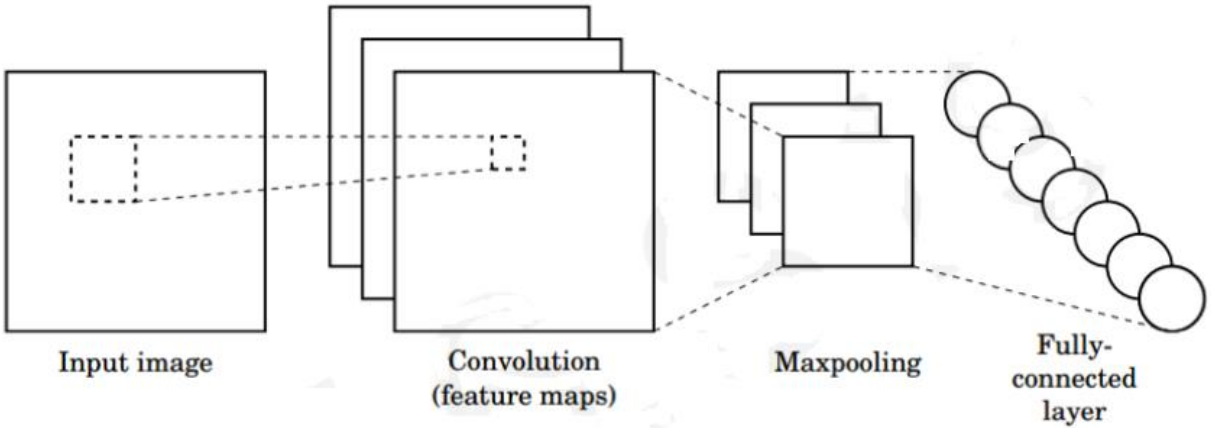
Lecture 7

Deep Learning

Convolution Neural Networks (CNNs)

CNN for Classification



1. **Convolution**: Apply filters to generate feature maps.
2. **Non-linearity**: Often ReLU.
3. **Pooling**: Downsampling operation on each feature map.

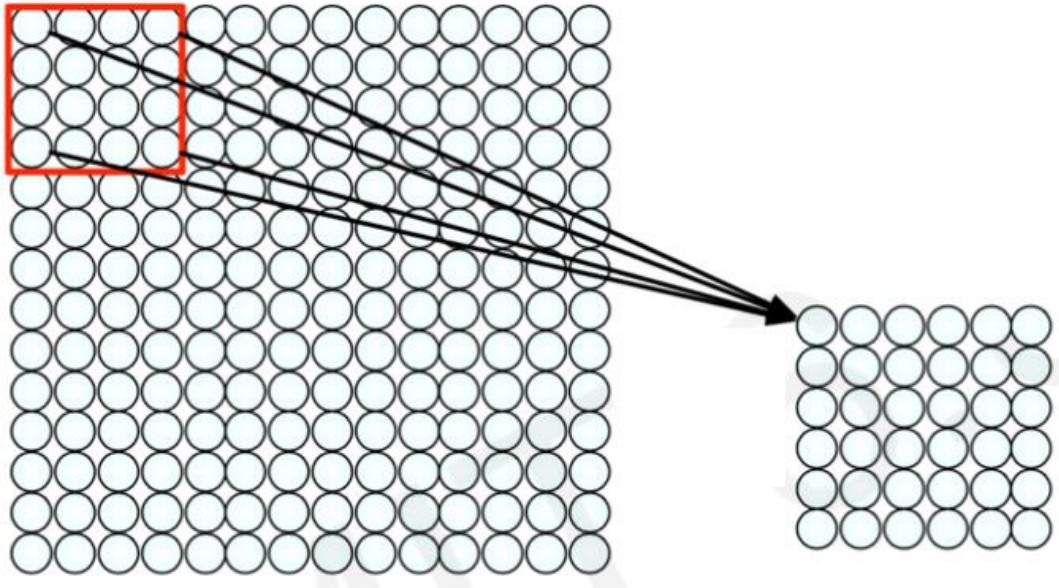
`tf.keras.layers.Conv2D`

`tf.keras.activations.*`

`tf.keras.layers.MaxPool2D`

Train model with image data.
Learn weights of filters in convolutional layers.

Convolution Layers: Local connectivity



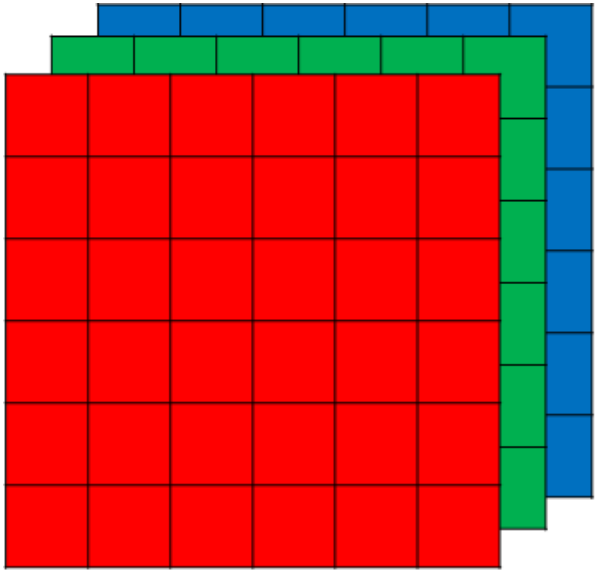
`tf.keras.layers.Conv2D`

For a neuron in hidden layer:

- Take inputs from patch
- Compute weighted sum
- Apply bias

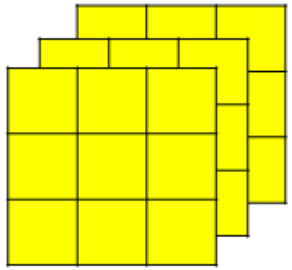
1) applying a window of weights
2) computing linear combinations
3) activating with non-linear function

Convolution for RGB Images



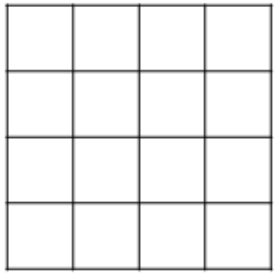
6 x 6 x 3

*



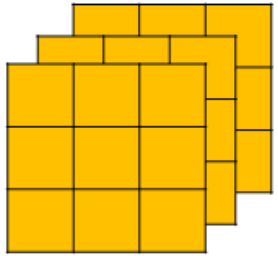
3 x 3 x 3

=



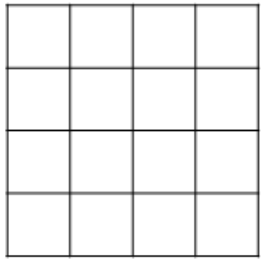
4 x 4

*



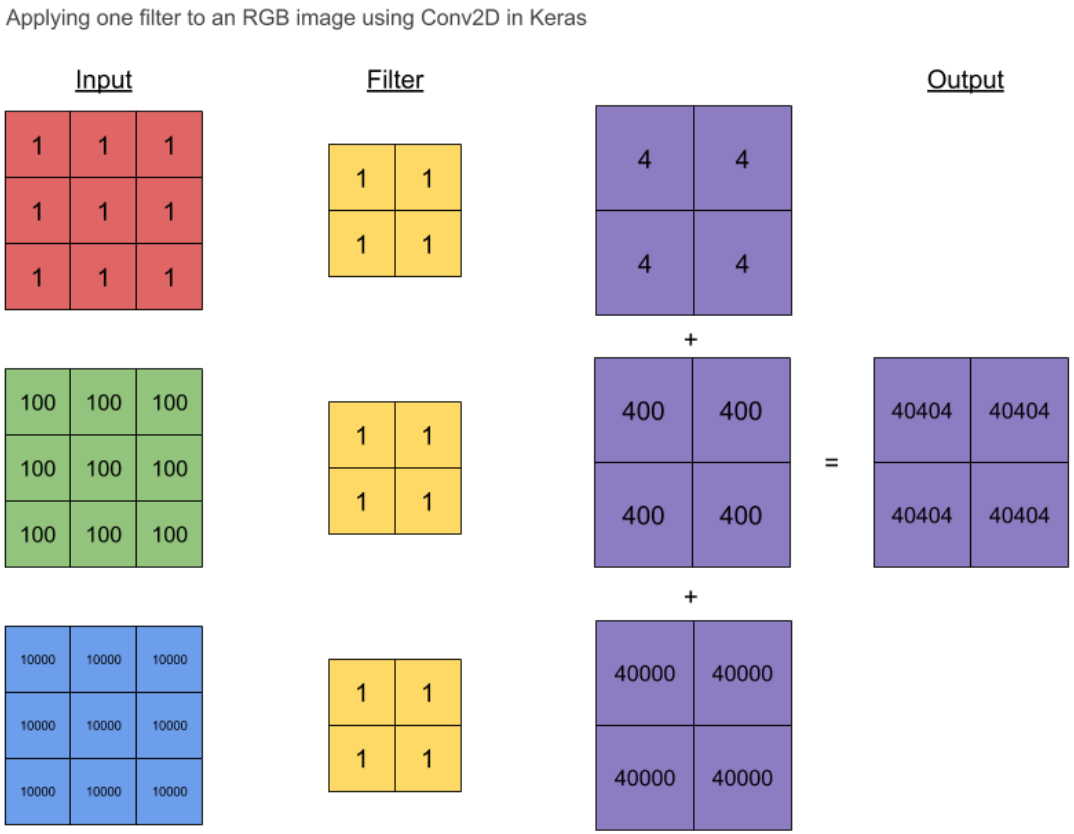
3 x 3 x 3

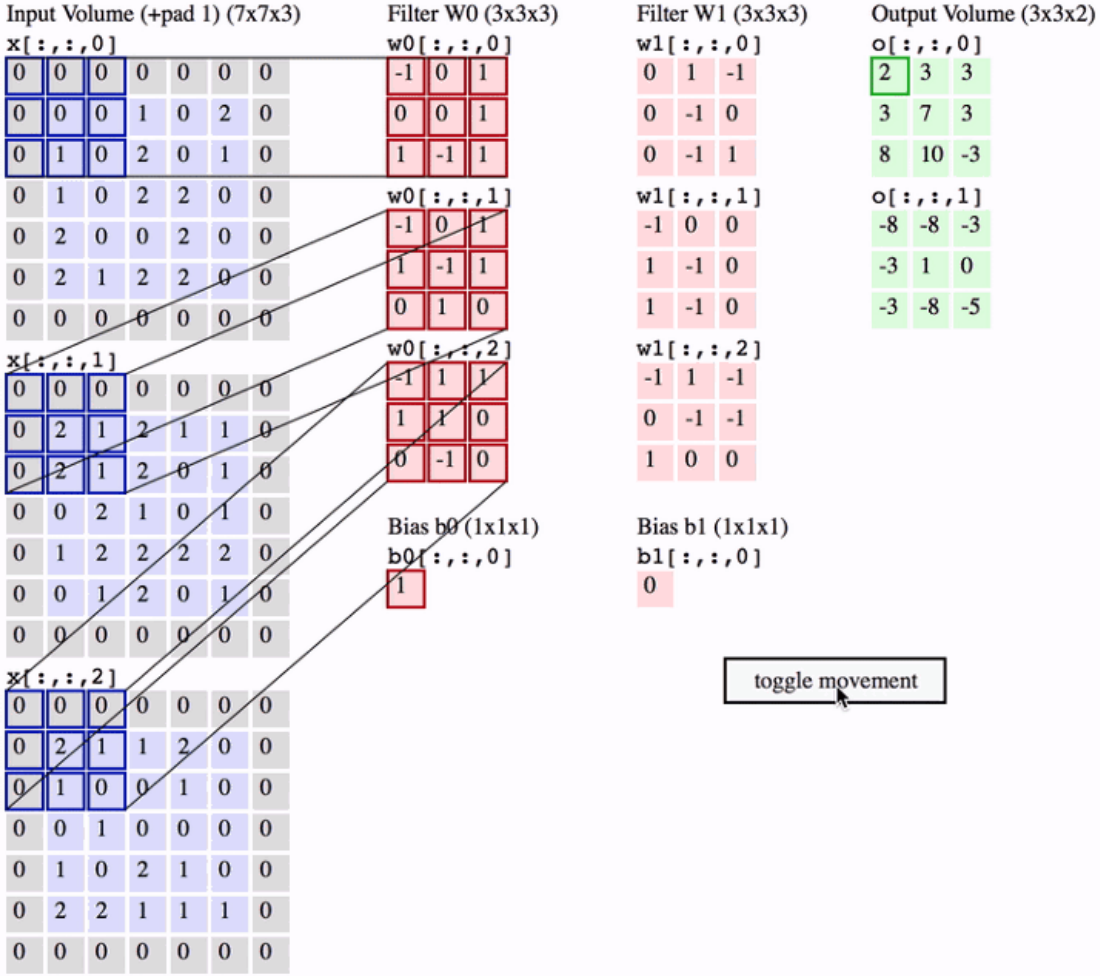
=



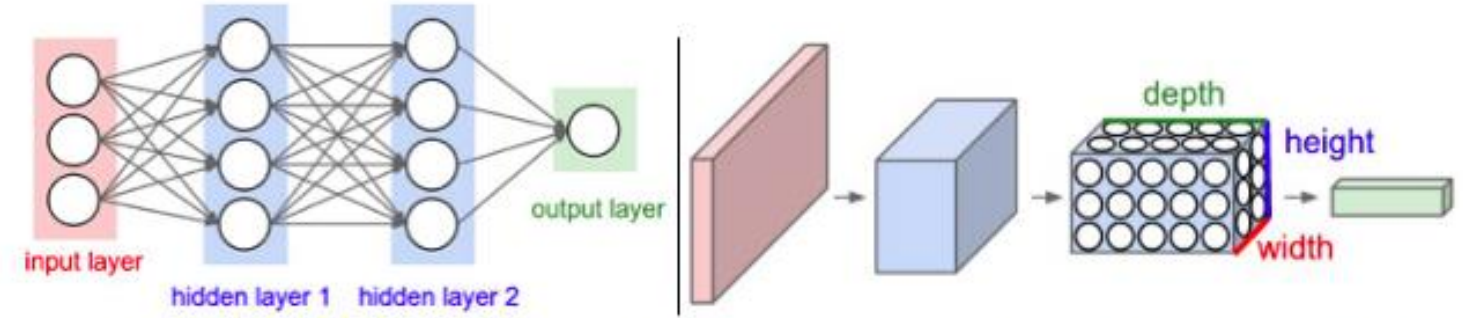
4 x 4

Convolution for RGB images




Input Volume (+pad 1) (7x7x3)
x[:,:,0]
0 0 0 0 0 0 0
0 0 0 1 0 2 0
0 1 0 2 0 1 0
0 1 0 2 2 0 0
0 2 0 0 2 0 0
0 2 1 2 2 0 0
0 0 0 0 0 0 0
x[:,:,1]
0 0 0 0 0 0 0
0 2 1 2 1 1 0
0 2 1 2 0 1 0
0 0 2 1 0 1 0
0 1 2 2 2 2 0
0 0 1 2 0 1 0
0 0 0 0 0 0 0
x[:,:,2]
0 0 0 0 0 0 0
0 2 1 1 2 0 0
0 1 0 0 1 0 0
0 0 1 0 0 0 0
0 1 0 2 1 0 0
0 2 2 1 1 1 0
0 0 0 0 0 0 0
Filter W0 (3x3x3)
w0[:,:,0]
-1 0 1
0 0 1
1 -1 1
w0[:,:,1]
-1 0 1
1 -1 1
0 1 0
w0[:,:,2]
-1 1 1
1 1 0
0 -1 0
Bias b0 (1x1x1)
b0[:,:,0]
1
Filter W1 (3x3x3)
w1[:,:,0]
0 1 -1
0 -1 0
0 -1 1
w1[:,:,1]
-1 0 0
1 -1 0
1 -1 0
w1[:,:,2]
-1 1 -1
0 -1 -1
1 0 0
Bias b1 (1x1x1)
b1[:,:,0]
0
Output Volume (3x3x2)
o[:,:,0]
2 3 3
3 7 3
8 10 -3
o[:,:,1]
-8 -8 -3
-3 1 0
-3 -8 -5
toggle movement

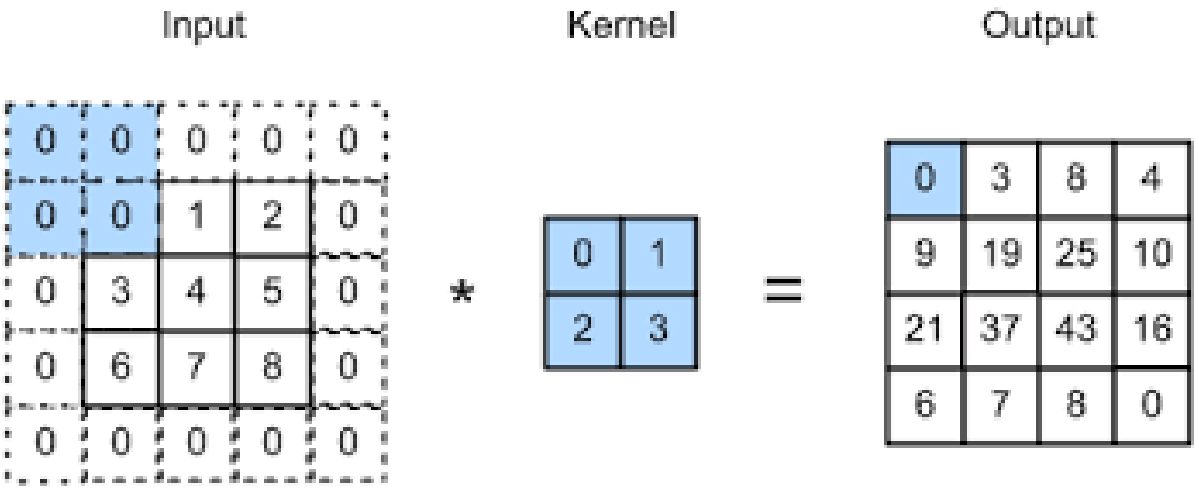
Normal Neural Network vs CNN



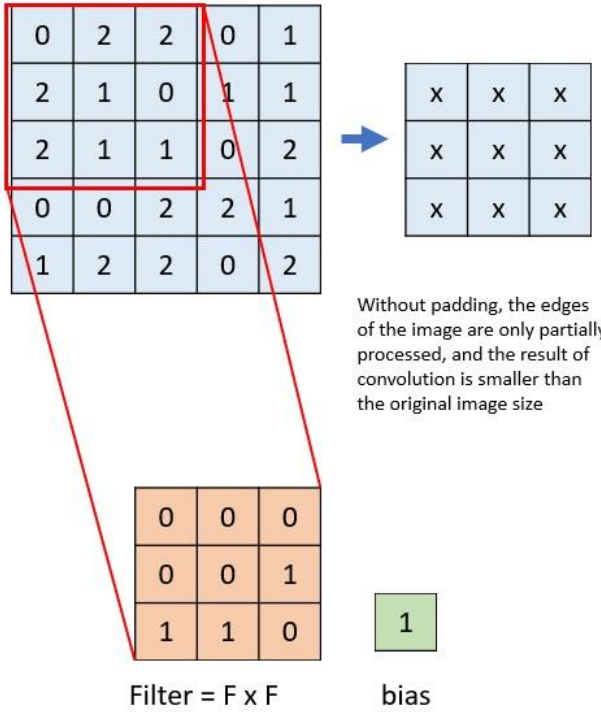
Left: A regular 3-layer Neural Network. Right: A ConvNet arranges its neurons in three dimensions (width, height, depth), as visualized in one of the layers. Every layer of a ConvNet transforms the 3D input volume to a 3D output volume of neuron activations. In this example, the red input layer holds the image, so its width and height would be the dimensions of the image, and the depth would be 3 (Red, Green, Blue channels).

Padding

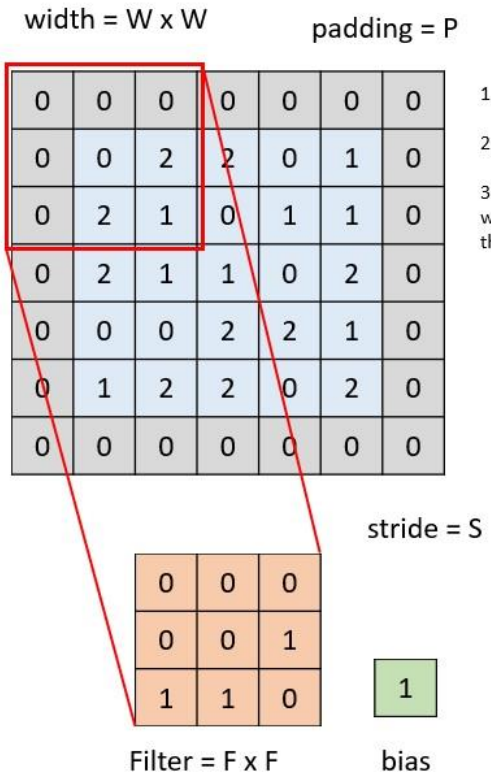
Padding


Input
Kernel
Output
0 0 0 0 0
0 0 1 2 0
0 3 4 5 0
0 6 7 8 0
0 0 0 0 0
*
0 1
2 3
=
0 3 8 4
9 19 25 10
21 37 43 16
6 7 8 0

Padding



Without padding, the edges of the image are only partially processed, and the result of convolution is smaller than the original image size



1. Convolution result size = (W – F + 2P) / S + 1

2. (W – F + 2P) / S + 1 should be an integer

3. If you set S = 1, then setting P = (F – 1) / 2 will generate convolution result size equal to the image size.

Padding

- **Convolution Problems without Padding**
  - output shrinkage problem
  - throw away information from edges

- **Valid and same Convolution**
  - “valid” (without padding)
  - “same”: pad so that output size is same as input size

Strided Convolution

Strided Convolutions

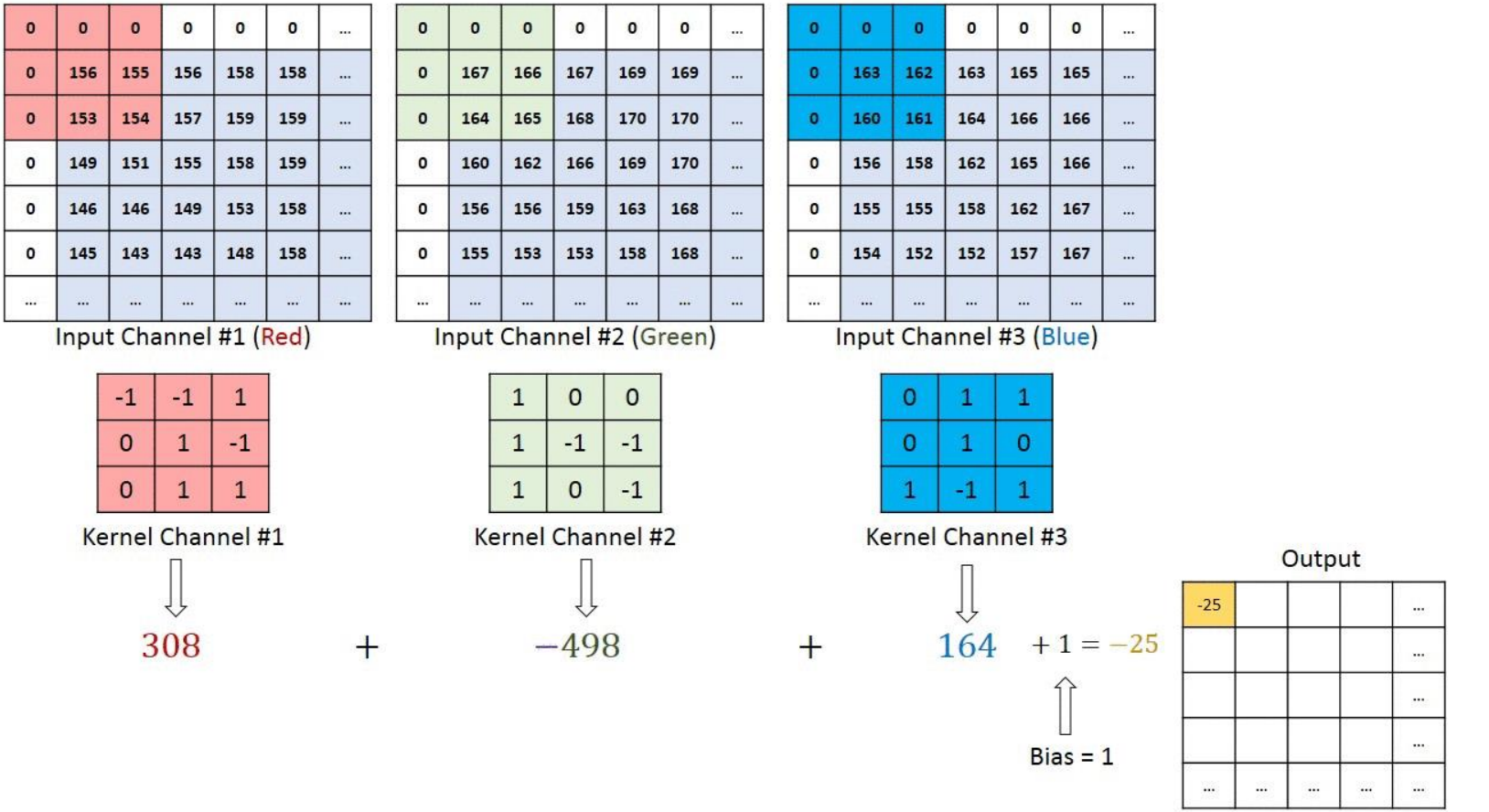
2	3	7	4	6	2	9
6	6	9	8	7	4	3
3	4	8	3	8	9	7
7	8	3	6	6	3	4
4	2	1	8	3	4	6
3	2	4	1	9	8	3
0	1	3	9	2	1	4

$*$

3	4	4
1	0	2
-1	0	3

$=$

Strided Convolutions



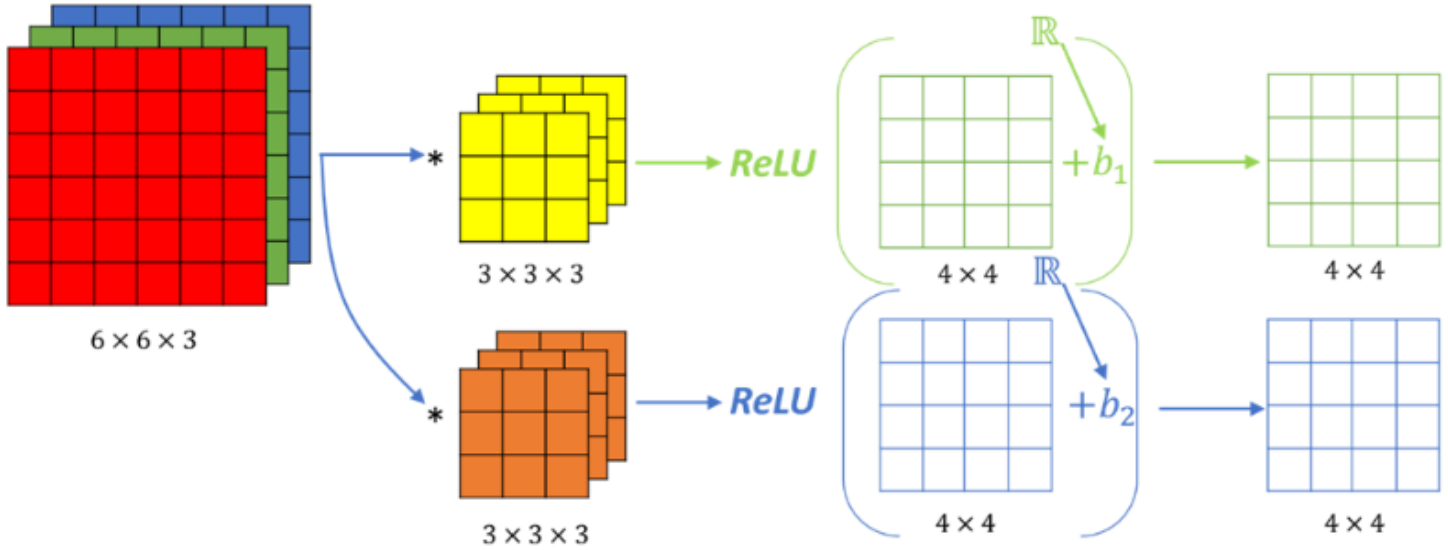
Overall summary

$n \times n$ image $\quad f \times f$ filter

padding p $\quad$ stride s

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

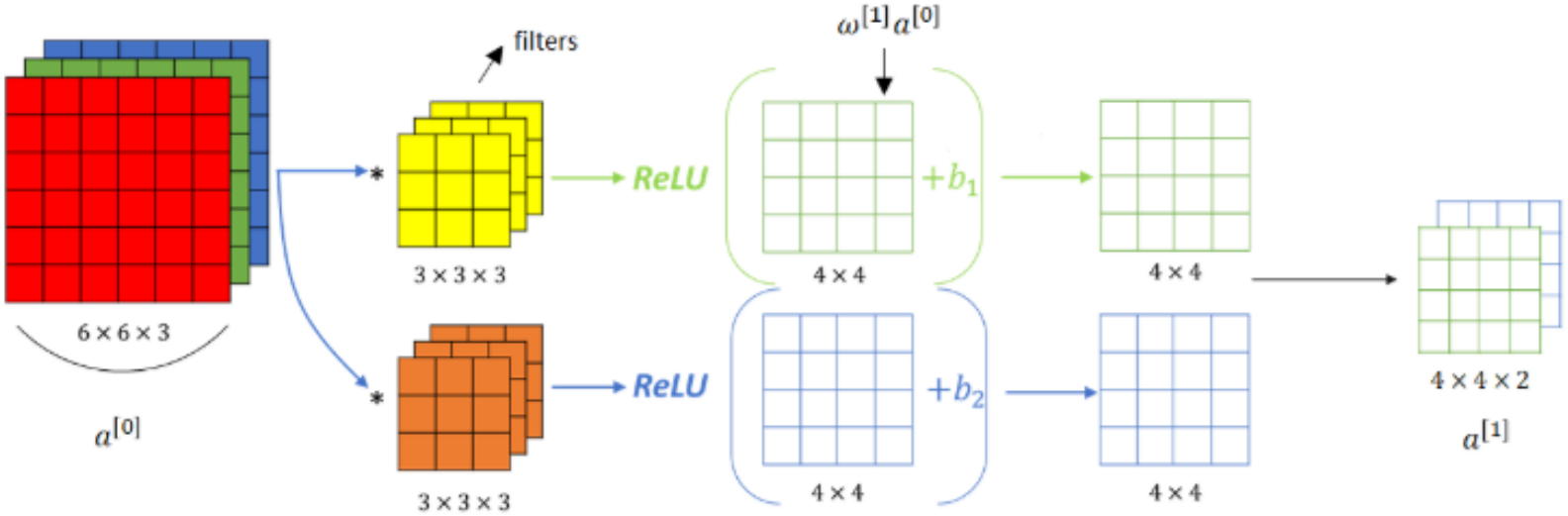
One Layer of convolution NN



The result of convolution with two filters

- Input -> Filtering -> output_tmp -> add bias -> relu -> final ouput
- Parameters are the number of elements in these filters cubic. NOT the number of weights in the full connected neural network. Less parameters + less overfitting + easier to learn + more robust.

Next, we will repeat previous steps. Then we stack end up with a $4 \times 4 \times 2$ output. This computation have gone from $6 \times 6 \times 3$ to a $4 \times 4 \times 2$ and it represents one layer of a convolutional neural network.

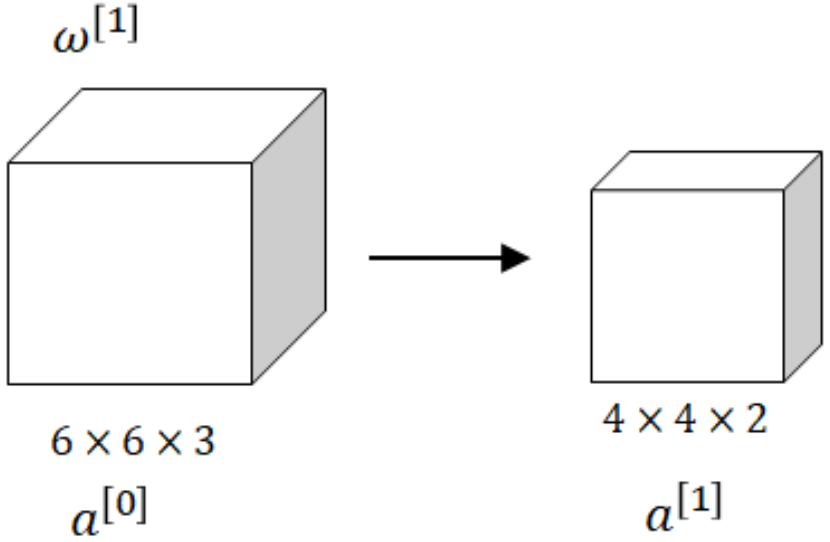


The result of a convolution of $6 \times 6 \times 3$ with two $3 \times 3 \times 3$ is a volume of dimension $4 \times 4 \times 2$

In neural networks one step of a forward propagation step was: $Z^{[1]} = W^{[1]} \times a^{[0]} + b^{[1]}$, where $a^{[0]} = x$. Then we applied the non-linearity to get $a^{[1]} = g^{Z^{[l]}}$. The same idea we will apply in a layer of the Convolutional Neural Network.

$$Z^{[1]} = \omega^{[1]} a^{[0]} + b^{[1]}$$

$$a^{[1]} = g^{(Z^{[1]})}$$



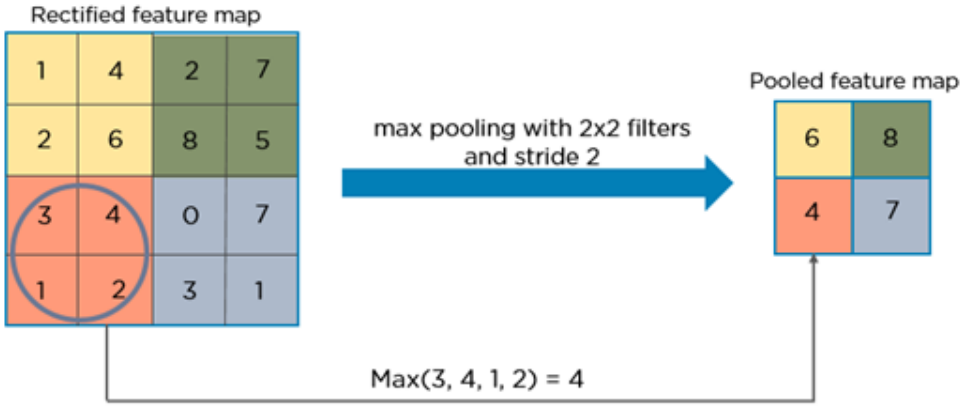
A convolutional layer

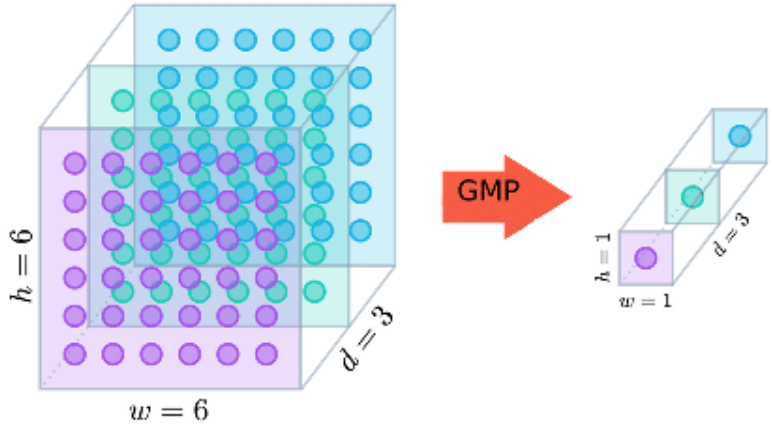
Types of Layers in CNN

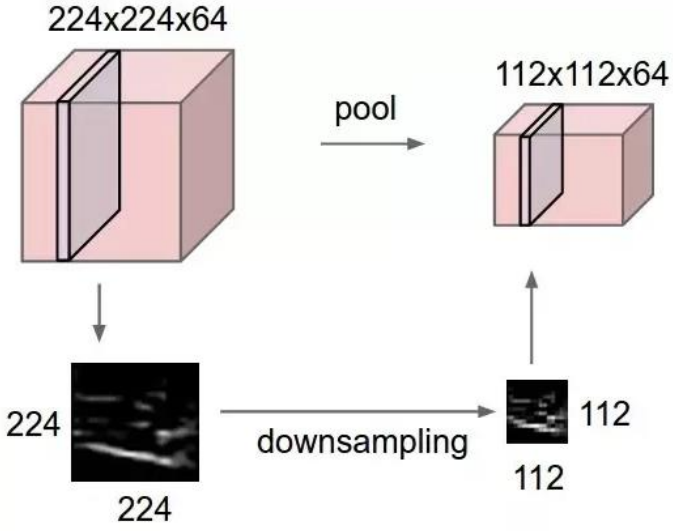
- Convolution
- Pooling
- Fully connected

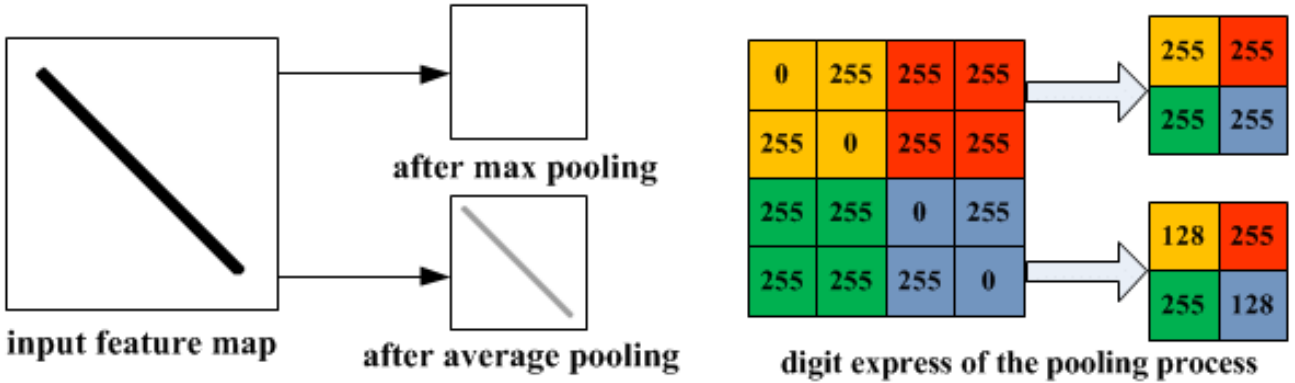
Pooling Layers

Max Pooling

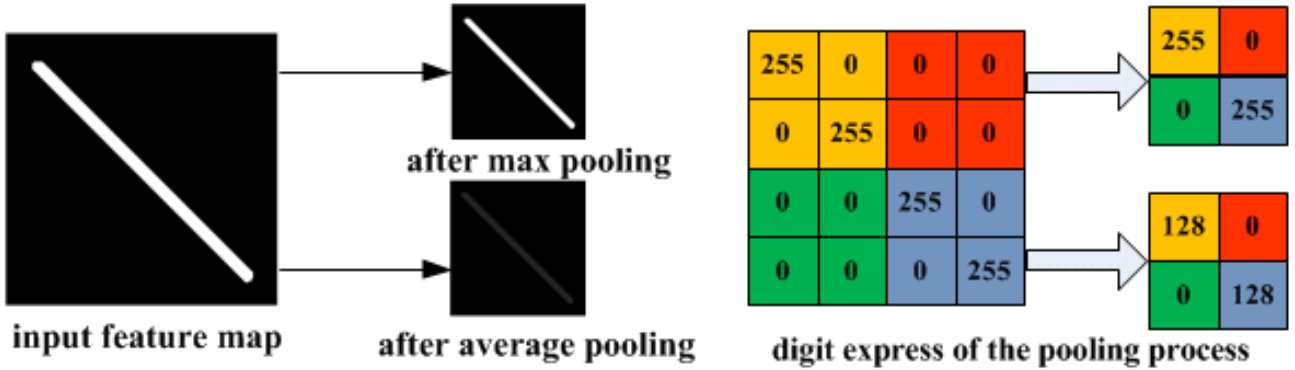

Rectified feature map
1 4 2 7
2 6 8 5
3 4 0 7
1 2 3 1
max pooling with 2x2 filters and stride 2
Pooled feature map
6 8
4 7
Max(3, 4, 1, 2) = 4


$h = 6$
$w = 6$
$d = 3$
GMP
$h = 1$
$w = 1$
$d = 3$


224x224x64
pool
112x112x64
224
224
downsampling
112
112

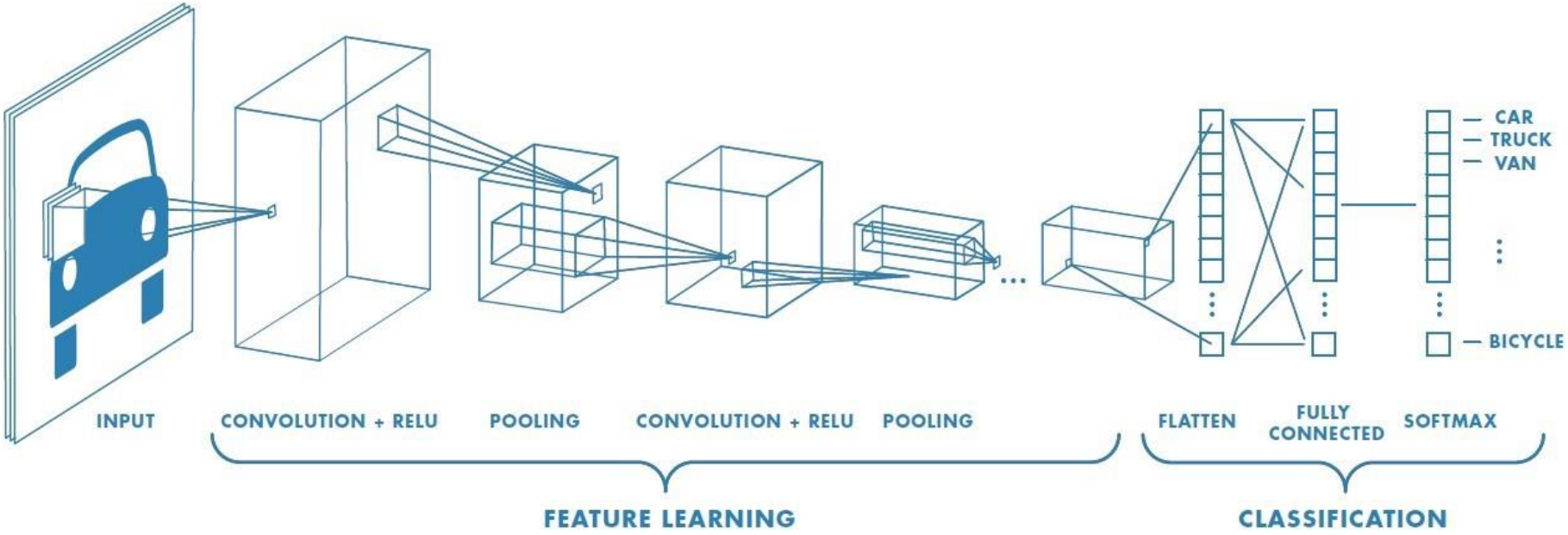


(a) Illustration of max pooling drawback



(b) Illustration of average pooling drawback

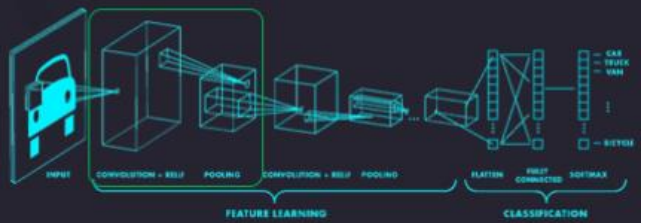
CNNs for Classification



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

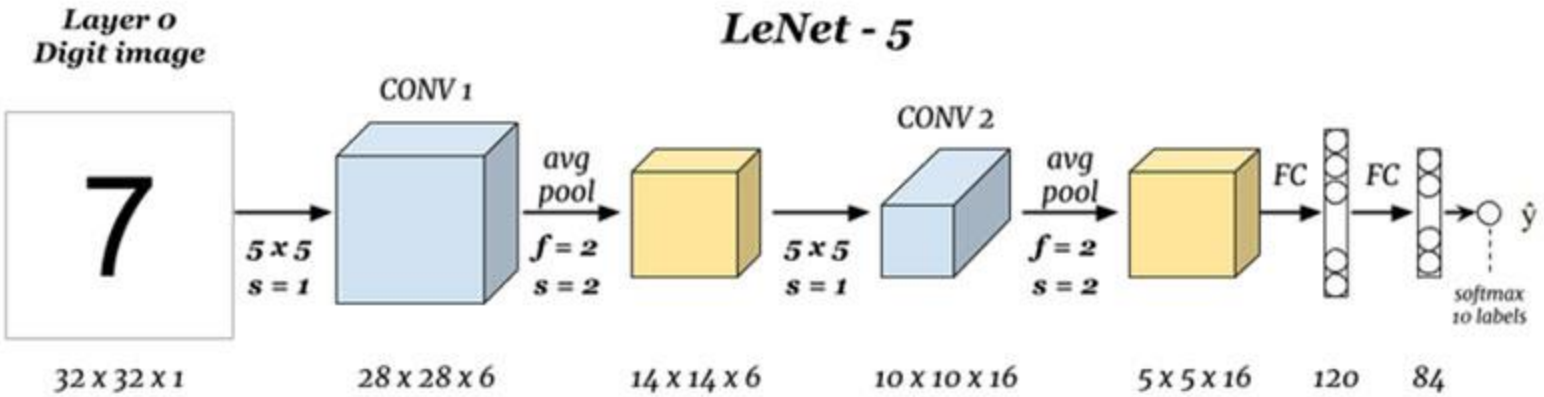
CNNs for Classification

```python
import tensorflow as tf

def generate_model():
    model = tf.keras.Sequential([
        # first convolutional layer
        tf.keras.layers.Conv2D(32, filter_size=3, activation='relu'),
        tf.keras.layers.MaxPool2D(pool_size=2, strides=2),

        # second convolutional layer
        tf.keras.layers.Conv2D(64, filter_size=3, activation='relu'),
        tf.keras.layers.MaxPool2D(pool_size=2, strides=2),

        # fully connected classifier
        tf.keras.layers.Flatten(),
        tf.keras.layers.Dense(1024, activation='relu'),
        tf.keras.layers.Dense(10, activation='softmax')  # 10 outputs
    ])
    return model
```



LeNet-5

Convolution+pool = one layer.
LeNet-5: Input -> layer 1 -> layer 2 -> fully connected layer 3 -> fully connected layer 4 -> softmax -> output



Y. LeCun, L. Bottou, Y. Bengio and P. Haffner: Gradient-Based Learning Applied to Document Recognition, Proceedings of the IEEE, 86(11):2278-2324, November 1998

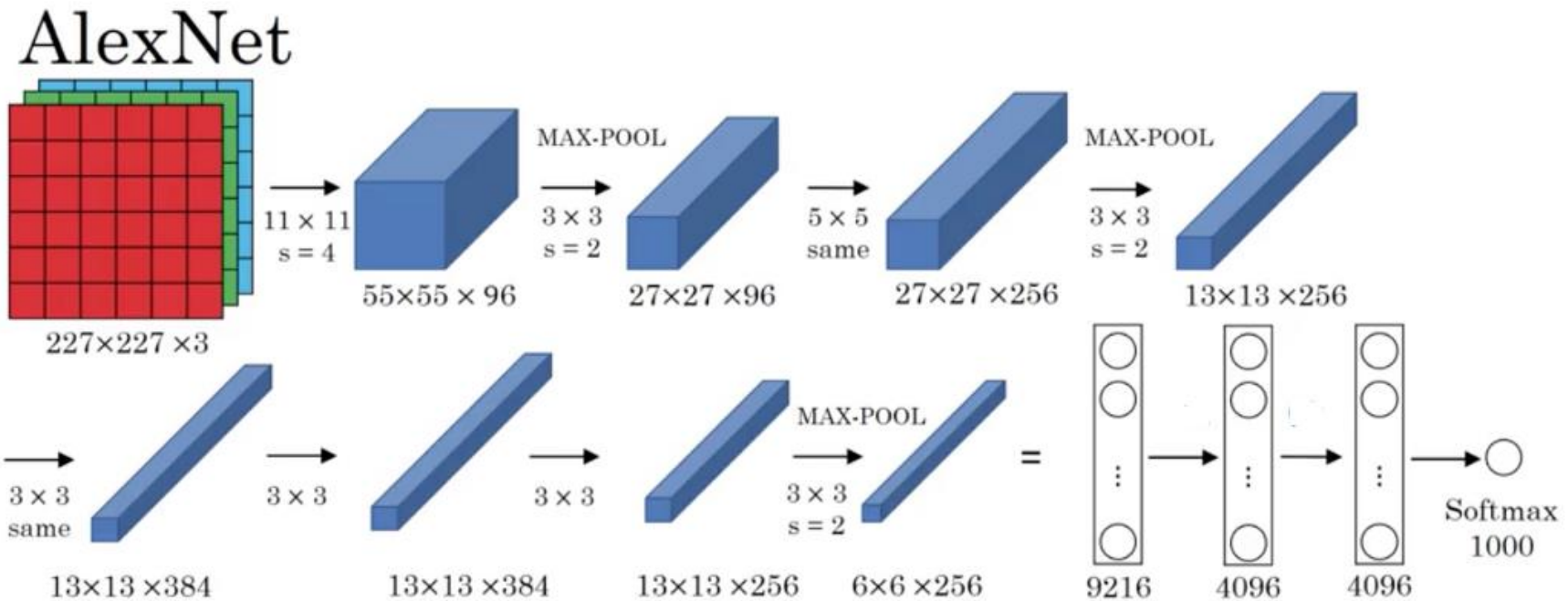
Activation size decreases fastly, but the number of parameters increase (still a lot smaller than the full connected weights).

#Trainable Parameters = kernel size x number of kernels + Bias

	Activation Shape	Activation Size	# Parameters
Input Layer:	(32, 32, 3)	3072	0
CONV1 (f=5, s=1)	(28, 28, 8)	6272	608
POOL1	(14, 14, 8)	1568	0
CONV2 (f=5, s=1)	(10, 10, 16)	1600	3216
POOL2	(5, 5, 16)	400	0
FC3	(120, 1)	120	48120
FC4	(84, 1)	84	10164
Softmax	(10, 1)	10	850

Y. LeCun, L. Bottou, Y. Bengio and P. Haffner: Gradient-Based Learning Applied to Document Recognition, Proceedings of the IEEE, 86(11):2278-2324, November 1998

AlexNet



Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. InAdvances in neural information processing systems 2012 (pp. 1097-1105).

AlexNet

- This paper, titled “ImageNet Classification with Deep Convolutional Networks”, is widely regarded as one of the most influential publications in the field.
- Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton created a “large, deep convolutional
neural network” that was used to win the 2012 ILSVRC (ImageNet Large-Scale Visual Recognition Challenge).
- 2012 marked the first year where a CNN was used to achieve a top 5 test error rate of 15.4% (Top 5 error is the rate at which, given an image, the model does not output the correct label with its top 5 predictions).
- The next best entry achieved an error of 26.2%, which was an astounding improvement that pretty much shocked the computer vision community. Safe to say, CNNs became household names in the competition from then on out.

AlexNet – Main points

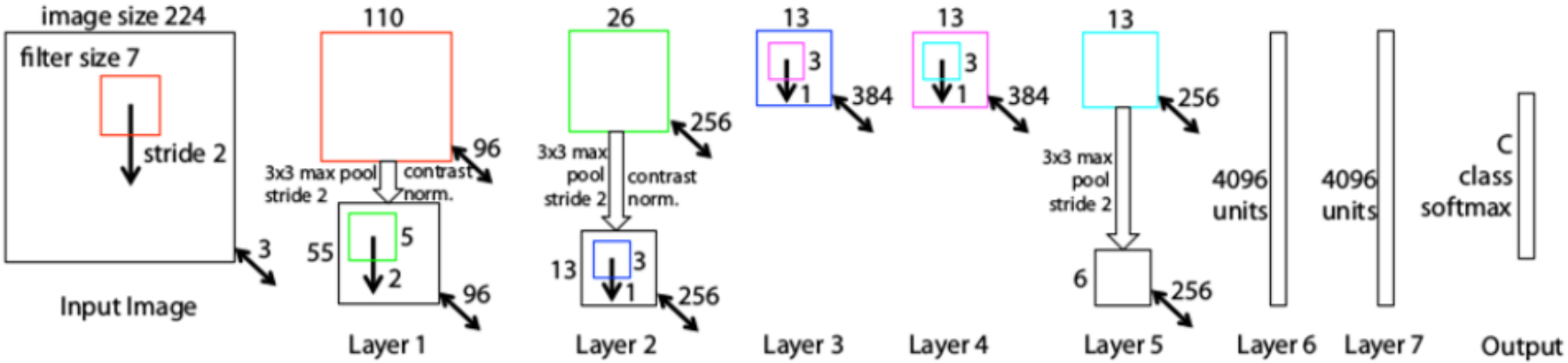
- Trained the network on ImageNet data, which contained over 15 million annotated images from a total of over 22,000 categories.
- Used ReLU for the nonlinearity functions (Found to decrease training time as ReLUs are several times faster than the conventional tanh function).
- Used data augmentation techniques
- Implemented dropout layers in order to combat the problem of overfitting to the training data.
- Trained the model using batch stochastic gradient descent, with specific values for momentum and weight decay.
- Trained on two GTX 580 GPUs for five to six days.

ZF Net

- With AlexNet stealing the show in 2012, there was a large increase in the number of CNN models submitted to ILSVRC 2013.
- The winner of the competition that year was a network built by Matthew Zeiler and Rob Fergus from NYU. Named ZF Net.
- This model achieved an 11.2% error rate.
- This architecture was more of a fine tuning to the previous AlexNet structure, but still developed some very keys ideas about improving performance.
- Another reason this was such a great paper is that the authors spent a good amount of time explaining a lot of the intuition behind ConvNets and showing how to visualize the filters and weights correctly.

Zeiler MD, Fergus R. Visualizing and understanding convolutional networks. InEuropean conference on computer vision 2014 Sep 6 (pp. 818-833). Springer, Cham.

ZF Net



ZF Net Architecture

Zeiler MD, Fergus R. Visualizing and understanding convolutional networks. In European conference on computer vision 2014 Sep 6 (pp. 818-833). Springer, Cham.

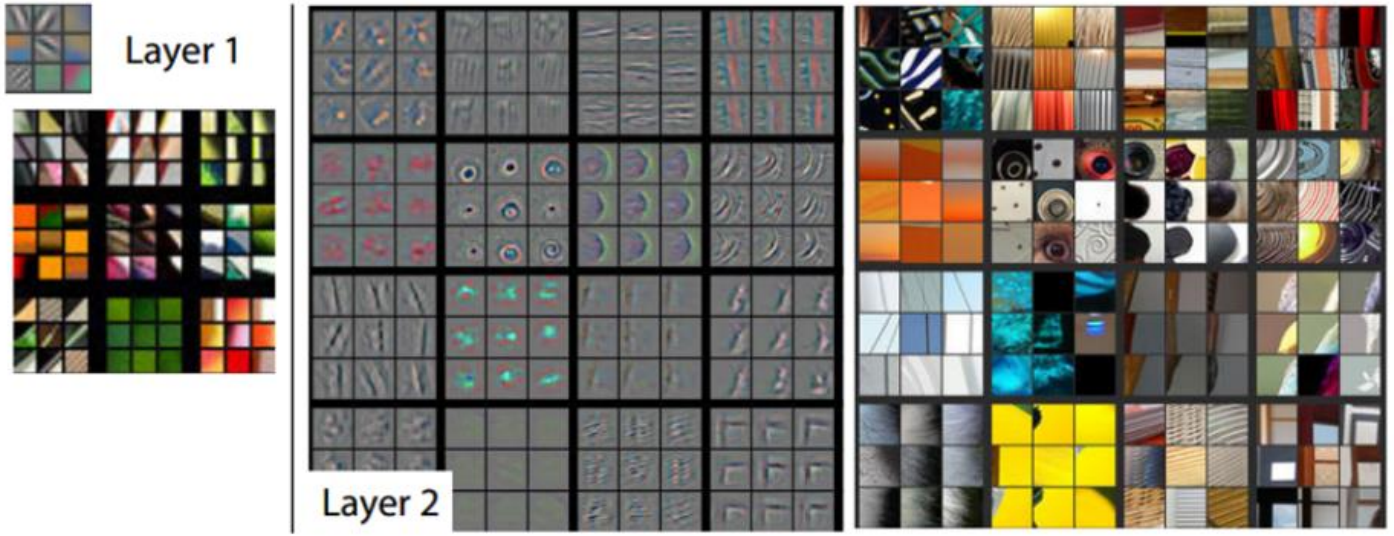
ZF Net – Main Points

- Very similar architecture to AlexNet, except for a few minor modifications.
- AlexNet trained on 15 million images, while ZF Net trained on only 1.3 million images.
- Instead of using 11x11 sized filters in the first layer (which is what AlexNet implemented), ZF Net used filters of size 7x7 and a decreased stride value. The reasoning behind this modification is that a smaller filter size in the first conv layer helps retain a lot of original pixel information in the input volume. A filtering of size 11x11 proved to be skipping a lot of relevant information, especially as this is the first conv layer.
- As the network grows, we also see a rise in the number of filters used.
- Used ReLUs for their activation functions, cross-entropy loss for the error function, and trained using batch stochastic gradient descent.
  Trained on a GTX 580 GPU for twelve days.

Zeiler MD, Fergus R. Visualizing and understanding convolutional networks. In European conference on computer vision 2014 Sep 6 (pp. 818-833). Springer, Cham.

ZF Net – visualization of layers

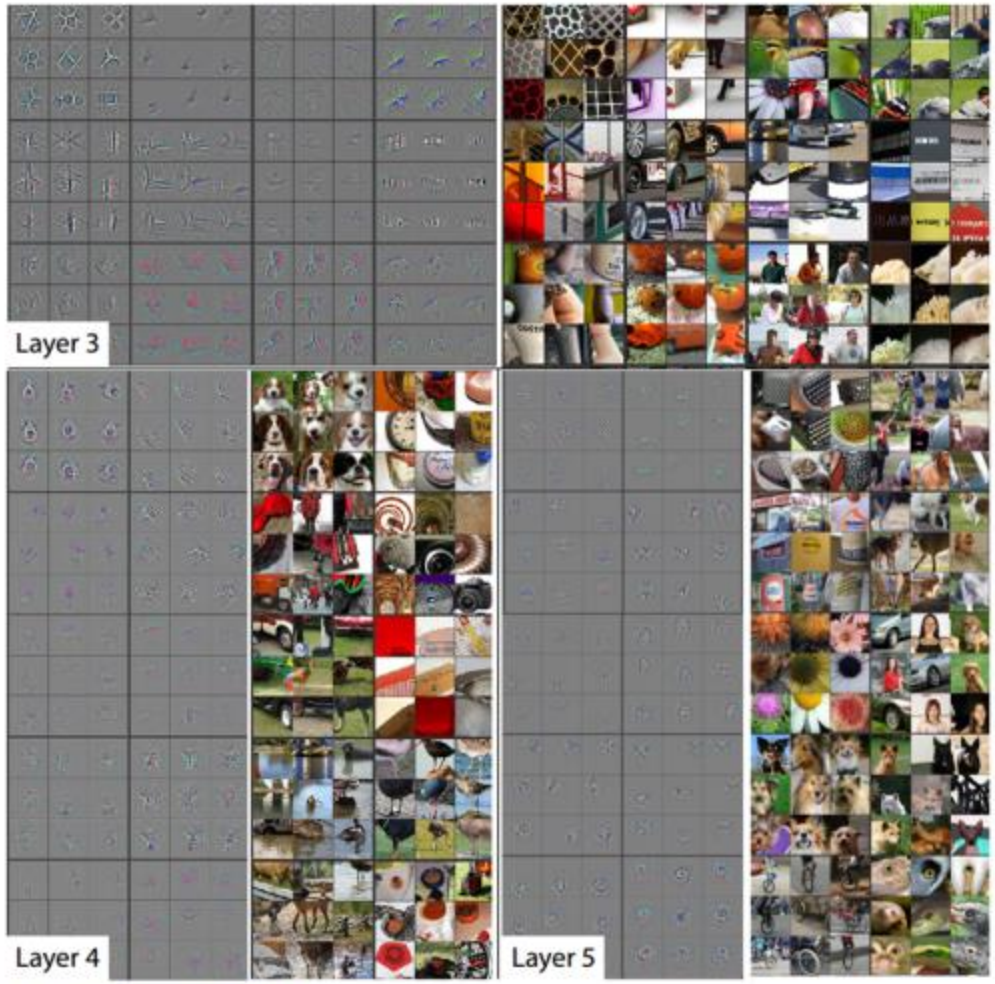
- Developed a visualization technique named Deconvolutional Network, which helps to examine different feature activations and their relation to the input space. Called “deconvnet” because it maps features to pixels (the opposite of what a convolutional layer does)



Visualizations of Layer 1 and 2. Each layer illustrates 2 pictures, one which shows the filters themselves and one that shows what part of the image are most strongly activated by the given filter. For example, in the space labled Layer 2, we have representations of the 16 different filters (on the left)

Zeiler MD, Fergus R. Visualizing and understanding convolutional networks. In European conference on computer vision 2014 Sep 6 (pp. 818-833). Springer, Cham.

ZF Net – visualization of layers

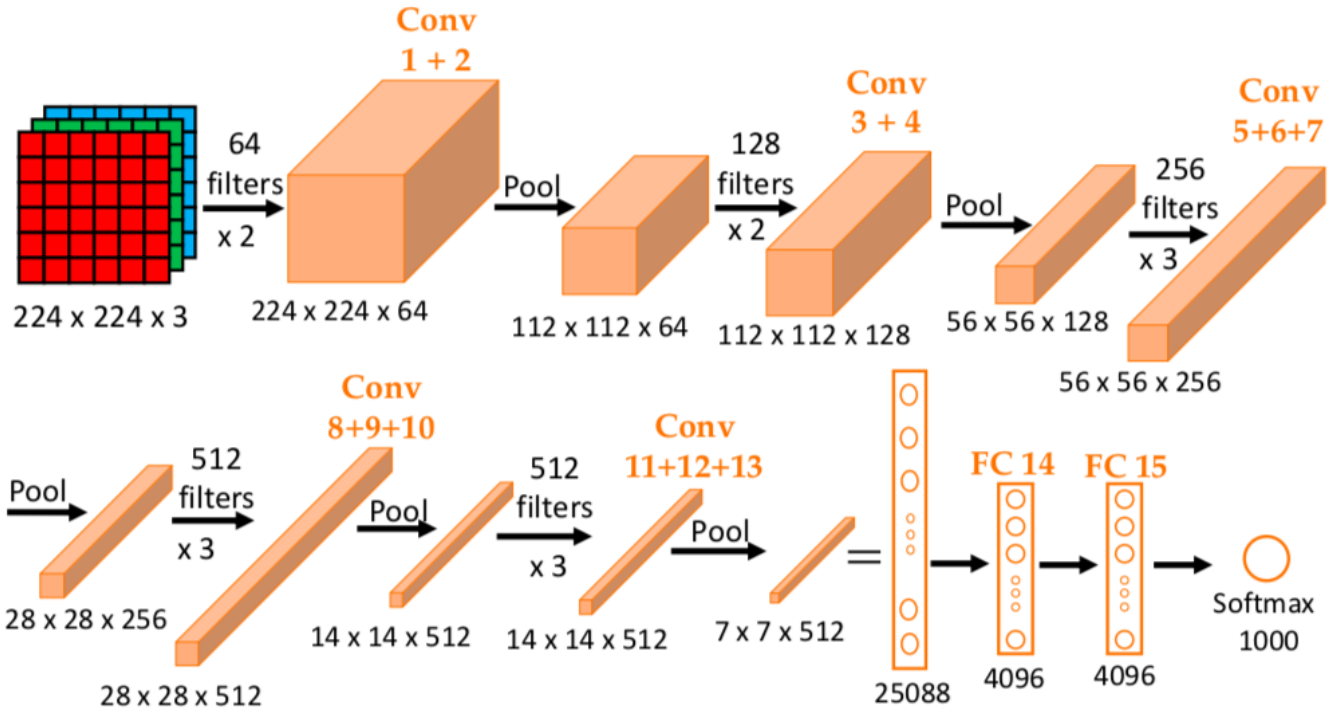


Visualizations of Layers 3, 4, and 5

Zeiler MD, Fergus R. Visualizing and understanding convolutional networks. In European conference on computer vision 2014 Sep 6 (pp. 818-833). Springer, Cham.

VGG-16

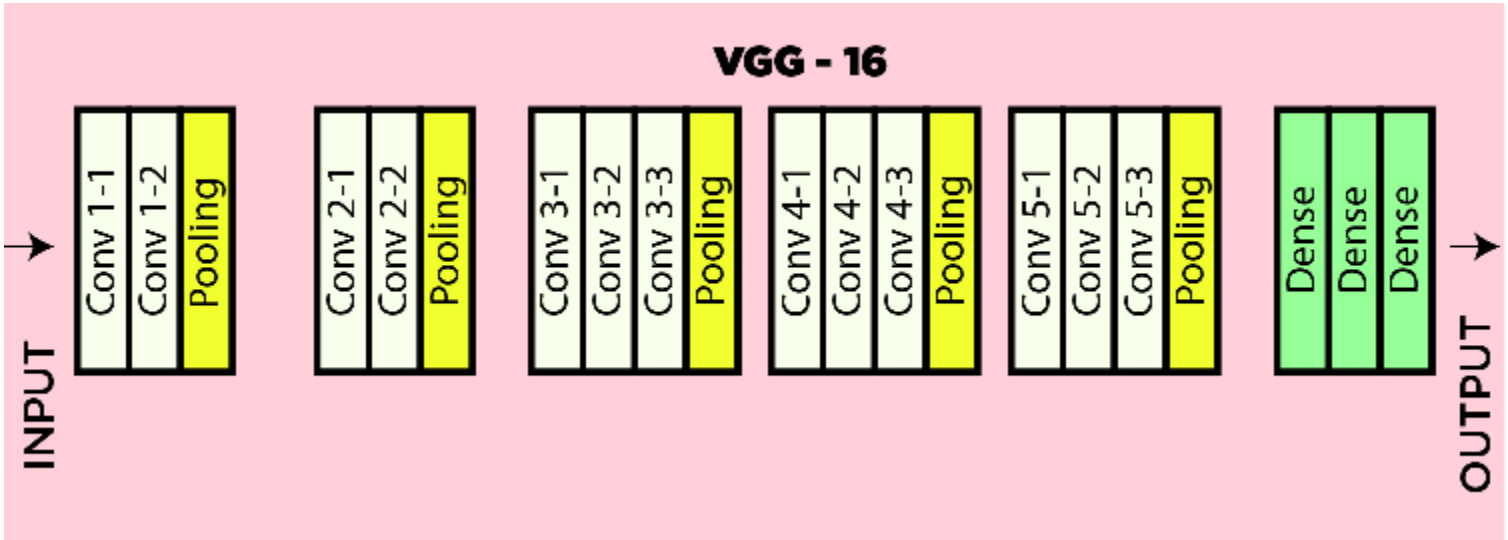
CONV = 3x3 filter, s=1, same (in this case padding of 1). **MAX-POOL = 2x2, S=2**



Liu S, Deng W. Very deep convolutional neural network based image classification using small training sample size. In2015 3rd IAPR Asian conference on pattern recognition (ACPR) 2015 Nov 3 (pp. 730-734). IEEE.

VGG-16

CONV = 3x3 filter, s=1, same. **MAX-POOL = 2x2, S=2**



Liu S, Deng W. Very deep convolutional neural network based image classification using small training sample size. In2015 3rd IAPR Asian conference on pattern recognition (ACPR) 2015 Nov 3 (pp. 730-734). IEEE.

VGG Net

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB imag)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

The 6 different architecures of VGG Net. Configuration D produced the best results

Liu S, Deng W. Very deep convolutional neural network based image classification using small training sample size. In2015 3rd IAPR Asian conference on pattern recognition (ACPR) 2015 Nov 3 (pp. 730-734). IEEE.

VGG-16

- The use of only 3x3 sized filters is quite different from AlexNet's 11x11 filters in the first layer and ZF Net's 7x7 filters. The authors' reasoning is that the combination of two 3x3 conv layers has an effective receptive field of 5x5.
- This in turn simulates a larger filter while keeping the benefits of smaller filter sizes. One of the benefits is a decrease in the number of parameters.
- Also, with two conv layers, we're able to use two ReLU layers instead of one.
- 3 conv layers back to back have an effective receptive field of 7x7.
- As the spatial size of the input volumes at each layer decrease (result of the conv and pool layers), the depth of the volumes increase due to the increased number of filters as you go down the network.
- Interesting to notice that the number of filters doubles after each maxpool layer. This reinforces the idea of shrinking spatial dimensions, but growing depth.
- Worked well on both image classification and localization tasks.
- Used ReLU layers after each conv layer and trained with batch gradient descent.
- Trained on 4 Nvidia Titan Black GPUs for two to three weeks.
- Why It's Important VGG Net is one of the most influential papers in my mind because it reinforced the notion that convolutional neural networks have to have a deep network of layers in order for this hierarchical representation of visual data to work. Keep it deep. Keep it simple