Machine learning

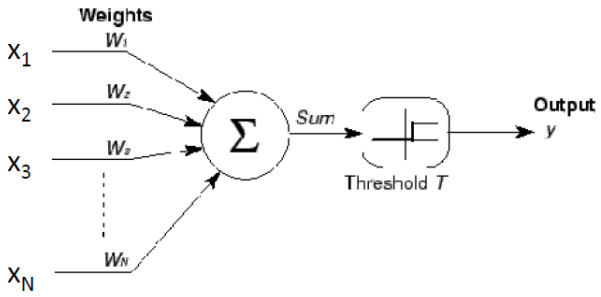
Deep Neural Networks
convolutional networks (CNN)

Mohammad-Reza A. Dehaqani

dehaqani@ut.ac.ir

Slides are adopted form CMU deep NN course

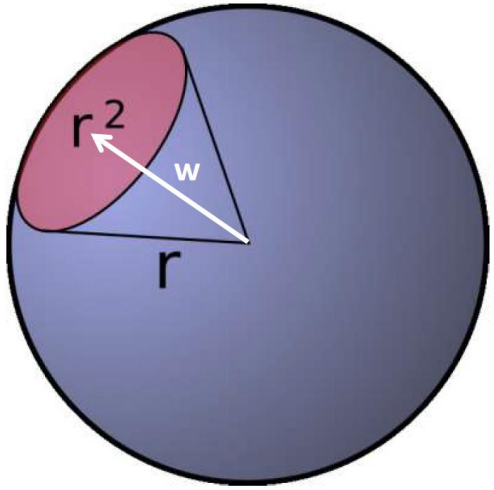
The weight as a "template"



$$X^T W > T$$

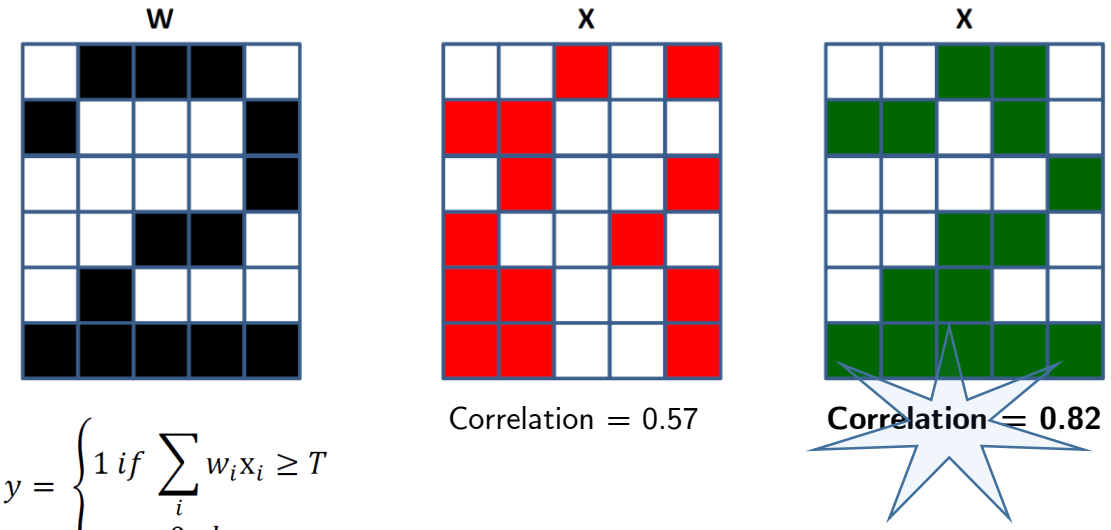
$$\cos\theta > \frac{T}{|X|}$$

$$\theta < cos^{-1}\left(\frac{T}{|X|}\right)$$



- The perceptron fires if the input is within a specified angle of the weight
- Neuron fires if the input vector is close enough to the weight vector.
  - If the input pattern matches the weight pattern closely enough
- If the correlation between the weight pattern and the inputs exceeds a threshold, fire
  - The perceptron is a **correlation filter!**

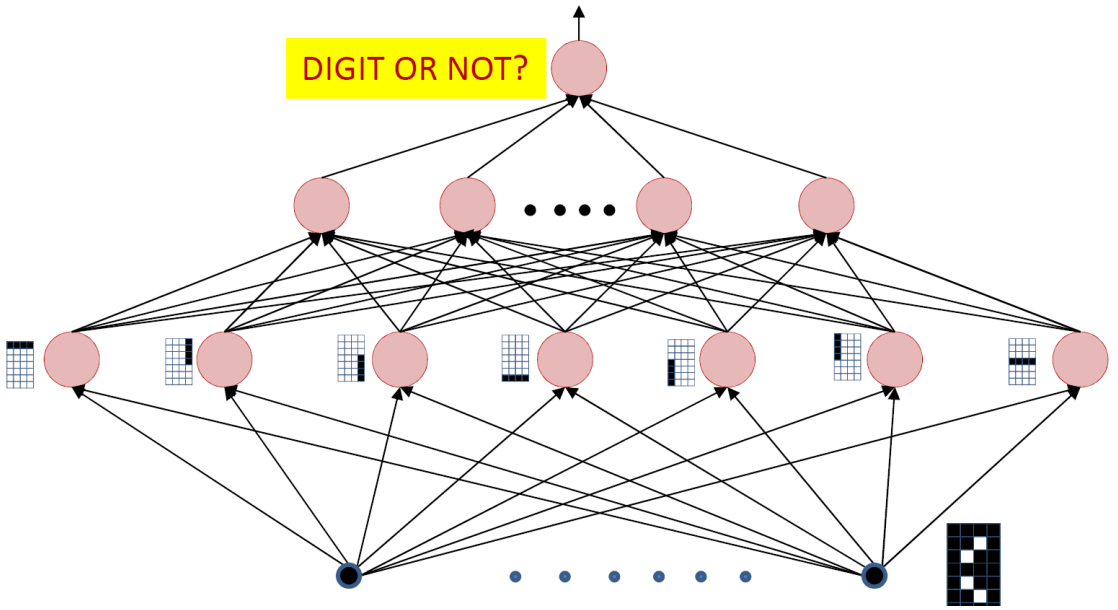
The MLP as a Boolean function over feature detectors



correlation filter

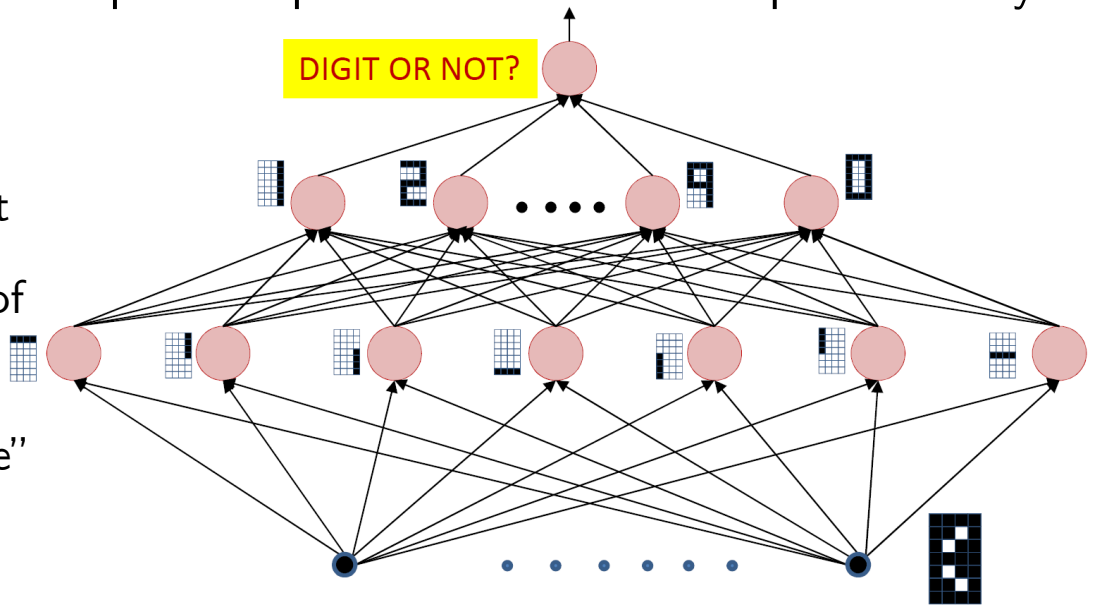
$$y = \begin{cases} 1 \; if \; \sum_i w_i x_i \geq T \\ 0 \; else \end{cases}$$

- The input layer comprises "feature detectors"
  - Detect if certain patterns have occurred in the input
- The network is a Boolean function over the feature detectors
  - I.e. it is important for the first layer to capture relevant patterns



The MLP as a cascade of feature detectors

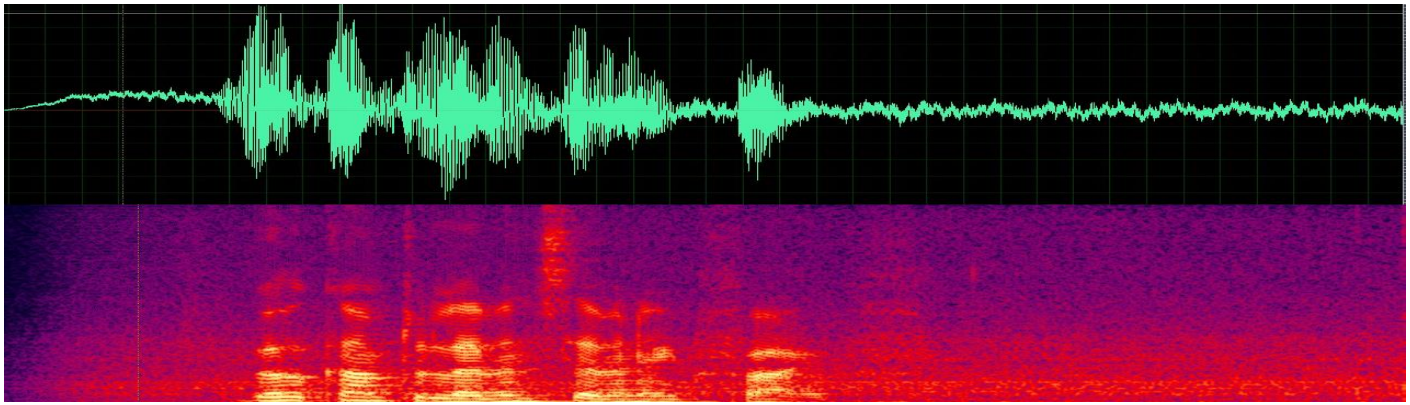
- Higher level neurons compose complex templates from features represented by lower-level neurons



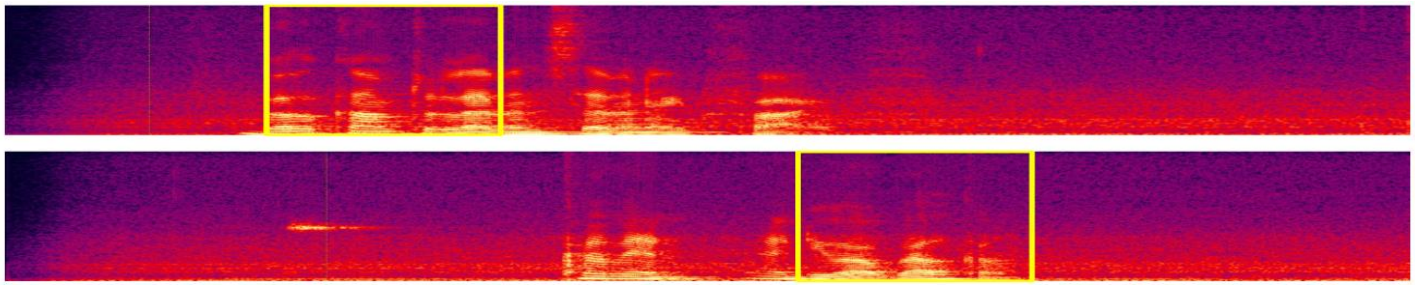
- They detect patterns in the input
- Layers in an MLP are detectors of increasingly complex patterns
- The representation of "acceptable" input patterns is distributed over the layers of the network

- MLP in classification:
  - The network will fire if the combination of the detected basic features
  - matches an "acceptable" pattern for a desired class of signal
    - E.g. Appropriate combinations of (Nose, Eyes, Eyebrows, Cheek, Chin) → Face

Changing gears..

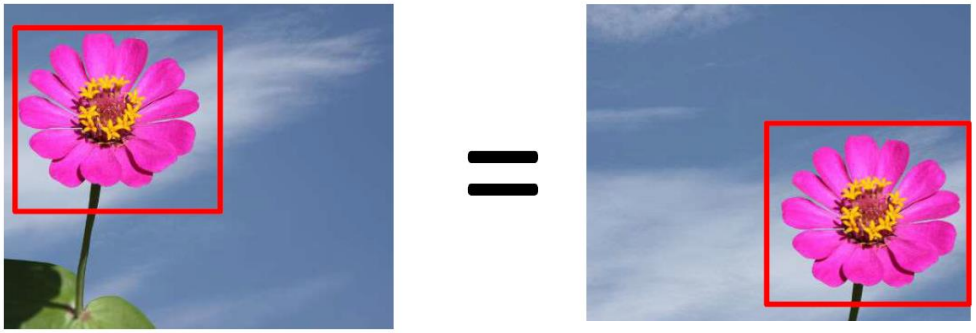


- Does this signal contain the word "Welcome"?
- Trivial solution: Train an MLP for the entire recording.
  - MLP Network that finds a "welcome" in the top recording will not find it in the lower one. Unless trained with both which require a very large network and big training data



- We need a simple network that will fire regardless of the location of "Welcome"

The need for shift invariance



- In many problems the location of a pattern is not important

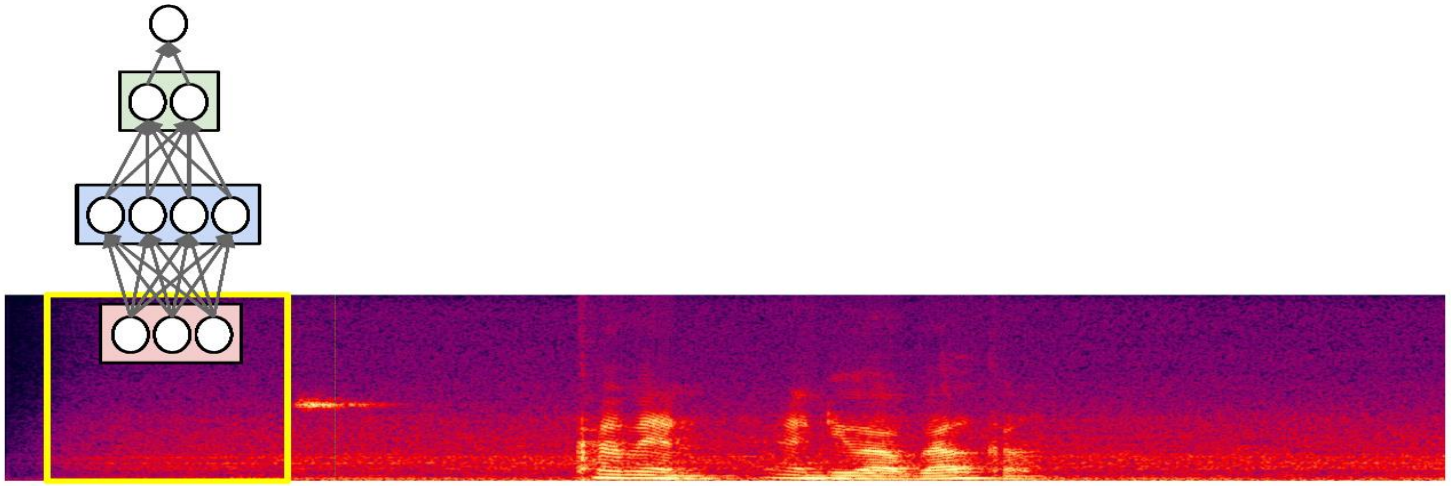
  – Only the presence of the pattern

- Conventional MLPs are sensitive to the location of the pattern (two images result in an entirely different input to the MLP)

Solution: Scan

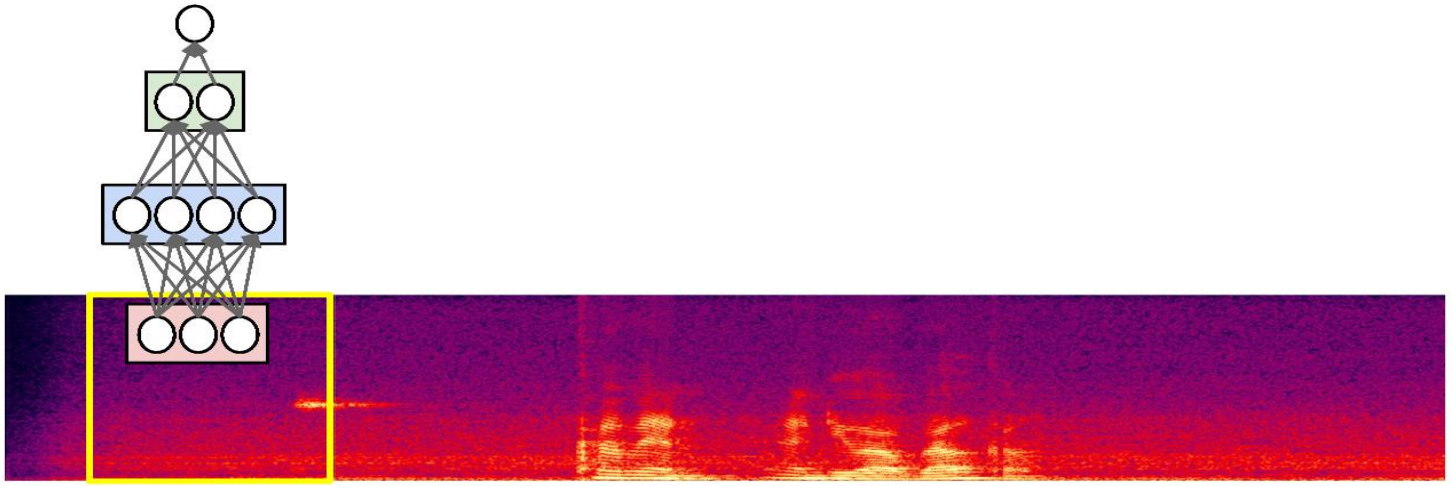
- Scan for the target word
- – The spectral time-frequency components in a "window" are input to a "welcome-detector" MLP

Solution: Scan



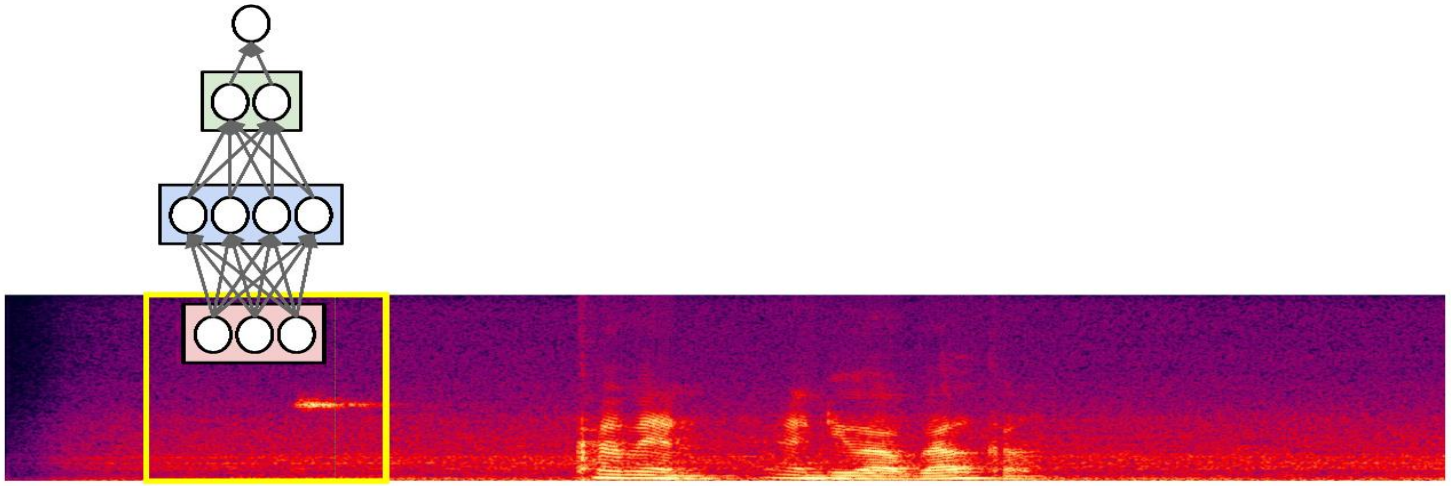
- Scan for the target word
- – The spectral time-frequency components in a "window" are input to a "welcome-detector" MLP

Solution: Scan



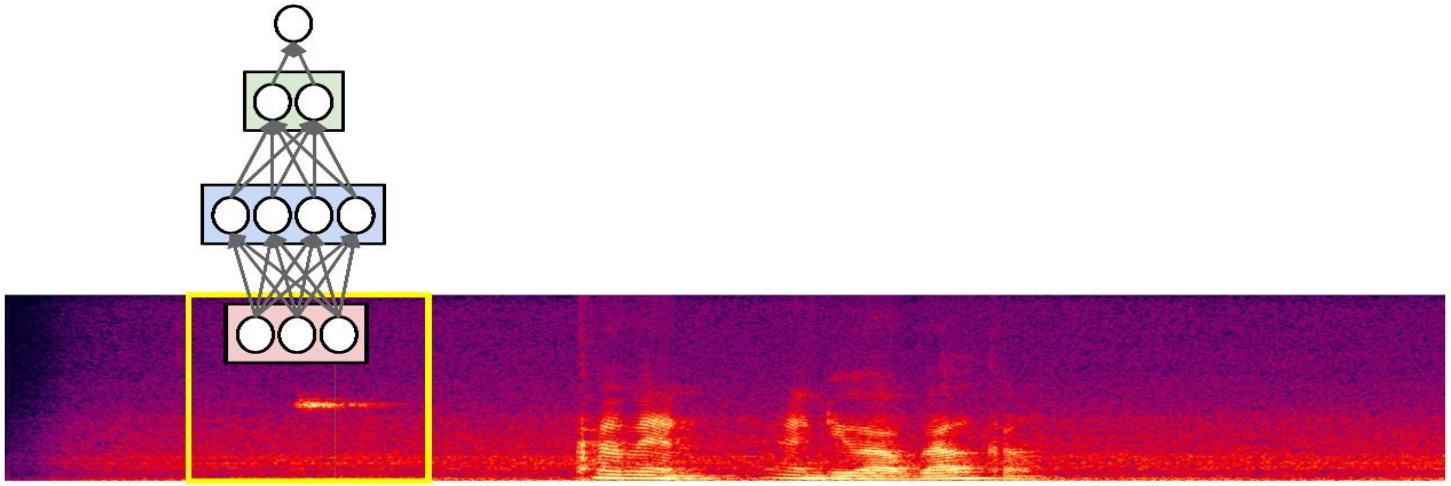
- Scan for the target word
- – The spectral time-frequency components in a "window" are input to a "welcome-detector" MLP

Solution: Scan



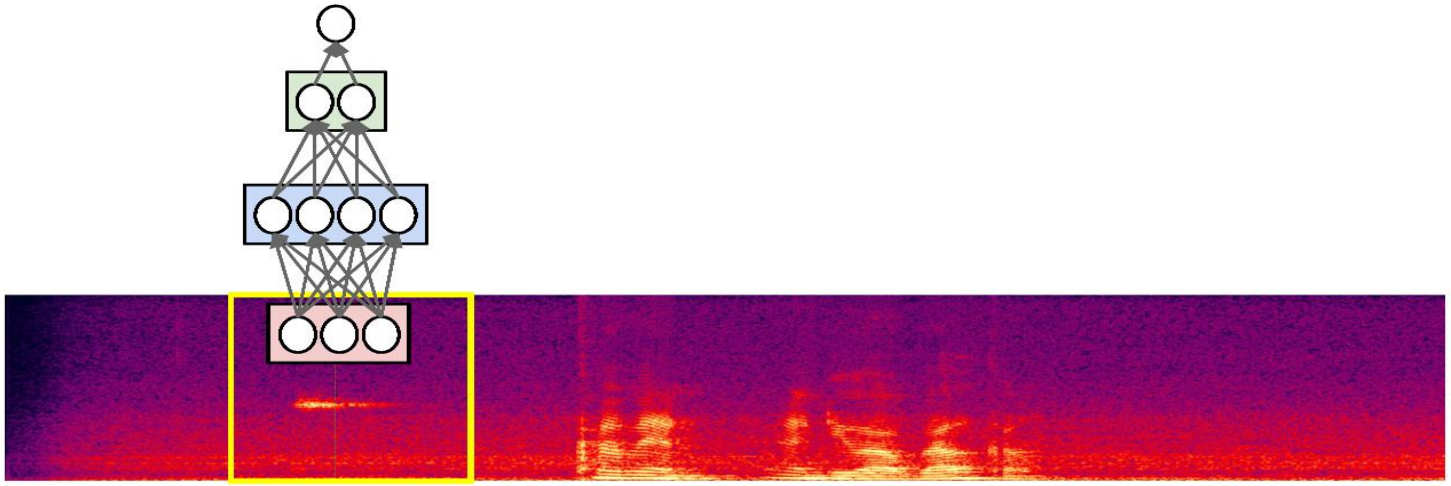
- Scan for the target word
- – The spectral time-frequency components in a "window" are input to a "welcome-detector" MLP

Solution: Scan



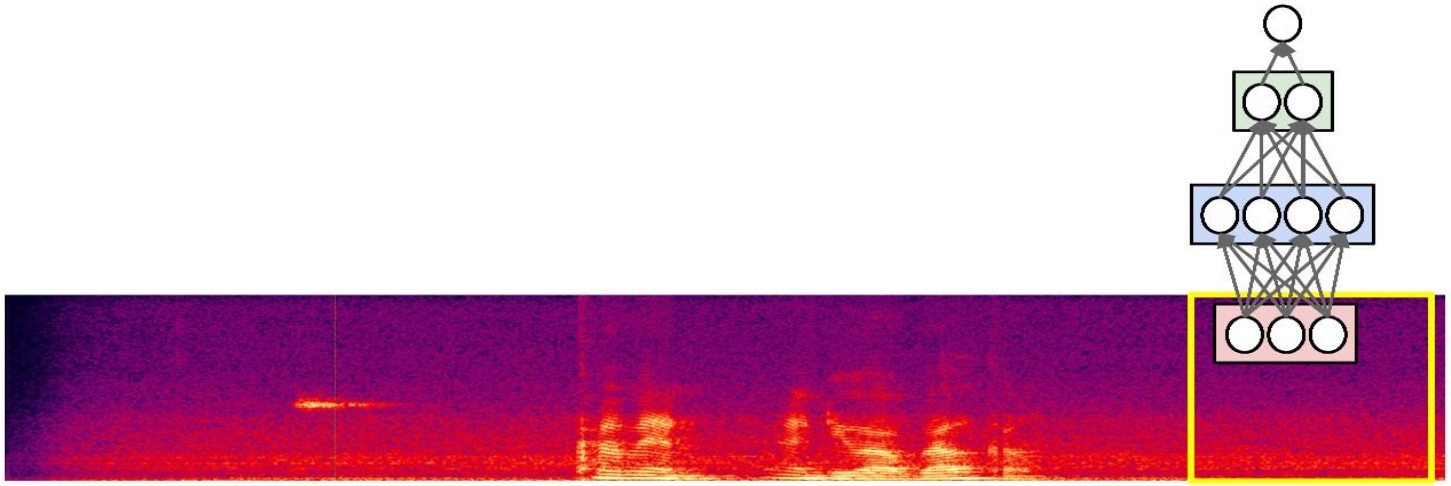
- Scan for the target word
- – The spectral time-frequency components in a "window" are input to a "welcome-detector" MLP

Solution: Scan



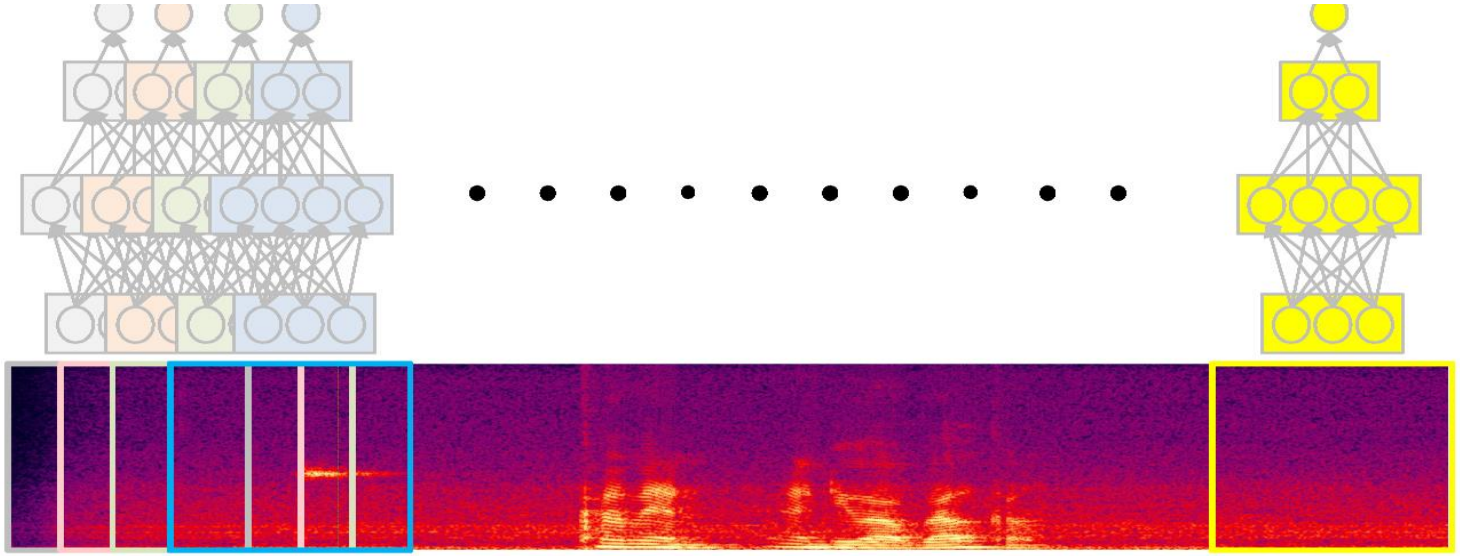
- Scan for the target word
- – The spectral time-frequency components in a "window" are input to a "welcome-detector" MLP

Solution: Scan



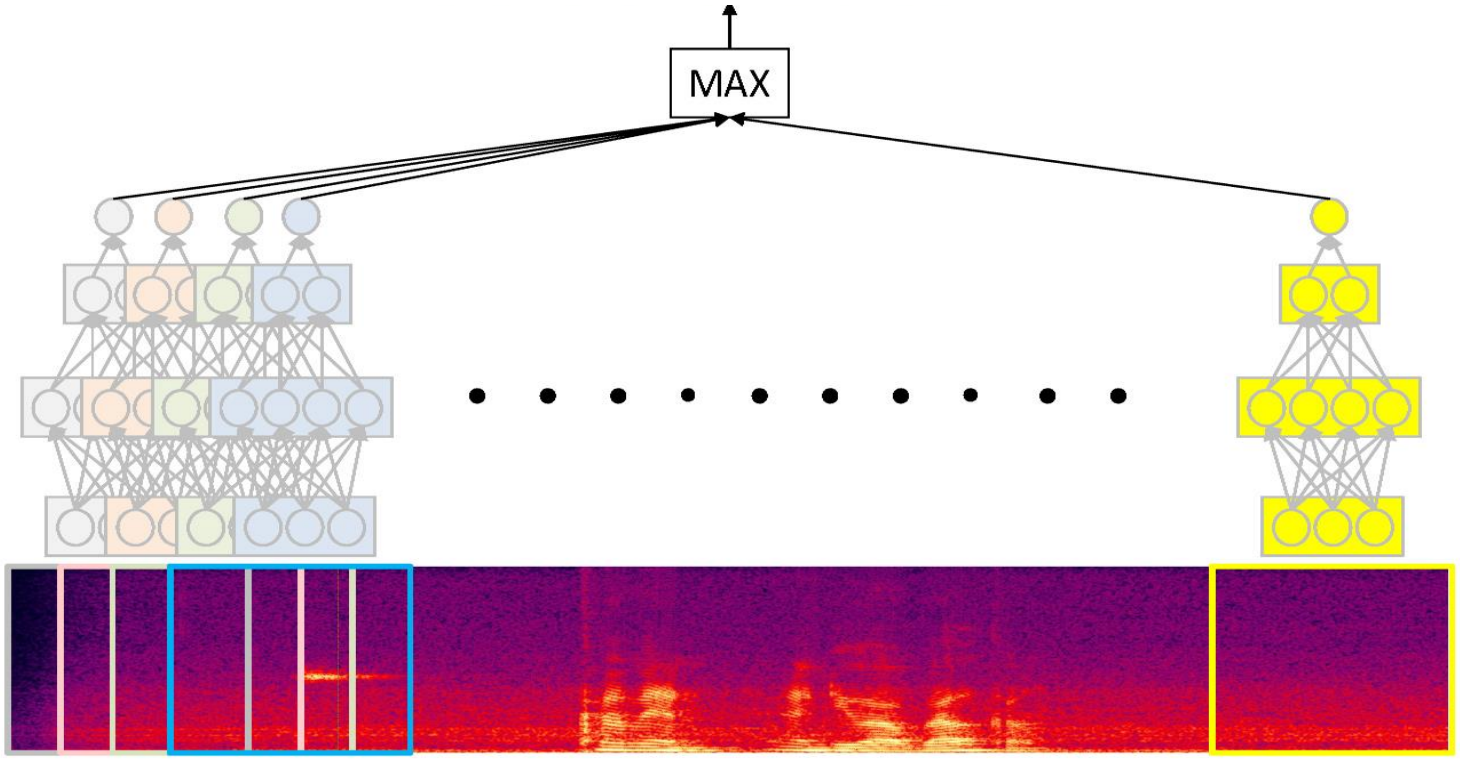
- Scan for the target word
- – The spectral time-frequency components in a "window" are input to a "welcome-detector" MLP

Solution: Scan



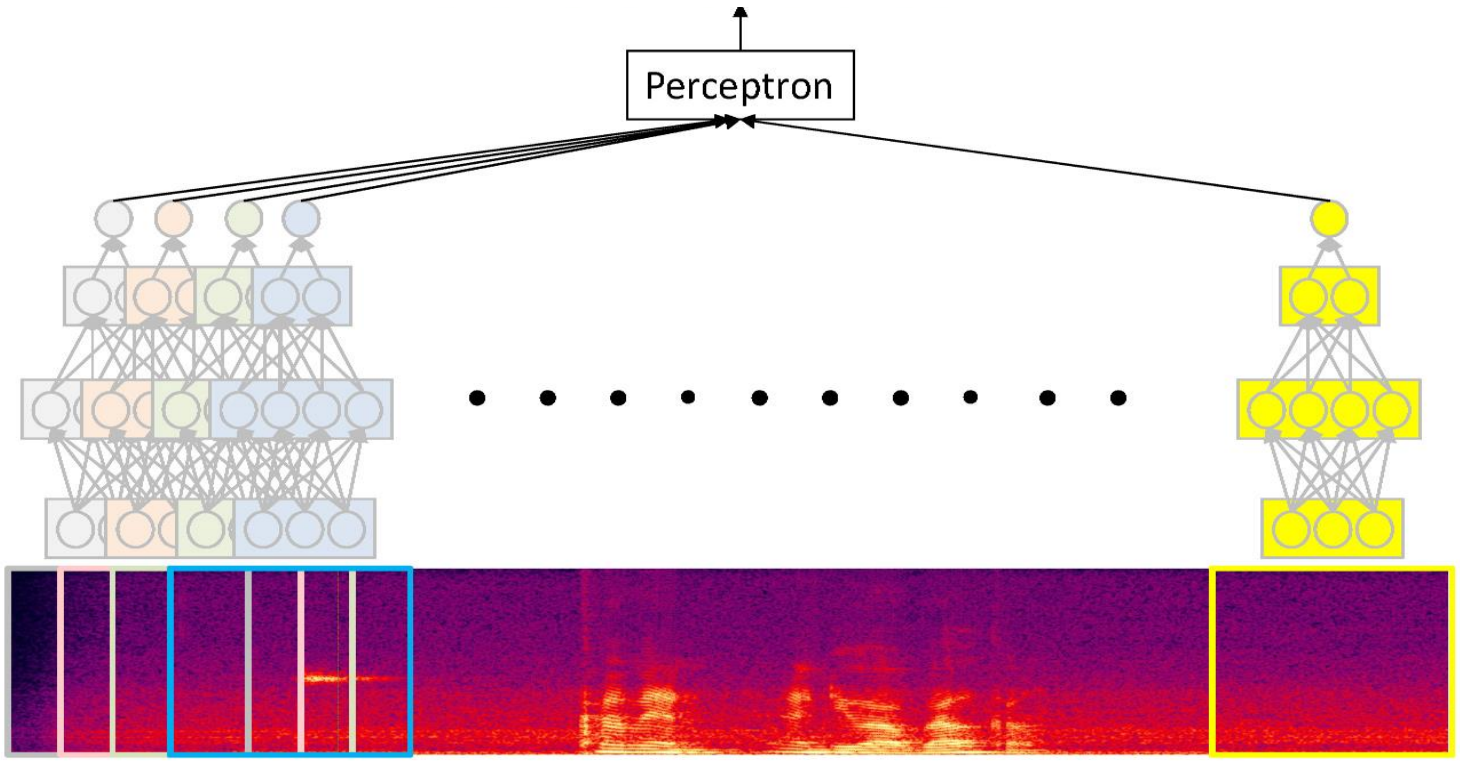
- "Does welcome occur in this recording?"
  - We have classified many "windows" individually
  - "Welcome" may have occurred in any of them

Solution: Scan



- "Does welcome occur in this recording?"
  - Maximum of all the outputs (Equivalent of Boolean OR)

Solution: Scan

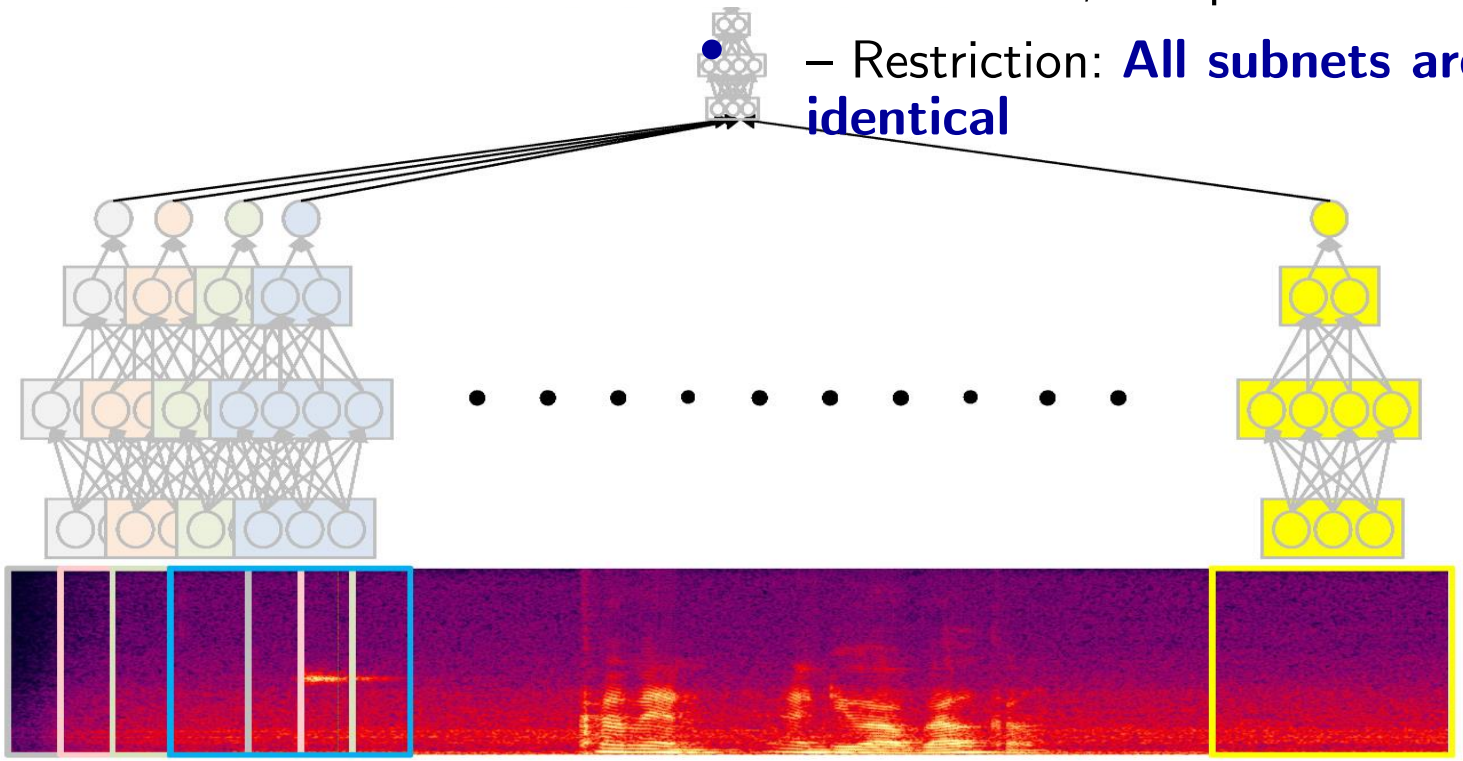


- "Does welcome occur in this recording?"
  - Maximum of all the outputs (Equivalent of Boolean OR)
  - Or a proper softmax/logistic

    Finding a welcome in adjacent windows makes it more likely that we didn't find noise

Solution: Scan

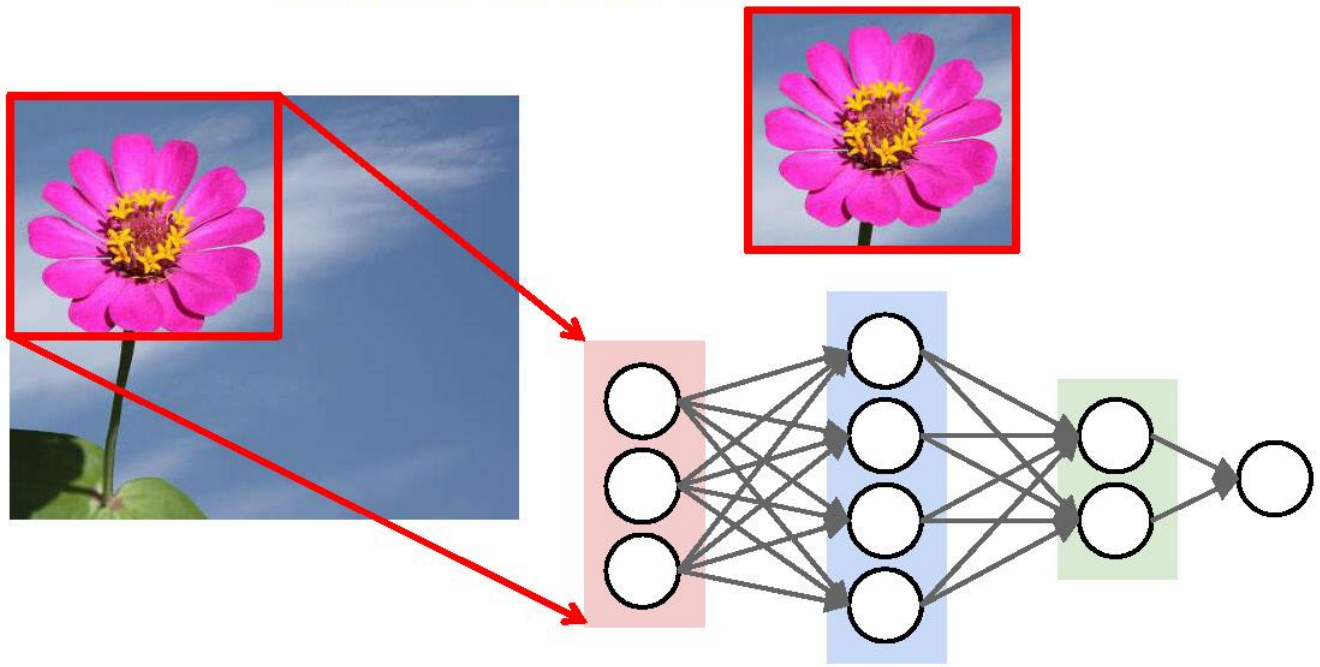
- The entire operation can be viewed as one giant network (With many subnetworks, one per window)
- – Restriction: **All subnets are identical**



- "Does welcome occur in this recording?"
  - Maximum of all the outputs (Equivalent of Boolean OR)
  - – Or a proper softmax/logistic

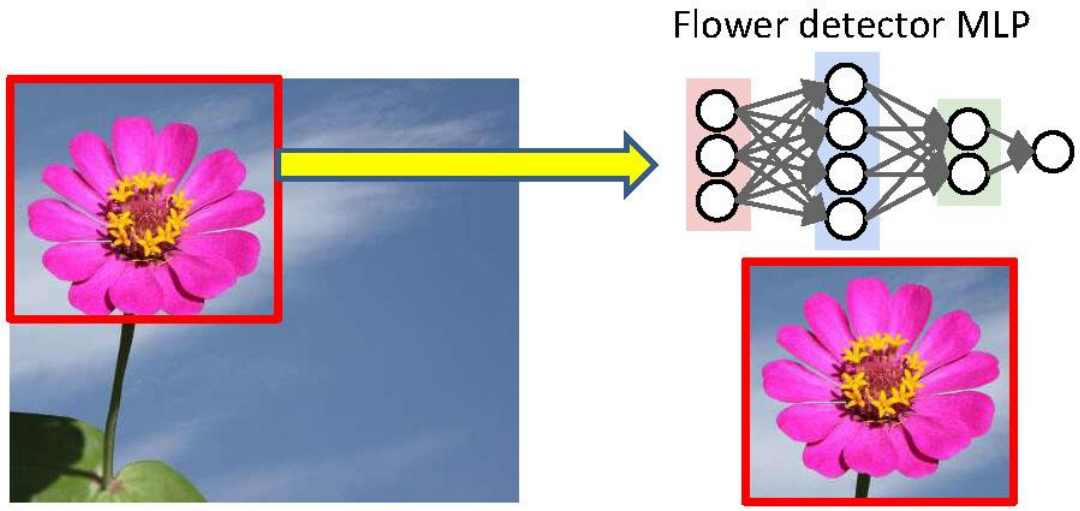
    Adjacent windows can combine their evidence
  - – Or even an MLP

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



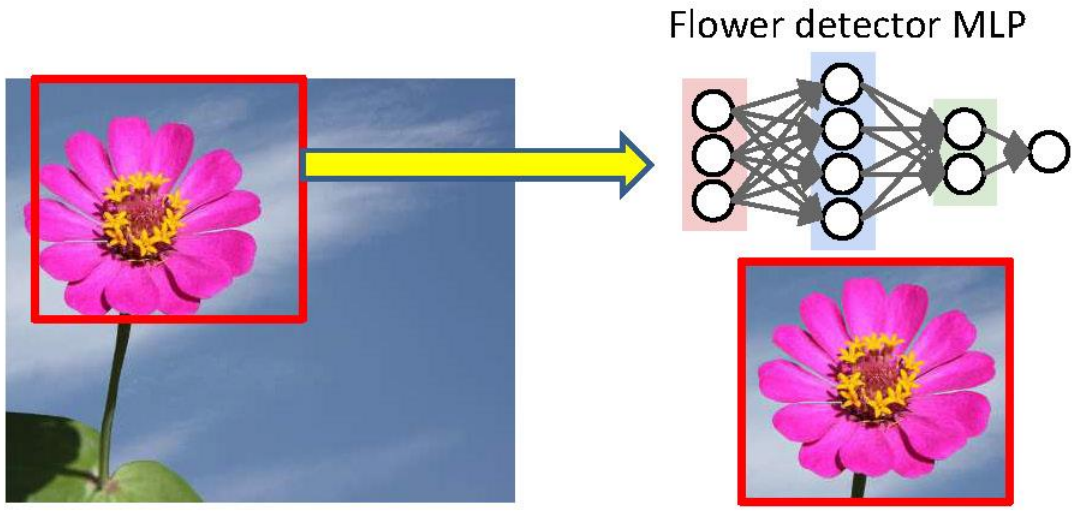
- Scan for the desired object
  - "Look" for the target object at each position

Solution: Scan



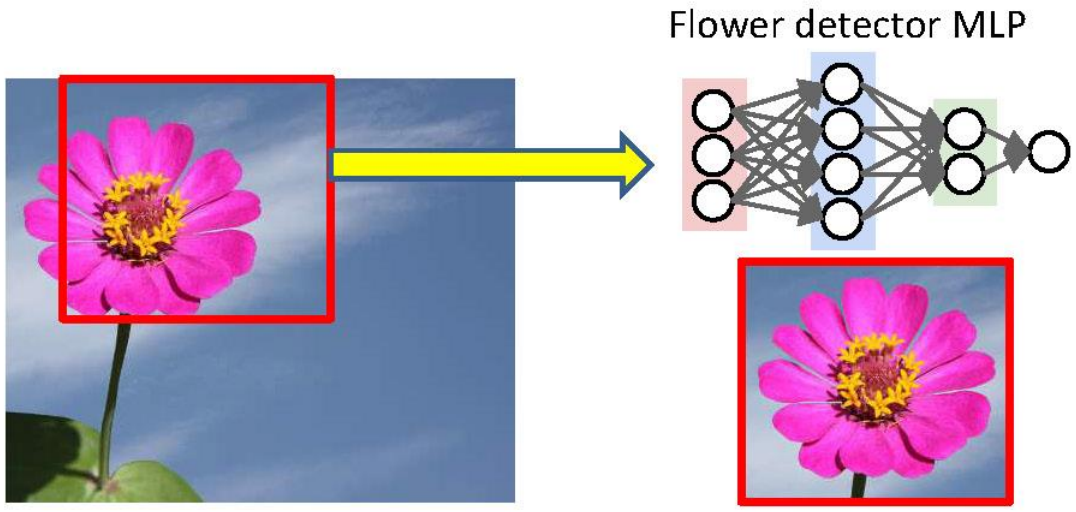
- Scan for the desired object

Solution: Scan



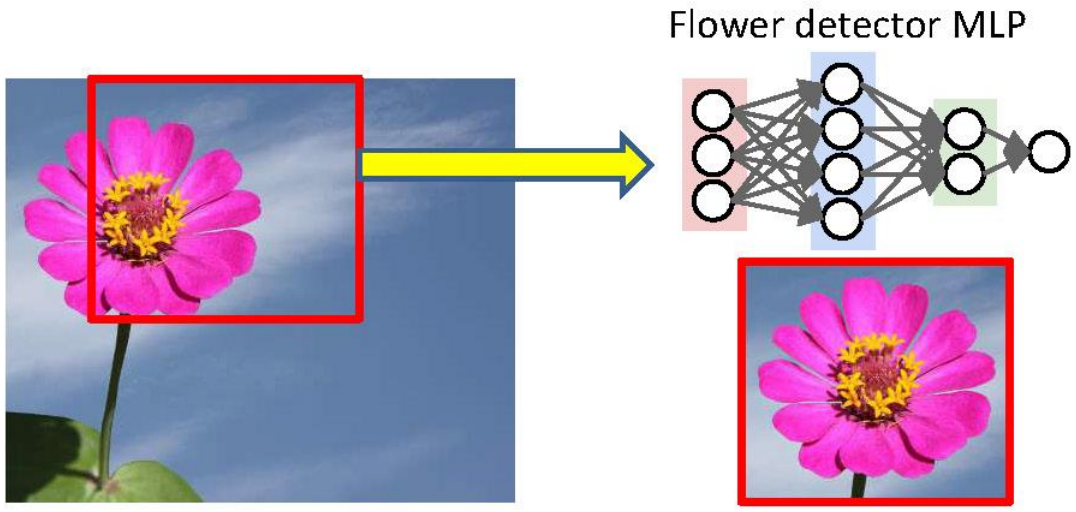
- Scan for the desired object

Solution: Scan



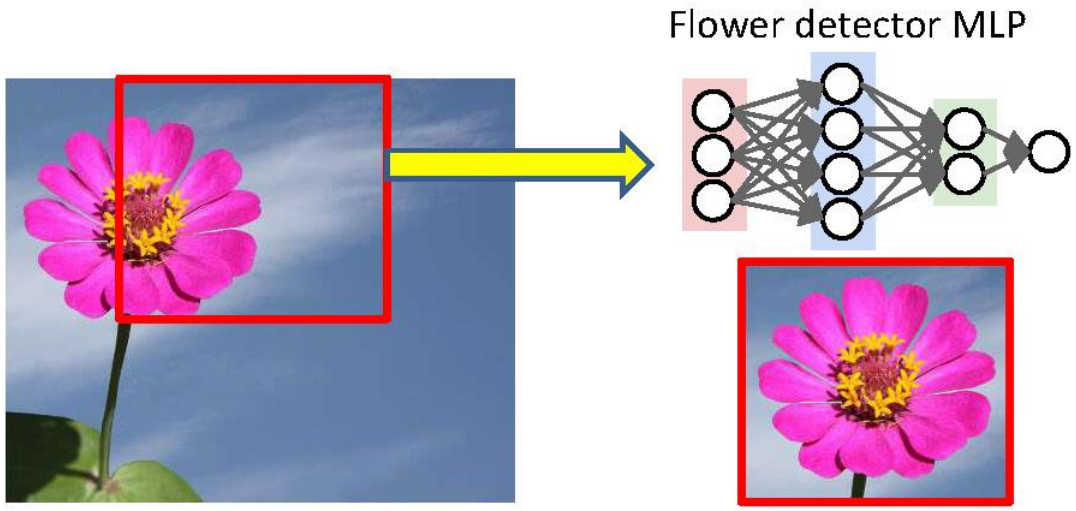
- Scan for the desired object

Solution: Scan



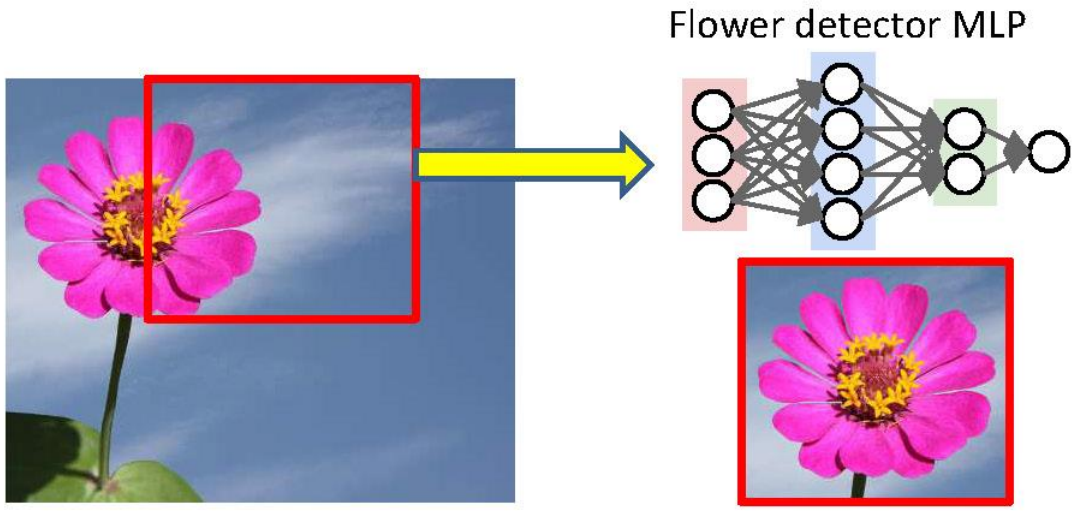
- Scan for the desired object

Solution: Scan



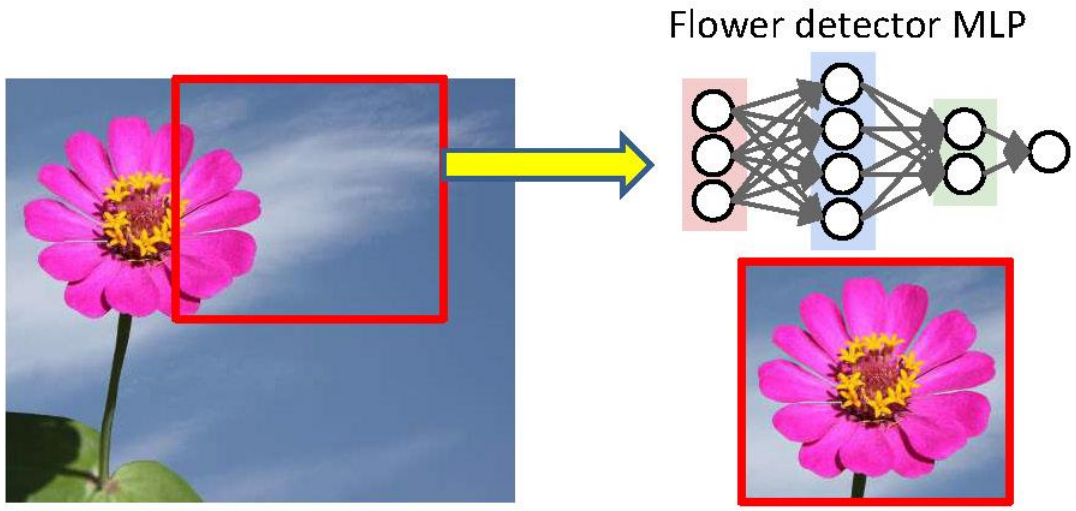
- Scan for the desired object

Solution: Scan



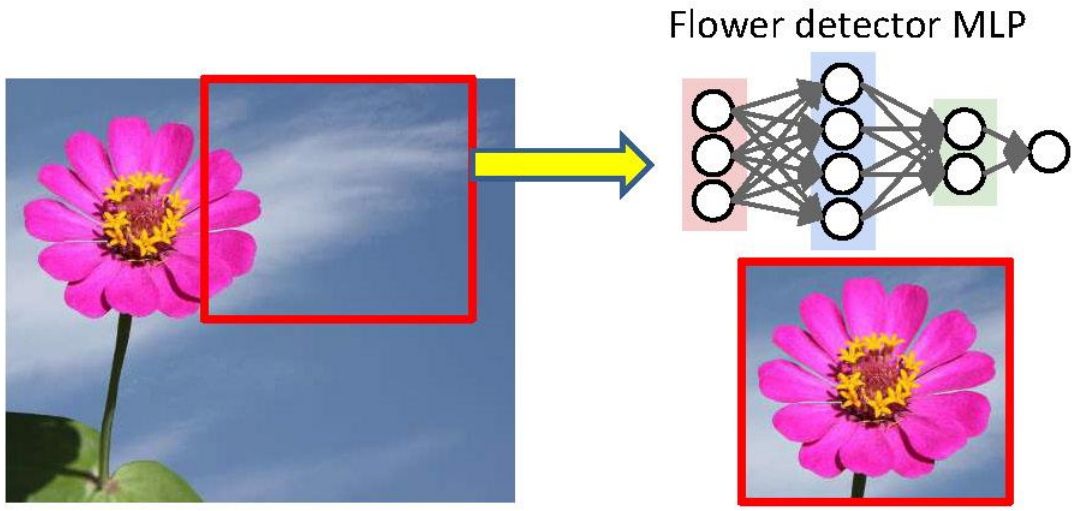
- Scan for the desired object

Solution: Scan



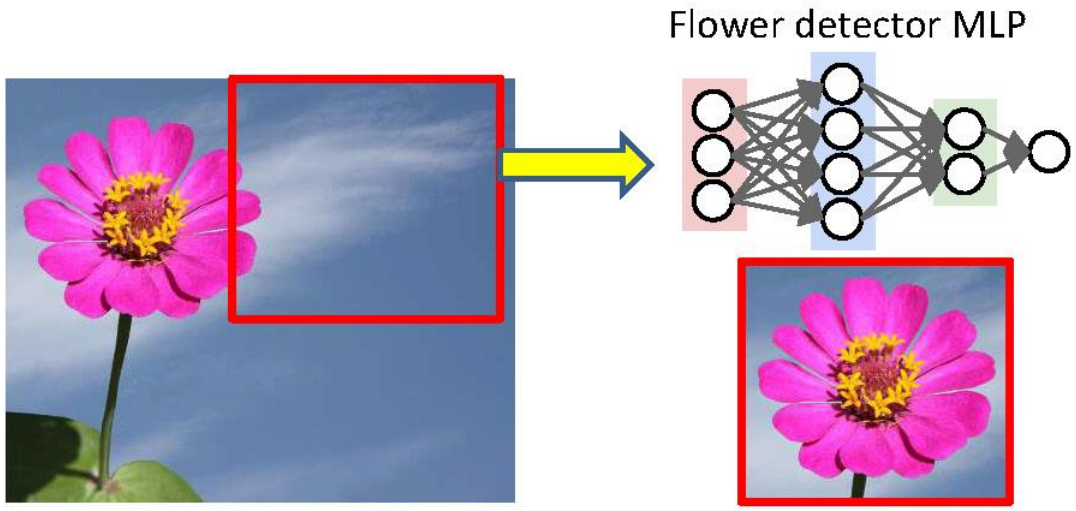
- Scan for the desired object

Solution: Scan



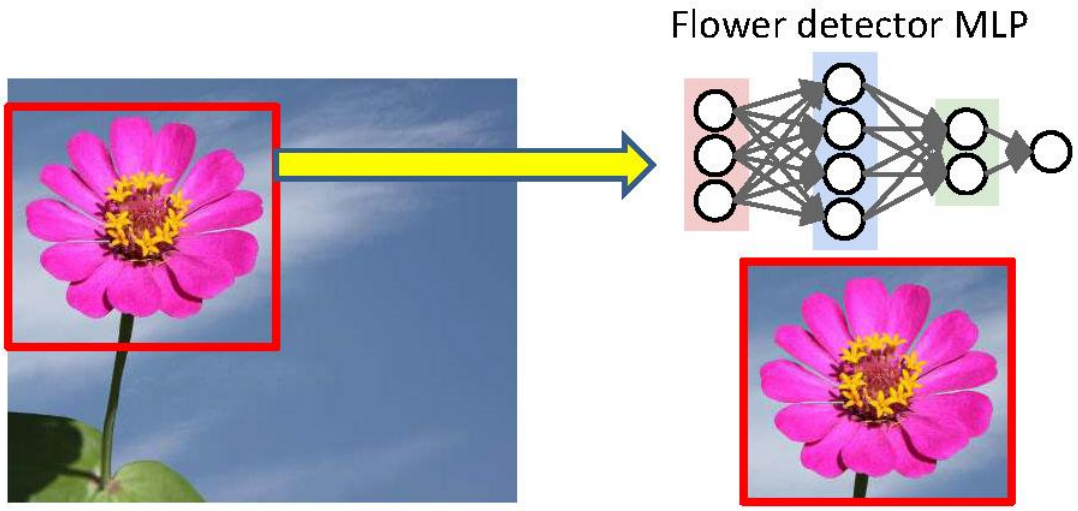
- Scan for the desired object

Solution: Scan



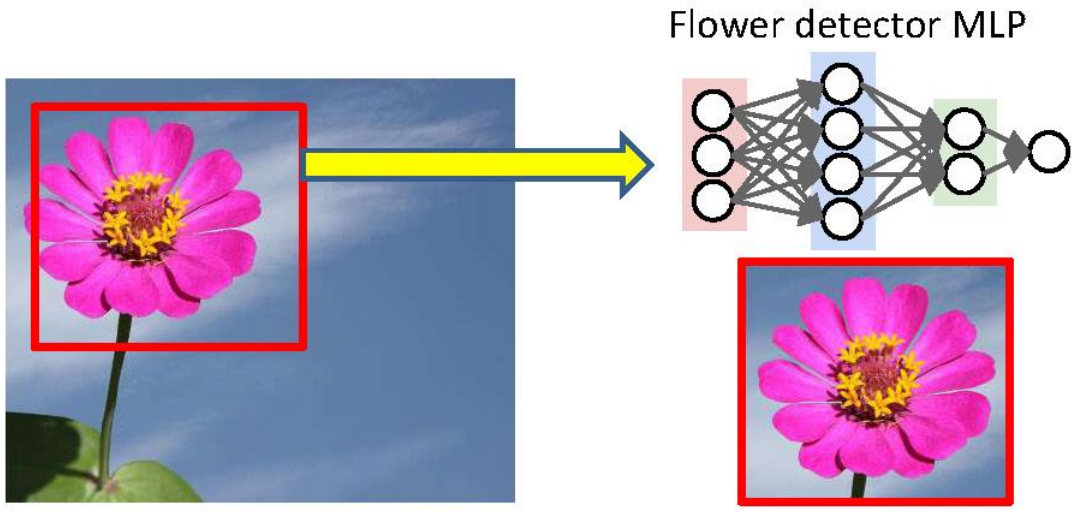
- Scan for the desired object

Solution: Scan



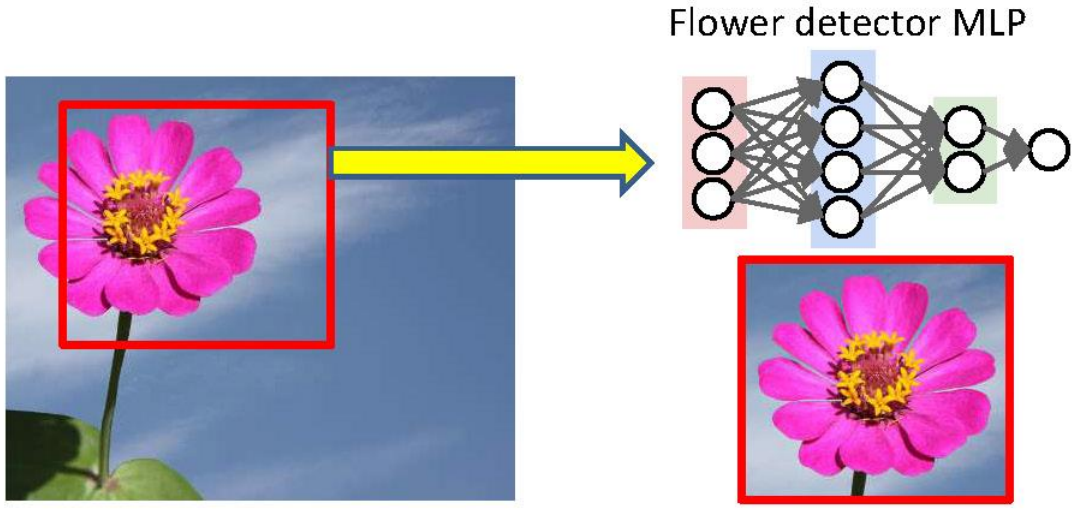
- Scan for the desired object

Solution: Scan



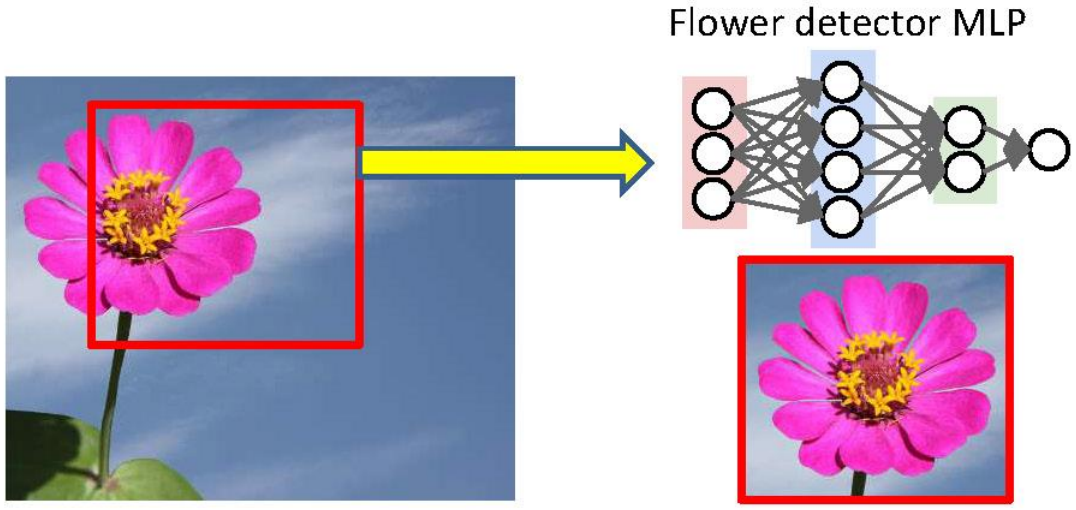
- Scan for the desired object

Solution: Scan



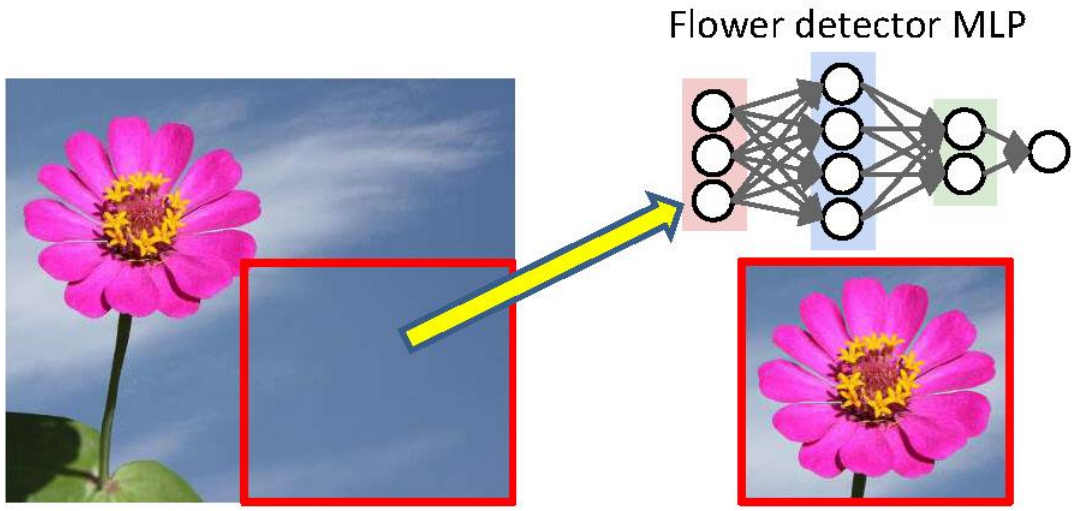
- Scan for the desired object

Solution: Scan



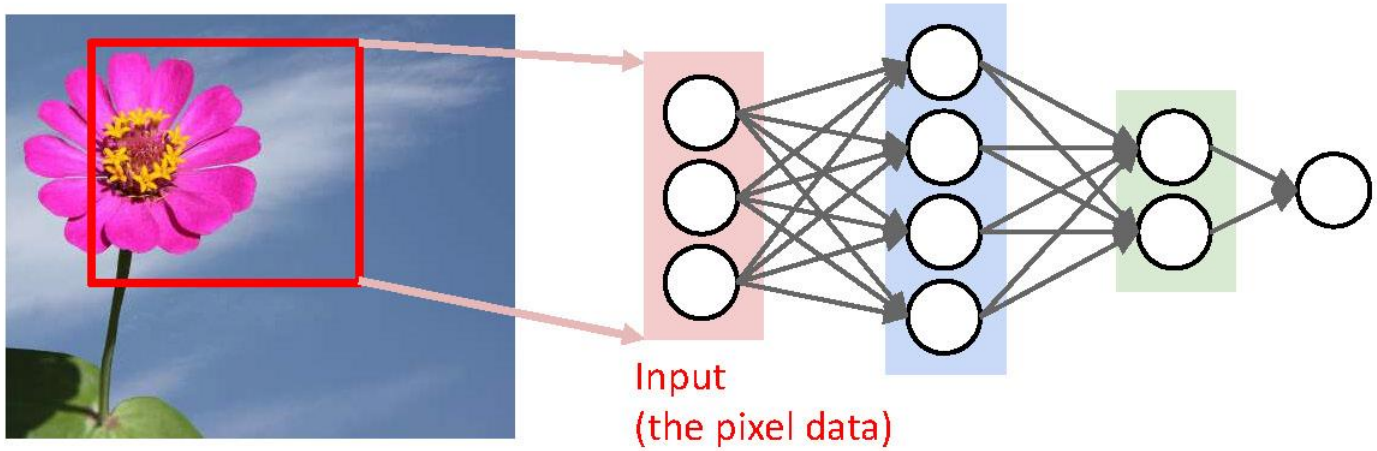
- Scan for the desired object

Solution: Scan



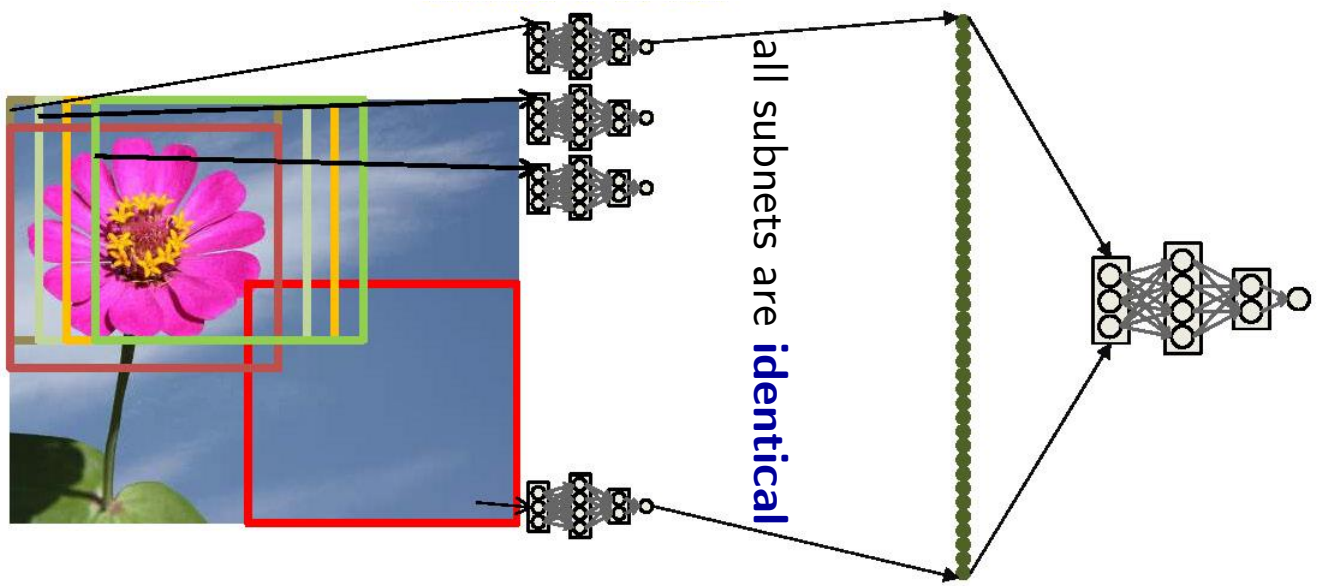
- Scan for the desired object

Scanning



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

Scanning the picture to find a flower

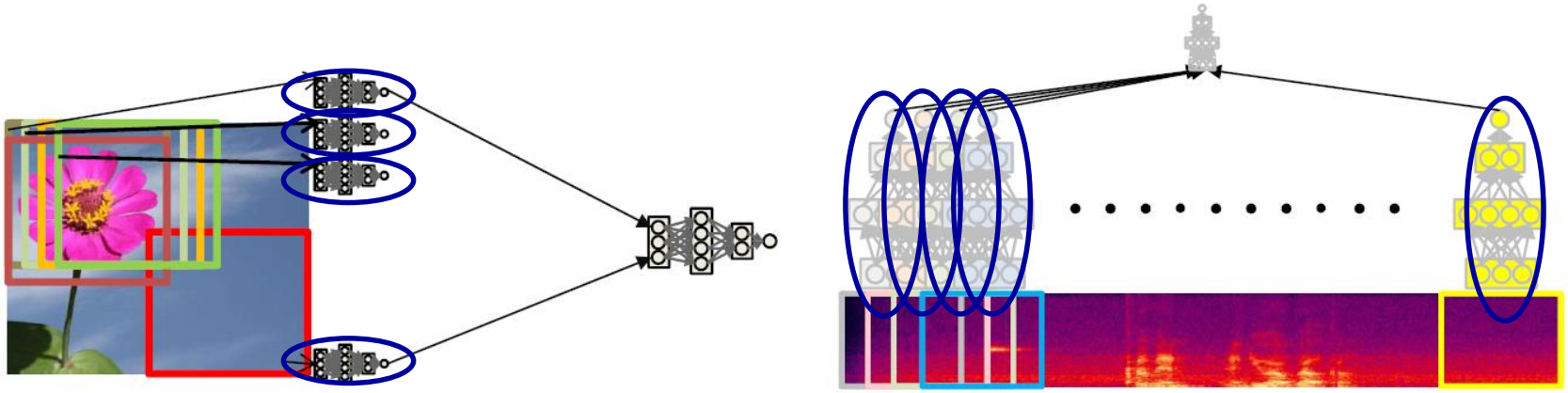


- Determine if any of the locations had a flower
  - We get one classification output per scanned location
  - Each dot in the right represents the output of the MLP when it classifies one location i input figure
    - The score output by the MLP
    - Look at the maximum value
    - **Or pass it through a softmax or even an MLP 52**

Training the network

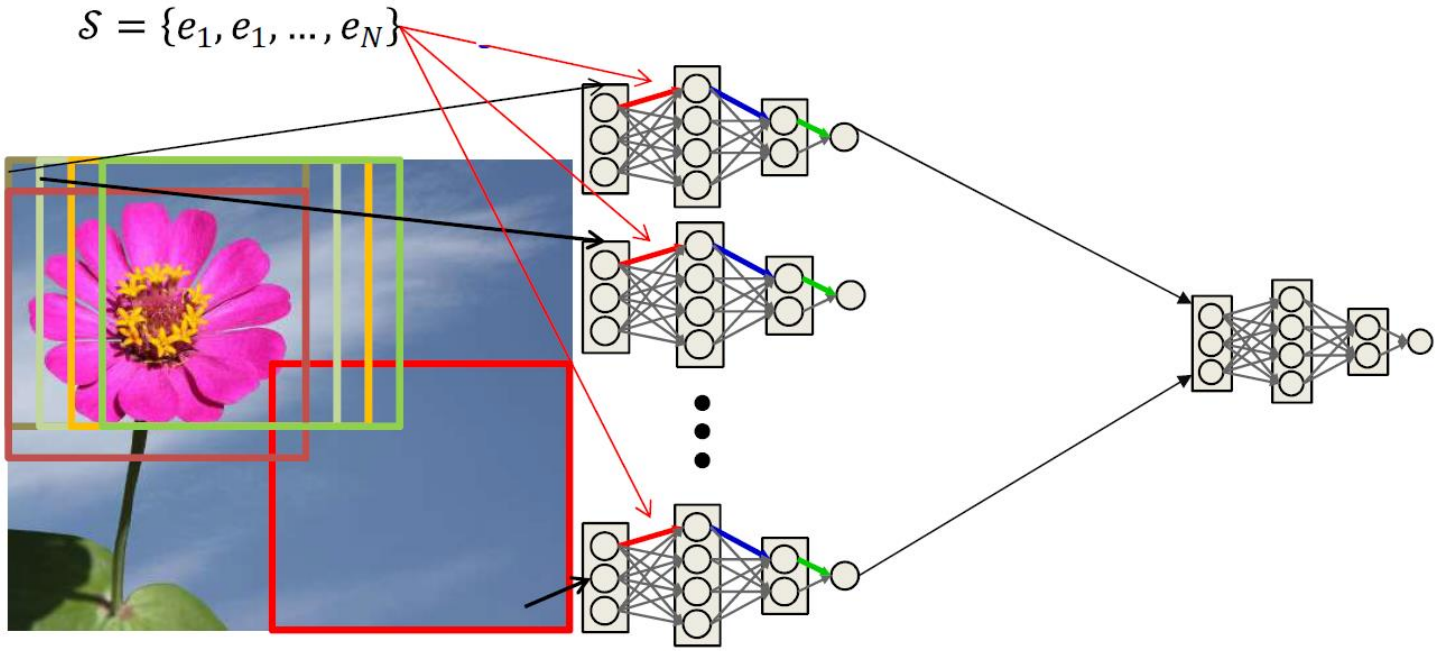
- These are really just large networks
  - Can just use conventional backpropagation to learn the parameters
    - Provide many training examples
      - Images with and without flowers
      - Speech recordings with and without the word "welcome"
- Gradient descent to minimize the total divergence between predicted and desired outputs

Shared parameter



- 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

Computing the divergence of shared parameters

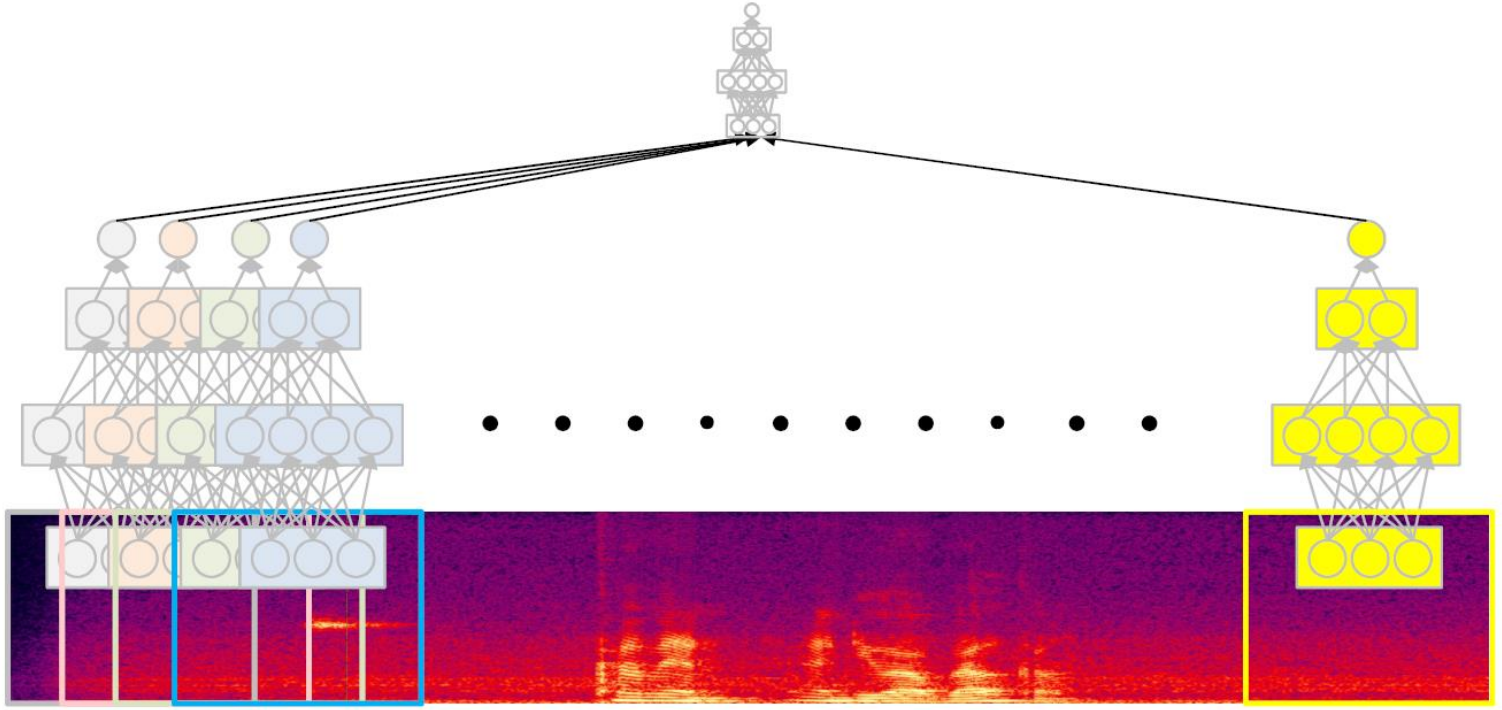


- More generally, let be any set of edges that have a common value, and $\mathcal{S}$ be the common weight of the set
  - E.g. the set of all red weights in the figure

$$\frac{dDiv}{dw^{\mathcal{S}}} = \sum_{e \in \mathcal{S}} \frac{\partial Div}{\partial w^e}$$

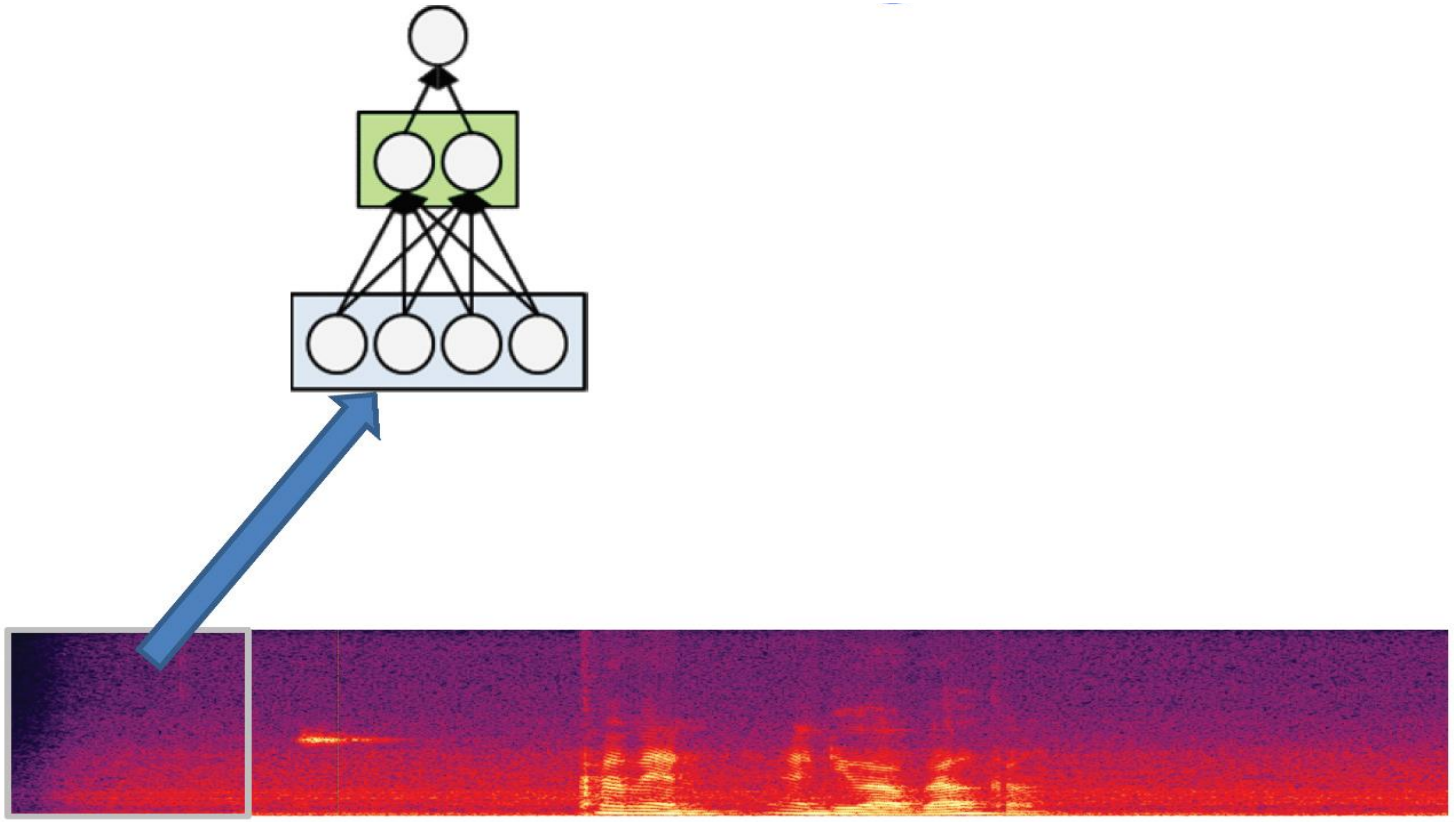
  - The individual terms in the sum can be computed via backpropagation

Scanning: A closer look



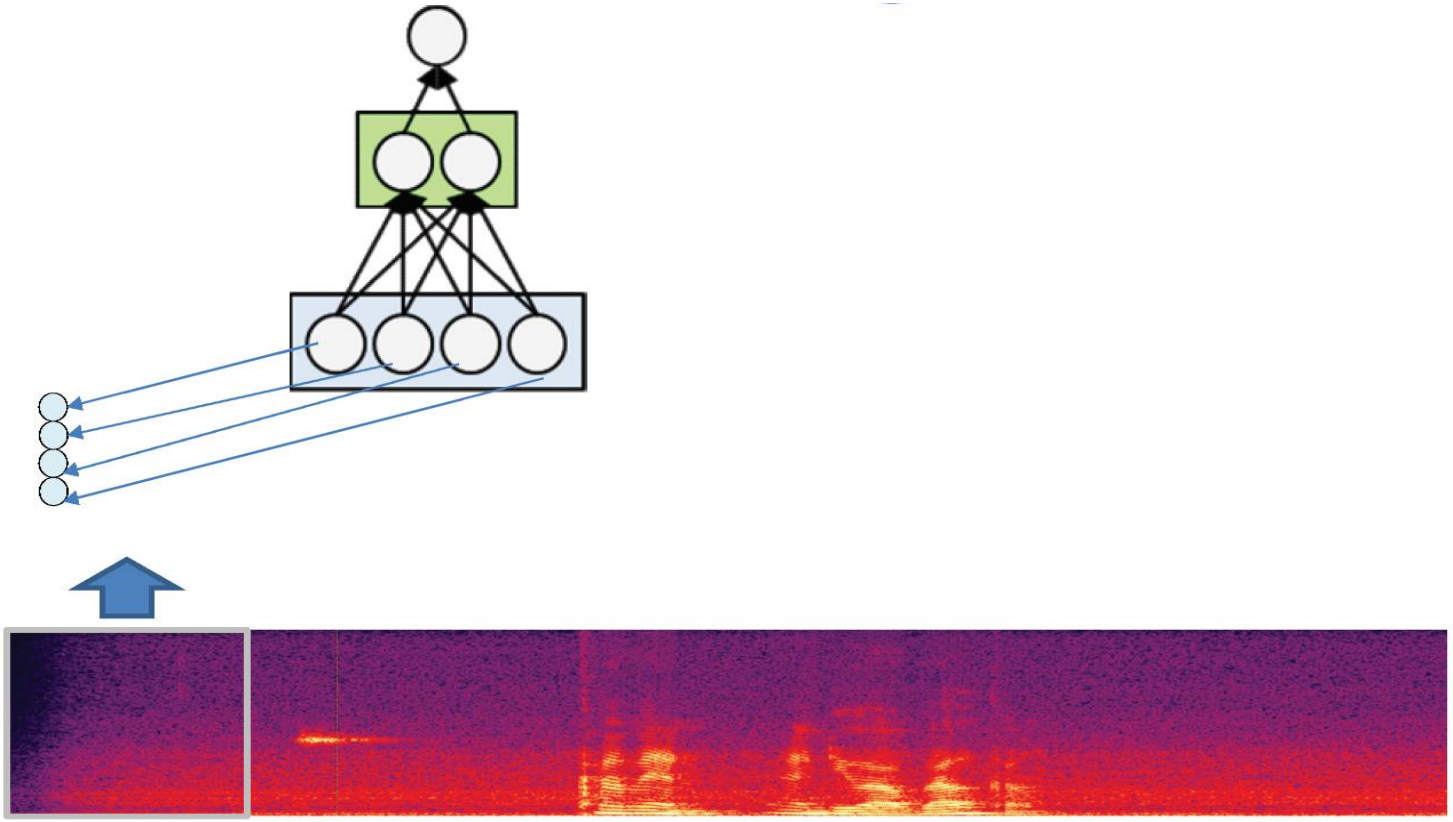
- The entire MLP operates on each “window” of the input

Scanning



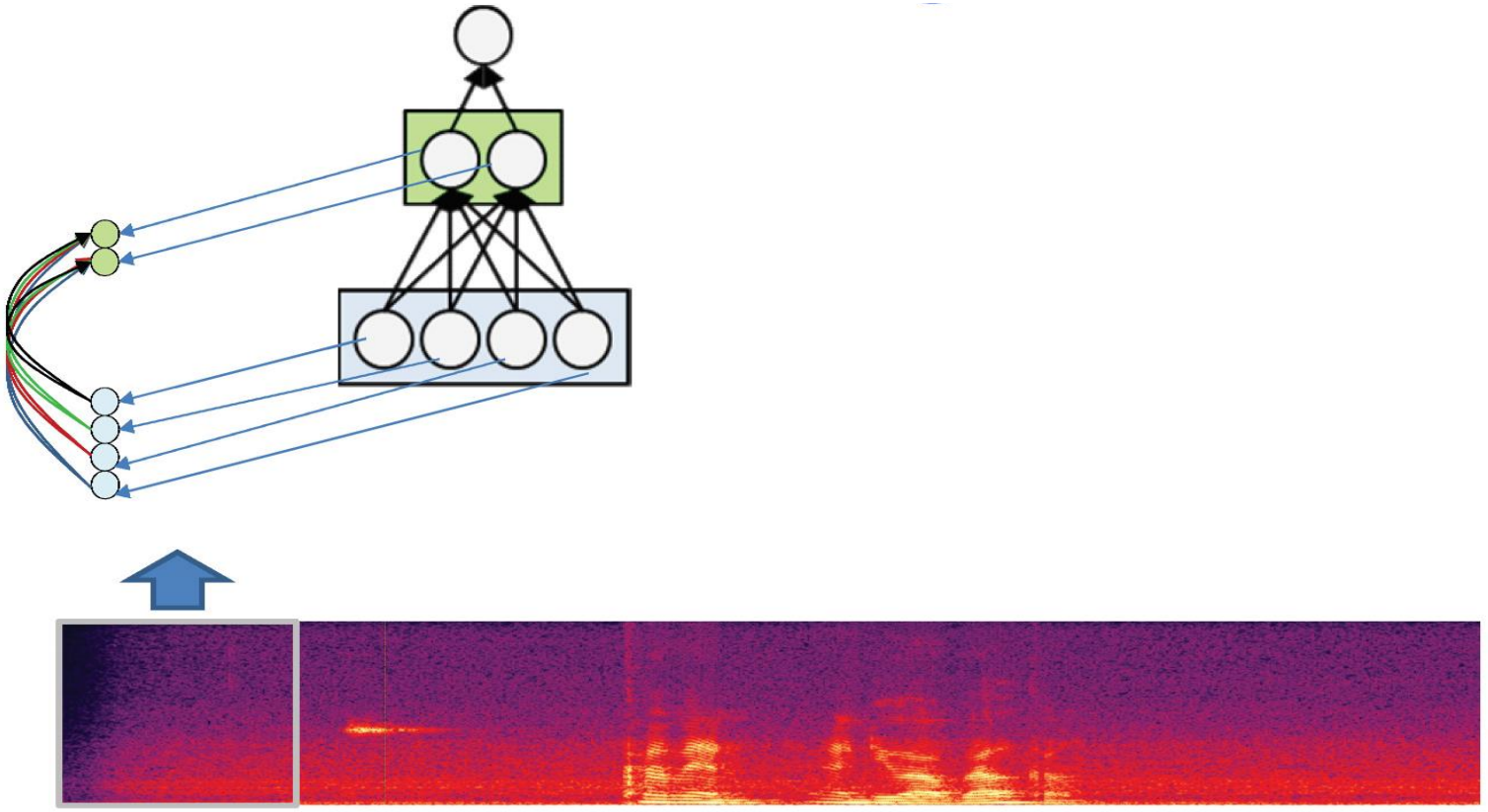
- At each location, each neuron computes a value based on its inputs
  - Which may either be the input image or the outputs of the previous layer

Scanning



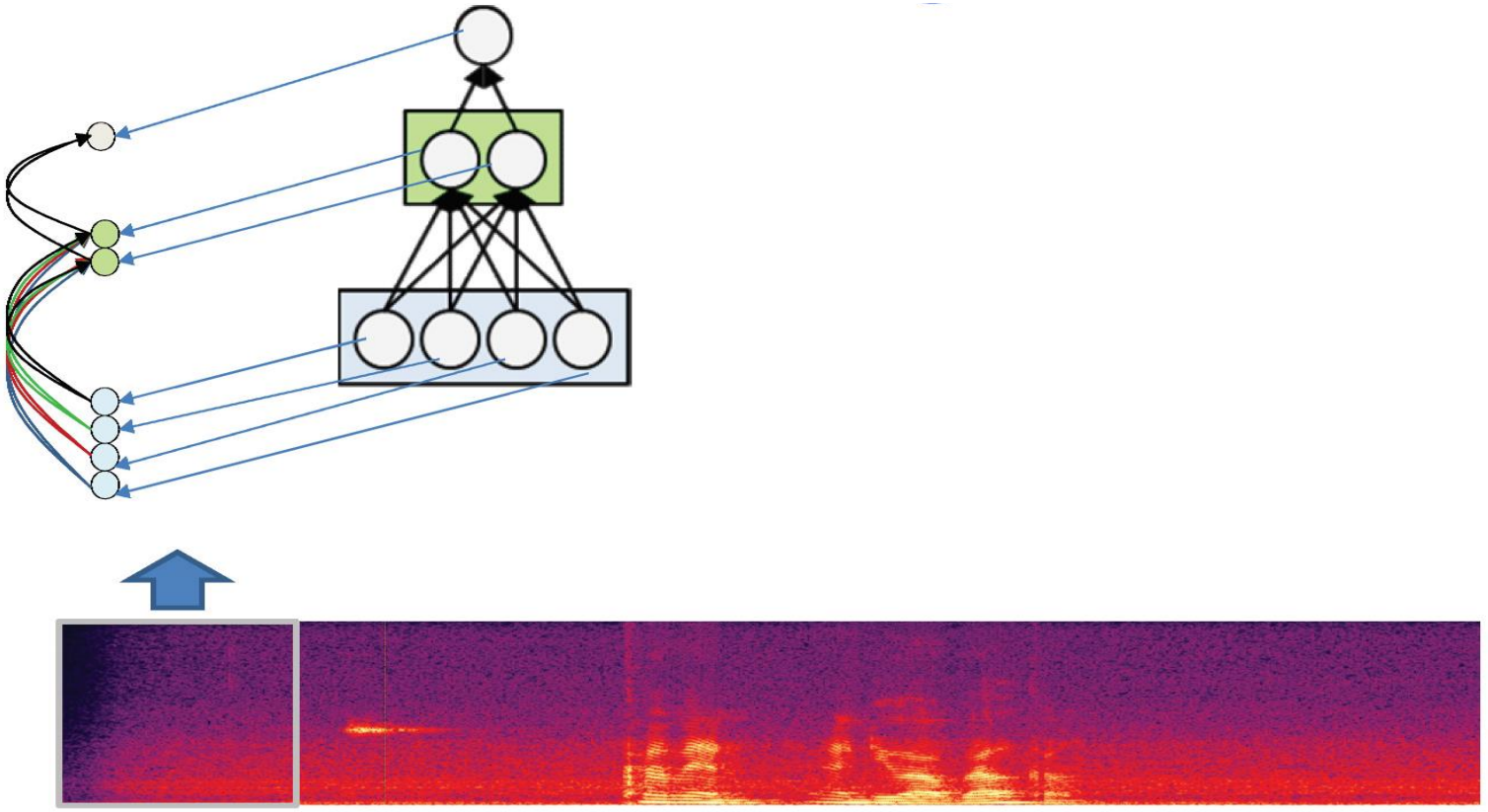
- At each location, each neuron computes a value based on its inputs
  - Which may either be the input image

Scanning



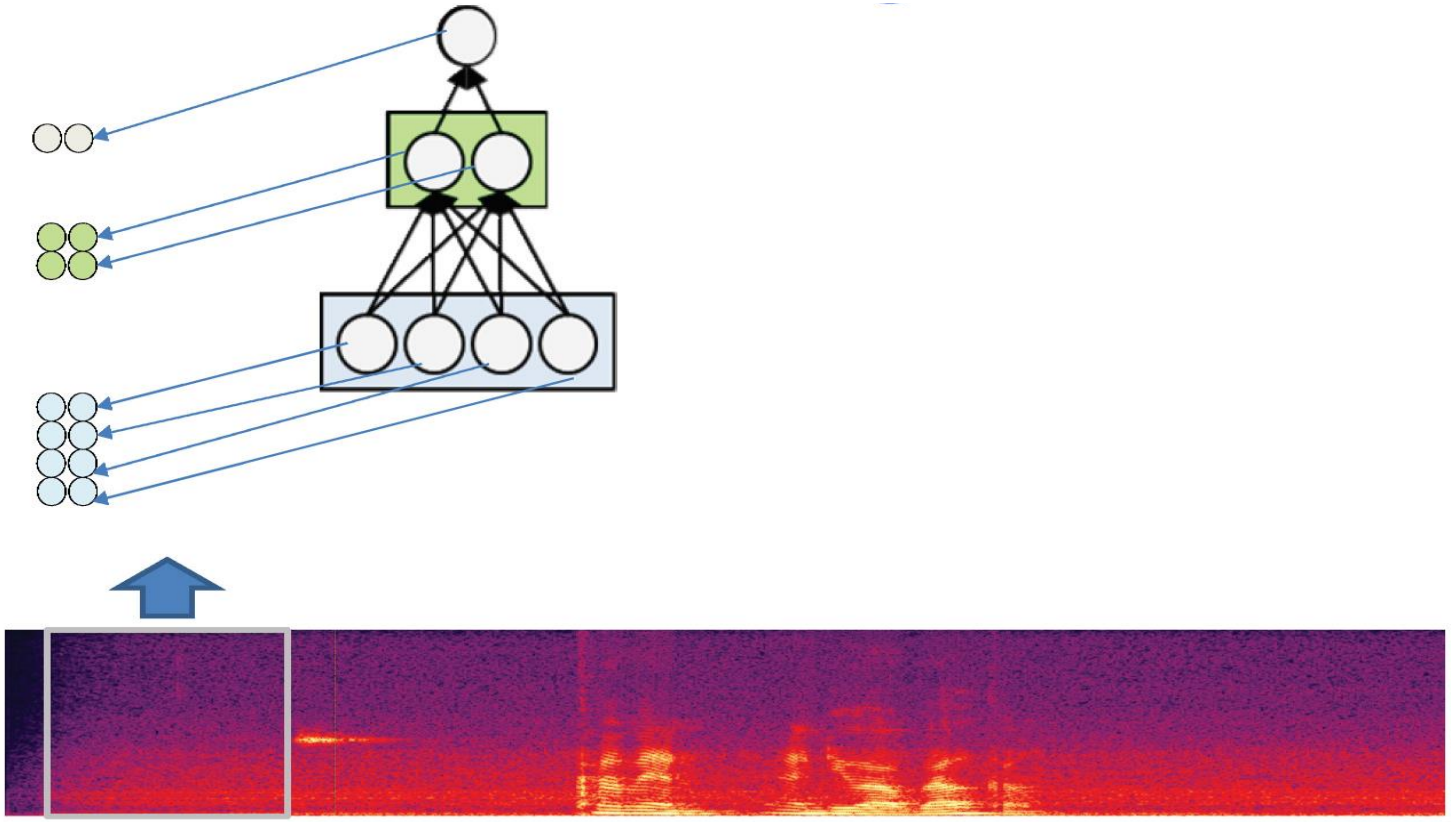
- At each location, each neuron computes a value based on its inputs
  - Which may either be the input image or the outputs of the previous layer

Scanning



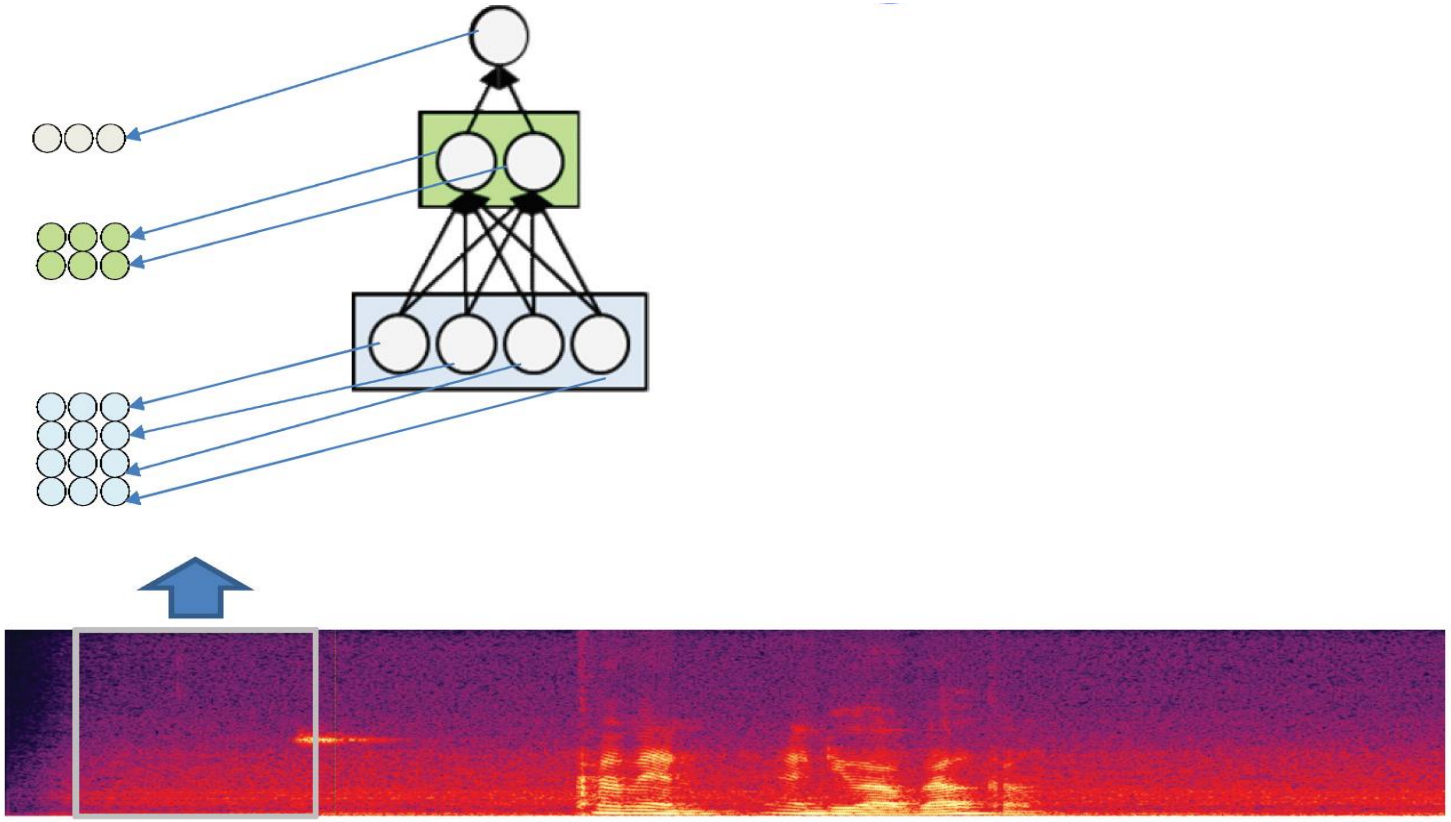
- At each location, each neuron computes a value based on its inputs
  - Which may either be the input image or the outputs of the previous layer

Scanning



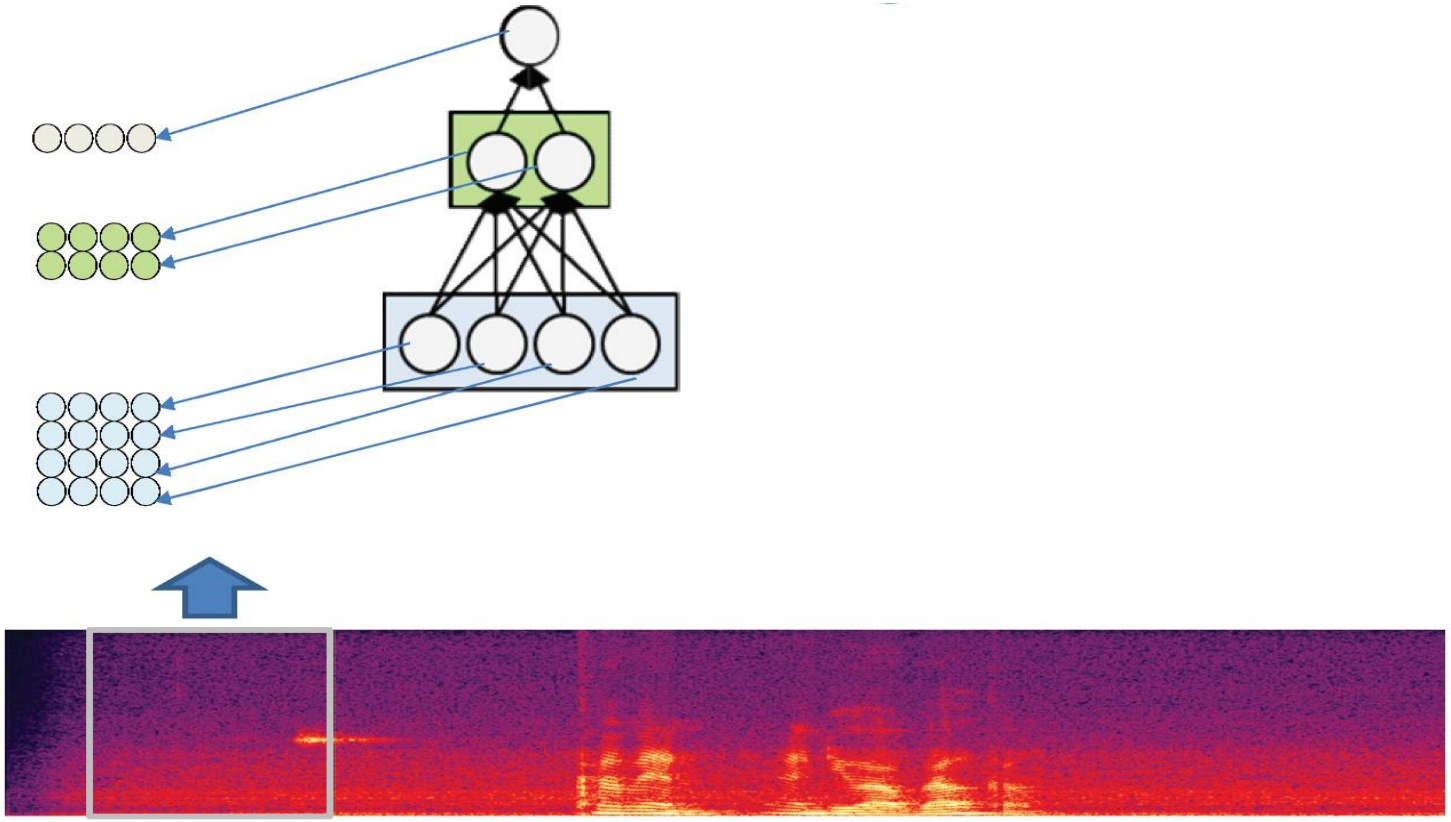
- The same sequence of computations is performed at each location
  - Producing the same set of values
    - One value per neuron in each layer

Scanning



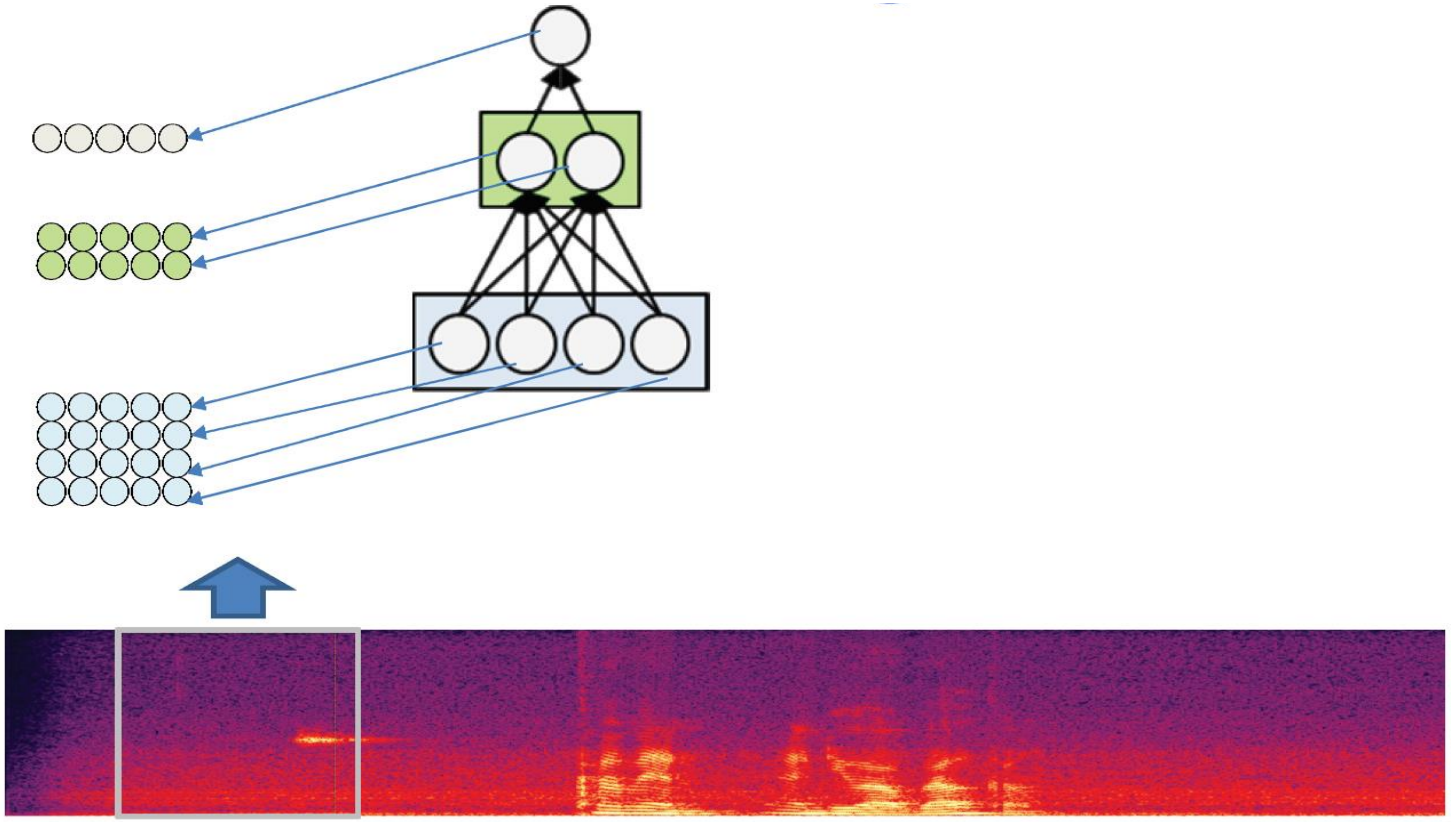
- The same sequence of computations is performed at each location
  - Producing the same set of values
    - One value per neuron in each layer

Scanning



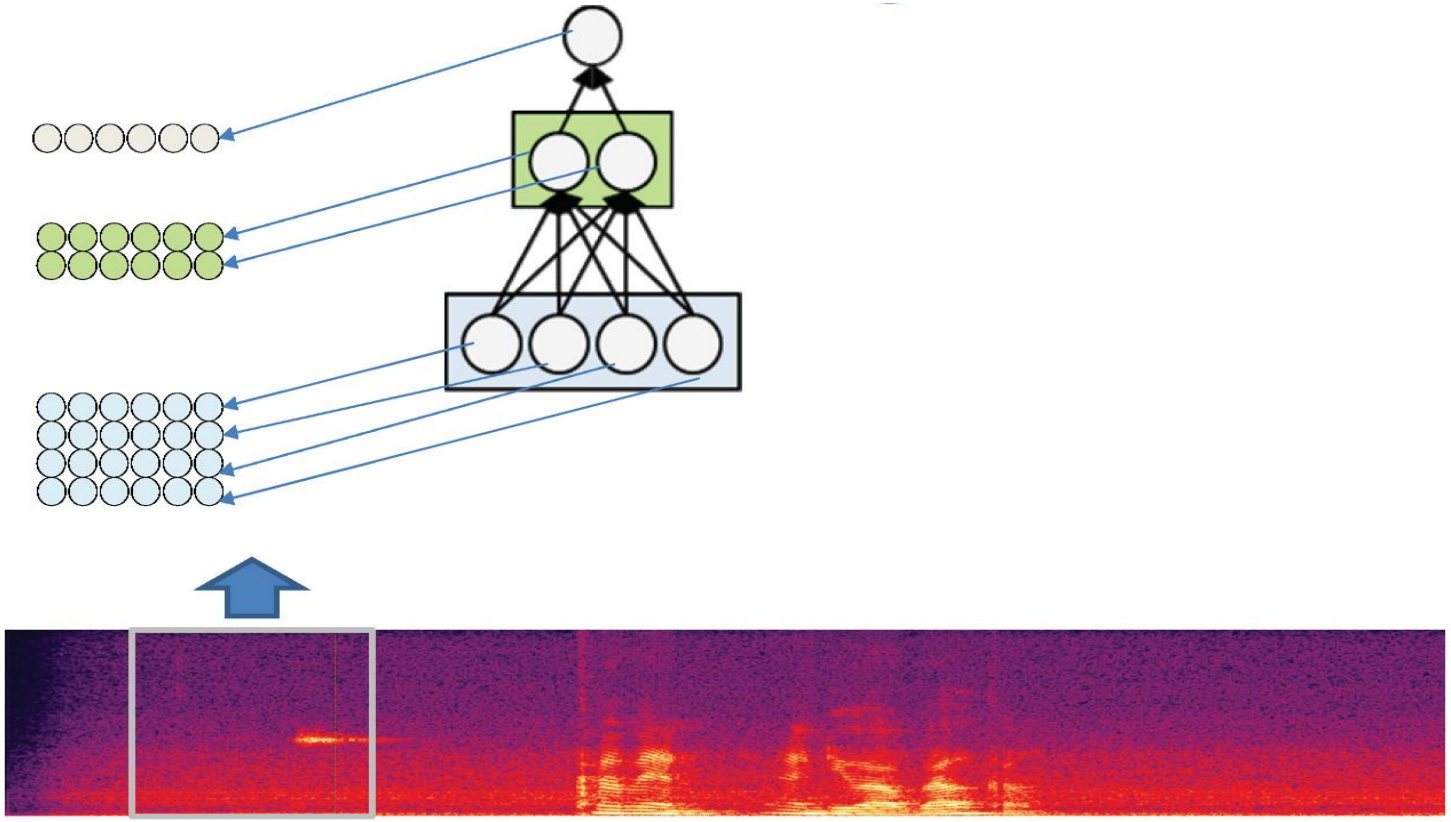
- The same sequence of computations is performed at each location
  - Producing the same set of values
    - One value per neuron in each layer

Scanning



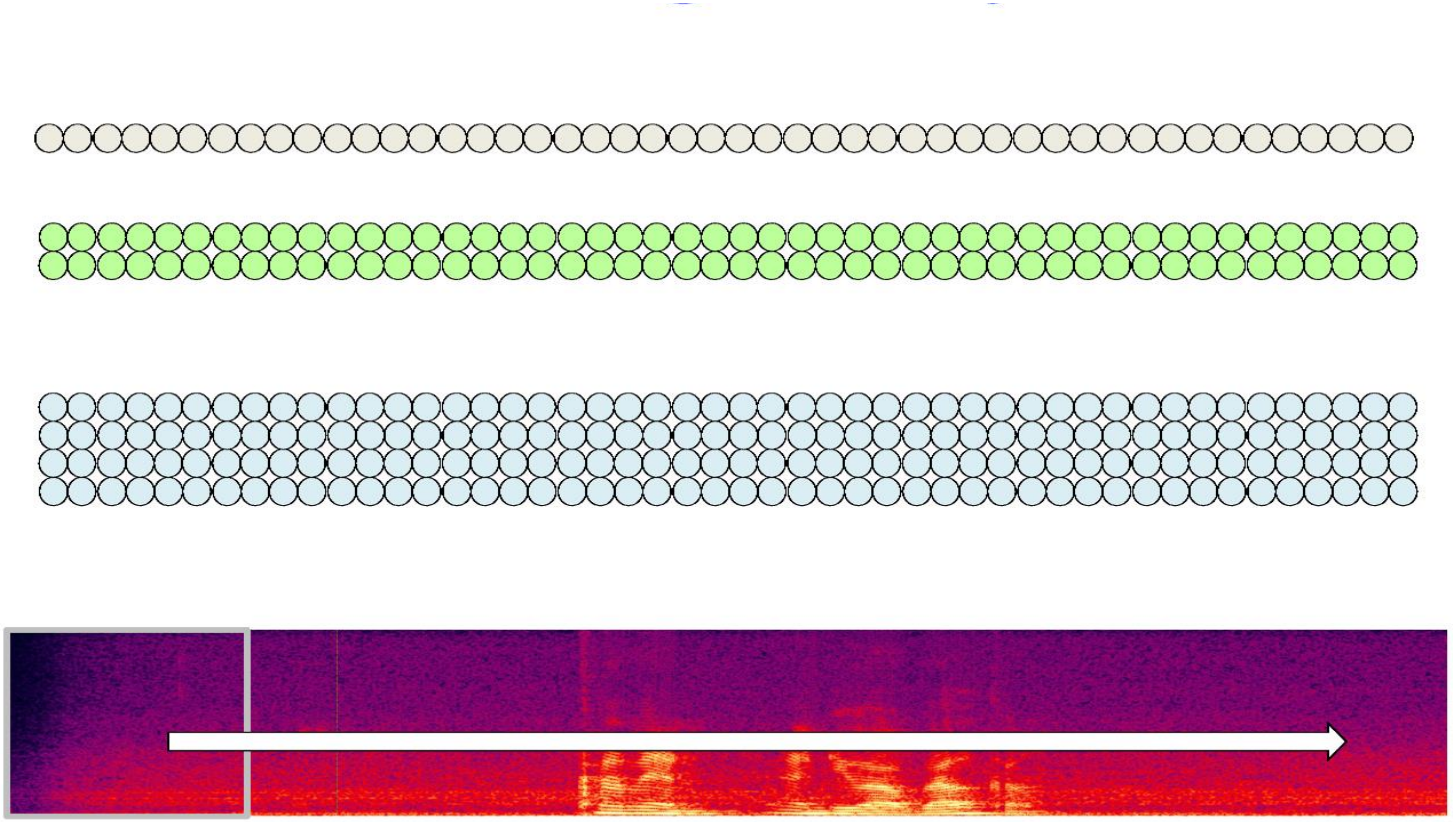
- The same sequence of computations is performed at each location
  - Producing the same set of values
    - One value per neuron in each layer

Scanning



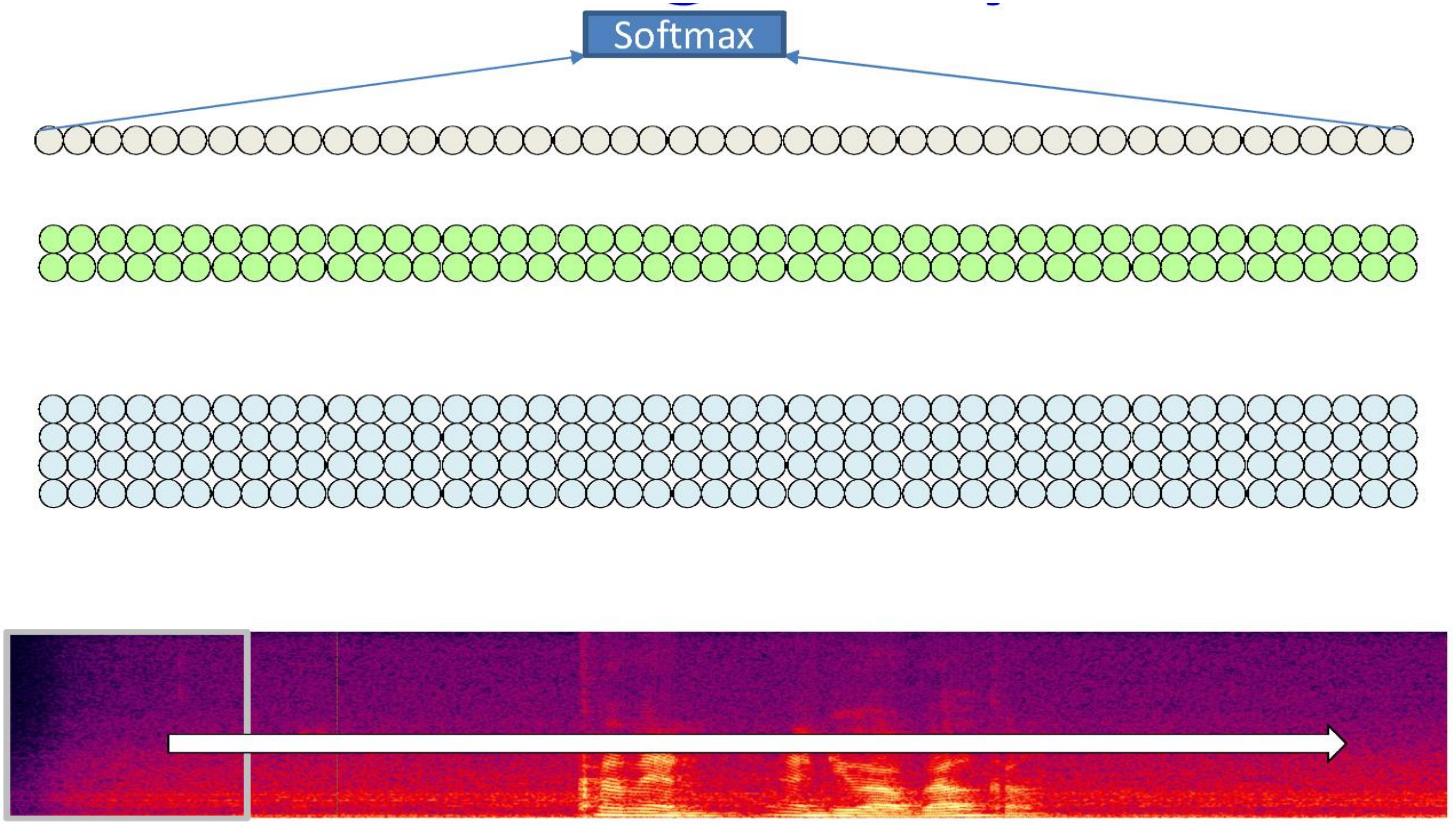
- The same sequence of computations is performed at each location
  - Producing the same set of values
    - One value per neuron in each layer

Scanning the input



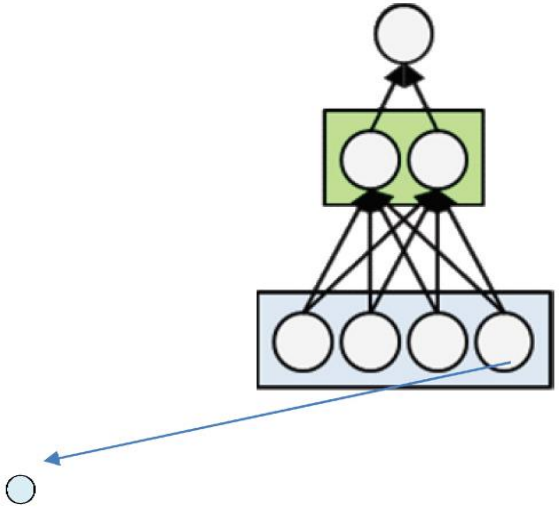
- We get a complete set of values (represented as a column) at each location evaluated by the MLP during the scan

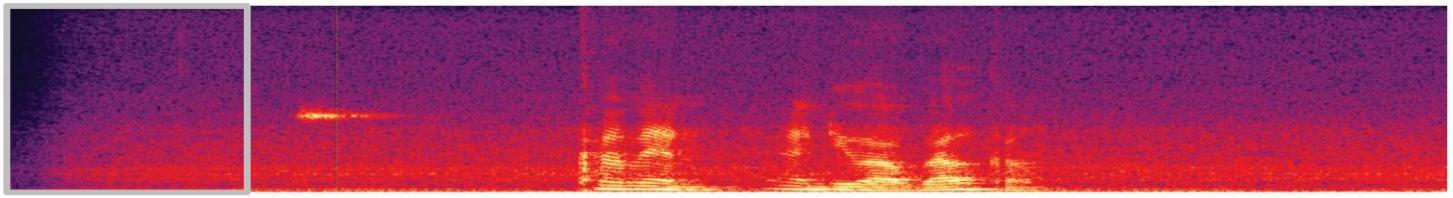
Scanning the input



- We get a complete set of values (represented as a column) at each location evaluated by the MLP during the scan
  - Which we put through our final softmax to decide if the recording includes the word “Welcome”

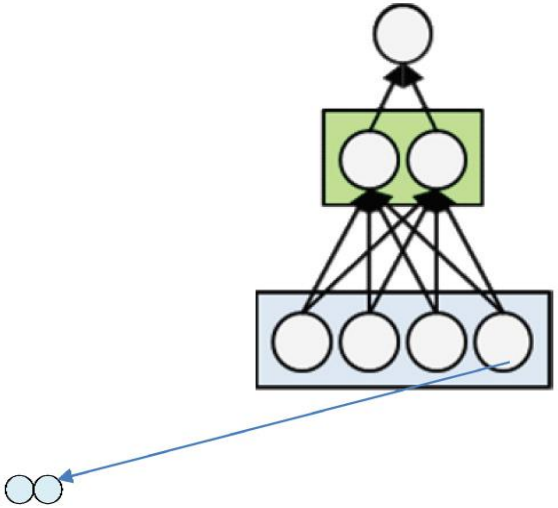
Lets do it in an different order

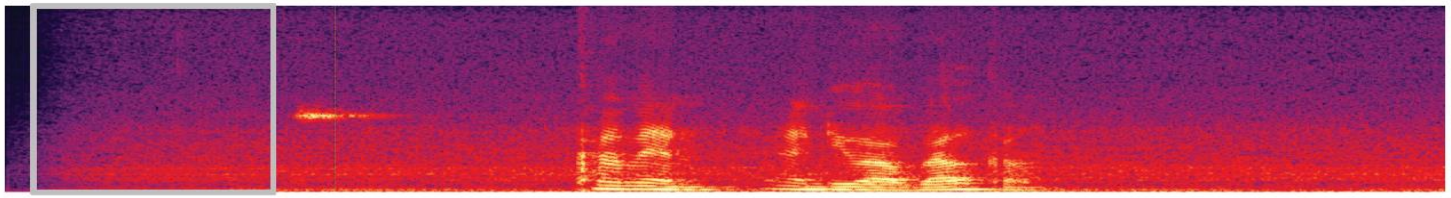




- Let us do the computation in a different order
- The first neuron evaluates each image first
  - "Scans" the input

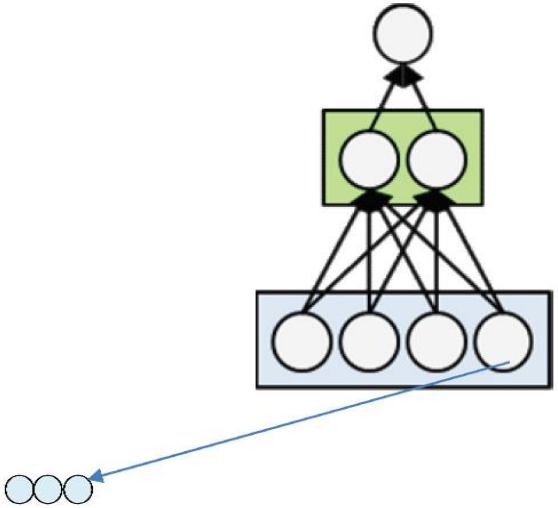
Lets do it in an different order

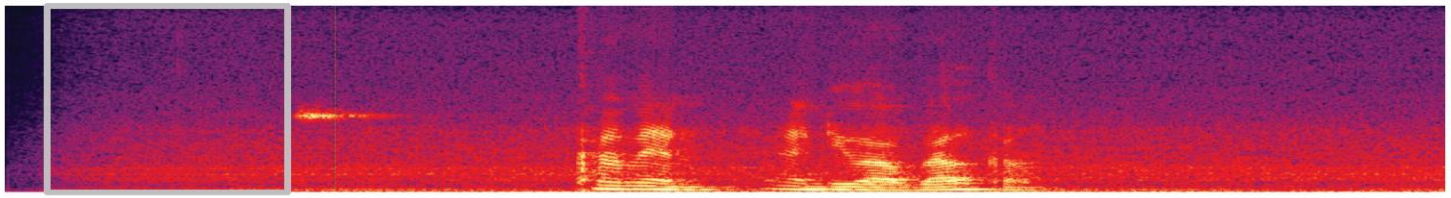




- Let us do the computation in a different order
- The first neuron evaluates each image first
  - "Scans" the input

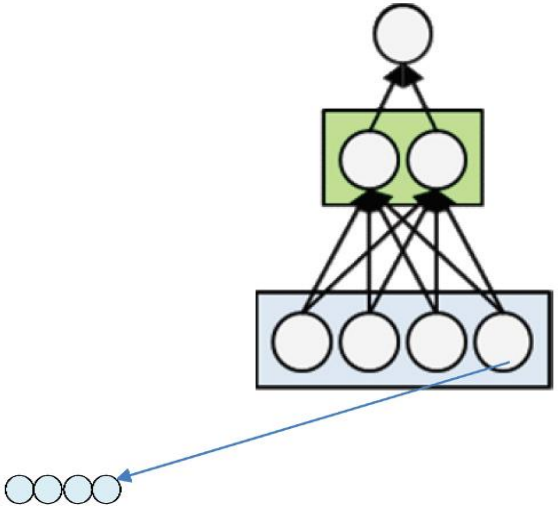
Lets do it in an different order

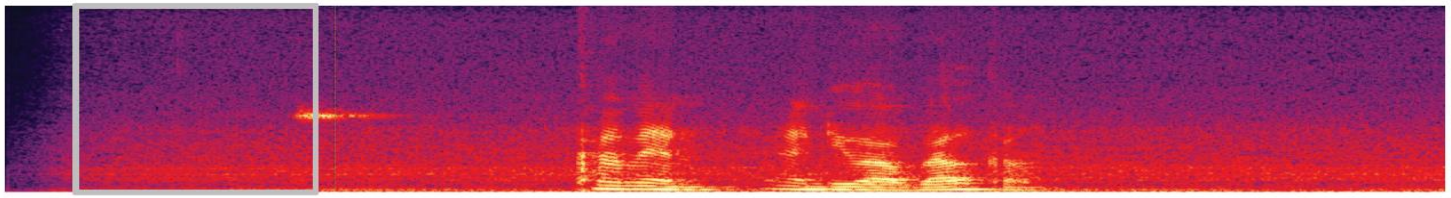




- Let us do the computation in a different order
- The first neuron evaluates each image first
  - "Scans" the input

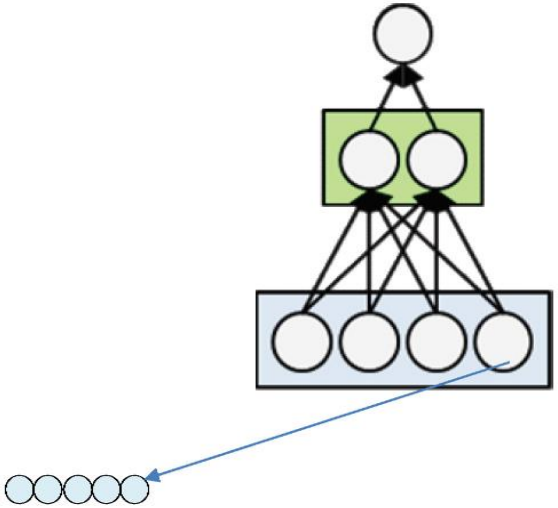
Lets do it in an different order

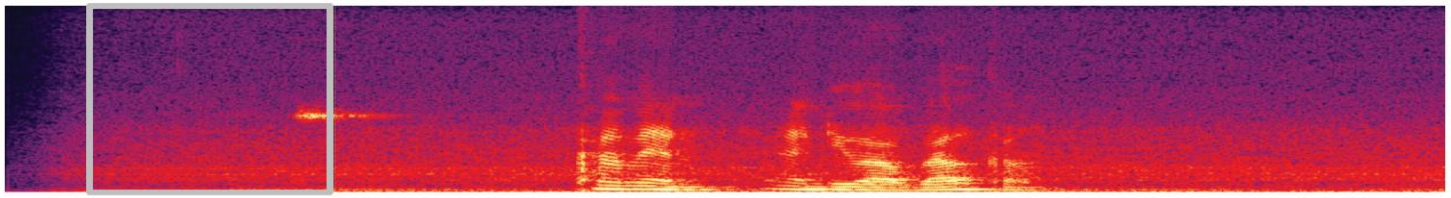




- Let us do the computation in a different order
- The first neuron evaluates each image first
  - "Scans" the input

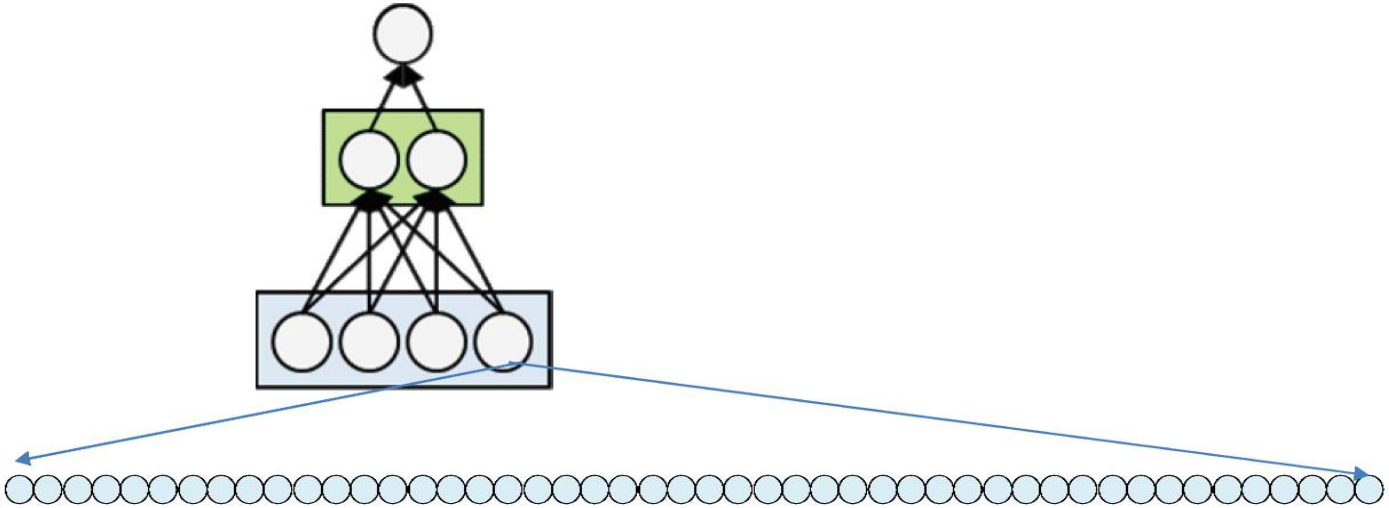
Lets do it in an different order

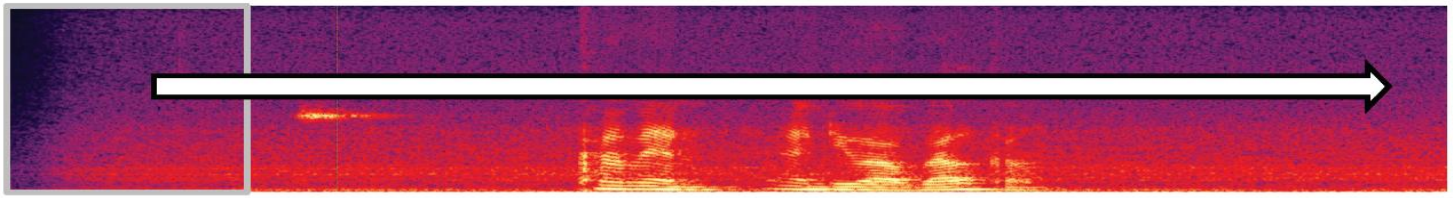




- Let us do the computation in a different order
- The first neuron evaluates each image first
  - "Scans" the input

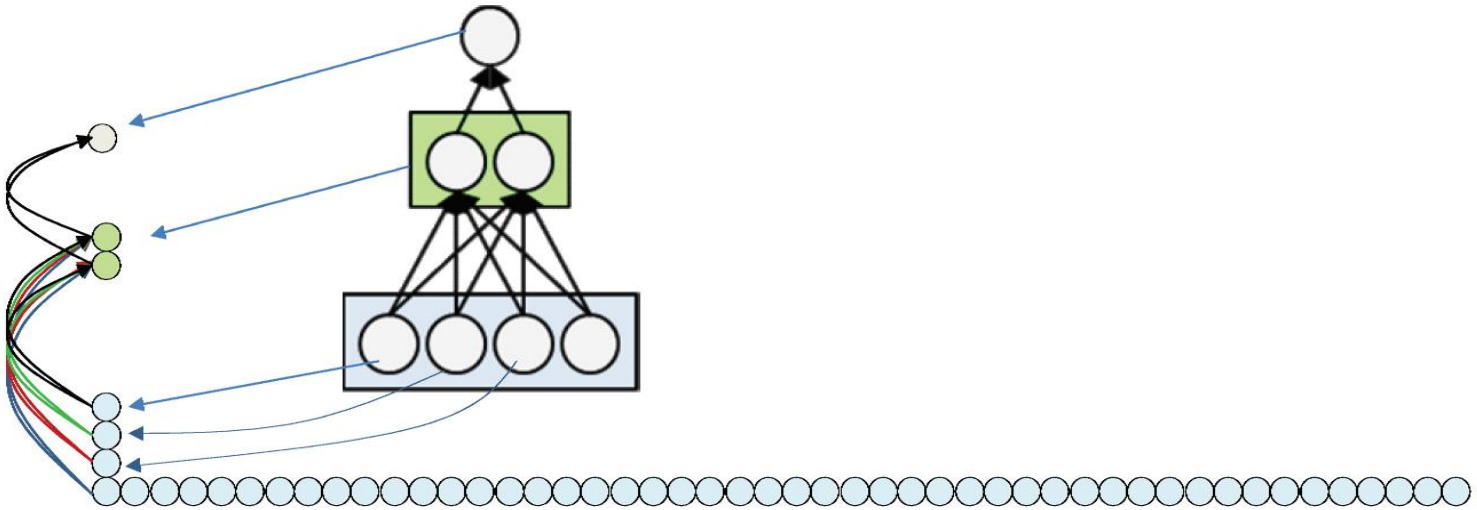
Lets do it in an different order

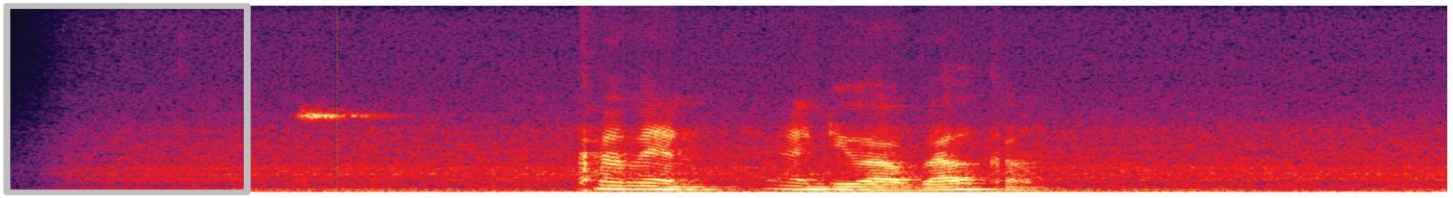




- Let us do the computation in a different order
- The first neuron evaluates each image first
  - "Scans" the input

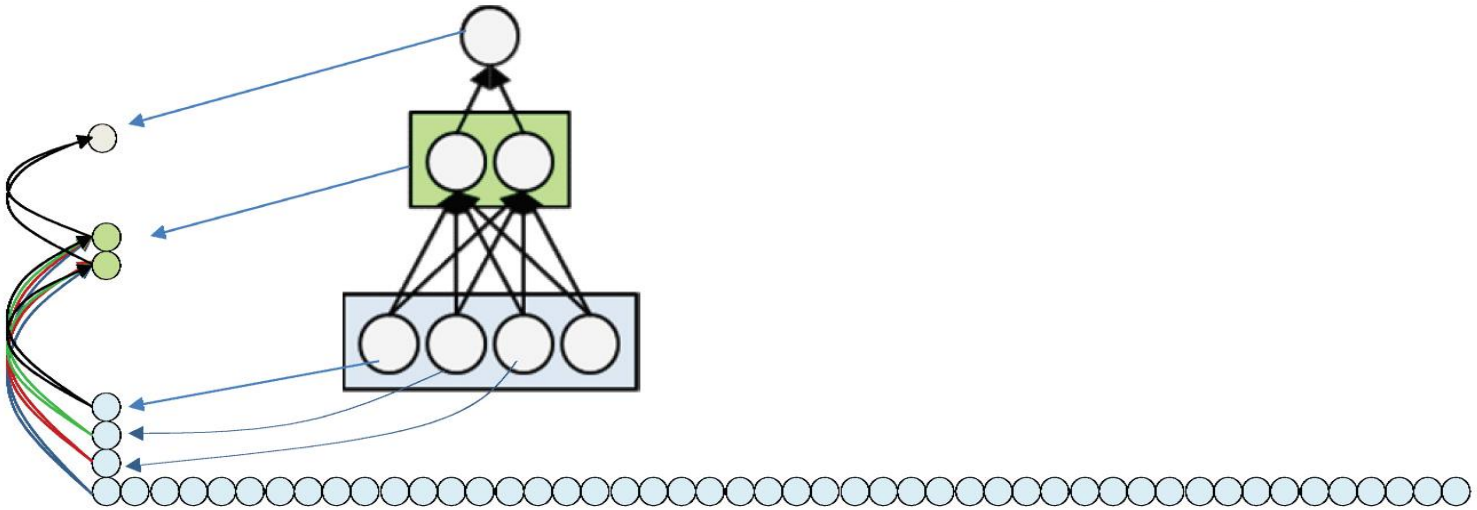
Lets do it in an different order

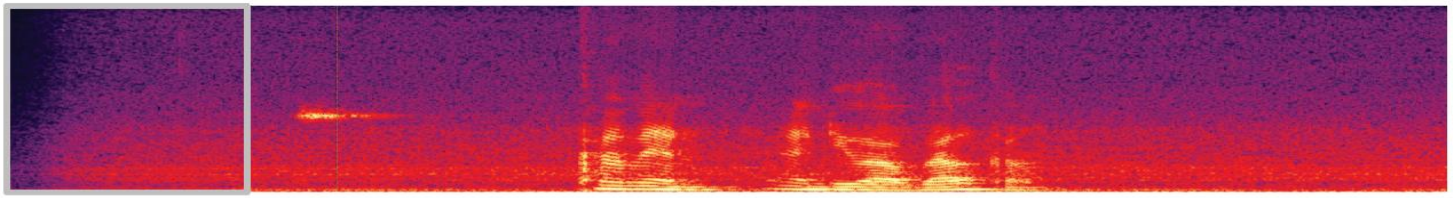




- Subsequently the rest of the neurons in the first layer operate on the first block
  - And the downstream layers as well
- Would the output of the MLP at the first block be different?

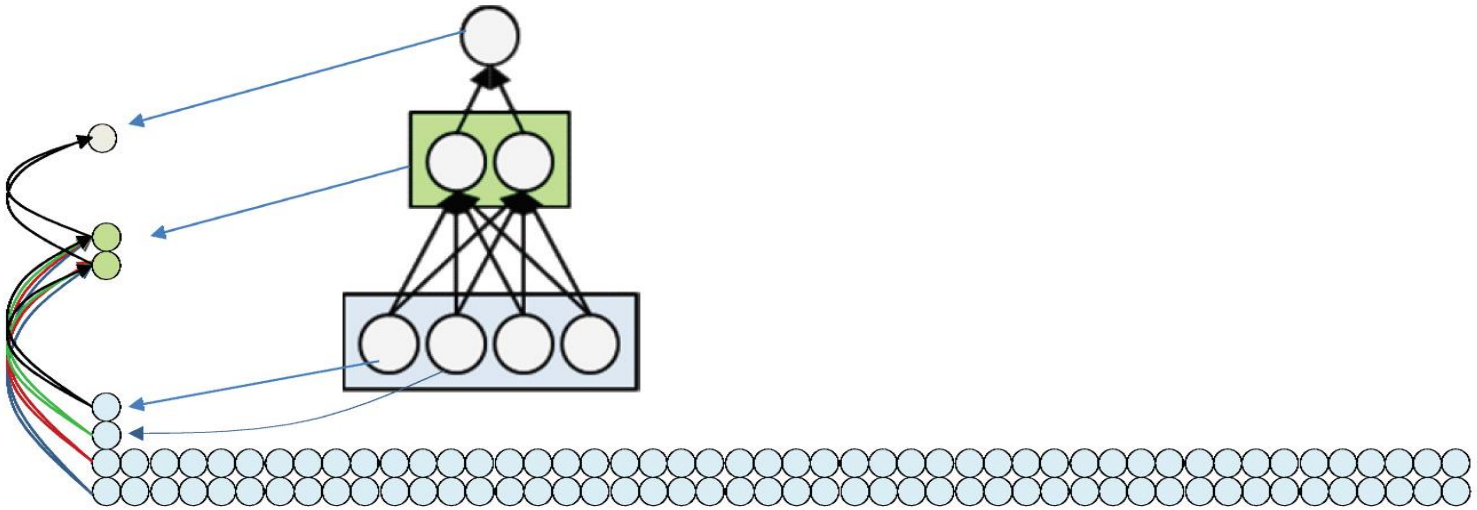
Lets do it in an different order

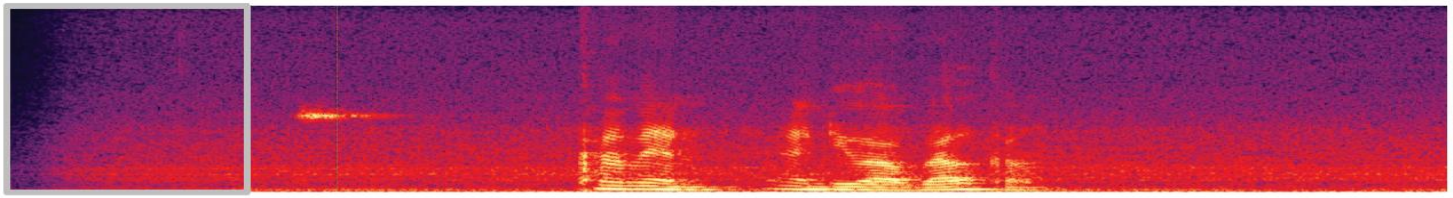




- Subsequently the rest of the neurons in the first layer operate on the first block
  - And the downstream layers as well
- Would the output of the MLP at the first block be different?
  - The fact that the first neuron has already evaluated the future blocks does not affect the output of that neuron, or the network itself, at the current block

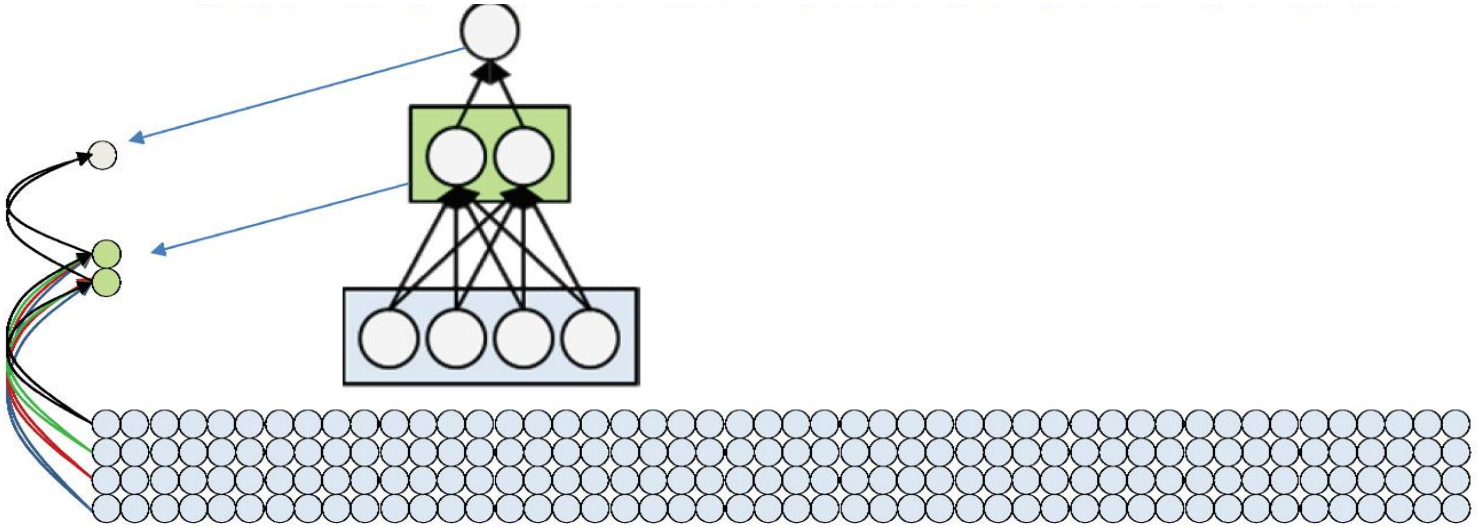
Lets do it in an different order

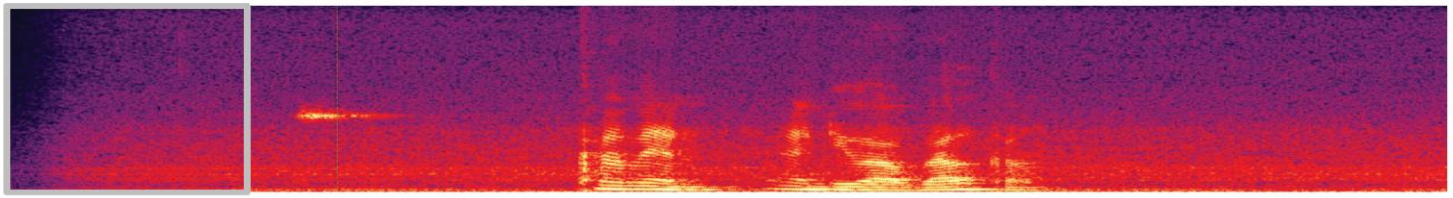




- What about now?
- The second neuron too has fully evaluated the entire input before the rest of the network evaluates the first block
  - This too should not change the output of the network for the first block

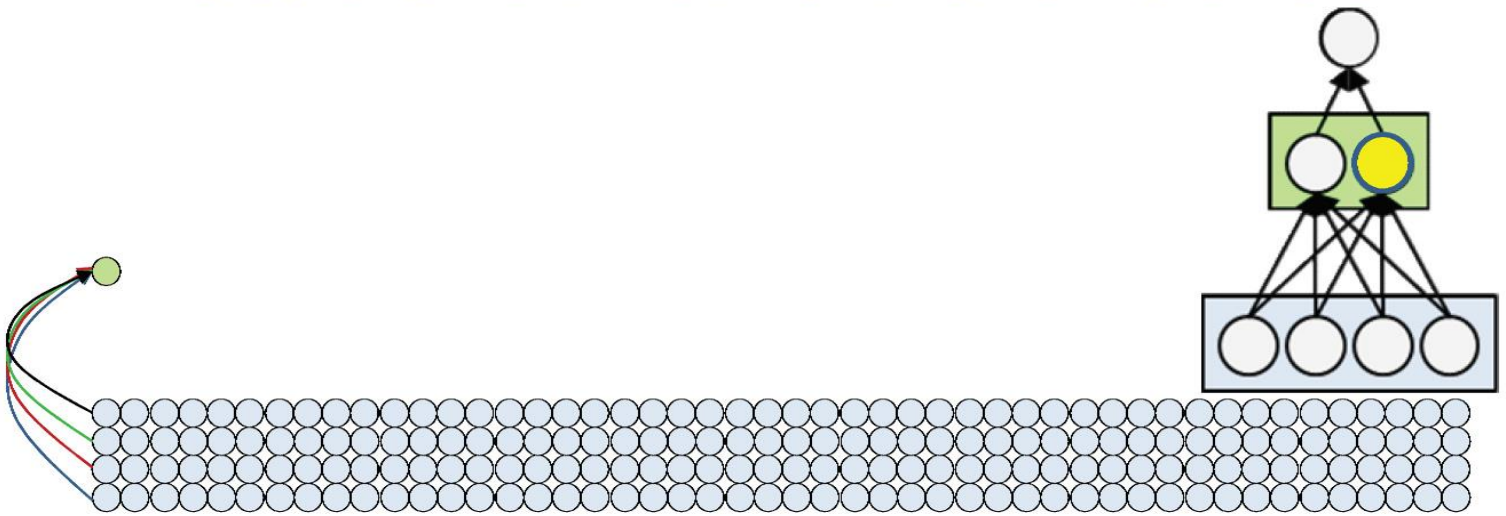
Lets do it in an different order

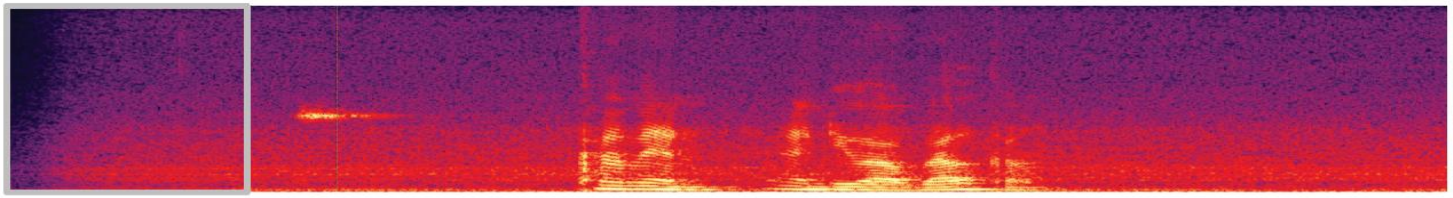




- In fact if all of the neurons in the first layer fully evaluate the entire input before the rest of the network evaluates the first block, this will not change the output of the network at the first block

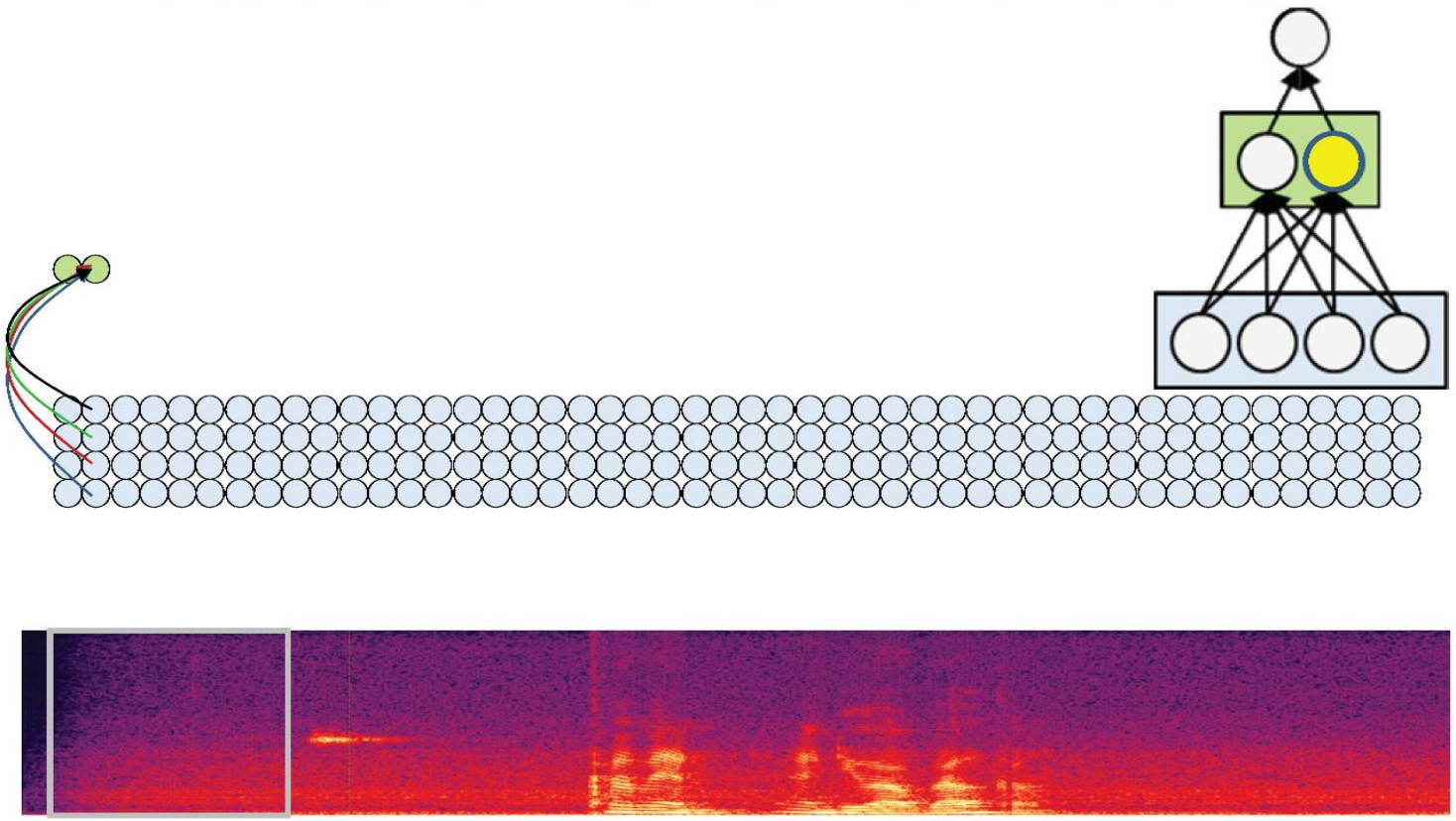
Lets do it in an different order





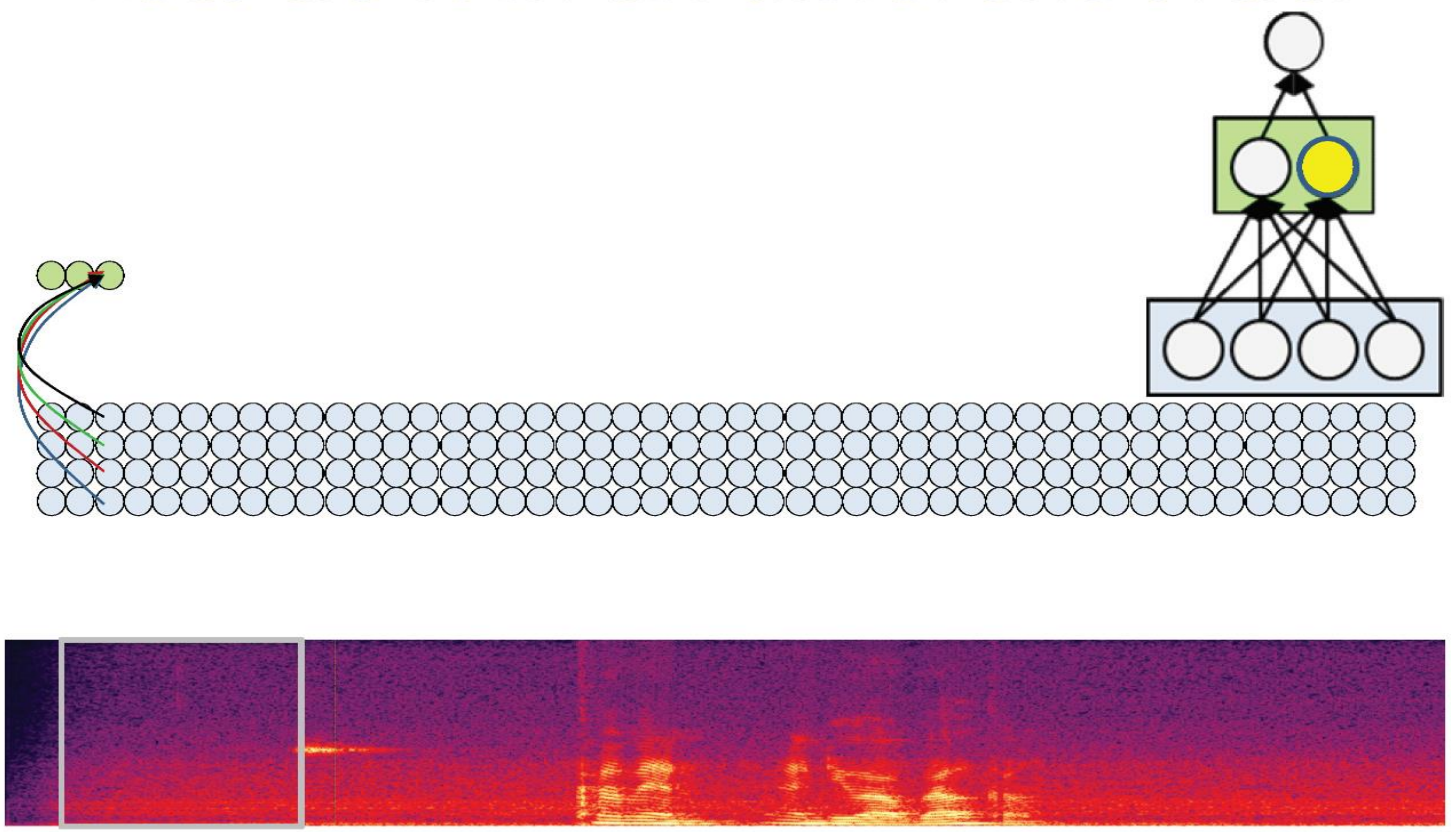
- But now, since the first layer neurons have already produced outputs for every location, each neuron in the second layer can go ahead and produce outputs for every position without waiting for the rest of the net
  - "Scan" the outputs of the first layer neurons

Lets do it in an different order



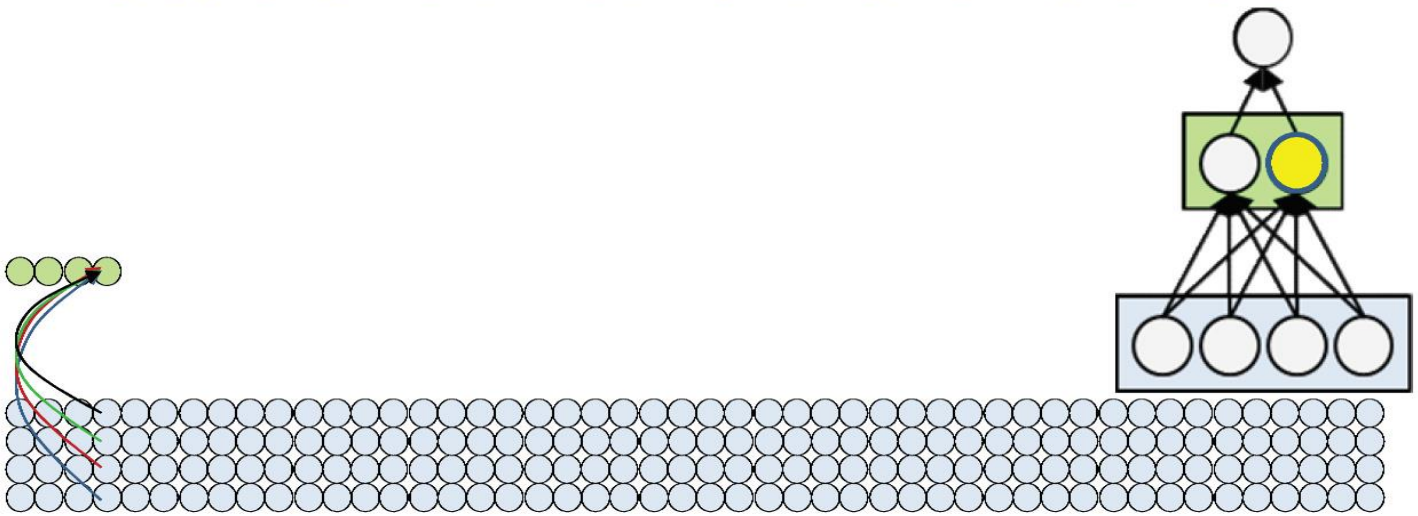
- But now, since the first layer neurons have already produced outputs for every location, each neuron in the second layer can go ahead and produce outputs for every position without waiting for the rest of the net
  - "Scan" the outputs of the first layer neurons

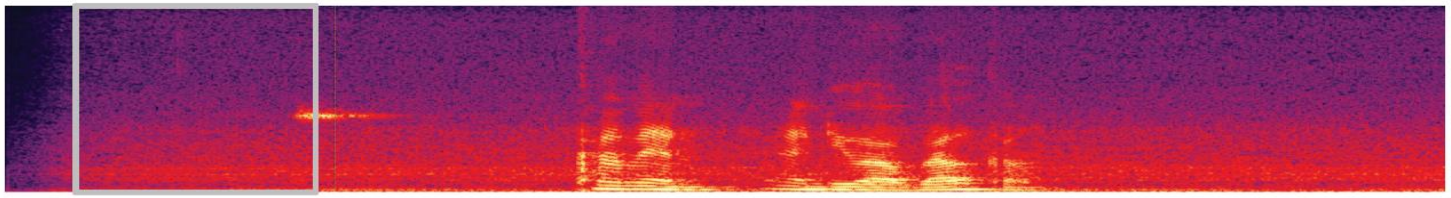
Lets do it in an different order



- But now, since the first layer neurons have already produced outputs for every location, each neuron in the second layer can go ahead and produce outputs for every position without waiting for the rest of the net
  - "Scan" the outputs of the first layer neurons

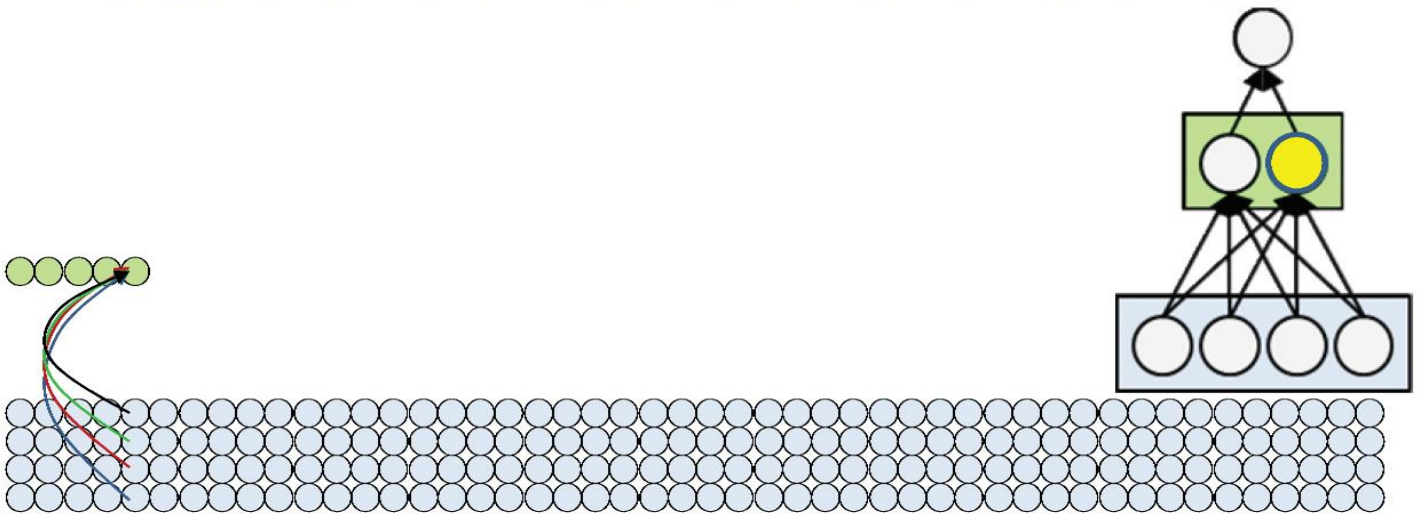
Lets do it in an different order

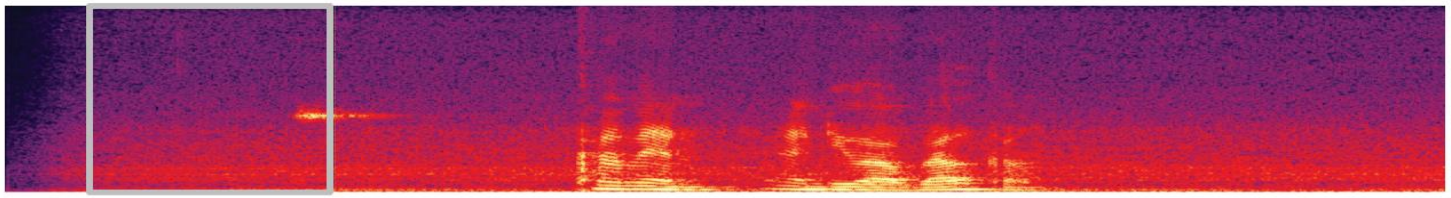




- But now, since the first layer neurons have already produced outputs for every location, each neuron in the second layer can go ahead and produce outputs for every position without waiting for the rest of the net
  - "Scan" the outputs of the first layer neurons

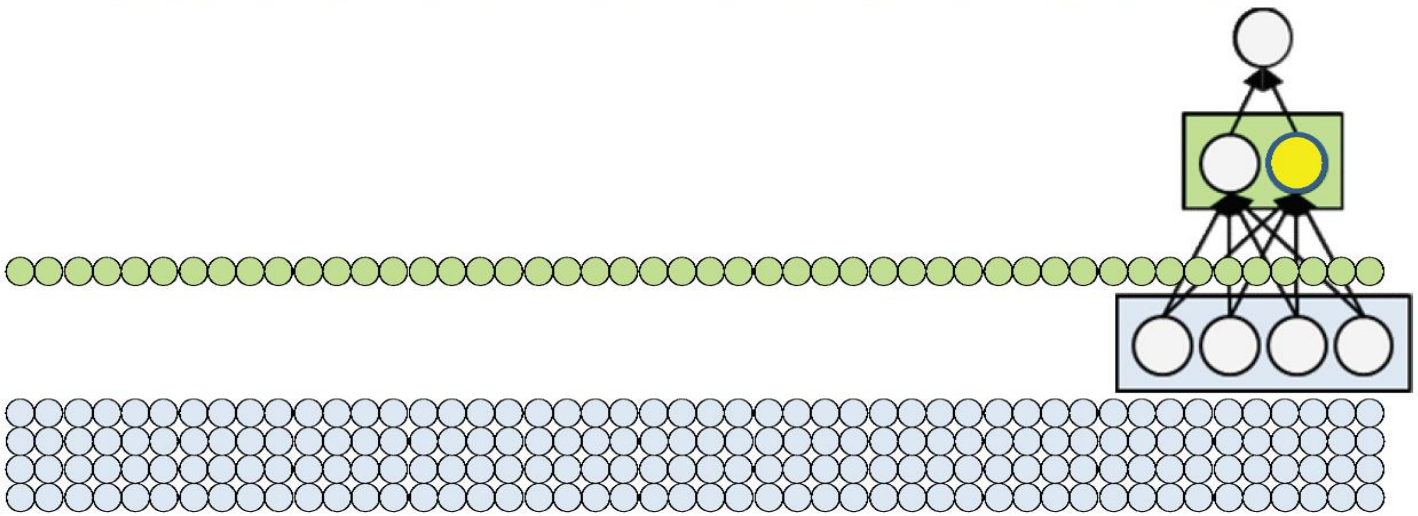
Lets do it in an different order

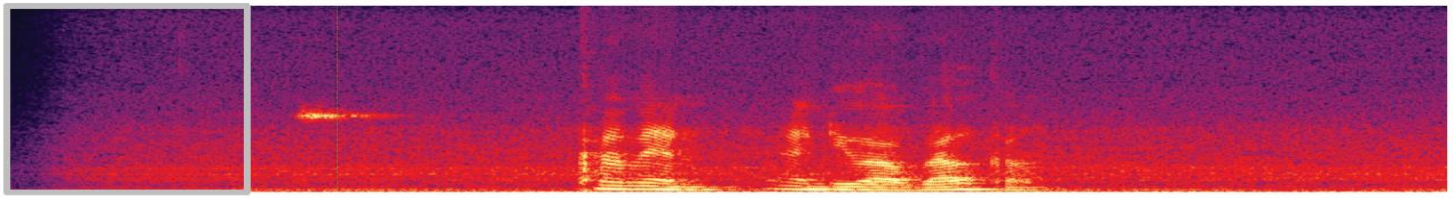




- But now, since the first layer neurons have already produced outputs for every location, each neuron in the second layer can go ahead and produce outputs for every position without waiting for the rest of the net
  - "Scan" the outputs of the first layer neurons

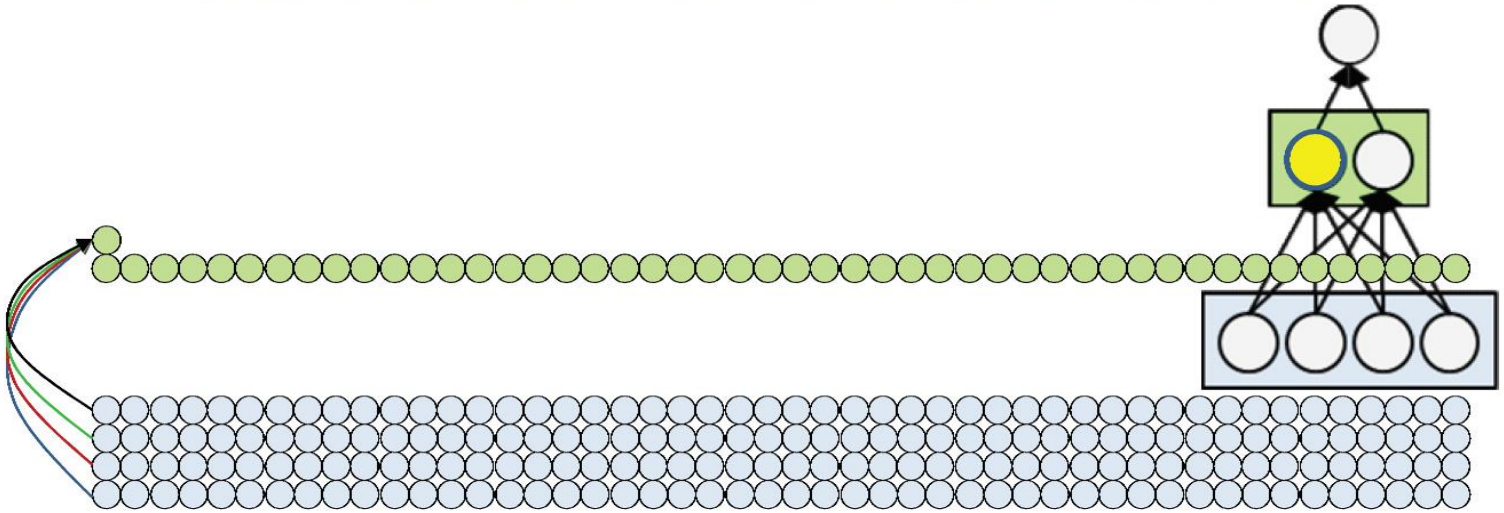
Lets do it in an different order

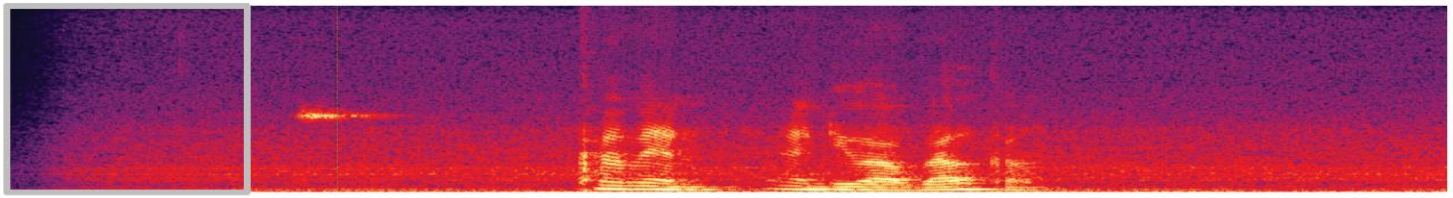




- But now, since the first layer neurons have already produced outputs for every location, each neuron in the second layer can go ahead and produce outputs for every position without waiting for the rest of the net
  - "Scan" the outputs of the first layer neurons

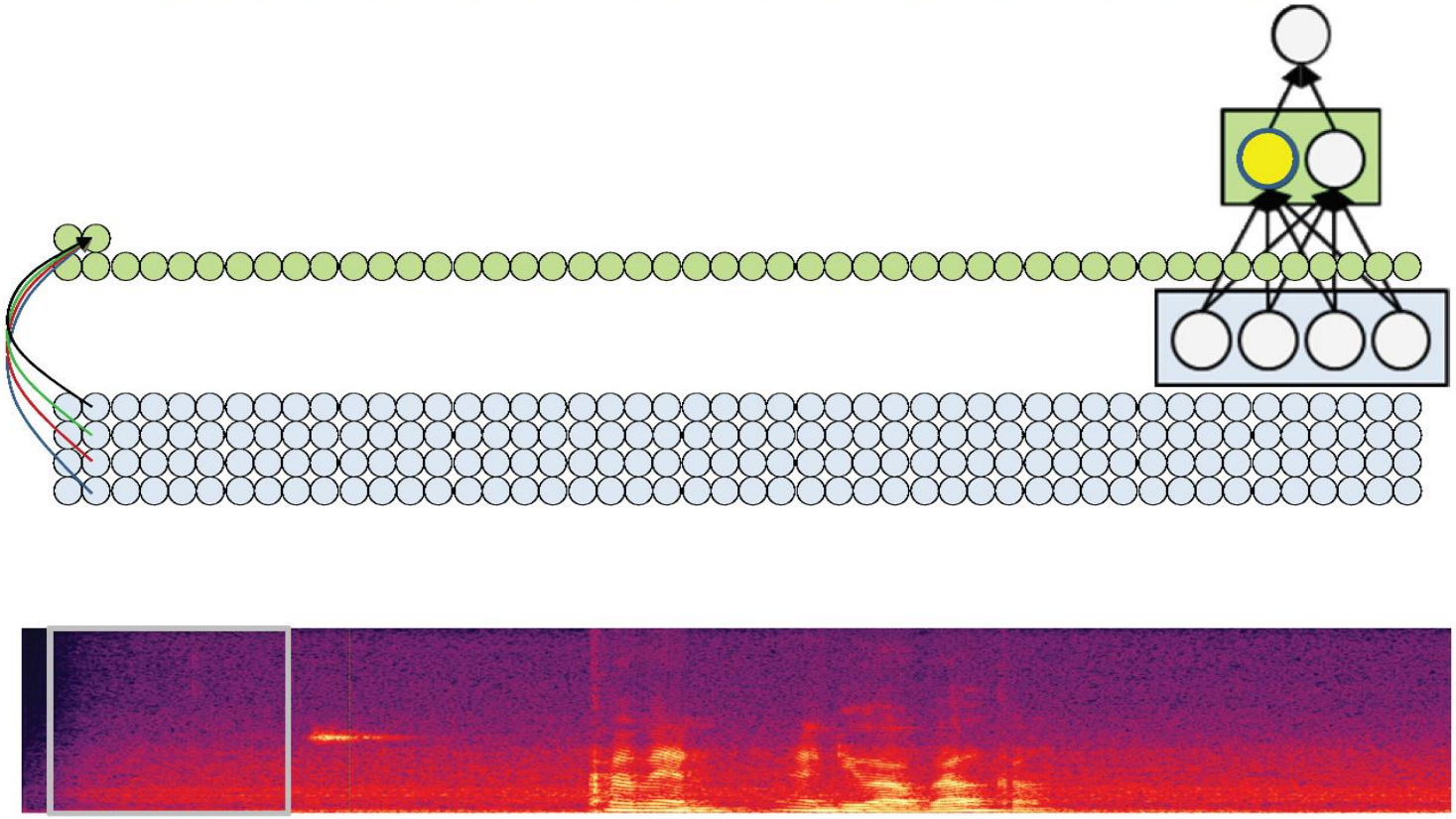
Lets do it in an different order





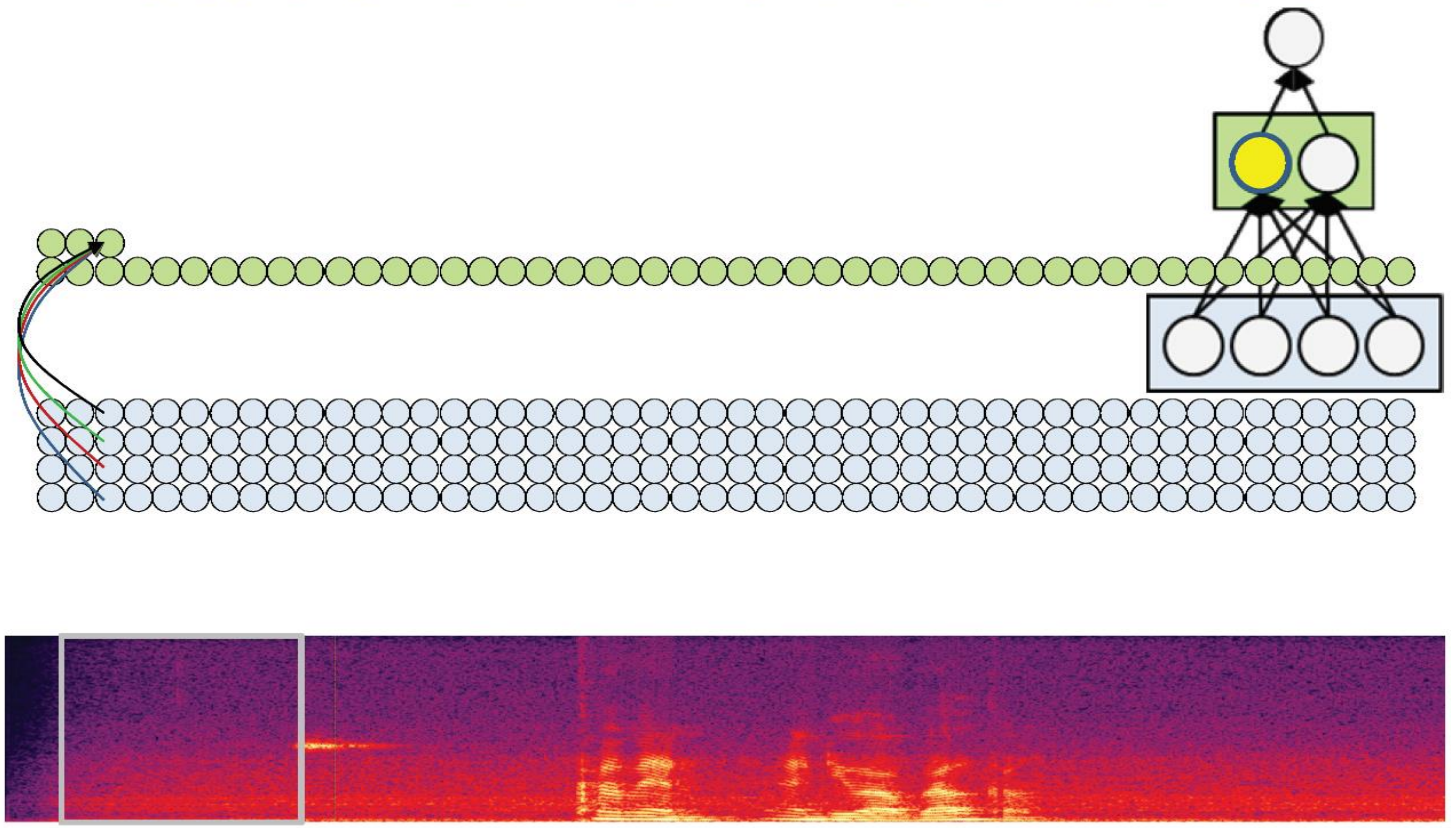
- But now, since the first layer neurons have already produced outputs for every location, each neuron in the second layer can go ahead and produce outputs for every position without waiting for the rest of the net
  - "Scan" the outputs of the first layer neurons

Lets do it in an different order



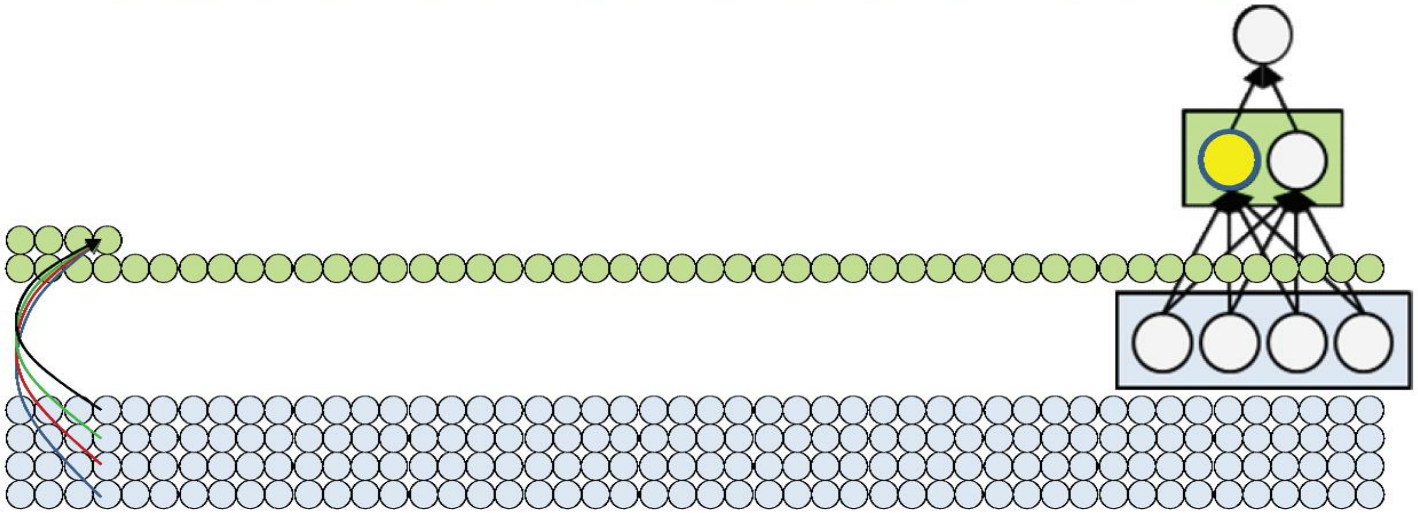
- But now, since the first layer neurons have already produced outputs for every location, each neuron in the second layer can go ahead and produce outputs for every position without waiting for the rest of the net
  - "Scan" the outputs of the first layer neurons

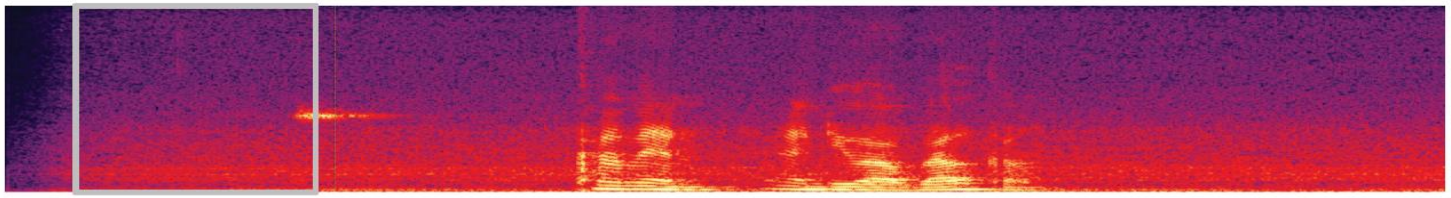
Lets do it in an different order



- But now, since the first layer neurons have already produced outputs for every location, each neuron in the second layer can go ahead and produce outputs for every position without waiting for the rest of the net
  - "Scan" the outputs of the first layer neurons

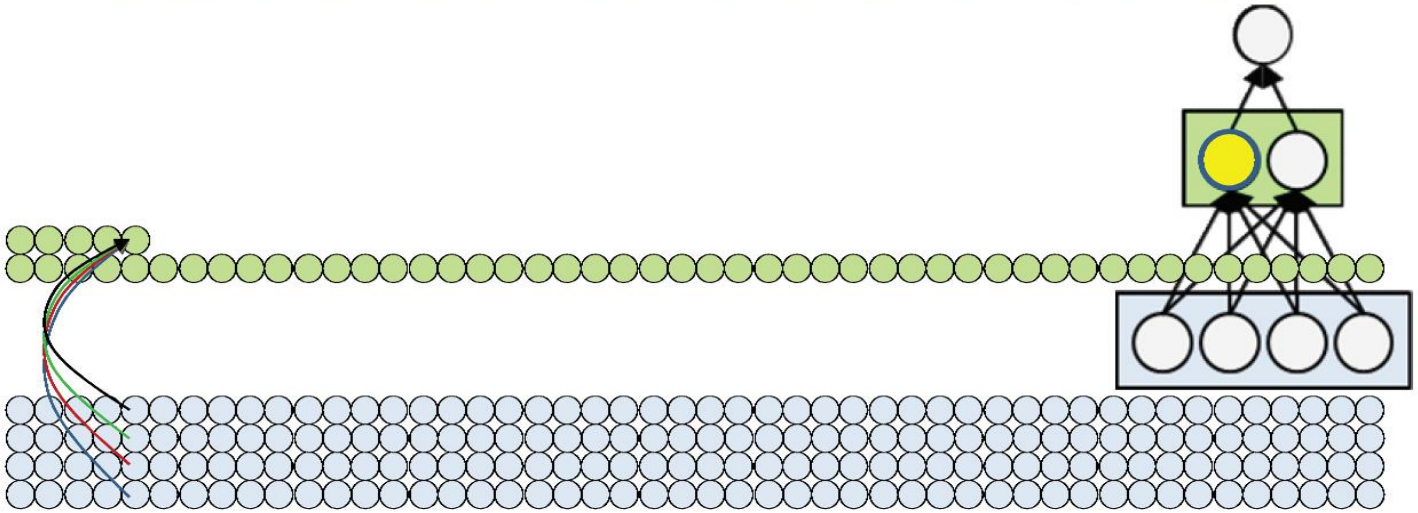
Lets do it in an different order

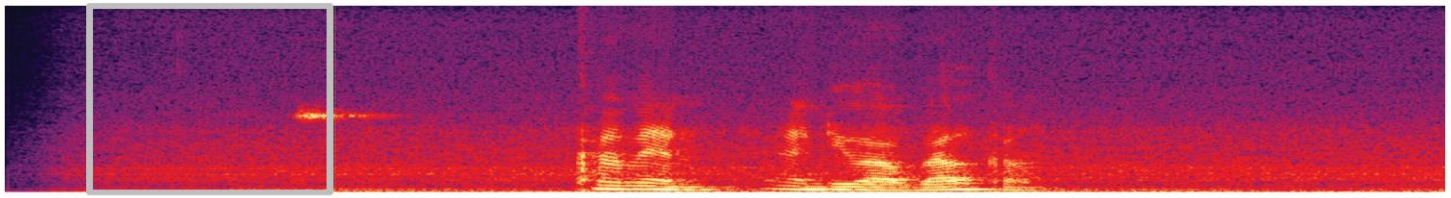




- But now, since the first layer neurons have already produced outputs for every location, each neuron in the second layer can go ahead and produce outputs for every position without waiting for the rest of the net
  - "Scan" the outputs of the first layer neurons

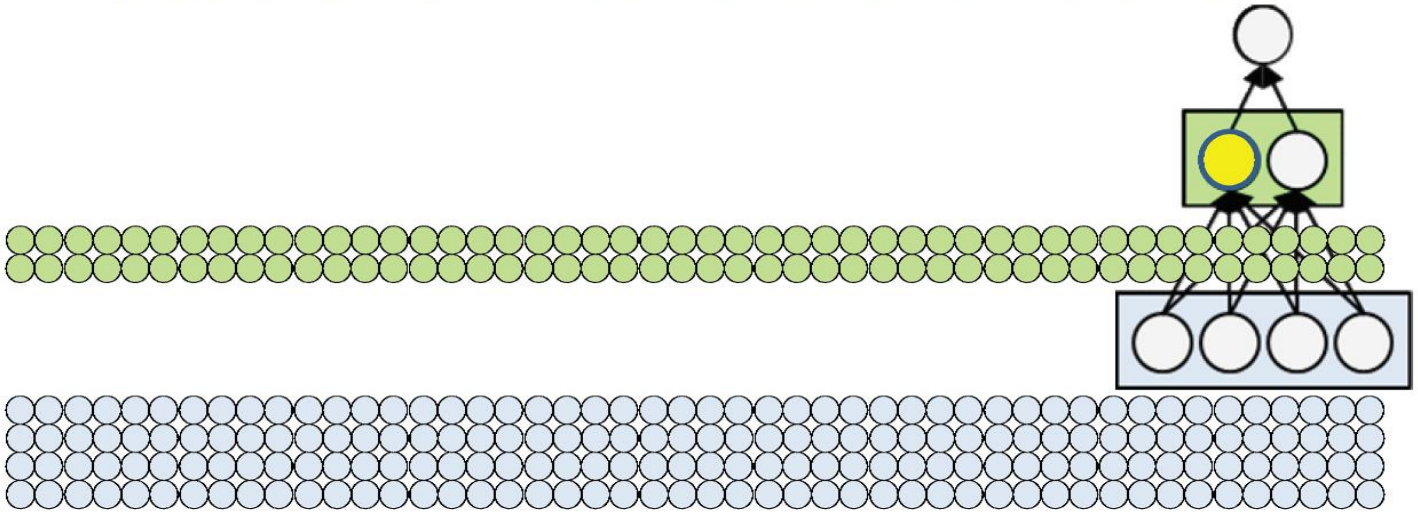
Lets do it in an different order

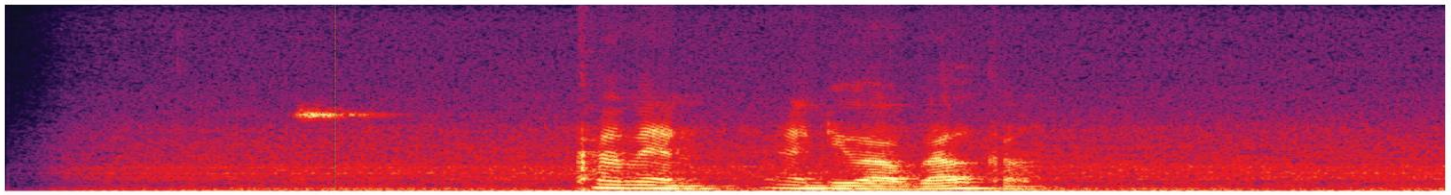




- But now, since the first layer neurons have already produced outputs for every location, each neuron in the second layer can go ahead and produce outputs for every position without waiting for the rest of the net
  - "Scan" the outputs of the first layer neurons

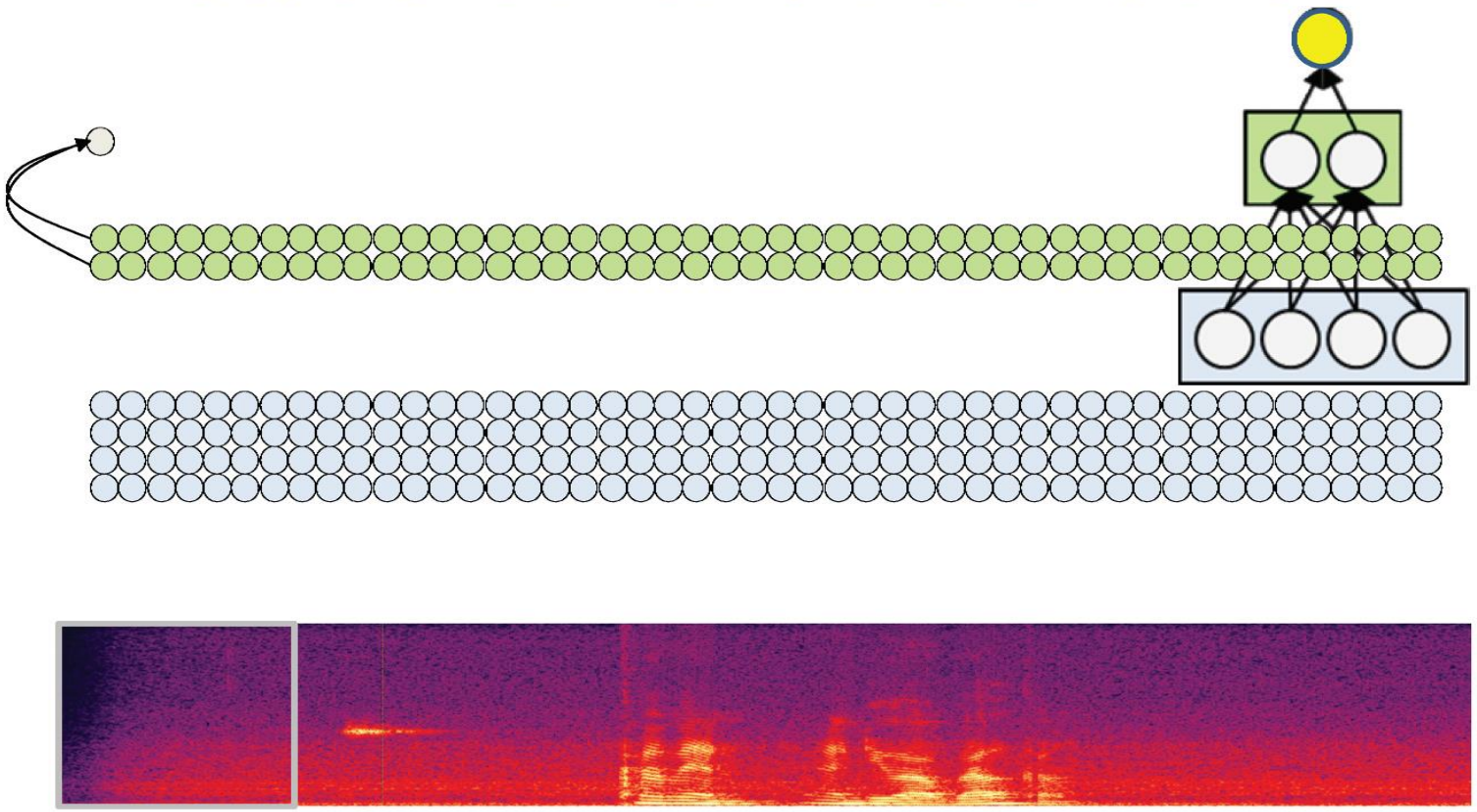
Lets do it in an different order





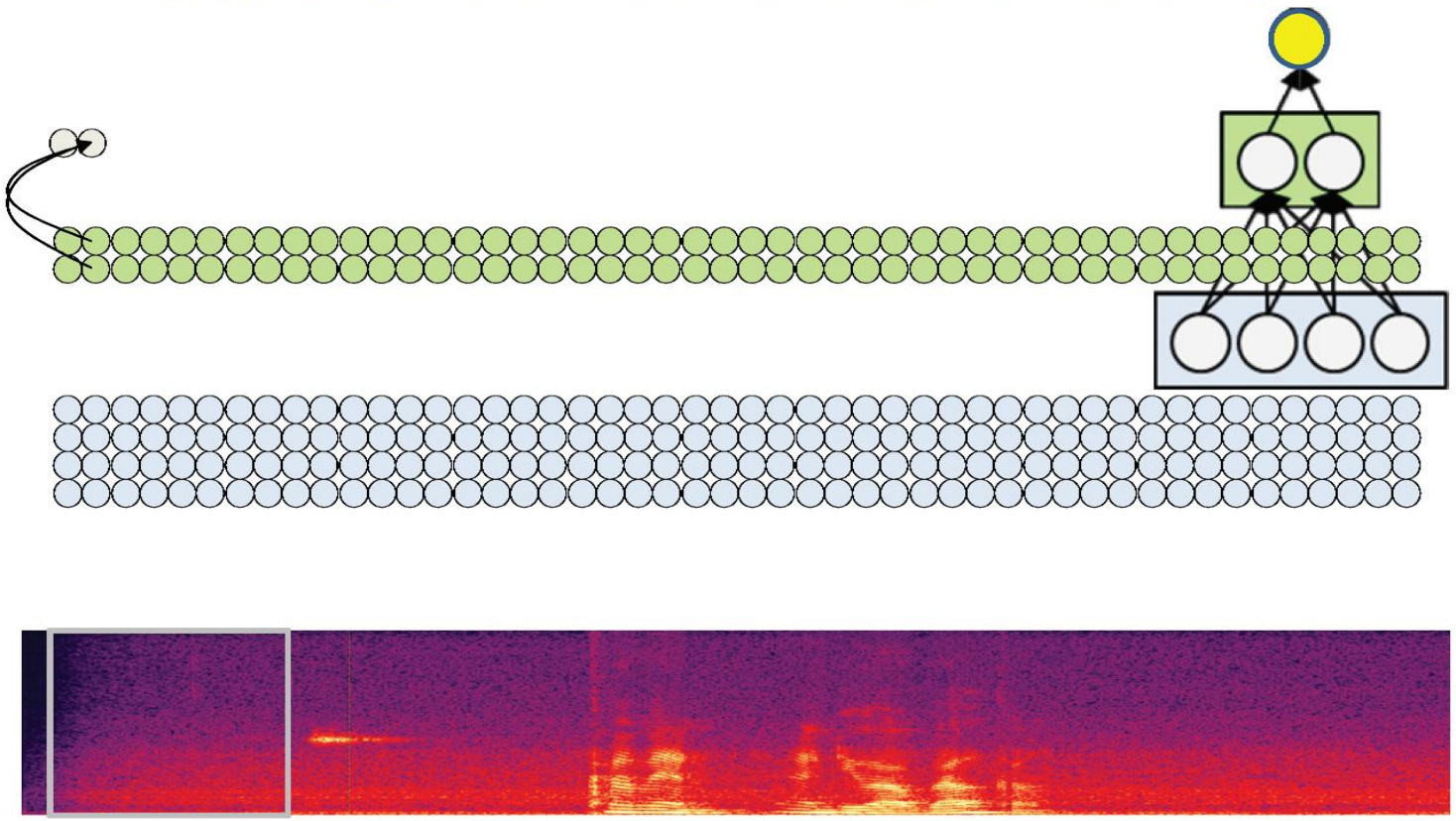
- But now, since the first layer neurons have already produced outputs for every location, each neuron in the second layer can go ahead and produce outputs for every position without waiting for the rest of the net
  - "Scan" the outputs of the first layer neurons

Lets do it in an different order



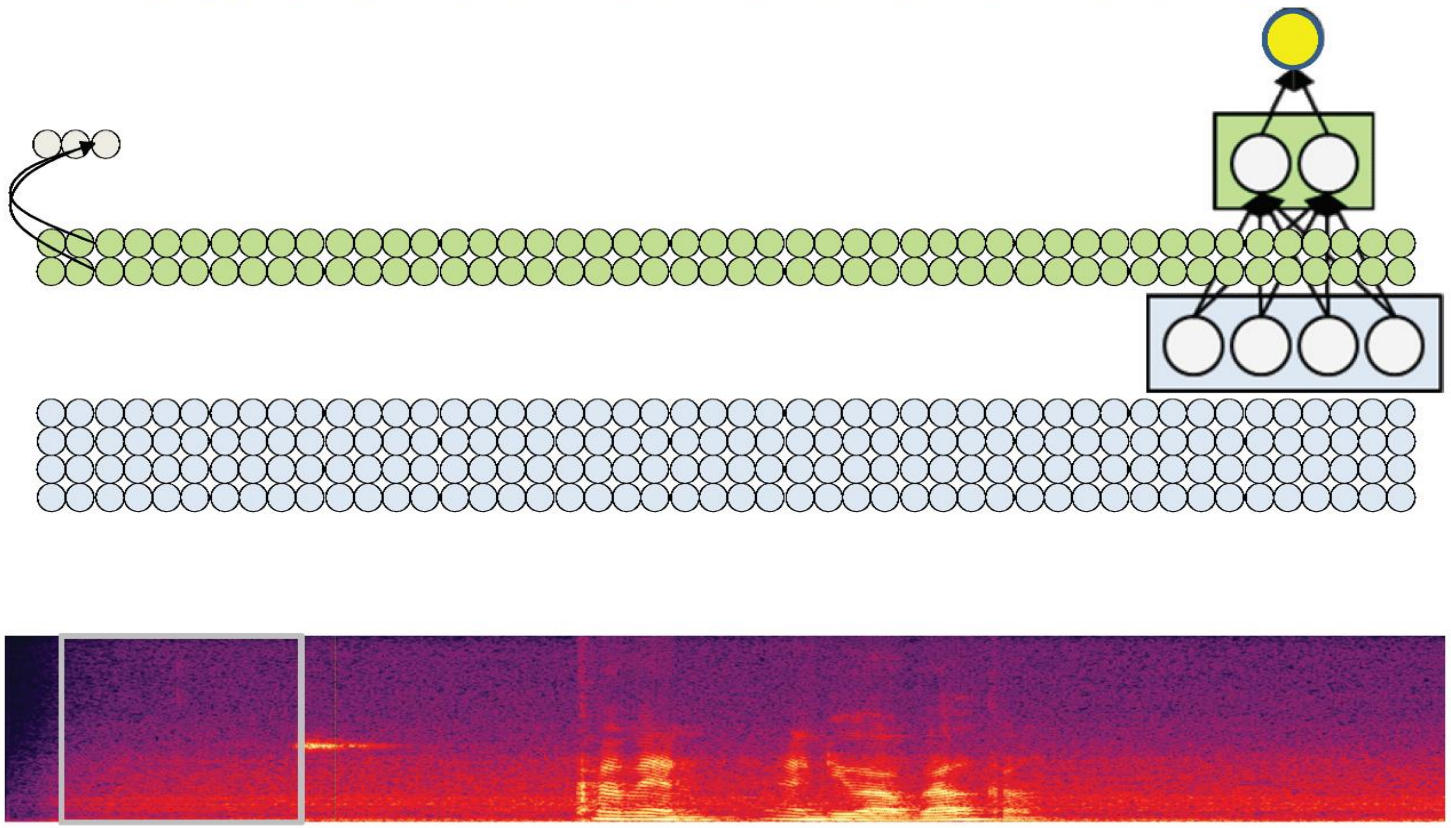
- At each position the output layer neurons can now operate on the outputs of the penultimate layer and produce the correct classification for the corresponding block!

Lets do it in an different order



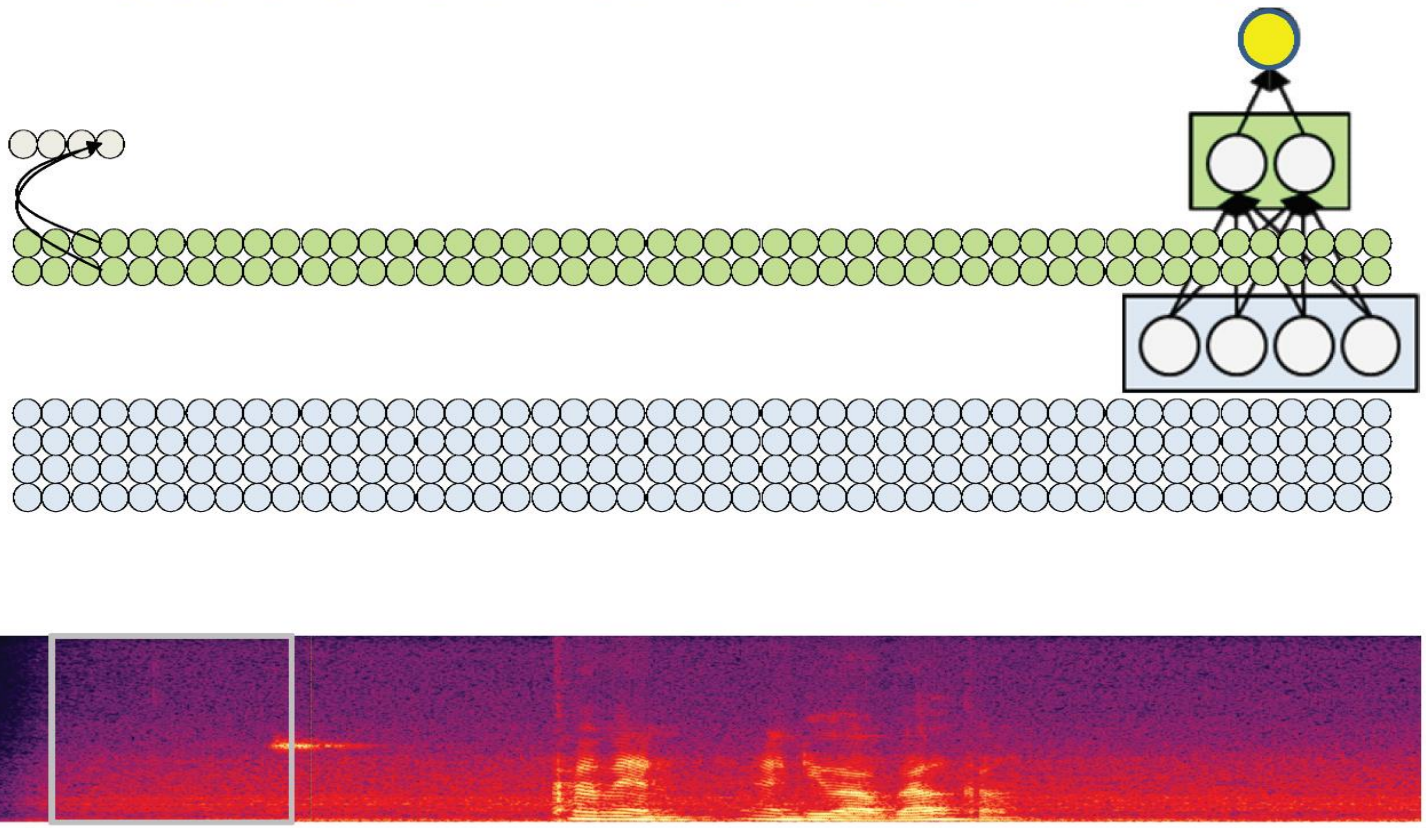
- At each position the output layer neurons can now operate on the outputs of the penultimate layer and produce the correct classification for the corresponding block!

Lets do it in an different order



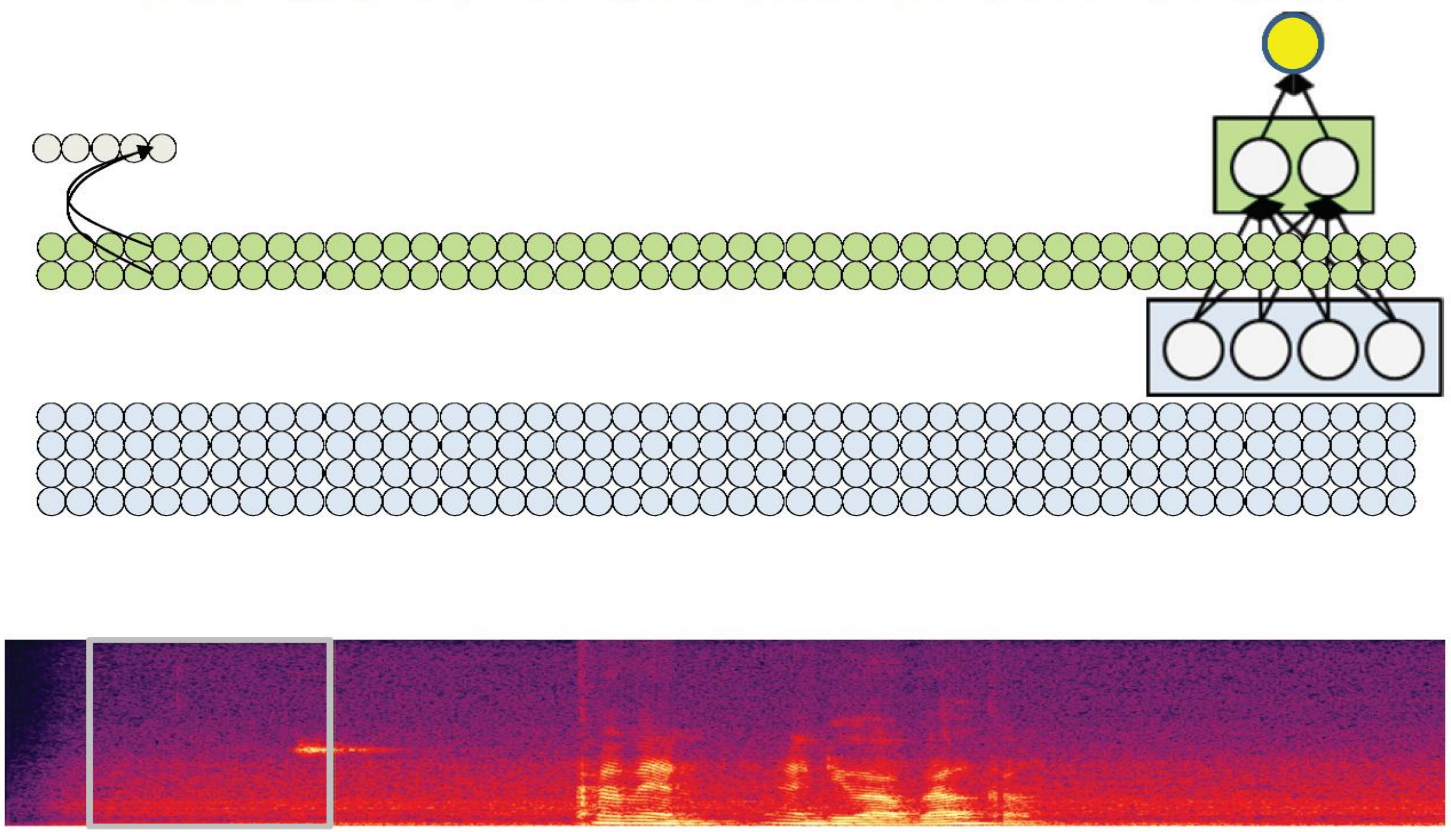
- At each position the output layer neurons can now operate on the outputs of the penultimate layer and produce the correct classification for the corresponding block!

Lets do it in an different order



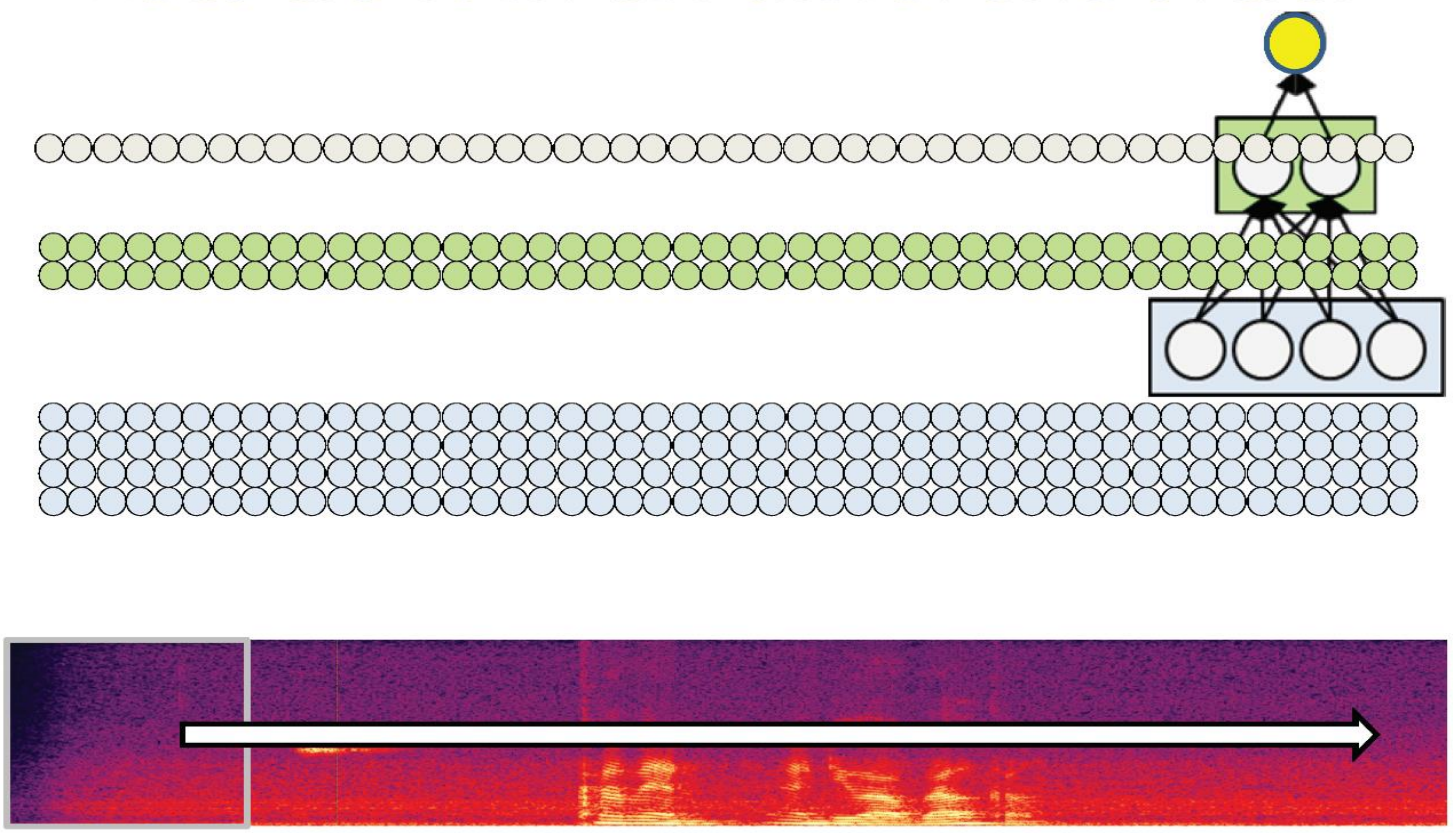
- At each position the output layer neurons can now operate on the outputs of the penultimate layer and produce the correct classification for the corresponding block!

Lets do it in an different order



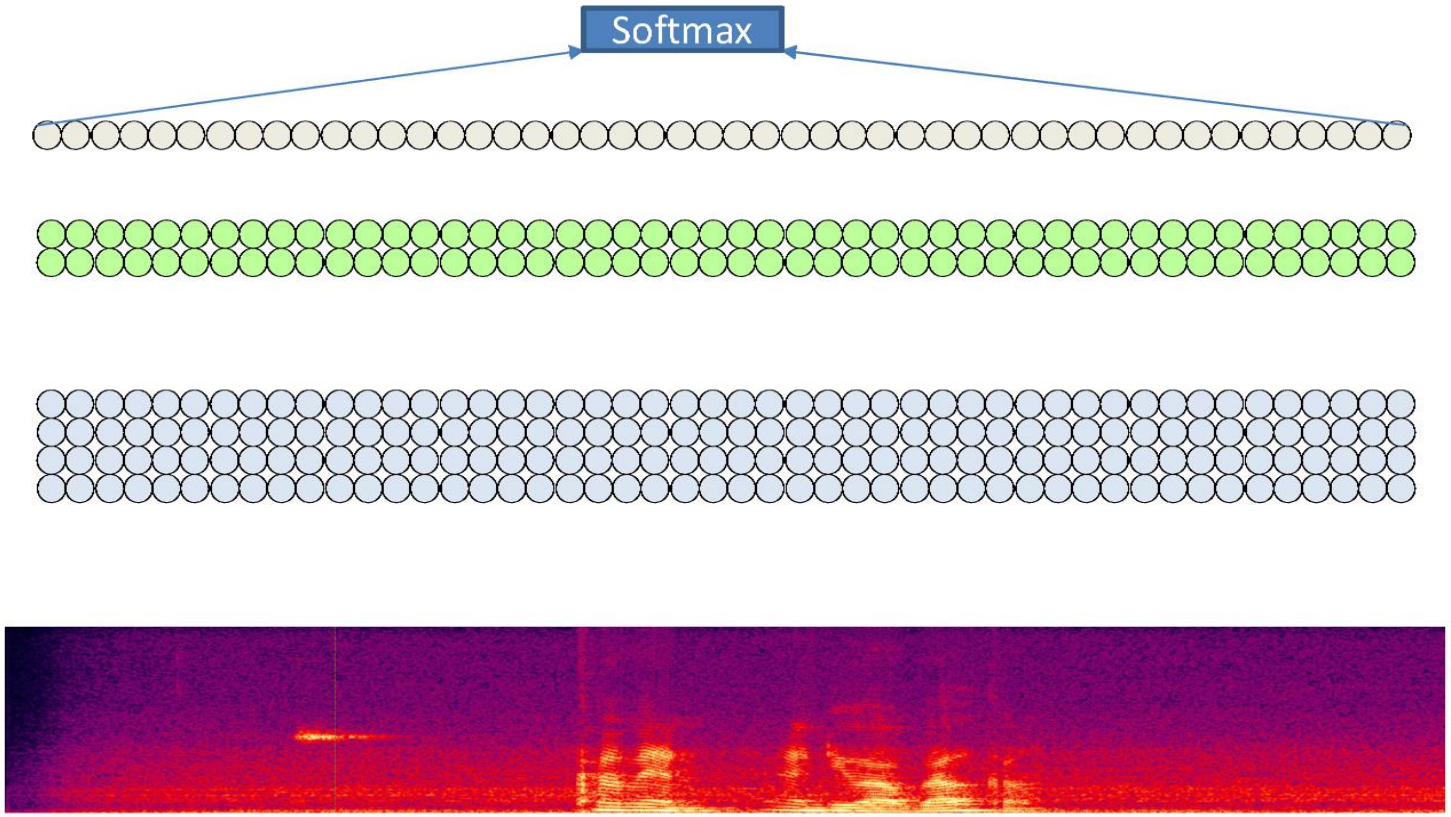
- At each position the output layer neurons can now operate on the outputs of the penultimate layer and produce the correct classification for the corresponding block!

Lets do it in an different order



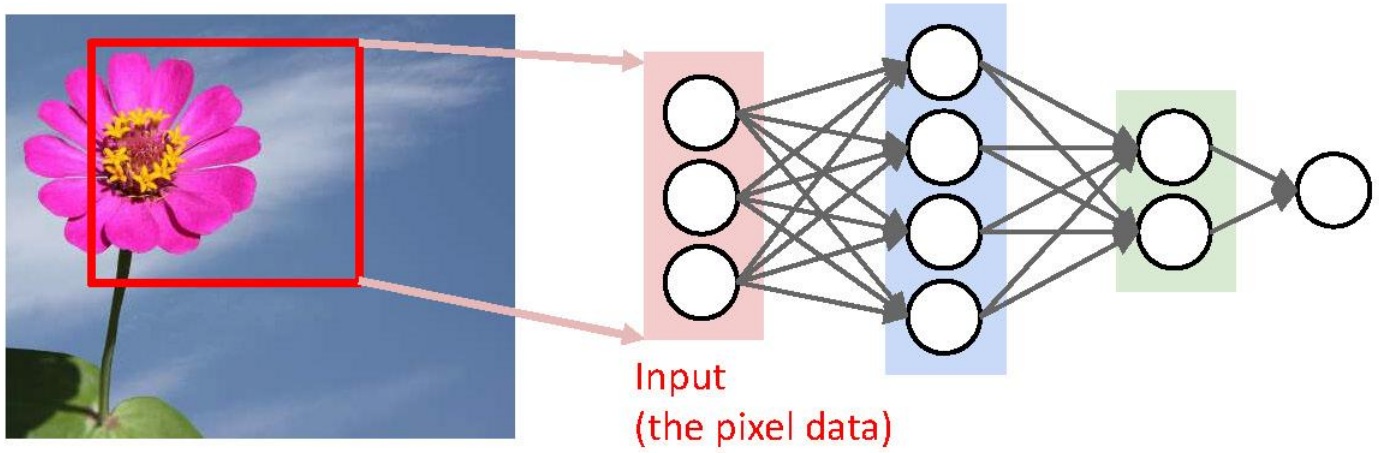
- At each position the output layer neurons can now operate on the outputs of the penultimate layer and produce the correct classification for the corresponding block!

Lets do it in an different order



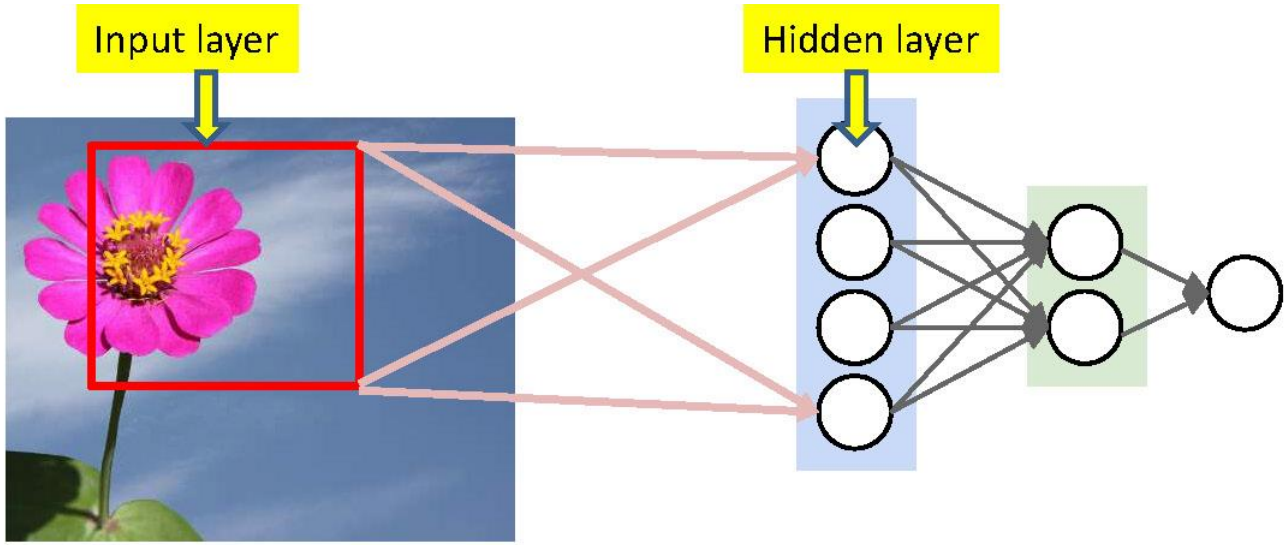
- At each position the output layer neurons can now operate on the outputs of the penultimate layer and produce the correct classification for the corresponding block!
  - The final softmax will give us the correct answer for the entire input

Scanning in 2D: A closer look



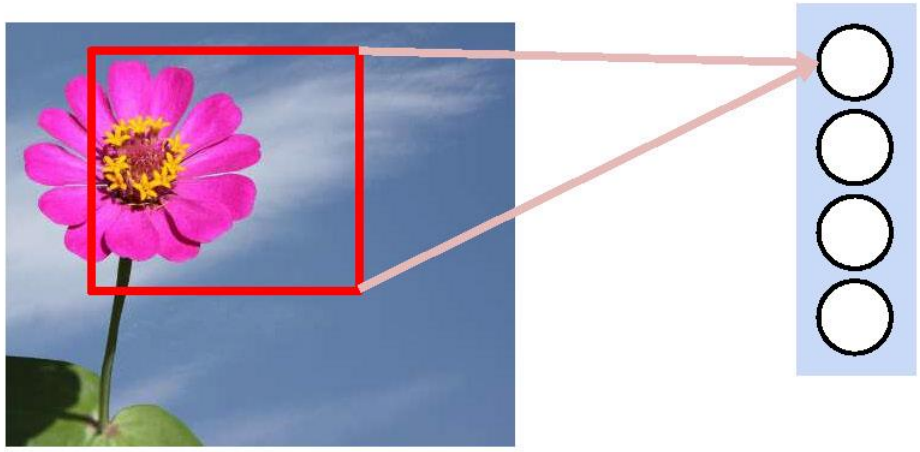
- Scan for the desired object
- At each location, the entire region is sent through an MLP

Scanning in 2D: A closer look



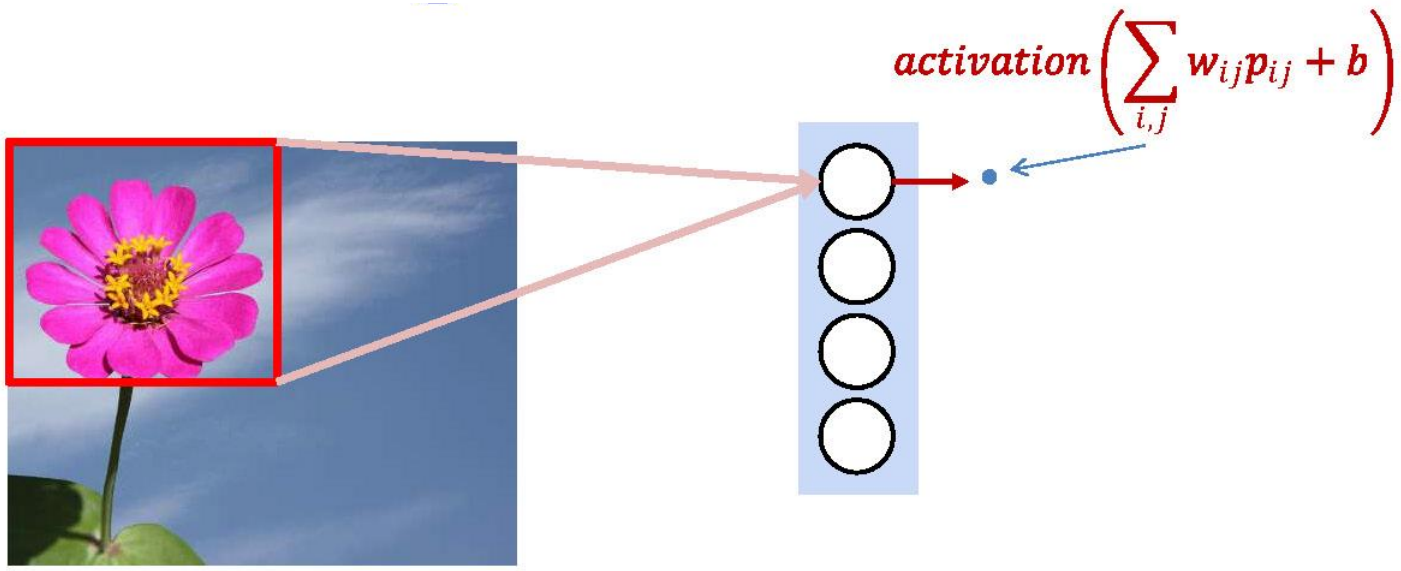
- The "input layer" is just the pixels in the image connecting to the hidden layer

Scanning in 2D: A closer look



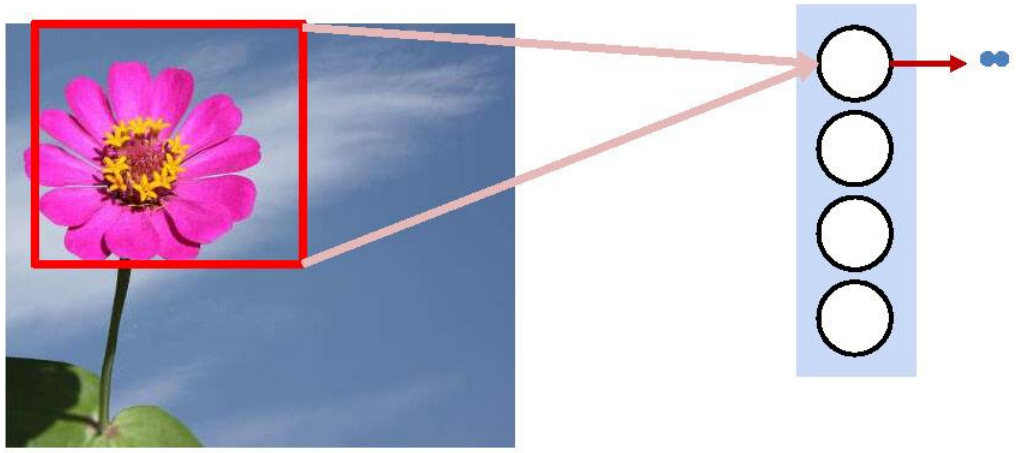
- Consider a single neuron

Scanning in 2D: A closer look



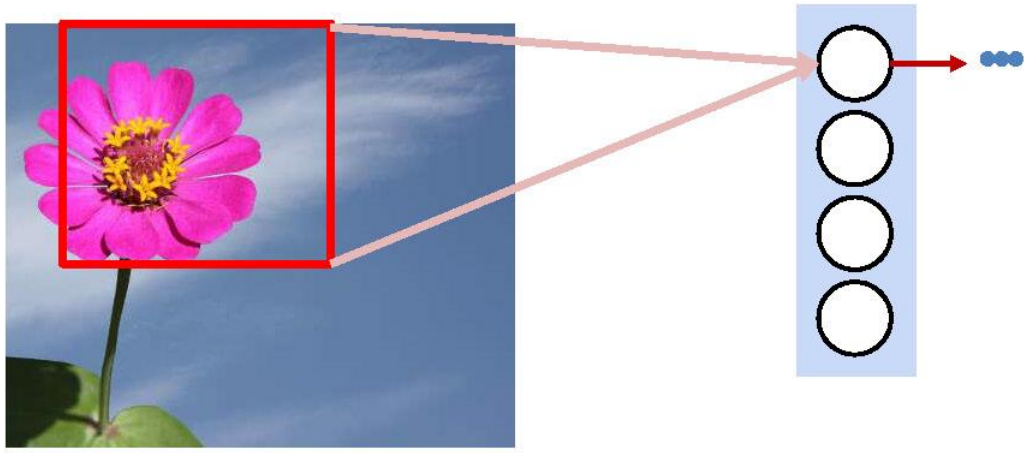
- Consider a single perceptron
- At each position of the box, the perceptron is evaluating the part of the picture in the box as part of the classification for that region
  - We could arrange the outputs of the neurons for each position correspondingly to the original picture

Scanning in 2D: A closer look



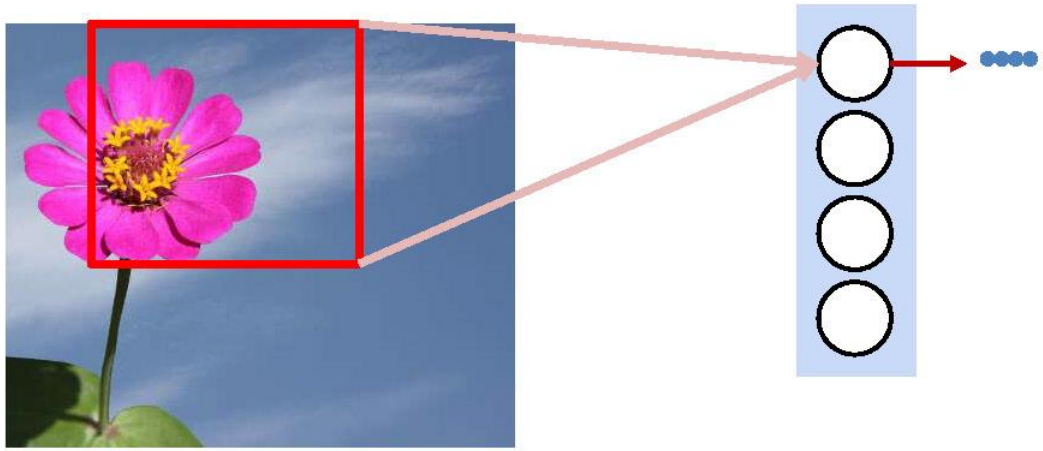
- Consider a single perceptron
- At each position of the box, the perceptron is evaluating the part of the picture in the box as part of the classification for that region
  - We could arrange the outputs of the neurons for each position correspondingly to the original picture

Scanning in 2D: A closer look



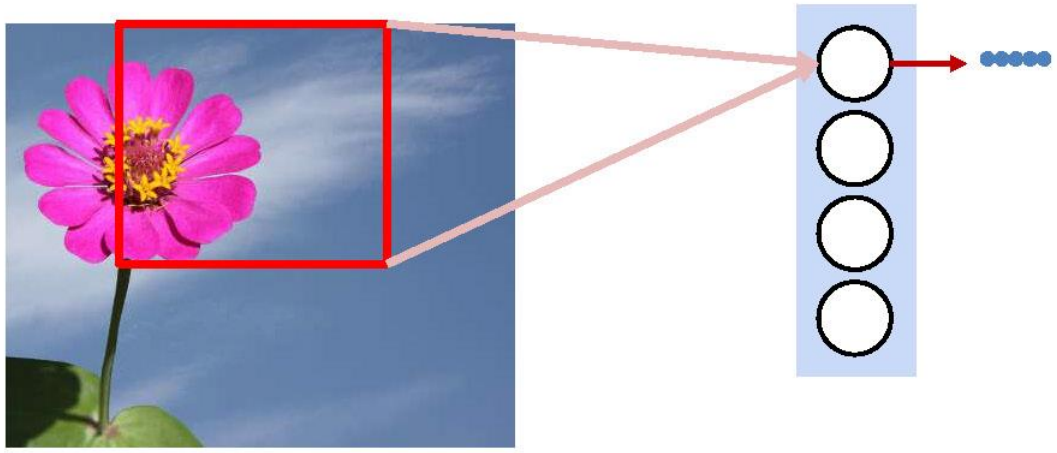
- Consider a single perceptron
- At each position of the box, the perceptron is evaluating the part of the picture in the box as part of the classification for that region
  - We could arrange the outputs of the neurons for each position correspondingly to the original picture

Scanning in 2D: A closer look



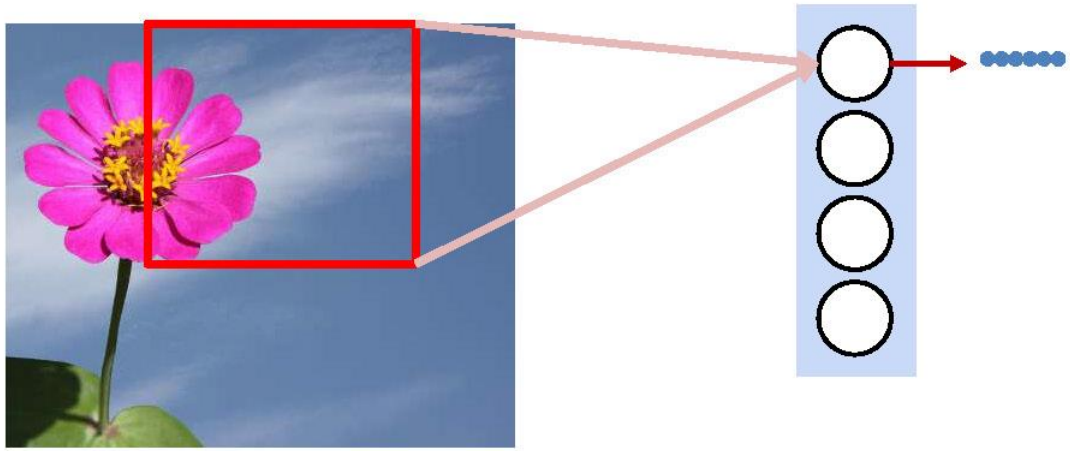
- Consider a single perceptron
- At each position of the box, the perceptron is evaluating the part of the picture in the box as part of the classification for that region
  - We could arrange the outputs of the neurons for each position correspondingly to the original picture

Scanning in 2D: A closer look



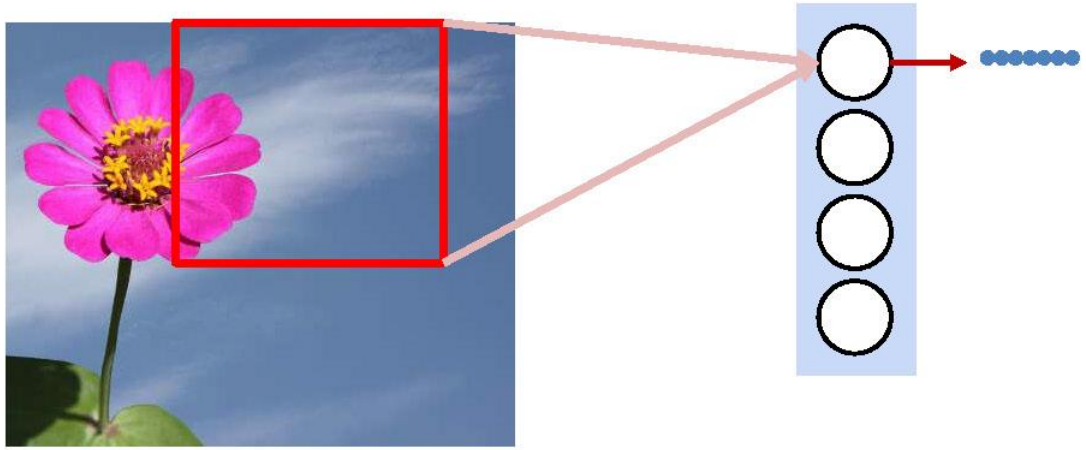
- Consider a single perceptron
- At each position of the box, the perceptron is evaluating the part of the picture in the box as part of the classification for that region
  - We could arrange the outputs of the neurons for each position correspondingly to the original picture

Scanning in 2D: A closer look



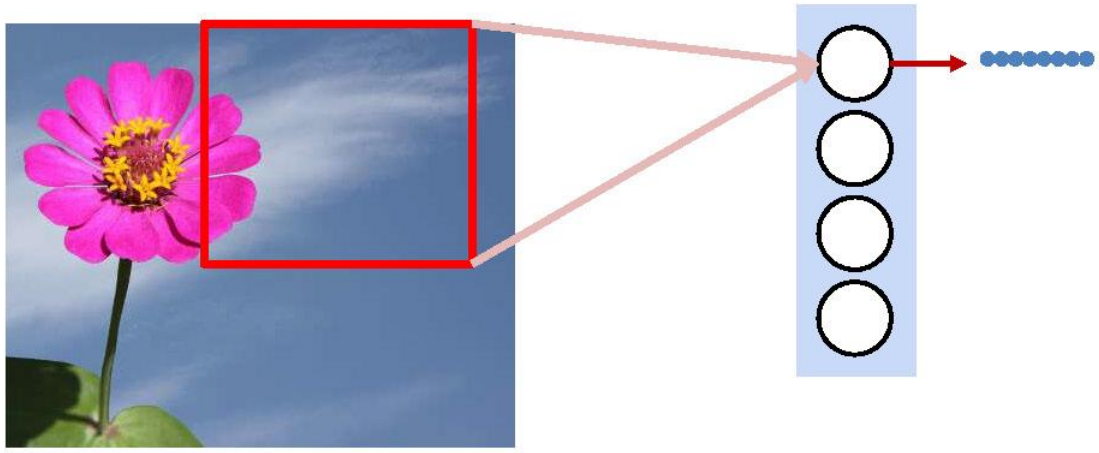
- Consider a single perceptron
- At each position of the box, the perceptron is evaluating the part of the picture in the box as part of the classification for that region
  - We could arrange the outputs of the neurons for each position correspondingly to the original picture

Scanning in 2D: A closer look



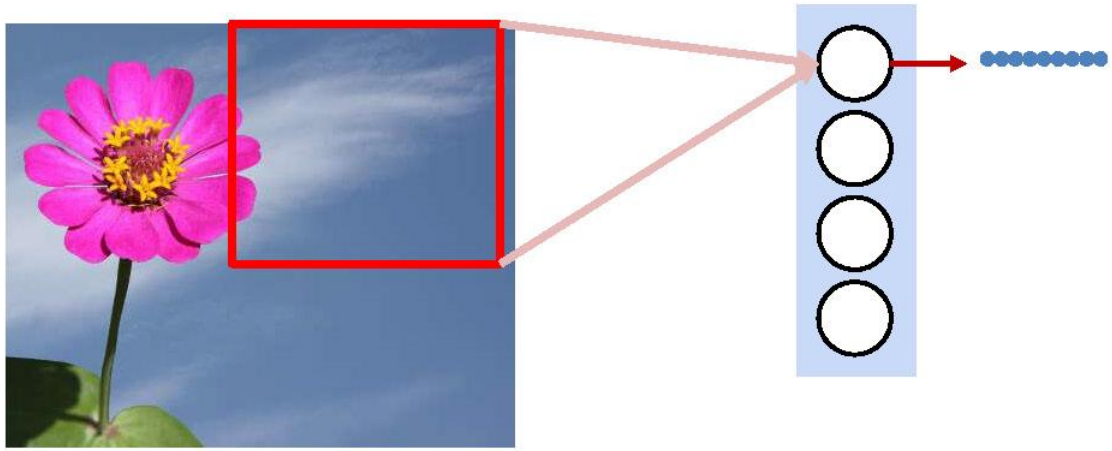
- Consider a single perceptron
- At each position of the box, the perceptron is evaluating the part of the picture in the box as part of the classification for that region
  - We could arrange the outputs of the neurons for each position correspondingly to the original picture

Scanning in 2D: A closer look



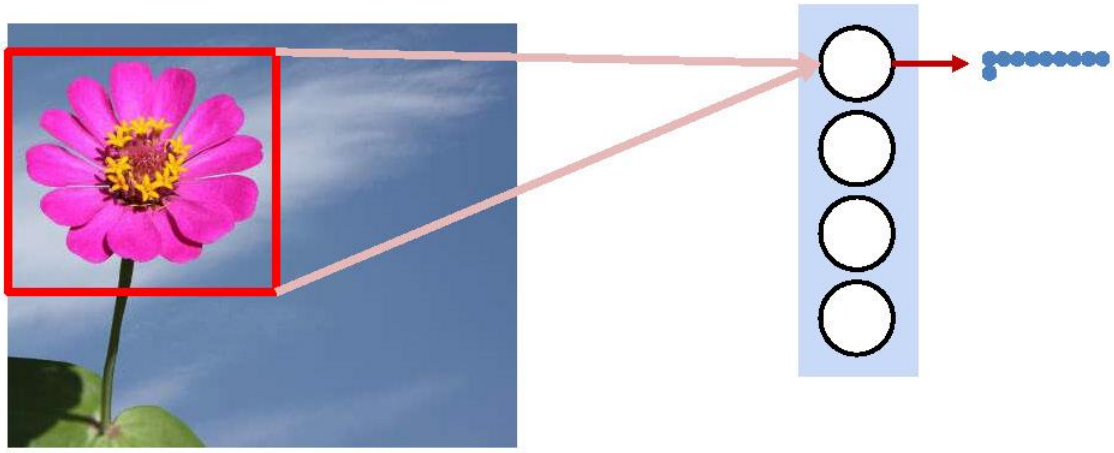
- Consider a single perceptron
- At each position of the box, the perceptron is evaluating the part of the picture in the box as part of the classification for that region
  - We could arrange the outputs of the neurons for each position correspondingly to the original picture

Scanning in 2D: A closer look



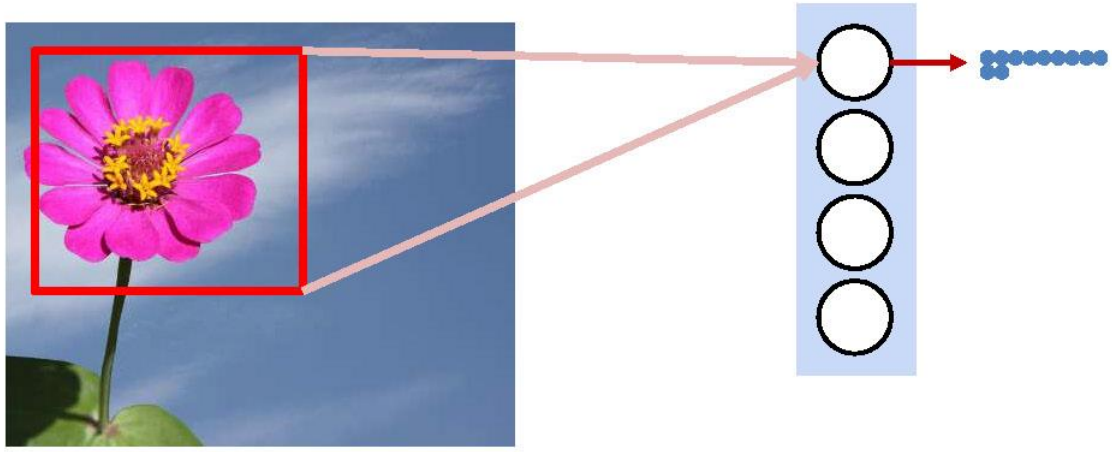
- Consider a single perceptron
- At each position of the box, the perceptron is evaluating the part of the picture in the box as part of the classification for that region
  - We could arrange the outputs of the neurons for each position correspondingly to the original picture

Scanning in 2D: A closer look



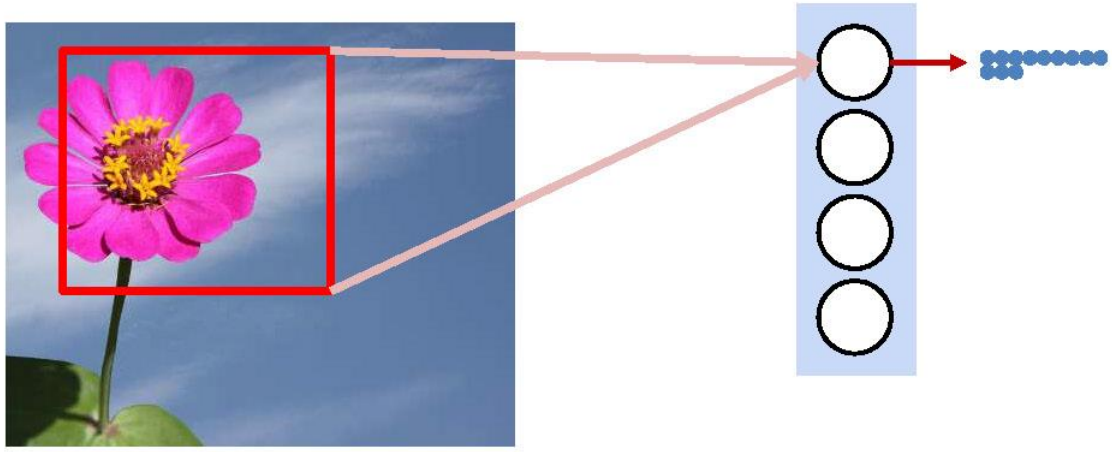
- Consider a single perceptron
- At each position of the box, the perceptron is evaluating the part of the picture in the box as part of the classification for that region
  - We could arrange the outputs of the neurons for each position correspondingly to the original picture

Scanning in 2D: A closer look



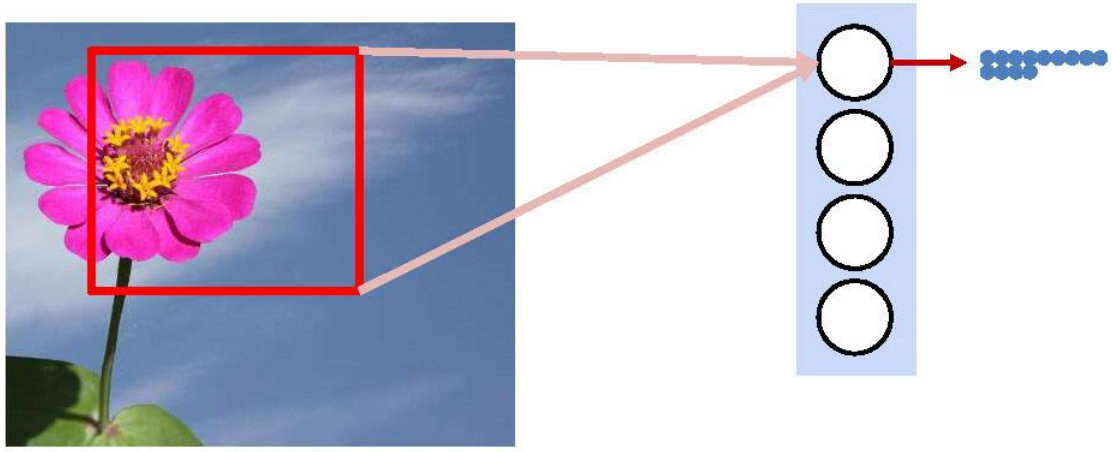
- Consider a single perceptron
- At each position of the box, the perceptron is evaluating the part of the picture in the box as part of the classification for that region
  - We could arrange the outputs of the neurons for each position correspondingly to the original picture

Scanning in 2D: A closer look



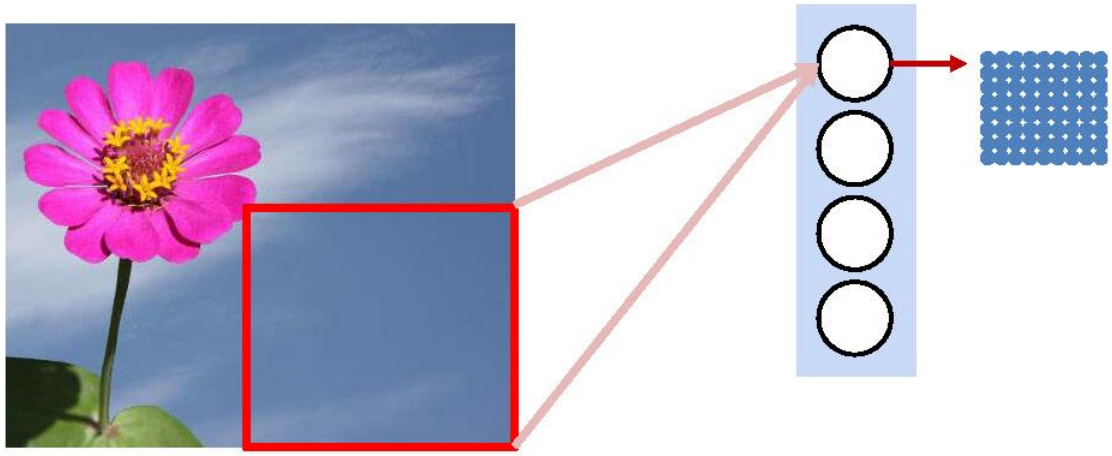
- Consider a single perceptron
- At each position of the box, the perceptron is evaluating the part of the picture in the box as part of the classification for that region
  - We could arrange the outputs of the neurons for each position correspondingly to the original picture

Scanning in 2D: A closer look



- Consider a single perceptron
- At each position of the box, the perceptron is evaluating the part of the picture in the box as part of the classification for that region
  - We could arrange the outputs of the neurons for each position correspondingly to the original picture

Scanning in 2D: A closer look

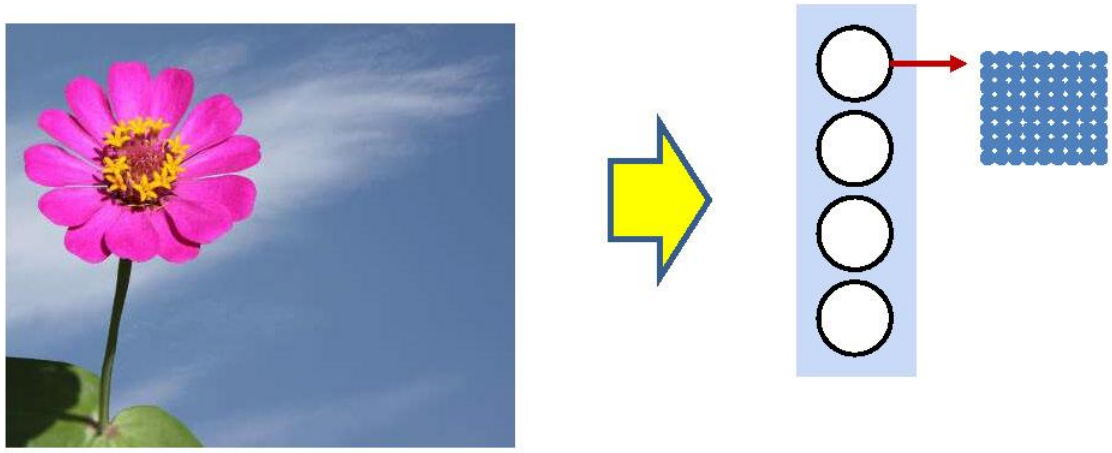


- Consider a single perceptron
- At each position of the box, the perceptron is evaluating the part of the picture in the box as part of the classification for that region

– We could arrange the outputs of the neurons for each position correspondingly to the original picture

Eventually, we can arrange the outputs from the response at the scanned positions into a rectangle that's proportional in size to the original picture

Scanning in 2D: A closer look

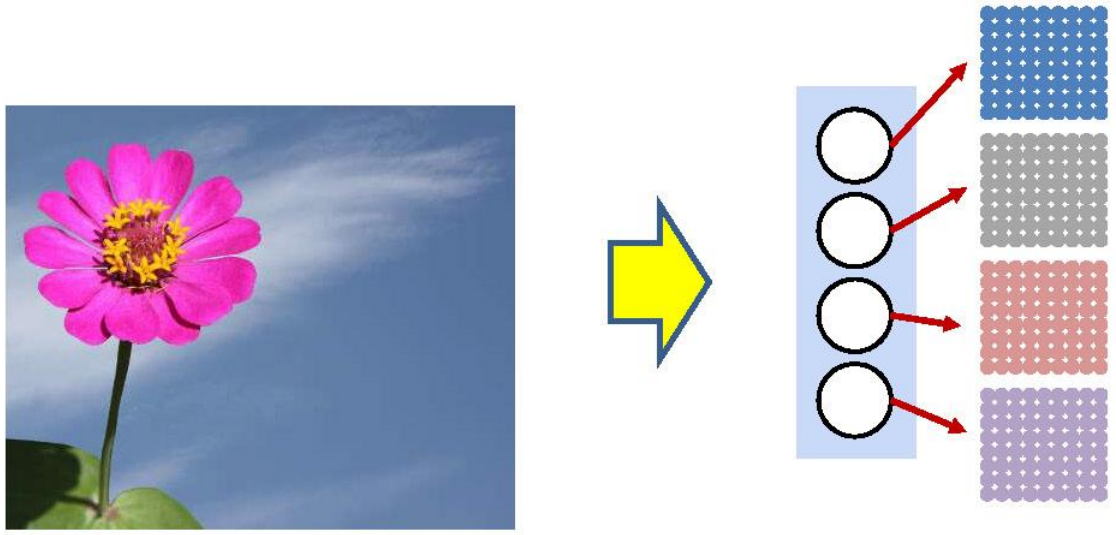


- Consider a single perceptron
- At each position of the box, the perceptron is evaluating the part of the picture in the box as part of the classification for that region

  – We could arrange the outputs of the neurons for each position correspondingly to the original picture

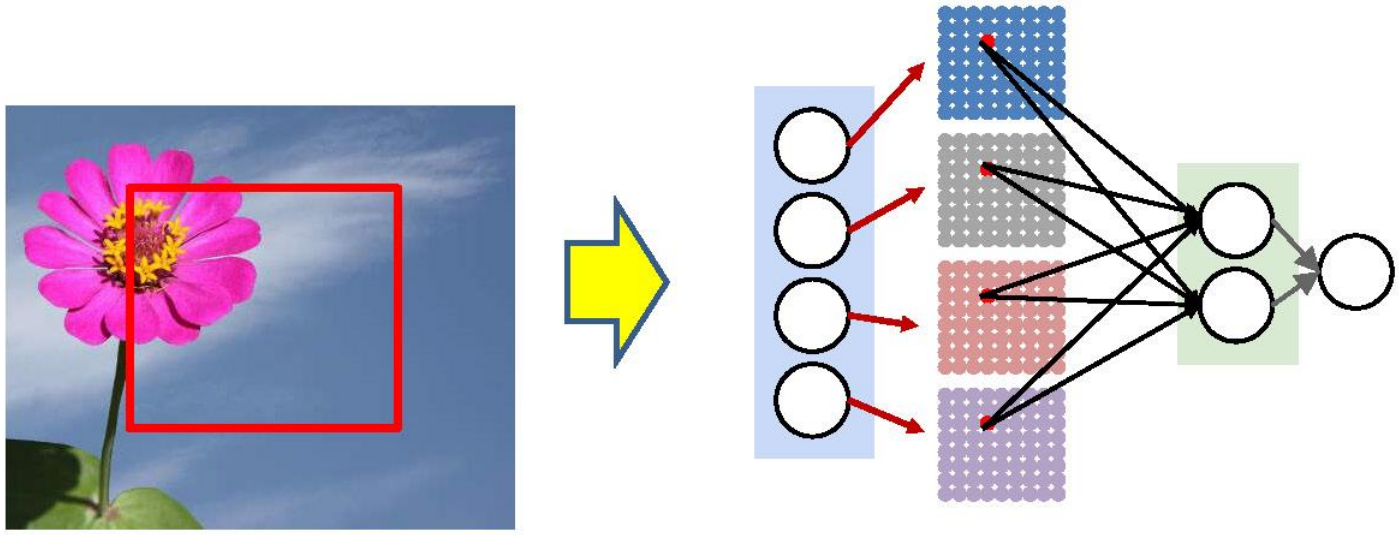
Eventually, we can arrange the outputs from the response at the scanned positions into a rectangle that's proportional in size to the original picture

Scanning in 2D: A closer look



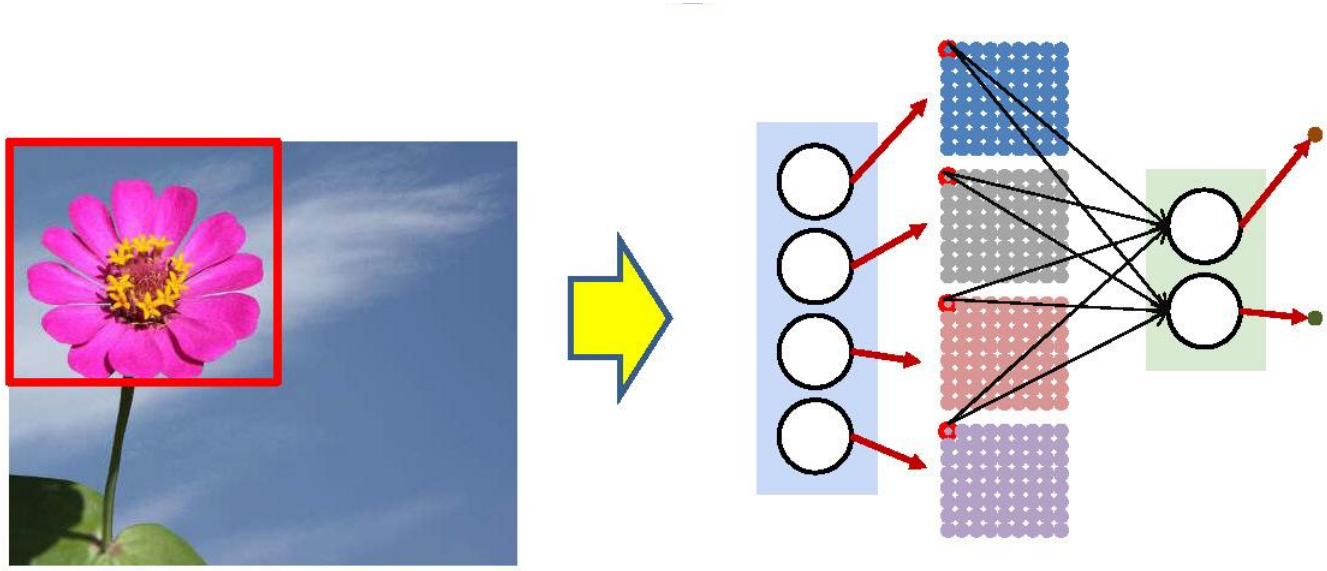
- Similarly, each first-layer perceptron's outputs from the scanned positions can be arranged as a rectangular pattern

Scanning in 2D: A closer look



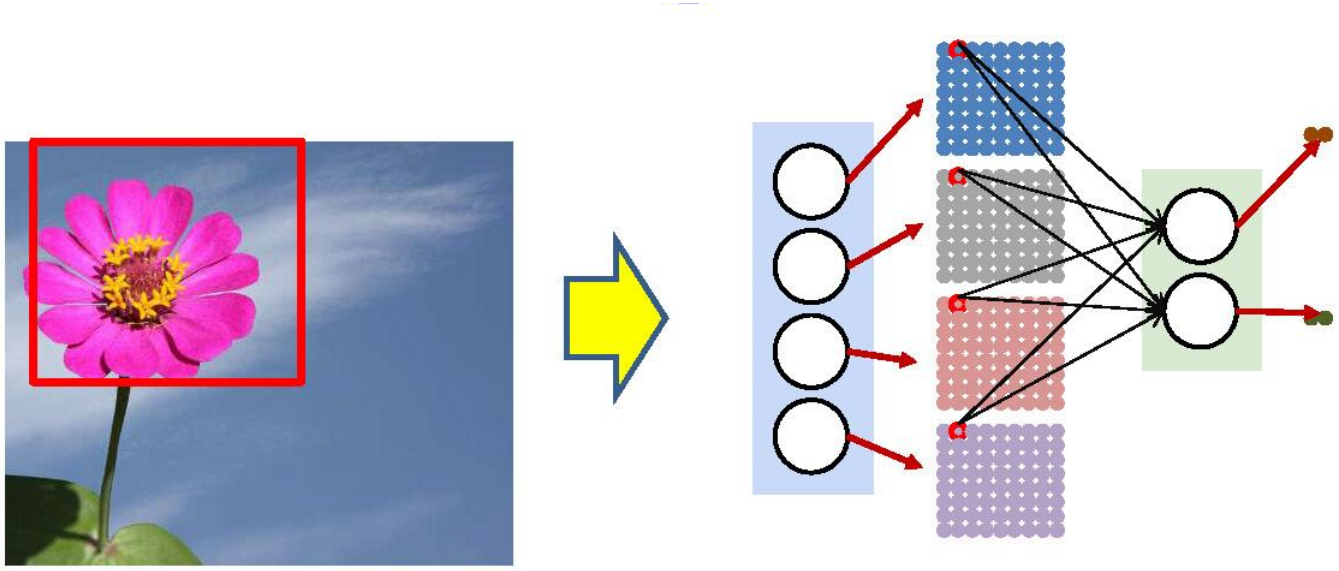
- To classify a specific "patch" in the image, we send the first level activations from the positions corresponding to that position to the next layer

Scanning in 2D: A closer look



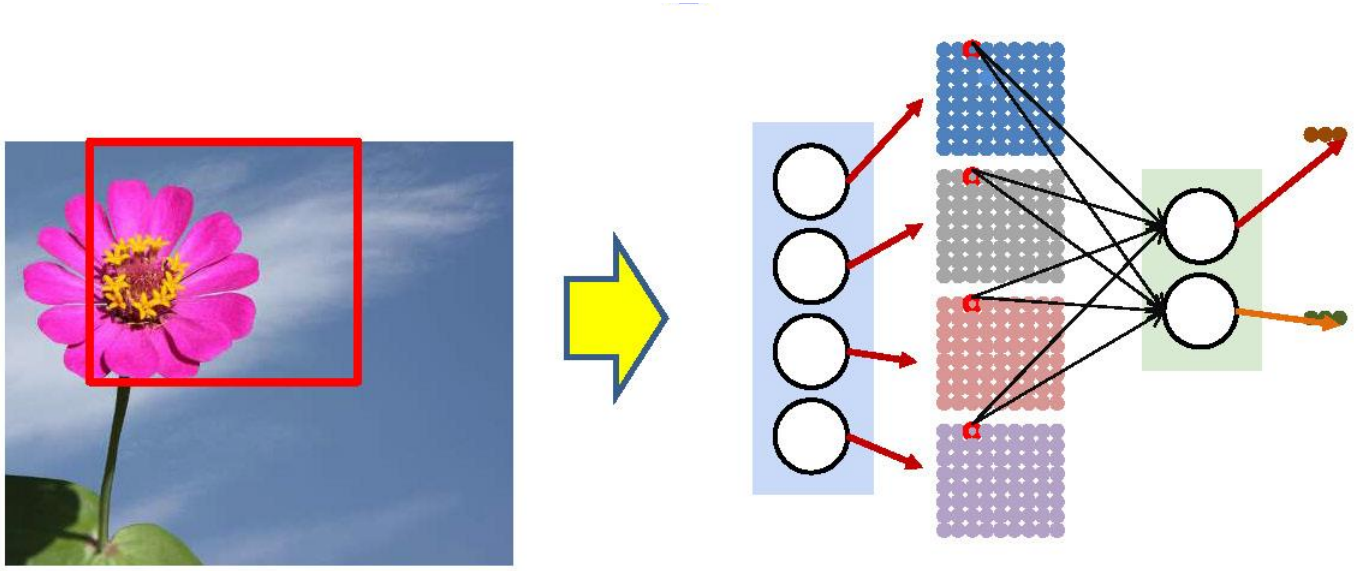
- We can recurse the logic
- – The second level neurons too are "scanning" the rectangular outputs of the first-level neurons
  - – (Un)like the first level, they are jointly scanning multiple "pictures"
- Each location in the output of the second level neuron considers the corresponding locations from the outputs of all the first-level neurons

Scanning in 2D: A closer look



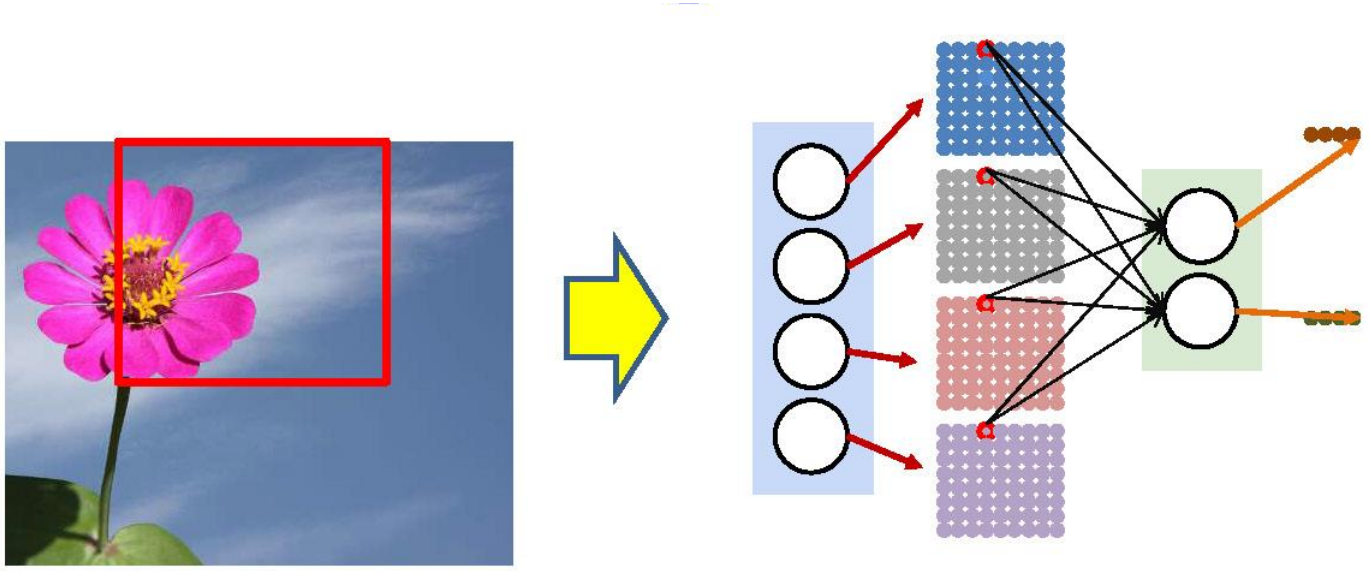
- We can recurse the logic
- – The second level neurons too are "scanning" the rectangular outputs of the first-level neurons
  - – (Un)like the first level, they are jointly scanning multiple "pictures"
- Each location in the output of the second level neuron considers the corresponding locations from the outputs of all the first-level neurons

Scanning in 2D: A closer look



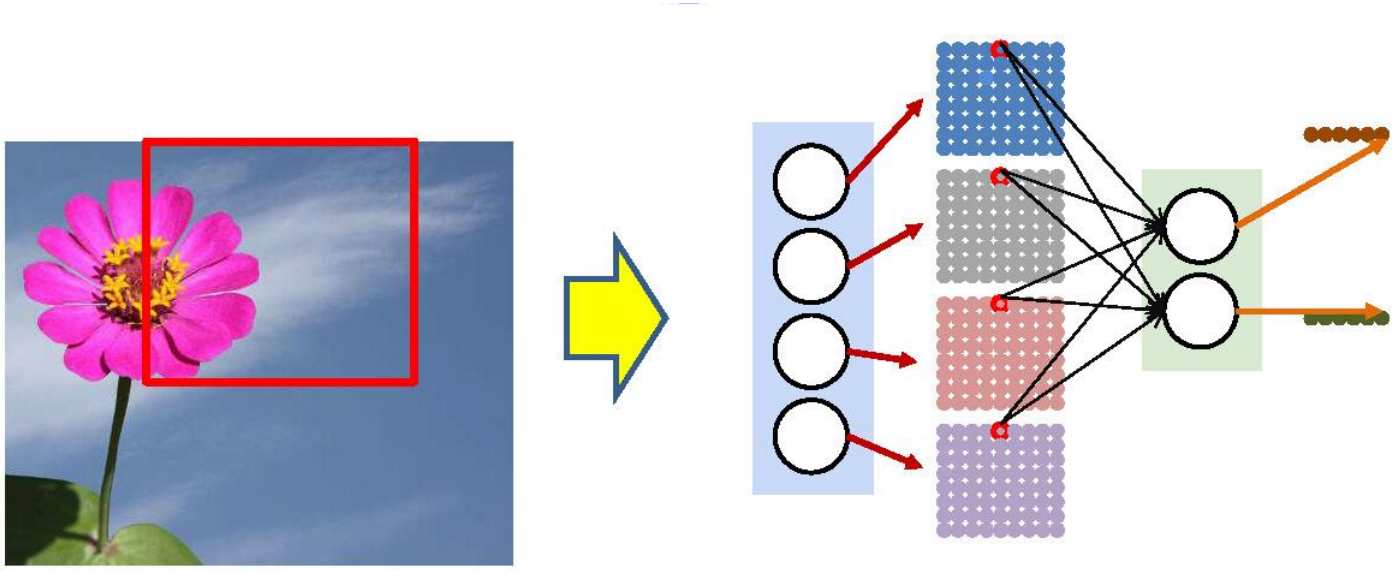
- We can recurse the logic
- – The second level neurons too are "scanning" the rectangular outputs of the first-level neurons
  - – (Un)like the first level, they are jointly scanning multiple "pictures"
- Each location in the output of the second level neuron considers the corresponding locations from the outputs of all the first-level neurons

Scanning in 2D: A closer look



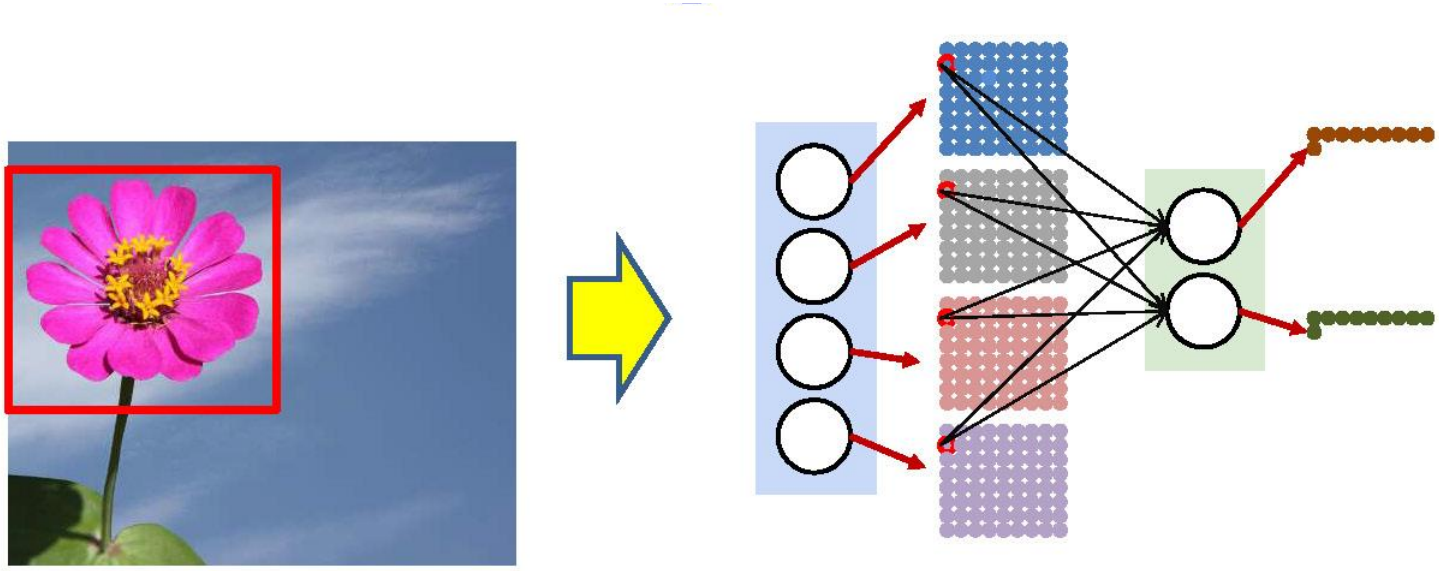
- We can recurse the logic
- – The second level neurons too are "scanning" the rectangular outputs of the first-level neurons
  - – (Un)like the first level, they are jointly scanning multiple "pictures"
- Each location in the output of the second level neuron considers the corresponding locations from the outputs of all the first-level neurons

Scanning in 2D: A closer look



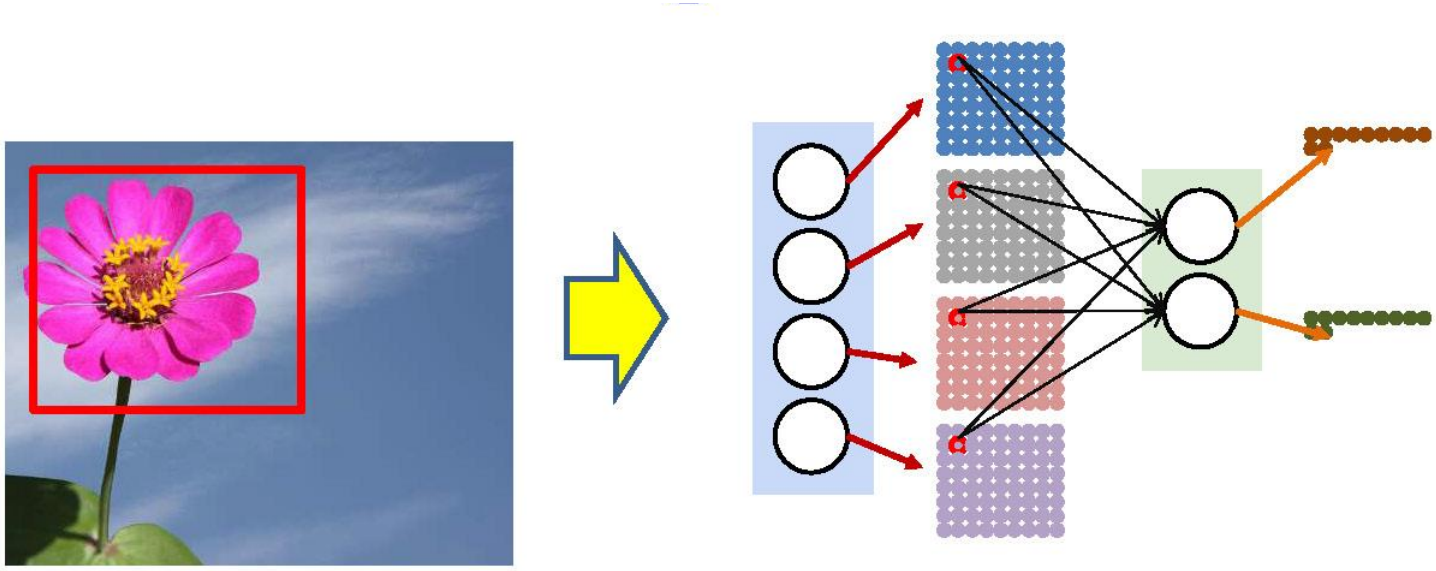
- We can recurse the logic
- – The second level neurons too are "scanning" the rectangular outputs of the first-level neurons
  - – (Un)like the first level, they are jointly scanning multiple "pictures"
- Each location in the output of the second level neuron considers the corresponding locations from the outputs of all the first-level neurons

Scanning in 2D: A closer look



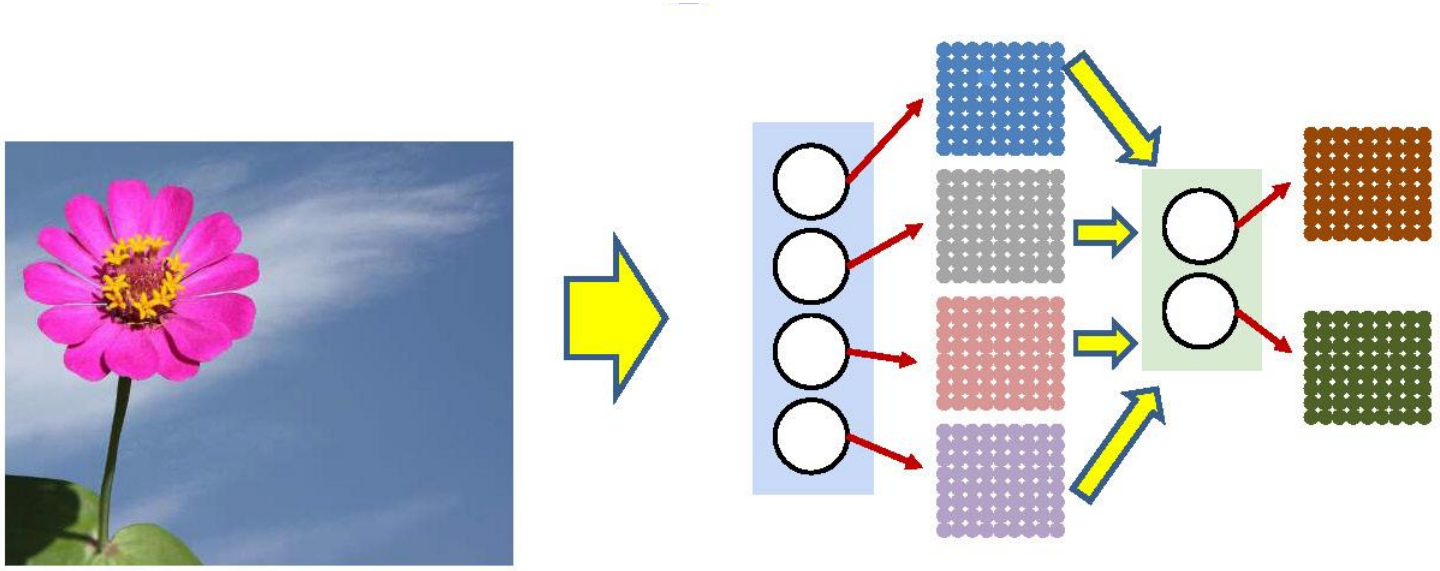
- We can recurse the logic
- – The second level neurons too are "scanning" the rectangular outputs of the first-level neurons
  - – (Un)like the first level, they are jointly scanning multiple "pictures"
- Each location in the output of the second level neuron considers the corresponding locations from the outputs of all the first-level neurons

Scanning in 2D: A closer look



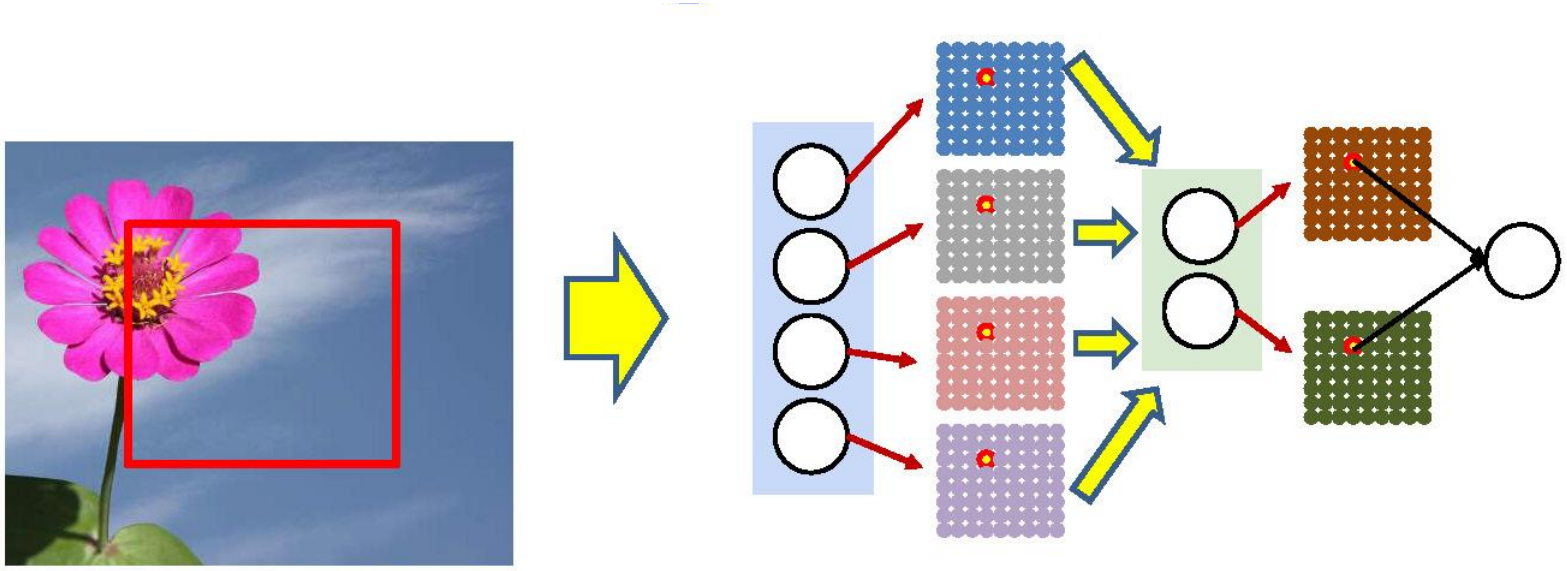
- We can recurse the logic
- – The second level neurons too are "scanning" the rectangular outputs of the first-level neurons
  - – (Un)like the first level, they are jointly scanning multiple "pictures"
- Each location in the output of the second level neuron considers the corresponding locations from the outputs of all the first-level neurons

Scanning in 2D: A closer look



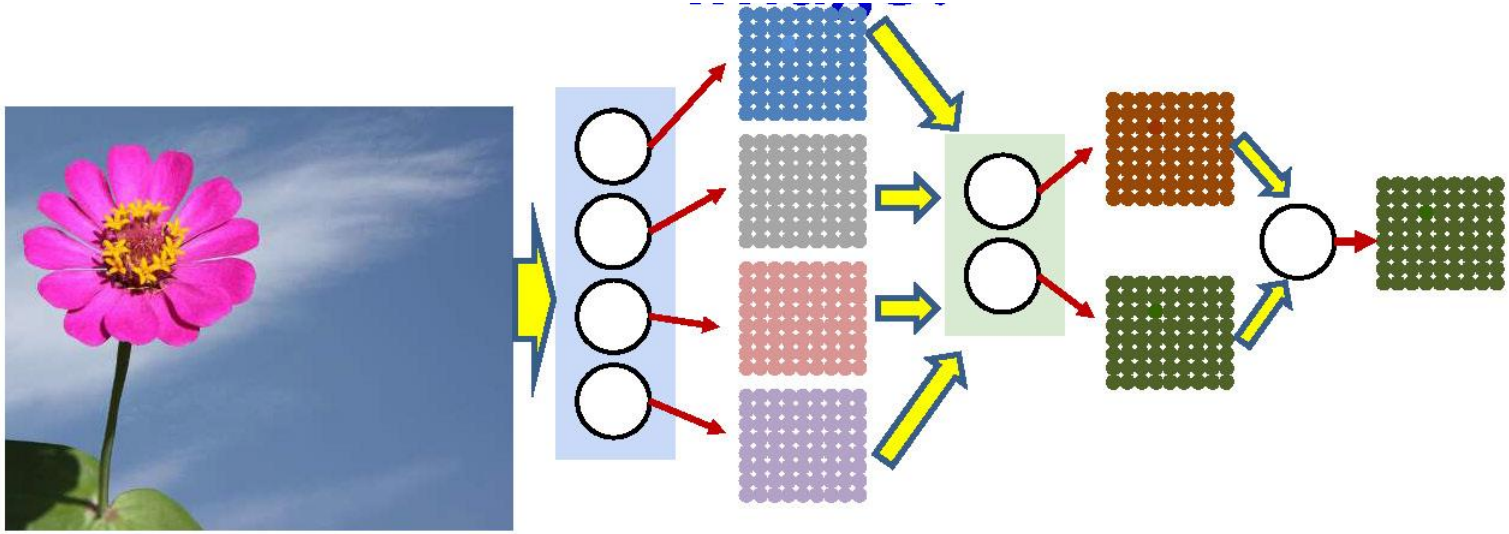
- We can recurse the logic
- – The second level neurons too are "scanning" the rectangular outputs of the first-level neurons
  - – (Un)like the first level, they are jointly scanning multiple "pictures"
- Each location in the output of the second level neuron considers the corresponding locations from the outputs of all the first-level neurons

Scanning in 2D: A closer look



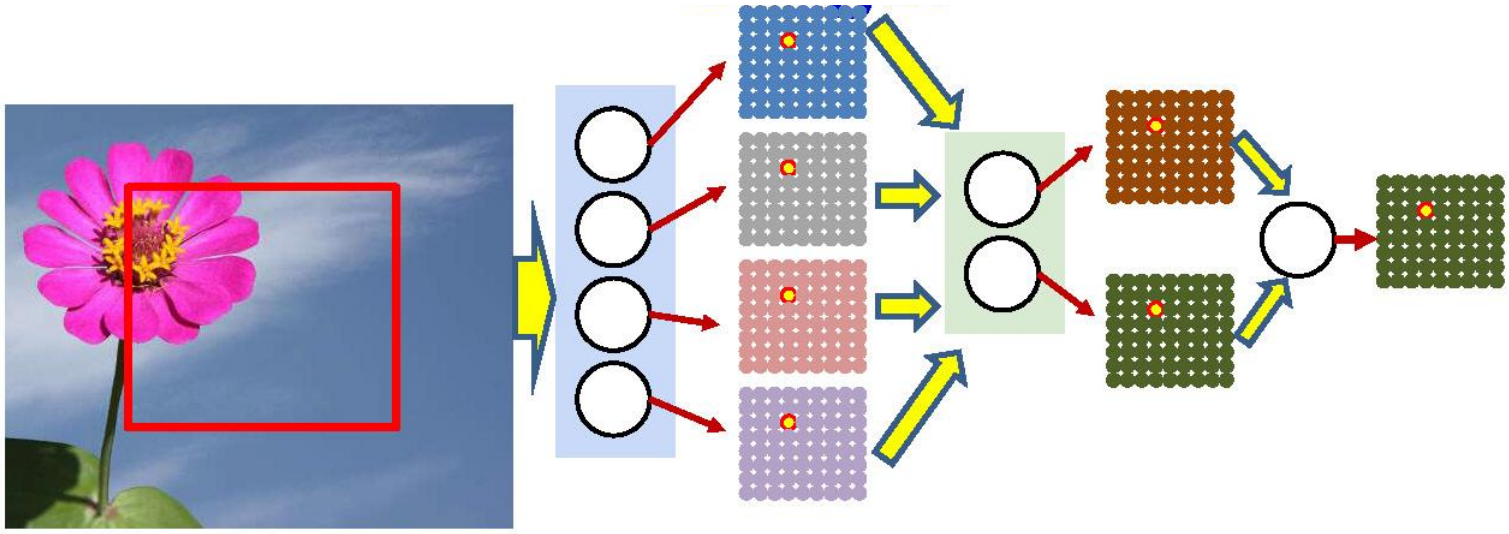
- To detect a picture **at any location** in the original image, the output layer must consider the corresponding outputs of the last hidden layer

Detecting a picture anywhere in the image?



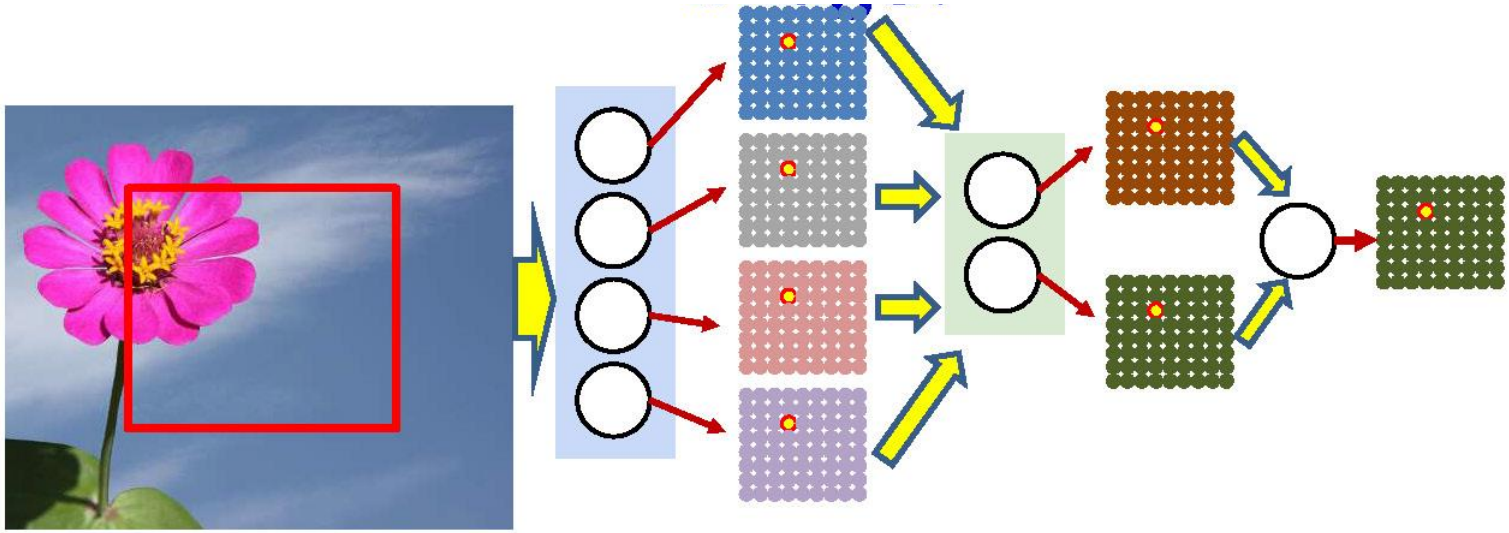
- Recursing the logic, we can create a map for the neurons in the next layer as well
  - The map is a flower detector for each location of the original image

Detecting a picture anywhere in the image?



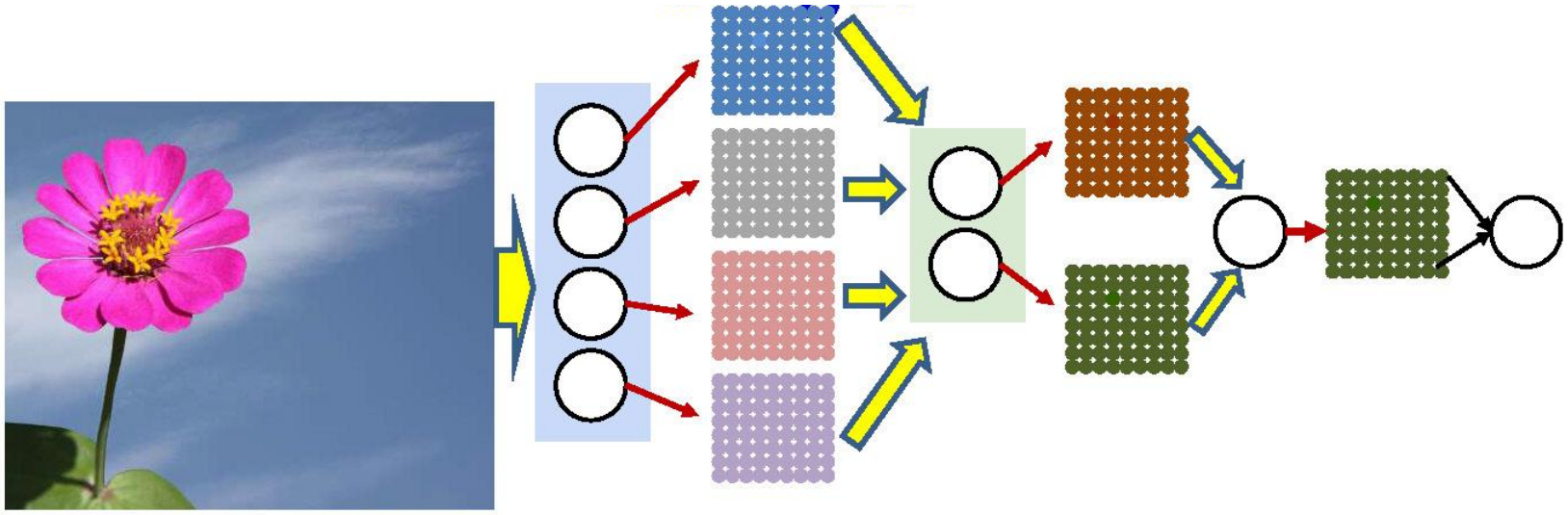
- To detect a picture **at any location** in the original image, the output layer must consider the corresponding outputs of the last hidden layer

Detecting a picture anywhere in the image?



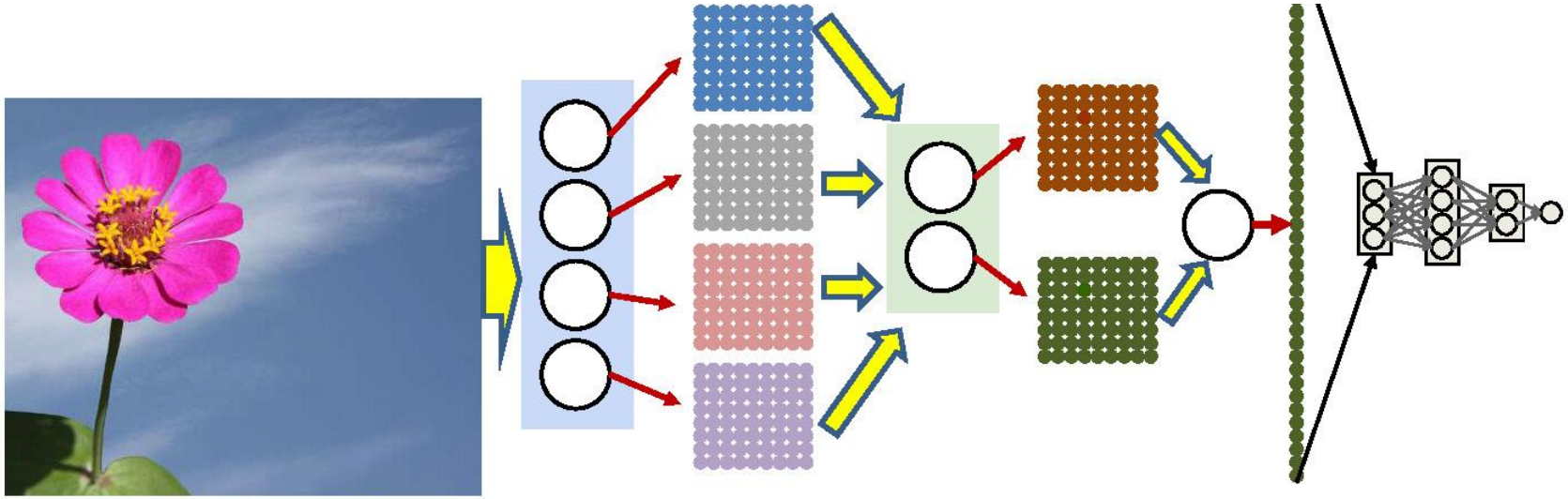
- To detect a picture **at any location** in the original image, the output layer must consider the corresponding outputs of the last hidden layer
- Actual problem? Is there a flower in the image
  - Not "detect the location of a flower"

Detecting a picture anywhere in the image?



- Is there a flower in the picture?
  - The output of the almost-last layer is also a grid/picture
  - The entire grid can be sent into a final neuron that performs a logical "OR" to detect a flower in the full picture
  - Finds the max output from all the positions
  - Or a softmax, or a full MLP.

Detecting a picture anywhere in the image?

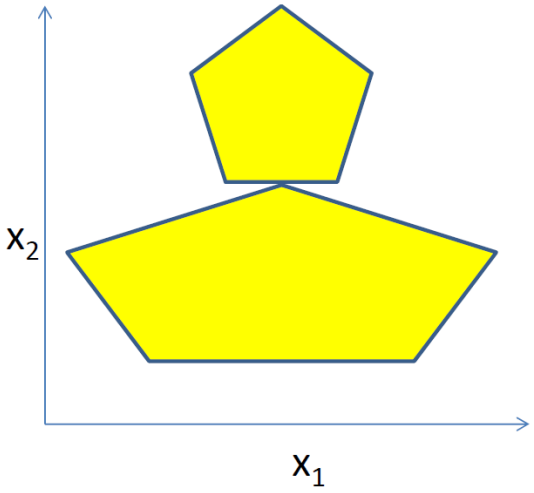


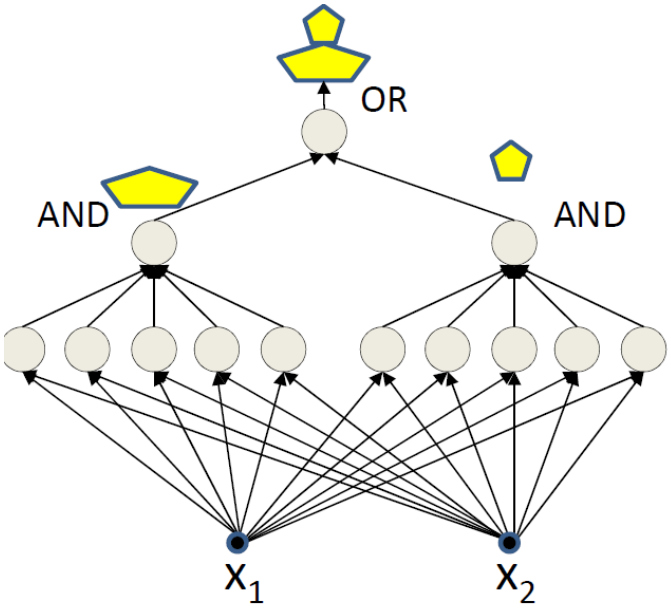
- Redrawing the final layer

  – "Flatten" the output of the neurons into a single block, since the arrangement is no longer important

  – Pass that through a max/softmax/MLP"

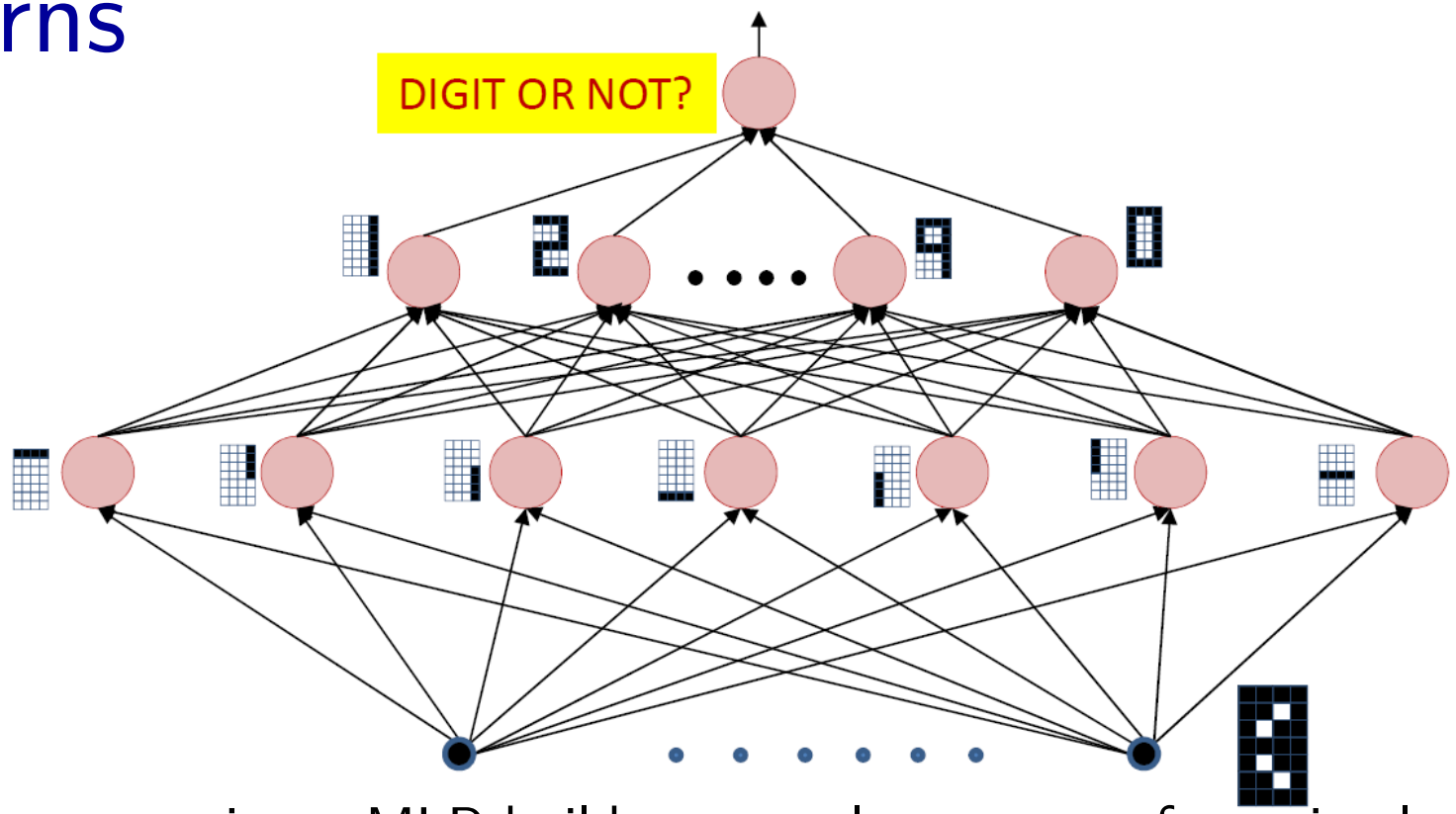
Recall: What does an MLP learn?





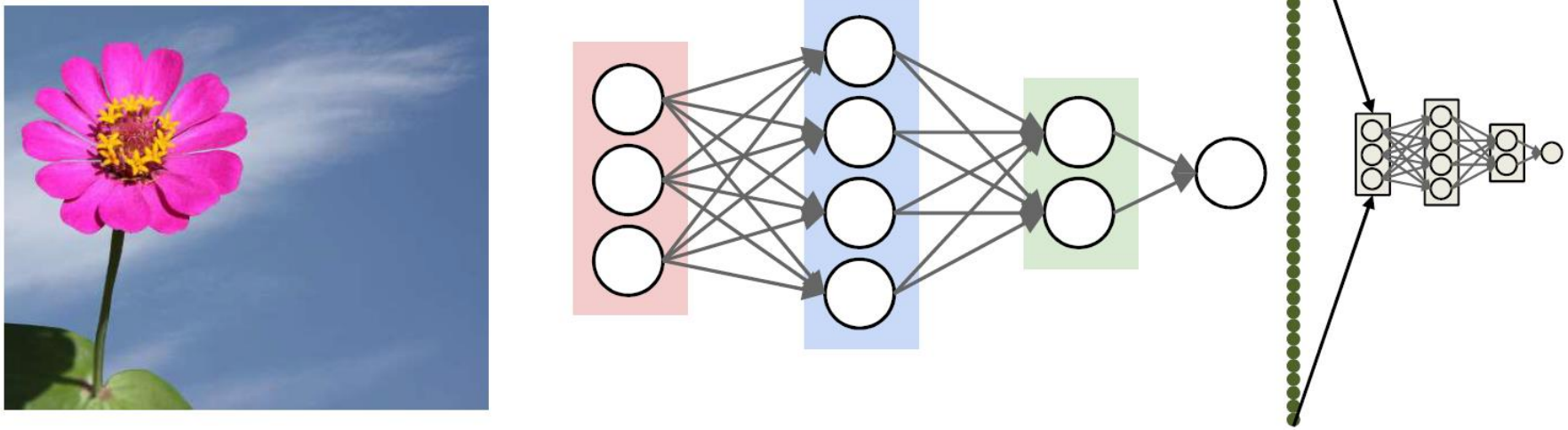
- The lowest layers of the network capture simple patterns
  - The linear decision boundaries in this example
- The next layer captures more complex patterns
  - The polygons
- The next one captures still more complex patterns

Recall: How does an MLP represent patterns



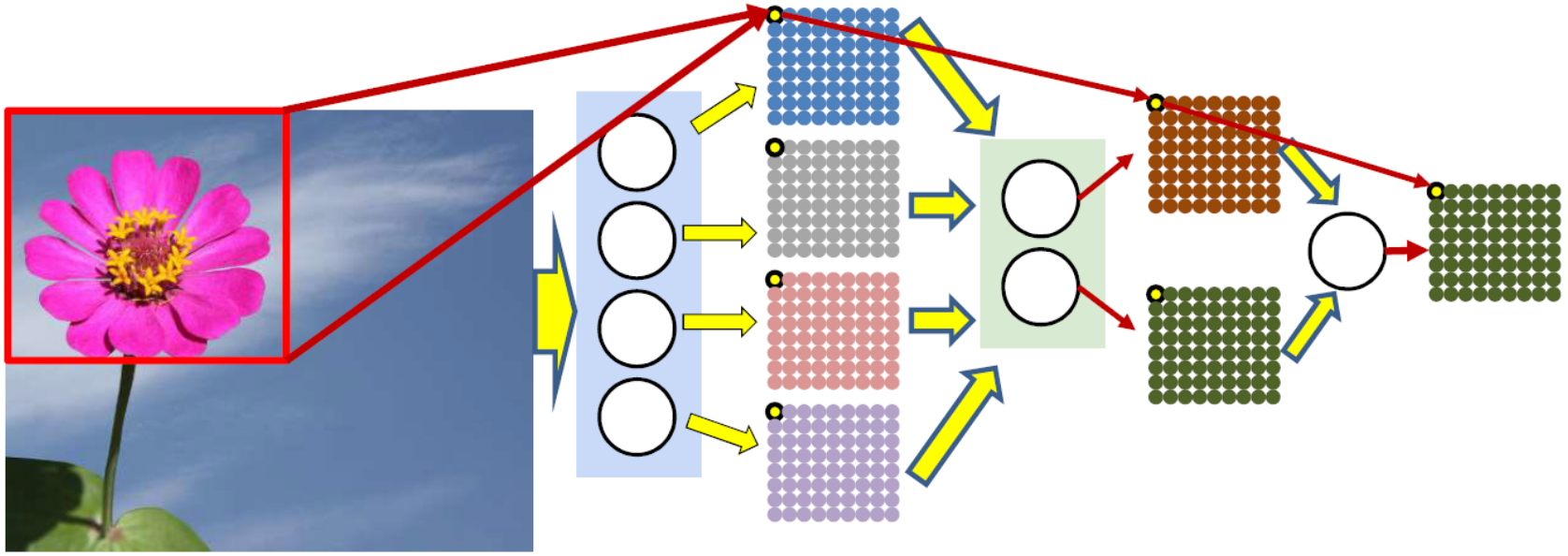
- The neurons in an MLP build up complex patterns from simple pattern hierarchically
  - Each layer learns to "detect" simple combinations of the patterns detected by earlier layers

What does our network learn?



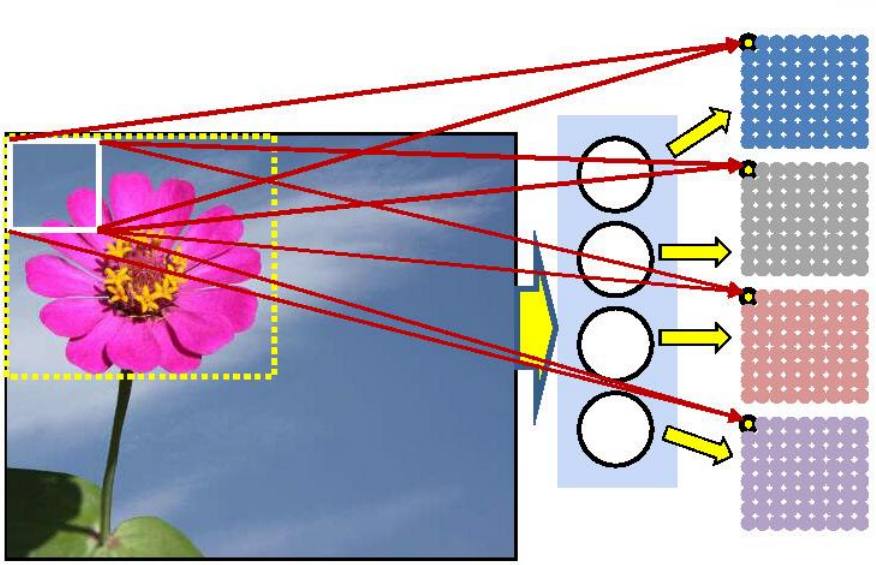
- The entire MLP looks for a flower-like pattern at each location

The behavior of the layers



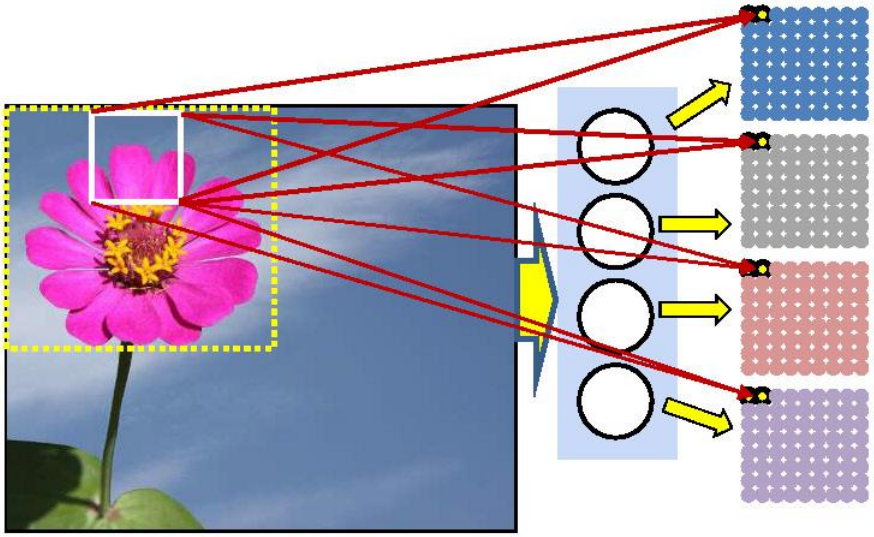
- The first layer neurons "look" at the entire "block" to extract block-level features
  - Subsequent layers only perform classification over these block-level features
- The first layer neurons is responsible for evaluating the entire block of **pixels**
  - Subsequent layers only look at a single pixel in their input maps

Distributing the scan



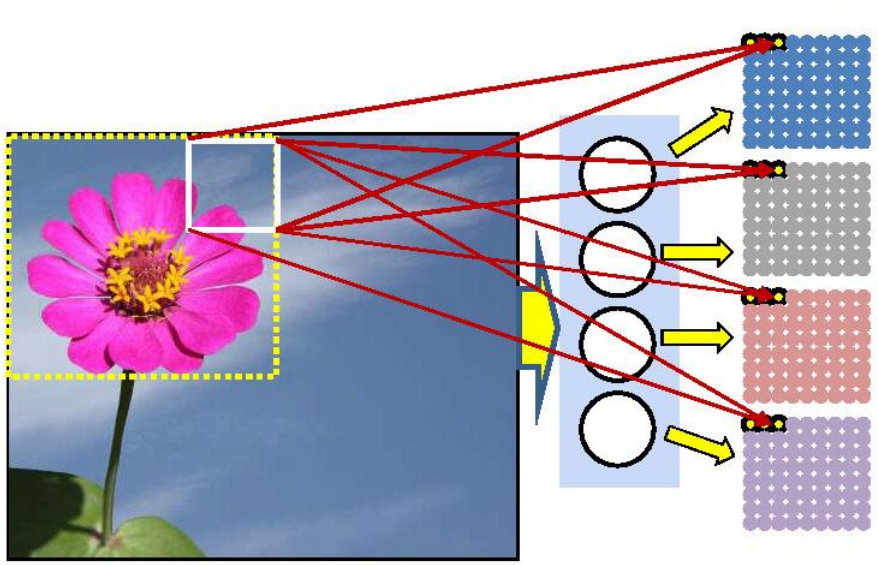
- We can distribute the pattern matching over two layers and still achieve the same block analysis at the second layer
  - The first layer evaluates smaller blocks of pixels

Distributing the scan



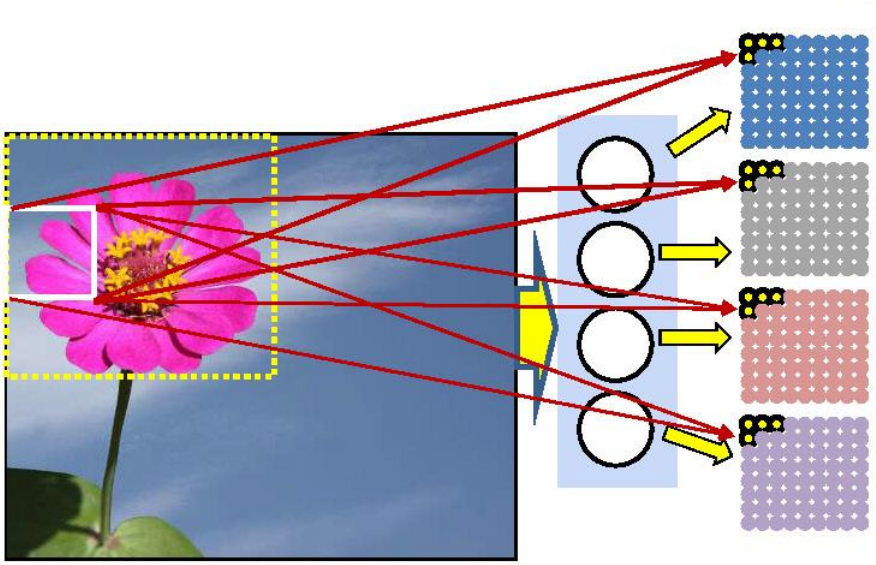
- We can distribute the pattern matching over two layers and still achieve the same block analysis at the second layer
  - The first layer evaluates smaller blocks of pixels

Distributing the scan



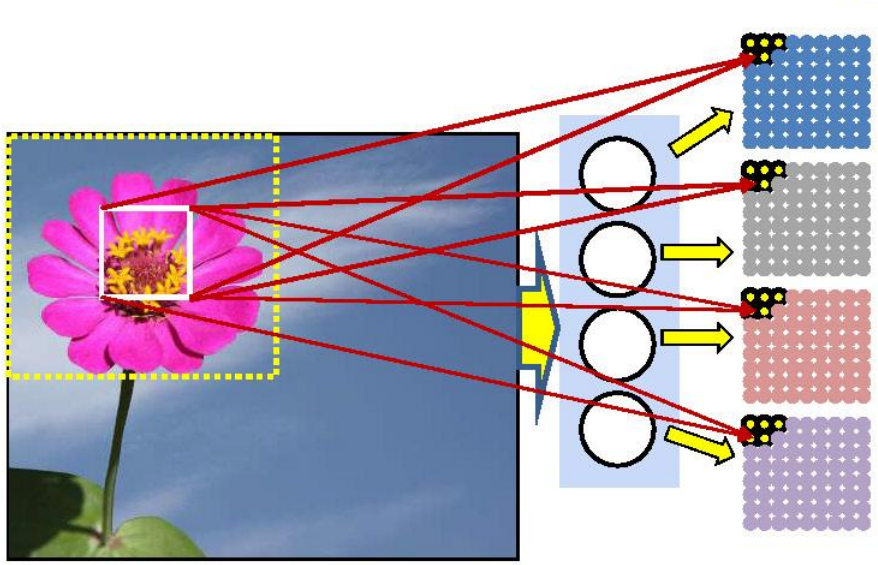
- We can distribute the pattern matching over two layers and still achieve the same block analysis at the second layer
  - The first layer evaluates smaller blocks of pixels

Distributing the scan



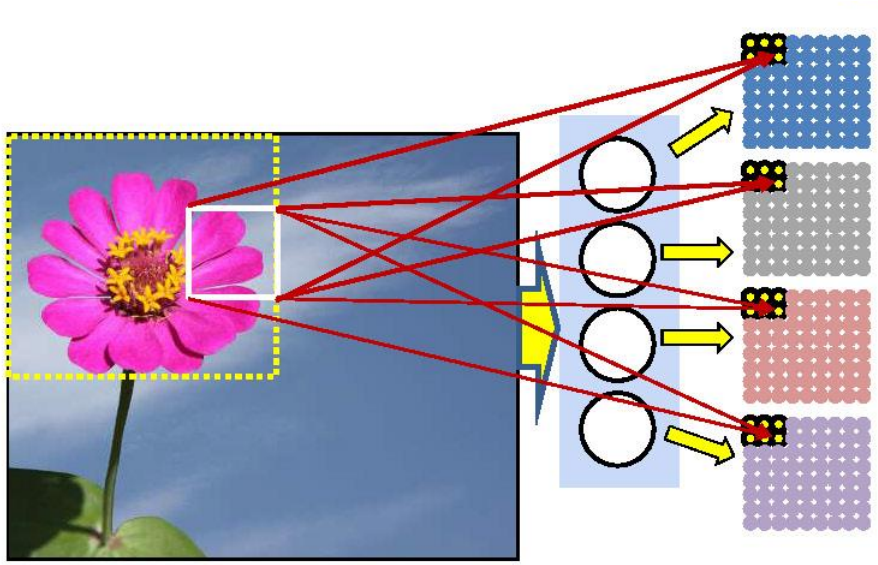
- We can distribute the pattern matching over two layers and still achieve the same block analysis at the second layer
  - The first layer evaluates smaller blocks of pixels

Distributing the scan



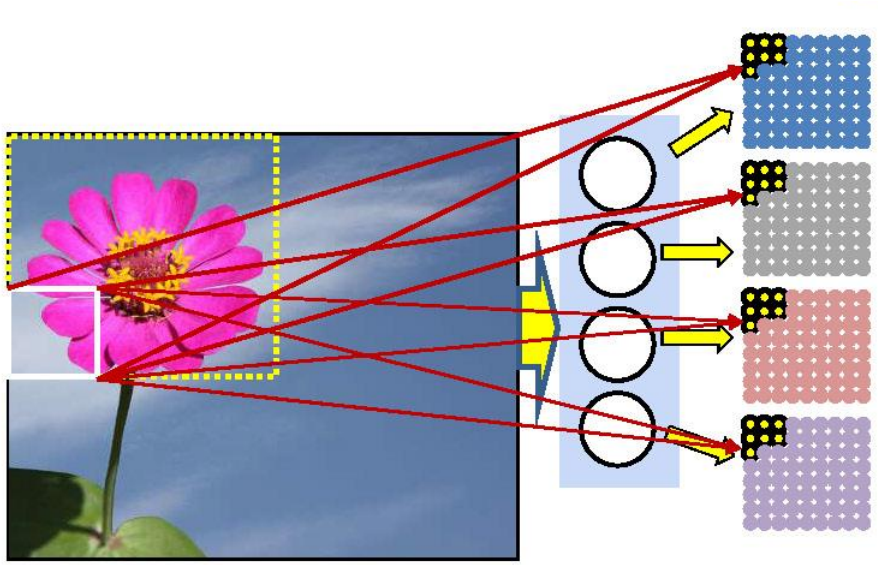
- We can distribute the pattern matching over two layers and still achieve the same block analysis at the second layer
  - The first layer evaluates smaller blocks of pixels

Distributing the scan



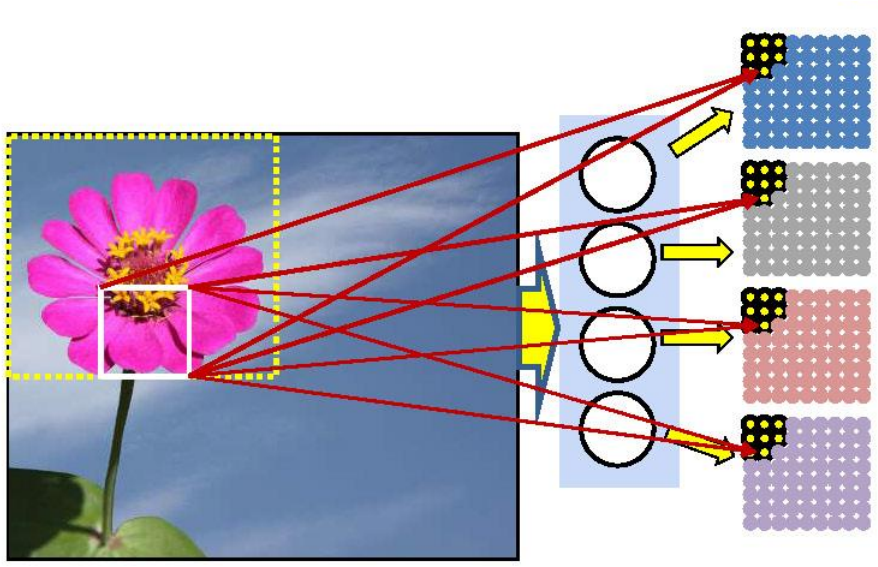
- We can distribute the pattern matching over two layers and still achieve the same block analysis at the second layer
  - The first layer evaluates smaller blocks of pixels

Distributing the scan



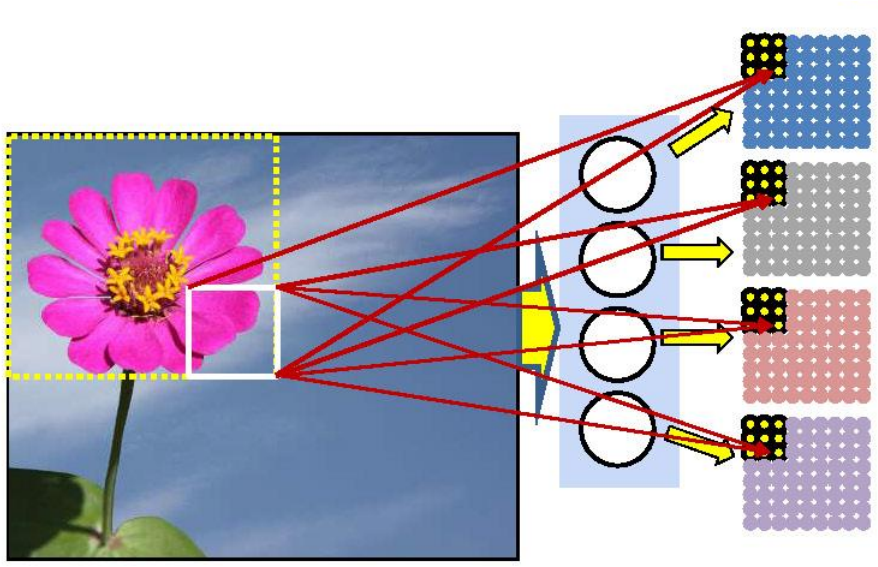
- We can distribute the pattern matching over two layers and still achieve the same block analysis at the second layer
  - The first layer evaluates smaller blocks of pixels

Distributing the scan



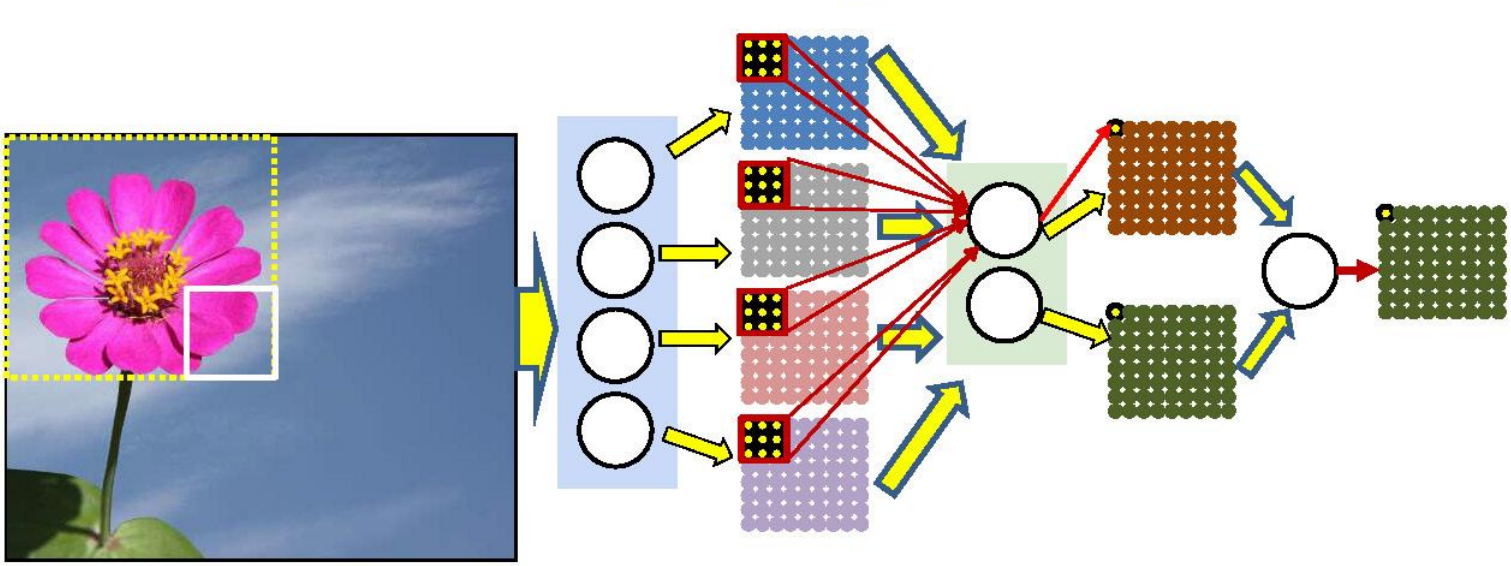
- We can distribute the pattern matching over two layers and still achieve the same block analysis at the second layer
  - The first layer evaluates smaller blocks of pixels

Distributing the scan



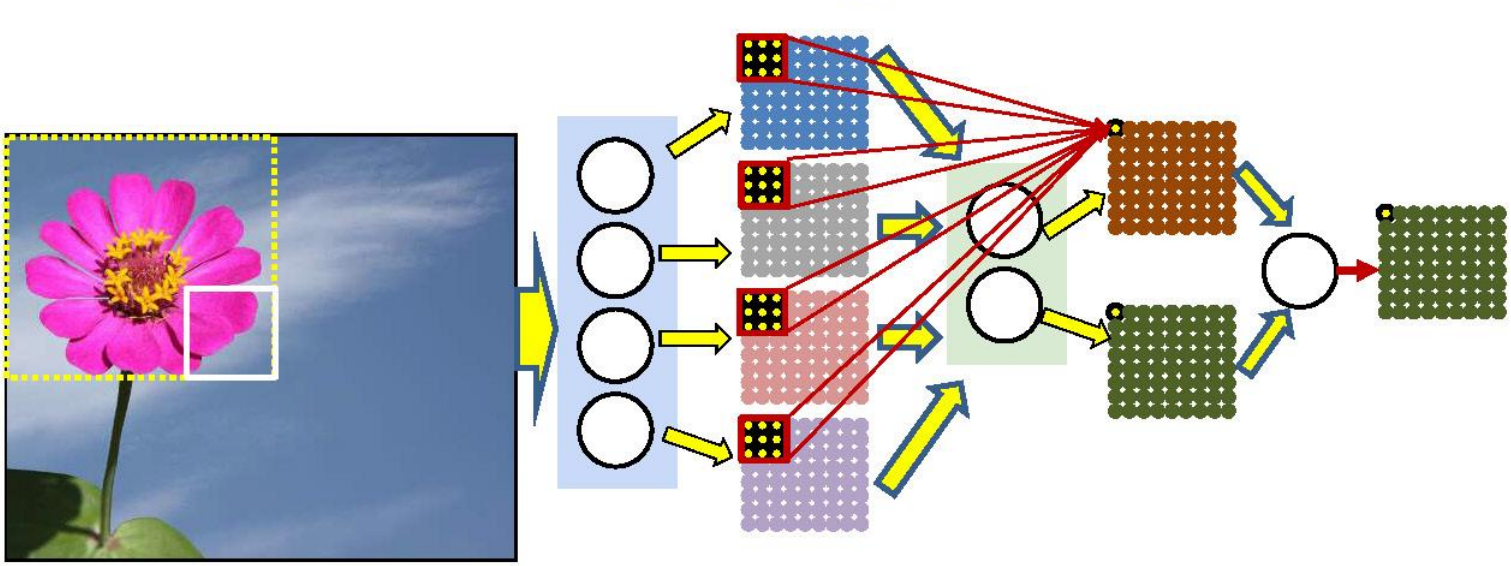
- We can distribute the pattern matching over two layers and still achieve the same block analysis at the second layer
  - The first layer evaluates smaller blocks of pixels

Distributing the scan



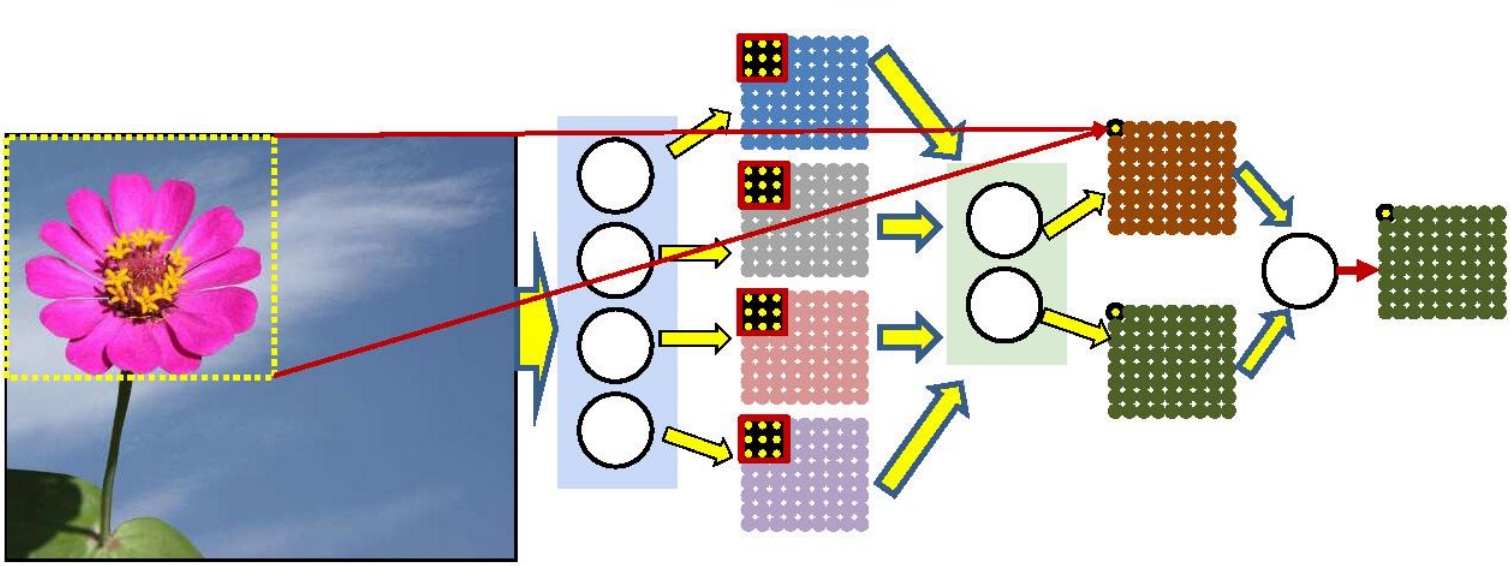
- We can distribute the pattern matching over two layers and still achieve the same block analysis at the second layer
  - The first layer evaluates smaller blocks of pixels
  - The next layer evaluates blocks of outputs from the first layer

Distributing the scan



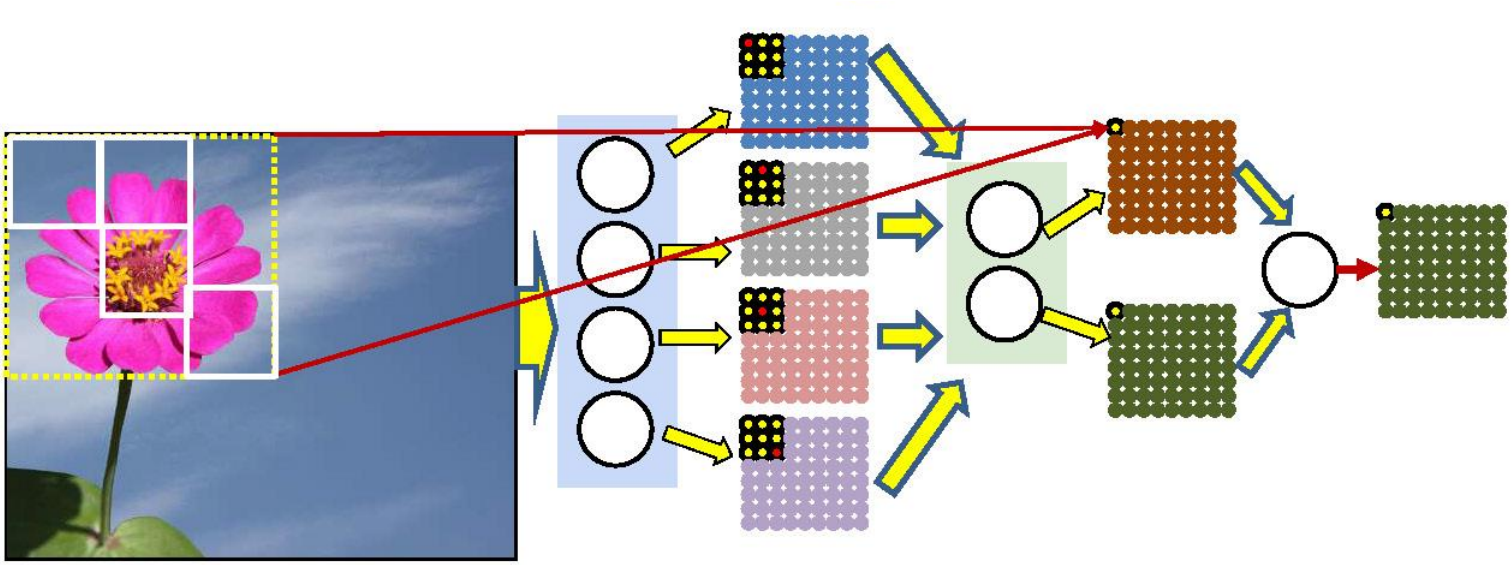
- We can distribute the pattern matching over two layers and still achieve the same block analysis at the second layer
  - The first layer evaluates smaller blocks of pixels
  - The next layer evaluates blocks of outputs from the first layer

Distributing the scan



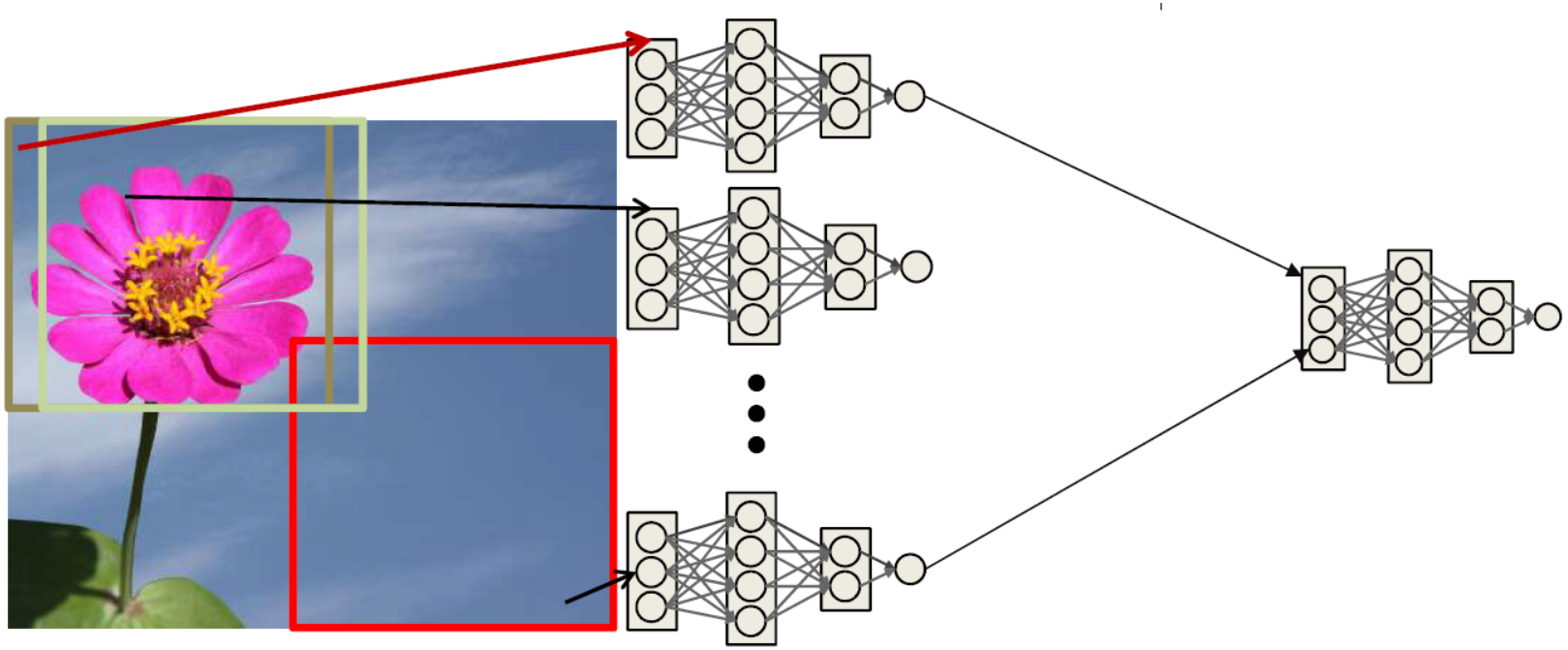
- We can distribute the pattern matching over two layers and still achieve the same block analysis at the second layer
  - The first layer evaluates smaller blocks of pixels
  - The next layer evaluates blocks of outputs from the first layer
  - This effectively evaluates the larger block of the original image

Distributing the scan



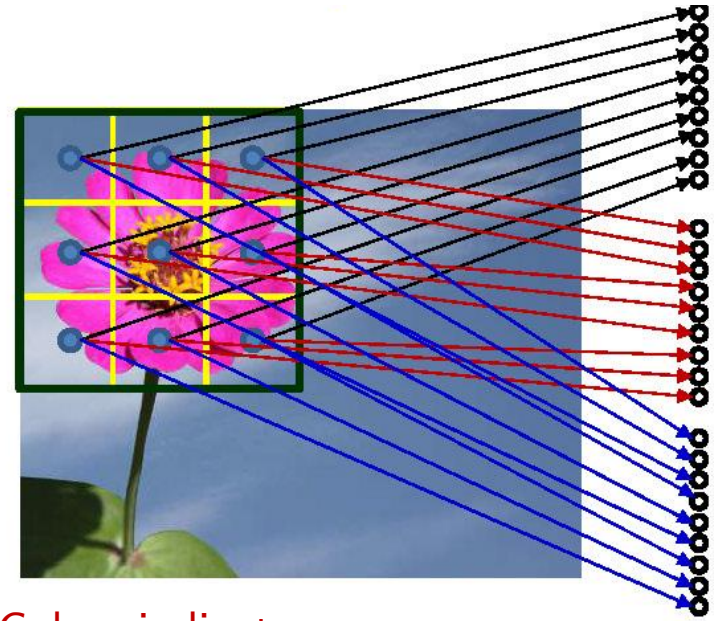
- The higher layer implicitly learns the arrangement of sub patterns that represents the larger pattern (the flower in this case)

This is still just scanning with a shared parameter network



- With a minor modification...

This is still just scanning with a shared parameter network

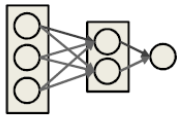


Colors indicate neurons with shared parameters

Layer 1

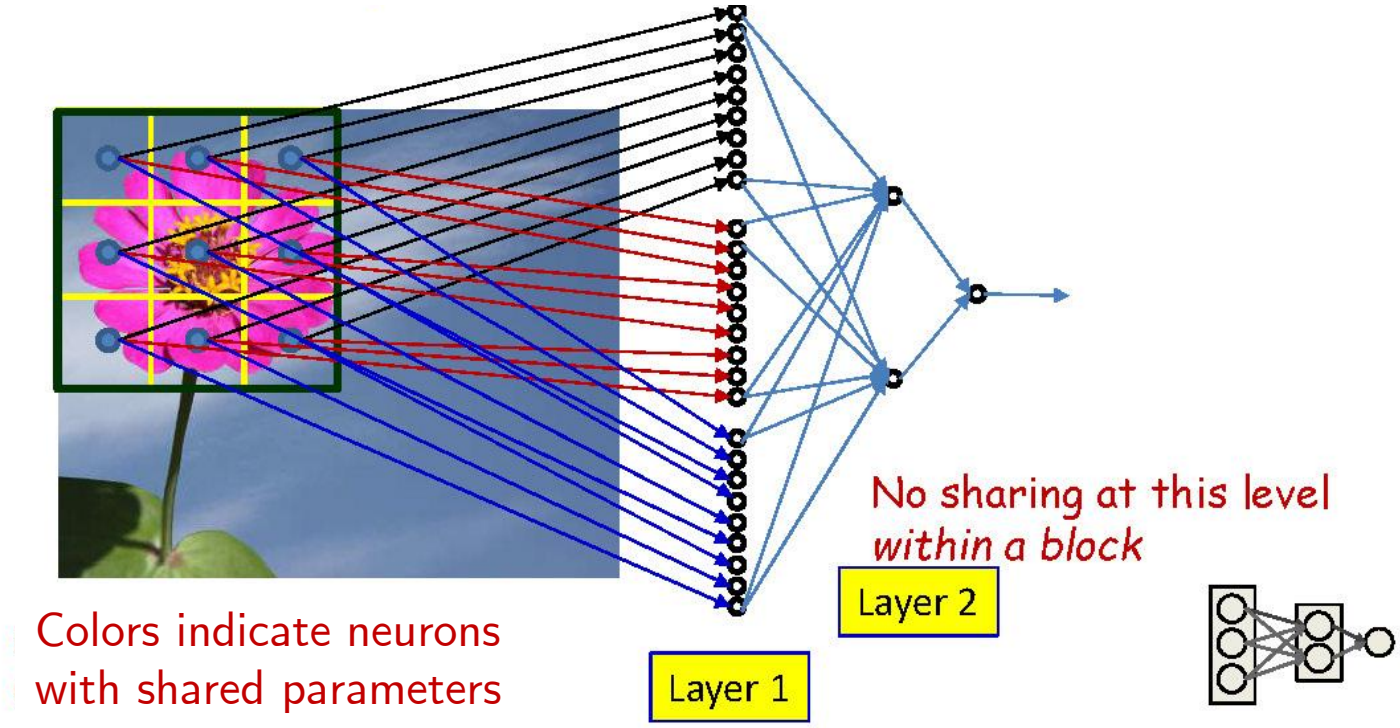
Each arrow represents an entire set of weights over the smaller cell

The pattern of weights going out of any cell is identical to that from any other cell.



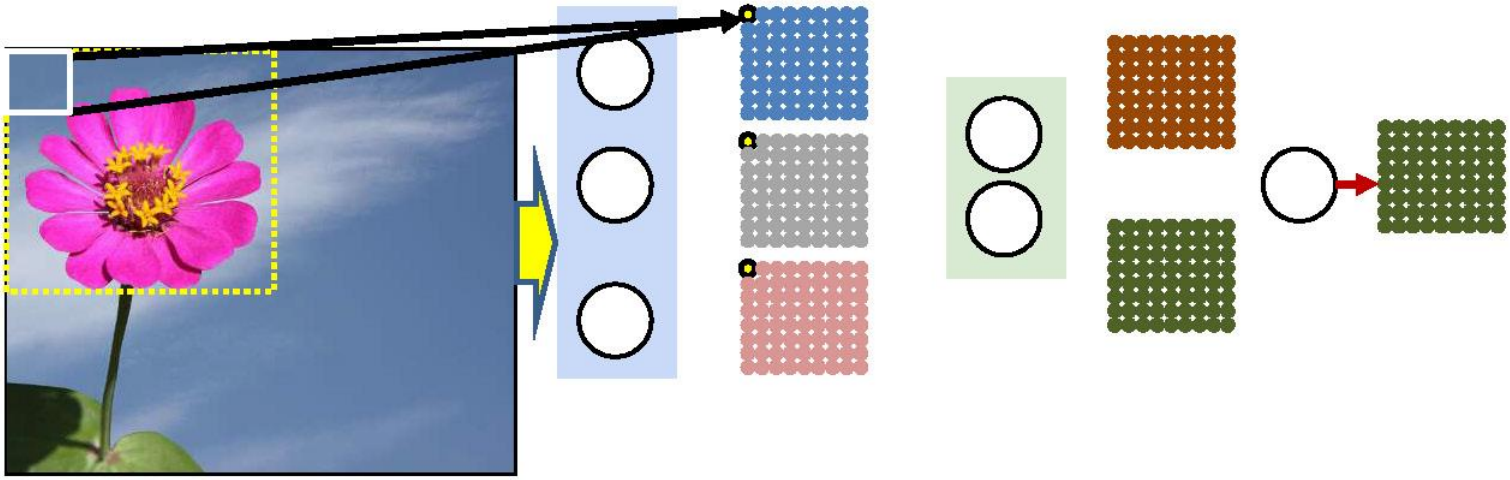
- The network that analyzes individual blocks is now itself a shared parameter network..

This is still just scanning with a shared parameter network



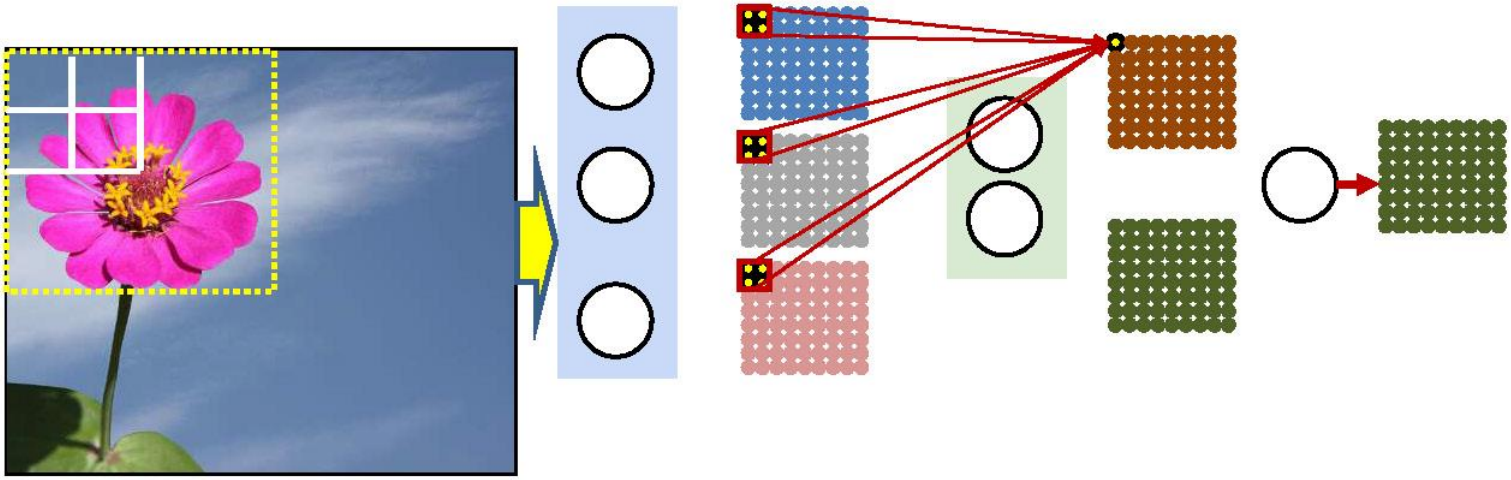
- The network that analyzes individual blocks is now itself a shared parameter network..

This logic can be recursed



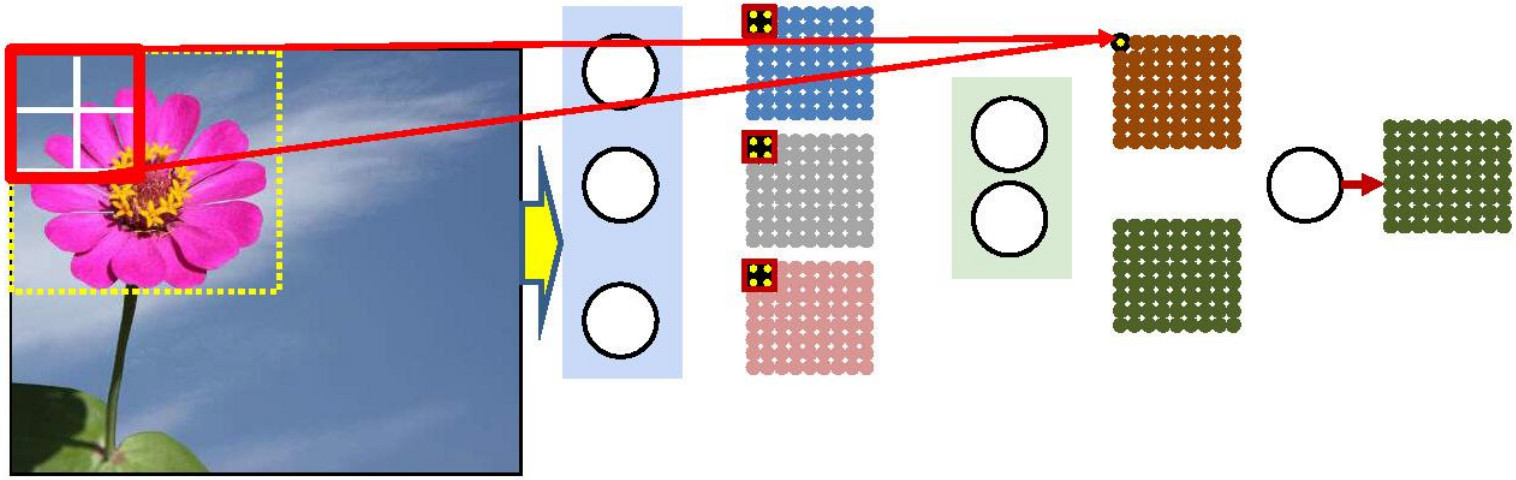
- Building the pattern over 3 layers

This logic can be recursed



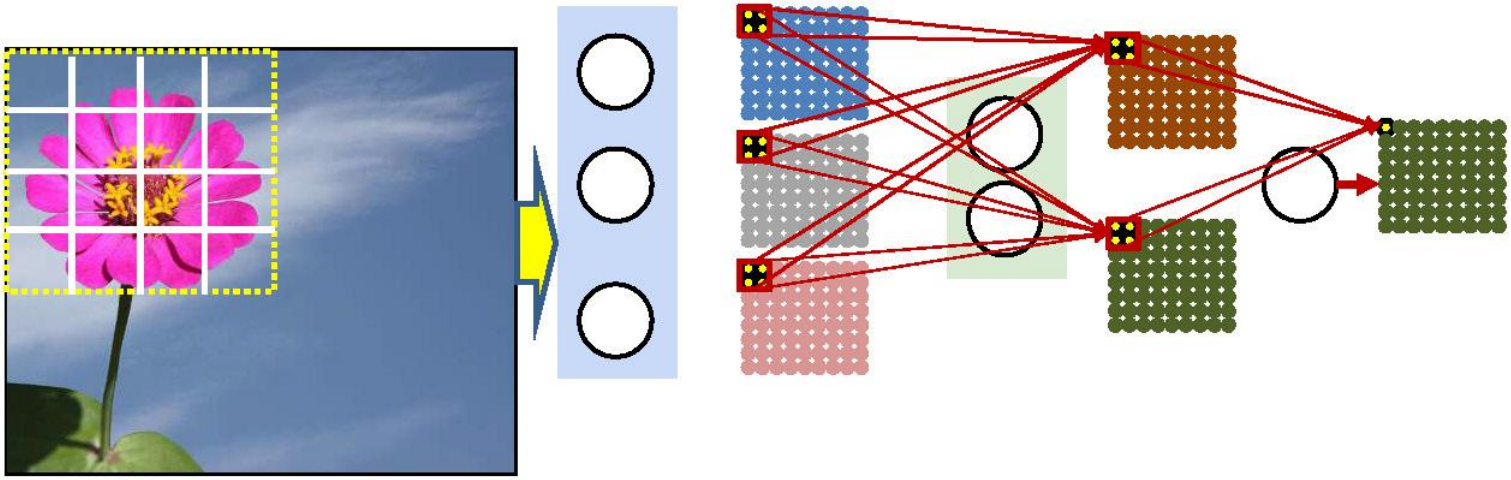
- Building the pattern over 3 layers

This logic can be recursed



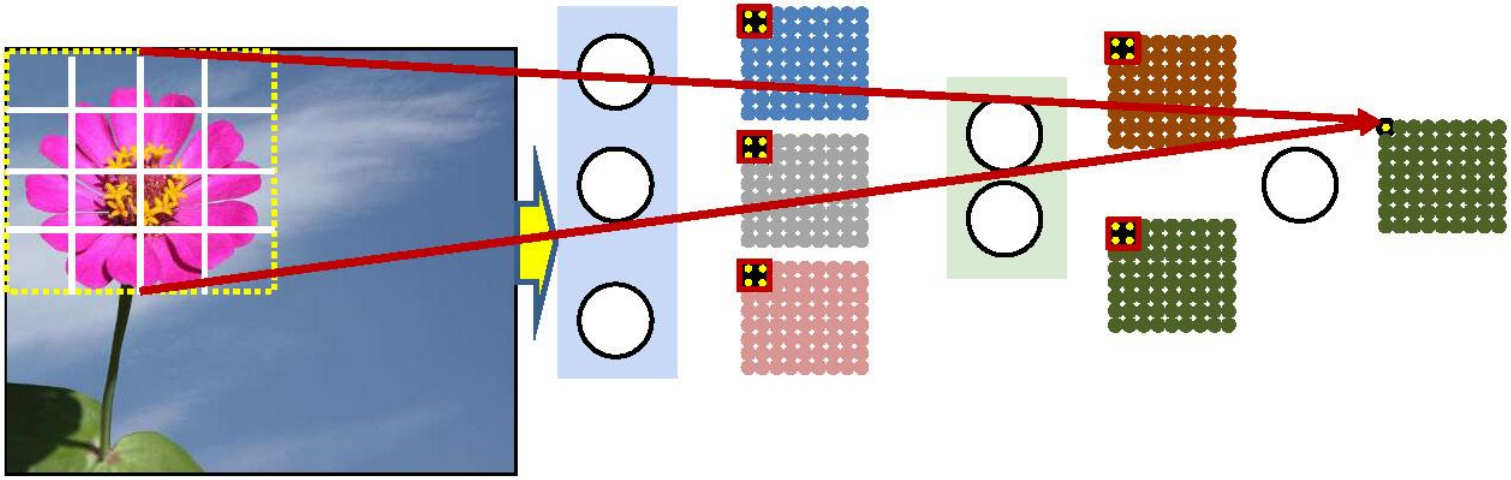
- Building the pattern over 3 layers

This logic can be recursed



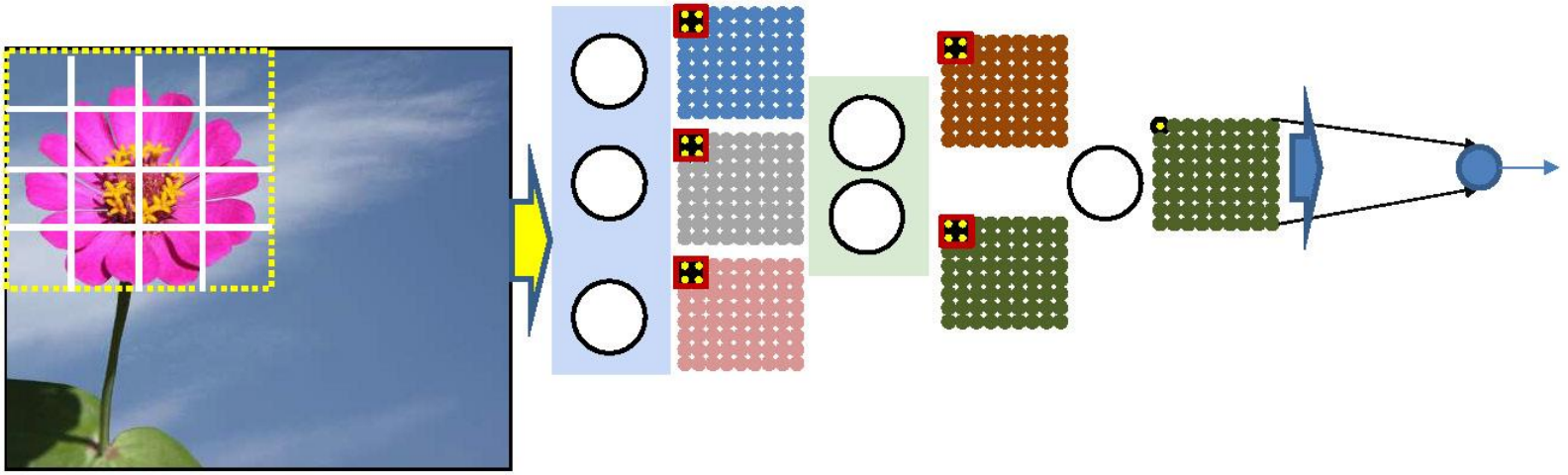
- Building the pattern over 3 layers

This logic can be recursed



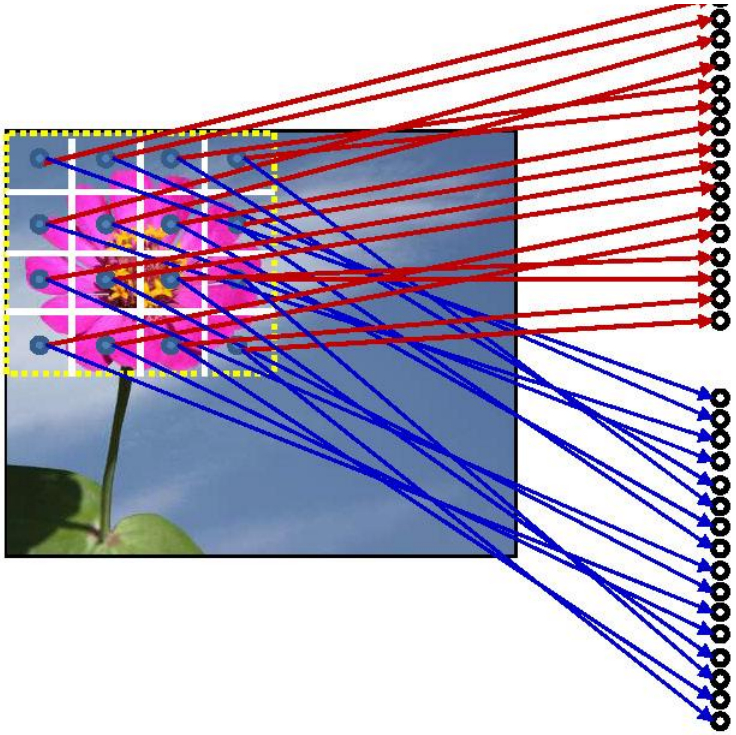
- Building the pattern over 3 layers

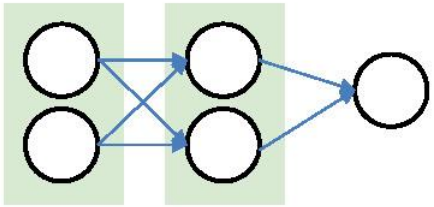
Does the picture have a flower



- Building the pattern over 3 layers
- The final classification for the entire image views the outputs from all locations, as seen in the final map

The 3-layer shared parameter net

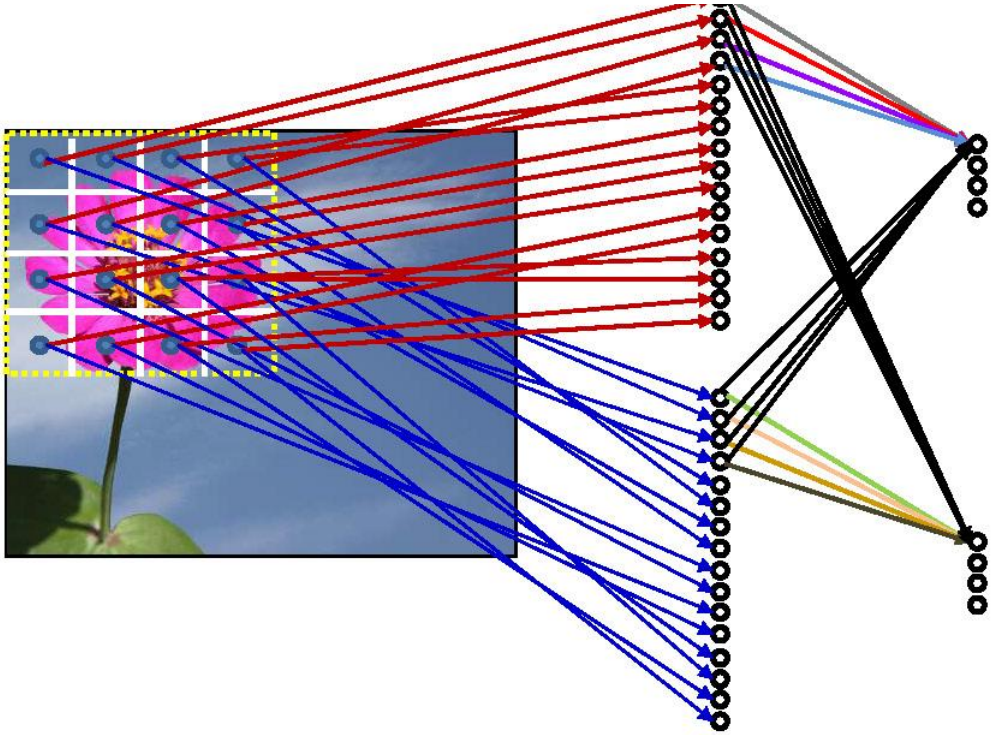




Showing a simpler 2x2x1 network to fit on the slide

- Building the pattern over 3 layers

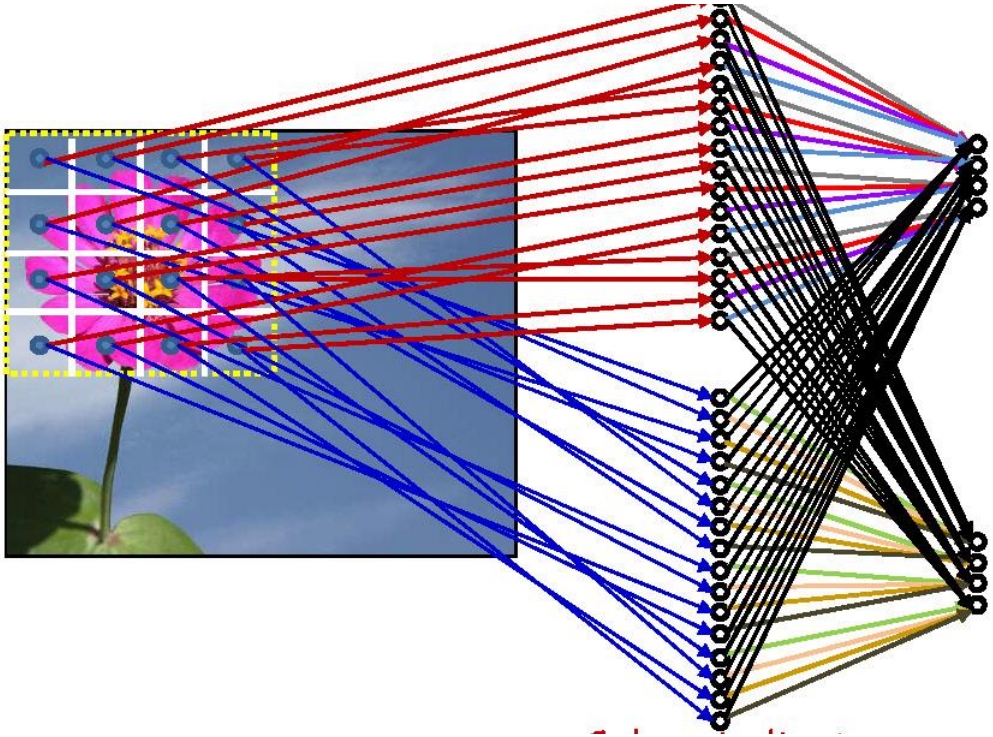
The 3-layer shared parameter net



All weights shown are unique

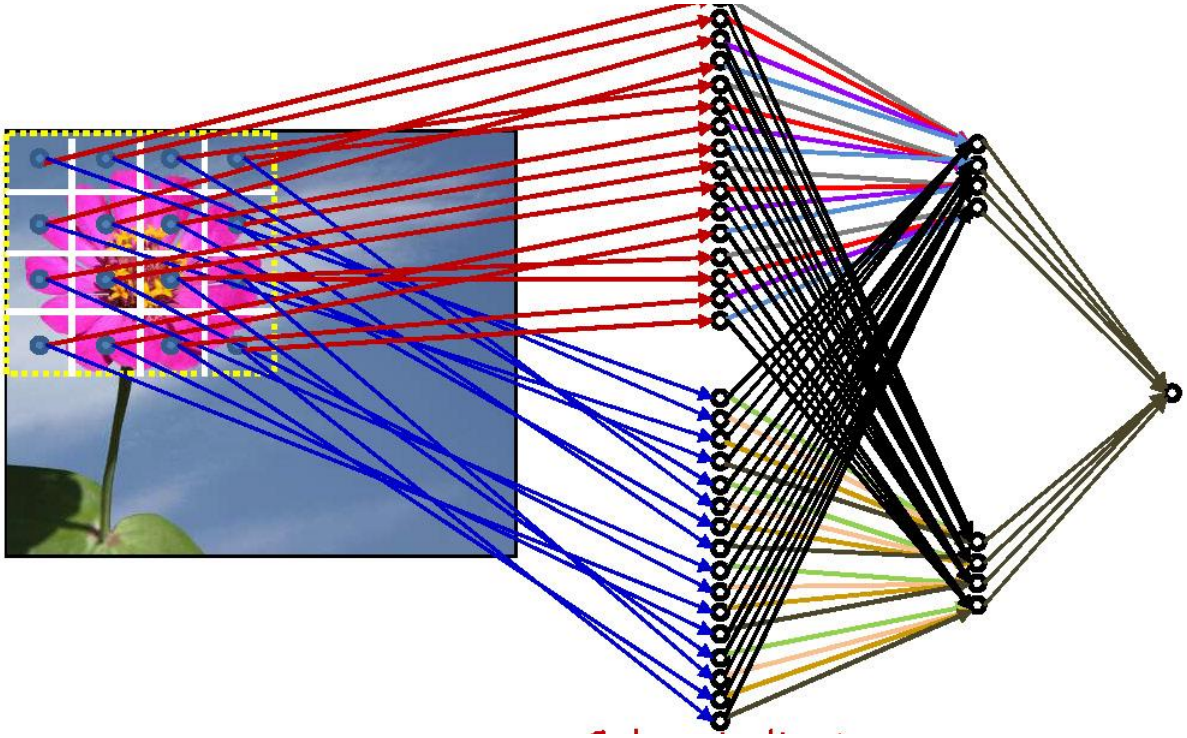
- Building the pattern over 3 layers

The 3-layer shared parameter net



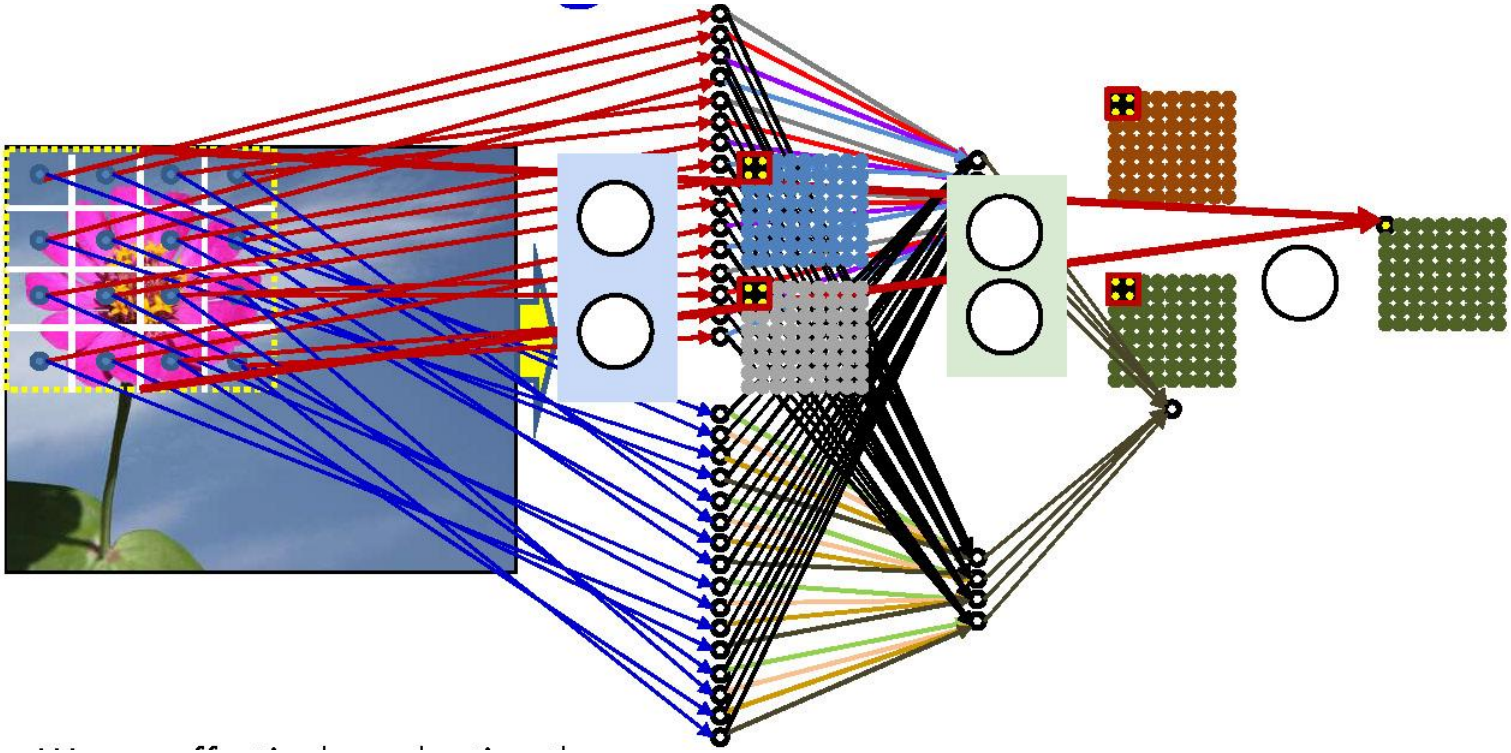
- Building the pattern over 3 layers
- Colors indicate shared parameters

The 3-layer shared parameter net



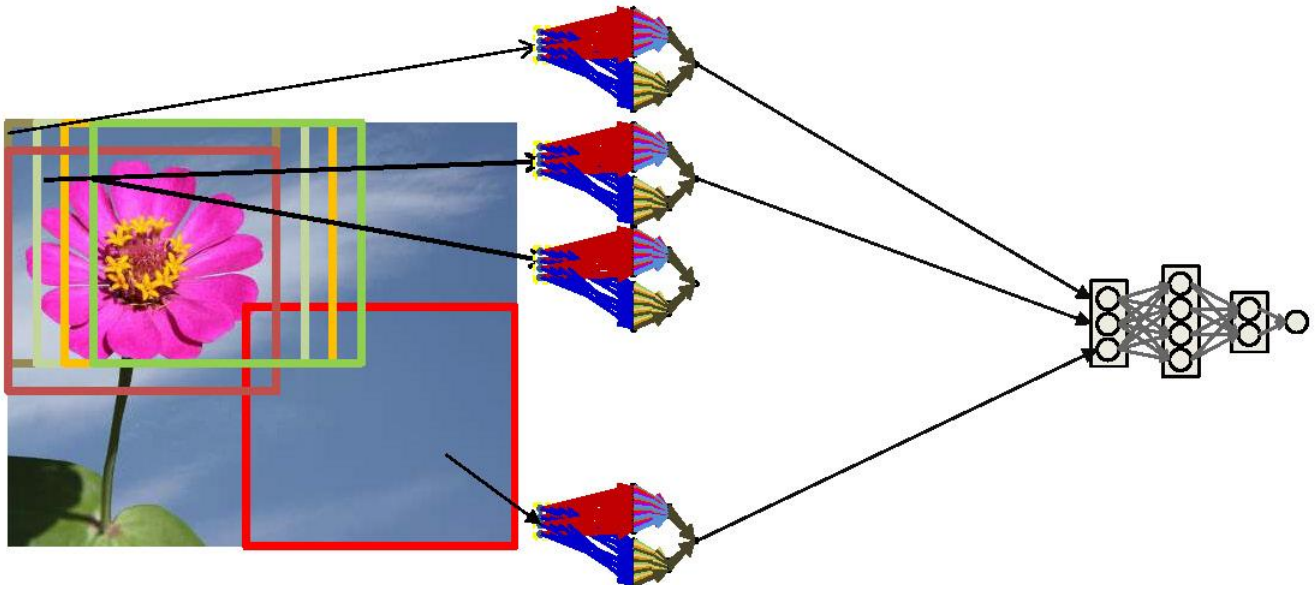
- Building the pattern over 3 layers
- Colors indicate shared parameters

This logic can be recursed



- We are effectively evaluating the yellow block with the shared parameter net to the right
- Every block is evaluated using the same net in the overall computation

This logic can be recursed



- The individual blocks are now themselves shared-parameter networks
- We scan the figure using the shared parameter network
  - The entire operation can be viewed as a single giant network
  - Where individual subnets are themselves shared-parameter nets

Scanning with an MLP

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

```
for x = 1:W-K+1
    for y = 1:H-K+1
        ImgSegment = Img(*, x:x+W-1, y:y+W-1)
        Y(x,y) = MLP(ImgSegment)
Y = softmax( Y(1,1)..Y(W-K+1,H-K+1) )
```

Scanning with an MLP

```
for x = 1:W-K+1
   for y = 1:H-K+1
        for l = 1:L   # layers
            for j = 1:D_l
                Compute z(l,j,x,y) [not expanded]
                Y(l,j,x,y) = activation(z(l,j,x,y))

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

Reordering the computation

```
for l = 1:L  # layers
    for j = 1:D_l
        for x = 1:W-K+1
            for y = 1:H-K+1
                Compute z(l,j,x,y) [not expanded]
                Y(l,j,x,y) = activation(z(l,j,x,y))

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

Reordering the computation

```
for l = 1:L   # layers
    for j = 1:D_l
        for x = 1:W_{l-1}-K_l+1
            for y = 1:H_{l-1}-K_l+1
                Compute z(l,j,x,y) [not expanded]
                Y(l,j,x,y) = activation(z(l,j,x,y))

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

Each layer's map is now a different size: Maps progressively by K_l in each layer

Reordering the computation

```
Y(0,:,:,:) = Image
for l = 1:L  # layers operate on vector at (x,y)
   for j = 1:D_l
      for x = 1:W_{l-1}-K_l+1
         for y = 1:H_{l-1}-K_l+1
            z(l,j,x,y) = 0
            for i = 1:D_{l-1}
               for x' = 1:K_l
                  for y' = 1:K_l
                     z(l,j,x,y) += w(l,i,j,x',y')
                                   Y(l-1,i,x+x'-1,y+y'-1)
            Y(l,j,x,y) = activation(z(l,j,x,y))

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

Reordering the computation

```
Y(0,:,:,:) = Image
for l = 1:L  # layers operate on vector at (x,y)
    for j =
        for x = 1:W_{l-1}-K_l+1
            for y = 1:H_{l-1}-K_l+1
                z(l,j,x,y) = 0
                for i = 1:D_{l-1}
                    for x' = 1:K_l
                        for y' = 1:K_l
                            z(l,j,x,y) += w(l,i,j,x',y')
                                          Y(l-1,i,x+x'-1,y+y'-1)
            Y(l,j,x,y) = activation(z(l,j,x,y))

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

This operation is a "convolution"

"Convolutional Neural Network" (aka scanning with an MLP)

```
Y(0,:,:,:) = Image
for l = 1:L  # layers operate on vector at (x,y)
   for j = 1:D_l
      for x = 1:W_{l-1}-K_l+1
         for y = 1:H_{l-1}-K_l+1
            z(l,j,x,y) = 0
            for i = 1:D_{l-1}
               for x' = 1:K_l
                  for y' = 1:K_l
                     z(l,j,x,y) += w(l,i,j,x',y')
                                   Y(l-1,i,x+x'-1,y+y'-1)
            Y(l,j,x,y) = activation(z(l,j,x,y))

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

Convolutional neural net: Vector notation

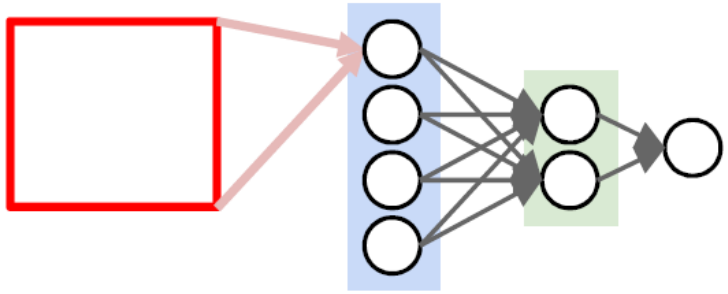
The weight W(l,j) is now a 3D $D_{l-1} \times K_l \times K_l$ tensor (assuming square receptive fields)

The product in blue is a tensor inner product with a scalar output

```
Y(0) = Image
for l = 1:L  # layers operate on vector at (x,y)
  for j = 1:D_l
    for x = 1:W_{l-1}-K_l+1
       for y = 1:H_{l-1}-K_l+1
          segment = Y(l-1,:,x:x+K_l-1,y:y+K_l-1) #3D tensor
          z(l,j,x,y) = W(l,j).segment #tensor inner prod.
          Y(l,j,x,y) = activation(z(l,j,x,y))

Y = softmax( Y(L) )
```

Comparing Number of Parameters

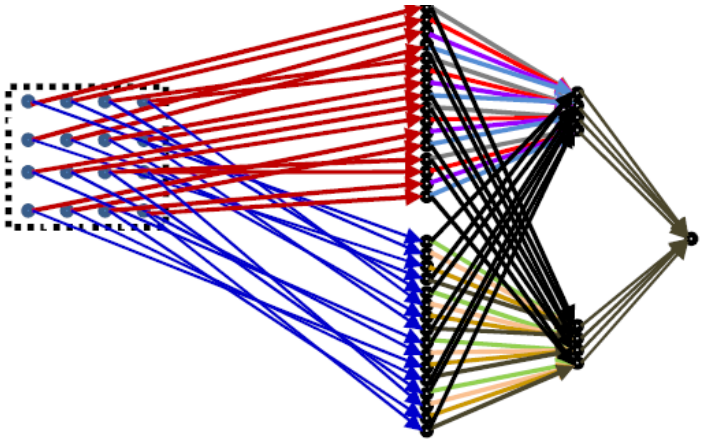
Conventional MLP, not distributed



$$\mathcal{O}(K^2 N_1 + N_1 N_2 + N_2 N_3 \dots)$$

- For this example, let K=16, N_1=4, N_2=2,N_3=1
  - Total 1034 weights

Distributed (3 layers)



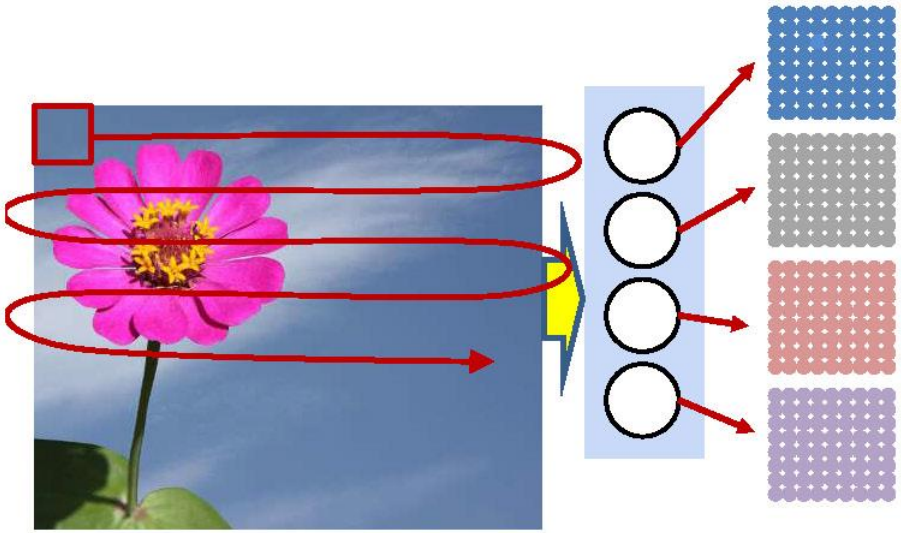
$$\mathcal{O}\left(L_1^2 N_1 + L_2^2 N_1 N_2 + \left(\frac{K}{L_1 L_2}\right)^2 N_2 N_3 + \cdots\right)$$

- Here, let K=16, L1=4, N_1=4, N_2=2,N_3=1
  - Total 64+128+8 = 160 weights

Hierarchical composition: A different perspective

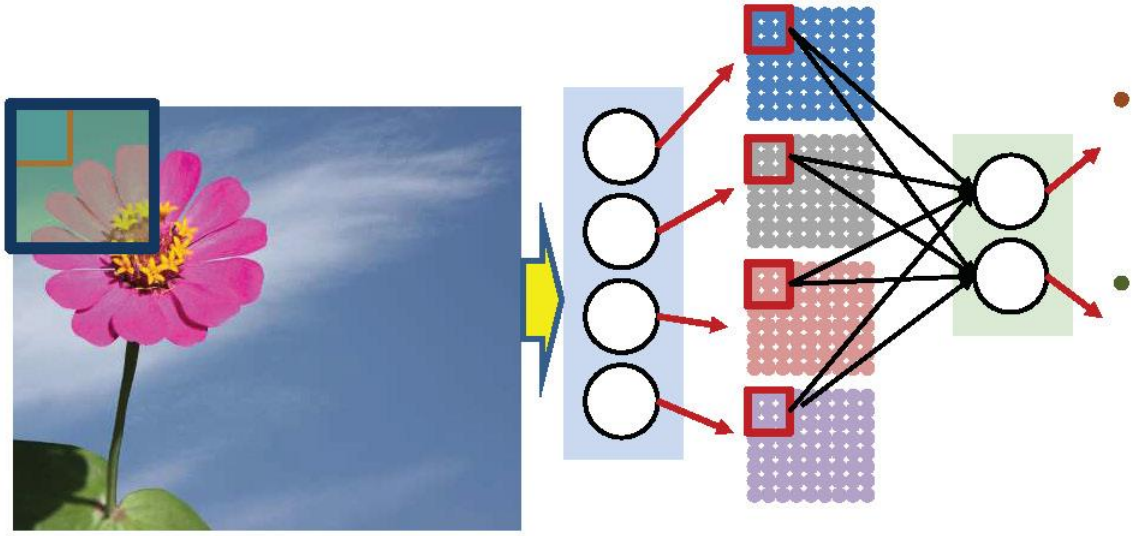
- The entire operation can be redrawn as before as maps of the entire image

Building up patterns



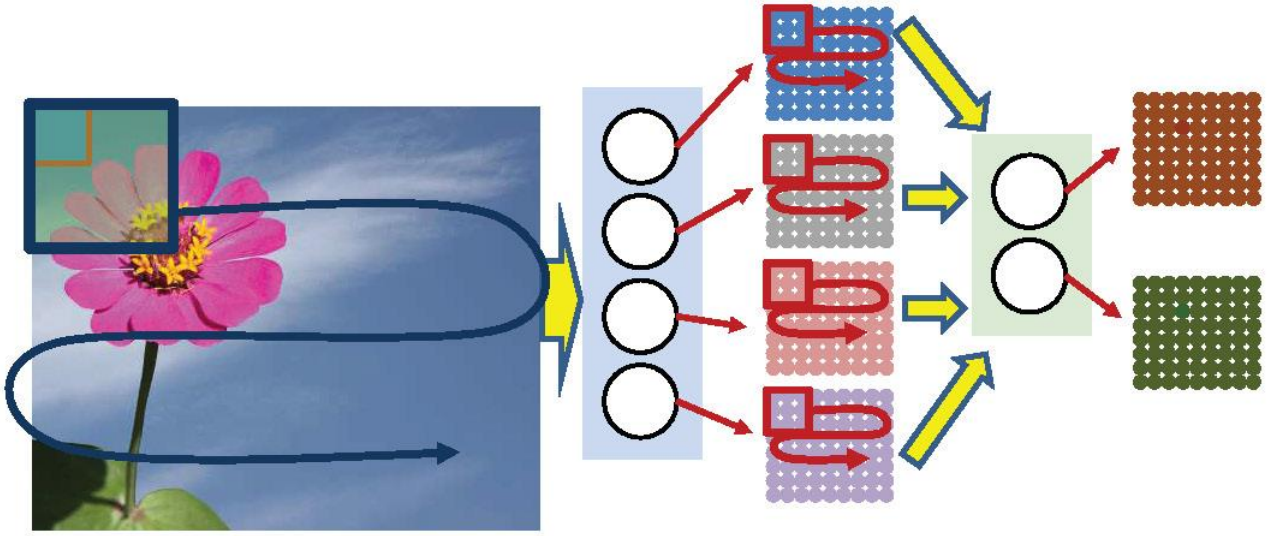
- The first layer looks at small sub regions of the main image
  - Sufficient to detect, say, petals

Some modifications



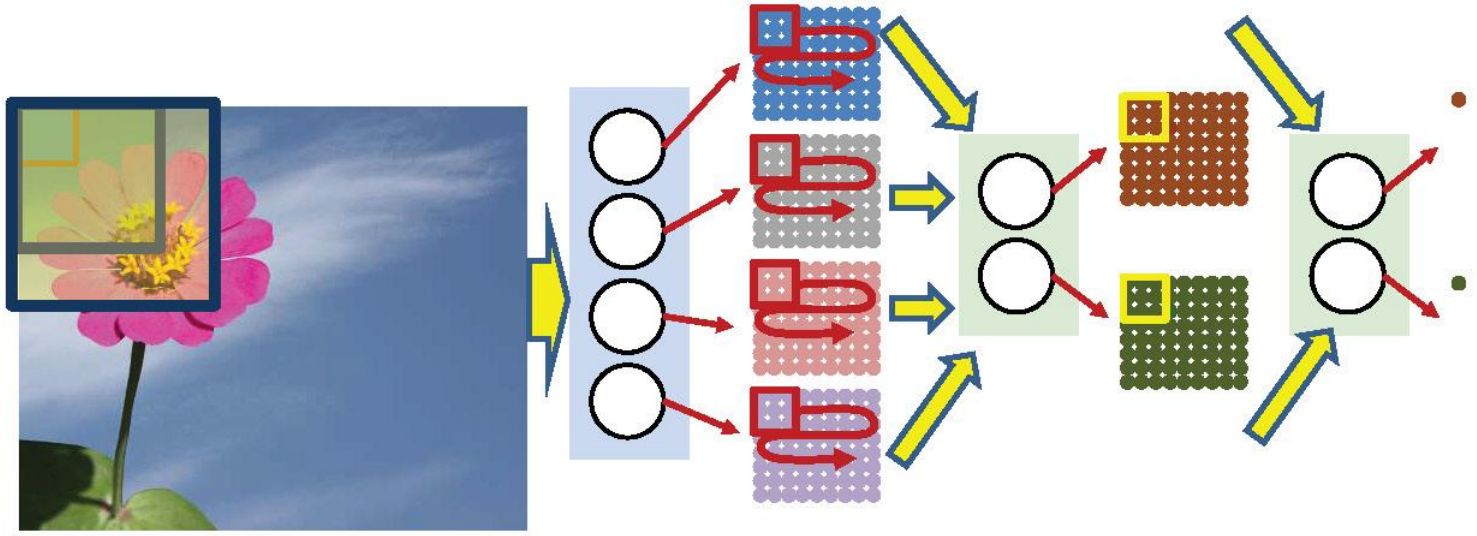
- The first layer looks at small sub regions of the main image
  - Sufficient to detect, say, petals
- The second layer looks at regions of the output of the first layer
  - To put the petals together into a flower
  - This corresponds to looking at a larger region of the original input image

Some modifications



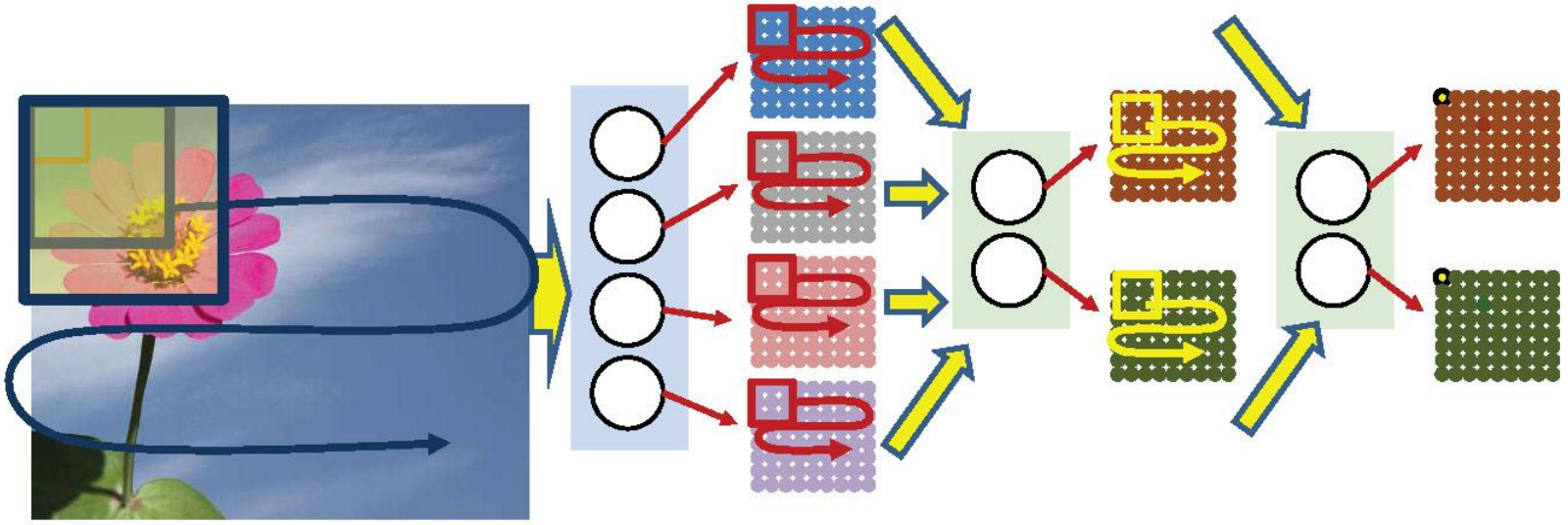
- The first layer looks at small sub regions of the main image
  - Sufficient to detect, say, petals
- The second layer looks at regions of the output of the first layer
  - To put the petals together into a flower
  - This corresponds to looking at a larger region of the original input image

Some modifications



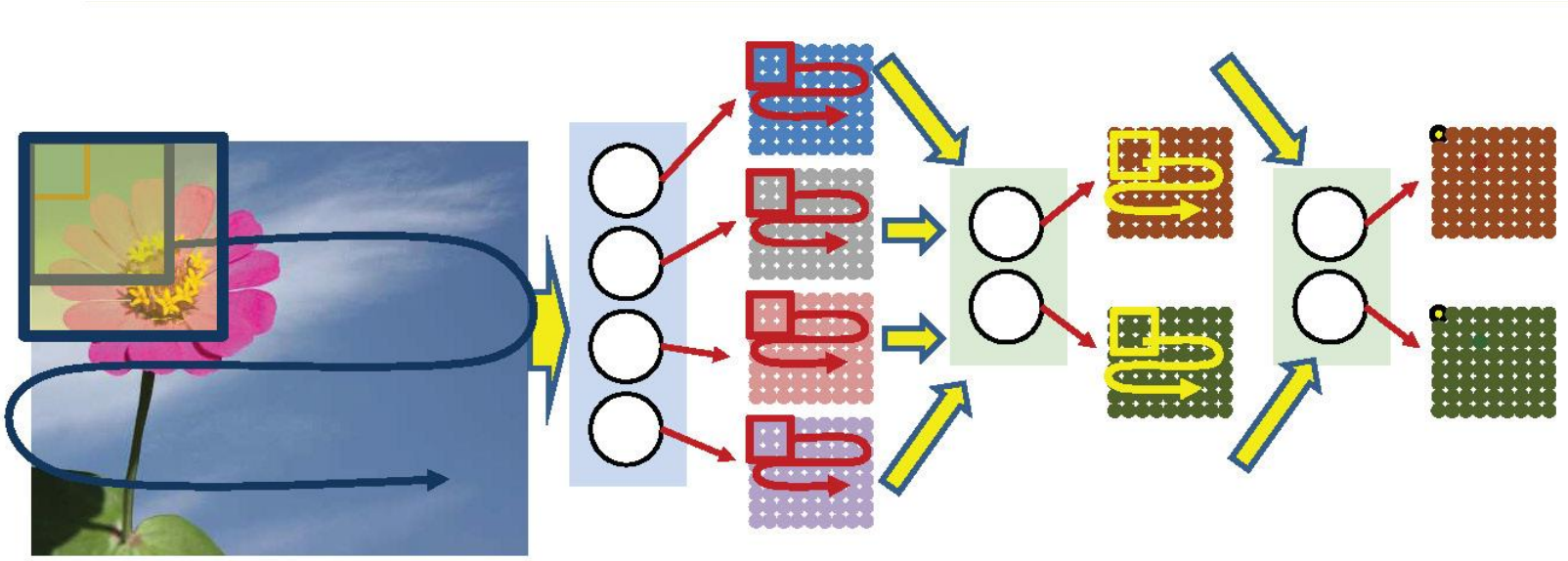
- The first layer looks at small sub regions of the main image
  - Sufficient to detect, say, petals
- The second layer looks at regions of the output of the first layer
  - To put the petals together into a flower
  - This corresponds to looking at a larger region of the original input image
- We may have any number of layers in this fashion

Some modifications



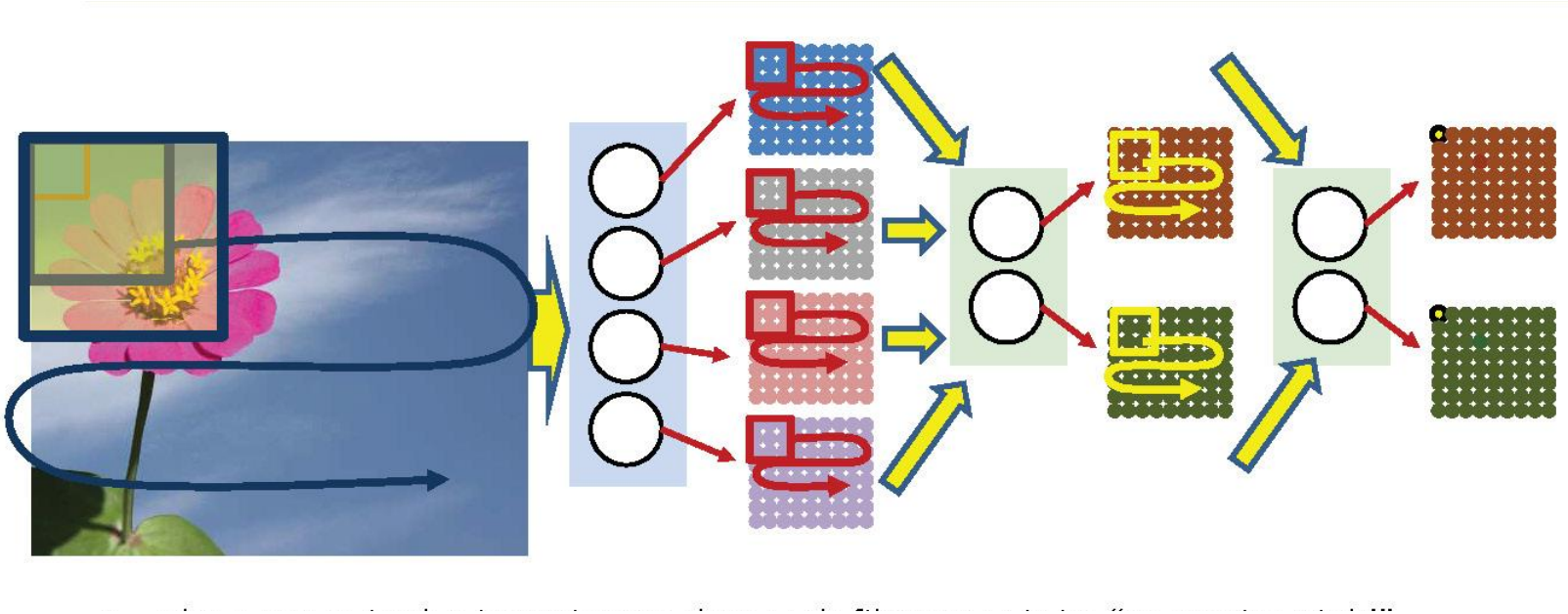
- The first layer looks at small sub regions of the main image
  - Sufficient to detect, say, petals
- The second layer looks at regions of the output of the first layer
  - To put the petals together into a flower
  - This corresponds to looking at a larger region of the original input image
- We may have any number of layers in this fashion

Terminology



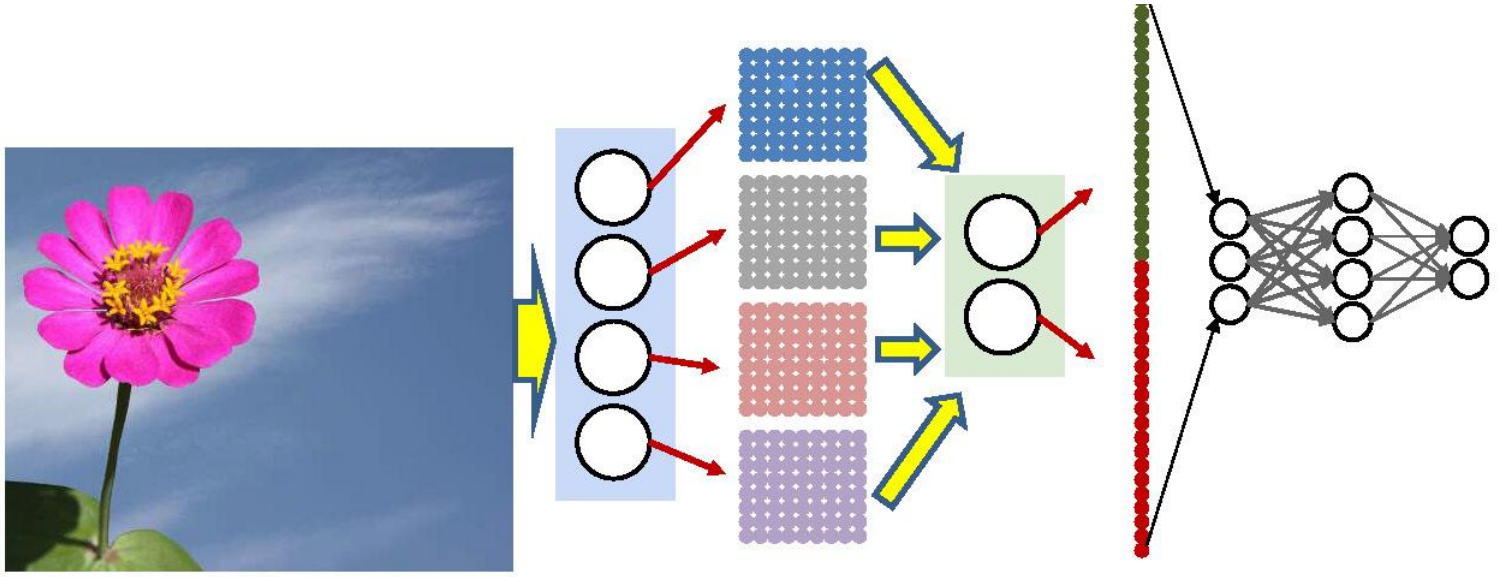
- Each of the scanning neurons is generally called a "filter"
  - Its really a correlation filter as we saw earlier
  - Each filter scans for a pattern in the map it operates on

Terminology



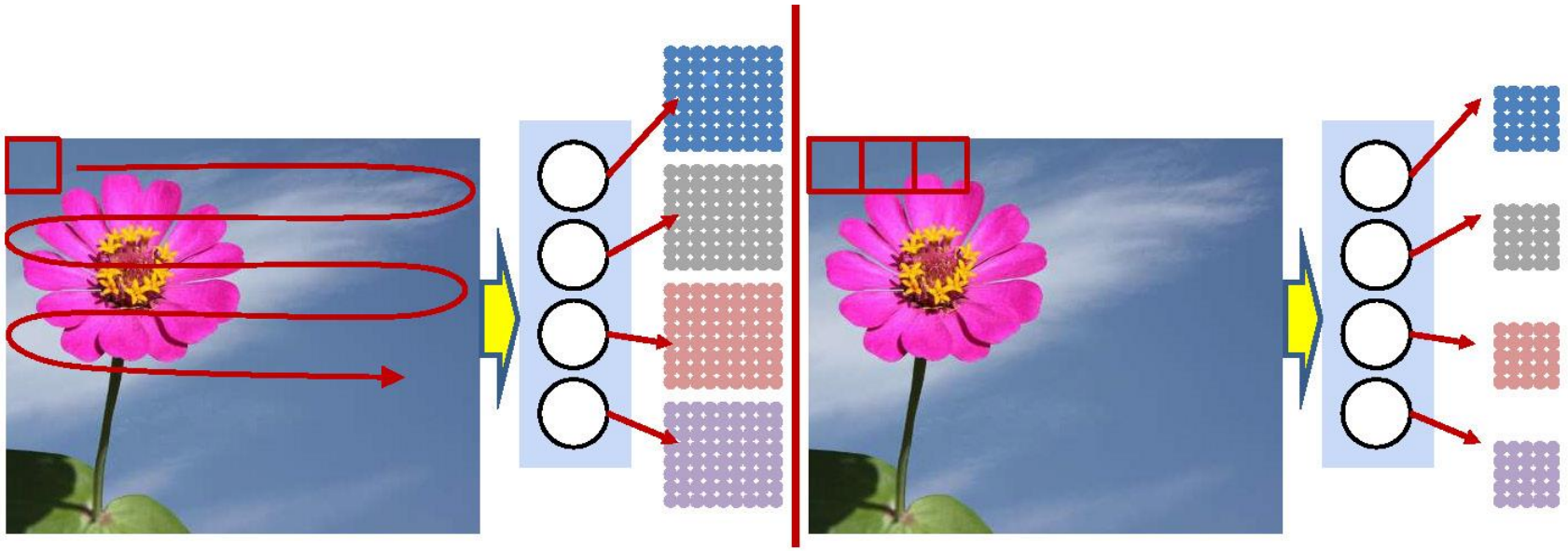
- The pattern in the input image that each filter sees is its "Receptive Field"
  - The squares show the sizes of the receptive fields for the first, second and third-laye neurons
- The actual receptive field for a first layer filter is simply its arrangement of weights
- For the higher level filters, the actual receptive field is not immediately obvious and must be calculated
  - What patterns in the input do the filters actually respond to?
  - Will not actually be simple, identifiable patterns like "petal" and "inflorescence"

Some modifications



- The final layer may feed directly into a multi layer perceptron rather than a single neuron
- This is exactly the shared parameter net we just saw

Modification 1: Convolutional "Stride"



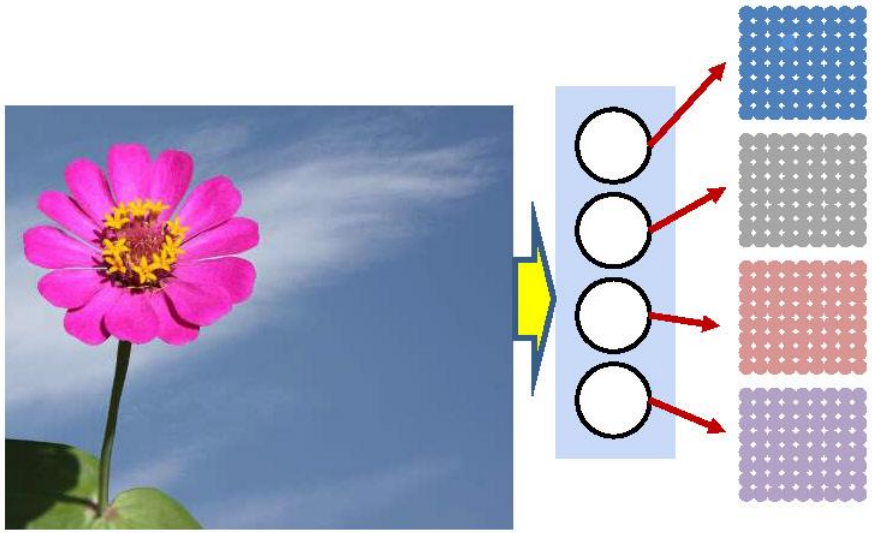
- The scans of the individual "filters" may advance by more than one pixel at a time
  - The "stride" may be greater than 1
  - Effectively increasing the granularity of the scan
- Saves computation, sometimes at the risk of losing information
- This will result in a reduction of the size of the resulting maps
  - They will shrink by a factor equal to the stride
- This can happen at any layer

Convolutional neural net

The weight W(l,j) is now a 3D $D_{l-1}xK_lxK_l$ tensor (assuming square receptive fields)

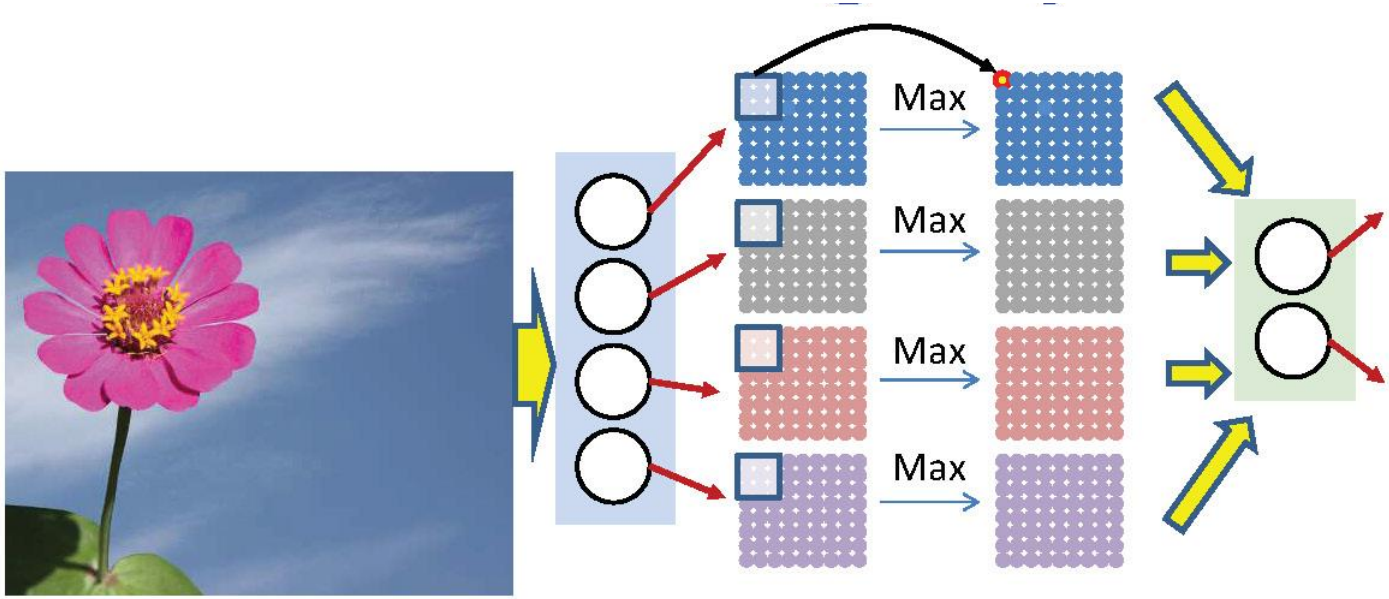
```
Y(0) = Image
for l = 1:L  # layers operate on vector at (x,y)
  for j = 1:D_l
    m = 1
    for x = 1:stride:W_{l-1}-K_l+1
       n = 1
       for y = 1:stride:H_{l-1}-K_l+1
          segment = Y(l-1,:,x:x+K_l-1,y:y+K_l-1) #3D tensor
          z(l,j,m,n) = W(l,j).segment #tensor inner prod.
          Y(l,j,m,n) = activation(z(l,j,m,n))
          n++
        m++

Y = softmax( Y(L) )
```

Accounting for jitter



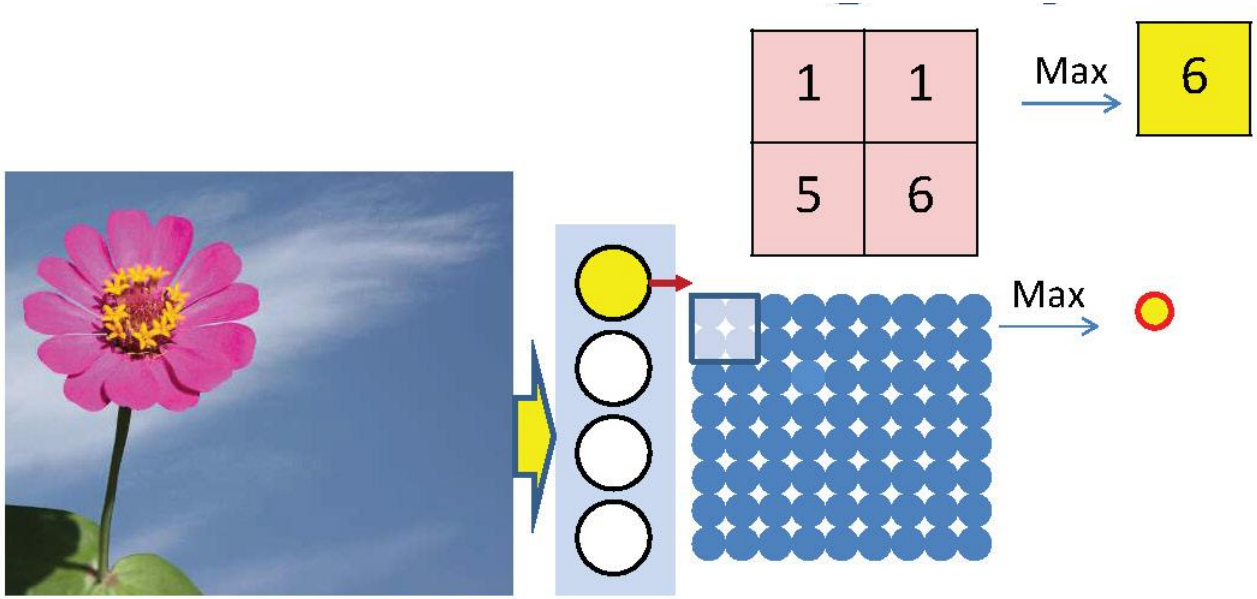
- We would like to account for some jitter in the first-level patterns
  - If a pattern shifts by one pixel, is it still a petal?

Accounting for jitter



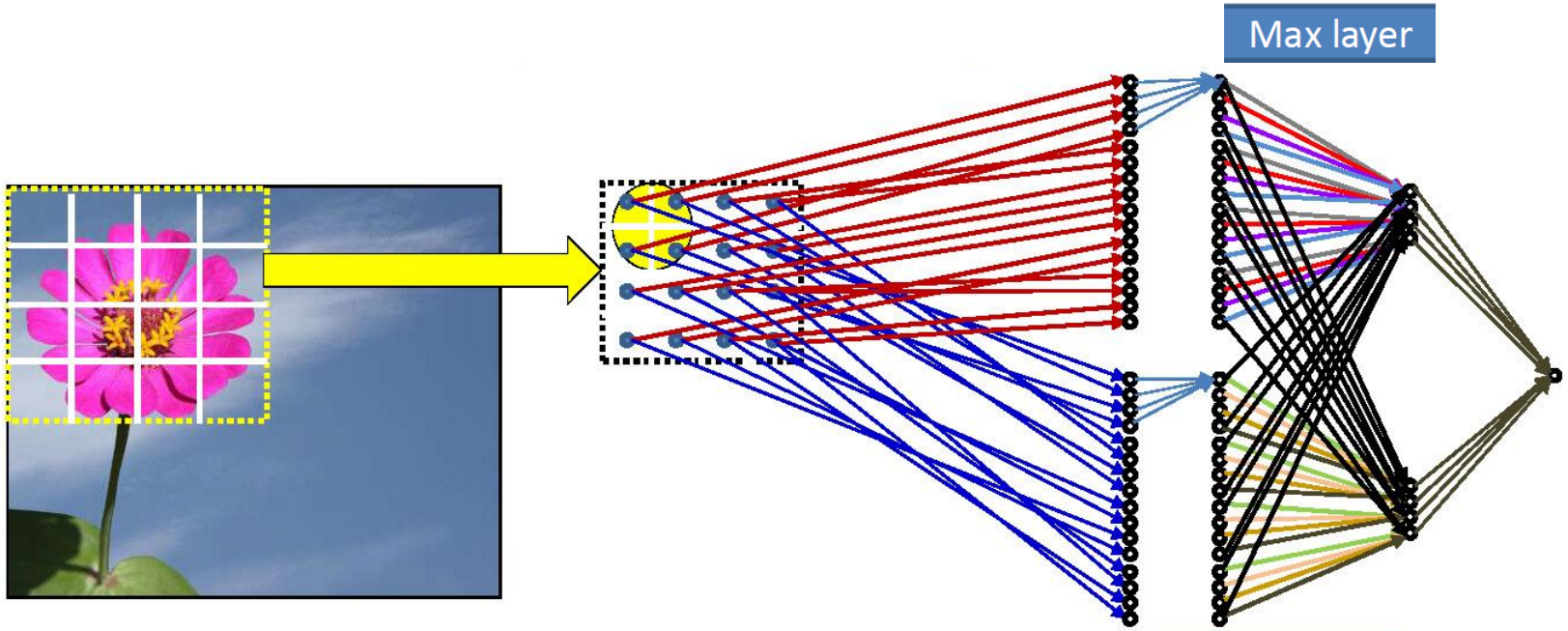
- We would like to account for some jitter in the first-level patterns
  - If a pattern shifts by one pixel, is it still a petal?
  - A small jitter is acceptable
- Replace each value by the maximum of the values within a small region around it
  - Max filtering or Max pooling

Accounting for jitter



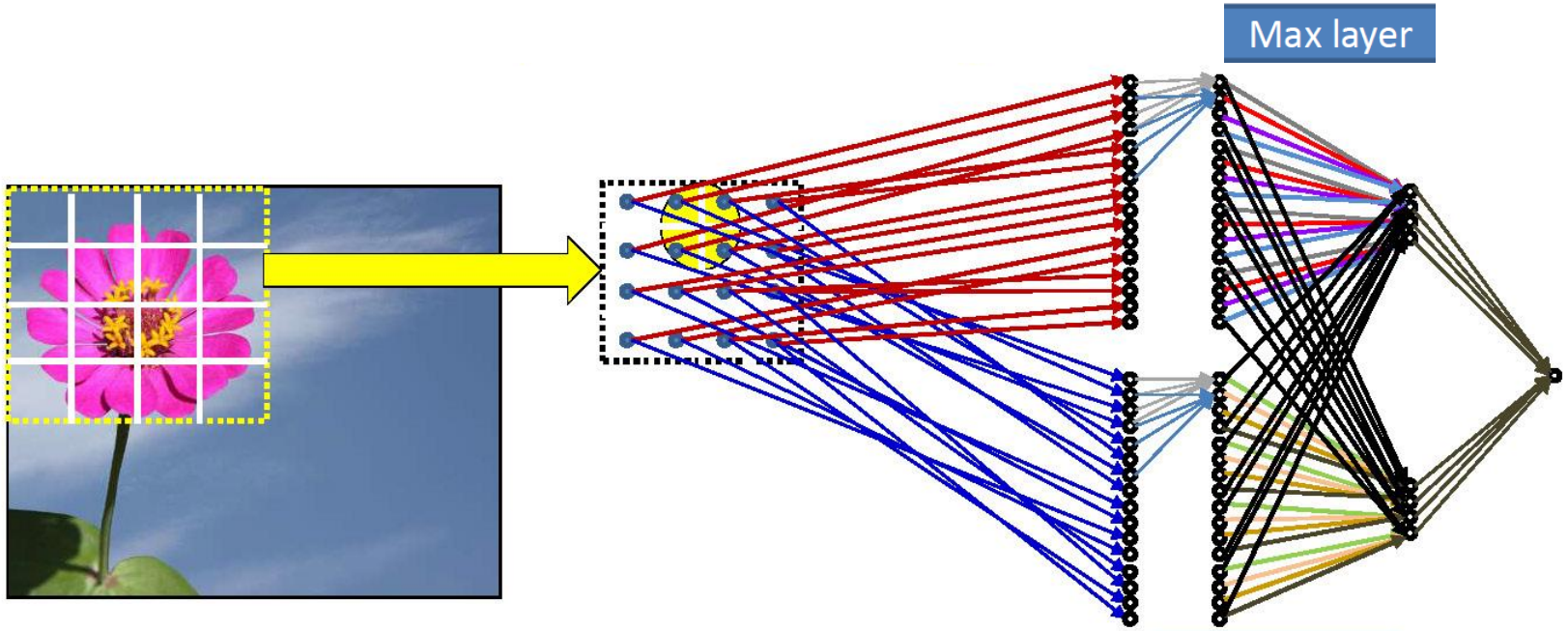
- We would like to account for some jitter in the first-level patterns
  - If a pattern shifts by one pixel, is it still a petal?
  - A small jitter is acceptable
- Replace each value by the maximum of the values within a small region around it
  - **Max filtering or Max pooling**

The max operation is just a neuron



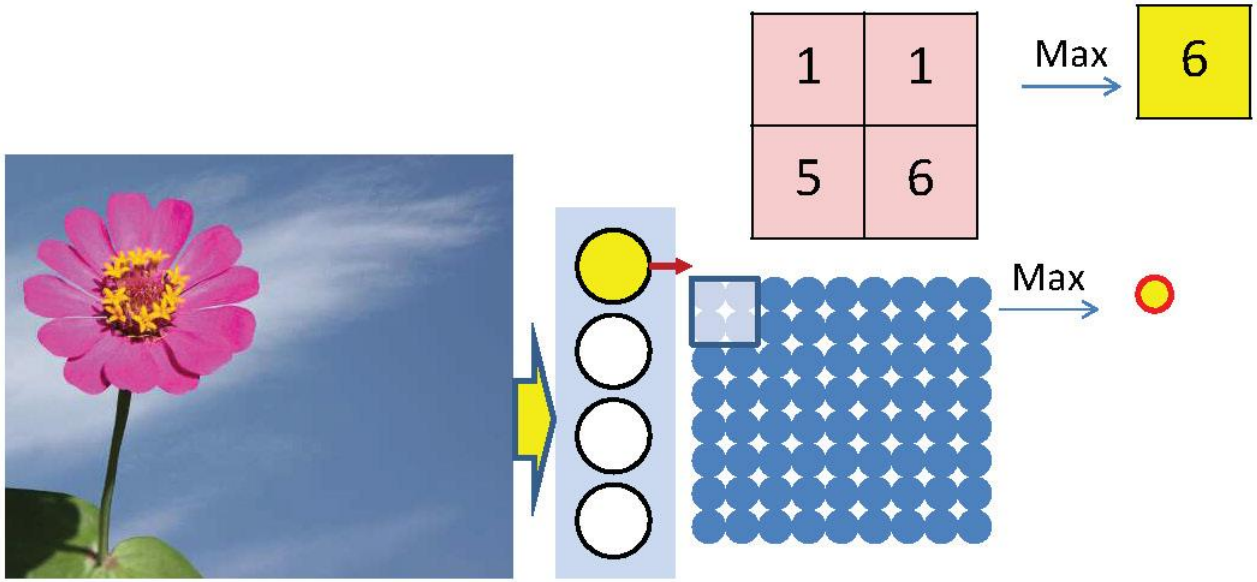
- The max operation is just another neuron
- Instead of applying an activation to the weighted sum of inputs, each neuron just computes the maximum over all inputs

The max operation is just a neuron



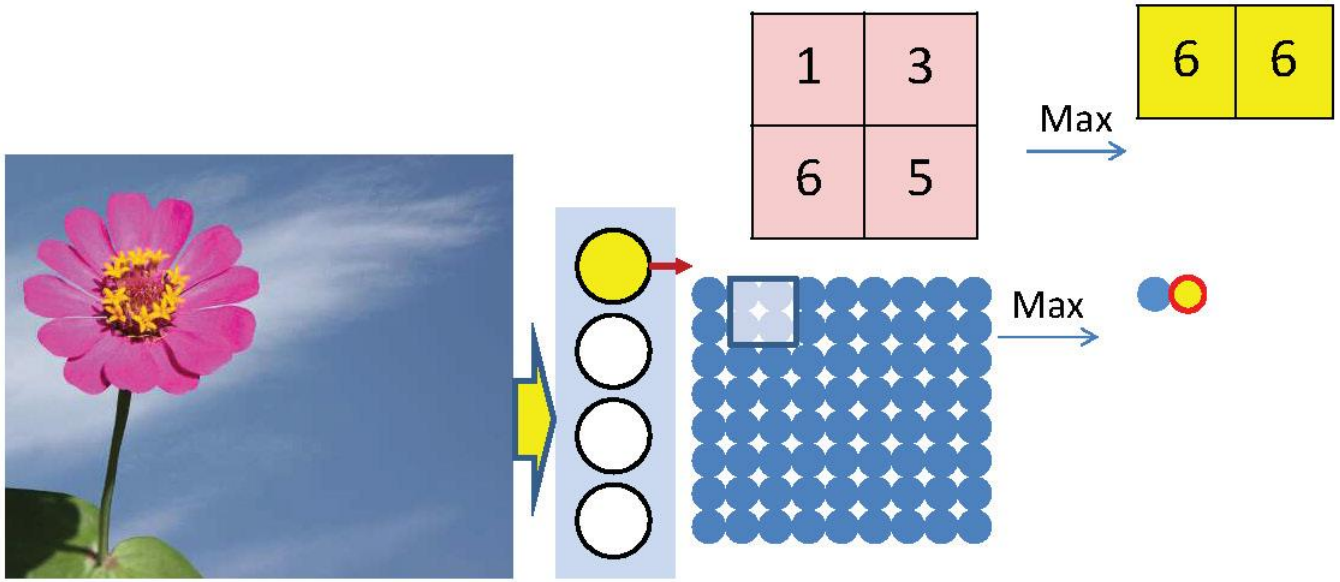
- The max operation is just another neuron
- Instead of applying an activation to the weighted sum of inputs, each neuron just computes the maximum over all inputs

Accounting for jitter



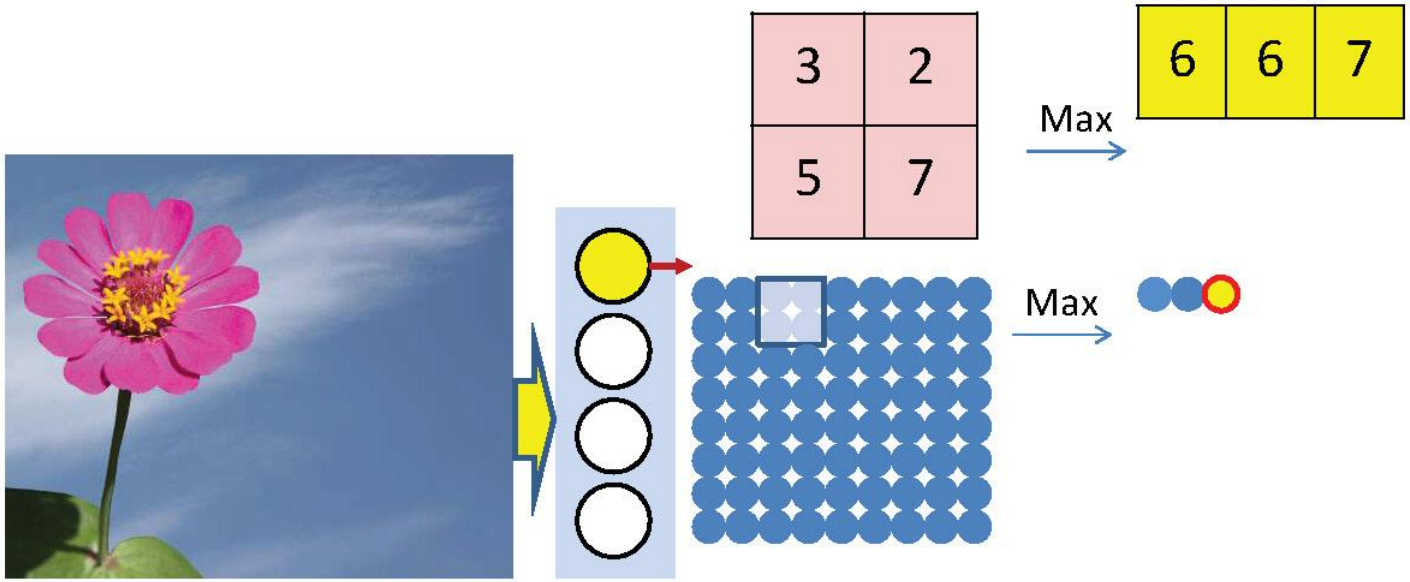
- The max filtering can also be performed as a scan

Accounting for jitter



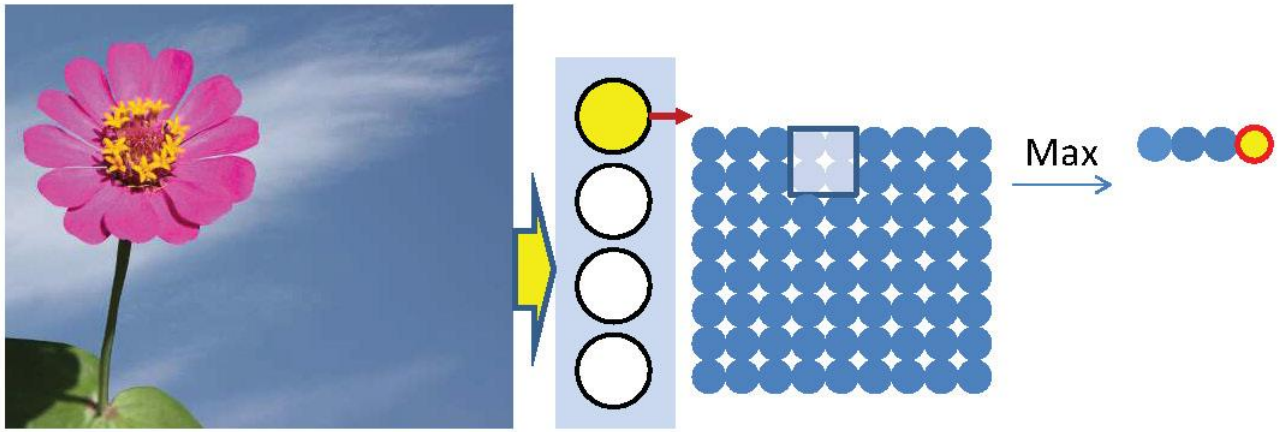
- The max filtering can also be performed as a scan

Accounting for jitter



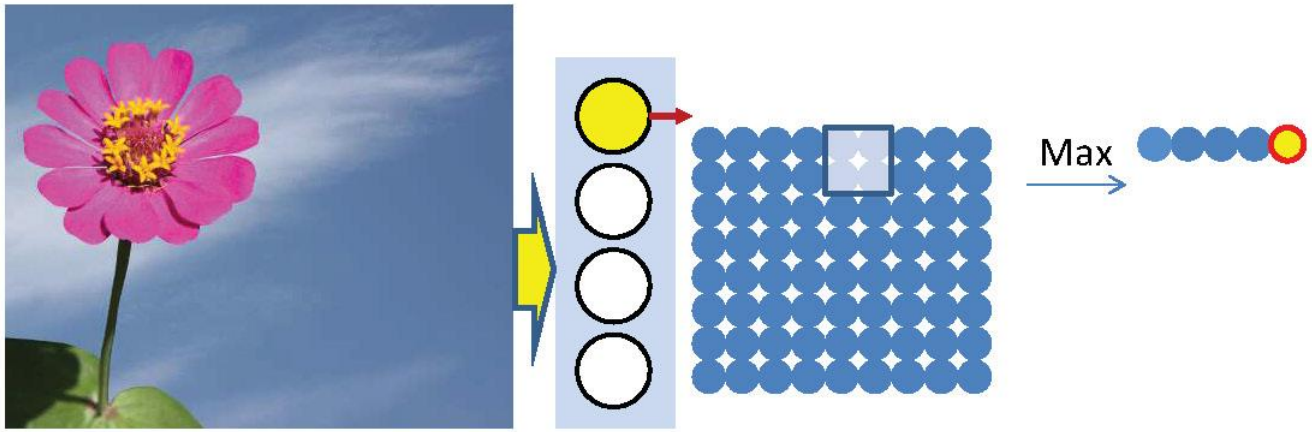
- The max filtering can also be performed as a scan

Accounting for jitter



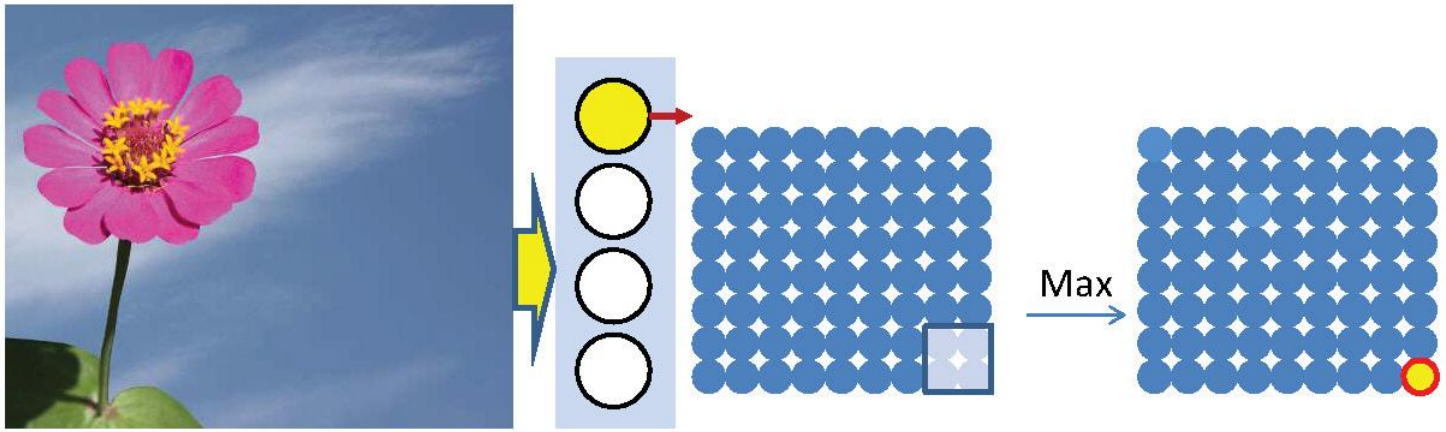
- The max filtering can also be performed as a scan

Accounting for jitter



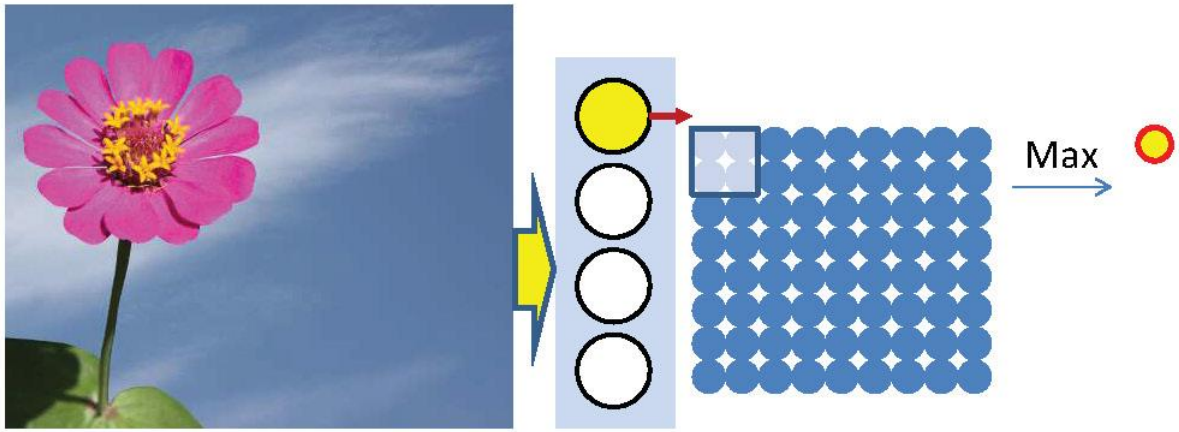
- The max filtering can also be performed as a scan

Accounting for jitter



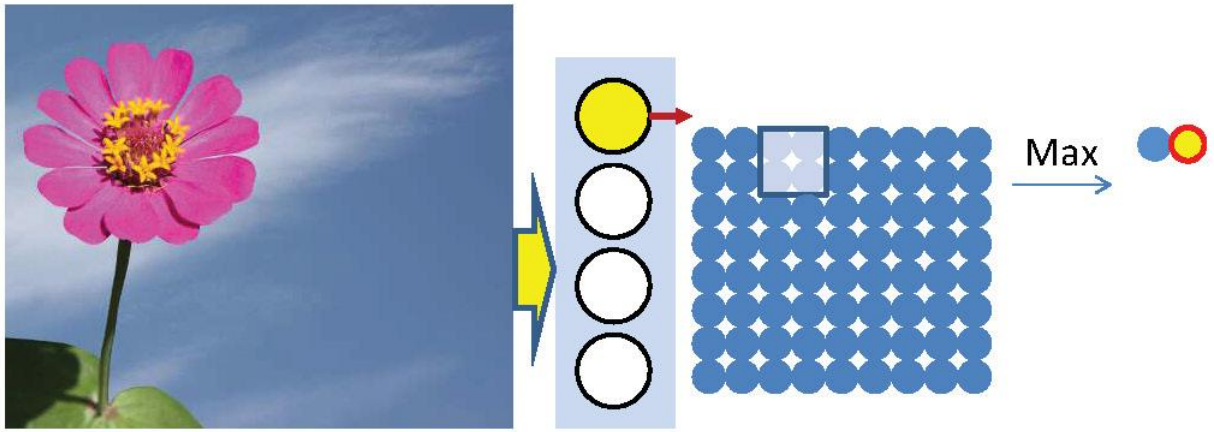
- The max filtering can also be performed as a scan

Max pooling "Strides"



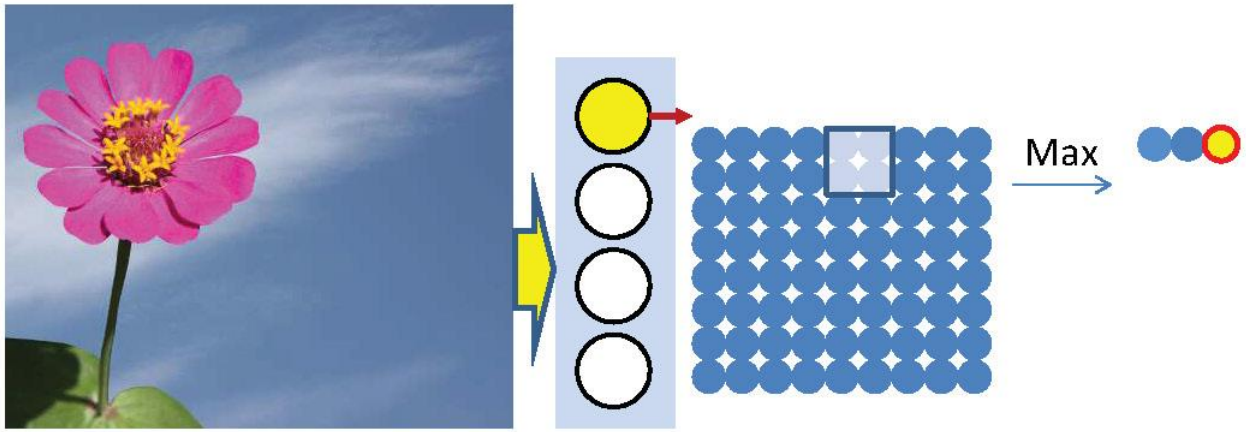
- The "max" operations may "stride" by more than one pixel

Max pooling "Strides"



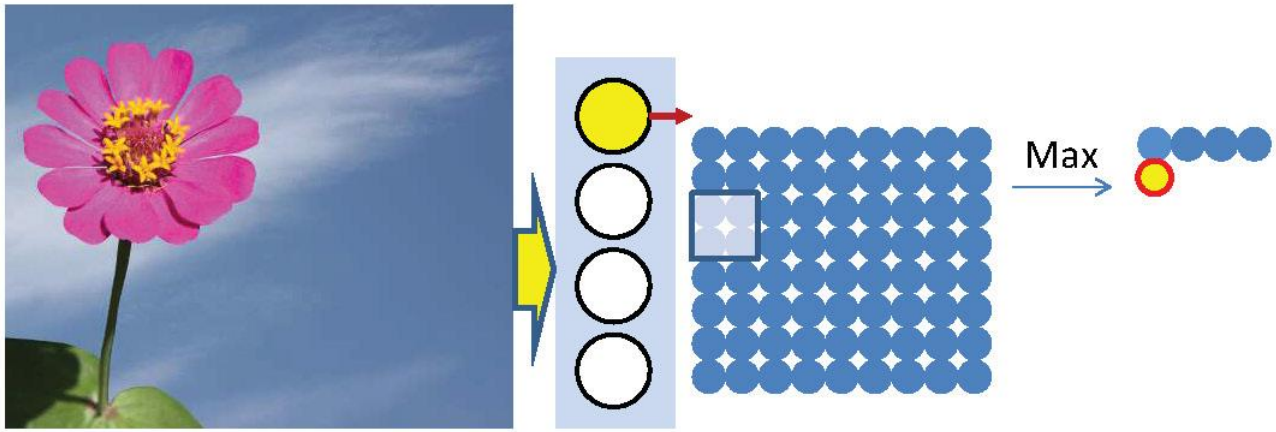
- The "max" operations may "stride" by more than one pixel

Max pooling "Strides"



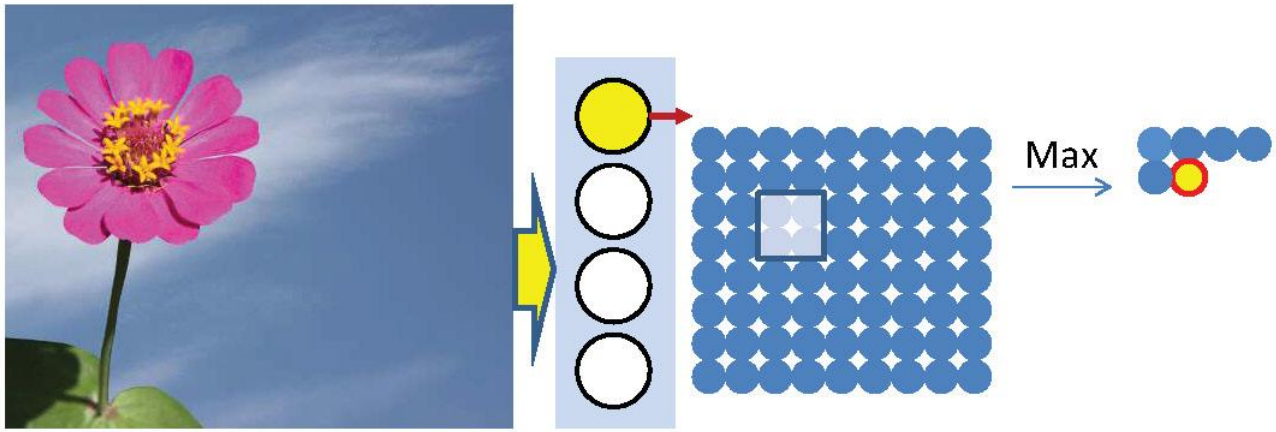
- The "max" operations may "stride" by more than one pixel

Max pooling "Strides"



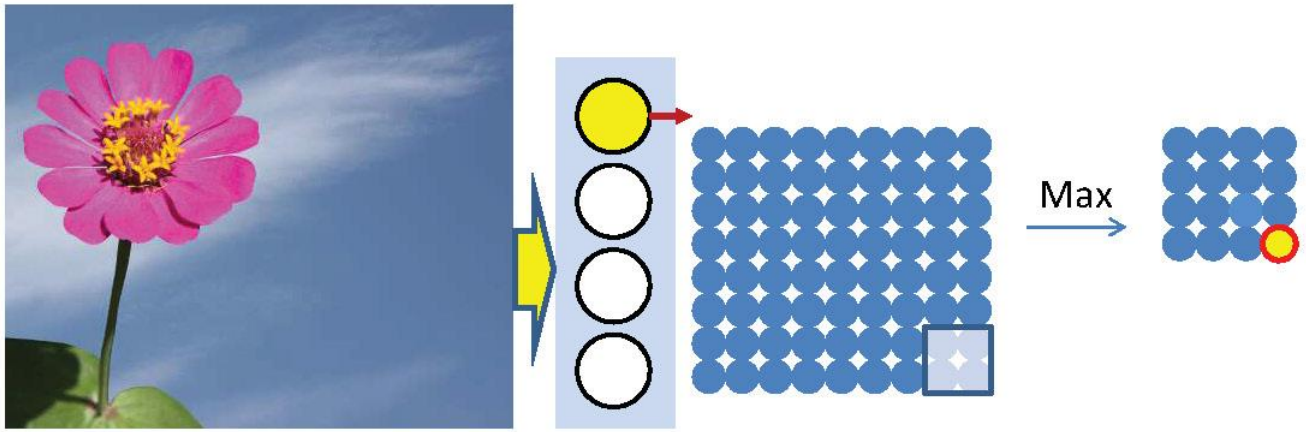
- The "max" operations may "stride" by more than one pixel

Max pooling "Strides"



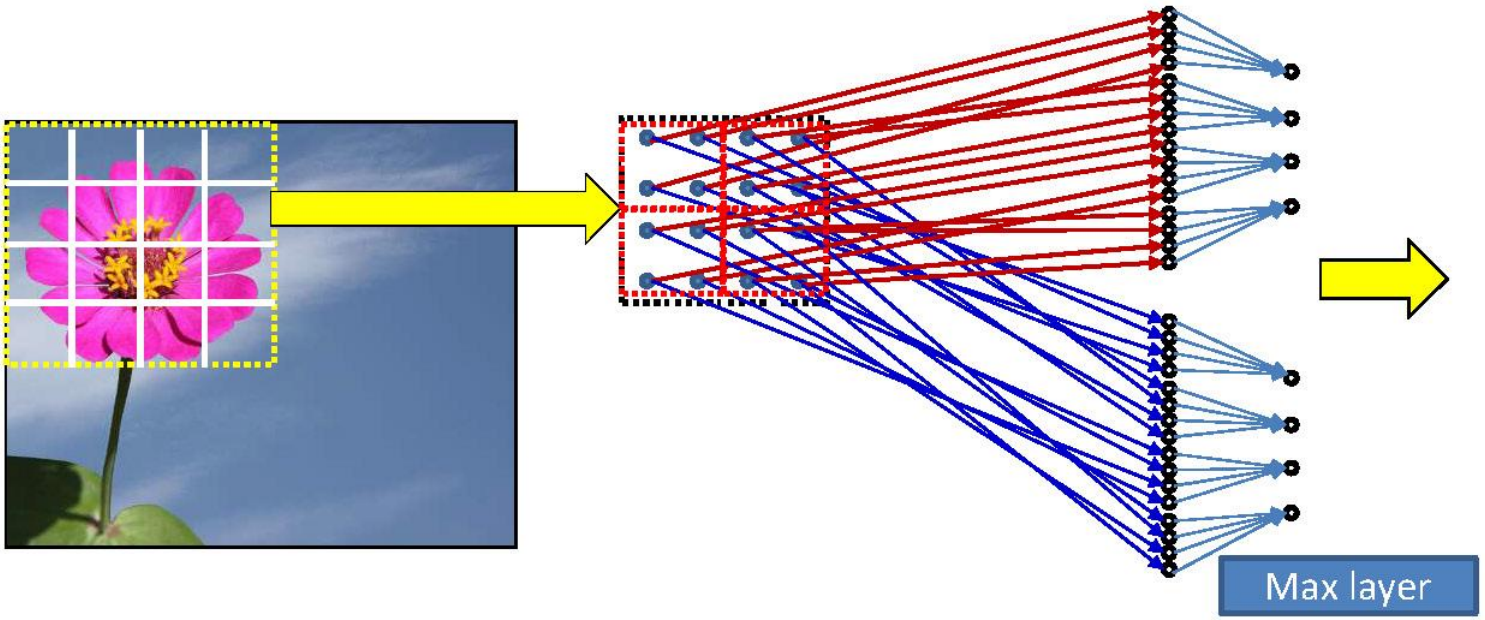
- The "max" operations may "stride" by more than one pixel

Max pooling "Strides"



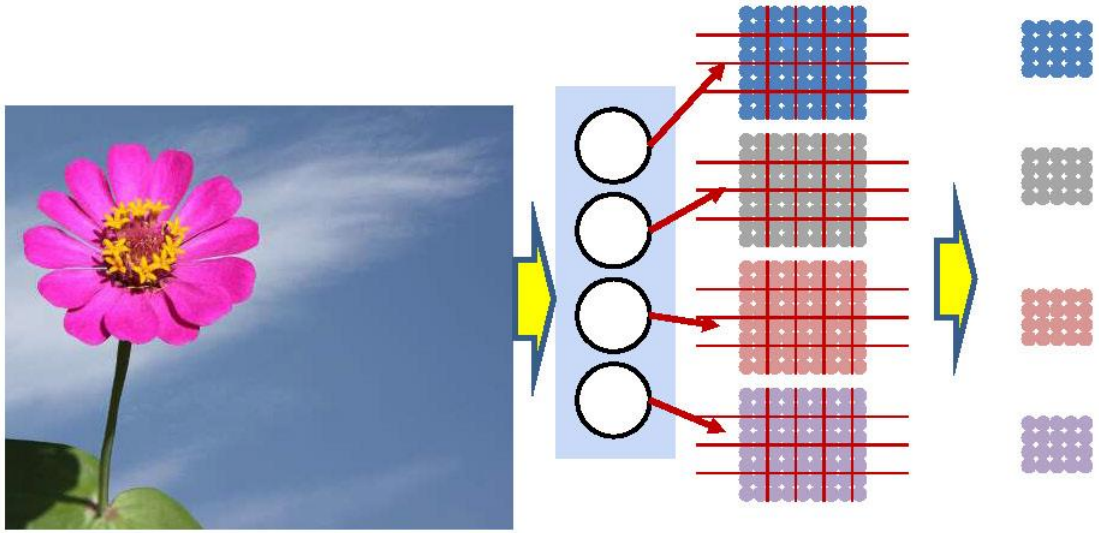
- The "max" operations may "stride" by more than one pixel
  - This will result in a shrinking of the map
  - The operation is usually called "pooling"
    - Pooling a number of outputs to get a single output
    - When stride is greater than 1, also called "Down sampling"

Shrinking with a max



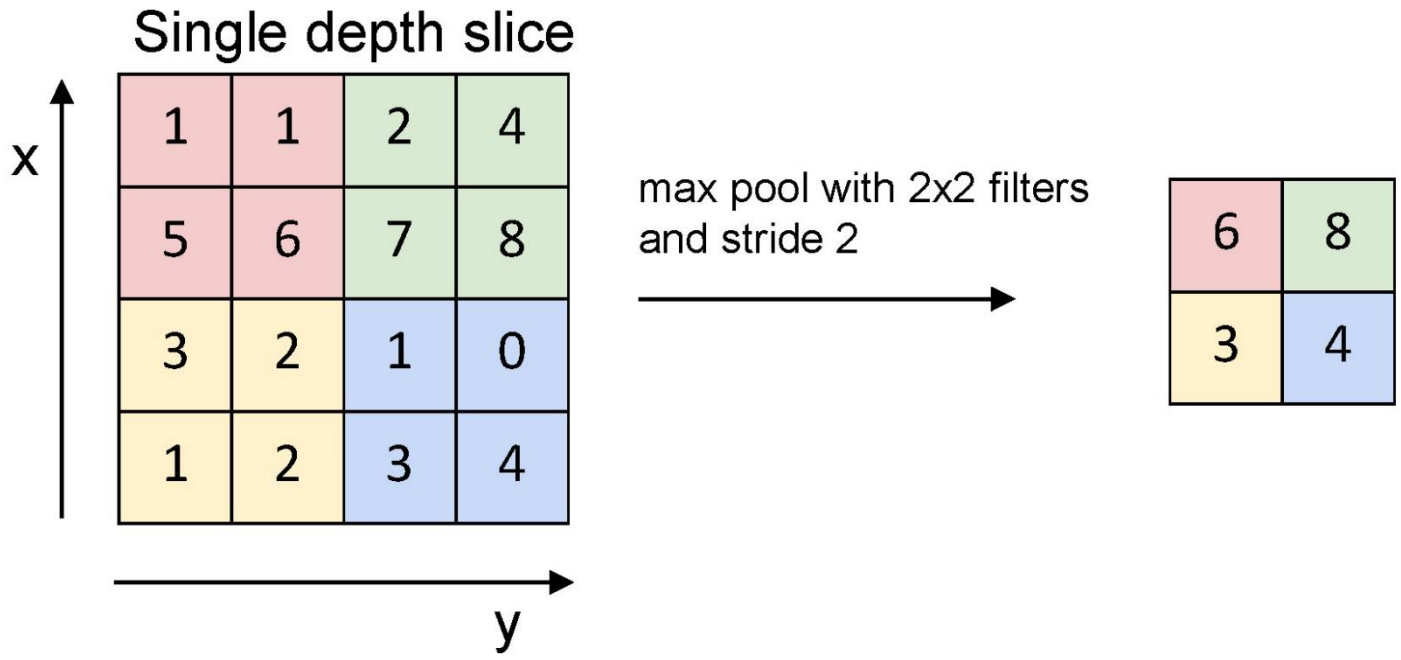
- The "max" operations may "stride" by more than one pixel
  - This will result in a shrinking of the map
  - The operation is usually called "pooling"
    - Pooling a number of outputs to get a single output
    - When stride is greater than 1, also called "Down sampling"

Non-overlapped strides

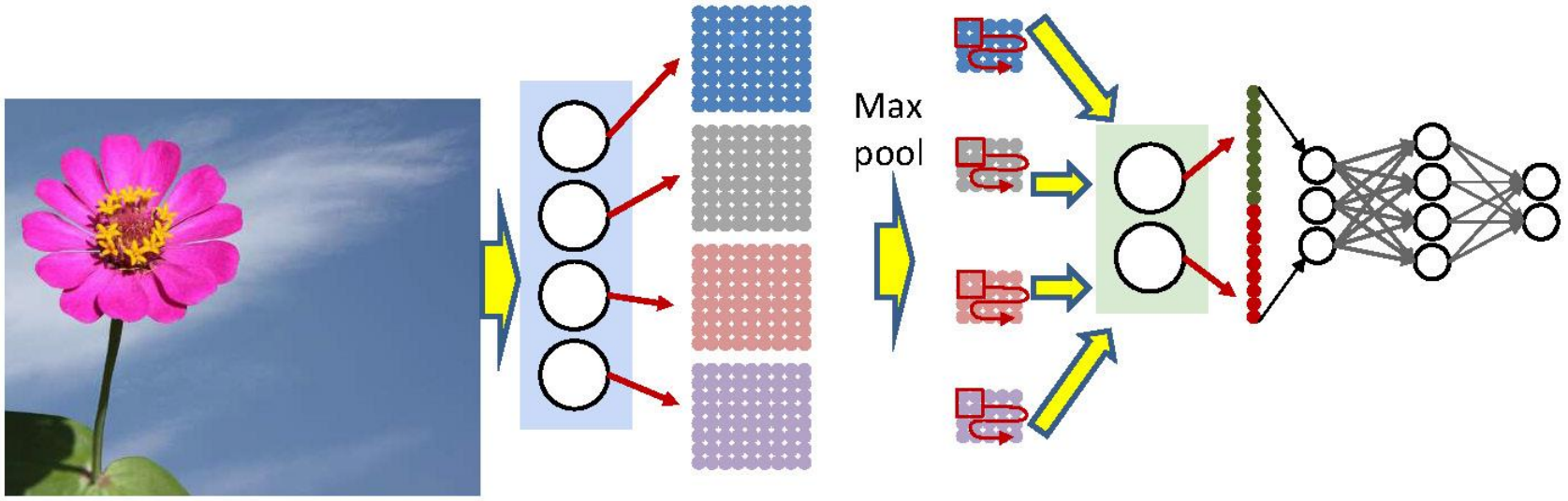


- Non-overlapping strides: Partition the output of the layer into blocks
- Within each block only retain the highest value
  - If you detect a petal anywhere in the block, a petal is detected..

Max Pooling

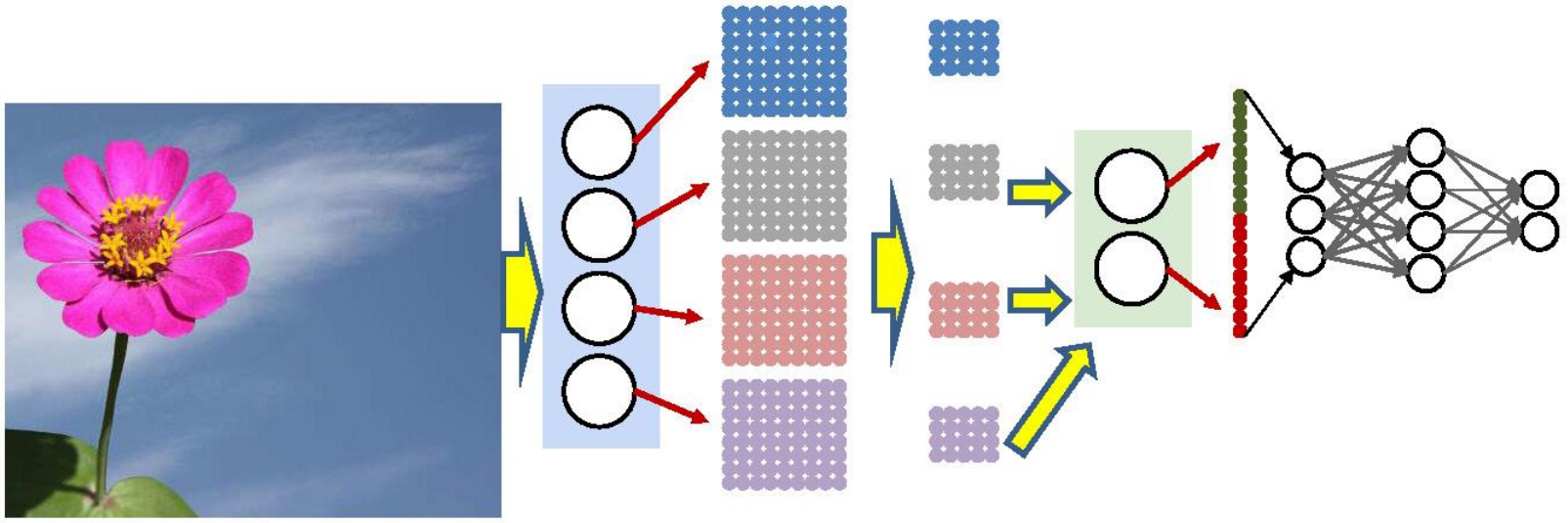

Single depth slice
x
y
1 1 2 4
5 6 7 8
3 2 1 0
1 2 3 4
max pool with 2x2 filters
and stride 2
6 8
3 4

Higher layers



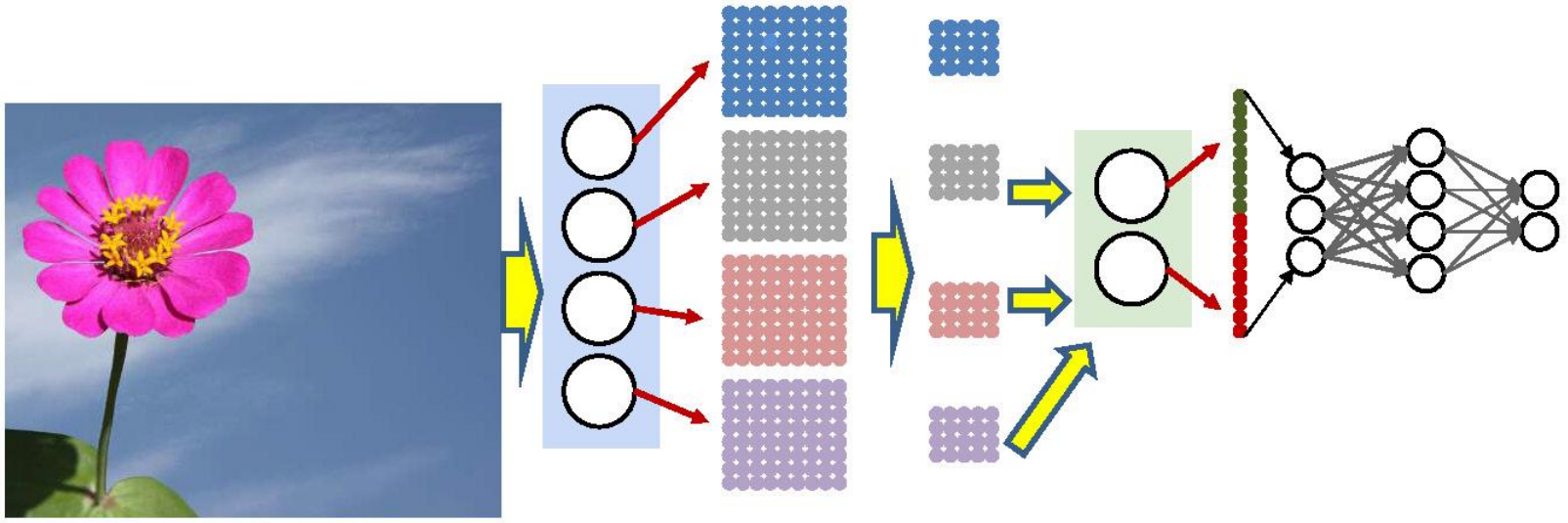
- The next layer works on the max-pooled maps

The overall structure



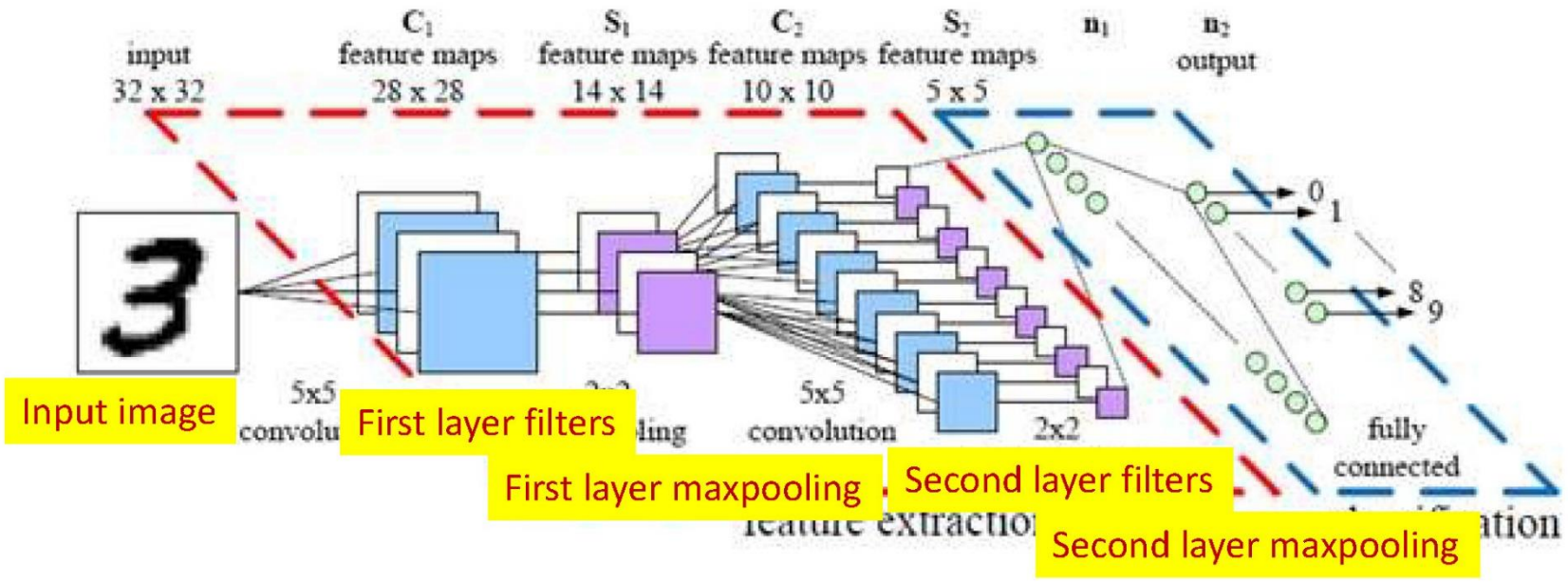
- In reality we can have many layers of "convolution" (scanning) followed by max pooling (and reduction) before the final MLP
  - The individual perceptrons at any "scanning" or "convolutional" layer are called "filters"
    - They "filter" the input image to produce an output image (map)
  - The individual max operations are also called max pooling or max filters

The overall structure

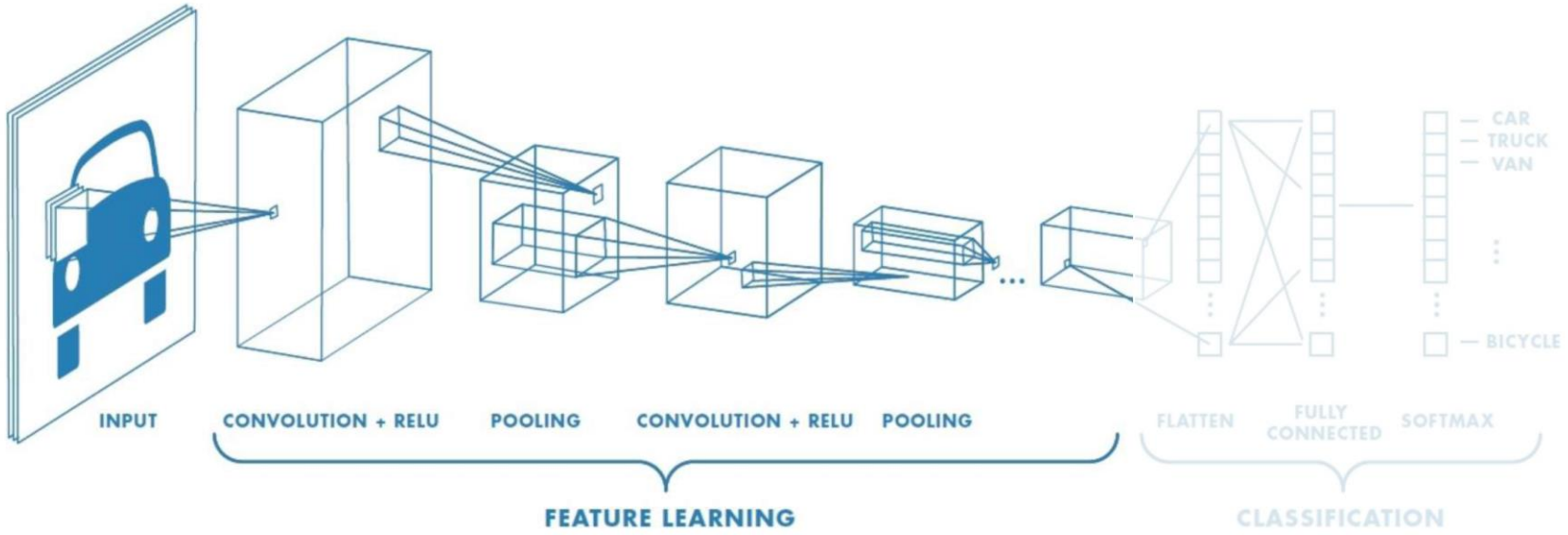


- This entire structure is called a **Convolutional Neural Network**

Convolutional Neural Network

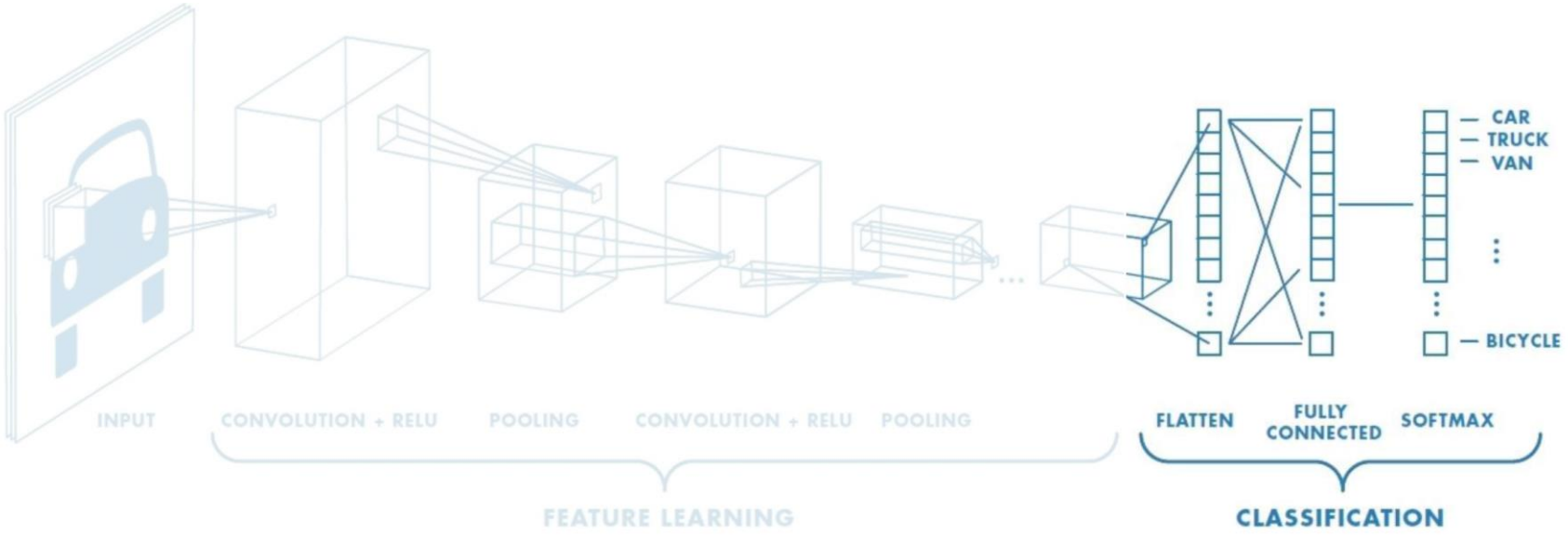


CNNs for Classification: Feature Learning



1. **Learn features** in input image through **convolution**
2. Introduce **non-linearity** through activation function (real-world data is non-linear!)
3. Reduce dimensionality and preserve spatial invariance with **pooling**

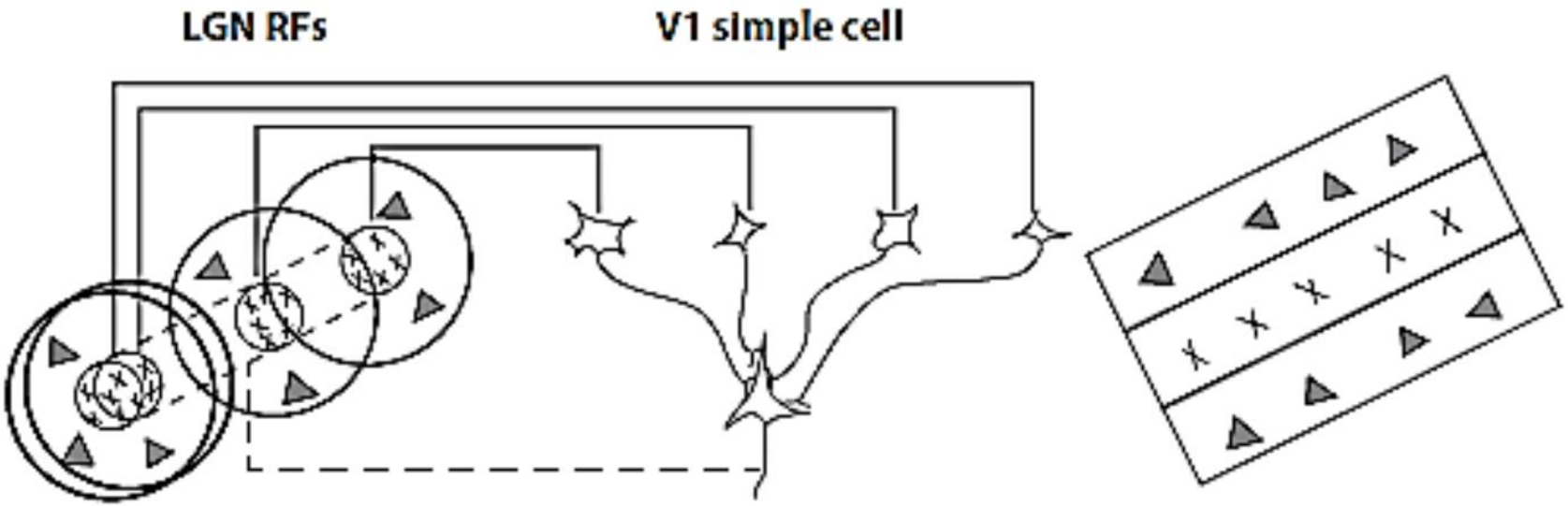
CNNs for Classification: Class Probabilities



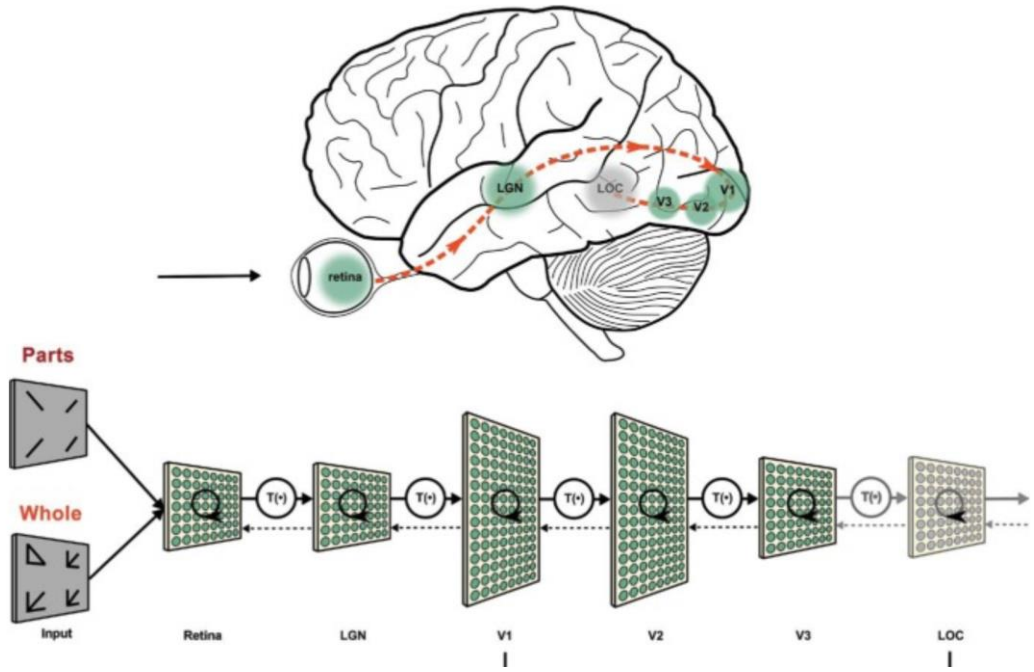
- **CONV and POOL** layers output **high-level features** of input
- **Fully connected** layer uses these features for classifying input image
- Express output as **probability** of image belonging to a particular class

$$\text{softmax}(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}}$$

Complex feature extraction in brain

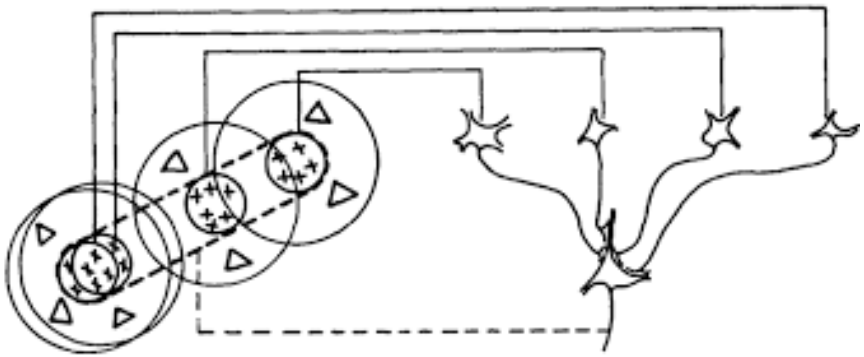


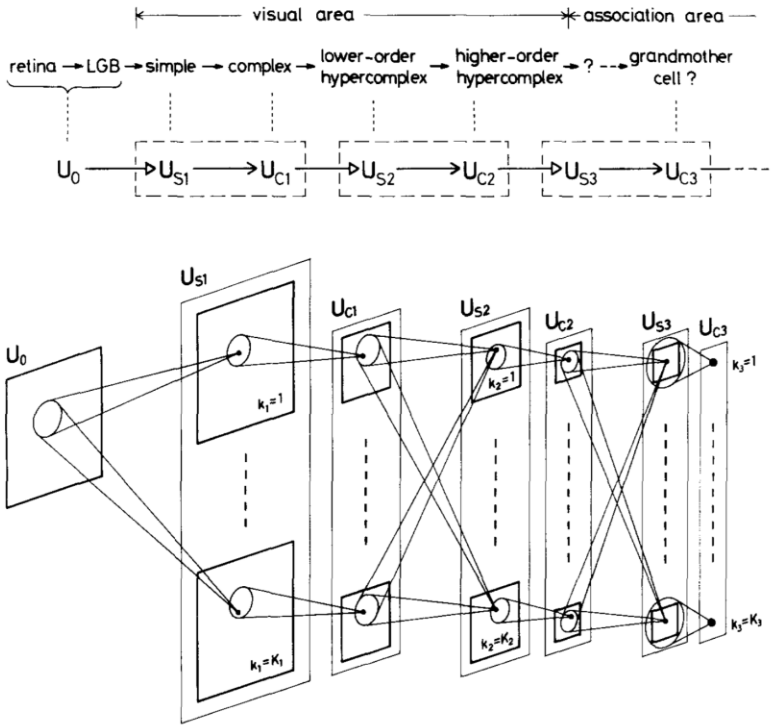
Hierarchical visual processing to transform images into a better format across many layers



Neocognitron; Emergence of convolutional neural network (1982)

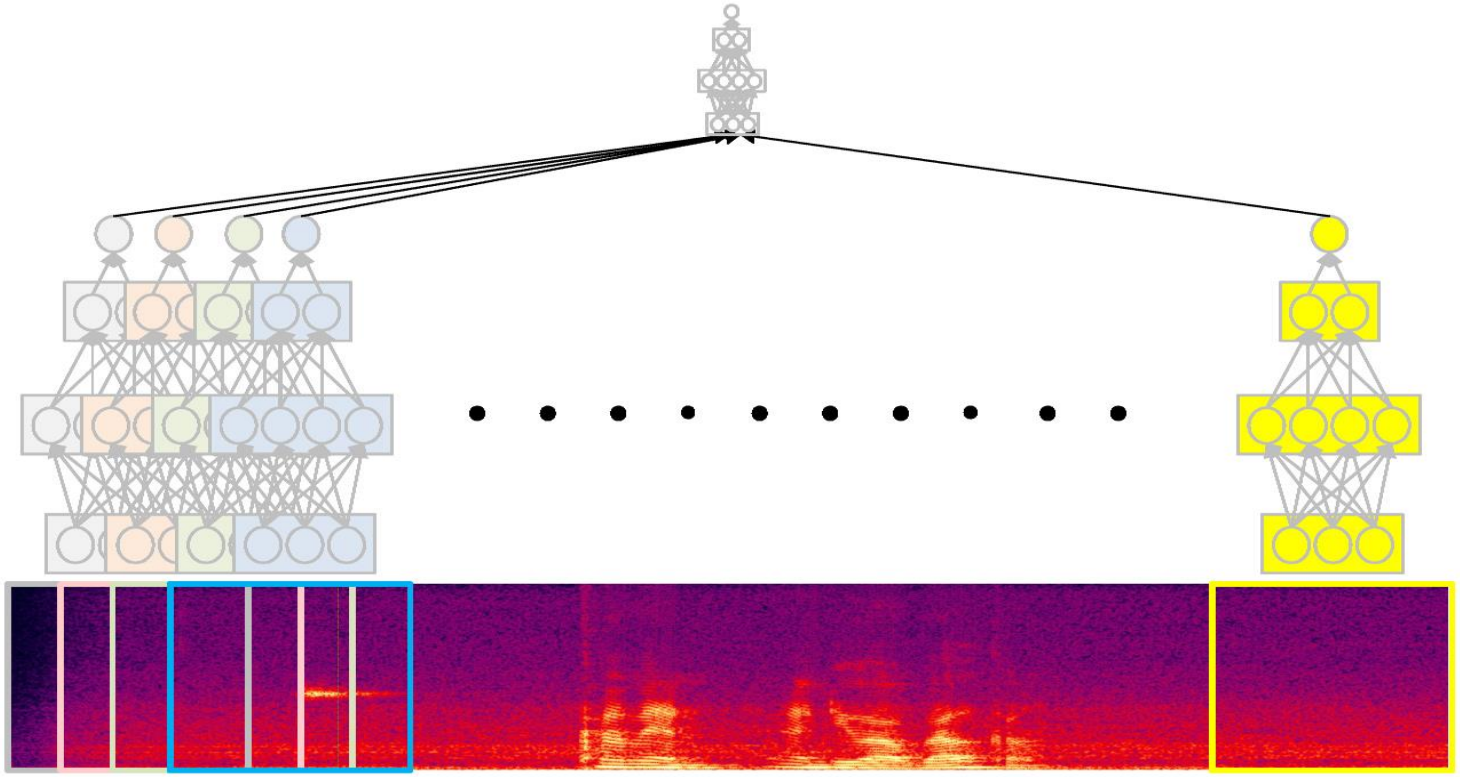
simple-to-complex pooling in Fukushima's **neocognitron**





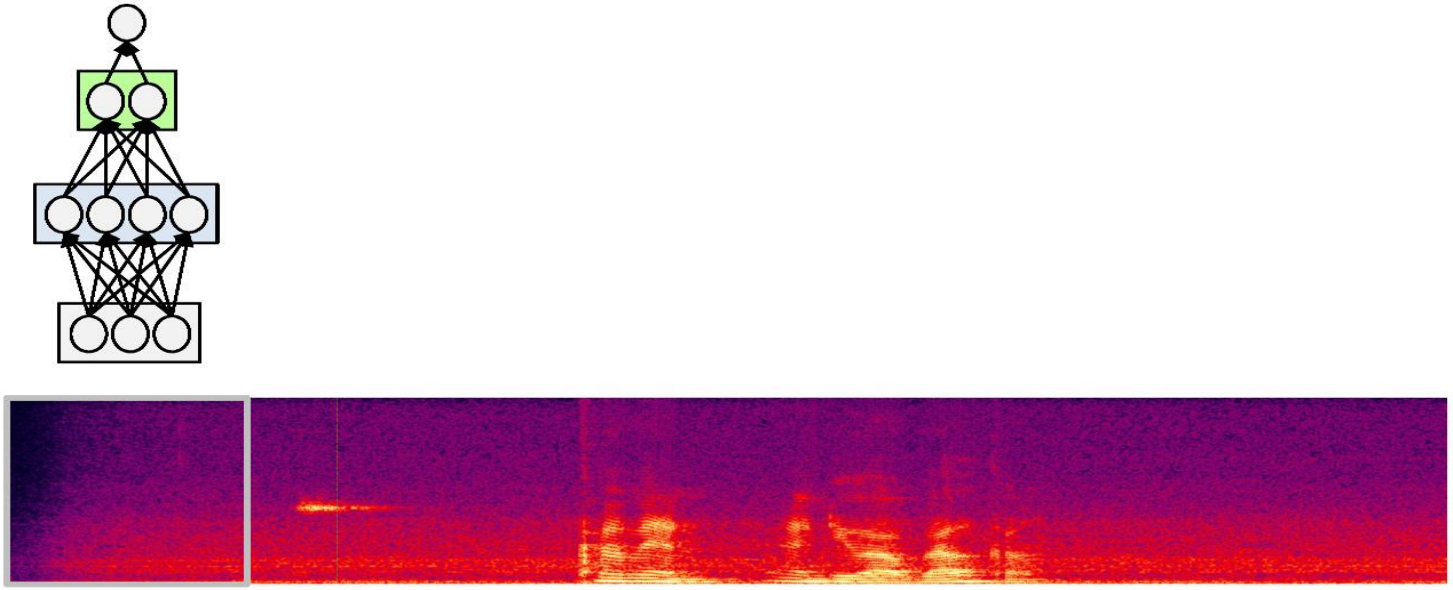
Fukushima, Kunihiko, and Sei Miyake. "Neocognitron: A self-organizing neural network mode
and cooperation in neural nets. Springer, Berlin, Heidelberg, 1982. 267-285.

1-D convolution



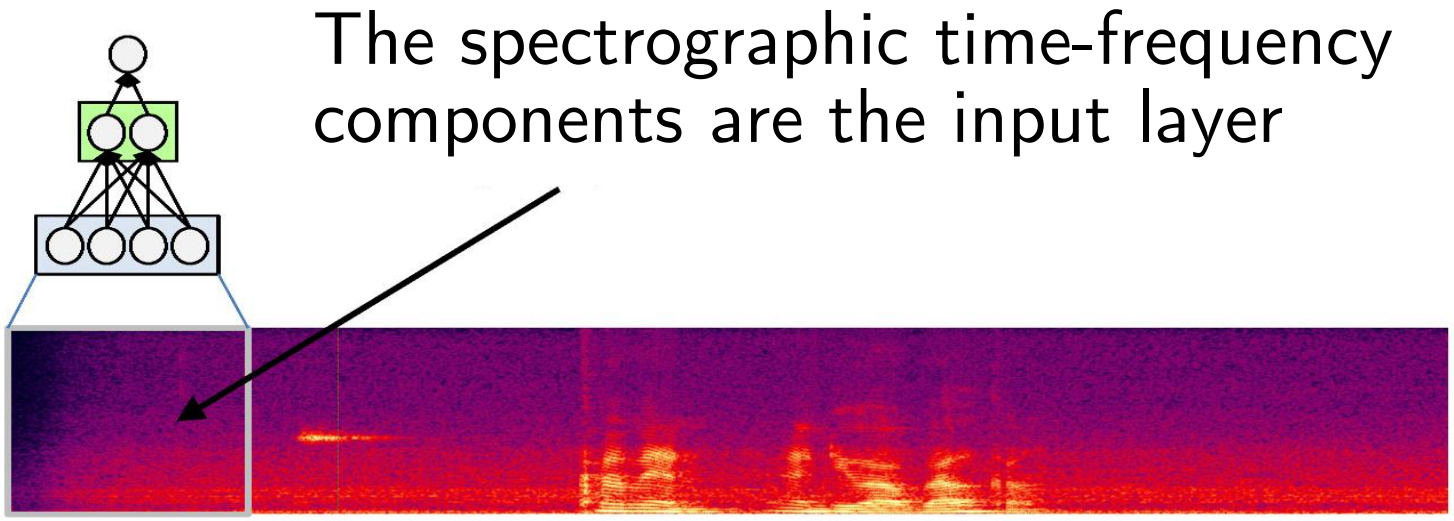
- The 1-D scan version of the convolutional neural network is the time-delay neural network
  - Used primarily for speech recognition

1-D convolution



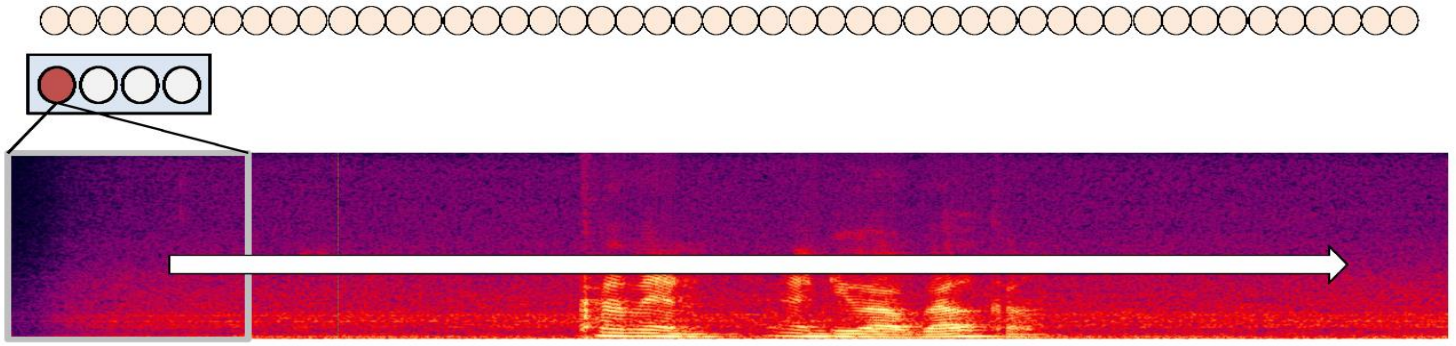
- The 1-D scan version of the convolutional neural network

1-D convolution



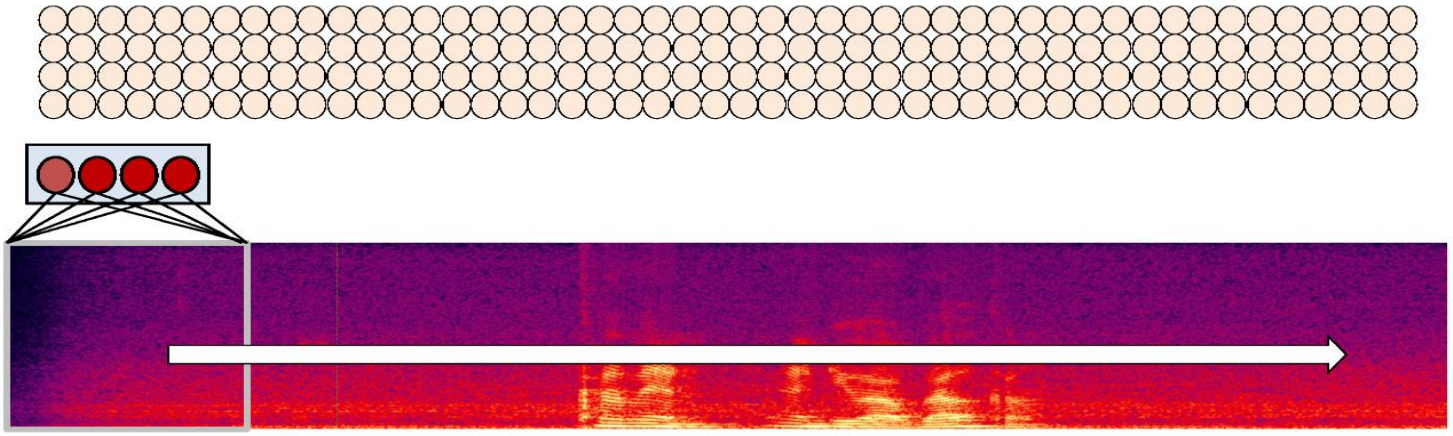
- The 1-D scan version of the convolutional neural network

1-D convolution



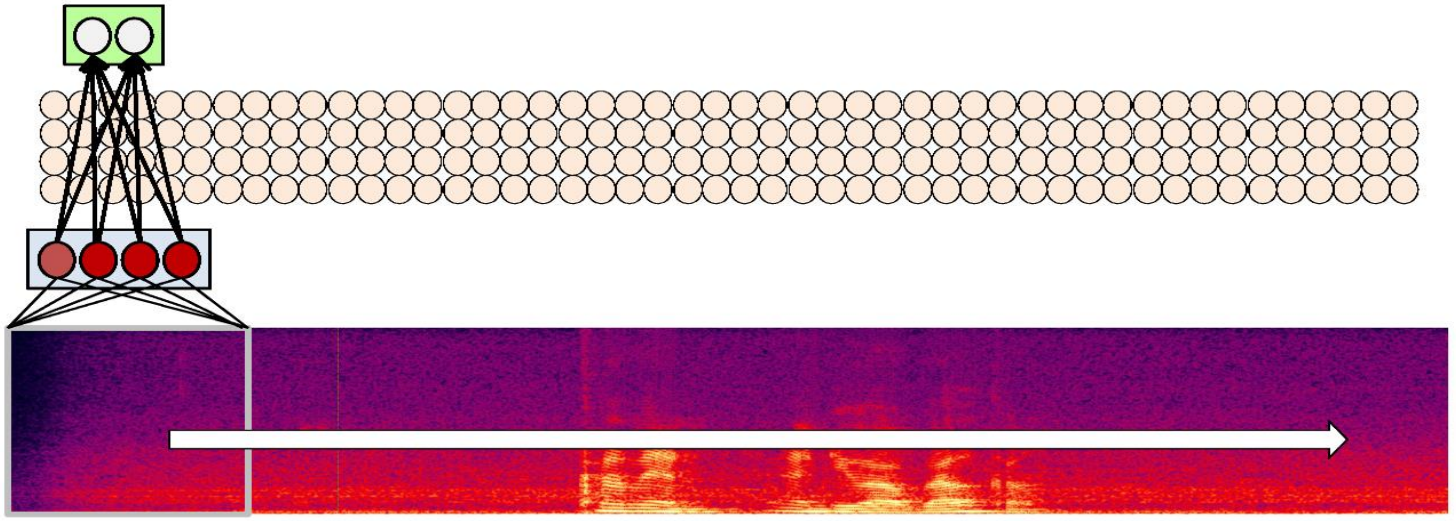
- The 1-D scan version of the convolutional neural network

1-D convolution



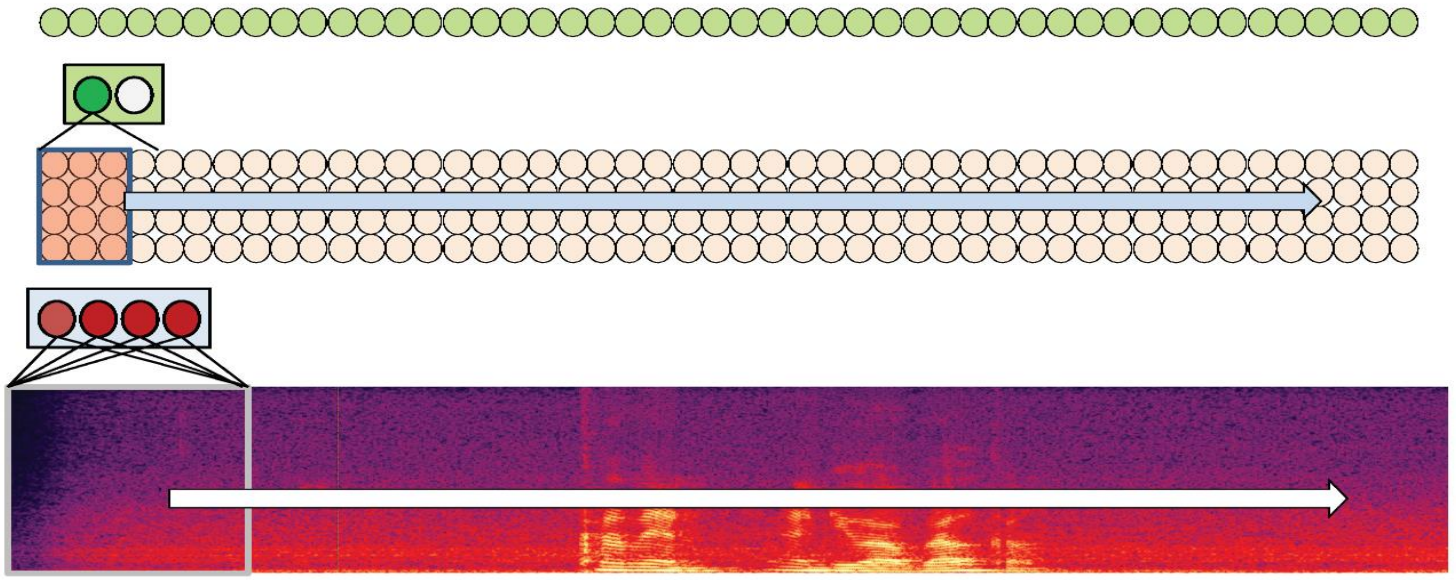
- The 1-D scan version of the convolutional neural network

1-D convolution



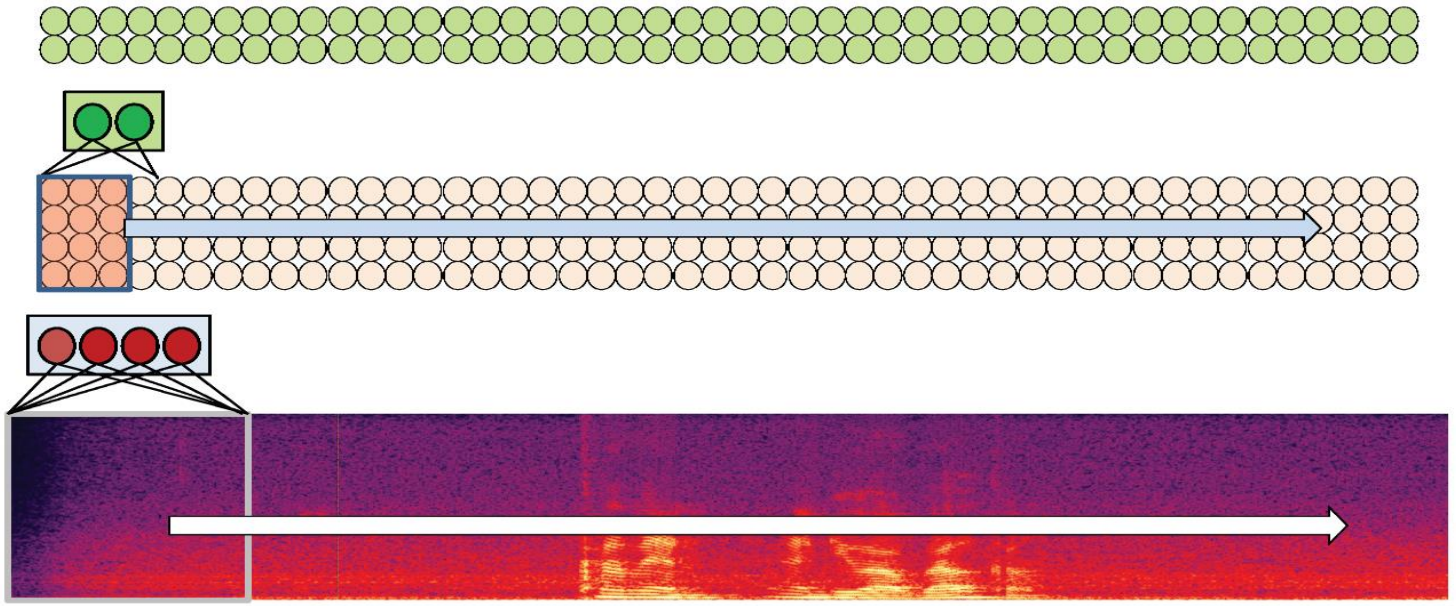
- The 1-D scan version of the convolutional neural network
  - Max pooling optional
    - Not generally done for speech

1-D convolution



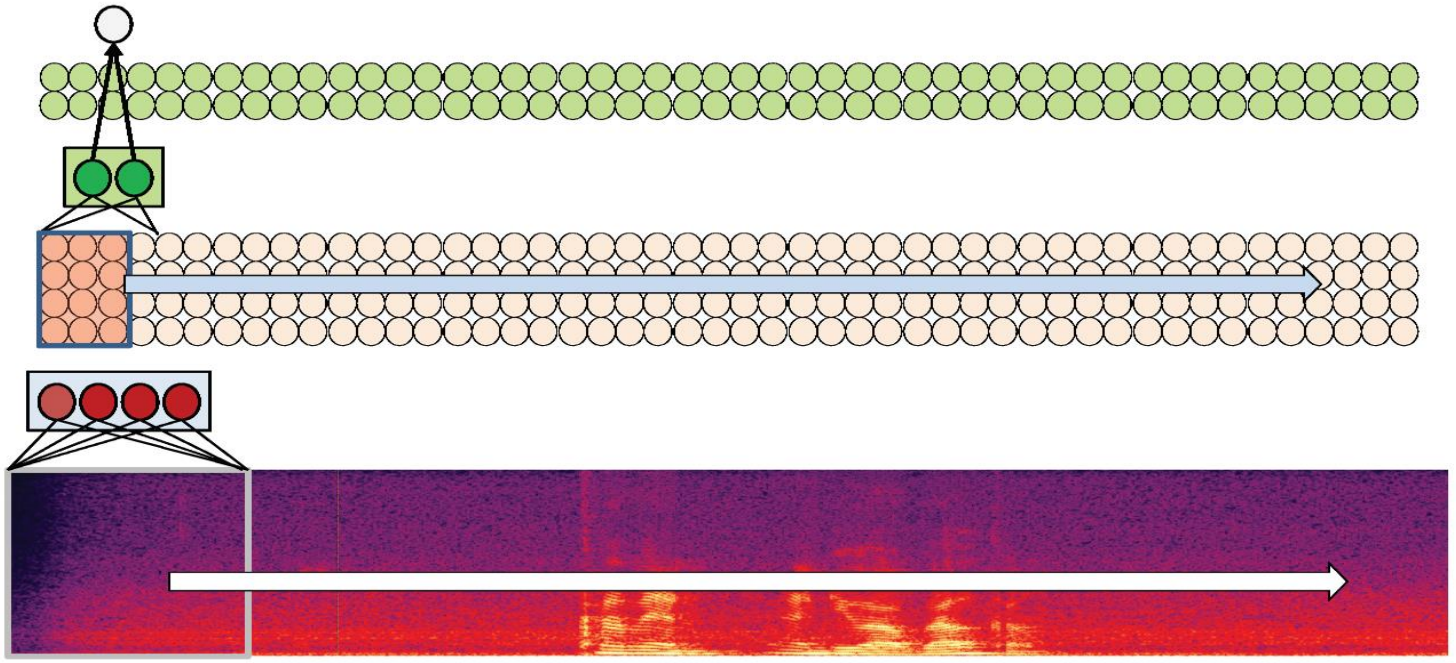
- The 1-D scan version of the convolutional neural network
  - Max pooling optional
    - Not generally done for speech

1-D convolution



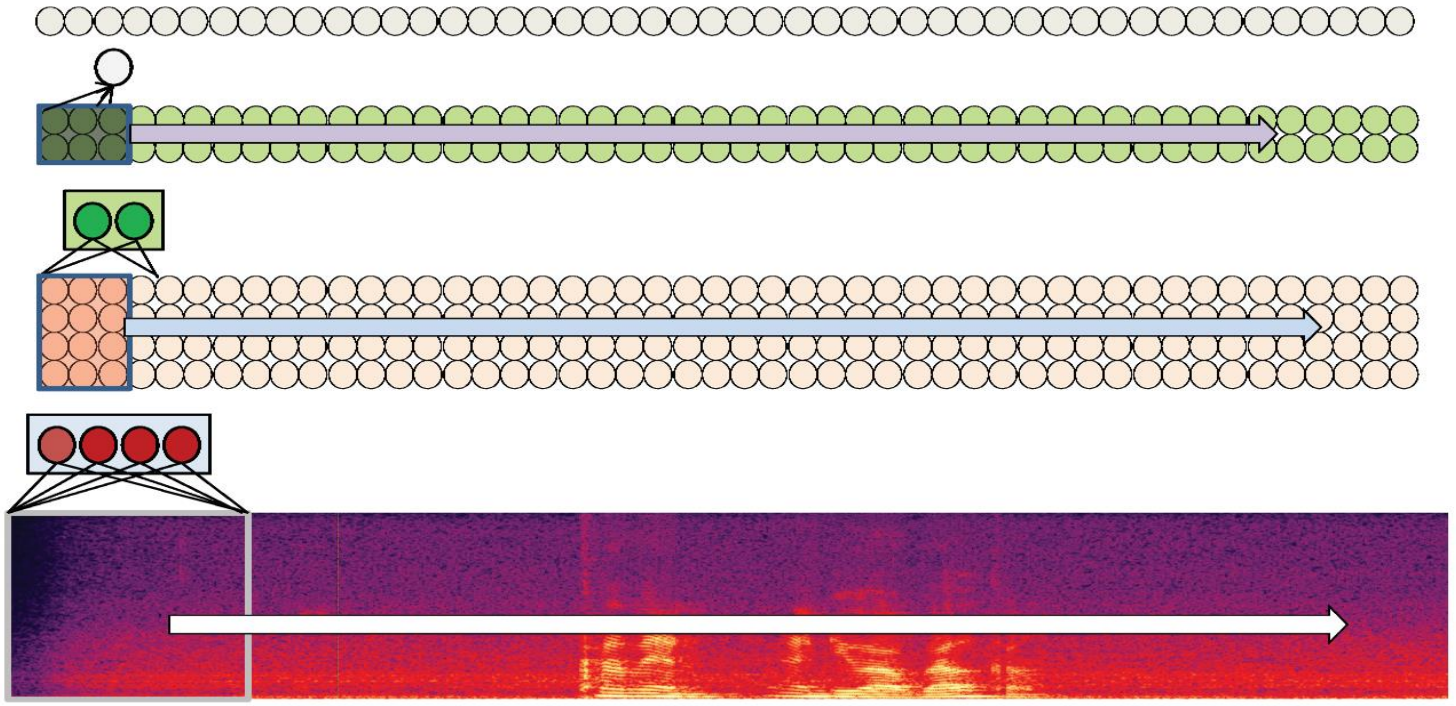
- The 1-D scan version of the convolutional neural network
  - Max pooling optional
    - Not generally done for speech

1-D convolution



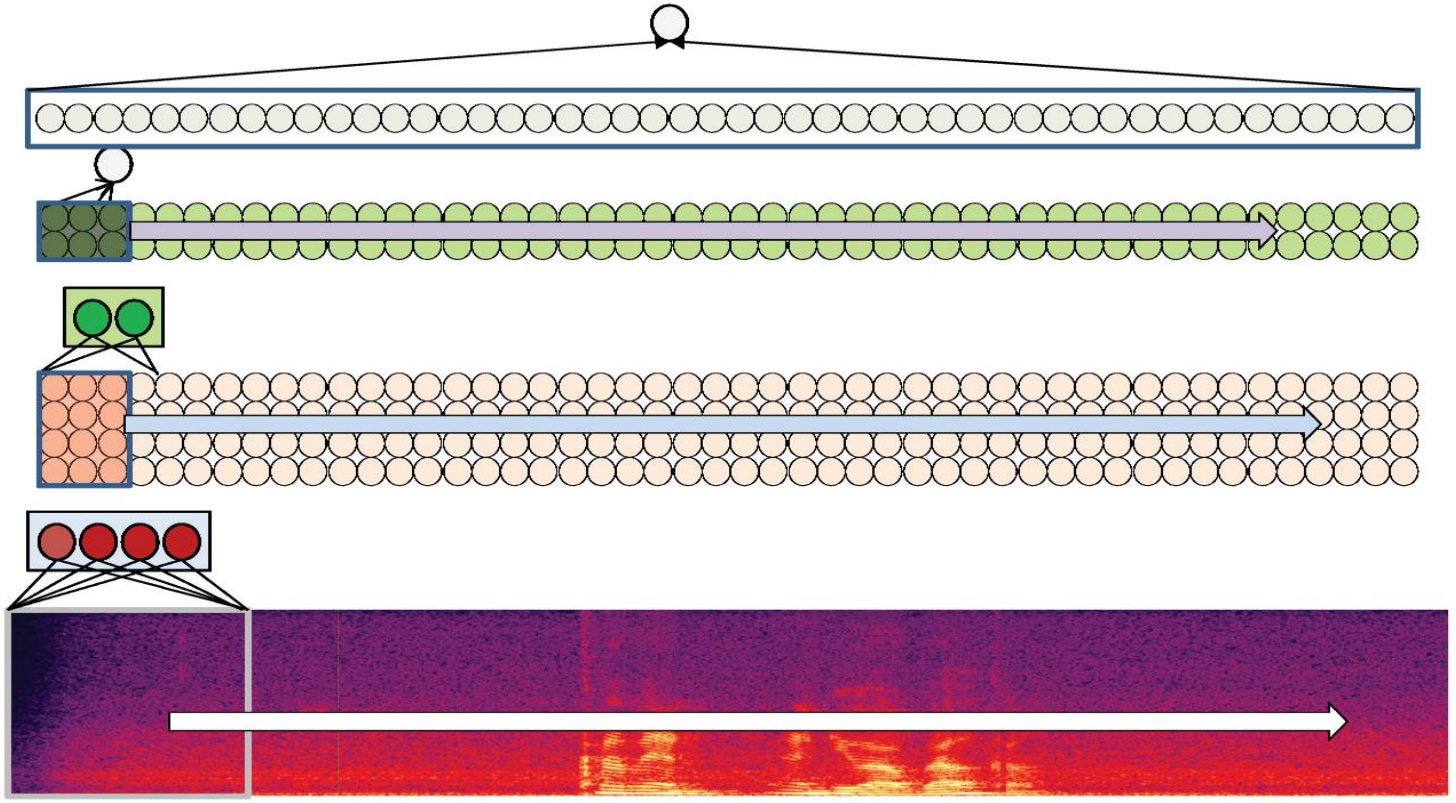
- The 1-D scan version of the convolutional neural network
  - Max pooling optional
    - Not generally done for speech

1-D convolution



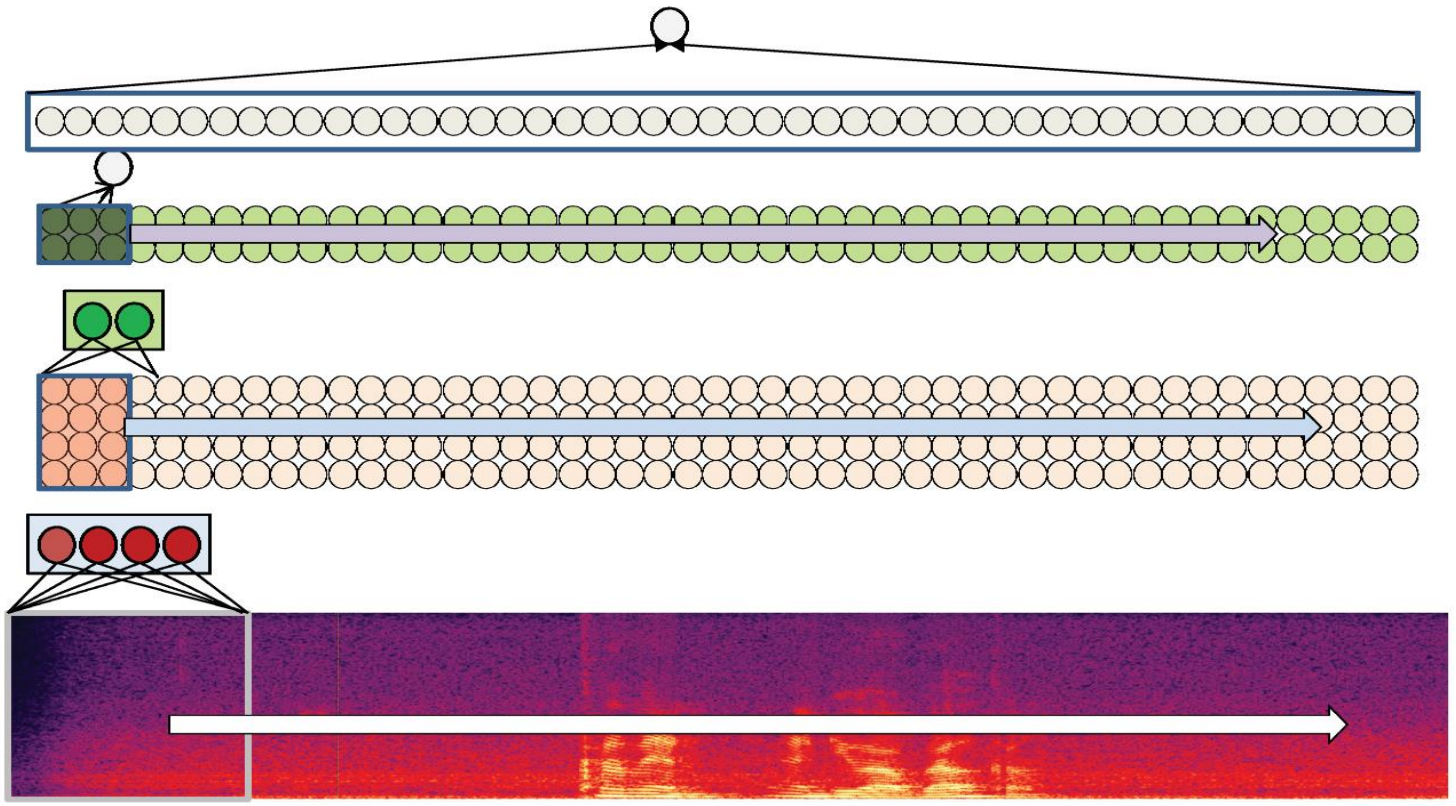
- The 1-D scan version of the convolutional neural network
  - Max pooling optional
    - Not generally done for speech

1-D convolution



- The 1-D scan version of the convolutional neural network
- A final perceptron (or MLP) to aggregate evidence
  - "Does this recording have the target word"

1-D convolution



- This structure is called the Time-Delay Neural Network