CSE 673
COMPUTATIONAL VISION

VENU GOVINDARAJU
DEEN DAYAL MOHAN

University at Buffalo
Department of Computer Science and Engineering
School of Engineering and Applied Sciences

Covid-19 Guidelines

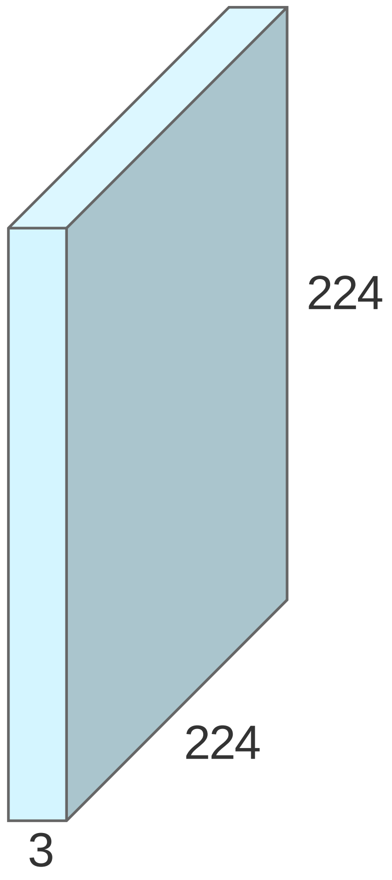
- Effective Aug. 3, the University at Buffalo will require all students, employees and visitors – regardless of their vaccination status – to wear face coverings while inside campus buildings. This includes classrooms, hallways, libraries and other common spaces, as well as UB buses and shuttles.

- Students are expected to wear mask in class during lectures (unless you have a UB approved exception)

- Public Health Behavior Expectations https://www.buffalo.edu/studentlife/who-we-are/departments/conduct/coronavirus-student-compliance-policy.html

Convolutional Neural Network

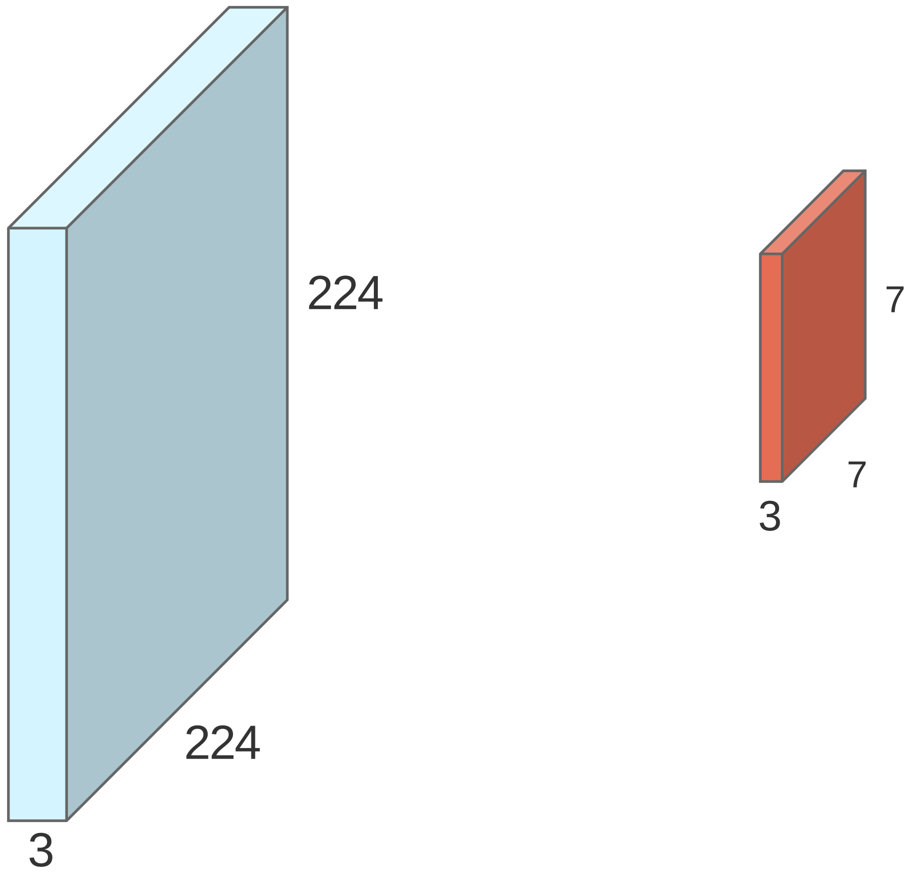
- Let us look at an image as a volume



- Width and height of the image is 224
- The Depth of the image is 3. This indicate the number of channels in the image.
- Total number of pixels in the image is 224 X 224 X 3

Convolutional Neural Network

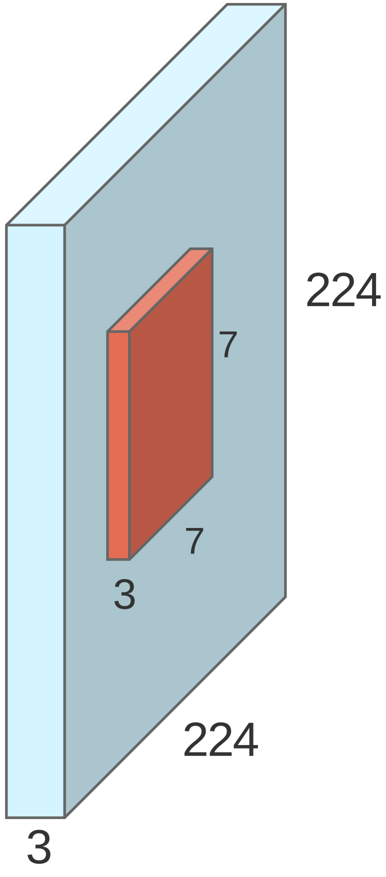
- Let us look at a Convolutional Layer



- Second small volume has a height and width of 7 and depth of 3.
- This image is often referred to as a filter.
- Sometimes in literature the filter size is only written as 7 X 7
- The depth of filter extends to the entire depth of the image
- In effect, an ordinary 7X7 filter is 7X7XD where D is the depth of the input volume

Convolutional Neural Network

- Let us look at a Convolutional Layer



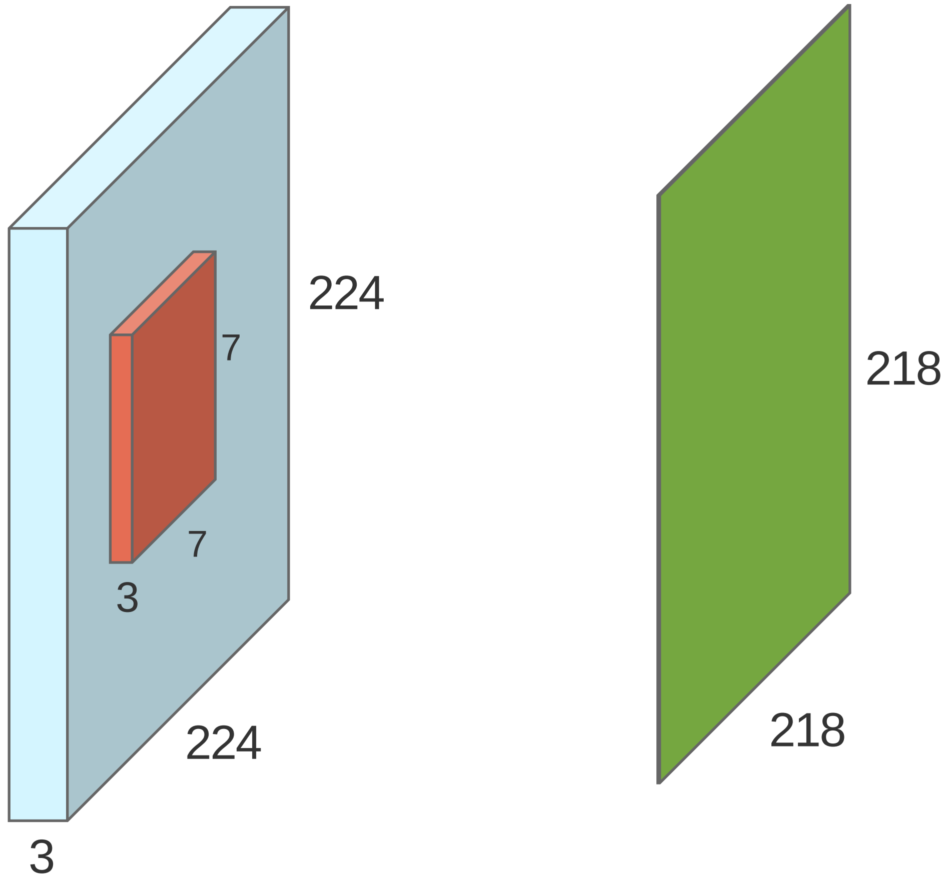
- The filter will be convolved with a part of the input volume
- At each location W^TX is computed
- That in effect would be taking element wise product of the filter with the part of the input volume that it overlaps
- This can be written as

$$\sum_{1=1}^{wxh \times D} X_i \times w_i$$

- Where X is the image and w is the filter
- This operation repeated by sliding the filter throughout the image

Convolutional Neural Network

- Let us look at a Convolutional Layer



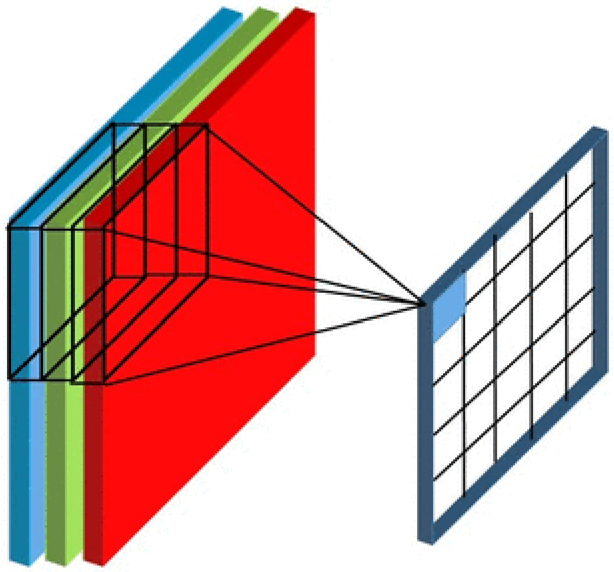
- The output volume after performing this operation has a size of

$$W^{new} \times H^{new} \times 1$$

- In this case we get an output of 218 X 218 X 1
- The value of width and height is determined by the number of unique location in which the filter can slide over
- The depth of the output is one, if we use one such filter
- Exact math of how we got 218 X 218, we will look at later.

Convolutional Neural Network

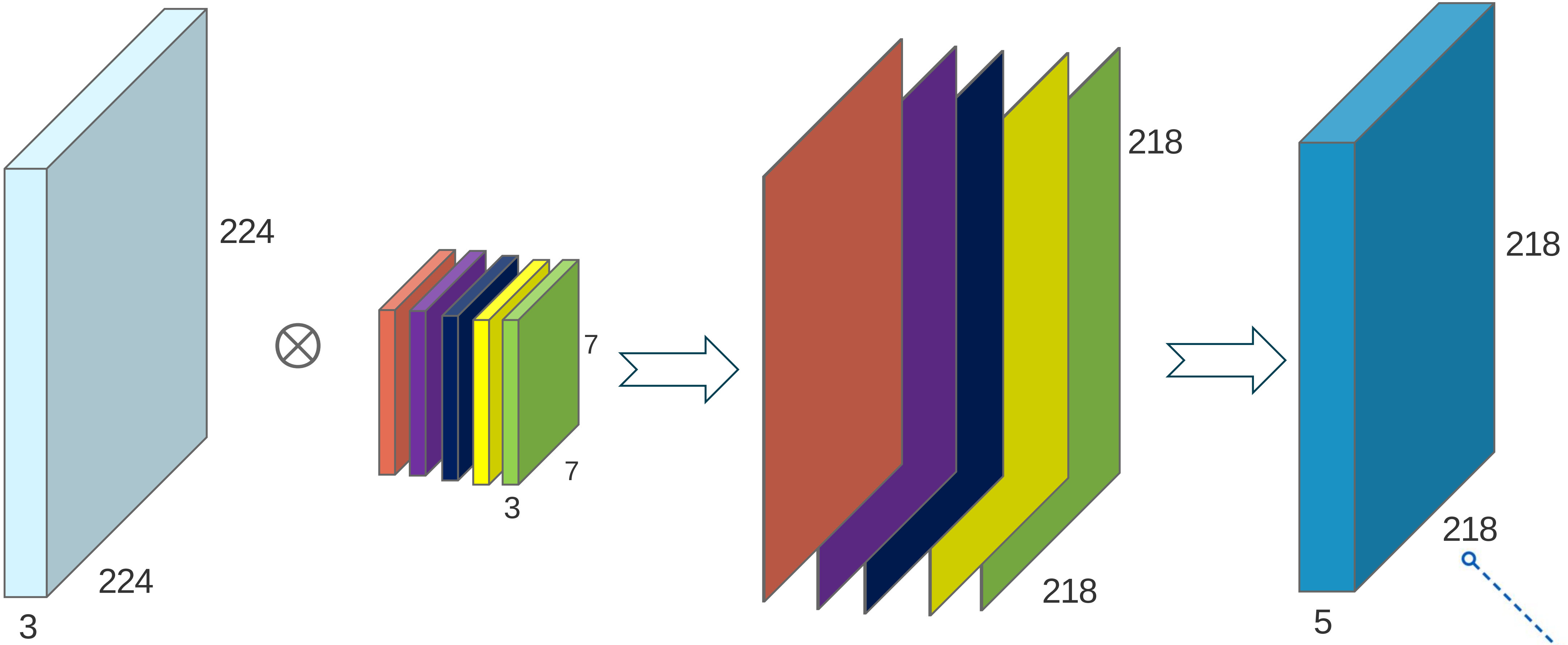
- Let us look at a Convolutional Layer



- The filter size extends entire channels of the input volume.

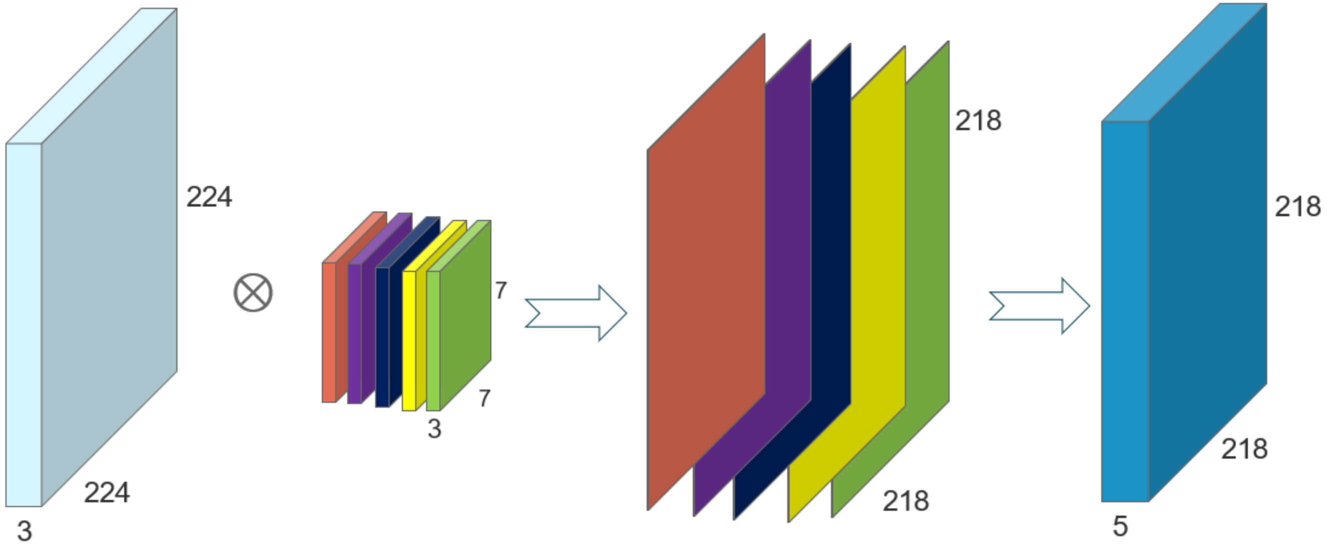
Convolutional Neural Network

- Let us look at a Convolutional Layer



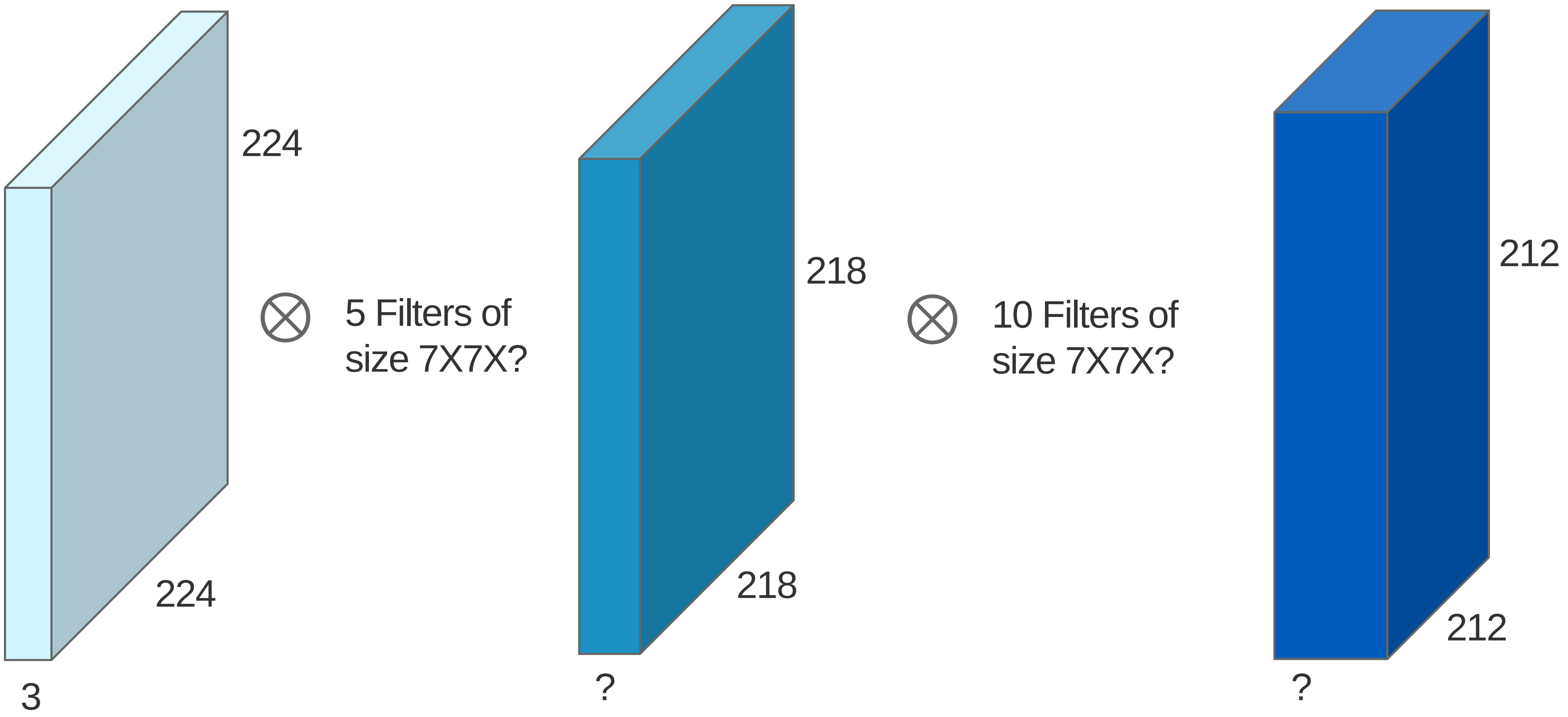
Convolutional Neural Network

- Each filter is applied separately on the input volume
- Each output created is stacked together to create the output volume
- Output volume is often called convolutional map or convolutional activation map
- The depth of the output volume is equal to the number of filters applied



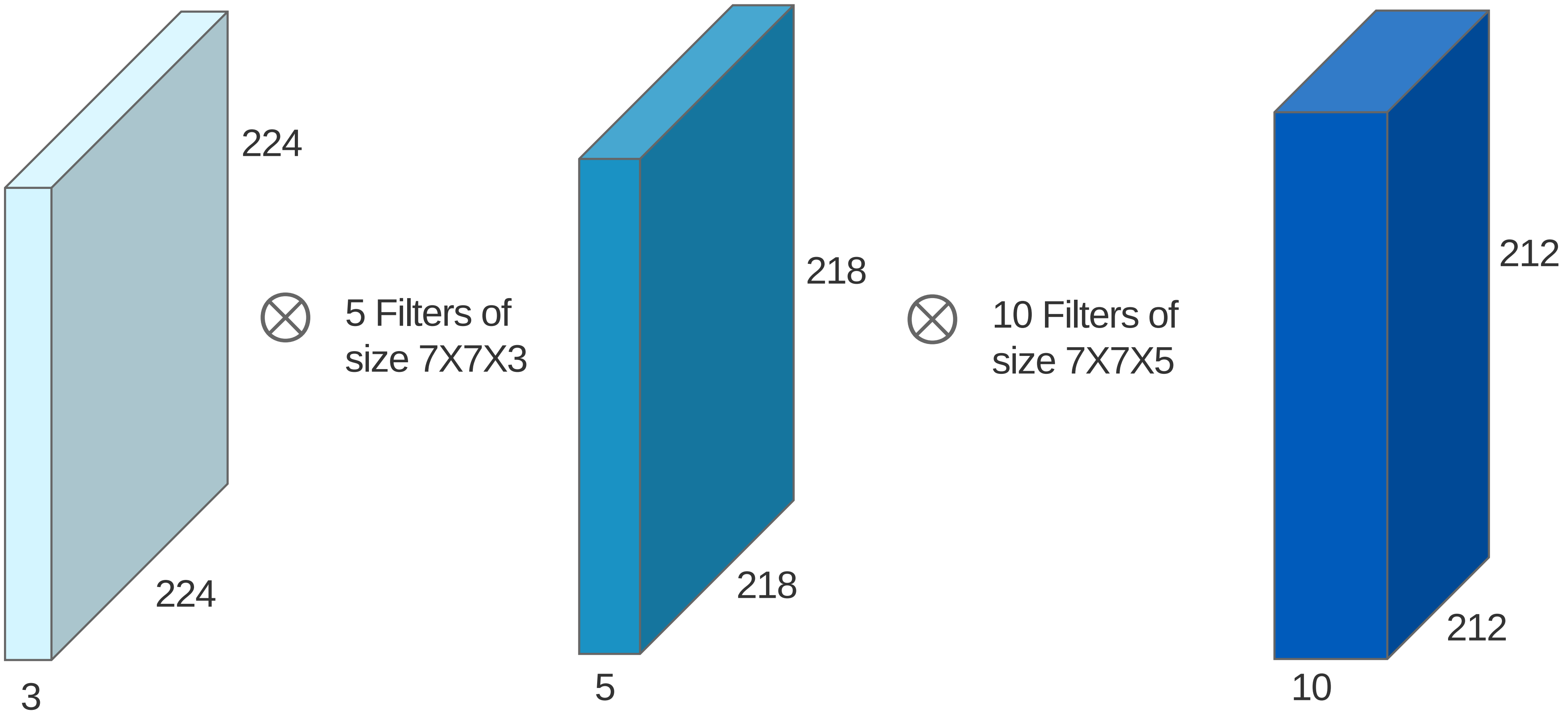
Convolutional Neural Network

- How are these convolutional layers arranged in a Neural Network?



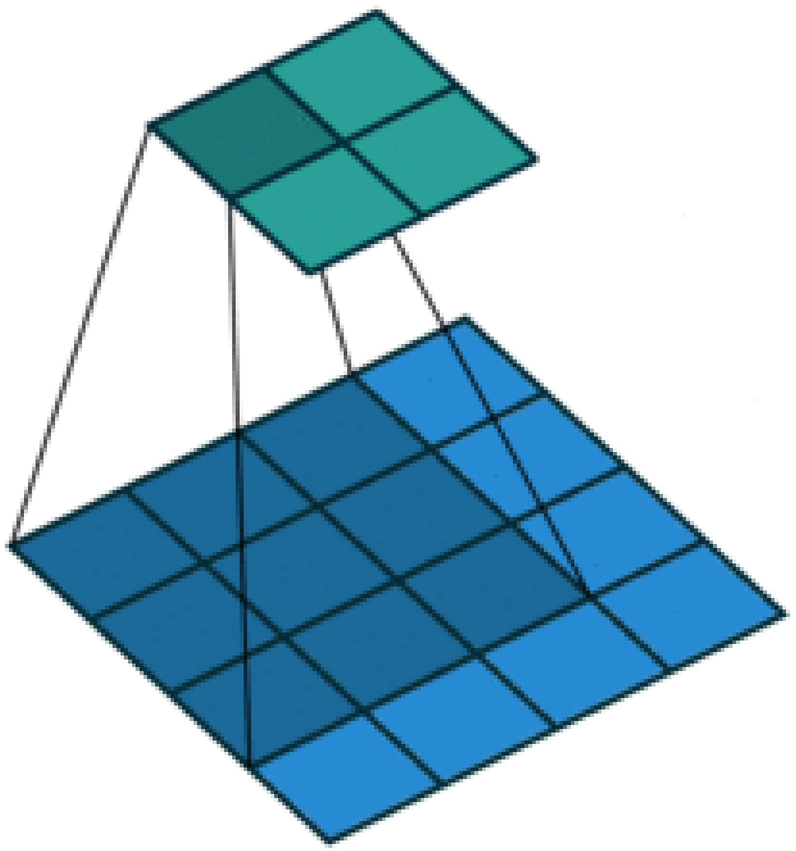
Convolutional Neural Network

- How are these convolutional layers arranged in a Neural Network?



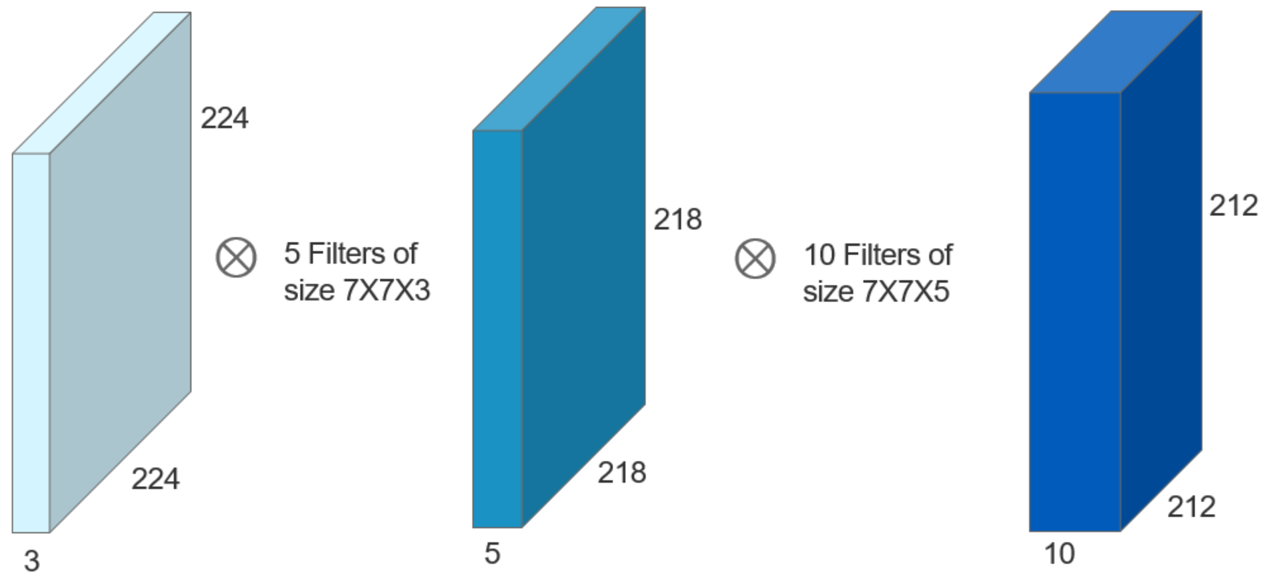
Convolutional Neural Network

- Let us look at top-down view of a convolutional layer
- What is the image size in this case?
- What is the filter size in this case?
- What is output spatial size in this case?
- What about depth?
- What is the stride?



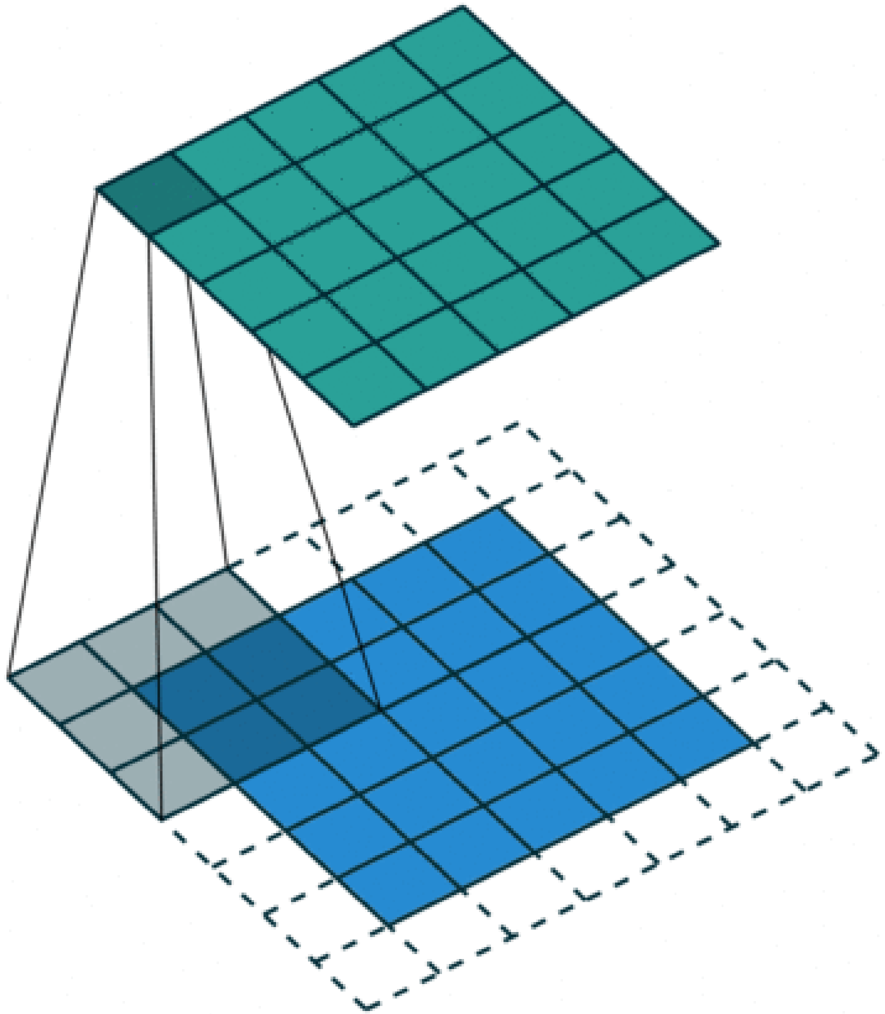
Convolutional Neural Network

- If you look closely the spatial size of the input decreases when a convolution filter is applied.
- This decrease in size is not ideal, since we want to build deep neural networks
- In order to avoid this size reduction, we use padding
- We pad the original image with zeros so that we get the original image size back after convolution



Convolutional Neural Network

- The image is padded with two rows and two columns of zeros
- This essentially gives the filter more unique locations to fit.
- The output spatial size created after the reduction associated with convolution operation is the same special size of the input
- Since the same spatial size is returned, this particular padding is sometimes referred as "same" padding



Convolutional Neural Network

- Let us now look at the arithmetic of the convolution operation
- Let the filter have width F_w and height of F_h
- Let the input image volume has a width of W, height of H and depth of K
- If the filter is applied with a stride of S and padding of the original image is P, then the final image size is given by the formula

$$W_{out} = [(W-F_w + 2P)/S] + 1$$

$$H_{out} = [(H-F_h + 2P)/S] + 1$$

C_{out} = Number of such filters applied

Convolutional Neural Network

- Let us look at an example

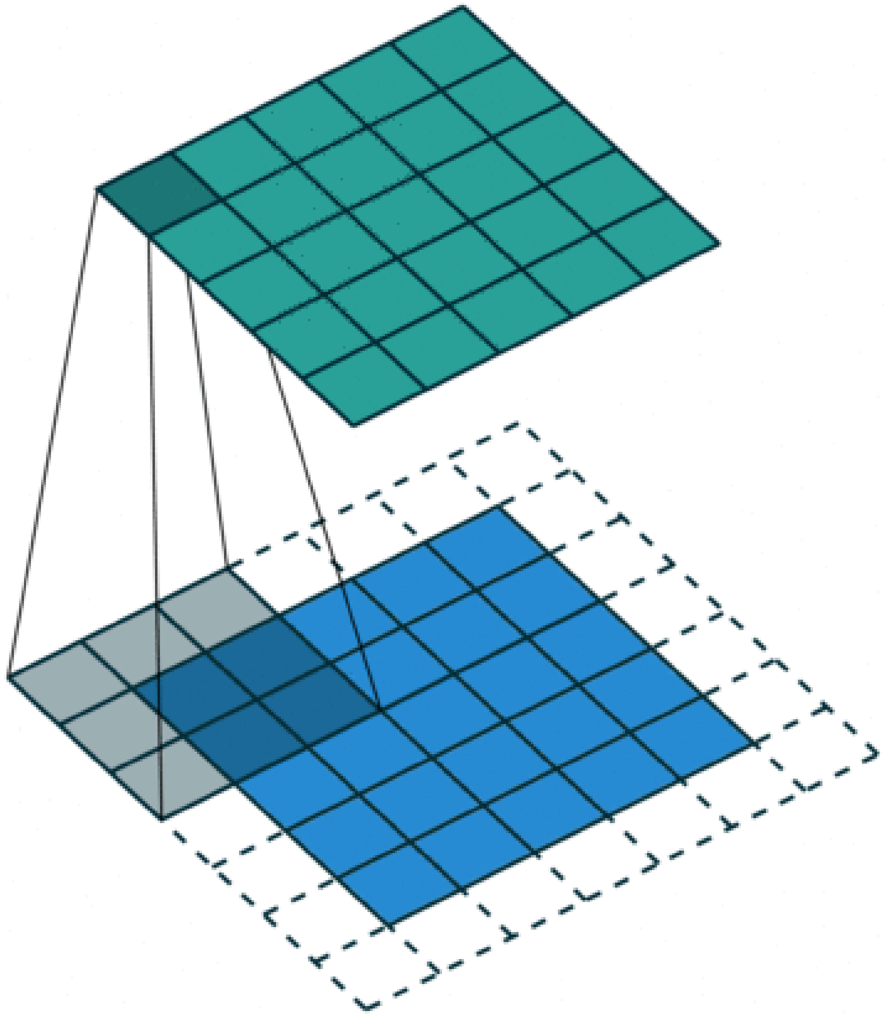
- Here input size of the image is 5 X 5. Assume the image has 3 channels. The filter size is 3X3X3 (since it extends the full depth)

- The padding in this particular case is 1 on each size of the input image and the stride is 1

- The final output size would be

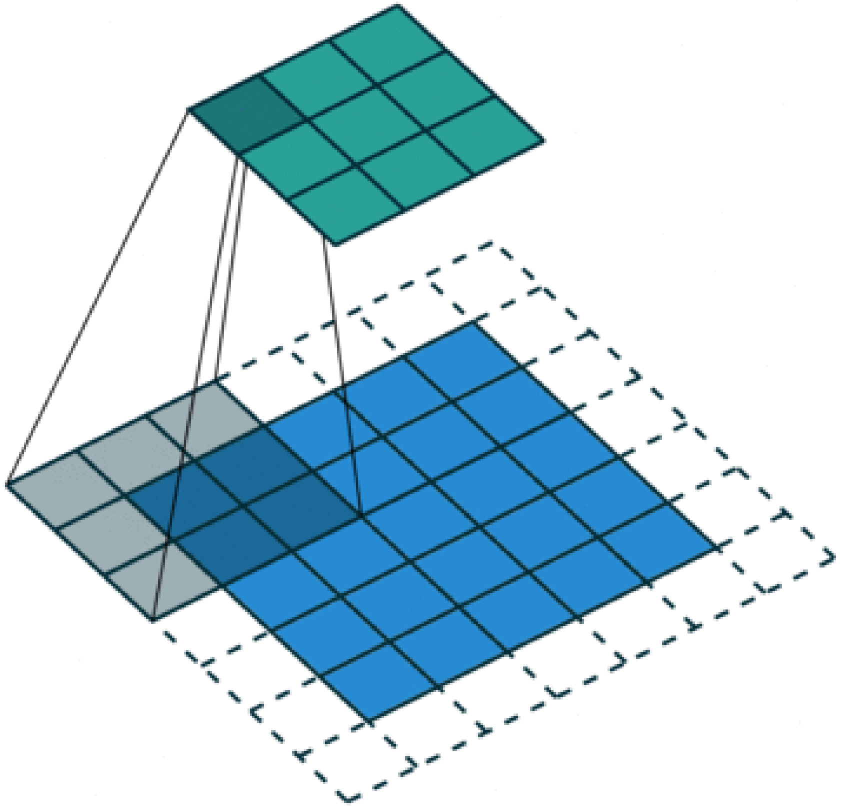
$$W_{out} = [((5-3) + 2*1)/1] + 1 = 5$$

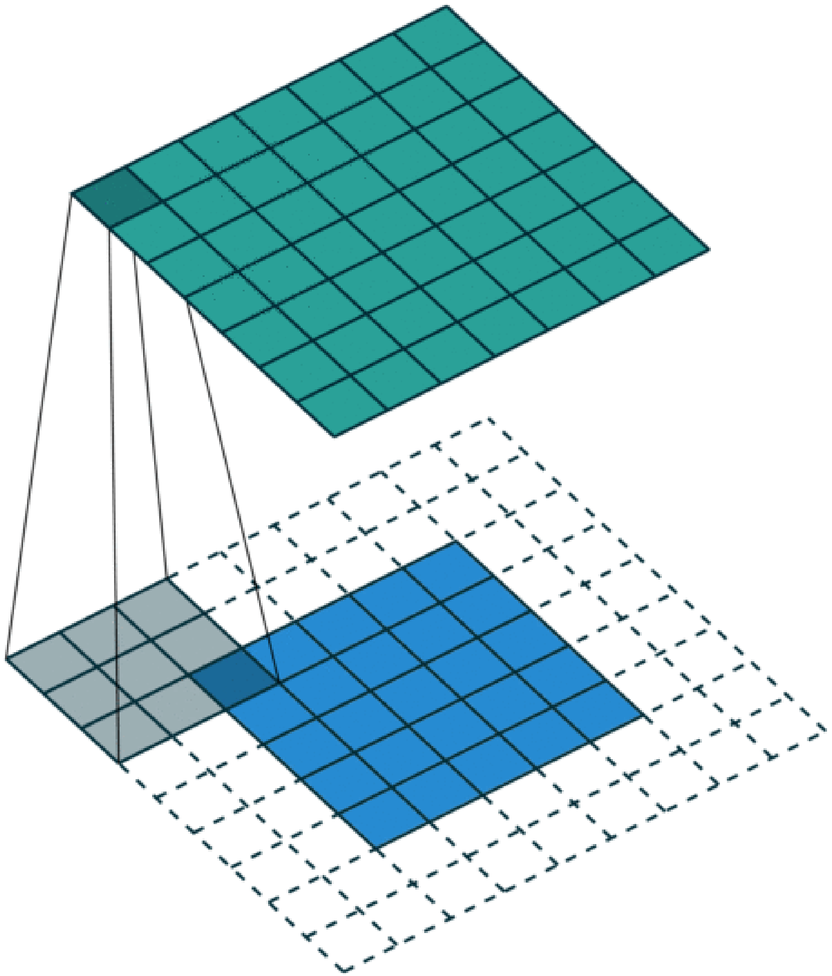
$$H_{out} = [((5-3) + 2*1)/1] + 1 = 5$$



Convolutional Neural Network

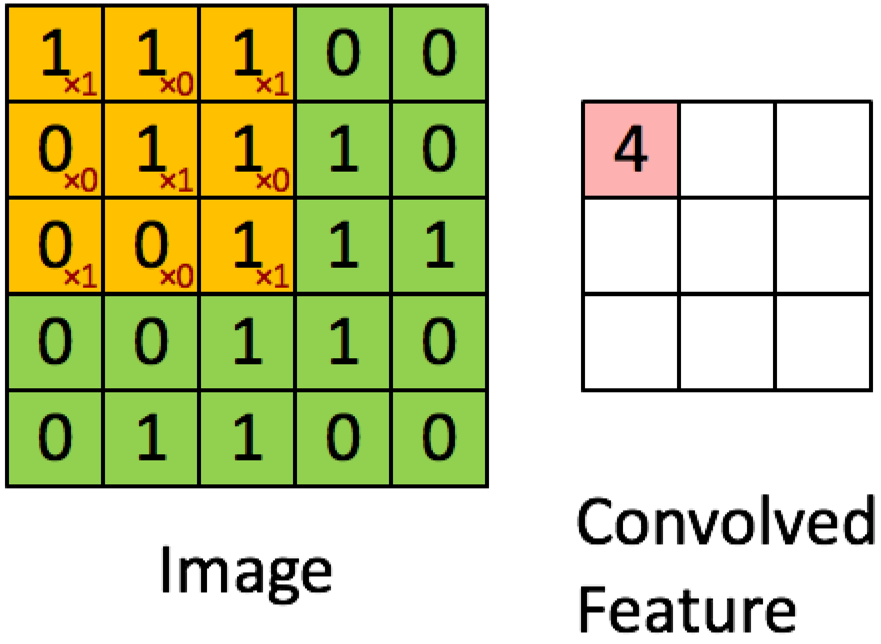
- What about these examples





Convolutional Neural Network

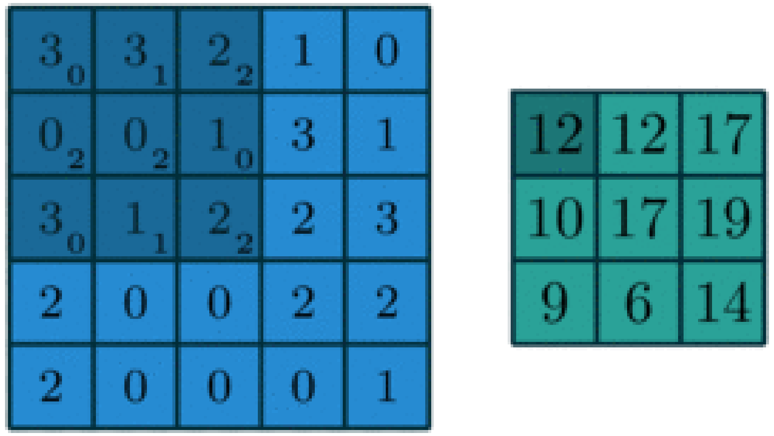
- Convolution with numbers

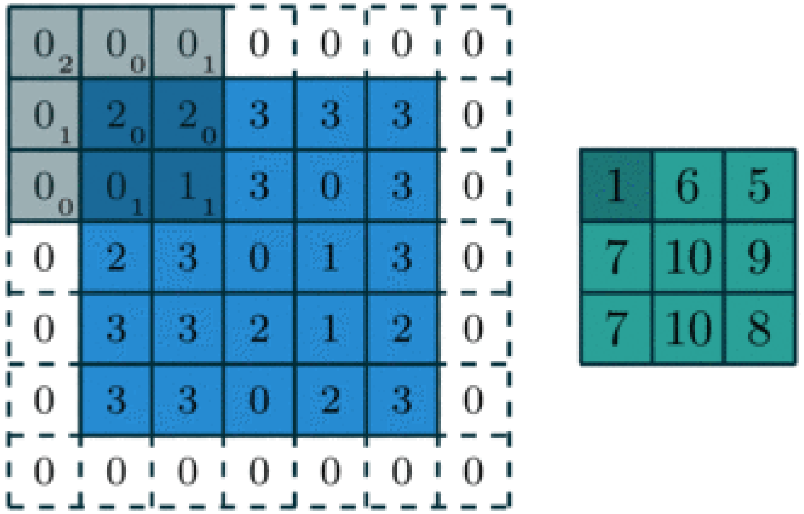


- Image size is 5X5
- Filter size is 3x3
- Stride 1
- No padding

Convolutional Neural Network

- Convolution with numbers



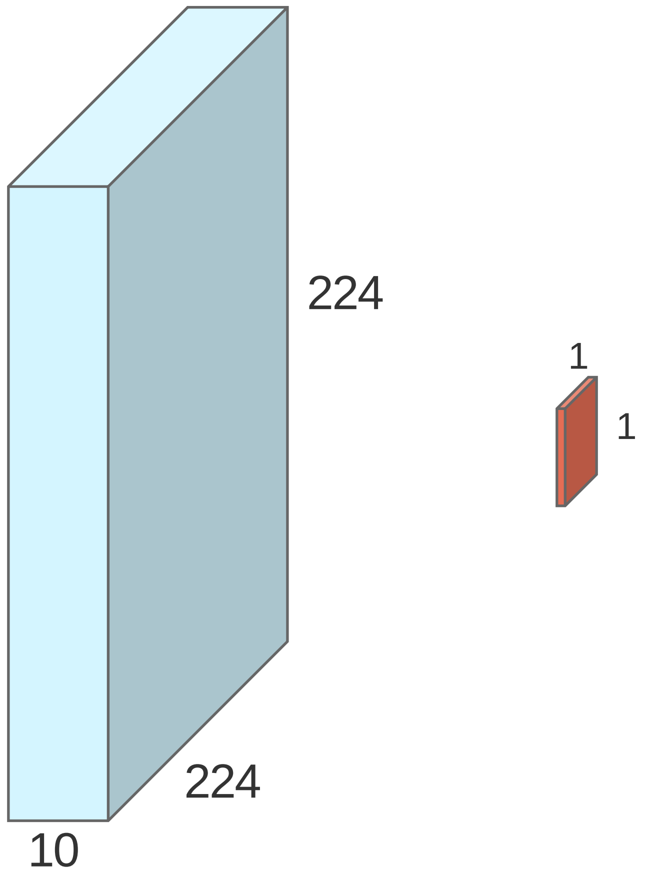


- Image size is 5X5
- Filter size is 3x3
- Stride 1
- No padding

- Image size is 5X5
- Filter size is 3x3
- Stride 2
- Padding 1

Convolutional Neural Network

- Let us look at a convolutional layer



- Special type of convolution.
- Usually called 1x1 convolution.
- Does not aggregate spatial information.
- The filter is 1x1xD size.
- It performs a dot product at each pixel along the depth.

Agenda

- Backpropagation in CNN
- Pooling
- Parameter calculations
- Convolutional Architectures

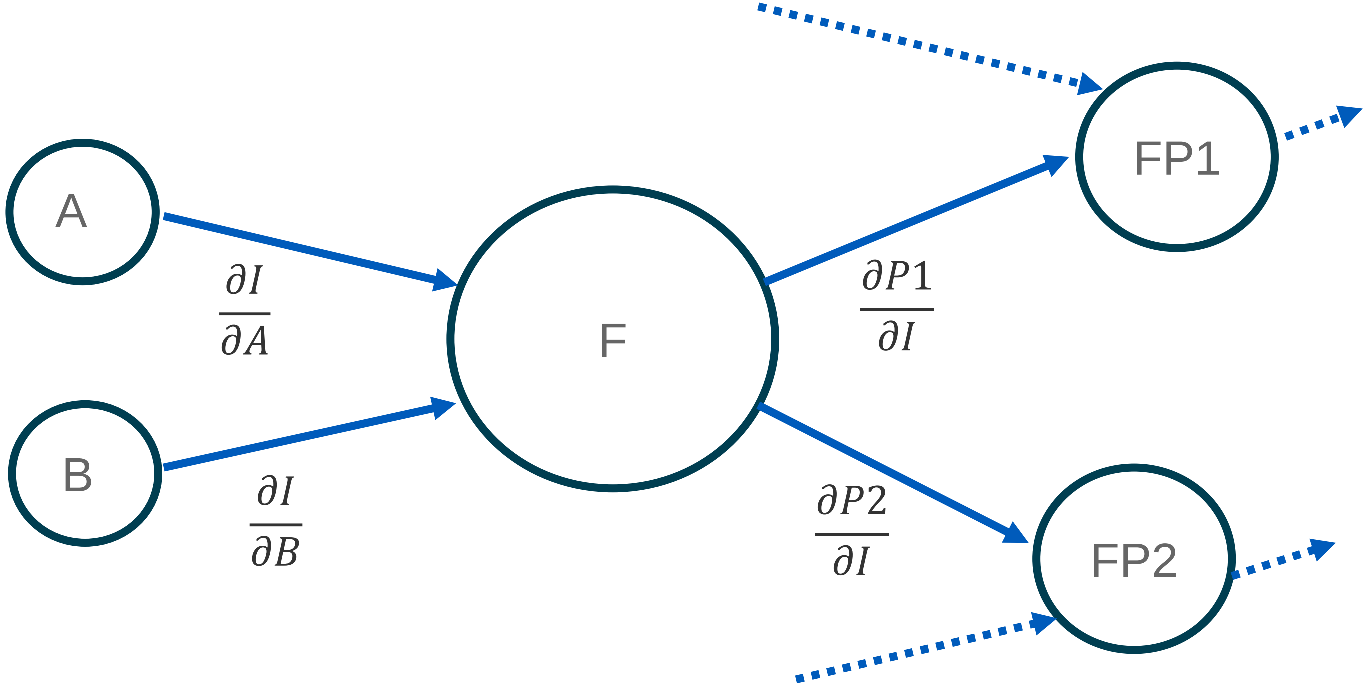
Convolutional Neural Network

- How does back propagation work in a convolutional neural network?

- Imagine if the intermediate result of the function is used in two different computations FP1 and FP2

- In order to compute the gradient $\frac{\partial output}{\partial A}$ and $\frac{\partial output}{\partial B}$, we will sum up all the gradients at F

- $$\frac{\partial Output}{\partial A} = \frac{\partial I}{\partial A} * \left(\frac{\partial P1}{\partial I} + \frac{\partial P2}{\partial I}\right)$$
  $$\frac{\partial Output}{\partial B} = \frac{\partial I}{\partial B} * \left(\frac{\partial P1}{\partial I} + \frac{\partial P2}{\partial I}\right)$$



Convolutional Neural Network

- How is the convolutional layer defined in the deep learning packages?

PyTorch

CONV2D

CLASS `torch.nn.Conv2d(in_channels, out_channels, kernel_size, stride=1, padding=0, dilation=1, groups=1, bias=True, padding_mode='zeros', device=None, dtype=None)` [SOURCE]

Applies a 2D convolution over an input signal composed of several input planes.

In the simplest case, the output value of the layer with input size (N, C_{in}, H, W) and output $(N, C_{\text{out}}, H_{\text{out}}, W_{\text{out}})$ can be precisely described as:

$$\text{out}(N_i, C_{\text{out}_j}) = \text{bias}(C_{\text{out}_j}) + \sum_{k=0}^{C_{\text{in}}-1} \text{weight}(C_{\text{out}_j}, k) \star \text{input}(N_i, k)$$

where $\star$ is the valid 2D cross-correlation operator, N is a batch size, C denotes a number of channels, H is a height of input planes in pixels, and W is width in pixels.

This module supports TensorFloat32.

- `stride` controls the stride for the cross-correlation, a single number or a tuple.
- `padding` controls the amount of padding applied to the input. It can be either a string {'valid', 'same'} or a tuple of ints giving the amount of implicit padding applied on both sides.
- `dilation` controls the spacing between the kernel points; also known as the à trous algorithm. It is harder to describe, but this link has a nice visualization of what `dilation` does.
- `groups` controls the connections between inputs and outputs. `in_channels` and `out_channels` must both be divisible by `groups`. For example,
  - At groups=1, all inputs are convolved to all outputs.
  - At groups=2, the operation becomes equivalent to having two conv layers side by side, each seeing half the input channels and producing half the output channels, and both subsequently concatenated.
  - At groups= `in_channels`, each input channel is convolved with its own set of filters (of size $\frac{\text{out\_channels}}{\text{in\_channels}}$).

Convolutional Neural Network

- How is the convolutional layer defined in the deep learning packages?

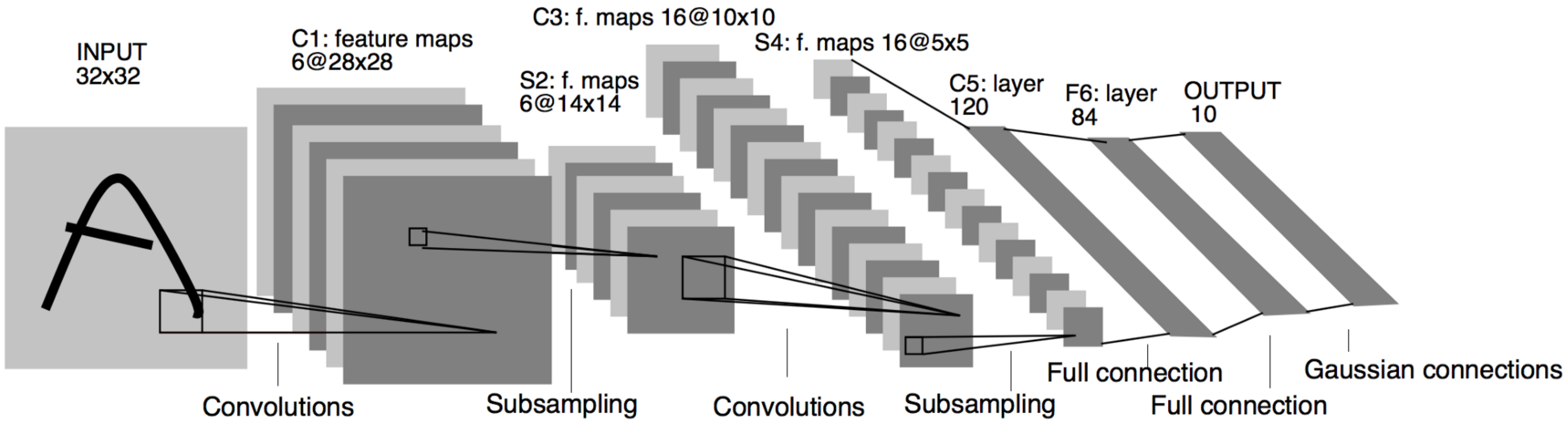
Keras

Conv2D layer

Conv2D class

```
tf.keras.layers.Conv2D(
    filters,
    kernel_size,
    strides=(1, 1),
    padding="valid",
    data_format=None,
    dilation_rate=(1, 1),
    groups=1,
    activation=None,
    use_bias=True,
    kernel_initializer="glorot_uniform",
    bias_initializer="zeros",
    kernel_regularizer=None,
    bias_regularizer=None,
    activity_regularizer=None,
    kernel_constraint=None,
    bias_constraint=None,
    **kwargs
)
```

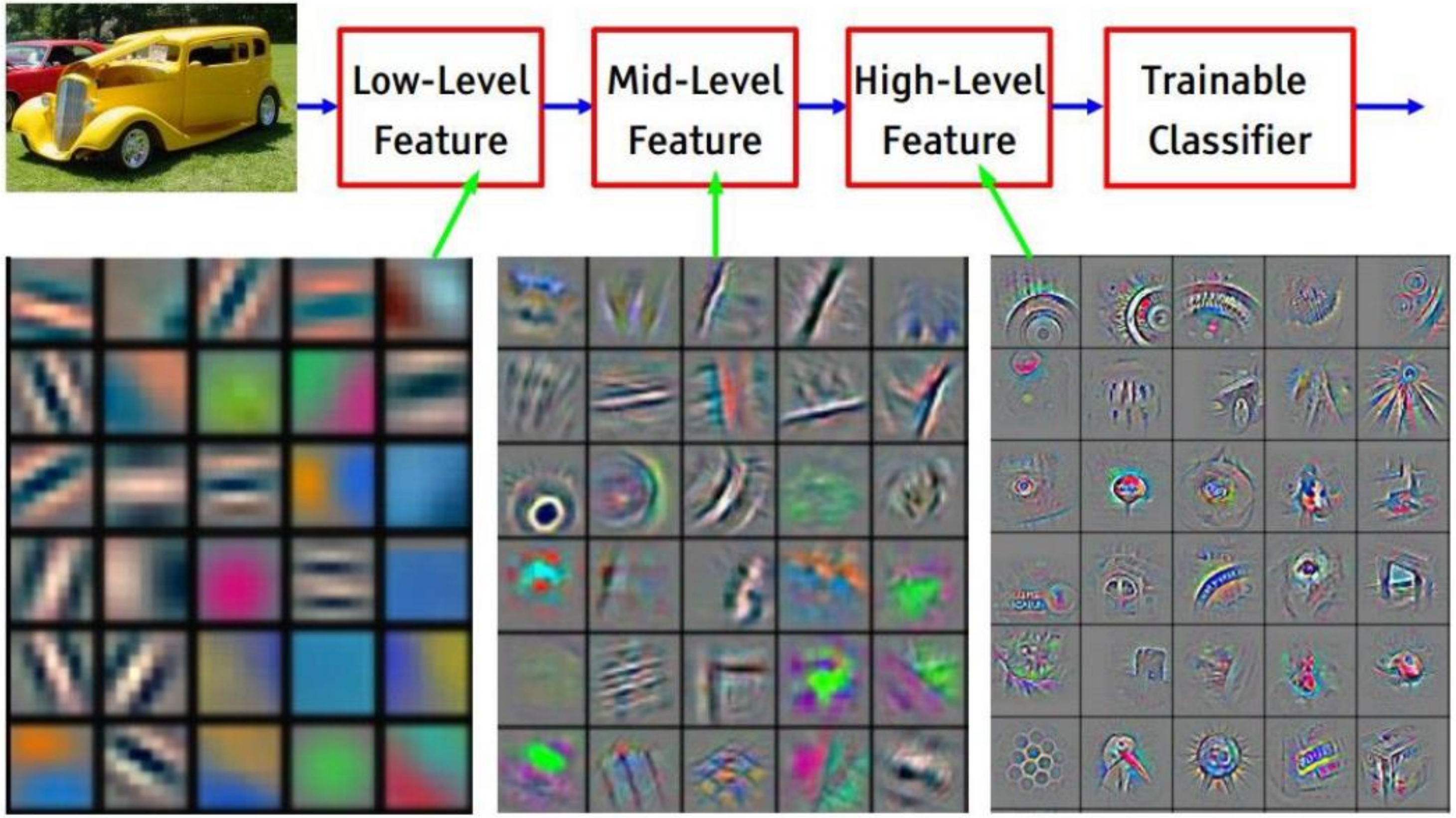
Convolutional Neural Network



- LeNet architecture

Convolutional Neural Network

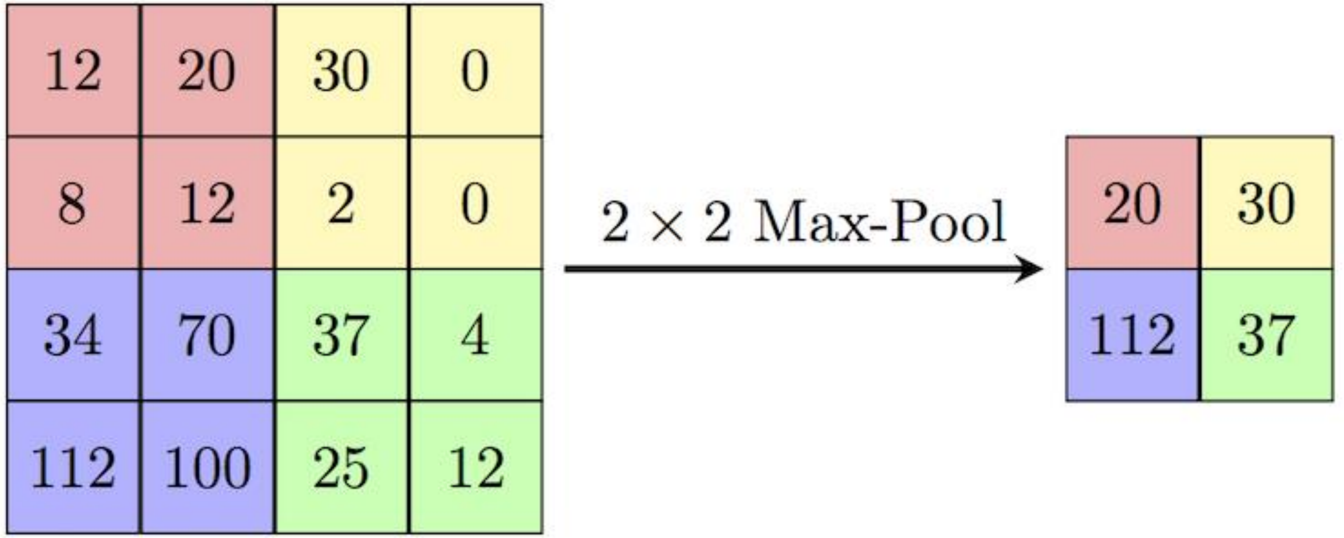
- What is intuition behind stacking convolutional filters?



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Convolutional Neural Network

- If we have large images, how do we make the information more manageable?
- We use pooling for the reducing the size of the image; works on each convolutional map independently
- Most used pooling technique is Max Pooling



Example uses max pooling with stride of 2

Convolutional Neural Network

- Let us look at an example

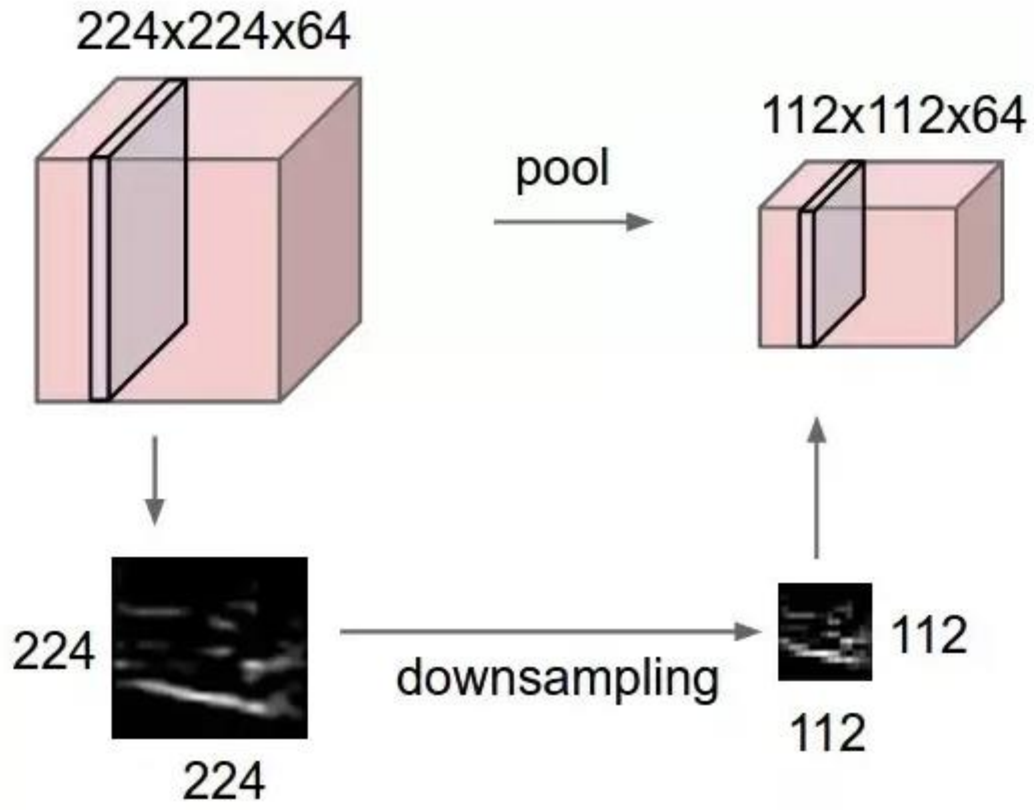
- Here input size of the image is 10 X 10. Assume the image has 3 channels. The max pooling size is 2X2 with stride of 2

- The final output size would be

  $W_{Out} = [(W-F_w)/S] + 1$

  $H_{Out} = [(H-F_w)/S] + 1$

  C_{out} = Number of input channels



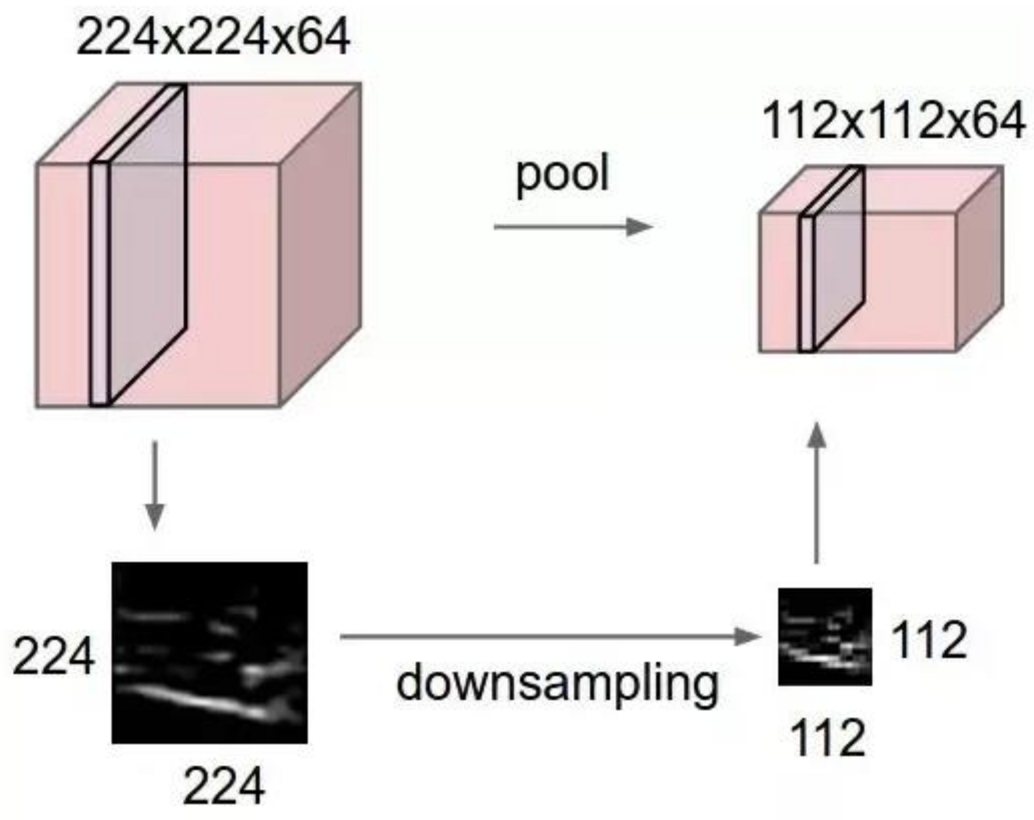
Convolutional Neural Network

- Let us look at an example
- Here input size of the image is 10 X 10. Assume the image has 3 channels. The max pooling size is 2X2 with stride of 2
- The final output size would be

  Wout = [((10-2))/2] + 1 = 5

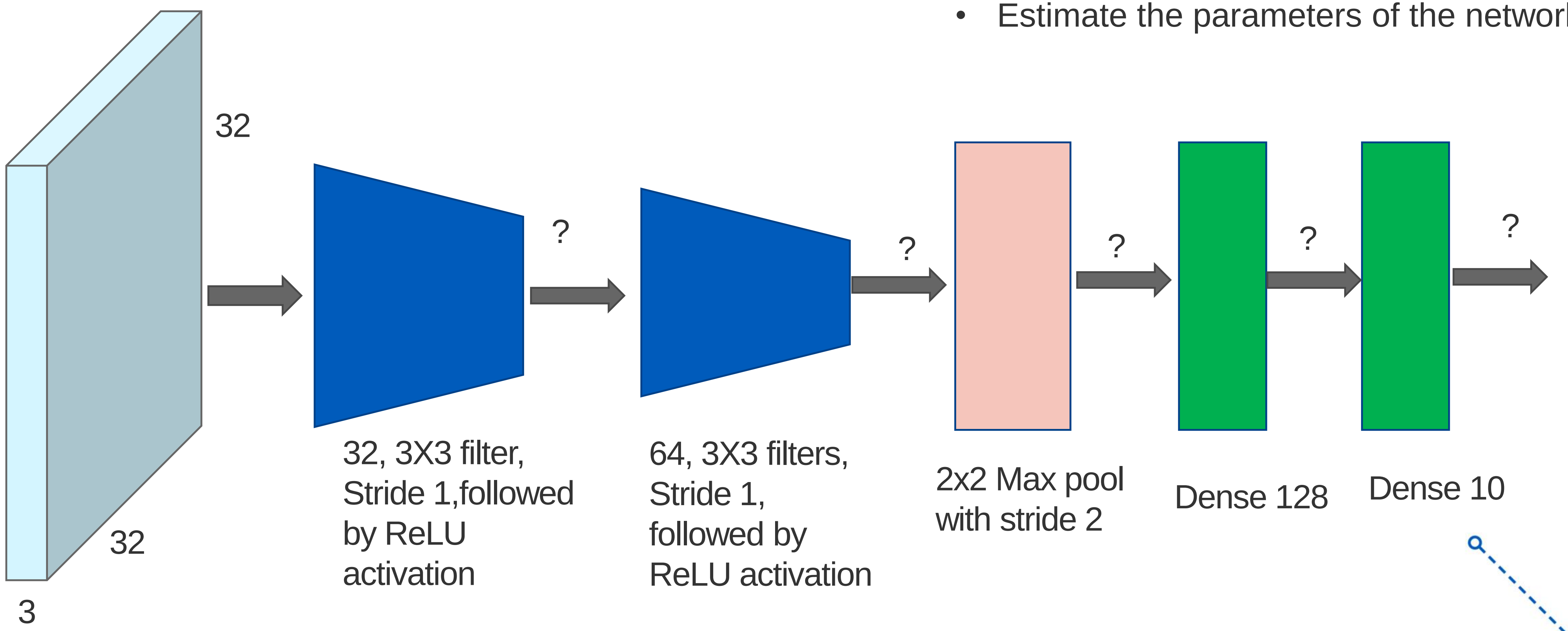
  Hout = [((10-2))/2] + 1 = 5

  Cout = Number of input channels



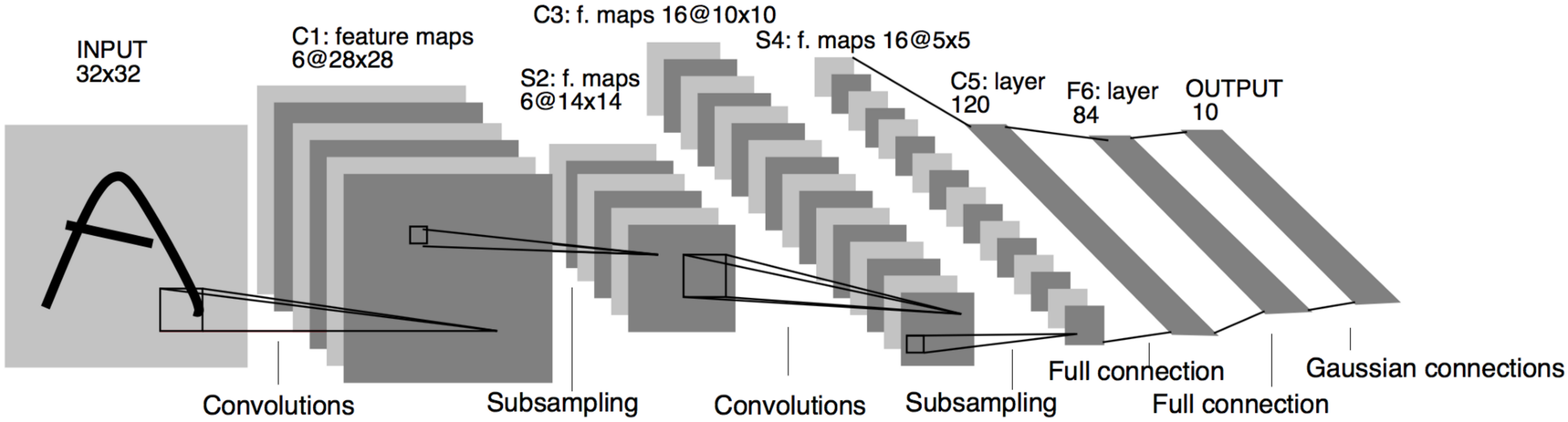
Convolutional Neural Network

- Let us look at a simple CNN architecture
- Estimate the parameters of the network



Convolutional Neural Network

- So how many parameters does this neural network have?

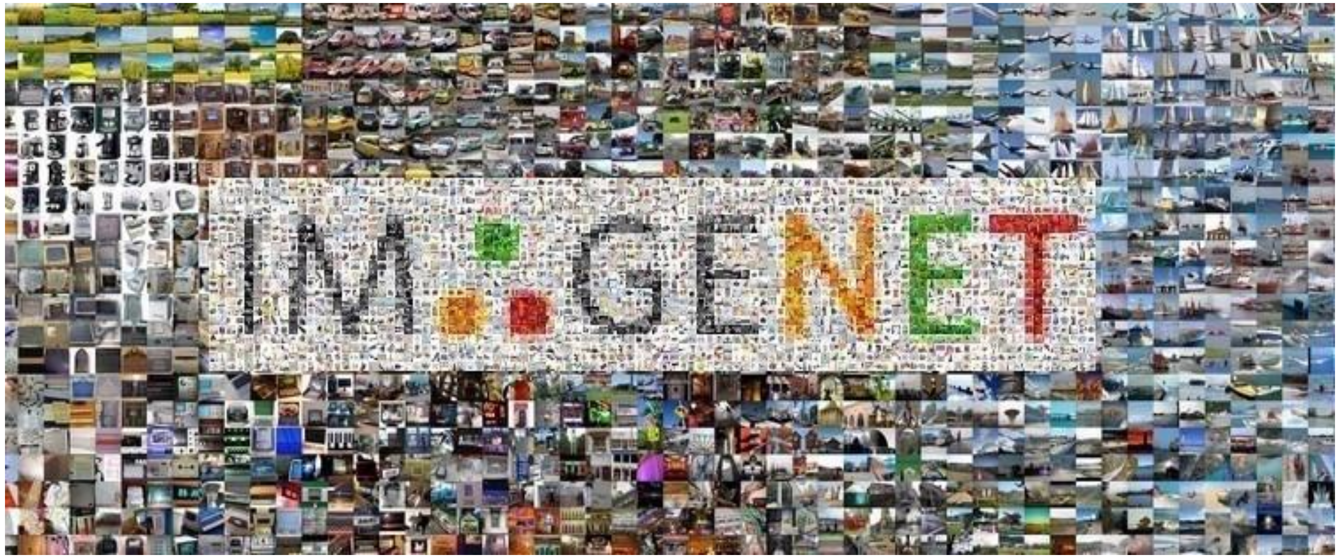


Convolutional Neural Network

- Given a problem that you are trying to solve with Convolutional Neural Networks (Deep Neural Networks in general), in order to improve the performance you can innovate on 4 different aspects
  - Data
    - Representative Train set
    - Augmentation
  - Architectures
    - Design better neural network architectures
  - Loss Formulations
    - Design loss functions to create better representations
  - Training Strategies
    - Choose better techniques to improve the training process, like curriculum learning etc.

ImageNet Dataset

- Introduced first in 2010, the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) evaluates algorithms for object detection and image classification at large scale.

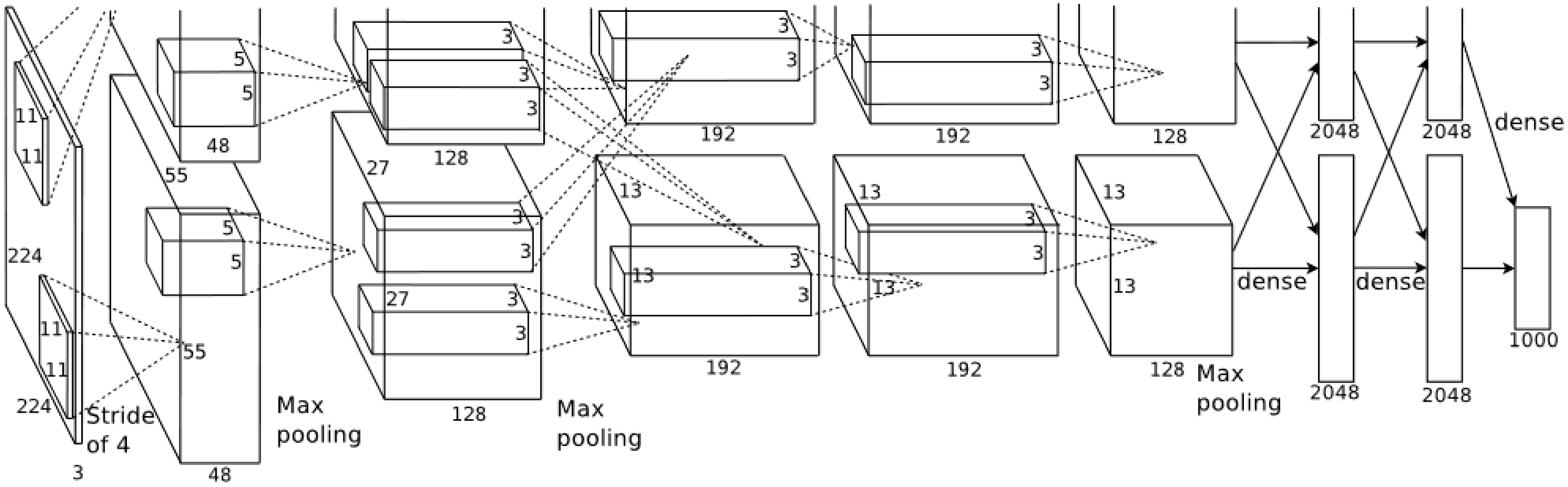


- The ImageNet dataset contains 14,197,122 annotated images

- 21K classes or groups

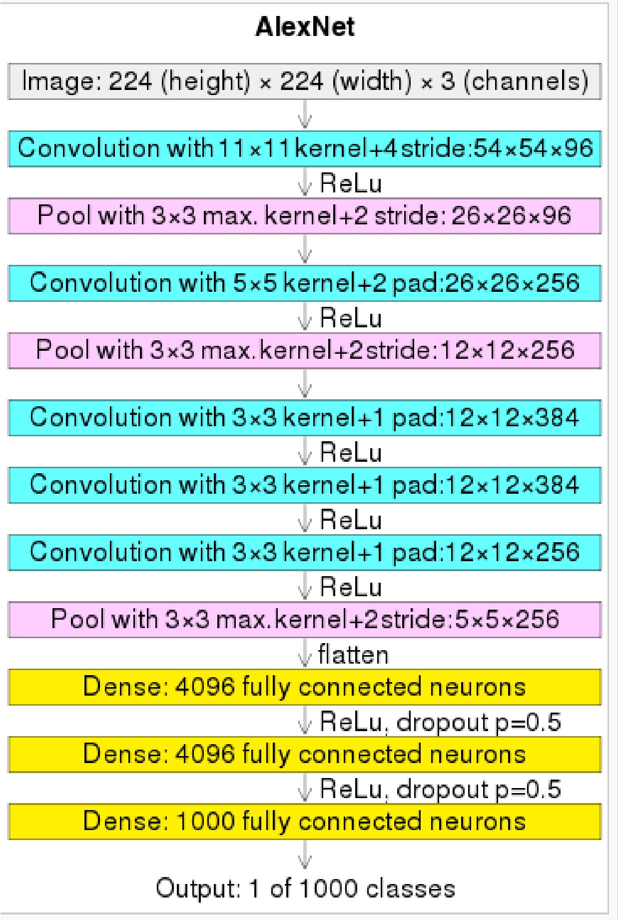
Convolutional Architectures

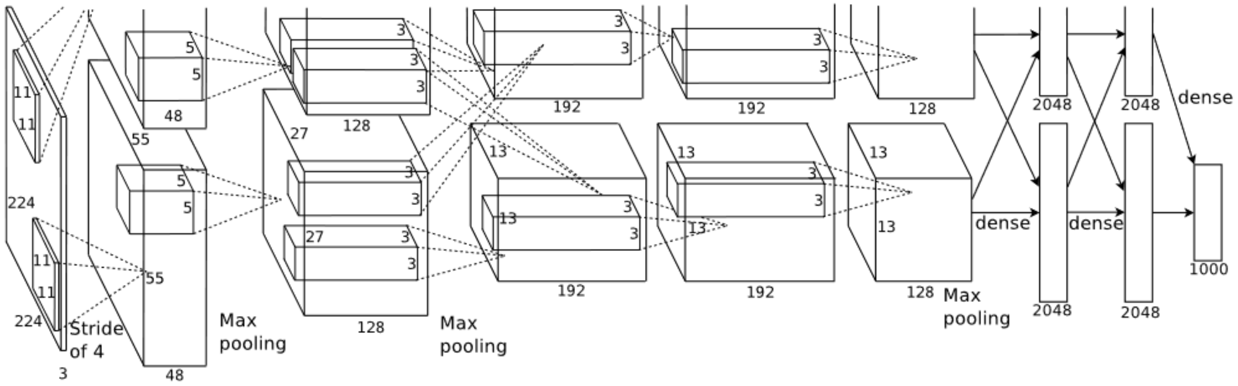
- AlexNet:- [Paper]



Convolutional Architectures

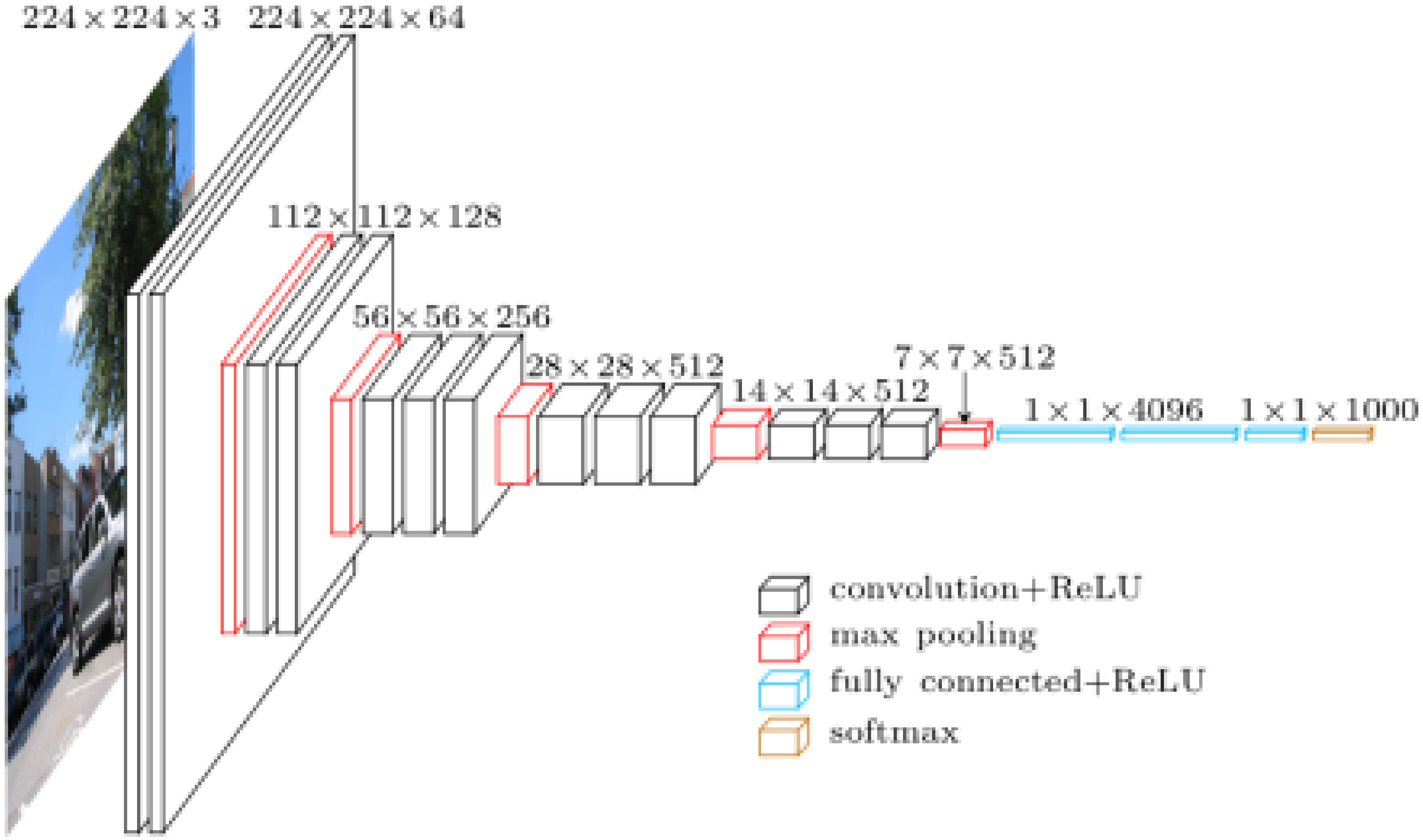
- AlexNet :- Paper





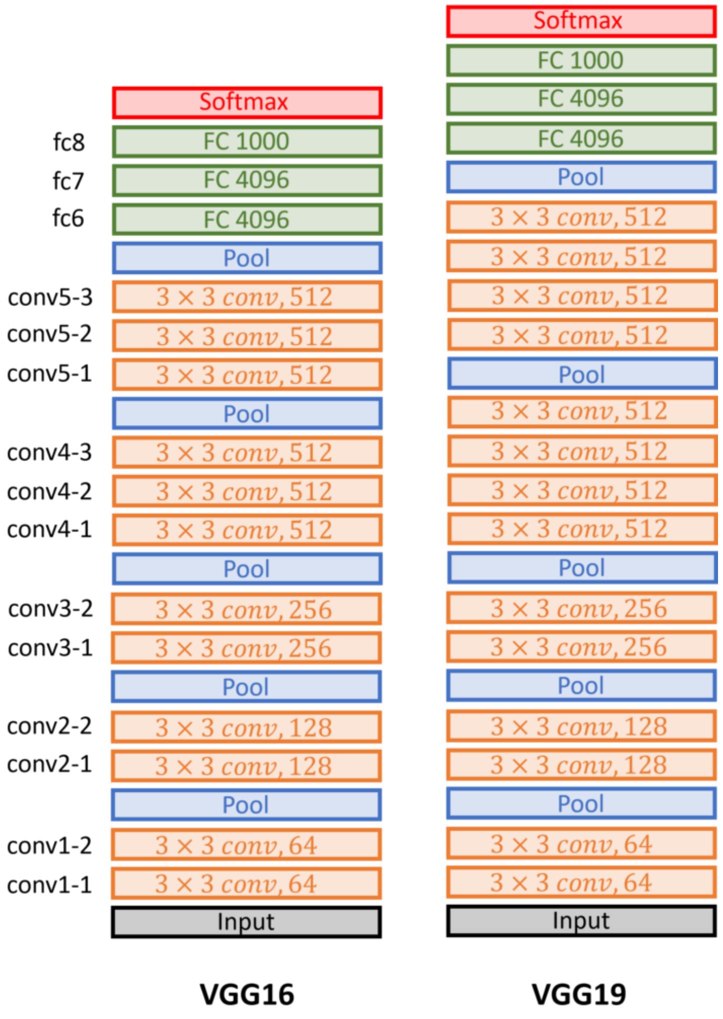
Convolutional Architectures

- VGGNet : Paper



Convolutional Architectures

- VGGNet:- [Paper]
- Two versions: VGG16 and VGG 19
- Why does VGG use filters of less size compared to AlexNet?
- How many parameters does VGGNet have?
- HW. What are the default settings the authors used to train the network



Convolutional Architectures

- VGGNet :- Paper

(not counting biases)

INPUT: [224x224x3] memory: 224*224*3=150K params: 0
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*3)*64 = 1,728
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*64)*64 = 36,864
POOL2: [112x112x64] memory: 112*112*64=800K params: 0
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*64)*128 = 73,728
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*128)*128 = 147,456
POOL2: [56x56x128] memory: 56*56*128=400K params: 0
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*128)*256 = 294,912
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
POOL2: [28x28x256] memory: 28*28*256=200K params: 0
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*256)*512 = 1,179,648
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
POOL2: [14x14x512] memory: 14*14*512=100K params: 0
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
POOL2: [7x7x512] memory: 7*7*512=25K params: 0
FC: [1x1x4096] memory: 4096 params: 7*7*512*4096 = 102,760,448
FC: [1x1x4096] memory: 4096 params: 4096*4096 = 16,777,216
FC: [1x1x1000] memory: 1000 params: 4096*1000 = 4,096,000

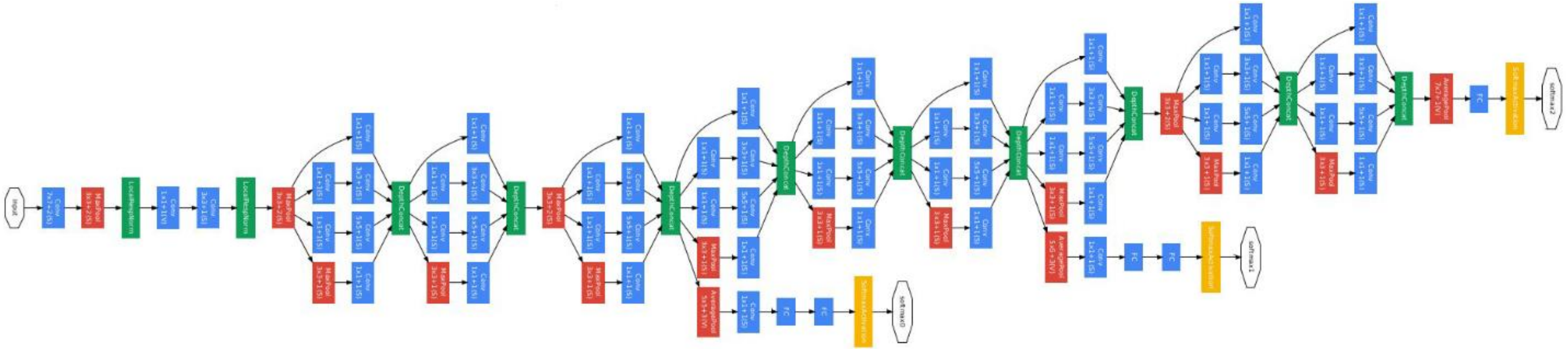
TOTAL memory: 24M * 4 bytes ~= 93MB / image (only forward! ~*2 for bwd)
TOTAL params: 138M parameters

ConvNet Configuration			
B	C	D	
13 weight layers	16 weight layers	16 weight layers	19
put (224 × 224 RGB image)			
conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	cc cc
maxpool			
conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	co co
maxpool			
conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	co co co **co**
maxpool			
conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	co co co **co**
maxpool			
conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	co co co **co**
maxpool			
FC-4096			
FC-4096			
FC-1000			
soft-max			

Ref:http://cs231n.stanford.edu/slides/2016/winter1516_lecture7.pdf

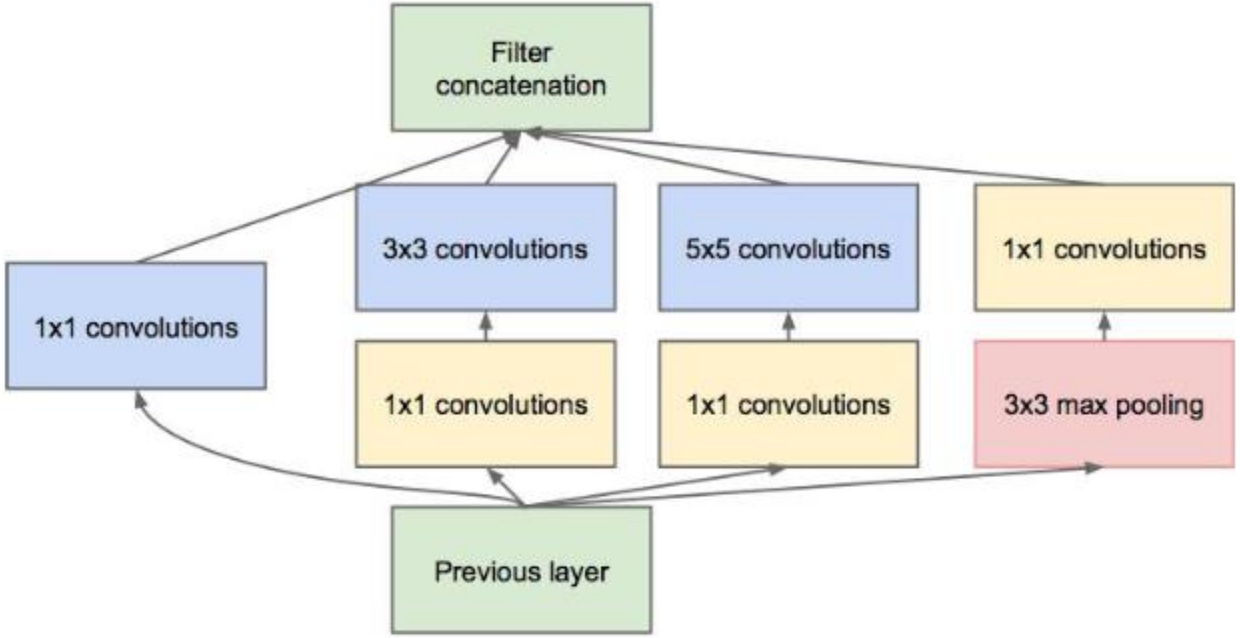
Convolutional Architectures

- GoogleNet:- [Paper]



Convolutional Architectures

- GoogleNet :- Paper
- Introduced Inception module
- Multiple branches for the gradient to flow
- The inception block is used to increase the depth of the network
- The entire network has only 5 Million parameters
- Auxiliary classifiers to improve gradient flow to initial layers



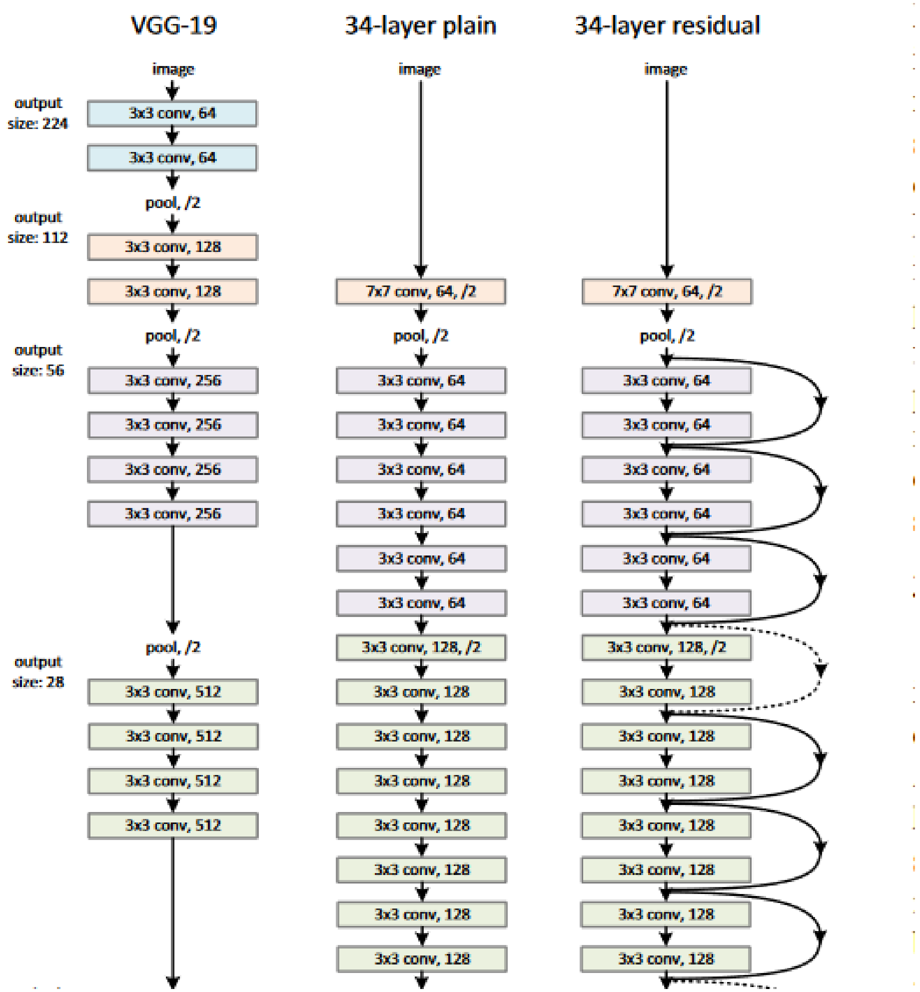
Convolutional Architectures

- GoogleNet :- Paper

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

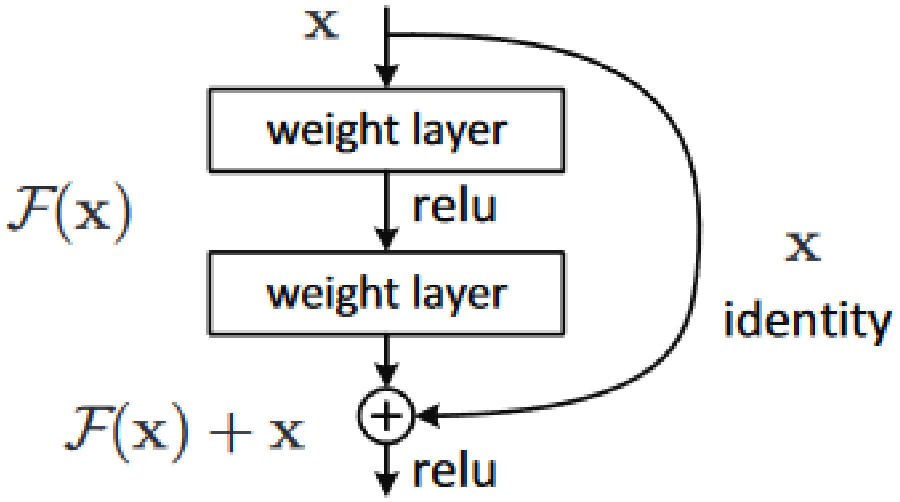
Convolutional Architectures

- ResNet :- [Paper]



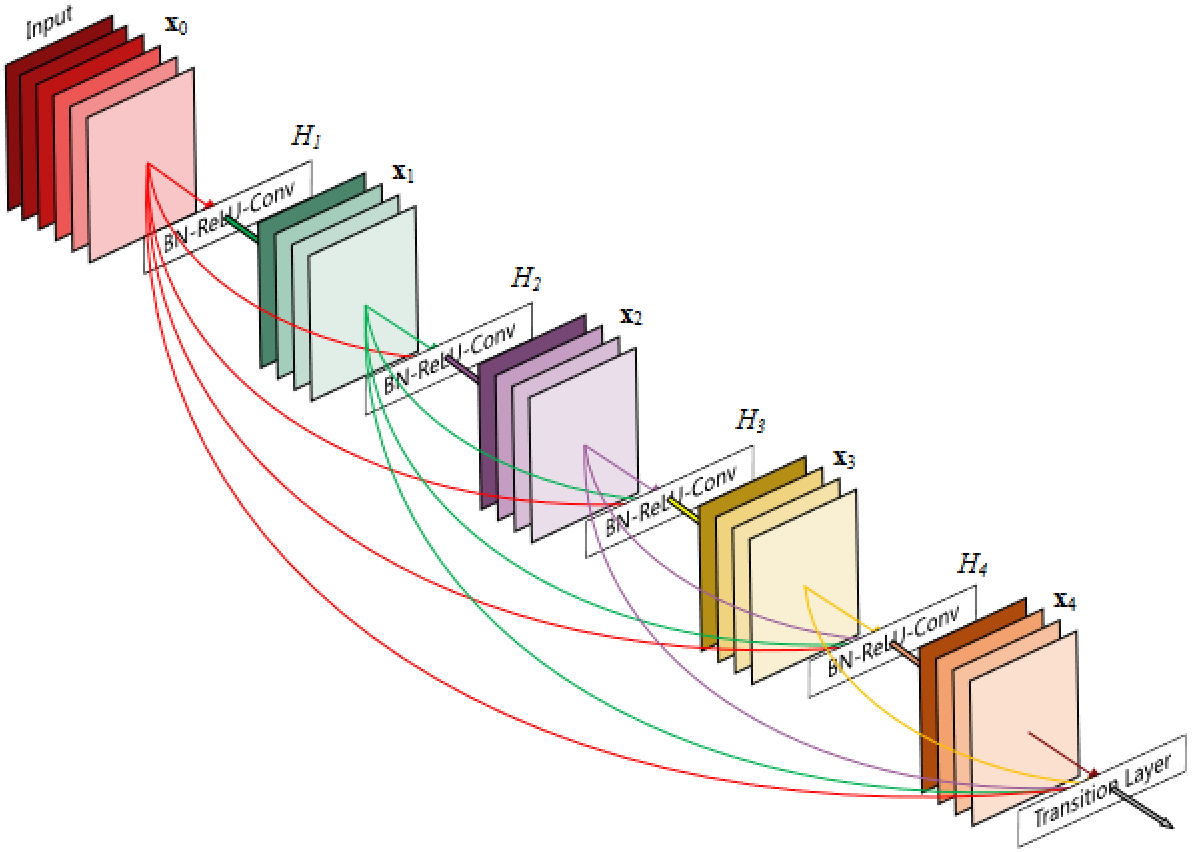
Convolutional Architectures

- ResNet : - Paper
- Introduced the residual block
- Improved gradient flow due to skipped connections
- Increased the number of layers to 1K
- One of the most commonly used network backbones



Convolutional Architectures

- DenseNet : - Paper
- Introduced the Dense Blocks, which connect output of each layer to all the subsequent layers inside the Dense Block
- Instead of addition of Feature maps, DenseNets use concatenation
- Performance improvement over ResNet
- Commonly used backbone




ANY QUESTIONS
?

References

- http://proceedings.mlr.press/v28/sutskever13.html
- This lecture is inspired from cse 231n https://www.youtube.com/watch?v=i94OvYb6noo&t=2051
- http://neuralnetworksanddeeplearning.com/chap5.html
- https://ruder.io/optimizing-gradient-descent/
- http://cs231n.stanford.edu/
- https://github.com/vdumoulin/conv_arithmetic
- https://www.google.com/imgres?imgurl=https%3A%2F%2Fmiro.medium.com%2Fmax%2F1400%2F1*Di4V69e4gC16ooF6PZPt-A.png&imgrefurl=https%3A%2F%2Ftowardsdatascience.com%2Feverything-you-need-to-know-about-neural-networks-and-backpropagation-machine-learning-made-easy-e5285bc2be3a&tbnid=PFKzBNejYXM4hM&vet=12ahUKEwism4aItO3yAhVrqnIEHSHUCNkQMyhDegQIARBf..i&docid=OXeL--Z4fRwo6M&w=1250&h=1057&q=neural%20networks%20with%20math&hl=en&client=firefox-b-1-d&ved=2ahUKEwism4aItO3yAhVrqnIEHSHUCNkQMyhDegQIARBf