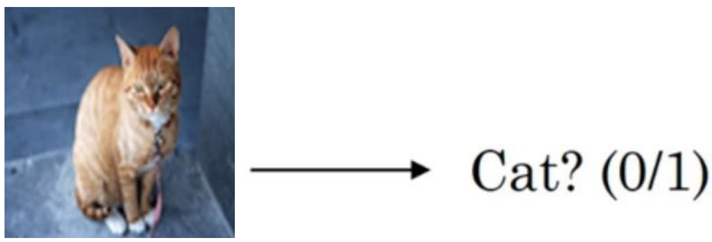
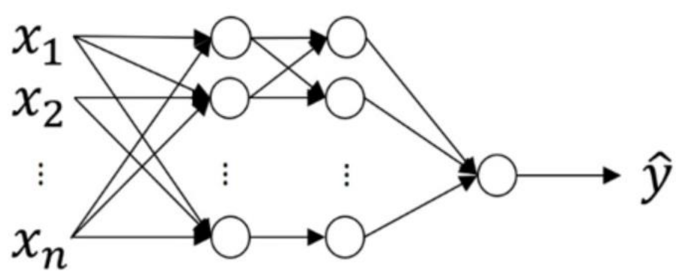
Convolutional Neural Networks (CNN)

Motivation of CNN:

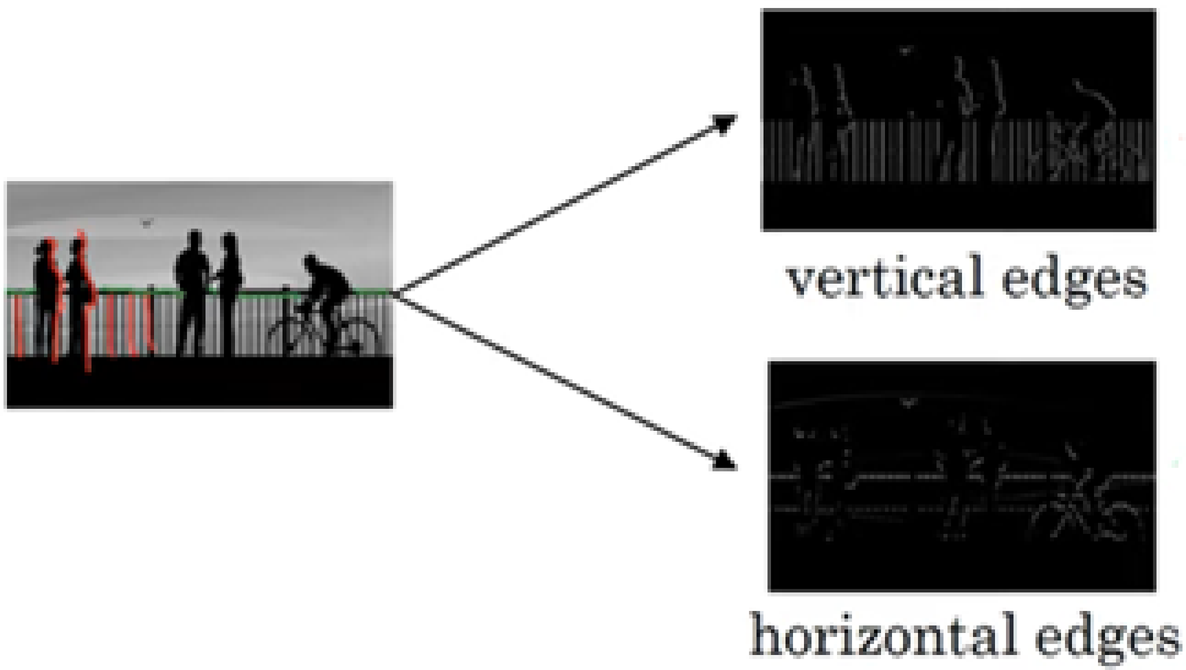
- **A big challenge of computer vision problem:** One of the challenges of computer vision problems is that the inputs can get really big. Deep learning on large images: consider the following LR (64x64x3=12288) and HR (1000x1000x3 = 3M)





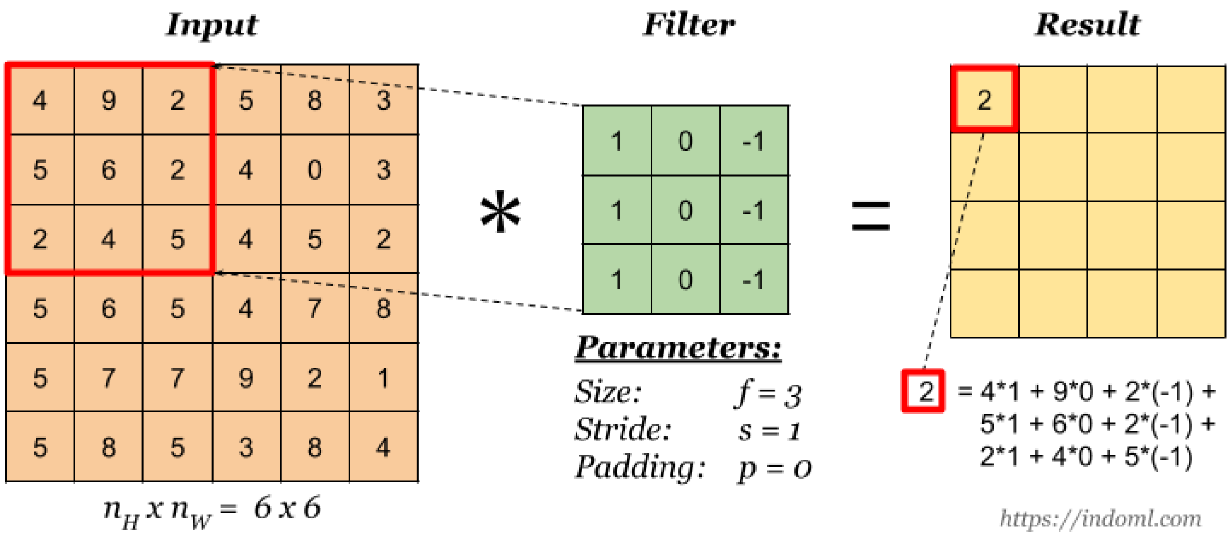
- In NN with input of HR image (3M), say hidden layer has 1000 neuron, it means weight matrix size is (1000 x3M = 3Billion) = > very larger number parameters.
- Issue with NN due to larger image size:
    - Overfitting due to higher no of parameters
    - Computational and memory requirements are higher
- But for CV application we need to use larger images
- Solution: Use **convolutional neural networks.**

- **Convolution Operation: Conv operation** detects the different types of features at hidden layers of the CNN. **Earlier layers find the edges while the later layers find the complex objects.** Feature detection: edge, corner, smaller objects, and complete objects detection

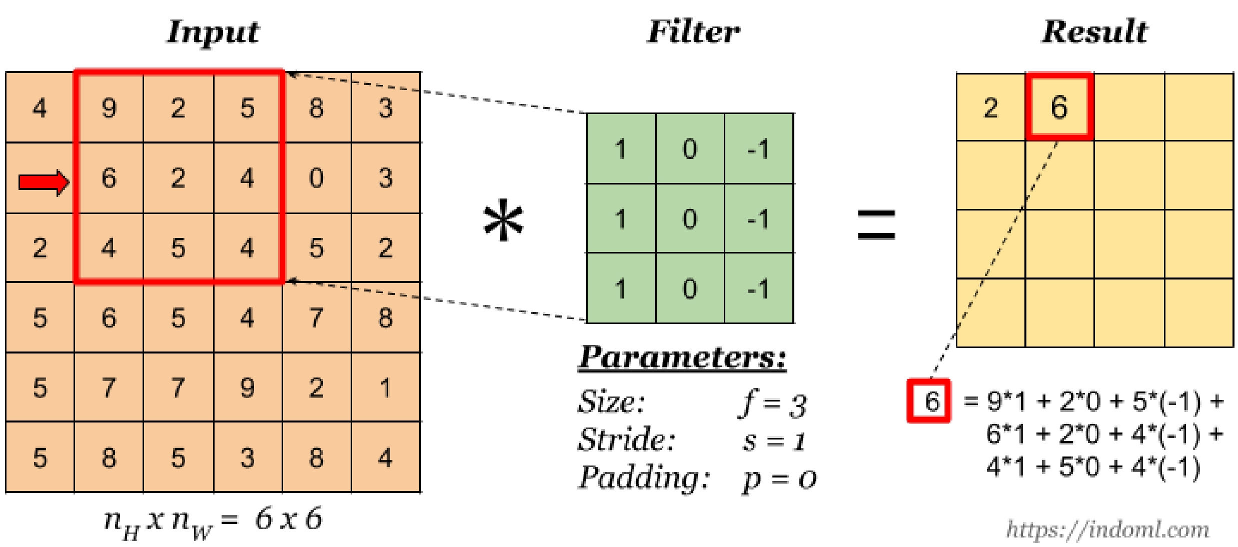


- Edge is detected at a location in the image where is sudden changes in the intensity/colors of pixels. It is a location where transition between objects or object and background.
- If the intensity values on input image are constant, then output of convolution operation will be zero.

- **Convolution operation:** Vertical and horizontal edge detection.
    - Vertical Edge detector:

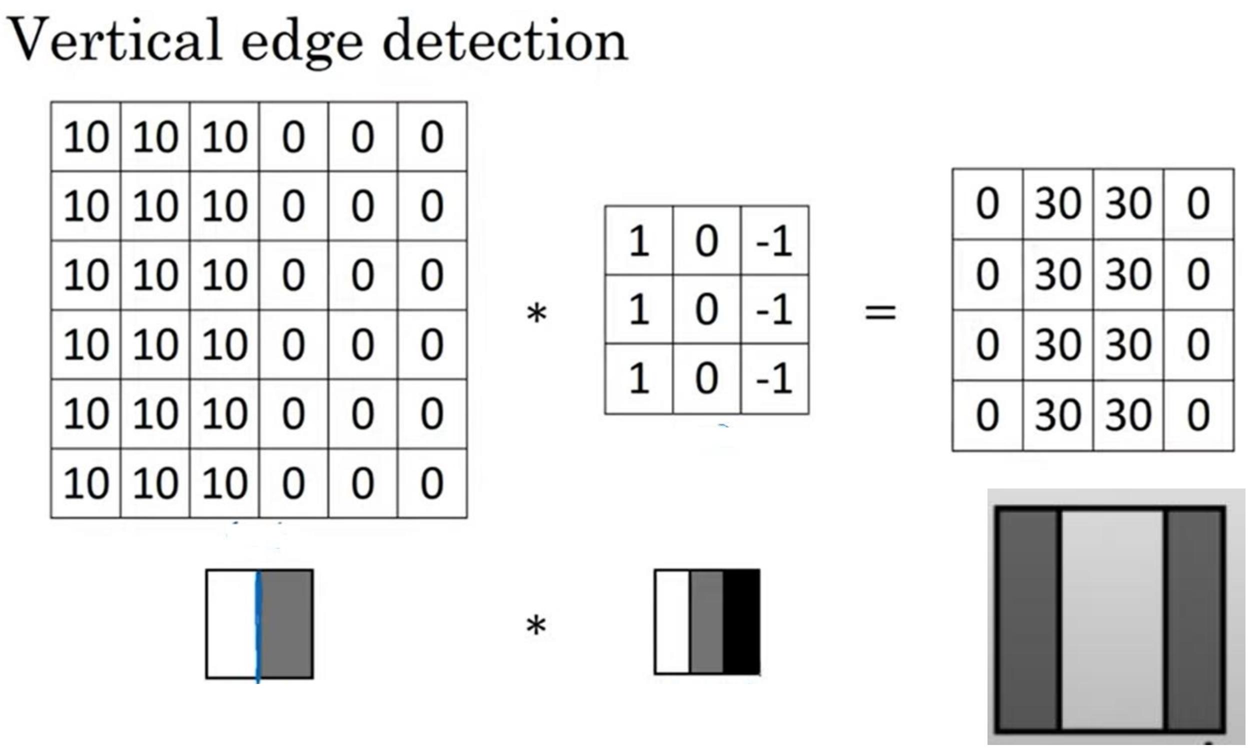


- For next step move the overlay right one position (or according to the **stride** setting) and do the same calculation above to get the next result. And so on.



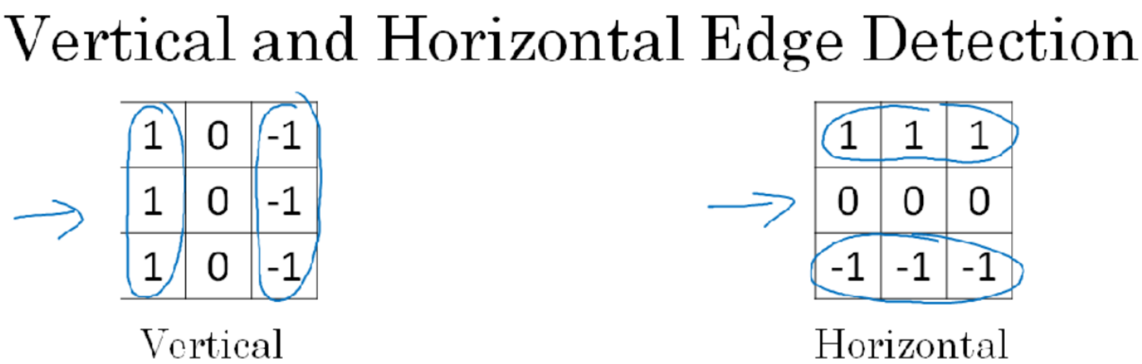
- **Fill the empty cells by performing convolution operations? [homework#1]**

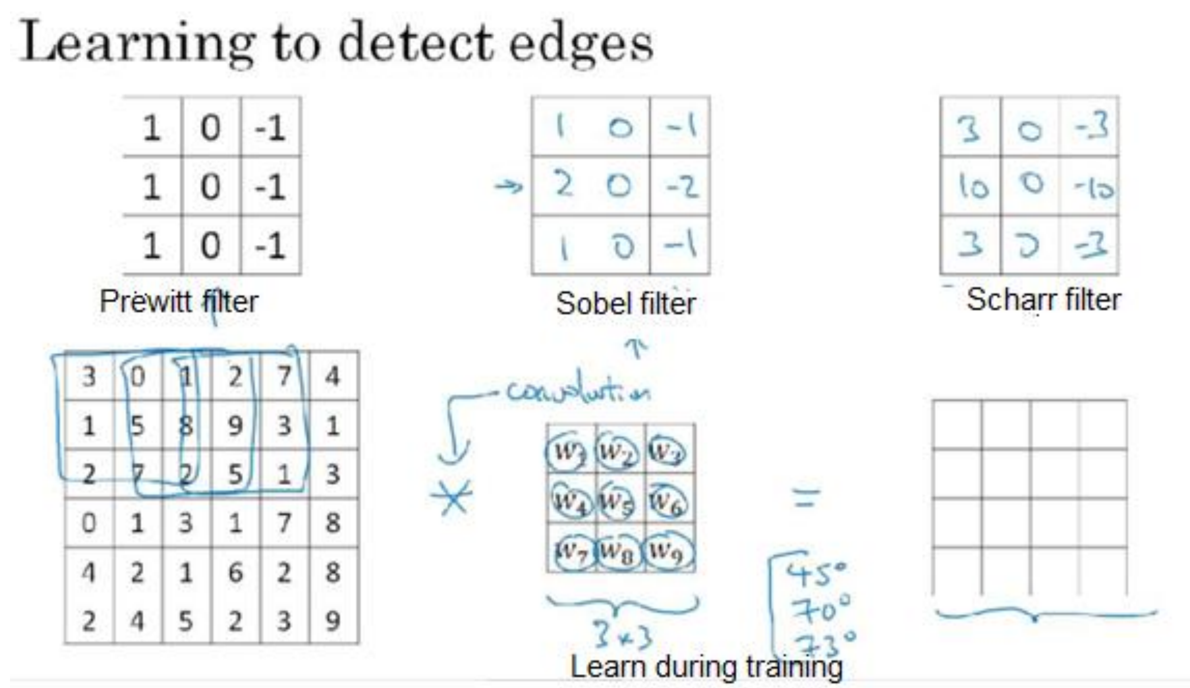
- Let consider another example to see how convolution operation helps to identify the vertical edges more clearly.



Observations:

- Two columns edge is seemed to be a very thick, as it is for 4x4 image. But for 1000x1000 image it would be look like a thin edge.
- Edges are detected at sudden change in intensities.
- Intensity value of output image is the outcome of 3x3 region of input image **(spare connection)**
- Same filter is used to detect the vertical edge at different 3x3 region of the input images **(parameter/filter sharing)**

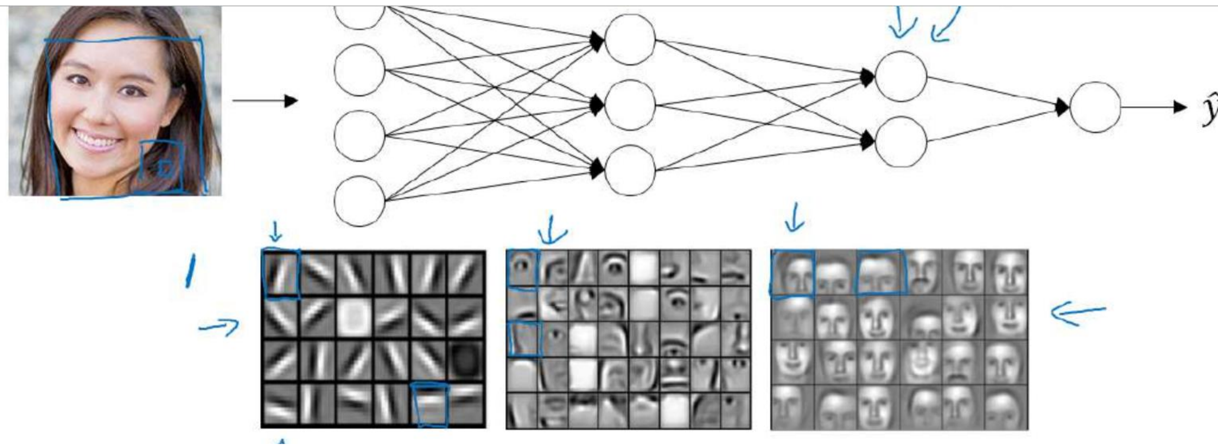




- Prewitt, Sobel, and Scharr filters are examples of edge detector
- **In CNN, the weights (parameters) of filters/kernels of different orientations (vertical, horizontal, 45, 90, 70 degrees, etc.) are learned during the training process.**
- torch.nn.conv2d**()**

- **Demo and Intuition of Convolution => Dr Rawat slides**
  - **Conv operation**
  - **High and low activation values**

5. Face recognition problem:



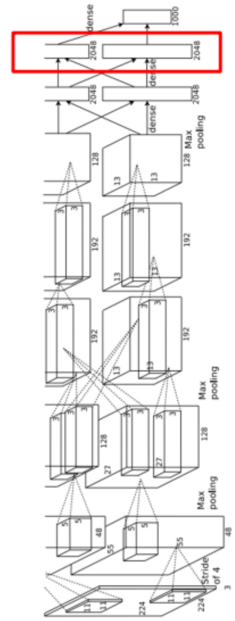
Beneath the face, there is a series of grids showing progressively more abstract features being extracted at different layers of the neural network:

- **First Layer:** Detects low-level features like edges, corners, and textures.
- **Second Layer:** Combines these to detect more complex patterns like shapes and parts of the face (e.g., eyes, nose, mouth).
- **Third Layer and Beyond:** Builds a high-level representation of the entire face.

6. Last layer features of AlexNet: example of CNN

Last Layer

FC7 layer



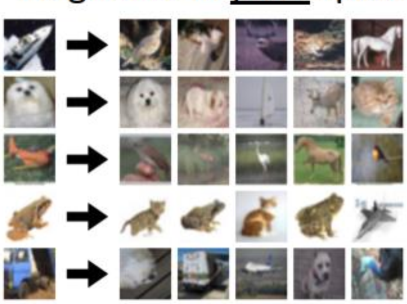
4096-dimensional feature vector for an image (layer immediately before the classifier)

Run the network on many images, collect the feature vectors

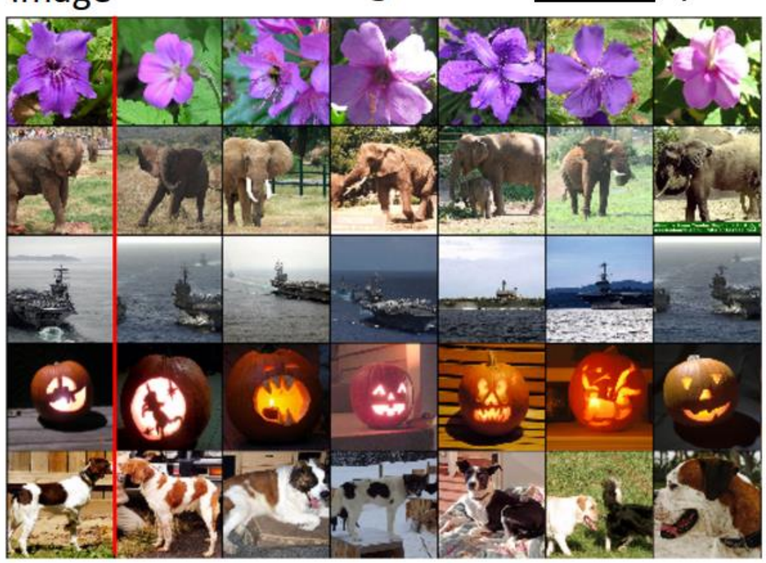
Krizhevsky et al, "ImageNet Classification with Deep Convolutional Neural Networks", NeurIPS 2012.
Figures reproduced with permission.

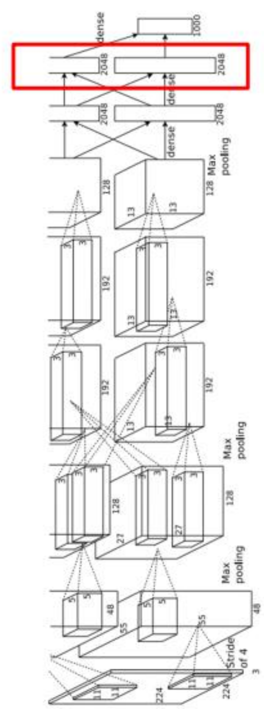
Last Layer: Nearest Neighbors

Recall: Nearest neighbors in pixel space



Test image L2 Nearest neighbors in feature space



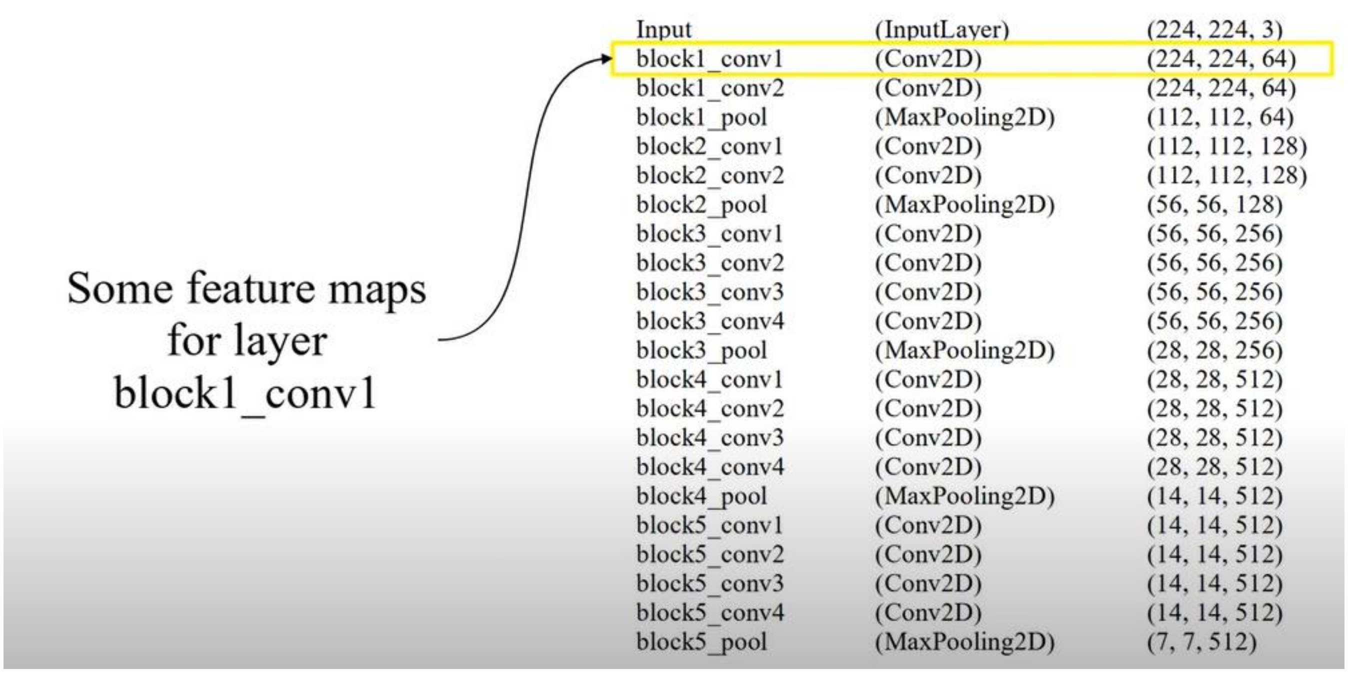


Krizhevsky et al, "ImageNet Classification with Deep Convolutional Neural Networks", NeurIPS 2012.
Figures reproduced with permission.

Pixel Space vs. Feature Space:

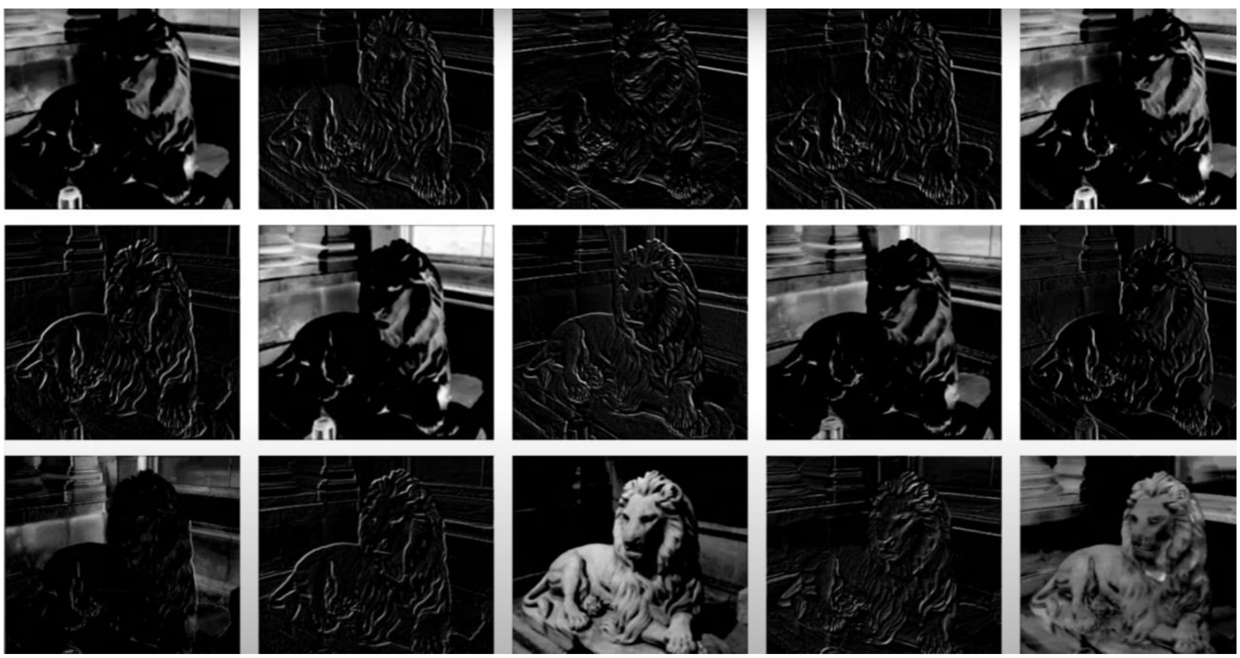
- **Pixel Space:** Nearest neighbors might look similar at the raw level but are often unrelated semantically.
- **Feature Space:** Nearest neighbors are based on semantic meaning, thanks to the network's learned representations.

7. Hidden layers of VGG-16 : example of CNN

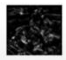


Input	(InputLayer)	(224, 224, 3)
block1_conv1	(Conv2D)	(224, 224, 64)
block1_conv2	(Conv2D)	(224, 224, 64)
block1_pool	(MaxPooling2D)	(112, 112, 64)
block2_conv1	(Conv2D)	(112, 112, 128)
block2_conv2	(Conv2D)	(112, 112, 128)
block2_pool	(MaxPooling2D)	(56, 56, 128)
block3_conv1	(Conv2D)	(56, 56, 256)
block3_conv2	(Conv2D)	(56, 56, 256)
block3_conv3	(Conv2D)	(56, 56, 256)
block3_conv4	(Conv2D)	(56, 56, 256)
block3_pool	(MaxPooling2D)	(28, 28, 256)
block4_conv1	(Conv2D)	(28, 28, 512)
block4_conv2	(Conv2D)	(28, 28, 512)
block4_conv3	(Conv2D)	(28, 28, 512)
block4_conv4	(Conv2D)	(28, 28, 512)
block4_pool	(MaxPooling2D)	(14, 14, 512)
block5_conv1	(Conv2D)	(14, 14, 512)
block5_conv2	(Conv2D)	(14, 14, 512)
block5_conv3	(Conv2D)	(14, 14, 512)
block5_conv4	(Conv2D)	(14, 14, 512)
block5_pool	(MaxPooling2D)	(7, 7, 512)


To exit full screen, press Esc



Even if a low resolution feature map is resized,
the result is hard to interpret



This is the situation for layer block3_conv1

Think about higher layers that have even lower spatial resolutions

Observing feature maps directly

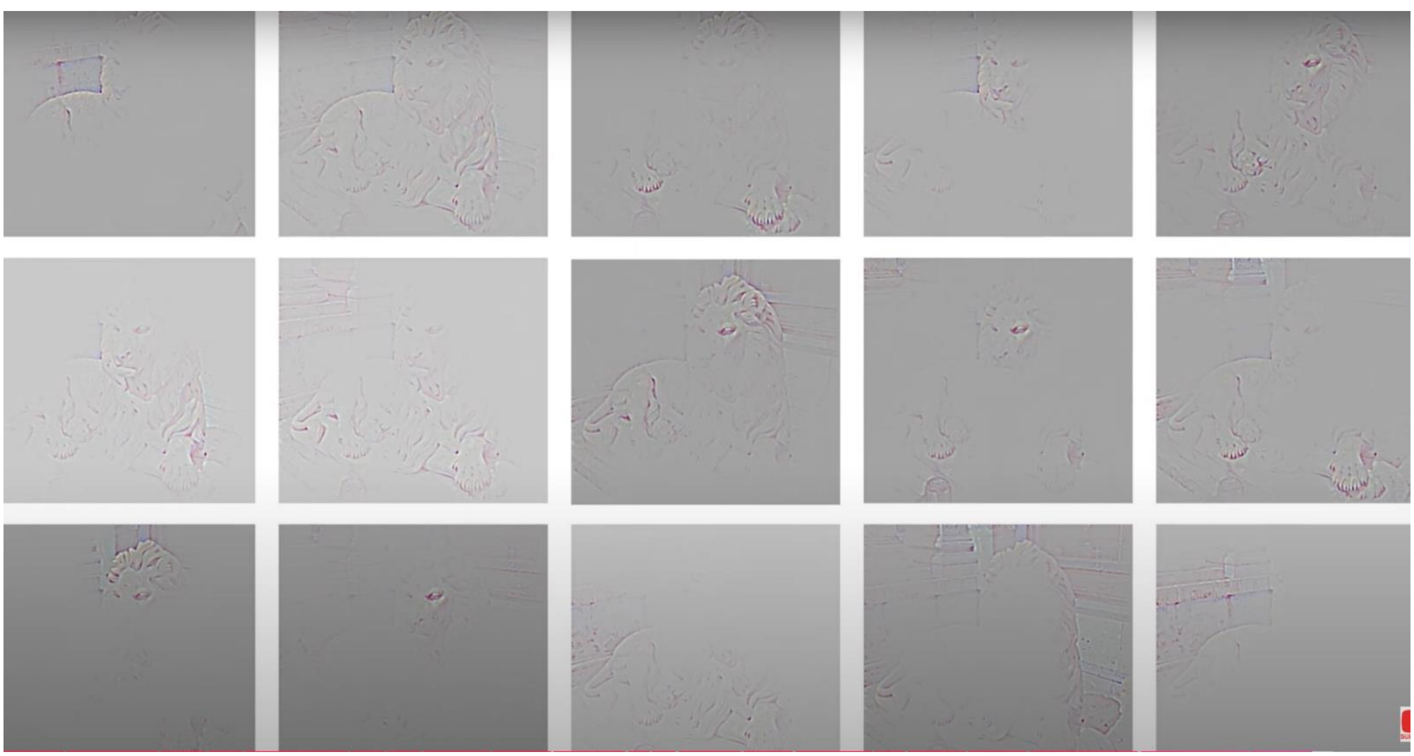
does not provide the desired visual representation

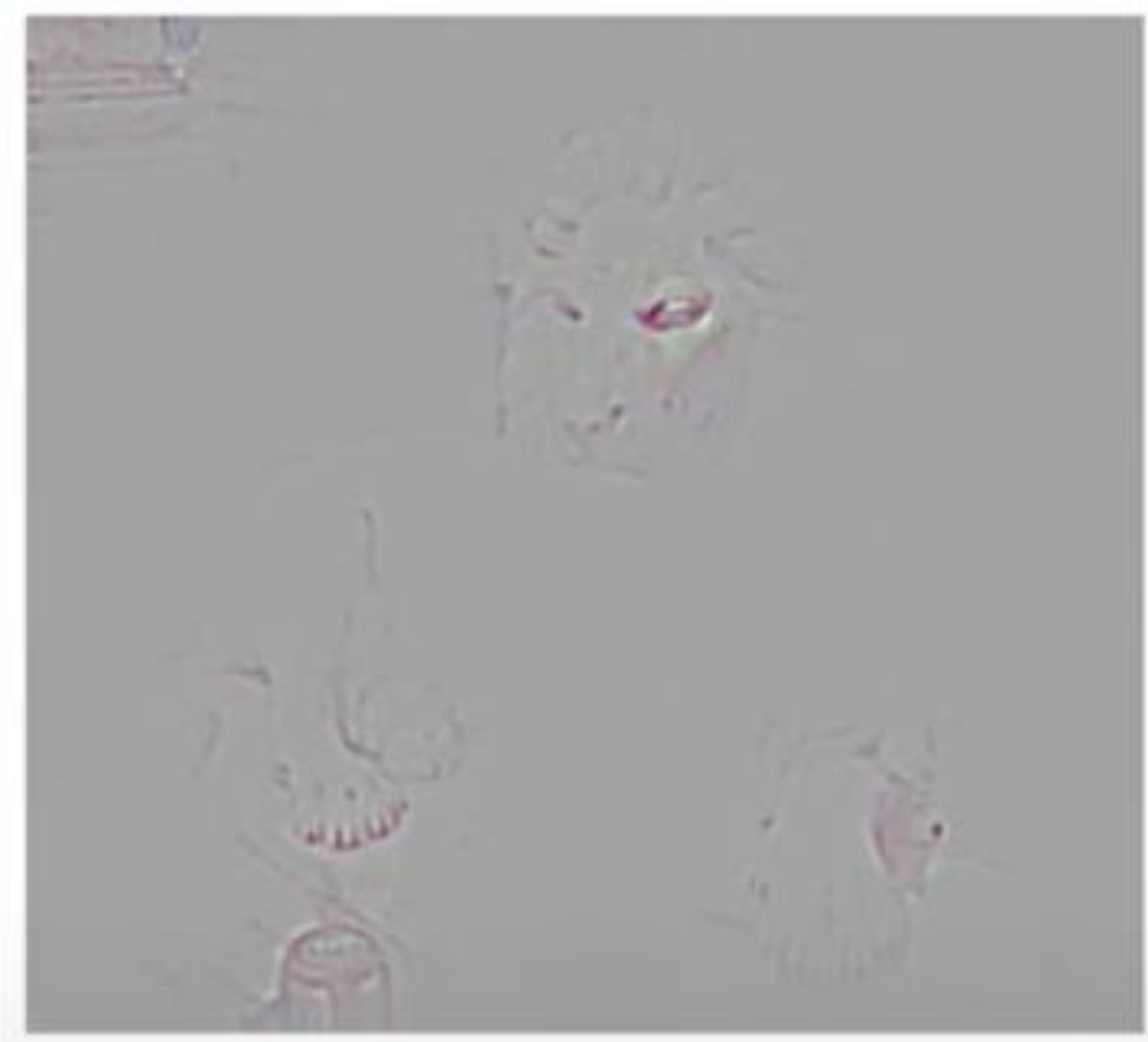
except first convolutional layer

Project Feature Maps To Input Pixel Space

Some feature maps for layer block5_conv1

Input	(InputLayer)	(224, 224, 3)
block1_conv1	(Conv2D)	(224, 224, 64)
block1_conv2	(Conv2D)	(224, 224, 64)
block1_pool	(MaxPooling2D)	(112, 112, 64)
block2_conv1	(Conv2D)	(112, 112, 128)
block2_conv2	(Conv2D)	(112, 112, 128)
block2_pool	(MaxPooling2D)	(56, 56, 128)
block3_conv1	(Conv2D)	(56, 56, 256)
block3_conv2	(Conv2D)	(56, 56, 256)
block3_conv3	(Conv2D)	(56, 56, 256)
block3_conv4	(Conv2D)	(56, 56, 256)
block3_pool	(MaxPooling2D)	(28, 28, 256)
block4_conv1	(Conv2D)	(28, 28, 512)
block4_conv2	(Conv2D)	(28, 28, 512)
block4_conv3	(Conv2D)	(28, 28, 512)
block4_conv4	(Conv2D)	(28, 28, 512)
block4_pool	(MaxPooling2D)	(14, 14, 512)
block5_conv1	(Conv2D)	(14, 14, 512)
block5_conv2	(Conv2D)	(14, 14, 512)
block5_conv3	(Conv2D)	(14, 14, 512)
block5_conv4	(Conv2D)	(14, 14, 512)
block5_pool	(MaxPooling2D)	(7, 7, 512)



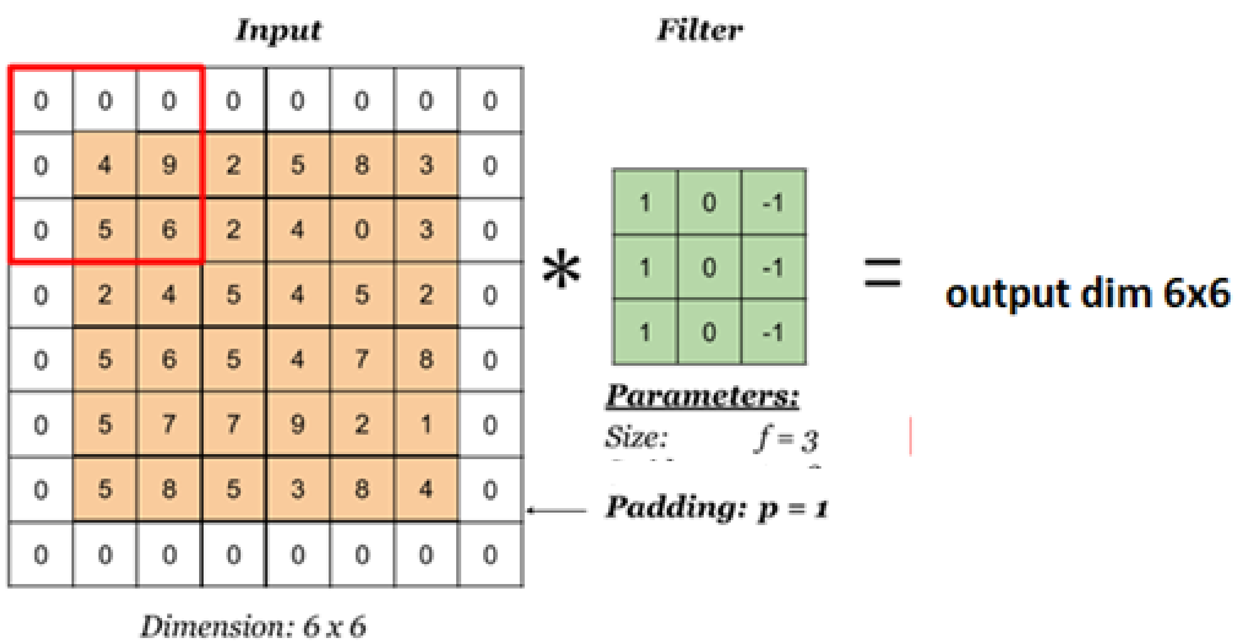


8: Visualizing Decisions (Explainability)

- **Saliency Maps** : Helps explain which pixels are most relevant to the CNN's decision.
- **Grad-CAM (**Gradient-Weighted Class Activation Mapping): Highlights regions in the image that strongly influence the output.

9. Padding

- In order to use deep neural networks (100 layers) we really need to use “**paddings**”.
- **Convolution operation shrinks the image:**
  - In the last section we saw that a **`6x6`** matrix convolved with **`3x3`** filter/kernel gives us a **`4x4`** matrix.
  - To give it a general rule, if a matrix **`nxn`** is convolved with **`fxf`** filter/kernel give us **`n-f+1,n-f+1`** matrix.
  - The convolution operation shrinks the matrix if f>1.
  - Problems:
    - Shrinks output
    - Throwing away a lot of information that is on the edges
- **Solutions:**
  - To solve these problems, we can pad the input image before convolution by **adding some rows and columns to it**. We will call the padding amount `P` the number of rows/columns that we will insert on the top, bottom, left, and right of the image.
  - In almost all the cases the **padding values are zeros**.

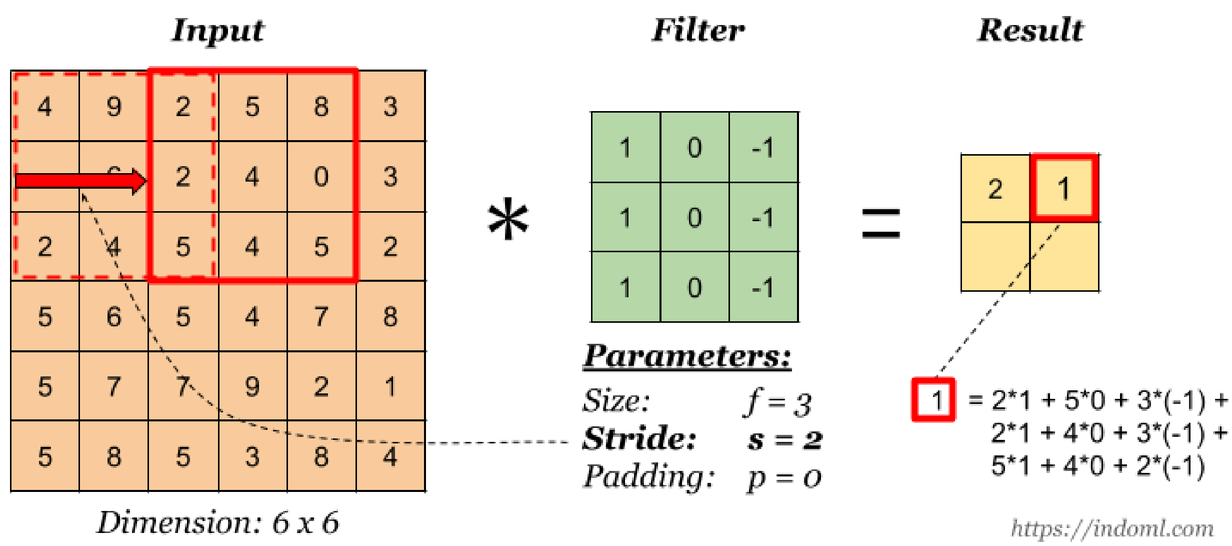


- The general rule now, if a matrix `nxn` is convolved with `fxf` filter/kernel and padding `p` give us `**n+2p-f+1**,**n+2p-f+1**` matrix.
- If n = 6, f = 3, and p = 1 Then the output image will have `n+2p-f+1 = 6+2-3+1 = 6`. We maintain the size of the image.
- The Same convolution is a convolution with a pad so that the output size is the same as the input size. It is given by the equation:
- **P = (f-1) / 2**

- In computer vision f is usually odd. Some of the reasons are that it has a center value.

10. Strided convolution

- Strided convolution is another piece that are used in CNNs.
- We will call stride `S`.
- When making the convolution operation, we used `S` to tell us the number of pixels we will jump when we **are convolving filter/kernel. The last examples we described S was 1**.
- As the information is same in the neighborhood (correlated information) of we may skip it



- **Now the general rule is:**
- if a matrix `nxn` is convolved with `fxf` filter/kernel and padding `p` and stride `s` it give us **`floor((n+2p-f)/s + 1), floor((n+2p-f)/s + 1)`** matrix.

- In case `(n+2p-f)/s + 1` is **fraction** we can take “floor” of this value.

- The Same convolution is a convolution with padding so that the output size is the same as the input size. It’s given by the equation:

- **p = (n*s - n + f - s) / 2**

- We know the general formula for output dimension (for CONV and MAXPOOL operation)

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

- If we want to maintain the input & output dimension remain same.

$$\Rightarrow n = \frac{n+2p-f}{s} + 1$$

$$ns = n + 2p - f + s$$

$$ns - n + f - s = 2p$$

$$\boxed{p = \frac{ns - n + f - s}{2}}$$

→ it is general formula for adding padding for maintaining the input & output dimensions.

if $s = 1 \Rightarrow p = \frac{n \cdot 1 - n + f - 1}{2} \Rightarrow \boxed{\frac{f-1}{2} = p}$

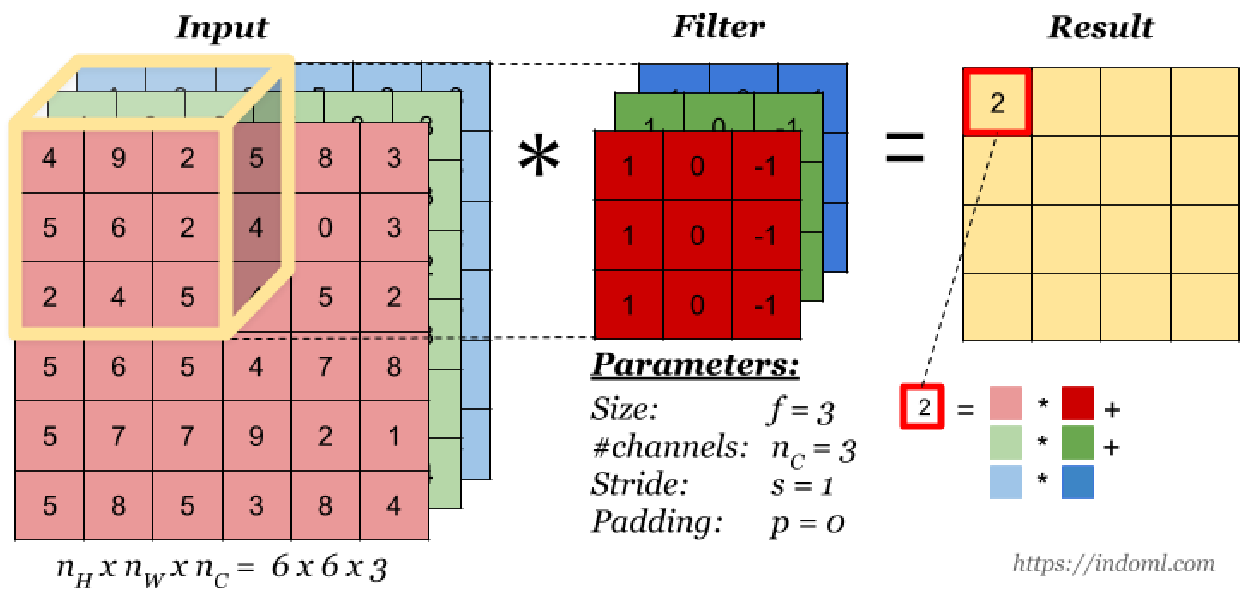
if $s = 2 \Rightarrow p = \frac{n \cdot 2 - n + f - 2}{2} \Rightarrow \boxed{\frac{n+f-2}{2} = p}$

If n = 6, f = 3, s = 1, then p = (3-1)/2 = 1
If n = 6, f = 3, s = 2, then p =(6+3-2)/2 = 3.5 (HomeWork#2: see either its floor or ceiling should take)

Convolutions over volumes

- We see how convolution works with 2D images, now let's see if we want to convolve 3D images (RGB image)

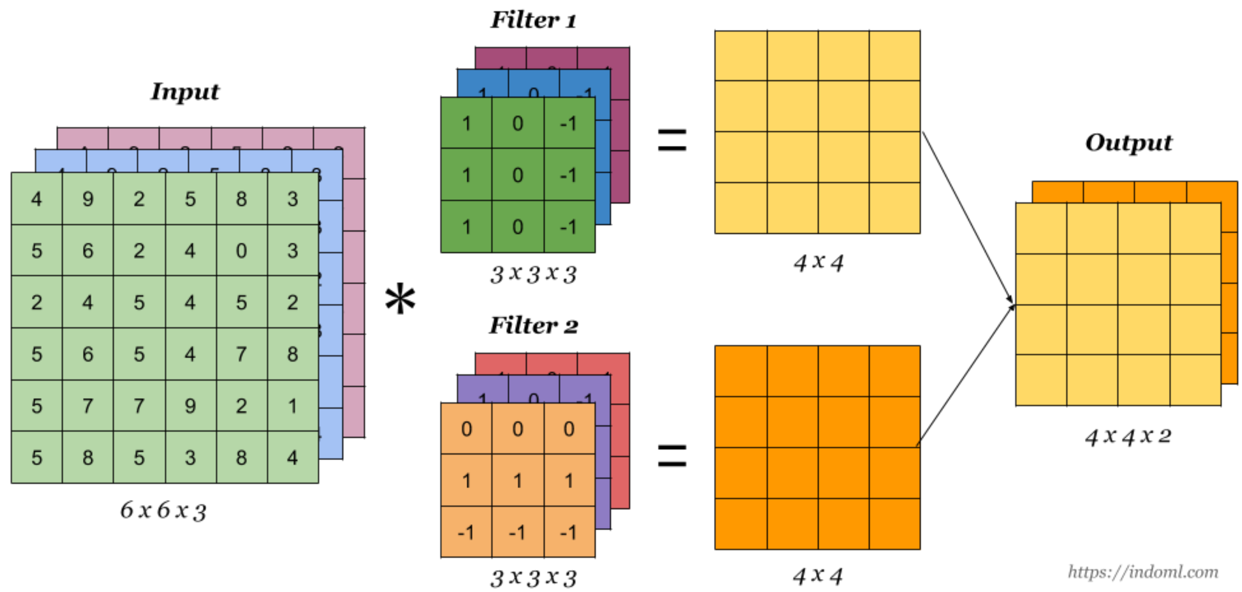
Convolution on RGB Image:



- Here for CONV operation 27 numbers of filters are multiplied by corresponding 27 numbers on RGB image and added to produce the single activation value.
- **Note:** The number of channels of filters must be the same as that of the number of channels of input image/feature maps.
- Example:
  - Input image: `6x6x**3**`
  - Filter: `3x3x**3**`
  - Result image: `4x4x1`
  - In the last result p=0, s=1
  - Hint the output here is only 2D.

Convolution Operation with Multiple Filters

- Multiple filters can be used in a convolution layer to detect multiple features. The output of the layer then will have the same number of channels as the number of filters in the layer.

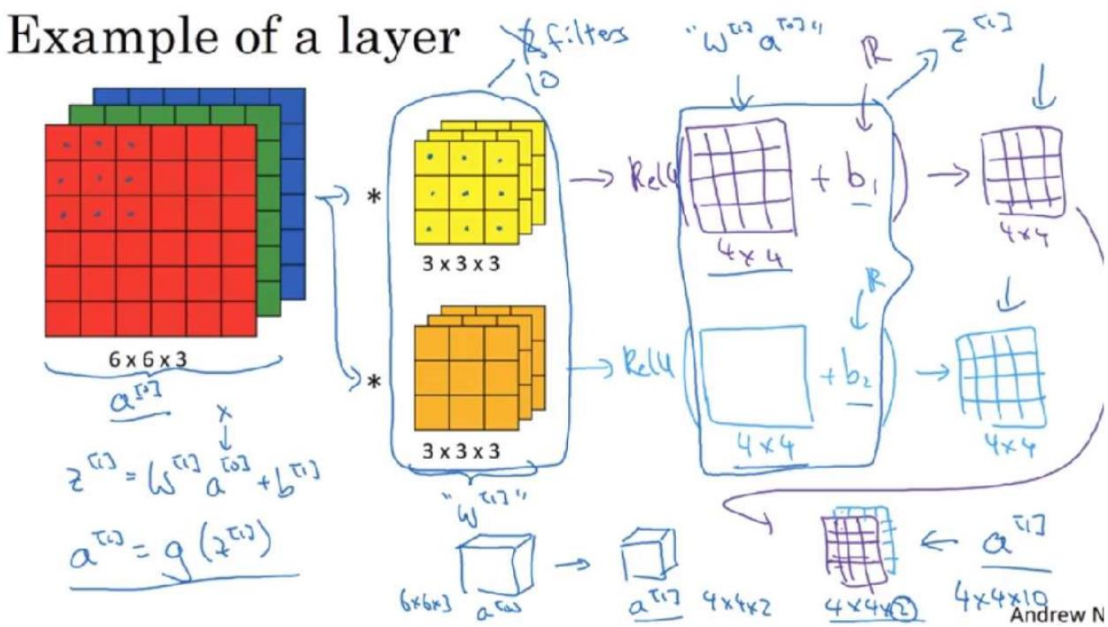


- If we can use multiple filters (let's say10 filters) to detect multiple features.
  - Input image: 6x6x3
  - 10 Filters: 3x3x3
  - output dim: 4x4x10
  - In the last result p=0, s=1

Output dim: n –f + 1 x n – f + 1 x # of filters

One Convolution Layer

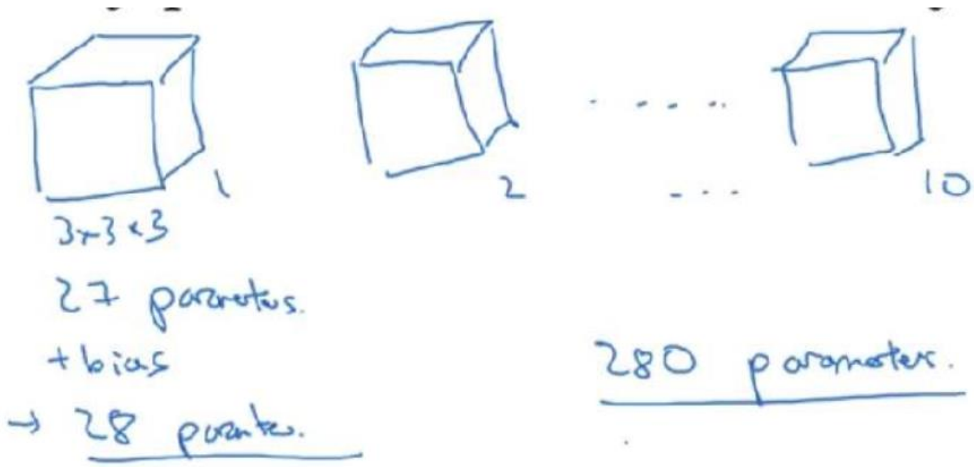
- Till now we did $\mathbf{z^{[1]} = W^{[1]} * a^{[0]}}$
- To make **a convolution layer**, a bias (ϵ R) is added ($\mathbf{z = W*x} + b$), and then apply activation function such as **ReLU** or **tanh** is applied to it. (**a = g(W*x + b)**)



Question: Number of parameters in one layer

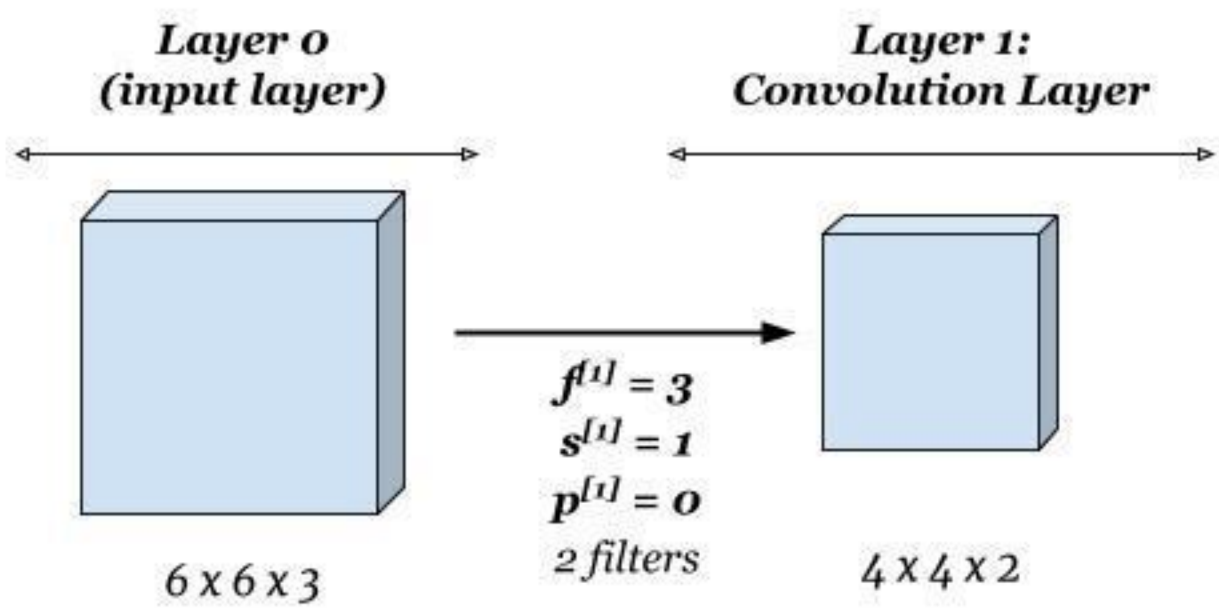
- If you have **10** filters of dim **3x3x3** in one layer of a CNN, how many parameters does that layer have??
- Answer: _______________ parameters???

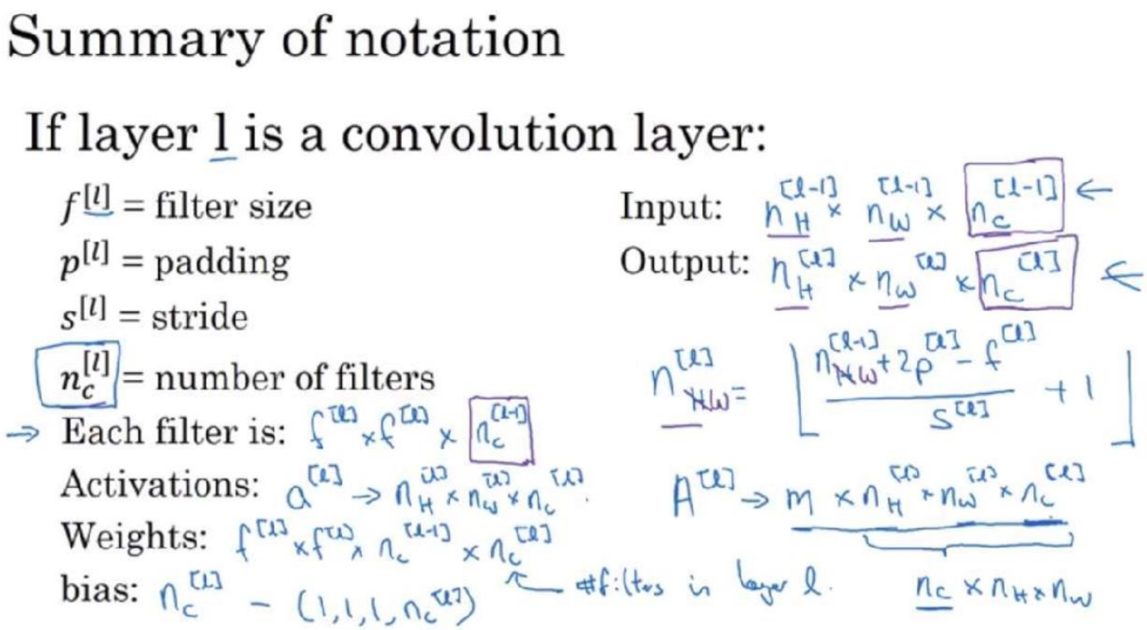
Answer: For a larger image of size says 1000x1000, but still the no of parameters is only 280. So, less overfitting in the case of CNN.



Shorthand Representation

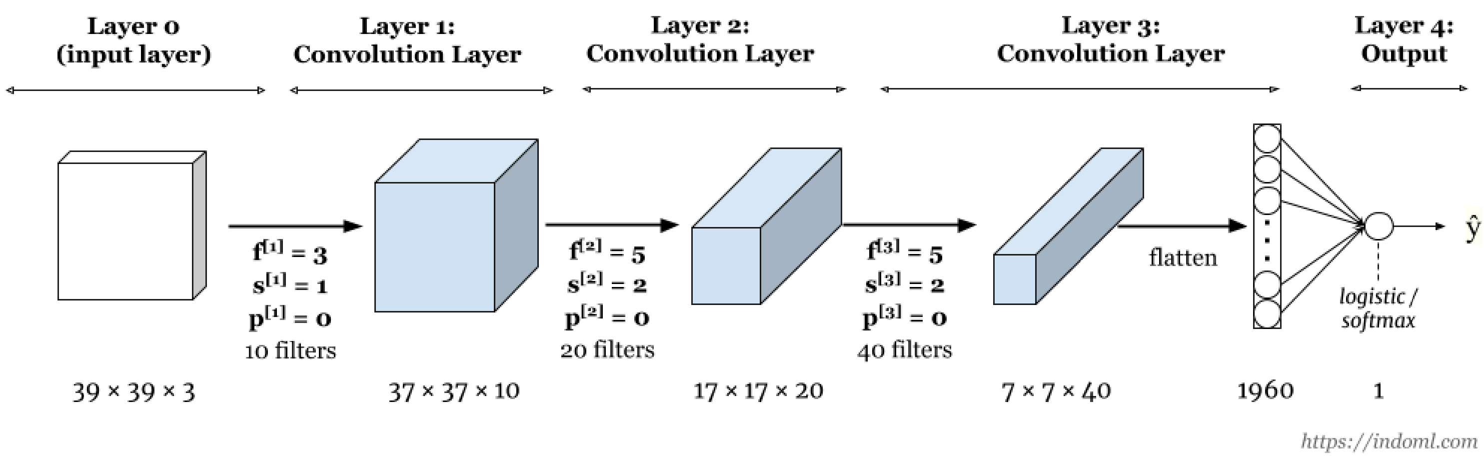
- This simpler representation will be used from now on to represent one convolutional layer:





Sample Complete Network:

- This is a sample network with three convolution layers. At the end of the network, the output of the convolution layer is flattened and is connected to a logistic regression or a softmax output layer.



The general trend for hyperparameters:

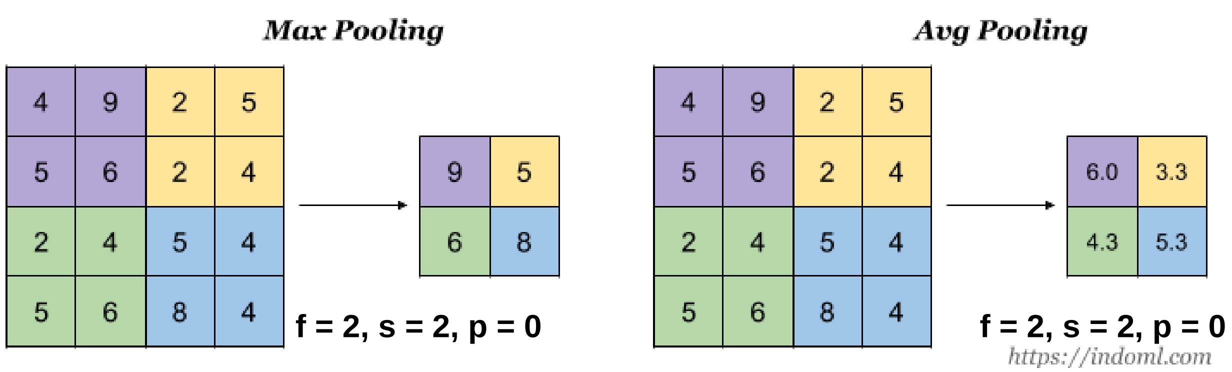
- Size of feature maps (activations) decreases with depth
- The number of channels is increased with depth
- Type of layers are **CONV- POOL – FC – logistic/softmax**

Layer in a convolutional network:

- Convolution: `Conv`
- Pooling: `Pool`
- Fully connected: `FC`

Pooling Layer

- Pooling layer is used to reduce the size of the representations and to **speed up calculations**, as well as to make some of the **features it detects a bit more robust**.
- Sample types of pooling are **max pooling** and **avg pooling**, but these days max pooling is more common.



- Down sample high dimensional image by taking the important information.
- It has hyper-parameters:
  - **size** (***f***)
  - **stride** (***s***)
  - **type** (max or avg)
  - **p = 0** (Generally)
- **but it doesn't have parameters; there's nothing for gradient descent to learn**

Max pool vs average pool:

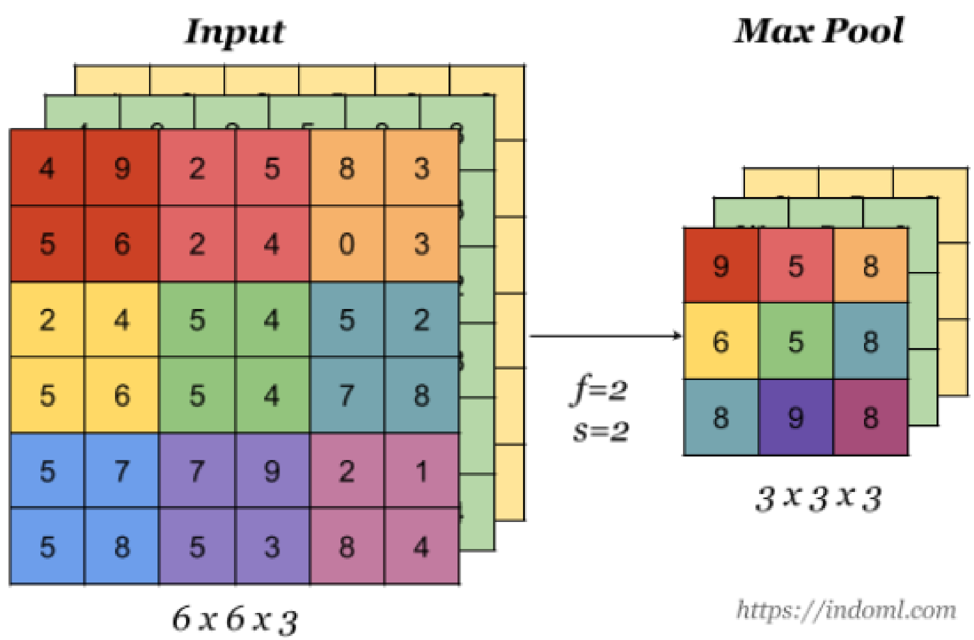
- Max pooling gives the most prominent feature in a particular patch of the feature map, average pooling gives the average of features present in a patch.
- Max pooling extracts only the most salient features of the data.
- Average pooling smoothly extracts features

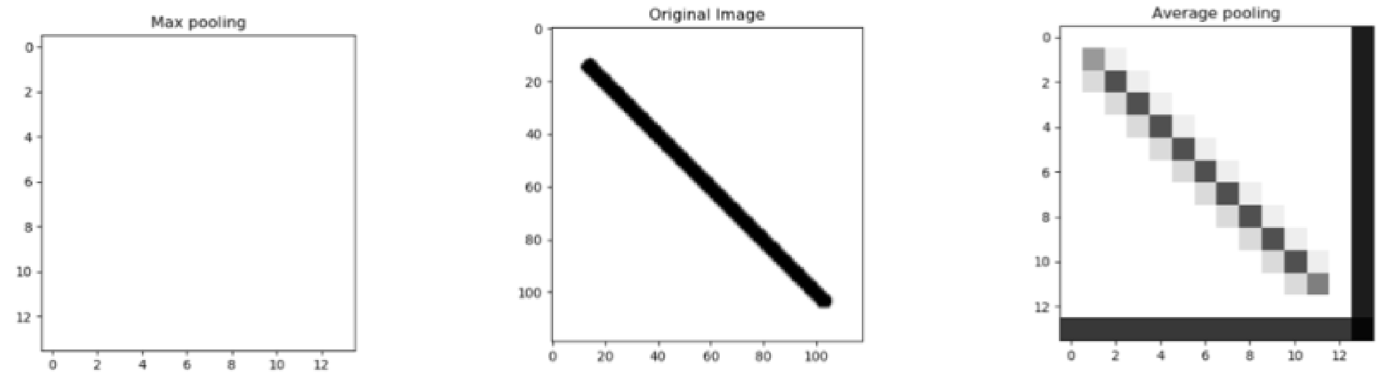
Hence,

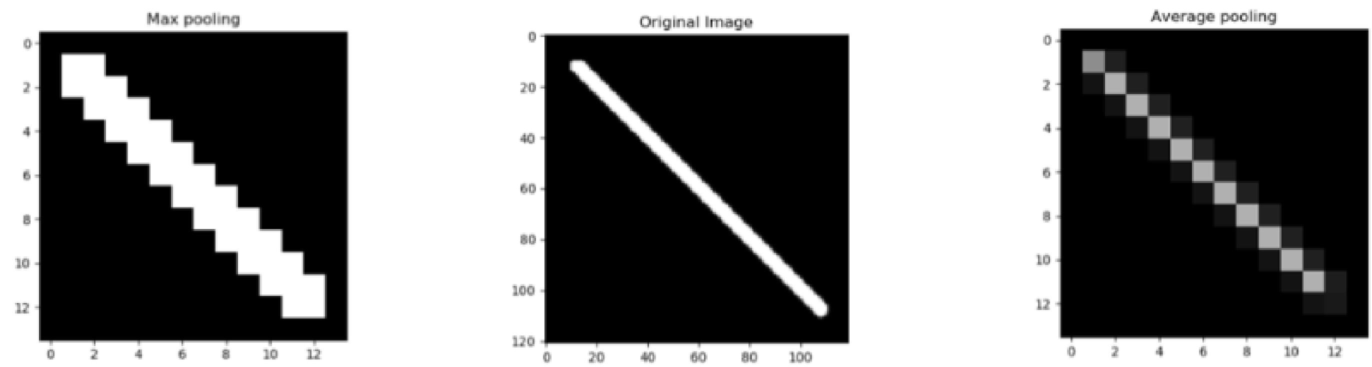
- Average pooling sometimes cannot extract the important features because it takes everything into account and gives an average value that may or may not be important.
- Max pooling focuses only on the very important features.

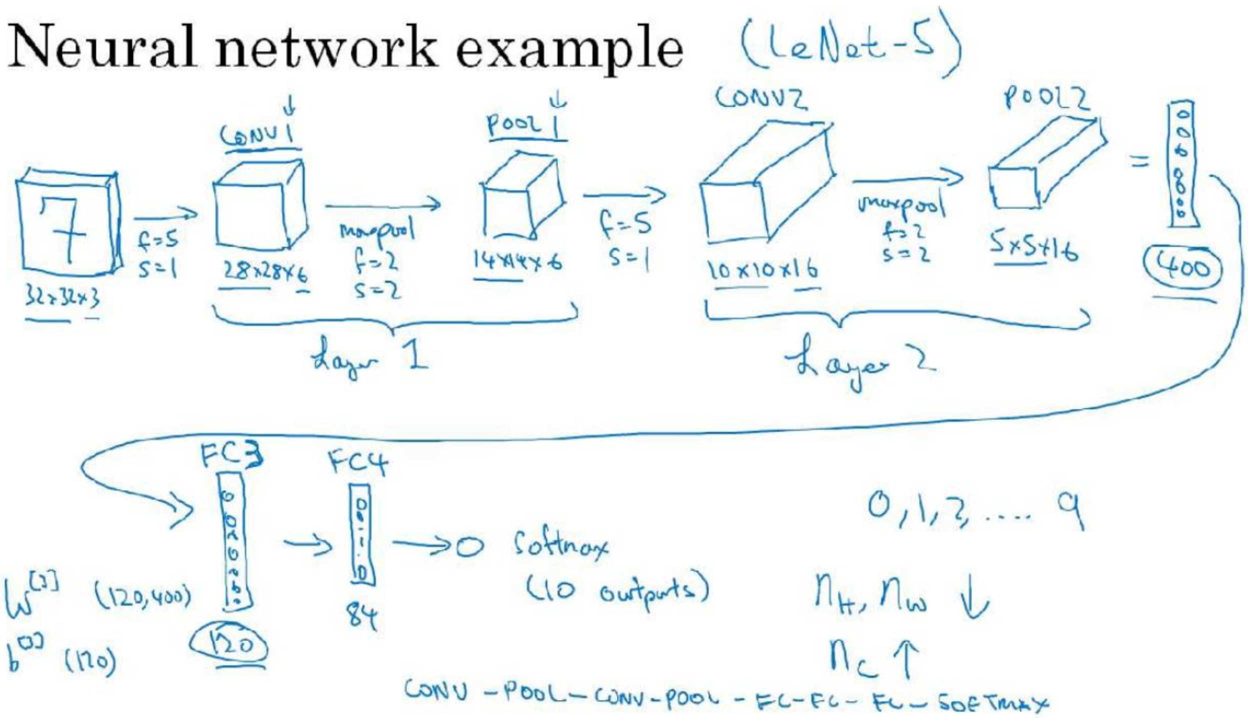
But this also means,

- Average pooling encourages the network to identify the complete extent of the object, whereas max pooling restricts that to only the very important features, and might miss out in some details.
- Hence, Choice of pooling method is dependent on the expectations from the pooling layer and the CNN




Max pooling
Original Image
Average pooling


Max pooling
Original Image
Average pooling



Homework: Fill the following table with correct values

CONV1 : activation shape= 28x28x6, activation size= 4704, parameters: (5x5x3 +1) *6 = 76* 6 = 456

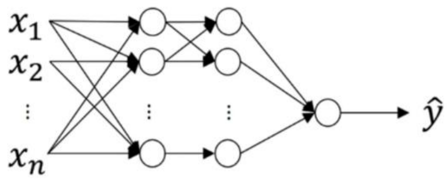
	Activation shape	Activation Size	# parameters
Input:	(32,32,3)	– 3,072 $a^{[0]}$	0
CONV1 (f=5, s=1)	(28,28,8)	6,272	208
POOL1	(14,14,8)	1,568	0
CONV2 (f=5, s=1)	(10,10,16)	1,600	416
POOL2	(5,5,16)	400	0
FC3	(120,1)	120	48,001
FC4	(84,1)	84	10,081
Softmax	(10,1)	10	841

Note: Trends in CNN are as follows:

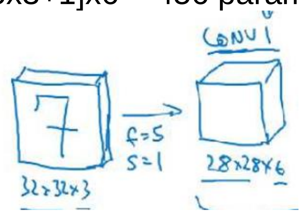
i. POOL layer requires no learnable parameters
ii. CONV layer requires fewer parameters compared to FC layers
iii. Activation size decreases gradually with the depth of CNN
iv. Number of channels increases gradually with the depth of CNN

Why convolution is good??

- 32 x32 x 3 * 5x5x3 = > 28 x28 x6 (#filter: 6)
- Input is 32 x32 x 3
- Output is 28 x28 x6
- In NN 3037(32x32x3) x 4704(28x28x6) = 14 million parameters



- But in CNN on [5x5x3+1]x6 = 456 parameters are needed



- **The Small number of parameters is because of:**
  - **Parameter sharing:** The same 5x5x3 filter (vertical edge detector) is used to perform the CONV operations for the whole image
  - **Sparsity of connection**: To compute the output of one neuron only the 5x5x3 region of the input image is seen (CONV) instead of whole image as did in NN