerimi için CNN'den ileriye bir geçiş gerektirmesidir. Bu geçişler, CNN'nin aynı hesaplamayı birden çok kez yapmasına neden olarak, her zaman birbirleriyle örtüşmektedir. Fast R-CNN'de uygulanan çözüm, uzaysal piramit havuzlama ağları ile yakından ilgili bir teknik olan ilgi bölgesi havuzu (Region of Interest Pooling, RoIPool) ile hesaplamaları paylaşmaktır. RoIPool, giriş görüntüsünün tamamı üzerinde yalnızca paylaşılan bir ortak evrişim özellik haritası hesaplar ve her nesne önerisi için özellik haritasından sabit uzunluklu bir özellik vektörü çıkarılır.

R-CNN'nin bir diğer zayıflığı da modül sayısıdır. R-CNN, sınırlayıcı kutuları sıkılaştırmak için kullanılan doğrusal regresyon modeline ek olarak üç farklı modül daha kullanır. Bu sıralı boru hattının eğitilmesini çok zorlaştırır. Hızlı R-CNN, tüm modülleri tek bir ağda birleştirerek bu yönü geliştirir. Eğitim, nesneleri ve nesne yerini sınıflandırmak için öğrenmeyi birleştiren bir tekli sürece sürecine dönüştürülür. Böylece eğitim hızı artar. Özellik vektörleri, iki çıkış katmanına dallanan tamamen bağlı katmanların bir dizisine beslenir. Birincisi, her nesne sınıfının olasılığını verir ve ikincisi, tahmin edilen konumun köşelerini temsil eden dört değeri döndürür. Doğrusal bir regresyon ağ içindeki sınırlayıcı kutuları otomatik olarak sıkılaştırır. Son olarak, SVM sınıflandırıcısı softmax sınıflandırıcısı ile değiştirilir. Tek modül ağının bir başka avantajı, R-CNN'de mümkün olmayan eğitim sırasında ağ katmanlarının güncellenebilmesi ve doğruluğu arttırmasıdır.

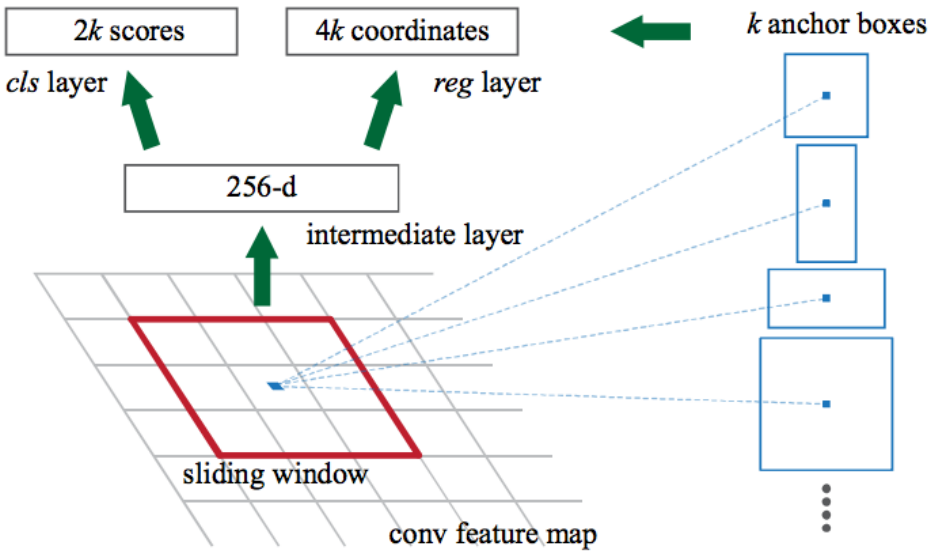
3.5.3. Daha hızlı R-CNN

Daha hızlı R-CNN [53], bölge teklif vericisini tamamen değiştirerek Hızlı R-CNN'de yapılacak iyileştirmelerin bir sonucudur. Bu mimari ile hem hız hem de hassasiyet iyileştirilmiştir. Daha hızlı R-CNN, görüntü başına 140 ms işlem süresiyle 73.2'lik bir MAP puanına ulaşmıştır. Yani, Hızlı R-CNN'den 10 kat daha hızlı ve R-CNN'den 100 kat daha hızlıdır. Temel mimari VGGnet'ten ResNet'e değiştirerek daha da yüksek bir (76.4) mAP puanı elde edilmiştir.

Daha hızlı R-CNN, bölgesel arama adımını büyük ölçüde hızlandırarak, seçici aramayı Bölge Teklif Ağı (Region Proposal Network, RPN) ile seçici aramayı değiştirerek Hızlı R-CNN'e katkıda bulunmaktadır. RPN, giriş olarak görüntü alan ve dikdörtgen bölge önerileri çıktısı alan, tamamen bağlı bir konvolüsyon ağdır. Özellik haritası üzerinde bir pencere kaydırarak çalışır. Her lokasyonda k koordinatlı 4 adet kutu, her bir çapa kutusu için iki güven puanının yanı sıra, bölge sınırlarının doğruluğu için bir puan ve bir bağlantı kutusu içinde bulunan nesnenin objeliği için bir puan önerilmiştir. Çapa kutusu, seçici olarak seçilen bir sınırlama kutusudur. Sezgisel olarak, bir ağ insanları algılamak için eğitildiğinde, insan şeklini andıran dikdörtgen sınırlayıcı kutular istenmektedir. Uzun ve ince sınırlayıcı kutular sıklıkla görünürken, geniş ve düşük sınırlayıcı kutular nadiren görünecektir. RPN, konum doğruluğunu iyileştiren ve bölge tekliflerinin sayısını azaltan ortak boy oranlarına sahip bağlantı kutuları önermektedir. Bu sayede hız artmaktadır. Şekil 3.7'de RPN sürecine genel bir bakış gösterilmektedir.

Bölge önerileri daha sonra bölgelerdeki nesneleri algılayan ve sınıflandıran Hızlı R-CNN ağına verilir. RPN ağı ve Hızlı R-CNN ağı, geri yayılma ve stokastik gradyan inişi kullanılarak bağımsız olarak eğitilmiştir. Hızlı R-CNN ağının RPN tarafından verilen bölge tekliflerini tam olarak kullanması için, iki ağın özellikleri paylaşması gerekir. Bunu yönetmek için, iki ağın eğitimi bittikten sonra iki adım ortaya atıldı. İlk olarak, Hızlı R-CNN ağı, RPN ağını başlatmak için kullanılır, ancak paylaşılan konvolüsyon katmanları sabit tutulurken, RPN'e özgü katmanlar ince ayar yapılır. İkincisi, Fast R-CNN'in katmanları ince ayar yapılmaktadır. Paylaşılan konvolüsyon

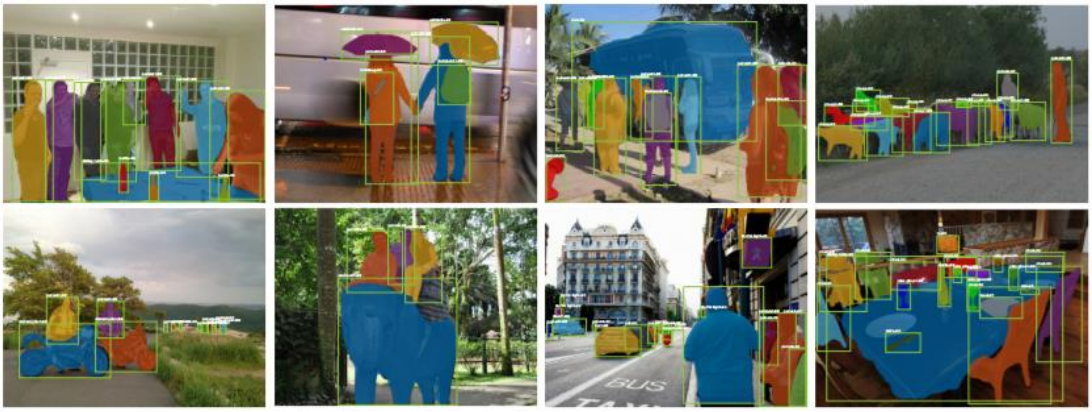
katmanlar sabit tutulur. Konvolüsyon özellikleri paylaşılarak, bölge önerisi adımı, ağda büyük bir tıkanıklığı gidererek hızı büyük ölçüde artırmaktadır ve neredeyse maliyet sıfırdır.



Şekil 3.7. RPN'e genel bakış. [53]

3.5.4. Maske R-CNN

Maske R-CNN [54] (2017), Daha Hızlı R-CNN'in bir uzantısıdır, yalnızca nesnelerin çevresindeki sınırlayıcı kutuları öngörmekle kalmaz, aynı zamanda nesneleri çevreleyen arka plandan bir maske ile ayırır. Sunduğu problem çözümü diğer R-CNN'lerden faklı olan bu mimarinin mAP puanı 35.7'dir. Bununla birlikte, Maske R-CNN mevcut tüm diğer modellerden daha iyi performans gösteren, görüntü segmentasyonu içindeki en gelişmiş teknolojidir. Maskelere ek olarak aynı seviyedeki sınırlayıcı kutuları daha hızlı R-CNN'de tahmin edebiliyor. Maske R-CNN, Faster R-CNN'e yalnızca küçük bir hesap yükü daha ekler. Bunun sonucu olarak hız biraz azalır. Yine de görüntü başına 200 ms'nin altındaki bir işlem süresiyle çalışabilmekedir. Maske R-CNN'den elde edilen sonuçlar, Şekil 3.8'de gösterilmektedir.



Şekil 3.8. Maske R-CNN'den elde edilen maskeler renkli olarak gösterilmiştir.[54]

Maske R-CNN, Faster R-CNN'nin sınıflandırma ve sınırlayıcı kutu regresyon ağına paralel özellik haritasının üstüne yeni bir dal ekleyerek piksel düzeyinde segmentasyon yapabilir. Bu dal, verilen bir pikselin bir nesnenin parçası olup olmadığını söyleyen ikili maskeyi çıkaran tamamen evrişimli bir ağdır.

Orijinal R-CNN'den başka bir küçük ayar, Maske R-CNN'nin amaçlandığı şekilde çalışması için gerekli. RoIPool, orijinal görüntüde ilgi bölgelerini bulur ve bunu Daha Hızlı R-CNN'deki özellik haritası üzerine eşler. Daha iyi ayarlanmış özellikler gerektiren görüntü bölütlemede kullanılacak kadar kesin değildir. Nedenini anlamak için 25x25 özellikli bir haritaya eşlenen 128x128 piksellik bir görüntü düşünebiliriz. İlgilenilen bölge 15x15 piksellik bir alan ise, bu özellik haritasındaki "15*(25/128) = 2.93" piksellik kenarlara sahip bir kareye karşılık gelir. RoIPool kullanıldığında, bu sayı aşağı doğru yuvarlanır ve ilgili bölgeyi biraz yanlış hizalayan 2x2 özellik alanı ortaya çıkar.

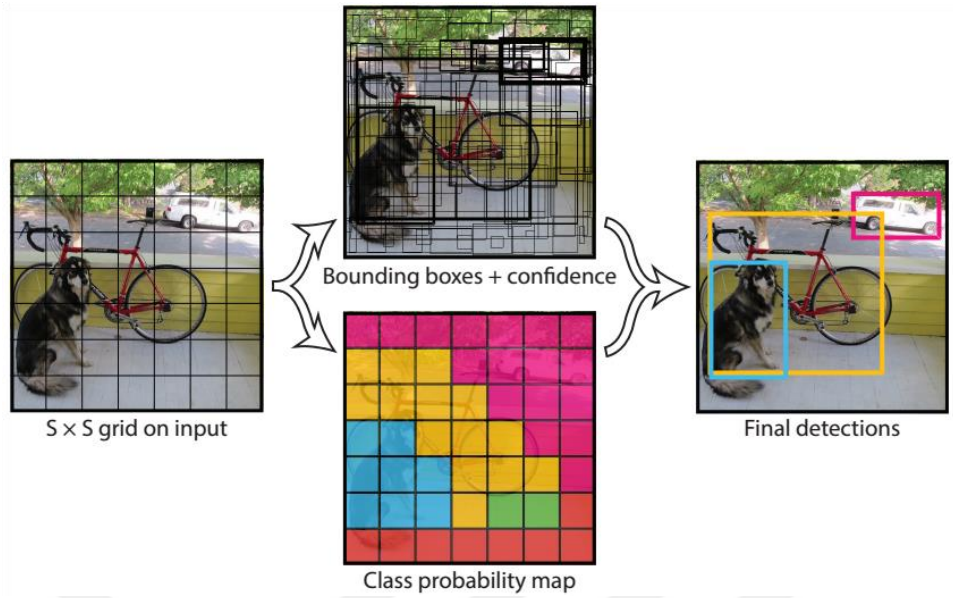
Literatürde, yuvarlanmayı önlemek için RoIAlign yöntemi tanıtılmıştır. RoIAlign, 2.93x2.93 özelliğinin ne olacağı konusunda kesin bir fikir veren pikseller arasındaki piksel değerlerini tahmin etmek için en yakın dört pikselin ortalama ağırlıklı ortalamasını kullanan, bir yeniden örnekleme yöntemi olan bilinear enterpolasyon kullanır. Böylece, daha hassas segmentasyona neden olan ilgili bölgenin yanlış hizalanmasını önler.

3.5.5. YOLO

YOLO (You Only Look Once) [55] (2015), R-CNN çerçevesinden oldukça farklı çalışan ilk başarılı tek çekimli CNN mimarisidir. Tek çekim mimarilerin işlemi, görüntü bölgeyi ve nesne önerilerini ortadan kaldırarak tek bir CNN'de görüntüyü yalnızca bir kez işlediği için basittir. Sonuç çok daha hızlı nesne tespiti sağlar. YOLO'nun işlem süresi, görüntü başına sadece 22ms'dir, Daha Hızlı RCNN'den yaklaşık 6-7 kat daha hızlıdır. Bu hız saniyede 45 kareye karşılık gelir ve video akışlarında nesnelerin gerçek zamanlı algılanmasını gerçekleştirmek için yeterlidir. Bununla birlikte, VOC 2007 veri setinde 63.4'lük bir mAP puanı elde etmiştir. Bu puan R-CNN ailesinden önemli ölçüde daha kötüdür.

YOLO, R-CNN'lerin ayrı bileşenlerini tek bir sinir ağında birleştirir. Ağ, tüm sınıflardaki sınırlayıcı kutuları tahmin etmek için görüntünün tamamındaki özellikleri kullanır. Giriş görüntüsü S × S ızgarasına bölünmüştür. Her bir ızgara hücresi, B sınırlayıcı kutularını ve her kutu için bir güven puanını öngörür. Bu güven puanı, sınırlama kutusundaki bir nesnenin içinde bulunma olasılığını belirleyen bir terimden ve sınırlama kutusunun öngörülen kutu ile yer gerçeği arasındaki birliktelik üzerindeki kesişme noktasında kesişim alarak ne kadar doğru olacağını hesaplayan bir terimden oluşur.

Her bir ızgara hücresi ayrıca C koşullu sınıf olasılıklarını Pr ($Class_i$ | Object) tahmin eder. Öngörülen çok sayıda kutu olmasına rağmen, hücre başına sadece bir sınıf olasılık seti öngörülmüştür. Test zamanında, sınıf olasılıkları, kutuda görünen bir sınıfın olasılığını ve kutunun nesneye ne kadar iyi uyduğunu kodlayan sınıfa özgü güven puanını veren her kutunun güven puanıyla çarpılır. Şekil 3.9.'da YOLO modelinin genel yapısı görülmektedir.



Şekil 3.9. YOLO'ya Genel Bakış. [55]

3.5.6. YOLOv2

YOLOv2 [56], YOLO9000 olarak da adlandırılır, orijinal YOLO ağının geliştirilmiş bir sürümüdür. Çok ölçekli bir eğitim yöntemi kullandığından, farklı boyutlarda çalışabilir ve hız ile doğruluk arasında bir denge sağlayabilir. Saniyede 67 kare hızında çalışan sürüm, VOC 2007'de 76.8 mAP'lik bir puan alırken, 40 FPS (frame per second)'de çalışan daha yavaş sürüm 78.6'lık bir mAP'ye ulaşmaktadır. YOLOv2, yalnızca tahmin doğruluğunu göz önüne alırken, ResNet ile Daha Hızlı R-CNN'den biraz daha iyi performans gösterir. Bununla birlikte, YOLOv2, 8 kat daha hızlı çalışan hızı dikkate alarak Faster R-CNN'den daha iyi performans göstermektedir.

Orijinal YOLO, sınıflandırma ağını 224×224 çözünürlükte çalıştırmaktadır, ancak tespit parçası için 448 × 448'e yükseltmektedir. Sınıflandırıcı, ağı tam 448×448'e ince ayar yaparak istemesinden dolayı, ImageNet'teki ön eğitim daha yavaştır, ancak sonuçta ortaya çıkan filtreler daha iyi ayarlanır. Bu mAP puanını neredeyse %4 arttırır.

Ağların daha küçük nesneleri yerelleştirme yeteneğini geliştirmek için bir geçiş katmanı eklenir. Katman yüksek tanımlı özelliklere sahip daha önceki katmanlardan

daha ince tanecikli özellikler getirir ve bitişik özellikleri farklı kanallara yerleştirerek daha düşük çözünürlük özellikleri ile birleştirir. Bu iyileşme sonucunda mAP puanı %1 artmıştır. YOLOv2'nin farklı boyutlardaki görüntülerde daha etkin çalışmasını sağlamak için çok ölçekli bir eğitim sunulmuştur. YOLOv2 için standart giriş çözünürlüğü 416×416'dır, yani her görüntü hemen bu çözünürlükte küçültülür. Çok ölçekli eğitim sırasında bu çözünürlük sabit tutulmaz ve ağ rasgele {32, 352, ..., 608} kümesinden her 10 partide yeni bir çözünürlük seçer ve bu da 32 faktöre sahiptir. Bu, ağı, çeşitli boyutlardaki çözünürlüklerde iyi tahmin etmeye zorlamaktadır.

Son olarak, YOLOv2 test süresindeki farklı çözünürlükler arasında seçim yapabilir. Çözünürlük hız ve doğruluk arasında bir denge oluşturur. Daha yüksek bir çözünürlük daha iyi bir mAP puanı verir, ancak saniyede daha az kare işlenir. 544×544'te, en yüksek olan 78.6, 40 FPS'de çalışmaktadır. 288×288'de, en düşük olan 69.0, 91 FPS'de çalışmaktadır.

BÖLÜM 4. SİSTEM MİMARİSİ

Bu bölümde tez çalışmasında otomatik plaka tanıma uygulaması için kullanılan sistem mimarisi tanıtılmıştır.

4.1. Giriş

Sinir ağlarının kullanıldığı sistemlerde problem çözümü için eğitim verilerinin önemli bir payı bulunmaktadır. Yapay sinir ağı son derece iyi tasarlanmış olsa bile, eğitim verileri test verilerini yansıtmıyorsa, iyi sonuçlar elde edilemez. Bu tezde, plaka içeren araç görüntüleri, plaka görüntüleri ve karakter görüntüleri eğitim verileri olarak kullanılmıştır. Ayrıca plaka konum, karakter konum ve karakter değerlerini içeren etiketler eğitim sırasında göz önünde bulundurulmuştur.

Sinir ağının eğitim performansını yükseltmek için iki yaklaşım göz önünde bulundurulmuştur. İlk yaklaşım nispeten küçük miktarlarda yüksek kaliteli veri kullanmaktır. İkinci yaklaşım ise yüksek miktarda düşük kaliteri veri üretmektir. Bu tezde gerçek zaman görüntülerinden oluşan, küçük miktarda yüksek kalitede veriler kullanılmıştır. Plaka tespit işlemi için 1219 araç görüntüsü, karakter tespit işlemi için 300 plaka görüntüsü, karakter tanıma işlemi için ise 10960 adet rakam, 24960 adet büyük harf karakter görüntüsü kullanılmıştır. CNN uygulamaları için bağlamında bu veriler küçük bir veri setidir.

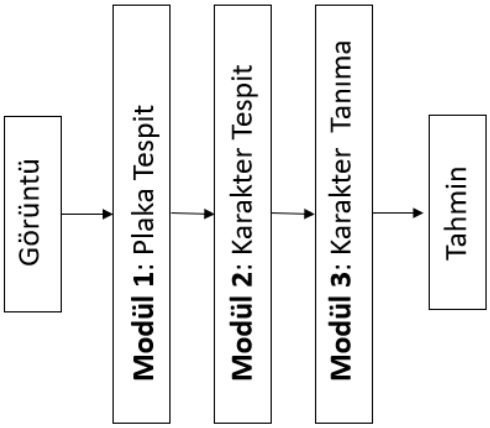
Ancak, Bölüm 3.4. ve Bölüm 3.5.'te de bahsedildiği gibi nesne sınıflandırılması ve nesne tespiti için var olan konvolüsyonel mimarilerin çoğu ImageNet ve VOC gibi büyük veri kümeleri üzerinde eğitilmiştir. Bu mimariler genel nesne sınıflandırma ve algılama için oluşturulmuş olsalar bile plaka tanıma / karakter tanıma uygulamaları için kullanılabilirler.

Sistem boru hattı, görevin ayrı modüllere nasıl dağıtıldığını tanımlamaktadır. Her modül önceki modülden giriş verilerini işler ve verileri bir sonraki modüle verir. İlk modül işlenmemiş görüntü verilerini alır, ardından gelen modüllere ileterek görüntü üzerinde işlemler gerçekleştirir. Sistem son modüle geldiğinde görüntünün içerdiği plakadaki karakterlerin tahminini vermektedir. ALPR problemini çözmenin birçok farklı yöntemi ve dolayısıyla birçok farklı sistem boru hattı mevcuttur. Ancak bu tezde sadece birkaçı açıklanmıştır. Her bir boru hattı önceki önceki bölümlerde sunulan mevcut derin öğrenme tekniklerine ne kadar uygun olduğuna göre değerlendirilir.

Karakter tespiti tek adımlı bir boru hattıdır. Burada karakterlerin doğrudan görüntüden tespit adilmesi amaçlanmıştır. R-CNN, plakalardan oluşan bir karakter veri kümesi üzerinde eğitilir. Öğrenilen özellikler giriş görüntüsündeki karakterleri tespit etmek ve tanımak için kullanılır.

Plaka ve karakter tespiti sistem boru hattında görev iki aşamaya bölünür. İlk aşamada plaka tespit işlemi, ikinci aşamada ise plakadaki karakterlerin tespiti amaçlanmıştır. Her iki aşamada da aynı veya farklı R-CNN'ler kullanılabilmektedir. Hem plaka hem de karakterlerin bölümlendirilmesi algılama işleminin bir parçasıdır. Burada, sadece plakanın bulunduğu bölgelerde karakter aranacağından karakterleri doğrudan tespit eden boru hattından daha iyi sonuçlar elde edilir. Diğer bir avantaj ise her iki basamağın da uygulamayı basitleştiren aynı nesne algılama mimarisiyle çözülebilmesidir. Yaklaşımın dezavantajı ise tek adımlı boru hattı çözümüne göre işlem hızının iki kat daha yavaş olmasıdır.

Plaka ve karakter tespiti, karakter tanıma ALPR problem çözümü için en çok kullanılan boru hattıdır. Üç aşamalı olarak gerçekleştirilmektedir. İlk adım, plakayı tespit etmektedir. İkinci adım, plakadaki karakterleri tespit edip ayırmaktadır. Son adım ise tespit edilen her bir karakteri tanımaktadır. Yöntemin sistem boru hattı Şekil 4.1.'de gösterilmektedir.



Şekil 4.1. Sistem Boru Hattı

Plaka tespit adımı R-CNN kullanılarak çözülebilmektedir. Karakter tespit adımı için birçok alternatif mevcuttur. Bunlardan bazıları, R-CNN kullanmak veya piksel bağlantı bazlı yöntemler ile karakter tespit edilmesi gibi bir veya daha fazla yöntem ile öğrenilebilmektedir. Karakter tanıma adımı, olası karakterleri tanıma için eğitilmiş CNN nesne sınıflamasını kullanmaktadır. Bu yapı sayesinde sistem performansı yükseltilebilir.

Bu tez için kullanılan sistem boru hattı Şekil 4.1.'de gösterilen boru hattıdır. Bu boru hattının seçilme sebebi değerlendirilen sistem boru hatları içersinde en doğru sonuçlar veren ve sistem için yeterli hız performansı veren yapı olmasıdır. Tek adımlı boru hattına kıyasla yavaş olmasına rağmen doğruluk oranın tek adımlı boru hattına oranla daha yüksektir. İki adımlı sistem boru hatlarında karakter bölütlendirme işleminin başarımının gürültüden kolay etkilenebilmesi ve doğruluğun kullandığımız sistem boru hattına oranla düşük olması seçimi belirlemiştir.

Otomatik lisanslı plaka tanıma işlemi plaka tespit, karakter tespit ve karakter tanıma adımlarından oluşmaktadır. Bu aşamalardan plaka tespit ve karakter tespit adımları için iki R-CNN ve karakter tanıma işlemi için çok katmanlı bir CNN kullanılmıştır.

Nesne tespiti için kullanılan altı farklı konvolüsyonel mimari önceki bölümlerde incelenilmiştir. Bunlar: R-CNN, hızlı R-CNN, daha hızlı R-CNN, maske R-CNN,

YOLO ve YOLOv2 mimarileridir. Tablo 4.1.'de bu mimariler ile bir görüntünün işlenmesi için geçen süre ve VOC 2007 veri setinde mAP oranları verilmiştir.

Daha Hızlı R-CNN yüksek doğruluk gerektiren, hız gereksiniminin çok olmadığı projelerde kullanılabilir. Daha hızlı R-CNN, R-CNN ve hızlı R-CNN'nin geliştirilmiş bir versiyonudur. YOLOv2 mimarisi yüksek hız gerektiren gerçek zamanlı sistemlerde tercih edilmektedir. YOLOv2 ise YOLO'nun geliştirilmiş versiyonudur.

Tablo 4.1. R-CNN ve YOLO mimarilerinin hız ve zaman karşılaştırması.

	mAP	FPS
R-CNN	66.0	0.05
Hızlı R-CNN	70	0.5
Daha Hızlı R-CNN	76.4	5
Mask R-CNN	73.2	5
YOLO	63.4	48
YOLOv2 288 x 288	69.0	91
YOLOv2 544 x 544	78.6	40

Mimari seçim için tahmin doğruluğu ve hız parametreleri göz önüne alınmaktadır. Bu parametreler göz önüne alındığında plaka ve karakter tespit işlemi için YOLOv2 seçimi uygun görünmektedir. Ancak bu çalışmada gerçek zamanlı bir çalışma yürütülmediği için tahmin doğruluğunun daha yüksek olan daha hızlı R-CNN mimarisi kullanılmıştır. Ayrıca karakter tanıma işlemi için çok katmanlı CNN mimarisi kullanılmıştır.

4.2. Tensorflow kullanarak nesne algılama uygulaması

ALPR sisteminde plaka ve karakter algılama aşamalarının gerçekleştirildiği Tensorflow Nesne Algılama API üzerine açıklamalar yapılmıştır. Ayrıca kullanılan sistemin konfigürasyonu için yapılan adımlar tanıtılmıştır [57].

4.2.1. Veri seti ve ön işleme

ALPR sisteminde veri seti olarak gerçek zamanlı çalışmakta olan kontrollü geçiş sisteminden alınan görüntüler kullanılmaktadır. Tensorflow Nesne Algılama API'ına sunulacak görüntünün tahmin edilebilmesi için eğitim işlemi yapılması gerekmektedir. Bu işlem için veri setinin tensorflow'a uygun formata dönüştürülmesi gerekmektedir. Nesne tespit modelini eğitmek ve değerlendirmek için ilk olarak, plaka ve karakter sınıflarının konumları kutu içerisinde belirtilerek veri setleri çıkartılmıştır. Bu işlem sırasında karşılaşabilecek bazı problemler şunlardır:

1. Eğitim ve değerlendirme için kullanılan görsellerin çok küçük olmamasına dikkat edilmelidir.
2. Her sınıf için sınırlayıcı kutu verilerinin sadece belirli bir sınıf için etiketlenmesi gerekmektedir.

Şekil 4.2.'de plaka bölgesi ve karakterlerin etiketlenmesi ile ilgili örnek bir görsel verilmiştir.

a. Plaka Etiketleme

b. Karakter Etiketleme

Şekil 4.2.Veri seti etiketleme, a: Plaka bölgesinin etiketlenmesi, b: Karakterlerin etiketlenmesi.

Yukarıdaki oluşabilecek problemler göz önünde bulundurularak, eğitim ve değerlendirme amaçlarına uygun veri kümeleri oluşturmak için aşağıdaki aşamalar gerçekleştirilmiştir:

1. Veri kümelerinin seçimi ve etiketlenmesi: Her sınıf için veri setinde etiket konumu ve ismi belirtilmiştir. Bu işlem tüm eğitim ve test veri setleri için el ile gerçekleştirilmiştir. Bu işlemin gerçekleştirmek için LabelImg [58], adında bir program kullanılmıştır. Bu program etiketlenecek sınıfın sınırlayıcı kutu koordinat ve etiket bilgilerini XML formatta kaydetmektedir.
2. Eğitim ve test veri setinin oluşturulması: Etiketleme işlemi tamamlandıktan sonra, her bir sınıf veri seti iki gruba ayrılmıştır: Bu gruplar eğitim veri seti ve test veri setidir. Her eğitim veri seti ve test veri seti sırasıyla %80'e %20 oranında veri içerecek şekilde ayrılmıştır. Daha sonra her sınıf için karşılık gelen veri kümesi birleştirilerek eğitim veri seti ve test veri seti oluşturulmuştur.
3. TFRecord oluşturulması: Tensorflow nesne algılama modelini eğitmek için veri setleri TFRecord formatına dönüştürülür. TFRecord, Tensorflow tarafından eğitim için desteklenen standart bir formattır. Eğitim ve test setleri başlangıçta XML formattadır. Bu veri setleri önce CSV (Virgülle Ayrılmış Değerler) formatına ve daha sonra bir TFRecord formatına dönüştürülürler.

Tensorflow modelleri birçok veri seti için eğitilmiştir. Bu projede COCO veri seti için eğitilmiş "faster_rcnn_inception_v2" modeli kullanılmıştır.

4.2.2. Daha hızlı R-CNN ile nesne algılama

Daha hızlı R-CNN Inception-v2 modeli, Bölüm 4.2.1.'de belirtilen yöntemlerle ayarlanan eğitim ve doğrulama veri setleri için kullanılmaktadır. Eğitime başlamak için sağlanan konfigürasyon dosyası (faster_rcnn_inception_v2_pets.config) temel alınmaktadır. Yapılandırma dosyası, Tensorflow tarafından daha hızlı R-CNN için

sağlanan ince ayar işlemini başlatmak için kontrol noktası gerekmektedir. Yapılandırma dosyasına eğitim kaydı ve test kaydı konumlarının yerlerinin bildirilmesi gerekmektedir. Eğitim kaydı, eğitim veri seti için oluşturulan TFRecord dosyası ve test kaydı, doğrulama veri seti için oluşturulan bir dosyadır. Yapılandırma dosyasına ayrıca nesne etiket haritası konumunun da bildirmesi gerekmektedir. Bir nesne haritasının yapısı Şekil 4.3.'de gösterilmektedir ve dosya uzantısı '.pbtxt' ile sona ermektedir.

```
item {
  id: 1
  name: 'plate'
}
```

Şekil 4.3. Plaka etiket haritası

Eğitim kaydı ve test kayıt dosya konumları Şekil 4.4.'de gösterildiği gibi nesne etiket haritasıyla birlikte eklenmektedir. Yapılandırma dosyası, model eğitimimiz için kullanılan sınıfların sayısına göre güncellenmektedir. Uygulama için iki ayrı eğitim gerçekleştirilmiştir. Her eğitim işlemi için bir sınıf mevcuttur.

```
train_input_reader: {
  tf_record_input_reader {
    input_path: "C:/tensorflow1/models/research/object_detection/train.record"
  }
  label_map_path: "C:/tensorflow1/models/research/object_detection/training/labelmap.pbtxt"
}

eval_config: {
  metrics_set: "coco_detection_metrics"
  num_examples: 322
}

eval_input_reader: {
  tf_record_input_reader {
    input_path: "C:/tensorflow1/models/research/object_detection/test.record"
  }
  label_map_path: "C:/tensorflow1/models/research/object_detection/training/labelmap.pbtxt"
  shuffle: false
  num_readers: 1
}
```

Şekil 4.4. Konfigürasyon dosyası güncelleme örneği

Yapılandırma dosyasında belirtilen diğer önemli parametreler şunlardır:

1. En-boy oranları: 0.5, 1.0, 2.0
2. Öğrenme oranı: Başlangıç değeri 0.00002'dir.
3. Batch size: 1
4. Tekrarlama Sayısı: 200000

Yapılandırma dosyası veri setine göre güncellendikten sonra, eğitim işlemine başlanmaktadır.

4.2.2.1. Tensorflow nesne algılama API'si model ayarları

Projede plaka algılama ve plakadaki karakterleri algılama işlemleri için bu tensorflow model seçimi yapılmalıdır. Her iki işlem için de belirlediğimiz "faster_rcnn_inception_v2_coco" modelini belirtilen veri setine ve belirtilen sınıflara göre ayarlanmıştır. İlgili modelin seçiminde, katı hal sürücüsü (SSD) modelleri en iyi performansı sergilerken, Tablo 4.1.'de belirttiğimiz gibi daha hızlı R-CNN en doğru tahmin üreten modeldir.

4.2.3. Sistem mimarisi

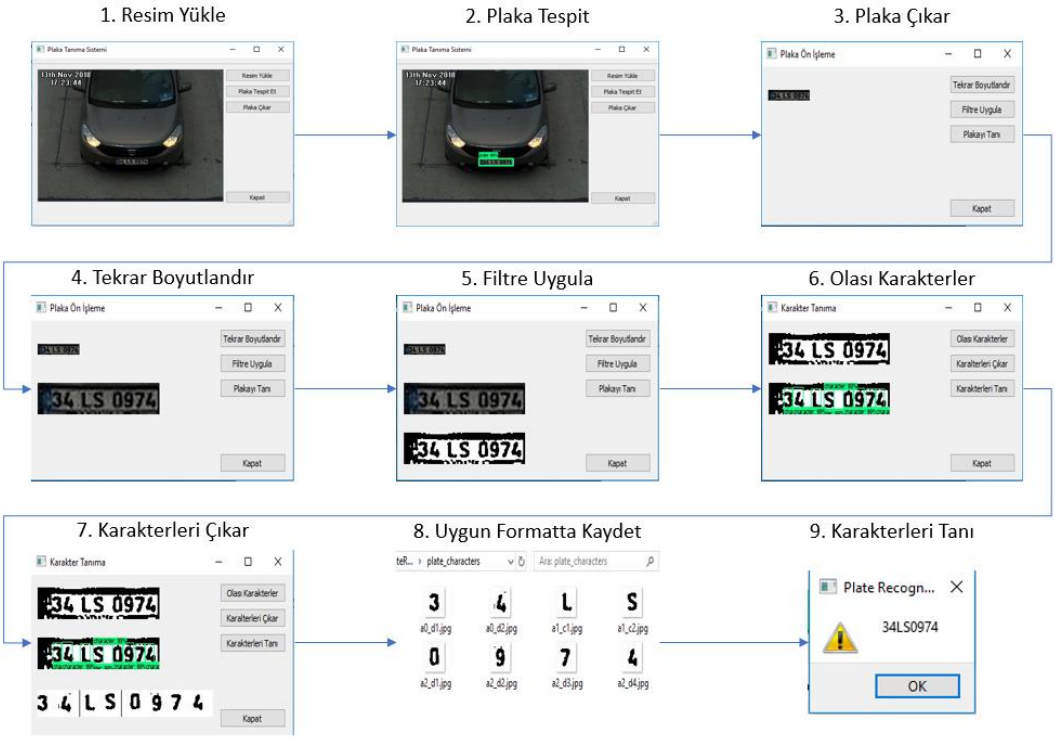
Dört aşamada gerçekleşen bu sistemin ilk aşamasında, sistem işlenmemiş görüntüyü girdi olarak alır. Görüntüdeki plaka veya plakalar tespit edilir. Plaka konumu belirtildikten sonra plaka görselden çıkartılır. Plaka çıkartılma işleminde plaka kısmının zarar görmesini engellemek adına çıkartılacak plaka görüntüsünün etrafında fazladan pay bırakılmaktadır.

İkinci aşamasında, çıkartılan plaka görüntüsüne tekrar boyutlandırma ve filtre işlemlerinin uygulandığı aşamadır. Bu aşamada karakter tespit işleminin başarımını artırmak için plaka görüntüsünün yeniden boyutlandırılması amaçlanmıştır. Yeniden boyutlandırma işleminden sonra oluşan gürültülerin temizlenmesi ve daha keskin bir

hale getirilmesi için filtre işlemi uygulanır. Elde edilen çıkış bir sonraki adım olan karakter tespit işlemine girdi olarak verilir.
Üçüncü aşama karakter tespit aşamasıdır. Bu aşamada bir önceki adımın sonucu girdi olarak verilen plakadaki karakterler tespit edilmektedir. Karakter tespit işlemi yapıldıktan sonra tespit edilen karakterlerin zarar görmesini engellemek için kenarlarında diğer karakter alanlarına zarar vermeyecek şekilde boşluklar bırakılarak görselden çıkartılır. Plaka görselinden çıkartılan karakterler, karakter tanıma işlemine tabi tutulur.

Dördüncü aşamada karakter tanıma işlemi gerçekleşir. Bu aşama iki adet çok katmanlı CNN modülünden oluşmaktadır. Birincisi modül, plakada bulunan karakterleri tanıma işlemi için eğitilmiş ağı içerirken, ikinci modül, plakada bulunan rakamları tanıma işlemi için eğitilmiş ağı içermektedir. Tüm sistemin çalışma yapısı Şekil 4.5.'de gösterilmektedir.

ALPR sisteminde gerçek zamanlı çalışan bir otoyol kontrollü geçiş sisteminden alınmış gerçek görseller ile eğitim ve test işlemleri gerçekleştirilmiştir. Plaka tespit işlemi eğitimi için 1380 araç görseli kullanılmıştır. Karakter tespit işlemi eğitimi için 300 plaka görseli kullanılmıştır. Karakter tanıma işlemi için ise 10960 adet rakam, 24960 adet ise karakter kullanılmıştır.

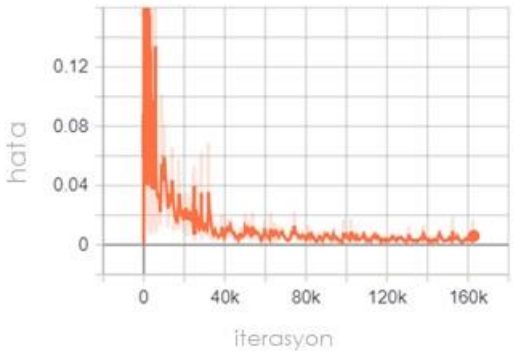


Şekil 4.5. Sistemin Çalışma Akışı

4.2.4. Plaka tespit modülü

Plaka tespit modül eğitminin gerçekleştirilmesi için plaka, eğitim ve test veri setlerinin Bölüm 4.2.1.'de belirtilen aşamaları gerçekleştirilmektedir. Daha sonra Bölüm 4.2.2.'de belirtilen genel konfigürasyon ayarları plaka tespit işlemine özel yapılandırılır. Nesne etiket haritası Şekil 4.3'de gösterildiği gibi ayarlanmaktadır. Bu işlemler gerçekleştirildikten sonra eğitim işlemine başlanmaktadır.

Plaka tespit modül veri setinde 1380 görsel bulunmaktadır. Bu veri setinden 897 görsel eğitim veri seti, 483 görsel ise test veri setine ayrılmıştır. Eğitim süresi yaklaşık 10 saat sürmüştür. Eğitim sonunda hata grafiği Şekil 4.6.'da gösterilmektedir.



Şekil 4.6. Plaka tespit eğitimi hata grafiği

Plaka tespit modül eğitimi 160000 tekrar ile gerçekleştirilmiştir. Eğitim sonucunda Şekil 4.6.'da görüldüğü gibi hata değeri 0.05'in altına düşmüştür.

Eğitimi gerçekleşen plaka tespit modülü, eğitilen modeli kaydetmektedir. Bu model proje ana uygulamasına entegre edilmiştir. Eğitilmiş model, sunulan giriş görüntüsünün plaka koordinatını vermektedir. Verilen plaka konum bilgileri doğrultusunda plaka tespit etme işlemi gerçekleştirilmektedir. Şekil 4.7.'de plaka tespit aşaması sonucu elde edilen görüntü gösterilmiştir.

Plaka tespit modulünden sağlanan plaka görüntüsüne yeniden boyutlandırma işlemi uygulanmaktadır. Tekrar boyutlandırılan görüntü, filtre işleminden geçirilerek karakter tespit modulüne sunulmaktadır. Plaka görselinin yeniden boyutlandırılması, karakter tespit modülünün karakter tespit başarımı için önemli bir parametredir. Karakter tespit modulüne sunulacak girdi 187x38 boyutuna getirilmektedir.

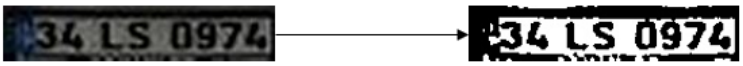
Şekil 4.7. Tespit edilen plaka

Karakter tespit modülünün eğitimde kullanılan görüntüdeki karakterlerin boyutları karakter tespit başarımını doğrudan etkilemektedir. Yeniden boyutlandırma işlemi için Python görüntüleme kütüphanesinden (Python Imaging Library, PIL) faydalanılmıştır [59]. Görüntü yeniden boyutlandırma işlemi için LANCZOS görüntü algoritması kullanılmıştır. Şekil 4.8.'de yeniden boyutlandırma işlemi gerçekleşmiş plaka görülmektedir.

Şekil 4.8. Plaka yeniden boyutlandırma

Yeniden boyutlanan plaka gri formata çevrilmektedir. Gri formata çevrilen görüntü sonraki adım olan filtreleme işlemine tabi tutulur. Filtreleme işlemi için Otsu eşik belirleme metodu kullanılmaktadır. Otsu metodu, gri seviye görüntüler üzerinde uygulanabilen bir eşik tespit yöntemidir. Otsu eşik belirleme metodu kullanımı için OpenCV-Python kütüphanesinden faydalanılmıştır. Şekil 4.9.'de yeniden boyutlandırılan plaka görselinin filtre işleminden geçtikten sonraki görüntüsü verilmektedir. Şekilden de görüldüğü gibi Otsu yöntemi ile eşik belirlendikten sonra

plakadaki karakterler siyah yapılarak ön plana çıkarılırken arka plandaki gri tonlar beyaz yapılmıştır.



Şekil 4.9. Görüntü filtre işlemi

Filtre işleminden geçirilen görüntü karakter tespit modülüne sunulmaktadır.

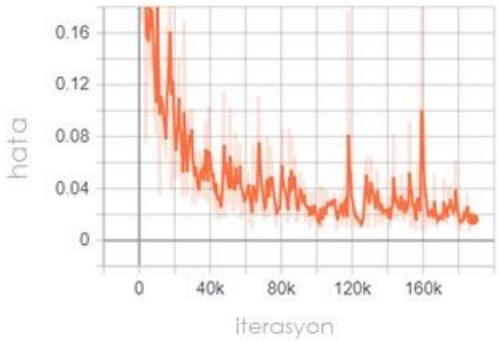
4.2.5. Karakter tespit modülü

Karakter tespit modül eğitiminin gerçekleştirilmesi için karakter, eğitim ve test veri setlerinin Bölüm 4.2.1.'de belirtilen aşamalarına tabi tutulur. Daha sonra Bölüm 4.2.2.'de belirtilen genel konfigürasyon ayarları karakter tespit işlemine özel yapılandırılır. Nesne etiket haritası Şekil 4.10'de gösterildiği gibi ayarlanmaktadır. Bu işlemler gerçekleştirildikten sonra eğitim işlemine başlanmaktadır.

```
item {
  id: 1
  name: 'character'
}
```

Şekil 4.10. Karakter tespit etiket haritası

Karakter tespit modül veri setinde 300 görüntü bulunmaktadır. Bu veri setinden 240 görüntü eğitim veri seti, 60 görüntü ise test veri setine ayrılmıştır. Eğitim süresi yaklaşık 6 saat sürmüştür. Eğitim sonunda hata grafiği Şekil 4.11.'de gösterilmektedir.



Şekil 4.11. Karakter tespit eğitimi hata grafiği

Plaka tespit eğitimi 160000 tekrar ile gerçekleştirilmiştir. Eğitim sonucunda Şekil 4.8.'de görüldüğü gibi hata değeri 0.05'in altına düşmüştür.

Eğitimi gerçekleşen karakter tespit modülü, eğitilen modeli kaydetmektedir. Bu model proje ana uygulamasına entegre edilmiştir. Eğitilmiş model, sunulan giriş görüntüsünün karakter bölge koordinatlarını geri dönmektedir. Geri dönülen karakter konum bilgileri doğrultusunda karakter tespit etme işlemi gerçekleştirilmektedir. Şekil 4.12.'de karakter tespit aşaması görülmektedir.

Şekil 4.12. Tespit edilen karakterler

Karakter tespit işleminden sonra hangi tanıma modülüne tahmin işlemini yaptıracağımızı belirtmemiz gerekmektedir. Bu işlem için plaka üzerindeki konum bilgileri ile rakam ya da harf olup olmadığı belirlenip buna uygun karakter tanıma modülüne verilmektedir.

Plaka üç kısma ayrılmıştır. Birinci kısımda rakam, ikinci kısımda harfler ve üçüncü kısımda rakamalar bulunmaktadır. Karater tespit işlemi yapıldığında bu durum göz

önüne alınarak tespit edilen karakter rakam veya harf olma durumuna göre etiketlenmektedir. Bu işlem Şekil 4.13.'de gösterilmektedir.

Şekil 4.13. Karakter bölümlendirme işlemi

4.2.6. Karakter tanıma modülü

Karakter tanıma modülü, çok katmanlı CNN mimarisi kullanılarak gerçekleşmektedir. Tespit edilen karakterler plaka konumuna göre etiketlenmekte ve bu etiketlere göre çok katlı CNN mimarisi ile eğitilmiş modüle sunulmaktadır.

Karakter tanıma aşaması, iki farklı modülden oluşmaktadır. İlk modül plaka üzerinde bulunan rakamları tanımak için kullanılır. Diğer modül ise plaka üzerinden bulunan harfleri tanıma işlemi için kullanılır. Her iki modül yapısında konvolüsyonel katmanlar Tablo 4.2.'de gösterildiği gibidir.

Tablo 4.2. Konvolüsyonel Katman Yapıları

Katman	TİP	FİLTRE	BOYUT/KAYDIRMA	ÇIKIŞ
1	Konvolüsyon	64	4 x 4 / 1	32 x 32 x 64
2	Maksimum Havuzlama		4 x 4 / 4	8 x 8 x 64
3	Konvolüsyon	128	2 x 2 / 1	8 x 8 x 128
4	Maksimum Havuzlama		4 x 4 / 4	2 x 2 x 128

Konvolüsyonel katmanlarda işleme tutulan görüntünün tam bağlantı katmanına bağlanabilmesi için düzleştirme uygulanır. Bu aşamada Tablo 4.2.'de belirtilen İkinci konvolüsyon katmanından sonraki maksimum havuzlama işleminin çıkışını düzleştirerek tam bağlantı katmanına hazır hale getirilir. Düzleştirme boyutu Tablo 4.3.'te gösterilmektedir.

Tablo 4.3. Konvolüsyon katman çıkışı düzleştirme işlemi

İşlem	TİP	BOYUT
1	Düzleştirme	512

Plakada bulunan sayıları tanıyan modülün tam bağlantı yapısı Tablo 4.4.'te belirtilmektedir. Görüldüğü gibi toplam üç tam bağlantı katmanından oluşmaktadır. Son katman nöron sayısı sınıf sayımızı belirtmektedir. Plakada 10 adet rakam bulunabileceğinden son katman çıkışımız bu şekilde tasarlanmıştır.

Tablo 4.4. Rakamlar için Tam Bağlantı Katman Yapısı

KATMAN	TİP	Giriş	Nöron Sayısı
1	Tam Bağlantı Katmanı	512	512
2	Tam Bağlantı Katmanı	512	256
3	Çıkarma/Dropout		
4	Tam Bağlantı Katmanı	256	10
5	Çıkarma/Dropout		
6	TAHMİN		

Plakada bulunan harfleri tanıyan modül tam bağlantı yapısı ise Tablo 4.5.'te verilmiştir. Plakada 23 adet harf bulunabileceğinden son katman çıkışımız bu şekilde tasarlanmıştır.

Tablo 4.5. Harfler için Tam Bağlantı Katman Yapısı

KATMAN	TİP	Giriş	Nöron Sayısı
1	Tam Bağlantı Katmanı	512	512
2	Tam Bağlantı Katmanı	512	256
3	Çıkarma/Dropout		
4	Tam Bağlantı Katmanı	256	23
5	Çıkarma/Dropout		
6	TAHMİN		

Tasarlanan plaka tanıma mimarisi eğitimi için Chars74k veri seti kullanılmıştır. Bu veri setindeki rakam ve büyük harfler plaka tanıma sistemi veri setinde kullanılmıştır. Türkiye Cumhuriyeti plakası üzerinde bulunabilecek karakterlerin büyük harf ve yalnızca Türkçe karakter olması göz önünde bulundurularak veri seti ayarlanmıştır. Veri seti ayarlandıktan sonra geliştirdiğimiz modül eğitime bırakılarak kaydedilmektedir. Kaydedilen eğitim modelimiz ana uygulaya entegre edilerek çalıştırılmaktadır. Karakter tespit modülünden gelen karakterter görseli, harf veya rakam bilgisine göre karakter tanıma aşamalarından geçirilmektedir.

BÖLÜM 5. TARTIŞMA VE SONUÇ

Bu tezde Tensorflow Nesne Algılama uygulamasından yararlanılarak otomatik lisanslı plaka tanıma işlemi gerçekleştirilmiştir. Plaka tanıma sistemi üç ana adımdan oluşmaktadır. Bu aşamalar sırasıyla plaka yerinin tespiti, plakada bulunan karakter yerlerinin tespiti ve plakada tespit edilen karakterlerin tanınması şeklinde verilebilir.

Önerilen sistemin testi için gerçek zaman çalışan bir kontrollü geçiş sisteminden alınmış 350 x 250 boyutlarında görüntüler kullanılmıştır. Geliştirilen yazılım yalnızca Türkiye Cumhuriyet plakalarını tanıyacak şekilde tasarlanmıştır. Program Python programlama dili kullanılmıştır. Ayrıca nesne algılama işlemi için Tensorflow Nesne Algılama API'si kullanılmıştır. Yazılımın gerçekleştiği ve test edildiği bilgisayar i5-8250 CPU 1.80 GHz işlemci ve 8 GB RAM özelliklerindedir.

Yukarıda belirtilen bilgisayar özelliklerinde, çalıştırılan programın bir araç görüntüsündeki tanıması için geçen toplam süre ve plaka tanınması için gerçekleşen üç ana aşamada geçen süreler Tablo 5.1.'de verilmektedir.

Tablo 5.1. Plaka tanıma işlemi için geçen süreler

ADIM	İŞLEM SÜRESİ
Plaka Tespiti	1.55 saniye
Plaka Karakterlerinin Tespiti	1.65 saniye
Plaka Karakterlerini Tanıma	2.05 saniye
Toplam Süre	5.25 saniye

Sistemin başarımı üç ana modülü baz alarak incelediğimizde: doğruluk bakımından plaka tespit başarımı %99.06, karakter tespit başarımı %95.03 ve karakter tanıma başarımı %90.03 olarak gözlenmektedir. Ancak genel ALPR sistem başarımı

modüler olarak elde edilen başarımdan oldukça düşük çıkmaktadır. Genel sistem başarımının düşük olmasının sebeplerinden bazıları şunlardır:

1. Tespit edilen plakanın yeniden boyutlandırma ve filtre işlemleri sonrasında deforme olması karakter tespit ve karakter tanıma modüllerini yanıltması.
2. Çevresel faktörler sebebiyle plakanın parlaklık ve görsel kalitesinin bozulmasıile plaka ve karakter tespit işlemini olumsuz etkilemesi.

Sonuç olarak üç ana modül üzerinden yapılan otomatik lisanslı plaka tanıma işleminin genel başarımı %70 doğrulukla elde edilmiştir. Bu doğruluk oranı tüm sisteme ait genel başarımdır. Başarımı yükseltmek için sonraki çalışmalarda yukarıda verilen sebepleri ortadan kaldırmaya veya görüntü işleme teknikleri kullanarak elimine etmeye yönelik ana adımlar gerçeklerştirilmesi hedeflenmektedir.

KAYNAKLAR

[1] Mohamed El-Adawi, Hesham Abd el Moneim Keshk ve Mona Mahmoud Haragi, Automatic License Plate Recognition, IEEE Transactions on Intelligent Transport Systems, Vol. 5, 42-53, 2004.

[2] R. Juntanasub ve N. Sureerattanan, Car License Plate Recognition through Hausdorff Distance Technique, Proceedings of the IEEE ICTAI'05, 607-612, 2005.

[3] M. Raus and L. Kreft, Reading Car License Plates by the Use of Artificial Neural Networks, Proceedings of the IEEE 38th Midwest Symposium on Circuits ve Systems, 538-541, 1995.

[4] L. Sirithinaphong ve K. Chamnongthai, Extracting of Car License Plates Using Motor Vehicle Regulation and Character Pattern Recognition, IEEE, 559–569, 1998.

[5] S. H. Park, K. I. Kim, K. Jung ve H. J. Kim, Locating Car License Plates Using Neural Networks, Electronics Letter, Vol. 35, No 17, 1475-1477, 1999.

[6] K. K. Kim, K. I. Kim, J. B. Kim ve H. J. Kim, Learning-Based Approach For License Plate Recognition, Proceedings of the IEEE Signal Processing Society Workshop, Vol. 2, 614-623, 2000.

[7] X. Jianfeng, L. Shaofa ve C. Zhibin, Color Analysis for Chinese Car Plate Recognition, Proceedings of the IEEE International Conference on Robotics, Intelligent Systems and Signal Processing, 1312-1316, 2003.

[8] A. Broumandnia ve M. Fathy, Application of Pattern Recognition for Farsi License Plate Recognition, ICGST, vol. 2, pp. 25-31, 2005.

[9] V. Ganapathy ve W. L. D. LUI, A Malaysian Vehicle License Plate Localization and Recognition System, Journal of Systemics, Cybernetics and Informatics, Vol. 6, No. 1, 2008.

[10] Naga Surya Sandeep Angara, "Automatic License Plate Recognition Using Deep Learning Techniques", University of Texas at Tyler, Electrical Engineering Theses, 2015.

[11] Muhammad Aasim Rafique, Witold Pedrycz, Moongu Jeon, Vehicle license plate detection using region-based convolutional neural networks**,** 22:6429–6440, 2018.

[12] Hogne Jørgensen, Automatic License Plate Recognition using Deep Learning Techniques, Norwegian University Science, Technology Department, Computer Science, Master Thesis, 2017.

[13] Wichai Puarungroj, Narong Boonsirisumpun, Thai License Plate Recognition Based on Deep Learning, 3rd International Conference on Computer Science and Computational Intelligence, Vol.135, 214–221, 2018.

[14] Yann LeCun, Yoshua Bengio ve Geoffrey Hinton, Deep Learning, Nature 521, 436-444, 2015.

[15] Osman Yakıt, Yılmaz Özkan, Kurumsal Kaynak Planlama Sistemlerinde Yapay Sinir Ağlarının Değerlendirilmesi Yaklaşımı, Siyaset, Ekonomi ve Yönetim Araştırmaları Dergisi, Cilt:5, Sayı:2, 2147-9035, 2017.

[16] Maad M. Mijwel, Yapay Sinir Ağlar Yapısı ve Fonksiyonu, Bağdat Üniversitesi, Bilgisayar Bilimleri, Eylül 2017.

[17] www.google.com., Erişim Tarihi: 01.05.2019.

[18] Sricastava, Nitish ve ark. "Dropout: a simple way to prevent neural networks from overfitting", JMLR, vol. 15, pp. 1929–1958, 2014.

[19] Y. LeCun, L. Bottou, Y. Bengio ve P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE, Nov. 1998.

[20] DC Cireşan, U Meier, LM Gambardella, J Schmidhuber, "Deep, big, simple neural nets for handwritten digit recognition", Neural computation 22 (12), 3207-3220, 2010.

[21] D. C. Ciresan, U. Meier, J. Masci, L. M. Gambardella ve J. Schmidhuber, "High performance neural neural networks for image classification.", Technical Report IDSIA-01-11, 2011.

[22] https://cs.stanford.edu/people/karpathy/, Erişim Tarihi: 01.05.2019.

[23] D. Zheng, Y. Zhao ve J. Wang, "An efficient method of license plate location" Pattern Recogn. Lett., vol. 26, pp. 2431–2438, Nov. 2005.

[24] C. Busch, R. Dörner, C. Freytag ve H. Ziegler, "Feature based recognition of traffic video streams for online route tracing," in Pathway to a Global Wireless Revolution. *CD-ROM*, pp. 1790–1794, Zentrum für Graphische Datenverarbeitung Darmstadt, 1998.

[25] F. Faradji, A. H. Rezaie ve M. Ziaratban, "A morphological-based license plate location," IEEE, 2007.

[26] A. N. S., Rasheed ve I. O., "Automated number plate recognition using hough lines and template matching," in Proc. World Cong. Engin. Comp. Sci., pp. 199–203, 2012.

[27] S. Kim, D. W. Kim ve H. J. Kim, "A recognition of vehicle license plate using a genetic algorithm based segmentation," in *ICIP (2)*, pp. 661–664, 1996.

[28] W. Jia, H. Zhang ve X. He, "Region-based license plate detection," J. Netw. Comput. Appl., vol. 30, pp. 1324–1333, Nov. 2007.

[29] C. N. Anagnostopoulos, I. E. Anagnostopoulos, V. Loumos, ve E. Kayafas, "A license plate-recognition algorithm for intelligent transportation system applications.", IEEE Intel Transp. Syst., pp. 377-39, 2008.

[30] K. Deb, H.-U. Chae ve K.-H. Jo, "Vehicle license plate detection method based on sliding concentric windows and histogram.," *JCP*, vol. 4, no. 8, pp. 771–777, 2009.

[31] H. Zhang, W. Jia, X. He ve Q. Wu, "Learning-based license plate detection using global and local features," in Proceedings of the 18th International Conference on Pattern Recognition - Volume 02, ICPR '06, (Washington, DC, USA), pp. 1102–1105, IEEE Computer Society, 2006.

[32] H. Li ve C. Shen, "Reading car license plates using deep convolutional neural networks and lstms," CoRR, vol. abs/1601.05610, 2016.

[33] K.-H. Lin, H. Tang ve T. S. Huang, "Robust license plate detection using image saliency.," in ICIP, pp. 3945–3948, IEEE, 2010.

[34] F. Porikli ve T. Kocak, "Robust license plate detection using covariance descriptor in a neural network framework", in Proc. IEEE Int. Conf. Video Signal Based Surveillance, p. 107, 2006.

[35] K. K. Kim, K. Kim, J. Kim ve H. J. Kim, "Learning-based approach for license plate recognition", Proceedings of the IEEE Signal Processing Society Workshop, Vol. 2, pp. 614–623, 2000.

[36] T. Nukano ve M. Khalid, ""vehicle license plate character recognition by neural networks", in Proc. Int. Symp. Intell. Signal Process. Commun. Syst., pp. 771–775, 2004.

[37] J. Jiao, Q. Ye ve Q. Huang, "A configurable method for multi-style license plate recognition," Pattern Recognition, vol. 42, no. 3, pp. 358 – 369, 2009.

[38] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, ve L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009.

[39] A. Krizhevsky, I. Sutskever ve G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in Advances in Neural Information Processing *Systems 25* (F. Pereira, C. J. C. Burges, L. Bottou ve K. Q. Weinberger, eds.), Curran Associates, Inc., pp. 1097–1105, 2012.

[40] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke ve A. Rabinovich, "Going deeper with convolutions," in Computer Vision and Pattern Recognition *(CVPR)*, 2015.

[41] K. Simonyan ve A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[42] K. He, X. Zhang, S. Ren ve J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015.

[43] A. Veit, M. J. Wilber and S. J. Belongie, "Residual networks are exponential ensembles of relatively shallow networks," *CoRR*, vol. abs/1605.06431, 2016.

[44] Q. Liao ve T. A. Poggio, "Bridging the gaps between residual learning, recurrent neural networks and visual cortex," *CoRR*, vol. abs/1604.03640, 2016.

[45] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn ve A. Zisserman, "The pascal visual object classes challenge: A retrospective," International Journal of Computer Vision, vol. 111, pp. 98–136, Jan. 2015.

[46] T. B. Dalal, N., "Histograms of oriented gradients for human detection," pp. 886–893, 2005.

[47] P. F. Felzenszwalb, R. B. Girshick, D. McAllester ve D. Ramanan, "Object detection with discriminatively trained part-based models" IEEE Trans. Pattern Anal. Mach. Intell., vol. 32, pp. 1627–1645, Sept. 2010.

[48] R. Girshick, J. Donahue, T. Darrell and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in Computer Vision and Pattern Recognition, 2014.

[49] I. Endres ve D. Hoiem, "Category independent object proposals," in Proceedings of the 11th European Conference on Computer Vision*:* Part V, ECCV'10, (Berlin, Heidelberg), pp. 575–588, Springer-Verlag, 2010.

[50] D. Hoiem, A. A. Efros ve M. Hebert, “Recovering occlusion boundaries from an image,” Int. J. Comput. Vision, vol. 91, pp. 328–346, Feb. 2011.

[51] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, ve A. W. M. Smeulders, “Selective search for object recognition,” International Journal of Computer Vision, vol. 104, no. 2, pp. 154–171, 2013.

[52] R. Girshick, “Fast R-CNN,” in International Conference on Computer Vision *(ICCV)*, 2015.

[53] S. Ren, K. He, R. Girshick ve J. Sun, “Faster R-CNN: Towards real-time object detection with region proposal networks,” in Advances in Neural Information Processing Systems (NIPS), 2015.

[54] K. He, G. Gkioxari, P. Dollár ve R. B. Girshick, “Mask R-CNN,” *CoRR*, vol. abs/1703.06870, 2017.

[55] J. Redmon, S. K. Divvala, R. B. Girshick ve A. Farhadi, “You only look once: Unified, real-time object detection,” CoRR, vol. abs/1506.02640, 2015.

[56] J. Redmon ve A. Farhadi, “Yolo9000: Better, faster, stronger,” arXiv preprint arXiv:1612.08242, 2016.

[57] https://github.com/EdjeElectronics/TensorFlow-Object-Detection-API-Tutorial-Train-Multiple-Objects-Windows-10, Erişim Tarihi: 01.02.2019.

[58] https://tzutalin.github.io/labelImg/, Erişim Tarihi: 04.02.2019.

[59] https://pillow.readthedocs.io/en/stable/, Erişim Tarihi: 01.02.2019.

ÖZGEÇMİŞ

İsmail DEMİRCİ, 09.04.1990'da İstanbul'da doğdu. İlk, orta ve lise eğitimini İstanbul'da tamamladı. 2009 yılında Tarık Buğra Lisesi'nden mezun oldu. 2009 yılında başladığı Dumlupınar Üniversitesi Elektrik-Elektronik Mühendisliği Bölümü'nü 2013 yılında bitirdi. 2016 yılında Sakarya Üniversitesi Elektrik-Elektronik Mühendisliği Bölümü'nde yüksek lisans eğitimine başladı. Özel sektörde AR-GE departmanında gömülü sistem mühendisi olarak çalışmaktadır.