Artificial Neural Networks
人工神经网络

权小军教授
中山大学计算机学院

quanxj3@mail.sysu.edu.cn

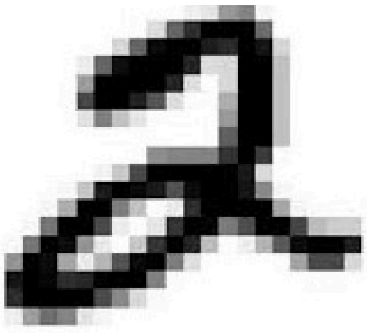
2023 年 4 月 21 日

Lecture 6 - Convolutional Neural Networks I

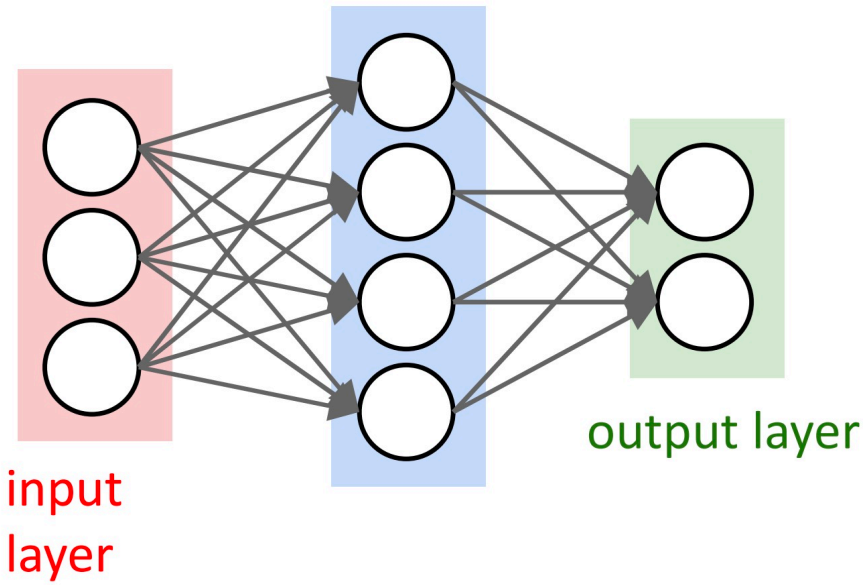
(*Part of the slides are adapted from CMU 11-785*)

Lecture 6.1 Shift Invariant 平移不变性

The model so far

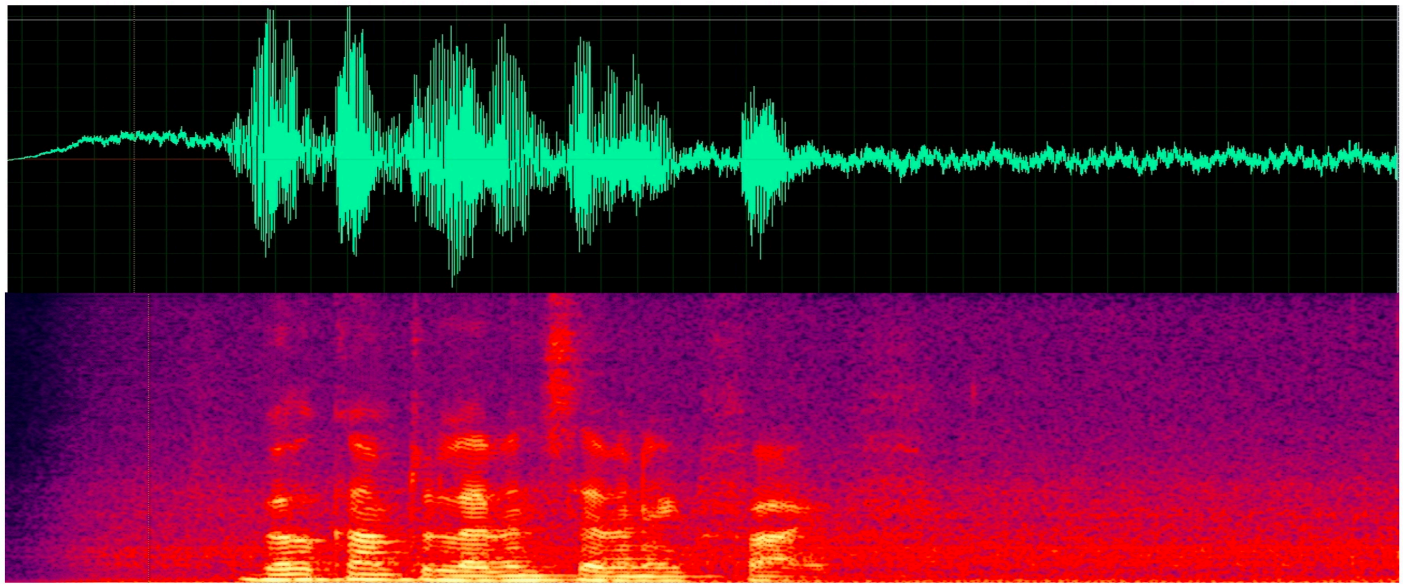


Or, more generally
a vector input



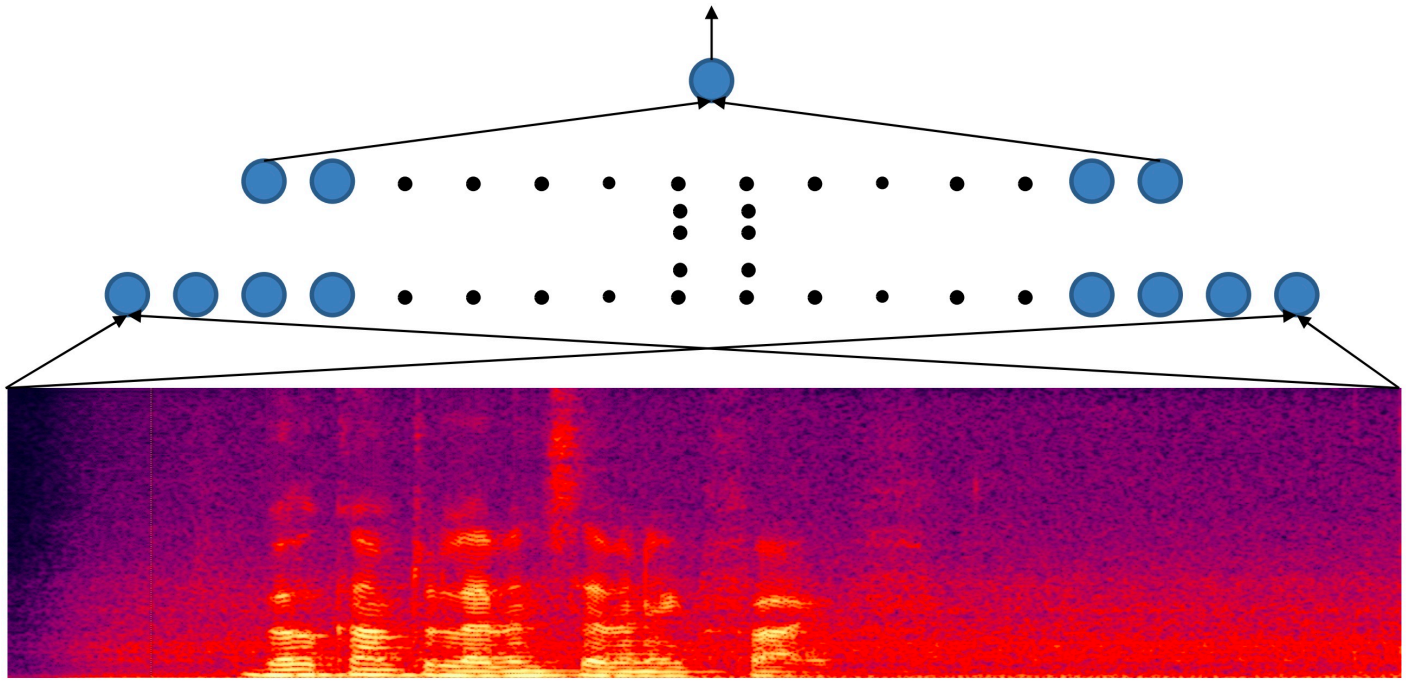
- Can recognize patterns in data
  - E.g. digits
  - Or any other vector data

A new problem



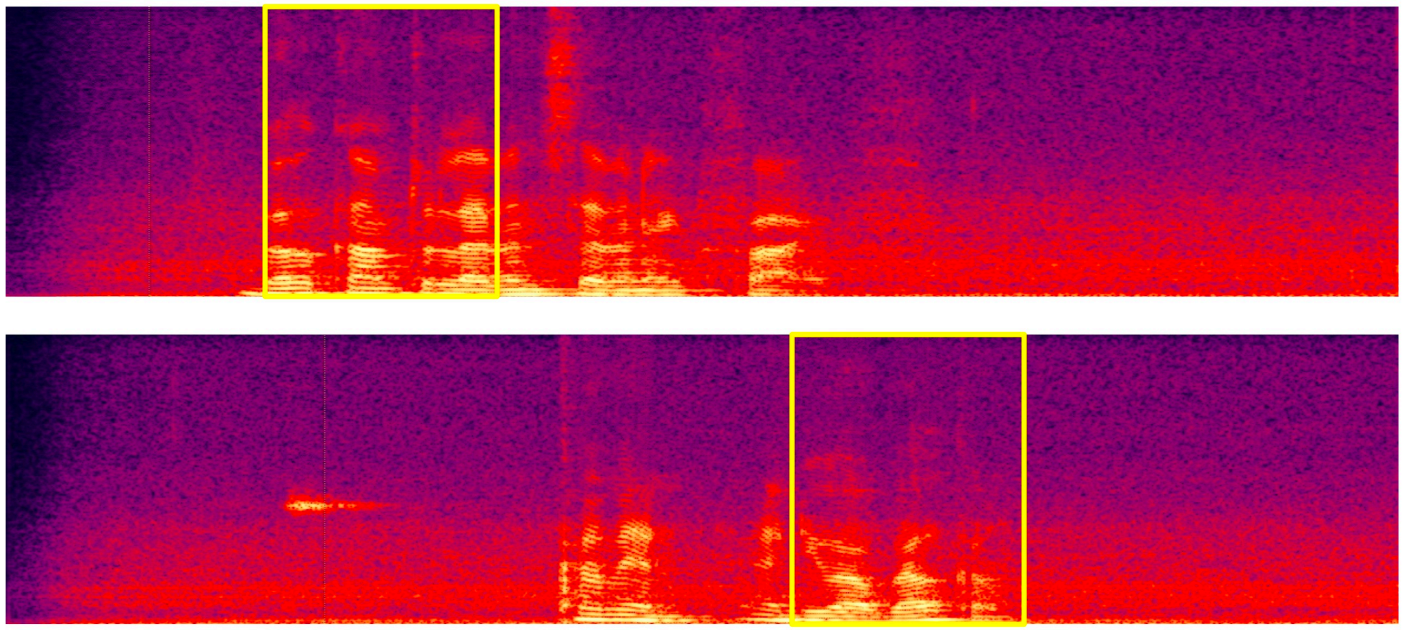
- Does this signal contain the word "Welcome"?
- Compose an MLP for this problem.
  - Assuming all recordings are exactly the same length..

Finding a Welcome



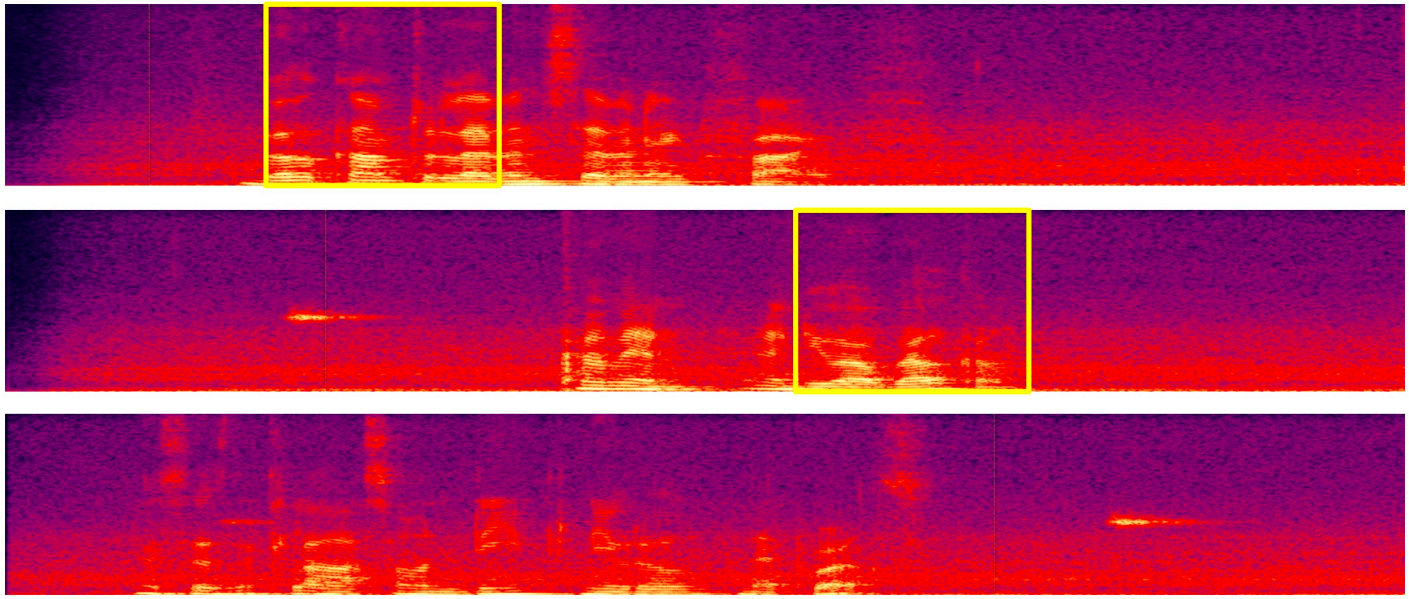
- Trivial solution: Train an MLP for the entire recording

Finding a Welcome



- Problem with trivial solution: Network that finds a "welcome" in the top recording will not find it in the lower one
  - Unless trained with both
  - Will require a very large network and a large amount of training data to cover every case

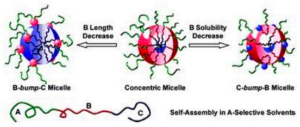
Finding a Welcome

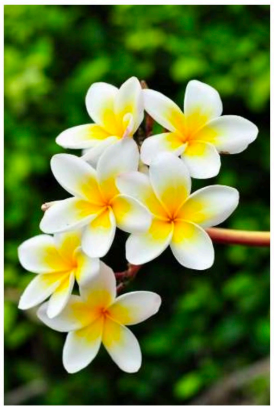


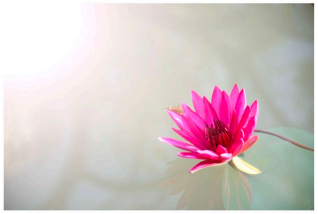
- Need a simple network that will fire regardless of the location of "Welcome"
  - and not fire when there is none

Another example: Flowers



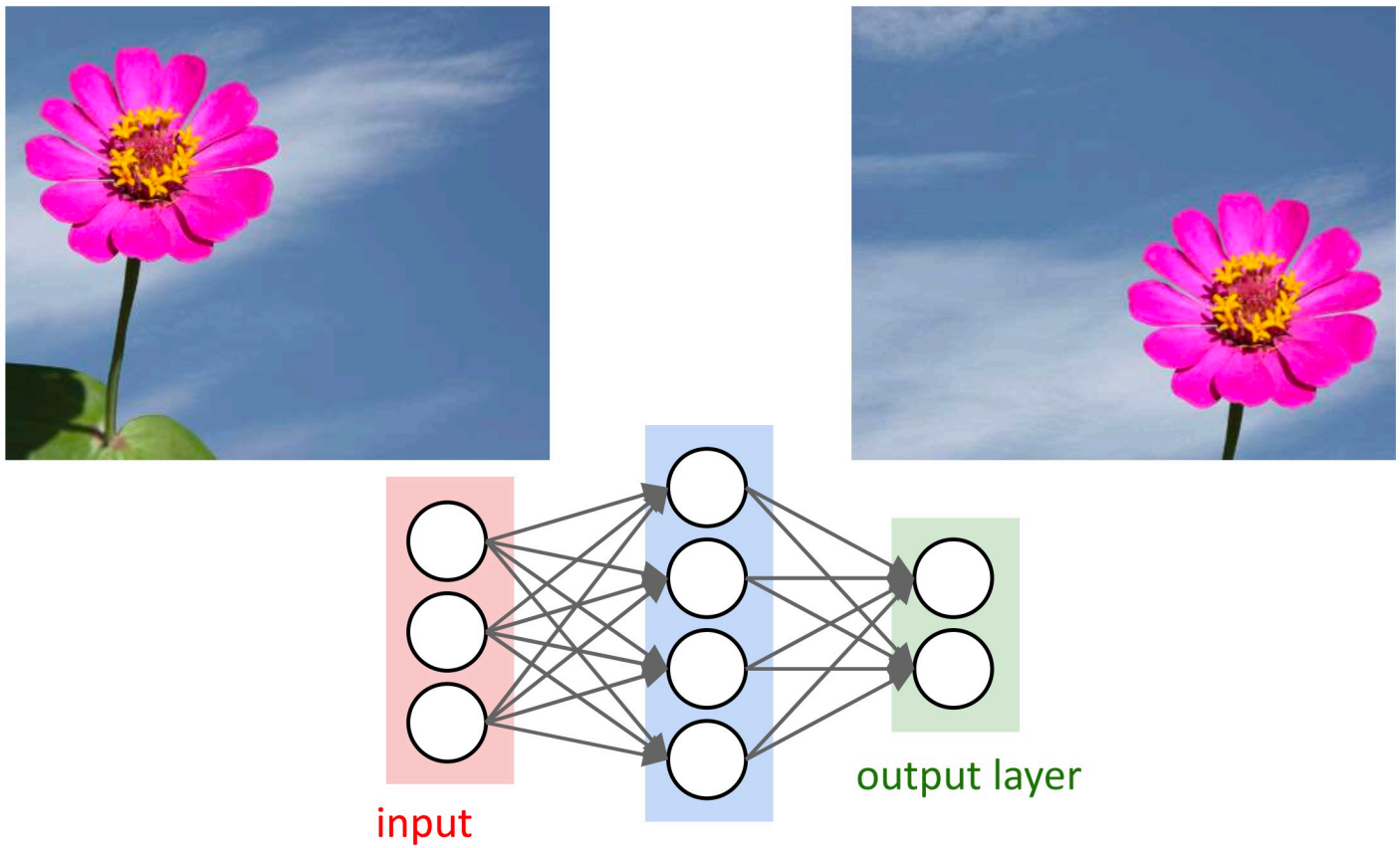





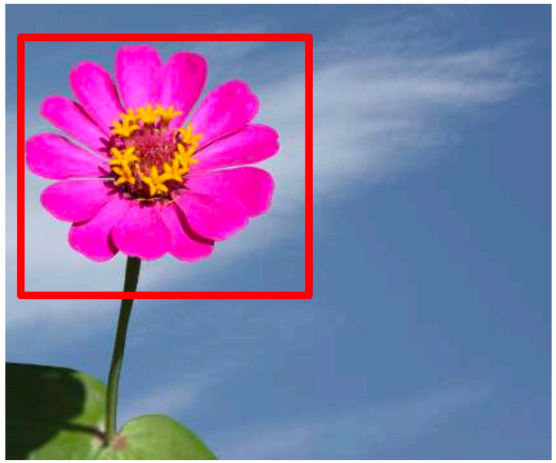
- Is there a flower in any of these images

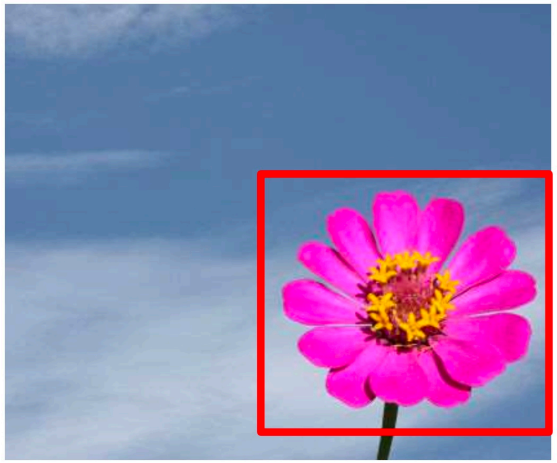
A problem with this example



- Will an MLP that recognizes the left image as a flower also recognize the one on the right as a flower?

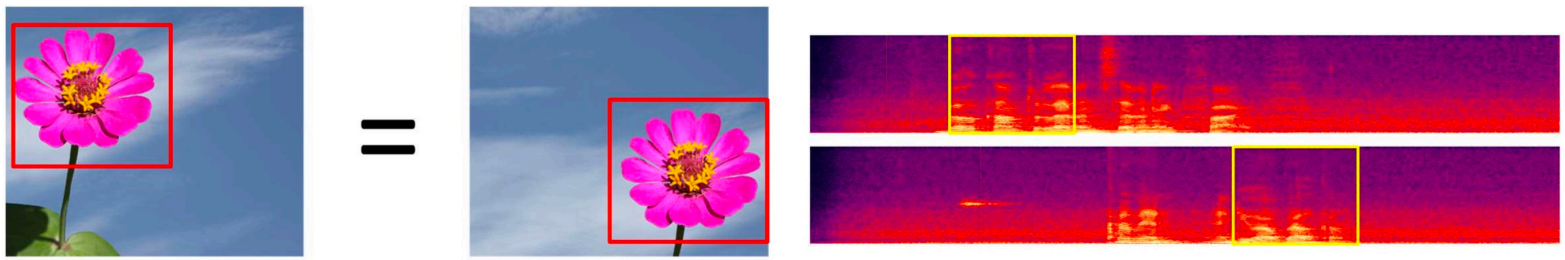
A problem with this example





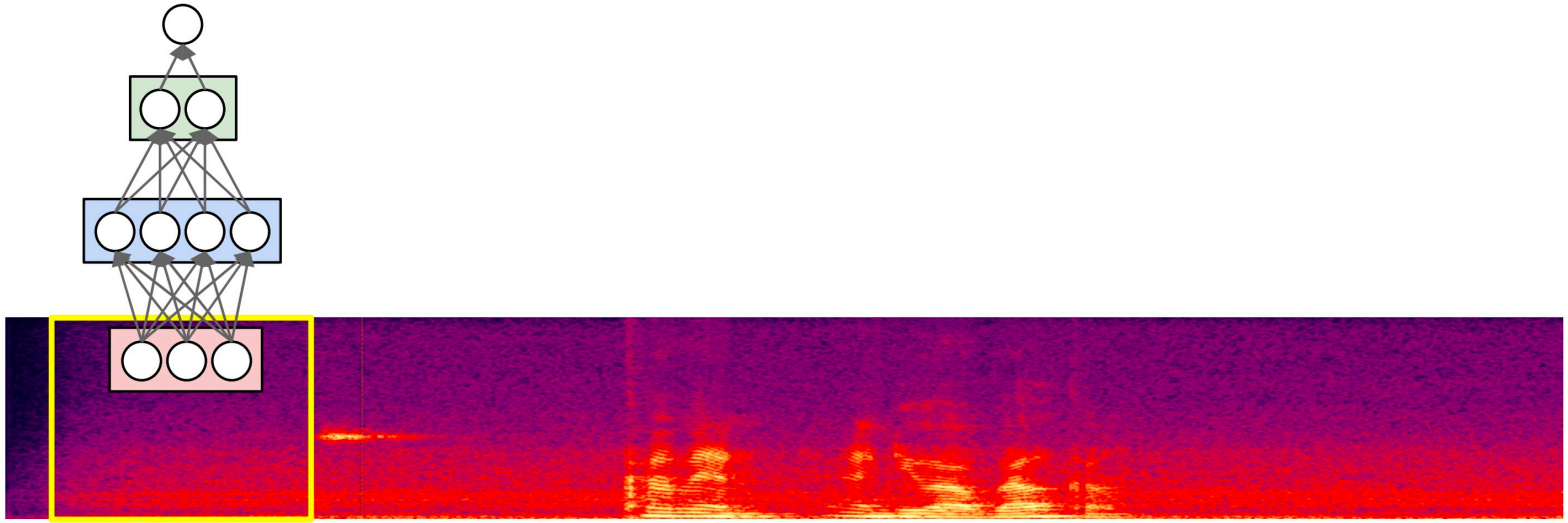
- Need a network that will "fire" regardless of the precise location of the target object

The need for shift invariance 平移不变性



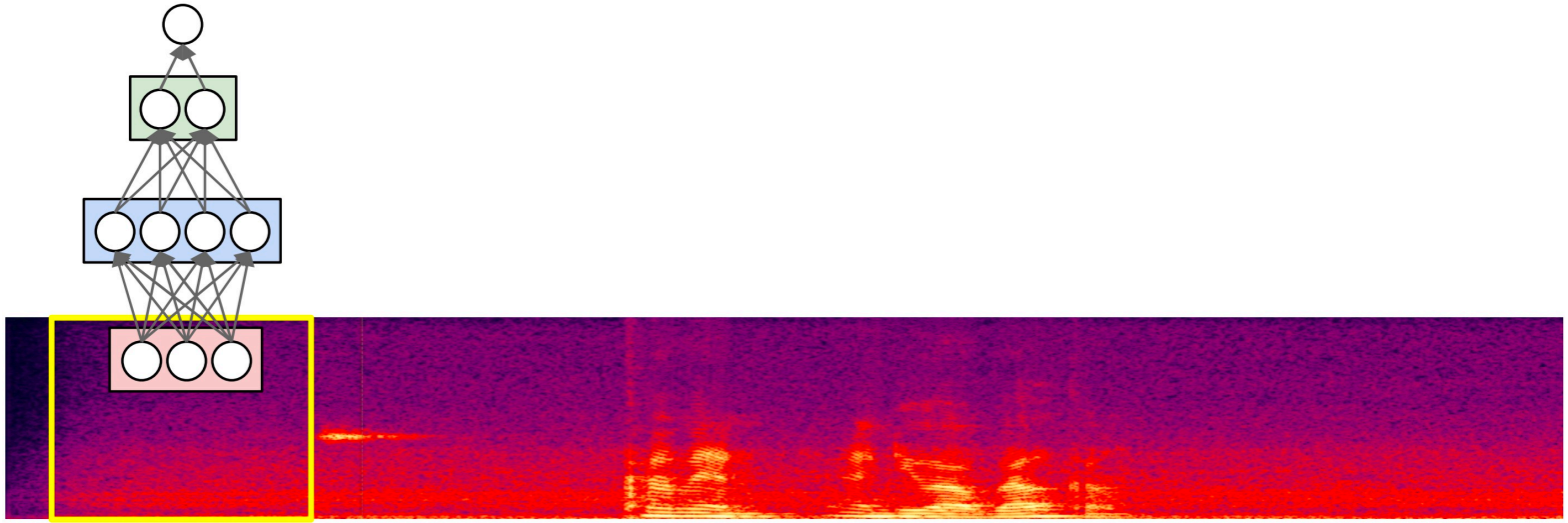
- In many problems the location of a pattern is not important
  - Only the presence of the pattern
- MLPs are sensitive to the location of the pattern
  - Moving it by one component results in an entirely different input that the MLP won't recognize
- Requirement: Network must be shift invariant (平移不变性)

Solution: Scan



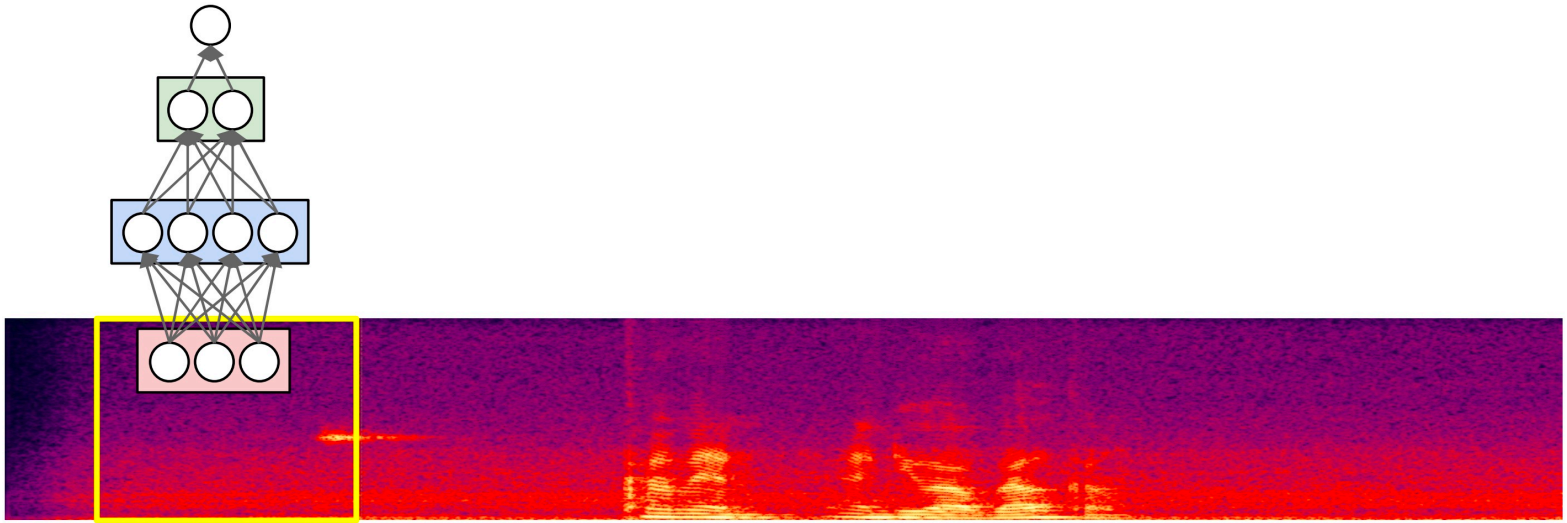
- Scan for the target word
  - The spectral time-frequency (频谱时频) components in a "window" are input to a "welcome-detector" MLP

Solution: Scan



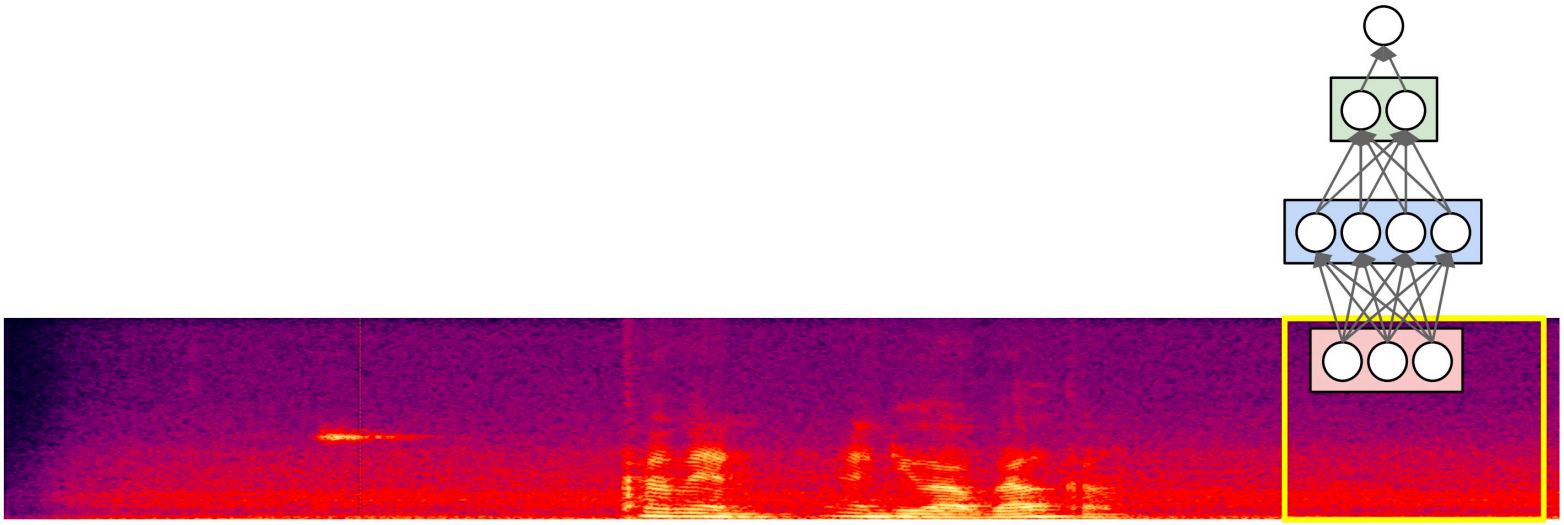
- Scan for the target word
  - The spectral time-frequency (频谱时频) components in a "window" are input to a "welcome-detector" MLP

Solution: Scan



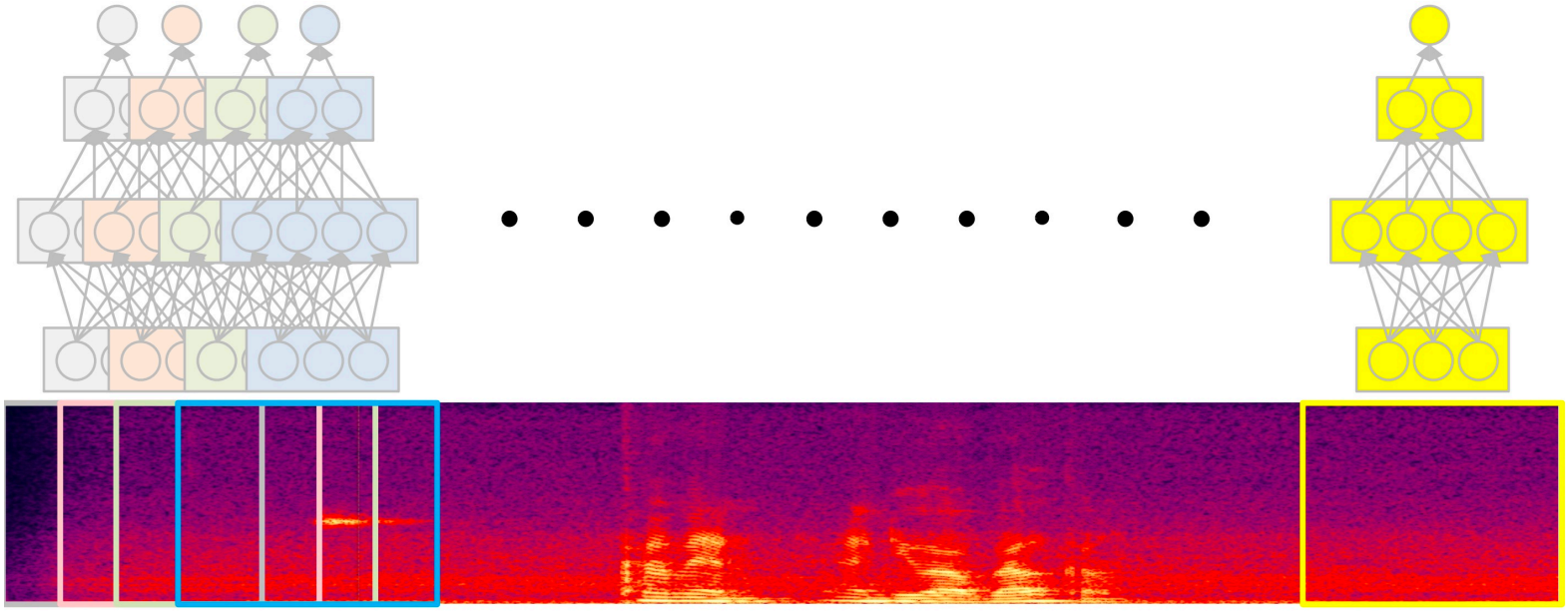
- Scan for the target word
  - The spectral time-frequency components (频谱时频) in a "window" are input to a "welcome-detector" MLP

Solution: Scan



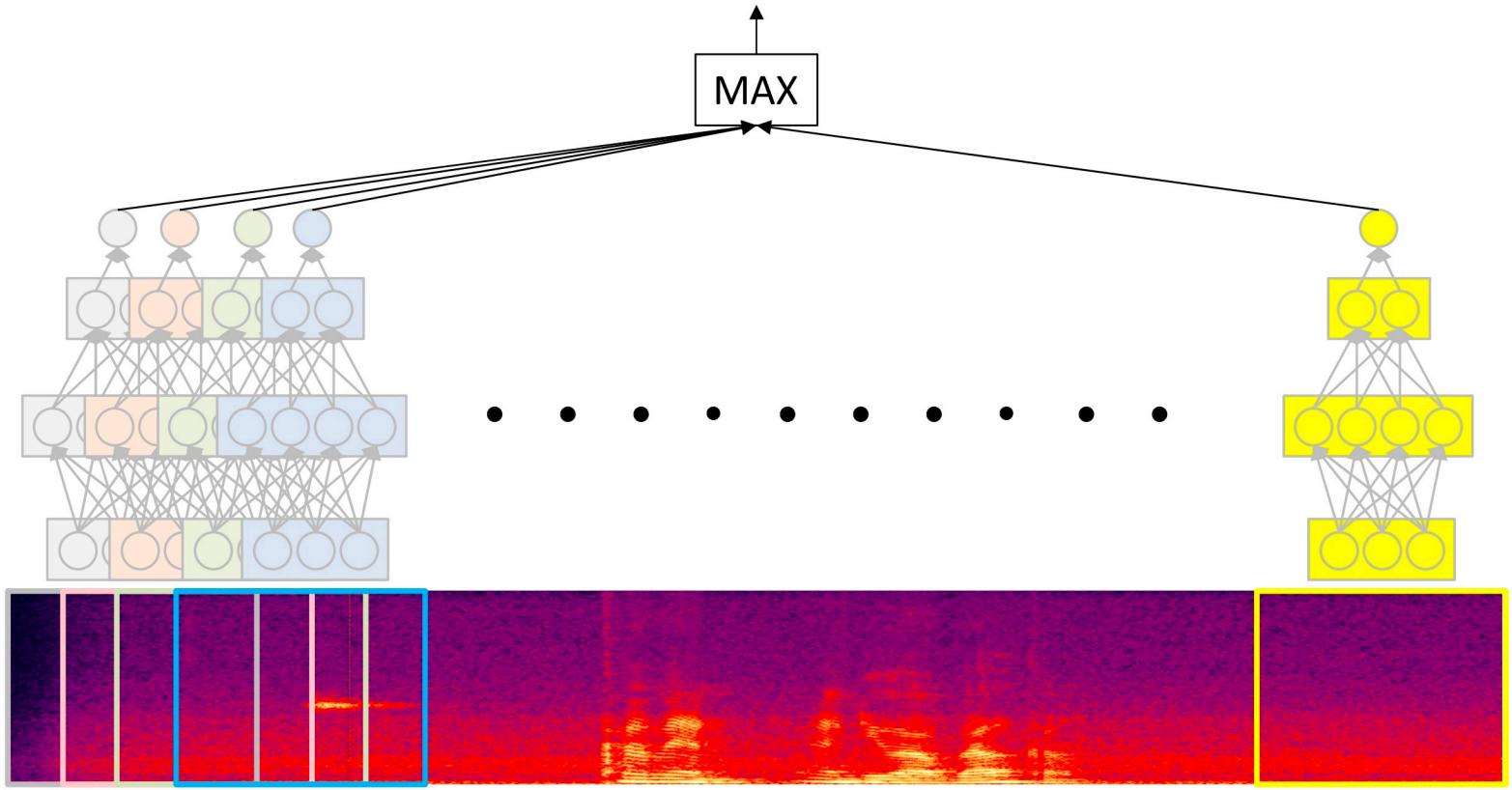
- Scan for the target word
  - The spectral time-frequency (频谱时频) components in a "window" are input to a "welcome-detector" MLP

Solution: Scan



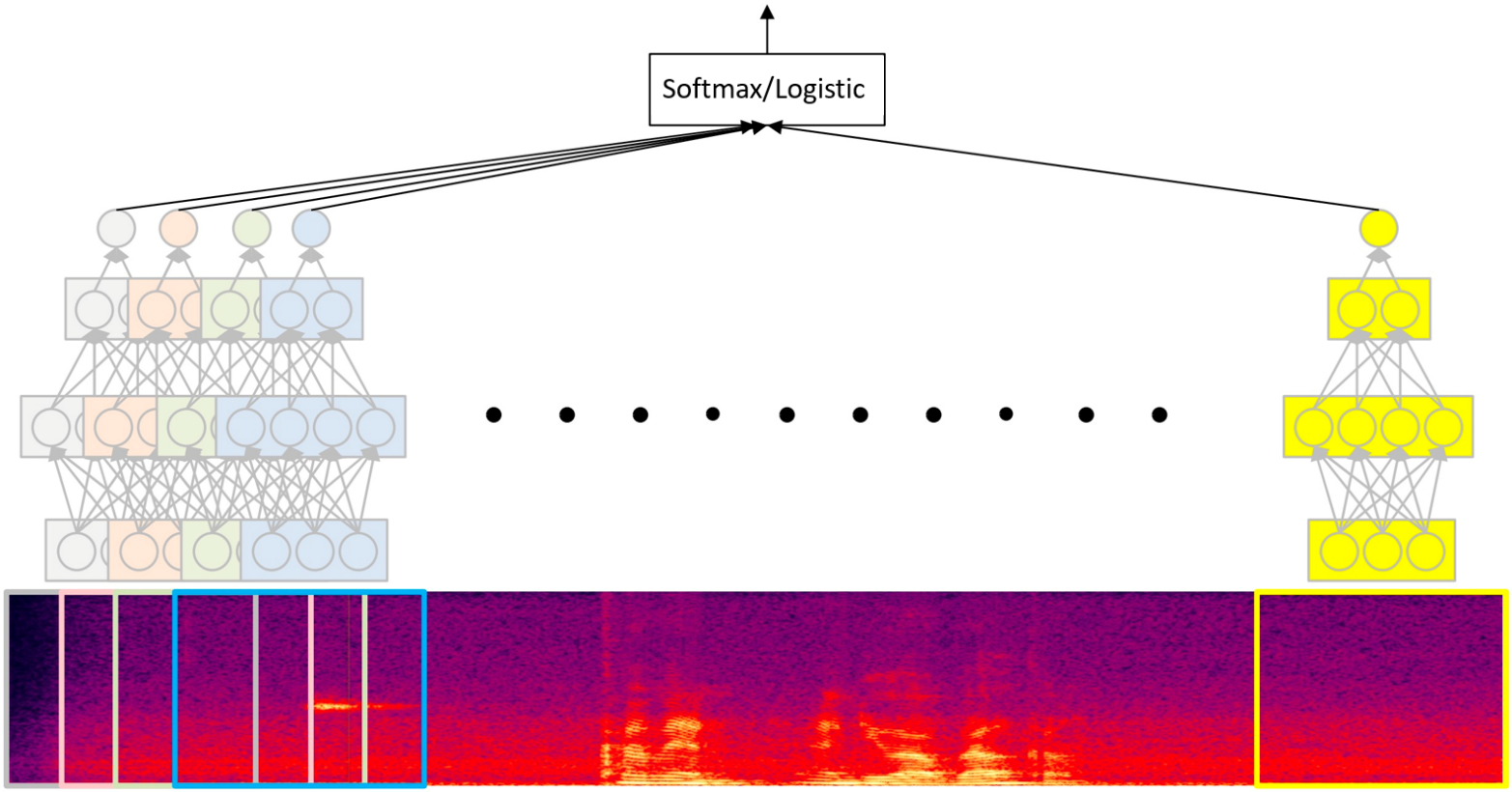
- "Does welcome occur in this recording?"
  - We have classified many "windows" individually
  - "Welcome" may have occurred in any of them

Solution: Scan



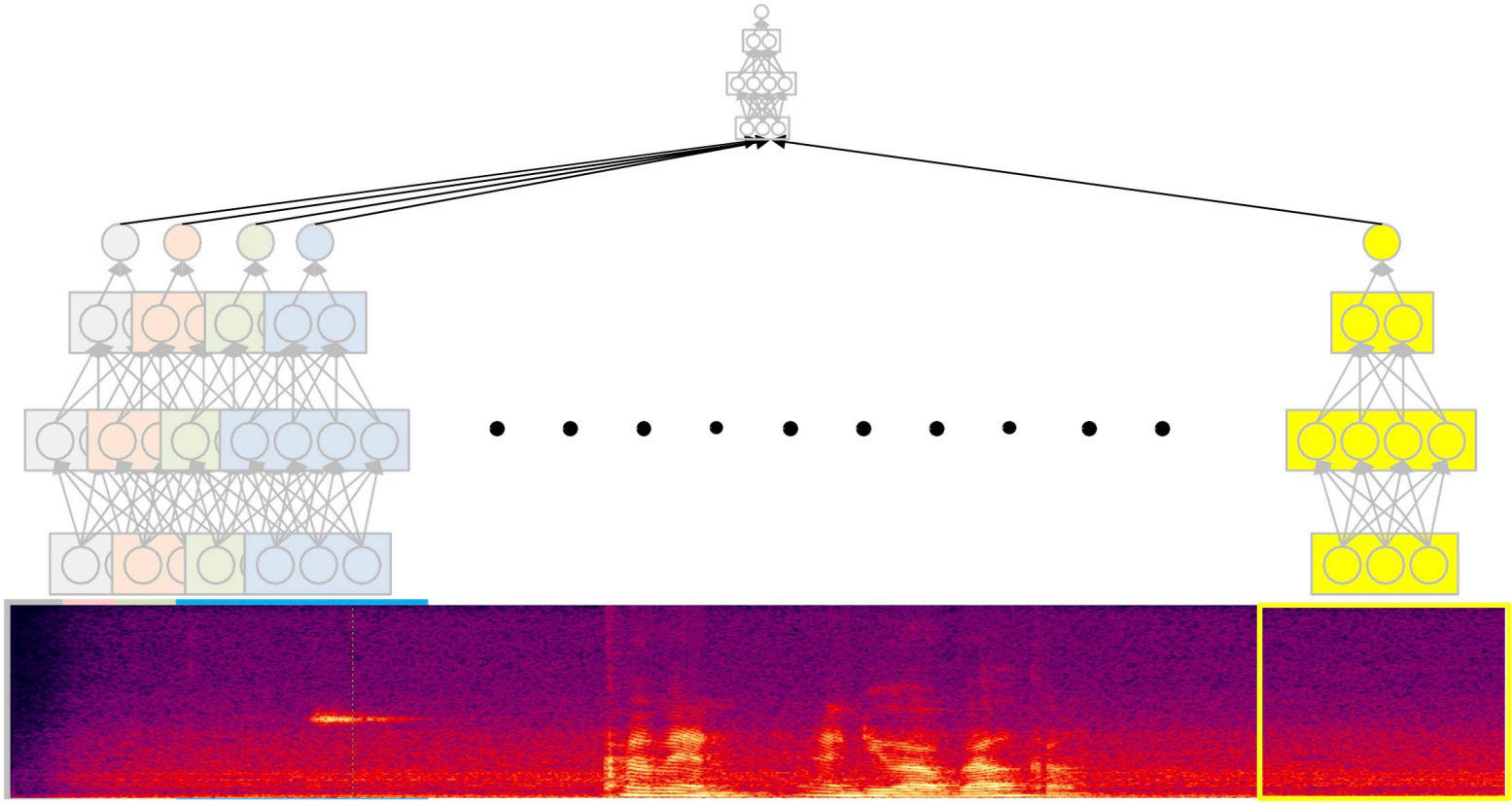
- "Does welcome occur in this recording?"
  - Maximum of all the outputs

Solution: Scan



- "Does welcome occur in this recording?"
  - Maximum of all the outputs
  - Or a proper softmax/logistic

Solution: Scan



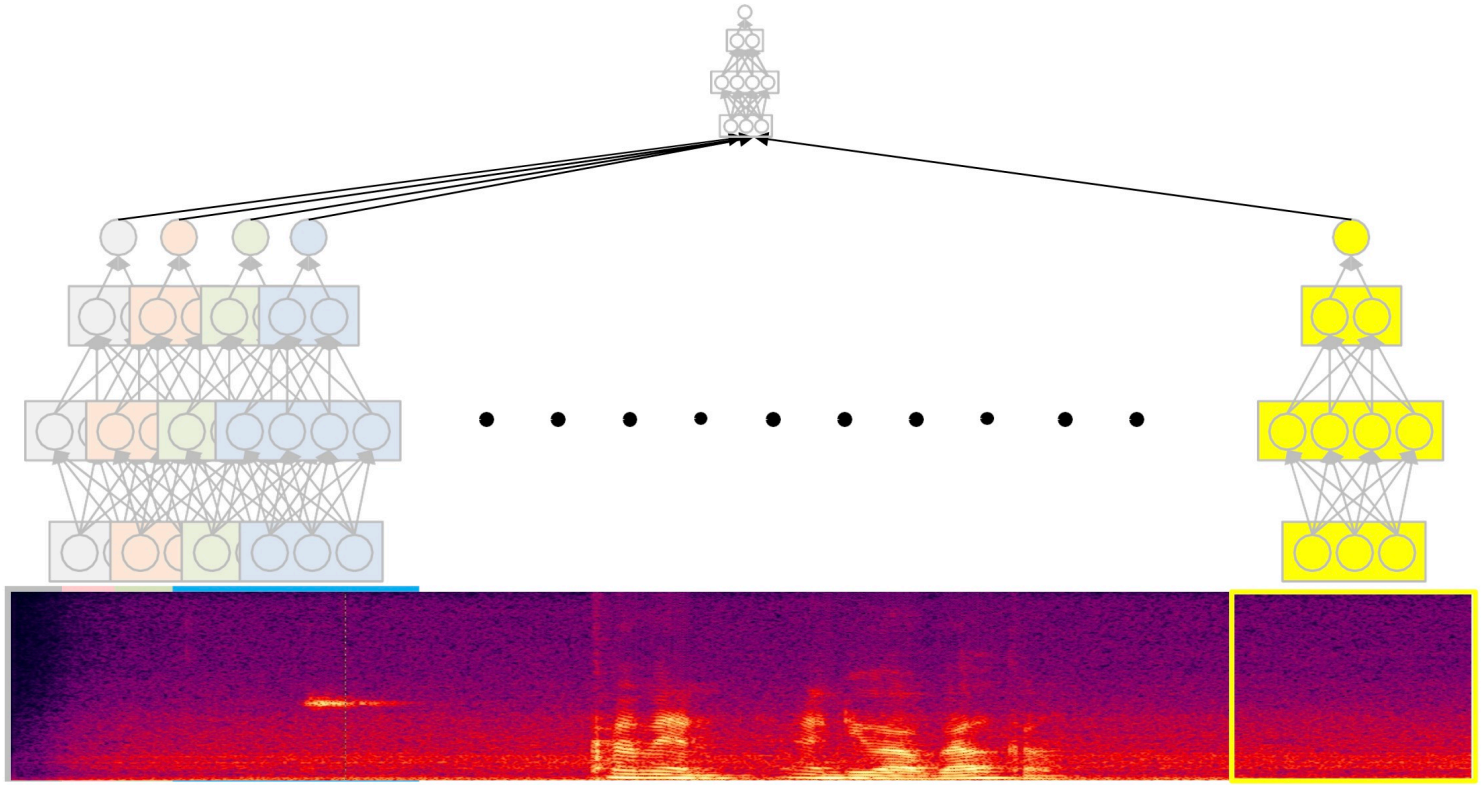
- "Does welcome occur in this recording?"
  - Maximum of all the outputs
  - Or a proper softmax/logistic
    - Adjacent windows can combine their evidence
  - Or even an MLP

Scanning with an MLP

- `K` = width of "patch" evaluated by MLP

```
For t = 1:T-K+1
   XSegment = x(:, t:t+K-1)
   y(t) = MLP(XSegment)

Y = softmax(y(1)..y(T-K+1))
```

Its actually just one giant network



- The entire operation can be viewed as one giant network
  - With many subnetworks, one per window
  - Restriction: All subnets are identical
- The network is shift-invariant (平移不变性)!

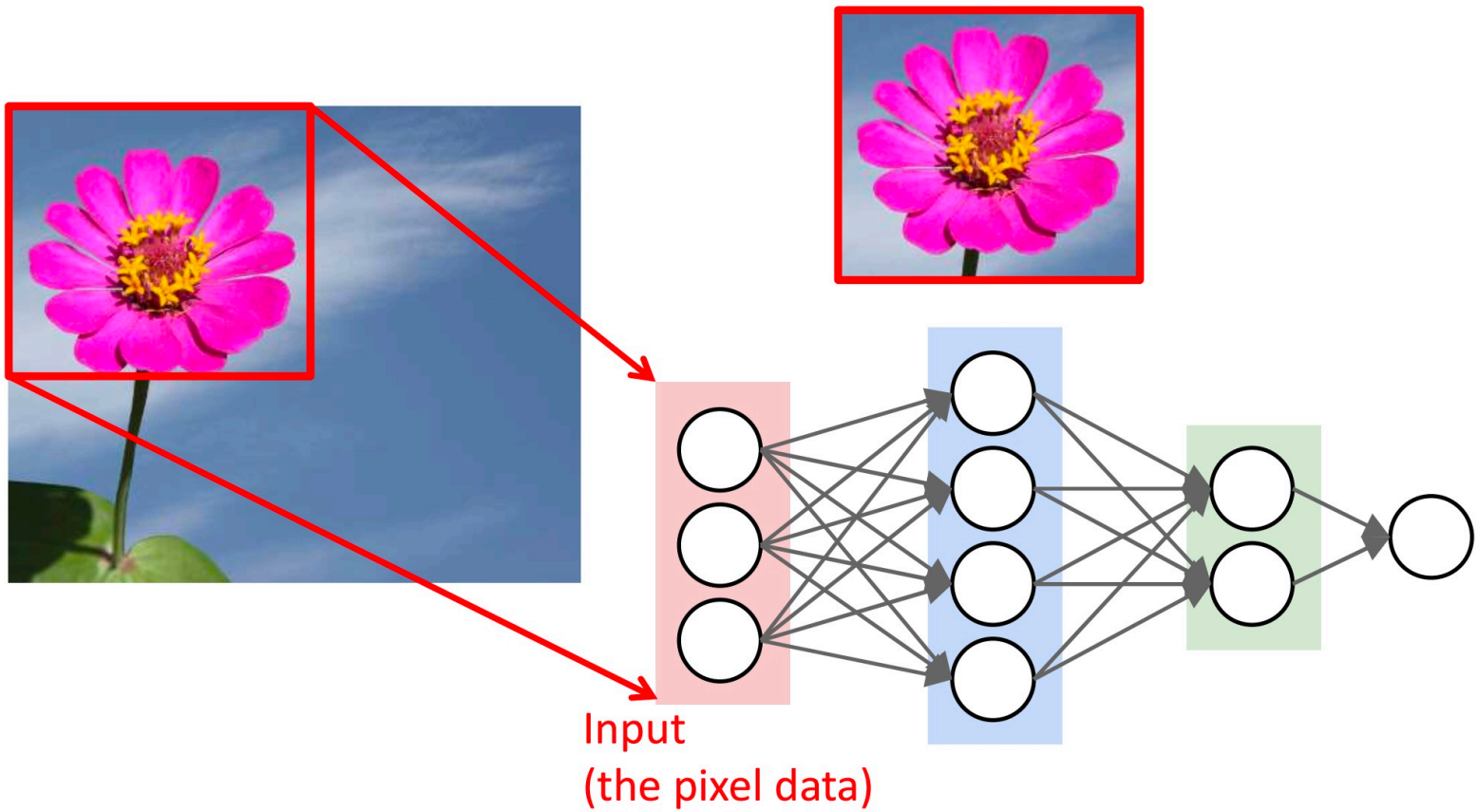
Scanning with an MLP

- `K` = width of "patch" evaluated by MLP

```
For t = 1:T-K+1
   XSegment = x(:, t:t+K-1)
   y(t) = MLP(XSegment)

Y = softmax(y(1)..y(T-K+1))
```

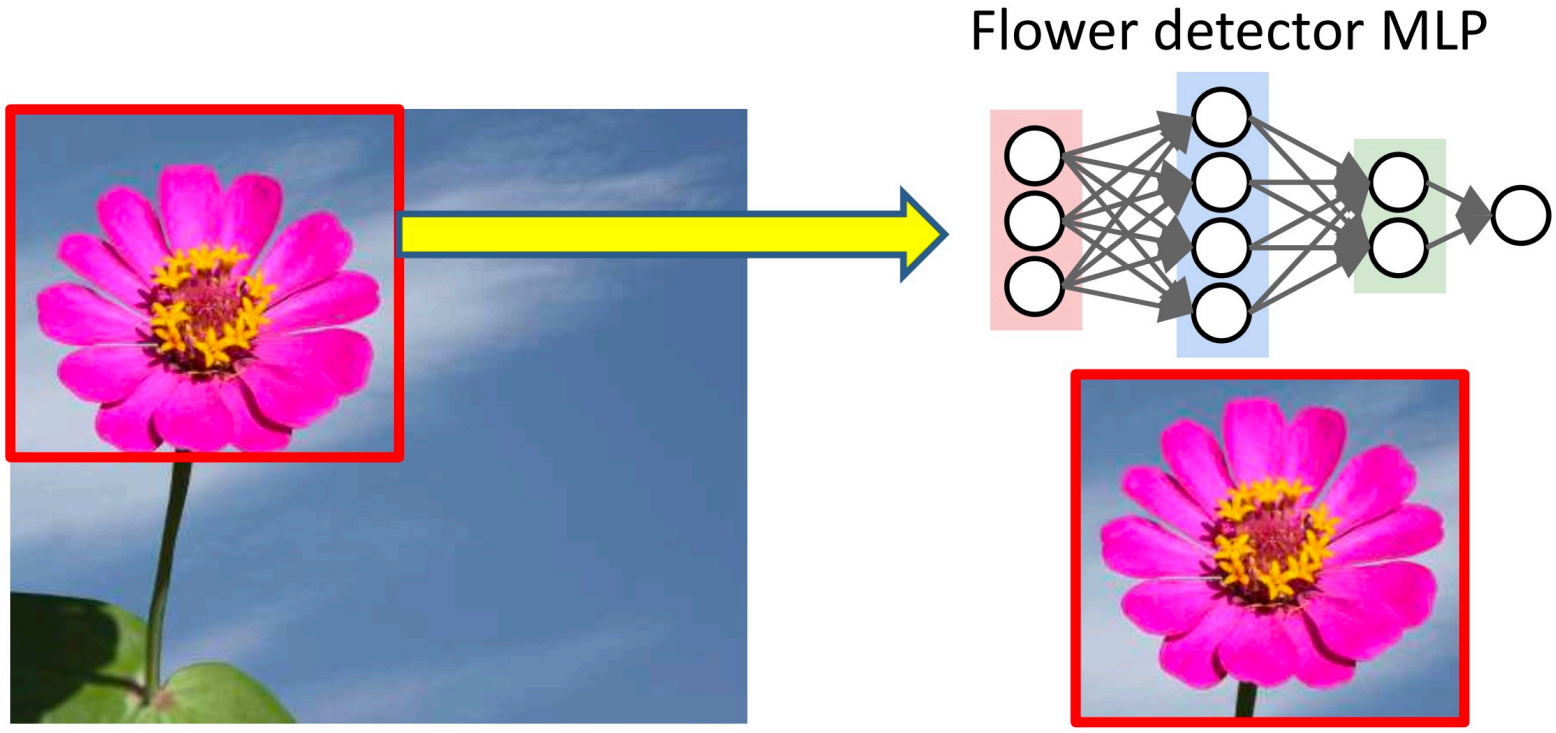
Just the final layer of the overall MLP

The 2-d example: Does this picture have a flower?



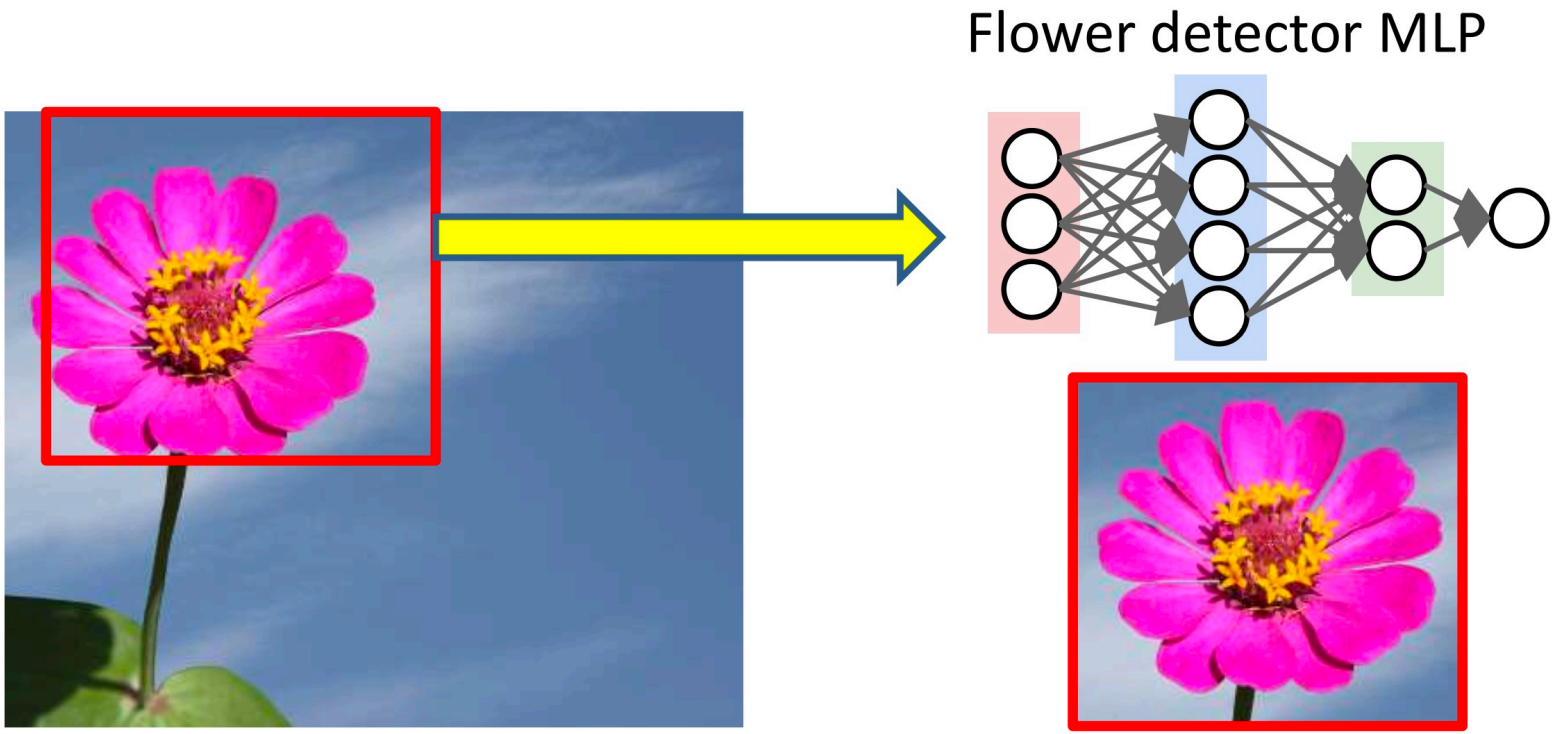
- Scan for the desired object
  - "Look" for the target object at each position

Solution: Scan



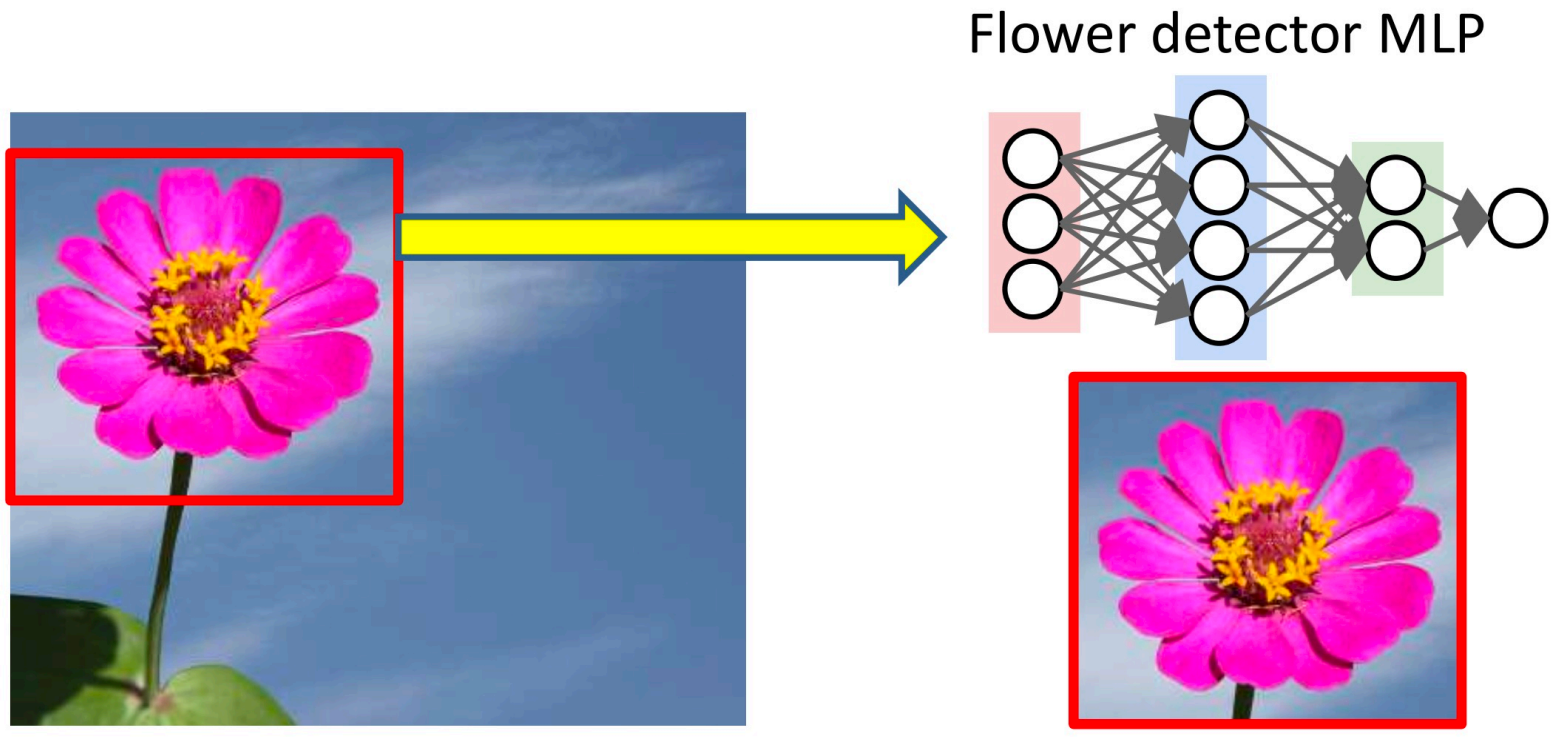
- Scan for the desired object
  - At each location, the entire region is sent through the MLP

Solution: Scan



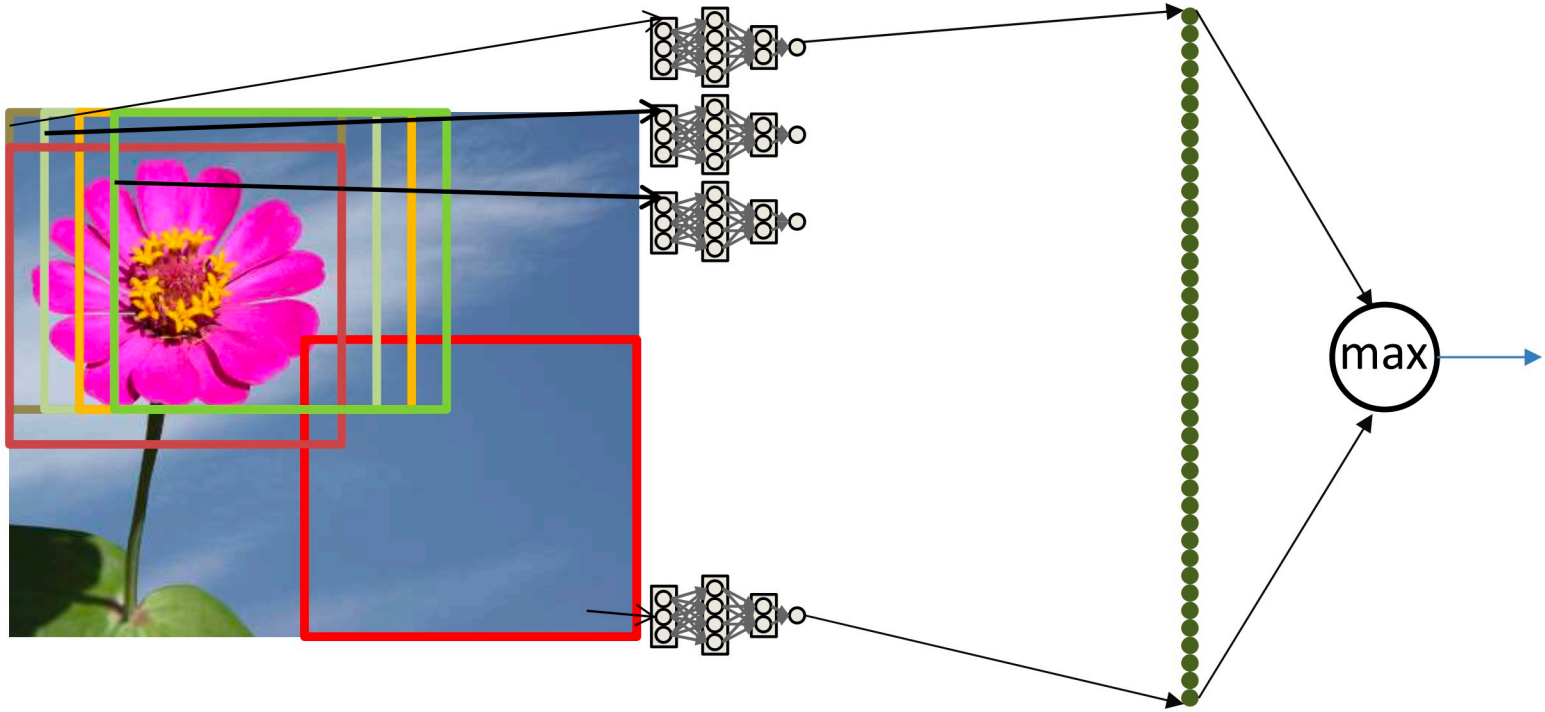
- Scan for the desired object
  - At each location, the entire region is sent through the MLP

Solution: Scan



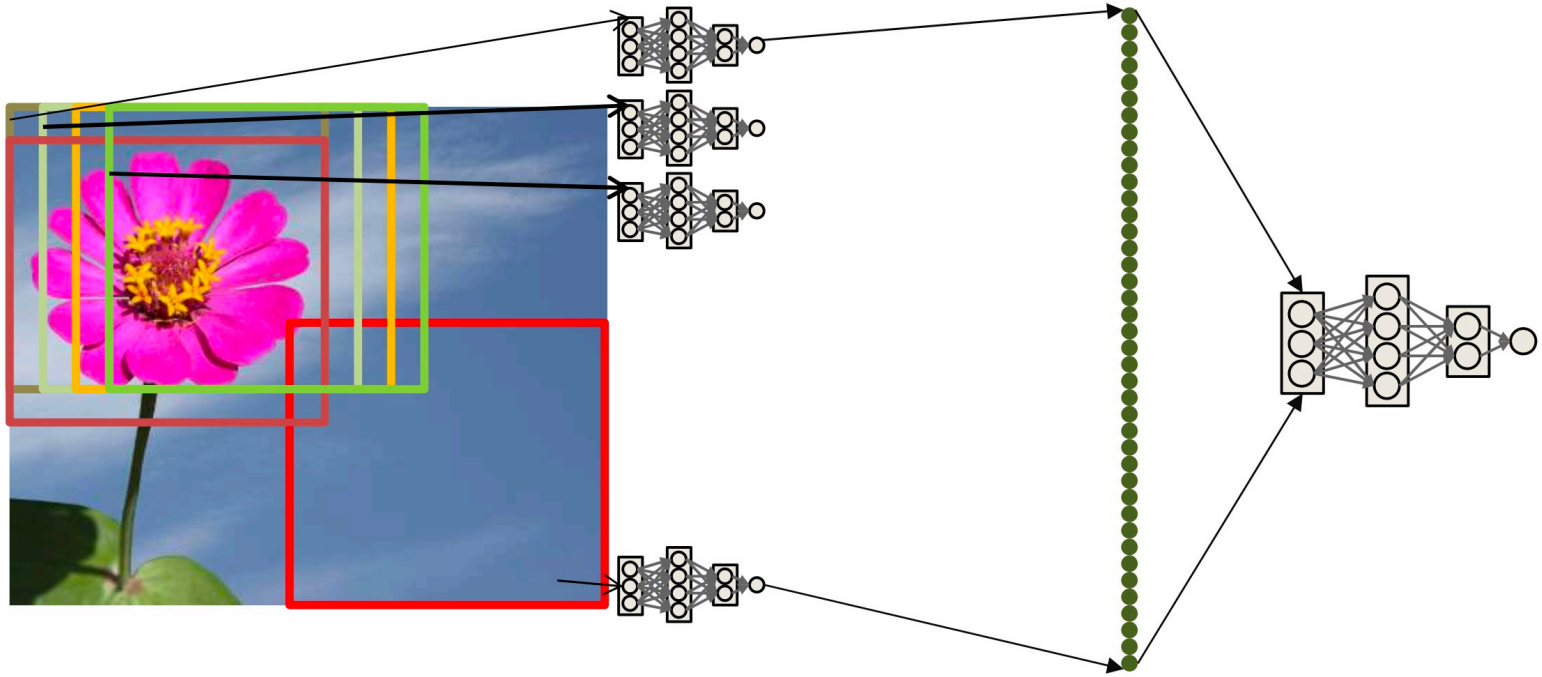
- Scan for the desired object
  - At each location, the entire region is sent through the MLP

Scanning the picture to find a flower



- Determine if any of the locations had a flower
  - We get one classification output per scanned location
    - Each dot in the right represents the output of the MLP
  - Look at the maximum value
    - If the picture has a flower, the location with the flower will result in high output value

Scanning the picture to find a flower



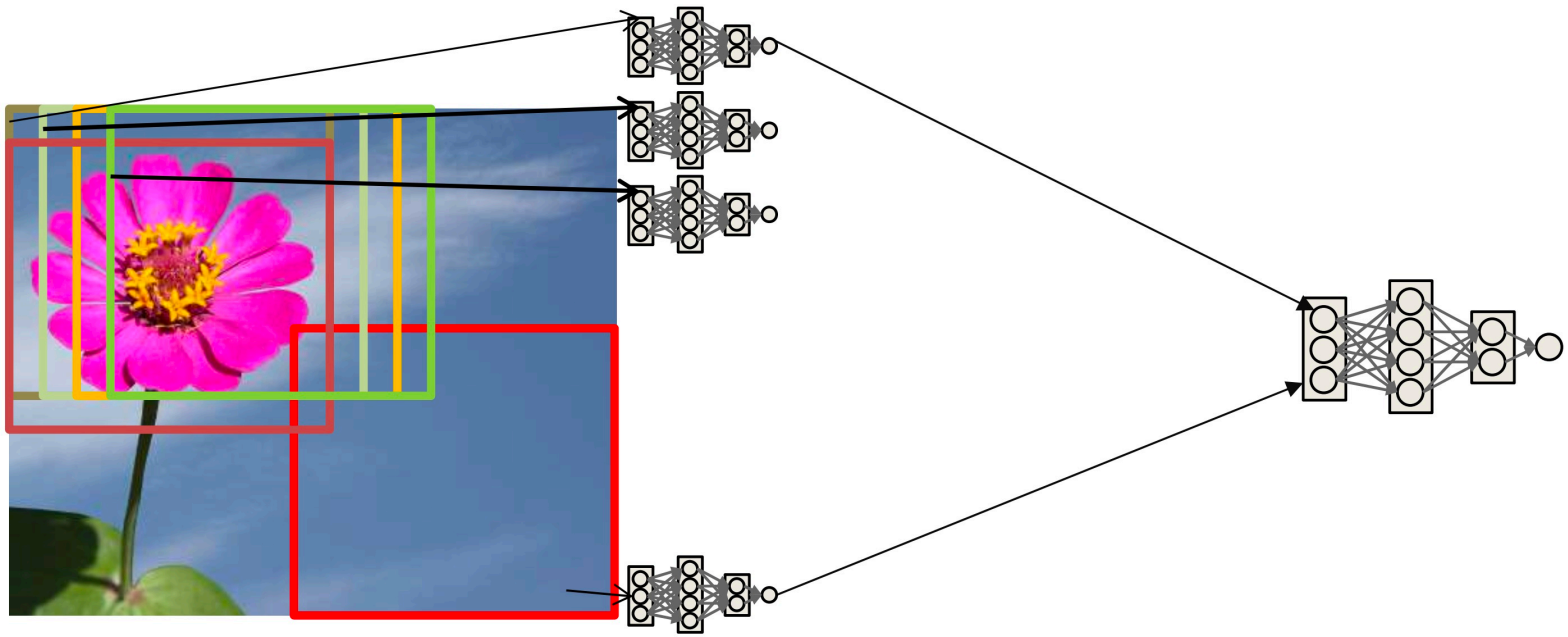
- Determine if any of the locations had a flower
  - We get one classification output per scanned location
    - Each dot in the right represents the output of the MLP when it classifies one location in the input figure
  - Look at the maximum value
  - Or pass it through a softmax or even an MLP

Scanning with an MLP

- `KxK` = size of "patch" evaluated by MLP
- `W` is width of image
- `H` is height of image

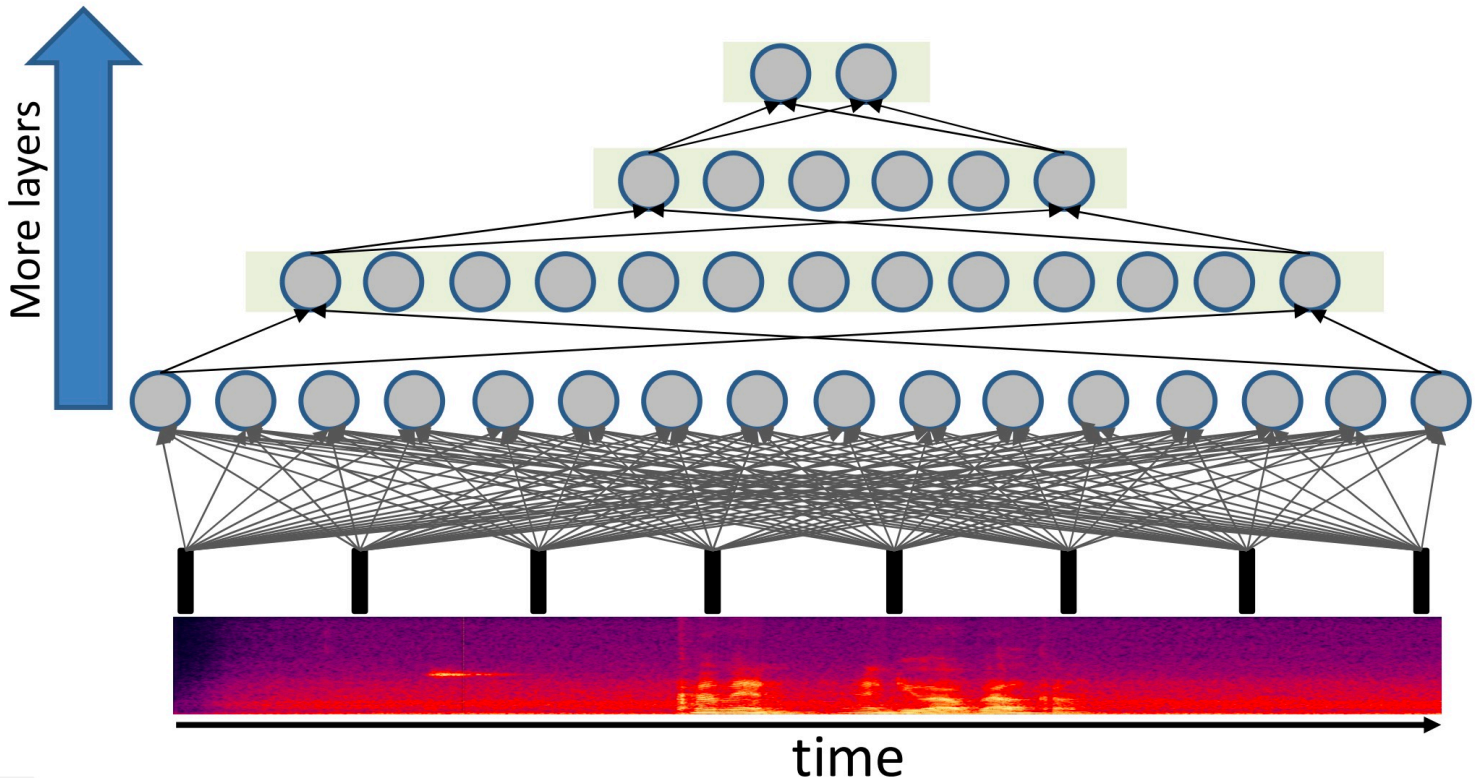
```
For i = 1:W-K+1
    For j = 1:H-K+1
        ImgSegment = Img(i:i+K-1, j:j+K-1)
        y(i,j) = MLP(ImgSegment)

Y = softmax( y(1,1)..y(W-K+1,H-K+1) )
```

Its just a giant network with common subnets



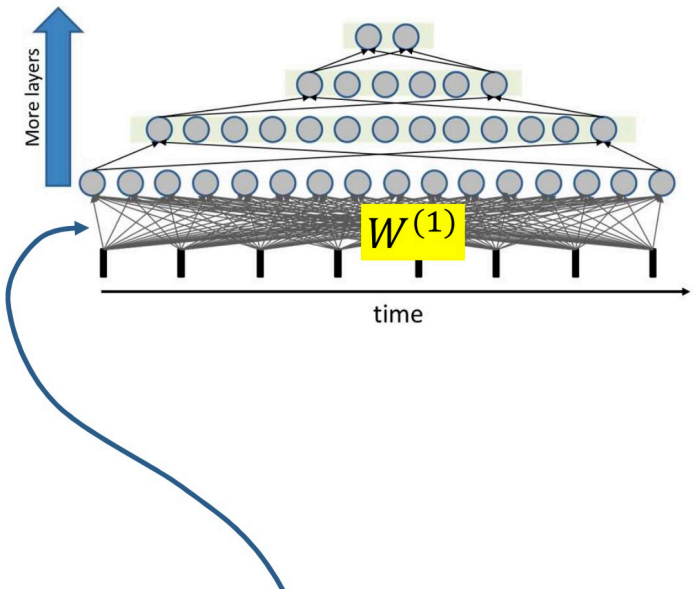
- The entire operation can be viewed as a single giant network
  - Composed of many "subnets" (one per window)
  - With one key feature: all subnets are identical
- The network is shift invariant (平移不变性)

Regular networks vs. scanning networks



- In a regular MLP every neuron in a layer is connected by a unique weight to every unit in the previous layer
  - All entries in the weight matrix are unique
  - The weight matrix is (generally) full

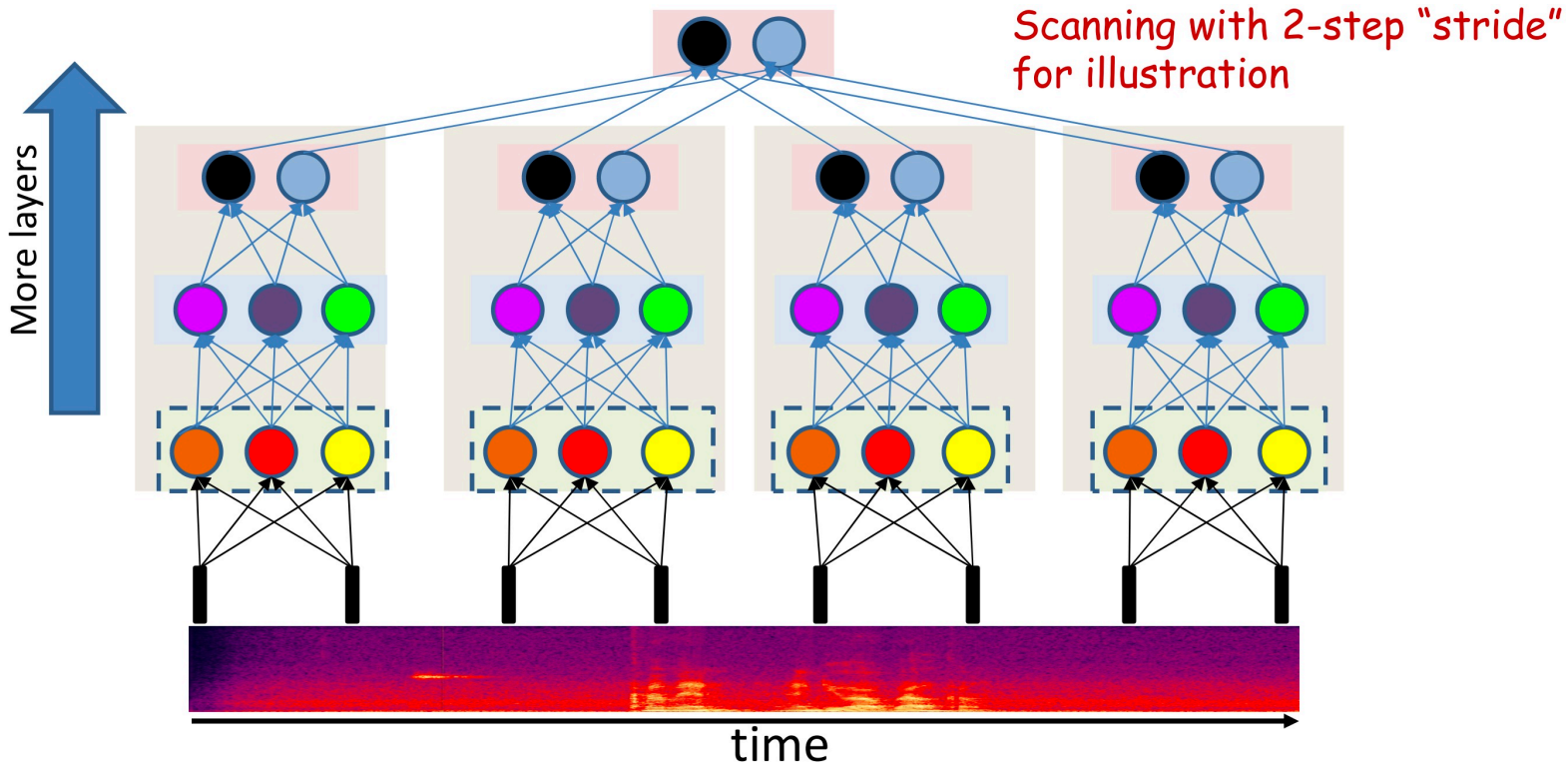
Regular network



$$W^{(1)} = \begin{bmatrix} w_{11} & w_{21} & w_{31} & w_{41} & \cdots & \cdots & \cdots & w_{N1} \\ w_{12} & w_{22} & w_{32} & w_{42} & \cdots & \cdots & \cdots & w_{N2} \\ w_{13} & w_{23} & w_{33} & w_{43} & \cdots & \cdots & \cdots & w_{N3} \\ w_{14} & w_{24} & w_{34} & w_{44} & \cdots & \cdots & \cdots & w_{N4} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ w_{1M} & w_{2M} & w_{3M} & w_{4M} & \cdots & \cdots & \cdots & w_{NM} \end{bmatrix}$$

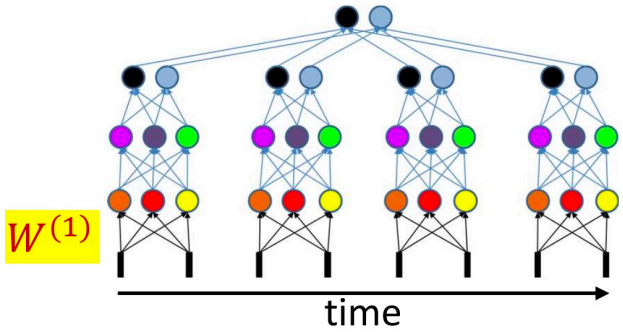
- Consider the first layer
  - Assume N inputs and M outputs
- The weights matrix is a full $N \times M$ matrix
  - Requiring MN unique parameters

Scanning networks



- In a scanning MLP each neuron is connected to a subset of neurons in the previous layer
  - The weights matrix is sparse
  - The weights matrix is block structured with identical blocks

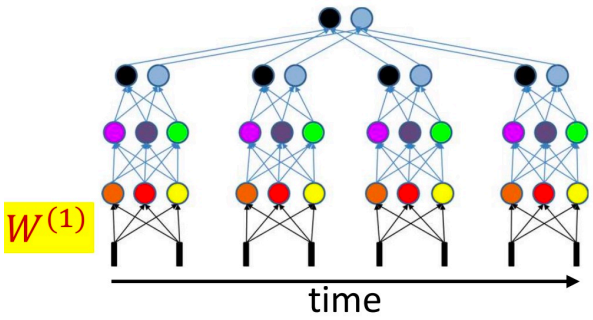
Scanning networks



$$W^{(1)} = \begin{bmatrix} w_{11} & w_{12} & 0 & 0 & 0 & \cdots & \cdots & \cdots & 0 \\ w_{21} & w_{22} & 0 & 0 & 0 & \cdots & \cdots & \cdots & 0 \\ w_{31} & w_{32} & 0 & 0 & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & 0 & w_{11} & w_{12} & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & w_{21} & w_{22} & 0 & 0 & 0 & \vdots & \vdots \\ 0 & 0 & w_{31} & w_{32} & 0 & 0 & 0 & \vdots & \vdots \\ 0 & 0 & 0 & 0 & w_{11} & w_{12} & 0 & 0 & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & w_{31} & w_{32} \end{bmatrix}$$

- In a scanning MLP each neuron is connected to a subset of neurons in the previous layer
  - The weights matrix is sparse
  - The weights matrix is block structured with identical blocks
  - The network is a shared parameter model

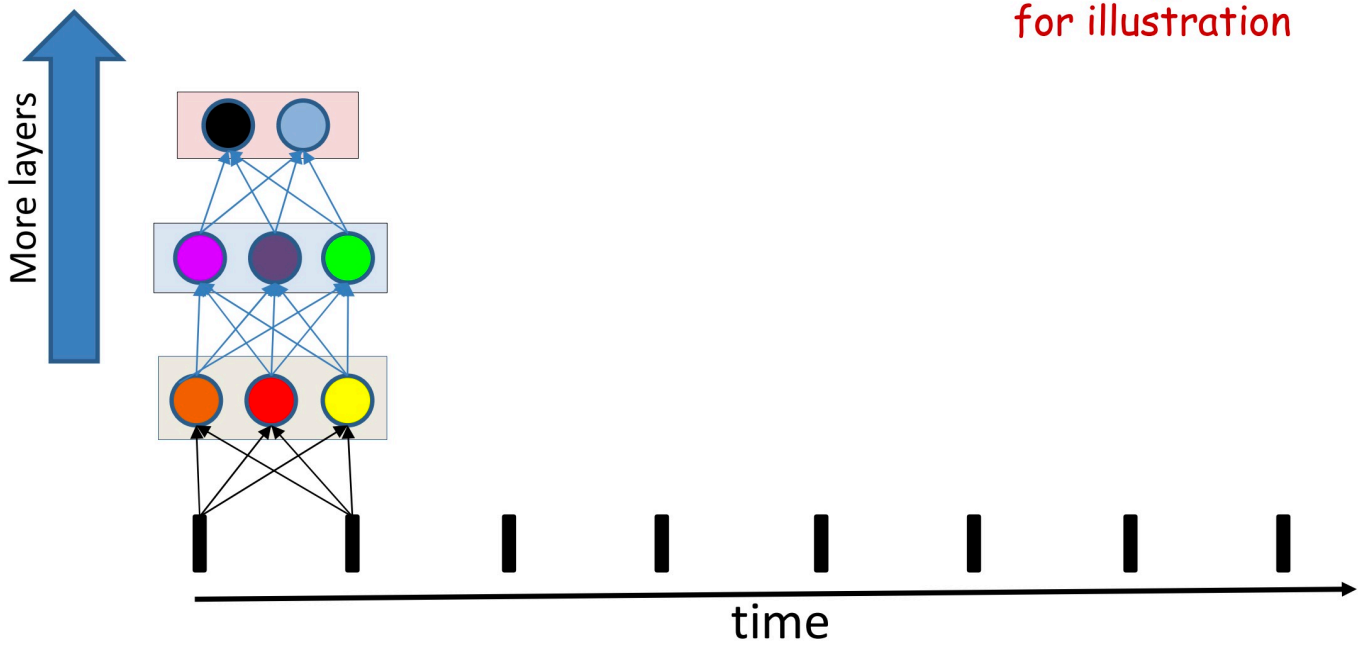
Scanning networks



$$W^{(1)} = \begin{bmatrix} w_{11} & w_{12} & 0 & 0 & 0 & \cdots & \cdots & \cdots & 0 \\ w_{21} & w_{22} & 0 & 0 & 0 & \cdots & \cdots & \cdots & 0 \\ w_{31} & w_{32} & 0 & 0 & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & 0 & w_{11} & w_{12} & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & w_{21} & w_{22} & 0 & 0 & 0 & \vdots & \vdots \\ 0 & 0 & w_{31} & w_{32} & 0 & 0 & 0 & \vdots & \vdots \\ 0 & 0 & 0 & 0 & w_{11} & w_{12} & 0 & 0 & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & w_{31} & w_{32} \end{bmatrix}$$

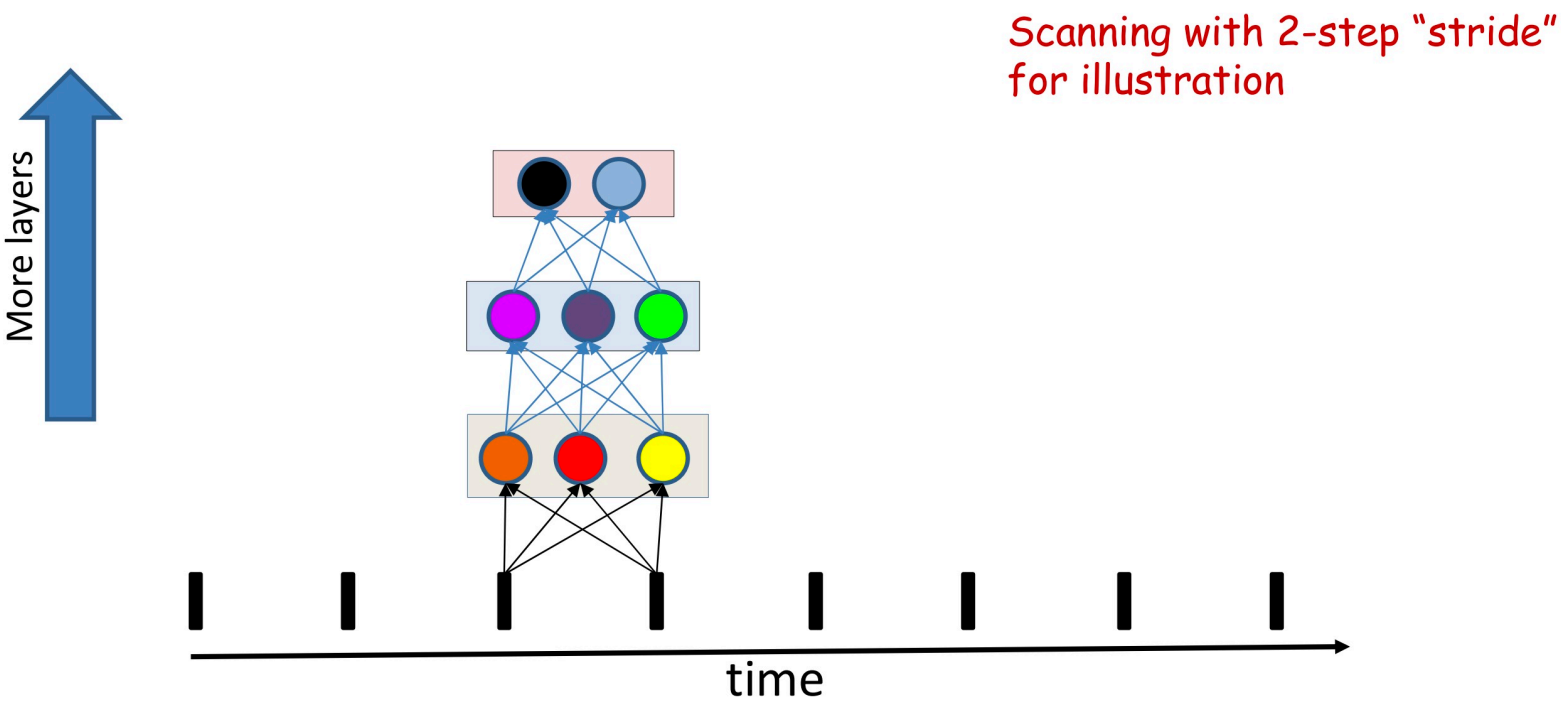
- In a scanning MLP each neuron is connected to a subset of neurons in the previous layer
  - The weights matrix is sparse
  - The weights matrix is block structured with identical blocks
  - The network is a shared parameter model
- Also, far fewer parameters

Scanning networks



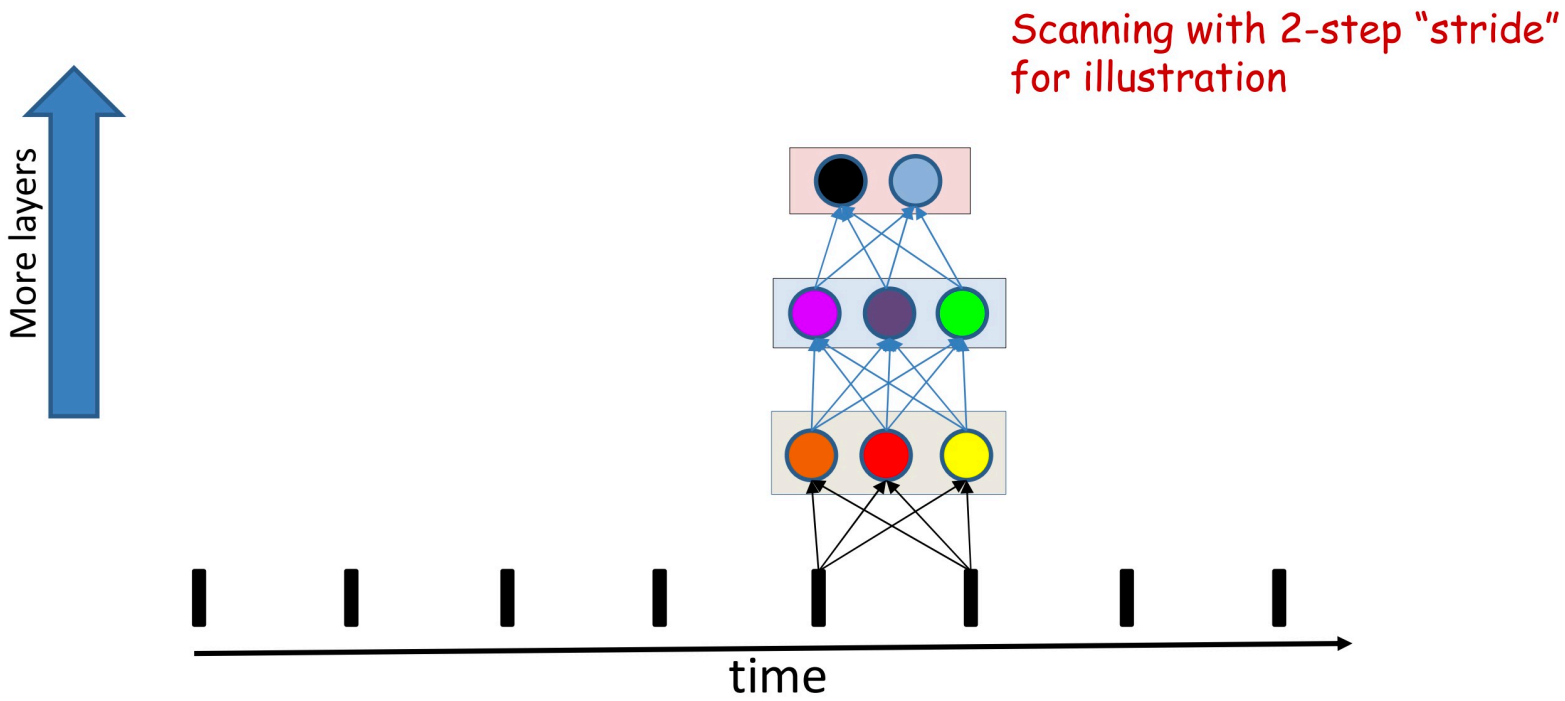
- Modifying the visualization for intuition..
  - Will still be the same network

Scanning networks



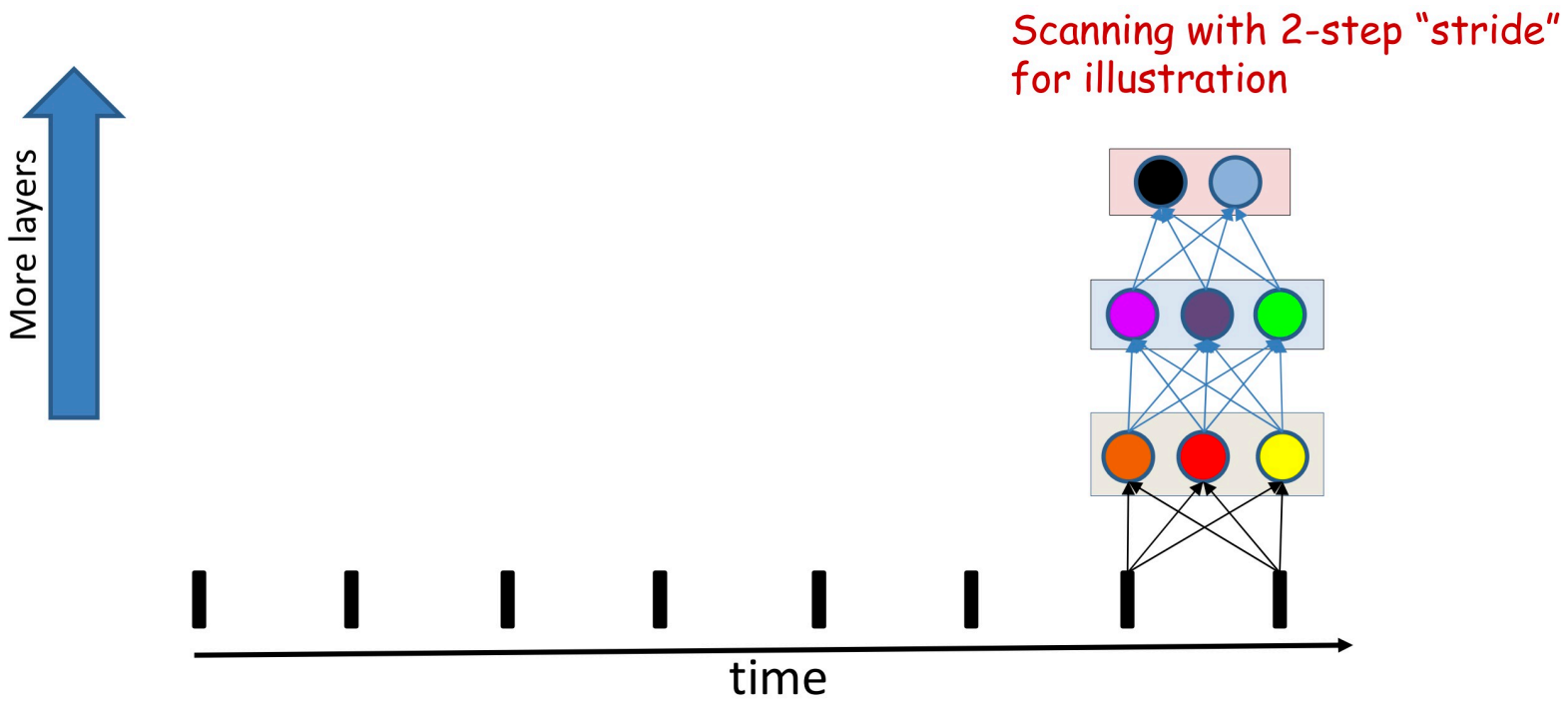
- Modifying the visualization for intuition..
  - Will still be the same network

Scanning networks



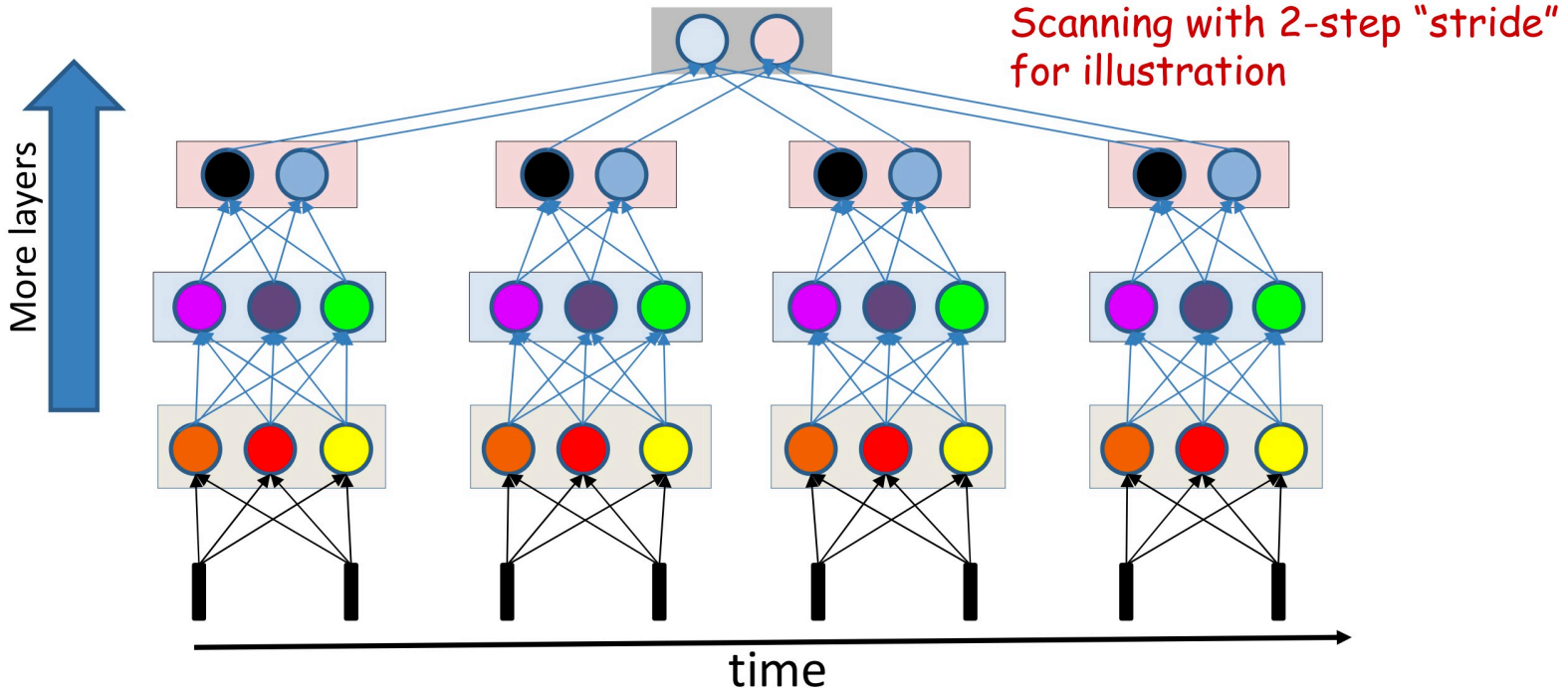
- Modifying the visualization for intuition..
  - Will still be the same network

Scanning networks



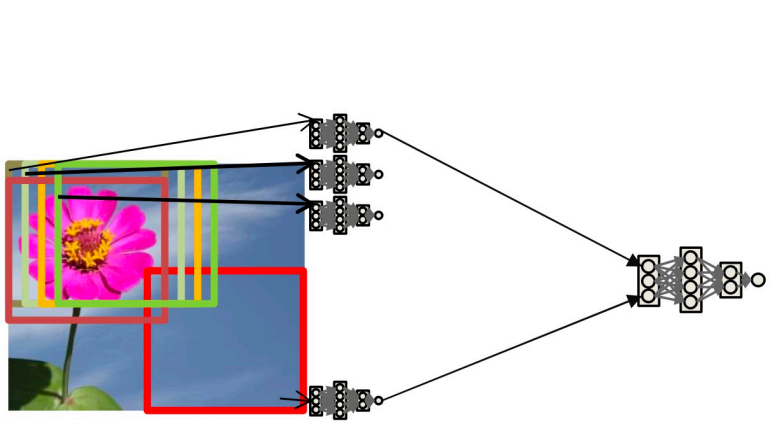
- Modifying the visualization for intuition..
  - Will still be the same network

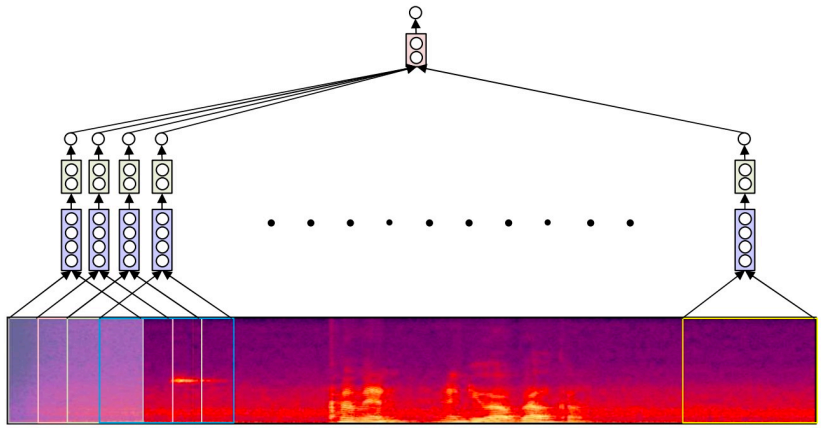
Scanning networks



- Modifying the visualization for intuition..
  - Will still be the same network

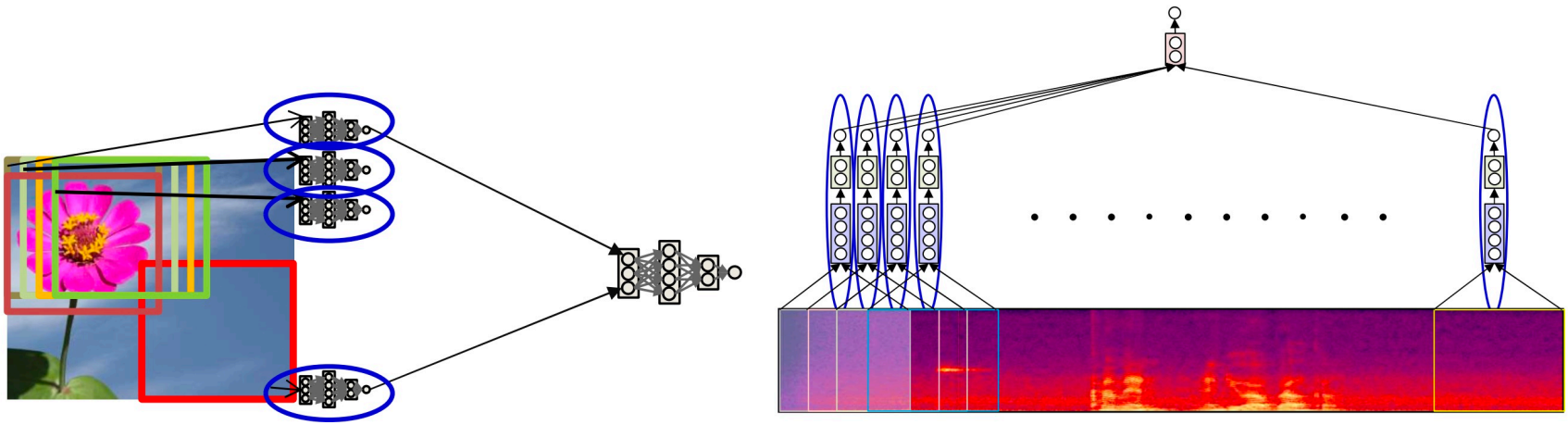
Training the network





- These are really just large networks
- Can just use backpropagation to learn the parameters
  - Provide many training examples
    - Images with and without flowers
    - Speech recordings with and without the word "welcome"
  - Gradient descent to minimize the total divergence between predicted and desired outputs
- Will actually learn the lower-level flower (or welcome) detector

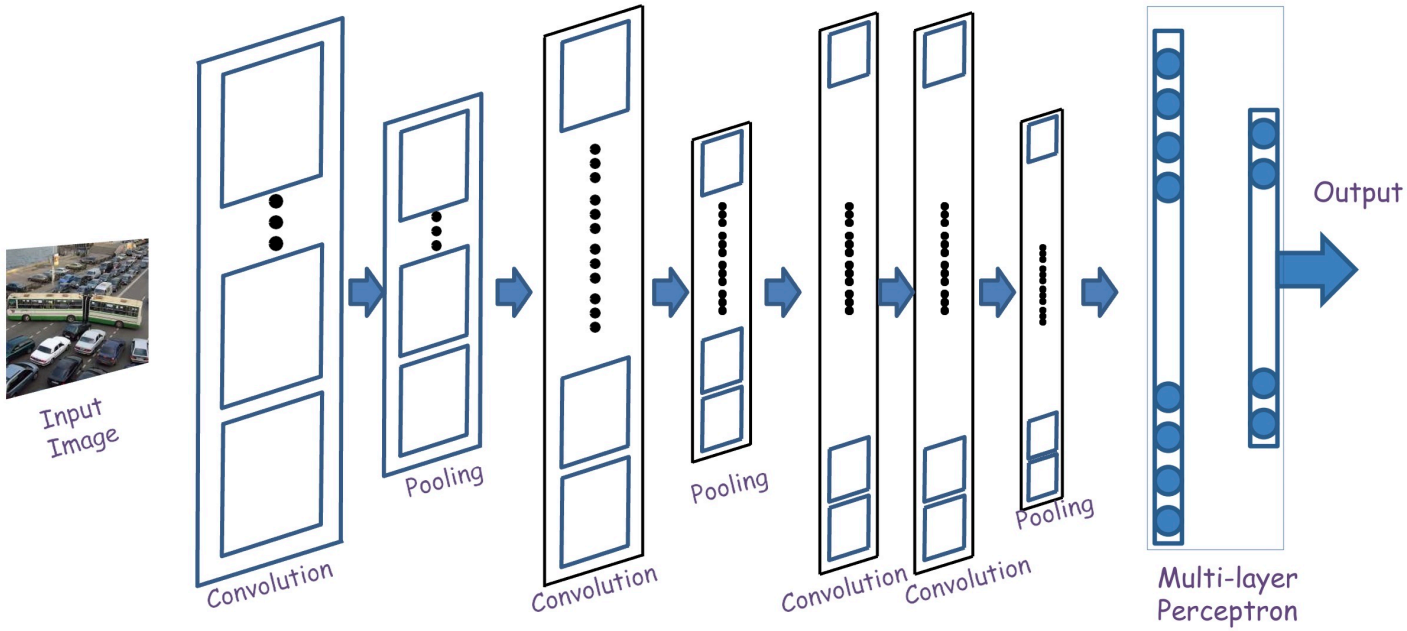
Training the network: constraint



- These are shared parameter networks
  - All lower-level subnets are identical
    - Are all searching for the same pattern
  - Any update of the parameters of one copy of the subnet must equally update all copies

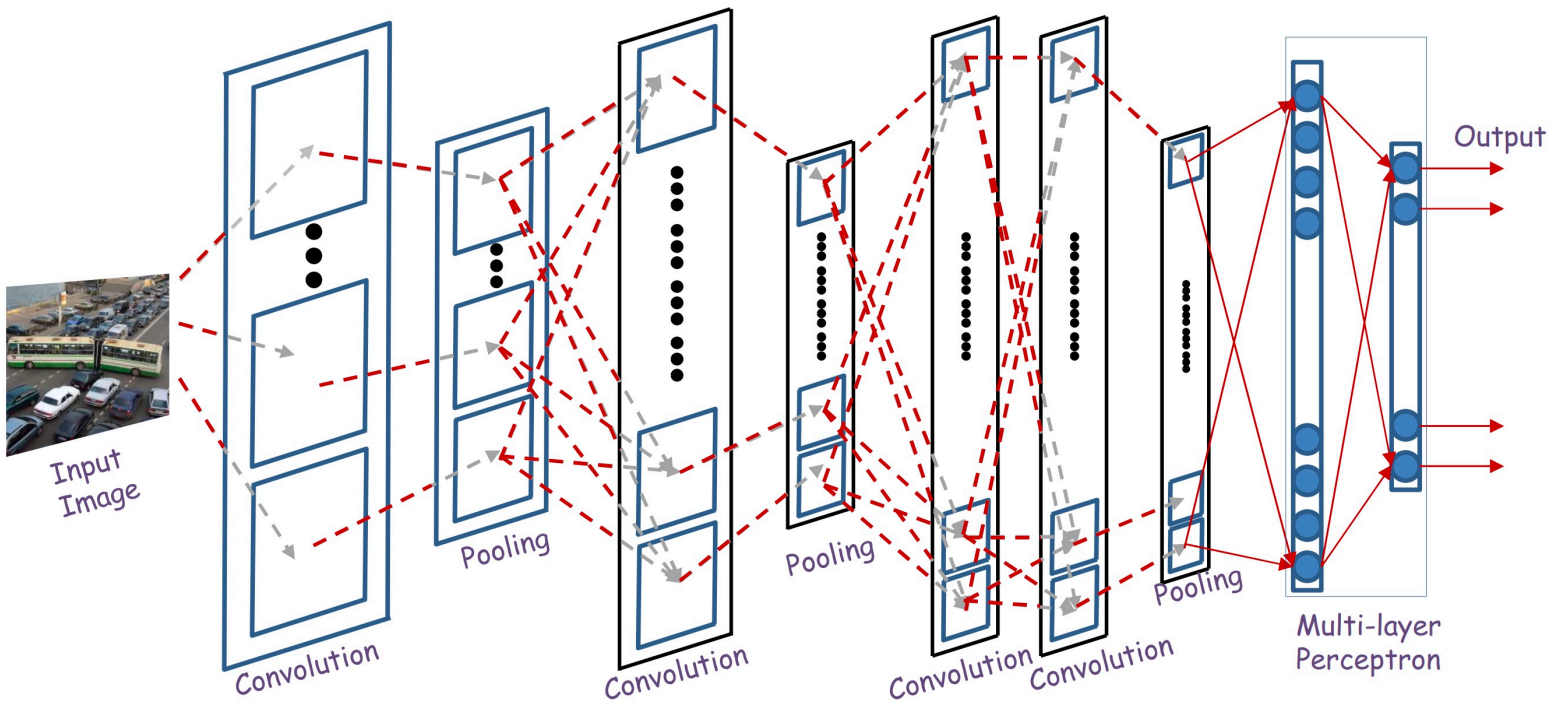
Lecture 6.2 Convolutional Neural Networks
卷积神经网络

The general architecture of a convolutional neural network



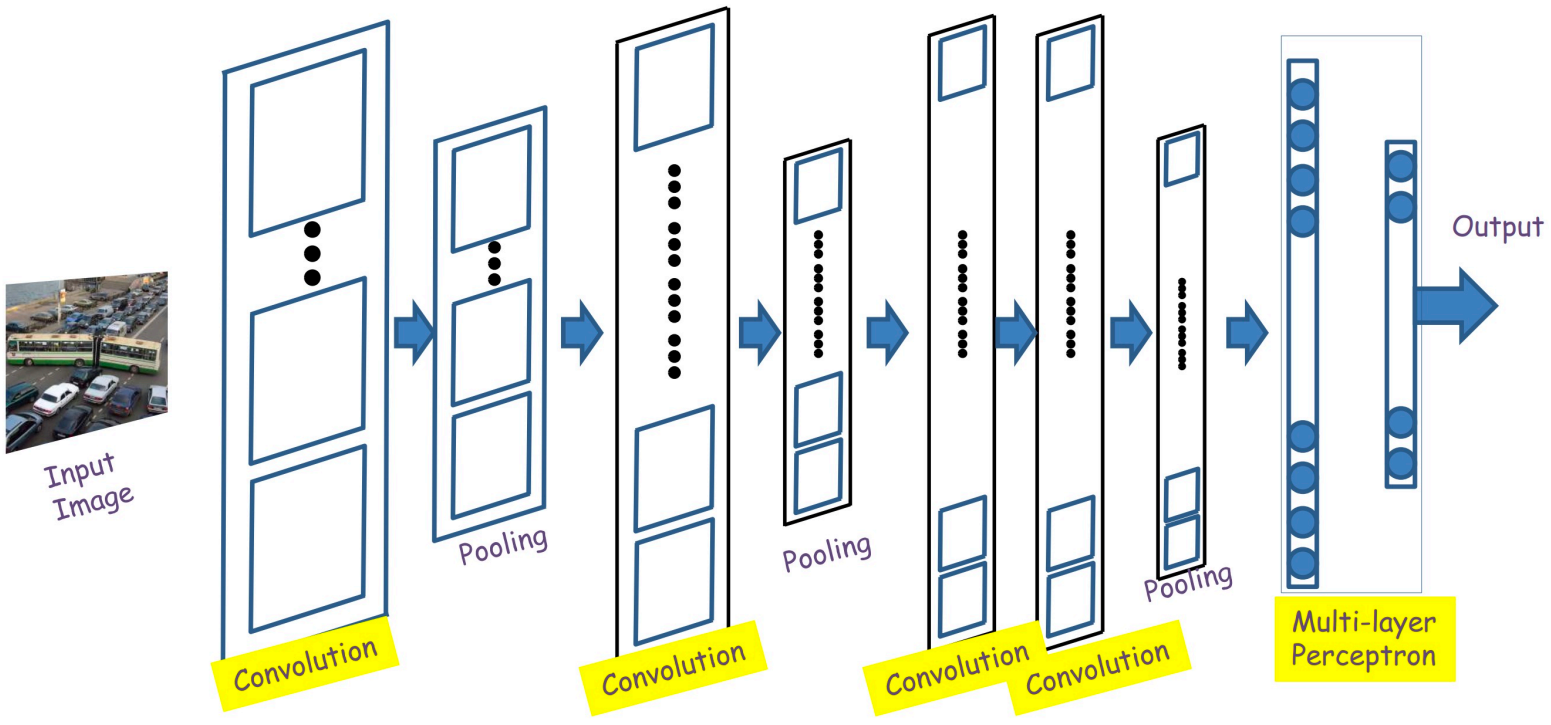
- A convolutional neural network comprises "convolutional" (卷积) and "pooling" (池化) layers
  - *Convolutional* layers include neurons to scan input for patterns
  - *Pooling* layers perform max operations on groups of outputs from the convolutional layers
  - They may occur in any sequence, but typically alternately
- Followed by an MLP with one or more layers

The general architecture of a convolutional neural network



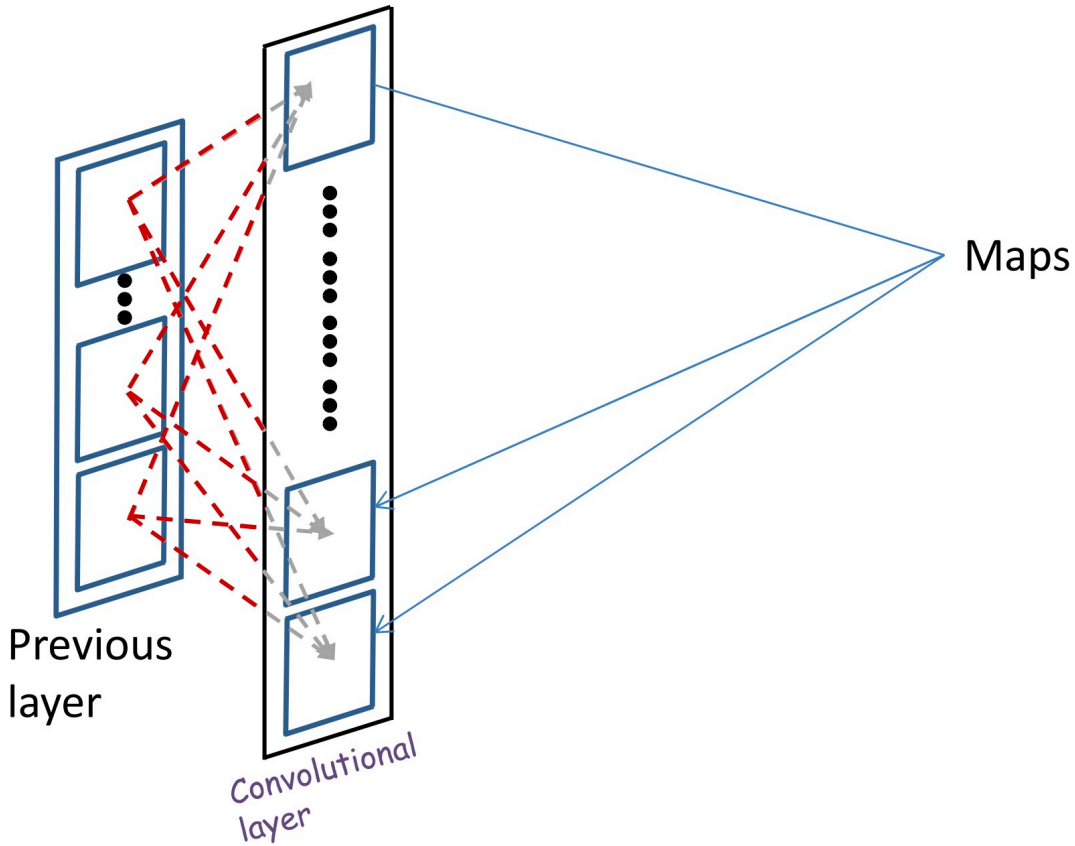
- A convolutional neural network comprises "convolutional" (卷积) and "pooling" (池化) layers
  - The two may occur in any sequence, but typically alternately
- Followed by an MLP with one or more layers

The general architecture of a convolutional neural network



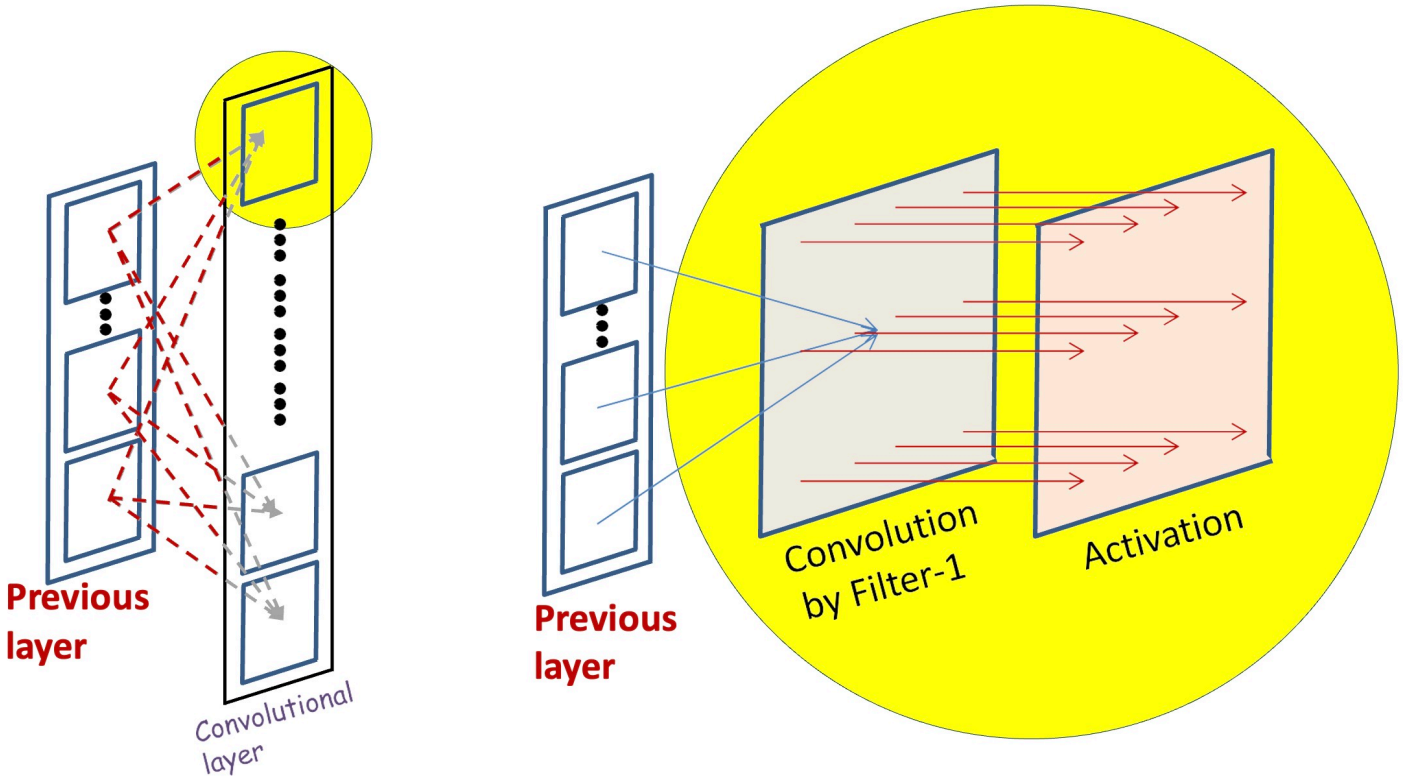
- Convolutional layers and the MLP are learnable
  - Their parameters must be learned from training data for the target classification task
- Pooling layers are fixed and generally not learnable

A convolutional layer



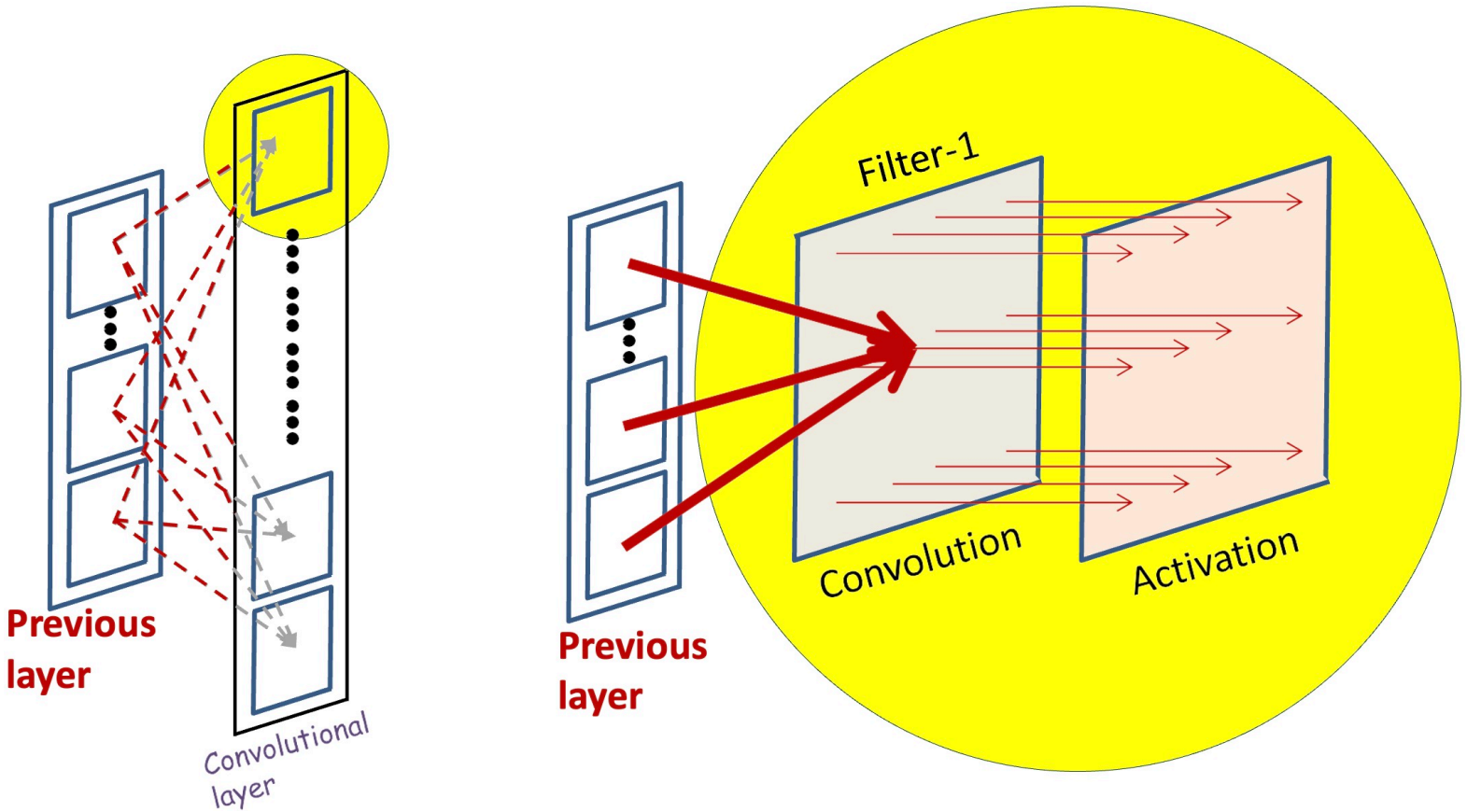
- A convolutional layer comprises of a series of "maps"
  - Called feature maps (特征图) or activation maps (激活图)

A convolutional layer



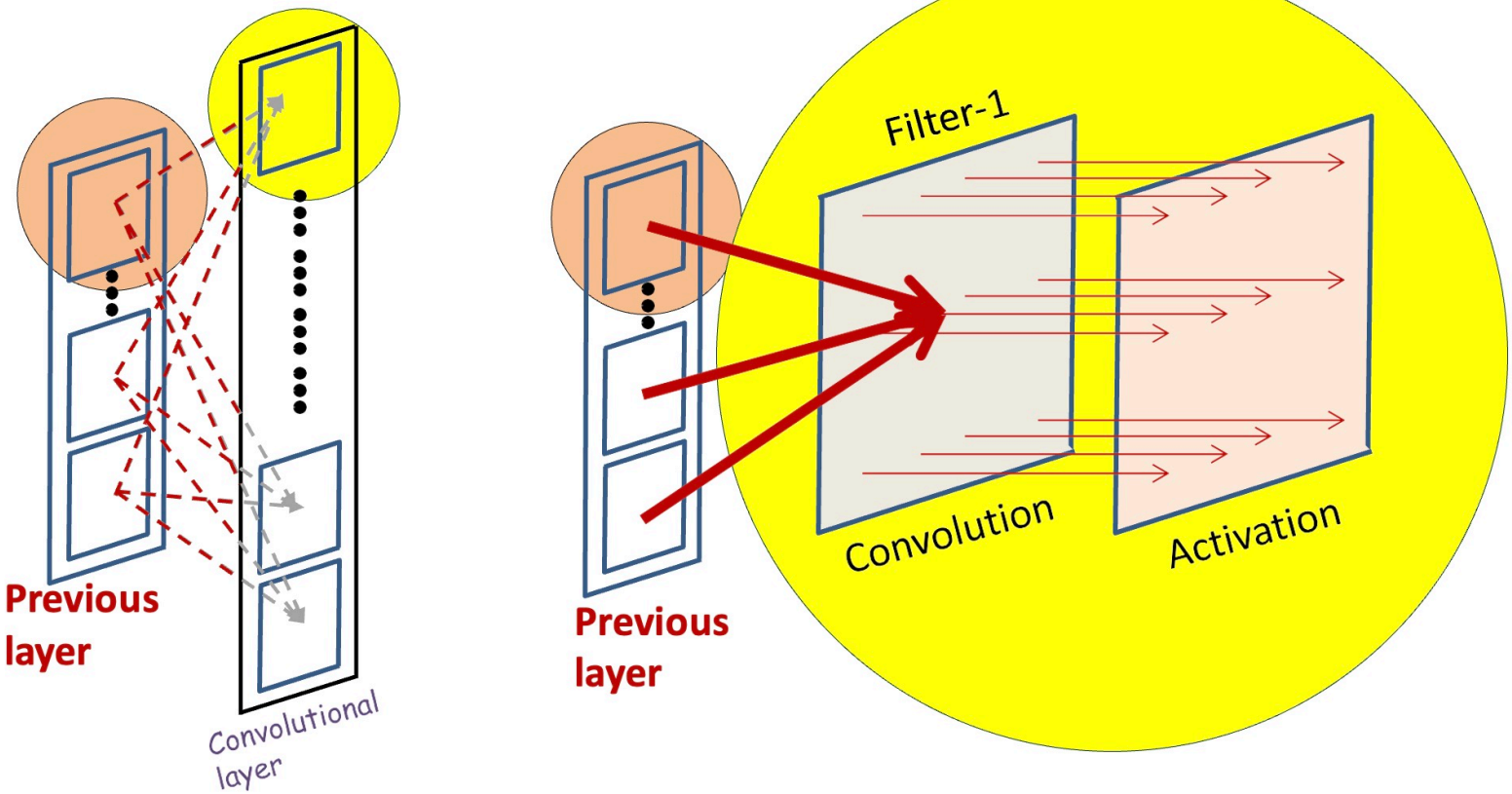
- Each feature map has two components
  - An affine (仿射) map: convolution over maps in previous layer
    - Each associated with a learnable filter/kernel (过滤器/卷积核)
  - An activation that operates on the output of the convolution

A convolutional layer: affine map (仿射映射)



- All the maps in the previous layer contribute to each convolution

A convolutional layer: affine map (仿射映射)



- All the maps in the previous layer contribute to each convolution
  - Consider the contribution of a single map

What is a convolution

Example 5x5 image with binary pixels

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

Example 3x3 filter

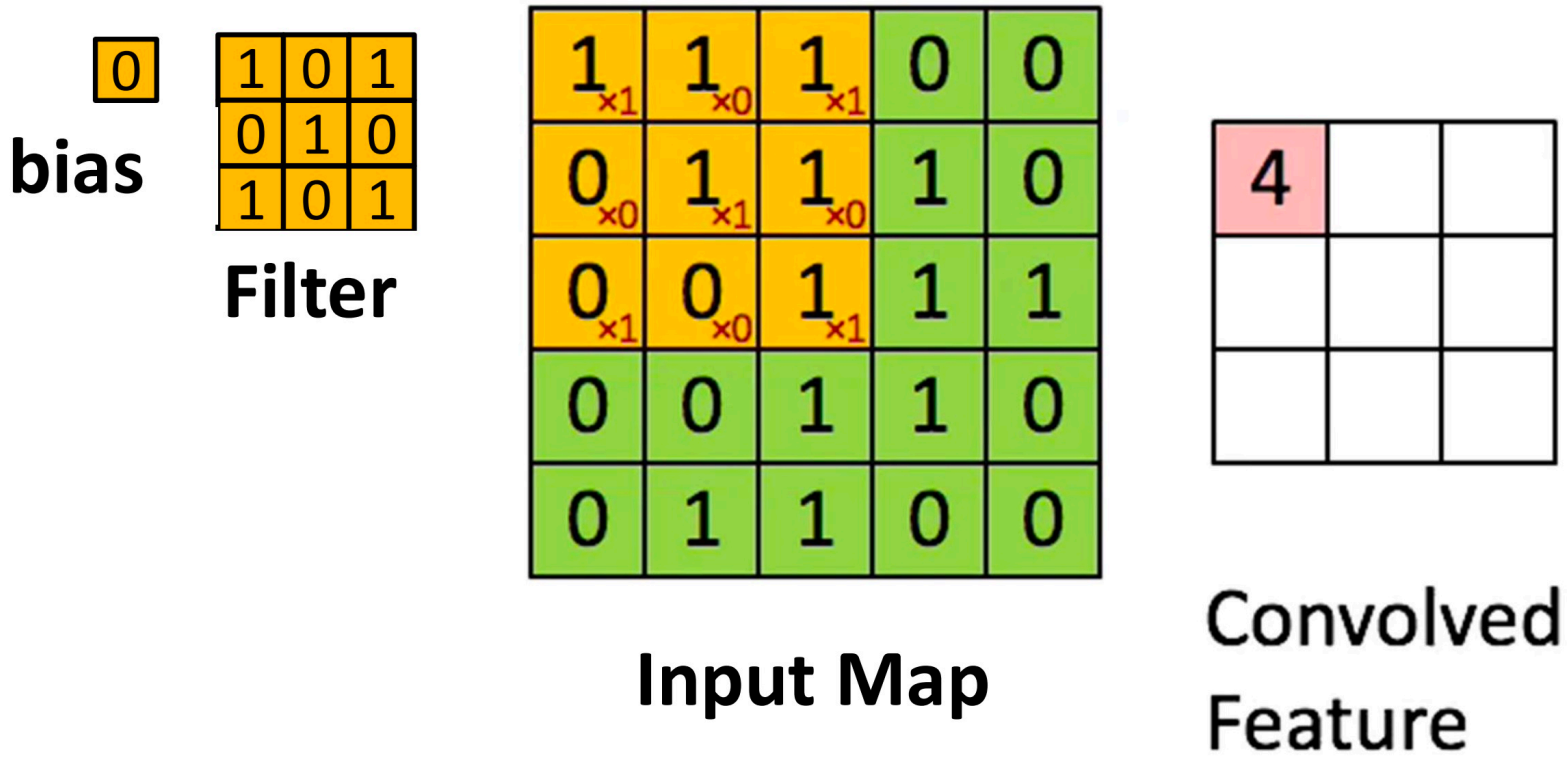
1	0	1
0	1	0
1	0	1

bias

0

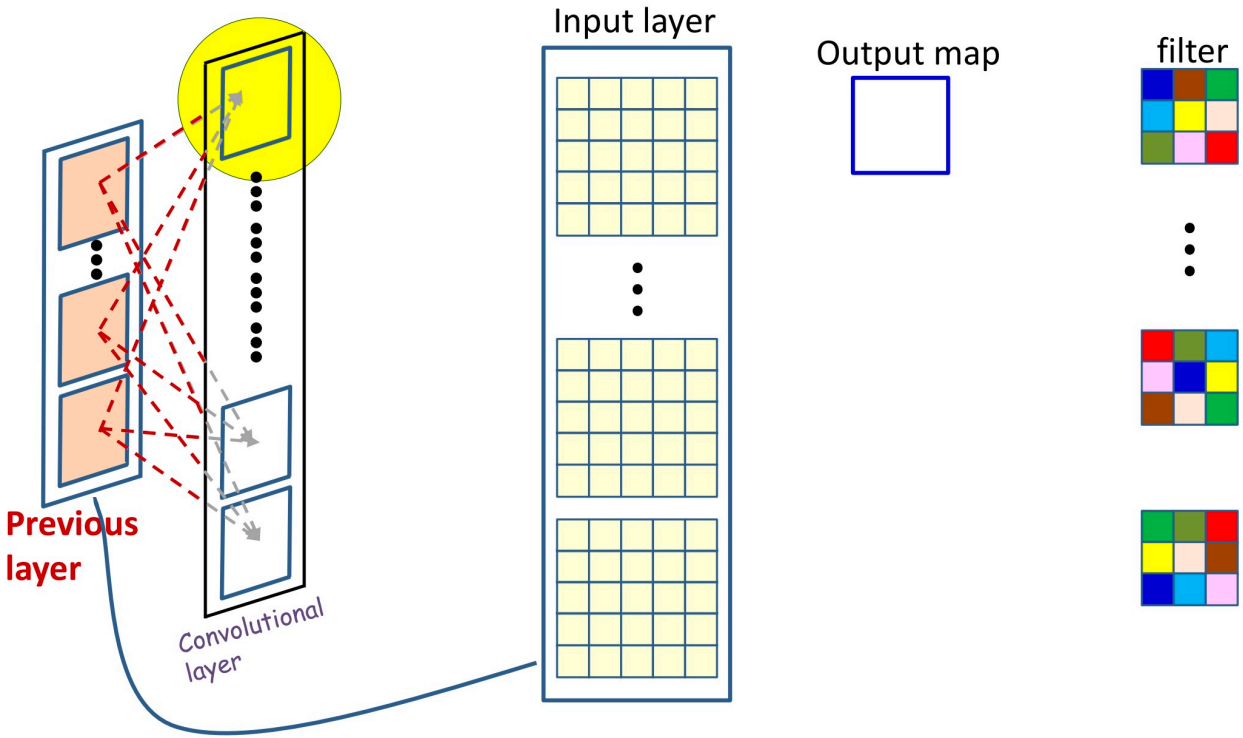
- Scanning an image with a "filter"
  - Note: a filter is really a perceptron, with weights and a bias

What is a convolution



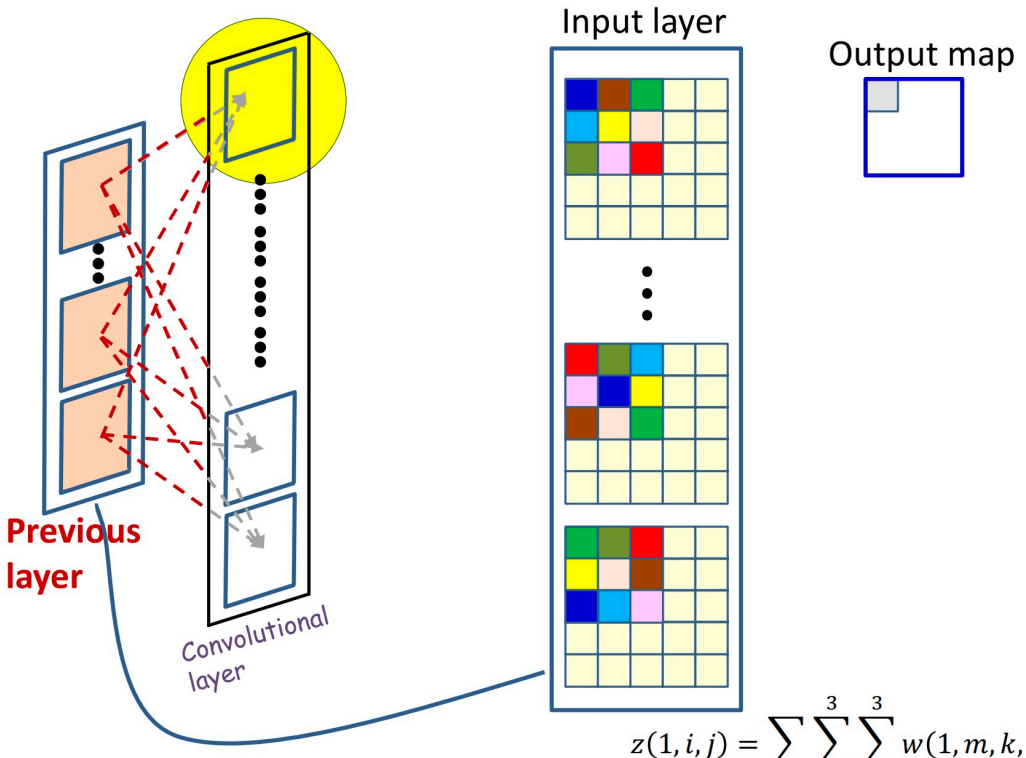
- Scanning an image with a "filter"
  - At each location, the "filter and the underlying map values are multiplied component wise, and the products are added along with the bias

What really happens



- Each output is computed from multiple maps simultaneously
- There are as many weights (for each output map) as:
  size of the filter $\times$ *no. of maps in previous layer*

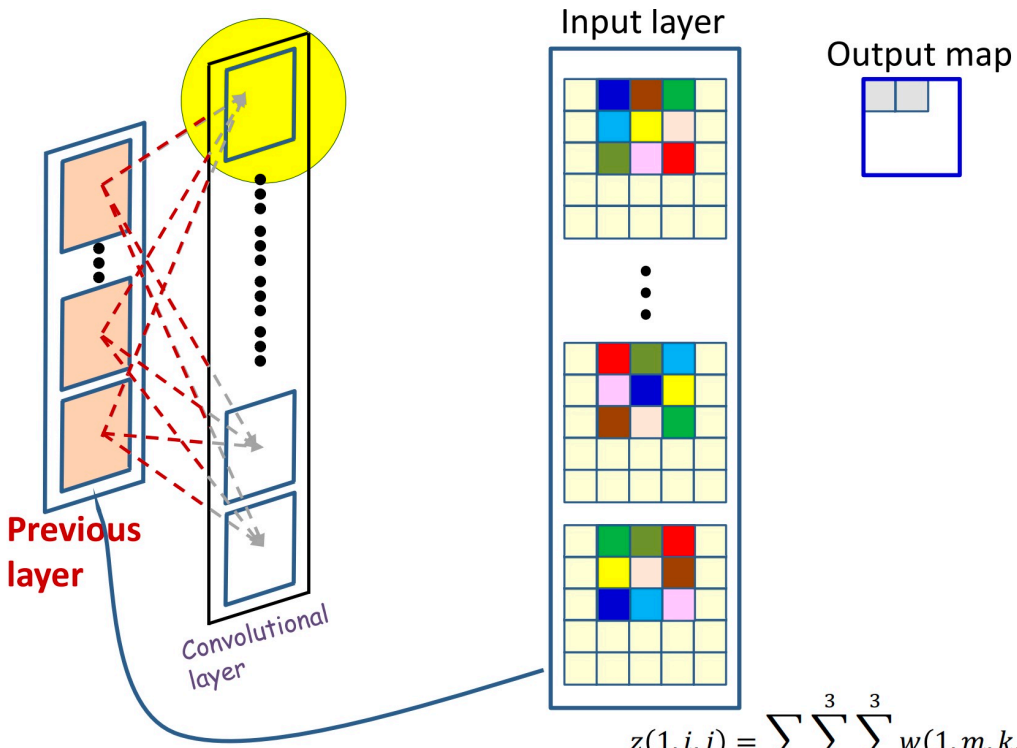
What really happens



$$z(1,i,j) = \sum_{m} \sum_{k=1}^{3} \sum_{l=1}^{3} w(1,m,k,l) I(m, i+l-1, j+k-1) + b$$

- Each output is computed from multiple maps simultaneously
- There are as many weights (for each output map) as:
  size of the filter $\times$ *no. of maps in previous layer*

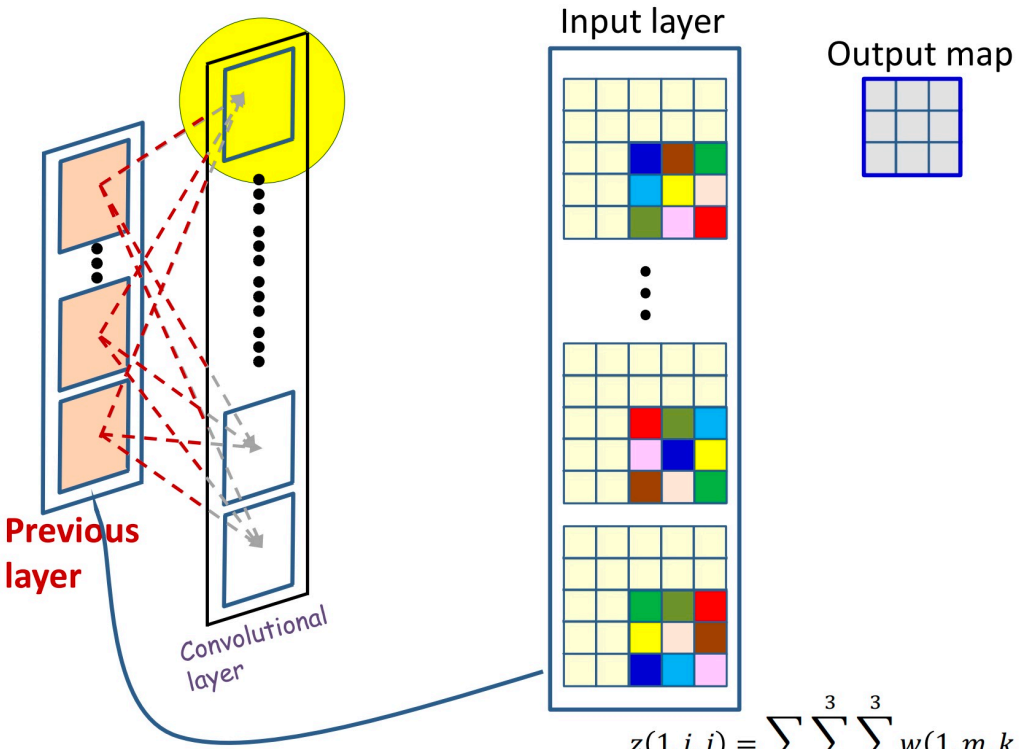
What really happens



$$z(1,i,j) = \sum_{m} \sum_{k=1}^{3} \sum_{l=1}^{3} w(1,m,k,l) I(m, i+l-1, j+k-1) + b$$

- Each output is computed from multiple maps simultaneously
- There are as many weights (for each output map) as:
  size of the filter $\times$ *no. of maps in previous layer*

What really happens

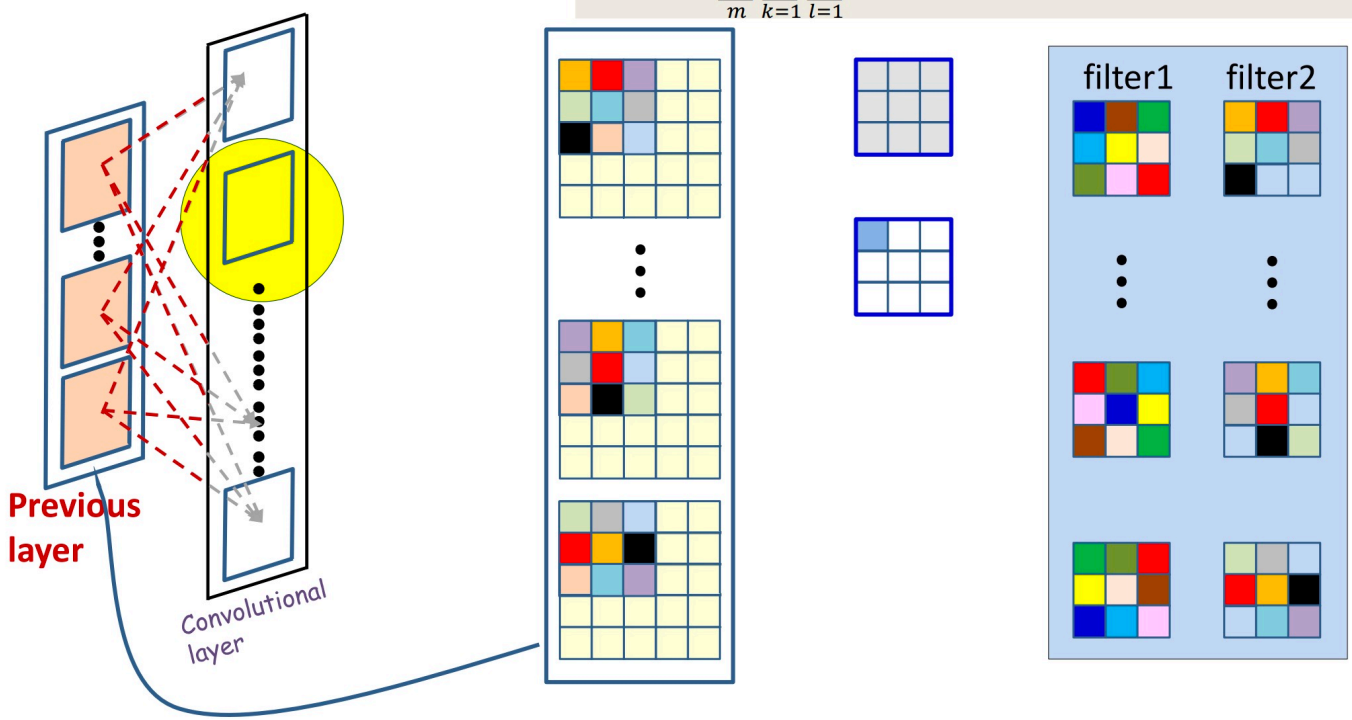


$$z(1,i,j) = \sum_{m} \sum_{k=1}^{3} \sum_{l=1}^{3} w(1,m,k,l) I(m, i+l-1, j+k-1) + b$$

- Each output is computed from multiple maps simultaneously
- There are as many weights (for each output map) as:
  size of the filter $\times$ *no. of maps in previous layer*

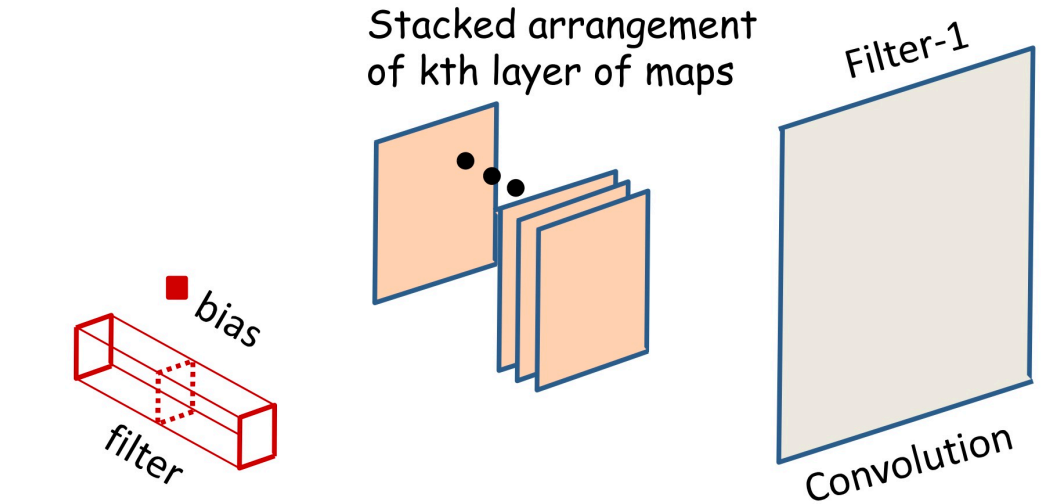
What really happens

$$z(2,i,j) = \sum_{m} \sum_{k=1}^{3} \sum_{l=1}^{3} w(2,m,k,l) I(m, i+l-1, j+k-1) + b(2)$$



- Each output is computed from multiple maps simultaneously
- There are as many weights (for each output map) as:
  size of the filter $\times$ *no. of maps in previous layer*

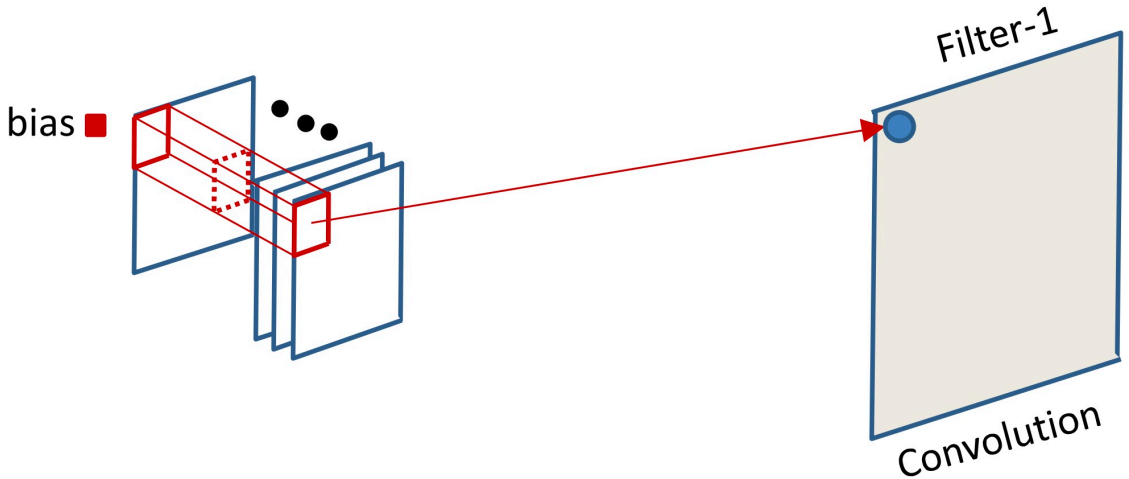
A different view



Filter applied to kth layer of maps
(convolutive component plus bias)

- ..A stacked arrangement of planes
- We can view the joint processing of the various maps as processing the stack using a three-dimensional filter

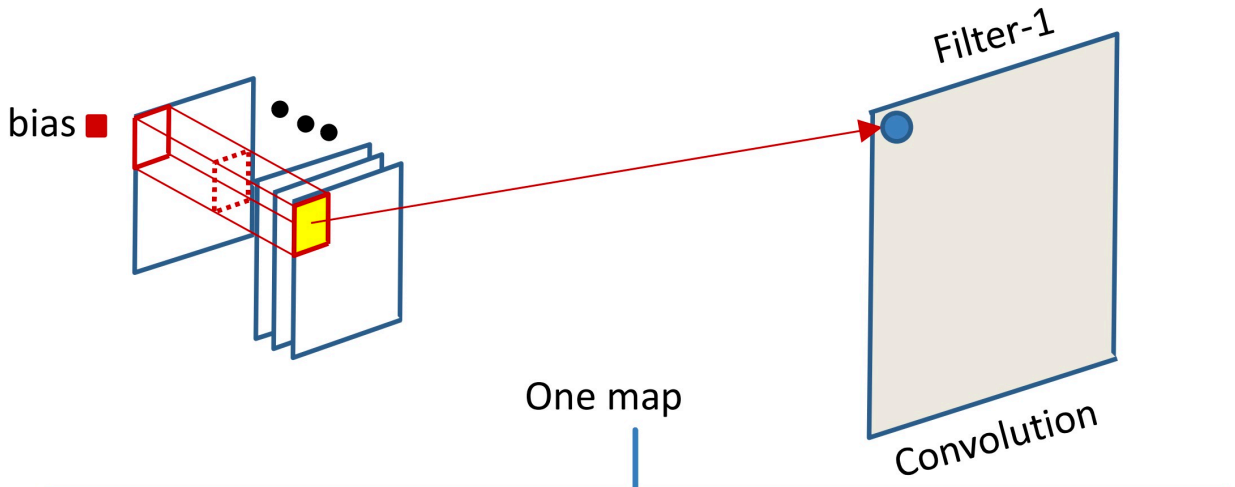
The "cube" view of input maps



$$z(s,i,j) = \sum_{p} \sum_{k=1}^{L} \sum_{l=1}^{L} w(s,p,k,l) Y(p, i+l-1, j+k-1) + b(s)$$

- The computation of the convolutional map at any location sums the convolutional outputs at all planes

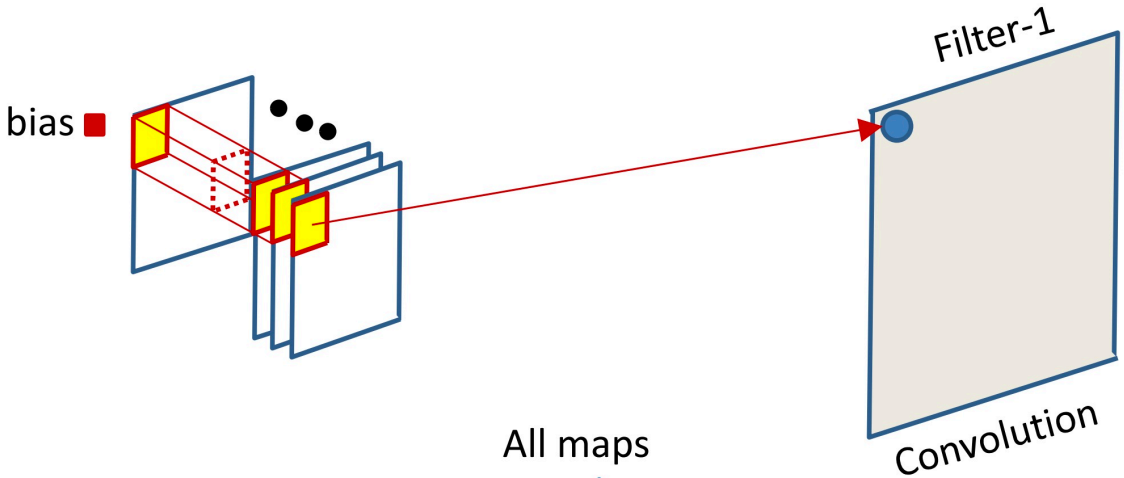
The "cube" view of input maps



$$z(s,i,j) = \sum_{p} \sum_{k=1}^{L} \sum_{l=1}^{L} w(s,p,k,l) Y(p, i+l-1, j+k-1) + b(s)$$

- The computation of the convolutional map at any location sums the convolutional outputs at all planes

The "cube" view of input maps

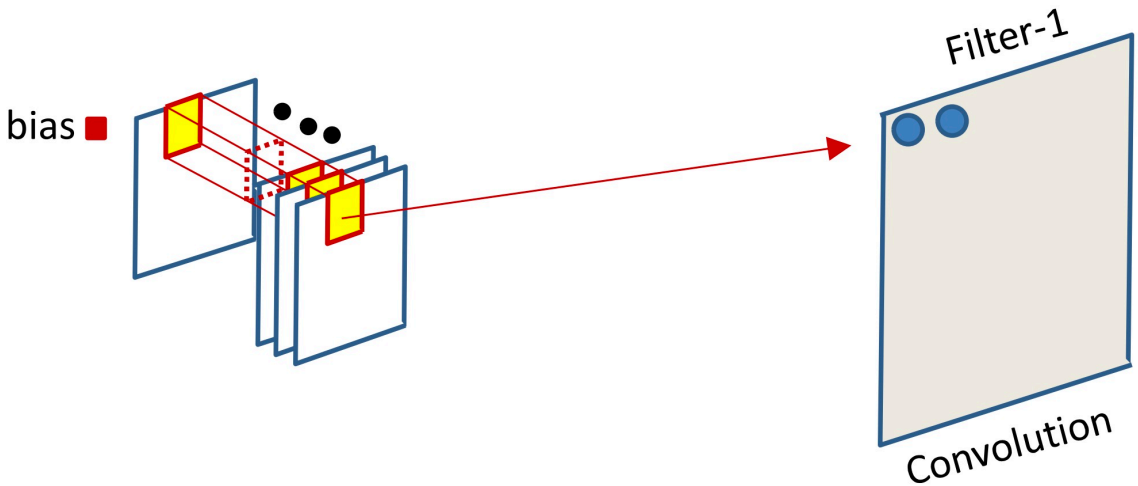


All maps

$$z(s,i,j) = \sum_{p} \sum_{k=1}^{L} \sum_{l=1}^{L} w(s,p,k,l) Y(p, i+l-1, j+k-1) + b(s)$$

- The computation of the convolutional map at any location sums the convolutional outputs at all planes

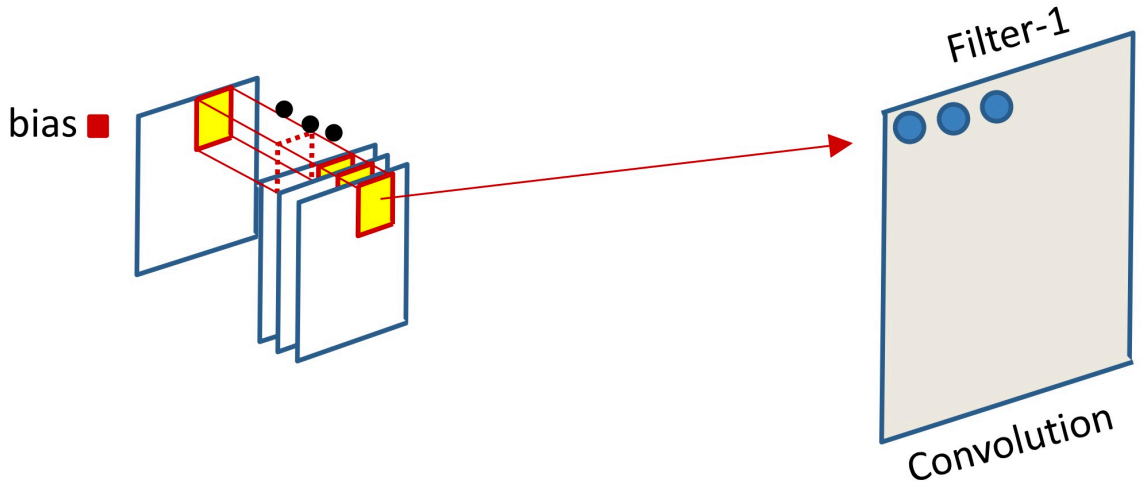
The "cube" view of input maps



$$z(s,i,j) = \sum_{p} \sum_{k=1}^{L} \sum_{l=1}^{L} w(s,p,k,l) Y(p, i+l-1, j+k-1) + b(s)$$

- The computation of the convolutional map at any location sums the convolutional outputs at all planes

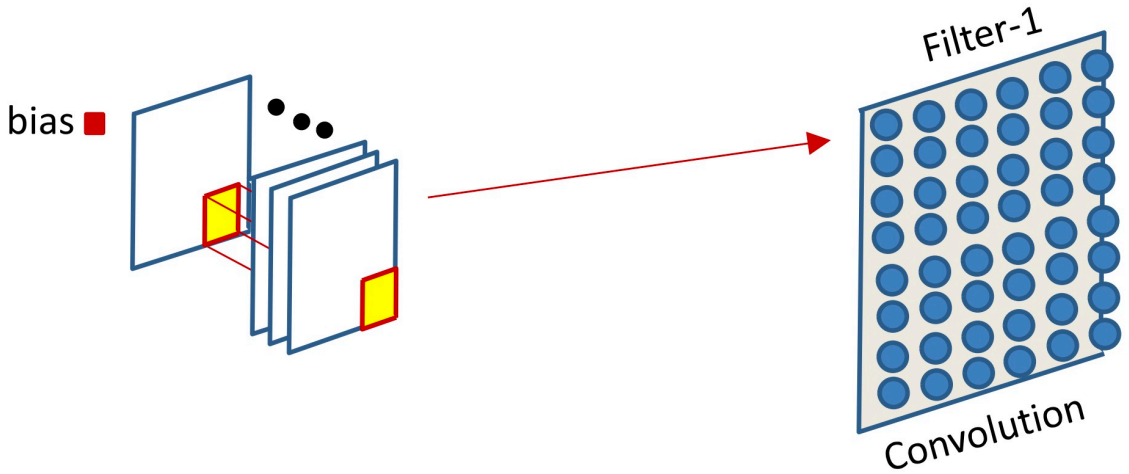
The "cube" view of input maps



$$z(s,i,j) = \sum_{p} \sum_{k=1}^{L} \sum_{l=1}^{L} w(s,p,k,l) Y(p, i+l-1, j+k-1) + b(s)$$

- The computation of the convolutional map at any location sums the convolutional outputs at all planes

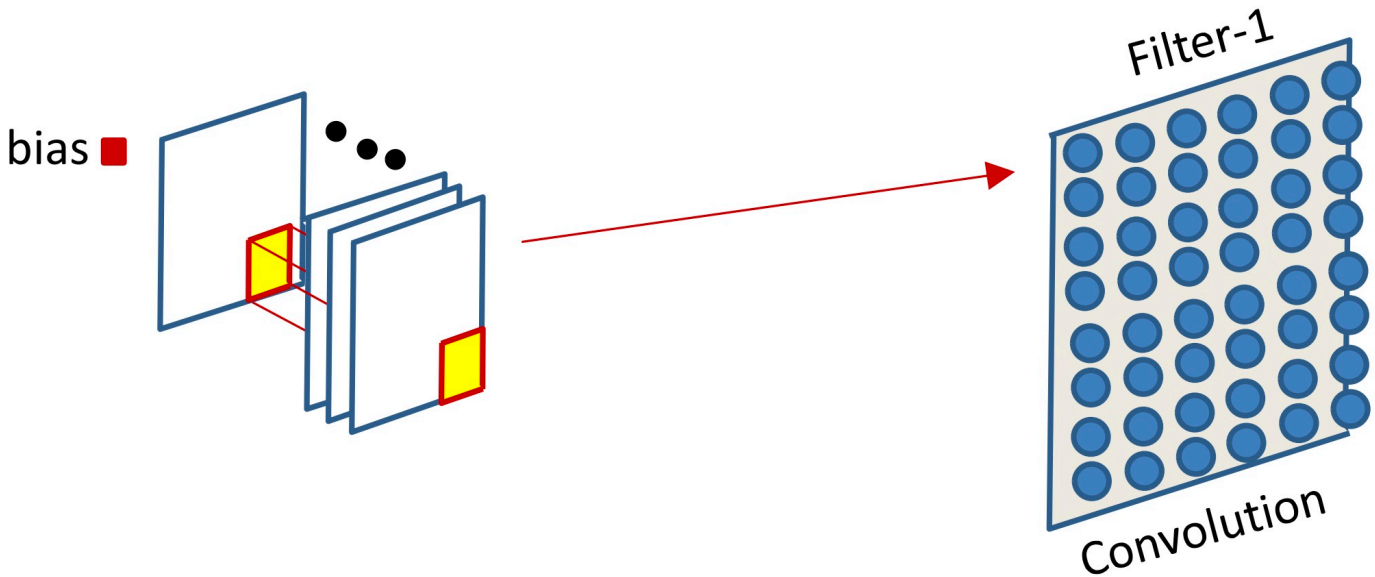
The "cube" view of input maps



$$z(s,i,j) = \sum_{p} \sum_{k=1}^{L} \sum_{l=1}^{L} w(s,p,k,l) Y(p, i+l-1, j+k-1) + b(s)$$

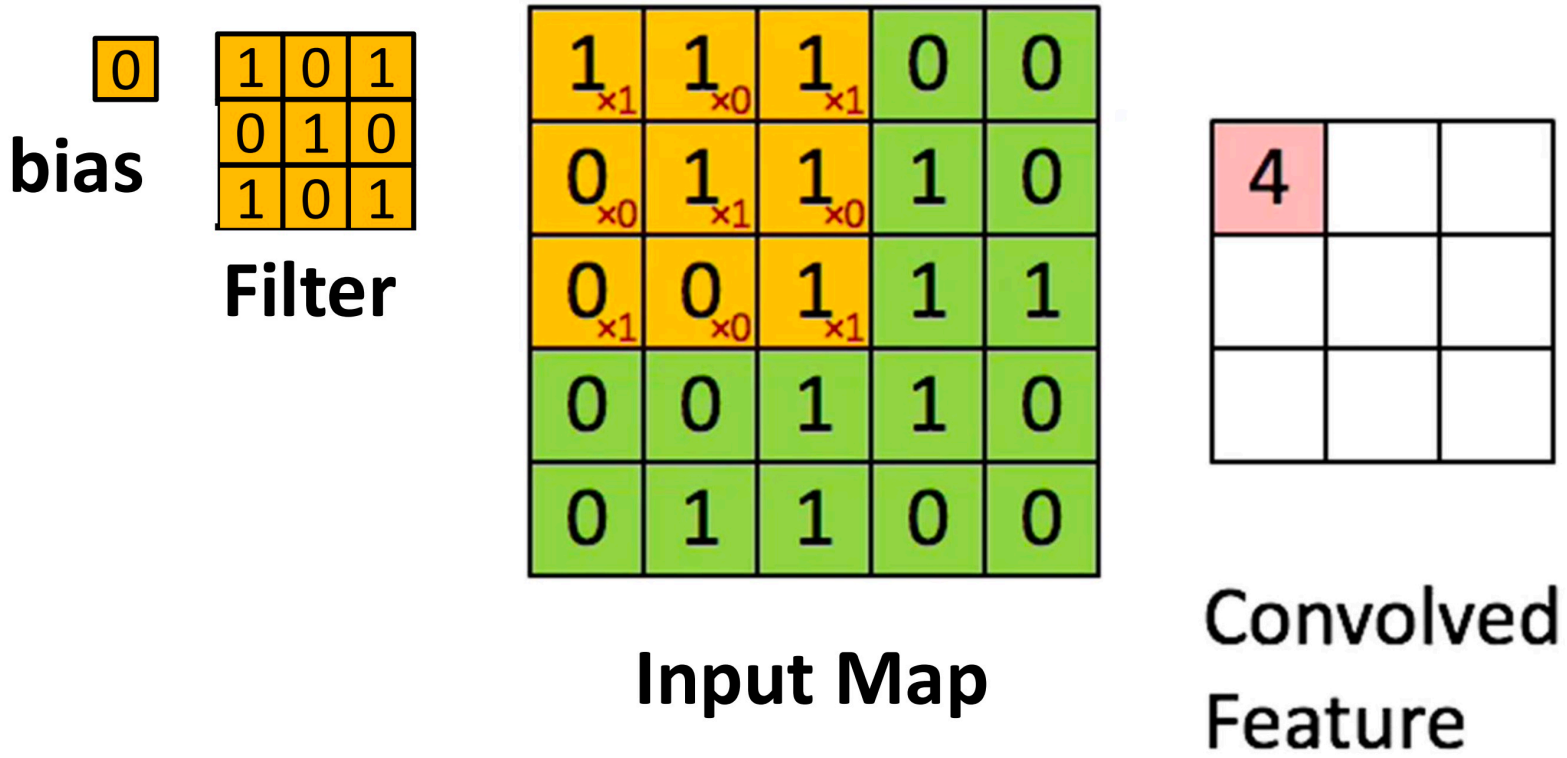
- The computation of the convolutional map at any location sums the convolutional outputs at all planes

Engineering consideration: The size of the result of the convolution



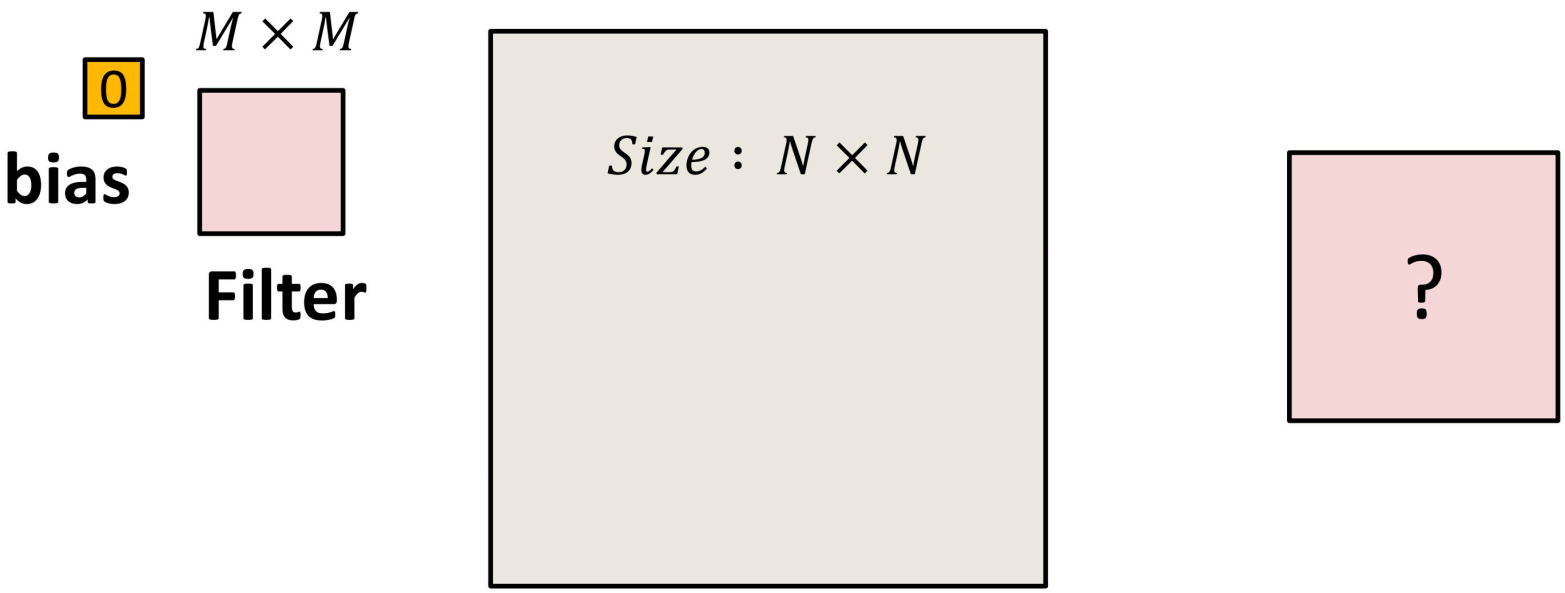
- The size of the output of the convolution operation depends on implementation factors
  - The size of the input, the size of the filter
- And may not be identical to the size of the input
  - Let's take a brief look at this for completeness sake

The size of the convolution



- Image size: 5x5
- Filter: 3x3
- Output size = ?

The size of the convolution

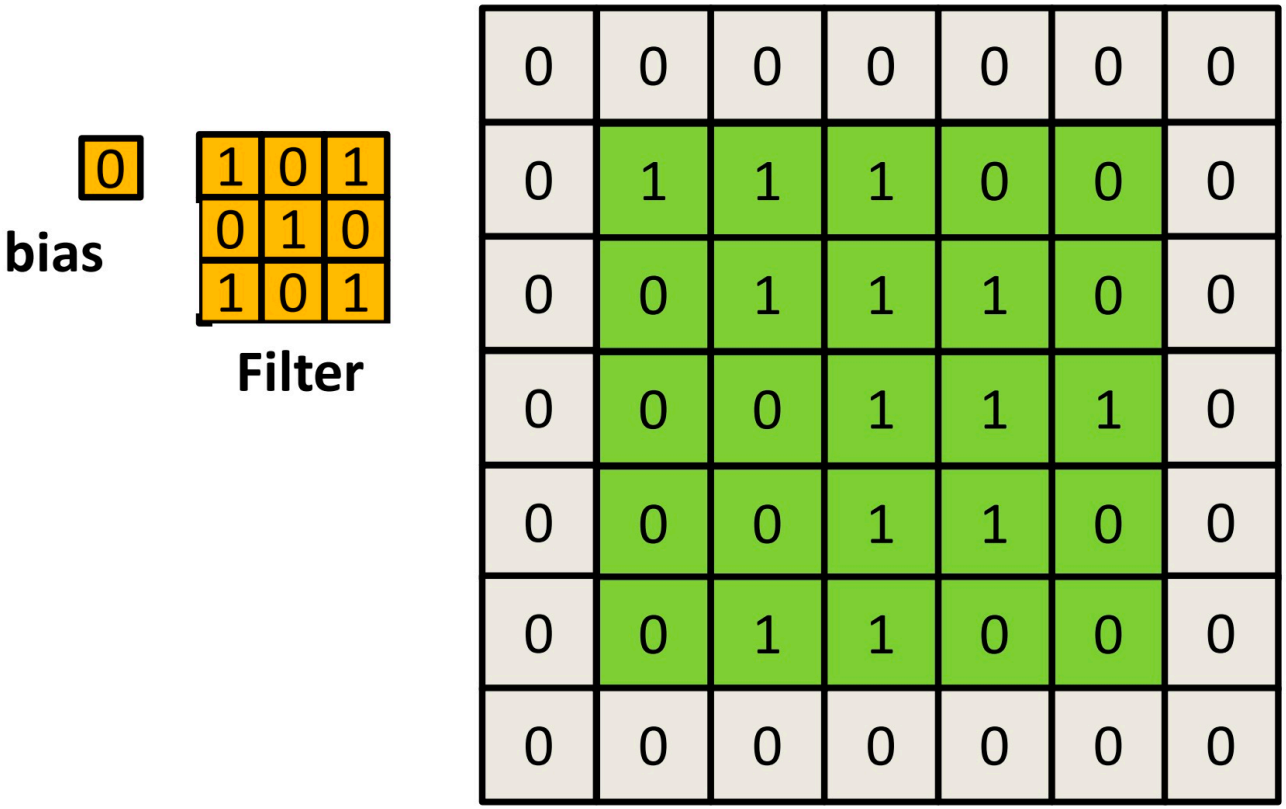


- Image size: $N \times N$
- Filter: $M \times M$
- Output size $= (N - M) + 1$ on each side

The size of the convolution

- Simple convolution size pattern:
  - Image size: $N \times N$
  - Filter: $M \times M$
  - Output size (each side)$= (N - M)+1$
    - Assuming not allowed to go beyond the edge of the input
- Results in a reduction in the output size
  - Sometimes not considered acceptable

Solution

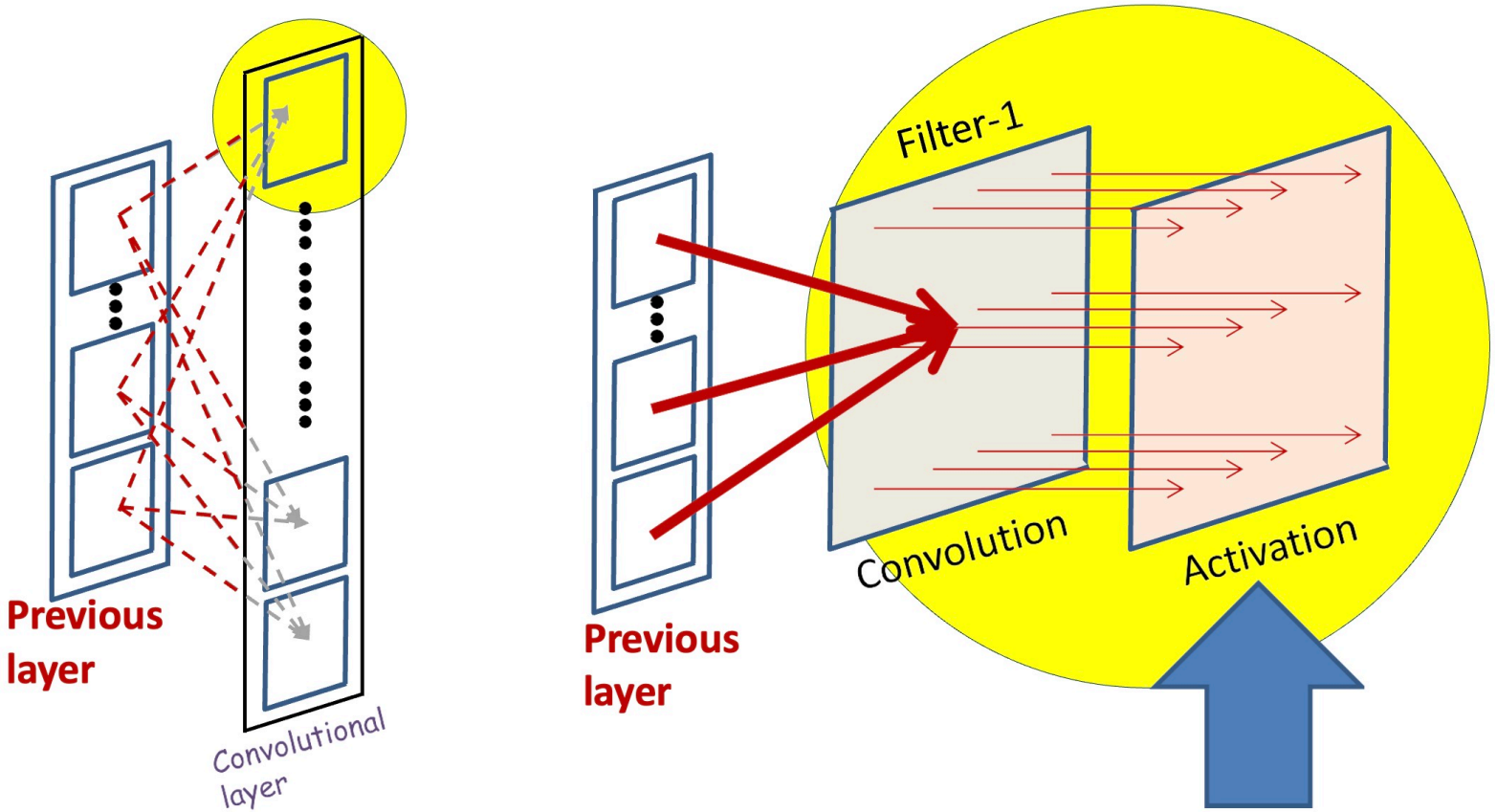


- Zero-pad the input
  - Pad the input image/map all around
  - Pad as symmetrically as possible
  - Result of the convolution is the same size as the original image

Zero padding

- For an L width filter:
  - Odd L: Pad on both sides with $(L-1)/2$ columns of zeros
  - Even L: Pad one side with $L/2$ columns of zeros, and the other with $L/2-1$ columns of zeros
  - The resulting image is width $N+L-1$
  - The result of the convolution is width N
- The top/bottom zero padding follows the same rules to maintain map height after convolution

A convolutional layer



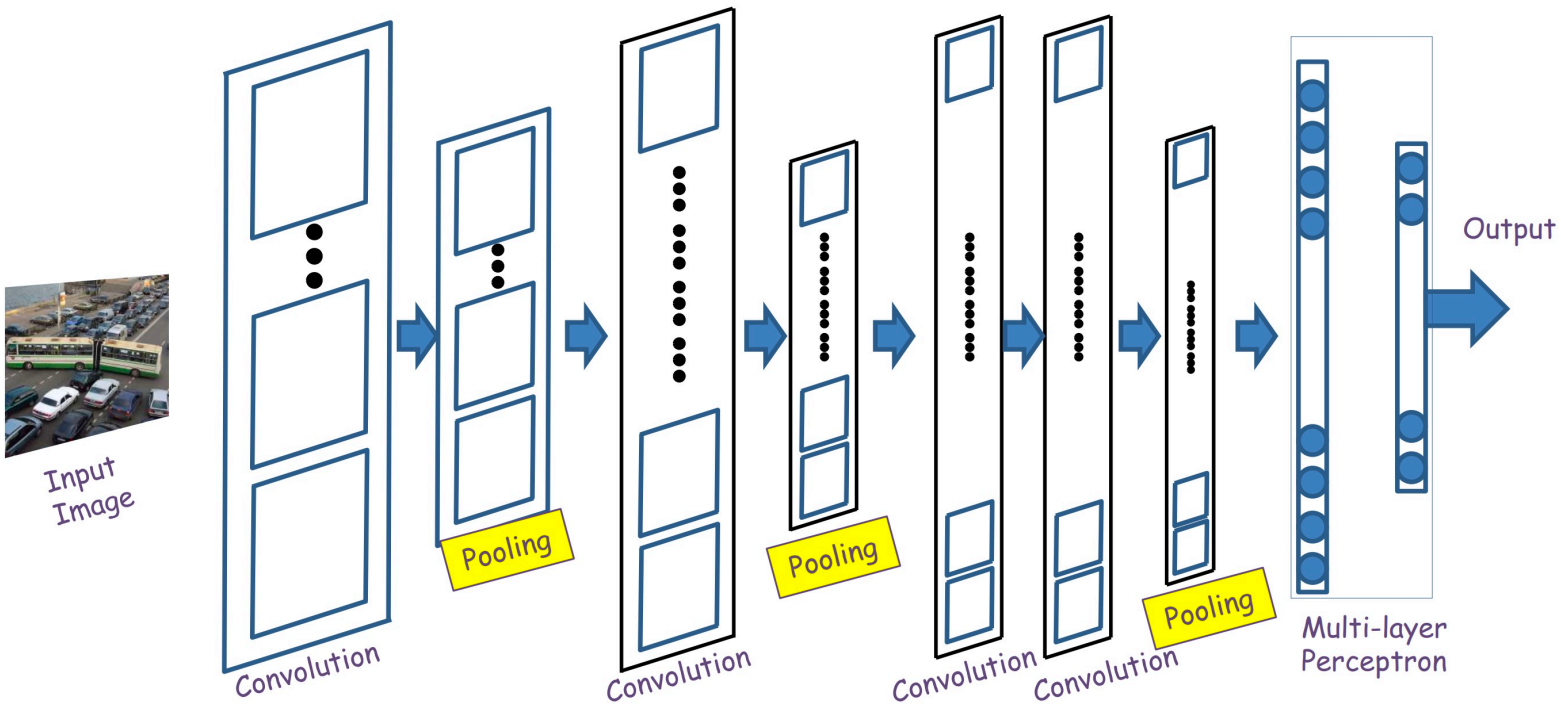
- The convolution operation results in an affine map
- An Activation is finally applied to every entry in the map

Convolution Summary

- Convolutional layers "scan" the input using a bank of "filters"
  - A "filter" is just a neuron in a scanning layer
- Each filter jointly scans the maps in the previous layer to produce an output "map"
  - As many output maps as filters (one output map per filter)
    - Regardless of the number of input maps
- We may have to pad the edges of the input maps to ensure that the output maps are the same size as input maps
  - If not, convolution loses rows/columns at the edges of the scan

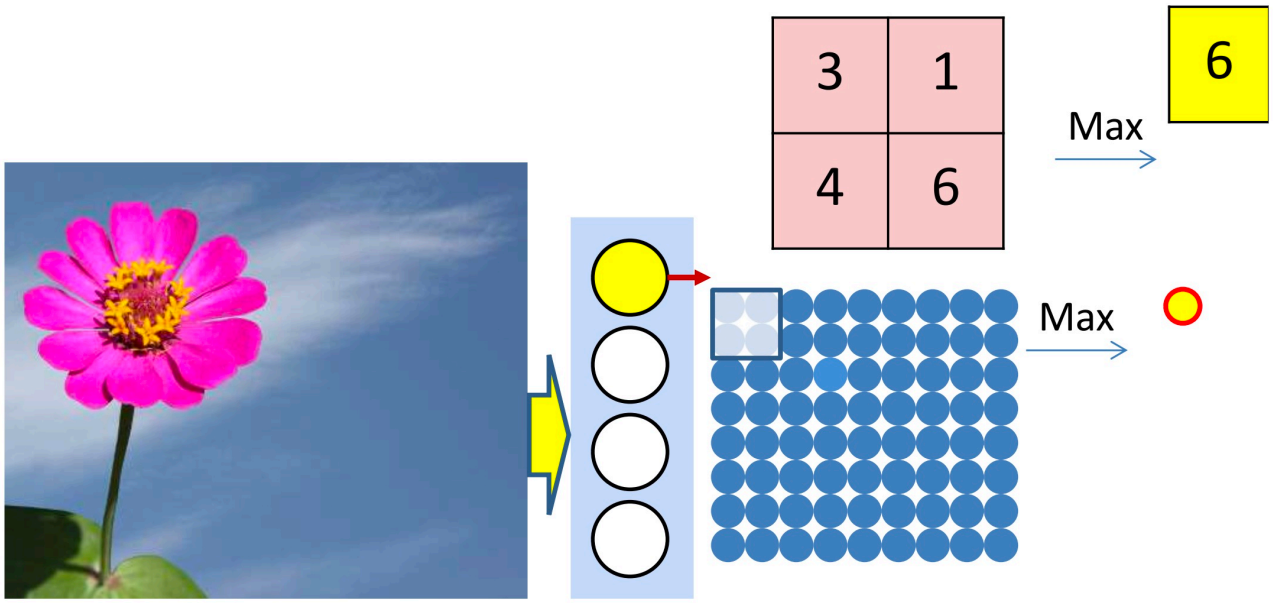
Lecture 6.3 Pooling/Downsampling 池化/下采样

Pooling/Downsampling 池化/下采样



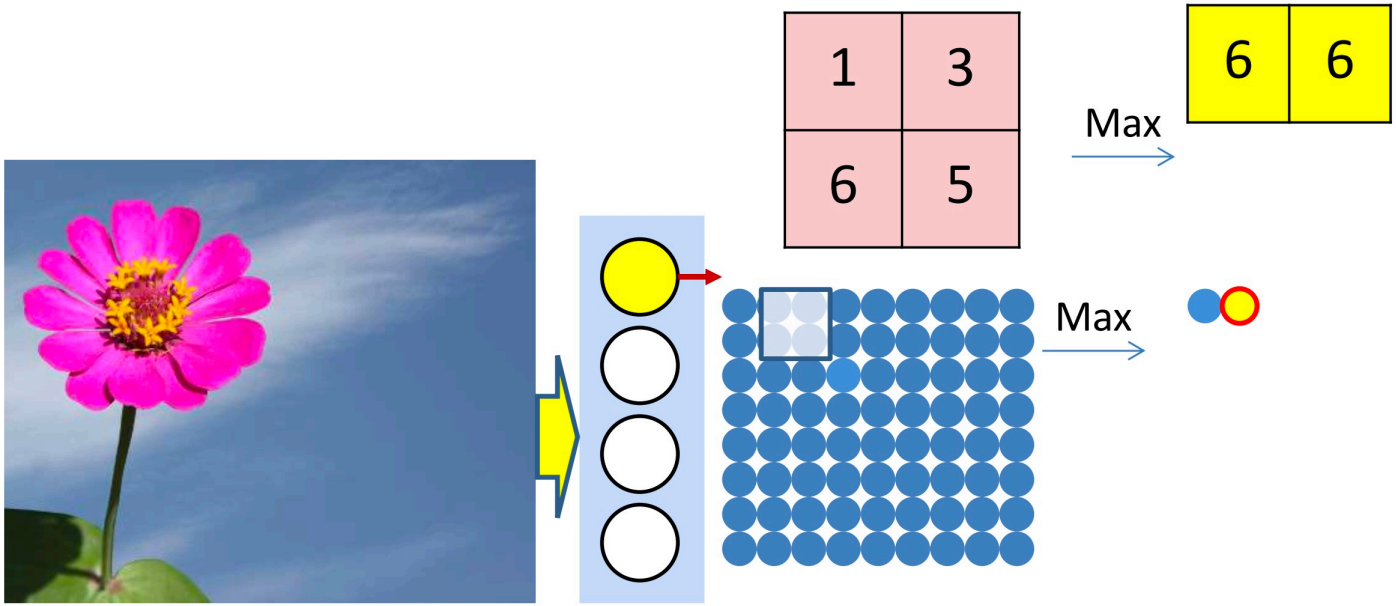
- Convolution (and activation) layers are followed intermittently by "pooling" layers
  - Typically (but not always) "max" pooling
  - Often, they alternate with convolution, though not necessarily

Max pooling 最大池化



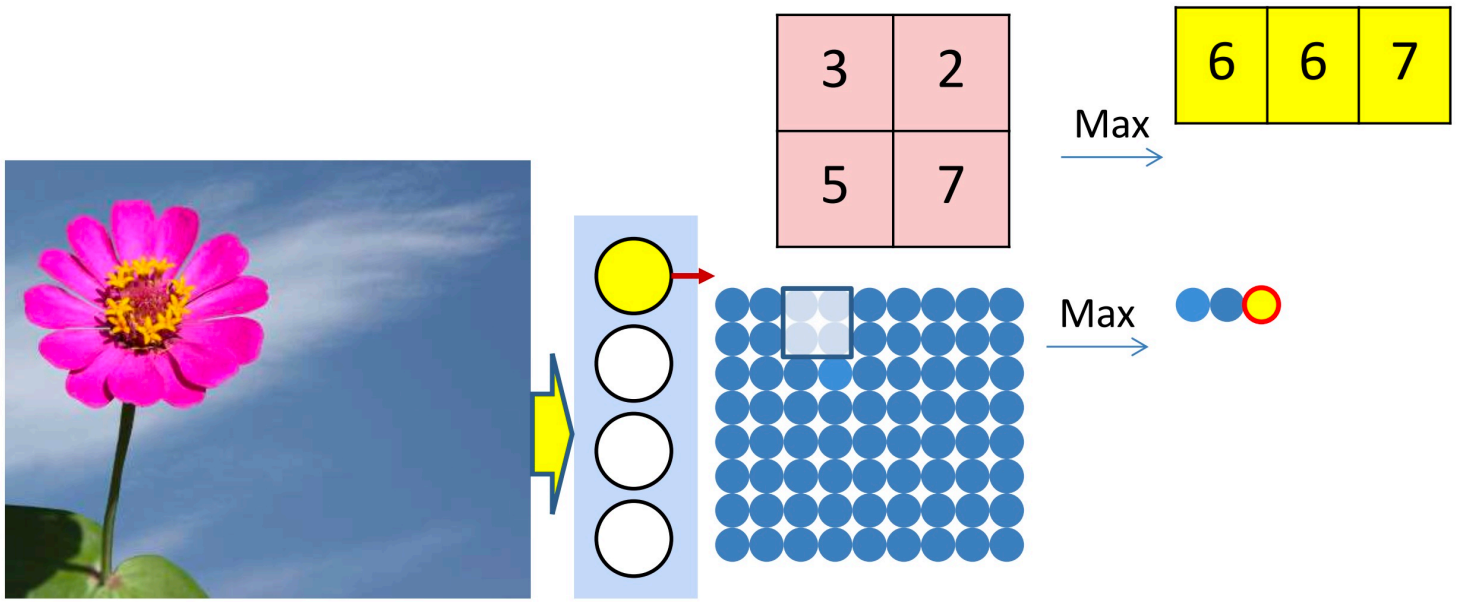
- Max pooling selects the largest from a pool of elements
- Pooling is performed by “scanning” the input

Max pooling 最大池化



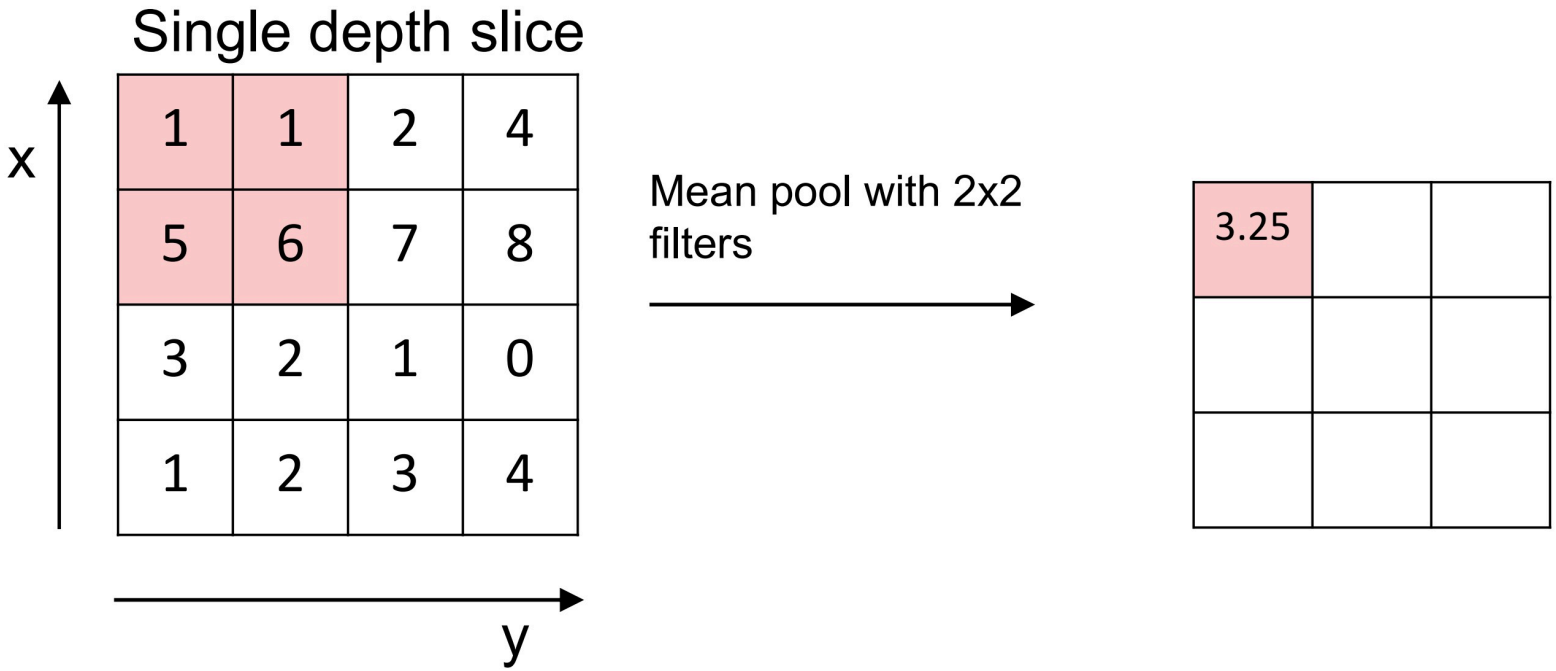
- Max pooling selects the largest from a pool of elements
- Pooling is performed by “scanning” the input

Max pooling 最大池化



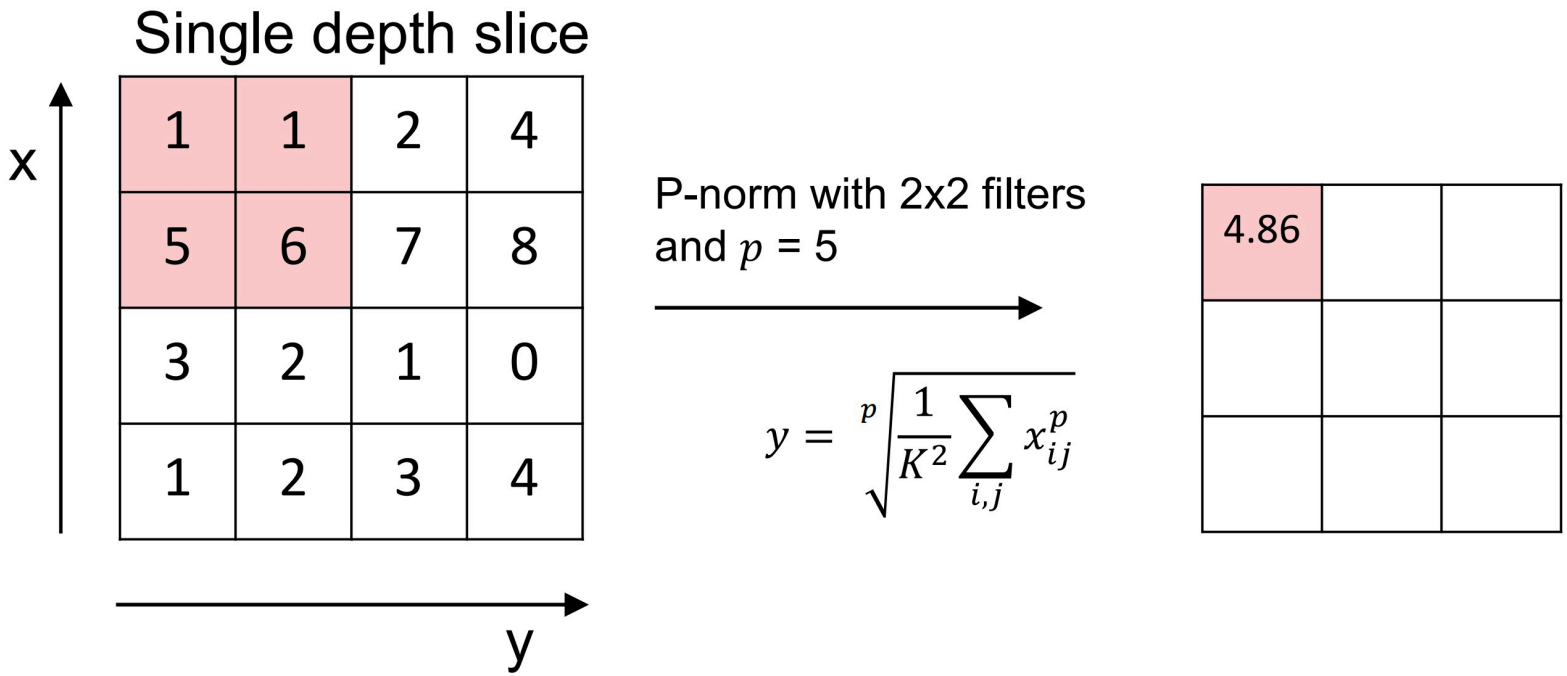
- Max pooling selects the largest from a pool of elements
- Pooling is performed by “scanning” the input

Alternative to Max pooling: Mean Pooling 平均池化



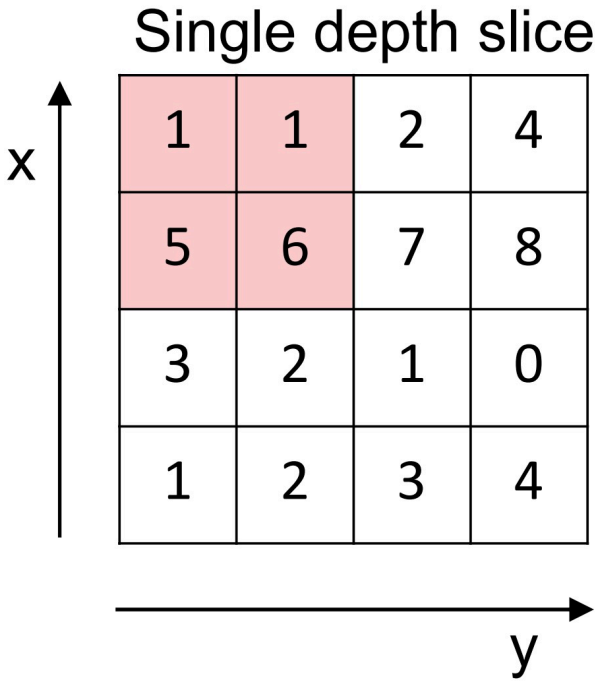
- Compute the mean of the pool, instead of the max

Alternative to Max pooling: p-norm

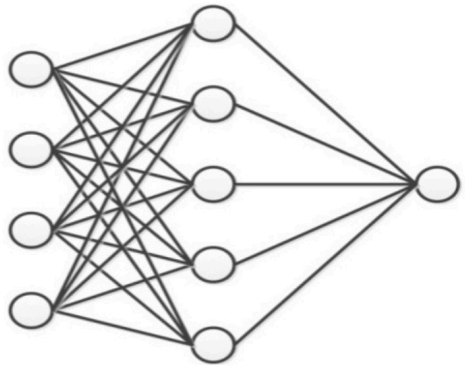


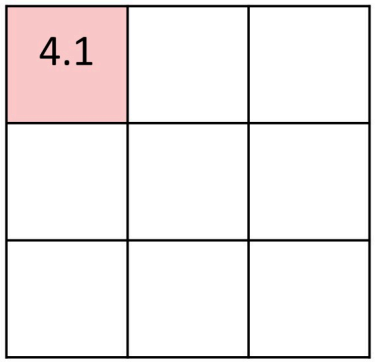
- Compute a p-norm of the pool

Other options



Network applies to each 2x2 block n this example





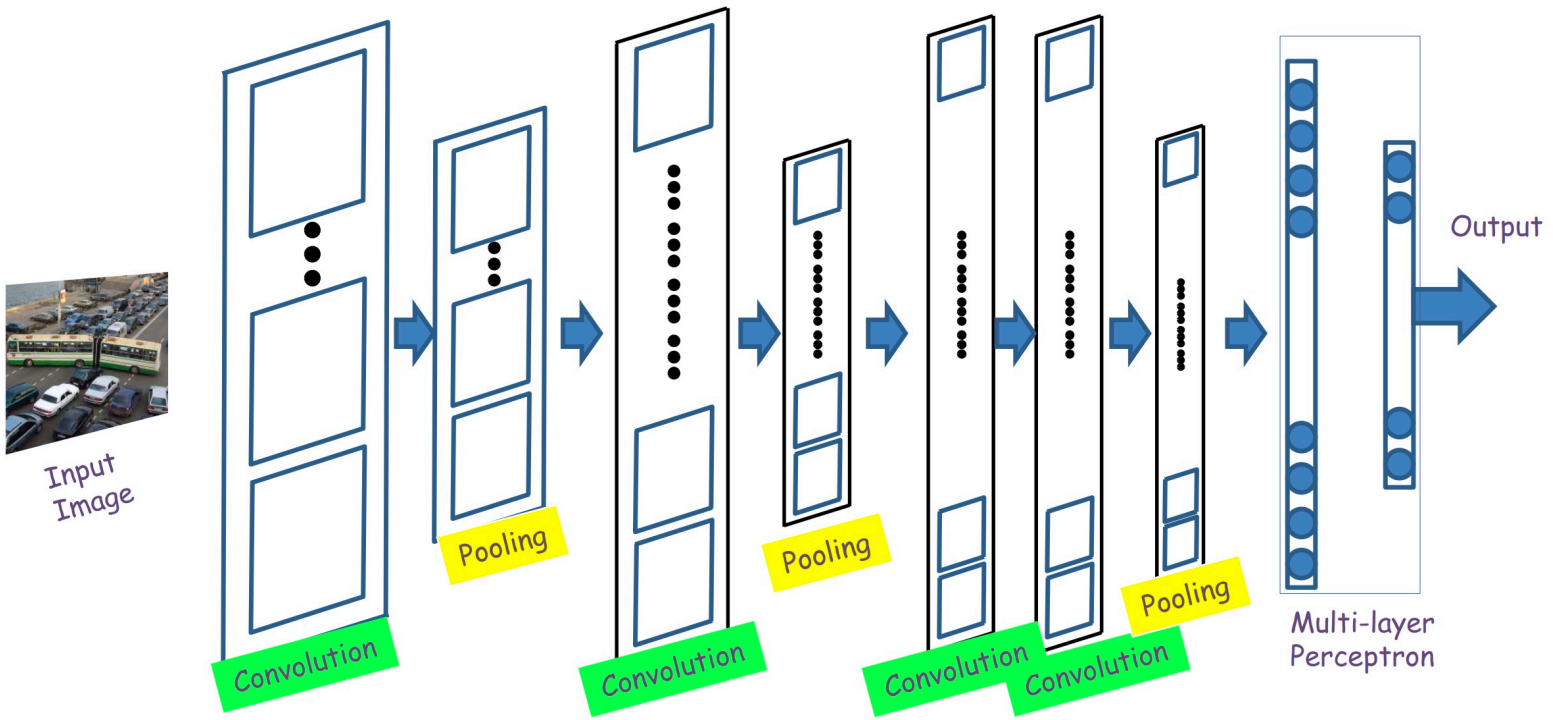
Network in network

- The pooling may even be a learned filter
  - The same network is applied on each block
    - (Again, a shared parameter network)

Pooling Summary

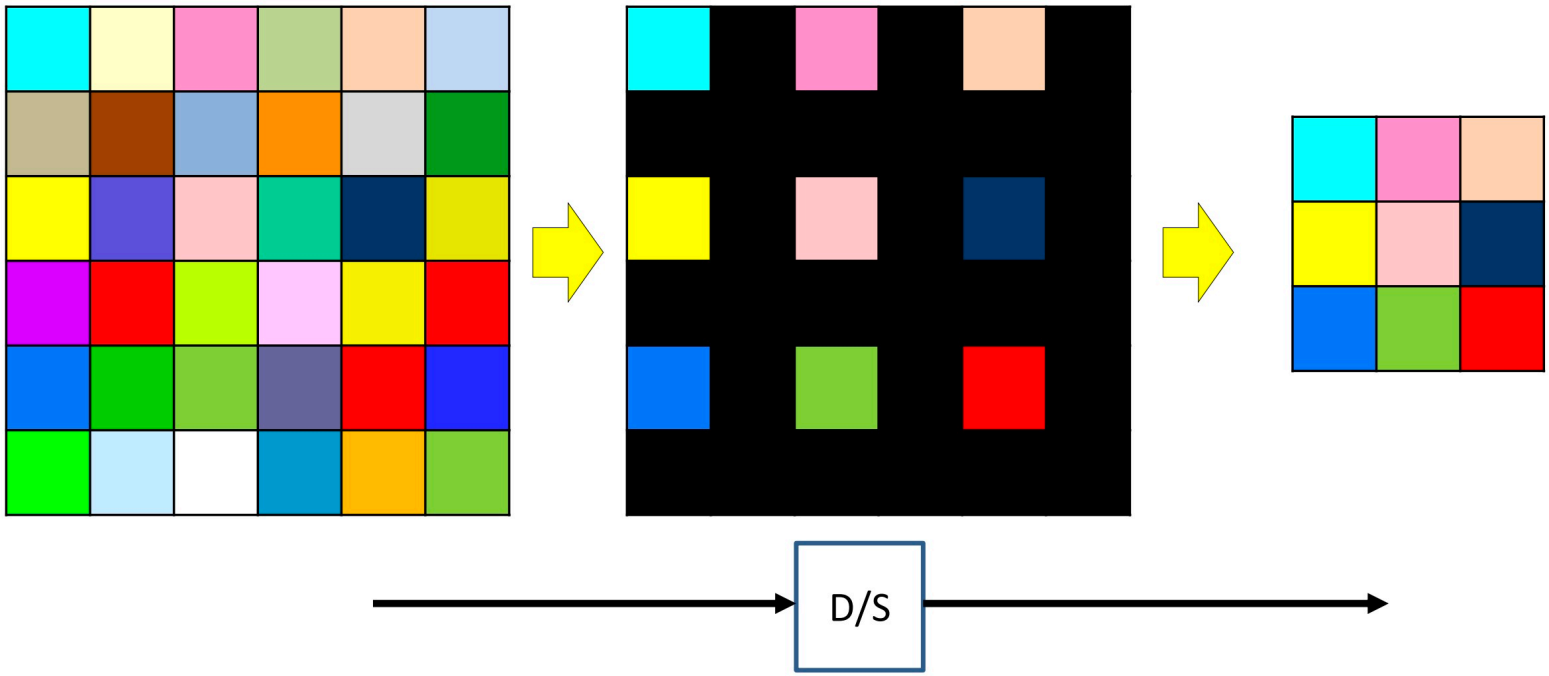
- Pooling layers "scan" the input using a "pooling" operation
  - E.g. selecting the max from a KxK block of input
- Each "pooling filter" scans an individual maps in the previous layer to produce an output "pooled map"
  - As many output maps as input maps
- For pooling we do not generally pad the edges

The types of layers considered so far



- So far we have only considered layers where the output size is approximately equal to input size
- There are two other operations that change the size of the output

The Downsampling Layer 下采样

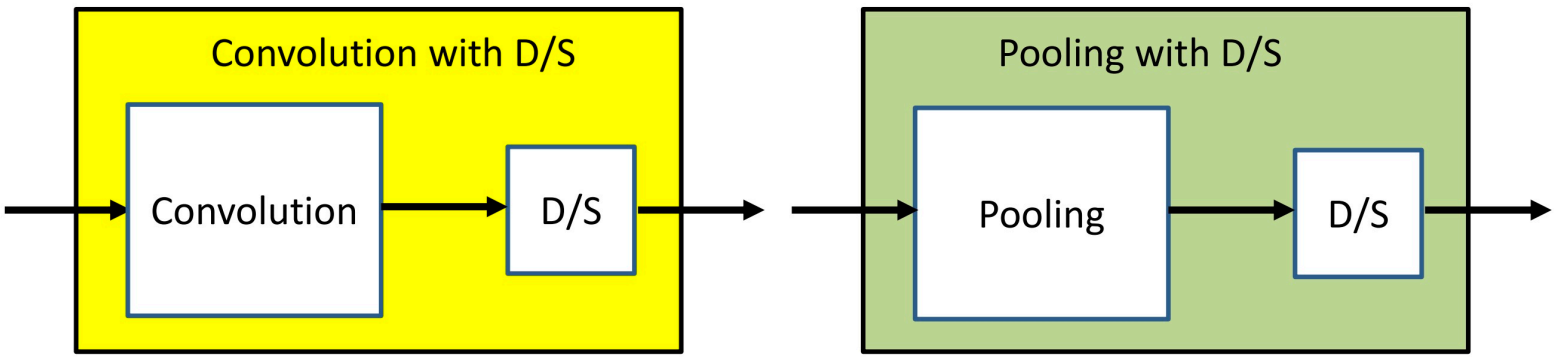


- A *downsampling* layer simply "drops" $N-1$ of N rows and columns for every map in the layer
  - Reducing the size of the map by factor S in every direction

Downsampling Pseudocode

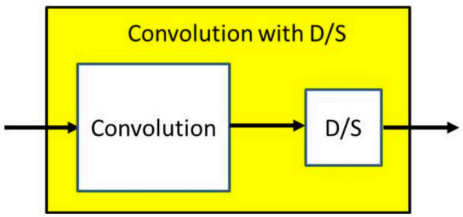
```
m = 1
for i = 1:S:W
    n = 1
    for j = 1:S:H
        y(m,n) = x(i,j)
        n++
    end
    m++
end
```

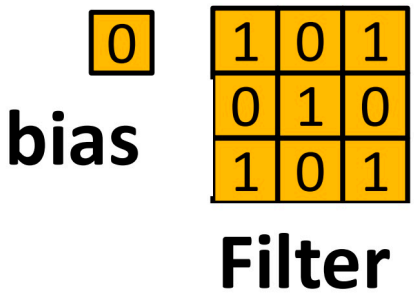
Downsampling in practice

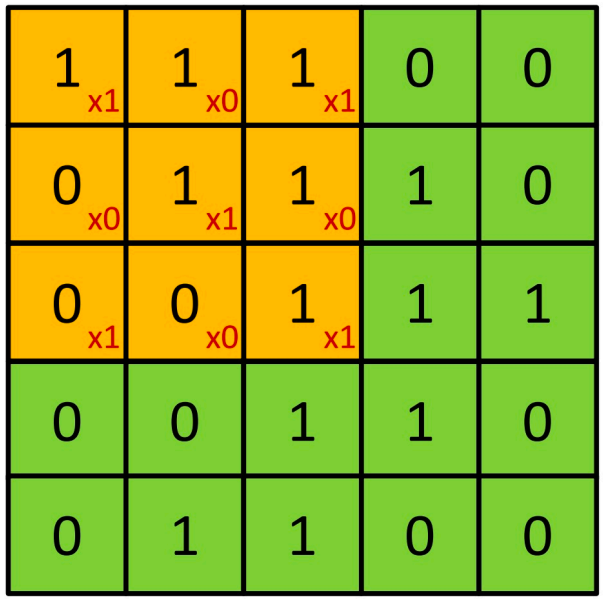


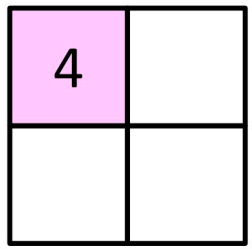
- In practice, the downsampling is combined with the layers just before it
  - Which could be convolutional or pooling layers

Downsampling after Convolution



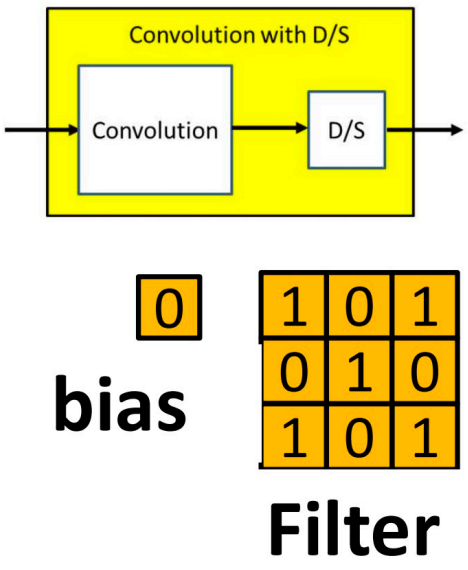


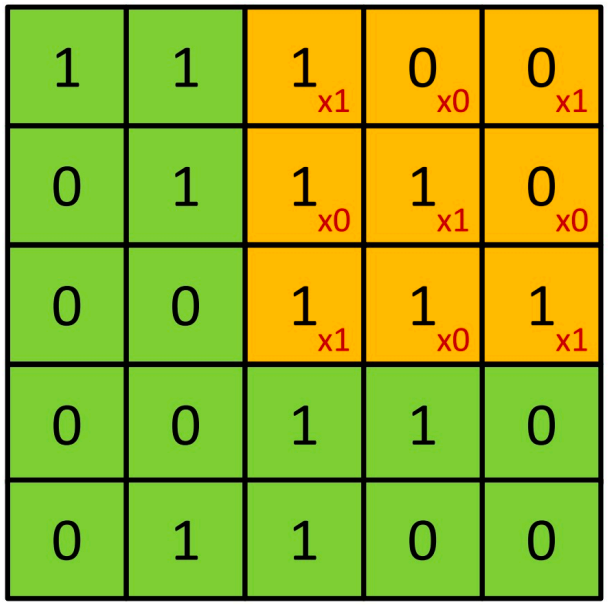




- Downsampling can be combined with convolutional into a single convolutional layer with convolution stride (步长) S

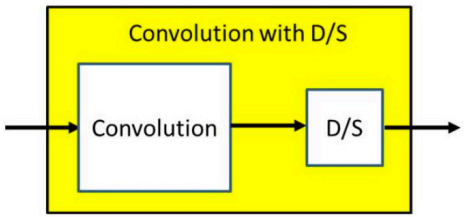
Downsampling after Convolution

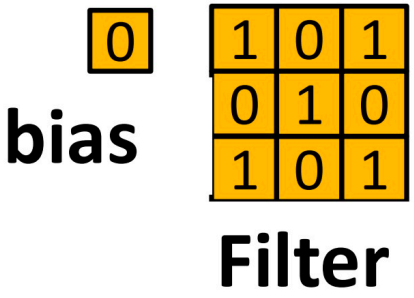


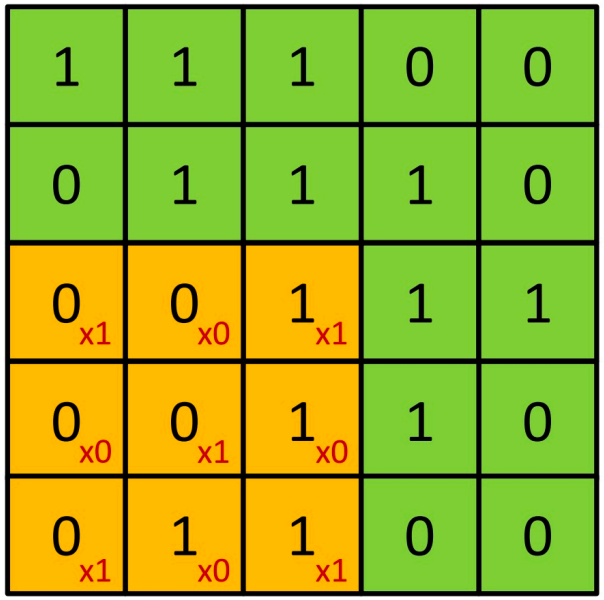


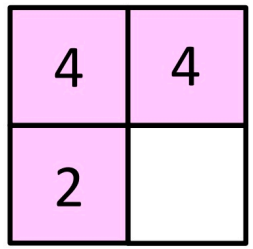
- Downsampling can be combined with convolutional into a single convolutional layer with convolution stride (步长) S

Downsampling after Convolution



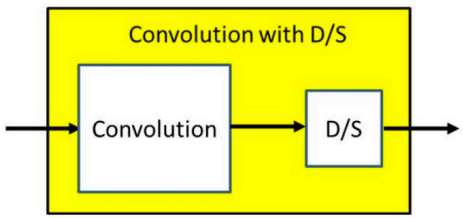


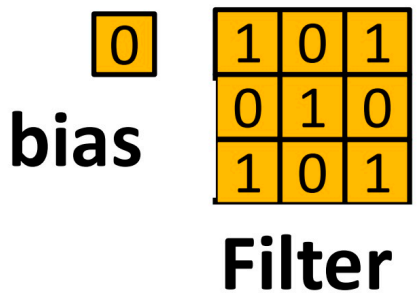


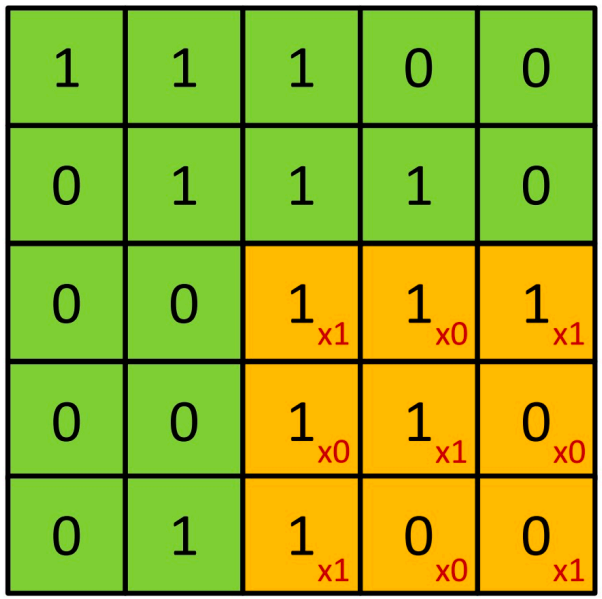


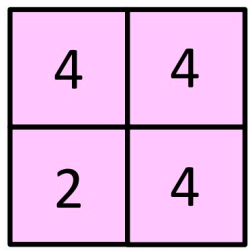
- Downsampling can be combined with convolutional into a single convolutional layer with convolution stride (步长) S

Downsampling after Convolution



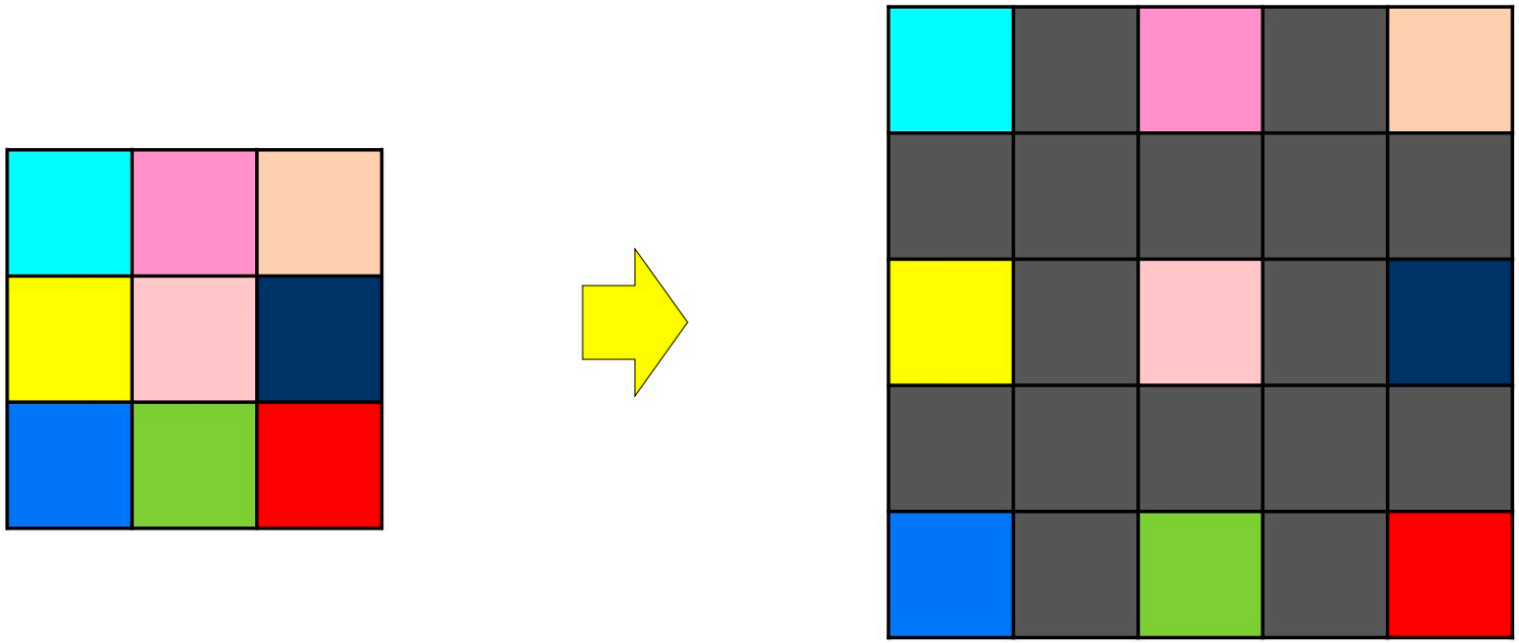






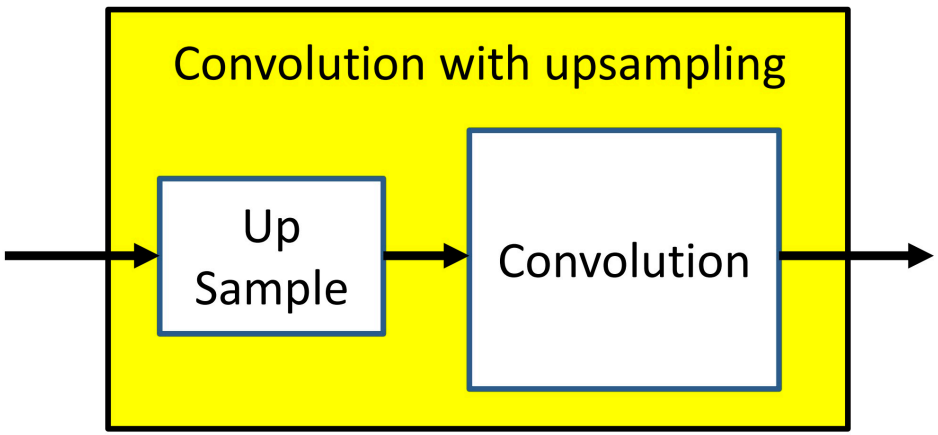
- Downsampling can be combined with convolutional into a single convolutional layer with convolution stride (步长) S

The Upsampling Layer 上采样



- A upsampling layer simply introduces $S-1$ rows and columns for every map in the layer
  - Increase the size of the map by factor S in every direction
- Used explicitly to increase the map size by a uniform factor

The Upsampling Layer 上采样

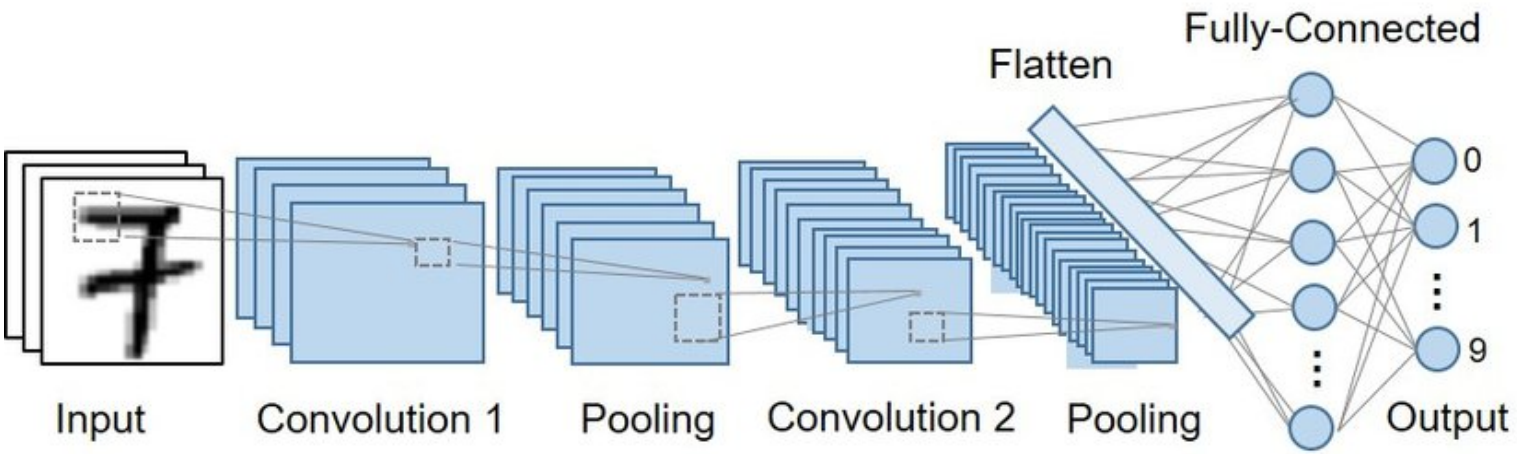


- Upsampling layer is often followed by a convolutional layer
  - It is not useful to follow it by a pooling layer
  - It is also not useful as the final layer of a CNN

Resampling Summary

- Map sizes can be changed by downsampling or upsampling
  - Downsampling: Drop $S-1$ of S rows and columns
  - Upsampling: Insert $S-1$ zeros between rows/columns
- Downsampling typically follows convolution or pooling
  - Reduces the size of the maps
- Upsampling occurs before convolution
  - Increases the size of the map
  - Does not generally occur before pooling layers

Digit classification 手写数字识别



Thank you!