Research Article

Araştırma Makalesi

UMÜFED Uluslararası Batı Karadeniz Mühendislik ve Fen Bilimleri Dergisi, 6(2), 33-52, 2024.

UMUFED International Journal of Western Black Sea Engineering and Science

e-ISSN: 2687-2927

YAPAY ZEKA DESTEKLİ ÇUKURLARI TESPİT EDEN VE HARİTA ÜZERİNDE İŞARETLEYEN SİSTEM TASARIMI

Ezgi KARAKAZAN[1,a,*], Eyüp Burak CEYHAN[2,b]
[1]Bartın Üniversitesi MMTF Bilgisayar Mühendisliği Bölümü,
[2]Bartın Üniversitesi MMTF Bilgisayar Mühendisliği Bölümü
[a]ezgi_kazan58@hotmail.com, ORCID: 0009-0001-5144-4168
[b]eyupburak@gmail.com, ORCID: 0000-0002-7088-875X

ÖZET

Bu çalışmada, Kinect sensörü kullanılarak yol yüzeylerinde bulunan çukurların tespiti ve harita üzerinde gösterimi amaçlanmaktadır. Başlangıçta YOLOv5 modeli ile çukur tespiti yapılmış, ancak doğruluk oranının %70 çıkmasından dolayı daha gelişmiş bir model olan YOLOv8'e geçiş yapılmıştır. YOLOv8 kullanılarak %75 başarı elde edilmiş olup daha yüksek doğruluk ve hızlı işlem kapasitesi sayesinde çukurların tespiti iyileştirilmiştir. Geliştirilen sistem sayesinde tespit edilen çukurların GPS koordinatları, Kinect sensöründen elde edilen derinlik verileri ile birleştirilerek harita üzerinde işaretlenmektedir. .NET MAUI ile yapılan uygulama, kullanıcılara bu tespitleri anlık olarak gösterecek şekilde geliştirilmiştir. Tespit edilen her çukur, harita üzerinde belirgin bir şekilde işaretlenmektedir. Böylelikle sürücülerin çukurun bulunduğu konumdan geçerken uyarılması sağlanacaktır. Ayrıca bu uygulama sayesinde hem yol bakım çalışmalarında kullanılabilecek etkili bir çukur tespit ve izleme çözümü sunulmakta hem de otonom sürüş sırasında araçlara yol güzergahında bulunan çukurların konumu bildirilerek olası kazaların ve arızaların önüne geçilebilecektir.

ANAHTAR KELİMELER: Görüntü İşleme, Yolov8, GPS, Kinect, Yol, Çukur, Konumlandırma.

***Sorumlu Yazar (Corresponding Author)**

Atıf (Citation): KARAKAZAN, E., CEYHAN, E.B., "Yapay Zeka Destekli Çukurları Tespit Eden Ve Harita Üzerinde Işaretleyen Sistem Tasarımı", UMÜFED Uluslararası Batı Karadeniz Mühendislik ve Fen Bilimleri Dergisi, 6(2), ss. 33-52, 2024.

Geliş (Received):08.09.2024

Kabul (Accepted): 31.12.2024

Yayın (Published): 31.12.2024

UMÜFED Uluslararası Batı Karadeniz Mühendislik ve Fen Bilimleri Dergisi, 6(2): ss. 33-52,2024.

ARTIFICIAL INTELLIGENCE-SUPPORTED SYSTEM DESIGN THAT DETECTS POTHOLES AND MARKS THEM ON THE MAP

ABSTRACT

This study aims to detect potholes on road surfaces and display them on a map using a Kinect sensor. Initially, pothole detection was conducted with the YOLOv5 model, but due to an accuracy rate of only 70%, a more advanced model, YOLOv8, was adopted. With YOLOv8, a 75% success rate was achieved, improving pothole detection accuracy and processing speed. Through the developed system, the GPS coordinates of detected potholes are combined with depth data obtained from the Kinect sensor and marked on a map. The application, developed with .NET MAUI, allows users to view these detections in real time. Each detected pothole is prominently marked on the map, warning drivers as they approach its location. This application offers an effective pothole detection and monitoring solution that can assist in road maintenance operations and provide autonomous vehicles with information on pothole locations, preventing potential accidents and damage.

KEYWORDS: Image Processing, Yolov8, GPS, Kinect, Road, Pothole, Localization.

1. GİRİŞ

Yollar, insanlar için ulaşımın temelini oluşturur ve farklı yerler arasında bağlantı sağlar. Yolların boyutu, işlevlerine bağlı olarak değişir. Örneğin, otoyollar, yoğun trafik için tasarlanmış çok şeritli büyük yollardır. Ancak, şehir içindeki yollar daha küçük olup bir veya iki şeritten oluşur. Yollar, insanların günlük yaşamında hayati bir öneme sahiptir, bu nedenle işlevsel ve güvenli olmalarını sağlamak için periyodik bakım yapılmalıdır. Bir ülkedeki birçok yol, yolların sürekli olarak değerlendirilmesini zorlaştırır; bu nedenle, çukurların oluşumu tahmin edilemez. Yol yüzeyi bozulması, yolların kusurlarının ana sebebidir. Yol yüzeyi bozulması üç sınıfa ayrılabilir [1]: Bunlardan ilki yüzey deformasyonu (kayma, dalgalanma ve tekerlek izleri), ikincisi çatlaklar (yorgunluk, kenar bozulması ve çatlama) ve üçüncüsü yüzey dağılmasıdır (aşınma ve soyulma). Bu tür bozulmaların ana nedeni, çevresel koşulların ve trafik kaynaklı yol yüzeyi streslerinin bir kombinasyonu ile ilişkilendirilebilir. Çukurlar, dünya çapında bir sorundur ve her yıl hükümetlere ve vatandaşlara milyarlarca dolara mal olmaktadır [2,3]. Yıllık olarak 1,25 milyon kişi trafik kazaları nedeniyle hayatını kaybetmektedir ve bunların %34'ü yol çukurları ile ilişkilidir [4].

Görüntü işleme, bir veya birden çok görüntü verilerinden bilgisayar yazılımları kullanarak bir veya daha çok ana işlemciyle zaman sıralamasına göre çeşitli tekniklerle görüntülerin analiz edilmesi işlemidir [5]. Sensörden veriyi okuması görüntü işleme tekniği ile gerçekleşmektedir. Derin öğrenme, insan beyninin çalışma şekline göre genel hatlarıyla modellenen algoritmalar olan sinir ağlarının katmanları tarafından desteklenir. Büyük miktarlarda veri ile eğitim, sinir ağındaki nöronları konfigüre etmektir. Eğitildikten sonra yeni verileri işleyen derin öğrenme modelidir [6]. Derin öğrenme özellikle sınıflandırma, tanıma ve tespit için kullanılmaktadır. Son zamanlarda derin öğrenme algoritmaları pek çok alanda kullanılmakta ve günlük yaşamımızda pek çok teknolojik yapı içerisinde kullanılmaktadır. Yüksek başarımların elde edilmesi bu konuya eğilimi artırmıştır. Yapılan pek çok sınıflandırma, tanıma çalışmaları mevcuttur [7]. Verilerin eğitilip yeni veriler üzerinden tespit edilmesi derin öğrenme algoritmalarıyla gerçekleşmektedir.

Çalışmada kullanılan YOLOv5 (You Only Look Once) modeli, nesne tespiti yapmakta ve bir derin öğrenme modelidir. Bu model, bir görüntüdeki nesneleri hızlı ve etkili bir şekilde tanımlayabilir ve bunların yerlerini belirleyebilir. YOLOv5 (v6.0/6.1) Ultralytics tarafından geliştirilen güçlü bir nesne algılama algoritmasıdır [8]. Hız ve doğruluk açısından üstün performans sergileyen bu model, gerçek zamanlı uygulamalarda yaygın olarak kullanılmakta ve tercih edilmektedir. Gözetim sistemleri, otonom araçlar ve dronlarla yapılan izleme gibi çeşitli alanlarda kullanımı yaygındır. YOLOv5, n, s, m, l, x gibi 4 farklı model sunmaktadır [9]; bu sayede, kullanıcının ihtiyaçlarına göre daha hızlı veya daha doğru sonuçlar almak mümkün olmaktadır. Modelin PyTorch tabanlı olması, onu mevcut iş akışlarına kolayca entegre edilebilir hale getirmektedir. Eğitim süreci boyunca model, her döngüde verilerden öğrenmekte ve doğruluğunu arttırmaktadır. Eğitilen model, gerçek zamanlı görüntülerde veya kaydedilmiş videolarda nesneleri tespit edebilme yeteneğine sahiptir. Burada veri kümesi hazırlığı için COCO ve Pascal VOC gibi veri kümeleri kullanılmakta, resimler ve etiketler içermektedir. Model konfigürasyonu yaml dosyaları kullanılarak modelin yapısı ve hiperparametreleri tanımlanmaktadır. Eğitim için Python uzantılı eğitim dosyası bulunmaktadır. Eğitim sırasında hiperparametreler, öğrenme oranı (learning rate) ve devir(epoch) sayısı gibi ayarlar yapılmaktadır. Modelin doğruluğu, doğrulama ve test veri kümeleri üzerinde değerlendirilmektedir. Eğitilmiş model, yeni görüntüler üzerinde nesne tespiti yapmak için kullanılmaktadır. Model tespit edilen nesnelerin sınıf etiketlerini, sınırlayıcı kutu (bounding box) koordinatlarını ve çukur olma olasılıklarını yüzde şeklinde döndürmektedir. Tahmin

sonuçları sunulmakta ve görüntü üzerinde tespit edilen nesneler görselleştirilmektedir. Precision ve Recall modelin doğruluğu ve duyarlılığı ölçütleriyle değerlendirilmektedir. mAP, farklı eşik değerleri için modelin genel performansını ölçen bir metriktir. Elde edilen mAP değerleriyle doğruluk karşılaştırılması yapılmaktadır. Karışıklık matrisi (Confusion Matrix) modelin sınıflandırılması için kullanılır. Maksimum olmayan baskılama (Non-Maximum Suppression (NMS)) algoritması, bilgisayarla görme ve nesne tespiti gibi alanlarda sıkça kullanılan bir yöntemdir. NMS, özellikle aynı nesnenin birden fazla kez tespit edildiği durumlarda, en doğru tespiti seçmek ve gereksiz olanları elemek için kullanılır. Bu sayede, tespit edilen nesnelerin daha güvenilir ve doğru sonuçlar vermesi sağlanır [10].

Diğer bir algoritma olan YOLOv8, önceki sürümlere göre daha optimize edilmiş bir mimari kullanır. Bu optimizasyonlar, modelin daha hafif ve daha hızlı çalışmasını sağlar ancak doğruluğundan ödün vermez. Model, görüntüdeki nesneleri daha doğru bir şekilde algılayabilmek için bağlam bilgisine daha fazla önem vermektedir. Modelin özellikle kalabalık veya karmaşık sahnelerde daha iyi performans göstermesini sağlar. Nesne algılama başlığında iyileştirmeler yaparak, algılanan nesnelerin sınıflandırılmasında ve konumlandırılmasında daha yüksek doğruluk elde edilmektedir. YOLOv8, güvenlik kameralarında gerçek zamanlı olarak nesne ve insan algılama için kullanılabilmektedir. Hızlı tepki süresi, bu tür uygulamalar için idealdir. Otonom sürüş sistemlerinde, yol üzerindeki araçlar, yayalar ve diğer nesneler hızlı bir şekilde algılanmaktadır. YOLOv8, bu tür sistemlerin temel bir parçası olabilmektedir. Fabrikalarda ve üretim hatlarında, nesnelerin otomatik olarak algılanması ve sınıflandırılması için YOLOv8 kullanabilmektedir. Üretim süreçlerinin otomasyonunu ve verimliliğini arttırmaktadır. Gerçek zamanlı uygulamalar için son derece hızlıdır. Tek geçişte nesne algılama yapabilmesi, bu algoritmayı birçok senaryoda ideal kılmaktadır. YOLOv8 hem hız hem de doğruluk açısından diğer birçok nesne algılama algoritmasından daha iyi performans gösteren bir nesne algılama algoritmasıdır [11]. Bu algoritma, karmaşık ve yoğun sahnelerde bile doğru nesne algılama sağlamaktadır. Farklı uygulamalara kolayca uyarlanabilmektedir. Modelin mimarisi, çeşitli veri ve görevler üzerinde etkili olmasını sağlamaktadır. Geliştiriciler ve araştırmacılar için kullanım kolaylığı sağlayan bir yapıya sahiptir. YOLOv8, derin öğrenme ve bilgisayarla görme alanındaki en son gelişmeler üzerine inşa edilmiştir ve hız ve doğruluk açısından benzersiz bir performans sunar [12]. Hem hız hem de doğruluk açısından önemli avantajlar sunar ve geniş bir uygulama yelpazesi için uygundur. YOLO'nun gelişiminde önemli bir adım olan bu model, nesne algılama ve ilgili görevlerde yeni bir standart belirlemektedir.

YOLOv8, COCO gibi geniş ve zorlu veri kümelerinde, daha yüksek mAP 0,5:0,95 değerlerine ulaşarak birçok modelin üzerinde bir performans sergilemiştir. Otonom araçlar ve trafik izleme çalışmalarında, IoU eşiği 0,5 olan mAP (mean Average Precision) genellikle 0,7 ile 0,75 aralığında gözlemlenir. Bu doğruluk seviyesi, yaya ve araç takibi gibi nesne algılama tabanlı görevlerde ortaya çıkar ve otonom sürüş bağlamında nesne algılama ve sahne anlama görevlerinin değerlendirilmesinde yaygın olarak kullanılmakta olup, model geliştirme ve doğrulama için kritik öneme sahiptir [13, 14]. COCO gibi karmaşık veri kümelerinde bu ölçüt genellikle 0,4 civarındadır.

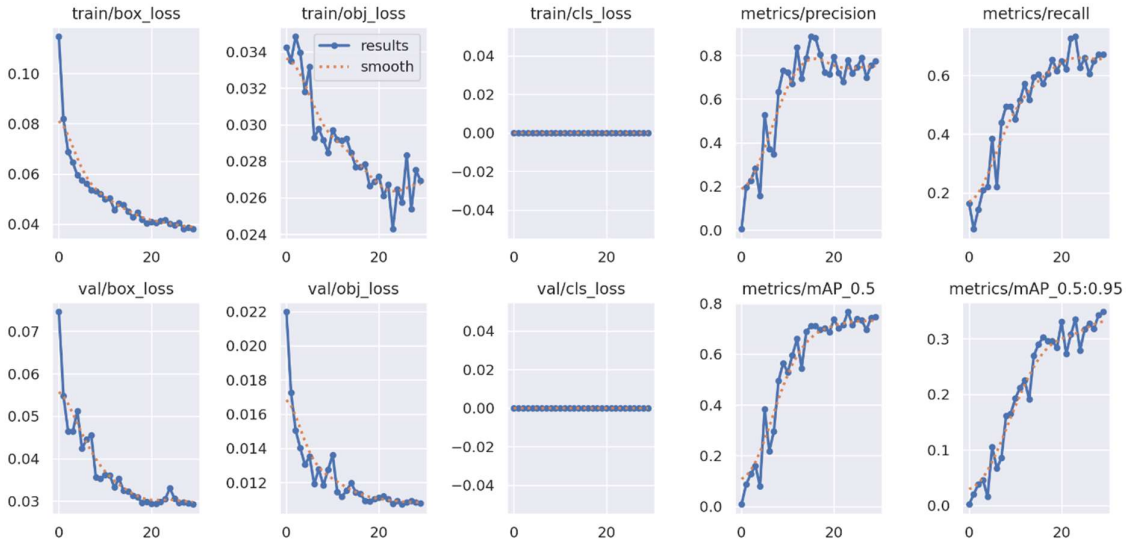
Görüntü işleme için Kinect v2 sensöründen renk ve derinlik verileri elde edilebilir. Derinlik verileri, her pikselin sensörden olan mesafesini göstermekte ve bu mesafeyi kullanarak yüzeydeki düzensizlikler tespit edilebilmektedir. Bir çukur tespiti için, belirli bir alandaki derinlik değerlerindeki ani değişimler aranmaktadır. Kenar algılama teknikleri kullanılarak, yüzeydeki keskin değişiklikler tespit edilebilmektedir. Belirli bir eşik değeri belirlenerek, bu eşik üzerinde olan alanlar çukur olarak sınıflandırılmaktadır. Bu işlem, derinlik verilerindeki büyük farklılıkları tespit ederek yapılabilmektedir. Daha karmaşık bir tespit sistemi oluşturulmak istenirse, YOLO veya başka bir derin öğrenme modeliyle yüzeydeki çukur ve düzensizlikleri tanıyacak bir model eğitebilmektedir. Çukur tespit edilen alanlar renkli olarak işaretlenip 3D bir model üzerinde gösterilmektedir. Bu model, kullanıcının yüzeydeki anormallikleri kolayca görmesini sağlamaktadır. Derinlik sensörü sayesinde yol yüzeylerinden gelen üç boyutlu derinlik bilgileri yüksek doğrulukla elde edilmektedir. Çukur tespiti, yüzeydeki yükseklik farklarının algılanmasını gerektirdiği için Kinect'in bu özelliği, yol üzerindeki bozulmaları belirlemek için idealdir. Kinect kızılötesi sensörleri kullanır ve yaklaşık derinlik ölçüm doğruluğuna sahiptir [15]. YOLO gibi nesne algılama algoritmalarıyla işlenen derinlikli ve renkli görüntüler, yüzeydeki bozukluklar ve çukurların hassas bir şekilde tanımlanmasını sağlamaktadır. YOLOv8 gibi gelişmiş modeller kullanılarak daha hızlı ve isabetli tespitler yapılabilmektedir. Ayrıca, Kinect sensöründen gelen bu veriler GPS verileriyle birleştirilerek, tespit edilen çukurların harita üzerinde gerçek zamanlı olarak gösterilmesi sağlanabilmektedir. Bu tür bir sistem hem yollardaki sorunların tespiti hem de izlenmesi açısından kullanışlıdır. Kinect v2 sensörü USB girişli aparatıyla birleştirilmektedir. SDK Browser v2.0 programı indirerek bilgisayarın Kinect v2'yi görmesi sağlanmaktadır. Kinect Configuration Verifier menüsünden Kinect v2'nin bağlanıp bağlanmadığı görülmektedir.

Arduino ile GPS modülü kullanımı, çalışmalarda konum verilerini gerçek zamanlı olarak elde etmek için yaygın bir yöntemdir. GPS (Global Positioning System) modülleri, dünya üzerindeki konum, hız ve zaman bilgilerini uydu sinyalleri aracılığıyla sağlar. Arduino ile bir GPS modülü entegre edildiğinde, bu veriler kullanılarak çeşitli uygulamalar geliştirilebilir. Örneğin, harita üzerinde izleme, rotalama ve konum tabanlı tetikleme sistemleri [16] gibi sistemlerde kullanılmaktadır.

2. MATERYAL VE YÖNTEM

Çalışma kapsamında Kaggle'da bulunan Pothole Detection [17] veri seti kullanılmıştır. Bu veri setinde çukur bulunan 665 adet resim bulunmaktadır. Veri setindeki resimlerde bulunan çukur verileri sınırlayıcı kutular olarak ayarlanarak eğitim (train) ve test dosyaları oluşturulmuştur. Böylelikle 665 resim bulunan veri seti bir xml dosyası olarak kaydedilmektedir. Dosyalar oluşturulduktan sonra sonuçların görselleştirilmesi için Google Colab'da *yolo_training* dosya dizini Drive'da bulunması için değiştirilmektedir. Git clone ile github'daki YOLOv5 github deposunu klonlamaktadır. *Requirements.txt* dosyası içinde gerekli bulunan numpy, pandas, matplotlib, opencv-python, jupyter, labelImg kütüphaneleri pip install ile indirilmektedir. Python betiği olan *train.py* çalıştırılmakta ve belirli parametreler sağlanmaktadır. Veri için *data.yaml* dosyası eğitim ve doğrulama veri yolları gibi özellikleri içeren bir yapılandırma dosyasını tanımlamaktadır. Ağırlıklar (Weights) kısmı için önceden eğitilmiş ağırlıkları tespit etmektedir. Aynı anda eğitilmesi gereken veri sayısı (batch size) 8 olarak belirlenmiş, eğitim sırasında her adımda 8 görüntü işlenmiştir. 640*640 piksel olması için img 640 olarak boyutlandırılmıştır. Model, eğitim sırasında veri kümesi üzerinde 30 devir (epoch) olacak şekilde ayarlanmış ve her bir veri kümesi örneği 30 kez işlenmiştir. Bunun nedeni modelin eğitim sürecinde yeterli bir öğrenme kapasitesine ulaşmasını sağlarken aynı zamanda aşırı öğrenme riskini en aza indirmek için yapılmaktadır. Aşırı öğrenme, modelin eğitim kümesindeki örüntüler yerine gözlemleri öğrenmesi durumunda ortaya çıkmaktadır. Bu durumda oluşturulan model eğitim aşamasında kullanılan veri kümesini öğrenir, ancak yeni gelen gözlemler için başarılı bir tahmin yapamaz. Genelde aşırı öğrenme modelleri eğitim aşamasını küçük hata oranı ile tamamlarken, test aşamasında büyük bir hata oranı ile tahmin etmektedir [18]. Veri seti boyutu ve modelin karmaşıklığı dikkate alınarak, devir sayısının doğruluk ve işlem süresi açısından optimum denge sağladığı gözlemlenmektedir. Eğitimin

sonunda, algoritmanın performansını gösteren bir *results.png* dosyası oluşturulmuştur. Şekil 1'de YOLOv5 kullanılarak eğitilen modelin performans grafiği sunulmuştur.

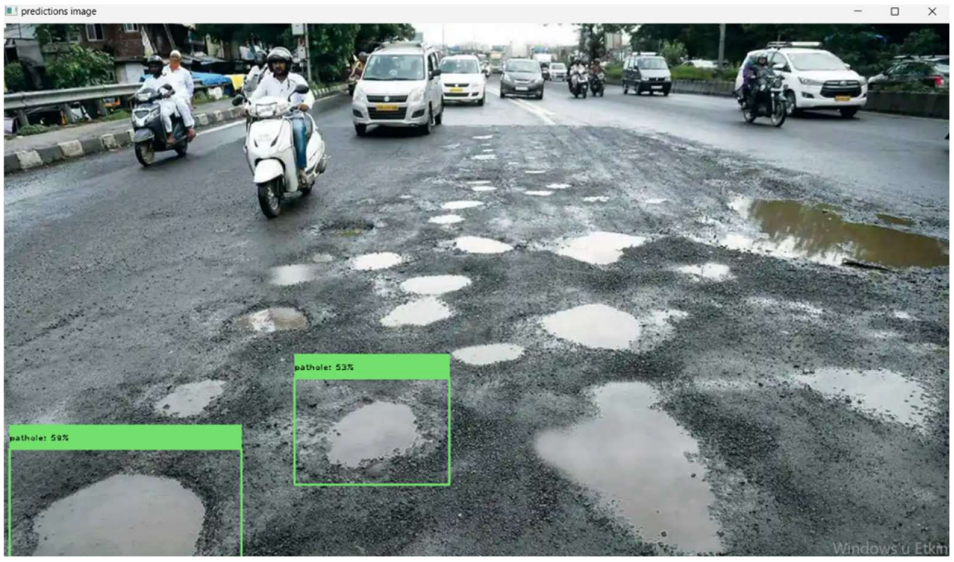


Şekil 1. YOLOv5 kullanılarak eğitilen modelin performans grafiği

Jupyter Notebook'a tekrar dönüldüğünde cv2, numpy, os, yaml, SafeLoader kütüphaneleri içe aktarıldıktan sonra *data.yaml* dosyasını açmak için open metodu kullanılmaktadır. OpenCV'nin derin sinir ağı (DNN) modülü kullanılarak bir YOLO modeli yüklenmekte ve yapılandırılmaktadır. YOLO modeli ONNX formatında yüklenmektedir. cv2.dnn.readNetFromONNX OpenCV'nin kendi DNN arka kısımında (backend) kullanmaktadır. Çalışmada, DNN_BACKEND_OPENCV Modeli CPU üzerinde çalıştırılan DNN_TARGET_CPU ile görüntüyü yüklemekte ve orijinalini bozmadan kopyasını oluşturmaktadır. YOLO modeline uygun bir formatta girdi oluşturmaktadır. YOLO modeli, bu girdi ile nesne tespiti yapmakta ve sonuçlar döndürülmektedir. YOLO modelinden elde edilen tahminler çok boyutlu bir dizi olarak dönmektedir. İlk tahmin seti preds[0] olarak alınmakta ve tespit (detections) değişkenine atanmaktadır. Bu listeler, sırasıyla sınırlayıcı kutuların koordinatlarını, güven puanlarını ve sınıf kimliklerini saklamak için kullanılmaktadır. Giriş görüntüsünün genişlik (image_w) ve yüksekliği (image_h) hesaplanmaktadır. YOLO modeli 640x640 piksel girdi beklemekte, bu yüzden çıktılar bu etkenlerle orijinal görüntü boyutlarına ölçeklenmektedir. Bu tahmin edilen nesne için YOLO modelinin verdiği güven puanıdır. Bu sadece güven puanı 0,4'den büyük olan tahminlerin dikkate alınmasını sağlamaktadır. Bu satırlar, en yüksek sınıf skorun ve bu skoru alan sınıfın kimliğini belirlemektedir. Sadece sınıf

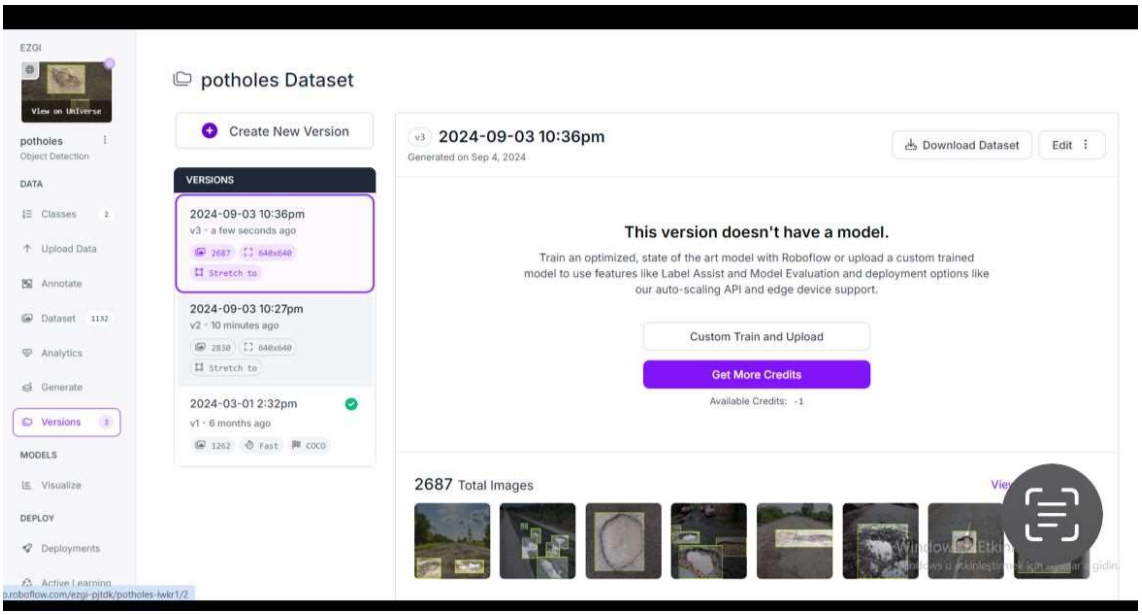
skoru 0,25'den daha büyük olan tahminler değerlendirilmektedir. Bu tahmin edilen nesnenin merkez koordinatlarını (cx, cy), genişliğini (w) ve yüksekliğini (h) çıkarmaktadır. Nesne tespiti sürecinde sınırlayıcı kutuların koordinatlarını ve güven puanlarını belirlemektedir. Tespit edilen nesnelerin genişlik ve yükseklik değerleri görüntü boyutlarına göre ölçeklendirilir. Maksimum olmayan baskılama (NMS) algoritması ile kesişen kutular arasında en güvenilir olanı seçmektedir. Bu işlemde, minimum güven puanı ve kesişme eşik değerleri sırasıyla 0,25 ve 0,45 olarak belirlenmiştir. Kalan sınırlayıcı kutuların koordinatları, güven puanları (yüzde formatında) ve sınıf kimlikleri alınır. Sınıf kimlikleri, etiketler (labels) listesi sayesinde sınıf adlarına dönüştürülmektedir. Önce orijinal görüntüyü, ardından YOLO ile yapılan tahminlerin işlenmiş olduğu görüntüyü ekranda ayrı pencerelerde gösterir. Kullanıcı bir tuşa bastığında, her iki pencere de kapatılmaktadır. Bu, nesne tespiti yapılan görüntüyü görsel olarak incelemek için kullanılmaktadır.

Yolo tahminleri (predictions) tamamlandıktan sonra artık fotoğraf üzerinden nesne tespiti yapılabilmektedir. Başka bir Jupyter sayfasında *yolo_predictions* tanımladıktan sonra *YOLO_Pred* adlı bir sınıfın bir örneği oluşturulmaktadır. Bu örnek, YOLO modelini *best.onnx* dosyasını ve *data.yaml* dosyasını kullanarak nesne tespiti yapacak şekilde oluşturulmaktadır. OpenCV kullanarak çukur resmini ekranda gösterir. YOLO modelini kullanarak bir görüntü üzerinde nesne tespiti yapar ve tespit edilen nesneleri içeren görüntüyü ekranda göstermektedir. Şekil 2'de YOLOv5 performans çıktısı gösterilmektedir.



Şekil 2. YOLOv5 performansı çıktısı

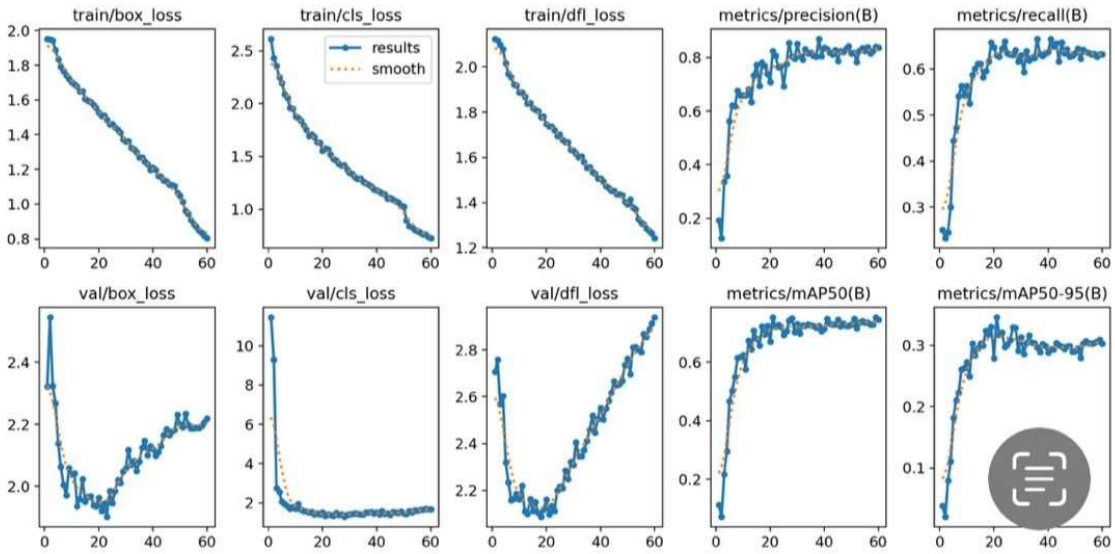
YOLOv8 ile yapılan nesne tespitinde ise, öncellikle bir veri seti oluşturulması ve bu veri setinin eğitim ve test olarak ayarlanması için farklı bir program kullanılmıştır. RoboFlow'dan görüntüleri etiketlemek için RoboFlow'un sunduğu anotasyon aracı kullanılmaktadır. Yüklenen her görüntü için sınırlayıcı kutular çizilmektedir. Şekil 3'de Roboflow veri seti oluşturma gösterilmektedir. Tüm görüntülerini etiketledikten sonra, RoboFlow verileri eğitim ve test olarak ayırma fırsatı verir. Sağ üstteki "Oluşturma" (Generate) butonuna tıklanır. Eğitim, onaylama ve test veri setlerinin yüzdelerinin belirlenmesini sağlar. Standart olarak %70 eğitim, %20 validasyon, %10 test önerilir. Bu oranlar ihtiyaca göre ayarlanabilir. Böylelikle RoboFlow, eğitim ve test dosyaları oluşturulmuş bir kod verir.



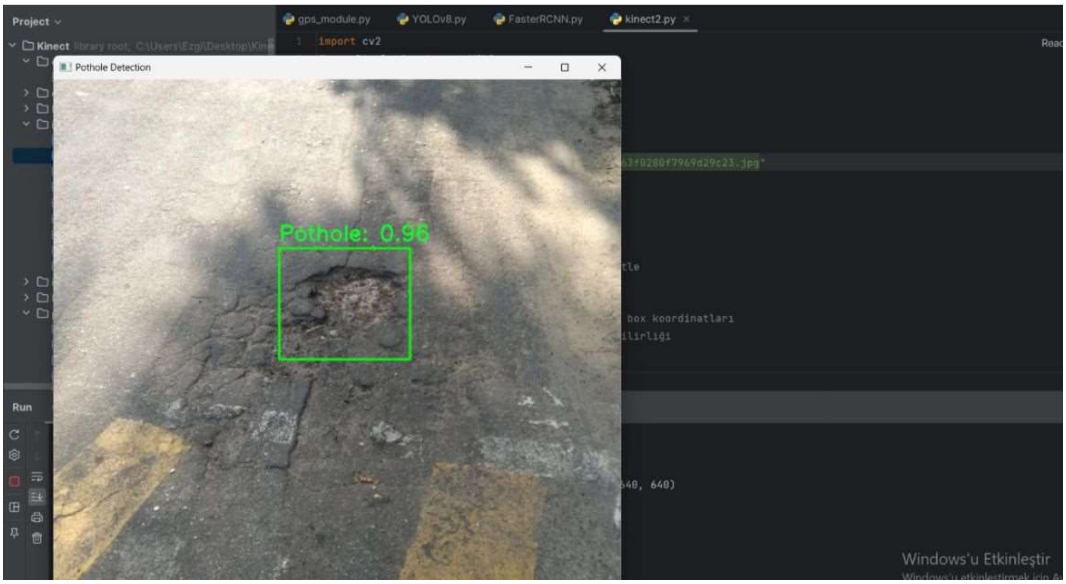
Şekil 3. Roboflow veri seti oluşturma

Bu kod Google Colab'da YOLOv8 ile çukur tespiti yaparken kullanılmaktadır. Google Colab içerisinde *best.pt* dosyası için şu adımlar izlenir: *yolov8training* adlı jpynb dosya uzantılı dosya için öncellikle işletim sistemini (os), glob, Image, display gibi kütüphaneleri tanımlamak gerekmektedir. Pip indirme (install) ile YOLOv8 için Ultralytics indirilmektedir. Ultralytics tanımlandıktan sonra RoboFlow içerisinde oluşan kod burada indirilmektedir. YOLO V8'in "orta" (medium) boyutlu modeli kullanılarak çukur tespiti için bir nesne tespiti modeli eğitilir. Model, */content/potholes/data.yaml* dosyasında tanımlı olan veri seti üzerinde 120 devir boyunca eğitilmekte ve bu sırada görüntüler 640x640 piksel boyutunda işlenmektedir. Eğitim tamamlandıktan sonra, modelin performansı değerlendirilerek çukur tespitinde

kullanılabilmektedir. Daha sonrasında oluşan results.png dosyasına ulaşılabilmektedir. Şekil 4'de YOLOv8 eğitim performans grafiği gösterilmektedir. Şekil 5'de YOLOv8 ile eğitildikten sonraki performans çıktısı gösterilmektedir. Önemli kısım ise *runs* dosyasının içerisinde oluşan *best.pt* dosyasıdır ve bu dosya YOLO modeli olarak çalışmada kullanılmaktadır.



Şekil 4. YOLOv8 eğitim performansı grafiği



Şekil 5. YOLOv8 performans çıktısı

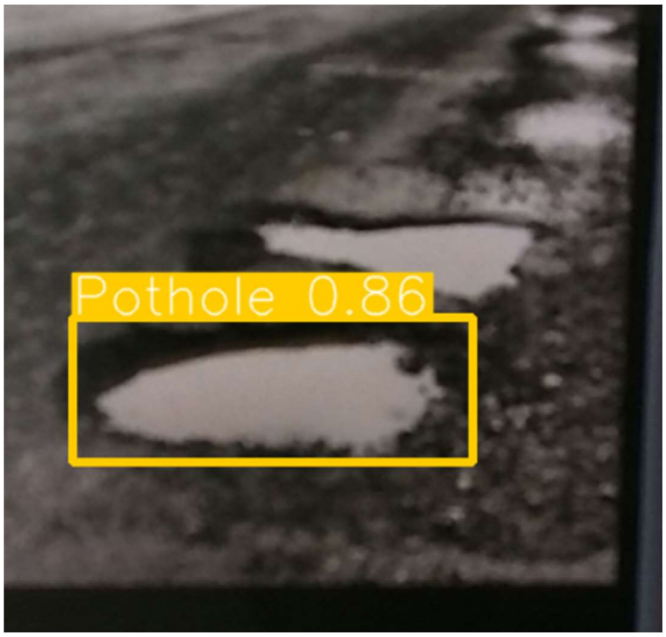
Çalışmada üç farklı algoritma kullanılarak incelenmiştir. Sonuçlara göre hız ve doğruluk unsurları karşılaştırılarak en uygun algoritma tespit edilmiştir.

PyKinectRuntime ile Kinect sensörüyle etkileşim sağlanır ve renk ile derinlik kareleri alınmaktadır. folium kütüphanesi, çukur konumlarını gösteren haritalar oluşturmak için kullanılmaktadır. *read_gps_data* ve *plot_location*, GPS verilerini işlemek için kullanılan özel fonksiyonlardır. Kinect sensörü hem renk hem de derinlik karelerini yakalamak için başlatılmaktadır. Daha sonra PyTorch Hub kullanarak özel bir YOLOv5 modeli yüklenmiştir. Bu modelin amacı, özellikle çukurları tespit etmektedir. *pothole_data* listesi, tespit edilen çukurların GPS koordinatlarını ve görüntü yollarını saklamak için kullanılmaktadır.

Tespit edilen çukurların görüntülerini kaydetmek için *pothole_images* adlı bir dizin oluşturulmaktadır. GPS verisi, ayrı bir kısımda sürekli olarak okunmaktadır. Bu sayede programın ana döngüsü GPS verisi alımı nedeniyle engellenmez. Çalışmada, Kinect sensöründen gelen yeni bir renk ve derinlik karesi olup olmadığı sürekli kontrol edilmektedir. Eğer yeni bir kare varsa renk çerçevesi alınmakta ve YOLOv5 modelinde işlenmektedir. Model, çukurları tespit ettiğinde çukurun konumu, derinlik bilgisi ve güvenilirlik değeri hesaplanmaktadır. Çukurun görüntüsü kaydedilmekte ve GPS koordinatları ile *pothole_data* listesine eklenmektedir. Program ayrıca bu kareleri ekranda görüntülemekte ve kullanıcı 'q' tuşuna bastığında döngüyü durdurmaktadır. Program sona erdiğinde Kinect serbest bırakılmakta ve OpenCV kütüphanesi kapatılmaktadır. Eğer *pothole_data* listesinde veri varsa, bu veri ile bir harita oluşturulmaktadır. folium.Map ile tespit edilen ilk çukurun merkez olarak alındığı bir harita oluşturulmaktadır. Her çukurun bulunduğu konuma sembol olarak bir işaretleyici eklenmekte ve üzerine tıklanınca çukurun görüntüsü gösterilmektedir. Harita *pothole_map.html* dosyasına kaydedilmektedir. Bilgisayar ile Arduino arasında seri bağlantı kurulmaktadır. GPS verileri kullanılarak harita üzerinde bir işaretleyici (marker) yerleştirilmektedir. ser.readline() ile Arduino'dan gelen bir satırlık veri okunur ve bu veri önce UTF-8 formatında kod çözücü (decode) fonksiyonuyla temizlenmektedir. Eğer satır "Enlem:" ile başlıyorsa, bu satırda enlem (latitude) değeri bulunur ve lat değişkenine atanmaktadır. Eğer satır "Boylam:" ile başlıyorsa, bu satırda boylam (longtitude) değeri bulunur ve lng değişkenine atanmaktadır. Boylam değeri bulunduğunda, lat ve lng ikilisi geri döndürülmektedir. Verilen enlem (lat) ve boylam (lng) değerlerine göre bir harita oluşturulmaktadır. folium.Map fonksiyonu, haritayı verilen koordinatlarda merkezlemekte ve bir işaretleyici eklemektedir. Harita, *gps_location.html* adlı bir HTML dosyası olarak kaydedilmektedir. Bu dosya, tarayıcıda

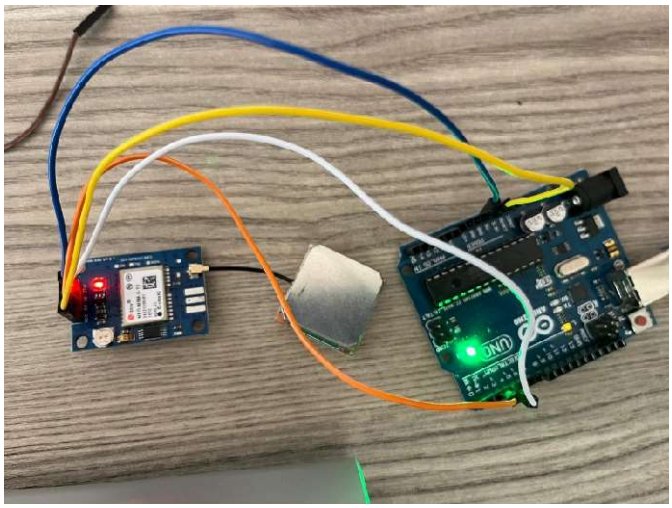
açıldığında GPS konumunu göstermektedir. Program çalıştırıldığında sürekli okunan bir GPS verisi haritaya yerleştiren bir döngü başlatılmaktadır. GPS verisi her okunduğunda enlem ve boylam konsola yazdırılmakta ve ardından bu konumun haritası güncellenmektedir.

Tespit edilen çukurun üzerine dikdörtgen bir şekil çizilmekte ve derinlik bilgisi gösterilmektedir. Çukurun bulunduğu bölgenin görüntüsü kaydedilmektedir. Eğer GPS verisi mevcutsa, bu koordinatlar *pothole_data* listesine kaydedilmektedir. Ana döngü sona erdiğinde, tespit edilen çukur verileri kullanılarak bir harita oluşturulmaktadır. Harita, tespit edilen ilk çukurun koordinatları merkez alınarak başlatılmaktadır. Tüm çukur konumları, haritada işaretleyiciler olarak gösterilmekte ve bu işaretleyicilere tıklanarak çukur görüntüleri görülebilmektedir. Harita, "*pothole_map.html*" dosyası olarak kaydedilmekte ve eğer bu işlem başarılı olursa, "Harita başarıyla kaydedildi." mesajı yazdırılmaktadır. Eğer herhangi bir sorun oluşursa, hata mesajı görüntülenmektedir. Kinect sensöründen gelen görüntüler kullanılarak çukur tespiti yapılmakta ve GPS verisiyle birlikte bu tespitleri harita üzerinde göstermektedir. Şekil 6'da program çalıştırıldığında Kinect sensöründen aldığı veri üzerinden yapılan çukur tespiti gösterilmektedir. Sonuç olarak, tespit edilen çukurların haritada görselleştirildiği bir HTML dosyası oluşturulmaktadır. YOLOv8 ile oluşturulan kod ile program hem hızlı hem de doğruluk açısından YOLOv5'den daha yüksektir. Bu yüzden geliştirilen sistemde tercih sebebi olmuştur.



Şekil 6. Kinect sensöründen alınan görüntüden çukur tespiti

Arduino Uno, u-blox 8M GPS modülü ve jumper kabloları kullanılarak oluşturulan devrede, GPS modülü ile Arduino arasındaki seri iletişim, dijital pinler üzerinden sağlanmaktadır. GPS modülünün TX pini Arduino'nun pin 4'üne, RX pini ise pin 3'e bağlanmaktadır. Şekil 7'de oluşturulan GPS modülü devresi gösterilmektedir. SoftwareSerial kütüphanesi kullanılarak yazılımsal bir seri port oluşturulmasıyla gerçekleştirilmektedir.

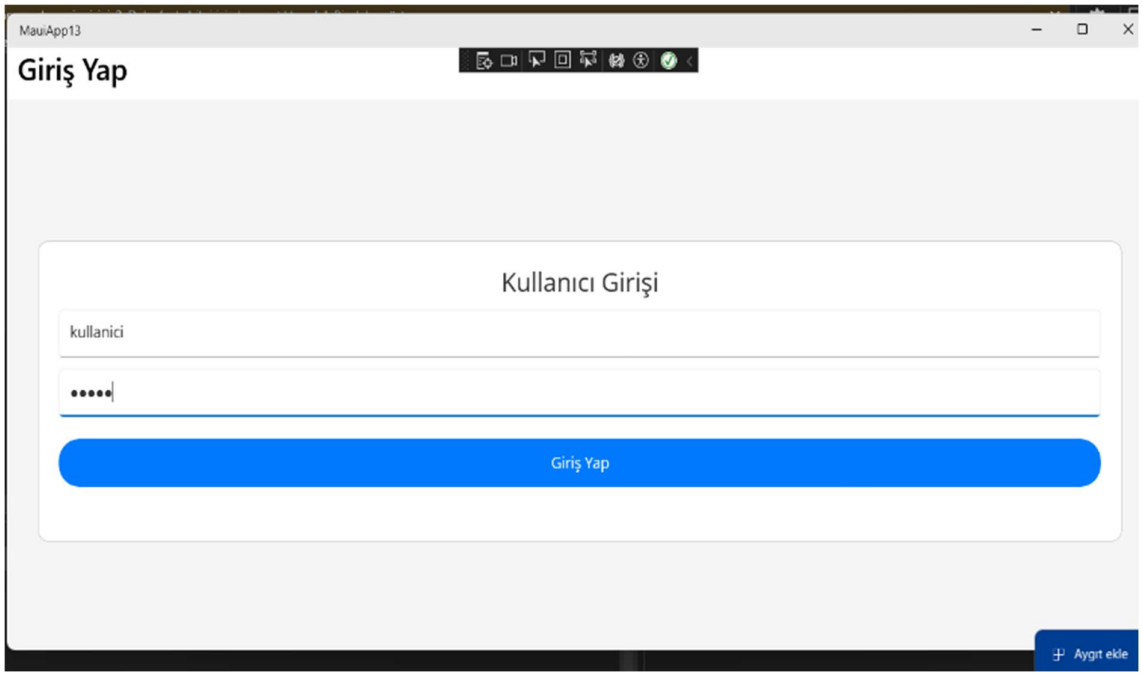


Şekil 7. GPS modülü devresi

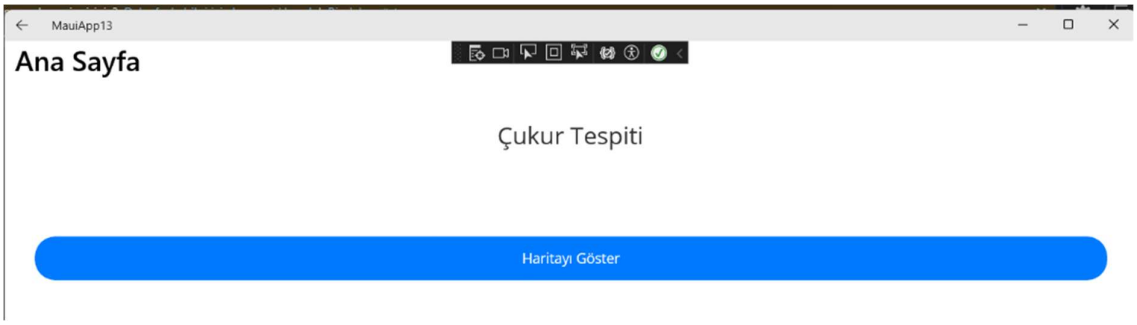
Kod içerisinde ilk olarak SoftwareSerial ve Adafruit_GPS kütüphaneleri kullanılarak GPS modülü tanımlanmaktadır. Setup fonksiyonunda, Arduino ve GPS modülü arasında 9600 baud hızında iletişim başlatılmakta, GPS'in her saniyede 1 Hz veriyi günceller ve haberleşme hızının 9600 baud(bit/s) olması sağlanmaktadır. Ana döngüde ise GPS modülünden gelen veri sürekli olarak okunmakta ve yeni bir NMEA cümlesi alındığında bu cümleler ayrışarak enlem ve boylamın koordinatlarını vermektedir. Elde edilen bu veri ile, Arduino'nun seri monitöründe gerçek zamanlı konum bilgileri görüntülenmektedir. Bu sayede, Arduino ile GPS modülü arasındaki seri haberleşme başarılı bir şekilde gerçekleştirilmiş olmakta ve koordinat bilgileri çalışmalarda kullanılabilir hale getirilmektedir. GPS kodu Arduino IDE'de çalıştırıldıktan sonra GPS modül dosyası Pycharm üzerinden çalışabilir hale gelmektedir.

Sonuçlardan yola çıkarak kod içerisinde sadece YOLOv8 kullanımı ile uygulamada haritada gösterilmesi için .NET MAUI programı kullanılmaktadır. Bu kısımda bir uygulama ile kullanıcı giriş ve şifre girilerek 'Giriş Yap' sayfası oluşturulmaktadır. Şekil 8'de kullanıcı ve şifrenin girilerek giriş yapılan kullanıcı girişi sayfası gösterilmektedir. Şekil 9'da giriş yapıldıktan sonra çıkan anasayfa görünmektedir. Haritayı göster butonuna basıldığında

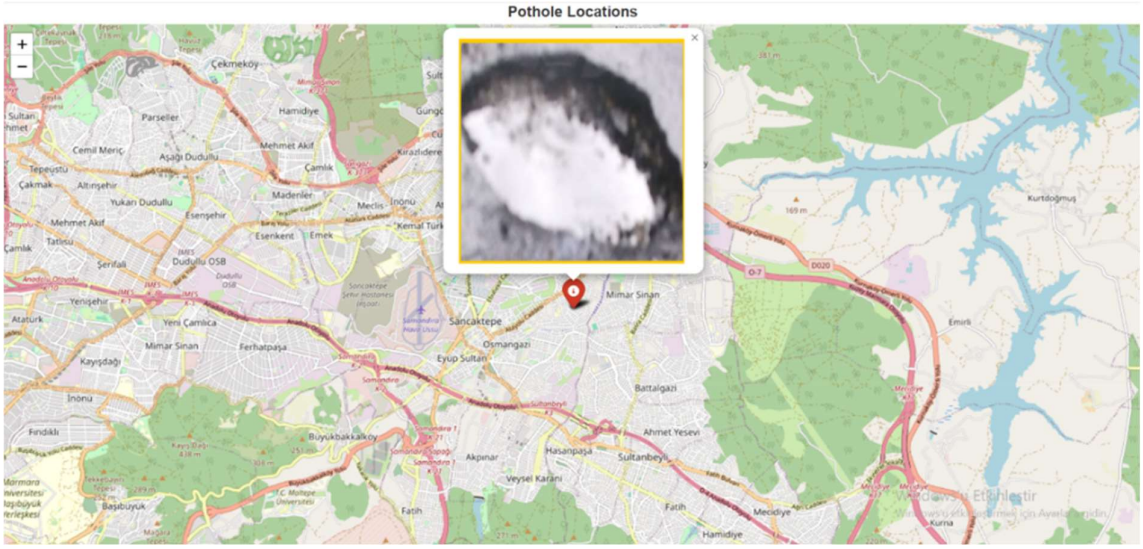
pothole_map.html dosyası devreye girmektedir. Şekil 10'da haritayı göster butonuna basıldığında ekrana gelen *pothole_map.html* sayfası gösterilmektedir.



Şekil 8. Uygulamada kullanıcı girişi sayfası



Şekil 9. Uygulamanın anasayfası

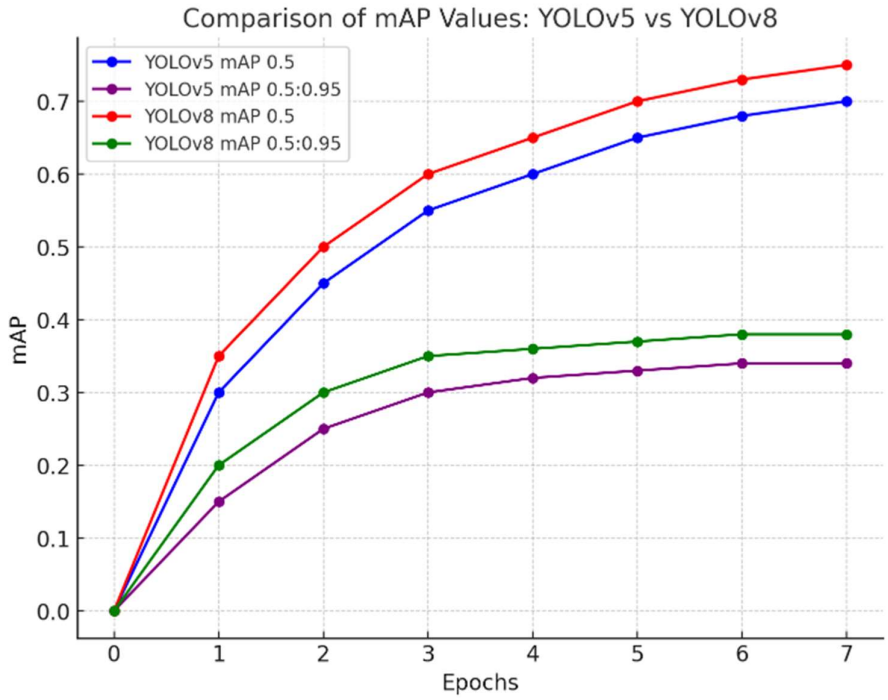


Şekil 10. Haritayı göster sayfası (pothole_map.html)

3. BULGULAR VE TARTIŞMA

YOLO modelinin V5 ve V8 sürümleri hem doğruluk hem hız açısından önemli gelişmeler göstermektedir. YOLOv5, hız ve doğruluk arasında dikkat edici bir uyum sağlamaktadır. Özellikle düşük bellek tüketimi ve hızlı işlem süresi sayesinde gerçek zamanlı uygulamalarda etkili bir performans sergilemiştir. YOLOv5'in çeşitli versiyonları (Küçük, Orta, Büyük) farklı hız ve doğruluk sunarken kullanıcının gereksinimlerine göre özelleştirilen çözümler sunmaktadır. YOLOv8, daha gelişmiş bir mimariye ve daha iyi öğrenme stratejisine sahip olup hem hız hem de doğruluk açısından önemli avantajlar sunmaktadır. YOLOv8'in yüksek çözünürlük desteği, daha karmaşık nesne tanıma görevlerinde yüksek doğruluk oranlarını sağlarken, optimize edilmiş hesaplama stratejileri sayesinde işlem hızını da artırmaktadır. Bu sürüm, büyük veri seti ile eğitim yapabilme kapasitesine sahip olup bu sayede daha geniş ve kapsamlı bir yetenek sunmaktadır. İki algoritma arasında karşılaştırmalar yapılabilmektedir. YOLOv8, YOLOv5'e göre daha güncel bir model olduğundan, genel olarak daha hızlı öğrenme, daha düşük kayıplar ve daha yüksek performans metrikleri göstermektedir. Eğitim eğrilerinin eğilimi, YOLOv8'in daha yüksek doğruluk ve tutarlılıkta daha iyi öğrenme gösterdiği görülmektedir. YOLOv5 ise hala güçlü bir model olmasına rağmen, bazı kayıp metriklerinde daha yavaş iyileşme ve kesinlik/duyarlılık metriklerinde daha fazla dalgalanma göstermektedir. YOLOv5'in özellikle daha karmaşık veri setlerinde YOLOv8 kadar hızlı uyum

sağlamayacağı düşünülmektedir. Şekil 11'de, çalışmada kullanılan YOLOv5 ve YOLOv8 algoritmalarının yapılan çukur tespitlerinden elde edilen mAP değerlerini karşılaştıran grafik gösterilmektedir. Elde edilen sonuçlara göre YOLOv8, her iki mAP metriğinde de YOLOv5'den daha iyi performans gösterildiği görülmektedir. mAP 0,5 için YOLOv8, özellikle sonlara doğru daha yüksek bir performans sunarak 0,75'i geçerken, YOLOv5 yaklaşık 0,7'de kalmaktadır. mAP 0,5:0,95 için ise YOLOv8 yine daha iyi sonuçlar vermekte ve yaklaşık 0,38'e ulaşılmaktadır. YOLOv5 ise yaklaşık 0,34'de kalmaktadır.



Şekil 11. YOLOv5 ve YOLOv8 mAP değerlerinin karşılaştırılması

Yapılan literatür taramasında daha önce bu veriseti ile sınıflandırma yapan bir çalışma tespit edilememiştir. Bu sebeple kullanılan iki algoritma sonucu hazırlanan karşılaştırma tablosu Şekil 12'de sunulmuştur. YOLOv8, %75 doğruluk YOLOv5'e %70 kıyasla daha iyi bir doğruluk sağlamaktadır. Duyarlılık (Precision) açısından YOLOv5 0,68 değerini gösterirken, YOLOv8 0,72 değeriyle yanlış pozitif tespitleri daha fazla azaltmıştır. Hassasiyet (Recall) metriğinde YOLOv8, 0,70 ile YOLOv5'in 0,65 değerine göre daha fazla doğruluk tespit gerçekleştirmiştir. F1 Score'da YOLOv8, 0,71 ile dengeli bir performans sunarken YOLOv5'in skoru 0,67'de kalmıştır. Ayrıca, mAP (0,5) ve mAP (0.5:0.95) değerlerinde YOLOv8, sırasıyla 0,75 ve 0,38 sonuçlarıyla YOLOv5'in (0,7 ve 0,34) üzerine

çıkmıştır. İşlem süresi açısından bakıldığında, YOLOv8'in 0,9 saniyelik süresi, YOLOv5'in 1,2 saniyelik süresine göre daha hızlıdır. Genel olarak, YOLOv5 hızlı ve bellek dostu bir model olmasına rağmen doğruluğu sınırlıdır. Buna karşın YOLOv8, daha yüksek hız, doğruluk ve optimizasyon sergilerken daha üstün bir performans göstermektedir.

Model/Yöntem	Doğruluk (Accuracy)	Precision (Duyarlılık)	Recall (Hassasiyet)	F1 Score	mAP (0.5)	mAP (0.5:0.95)	İşlem Süresi (s)	Yorum
YOLOv5	%70	0.68	0.65	0.67	0.7	0.34	1.2	Hızlı ve bellek dostu ancak doğruluk sınırlı
YOLOv8	%75	0.72	0.70	0.71	0.75	0.38	0.9	Yüksek doğruluk, daha hızlı ve optimize

Şekil 12. Algoritmalar karşılaştırılması ve yorumu

4. SONUÇ VE ÖNERİLER

YOLOv8, hem mAP 0,5 hem de mAP 0,5:0,95 metriklerinde YOLOv5'ten daha iyi bir doğruluk sunmaktadır. YOLOv8'in geliştirilmiş mimarisi veya optimizasyonları sayesinde daha başarılı bir tespit performansı sergilediği görülmüştür. Elde edilen sonuçlara göre YOLOv8, her iki mAP metriğinde de YOLOv5'den daha iyi performans göstermektedir. mAP 0,5 için YOLOv8, özellikle sonlara doğru daha yüksek bir performans sergileyerek 0,75'i geçerken, YOLOv5 yaklaşık 0,7'de kalmaktadır. mAP 0,5:0,95 için ise YOLOv8 yine daha iyi sonuçlar vermekte ve yaklaşık 0,38'e ulaşılmaktadır. YOLOv5 ise yaklaşık 0,34'de kalmaktadır. Bu grafikte görüldüğü gibi YOLOv8'in 0,38 mAP 0,5:0,95 seviyesine ulaşması, endüstri standardına yakın bir performans sunduğu gösterilmektedir.

Bu karşılaştırma, YOLOv8'in nesne tespiti performansını doğruluğunu ölçmek için kullanılan farklı IoU eşiklerinde daha iyi doğruluk sağladığını göstermektedir. Bu sebeple YOLOv8 daha yüksek doğruluk göstermesinden dolayı geliştirilen çukur tespiti programına YOLOv8 entegre edilmiştir. YOLOv5 algoritmasının da doğruluk ve hız gibi unsurları

değerlendirilmiştir. Pycharm ortamında oluşturulan YOLOv5 ve YOLOv8 algoritmaları kullanılarak çukur tespitleri yapılabilmektedir. Sonuç olarak, YOLOv8'in gelişmiş teknolojisi, yüksek doğruluk ve hız dengesi sağlamakla birlikte, YOLOv5'in etkili ve sağlam performansı, belirli uygulamalar için hala tercih edilen bir seçenek olabilmektedir. YOLOv8'in avantajları, özellikle yüksek doğruluk gerektiren ve büyük veri setine ihtiyaç duyan uygulamalarda öne çıkmaktadır. YOLOv5 için önerilen durum daha fazla verinin kullanımı ile ancak belirli bir doğruluğa erişilebilmektedir. Fakat daha geliştirilmiş bir model olan YOLOv8'i kullanmak hem doğruluk hem hız açısından çukur tespitinde tercih edilmiştir.

Çukur tespiti sadece 2 boyutlu görüntü verisiyle değil derinlik de ölçülerek 3 boyutlu olarak belirlenmektedir. Çukurun sadece varlığı değil, aynı zamanda derinliği de hesaplanarak çukurun tehlike derecesi de belirlenebilmektedir. Bu özellik, çukurların daha hassas bir şekilde analiz edilmesine ve sınıflandırılmasına olanak tanımaktadır, özellikle yol güvenliği açısından büyük önem taşınmaktadır. Çukur tespiti sırasında, tespit edilen çukurların GPS koordinatları otomatik olarak kaydedilmekte ve bu veriler bir harita üzerinde görselleştirilmektedir. Tespit edilen çukurların coğrafi konumlarının hızlı bir şekilde belirlenmesine ve raporlanmasına yardımcı olmakta, özellikle şehir yönetimleri ve yol bakım ekipleri için oldukça değerlidir.

Çalışmanın modüler yapısı, farklı nesne tespiti modelleri arasında kolayca geçiş yapma imkânı sağlamaktadır. Ayrıca, programın kolayca yeni modellere veya farklı veri kaynaklarına uygun olması, gelecekteki genişletmeler, iyileştirme ve geliştirme için güçlü bir temel sunmaktadır. Genellikle çukur tespiti zorlu bir bilgisayarla görü problemidir ve genellikle özel çözümler gerektirmektedir. Bu çalışma mevcut teknolojilerin bu özel problemde nasıl kullanılabileceğini göstermektedir. YOLOv5 ve YOLOv8 gibi farklı algoritmalar arasında geçiş yaparak, bu modellerin çukur tespitindeki doğruluk performanslarını değerlendirme imkânı sunmaktadır. Bu çalışma nesne tespiti ve derinlik ölçümünü bir araya getiren, farklı modelleri aynı anda karşılaştıran, gerçek zamanlı GPS entegrasyonuyla desteklenen ve modüler yapısıyla esnek bir şekilde genişletilebilen bir çözüm sunmaktadır. Özellikle yol güvenliği ve bakımında kullanılabilecek yenilikçi bir araç olarak öne çıkmaktadır. Bu özellikler, programın hem araştırma hem de endüstriyel uygulamalar için yüksek özgün bir değer taşımasını sağlamaktadır. İl özel idareleri, ilçe belediyeleri ilk önce uygulamadan tespit edilen çukurlar hakkında plan oluşturur, bu konuda nasıl önlem alacağı tartışılır ve hasarı gidermek için yol çalışması yapmaktadır.

TEŞEKKÜR

Bu çalışma, TÜBİTAK 2209-B (Proje Numarası: 1139B412300872) kapsamında desteklenmiştir. Projedeki desteklerinden dolayı Özlem Hakdağlı'ya teşekkür ederim.

KAYNAKÇA

[1] Teng, T.P., "Distress Identification Manual for the Long-Term Pavement Performance Project," U.S Department of Transportation Federal Highway Administration, 2014.

[2] Heggie, "Commercial Management and Financing of Roads," World Bank, Washington, 1998.

[3] Legreid A.M., "Potholes and Strategies on the Road to Campus Internationalization," International Research and Review: Journal of Phi Beta Delta Honor Society for International Scholars, vol. 6, no. 1, 2016.

[4] Rita Justo-Silva A.F., "Pavement maintenance considering traffic accident costs," International Journal of Pavement Research and Technology, 2019.

[5] Ümit Demir, Bora Uğurlu, "El Yazısından Kişilik Analizinde El Yazısı Tanımlamaya Yönelik Bir Karar Destek Modeli Önerisi", UMÜFED Uluslararası Batı Karadeniz Mühendislik ve Fen Bilimleri Dergisi, 3(2), ss, 47-71, 2021.

[6] Derin Öğrenme Nedir? : https://www.oracle.com/tr/artificial-intelligence/machine-learning/what-is-deep-learning/ [Ziyaret Tarihi: 19.10.2024]

[7] Ferdi Doğan, İbrahim Türkoğlu, "Derin Öğrenme Algoritmalarının Yaprak Sınıflandırma Başarımlarının Karşılaştırılması", Sakarya University Journal Of Computer and Information Science Vol. 1, Id. Saucis-1-2018, April 2018.

[8] Ultralytics Yolov5 Mimarlık: https://docs.ultralytics.com/tr/yolov5/tutorials/architecture_description/ [Ziyaret tarihi:03.09.2024]

[9] Yolov5 İle Nesne Tespiti: https://medium.com/@akdenizz7/yolov5-ile-nesne-tespiti-8aa370febfc0 [Ziyaret tarihi:03.09.2024]

[10]A Deep Dive Into Non-Maximum Suppression(NMS): https://builtin.com/machine-learning/non-maximum-suppression [Ziyaret tarihi:04.09.2024]

[11] Yolov8'in Gücünü Keşfedin: Yeni Nesil Nesne Tespiti Algoritması: https://medium.com/@meryemmsakinn/yolov8in-g%C3%BCc%C3%BCn%C3%BC-ke%C5%9Ffedin-yeni-nesil-nesne-tespit-algoritmas%C4%B1-d98efcda3e8d [Ziyaret tarihi:08.09.2024]

[12] Ultralytics Yolo Dökümanlar: https://docs.ultralytics.com/tr [Ziyaret tarihi:08.09.2024]

[13] Autonomous: Vehicles And Urban Traffic Management For Sustainbility: Impacts Of Transition Of Control And Dedicated Lanes : https://www.mdpi.com/2071-1050/16/19/8323 [Ziyaret Tarihi: 16.09.2024]

[14] Image Analysis In Autonomous Vehicles: A Review Of The Latest AI Solutions And Their Comparison: https://www.mdpi.com/2076-3417/14/18/8150 [Ziyaret Tarihi: 16.09.2024]

[15] Yol Yüzeyi Kusur Tespiti - Görüntü Tabanlıdan Görüntü Tabansıza : Bir Araştırma: https://arxiv.org/html/2402.04297v1 [Ziyaret tarihi:11.09.2024]

[16] Arduino Gps Modülü Kullanımı: https://devreyakan.com/arduino-gps-modulu-kullanimi/[Ziyaret tarihi:12.09.2024]

[17] Pothole Detection: https://www.kaggle.com/datasets/andrewmvd/pothole-detection [Ziyaret tarihi:14.09.2024]

[18] Sylvain Arlot. Alain Celisse. "A survey of cross-validation procedures for model selection." Statist. Surv. 4 40 - 79, 2010.