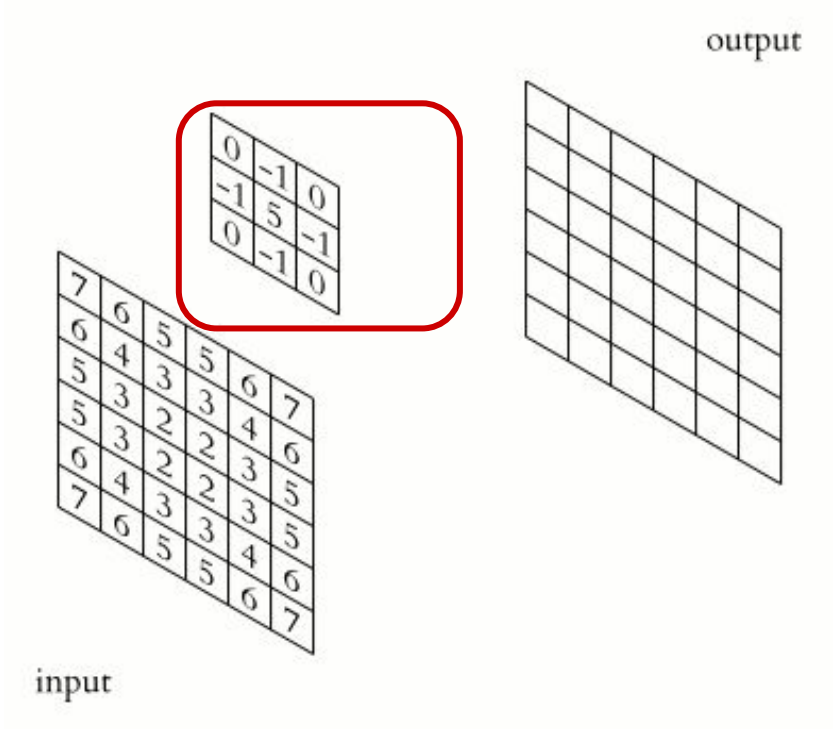
Announcements

- Pset-3 due today.
- Pset4 out, due Thursday, April 10th
- No laptops during class. Feel free to leave now

Are kernels same as convolutions

- Yes, convolutional kernels == convolutional filters



Quiz-3

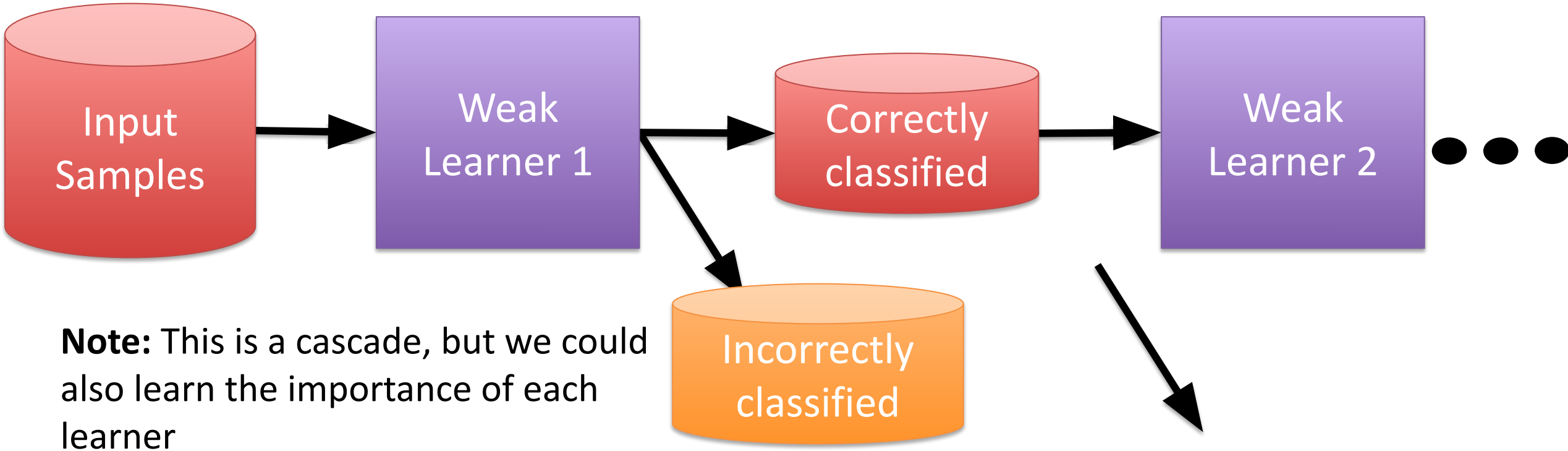
Q12 Why are the learners used in boosting called “weak” learners? Select all that apply. (You must select all correct options to receive full credit; no partial credit will be given.)

1 Point

- [x] They are intentionally designed for low precision.
- [ ] They are intentionally designed for high precision.
- [x] They are intentionally designed to have fast inference.
- [ ] They are intentionally designed to have poor performance when run in isolation.

Weak Learners

A set of simpler models that are combined to build a strong predictor



Note: This is a cascade, but we could also learn the importance of each learner

Cascaded weak learners

- **Low individual performance:** A single weak learner on its own doesn't achieve high prediction accuracy.
- **Sequential combination:**
  - Iteratively train weak learners
  - Each new learner focuses on correcting the errors made by the previous ones.

Quiz-3

Q12 Why are the learners used in boosting called “weak” learners? Select all that apply. (You must select all correct options to receive full credit; no partial credit will be given.)

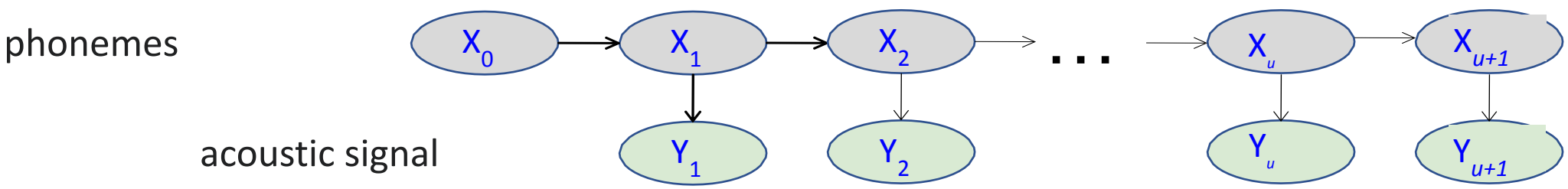
1 Point

- [x] They are intentionally designed for low precision.
- [ ] They are intentionally designed for high precision.
- [x] They are intentionally designed to have fast inference.
- [ ] They are intentionally designed to have poor performance when run in isolation.

Quiz-3

Q5 Assume you are using an HMM to model speech (model a word character and its acoustic sound). Which one of the following probabilistic interpretations is the closest to what this model does?

1 Point



Q5 Assume you are using an HMM to model speech (model a word character and its acoustic sound). Which one of the following probabilistic interpretations is the closest to what this model does?

1 Point

- (●) Model the joint probability of character and sound.
- ( ) Model the probability of a character given its sound.
- ( ) Model the unary probability of a character
- ( ) Model the unary probability of the sound.

Quiz-3

Q2 Imagine k-means clustering were given the following set of data:
2 Points

	x_1	x_2	x_3
1	1	1	1
2	0	0	0
3	0	1	1
4	1	1	0
5	0	0	1
6	0	1	0
7	1	0	0
8	0	0	1

If k=2 and where the initial starting points are the first two examples (1,1,1) and (0,0,0).

- Determine which points belong to Cluster-1 and which to Cluster-2?
- Number of iterations does it take?

  - **2 mins** ⏰
    - Enter in slido in the next slide.

What are the resulting clusters?

Quiz-3

Q2 Imagine k-means clustering were given the following set of data:
2 Points

	x_1	x_2	x_3
1	1	1	1
2	0	0	0
3	0	1	1
4	1	1	0
5	0	0	1
6	0	1	0
7	1	0	0
8	0	0	1

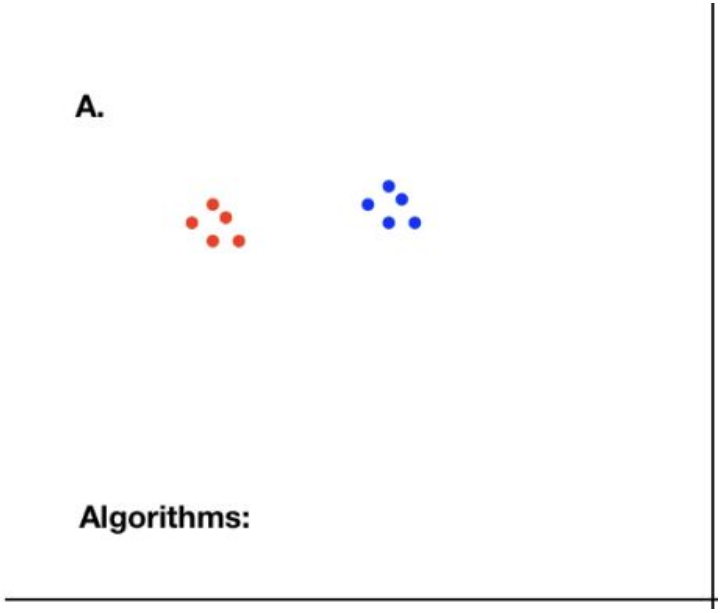
If k=2 and where the initial starting points are the first two examples (1,1,1) and (0,0,0).

Q2.1 Which points are in each cluster when it halts?
1 Point

- ○ Cluster1: (1,3,4,5,6,7,8) | Cluster2: (2)
- ○ Cluster1: (1,3,4,5,6) | Cluster2: (2,7,8)
- ◉ Cluster1: (1,3,4) | Cluster2: (2,5,6,7,8)
- ○ Cluster1: (1,3) | Cluster2: (2,4,5,6,7,8)
- ○ Cluster1: (1) | Cluster2: (2,3,4,5,6,7,8)

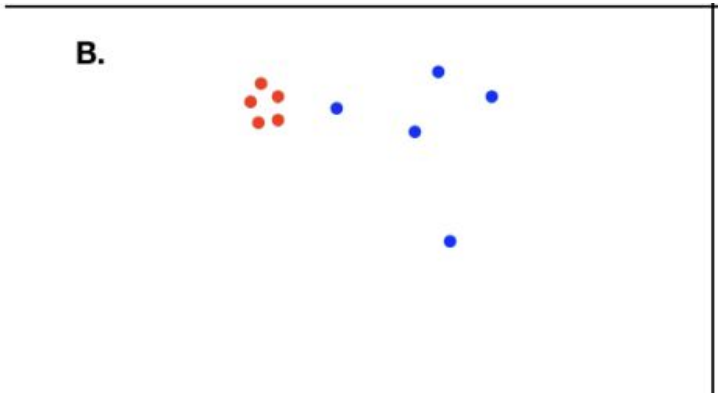
Q2.2 How many iterations does it take?
1 Point

- ◉ 1
- ○ 2
- ○ 3
- ○ 4
- ○ 5



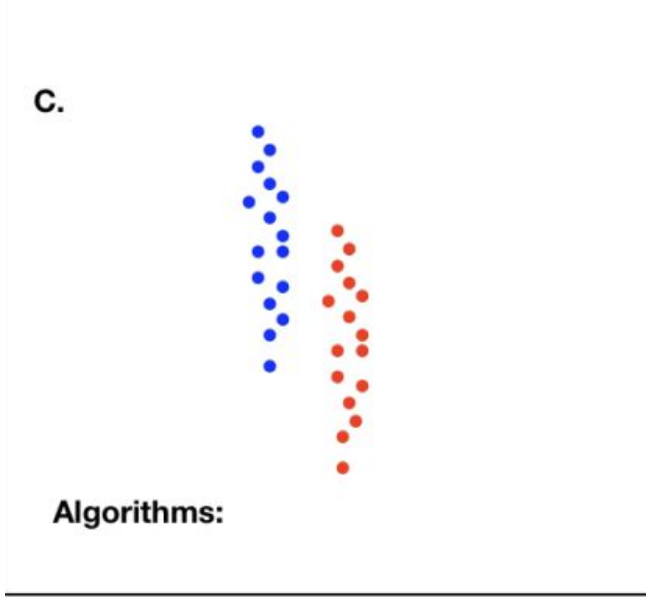
- [x] K-means
- [x] single link clustering
- [x] gaussian mixture models
- [ ] None of these

Why: When the points are so nicely separated, most algorithms converge to the same result.



- K-means
- single link clustering
- **gaussian mixture models**

Why: When the points / cluster assignment are more spread out, think Gaussian models with varied standard deviations.



- K-means
- **single link clustering**
- **gaussian mixture models**

Why:

- When the points / cluster assignment are more spread out, think Gaussian models with varied standard deviations.
- Single link (or agglomerative) also seems very plausible.

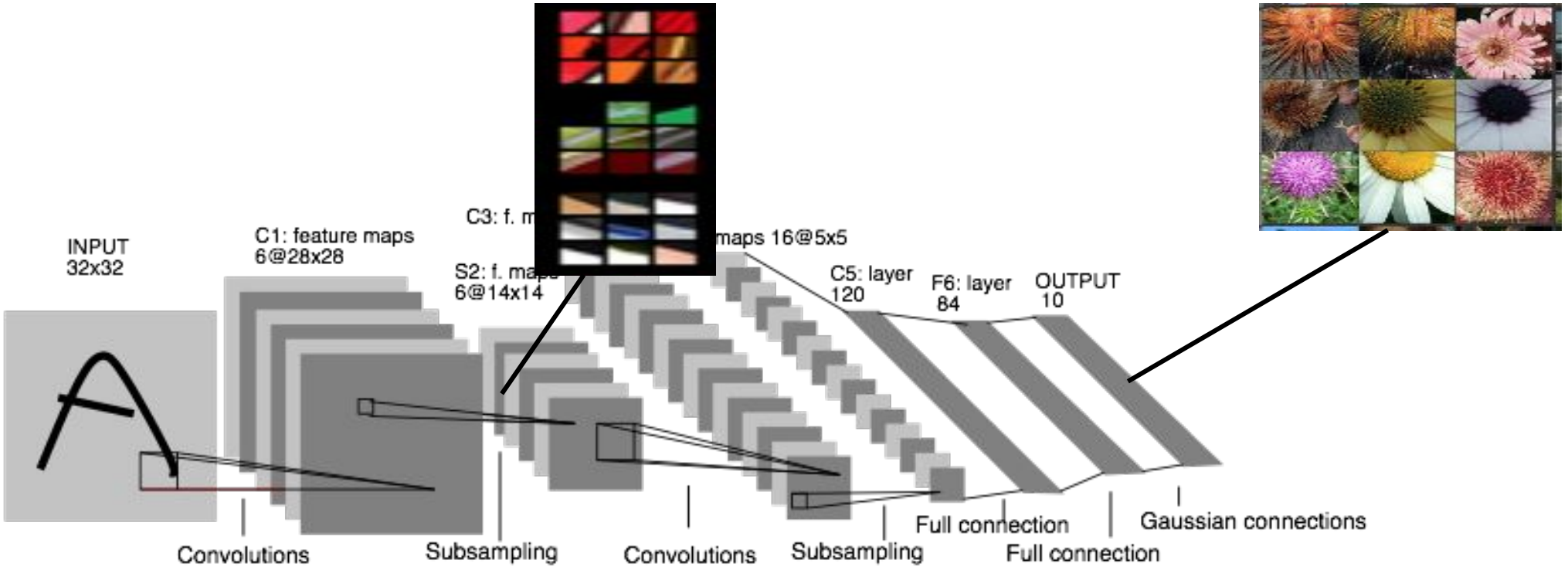
Last time: Convolutional neural networks

Convolution Filter	**M X N**
Channels	Stack of convolutional filters **C**
Feature maps	Outputs after convolution operation applied at a layer
Layers (N)	Stack of feature maps
Pooling (nothing to learn!)	Max, mean - across channels or within a feature map

Remember: We are learning the convolutional filters and channels during forward and backward pass!

Convolutional Neural Network

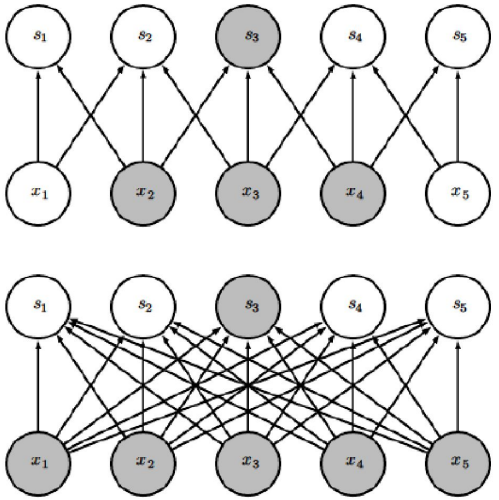
A better architecture for 2d signals

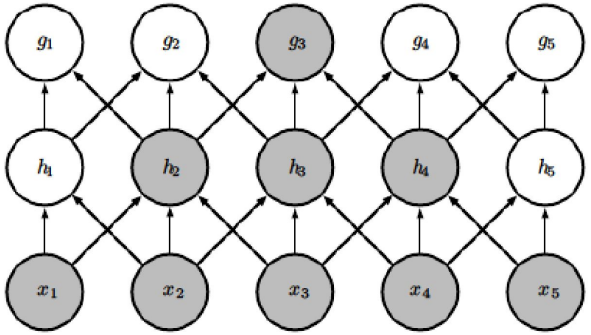


LeNet

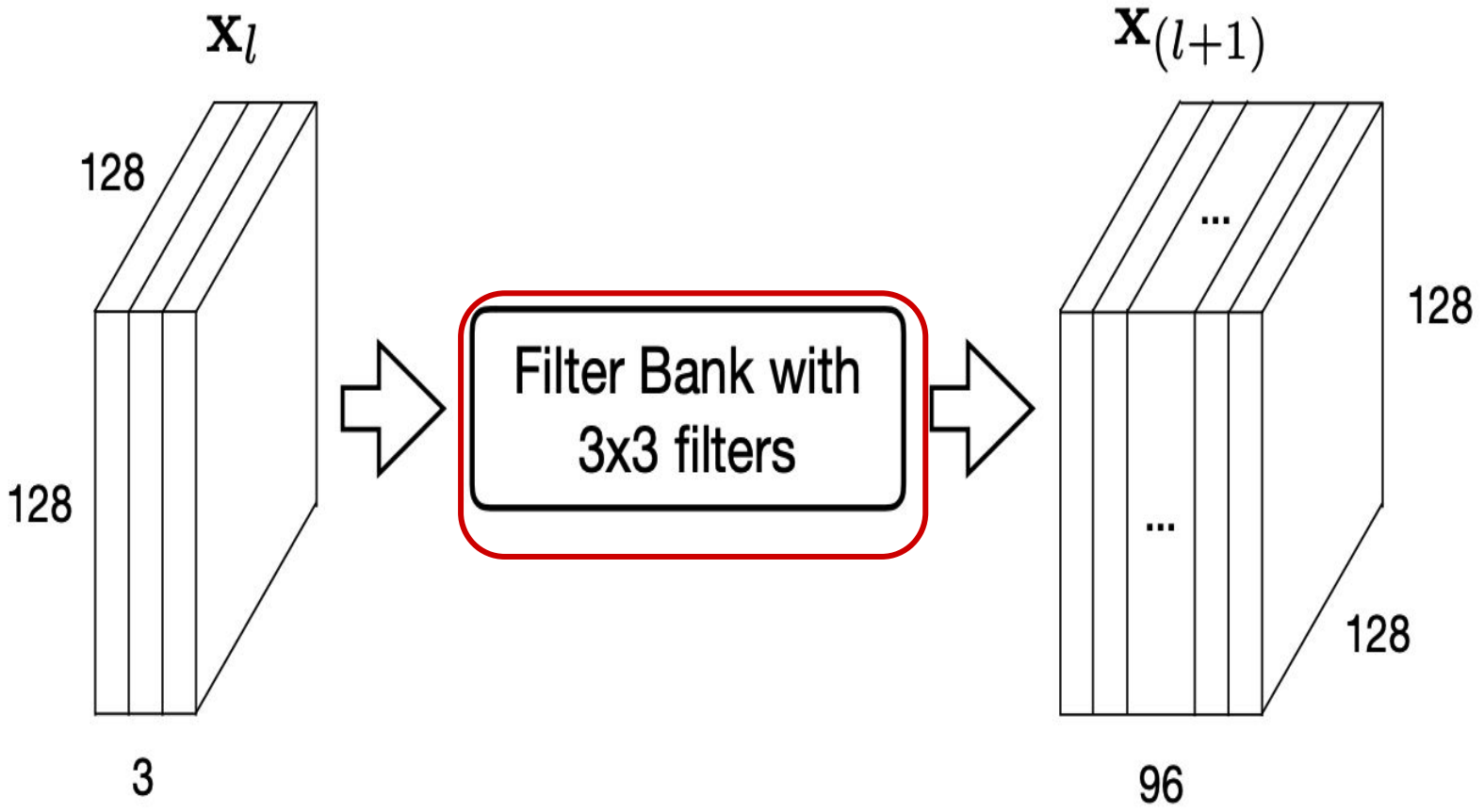
CNN Benefits: Sparsity

- CNNs have sparse interactions, because the kernel is smaller than the input
  - E.g. in thousands or millions pixel image, can detect small meaningful features such as edges
- Very efficient computation!
  - For m inputs and n outputs, matrix multiplication requires $O(m \times n)$ runtime (per example)
  - For k connections to each output, need only $O(k \times n)$ runtime
- Deep layers have larger effective inputs, or receptive fields





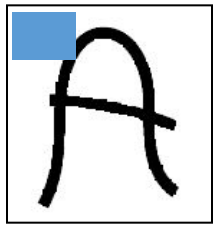
CNN Benefits: Parameter sharing

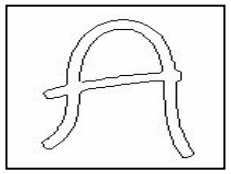


- Filter weights are shared across all locations
- Statistically efficient – learn from more data

CNN Benefits: Translation invariance

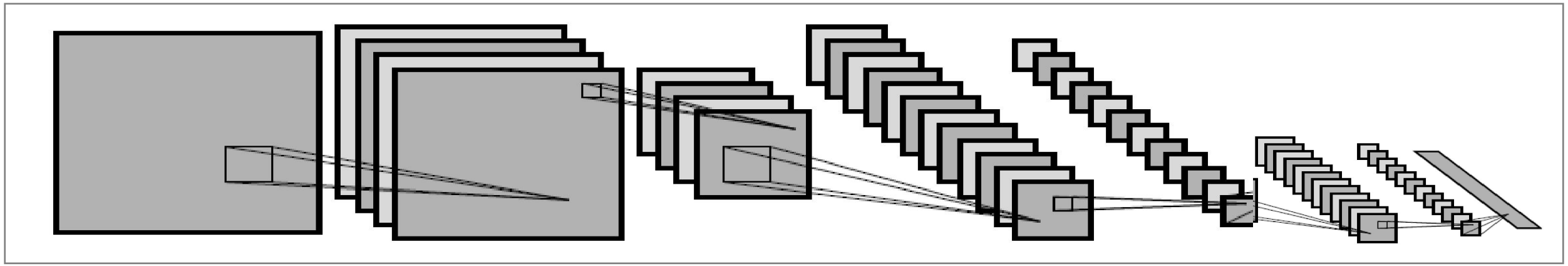
- Output is invariant to translation of input
  - spatial translation for images
  - temporal translation for time sequences
- useful when some function of a small local window is useful when applied to multiple input locations
- Note, not invariant to other transformations of input, such as large image rotation
- Pooling provides additional invariance to distortions



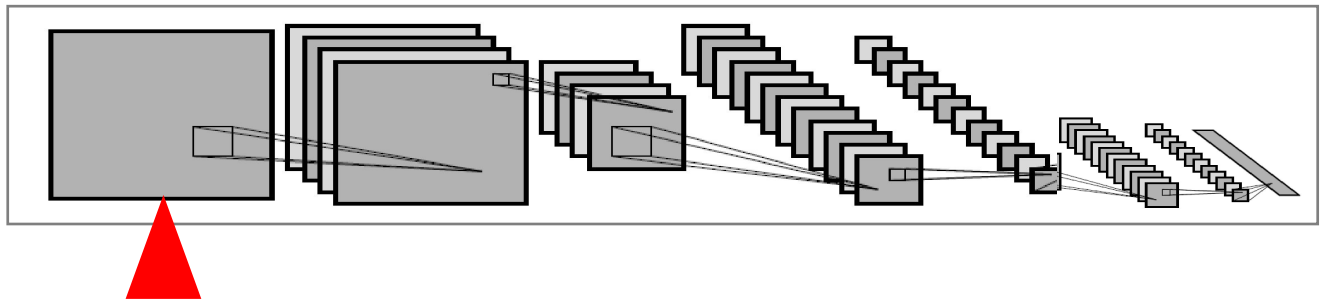


Examples of very popular CNNs

CIFAR-10 Demo ConvJS Network

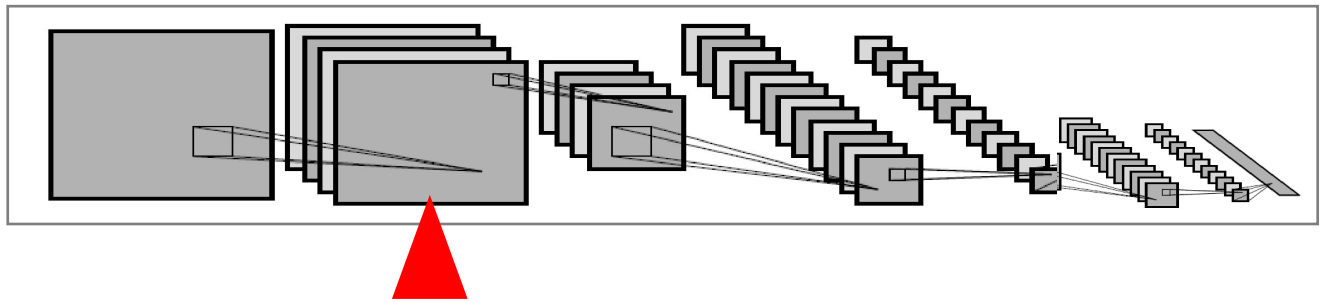


input image conv RELU pool conv RELU pool conv RELU pool fc



input (32x32x3)

filter size 5x5x3, stride 1

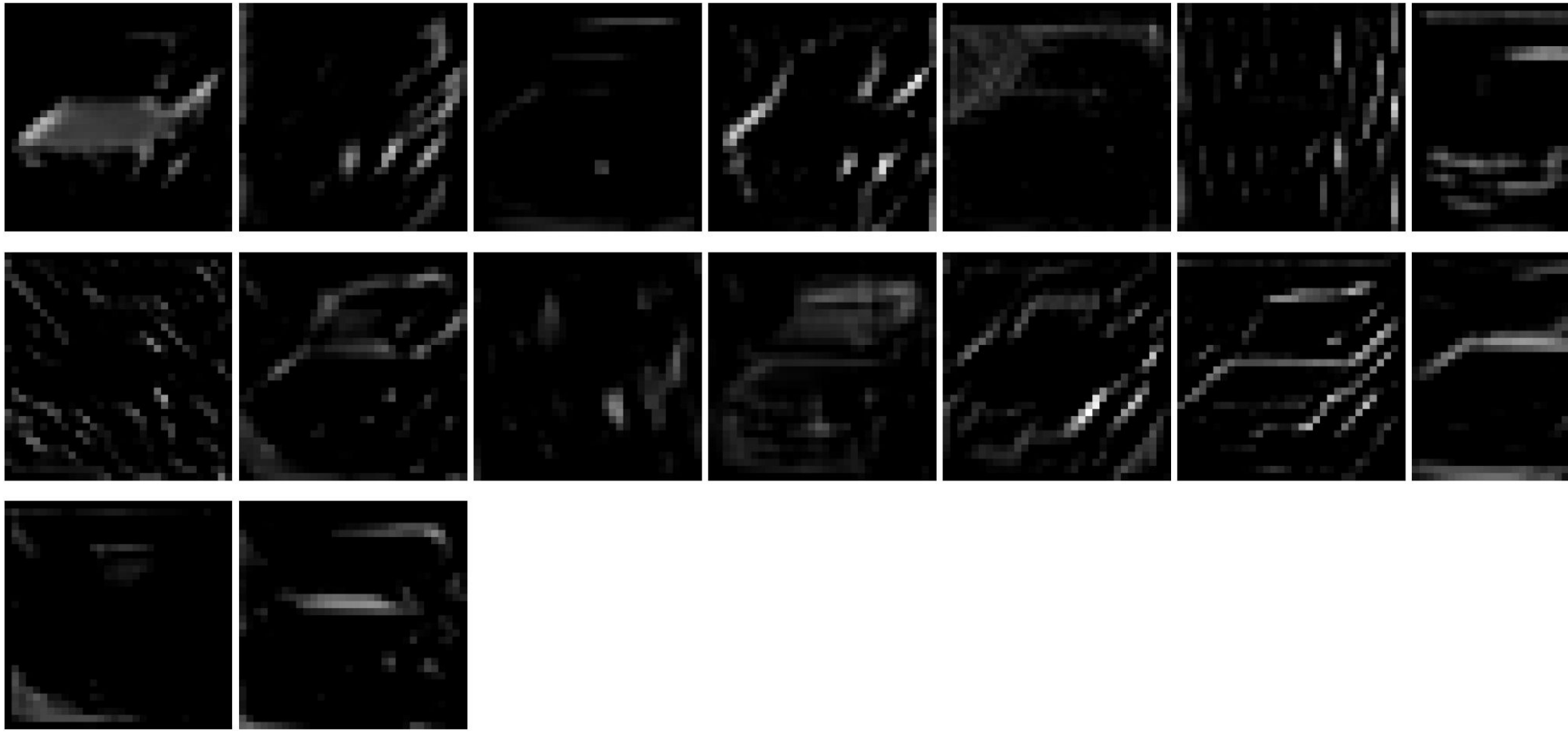


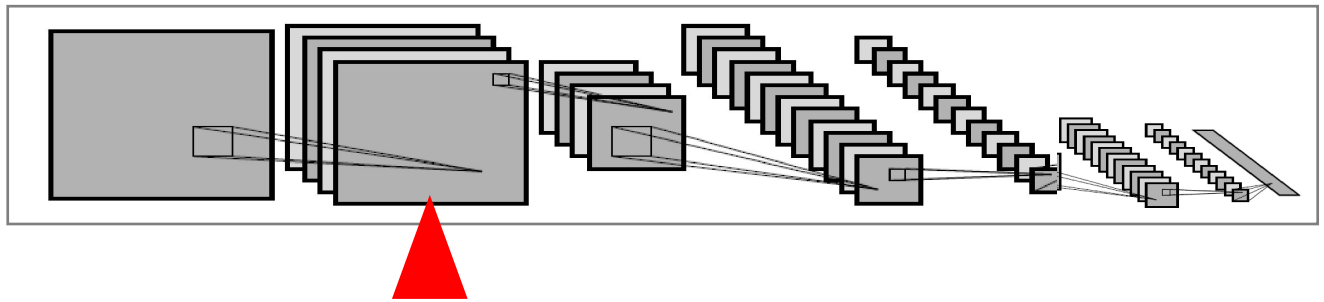
filter size 5x5x3, stride 1

input (32x32x3)

RELU

conv (32x32x16) params: 16x5x5x3+16 = 1216

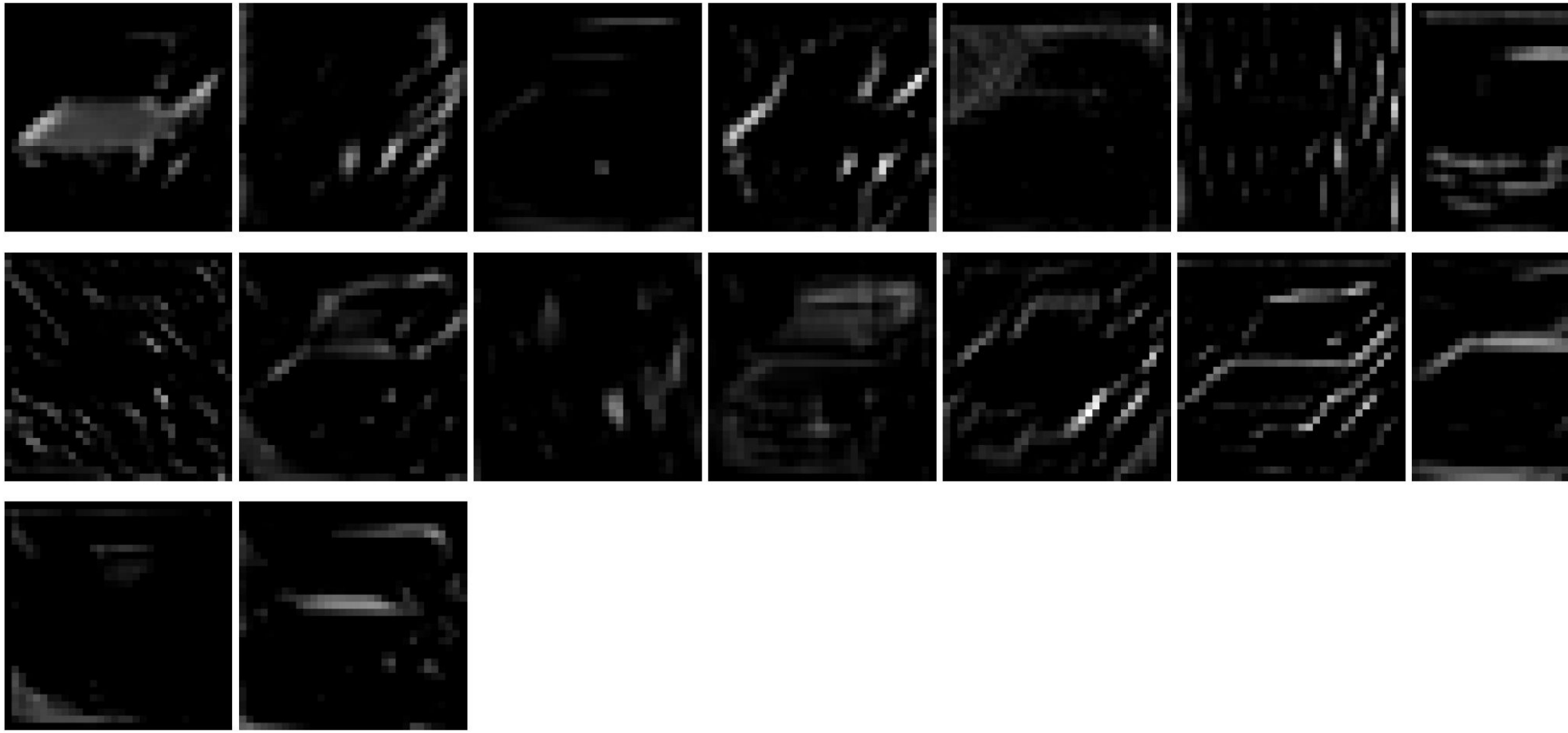


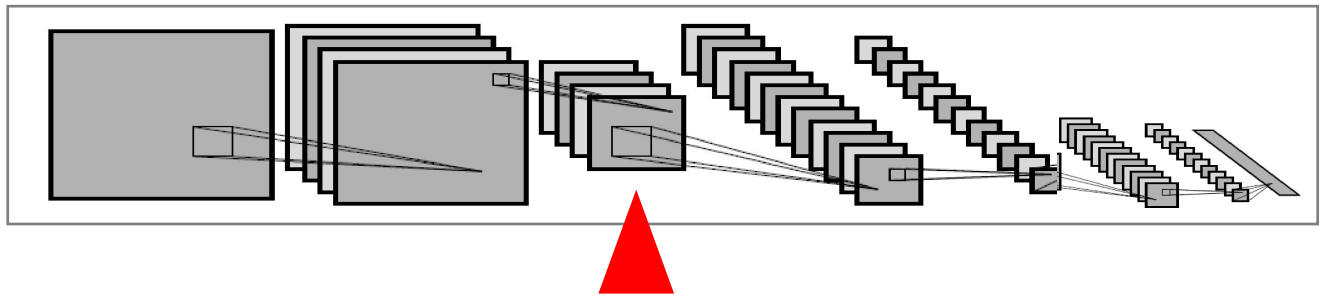


filter size 5x5x3, stride 1

input (32x32x3)

conv (32x32x16) params: 16x5x5x3+16 = 1216



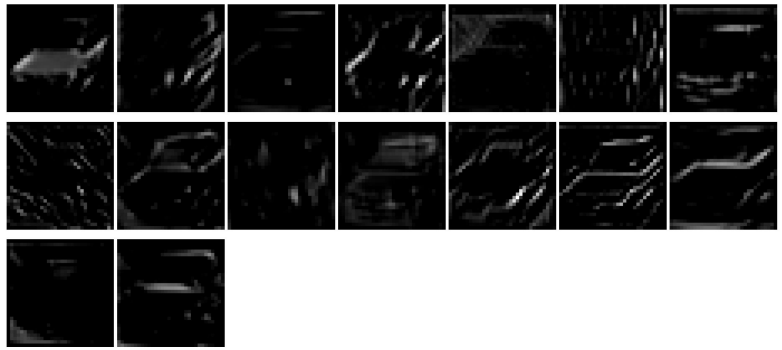


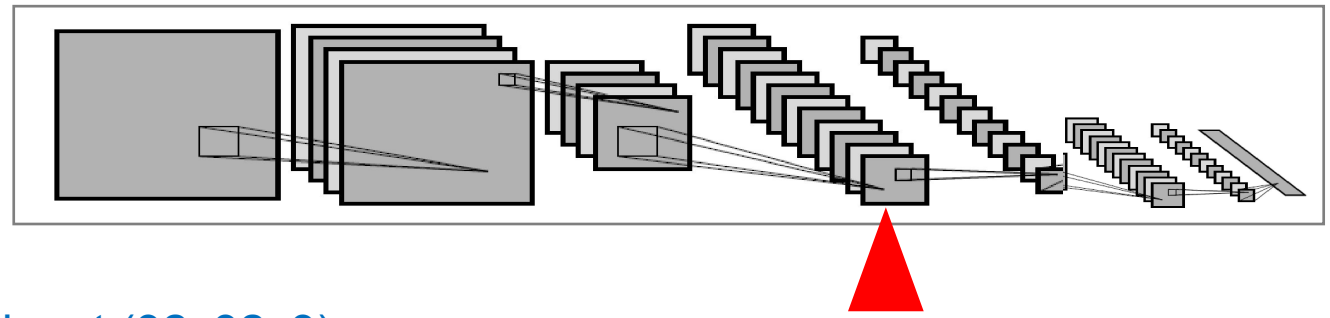
filter size 5x5x3, stride 1

input (32x32x3)

pool (16x16x16)
pooling size 2x2, stride 2

conv (32x32x16) params: 16x5x5x3+16 = 1216



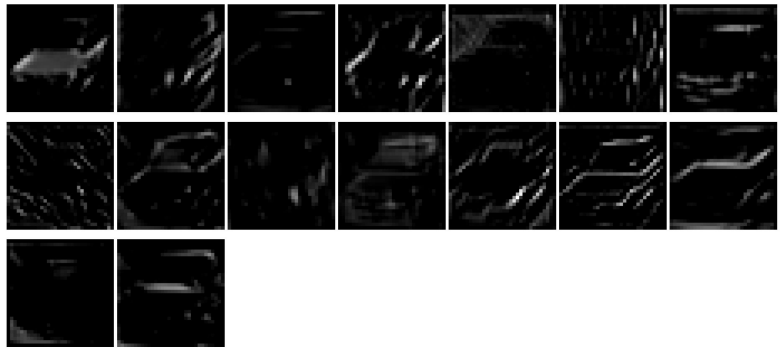


filter size 5x5x3, stride 1

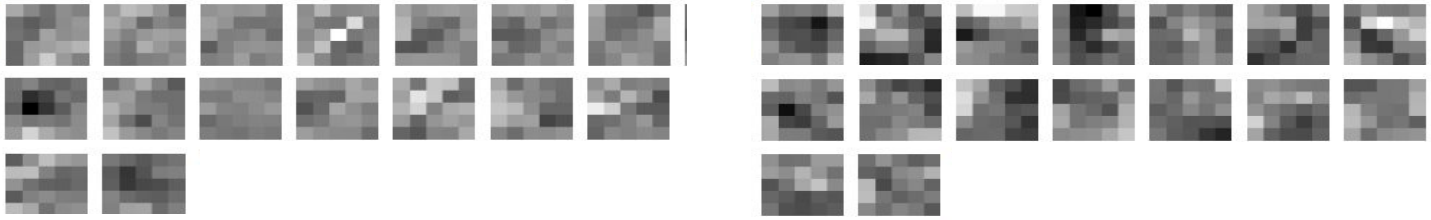
input (32x32x3)

pool (16x16x16)
pooling size 2x2, stride 2

conv (32x32x16) params: 16x5x5x3+16 = 1216

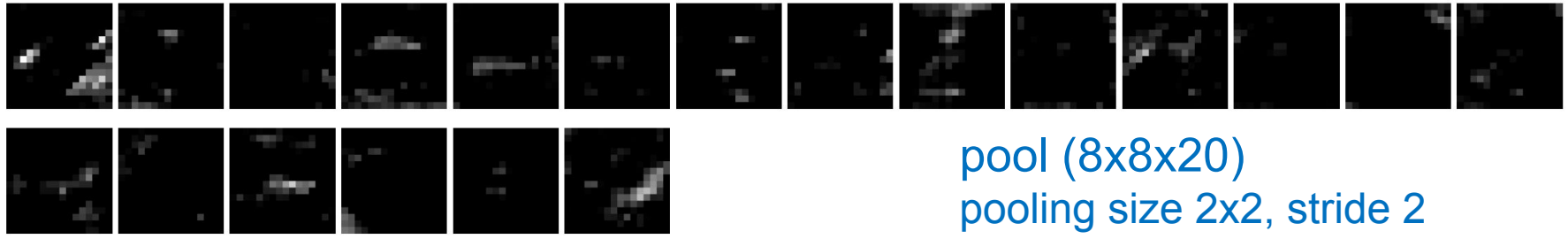


filter size 5x5x16, stride 1

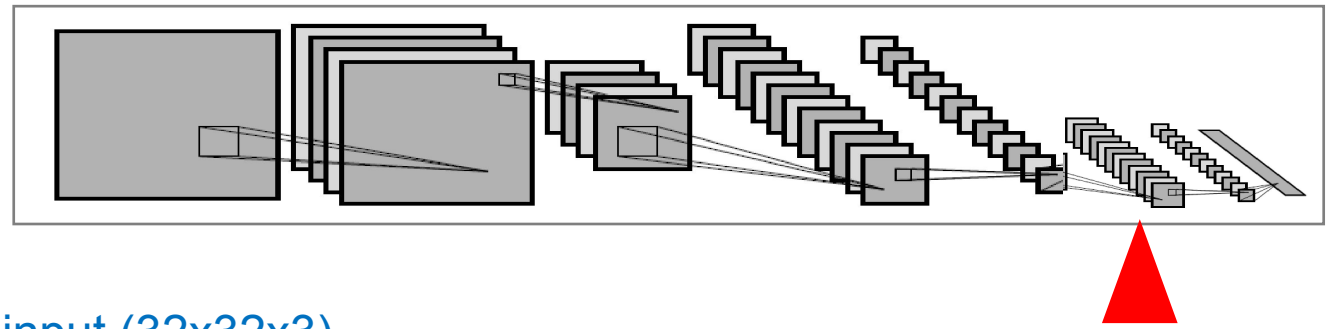


RELU

conv (16x16x20) params: 20x5x5x16+20 = 8020



pool (8x8x20)
pooling size 2x2, stride 2



input (32x32x3)

One more conv+RELU+pool:

conv (8x8x20)
filter size 5x5x20, stride 1
relu (8x8x20)
pool (4x4x20)
pooling size 2x2, stride 2
parameters: 20x5x5x20+20 = 10020

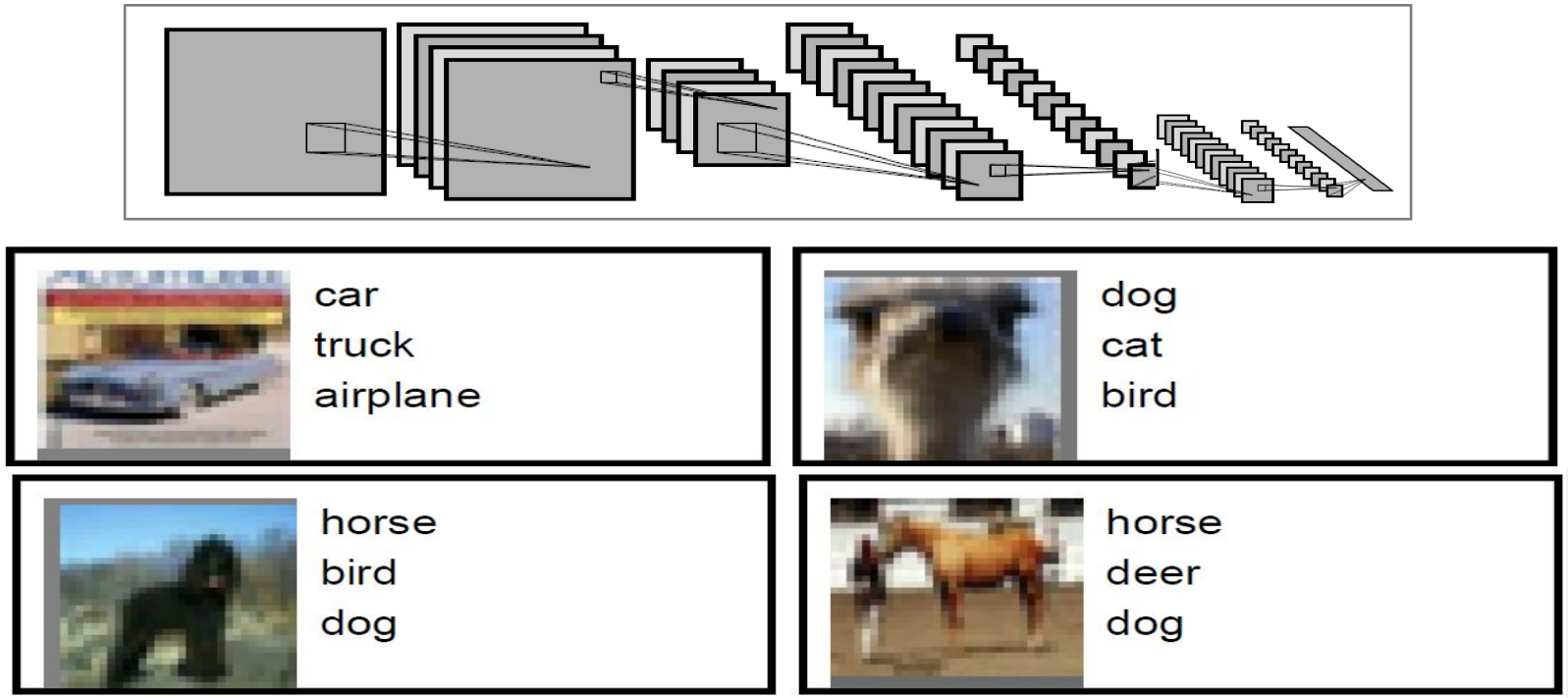
fc (1x1x10); parameters: 10x320+10 = 3210

softmax (1x1x10)

dog car cat ...

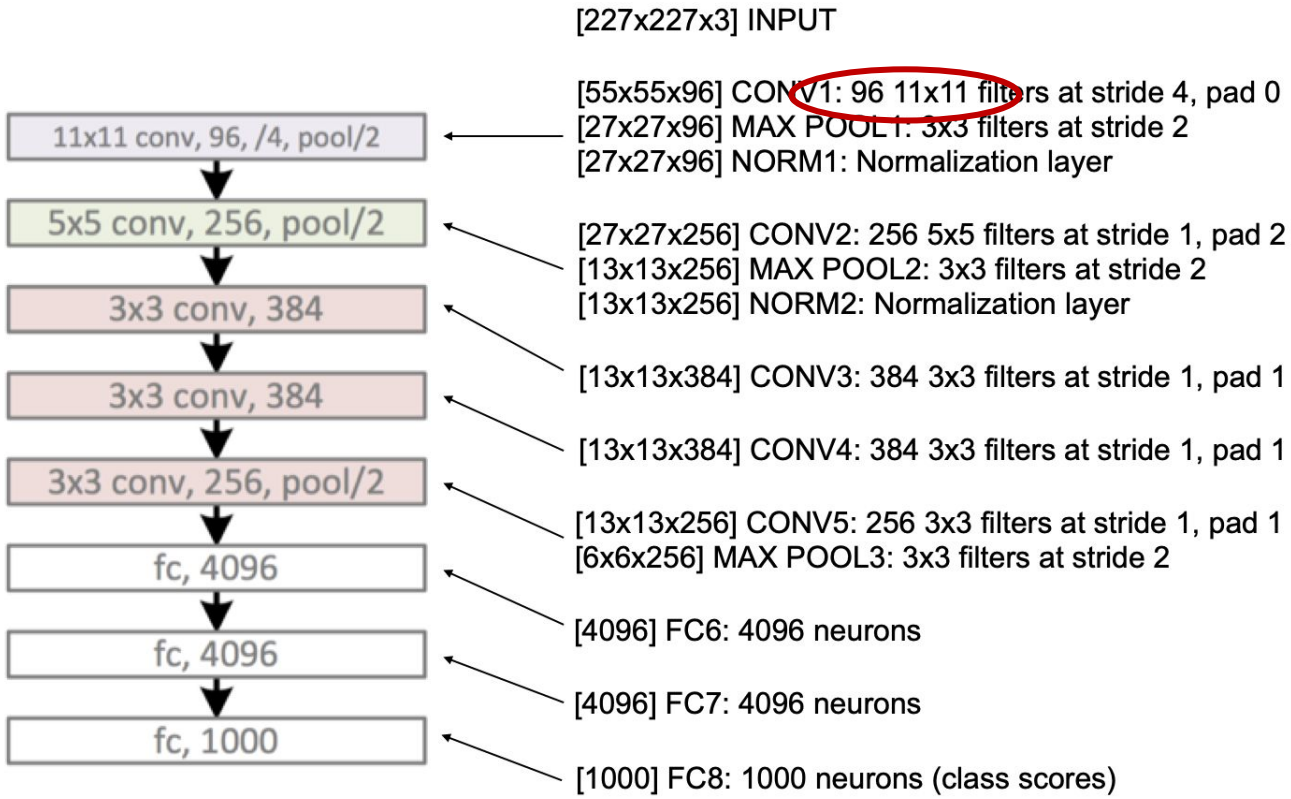
Testing the network

- Show top three most likely classes



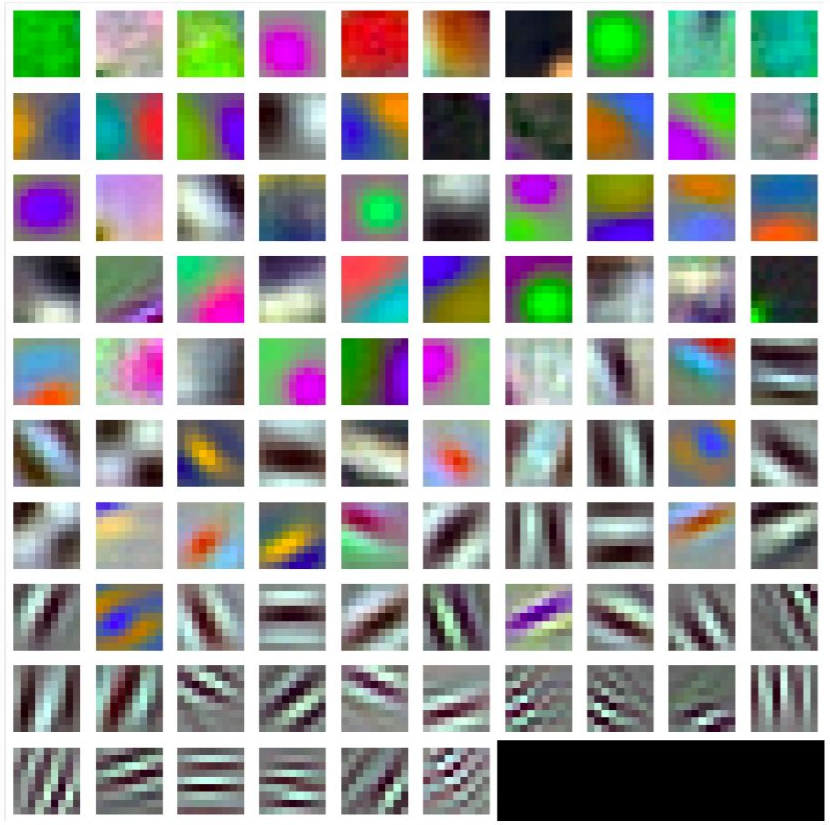
http://cs.stanford.edu/people/karpathy/convnetjs/demo/cifar10.html

Alexnet — [Krizhevsky et al. NIPS 2012]

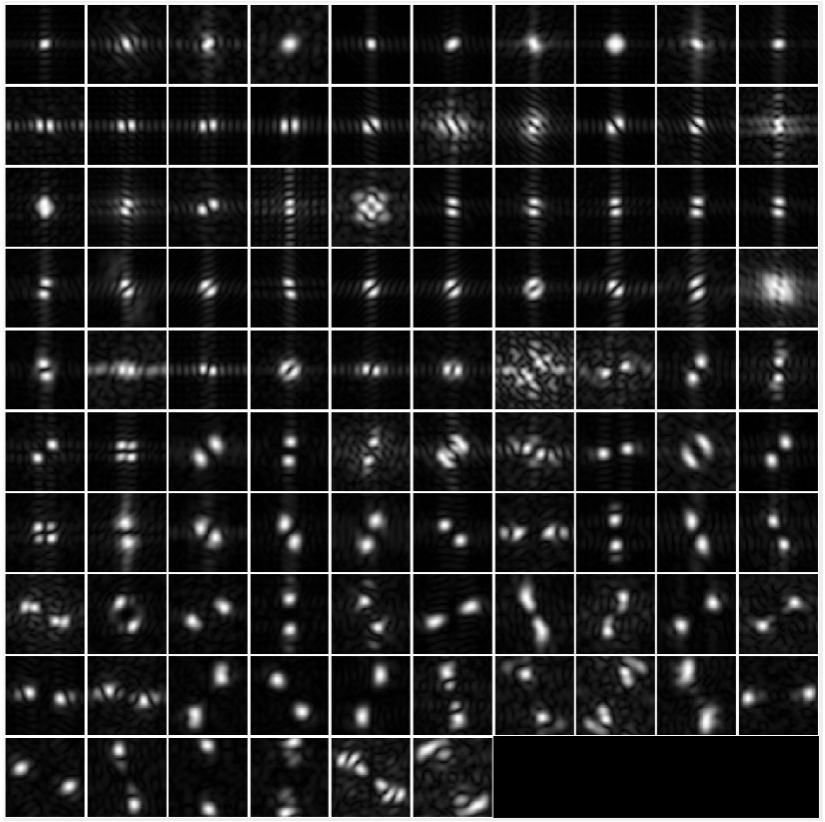


- AlexNet – Winner of the ImageNet challenge for classification (2012)

Get to know your units

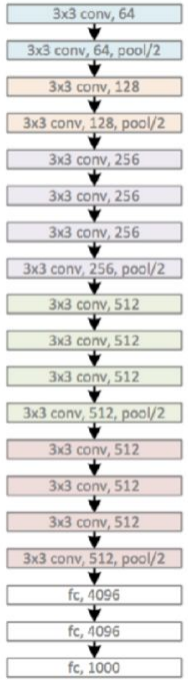


96 Units in conv1



VGG-Net [Simonyan & Zisserman, 2015]

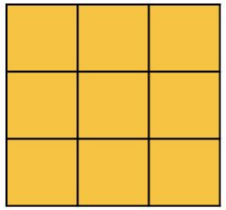
2014: VGG
16 conv. layers



Error: 7.3%

Main developments

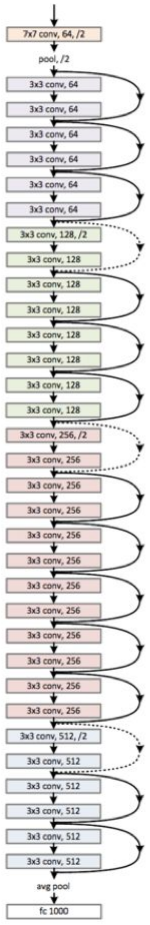
- Small convolutional kernels: only 3x3



- Increased depth (5 -> 16/19 layers)

2016: ResNet
>100 conv. layers

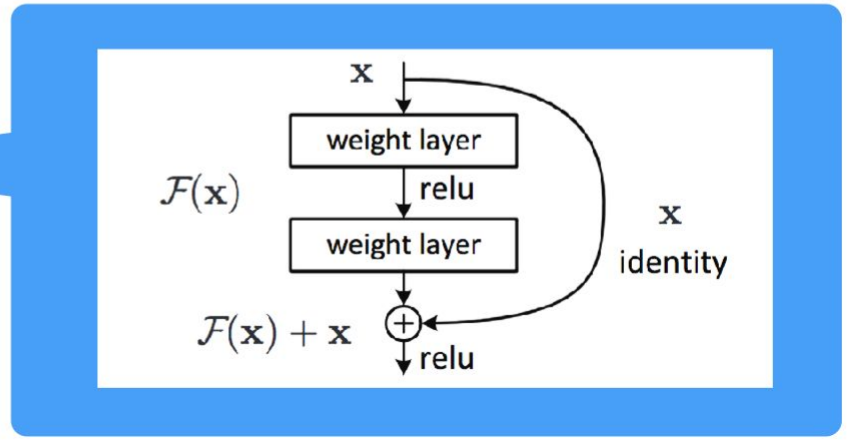
ResNet [He et al, 2016]



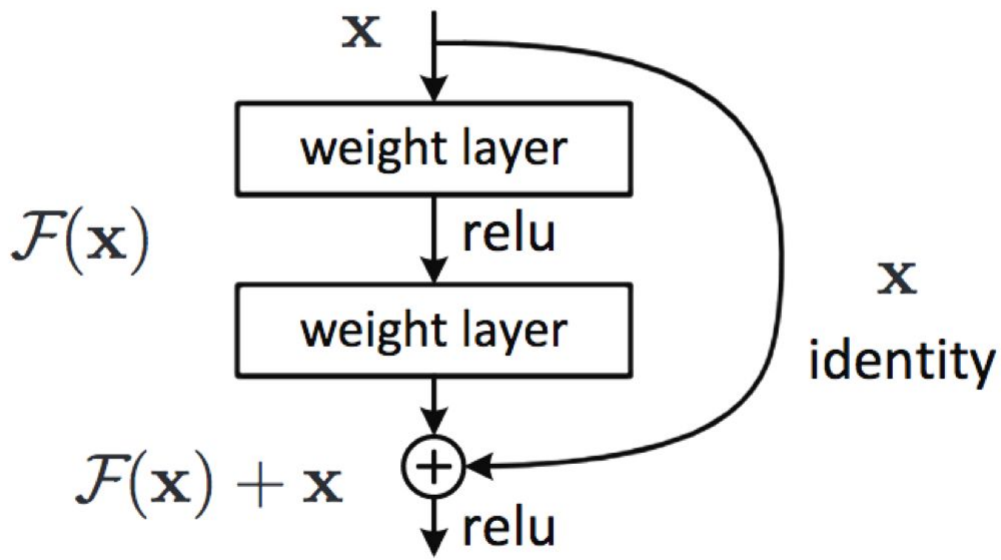
Error: 3.6%

Main developments

- Increased depth possible through residual blocks

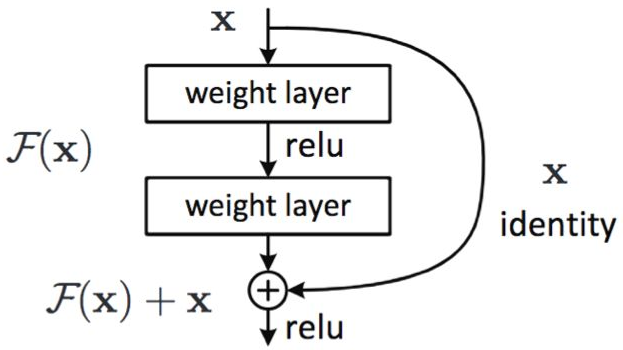


Residual Blocks



Why is residual block helpful?

Residual Blocks



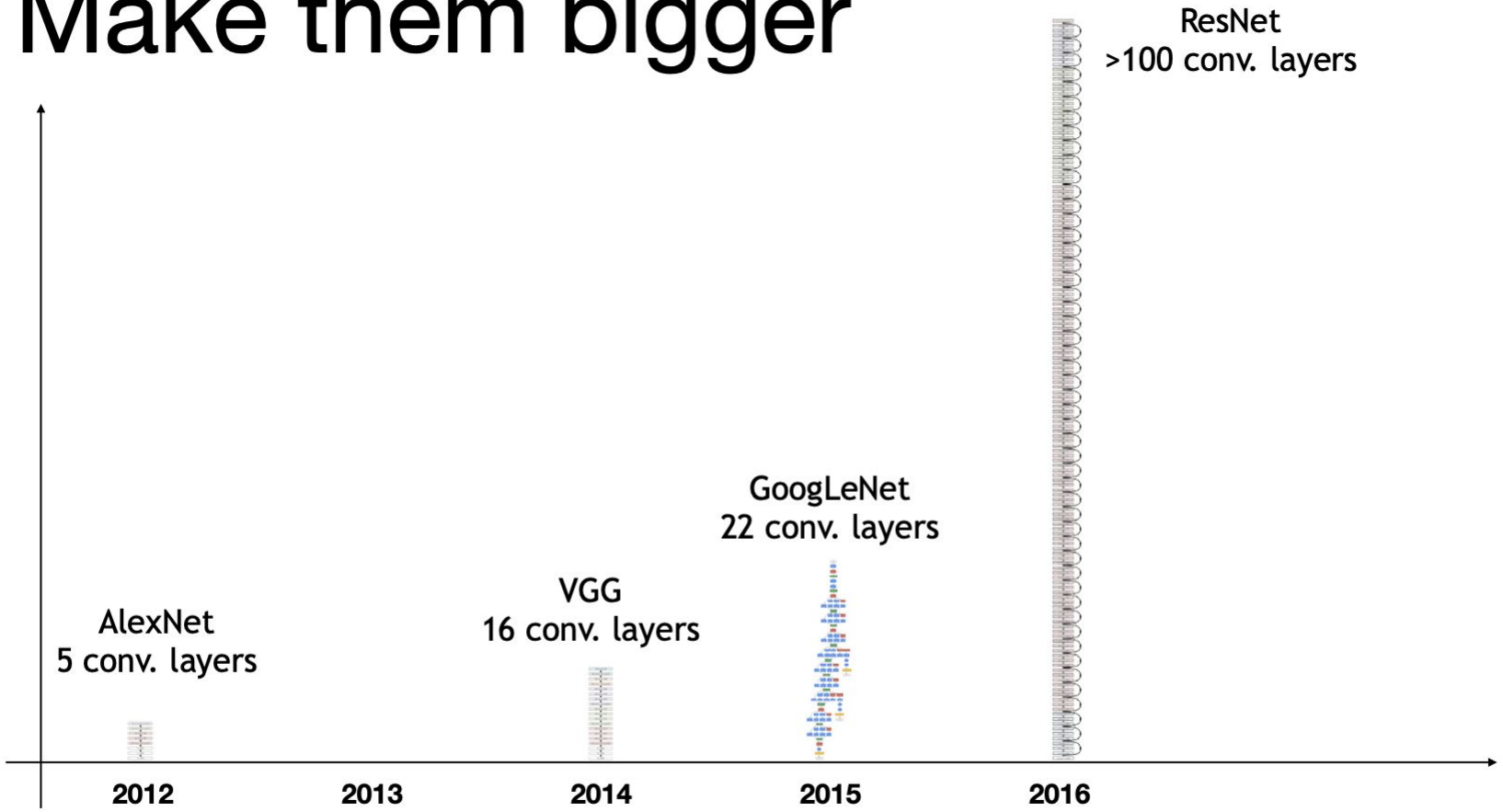
Why do they work?

- Gradients can propagate faster (via the identity mapping)

The direct path allows gradients to flow more easily from the output of the network back to the earlier layers, preventing them from vanishing.

Enable the training of much deeper and more complex neural networks

Make them bigger

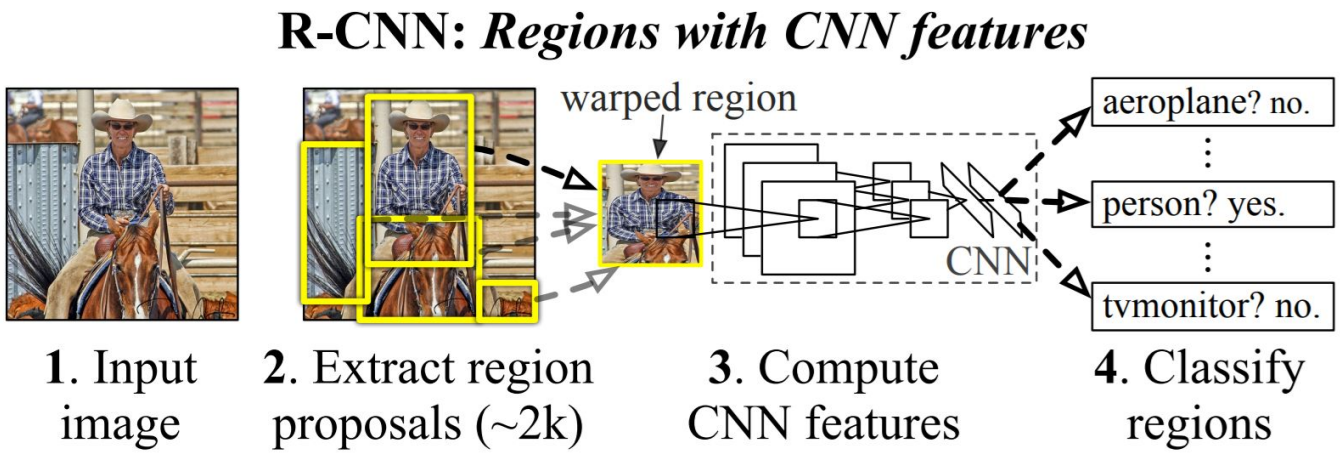


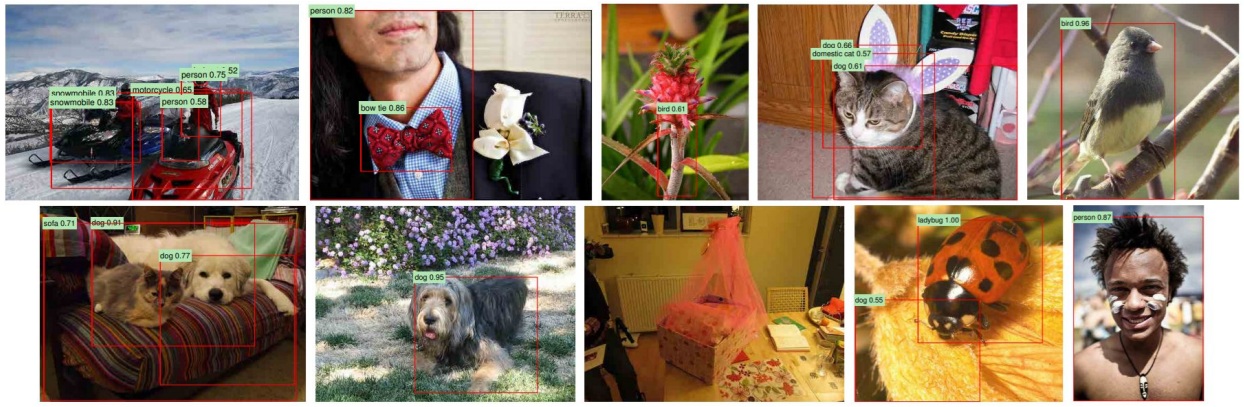
Meet the ResNet family

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$

CNNs for Detection

- R-CNN (ImageNet Large Scale Visual Recognition Challenge 2013 (ILSVRC2013)





From 2D -> 3D: CNN for videos

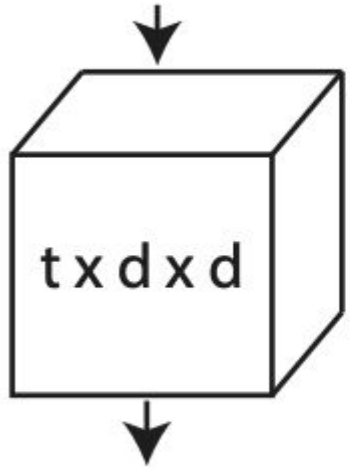
(a) R2D

From 2D -> 3D: CNN for videos

(a) R2D

3D convolution

c X t X d X d

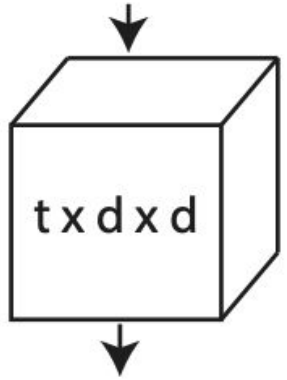


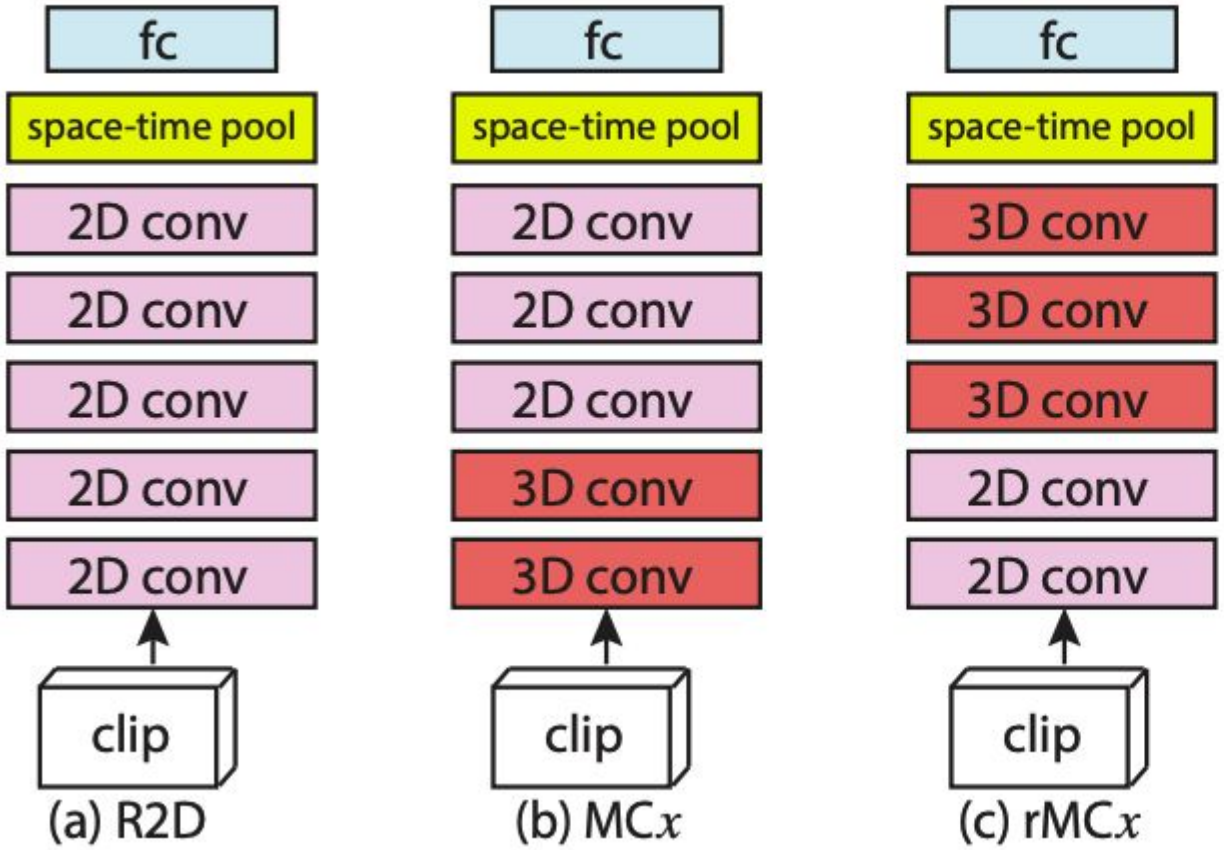
2D convolution

c X d X d

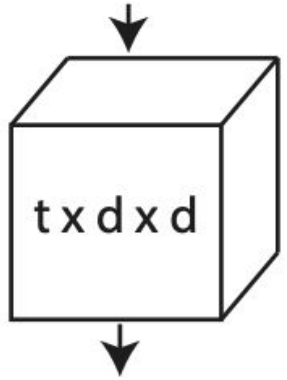
c: channels
t: time
d: dimension of the 2D conv filter.

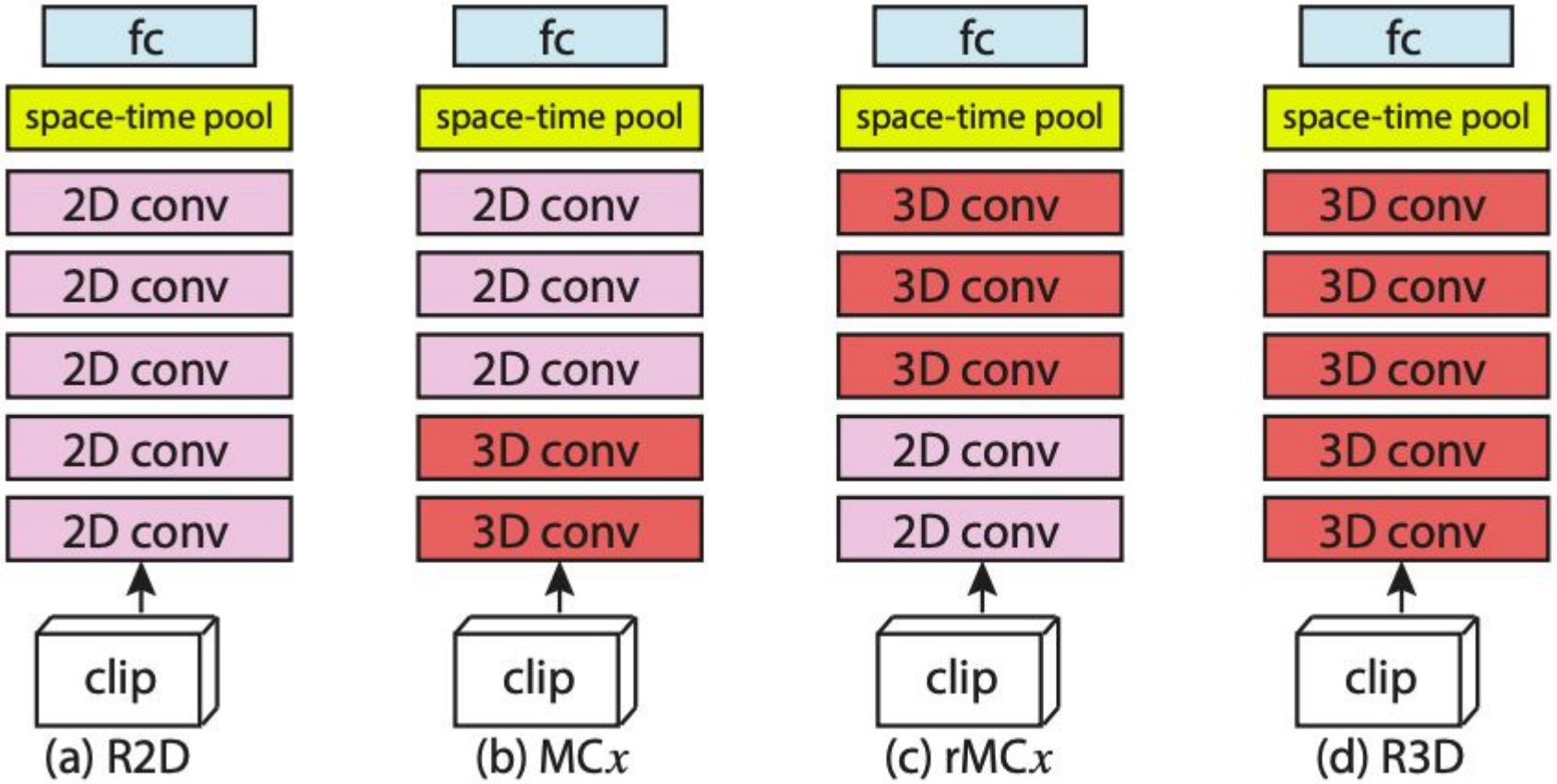
From 2D -> 3D: CNN for videos



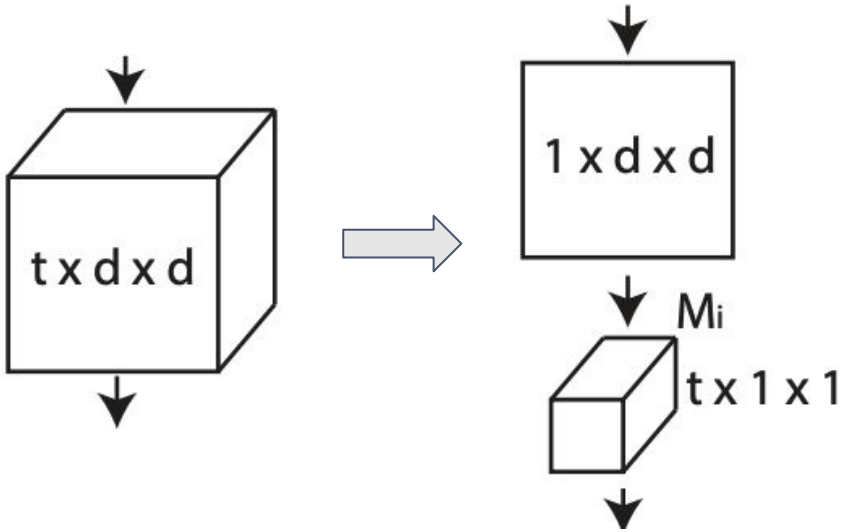


From 2D -> 3D: CNN for videos





From 2D -> 3D -> **(2+1)D**



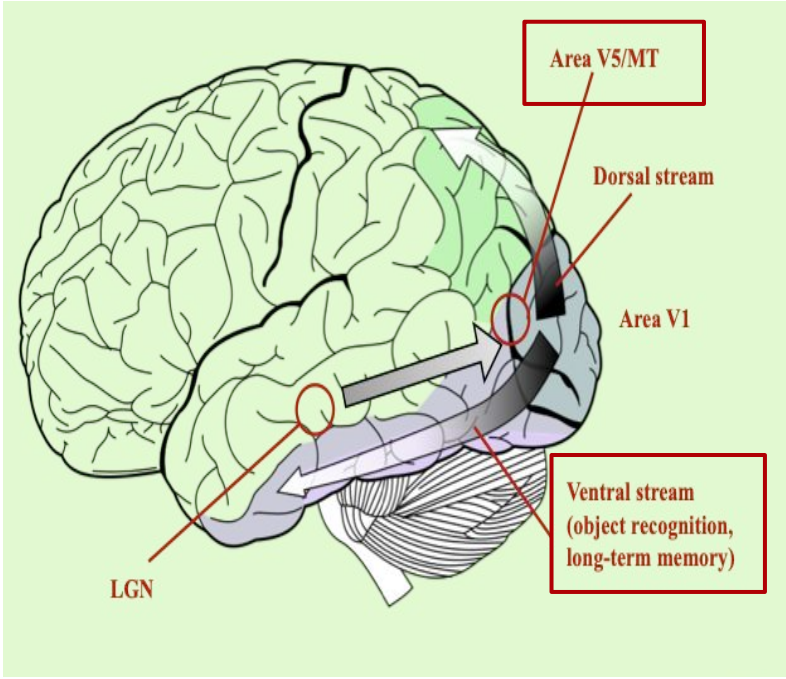
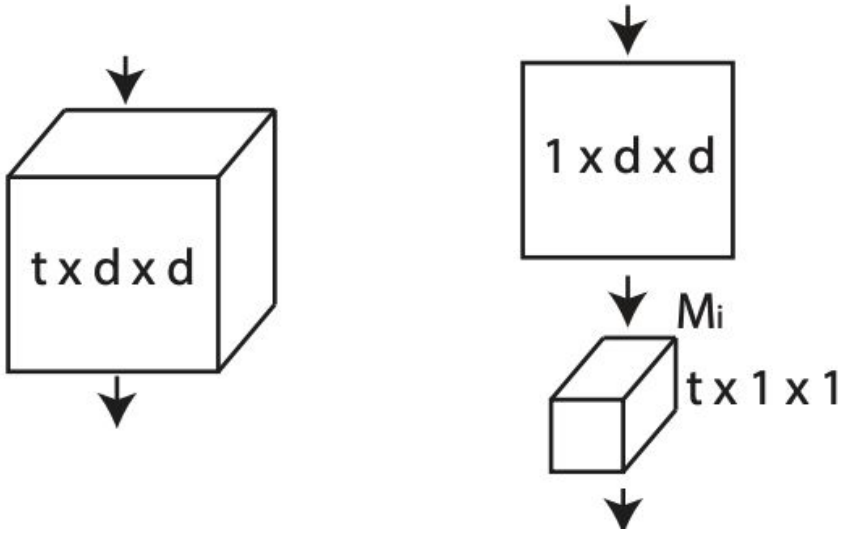


Figure from: Al Bovik

Compute the number of arithmetic operations involved in 3D and (2+1)D



- **2 mins** ⏰
  - Enter in slido in the next slide.

How many computational operations are done in 3D v/s (2+1)D?

How many computational operations are done in 3D v/s (2+1)D?

3D: c+t*d*d | (2+1)D: c+t+d*d

20%

3D: c*d*d+t | (2+1)D: t+c+d*d

14%

3D: c*t*d*d | (2+1)D: t+c*d*d ✓

67%

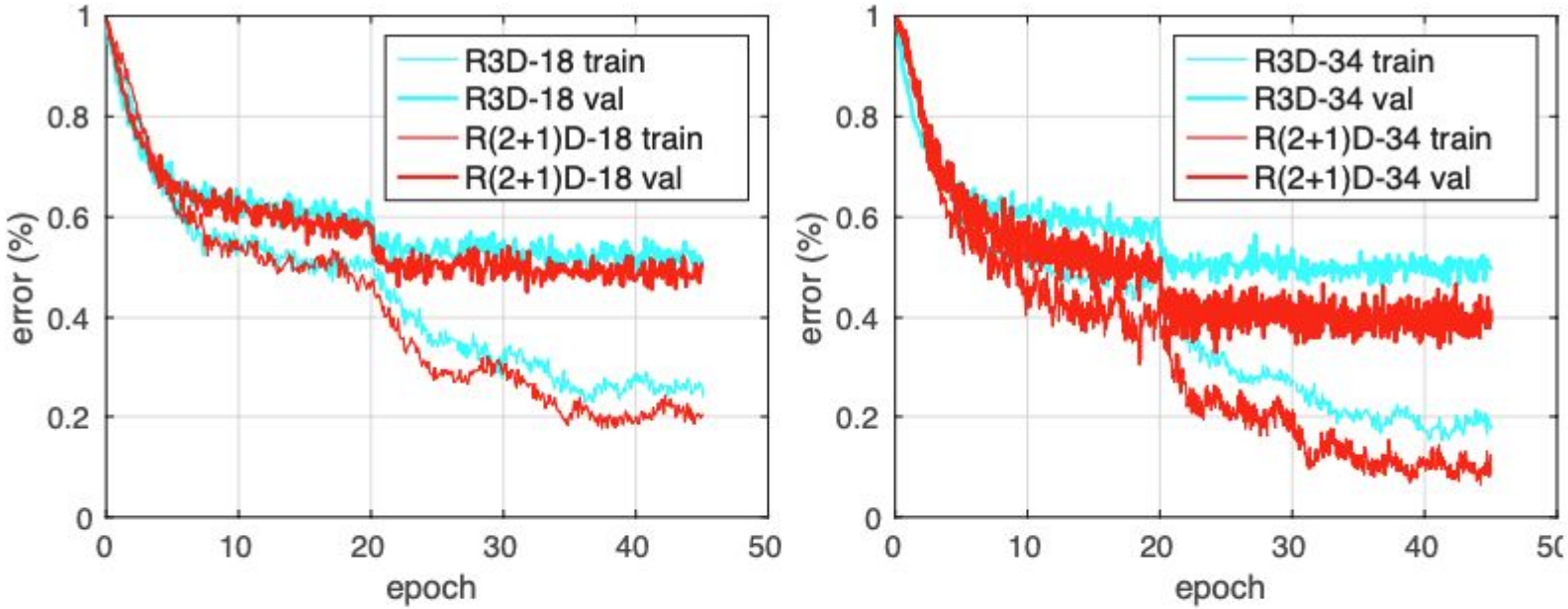


Figure 3. **Training and testing errors for R(2+1)D and R3D.** Results are reported for ResNets of 18 layers (left) and 34 layers (right). It can be observed that the training error (thin lines) is smaller for R(2+1)D compared to R3D, particularly for the network with larger depth (right). This suggests that the the spatial-temporal decomposition implemented by R(2+1)D eases the optimization, especially as depth is increased.