Fırat Üniversitesi Müh. Bil. Dergisi
31(2), 571-581, 2019

Derin Öğrenme ile Resim ve Videolarda Nesnelerin Tanınması ve Takibi

Resul DAŞ[1], Berna POLAT[2], Gürkan TUNA[3]
[1,2]Fırat Üniversitesi, Teknoloji Fakültesi, Yazılım Mühendisliği Bölümü, 23119, Elazığ.
[3]Trakya Üniversitesi, Teknik Bilimler MYO, Bilgisayar Teknolojileri Bölümü, 22020, Edirne.
[1*] rdas@firat.edu.tr, [2] brnpolatt@gmail.com, [3] gurkantuna@trakya.edu.tr

(Geliş/Received: 21/07/2019; Kabul/Accepted: 29/08/2019)

Öz: Görüntü işleme yöntemleri kullanılarak durağan görüntülerin analizleri gerçekleştirilebilir ve söz konusu görüntülerden anlamlı bilgiler çıkarılabilir. Tespit ve tanıma sonrasında takip edilecek olan nesnenin değişken bir ortam içinde bulunması zorlaştırıcı unsurlardan birisidir. Bunun gibi zorlaştırıcı unsurlarla başa çıkabilmek ve nesne takibini başarıyla gerçekleştirebilmek için farklı yöntemler geliştirilmiştir. Askeri uygulamalarda yaygın olarak kullanılan elektro-optik algılayıcı sistemleri hareketli ve sabit hedeflerin belirlenmesini sağlamaktadır. Son yıllarda yapay zekâ tabanlı bileşenlerle güçlendirilen bu sistemler hem daha hızlı hem de daha kesin hedef tespiti yapmayı sağlamaktadır. Öte yandan, derin öğrenme algoritmaları yapay zekâ alanında bir devrim yaratmıştır. Derin öğrenme algoritmalarının görüntü işlemede kullanılması oldukça başarılı sonuçlar alınmasını ve karmaşık görüntü işleme problemlerinin kolaylıkla çözüme kavuşturulabilmesini sağlamaktadır. Bu çalışmada derin öğrenme ile hareketli nesne tanıma ve takibi için Google'ın açık kaynak kodlu makine öğrenmesi kütüphanesi olan TensorFlow kullanılmıştır. Nesne takibi için Region Based Convolutional Networks kütüphanesinden Faster R-CNN modeli ele alınmıştır. Bu kütüphaneler ile durağan görüntüler, video görüntüleri ve webcam görüntüleri üzerinde nesne tanıma işlemi gerçekleştirilmiş ve incelenen kütüphanelerin güçlü ve zayıf yönleri ortaya konmuştur.

Anahtar kelimeler: Nesne tanıma, nesne takibi, Faster R-CNN, görüntü işleme, derin öğrenme.

Recognition and Tracking of Objects in Pictures and Videos Using Deep Learning

Abstract: Analysis of still images using image processing methods can be performed and meaningful information can be extracted from the images. The presence of the object to be followed in a dynamic environment after detection and recognition is one of the factors that make it difficult. Different methods have been developed in order to cope with such challenging factors and to carry out object tracking successfully. Electro-optical sensor systems, which are commonly used in military applications, enable the determination of moving and fixed targets. In recent years, these systems, which are reinforced by artificial intelligence-based components, provide both faster and more precise target detection. On the other hand, deep learning algorithms have revolutionized the field of artificial intelligence. The use of deep learning algorithms in image processing provides very successful results and easy to solve complex image processing problems. In this study, TensorFlow, Google's open source machine learning library, is used for deep learning and moving object recognition and tracking. Faster R-CNN model from the Region Based Convolutional Networks libraries was used for object tracking. With these libraries, object recognition was performed on still images, video images and webcam images, and the strengths and weaknesses of the libraries examined were revealed.

Key words: Object recognition, object tracking, Faster R-CNN, image processing, deep learning, open source library.

1. Giriş

Dijital görüntü işleme uygulamalarının vazgeçilmez unsurlarından olan nesne tespiti ve nesne tanıma uzun yıllardır üzerinde çalışılan konulardır. Nesne tespiti ve nesne tanıma için farklı algoritmalar ve yöntemler geliştirilmiştir. Dijital görüntülerdeki nesnelerin hızlı tespitini etkin biçimde gerçekleştiren ilk algoritma Viola Jones [1] olmuştur. Son yıllarda grafik işleme birimlerindeki gelişmeler ve derin öğrenme sayesinde daha fazla doğruluk oranı ile nesne tespiti ve tanımlama yapabilen yöntemler geliştirildi. Literatürde nesne takibinin genel olarak dört farklı aşamada ele alınmaktadır. Bu aşamalar ön işlemler, nesne tespiti, nesne sınıflandırma ve nesne takibidir [2]. Bu aşamalardan özellikle nesne tespiti daha fazla önem taşımaktadır ve bu aşamanın başarısı sonraki aşamalarının başarısını da etkilemektedir. Nesne tespiti genel olarak video imgelerinde nesnenin belirginleşmesi ve işlenecek olan nesnenin arka plandan ayrılması olarak tanımlanabilir. Nesne tespiti ve tanımada kullanılan popüler kütüphaneler arasında Single Shot Multibox Detactor (SSD), Region Based Convolutional Networks (R-CNN), Fast R-CNN, Faster R-CNN ve Mask R-CNN bulunmaktadır [2].

[1] Sorumlu yazar: Resul Daş, Yazarların ORCID numarası: [1]0000-0002-6113-4649, [2]0000-0002-1564-0604, [3] 0000-0002-6466-4696

Derin öğrenme insan beyninin işlevlerini taklit eden hesaplama sistemlerine dayanır. Derin öğrenmenin tarihi, 1943 yılında McCulloch ve Pitts tarafından düşünce sürecini taklit etmek için matematiğe ve sinir mantığı olarak adlandırılan algoritmalara dayalı bir hesaplama modeli geliştirmelerine dayanmaktadır [3]. 1958 yılında Rosenblatt iki katmanlı bir bilgisayar yapay sinir ağına dayalı, denetimli öğretimli bir desen tanıma algoritması geliştirmiştir [4]. 1965 yılında Ivakhnenko ve Lapa derin öğrenme algoritmalarını geliştirmeye yönelik olarak karmaşık denklemlerin aktivasyon fonksiyonlarına sahip modelleri kullanmışlardır [5]. 1988 yılında Fukushima el yazısı tanıma ve diğer desen tanıma sorunları için kullanılan hiyerarşik ve çok katmanlı yapay sinir ağı olan Neocognitron'ı önermiştir [6]. 1992 yılında Weng, Cohen ve Herniou karma sahnelerden otomatik olarak üç boyutlu nesne tanıma işlemini gerçekleştiren Cresceptron yöntemini yayımlamıştır [7]. 1995 yılında Cortes ve Vapnik tarafından benzer verilere sahip iki grubun sınıflandırılması için destek vektör ağları kullanılmıştır [8]. 1997 yılında Hochreiter ve Schmidhuber tarafından bilgiyi tekrarlayan geri yayılımla uzun süre boyunca saklamayı öğrenme için Uzun Kısa-Süreli Bellek (LTSM) adlı bir model önerilmiştir [9]. [10] "Derin Öğrenme" alanının popülerlik kazanmasını sağlayan en önemli çalışmalardan birisidir. 2012 yılında gerçekleştirilen bir çalışmada Google'ın araştırma ekibi tarafından tasarlanan ve 16000 işlemciden ve bir milyardan fazla bağlantıdan oluşan yapay desen tanıma algoritmalarının performansı insan düzeyine ulaşmıştır [11]. Facebook 2014 yılında fotoğraflarda kullanıcılarını otomatik olarak etiketlemek için 120 milyon parametreyi R-CNN katarak yüz tanıma görevlerini gerçekleştiren DeepFace adlı derin öğrenme teknolojisini kullanmıştır [12].

Derin öğrenme biyolojik sinir sistemlerinin bilgi işleme yöntemlerinden esinlenen yapay sinir ağları olarak bilinen algoritmaları kullanır. Böylece, bilgisayarların her bir verinin temsil ettiği şeyi tanımlamasına ve modelleri öğrenmesine olanak tanır [13]. Derin öğrenmede en popüler araçlardan birisi olan TensorFlow, makine öğrenimi ve derin sinir ağları araştırmalarını yürütmek amacıyla Google Beyin Ekibi'nde çalışan araştırmacılar ve mühendisler tarafından geliştirilmiştir. Açık kaynak kodlu yapay zekâ ve makine öğrenmesi kütüphanesi olan ve algılama, keşfetme, sınıflama, anlama ve öngörü uygulamalarında kullanılan TensorFlow, modeller oluşturmak için veri akış grafikleri kullanır ve yazılımcıların çok katmanlı ve geniş ölçekli yapay sinir ağları oluşturmalarına olanak tanır [14].

TensorFlow özellikle derin öğrenme için kullanılmakta olup [13] bu çalışmada da uygulanmaktadır. Ancak, literatürde bulunan ve derin öğrenme süreçleri için TensorFlow kullanan çalışmalardan farklı olarak bu çalışmada nesne tanıma ve takibi R-CNN kütüphanesinin kullanımı ele alınmıştır. Bu kütüphane ve TensorFlow kullanılarak durağan görüntüler, video görüntüleri ve webcam görüntüleri üzerinde nesne tanıma ve takibi işlemi gerçekleştirilebilecek bir uygulama geliştirilmiştir. Gerçekleştirilen uygulama aracılığıyla farklı görüntüler üzerinde testler gerçekleştirilmiş ve güçlü ve zayıf yönleri ortaya konmuştur.

2. Kullanılan Yöntemler

Nesne tespit ve tanıma süreçleri veri girişi, veri ön işleme ve aşamaları, öznitelik çıkarımı ve öz nitelik seçimi, tanımlama aşamaları altında farklı algoritmalar yardımıyla çözüm bulmaktadır. Veri girişi aşamasında hazırlanan veri gürültülerinden ayrıştırılmak, istenilen formata getirilmek gibi amaçlar sisteme girdi olarak verilir. Bir insan, otomobil veya ev istediğimiz veriye örnek iken, üzerinde insan bulunan bir orman fotoğrafı girdi olarak tanımlanabilir. Dijital fotoğraf makineleri ile çekilmiş fotoğraflar, dijital olarak oluşturulmuş resimler, videolar ve taranmış metinler veri türlerine örnektir. Veri formatı birçok etmene bağlı olmakla birlikte kullanacağımız algoritmalar için geçerli format matris haline getirilmiş görüntü pikselleridir [2]. Ön İşleme verilerin yapılacak olan işlemin amacına uygun hale gelmesi için hazırlamak veya engel teşkil etmesinin önüne geçmek için uygulanan yöntemlerdir. Ön işlem süreçleri veri, isterler, durum, ortam gibi birçok farklı etmene göre değişebilen genellikle önceden bilinemeyen ve deneme yanılma ile karar verilen yöntemlerdir. Gürültülü veriler ile eksik veya tutarsız veriler bulunması veri ön işlemenin nedenleri arasındadır. Veri ön işleme için birçok algoritma ve yöntem mevcuttur. Regresyon, eşikleme, kümeleme, filtreleme ve karar ağaçları veri ön işlemede kullanılan yöntemlerden bazılarıdır. Bazen tek başına bir yöntemi uygulamak yeterli olmayabilir, bu durumlarda birkaç farklı yöntem peş peşe kullanılmalıdır. Öznitelik, ön işleme tabi tutulmuş veri üzerinde daha önceden belirlenen nesnenin ve isterin elde edilmesidir. Bir cismin önceden tanımlanmış ölçütleri ve özellikleri sayesinde görüntüler üzerinden otomatik olarak tespit edilmesi ve detaylarının elde edilmesi işlemi öznitelik çıkarma olarak tanımlanabilir [15]. Tanımlanması gereken çok fazla unsur olmakla birlikte, bu unsurlar tek tek matematiksel olarak tanımlanabileceği gibi bir algoritmada yardımıyla da elde edilebilir [2]. Tanımlama, tespit edilen görüntüden çıkarım yapabilmek ve bu görüntünün ne olduğunu anlamak olarak tanımlanabilir. Bu noktada sınıflandırma ve kümeleme önem kazanmaktadır. Sınıflandırma, veriyi önceden belirlenmiş sınıflardan birine dâhil etmektir. Kümeleme ise benzer verileri, benzer özellik gösterenleri aynı grupta toplamadır [2]. Belirli

kategorileri temel alarak dijital görüntü veya video üzerinde nesne tanıma yapılırken, belirli bir nesneden yan yana veya birden çok olsa bile nesnenin tanıma işlemi gerçekleştirilebilmelidir. Nesne tanımayı normal sınıflandırma problemlerinden ayıran budur.

Nesne tespiti ile ilgili olarak literatürde farklı yöntemler bulunmaktadır. Bu yöntemler çerçeveler arasındaki fark, optik akış, arka plan modeli çıkartma olarak gruplanabilir. Bir görüntü veya videodaki nesnenin tespiti için iki temel bilgi kullanılır. Bunlar görsel öznitelik (renk, doku, şekil gibi) ve hareket bilgileridir. Bu bilgileri tek başına kullanan yöntemler olduğu gibi özellikle nesne takibi ve tespitinin zor olduğu durumlarda farklı öznitelikleri ve hareket bilgilerini birlikte kullanan yöntemler de bulunmaktadır [16]. Nesne tespiti için kullanılan bilgilerin çeşitliliği başarıyı arttırmakla beraber işlem zamanında bir artışa sebep olmaktadır. Bu durum gerçek zamanlı uygulamalarda nesne takibi için oldukça önemli bir problem oluşturmaktadır [17, 18]. Yayın kullanılan yöntemlerden birisi videolarda arka arkaya gelen iki görüntünün arasındaki geçici değişiklikleri bulmayı hedefler. Bir video içerisindeki arka arkaya gelen iki video çerçevesinin birbirinden çıkartılması ile bilgi elde etmeye dayanır [19]. Bu yöntem oldukça basit ve hızlı olmakla beraber ışık değişimi ve gürültüye karşı oldukça hassastır. Bir görüntüde nesne tespit işlemi için kullanılan geçici değişiklikler yönteminin hata oranı yüksek çıkabilmektedir. Bundan dolayı daha kullanışlı ve başarılı olan diğer yöntemlerden birisi ise belirlenmiş nesnenin olduğu görüntüden sadece arka plan görüntüsünün bulunduğu çerçevenin çıkarılmasıdır. Sonraki adımda ise elde edilen fark görüntüsü belli bir eşik değerden geçilir ve istenmeyen nesneler elenir [18, 20]. Bu yöntemle elde edilen yeni görüntüden gürültüleri temizlemek için morfolojik işlemler yapılır ve belirli oranlarda istenen nesne ortaya çıkarılmış olur [2].

Nesne takibi genel olarak bir video veya sıralı gelen görüntü dizisinde bulunan nesnenin takip edilmesidir. Nesne takibi yöntemleri temel olarak üç kategoriye ayrılır. Bunlar sırasıyla nokta tabanlı, çekirdek tabanlı ve silüet tabanlı yöntemlerdir. Nokta takip yönteminde takip edilecek nesne noktalar ile ifade edilir. Bu noktaların bir sonraki imgedeki konumları ve birbirlerine olan uzaklıkları gibi verilerin sonraki gelen video çerçevesinde de birbirine paralel olması beklenir. Bu bilgilerden yola çıkılarak nesne takibi sağlanır. Bu yöntemde temel amaç nesnenin video çerçevesi içinde tespit edilmesi ve bir önceki çerçevede kullanılan nokta benzerliklerin hesaplamasıdır. Bu yöntem deterministik ve istatistiksel yöntemler olarak kendi içinde alt sınıflandırmalar içermektedir [21]. Nokta takibi yöntemlerinde en yaygın kullanılan yöntemlerden birisi Kalman filtresidir. Bu yöntem nesnenin Gaussian dağılıma sahip durum değişkenleri yardımıyla videodaki bir sonraki gelen çerçevede nesne konumunu tahmin etmektedir. Kalman filtresi basit ve hızlı olma açısından gerçek zamanlı nesne takip uygulamalarında kullanıma uygundur [9, 10]. Durum değişkenleri Gaussian dağılımına sahip olmayan sistemlerde kalman filtresi başarısız olabilmektedir. Bu tür problemlerin giderilmesi için Parçacık filtresi yöntemi geliştirilmiştir. Parçacık filtresi olasılıksal yönteme dayanmaktadır. Bu yöntemin en büyük avantajı doğrusal olmayan ve çoklu dağılıma sahip sistemlerde çalışabilmesidir [22]. Çekirdek tabanlı yöntemlerde bir geometrik şekil yardımıyla takip edilecek nesne çerçevelenir. Bu çerçeve içerisinde bulunan nesne parçasının anlamlı bilgileri hesaplanarak başlangıçtaki şekil yardımıyla nesne takip edilir. Bu yöntemde nesnenin şeklinden ziyade kullanılan geometrik şeklin içerisinde bulunan nesne bilgilerinin çıkarılması yeterli olabilmektedir. Bu şekil içinde bulunan piksellerin hesaplanan olasılık yoğunluk bilgileri veya histogram özellikleri gibi bilgileri sonraki video çerçevelerinde takip edilebilmektedir. Silüet tabanlı yöntemler genellikle takip edilen nesnenin insan ya da hayvan gibi belli bir geometrik şekille ifade edilemediği durumlarda kullanılır. Bu yöntemin temel amacı nesneyi tanımlayacak kenar bilgisi ya da şekil bilgisi çıkartılarak sonraki imgelerde bu bilgiyi aramaktır. Bu yöntem şekil değişikliğine karşı oldukça hassas olmaktadır. Çekirdek ve siluet tabanlı yöntemler kıyaslandığında, çekirdek tabanlı yöntemlerin daha düşük işlem zamanına ve daha yüksek başarı oranlarına sahip oldukları görülmektedir. Bu sebepten dolayı çalışmalarda çekirdek tabanı yöntemler geniş bir kullanım alanına sahiptir. Nokta tabanlı yöntemler diğer yöntemlere oranla daha düşük işlem zamanına sahip olmakla birlikte daha düşük başarı oranına sahiptirler [2].

Görüntü işlemede sınıflandırma, bir veri kümesindeki belirli nesnelerin özelliklerine göre gruplara ayrılması şeklinde açıklanabilmektedir. Bu ayrıştırma yapılırken nesnenin kendine has öznitelikleri kullanılmaktadır. Kullanılan öznitelik nesnenin tanımlanması için kullanılan sayısal değerlerden oluşmaktadır. Bu değerler bir pikselin değeri veya bir görüntüdeki ortalama yoğunluk değeri gibi anlamlı ifadeler olabilmektedir. Nesne takibi veya sınıflandırması yöntemlerinde öznitelik seçimi ve kullanımı performansı doğrudan etkileyen işlemlerdir. Bir sınıflandırma veya takip algoritmasında kullanılan öznitelikler seçilirken başarıyı artırmasının yanı sıra getirdiği işlem yüküne de bakılması gerekmektedir. Gerçek zamanlı uygulamalarda daha basit öznitelikler kullanılırken zaman problemi olmayan uygulamalarda ise çok daha farklı ve zor öznitelikler kullanılmaktadır [22]. Sınıflandırma algoritmaları nesne takip yöntemlerinde önemli bir yere sahiptir. Özelikle çoklu nesne takip yöntemlerinde takip edilen her bir nesnenin diğer nesneler ile karıştırılmaması için doğru bir şekilde sınıflandırılması oldukça önemli olmaktadır. Ayrıca bazı nesne takip yöntemlerinde sınıflandırma algoritmaları

kullanılarak arka plan ve hedef nesne iki sınıfta tanımlanarak nesne takibi yapılmaktadır [16, 23]. Hareket tabanlı sınıflandırma yönteminde, nesne hareket bilgileri kullanılarak sınıflandırma yapılmaktadır. Nesne takip çalışmalarında hareket öznitelikleri kullanılarak, bir sonraki gelecek video çerçevesinde nesnenin potansiyel olarak varacağı konum tahmin edilmektedir. Bu ön tahmin nesnenin konumunu bulmak için aranacak alanı da azaltmaktadır. Bu yöntemlerde genellikle nesnenin belli bir ortalama hızla hareket edildiği varsayılmaktadır [24]. Belli bir şekli olmayan nesnelerin takibinde kenar, şekil ve renk bilgileri seçici olmadığından, hareket tabanlı sınıflandırıcıların başarı oranları yüksek olabilmektedir. Optik akış özniteliği bu başlık altında ayrı bir öneme sahip olmaktadır. Bu yöntem ön eğitim yapılmadan nesne takibi üzerinde yapılan çalışmalarda önemlidir [17]. Optik akış yöntemi hareket tabanlı sınıflandırmada çoklu nesne takibi yapılan çalışmalarda nesne tespitini kolaylaştırmaktadır [24]. Renk ve doku tabanlı sınıflandırmalar çok farklı alanlarda kullanılabilmektedir. Doku tabanlı sınıflandırma yardımıyla nesne sınıflandırma işlemlerinde nesnenin yapısına ait öznitelikler kullanılmaktadır. Özniteliklerin çıkartılması ve kullanılması işlem maliyetini yükseltir. Bununla birlikte yüksek başarılar elde edilmektedir. Renk tabanlı sınıflandırma yöntemleri en yaygın kullanılan yöntemlerdir. Renk özniteliği kullanımı işlem zamanın azaltır. Bununla birlikte gürültü ve ışık değişimi içermeyen videolarda başarı oranını yükseltir [24].

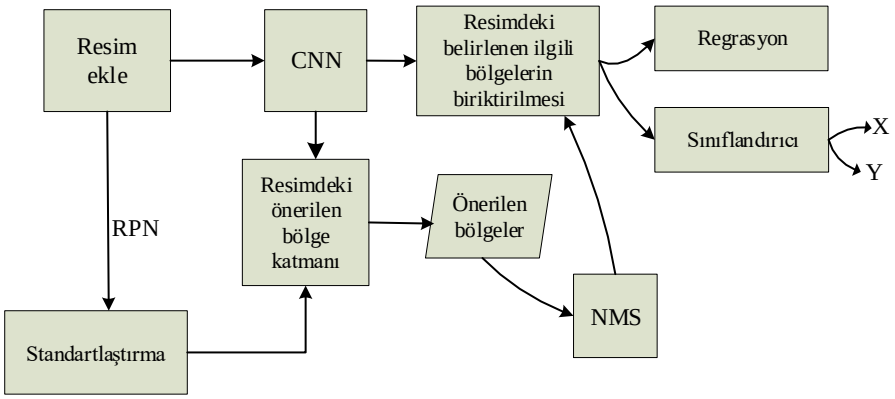
2.1. Nesne bulma ve tanımada kullanılan yaklaşımlar

Son yıllarda nesne tanıma ve tespiti için R-CNN, Fast R-CNN, Faster R-CNN, Mask R-CNN ve SSD kütüphaneleri yaygın olarak kullanılmaktadır. R-CNN ilgili kütüphanenin bölge önerisi yaklaşımını kullanan en temel modelidir. Öte yandan, R-CNN kütüphanesinin Fast R-CNN, Faster R-CNN ve Mask R-CNN gibi geliştirilmiş modelleri de bulunmaktadır. Bu modeller içerisinde genel başarımı itibariyle en iyi model Faster R-CNN'dir. Farklı boyutlarda bölge önerisi yapan modelde ilgili pencereler konvansiyonel sinir ağlarından geçirilerek pencerelerin boyutları eşitlenmektedir. Sinir ağının sonucunda o bölgede sınıflandırma yapmak için Destek Vektör Makineleri (SVM) sınıflandırıcısı kullanılmaktadır. Bu gözetimli öğrenmede kullanılan bir makine öğretimi yöntemidir. R-CNN bölgede bir nesne arar ve nesneyi bulduğunda bu nesnenin sınıfını döndürür. Aynı zamanda nesnenin resimdeki yerini gösteren 4 tane değer verir. Bu değerler dikdörtgenin sol üst köşesinin X ve y koordinatları genişlik ve uzunluk değerleridir [25]. Faster R-CNN'de SVM kullanılmamaktadır. Bağımsız değişkenler ile bir bağımlı değişken arasındaki bağlantıyı modellemek için kullanılan doğrusal regresyon ile nesnenin sınırları belirlenmektedir. Bu şekilde etrafına dikdörtgen çizebilmektedir. Doğrusal regresyon ile nesnenin bulunduğu yeri tam olarak tespit ederek nesnenin etrafında çizilecek olan dikdörtgenin tam oturacak şekilde ayarlanmalıdır. İlgi bölgesi ile belirlenen bölgelerde nesne bulunursa ve bu nesnenin bir parçası alanın dışında ise sonuç olarak çizilen dikdörtgen ilgi bölgesinin dışına çıkabilmektedir.

Fast R-CNN, R-CNN modelinin bir türevi olup R-CNN modelinin yavaşlığının giderilmesi için geliştirilmiştir. R-CNN oldukça yavaş çalışan bir modeldir ve gerçek zamanlı nesne tanıma uygulamalarında kullanılması mümkün değildir. Fast R-CNN, R-CNN'e benzemekle birlikte hızlandırma için farklı teknikler kullanmaktadır. Resimlere bölge önerisi yapmak yerine görüntüleri doğrudan konvonsiyonel sinir ağından geçirir ve daha sonra orjinal resime uyan yüksek çözünürlüklü bir özellik haritası çıkartır. Özellik haritasında seçici arama ile bölge önerisi çıkartılarak, birçok bölge önerisi oluşturmak yerine aynısı özellik haritasında yapılmaktadır. Böylece her bölgeyi ayrı ayrı konvansiyonel sinir ağından geçirmek zorunda kalınmamaktadır ve ilgili görüntü sadece bir defa konvansiyonel sinir ağından geçirilmektedir. Sonra önerilen bölgeler alınıp boyut eşitleme işlemi yapılmaktadır. Fast R-CNN ile sınıflandırma yapmak için softmax layer kullanılmaktadır. Sınıflandırma yapabilmesi için yeni bir model oluşturmak yerine sinir ağını genişletip sinir ağı içerisinde sınıflandırma yapılmaktadır. Fast R-CNN tek bir zayıf yönü vardır. Zamanın çoğunu test aşamasında bölge önerisi yapmak için harcamaktadır. Bölge önerilerinin daha hızlı bir şekilde gerçekleştirebilme olasılığının bu modele ciddi kazanç sağlayacağı fikrinden yola çıkılarak Faster R-CNN geliştirilmiştir.

Faster R-CNN RCNN'de görülen yavaşlık problemini tam olarak çözmektedir. Seçici arama ile bölge önerisi almak yerine bu önerileri ağ içerisinde yaparak hız kazanabilmektedir. Faster R-CNN'de giriş görüntüsü konvansiyonel sinir ağlarından geçirip bir özellik haritası çıkartılmaktadır. Daha sonra bölge önerisi ağı oluşturulup ağ bölgeleri belirlendikten sonra Fast R-CNN ile aynı işlemler gerçekleştirilmektedir. Belirlenen bölgeler alınıp yeniden şekillendirdikten sonra sınıflandırma yapılmaktadır. Eğitilmesi gereken 4 farklı parametre vardır. Hem bölge önerisi veren ağ eğitilecek hem de normal konvansiyonel işlemlerin yapıldığı ağ eğitilecektir. Bu eğitimlerin dengesini tutturmak biraz zor olabilmektedir [25]. Bölge önerisi ağının iki tane görevi vardır; her bir öneri için nesne var mı yok mu karar verilmesi ve önerilerin pencere büyüklüğünün belirlenmesi. Daha sonra asıl sinir ağı sınıflandırma işlemi gerçekleştirilerek bakılan bölge içerisinde nesne var mı yok mu tespit edilecektir.

En sonuda bulunan nesnenin sınırları belirlenecektir [25]. Şekil 1'de Faster R-CNN'in akış diyagramı gösterilmektedir.



Şekil 1. Faster R-CNN modelinin akış diyagramı.

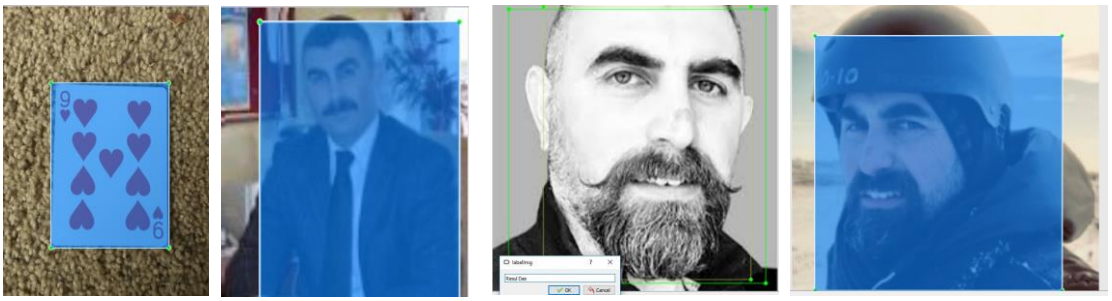
Mask R-CNN modelinde hem nesnenin bulunduğu yere dikdörtgen çizilip hem de bu nesnenin resimde kapladığı tüm pikseller tespit edilmektedir. Resimde nesnenin olduğu tüm pikselleri belirleyerek görselleştirmek için o kısımda maskeleme işlemi yapılmaktadır. Nesne tanımak için Faster R-CNN kullanılırak bulunan nesneler maskelenebilmektedir. Mask R-CNN temel olarak Faster R-CNN kullanmaktadır. Farklı olarak Mask R-CNN'de maskeleme yapmak için eklenen kısım ikilik sayı sisteminde olacaktır. Belirlenen pikselde maske olacak mı olmayacak mı kararını veren bir mekanizma kullanılmaktadır. Maskeleme yapılacak bu ağda girdi olarak Faster R-CNN'den gelen özellik haritasını alınmaktadır. Çıktı olarak ise 1'ler ve 0'lardan oluşan bir matris elde edilmektedir. Bu matriste maskeleme olması gereken yerlerde 1, maskeleme olmaması gereken yerlerde 0 vardır.

Faster R-CNN'den daha hızlı çalışan bir model olan SSD tek seferde nesne tanıma işlemi yapmaktadır. R-CNN ile bölge önerisini iki farklı aşamada yapılmakta iken ilk önce resimde nesne olması beklenen bölgeler belirlenmektedir. Daha sonra tam bağlı katman ile sınıflandırma yapılmaktadır. SSD bu ikisini tek seferde gerçekleştirmektedir. Diğer modellerde olduğu gibi her zamanki gibi ilk öncelikle girdi olarak bir resim alınmaktadır. Resim konvansiyonel sinir ağından geçirilmektedir. Böylece farklı boyutlarda özellik haritaları çıkmaktadır. Tüm özellik haritalarında 3x3 konvansiyonel filtre yardımıyla az miktarda sınırlayıcı dikdörtgenler elde edilmektedir. Her dikdörtgen için aynı zamanda hem sınırlar hem sınıflandırmalar belirlenmektedir. Bu dikdörtgenler her aktivasyon haritasında olduğu için hem küçük nesneleri hem büyük nesneleri tanıyabilmektedir.

3. Geliştirilen Nesne Tanıma Uygulaması

Geliştirilen uygulama Python programlama dili platformunda geliştirilmiştir. Bu bölümde geliştirilen uygulamanın ve yapılan testlerin geliştirilme aşamaları adım adım aşağıda verilmektedir.

Adım 1: Resim toplama ve etiketleme: Belirli nesnelerde eğitebilmek için öncelikle bir dataset oluşturulmalıdır. Dataset bünyesinde birkaç yüz adet resim bulunması ve bu resimlerin farklı ortamlarda ve farklı açılardan çekilmiş olması modelin daha iyi eğitilmesini sağlayabilecektir. Nesnenin bazı yerleri kapatılmış şekilde ve farklı ışıklandırmalarda fotoğraflar çekilmelidir [14]. Çok büyük resimlerin eğitimi uzun süreceği için resimlerin boyutu küçültülmelidir. Resimler topladıktan sonra train ve test klasörlerine paylaştırılmalıdır. Resimler uygun klasörlere atıldıktan sonra hepsinin tek tek etiketlenmesi gerekmektedir. Labellmg ile tüm resimlerdeki tanıtılmak istenilen nesneler tek tek seçerek etiketleme işlemi yapılmalıdır [14]. Her etiketlenen resim için .xml dosyası oluşturulacaktır. Bu .xml dosyaları tfrecords'a çevirilerek eğitim yapılacaktır. Test ve train klasörleri içerisindeki her bir resim için bir tane .xml dosyası olması gerekmektedir. Şekil 2'de gösterildiği gibi LabelImg programıyla xml dosyaları çıkarılan resimler tek tek işaretlenip isimleri atanmalıdır.



Şekil 2. Resimlerin LabelImg ile etiketlenmesi.

Adım 2: Eğitim verisi oluşturma: Her resim etiketlendikten sonra TFRecord oluşturmak gerekmektedir. Öncelikle etiket bilgisini barındıran .xml dosyalarını .csv dosyasına çevirilmelidir. .csv dosyaları oluşturulduktan sonra generate_tfrecord.py dosyası herhangi bir metin editörü ile açılıp sınıf bilgilerinin girilmesi gerekmektedir. Kaç tane sınıf varsa o kadar id bilgisi girilmelidir [13]. Geliştirilen uygulama için “kedi”, “kopek”, “dokuz”, “on”, “uc”, “as” ve “resuldas” etiketleri oluşturuldu.

Adım 3: Label map oluşturma: Label map modele hangi ID’de hangi sınıf var onu belirtir. Object detection training klasörü içiresinde yeni bir dosya oluşturulmalıdır. Oluşturulan dosyaya "labelmap.pbtxt" ismi verilmeli ve dosya bir metin editörü açılıp içine aşağıdaki şekilde sınıf ve ID bilgileri girilmelidir [13]. Adım 2’de belirlenen her etiket için bir ID numarası verilmiştir.

Adım 3: Eğitim ayarlarının yapılması: Bu aşamada, programlama platformu üzerinde hangi model ve hangi parametrelerin kullanılacağı belirlenmelidir. Önce, çalışmak istenilen modelin yolu verilmelidir. Sonra, oluşturulan record ve labelmap dosyasının yolu verilmelidir. Daha sonra, kaç tane test için resim varsa sayısının yazılması gerekmektedir. Son aşamada ise oluşturulan test record ve label map dosya yollarının belirtilmelidir.

Adım 4: Eğitimin gerçekleştirilmesi: Konsoldan TensorFlow aktif edildikten sonra Phyton path ayarı gerekmektedir. İşlem sonrasında her adımda ekrana loss değeri yazdırılacaktır. Loss değeri tahmin edilen değerin gerçek değerden ne kadar uzak olduğunu belirtmektedir. Eğitim esnasında zamanla sıfır yaklaşması beklenir [13].

Adım 5: Nesne tespiti: Bu adımda resim, video ve webcam verileri üzerinde nesne tanıma uygulaması gerçekleştirilmektedir. Bu aşamada aşağıdaki işlem adımları tek tek uygulanmıştır.

- Resim tanıma işleminde, öncelikle os, cv2, numpy, sys ve tensorflow paketleri program içine aktarılmalıdır.
- Bulunan dizine göre göreceli olarak üstte bulunan dizin için bir yol tanımı (path) oluşturulmalıdır.
- Daha sonra ihtiyaç duyulan yardımcı programlar içe aktarılmalıdır. Kullanılan nesne algılama modülünü içeren dizinin adı girilmelidir.
- Geçerli çalışma dizinine giden yol tutulmalıdır.
- Nesne tespiti için kullanılan modeli içeren donmuş algılama grafiği .pb dosyasının yolu belirtilmelidir.
- Harita dosyasını etiketleme yolu belirtilmelidir.
- Resmin yolu belirtilmelidir.
- Nesne algılayıcısının tanımlayabileceği sınıf sayısı belirtilmelidir. Etiket haritası yüklenmelidir. Label harita indekslerini kategori isimlerine göre eşleştirir.
- TensorFlow modeli belleğe yüklenmelidir.
- Nesne algılama sınıflandırıcısı için giriş ve çıkış tensörleri (veri) tanımlanmalıdır. Giriş tensörü görüntüdür. Çıktı tensörleri tespit kutuları, skorlar ve sınıflardır. Her kutu, görüntünün belirli bir nesnenin algılandığı bölümünü gösterir.
- Skor, nesnelerin her biri için güven düzeyini temsil eder. Skor, sonuç etiketinde, sınıf etiketiyle birlikte gösterilir.
- Algılanan nesne sayısı tespit edilmelidir. OpenCV kullanılarak görüntüyü yüklenir ve şekli olacak şekilde resim boyutlarını genişletilir. Sütundaki her bir öğenin piksel RGB değerine sahip olduğu tek bir sütun dizisi ifade edilmektedir.

- Giriş olarak model görüntüyle çalıştırılarak gerçek algılama işlemini gerçekleştirilir.
- Tüm sonuçlar görsel olarak elde edilir ve görüntüyü göster komutu girilir. Resmi kapatmak için herhangi bir tuşa basılır.

Resim tanıma işleminde olduğu gibi video verisinde nesne tanıma işleminde de öncelikle paketler içe aktarılmalıdır. Video verisinde nesne algılaması gerçekleştirmek için bir TensorFlow tarafından eğitilmiş sınıflandırıcı kullanır. Sınıflandırıcı videoda nesne algılamak için kullanır ve her karedeki ilgi nesnelerinin çevresine kutular çizer ve skorları belirtir.

Video verisinden nesne tanıma işlemi için aşağıdaki işlem adımları izlenmiştir.

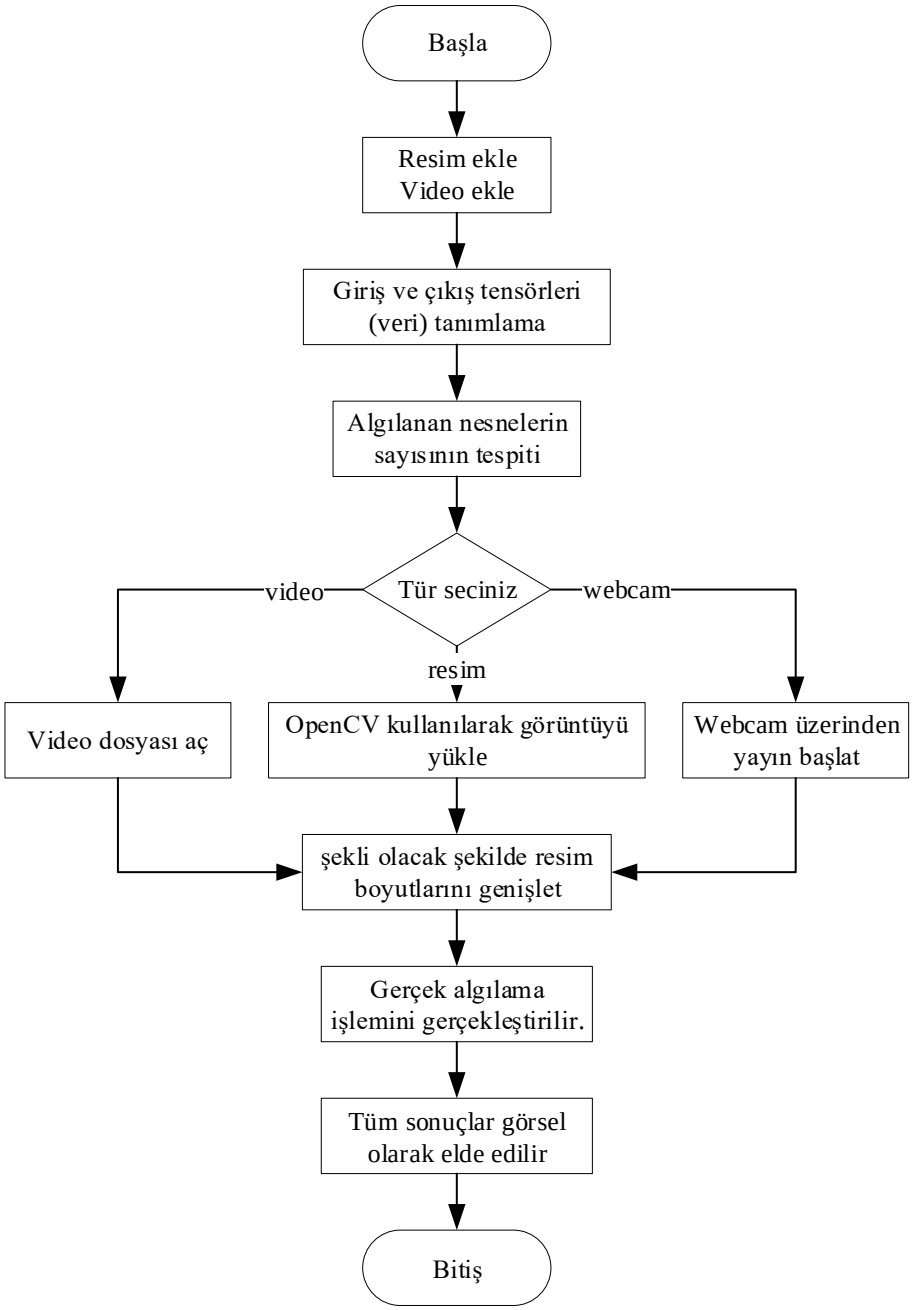
- Bulunan dizine göre göreceli olarak üstte bulunan dizin için bir path oluşturulmalıdır.
- Yardımcı programlar içe aktarılmalıdır.
- Kullanılan nesne algılama modülünü içeren dizinin adı girilmelidir.
- Geçerli çalışma dizinine giden yol tutulmalıdır.
- Nesne tespiti için kullanılan modeli içeren donmuş algılama grafiği .pb dosyasının yolu belirtilmelidir.
- Harita dosyasını etiketleme yolu belirtilmelidir.
- Videonun yolu belirtilmelidir.
- Nesne algılayıcısının tanımlayabileceği sınıf sayısı belirtilmelidir.
- Etiket haritası yüklenmelidir. Label harita indekslerini kategori isimlerine göre eşleştirir.
- TensorFlow modeli belleğe yüklenmelidir.
- Nesne algılama sınıflandırıcısı için giriş ve çıkış tensörleri (veri) tanımlanmalıdır. Giriş tensörü görüntüdür.
- Çıktı tensörleri tespit kutuları, skorlar ve sınıflardır. Her kutu, görüntünün belirli bir nesnenin algılandığı bölümünü gösterir.
- Skor, nesnelerin her biri için güven düzeyini temsil eder. Skor, sonuç etiketinde, sınıf etiketiyle birlikte gösterilir.
- Algılanan nesne sayısı tespit edilmelidir.
- Video dosyası açılmalıdır.
- Çerçeve alınır ve şekillendirilmek için boyutları genişletilir. Sütundaki her bir öğenin piksel RGB değerine sahip olduğu tek bir sütun dizisidir.
- Giriş olarak model görüntüyle çalıştırılarak gerçek algılama işlemini gerçekleştirilir.
- Tüm sonuçlar görsel olarak elde edilir ve görüntüyü göster komutu girilir.

Video tanıma işleminde olduğu gibi webcam verisinde nesne tanıma işleminde de öncelikle paketler içe aktarılmalıdır. Webcam verisinde nesne algılaması gerçekleştirmek için bir TensorFlow tarafından eğitilmiş sınıflandırıcı kullanır. Sınıflandırıcı webcam verisionde nesne algılamak için kullanır ve her karedeki ilgi nesnelerinin çevresine kutular çizer ve skorları belirtir.

- Bulunan dizine göre göreceli olarak üstte bulunan dizin için bir path oluşturulmalıdır.
- Yardımcı programlar içe aktarılmalıdır.
- Kullanılan nesne algılama modülünü içeren dizinin adı girilmelidir. Geçerli çalışma dizinine giden yol tutulmalıdır.
- Nesne tespiti için kullanılan modeli içeren donmuş algılama grafiği .pb dosyasının yolu belirtilmelidir.
- Harita dosyasını etiketleme yolu belirtilmelidir.
- Videonun yolu belirtilmelidir.
- Nesne algılayıcısının tanımlayabileceği sınıf sayısı belirtilmelidir.
- Etiket haritası yüklenmelidir. Label harita indekslerini kategori isimlerine göre eşleştirir.
- TensorFlow modeli belleğe yüklenmelidir.
- Nesne algılama sınıflandırıcısı için giriş ve çıkış tensörleri (veri) tanımlanmalıdır. Giriş tensörü görüntüdür.
- Çıktı tensörleri tespit kutuları, skorlar ve sınıflardır. Her kutu, görüntünün belirli bir nesnenin algılandığı bölümünü gösterir.
- Skor, nesnelerin her biri için güven düzeyini temsil eder. Skor, sonuç etiketinde, sınıf etiketiyle birlikte gösterilir.
- Algılanan nesne sayısı tespit edilmelidir.
- Webcam üzerinden yayın başlatılır. Çerçeve alınır ve şekillendirilmek için boyutları genişletilir. Sütundaki her bir öğenin piksel RGB değerine sahip olduğu tek bir sütun dizisidir.

- Giriş olarak model görüntüyle çalıştırılarak gerçek algılama işlemini gerçekleştirilir.
- Tüm sonuçlar görsel olarak elde edilir ve görüntüyü göster komutu girilir.

Şekil 3'de resim ya da video içerisinde nesne tanıma işlemlerinin akış diyagramı verilmiştir.



Şekil 3. Nesne tanıma işlemlerinin akış diyagramı.

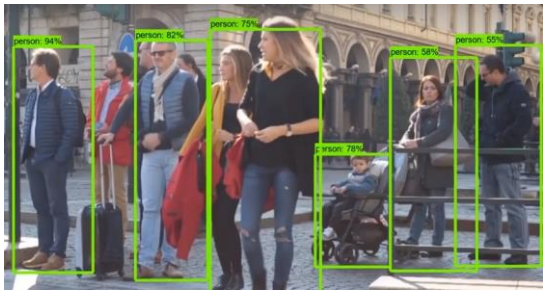
4. Uygulama Sonuçları ve Değerlendirilmesi

Resim veya görüntü üzerinden nesne tanıma veya yüz tanıma üzerine birçok farklı yaklaşımlar ve uygulamalar mevcuttur. Geliştirilen bu yaklaşımlarda izlenilen metotlar farklı olduğu gibi, literaürdeki son yaklaşımlara göre

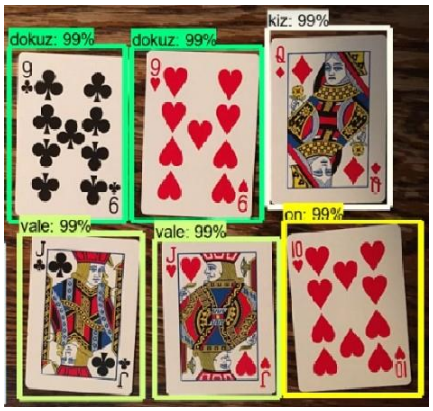
başarım oranları düşüktür [26]. Ancak derin öğrenme ile geliştirmiş olduğumuz bu yaklaşımın başarım oranı çok daha yüksek olmuştur.

3. bölümde detayları açıklanan uygulama Faster R-CNN modeliyle resim, video, webcam verileri üzerinde kullanılarak nesne tanıma işlemi yapılmıştır. Öncelikle object detection hazır eğitilmiş kütüphanesi kullanılarak video, resim ve webcam verileri için nesne algılama yapılmıştır. Şekil 5'de video görüntüsü üzerinden ve Şekil 6'da ise webcam video verileri üzerinde nesne tespiti ve tanıma işlemi gösterilmektedir. Şekil 7'de iskambil kartları üzerinde gerçekleştirilen nesne tanıma işlemi gösterilmektedir. Kartların her yönden birçok fotoğrafı çekilmiş olup test ve train adında ik farklı dosyada kaydedilmiştir. Bu resimlerin test kısmında olanları eğitim için kullanılmıştır. Resimler tek tek LabelImg ile etiketlenip xml dosyaları oluşturulmuştur. Oluşturulan sınıf kadar her resime id atanmıştır. Bu id lere sınıf isimleri atanmıştır. Yeterli düzeyde eğitim gerçekleştikten sonra geliştirilen model iskambil kartlarını tanımayı öğrenmiştir. Şekil 8'de insan için nesne tanıma işlemi yapılmıştır. Bunun için her ortamda çekilen farklı ışık tonlarında fotoğrafları toplanıp test ve train klasörü içinde toplanmıştır. Bunların xml ve csv dosyaları oluşturulmuştur. Eğitim gerçekleştikten sonra Faster R-CNN modelinde %98 başarı oranı yakalanmıştır.

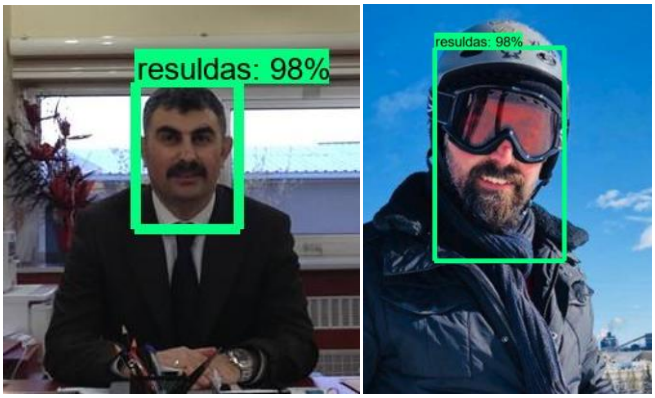
Şekil 5. Video verileri üzerinde nesne tespiti.



Şekil 6. Webcam video verileri üzerinde nesne tespiti.



Şekil 7. Kartlarla geliştirilen nesne modeli.



Şekil 8. Resim üzerinden kişi tanıma.

Geliştirilen uygulamada yapılan testlerde görüldüğü üzere, nesne tespiti ve tanıma işleminde başarı oranının daha yüksek olması için Faster R-CNN kullanılmalıdır. Ancak, Faster R-CNN benzerlerine kıyasla daha güçlü bir donanıma ihtiyaç duymaktadır ve Single Shot Multibox Detactor (SSD)'ye göre daha yavaştır. Ayrıca, SSD sadece tek ileri gidiş yaptığı için düşük donanımlı cihazlarda bile hızlı bir şekilde nesne tanıma yapabilmektedir. Ancak SSD ile çok isabetli modeller oluşturulamamaktadır. Bu nedenle, sadece hız önemli ise ve gerçek zamanlı nesne tespiti ve tanıma işlemi gerçekleştirilecekse SSD tercih edilmelidir.

5. Sonuç

Derin öğrenme, makine öğreniminin birçok pratik uygulamasını ve yapay zekâ alanının genişletilmesini sağlamıştır. Görevlere özgü algoritmaların aksine, öğrenme verilerini temsil etmeye dayanan derin öğrenme, görüntü işleme alanında oldukça başarılı sonuçlar alınmasını ve karmaşık görüntü işleme problemlerinin kolaylıkla çözüme kavuşturulabilmesini sağlamaktadır. Derin öğrenme yöntemlerinin eğitim süreleri uzun olmasına rağmen test aşamasında elde edilen başarı oranları derin öğrenme yöntemlerine olan güveni arttırmıştır.

Bu çalışmada derin öğrenme ile hareketli nesne tanıma ve takibi için Google'ın açık kaynak kodlu kütüphanesi olan TensorFlow kullanılmıştır. Nesne takibi için Region Based Convolutional Networks kütüphanelerinden Faster R-CNN modeli ele alınmıştır. Bu kütüphaneler ile durağan görüntüler, video görüntüleri ve webcam görüntüleri üzerinde nesne tanıma işlemi gerçekleştirilmiş ve incelenen kütüphanelerin güçlü ve zayıf yönleri ortaya konmuştur. Çalışmada elde edilen sonuçlar, basit görüntü işleme problemlerinde işlem zamanı ve hesaplama karmaşıklığı az olan algoritmaların kullanılmasının daha uygun olabileceğini göstermiştir. Ayrıca, literatür taraması sonuçları Kalman Filtresi ve Parçacık Filtresi gibi yöntemlerinin nesne takibinde oldukça başarılı sonuçlar elde edilmesini sağladığını göstermiştir.

Kaynaklar

[1] Viola P. and Jones M., Rapid object detection using a boosted cascade of simple features, in Computer Vision and Pattern Recognition, CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on, vol. 1. IEEE, 2001.
[2] Özbaysar E. And Borandağ E., Vehicle plate tracking system, in 2018 26th Signal Processing and Communications Applications Conference (SIU). IEEE, 2018, pp. 1–4.
[3] McCulloch WS. And Pitts W., A logical calculus of the ideas immanent in nervous activity, The bulletin of mathematical biophysics, 1943, 5(4): 115–133.
[4] Haykin SS., Neural networks and learning machines. 3, Pearson Upper Saddle River, 2009.
[5] Ivakhnenko AG. and. Lapa VG., Cybernetic predicting devices. CCM Information Corporation, 1965.
[6] Fukushima K., Neocognitron: A hierarchical neural network capable of visual pattern recognition. Neural networks, 1988, 1(2): 119–130.
[7] Weng J., Cohen P., and Herniou M., Camera calibration with distortion models and accuracy evaluation, IEEE Transactions on Pattern Analysis & Machine Intelligence, 1992, no. 10, pp. 965–980.
[8] Cortes C. and Vapnik V., Support-vector networks, Machine learning,1995, 20(3): 273–297,
[9] Hochreiter S. and Schmidhuber J., Long short-term memory, Neural computation, 1997, 9(8): 735–1780.
[10] Hinton GE., Srivastava N., Krizhevsky A., Sutskever I., and Salakhutdinov RR., Improving neural networks by preventing co-adaptation of feature detectors. july 3, 2012, URL http://arxiv.org/abs/1207.0580, 2016.
[11] Lohr S., The age of big data, New York Times, 2012, 11(2012).

[12] Taigman Y., Yang M., Ranzato M., and Wolf L., Deepface: Closing the gap to human-level performance in face verification, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2014, pp. 1701–1708.
[13] Lee K. and Son M., Deepspotcloud: leveraging cross-region gpu spot instances for deep learning, in Cloud Computing (CLOUD), 2017 IEEE 10th International Conference on. IEEE, 2017, pp. 98–105.
[14] Tokui S., Oono K., Hido S., and Clayton J., Chainer: a next-generation open source framework for deep learning, in Proceedings of workshop on machine learning systems (LearningSys) in the twenty-ninth annual conference on neural information processing systems (NIPS), vol. 5, 2015, pp. 1–6.
[15] Şahin Ö., Kurtoğlu A., and Ercan G., Computer science terminology extraction from parallel corpora, in 2018 26th Signal Processing and Communications Applications Conference (SIU). IEEE, 2018, pp.1–4.
[16] Luo W., Xing J., Milan A., Zhang X., Liu W., Zhao X., and Kim T.-K., Multiple object tracking: A literature review, arXiv preprint arXiv:1409.7618, 2014.
[17] Chen Y., Yang X., Zhong B., Pan S., Chen D., and Zhang H., Cnntracker: Online discriminative object tracking via deep convolutional neural network, Applied Soft Computing, 2016, 38, pp. 1088–1098.
[18] Hardas A., Bade D., and Wali V., Moving object detection using background subtraction shadow removal and post processing, in International Journal of Computer Applications (0975–8887) International Conference on Computer Technology (ICCT 2015), 2015.
[19] Shaikh S.H., Saeed K., and Chaki N., Moving object detection approaches, challenges and object tracking, in Moving Object Detection Using Background Subtraction. Springer, 2014, pp. 5–14.
[20] Aldhaheri A.R. and Edirisinghe E.A., Detection and classification of a moving object in a video stream, in Proc. of the Intl. Conf. on Advances in Computing and Information Technology-ACIT, 2014.
[21] Yilmaz A., Javed O., and Shah M., Object tracking: A survey, Acm computing surveys (CSUR), 2006, 38(4): p. 13.
[22] Trier O.D., Jain A.K., Taxt T. et al., Feature extraction methods for character recognition-a survey, Pattern recognition, 1996, 29(4): pp. 641–662.
[23] Avidan S., Support vector tracking, IEEE transactions on pattern analysis and machine intelligence, 2004, 26(8): 1064-1072.
[24] Fan L., Wang Z., Cail B., Tao C., Zhang Z., Wang Y., Li S., Huang F., Fu S., and Zhang F., A survey on multiple object tracking algorithm, in Information and Automation (ICIA), 2016 IEEE International Conference on. IEEE, 2016, pp. 1855–1862.
[25] Redmon J., Divvala S., Girshick R., and Farhadi A., You only look once: Unified, real-time object detection, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp.779–788.
[26] Baykara M and Daş R, Real time face recognition and tracking system, 2013 International Conference on Electronics, Computer and Computation (ICECCO), Ankara, 2013, pp. 159-163. doi: 10.1109/ICECCO.2013.6718253