KERAS İLE DERİN ÖĞRENME

1. Keras Nedir?

Keras dediğimiz yapı aslında Tensorflow, Theano, CNTK gibi yapıları daha kolay ve verimli kullanabilmek için geliştirilmiş bir derin öğrenme kütüphanesidir. Biz bu dokümanda Tensorflow üzerinden gideceğiz.

2. Model Nedir?

Model içinde katmanları depolayan bir katman grubudur. Keras içerisinde bir model sınıfı barındırır. Gelin model sınıfına bir göz atalım.

- ### Model Sınıfı

Model sınıfı katmanları içerisinde doğrusal olmayan bir şekilde barındırabilir. Doğrusal olmaması nasıl avantajlar sağlıyor? Doğrusallık durumu sıralı bir katman mekanizması gerektirir. Yani diyelim ki ben veriyi bir katmana verdim. Doğrusal olsaydı bu sadece sonraki katmana gidecekti. Ancak doğrusal olmadığı için ilk katmandan çıkan veriyi birden çok katmana gönderebilirim. Bu sayede birden çok farklı çıktı alabilirim. Gelin model sınıfını çağıralım ve görelim.

```
1  import tensorflow.keras as keras
2
3  yeni_model = keras.Model()
4
5  print(yeni_model)
6
```

```
Run:  1-keras_model
C:\Users\ghost\AppData\Local\Programs\Python\Python38\python.exe C:/Users/ghost
Using TensorFlow backend.
2022-03-08 11:27:51.033024: I tensorflow/core/platform/cpu_feature_guard.cc:151
To enable them in other operations, rebuild TensorFlow with the appropriate com
2022-03-08 11:27:51.438116: I tensorflow/core/common_runtime/gpu/gpu_device.cc:
<tensorflow.python.keras.engine.training.Model object at 0x000002159CE896D0>
```

Model sınıfımızı kullanarak yeni_model adında bir nesne oluşturduk. Peki, gelin bu nesnenin fonksiyonlarına bakalım.

- compile() Fonksiyonu

Bu fonksiyonumuzun görevi model eğitilirken kullanılacak ayarları belirlemektir. Ne demek istiyorum? Eğitilirken, hata hesaplayıcıları, metrik hesaplayıcıları, optimize edicileri bu fonksiyon ile belirleriz. Parametrelerine bakalım.

```
Model.compile(
    optimizer="rmsprop",
    loss=None,
    metrics=None,
    loss_weights=None,
    weighted_metrics=None,
    run_eagerly=None,
    steps_per_execution=None,
    jit_compile=None,
    **kwargs
)
```

En sık kullanılan parametreleri biraz tanıyalım.

optimizer Parametresi: Bu parametreye ait birçok değer mevcut diyebiliriz. Peki, bir optimizer ne işe yarar. Optimizer dediğimiz fonksiyonlar kayıpları azaltmak için ağırlıkları ve öğrenme oranı gibi değerleri değiştirmek için kullanılır. Bu da sonuçları daha hızlı bir şekilde almamızı sağlar. Eğitim sürecini hızlandırır diyebiliriz.

loss Parametresi: Bu parametre ile oluşan kayıplar hesaplanır. Oluşan kayıplar ise bize hata miktarını verir. Bu değer eğitim sırasında minimuma indirilmeye çalışılır.

metrics Parametresi: Bu parametreye birçok metrik değeri verilebilir. Ancak biz genel olarak accuracy yani doğruluk metriğini veririz. Bunun sebebi ise sonuç olarak eğitim doğruluğunu görebilmek. Bu metrik sayesinde modelin ne kadar doğru eğitildiğini görebiliriz. Yani demek istediğim aslında bu parametreye model eğitim sırasında neyi hesaplamak istiyorsak onu vermeliyiz. Ve biz de genellikle doğruluk oranını öğrenmek istediğimizden accuracy değerini veriyoruz.

- fit() Fonksiyonu

Bu fonksiyonumuz belirttiğimiz parametrelere göre modelimizin eğitime başlamasını sağlar. Aldığı parametrelere bakalım.

```
Model.fit(
    x=None,
    y=None,
    batch_size=None,
    epochs=1,
    verbose="auto",
    callbacks=None,
    validation_split=0.0,
    validation_data=None,
    shuffle=True,
    class_weight=None,
    sample_weight=None,
    initial_epoch=0,
    steps_per_epoch=None,
    validation_steps=None,
    validation_batch_size=None,
    validation_freq=1,
    max_queue_size=10,
    workers=1,
    use_multiprocessing=False,
)
```

Gördüğünüz üzere birçok parametresi var. Ancak ben bu dokümanda sık kullanacağımız parametreleri açıklayacağım.

x Parametresi: Bu parametre giriş verilerini ifade eder. Yani modelin tanımasını istediğimiz veriler. Bu giriş verileri birçok tipte olabilir. Örnek vermek gerekirse bu veri, numpy dizisi, Tensorflow tensorü, generator veya DatasetCreator yani veri seti oluşturucu olabilir.

Peki, biz hangisini kullanmalıyız? Burada böyle bir soruyu cevaplamak mümkün değil. Neyin üzerine çalışıyorsanız ona göre bir veri vermelisiniz. Örnek verirsek, OpenCV ile görüntü işlemelerde görüntü yani resim bir numpy dizisi olarak içe aktarılır. Biz de burada x parametresini, birden çok resmi içerisinde barındıran, numpy diziler topluluğunun oluşturduğu bir numpy dizisi olarak verebiliriz.

y Parametresi: Bu parametre ise hedef verileri ifade eder. Demek istediğim giriş verilerinin etiketlerini tutar. Eğer giriş verilerini bir oluşturucu yani generator ile

oluşturduysak bu parametreyi boş bırakmalıyız. Çünkü generator dediğimiz yapı verileri okurken bize hem etiketleri hem de resimleri belirten sözlükler döndürür.

Örnek verirsek diyelim ki bir araba resmimiz var ve biz bunu Keras'a araba şeklinde tanıtmalıyız. X parametresinde arabanın resmini veriyoruz. Y parametresinde ise bunun araba olduğunu belirten bir etiket veriyoruz.

batch_size Parametresi: Bu parametrenin amacı tek bir döngüde kullanılacak veri miktarını belirlemektir. Açıklamasını şöyle bir örnekle anlatayım. Diyelim ki 1024 adet resmim var ve ben bu resimleri 32'şerli parçalara ayırarak sürekli oluşturduğum ağırlıklarla çarpıyorum. Eğer bunların hepsini yani 1024 resmi birden elimdeki ağırlıklarla çarpmaya çalışsaydım mevcut donanımım bu hesaplamaları kaldıramazdı. Bu sebeple parametremi 32 yaptım. Bu durumda, bir turda 32 resmimi ağırlıklarla hesapladı ardından sonraki 32'yi ve böyle tüm resimlerimi ağırlıklarla çarparak hesaplamaları bitirdim. Bu sayede de bilgisayarımı büyük bir yükten kurtarmış oldum. Bu değeri ne kadar küçültürseniz model eğitim sürenizde normal olarak uzayacaktır. Sık kullanılan değerleri 32, 64 ve 128 değerleridir. Eğer bu parametreye bir değer vermezseniz varsayılan olarak 32 değerini alacaktır.

epoch Parametresi: Epoch parametresi back propagation için büyük önem taşır. Peki, back propagation nedir? Back propagation durumu hata miktarını en aza indirmek için ağırlıkların sürekli düzenlenmesidir. Bir resmi en başta küçük ağırlıklarla çarparak derin öğrenme algoritmamız bir sonuca ulaşır. Maliyet hesaplama algoritmalarının sonucunda eğer ortada verimli bir sonuç yoksa yani sonucun hata değeri yüksekse back propagation devreye girer. Burada ise tüm katmanlar arası ağırlıklar yeniden ayarlanır ve resimler yeniden ağırlıklarla çarpılarak yeni hata oranları bulunur. Epoch ise tam olarak burada devreye giriyor. Bu back propagation dediğimiz olayı kaç defa uygulayacağını bu parametre ile belirtiriz. Eğer bu parametreye 32 değerini girersem 32 defa yeniden ağırlık ayarlayarak hatayı düşürmeye çalışacaktır.

verbose Parametresi: Verbose bize süreç içerisindeki ayrıntıyı verir. 3 değer alabilir. Eğer bu parametreye 0 değerini verirsem bana hiçbir ayrıntı vermez tüm işlemleri arka planda yapar bende boş ekranı izlerim. Eğer 1 verirsem genel sürecin yüzde kaçının bittiğine dair bana bir progress bar üzerinden bilgi döndürür. Yani, bilgisayara bir program yüklenirken ilerleme barının hareketiyle aynı. 2 değerini verirsem bana her epoch sürecinde kaç resmi işlediğine dair ve o epoch sürecinin hangi aşamasında olduğunu ekranda verir. 2 değeri maksimum ayrıntıyı verir.

validation_split Parametresi: Bu parametre eğitim verilerinin ne kadar kısmının doğrulama için kullanılacağını belirtir. 0 ile 1 arasında değer alan bu parametre eğitim verisinin verilen değer kadarlık kısmı alır ve o kısmı eğitim amaçlı kullanmaz. O kısım sadece eğitim sonucunda doğrulama verisi elde etmek için kullanılır. Doğrulama ise hata miktarını ortaya koyar. Bu veriler her bir epoch sonucunda denenir. Hata miktarı

ve metrikler hesaplanır buna göre back propagation durumu belirlenir. Ve hata miktarına göre back propagation ile ağırlıklar yeni değerini alır.

validation_data Parametresi: Bu parametre yukarıdakine benzerdir. Ortada tek bir fark mevcut o da elimizdeki eğitim verilerinden kullanmayız. Diyelim ki biz validation_split parametresini kullanmadık yani en başta kaç resmimiz varsa o kadarını eğitim için kullandık. Bunun haricinde bir doğrulama için veri seti oluşturduk. Oluşturduğumuz veri seti yine resimlerimiz ve resimlerimizi labelları olmak üzere iki listeden oluşmalı. Bu iki listeyi tuple şeklinde bu parametreye verdiğimizde bu veriler üzerinden hata miktarlarını ve metrikleri hesaplayacaktır. Yine bu değerler sonucunda back propagation için değerler belirlenecektir.

shuffle Parametresi: Shuffle parametremiz her epoch sonunda verilerin karıştırılıp karıştırılmayacağını belirler. Bu logic bir parametredir. Alabileceği değerler True veya False değerleridir. Normalde model eğitilirken veriler aynı sırayla eğitime girerler ancak eğer shuffle parametremiz True değerindeyse her epoch sonunda verilerin sıraları değiştirilir. Bu durum derin öğrenme algoritmamızdaki ezberleme durumunu bir nebze ortadan kaldırır.

- summary() Fonksiyonu

  Bu fonksiyon bize oluşturduğumuz ağın bir özetini çıkartır. Tabi bir özet alabilmek için bu ağın belirli katmanlar içermesi gerekir. Bu fonksiyon sonucunda verinin hangi katmana gireceği, hangi katmanlarda işlem göreceği ve hangi katmanlardan çıkacağı ekrana yazdırılır. Aynı zamanda verinin boyutunun nasıl değişeceği gibi belirli bilgileri de ekrana yazdırır.

• Sequential Sınıfı

Sequential Model yapısı Keras üzerinde bulunan bir model türüdür. Bu model sayesinde biz doğrusal bir katman yığını oluşturabiliriz. Yani Sequential Model, katmanları içinde depolayarak birçok katmandan oluşan doğrusal katman yığınıdır. Doğrusal olduğu için veri bir katmana girer ve bu katmandan çıktıktan sonra sadece diğer katmana girebilir. Bir katmandan çıkan veri birden çok katmana gidemez. Bu da modelin eğitilmesi sonucunda bize tek bir çıkış verir.

- Add() Fonksiyonu

  Bu fonksiyon oluşturduğumuz modele bir katman eklememizi sağlar. Keras içerisinde birçok katman barındırır. Biz de bu katmanları modelimize eklemek için bu fonksiyonu kullanacağız.

- Pop() Fonksiyonu

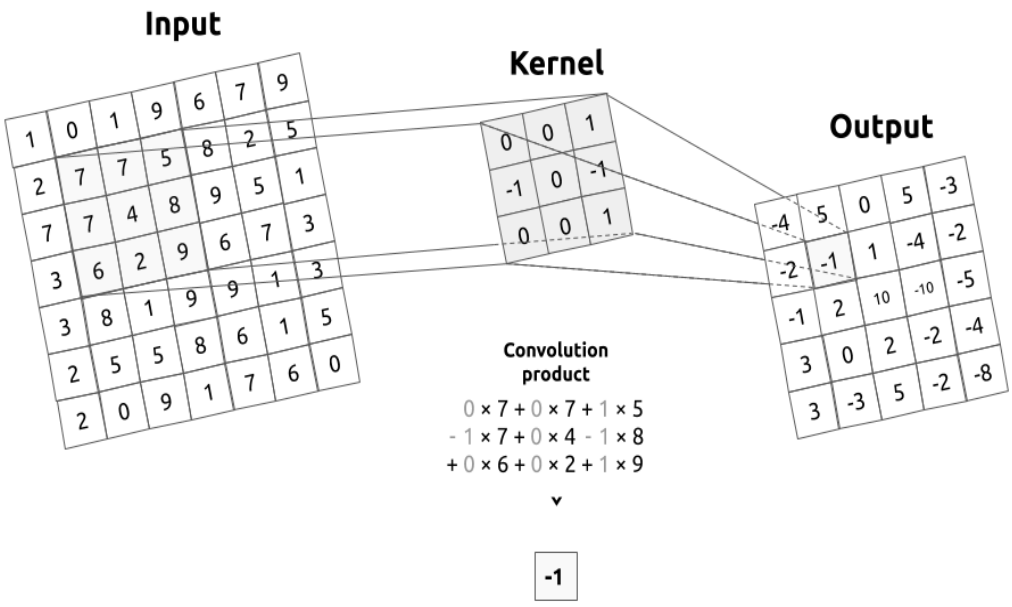
  Bu fonksiyon ise modelimize son eklediğimiz katmanı silmek için kullanılır. Python'da listelerdeki pop fonksiyonuyla aynı işi yapıyor diyebiliriz.

3. Katman Nedir?

Katmanlar Keras'ın temel yapı taşlarıdır. Katmanların kendince belirli görevleri vardır. Bu dokümanda Dense, MaxPooling2D, Conv2D, Flatten ve Dropout katmanlarına değineceğiz. Şimdi gelin Conv2D katmanından başlayalım.

- Conv2D Katmanı

Conv2D katmanı aslında çok sık kullanılan bir katmandır. Görevi ise 2 boyutlu dizilerde evrişim filtresi olarak işlem yapmaktır. Peki, Evrişim filtresi nedir?

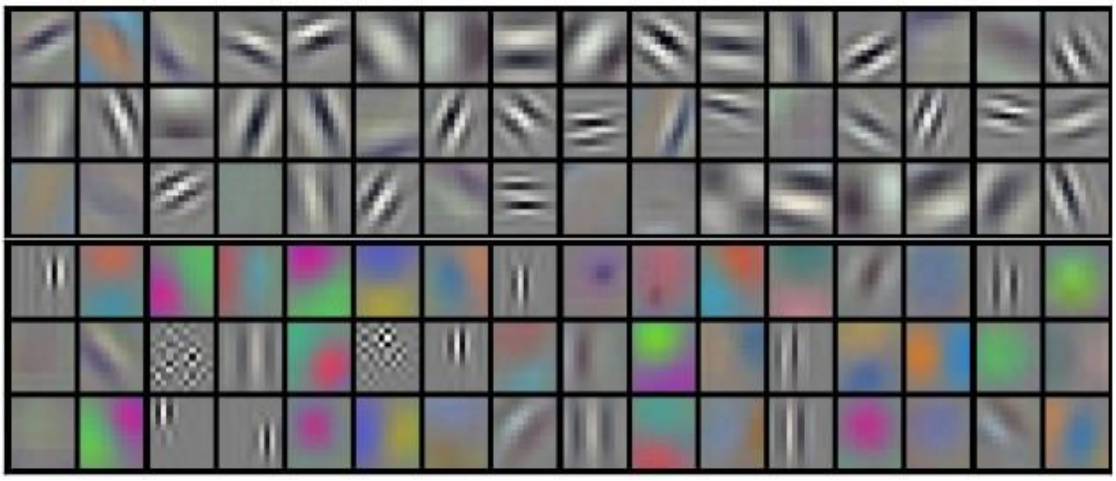


Evrişim filtresi bir sequential modelde ilk katman olarak kullanılır. Amacı içerisinde belirli ağırlıklar belirleyerek resim veya farklı bir iki boyutlu dizide özellikleri ortaya çıkartmaktır. Şimdi parametrelere bakarak daha iyi anlamaya çalışalım.

filters Parametresi: Bu parametrenin görevi ağırlık sayısını belirlemektir. Verdiğimiz sayı miktarı kadar bizim için ağırlıklar oluşturacaktır. Bu değer tam sayı olmalıdır.

kernel_size Parametresi: Bu parametre dolaşacak evrişim filtresinin matris boyutunu belirtir. İki boyutta çalıştığımız için bu iki elemanlı bir tuple olmalıdır. Örnek

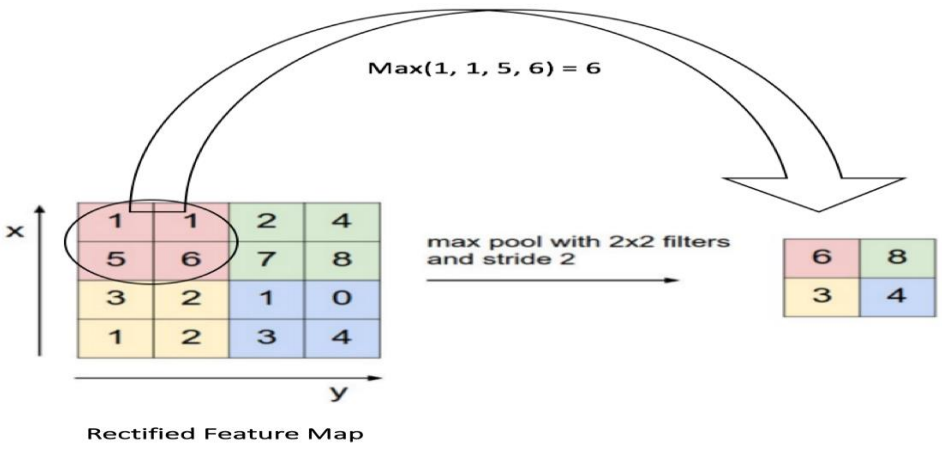
veriyorum 128x128 boyutunda bir resimde evrişim filtremizi uygulayacağız. Yukarıdaki resimde verdiğim örnek şeklinde bu 128x128'lik resim üzerinde dolaşacak 3x3 bir matris oluşturuyoruz. Bu matrisin içerisine de sırasıyla ağırlıklarımız yerleşiyor. Ardından sol üstten dolaştırmaya başladığımız zaman birbirine karşılık gelen değerler birbiriyle çarpılarak toplanıyor. Sonuç olarak çıkan değer yeni oluşturulmuş bir matrisin tek bir karesine yani pikseline yerleşiyor. Bu sayede biz o bölgeden bir özelliği çıkartmış oluyoruz. Demek istediğimi daha iyi anlatmak adına aşağıdaki görseli bırakıyorum.



input_shape Parametresi: Bu parametremiz ise giriş olarak verilecek verinin boyutunu katmana belirtmek için kullandığımız parametremizdir. Örneğin 128x128'lik bir resmi rgb olarak vermek istersem katmana bu parametrenin değeri (128, 128, 3) şeklinde olacaktır. İlk iki değer boyut son değer ise renk skalasını temsil eder.

- MaxPooling2D Katmanı

Bu katmanın görevi belirtilen değerdeki matrisi resim üzerinde dolaştırarak maksimum değeri tek bir piksel olarak belirlemesidir. Aşağıdaki resimle aslında her şeyi anlayabilirsiniz.



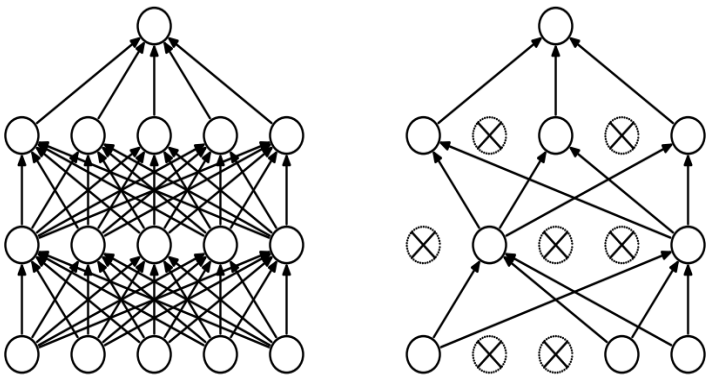
Sık kullanılan parametrelere göz atalım.

pool_size Parametresi: Bu parametremiz oluşacak maxpooling matrisinin boyutunu belirlememizi sağlar. Bu parametre bir tuple değer alır. Örnek olarak (2, 2) lik bir boyut.

strides Parametresi: Bu parametre ise matrisin kaçlık adımlarla atlayacağını belirtir.

- Dropout Katmanı

Bu katman oluşan nöronların bazılarını kullanılmaz duruma getirir. Önceki katmanlardan çıkan yeni değerler nöronlara atanır. Ancak şöyle bir durum söz konusudur. Eğer biz bu katmanı kullanmazsak ortaya bir ezber durumu çıkabilir. Amacımız makinenin öğrenmesi olduğu için bu durumu istemiyoruz. Bu parametre ile bazı nöronları kapatıyoruz ve sonuç olarak ortaya bir ezber durumu çıkmıyor. Aşağıdaki resmi incelerseniz durumu daha iyi bir şekilde anlayabilirsiniz.



(a) Standard Neural Net (b) After applying dropout.

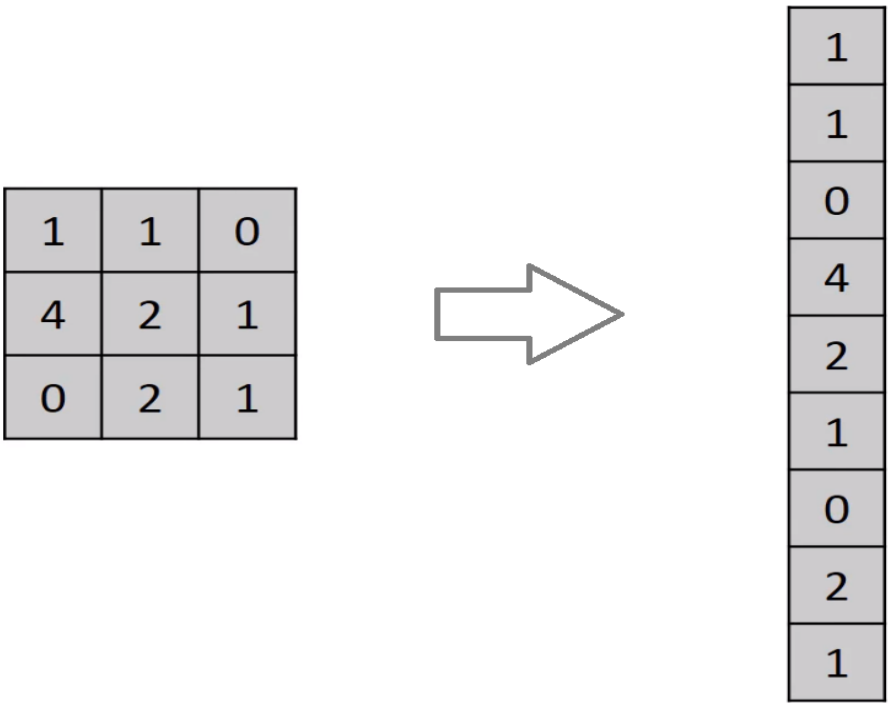
Figure 1: Dropout Neural Net Model. **Left**: A standard neural net with 2 hidden layers. **Right**: An example of a thinned net produced by applying dropout to the network on the left. Crossed units have been dropped.

Yukarıdaki resme baktığımızda hidden layer dediğimiz gizli katmanlardaki bazı nöronların kapandığını görebiliriz. Gelin şimdi parametresine bakalım.

rate Parametresi: Bu parametre 0 ile 1 arasında float değerleri alır. Bunu yüzdelik dilim olarak düşünürsek kapatılacak nöron sayısı / mevcut nöron sayısı diyebiliriz.

- ## Flatten Katmanı

Bu katman artık sonlara yaklaştığımızı ifade ediyor desek yanlış olmaz. Görevi bu zamana kadar oluşturduğumuz birçok 2 boyutlu matrisi tek boyuta düşürmek. Yani bir zincir haline getirmek. Lütfen aşağıdaki resmi inceleyiniz.



Bu katmanı çağırmamız yeterlidir. Herhangi bir parametre şart değildir.

- ## Dense Katmanı

Son olarak Dense katmanına gelirsek. Bu katman elimizdeki nöron sayısını belirttiğimiz nöron sayısına eşitlemek için nöronları birbirine bağlar. Parametrelere bakalım.

units Parametresi: Bu parametre nöron sayısını belirtir. Örneğin 700 adet nöron varsa 64 değerini verdiğimizde nöronları birbirine bağlar ve nöron sayısı 64'e düşer.

activation Parametresi: Bu parametre ise Conv2D katmanındaki activation parametresi ile aynı işlevi görür. Değerleri 0 ile 1 arasına sıkıştırmak için bir aktivasyon fonksiyonu belirtilir. Ve bu fonksiyon değerleri buna göre ayarlar. Eğer bu son katman değilse genellikle relu kullanılır eğer son katmansa alınacak sonuca softmax veya sigmoid fonksiyonları kullanılabilir.