**Resim ya da Videodan Uçak Nesnesi Tespiti**

**Yapılan Adımlar**

1. 1000 adet farklı farklı uçak görseli içeren veri seti indirilmesi.
2. Bu resimlerin içerdikleri uçak konumlarına göre etiketlenmesi.
3. YOLOv3 ile eğitileceği için darknet klasöründe gerekli ayarlamaların yapılması.
4. Ayarlanmaların yapıldığı darknet klasörünün Google’ ın sunduğu ücretsiz GPU sağlayan Google Colab aracılığıyla eğitilmesi.
5. Eğitim sonucunda elde edilen ağırlığa göre video ya da görselden nesne tespitinin yapılması.

**Kullanılan Teknolojiler**

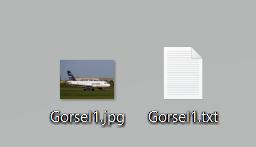
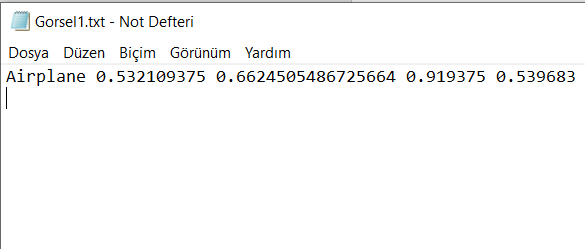
* YOLOv3
* Google Colab
* Python
* OIDv4\_ToolKit
* OpenCV

1. **Veri Setinin Hazırlanması**

Veri seti hazırlamak oldukça zahmetli bir iştir. Çünkü her bir görselde nesnenin işaretlenmesi ve bu işaretli yerin koordinatlarının dosya ile aynı isimde bir txt dosyasında saklanmalıdır.

Örnek verecek olursak :

Gorsel1.jpg dosyası var ise Gorsel1.txt adında bir dosyamız daha olmalı ve bu dosyada ***<object-class> <x\_center> <y\_center> <width> <height>*** şeklinde bir ifade yer almalıdır. (Görsel – 1 ve Görsel – 2 bkz.)

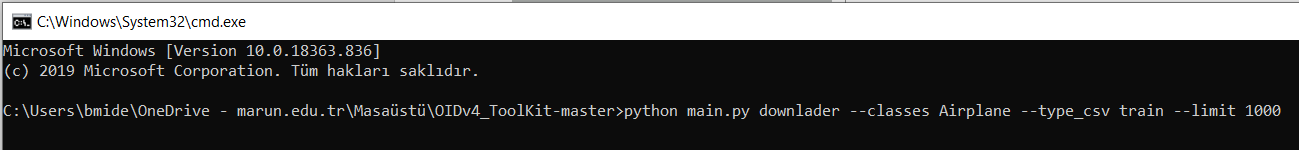
İnternetten tek tek görsel bulup bunlar için txt dosyası hazırlamak zahmetli bir iş fakat OIDv4\_ToolKit adlı program sayesinde 9 milyondan fazla resim barındıran **Google Open Images Dataset** kaynağından etikenlenmiş yani txt dosyaları ile birlikte binlerce görseli kolayca indirebiliyoruz.

OIDv4\_ToolKit programına <https://github.com/theAIGuysCode/OIDv4_ToolKit> adresinden ulaşabilirsiniz. Programın kurulumu linkteki README.md dosyasında anlatılmıştır.

Ben indirdiğim OIDv4\_ToolKit dosyası içerisinde iken cmd ekranında “pip install -r requirements.txt” kodunu yazarak gerekli kurulumları yapmış oldum.

Daha sonra istediğimiz sınıfa ait görselleri indimek için sınıfımızın adını <https://storage.googleapis.com/openimages/web/index.html> adresinden explore dedikten sonra arama yaparak belirleyebiliriz.

Sınıfımızı belirledikten sonra yine OIDv4\_ToolKit dosyası içerisinde cmd ekranında



Görseldeki kodu yazarak Airplane adlı sınıfa ait dosyaları indirdim.

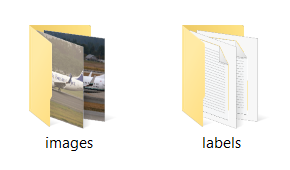
Burada

**- -classes** ifadesine inidireceğimiz sınıfları yazıyoruz.

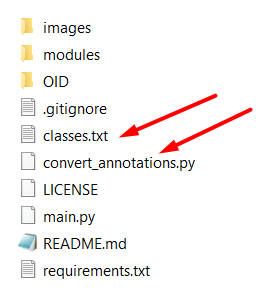
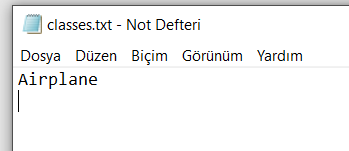
**- -type\_csv** ifadesine eğitim modeli olduğu için train yazıyoruz.

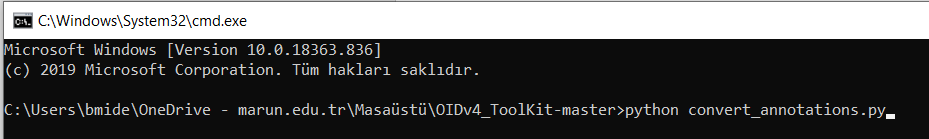
**- -limit ifadesi** indirmek istediğimiz resim sayısıdır. Birden fazla sınıf girilir ise her bir sınıftan o limit kadar resim indirilir.

İndirilen dosyalar **\OIDv4\_ToolKit-master\OID\Dataset\train\Airplane** içerisinde **labels** ve **images** adında iki klasör altında saklanır.

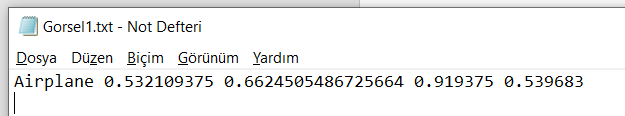


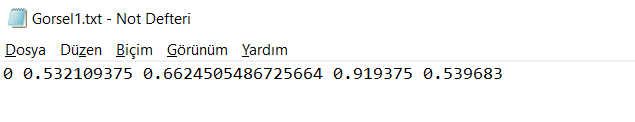
Burada şöyle bir sıkıntı var labels klasöründe dosyalarda class ismi 0-1-2 gibi değil de Airplane Balloon şeklinde yazılıdır. Ancak YOLOv3 bunu kabul etmez bu yüzden programımızın klasöründe bulunan **classes.txt** dosyasını açıyoruz ve indirdiğimiz nesnenin adını yazıyoruz ardından komut satırında aşağıda bulunan kodu çalıştırıyoruz.





Bu kod sonrasında aşağıdaki dönüşüm yapılır ve resimlerin bulunduğu klasöre txt dosyaları geçer.



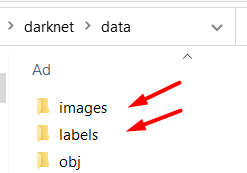


1. **Darknet Klasörü İçerisinde Ayarlamaların Yapılması**

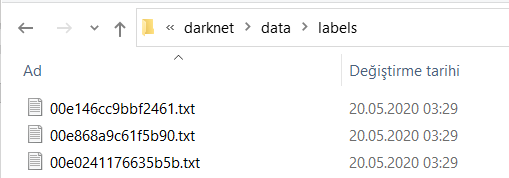
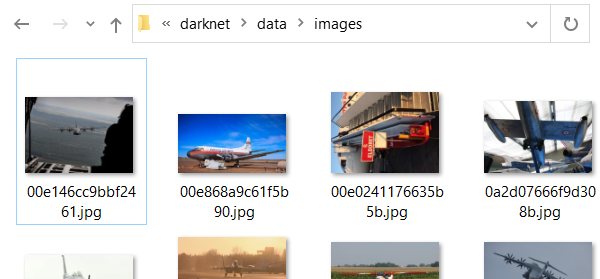
Veri setini indirdikten sonra <https://github.com/pjreddie/darknet> adresinden **darknet** dosyasını indiriyoruz.

Daha sonra darknet klasörü içerisinde **data** klasörüne indirdiğimiz resimleri içeren **images** klasörünü, yine data içerisinde **labels** klasörüne ise resimlerin **txt** dosyalarını atıyoruz.

Darknet/data içerisi aşağıdaki gibi olmalıdır.



Ve klasör içerikleri aşağıdaki gibidir.

Daha sonra **darknet** klasöründe bazı değişiklikler yapmamız gerekecek .

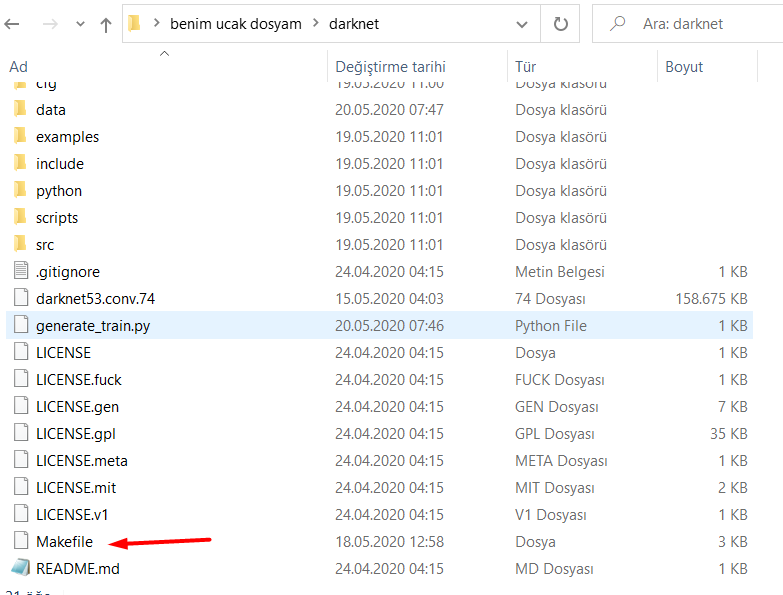
Görselde görüldüğü gibi darknet klasörü içerisinde **Makefile** dosyasını not defteri ile açıyoruz ve

**GPU =**1 ,

**CUDNN=1**

**OPENCV=1**

Değişikliklerini yapıp kaydediyoruz.





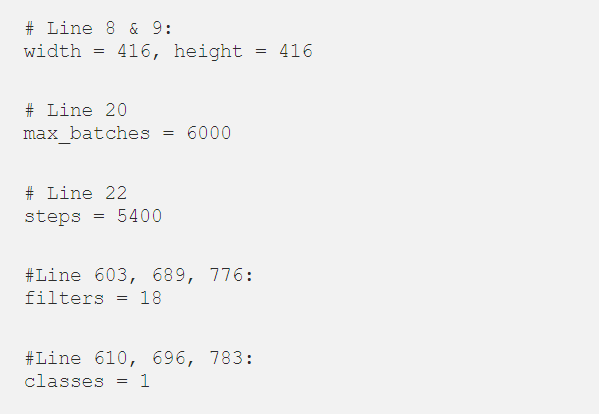
Sonraki adımda <https://pjreddie.com/media/files/darknet53.conv.74> linkinden önceden eğitilmiş bir darknet modelini **darknet** klasörümüzün içine indiriyoruz.

Sonraki adım **configuration** dosyasının değiştirilmesi. **darknet/cfg** içerisinde bulunan **yolov3.cfg** dosyasını aynı yere ismini **yolov3\_custom\_train.cfg** olarak kopyalıyoruz.

darknet/cfg içerisinde aşağıdaki gibi bulunmalıdır.



Oluşturduğumuz **yolov3\_custom\_train.cfg** dosyadını not defteri ile açıyoruz ve aşağıda yazılı değişiklikleri yapıyoruz. *(Line olan parametreler değerlerin hangi satırda olduğunu ifade ediyor)*



Bu değişiklikleri de yaptıktan sonra son yapmamız gereken değişiklik olan **darket/examples** klasörü içerisinde bulunan **detector.c** dosyasının 183. Satırında

if(i%10000==0 || (i < 1000 && i%100 == 0)){

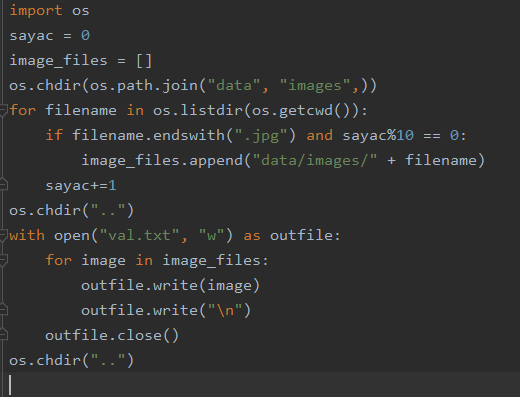


if(i%1000==0 || (i < 1000 && i%100 == 0)) {

değişikliğini yapmak. Bu değişiklik ile modelimizin **her 100 epoch** da bir kaydedilmesini **1000 den büyük olduğunda da her 1000 epoch da** kaydedilmesini sağlamak. Çünkü Colab tek seferde **maksimum 12 saat** eğitime izin verir.

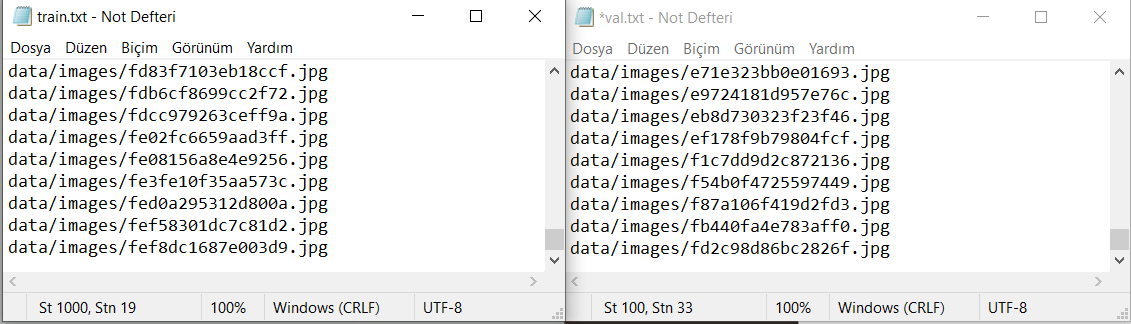
Sonraki adımımızda **images** klasörümüzdeki resimlerin pathlerini satır satır içeren **train.txt** ve **val.txt** dosyalarını oluşturmamız gerekiyor. Burada val.txt dosyası tüm verilerimizin %30 kadarını içermesi önerilir. Bunun için yazılmış bir python programı kullandım. Koda <https://github.com/theAIGuysCode/YoloGenerateTrainingFile/blob/master/generate_train.py> adresinden erişebilirsiniz.

Buradaki kod ile tüm pathleri içeren **train.txt** dosyası oluşturulur. **val.txt** için görseldeki kodu kullandım kodda her 10 da bir alarak val.txt dosyasına eklemektedir.

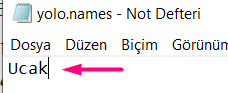


*val.txt dosyası için kullandığım kod*

Oluşan dosyalara bakacak olursak data klasöründe bulunan train.txt **1000** satır(1000 tane resim olduğu için) val.txt **100** satırdan oluşuyor. Görselde içerikleri görebilirisiniz.

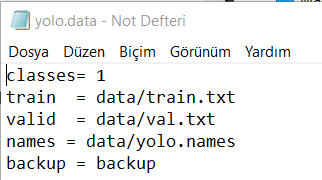
 *train.txt ve val.txt dosya içerikleri*

Sonraki adımda **darknet/data** klasörüne **yolo.names** adında bir dosya oluşturuyoruz ve içerisine nesnemizin ismini yazıyoruz yani : **Ucak**.



*yolo.names dosya içeriği*

Daha sonra yine **darknet/data** klasörüne **yolo.data** diye bir dosya oluşturuyoruz ve içeriğini aşağıdaki gibi yapıyoruz.



*yolo.data dosya içeriği*

**Not:** classes değerinin 1 olmasının sebebi bir adet nesnemizin olmasıdır (Uçak).

Tüm bunlardan sonra

* **darknet/data** içerisinde 1000 adet resim içeren **images** **klasörü**
* **daknet/data** içerisinde 1000 adet txt dosyası (+ önceden içinde olanlar) içeren **labels klasörü**
* **darknet/data** içerisinde **train.txt, val.txt, yolo.data** ve **yolo.names** dosyaları
* **darknet/cfg** içerisinde **yolov3\_custom\_train.cfg** dosyasını

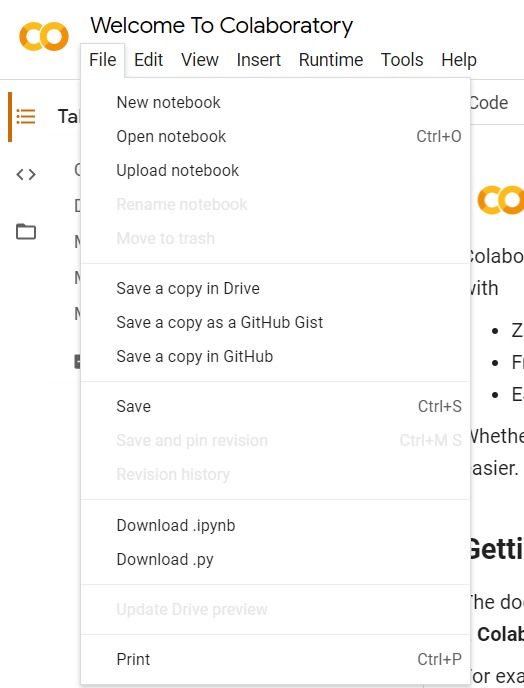
içeren darknet klasörümüzü darknet.zip olacak şekilde sıkıştırıyoruz.*(zip olması önemli)*

1. **Hazırlanan Veri Setinin Google Colab’ da Eğitilmesi**

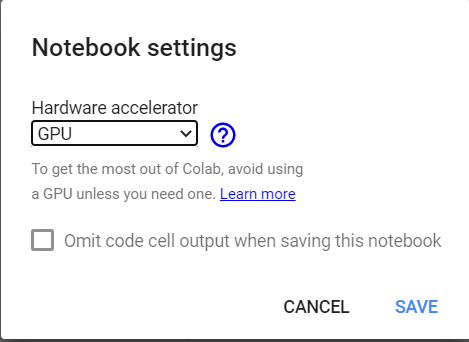
Sıkıştırılan bu darknet.zip dosyasını Google Drive’ a atıyoruz.

Şimdi sıra geldi Colab ortamımızı hazırlamaya.

<https://colab.research.google.com/> adresinde Drive’ a yüklediğimiz dosya ile aynı olan Google hesabımızı açıyoruz. Ve **File -> New Notebook** diyoruz.



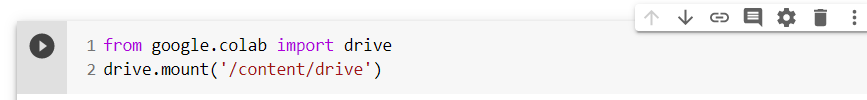
Daha sonra GPU aktif etmek için **Runtime -> Change runtime type** diyerek resimdeki gibi ayarlıyoruz ve **save** diyoruz.



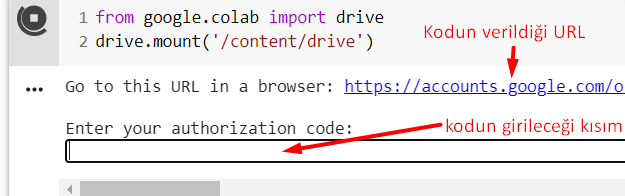
Defterimizin ayarlarını yaptıktan sonra aşağıda yer alan kodları **tek tek sırasına göre** çalıştıracağız.

Çalıştırmak için yazdıktan sonra solda bulunan **play** tuşunu ya da **shift + enter** tuş kombinasyonunu kullanabiliriz.

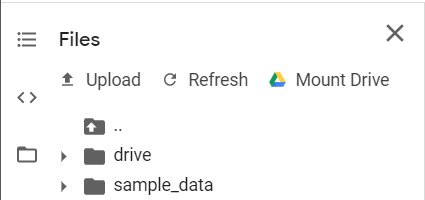
**İlk olarak google drive’ı colaba bağlıyoruz.**



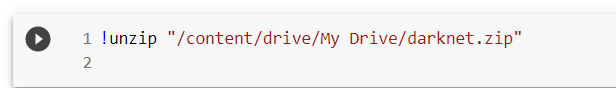
Play dedikten sonra bize onay için bir URL verir bu URL’e tıklayıp oradaki kodu Colab’da açılan alana girmemiz gerekir.



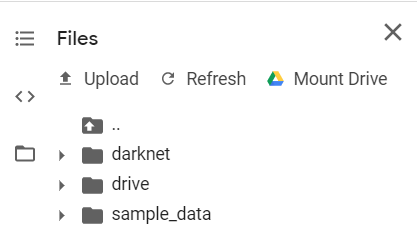
Bağlandıktan sonra sol tarafta dosya dizininde **drive** klasörünü görebilirsiniz.



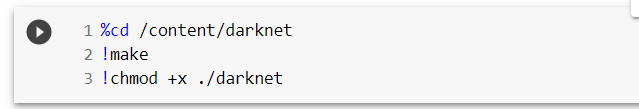
**İkinci olarak drive’da bulunan darknet.zip dosyasını Colab çalışma alanımıza çıkarıyoruz.**



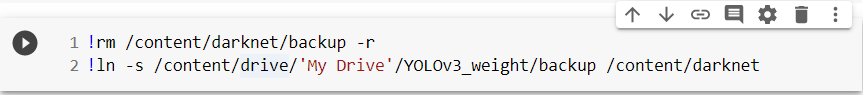
Bu işlemden sonra ise dosya dizininde **darknet** klasörü gelmiş olmalıdır.



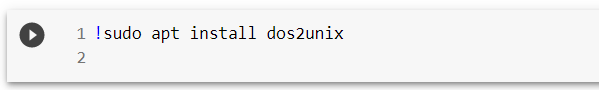
**Üçüncü olarak aşağıdaki kodu çalıştırarak darknet içeriğini derliyoruz.**



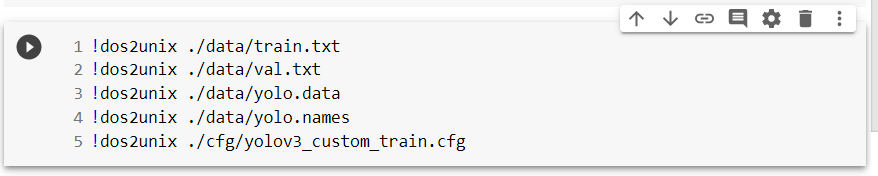
**Dördüncü işlemde driveımızda YOLOv3\_weight diye bir klasör açıyoruz ve bunun da içerisine backup adında bir klasör açıyoruz. Kaydedilen her ağırlığın drive’a da kaydedilmesi için aşağıdaki kodu yazıyoruz. Bu adım işimizi kolaylaştırmak amaçlıdır dilerseniz eğitim sonunda da elle eğitim dosyalarını indirebilirsiniz.**



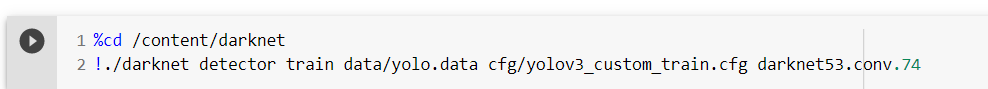
**Sonraki işlemimiz Train.txt, val.txt, yolo.data, yolo.names, yolov3\_custom\_train.cfg dosyalarını Unix sisteme dönüştürmek için dos2unix’ i Colab ortamımıza yüklüyoruz.**



**Yükledikten sonra dosyaları UNIX için çeviriyoruz.**



**Tüm işlemler sorunsuz olduğunda eğitimimizi başlatmak için aşağıdaki kodu yazıyoruz.**

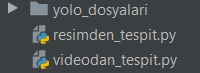


Artık eğitim işlemimiz başlamış bulunmakta ve 1000 epoch olana kadar her 100 epoch’da (1000 den sonra 1000, 2000 şeklinde) bir hem **darknet/backup** hem de **drive/YOLOv3\_weight/backup** klasörüne kaydedilecektir.

<https://colab.research.google.com/drive/13-9pAz9nxUYm-0LlNV9tVtS57g8mHAOb> adresinde işlemler ile ilgili çıktıları bulabilirsiniz.

1. **OpenCV ile Nesne Tespitinin Yapılması**

Eğitim setimizi eğittikten sonra sıra geldi nesne tespiti yapmaya. Nesne tespiti için hazır kod kullandım. Kod içerisinde video ya da görselden tespit yapmak için iki ayrı .py uzantılı dosya var.



Burada kodları inceleyerek daha iyi anlayabiliriz.

Projede **videos, images, output** ve **yolo\_dosyalari** olarak isimlendirilmiş 4 klasör bulunmakta.

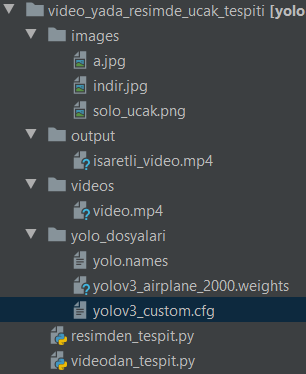
**images** klasörü içerisinde resimden tespit için kullanacaksak gerekli resim dosyasını içermelidir.

**videos** isimli klasörde ise tespit yapmak istediğimiz videoyu içermeli.

**output** klasöründe tespit yapmak istediğimiz videonun nesnesi işaretli hali kaydedilmekte.

**yolo\_dosyalari** klasöründe **yolo.names**, eğittiğimiz modelin .**weights** dosyası ve daha önce de kullandığımız **yolov3\_custom.cfg** dosyamız bulunmakta.

**videodan\_tespit.py** ve **resimden\_tespit.py** dosyaları ise proje klasöründe yer alıyor.



*Proje Klasörü İçeriği*

**Adım adım resimden nesne tespiti :**

İlk önce images klasörüme aşağıdaki görseli bayraktar-tb2.jpg olarak kaydediyorum.



*Bayraktar-tb2.jpg*

Daha sonra proje klasörü içerisindeyken terminalden aşağıdaki komutu çalıştırıyoruz.



*Resimden nesne tespiti için gerekli kod.*

Komuttaki :

**--image** parametresi ile görselin bulunduğu konumu veriyoruz.

**--yolo** parametresi ile yolo dosyalarımızın bulunduğu konumu veriyoruz.

Ve resmimiz işaretlenmiş olarak ekrana geliyor. Uçak yazısının yanındaki 0 ile 1 arasındaki değer doğruluğu göstermektedir.



*İşaretlenmiş olan görsel*

**Adım adım videodan nesne tespiti :**

İlk önce videos klasörüme video.mp4 adında bir video kaydediyorum. Video içerisinde birkaç görseli aşağıya ekledim.





Daha sonra proje klasörü içerisindeyken terminalden aşağıdaki komutu çalıştırıyoruz.



*Videodan nesne tespiti için gerekli kod.*

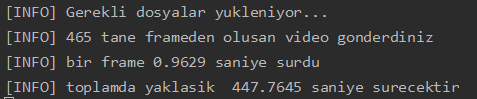
Komuttaki :

**--video** parametresi ile videonun bulunduğu konumu *(videos/video.mp4)* veriyoruz.

**--output** parametresi ile işaretli videomuzun kaydedileceği konumu *(output/)* ve kaydedileceği adı *(işaretli\_hal.avi)* veriyoruz. Burada **.avi** olması önemlidir.

**--yolo** parametresi ile yolo dosyalarımızın bulunduğu konumu veriyoruz.

Komuttan sonra bize videonun kaç frameden oluştuğunu ve bir frame in işlenmesinin kaç saniye sürdüğü gibi bilgiler verilmekte.



Ve işlem bittikten sonra videomuz işaretlenmiş olarak **output/** altında kaydediliyor.

İşlemin bittiğini Bitiriliyor yazısı ile anlayabiliriz.



İşaretlenmiş videodan birkaç görsel aşağıdaki gibidir.

