Linear layer input neurons number calculation after conv2d

andreiliphd Andrei Li **Nov '18**

I will be really thankful to those who can explain me this. I know the formula for calculation but after some iterations, I haven't got to the answer yet.

The formula for output neuron:
Output = ((I-K+2P)/S + 1), where
I - a size of input neuron,
K - kernel size,
P - padding,
S - stride.

Input tensor shape:

```
torch.Size([36, 200, 150, 3])
```

I have the following model:

```
class Flatten(torch.nn.Module):
    def forward(self, x):
        return x.view(x.size()[0], -1)

model = torch.nn.Sequential(
        torch.nn.Conv2d(32, 64, kernel_size=(3, 3)),
        torch.nn.ReLU(),
        torch.nn.Conv2d(64, 128, kernel_size=(3, 3)),
        torch.nn.ReLU(),
        torch.nn.MaxPool2d(kernel_size=(2, 2)),
        torch.nn.Dropout(0.25),
        Flatten(),
        torch.nn.Linear(128, 128),
        torch.nn.ReLU(),
        torch.nn.Linear(128, 2),
        torch.nn.Softmax()
        )
```

I can't calculate the number of neurons in Linear layer.
Could you kindly help and explain me the calculations of Linear input neuron size behind this network?

> Solved by **ptrblck** in **post #2**
>
> Your input shape seems to be a bit wrong, as it looks like the channels are in the last dimension. In PyTorch, image data is expected to have the shape [batch_size, channel, height, width]. Based on

Nov 2018

1 / 6
Nov 2018

5d ago

> your shape, I guess 36 is the batch_size, while 3 seems to be the number channels. However, as you…

Input size of fc layer in tutorial?

ptrblck **Nov '18**

Your input shape seems to be a bit wrong, as it looks like the channels are in the last dimension.
In PyTorch, image data is expected to have the shape `[batch_size, channel, height, width]`.
Based on your shape, I guess 36 is the `batch_size`, while 3 seems to be the number channels.

However, as your model expects 32 input channels, your input won't work at all currently.

Let's just assume we are using an input of `[1, 32, 200, 150]` and walk through the model and the shapes.
Since your `nn.Conv2d` layers don't use padding and a default stride of 1, your activation will lose one pixel in both spatial dimensions.
After the first conv layer your activation will be `[1, 64, 198, 148]`, after the second `[1, 128, 196, 146]`.
`nnMaxPool2d(2)` will halve the activation to `[1, 128, 98, 73]`.

If you set the number of `in_features` for the first linear layer to `128*98*73` your model will work for my input.

I also recommend to just print out the shape of your activation before the linear layer, if the shape calculation is too cumbersome, and set the input features according to this.
For your Sequential model you can just create a print layer with:

```
class Print(nn.Module):
    def forward(self, x):
        print(x.size())
        return x
```

andreiliphd Andrei Li **Nov '18**

Thank you very much **ptrblck**. It was really helpful.
It seems that I manage to make a function out off it for those who struggle with the issue.

```
def count_input_neuron(model, image_dim):
    return model(torch.rand(1, *(image_dim))).data.view(1, -1).size(1)
```

> ptrblck:
>
> However, as your model expects 32 input channels, your input won't work at all currently.

So, I have to change it to 3? The number of features or number of color channels?

ptrblck Nov '18

The `in_channels` of the first conv layer correspond to the channels of your input.
In case you are using a color image tensor, i.e. 3 channels, you would have to set `in_channels=3`.

Captain_90_s Captain 90's Feb 3

First of all there is a problem with your input shape. The shape should be BATCH_SIZE * CHANNEL * HEIGHT *WIDTH. So lets correct your size and I assume you BATCH_SIZE = 36, CHANNEL = 3, HEIGHT = 200 , WIDTH = 150.

```
images = image.permute(0,3,1,2)
```

Next lets change your first Conv2d code. IT should be

```
torch.nn.Conv2d(3, 64, kernel_size=(3, 3))
```

So after the first convolution using your formular, we will have

```
[3, 64, 198, 148]
```

After the second Conv2d operation, we will have

```
[3, 128, 196, 146].
```

The maxpooling which halves the activations we will have

```
[3, 128, 98, 73]
```

And finally the input of the fully connected layer will be 128×98×73 = 915712

Rui_Li 5d

Hi, I have some questions here:
"
So after the first convolution using your formular, we will have

```
[3, 64, 198, 148]
```

After the second Conv2d operation, we will have

```
[3, 128, 196, 146].
```

"
Since we assume batch_size=36, maybe it should be that after the first convolution layer, we will have

`[36, 64, 198, 148]`.

And after the second convolution layer, we will have

`[36, 128, 196, 146]`.