

1.Bölüm

- Bölümün içeriği:
 - Veri oluşturma süreci ve Veri Ön işleme
 - Modelin Eğitilmesi
 - Grafik Yorumlama
 - Modelin Gerçek Zamanlı Testi

Temelde 4 başlıktan oluşur.

Veri oluşturma süreci ve Veri Önileme

- Verilerimi oluşturmak için kendi telefonumu ve kendi ortamımı kullandım.
- Örnek fotoğraf yanda ki gibi verilmiştir.
- **Çekilen görüntülerin arka planının beyaz olması sağlanarak modelin öğrenme ve test aşamasında ki başarımı arttırılmaya çalışılmıştır.**
- Bir video çekilerek video frame frame ayrılarak fotoğraflara dönüştürüldü.

Yaklaşık 500-600 civarında veri oluşturulmuştur.



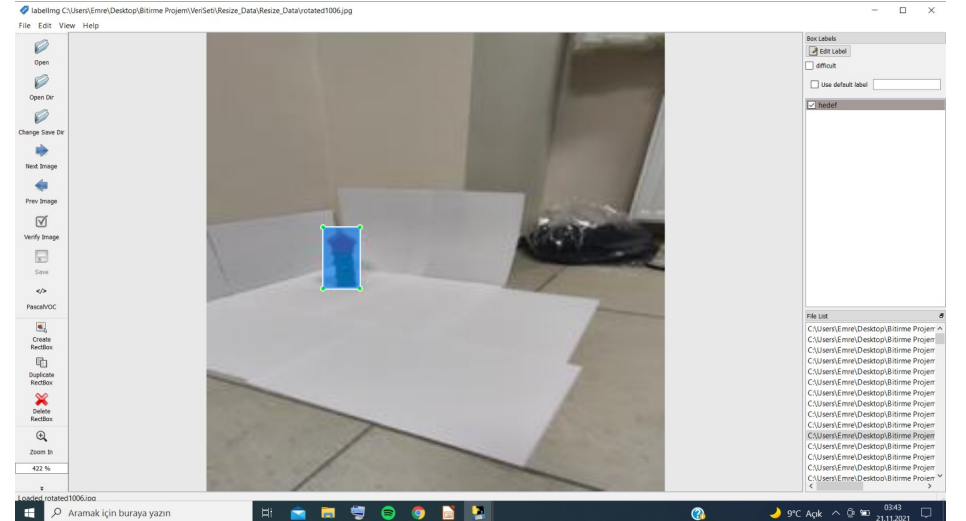
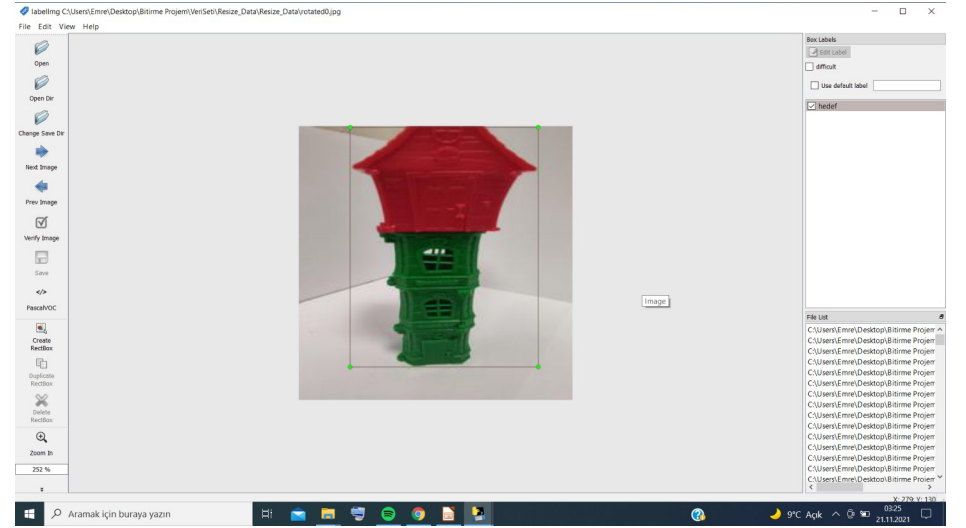
```
import cv2
vidcap = cv2.VideoCapture('C:/Users/Emre/Desktop/VID_20211109_022216.mp4')
success,image = vidcap.read()
count = 0
while success:
    cv2.imwrite("frame%d.jpg" % count, image)    # save frame as JPEG file
    success,image = vidcap.read()
    print('Read a new frame: ', success)
    count += 1
```

Video frame frame ayrıldıktan sonra 1920x1080 piksel görüntüler ortaya çıkmıştır. Modelin optimum giriş değerinden oldukça uzak piksele sahip görüntüler elde edilmiştir. Image_resizer programı kullanılarak 240x240 piksele düşürülmüştür.

- Görüntülerde rotasyon problemi yaşandığı için toplu bir şekilde 90 derece döndürme yapılmıştır. Döndürme yapılırken Spyder IDE ve PIL kütüphanesinin fonksiyonlarından yararlanılmıştır.
- Bu süreçte Google Colab üzerinde başkalarının hazırladığı kodlar ve veriseti kullanılarak eğitimler gerçekleştirilmiştir. Çeşitli derin öğrenme kavramlarına aşinalık kazanılmıştır.
- Etiketlenen veriler xml formatında aşağıda ki gibidir:

```
<annotation>
  <folder>Resize_Data</folder>
  <filename>rotated2.jpg</filename>
  <path>C:\Users\Emre\Desktop\Bitirme Projesi\VeriSeti\Resize_Data\Resize_Data\rotated2.jpg</path>
  <source>
    <database>Unknown</database>
  </source>
  <size>
    <width>224</width>
    <height>224</height>
    <depth>3</depth>
  </size>
  <segmented>0</segmented>
  <object>
    <name>hedef</name>
    <pose>Unspecified</pose>
    <truncated>1</truncated>
    <difficult>0</difficult>
    <bndbox>
      <xmin>40</xmin>
      <ymin>1</ymin>
      <xmax>195</xmax>
      <ymax>196</ymax>
    </bndbox>
  </object>
</annotation>
```

- Video çekilirken kamera hızlı hareket ettirildiğinden bazı frame'ler bulanık görüntülenmiştir. Bulanık görüntüler etiketleme sürecine dahil edilmemiştir.
- Farklı açılar ve uzaklıklardan fotoğraflar etiketlenmiştir.
- Etiketlemek için “**labelimg**” programı kullanılmıştır.
- Görüntüler “**Hedef**” tagıyla etiketlenmiştir.

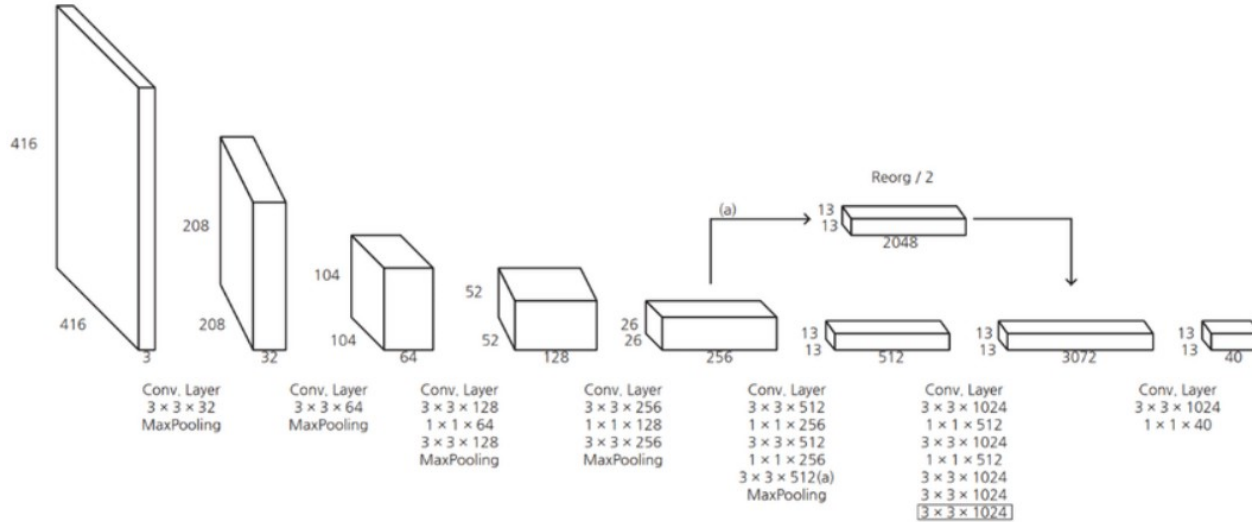


Model Seçimi ve Model Eğitimi Aşamasına Giriş

- Model seçimi konusuna karar verme sürecinde çeşitli literatür araştırmaları yapılmıştır. Bu araştırmalardan edinilen bilgiler kısaca aşağıda verilmiştir:
 - Çeşitli bilgisayarlı görü uygulamaları vardır. Bunlar temelde image classification , image segmentation, object detection olarak ayrılabilir. Nesne sınıflandırma ,nesnenin konumuna dair pek bir bilgi sağlamaz. Yalnızca görüntülerin içeriğinin ne olduğuyla ilgili bir çalışma yapar. Görüntü segmentasyonu ise görüntüyü parçalara bölütler. Bölütlenen fotoğraflarda nesneler değişik renklerle işaretlenir. Daha çok coğrafik çalışmalar ve insansız araçlarda tercih edilir. Bu sebepten dolayı görüntüden uzaklık algılama problemini çözmek için “**object detection**” tercih edilmiştir.
- You Only Look Once, yolunun kısaltmasıdır. Adından da anlaşılacağı gibi “bir kere bak” gibi bir anlama sahiptir. **Gerçek zamanlı projelerde yüksek fps** değerine sahip olduğu için modelin yolo seçilmesinin nedenide budur.
- Bütün araştırmalar sonucunda maixduino kartında çalışabilen **Yolo v2** modeli ,öznitelik çıkarımı içinse **mobilenet v1_0.75_224** omurgası seçilmiştir.
- Nesne tespiti modeli bir baş bir boyun ve bir omurga olmak üzere üç bölümden oluşur.

Yolo v2 Mimarisi

- Yolo evrişimli(Konvolüsyon) sinir ağlarını kullanan bir derin öğrenme modelidir.** Nesne tespiti ve nesne algılamak için kullanılır. Aşağıda yolo v2 mimarisi görülmektedir. Bu görsel giren görüntünün havuzlama, konvülüsyon ve diğer işlemlerden geçtiğini göstermektedir. Özetle Yapay sinir ağları kullanılarak öznetelik çıkarımı yapılmaktadır.



Omurga Algoritmasının Seçimi

Table 6. MobileNet Width Multiplier

Width Multiplier	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	70.6%	569	4.2
0.75 MobileNet-224	68.4%	325	2.6
0.5 MobileNet-224	63.7%	149	1.3
0.25 MobileNet-224	50.6%	41	0.5

Tablo incelendiğinde 1.0 Mobilenet ve 0.75 Mobilenet algoritmalarının arasında %2.2 değerinde doğruluk farkı olduğu görülmektedir. 0.75 Mobilenet algoritmasının her ne kadar doğruluk başarımı düşük olsa da giriş parametre değerleri bakımından çok büyük bir fark vardır. 0.75 Mobilenet algoritmasının giriş parametreleri 2.6 milyondur. 1.0 Mobilenete bakıldığında 4.2 milyon giriş parametreye sahiptir. Ram maliyetini düşündüğümüzde çok büyük bir fark yaratır. Bundan dolayı 0.75 Mobilenet-224 algoritması seçilmiştir.

Modeli eğitmek için Google Colab kullanılmıştır. Sahip olduğum bilgisayarın harici GPU birimi olmadığından google colab kullanmak bana fazlasıyla zaman kazandırmıştır. Bunların dışında yapılandırma gerektirmemesi ve paylaşma seçeneği olduğu için tercih edilmiştir.

- Oluşturulan colab defterinde “Maixduino” Kartını üreten firmanın github reposu kullanılmıştır.
- Kod özetle şunları yapar:
 - Github reposunu çeker.
 - Gerekli kütüphaneler ve environmentların uygun versiyonlarını kurar.
 - Drive adresime yüklediğim veri setini çeker.
 - Girdiğim hiperparametrelere sahip config dosyasını oluşturur.
 - Model girilen hiperparametrelere bağlı olarak eğitimi gerçekleştirir.
 - NCC yazılımı modelin mimarisini karta uygun hale getirir. Bu yazılımı colab defterine indirir ve eğitilmiş modeli mikrodenetleyicinin mimarisine uygun hale getirir.(.h5 to .kmodel)
 - Kodun kabataslak görüntüsü bir sonraki slaytta verilmiştir.

Colab Model Training Kodları aşağıda verilmiştir:

```
[ ] 1 !git clone https://github.com/sipeed/maix_train --recursive #kullanılan github deposu

[ ] 1 %cd /content/maix_train
2 %pwd

[ ] 1 !pip3 install -r requirements.txt #kullanılan kütüphanelerin versiyon gereksinimleri

▶ 1 import shutil
2 import os
3 !rm -r /content/maix_train/datasets/test-TFRecords-export.zip
4 !rm -r /content/maix_train/datasets/test_classifier_datasets.zip
5 !rm -r /content/maix_train/datasets/test_detector_xml_format.zip
6 !gdown https://drive.google.com/uc?id=1hVDGAtqihsGdiu22LCt2qf6bNlueifs_

[ ] 1 #shutil.move("test_detector_xml_format.zip","datasets")
2 %cd /content/maix_train

[ ] 1 !gdown https://drive.google.com/uc?id=1ugqTG0g7KmV94I6KKFC6Cb20lBY9Ad_K

[ ] 1 !python3 train.py init #config dosyası oluşturur

[ ] 1 !gdown https://drive.google.com/uc?id=17QlZfTJq7t4QNXH0GTZcVnTM-tLODpqg #kendi config dosyamı yüklüyor
2 shutil.move("config.py", "/content/maix_train/instance/config.py") #taşımaya

[ ] 1 !wget -P /content/maix_train/tools/ncc/ncc_v0.1 https://github.com/kendryte/nncase/releases/download/v0.1.0-rc5/ncc-linux-x86_64.tar.xz
2 %cd /content/maix_train/tools/ncc/ncc_v0.1
3 !tar xvf /content/maix_train/tools/ncc/ncc_v0.1/ncc-linux-x86_64.tar.xz

▶ 1 %cd /content/maix_train
2 !python3 train.py -t detector -z /content/maix_train/xml_format.zip train
```

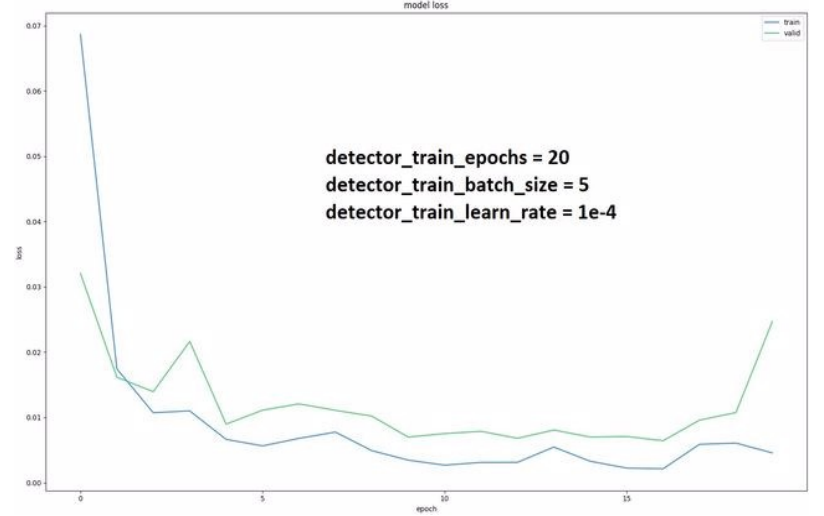
Modelin Input Parametreleri ve Katmanları

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 224, 224, 3)]	0
mobilenet_0.75_224 (Function)	(None, 7, 7, 768)	1832976
Total params: 1,832,976		
Trainable params: 1,816,560		
Non-trainable params: 16,416		
Model: "functional_3"		
Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 224, 224, 3)]	0
mobilenet_0.75_224 (Function)	(None, 7, 7, 768)	1832976
detection_layer_30 (Conv2D)	(None, 7, 7, 30)	23070
reshape (Reshape)	(None, 7, 7, 5, 6)	0
Total params: 1,856,046		
Trainable params: 1,839,630		
Non-trainable params: 16,416		

Model Eğitime Giriş

(1)

İlk eğitim yanda ki hiperparametrelerle gerçekleştirilmiştir. Görselde görüldüğü gibi loss değerleri gayet iyidir. Fakat 15.epochtan sonra modelin underfit ettiği söylenebilir. Gerçek zamanlı testi yapıldığında model “hedef” nesnesini tespit etmekte **başarılı** olmuştur.

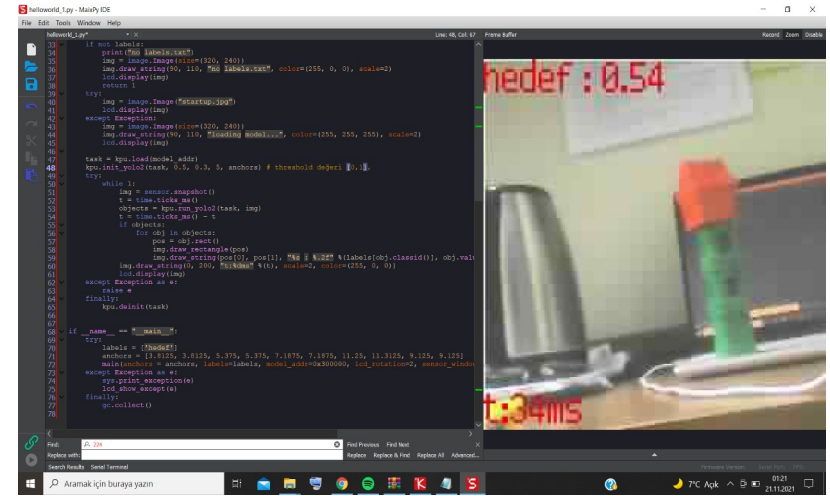


Model revize edilmeden yani 20.epochta ki çıktıyla gerçek zamanlı testi yapılmıştır.

Eğitilen Modelin Gerçek Zamanlı Testi

Threshold değeri eşik değerini ifade eder. Hedef nesnenin tespiti Modelin tahmini üzerine kurguludur. Modelin tahminleri görsel içerisinde kareler oluşturmasıyla gerçekleşir. Buna bağlı olarak bir sürü tahminde bulunabilir. Veya bir başka deyişle bir sürü dikdörtgen çizer. Fakat çizilen her dikdörtgen ekrana yansıtılmaz ve olasılık oranları da aynı değildir. Dikdörtgen içerisinde bulunan nesnenin hedef görseli olma durumuysa threshold ile anlaşılabilir veya tahmin edilebilir. **Threshold değeri şuan için 0.5 belirlenmiştir.**

Yanda ki görseldeyse eğitilen modelin 0.5 threshold eşik değeri kullanarak tespit ettiği hedef görseli görünmektedir. Çizilen bounding boxın bütün ekranı kapladığı ve aslında çokta başarılı olmadığı görülüyor. Modelin başarımını arttırmak için farklı hiperparametrelerle model yeniden eğitilecektir.



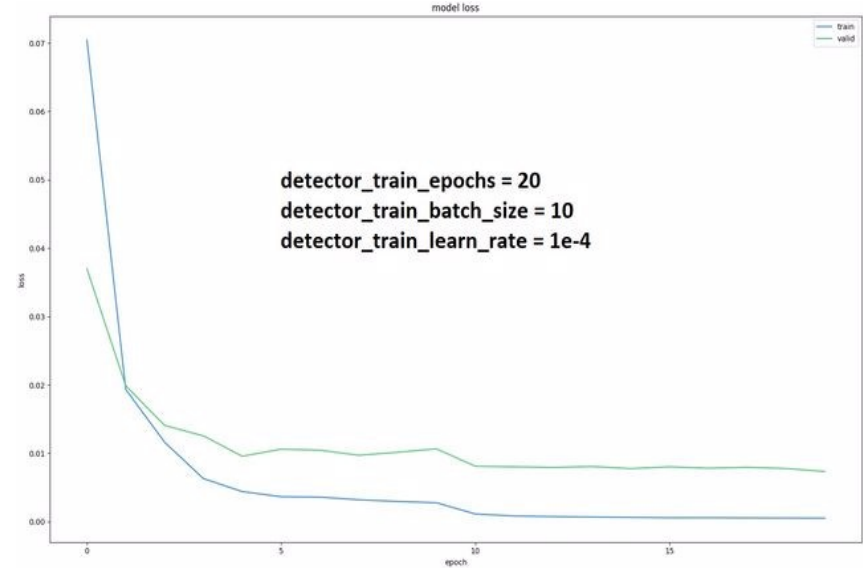
```
13 # -*- coding: utf-8 -*-
14 # Python 3.6.5
15 # YOLOv3
16 # YOLOv3
17 # YOLOv3
18 # YOLOv3
19 # YOLOv3
20 # YOLOv3
21 # YOLOv3
22 # YOLOv3
23 # YOLOv3
24 # YOLOv3
25 # YOLOv3
26 # YOLOv3
27 # YOLOv3
28 # YOLOv3
29 # YOLOv3
30 # YOLOv3
31 # YOLOv3
32 # YOLOv3
33 # YOLOv3
34 # YOLOv3
35 # YOLOv3
36 # YOLOv3
37 # YOLOv3
38 # YOLOv3
39 # YOLOv3
40 # YOLOv3
41 # YOLOv3
42 # YOLOv3
43 # YOLOv3
44 # YOLOv3
45 # YOLOv3
46 # YOLOv3
47 # YOLOv3
48 # YOLOv3
49 # YOLOv3
50 # YOLOv3
51 # YOLOv3
52 # YOLOv3
53 # YOLOv3
54 # YOLOv3
55 # YOLOv3
56 # YOLOv3
57 # YOLOv3
58 # YOLOv3
59 # YOLOv3
60 # YOLOv3
61 # YOLOv3
62 # YOLOv3
63 # YOLOv3
64 # YOLOv3
65 # YOLOv3
66 # YOLOv3
67 # YOLOv3
68 # YOLOv3
69 # YOLOv3
70 # YOLOv3
71 # YOLOv3
72 # YOLOv3
73 # YOLOv3
74 # YOLOv3
75 # YOLOv3
76 # YOLOv3
77 # YOLOv3
78 # YOLOv3
79 # YOLOv3
80 # YOLOv3
81 # YOLOv3
82 # YOLOv3
83 # YOLOv3
84 # YOLOv3
85 # YOLOv3
86 # YOLOv3
87 # YOLOv3
88 # YOLOv3
89 # YOLOv3
90 # YOLOv3
91 # YOLOv3
92 # YOLOv3
93 # YOLOv3
94 # YOLOv3
95 # YOLOv3
96 # YOLOv3
97 # YOLOv3
98 # YOLOv3
99 # YOLOv3
100 # YOLOv3
```

hedef : 0.54

t:34ms

Farklı Hiperparametrelerle Tekrardan Eğitim (2)

Farklı hiper parametrelerle model tekrardan eğitilmiştir. Tablo incelendiğinde özellikle 15.epochtan sonra loss değerlerinde düşüş olmamıştır. Modelin burada belirtilen epochtan sonra veri setini açıklayamadığı ifade edilebilir. Fakat yinede en son epochta ki model NCC yazılımıyla convert edilip gerçek zamanlı testi yapılmıştır. Testte görülen sonuç beklenen sonuçtur. Model “hedef” nesnesini tespit etmekte **başarısız** olmuştur.

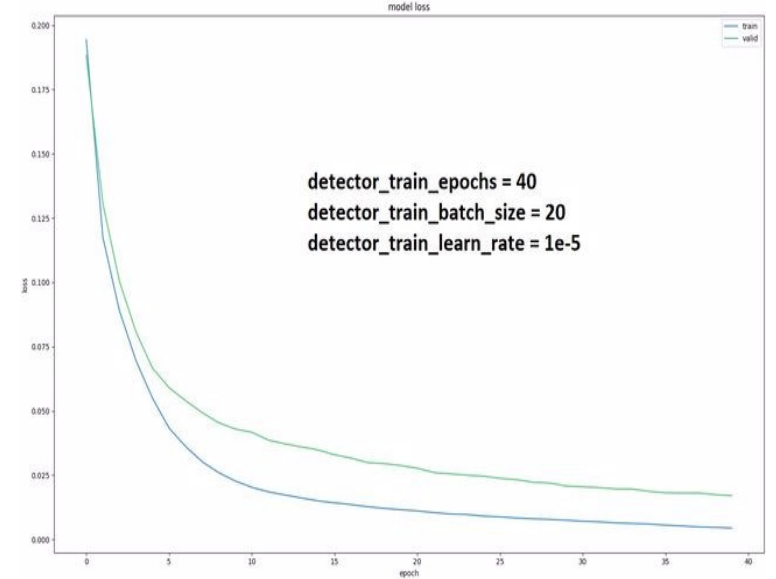


Bu eğitilen modelin 15.epoch olarak tekrardan eğitilip bir kez dahagerçek zamanlı testi yapılacaktır. Beklenen sonuç daha iyi olması yönündedir.

Farklı Hiperparametrelerle Tekrardan Eđitim (3)

Eđitim oranı dűşürűlműş batch size ve epoch ise arttırılmıřtır. Bu eđitim - kayıp grafiđi incelendiđindeyse loss deđerlerinin yűksek olduđu sűylenebilir. Fakat batch size arttırılarak birkaç eđitimden sonra dűřűk loss deđerleri gelebileceđi dűřűnűlmektedir.

Yinede modelin gerček zamanlı testi bu haliyle yapılmıřtır. Model “hedef” nesnesini tespit etmekte **bařarısız** olmuřtur.



2.Bölüm

2.Bölümün İçeriği ve akış şeması aşağıda mevcuttur.

Nesne Tespitinde Yaşanan Problemin Giderilmesi

- HiperParametre Seçimi
- OV2640 Kamera Kalitesi
- Fotoğraf Küçültürken(HD to 240x240) Lanczos3 Algoritmasında Faktör Değer Seçimi (50)
- Veri Setinin OV2640(kart kamerası) Üzerinde Oluşturulması Yerine fotoğraf küçültme kullanılması

Nesne Takibi İçin “Renk Takibi” ve Derin Öğrenmenin Birlikte Kullanılması

- Nesnenin Takibi ve Donanımın Yeterliliği
- Green ve Red kanalı için threshold ile blob araması gerçekleştirme

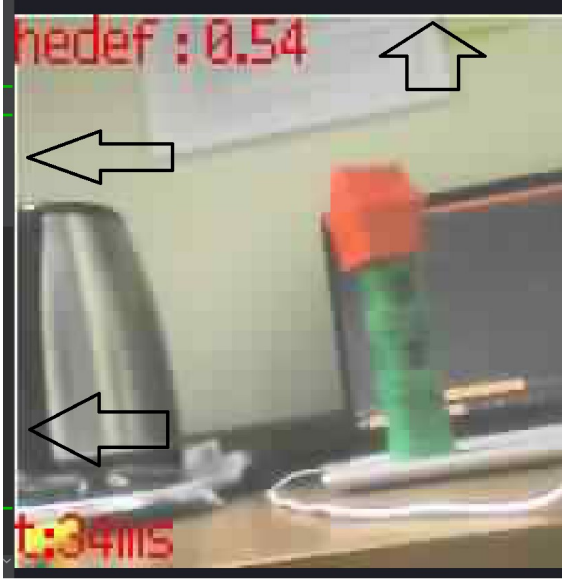
Arduino ve Maixduinonun Haberleşmesi

- FPIO VE GPIO kavramlarına giriş, devreler arasında -ilkel seviyede- giriş ve çıkış mantığına dayalı haberleşme

Nesne Tespitinde Geliştirilen Son Model

Farklı Hiperparametrelerle Tekrardan Eğitim Aşamasından Sonra Yaşanan Problemin Analizi

Vize Sunumunun Sonunda Yaşanan Problemin Tanımı: Nesneyi sınıflandıran algoritmanın başarılı çalıştığı fakat konumuyla ilgili problem yaşandığı söylenebilir.



Yanda ki görselde görüldüğü gibi nesneyi işaretleyen kutu bütün bir pencereyi kaplamış durumdadır. Bu problemin sebebi üzerine düşünülmüştür:

- 1) HiperParametre Seçimi
- 2) OV2640 Kamera Kalitesi
- 3) Veri Setinin OV2640(kart kamerası) Üzerinde Oluşturulması Yerine fotoğraf küçültme kullanılması
- 4) Fotoğraf Küçültürken(HD to 240x240) Lanczos3 Algoritmasında Faktör Değer Seçimi (50)
- 5) Veri Setinin Az Sayıda Olması(800)

Problem üzerine analizler ve çözümler sırayla sonraki sayfalarda anlatılmıştır. Problem çözülmüş ve başarılı bir nesne tespiti gerçekleştirilmiştir. **1.dönem hedefleri tamamen gerçekleştirilmiştir.**

1)Nesne Tespitinde Hiper Parametre Ayarı

Hiper Parametreler veriden öğrenme süreci için çok kritik öneme sahiptir. Doğru ince ayar yapmayı gerektirir. Aşağıda benim izlediğim yaklaşım mevcuttur. Fakat bir veri setini doğru şekilde ifade eden bir çok hiper parametre grubu mevcuttur.

Epoch: Verisetinin miktarına göre ayarlanması gerekir. Veriseti çok değilse epoch değeri çok yüksek olmaması gerekir. Yoksa overfitting durumuyla karşılaşmak olasıdır.

Batch – Size: Aynı anda kaç veriye bakılması gerektiğini ifade eder. Düşük veri setinde yüksek değerler girilebilir.

Learn – Rate: Öğrenme oranı anlamına gelir. Gradient Descent optimizasyon algoritmasında düşük loss değeri elde etmemizi ayarlayan hiperparametre olarak söylenebilir.

İzlenen yaklaşım:

- 1) Learn rate olabildiğince büyük bir değerle başlanır. Eğitim sonuçlarına göre adım adım küçült.
- 2) Batch size 16 ile başla adım adım arttır.
- 3) Epoch batch size ile ilişkili olacak şekilde değiştir.

2)OV2640 Kamera Kalitesi

- Eğitim veri setiyle kameranin ürettiđi görüntüler arasında bariz bir fark vardır. Bu nesnenin konumuyla ilgili bir sıkıntı oluşturabilecektir. Kameranin bu problemini çözmek için geniş açılı bir lens alınmıştır.
- Kameranin yazılımsal olarak sensor.QVGA: 320x240'dan sensor.VGA: 640x480'ya yükseltilmiştir.
- Aşağıda ki görseller incelendiğinde deđişim görülebilecektir.(Sağdaki VGA soldaki QVGA görüntüdür.)

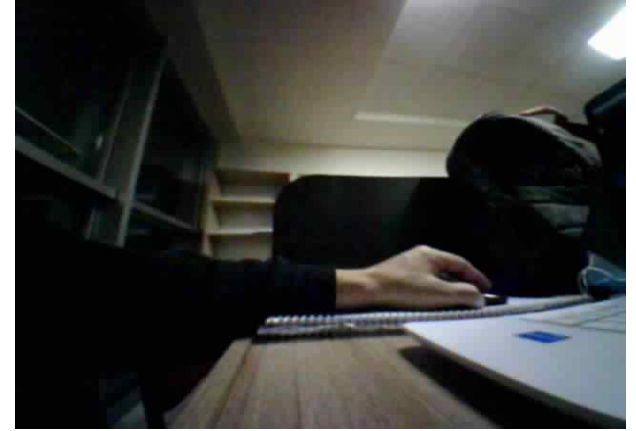
160 derece
75mm OV2640



QVGA: 320x240



VGA: 640x480



3)VeriSetinin OV2640(kart kamerası) üzerinde oluşturulması yerine fotoğraf küçültme kullanılması

Veri seti oluşturulurken HD kalitede çekim yapabilen telefon tercih edilmiştir. Fakat test gerçekleştirilen fotoğrafların kalitesine yakın bir kalitede verisetinin olmamasının problem olabileceği düşünülmüştür. OV2640 ile de görüntüler elde edilmiş veri setine eklenilmiştir.

4)Fotoğraf Küçültürken(HD to 240x240) Lanczos3 Algoritmasında Faktör Değer Seçimi (50)

Lanczos3 algoritmasında 0-100 arasında bir değer seçilerek görüntüler küçültülüyor. Yani kısaca bir biti ifade eden değeri belirleyen değerın seçimi konusunda sıkıntılar yaşanmıştır. Bu seçimin kararsızlıkları yüzünden model tespit konusunda başarısız olmuştur. Çünkü kalite arttırıldığıın verinin miktarının da artması gerekiydi.

5)Veri Setinin Az Sayıda Olması: Sentetik Görüntü Arttırma

Aşağıda ki yöntemler kullanılarak görüntüler çoğaltılmıştır. Bunun temel sebebi olarak OV2640 ile görüntü oluştururken kameranın sabit hareket ettiremediğinden görüntü oluşturma konusunda sorunlar yaşanmıştır. Çözüm olarak veri arttırma yoluna gidilmiştir.

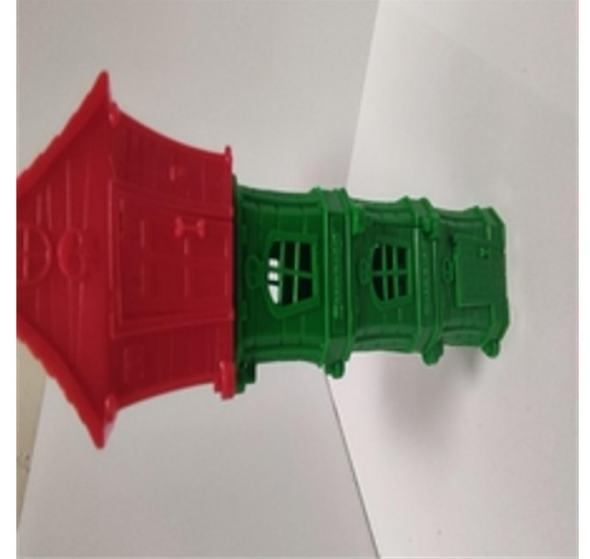
Parlaklık Arttırma



Bulanıklık ve Keskinlik



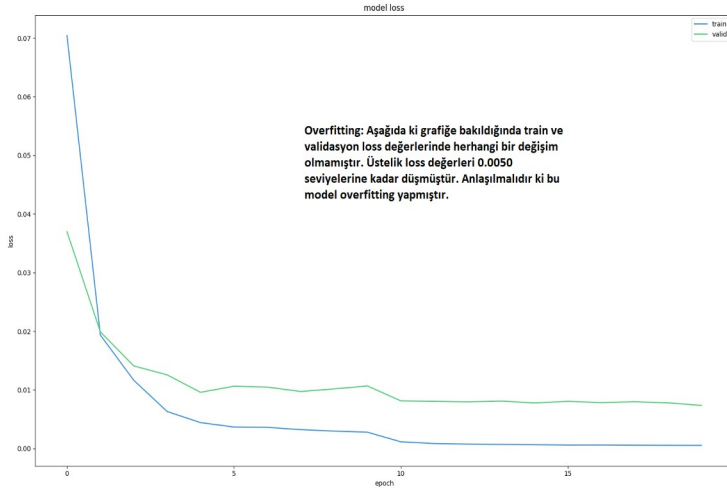
Rotasyon



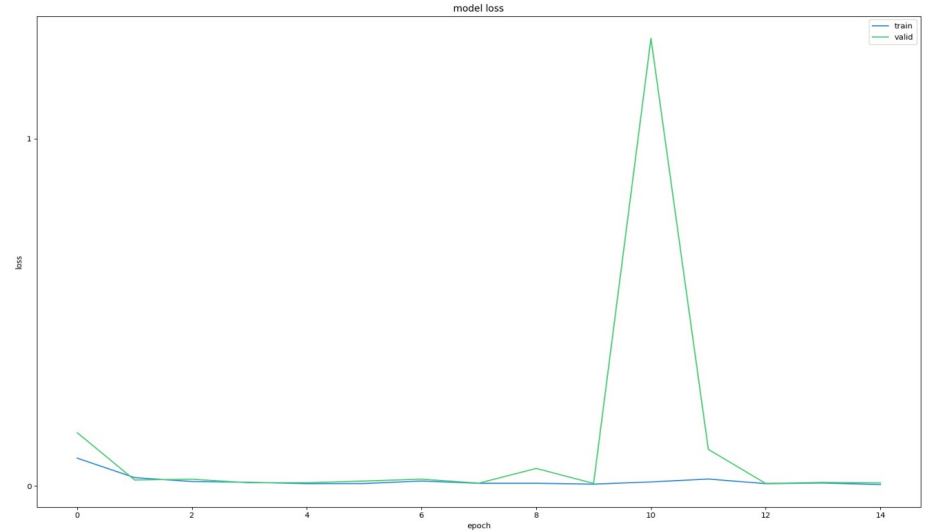
Çıkarım

Problemler ve çözümler üzerine düşünölmüş bunlara çeşitli ve çözümler geliştirilmeye çalışılmıştır. Burada bahsedilen yöntemler ve çözümler üzerine defalarca kez model eğitimi gerçekleştirilmiş bazıları başarılı bazıları ise başarısız olunmuştur. Bu raporda hepsine yer verilmesi mümkün olmaz. Fakat aşağıda değişik çözümlerle yapılmış eğitimlerle ilgili loss değerleri aşağıda verilmiştir.

Losslar yataya geçmiş ve çok düşük seviyelere inmiş yani **overfit** etmiştir.



Grafiğin lossunda yüksek bir artış olup tekrardan inmiş. Buna **underfit** ettiğini söyleyebiliriz. Bu durum için batch size değerini arttırmak çözüm olabilir



Nesnenin Takibi ve Donanımın Yeterliliği

MaixDuino geliştirme kartında üreticinin belirttiği datasheet incelendiğinde görülecektir.”QVGA@60FPS/VGA@30FPS image identification” olarak belirlenmiştir. Fakat bu değer gerçeği yansıtmamaktadır.

- 160 derece lens ile VGA da 3-7fps arasında değerler alınmıştır.
- Lensi çıkardığımda ve test ettiğimde ise 9fps değerine kadar çıktığı görülmüştür.

Not: Bu değerler sadece görüntünün sensor okunmasıyla çıkmıştır. Başka herhangi bir işlem yoktur.

Projede kamera arac üzerine sabitlenerek aracın hedef nesnenin yanına gitmesi hedeflenmekteydi. Burada da nesnenin takibinin yapılması çok önemli bir konu. Yani her ne kadar nesneyi tespit etme kısmı önemli olsa da takip kısmı da önemlidir. Takip kısmında derin öğrenmeyle yapılacağı düşünülürse fps değerinde maliyetten dolayı çok ciddi düşüşler olacaktır.

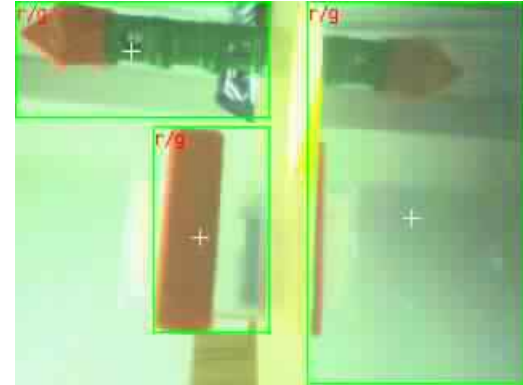
Sayısal Görüntü İşleme Kullanarak Nesne Takibinin Yapılması



Hedef görseli incelendiğinde görülecektir ki RGB kanallı bir görüntüde (değer,değer,0) ile hedef nesneyi bulmak yeterli olacaktır. Bu durumun kullanılması düşük FPS değerine çözüm olabileceği düşünülmüştür. Buraya yapılmaya çalışan blob yöntemiyle renk aramaktır. Görünütün içinde damla şeklinde verilen thresholdlara ait parçacıklar aramaya dayanır. Maliyeti derin öğrenmeye göre çok daha az ve hızlıdır.

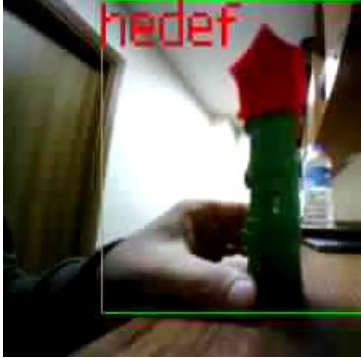
thresholds = [(30, 100, 15, 127, 15, 127),
(30, 100, -64, -8, -32, 32)]

Not: Geniş açılı kamerada bu yöntem ne yazık ki istenildiği gibi sonuç vermemektedir. Araştırmaya ve çözüm bulunmaya devam edilecektir.



Modelin Nesne Tespiti

Belirlenen problem ve analizler üzerine nesne tespiti başarılı yapabildiği söylenen bir model geliştirilmiştir. Aşağıda bu modelin test görüntüsü görülmektedir. Bu başarımda kameranın geniş açılı olması, QVGA kalite görüntü, hiperparametre ayarı olduğu söylenebilir. Fakat geniş açılı kamera renkli blob yöntemiyle birlikte çalışmadığı için iki hafta daha model daha başarılı hale getirilmeye çalışılacaktır.



Model geliştirilirken aşağıda ki gözden kaçırılanlara dikkat edilip,tekrardan eğitilecektir.

- 1) Lanczos algoritmasında faktör değeri 100'e çıkarılacaktır. Bir pixel ifade eden bit değeri artılacak. Böylelikle fotoğraf küçültülürken kaybolan kaliteyi burdan alınması sağlanacaktır.
- 2) Video kayıt ile oluşturulan veri setinde 1sn'de 24 frame çıkıyorsa her 6 framede ki bir görüntü eğitim veri setine dahil edilecektir. Amaç verilerde ki değişim oranını arttırıp, öğrenmeyi sağlamaktır.
- 3) Bütün bunların yanında hiperparametre ayarı olarak bayes kullanılacaktır. Modelin başarımları 0-4 arasında derecelendirilip, bayes yöntemiyle en doğru hiperparametre ayarı belirlenmeye çalışılacaktır.
- 4) Verisetinde birden farklı kaynaktan elde edilen verilerin modelin öznelik çıkarımını zorladığı düşünülmektedir. Yani kameradan aynı zamanda telefondan elde edilip(küçültüp kullanılan) aynı eğitim veri seti içerisinde eğitime verilmeyecektir.

Bu bahsedilen yöntemler uygulanarak daha başarılı bir model için çalışmaya devam edilecektir. Fakat bu aşamada ki modelde başarılı kabul edilebilir. O yüzden takvimde ki diğer çalışmalara devam edilmiştir.

Maixduino İle Arduinonun Haberleşmesine Giriş

FPIO, kavramıyla tanışılması üzerine maixduinoda çok basit bir çalışma yapılmıştır. FPIO kısaca programlanabilir giriş çıkış kapılarını ifade eder. Fpio_manager ile bu işlemi yaparız.

Bu çalışma sırasıyla şunu hedeflemiştir:

- 1) FPIO değerlerini GPIOlar ile eşleştir.
- 2) Maixduinoda bir butona basıldığında başka bir pini lojik "1" seviyesine çek. (Karşılığı 3.3V)
- 3) Maixduinodan çıkan pin doğrudan arduinoya bağlı olan bu pin lojik 1 seviyesine çekildiğinde arduinoda bu pinden lojik 1 değerini okur.
- 4) Arduino gelen lojik 1 değeri üzerine başka bir pinden ledi yakar.

```
from fpioa_manager import fm
from Maix import GPIO
from Maix import FPIOA
...

io_led_red = 13
fm.register(io_led_red, fm.fpioa.GPIO1)

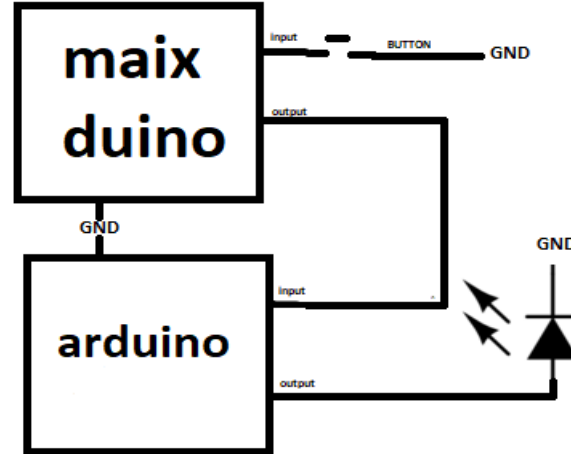
led_r=GPIO(GPIO.GPIO1, GPIO.OUT)
led_r.value(1)

fpioa = FPIOA()
fpioa.set_function(13, fm.fpioa.GPIOHS0)
...

fm.register(13, fm.fpioa.GPIO5) #GPIO 9
oku=GPIO(GPIO.GPIO5, GPIO.IN)

fm.register(12, fm.fpioa.GPIO1) #GPIO 10
led_r=GPIO(GPIO.GPIO1, GPIO.OUT)
led_r = led_r.value(0)

sayac = 0
while(1):
    if(oku.value()==0):
        led_r.value(1)
        print("debug1")
        sayac = sayac + 1
        print(sayac)
    else:
        print("debug2")
        led_r.value(0)
```



Nesne Tespitinde Başarımı Arttırma Ve Modelin Son Halini Elde Etme

Transfer learning(Öğrenme Aktarımı) yapılarak eğitilen bir önceki model tekrardan eğitilmiştir. Aynı veriseti kullanılmış fakat görseller küçültülürken lanczos algoritmasında(görüntü küçültme algoritması) faktör değeri değiştirilmiştir. Yanda ki görsel ilk başta gerçekleştirilen eğitim verisetinden bir örnek olarak verilmiştir.

İlk başta eğitilen model, sınırlayıcı kutuyu çizmekte ve uzakta ki hedef görselini tespit etmekte zorlanmıştır. Veri setini çeşitlendirmek adına çalışmalar yapmayı düşünürken transfer learning uygulanmış ve probleme çözüm getirilmiştir.

Bu verilerin farkı bir biti ifade eden değerin daha yüksek olmasıdır. Yani bir başka deyişle görsellerin kaliteleri arasında ki farkları onları birbirinden farklı yapıyor.

İlk Eğitimde ki Veri



224x224pixel
5.494 bayt

Transfer Learning Aşamasında ki Veri



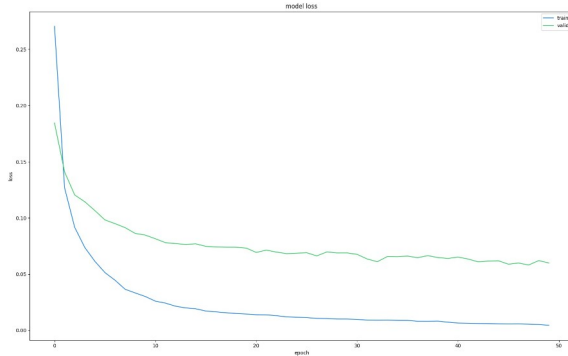
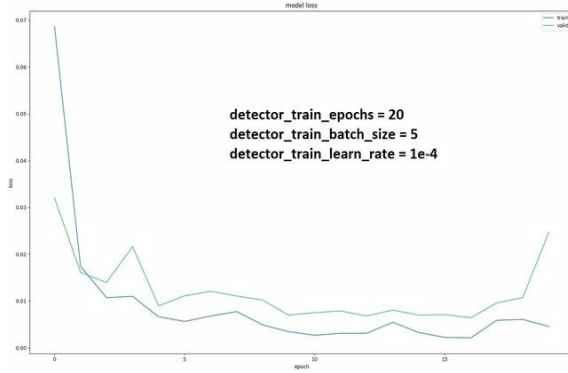
224x224pixel
29.320 bayt

Modelin Gerçek Zamanlı Testi

Birinci sayfada bahsedilen modele early stopping(erken durdurma) yapılarak verimli sayılabilecek bir model elde edilmiştir. İlk modeli eğitirken aşağıda ki hiperparametreler kullanılmıştır:

Epoch ---> 20
Batch Size --> 5
Learning Rate --> 1e-4

Bu eğitilen ilk modelin gerçek zamanlı testi yapılarak başarılı olduğu görülmüştür. Ama sınırlayıcı kutunun(bounding box) nesneyi tam sınırlayacak şekilde kutuyu çizemediği görülmüştür. Aynı zamanda uzakta ki hedef nesnesini de tespit etmediği görülmüştür. Transfer learning yapılarak tekrardan eğitilmiştir.



Transfer learning aşaması için 7-8 tane model eğitilmiştir. Fakat yanda ki model en düşük loss değerlerini verdiği için tercih edilmiştir.

Epoch ---> 50
Batch Size --> 5(yanlış girilmiş olabilir)
Learning Rate --> 1e-5
bbox accuracy(IOUS): 75.25%

TF yapılarak eğitilen modelin gerçek zamanlı testi yapılmış ve beklenen başarımlar sağlanmıştır.

Modelin Gerçek Zamanlı Testinden Bazı Görseller

