- Homework 3 due Friday, February 25th
- Start early!!!

Lecture 20: Convolutional Neural Network

- How to make the neural networks compact with smaller number of parameters for computer vision tasks

Multi-layer Neural Network

$$a^{(1)} = x$$

$$z^{(2)} = \Theta^{(1)} a^{(1)}$$

$$a^{(2)} = g\left(z^{(2)}\right)$$

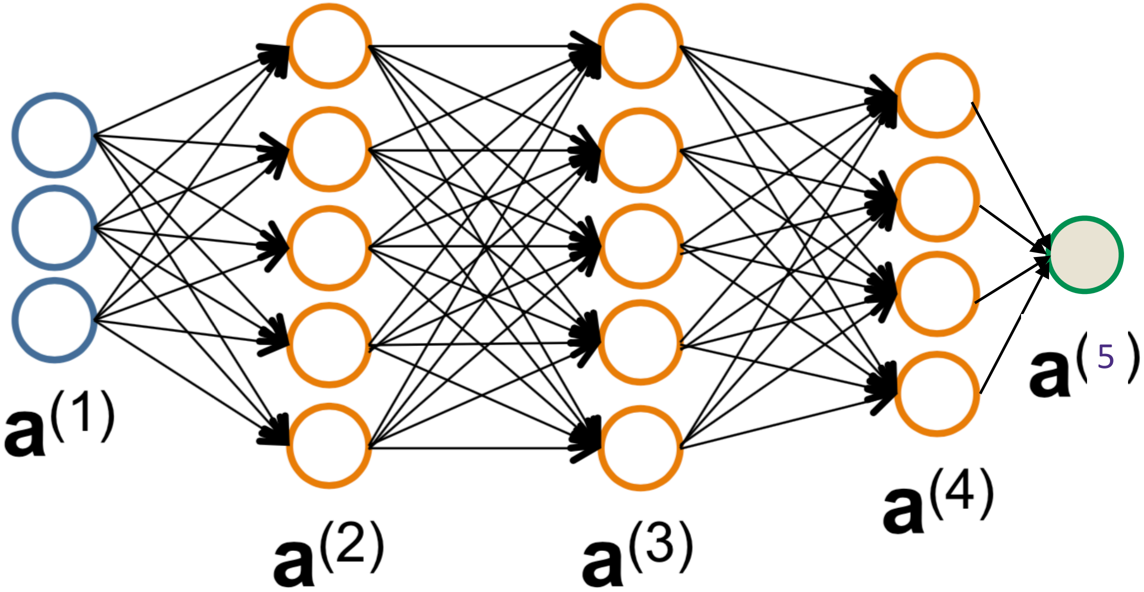
$$\vdots$$

$$z^{(l+1)} = \Theta^{(l)} a^{(l)}$$

$$a^{(l+1)} = g\left(z^{(l+1)}\right)$$

$$\vdots$$

$$\hat{y} = a^{(L+1)}$$



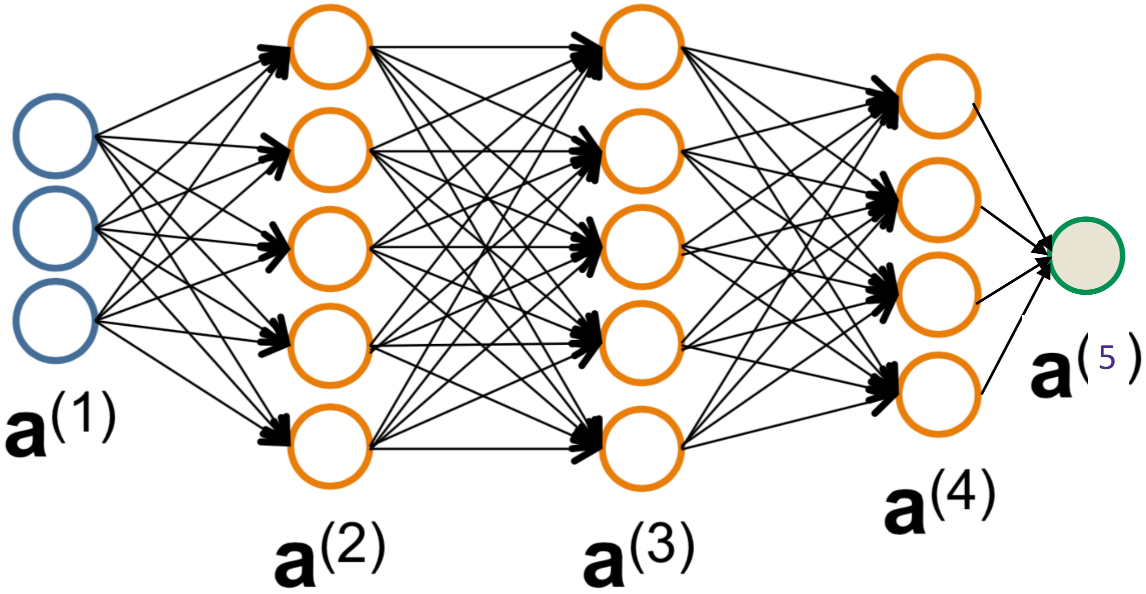
$$L(y, \hat{y}) = y \log(\hat{y}) + (1 - y)\log(1 - \hat{y})$$

$$g(z) = \frac{1}{1 + e^{-z}}$$

Binary
Logistic
Regression

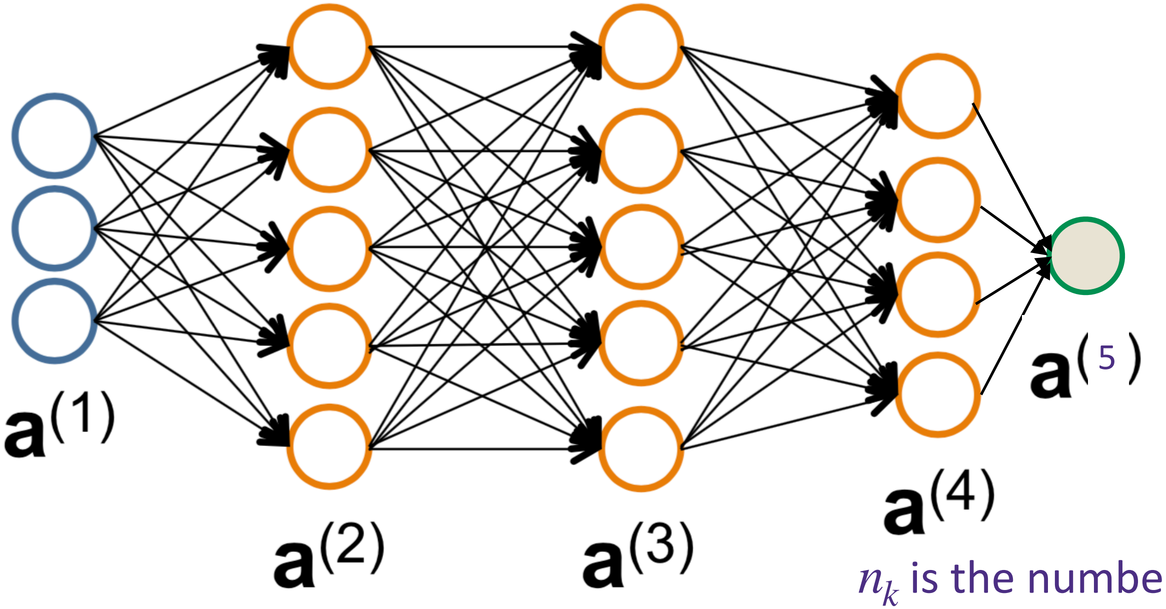
Neural Network Architecture

- The neural network architecture is defined by
  - the number of layers (depth of a network),
  - the number of nodes in each layer (width of a layer),
  - and also by **allowable edges** and **shared weights**.



Neural Network Architecture

The neural network architecture is defined by the number of layers, and the number of nodes in each layer, and also by **allowable edges** and **shared weights**.



n_k is the number of nodes in layer k

We say a layer is **Fully Connected (FC)** if all linear mappings from the current layer to the next layer are permissible.

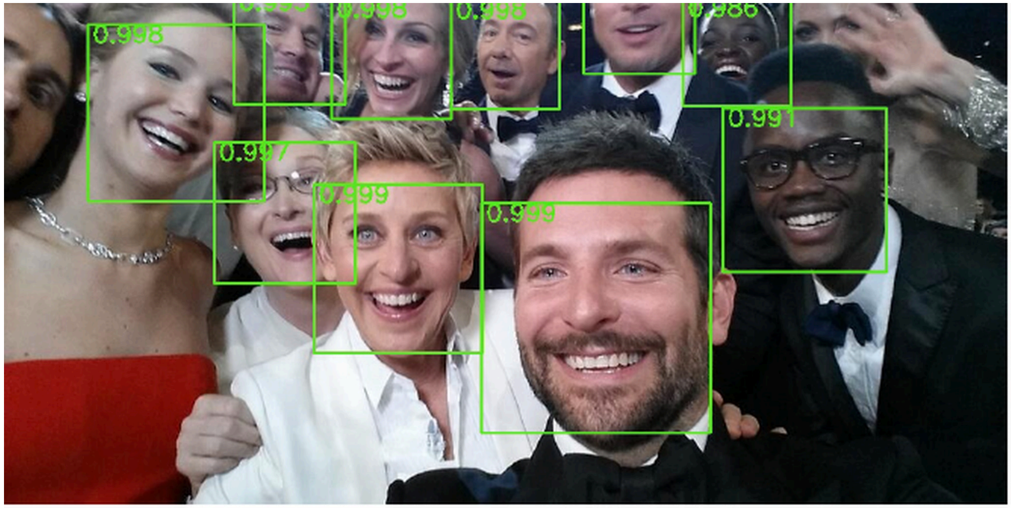
$$\mathbf{a}^{(k+1)} = g(\Theta \mathbf{a}^{(k)}) \quad \text{for any } \Theta \in \mathbb{R}^{n_{k+1} \times n_k}$$

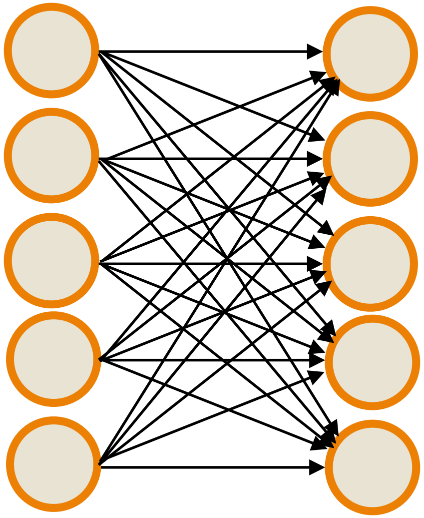
A lot of parameters!! $n_1 n_2 + n_2 n_3 + \cdots + n_L n_{L+1}$

Neural Network Architecture

- Objects in an image are often **localized in space** so to find the faces in an image, not every pixel is important for classification
- Makes sense to drag a window across an image, focusing a local region at a time
- Although images are two-dimensional, we use one-dimensional examples to illustrate the main idea
    - Similarly, to identify edges or other local structure, it makes sense to only look at **local information**

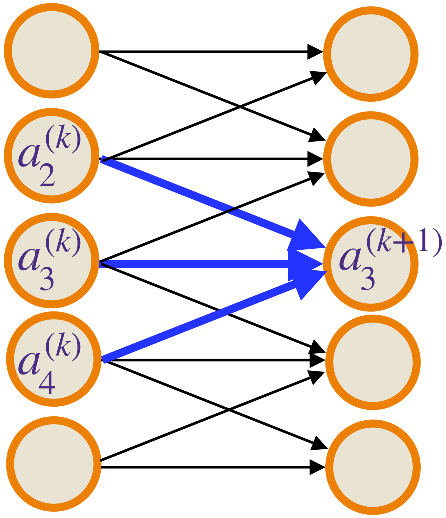
Finding faces require only local patterns





This is a fully connected layer

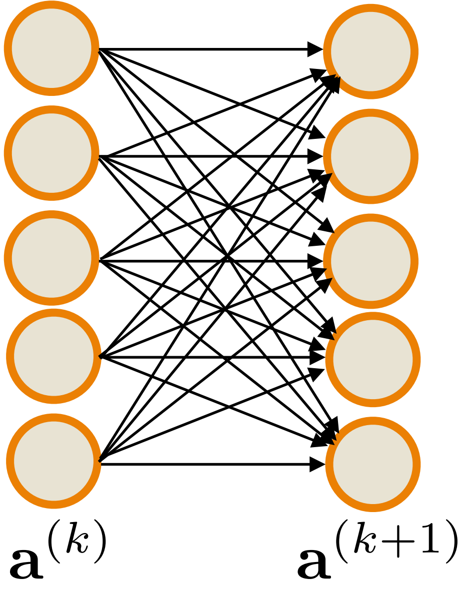
vs.



This has sparse and local connections

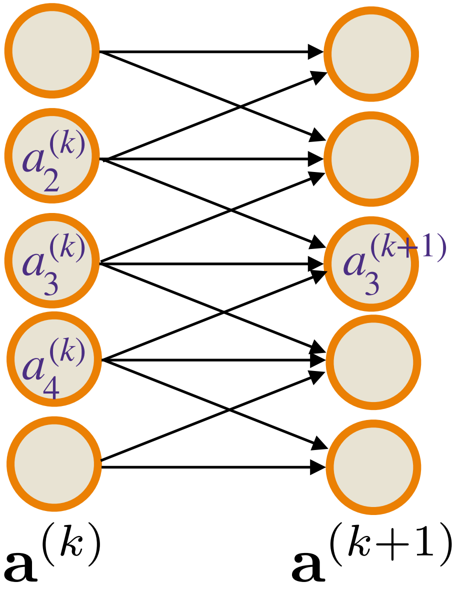
Neural Network Architecture

Fully connected



vs.

sparse and local connections



$$\Theta^{(k)} = \begin{bmatrix} \Theta_{0,0} & \Theta_{0,1} & \Theta_{0,2} & \Theta_{0,3} & \Theta_{0,4} \\ \Theta_{1,0} & \Theta_{1,1} & \Theta_{1,2} & \Theta_{1,3} & \Theta_{1,4} \\ \Theta_{2,0} & \Theta_{2,1} & \Theta_{2,2} & \Theta_{2,3} & \Theta_{2,4} \\ \Theta_{3,0} & \Theta_{3,1} & \Theta_{3,2} & \Theta_{3,3} & \Theta_{3,4} \\ \Theta_{4,0} & \Theta_{4,1} & \Theta_{4,2} & \Theta_{4,3} & \Theta_{4,4} \end{bmatrix}$$

$$\Theta^{(k)} = \begin{bmatrix} \Theta_{0,0} & \Theta_{0,1} & 0 & 0 & 0 \\ \Theta_{1,0} & \Theta_{1,1} & \Theta_{1,2} & 0 & 0 \\ 0 & \Theta_{2,1} & \Theta_{2,2} & \Theta_{2,3} & 0 \\ 0 & 0 & \Theta_{3,2} & \Theta_{3,3} & \Theta_{3,4} \\ 0 & 0 & 0 & \Theta_{4,3} & \Theta_{4,4} \end{bmatrix}$$

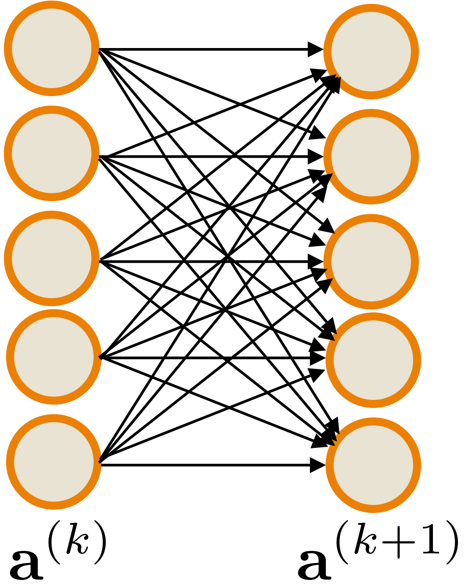
\# of Parameters in this layer: n^2 $3n - 2$

$$\mathbf{a}_i^{(k+1)} = g\left(\sum_{j=0}^{n-1} \Theta_{i,j} \mathbf{a}_j^{(k)}\right)$$

Neural Network Architecture

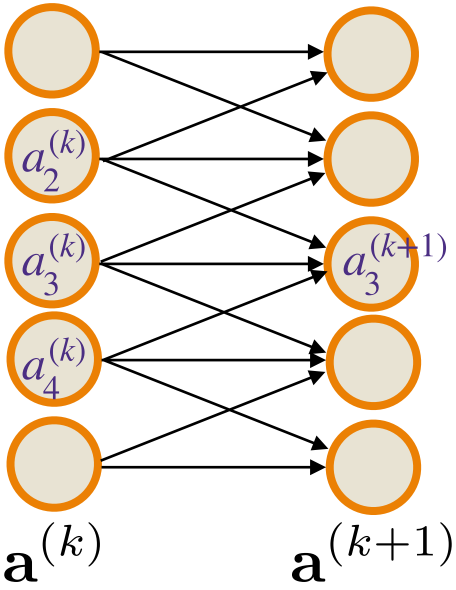
Shift invariance: A local pattern of interest can appear anywhere in the image

Fully connected



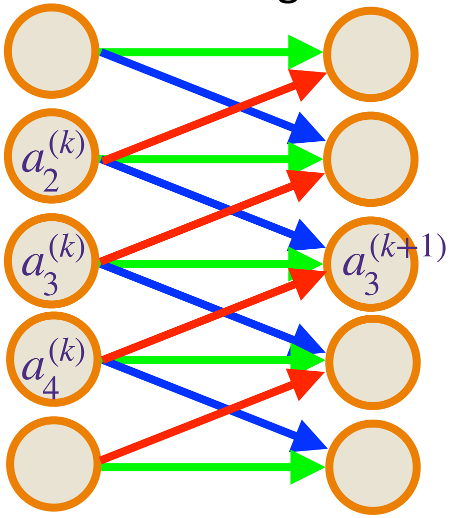
vs.

sparse and local connections



vs.

sparse local connections and shared weights



$$\Theta^{(k)} = \begin{bmatrix} \Theta_{0,0} & \Theta_{0,1} & \Theta_{0,2} & \Theta_{0,3} & \Theta_{0,4} \\ \Theta_{1,0} & \Theta_{1,1} & \Theta_{1,2} & \Theta_{1,3} & \Theta_{1,4} \\ \Theta_{2,0} & \Theta_{2,1} & \Theta_{2,2} & \Theta_{2,3} & \Theta_{2,4} \\ \Theta_{3,0} & \Theta_{3,1} & \Theta_{3,2} & \Theta_{3,3} & \Theta_{3,4} \\ \Theta_{4,0} & \Theta_{4,1} & \Theta_{4,2} & \Theta_{4,3} & \Theta_{4,4} \end{bmatrix}$$

$$\Theta^{(k)} = \begin{bmatrix} \Theta_{0,0} & \Theta_{0,1} & 0 & 0 & 0 \\ \Theta_{1,0} & \Theta_{1,1} & \Theta_{1,2} & 0 & 0 \\ 0 & \Theta_{2,1} & \Theta_{2,2} & \Theta_{2,3} & 0 \\ 0 & 0 & \Theta_{3,2} & \Theta_{3,3} & \Theta_{3,4} \\ 0 & 0 & 0 & \Theta_{4,3} & \Theta_{4,4} \end{bmatrix}$$

$$\Theta^{(k)} = \begin{bmatrix} \theta_1 & \theta_2 & 0 & 0 & 0 \\ \theta_0 & \theta_1 & \theta_2 & 0 & 0 \\ 0 & \theta_0 & \theta_1 & \theta_2 & 0 \\ 0 & 0 & \theta_0 & \theta_1 & \theta_2 \\ 0 & 0 & 0 & \theta_0 & \theta_1 \end{bmatrix}$$

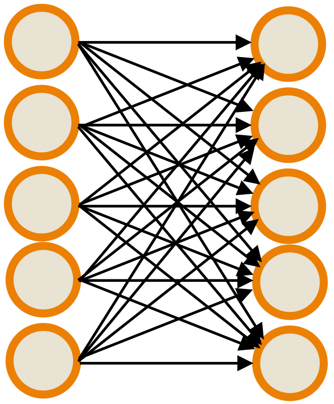
# of Parameters in this layer: n^2 $3n-2$ 3

$$\mathbf{a}_i^{(k+1)} = g\left(\sum_{j=0}^{n-1} \Theta_{i,j}\mathbf{a}_j^{(k)}\right)$$

$$\mathbf{a}_i^{(k+1)} = g\left(\sum_{j=-(m-1)/2}^{(m-1)/2} \theta_{j+(m-1)/2}\, a_{i+j}^{(k)}\right)$$

Neural Network Architecture

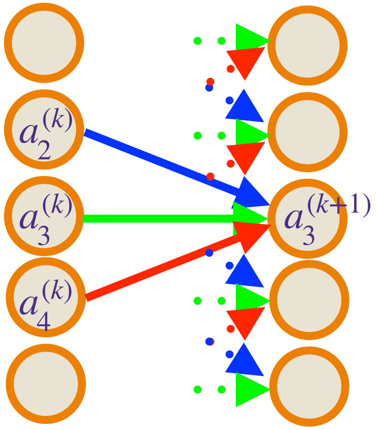
Fully Connected (FC) Layer



$$\begin{bmatrix} \Theta_{0,0} & \Theta_{0,1} & \Theta_{0,2} & \Theta_{0,3} & \Theta_{0,4} \\ \Theta_{1,0} & \Theta_{1,1} & \Theta_{1,2} & \Theta_{1,3} & \Theta_{1,4} \\ \Theta_{2,0} & \Theta_{2,1} & \Theta_{2,2} & \Theta_{2,3} & \Theta_{2,4} \\ \Theta_{3,0} & \Theta_{3,1} & \Theta_{3,2} & \Theta_{3,3} & \Theta_{3,4} \\ \Theta_{4,0} & \Theta_{4,1} & \Theta_{4,2} & \Theta_{4,3} & \Theta_{4,4} \end{bmatrix}$$

$$\mathbf{a}_i^{(k+1)} = g\left(\sum_{j=0}^{n-1} \Theta_{i,j} \mathbf{a}_j^{(k)}\right)$$

Convolutional (CONV) Layer (1 filter)



Filter with $m = 3$

$$a_3^{(k+1)} \rightarrow \begin{bmatrix} \theta_1 & \theta_2 & \overset{\overset{a_3^{(k)}}{\downarrow}}{0} & 0 & 0 \\ \theta_0 & \theta_1 & \theta_2 & 0 & 0 \\ 0 & \multicolumn{3}{c}{\boxed{\theta_0 \quad \theta_1 \quad \theta_2}} & 0 \\ 0 & 0 & \theta_0 & \theta_1 & \theta_2 \\ 0 & 0 & 0 & \theta_0 & \theta_1 \end{bmatrix}$$

$$\mathbf{a}_i^{(k+1)} = g\left(\sum_{j=-(m-1)/2}^{(m-1)/2} \theta_{j+(m-1)/2}\, a_{i+j}^{(k)}\right) = g([\theta \star \mathbf{a}^{(k)}]_i)$$

$\star$ = Convolution

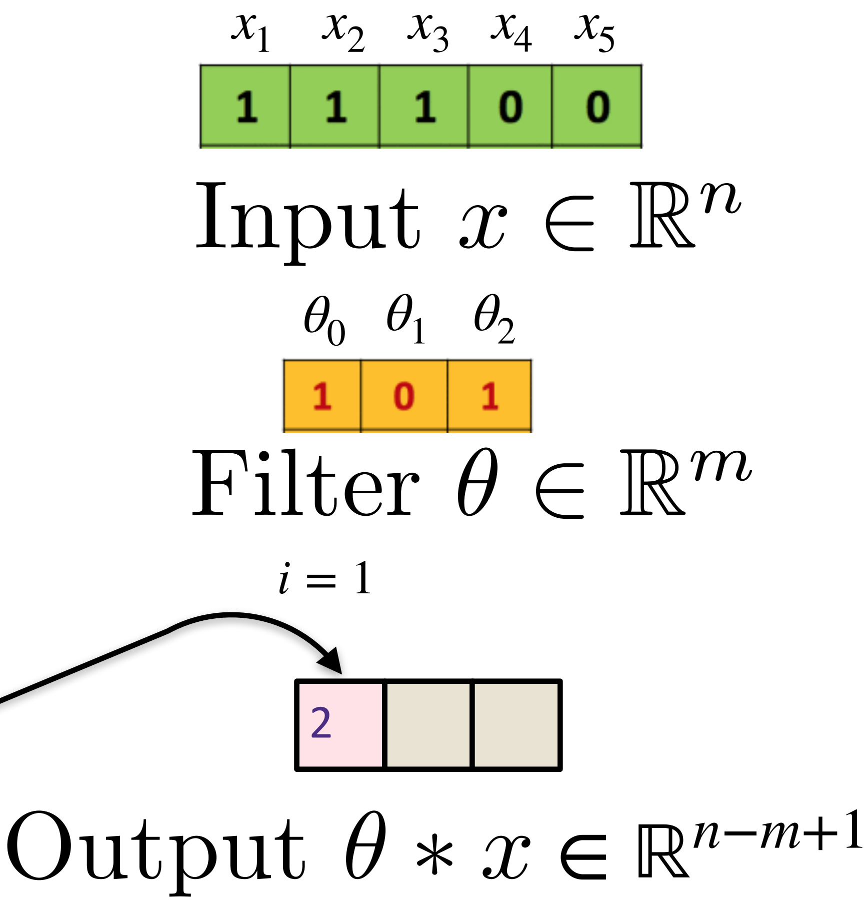
$\theta = (\theta_0, \ldots, \theta_{m-1}) \in \mathbb{R}^m$ is referred to as a "filter"

Because of shift invariance and locality of computer vision tasks, convolution is extremely powerful

Example (1d convolution)

- Notice that the indexing of the convolution is slightly different from previous slide
- There are many different ways to write the same convolution

$$(\theta * x)_i = \sum_{j=0}^{m-1} \theta_j x_{i+j}$$



x_1	x_2	x_3	x_4	x_5
1	1	1	0	0

Input $x \in \mathbb{R}^n$

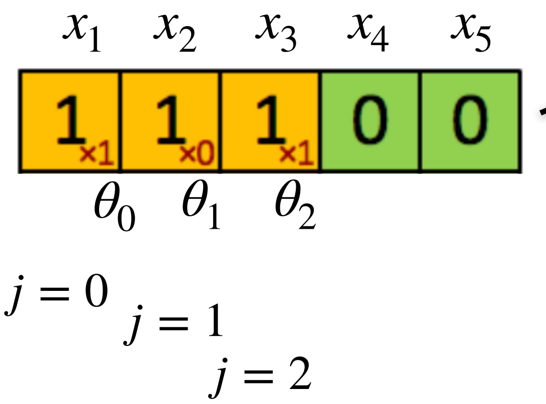
θ_0	θ_1	θ_2
1	0	1

Filter $\theta \in \mathbb{R}^m$

$i = 1$

2		

Output $\theta * x \in \mathbb{R}^{n-m+1}$



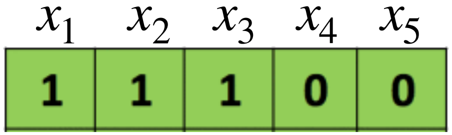
x_1	x_2	x_3	x_4	x_5
1 ×1	1 ×0	1 ×1	0	0
θ_0	θ_1	θ_2		

$j = 0$

$j = 1$

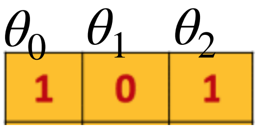
$j = 2$

Example (1d convolution)

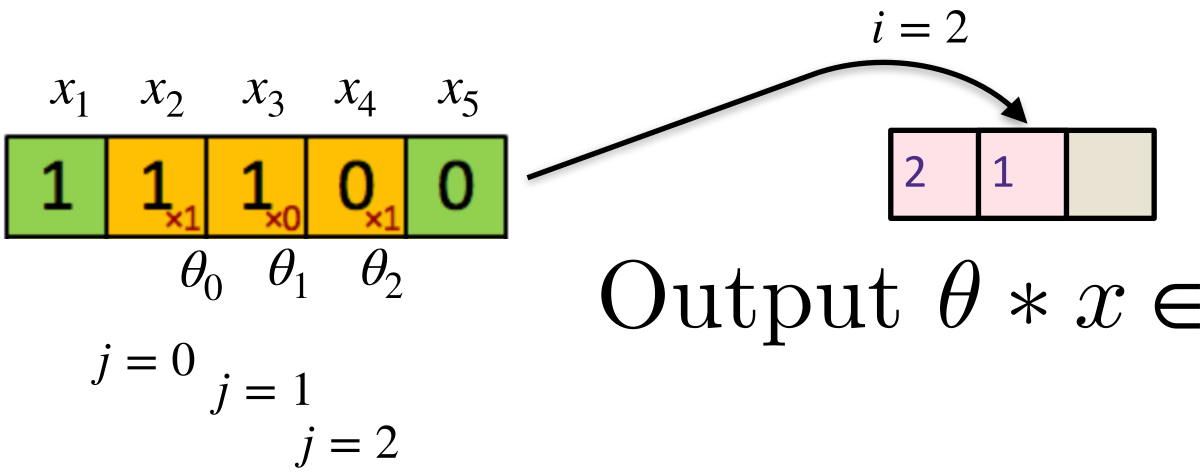


Input $x \in \mathbb{R}^n$

$$(\theta * x)_i = \sum_{j=0}^{m-1} \theta_j x_{i+j}$$



Filter $\theta \in \mathbb{R}^m$



Output $\theta * x \in \mathbb{R}^{n-m+1}$

Example (1d convolution)

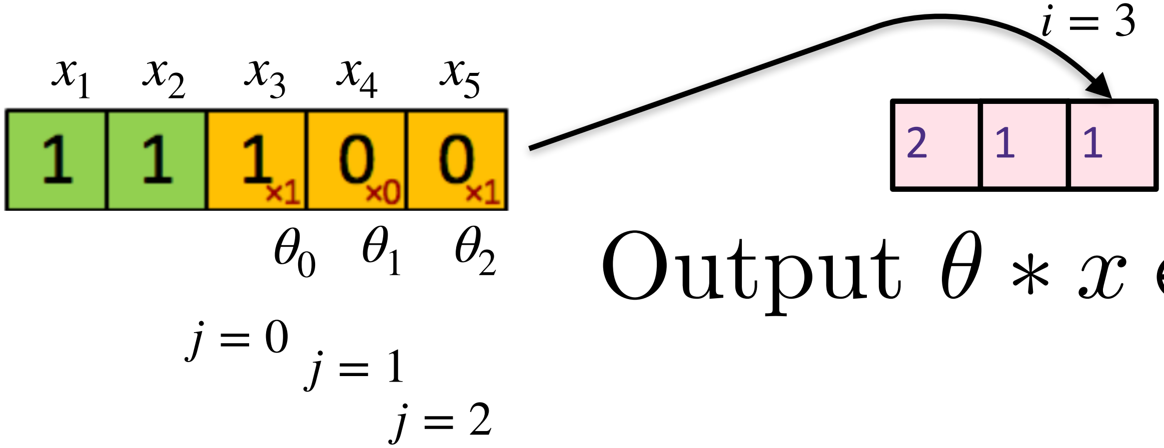
$$(\theta * x)_i = \sum_{j=0}^{m-1} \theta_j x_{i+j}$$

x_1	x_2	x_3	x_4	x_5
1	1	1	0	0

Input $x \in \mathbb{R}^n$

θ_0	θ_1	θ_2
1	0	1

Filter $\theta \in \mathbb{R}^m$

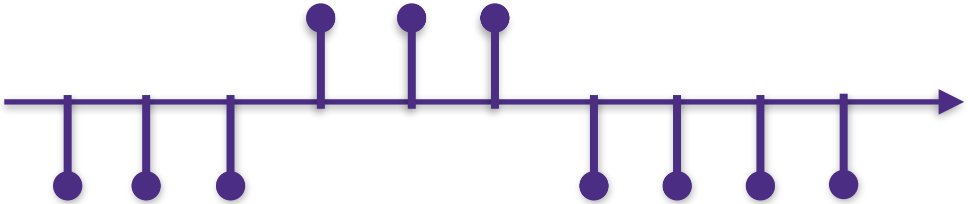


Output $\theta * x \in \mathbb{R}^{n-m+1}$

1d convolution

- Each filter finds a specific pattern over the input

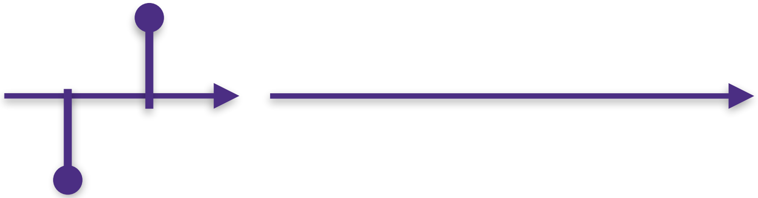
x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
-1	-1	-1	1	1	1	-1	-1	-1	-1



Filter 1

θ_0	θ_1
-1	1

$x \star \theta =$



Filter 2

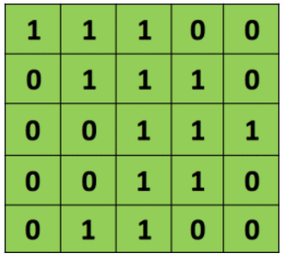
θ_0	θ_1
1	1

$x \star \theta =$

- We use many such convolutional filters per layer in practice
- Each convolutional filter output vector (or a matrix if 2D convolution) is called a channel

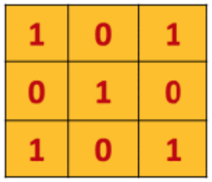
Convolution of images (2d convolution)

$$(I * K)(i,j) = \sum_{m} \sum_{n} I(i+m, j+n)K(m,n).$$



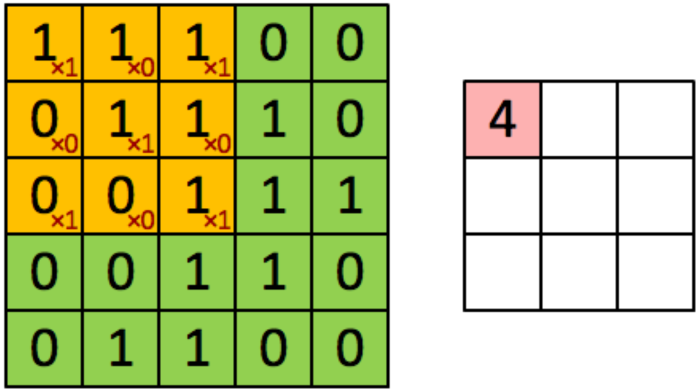
1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Image I



1	0	1
0	1	0
1	0	1

Filter K



1 ×1	1 ×0	1 ×1	0	0
0 ×0	1 ×1	1 ×0	1	0
0 ×1	0 ×0	1 ×1	1	1
0	0	1	1	0
0	1	1	0	0

4		

Image

Convolved Feature

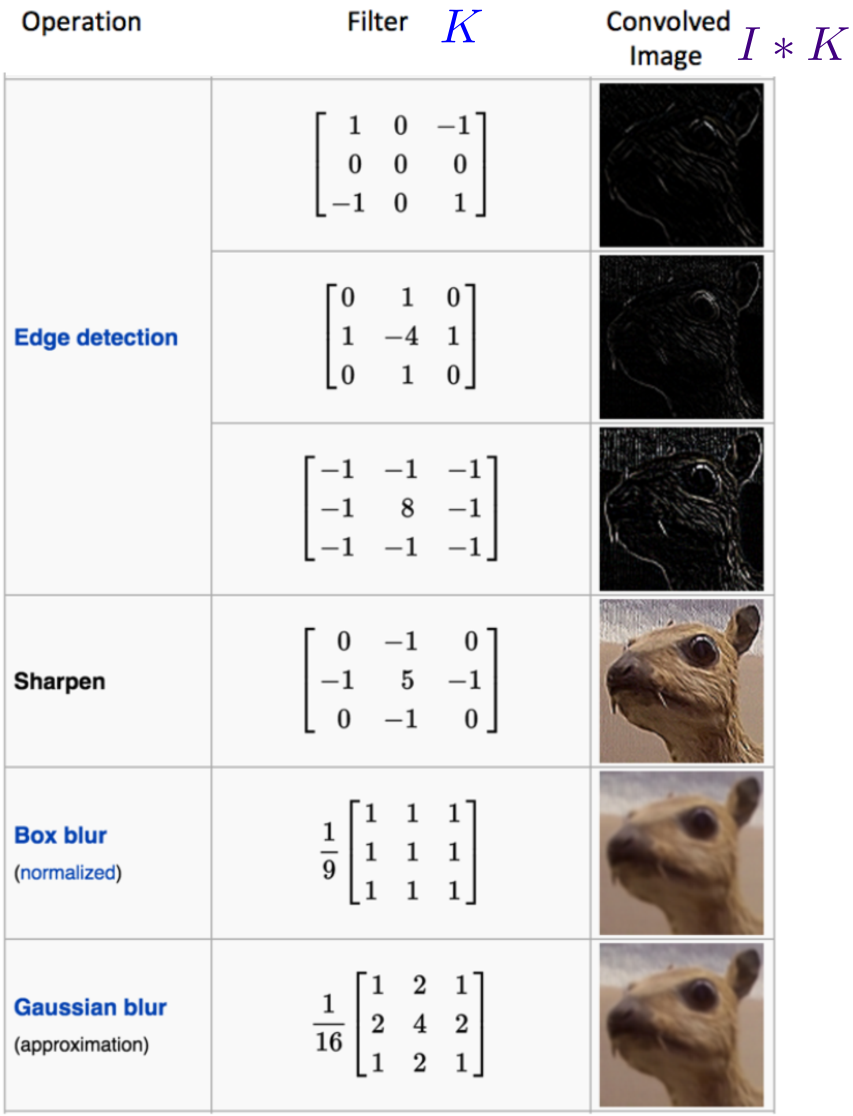
$I * K$

Convolution of images

- These are hand-crafted filters, to illustrate what the weights of a filter mean
- Filter in a Convolutional Neural Network (CNN) is learned, and we might be able to interpret what we learned

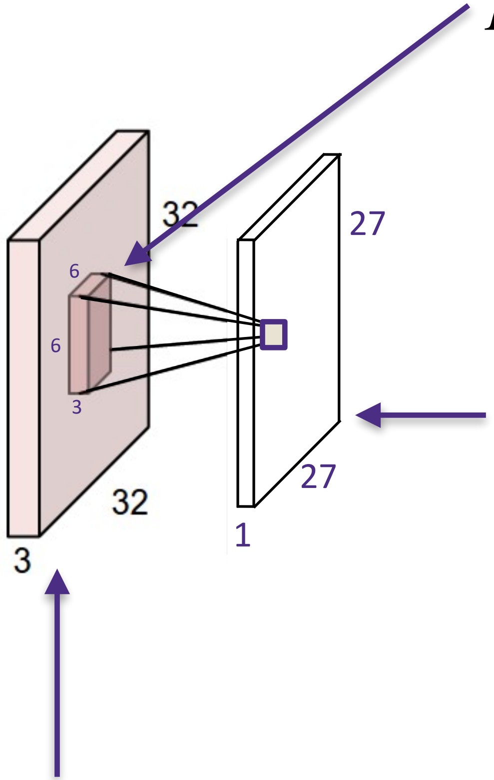
$$(I * K)(i,j) = \sum_m \sum_n I(i+m, j+n) K(m,n)$$

Image I



Operation	Filter K	Convolved Image $I * K$
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

Stacking convolved images



$$K \in \mathbb{R}^{m \times m \times r}$$

- If we use a convolutional layer with 1 filter of size m*m*r=6*6*3, then the output is a matrix of dimension (n+1-m)*(n+1-m)=27*27

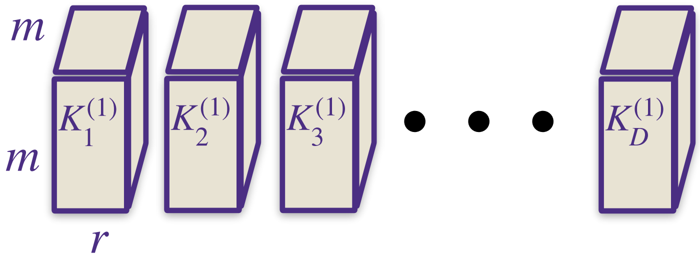
$$x \star K \in \mathbb{R}^{(n+1-m)\times(n+1-m)}$$

- Input is a multi-array or a tensor, because it has 3 color channels

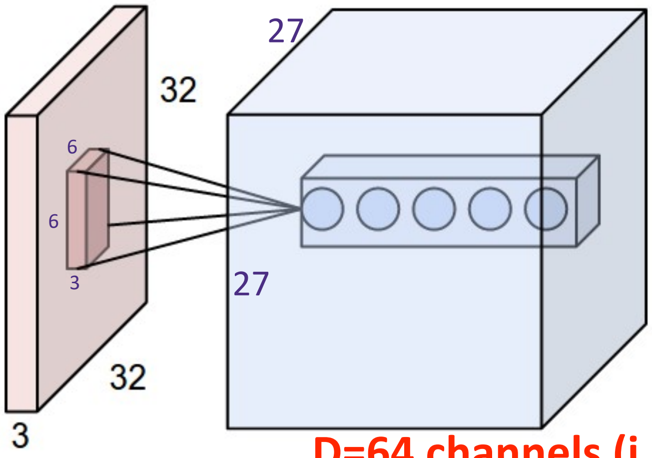
$$x \in \mathbb{R}^{n \times n \times r}$$

Stacking convolved images

- Typical convolutional layer has multiple filters to capture multiple patterns
- Each one is called a channel
- Each channel has a filter of the same size m*m*r but with different weights



- Each channel outputs a matrix of dimension (n+1-m)*(n+1-m)
- Put together the output is a tensor of dimension (n+1-m)*(n+1-m)*D

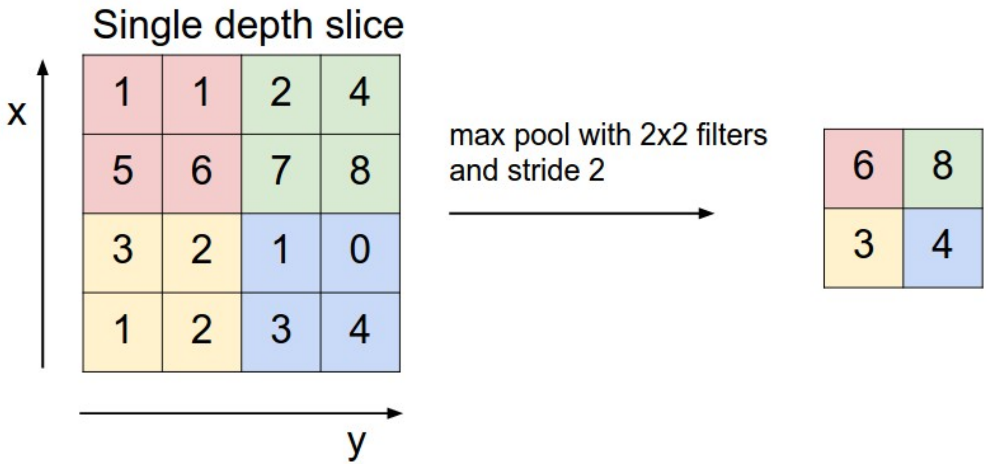


D=64 channels (i.e. filters)

Repeat with D filters!

Max Pooling gives a summary of a region

Pooling reduces the dimension and can be interpreted as “This filter had a high response in this general region”

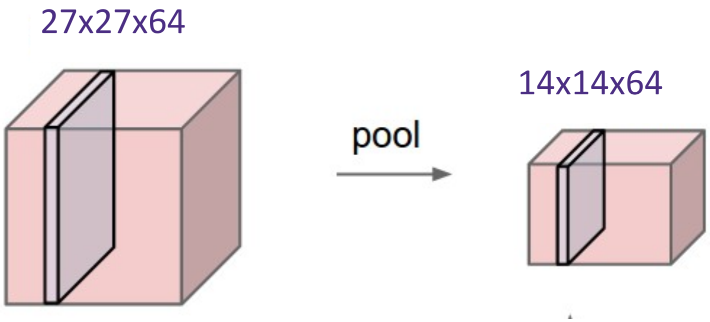


Single depth slice

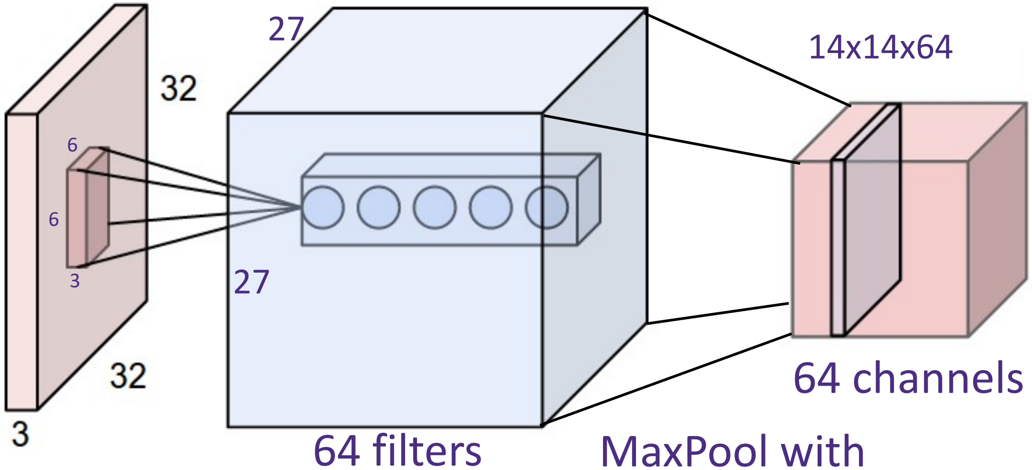
1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

max pool with 2x2 filters and stride 2

6	8
3	4



Pooling Convolution layer



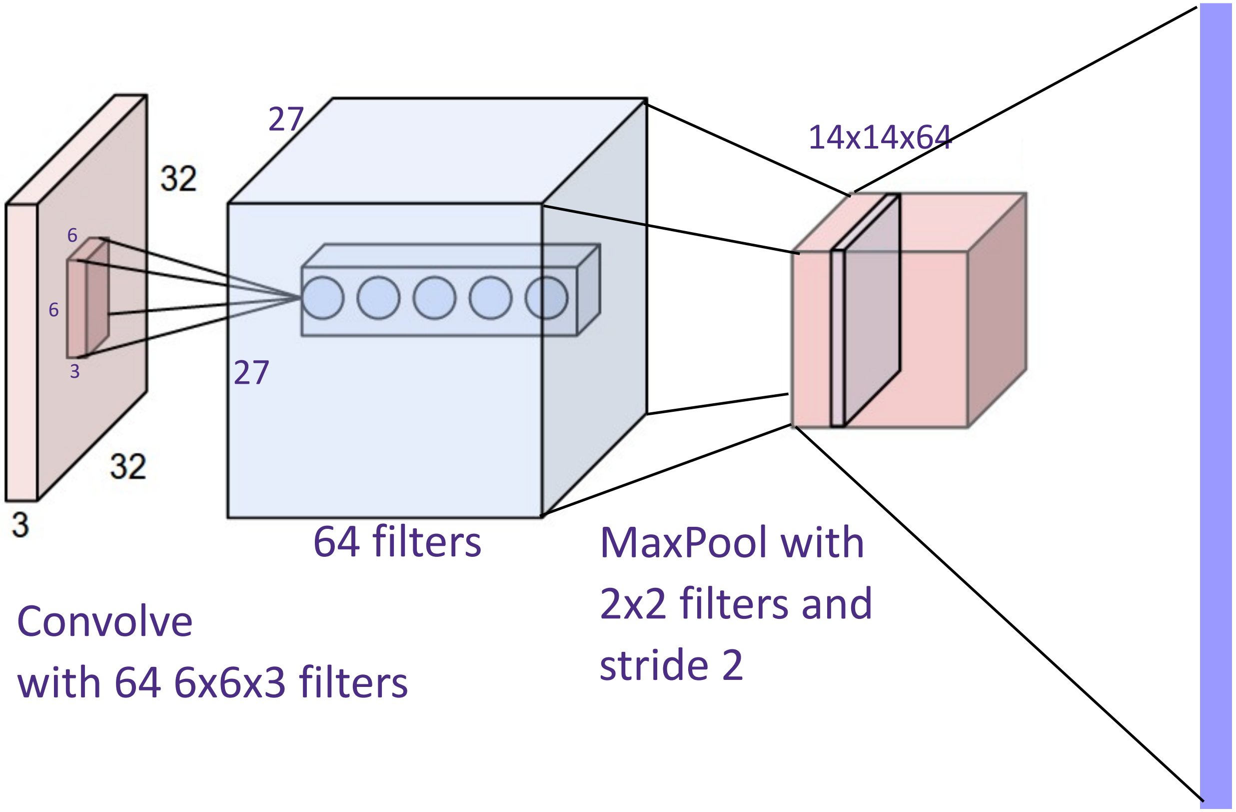
64 channels

64 filters

Convolve
with 64 6x6x3 filters

MaxPool with
2x2 filters and
stride 2

Flattening

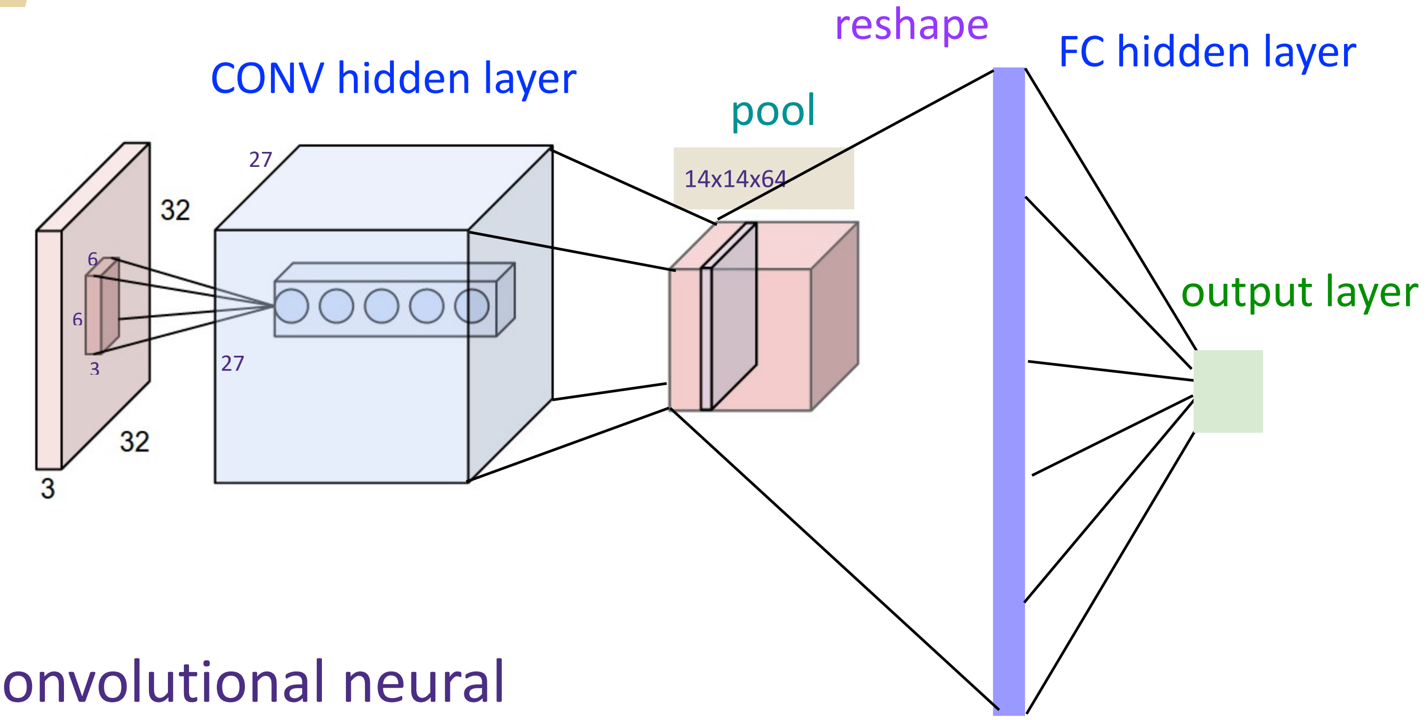


Convolve
with 64 6x6x3 filters

MaxPool with
2x2 filters and
stride 2

Flatten into a single
vector of size
14*14*64=12544

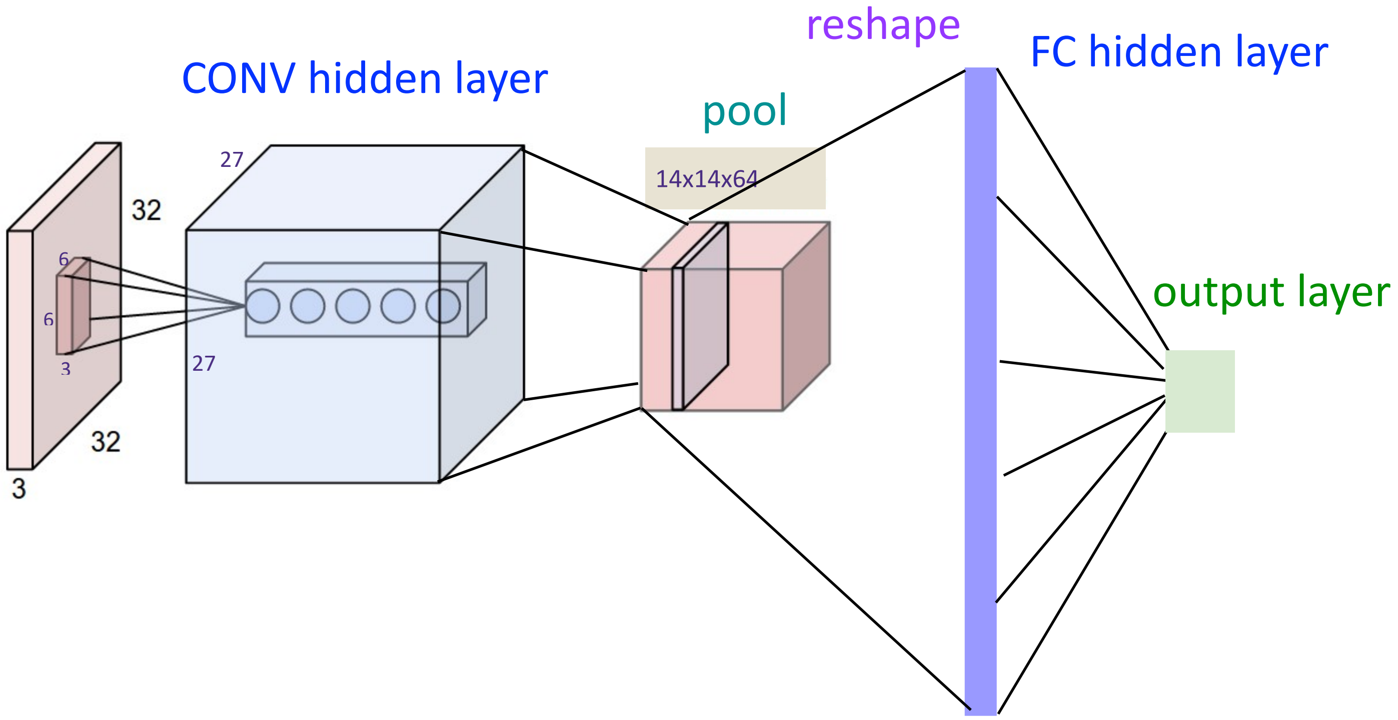
Training Convolutional Networks



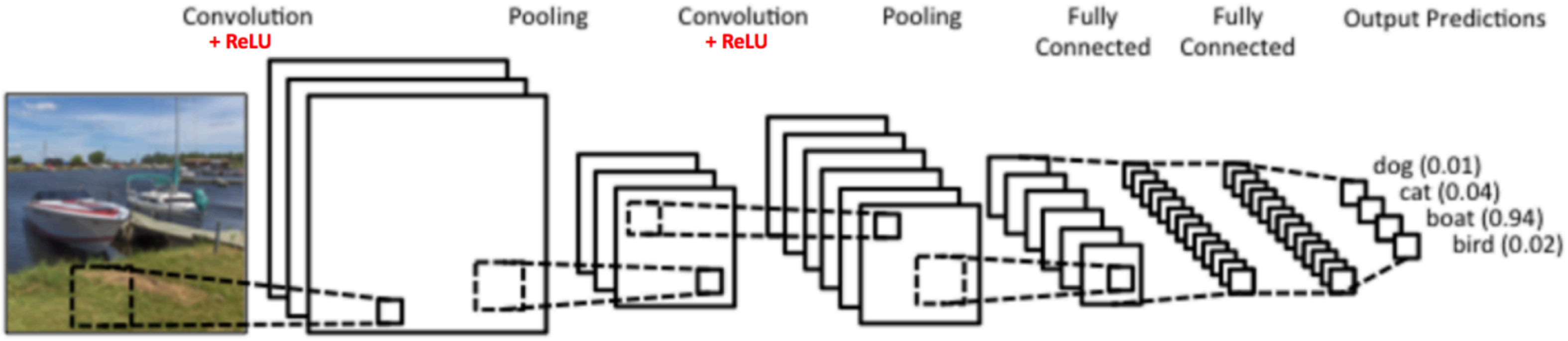
Recall: Convolutional neural networks (CNN) are just regular fully connected (FC) neural networks with some connections removed and some weights shared.

Train with SGD!

Training Convolutional Networks

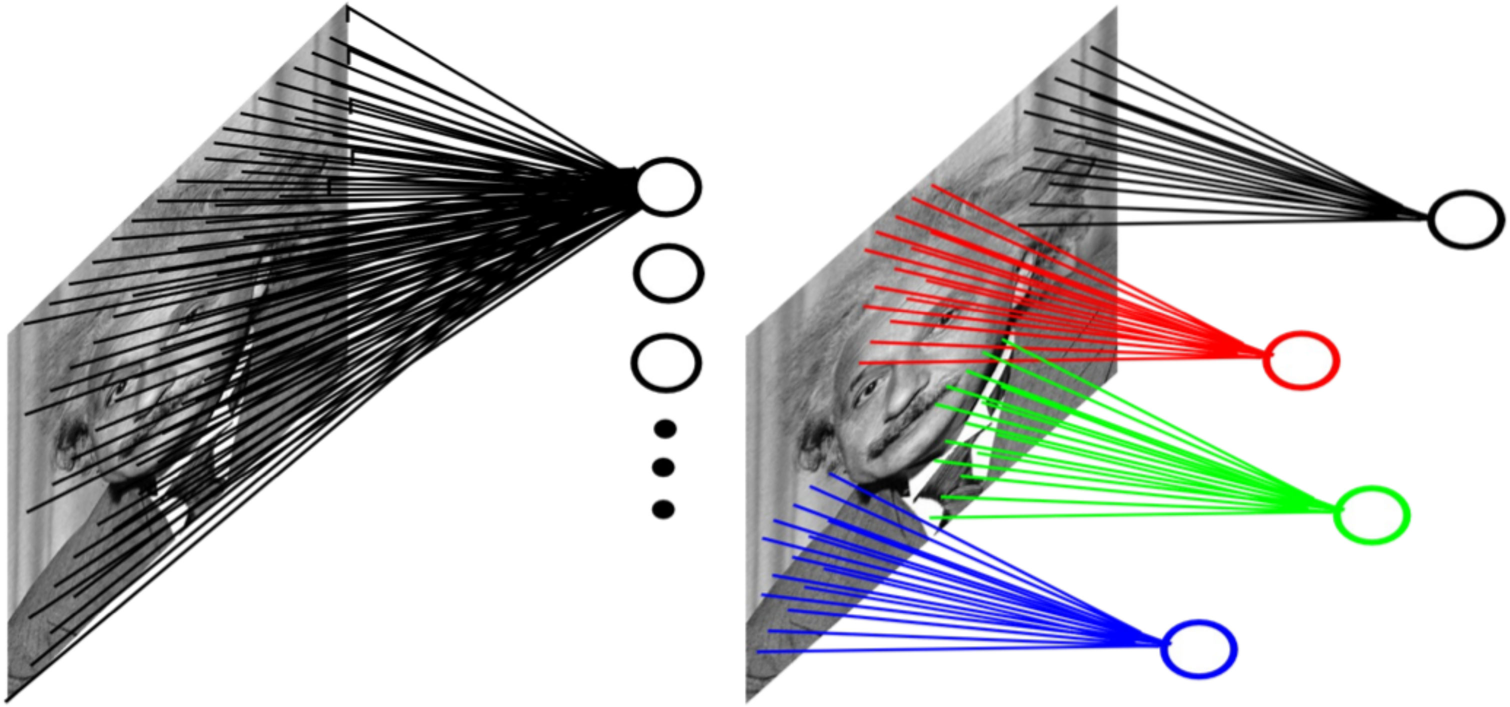


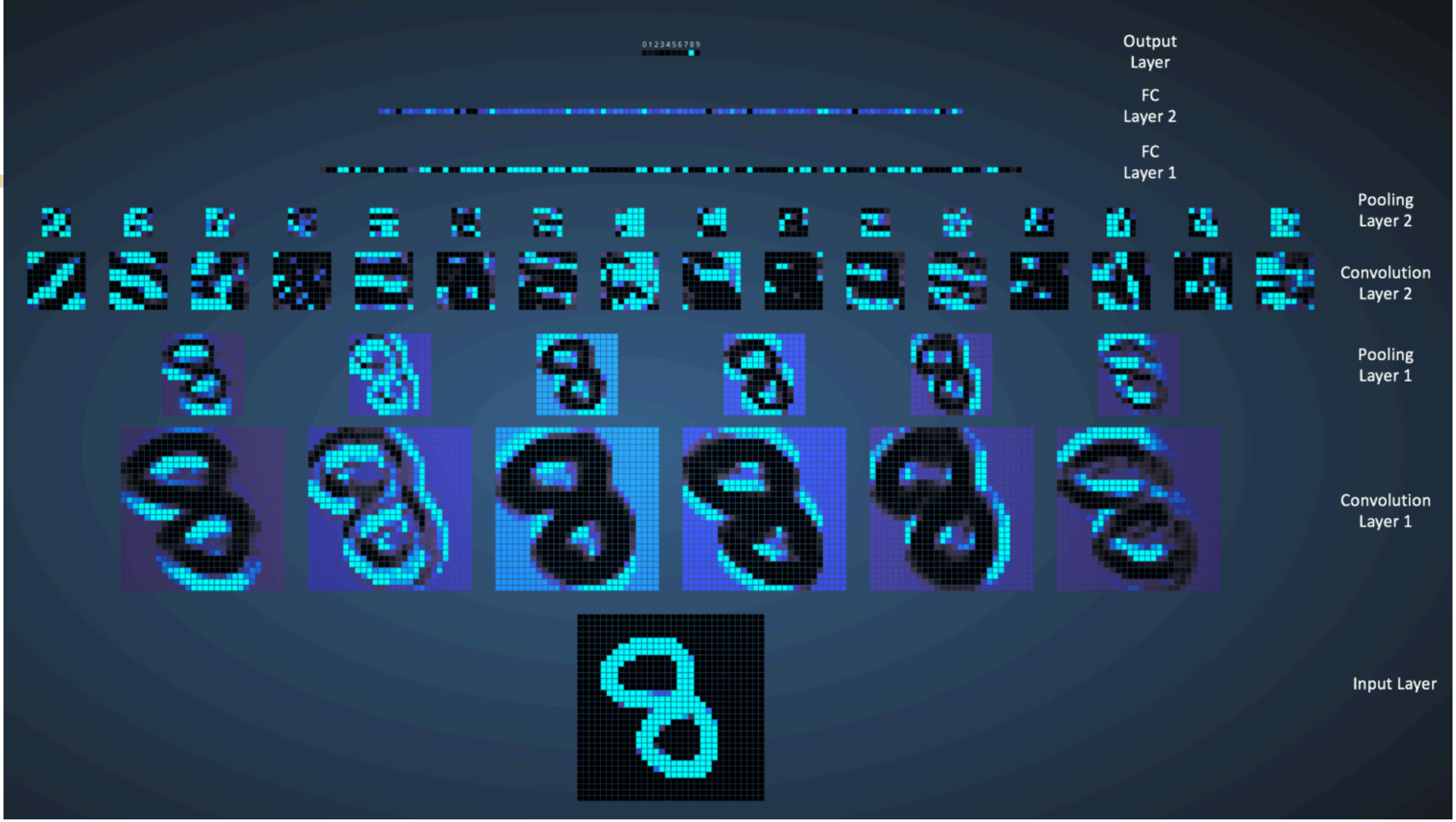
Real example network: LeNet



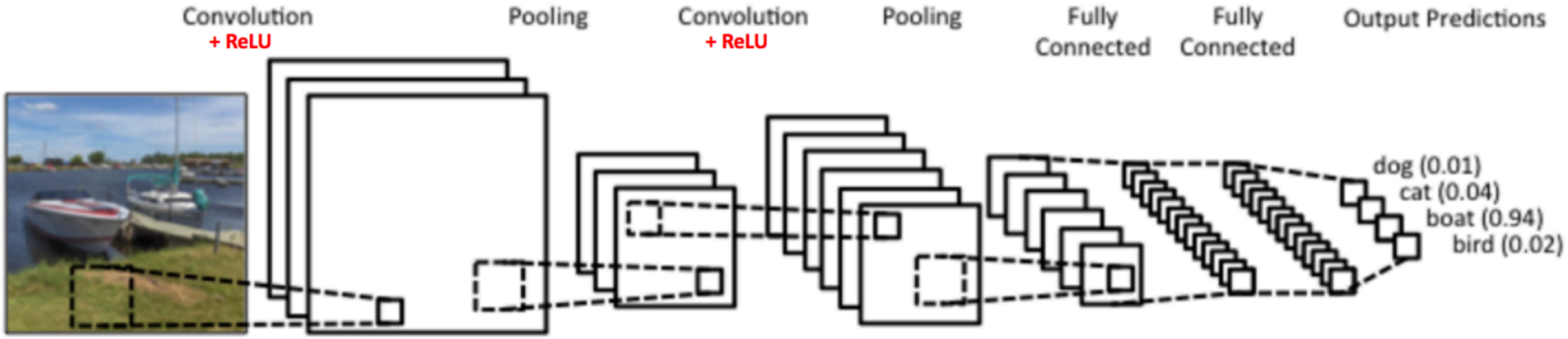
2d Convolution Layer

- **Example: 200x200 image**
  - Fully-connected, 400,000 hidden units = 16 billion parameters
  - Locally-connected, 400,000 hidden units 10x10 fields = 40 million params or channels or filter
  - Local connections capture local dependencies





Real example network: LeNet



Remarks

- Convolution is a fundamental operation in signal processing. Instead of hand-engineering the filters (e.g., Fourier, Wavelets, etc.) **Deep Learning *learns* the filters and CONV layers with back-propagation**, replacing fully connected (FC) layers with convolutional (CONV) layers
- **Pooling** is a dimensionality reduction operation that summarizes the output of convolving the input with a filter

- Typically the last few layers are **Fully Connected (FC)**, with the interpretation that the CONV layers are feature extractors, preparing input for the final FC layers. Can replace last layers and retrain on different dataset+task.

- Just as hard to train as regular neural networks.
- More exotic network architectures for specific tasks

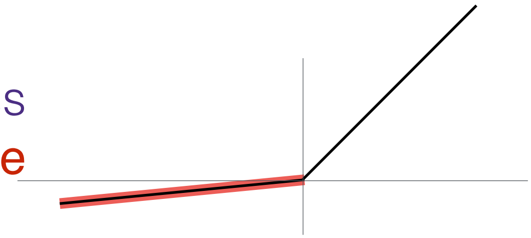
Real networks

Residual Network of [HeZhangRenSun’15]

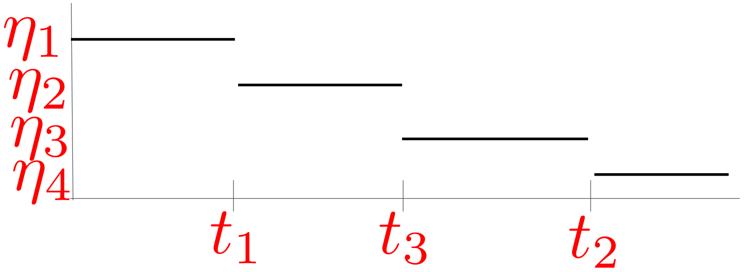
Modern networks have dozens of parameters to tune.

Data augmentation?
Batch norm?

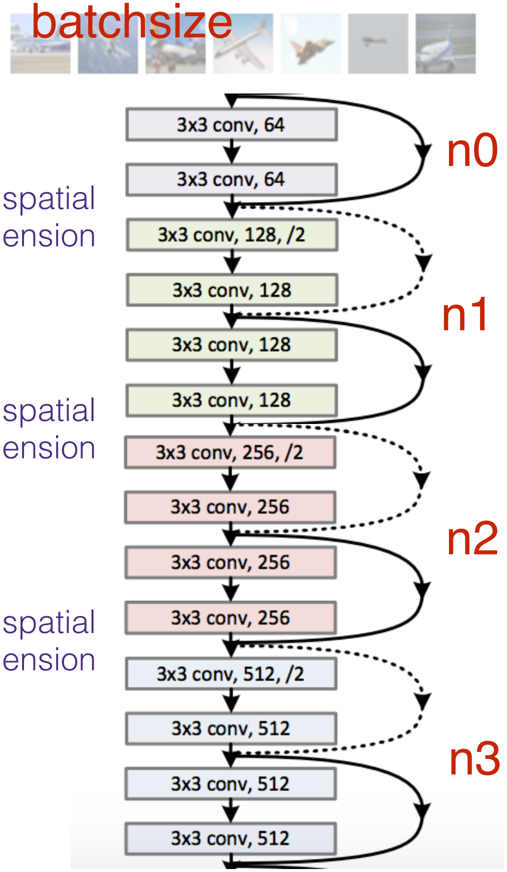
RELU leakiness
slope



Learning rate schedule



batchsize



Reduce spatial dimension

Reduce spatial dimension

Reduce spatial dimension

n0 layers of f0 filters

n1 layers of f1 filters

n2 layers of f2 filters

n3 layers of f3 filters