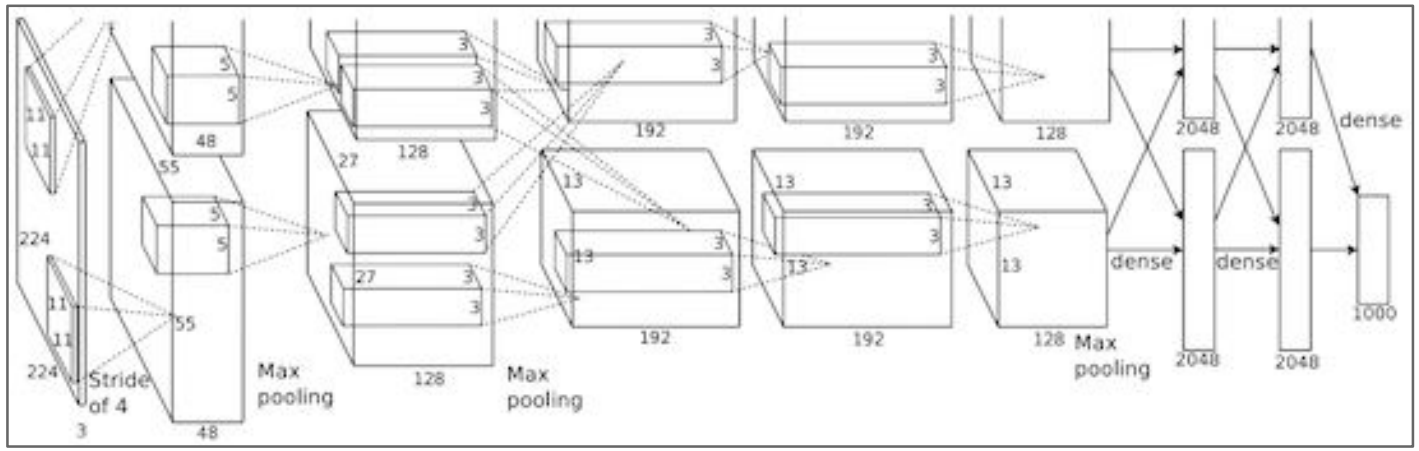
Convolutional Neural Networks

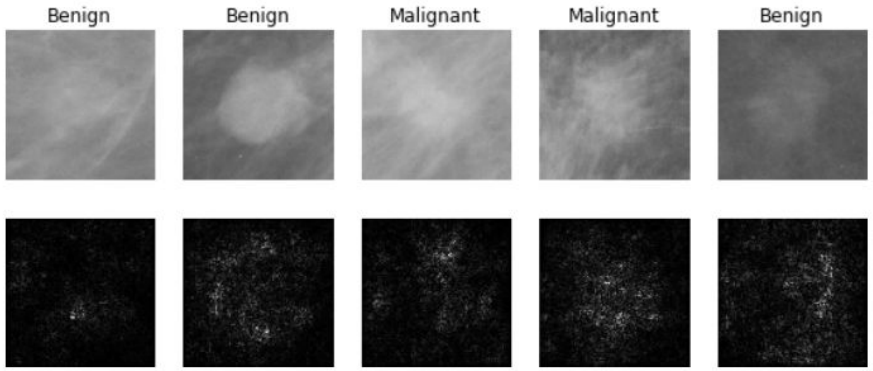
Convolutional Neural Networks Everywhere

- Special kind of architecture that were popularized in late 2010
  - Hierarchical structure
  - local receptive fields
  - convolutions
- Inspired from studying the visual cortex of mammals



[Krizhevsky et al 2012]

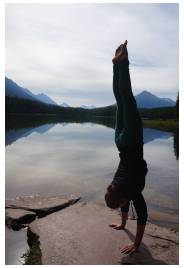
CNNs everywhere



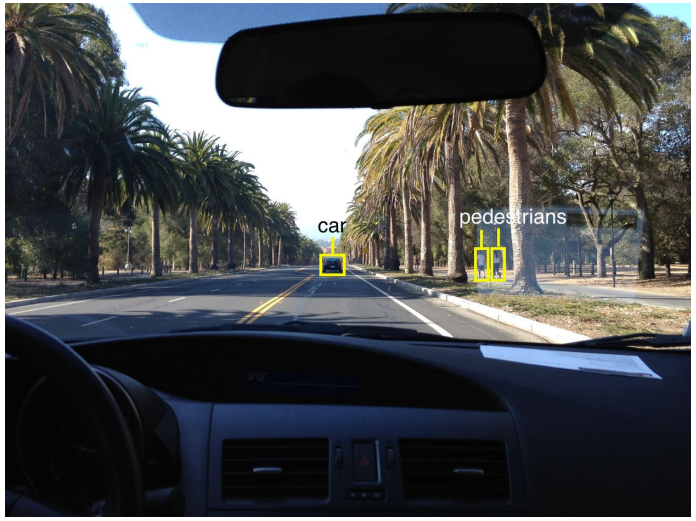
[Levy et al. 2016]

A man riding a wave on top of a surfboard

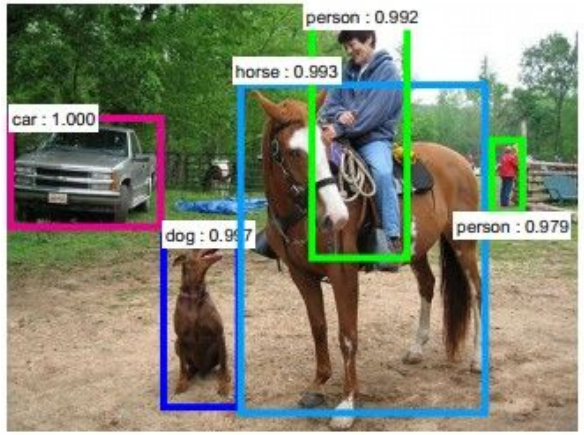
A cat sitting on a suitcase on the floor

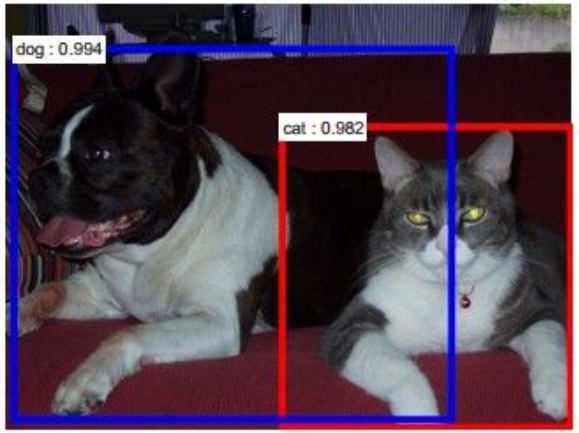


A woman standing on a beach holding a surfboard

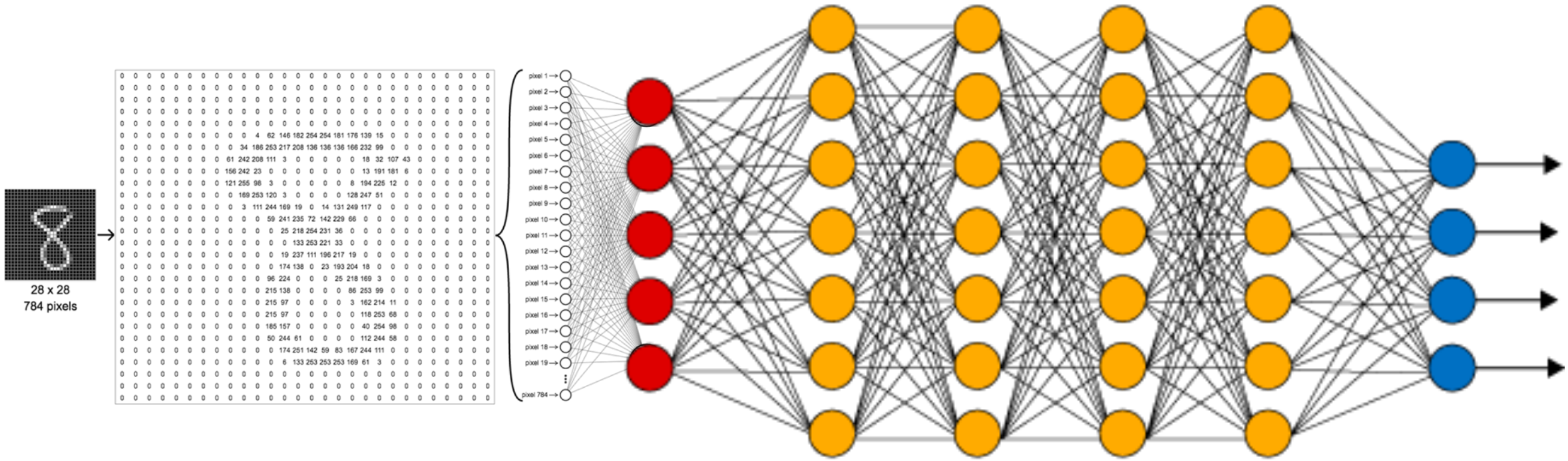


self-driving cars





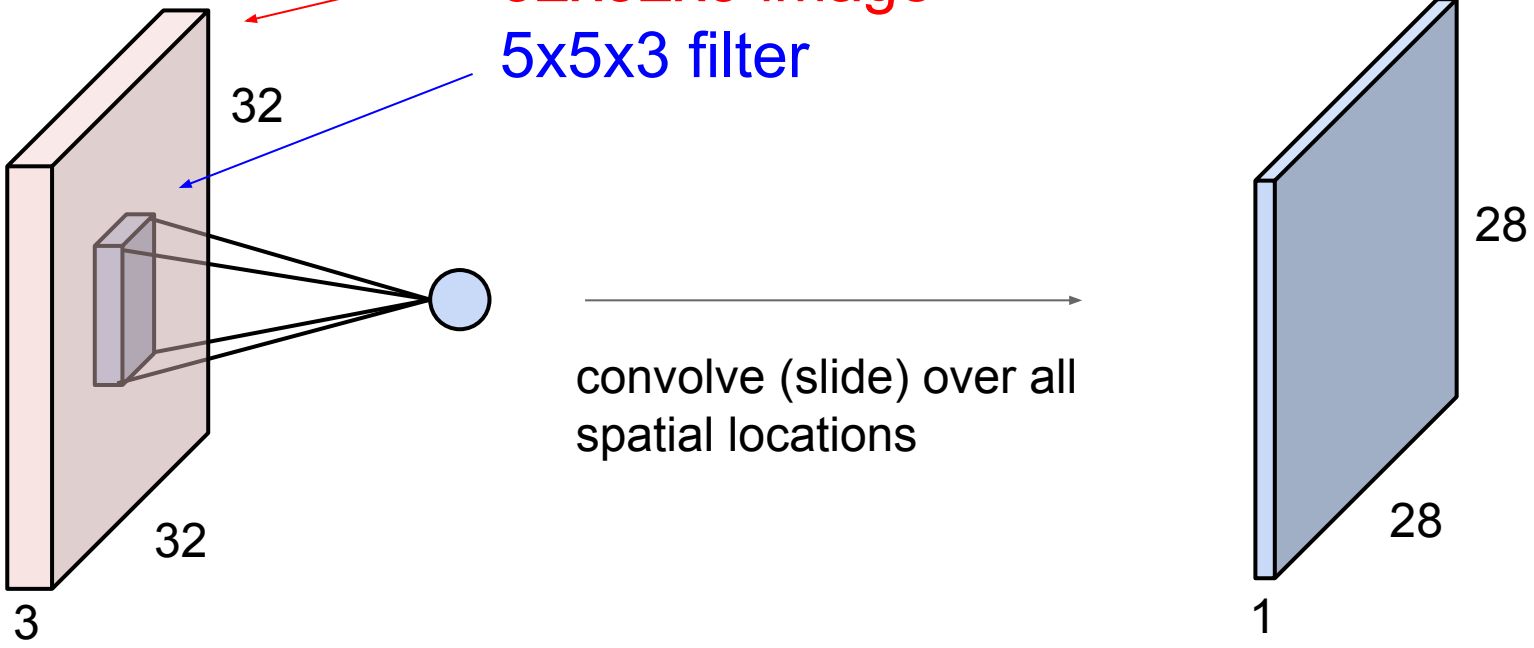
Fully Connected NN



*The entire image input is **linearized** into an array of length 784*

No locality preservation

Convolution Layer

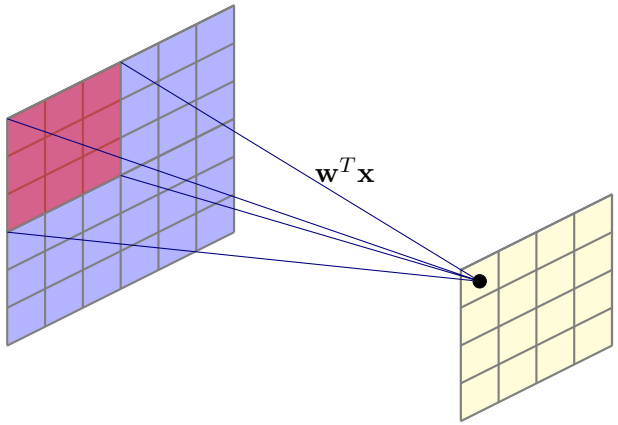


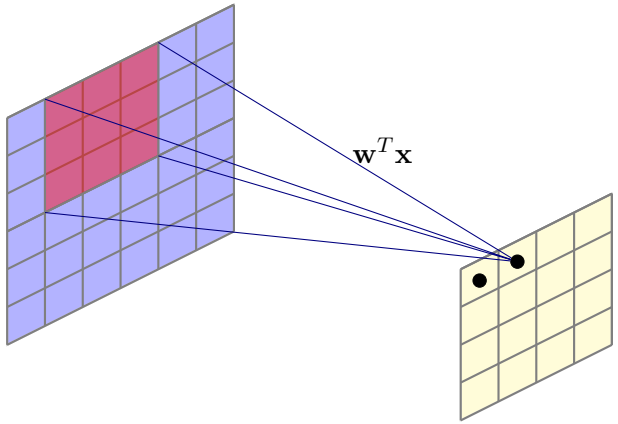
Convolve the filter/kernel with the image by sliding the filter w

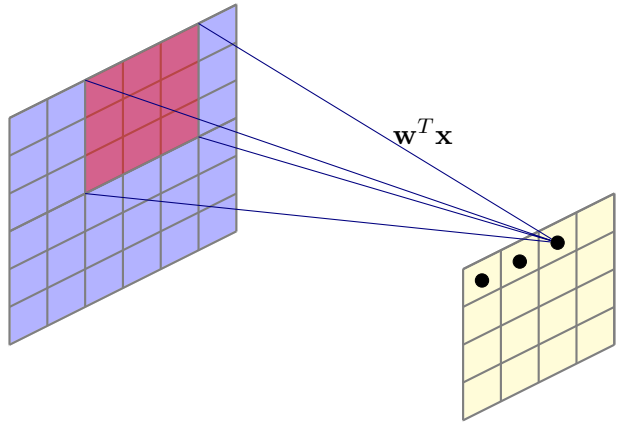
For each configuration, compute the dot product $w^T x + b$

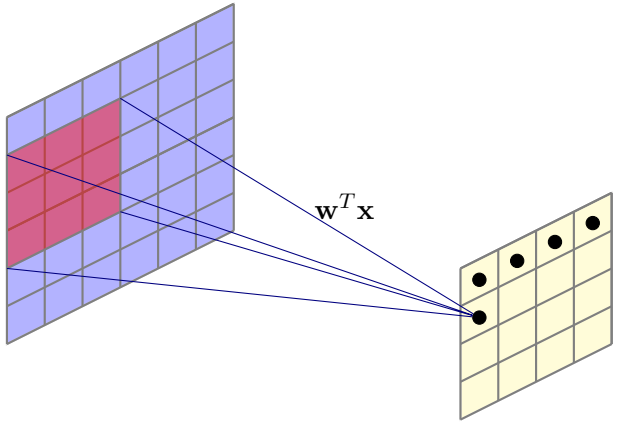
Why the dimensions are 28x28x1 ? Hint: stride 1

Convolution Layer

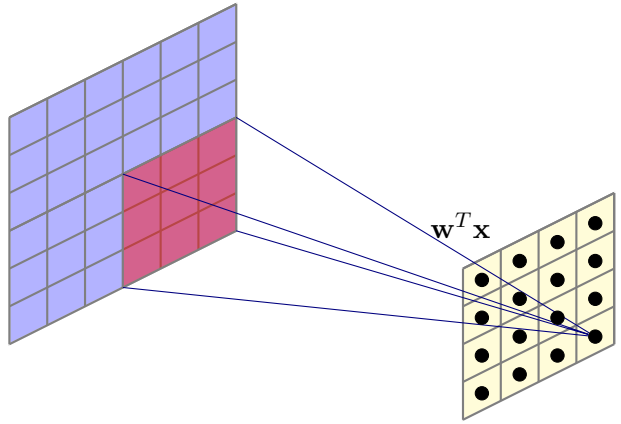




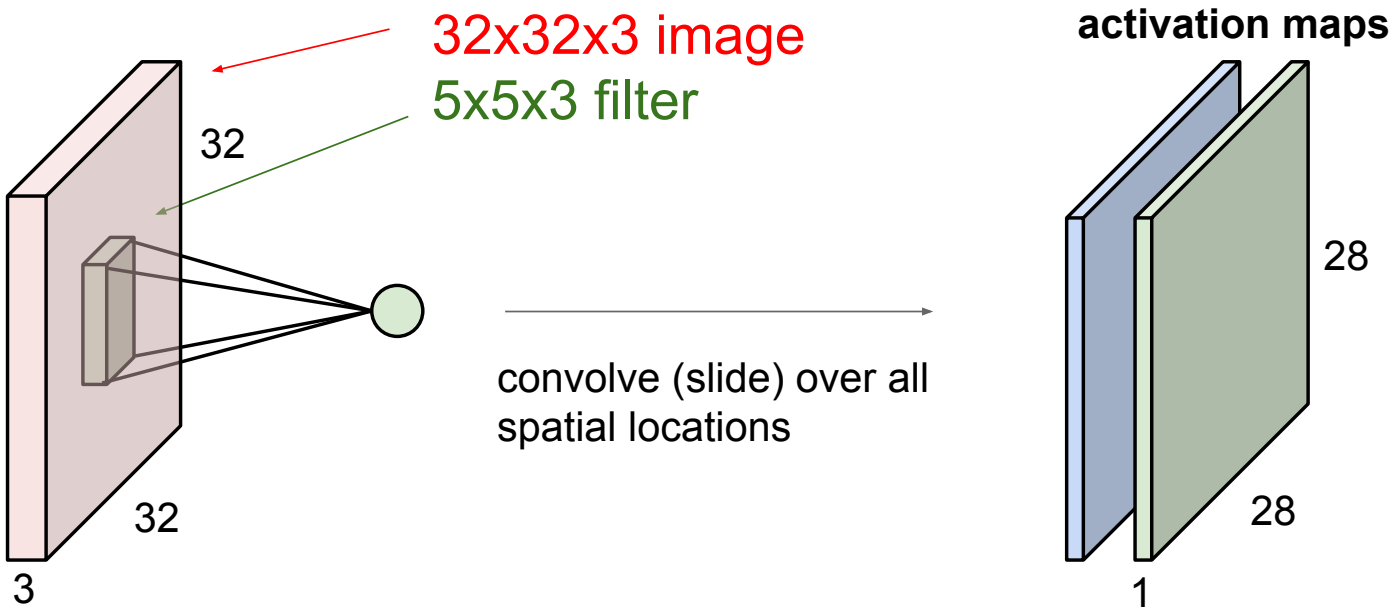




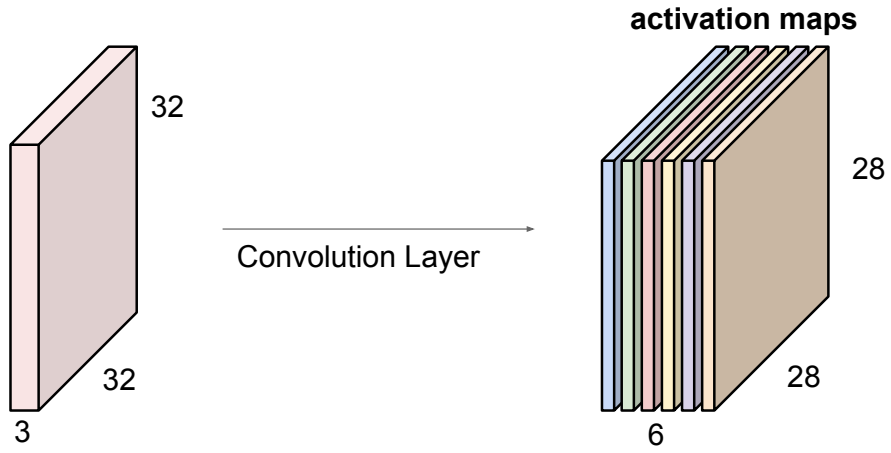
.....



Multiple filters



Multiple filters can be used over the same input image



6 separate activation maps

Why the dimensions are 28x28x1 ?

Filters in ConvNets

Original

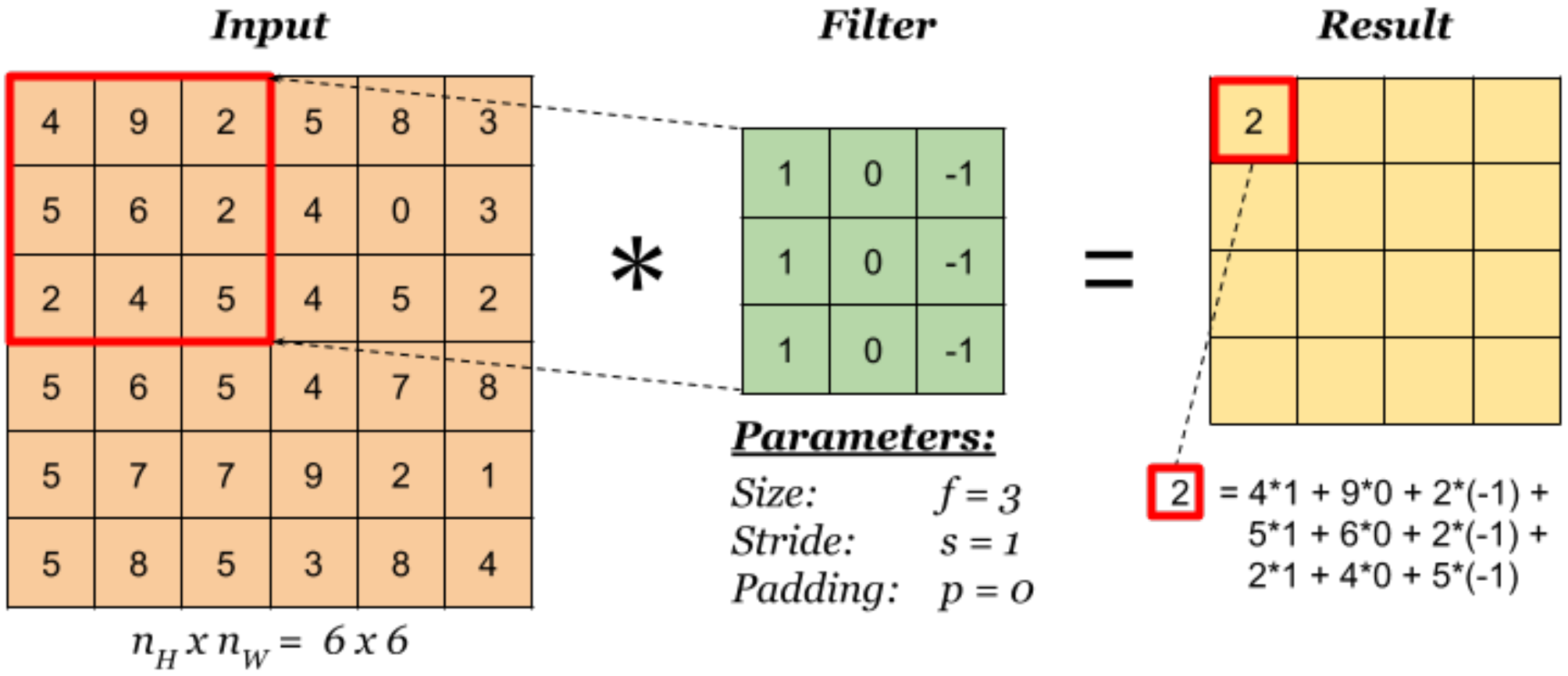
Sharpen

Edge Detect

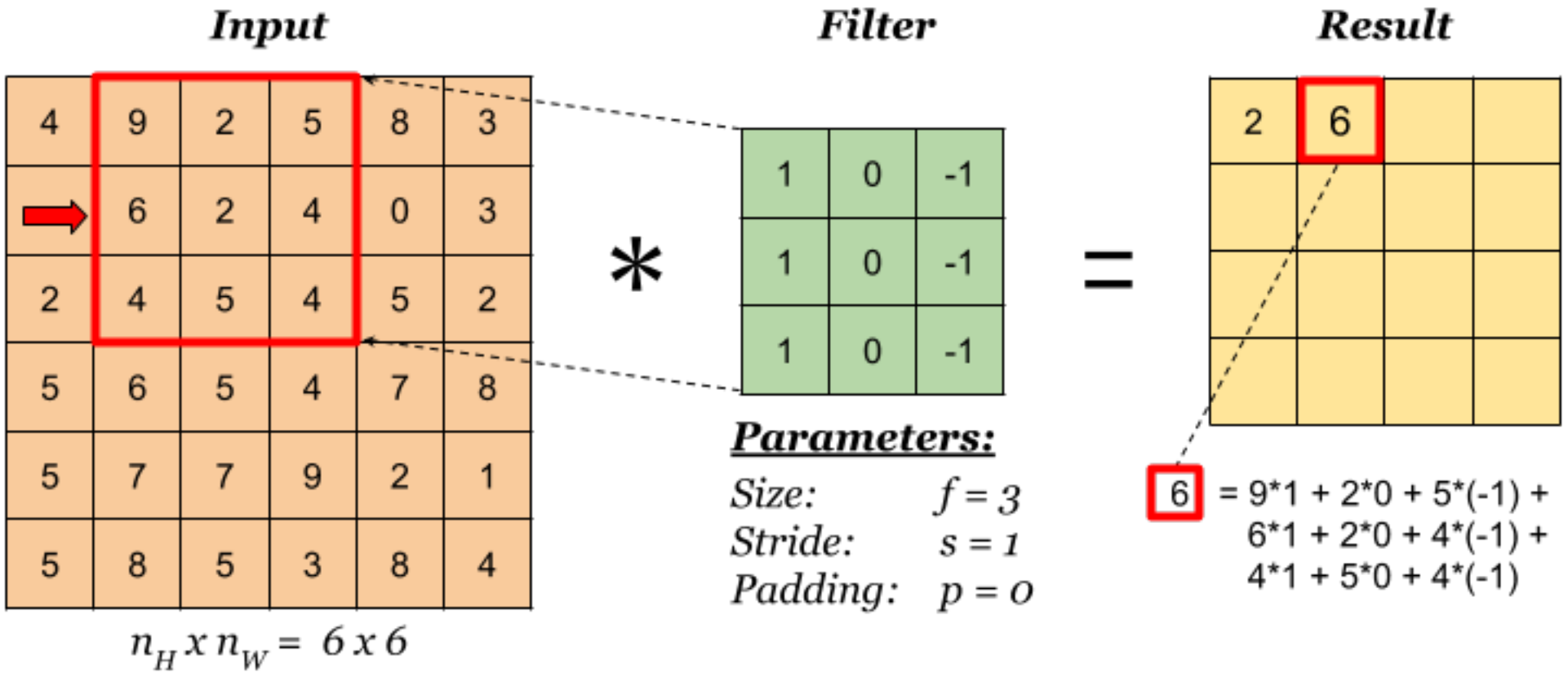
"Strong" Edge Detect

- *Apply a set of weights – a filter – to extract local features*
- *Use multiple filters to extract different features*
- *Spatially share parameters of each filter*

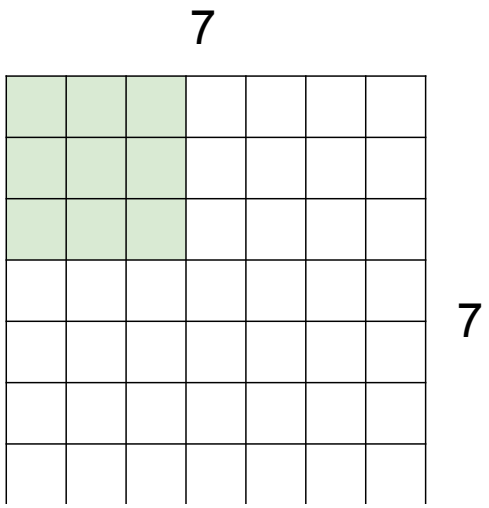
Convolution Layer



Convolution Layer

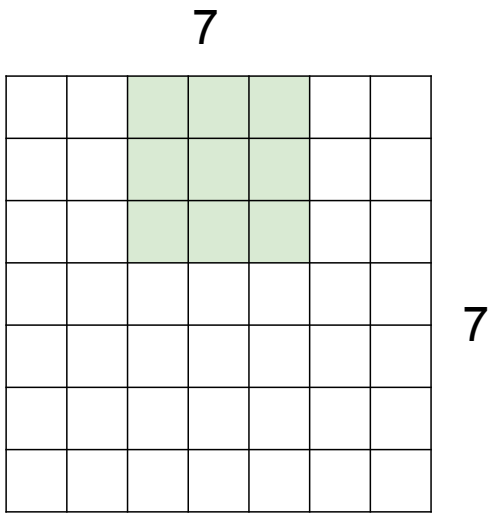


Dimensions



7x7 input with a ***filter 3x3*** with ***stride 1.***

What is the output size ?

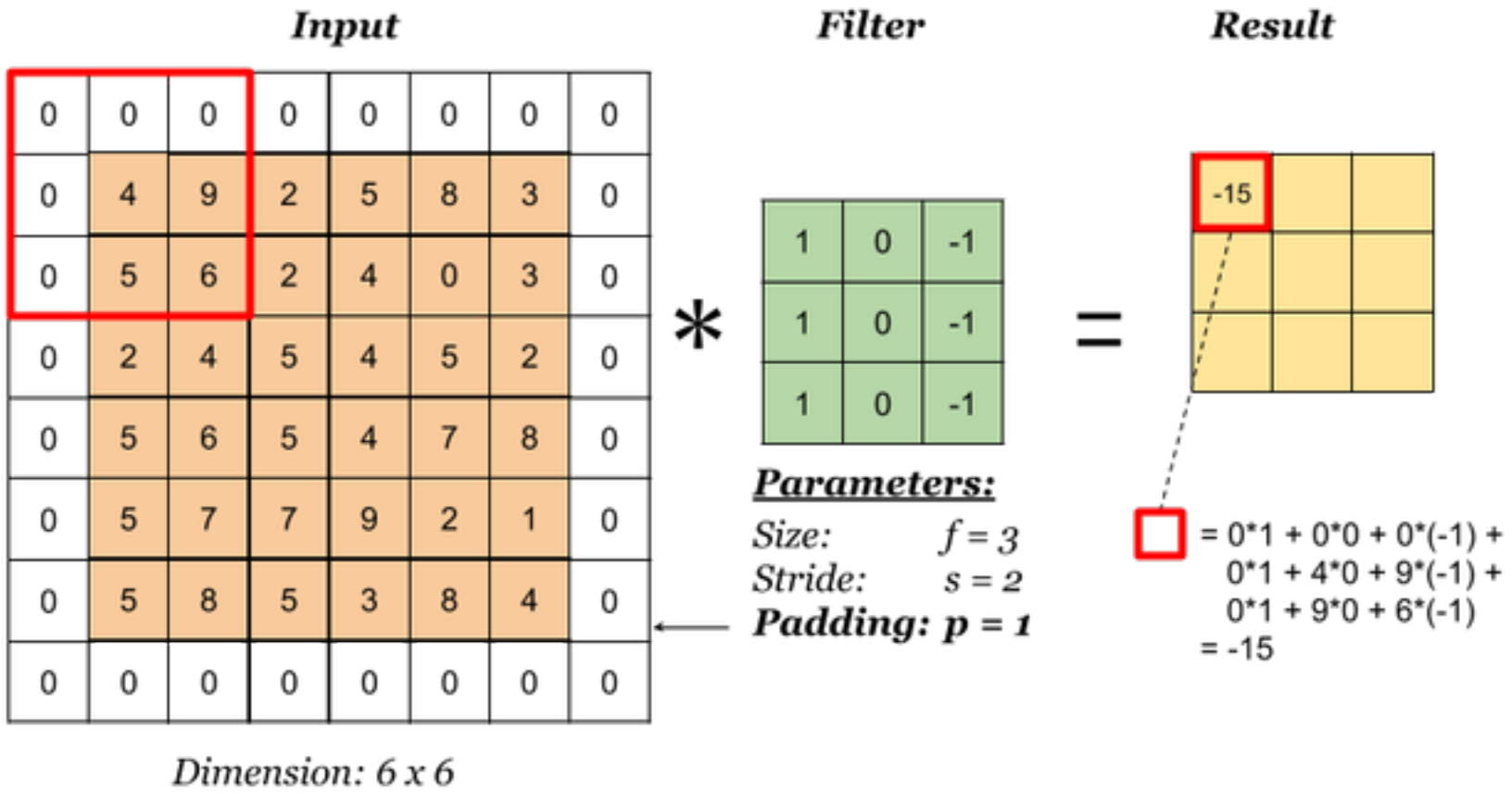


7x7 input with a ***filter 3x3*** with ***stride 2.***

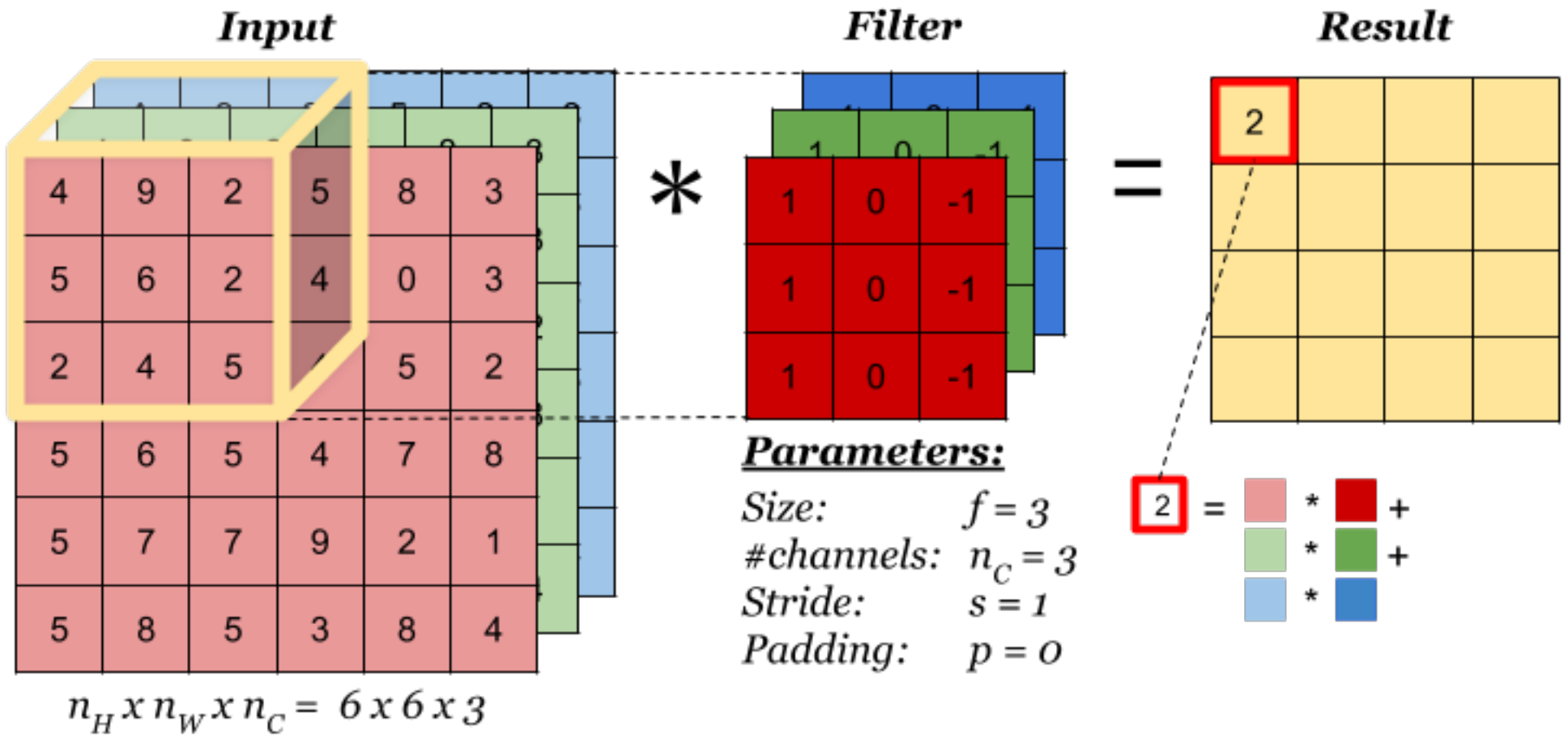
What is the output size ?

(N-F) / Stride + 1

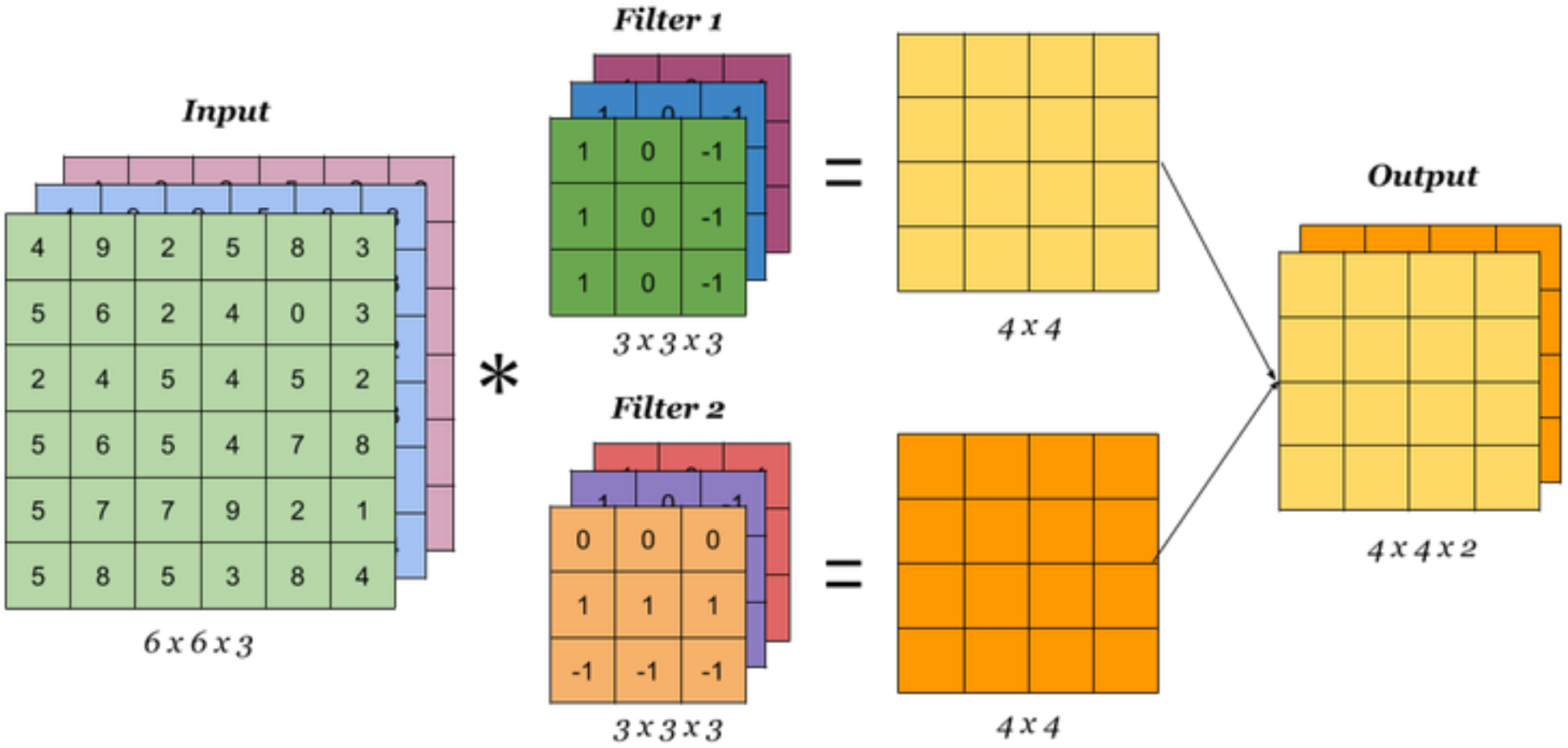
Zero Padding



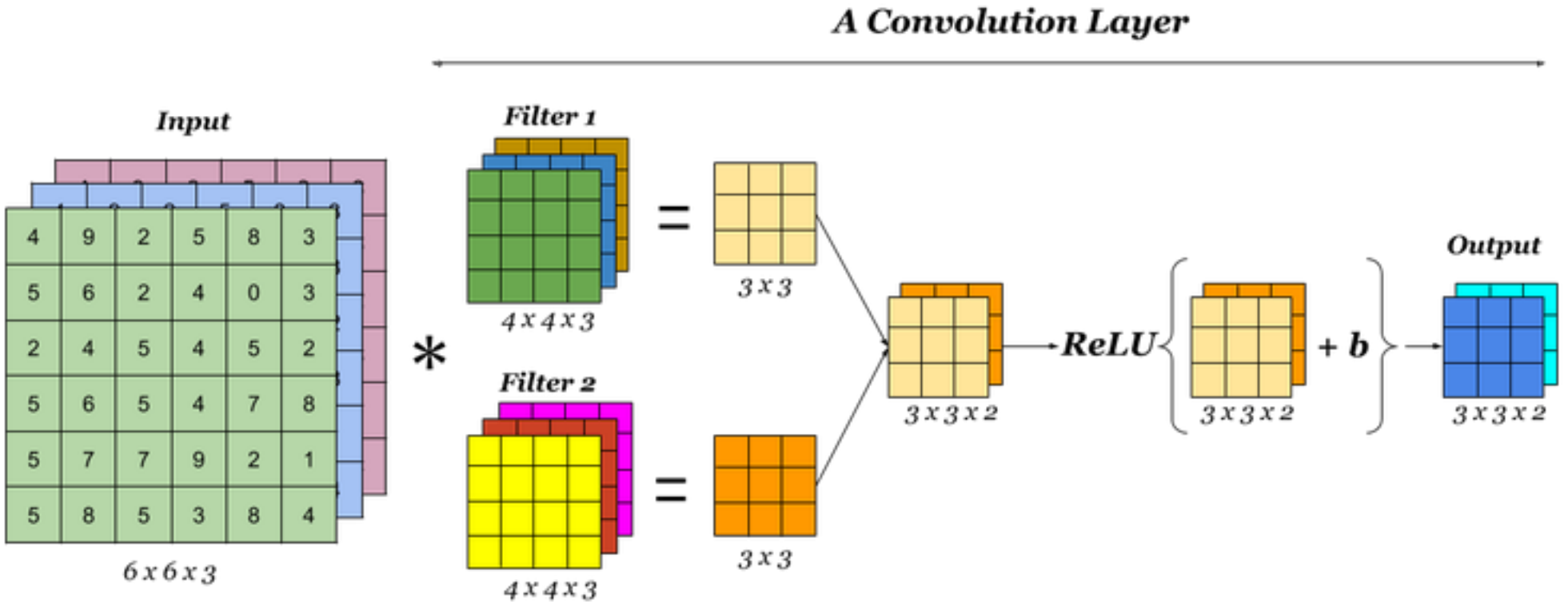
Convolutions on 3D or volume



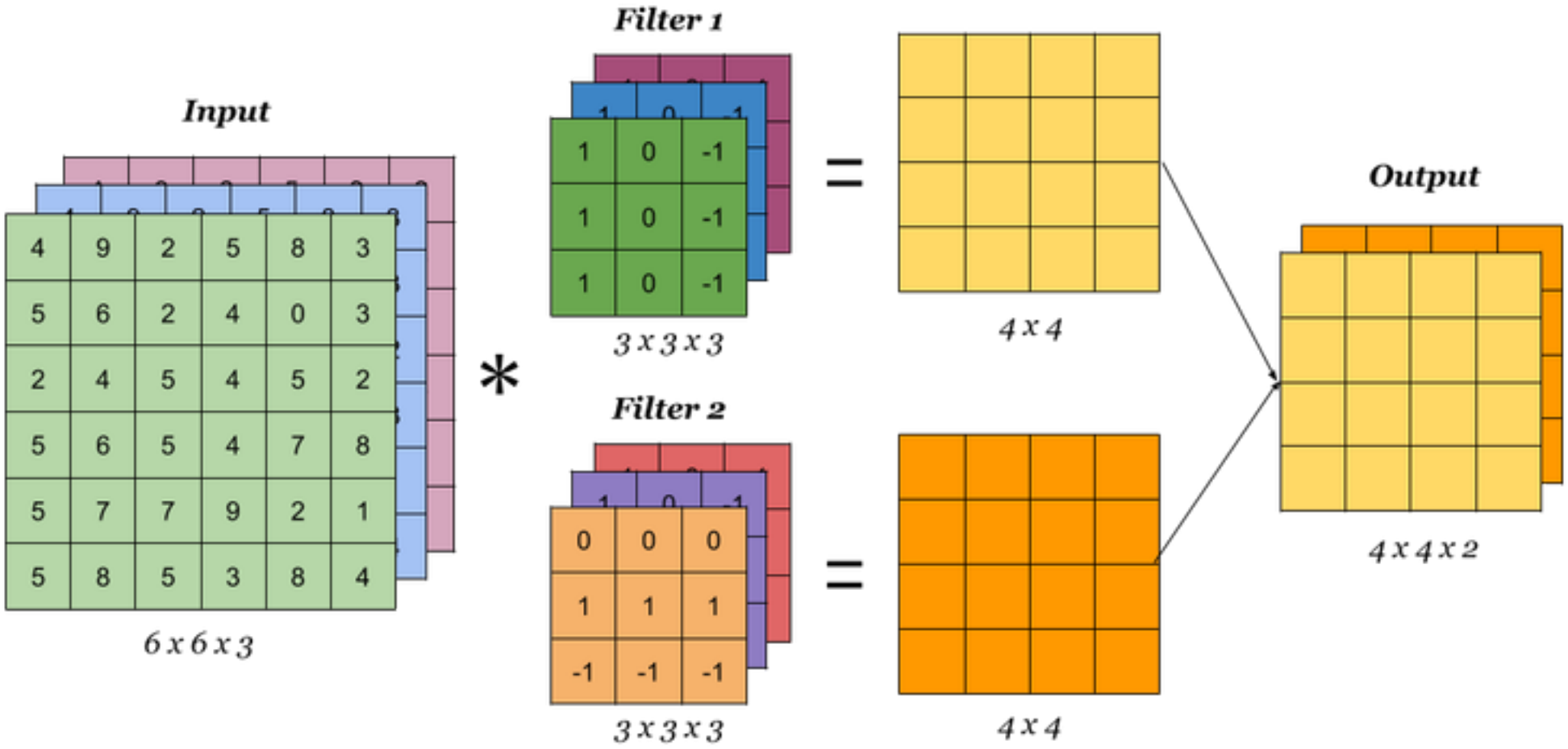
Convolutions with multiple filters



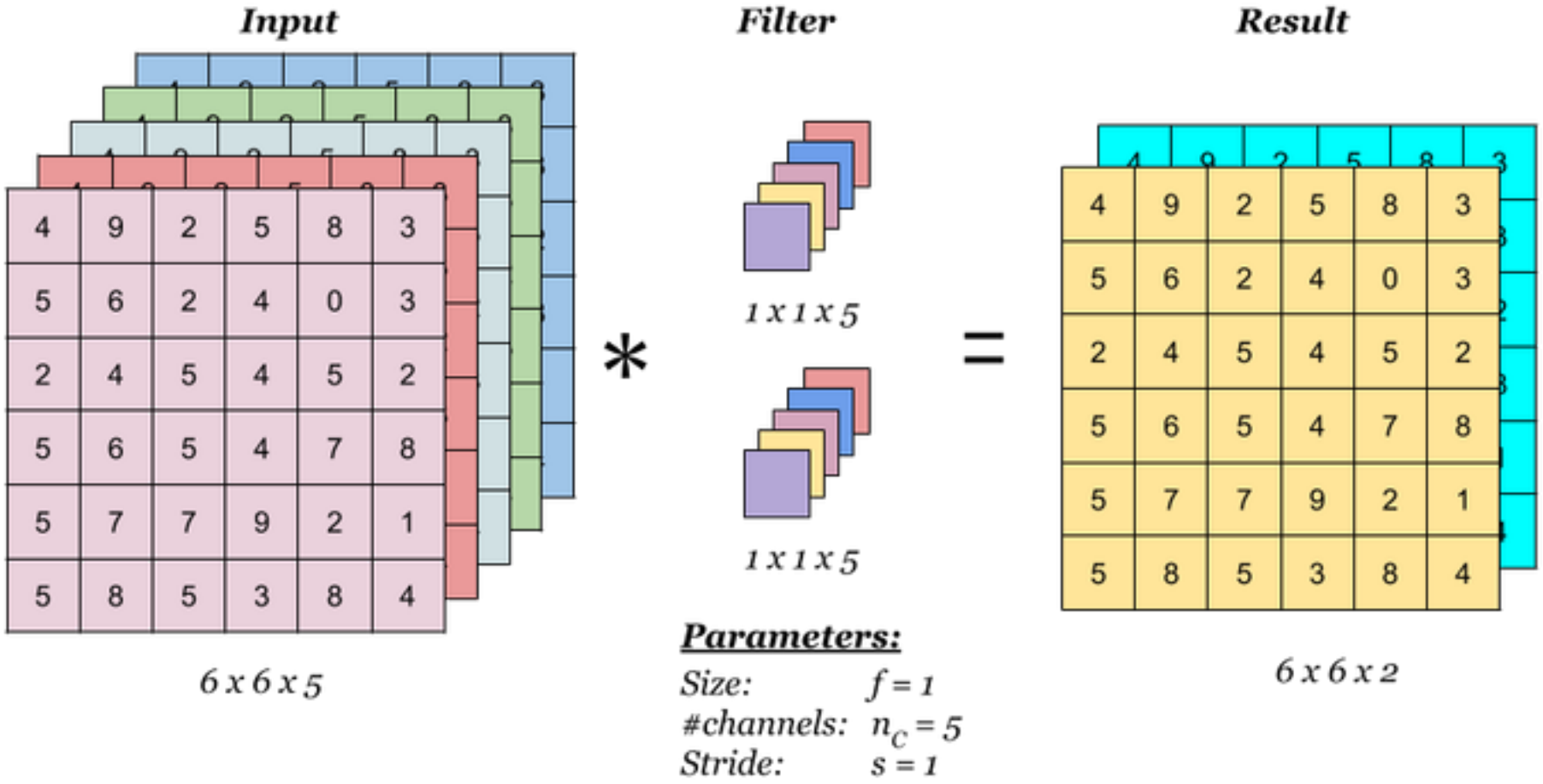
Single Convolutional Layer



Convolutions with multiple filters



Convolutions 1x1 filter



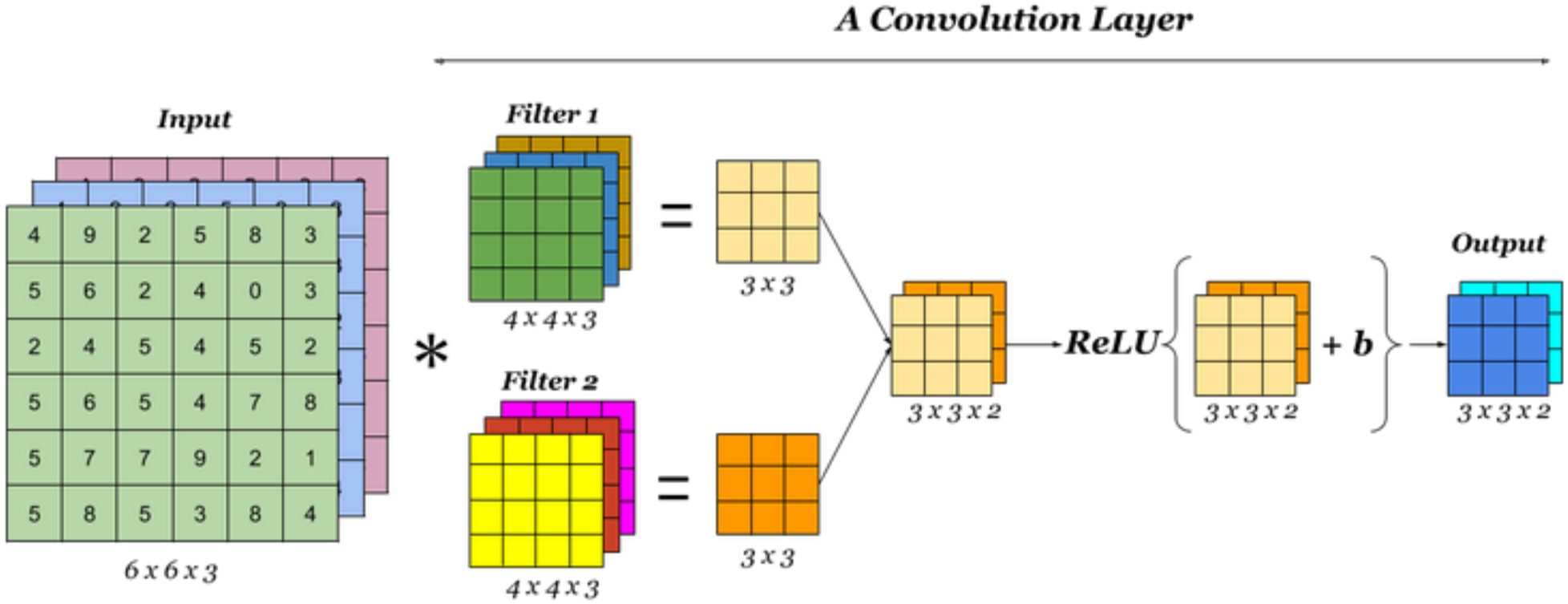
Parameters:

Size: $f = 1$

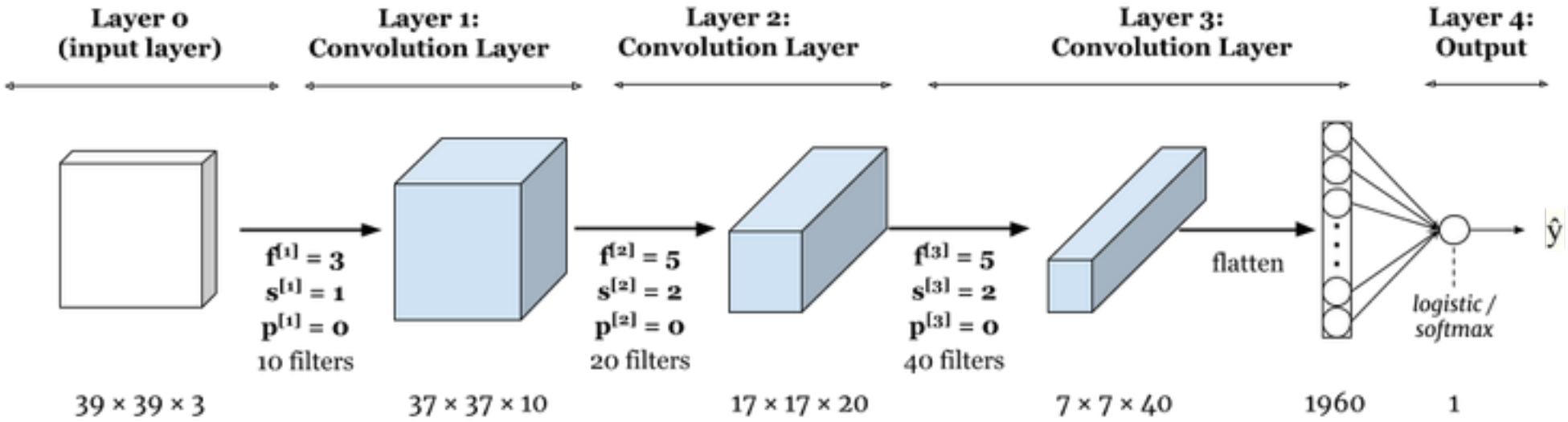
#channels: $n_C = 5$

Stride: $s = 1$

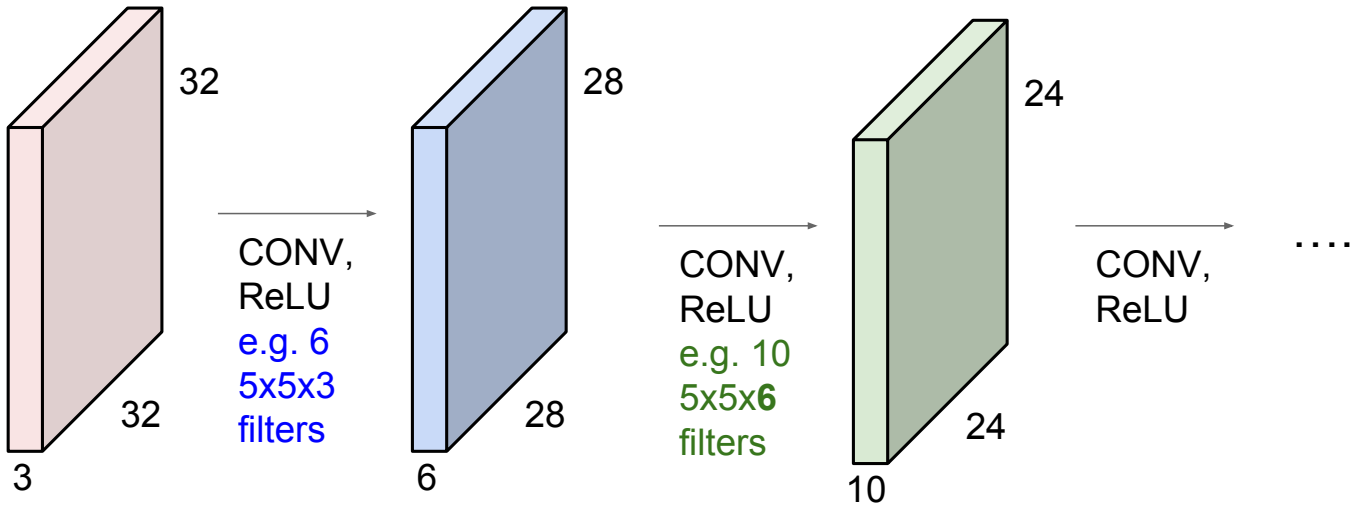
Convolution Layer



Sample Complete Network



ConvNet



A convnet is a sequence of convolution layers interspersed with activation functions (ReLU)

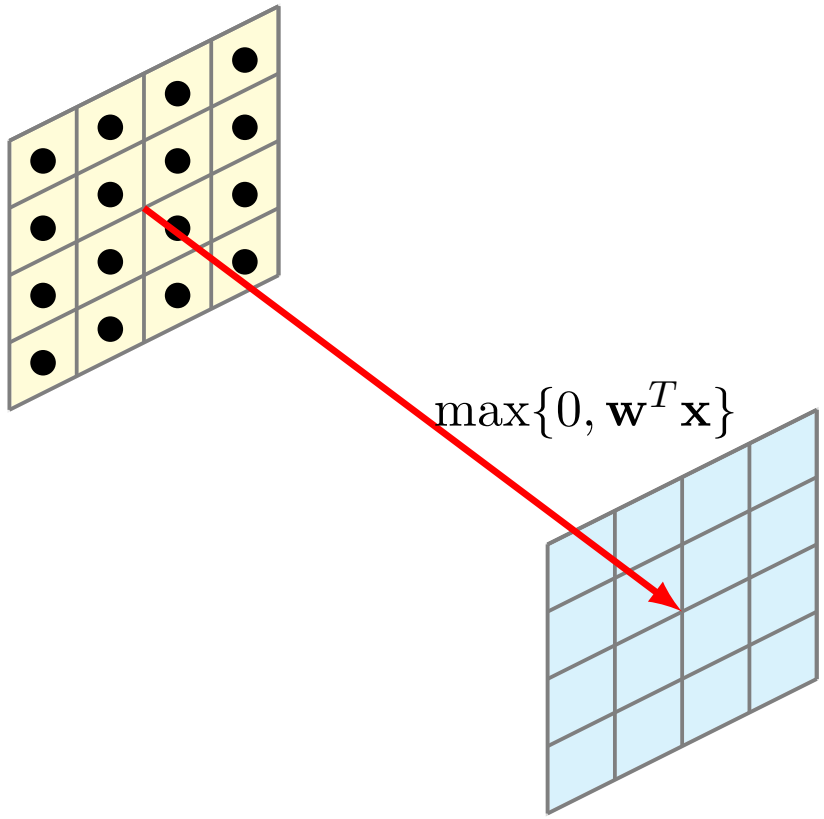
$$f[x,y] * g[x,y] = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f[n_1,n_2] \cdot g[x-n_1, y-n_2]$$

Convolution of two signals, g is the filter and f is the input

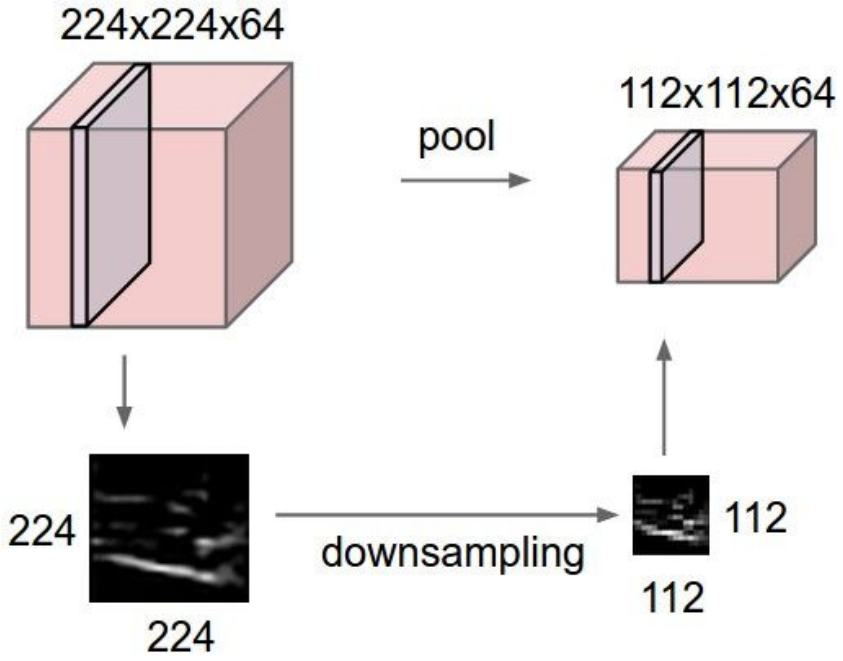
Filter centered around (n_1, n_2)

Non-Linearity

After obtaining feature map, apply an elementwise non-linearity to obtain a transformed feature map (same size)



Pooling Layer



- *Makes representations more manageable*
- *Depth doesn't change*
- *Down sampling*
- *Speeds up calculations*

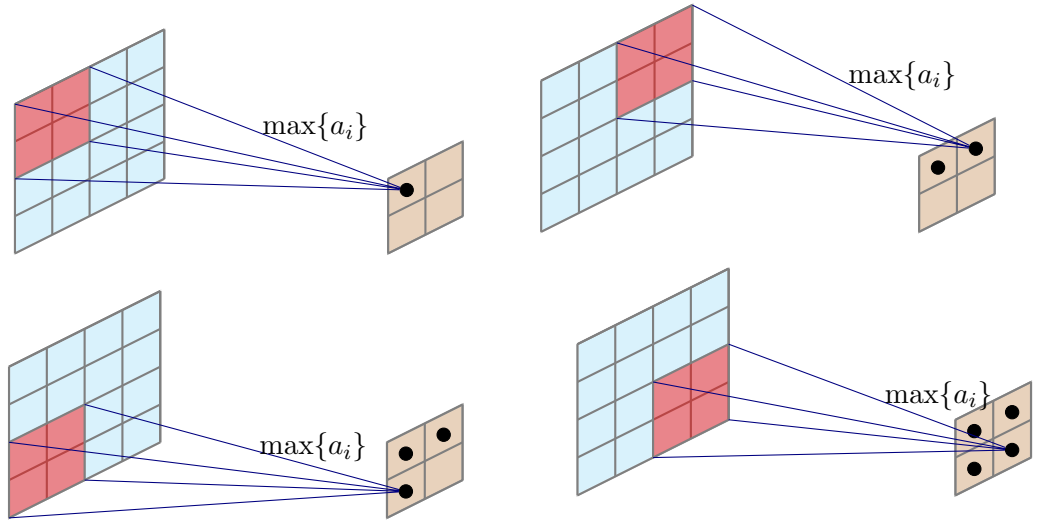
Max Pooling

Single depth slice

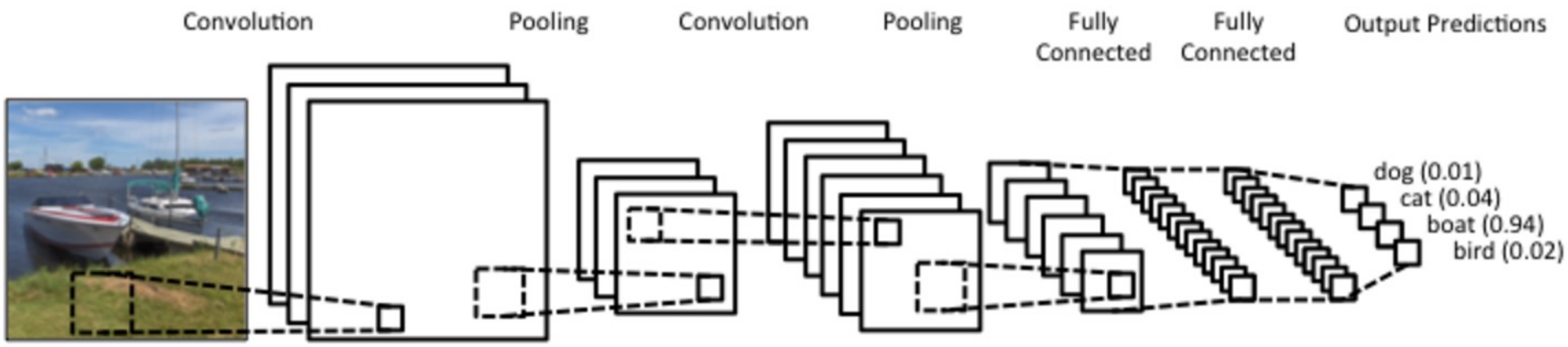
1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

max pool with 2x2 filters and stride 2

6	8
3	4



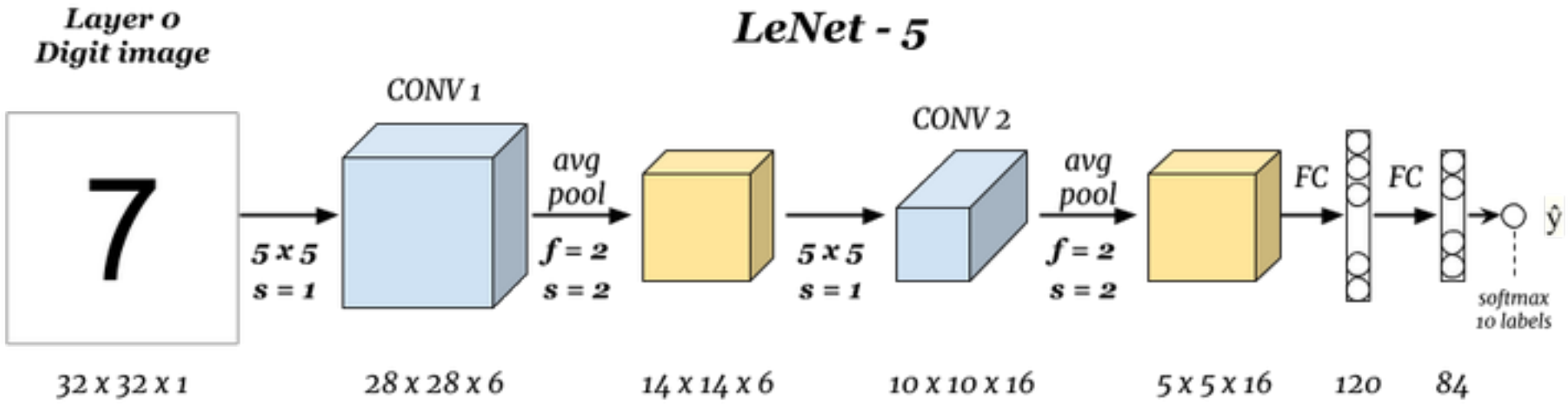
CNN Pipeline



- **Convolution:** Apply filters with learned weights to generate feature maps
- **Non-linearity:** Often ReLU
- **Pooling:** Downsampling operation on each feature map
- **Fully Connected Layers:** at the end for classification

- Train model with image data and learn weights of filters in the convolution layers

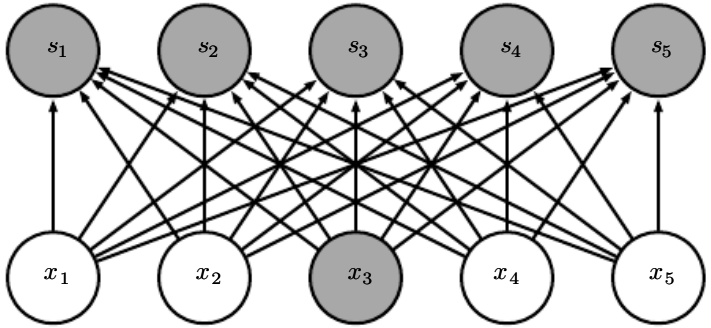
LeNet 1989

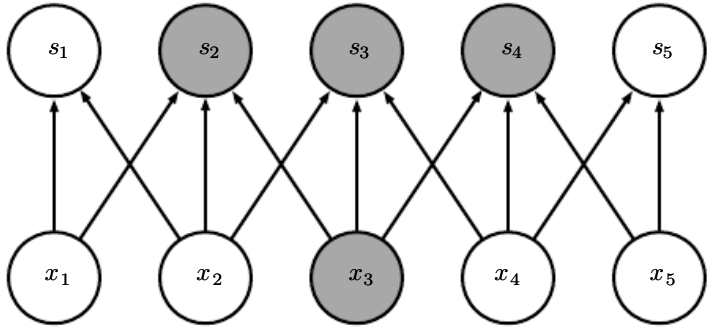


- *Filters are of size 5 × 5, stride 1 Pooling is 2 × 2, with stride 2.*
- *How many parameters?*

Why Convolution ? Sparse Interactions

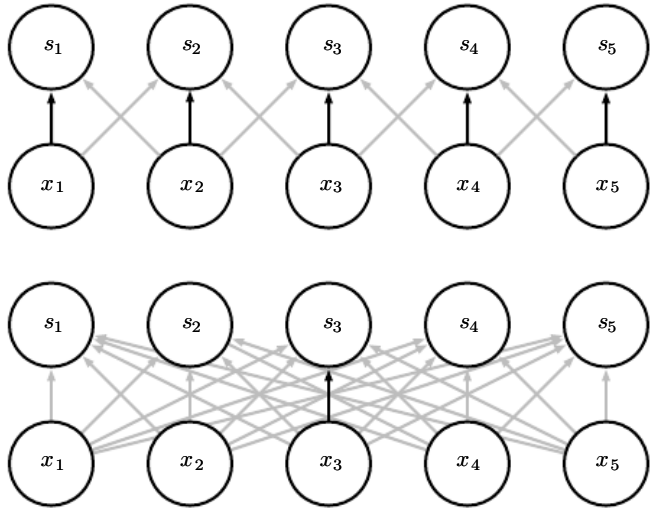
- **Traditional NN:** 1 parameter for each interaction (no sparsity), most interactions = 0
- **CNN:** sparse connections due to small filter size
  - Can detect small, meaningful features such as **edges** with small kernels
- Store fewer parameters -- O(m × n) versus O(k × n)
  - Low memory requirements
  - Faster training and inference
- Efficiently describe complicated interactions between many variables from simple building blocks that each describe only sparse interactions

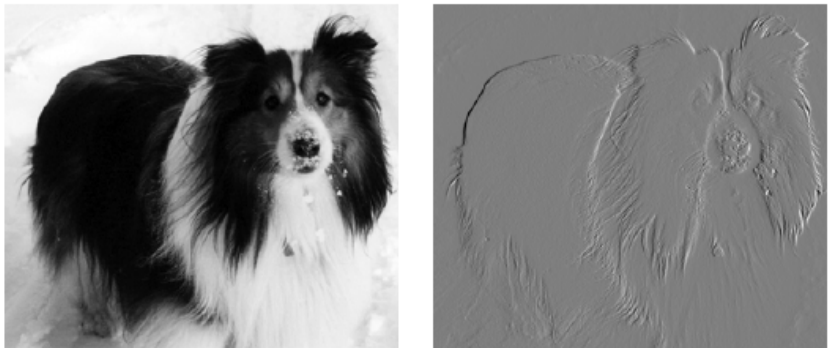




Why Convolution ? Parameter Sharing

- **Traditional NN:** Each weight/parameter used only once (multiplied by one element of the input and then never revisited)
- **Parameter sharing (tied weights)** refers to using the same parameter for more than one function in a model
- **CNNs:** rather than learning a separate set of parameters for every location, we learn only one set





Filter of size 2 (250k vs 8 Bi. Params)

Why Convolution ? Equivariance

- Equivariance: $f(T(x)) = T(f(x))$
- **CNNs equivariance to trainslation:** Unshifted representation for the object same as the representation for the object after shifting
  - The form of parameter sharing used by CNNs causes each layer to be equivariant to translation
  - property is useful when we know some local function is useful everywhere (e.g. edge detectors)

- **CNN:** not naturally equivariant to scale or rotation of an image

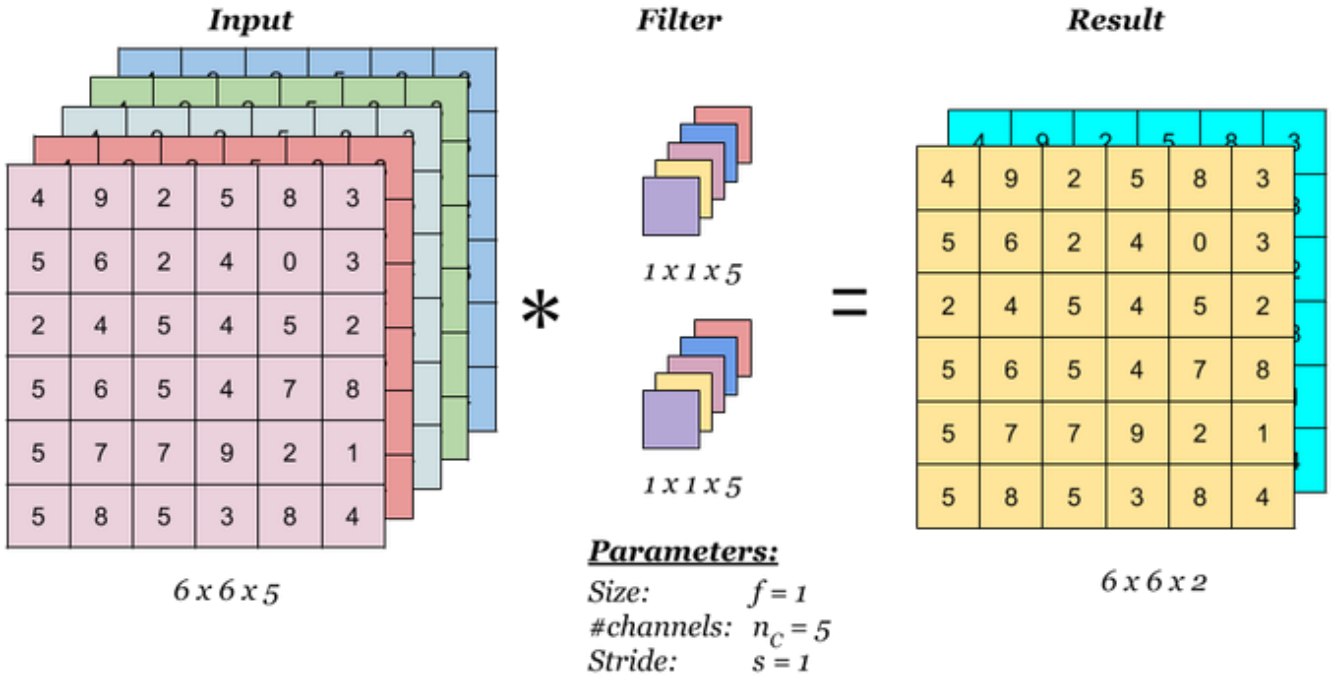
Computing number of parameters

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires three hyperparameters:
  - their spatial extent F,
  - the stride S,
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = (W_1 - F)/S + 1$
  - $H_2 = (H_1 - F)/S + 1$
  - $D_2 = D_1$
- Introduces zero parameters since it computes a fixed function of the input
- Note that it is not common to use zero-padding for Pooling layers

Computing parameters

- The output dimension is calculated with the following formula:

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



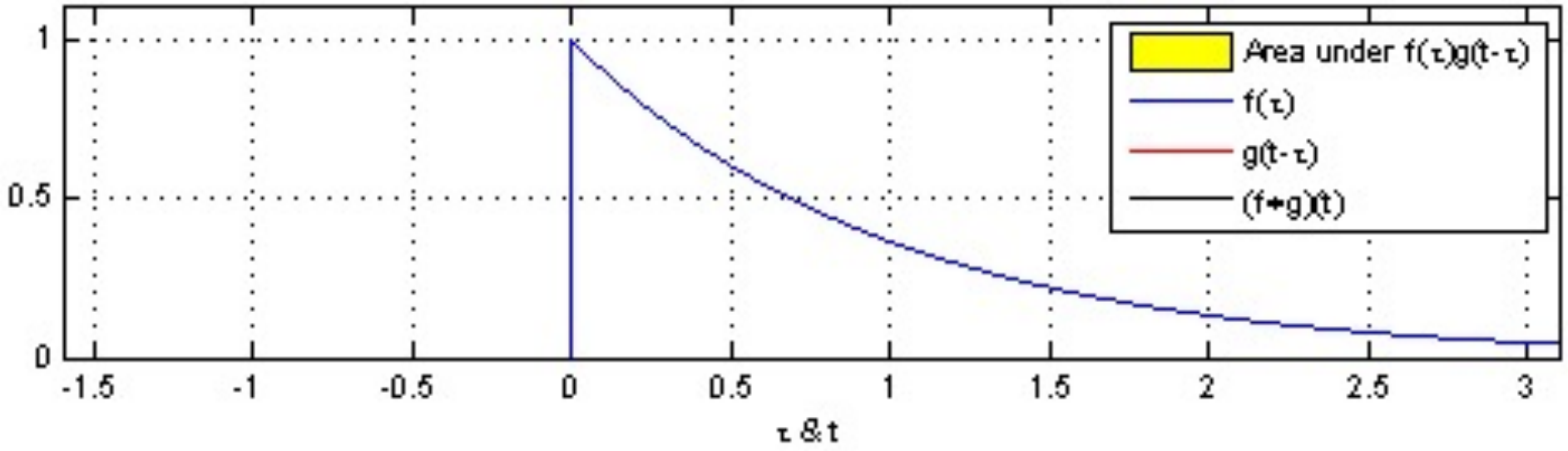
Mathematics behind CNNs

- A mathematical operation, denoted with an asterisk (*)
- General formula:
  - $(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau = \int_{-\infty}^{\infty} f(t-\tau)g(\tau)d\tau$
  - f is the *input*
  - g is the *kernel*
  - Output referred to as the *feature map*
- t not continuous but discrete:
  - $(f * g)(t) = \sum_{a=-\infty}^{\infty} f(\tau)g(t-\tau)$

Example of convolution

- Tracking a spaceship with a laser sensor.
  - Provides output *x(t)* with real-values *x,t*
- Sensor noisy, average to smooth the estimation
- Weighting function *w(a)*, with *a* as measure of age
  - w needs to be a probability density function
- Function: $s(t) = (x * w)(t) = \int_{-\infty}^{\infty} x(a)w(t-a)da$
- Unable to provide measurements every instant, discrete values:
  - $s(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a)$

Example of Convolution



2D-Convolutions

- Convoluting over two axes (for discrete variables):

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_m I(m,n)K(i-m, j-n)$$

  - This is commutative:

$$S(i,j) = (K * I)(i,j) = \sum_\infty \sum I(i-m, j-n)K(m,n)$$

- Cross-correlation, convolution without flipping the kernel:

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n)$$

  - Often used interchangeably as convolution in ML libraries.
    - Tensorflow implements cross-correlation as convolution
- Discrete convolution may be viewed as matrix multiplication
  - Condition: several entries constrained to be equal to other entries

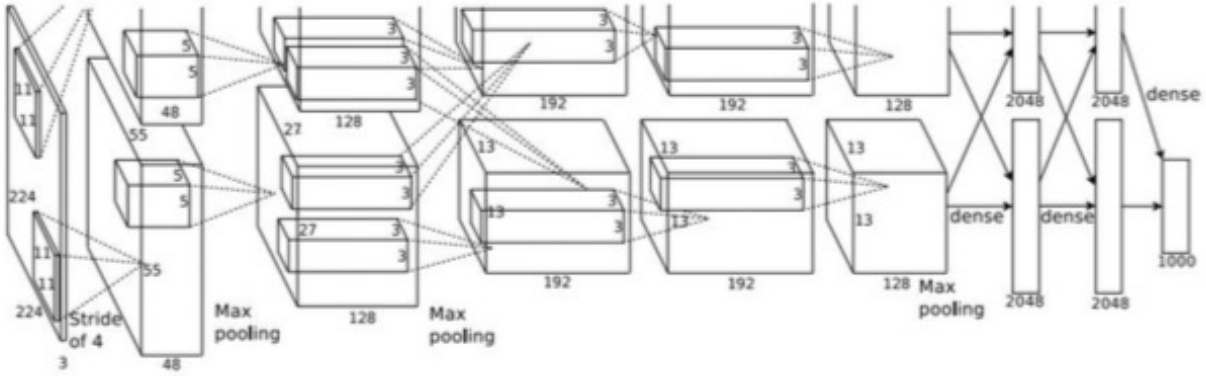
Some Applications of CNN

ImageNet

- **Dataset:** With over 14 Million images across 21,841 categories
- **Classification task:** Produce a list of object categories present in image. (**1000 categories**)
- "Top 5 error": rate at which the model does not output correct label in top 5 predictions

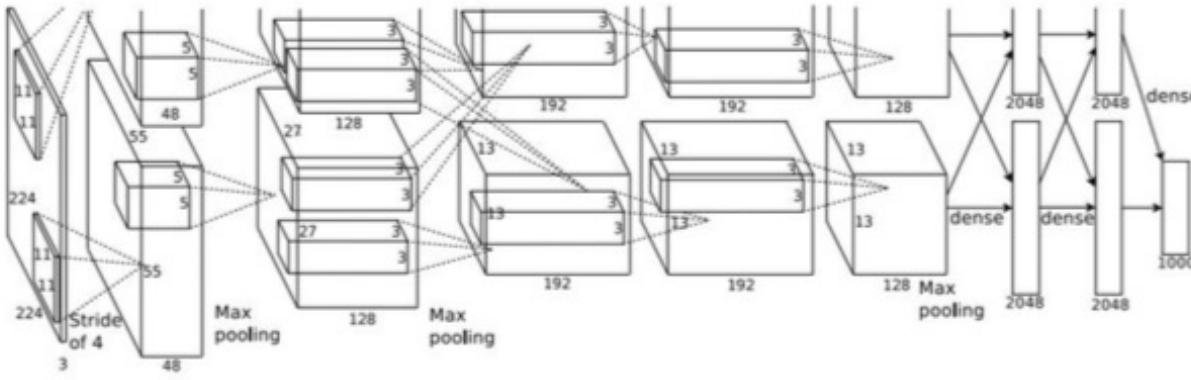
AlexNet (2012)

- Input 227x227x3 images
- **First layer** (CONV1): 96 11x11 filters applied at stride 4
- what is the output volume size? Hint: (227-11)/4+1 = 55
  - Output volume **[55x55x96]**
  - Parameters: (11*11*3)*96 = **35K**

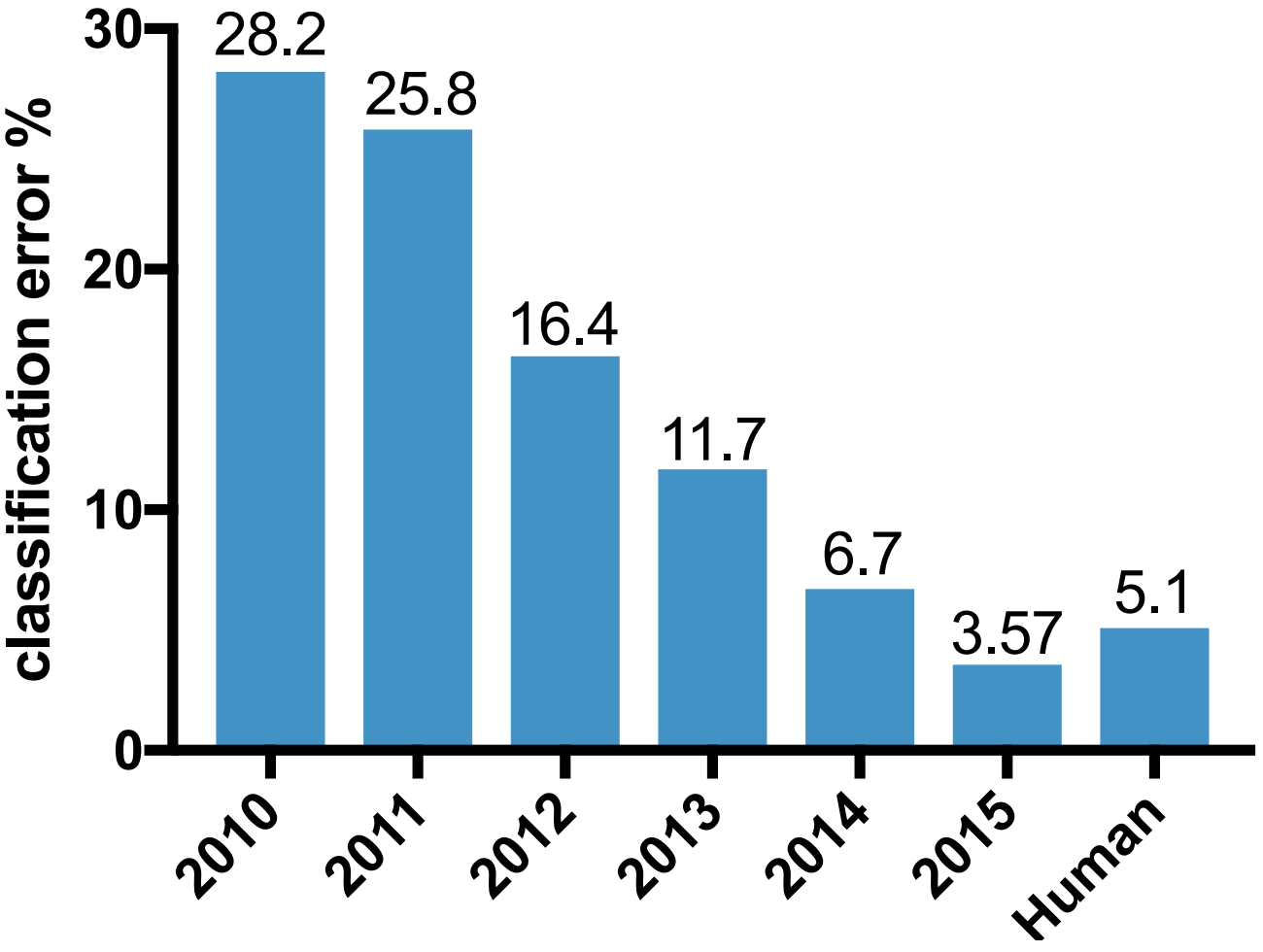


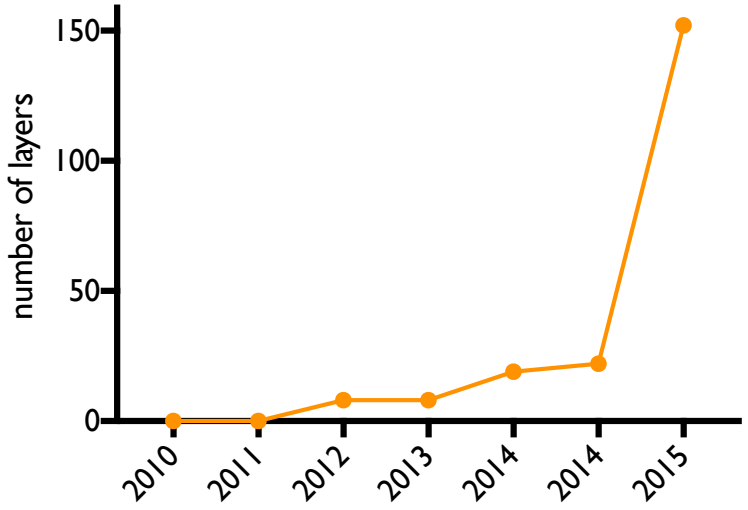
AlexNet (2012)

Full (simplified) AlexNet architecture:
[227x227x3] INPUT
[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0
[27x27x96] MAX POOL1: 3x3 filters at stride 2 [27x27x96]
NORM1: Normalization layer
[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2
[13x13x256] MAX POOL2: 3x3 filters at stride 2
[13x13x256] NORM2: Normalization layer [13x13x384]
CONV3: 384 3x3 filters at stride 1, pad 1 [13x13x384]
CONV4: 384 3x3 filters at stride 1, pad 1 [13x13x256]
CONV5: 256 3x3 filters at stride 1, pad 1 [6x6x256] MAX POOL3: 3x3 filters at stride 2
[4096] FC6: 4096 neurons
[4096] FC7: 4096 neurons
[1000] FC8: 1000 neurons (class scores)



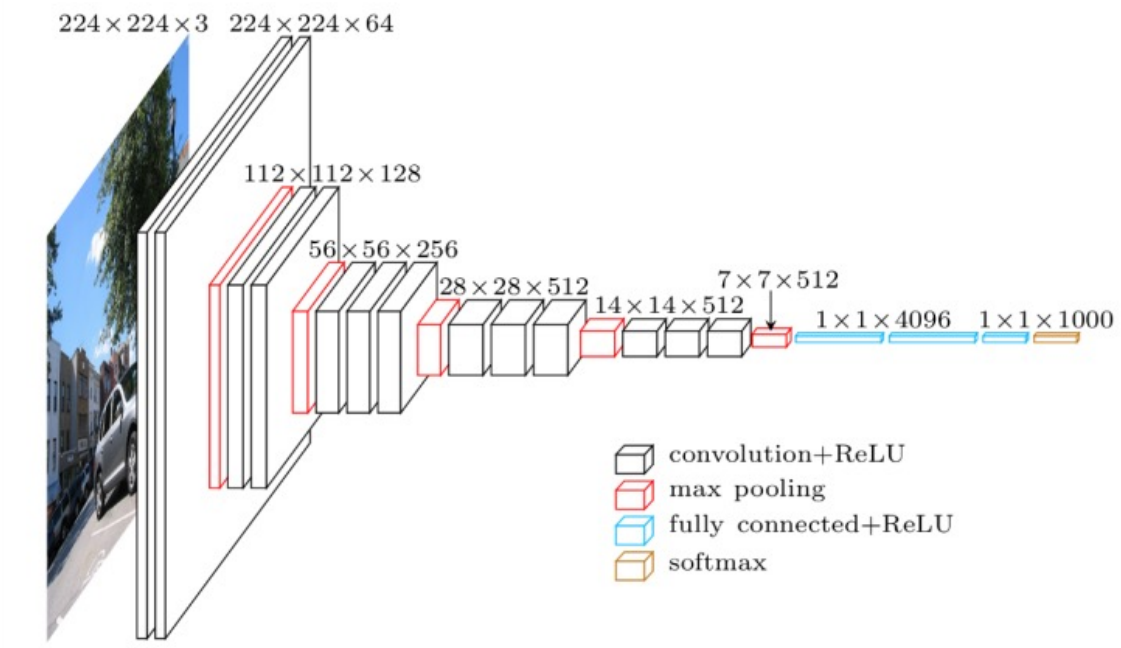
Progress using CNNs





- Large datasets help
- Over paramterized deep layers help

VGG-16 (2014)



- *The strength is in the simplicity: the dimension is halved and the depth is increased on every step (or stack of layers)*
- *Number of parameters: ~ 138 millions.*

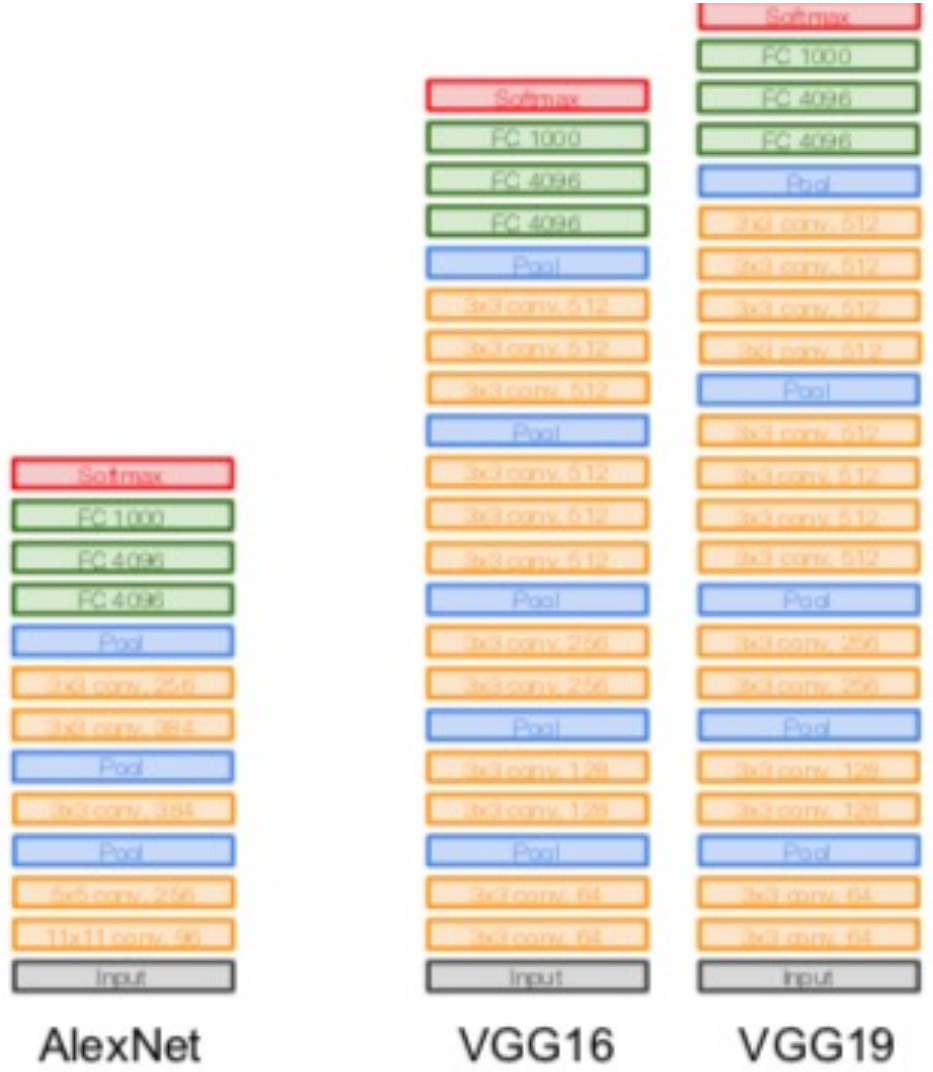
VGG-16

Small filters, Deeper networks

8 layers (AlexNet)
-> 16 - 19 layers (VGG16Net)

Only 3x3 CONV stride 1, pad 1 and 2x2 MAX POOL stride 2

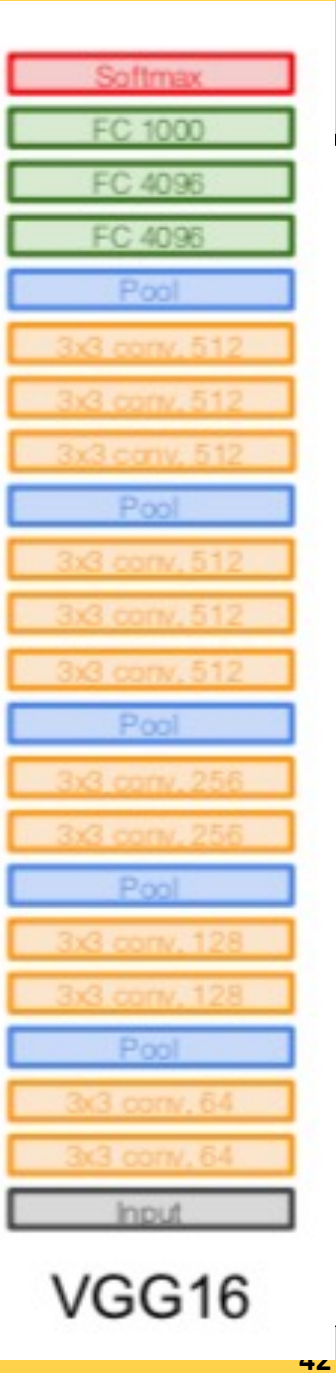
11.7% top 5 error in ILSVRC'13 (ZFNet)
-> 7.3% top 5 error in ILSVRC'14



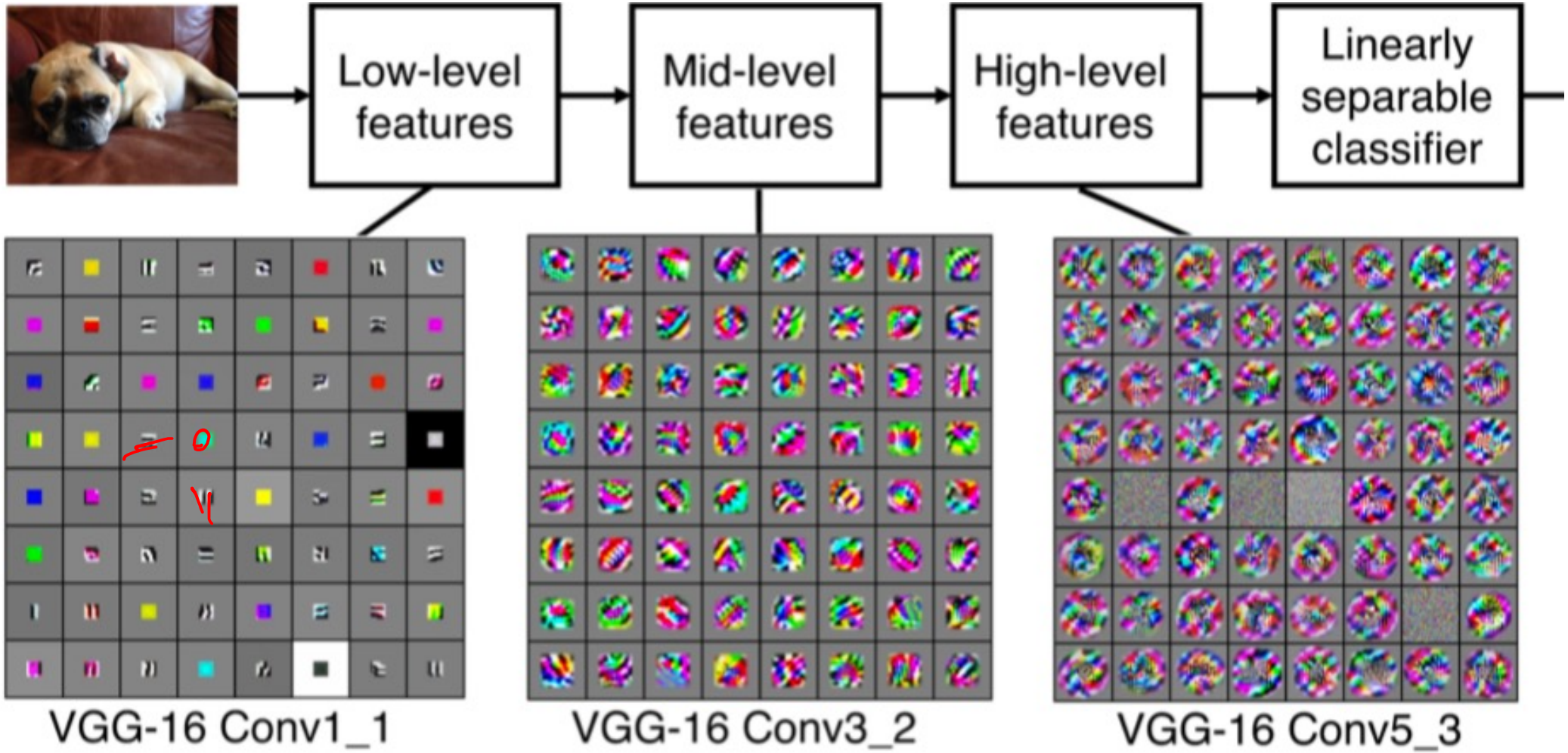
VGG

Q: Why use smaller filters? (3x3 conv)

Stack of three 3x3 conv (stride 1) layers has same **effective receptive field** as one 7x7 conv layer



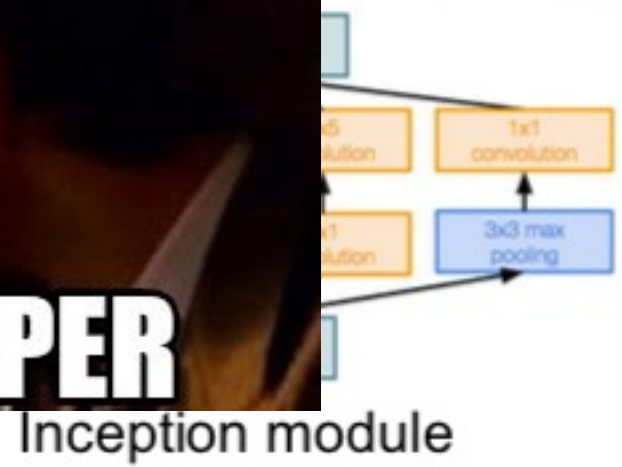
VGG-16 Example



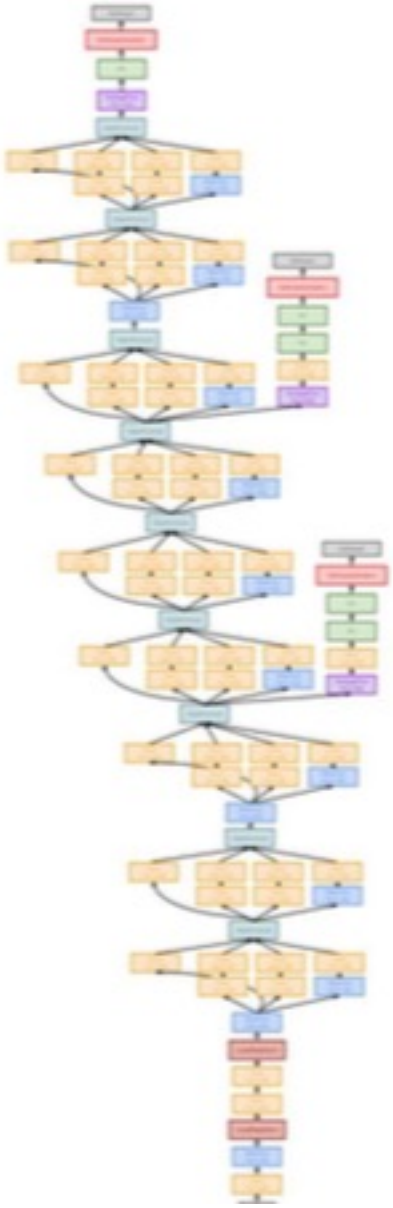
Inception Module

"Inception module": design a good local
within
these

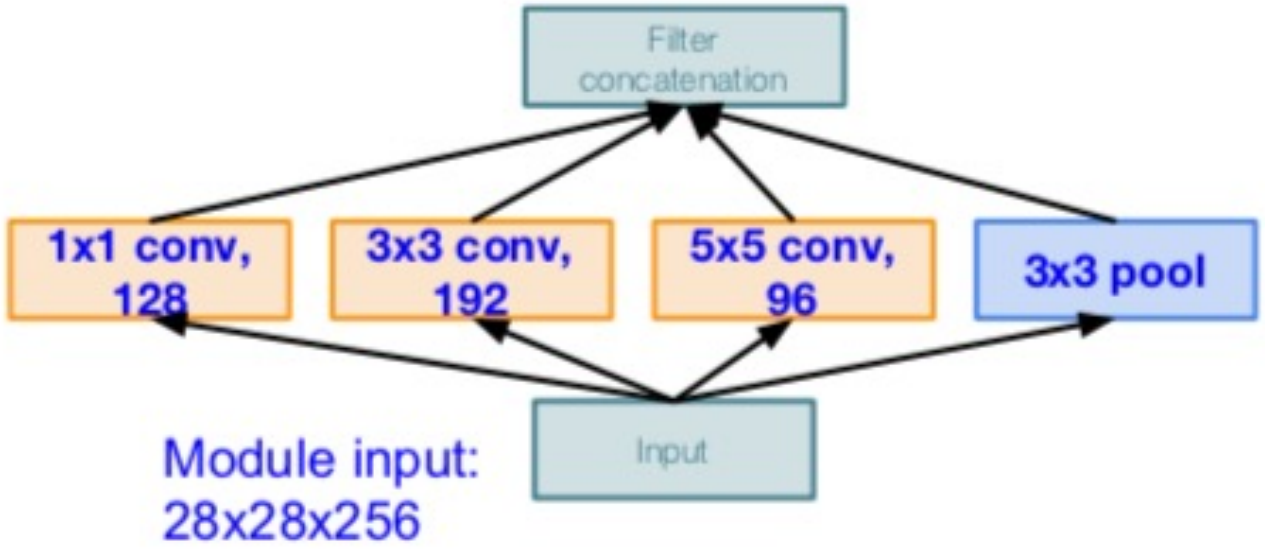
Apply para
the input fr
•- Multiple
for convolu
•- Pooling operation (3x3)
•Concatenate all filter outputs together depth-wise



Inception module

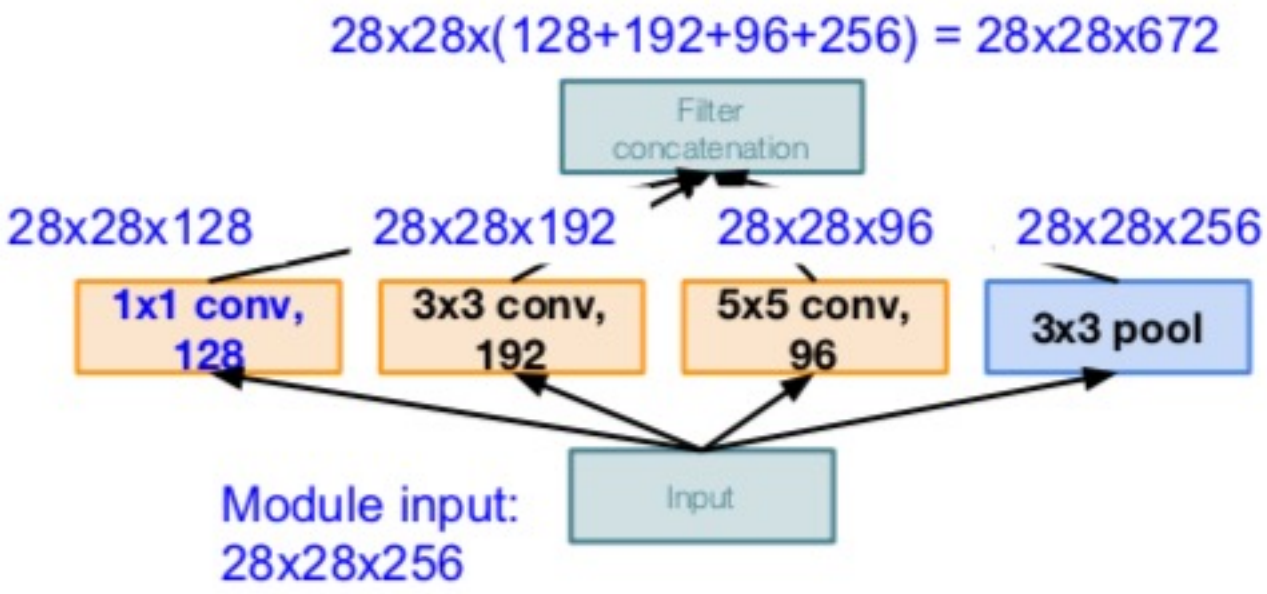


Naïve Inception Module



Q: what is the problem with this?

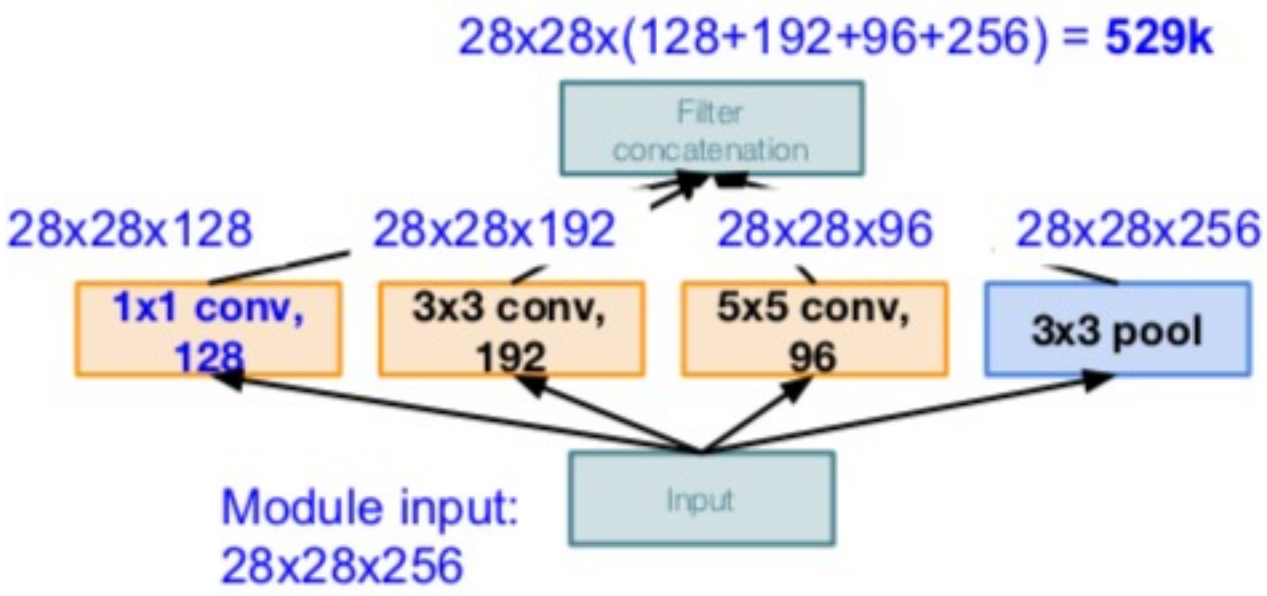
Naïve Inception Module



Q: what is the problem with this?

Conv Ops:
[1x1 conv, 128] 28x28x128x1x1x256
[3x3 conv, 192] 28x28x192x3x3x256
[5x5 conv, 96] 28x28x96x5x5x256
Total: 854M ops
Very expensive compute
Pooling layer also preserves feature depth, which means total depth after concatenation can only grow at every layer

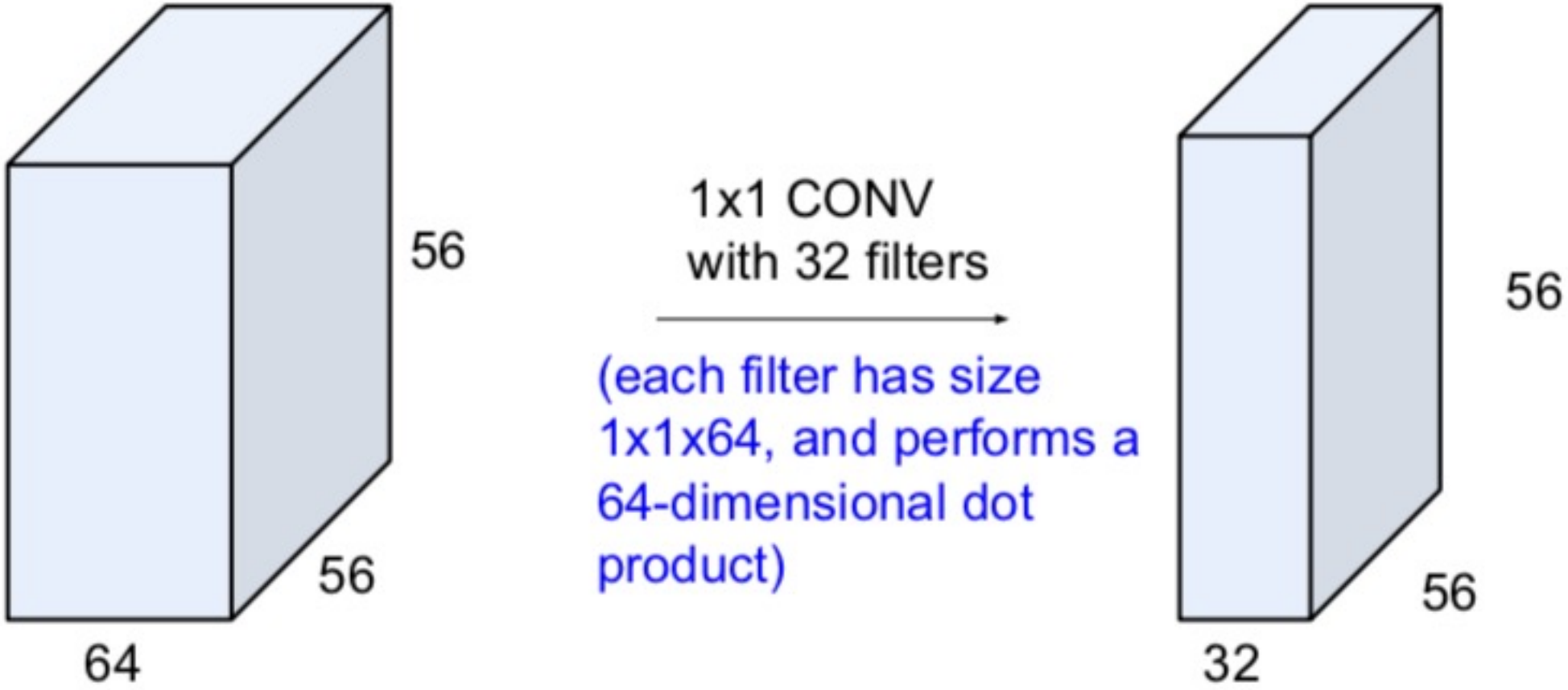
Improved Inception



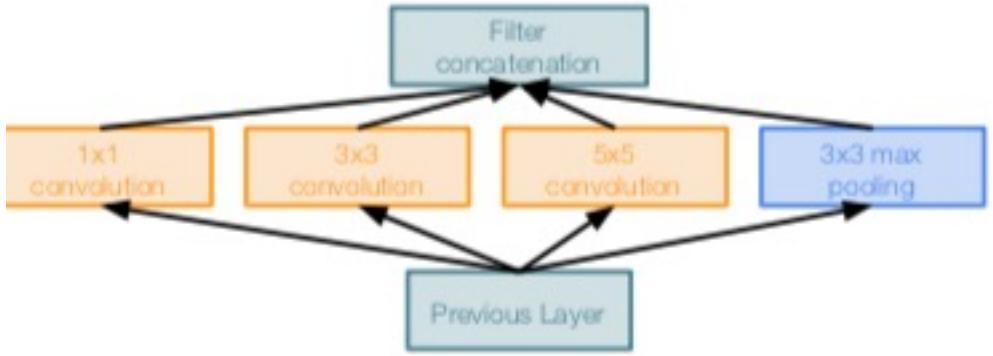
Q: what is the problem with this?

Solution: “bottleneck” layers that use 1x1 convolutions to reduce feature depth

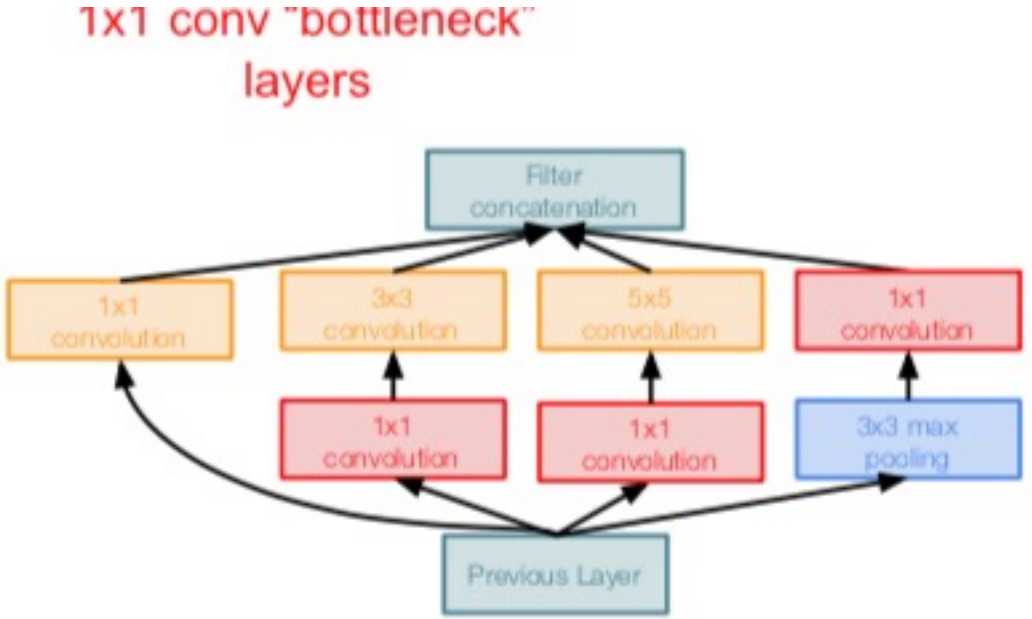
Why does 1x1 conv help? 1x



Improved Inception



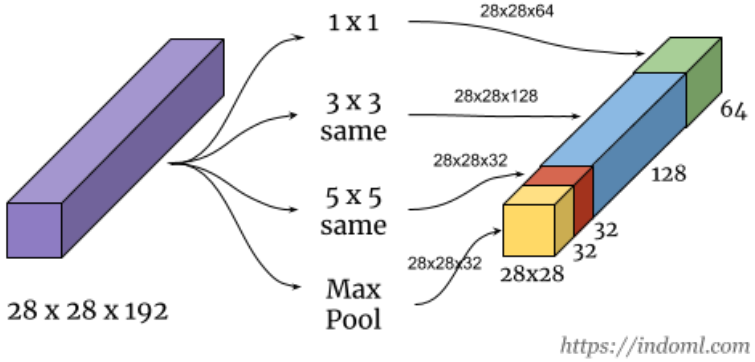
Naive Inception module

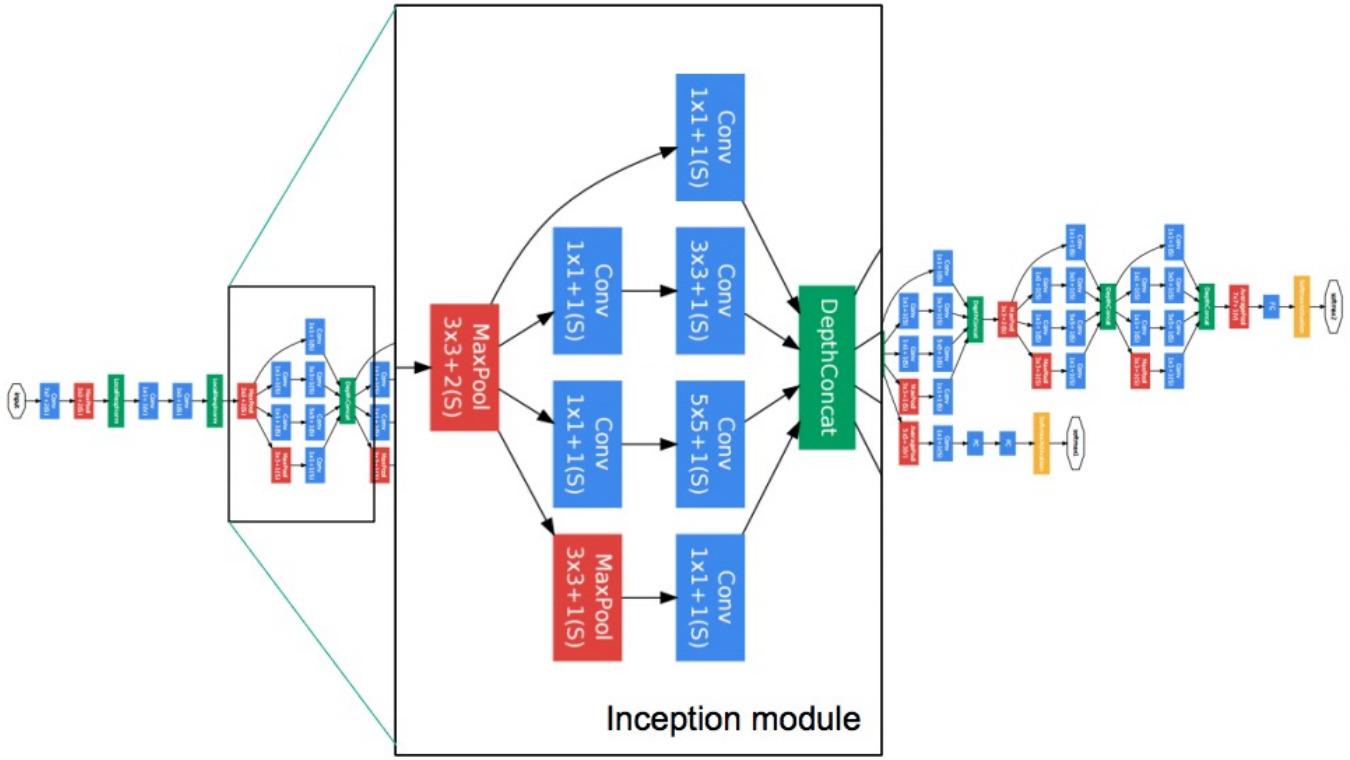


Inception module with dimension reduction

Inception

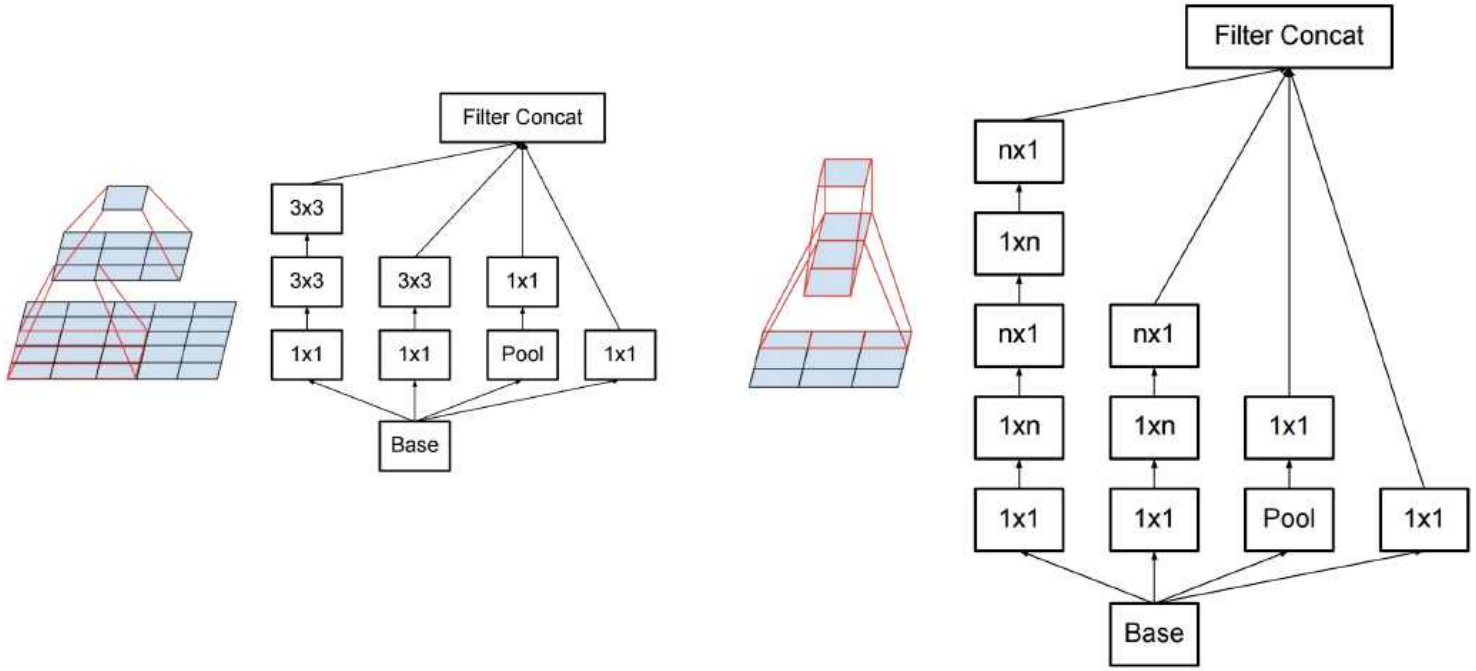
- Parallel paths with different receptive field sizes - capture sparse patterns of correlation in stack of feature maps
- Also include auxiliary classifiers for ease of training Also note 1 by 1 convolutions
- goes deeper in parallel paths with different receptive field sizes





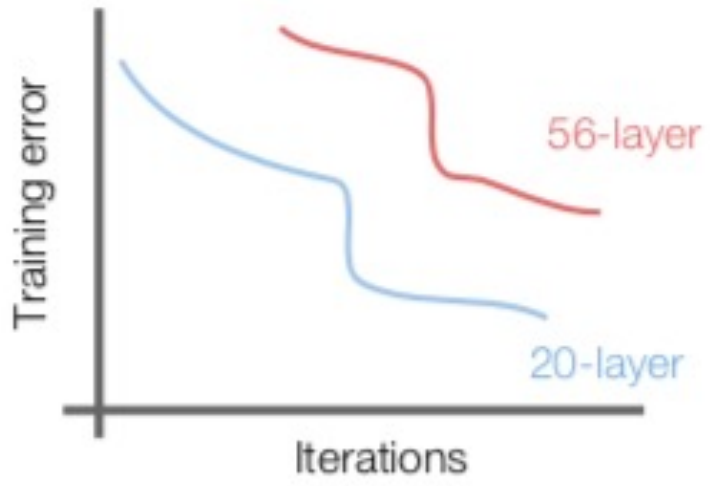
Inception V2, V3

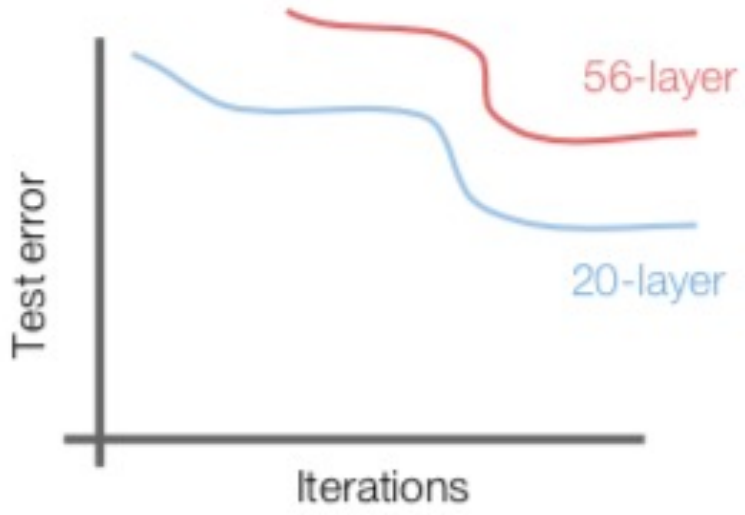
- Use Batch Normalization during training to reduce dependence on auxiliary classifiers
- More aggressive factorization of filters



Even deeper architectures

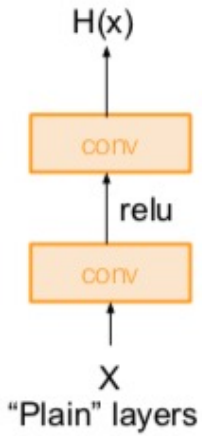
- What happens when we continue stacking deeper layers on a “plain” convolutional neural network?

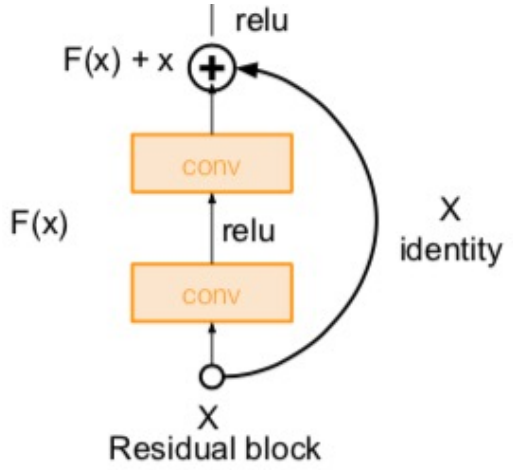




Solution for deeper networks

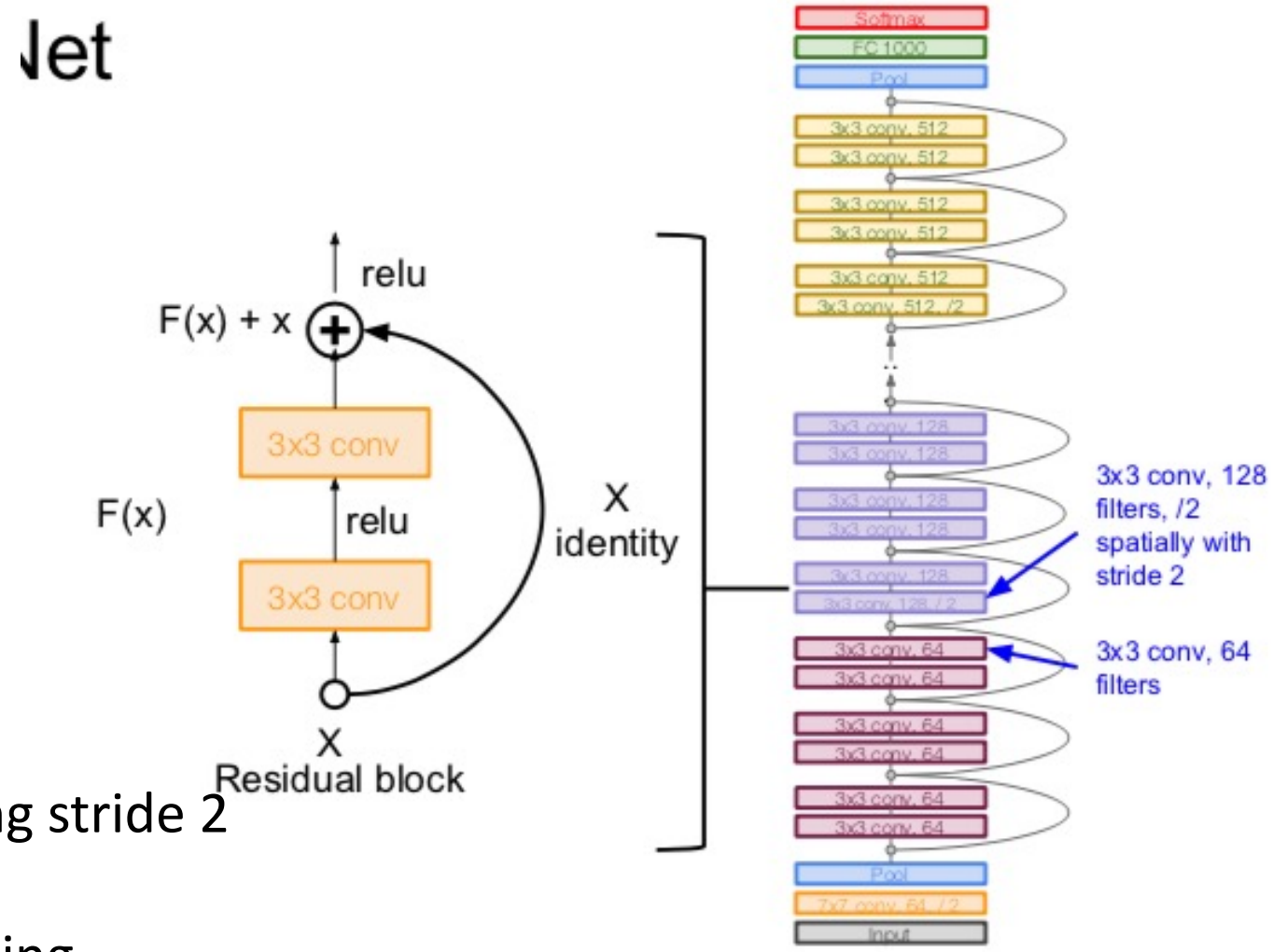
- **Hypothesis**: the problem is an *optimization* problem, deeper models are harder to optimize
- **Solution**: Use network layers to fit a residual mapping instead of directly trying to fit a desired underlying mapping





ResNets

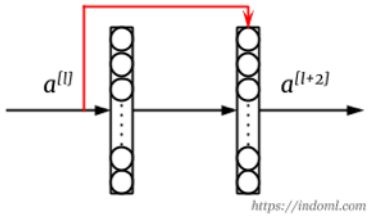
- Full ResNet architecture:
- - Stack residual blocks
- - Every residual block has
  - two 3x3 conv layers
- - Periodically, double # of
  - filters and downsample spatially using stride 2 (/2 in each dimension)
  - - Additional conv layer at the beginning



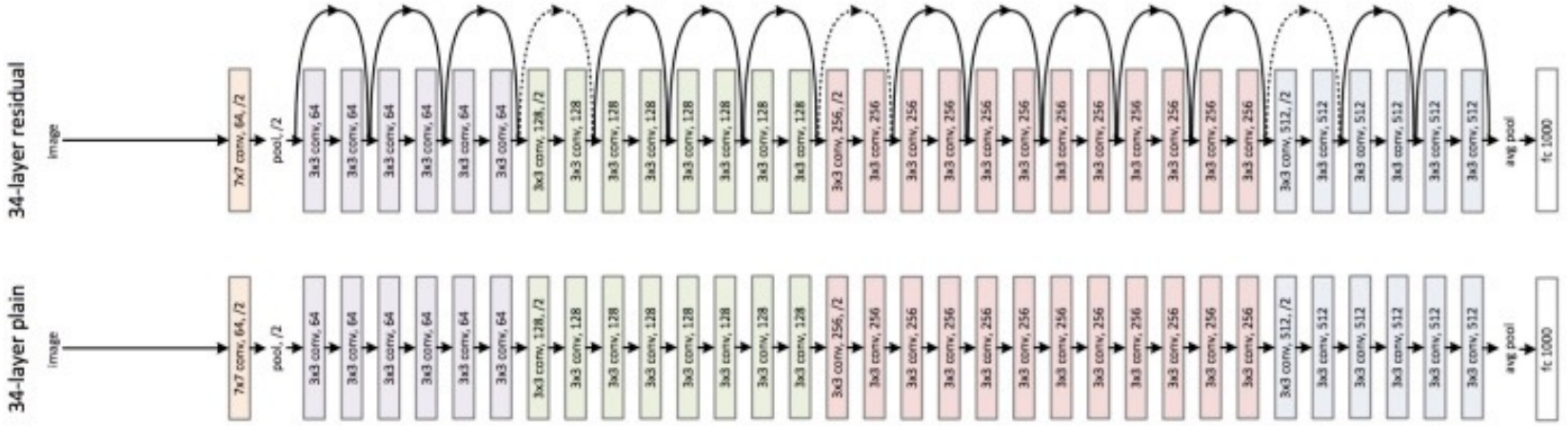
ResNet

- **Previous methods:** the input matrix calculates in two linear transformation with ReLU
- Problem with deep networks:
  - Once the number of layers reach certain number, the training error starts to raise again
  - Deep networks are also harder to train due to exploding and vanishing gradients problem.

$$z^{[l+2]} = W^{[l+2]} a^{[l+1]} + b^{[l+2]}$$
$$a^{[l+2]} = g^{[l+2]}(z^{[l+2]} + a^{[l]})$$



- **Residual network**: directly copy the input matrix to the second transformation output and sum the output in final ReLU

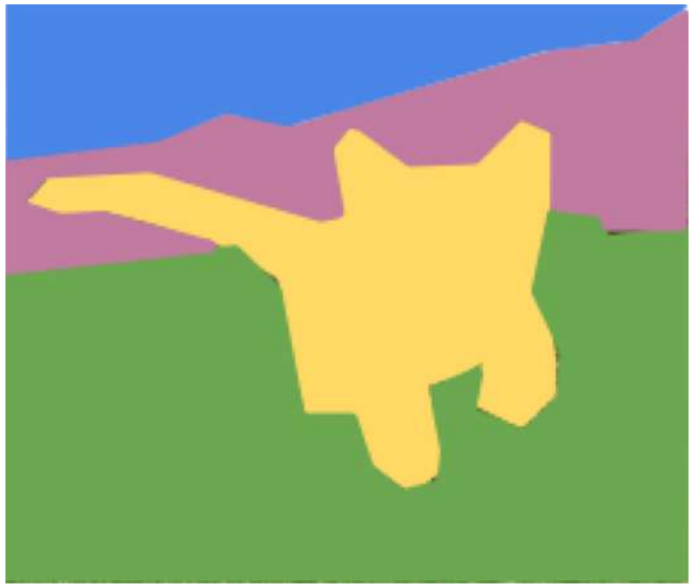


Advantages:

- performance doesn't degrade with very deep network
- cheaper to compute
- ability to train very very deep network

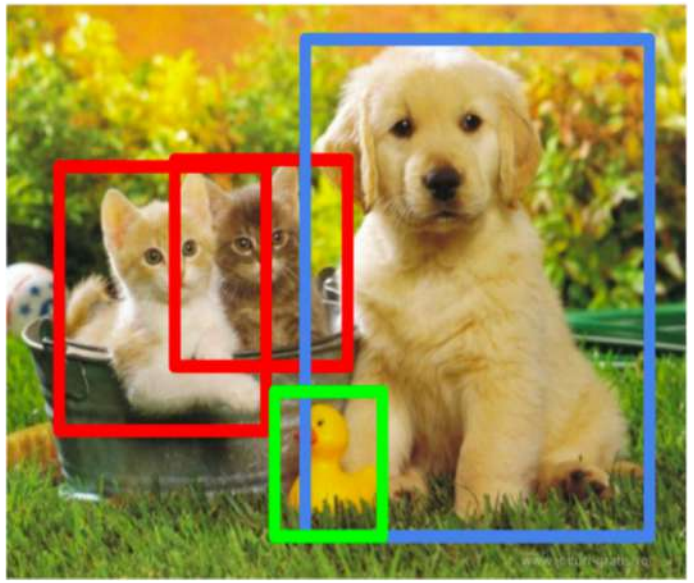
Beyond Classification

Semantic Segmentation



CAT

Object Detection

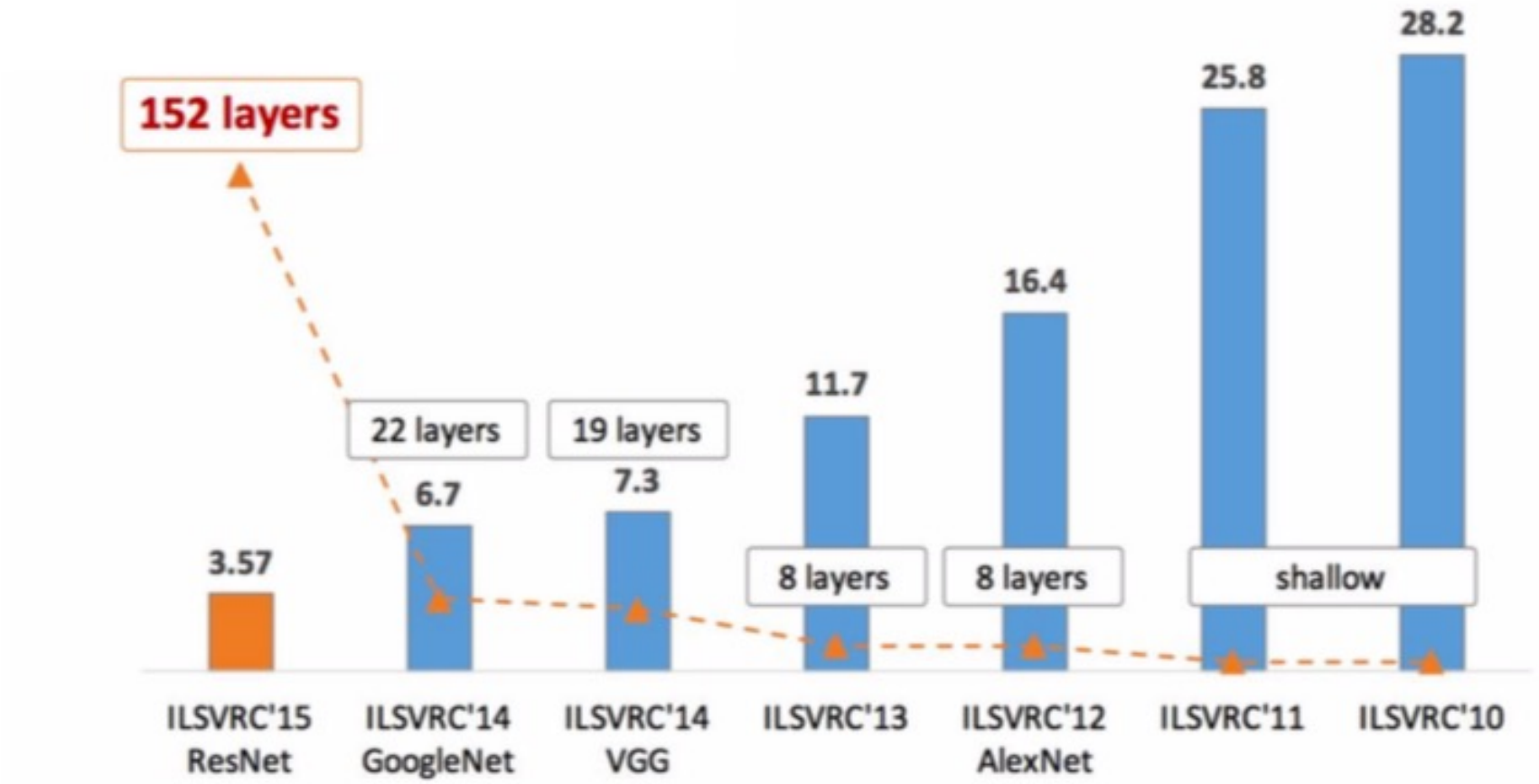


CAT, DOG, DUCK

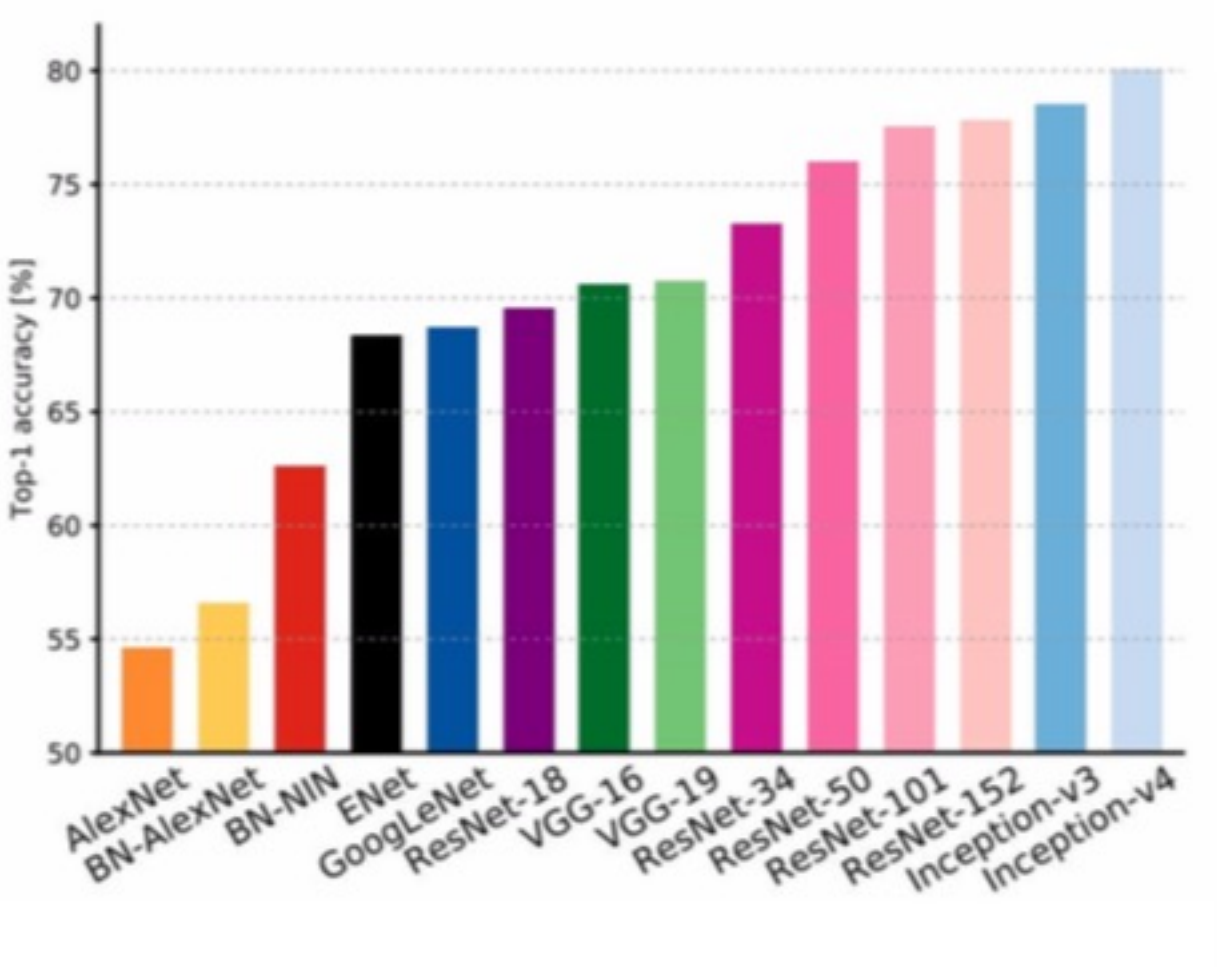
Image Captioning

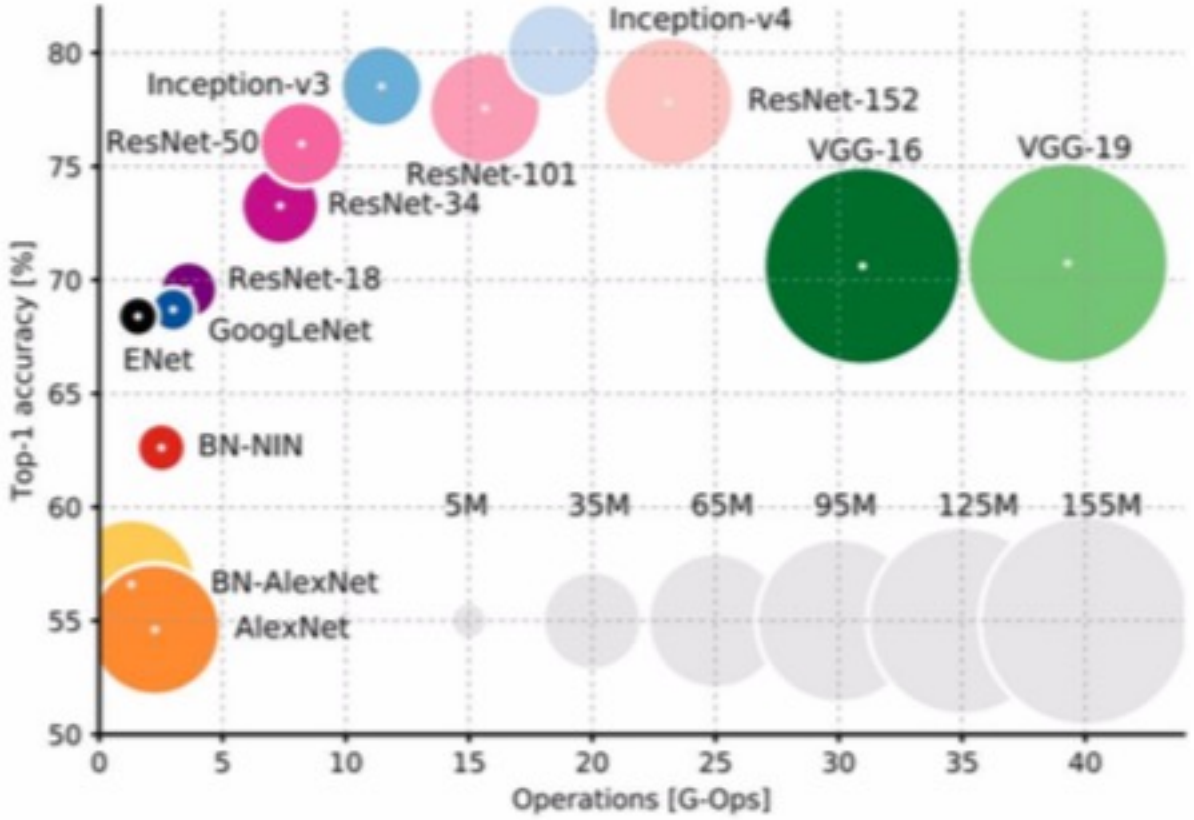
The cat is in the grass.

ImageNet Winners



Comparing Complexity

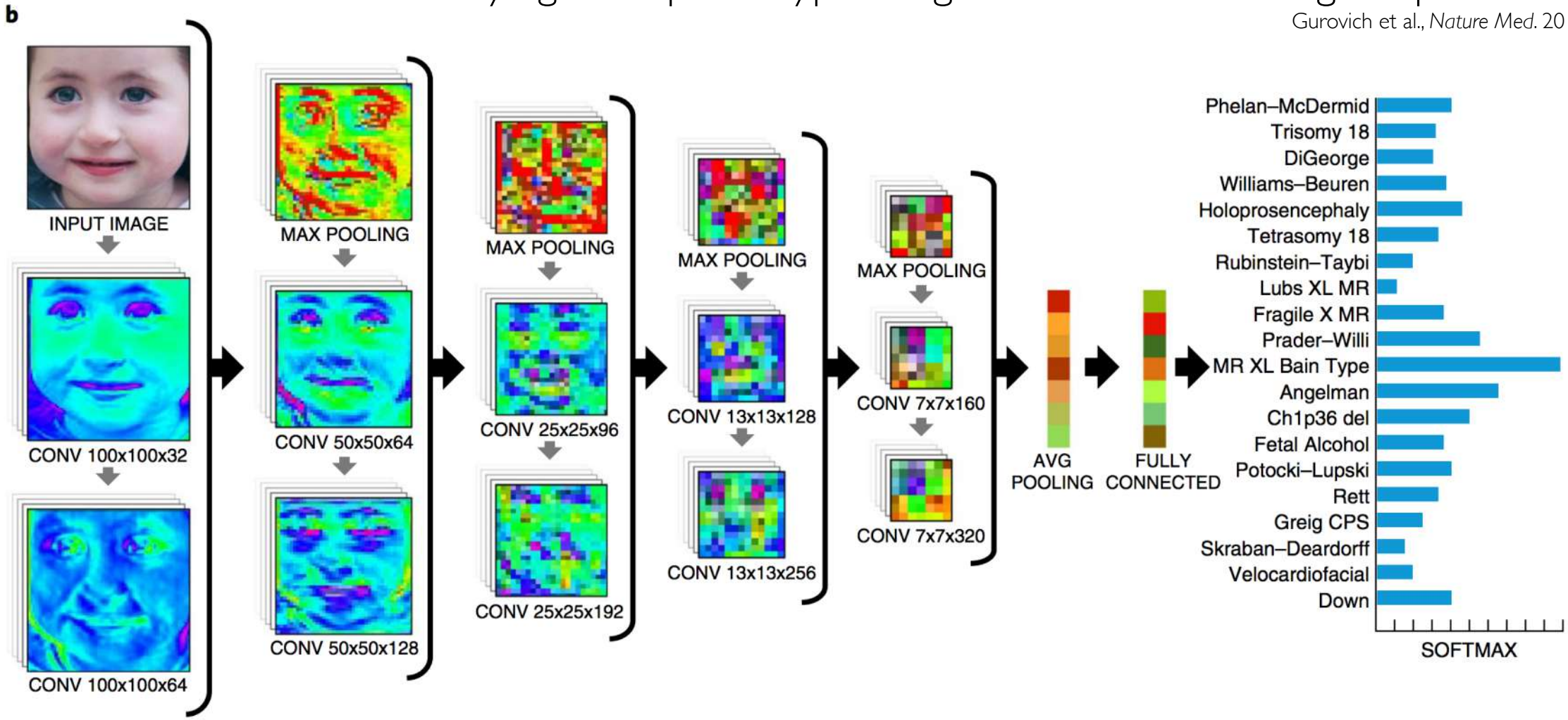




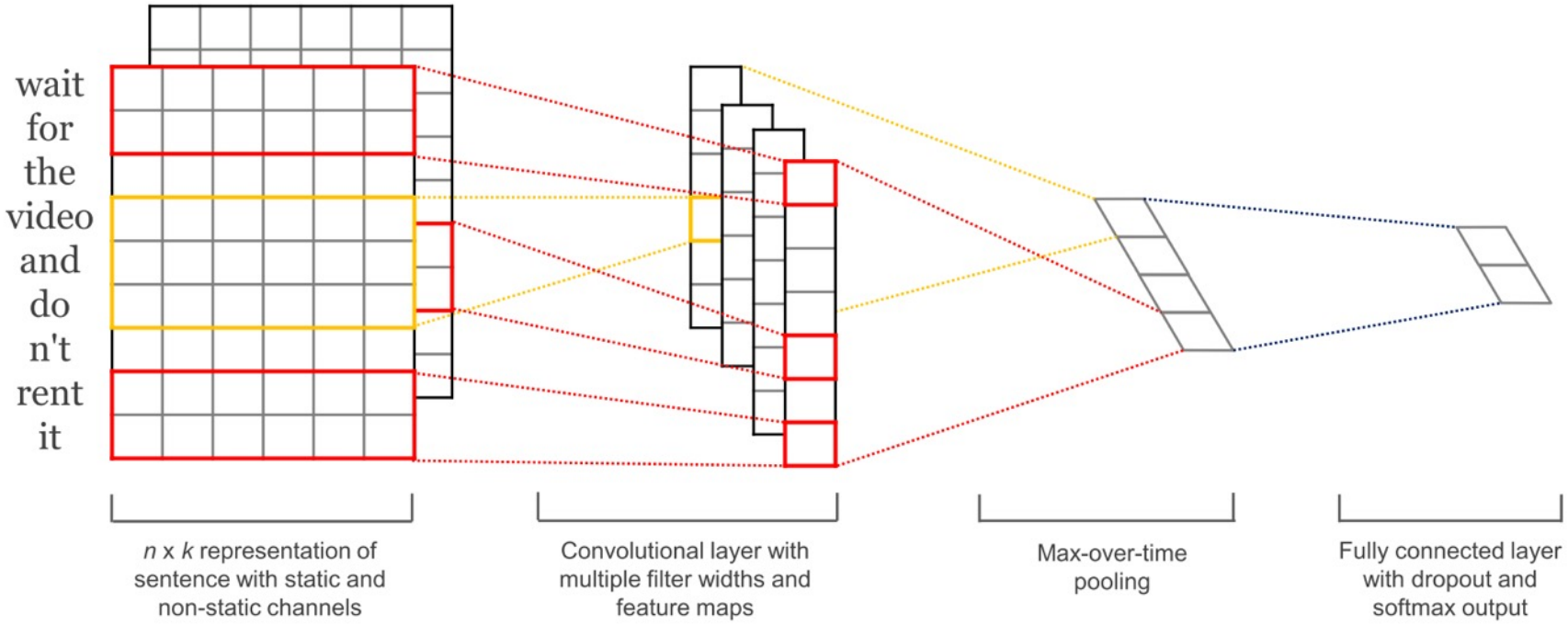
Healthcare

Identifying facial phenotypes of genetic disorders using deep learning

Gurovich et al., *Nature Med.* 2019

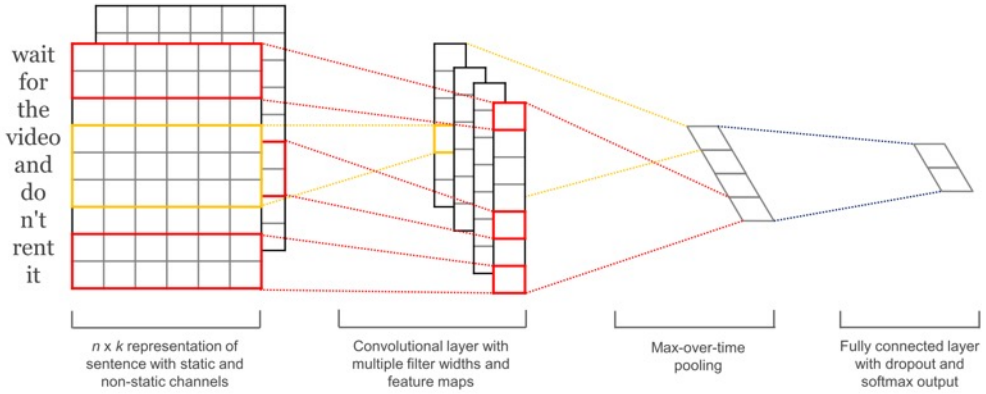


CNNs in Language



Kim, Y. (2014). Convolutional Neural Networks for Sentence Classification. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014), 1746–1751.

CNNs in Language



- Used for text classification, sentiment calssification,
  - Example: Movie review, does this person have positive sentiment on the movie or negative?
  - Product review
  - Politics: What is people's opinion about Trump given their comments?
- Many traditional NLP tasks,
  - such as POS tagging,
  - Semantic parsing
  - Search query retrieval

Kim, Y. (2014). Convolutional Neural Networks for Sentence Classification. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014), 1746–1751.

References

- Glorot & Bengio 2010 “Understanding the difficulty of training deep feedforward neural networks”
- Vanishing and exploding gradients: http://www.cs.toronto.edu/~rgrosse/courses/csc321_2017/readings/L15%20Exploding%20and%20Vanishing%20Gradients.pdf
- Dive into Deep Learning: https://en.d2l.ai/d2l-en.pdf