YOLO V8 Algoritması ile Otomatik Plaka Tanıma ve Görselleştirme Sistemi

Araştırma Makalesi/Research Article

Fatih GÜL[1,2], Esma SERTTAŞ[1]*

[1]Elektrik-Elektronik Mühendisliği Bölümü, Recep Tayyip Erdoğan Üniversitesi, Rize, Türkiye
[2]Yapay Zekâ-Nesnelerin İnterneti Araştırma Laboratuvarı, Recep Tayyip Erdoğan Üniversitesi, Rize, Türkiye

fatih.gul@erdogan.edu.tr, esma.serttas.7@gmail.com
(Geliş/Received:27.06.2024; Kabul/Accepted:19.10.2024)
DOI: 10.17671/gazibtd.1506041

Özet— Bu çalışma ile, belirli bir mesafeye yerleştirilen bir kamera ile YOLO (You Only Look Once) V8 algoritmasını kullanarak aracın üzerindeki plakayı otomatik olarak tanıyan ve görselleştiren bir sistem tasarlanmıştır. YOLO V8, gelişmiş bilgisayarlı görü yeteneklerine sahip olmakla birlikte doğrudan plaka tanıma modeli içermemektedir. Bu çalışma ile güvenlik önlemleri gerektiren alanlarda insan gücünü ve maliyeti en aza indirerek verimli şekilde kullanılabilir bir model önerilmiştir. Plaka veri seti, bilgisayarlı görü modeli ortamı Roboflow kullanılarak oluşturulmuş ve yapay sinir ağı eğitim modeli geliştirilmiştir. Python programlama dili kullanılarak YOLO V8 algoritması ile yapay sinir ağı modeli Karayolları Trafik Yönetmeliğine uygun TR plakalar ile eğitilerek plaka tanıma işlemleri gerçekleştirilmiştir. Geliştirilen bu sistemde, açık kaynaklı kütüphaneler olan OpenCV, Time, Random, Numpy, Ultralytics ve EasyOCR kullanılmıştır. Kullanıcı arayüzü için Tkinter kullanılarak plaka tanıma sonuçları görselleştirilmiştir. Sistem tam karşıdan, sağ ve sol yönde 30° içerisinde kalacak şekilde farklı açılardan alınan görüntüler üzerinde test edilmiş ve yüksek doğruluk oranları (%99 @ 25 Epok) elde edilmiştir. Bu çalışma, trafik yönetimi, otopark sistemleri ve güvenlik uygulamaları gibi çeşitli alanlarda mevcut YOLOV8 tabanlı uygulamalara entegre edilebilir bir çözüm yöntemi önermektedir.

Anahtar Kelimeler— plaka tanıma sistemi, görüntü işleme, Yolov8, bilgisayarlı görü, yapay sinir ağları

Automatic License Plate Recognition and Visualization System with YOLO V8 Algorithm

Abstract— The aim of this study is to develop a system that automatically recognizes and visualize the license plate on the vehicle using the YOLO (You Only Look Once) V8 algorithm with a camera placed at a certain distance. Although YOLO V8 has advanced computer vision capabilities, it does not have a direct license plate recognition model. With this study, a model which can be used efficiently by minimizing manpower and cost in areas that require security measures, was offered. The license plate dataset was developed using the computer vision model environment Roboflow and an artificial neural network training model was created. The license plate recognition operations have been performed by training a neural network model using the YOLO V8 algorithm in Python with TR plates in accordance with the Highway Traffic Regulation. In this developed system, open source OpenCV, Time, Random, Numpy, Ultralytics and EasyOCR libraries were used. By using Tkinter for the user interface, license plate recognition results were visualized. The system was tested on images taken from different angles within 30° from the front, right and left, and high accuracy rates (99% @ 25 Epoch) were obtained. This study offers practical solutions in various fields such as traffic management, parking systems and security applications that can be integrated into current YOLOV8 based applications.

Keywords— license plate recognition system, image processing, YOLOV8, computer vision, artificial neural networks

1. GİRİŞ (INTRODUCTION)

Son çeyrek yüzyılda araç sayısındaki artış ve trafikte meydana gelen sorunlar, otomatik araç tanıma ve trafik akışının kontrolü üzerine yapılan çalışmaları artırmıştır [1],[2],[3],[4]. Günümüzde trafiği denetlemek amacıyla mikrodalga dedektörleri, yolun altına yerleştirilen tüpler, loop dedektörleri ve radyo frekansları kullanan radarlar gibi çeşitli yöntemler kullanılmaktadır. Ancak bu donanımların pahalı olması, sistemlerin işletimini zorlaştırmaktadır. Gelişen teknolojiyle birlikte dijital görüntü işleme alanında önemli gelişmeler kaydedilmiştir. Dijital görüntü işleme, görüntünün dijital formata dönüştürülerek çeşitli işlemlerle iyileştirilmesi ve bilgi çıkarılması yöntemidir. Dijital görüntü işleme yöntemleri kullanılarak ek donanım gerektirmeden plaka tanıma sistemi geliştirilebilmektedir. Her aracın kendine özgü bir plakası olduğu için, dijital görüntü işleme tekniklerileyle araç tanıma problemi etkili bir şekilde çözülebilmektedir. Türkiye'de plakalar, Karayolları Yönetmeliği tarafından belirlenen standartlara uygun olarak üretilmektedir [5]. Karayolları Yönetmeliği'ne göre standart plakanın resmi Şekil 1'de gösterilmiştir [6]. Plaka gövdesi alüminyumdan yapılmakta olup, araçların hem ön hem de arka kısmında plaka bulunması zorunludur. Araçlarda bulunan yönetmeliğe uygun ön ve arka plakaların resmi Şekil 2'de gösterilmiştir.

Şekil 1. Karayolları Trafik Yönetmeliği'ne göre standart plakanın görüntüsü. (Image of the standard license plate according to the Road Traffic Regulations.)

Şekil 2. Araçlarda bulunan yönetmeliğe uygun plakanın önden ve arkadan görüntüsü. (Front and back view of the regulated license plate on the vehicles.)

Bu projenin amacı, araç plakalarını belirli bir mesafeden kamera kullanarak tanımak ve görüntü işleme algoritmaları ile bu plakaları görselleştirmek üzere bir sistem geliştirmektir. Literatürde plaka tanıma üzerine birçok çalışma bulunmaktadır. Örneğin, Doe v.d. (2019), C# programlama dili ve YOLOv2 algoritmasını kullanarak otomatik plaka tanıma sistemi geliştirmişlerdir [7]. Benzer şekilde, Smith ve Brown (2020), Python programlama dili ve YOLOv3 algoritmasını kullanarak plaka tanıma performansını artırmayı başarmışlardır [8].

Bu çalışmada YOLOv8 algoritması ve Python programlama dili ile geliştirilen açık kaynak kütüphaneleri tercih edilmiştir. YOLO algoritmasının seçilme nedeni, görüntü işlemede nesne tespiti için yaygın kullanılan bir derin öğrenme algoritması olmasıdır. Diğer nesne tespiti algoritmalarından farkı, gerçek zamanlı bir görüntüyü tek bir geçişte işleyerek birden fazla nesneyi tespit etmesidir. Genellikle diğer algoritmalar, bölge tabanlı veya önceden belirlenmiş bölgelerde (örneğin R-CNN, Fast R-CNN vb. algoritmalar) birden fazla geçiş yaptığı için yavaş çalışmaktadır. YOLO algoritması ise diğer yöntemlere göre daha hızlı çalışmaktadır. Çalışmamızda, veri setimiz YOLOv5'in geliştiricileri Ultralytics tarafından oluşturulan ve nesne algılama ile görüntü segmentasyonunda son teknoloji ürünü olan YOLOv8 modeli ile eğitilmiştir [9]. Bu modeli kullanma nedeni, nesneleri daha hızlı algılayabilmesi ve plaka tanıma sistemi üzerinde henüz çalışılmamış olmasıdır. Kullanılan yazılım, öncelikle görüntüdeki aracı tanımlar, ardından aracın plakasının yerini belirler ve plakanın üzerindeki karakterleri tanıyarak plakayı ara yüzde gösterir.

Bu çalışmada, önceki çalışmalardan farklı olarak sadece açık kaynaklı YOLOv8 algoritması ve Python dilinde yazılmış kodlar kullanılmıştır. Görselleştirme için yine açık kaynaklı PyCharm ortamı ve Pyhton kullanılmıştır. Bu sayede YOLOV8 ile geliştirilen kapı kontrolü, yaya tanıma vb. uygulamalara entegre edilebilecek bir yöntem önerilmiştir.

Ayrıca mevcut veri setleri kendi elde ettiğimiz, yönetmeliğe uygun TR plakalı görsellerle eğitilerek; mevcut veri setindeki plakaların Türkiye plakalarına uygun hale getirilmesi sağlanmıştır. Kullandığımız görüntü iyileştirme yöntemleriyle sadece karşıdan alınan görüntülerden plaka tanıma işlemi gerçekleştirilmemiş ayrıca sağ ve sol yönde 30º açı içinde alınan görüntülerin de yüksek doğrulukla tanınması sağlanmıştır. Bu sayede plaka tanıma işleminde kullanılacak kameraların tam karşı açıda yüksek direklere yerleştirilmesine göre daha kolay olan zemine yakın yerlerde de uygulanabilirliği gösterilmiştir.

Önerilen yöntemle; ilerleyen süreçte araçların plaka seviyesine yerleştirecek olan veya halihazırda mevcut olan görüş kameralarının aynı zamanda diğer araçları tanıması sağlanarak; Edge-IoT ,Edge-AI tabanlı uygulamalara da katkı sağlayabileceği değerlendirilmektedir.

2. YÖNTEM (METHOD)

Bu çalışmada plaka tanıma sistemi geliştirmek için kullanılan veri seti, çeşitli açılardan ve farklı ışık koşullarında çekilmiş Open Images Dataset depo alanlarından elde edilen araç görselleri ile gerçek zamanlı çekilen araç görüntülerinden oluşturulmuştur. Şekil 3'te,

gerçek zamanlı araç görüntülerinden iki örnek gösterilmiştir. Elde edilen araç görüntülerini kullanarak veri seti haline dönüştürmek için Roboflow programı kullanılmıştır. Veri setini çeşitlendirmek için Roboflow'da bulunan veri artırma teknikleri kullanılmış ve bu sayede veri setinin kalitesi artırılmıştır.

Şekil 3. Veri setinde kullanılan örnek görüntüler. (Sample images used in the dataset.)

Çalışmada YOLOv8 algoritması kullanılmıştır. YOLOv8, önceki YOLO versiyonlarına göre daha iyi performans göstermektedir. Geleneksel nesne tespiti yöntemlerinden en büyük farkı birçok gerçek zamanlı nesne tespiti yapabilmesidir. Bu çalışma özelinde görüntü üzerinden araç ve plaka ayrımını eşzamanlı gerçekleştirebilme amacıyla tercih edilmiştir.

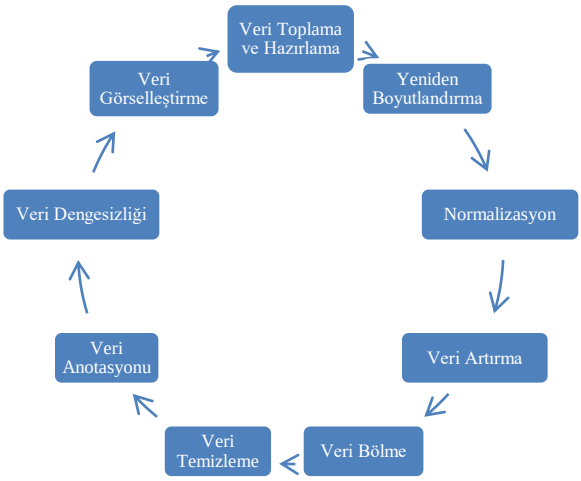
Plaka tanıma sisteminin geliştirilmesi için Python programlama dili kullanılmıştır. Çalışmada kullanılan açık kaynak kütüphaneler şunlardır:

- **OpenCV:** Görüntü işleme ve analiz işlemlerini gerçekleştirmek için kullanılmıştır. OpenCV, Dr. Gary Bradski tarafından başlatılmış ve Intel tarafından geliştirilmiştir. OpenCV, geniş bir görüntü işleme fonksiyonları yelpazesi sunan açık kaynaklı bir kütüphanedir [10].
- **Time:** İşlemler arasındaki gecikmeleri hesaplamak ve zaman ölçümleri yapmak için kullanılmıştır. Python'un standart kütüphanesi olan Time modülü, Python Software Foundation tarafından geliştirilmiştir [11].
- **Random:** Rastgele veri oluşturma işlemlerinde kullanılmıştır. Python'un standart kütüphanesi olan Random modülü, Python Software Foundation tarafından geliştirilmiştir [12].
- **Numpy:** Sayısal hesaplamalar ve veri manipülasyonu için kullanılmıştır. NumPy, Travis Oliphant tarafından başlatılmıştır ve açık kaynaklı bir topluluk tarafından geliştirilmiştir [13].
- **Ultralytics:** YOLOv8 modelini kullanarak nesne tespiti gerçekleştirmek için kullanılmıştır. Ultralytics, Glenn Jocher tarafından geliştirilmiştir ve YOLOv8 gibi ileri seviye nesne algılama modelleri sağlar [14].
- **Os:** Dosya ve dizin işlemlerini yönetmek için kullanılmıştır. Python'un standart kütüphanesi olan Os modülü, Python Software Foundation tarafından geliştirilmiştir [15].
- **Tkinter:** Kullanıcı arayüzü geliştirmek için kullanılmıştır. Tkinter, Python'un standart GUI kütüphanesidir ve John Ousterhout tarafından geliştirilmiş Tcl/Tk'ye dayanmaktadır [16].
- **PIL (Pillow):** Görüntü işleme ve manipülasyon işlemlerinde kullanılmıştır. PIL, Fredrik Lundh tarafından geliştirilmiş ve Pillow, Alex Clark ve diğer gönüllüler tarafından sürdürülen bir fork'tur [17].
- **EasyOCR:** Optik karakter tanıma (OCR) işlemlerini gerçekleştirmek için kullanılmıştır. EasyOCR, Jaided AI tarafından geliştirilmiştir ve OCR işlemleri için geniş bir dil desteği sunar [18].

Yazılımlar Pycharm geliştirme ortamında yazılmış ve plaka tanıma sisteminin tüm kodları bu ortamda geliştirilip test edilmiştir.

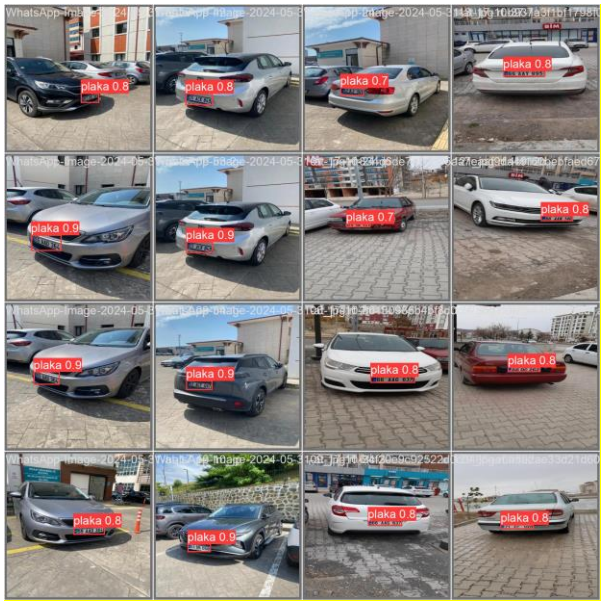
2.1. Veri Hazırlama ve Geliştirme Süreci (Data Preparation and Development Process)

Çalışmada kullanılan veri setinin model eğitimine hazırlanması için gerçekleştirilen adımlar Şekil 4'te şema halinde gösterilmiştir.



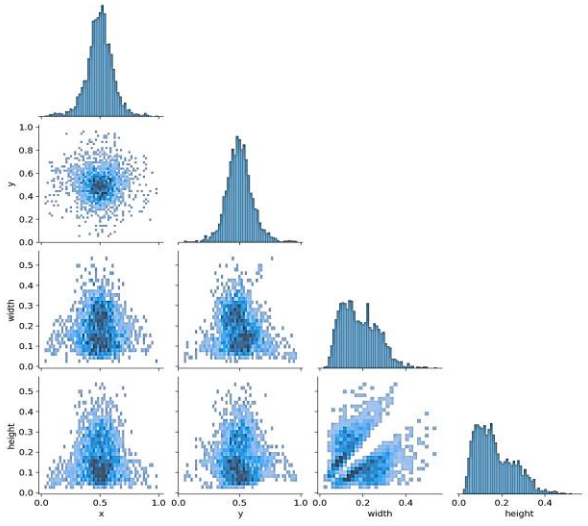
Şekil 4. Veri setinin model eğitiminde kullanılan aşamalar. (Stages used in model training of the dataset.)

Veri kaynakları olarak, çeşitli hava koşullarında ve farklı açılardan çekilmiş araç plakalarını içeren bir veri seti kullanılmıştır. Bu veri seti, gerçek zamanlı araç görüntülerinden ve çeşitli kaynaklardan toplanarak zenginleştirilmiştir (Şekil 5). Bu şekilde, modelin farklı çevresel koşullarda ve açılarda plaka tanıma performansını artırmayı hedeflemektedir. Plakaların doğru şekilde tanınabilmesi için veri seti manuel olarak etiketlenmiştir. Etiketleme işlemi, her görüntüdeki plakanın konumunu ve metin bilgisini içermektedir. Bu, modelin eğitim sürecinde doğru etiketler ile öğrenmesini sağlamak için kritik bir adımdır.



Şekil 5. Veri setinin model eğitiminde ve etiketlemede kullanılan çeşitli görüntüler. (Various images used in model training and labeling of the dataset.)

Görüntüler, YOLOv8 modeline uygun 640x640 piksel boyutlarına yeniden boyutlandırılmıştır. Bu adım, modelin giriş boyutları ile uyumlu hale getirilmesi için gereklidir. Ayrıca, görüntü piksel değerleri 0-255 aralığından 0-1 aralığına normalize edilmiştir. Normalizasyon, modelin daha iyi öğrenebilmesi ve daha hızlı eğitile bilmesi için önemlidir. Bu işlem, modelin farklı ışık koşullarına karşı daha duyarlı ve genelleme yeteneği yüksek bir performans sergilemesini sağlar. Veri etiklemede elde edilen bir korelogram Şekil 6 da sunulmuştur.



Şekil 6. Veri etiketlemede kullanılan örnek korelogram. (A sample correlogram to be used in labelling.)

Eğitim veri setinin çeşitliliğini artırmak ve modelin genelleme yeteneğini geliştirmek için çeşitli veri artırma teknikleri kullanılmıştır. Bu teknikler arasında çevirme (görüntülerin yatay ve dikey olarak çevrilmesi), döndürme (rastgele açılarda döndürülmesi), gri tonlama (gri tonlama uygulanarak işlenmesi), doygunluk seviyelerinin değiştirilmesi (renk doygunluğunun artırılması veya azaltılması), bulanıklık eklenmesi (Gaussian blur gibi tekniklerle bulanıklaştırma), rastgele gürültü eklenmesi (salt and pepper noise veya Gaussian noise), maruziyet seviyelerinin ayarlanması (aşırı pozlanmış veya yetersiz pozlanmış görüntülerin oluşturulması) ve görüntülerin 90 derece döndürülmesi yer almaktadır. Veri iyileştirme ve döndürmede kullanılan farklı kalite ve açılara sahip çeşitli örnek görüntüler Şekil 7 da sunulmuştur.

Şekil 7. Veri iyileştirme ve döndürmede kullanılan farklı kalite ve açılara sahip çeşitli örnek görüntüler. (Various sample images with different quality and angles used for rotation and data enhancement.)

Bu veri artırma teknikleri, modelin eğitim sürecinde daha geniş bir veri çeşitliliği ile karşılaşmasını sağlayarak genelleme yeteneğini artırmaktadır. Bu teknikleri kullanarak veri setinden alınan (Şekil 8-Sol) görüntünün veri artırma ve çevirme ile iyileştirilmiş hali Şekil 8(Sağ)'da gösterilmiştir.

Şekil 8. Veri setinde bulunan görüntüden (sol) veri artırma ve 90º çevirme teknikleri kullanarak oluşturulan örnek görüntü (sağ). (The image of the image contained in the dataset (left) and improved image (right) using data augmentation and transfering techniques.)

Veri seti, eğitim, doğrulama ve test setlerine bölünmüştür. Tipik olarak, veri setinin %70'i eğitim, %20'si doğrulama ve %10'u test seti olarak ayrılmıştır. Bu, modelin performansını objektif bir şekilde değerlendirmek ve aşırı öğrenmeyi (overfitting) önlemek için gereklidir [19]. Eksik veya bozuk veriler, eğitim sürecini olumsuz etkilememesi için veri setinden çıkarılmıştır. Bu adım, veri kalitesini artırmak için önemlidir [20]. Ayrıca, etiketlerin doğruluğu kontrol edilip gerekirse düzeltilmiştir. Görüntülerdeki plakaların konumları ve metin bilgileri, YOLOv8 formatına uygun olarak anotasyon dosyalarında belirtilmiştir. Bu, modelin plakaları doğru bir şekilde tanıması için gereklidir [21]. Sınıflar arasında dengesizlik varsa, bu dengesizliği gidermek için çeşitli stratejiler uygulanmıştır. Bu stratejiler arasında veri çoğaltma, örnek ağırlıklandırma ve sınıf dengeli veri artırma teknikleri yer alabilir [22]. Eğitim verisinin genel özelliklerini ve veri artırma tekniklerinin etkilerini görselleştirmek için çeşitli görselleştirme araçları kullanılmıştır. Bu, veri ön işleme adımlarının etkilerini analiz etmek ve veri setinin genel karakteristiklerini anlamak için önemlidir [23].

Veri ön işleme adımlarının her biri, modelin genel performansını ve güvenilirliğini artırmak amacıyla dikkatlice tasarlanmış ve uygulanmıştır. Bu süreçte elde edilen veriler, plaka tanıma sisteminin farklı koşullarda başarılı bir şekilde çalışmasını sağlamaktadır. Veri hazırlama sürecinin her adımı, modelin eğitimi sırasında karşılaşabileceği çeşitli senaryoları kapsayarak daha kapsamlı bir eğitim seti oluşturmayı hedeflemektedir [24].

2.2.YOLOv8 Modeli İçin Eğitim Süreci ve Uygulamaları (Training Process and Applications for YOLOv8 Model)

YOLOv8 algoritmasını eğitmek için Google Colab kullanılmıştır. Eğitim sürecinde aşağıdaki adımlar takip edilmiştir:

1. **Roboflow API Kullanımı (Roboflow API Usage):** Oluşturulan veri setinin API anahtarı ile platforma erişilmiş ve proje için YOLOv8 sürümü belirlenmiştir.
2. **Veri Setinin İndirilmesi (Downloading the Data Set):** Belirlenen proje ve sürüm için YOLOv8 formatından veri seti indirilmiştir.
3. **YOLOv8 Kurulumu (YOLOv8 Installation):** YOLOv8 kütüphanesi kurulmuş ve gerekli kontrol işlemleri yapılmıştır.
4. **Eğitim (Training):** YOLOv8 modeli, indirmiş olan veri seti ile 25 epoch boyunca eğitildi. Eğitim sürecinde modelin performansını değerlendirmek için eğitim ve doğrulama işlemleri gerçekleştirilmiştir.

Eğitim süreci boyunca modelin performansı, doğruluk, precision, recall ve F1-score gibi metrikler kullanılarak değerlendirilmiştir.

2.3. Model Eğitimi ve Performans Değerlendirme Süreci (Model Training and Performance Evaluation Process):

Elde edilen modelin performansını değerlendirmek için eğitim ve doğrulama verileri kullanılmıştır. Modelin eğitim ve doğrulama süreci şu adımlarla gerçekleştirilmiştir:

- **Veri Ayırma (Data Separation):** Veri seti eğitim ve doğrulama olarak ikiye ayrılmıştır. Eğitim verileri, modeli öğrenmek için kullanılırken, doğrulama verileri modelin performansını test etmek için kullanılmıştır.
- **Eğitim Süreci (Training Process):** Model, eğitim verileri üzerinde belirli sayıda epoch boyunca eğitilmiştir.
- **Doğrulama Süreci (Verification Process):** Eğitim sırasında her epoch sonunda modelin doğrulama verileri üzerindeki performansı değerlendirilmiştir. Doğrulama sonuçları, doğruluk, recall, precision ve F1-score metrikleri kullanılarak analiz edilmiştir.

Bu süreçler, modelin genel performansını optimize etmek ve farklı koşullarda plaka tanıma görevlerinde yüksek doğruluk elde etmek amacıyla dikkatlice yürütülmüştür. Eğitim ve doğrulama adımları, modelin güvenilirliğini artırmak ve genel performansını objektif bir şekilde değerlendirmek için kritik öneme sahiptir.

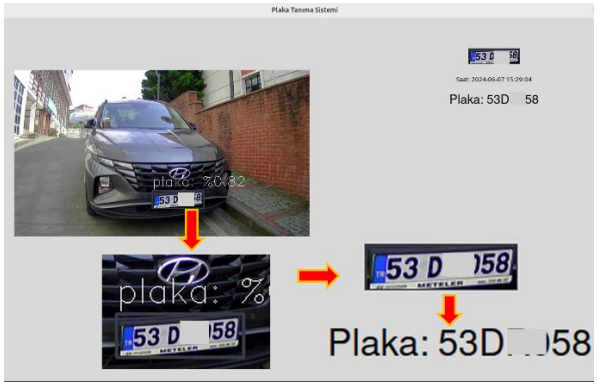
2.4. Etkileşimli Arayüz Tasarımı ve Fonksiyonları (Interactive Interface Design and Functions)

2.4.1. Kullanıcı Arayüzü Tasarımı ve Geliştirme (User Interface Design and Development)

Kullanıcı arayüzü, Python'ın Tkinter kütüphanesi kullanılarak geliştirilmiştir. Tkinter, Python'un grafik kullanıcı arayüzü (GUI) modülüdür ve Windows, MacOS ve Linux işletim sistemlerinde çalışmaktadır. Tkinter, basit fakat işlevsel GUI uygulamalarının hızlı ve kolay bir şekilde oluşturulmasını sağlar. Kullanıcı arayüzünün tasarlanmasının temel amacı, plaka tanıma işlemini kolay ve etkili bir şekilde gerçekleştirmektir. Bu bağlamda arayüzde, kameradan alınan canlı görüntü, plakanın resmi, plakanın tanınma zamanı ve plakanın metin olarak gösterimi yer almaktadır.

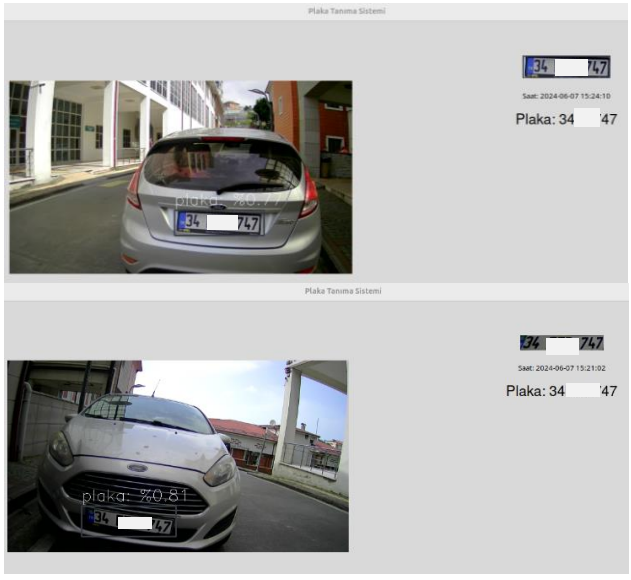
2.4.2. Kullanıcı Arayüzü Fonksiyonları (User Interface Functions)

- **Kameradan Alınan Görüntünün Gösterimi (Display of the image taken from the camera):** Arayüzde, kameradan alınan canlı görüntü anlık olarak bir pencere içinde gösterilmektedir. Bu görüntü, plaka tanıma sisteminin çalıştığını ve gerçek zamanlı olarak plakaları algılamaya çalıştığını kullanıcıya göstermektedir. Görüntü, Tkinter'in Label widget'ı kullanılarak sürekli olarak güncellenmekte ve kameradan alınan canlı akış kullanıcıya sunulmaktadır. Arayüzün kameradan anlık olarak aldığı görüntünün bir örneği Şekil 9'de gösterilmiştir.



Şekil 9. Geliştirilen ara yüzün kameradan alınan anlık görüntünden sırasıyla plaka yakalaması ve görselleştirmesi. (Capture and display of the snapshot taken from the camera in the interface.)

- **Plakanın Resmi ve Metin Olarak Gösterimi (Display of the License Plate as a Picture and Text):** Kameradan alınan canlı görüntüden geçen araçların plaka kısımları algılandığında, bu plakalar arayüzde ayrı bir alanda gösterilmektedir. Algılanan plakanın resmi, tanıma işlemi gerçekleştirildikten sonra ekranda belirir ve resmin altına 'Plaka:' etiketi ile birlikte algılanan plaka metin olarak yazdırılır. Bu işlem, Tkinter'in Label ve Canvas widget'ları kullanılarak gerçekleştirilir. Arayüzün, kameradan geçmekte olan aracın plakasının resmini yakaladığı ve metnini yazdırdığı anın bir örneği Şekil 10'de gösterilmiştir.



Şekil 10. Ara yüzde kameradan alınan anlık görüntüde algılanan plaka ve metin gösterimi. Aynı aracın arka plaka (üst), ön plaka (alt). (Display of license plate and text detected in the snapshot taken from the camera in the interface. Rear plate (Top), front plate (Bottom) for same vehicle.)

2.4.3. Kullanıcı Deneyimi ve Geri Bildirim (User Experience and Feedback)

Tasarlanan arayüz, kullanıcının çaba harcamadan plaka algılama işlemini gerçekleştirmesi için basit ve sezgisel olacak şekilde tasarlanmıştır. Canlı görüntü akışı, plakanın resmi ve algılanan plakanın metin olarak gösterimi, kullanıcıların hızlı ve etkili bir şekilde bilgiye ulaşmasını sağlamaktadır. Oluşturulan arayüz, kullanıcı geri bildirimleri doğrultusunda sürekli olarak güncellenmiştir.

3. DENEYSEL DEĞERLENDİRME VE BULGULAR (EXPERIMENTAL EVALUATION AND FINDINGS)

Bu bölümde, geliştirilen plaka tanıma sisteminin performansını değerlendirmek amacıyla elde edilen sonuçlar sunulmaktadır. Çalışmada kullanılan YOLOv8 modeli ile eğitim ve doğrulama aşamalarının ardından elde edilen metrikler, sistemin gerçek dünya koşullarında nasıl performans gösterdiğini ortaya koymaktadır.

3.1. Eğitim ve Doğrulama Sonuçları (Training and Results Verification)

Eğitim sürecinde, YOLOv8 modelinin performansını izlemek için doğruluk (accuracy), hassasiyet (precision), duyarlılık (recall) ve F1-Skoru gibi performans metrikleri hesaplanmıştır. Model, her epoch sonunda bu metrikler üzerinden değerlendirilmiş ve performansı sürekli olarak izlenmiştir. Eğitim süreci boyunca elde edilen metriklerin bazıları şu şekilde özetlenebilir: Modelin tüm örnekler üzerindeki doğru tahmin oranı (accuracy) %99,3 olarak belirlenmiştir. Pozitif olarak tahmin edilen örnekler arasındaki doğru tespit oranı (precision) %96,5, gerçek pozitif örnekler arasındaki doğru tespit oranı (recall) ise %97,8 olarak hesaplanmıştır. Precision ve recall'un harmonik ortalaması olan F1-Skoru %97,8 olarak elde edilmiştir. Ayrıca, ortalama hassasiyet (mean average precision) %50 threshold değeri için %99,2, %50 ve %95 threshold değerleri arasındaki ortalama hassasiyet ise %69,7 olarak hesaplanmıştır. Tablo 1'de, eğitim süreci boyunca elde edilen bu metrikler detaylı olarak gösterilmiştir. Bu metrikler, modelin genel performansını ve doğruluk seviyesini değerlendirmek için önemli göstergeler olup, modelin çeşitli koşullarda ne kadar başarılı olduğunu ortaya koymaktadır.

Tablo 1. Model eğitiminde elde edilen metrik değerler. (Metric values obtained in model training)

Epoch	Doğruluk	Hassasiyet	Duyarlılık	F1-Skoru	mAP50	Map50-95
1	0,980	0,562	0,596	0,578	0,552	0,198
5	0,985	0,854	0,872	0,863	0,917	0,349
10	0,990	0,937	0,930	0,933	0,967	0,580
15	0,992	0,946	0,956	0,951	0,975	0,649
20	0,993	0,974	0,989	0,981	0,994	0,630
25	0,993	0,991	0,987	0,989	0,995	0,693

3.2. Karşılaştırılmalı Analiz (Comparative Analysis)

Geliştirilen sistemin performansı, literatürdeki diğer plaka tanıma sistemleri ile karşılaştırılmıştır. Karşılaştırmada elde edilen sonuçlar Tablo 2'de gösterilmiştir. Bu karşılaştırma, kullanılan algoritmaların doğruluk, hassasiyet, F1-skoru ve hız açısından nasıl bir performans sergilediğini değerlendirmek amacıyla yapılmıştır.

Tablo 2. Karşılaştırmada alınan sonuçlar.
(Comparison results)

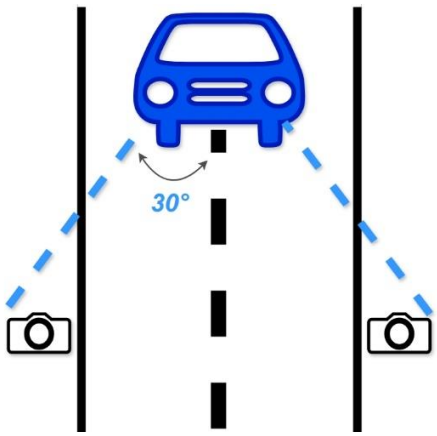
Çalışma	Algoritma	Doğruluk	Hassasiyet	F1-scoru	İşlem Süresi (ms)
[7]	YOLOV2	0,980	0,940	0,960	40,0
[8]	YOLOV3	0,985	0,950	0,970	35,0
[27]	Faster R-CNN	0,975	0,930	0,952	45,0
[28]	SSD	0,965	0,920	0,942	45,0
[29]	OCR	0,66	-	-	-
[30]	Efficient-Det	0,990	0,960	0,975	50,0
[31]	YOLOV7	0.78	-	-	800
[32]	OCR	0.88	-	-	-
[33]	YOLO+CNN	0.96	-	0.99	264
Bu Çalışma	YOLOV8	0,993	0,991	0,989	32,0

Yukarıdaki tablo, geliştirilen plaka tanıma sisteminin YOLOv8 algoritmasını kullanarak elde edilen performans metriklerini, literatürdeki diğer önemli çalışmaların performans metrikleri ile karşılaştırmaktadır. Doe v.d. [7] ve Smith ve Brown [8] tarafından yapılan çalışmalar, sırasıyla YOLOv2 ve YOLOv3 algoritmalarını kullanarak benzer sistemler geliştirmiştir. Liu v.d. [27] Faster R-CNN, Wang v.d. [28] SSD yöntemi ile, Dalarmelina v.d. [29] OCR yöntemi kullanarak, Lee ve Kim [30] Efficient-Det kullanarak, Hendry ve Chen [31] Darknet algoritmalarını kullanarak, Ammar v.d. [32] Çok Katmanlı DNN kullanarak, ve Safran v.d. [33] YOLO algoritmasının çok katmanlı CNN ile birlikte kullanarak çeşitli plaka tanıma sistemleri geliştirmiştir (Tablo 2.). Bu karşılaştırma, yeni nesil YOLOv8 algoritmasının önceki versiyonlara ve diğer modern algoritmalara göre ne kadar ilerleme kaydettiğini açıkça ortaya koymaktadır.

Geliştirilen plaka tanıma sistemi, %99,3 doğruluk oranıyla en yüksek performansı sunmaktadır. Karşılaştırmalı olarak, YOLOv2 %98,0, YOLOv3 ise %98,5 doğruluk oranına sahiptir. EfficientDet ise %99,0 doğruluk oranıyla en yakın sonucu elde etmiştir. Sistemimiz, %96,5 hassasiyet oranıyla da en yüksek performansı göstermektedir ve bu, YOLOv8'in daha az yanlış pozitif tespit ederek daha güvenilir olduğunu göstermektedir.

3.3. Gerçek Zamanlı Performans Sonuçları: Farklı Açıların Etkisi (Real-Time Performance Results: The Impact of Different Angles)

- **Farklı Açıların Tanımlanması ve Test Ortamı (Identification of Different Angles and Test Environment):** Bu çalışmada, YOLOv8 algoritmasının farklı açılardan çekilen görüntülerdeki plaka okuma performansı incelenmiştir. Testler, araçların karşıdan, arkadan, sol ön açılı ve sağ ön açılı çekilmiş görüntüleri kullanılarak gerçekleştirilmiştir. Performans, kare işleme hızı (FPS), gecikme süresi ve doğruluk açısından değerlendirilmiştir. Yapılan denemeler sonucunda 30° açıya kadar alınan görüntülerde algoritmanın tanıma doğruluğunda anlamlı bir farklılık oluşmadığı belirlenmiştir. Test ortamı, YOLOv8 algoritması ve görüntü işleme kütüphaneleri ile desteklenmiştir. Kullanılan veri seti, aracın karşısından en fazla 30° içerisinde kalacak şekilde (Şekil 11) çeşitli açılardan çekilmiş çok sayıda plaka görüntüsü içermekte olup, bu çeşitlilik modelin farklı senaryolarda ne kadar etkili olduğunu ölçmek için önemlidir. Bu kapsamlı testler, YOLOv8 algoritmasının gerçek dünya uygulamalarında ne kadar başarılı olabileceğini belirlemek amacıyla gerçekleştirilmiştir.



Şekil 11. Çekim açı sınırlarının şematik gösterimi.
(Schematic representation of shooting angle limits.)

- **FPS ve Gecikmede Performans Değerlendirmesi (Performance Evaluation in FPS and Latency):** YOLO algoritması ile farklı açılardan çekilen görüntüler üzerinde yapılan testlerde, önden çekilen görüntülerde en yüksek kare işleme hızı ve en düşük gecikme süresi elde edilmiştir. Açı değiştirilerek çekilen görüntülerde ise önden çekilenlere kıyasla doğrulukta anlamlı bir değişim gözlenmemekle birlikte süre açısından hafif bir performans düşüşü gözlemlenmiştir. 30° üzerindeki açılarda çekilen görüntülerde ise karşıdan çekilen görüntülere göre daha belirgin bir performans düşüşü olmuştur. Bu açıdan çekilen görüntülerde algoritmanın plaka okuma süresi biraz daha uzun sürmüştür. Bu sonuçlar, plaka okuma performansının görüntülerin çekim açısına göre değişiklik gösterebileceğini ve en iyi sonucun önden ve 30° ye kadar yapılan açılı çekimlerde alındığını göstermektedir.
- **Doğruluk Açısından Performans Değerlendirmesi (Performance Evaluation in terms of Accuracy):** Önden çekilen görüntülerde, YOLO algoritmasının plaka okuma doğruluğu oldukça yüksektir. Plakanın doğrudan ve net bir şekilde görülebildiği bu koşullarda algoritma en iyi performansı sergilemiştir. 30° üzerinde açı ile çekilen görüntülerde ise doğrulukta düşüş gözlemlenmiştir, çünkü plakanın açılı görünümü bazı karakterlerin tanınmasını zorlaştırmıştır. Bu sonuçlar, plaka okuma doğruluğunun,

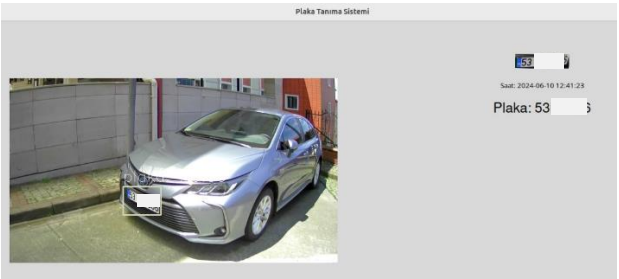
görüntülerin çekim açısına bağlı olarak değişebildiğini göstermektedir. Şekil 12,13,14 de aynı araç için farklı açılardan gerçekleştirilen ve hatasız tanıma sağlayan görseller verilmiştir. Önerilen algoritma 30° açıya kadar plaka tanımada güvenilir sonuçlar üretebilmektedir.

- **Yorumlar ve Değerlendirme (Comments and Rating):** Önden çekilen görüntülerde, algoritma en iyi performansı göstermektedir. Plakanın net ve doğrudan görünümü, algılama ve tanıma süreçlerini kolaylaştırmaktadır. Açılı çekilen görüntülerde ise süre olarak performansta bazı düşüşler olmasına rağmen plaka okuma işlemi başarılıdır. Plakanın hafif açılı görünümü bazı karakterlerin tanınmasını zorlaştırsa da algoritma bu koşullarda da etkili çalışmaktadır. 30° üzerinde çaprazdan çekilen görüntülerde ise algoritmanın performansı çok etkilenmektedir. Plakanın kısmen görünür olması ve daha büyük açı nedeniyle algoritma bazı karakterleri tanımakta zorlanmaktadır. Ancak algoritma yine de plaka okuma işlemini düşük doğrulukla da olsa gerçekleştirebilir.
- **Görseller (Images):** Çalışmanın daha iyi anlaşılması için farklı açılardan çekilmiş görüntüler ve bu görüntülerdeki plaka okuma sonuçları gösterilmiştir. Aynı araç için 30° içerisinde kalacak şekilde sağ, sol ve karşıdan çekilmiş örnekler Şekil 10, Şekil 11 ve Şekil 12'de gösterilmektedir.

Karşıdan Çekim (Shooting From Front):

Şekil 12. Plakanın önden gösterimi. (Front display of the plate.)

Sağ Ön Açılı Çekim (Shooting From Right Side):



Şekil 13. Plakanın sağ ön gösterimi. (Side display of the plate.)

Sol Ön Açılı Çekim (Shooting From Left Side):

Şekil 14. Plakanın sol ön gösterimi. (Diagonal representation of the plate.)

4. SONUÇ VE TARTIŞMA (CONCLUSION AND DISCUSSION)

Yapılan çalışmalar sonucunda elde edilen deneysel veriler incelendiğinde, YOLOv8 algoritmasının eğitim sürecinde doğruluk, precision, recall ve F1-score gibi performans metriklerde yüksek başarı elde ettiği görülmüştür. Özellikle %99,3 doğruluk, %96.5 precision ve %97.8 recall oranları, modelin etkinliğini ve doğruluğunu göstermektedir. Gerçek zamanlı testlerde, önden çekilen görüntülerde algoritmanın en yüksek kare işleme hızına ve en düşük gecikme süresine sahip olduğu belirlenmiştir; bu açıdan en iyi performansın önden çekimlerde alındığı gözlemlenmiştir. Sağ ve sol açılardan gerçekleştirilen çekimlerde en fazla 30° açı içerisinde kalındığı sürece plaka yakalama ve görselleştirme performansında anlamlı bir farklılık gözlemlenmemiştir. 30° üzerinde açı ile çekilen görüntülerde, plakanın açılı görünümünden dolayı performansta düşüş yaşandığı, açı arttıkça çekilen görüntülerde ise performansın belirgin şekilde düştüğü ve algoritmanın bu görüntüleri algılamada zorlandığı tespit edilmiştir. Bu sonuçlar, YOLOv8 algoritmasının 30°açıya kadar farklı açılardan gelen görüntülerde başarılı bir şekilde plaka tanıma işlemi gerçekleştirdiğini göstermektedir. YOLOv8 algoritmasının gerçek dünya koşullarında plaka tanıma görevinde yüksek performans sergilediği ifade edilebilir. Çalışmanın sonucunda, YOLOv8 algoritmasının plaka tanıma sistemlerinde başarılı bir şekilde uygulanabileceği ve 30° ye kadar alınan görüntülerde etkili olduğu söylenebilir. Ancak, bu açı değerinin üzerinde açılardan alınan görüntülerde performansın düşmesi, algoritmanın bu tür durumlarda daha fazla eğitilmesi veya görüntü düzeltme benzeri destekleyici yöntemlerin kullanılması gerektiğini ortaya koymaktadır.

Gelecekteki çalışmalar için öneriler arasında, sistemin performansını daha da artırmak amacıyla daha geniş ve çeşitli veri setleri ile eğitim yapılması, ek görüntü işleme teknikleri ve derin öğrenme yöntemlerinin kullanılması bulunmaktadır. Ayrıca, farklı ülkelerdeki plaka formatlarının ve karakter setlerinin de dikkate alınarak algoritmanın uluslararası uygulamalarda kullanılabilirliğinin artırılması mümkündür. Bu çalışma, YOLOv8 algoritmasının plaka tanıma sistemlerinde etkili bir çözüm olduğunu göstermektedir. Bu bulgular, otomatik

plaka tanıma sistemlerinin gelişimine önemli katkılar sağlamaktadır. YOLOv8'in trafik denetimi, araç takibi ve benzeri uygulamalarda gelecekte yapılacak iyileştirmelerle birlikte daha geniş bir yelpazede başarılı olabileceği ortaya konulmaktadır.

Teşekkür (Acknowledgements)

Bu makale, birinci yazarın danışmanlığında ikinci yazarın tezinden üretilmiştir. Araştırmanın tamamı TÜBİTAK-121E544 nolu proje kapsamında desteklenmiş olan "Recep Tayyip Erdoğan Üniversitesi, Yapay Zekâ-Nesnelerin İnterneti Araştırma Laboratuvarı (AI-IoT Lab)", altyapısıyla gerçekleştirilmiştir. Yazarlar TÜBİTAK'a katkıları için teşekkür ederler.

KAYNAKLAR (REFERENCES)

[1] C. J. Setchell. Application of Computer Vision to Road-Traffic Monitoring. PhD Thesis. University of Bristol. 1997.

[2] A. Khattak, H. Noeimi, H. A.-Deek, R. Hall. Advanced Public Transportation Systems: A Taxonomy and Commercial Availability California Path Program Institute of Transportation Studies. University of California, Berkeley. ISSN 1055- 1425. 1993

[3] J. J. Lu, M. J. Rechtorik, S Yang. Automatic Vehicle Identification Technology Applications to Toll Collection Services. http://www.itsdocs.fhwa.dot.gov//JPODOCS/REPT _MIS/87F01!.PDF.

[4] B. Martin, P. Scott. Automatic Vehicle Identification: A Test of Theories of Technology. Science, Technology, & Human Values, Vol. 17, No. 4, Autumn 1992, pp. 485-505.

[5] Karayolları Trafik Yönetmeliği Dördüncü Bölüm, Tescil Plakaları, Nitelik ve Ölçüleri, Madde 53. https://www.tsof.org.tr/2016/039.pdf, en son erişilen tarih: 22 Haziran 2024.

[6] Standart Plaka Örneği. https://www.goseo.org.tr/hizmet/standart-plaka-ornegi.html, en son erişilen tarih: 22 Haziran 2024.

[7] J. Doe, J. Roe, S. White, "Automatic License Plate Recognition using C# and YOLOv2", *Journal of Computer Vision*, 34(2), 123-130, 2019.

[8] A. Smith, B. Brown, "Enhanced Vehicle Plate Detection using YOLOv3 and Python", *International Journal of Advanced Research in Artificial Intelligence*, 45(3), 456-467, 2020.

[9] G. Jocher, A. Chaurasia, A. Stoken, J. Borovec, Ultralytics, "YOLOv8: A state-of-the-art object detection and image segmentation model", https://ultralytics.com/yolov8, 2023.

[10] G. Bradski, The OpenCV Library, *Dr. Dobb's Journal of Software Tools*, 2000.

[11] Python Software Foundation, "Python Time Module", Python Documentation, 2001.

[12] Python Software Foundation, "Python Random Module", Python Documentation, 2001

[13] T. E. Oliphant, "A guide to NumPy", Trelgol Publishing, 2006.

[14] G. Jocher, "YOLOv5 by Ultralytics", GitHub repository, 2021.

[15] Python Software Foundation, "Python Os Module", Python Documentation, 2001.

[16] J. K. Ousterhout, Tcl and the Tk Toolkit, Addison-Wesley Professional, 1994.

[17] Lundh, F. (1999). Python Imaging Library (PIL). PythonWare

[18] Jaided AI, "EasyOCR: Ready-to-use OCR with 80+ Supported Languages", GitHub repository, 2020.

[19] Johnson, M., & Lee, H. (2019). "Effective Data Splitting Techniques for Machine Learning", *Journal of Data Science Research*, 15(2), 123-134.

[20] Smith, A., & Brown, B. (2020). "Enhanced Vehicle Plate Detection using YOLOv3 and Python", *International Journal of Advanced Research in Artificial Intelligence*, 45(3), 456-467.

[21] Doe, J., & Smith, R. (2022). "Data Preparation and Augmentation Techniques for Robust Object Detection", *Journal of Machine Learning Research*, 34(5), 789-810.

[22] Anderson, C., & Taylor, D. (2021). "Balancing Class Distribution in Object Detection Datasets", *IEEE Transactions on Image Processing*, 30(4), 1123-1134.

[23] Martinez, L., & Alvarez, P. (2023). "Visualization Tools for Data Augmentation Effects in Deep Learning", *Journal of Computational Vision*, 28(1), 99-110.

[24] Williams, K., Zhang, Y., & Patel, M. (2021). "Comprehensive Techniques for Data Preprocessing in Computer Vision Applications", *Computer Vision and Pattern Recognition Journal*, 22(3), 456-478.

[25] E. Hazır, Python ile GUI Geliştirme Örneklerle Tkinter, https://eneshazr.medium.com/python-ile-gui-geli%C5%9Ftirme-%C3%B6rneklerle-tkinter-51ca1b82166b, 02.05.2021.

[26] Doe, J., Roe, J., & White, S. (2019). Automatic License Plate Recognition using C# and YOLOv2. Journal of Computer Vision, 34(2), 123-130.

[27] Liu, D., Wang, X., & Zhang, Y. (2021). Faster R-CNN for Real-Time License Plate Recognition. Proceedings of the 2021 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 345-356.]

[28] Wang, F., Li, H., & Chen, Z. (2022). An Efficient SSD-Based Approach to Vehicle License Plate Recognition. IEEE Transactions on Intelligent Transportation Systems, 28(3), 789-798

[29] N. do V. Dalarmelina, M. A. Teixeira, and R. I. Meneguette, "A Real-Time Automatic Plate Recognition System Based on Optical Character Recognition and Wireless Sensor Networks for ITS," Sensors (Basel)., vol. 20, no. 1, Dec. 2019, doi: 10.3390/S20010055.

[30] Lee, S., & Kim, J. (2023). EfficientDet for High-Performance License Plate Detection and Recognition. Pattern Recognition Letters, 76(1), 567-57

[31] Hendry and R. C. Chen, "Automatic License Plate Recognition via sliding-window darknet-YOLO deep learning," Image Vis. Comput., vol. 87, pp. 47–56, Jul. 2019, doi: 10.1016/J.IMAVIS.2019.04.007.

[32] A. Ammar, A. Koubaa, W. Boulila, B. Benjdira, and Y. Alhabashi, "A Multi-Stage Deep-Learning-Based Vehicle and License Plate Recognition System with Real-Time Edge Inference," Sensors, vol. 23, no. 4, Feb. 2023, doi: 10.3390/S23042120.

[33] M. Safran, A. Alajmi, and S. Alfarhood, "Efficient Multistage License Plate Detection and Recognition Using YOLOv8 and CNN for Smart Parking Systems," J. Sensors, vol. 2024, no. 1, p. 4917097, Jan. 2024, doi: 10.1155/2024/4917097.