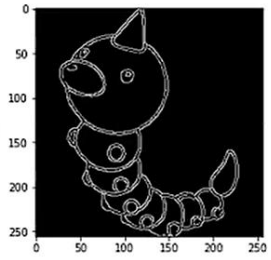
更多的parameter, easier to overfit ! why ?

对于数码宝贝与宝可梦：

$f($  $)$ = 宝可梦 or 数码宝贝

↳由图可知：数码宝贝比较复杂，宝可梦比较简单

宝可梦



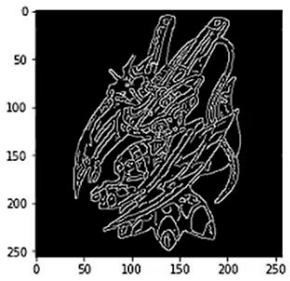
Edge detection

3558

$e($  $) = 3558$

白色区域像素量

数码宝贝



7389

$e($  $) = 7389$

可以用 Edge detection 技术将图变成一个黑白图，白色是线边

模型复杂度高，代表有很多的未知参数（feature）

模型的好坏取决于取到何 train，若不是独立同分布的话，train 的模型会较差，则在 model 遇到一个 new test 时，效果会更差。

具体的试指导没看见，在 P23 的视频中讲的比较详细。

CNN的第一种解释：

Image Classification

100 x 100

(All the images to be classified have the same size.)

CNN

Model

$$\begin{bmatrix} \vdots \\ 0.2 \\ 0.7 \\ 0.1 \\ \vdots \end{bmatrix} = y'$$

Cross entropy

$$\hat{y} = \begin{bmatrix} \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \end{bmatrix} \begin{matrix} \text{dog} \\ \text{cat} \\ \text{tree} \end{matrix}$$

这有多少个数，就代表可以辨识多少种物

例：$\hat{y} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \rightarrow$ cat

可辨识4个，其中输入的图片是1的结果cat

一般输入的图像是相同像素

里面自带softmax

原本是概率，过一个cross entropy得到分类结果

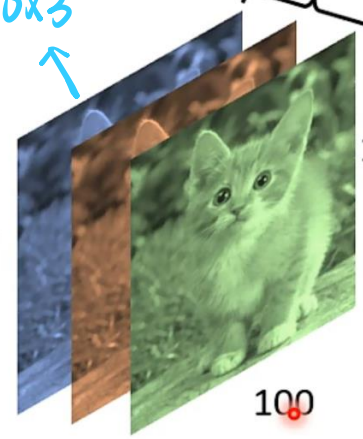
Image Classification

100 x 100

3-D tensor

3 channels

100x100x3



100

100

拉成一个矩阵

100 x 100

100 x 100

100 x 100

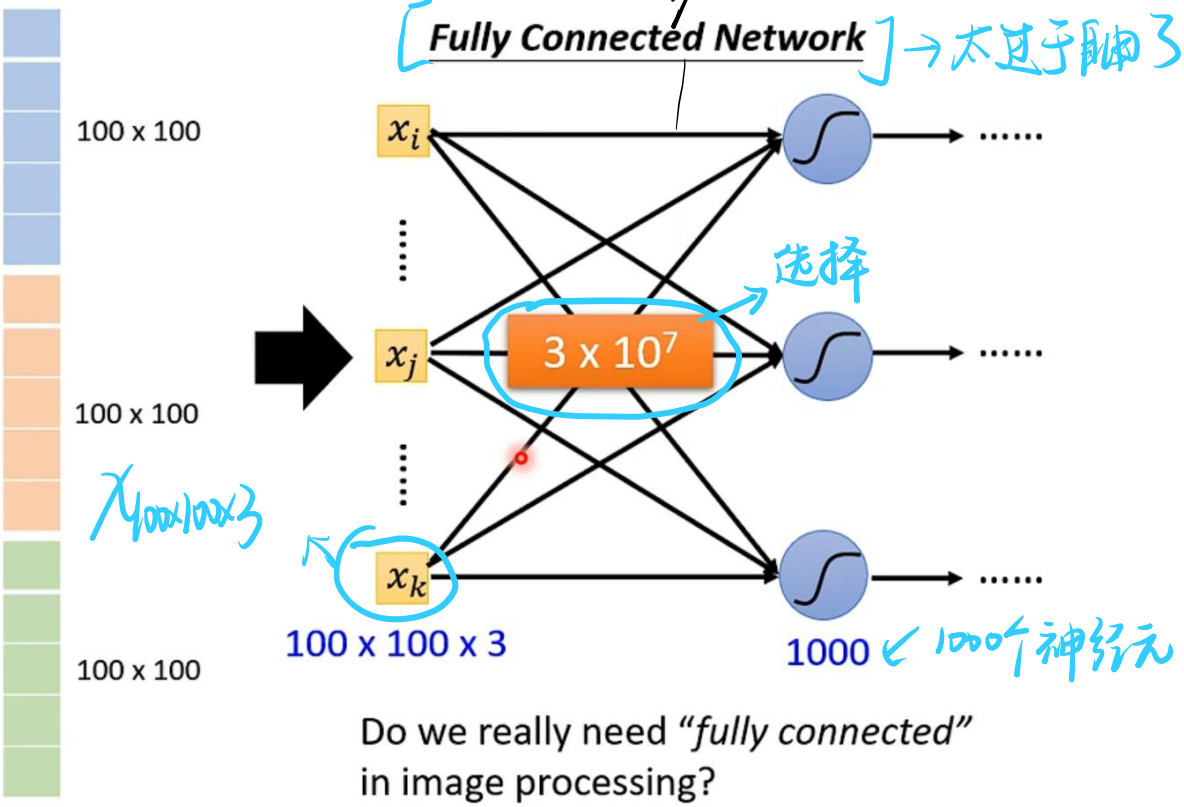
对于一个黑白图：channel = 1

一个图片是RGB，即三色

故 channel = 3

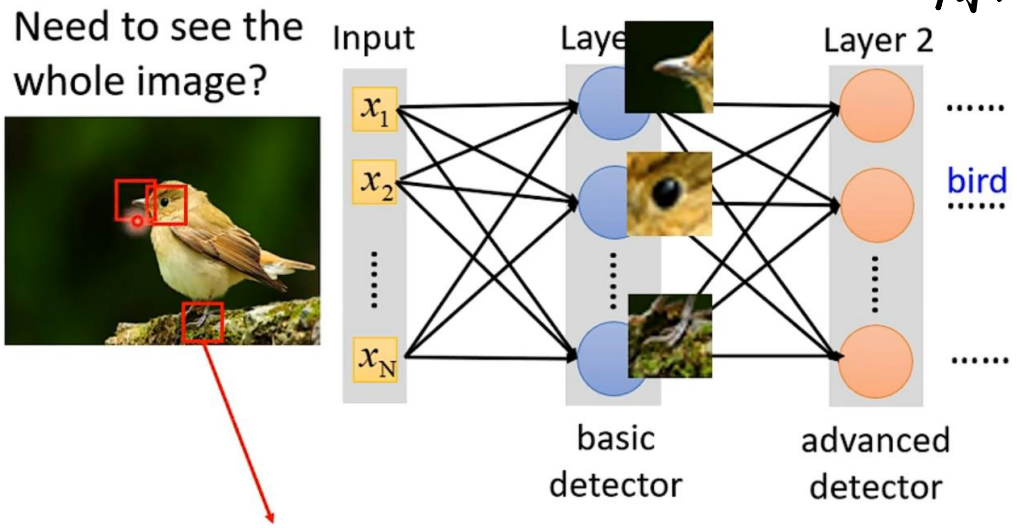
如果我们对这个picture的3x100x100进行Fully Connected(全连接)

太容易overfitting了



Do we really need "*fully connected*" in image processing?

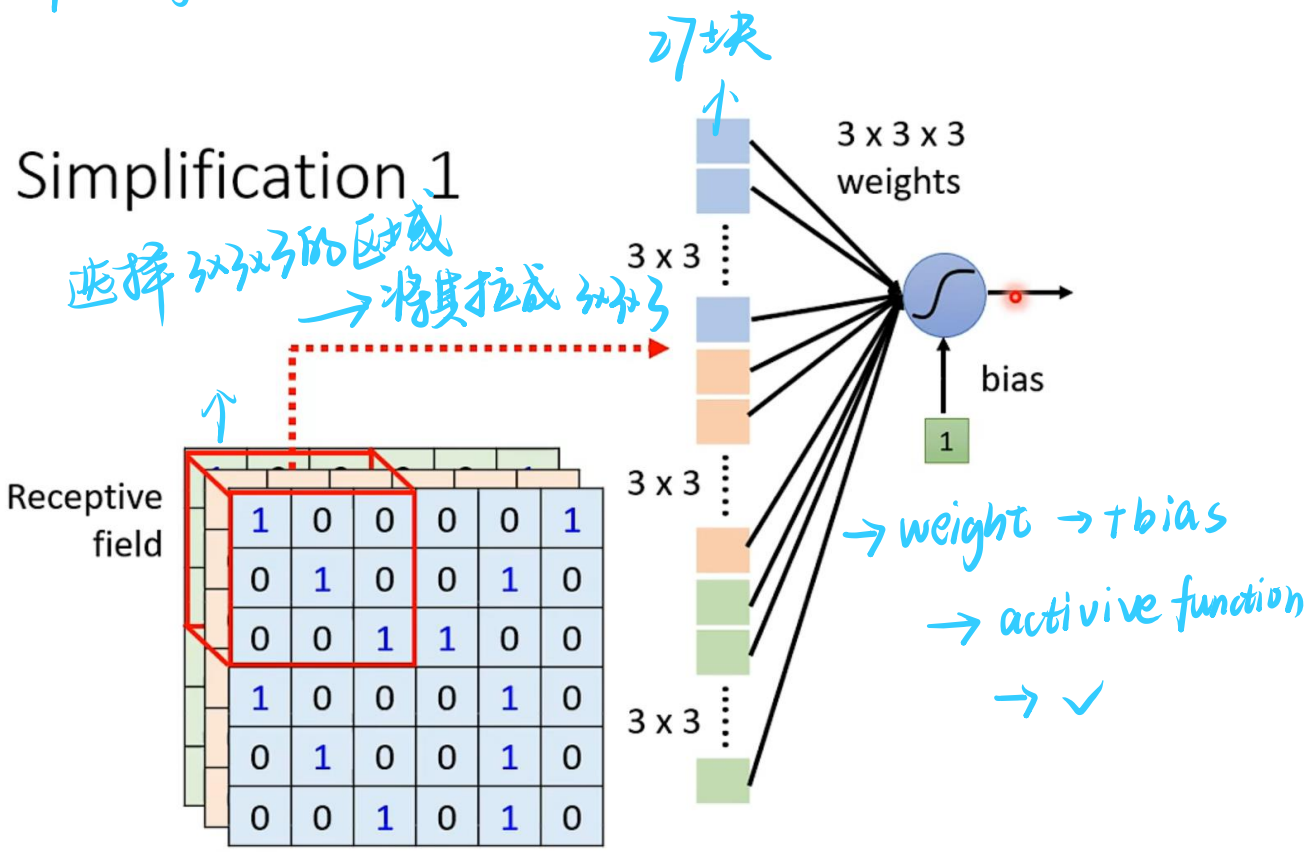
但是：我们观察一张图片，要把每部都输进去吗？→NO，一般只用看局部便能识别



Some patterns are much smaller than the whole image.

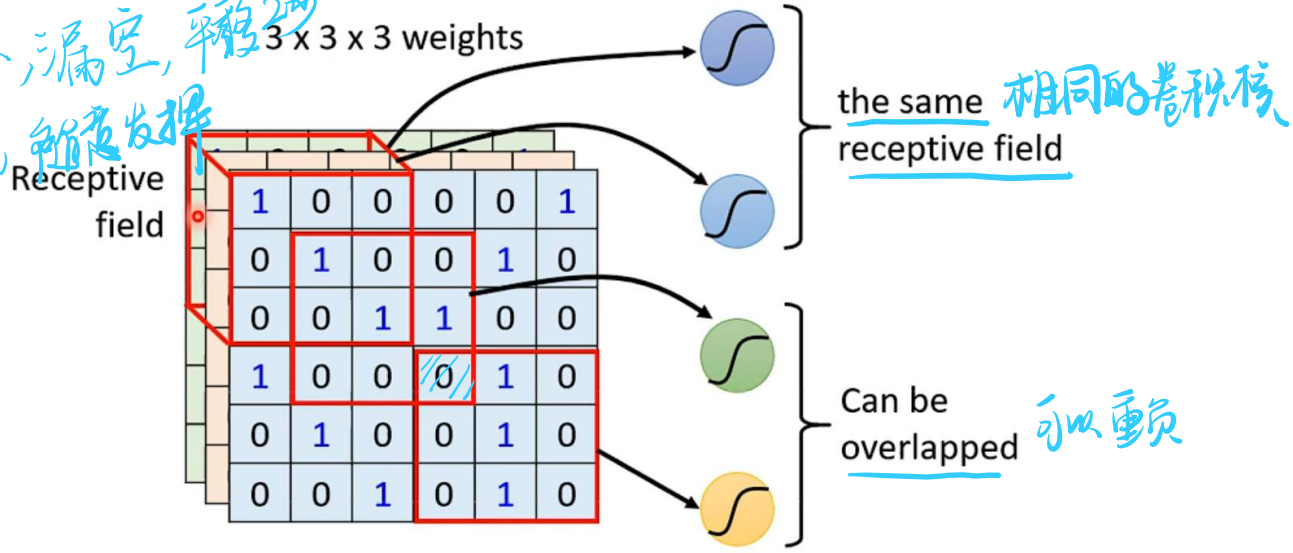
Receptive field

Simplification 1



Simplification 1

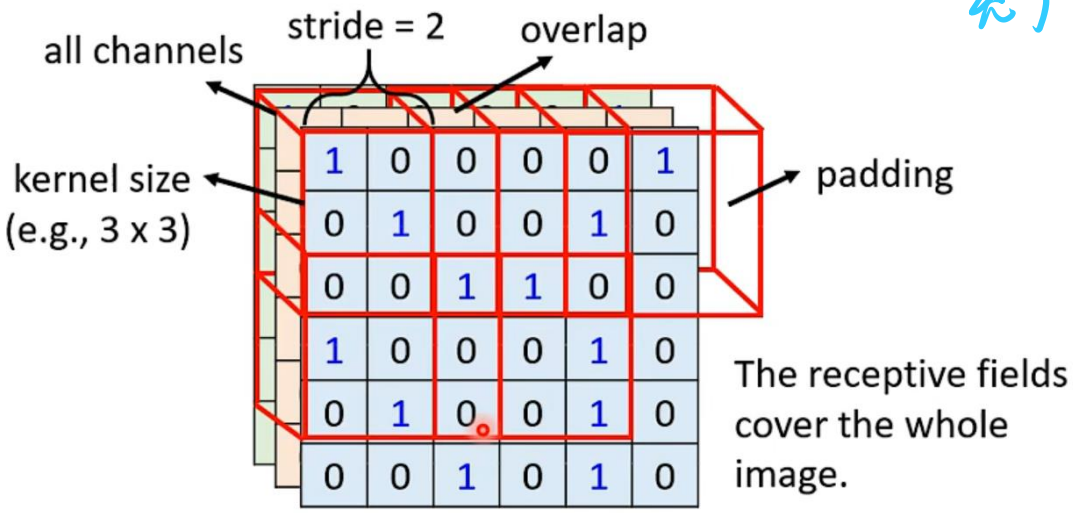
- Can different neurons have different sizes of receptive field?
- Cover only some channels?
- Not square receptive field?



经典1：Kernel size = 3x3（卷积核）

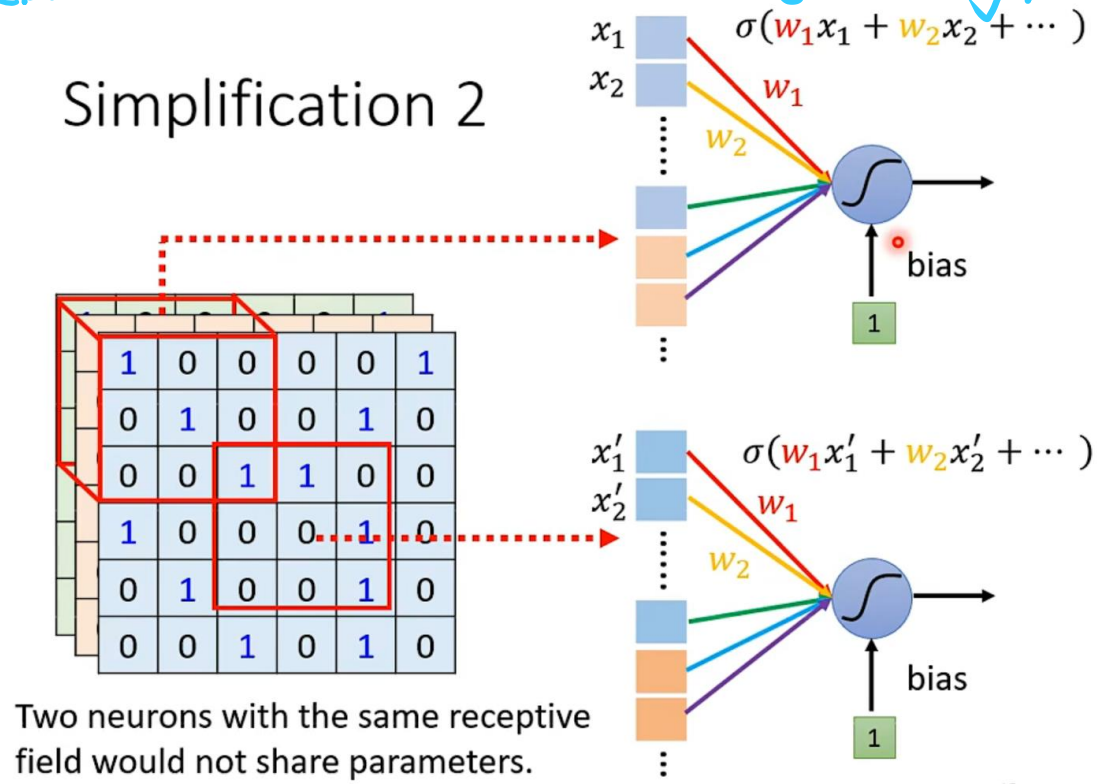
有些部分重叠，多余部用padding（可用全0填充，可用平均填充，也可用就近填充）

Each receptive field has a set of neurons (e.g., 64 neurons).



经典2：Sharing parameters.

来源是因为有些图片是位置不一，而使用的参数差不多，为减到3 sharing parameters.



Two neurons with the same receptive field would not share parameters.

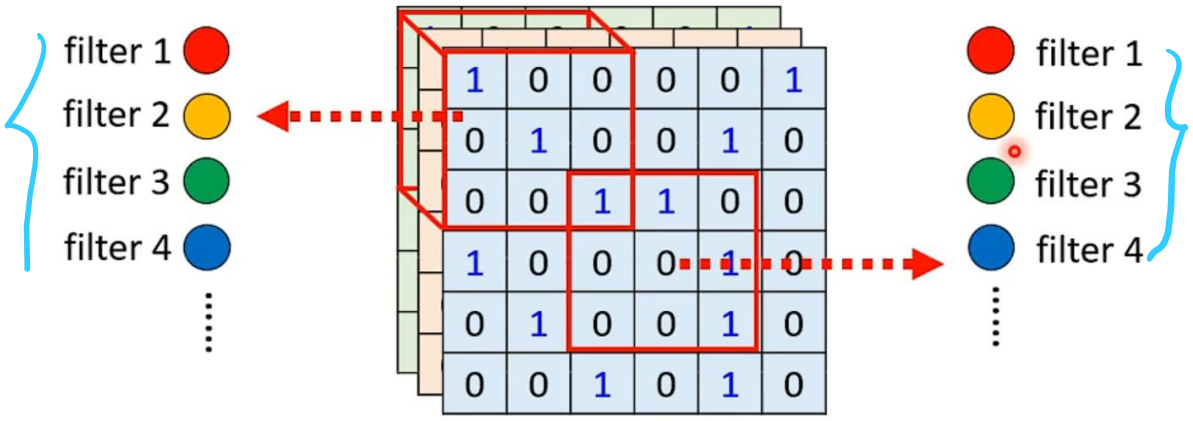
problem？weight与bias相同，会导致相同的结果吗？

对于不同的卷积层，

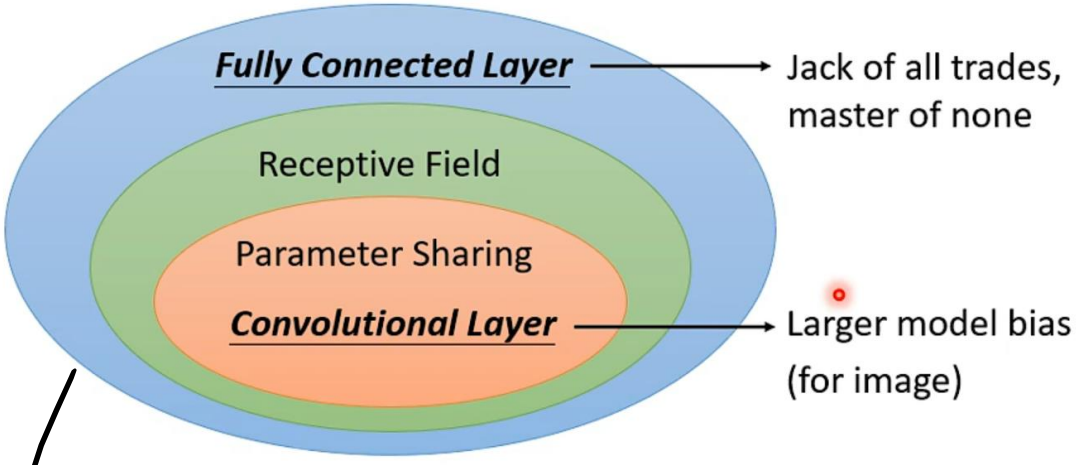
Answer = 不会，因为输入不同，其输出也会不同！

Each receptive field has a set of neurons (e.g., 64 neurons).

Each receptive field has the neurons with the same set of parameters.



Benefit of Convolutional Layer

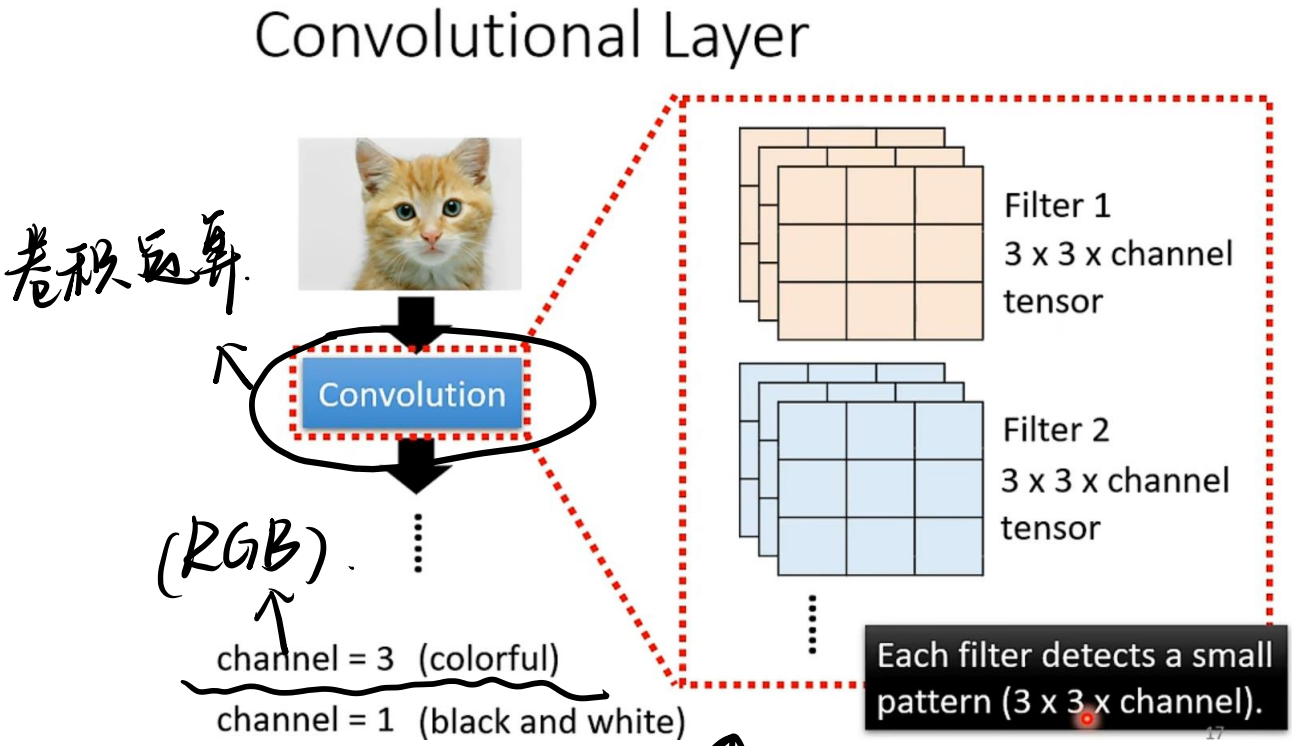


- Some patterns are much smaller than the whole image.
- The same patterns appear in different regions.

一层一层的限制，使fully connected并没有那么freestyle

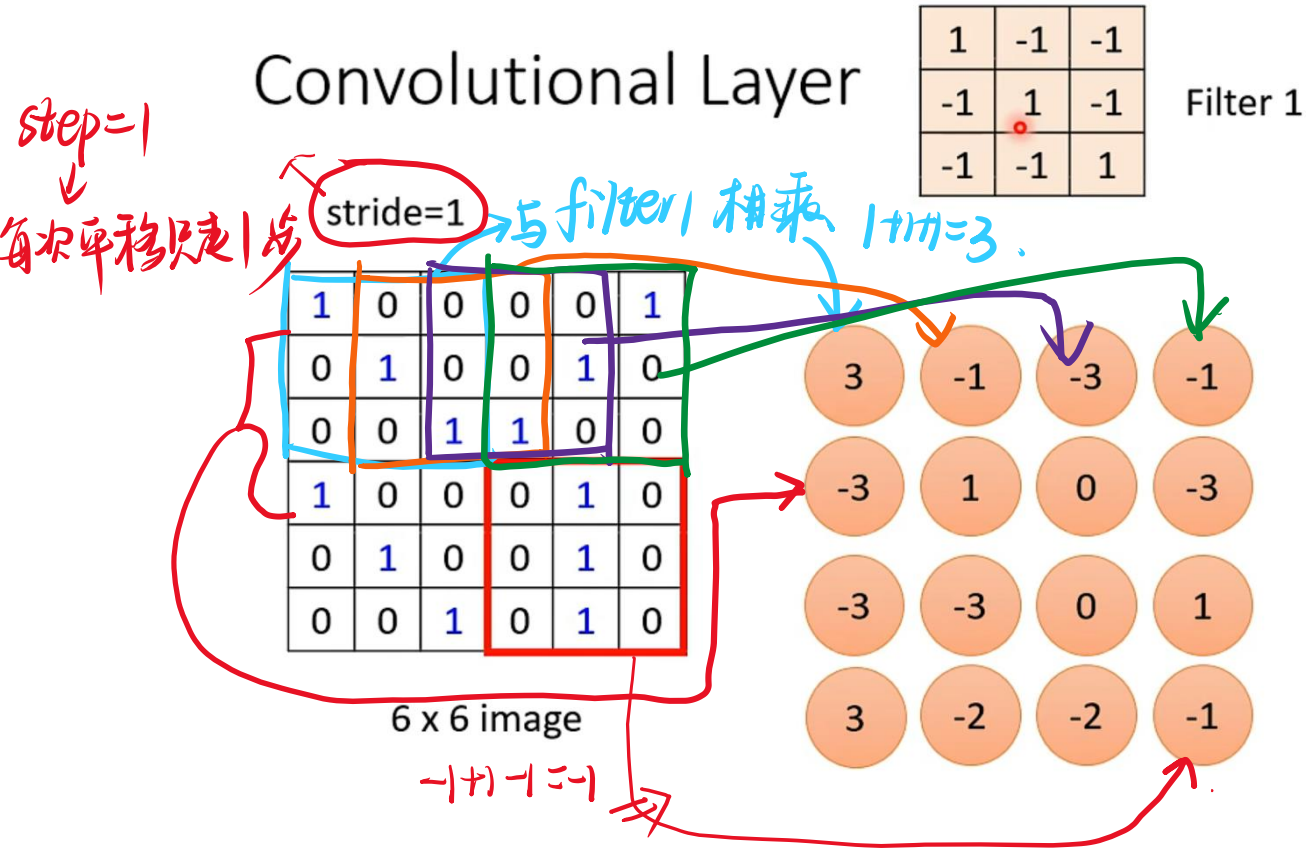
向CNN就是在共享参数和卷积层来的

第二种常见的关于CNN的解释：



convolution是有2个filter

那：convolutional如何运算：



再将刚刚的operation用filter 2的一次，得到蓝色区域

Convolutional Layer

-1	1	-1
-1	1	-1
-1	1	-1

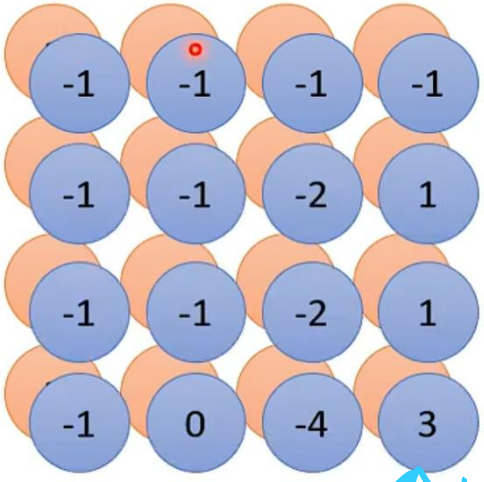
Filter 2

stride=1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

Do the same process for every filter



相乘 filter 2

得力=3

Convolutional Layer

-1	1	-1
-1	1	-1
-1	1	-1

Filter 2

故即有多少个Filter，就有多少层结果

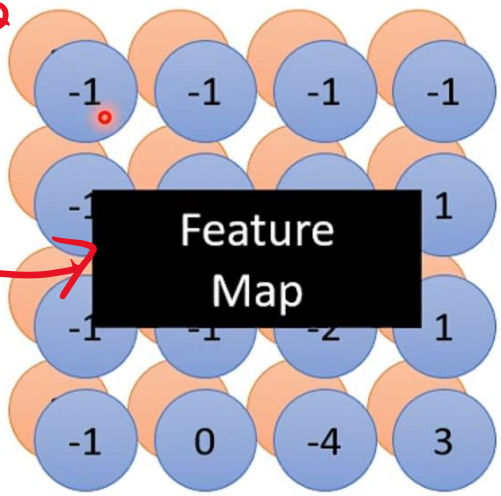
这些结果的总和称为

stride=1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

Do the same process for every filter



若有5个filter，故即有：

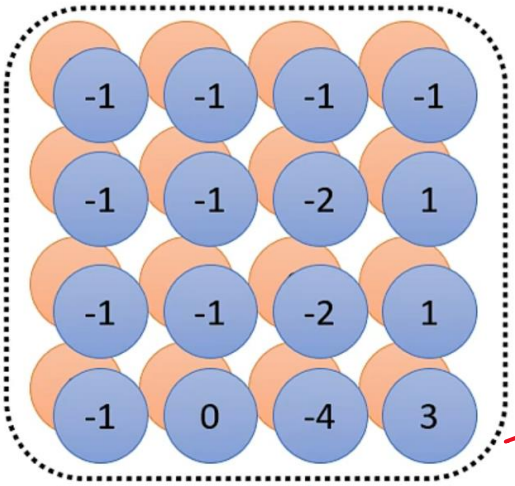
5x4x4

Multiple Convolutional Layers

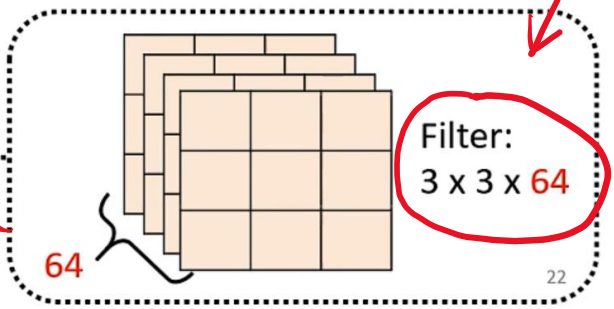
64 filters

Convolution ①

Convolution ②



"Image" with 64 channels



22

第②层的convolution的输入为 3x3x64

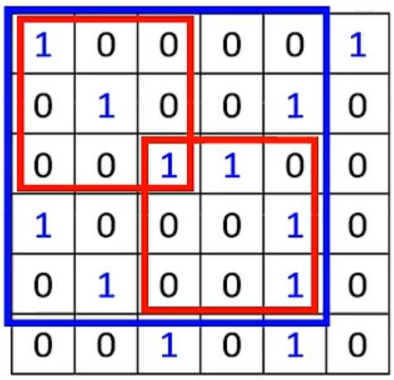
problem：很明显从64张变为4x4，会不会有点消失或者损失呢？

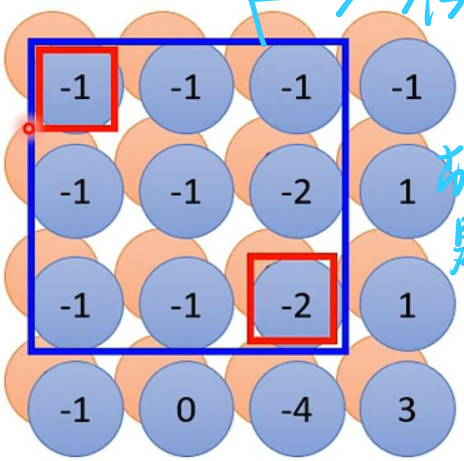
Multiple Convolutional Layers

64 filters

Convolution

Convolution



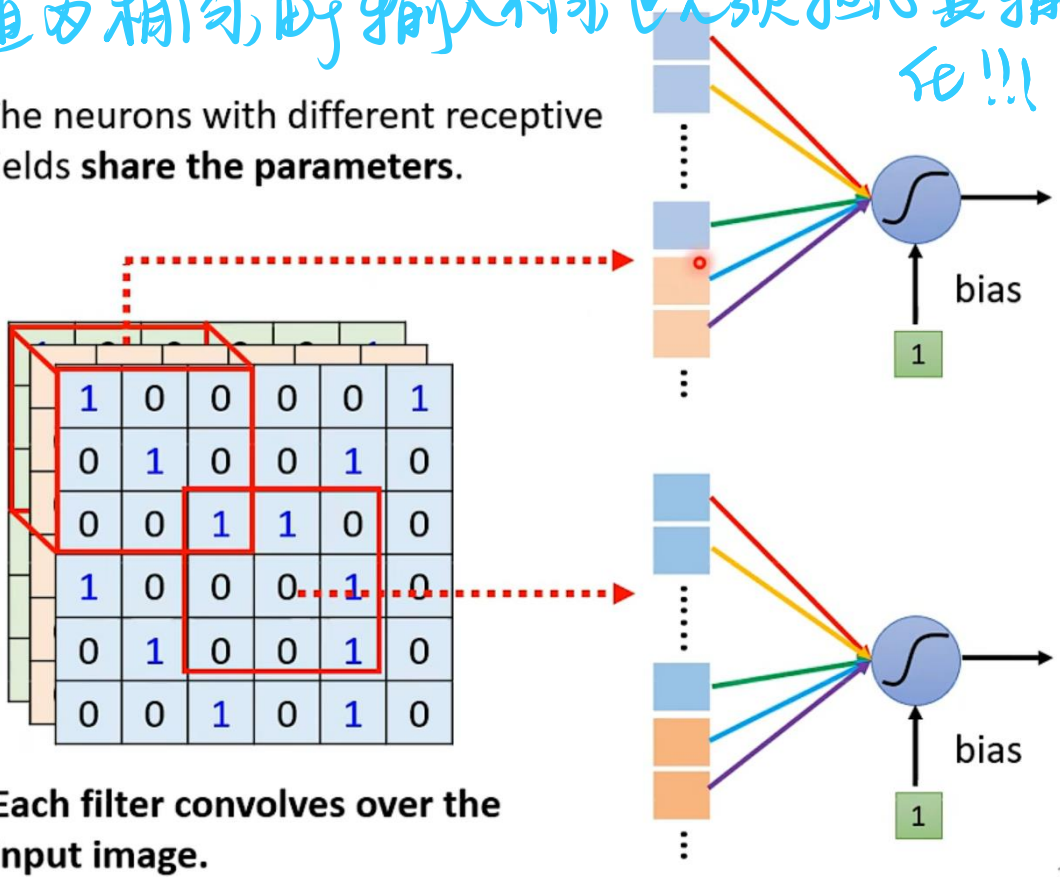


一个3x3的在原来情况下表示5x5，故不会有上述问题！

23

故而同样对此进行 share parameters 处理，对于这些 weight 和 bais 设置为相同，即输入不同也无须担心最终结果会变化!!!

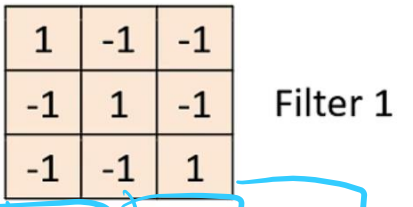
The neurons with different receptive fields **share the parameters.**

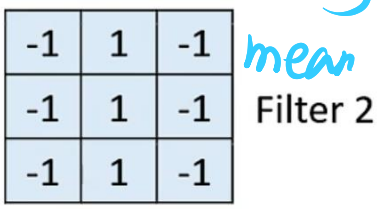


Each filter convolves over the input image.

池化/下采样/pooling，又这里别人采用的是 max Pooling 也有采用 min Pooling 或 mean pooling 等

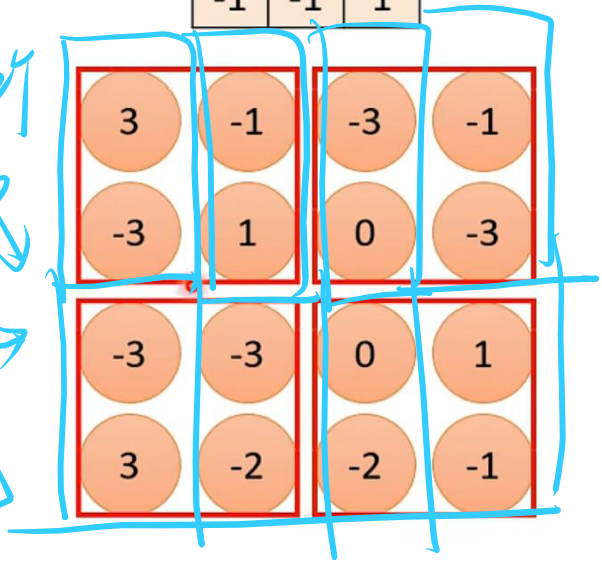
Pooling – Max Pooling

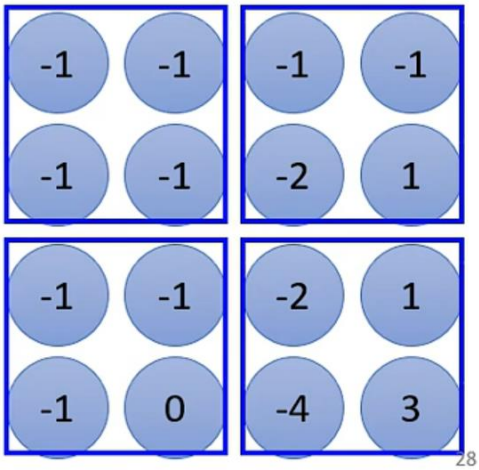




也可以这样做如下图

这是自己设定的





Pooling – Max Pooling

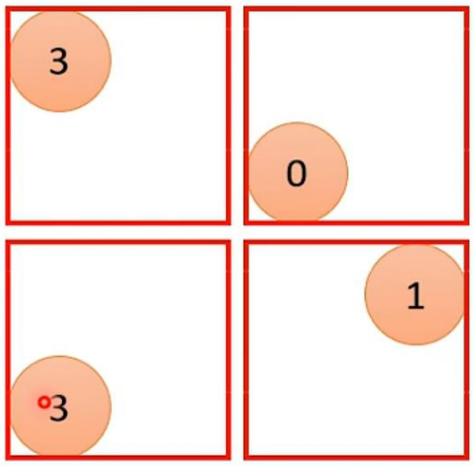
提取最大的数

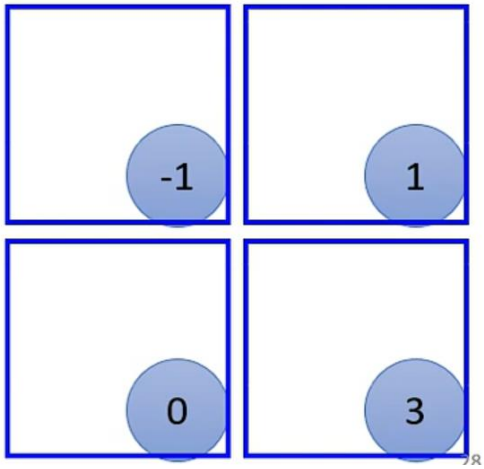
1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

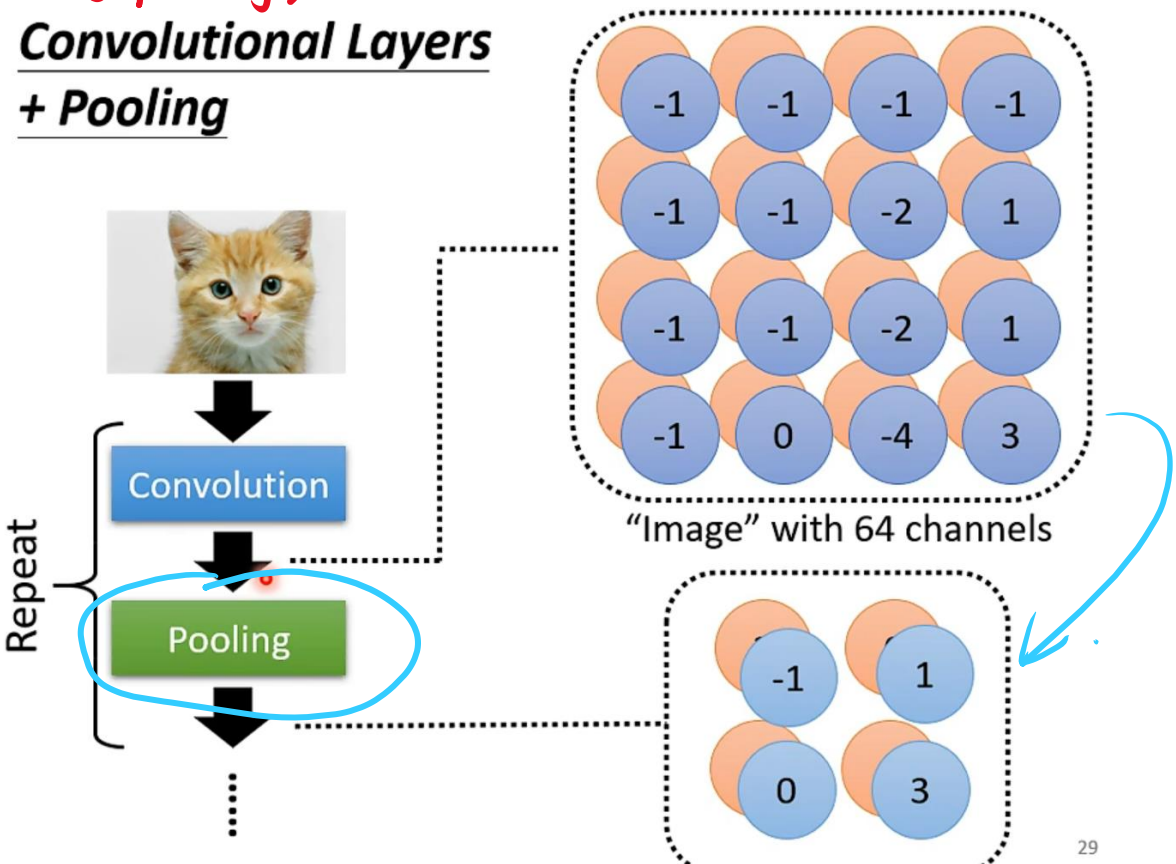
-1	1	-1
-1	1	-1
-1	1	-1

Filter 2





在一些大型数据中，pooling可以减少运算量，使其简化（若Gpu够，也可不进行pooling）



这个pooling可以使4x4x6 filter缩减成2x2x64filter

确实缩减了cpu所计资源，但对于一些小数据作

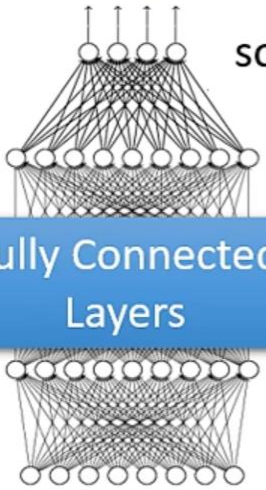
精细处理，一般采用pooling效果会很差！

一整个CNN的步骤

The whole CNN

y=[0,1,0,1,0...,0] → cat dog .....

softmax 归一化



全连接层

Fully Connected Layers

Convolution

Pooling

repeat

可以多很多层

Convolution

Pooling

Flatten

但是，真的需要pooling吗???

Why CNN for Go playing?

- Subsampling the pixels will not change the object

Pooling → How to explain this???

围棋大师选取的最优维数48

(48表示)

Neural network architecture. The input to the policy network is a 19 × 19 × 48 image stack consisting of 48 feature planes. The first hidden layer zero pads the input into a 23 × 23 image, then convolves *k* filters of kernel size 5 × 5 with stride 1 with the input image and applies a rectifier nonlinearity. Each of the subsequent hidden layers 2 to 12 zero pads the respective previous hidden layer into a 21 × 21 image, then convolves *k* filters of kernel size 3 × 3 with stride 1, again followed by a rectifier nonlinearity. The final layer convolves 1 filter of kernel size 1 × 1 with stride 1, with a different bias for each position, and applies a softmax function. The match version of AlphaGo used *k* = 192 filters; Fig. 2b and Extended Data Tab... 256 and 384 filters

Alpha Go does not use Pooling .....

输入：19×19×48

① 23×23，用0填充

② 用5×5，步1进行平移.

其中：Alpha Go没有采用pooling，因为每一个模型不模糊做到很精确，若pooling的话，误差会很大，从而结果很垃圾，故

不用pooling在AlphaGo上。

但是对于CNN而言，若将picture放大或缩小，可能无法认出picture是对的了。这时便引入一个新的技术：Spatial Transformer layer

To learn more ...

- CNN is not invariant to scaling and rotation (we need data augmentation ☺).

Dog ✓

Cat X

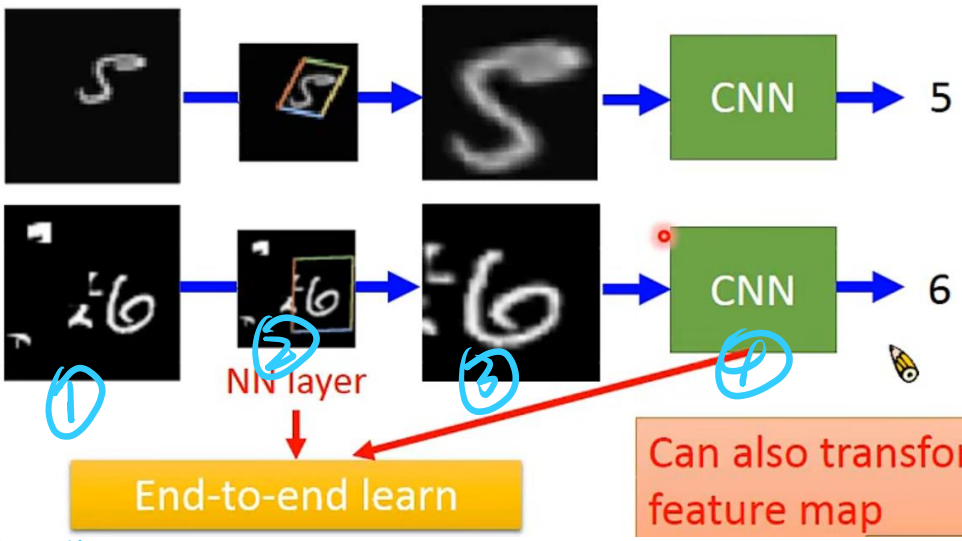
Spatial Transformer Layer

https://youtu.be/SoCywZ1hZak
(in Mandarin)

Spatial Transformer Layer

CNN无法识别缩放与旋转后的。

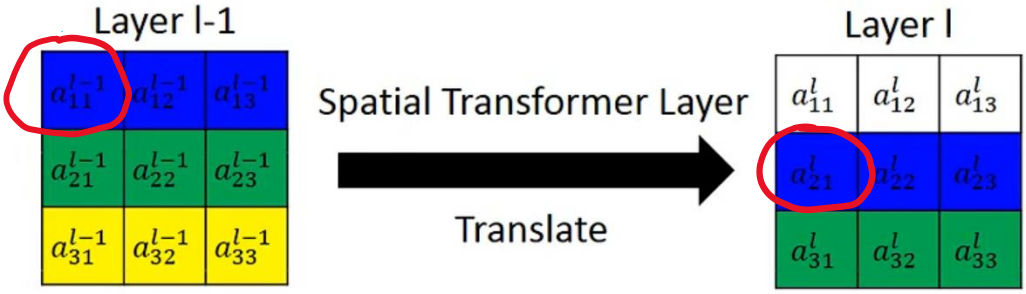
- CNN is not invariant to scaling and rotation



把①缩放 → ② 旋转 → ③ → 过CNN → ④ → 结果

Spatial Transformer Layer

- How to transform an image/feature map



General layer: $a_{nm}^{l} = \sum_{i=1}^{3}\sum_{j=1}^{3} w_{nm,ij}^{l} a_{ij}^{l-1}$

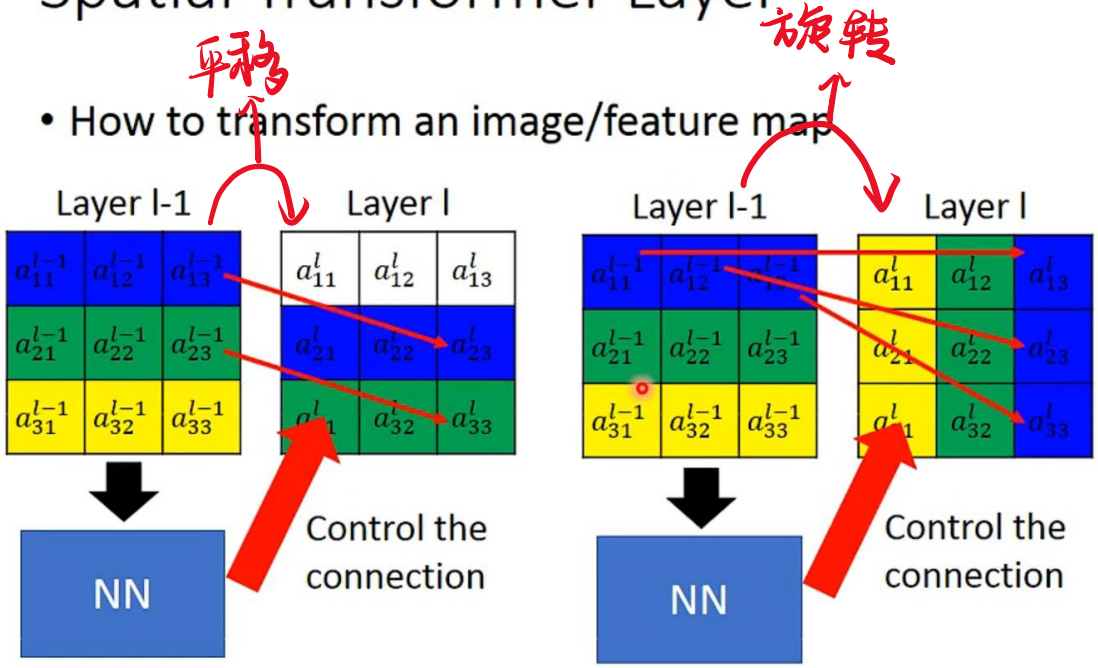
If we want translate as above: $a_{nm}^{l} = a_{(n-1)m}^{l-1}$

$w_{nm,ij}^{l} = 1 \quad if\ i = n-1, j = m \qquad w_{nm,ij}^{l} = 0 \quad otherwise$

$\rightarrow a_{21}^{l} = a_{11}^{l-1}$

Spatial Transformer Layer

- How to transform an image/feature map



$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 2x \\ 2y \end{bmatrix} = 2\begin{bmatrix} x \\ y \end{bmatrix}$$

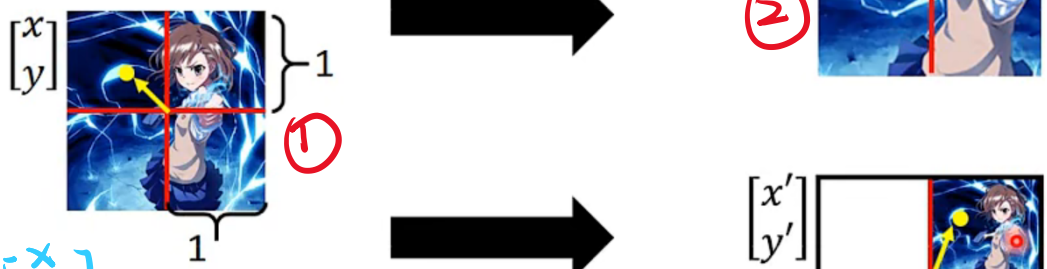
即②是由①在中心进行扩大2倍得到的结果

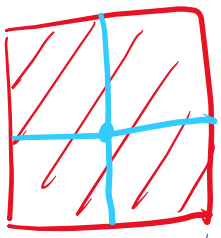
Image Transformation

Expansion, Compression, Translation

缩放 平移

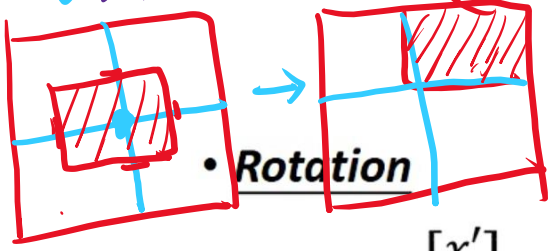
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$





$\begin{bmatrix} x \\ y \end{bmatrix}$

↓ 以中心为点进行缩放

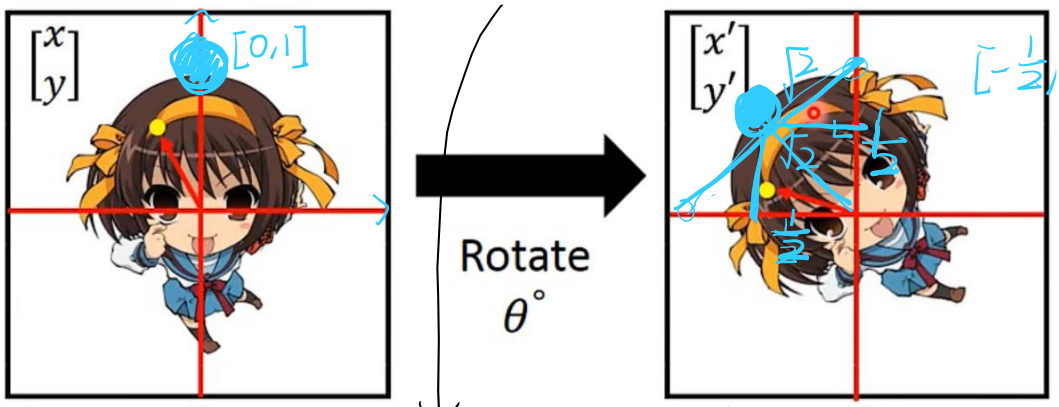


$$= \begin{bmatrix} 0.5x \\ 0.5y \end{bmatrix} + \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$$

- **Rotation**

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$



$\theta = 45°$, $\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \frac{1}{2}x - \frac{1}{2}y \\ \frac{1}{2}x + \frac{1}{2}y \end{bmatrix}$

时 $\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ 故 $\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -\frac{1}{2} \\ \frac{1}{2} \end{bmatrix}$

结果一致 ok

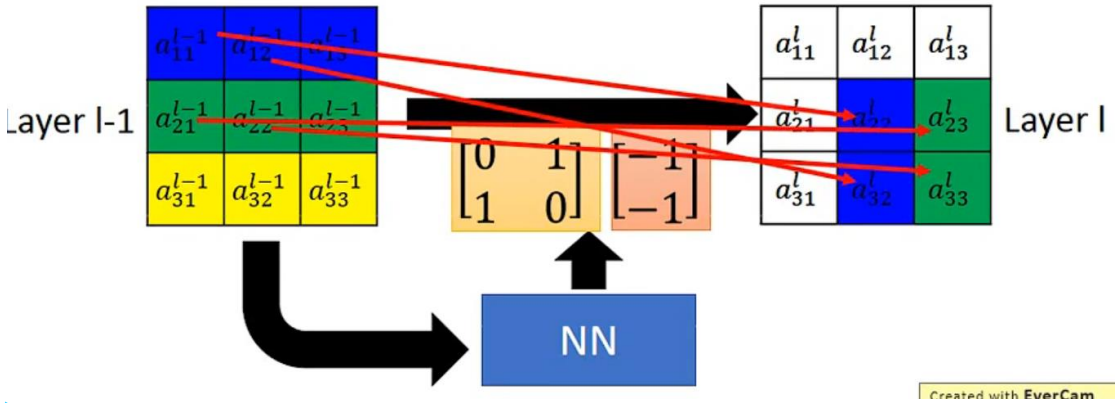
若均为整数，此时一块对应一块完全 NO problem!

Spatial Transformer Layer

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} -1 \\ -1 \end{bmatrix}$$

6 parameters to describe the affine transformation

Index of layer l-1　　Index of layer l



Created with EverCam

可见，若等号右侧为整数？

Spatial Transformer Layer

$$\begin{bmatrix} 1.6 \\ 2.4 \end{bmatrix} = \begin{bmatrix} 0 & 0.5 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 2 \\ 2 \end{bmatrix} + \begin{bmatrix} 0.6 \\ 0.4 \end{bmatrix}$$

6 parameters to describe the affine transformation

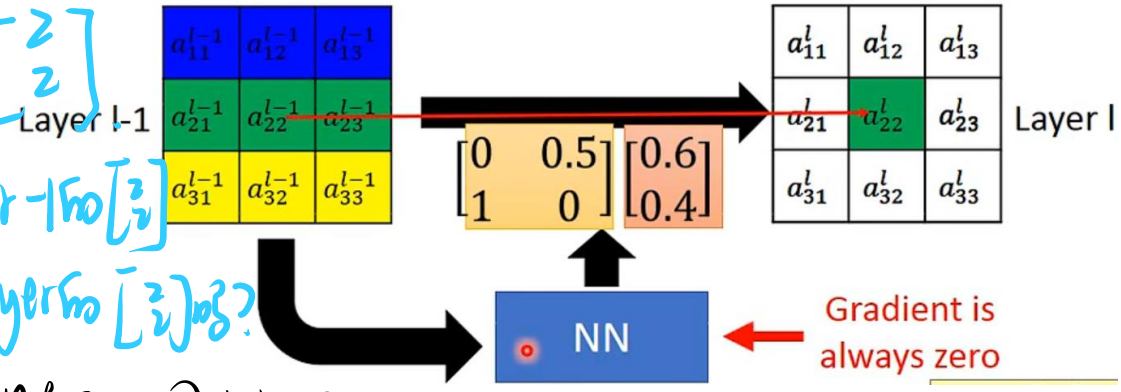
Index of layer l-1　　Index of layer l

若以小数点后为准

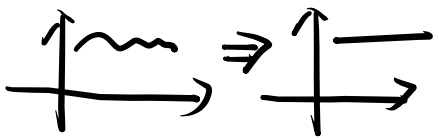
$\begin{bmatrix} 1.6 \\ 2.4 \end{bmatrix} \Rightarrow \begin{bmatrix} 2 \\ 2 \end{bmatrix}$

那原layer-1的[2,2]还能对layer的[2,2]吗？

What is the problem?



显然是不能的，因为如果小数部分全为一个格子，此时得由一个连续变成分段：



那 Gradient = 0，无法再继续

故而：不可采用简单的回传法 new way!

Interpolation

Now we can use gradient descent

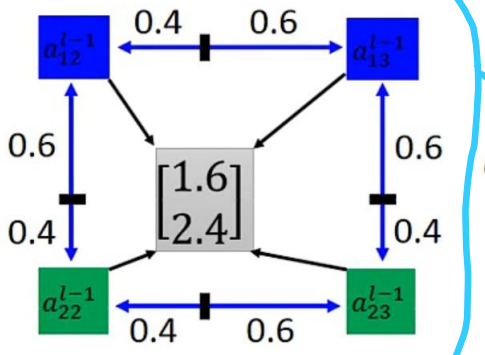
$$\begin{bmatrix}1.6\\2.4\end{bmatrix} = \begin{bmatrix}0 & 0.5\\1 & 0\end{bmatrix}\begin{bmatrix}2\\2\end{bmatrix} + \begin{bmatrix}0.6\\0.4\end{bmatrix}$$

6 parameters to describe the affine transformation

Index of layer l-1 　　Index of layer l

a^l_{11}	a^l_{12}	a^l_{13}
a^l_{21}	a^l_{22}	a^l_{23}
a^l_{31}	a^l_{32}	a^l_{33}

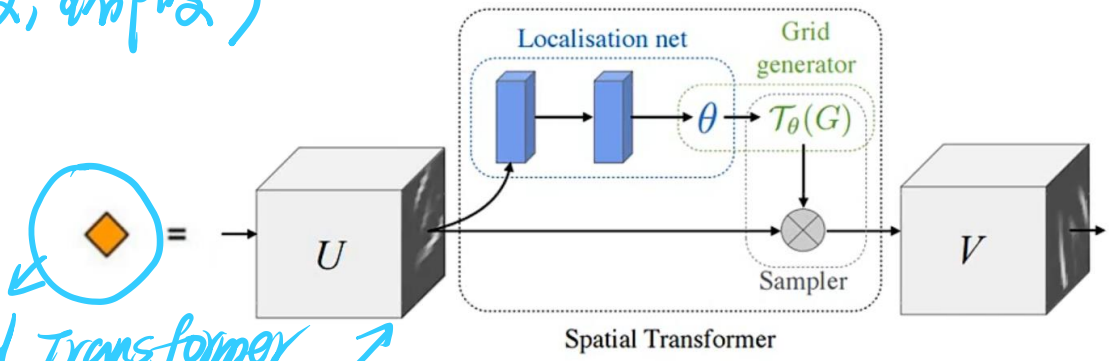
Layer l



$$\begin{aligned}a^l_{22} &= (1-0.4)\times(1-0.4)\times a^{l-1}_{22}\\ &+ (1-0.6)\times(1-0.4)\times a^{l-1}_{12}\\ &+ (1-0.6)\times(1-0.6)\times a^{l-1}_{13}\\ &+ (1-0.4)\times(1-0.6)\times a^{l-1}_{23}\end{aligned}$$

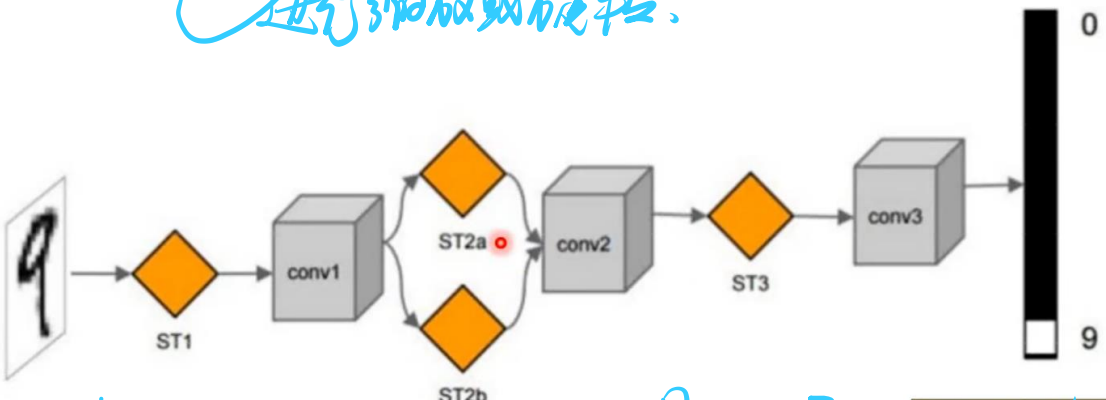
也就意味着：a^l_{22} 的值与 a^{l-1}_{12}、a^{l-1}_{13}、a^{l-1}_{22}、a^{l-1}_{23} 相关

（一阶，局部的）



Spatial Transformer

进行缩放或旋转。

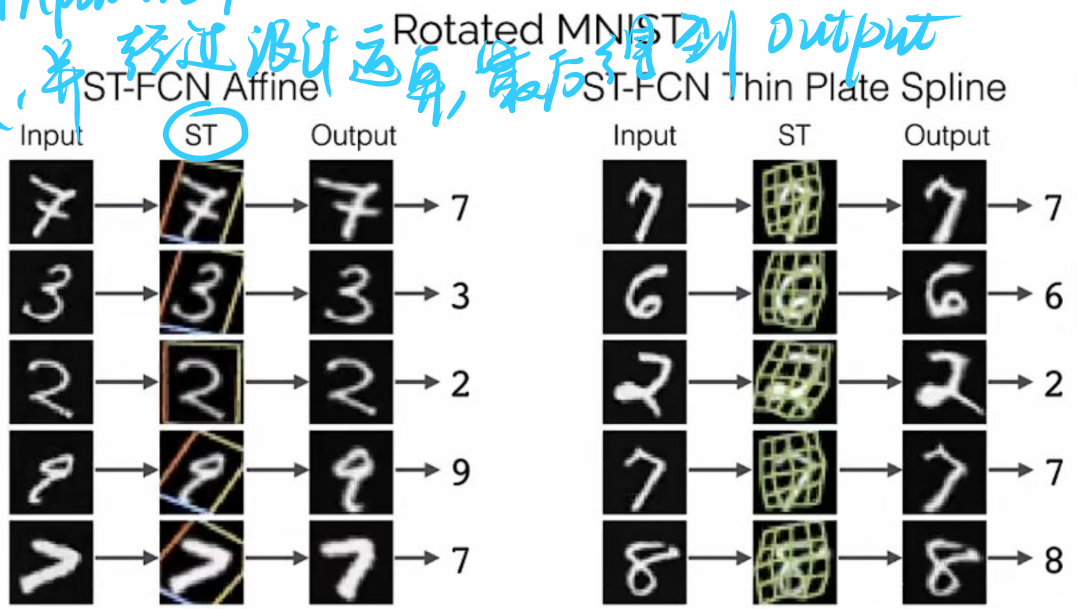


而spatial transformer 不仅可以用于最前的conv convolution，也可以在后面使用，也可以2个 spatial transformer 同步使用！

对于手写 dataset：MNIST 为例：

ST ⇒ Spatial tranforma

ST ⇒ 把 input 的 picture 的 重要区域圈定起来，再经过设计运算，拿后续到 output CNN 输入，并经过设计运算，拿后续到 output

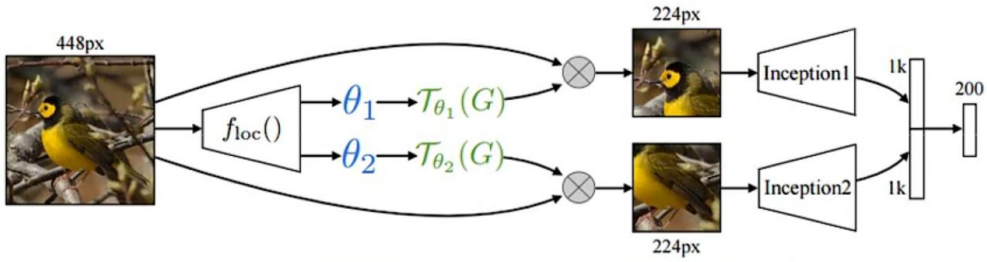


对于 Bird 的分类：

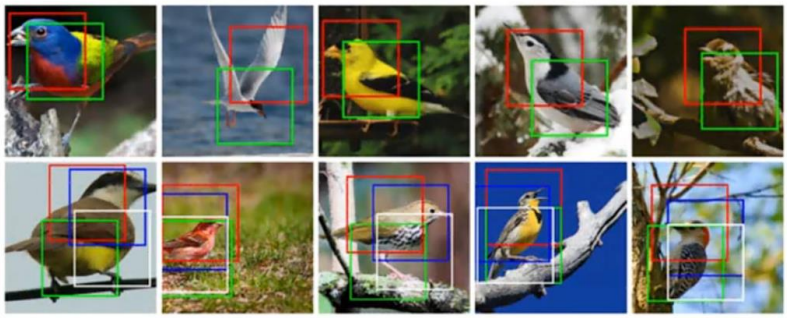
在确定图片最终输出的前面采用了 ST 技术（对于本题：红色区域更重的是收集鸟的嘴，绿色部分更多是收集支撑脚）

Brid Recognition

$$\begin{bmatrix} a & 0 \\ 0 & d \end{bmatrix} \begin{bmatrix} e \\ f \end{bmatrix}$$



Model	
Cimpoi '15 [4]	66.7
Zhang '14 [30]	74.9
Branson '14 [2]	75.7
Lin '15 [20]	80.9
Simon '15 [24]	81.0
CNN (ours) 224px	82.3
2×ST-CNN 224px	83.1
2×ST-CNN 448px	83.9
4×ST-CNN 448px	**84.1**



最终，可以识别 放大，缩小，旋转的 picture 是哪种鸟？更好的