KARADENİZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

MASK R-CNN ALGORİTMASI İLE ASKERİ KAMUFLAJ SINIFLANDIRMASI

YÜKSEK LİSANS TEZİ

İlkay KARATEPE

NİSAN 2023
TRABZON

KARADENİZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

MASK R-CNN ALGORİTMASI İLE ASKERİ KAMUFLAJ SINIFLANDIRMASI

İlkay KARATEPE
ORCID : - - -

Karadeniz Teknik Üniversitesi Fen Bilimleri Enstitüsünce
"BİLGİSAYAR YÜKSEK MÜHENDİSİ"
Unvanı Verilmesi İçin Kabul Edilen Tezdir.

Tezin Enstitüye Verildiği Tarih : 09 / 02 /2023
Tezin Savunma Tarihi : 12 / 04 /2023

Tez Danışmanı : Prof. Dr. Vasif NABİYEV
ORCID : - - -

Trabzon 2023

ÖNSÖZ

Bu tez, Karadeniz Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği Anabilim Dalı, Bilgisayar Bilimleri Yüksek Lisans Programı'nda yapılan bir çalışmadır. ''Mask R-CNN Algoritması ile Askeri Kamuflaj Sınıflandırması'' adlı çalışmada, 7 farklı ülkeye ait kamuflajlı asker görüntülerinin sınıflandırılması ve segmentasyonu yapılmaya çalışılmıştır.

Bu tez çalışması boyunca gerek konu seçimi ve gerekse çalışmaların yürütülmesi sırasında bilgi ve deneyimlerini bana aktaran, danışman hocam Prof. Dr. Vasif NABİYEV'e ilgi, destek ve tecrübelerini esirgemediğinden dolayı teşekkür ediyorum.

Hayatım boyunca maddi ve manevi olarak yanımda olan annem Zübeyde KARATEPE'ye ve babam Adem KARATEPE'ye, çalışmanın her safhasında beni motive eden fedakâr eşim Seda KARATEPE'ye teşekkür ederim.

İlkay KARATEPE

Trabzon 2023

TEZ ETİK BEYANNAMESİ

Yüksek Lisans Tezi olarak sunduğum “Mask R-CNN Algoritması ile Askeri Kamuflaj Sınıflandırması” başlıklı bu çalışmayı baştan sona kadar danışmanım Prof. Dr. Vasif NABİYEV’in sorumluluğunda tamamladığımı, verileri/örnekleri kendim topladığımı, çalışmaları kendim yaptığımı, başka kaynaklardan aldığım bilgileri metinde ve kaynakçada eksiksiz olarak gösterdiğimi, çalışma sürecinde bilimsel araştırma ve etik kurallara uygun olarak davrandığımı ve aksinin ortaya çıkması durumunda her türlü yasal sonucu kabul ettiğimi beyan ederim. 12/04/2023

İlkay KARATEPE

İÇİNDEKİLER

Sayfa No

Yüksek Lisans Tezi

ÖZET

Mask R-CNN Algoritması ile Askeri Kamuflaj Sınıflandırması

İlkay KARATEPE

Karadeniz Teknik Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı
Danışman: Prof. Dr.Vasif NABİYEV
2023, 68 Sayfa, 7 Ek Sayfalar

Doğada canlıların bir gizlenme sanatı olarak kullandığı kamuflaj 19. yüzyılda askeri alanda kullanılmaya başlanmıştır. Farklı uluslar, çevre, iklim gibi etmenler düşünüldüğünde çeşitli renk ve desende kamuflajlar karşımıza çıkmaktadır. Kamuflajlı alanın arka planla benzerliği bölütlemeyi zorlaştırmakla birlikte her bir kamuflaj deseninin kumaşın kesiminden ve desen parçalarının her askerde farklı yerlerde olmasından dolayı sınıflandırılması zorlaşmaktadır. Literatürde kamuflaj veya desen sınıflandırması olarak geçen farklı çalışmalar vardır. Bahsi geçen çalışmalar kamufle olmuş nesnenin bölütlenmesi veya kamufle olmuş farklı türden nesnelerin sınıflandırılması şeklindedir. Bu çalışmada bölütlenen ve sınıflandırılan nesneler kamuflajlı askerler olduğu için derin öğrenme algoritmasından beklenen nesnelerin ana hatlarına göre değil temel olarak kamuflaj desenine göre sınıflandırmasıdır. Bu çalışmada 7 ülke için 1233 kamuflajlı asker görüntüsü toplanmış ve günümüzde nesne tespiti, bölütlemesi ve sınıflandırılması için yaygın olarak kullanılan Mask R-CNN algoritmasıyla askeri kamuflaj sınıflandırma problemi ele alınmış ve CNN algoritmalarının önemi böyle zor bir problemle ispatlanmıştır.

Anahtar Kelimeler: Kamuflaj, Derin Öğrenme, Mask R-CNN, Sınıflandırma, Segmentasyon

Master Thesis

SUMMARY

Military Camouflage Classification with Mask R-CNN Algorithm

İlkay KARATEPE

Karadeniz Technical University
The Graduate School of Natural and Applied Sciences
Computer Engineering Graduate Program
Supervisor: Prof. Dr.Vasif NABİYEV
2023, 68 Pages, 7 Inset Pages

Camouflage, which is used as an art of hiding by living things in nature, started to be used in the military field in the 19th century. When factors such as different nations, environment and climate are considered, we come across camouflages in various colors and patterns. While the similarity of the camouflaged area with the background makes segmentation difficult, it becomes difficult to classify each camouflage pattern due to the cut of the fabric and the different locations of the pattern pieces on each soldier. There are different studies in the literature that are referred to as camouflage or pattern classification. The mentioned studies are in the form of segmentation of camouflaged object or classification of camouflaged objects of different types. Since the segmented and classified objects in this study are camouflaged soldiers, what is expected from the deep learning algorithm is to classify the objects mainly according to the camouflage pattern, not their outlines. In this study, 1233 images of soldiers in camouflage were collected for 5 countries and the military camouflage classification problem was solved with the Mask R-CNN algorithm, which is widely used today for object detection, segmentation and classification, and the importance of CNN algorithms was proved with such a difficult problem.

Key Words: Camouflage, Deep Learning, Mask R-CNN, Classification, Segmentation

ŞEKİLLER DİZİNİ

TABLOLAR DİZİNİ

SEMBOLLER DİZİNİ

YSA	: Yapay Sinir Ağları
ANN	: Artificial Neural Network-ANN
ESA	: Evrişimli Sinir Ağları
X-OR	: Exclusive-OR
CNN	: Convolutional Neural Network
LBP	: Local Binary Patterns
SVM	: Destek Vektör Makineleri
KNN	: K-en Yakın Komşuluk Algoritması
RPN	: Bölge Öneri Ağı
HOG	: Histogram of Oriented Gradient
ResNets	: Artık Ağlar
SSD	: Single Shot MultiBox Detector
SPP-net	: Spatial Pyramid Pooling
DPM	: Disruptive Pattern Material
AI	: Artificial Intelligence
MLP	: Multi Layer Perceptron
MRI	: Magnetic Resonance Image
Adaline	: Adaptive Linear Neuron
Madaline	: Multi-layer Adaptive Linear Neuron
MLP	: Multi Layer Perceptron
MSE	: Mean Squared Error
RMSE	: Root Mean Square Error
MAE	: Mean Absolute Error
RPN	: Region Proposal Network
lr	: learning rate

1. GENEL BİLGİLER

1.1. Giriş

1800'lerin ortasına kadar ordular askerlerine parlak kıyafetler giydiriyordu [1]. Daha isabetli, uzun menzilli ateşli silahların gelişmesiyle askeri alanda ilk kamuflaj 1848'de İngiliz Hint Ordusu'nun bir alayı tarafından sarımsı donuk üniformalar olarak kullanıldı [2]. 19. Yüzyılın sonlarına doğru İngiliz ordusu tarafından düz hâkî renkli üniformalar benimsendi [3]. Bunun yanında Birinci Dünya Savaşı'nda Alman ordusunda 'Fragmentation Camouflage' adını verdikleri farklı renklerden oluşan desenli üniformalar kullanılmıştır [3]. Askeri operasyonların yapıldığı bölgeler (çöl, deniz, orman…), dönemin iklim şartları gibi faktörler kamuflaj desenlerinin çeşitliliğini artırmıştır. Kamuflaj desenlerinin tasarımı 20. yüzyılın başlarında tasarımcılar ve sanatçıların elindeyken 1970'lerin sonunda bilgisayar teknolojisi kamuflaj alanında kullanılmaya başlanmıştır [3]. Günümüzde birçok ülke farklı kamuflaj desenlerini benimsemiş ve kullandığı kamuflaj desenleri ulusal kimlik haline gelmiştir. Ülkelerin askeri ittifakları, ticari ilişkileri, ortak sınırlardan doğan stratejiler ise bazı kamuflaj ailelerinin doğmasına sebep olmuştur. Bu ilişkilerden biri de Azerbaycan-Türkiye'dir. Türkiye ve Azerbaycan ortak askeri çalışmalardan dolayı kamuflajları benzerdir. Bu sebeple bu tez çalışmasında bir sınıf olarak tercih edilmiştir. Bunun yanı sıra Almanların "Flecktarn" isimli kamuflajını, Danimarka, Japonya, Polonya, Çin ve Belçika gibi ülkeler farklı türevlerini kullanmışlardır [4] ve yine bu tez çalışmasında Almanya ve Çin'in kullandıkları kamuflaj paternleri birbirine çok yakın olanlar seçilmiştir (farklı sınıflar olarak). Bu tez çalışmasında tercih edilen ülkeler kamuflaj benzerliklerine göre tercih edilmiştir. Buradaki amaç benzer kamuflaj ailelerine sahip ülkelerin kamuflaj sınıflandırılmasının daha zor olacağı sebebiyle derin öğrenme algoritmasının işini zorlaştırmaktır.

Kamuflajların tarihi gelişimine baktığımızda özellikle askeri anlamda önemli bir taktik sağladığı görülmektedir. Kamuflajlı bir askerin insan gözü ile seçilmesi zor olmaktadır. Öte yandan tespit edilen bir askerin hangi ülkenin askeri olduğunu bilmek operasyonel anlamda büyük kazanımlar sağlayacaktır. Özelikle günümüzde İHA, SİHA ve drone gibi teknolojilerin kullanımı ile anlık, insan gözünden kaçan kamuflajlı unsurlar rahatça tespit edilip gerekli önlemler alınabilir. Çalışmanın askeri yöndeki getirilerine ek olarak medyada

bir başka ülkenin askerinin farklı bir ülke askeri olarak servis edildiği sahte içerikler ile dezenformasyon yapılmaktadır. Bu sebeple bu çalışma bu tip sahteliklerin önlenmesi için de bir çözüm olabilir. Derin öğrenmenin günümüzde birçok kullanım alanı mevcuttur. Kamuflaj deseni sınıflandırması gibi bir çalışma derin öğrenmenin kullanım alanlarına önemli bir getirisi olabilir.

Kullanılan kamuflajlar belirli örüntülere sahip olsalar bile bu örüntülerin sınıflandırma için matematiksel modelini çıkarmak oldukça zordur. Bir nesnenin tespiti, bölütlenmesi veya sınıflandırılması için görüntü işleme yaklaşımları olmakla birlikte derin öğrenme modellerinin evrimi ile bu problemler daha doğru ve hızlı yapılabilmektedir.

1.2. Tezin Amacı

Günümüzde derin öğrenme hayatın birçok alanına girmektedir. Derin öğrenme sayesinde geçmişte yapılan birtakım işler (nesne yakalama, nesne tanıma, segmentasyon vb.) daha doğru ve hızlı çözülebilmektedir. Bunun yanı sıra klasik yöntemlerle çözülemeyen problemler derin öğrenme yaklaşımlarıyla çözülebilmektedir. Yapay sinir ağlarının (YSA) çalışma mantığı gereği problemin boyutu azaltılabilir, artırılabilir veya problem farklı uzaya taşınarak çözüm aranabilir. Bu sayede geçmişte doğrusal olmayan problemler (XOR problemi vb.) çözülebilmiştir. Problemlerin matematiksel karmaşası artıkça daha derin YSA modelleri geliştirilmiştir. Bu derin ağların bir türü de bilgisayarlı görü alanında gelişen evrişimli sinir ağlarıdır (ESA). ESA'lar sayesinde günümüzde nesne tespit, tanıma, segmentasyon gibi problemler çözülebilmektedir. Bu tez çalışmasında derin öğrenmenin gücünü göstermek amacıyla askeri kamuflaj segmentasyonu ve sınıflandırılması yapılmıştır. Problemin seçimindeki amaç nesneler segmente edildikten sonra sınıflandırma için kamuflaj paternlerinin bir kriter olarak derin öğrenme tarafından öğrenilerek kullanmasıdır. Çalışma sayesinde derin öğrenme, karmaşası yüksek bir problem için denenmiş ve başarısı gösterilmeye çalışılmıştır.

1.3. Literatür Taraması

Literatürde kamuflaj veya desen sınıflandırılmasına yönelik farklı çalışmalar yapılmıştır. Doğan vd. yaprak desenlerini sınıflandırmak üzerine AlexNet, Vgg16, Vgg19, ResNet50 ve GoogleNet gibi farklı derin öğrenme algoritmalarını karşılaştırdılar [5]. Gupta

vd. özelleştirdikleri 58 katmanlı Convolutional Neural Network (CNN) modeli ile 22 sınıftan oluşan sisli, karlı, gece görüşlü çeşitli asker ve ekipmanlarını algılayan çalışma yaptılar [6]. Bayram vd. kamuflaj görüntüler için Local Binary Patterns (LBP) kullanarak dokusal öznitelikleri çıkarmış Yapay Sinir Ağları (YSA), Destek Vektör Makineleri (SVM) ve K-en Yakın Komşuluk Algoritması (KNN) ile sınıflandırma yapmıştır [7]. Karatepe vd. 5 sınıf için askeri kamuflaj sınıflandırması yapmıştır [8]. Yaptıkları çalışmada bölgesel tabanlı nesne tespit, bölütleme ve segmentasyon için geliştirilmiş olan Mask R-CNN algoritması [9] 5 farklı kamuflaj ailesine ait askeri kamuflajlı asker görüntülerinin segmentasyonu ve sınıflandırılması için kullanılmıştır. Ömeroğlu vd. Mask R-CNN algoritmasıyla yüksek çözünürlüklü uydu görüntülerinde hangar tespitini %85 ortalama kesinlik ile yapmışlardır [10]. Amri vd. multispektral görüntüler üzerinde stadyum tespiti için YOLOv5 [11] ve Mask R-CNN kullanmıştır [12].

1.3.1. Nesne Tespit Yaklaşımları

Bilgisayarlı görü alanında giderek yaygınlaşan birçok yöntem vardır. 2015 yılında Fast R-CNN [13] Ross Girshick tarafından önerilmiştir. Önceki çalışmaları olan R-CNN [14]'i temel alınarak geliştirilmiştir. R-CNN ile kıyaslandığında Fast R-CNN, algılama doğruluğunu artırırken eğitim ve test hızını iyileştirmek için çeşitli yenilikler kullanır. Daha sonra Ross Girshick'in de içinde bulunduğu bir grup tarafından Faster R-CNN [15] geliştirilmiştir. Burada yazarlar bölge önerilerinin algılanması için kullanılan çalışma süresini azaltmak için "Bölge Öneri Ağını (RPN)" önerdiler. Bölge tabanlı yaklaşımların dışında Dalal ve Triggs SVM tabanlı yönlendirilmiş gradyanların histogramlarını (HOG) kullanarak insan algılama üzerine çalışma yaptılar [16]. Dai vd. nesne tespiti için tamamen evrişimli ağ olan R-FCN [17]'yi önerdiler. R-FCN, R-CNN grubu gibi bölge tabanlı bir yaklaşımdır. Onu diğerlerinden ayıran özellik konuma-duyarlı skor haritaları oluşturmasıdır. Ayrıca R-FCN, "Artık Ağlar (ResNets)" [18] gibi tamamen evrişimli görüntü sınıflandırıcılarını omurgasında kullanır. Liu vd., R-CNN'de olan nesnelerin bölgesinin tespiti ve ardından sınıflandırılması aşamalarını tek adıma indirdikleri "Single Shot MultiBox Detector (SSD) [19]" yöntemini geliştirdiler. Kaiming He ve arkadaşları, mevcut evrişimli sinir ağlarında olan sabit boyutlu giriş görüntüsü problemi için "Spatial Pyramid Pooling (SPP-net) [20]" ağını önerdiler. Bunun için havuzlama stratejisi olan "uzaysal piramit havuzlama" ismini verdikleri yaklaşımı kullandılar. Ayrıca SPP-net, nesne

deformasyonlarına karşı da dayanıklıdır. Günümüzde mevcut CNN yaklaşımlarından oldukça hızlı olan YOLO [11] ise görüntüyü ızgaralara bölerek her bölgeye aynı anda bakar YOLO'nun yazarları bu yönteme "one shot" ismini vermekteler.

1.3.2. Segmentasyon-Bölütleme Yaklaşımları

Nesne tespiti için nesnenin sınırlarının bilinmesi yeterliyken segmentasyonda nesnenin arka plan ile tamamen ayrışması beklenmektedir. Segmentasyon algoritmalarının çıktısı bir maske veya matrixtir. Bu doğrultuda literatürde konusuna göre birçok çalışma mevcuttur. Özellikle tıp ve endüstride uygulamaları oldukça yaygındır. Segmentasyon için derin öğrenmenin yanı sıra sezgisel yöntemlerde mevcuttur. Bu yöntemler; renk, kontrast, histogram ve kenar gibi görüntü bilgilerine göre tasarlanan yaklaşımlardır. Bu tür sezgisel yöntemlere dayalı geleneksel görüntü bölütleme teknikleri hızlı ve basit olabilir, ancak bunlar genellikle manuel olarak tasarlanmış belirli kullanım durumlarını desteklemek için önemli ölçüde ince ayar gerektirir. Kamuflajlı bir alanı bölütlemek gibi karmaşık görüntüler için kullanmak her zaman yeterince doğru sonuçlar vermeyebilir.

1.3.2.1. Kenar Tabanlı Segmentasyon

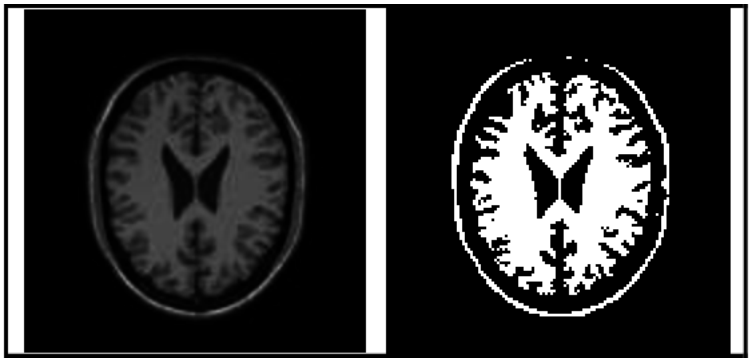
Kenar tabanlı segmentasyon, belirli bir görüntüdeki çeşitli nesnelerin kenarlarını tanımlayan yaygın bir görüntü işleme tekniğidir. Kenar bilgilerini kullanarak görüntüdeki ilişkili nesnelerin özelliklerinin bulunmasına yardımcı olur. Kenar algılama, gereksiz bilgilerin görüntülerinden çıkarılmasına yardımcı olur, boyutlarını küçültür ve analizi kolaylaştırır. Kenar tabanlı segmentasyon algoritmaları, kenarları kontrast, doku, renk ve doygunluk parametrelerine göre tanımlar. Ayrı kenarları içeren kenar zincirlerini kullanarak bir görüntüdeki nesnelerin sınırlarını doğru bir şekilde bulmaya çalışırlar. Kenar tabanlı bölütleme için örnek Şekil 1'de verilmiştir. Chu vd. bölge ve kenar tabanlı segmentasyon [21] önermişlerdir. Burada girdi olarak hem bölge hem de kenar bölütleme haritalarını kullanarak bölütleme haritalarını entegre eden bir algoritma sunulmuştur. Entegrasyonun sonucu, her bölgenin büyük ve kompakt olduğu bir bölge haritasıdır. Sahar Zafari [22]'de örtüşen dışbükey nesnelerin segmentasyonu için bir çalışma sunar. Sunduğu bu çalışma çekirdek noktası çıkarımı, kontur iz çıkarımı ve kontur tahmini aşamalarının ele alındığı üç bileşenden oluşan bir çerçeveye dayanmaktadır. Aydın vd. [23]' de kenar tabanlı düzey

kümeleriyle tıbbi görüntüler için segmentasyon çalışması yaparak istemci-sunucu temelli bir uygulama geliştirmişlerdir. Çalışmalarında canny kenar bulma tekniğini kullanmışlardır.

Şekil 1. Kenar tabanlı segmentasyon örneği [22].

1.3.2.2. Eşik Tabanlı Segmentasyon

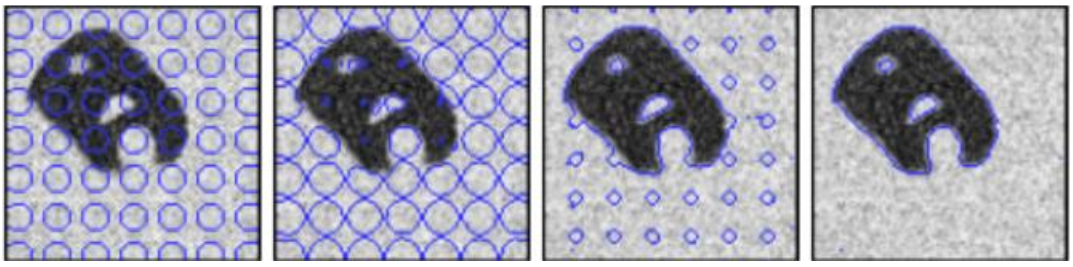
Eşik Tabanlı Segmentasyon, pikselleri belirli bir eşik değeri için yoğunluklarına göre bölen en basit görüntü bölümleme yöntemidir. Diğer nesnelerden veya arka planlardan daha yüksek yoğunluğa sahip nesneleri segmentlere ayırmak için uygundur. Düşük gürültülü görüntülerde sabit olarak çalışabilir. Bazı durumlarda, dinamik eşikler kullanmak mümkündür. Eşikleme, gri tonlamalı bir görüntüyü eşik değeri ile ilişkilerine göre iki parçaya bölerek ikili bir görüntü oluşturur. Şekil 2'de bölütleme örneği verilmiştir. Bu örnek Afifi vd. tarafından yapılan manyetik rezonans görüntüleri (magnetic resonance image-MRI) için eşik tabanlı görüntü bölütleme çalışmasından [24] alınmıştır. Önerdikleri yöntemde daha ideal eşik değerleri elde etmek için yerel arama yöntemi ve bölge büyütmeyle eşikleme yöntemlerini birleştirerek kullanılır.



Şekil 2. Eşik tabanlı bölütleme örneği [24].

1.3.2.3. Bölge Tabanlı Segmentasyon

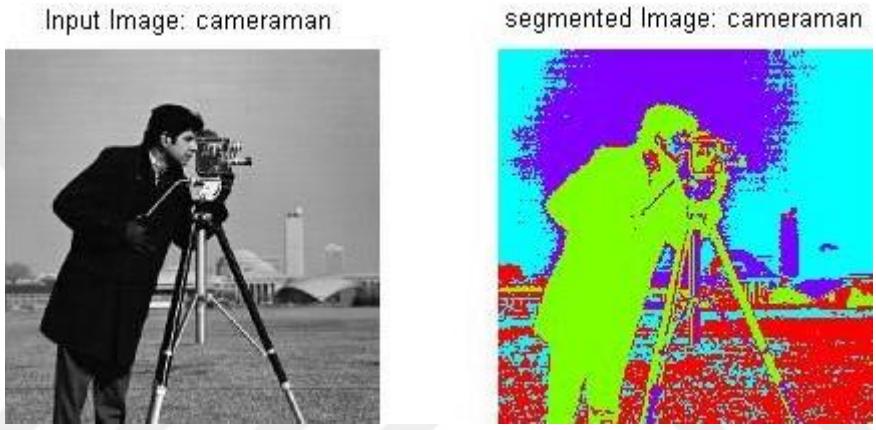
Kenar tabanlı bölütleme modellerinde nesne sınırlarını belirlemek için görüntü gradyanları kullanıldığından ve görüntü içeriğindeki gürültü de yüksek gradyan içerebileceğinden dolayı sorunlar çıkabilmektedir. Bu sebeple Zhu ve Yuille'in [25] çalışmasından yola çıkılarak bölge tabanlı akışlara yaklaşımlar geliştirilmiştir. Bölge tabanlı segmentasyon, bir görüntüyü benzer özelliklere sahip bölgelere ayırmayı amaçlar. Her bölge, algoritmanın bir pivot noktası aracılığıyla bulduğu bir piksel grubudur. Algoritma pivot noktalarını bulduğunda, daha fazla piksel ekleyerek veya küçülterek ve diğer noktalarla birleştirerek bölgeleri büyütebilir. Yezzi vd. [26]'da bölge tabanlı segmentasyon önermişlerdir. Önerdikleri yöntemde görüntüdeki görüntünün tamamından elde edilen istatistiğe göre göre bölge seçimleri yapılır. Yezzi'nin çalışmasında bölütlenecek nesnelerin yoğunluğu sabit kabul edilir.



Şekil 3. Bölge tabanlı bölütleme örneği [27].

1.3.2.4. Küme Tabanlı Segmentasyon

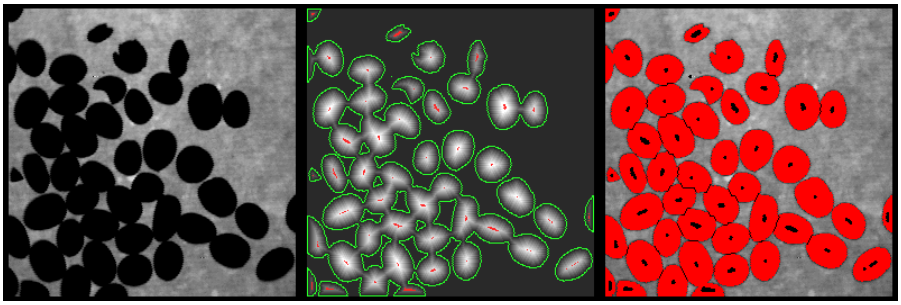
Kümeleme algoritmaları, görüntülerdeki gizli bilgileri tanımlamaya yardımcı olan denetimsiz sınıflandırma algoritmalarıdır. Algoritma, görüntüleri benzer özelliklere sahip piksel kümelerini veri öğeleriyle birlikte ayırır ve benzer öğeleri kümeler halinde gruplandırır. Zhou vd. [28], cilt lezyonlarını bölütlemek için fuzzy c-means [29] yöntemini kullanarak küme tabanlı bir segmentasyon önerisinde bulundular. Şekil 4'te küme tabanlı bölütleme örneği verilmiştir.



Şekil 4. Küme tabanlı bölütleme örneği [30].

1.3.2.5. Watershed Segmentasyonu

Bu segmentasyon yaklaşımında gri görüntüler kullanılır. Her nesne kendi içinde daha homojen bir yapı göstereceğinden bu bölgelerin arası topografik harita gibi işaretlenerek bölütleme yapılır [31]. Watershed segmentasyonu ile ilgili örnek şekil 5'te verilmiştir.

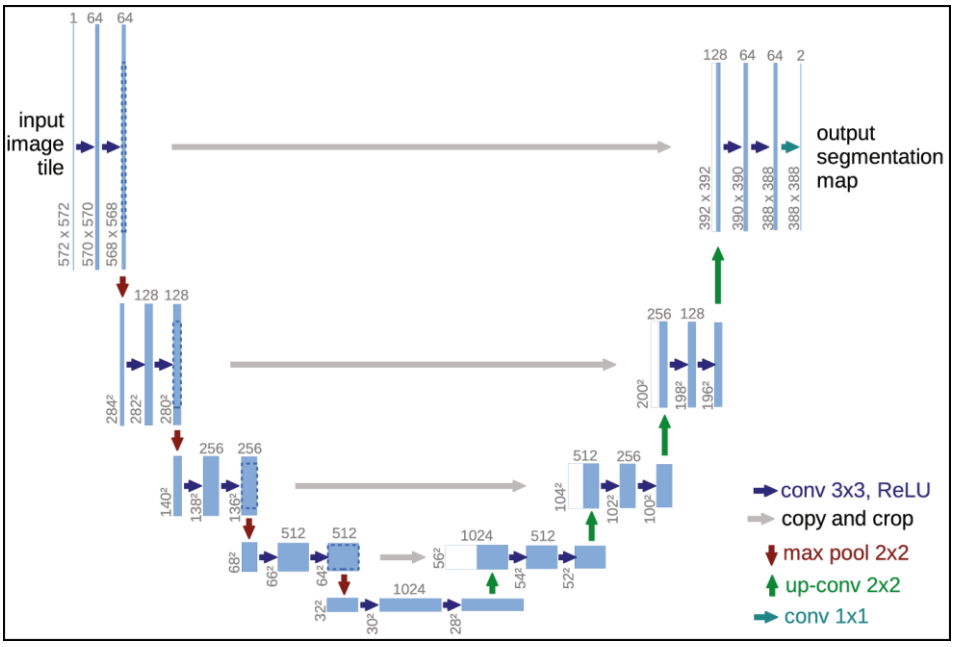


Şekil 5. Watershed segmentasyonu [31].

1.3.2.6. Makine Öğrenimi Tabanlı Görüntü Segmentasyonu

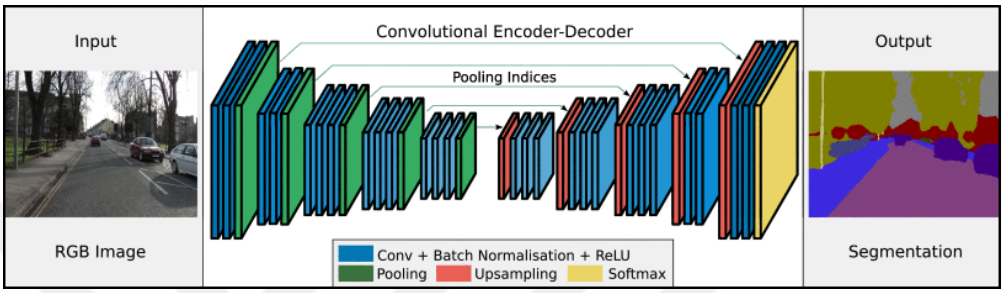
Daha yeni segmentasyon teknikleri, doğruluğu ve esnekliği artırmak için makine öğrenimini ve derin öğrenmeyi kullanır. Makine öğrenimi tabanlı görüntü bölümleme yaklaşımları, programın önemli özellikleri belirleme yeteneğini geliştirmek için model eğitimini kullanır. Derin sinir ağı teknolojisi, özellikle görüntü bölümleme görevleri için etkilidir. Görüntü segmentasyonu için çeşitli sinir ağı tasarımları ve uygulamaları literatürde mevcuttur.

Ronneberger vd. biyomedikal görüntü segmentasyonu için U-Net [32]'i geliştirdiler. U-Net, geniş bir görüntü veri kümesini kullanarak görüntüdeki nesneleri veya bölgeleri ayırmayı amaçlar. U-Net, yapısal bir benzerliğe dayalı bir arayüz (encoder) ve düzgün bir görüntü çıktısı (decoder) üretmeyi amaçlayan iki farklı yapıdan oluşur. Encoder, görüntüdeki bilgiyi sıkıştırır ve düzenler, decoder ise sıkıştırılmış bilgiyi ayrıntılı bir görüntü olarak çıkarır. U-Net, görüntü segmentasyonu için etkili bir yaklaşımdır ve görüntü veri kümelerinin fazla olması durumunda daha iyi sonuçlar verir. Ayrıca, görüntü verilerinin büyüklüğünde de azaltma yapmasına rağmen, görüntüdeki ayrıntıların kaybını minimize etmeyi amaçlar. U-Net, biomedikal görüntüler gibi zor görüntü verileri için de kullanılabilir. Şekil 6'da U-Net mimarisi verilmiştir.



Şekil 6. U-net mimarisi [32].

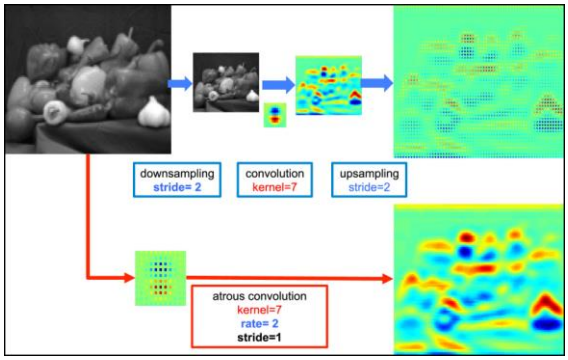
Badrinarayanan vd., semantik piksel bazında bölümleme için tamamen evrişimli bir sinir ağı mimarisi olan SegNet [33]'i geliştirmişlerdir. SegNet, girdi görüntülerin her pikselini belirli bir nesne veya arka plan sınıfına atamaya çalışır. Bunu yapmak için, SegNet'de max-pooling işlemlerinin geri beslemesi (unpooling) yapılır, bu da girdi görüntüdeki her pikselin orijinal yerini korur. Sonuç olarak, SegNet, belirgin ve ayrıntılı görüntü segmentasyonları üretebilir. Şekil 7'de SegNet mimarisi verilmiştir.



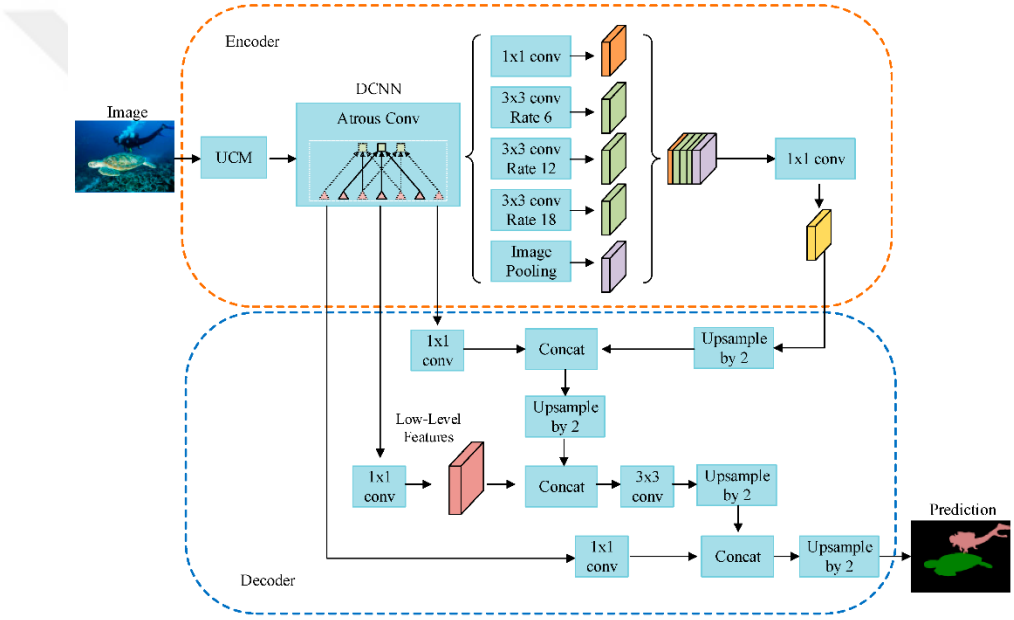
Şekil 7. SegNet mimarisi [33].

Mask R-CNN [9], görüntü segmentasyonu ve nesne tanıma için bir yapısal olarak ekstra bir maske tahmini modülü içeren bir CNN uygulamasıdır. Mask R-CNN, Faster R-CNN'den türetilmiştir ve nesneleri tanımlamak için region proposal network kullanır ve bu tanımlanmış bölgelerin her birinde bir maske tahmin yapar. Mask R-CNN, segmentasyon ve nesne tanıma problemlerinde çok yönlü bir şekilde kullanılabilir ve çok sayıda görüntü analitik uygulamada başarılı sonuçlar vermiştir.

Chen vd. tarafından geliştirilen DeepLab [34], semantik görüntü segmentasyonu için son teknoloji derin öğrenme tabanlı bilgisayarlı görü teknolojisidir. Bir görüntüdeki her piksele semantik etiketler (insan, kedi, köpek, arka plan vb.) atamayı amaçlar. DeepLab, özellikleri çıkarmak ve birden çok ölçekte tahminler yapmak için evrişimli sinir ağlarını ve "atrous convolutions" denilen genişletme konvolüsyonunu kullanır. Atrous konvolüsyon, aşağı örnekleme katmanı için bir alternatiftir. Özellik haritalarının uzamsal boyutunu korurken alıcı alanı artırır. Sürücüsüz arabalar, görüntü analizi ve nesne algılama gibi çeşitli uygulamalarda yaygın olarak kullanılmaktadır. DeepLab ayrıca Google tarafından açık kaynaklıdır. Şekil 8'de atrous konvolüsyonunun yapısı ve şekil 9'da DeepLab mimarisi verilmiştir.

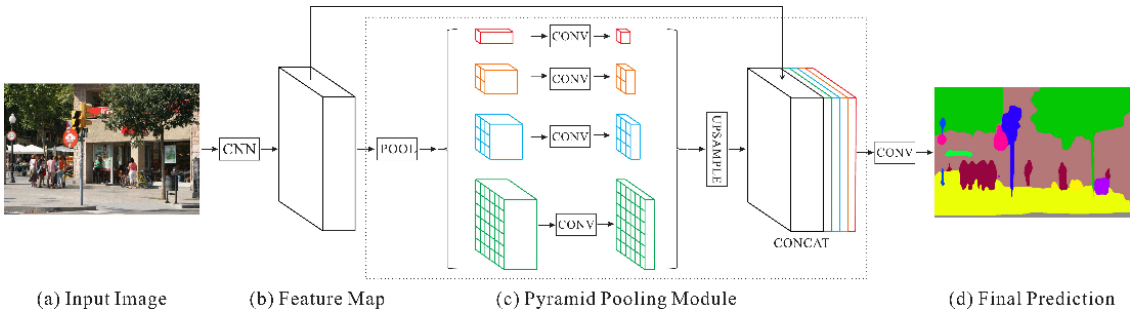


Şekil 8. Atrous konvolüsyonunun yapısı [34].



Şekil 9. DeepLab mimarisi [35].

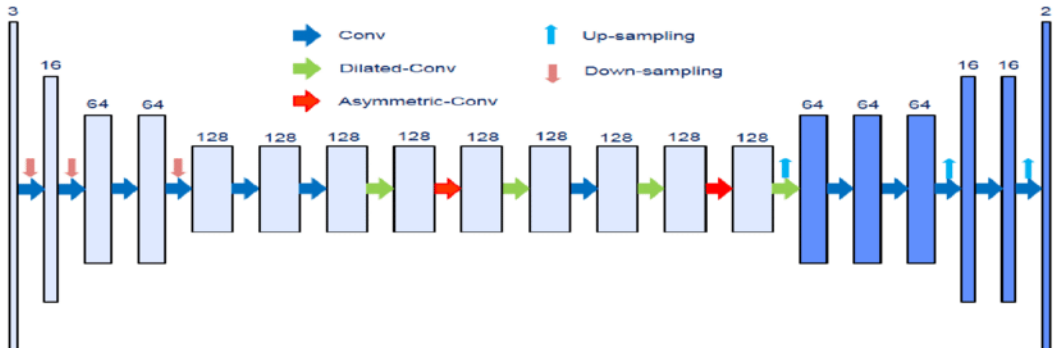
Pyramid Scene Parsing Network (PSPNet) [36], Zhao vd. tarafından semantik görüntü segmentasyonu için geliştirilmiş derin bir sinir ağı mimarisidir. Küresel bağlam bilgisini bir piramit havuzlama modülü aracılığıyla yerel yoğun tahminlere dahil etmek ve sahne ayrıştırma görevlerinde daha iyi performans elde etmek için tasarlanmıştır. Ağ, yüksek bir doğruluğa sahiptir ve otonom sürüş, tıbbi görüntüleme ve nesne algılama dahil olmak üzere çeşitli bilgisayarla görme uygulamaları için yaygın olarak kullanılmaktadır. Şekil 10'da PSPNet mimarisi verilmiştir.



Şekil 10. PSPNet mimarisi [36].

DenseNet [37], 2017 yılında Densely Connected Convolutional Networks adıyla tanımlanmış bir deep learning arayüzüdür. Bu arayüz, katmanlar arasındaki bağlantıların yoğun olmasını ve her bir katmanın önceki tüm katmanların çıktılarını kabul etmesini sağlar. Bu, ağın önceki katmanların öğrenilen özelliklerinin diğer katmanlar tarafından kullanılmasını ve ağın derinliğinin artmasına rağmen ağırlıkların fazla uçmasını önler. Bu özellikler, DenseNet'in performansını arttırmak için kullanılır ve güncel görüntü sınıflandırma, bölütleme ve diğer görsel analitik görevlerinde sık kullanılır.

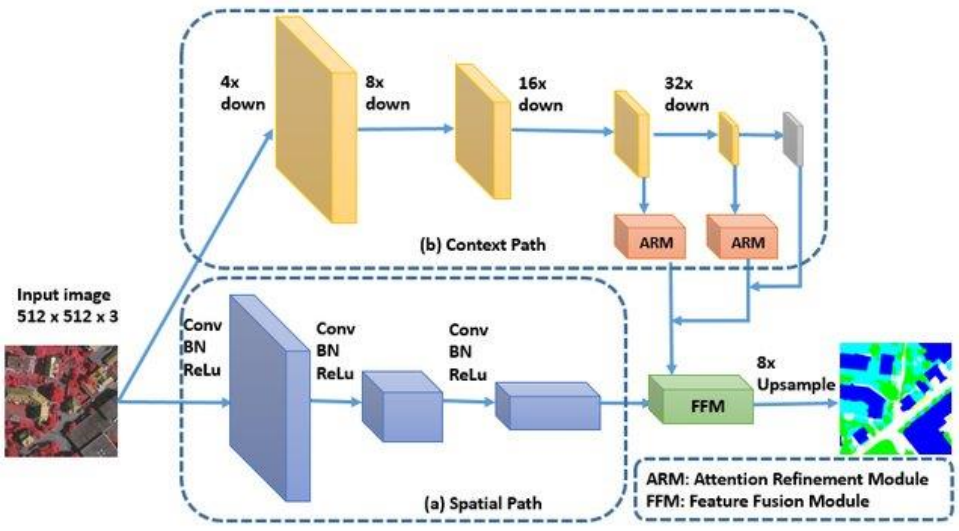
Enet [38] Paszke vd. tarafından gerçek zamanlı bölütleme için geliştirilmiş mimaridir. Bu algoritma, küçük boyutlu ve hızlı bir modele sahip olduğundan, görüntü segmentasyonu görevleri için mobil cihazlar veya güçsüz sistemler gibi enerji sınırlı ortamlarda kullanılabilir. Enet, Semantik Segmentasyon görevleri için kullanılan diğer popüler algoritmaların performansını iyileştirirken, aynı zamanda veri boyutunu ve eğitim süresini azaltır. Şekil 11'de Enet mimarisi verilmiştir.



Şekil 11. Enet mimarisi [39].

YOLACT [40] (You Only Once Look Consolidated and Take One), NVIDIA tarafından geliştirilmiş gerçek zamanlı bir nesne algılama sistemidir. YOLO [11] mimarisine dayalıdır ve gerçek zamanlı olarak nesne algılama ve örnek bölütleme gerçekleştirmek için tek bir sinir ağı kullanır. YOLACT, sürücüsüz arabalar, güvenlik kameraları ve video oyunları gibi çeşitli uygulamalarda kullanıma uygun hale getirerek hızlı, doğru ve verimli olacak şekilde tasarlanmıştır. Sistem, büyük veri kümeleri üzerinde eğitilmiştir ve her nesneye benzersiz bir etiket ve bir dizi sınırlayıcı kutu atanarak, bir görüntü veya video akışındaki birden çok nesneyi algılayabilir.

Bilateral Segmentation Network (BiSeNet) [41], 2018 yılında Changqian Yu vd. tarafından yayınlanan bir görsel segmentasyon modelidir. BiSeNet, görüntülerin detaylı ve genel özelliklerini ayırmak için iki adet farklı ağdan oluşur. Bu ağlar birbirlerine bağlı ve sıkıca bir şekilde entegre edilmiştir, böylece model detaylı ve genel özellikleri bir arada düzgün bir şekilde öğrenebilir. BiSeNet, yüksek çözünürlüklü görüntüler için etkileyici bir performans sunar ve diğer popüler görsel bölütleme modellerine göre daha hızlıdır. Şekil 12'de BiSeNet mimarisi verilmiştir.

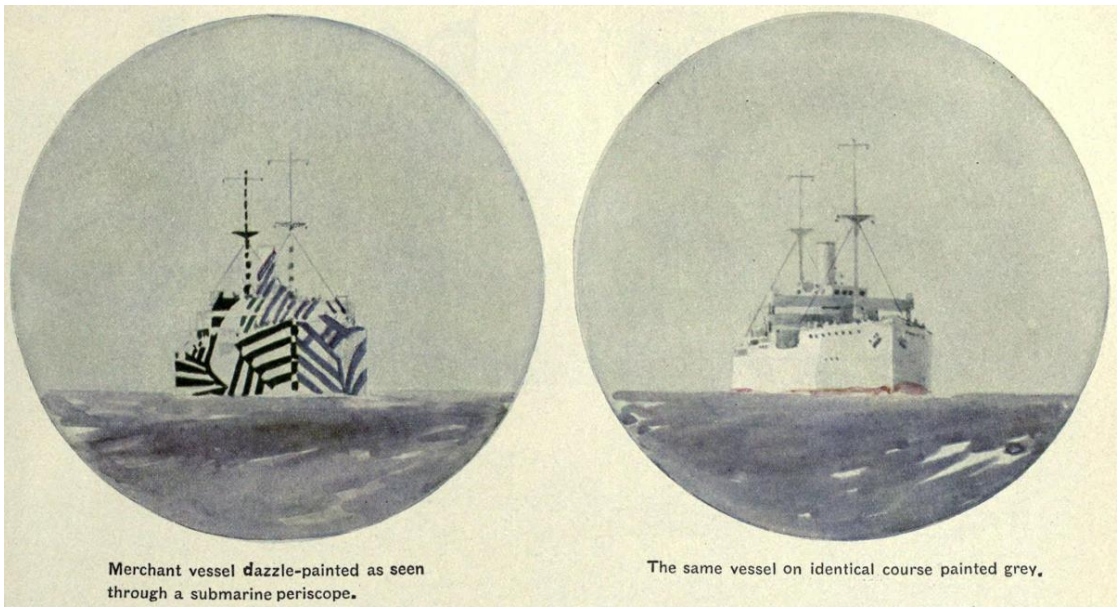


Şekil 12. BiSeNet mimarisi [42].

1.4. Askeri Kamuflaj Paternleri

Geleneksel askeri kamuflaj giysileri renk ve desen olmak üzere iki bileşenden oluşur [43]. Kamuflajlar arka planla benzeşmenin yanı sıra, gölge kaldırma, karşıt gölge oluşturma,

bozulma gibi çeşitli taktiklerle tasarlanırlar zıt renk ve karmaşık geometrilere sahiplerdir [44]. Kamuflajda amaç sadece görünmez olmak değil hız ve yön karmaşası oluşturmak, daha büyük veya küçük görünmek gibi farklı gerekçelerde vardır [44]. Bu sebeple birbirinden çok farklı kamuflaj paternleri tasarlanmıştır. Bu tasarımların taklit edilmesi, farklı çevrelere göre değiştirilmesi sonucu günümüzde birçok çeşidi vardır. Şekil 13'te Birinci Dünya Savaşında İngilizler tarafından denizaltılarını manipüle etmek için kamufle edilmiş bir gemi ve aynı büyüklükte kamufle edilmemiş bir başka geminin periskop görüntüsü bulunmaktadır. "Dazzle paint" adını verdikleri bu kamuflaj taktiği ile geminin hız ve yön bilgisinin anlaşılmasını zorlaştırmayı amaçlamışlardır.



Şekil 13. Kamufle edilmiş bir gemi ve aynı büyüklükte kamufle olmamış bir başka geminin denizaltı periskobundan görünümü [44].

1.4.1 . Disruptive Pattern Material (DPM) Kamuflaj Ailesi

Günümüzde birçok ülke (Birleşik Krallık, Umman, Filipinler, Portekiz, Yemen, Yeni Zelanda, Bulgaristan, Endonezya ve Kamboçya) tarafından kullanıldığı için en evrensel kamuflaj deseni olabilir. "Disruptive Pattern Material" tanımı İngiliz Savunma Bakanlığı tarafından 1960'tan beri yayınladıkları birçok kamuflaj deseni için kullanılmaktadır. DPM deseni ılıman iklimler için geliştirilmiştir. Hâkî veya ten rengi bir arka plan üzerinde siyah,

kahverengi ve parlak yeşil renklerde şekillerden oluşur. Günümüzde orman veya çöl gibi ortamlar için farklı varyasyonları vardır. Şekil 14'te DPM deseni için örnek verilmiştir.

Şekil 14. Disruptive Pattern Material (DPM) Kamuflaj Deseni

1.4.2 . Brushstroke (Fırça Darbesi) Kamuflaj Deseni

Bu kamuflajda desen geniş ağızlı fırça darbeleriyle elde edilmektedir. Desende yüksek kontrastlı tonlar veya gözle görülür renk farklılıkları kullanılarak karşıt gölgeleme yapılmıştır [45]. Şekil 15'te brushstroke deseniyle ilgili örnek verilmiştir.

Şekil 15. Brushstroke kamuflaj deseni

1.4.3 . Chocolate Chip (Çikolatalı Çentik) Kamuflaj Deseni

İlk olarak 1971'de Amerika Birleşik Devletleri tarafından altı renkli çöl kamuflaj deseni olarak geliştirildi [4]. Bu tasarım Basra Körfezi Savaşında ve farklı çöl operasyonlarında kullanılmıştır. Daha sonraları 3 renkli çöl deseni gibi kullanımları da olmuştur. Şekil 16'da chocolate chip deseniyle ilgili örnekler verilmiştir.

Şekil 16. Chocolate chip (çikolatalı çentik) [46].

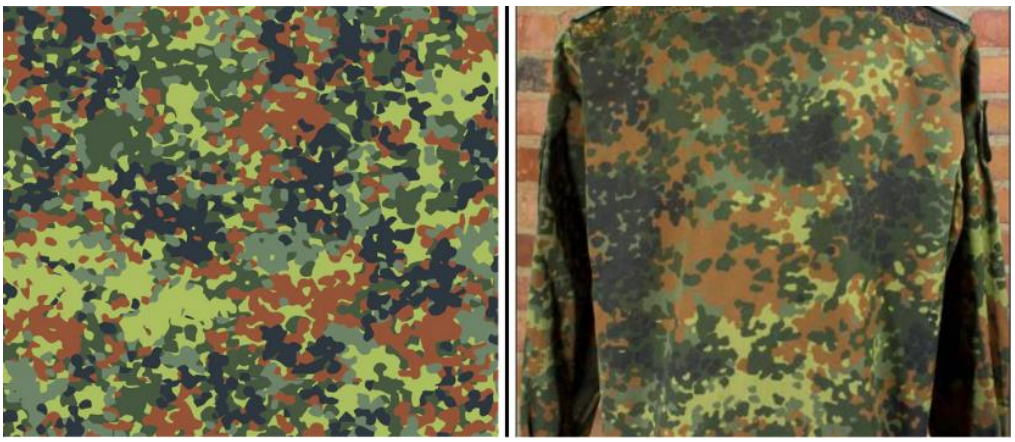
1.4.4 . Leaf Pattern (Yaprak Deseni)

1948'de ABD Ordusu Mühendis Araştırma ve Geliştirme Laboratuvarı (US Army Engineer Research and Development Laboratory-ERDL), yeşil bir arka plan üzerinde siyah "dalları" olan orta kahverengi ve çimen yeşili organik şekillerden oluşan genel amaçlı bir orman kamuflajı tasarladı. Bu modele genellikle ERDL modeli denir, ancak yıllar içinde koleksiyoncular arasında "yaprak deseni" takma adını kazanmıştır. Bu kamuflaj deseni tropikal bölgeler için tasarlanmış bir modeldir. Bu askeri kamuflaj deseni şekil 17'de örnek olarak verilmiştir.

Şekil 17. Yaprak deseni

1.4.5 . Flecktarn Kamuflajı

Kamuflaj deseni, Alman tasarımcılar tarafından 1970'lerde Bundeswehr Truppenversuch 76 veya 1976 Alman Ordusu Üniforma Denemeleri sırasında icat edildi. Kamuflaj deseninin ismi Almanca nokta anlamına gelen “fleck” sözcüğünden türetilmiştir. Flecktarn, yosun yeşili bir arka plan üzerinde siyah, kırmızımsı kahverengi, koyu zeytin ve orta zeytin yeşili benekler içeren beş renkli bir desendir. Desenle ilgili örnek şekil 18’de verilmiştir.



Şekil 18. Flecktarn kamuflaj deseni

1.4.6 . Yağmur Deseni

Bu kamuflaj deseni ailesinde yüksek oranda dikey olarak düşen yağmur ve lekelerden oluşan geometriler kullanılmıştır. Tek renkli bir arka plan rengi kullanılan bu tasarımdan yola çıkılarak tasarlanmış farklı isimlerde modeller de vardır (Örneğin; Almanların kullandığı "Splittermuster (kıymık)" modeli veya Güney Afrika'da kullanılan "pirinç lekesi" modeli). Bu kamuflaj desenine ait örnek şekil 19'da verilmiştir.

Şekil 19. Rain (yağmur) kamuflaj patern örneği

1.4.7 . "Duck Hunter" Kamuflaj Deseni Ailesi

Kamuflaj deseni genellikle düz bir arka plan üzerinde çeşitli renklerde büyük, düzensiz noktalara sahip bir tasarımdır. "Ördek avcısı" desenlerinin kökeni, İkinci Dünya Savaşı'nın ABD M1942 nokta desenli kamuflajına kadar uzanır ve öncelikle Pasifik Harekat Tiyatrosu'nda giyilir [4]. Orijinal modelin varyasyonları, 1960'lardan 1990'lara kadar çok sayıda Amerikalı ve yabancı şirket tarafından yeniden üretildi ve sporcular için av kıyafeti olarak pazarlandı. Şekil 20'de ördek avcısı deseniyle ilgili örnek verilmiştir.

Şekil 20. Duck Hunter kamuflaj desen örneği

1.4.8 . Kertenkele Desen Ailesi

"Kertenkele" deseninin tanımı, İkinci Dünya Savaşı döneminde ortaya çıkan İngiliz brushstroke deseninin bir türü olan 1950'lerin Fransız yatay çizgili desenlerini ifade etmek için kullanılmıştır. Fransızlar "leopar kamuflajı" olarak adlandırırken, "kertenkele" terimi Fransız paraşütçüler tarafından Cezayir Savaşı sırasında kullanılan bir takma adıdır. İki tip kertenkele deseni bulunur, yatay ve dikey. Orijinal Fransız tarzı yataydır ve Portekiz tarafından geliştirilen dikey tarz aynı şekilde etkilenmiştir. Sonuç olarak hem yatay hem de dikey tarzlarda bugün de üretilen kertenkele tasarımları vardır. Kaplan çizgili kamuflaj ise kertenkele tasarımından türetilmiş ve aynı kamuflaj türüdür ancak daha tutarlı bir şekildedir. Şekil 21'de kertenkele deseniyle ilgili örnek verilmiştir.

Şekil 21. Kertenkele askeri kamuflaj deseni

1.4.9 . Kaplan Şeritli Kamuflaj Deseni

"Kaplan şeridi" kamuflaj deseni, 1960'larda Güneydoğu Asya'da ortaya çıktı ve 1950'lerde Fransız "tenue du leopard" veya kertenkele tasarımından türedi. Fransız tarafından Birinci Çinhindi Savaşı sırasında Vietnam'da kullanılan kamuflaj üniformalarından türetilmişti. Çizgili üniforma adı verilen bu tasarımlar arasından kaplan şeridi terimi, kamuflaj tasarımının post tasarımına benzerliğini ifade eder. Kaplan şeritli desen, zeytin yeşil renkli ve kahverengimsi-donuk şeritlerin bulunduğu açık yeşil şekiller üzerinde koyu siyah şeritler içeren bir kamuflaj deseniydi. Üretimi 1967'de sona erdi ve 1970'e kadar giyilmeye devam edildi. Şekil 22'de kaplan şeridi kamuflaj deseniyle ilgili örnekler verilmiştir.

Şekil 22. Kaplan şeridi kamuflaj deseni örnekleri

1.4.10 . Puzzle Kamuflaj Deseni

"Puzzle Kamuflaj Deseni" kategorisi, koleksiyoncular ve tarihçiler tarafından bir yapboza benzeyen şekiller içeren kamuflaj tasarımlarına atıfta bulunur. Bu özel tasarım dalı belirli bir desene dayanmaz ancak birkaç küçük yönlerinin bir tema ile ilişkisi vardır. Bu desenler, görsel olarak yapboz parçalarına benzemekte veya onları hatırlatmaktadır. Belçika tarafından geliştirilen eski yapboz desenleri 1960'larda popülerdir ve hala bugün bu ülkede giyilmektedir. Ancak, Yugoslavya, Filipinler ve diğer yerlerde de benzer puzzle tasarımları

geliştirilmiştir ve birbirleriyle bağlantısı yok gibi görünür. Şekil 23'te puzzle kamuflaj deseni için örnekler verilmiştir.

Şekil 23. Puzzle kamuflaj deseni örnekleri

1.4.11 . Woodland Kamuflaj Desen Ailesi

"Woodland" Deseni, 1981 yılında tanıtılan ABD M81 Ormanlık Kamuflaj Deseni ve tüm türevleri için kullanılan bir terimdir. M81 deseni, ERDL modelinden türetilmiştir ve orijinal çizimlerin %60'ının büyütülmesiyle 2. nesil ERDL tasarımına benzer renk grubunu koruyan bir tasarımdır. M81, tasarlandığından bu yana en çok kopyalanan ve değiştirilen kamuflaj desenlerinden biri haline gelmiş ve hala dünya çapında askeri kuvvetler tarafından kullanılmaktadır. Şekil 24'te woodland kamuflaj deseni için örnekler verilmiştir.

Şekil 24. Woodland kamuflaj desen örnekleri

1.4.12 . Splinter (Kıymık) Kamuflaj Deseni

Splinter (Kıymık) deseni, kumaş üzerinde yağmur yağış şekilleri içeren orijinal Alman "Wehrmacht" kamuflaj tasarımlarına benzerdir. "Yağmur" terimi sadece Alman tasarımlı baskıların özelliğini tanımlarken, "kıymık" terimi daha geniş bir anlam taşır ve geometrik şekillerine sahip tüm kırılgan maddeleri benzer tasarımları kapsar. Şekil 25'te splinter kamuflaj deseni için örnek verilmiştir.

Şekil 25. Splinter kamuflaj desen örneği

1.4.13. Dijital Desenli Kamuflaj Desenleri

"Dijital" kamuflaj tasarımı, programlanmış bilgisayar algoritmaları kullanılarak oluşturulan mikro desenli bir kamuflaj desenidir. Bu tasarımlarda, büyük renk lekeleri yerine mikro desenler kullanılır ve bu, lekelerin bulanıklaşmasını veya titremesini engelleyerek etkili bir bozulma yaratır. Kanada, 1997 yılında CADPAT tasarımı ile ilk dijital kamuflaj tasarımını tanıtmıştır ve sonrasında birçok ülke de bu tarz tasarımları benimsemiştir. Şekil 26'da dijital desenli kamuflajlar için örnekler verilmiştir.

Şekil 26. Dijital desenli kamuflaj örnekleri

Bu desenlerin haricinde “urban” (Şehir ortamları için kullanılan gri, siyah ve beyaz desenlerdir.), “multi-terrain” (Çeşitli arazi tipleri için kullanılan çok renkli desenlerdir.), “snow” (Kar alanları için kullanılan beyaz, gri ve mavi desenlerdir.) gibi askeri kamuflaj aileleri de vardır.

1.5. Veri Seti

Bu çalışma için 7 ülkeye (Türkiye-Azerbaycan, Amerika, Rusya, Çin, Fransa, Almanya ve Irak) ait kamuflajlı asker görüntüleri toplanarak veri seti olarak kullanılmıştır. Azerbaycan ve Türkiye, ortak askeri çalışmalarından dolayı kamuflaj desenleri arasında benzerlik gösterir. Bu sebeple bir sınıf olarak tercih edilmiştir. Ülkelerin seçiminde çeşitliği artırmak için birbirinden çok bağımsız kamuflaj desenlerine sahip olmaları tercih edilmiştir fakat bazı ülkeler ise benzer kamuflaj ailesinden seçilmiştir. Bunun nedeni derin öğrenmeyi daha zorlu bir sınıflandırma probleminde test edebilmektir. Örneğin Amerika ve Fransa benzer woodland desen ailesinden kamuflajlara sahiptir fakat desenlerde bazı küçük farklar mevcuttur. Yine Rusya, Çin ve Almanya kamuflajları Amerika ve Fransa gibi benzer olmasalar da benzerlikler göstermektedir. Çin ve Rusya dijital tabanlı desen iken Almanya flecktarn desen ailesinden kamuflajlara sahiptir. Çin’in aynı desene sahip farklı 3 renkte kamuflajı kullanılmıştır. Türkiye ve Irak bağımsız desenlerdedirler. Türkiye’de dijital tabanlı desen kullanmıştır. Irak ise chocolate chip desen ailesinden kullanmıştır.

Fotoğrafların toplam sayısı 743’tür ve toplam 1233 adet asker görüntüsü içermektedir. Literatüre baktığımızda [5]’te yaprak örüntülerinin sınıflandırılması problemi için yapılan

çalışmada 32 sınıf için toplam 1900 görüntü kullanılmıştır ve [7]'de 4 farklı kamuflajlı nesne (kamuflaj asker, baykuş, beyaz kurt ve kelebek) sınıflandırma problemi için 200 farklı görüntü kullanılmıştır.

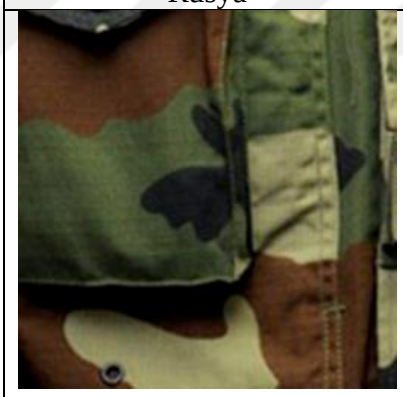
Fotoğraflar 700x700 boyutlarına ayarlanmış ve yaklaşık %90 (1233 adet asker görüntüsü) eğitim ve %10 (115 adet asker görüntüsü) test için rastgele ayrılmıştır. Görüntü artırımı ve algoritmanın desen paternine göre sınıflandırma yapmasına odaklanması için rasgele bazı fotoğraflara ön işlem uygulanmıştır. Her fotoğraf için aynı ön işlemler uygulanmamıştır ve her bir fotoğrafta da tüm işlemler yapılmamıştır. Yani, her fotoğraf için farklı ön işlemler seçilmiş ve her bir fotoğrafta sadece belirli işlemler gerçekleştirilmiştir. Bu yaklaşımın amacı, farklı görüntülerin özelliklerini çeşitlendirmek ve desen paternine dayalı sınıflandırma algoritmasının daha genel ve genellemesi güç olan durumlarla başa çıkabilmesini sağlamaktır. Böylece, çeşitli ön işlemlerle zenginleştirilmiş fotoğraflar, algoritmanın daha etkili bir şekilde sınıflandırma yapmasını ve daha yüksek başarı oranları elde etmesini hedeflemektedir.

Görüntüler LabelMe [47] aracıyla etiketlenmiş ve eğitime hazır hale getirilmiştir. Tablo 1'de kullanılan asker görüntü sayıları her ülke için ayrı ayrı verilmiştir. Tablo 2'de'de 7 sınıfa ait kamuflaj kumaş örnekleri ve tablo 3'te kullanılan 8 ülke için asker görüntülerinden örnekler verilmiştir (Azerbaycan, Türkiye, Amerika, Rusya, Çin, Fransa, Almanya, Irak).

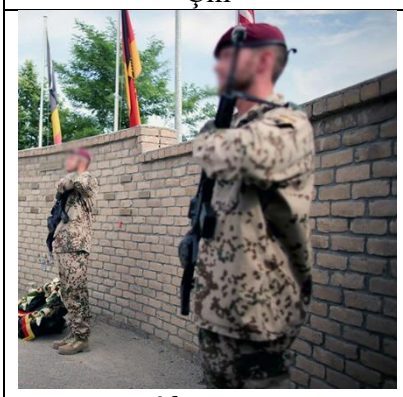
Tablo 1. Çalışmada Kullanılan Görüntülerin Dağılımı

	Eğitim	**Test**
Türkiye-Azerbaycan	146	12
Amerika	131	15
Rusya	143	21
Çin	199	18
Fransa	158	18
Almanya	105	7
Irak	236	24
Toplam	1118	115

Tablo 2. Kamuflaj Kumaş Örnekleri

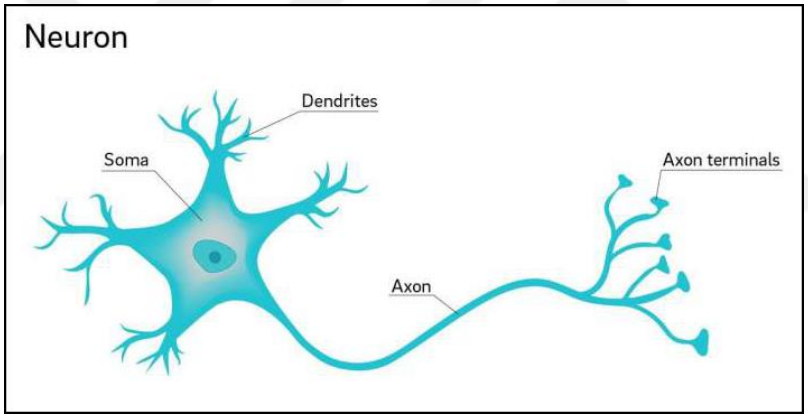
 Türkiye-Azerbaycan	 Amerika
 Rusya	 Çin
 Fransa	 Almanya
 Irak	

Tablo 3. Kamuflajlı Asker Örnekleri

 Azerbaycan	 Türkiye
 Amerika	 Rusya
 Çin	 Fransa
 Almanya	 Irak

1.6. Derin Öğrenme

Derin öğrenme, bilgisayarın insan gibi davranmasını (genelleme, özellik çıkarma, örüntü tanıma vb.) isteme sonucu insanın sinir ağının modellenmesi ile yapay sinir ağlarının oluşturulması sonucu olarak 1990'lı yıllarda geliştirilmeye başlanmış ve Hinton vd. tarafından yayınlanan önemli bir makale [48], derin öğrenme terimini tanıtmış ve derin öğrenme alanında büyük bir başlangıç noktası olmuştur. Yapay sinir ağları, biyolojik organizmaların öğrenme tekniklerini taklit eden makine öğrenme yöntemlerinden biridir [49]. Şekil 27, biyolojik organizmalarda bulunan bir nöron ve bileşenlerini göstermektedir. İnsan sinir sistemi, nöron adı verilen hücrelerden oluşur. Aksonlar ve dendritler, farklı nöronları birbirine bağlamak için kullanılan bağlantı elemanlarıdır. Sinapslar, aksonun ucunda bulunur ve nöronlar arasındaki bağlantıyı sağlar [50].



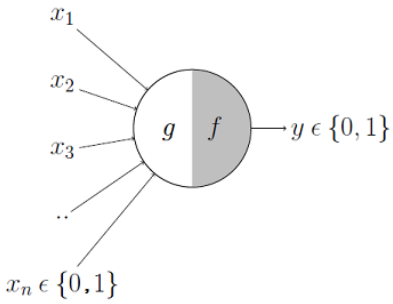
Şekil 27. Biyolojik Sinir Ağı [51].

Günümüzde bilgisayar işlemci gücünün artırılması, büyük veri kümelerine ulaşımın kolaylaşması gibi nedenlerle bilgisayar bilimleri alanında büyük ilgi odağı haline gelmiştir. Derin öğrenme makine öğrenimi türüdür. Yapay zekâ için makine öğrenimi algoritmaları oldukça popüler hale gelmiş ve birçok yeni mimariler geliştirilmiştir. Makine öğrenimi içerisinde yer alan derin öğrenme şekillendirilebilirliği sayesinde günümüz problemlerine hızlıca çözümler sunmaktadır. Burada geçen derin kelimesi tasarlanan mimarinin derinliğidir ya da YSA'nın çok katmanlı oluşudur [48]. Eğitimden endüstriye, sağlık alanına kadar, bilgisayar görmesi, ses işleme, görüntü işleme, robotik, çeviri, gelecek tahmini, askeri uygulamalar gibi pek çok alanda akıllı çözümler sunmaktadır. Derin öğrenme

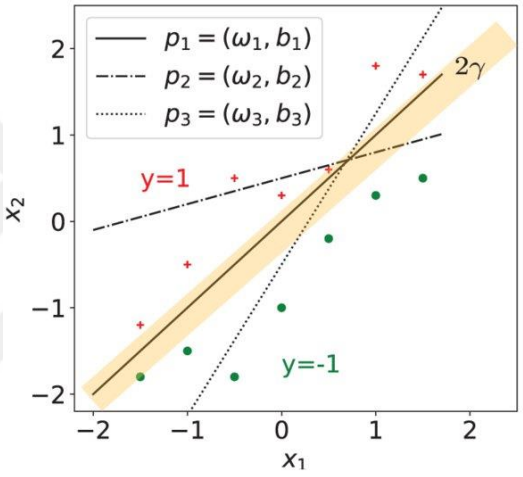
yaklaşımlarının önemi doğrusal olmayan problemler için çözümler sunabilmesidir [52]. Bu sebeple özellikle bilgisayarlı görü alanında evrişimli sinir ağları görüntüler için sınıflandırma ve nesne tanıma gibi problemleri çözmek amaçlı geliştirilmiş bir derin öğrenme yaklaşımıdır [53]. CNN yaklaşımında model, verilen görüntülerin özelliklerini ağ yapısı boyunca yayarak öğrenir. Problemin türüne, görüntü yapısına, veri büyüklüğüne ve hata toleransına göre farklı birçok CNN modeli geliştirilmiştir.

1.7. Derin Öğrenme Tarihi

Yapay zekâ (Artificial Intelligence-AI), çeşitli matematiksel ve mantıksal araçları kullanarak insan zekasını taklit etmeyi amaçlar. İlk AI sistemleri kural tabanlı sistemlerdi. Bu sistemler, problemi çözmek için biçimsel matematiksel kuralları öğrenebildiler ve akıllı sistemler olarak kabul edildiler. Ancak bu sistemler, biçimsel kuralları olmayan sorunları çözmede iyi performans göstermiyordu fakat insanlar bunları kolaylıkla çözebiliyordu (Örneğin; nesneleri tanımlama, konuşulan kelimeleri anlama vb.). Bu sorunu çözmek için, insan zihnini taklit etme konusunda aktif araştırmalar başladı ve 1958'de Frank Rosenblatt tarafından "Perceptron" [54] adlı yapay sinir ağı önerildi. Perceptronlar o dönemde çok ilgi gördü ve daha sonra perceptron ağlarının birçok varyasyonu zamanla ortaya çıktı. Ancak, herkes perceptron ağlarının potansiyeline inanmıyordu, gerçek yapay zekanın kural tabanlı olduğuna ve perceptron'un kural tabanlı olmadığına inanan insanlar vardı. Minsky ve Papert, perceptron'un bir analizini yaptılar ve perceptron ağlarının yalnızca doğrusal olarak ayrılabilen sınıfları ayırdığı sonucuna vardılar [55]. Makaleleri Exclusive-OR(X-OR) problemini doğurdu [55]. X-OR problemi 70'li yıllarda yapay zekanın kışı sayılan klasik bir problemdir. Bu problemi anlamak için perceptron ağının nasıl çalıştığını anlamalıyız. Perceptron, McCulloch-Pitts tarafından önerilen [56] ve McCulloch-Pitts nöronu olarak adlandırılan nöron mimarisinin basitleştirilmesine dayanmaktadır. Bu nöron mimarisinin iki girişi ve bir çıkışı vardır ve nöronun önceden tanımlanmış bir eşiği vardır, girişlerin toplamı eşiği aşarsa çıkış aktiftir, aksi takdirde aktif değildir [56]. Şekil 28’de McCulloch-Pitts nöronunun çok girişli tek çıkışlı hali verilmiştir.

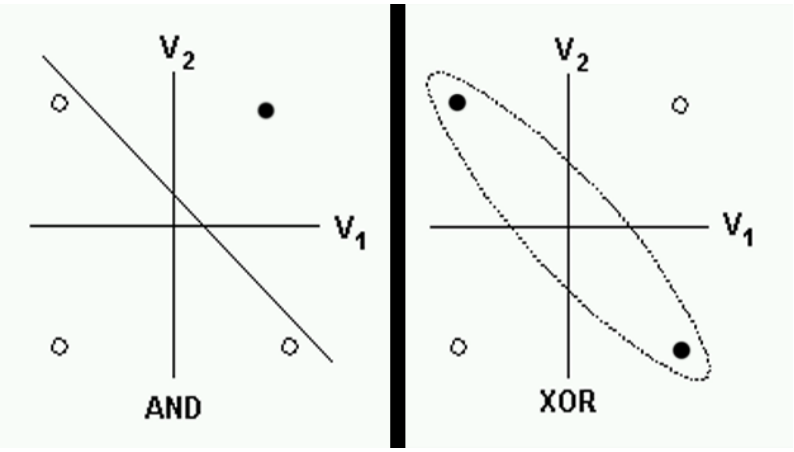


Şekil 28. McCulloch-Pitts nöronu [57].



Şekil 29. Perceptron öğrenme [58].

Şekil 29'da perceptron ağının doğrusal problemleri nasıl çözdüğü gösterilmiştir. Minsky ve Papert, [55]'de yaptıkları perceptron ağının basitleştirmesiyle, çok basit sınıflandırma problemlerini öğrenemediğini kanıtlamak için kullandılar. Örnek olarak Exclusive-OR'u seçtiler ve perceptron ağının X-OR problemini öğrenme yeteneğinin olmadığını kanıtladılar. Şekil 30'da gösterildiği gibi, X-OR problemi perceptron ağı ile 2 boyutlu düzlemde çözülemez. Dolayısıyla perceptron, giriş örneklerini doğru bir şekilde sınıflandırmak için bir ayırma düzlemi öneremez. Perceptron ağının diğer bazı faktörlerle birlikte X-OR problemi ile başa çıkamaması, yapay sinir ağlarına ilginin azalmasına sebep oldu. Daha sonra perceptron temelli YSA'lar ile X-OR'u çözme yeteneğine sahip birçok yaklaşım ortaya çıktı.



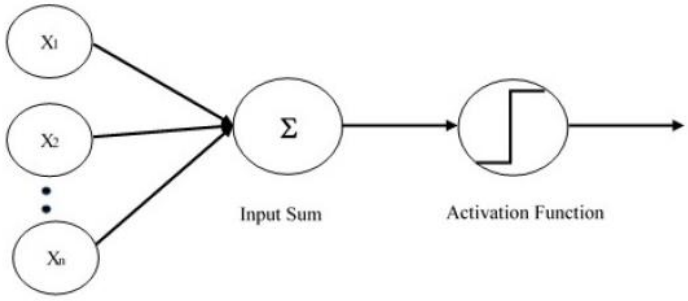
Şekil 30. Perceptron ağının çözebildiği AND problemi ve çözemediği X-OR problemi

Derin Öğrenme, perceptron tabanlı nöron yığınları oluşturduğumuz ve onları birden çok katmanda sıraladığımız derin yapay sinir ağlarıdır ve bir makine öğrenmesi türü olarak geçer. Tek gizli katmanlara sahip ilk modeller, çok katmanlı perceptron ağlar (Multi Layer Perceptron-MLP) olarak adlandırıldı ve sığ ağlar olarak kabul edildi. Derin ağların birden fazla katmanı vardır ve son çalışmalar, nesne tanımlama, doğal dil işleme, dil çevirisi ve daha pek çok sorunu verimli bir şekilde çözme becerisi göstermiştir. Sinir ağları insan zihninden esinlenirken, derin öğrenmedeki amaç insan zihnini kopyalamak değil, görüntü tanıma, ses tanıma, dil çevirisi, sanat üretimi vb. sorunları çözmede iyi performans gösteren modeller oluşturmak için matematiksel araçları kullanmaktır.

1.7.1. Perceptron Ağı

Bir yapay sinir ağının en basit halidir. Şekil 31'de gösterildiği gibi bir yapıya sahiptir. Ağın girişinde giriş değerleri, bu değerlere karşılık gelen ağırlık vektörleri vardır. Bu değerler çarpılarak bir sonraki adım olan toplama bölümüne iletilir. Bu bölüme akümülatör de denir. Burada ağın tasarımına göre bias (β) değeri de eklenerek belirli bir sapma oluşturulur. Bias girdisi, bu tip ağlardaki nöronların değerlerinin 0 olmamasını sağlar. Bias girdisi her zaman 1 olarak ayarlanır. Son işlemimiz aktivasyon fonksiyonudur. Akümülatörden gelen değer burada doğrusal olmayan bir fonksiyona girdi olarak verilir. Günümüzde farklı aktivasyon fonksiyonları kullanılır. En yaygın olan ReLu fonksiyonudur. ReLu işlemi sonucu ağın çıkışında 1 veya 0 sonucu verilir. Aslında amaç genel çözüme bu düğümün etkisinin olup olmadığıdır. Ağda eğitim ağırlık vektörlerinin güncellenmesi ile

gerçekleştirilir. Bu eğitim danışmanlı (geri beslemeli) ve danışmansız (ileri yayılımlı) olarak iki türdür.



Şekil 31. Rosenblatt tarafından geliştirilen tek bir sinir elemanı

Şekil 31'deki modelin matematiksel gösterimi denklem 1'de verilmiştir. Burada X_i (i = 1, 2, …, n), ağın girdilerini, W_i (i = 1, 2, …, n), girdilerin karşılığındaki ağırlık değerlerini ve β ise eşik değerini gösterir. Buradaki f(), aktivasyon fonksiyonudur. Tek katmanlı ağlarda f aktivasyon fonksiyonu doğrusaldır. Eğer hesaplanan toplam değer belirli bir eşik değerinden büyükse, çıkış aktif hale gelir. Değilse, çıkış pasif kalır. Bu durumlar sırasıyla 0 ve 1, ya da -1 ve 1 olarak değerlendirilir.

$$y = f\left(\beta + \sum_{i=1}^{n} W_i X_i\right) \tag{1}$$

$$f(net) = \begin{cases} 1 & net > \theta \\ 0 & net \leq \theta \end{cases} \tag{2}$$

$$W_i(t+1) = W_i(t) + \Delta W_i(t) \tag{3}$$

$$\beta_i(t+1) = \beta_i(t) + \Delta\beta_i(t) \tag{4}$$

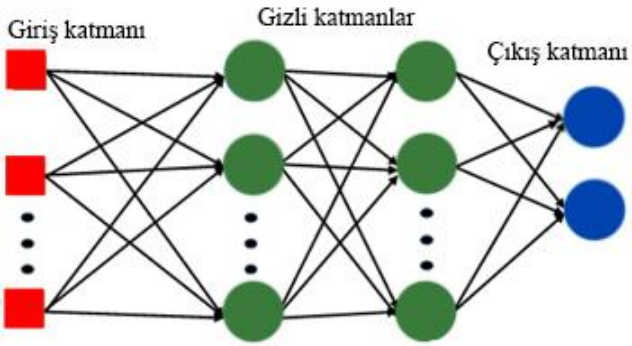
Tek katmanlı sinir ağında öğrenme işlemi denklem 3 ve denklem 4'teki gibi ağın güncellenmesiyle olur.

Tek katmanlı sinir ağı modellerinde “Perceptron [54]” ve “Adaline (Adaptive Linear Neuron) /Madaline (Multi-layer Adaptive Linear Neuron) [59]” modelleri kullanılmaktadır. Perceptron modeli, birden fazla girdiyi kullanarak bir çıktı üreten bir sinir hücresi prensibine dayanır. Ağın çıktısı, girdilerin ağırlıklı toplamının bir eşik değeriyle kıyaslanması sonucu bulunur. Eşik değeri aşıldığında çıktı 1, eşik değeri altında ise 0 olarak belirlenir. Adaline'e göre daha basit bir modeldir ve tek bir çıkış üretir. Perceptron, girdi verilerinin lineer olarak ayrıştırılması durumunda sınıflandırma problemlerine uygulanabilir. Ayrıca, Perceptron'un ağırlık ve bias değerleri, yanlış tahmin edilen girdi verilerine göre iteratif olarak güncellenir, ancak sınıflandırma için sadece iki sınıfı (1 veya 0) tanımlar. Adaline, Perceptron'a göre daha gelişmiş bir modeldir ve sürekli olmayan (regresyon) problemlere de uygulanabilir. Madaline ise, birden fazla Adaline modülünü kullanarak birçok çıkış üretebilen bir sinir ağı modeldir. Bu modelde, çıkışlar arasındaki bağımlılıklar dikkate alınır ve ağırlık ve bias değerleri her iterasyonda birden fazla çıkışı optimize etmeyi hedefler, daha fazla sınıfı tanımlayabilir.

1.7.2. Multi Layer Perceptron (MLP)

Her ne kadar yapay sinir ağları çözümler sunsa da doğrusal olmayan X-OR gibi problemlerin çözümünde yetersizdi. Çok katmanlı ağların kullanılması ve Rumelhart vd. ortaya attığı geri yayılımlı [60] yaklaşımla yapay sinir ağlarına olan ilgi artmış ve doğrusal olmayan problemler için çözümler sunulur hale gelmiştir.

Çok katmanlı perceptron ağı girdi katmanı, gizli katman ve çıkış katmanından oluşur. Giriş katmanı, işlenecek giriş verisini alır bu katmanda aktivasyon fonksiyonu bulunmaz. Tahmin ve sınıflandırma gibi gerekli işler çıktı katmanı tarafından gerçekleştirilir. Giriş ve çıkış katmanı arasına yerleştirilen gizli katman ise MLP'nin gerçek hesaplama birimidir. Gizli katman sayısı birden fazla olabilir. MLP’nin yapısı şekil 32’de gösterildiği gibidir.



Şekil 32. Çok katmanlı perceptron ağı

$$y_i = f\left(\beta + \sum_{j=1}^{n} W_{ij} X_i\right) \tag{5}$$

$$e_i = y_i^* - y_i \tag{6}$$

MLP'de y_i = (i = 1, 2, …, m) denklem 5'te gösterildiği gibi her Çıkış için ayrı ayrı hesaplanır. Denklem 6'da e_i, i çıkışındaki hatayı gösterir. Burada y_i^* beklenen gerçek değerdir $y_i$ise ağın verdiği cevaptır, ikisi arasındaki fark o çıkıştaki o anki hatadır. Bu denklemden türetilmiş farklı kayıp (loss) fonksiyonları vardır. Kayıp fonksiyonu, bir makine öğrenme modelinin performansını ölçmek için kullanılan bir metriktir. Bu fonksiyon, modelin tahminlerinin gerçek değerlere olan uyumunu ölçer ve modelin eğitilmesi sırasında azaltılması amaçlanır. Kayıp fonksiyon değeri düşerse, modelin performansı iyileşir. En yaygın kullanılan kayıp fonksiyonları arasında Ortalama Karesel Hata (Mean Squared Error-MSE) [61], Ortalama Karekök Hata (Root Mean Square Error-RMSE), Ortalama Mutlak Hata (Mean Absolute Error-MAE) "Categorical Crossentropy" ve "Binary Crossentropy" bulunur.

- Mean Squared Error (MSE), regresyon problemleri için en yaygın kullanılan kayıp fonksiyonlarından biridir. Öngörülen ve gerçek hedef değerler arasındaki kare farklarının ortalamasını ölçer. Denklem 7'de gösterildiği gibi hesaplanır.

$$MSE = \frac{1}{n}\sum_{i=1}^{n} e_i^2 \tag{7}$$

• Root Mean Square Error (RMSE), regresyon modellerinin performansını değerlendirmek için yaygın olarak kullanılan bir kayıp fonksiyonudur genellikle farklı modellerin performansını karşılaştırmak için bir ölçüt olarak kullanılır. RMSE, aykırı değerlere karşı hassastır ve tahmindeki büyük hataların varlığından etkilenebilir. RMSE, MSE'nin karekökü şeklinde hesaplanır denklem 8'de gösterilmiştir.

•

$$RMSE = \sqrt{MSE} \tag{8}$$

• Mean Absolute Error (MAE), tahmin edilen değerler ile gerçek değerler arasındaki mutlak farkların ortalama büyüklüğünü ölçer. RMSE'den farklı olarak MAE, hesaplamada farkların mutlak değeri alındığından aykırı değerlerin varlığından etkilenmez. Bu, MAE'yi tahmindeki büyük hatalara karşı duyarsız olan sağlam bir ölçüm yapar. Ancak, farkların mutlak değeri alındığından, MAE, RMSE'den daha az yorumlanabilir bir ölçüm sağlar. Ayrıca, MAE'nin hedef değişkenle aynı birimlerde olmaması bazı uygulamalarda hatanın büyüklüğünü yorumlamayı zorlaştırabilir. Genel olarak MAE, regresyon problemleri için yararlı bir kayıp fonksiyonudur ve özellikle aykırı değerlerin varlığının bir endişe kaynağı olduğu uygulamalarda genellikle RMSE'ye bir alternatif olarak kullanılır. Denklem 9'da gösterildiği gibidir.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|e_i| \tag{9}$$

• Binary Cross-Entropy, hedef değişkenin sadece iki değer alabildiği ikili sınıflandırma problemlerinde kullanılan bir kayıp fonksiyonudur. Gerçek ikili etiketler ile pozitif sınıfın tahmin edilen olasılıkları arasındaki farklılığı ölçer.

• Categorical Cross-Entropy, hedef değişkenin ikiden fazla değer alabildiği çok sınıflı sınıflandırma problemlerinde kullanılan bir kayıp fonksiyonudur. Gerçek kategorik etiketler ile tahmin edilen sınıf olasılıkları arasındaki farklılığı ölçer.

1.7.3. Yapay Sinir Ağında Öğrenme İşlemi

Ağın verileri kaynağından öğrenme yeteneği, YSA'ların en önemli özelliğidir. Bu özellik, biyolojik ağlardaki sinaps gibi ağdaki verilerin korunmasını sağlar, bu da ağırlıkların önemini artırır. Tüm ağın en uygun değere sahip olması gerektiği unutulmamalıdır. Ağırlıklar, veriler için en uygun değeri bulma odaklıdır ve bu ağın eğitilmesi anlamına gelir. YSA'da eğitim genel olarak 3 adımdır. Bunlar verinin ağ girişinden sunulması (forward), hata veya kayıpların hesaplanması, hatanın ağa yayılarak ağırlıkların güncellenmesi (backward) olarak genellenebilir. İlk adım olan forward denklem 1 ve denklem 2'de verilmiştir. Diğer iki adımda yapılan değişikliklere göre farklı öğrenme stratejileri geliştirilmiştir. Genel olarak yaklaşım ilk adımda oluşan değer ile beklenen arasındaki değerin bir kayıp olduğu ve bu kayıp değerine göre ağdaki ağırları değiştirerek iteratif bir şekilde eldeki veriye göre en iyi ağırlık değerlerinin bulunmasına kadar eğitim işleminin devam ettirilmesidir. Genel olarak zaman içinde geliştirilen yapay sinir ağlarındaki öğrenme kuralları tablo 4'te verildiği gibidir.

Tablo 4. Öğrenme Yaklaşımları

Yaklaşım	**Tanım**	**Kaynakça**
Gradient Descent	Ağın ağırlıklarını minimum hata fonksiyonu değerine ulaşmak için optimize etmek için kullanılan bir yöntemdir.	[62]
Stochastic Gradient Descent	Ağırlıkları tek tek güncellemeyi ve rastgele seçilen bir örnekle çalışmayı içeren bir gradient descent türüdür.	[63]
Mini-batch Gradient Descent	Ağırlıkları, belirli sayıdaki örnekle birden fazla kez güncellemeyi içeren bir gradient descent variantıdır.	[64]
Momentum	Ağırlıkların hızlı hareket ettiği yönlerin devamını sağlamak için kullanılan bir yöntemdir.	[65]
Adagrad	Her ağırlık için farklı öğrenme hızı sağlamak için kullanılan bir yöntemdir.	[66]
Adadelta	Adagrad'ın adımlarını normalize eden bir variantıdır.	[67]
Adam	Momentum ve Adagrad yöntemlerinin birleştirilmesiyle oluşan bir yöntemdir.	[68]

1.7.4. Aktivasyon Fonksiyonları

Aktivasyon fonksiyonları, bir sinir ağındaki her bir nöronun girdi verilerine göre çıkış değerini belirlemek için kullanılan fonksiyonları ifade eder. Aktivasyon fonksiyonları, girdi verilerinin bir nöron tarafından "aktif" veya "pasif" olarak değerlendirilmesine göre çıkış değerlerini sınırlar. Bu sınırlama, sinir ağının belirli bir girdinin ne kadar önemli olduğunu veya girdinin model tarafından nasıl işleneceğini belirler.

Aktivasyon fonksiyonları, modelin girdi verilerinde bulunan özellikleri ve nesneleri tanımasını ve belirli bir görev için gerekli olan çıkış değerini üretmesini sağlar. Örneğin, bir sınıflandırma probleminde, aktivasyon fonksiyonu verilen girdinin hangi sınıfı temsil ettiğini belirleyebilir. Farklı aktivasyon fonksiyonları farklı veri türlerine ve problemlere uygun olarak seçilebilir. Örneğin, sigmoid fonksiyonu sınıflandırma görevleri için uygun bir fonksiyondur ancak ReLU fonksiyonu görüntü işleme görevleri için daha uygun bir seçenek olabilir [69]. Doğru aktivasyon fonksiyonunun seçimi, modelin verimliliği ve doğruluğu açısından önemlidir. Aşağıda bazı aktivasyon fonksiyonları ve kullanım amaçları açıklanmıştır.

- "Unit Step veya Unipolar threshold" bu aktivasyon fonksiyonu, derin öğrenme modellerinde yaygın olarak kullanılmaz çünkü türevlenemez, bu da modelin gradyan tabanlı optimizasyon algoritmaları kullanılarak eğitilmesini zorlaştırır. Ek olarak, birçok derin öğrenme görevi için gerekli olan sürekli, doğrusal olmayan aktivasyon modellerini üretme kapasitesinden yoksundur. Çıkış değerleri {0, 1} şeklindedir. Denklem 10'da gösterildiği gibidir.

$$f(x) = \begin{cases} 1, & x > 0 \\ 0, & x \leq 0 \end{cases} \tag{10}$$

- "Sign veya Bipolar Threshold" aktivasyon fonksiyonu negatif giriş değerleri için -1 ve pozitif giriş değerleri için 1 veren basit bir aktivasyon fonksiyonudur. Birim Adım işlevi gibi, işaret işlevi de farklılaştırılamaması nedeniyle derin öğrenme modellerinde yaygın olarak kullanılmaz. Ek olarak, karmaşık, doğrusal olmayan aktivasyon kalıpları üretme kapasitesi sınırlıdır ve bu da onu birçok derin öğrenme görevi için daha az uygun hale getirir. Bununla birlikte, ikili sınıflandırma problemleri için destek vektör makinelerinde (SVM'ler) olduğu gibi bazı özel durumlarda yararlı olabilir. Çıkış değerleri {-1, 1} şeklindedir. Denklem 11'de gösterildiği gibidir.

$$f(x) = \begin{cases} 1, & x > 0 \\ -1, & x \leq 0 \end{cases} \quad (11)$$

- "Identy / Linear" aktivasyon fonksiyonu, giriş değerini olduğu gibi veren basit bir aktivasyon fonksiyonudur. Amacın sürekli, gerçek değerli çıktıları tahmin etmek olduğu regresyon problemlerinde genellikle çıktı katmanı için aktivasyon fonksiyonu olarak kullanılır. Ancak, birçok derin öğrenme görevi için gerekli olan doğrusal olmayan aktivasyon modellerini üretme yeteneğinden yoksun olduğu için, derin öğrenme modellerinin gizli katmanlarında bir aktivasyon işlevi olarak yaygın olarak kullanılmaz. Çıkış değerleri {-∞, +∞ } şeklindedir. Denklem 12'de gösterildiği gibidir.

$$f(x) = x \quad (12)$$

- "Sigmoid / Logistic" aktivasyon fonksiyonu, 0 ve 1 aralığında değerler veren sürekli, türevlenebilir bir aktivasyon fonksiyonudur. Sigmoid aktivasyon fonksiyonu, genellikle amacın tahmin etmek olduğu ikili sınıflandırma problemlerinde çıkış katmanı için aktivasyon fonksiyonu olarak kullanılır. Fonksiyon, girdi değerlerini, pozitif sınıfın bir olasılığı olarak yorumlanabilecek 0 ile 1 arasında bir değerle eşler. Bununla birlikte, Sigmoid aktivasyon fonksiyonunun derin öğrenme modellerinde doygunluk sorunu gibi bazı dezavantajları vardır. Giriş değerleri çok büyük veya çok küçük olduğunda, Sigmoid fonksiyonunun çıkışı 0 veya 1'e yaklaşır ve bu da model için yavaş bir öğrenme süreci ile sonuçlanabilir. Ek olarak, Sigmoid işlevi yavaş bir yakınsama ile sonuçlanabilecek yavaş bir eğime sahiptir. Çıkış değerleri (0, 1) şeklindedir. Denklem 13'te gösterildiği gibidir.

$$f(x) = \frac{1}{1 + e^{-x}} \quad (13)$$

- "TanH (Hiperbolik Tanjant)" aktivasyon fonksiyonu, -1 ve 1 aralığında çıkış değerleri veren sürekli, türevlenebilir bir aktivasyon fonksiyonudur. Tanh aktivasyon fonksiyonu, Sigmoid aktivasyon fonksiyonuna benzer, ancak (-1 ve 1) aralıkta çıkış değerleri üretme avantajına sahiptir. Bu, modelin daha karmaşık, doğrusal olmayan aktivasyon kalıpları üretmesini kolaylaştırabilir. Ek olarak Tanh aktivasyon fonksiyonu, Sigmoid aktivasyon fonksiyonundan daha dik bir eğime sahiptir ve bu da daha hızlı yakınsamaya neden olabilir. Ancak, Sigmoid aktivasyon fonksiyonu gibi, Tanh aktivasyon fonksiyonu da

giriş değerleri çok büyük veya çok küçük olduğunda doygunluk sorunu yaşayabilir. Bu model için yavaş bir öğrenme sürecine neden olabilir. Denklem 14'te gösterildiği gibidir.

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{14}$$

- "Softsign" aktivasyon fonksiyonu, -1 ve 1 aralığında değerler veren sürekli, türevlenebilir bir aktivasyon işlevidir. Softsign aktivasyon işlevi, Tanh aktivasyon işlevine benzer, ancak daha yumuşak bir eğime sahiptir, bu da sorun yaşama olasılığını azaltabilir (doygunluk sorunları). Ek olarak, orijinde iyi davranma özelliğine sahiptir, yani fonksiyonun gradyanının 0'a yakın 1'e yakın olduğu anlamına gelir. Bu özellik, Softsign aktivasyon fonksiyonunu, girdiyi işleyebilmesi gereken modeller için iyi bir seçim haline getirebilir (0'a yakın değerler). Ancak Softsign aktivasyon fonksiyonu, derin öğrenme modellerinde ReLU gibi diğer aktivasyon fonksiyonları kadar yaygın olarak kullanılmaz, bu da pratikte kullanımına ilişkin kaynak ve örnek bulmayı zorlaştırabilir. Denklem 15'te gösterildiği gibidir.

$$f(x) = \frac{1}{1 + |x|} \tag{15}$$

- "Rectified Linear Unit / ReLU" aktivasyon fonksiyonu, negatif giriş değerleri için 0 ve pozitif giriş değerleri için olduğu gibi çıkış değeri veren parçalı bir doğrusal aktivasyon fonksiyonudur. ReLU aktivasyon fonksiyonu, basitliği, hızı ve etkinliği nedeniyle derin öğrenme modellerinde en yaygın kullanılan aktivasyon fonksiyonlarından biridir. Sigmoid ve Tanh aktivasyon fonksiyonlarından farklı olarak, ReLU aktivasyon fonksiyonu doyma problemlerinden muzdarip değildir ve hızlı bir yakınsama oranına sahiptir. Ek olarak, yalnızca bir karşılaştırma işlemi ve maksimum işlem gerektirdiğinden, işlevin hesaplanması kolaydır. Bununla birlikte, ReLU aktivasyon işlevinin, giriş değerleri sürekli olarak negatifse bazı nöronların herhangi bir aktivasyon üretmeyi durdurabileceği " dying ReLU (ölen ReLU)" sorunu gibi bazı sınırlamaları vardır. Bu, ağ performansının düşmesine ve zayıf model doğruluğuna neden olabilir. Ek olarak, negatif giriş değerleri için ReLU aktivasyon işlevi tanımlanmamıştır, bu da bazı nöronların ölmesine ve asla iyileşmemesine neden olabilir. Bu sınırlamalar, "Leaky ReLU" ve "parametrik ReLU" gibi ReLU aktivasyon

fonksiyonunun varyantlarının geliştirilmesine yol açmıştır. Denklem 16'da gösterildiği gibidir.

$$f(x) = \begin{cases} x, & x > 0 \\ 0, & x \leq 0 \end{cases} \tag{16}$$

- "Sinusoid" aktivasyon fonksiyonu, sinüs fonksiyonuna dayalı olarak değerler veren sürekli, türevlenebilir bir aktivasyon fonksiyonudur. sinüzoid aktivasyon fonsiyonu, periyodik davranışı bir derin öğrenme modeline dahil etmek için kullanılabilir, bu da onu zaman serisi verilerini veya diğer döngüsel veri biçimlerini içeren problemler için uygun hale getirir. Ek olarak, fonksiyonun düz ve sürekli olma özelliği, onu birden fazla gizli katmana sahip modellerde kullanıma uygun hale getirebilir. Ancak sinüzoid, ReLU gibi diğer aktivasyon işlevleri kadar yaygın olarak kullanılmaz ve birçok derin öğrenme görevi için en iyi seçenek olmayabilir. Ek olarak, sinüs fonksiyonunun hesaplanmasını gerektirdiğinden ve bu da daha yavaş eğitim sürelerine neden olabileceğinden, fonksiyon hesaplama açısından pahalı olabilir. Ek olarak, sinüzoid fonsiyonu periyodiktir ve bu da daha az yorumlanabilir aktivasyon modellerine sahip modellerle sonuçlanabilir. Çıkış değerleri [-1, 1] şeklindedir. Denklem 17'de gösterildiği gibidir.

$$f(x) = \sin x \tag{17}$$

- "Gauss" aktivasyon fonksiyonu, değerleri Gauss (normal) dağılımına dayalı olarak çıkaran sürekli, türevlenebilir bir aktivasyon fonksiyonudur. Gauss aktivasyon fonksiyonu, derin bir öğrenme modeline düz, doğrusal olmayan davranış getirmek için kullanılabilir. Ek olarak, fonksiyon sonsuz türevlenebilir olma özelliğine sahiptir, bu da onu birden çok gizli katmana sahip modellerde kullanıma uygun hale getirebilir. Ancak Gauss aktivasyon fonksiyonu, ReLU gibi diğer aktivasyon fonksiyonları kadar yaygın olarak kullanılmaz ve birçok derin öğrenme problemi için en iyi seçenek olmayabilir. Ek olarak, fonksiyon, üstel fonksiyonun hesaplanmasını gerektirdiğinden hesaplama açısından pahalı olabilir ve bu da daha yavaş eğitim sürelerine neden olabilir. Gauss aktivasyon fonksiyonu 0 civarında simetriktir ve bu da daha az yorumlanabilir aktivasyon modellerine sahip modellerle sonuçlanabilir. Çıkış değerleri (0, 1] şeklindedir. Denklem 18'de gösterildiği gibidir.

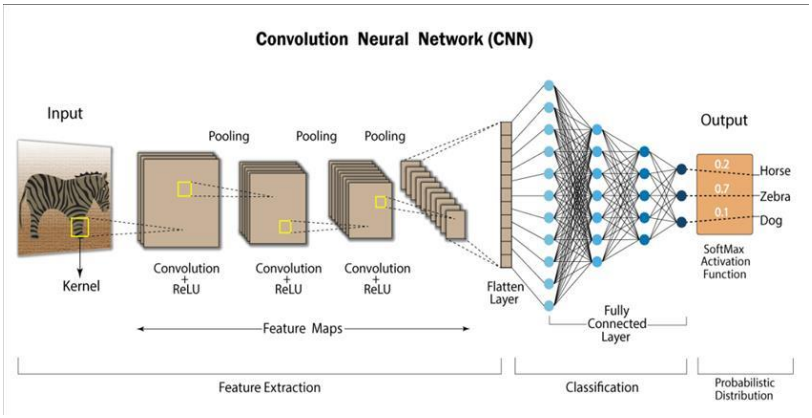
$$f(x) = e^{-x^2} \tag{18}$$

- “Softmax” [60] aktivasyon fonksiyonu, bir sınıflandırma probleminde birden çok çıktı nöronu için sınıf olasılıklarını hesaplamak için kullanılan bir aktivasyon fonksiyonudur. Softmax fonksiyonu her çıktı nöronunun olasılığını 0 ile 1 arasında normalize eder ve tüm çıktı nöronlarının olasılıkları toplamı 1'e eşittir. Denklem 19’da gösterildiği gibidir.

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}} \tag{19}$$

1.8. Evrişimli Sinir Ağları

Evrişimli Sinir Ağları (Convolutional Neural Networks veya CNN), görüntü sınıflandırma, nesne tespit ve video analizi gibi uygulamalarda yaygın olarak kullanılan bir tür yapay sinir ağıdır. CNN, ConvNets olarakta adlandırılır. CNN'lerin literatüre ilk girişi, 1998 yılında Lecun vd. tarafından yayınlanan "Gradient-Based Learning Applied to Document Recognition" (Gradient Tabanlı Öğrenme Belge Tanıma Uygulaması) adlı makaleyle [53] olmuştur. Bu makale, evrişimli sinir ağlarının uygulanmasıyla el yazısı rakam tanıma problemini ele almakta ve elde edilen başarılı sonuçları göstermektedir. Bu ağlar, girdi görüntüsünü tarayan ve analiz eden birçok katmanlı dizi olarak adlandırılan filtre veya çekirdekleri kullanarak veriyi işlemek için tasarlandı. CNN ayrıca özellik haritalarının uzaysal boyutlarını azaltmak için havuzlama katmanlarını da kullanır, bu da onlara girdideki çevirmelere karşı dirençli olmasını sağlar. Bu yapı, CNN'in etkili bir şekilde uzaysal özelliklerin hiyerarşilerini öğrenmelerine olanak tanır ve görüntü analiz görevleri için uygun hale getirir. Şekil 33’te CNN mimarisinin genel yapısı verilmiştir.



Şekil 33. Genel olarak CNN mimarisi [71].

Evrişimli sinir ağları genellikle input layer (giriş katmanı), convolutional layers (konvolüsyon katmanları), non-linear activation function (düzleştirilmiş doğrusal birim katmanı -ReLU), pooling layers (havuzlama katmanları), fully connected layer (tam bağlı katman) ve softmax activation function (sınıflandırma katmanı) gibi adımlardan veya bileşenlerden oluşur. Bu adımlar, ağı giriş verileri üzerinde eğitmek için farklı ağ mimarileri ve hiperparametre varyasyonlarıyla birçok kez tekrarlanır. Ağ eğitildikten sonra görüntü sınıflandırma, nesne algılama ve diğer ilgili görevler için kullanılabilir.

1.8.1. Input Layer (Giriş Katmanı)

Giriş katmanı, ağdaki ilk katmandır ve giriş görüntüsünü kabul etmekten sorumludur. Giriş katmanı tipik olarak her bir nöronun girdi görüntüsündeki tek bir pikseli işlemekten sorumlu olduğu birden çok nörondan oluşur. Giriş katmanının herhangi bir öğrenilebilir parametresi yoktur ve birincil amacı giriş görüntüsünü daha sonraki işlemler için sonraki katmanlara iletmektir. Giriş katmanının boyutunun seçimi ağ performansı için önemlidir. Çok geniş bir giriş katmanı ağı genişleteceğinden hesaplama yükünü ciddi bir şekilde artıracaktır.

Bir CNN'de, giriş görüntüsü tipik olarak ağa girmeden önce ilk olarak işlenir ve normalleştirilir. Bu ön işlem, görüntünün standart bir boyuta yeniden boyutlandırılmasını, görüntünün gri tonlamaya dönüştürülmesini veya piksel değerlerinin belirli bir aralığa

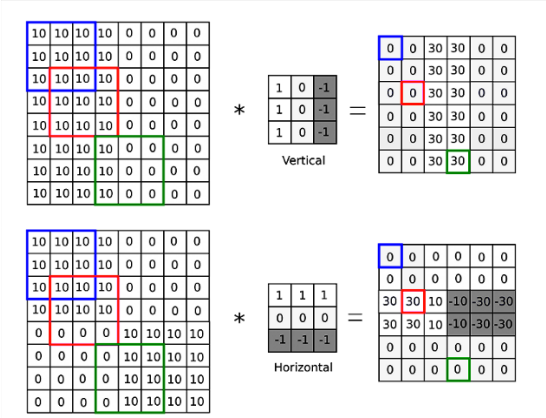
normalleştirilmesini içerebilir. Ön işlem adımı, girdi görüntüsünün ağın işlemesi için doğru formatta ve ölçekte olmasını sağlamaya yardımcı olduğundan önemlidir.

1.8.2. Convolutional Layers (Konvolüsyon Katmanları)

Konvolüsyon katmanları, görüntü verilerini işlemek için özel olarak tasarlanmış bir katman türüdür. Konvolüsyon katmanları, girdi görüntüsünün özelliklerini öğrenmekten ve girdi görüntüsünü, görüntünün farklı özelliklerini yakalayan bir dizi özellik haritasına dönüştürmekten sorumludur. Bir CNN'de konvolüsyon katmanı, giriş görüntüsüne bir dizi öğrenilebilir filtre uygular. Her filtre, görüntüdeki kenarlar, köşeler, dokular veya desenler gibi belirli bir özelliği öğrenmekten sorumludur. Katmanlar, filtreyi görüntü boyunca kaydırarak ve filtre ile girdi görüntüsü arasında eleman bazında çarpmalar gerçekleştirerek girdi görüntüsüne uygulanır. Bu işlemin sonucu, filtreler tarafından öğrenilen özellikleri yakalayan bir dizi özellik haritasıdır.

Konvolüsyon katmanları genellikle filtrenin çıktısına eklenen bir bias terimi içerir [53]. Bias terimi, her filtre için ayrı bir parametre olarak kullanılır ve modelin performansını iyileştirmeye yardımcı olur. Bias terimi, katmanın daha karmaşık bir özellik kümesini öğrenmesine izin verir. Örneğin, bir filtre yalnızca giriş verisinde belirli bir özelliği algılayabiliyorsa, bias terimi filtre çıktısını ayarlayarak daha geniş bir özellik kümesini yakalayabilir. Bu şekilde, konvolüsyon katmanları daha genel ve kapsamlı özellikler öğrenerek modelin daha karmaşık verileri daha iyi temsil etmesini sağlar. Bias terimi, katmanın her bir çıkış noktasına eklenir ve filtre çıktılarının doğrusal olmayan bir dönüşümden geçmesini sağlar. Bu, modelin daha karmaşık ilişkileri yakalamasına ve daha esnek bir öğrenme sürecine olanak tanır. Bias terimleri, genellikle başlangıçta rastgele değerlerle başlatılır ve model eğitimi sırasında optimize edilir. Sonuç olarak, konvolüsyon katmanlarına eklenen bias terimleri, modelin öğrenme kapasitesini artırarak daha karmaşık özelliklerin temsil edilmesine yardımcı olur ve genel performansı iyileştirir.

Filtrelerin boyutu, filtrelerin sayısı ve filtrelerin adımı, göreve ve giriş görüntüsünün boyutuna göre ayarlanabilen hiper parametrelerdir. Filtreler tipik olarak 3x3 veya 5x5 gibi küçüktür ve katmandaki filtre sayısı birkaç ila birkaç yüz arasında değişebilir. Genel olarak, konvolüsyon katmanları CNN mimarisinde çok önemli bir rol oynar ve derin öğrenme modellerinin girdi görüntüsünden üst düzey özellikleri öğrenmesi ve doğru tahminler yapması için gereklidir. Şekil 34’te konvolüsyon işlemi kenar bulma filtresi için verilmiştir.

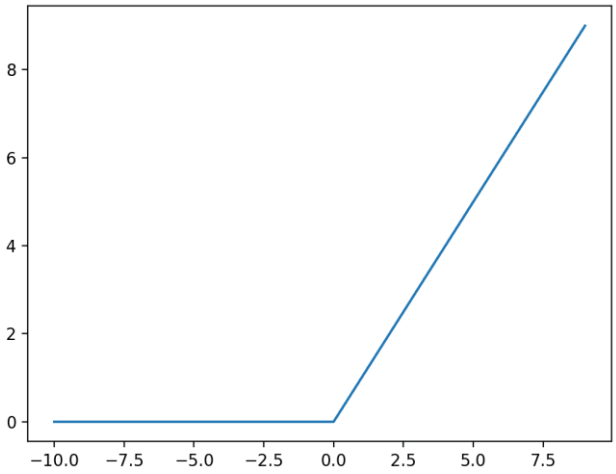


Şekil 34. Konvolüsyon işlemi

1.8.3. Non-Linear Activation Function (ReLU)

ReLu (Rectified Linear Unit), CNN'ler ve diğer derin öğrenme modellerinde yaygın olarak kullanılan bir aktivasyon işlevidir [72]. Basit bir eşikleme işlemine dayanan doğrusal olmayan bir fonksiyondur ve girdiyi çıktıya eşler. Fonksiyon denklem 16'da verildiği gibidir. Aktivasyon fonksiyonunun girişi x ve çıkışı f(x)'dir. Giriş (x) pozitifse, ReLU aktivasyon işlevi çıkış olarak x'i döndürür, x girişi negatifse, ReLU aktivasyon işlevi çıkış olarak 0 döndürür.

ReLU aktivasyon fonksiyonunun, basitliği, hızlı hesaplama süresi ve lojistik fonksiyon veya hiperbolik tanjant fonksiyonu gibi sigmoidal aktivasyon fonksiyonlarına kıyasla gelişmiş yakınsama dahil olmak üzere diğer aktivasyon fonksiyonlarına göre çeşitli avantajları vardır. ReLU, birçok derin öğrenme modelinde kullanılır ve modele doğrusal olmama özelliği getirme ve görüntü sınıflandırma ve nesne algılama gibi karmaşık görevlerde performansını artırmaya yardımcı olma yetenekleri nedeniyle özellikle CNN'lerde popülerdir. Şekil 34'te gösterildiği gibidir.



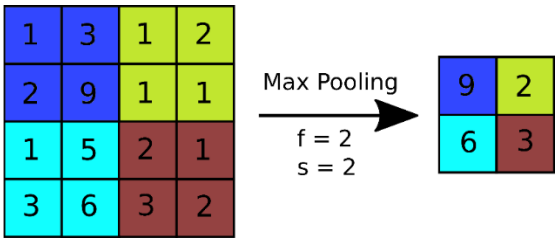
Şekil 35. ReLU aktivasyon fonksiyonu

1.8.4. Pooling Layers (Havuzlama Katmanları)

Havuzlama katmanları, konvolüsyon katmanları tarafından üretilen özellik haritalarının uzamsal boyutlarını azaltmak için kullanılan bir tür aşağı örnekleme katmanıdır. Katmanları birleştirmenin birincil amacı, modelin hesaplama karmaşıklığını azaltmak ve özellik haritalarındaki en önemli özelliklerin korunmasına yardımcı olmaktır. İki popüler havuzlama katmanı türü vardır: maksimum havuzlama ve ortalama havuzlama. Maksimum havuzlama, özellik haritasındaki bir dizi bitişik nörondan maksimum değeri döndürürken, ortalama havuzlama, aynı nöron kümesinden ortalama değeri döndürür. CNN'de, havuzlama katmanları konvolüsyon katmanları ile aktivasyon katmanları arasına yerleştirilir. Havuzlama katmanı, özellik haritasının her bir alt bölgesine bir havuzlama işlemi uygulayarak azaltılmış uzamsal boyutlara sahip alt-örneklenmiş bir özellik haritası üretir. Bu aşağı-örnekleme işlemi, modelin hesaplama karmaşıklığının azaltılmasına yardımcı olur ve aynı zamanda özellik haritalarındaki en önemli özelliklerin korunmasına yardımcı olarak ağın görünmeyen verilere daha iyi genelleme yapmasına olanak tanır.

Havuzlama işleminin boyutu ve adımı, göreve ve giriş görüntüsünün boyutuna göre ayarlanabilen bir hiperparametredir. Yaygın olarak kullanılan havuzlama boyutları 2x2 veya 3x3'tür. Genel olarak, havuzlama katmanları CNN mimarisinde önemli bir rol oynar ve görüntü sınıflandırma, nesne algılama ve semantik bölümleme gibi görevler için derin

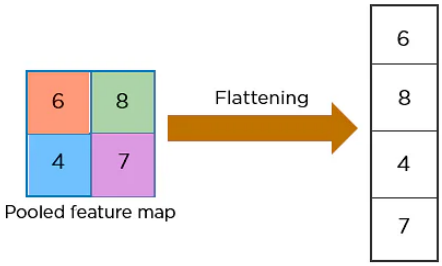
öğrenme modellerinde yaygın olarak kullanılır. Şekil 36'da max havuzlama örneği verilmiştir. Burada f, filtrenin boyutudur (2x2) ve s, filtrenin kaydırılma adımıdır.



Şekil 36. Max havuzlama işlemi örneği

1.8.5. Flatten Layer (Düzleştirme Katmanı)

Flatten layer, bir sinir ağındaki bir katman türüdür. Bu katman, önceki katmandaki çok boyutlu girdileri tek boyutlu hale dönüştürür. Örneğin, resim verilerinin önceki katmandaki tensör şeklindeki çıkışları, bu katman aracılığıyla tek boyutlu bir vektör haline dönüştürülebilir. Flatten layer, önceki katmanlardaki öğrenmeleri tek boyutlu olarak alır ve sonrasında bu verilerin tam bağlı katmanlar veya diğer katmanlar tarafından işlenmesine olanak tanır. Şekil 37'de düzleştirme katmanı için örnek verilmiştir.



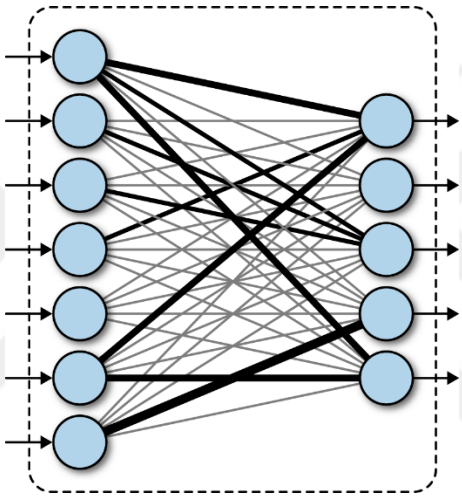
Şekil 37. Düzleştirme katmanı örneği

1.8.6. Fully Connected Layer (Tam Bağlı Katman)

Bu katman önceki katmandaki tüm nöronlarla bağlı olduğu için "tam bağlı" olarak adlandırılır. Bu, her nöronun önceki katmandaki tüm nöronların çıkışlarından girdiler alması

ve sonraki katmandaki tüm nöronlara çıkışlar vermesi anlamına gelir. Tam bağlı bir katmanda, her nöron girdileri ile ağırlıkları arasında bir bağlantı görevi yapar, ardından bir bias eklenir ve aktivasyon fonksiyonu hesaplaması yapılır. Aktivasyon fonksiyonu sigmoid, tanh, ReLU veya diğerleri gibi herhangi bir aktivasyon fonksiyonu olabilir.

Tam bağlı katmanlar, resim sınıflandırması, doğal dil işleme ve diğer uygulamalarda kullanılır. Daha önceki katmanlar tarafından öğrenilen daha basit temsillerden daha karmaşık temsillere geçmek için ve sonrasında nihai tahminleri üretmek için çıkış katmanında kullanılır. Şekil 38'de gösterildiği gibidir.



Şekil 38. Tam Bağlı Katman [73].

1.8.7. Softmax Activation Function (Sınıflandırma Katmanı)

Derin öğrenme mimarisinde sınıflandırma yapmak için kullanılan bir katmandır. Tam bağlantılı katmandan sonra gelir ve sınıflandırma işlemi yapar. Çıkış değeri sınıflandırılacak nesne sayısına eşittir. Tam bağlantılı katmandan alınan çıkış değeri kullanılarak sınıflandırma katmanı için ağırlık matrisi oluşturulur. Farklı sınıflandırıcılar kullanılabilir ancak en yaygın olarak softmax sınıflandırıcı tercih edilir. Sınıflandırma sonucu, her nesne için 0-1 aralığında bir değer üretir. Bu değer 1'e yakın olduğunda, ağın tahmini olan nesne anlaşılır. Denklem 19'da ifade edildiği gibidir.

1.9. Nesne Yakalama, Segmentasyon, Kümeleme

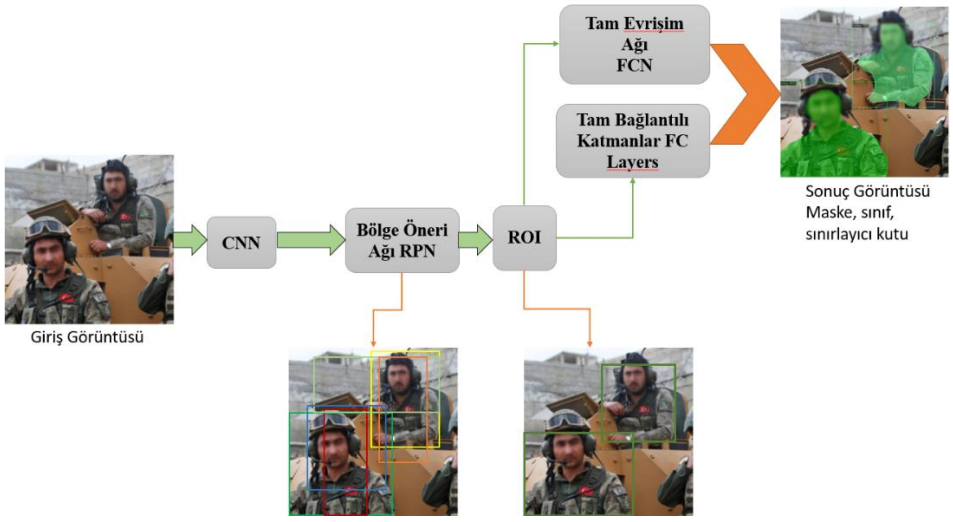
- Nesne yakalama, bir görüntü içindeki nesnelerin belirlenmesi ve pozisyonlarının tahmin edilmesi için kullanılan bir görüntü işleme tekniğidir. Genellikle derin öğrenme teknikleri ile yapılır ve çok sayıda veri seti kullanarak eğitilen bir model yardımıyla gerçekleştirilir. Nesne yakalama, farklı alanlarda kullanılabilir, örneğin güvenlik sistemleri, video analitik, araba veya drone görüntüleri gibi.
- Görüntü segmentasyonu, görüntü içindeki nesne veya bölgelerin ayrıştırılması ve belirlenmesi için kullanılan bir görüntü işleme tekniğidir. Bu teknik, bir görüntü içindeki nesnelerin belirlenmesi veya ayrılması için kullanılır. Örneğin, insanlar, arabalar, evler, doğal öğeler gibi. Segmentasyon, genellikle renk, doku veya yapısal benzerlik gibi farklı özelliklere dayalı algoritmalar kullanılarak gerçekleştirilir.
- Kümeleme, verileri benzer özelliklere sahip gruplara ayırma işlemidir. Bu teknik, veri madenciliği, makine öğrenmesi ve veri analitik gibi alanlarda kullanılabilir. Kümeleme, verilerin özelliklerine dayalı olarak benzer grupları tanımlamaya ve bu gruplar üzerinde analiz yapmaya yardımcı olur.

1.10. Mask R-CNN (Masks Region-based Convolutional Neural Network)

Mask R-CNN, bilgisayarlı görü problemlerinde, özellikle nesne örnek (instance) segmentasyonunda yaygın olarak kullanılan derin öğrenme tabanlı bir modeldir. Popüler nesne algılama modeli olan Faster R-CNN'nin [15] bir uzantısıdır. Mask R-CNN, Faster R-CNN'ye ek olarak sınırları (bounding box) bulunan nesnenin maskesini de tahmin etmeye çalışır. Bu maskeleme işlemi Faster R-CNN ile tahmin edilen sınırlayıcı kutudaki hedef nesne pixellerinin de kimliklendirilmesidir.

Mask R-CNN üç bileşenden oluşur: bir bölge öneri ağı (Region Proposal Network-RPN), en yüksek kesişimin birleşime oranı olan IoU (Intersection over Union) değerini alan bölge ağdaki ROI (Region of Interest) olarak adlandırılan kesişim bölgeleri ve son olarak tahminlerin yapıldığı tam bağlı katmandan sonrası tüm katmanlar. RPN, ilgilenilen bir nesneyi içerme olasılığı en yüksek olan bölge teklifleri üretir. Bu bölge teklifleri daha sonra her bir teklifin özelliklerini çıkarmak için özellik çıkarıcıya iletilir. Son olarak, tahmin kısmı önerileri farklı kategorilerde sınıflandırır ve her nesne için örnek maskesini tahmin eder.

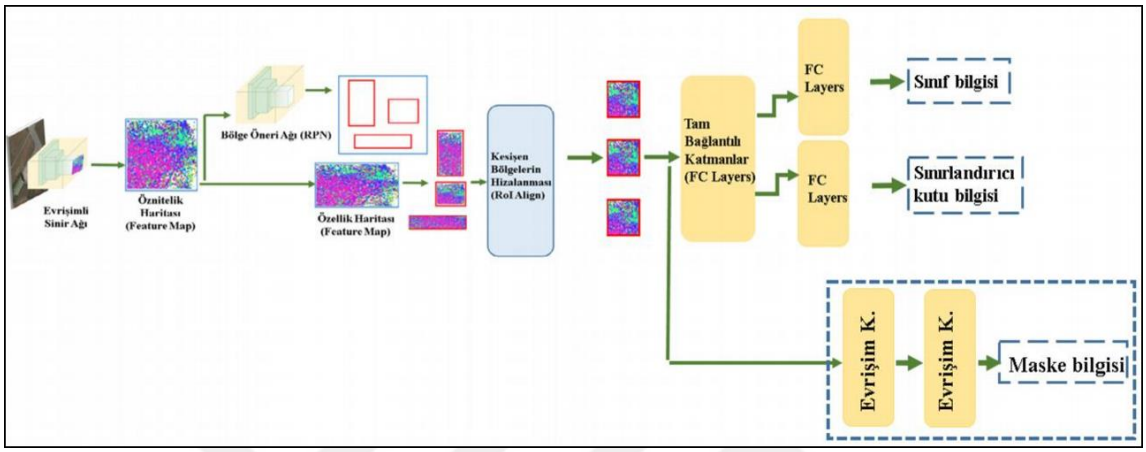
- Regional Proposal Network (RPN), bölge öneri ağı olarak geçmektedir. Mask R-CNN, nesne algılama ve örnek segmentasyon problemleri için RPN ile bir CNN'i birleştiren bir derin öğrenme modelidir. Modelin RPN bileşeni, bir görüntüde aday nesne bölgelerini önermekten sorumluyken, ağın geri kalanı nihai algılama çıktılarını üretmek için bu önerileri sınıflandırır ve iyileştirir. RPN ağı, potansiyel nesneler için bir görüntüyü verimli bir şekilde taramak için bağlantı kutuları ve kayan pencereler kullanır. Ağın geri kalanıyla paralel çalışır ve işlemesi için bir dizi bölge önerisi sağlar, bu da modelin çoklu nesne algılama ve bölümleme görevlerini verimli bir şekilde gerçekleştirmesini sağlar.
- ROI, RPNtarafından belirlenen bölgeler için iki ayrı çıktı oluşturulur. İlk çıktı, belirlenen bölgelerin sınıflandırmasıdır ve binary sınıflandırma işlemindeki "var/yok" seçeneğinden farklı olarak, bölgelerin hangi çoklu sınıfa (insan, araba, bina vb.) ait oldukları belirlenir. İkinci çıktı ise sınırlama kutusudur. Nesnenin içine alınması için sınırlama kutusunun konumu ve boyutu daha da geliştirilir. Şekil 39'da Mask R-CNN algoritmasının evreleri ve bunlara karşılık gelen çıktılar verilmiştir.



Şekil 39. Mask R-CNN algoritmasının evreleri ve bunlara karşılık gelen çıktılar

Mask R-CNN'nin en güçlü yönlerinden biri, bir görüntüdeki nesneleri aynı anda algılama ve her nesne için piksel düzeyinde segmentasyon maskeleri oluşturma yeteneğidir. Bu, amacın bir görüntüdeki bir nesnenin her bir örneğini tam olarak bölümlere ayırmak olduğu nesne intance segmentasyon gibi problemler için uygun hale getirir. Sonuç olarak Mask R-CNN, nesne algılama ve intance segmentasyonu tek bir ağda birleştiren güçlü bir

modeldir. Çeşitli bilgisayarlı görü problemlerinde yaygın olarak kullanılmaktadır. Mask R-CNN'nin genel yapısı şekil 40'ta verilmiştir. Mask R-CNN'de kayıp fonksiyonu (L) sınıflandırma kayıpları (L_{cls}), maskeleme kayıpları (L_{mask}) ve çerçeve kayıplarının (L_{box}) toplamından oluşur [73]. Kayıp fonksiyonu Denklem 20'de gösterilmiştir.



Şekil 40. Mask R-CNN Genel yapısı [10].

$$L = L_{cls} + L_{mask} + L_{box} \quad (20)$$

2. YAPILAN ÇALIŞMALAR, BULGULAR VE İRDELEME

2.1. Giriş

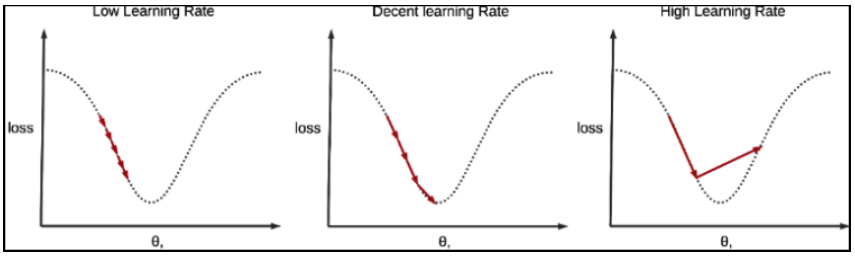
Bu tez çalışması, Google Colab ile Intel(R) Xeon(R) CPU @ 2.20GHz işlemcili, Tesla T4 ekran kartlı ve 26 GB birincil bellekten oluşan bilgisayar üzerinde Facebook Yapay Zekâ Araştırması (FAIR) tarafından geliştirilen Detectron2 [75] ile gerçekleştirilmiştir. Google Colab güçlü işlemci özelliğinden dolayı tercih edilmiştir. Detectron2 de ise Mask R-CNN'in uygulanması kolay ve hızlıdır bu sebeple tercih edilmiştir. Eğitim için 7 sınıftan oluşan 743 fotoğraf toplanmıştır. Bu veri seti toplam 1233 adet kamuflajlı asker içermektedir. Veri setinde her sınıf kendi içinde yaklaşık %90 eğitim ve %10 test olacak şekilde rastgele ayrılmıştır. Bu çalışmada görüntüler LabelMe aracıyla poligonal şekilde etiketlenerek eğitime hazır hale getirilmiştir. Çalışmada kullanılan görüntülerin dağılımı tablo 1'de verilmiştir.

Tez çalışmasında derin öğrenmenin önemi vurgulanmaya çalışılmış, matematiksel karmaşıklığı yüksek bir problemle kanıtlanmak istenmiştir. Mask R-CNN, kamuflajlı asker görüntüleri için örnek bölütleme (instance segmentation) başarılı bir şekilde sağlamıştır.

Bu çalışmada Mask R-CNN'in tercih edilmesinde birçok faktör vardır. Pikselden piksele hizalama yöntemi kullanması tercih edilmesindeki en önemli etmendir. Bu yöntem sayesinde, nesnelerin şekilleri ve konumları daha iyi korunur ve daha hassas maskeler elde edilir. Bunun dışında instance segmentasyon özelliği olması, RoI Pooling yerine RoI Align kullanılması tercih sebebidir. RoI Align, RoI Pooling'den farklı olarak, sınırlayıcı kutuların tam sayı değerlerine yuvarlanmasını önler ve interpolasyon yaparak özellikleri çıkarır. Bu durumda hassas bilgisayarlı görü problemlerinde iyi sonuçlar getirmektedir.

Öğrenme oranının etkisini de gözlemlemek adına, derin öğrenme literatüründe öğrenme oranının çürümesi, ya da küçültülmesi olarak geçen gamma (γ) çarpanıyla belirli adımlarda küçültülmüş ve toplam kayıp (total loss) sabit değerler alıncaya kadar eğitim işlemi devam ettirilmiştir. Sabit ve büyük bir öğrenme oranı kullanmak toplam hata oranının hızlı düşmesini sağlar fakat bir müddet sonra yerel minimum noktasının atlanmasına sebep olur. Sabit ve küçük bir öğrenme oranı kullanmak ise eğitim sürecini uzatacaktır bunun yanı sıra aşırı öğrenmeye (overfitting) sebep olabilir. Problemde bulunan sınıfların aşırı benzer

özellikler göstermesi de göz önüne alındığında küçülen bir öğrenme adımı yaklaşımının önemi gösterilmeye çalışılmıştır. Gamma çarpanının etkisi şekil 41'de gösterilmiştir.



Şekil 41. Gamma çarpanının öğrenme sürecine etkisi: düşük-sabit öğrenme oranı, azalan öğrenme oranı, yüksek-sabit öğrenme oranı [76]

2.2. Askeri Kamuflaj Desenlerinin Karmaşıklığının Tanımlanması

Askeri kamuflaj desenleri algoritmik karmaşıklığı yüksek geometrik şekillerdir. Burada bahsedilen karmaşıklık desenin ne kadar düzenli veya düzensiz olduğuna bağlıdır. Düzenli bir desen, basit bir kural veya formül ile tanımlanabilir ve bu da onun algoritmik karmaşıklığını düşük hale getirir. Diğer bir deyişle düzenli bir desenin ardında belli bir düzen veya sistem bulunur ve bu düzeni anlamak veya tahmin etmek kolaydır. Matematiksel bir formülle ifade edilebilen bir geometrik desen düzenli bir desen olarak kabul edilebilir. Öte yandan düzensiz bir desen rastgele veya kaotik bir şekilde oluşmuştur ve herhangi bir kural veya formül ile tanımlanması mümkün değildir. Bu tür desenlerde bir düzen veya sistem bulunmaz ve bu nedenle tahmin etmek veya analiz etmek daha zordur. Doğadaki çeşitli yaprak desenleri gibi fenomenler örnek olarak verilebilir.

Bir desenin karmaşıklığının tanımı hesaplamalı karmaşıklık teorisi [77] ve Kolmogorov karmaşıklığı (algoritmik karmaşıklık) [78] ile yapılabilir. Bir kamuflaj desenin hesaplamalı karmaşıklığı (Computational Complexity), onu üretmek için gerekli olan kaynakların miktarı olarak tanımlanabilir. Bu kaynaklar zaman, alan, iletişim veya devre kapısı gibi farklı ölçümler olabilir [77]. Hesaplamalı karmaşıklık teorisi, problemleri kendi zorluklarına göre sınıflandırmaya ve bu sınıfları birbirleriyle ilişkilendirmeye odaklanan bir bilim dalıdır.

Kamuflaj deseninin hesaplamalı ve algoritmik karmaşıklığına ilişkin bir değerlendirme yapmak için, desenin nasıl oluşturulduğunu ve kullanılan algoritmaların veya formüllerin özelliklerini bilmek gerekmektedir. Genel olarak, kamuflaj desenleri karmaşık

ve genellikle rastgelelik veya özyineleme gibi tekniklerle oluşturulur. Algoritmik karmaşıklık, kamuflaj desenini üreten algoritmanın veya formülün karmaşıklığına karşılık gelir. Algoritmanın veya formülün uzunluğu, desenin özelliklerini belirleyen renklerin, şekillerin ve dağılımların sayısı ve ilişkileri gibi faktörlerin karmaşıklığını yansıtabilir. Örneğin, gürültü veya fraktal geometri gibi rastgelelik veya özyineleme teknikleri kullanılarak oluşturulan bir kamuflaj deseni, kullanılan algoritmanın karmaşıklığına bağlı olarak değişir. Algoritmanın daha karmaşık olması, desenin daha fazla ayrıntıya veya çeşitliliğe sahip olmasına neden olabilir. Hesaplamalı karmaşıklık ise, desenin oluşturulması için gereken kaynaklara bağlıdır. Bu kaynaklar, kullanılan algoritmanın işlem gücü, bellek ve zaman gibi unsurları içerebilir. Örneğin, çok büyük bir kamuflaj deseni üretmek için daha fazla işlem gücüne veya belleğe ihtiyaç duyulabilir. Bu durumda, desenin hesaplanması daha fazla kaynağa ihtiyaç duyabilir ve dolayısıyla hesaplamalı karmaşıklığı artar.

Bir kamuflaj desenin algoritmik karmaşıklığı, onu tanımlamak için gerekli olan en kısa programın uzunluğu olarak tanımlanabilir. Bu tanım Kolmogorov karmaşıklığı [78] olarak da bilinir. Kolmogorov karmaşıklığı, tanımsal karmaşıklık, Kolmogorov-Chaitin karmaşıklığı, stokastik karmaşıklık, algoritmik entropi veya program boyu karmaşıklığı olarak da bilinir [79]. Kolmogorov karmaşıklığı, bir verinin en kısa tanımının uzunluğunu ifade eder. Örneğin, bir bit dizisi olan “0101010101…” Kolmogorov karmaşıklığı düşüktür, çünkü bu diziyi tanımlamak için sadece 01'in sonsuz bir şekilde tekrarlandığını söylemek yeterlidir. Ancak “0110010111000...” gibi rastgele bir bit dizisinin Kolmogorov karmaşıklığı yüksektir, çünkü bu diziyi tanımlamak için daha kısa bir tanım bulmak zordur. Bir kamuflaj deseninin Kolmogorov karmaşıklığını belirlemek için, deseni en kısa şekilde tanımlayan bir program yazmak gerekmektedir. Bu programın uzunluğu, desenin algoritmik karmaşıklığını gösterir. Eğer bu programın uzunluğu, desenin boyutundan çok daha küçükse, desenin algoritmik karmaşıklığı düşük olarak kabul edilir. Eğer bu programın uzunluğu, desenin boyutuna yakın veya eşitse, desenin algoritmik karmaşıklığı yüksek olarak kabul edilir.

Veri setinde kullanılan desenlerin üretim formüllerini düşünecek olursak fraktal geometri, gürültü, rastgelelik ve yineleme gibi tekniklerin uygulaması ile üretilen bir desen belirli bir formülle üretilen desenden hesaplamalı karmaşıklık teorisine göre daha karmaşıktır denilebilir.

Algoritmik karmaşıklık üzerinden değerlendirmek gerekirse, karmaşıklık bir verinin en kısa açıklamasının uzunluğu olarak tanımlanır. Bir kamuflaj desenin algoritmik

karmaşıklığını bulmak için, onu en küçük parçalara ayırarak, her parçanın ne kadar bilgi içerdiğini ölçmek gerekir. Bu işlemi yapmak için, deseni sayısal bir veriye dönüştürmek ve onu sıkıştırmak mümkündür. Sıkıştırma, veriyi daha az yer kaplayacak şekilde yeniden kodlamak anlamına gelir. Sıkıştırma oranı, sıkıştırılmış verinin orijinal veriye oranıdır. Sıkıştırma oranı ne kadar düşükse, algoritmik karmaşıklık o kadar yüksektir. Bu sebeple veri setindeki desenlerin tekrarsızları düşünüldüğünde sıkıştırma işlemi az olacağı ya da hiç olmayacağından karmaşıklığın yüksek olduğu söylenebilir.

2.3. Kullanılan Yöntemler ve Teknolojiler

Bu çalışmada, çeşitli yapay zekâ ve görüntü işleme teknolojileri kullanılarak hedeflenen problemin çözümüne yönelik bir yaklaşım benimsenmiştir. Aşağıda, kullanılan yöntemler ve teknolojilerin daha detaylı açıklamaları alt başlıklar halinde verilmiştir.

2.3.1. Detectron2

Detectron2 [75], Facebook AI Research tarafından geliştirilen ve nesne tespiti ve görüntü segmentasyonu gibi bilgisayarlı görü görevleri için son teknoloji algoritmalar sunan bir çerçevedir. Detectron2, bilgisayarlı görü araştırmalarında hızlı ve esnek bir şekilde yeni fikirleri hayata geçirmek ve değerlendirmek için tasarlanmıştır. Detectron2 çatısı ile Mask R-CNN, Faster R-CNN, Fast R-CNN, RetinaNet, RPN, DensePose, TensorMask, PointRend gibi bilgisayarlı görü algoritmalarının uygulamaları yapılabilir. Detectron2'yi kullanmak için PyTorch adlı açık kaynaklı bir derin öğrenme kütüphanesi gereklidir.

2.3.2. PyTorch

PyTorch [80], Torch kütüphanesine dayalı bir açık kaynak makine öğrenme kütüphanesidir. Bilgisayarlı görü ve doğal dil işleme gibi çeşitli uygulamalar için kullanılmaktadır. PyTorch'un temel özellikleri arasında, grafik işlem birimleri (GPU) ile hızlandırılmış tensör hesaplamaları ve derin sinir ağları üzerinde çalışan bir bant tabanlı otomatik türev sistemi bulunur. Hem Python hem de C++ dillerinde yazılan PyTorch, yüksek ve düşük seviyeli API'ler sunarak esneklik sağlar. PyTorch'un avantajları arasında dinamik hesaplama grafikleri, kolay hata ayıklama imkânı ve modüler olması yer almaktadır.

2.3.3. TensorFlow ve TensorBoard

TensorFlow [81], Google tarafından geliştirilen ve açık kaynak kodlu olarak sunulan bir makine öğrenimi kütüphanesidir. Derin öğrenme alanında kullanılan bu kütüphane, sembolik bir matematik kitaplığı gibi işlev sunmaktadır. TensorFlow, çok boyutlu veri dizileri olan tensörler üzerinde işlemler yaparak sinir ağlarının eğitimini yapar. Bu kütüphane esnek bir yapıya sahiptir ve Python, R, C++, Java ve JavaScript gibi programlama dillerini destekler. CPU ve GPU üzerinde çalışabilme yeteneğine sahiptir ve mobil, web ve IoT cihazlarında kullanılmaya uygun hale getirilebilir. Google, TensorFlow'u 2015 yılında açık kaynak olarak yayınlamış ve o tarihten bu yana makine öğrenimi alanında en popüler kütüphanelerden biri haline gelmiştir. Hem araştırma hem de ticari projelerde yaygın olarak kullanılan TensorFlow, görüntü işleme, doğal dil işleme, ses tanıma ve bilgisayarlı görü gibi alanlarda başarılı uygulamaların geliştirilmesine katkı sağlamıştır.

TensorBoard, makine öğrenimi deneylerinin görselleştirilmesi ve paylaşılması için kullanılan bir araçtır. Bu araç sayesinde, modelin yapısı, kayıp ve doğruluk gibi metrikler, histogramlar, görüntüler ve daha fazlasını izleyebilir ve görselleştirebilirsiniz. TensorBoard, TensorFlow ile kurulur ve Google Colab üzerinden de kullanılabilir. TensorBoard, makine öğrenimi projelerinizin performansını analiz etmenize, modelinizi ayarlamanıza ve sonuçları paylaşmanıza yardımcı olur.

2.3.4. OpenCV

Opencv [82], açık kaynak kodlu bir görüntü işleme kütüphanesidir. 1999 yılında Intel tarafından başlatılan proje, daha sonra Itseez, Willow Garage, Nvidia, AMD, Google gibi şirketlerin ve toplulukların katkılarıyla geliştirilmeye devam etmiştir. BSD lisansı altında yayınlanan OpenCV, Windows, Linux, Mac OS, Android ve iOS gibi farklı platformlarda çalışabilmektedir. C++, C, Python, Java, Matlab gibi programlama dilleriyle kullanılabilen OpenCV, gerçek zamanlı bilgisayar görüsü uygulamalarında yaygın olarak tercih edilmektedir.

Opencv kütüphanesi, görüntü işleme ve makine öğrenmesine yönelik 2500'den fazla algoritma içermektedir [83]. Bu algoritmalar sayesinde yüz tanıma, nesne tanıma, hareket takibi, optik karakter tanıma, artırılmış gerçeklik gibi pek çok işlem kolaylıkla gerçekleştirilebilmektedir. OpenCV'nin temel bileşenleri arasında Core, HighGui, Imgproc,

Imgcodecs ve Videoio modülleri bulunmaktadır. Bu modüller, görüntüler üzerinde çeşitli işlemler yapmak için gerekli fonksiyonları ve veri yapılarını sağlamaktadır.

2.3.5. NumPy

NumPy [84], Python programlama dili için büyük, çok boyutlu diziler ve matrisler destekleyen, bu diziler üzerinde çalışacak üst düzey matematiksel işlevler ekleyen bir kütüphanedir. NumPy, 2005 yılında Travis Oliphant tarafından oluşturulmuştur. Açık kaynak kodlu bir projedir ve özgürce kullanabilirsiniz. NumPy'nin atası Numeric, ilk olarak Jim Hugunin tarafından diğer birkaç geliştiricinin katkılarıyla oluşturuldu [85]. NumPy, Sayısal Python'un (Numerical Python) kısaltmasıdır. NumPy dizileri, Python listelerine kıyasla daha hızlı ve verimli bir şekilde işlenebilir. Matematiksel alanlarda doğrusal cebir, Fourier dönüşümü ve istatistik gibi konularda NumPy dizilerinden yararlanılabilir.

2.3.6. MatPlotlib

MatPlotlib [86], Python programlama dilinde veri görselleştirmek için kullanılan bir kütüphanedir. MatPlotlib ile 2 boyutlu ve 3 boyutlu grafikler, histogramlar, çubuk grafikleri, daire grafikleri, kutu grafikleri, saçılma grafikleri ve daha birçok çizim türü oluşturmak mümkündür. MatPlotlib, Matlab benzeri bir arayüze sahiptir ve verileri anlamak ve sunmak için etkili bir araçtır.

2.3.7. Google Colab

Colab, Google'ın bulut sunucularında kod çalıştıran bir web uygulamasıdır [87]. Python dilini destekler ve popüler kütüphaneleri içerir. Colab ile verileri analiz edebilir, grafikler çizebilir, modeller eğitebilir ve sonuçları paylaşabilirsiniz. Colab'ın arayüzü Jupyter Notebook ile aynıdır ve not defterlerini Google Drive veya GitHub hesaplarınızda saklayabilirsiniz. Colab'ı kullanmak için herhangi bir yazılım yüklemenize gerek yoktur ve Google size GPU veya TPU gibi güçlü bilgi işlem kaynaklarını ücretsiz olarak sunar.

2.4. Mask R-CNN'in Detectron2 ile Uygulaması

Mask R-CNN'in uygulaması için gerekli tüm adımlar gerek Google Colab komutları gerekse kod bloklarının olduğu görseller ile aşağıda verilmiştir.

1. Adım: **!apt-get update**

Komutu ile sistemdeki paket deposunun güncellenmesini sağlar ve en son paket listesini alır. Bu sayede güncel olmayan paketler güncellenebilir.

2. Adım: **from google.colab import drive**

drive.mount('/content/drive')

Bu komutlar, Google Drive ile Colab arasında bağlantı kurarak Google Drive'da bulunan dosyalara Colab üzerinden erişilebilmesini sağlar.

3. Adım: **!unzip "/content/drive/MyDrive/veriseti zip"**

Bu komut ile Google Drive'da bulunan veri seti zipten çıkarılır.

4. Adım: **!pip install pyyaml==5.1**

!pip install torch==1.8.0+cu101 torchvision==0.9.0+cu101 -f https://download.pytorch.org/whl/torch_stable.html

Komutlarıyla sırasıyla "PyYAML" ve "PyTorch" kütüphaneleri pip (Python Package Installer) paket yükleyicisiyle kurulur. Detectron2 ve Google Colab ile uyumlu sürümleri komutda görüldüğü gibidir.

PyYAML [85], Python programlama dilinde YAML (YAML Ain't Markup Language) veri biçimini işlemek için kullanılan bir kütüphanedir. YAML, insanlar tarafından okunması ve yazılması kolay olan bir veri serileştirme formatıdır ve genellikle yapılandırma dosyaları, veri değişimleri ve veri paylaşımı için kullanılır.

5. Adım: **!pip install detectron2 -f https://dl.fbaipublicfiles.com/detectron2/wheels/cu101/torch1.8/index.html**

Komutu ile Detectron2 kütüphanesi kurulur ve ardından reset atılır.

6. Adım: Şekil 42'de verilen kod bloğu çalıştırılır. Bu kod bloğu ile PyTorch, OpenCV, Numpy, MatPlotlib, Detectron2 vd. kütüphaneler çalışmaya dahil edilir.

```
import torch
assert torch.__version__.startswith("1.8")
import torchvision
import cv2
```

```
import os
import numpy as np
import json
import random
import matplotlib.pyplot as plt
%matplotlib inline

from detectron2.structures import BoxMode
from detectron2.data import DatasetCatalog, MetadataCatalog
```

Şekil 42. PyTorch, OpenCV, Numpy, MatPlotlib, Detectron2 vd. kütüphanelerin çalışmaya eklenmesi

7. Adım: Şekil 43'te verilen kod bloğu çalıştırılır. Bu kod bloğu, bir dizin içindeki JSON dosyalarını okuyarak, nesne tespiti veya segmentasyon için veri seti sözlüklerini oluşturan bir fonksiyonu tanımlar. Fonksiyonun girişleri, bir dizin yolunu (directory) ve sınıf etiketlerinin bir listesini (classes) alır. JSON dosyasındaki etiketler döngüyle gezilerek, her bir etiket için gerekli bilgiler alınır. Koordinatları kullanarak sınırlayıcı kutuyu (bbox) hesaplar, sınıf etiketinin indeksini alır, segmentasyon için çokgeni oluşturur ve bu bilgileri bir nesne (obj) olarak kaydeder. Her bir etiket için bu işlem tekrarlanır ve nesneler bir liste olan objs listesine eklenir. Son olarak, record sözlüğüne oluşturulan nesne listesi (annotations) eklenir ve dataset_dicts listesine kaydedilir. Bu, işlevin her bir JSON dosyası için sözlükleri oluşturduğu ve döndürdüğü bir liste oluşturur.

```python
def get_data_dicts(directory, classes):
    dataset_dicts = []
    for filename in [file for file in os.listdir(directory) if file.endswith('.json')]:
        json_file = os.path.join(directory, filename)
        with open(json_file) as f:
            img_anns = json.load(f)

        record = {}

        filename = os.path.join(directory, img_anns["imagePath"])

        record["file_name"] = filename
        record["height"] = 700
        record["width"] = 700

        annos = img_anns["shapes"]
        objs = []
        for anno in annos:
            px = [a[0] for a in anno['points']] # x coord
            py = [a[1] for a in anno['points']] # y-coord
            poly = [(x, y) for x, y in zip(px, py)] # poly for segmentation
            poly = [p for x in poly for p in x]

            obj = {
                "bbox": [np.min(px), np.min(py), np.max(px), np.max(py)],
                "bbox_mode": BoxMode.XYXY_ABS,
                "category_id": classes.index(anno['label']),
                "segmentation": [poly],
                "iscrowd": 0
            }
            objs.append(obj)
        record["annotations"] = objs
        dataset_dicts.append(record)
    return dataset_dicts
```

Şekil 43. Veri setinin eğitim için gerekli objelere dönüştürülmesi

8. Adım: Şekil 44'te verilen kod bloğu ile veri setinde train ve test olarak ayrı ayrı dizinlerde tutulan görüntüler "MetadataCatalog" nesnesine sınıf bilgileriyle birlikte kaydedilir.

```python
classes = [ 'turkey', 'america', 'china','france','russian', "german", "iraq"]

data_path = '/content/veriseti/'

for d in ["train", "test"]:
    DatasetCatalog.register(
        "my_" + d,
        lambda d=d: get_data_dicts(data_path+d, classes)
    )
    MetadataCatalog.get("my_" + d).set(thing_classes=classes)
```

Şekil 44. "MetadataCatalog" nesnesine veri setinin kaydedilmesi

9. Adım: Şekil 45'te verilen kod bloğu, Detectron2 kütüphanesini kullanmak için gerekli modül ve sınıfları içerir. İlk satırda bulunan "model_zoo" modülü hazır önceden

eğitilmiş model ağırlıklarını indirmek ve kullanmak için kullanılır. İkinci satırda, “detectron2.engine” modülünden “DefaultTrainer” ve “DefaultPredictor” sınıfları içe aktarılıyor. “DefaultTrainer”, modelin eğitimini yönetmek için kullanılırken, “DefaultPredictor”, eğitilmiş bir modeli kullanarak tahminler yapmak için kullanılır. Üçüncü satırda, “detectron2.config” modülünden “get_cfg” fonksiyonu içe aktarılıyor. Bu fonksiyon, Detectron2’nin yapılandırma dosyalarını yönetmek için kullanılır. Yapılandırma dosyası, modelin hiper parametrelerini ve yapılandırma ayarlarını içerir. Dördüncü satırda, “detectron2.utils.visualizer” modülünden “ColorMode” ve “Visualizer” sınıfları içe aktarılıyor. Bu sınıflar, nesne algılama sonuçlarını görselleştirmek için kullanılır. “ColorMode”, görsel sonuçlarda kullanılacak renk paletini belirlerken, “Visualizer sınıfı”, sonuçları görüntülemek için kullanılır. Son satırda, “matplotlib.patches” modülünden “Rectangle” sınıfı içe aktarılıyor. Bu sınıf, dikdörtgen sınırlayıcı kutuları (bounding box) çizmek için kullanılır.

```
from detectron2 import model_zoo
from detectron2.engine import DefaultTrainer, DefaultPredictor
from detectron2.config import get_cfg
from detectron2.utils.visualizer import ColorMode, Visualizer
from matplotlib.patches import Rectangle
```

Şekil 45. Detectron2 kütüphanesini kullanmak için gerekli modül ve sınıflar

10. Adım: Şekil 46’da verilen kod bloğu ile Detectron2’nin dosyası oluşturulur ve ayarları yapılandırılır.

```
cfg = get_cfg() # Yapılandırma Dosyasını Oluşturuyoruz

cfg.merge_from_file(model_zoo.get_config_file("COCO-InstanceSegmentation/mask_rcnn_R_50_C4_3x.yaml"))
# Yapılandırma Dosyasını Çeker ve Yapılandırma Dosyasına Ekler

cfg.DATASETS.TRAIN = ("my_train",) # Train Verilerimiz Yapılandırma Dosyasına Kaydeder
cfg.DATASETS.TEST = ()
cfg.DATALOADER.NUM_WORKERS = 4 # Çalışan Sayısı

cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url("COCO-InstanceSegmentation/mask_rcnn_R_50_C4_3x.yaml")
# Ağırlıkları Çeker ve Yapılandırma Dosyasına Ekler

cfg.SOLVER.IMS_PER_BATCH = 2 # Batch Size
cfg.SOLVER.BASE_LR = 0.001 # Learning Rate (Öğrenme Oranı)
#cfg.SOLVER.GAMMA = 0.99 # Learning Rate Azaltma Çarpımı
cfg.SOLVER.STEPS = [200] # Learning Rate Azaltma Adım Sayısı
cfg.TEST.EVAL_PERIOD = 70 # Eğitim Sırasında Modeli Değerlendirmek İçin Adım Sayısı

cfg.SOLVER.MAX_ITER = 50000 # İterasyon Sayısı
cfg.MODEL.ROI_HEADS.NUM_CLASSES = 7 # Sınıf Sayısı
```

Şekil 46. Detectron2’nin yapılandırma dosyasının düzenlenmesi

11. Adım: Şekil 47'de verilen kod parçası modelin sonuçlarını kaydetmek için bir klasör oluşturur ve eğitim sürecini başlatır. Eğer eğitim belirli iterasyonlarda kesilip başlatılacaksa eğitime kaldığı yerden devam etmesi için ilk satırda oluşturulan "output" dizini üçüncü satırda girdi olarak verilmelidir. Bu şekilde eğitim belirli bir iterasyonda durdurulup yapılandırma dosyası düzenlenip yeniden başlatılabilir.

```
os.makedirs(cfg.OUTPUT_DIR, exist_ok=True) # Model Sonucu İçin Klasör Oluşturur
trainer = DefaultTrainer(cfg) # Modeli Train Moduna Geçirir Yapılandırma Dosyası ile Birlikte
trainer.resume_or_load(resume=False) # Model Eğitimine 0'dan Başlamak İçin False Yapıyoruz
```

Şekil 47. Sonuç klasörünün oluşturulması ve eğitimin train moduna getirilmesi

12. Adım: **trainer.train()** Python kodu ile model eğitime başlar.

13. Adım: **%load_ext tensorboard**

%tensorboard --logdir output --host localhost --port 6009

Komutları TensorBoard'u kullanarak modelin eğitim ilerlemesini ve sonuçlarını görselleştirmek içindir. Birinci komut IPython hücresi içinde TensorBoard eklentisini yükler ve TensorBoard'u kullanmamızı sağlar. İkinci komut, TensorBoard sunucusunu başlatır ve belirtilen log dizinini "output" izler. "--host localhost" ifadesi, TensorBoard sunucusunu yerel makinede çalıştıracağını belirtir ve "--port 6009" ifadesi, kullanılacak olan bağlantı noktasını belirtir.

14. Adım: Şekil 48'de verilen kod parçası, eğitilmiş bir modelin test aşamasında kullanılması için gerekli ayarları yapılandırır ve test veri kümesini yükler. İkinci satırda gösterildiği gibi nesne algılama tahminlerini yaparken kullanılacak olan eşik değeri 0.7'dir.

```
cfg.MODEL.WEIGHTS = os.path.join(cfg.OUTPUT_DIR, "model_final.pth")
cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST = 0.7 # Test için Eşik Değerimiz
cfg.DATASETS.TEST = ("my_test", ) # Tets Verilerimiz Yapılandırma Dosyasına Kaydeder
predictor = DefaultPredictor(cfg) #Modeli Test Moduna Geçirir Yapılandırma Dosyası ile Birlikte
test_metadata = MetadataCatalog.get("my_test")
test_dataset_dicts = get_data_dicts(data_path+'test', classes)
```

Şekil 48. Modelin test veri kümesi üzerinde tahminler yapması için gerekli ayarlar

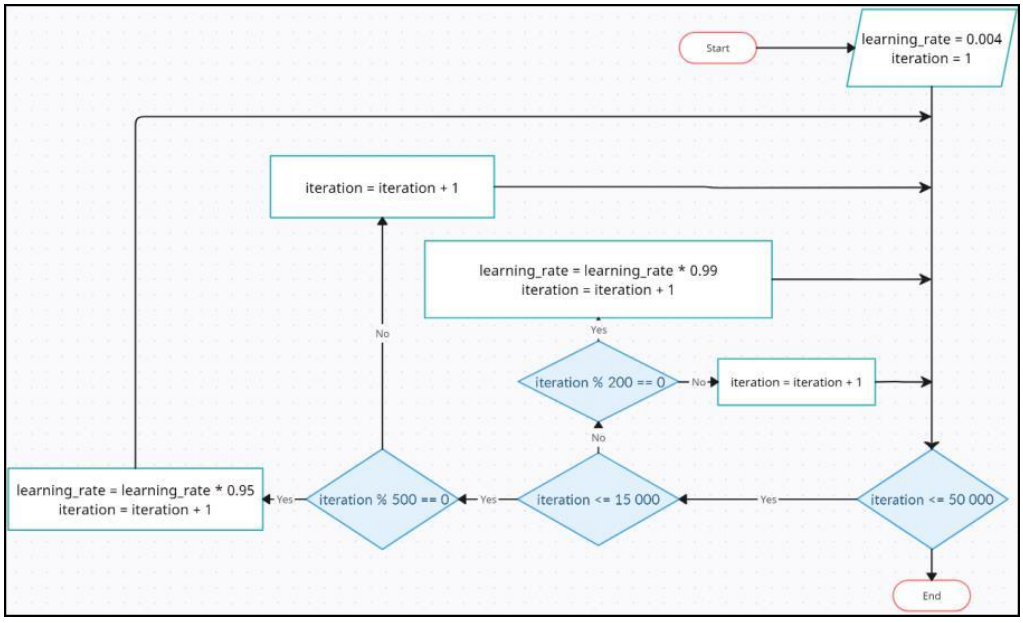
15. Adım: Şekil 49'da verilen kod parçası, modelin rastgele seçilen test örnekleri üzerinde tahminler yapmasını sağlar ve tahmin sonuçlarını görsel olarak görüntüler. Böylece, modelin performansını görsel olarak değerlendirilebilir.

```python
for d in random.sample(test_dataset_dicts, 30):
    img = cv2.imread(d["file_name"])
    outputs = predictor(img)
    predictions_list.append(outputs['instances'].scores.cpu().data.numpy())
    v = Visualizer(img[:, :, ::-1],
                   metadata=microcontroller_metadata,
                   scale=0.8,
    )
    v = v.draw_instance_predictions(outputs["instances"].to("cpu"))
    plt.figure(figsize = (20, 10))
    plt.imshow(cv2.cvtColor(v.get_image()[:, :, ::-1], cv2.COLOR_BGR2RGB))
    plt.show()
```

Şekil 49. Modelin rastgele görüntüler üzerinde test edilmesi

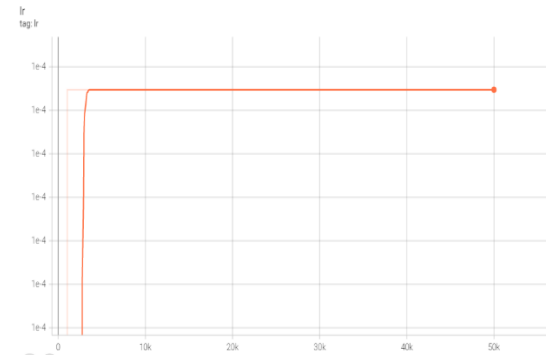
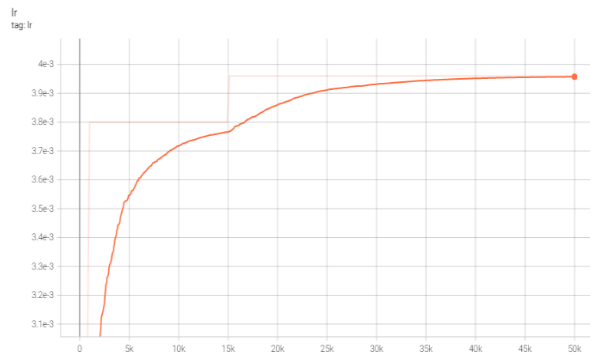
2.5. Eğitim Stratejisi

Eğitim işlemi 50 bin iterasyon olarak 2 farklı öğrenim stratejisi için Mask R-CNN algoritması ile gerçekleştirilmiştir. Strateji 1'de öğrenme oranı (learning rate-lr) sabit 0.001 olarak belirlenmiş ve 50 bin iterasyon boyunca değiştirilmemiştir. Strateji 2'de lr 0.004 değeriyle başlatılmış ilk 15 bin iterasyonda her 500 iterasyonda %5 düşmesi için 0.95 gamma çarpanıyla çarpılarak güncellenmiştir. 15 bin iterasyondan sonra her 200 iterasyonda %1 azaltacak şekilde 0.99 gamma çarpanıyla çarpılmıştır. Şekil 50 ile verilen akış diyagramı, strateji 2'de bahsedilen öğrenme oranının değişimini açıklamaktadır. Tablo 5'te 2 strateji için kullanılan lr'nin zamansal değişimi verilmiştir.



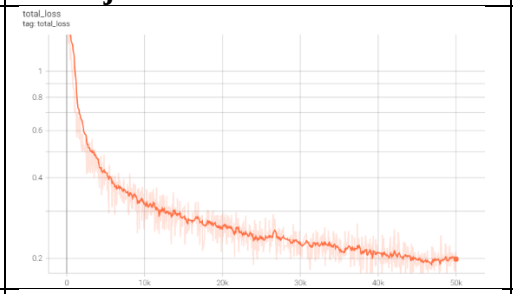
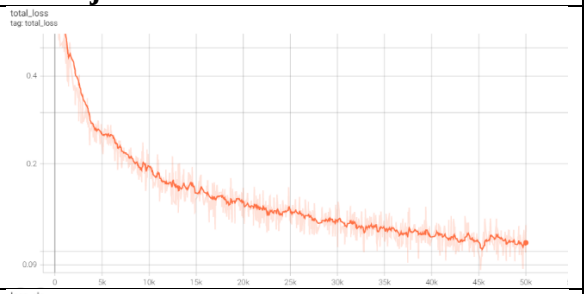
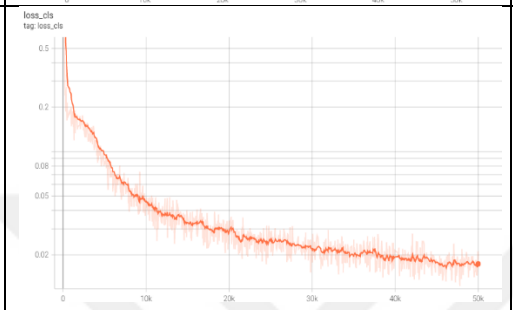
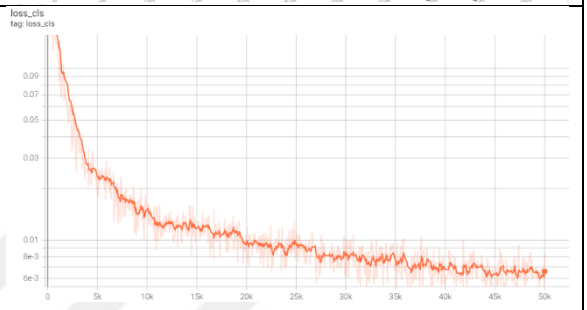
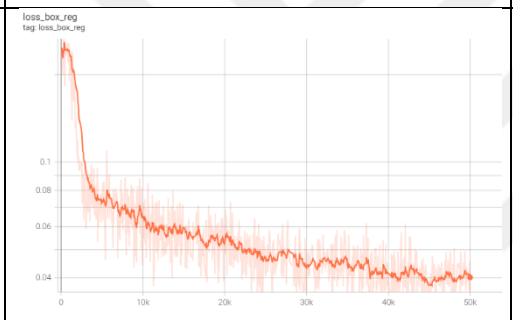
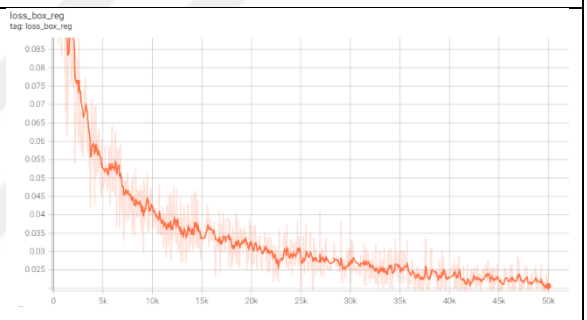
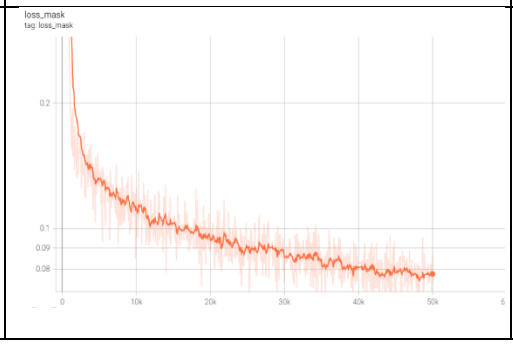
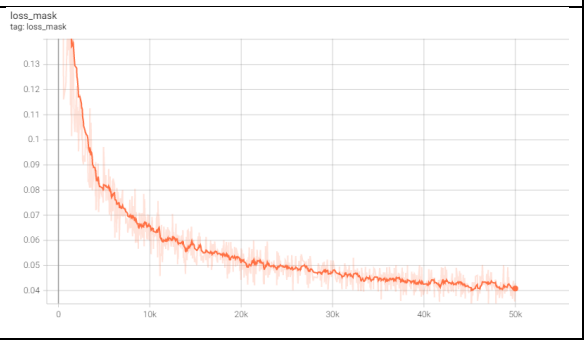
Şekil 50. Öğrenme oranının değişimini gösteren akış diyagramı

Tablo 5. Kullanılan öğrenme oranının zamansal değişimi

Strateji 1	**Strateji 2**
	

Modelin test görüntülerine uygulanması ile her iki strateji içinde başarılı sonuçlar elde edilse de strateji 1'de az da olsa sınıflandırma hataları görülmüştür. Strateji 1'de ortalama %93 doğrulukla strateji 2'de ise %96 doğrulukta tahmin değerleri aldığı görülmüştür. Tablo 6'da her iki strateji içinde toplam kayıp (total loss), sınıflandırma kaybı (classification loss), çerçeve kaybı (box loss) ve maskeleme kaybı (mask loss) grafikleri verilmiştir. Tablo 6'da ki metrikler incelendiğinde: Grafikler incelendiğinde strateji 1'de öğrenme işleminin çok aşırı yavaş olduğu görülmüştür. Hatta toplam kayıp grafiğinin 0.2 gibi bir değerde sabitlendiği ve öğrenme işleminin devam etmediği gözlenmiştir. Strateji 2'de ise toplam kaybın 0.1 altına indiği ve grafiğin yataylaşmadığı yani iterasyon devam etse daha da düşeceği görülmüştür. Bu durum aşırı öğrenmeye (overfitting) sebep olacağı için eğitim devam ettirilmemiştir.

Tablo 6. Eğitim metrikleri

	Strateji 1	**Strateji 2**
Toplam Kayıp		
Sınıflandırma Kaybı		
Çerçeve Kaybı		
Maskeleme Kaybı		

2.6. Görsel Sonuçlar

Bu başlık altında her iki strateji içinde Mask R-CNN ile elde edilen sonuçlardan örnekler verilmiştir.

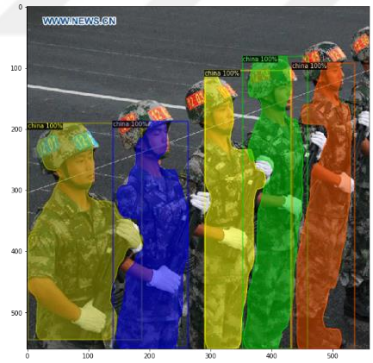
2.6.1. Strateji 1

Türkiye

Azerbaycan

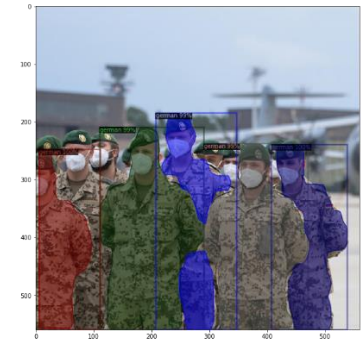
Amerika

Rusya



Çin

Fransa



Almanya

Irak

Şekil 51. Strateji 1 için görsel sonuçlar

2.6.2. Strateji 2

Şekil 52. Strateji 2 için görsel sonuçlar

3. SONUÇLAR

Literatürdeki çalışmalara bakıldığında kamuflajlı nesnelerin bölütlenmesi için görüntü işleme yöntemleri sonuç verse de bir takım eşik değerlerin seçimi için farklı yöntemler ek olarak tercih edilmiştir. Yapılan kamuflajlı nesne sınıflandırmalarında ise sınıflandırılan nesneler birbirinden farklı nesneler olduğundan desenden ziyade nesnelerin genel hatları sınıflandırmada etkin bir faktördür. Askeri kamuflaj desenlerinin matematiksel karmaşıklığı düşünüldüğünde görüntü işleme yöntemleriyle sınıflandırılması zor bir problemdir. Çalışmamızda segmente edilen nesnelerin sadece asker olması kullanılan algoritmanın sınıflandırma için desenleri baz almasını sağlamıştır. Tablo 6'da verilen eğitim metriklerinden sınıflandırma kaybına bakıldığında Mask R-CNN'nin bu karmaşık sınıflandırma problemini çözmede başarılı olduğu görülmektedir. Bu tez çalışmasında elde edilen doğruluk değerleri yaklaşık olarak strateji 1'de %96 ve strateji 2'de ise %98'dir. Özellikle strateji 2'de sınıflandırma kaybı oldukça iyi bir değerdedir. Strateji 1'de sınıflandırma kaybı 0.006'dır ve strateji 2'de ise bu değer 0.0047'dir. Yaprak sınıflandırması için AlexNet, GoogleNet, Vgg16, Vgg19 ve ResNet50'nin kullanıldığı [5]'deki çalışmada başarı oranları sırasıyla %99.72, %97.77, %99.12, %98.36 ve %99.15'dir ve bu çalışma sadece bir sınıflandırma çalışmasıdır. O sebeple bu tez çalışmasında segmentasyonun da bir zorluk seviyesi olduğunu göz önüne alırsak elde ettiğimiz sonuçlar başarılıdır. Yine [7]'de yapılan kamuflajlı nesnelerin sınıflandırması probleminde LBP ile öznitelikleri çıkarılıp sınıflandırması YSA, SVM ve KNN ile yapılan çalışmada başarı oranları sırasıyla %92, %89 ve %87'dir. Bu çalışmayla da kıyaslarsak Mask R-CNN'nin bize iyi sonuçlar verdiği söylenebilir. Şekil 42 ve 43 incelendiğinde Mask R-CNN'nin çok iyi segmentasyon yapabildiği de görülmektedir.

Genel olarak bu tez çalışması göstermiştir ki derin öğrenme bilgisayarlı görü alanında azımsanamayacak bir değere sahiptir. Sonuçlar, Mask R-CNN'in yüksek doğruluk oranlarına sahip olduğunu ve objeleri etkileyen faktörleri çok iyi modelleme kabiliyetine sahip olduğunu göstermiştir. Bununla birlikte, modelin eğitimi için gerekli olan büyük miktardaki veri ve yüksek performanslı GPU'lar gibi durumların mevcut olması gerekmektedir. Öte yandan U-Net [32] ve SegNet [33] gibi encoder-decoder şeklinde tasarlanmış mimarilerde ağın hız performansı yüksektir. Çünkü U-Net ve SegNet bölütleme yapmaktadır fakat Mask R-CNN örnek bölütleme yapmaktadır. Yani U-Net ve SegNet mimarileri gereği ayrı ayrı

nesne tespiti yapmamaktadır. Bu durum Mask R-CNN'den hızlı olmalarını sağlasa bile örnek bölütleme özelliğinden dolayı Mask R-CNN tercih edilmiştir. Ağın hızının yanı sıra Mask R-CNN [89]'da ki çalışmaya göre Mask R-CNN daha karmaşık nesnelerin bölütlenmesinde daha iyi sonuç vermektedir ve bu durumda yine tercih sebebi olmuştur.

Bu tez çalışmasındaki problemin çözümü için toplanan veri miktarı problemi çözmede yeterli olmuştur. Veri setinde bazı sınıflardaki örnekler diğer sınıflara göre çok fazla (örnek; Irak 236 adet veri içerir) bazıları ise çok az (örnek; Almanya 105 adet veri içerir) seçilmiştir. Buna rağmen çok olanın az olana karşı en azından bu problem için fark edilir bir üstünlüğü olmamıştır.

4. ÖNERİLER

4.1. Sonraki Araştırmacılara Öneriler

- Mask R-CNN gibi büyük modellerin eğitiminde öğrenme oranının azalan bir şekilde uygulanması eğitim sürecini kısaltacağı gibi belirli bir süre sonra yerel minimum noktasına takılma durumunu ortadan kaldıracaktır.
- Kamuflaj gibi zorlu sınıflandırma problemlerinde Mask R-CNN rahatlıkla tercih edilebilir. İleri çalışmalarda sınıf sayısı artırılarak farklı çalışmalar yapılabilir. Gerçek zamanlı uygulamalar için "Enet", YOLO gibi farklı algoritmalar test edilebilir.
- Askeri kamuflaj segmentasyonu ve sınıflandırılması için farklı uygulama alanları ve senaryolar incelenebilir. Örneğin, gece görüşü, termal görüntüleme, çoklu kamera sistemi gibi yöntemler kullanılabilir.
- Mask R-CNN algoritmasının farklı mimarilerle ve hiper parametrelerle karşılaştırılması, en uygun modelin belirlenmesine yardımcı olabilir.
- Askeri kamuflaj veri setinin genişletilmesi ve çeşitlendirilmesi, algoritmanın genelleştirme yeteneğini artırabilir.
- Mask R-CNN algoritmasının performansını artırmak için farklı backbone ağlarının (ResNet, MobileNet, DenseNet vb.) denenmesi ve karşılaştırılması.

4.2. Bu Çalışmayı Kullanacaklara Öneriler

- Askeri kamuflaj segmentasyon ve sınıflandırması, askeri araçların veya personelin görüntülerdeki konumlarını ve türlerini belirlemek için önemli bir uygulama alanıdır. Bu uygulama, askeri istihbarat, gözetleme ve keşif gibi alanlarda kullanılabilir.
- Güvenlik sistemlerinde kamuflajlı askerlerin tespit edilmesi ve izlenmesi için kullanılabilir. Özellikle sınır güvenliği, tesis güvenliği veya olay yerindeki tehlikelerin izlenmesi gibi alanlarda kullanılabilir.
- Medyada ortaya çıkan dezenformasyon içerikli haberlerin tespit edilmesi, doğruluğunun değerlendirilmesi gibi bir uygulama alanı olabilir.

- Uydu görüntülerinden askeri birliklerin konumlarını, sayılarını ve hareketlerini belirlemek için kullanılabilir.
- Kamuflajlı hayvanlar veya bitki türleri gibi doğal yaşamın parçası olan nesnelerin tespiti ve sınıflandırılması için proje kullanılabilir. Bu, biyolojik çeşitliliğin izlenmesi, doğal yaşam alanlarının korunması ve doğa koruma çalışmalarının planlanması için önemli bir araç olabilir.

5. KAYNAKLAR

1. Talas, L., Baddeley, R., & Cuthill, I. (2017). Cultural evolution of military camouflage, Philosophical Transactions of The Royal Society B Biological Sciences.

2. Hodson-Pressinger, S. (2004). KHAKI UNIFORM, 1848-49: FIRST INTRODUCTION BY LUMSDEN AND HODSON. Journal of the Society for Army Historical Research, 82(332), 341–347.

3. Singh, R. (2014). Study of Graphic Camouflage Patterns on Battle Uniform and improving the pattern used by Indian Army, Doctoral dissertation, PDPM INDIAN INSTITUTE OF INFORMATION TECHNOLOGY, DESIGN ANDMANUFACTURING JABALPUR.

4. https://www.camopedia.org Camopoedia the Camouflage Encyclopedia. 4 Ekim 2022

5. Doğan, F., & Türkoğlu, I. (2018). Derin Öğrenme Algoritmalarının Yaprak Sınıflandırma Başarımlarının Karşılaştırılması.

6. Gupta, A., & Gupta, U. (2018). Military Surveillance with Deep Convolutional Neural Network. In 2018 International Conference on Electrical, Electronics, Communication, Computer, and Optimization Techniques (ICEECCOT) (pp. 1147-1152).

7. Bayram, E., & Nabiyev, V. (2021). Classification of Camouflage Images Using Local Binary Patterns (LBP). In 2021 29th Signal Processing and Communications Applications Conference (SIU) (pp. 1-4).

8. Karatepe, İ. & Nabiyev, V. (2023). Military camouflage classification with Mask R-CNN algorithm. Communications Faculty of Sciences University of Ankara Series A2-A3 Physical Sciences and Engineering, 65 (1), 69-78.

9. He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask R-CNN. In 2017 IEEE International Conference on Computer Vision (ICCV) (pp. 2980-2988).

10. Ömeroglu, A.N., Kumbasar, N., Oral, E.A., & Özbek, I.Y. (2019). Mask R-CNN Algoritması ile Hangar Tespiti Hangar Detection with Mask R-CNN Algorithm. 2019 27th Signal Processing and Communications Applications Conference (SIU), 1-4.

11. Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 779-788).

12. Amri, M., Yedjour, D., El Amin Larabi, M., & Bakhti, K. (2022). Stadium Detection From Alsat-2 and Google-Earth Multispectral Images using YOLOv5 and Mask R-CNN. In 2022 4th International Conference on Pattern Analysis and Intelligent Systems (PAIS) (pp. 1-4).

13. Girshick, R. (2015). Fast R-CNN. In 2015 IEEE International Conference on Computer Vision (ICCV) (pp. 1440-1448).

14. Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2016). Region-Based Convolutional Networks for Accurate Object Detection and Segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 38(1), 142-158.

15. Ren, S., He, K., Girshick, R., & Sun, J. (2017). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. IEEE Transactions on Pattern Analysis and Machine Intelligence, 39(6), 1137-1149.

16. Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. In 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) (pp. 886-893 vol. 1).

17. Dai, Jifeng & Li, Yi & He, Kaiming & Sun, Jian. (2016). R-fcn: Object detection via region-based fully convolutional networks.

18. Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun (2015). Deep Residual Learning for Image Recognition. CoRR, abs/1512.03385.

19. Liu, W. et al. (2016). SSD: Single Shot MultiBox Detector. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds) Computer Vision – ECCV 2016. ECCV 2016. Lecture Notes in Computer Science(), vol 9905. Springer, Cham. https://doi.org/10.1007/978-3-319-46448-0_2

20. He, K., Zhang, X., Ren, S., Sun, J. (2014). Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds) Computer Vision – ECCV 2014. ECCV 2014. Lecture Notes in Computer Science, vol 8691. Springer, Cham. https://doi.org/10.1007/978-3-319-10578-9_23

21. C. . -C. Chu and J. K. Aggarwal, "The integration of region and edge-based segmentation," [1990] Proceedings Third International Conference on Computer Vision, Osaka, Japan, 1990, pp. 117-120, doi: 10.1109/ICCV.1990.139507.

22. Zafari, Sahar. (2014). SEGMENTATION OF OVERLAPPING CONVEX OBJECTS. 10.13140/RG.2.1.1339.0803.

23. Aydın, H., & Dizdaroğlu, B. (2017). Medical image segmentation with edge based level sets: Mobile client-server application. In 2017 International Conference on Computer Science and Engineering (UBMK) (pp. 994-999).

24. Afifi, Ashraf & Said, Ghoniemy & Zanaty, E. & El-Zoghdy, S.. (2015). New Region Growing based on Thresholding Technique Applied to MRI Data. International Journal of Computer Network and Information Security. 7. 61-67.

25. Zhu, S., & Yuille, A. (1996). Region Competition: Unifying Snakes, Region Growing, and Bayes/MDL for Multiband Image Segmentation. IEEE Trans. Pattern Anal. Mach. Intell., 18(9), 884–900.

26. Yezzi, A., Tsai, A., & Willsky, A. (2002). A Fully Global Approach to Image Segmentation via Coupled Curve Evolution Equations. Journal of Visual Communication and Image Representation, 13, 195-216.

27. Rousson, Mikaël & Deriche, R.. (2003). A Variational Framework for Active and Adaptative Segmentation of Vector Valued Images. 56- 61. 10.1109/MOTION.2002.1182214.

28. Zhou, H., Schaefer, G., Sadka, A., & Celebi, M. (2009). Anisotropic Mean Shift Based Fuzzy C-Means Segmentation of Dermoscopy Images. Selected Topics in Signal Processing, IEEE Journal of, 3, 26 - 34.

29. Windham, M.P. (1981). Cluster validity for fuzzy clustering algorithms. Fuzzy Sets and Systems, 5, 177-185.

30. http://pixelsciences.blogspot.com/2017/07/image-segmentation-k-means-clustering.html. Image Segmentation-K-means Clustering. 24 Kasım 2022

31. https://people.cmm.minesparis.psl.eu/users/beucher/wtshed.html. The Watershed Transformation. 24 Kasım 2022

32. Ronneberger, O., Fischer, P., Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. In: Navab, N., Hornegger, J., Wells, W., Frangi, A. (eds) Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015. MICCAI 2015. Lecture Notes in Computer Science(), vol 9351. Springer, Cham. https://doi.org/10.1007/978-3-319-24574-4_28

33. Badrinarayanan, V., Kendall, A., & Cipolla, R. (2017). SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. IEEE transactions on pattern analysis and machine intelligence, 39(12), 2481–2495. https://doi.org/10.1109/TPAMI.2016.2644615

34. Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, & Alan L. Yuille. (2017). DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs.

35. Liu F, Fang M. Semantic Segmentation of Underwater Images Based on Improved Deeplab. Journal of Marine Science and Engineering. 2020; 8(3):188. https://doi.org/10.3390/jmse8030188

36. Zhao, H., Shi, J., Qi, X., Wang, X., & Jia, J. (2016). Pyramid Scene Parsing Network. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 6230-6239.

37. Huang, G., Liu, Z., & Weinberger, K.Q. (2016). Densely Connected Convolutional Networks. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2261-2269.

38. Paszke, A., Chaurasia, A., Kim, S., & Culurciello, E. (2016). ENet: A Deep Neural Network Architecture for Real-Time Semantic Segmentation. ArXiv, abs/1606.02147.

39. Khoshdeli, Mina & Parvin, Bahram. (2018). Deep Leaning Models Delineates Multiple Nuclear Phenotypes in H&E Stained Histology Sections.

40. Bolya, D., Zhou, C., Xiao, F., & Lee, Y.J. (2019). YOLACT: Real-Time Instance Segmentation. 2019 IEEE/CVF International Conference on Computer Vision (ICCV), 9156-9165.

41. Yu, C., Wang, J., Peng, C., Gao, C., Yu, G., & Sang, N. (2018). BiSeNet: Bilateral Segmentation Network for Real-time Semantic Segmentation. European Conference on Computer Vision.

42. Benjdira, Bilel & Ammar, Adel & Koubaa, Anis & Ouni, Kais. (2020). Data-Efficient Domain Adaptation for Semantic Segmentation of Aerial Imagery Using Generative Adversarial Networks. Applied Sciences. 10. 1092. 10.3390/app10031092.

43. https://science.howstuffworks.com/military-camouflage.htm. How Military Camouflage Works. 19 Ekim 2022

44. https://worksthatwork.com/7/the-art-and-science-of-military-camouflage. The Art and Science of Military Camouflage. 19 Ekim 2022

45. Baumbach, Johannes (2012). Sparks, Emma (ed.). Advances in Military Textiles and Personal Equipment. Cambridge: Woodhead Publishing. ISBN 978-1845696993.

46. https://en.wikipedia.org/wiki/Desert_Battle_Dress_Uniform. Desert Battle Dress Uniform. 19 Ekim 2022

47. Torralba, A., Russell, B., & Yuen, J. (2010). LabelMe: Online Image Annotation and Applications. Proceedings of the IEEE, 98(8), 1467-1484.

48. Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. Science (New York, N.Y.), 313(5786), 504–507. https://doi.org/10.1126/science.1127647

49. Başaran, M. (2009). A new approach based on artificial neural networks for high order multivariate fuzzy time series. Expert Systems with Applications.

50. Vinícius Gonçalves Maltarollo, Káthia Maria Honório, & Albérico Borges Ferreira da Silva. (2013). Applications of Artificial Neural Networks in Chemical Problems. In Kenji Suzuki (Ed.), Artificial Neural Networks (p. Ch. 10). doi:10.5772/51275

51. https://medicalxpress.com/news/2018-07-neuron-axons-spindly-theyre-optimizing.html Why are neuron axons long and spindly? Study shows they're optimizing signaling efficiency. 2 Kasım 2022

52. Deng, L., & Yu, D. (2014). Deep Learning: Methods and Applications. Found. Trends Signal Process., 7(3–4), 197–387.

53. Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324.

54. Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. Psychological review, 65 6, 386-408.

55. Arbib, Michael. (1969). Review of 'Perceptrons: An Introduction to Computational Geometry' (Minsky, M., and Papert, S.; 1969). Information Theory, IEEE Transactions on. 15. 738- 739. 10.1109/TIT.1969.1054388.

56. McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. The bulletin of mathematical biophysics, 5(4), 115-133.

57. https://towardsdatascience.com/mcculloch-pitts-model-5fdf65ac5dd1 McCulloch-Pitts Neuron — Mankind’s First Mathematical Model Of A Biological Neuron. 17 Kasım 2022

58. Liao, Pengcheng & Sanders, Barry & Byrnes, Tim. (2021). Quadratic Quantum Speedup for Perceptron Training.

59. Widrow, B., & Hoff, M. E. (1960). Adaptive switching circuits (No. TR-1553-1). Stanford Univ Ca Stanford Electronics Labs.

60. Rumelhart, D.E., Hinton, G.E., & Williams, R.J. (1986). Learning internal representations by error propagation.

61. Wiener, N. (1949). A Heuristic Exposition of Wiener's Mathematical Theory of Prediction and Filtering.

62. Rumelhart, D.E., Hinton, G.E., & Williams, R.J. (1986). Learning representations by back-propagating errors. Nature, 323, 533-536.

63. Robbins, H.E. (1951). A Stochastic Approximation Method. Annals of Mathematical Statistics, 22, 400-407.

64. Scardapane, S., & Di Lorenzo, P. (2017). Stochastic Training of Neural Networks via Successive Convex Approximations. IEEE Transactions on Neural Networks and Learning Systems, 29, 4947-4956.

65. Polyak, B. T. (1987). Introduction to optimization. Optimization Software, Publications Division

66. Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. Journal of Machine Learning Research, 12(61), 2121–2159.

67. Zeiler, M.D. (2012). ADADELTA: An Adaptive Learning Rate Method. ArXiv, abs/1212.5701.

68. Kingma, D.P., & Ba, J. (2014). Adam: A Method for Stochastic Optimization. CoRR, abs/1412.6980.

69. Szandała, T. (2020). Review and Comparison of Commonly Used Activation Functions for Deep Neural Networks. ArXiv, abs/2010.09458.

70. Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324.

71. Yılmaz M. D. (2022), Malware classification with using deep learning. Computers and Informatics. 2(2): 21-40.

72. Nair, V., & Hinton, G. (2010). Rectified Linear Units Improve Restricted Boltzmann Machines. In Proceedings of the 27th International Conference on International Conference on Machine Learning (pp. 807–814). Omnipress.

73. https://www.oreilly.com/library/view/tensorflow-for-deep/9781491980446/ch04.html. Fully Connected Deep Networks. 28 Kasım 2022

74. Yayla, R., & Şen, B. (2020). Region-based Segmentation of Terrain Fields in SAR Images. In 2020 28th Signal Processing and Communications Applications Conference (SIU) (pp. 1-4).

75. Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, & Ross Girshick. (2019). Detectron2.

76. https://www.deeplearningwizard.com/deep_learning/boosting_models_pytorch/lr_scheduling/. Deep Learning Wizard. 26 Kasım 2022

77. Papadimitriou, C. (1994) Computational Complexity. Addison Wesley, Boston.

78. M. Li and P. Vitanyi, (1997). An Introduction to Kolmogorov Complexity and Its Applications, Springer, New York, pp. 1-185.

79. https://en.wikipedia.org/wiki/Kolmogorov_complexity. Kolmogorov complexity. 24 Kasım 2022

80. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., … Chintala, S. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. In Advances in Neural Information Processing Systems 32 (pp. 8024–8035). Curran Associates, Inc. Retrieved from http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

81. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., & others. (2016). TensorFlow: A system for large-scale machine learning.

82. Bradski, G. (2000) The OpenCV Library. Dr. Dobb's Journal of Software Tools, 120; 122-125.

83. https://mesutpiskin.com/blog/opencv-nedir.html. OpenCV Nedir?. 2 Şubat 2023

84. Harris, C., Millman, K., Walt, S., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N., Kern, R., Picus, M., Hoyer, S., Kerkwijk, M., Brett, M., Haldane, A., Río, J., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., & Oliphant, T. (2020). Array programming with NumPy. Nature, 585, 357–362.

85. https://tr.wikipedia.org/wiki/NumPy. NumPy. 6 Şubat 2023

86. Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. Computing in Science & Engineering, 9(3), 90-95.

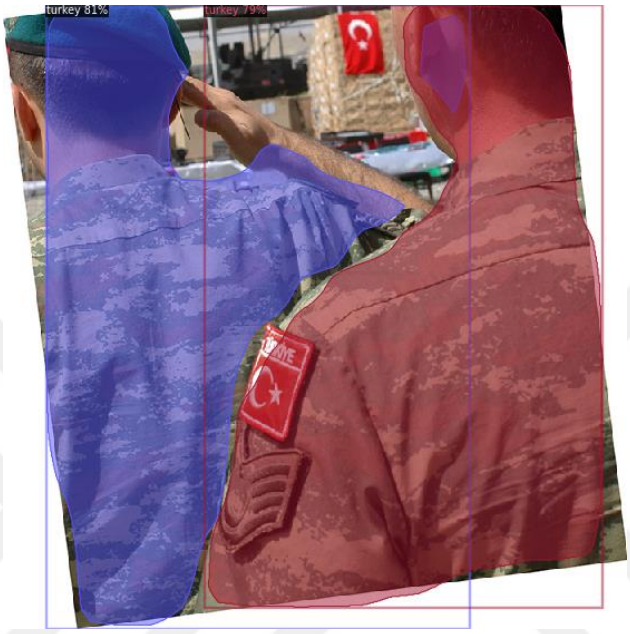
87. https://colab.research.google.com/. Colab nedir?. 6 Şubat 2023

88. https://pypi.org/project/PyYAML/. Project description. 6 Şubat 2023

89. Aarno Oskar Vuola, Saad Ullah Akram, & Juho Kannala. (2019). Mask-RCNN and U-net Ensembled for Nuclei Segmentation.

6. EKLER

EK A.1. Şekiller



(a) Arkası dönük iki Türk askeri (soldan sağa): %81, %79

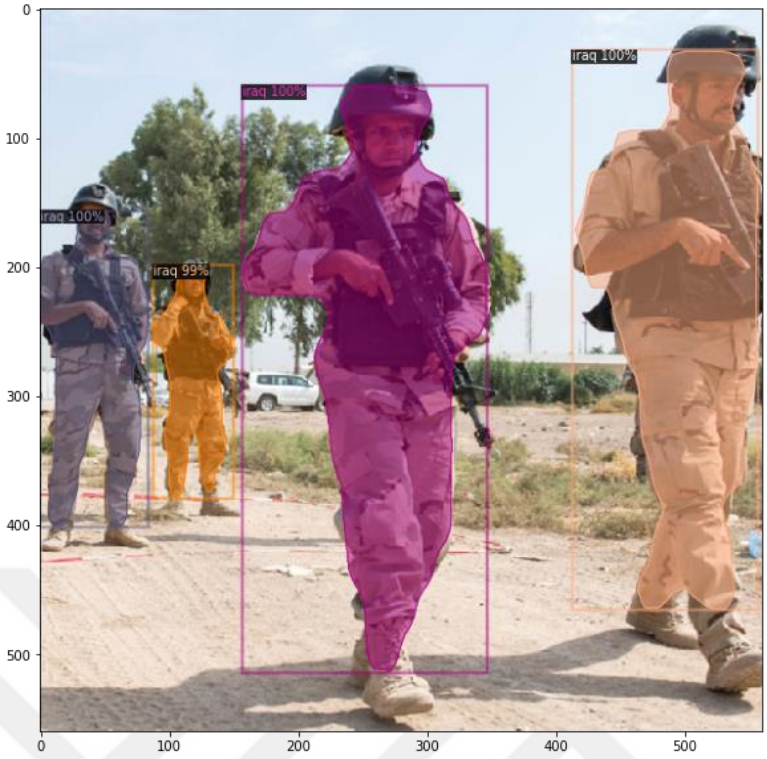
(b) Farklı açılardan Rusya (soldan sağa): %97, %100

(c) Silahlı ön cepheden Fransa askeri: %99

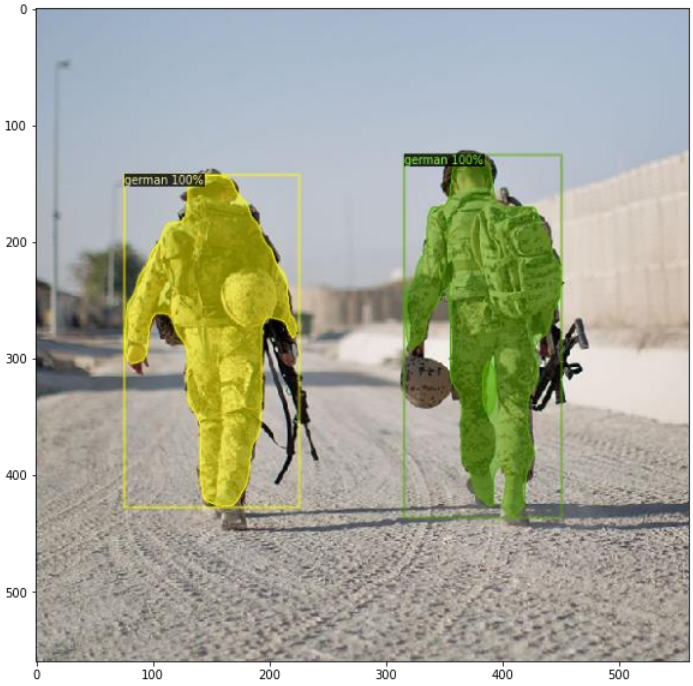
(d) Farklı pozisyonlarda Çin askerleri (üstten alta): %80, %99

(e) Yan açıdan Çin askeri: %93

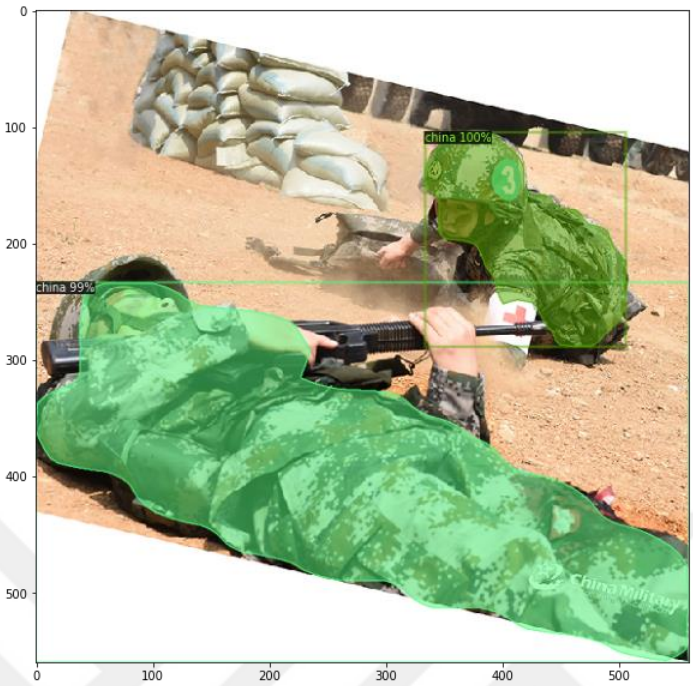
(f) Farklı açılardan Amerika askerleri (soldan sağa): %81, %89



(g) Silahlı ayakta Irak askerleri (soldan sağa): %100, %99, %100, %100

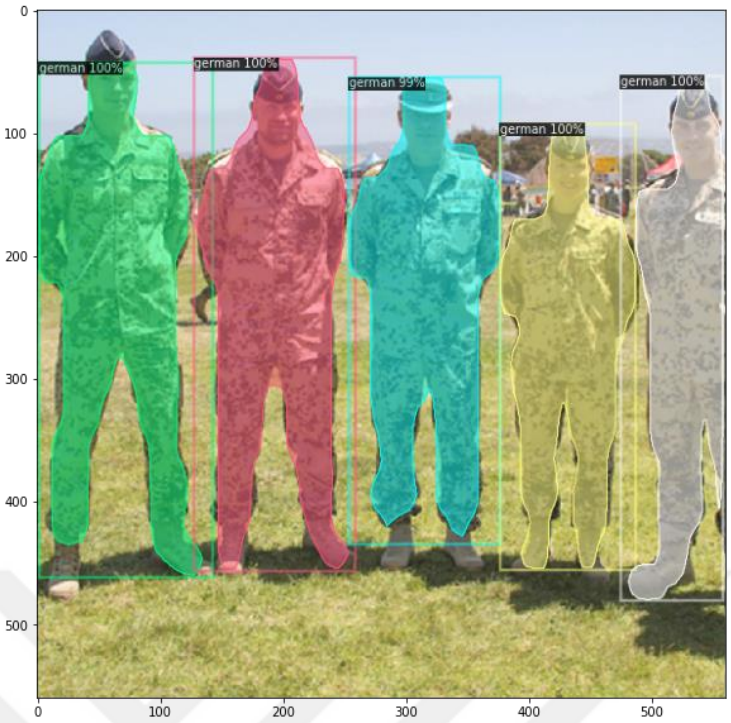


(h) Arkası dönük ayakta Alman askerleri: %100, %100



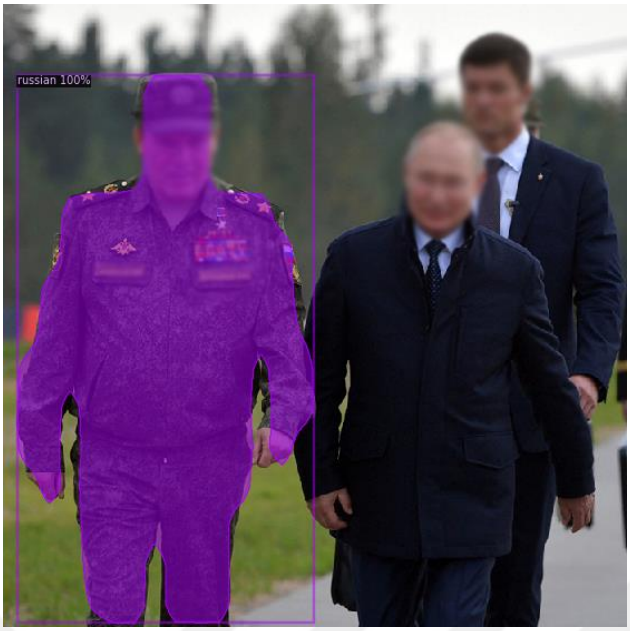
(i) Yerde farklı açılarda Çin askerleri (soldan sağa): %99, %100

(j) Ayakta Irak askerleri (soldan sağa): %99, %99, %99, %100



(k) Alman askerleri: Başarı oranları ~%100

(l) Farklı açılardan Amerikan askerleri (soldan sağa): %97, %100, %99, %100



(m) Kamuflajlı ve kamuflajsız insan görüntüleri: Tespit edilen Rusya %100

Figure 1(n) Türk ve Fransız askerler (soldan sağa): Fransa %76, Türkiye %100

ÖZGEÇMİŞ

Karadeniz Teknik Üniversitesi Bilgisayar Mühendisliği Bölümünden mezun olmuştur. 2019 yılında Karadeniz Teknik Üniversitesi Bilgisayar Mühendisliği Bölümünde Bilgisayar Bilimleri Anabilim dalında yüksek lisans eğitimine başlamıştır.

Yayınlar

- Karatepe, İ. & Nabiyev, V. (2023). Military camouflage classification with Mask R-CNN algorithm. Communications Faculty of Sciences University of Ankara Series A2-A3 Physical Sciences and Engineering, 65 (1), 69-78.

- Karatepe and V. Nabiyev, "Military Camouflage Classification with Mask R-CNN Algorithm," 2023 2nd International Conference on Innovative Academic Studies (ICIAS), 2023, pp 211-212, ISBN: 978-605-72325-2-6