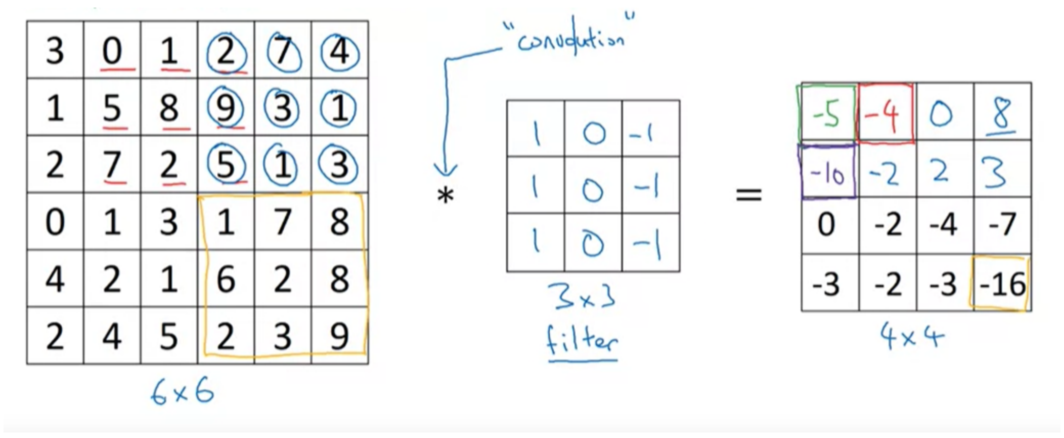
dl.ai_CONVOLUTION

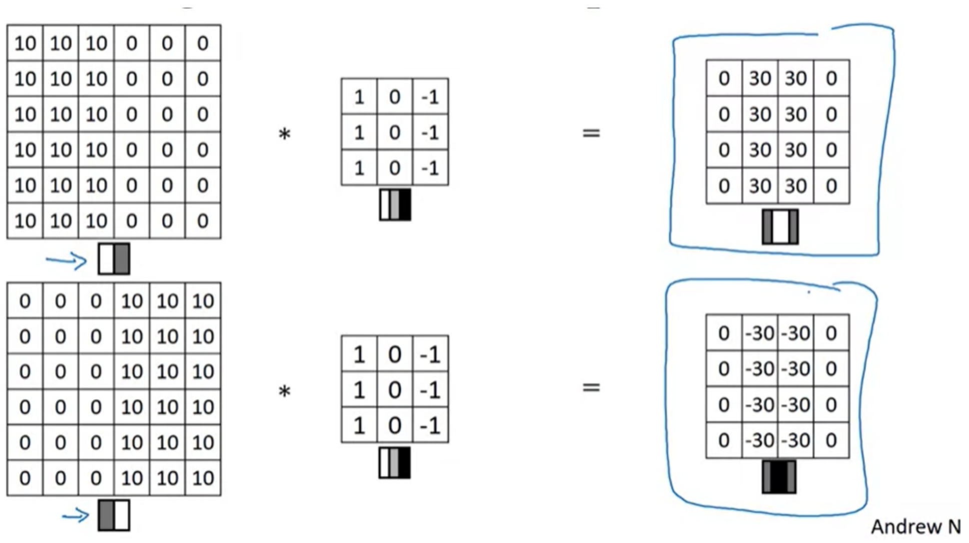
a)Filter convolution

- Note that the size of the matrix decreases when we convolute it



$$\underbrace{\begin{bmatrix}3&0&1&2&7&4\\1&5&8&9&3&1\\2&7&2&5&1&3\\0&1&3&1&7&8\\4&2&1&6&2&8\\2&4&5&2&3&9\end{bmatrix}}_{6\times 6} \overset{\text{"convolution"}}{*} \underbrace{\begin{bmatrix}1&0&-1\\1&0&-1\\1&0&-1\end{bmatrix}}_{\substack{3\times 3\\ \text{filter}}} = \underbrace{\begin{bmatrix}-5&-4&0&8\\-10&-2&2&3\\0&-2&-4&-7\\-3&-2&-3&-16\end{bmatrix}}_{4\times 4}$$

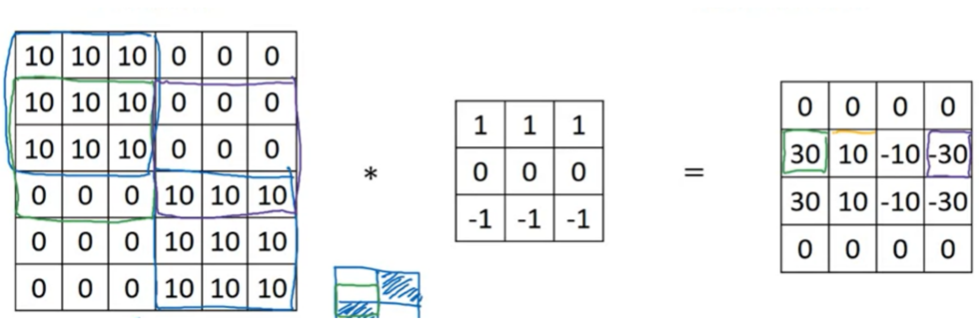
- The above filter is for vertical kernels as color change from left to right will cause a non zero value in the output.
- Zero represents grey colour the positive or negative value can either represent white or black(if you have only one channel)
- Hence we can detect two types of filter



$$\begin{bmatrix}10&10&10&0&0&0\\10&10&10&0&0&0\\10&10&10&0&0&0\\10&10&10&0&0&0\\10&10&10&0&0&0\\10&10&10&0&0&0\end{bmatrix} * \begin{bmatrix}1&0&-1\\1&0&-1\\1&0&-1\end{bmatrix} = \boxed{\begin{bmatrix}0&30&30&0\\0&30&30&0\\0&30&30&0\\0&30&30&0\end{bmatrix}}$$

$$\begin{bmatrix}0&0&0&10&10&10\\0&0&0&10&10&10\\0&0&0&10&10&10\\0&0&0&10&10&10\\0&0&0&10&10&10\\0&0&0&10&10&10\end{bmatrix} * \begin{bmatrix}1&0&-1\\1&0&-1\\1&0&-1\end{bmatrix} = \boxed{\begin{bmatrix}0&-30&-30&0\\0&-30&-30&0\\0&-30&-30&0\\0&-30&-30&0\end{bmatrix}}$$

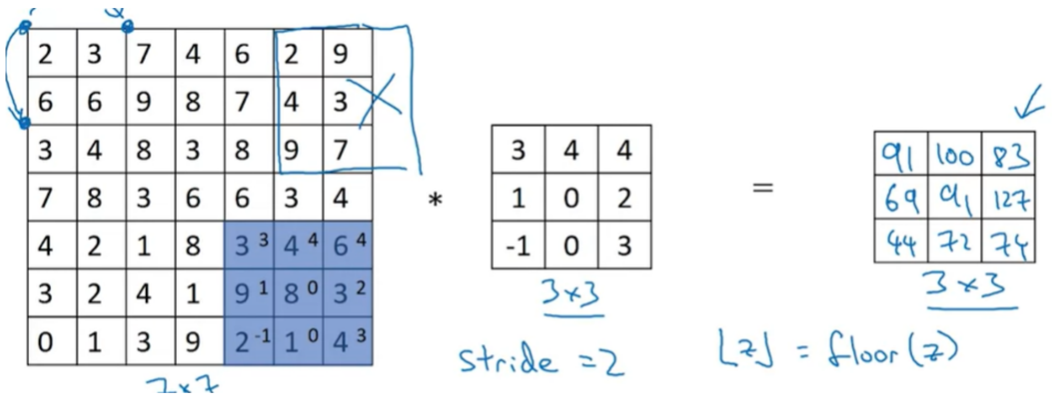
- In more complex inputs we will have multiple nonzero values with intensity depicting how good border is



- Sobel filter gives more weightage to the middle values (pixels closer to the mask)leading to a more finer boder especially for cases involving a diagonal line.

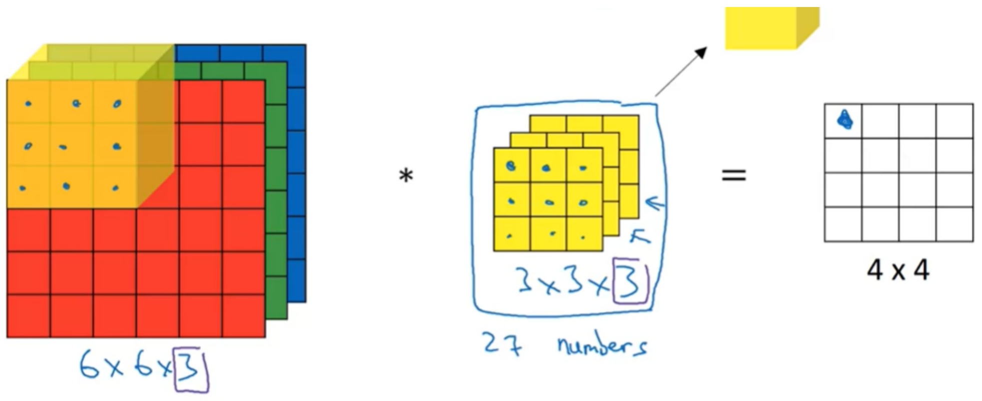
b) Padding and Striding

- We notice that the matrix shrinks on undergoing a convolution.We can avoid this by adding padding to the original matrix.
- In valid convolution where we dont use any padding we would end up with a n-k+1 output matrix.
- In same convolution where we use a padding of p=(k-1)/2 to get same size n for output matrix
- In striding instead of moving 1 step right or down when doing convolution we move s steps.The size of the output matrix is given by $\frac{n+2p-k}{s} + 1 * \frac{n+2p-k}{s} + 1$

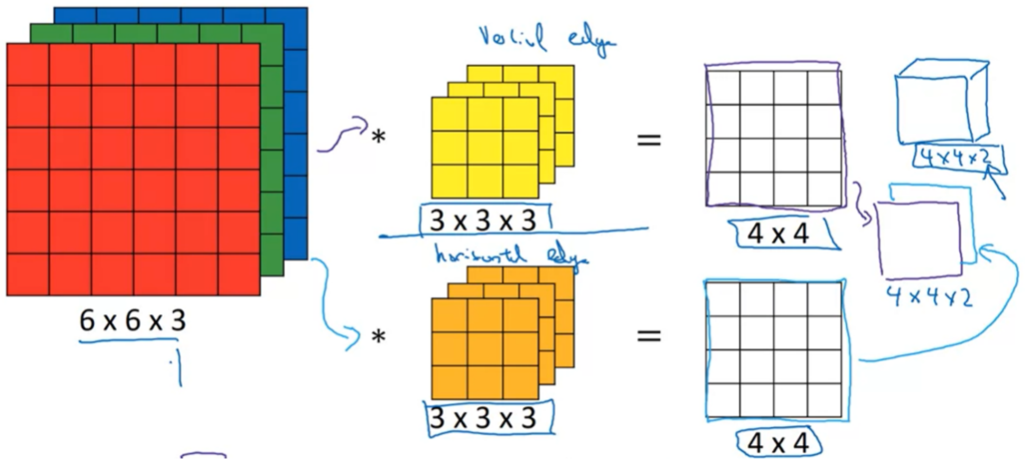


c)Convolution over volume

- if our input image has 3 channels we need to covolute over a volume rather than an area
- We simply define seperate 2d filter for each channel independent of each other,and paralize the convolution over a given area across three channels.
- we wont convolve in the z direction.
- Note that the number of channels in kernel and input must be same. However the output matrix can be 2d or 3d

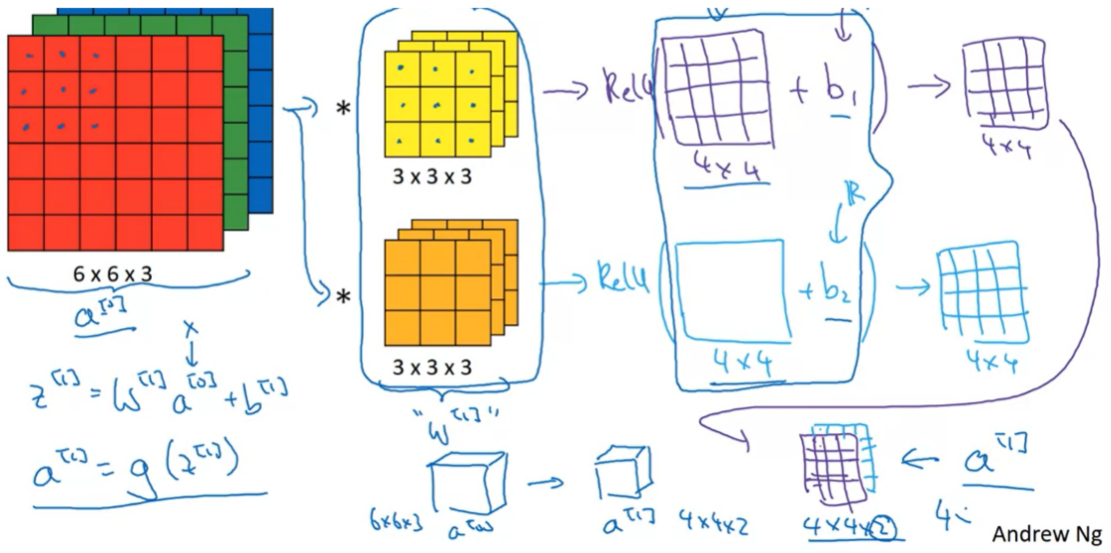


- We can use multiple filters and stack the results

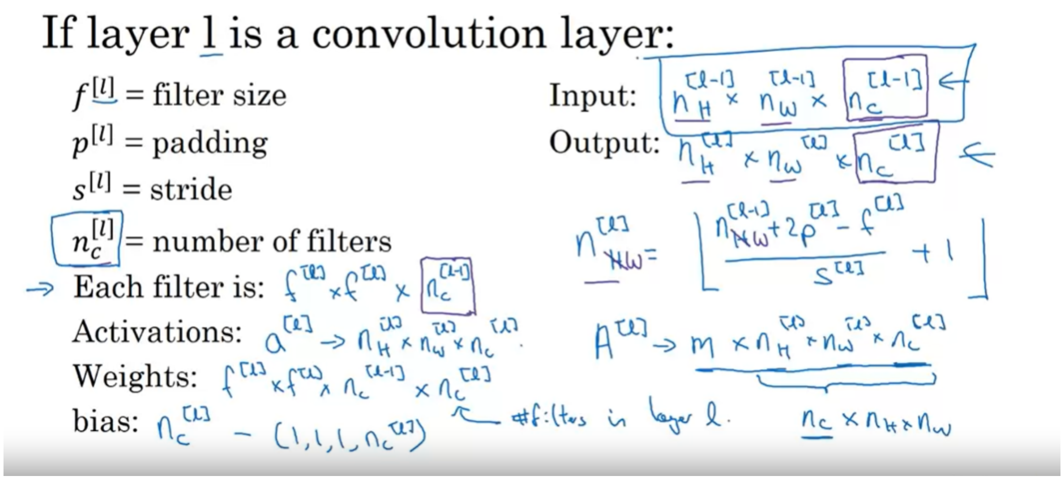


d) Convolutional layer

- In a convolutional layer we simply take a h*w*c input (like X in ann) we convolute it with multiple filters(like $W^{[l]}$ in ann,with each filter like $W^{[l](i)}$)we add an bias to the convoluted result and apply relu over it,get the result and stack them to get our output (like $A^{[l]}$ in ann,with each matrix like $A^{[l](i)}$)



- Note that the number of channels in $A^{[l]}$ depend on the number of filters in the layer which produces it.If you assume that 2d matrixes can always undergo convolution to be just like numbers which can always undergo multiplication.We can simply neglect the dimensions of the 2d array and say 1×3 ∗ 3∗10 =1×10, we can see that weight layer has dimension (no of filters is previous layer)x(no of filters in current layer).
- THe number of parameters in this layer is 3∗3∗3∗10 + 10=280, 3∗3∗3 is the no of weight parameters(pixels) in each filter,1 is the bias parameter which is common for all pixels in a filter.



If layer l is a convolution layer:

$f^{[l]}$ = filter size

$p^{[l]}$ = padding

$s^{[l]}$ = stride

$\boxed{n_c^{[l]}}$ = number of filters

→ Each filter is: $f^{[l]} \times f^{[l]} \times \boxed{n_c^{[l-1]}}$

Activations: $a^{[l]} \to n_H^{[l]} \times n_W^{[l]} \times n_c^{[l]}$

Weights: $f^{[l]} \times f^{[l]} \times n_c^{[l-1]} \times n_c^{[l]}$

bias: $n_c^{[l]}$ — $(1,1,1,n_c^{[l]})$ ← #filters in layer l.

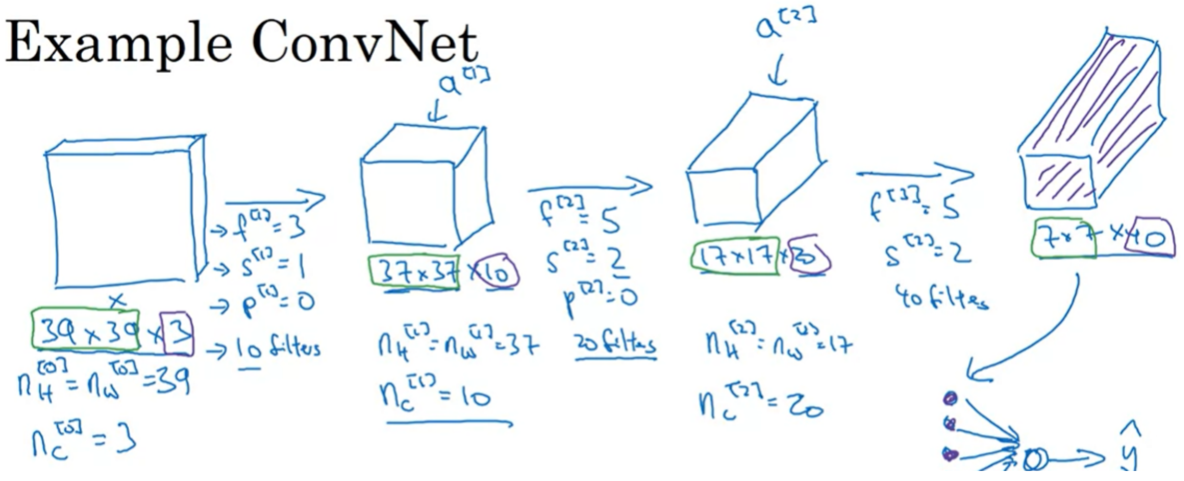
Input: $n_H^{[l-1]} \times n_W^{[l-1]} \times \boxed{n_c^{[l-1]}}$ ←

Output: $n_H^{[l]} \times n_W^{[l]} \times \boxed{n_c^{[l]}}$ ←

$$n_{H/W}^{[l]} = \left\lfloor \frac{n_{H/W}^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1 \right\rfloor$$

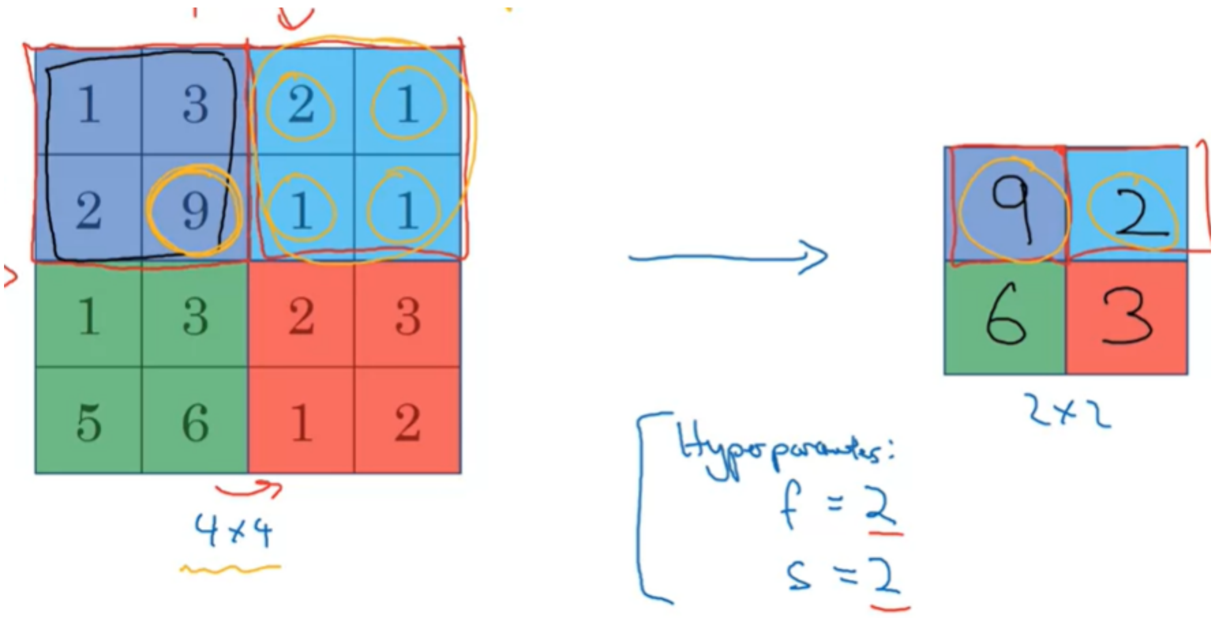
$A^{[l]} \to m \times n_H^{[l]} \times n_W^{[l]} \times n_c^{[l]}$

$n_c \times n_H \times n_W$

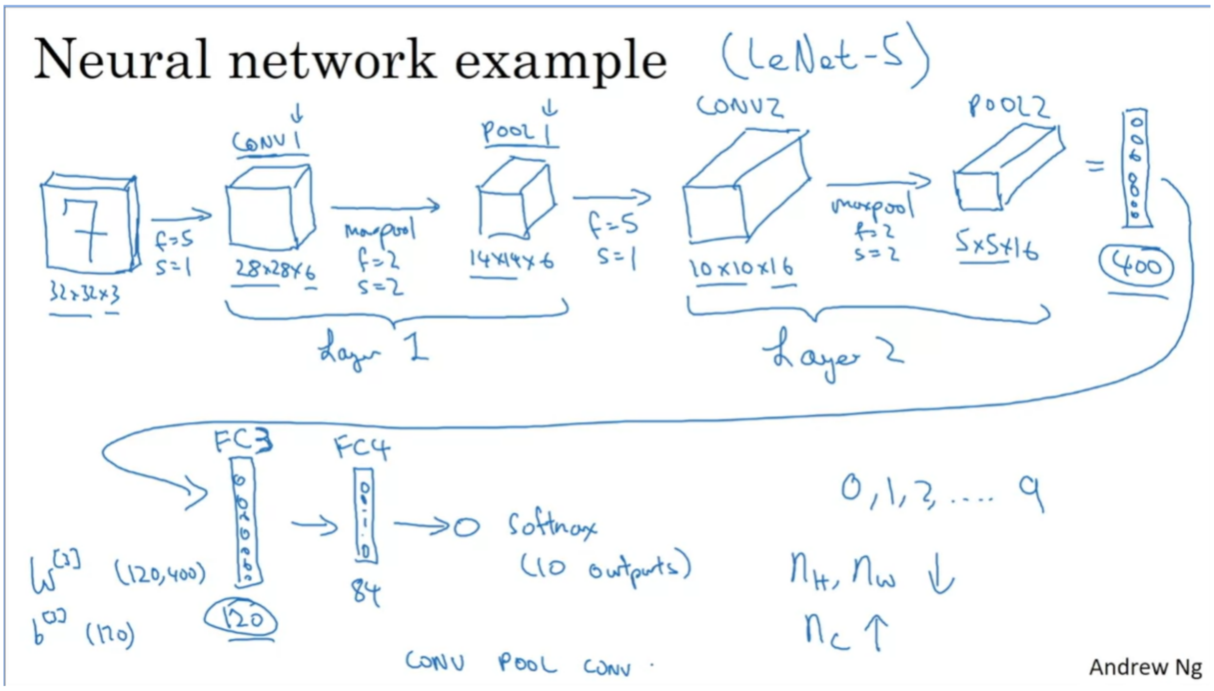


e) Pooling layer

- In max pooling we downsample a matrix to half its size by using a max-filter with size 2 and stride 2 which gives us a smaller matrix with only the maximum values.



- Maxpooling reduces the spatial dimensions of the feature maps while retaining the most important information. By reducing the size of feature maps, it speeds up the training and inference of CNNs, and it helps prevent overfitting, which can occur when the network has too many parameters for the available data.
- Invariance-if a particular feature (e.g., an edge or a texture) is detected in a region of the input, maxpooling will preserve the detection even if the feature shifts slightly within that region. This is because maxpooling selects the maximum value within a local region, so as long as the feature is present in that region, it will be captured.
- There is no learnable parameters in a max pooling layer
- In average pooling we just take the average of the 2×2 slot .It is used only when we need to convert the 2d * Nc layer to 1* Nc layer.



f) Why?

- The main reason we call have a seperate fully connected layer is because conv layer is not fully connected if all the pixels in our input image are mapped to all the pixels in our output image we will have lot of parameters.
- Parameter Sharing: Any given filter can be used across all the pixels in the input image,like if the filter is 3∗3 and the input image is 5∗5 we can use that filter for 9 times.
- A filter like vertical edge detector can be used throughout the image.
- Sparsity of connections: each pixel in the output layer is mapped to only a small number of the input pixels, Like an pixel in the top right corner of the output doesn't depend on pixel on the pixel to bottom left corner of image. Just say f*f filters don't complicate.