İZMİR KÂTİP ÇELEBİ ÜNİVERSİTESİ
2010
FEN BİLİMLERİ ENSTİTÜSÜ

Çocuk Resimlerindeki Nesnelerin Derin Öğrenme Yöntemleriyle Tespit Edilmesi

Yazılım Mühendisliği Ana Bilim Dalı

Yüksek Lisans Tezi

Betül GÜLTER

Tez Danışmanı: Prof. Dr. Ayşegül ALAYBEYOĞLU SOY

Ocak 2023

İzmir Kâtip Çelebi Üniversitesi Fen Bilimleri Enstitüsü öğrencisi **Betül Gülter** tarafından hazırlanan **Çocuk Resimlerindeki Nesnelerin Derin Öğrenme Yöntemleriyle Tespit Edilmesi** başlıklı bu çalışma tarafımızca okunmuş olup, yapılan savunma sınavı sonucunda kapsam ve nitelik açısından başarılı bulunarak jürimiz tarafından YÜKSEK LİSANS TEZİ olarak kabul edilmiştir.

ONAYLAYANLAR:

Tez Danışmanı: **Prof. Dr. Ayşegül ALABEYOĞLU SOY**
İzmir Kâtip Çelebi Üniversitesi

Tez Eş-danışmanı: **Doç. Dr. Aytuğ ONAN**
İzmir Kâtip Çelebi Üniversitesi

Çocuk Resimlerindeki Nesnelerin Derin Öğrenme Yöntemleriyle Tespit Edilmesi

Öz

Görüntü işleme yöntemleri kullanılarak durağan ya da hareketli 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. Son yıllarda yapay zekâ tabanlı bileşenlerle güçlendirilen bu sistemler hem daha hızlı hem de daha kesin hedef tespiti yapmayı sağlamaktadır. 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 açık kaynak kodlu görüntü işleme kütüphanesi olan OpenCV(Open Source Computer Vision) kullanılmıştır. Kademeli sınıflandırıcı modelleri eğitmek, test etmek ve geliştirmek için kullanılan Cascade Trainer GUI'den yararlanarak haarcascade oluşturulmuştur. Bu kütüphaneler ile durağan görüntüler, video görüntüleri ve webcam görüntüleri üzerinde nesne tanıma işlemi ile çocukların çizmiş olduğu kaktüs resimlerinin tanınması hedeflenmiştir.

Anahtar Sözcükler: Yapay zeka, Derin Öğrenme, Object Detection, OpenCV, Cascade Trainger GUI, Çocuk Resimleri, Adaboost.

Teşekkür

Tez çalışmasında veri seti olarak kullanacağım çocuk resimlerini toplamamdaki katkılarından dolayı ilkokul öğretmenlerim Fatma BÜLBÜL ve Şükran KUL' a , rehabilitasyon merkezindeki öğrencilerinin çizimlerini benimle paylaşan yengem Züleyha'ya ve Rota Koleji'ne teşekkür ve saygılarımı sunarım.

İçindekiler

Şekiller Listesi

Bölüm 1

Giriş

Nesne tespiti ve nesne tanıma dijital görüntü işleme uygulamalarının vazgeçilmez unsurlarındandır. Nesne tespiti ve nesne tanıma konusu üzerine uzun yıllar çalışılmış, farklı algoritmalar ve yöntemler geliştirilmiştir. Dijital görüntülerdeki nesnelerin hızlı tespitini etkin biçimde gerçekleştiren ilk algoritma Viola Jones algoritmasıdır. Son yıllarda grafik işleme birimlerindeki gelişmeler ve derin öğrenme sayesinde daha fazla doğruluk oranı ile nesne tespiti ve tanımlama yapabilen yöntemler geliştirilmiştir. Literatürde nesne takibi genel olarak ön işlemler, nesne tespiti, nesne sınıflandırma ve nesne takibi olmak üzere dört farklı aşamada ele alınmaktadır. Bu aşamalardan özellikle nesne tespiti daha fazla önem taşımaktadır ve bu aşamanın başarısı sonraki aşamalarının başarısını 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. [1]

Derin öğrenme insan beynindeki işlevleri taklit eden bir hesaplama sistemine dayanır. Derin öğrenme, 1943 yılında McCulloch ve Pitts tarafından fikir üretme sürecini taklit etmek için matematiğe ve sinir mantığı olarak adlandırılan algoritmalara dayalı bir hesaplama modeli geliştirmelerine dayanmaktadır. 1958 yılında Rosenblatt iki katmanlı bilgisayar yapay sinir ağına dayalı, denetimli öğretimli bir desen tanıma algoritması geliştirmiştir. 1965 yılında Ivakhnenko ve Lapa derin öğrenme algoritmalarını geliştirmeye yönelik olarak karmaşık denklemlerin aktivasyon fonksiyonlarına sahip modelleri kullanmışlardır. 1988 yılında Fukushima el yazısı tanıma ve diğer desen tanıma sorunları için kullanılan hiyerarşik ve çok katmanlı yapay sinir ağı olan Neocognitron'ı önermiştir. 1992 yılında Weng, Cohen ve Herniou karma sahnelerden otomatik olarak üç boyutlu nesne tanıma işlemini gerçekleştiren Cresceptron yöntemini yayımlamıştır. 1995 yılında Cortes ve Vapnik tarafından benzer verilere sahip iki grubun sınıflandırılması için destek vektör ağları

kullanılmıştır. 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. “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. 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 [1].

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. 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.[1]

Bu çalışmada nesne tanıma ve takibi Faster R-CNN kütüphanesinin kullanımı ele alınmıştır. Bu kütüphane ile birlikte özellikle derin öğrenme için kullanılan TensorFlow kullanılarak çocukların çizmiş olduğu kaktüs resimlerinde nesne tanıma yapmak amaçlanmıştır.

Bölüm 2

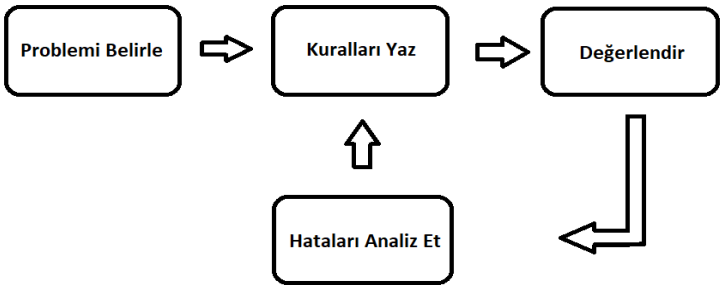
2.1 Yapay Zekâ

Doğadaki varlıkların akıllı davranışlarını yapay olarak üretmeyi amaçlayan ve işini mükemmel yapan canlı sistemlerini ve insan beynini model alan yapay zekâ çalışmaları; günlük hayatın farklı alanlarında ürünler vermesinin yanı sıra, tahmin, sınıflandırma, kümeleme gibi amaçlar için de kullanılmaktadır. Başlıca olarak uzman sistemler, genetik algoritmalar, bulanık mantık, yapay sinir ağları, makine öğrenmesi gibi teknikler, genel olarak yapay zekâ teknolojileri olarak adlandırılmaktadır. Bu tekniklerin yanı sıra doğanın taklidi amacıyla da canlılar incelenmekte ve benzeri akıllı yöntemler önerilmektedir. Karınca kolonisi, parçacık sürü ve yapay arı gibi algoritmalar, yapay zekâ optimizasyon teknikleri olarak kullanılmaktadır. Genel anlamda yapay zekâdan kastedilen; insan zekâsının, sinir sistemi, gen yapısı gibi fizyolojik ve nörolojik yapısının ve doğal olayların modellenerek bilgisayar ve yazılımlara aktarılmasıdır. Özetle yapay zekâ; "insan gibi düşünen, insan gibi davranan, rasyonel düşünen ve davranan" canlıların zekice olarak kabul edilen davranışlarına sahip bilgisayar sistemleridir ve makine öğrenmesi bu anlamda yapay zekânın son evresi olarak kabul edilmektedir [2].

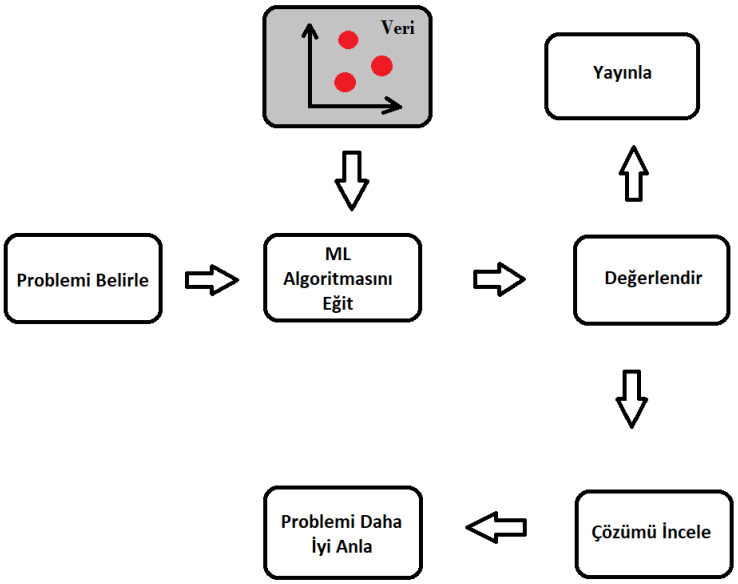
2.2 Makine Öğrenmesi

Makine öğrenmesi, bir problemi o probleme ait veriye göre modelleyen bilgisayar algoritmalarının genel adıdır. Mevcut veri seti ve kullanılan algoritma ile oluşturulan model, en yüksek performansı vermek üzere kurulmaktadır. Bu nedenle pek çok makine öğrenmesi yöntemi geliştirilmiştir. Bunlardan bazıları; K-en yakın komşu algoritması, Naive Bayes sınıflandırıcı, karar ağaçları, lojistik regresyon analizi, K-ortalamalar algoritması, destek vektör makinaları ve yapay sinir ağlarıdır. Bu yaklaşımların bir kısmı tahmin ve kestirim, bir kısmı kümeleme ve bir kısmı da sınıflandırma yapabilme yeteneğine sahiptir. Bu yöntemlerde öğrenme stratejileri; denetimli, denetimsiz ve pekiştirmeli olmak üzere üç grupta incelenmektedir. Denetimli öğrenmede oluşturulan model ile, bir grup girdi değerine karşılık onlara ait hedef değerleri verilerek aralarındaki ilişkiyi öğrenmesi ve hedef değerlere en yakın

çıktıların üretilmesi amaçlanır. Elde edilen en iyi model, yeni girdi değerleri için en yakın çıktıyı da verebilecektir. Denetimsiz öğrenmede ise hedef değerleri olmadan sadece girdi değerleri arasındaki ilişki ortaya çıkarılmaya çalışılır. Bu ilişki veya ilişkiler yardımıyla birbirine yakın değerler gruplandırılarak kümeleme işlemi yapılır. Yeni girdi bu kümelerden hangisiyle ilişkiliyse o kümeye ait olacaktır. Pekiştirmeli öğrenme yönteminde, hedef çıktıyı vermek için bir danışman yerine, elde edilen çıkışın verilen girişe karşılık iyi ya da kötü olarak değerlendiren bir kriter kullanılmaktadır [2].



Şekil 2.1: Geleneksel programlarda eğitim işleyişi



Şekil 2.2: Makine Öğrenimi Algoritması

2.3 Çok Katmanlı Yapay Sinir Ağları

Yapay Sinir ağları insan beyninin en temel özelliği olan öğrenme fonksiyonunu gerçekleştiren bilgisayar sistemleridir. Öğrenme işlemini örnekler yardımı ile gerçekleştirirler. Bu ağlar birbirine bağlı yapay sinir hücrelerinden oluşur. Her bağlantının bir ağırlık değeri vardır. Yapay sinir ağının sahip olduğu bilgi bu ağırlık değerlerinde saklı olup ağa yayılmıştır. Yapay sinir ağları bilinen hesaplama yöntemlerinden farklı bir hesaplama yöntemi önermektedir. Bulundukları ortama uyum sağlayan, adaptif, eksik bilgi ile çalışabilen, belirsizlikler altında karar verebilen, hatalara karşı toleranslı olan bu hesaplama yönteminin hayatın hemen hemen her alanında başarılı uygulamalarını görmek mümkündür. Oluşturulacak olan ağın yapısının belirlenmesinde, ağ parametrelerinin seçiminde, belirli bir standardın olmaması, problemlerin sadece numerik bilgiler ile gösterilebilmesi, eğitimin nasıl bitirileceğinin bilinmemesi ve ağın davranışlarını açıklayamamasına rağmen bu ağlara olan ilgi her geçen gün artmaktadır. Özellikle, sınıflandırma, örüntü tanıma, sinyal filtreleme, veri sıkıştırma ve optimizasyon çalışmalarında yapay sinir ağları en güçlü teknikler arasında sayılabilirler. Veri madenciliği, optik karakter taşıma, optimum rota belirleme, parmak izi tanıma, malzeme analizi, iş çizelgelemesi ve kalite kontrol, tıbbi analizler gibi birçok alanda günlük hayatımızda göreceğimiz başarılı örneklerine rastlamak mümkündür.

Yapay sinir ağları biyolojik sinir sisteminden etkilenerek geliştirilmiştir. Biyolojik sinir hücreleri birbirleri ile synapsler vasıtası ile iletişim kurarlar. Bir sinir hücresi işlediği bilgileri axon'ları yolu ile diğer hücrelere gönderirler. Benzer şekilde yapay sinir hücreleri dışarıdan gelen bilgileri bir toplama fonksiyonu ile toplar ve aktivasyon fonksiyonundan geçirerek çıktıyı üretip ağın bağlantılarının üzerinden diğer hücrelere (proses elemanlarına) gönderir. Değişik toplama ve aktivasyon fonksiyonları vardır. Yapay sinir ağlarını birbirlerine bağlayan bağlantıların değerlerine ağırlık değerleri denmektedir. Proses elemanları birbirlerine paralel olarak 3 katman halinde bir araya gelerek bir ağ oluştururlar. Bunlar;

• Girdi katmanı

• Ara katmanlar

• Çıktı katmanıdır.

Bilgiler ağa girdi katmanından iletilir. Ara katmanlarda işlenerek oradan çıktı katmanına gönderilirler. Bilgi işlemeden kasıt ağa gelen bilgilerin ağın ağırlık değerleri kullanılarak çıktıya dönüştürülmesidir. Ağın girdiler için doğru çıktıları üretebilmesi için ağırlıkların doğru değerlerinin olması gerekmektedir. Doğru ağırlıkların bulunması işlemine ağın eğitilmesi denmektedir. Bu değerler başlangıçta rasgele atanırlar. Daha sonra eğitim sırasında her örnek ağa gösterildiğinde ağın öğrenme kuralına göre ağırlıklar değiştirilir. Daha sonra başka bir örnek ağa sunularak ağırlıklar yine değiştirilir ve en doğru değerleri bulunmaya çalışılır. Bu işlemler ağ eğitim setindeki örneklerin tamamı için doğru çıktılar üretinceye kadar tekrarlanır. Bu sağlandıktan sonra test setindeki örnekler ağa gösterilir. Eğer ağ test setindeki örneklere doğru cevaplar verirse ağ eğitilmiş kabul edilmektedir. Ağın ağırlıkları belirlendikten sonra her bir ağırlığın ne anlama geldiği bilinmemektedir. O nedenle yapay sinir ağlarına “kara kutu” yakıştırması yapılmaktadır. Ağırlıkların tek tek ne anlama geldikleri bilinmemekle birlikte ağın girdiler hakkındaki kararını bu ağırlıkları kullanarak vermesi, ağın zekasının bu ağırlıklarda saklandığı söylenebilir. Ağın bir olayı öğrenmesi o olay için en doğru yapay sinir ağı modelini seçmekle mümkündür. Şu ana kadar birçok yapay sinir ağı modeli geliştirilmiştir. Bir yapay sinir ağının modelini şu bilgiler karakterize etmektedir.

• Ağın topolojisi

• Kullanılan toplama fonksiyonu

• Kullanılan aktivasyon fonksiyonu

• Öğrenme stratejisi

• Öğrenme kuralı

Geliştirilen modeller arasında en yaygın olarak kullanılanları, tek ve çok katmanlı algılayıcılar, LVQ, ART ağları, SOM, Elman ağı gibi ağlardır.[3]

2.4 Derin Öğrenme

Derin Öğrenme; nesne veya konuşma tanıma, doğal dili işleme gibi çeşitli alanlarda çok katmanlı yapay sinir ağlarını kullanan bir yapay zekâ yöntemi ve makine öğrenmesi çeşitlerinden biridir. Derin öğrenmesi, geleneksel makine öğrenmesi yöntemlerinden farklı olarak kodlanmış bir takım kural ile öğrenmek yerine resim, video, ses ve metin verilerinin simgelerinden otomatik şekilde öğrenim sağlayabilmektedir. Esnek yapıda olmaları sebebiyle, ham resim ya da metin verilerinden de öğrenebilmektedir. Derin öğrenme örnekler üzerinden öğrenme gerçekleştirir. Makineye, çözmesi istenen bir problemi kural setlerini kullanarak çözmek yerine örnekleri değerlendirerek probleme çözüm getirmesini sağlayan bir model verilmesi yeterlidir. Problemin çözümündeki hatayı düzeltebilmesi için de basit bir komut listesi verilerek makinenin öğrenme işlemini gerçekleştirmesi beklenir. Model seçimi, problemin çözümünde etkindir. Probleme uygun olarak belirlenecek model, problemin çözümüne daha fazla katkıda bulunacaktır. Derin öğrenme kavramı ilk kez 2006 yılında Hinton tarafından çok katmanlı yapay sinir ağlarının daha verimli eğitilebileceğinin öne sürülmesiyle ortaya çıkmıştır [4].

Derin öğrenmede, verilerin birden fazla özellik seviyesinin veya temsillerinin öğrenilmesine dayanan bir yapı söz konusudur. Üst düzey özellikler, alt düzey özelliklerden türetilerek hiyerarşik bir temsil oluşturur. Bu temsil, soyutlamanın farklı seviyelerine karşılık gelen birden çok temsil seviyesini öğrenmektedir. Derin öğrenme temel olarak verinin temsilinden öğrenmeye dayalıdır. Bir görüntü için temsil denildiğinde; piksel başına yoğunluk değerlerinin bir vektörü veya kenar kümeleri, özel şekiller gibi özellikler düşünülebilir. Bu özelliklerin içinden bazıları veriyi daha iyi temsil etmektedir. Bu aşamada yine bir avantaj olarak, derin öğrenme yöntemleri, elle çıkarılan özellikler yerine veriyi en iyi temsil eden hiyerarşik özellik çıkarımı için etkin algoritmalar kullanmaktadır.[5]

Bölüm 3

3.1 Bilgisayar Görüşü

Bilgisayar görüşü, dijital görüntülerden bilgi alan bilimsel bir alan olarak tanımlanabilmektedir. Nesne tespitinde görüntüler bilgisayar görüşünden faydalanılarak elde edilmektedir. Elde edilen görüntüler görüntü işleme teknikleri aracılığıyla nesne tespitinde oldukça başarılı olmaktadır. Bilgisayar görüşü, görüntülerin içeriğini anlayabilir hale getirmek ve uygulamalar için kullanılabilir algoritmalar oluşturmaktır. Bilgisayar görüşü, görüntülerin otomatik bir şekilde çıkarılmasıdır. Bilgi, 3B modeller, kamera konumu, nesne algılama ve tanıma ile görüntü içeriğini gruplama ve arama arasında bir anlam ifade edebilmektedir. Pratik bilgisayar görüşü, programlama, modelleme ve matematiğin bir karışımını içermektedir.

Bilgisayar Görüşü bir fotoğraf ya da video kameradan elde edilen verinin bir karara ya da yeni bir veriye dönüştürülmesidir. Belirli bir amaca ulaşmak için yapılan bu dönüşümler bir kamera veya buna benzer bir aygıttan alınan verileri karar olarak görüntü içeriği hakkında bilgi sağlamaktır. Bir başka amacı ise renkli bir görüntüyü gri tonlamalı bir görüntüye çevirmek veya kamera hareketini bir görüntü dizisinden kaldırmaktır. İnsan beyni, görme sinyalini farklı türden bilgiler aktaran birçok kanala bölmektedir. Ancak bilgisayar görüşünde gözün yaptığı işlemi bilgisayar gördüğü şeyi sadece bir sayı ızgarası olarak algılamaktadır. Bilgisayarın algıladığı görüntüler sayısal bir ızgaraya dönüştürülmektedir. Izgara içinde verilen herhangi bir sayı oldukça büyük bir gürültüye sahiptir ve bu yüzden kendi başına çok az bilgi vermektedir. Bilgisayarın görüşüyle ilgili sorun gürültüdür. Genelde istatistiksel yöntemleri kullanarak gürültüyü yok etmeye uğraşılmaktadır. Bilgisayarların algıladığı görüntülerdeki bu sorunları çözmek için çeşitli görüntü işleme algoritmaları kullanılmaktadır [6].

Görüntüdeki gürültülerin giderilmesinin bu kadar zor olmasının nedeni, görüşün yetersiz bilgi aktarması, bilinmeyen bazı bilgileri bilinmeyen bir şekilde çözmek hedeflendiğinde ters bir problem oluşturmasıdır. Fakat, görsel hafızamızda bulunan

dünyayı bütün karmaşıklığı ile modellemek, konuşulan sesleri ise üreten ses yolunu modellemekten çok daha zordur. Bilgisayarlı görmede kullanılan ileri modeller genellikle fizikte (radyometri, optik ve sensör tasarımı) ve bilgisayar grafikleriyle geliştirilmektedir. Her iki alan da nesnelerin nasıl hareket ettiği, canlandırdığı, ışığın yüzeyleri nasıl yansıttığı, atmosfer tarafından nasıl dağıldığı, kamera lenslerinden (veya insan gözlerinden) nasıl kırıldığı ve bir görüntü düzlemine nasıl yansıtıldığını göstermektedir. İnsanların ve hayvanların bu işlevi kusursuzca ve hiçbir emek harcamadan yapmaları benzersizken, bilgisayar görme algoritmaları ise hataya oldukça açık ve yeteri kadar başarılı değildir [6].

Bilgisayar görüşünde pikseller ve anlam arasında büyük bir boşluk olması sebebiyle modelleme zordur. Bilgisayarın 200 × 200 RGB görüntüde gördüğü şey 120,000 değer kümesidir. Bu rakamlardan anlamlı bilgiye giden yolu bulması çok zordur. Muhtemelen, insan beyninin görsel korteksi retinaya yansıtılan ve nöron sinyallerine dönüştürülen görüntüleri anlama problemini zor olsa da çözmektedir. Bu sorunu çözmek için bilgisayar görüşünün iki avantajı bulunmaktadır.

İlk olarak, bir algılama cihazı bir görüntüden mümkün olduğunca fazla ayrıntı yakalamaktadır. Göz, iris içinden gelen ışığı yakalayarak özel hücrelerin beyne nöronlar aracılığıyla bilgi ileteceği retinaya yansıtmaktadır. Bir kamera da görüntüleri benzer şekilde yakalamakta ve pikselleri bilgisayara aktarmaktadır. Kameralar da kızılötesini görebildikleri, daha uzağı daha hassas ayarda görebildiği için kameralar insanlardan daha iyidir [6].

İkinci olarak, yorumlama cihazı bilgiyi işlemek ve ondan anlam çıkarmak zorundadır. İnsan beyni bunu beynin farklı bölgelerinde birçok adımda çözmektedir. Bilgisayar görüşü hala bu alanda insan performansının gerisinde kalmaktadır. Kameralar her yerde ve internete yüklenen görüntü sayısı katlanarak artmaktadır. Instagram'da yer alan görüntüler, YouTube'da bulunan videolar, güvenlik kameraları, tıbbi ve bilimsel görüntüler bulunmaktadır. Bilgisayar görüşü çok önemlidir. Çünkü bu görüntüler arasında sıralama yapmak ve bilgisayarların içeriklerini anlamasını sağlamak gerekmektedir [6].

Bilgisayar görüşü alanı nesne tespit probleminin çözümünde görüntü işleme teknikleriyle birlikte nesne tespitinde kolaylık sağlamaktadır. Görüntülerden elde

edilen nesnenin içinde bulunduğu pencerelerden nesnenin tanımlanmasını nesne tespit algoritmaları kullanarak gerçekleştirmektedir. Bilgisayarlı görüş uygulamasında ilk aşama yani görüntü işleme öncesi görüntü işleme sonrası ve görüntüyü uygun bir analize dönüştürmek için görüntü işleme kullanımı örnek olarak pozlama, düzeltme ve renk dengeleme, görüntü parazitinin azaltılması, keskinliğin artırılması veya görüntünün döndürülerek düzleştirilmesidir. Bazıları görüntü işlemenin bilgisayar görüşünün dışında olduğunu düşünebilir, ancak bilgisayarlı fotoğrafçılık ve hatta tanıma gibi çoğu bilgisayar görme uygulaması kabul edilebilir sonuçlar elde etmek için görüntü işleme aşamalarını kullanmaya özen gösterilmektedir [6].

3.2 Derin Öğrenme Mimarisi

Derin öğrenme mimarisinde yer alan 6 mimari yapısı bulunmaktadır. Bu yapılardan ilki, “Konvolüsyonel Sinir Ağları” mimarisidir. Bilgisayarlı görü görevlerinde etkili ve bu çalışmanın modellenmelerinde ayrıntılı kullanılan bir mimaridir[12].

CNN, derin öğrenme kavramına ait temel mimari yapısıdır. İlk CNN 1988 yılında Yann Le Cun tarafından oluşturulmuştur. 1998’lere kadar sürekli geliştirilen Le Net ilk CNN mimarisidir. CNN’ler görüntüleri girdi olarak almak için tasarlanmış yapılardır ve bilgisayarlı görmede etkili bir şekilde kullanılmaktadır. CNN, bir veya birden çok konvolüsyonel katman ve standart çok katmanlı bir sinir ağı gibi bir veya daha fazla tamamen bağlı katmanlardan oluşmaktadır[12].

Diğer mimari yapılardan biri olan “Tekrarlayan Sinir Ağları”; birimler arası verilerin yönlendirilmiş bir döngü oluşturduğu yapay sinir ağıdır. Tekrarlayan sinir ağları, görüntü tanımlayıcıları kullanmak için diğer mimarilerle birlikte kullanılmaktadır. “Uzun-Kısa Vadeli Hafıza Ağları” mimarisi; değerleri rastgele aralıklarla hatırlayan bir tekrarlayan sinir ağı mimarisidir. Bu mimari yapısı, konuşma/metin işleme çalışmalarında kullanıldığında iyi sonuçlar vermektedir[12].

“Sınırlı Boltzman Makineleri”; girdi seti üzerinde olasılık dağılımlarını öğrenebilen görünür katman ve gizli katman olmak üzere 2 katmandan oluşan bir yapay sinir ağı yapısıdır. Sınırlı Boltzman Makineleri; sınıflandırma, özellik öğrenimi ve modelleme için uygun bir ağ olarak bilinir[12].

“Derin İnanç Ağları” yapısı; Sınırlı Boltzman Makineleri’nin yığını ve oto kodlayıcılarla bileşimi olarak görülür. Derin İnanç Ağları, görüntü tanıma uygulamalarında kullanılmaktadır.[12]

“Derin Oto Kodlayıcılar” mimari yapısı ise; denetimsiz öğrenme için ve verinin üretken modellerini öğrenmek için kullanılan, girdi katmanı verilerini, çıktı katmanına kopyalayan bir yapıdır[12].

3.3 Derin Öğrenme Kütüphaneleri

3.3.1 Tensorflow

2011 yılında TensorFlow Google araştırma ekibi olan Google Brain tarafından geliştirilen bir derin öğrenme kütüphanesidir. Bu grup ilk olarak DistBelief makine sistemini oluşturmuşlardır. DistBelief ile öğretmensiz öğrenme, dil çevrimi, görüntü sınıflandırması ve nesne tespiti, video sınıflandırması, konuşma tanıma , dizi tahmini , yaya saptama, takviyeli öğrenme gibi pek çok çalışmada kullanılmıştır. Daha sonra makine öğrenimini daha etkin hale getirebilecek için ikinci nesil derin öğrenme kütüphanesi olan TensorFlow’u geliştirmişlerdir. Bu kütüphanedeki yapıda temel olarak bir dizi hesaplamalardan oluşan veri akış grafiklerinden oluşmaktadır. Bu akış grafikleri düğümlerin durumunu korumak, güncellemek için dallanma ve döngü kontrolüne izin veren bir veri akışı hesaplamasını sunar. Her bir düğüm 0 veya daha fazla girişe ve 0 veya daha fazla çıkışa sahip olan işlem örneği gösterir. Açık kaynaklı yapısı ile geniş çaplı şekilde kullanılması amaçlanmıştır. Mobil cihazlar, tabletler, telefonlar, büyük ölçekli dağıtık sistemlerde, GPU kartları gibi pek çok hesaplama aygıtlarında herhangi bir değişiklik yapılmaksızın kullanılabilmesi mümkündür[11].

3.3.2 Torch

Ronan Collobert ve arkadaşları tarafından derin öğrenme ve makine öğrenmesine destek olmak için geliştirilmiş açık kaynak kodlu bir kütüphanedir. Birkaç satır kod ile çok kolay şekilde derin öğrenme yapısı oluşturulabilmektedir. Sayısal optimizasyon yapılması mümkündür (Collobert vd., 2011). Açık kaynak kodlu olması sebebiyle ticari olarakta kullanılabilmektedir. Bunlara örnek olarak Facebook, twitter, Google

DeepMind verilebilir. Pek çok üniversite ve araştırma merkezleri tarafından kullanılmaktadır[11].

3.3.3 Keras

Keras, üst düzey sinir ağları kütüphanesidir. Theano veya Tensorflow kütüphaneleri tabanda Keras kullanmıştır. Python dili ile yazılmıştır. Hızlı sonuçlar vermeyi odak noktası kabul ederek geliştirilmiştir. Temel anahtar kelimesi fikirden sonuca mümkün olan en az gecikmeyle gidebilmektir. Keras'ın temel üstünlükleri aşağıda belirtilmiştir.

Ø Kolay ve hızlı prototipleme sağlar.

Ø Hem konvolüsyonel ağları hem de tekrar eden ağları ve ikisinin kombinasyonlarını destekler.

Ø Çoklu giriş ve çoklu çıkış eğitimi dahil raslantısal bağlantı planlarını destekler.

Ø CPU ve GPU üzerinde kusursuz çalışır.

3.3.4 Caffe

Caffe Berkeley Vision and Learning Center ve kullanıcı topluluğu tarafından geliştirilmiş ve sürekli güncellenmektedir. Yangqing JİA'nin doktora sırasında ortaya koymuş olduğu bir projedir. Derin öğrenme modellerinin eğitimi için hem CPU hemde GPU ile eğitim yapılmasına olanak sağlamaktadır. C++ programlama dilinde geliştirilmiştir. Python ve Matlab arabirimlerini içerir. Araştırma deneyleri ve endüstride kullanım için Caffe ileri düzeyde kullanılmaktadır. Caffe, tek bir NVIDIA K40 GPU ile günlük 60M'tan fazla görüntü işleyebilir. Çıkarım hızı 1 ms / görüntü, iken eğitim için 4 ms / görüntü hızında çalışmaktadır.

3.3.5 OpenCV

OpenCV(Open Source Computer Vision) açık kaynak kodlu görüntü işleme kütüphanesidir. 1999 yılında İntel tarafından geliştirilmeye başlanmış daha sonra Itseez, Willow, Nvidia, AMD, Google gibi şirket ve toplulukların desteği ile gelişim süreci devam etmektedir. İlk sürüm olan OpenCV alfa 2000 yılında piyasaya çıkmıştır. İlk etapta C programlama dili ile geliştirilmeye başlanmış ve daha sonra birçok

algoritması C++ dili ile geliştirilmiştir. Açık kaynak kodlu bir kütüphanedir ve BSD lisansı altında geliştirilmektedir. BSD lisansına sahip olması bu kütüphanenin istenilen projede ücretsiz olarak kullanabileceği anlamına gelmektedir. OpenCV platform bağımsız bir kütüphanedir, bu sayede Windows, Linux, FreeBSD, Android, Mac OS ve iOS platformlarında çalışabilmektedir. C++, C, Python, Java, Matlab, EmguCV kütüphanesi aracılığıyla da Visual Basic.Net, C# ve Visual C++ dilleri ile topluluklar tarafından geliştirilen farklı wrapperlar aracılığıyla Perl ve Ruby programlama dilleri ile kolaylıkla OpenCV uygulamaları geliştirilebilir. 2016-05-27 tarihli güncelleme, OpenCV geliştirici Itseez firması Intel tarafından satın alınmış ve OpenCV geliştirmesine Intel çatısı altından devam edeceği duyurulmuştur[7].

OpenCV kütüphanesi içerisinde görüntü işlemeye (image processing) ve makine öğrenmesine (machine learning) yönelik 2500'den fazla algoritma bulunmaktadır. Bu algoritmalar ile yüz tanıma, nesneleri ayırt etme, insan hareketlerini tespit edebilme, nesne sınıflandırma, plaka tanıma, üç boyutlu görüntü üzerinde işlem yapabilme, görüntü karşılaştırma, optik karakter tanımlama OCR (Optical Character Recognition) gibi işlemler rahatlıkla yapılabilmektedir[7].

OpenCV; mobil ve masaüstü olmak üzere farklı platformlarda çalışabilmektedir. OpenCV hesaplama verimliliğini arttırmak amacıyla gerçek zamanlı uygulamalar için tasarlanmıştır. Bu çalışmada tasarlanan bilgisayarlı görü sisteminde kaktüs nesnesinin varlığının tespiti için OpenCV'nin Python için geliştirilmiş Haar Cascade Classifier kütüphanesinden yararlanılmıştır.

3.4 Boosting

Boosting (Arttırma), bir çok zayıf öğreniciyi bir araya getirerek bir güçlü öğrenici oluşturmaktır. Birçok boosting metodunun temel yaklaşımı, tahmin edicileri kümülatif olarak eğitmektir. Genelde tahminleyici model olarak karar ağaçları kullanılır. Her ağaç orijinal bir veri kümesinin değiştirilmiş bir versiyonu ile eğitilir ve sonunda güçlü bir sınıflandırıcı oluşturulur. En sık kullanılan boosting modelleri:

-AdaBoost,

-Gradient Boosting,

-XGBoost,

-LightGBM ve

-CatBoost modelleridir.

3.4.1 Adaboost

AdaBoost, ilk boosting algoritması olarak sayılır ve bilgisayar dünyasında önemli ödüllerden biri olan Gödel ödülünü kazanmıştır. Bu modelde eğitim kümesi önce bir zayıf öğrenici ile eğitilir. Eğitim sonrası yanlış olarak tahminlenen örnekler bu algoritma için önemlidir. Bir sonraki eğitimde ilk tahminlemede yanlış öğrenilen eğitim verilerine daha fazla öncelik verilerek yani ağırlıkları artırarak tekrar eğitilir.

Bu şekilde zayıf öğrenici çıkışı diğer öğreniciye giriş olacak şekilde eğitilerek devam edilir ve en sonunda sonuçlar birleştirilerek nihai karar sınırları oluşturulur. Eğitim kümesinden eğitilmiş maksimum derinliğe sahip karar ağaçları kullanılmaz. Bunun yerine zayıf öğrenici olarak AdaBoost ile bir nod ve iki yapraktan oluşan derinliği bir olan karar ağaçları kullanılır. Kümülatif bir yapı olduğu için sıralama önemlidir. İlk karar ağacının yaptığı hata, ikinci ağacın ağırlıklarını etkiler. Bagging yönteminin aksine parelel bir hesaplama yapmaz bunun yerine ardışık bir hesaplama yapılır[8].

Bölüm 4

4.1 Veri Seti

Kaktüs nesnesini tanımada kullanılacak görseller 4-18 yaş arası çocukların ‘Bir Kaktüs Çiz Testi’ adı altında yapılan çalışma ile 4 okul, 1 rehabilitasyon merkezi ve 1 anaokulunda bulunan toplam 371 öğrencinin çizimlerinden elde edilmiştir. Şekil 4.1 ve Şekil 4.2 ‘de Labelimg ile kaktüs, çiçek, saksı ve güneş sınıf etiketlemelerinin her bir görsel için nasıl yapıldığı örneği görülmektedir. Etiketleme işlemini yapmak için test, train ve valid klasörü oluşturulmuştur ve %70 train, %10 validation, %20 test verisi olacak şekilde klasörlere resimler dağıtılmıştır. Resimleri etiketlemede

kullanılan LabelImg uygulamasında klasörler tek tek açılıp etiketleme işlemi yapılmıştır.

Şekil 4.1 Labelimg ile görsel etiketleme işlemi

Şekil 4.2 Labelimg ile görsel etiketleme işlemi

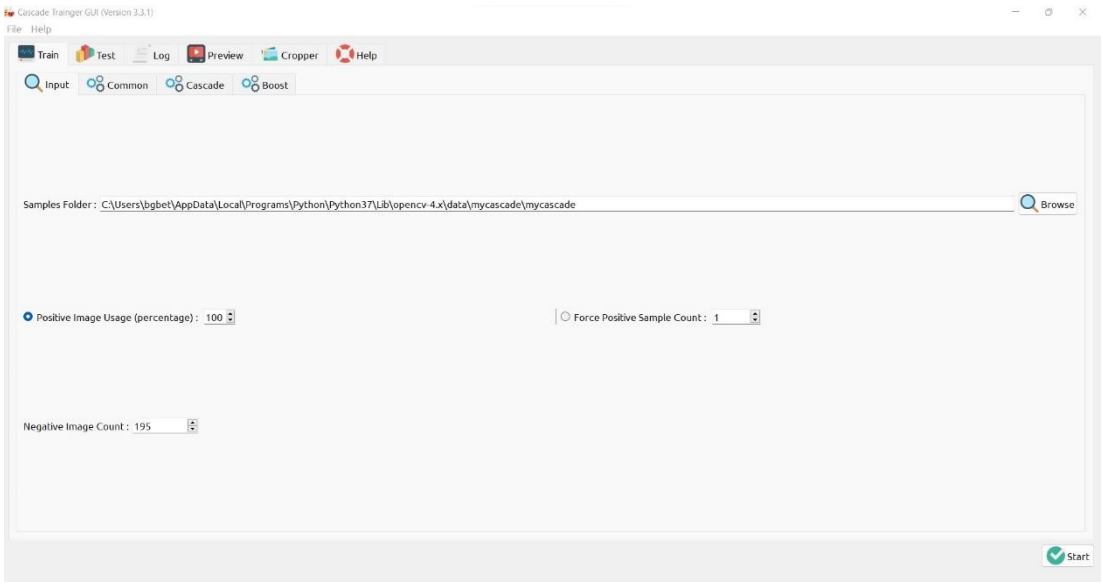
Etiketleme işlemi tamamlandığında her bir görselin sınıf ismi ve koordinasyon bilgilerinin bulunduğu .xml uzantılı dosyalar ilgili klasörlerde elde edilir. Oluşturulan veriseti Roboflow artificial intelligence uygulamasında açılarak önişlem olarak

görseller ölçeklendirilip boyutları ortak bir boyuta getirilmiştir. Renkli görseller gri ölçeklendirilerek eğitimin hızlandırılması sağlanmıştır.

Sonraki ‘augmentations’aşamasında resimlerin rotate edilmesi, blur işlemi, ayna görüntüsünü alma gibi işlemler ile veriseti çoğaltılmış ve makine zorlanarak daha iyi sonuçlar elde etmek hedeflenmiştir. Sonuç olarak 908 görsel elde edilmiştir.

4.2 Eğitim

Bu çalışmada kaktüs tespiti işleminde Haar sınıflandırıcıyı eğitmek amacıyla bir eğitim veriseti oluşturulmuştur. Eğitim veriseti, hem içerisinde bulunması istenen nesnenin yer aldığı pozitif resimleri hem de yer almadığı negatif resimleri içermektedir. Pozitif resim içerisindeki nesneyi tespit etmek amacıyla 24x24 piksel boyutlarındaki alt-pencereler tüm resim boyunca kaydırılarak nesne taraması gerçekleştirilerek Cascade Trainer GUI yazılımı ile eğitim işlemi gerçekleştirilmektedir. Bu pencereler Haar-benzeri öznitelikler olarak adlandırılmaktadır. Cascade Trainer GUI programı başlatıldığında, aşağıdaki ekranla karşılaşılmaktadır.



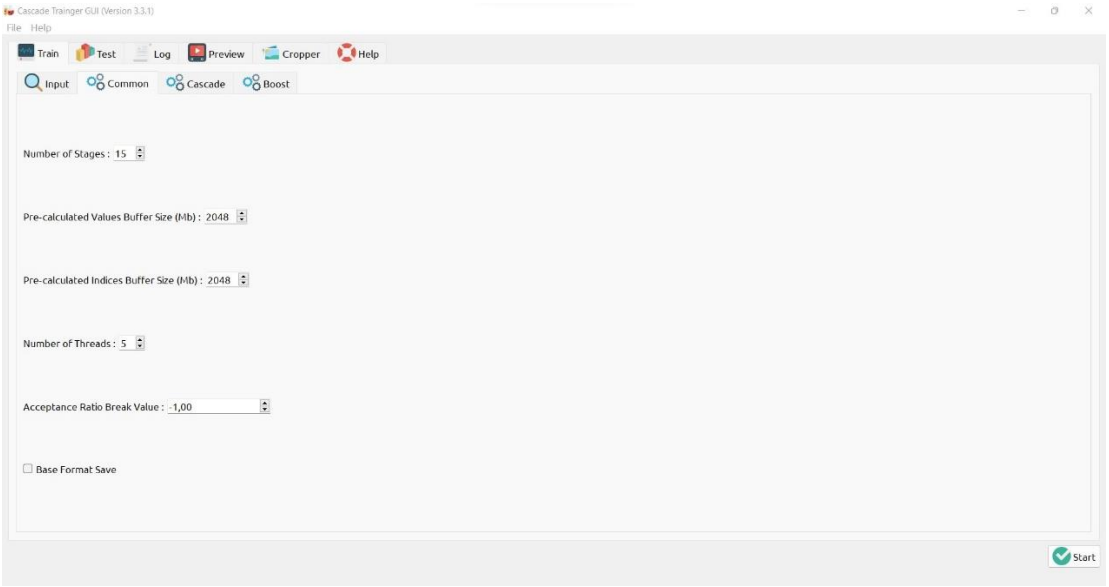
Şekil 4.3: Cascade Trainger GUI Input işlemleri

Bu başlangıç ekranıdır ve sınıflandırıcıları eğitmek için kullanılabilir. Sınıflandırıcıları eğitmek için genellikle yardımcı programa binlerce pozitif ve negatif görüntü örneği sağlamak gerekir, ancak aynı şeyi daha az örnekle de başarılabildiği durumlar vardır.

Eğitime başlamak için sınıflandırıcı için bir klasör oluşturmak gerekmektedir. Bu sebeple "opencv" klasörünün içerisine "mycascade" isimli klasör oluşturuldu. Ve bu klasörün içinde biri "p" (pozitif görüntüler için), diğeri "n" (negatif görüntüler için) iki klasör oluşturuldu. Pozitif görüntü örnekleri, sınıflandırıcıyı eğitmek ve algılaması istenilen nesnenin görüntüleridir. "p" klasöründe kaktüs nesnesini barındıran jpeg. Formatında olan olumlu görüntü örnekleri bulunmaktadır. "n" klasöründe ise kaktüs nesnesini hiçbir şekilde barındırmayan jpeg. formatında görseller bulunmaktadır.

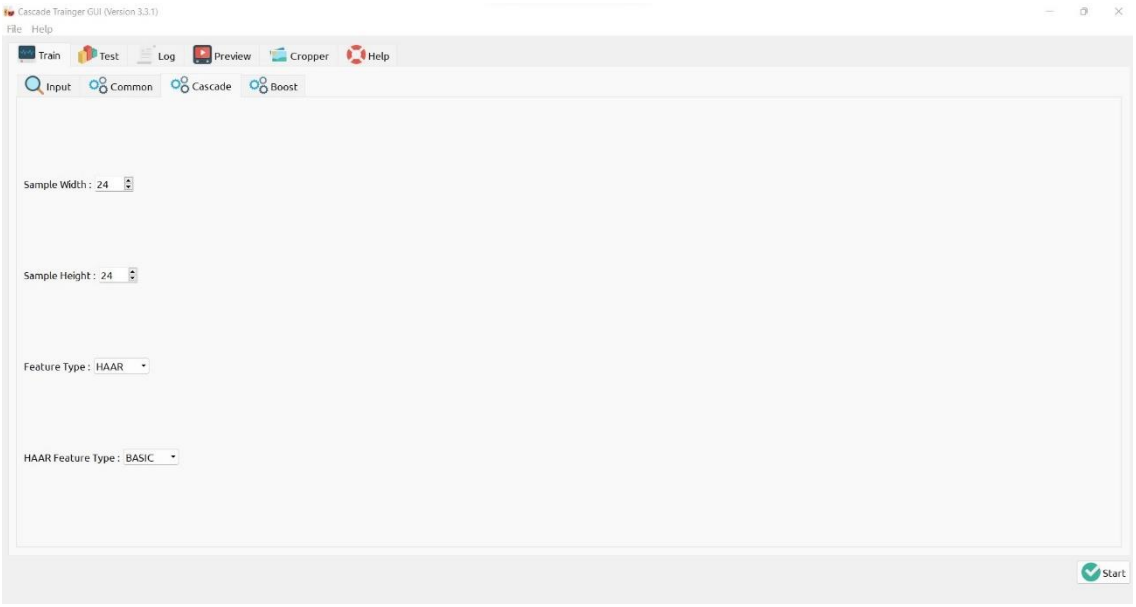
Düzgün bir eğitim için birçok olumlu ve olumsuz imaj klasörlere eklenmelidir. Bu noktada en önemli 2 nokta vardır; ilk önemli nokta olumsuz görüntülerin içinde asla tanınmak istenen nesne bulunmamalıdır. İkinci nokta ise teoride negatif imajlar, pozitif imaj olmayan herhangi bir imaj olabilir ama pratikte negatif imajlar pozitif imajlarla alakalı olmalıdır.

Şekil 4.3'de görülmekte olan arayüzde giriş ekranındaki Train sekmesinde "browse" kısmına sınıflandırıcı için oluşturulan "mycascade" klasörünün yolu eklenir. "p" klasörüne eklenen positive image'lerin yüzde olarak kullanılmak istenen miktarı belirtilir. Projede tüm görseller kullanılacağı için 100 girilmiştir.



Şekil 4.4: Cascade Trainger GUI Common işlemleri

Şekil 4.4 de ekrandaki Common sekmesinde Number of stages kısmında kaç aşamalı bir eğitim olması isteniyorsa belirtilir. 15 aşamalı olarak sisteme girilmiştir. Bu işlem eğitimin hızını belirleyecektir. Pre-calculated Values Buffer Size(mb) kısmında eğitim yapılacak ortamın arabellek boyutuna göre değer girilir. Örneğin kullanılan bilgisayar 8Gb RAM'e sahipse her iki arabellek boyutu da güvenle 2048'e ayarlanabilir.

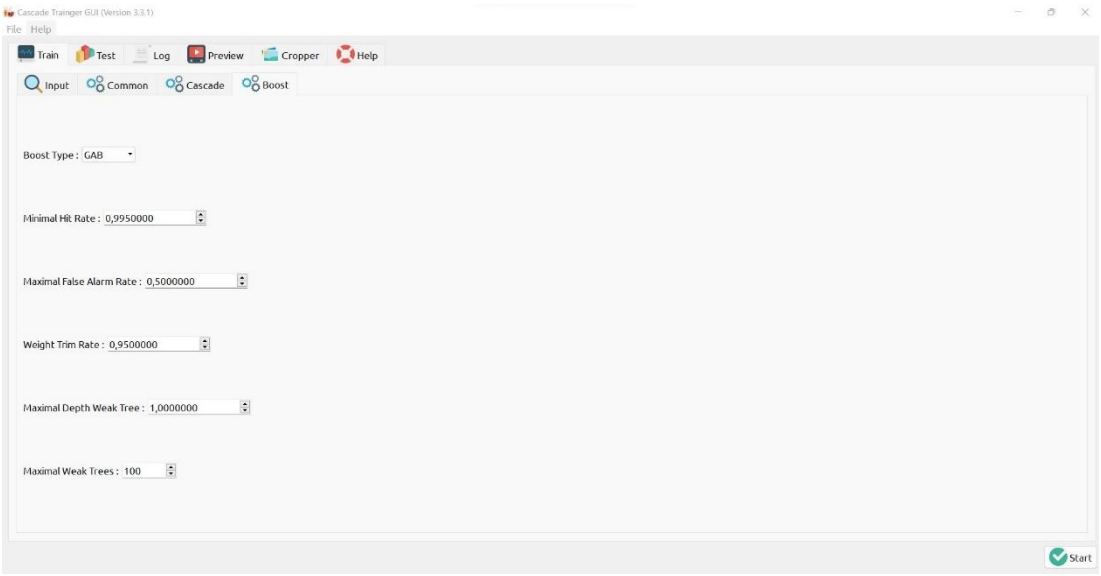


Şekil 4.5: Cascade Trainger GUI Cascade işlemleri

Daha sonra örnek genişliği ve yüksekliği ayarlanmıştır. Algılamayı çok yavaşlatacağı için görseller çok büyük bir boyuta ayarlanmamalıdır. Aslında bunun için her zaman küçük bir değer belirlemek oldukça güvenlidir. Örnek genişliği ve yüksekliği için önerilen ayarlar, bir yönü 24'te tutmak ve diğerini buna göre ayarlamaktır. Örneğin, 320×240 olan örnek resimler varsa, önce bu durumda 1,33:1 olan en boy oranı hesaplanır ardından büyük sayı 24 ile çarpılır. Örnek genişliği ve yüksekliği için 32×24 elde edilir. Projede kullanılacak görsellerin boyutu 640x640 'tır. Bu nedenle görsel boyutları 32x32 olarak elde edilmiştir.

Özellik türü HAAR, LBP veya HOG olarak belirlenebilir. Fakat HOG yalnızca OpenCV 3.1 veya sonraki sürümler kullanılıyorsa seçilmelidir. HAAR sınıflandırıcıları çok hassastır ancak eğitilmesi çok daha fazla zaman gerektirir, bu nedenle sınıflandırıcılarınıza birçok örnek görüntü sağlayabiliyorsanız LBP'yi kullanmak çok daha akıllıca olacaktır. Öte yandan, LBP sınıflandırıcıları daha az hassastır ancak çok daha hızlı çalışır ve neredeyse 3 kat daha hızlı tespit eder. Veri

setindeki görsellerin sınıflandırılması çok uzun sürmeyeceği düşünüldüğünden ve hassasiyeti göz ardı etmemek için HAAR sınıflandırıcısı seçilmiştir.

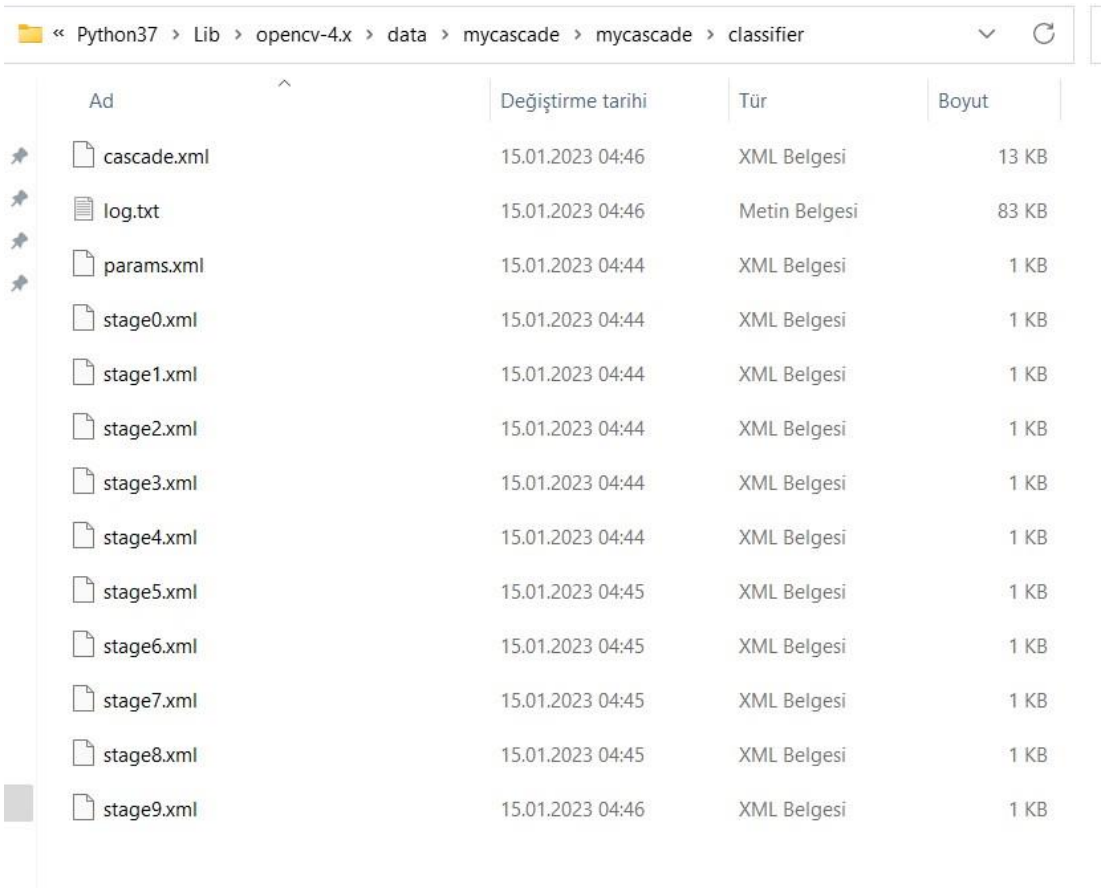


Şekil 4.6: Cascade Trainger GUI Boost işlemleri

Boost type olarak 4 farklı algoritma bulunmaktadır; DAB- Discrete AdaBoost, RAB- Real AdaBoost, LB– LogitBoost, GAB- Gentle AdaBoost. [9]

Real AdaBoost, Güven dereceli tahminleri kullanan ve kategorik verilerle iyi çalışan bir tekniktir. LogitBoost, İyi bir regresyon uyumu üretebilmektedir. Gentle AdaBoost, aykırı veri noktalarına daha az ağırlık verir ve bu nedenle regresyon verileriyle genellikle iyidir. Gentle AdaBoost ve Real AdaBoost genellikle tercih edilen seçenekler olarak karşılaşılmaktadır [10].

Tüm işlemler gerçekleştirildiktenden sonra sayfanın sağ alt köşesindeki “Start” butonuna basarak training işlemi gerçekleştirilmiştir. İşlem tamamlandığında “mycascade” klasörünün içinde “neg.lst” ve “pos.lst” adında iki MASM listing formatında dosya oluşturulur ve içerisinde görsellerin isimlerinin yer aldığı bir liste bulunmaktadır. Ayrıca “pos_samples.vec” dosyası ve “classifier” adlı bir klasör oluşturulmuştur. “classifier” klasörünün içeriği Şekil 4.7’ te gösterilmiştir.



Şekil 4.7: Classifier klasör içeriği

Bu klasör içindeki cascade.xml eğitilmiş veriyi barındırmaktadır. Bu .xml dosyası PyCharm'da yazılan kod içerisinde çağırılarak nesne tanıma işlemi sağlanacaktır. Önceden eğitilmiş bir sınıflandırıcı olan bu klasör nesnelerin nasıl tanınacağını içerir.

4.3 Kodlama

İlk olarak "import cv2" komutuyla OpenCV kütüphanesi sisteme eklenmiştir.

2.satırda "cap = cv2.VideoCapture(0)" komutu ile varsayılan kameradan video görüntülerini yakalamak için "cv2.VideoCapture(0)" fonksiyonu kameranın genişlik, yükseklik ve parlaklık ayarları için kullanılmıştır. '0' değeri varsayılan kamerayı ifade etmektedir.

3.satırda Şekil 4.2.5 'te görülmekte olan cascade.xml dosyasını çağırmak için "mycascade=cv2.CascadeClassifier("C:/Users/bgbet/AppData/Local/Programs/Python/Python37/Lib/opencv-4.x/data/mycascade/mycascade/classifier//cascade.xml")" komutu kullanılmıştır.

4.satırda “font1=cv2.FONT_HERSHEY_SIMPLEX” komutu ile algılanan nesnenin üstüne yazı yazmak için font tipi belirlenmiştir.

5-17.satırlar arasında bir while döngüsü kullanılmıştır.

6.satırdaki “ret,frame=cap.read()” komutu ile kameradan görüntü almak için cap.read() fonksiyonu kullanılmıştır.

7.satır “frame=cv2.flip(frame,1)” kod parçasında görüntüyü sağdan sola çevirmek için "cv2.flip()" fonksiyonu kullanılmıştır.

8.satırda “gray=cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)” kodu ile görüntünün gri tonlamasını almak için "cv2.cvtColor()" fonksiyonu kullanılmıştır.

9.satırda “kaktus=mycascade.detectMultiScale(gray,1.3,7)” koduyla eğitilmiş sınıflandırıcıyı kullanarak nesneleri tespit etmek için "mycascade.detectMultiScale()" fonksiyonu kullanılır.

10.satırda tespit edilen nesneler için bir for döngüsü başlatılmaktadır.

11.satırda tespit edilen kaktüs nesnesinin çerçevesini çizmek için "cv2.rectangle()" fonksiyonu kullanılmıştır.

12.satırda tespit edilen nesnenin üstüne "KAKTUS" yazısı yazdırmak için "cv2.putText()" fonksiyonu kullanılmıştır.

14.satırda güncellenen görüntüyü ekranda göstermek için "cv2.imshow()" fonksiyonu kullanılmıştır.

16.satırda kullanıcının "q" tuşuna basıp basmadığını kontrol etmek için "cv2.waitKey()" fonksiyonu kullanılmıştır.

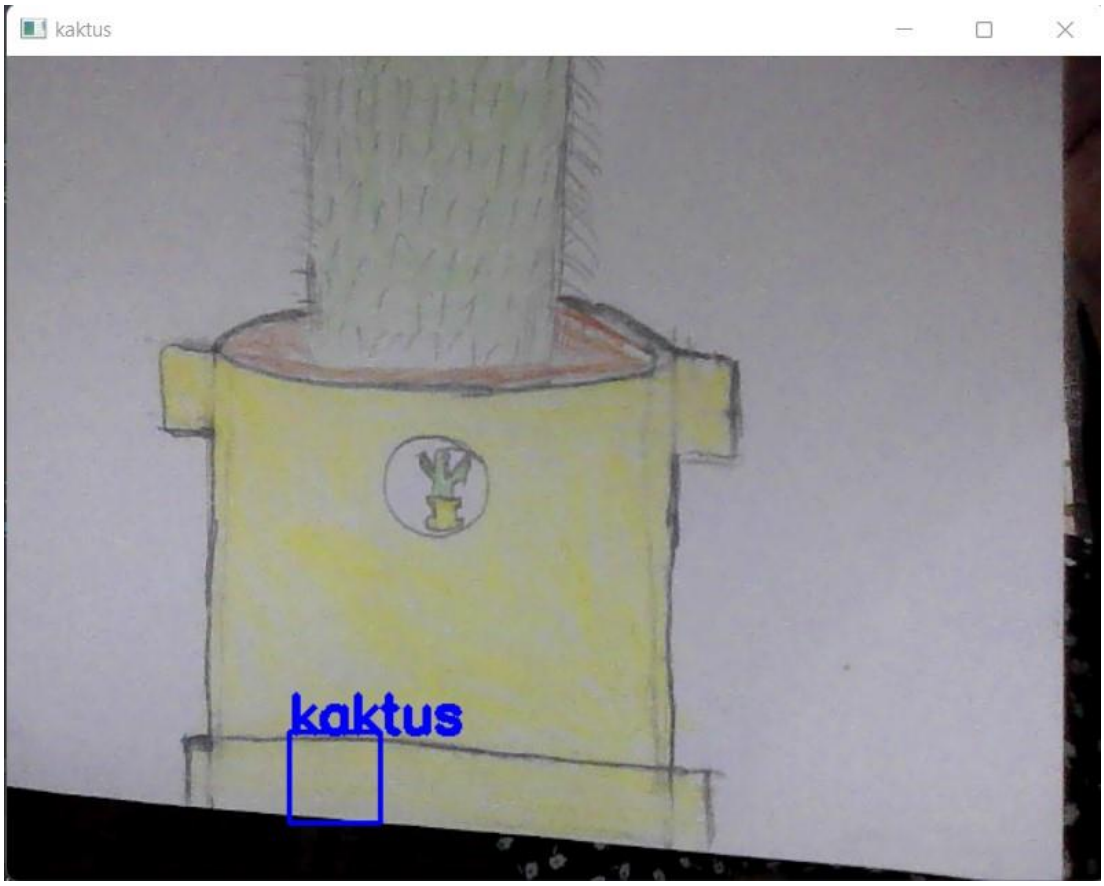
17.satır döngüyü sonlandırır.

18.satırda kullanıcı "q" tuşuna bastığında kamerayı kapatmak için "cap.release()" kullanılmıştır. "if cv2.waitKey(1) & 0xFF==ord("q"):" ifadesi kullanıcının klavyede "q" tuşuna basıp basmadığını kontrol eder. Eğer kullanıcı "q" tuşuna basarsa kodun çalışması sonlandırılır.

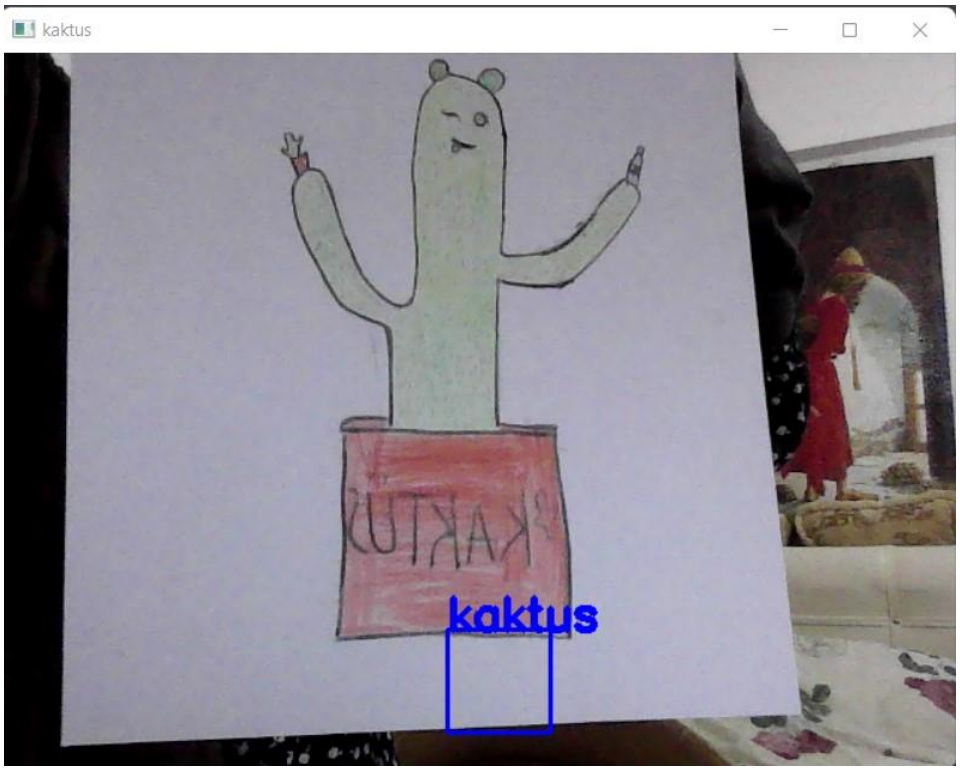
19.satırda tüm açık olan pencereleri kapatmak için "cv2.destroyAllWindows()" komutu kullanılır.

4.4 Sonuç

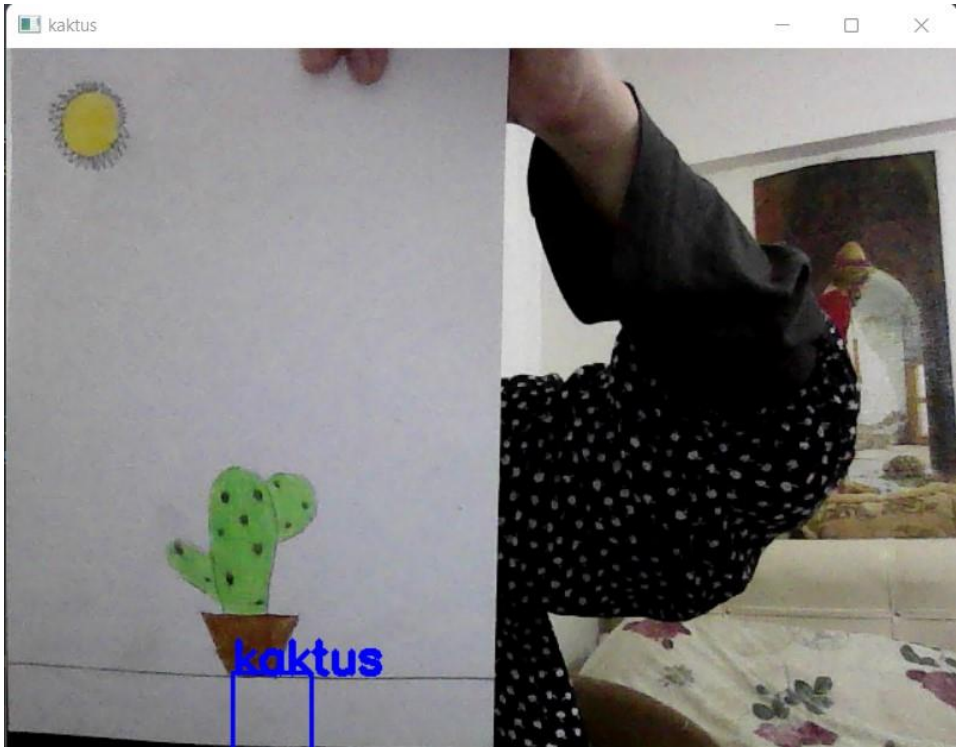
Kodlar çalıştırılıp bir kaktüs çizimi ekrana gösterildiğinde çıkan sonuçlar Şekil 4.8, Şekil 4.9, Şekil 4.10'te görülmektedir:



Şekil 4.8: Kaktüs nesnesinin tespiti



Şekil 4.9: Kaktüs nesnesinin tespiti



Şekil 4.10: Kaktüs nesnesinin tespiti

Kaynaklar

[1] Daş R, Polat B, Tuna G, "Derin Öğrenme ile Resim ve Videolarda Nesnelerin Tanınması ve Takibi", Fırat Üniversitesi Mühendislik Bilimleri Dergisi, Eylül 2019: 571-581, doi:10.35234/fumbd.608778.

[2] Atalay M, Çelik E, "Büyük Veri Analizinde Yapay Zekâ Ve Makine Öğrenmesi Uygulamaları", Mehmet Akif Ersoy Üniversitesi Sosyal Bilimler Enstitüsü Dergisi, 2017,9 (22) : 155-172 . DOI: 10.20875/makusobed.309727.

[3] Öztemel E. Yapay Sinir Ağları, 2.baskı. Papatya Yayıncılık; 2006

[4] Yılmaz A,Kaya U,Soylu İ.(Ed.) Derin Öğrenme, 4th ed.; 2022.

[5] Etiya Marketing. Makine Öğrenmesi ve Derin Öğrenme Nedir? [İnternet]. İstanbul; 2022 [erişim tarihi 13.01.2023]. https://www.etiya.com/tr/blog/makine-ogrenmesi-machine-learning-ve-derin-ogrenme-nedir?gclid=Cj0KCQiA_P6dBhD1ARIsAAGI7HDUOn9lF2rO1GROYqlPFtihc5jfwBiBISgkTrmYMPVnDiGj8F-Y0KoaAv82EALw_wcB

[6] Burgaz, M. Derin Öğrenme Algoritmalari Kullanarak İnsansiz Hava Araçlari İle Silah Tespiti (yüksek lisans tezi). Batman: Batman Üniversitesi; 2020. https://tez.yok.gov.tr/

[7] Mesut Pişkin Blog. OpenCV Nedir? [İnternet].[erişim tarihi 15.01.2023] https://mesutpiskin.com/blog/opencv-nedir.html

[8] Medium. Kadir Güzel, Boosting Nedir? Adım Adım AdaBoost Algoritması[İnternet]. 2020 [erişim tarihi 15.01.2023] https://kadirguzel.medium.com/boosting-nedir-ad%C4%B1m-ad%C4%B1m-adaboost-algoritmas%C4%B1-439cce20ab9a

[9] OpenCV. Cascade Classifier Training. [İnternet] .[erişim tarihi 15.01.2023] https://docs.opencv.org/4.x/dc/d88/tutorial_traincascade.html

[10] Stackoverflow. Opencv haartraining.[İnternet]. [erişim tarihi 15.01.2023] https://stackoverflow.com/questions/15519066/opencv-haartraining

[11] DÜMF Mühendislik Dergisi 10:2 (2019) : 409-445

[12] SAKARYA UNIVERSITY JOURNAL OF COMPUTER AND INFORMATION SCIENCES Derin Öğrenme Algoritmalarını Kullanarak Görüntüden Cinsiyet Tahmini

Ekler

Ek A

```
import cv2

cap = cv2.VideoCapture(0)

mycascade=cv2.CascadeClassifier("C:/Users/bgbet/AppData/Local/Programs/Python
/Python37/Lib/opencv-4.x/data/mycascade/mycascade/classifier//cascade.xml")

font1=cv2.FONT_HERSHEY_SIMPLEX

while True:

    ret,frame=cap.read()

    frame=cv2.flip(frame,1)

    gray=cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)

    kaktus=mycascade.detectMultiScale(gray,1.3,7)

    for (x,y,w,h) in kaktus:

        cv2.rectangle(frame,(x,y),(x+w,y+h),(255,0,0),2)

        cv2.putText(frame,"kaktus",(x,y),font1,1,(255,0,0),cv2.LINE_4)

    cv2.imshow("kaktus",frame)

    if cv2.waitKey(1) &  0xFF==ord("q"):

        break

cap.release()

cv2.destroyAllWindows()
```

Özgeçmiş

Adı Soyadı:	Betül GÜLTER
E-mail (1):	y210240014@ogr.ikcu.edu.tr
E-mail (2):	betullgulter@gmail.com

Eğitim:

2021–2023	İzmir Kâtip Çelebi Üniversitesi, Yazılım Müh. Bölümü
2020-2022	Anadolu Üniversitesi, Web Tasarımı ve Kodlama Bölümü
2016–2020	İzmir Dokuz Eylül Üniversitesi, Şehir ve Bölge Planlama Bölümü

İş Deneyimi:

2019 – 2020	Manisa Yunusemre Belediyesi Stajyer Şehir Plancısı