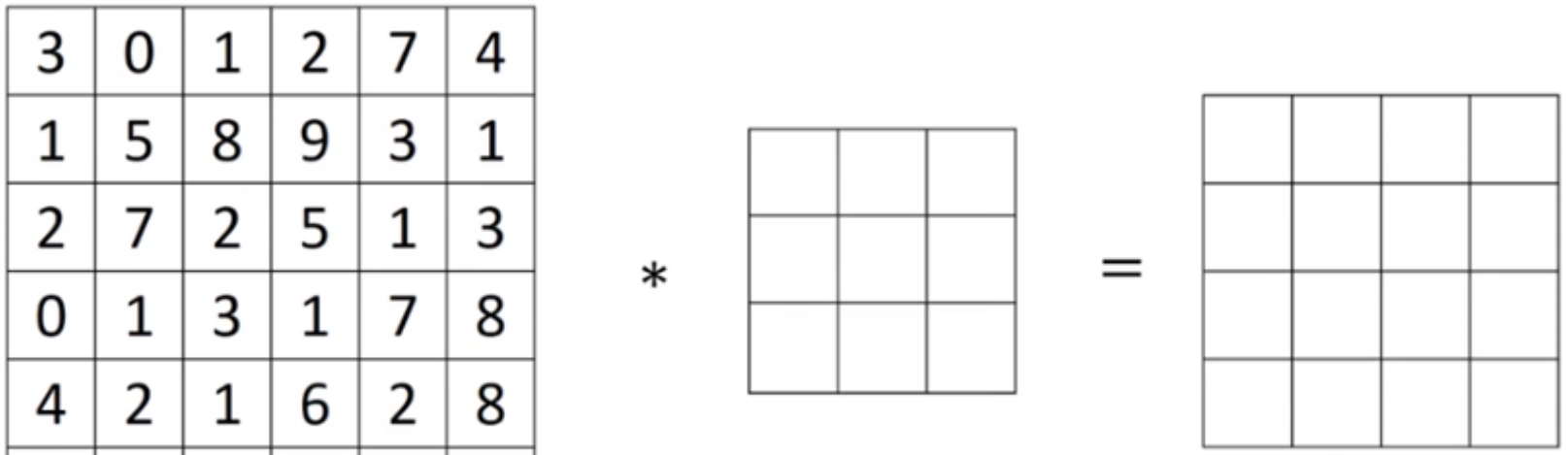
Convolutional Neural Network

SC-yjtian

2019.09.10

Convolution



n x n image

f x f filter

n-f+1 x n-f+1

Padding

- Valid: no padding

- Same: output size = input size

$$n+2p-f+1 = n$$

Stride

2	3	7	4	6	2	9
6	6	9	8	7	4	3
3	4	8	3	8	9	7
7	8	3	6	6	3	4
4	2	1	8	3	4	6
3	2	4	1	9	8	3
0	1	3	9	2	1	4

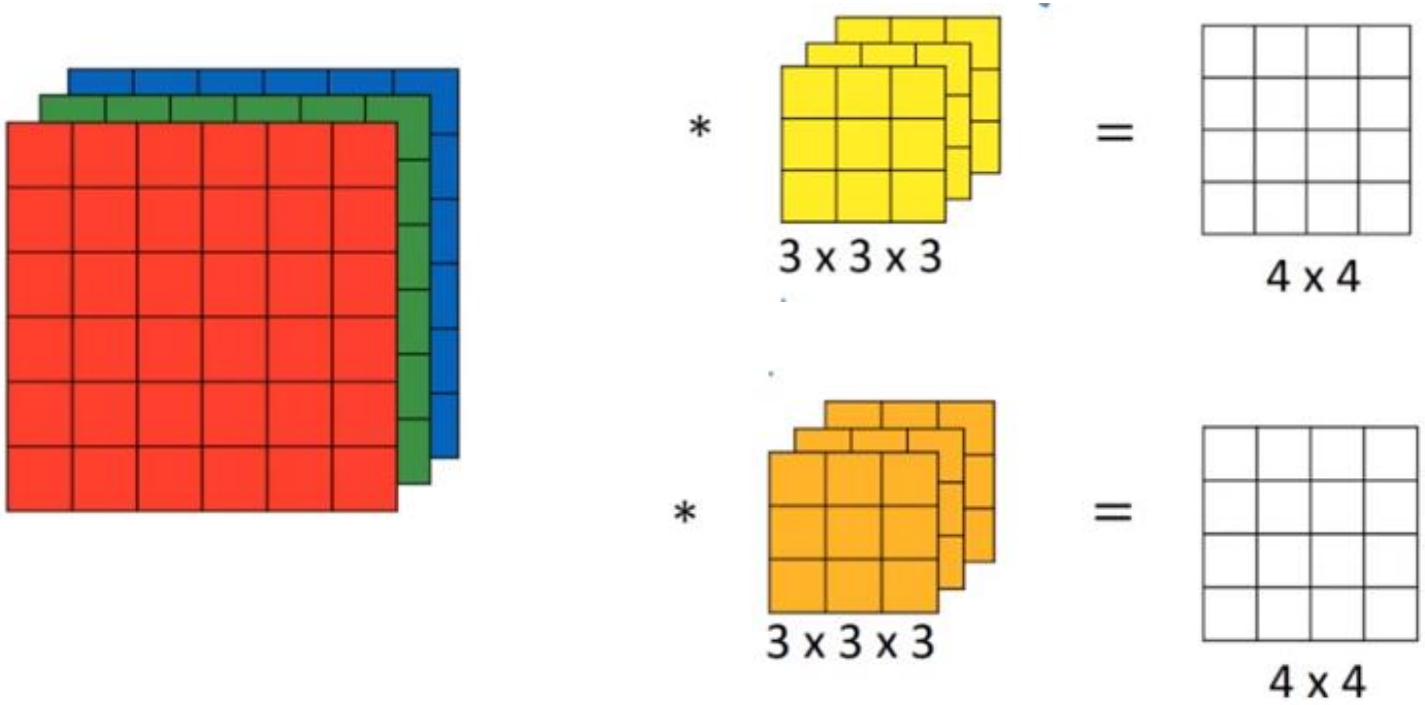
$*$

3	4	4
1	0	2
-1	0	3

$=$

$$\left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor$$

Conv layer



$$Z^1 = W^1 \cdot a^0 + b^1$$
$$a^1 = g(Z^1)$$

Test

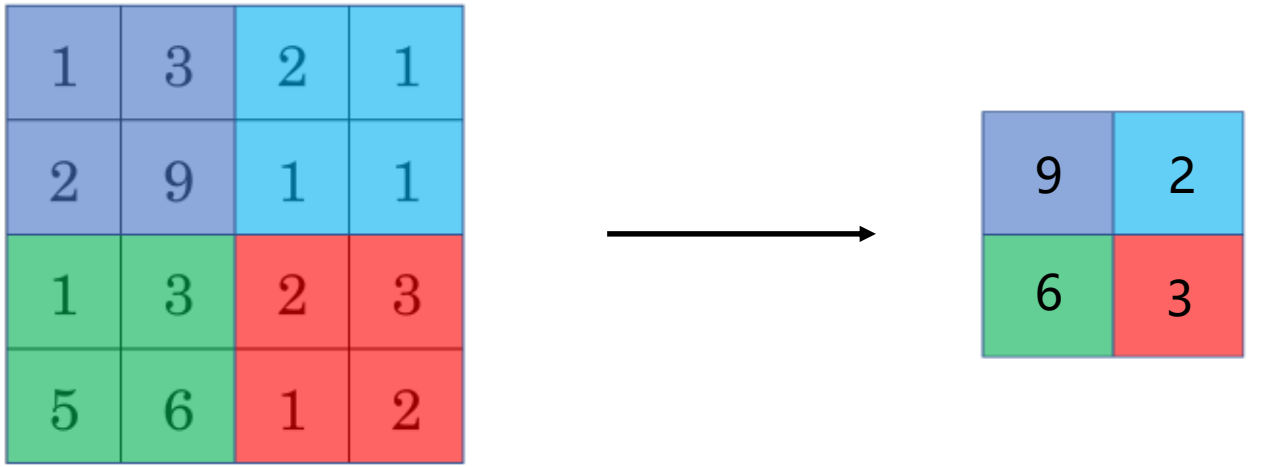
One conv layer:

5x5x3 input size, 10 filters that are 3x3x3, padding 1, stride 2

how many parameters? output size?

Pooling layer

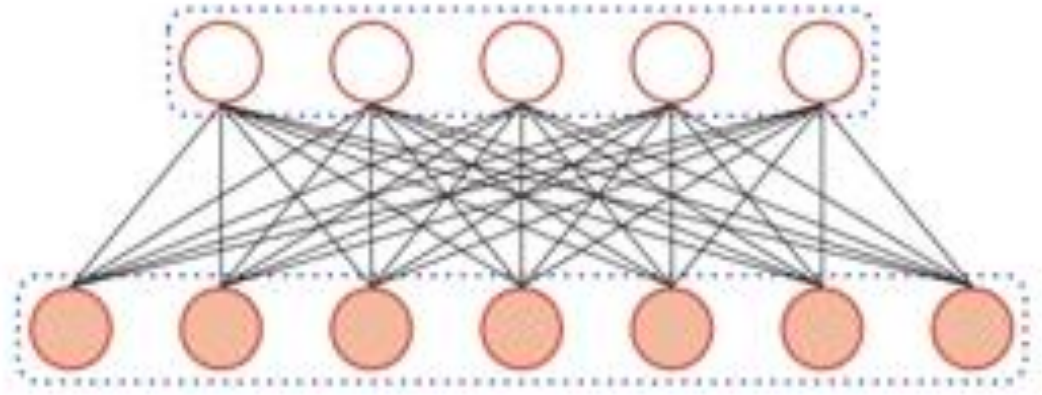
- Max Pooling



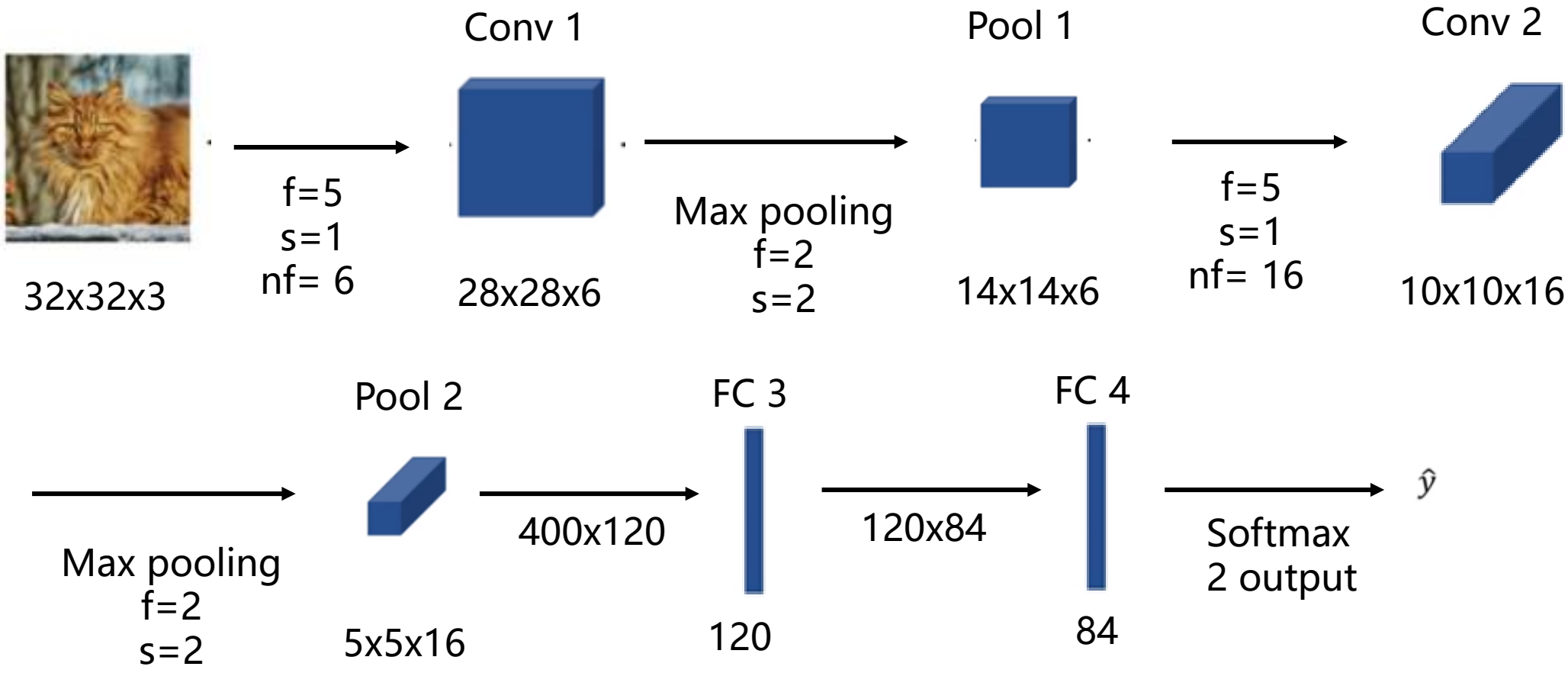
f = 2
s = 2
p = 0

- Average Pooling

Fully Connected layer



CNN Example

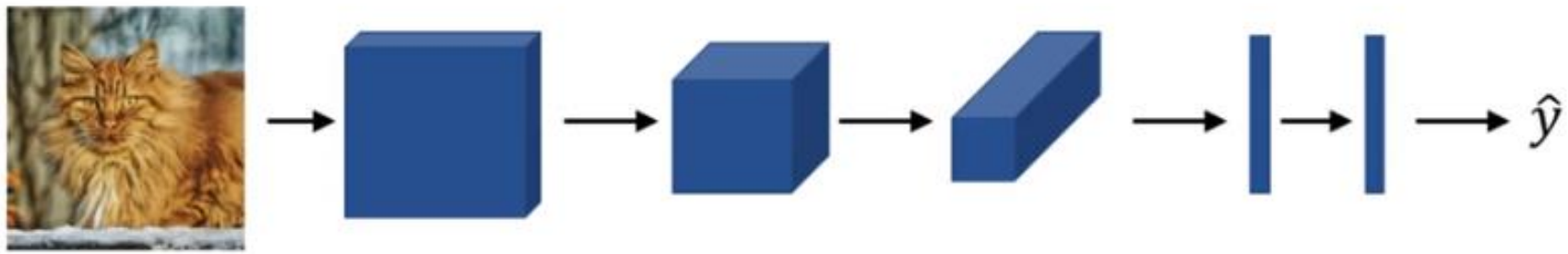

Conv 1
Pool 1
Conv 2
32x32x3
f=5
s=1
nf= 6
28x28x6
Max pooling
f=2
s=2
14x14x6
f=5
s=1
nf= 16
10x10x16
Pool 2
FC 3
FC 4
Max pooling
f=2
s=2
5x5x16
400x120
120
120x84
84
Softmax
2 output
$\hat{y}$

CNN Example

	Activation shape	Activation Size	# parameters
Input:	(32,32,3)	3,072	0
CONV1 (f=5, s=1)	(28,28,8)	6,272	208
POOL1	(14,14,8)	1,568	0
CONV2 (f=5, s=1)	(10,10,16)	1,600	416
POOL2	(5,5,16)	400	0
FC3	(120,1)	120	48,001
FC4	(84,1)	84	10,081
Softmax	(10,1)	10	841

CNN Train

Training set $(x^{(1)}, y^{(1)}) \ldots (x^{(m)}, y^{(m)})$.



$$\text{Cost } J = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(\hat{y}^{(i)}, y^{(i)})$$

Use gradient descent to optimize parameters to reduce J

CNN

Parameter sharing: A feature detector (such as a vertical edge detector) that's useful in one part of the image is probably useful in another part of the image.

Sparsity of connections: In each layer, each output value depends only on a small number of inputs.

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

*

1	0	-1
1	0	-1
1	0	-1

=

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

Exercise

Code: https://github.com/Tianyijian/nndl-exercise

tf.nn.conv2d

tf.nn.conv2d(input, filter, strides, padding, use_cudnn_on_gpu=None, name=None)

除去name参数用以指定该操作的name，与方法有关的一共五个参数：

第一个参数input：指需要做卷积的输入图像，它要求是一个Tensor，具有[batch, in_height, in_width, in_channels]这样的shape，具体含义是[训练时一个batch的图片数量, 图片高度, 图片宽度, 图像通道数]，注意这是一个4维的Tensor，要求类型为float32和float64其中之一

第二个参数filter：相当于CNN中的卷积核，它要求是一个Tensor，具有[filter_height, filter_width, in_channels, out_channels]这样的shape，具体含义是[卷积核的高度，卷积核的宽度，图像通道数，卷积核个数]，要求类型与参数input相同，有一个地方需要注意，第三维in_channels，就是参数input的第四维

第三个参数strides：卷积时在图像每一维的步长，这是一个一维的向量，长度4

第四个参数padding：string类型的量，只能是"SAME","VALID"其中之一，这个值决定了不同的卷积方式（后面会介绍）

第五个参数：use_cudnn_on_gpu:bool类型，是否使用cudnn加速，默认为true

结果返回一个Tensor，这个输出，就是我们常说的feature map

tf.nn.max_pool

```
tf.nn.max_pool(value, ksize, strides, padding, name=None)
```

参数是四个，和卷积很类似：

第一个参数value：需要池化的输入，一般池化层接在卷积层后面，所以输入通常是feature map，依然是[batch, height, width, channels]这样的shape

第二个参数ksize：池化窗口的大小，取一个四维向量，一般是[1, height, width, 1]，因为我们不想在batch和channels上做池化，所以这两个维度设为了1

第三个参数strides：和卷积类似，窗口在每一个维度上滑动的步长，一般也是[1, stride,stride, 1]

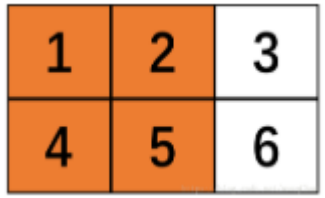
第四个参数padding：和卷积类似，可以取'VALID' 或者'SAME'

返回一个Tensor，类型不变，shape仍然是[batch, height, width, channels]这种形式

Padding

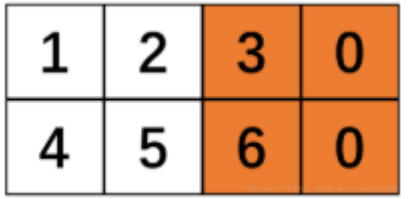
X: 2x3 filter: 2x2

- Valid: no padding, remove



$$\left\lceil \frac{(W-F+1)}{S} \right\rceil$$

- Same: padding, add



$$\left\lceil \frac{W}{S} \right\rceil$$

Ref: https://blog.csdn.net/wuzqChom/article/details/74785643

Thank You