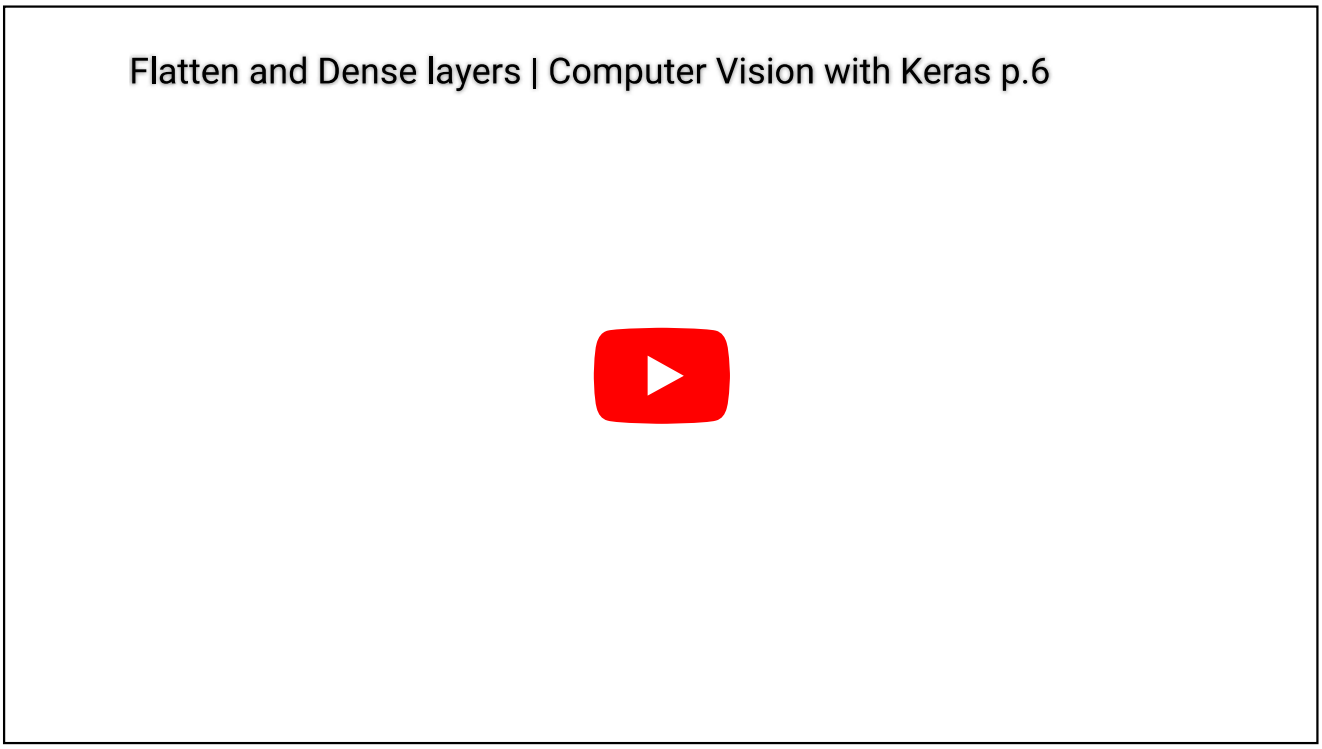
Flatten And Dense Layers | Computer Vision With Keras P.6

Click here to download the Source code



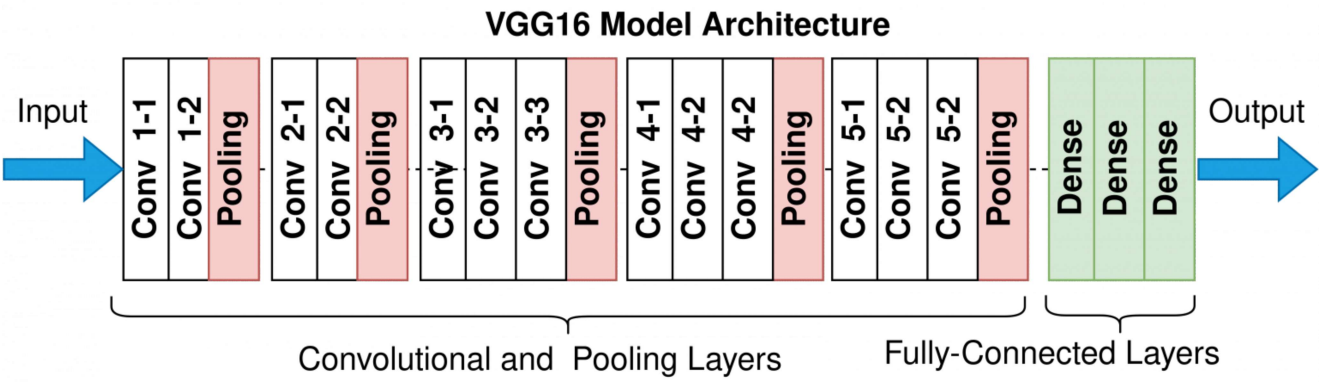
We will see how to apply flatting operation and dense layer into a convolutional neural network with Keras

Flatten And Dense Layers In A Simple VGG16 Architetture

To better understand the concept and purpose of using Flatten and Dense layers let's see this simple architecture of the VGG16 model as an example.

There are several convolutional groups that end with a pooling layer. At the end of these

There are several convolutional groups that end with a pooling layer. At the end of these elaborations, there is the Dense layer. These processes aim to have numerous image data at the input and reduce it to minimal information. If you want to know more about this aspect, I recommend that you see the research about convolutional neural network



Wikipedia

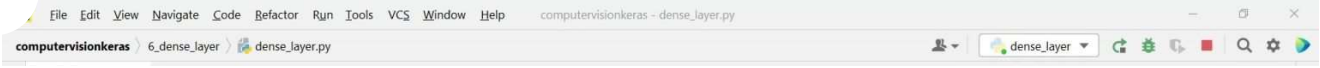
In practice, we have to transform the image into very small numbers. For example, if we wanted to identify a dog we could say that the number close to zero identifies the dog and the number nearest to 1 identifies the cat.

How To Use Flatten And Dense Layers

As we saw in the previous lesson **Max pooling layer | Computer Vision with Keras p.5** we must always prepare the image and apply the Max pooling layer in addition to the Conv2D layer so for this part the code always remains the same.

```
6.  img_size = 32
7.  img = cv2.imread("dog.jpg")
8.  img = cv2.resize(img, (img_size, img_size))
9.
10. # Model
11. model = keras.Sequential()
12. model.add(layers.Conv2D(input_shape=(img_size, img_size, 3), filters=64,
    kernel_size=(3, 3)))
13. model.add(layers.MaxPooling2D(pool_size=(2, 2), strides=(1, 1)))
```

Trying to show the result of these elaborations we obtain an array with various numbers



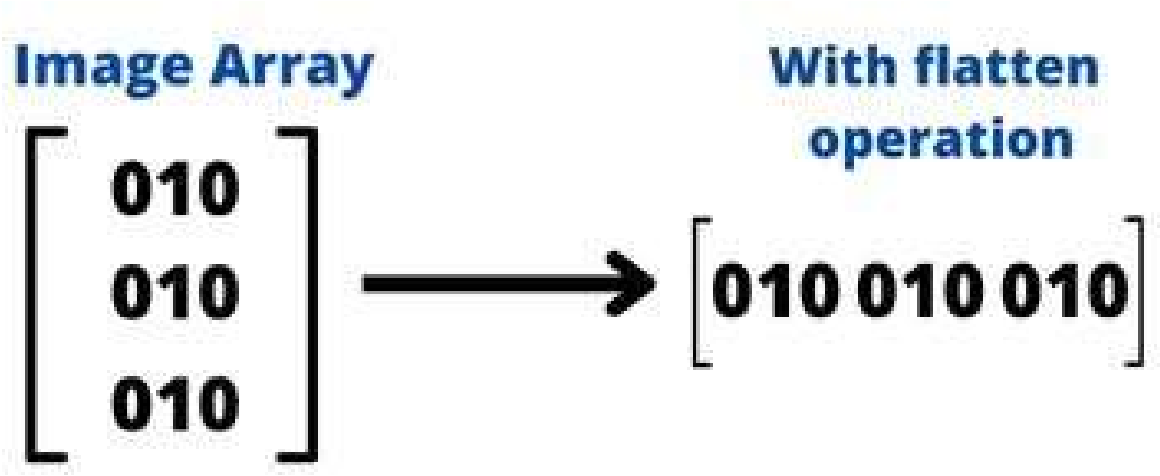
```
model.add(layers.Conv2D(input_shape=(img_size, img_size, 3), filters=64, kernel_size=(3, 3)))
model.add(layers.MaxPooling2D(pool_size=(2, 2), strides=(1, 1)))

Run: dense_layer
E:\DeepLearning\PythonVENVS\computervisionkeras\Scripts\python.exe E:/PythonProjects/computervisionkeras/6_dense_lay
2022-08-11 16:42:19.448926: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow binary is optimized
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
2022-08-11 16:42:20.201094: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1532] Created device /job:localhost/r
2022-08-11 16:42:22.962755: I tensorflow/stream_executor/cuda/cuda_dnn.cc:384] Loaded cuDNN version 8204
1/1 [==============================] - 6s 6s/step
[[[[  -7.0900993   52.753048    -2.1591403 ...   10.845272
      -3.1192226    0.7399052]
   [  -6.295644    53.3319      -3.8455932 ...   10.845272
      -3.8606682    1.8363645]
   [  -6.295644    53.3319      -4.7259927 ...   10.54854
      -3.524365     1.8363645]
   ...
   [ -64.993576    64.82611    -30.864532  ...   35.397926
     -29.87637     13.411712 ]
```

all of this can be simplified with Flatten and Dense layers.

Apply Flatten Layer

By applying the flatten operation we can have a simplification of the array. In fact, we will get an array that contains the values one after the other.

To better explain the concept, see the diagram below. On the left, you can see the array of an image that contains only the black and white color, on the right, the flatten operation is applied and the array contains the values one behind the other.

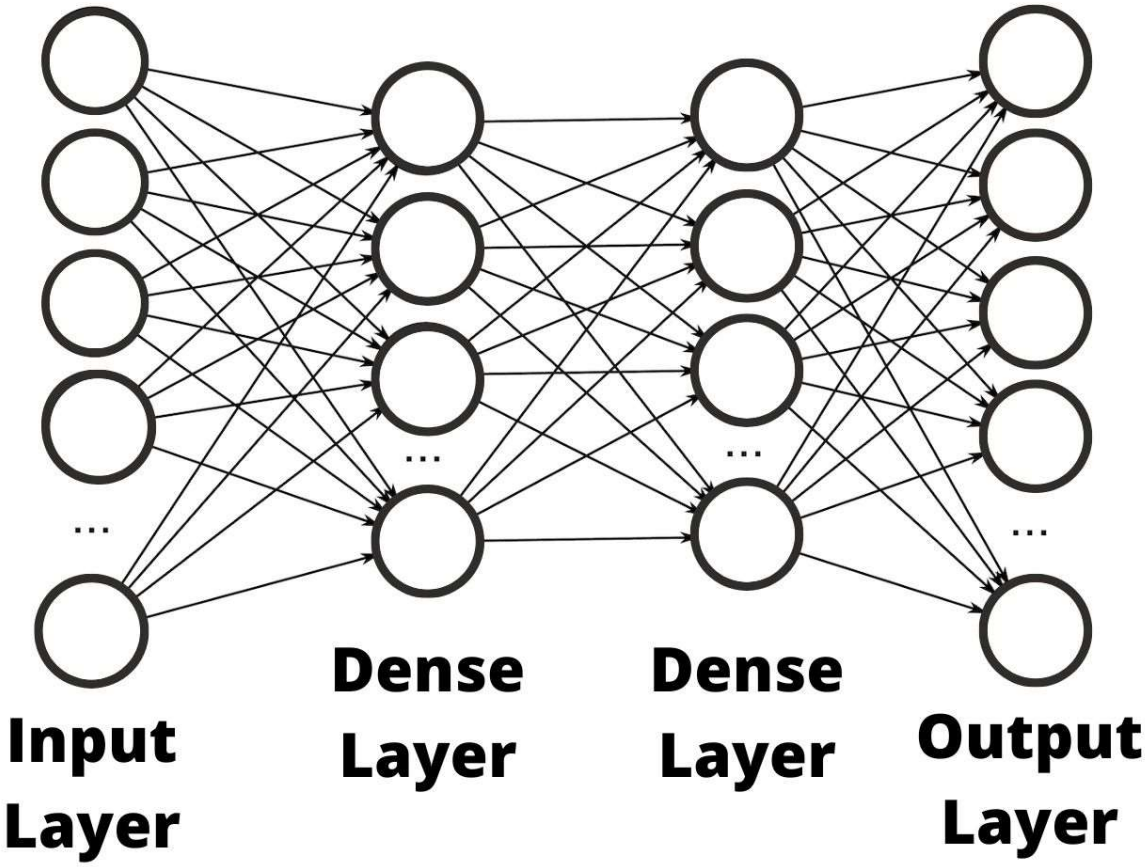


Its application is also very simple

```
14.  model.add(layers.Flatten())
```

Apply Dense Layers

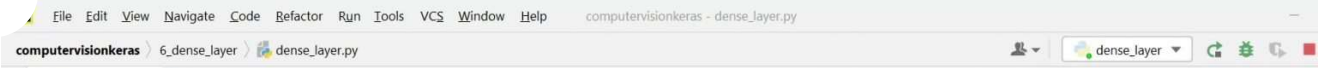
The dense layer is perhaps the best-known part of the convolutional neural network and the image below represents this passage well. Their job is to process all the information and return only a few values to determine only if the object is present or not in the image.

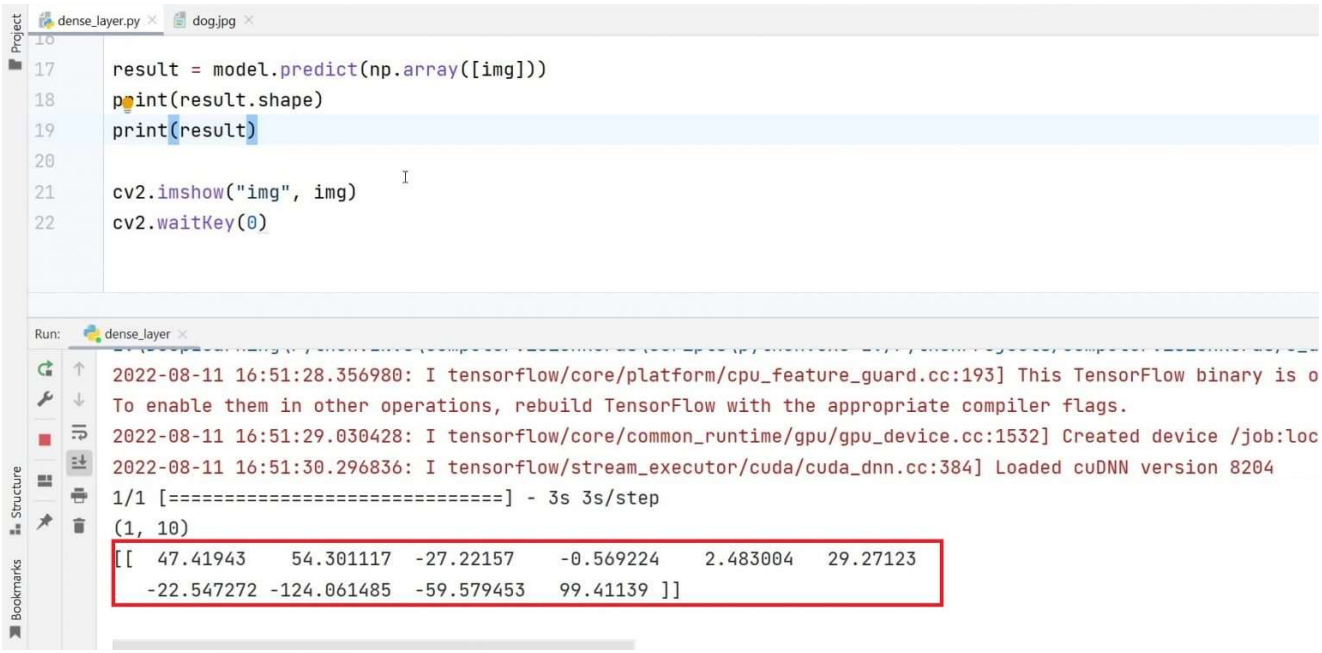


In our example, we set units=10 in order to obtain 10 output values. The general purpose is to go from the image array to get very few values just to know if the image is present or not.

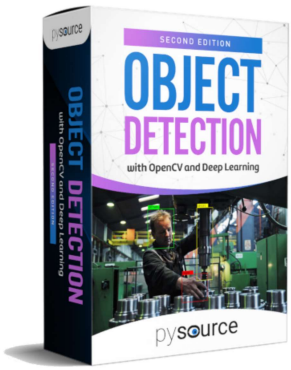
```
15.  model.add(layers.Dense(units=10))
```

This is the output we get with our example.





Click here to download the Source code



- Real-Time Traffic Analysis
- Custom AI Solutions

Office:

Nikolay Kopernik 22,

1111, Sofia, Bulgaria

VAT: BG205838657