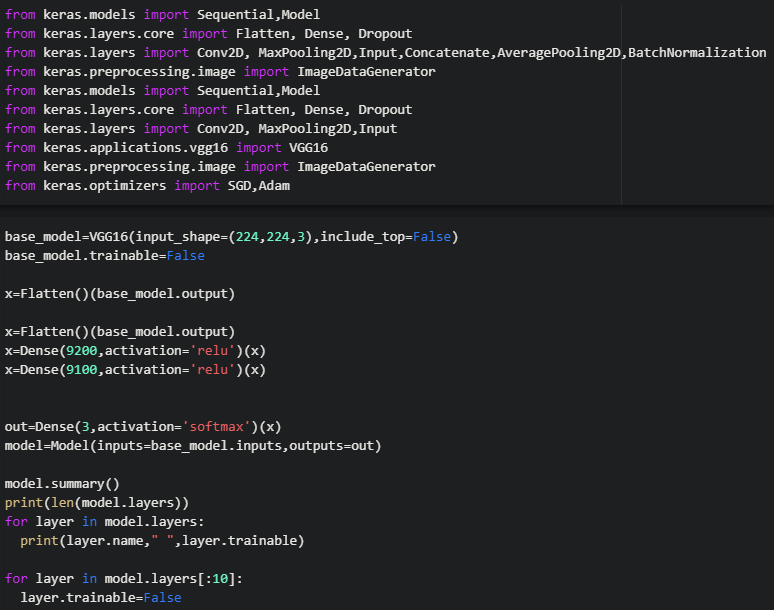
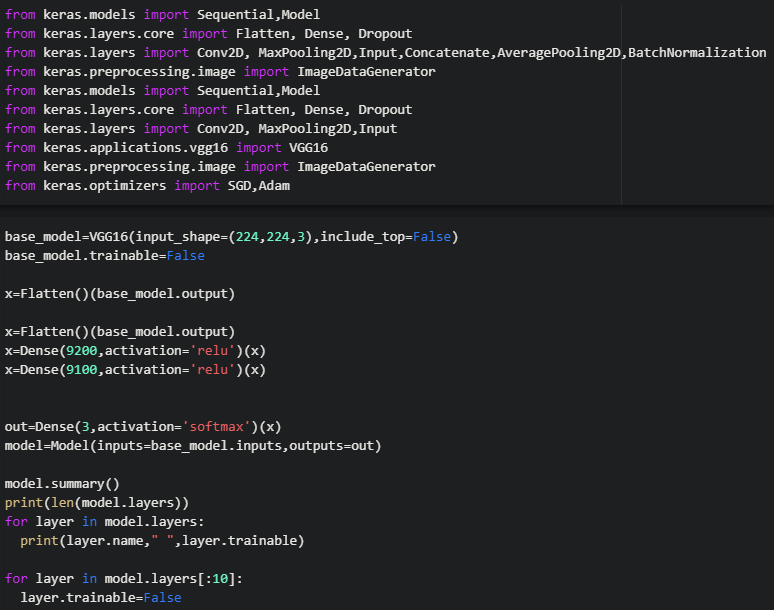
Model 1:

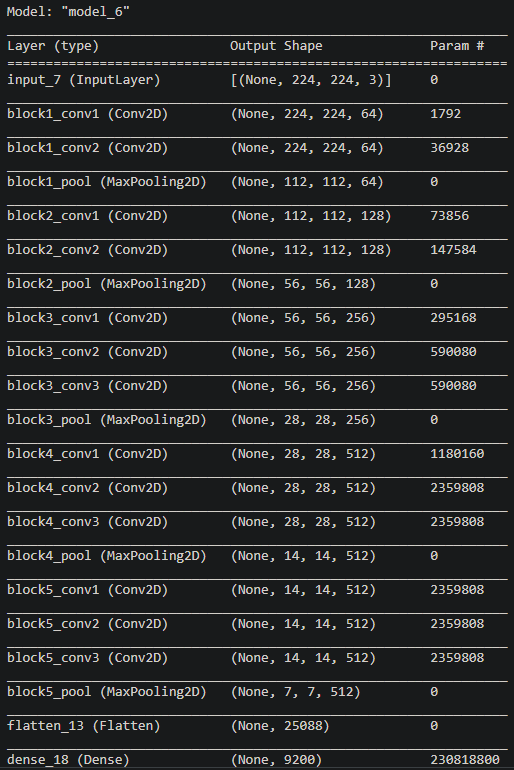


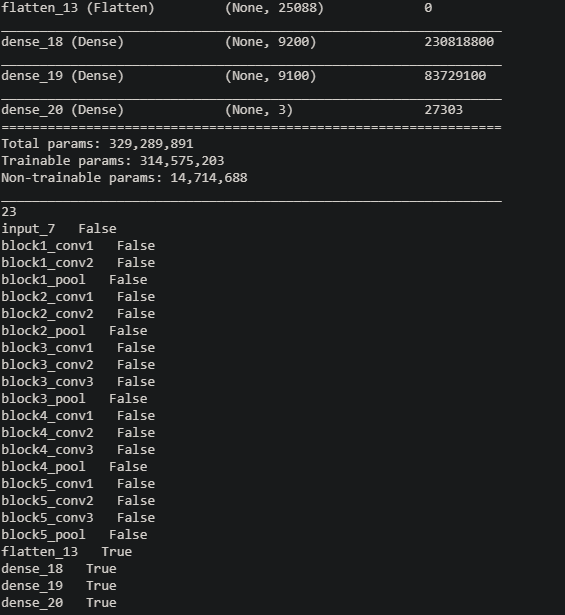
Yukarıda görmüş olduğunuz görselde ilgili kütüphaneleri ekledik ardından

Aşağıda modelimizdeki düğüm sayısını ve bu düğümler içerisindeki miktarı belirtiyoruz bu modelimizde doprout kullanmadım (ezberi önlemek için kullanılır) ama yazdığım bazı modellerde dropout kullanarak eğitimde ezberi engellemeye çalışıyorum.



Yukarıda ki kod bloğu model.layers’ı bastırılır çıktısını aşağıda göre bilirsiniz.





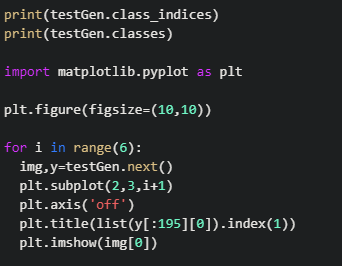
Aşağıdaki kod bloğunda drive ile bağlantımızı kurduk.



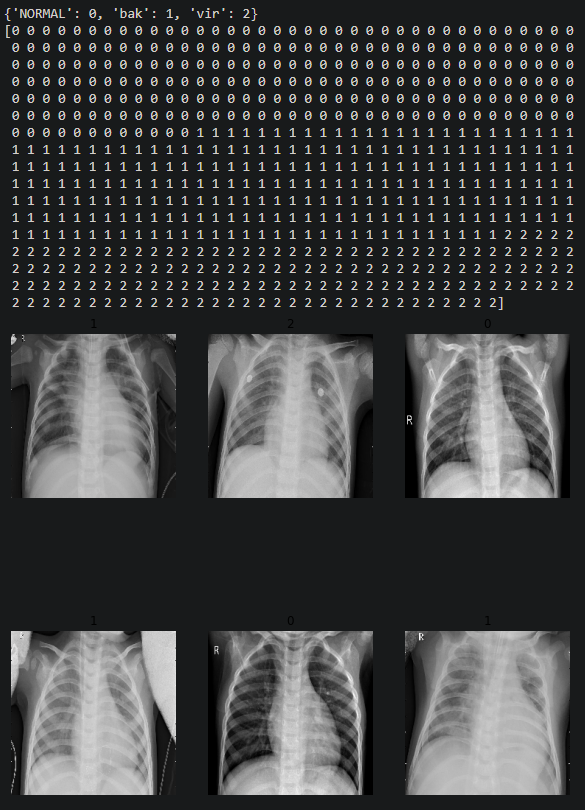
Ardından drive içerisinde bulunan dosyalarımızın bağlantısı için aşağıda görmüş olduğunuz ilk 2 satırı yazdım.



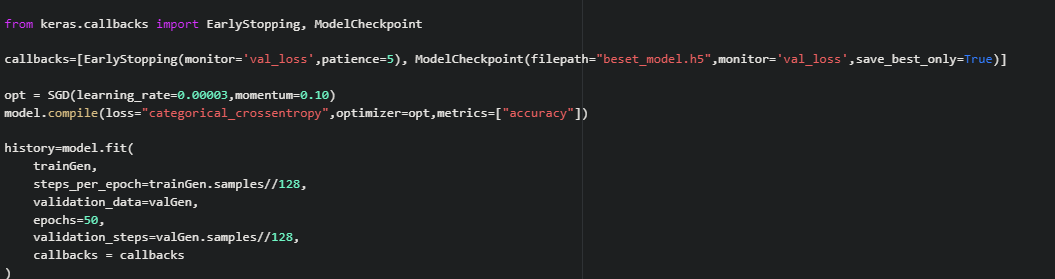
Burada target size boyutlandırma işlemi yapılıyor ardından alınan fotoğrafa zoom, verticalflip, horizanal flip, whitshift range, heihgt shift range, shear range diyerekten görselleri farklı şekilde tanıtırız aynı zamanda da validation gen 0.1 diyerek train validationluk kısımdan %10 luk bir veri ayrılır.



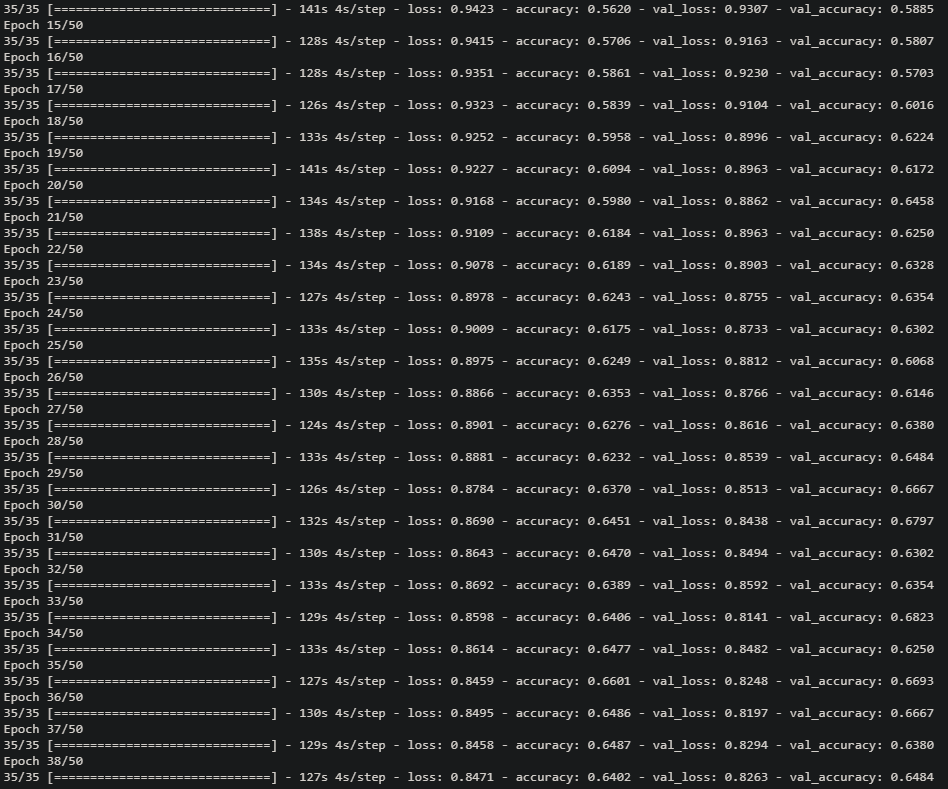
Yukarıda üstte gösterilenlere ek olarak bir kütüphane daha eklenir bu kod bloğu çalıştırıldığında ise çıktısı aşağıdaki gibi olur.



Aşağıda early stop ve model check pointi ekledik bu sayede eğer model eğitiminde 5 epoch da ilerleme olmadığını fark eder ise epochu erken keser ve alınan en iyi sonucu baz alır örneğin 30 ephoc ise 20 de kese bilir kestik den sonra ise 20 ephoc arasından gelen en iyi sonucu kaydeder. Ardından optimizer’ın SGD olduğunu belirterek devam ediyoruz SGD için bazı özel değerler giriyoruz ardından bu işlemimizi 50 epochs kadar ileri götürüyoruz.



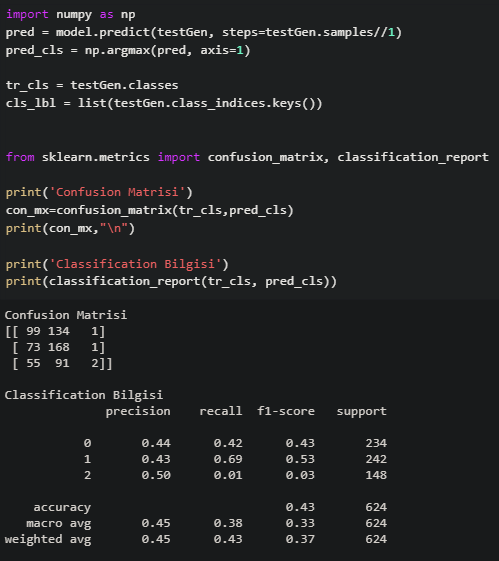
Yukarıda yapılan işlemleri konsolda görebiliriz.



Aşağıda yukarıdaki kodda elde edilen sonuçları grafik haline dönüşür. Kısaca burada grafik adlandırılması, renklerini belirlenmesi gibi şeyler yer alır.



Bu grafikten sonra modeli test etmek için aşağıdaki kodu kullanıyoruz. Ardından son olarak hem test edilen sonuçlara bakıyoruz hem de confusion matrsine bakıyoruz.

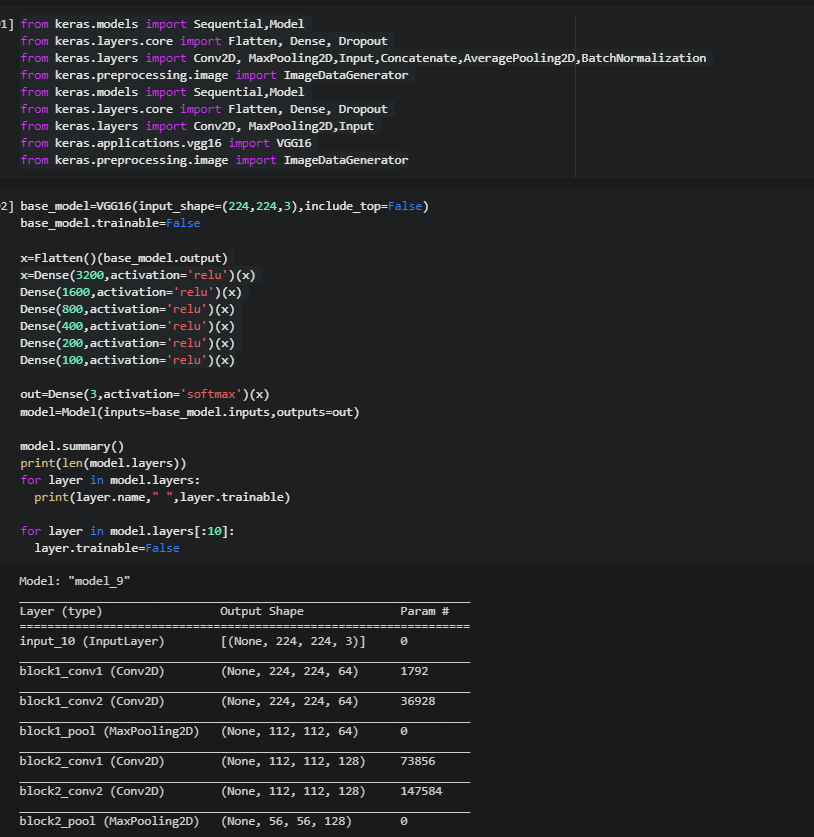


Bu modelde validation loss değeri çok yüksek ve aynı zamanda da model yeterince iyi öğrenemedi.

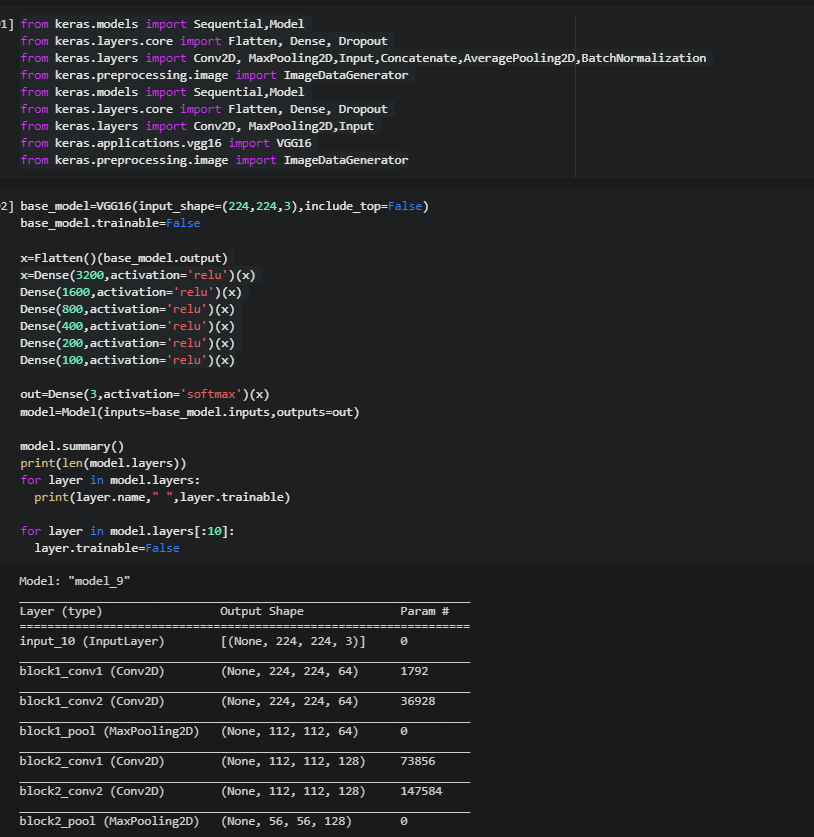
Sonuçlar:

Elimizde olan veriler birbirine çok benziyor bu yüzden ayrılmakta(öğrenme) güçlük çekiliyor. Verilerin görüntülerini değiştirip tekrar tekrar eğitmeye çalışsam da veriler bir yerden sonra gelen görüntüleri ezberlemeye başlıyor. Ezberlenen verilerde test tarafında sıkıtı çıkartıyor. Model ezber yapmadığında ise verileri düzgün bir biçimde öğrenememesinden kaynaklı olarak iyi bir neticeye ulaşamamaktayız bu da öğretilmeye çalışılan verilerin birbirine çok benzer olmasından kaynaklanıyor.

Model 2:

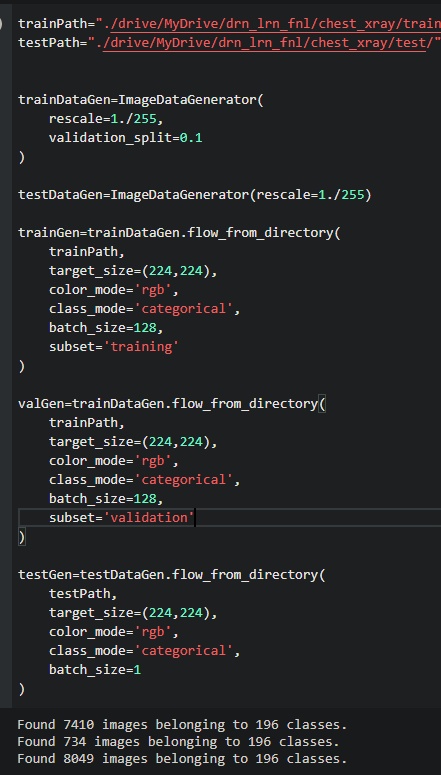
 Yukarıda görmüş olduğunuz görselde ilgili kütüphaneleri ekledik ardından

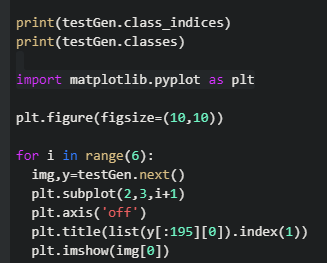
Aşağıda modelimizdeki düğüm sayısını ve içerisindeki miktarı belirtiyoruz model 1 e göre parametre sayısını azalttım. Ardından “model.layer” den gelen değerleri konsola bastırdım.



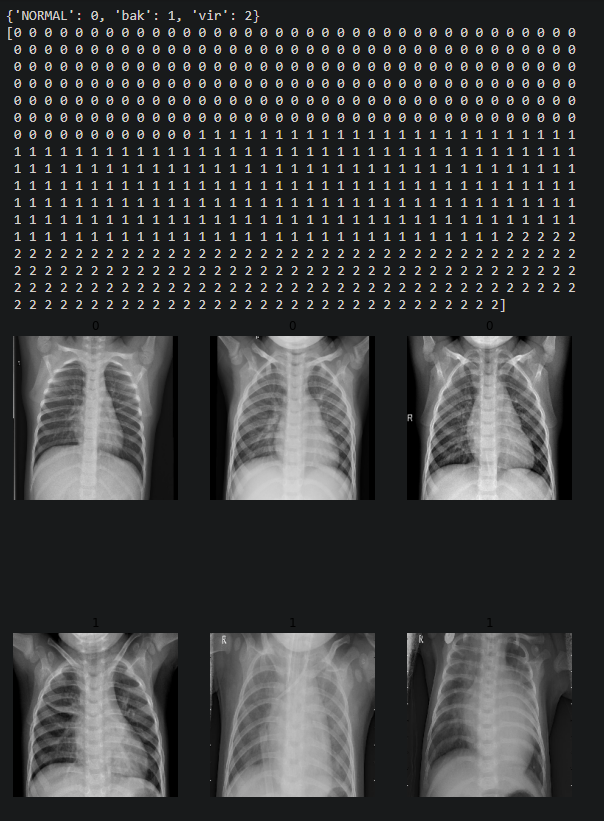
Aşağıdaki kod bloğunda drive ile bağlantımızı kurduk. 

Ardından drive içerisinde bulunan dosyalarımızın bağlantısı için aşağıda görmüş olduğunuz ilk 2 satırı yazdım.

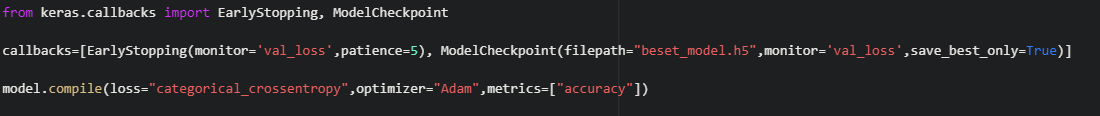




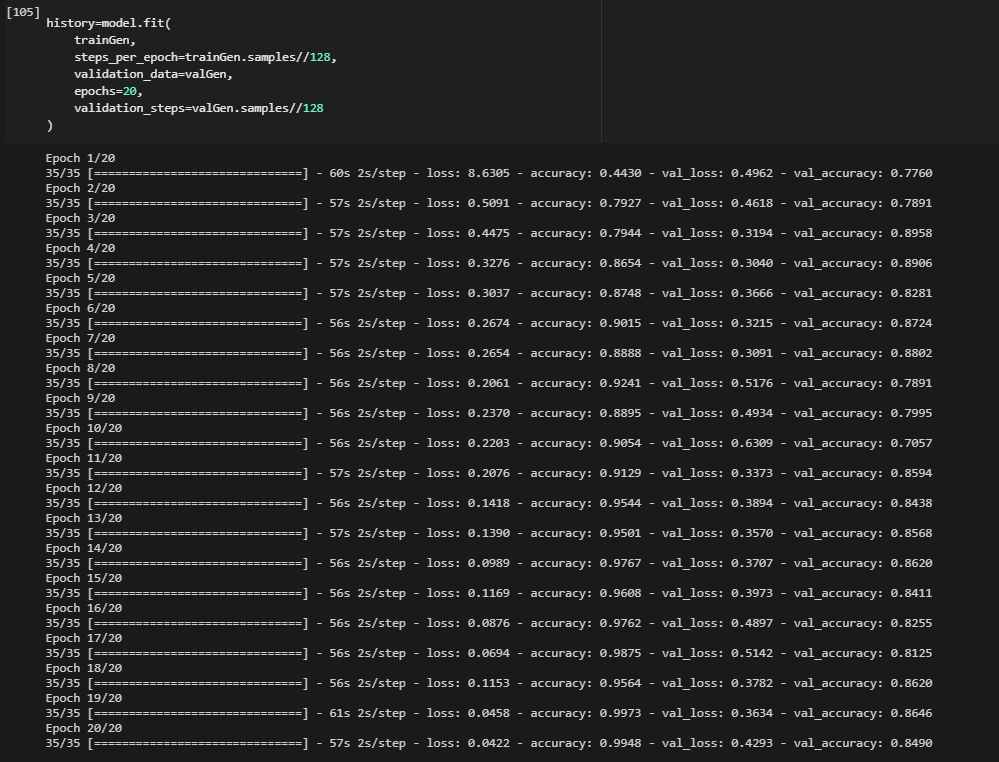
Yukarıdaki kodda çalıştığında aşağıdaki matris ve görseller oluşur



Aşağıda early stop ve model check pointi ekledik bu sayede eğer model 5 epoch da eğitim seviyesini ilerletemez ise kesip en iyi sonucu alır.

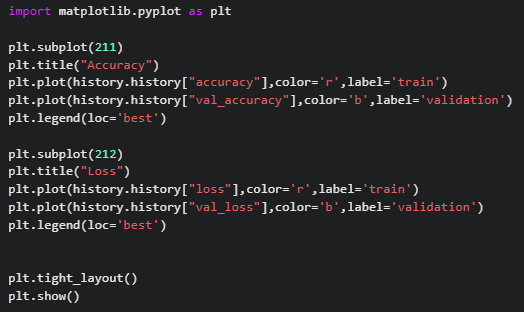


Burada 20 ephoc yazdım.



Loos ve accurary değerlerimizde baktığımda aslında çok kötü değil ama loss hala çok yüksek değerde.

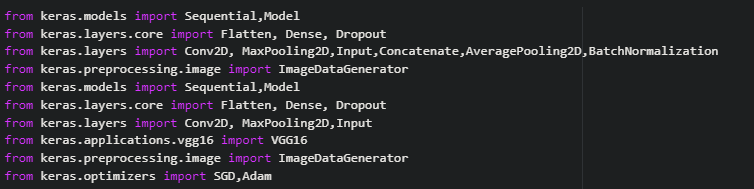
Yukarıdaki kodda elde edilen sonuçları grafik haline dönüştürüyoruz.



Grafiği aşağıdaki göreselde görebiliriz grafik çıkarıldıktan sonra ise testin gerçekleştirileceği kod bloğu geliyor.



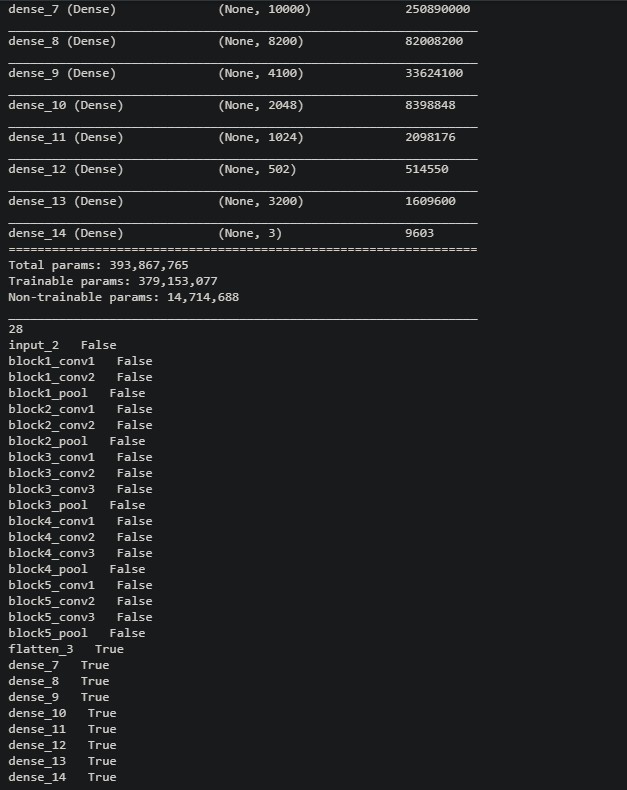
Model 3:

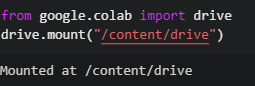


Yukarıda görmüş olduğunuz görselde ilgili kütüphaneleri ekledik ardından modelimizdeki düğüm sayısını ve bu düğümler içerisindeki miktarı belirtiyoruz bu model 2 ye göre düğüm ve değer sayısı daha fazla.

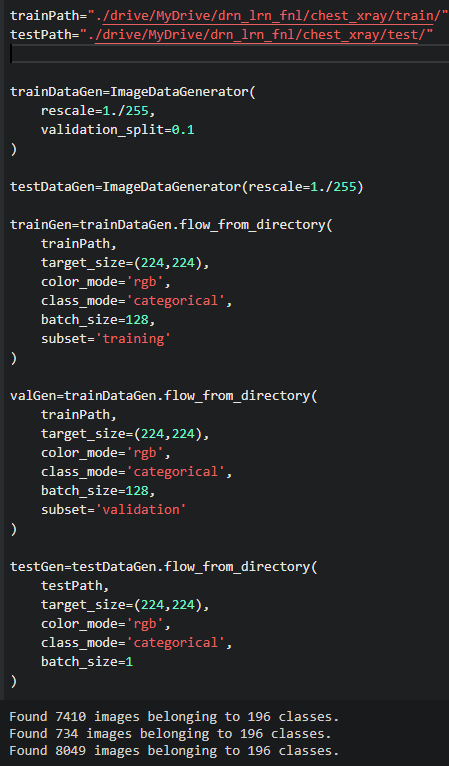
Sonra print(model.layer) diyerek modeli bastırıyoruz ardından gelen for ile model.layer içinde olan değerler konsola bastırılır.

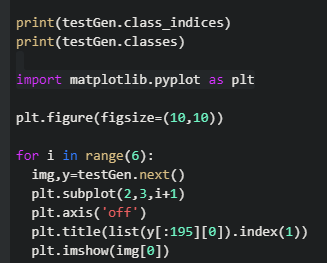




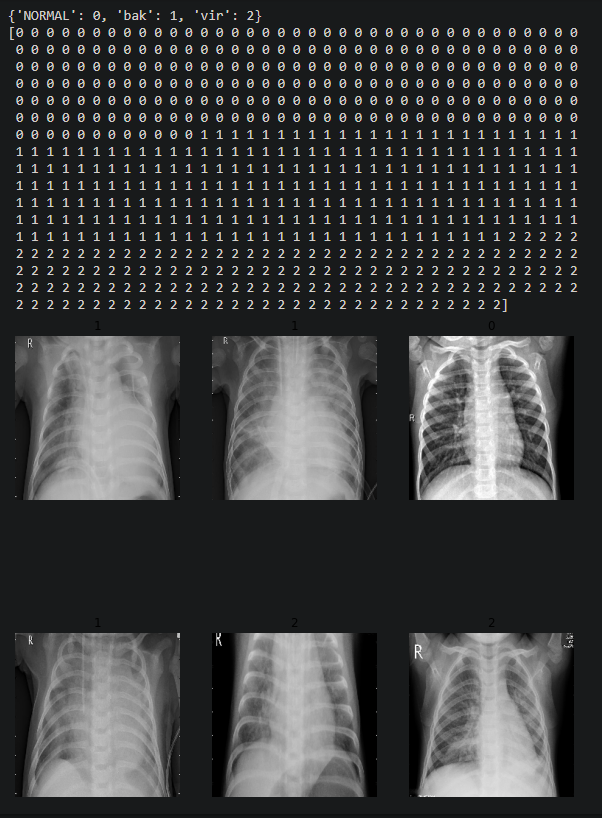


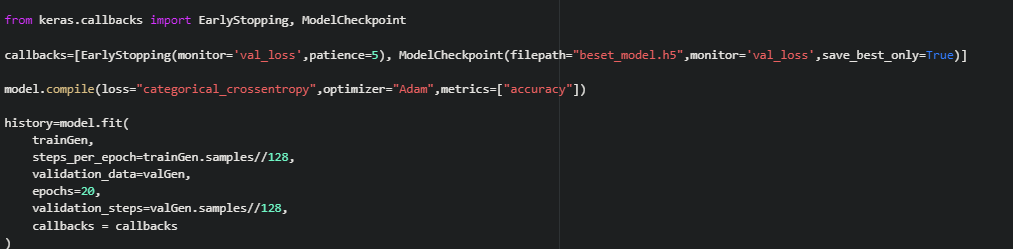
Google ile bağlantıyı kurdum yukarıda. Sonra ise Google drive içerisinde olan görsellerin konumunu belirterek projeye çektim. Bu method için ise trainDataGen de bulunan zoom horizonal siplit, vertical flip gibi değerleri kaldırdım.



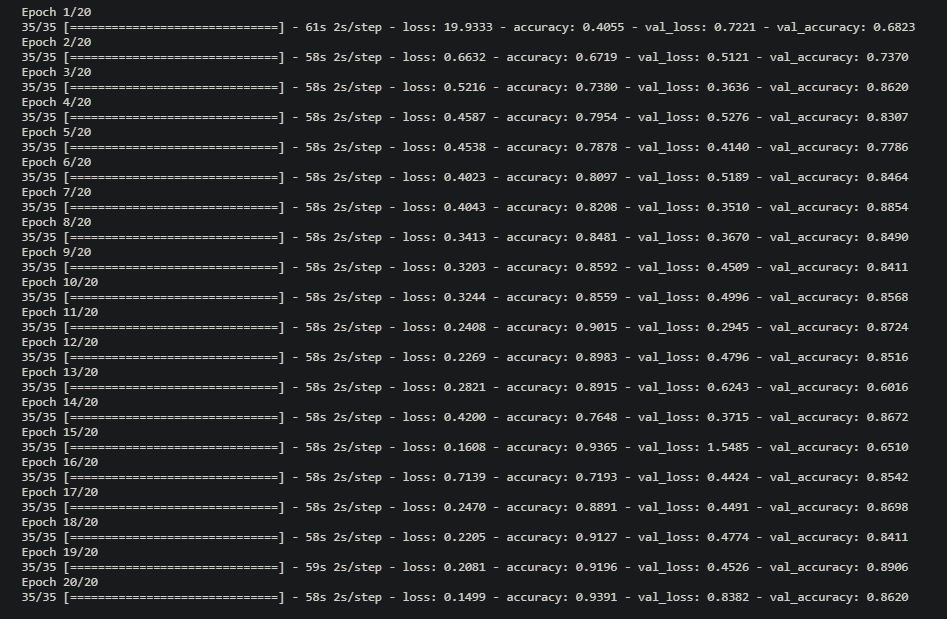


Yukarıdaki kodda bloğu çalıştığın zaman aşağıdaki çıktı oluşur

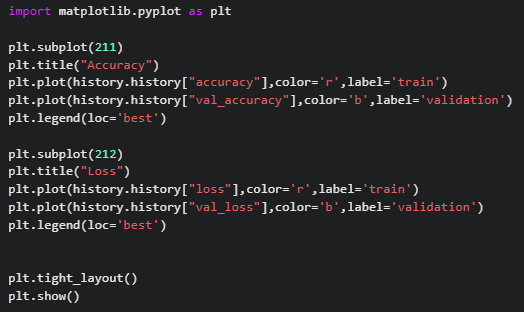


Early stop ve model check pointi ekledik bu sayede eğer model eğitiminde geriye doğru gider ise epochu erken bitirir. 

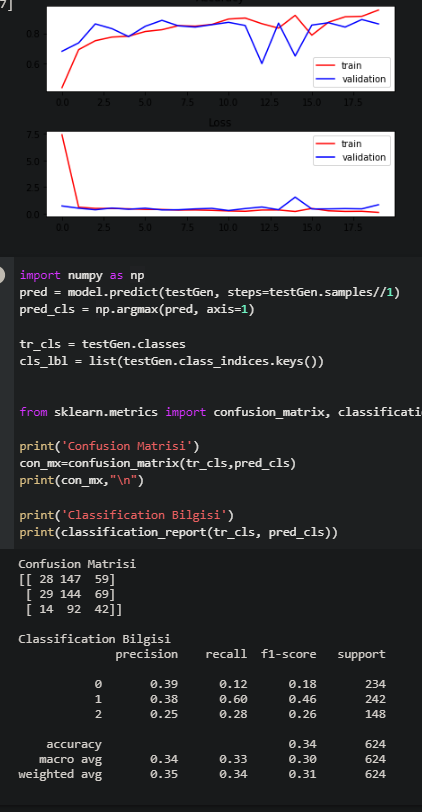
Optimizer olarak adam kullanılır aşağıda ise bu kodun değerli görülmektedir.



Eğitim seviyesinin yüksek olmasına nazaran Loss değeri çok yüksektir bu da ilerideki aşamalarda sıkıntı çıkmasına sebep olacaktır.

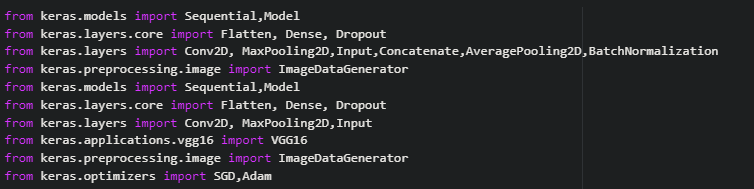


Grafiği görmek için yukarıdaki kod bloğunu yazarım. Ardından da model test edilir. Modelin eğitim seviyesi yüksek fakat ezberliyor bunun için dropout kullanarak ezber seviyesi düşürülmeli.

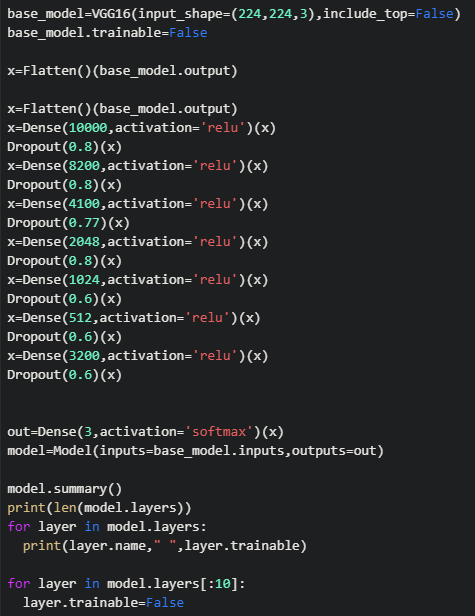


Model 4:

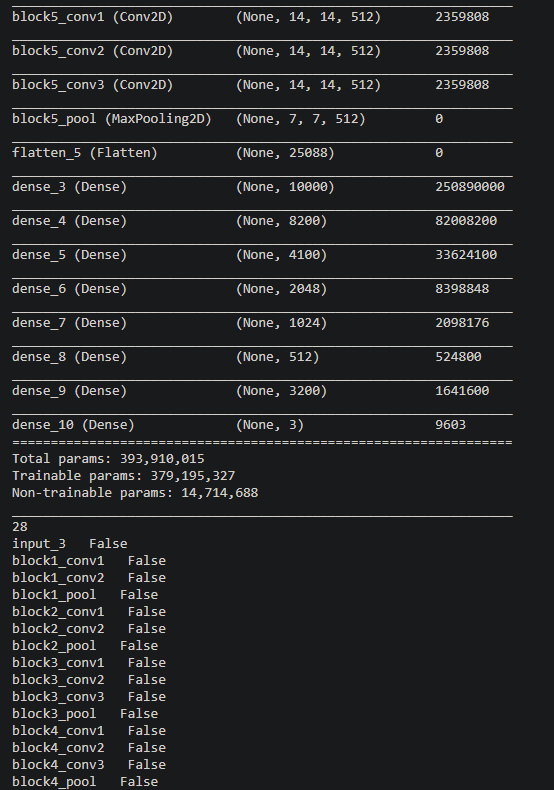
Aşağıda görmüş olduğunuz görselde ilgili kütüphaneleri ekledim



Her bir düğüm arasına dropout yazılarak ezber yapmasını engellemeye çalıştım. Sonrasında yine çıktıları aşağıda göstermesi için for ve print yapısını kullandım.



Çıktı şekli aşağıda ki gibidir.

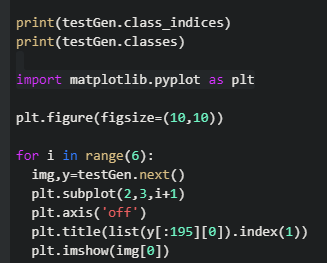


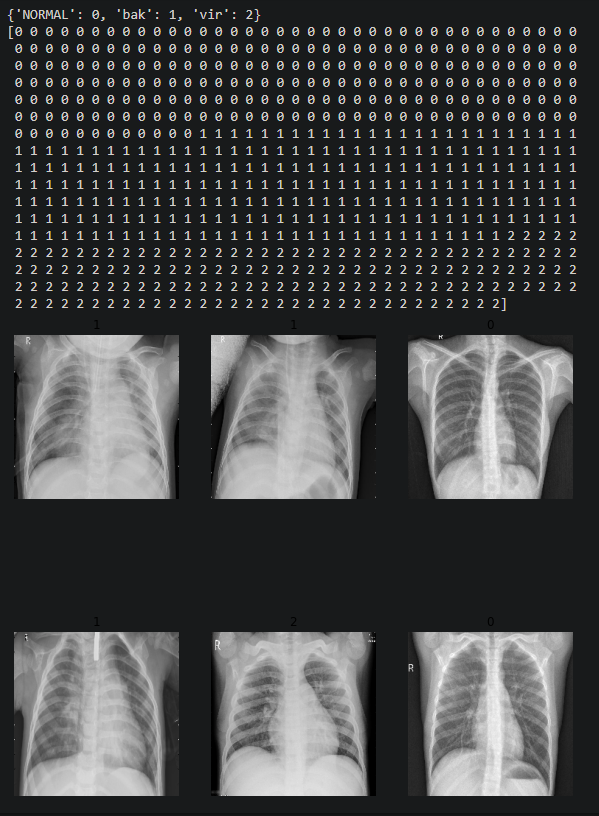
Aşağıda drive ile bağlantı kuruldu.

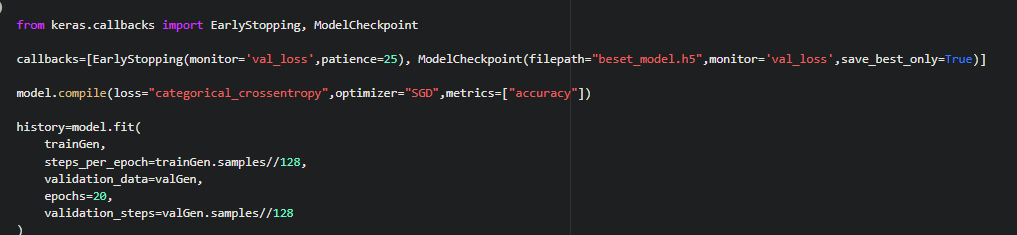


Drive bağlandıktan sonra drive içerisinde ki dosya konumunu belirtirim aşağıdaki 2 satır bunun içindir.

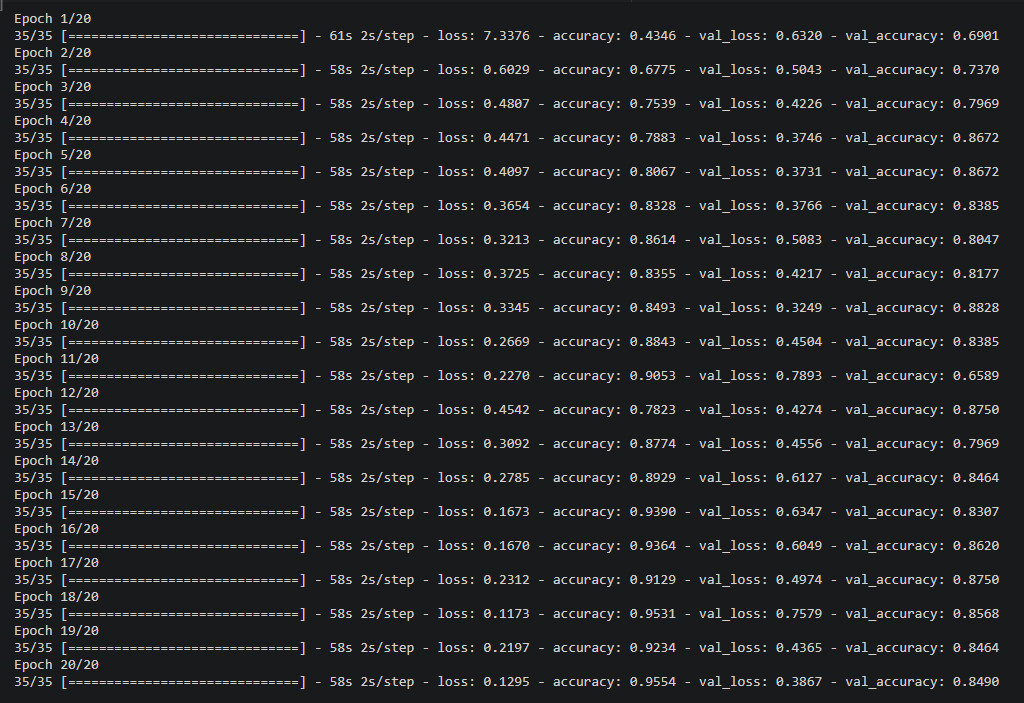




Yukarıda belirtilen kod çalıştığın zaman aşağıdaki çıktıyı verir

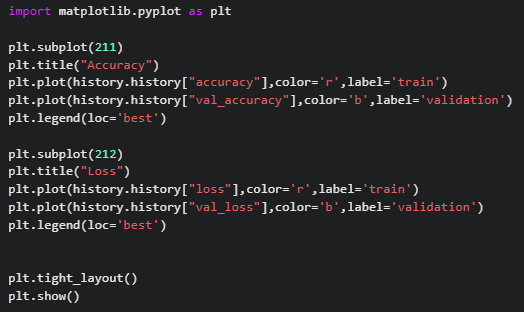


Yukarıda 20 ephoc ileri gider aynı zamanda da önceki modellerdeki gibi early stop ve model check poin kullanılır.

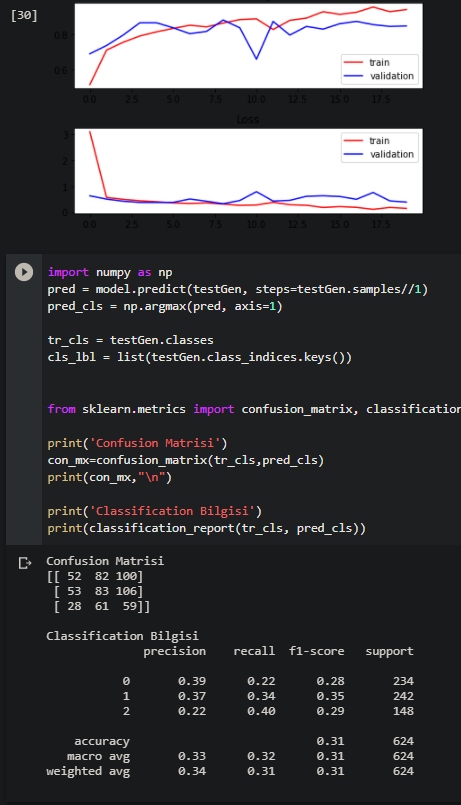


Loss değerinin önceki modellere göre daha az olduğunu ve aynı zamanda da accuracy seciyesinin yüksek olduğunu görülmektedir.

Sonrasında ise kodda elde edilen sonuçları grafik haline dönüştürüyoruz.

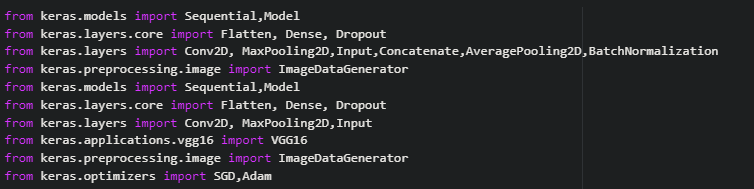


Grafiği çizdirdikten sonra ilgli matris bastırılır ve test edilir.



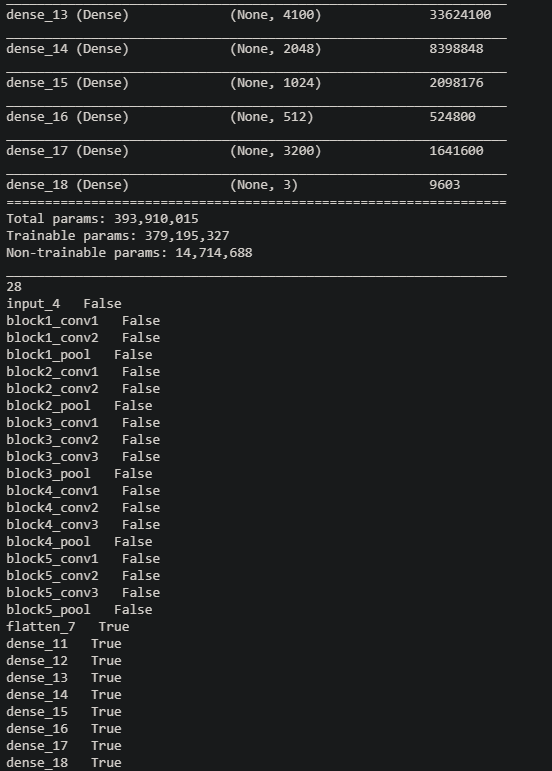
Model 5:

İlgili kütüphaneleri aşağıda ekleriz ardından.



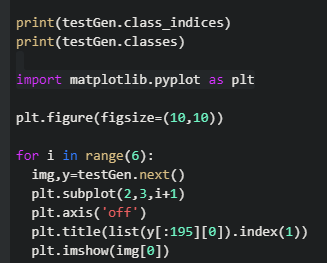
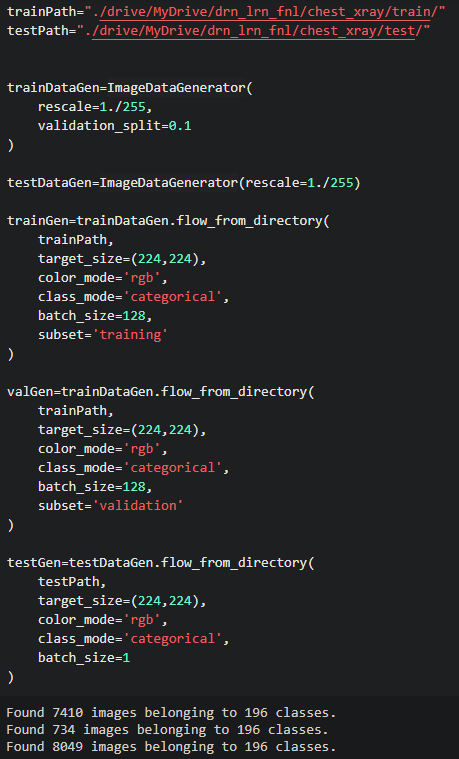
Düğüm sayısı belirlendi önceki modele göre (Model 4) Dropoutların bazılarını kaldırdım ve seviyelerini düşürdüm. İlerleyen kod kısmında da modeli konsola bastırdım.





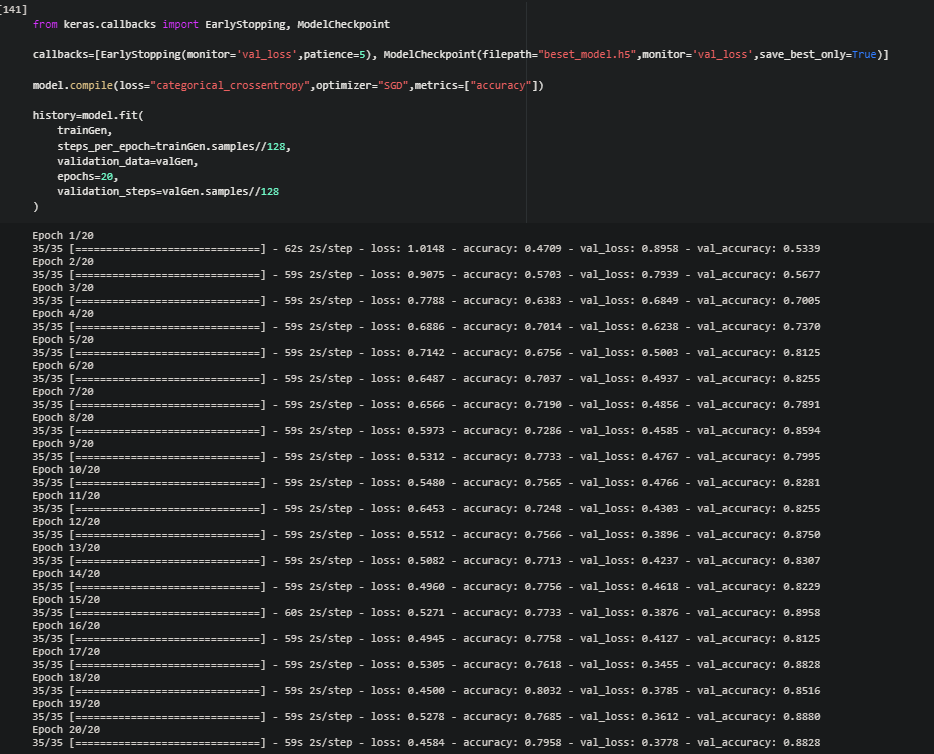
Kodun konsol çıktısı yukarıdaki gibidir. Ardından drive bağlantısı kurulur.



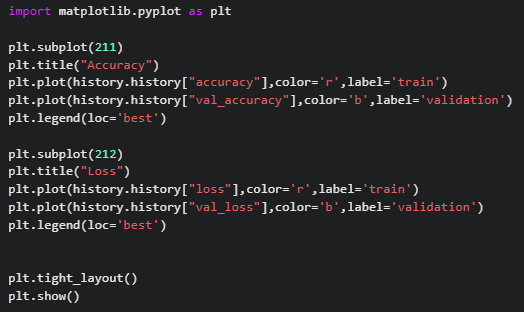
Direve bağlantısı sağlandıktan sonra dosyaların olduğu konum belirlenir.

Yukarıda belirtilen kodun çalışması ile birlikte aşağıdaki çıktıyı verir. 

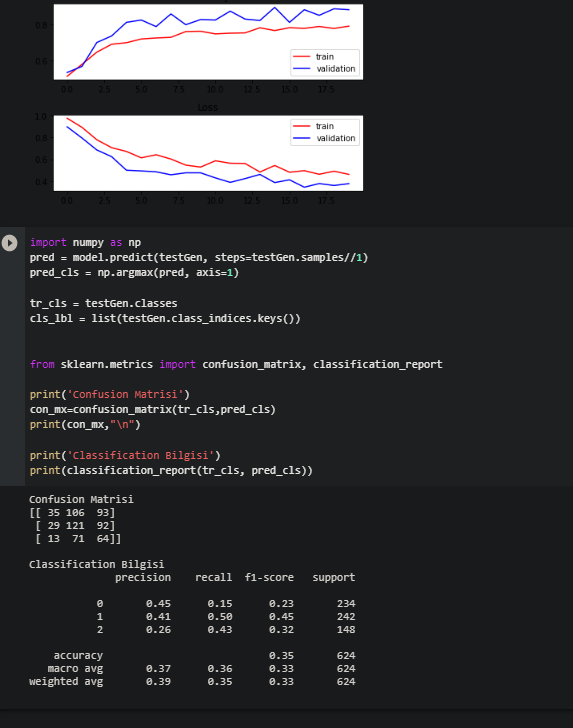
Model optimizeri olarak burada SGD kullanılır aynı zamanda da önceki modellerde olduğu gibi early stop ve modele check pointler de bu modelde kullanılır.



Matploitlib.pyplot kütüphanesi kullanılarak yukarıda ki çıktının grafiksel hale getirilmesi sağlanır.

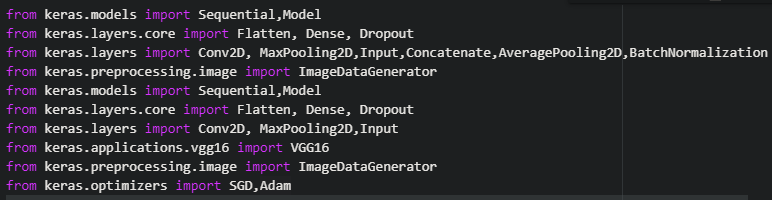


Yukarıdaki kod çalıştırıldıktan sonra aşağıdaki gibi bir grafik çıkar.



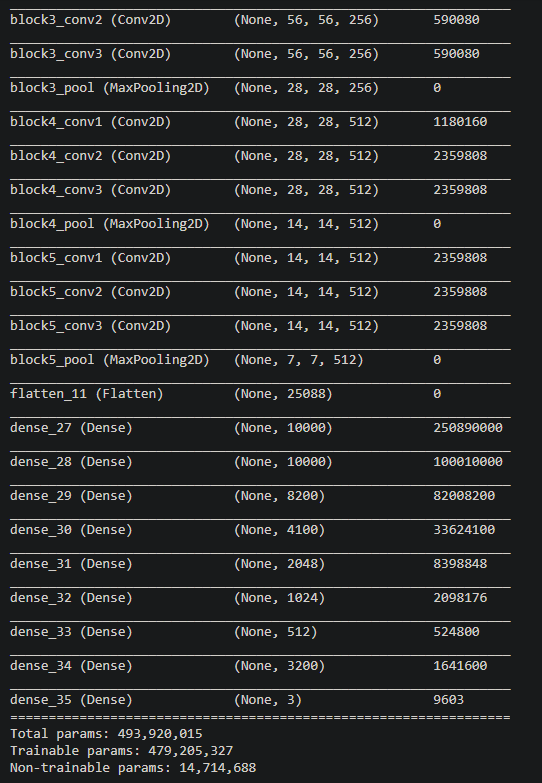
Grafik çıkarıldıktan sonra ise matris bastırılır ve veriler test edilir model önceki modele göre daha az dropout ve daha düşük dropout seviyesi verilmesine rağmen gözle görülür çok büyük bir fark yoktur arada.

Model 6:



Yukarıda ilgili kütüphaneler eklendi. Aşağıda ise bir önceki modele göre (Model 5) düğüm sayısını ve düğüm içerisinde bulunan değeri artırdım aynı zamanda da önceki modele göre her bir düğüm arasına Dropout ekledim.

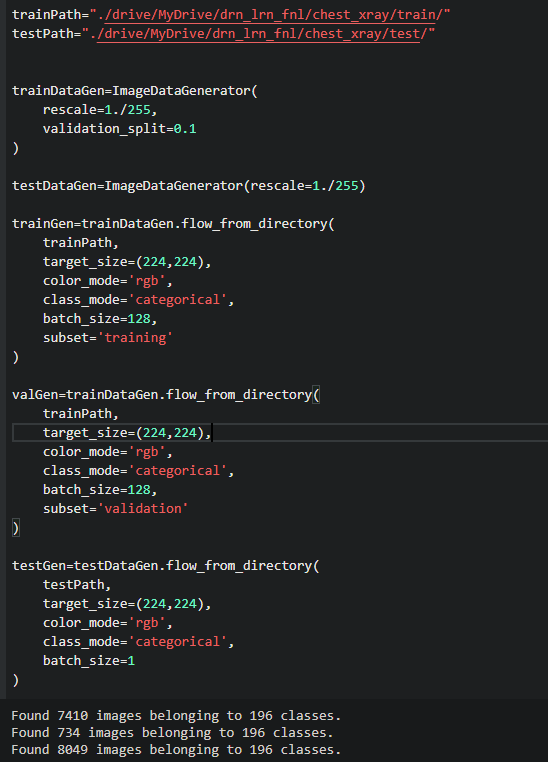


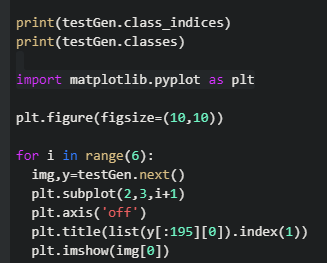
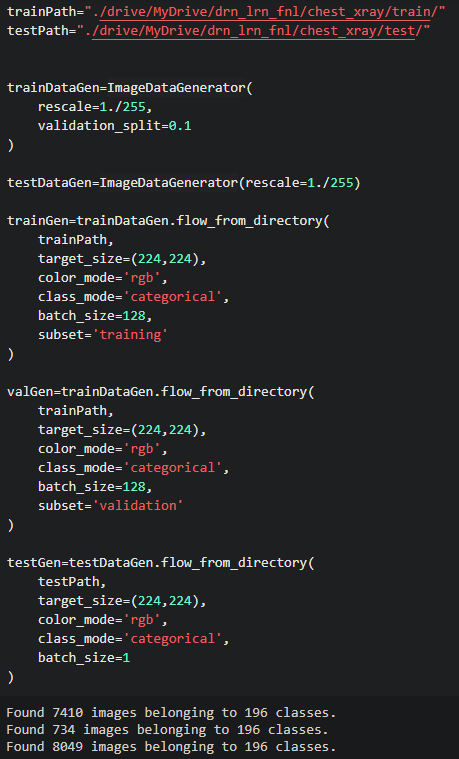


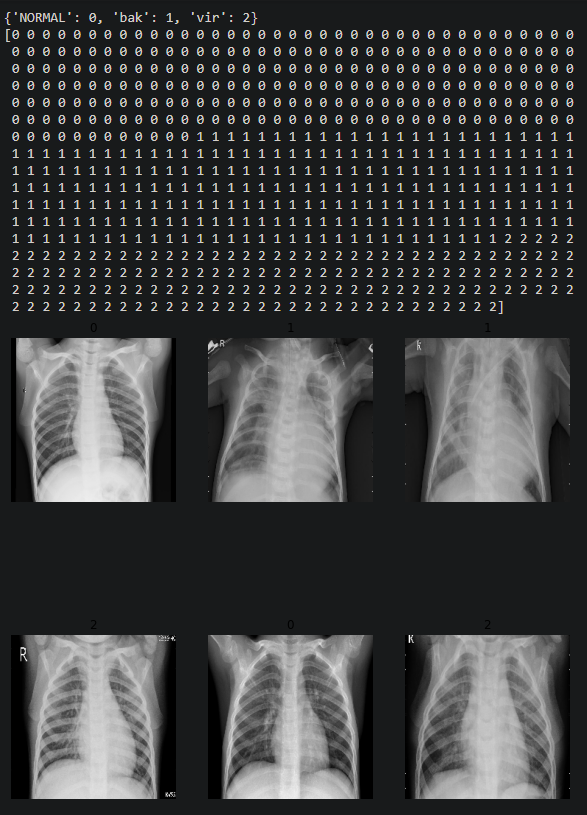
For ve print ile model.layers’ın değerlerini konsola bastırmış oldum. Aşağıdaki kod bloğunda drive ile bağlantı gerçekleştirdim.

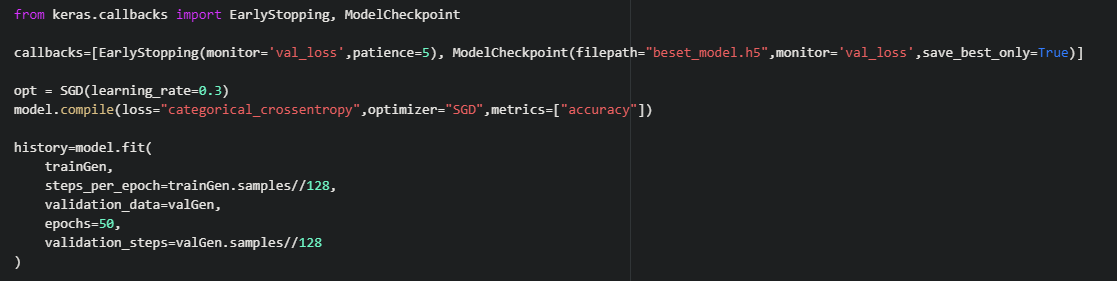


Bağlantı yapıldıktan sonra ise aşağıdaki ilk iki satırda train ve test dosya yolunu belirttim validation split ile %10 luk kısmı validation için ayırmış oldum.

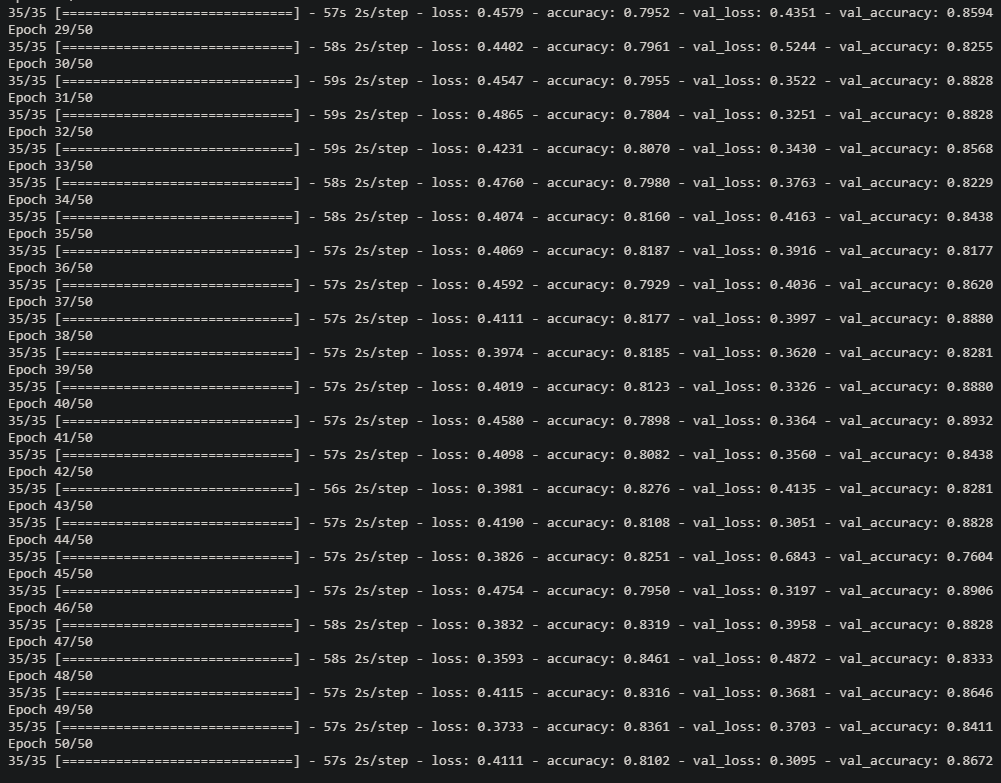


.

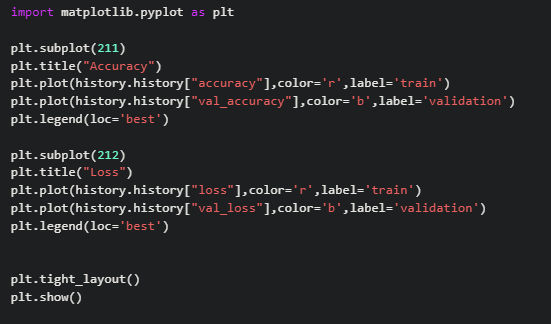
Yukarıda ki kodun çalışması dahilinde aşağıda görmüş olduğunuz çıktıyı verir çıktıyı verir. 



Yukarıda ise optimizer olarak SGD kullandım ve verileri öğrenme de herhangi bir ilerleme olması dahilinde early stop yapacak ve en iyi eğitilen noktayı baz alarak test edecek. Aşağıdaki görselde yukarıdaki kod bloğunun çıktısını görmektesiniz. Burada eğitimimiz aşamasında early stop kullanmaya gerek kalmamış 50 ephoc ilerle demişiz ve modelde 50 ephoca kadar gidebilmiş.

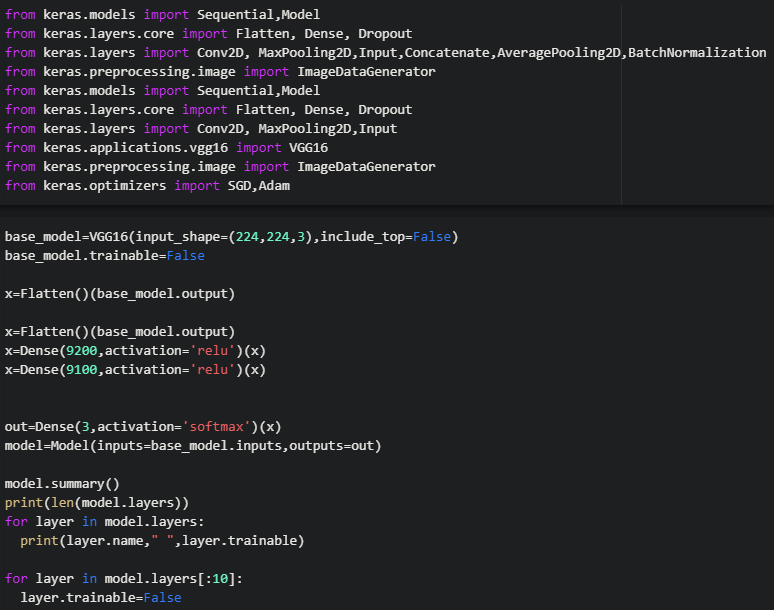


Elde edilen sonuçları grafik haline dönüştürme için aşağıdaki kod bloğunu kullanırız.

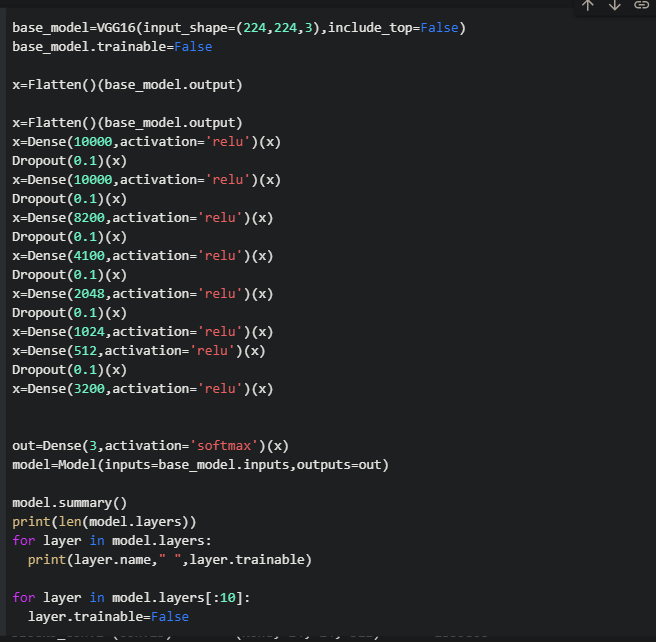


Yukarıda bulunan görselde hem grafiği hem de test sonuçlarını görmekteyiz. Bu modelimiz bir önceki modele benzer bir sonuç çıkardı fakat önceki model 20 ephoc gider iken bu modelde 50 ephoc ileri gitti. Ephoc sayısının artmasına rağmen sonuç olarak pek bir şey değişmedi.

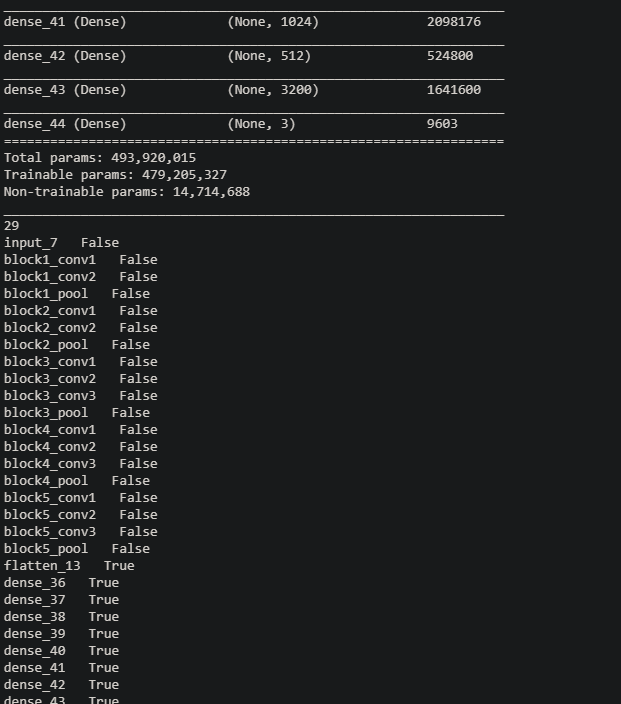
Model 7:



Yukarıda görmüş olduğunuz görselde ilgili kütüphaneleri ekledik. Sonrasında ise önceki modelimizden farklı olarak dropout değerleri 0,1 e düşürdüm.



Konsolda ki çıktısını aşağıda göre bilirsiniz.

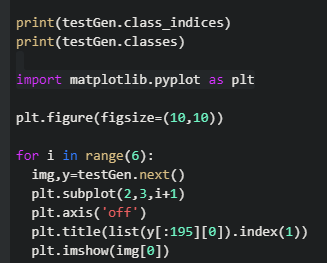




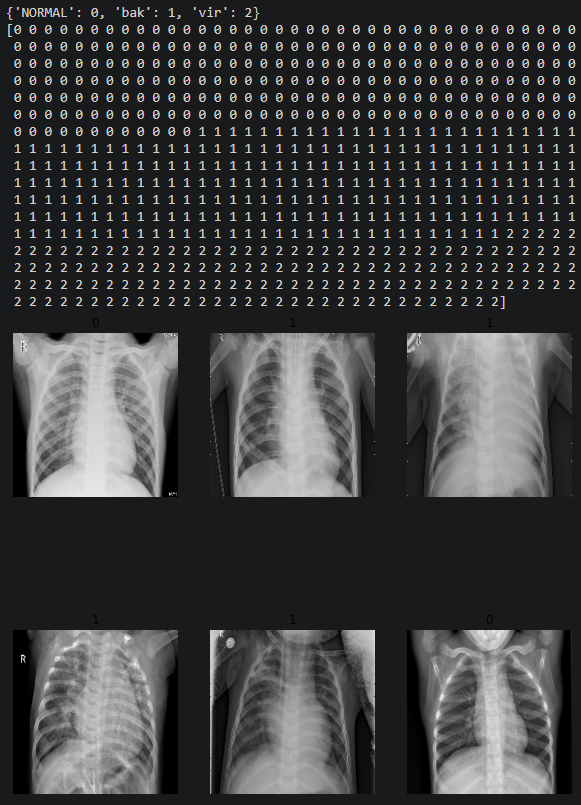
Yukarıdaki kod bloğunda Google drive bağlantısı yapılır.

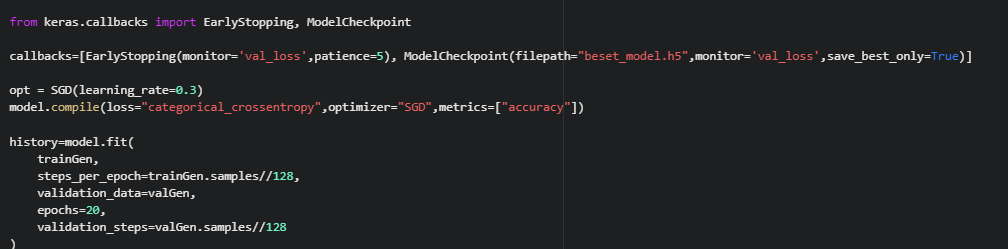
Aşağıda ki iki satırda verilerin dosya konumları belirtilir ardından bu model de önceki modelde kullandığım gibi zoom range, horizonal flip, vertical flip gibi değerleri kullandım fakat bu modelin dense değeri ve dropout değerleri diğerlerinden farklı.



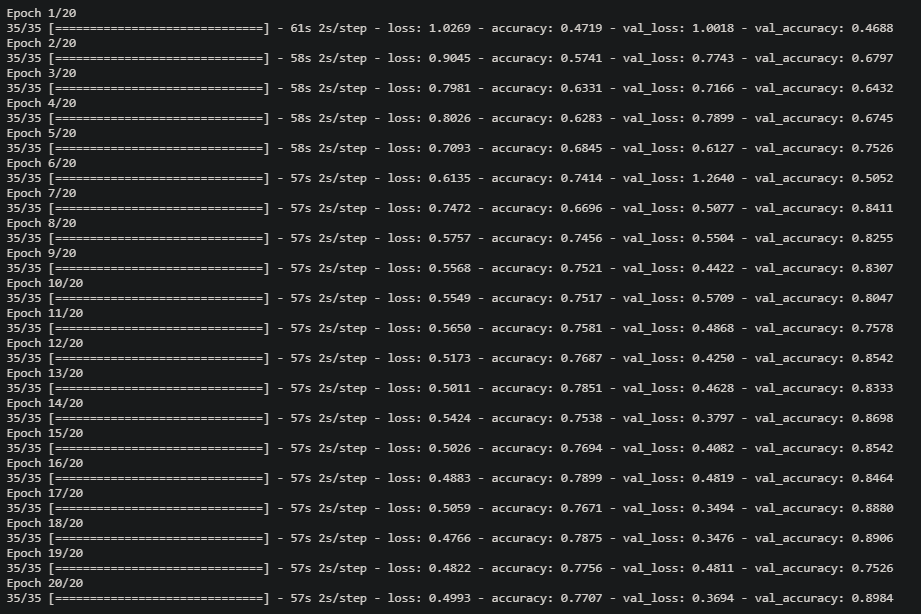
.

Yukarıda ki kodun çalışması dahilinde aşağıda görmüş olduğunuz çıktıyı verir çıktıyı verir.

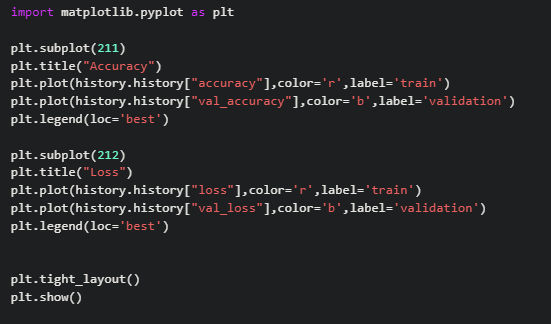




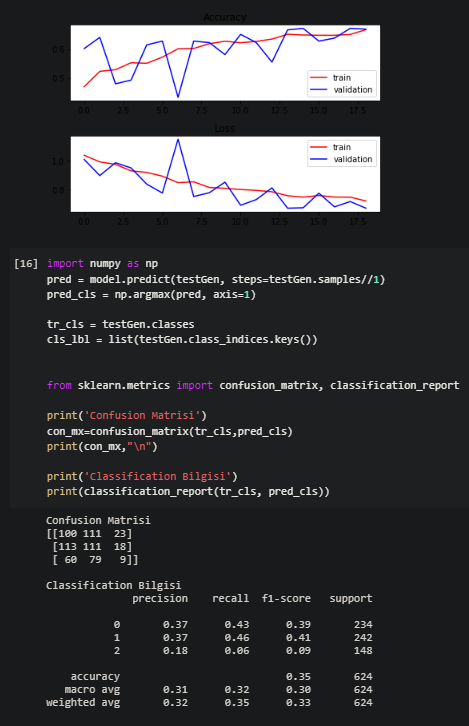
Yukarıdaki kod bloğu için early stop ve model check point kullanırım optimeze olarak ise SGD kullanırım.



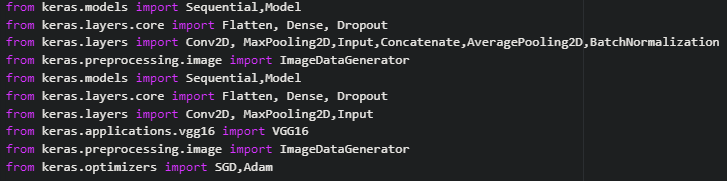
Aşağıda ilgili kütüphane dahil edildikten sonra grafik çizdirilmesi için kodlar vardır bu kod bloğu çalıştığı zaman

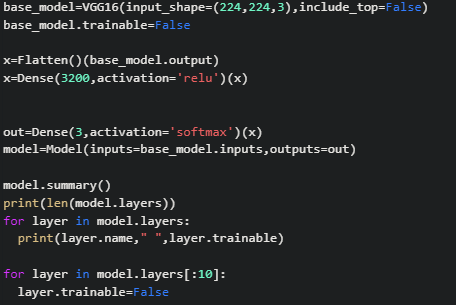


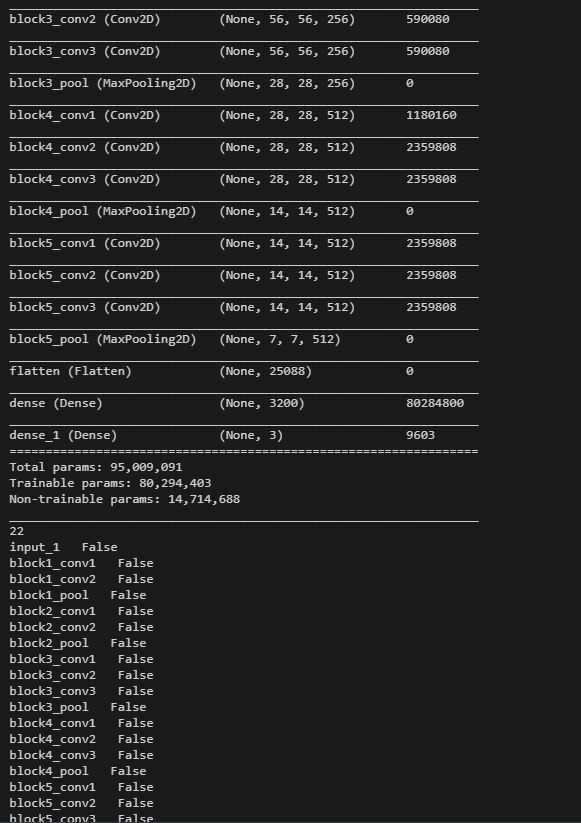
Aşağıdaki çıktıyı verecektir. Çıktı sağlandıktan sonra ki diğer kod bloğunda confusion matrisi ile test verilerini görmekteyiz.



Model 8:

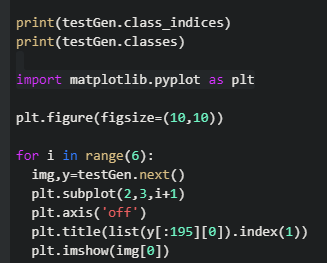


Yukarıda ilgili kütüphanelerimizin tümünü ekledim ardaından sadece bir adet düğüm ekledim ve dropout değeri koymadım.

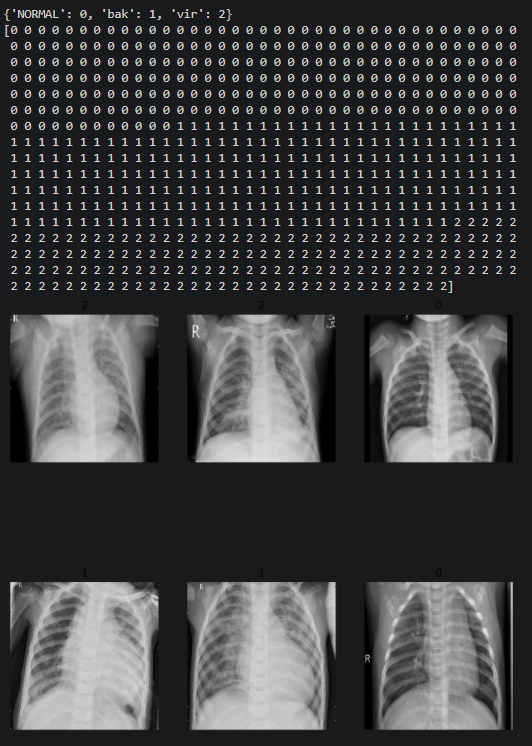
Sonrasında yukarıdaki kod bloğu çalıştırıldığında aşağıdaki gibi bir çıktı verecektir. 

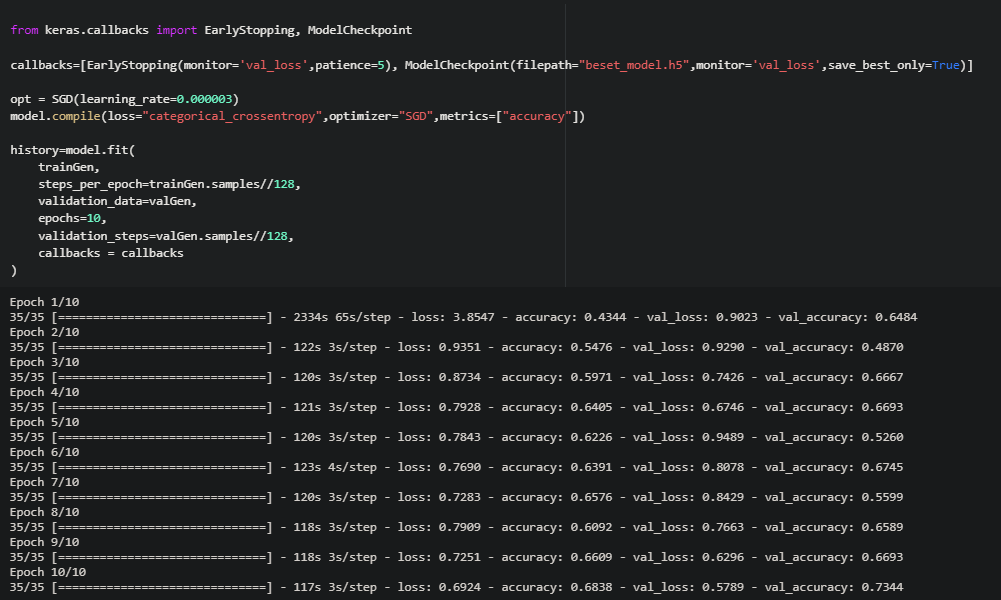


Yukarıda diğer modellerinden farklı olması için bazı değerler değiştirilmiştir. Örneğin width shift range, xoom range gibi değerleri değiştirdim ve validation split diyerek train validationluk kısımdan %10 luk veri ayrılır.

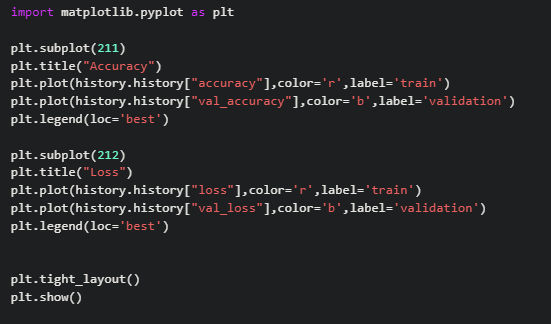
.

Yukarıda ki kod çalıştığında aşağıdaki çıktıyı verir.

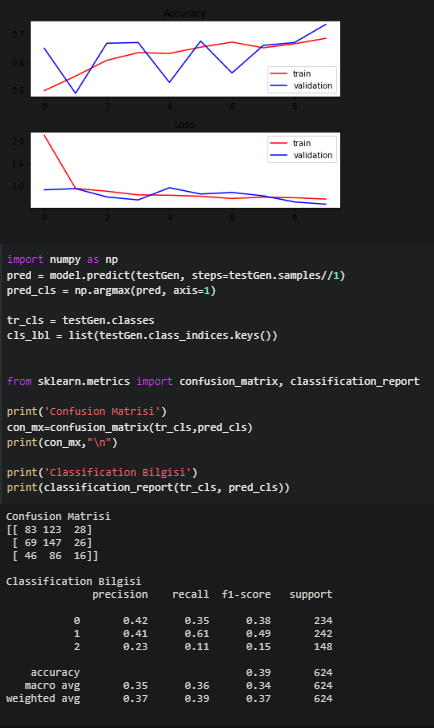




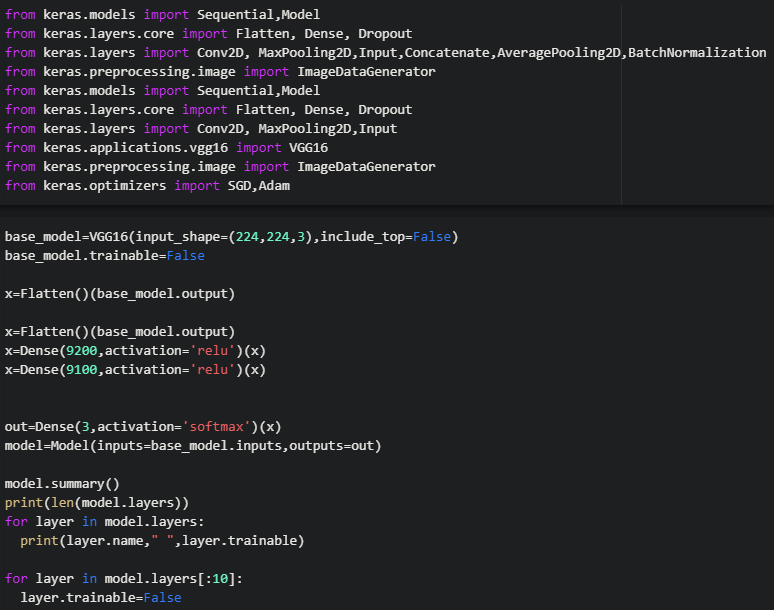
Yukarıda eğer model belirli bir yerden sonra eğitimde daha kötüye gitmesi durumunda durması için early stop kullanıldı. Optimizer olarak SGD kullanıldı.

Yukarıda konsolda görmüş olduğunuz verilerin grafiğini çıkarmak için ise aşağıdaki kod bloğu kullanılır.

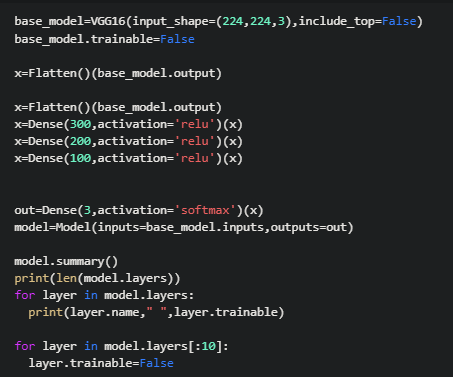
Bu kodun çıktısı aşağıdaki gibidir grafik çizdirildikten sonraki kod bloğunda çıkan matris ve test sonuçlarını görmekteyiz.



Model 9:



Yukarıda görmüş olduğunuz görselde ilgili kütüphaneleri ekledik ardından

Aşağıda modelimizdeki düğüm sayısını ve bu düğümler içerisindeki miktarı belirtiyoruz bu modelde. 

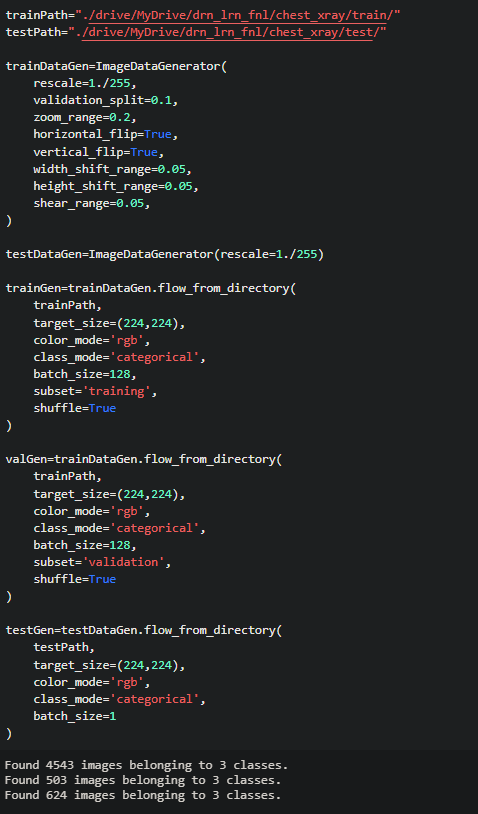
Yukarıda ki kod bloğu model.layer bastırılır çıktısını aşağıda göre bilirsiniz.



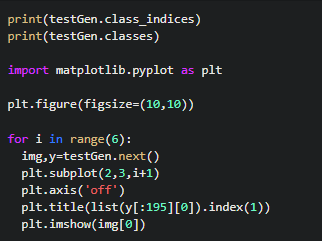
Aşağıdaki kod bloğunda drive ile bağlantımızı kurduk.

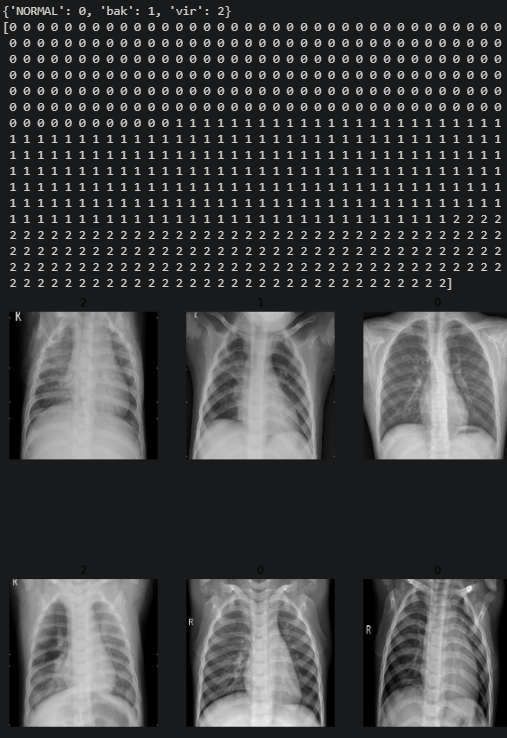


Ardından drive içerisinde bulunan dosyalarımızın bağlantısı için aşağıda görmüş olduğunuz ilk 2 satırı yazdım.

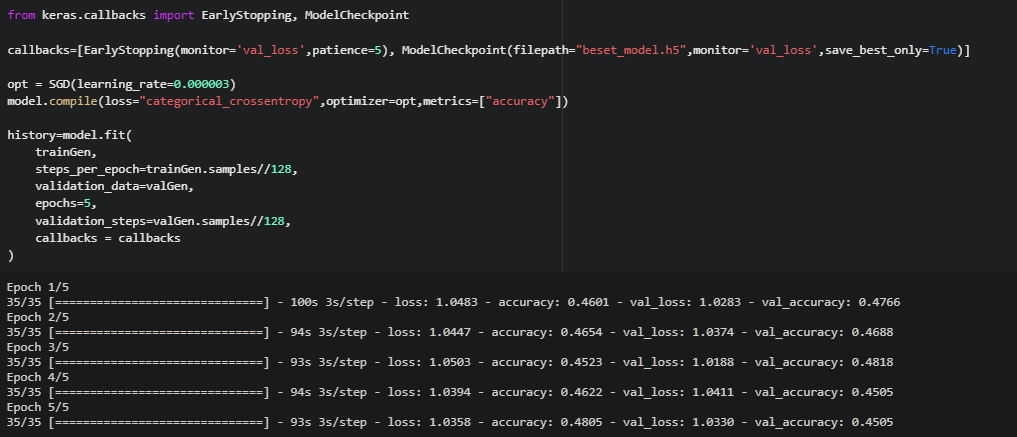


Diğer modellere göre shear range, height shift gibi değerleri ola bildiğince düşük tuttum.

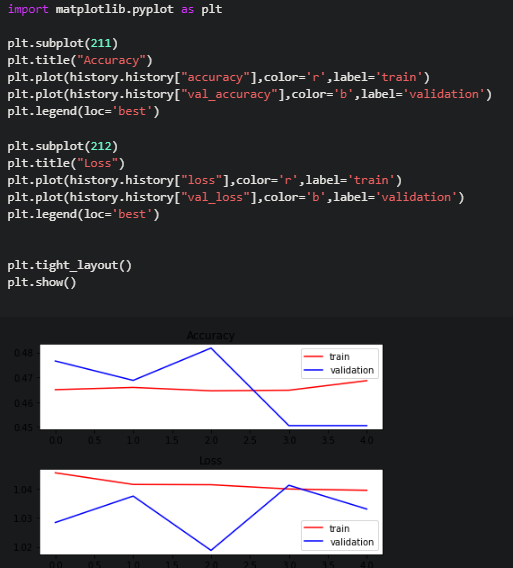


Yukarıda ki kod çalıştığın zaman aşağıdaki görmüş olduğunuz çıktıyı verir.

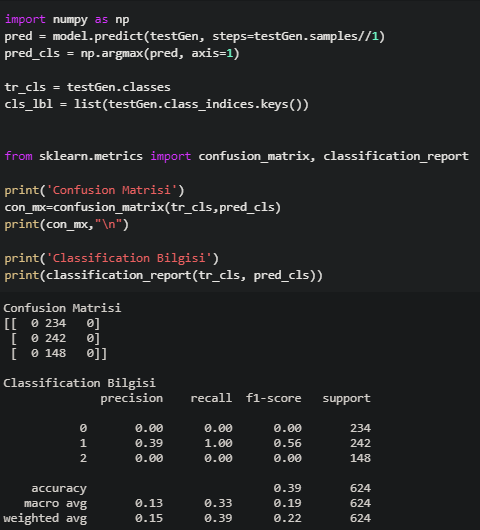
SGD optimizer için bazı değerleri belirtikten sonra devam ediyoruz.



Sonra elde edilen sonuçları görsel bir grafik haline getirmek için aşağıdaki kod bloğunu kullanırız.

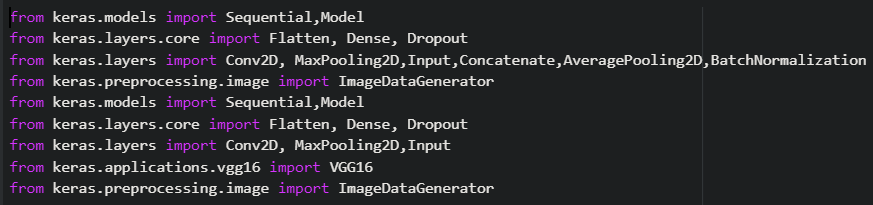


Bu işlem gerçekleştirildikten sonra ise test aşamasına geçebiliriz aşşağıdaki kod ile birlikte bunu sağlarız.

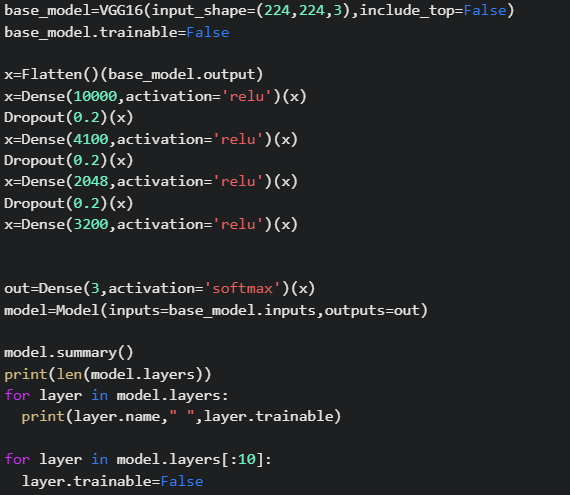


Model 10:

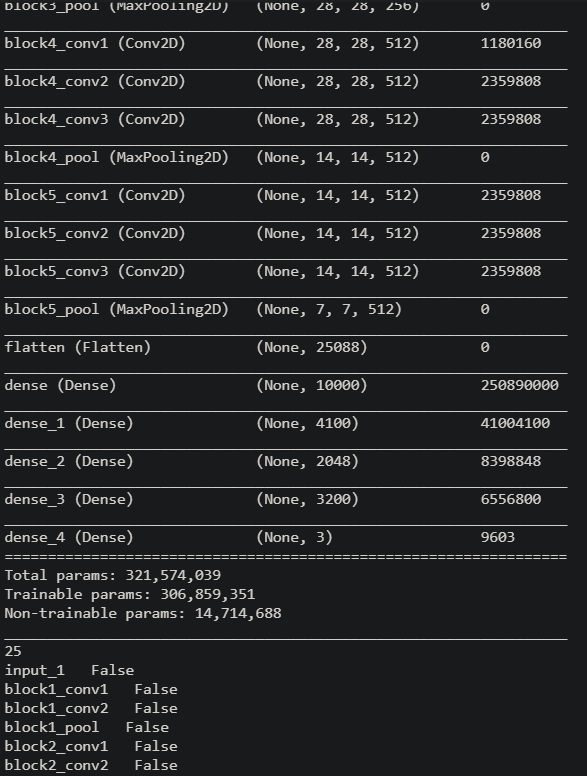
Kütüphaneleri aşağıdaki gibi ekliyoruz.

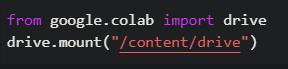


Aşağıda düğüm sayısını ve alacağı değerleri belirttikten sonra for ile printi kullanarak model.layersı bastırırız.



Çıktısı aşağıdakigibidir.

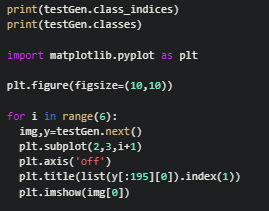




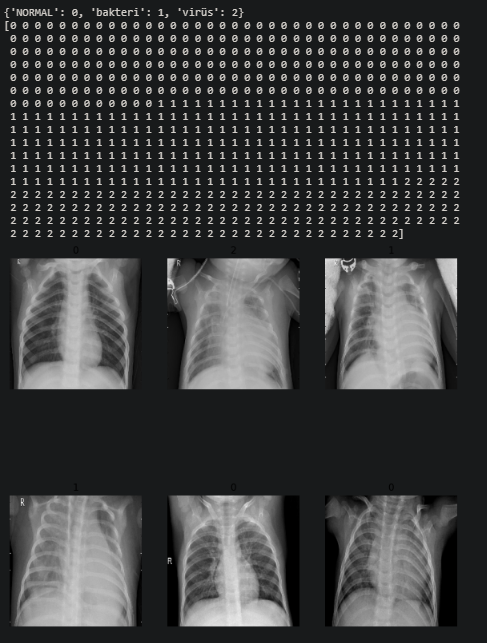
Yukarıdaki kod ile Google Drive bağlantısı sağnadı ardından aşağıdaki ilk iki satır ile dosya da ki konumları belirtildi. Aşağıda diğer modellerden farklı olarak horizonal flip, zoom range ve vertical flip yok.

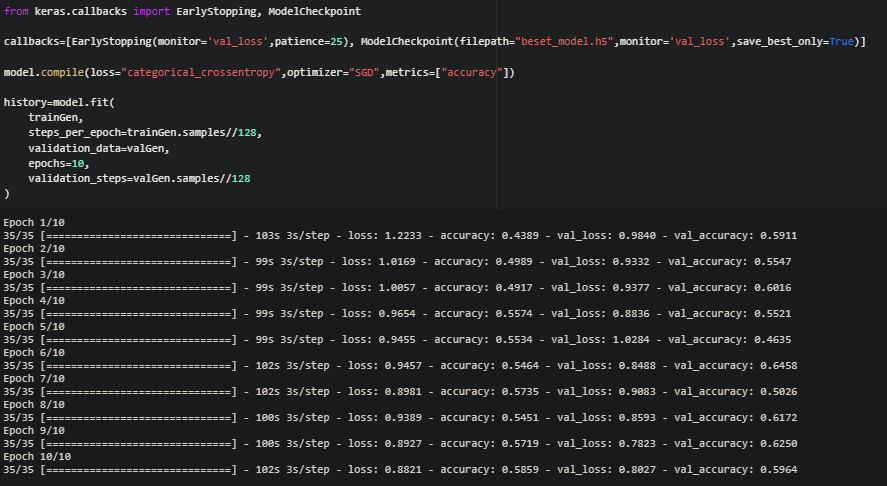


Aşşağıdaki kod bloğu çalıştırıldığında



Bu çıktı ile karşılaşacağız





Yukarıda early stop ve model check pointi ekledik bu sayede eğer model eğitiminde 5 epoch da ilerleme olmadığını fark eder ise epochu erken kesilecek ve alınan en iyi sonuç baz alınacak. Ardından optimizer SGD diyoruz. Ardından bir grafik oluşturma için aşağıdaki kodu çalıştırıyoruz.



İlgili grafik geldikden sonra confusion matrisini ve test sonucunu ekrana bastırıyoruz.

