What is the role of “Flatten” in Keras?

Asked 2 years, 9 months ago Active 29 days ago Viewed 67k times

89

34

I am trying to understand the role of the `Flatten` function in Keras. Below is my code, which is a simple two-layer network. It takes in 2-dimensional data of shape (3, 2), and outputs 1-dimensional data of shape (1, 4):

```
model = Sequential()
model.add(Dense(16, input_shape=(3, 2)))
model.add(Activation('relu'))
model.add(Flatten())
model.add(Dense(4))
model.compile(loss='mean_squared_error', optimizer='SGD')

x = np.array([[[1, 2], [3, 4], [5, 6]]])

y = model.predict(x)

print y.shape
```

This prints out that `y` has shape (1, 4). However, if I remove the `Flatten` line, then it prints out that `y` has shape (1, 3, 4).

I don't understand this. From my understanding of neural networks, the `model.add(Dense(16, input_shape=(3, 2)))` function is creating a hidden fully-connected layer, with 16 nodes. Each of these nodes is connected to each of the 3x2 input elements. Therefore, the 16 nodes at the output of this first layer are already "flat". So, the output shape of the first layer should be (1, 16). Then, the second layer takes this as an input, and outputs data of shape (1, 4).

So if the output of the first layer is already "flat" and of shape (1, 16), why do I need to further flatten it?

machine-learning tensorflow neural-network deep-learning keras

edited Oct 15 '19 at 16:54
nbro
9,134 13 67 114

asked Apr 5 '17 at 16:48
Karnivaurus
14k 34 99 180

3 Answers

103

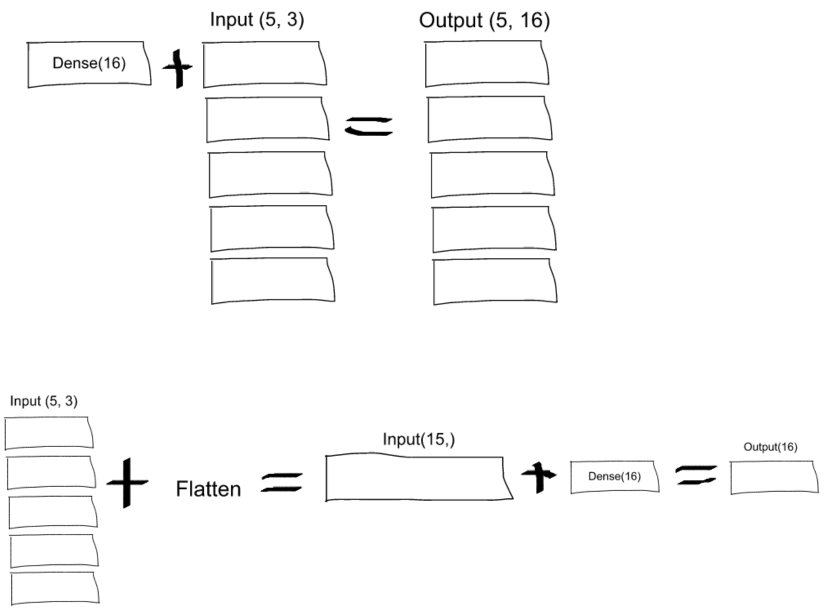
If you read the Keras documentation entry for `Dense` , you will see that this call:

```
Dense(16, input_shape=(5,3))
```

would result in a `Dense` network with 3 inputs and 16 outputs which would be applied independently for each of 5 steps. So, if `D(x)` transforms 3 dimensional vector to 16-d vector, what you'll get as output from your layer would be a sequence of vectors: `[D(x[0,:]), D(x[1,:]),..., D(x[4,:])]` with shape `(5, 16)` . In order to have the behavior you specify you may first `Flatten` your input to a 15-d vector and then apply `Dense` :

```
model = Sequential()
model.add(Flatten(input_shape=(3, 2)))
model.add(Dense(16))
model.add(Activation('relu'))
model.add(Dense(4))
model.compile(loss='mean_squared_error', optimizer='SGD')
```

EDIT: As some people struggled to understand - here you have an explaining image:



edited Dec 22 '19 at 3:34
charlesreid1
2,540 1 20 34

answered Apr 5 '17 at 17:20
Marcin Możejko
28.4k 8 77 95

Thanks for your explanation. Just to clarify though: with `Dense(16, input_shape=(5,3)` , will each output neuron from the set of 16 (and, for all 5 sets of these neurons), be connected to all (3 x 5 = 15) input neurons? Or will each neuron in the first set of 16 only be connected to the 3 neurons in the first set of 5 input neurons, and then each neuron in the second set of 16 is only connected to the 3 neurons in the second set of 5 input neurons, etc.... I'm confused as to which it is! – Karnivaurus Apr 6 '17 at 12:49

1 Ah ok. What I am trying to do is take a list of 5 colour pixels as input, and I want them to pass through a fully-connected layer. So `input_shape=(5,3)` means that there are 5 pixels, and each pixel has three channels (R,G,B). But according to what you are saying, each channel would be processed individually, whereas I want all three channels to be processed by all neurons in the first layer. So would applying the `Flatten` layer immediately at the start give me what I want? – Karnivaurus Apr 6 '17 at 13:08

6 A little drawing with and without `Flatten` may help to understand. – Xvolks Aug 28 '17 at 14:52

2 Ok, Guys - I provided you an image. Now you could delete your downvotes. – Marcin Możejko Sep 13 '17 at 21:34

26

short read:

> Flattening a tensor means to remove all of the dimensions except for one. This is exactly what the Flatten layer do.

long read:

If we take the original model (with the Flatten layer) created in consideration we can get the following model summary:

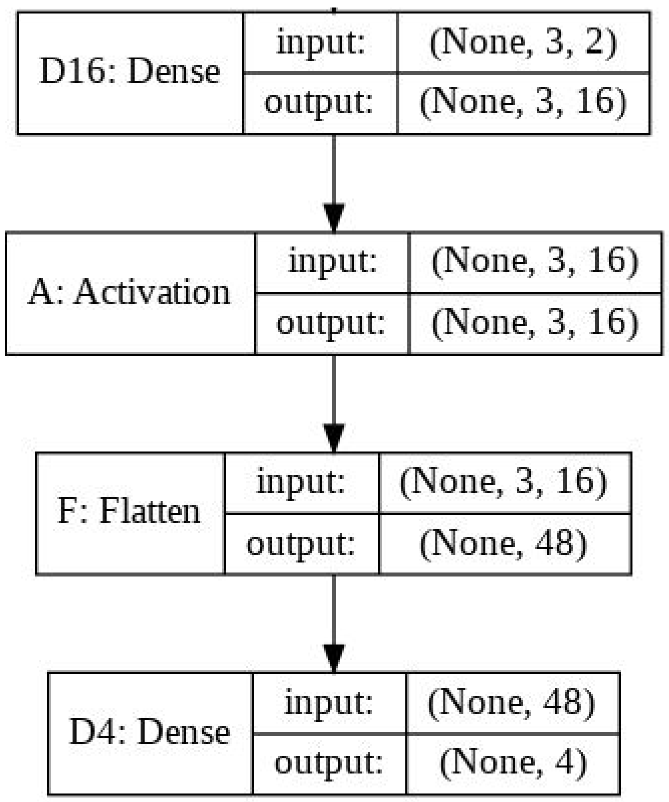
```
Layer (type)                 Output Shape              Param #
=================================================================
D16 (Dense)                  (None, 3, 16)             48
_________________________________________________________________
A (Activation)               (None, 3, 16)             0
_________________________________________________________________
F (Flatten)                  (None, 48)                0
_________________________________________________________________
D4 (Dense)                   (None, 4)                 196
=================================================================
Total params: 244
Trainable params: 244
Non-trainable params: 0
```

For this summary the next image will hopefully provide little more sense on the input and output sizes for each layer.

The output shape for the Flatten layer as you can read is `(None, 48)`. Here is the tip. You should read it `(1, 48)` or `(2, 48)` or ... or `(16, 48)` ... or `(32, 48)`, ...

In fact, `None` on that position means any batch size. For the inputs to recall, the first dimension means the batch size and the second means the number of input features.

A flatten operation on a tensor reshapes the tensor to have the shape that is equal to the number of elements contained in tensor **non including the batch dimension**.



Note: I used the `model.summary()` method to provide the output shape and parameter details.

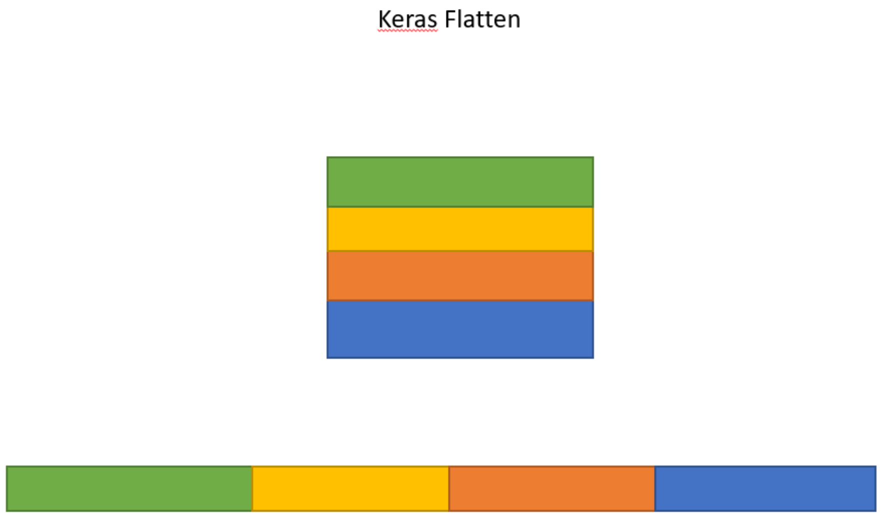
edited Jun 26 '19 at 8:40

answered Jan 14 '19 at 23:25
prosti
14k 3 55 59

1 Very insightful diagram. – Shrey Joshi Jul 9 '19 at 13:44

1 Thanks for the diagram. It gives me a clear picture. – Sagor Jan 11 at 16:41

19



This is how Flatten works converting Matrix to single array.

answered Jul 19 '19 at 11:54

Mahesh Kembhavi

209 1 3

2 This guy needs to make more pictures. I like this. It makes sense. – alofgran Nov 21 '19 at 6:28

2 Yes, but *why* is it needed, this is the actual question I think. – Helen - down with PCorrectness Dec 1 '19 at 12:00