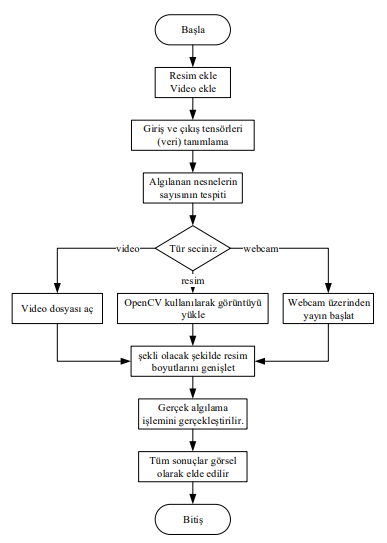
**Fruit-Classification: Work-In-Progress**

Fruits 360 veri kümesinde TensorFlow-Keras kullanarak Meyve Sınıflandırma / Tanımlama

Nesne tanıma işlemlerinde akış şeması aşağıdaki gibi oluşturulmaktadır:



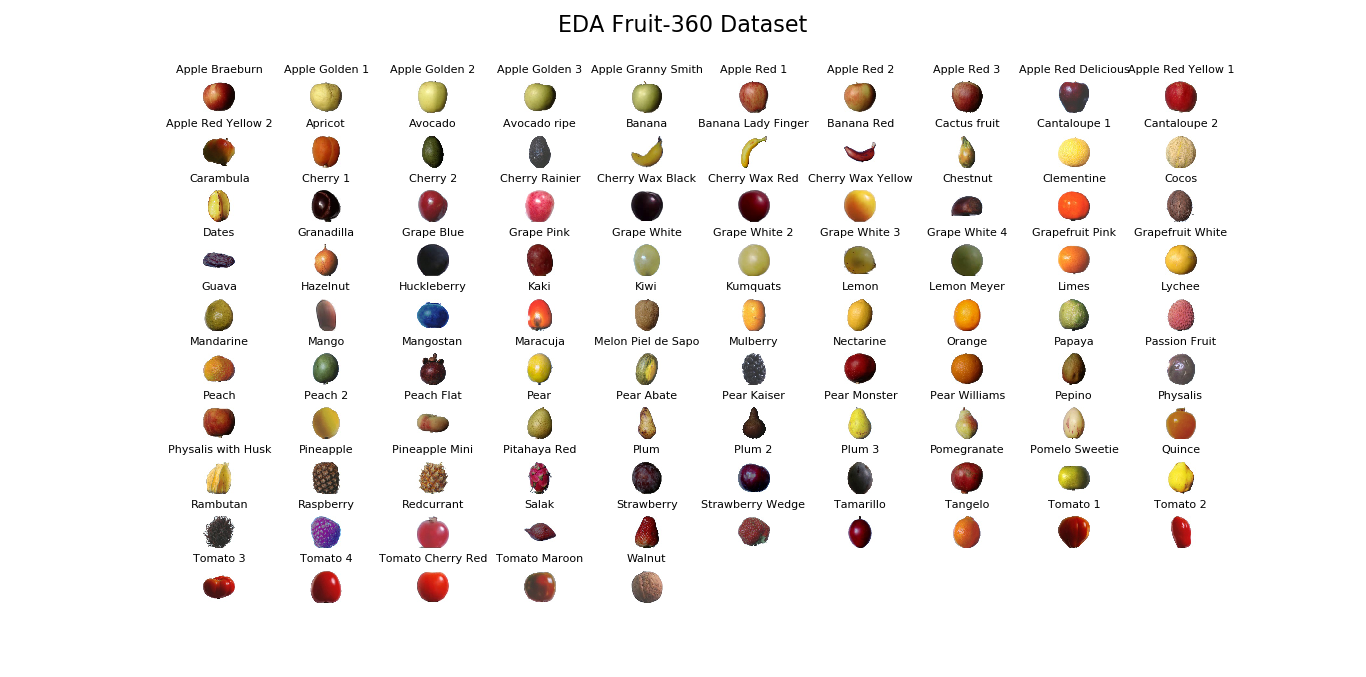
Yukarıdaki akış şeması nesne tanıma işlemlerinin genel akış şemasıdır. Ancak bu projede video dosyası veya webcam üzerindne tanıtma kullanılmamış olup yalnızca OpenCV kullanırak görüntü yüklenmiştir.

OpenCV: OpenCV (Open Source Computer Vision) açık kaynak kodlu görüntü işleme kütüphanesidir. 1999 yılında İntel tarafından geliştirilmeye başlanmış daha sonra Itseez, Willow, Nvidia, AMD,  Google gibi şirket ve toplulukların desteği ile gelişim süreci devam etmektedir. İlk sürüm olan OpenCV alfa 2000 yılında piyasaya çıkmıştır. İlk etapta C programlama dili ile geliştirilmeye başlanmış ve daha sonra birçok algoritması C++ dili ile geliştirilmiştir. Open source yani açık kaynak kodlu bir kütüphanedir ve BSD lisansı ile altında geliştirilmektedir. BSD lisansına sahip olması bu kütüphaneyi istediğiniz projede ücretsiz olarak kullanabileceğiniz anlamına gelmektedir.  OpenCV platform bağımsız bir kütüphanedir, bu sayede Windows, Linux, FreeBSD, Android, Mac OS ve iOS platformlarında çalışabilmektedir. C++, C, Python, Java, Matlab, EmguCV kütüphanesi aracılığıyla da Visual Basic.Net, C# ve Visual C++ dilleri ile topluluklar tarafından geliştirilen farklı wrapperlar aracılığıyla Perl ve Ruby programlama dilleri ile kolaylıkla OpenCV uygulamaları geliştirilebilir.

2016-05-27 tarihli güncelleme, OpenCV geliştirici Itseez firması Intel tarafından satın alındı. OpenCV geliştirmesine Intel çatısı altından devam edeceğini duyurdu.

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

**Veri Kümesi:**

[](https://github.com/MeAmarP/Fruit-Classification/blob/master/results/EDA_images_v22.png)

## Veri kümesi özellikleri

Toplam resim sayısı: 90483.

Eğitim seti boyutu: 67692 görüntü (görüntü başına bir meyve veya sebze).

Test seti boyutu: 22688 görüntü (görüntü başına bir meyve veya sebze).

Çok meyveli set boyutu: 103 görüntü (görüntü başına birden fazla meyve (veya meyve sınıfı))

Sınıf sayısı: 131 (meyve ve sebzeler).

Görüntü boyutu: 100x100 piksel.

Dosya adı formatı: görüntü *dizini* 100.jpg (ör. 32 *100.jpg) veya r* görüntü *dizini* 100.jpg (ör. R *32* 100.jpg) veya r2 *görüntü* dizini *100.jpg veya r3* görüntü *dizini* 100.jpg. "r", döndürülmüş meyve anlamına gelir. "r2", meyvenin 3. eksen etrafında döndürüldüğü anlamına gelir. "100", görüntü boyutundan (100x100 piksel) gelir.

Aynı meyvenin farklı çeşitleri (örneğin elma), farklı sınıflara ait olarak saklanır.

## Nasıl başardık

Düşük devirli bir motorun (3 rpm) şaftına meyve ve sebzeler dikildi ve 20 saniyelik kısa bir film kaydedildi.

Meyveleri filme almak için bir Logitech C920 kamera kullanıldı. Bu, mevcut en iyi web kameralarından biridir.

Meyvelerin arkasına arka plan olarak beyaz bir kağıt koyduk.

Bununla birlikte, aydınlatma koşullarındaki farklılıklar nedeniyle, arka plan tek tip değildi ve meyveyi arka plandan çıkaran özel bir algoritma yazdık. Bu algoritma flood dolgu tipindedir: Görüntünün her kenarından başlarız ve oradaki tüm pikselleri işaretleriz, ardından renkler arasındaki mesafenin öngörülen bir değerden daha az olduğu önceden işaretlenmiş piksellerin çevresinde bulunan tüm pikselleri işaretleriz. Daha fazla piksel işaretlenmeyene kadar önceki adımı tekrarlıyoruz.

İşaretlenen tüm pikseller arka plan olarak kabul edilir (daha sonra beyazla doldurulur) ve geri kalan pikseller nesneye ait olarak kabul edilir.

2 komşu piksel arasındaki mesafe için maksimum değer, algoritmanın bir parametresidir ve her film için (deneme yanılma yoluyla) ayarlanır.

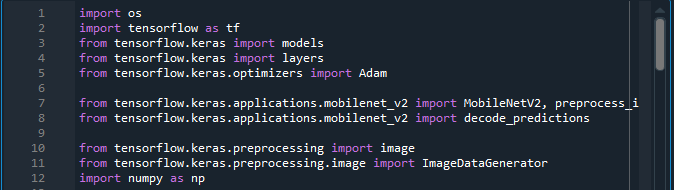
Test-multiple\_fruits klasöründeki resimler Nexus 5X telefonla çekildi.

Not: Veri Kümesi çok fazla olduğu için aşağıdaki gibi sınırlandırılmıştır.

Bunun için veri setimiz train(eğitim) ve test verisi olarak parçalanmıştır. Bunun mantığını şu şekilde açıklayabiliriz. Makine öğrenmesinde özellikle de denetimli (supervised) öğrenmede modelimizi veri ile eğitiriz. Yani veriden model öğrenir ve başka veri setlerinde öğrendiklerini kullanır bize sonuçlar sunar.  Biz bu eğitimin ne kadar doğru ve sağlıklı olduğunu da test verileriyle test ederiz ki modelin kullanılabilir olup olmadığına karar verelim. Veri setimizi eğittikten sonra sınırlandırmalar aşağıda gösterildiği gibi olmuştur.

{0: 'Apple Braeburn', 1: 'Avocado', 2: 'Banana', 3: 'Corn', 4: 'Eggplant', 5: 'Fig', 6: 'Grapefruit White', 7: 'Kiwi', 8: 'Lemon', 9: 'Mandarine', 10: 'Mango', 11: 'Mulberry', 12: 'Nut Forest', 13: 'Onion Red', 14: 'Orange', 15: 'Papaya', 16: 'Peach', 17: 'Pear', 18: 'Pepino', 19: 'Pepper Green', 20: 'Physalis', 21: 'Pineapple', 22: 'Plum', 23: 'Pomegranate', 24: 'Pomelo Sweetie', 25: 'Potato White', 26: 'Quince', 27: 'Raspberry', 28: 'Redcurrant', 29: 'Strawberry', 30: 'Tomato 1'}

Daha sonra ilk adım olarak projede kullanacağımız algoritma ve kütüphaneleri projemize import ettik. Bu proje için meyve tanıtabilmek amacıyla keras ve tenserflow kütüphanelerinden yararlanılmıştır.

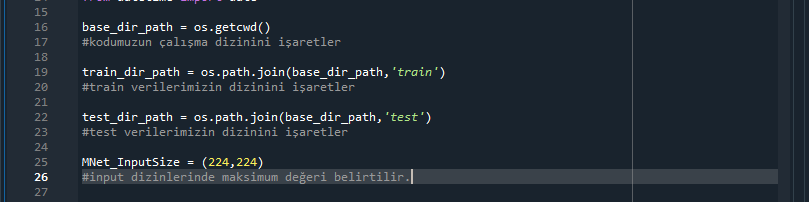


Keras : Keras, en kısa tanımıyla bir Derin Öğrenme kütüphanesidir. Yapı olarak TensorFlow ve Theano kütüphanelerini kullanarak çalışır, bu kütüphanelerin zorlayıcı yazım biçimlerine nazaran işlemlerinizi daha hızlı ve basitçe gerçekleştirmeye yardımcı fonksiyonları içerir.

Tensorflow: Derin öğrenmede en popüler araçlardan birisi olan TensorFlow, makine öğrenimi ve derin sinir ağları araştırmalarını yürütmek amacıyla Google Beyin Ekibi’nde çalışan araştırmacılar ve mühendisler tarafından geliştirilmiştir. Açık kaynak kodlu yapay zekâ ve makine öğrenmesi kütüphanesi olan ve algılama, keşfetme, sınıflama, anlama ve öngörü uygulamalarında kullanılan TensorFlow, modeller oluşturmak için veri akış grafikleri kullanır ve yazılımcıların çok katmanlı ve geniş ölçekli yapay sinir ağları oluşturmalarına olanak tanır.

**Step 1 - EDA:**

**Yöntem ve Kod:**



**Kod Çıktısı:**

Total Number of Classes in train DataSet: 95

Total Number of Classes in test DataSet: 95

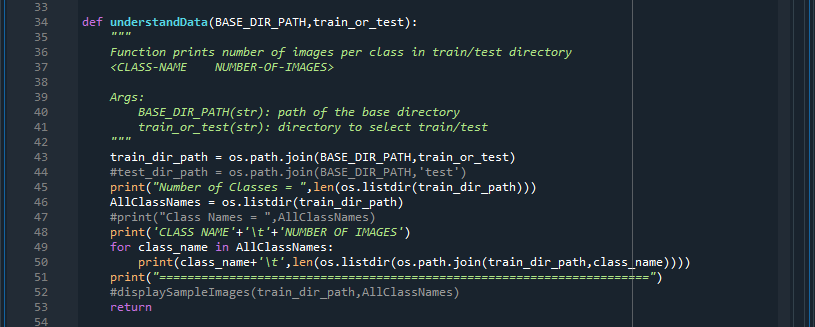
Total Number of train samples: 48905

Total Number of test samples: 16421

Bu işlemleri yaptıktan sonra test veri setindeki topma sınıf 95, eğitim veri setindeki toplam sınıf 95 olarak bulunmuştur. Daha önceden eğitim ve test verileirnde yaptığımız sınırlandırma ve birbirine yaklaştırma işlemlerinin sonuçlarını burada görmüş oluyoruz.

Eğitim veri setindeki toplam model sayımız 48905, test veri setindeki toplam model sayımız da 16421 şeklinde bulunmuş ve çıktı olarak ekrana basılmıştır.

**Yöntem ve Kod:**



**Kod Çıktısı:**

CLASS NAME NUMBER OF IMAGES

Apple Braeburn 492

Apple Golden 1 492

Apple Golden 2 492

Apple Golden 3 481

Apple Granny Smith 492

.

.

.

Buradaki fonksiyon train ve test olarak parçaladığımız verilerimizi tarayarak her bir meyve sınıfının içerisinde tanıtılacak kaç adet meyve olduğunu gösterir

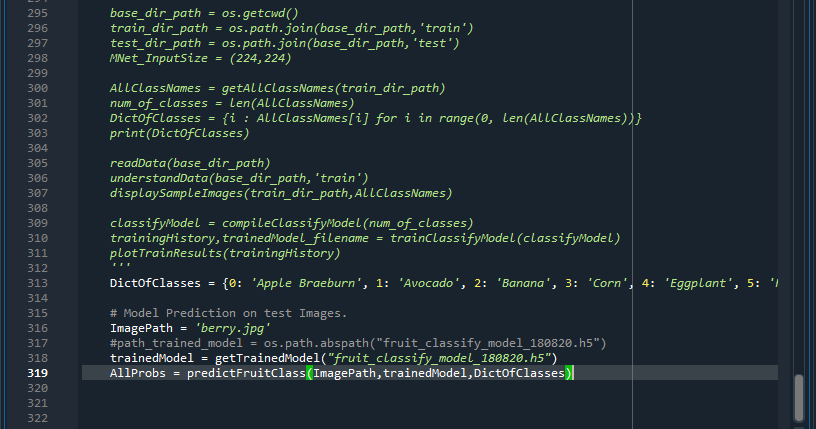
**Model Oluşturma ve Veri Kümesini Eğitme:**

**Yaklaşım:**

* Özellik çıkarımı için ImageNet veri setinde önceden eğitilmiş MobileNetV2 mimarisini kullandım.
* Daha sonra bu özellikleri kullanarak sıfırdan eğitilmiş yeni bir sınıflandırıcıyla karşılaştım.
* MobileNetV2'nin evrişimli tabanını (evrişim tabanı) aldık, bunun üzerinden yeni verileri çalıştırdık ve çıktının üstüne yeni bir sınıflandırıcı eğittik.
* Temel olarak, Yoğun katmanı ve ardından DropOut katmanı ekleyerek ve tüm ağı veri artırma ile girdi verileri üzerinde çalıştırarak conv\_base'i genişlettim.
* Ağırlıklarını güncellemekten kaçınmak için evrişimli tabanı döndürüyorum.

**Step 2 – Derleme Modeli:**

**Yöntem ve Kod:**



**Kod Çıktısı:**

Layer (type) Output Shape Param #

=================================================================

mobilenetv2\_1.00\_224 (Model) (None, 1280) 2257984

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

flatten\_1 (Flatten) (None, 1280) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_2 (Dense) (None, 512) 655872

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_1 (Dropout) (None, 512) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_3 (Dense) (None, 95) 48735

=================================================================

Total params: 2,962,591

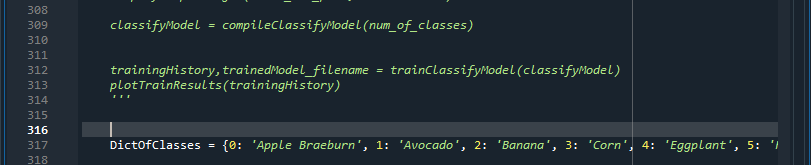
Trainable params: 704,607

Non-trainable params: 2,257,984

MobilNetv2 Mimarisi: MobileNet v2 mimarisi, artık bloğun girdi ve çıktısının, girdide genişletilmiş gösterimler kullanan geleneksel artık modellere zıt ince tıkanıklık katmanları olduğu tersine çevrilmiş bir artık yapıya dayanmaktadır. MobileNet v2, ara genişletme katmanındaki özellikleri filtrelemek için hafif ağırlıklı derinlemesine evrişimler kullanır. Ek olarak, temsil gücünü korumak için dar katmanlardaki doğrusal olmayanlıklar kaldırıldı.

### Step 3 - Eğitimin Derlenmesi Modeli:

**Yöntem ve Kod:**



Burada eğitim data setinin eğitildikten sonra derlenme aşamasıyla birlikte ekrana modelin çizilmesi işlemi yapılmıştır. Ancak programın çalışma hızını etkilediğinden dolayı yorum satırına alınmıştır. Dilendiği takdirde açılabilir.

**Eğitim Sonuçları:**

**Epcohs:20**

**Loss Fonksiyonu:** Loss fonksiyonu tasarlanan modelin hata oranını aynı zamanda başarımını ölçen fonksiyondur. Derin ağların son katmanı loss fonksiyonun tanımlandığı katmandır. Loss fonksiyonu, hata hesaplama işini problemi bir optimizasyon problemine dönüştürerek yaptığı için optimizasyon terminolojinde kullanılan objective function, cost function isimleriyle de tanımlanmaktadır. Loss fonksiyonu temelde modelin yaptığı tahminin, gerçek değerden (ground truth) ne kadar farklı olduğunu hesaplamaktadır. Bu nedenle iyi tahmin eden bir model oluşturmamışsak, gerçek değer (ground turth) ile tahmin edilen değer arasındaki fark yüksek olacak dolayısıyla loss değeri yüksek olacak, iyi modele sahipsek loss değeri az olacaktır. Birebir aynı olduğu durumda loss 0 olacaktır.

**Accuracy Fonksiyonu:** Algoritmanın performansını yorumlanabilir bir şekilde ölçmek için bir doğruluk ölçütü kullanılır. Bir modelin doğruluğu genellikle model parametrelerinden sonra belirlenir ve bir yüzde şeklinde hesaplanır. Modelinizin tahmininin gerçek verilerle ne kadar doğru olduğunun ölçüsüdür.

Kayıp, kötü bir tahminin sonucudur. Bir kayıp modelin tahmini tek örnek üzerinde ne kadar kötü gösteren bir sayıdır.

Modelin tahmini mükemmelse, kayıp sıfırdır; aksi takdirde kayıp daha büyüktür. Bir modeli eğitmenin amacı, tüm örneklerde ortalama olarak düşük kayıplı bir dizi ağırlık ve önyargı bulmaktır. Herhangi bir model için daha yüksek kayıp, daha kötüdür (kötü tahmin).

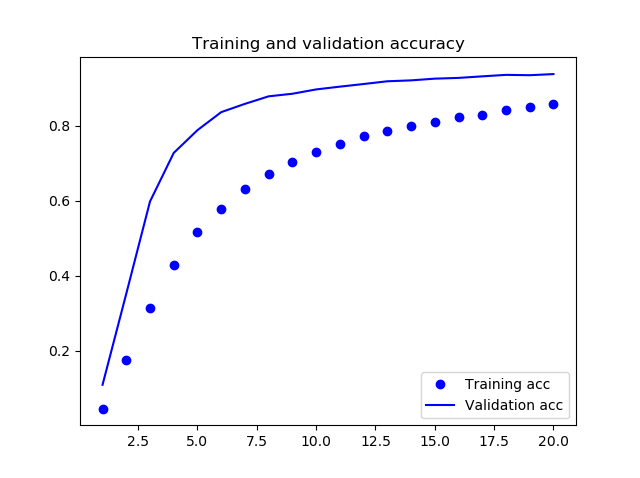
Kayıp, eğitim ve doğrulama üzerine hesaplanır ve yorumlanması, modelin bu iki set için ne kadar iyi gittiğidir. Doğruluktan farklı olarak, kayıp bir yüzde değildir. Eğitim veya doğrulama setlerinde her örnek için yapılan hataların toplamıdır.

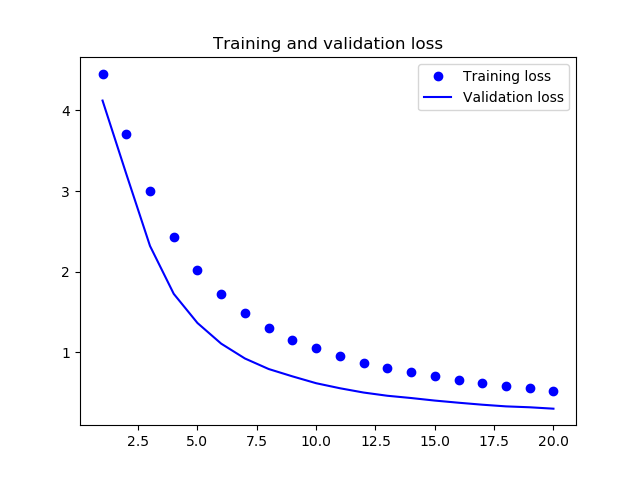
Modeli keras'ta eğitirken, doğrulama verileri için keras modelindeki doğruluk ve kayıp, farklı durumlarda farklılık gösterebilir. Genellikle her çağ arttıkça, kayıp azalmalı ve doğruluk yükselmelidir.

Ancak val\_loss (keras doğrulama kaybı) ve val\_acc (keras doğrulama doğruluğu) ile aşağıdaki gibi birçok durum mümkün olabilir:

val\_loss artmaya başlar, val\_acc azalmaya başlar. Bu, modelin öğrenmeyen değerleri sıkıştırdığı anlamına gelir

val\_loss artmaya başlar, val\_acc da artar. Bu , softmax'ın çıktı katmanında kullanıldığı durumlarda aşırı uyum veya farklı olasılık değerleri olabilir.

[](https://github.com/MeAmarP/Fruit-Classification/blob/master/results/train_valid_acc_16JUL_20epochs.png)val\_loss azalmaya başlar, val\_acc artmaya başlar. Bu aynı zamanda iyidir, çünkü inşa edilen modelin öğrenmesi ve iyi çalıştığı anlamına gelir.

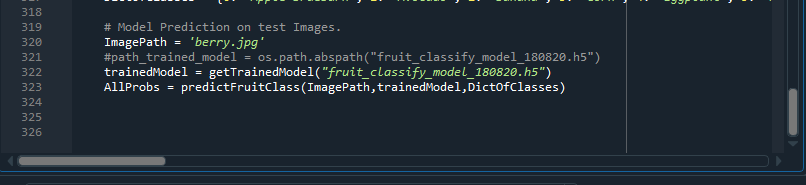
[](https://github.com/MeAmarP/Fruit-Classification/blob/master/results/train_valid_Loss_16JUL_20epochs.png)

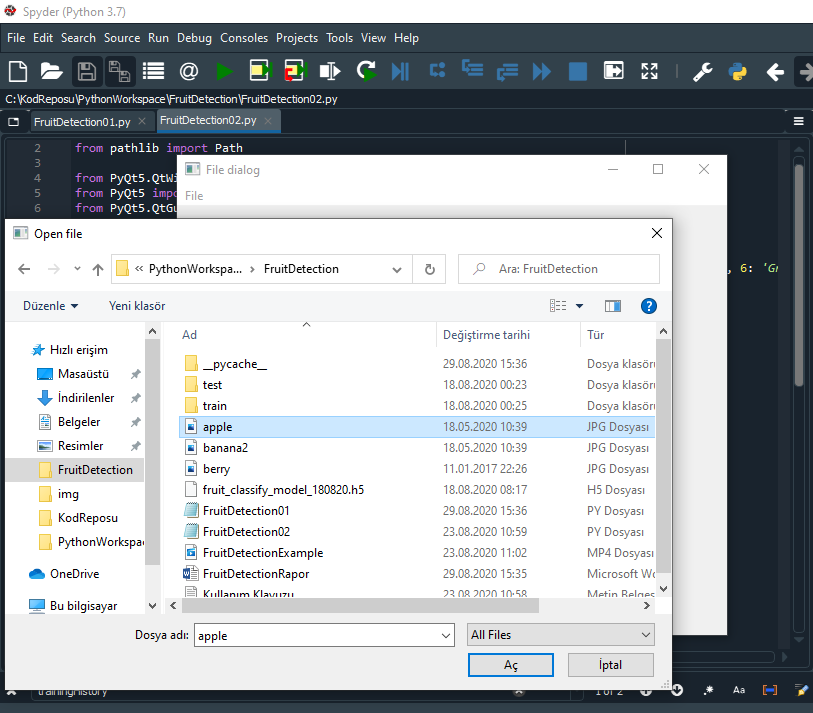
Loss ve accuracy fonksiyonlarına bizim programımız için bakılacak olursa yukarıdaki grafiklerde de görüldüğü gibi accuracy değeri oldukça yüksek loss değeri ise oldukça düşük olarak hesaplanmıştır.

Bu bize yazdığımız programın olabildiğince iyi bir şekilde meyveleri tanıdığını gösteriyor.

**Step 4 - Tahmin:**

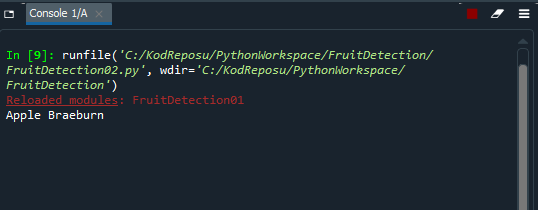
**Yöntem ve Kod:**





Çizilen arayüz ile dosyadan tanıtılmak istenen meyve seçilmiş ve tanıyabilmesi için programa yollanmıştır.

**Kod Çıktısı:**



Programın son aşaması olarak tanıtılan meyveleri programın tanıyabilmesi sağlandıktan sonra ekrana tahmin sonucu yazdırılmıştır.