T.C.
NECMETTİN ERBAKAN
ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YOL HIZ LİMİTİ TABELALARININ DERİN ÖĞRENME KULLANARAK GERÇEK ZAMANLI TESPİTİ

Fatih Kamil KOÇAK

YÜKSEK LİSANS TEZİ

Elektrik - Elektronik Mühendisliği Anabilim Dalı

Nisan - 2025

KONYA

TEZ KABUL VE ONAYI

Fatih Kamil KOÇAK tarafından hazırlanan "*Yol Hız Limiti Tabelalarının Derin Öğrenme Kullanarak Gerçek Zamanlı Tespiti*" adlı tez çalışması 24/04/2025 tarihinde aşağıdaki jüri tarafından oy birliği / oy çokluğu ile Necmettin Erbakan Üniversitesi Fen Bilimleri Enstitüsü Elektrik - Elektronik Mühendisliği Anabilim Dalı'nda YÜKSEK LİSANS TEZİ olarak kabul edilmiştir.

Jüri Üyeleri	**İmza**
Başkan	
Doç. Dr. Hasan KOYUNCU	_____
Danışman	
Dr. Öğr. Üyesi Sabri ALTUNKAYA	_____
Üye	
Doç. Dr. Hakkı SOY	_____

Fen Bilimleri Enstitüsü Yönetim Kurulu'nun ..../ .../20.. gün ve ..... sayılı kararıyla onaylanmıştır.

Prof. Dr. Havvanur UÇBEYİAY
FBE Müdürü

TEZ BİLDİRİMİ

Bu tezdeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edildiğini ve tez yazım kurallarına uygun olarak hazırlanan bu çalışmada bana ait olmayan her türlü ifade ve bilginin kaynağına eksiksiz atıf yapıldığını bildiririm.

DECLARATION PAGE

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

_____

Fatih Kamil KOÇAK

Tarih: 24/04/2025

ÖZET

YÜKSEK LİSANS TEZİ

YOL HIZ LİMİTİ TABELALARININ DERİN ÖĞRENME KULLANARAK GERÇEK ZAMANLI TESPİTİ

Fatih Kamil KOÇAK

Necmettin Erbakan Üniversitesi Fen Bilimleri Enstitüsü

Elektrik - Elektronik Mühendisliği Anabilim Dalı

Danışman: Dr. Öğr. Üyesi Sabri ALTUNKAYA

2025, 48 Sayfa

Jüri

Doç. Dr. Hasan KOYUNCU

Dr. Öğr. Üyesi Sabri ALTUNKAYA

Doç. Dr. Hakkı SOY

Gelişen teknoloji ile birlikte otomotiv endüstrisi de büyük bir gelişme göstermiştir. Otomobillerde gün geçtikçe daha çok elektronik sistem sürüş konforu ve sürüş güvenliğini arttırmak amacıyla kullanılmaya başlanmıştır. ADAS (advanced driving assistance systems – gelişmiş sürücü destek sistemleri) sürücünün görevlerinin bazılarını devralarak üzerindeki yükü azaltmak ve daha güvenli ve konforlu sürüş sağlamayı amaçlayan sistemlerdir. Bu gelişmiş sistemlerden bir tanesi de yol hız limitlerinin belirlenmesi için kullanılan sistemlerdir.

Yol hız limiti tabelalarının belirlenmesi temel olarak trafik tabelalarının tanınması probleminin bir alt kümesidir. Trafik tabelalarının tanınması iki temel işlemden oluşur. Öncelikle görüntü içerisinde trafik işaret tabelasının olduğunun tespiti, daha sonra bu tespit edilen işaretin sınıflandırılması yapılır. Tespit aşaması renk ve şekil analizine dayalı yöntemler kullanırken, sınıflandırma aşaması ise şablon eşleştirme, şekil bilgisi ve makine öğrenmesi tabanlı yöntemler kullanabilmektedir.

Bu tez çalışmasında yol hız limitlerinin belirlenmesi için yapay sinir ağları ve derin öğrenmeye dayalı bir uygulama geliştirilmiştir. Eğitim için hali hazırda var olan veri setlerinin yanı sıra hem yeni veriler toplayarak hem de veri seti üzerinde sentetik veri arttırma işlemleri uygulanarak amaca uygun daha gelişmiş bir veri seti elde edilmiştir. Ayrıca sistem başarısını arttırmak için ilgi alanı belirlemeye dayalı teknikler uygulanmıştır. Tüm bunlar ile sistem başarı oranı iyileştirilmiş ve yol üzerindeki hız limiti tabelalarının gerçek zamanlı olarak tespit edilip yoldaki hız limitinin belirlenmesi sağlanmıştır.

Anahtar Kelimeler: ADAS, Derin Öğrenme, Trafik İşareti Tanıma, Yapay Sinir Ağları

ABSTRACT

MS THESIS

REAL-TIME DETECTION OF ROAD SPEED LIMIT SIGNS USING DEEP LEARNING

Fatih Kamil KOÇAK

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCE OF NECMETTİN ERBAKAN UNIVERSITY

THE DEGREE OF MASTER OF SCIENCE / ELECTRICAL AND ELECTRONIC ENGINEERING

Advisor: Asst. Prof. Dr. Sabri ALTUNKAYA

2025, 48 Pages

Jury

Assoc. Prof. Dr. Hasan KOYUNCU

Asst. Prof. Dr. Sabri ALTUNKAYA

Assoc. Prof. Dr. Hakkı SOY

With the developing technology, the automotive industry has also shown great development. More and more electronic systems have been used in automobiles to increase driving comfort and driving safety. ADAS (advanced driving assistance systems) are systems that aim to reduce the burden on the driver by taking over some of his duties and to provide safer and more comfortable driving. One of these advanced systems is the systems used to determine road speed limits.

Determining road speed limit signs is basically a subset of the problem of recognizing traffic signs. Recognizing traffic signs consists of two basic operations. First, determining that there is a traffic sign in the picture, then classifying this detected sign. While the detection stage uses methods based on color and shape analysis, the classification stage can use methods based on template matching, shape information and machine learning.

In this thesis, an application based on artificial neural networks and deep learning has been developed to determine road speed limits. In addition to the existing data sets for training, a more advanced data set suitable for the purpose has been obtained by both collecting new data and applying synthetic data augmentation operations on the data set. In addition, techniques based on determining the region of interest were applied to increase the system success. With all these, the system success rate was improved and the speed limit signs on the road were detected in real time and the speed limit was determined.

Keywords: ADAS, Artificial Neural Network, Deep Learning, Traffic Sign Recognition

ÖNSÖZ

Çalışmalarımda, yardımlarını hiçbir zaman esirgemeyen tez danışmanım Dr. Öğr. Üyesi Sabri ALTUNKAYA'ya, bana her türlü kolaylığı sağlayan tüm hocalarıma teşekkürü bir borç bilirim.

Ayrıca bugünlere gelmemde en büyük pay sahibi olan, beni her konuda daima destekleyen eşime ve sevgili aileme teşekkür ederim.

Fatih Kamil KOÇAK

KONYA-2025

İÇİNDEKİLER

ŞEKİLLER LİSTESİ

Şekil **Sayfa**

ÇİZELGELER LİSTESİ

SİMGELER VE KISALTMALAR

Kısaltmalar

ABS: Anti-lock Brake System - Kilitlenme Önleyici Fren Sistemi

ACC: Active Curuise Control - Aktif Hız Sabitleyici

ADAS: Advanced Driving Assistance System - Gelişmiş Sürücü Destek Sistemi

ANN: Artificial Neural Network - Yapay Sinir Ağı

CNN: Convolutional Neural Network - Evrişimsel Sinir Ağı

CPU: Central Processing Unit - Merkezi İşlem Birimi

ÇKP: Çok Katmanlı Perceptron

DBC: Dynamic Brake System - Dinamik Fren Sistemi

EBD: Elektronic Brakeforce Distubition - Elektronik Fren Dağıtımı

ECU: Enigine Control Unit - Motor Kontrol Ünitesi

ECT: Electronically Controlled Transmission - Elektronik Kontrollü Şanzıman

ELU: Exponential Linear Unit - Üstel Lineer Birim

GPIO: General Purpose Input Output - Genel Amaçlı Giriş Çıkış

GPU: Graphics Preocessor Unit - Grafik İşlem Birimi

GTSDB: German Traffic Sign Detection Benchmark - Alman Traifk İşaret Algılama Kıyaslaması

HDMI: High Definition Multimedia Interface - Yüksek Çözünürlüklü Multimedya Arayüzü

HSV: Hue Saturation Value - Renk Özü Doygunluk Parlaklık

I2C: Inter Integrated Circuit - Tümdevreler Arası

I2S: Inter IC-Sound - Tümdevreler Arası Ses

LDW: Lane Departure Warning - Şerit Terk Uyarısı

LPDDR: Low Power Double Data Rate - Düşük Güç Çift Veri Hızı

mAP: Mean Average Precision - Genel Ortalama Kesinlik

MIPI: Mobile Industry Processor Interface - Mobil Endüstri İşlemci Arayüzü

MLP: Multi Level Perceptron - Çok Katmanlı Perceptron

RELU: Rectified Linear Unit - Doğrultulmuş Lineer Birim

RGB: Red Green Blue - Kırmızı Yeşil Mavi

RAM: Random Access Memory - Rastgele Erişim Bellek

SBC: Single Board Computer - Tek Kart Bilgisayar

SDK: Software Development Kit - Yazılım Geliştirme Kiti

SPI: Serial Peripheral Interface - Seri Çevre Arayüzü

SSD: Single-Shot Detector - Tek Çekim Dedektörü

TSR: Traffic Sign Recognition - Trafik İşareti Tanıma

UART: Universal Asynchronous Receiver-Transmitter - Evrensel Asenkron Alıcı / Verici

USB: Universal Serial Bus - Evrensel Seri Veriyolu

1. GİRİŞ

Gelişen teknoloji ile birlikte otomotiv endüstrisi de büyük bir gelişme göstermiştir. Otomobillerde gün geçtikçe daha çok elektronik sistem kullanılmaya başlanmıştır. Bu elektronik sistemlerden bazıları: ECU(motor kontrol ünitesi) ECT (electronically controlled transmission) gibi ana sistemler, ABS (anti-lock brake system) DBC (dynamic brake system) EBD (electronic brakeforce distrubition) gibi güvenlik sistemleri ile birlikte ACC (active cruise control) otomatik park sistemi, kör nokta monitörü, çarpışma önleme sistemi gibi sürücü destek sistemlerinden oluşmaktadır.

ADAS (advanced driving assistance systems – gelişmiş sürücü destek sistemleri) sürücünün görevlerinin bazılarını devralarak üzerindeki yükü azaltmak ve daha güvenli ve konforlu sürüş sağlamayı amaçlayan sistemlerdir. Bu sistemler gelişmiş sensörler ve kontrol metotları kullanarak sürücüye yardımcı olurlar.

Araç sürmek büyük dikkat ve sorumluluk gerektirir. Bir kaza durumunda en iyi ihtimalle maddi kayıplar en kötü durumda da can kayıpları olabilmektedir. Yol, trafik durumu ve yakın çevre ile ilgili bilgileri vermek, yasak ve kısıtlamalar ile ilgili bilgiler vermek ve trafik düzen ve güvenliğini sağlamak amacıyla yollara trafik işaretleri yerleştirilmiştir (Resmî Gazete, 1985). Araç kazalarının büyük bir çoğunluğu 1.1'de görüldüğü gibi sürücü hatalarında kaynaklanmaktadır (Türkiye İstatistik Kurumu, 2022). Sürücü destek sistemleri ile sürücü hatalarını azaltarak daha güvenli bir sürüş amaçlanmaktadır.

Çizelge 1.1. 2022 yılı yaralanmalı veya ölümlü trafik kazaları (Türkiye İstatistik Kurumu, 2022)

Sürücü kusurları - Driver faults	204.233
Alkollü araç kullanmak	1.802
Araç hızını yol, hava ve trafiğin gerektirdiği şartlara uydurmamak	75.287
Arkadan çarpmak	14.108
Aşırı hızla araç kullanmak	1.806
Doğrultu değiştirme (dönüş) kurallarına uymamak	14.177
Geçme yasağı olan yerlerden geçmek	1.491
Kavşaklarda geçiş önceliğine uymamak	26.737
Kırmızı ışık veya görevlinin dur işaretine uymamak	5.223
Kurallara uygun olarak park etmiş araçlara çarpmak	2.962
Manevraları düzenleyen genel şartlara uymamak	17.072
Şerit izleme ve değiştirme kuralına uymamak	2.974
Taşıt giremez işareti bulunan yerlere girmek	5.488
Sürücünün diğer kusurlu halleri	35.106

Hız limitlerinin araç içerisinde gösterilmesi ile sürücüler dalgınlık yüzünden ya da yolun başka bir bölümüne odaklandıkları için kaçırdıkları tabelalardaki hız limitini devamlı olarak takip edebilir durumda olacaktır. Bu limit değeri araç ön konsolda veya ön cama yansıtılan HUD (head up display – uyarı ekranı) kısmında gösterilmektedir. Ayrıca bu hız limitleri aşıldığında uyarı vererek sürücüyü ikaz ederek güvenli bir sürüş sağlayabilmektedir.

Yolun mevcut bölümündeki hız limitlerinin tespiti iki şekilde yapılabilmektedir. Bunlardan ilki navigasyon sistemine entegre olarak çalışan çevrimiçi ya da çevrimdışı veritabanı ile sağlanan limit bilgileridir. Bu tip sistemler önceden belirlenmiş ve bir veri tabanında saklanmış veriler içerisinden GPS sisteminden alınan koordinat bilgilerini kullanarak mevcut limit bilgisinin çekilmesi ile olmaktadır. Aracın konumu değiştikçe uygun limit bilgisi veritabanından alınarak sürücüye gösterilmektedir. Bu veritabanı çevrimiçi olabileceği gibi çevrimdışı da olabilmektedir.

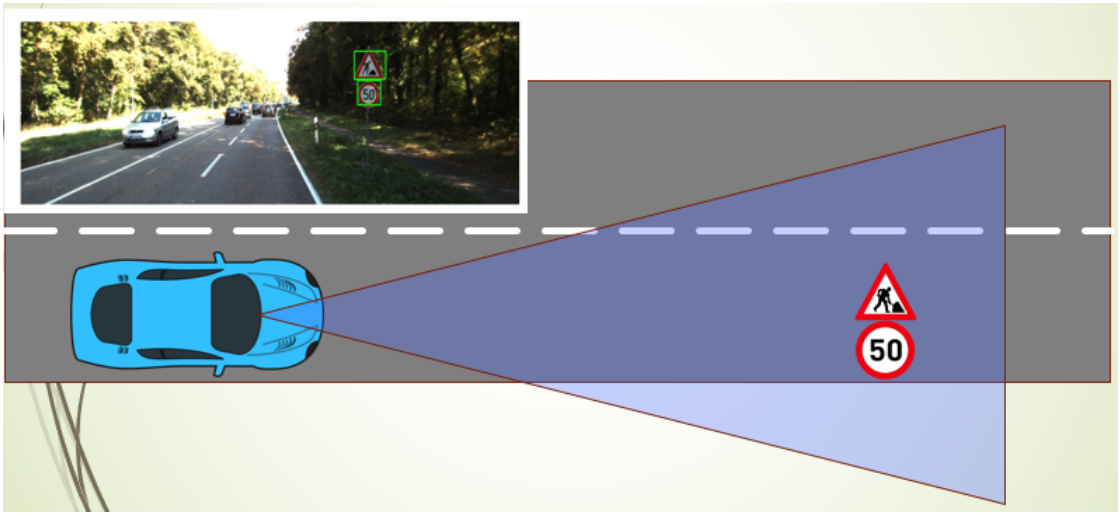
Hız limitlerinin saklandığı veritabanı çevrimiçi olduğu durumda bu servisin kullanılabilmesi için sürekli bir internet bağlantısına ihtiyaç duyulacaktır. İnternet bağlantısı sağlanamayan bölgelerde mevcut hız limitleri alınamayacak ve bu sistem çalışmayacaktır. Bunun önlenmesi için bu veri tabanı çevrimdışı olarak da saklanabilir. Böylece sabit bir internet bağlantısı olmasa dahi gerekli limit değerlerine ulaşılabilecektir. Bunun için araç içerisinde bu iş için ayrılmış bir depolama alanına ihtiyaç duyulacaktır. Bu yöntem her ne kadar verilerin sürekliliğini sağlıyor olsa da güncelliğini garanti edememektedir. Verilerin çevrimdışı saklanması yollardaki herhangi bir değişiklik durumunda hatalı limitlerin gösterilmesine neden olabilir. Ayrıca tüm yolların verilerinin çevrimdışı olarak saklanması pek mümkün olamayacağından bu veritabanının kapsamı yerel verilerle (tüm dünya yerine belirli bir şehir ya da ülke) sınırlandırılması olasıdır. Bu da mevcut konum için her zaman doğru verilere erişilememesi anlamına gelir. Bu sorunların önlenmesi için çevrim içi ve çevrim dışı haritalar birlikte kullanılarak internet bağlantısı olamayan durumlar için yakın çevrenin bilgileri çevrim dışı olarak tutularak internet bağlantısı sağlandığı zaman veriler güncellenerek sürekli doğru ve güncel verilere ulaşım sağlanabilir.

Her iki durumda da yoldaki hız limiti bilgilerinin önceden toplanıp bir veritabanına kaydedilmiş olması gerekmektedir. Ayrıca bu bilgilerin güncel tutulması da elzemdir. Karayollarında zaman zaman yolların özellikleri değişmekte ya da belirli durumlarda uygulanan hız limitlerinde değişiklik olabilmektedir. Bunların haricinde yol

çalışmaları veya geçici durumlar için belirli bir süre için geçici tabelalar yerleştirilerek yolun belirli bölümlerinde farklı hız limitleri uygulanabilmektedir. Böyle bir sistemde bu gibi durumların da takip edilerek verilerin güncelliği sağlanması gerekmektedir.

Yoldaki hız limitlerinin araç içerisinde tespiti için kullanılan diğer bir yöntem ise araçta bulunan bir kamera yardımı ile yoldaki hız limiti tabelalarının izlemesi ve karşılaşılan tabelalarda belirtilen limitlerin sürücüye gösterilmesi şeklinde yapılmaktadır. Bu yöntem aktif bir internet bağlantısı gerektirmediği gibi bilgilerin güncelliği ile ilgili de bir problem barındırmamaktadır. Veriler gerçek zamanlı olarak tespit edildiği için her zaman en güncel haliyle hız limitleri tespit edilmiş olacaktır. Sistemin çalışması için önceden bir veri toplama işlemi de gerekmeyecektir.

Bu tespit yöntemi için aracın ön tarafında yolu sürekli izleyen bir kamera bulunmaktadır. Bu kamera ile alınan canlı görüntü çeşitli teknikler kullanılarak analiz edilip hız limiti tabelası aranmakta ve bulunduğu taktirde bu tabeladaki limit bilgisi sürücü ile paylaşılmaktadır. Burada kameradan alınan görüntüyü işleyecek bir donanım bulunması gerekmektedir. Navigasyon ile tespit yöntemine göre kamera ve görüntü işleme donanımı maliyet açısından dezavantajlı olabilmektedir. Kullanılacak kameraların yüksek çözünürlüklü ve yüksek çerçeve hızına ve araç farlarından etkilenmeyen gece görüş yeteneğine sahip olması gerekmektedir. Ayrıca görüntü işlemenin yapılacağı donanımların da gelen görüntüyü gerçek zamanlı olarak işleyebilecek kapasitede olmaları gerekmektedir.



Şekil 1.1. Trafik işaret tanıma sistemi

Örnek bir çalışma olarak Opel firmasının yeni geliştirdiği Opel Eye teknolojisi incelenmiştir (Opel, 2023). Bu sistemde ön cam ile dikiz aynası arasına yerleştirilmiş,

geniş açılı lensi olan, yüksek çözünürlüğe sahip, saniyede 30 kare görüntü yakalayabilen ve boyutları cep telefonu kadar olan bir kamera kullanılmıştır. Elde edilen görüntü, sistemde bulunan iki adet işlemciye aktarılmaktadır. İşlemcilerden birinin üzerinde yol çizgilerini takip eden ve araç şeridinden kaymaya başladığında sürücüyü sesli ve görsel uyaran LDW (Şerit Terk Uyarısı - Lane Departure Warning) uygulaması çalışmaktadır. Diğer işlemci üzerinde ise 100 metre mesafe içinde, hız limitlerini gösteren trafik levhalarını tanıyan, otomobilin hızı limit üzerinde ise veya hız limiti değişmiş ise sürücüyü haberdar eden ve ayrıca sollama yapılmaz işaretini de tanıyarak sürücüyü uyaran TSR (Trafik İşareti Tanıma - Traffic Sign Recognition) uygulaması çalışmaktadır. İki işaret aynı anda tanınır ise öncelikli olarak sollama yapılmaz işareti sürücüye uyarı olarak gösterilmektedir. Tanınan hız limitinin araç içerisinde sürücüye gösterildiği ekranlar ve kameranın görüş açısı Şekil 1.2'de gösterilmektedir.

Şekil 1.2. Opel MobilEye sistemi (Opel, 2023)

2. YOLDAKİ HIZ LİMİTLERİNİN TESPİT EDİLMESİ

Yoldaki hız tabelalarının tespiti işlemi iki temel aşamadan oluşmaktadır. Alınan görüntü ilk önce bir hız limiti tabelasının varlığını tespit etmek için taranır. Görüntü içerisinde bir tabela tespit edildiğinde bu tabelanın bulunduğu alan sınıflandırma işlemine tabi tutulur. Bu sınıflandırmanın çıktısı da yolun mevcut kısmındaki hız limitini bize verir.

Bu tespit için iki temel yaklaşım bulunmaktadır. Birinci yaklaşım görüntü işleme tekniklerine dayalı yaklaşım, ikincisi ise derin öğrenme temeline dayalı yaklaşımdır. Her iki yaklaşım da girdi olarak kameradan alınan görüntünün piksel verilerini kullanır. Çıktı olarak da bir sınıf bilgisi verir.

2.1. Görüntü İşleme Temelli Yöntem

Görüntü işleme temelli yaklaşımda kameradan alınan görüntü üzerinde çeşitli matematiksel işlemler uygulanarak öznitelik çıkarımı yapılır. Daha sonra bu öznitelikler değerlendirilerek sınıf bilgisine ulaşılır.

2.2. Makine Öğrenmesi Temelli Yöntem

Makine öğrenmesi özel olarak programlanmadan, deneyim yoluyla öğrenmesini sağlayan bir yapay zeka alt dalıdır. Girdilere karşılık gelen çıktıları üretmek için katı kurallara ihtiyaç duymazlar. Bunun yerine belirli yaklaşımlar ile kestirimler yaparak daha önce hiç karşılaşmadığı girdiler için kodun içinde bulunmamasına rağmen çıktı üretebilirler.

(Mitchell, 1997) tanımlamasına göre: “Bir bilgisayar programı, T görevindeki, P performansını E tecrübesi ile artırıyorsa, bu bilgisayar programı, T görevini, P performans ölçüsüne göre E deneyimlerinden öğrenerek yerine getiriyor demektir.”

Bu tanıma göre bir T görevini yerine getirmek için E deneyimlerini kullanarak P performansını artıran programlar, öğrenen programlar demektir. Öğrenme işlemi 3 farklı biçimde olabilmektedir:

- Denetimli Öğrenme (Supervised learning)

- Denetimsiz Ögrenme (Unsupervised learning)
- Pekiştirmeli Öğrenme (Reinforcement learning)

2.2.1. Denetimli öğrenme

Bu öğrenme çeşidinde muhtemel girdiler ve bu girdilere karşılık gelen doğru çıktılar sisteme birlikte verilir. Sistem elindeki bu veri setine göre öğrenme işlemini gerçekleştirir. Ancak öğrenme bittikten sonra daha önce karşılaşmadığı girdilere karşılık gelen doğru çıktıları tahmin edebilme yeteneğine kavuşmuş olur (Haykin, 1999).

2.2.2. Denetimsiz öğrenme

Bu öğrenme şeklinde sisteme yalnızca girdiler verilir. Karşılık gelen çıktılar verilmeden sistemin girdiler arasındaki ilişkilerin bulunması beklenir. Daha çok kümeleme problemlerinde kullanılan bir yöntemdir. Eğitim işlemi bittikten sonra çıktı kümelerinin ne anlama geldiği belli değildir. Bunun anlamlandırılması ve etiketlendirilmesi kullanıcı tarafından yapılmalıdır (Haykin, 1999).

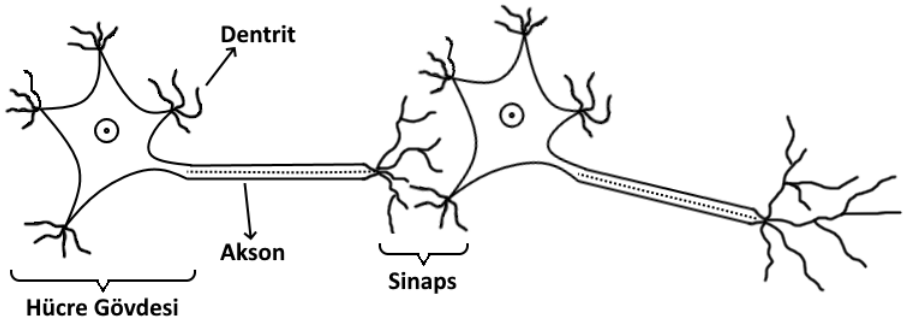
2.2.3. Pekiştirmeli öğrenme

Bu öğrenme şeklinde sisteme girdiler ile birlikte doğru çıktılar verilmez. Bunun yerine sistemin kendince bir tahminde bulunması beklenir. Tahmin gerçekleştikten sonra tahminin doğru ya da yanlış olduğu sisteme verilerek öğrenmesi sağlanır (Haykin, 1999).

2.3. Yapay Sinir Ağları

Makine öğrenmesinin bir alt dalı olarak yapay sinir ağları geliştirilmiştir. Temelde biyolojik sinirlerden esinlenerek oluşturulmuştur. İnsan beynindeki sinirlerin öğrenme kabiliyetin taklit ederek öğrenme gerçekleştirmeyi amaçlar. Yapı olarak tıpkı beyindeki sinir hücreleri gibi yapay sinir hücrelerinden oluşurlar. Sinir hücrelerine nöron adı verilir. Nöronlar üç kısımdan oluşurlar. Hücre gövdesi çekirdek sitoplazma ve organellerden oluşur. Hücre gövdesinden çıkan uzantılar vardır. Dendritler gövdeden çıkan kısa uzantılardır. Bir veya birden fazla olabilmektedir. Akson gövdeden çıkan

uzun uzantıdır. Yalnız bir tane akson vardır. Dendritlerden alınan sinyalleri başka nöronlara akson aracılığı ile taşır. Nöronların birleşme noktalarına sinaps adı verilir. Doğal nöronlar, sinyalleri dendritlerde bulunan sinapslar aracılığıyla alır. Alınan sinyaller yeterince güçlü olduğunda (belirli bir eşiği aştığında), nöron aktive olur ve aksondan bir sinyal yayar. Bu sinyal başka bir sinapsa gönderilebilir ve diğer nöronları aktive edebilir.

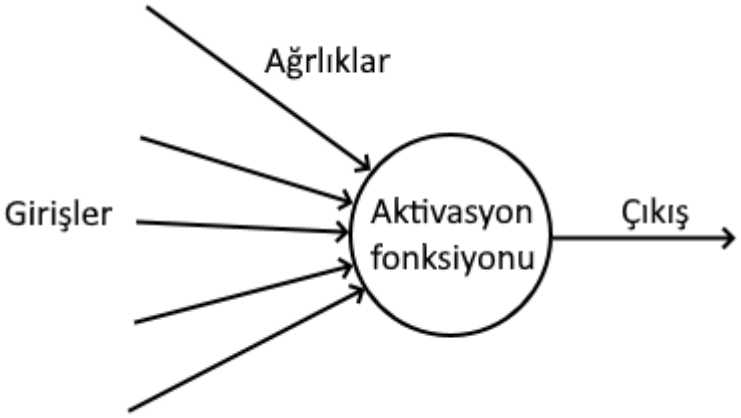


Şekil 2.1. Doğal nöron

Yapay nöronlar da benzer bir yapıda modellenmişlerdir. Her nöronun bir veya birden fazla girdisi ve bir adet çıktısı vardır. Bir nöronun çıktısı bir veya birden fazla başka nöronun girdisi olabilir. Her girdinin bir ağırlığı bulunur. Nörona girerken bu katsayı ile çarpılarak işleme alınır. Her nöronun bir de aktivasyon fonksiyonu vardır. Tüm girdiler toplandıktan sonra çıktı oluşturmak için aktivasyon fonksiyonu kullanır. Aktivasyon fonksiyonu nöronlara doğrusal olmayan bir özellik katmak için kullanılır (Haykin, 1999). Bu yapay nörondan perceptron (algılayıcı) oluşturulmuştur. Perceptron yapay sinir ağının temel bileşenidir. Perceptronlar doğrusal bir fonksiyonla iki parçaya ayrılabilen problemlerde kullanılabilir. Örnek olarak AND, OR, NAND gibi lojik fonksiyonlar gösterilebilir. Ancak XOR gibi bir fonksiyonu tek katmanlı perceptron ile gerçekleştirmek mümkün değildir (Haykin, 1999). Bir perceptron x adet girişe sahip ise her bir giriş için bir adet ağırlık olmak üzere x adet ağırlığa sahiptir. Bu x adet girişin w ağırlıkları toplamını hesaplar ve eşik değerleri ekleyerek aktivasyon fonksiyonundan geçirerek y çıktısını üretir (Denklem 2.1).

$$y = f(\sum(w.x) + b) \ . \tag{2.1}$$

Burada b eşik değeri, f aktivasyon fonksiyonu ve w ağerılıkları gösterir. Yapay sinir ağlarında kullanılan başlıca aktivasyon fonksiyonları aşağıda açıklanmıştır.

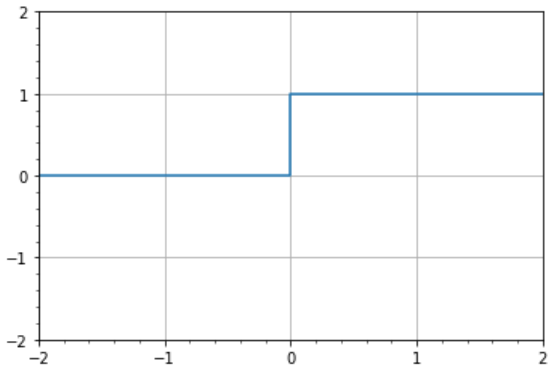


Şekil 2.2. Yapay nöron

2.3.1. İkili basamak fonksiyonu (Binary step function)

Giriş değerine göre yalnızca bir ve sıfır üreten aktivasyon fonksiyonudur. Sıfırdan küçük değerler için sıfır, sıfırdan büyük değerler için bir üretir. Eğer eşik değeri girilir ise eşik değerinden küçük değerler için sıfır, büyük değerler için bir üretir (Sharma vd., 2017). Denklem 2.2'de ikili basamak fonksiyonunun matematiksel ifadesi, Şekil 2.3'te de grafiği görülmektedir. Tüm girdi değerleri için türev 0 olmaktadır. Bu da eğitim sırasında kullanılan algoritmalarda zorluk çıkartabilmektedir.

$$f(x) = \begin{cases} 1 & : x \geq 0 \\ 0 & : x < 0 \end{cases} \tag{2.2}$$

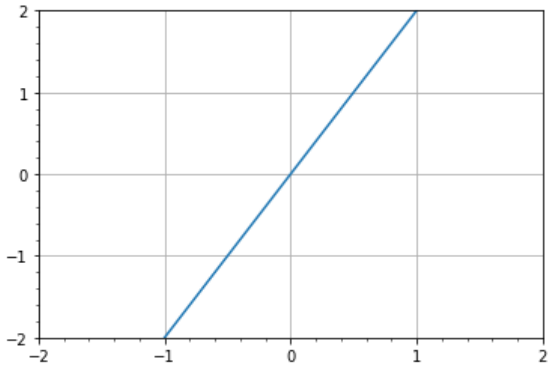


Şekil 2.3. İkili basamak fonksiyonu

2.3.2. Doğrusal fonksiyon (Linear funciton)

Basamak fonksiyonundaki türev problemi doğrusal fonksiyonda bulunmamaktadır. Denklem 2.3'te doğrusal fonksiyonun matematiksel ifadesi, Şekil 2.4'te de grafiği görülmektedir. Doğrusal fonksiyonda türev rahatlıkla hesaplanabilir. Türevin alınabiliyor olması ise geri yayılım algoritması ile katsayıların güncellenmesi için hayati öneme sahiptir (Dubey vd., 2022).

$$f(x) = ax \tag{2.3}$$



Şekil 2.4. Doğrusal fonksiyon

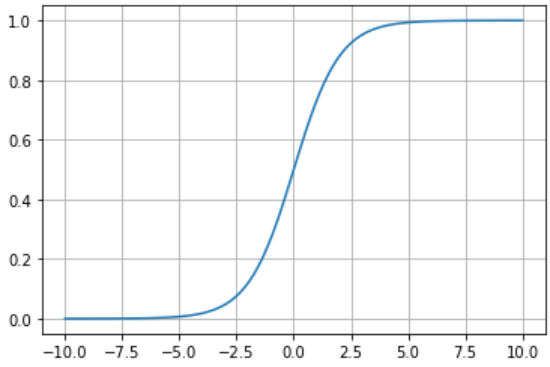
2.3.3. Sigmoid fonksiyonu

En sık kullanılan doğrusal olmayan aktivasyon fonksiyonlarından biri sigmoid fonksiyonudur. Tüm girdi değerleri için çıkış 0 ile 1 arasına sıkıştırılır. Matematiksel ifadesi Denklem 2.4'te grafiği Şekil 2.5'te verilmiştir (Dubey vd., 2022). Doğrusal olmayan bir fonksiyon olduğu için çıktıları da doğrusallıktan kurtarır. Ayrıca tüm girdi değerleri için sürekli türevi vardır (Dubey vd., 2022).

$$f(x) = \frac{1}{(1 + e^{-x})} \tag{2.4}$$

2.3.4. Tanh fonksiyonu

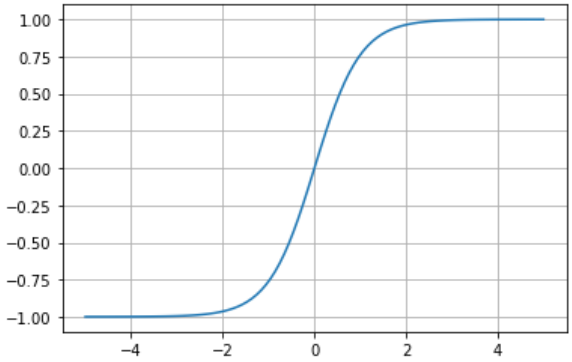
Sigmoid fonksiyonuna çok benzer. Tek farkı sigmoid orijinden geçmez iken tanh fonksiyonu orijinden geçer ve girdileri -1 ile 1 arasına sıkıştırarak çıktı üretir.



Şekil 2.5. Sigmoid fonksiyonu

Matematiksel ifadesi Denklem 2.5'te grafiği Şekil 2.6'da verilmiştir. Tanh fonksiyonunun da tüm girdi değerleri için türevi vardır (Dubey vd., 2022).

$$\tanh(x) = 2 * sigmoid(2x) - 1 \tag{2.5}$$

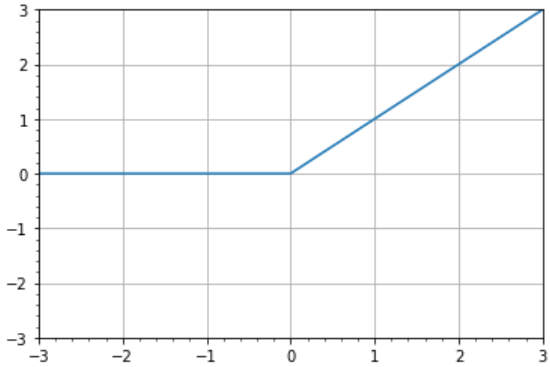


Şekil 2.6. Tanh fonksiyonu

2.3.5. ReLU (Rectified Linear Unit) fonksiyonu

Doğrultulmuş doğrusal birim, doğrusal olmayan bir aktivasyon fonksiyonudur. Sıfırdan küçük girdiler için sıfır çıkışı üretirken sıfırdan büyük değerleri aynen çıkışa aktarır. Yapısı gereği bilgisayar hesaplamalarında oldukça verimli bir fonksiyondur. Bu özelliği sayesinde oldukça sık kullanılan bir aktivasyon fonksiyonudur (Dubey vd., 2022). Doğrultulmuş doğrusal birim matemetiksel ifadesi Denklem 2.6'da grafiği Şekil 2.7'de verilmiştir.

$$f(x) = max(0, x) \tag{2.6}$$

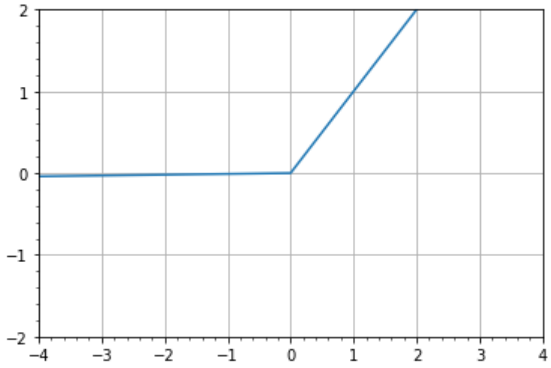


Şekil 2.7. RELU fonksiyonu

2.3.6. Sızıntı ReLU fonksiyonu

ReLU fonksiyonunun sıfırdan küçük değerler için türevinin sıfır olması eğitim sırasında bazı nöronlara ait katsayıların güncellenmesini engellemektedir. Bunun önüne geçmek için ReLU fonksiyonunda sıfırdan küçük değerleri için çok küçük eğime sahip doğrusal bir fonksiyon kullanılır. Bu sayede sıfırdan küçük girdiler için türev sıfırdan farklı olacak ve ReLU fonksiyonundaki problem aşılmış olacaktır (Dubey vd., 2022). Sızıntı ReLU fonksiyonunun matematiksel ifadesi Denklem 2.7'de grafiği Şekil 2.8'de verilmiştir.

$$f(x) = \begin{cases} x & : x \geq 0 \\ 0.01x & : x < 0 \end{cases} \tag{2.7}$$

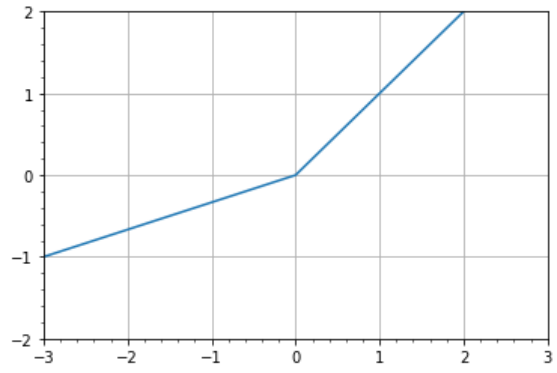


Şekil 2.8. Sızıntı RELU fonksiyonu

2.3.7. Parametrik ReLU fonksiyonu

Parametrik ReLU sızıntı ReLU fonksiyonun biraz daha genellenmiş hali gibidir. Sıfırdan küçük değerler için eğim bir parametre ile kontrol edilir. Bu eğim parametresi 0.01 olduğunda sızıntı ReLU ile aynı forma girer (He vd., 2015). Parametrik ReLU fonksiyonunun matematiksel ifadesi Denklem 2.8'de grafiği Şekil 2.9'da verilmiştir.

$$f(x) = \begin{cases} x & : x \geq 0 \\ ax & : x < 0 \end{cases} \tag{2.8}$$



Şekil 2.9. Parametrik RELU fonksiyonu

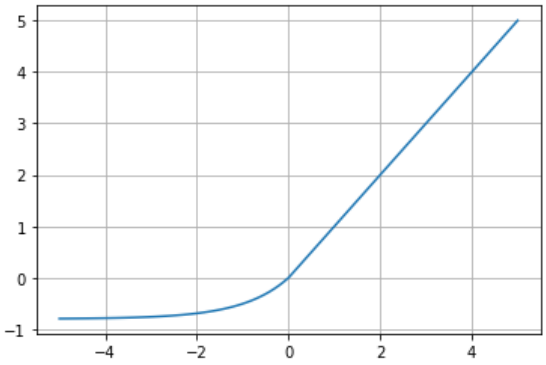
2.3.8. ELU (Exponential Linear Unit) fonksiyonu

Üstel doğrusal birim fonksiyonu da ReLU fonksiyonunun bir varyantıdır. Sıfırdan küçük girdiler için diğer ReLU varyantlarının aksine doğrusal olmayan bir fonksiyon kullanır (Clevert vd., 2016). ELU fonksiyonunun matematiksel ifadesi Denklem 2.9'da grafiği Şekil 2.10'da verilmiştir.

$$f(x) = \begin{cases} x & : x \geq 0 \\ a(e^{x-1}) & : x < 0 \end{cases} \tag{2.9}$$

2.3.9. Softmax fonksiyonu

Birden fazla sınıflandırma probleminde sınıflandırıcı olarak kullanılır. Çıkış her zaman 0 ile 1 arasındadır. Ve tüm çıkışların toplamı 1 etmektedir. Softmax fonksiyonu ile tüm sınıflara ait olma olasılıkları hesaplanır. Bunun sonucunda da en yüksek değere sahip



Şekil 2.10. ELU fonksiyonu

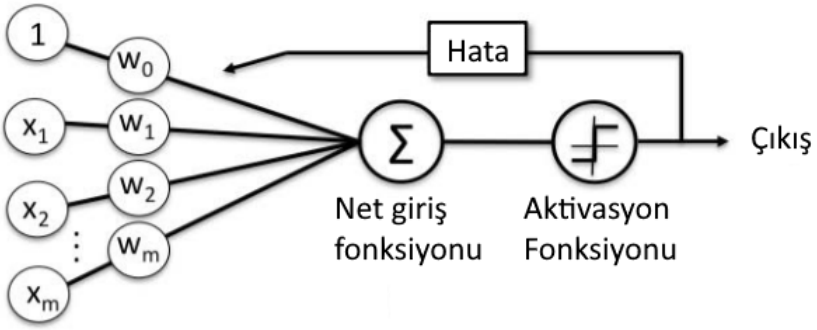
sınıf seçilerek sınıflandırma tamamlanmış olur. Softmax fonksiyonunun matematiksel ifadesi Denklem 2.10'da verilmiştir (Goodfellow vd., 2016).

$$\sigma(x_j) = \frac{e^{x_j}}{\sum_{i=1}^{n} e^{x_i}}, i = 1...n \ . \tag{2.10}$$

2.4. Perceptron

Bir perceptronun belirli girdilere karşılık gelen çıktıları üretebilmesi için giriş ağırlıklarının ayarlanması eğitim olarak tanımlanır. Eğitim süreci boyunca, perceptrona çeşitli giriş vektörleri sunulur ve karşılık gelen çıktılar hesaplanır. Elde edilen çıktı beklenen çıktıdan sapma gösterdiğinde, ağırlıklar önceden tanımlanmış bir kural (genellikle perceptron öğrenme kuralı olarak adlandırılır) çerçevesinde güncellenir ve çıktı hesaplama işlemi tekrarlanır. Bu iteratif süreç, elde edilen çıktılar arzu edilen doğruluk seviyesine ulaşana kadar devam eder. Bu metodolojik işleyiş perceptron eğitim kuralı olarak literatürde yer almaktadır (Haykin, 1999). Perceptron öğrenme kuralının temel prensibi, öğrenme aşamasında başlangıçta tüm ağırlıkların rassal (random) değerlere atanmasıdır. Eşik değeri (bias), girdisi sabit '1' olan ek bir ağırlık olarak modellenebilir. Ardından, perceptronun verilen girdilere karşılık ürettiği çıktı değeri hesaplanır. Hesaplanan çıktı ile beklenen hedef değer arasındaki fark, hata sinyali olarak tanımlanır. Bu hata sinyali, mevcut ağırlıkların güncellenmesi için kullanılır ve bu işlem, perceptronun tüm eğitim örnekleri için kabul edilebilir doğrulukta çıktılar üretene kadar sürdürülür (Şekil 2.11). Hedef çıktı değerlerine yeterli düzeyde yaklaşıldığında, eğitim süreci tamamlanmış kabul edilir. Temelde, bu eğitim süreci çok boyutlu bir optimizasyon problemi olarak formüle edilebilir. Perceptronun giriş sayısı

kadar parametresi (ağırlıklar) bulunan bir fonksiyon söz konusudur ve amaç, bir hata fonksiyonunu (çıktı ile hedef arasındaki farkın bir ölçüsü) minimize edecek parametre değerlerini bulmaktır. Bu temel eğitim algoritması sayesinde perceptron, doğrusal olarak ayrılabilir (linearly separable) bazı basit sınıflandırma ve regresyon problemlerini çözebilmekle birlikte, non-lineer ilişkileri içeren daha karmaşık problemlerin modellenmesinde yetersiz kalmaktadır. Bu sınırlılığın üstesinden gelmek amacıyla çok katmanlı perceptron (Multi-Layer Perceptron - MLP) mimarileri geliştirilmiştir.

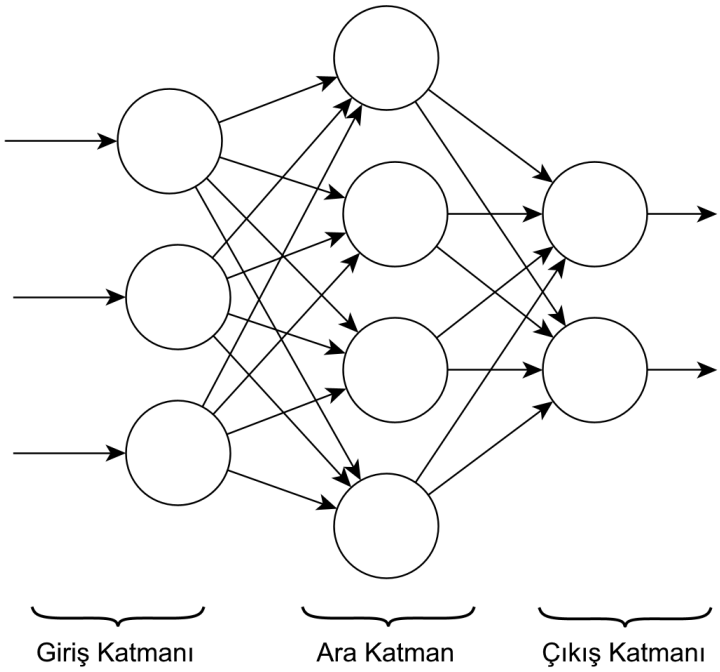


Şekil 2.11. Tek katmanlı perceptron çalışma prensibi

2.5. Çok Katmalı Perceptron (MLP - Multi Level Perceptron)

Bir katmandaki perceptronların çıktılarının başka perceptronların girişlerini oluşturduğu yapılara çok katmanlı perceptron (ÇKP) adı verilmektedir. Çok katmanlı perceptronlarda giriş katmanı, ara katman ve çıkış katmanları bulunur. Ara katmanlar, kendinden önceki katmanlara ve kendinden sonraki katmanlara bağlantılı olan katmanlardır. Giriş ve çıkış perspektifinden bakıldığında görünmez oldukları için gizli katman da denmektedir. Şekil 2.12'de çok katmanlı perceptron yapısı verilmiştir. Birçok katmanlı ağda birden fazla gizli katman bulunabilir. Gizli katmanların görece çok olduğu sinir ağlarına derin sinir ağları denilmektedir. Gizli katmanların çokluğu sayesinde öznitelik çıkarma işlemi yapılmadan ham veriler ile çalışabilirler. Öznitelik çıkarma işi gizli katmanlarda otomatik olarak gerçekleşmektedir (Haykin, 1999).

Çok katmanlı perceptronlarda giriş katmanı, ara katman, çıkış katmanları bulunur. Ara katmanlar kendinden önceki katmanlara ve kendinden sonraki katmanlara bağlantılı olan katmanlardır. Giriş ve çıkış perspektifinden bakıldığında görünmez oldukları için gizli katman da denmektedir. Bir çok katmanlı ağda birden fazla gizli katman bulunabilir.



Şekil 2.12. Çok katmanlı perceptron - MLP

Gizli katmanların görece çok olduğu sinir ağlarına derin sinir ağları denilmektedir. Gizli katmanları çokluğu sayesinde öznitelik çıkartma işlemi yapılmadan ham veriler ile çalışabilirler. Öznitelik çıkartma işi gizli katmanlarda otomatik olarak gerçekleşmektedir.

Her perceptronun girişlerinin birer ağırlığı vardır. Bu ağırlıkların istenen çıkış değerlerini verecek şekilde ayarlanmasına eğitim denir. Sinir ağlarının eğitimi genellikle denetimli öğrenme şeklinde yapılır. Önceden hazırlanmış girdi ve çıktı setleri ağın girişine verilerek uygun çıkışlar alınana kadar katsayılar değiştirilir.

2.5.1. Çok katmanlı perceptronlarda öğrenme

Çok katmanlı perceptronlarda her perceptron için kendinden önceki katmandaki perceptron sayısı kadar ağırlık bulunmaktadır. Bu yüzden derin ağlarda hesaplanması gereken ağırlık sayısı üstel olarak artmakta ve hesap süresi de buna bağlı olarak artmaktadır. Ağırlıkların hesaplanmasında kullanılacak algoritmanın hem yeterli hızda öğrenmeye hem de en az işlem yükü ile çalışmasına ihtiyaç vardır.

Geri yayılım yöntemi bu ihtiyaçları karşılayacak bir algoritmadır. Öncelikle ileri yönde hesaplama yapılır ve tüm ağ yapısının hata değeri bulunur. Daha sonra bu hata

değer geri yönde dağıtılarak sırasıyla katmanlarda ağırlıklar değiştirilir. En baştaki katmana gelindiğinde tekrar yeni değerler ile ileri yönlü hesaplama yapılarak işlemler tekrarlanır (Haykin, 1999).

3. KAYNAK ARAŞTIRMASI

Hız tabelalarının tanınması problemi temel anlamda bir nesne tanıma problemidir. Bir nesne tanıma probleminin çözümü iki aşamadan oluşmaktadır. Bunlar ilgili nesnenin konumunun tespiti ve daha sonra da bu nesnenin sınıflandırılmasıdır. Genel anlamda nesne tanıma problemi incelendiğinde iki temel yaklaşım görülmektedir. Birincisi renk ve şekil analizine dayalı klasik görüntü işleme teknikleridir. Bu yöntemde görüntü üzerinde bir takım görüntü işleme algoritmaları kullanılarak görüntü ile ilgili öznitelikler çıkartılır. Bu öznitelikler ile de nesne tespiti yapılır. Daha sonra bu nesne sınıflandırma algoritmaları ile sınıflandırılır. İkinci yöntem ise yapay zeka temelli yöntemdir. Bu yönteme uygun bir veri seti kullanılarak sistem eğitilir ve hem nesne tespiti hem de sınıflandırması yapılabilir. Ayrıca bu iki yöntem birlikte kullanılarak da bir tespit sistemi oluşturulabilir.

(Fleyeh, 2006) trafik işareti algılamak için renk bölütlemesine dayanan bir sistem önermiştir. Öncelikle görüntü üzerinde renk eşiklemesi kullanmıştır. Trafik işaret tabelası rengi eşikleme yapılarak diğer renklerden ayrılır. Bu işlem resmin yakalandığı RGB renk uzayında yapılamadığı için görüntü öncelikle HSV renk uzayına dönüştürülür. Bunun sebebi RGB renk uzayında çalışırken renkler aşırı ışık ya da gölgelenme gibi etmenlerden çok fazla etkilenir ve eşikleme düzgün yapılamayabilir. Ancak HSV renk uzayında renk özü bileşeni gölgeleme ve aşırı aydınlanmadan çok fazla etkilenmez. Doygunluk ve parlaklık değerleri renk özü bilgisinin anlamlı olduğu yerlerde kullanılır. Tarama yapılacak kare önce 16x16 parçaya bölünür. Her bir parça bölge büyütme (region growing) algoritması için başlangıç pikseli görevi görür. Bu bölgede bulunan beyaz pikseller sayılır. Eğer beyaz piksellerin sayısı bir eşik değeri aşıyor ise bölge büyütme algoritması bu piksellere uygulanır. Bu sayede bu bölgede trafik işaretinin olup olmadığı araştırılır.

(Ghica vd., 1995) eşikleme yapılırken RGB renk uzayında vektör uzunluklarına dayalı bir yöntem önermişlerdir. Eşikleme yapılırken referans bir renk ve eşik değeri belirlenir. Belirli bir pikselin referans renge olan mesafesi hesaplanır. Verilen piksel ve referans renk arasındaki mesafe eşik değerinden küçük ise piksel nesneye ait, büyük ise arka plana aittir. RGB renk uzayında c pikseli kırmızı yeşil ve mavi renk bileşenlerinin doğrusal kombinasyonu şeklinde tanımlanır. Burada:

$$c = c_1 \cdot Red + c_2 \cdot Green + c_3 \cdot Blue \tag{3.1}$$

$$0 < c_1 < c_2 < c_3 < 1 \tag{3.2}$$

dir ve piksel c ve referans renk arasındaki uzaklık

$$d = \sqrt{(r_1 - c_1)^2 + (r_2 - c_2)^2 + (r_3 - c_3)^2} \tag{3.3}$$

denklemleriyle hesaplanır.

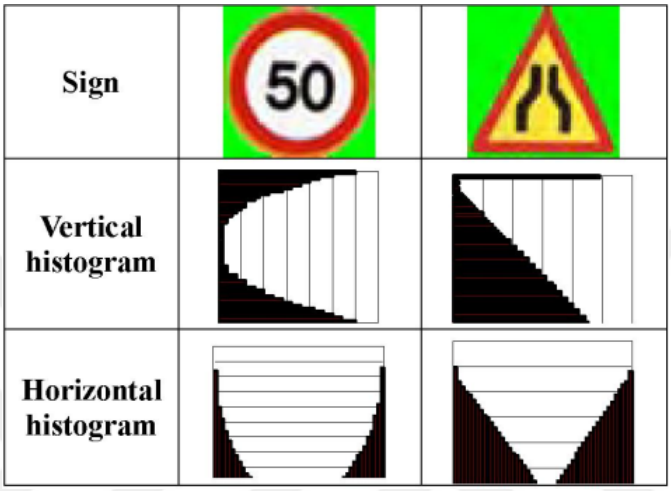
(Garcia-Garrido vd., 2006) dairesel ve üçgen trafik işaretlerinin tespiti için Hough dönüşümü kullanmıştır. Hough dönüşümü için kullanılacak kenar çizgilerini elde etmek için Canny kenar detektörü kullanmışlardır. Canny algoritmasında dinamik eşik kullanılmıştır. Bu eşik histogram tarafından belirlenir. Dinamik eşik aydınlatma koşulları için sağlamlık sağlar. Kenarlar belirlendikten sonra üçgen işaretler için düz çizgiler algılanır ve çizgilerin kesiştiği yer bulunur ve bunlar arasındaki açı kontrol edilir. Çizgilerin başlangıç ve bitiş noktaları Hough dönüşümü kullanılarak tespit edilemez, ancak Hough dönüşümü tüm görüntüye uygulanabilir ve kesişen çizgiler bulunur. Dairesel işaretleri tespit etmek için Dairesel Hough Dönüşümü kullanılır. Örnek işlemler Şekil 3.1'de gösterilmektedir. Bu işlemde ilk olarak orijinal görüntü, daha sonra eşikleme ve son olarak da kenar inceltme işlemi yapılmıştır. Eşikleme işlemi ile görüntü ikili hale getirilmiştir. Kenar inceltme işlemi ile de kenarlar belirginleştirilmiştir. Bu aşamadan sonra Hough dönüşümü uygulanarak üçgen işaret tespit edilmiştir.

(a) Orijinal görüntü (b) Eşiklemeden sonra (c) Kenar inceltmeden sonra

Şekil 3.1. Hough dönüşümü ile üçgen tabelanın tespiti (Garcia-Garrido vd., 2006)

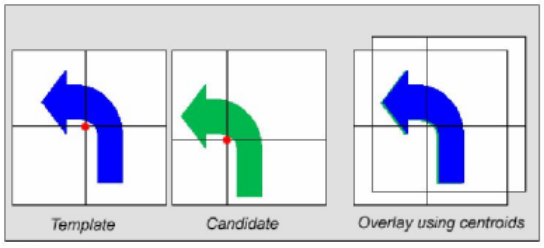
(Andrey ve Jo, 2006), şekil analizine dayalı bir şablon eşleştirme tekniği geliştirmiştir. Bu teknikte trafik işaretleri iki gruba ayrılmıştır. Birinci grup "kırmızı

kenarlı" işaretlerden, ikinci grup ise bilgi işaretlerinden oluşmaktadır. Her grup için farklı tanıma algoritmaları kullanılmıştır. Tespit aşamasında, öncelikle aday trafik işaretleri renk özellikleri kullanılarak kırmızı kenarlı veya bilgi işaretlerinden hangisine ait olduğu belirlenmiştir. Daha sonra ilk grup olan kırmızı kenarlı işaretleri tanımak için trafik işaretinin iç kısmı analiz edilmiştir. Bu gruptaki her aday işaret, daire veya üçgen olabilir. Aday şekillerin daire veya üçgen gruplarından hangisine dahil olduğunun belirlenmesinde Şekil 3.2'deki gibi arka plan histogramları kullanılır.



Şekil 3.2. Arka plan histogramı (Andrey ve Jo, 2006)

Tanıma adımı için, işaretin siyah bölgesi analiz edilmelidir. İşaret adayında, işaretin siyah kısmı çıkarılır ve her aday bir ikili maske olarak sunulabilir. Şablon eşleştirmede, Şekil 3.3'teki gibi şablon maskeleri kullanılır. Bu teknik 172 trafik işaret için test edilmiştir. Yöntemin başarı oranı yaklaşık %90'dır.



Şekil 3.3. Şablon eşleştirme (Andrey ve Jo, 2006)

(Malik vd., 2007) piktogram sınıflandırmasına dayalı bir şablon eşleştirme tekniği sunar. Bu teknik sadece siyah iç kısmı olan işaretleri ve kırmızı kenarlı daire işaretlerini dikkate almaktadır. İlk olarak, RGB renk uzayındaki giriş görüntüsü HSV renk uzayına dönüştürülür. Daha sonra bu görüntüye kırmızı renk segmentasyonu uygulanır. Bölümlenmiş kırmızı bölge, kırmızı renkli reklamlar gibi istenmeyen kırmızı

bölgeleri de içermektedir. İstenmeyen bölgeleri uzaklaştırmak için filtreleme uygulanır. Filtreleme adımında, her bölgenin alanı bir eşik değer ile karşılaştırılır. Filtreleme aşamasından sonra, istenmeyen bölgeleri uzaklaştırmak için şekil tespiti uygulanır. Bölgenin şekli üçgen veya daire ise, bölge bir işaret olarak kabul edilir. İşaret algılandıktan sonra, trafik işaretinin beyaz bölgesi çıkarılır. Siyah bölge görüntüsü ile çıkarılan beyaz bölge görüntüsü arasında bir mantıksal VE işlemi uygulanır. Bu işlemin sonucu, çıkartılan piktogram görüntüsüdür. Çıkarılan iç kısım normalleştirilir ve şablon eşleştirmeye uygun hale getirilir. Eşleştirme tüm veritabanında gerçekleştirilmez. Veritabanındaki işaretler iki gruba ayrılır. Eşleşen grup, giriş trafik işaretinin şekline göre belirlenir. Yöntemin tanınma oranı yaklaşık %85,11'dir.

(Shi vd., 2008) şekil tabanlı tanıma modeline dayanan destek vektör makinesi (Support Vector Machine - SVM) önermiştir. Bu çalışmada dört SVM çekirdeği karşılaştırılmış ve iki farklı SVM türü kullanılmıştır. Doğrudan ikili gösterim özellik seçme algoritmasıyla her bir görüntü 36x36 piksel boyutuna yeniden boyutlandırılır ve bu görüntü 1296 giriş vektörü olarak alınır. Eğitim ve test için 600 adet trafik işareti görüntüsünden oluşan bir veri tabanı kullanılmıştır. Her veri seti için 50 görüntü bulunmaktadır. Her işaretin örnekleri eğitim için 30 ve test için 20 olmak üzere 2 kategoriye ayrılmıştır. İkili sınıflandırma ile SVM performansı neredeyse %100 ve Zernike Momentleri ile SVM performansı %95'in üzerindedir.

(Fištrek ve Lonari, 2011) yapay sinir ağlarını kullanan bir sistem önermişlerdir. Çalışmalarında yol hız tabelalarına odaklanmışlardır. Yol hız limitlerini belirten tabelalar yuvarlak ve kırmızı çerçevelidir. Bu tabelaların tespiti yapıldıktan sonra sisteme verilmesi için 16x16 piksele dönüştürülür. Bu 256 piksel daha sonra yapay sinir ağına giriş olarak verilir. Sistem eğitilirken her bir hız limiti tabelasından 10'ar adet örnek kullanılmıştır. Bu örnekler farklı ışıklandırma ve bozulmalar içermekte ve sistemin başarısını arttıracak şekilde seçilmiştir. YSA'nın çıkışı olarak da toplam hız limiti tabelası sayısının iki fazlası çıkış vermektedir. Her bir hız limiti tabelası için tanınması durumunda bir çıkış ve başarısız ve bilinmeyen tabela için de birer çıkış şeklindedir. Sistem 50 görüntü ile test edilmiştir. Tabelaların tespit edilmesindeki başarı oranı %58, sınıflandırmadaki başarı oranı ise %42'dir.

(Shustanova ve Yakimov, 2017) trafik işaretleri tanıma ile ilgili yaptıkları çalışmalarda CCN (convolutional neural network) kullanmışlardır. En çok kullanılan 16 trafik işaretini gerçek zamanlı olarak tanıyabilmek için araç ön camına yerleştirilmiş bir

kameradan toplanan görüntüler bir CNN ağından geçirilerek mevcut tabelanın tespiti ve sınıflandırılması yapılmıştır. Bu işlem üç ana adımda yapılmıştır. Resim önişleme, trafik tabelasının tespiti ve sınıflandırması. Resim önişleme sırasında kameradan alınan görüntü RGB renk uzayından HSV renk uzayına dönüştürülmüştür. Daha sonra tabelaların tespiti için Huogh dönüşümünün modifiye edilmiş hali kullanılmıştır. Daha sonra tespit edilen trafik tabelaları CNN ile sınıflandırmaya tabi tutulmuştur. Bu çalışmada başarı oranı %99.94 olmuştur.

(Alghmghama vd., 2019) arap trafik işaretlerini tanımlama üzerine çalışmışlardır. En yaygın 24 trafik tabelasının sınıflandırmasını yapmışlardır. Sınıflandırılacak tabela görüntüleri önce 30x30 piksellik gri görüntülere dönüştürülmüştür. Daha sonra aşağıdaki CNN mimarisi kullanılarak sınıflandırma yapılmıştır. Kullanılan veri seti 2728 görüntüden oluşmaktadır. Bunun 2183 (%80) adedi eğitim, 545 (%20) adedi test için kullanılmıştır. Test verilerinin de bir kısmı (436 adet) doğrulama için kullanılmıştır. Daha sonra mimari sadeleştirilerek katman sayısı azaltılmış ancak sistem performansı korunmuştur. Son mimari ile eğitimin 150. epoch turunda %100 test başarısı alınmıştır.

(Haque vd., 2021) trafik işareti tanıma için DeepThin adını verdikleri yeni bir CNN mimarisi önermiştir. Bu mimari, katman başına az sayıda özellik haritası kullanarak önemli öznitelikleri öğrenmeyi hedefleyen hafif ve derin bir ağ yapısına sahiptir. Özellikle düşük çözünürlüklü görüntüler için tasarlanan bu mimari, GPU olmadan eğitim yapmayı mümkün kılacak şekilde düşük işlem gücü gerektirir. DeepThin, daha hızlı eğitim ve düşük aşırı öğrenme (overfitting) sağlamak için örtüşen maksimum havuzlama ve adımlı evrişim tekniklerini kullanır. Ayrıca, topluluk öğrenimi için çok sayıda model yerine, yalnızca renkli ve gri tonlamalı örnekler üzerinde eğitilmiş az sayıda modelin tahminlerinin ortalamasını almanın yeterli olduğu gösterilmiştir. Bu yaklaşım, uygulama maliyetlerini düşürmekle birlikte donanım entegrasyonunu da kolaylaştırarak hafif donanımlarda verimli bir şekilde kullanılabilir hale gelmiştir. Yöntemin doğruluk oranı %99.29 olmuştur.

4. MATERYAL VE YÖNTEM

Bu tezde aracın ön camına yerleştirilen bir donanım ile gerçek zamanlı olarak yoldaki hız limiti tabelalarının tanımlanması yapılmıştır. Bunun için bir kamera ve kameradan alınan görüntülerin işlenmesi için bir görüntü işleme donanımı kullanılmıştır. Görüntü işleme donanımı için Jetson Nano platformunda bir yapay zeka modeli geliştirilmiştir. Bu yapay zeka modelini geliştirmek için GTSDB (German Traffic Sign Detection Benchmark) veri seti (Houben vd., 2013), Çin Trafik İşareti Veri Tabanı (Chinese Traffic Sign Database) (Huang, 2012) ve kendi topladığımız veriler kullanılarak oluşturduğumuz veri setinden yararlanılmıştır. Kullanılan materyallerin detaylı açıklaması aşağıdadır.

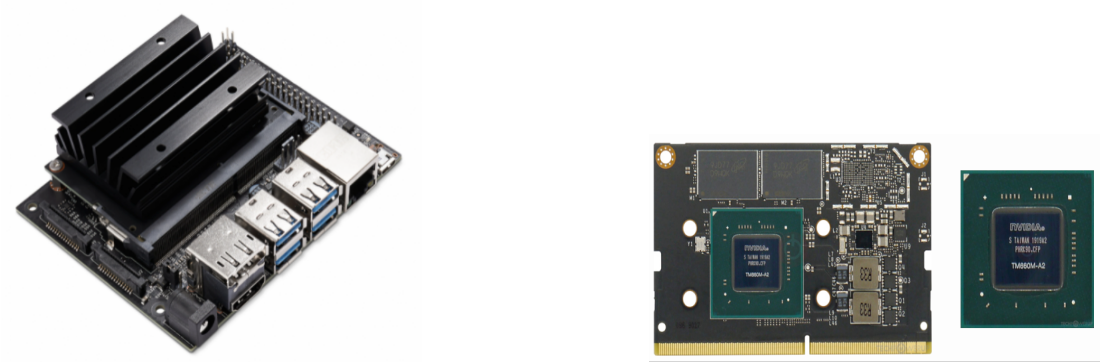
4.1. Kullanılan Donanımlar

Çalışmamız gerçek zamanlı olacağı için toplanan görüntülerin toplandığı noktada işlenmesi gerekliliğini doğurmaktadır. Ayrıca araç içerisinde kullanılacak olan görüntü işleme donanımının hem yeterince küçük hem de yeterince güçlü olması gerekmektedir. Yalnızca CPU donanımına sahip bir uygulama gerçek zamanda yeterli performansı vermeyeceğinden, yapay sinir ağlarının etkin bir şekilde işletilebileceği yüksek paralel hesaplama gücüne sahip bir GPU donanımının kullanılmasına karar verilmiştir. Bu yüzden boyut olarak küçük, güç tüketimi az ve yeterince güçlü işlem gücüne sahip bir GPU içeren donanım araştırılmıştır. NVIDIA firmasının yapay zeka uygulamaları için geliştirdiği ve Maxwell mimarisi ile hazırlamış olduğu içerisinde bir GPU barındıran Jetson Nano ürününün istenen gereksinimleri karşıladığı için bu çalışmada Jetson Nano 4GB geliştirici kiti, görüntü toplama ve işleme birimi olarak kullanılmıştır. Temelde bir SBC (Single Board Computer – Tek kart bilgisayar) olan bu donanım üzerinde Ubuntu işletim sistemi çalışmaktadır. Kit üzerinde 128 çekirdekli görüntü işleme birimi (CUDA core), 4 çekirdekli 64-bit ARM işlemci, 4 GB LPDDR4 RAM, depolama için SD kart yuvası, 2 adet MIPI kamera bağlantısı, Gigabit Ethernet modülü, 4 adet USB 3.0, görüntü çıkışı için HDMI ve Display Portu bulunmaktadır. Jetson nano geliştirici kitinin donanımına ait bilgiler 4.1'de, çalışmada kullanılan Jetson Nano geliştirici kiti Şekil 4.1a'da ve bu modülde kullanılan GPU'nun fotoğrafları Şekil 4.1b'de görülmektedir. Çalışmada görüntülerin toplanması için Raspberry Pi Kamera

Modülü V2 kullanılmıştır. Bu kamera 8 megapiksel (3280 x 2464) çözünürlüğe ve 30 fps görüntü alma hızına sahiptir. Jetson Nano, bu kamera modülünü doğrudan destekleyen sürücülere sahiptir. Sistemin gerçek zamanlı çalıştırılması Jetson Nano üzerinden gerçekleştirilmesine rağmen, sistem eğitiminin daha hızlı olabilmesi için Çizelge 4.2'de özellikleri verilen bilgisayar kullanılmıştır. Bilgisayar üzerinde eğitim tamamlandıktan sonra kullanılmak istenen ağırlıkların bulunduğu dosya Jetson nano kitine aktarılarak bilgisayarda eğitilmiş yapay zeka modeli bu cihaz üzerinde kullanılabilmektedir.

Çizelge 4.1. Jetson Nano özellikleri

GPU	128-core Maxwell
CPU	Quad-core ARM A57 @ 1.43GHz
RAM	4 GB 64-bit LPDDR4 25.6 GB/s
Bellek	microSD
Kamera	2x MIPI CSI-2 DPHY lanes
Bağlanabilirlik	Gigabit Ethernet, M.2 Key E
Ekran	HDMI and Display Port
USB	4x USB 3.0, USM 2.0 Micro-B
Diğer	GPIO, I2C, I2s, SPI, UART



(a) Jetson Nano geliştirici kiti (b) Jetson Nano'da kullanılan GPU

Şekil 4.1. Jetson Nano platformu

Çizelge 4.2. Eğitim bilgisayarının özellikleri

GPU	NVIDIA RTX 4060 Ti 16GB
CPU	AMD Ryzen 5 5500 3,6 GHz
RAM	32 GB DDR4 3200 MHz
Disk	1 TB M2 SSD

4.2. Kullanılan Yazılımlar

Bu çalışmanın yazılımında Jetpack SDK kullanılmıştır. Jetpack SDK, Jetson Nano donanımında çalışacak işletim sistemi ve gerekli sürücülerin yanı sıra, yapay zeka çalışmalarında kullanılacak başlıca yazılım ve kütüphaneleri içeren bir pakettir. Çalışma boyunca Jetpack SDK ile sağlanan sürücü ve kütüphaneler kullanılmıştır. Bunun temel sebebi, bu sürücülerin kullanılan donanım için özel olarak optimize edilmiş olmasıdır. Bu optimizasyon sayesinde donanımdan en yüksek verim alınarak işlemlerin gerçek zamanlı senaryolarda olabildiğince hızlı bir şekilde çalıştırılması amaçlanmaktadır. Bu SDK ile birlikte gelen kütüphaneler C++ ve Python dilleriyle hazırlanmış sürücü ve kütüphanelerden oluşmaktadır. Bu çalışmada Jetpack SDK hem C++ hem de Python ile geliştirilmiş kütüphane ve sürücüler sunmasına rağmen, Python dili tercih edilmiştir. Python'ın kolay kullanımı ve geniş kaynak erişimi bu tercihte önemli rol oynamıştır. Bu çalışmada kullanılan Python sürümü 3.6'dır. Bir SD kart imajı şeklinde dağıtılan Jetpack SDK paketinin içeriği aşağıda verilmiştir.

İşletim sistemi: Ubuntu 18.04 LTS

TensorRT: Görüntü sınıflandırma, segmentasyon ve nesne algılama sinir ağları için yüksek performanslı bir derin öğrenme çıkarım çalışma zamanıdır. TensorRT, NVIDIA'nın paralel programlama modeli olan CUDA üzerine kurulmuştur ve tüm derin öğrenme çerçeveleri için çıkarımı optimize etmenize olanak tanır. Derin öğrenme çıkarım uygulamaları için düşük gecikme süresi ve yüksek verim sağlayan bir derin öğrenme çıkarım iyileştiricisi ve çalışma zamanı içerir.

cuDNN: CUDA derin sinir ağı kitaplığı, derin öğrenme çerçeveleri için yüksek performanslı temel öğeler sağlar. İleri ve geri evrişim, havuzlama, normalleştirme ve etkinleştirme katmanları gibi standart rutinler için yüksek düzeyde ayarlanmış uygulamalar sağlar.

CUDA: CUDA Toolkit, GPU ile hızlandırılmış uygulamalar oluşturan C ve C++ geliştiricileri için kapsamlı bir geliştirme ortamı sağlar. Araç seti, NVIDIA GPU'lar için bir derleyici, matematik kitaplıkları ve hata ayıklama ve uygulamalarınızın performansını optimize etmeye yönelik araçlar içerir.

OpenCV: Bilgisayarla görme, görüntü işleme ve makine öğrenimi için önde gelen bir açık kaynak kitaplığıdır.

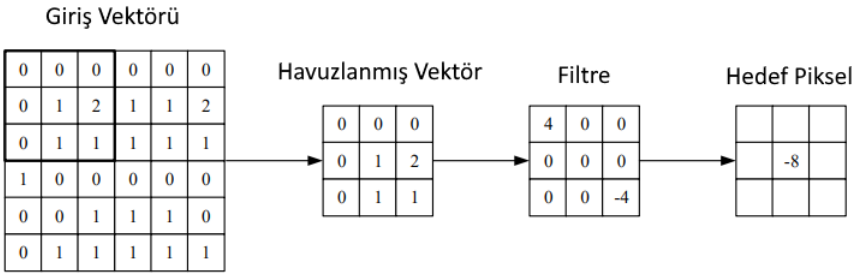
4.3. CNN Mimarisi

Bu çalışmada özellikle düşük güçlü sistemlerde verimli çalışması için tasarlanmış olan SSD mimarisi kullanılmıştır. Google'ın da desteğiyle Liu ve ark. tarafından geliştirilmiş olan bu mimari bir CNN (Convolutiona Neural Netwok – Evrişimsel Sinir ağı) mimarisidir (Liu vd., 2016b). CNN özellikle görüntü işleme uygulamalarında kullanılan bir mimari çeşididir. CNN mimarisinde görüntü bir dizi katmandan geçirilerek en son katmanda sınıflandırma sonucu verecek şekilde tasarlanmıştır. Kullanılan katmanlar çok çeşitli olmakla birlikte en sık kullanılanlar giriş katmanı, evrişim katmanı (Convolution Layer), RELU katmanı, havuzlama (pooling) katmanı, tam bağlantılı (fully connceted – FC) katmanıdır.

Giriş katmanından gelen girdi verileri, daha sonra evrişim katmanlarından (convolution layer) geçirilir. Bu katmanlar genellikle birden fazladır ve ardışık olarak uygulanır. Her katmanda, girdiye ilişkin farklı özellikler çıkarılarak ağ boyunca bu özelliklerin öğrenilmesi sağlanır.

4.3.1. Evrişim (Convolution) katmanı

Evrişim katmanı, bir girdi görüntüsünden özellikleri çıkaran ilk katmandır. Bu özellik çıkarma işlemini girdi görüntüsü üzerinde küçük filtreler (veya çekirdekler) kaydırarak ve her pozisyonda bir evrişim işlemi uygulayarak gerçekleştirir. Evrişim işlemi iki matrisin eleman bazında çarpımlarının toplamı şeklindedir. Şekil 4.2'de evrişim işlemi için bir örnek gösterilmiştir.



Şekil 4.2. Evrişim işlemi

Filtre tüm girdi üzerinde kaydırılarak tüm girdi matrisi taranan kadar işleme devam eder. Oluşan çıkış matrisinin boyutu:

$dim(girdi) = (n_H, n_W, n_C)$ ve,

$dim(filtre) = (f, f, n_C)$, ise

$$dim(cikti) = (n_H - f + 1, n_W - f + 1, 1) \tag{4.1}$$

şeklinde olacaktır. Burada n_H girdi görüntüsünün yüksekliğinin piksel sayısı, n_W girdi görüntüsünün genişliğinin piksel sayısı, n_C girdi görüntüsünün kanal sayısı, f ise filtre boyutudur. Görüntü üzerinde kaydırma işlemi sırasında filtre girdi görüntüsünün kenarlarına geldiğinde bazı elemanlar işleme dahil edilmemektedir. Bu durum, kenar kısımlarındaki elemanların daha az işlemden geçmesine ve dolayısıyla daha az etkili olmasına neden olmaktadır. Bu durumu gidermek için sıfır ekleme işlemi yapılmaktadır. Sıfır ekleme işlemi, girdi görüntüsünün etrafına sıfır eklenerek girdi görüntüsünün boyutunu arttırma işlemidir. Bu sayede evrişim işlemi sırasında kenar kısımlarındaki elemanlar da işleme dahil edilerek daha etkili bir sonuç elde edilebilir.

Sıfır ekleme işlemi ayrıca çıktı boyutlarını da etkilemektedir. Girdi boyutu arttığı için buna paralel olarak çıktı boyutu da artmaktadır. Sıfır eklendikten sonra oluşan çıkış matrisinin boyutu:

$dim(girdi) = (n_H, n_W, n_C)$,
$dim(filtre) = (f, f, n_C)$, ise

$$dim(cikti) = (n_H + 2p - f + 1, n_W + 2p - f + 1, 1) \tag{4.2}$$

şeklinde olacaktır. Burada n_H girdi görüntüsünün yüksekliğinin piksel sayısı, n_W girdi görüntüsünün genişliğinin piksel sayısı, n_C girdi görüntüsünün kanal sayısı, f ise filtre boyutu p eklenen sıfır sayısıdır.

Çıkışın boyutunu değiştiren bir diğer parametre ise kaydırma parametredir. Evrişim işlemi sırasında filtrenin kaydırılması birer birer yapılmaktadır. Ancak bazı durumlarda bu kaydırma parametresi birden farklı seçilebilir. Bu parametre arttıkça oluşan çıktının boyutu azalmaktadır. Hem sıfır ekleme hem de kaydırma parametresinin varlığı durumunda oluşacak çıkışın boyutları şu şekildedir:

$dim(girdi) = (n_H, n_W, n_C)$,
$dim(filtre) = (f, f, n_C)$, ise

$$dim(cikti) = (\frac{n_H + 2p - f}{s} + 1, \frac{n_H + 2p - f}{s} + 1, 1) \tag{4.3}$$

şeklinde olacaktır. Burada s kaydırma miktarı, p ise eklenen sıfır sayısıdır. n_H girdi

görüntüsünün yüksekliğinin piksel sayısı, n_W girdi görüntüsünün genişliğinin piksel sayısı, n_C girdi görüntüsünün kanal sayısı, f ise filtre boyutudur.

Görüldüğü üzere evrişim işlemi sonucunda girdi kanal sayısı birden fazla olsa da çıktı her zaman tek kanaldan oluşmaktadır. Ancak girdiye uygulanacak filtreler birden fazla olabilir. Bu durumda çıkışta kaç kanal isteniyor ise o kadar sayıda filtre uygulanarak çıkış katman sayısı arttırılabilir. Bu sayede aynı girdi üzerinde birden fazla filtre uygulanarak farklı özellikler tespit edilebilir.

4.3.2. Havuzlama (Pooling) katmanı

Evrişim katmanındaki sıfır ekleme ve kaydırma işlemi gibi havuzlama katmanı da girdinin boyutlarını değiştiren bir işlemdir. Havuzlama sayesinde girdi boyutları azaltılarak hem gerekli işlem gücü azaltılmış olur hem de önceki katmanda yakalanan gereksiz özellikler yok sayılarak önemli özelliklere odaklanılması sağlanabilir. Havuzlama işlemi de evrişim işlemi gibi girdi matrisi üzerinde gezen bir filtreden oluşmaktadır. Ancak burada yapılan işlem elemanların çarpımlarının toplamı şeklinde değil havuzlama tekniğine göre değişen bir işlemdir. Sıfır ekleme ve kaydırma parametreleri burada da kullanılabilir.

Havuzlama yapılırken kullanılabilecek birden fazla teknik mevcuttur. Bunlardan en çok kullanılanları, maksimum havuzlama (max pooling) ve ortalama havuzlama (average pooling)'dır. Maximum havuzlama tekniğinde filtre görüntü üzerinde kaydırılırken filtrenin kapsadığı alandaki en büyük değeri çıkış olarak verir. Ortalama havuzlama uygulanırken ise filtredeki değerlerin ortalamasını çıkış olarak verir. Şekil 4.3'te 4x4'lük bir matris üzerinde 2x2 boyutunda kayan bir filtre uygulandığında maksimum ve ortalama havuzlama işlemi ile elde edilen çıkışlar görülmektedir.

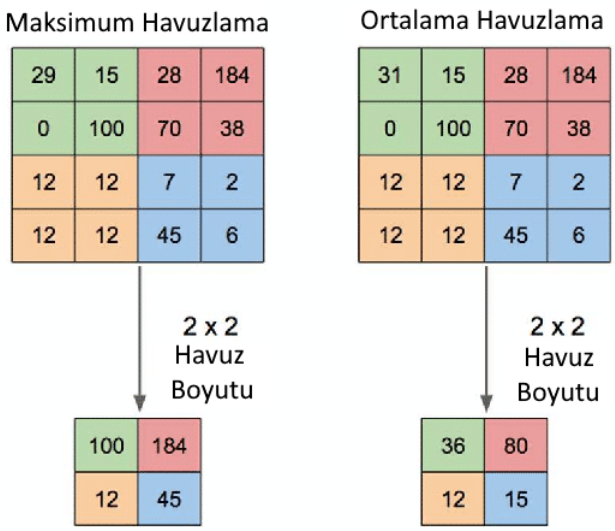
Hem sıfır ekleme hem de kaydırma parametresinin varlığı durumunda oluşacak çıkışın boyutları şu şekildedir:

$dim(girdi) = (n_H, n_W, n_C)$,

$dim(filtre) = (f, f, n_C)$, ise

$$dim(cikti) = (\frac{n_H + 2p - f}{s} + 1, \frac{n_H + 2p - f}{s} + 1, 1) \tag{4.4}$$

şeklinde olacaktır. Burada s kaydırma miktarı, p ise eklenen sıfır sayısıdır. n_H girdi görüntüsünün yüksekliğinin piksel sayısı, n_W girdi görüntüsünün genişliğinin piksel



Şekil 4.3. Havuzlama işlemi

sayısı, n_C girdi görüntüsünün kanal sayısı, f ise filtre boyutudur.

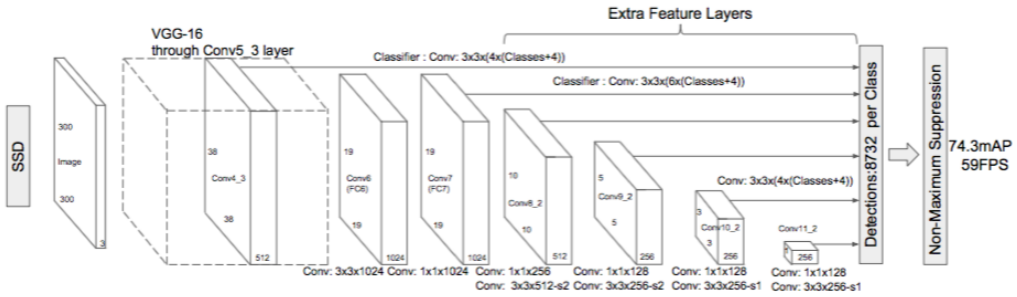
4.3.3. Tam bağlantılı (Fully connected) katman

Tam bağlantılı bir katmandaki nöronlar, normal sinir ağlarında görüldüğü gibi, önceki katmandaki tüm aktivasyonlarla tam bağlantıya sahiptir. Sınıflandırma yapılmadan önce tüm girdilerin tek düz bir vektör haline dönüştürüldüğü katmandır. Tam bağlantılı katmana yoğun katman (dense layer) da denmektedir. Bunun sebebi evrişim ve havuzlama katmanlarındaki gibi filtreleri girdinin yalnızca bir kısmı ile bağlantılı olmayıp kendinden önceki katmanın tüm çıktılarına bağlanması ve diğer katmanlardan çok daha fazla bağlantıya sahip olmasıdır.

4.4. SSD Multibox Mimarisi

SSD mimarisi, özellikle kısıtlı işlem gücüne sahip mobil ve gömülü sistemlerde gerçek zamanlı olarak kullanılabilecek bir mimari olarak tasarlanmıştır. Temelde iki temel özellik taşımaktadır. Bunlar bir görüntüdeki nesneleri tek bir ileri geçişte tespit edebilmesi yani tek atış (single-shot) nesne algılaması ve her seferinde birden fazla nesne için sınır kutusu tahmin edebilmesi yani çoklu kutu (multibox) tahminidir. Tek atış nesne algılama sayesinde daha hızlı bir işlem süresi sağlanmaktadır. Çoklu kutu tahmin sayesinde de tek bir görüntüde birden fazla nesne tespit edilebilir. SSD mimarisi hem tek atış hem de çoklu kutu tahminini bir araya getirerek, nesne algılama görevlerini

daha hızlı ve verimli bir şekilde gerçekleştirmeyi amaçlamaktadır. Bu mimari, derin öğrenme tabanlı nesne algılama sistemlerinde yaygın olarak kullanılmaktadır. SSD mimarisinin yapısı Şekil 4.4'te verilmiştir.



Şekil 4.4. SSD mimarisi (Liu vd., 2016a)

Orijinal SSD mimarisi VGG-16 mimarisinin temel yapısı üzerine inşa edilmiştir. Ancak, VGG mimarisinin tam bağlantılı katmanı yerine, ekstra özellik katmanları eklenmiştir. Bu katmanlar, ardışık olarak azalan boyutlarla çoklu boyutlu tespit yapılmasına olanak tanır. Ayrıca, bu katmanlar sayesinde algılama için evrişimsel tahminciler üretmek mümkündür.

Eklenen her özellik katmanı (veya isteğe bağlı olarak temel ağdan mevcut bir özellik katmanı), bir dizi evrişimli filtre kullanarak sabit bir algılama tahminleri seti oluşturabilir. c kanallı m × n boyutundaki bir özellik katmanı için, potansiyel algılamanın parametrelerini tahmin etmek amacıyla, her bir kategori için bir puan veya varsayılan kutu koordinatlarına göre bir fark değeri üreten 3 × 3 × c boyutunda küçük bir çekirdek kullanılır. Bu çekirdek, m × n konumlarının her birinde bir çıktı değeri üretir. Sınırlayıcı kutu ofset çıkış değerleri, her bir özellik haritası konumuna göre varsayılan bir kutu konumuna göre ölçülmektedir.

Ayrıca, çoklu özellik haritaları için her bir özellik haritası hücresiyle bir dizi varsayılan sınırlayıcı kutu ilişkilendirilir. Varsayılan kutular, özellik haritasını evrişimsel olarak kaplayarak, her kutunun ilgili hücreye göre konumunu sabitler. Her özellik haritası hücresinde, hücredeki varsayılan kutu şekillerine göre uzaklıklar ve bu kutuların her birinde bir sınıf örneğinin varlığını gösteren sınıf başına puanlar tahmin edilir. Farklı özellik haritalarında çeşitli varsayılan kutu şekillerine izin verilmesi, olası çıkış kutusu şekillerinin alanının verimli bir şekilde ayrılmasını sağlar.

SSD mimarisinde kullanılan temel katman VGG-16 mimarisinde olmasına rağmen, bu mimari ile birlikte kullanılabilecek başka mimariler de bulunmaktadır.

Bunlardan bir tanesi MobilNet mimarisi, bir diğeri de SqueezeNet mimarisidir. Bu mimariler, SSD mimarisinin temel yapısında kullanılan VGG-16 yerine kullanılabilir. MobilNet mimarisi, mobil cihazlarda ve gömülü sistemlerde kullanılmak üzere tasarlanmış bir derin öğrenme mimarisidir. SqueezeNet ise, daha az parametre ile yüksek doğruluk sağlamak amacıyla geliştirilmiş bir derin öğrenme mimarisidir. Bu iki mimari, SSD mimarisi ile birlikte kullanılarak daha verimli ve hızlı nesne algılama sistemleri oluşturulabilir.

4.4.1. MobileNet Mimarisi

MobileNet mimarisinde standart evrişim işlemi yerine derinlemesine ayrışabilir evrişim (depthwise seperable convolution) adlı bir metot kullanılmıştır. Bu işlem standart evrişim işlemini derinlemesine evrişim (depthwise convolution) ve noktasal evrişim (pointwise convolution) olmak üzere iki parçaya ayırarak hesaplamayı oldukça hızlandıran bir yöntemdir (Howard vd., 2017). Standart bir evrişim katmanında işlem yükü

$$n_H \times n_W \times n_C \times f \times f \times N \tag{4.5}$$

formülüyle belirlenir. Burada n_H girdi görüntüsünün yüksekliğinin piksel sayısı, n_W girdi görüntüsünün genişliğinin piksel sayısı, n_C girdi görüntüsünün kanal sayısı, f ise filtre boyutu, N filtre sayısıdır.

4.4.1.1. Derinlemesine evrişim (Depthwise convolution)

Standart evrişim katmanında her bir filtre tüm kanallara uygulanmaktadır. Bu durumda işlem yükü oldukça fazladır. Derinlemesine evrişimde ise kullanılan filtreler yalnızca bir kanala uygulanır. Eğer girdi 3 kanallı ise 5 adet filtre kullanılarak yine 5 kanallı bir çıktı elde edilir. Derinlemesine evrişim katmanında işlem yükü

$$n_H \times n_W \times n_C \times f \times f \tag{4.6}$$

formülüyle bulunabilir. Burada n_H girdi görüntüsünün yüksekliğinin piksel sayısı, n_W girdi görüntüsünün genişliğinin piksel sayısı, n_C girdi görüntüsünün kanal sayısı, f ise filtre boyutudur.

4.4.1.2. Noktasal evrişim (Pointwise convolution)

Noktasal evrişim 1x1 filtreye sahip standart bir evrişim katmanına eşittir. Derinlemesine evrişim katmanındaki kanal sayısını değiştirmek için kullanılır. Buradaki işlem yükü

$$n_H \times n_W \times n_C \times M \tag{4.7}$$

formülüyle bulunabilir. Burada M filtre sayısını gösterir. İki işlem art arda yapıldığında toplam işlem yükü

$$n_H \times n_W \times n_C \times f \times f + n_H \times n_W \times n_C \times M \tag{4.8}$$

formülüyle bulunur. Standart evrişim katmanının işlem yüküne oranı ise

$$\frac{n_H \times n_W \times n_C \times f \times f + n_H \times n_W \times n_C \times M}{n_H \times n_W \times n_C \times f \times f \times N} = \frac{1}{N} + \frac{1}{f^2} \tag{4.9}$$

olacaktır. Filtre boyutu $f = 3$ için gereken işlem gücü yaklaşık 8 ile 9 kat arası azalmaktadır.

MobileNet mimarisinde kullanılan katmanlar, filtre boyutları ve ilgili katmanın giriş boyutu Çizelge 4.3'te gösterilmiştir. MobileNet mimarisinde her evrişim işleminden sonra toplu normalleştirme (batch normalization) yapılmaktadır. Bu sayede katmanların girdilerini yeniden merkezleme ve yeniden ölçeklendirme yoluyla normalleştirme yapılarak yapay sinir ağlarını daha hızlı ve daha kararlı hale getirilir.

Orijinal SSD mimarisindeki VGG-16 mimarisinin yerine MobileNet mimarisi kullanıldığında, SSD mimarisinin temel yapısı değişmemektedir. Ancak, MobileNet mimarisi daha az parametre ile daha hızlı bir işlem süresi sağlamaktadır. Bu sayede mobil cihazlarda ve gömülü sistemlerde kullanılabilir hale gelmektedir. MobileNet mimarisinin temel yapısı Şekil 4.5'te verilmiştir.

4.4.2. SqueezeNet Mimarisi

SqueezeNet, Iandola ve diğerleri tarafından geliştirilen bir CNN mimarisidir. Bu mimaride modelin sıkıştırılması ve parametrelerin azaltılması için sonradan bir budama işlemi yerine, model oluşturulurken en baştan akıllıca bir sıkıştırma yöntemi geliştirilmiştir. Üç adımda gerçekleştirilen bu yeni yapı için "ateşleme modülü" (fire

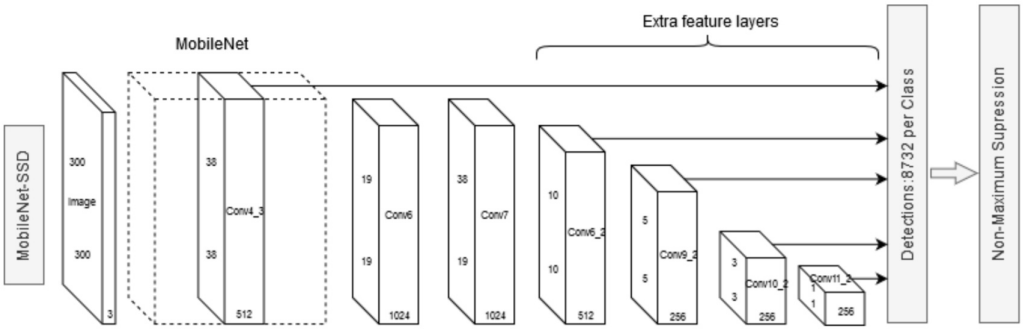
Çizelge 4.3. MobileNet mimarisi (Howard vd., 2017)

Katman Tipi / Kaydırma Sayısı	Filtre Boyutu	Giriş Boyutu
Conv / s2	3 x3 x3 x32	224 x 224 x 3
Conv dw / s1	3 x 3 x 32 dw	112 x 112 x 32
Conv / s1	1 x 1 x 32 x 64	112 x 112 x 32
Conv dw / s2	3 x 3 x 64 dw	112 x 112 x 64
Conv / s1	1 x 1 x 64 x 128	56 x 56 x 64
Conv dw / s1	3 x 3 x 128 dw	56 x 56 x 128
Conv / s1	1 x 1 x 128 x 128	56 x 56 x 128
Conv dw / s2	3 x 3 x 128 dw	56 x 56 x 128
Conv / s1	1 x 1 x 128 x 256	28 x 28 x 128
Conv dw / s1	3 x 3 x 256 dw	28 x 28 x 256
Conv / s1	1 x 1 x 256 x 256	28 x 28 x 266
Conv dw / s2	3 x 3 x 256 dw	28 x 28 x 256
Conv / s1	1 x 1 x 256 x 512	14 x 14 x 256
5x Conv dw / s1	3 x 3 x 256 dw	14 x 14 x 512
Conv / s1	1 x 1 x 512 x 512	14 x 14 x 512
Conv dw / s2	3 x 3 x 512 dw	14 x 14 x 512
Conv / s1	1 x 1 x 512 x 1024	7 x 7 x 512
Conv dw / s2	3 x 3 x 1024 dw	7 x 7 x 1024
Conv / s1	1 x 1 x 1024 x 1024	7 x 7 x 1024
Avg Pool / s1	Pool 7 x 7	7 x 7 x 1024
FC / s1	1024 x 1000	1 x 1 x 1024
Softmax / s1	Classifier	1 x 1 x 1000

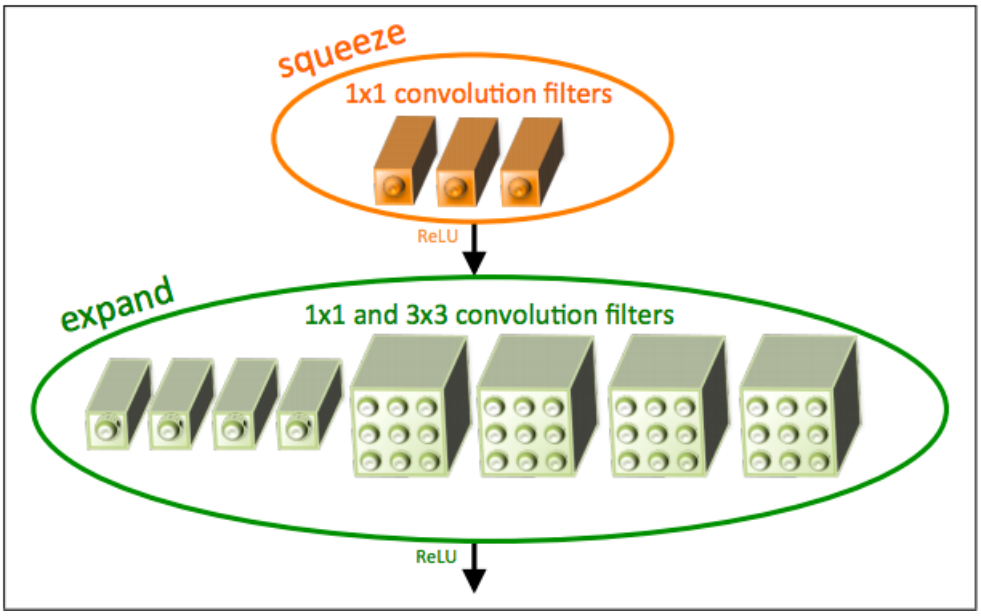
module) adında yeni bir blok önerilmiştir. Bu sayede ImageNet veri setinde 50 kat daha az parametre ile daha yüksek doğruluklar elde edilmiştir. Ateşleme modülünde, sıkıştırma katmanı için ReLU aktivasyon fonksiyonuna bağlı 1x1 filtreler kullanılmıştır. Yine ReLU aktivasyon fonksiyonuna bağlı genişleme katmanı ise 1x1 ve 3x3 filtrelerden oluşur. SqueezeNet ağı, bu ateşleme modüllerinin art arda bağlanmasıyla oluşturulmuştur (Iandola vd., 2016). Ateşleme modülünün yapısı Şekil 4.6'da, SqueezeNet'in yapısı ise Şekil 4.7'de verilmiştir.

4.5. Derin Öğrenme Modeli Eğitiminde Kullanılan Veri Setleri

Trafik işareti tanıma problemleri için oluşturulmuş birçok açık kaynak veri seti bulunmaktadır. Bu veri setleri hem oluşturulan sistemin test edilmesi hem de yapay zeka gibi öğrenen sistemlerin eğitilmesinde kullanılmaktadır. Çalışmamızda GTSDB (German Traffic Sign Detection Benchmark) veri seti (Houben vd., 2013), Çin Trafik İşareti Veri Tabanı (Chinese Traffic Sign Database) (Huang, 2012) ve kendi topladığımız veriler kullanılarak bir veri seti oluşturulmuştur. GTSDB veri seti, Almanya'da



Şekil 4.5. MobileNet mimarisi (NVIDIA, 2023)



Şekil 4.6. Ateşleme modülü (Iandola vd., 2016)

kullanılan trafik işaret tabelalarının görüntülerini içermektedir. Bu veri seti, hız limiti tabelalarının yanı sıra diğer trafik işaretlerini de kapsamaktadır. Çin Trafik İşareti Veri Tabanı ise Çin'deki trafik işaretlerini içermekte olup, farklı coğrafi bölgelerdeki işaretlerin çeşitliliğini artırmak için kullanılmıştır. Kendi topladığımız veriler ise araç içi kameralarla kaydedilen videolardan elde edilen hız limiti tabelası görüntülerinden oluşmaktadır. Bu veri setleri birleştirilerek, hız limiti tabelalarının tespiti ve sınıflandırılması için kapsamlı bir veri seti oluşturulmuştur. Kullanılan veri setleri içerisinde hız limiti tabelalarının yanı sıra diğer tabelalar da bulunduğu için yalnızca hız limiti tabelaları olan görüntüler ayıklanmıştır. Kullanılan bu veri setlerindeki toplam görüntü sayıları ve içinde hız limiti tabelası olan görüntü sayıları Çizelge 4.4'te verilmiştir. Oluşturulan bu karma veri seti "Limit Veri Seti" olarak adlandırılmıştır. Limit veri seti içerisinde 10 ile 120 arasında değişen hız limiti tabelaları bulunan toplam

layer name/type	output size	filter size / stride (if not a fire layer)	depth	s_{1x1} (#1x1 squeeze)	e_{1x1} (#1x1 expand)	e_{3x3} (#3x3 expand)	s_{1x1} sparsity	e_{1x1} sparsity	e_{3x3} sparsity	# bits	#parameter before pruning	#parameter after pruning
input image	224x224x3										-	-
conv1	111x111x96	7x7/2 (x96)	1				100% (7x7)			6bit	14,208	14,208
maxpool1	55x55x96	3x3/2	0									
fire2	55x55x128		2	16	64	64	100%	100%	**33%**	6bit	11,920	5,746
fire3	55x55x128		2	16	64	64	100%	100%	**33%**	6bit	12,432	6,258
fire4	55x55x256		2	32	128	128	100%	100%	**33%**	6bit	45,344	20,646
maxpool4	27x27x256	3x3/2	0									
fire5	27x27x256		2	32	128	128	100%	100%	**33%**	6bit	49,440	24,742
fire6	27x27x384		2	48	192	192	100%	**50%**	**33%**	6bit	104,880	44,700
fire7	27x27x384		2	48	192	192	**50%**	100%	**33%**	6bit	111,024	46,236
fire8	27x27x512		2	64	256	256	100%	**50%**	**33%**	6bit	188,992	77,581
maxpool8	13x12x512	3x3/2	0									
fire9	13x13x512		2	64	256	256	**50%**	100%	**30%**	6bit	197,184	77,581
conv10	13x13x1000	1x1/1 (x1000)	1				**20%** (3x3)			6bit	513,000	103,400
avgpool10	1x1x1000	13x13/1	0									
	activations		parameters				compression info				1,248,424 (total)	**421,098** (total)

Şekil 4.7. SqueezeNet mimarisi (Iandola vd., 2016)

1025 görüntüden oluşmaktadır. Bu görüntüler içerisindeki hız limiti tabelalarının (bir görüntüde birden fazla hız limiti tabelası olabilir) hız limitine göre dağılımları Çizelge 4.5'te verilmiştir. Limit veri setine ait örnek görüntüler ise Şekil 4.8'de verilmiştir.

Çizelge 4.4. Kullanılan veri setleri ve toplam görüntü sayıları

Veri Seti	**Görüntü Sayısı**	**Hız Limiti Tabelası Olan Görüntü Sayısı**
GTSDB	900	143
Çin Trafik İşareti Veri Tabanı	9898	622
Kendi Topladığımız Veriler	260	260
Toplam	**11.058**	**1.025**

Çizelge 4.5. Oluşturulan veri setindeki etiketler ve sayıları

Hız Limiti (km/sa)	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**	**120**
Adet	5	12	174	143	251	136	147	127	22	41	19

Yapay zeka modelini eğitmek için 1025 adet görüntünün yetersiz olacağı düşünüldüğünden alternatif arayışına gidilmiştir. Bu arayışta, Berkeley Üniversitesi Yapay Zeka Araştırma Laboratuvarı'nın (Berkeley Artificial Intelligence Research Lab - BAIR) otonom sürüş ve benzer görevler için hazırladığı BDD100K veri setine erişilmiştir. BDD100K veri seti, araç içi kameralardan çekilmiş videolardan elde edilen 100 bin adet görüntüden oluşmaktadır. Bu görüntüler nesne tanıma, segmentasyon ve şerit takibi gibi 10 farklı görev için düzenlenmiş ve etiketlenmiştir. BDD100K veri setinin nesne algılama için hazırlanan kısmında trafik ışığı, trafik işareti, araba, yaya, otobüs, kamyon, sürücü, bisiklet, motosiklet, tren, diğer taşıtlar, diğer insanlar ve

(a) Limit veri setinden bir görüntü

(b) Limit veri setinden bir görüntü

(c) Limit veri setinden bir görüntü

(d) Limit veri setinden bir görüntü

Şekil 4.8. Limit veri setinden dört farklı görüntü

römork olmak üzere 13 farklı nesne sınıfı bulunmaktadır. Her bir etikete ait görüntü sayıları Çizelge 4.6'da verilmiştir. Görüldüğü üzere toplam 1.460.825 etiketten yalnızca 272.994 tanesi trafik işaretidir ve bunların da küçük bir kısmı hız limiti tabelalarından oluşmaktadır.

BDD100K veri setinden elde edilen hız limiti tabelaları görüntü sayısının çalışmamız için yetersiz olacağı düşünülerek bu kapsamlı veri setinden maksimum fayda sağlayabileceğimiz transfer öğrenimi ile eğitimin iyileştirilmesi yoluna gidilmesine karar verilmiştir. Transfer öğrenimi, daha önce kapsamlı bir veri seti ile eğitilmiş bir derin öğrenme modelinin, benzer veya farklı bir görev için daha kısıtlı bir veri seti ile eğitilerek önceki veri seti ile elde edilen öznitelik ve kalıpların yeni göreve uyarlanması olarak tanımlanabilir. Bu sayede kısıtlı bir veri seti ile elde edilecek başarı, kapsamlı bir veri setinin kullanılması sayesinde artırılabilir (Shao vd., 2015).

Sonuç olarak BDD100K veri seti, transfer öğrenme aşamasında derin öğrenme modelinin ilk eğitiminde kullanılmıştır. Yukarıda tanımladığımız *Limit Veri Seti* ise derin öğrenme modelinin hız limiti tabelalarını algılama yeteneğinin geliştirilmesi için kullanılmıştır. Veri setlerindeki hız limiti tabelası içeren görüntü sayısının azlığı yanında karşılaştığımız bir diğer problem ise kullanılan modellerin orijinal giriş

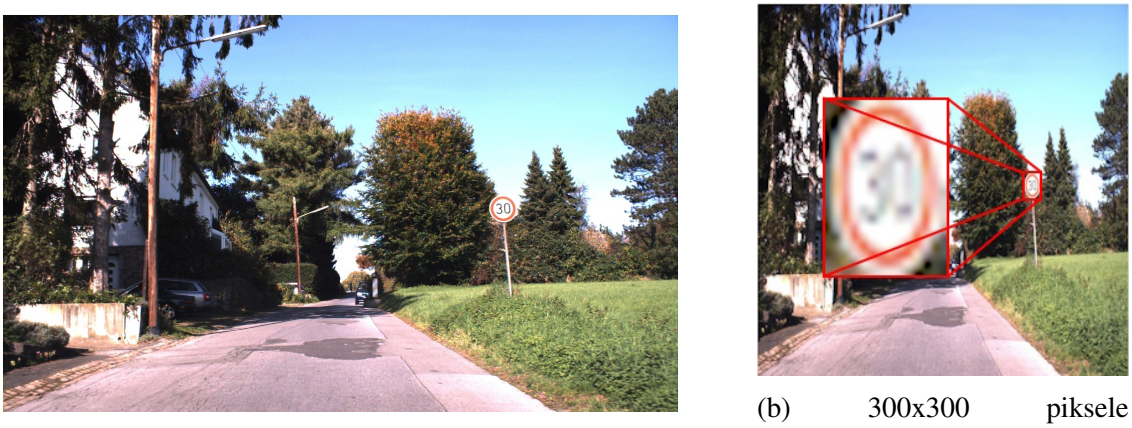
Çizelge 4.6. BDD100K veri setindeki etiketler ve sayıları

Kategoriler	**Adet**
Trafik ışığı	214.755
Trafik işareti	272.994
Araba	803.540
Yaya	105.584
Otobüs	13.637
Kamyon	32.135
Sürücü	5.218
Bisiklet	8.163
Motorsiklet	3.483
Tren	143
Diğer taşıtlar	889
Diğer insanlar	211
Römork	73
Toplam	1.460.825

çözünürlüklerinin veri setlerinde verilen görüntü çözünürlüklerine göre oldukça küçük olmasıdır. Bölüm 4.4.1'de anlatılan MobilNet için giriş çözünürlüğü 300x300 piksel, Bölüm 4.4.2'de anlatılan SqueezeNet için ise giriş çözünürlüğü 224x224 pikseldir.

Eğitim sırasında kullanılan görüntülerin çözünürlükleri 1280x720 piksele kadar ve kameradan alınacak görüntülerin çözünürlükleri 640x480 pikseldir ve bu değerler model giriş çözünürlüklerinden oldukça yüksektir. Bunun için hem eğitim hem de kameradan alınan görüntülerin çözünürlüklerinin modelin giriş çözünürlüğüne indirilmesi gerekmektedir. Ancak bu durumda orijinal resim içerisindeki tüm nesneler küçülecek ve detaylar kaybedilecektir. Örneğin Şekil 4.9a'da verilen 1280x720 çözünürlükteki bir görüntüde 50x50 piksel boyutunda yer kaplayan bir hız limiti tabelası orijinal görüntü 300x300 piksele düşürüldüğünde (4.9b) kapladığı alan yaklaşık 11x20 piksele inecektir. Bu durum çok fazla detayın kaybolmasına ve sistemin bu nesne için yeterli öznitelik çıkaramamasına yol açarak model başarısını düşürebilir. Bu yüzden çalışmanın transfer öğrenme aşamasında BDD100K veri setinde orijinal görüntü içerisinde 50x50 pikselden büyük yer kaplayan etiketlere sahip görüntüler kullanılmıştır. Bu sayede aşırı çözünürlük ve detay kaybı önlenmiştir.

Eğitiminden önce veri setini daha da zenginleştirmek amacıyla hem BDD100K veri seti hem de limit veri seti üzerinde bir görüntüde işaretlenmiş her nesne için yeni bir görüntü kırpma yoluyla elde edilmiştir. Kırpma işlemi nesnenin toplam alanı tüm görüntünün toplam alanının %5'i olacak şekilde ve nesnenin orijinal resim içerisin deki konumu oransal olarak korunarak gerçekleştirilmiştir. Şekil 4.10a'da verilen orijinal bir



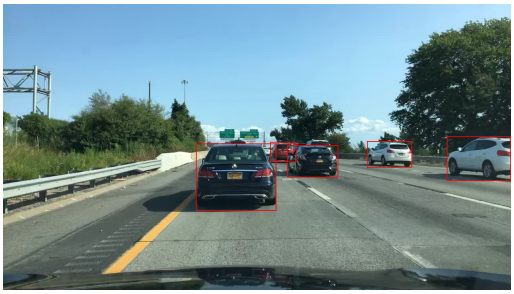
(a) Orijinal boyutlu görüntü

(b) 300x300 piksele küçültülmüş görüntü

Şekil 4.9. Orijinal ve 300x300 piksele küçültülmüş görüntülerin karşılaştırması

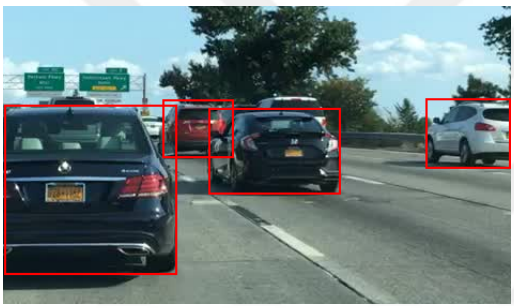
görüntüden elde edilmiş kırpılmış görüntüler Şekil 4.10b, Şekil 4.10c, Şekil 4.10d, Şekil 4.10e ve Şekil 4.10f'de verilmiştir. Daha sonra yeni resimdeki nesneler için tekrar etiketleme yapılarak veri seti sentetik olarak çoğaltılmıştır. Sonuç olarak BDD100K veri setindeki görüntü sayısı 79.863'ten 205.872'ye, limit veri setindeki görüntü sayısı 1.025'ten 1.077'ye çıkarılmıştır.

Her veri setinin farklı bir etiketleme notasyonuna sahip olması sebebiyle veri setleri üzerinde uyguladığımız diğer bir işlem veri setleri etiketlerinin standart hale getirilmesidir. Bunun için bütün görüntülerin etiketleme işlemi Microsoft firmasının Visual Object Tagging Tool (VoTT) adlı yazılımı kullanılmıştır. Bu programda görüntüler içerisindeki ilgili bölgeler sınırlayıcı kutu içerisine alınarak uygun etiket ismiyle etiketlenmiştir. Daha sonra tüm veri seti ve etiketleri, eğitimde kullanılan kütüphanelerin desteklediği format olan Pascal VOC formatına dönüştürülmüştür. BDD100K veri seti için ise basit bir Python betiği hazırlanarak etiketler VOC formatına dönüştürülmüştür.



(a) Kırpılmamış görüntü

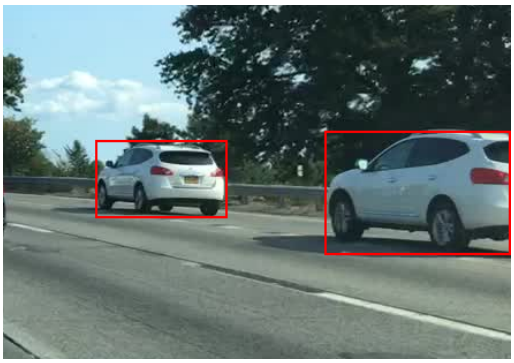
(b) Kırpılmış görüntü 1



(c) Kırpılmış görüntü 2

(d) Kırpılmış görüntü 3

(e) Kırpılmış görüntü 4



(f) Kırpılmış görüntü 5

Şekil 4.10. Kırpılmamış ve kırpılmış görüntülerin karşılaştırması

5. ARAŞTIRMA SONUÇLARI VE TARTIŞMA

Bu bölümde, geliştirilen hız limiti tabelası tespit sisteminin eğitim süreçleri ve performans değerlendirme sonuçları ayrıntılı olarak sunulmaktadır. İlk olarak, sistemin gerçek zamanlı çalışabilmesi için MobileNet ve SqueezeNet modellerinin farklı çözünürlüklerdeki performansları analiz edilerek en uygun model seçimi yapılmıştır. Ardından seçilen modelin iki aşamalı olarak gerçekleştirilen eğitim detayları verilmiştir. Son olarak, sistemin performansı araç içi kamera videoları kullanılarak test edilmiştir. İlk test başarısının beklenenin altında olması sebebiyle görüntüler modele verilmeden uygulanacak bir ön işleme stratejisi geliştirilmiş ve sistem doğruluğu artırılmıştır.

5.1. Model Seçimi

Eğitime başlamadan önce kullanılacak modelin seçilebilmesi amacıyla, kullanılması düşünülen iki faklı modelin faklı çözünürlüklerdeki gerçek zaman performansları test edilmiştir. Giriş çözünürlüğü arttıkça başarının artacağı ancak gerçek zaman performansları düşeceği bilinmektedir. Bu yüzden yeterli gerçek zaman performansı verebilen en yüksek çözünürlüklü model konfigürasyonu seçilmeye çalışılmıştır. Bu işlemin eğitimden önce yapılmasının sebebi eğitim işleminin oldukça uzun zaman alması ve tüm kombinasyonların eğitilip karşılaştırma yapılabilmesi için oldukça uzun zaman harcanması gerektiğidir.

Kullanmayı düşündüğümüz iki model MobileNet ve SqueezeNet'in gerçek zamanlı performanslarını karşılaştırabilmek için modeller henüz eğitilmemiş iken çalıştırılmıştır. Bu sayede her iki modelin her bir kareyi işleme süreleri bulunmuş ve buna göre gerçek zamanda kaç kare/saniye hızında çalışabileceği tespit edilmiştir. Bu testler her iki model için aynı çözünürlük değerlerinin kullanılması koşuluyla birden fazla çözünürlük için yapılmıştır. Yapılan testlerin sonucu Çizelge 5.1'de verilmiştir.

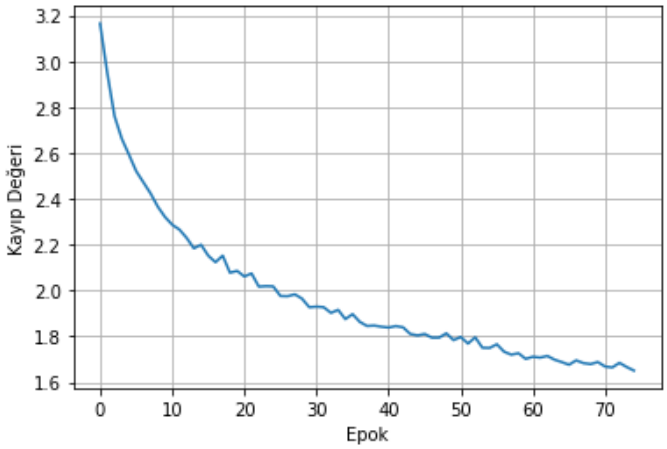
Çizelge 5.1. MobilNet ve SqueezeNet için farklı çözünürlüklerdeki gerçek zaman performansı karşılaştırması

	MobileNet (fps)	**SqueezeNet (fps)**
300x300	41	120
360x360	30	60
512x512	15	40
600x600	10	29

Testler sonucunda, 30 fps video çekim hızına sahip kameramızla uyumlu çalışabilecek en uygun modelin 600x600 piksel çözünürlüğünde SqueezeNet olduğu anlaşılmıştır. Bu yüzden 600x600 piksel çözünürlüğünde SqueezeNet modelinin eğitilmesine karar verilmiştir.

5.2. Eğitim

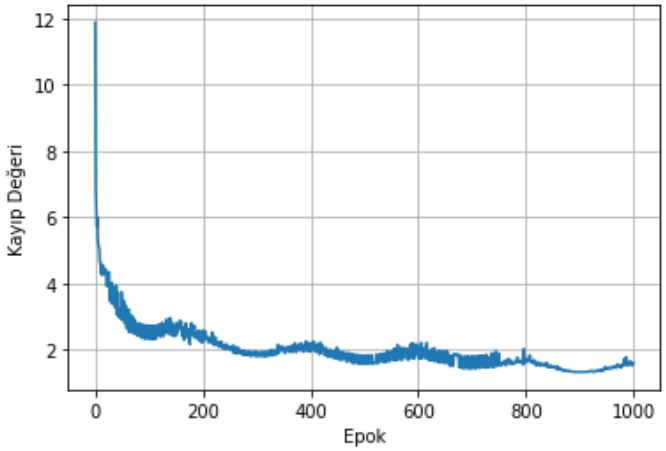
Bu çalışmada, SqueezeNet derin öğrenme modeli, hız limiti tabelalarının tespiti ve sınıflandırılması amacıyla iki aşamalı bir eğitim sürecinden geçirilmiştir. Eğitimin ilk aşaması, BDD100K veri seti kullanılarak modelin genel araç içi kamera görüntülerindeki nesneleri algılama yeteneğini geliştirmeyi hedeflemiştir. Bu kapsamlı eğitim, modelin trafikte karşılaşılabilecek çeşitli nesnelerin (arabalar, yayalar, trafik ışıkları, diğer işaretler vb.) özniteliklerini çıkarmasını sağlamıştır. Bu sayede, modelin hız tabelalarına özel eğitime geçmeden önce, genel algılama kabiliyetinin güçlendirilmesi ve diğer nesnelerden hız tabelalarını daha etkili bir şekilde ayırt edebilmesi için sağlam bir temel oluşturulmuştur. Bu aşamada BDDK100 veri setindeki toplam 205.872 görüntüden 148.228 adedi eğitim, 37.056 adedi doğrulama ve 20.588 adedi test için ayrılmıştır. İlk aşama eğitimi, 75 epok boyunca sürdürülmüştür. Şekil 5.1'de birinci aşama eğitim kayıp grafiği verilmiştir.



Şekil 5.1. Birinci aşama eğitim kayıp grafiği

Birinci aşamada eğitilen model, ikinci aşama için bir baz model olarak kullanılmıştır. İkinci aşama eğitimi, yalnızca hız tabelalarını içeren limit veri seti kullanılarak gerçekleştirilmiştir. Bu eğitimin amacı, genel nesne algılama yeteneği kazandırılmış modelin, özellikle hız limiti tabelalarını yüksek doğrulukla tespit etme ve

sınıflandırma yeteneğini iyileştirmektir. İkinci aşama eğitiminde limit veri setindeki toplam 1.077 görüntüden 754 adedi eğitim, 215 adedi doğrulama ve 108 adedi test için ayrılmıştır. İkinci aşama eğitimi, 1000 epok boyunca sürdürülmüştür. Şekil 5.2'de ikinci aşama eğitim kayıp grafiği verilmiştir.



Şekil 5.2. İkinci aşama eğitim kayıp grafiği

Tüm eğitim aşamalarında modelin varsayılan (default) parametreleri esas alınmıştır.

5.3. Test

Test işlemleri için model eğitiminde kullanılan görüntüler dışında araç içi kamerasından alınmış toplamda 10 saatlik birden fazla araba yolculuğu sırasında 640x480 çözünürlükte kaydedilen videolar kullanılmıştır. Bu videolarda hız tabelalarının görülme sıklığı oldukça düşüktür. Testlerin hızlı gerçekleştirilebilmesi için bu videolar içerisinde hız tabelasıyla karşılaşılan bölümlerin kesilip art arda eklenmesiyle tek bir videoya dönüştürülmüştür. Görüş alanında hiç tabela yok iken video başlayacak ve tabela görüş alanından tamamen çıkana kadar devam edecek şekilde art arda tüm tabelaların olduğu bir video oluşturulmuştur. Bu montaj işlemi yapılırken tabelaların görülmeye başlamasından önce kesme işlemi başlatılmış, tabela görüşten tamamen çıktıktan sonra kesme işlemi durdurulmuştur. Sonuç olarak içerisinde 317 adet hız limiti tabelası bulunan toplamda 75 dakikalık bir montaj videosu elde edilmiştir. Bu videolar sisteme sanki kameradan geliyormuş gibi bir akış (stream) ile beslenmiş ve gerçek zamanlı olarak işlenmiştir.

Sistemin tanımı yolda bulunan hız limiti tabelalarının tespit edilerek hız limitinin

belirlenmesi olduğu için başarı kriteri de buna uygun olarak seçilmelidir. Burada başarı kriteri olarak (Changzhen vd., 2016) tarafından kullanılan yöntem temel alınmıştır. Sistemimiz canlı görüntü içerisinde hız limiti tespit etmeyi hedeflediği için başarı kriteri, kameradan alınan video akışı içerisindeki hız limiti tabelasının tespit edilerek doğru sınıflandırılması ve bu işlemin gerçek zamanda yapılması şeklinde belirlenmiştir. Yolda hız tabelasının olduğu bölümde, tabelanın varlığı tespit edilerek sınıflandırması doğru yapılabildiği takdirde başarılı kabul edilmiştir. Hız tabelası tespit edilip sınıflandırması yanlış olduğu takdirde kısmi başarılı, hız tabelası olmadığı hâlde hız tabelası tespit edildiği takdirde kısmi başarısız, hız tabelası olduğu hâlde tespit edilemediği takdirde ise başarısız kabul edilmiştir. Başarı oranı hesaplanırken, başarılı tespit sayısı toplam hız limiti tabelası sayısına bölünerek yüzde değer olarak hesaplanmıştır.

Yukarıdaki değerlendirme kriterlilerine göre üretilen montaj videosu üzerinde yapılan test sonuçları Çizelge 5.2'de verilmiştir. Görüldüğü gibi, başarı oranı beklenenden daha düşük çıkmıştır. Eğitim sırasında öngörülen ve başarıyı etkilediği düşünülen olgulara karşı alınan önlemler faydalı olsa da tek başına yeterli gelmediği anlaşılmaktadır. Tespit edilmek istenen nesne görüntü içerisinde toplam alanın küçük bir kısmını kaplıyor ise çözünürlük kaybından dolayı detay kaybı yaşanmakta ve başarı olumsuz yönde etkilenmektedir. Eğitim sırasında nesnelerin alanının toplam alana oranı belli bir seviyenin üstünde kalacak şekilde orijinal resim kırpılarak bu sorunun önüne geçilmeye çalışılmıştır. Bu işlem, nesnenin konumunun eğitim setinde tam olarak bilinmesi sayesinde yapılabilmiştir. Aynı strateji testler sırasında uygulanabilirse başarının artacağı düşünülerek kameradan alınan görüntü üzerinde bazı ön işlemler yapılarak sistem başarısı artırılmaya çalışılmıştır.

Çizelge 5.2. Video test sonuçları

Tespit Sonucu	**Görüntü Sayısı**	**%**
Başarılı	190	%59,9
Kısmi Başarılı	25	%7,9
Kısmi Başarısız	1	%0,3
Başarısız	101	%31,9
TOPLAM	**317**	**%100**

Şekil 5.3'te görüleceği gibi bir hız limiti tabelası genellikle yolun iki tarafında ve ufuk çizgisine yakın bir hizada olmaktadır. Araç ilerledikçe resim içerisinde perspektiften dolayı resmin sağ ve sol kenarlarına doğru hareket etmekte ve bu süreçte yine perspektiften dolayı gittikçe büyümektedir. Tabela görüş alanından çıkmadan

hemen önce ekranın iki yanında ve en büyük haliyle bulunmaktadır. Bu durumdan faydalanmak için alınan video görüntü karesinde görüntünün sağında ve solunda hız limiti tabelasının en çok görüleceği bölgelerin (Şekil 5.3'te yeşil çizge ile gösterilen bölgeler) kesilip birleştirilmesiyle yeni bir görüntü oluşturularak derin öğrenme modeline verilmiştir. Böylece hem tabelanın maksimum çözünürlüklü hali elde edilmiş hem de oluşturulan yeni görüntünün çözünürlüğü orijinal kareden daha küçük olduğu için sisteme girerken daha az bir oranda küçültülmüştür. Ön işlemeden geçirilmiş örnek görüntüler Şekil 5.4'te görülebilir.

(a) Tabelanın görüntü karesindeki hareketi - 1

(b) Tabelanın görüntü karesindeki hareketi - 2

(c) Tabelanın görüntü karesindeki hareketi - 3

(d) Tabelanın görüntü karesindeki hareketi - 4

Şekil 5.3. Tabelanın görüntü karesindeki hareketi

(a) Ön işlemeden geçirilmiş görüntü - 1

(b) Ön işlemeden geçirilmiş görüntü - 2

(c) Ön işlemeden geçirilmiş görüntü - 3

Şekil 5.4. Ön işlemeden geçirilmiş görüntü örnekleri

Üretilen montaj videosu model önüne eklenen yeni ön işlemeyle birlikte tekrar test edildiğinde Çizelge 5.3'te verilen sonuçlar elde edilmiştir. Testler sonucunda, uygulanan ön işleme stratejisinin sistem performansını başarılı bir şekilde artırdığı açıkça görülmektedir. Gerçek zamanlı olarak çalıştırılan ve işlem gücü görece kısıtlı olan sistemimizde hem yeterli hızlar elde edilmiş hem de %95,9 oranında başarılı sonuçlar alınmıştır.

Çizelge 5.3. İlgi alanı belirlendikten sonra video test sonuçları

Tespit Sonucu	**Görüntü Sayısı**	**%**
Başarılı	304	%95,9
Kısmi Başarılı	10	%3,1
Kısmi Başarısız	0	%0
Başarısız	3	%1
TOPLAM	**317**	**%100**

6. SONUÇ VE ÖNERİLER

Bu tez çalışmasında araç içerisine yerleştirilen bir kamera ile yol hız tabelalarının tespiti yapılmıştır. Başarı kriteri, kameradan alınan video akışı içerisindeki hız limiti tabelasının tespit edilerek doğru sınıflandırması ve bu işlemi gerçek zamanda yapması olarak belirlenmiştir. Bu başarı kriteri ile araç içerisine konulacak düşük güçte bir donanım ile pratik bir tespit cihazı yapılabileceği gösterilmiştir.

Eğitim için çeşitli yöntemler denenmiştir. Bu yöntemlerin amacı hem en kısa sürede eğitimi gerçekleştirmek, hem de az sayıda eğitim verisi ile başarı kriterlerini sağlayan bir sistem inşa etmektir.

Çalışma sırasında hem düşük güçlü hem maliyet açısından görece ucuz hem de yeterli güce sahip bir donanım olan Jetson Nano kullanılmıştır. Yine görece ucuz bir kamera bağlanarak sistem çalıştırılmış ve başarı kriterlerinin sağlanabildiği gözlemlenmiştir. Ayrıca uygun veri seti ile eğitildiğinde başarının arttığı gözlemlenmiştir. Ayrıca başarıyı arttırabilmek için transfer öğrenim tekniğinden faydalanılmıştır.

Eğitim iki aşamada yapılmıştır. Birinci aşama için BDD100K veri setinden faydalanılmıştır. Bu veri seti kullanılmadan önce etiketeler uygun formata dönüştürülmüştür. Daha sonra sentetik olarak veri artırma işlemi uygulanarak veri seti zenginleştirilmiştir. İkinci aşamada ise hem standart olarak kullanılan GTSDB veri seti ve Çin Trafik İşareti Veri Tabanı hem de araç ile toplanan görüntülerden oluşturulmuş özel bir veri seti kullanılmıştır. Araç ile toplanan veri seti gerçek uygulamaya daha yakın olduğu için başarının arttığı gözlemlenmiştir. Eğitim görece daha güçlü bir bilgisayar üzerinde yapılmış ve elde edilen eğitilmiş model nihai donanım olan Jetson Nano üzerinde gerçek zamanlı olarak çalıştırılmıştır.

Çalışma sırasında kısıtlayıcı unsur olarak veri setinin yetersizliği ve kullanılan kısıtlı donanım gösterilebilir. Ancak yine de seçilen başarı kriterlerine uygun bir sistem elde edilmiştir. Yapılan testlerde **%95,9** başarı oranı elde edilmiştir.

KAYNAKLAR

Alghmghama, D. A., Latif, G., Alghazo, J., ve Alzubaidi, L. (2019). Autonomous traffic sign (atsr) detection and recognition using deep cnn. In *16th International Learning & Technology Conference*.

Andrey, V. ve Jo, K. (2006). Automatic detection and recognition of traffic signs using geometric structure analysis. In *SICE-ICASE International Joint Conference*, Bexco, Busan, Korea.

Changzhen, X., Cong, W., Weixin, M., ve Yanmei, S. (2016). A traffic sign detection algorithm based on deep convolutional neural network. In *2016 IEEE International Conference on Signal and Image Processing (ICSIP)*, s. 676–679.

Clevert, D.-A., Unterthiner, T., ve Hochreiter, S. (2016). Fast and accurate deep network learning by exponential linear units (elus).

Dubey, S. R., Singh, S. K., ve Chaudhuri, B. B. (2022). Activation functions in deep learning: A comprehensive survey and benchmark.

Fištrek, T. ve Lonari, S. (2011). Traffic sign detection and recognition using neural networks and histogram based selection of segmentation method. In *53rd International Symposium ELMAR-2011*, Zadar, Croatia.

Fleyeh, H. (2006). Shadow and highlight invariant colour segmentation algorithm for traffic signs. In *IEEE International Conference on Cybernetics and Intelligent Systems*, Bangkok, Thailand.

Garcia-Garrido, M. A., Sotelo, M. A., ve Martín-Gorostiza, E. (2006). Fast traffic sign detection and recognition under changing lighting conditions. In *2006 IEEE Intelligent Transportation Systems Conference*, Toronto, Canada.

Ghica, D., Lu, S., ve Yuan, X. (1995). Recognition of traffic signs by artificial neural network. In *IEEE International Conf. on Neural Networks*, Perth, Australia.

Goodfellow, I., Bengio, Y., ve Courville, A. (2016). *Deep Learning*. MIT Press. Softmax Units for Multinoulli Output Distributions.

Haque, W. A., Arefin, S., Shihavuddin, A., ve Hasan, M. A. (2021). Deepthin: A novel lightweight cnn architecture for traffic sign recognition without gpu requirements. *Expert Systems With Applications*, 168:114481.

Haykin, S. (1999). *Neural Networks: A Comprehensive Foundation*. Pearson Education, Inc., Hamilton, Ontario, Canada, 2nd edition.

He, K., Zhang, X., Ren, S., ve Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification.

Houben, S., Stallkamp, J., Salmen, J., Schlipsing, M., ve Igel, C. (2013). Detection of traffic signs in real-world images: The German Traffic Sign Detection Benchmark. In *International Joint Conference on Neural Networks*, number 1288.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., ve Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications.

Huang, L. (2012). Chinese traffic sign database. Supported by the National Natural Science Foundation of China (NSFC), Grant No. 61271306.

Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., ve Keutzer, K. (2016). Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <0.5mb model size.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., ve Berg, A. C. (2016a). *SSD: Single Shot MultiBox Detector*, s. 21–37. Springer International Publishing.

Liu, W., Rabinovich, A., ve Berg, A. C. (2016b). Ssd: Single shot multibox detector. In *European Conference on Computer Vision (ECCV)*, s. 21–37. Springer.

Malik, R., Khurshid, J., ve Ahmad, S. (2007). Road sign detection and recognition using colour segmentation, shape analysis and template matching. In *Proceedings of the Sixth International Conference on Machine Learning and Cybernetics*, Hong Kong.

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill Science/Engineering/Math.

NVIDIA (2023). Pytorch ssd: Single shot multibox detector. `https://github.com/dusty-nv/jetson-inference/blob/master/docs/pytorch-ssd.md`. [Accessed: October 2023].

Opel (2023). Opel cars can see: Opel eye camera reads signs, improves safety. `https://www.automobilesreview.com/auto-news/opel-cars-can-see/2309/`. [Ziyaret Tarihi: Haziran 18, 2023].

Resmî Gazete (1985). Trafik işaretleri hakkında yönetmelik, birinci kısım, birinci bölüm, madde 1. Resmî Gazete Tarihi: 19.06.1985, Resmî Gazete Sayısı: 18789.

Shao, L., Zhu, F., ve Li, X. (2015). Transfer learning for visual categorization: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 26(5):1019–1034.

Sharma, S., Sharma, S., ve Athaiya, A. (2017). Activation functions in neural networks. *Towards Data Sci*, 6(12):310–316.

Shi, M., Wu, H., ve Fleyeh, H. (2008). Support vector machines for traffic sign recognition. In *International Joint Conference on Neural Networks*, Hong Kong.

Shustanova, A. ve Yakimov, P. (2017). Cnn design for real-time traffic sign recognition. In *3rd International Conference "Information Technology and Nanotechnology"*, Samara, Russia.

Türkiye İstatistik Kurumu (2022). Karayolu trafik kaza İstatistikleri, 2022. Sayı: 49513, Yayım Tarihi: 25 Mayıs 2023.