Araştırma Makalesi
Zeki Sistemler Teori ve Uygulamaları Dergisi 3(2) 2020: 31-37.
DOI: 10.38016/jista.710144

Bir Montaj Parçasının Derin Öğrenme ve Görüntü İşleme ile Tespiti

Gültekin Çağıl[1]* Büşra Yıldırım[2]
[1,2] *Sakarya Üniversitesi, Mühendislik Fakültesi, Endüstri Mühendisliği Bölümü, Sakarya, Türkiye*

Öz

Bu çalışma; kapı, pencere ve kış bahçesi üretimi yapılan bir firmada, montaj projelerinde kullanılmakta olan aksesuarların kamera tespiti ile adedini belirlemek üzere yapılmış ve çalışmada görüntü işleme ve derin öğrenmeye dayalı nesne tespiti yöntemlerinden yararlanılmış, çalışmada Google Colab Platformu, Python Programlama Dili, OpenCV kütüphanesi kullanılmıştır. Öncelikle, ilgili parçaya ilişkin fotoğraf görüntüleri çekilip, LabelImg Grafiksel Resim Ekleme Aracı ile etiketleme işlemi yapılmış ve programdan parçanın koordinatları metin dosyası halinde elde edilmiştir. Bu çıktılar test ve eğitim verisi olarak ikiye ayrılmıştır. Bu çıktılar, Google Colab Platformuna aktarılmış, Yolo Algoritması ile eğitme işlemi yapılmış, eğitim sonunda OpenCV Kütüphanesi kullanılarak hem kameraya erişim hem de parça sayım işlemi gerçekleştirilmiştir. İlgili firma yurtdışına müşteri isteğine bağlı olarak farklı projeler ve bu projelerin yanında çeşitli aksesuar parçaları göndermektedir. Aksesuar parçalarının her biri için ayrı bir kod bulunmaktadır. İşe yeni başlayan işçiler için bu ürünleri tanımak, kodlarını ezberlemek zaman alıcı ve yorucu bir iştir. Alışma sürecinde montaj prosesinde çalışan işçi, proje mühendislerinin hazırladığı kâğıt üzerinde bulunan, oluşturulacak üründe gönderilecek aksesuarların kodlarını bilmediğinden, bunu araştırmak için ekstra bir zaman harcayacak, bazen hatalara ve verimsizliğe neden olacaktır. Çalışmada verimsizliği önlemek ve hatalardan dolayı ortaya çıkacak olan maliyetleri ortadan kaldırmak amacıyla Görüntü İşleme ve Derin Öğrenme yöntemleri kullanılmış, çalışma neticesinde uygun sonuçlara ulaşılmıştır.

Anahtar kelimeler: Derin Öğrenme, Görüntü İşleme, OpenCV Kütüphanesi, Google Colab Platformu, Yolo Algoritması

Detection of an Assembly Part with Deep Learning and Image Processing

Abstract

This study was carried out to determine the number of accessories used in assembly projects at a company that produces doors, windows and winter gardens by using camera detection. For this purpose, object detection methods based on image processing and deep learning were used. Throughout the study, Google Colab Platform, Python Programming Language, OpenCV Library were used, thus, the detection and counting of the accessory parts in production were performed. First of all, photographs of the related parts were taken, labeling was done with the LabelImg graphic image insertion tool and the coordinates of the parts were obtained from the program as a text file. The outputs were divided into test and training data. Those outputs were transferred to the Google Colab Platform, training was carried out with the Yolo algorithm and at the end of the training, both camera access and part counting were performed using the OpenCV Library. Depending on the customer's request, the related company sends different projects and various accessory parts besides these projects. There is a separate code for each of the accessory part. It is time-consuming and tiring for the new workers to recognize these accessories and memorize their codes. It will require extra time for a new worker in the adaptation period to investigate the codes of the accessories that will be sent out on the paper prepared by the project engineers in the assembly process. Sometimes it will even cause mistakes and it will result in inefficiency. To prevent inefficiency in the assembly process and eliminate extra costs that may arise due to mistakes, Image Processing and Deep Learning methods were decided to be used. Following this study, appropriate results were achieved.

Keywords: Deep Learning, Image Processing, OpenCV Library, Google Colab Platform, YOLO Algorithm

* Sorumlu yazar.
E-posta adresi: cagil@sakarya.edu.tr

Alındı : Mart 27, 2020
Revizyon : Ağustos 10, 2020
Kabul : Ağustos 14, 2020

1. Giriş (Introduction)

Günümüzde nesne tespiti için kullanılan yöntemlerin uygulamaları her alanda görülebilmektedir. Bu çalışmada, nesne tespiti ve görüntü işleme konusu ayrıntıyla incelenmiş, diğer nesne tanıma algoritmalarından daha hızlı ve güvenilir netice veren Yolo Algoritması incelenerek, uygulamaya konulmuş, sonuçlar Python Programlama Dili, OpenCV Kütüphanesi ve Darknet Sinir Ağı Çerçevesi kullanılarak Google Colab Platformu üzerinde elde edilmiştir.

İnsanlar daha önce bildikleri nesneleri anında tanımaktadırlar. Günümüzde bilgisayarların da tıpkı insanlar gibi nesneleri tanımasına yoğun bir şekilde ihtiyaç görülmektedir. Nesne tespiti ve görüntü işleme için birçok hızlı ve doğru algoritmalar geliştirilmiş ve geliştirilmeye de devam edilmektedir. Görüntü işleme; elde bulunan bir probleme çözüm getirmek için ilgili görüntüyü dijital hale çevirip, çeşitli yöntemler kullanılarak görüntü üzerinde yapılan değişikliklerle istenilen çözümü elde etmek olarak tarif edilebilir. Derin öğrenme, bir makine öğrenme tekniğidir. Derin öğrenme ile etiketlenen veriler sayesinde tahminlerde bulunulmaktadır. Geliştirilen algoritmalar içinde günümüzde sık kullanılanlardan birisi de Yolo Algoritmasıdır. Yolo Algoritması diğer algoritmalara oranla nesneyi daha hızlı bir biçimde işlemekte ve tanımaktadır. Bundan dolayı bu çalışmada, uygulamanın yapıldığı ilgili firmada tanıtılacak olan aksesuar parçası için Yolo Algoritması tercih edilmiş, OpenCV Kütüphanesi kullanılarak görüntü üzerinde bulunan aksesuar parçasının kaç adet olduğu tespit edilmiş, Google Colab Platformu ile de seçilen aksesuar parçasının bilgisayara tanıtılması için gerekli olan eğitim yapılarak uygulamaya geçilmiştir.

İlgili firmada çok sayıda çeşitli aksesuar bulunmaktadır ve hepsinin ayrı bir kodu vardır. Yeni gelen bir işçi kodları öğrenmede zorluk çekmekte ve hatalı durumların ortaya çıktığı görülebilmekte, sonucunda zaman ve para kaybına neden olabilmektedir. Çalışmada yukarıda bahsedilen yöntemler kullanılarak bu olumsuzlukların önüne geçilmesi amaçlanmıştır. Bilinmektedir ki artık bilgisayarlar insanlardan daha hızlı öğrenmekte ve hata yapma olasılığı insana kıyasla daha az olmaktadır. Bunda görüntü işleme ve nesne tespiti ile kullanılan yöntemlerin payı oldukça büyüktür.

Pu ve arkadaşı (2020)'de Faster-RCNN, R-FCN, SSD ve YOLO v3 algoritmalarını kullanarak çıkan yangınların erkenden tespit edilip kayıpların önlenmesi adına bir sistem geliştirmişlerdir. Çalışmalarında algoritmaları eğitmek için belli kurumlardan ve internetten aldıkları 29.180 adet yangın görüntülerini kullanmışlardır. Görüntüleri etiketleme işlemlerini 4 kategoriye ayırarak gerçekleştirmişleridir. Bu 4 kategori şu şekildedir; yangın, duman, yangına benzeyen ve dumana benzeyen. Veri setindeki yangın görüntülerinin sayısı 13.400'dür. Veri seti 9695 "ateş" ve 7442 "duman" nesnesi içermektedir. Ayrıca, veri setinde yangın içermeyen 15.780 görüntü bulunmaktadır. Bu çalışma için, veri setindeki görüntülerin %50'sini eğitim %50'sini test olarak kullanılmışlardır. Eğitimi kullanılan 4 algoritma için ayrı ayrı gerçekleştirmişlerdir. Karşılaştırma sonucunda YOLO v3 algoritmasının görüntülerdeki yangını %83,7 doğrulukla en hızlı şekilde tespit ettiği sonucuna ulaşmışlardır.

Hendry ve arkadaşı (2019)'da Yolo Algoritması ve Darknet Sinir Ağı Çerçevesini kullanarak otomatik araç plakası tespit edilmesiyle ilgili bir sistem geliştirmişlerdir. Yapılan bu çalışmada Sliding-Window Single Class Detection (Kayan-Pencere Tek Sınıf Tespiti) yaklaşımını kullanmışlardır. Burada kayan pencerelerin hesaplanması için 25 harften (Tayvan plakalarında O ve sıfır aynıdır), 10 rakamdan ve bir plaka etiketinden, yani toplamda 36 sınıf etiketinden karakter kullanmışlar ve BBox Etiketleme Aracı ile eğitimde rol alacak her karakteri sınırlayıcı kutular çizerek etiketlemişlerdir. Bir görüntüde birden fazla sınırlayıcı kutu bulunabilmektedir ve onlar bir sınıf etiketinin bir eğitim modeline ait olduğu tek bir sınıf tespit modelini kullanarak çalışmalarını yürütmüşlerdir. Kurdukları sistemde BBox Etiketleme Aracının çıktı değerleri Yolo formatına uygun olmadığından, bu değerleri Yolo formatına uygun hale getirmişlerdir. Böylece etiketleme sonucu elde edilen çıktı değerlerini kayan pencere tek sınıf tespiti yaklaşımı ile Yolo'ya yerleştirmişlerdir. Geliştirdikleri sistem, bir görüntüde bulunan tüm nesneleri tespit ettikten sonra plakanın lokasyonunu tanımlamaktadır. Bu sistemi eğitmek için Tayvan araç plakalarının bulunduğu 2049 adet resimden oluşan AOLP veri setini kullanmışlardır. Çalışma sonunda 2000 eğitim adımından sonra sistem plaka algılamada %98.22, plaka tanımada ise %78 doğruluk sağlamıştır.

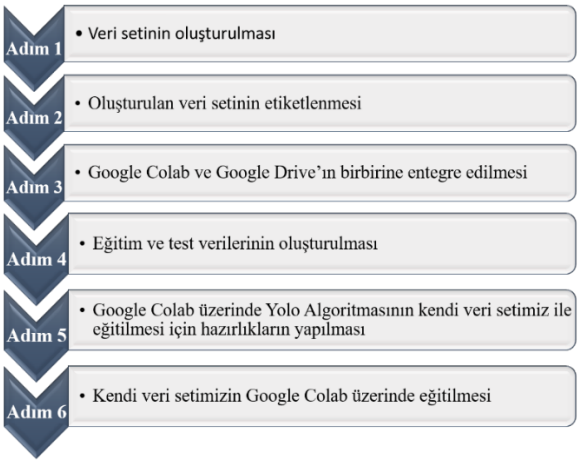
Shinde ve arkadaşları (2018)'de Yolo tabanlı insan eylemi tanıma ve lokasyonunu belirme üzerine bir sistem geliştirmişlerdir. Sistemi eğitmek için LİRİS insan faaliyetleri veri setini kullanmışlar ve bu veri setini Yolo için kullanılabilir etiket dosyalarına dönüştürmüşlerdir. Yapılan bu çalışmada bir video dizisinden periyodik kareler işlemişler ve insan-insan, insan-nesne, insan-insan-nesne etkileşimlerini içeren 10 adet görsel açıklamalı eylem kullanmışlardır. Çalışma süresince kullandıkları veri seti 167 videodan 367 eylem içermektedir. Geliştirdikleri sistemi, eğitim için ayırdıkları veri setinden uygun hareket içeren kareler kullanarak eğitmişler ve bu süreçte her iterasyon 5.25 saniye sürmüş, 40000 iterasyon ile eğitim sonlanmıştır. Çalışma sonunda test için sistemin çerçevedeki hareketi tanıması ortalama 61 ms sürdüğünü görmüşler ve elde edilen doğruluk oranını %88.372 bulmuşlardır.

2. Materyal ve Yöntem (Material and Method)

Bu çalışma kapı, pencere ve kış bahçesi üretimi yapan bir firmada gerçekleştirilmiştir. Firmada projelerde kullanılan bir aksesuar parçası Görüntü

İşleme ve Derin Öğrenme yöntemleri ile tespit edilmiş ve sayımı yapılmıştır. Çünkü yeni giren işçiler için aksesuar parça kodlarını öğrenmek zaman alan bir iştir ve öğrenme sürecinde hatalar ortaya çıkabilmektedir.

Bundan dolayı belirlenen aksesuar parçası üzerinde bir çalışma yapılmıştır. Yukarıda bahsedilen problemin çözümü için Şekil 1'deki adımlar sırayla uygulanarak çözüm elde edilmiştir.



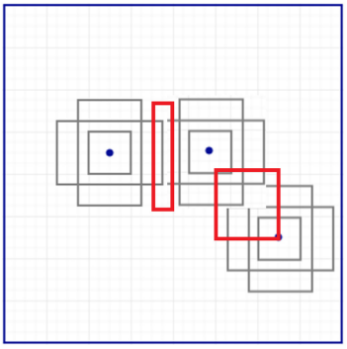
Şekil 1. Problemin çözümü için takip edilecek aşamalar (Stages for the solution of the problem)

2.1. Yolo Algoritması (Yolo Algorithm)

Yolo (You Look Only Once) "sadece bir kere bak" demektir ve son zamanlarda ortaya çıkmış, nesne tespitinde kullanılan bir algoritma olup gerçek zamanlı görüntü işleme için kullanılan bir nesne algılama sistemidir.

Yolo Algoritması diğer algoritmalara oranla hızlı tahmin yapabilmektedir. Nedeni şudur: Yolo resme tek bir CNN (Convolutional Neural Network, Evrişimsel Sinir Ağları) uygulayıp, resmi ızgaralara böler, sınırlayıcı kutular ve ilgili güven puanının tahminini her ızgara için hesaplar, sınırlayıcı kutular tahmin edilen güven puanı ile analiz edilir (Shindea-Kotharia-Guptab, 2018).

Yolo gerçek zamanlı işleme için hedeflenen konvolüsyonel bir sinir ağı nesnesi algılama sistemidir. Şekil 2. Yolo algoritmasının kurallarının, bir görüntüdeki nesnelerin algılanmasını nasıl sınırladığını göstermektedir. Bu sınırlar, tespit edilen nesneler arasındaki kalın dikdörtgenler ile gösterilir. (Hendry-Ching Chen, 2019) Yolo'nun giriş görüntülerini N ∗ N ızgara hücresine bölmektedir. Her bir ızgara hücresi yalnızca bir nesneyi ve sabit sayıda sınır kutusu tahmin eder. Mavi nokta, gri sınırlayıcı kutularla çevrili algılanan nesnenin merkezidir.



Şekil 2. Yolo algoritmasının ızgara hücresinde çalışma prensibi (Working principle of Yolo algorithm in grid cell)

Yolo algoritması görüntüleri yaklaşık 40-90 FPS (Frames per second, saniye başına kare)'de işleyebilir. Bundan dolayı diğer yöntemlere göre oldukça hızlıdır. Bu bir videonun gerçek zamanlı olarak birkaç milisaniye gecikme ile Yolo Algoritması tarafından işlenebileceğini gösterir (Shindea-Kothari-Gupta, 2018).

Diğer bir nesne tespiti yöntemi olan R-CNN ile karşılaştırıldığında Yolo, R-CNN'den 1000 kat daha, Faster R-CNN ile karşılaştırıldığında ise 100 daha hızlı olduğunu söylenmektedir (Shindea-Kothari-Gupta, 2018).

2.2. Nesne Tespiti ve Derin Öğrenme (Object Detection and Deep Learning)

Nesne tespiti nesnenin ait olduğu sınıfın örneğini belirleme prosedürüdür (R.Pathak-Pandey-Rautaray, 2018).

Nesne tespitinin amacı, görüntüdeki nesneleri etiketleyerek nesnelerin yerlerinin tahminin yapılmasıdır (Shafiee- Chywl- Li- Wong, 2017).

Görüntü sınıflandırma, nesne tespiti ve doğal dil işleme alanlarındaki gelişmelerden dolayı derin öğrenme günümüzde çok kullanılan bir sözcük haline gelmiştir. Derin öğrenmenin popülaritesinin nedenleri, büyük veri setlerinin ve grafik işleme birimlerinin kullanılabilirliğidir. Derin öğrenmede eğitim için büyük veri kümelerine ve güçlü kaynaklara ihtiyaç vardır. Gerekli olan bu iki faktör şu anki dönemde mevcut olarak bulunmaktadır (R.Pathak-Pandey-Rautaray, 2018).

2.3. Google Colab Platformu (Google Colab Platform)

Google Colab'tan bahsetmeden önce Jupyter Notebook'u tanıtmak gerekir. Jupyter, çeşitli yazılım dillerini ve kütüphanelerini birleştiren açık kaynaklı ve tarayıcı tabanlı bir araçtır. Google Colab ise Jupyter Notebook'lara dayalı makine öğrenmesi ve eğitimini yaygınlaştırmak amacıyla oluşturulan bir bulut hizmetidir. Derin öğrenme için tamamen yapılandırılmış bir runtime ve güçlü bir GPU

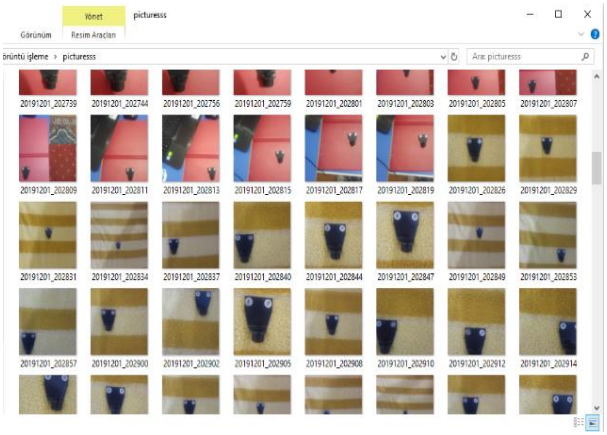
kullanımına ücretsiz erişim sağlar (Carneiro- NóBrega-Nepomuceno- Bian- Albuquerque- Filho, 2018).

3. Önerilen Yöntem (Proposed Method)

Şekil 1.'de verilen problemin çözümü için uygulanacak adımlar bu kısımda detaylı olarak açıklanmıştır.

3.1. Veri Setinin oluşturulması (Creating of a Dataset)

İlk olarak tanıtılması istenilen aksesuar parçası, ilgili firmadaki depodan seçilmiş olup bu parçanın, farklı arka planlarda ve farklı açılarda 312 adet fotoğrafı çekilmiştir. Parçanın farklı arka plan ve açılarda fotoğraflarının çekilmesinin nedeni, programın uygulama sonucunda herhangi bir ortam fark etmeksizin nesneyi tanıma olasılığının yüksek olmasının istenmesidir.



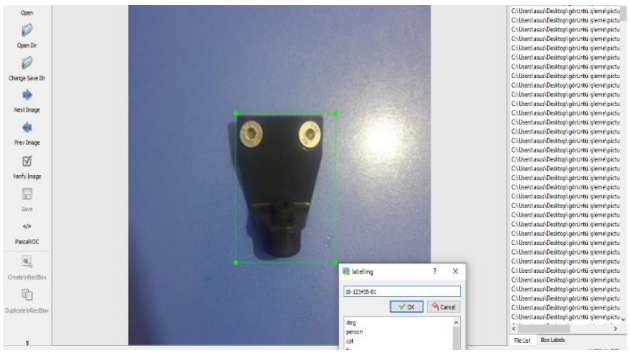
Şekil 3. Veri setinin oluşturulması (Creation of dataset)

Fotoğrafları çekme işleminin bitmesinin ardından, PhotoScape uygulaması kullanılarak boyutları 416x416 piksel haline getirilmiştir. Aslında piksel sayısı arttıkça program daha tutarlı bir sonuç vermekte ama eğitim zorlaşmaktadır. Çünkü, örnek olarak; çekilen fotoğraflar 600x600 piksel olarak ayarlandığında program 360000 parametre kullanmak zorunda kalmakta, bu da eğitimin zorlaşmasına neden olmaktadır. Fotoğraflar 416x416 piksel haline dönüştürülerek programın 173056 parametre ile çalışması sağlanmış, böylece eğitim süreci kolaylaştırılmıştır.

3.2 Oluşturulan Veri Setinin Etiketlenmesi (Tagging The Created Dataset)

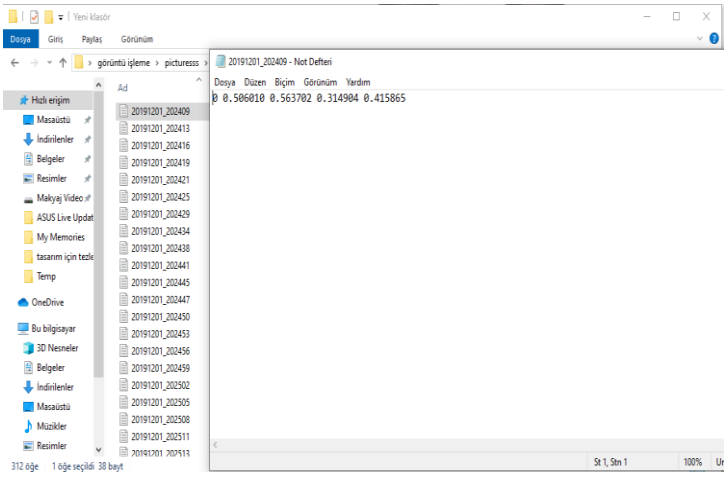
Çekilen fotoğrafların etiketlenerek koordinatlarının belirlenmesini sağlamak amacıyla LabelImg programı kullanılmıştır.

Programın açılmasının ardından "OpenDir" butonuna tıklayarak tüm görüntülerin bulunduğu dosya seçilmiş ve bütün görüntüler programa eklenmiştir. Etiketleme işlemine başlanmadan önce "PascalVOC" formatı "Yolo" formatına dönüştürülmüştür. Daha sonra tüm görüntüler tek tek etiketlenerek görüntüde bulunan aksesuar parçasına ait kod girilmiştir. Etiketleme yaparken görüntüdeki parçanın sınırlarının fazla aşılmamasına dikkat edilmiştir. Çünkü parça sınırları fazla aşılırsa algoritmanın nesneyi tanıma olasılığı düşecektir.



Şekil 4. Görüntüleri etiketleme işlemi (The process of tagging images)

Etiketleme işleminin sonunda bütün etiketli formatta bulunan görüntüler kaydedilmiş ve etiketlenen her bir görüntünün koordinatlarının metin dosyası halinde bulunduğu bir klasör oluşmuştur.



Şekil 5. Etiketli görüntülerin koordinatları (Coordinates of tagged images)

3.3 Google Colab ile Google Drive'ın Birbirine Entegre Edilmesi (Integration Of Google Colab And Google Drive)

Bütün bu işlemlerin ardından Google Colab'ta eğitim hesabı açılmıştır. Bilgisayarda hem görüntülerin hem de etiketleme işlemi sonrasında oluşan metin dosyalarının bulunduğu klasör Google Drive'a yüklenmiştir. Bu klasörün Google Colab'a aktarımının yapılabilmesi için gerekli kodlar Google Colab üzerinde yazılarak Google Drive ile entegre hale getirilmiştir. Böylece bilgisayarda bulunan klasör artık Google Colab üzerinde de ulaşılabilir hale gelmiştir.

3.4 Eğitim ve Test Verilerinin Oluşturulması (Creating Of Training and Test Data)

Çalışmada kullanılan Yolo algoritmasını kendi verilerimiz ile eğitilebilmemiz için bu verilerin test ve eğitim olmak üzere ikiye ayrılması gerekmektedir. Bundan dolayı çekilen görüntülerin yüzde 10'u test, yüzde 90'ı eğitim için kullanılmıştır. Test ve eğitimde kullanılacak görüntüler random olarak seçilmiştir. "test.txt" ile "train.txt" dosyalarında kullanılacak görüntüler belirlenmiştir. Çalışmada kullanılan bilgisayar ile daha uyumlu olduğundan Yolo'nun tiny versiyonu seçilmiştir.

Şekil 6. Test ve eğitim verilerinin oluşturulması (Creation of test and training data)

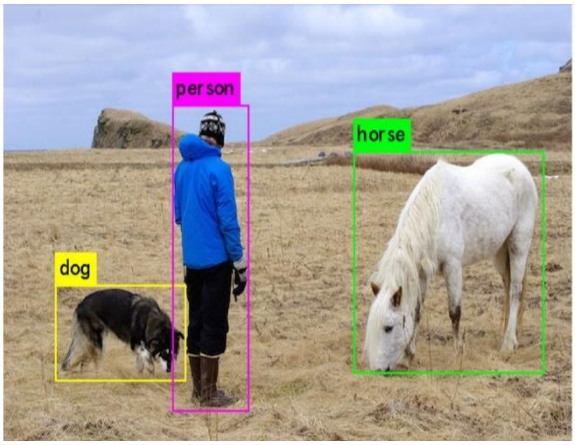
3.5 Google Colab Üzerinde Yolo'yu Kendi Veri Setimiz ile Eğitebilmek İçin Hazırlıkların Yapılması (Making Of Preparations To Train Yolo On Google Colab With Our Own Dataset)

Google Colab üzerinde Yolo algoritmasını ilk başta Yolo orijinal sitesinde bulunan başlangıç katsayıları ile eğitmemiz gereklidir ki kendi veri setimizin eğitimini başlatabilelim. Bunu için ilk olarak C ve CUDA ile yazılmış açık kaynaklı bir sinir ağı çerçevesi olan Darknet kurulmuştur.

Daha önceden veri setimizin bulunduğu klasörü Google Colab'a Google Drive aracılığı ile yüklemiştik. Bu klasör gerekli kodlar kullanılarak Darknet ile entegre hale getirilmiştir. Eğitim ve test olarak kullanacağımız görüntüleri random seçip test.txt ve train.txt dosyalarına atmıştık. Bu dosyalar gerekli kodlar yazılarak Google Colab'a yüklenmiştir. Yolo'nun kendi sitesinde bulunan başlangıç katsayıları Colab üzerine indirilmiştir.

Kendi veri setimizi eğitmeden önce görselleri göstermemize ve dosyaları yükleyip, indirmemize izin verecek olan fonksiyonlar tanımlanmıştır.

İnternetten alınan örnek bir uygulama ile Yolo'nun çalışabilirliği test edilmiştir.



Şekil 7. Örnek uygulamanın çıktısı (Output of the sample application)

Bu test başarı ile sonuçlanmıştır. Resimde de görüldüğü gibi algoritma insanı, köpeği ve atı doğru bir şekilde tanımıştır.

3.6 Kendi Veri Setimizin Google Colab Üzerinde Eğitilmesi (Training Of Our Own Dataset On Google Colab)

Yolo'nun kendi sitesinden başlangıç katsayıları indirilmişti ve bu katsayılar bir önceki adımda internetteki örnek bir uygulama ile eğitilmişti. Eğitilmiş katsayılar 1000.weights dosyasına kaydedilmiştir. Daha sonra kendi veri setimiz ile eğitim yapılmış, eğitim yaklaşık olarak 3 saat sürmüş ve 1000 iterasyon ile sonlanmıştır. Algoritma 1000 iterasyon sonunda eğitilmiş katsayıları 1000.weights dosyasına kendi kaydetmiştir. Eğitimin başından sonuna kadar iyileşmeler olduğu görülmüştür. Eğitim başında hata oranı 600'lerde iken eğitim sonrasında bu oran 0.2'lere kadar düşmüştür.

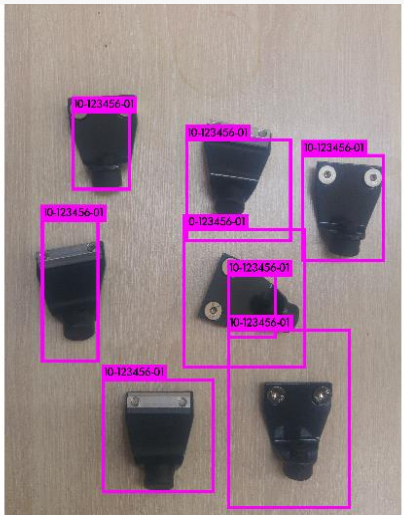
Eğitim sonunda elde edilen katsayılar kullanılarak algoritmanın çalışıp çalışmadığı Colab üzerinden gerekli kod yazılarak çekilen görüntülerden bir tanesi kullanılarak test edilmiş ve Yolo resimdeki 7 parçayı da doğru bir şekilde tanımıştır. Görüntüdeki 7 parça 262.58400 milisaniyede tahmin edilmiştir. Parçaların tahmin olasılıkları aşağıdaki gibidir:

```
data_for_colab/data/deneme4.jpg: Predicted in 262.584000 milli-seconds.
10-123456-01: 88%
10-123456-01: 97%
10-123456-01: 87%
10-123456-01: 18%
10-123456-01: 93%
10-123456-01: 16%
10-123456-01: 100%
10-123456-01: 95%
```

Şekil 8. Eğitimden sonra sistemi örnek bir görüntü ile test etme (Testing the system with a sample image after training)

Görüntü üzerinde Yolo'nun parçaları nasıl etiketlediğini de görmek istediğimizden aşağıdaki kod yazılarak görüntü alınmıştır.

```
[ ]  imShow('predictions.jpg')
```



Şekil 9. Uygulama sonucunda elde edilen örnek bir çıktı (A sample output obtained as a result of the application)

Gerekli kodlar Python üzerinde yazılarak, aynı zamanda OpenCV kullanılarak kameraya erişilmiştir. Kendi eğittiğimiz katsayılar OpenCV'ye gönderilmiştir. OpenCV'nin DNN (Deep Neural Networks) modülü kullanılarak görüntüde bulunan nesneler tahmin edilmiş, her bir nesnenin etrafına kutu çizilmiş ve Yolo'nun görüntü üzerinde kaç adet nesneyi tanıdığı çıktının üzerine yazılmıştır.

Kamera ile görüntü üzerinde test yapılabilmesi için Anaconda Prompt üzerine gerekli olan kod yazılıp çalıştırılmış ve kameraya bağlanılarak test yapılmıştır.

Yapılan test sonucunda Yolo'nun 4 adet parçanın dördünü de tanıyabildiği görülmüştür.

Şekil 10. Videodan gerçek zamanlı elde edilen çıktı (Real time output from the video)

Yapılan literatür taraması sonucu Yolo Algoritmasının Google Colab Platformu üzerinde bir örneği görülmemiştir. Bununla beraber Yolo Algoritması nesne tespiti için kullanılmış, eğitim sonunda tanıtılan nesnelerin saydırılma işlemi ek olarak yapılmamıştır. Bu çalışmada Yolo Algoritmasının Google Colab Platformu üzerinde yazılmış olması ve tanıtılan parçanın OpenCV ile sayım işleminin yapılması, çalışmanın yenilikçi yaklaşımını göstermektedir.

4. Sonuçlar (Conclusions)

Üretim yapılan fabrikalarda genellikle üretilecek ürün için çok fazla alt malzeme grupları bulunmakta ve fabrikalar her malzemeye ayrı bir kod atamaktadır. İşe yeni başlayan işçiler için malzemeleri tanımak, kodlarını ezberlemek zaman alıcı ve yorucu bir iş olduğundan hatalar meydana gelebilmektedir.

Bu çalışmada hatalı durumları önleyebilmek adına belirlenen aksesuar parçası; Python Programlama Dili, Google Colab Platformu, Darknet Sinir Ağı Çerçevesi, LabelImg Resim Etiketleme Aracı, Yolo Algoritması ve OpenCV Görüntü İşleme Kütüphanesi kullanılarak bilgisayara tanıtılmıştır. Görüntü üzerindeki nesneler Yolo Algoritması ile etiketlenip ilgili parça tespit ve test edilmiş, aynı zamanda bu işlem kamera kullanılarak gerçek zamanlı olarak da denenmiştir. Sistem çıktısında bulunan sayım işlemi Python üzerinde OpenCV kullanılarak yazılan kod ile yapılmıştır.

Uygulama sonunda kamera ile 25 deneme yapılmış ve 21 denemede sistem görüntü üzerinde bulunan parçaları tam olarak tanımıştır. Bu bilgiler ışığında çalışmanın %84 oranda doğru çalıştığı görülmüştür.

Oluşturulan bu sistemin ilgili firmada üretim hatlarında kullanılması, yurtdışına gönderilen yanlış parça durumlarını azaltarak çıkan ek maliyeti düşürecek, işçinin kodu ezbere bilmemesinden kaynaklanan malzeme arama süresini azaltarak, üretimin verimliliğini arttıracaktır. Böylece firma katlandığı bazı olumsuz durumlardan kurtulacaktır.

Teşekkür (Acknowledgment)

Bu çalışmanın gerçekleştirilmesinde, vaktini ayırıp değerli bilgilerini bizim ile paylaşan, yardımlarını ve desteğini bir an olsun eksik etmeyen, çalışmanın gerçekleştirildiği firmanın IT Müdürü Nahsen KAYHAN'a teşekkürü borç biliriz.

Kaynaklar (References)

Carneiro, T., NóBrega, R.V.M., Nepomuceno, T., Bian, G., Albuquerque, V.H., Filho, P.D., (2018), "Performance Analysis of Google Colaboratory as a Tool for Accelerating Deep Learning Applications", Makale, IEEE Access, Cilt 6, Sayfa 61677-61685.

Hendry, Rung, Chen C., (2019), "Automatic License Plate Recognition via sliding-window darknet-YOLO deep learning", Makale, Image and Vision Computing, Cilt 87,

Sayfa 47-56, Chaoyang University of Technology- Satya Wacana Christian University, Tayvan-Endonezya.

Kunduracı M.F., (2019), "Görüntü İşleme Yöntemleri Kullanarak Araç Marka ve Modelinin Tespit Edilmesi", Yüksek Lisans Tezi, Bilişim Teknolojileri Mühendisliği Anabilim Dalı, Konya.

Li P., Zhao W., (2020), "Image fire detection algorithms based on convolutional neural networks", Makale, Case Studies in Thermal Engineering, Cilt 19, Çin

Pişkin(2016), http://mesutpiskin.com/blog/opencv-nedir.html . (25 Aralık 2018)

Pogorelov, K., Riegler, M., Eskeland, S. L., Lange, T., Johansen, D., Griwodz, C., Schmidt P. T., Halvorsen, P., (2017), "Efficient disease detection in gastrointestinal videos – global features versus neural networks", Makale, Multimedia Tools and Applications, Cilt 76, Sayfa 22493–22525, Norweç.

R.Pathak, A., Pandey, M., Rautaray, S., (2018), "Application of Deep Learning for Object Detection", Makale, Procedia Computer Science, Cilt 132, Sayfa 1706-1717, Hindistan.

Redmon, J., Divvala, S., Girshick, R., Farhadi, A., (2015), "You Only Look Once: Unified, Real-Time Object Detection", Makale.

Shafiee, M. J., Chywl, B., Li, F., Wong, A., (2017), "Fast YOLO: A Fast You Only Look Once System for Real-time Embedded Object Detection in Video", Makale, Kanada.

Shinde, S., Kothari, A., Gupta, V., (2018), "YOLO based Human Action Recognition and Localization", Makale, Procedia Computer Science, Cilt 133, Sayfa 831-838, Hindistan.

https://github.com/tzutalin/labelImg (23.10.2019)

https://pjreddie.com/darknet/yolo/ (15.11.2019)