Carnegie Mellon University

Introduction to Deep Learning for Engineers

Spring 2025, Deep Learning for Engineers
Feb 8, 2025, Eighth Session

Amir Barati Farimani
Associate Professor of Mechanical Engineering and Bio-Engineering
Carnegie Mellon University

Story so far on CNN:

1. **Concept of Representation + Learning**

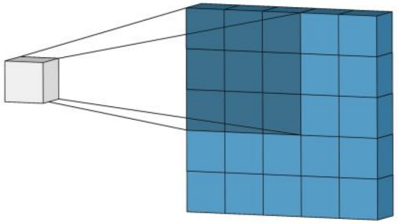
We need a UNIVERSAL REPRESENTATION LEARNER to ease the learning

2. How to learn robust representation?

We need a representation learner that can automatically learn the features needed for the task

3. How to learn spatial, high level features (Swan example)

Since high level features are spatial (not pixels) we need a scanner to scan patches of data instead of single pixels



Carnegie Mellon University

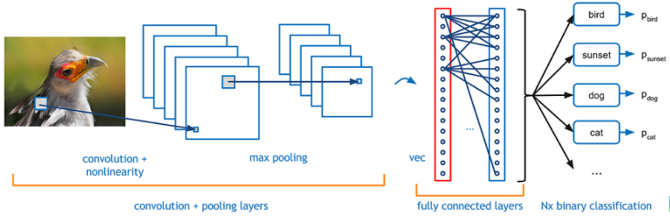
Story so far on CNN:

4. How to build a scanner for feature learning? what should be the properties of this scanner?

1. **It should be numbers (a matrix) because it should be machine readable**
2. **It should be learnable**
3. **It should be flexible in size and dimension**
4. **Should be pluggable to Neural Networks**

5. Can we design the scanners based on the learning tasks?

Yes, and we should. Because the mode of data might be different (sound, image, video) and features are needed based on the task to make a good model, the scanner should ONLY learn the relevant features connecting them to the output



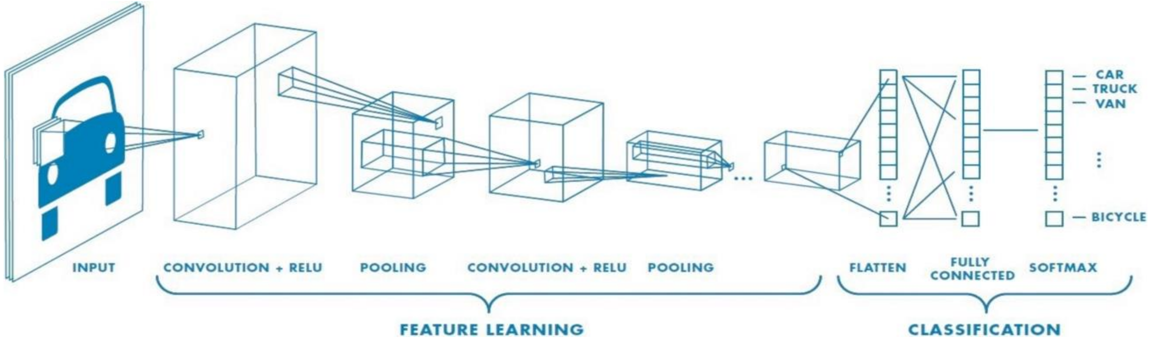
Story so far on CNN:

6. How can the scanners learn?

Inspired by iterative optimization and backpropagation in neural networks, we can iteratively learn the initialized weights of scanners (remember these are numbers)

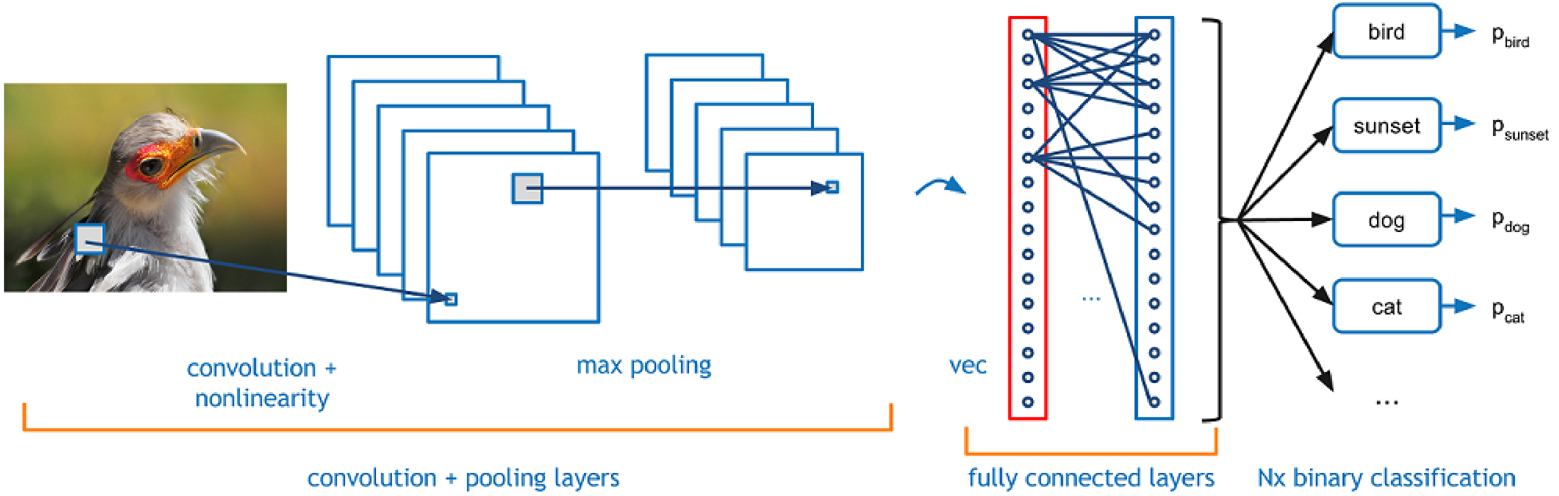
7. How can we plug in the scanners into Neural networks?

By flattening the output of the last convolved map and passing it to the FC layer, we can forward propagate, and we can backpropagate to learn the parameters of a filter (scanner)



Carnegie Mellon University

Recap: CNN Overall Architecture



CNNs are Automatic Feature Detectors

Sharing Weights

Feature Detection

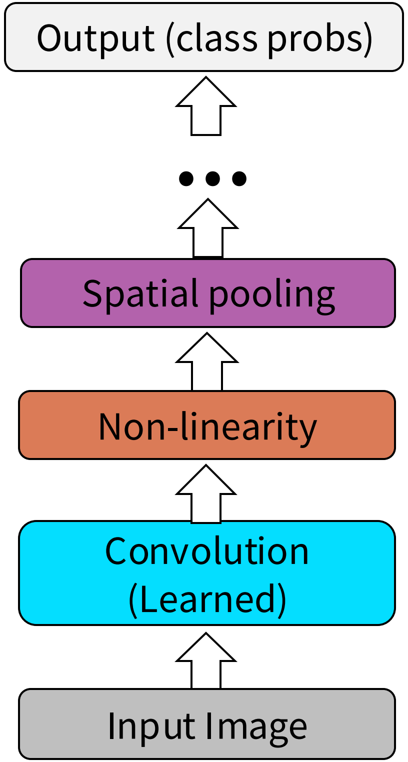
Spatial Local Features

Translation In-variant

Convolutional Neural Networks (CNN)

- FEED-FORWARD FEATURE EXTRACTION:
  1. **Convolve input with learned filters**
  2. Apply non-linearity
  3. **Spatial pooling (downsample)**

- SUPERVISED TRAINING OF CONVOLUTIONAL FILTERS BY BACK-PROPAGATING CLASSIFICATION ERROR

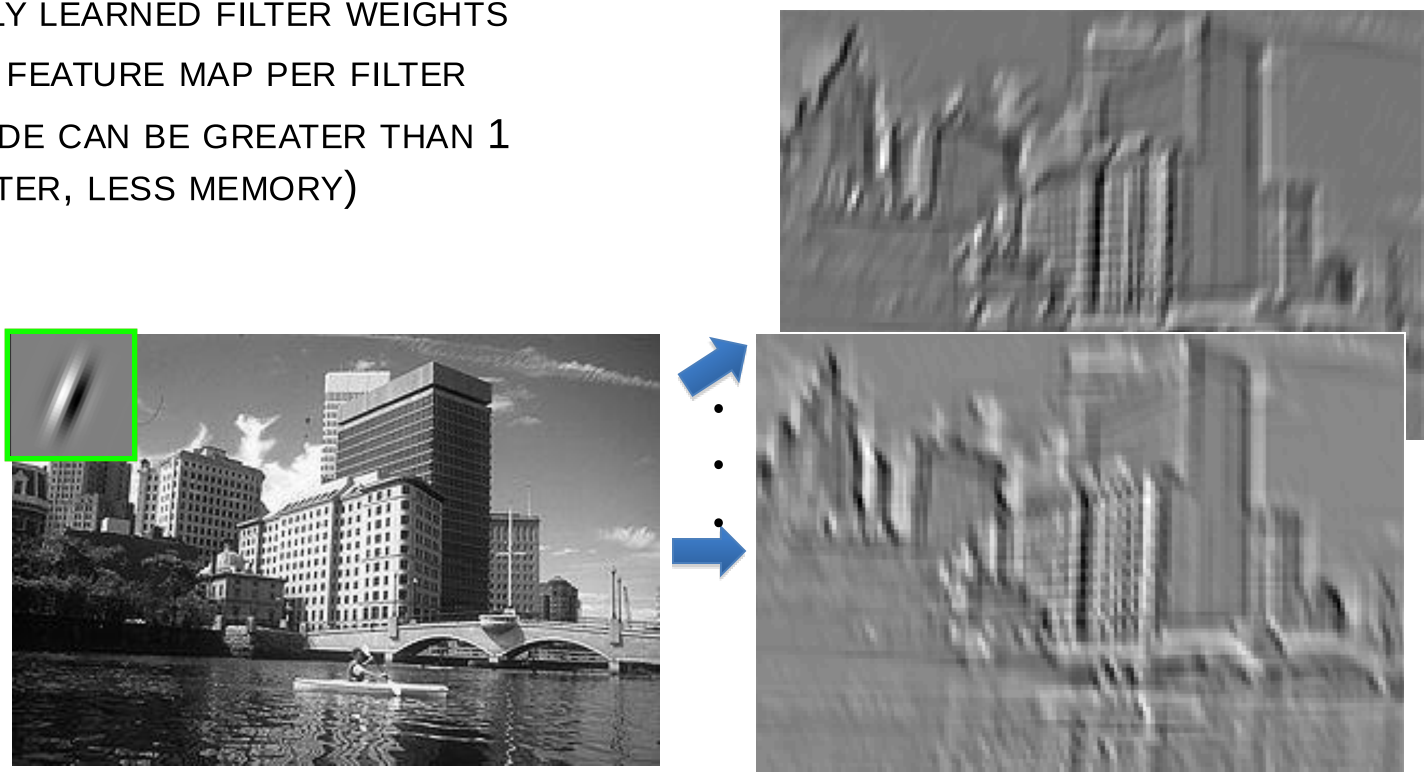


Carnegie Mellon University

Adapted from Lana Lazebnik

Convolution

- Apply learned filter weights
- One feature map per filter
- Stride can be greater than 1 (faster, less memory)



Input

Feature Map

Adapted from Rob Fergus

The Convolution operation

ASSUME A 6X6 MATRIX **M** AS INPUT. THE 2D CONVOLUTION OF **M** WITH *FILTER (OR KERNEL)* **F** AND *STRIDE* 1 IS A 4X4 MATRIX **CM** (SOMETIMES CALLED *FEATURE MAP*) COMPUTED AS FOLLOWS:

3	0	1	2	4	7
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

M

*

1	0	-1
1	0	-1
1	0	-1

F

=

CM

convoluti
on
operator

Carnegie Mellon University

The Convolution operation

Assume a 6x6 matrix **M** as input. The 2D convolution of **M** with *filter (or kernel)* **F** and *stride* 1 is a 4x4 matrix **CM** (sometimes called *feature map*) computed as follows:

3	0	1	2	4	7
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

M

*

1	0	-1
1	0	-1
1	0	-1

F

=

-5			

CM

$$3*1 + 1*1 + 2*1 + 0*0 + 5*0 + 7*0 + 1*(-1) + 8*(-1) + 2*(-1)$$

Carnegie Mellon University

The Convolution operation

Assume a 6x6 matrix **M** as input. The 2D convolution of **M** with *filter (or kernel)* **F** and *stride* 1 is a 4x4 matrix **CM** (sometimes called *feature map*) computed as follows:

3	0	1	2	4	7
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

M

*

1	0	-1
1	0	-1
1	0	-1

F

=

-5	-4		

CM

$$0*1 + 5*1 + 7*1 + 1*0 + 8*0 + 2*0 + 2*(-1) + 9*(-1) + 5*(-1)$$

The Convolution operation

Assume a 6x6 matrix **M** as input. The 2D convolution of **M** with *filter (or kernel)* **F** and *stride* 1 is a 4x4 matrix **CM** (sometimes called *feature map*) computed as follows:

3	0	1	2	4	7
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

M

*

1	0	-1
1	0	-1
1	0	-1

F

=

-5	-4	0	

C
M

1*1 + 8*1 + 2*1 + 2*0 + 9*0 + 5*0 + 4*(-1) + 3*(-1) + 1*(-1)

The Convolution operation

Assume a 6x6 matrix **M** as input. The 2D convolution of **M** with *filter (or kernel)* **F** and *stride* 1 is a 4x4 matrix **CM** (sometimes called *feature map*) computed as follows:

3	0	1	2	4	7
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

M

*

1	0	-1
1	0	-1
1	0	-1

F

=

-5	-4	0	8

CM

$$2*1 + 9*1 + 5*1 + 4*0 + 3*0 + 1*0 + 7*(-1) + 1*(-1) + 3*(-1)$$

The Convolution operation

ASSUME A 6X6 MATRIX **M** AS INPUT. THE 2D CONVOLUTION OF **M** WITH *FILTER (OR KERNEL)* **F** AND *STRIDE* 1 IS A 4X4 MATRIX **CM** (SOMETIMES CALLED *FEATURE MAP*) COMPUTED AS FOLLOWS:

3	0	1	2	4	7
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

M

*

1	0	-1
1	0	-1
1	0	-1

F

=

-5	-4	0	8
-10			

CM

$$1*1 + 2*1 + 0*1 + 5*0 + 7*0 + 1*0 + 8*(-1) + 2*(-1) + 3*(-1)$$

Carnegie Mellon University

The Convolution operation

Assume a 6x6 matrix **M** as input. The 2D convolution of **M** with *filter (or kernel)* **F** and *stride* 1 is a 4x4 matrix **CM** (sometimes called *feature map*) computed as follows:

3	0	1	2	4	7
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

M

*

1	0	-1
1	0	-1
1	0	-1

F

=

-5	-4	0	8
-10	-2		

CM

5*1 + 7*1 + 1*1 + 8*0 + 2*0 + 3*0 + 9*(-1) + 5*(-1) + 1*(-1)

The Convolution operation[1]

Assume a 6x6 matrix **M** as input. The 2D convolution of **M** with *filter (or kernel)* **F** and *stride* 1 is a 4x4 matrix **CM** (sometimes called *feature map*) computed as follows:

3	0	1	2	4	7
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

M

*

1	0	-1
1	0	-1
1	0	-1

F

=

-5	-4	0	8
-10	-2	2	3
0	-2	-4	-7
-3	-2	-3	-16

CM

1*1 + 6*1 + 2*1 + 7*0 + 2*0 + 3*0 + 8*(-1) + 8*(-1) + 9*(-1)

Larger Strides

ASSUME A M, A 7X7 MATRIX. THE 2D CONVOLUTION OF M WITH *FILTER* F AND ***STRIDE* 2** IS A 3X3 MATRIX **CM** COMPUTED AS FOLLOWS:

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

M

*

3	4	4
1	0	2
-1	0	3

F

=

91		

CM

$$2*3 + 6*1 + 3*(-1) + 3*4 + 6*0 + 4*0 + 7*4 + 9*2 + 8*3$$

Larger Strides

ASSUME A M, A 7X7 MATRIX. THE 2D CONVOLUTION OF M WITH *FILTER* F AND ***STRIDE* 2** IS A 3X3 MATRIX **CM** COMPUTED AS FOLLOWS:

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

M

$*$

3	4	4
1	0	2
-1	0	3

F

=

91	100	

C
M

$7*3 + 9*1 + 8*(-1) + 4*4 + 8*0 + 3*0 + 6*4 + 7*2 + 8*3$

Carnegie Mellon University

Larger Strides

ASSUME A M, A 7X7 MATRIX. THE 2D CONVOLUTION OF M WITH *FILTER* F AND ***STRIDE* 2** IS A 3X3 MATRIX **CM** COMPUTED AS FOLLOWS:

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

M

*

3	4	4
1	0	2
-1	0	3

F

=

91	100	88

CM

$$6*3 + 7*1 + 8*(-1) + 2*4 + 4*0 + 9*0 + 9*4 + 3*2 + 7*3$$

Larger Strides

ASSUME A M, A 7X7 MATRIX. THE 2D CONVOLUTION OF M WITH *FILTER* F AND ***STRIDE* 2** IS A 3X3 MATRIX **CM** COMPUTED AS FOLLOWS:

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

M

*

3	4	4
1	0	2
-1	0	3

F

=

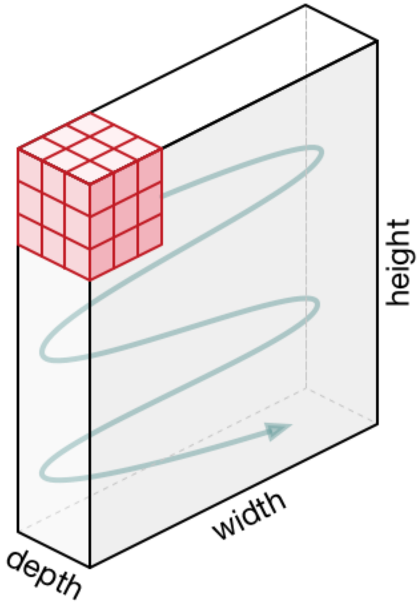
91	100	88
57		

C
M

3*3 + 7*1 + 4*(-1) + 4*4 + 8*0 + 2*0 + 8*4 + 3*2 + 1*3

Carnegie Mellon University

Recap: How it works for Images?



0	0	0	0	0	0	...
0	156	155	156	158	158	...
0	153	154	157	159	159	...
0	149	151	155	158	159	...
0	146	146	149	153	158	...
0	145	143	143	148	158	...
...	...	...	...	...	...	...

Input Channel #1 (Red)

0	0	0	0	0	0	...
0	167	166	167	169	169	...
0	164	165	168	170	170	...
0	160	162	166	169	170	...
0	156	156	159	163	168	...
0	155	153	153	158	168	...
...	...	...	...	...	...	...

Input Channel #2 (Green)

0	0	0	0	0	0	...
0	163	162	163	165	165	...
0	160	161	164	166	166	...
0	156	158	162	165	166	...
0	155	155	158	162	167	...
0	154	152	152	157	167	...
...	...	...	...	...	...	...

Input Channel #3 (Blue)

-1	-1	1
0	1	-1
0	1	1

Kernel Channel #1

1	0	0
1	-1	-1
1	0	-1

Kernel Channel #2

0	1	1
0	1	0
1	-1	1

Kernel Channel #3

$$308 \quad + \quad -498 \quad + \quad 164 \quad + 1 = -25$$

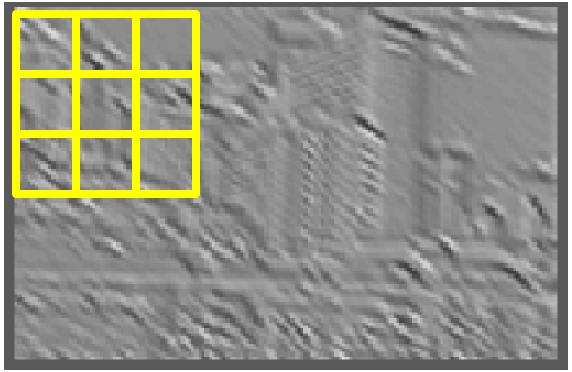
Bias = 1

Output

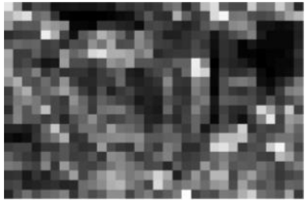
-25				...
				...
				...
				...
...	...	...	...	...

Spatial Pooling

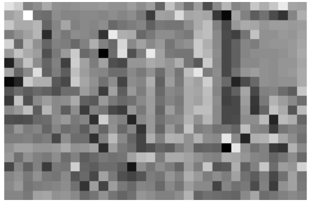
- Sum or max over non-overlapping / overlapping regions
- Role of pooling:
  - Invariance to small transformations
  - Larger receptive fields (neurons see more of input)



Max



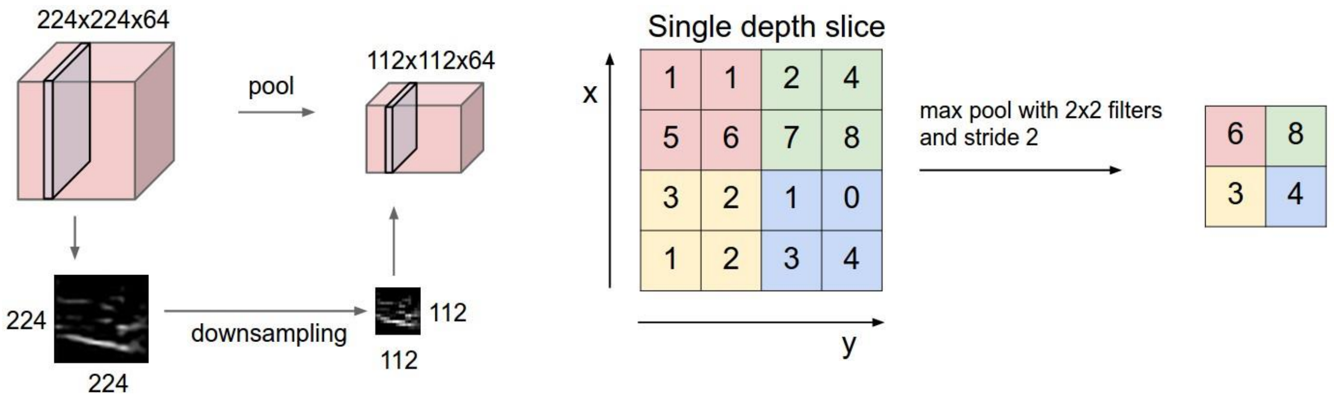
Sum



Adapted from Rob Fergus

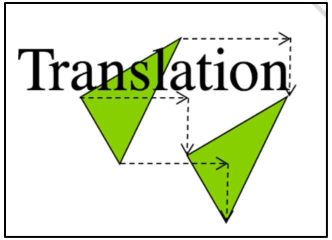
Spatial Pooling

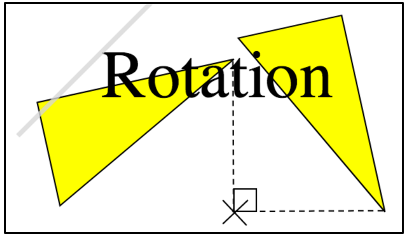
- Sum or max over non-overlapping / overlapping regions
- Role of pooling:
  - Invariance to small transformations
  - Larger receptive fields (neurons see more of input)

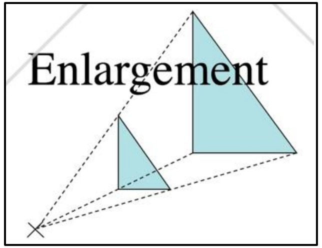


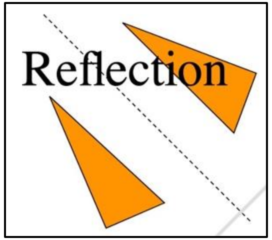
Rob Fergus, figure from Andrej Karpathy

Some Types of Linear Transformations









Types of Pooling functions

- A pooling function replaces the output of the net at a certain location with a summary statistic of the nearby inputs
- Popular pooling functions:
  1. *max pooling* operation reports the maximum output within a rectangular neighborhood
  2. Average of a rectangular neighborhood
  3. L^2 norm of a rectangular neighborhood
  4. Weighted average based on the distance from the central pixel

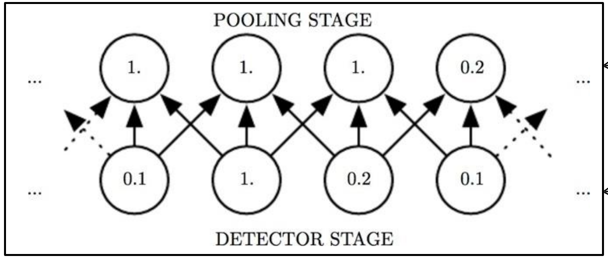
Pooling Causes Translation Invariance

- In all cases, pooling helps make the representation become approximately *invariant* to small translations of the input
  - If we translate the input by a small amount values of most of the outputs does not change (example next slide)
  - Pooling can be viewed as adding a strong prior that the function the layer learns must be invariant to small translations

Carnegie Mellon University

Max Pooling Introduces Invariance to Translation

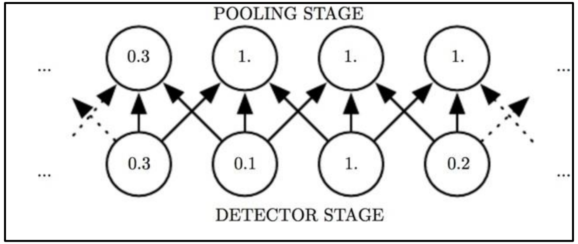
- View of middle of output of a convolutional layer



Outputs of maxpooling

Outputs of nonlinearity

- Same network after the input has been shifted by one pixel



- Every input value has changed, but only half the values of output have changed because maxpooling units are only sensitive to maximum value in neighborhood not exact value

Carnegie Mellon University

Importance of Translation Invariance

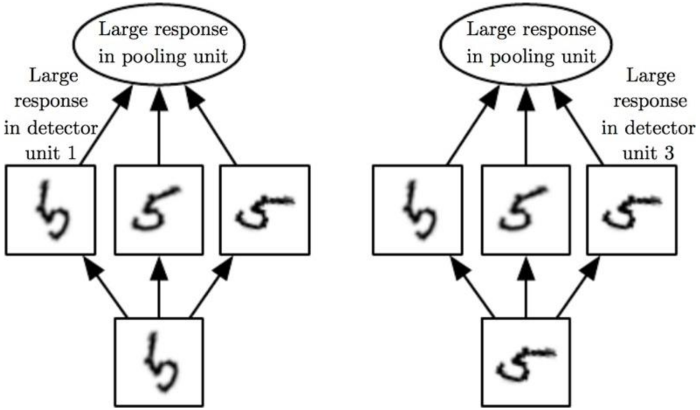
- Invariance to translation is important if we care about whether a feature is present rather than exactly where it is
  - Ex: for detecting a face just need to know that an eye is present in a region, not its exact location
- In other context it is more important to preserve location of a feature
  - Ex: to determine a corner we need to know whether two edges are present and test whether they meet

Learning other Invariances

- Pooling over spatial regions produces invariance to translation
- But if we pool over the results of separately parameterized convolutions, the features can learn which transformations to become invariant to

Learning Invariance to Rotation

- A pooling unit that pools over multiple features that are learned with separate parameters can learn to be invariant to transformations of the input



Input tilted left gets large response from unit tuned to left-tilted images

Tilted Right

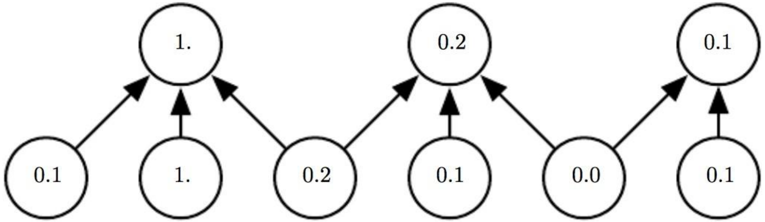
Carnegie Mellon University

Using Fewer Pooling Units than Detector Units

- Because pooling summarizes the responses over a whole neighborhood, it is possible to use fewer pooling units than detector units
  - This is due to the effect of reporting summary statistics for pooling regions spaced k pixels apart rather than one pixel apart
  - This improves computational efficiency because the next layer has k times fewer inputs to process
- An example is given next

Pooling with Downsampling

- Max-pooling with a pool width of three and a stride between pools of two

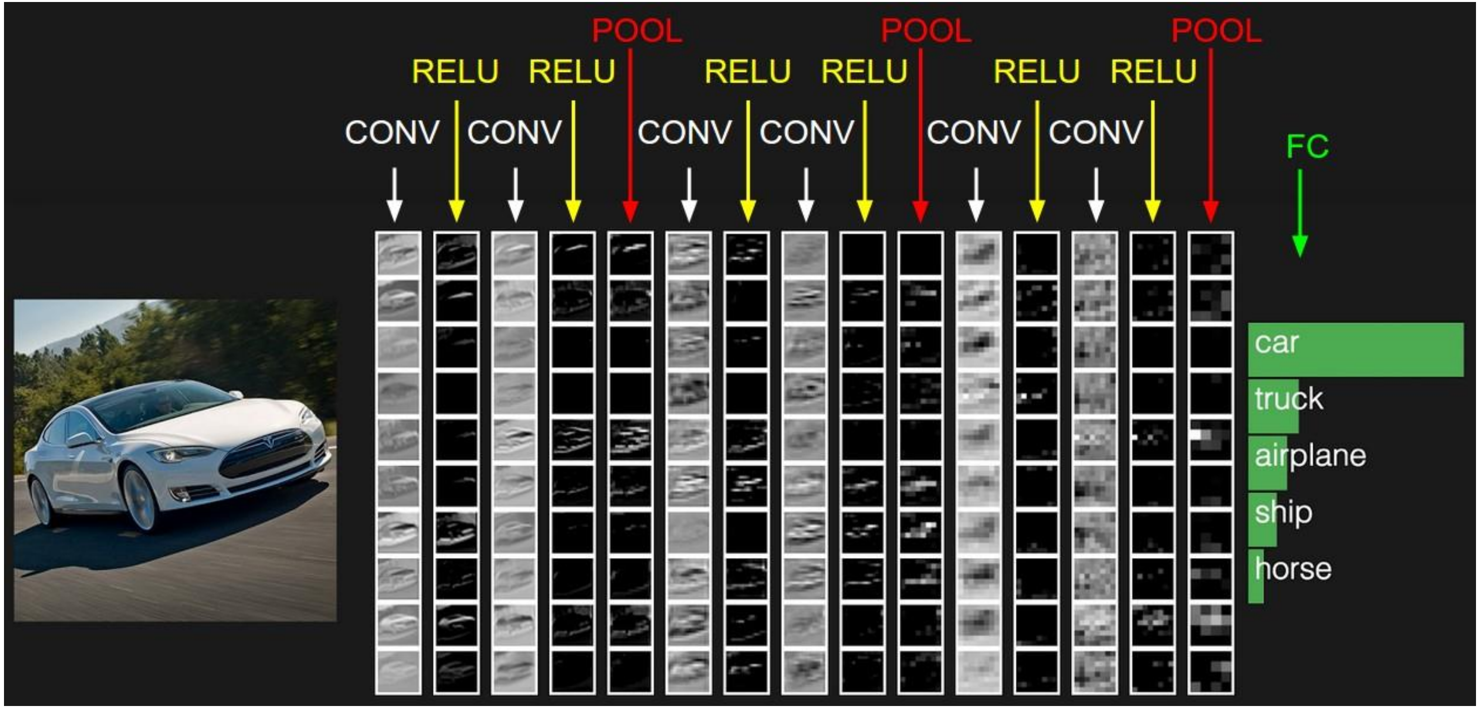


- This reduces representation size by a factor of two
  - Which reduces computational burden of next layer
  - Rightmost pooling region has a smaller size but must be included if we don't want to ignore some of the detector units

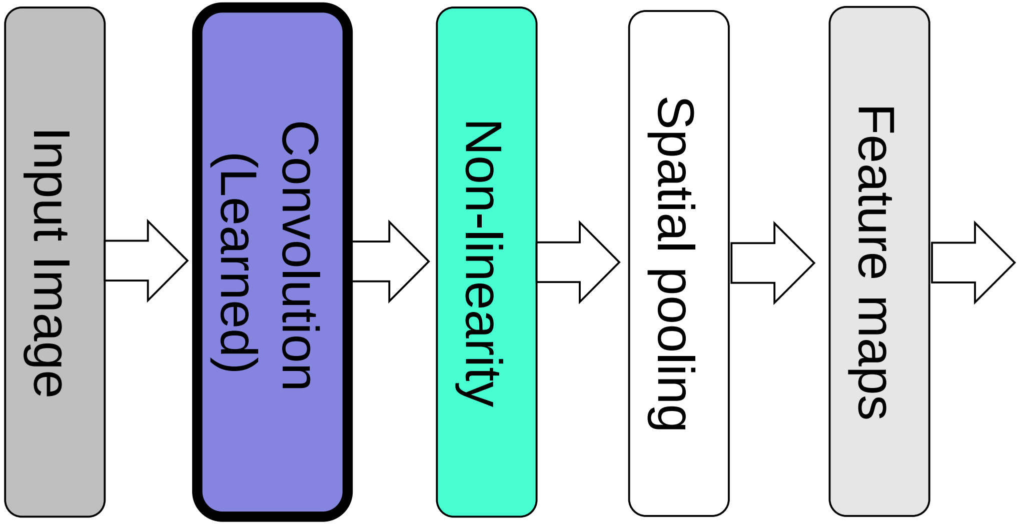
Downsampling makes a digital audio signal smaller by lowering its sampling rate or sample size (bits per sample). It decreases the bit rate when transmitting over a limited bandwidth or to convert to a more limited audio format

Carnegie Mellon University

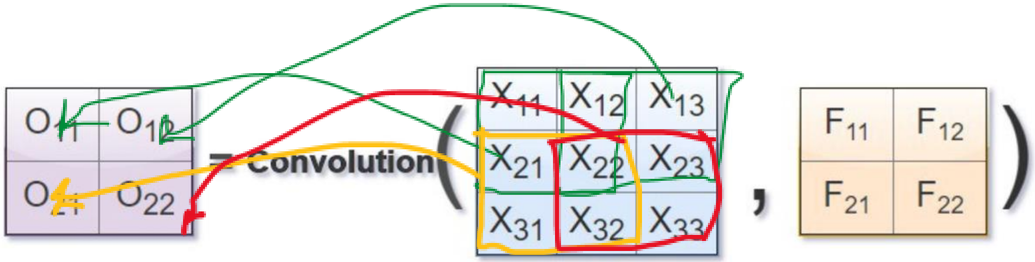
Feature Map (Convolution Layer)



CNN Components



Backprop in Convolution



$$O_{11} = F_{11}X_{11} + F_{12}X_{12} + F_{21}X_{21} + F_{22}X_{22}$$

$$O_{12} = F_{11}X_{12} + F_{12}X_{13} + F_{21}X_{22} + F_{22}X_{23}$$

$$O_{21} = F_{11}X_{21} + F_{12}X_{22} + F_{21}X_{31} + F_{22}X_{32}$$

$$O_{22} = F_{11}X_{22} + F_{12}X_{23} + F_{21}X_{32} + F_{22}X_{33}$$

Slide taken from Forward And Backpropagation in Convolutional Neural Network. - Medium

Backprop in Convolution

$$\begin{bmatrix} O_{11} & O_{12} \\ O_{21} & O_{22} \end{bmatrix} = \text{Convolution}\left(\begin{bmatrix} X_{11} & X_{12} & X_{13} \\ X_{21} & X_{22} & X_{23} \\ X_{31} & X_{32} & X_{33} \end{bmatrix}, \begin{bmatrix} F_{11} & F_{12} \\ F_{21} & F_{22} \end{bmatrix} \right)$$

$$O_{11} = F_{11}X_{11} + F_{12}X_{12} + F_{21}X_{21} + F_{22}X_{22}$$

$$O_{12} = F_{11}X_{12} + F_{12}X_{13} + F_{21}X_{22} + F_{22}X_{23}$$

$$O_{21} = F_{11}X_{21} + F_{12}X_{22} + F_{21}X_{31} + F_{22}X_{32}$$

$$O_{22} = F_{11}X_{22} + F_{12}X_{23} + F_{21}X_{32} + F_{22}X_{33}$$

$$\frac{\partial E}{\partial F_{11}} = \frac{\partial E}{\partial O_{11}}\frac{\partial O_{11}}{\partial F_{11}} + \frac{\partial E}{\partial O_{12}}\frac{\partial O_{12}}{\partial F_{11}} + \frac{\partial E}{\partial O_{21}}\frac{\partial O_{21}}{\partial F_{11}} + \frac{\partial E}{\partial O_{22}}\frac{\partial O_{22}}{\partial F_{11}}$$

$$\frac{\partial E}{\partial F_{12}} = \frac{\partial E}{\partial O_{11}}\frac{\partial O_{11}}{\partial F_{12}} + \frac{\partial E}{\partial O_{12}}\frac{\partial O_{12}}{\partial F_{12}} + \frac{\partial E}{\partial O_{21}}\frac{\partial O_{21}}{\partial F_{12}} + \frac{\partial E}{\partial O_{22}}\frac{\partial O_{22}}{\partial F_{12}}$$

$$\frac{\partial E}{\partial F_{21}} = \frac{\partial E}{\partial O_{11}}\frac{\partial O_{11}}{\partial F_{21}} + \frac{\partial E}{\partial O_{12}}\frac{\partial O_{12}}{\partial F_{21}} + \frac{\partial E}{\partial O_{21}}\frac{\partial O_{21}}{\partial F_{21}} + \frac{\partial E}{\partial O_{22}}\frac{\partial O_{22}}{\partial F_{21}}$$

$$\frac{\partial E}{\partial F_{22}} = \frac{\partial E}{\partial O_{11}}\frac{\partial O_{11}}{\partial F_{22}} + \frac{\partial E}{\partial O_{12}}\frac{\partial O_{12}}{\partial F_{22}} + \frac{\partial E}{\partial O_{21}}\frac{\partial O_{21}}{\partial F_{22}} + \frac{\partial E}{\partial O_{22}}\frac{\partial O_{22}}{\partial F_{22}}$$

Slide taken from Forward And Backpropagation in Convolutional Neural Network. - Medium

Backprop in Convolution

$$\begin{bmatrix} O_{11} & O_{12} \\ O_{21} & O_{22} \end{bmatrix} = \text{Convolution}\left(\begin{bmatrix} X_{11} & X_{12} & X_{13} \\ X_{21} & X_{22} & X_{23} \\ X_{31} & X_{32} & X_{33} \end{bmatrix}, \begin{bmatrix} F_{11} & F_{12} \\ F_{21} & F_{22} \end{bmatrix} \right)$$

Which evaluates to-

$O_{11} = F_{11}X_{11} + F_{12}X_{12} + F_{21}X_{21} + F_{22}X_{22}$

$O_{12} = F_{11}X_{12} + F_{12}X_{13} + F_{21}X_{22} + F_{22}X_{23}$

$O_{21} = F_{11}X_{21} + F_{12}X_{22} + F_{21}X_{31} + F_{22}X_{32}$

$O_{22} = F_{11}X_{22} + F_{12}X_{23} + F_{21}X_{32} + F_{22}X_{33}$

$$\frac{\partial E}{\partial F_{11}} = \frac{\partial E}{\partial O_{11}} X_{11} + \frac{\partial E}{\partial O_{12}} X_{12} + \frac{\partial E}{\partial O_{21}} X_{21} + \frac{\partial E}{\partial O_{22}} X_{22}$$

$$\frac{\partial E}{\partial F_{12}} = \frac{\partial E}{\partial O_{11}} X_{12} + \frac{\partial E}{\partial O_{12}} X_{13} + \frac{\partial E}{\partial O_{21}} X_{22} + \frac{\partial E}{\partial O_{22}} X_{23}$$

$$\frac{\partial E}{\partial F_{21}} = \frac{\partial E}{\partial O_{11}} X_{21} + \frac{\partial E}{\partial O_{12}} X_{22} + \frac{\partial E}{\partial O_{21}} X_{31} + \frac{\partial E}{\partial O_{22}} X_{32}$$

$$\frac{\partial E}{\partial F_{22}} = \frac{\partial E}{\partial O_{11}} X_{22} + \frac{\partial E}{\partial O_{12}} X_{23} + \frac{\partial E}{\partial O_{21}} X_{32} + \frac{\partial E}{\partial O_{22}} X_{33}$$

Slide taken from Forward And Backpropagation in Convolutional Neural Network. - Medium

Backprop in Convolution

If we look closely the previous equation can be written in form of our convolution operation.

$$\begin{bmatrix} \partial E/\partial F_{11} & \partial E/\partial F_{12} \\ \partial E/\partial F_{21} & \partial E/\partial F_{22} \end{bmatrix} = \textbf{Convolution}\left(\begin{bmatrix} X_{11} & X_{12} & X_{13} \\ X_{21} & X_{22} & X_{23} \\ X_{31} & X_{32} & X_{33} \end{bmatrix}, \begin{bmatrix} \partial E/\partial O_{11} & \partial E/\partial O_{12} \\ \partial E/\partial O_{21} & \partial E/\partial O_{22} \end{bmatrix} \right)$$

$$\frac{\partial E}{\partial F_{11}} = \frac{\partial E}{\partial O_{11}} X_{11} + \frac{\partial E}{\partial O_{12}} X_{12} + \frac{\partial E}{\partial O_{21}} X_{21} + \frac{\partial E}{\partial O_{22}} X_{22}$$

$$\frac{\partial E}{\partial F_{12}} = \frac{\partial E}{\partial O_{11}} X_{12} + \frac{\partial E}{\partial O_{12}} X_{13} + \frac{\partial E}{\partial O_{21}} X_{22} + \frac{\partial E}{\partial O_{22}} X_{23}$$

$$\frac{\partial E}{\partial F_{21}} = \frac{\partial E}{\partial O_{11}} X_{21} + \frac{\partial E}{\partial O_{12}} X_{22} + \frac{\partial E}{\partial O_{21}} X_{31} + \frac{\partial E}{\partial O_{22}} X_{32}$$

$$\frac{\partial E}{\partial F_{22}} = \frac{\partial E}{\partial O_{11}} X_{22} + \frac{\partial E}{\partial O_{12}} X_{23} + \frac{\partial E}{\partial O_{21}} X_{32} + \frac{\partial E}{\partial O_{22}} X_{33}$$

Carnegie Mellon University

Slide taken from Forward And Backpropagation in Convolutional Neural Network. - Medium

Backprop in Convolution

Similarly we can find the gradients of the error ‘E’ with respect to the input matrix ‘X’.

$$\frac{\partial E}{\partial X_{11}} = \frac{\partial E}{\partial O_{11}} F_{11} + \frac{\partial E}{\partial O_{12}} 0 + \frac{\partial E}{\partial O_{21}} 0 + \frac{\partial E}{\partial O_{22}} 0$$

$$\frac{\partial E}{\partial X_{12}} = \frac{\partial E}{\partial O_{11}} F_{12} + \frac{\partial E}{\partial O_{12}} F_{11} + \frac{\partial E}{\partial O_{21}} 0 + \frac{\partial E}{\partial O_{22}} 0$$

$$\frac{\partial E}{\partial X_{13}} = \frac{\partial E}{\partial O_{11}} 0 + \frac{\partial E}{\partial O_{12}} F_{12} + \frac{\partial E}{\partial O_{21}} 0 + \frac{\partial E}{\partial O_{22}} 0$$

$$\frac{\partial E}{\partial X_{21}} = \frac{\partial E}{\partial O_{11}} F_{21} + \frac{\partial E}{\partial O_{12}} 0 + \frac{\partial E}{\partial O_{21}} F_{11} + \frac{\partial E}{\partial O_{22}} 0$$

$$\frac{\partial E}{\partial X_{22}} = \frac{\partial E}{\partial O_{11}} F_{22} + \frac{\partial E}{\partial O_{12}} F_{21} + \frac{\partial E}{\partial O_{21}} f_{12} + \frac{\partial E}{\partial O_{22}} F_{11}$$

$$\frac{\partial E}{\partial X_{23}} = \frac{\partial E}{\partial O_{11}} 0 + \frac{\partial E}{\partial O_{12}} F_{22} + \frac{\partial E}{\partial O_{21}} 0 + \frac{\partial E}{\partial O_{22}} F_{11}$$

$$\frac{\partial E}{\partial X_{31}} = \frac{\partial E}{\partial O_{11}} 0 + \frac{\partial E}{\partial O_{12}} 0 + \frac{\partial E}{\partial O_{21}} F_{21} + \frac{\partial E}{\partial O_{22}} 0$$

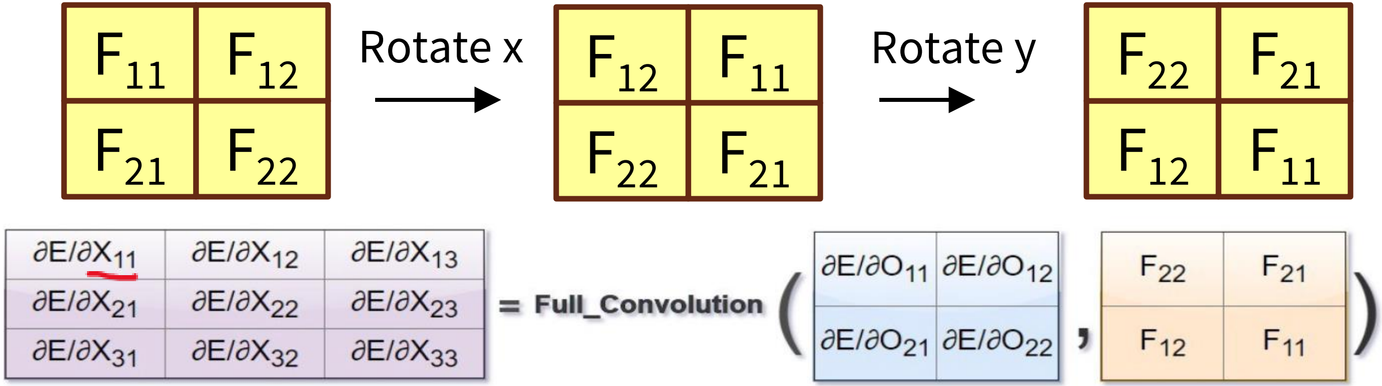
$$\frac{\partial E}{\partial X_{32}} = \frac{\partial E}{\partial O_{11}} 0 + \frac{\partial E}{\partial O_{12}} 0 + \frac{\partial E}{\partial O_{21}} F_{22} + \frac{\partial E}{\partial O_{22}} F_{21}$$

$$\frac{\partial E}{\partial X_{33}} = \frac{\partial E}{\partial O_{11}} 0 + \frac{\partial E}{\partial O_{12}} 0 + \frac{\partial E}{\partial O_{21}} 0 + \frac{\partial E}{\partial O_{22}} F_{22}$$

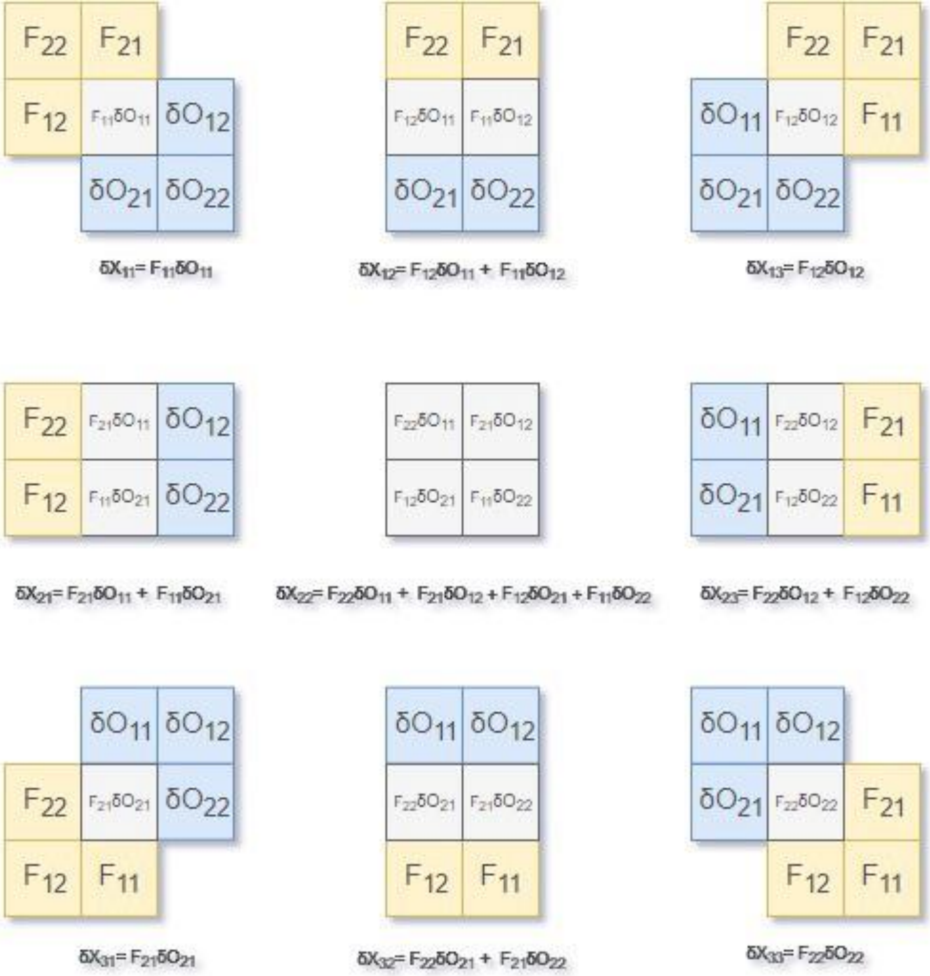
Slide taken from Forward And Backpropagation in Convolutional Neural Network. - Medium

The previous computation can be obtained by a different type of convolution operation known as full convolution.

In order to obtain the gradients of the input matrix we need to rotate the filter by 180 degree and calculate the full convolution of the rotated filter by the gradients of the output with respect to error.



Carnegie Mellon University



Carnegie Mellon University

Slide taken from Forward And Backpropagation in Convolutional Neural Network. - Medium

Backpropagation of max pooling

Suppose you have a matrix M of four elements:

a	b
c	d

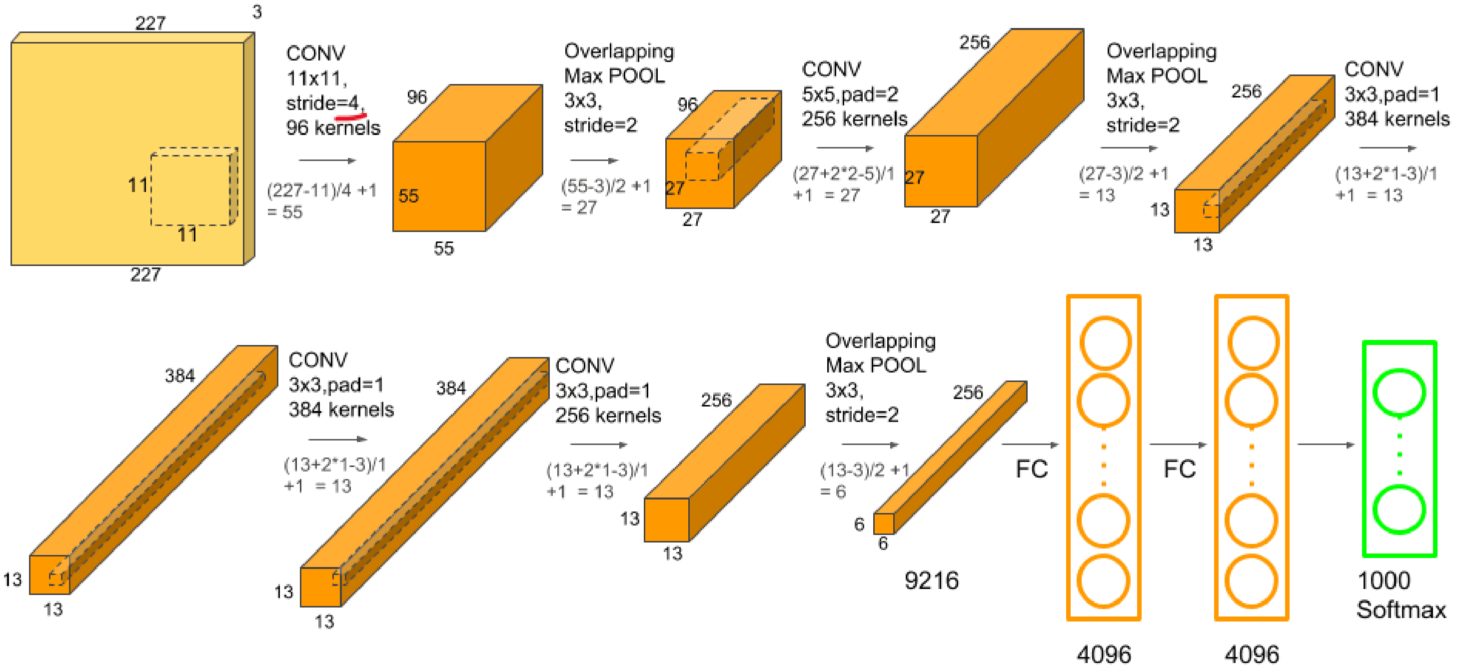
and maxpool(M) returns **d**.

Then, the maxpool function really only depends on d. So, the derivative of maxpool relative to d is 1, and its derivative relative to a,b,c is zero. So you backpropagate 1 to the unit corresponding to d, and you backpropagate zero for the other units.

Carnegie Mellon University

Slide taken from Forward And Backpropagation in Convolutional Neural Network. - Medium

AlexNet

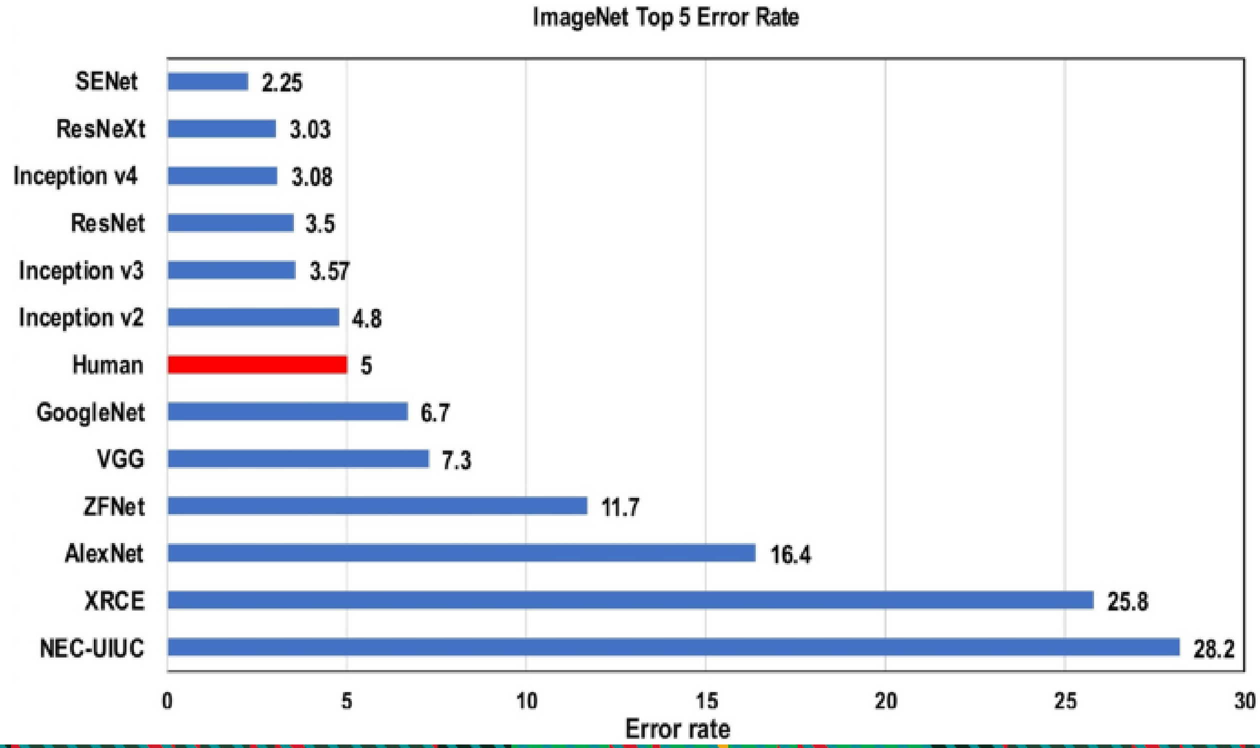


Alex Krizhevsky, ImageNet Classification with Deep Convolutional Neural Networks, 2012

Carnegie Mellon University

Different CNN Architectures

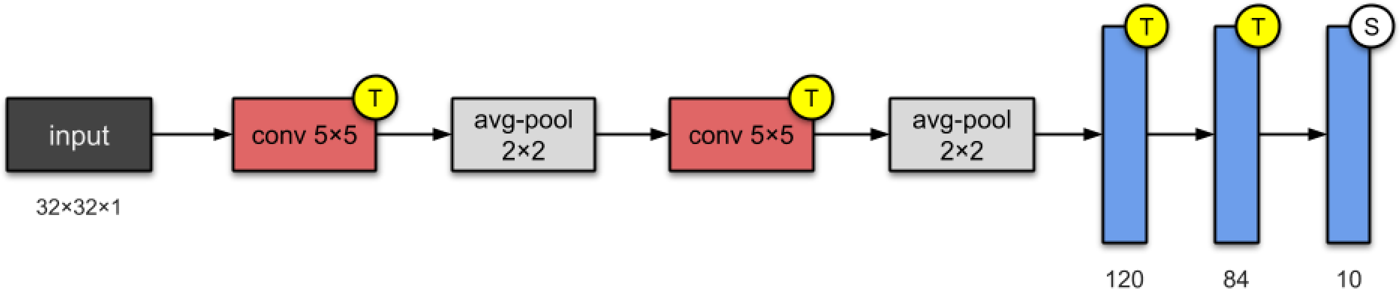
Notable CNN models[19]



Carnegie Mellon University

Notable CNN models

LENET (1998)[8]



(2) ENDING THE NETWORK WITH ONE OR MORE FULLY-CONNECTED LAYERS

LeNet-5

Gradient Based Learning Applied To Document Recognition
- Y. Lecun, L. Bottou, Y. Bengio, P. Haffner; 1998

Helped establish how we use CNNs today

Replaced manual feature

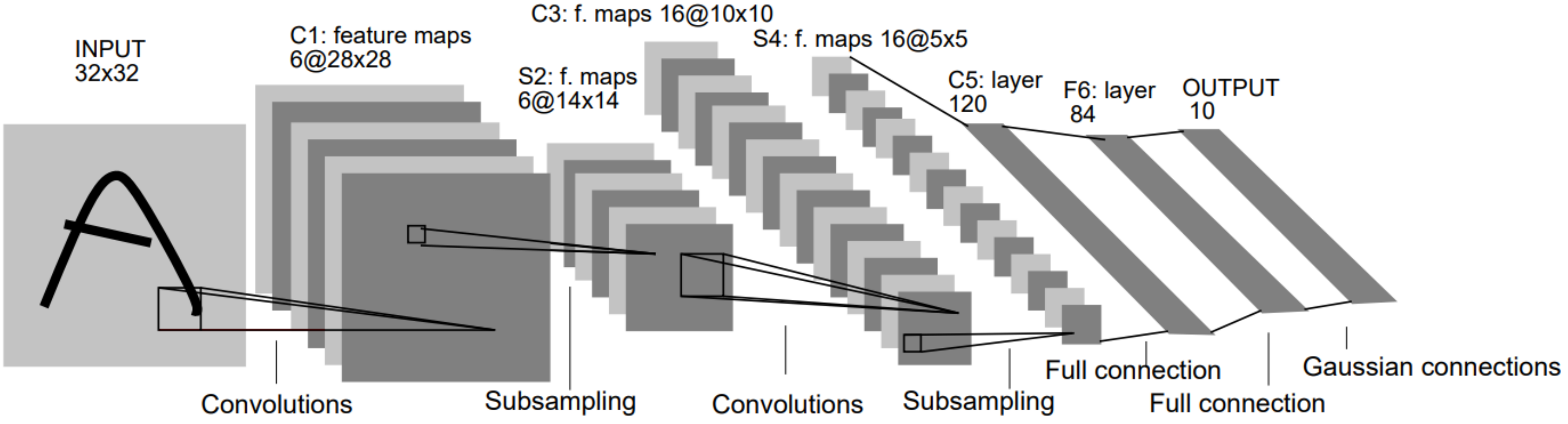
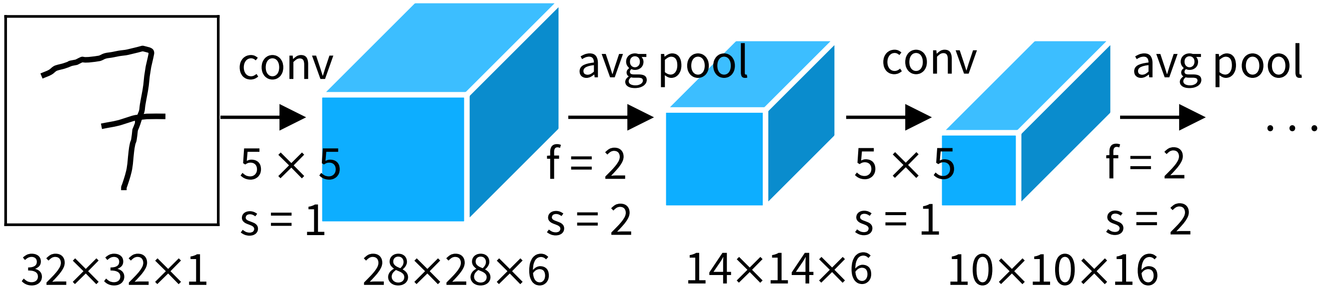


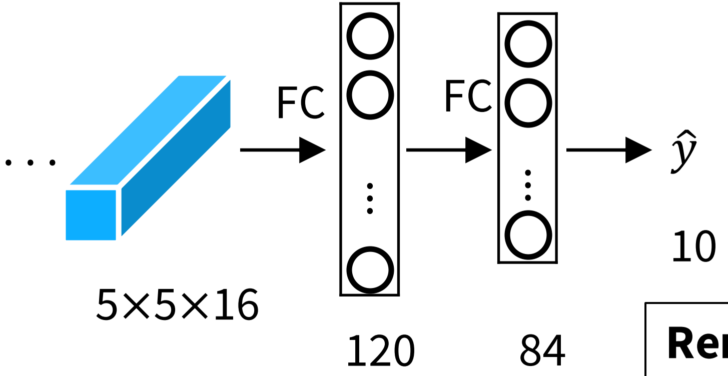
Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

Carnegie Mellon University

[**LeCun et al., 1998**]

LeNet-5





Only 60K parameters

As we go deeper in the network: $N_H \downarrow, \; N_W \downarrow, \; N_C \uparrow$

General structure:

conv->pool->conv->pool->FC->FC->output

Different filters look at different channels

Sigmoid and Tanh nonlinearity

Reminder:
Output size = (N+2P-F)/stride + 1

This slide is taken from Andrew Ng

[LeCun et al., 1998]

AlexNet

ImageNet Classification with Deep Convolutional Neural Networks

- *Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton; 2012*

Facilitated by GPUs, highly optimized convolution implementation and large datasets (ImageNet)

One of the largest CNNs to date

Has 60 Million parameter compared to 60k parameter of LeNet-5

[**Krizhevsky et al**., **2012**]

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners

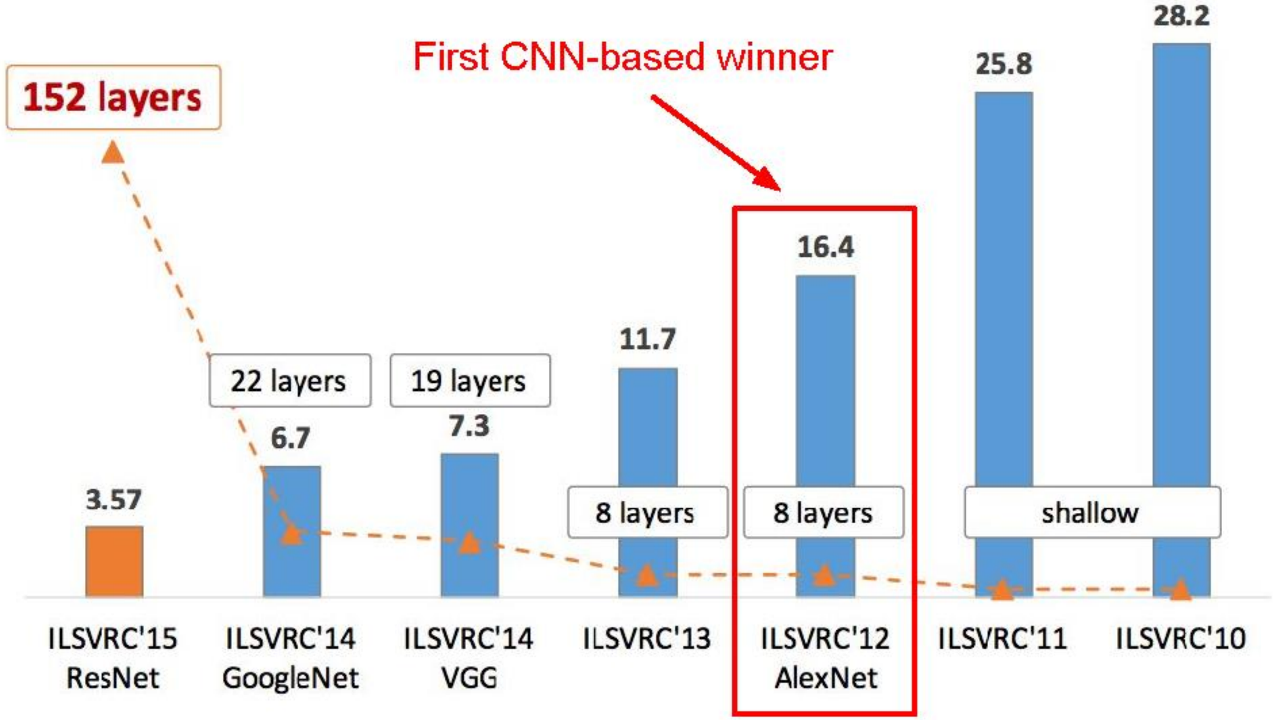
The annual "Olympics" of computer vision.

Teams from across the world compete to see who has the best computer vision model for tasks such as classification, localization, detection, and more.

2012 marked **the first year where a CNN was used** to achieve a top 5 test error rate of 15.3%.

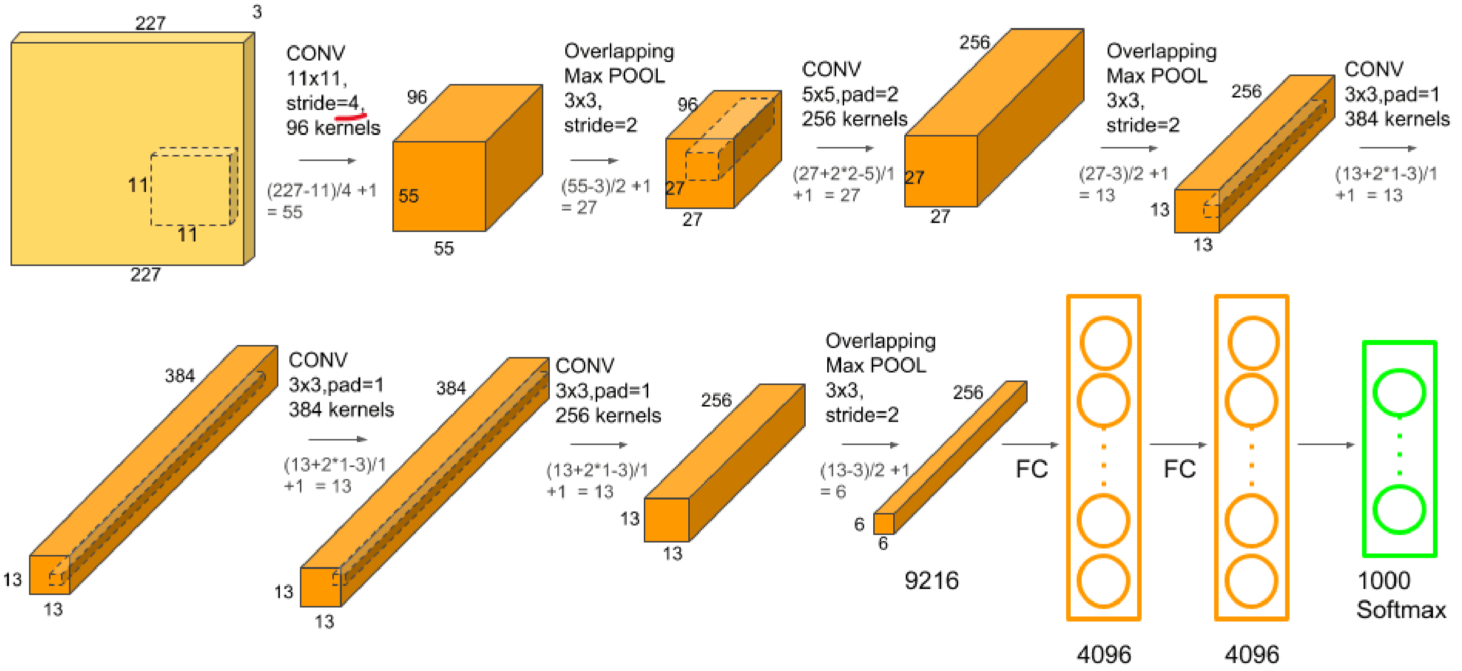
The next best entry achieved an error of 26.2%.

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



Slide taken from Fei-Fei & Justin Johnson & Serena Yeung. Lecture 9.

AlexNet



Alex Krizhevsky, ImageNet Classification with Deep Convolutional Neural Networks, 2012

AlexNet

Architectur
eCONV1
MAX POOL1
NORM1
CONV2
MAX POOL2
NORM2
CONV3
CONV4
CONV5
Max POOL3
FC6
FC7
FC8

INPUT: 227X227X3 IMAGES (224X224 BEFORE PADDING)

FIRST LAYER: 96 11X11 FILTERS APPLIED AT STRIDE 4

OUTPUT VOLUME SIZE?

(N-F)/s+1 = (227-11)/4+1 = 55 -> [55x55x96]

NUMBER OF PARAMETERS IN THIS LAYER?

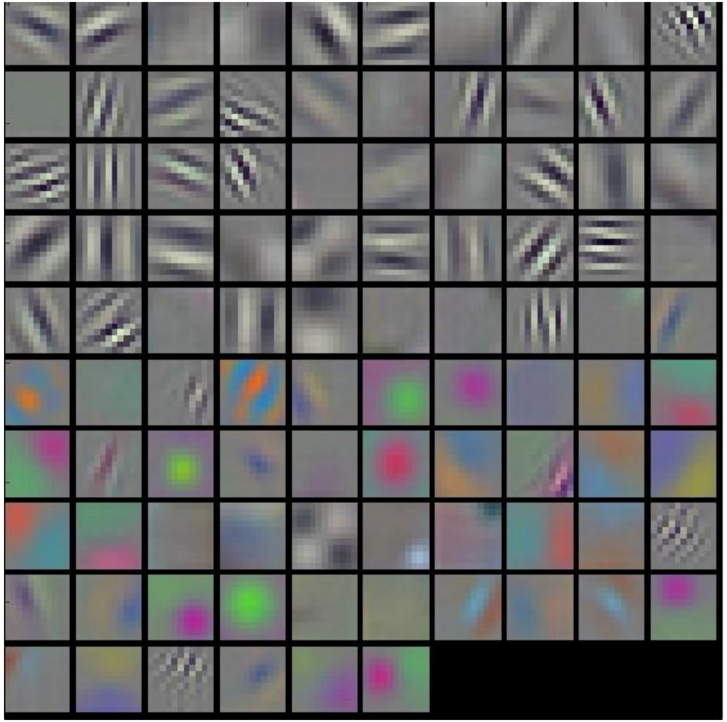
(11*11*3)*96 = 35K

Carnegie Mellon University

Slide taken from Fei-Fei & Justin Johnson & Serena Yeung. Lecture 9.

[**Krizhevsky et al., 2012**]

AlexNet filters



[Krizhevsky et al., 2012]

AlexNet

Architecture
CONV1
MAX POOL1
NORM1
CONV2
MAX POOL2
NORM2
CONV3
CONV4
CONV5
Max POOL3
FC6
FC7
FC8

INPUT: 227X227X3 IMAGES (224X224 BEFORE PADDING)

AFTER CONV1: 55X55X96

SECOND LAYER: 3X3 FILTERS APPLIED AT STRIDE 2

OUTPUT VOLUME SIZE?

(N-F)/s+1 = (55-3)/2+1 = 27 -> [27x27x96]

NUMBER OF PARAMETERS IN THIS LAYER?

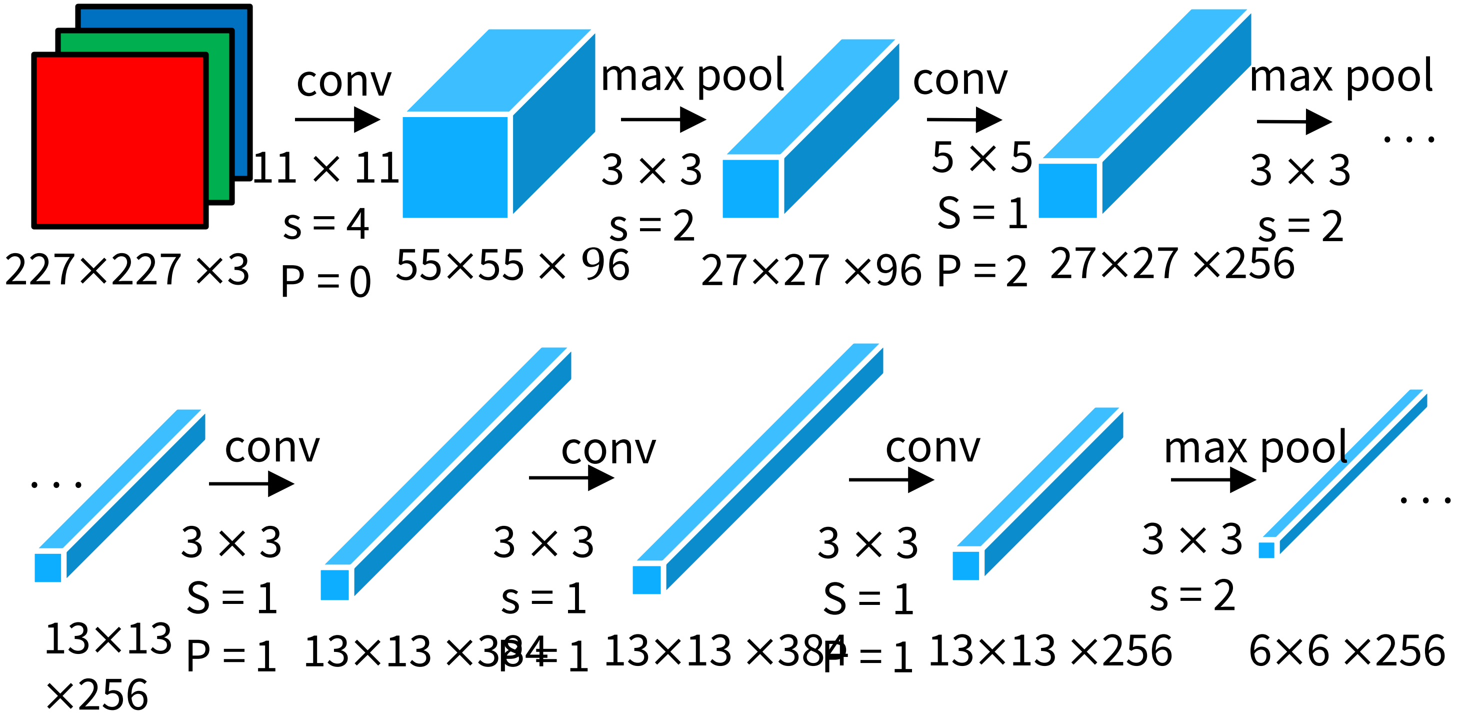
0!

Carnegie Mellon University

Slide taken from Fei-Fei & Justin Johnson & Serena Yeung. Lecture 9.

[Krizhevsky et al., 2012]

AlexNet

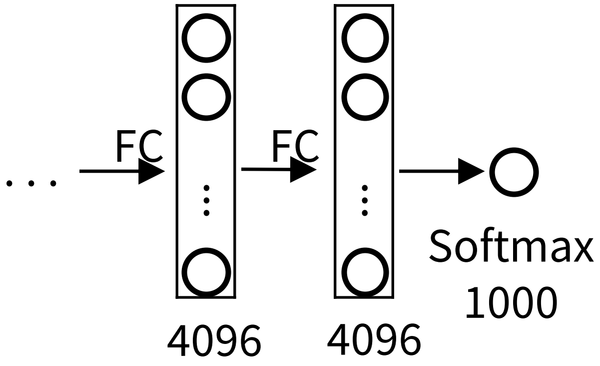


Carnegie Mellon University

This slide is taken from Andrew Ng

[Krizhevsky et al., 2012]

AlexNet



This slide is taken from Andrew Ng

[Krizhevsky et al., 2012]

DETAILS/RETROSPECTIVES:

- FIRST USE OF RELU
- USED NORM LAYERS (NOT COMMON ANYMORE)
- HEAVY DATA AUGMENTATION
- DROPOUT 0.5
- BATCH SIZE 128
- 7 CNN ENSEMBLE

Carnegie Mellon University

[Krizhevsky et al., 2012]

AlexNet

Trained on GTX 580 GPU with only 3 GB of memory.

Network spread across 2 GPUs, half the neurons (feature maps) on each GPU.

CONV1, CONV2, CONV4, CONV5:
Connections only with feature maps on same GPU.

CONV3, FC6, FC7, FC8:
Connections with all feature maps in preceding layer, communication across GPUs.

Carnegie Mellon University

[**Krizhevsky et al., 2012**]

AlexNet

AlexNet was the coming out party for CNNs in the computer vision community. This was **the first time a model performed so well on a historically difficult ImageNet dataset.** This paper illustrated the benefits of CNNs and backed them up with record breaking performance in the competition.

[**Krizhevsky et al., 2012**]

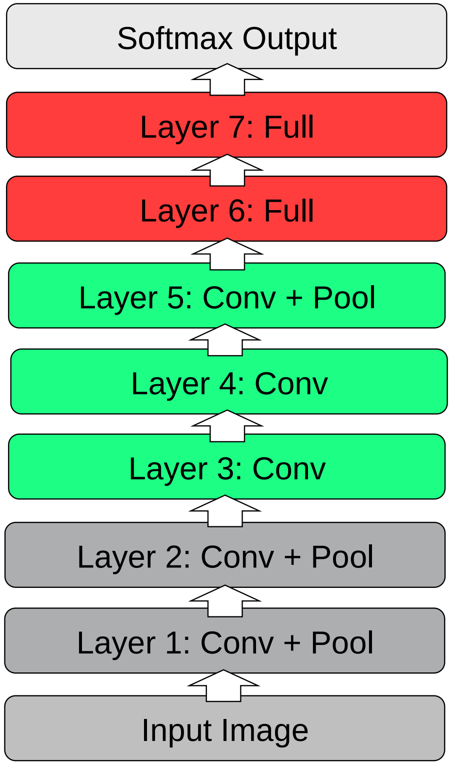
AlexNet

Model	**Top-1**	**Top-5**
Sparse coding [2]	*47.1%*	*28.2%*
SIFT + FVs [24]	*45.7%*	*25.7%*
CNN	**37.5%**	**17.0%**

Alex Krizhevsky, ImageNet Classification with Deep Convolutional Neural Networks, 2012

AlexNet

- 8 layers total!
- Trained on Imagenet dataset [Deng et al. CVPR'09]!
- 18.2% top-5 error !

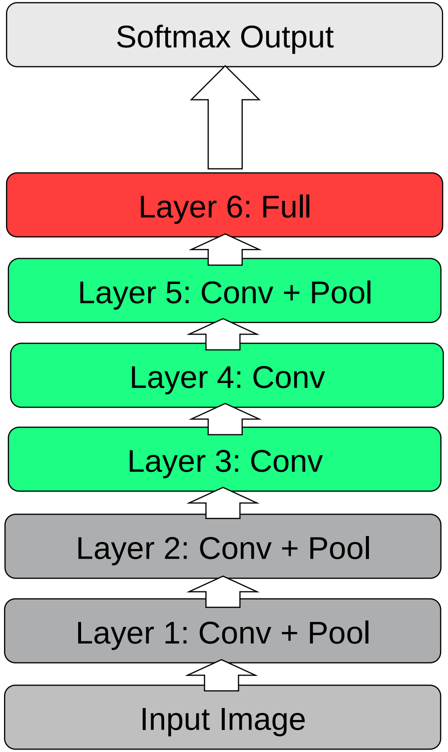


[From Rob Fergus' CIFAR 2016 tutorial] Krizhevsky et al., NIPS 2012

AlexNet

AlexNet

- Remove top fully connected layer 7 !
- Drop ~16 million parameters!
- Only 1.1% drop in performance!!



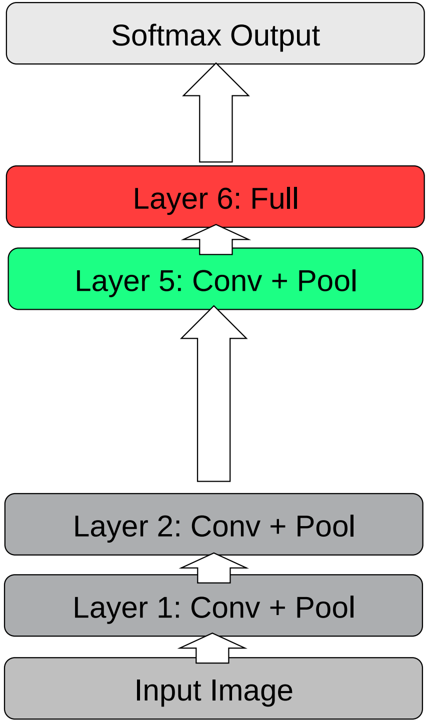
[From Rob Fergus' CIFAR 2016 tutorial]   Krizhevsky et al., NIPS 2012

AlexNet

AlexNet

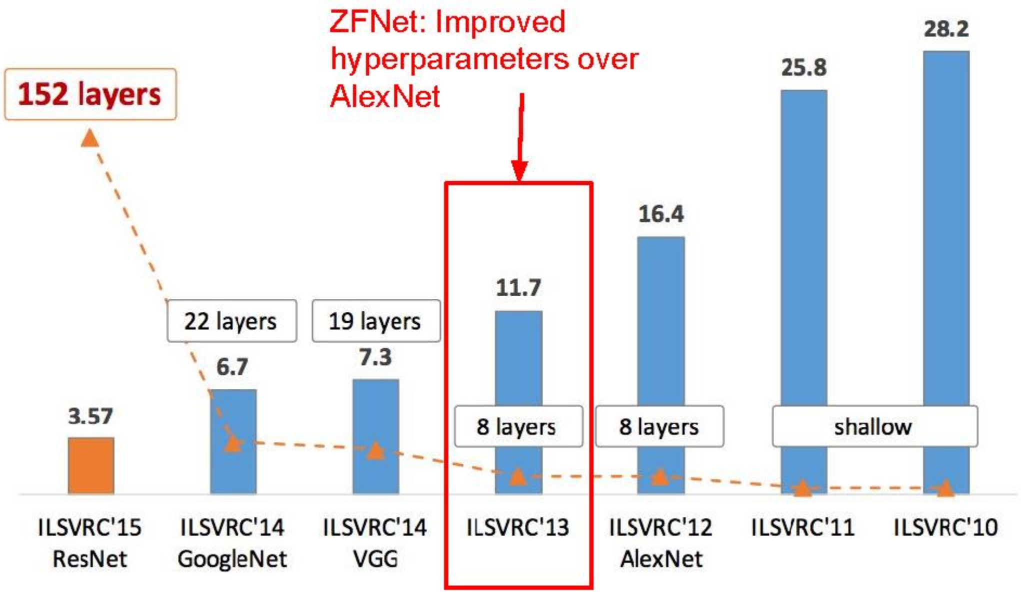
- Remove layers 3 4,6 and 7 !
- Drop ~50 million parameters!
- **33.5%** drop in performance!!

- Depth of the network is the key!



[From Rob Fergus' CIFAR 2016 tutorial] Krizhevsky et al., NIPS 2012

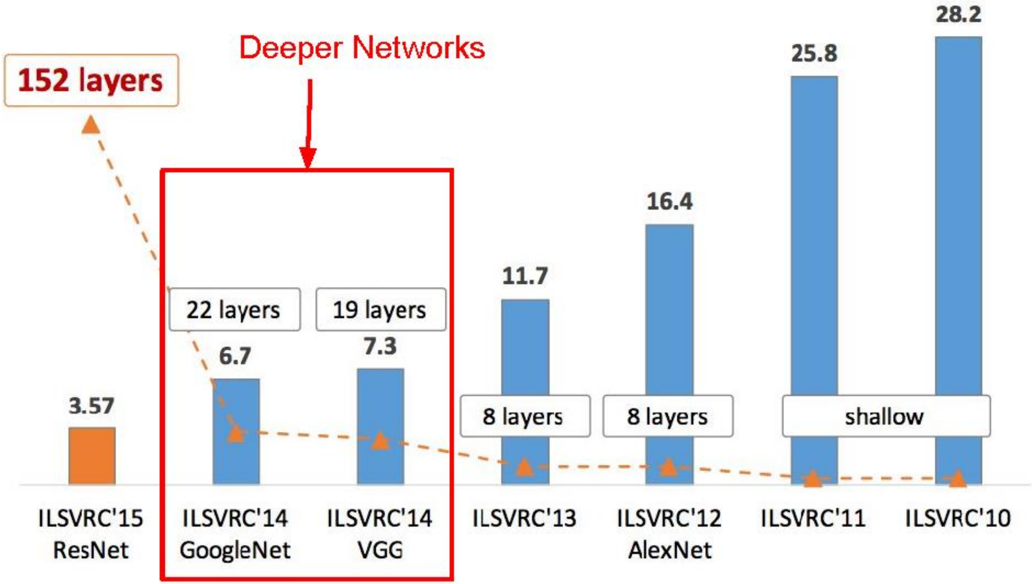
ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



Carnegie Mellon University

Slide taken from Fei-Fei & Justin Johnson & Serena Yeung. Lecture 9.

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



Slide taken from Fei-Fei & Justin Johnson & Serena Yeung. Lecture 9.

VGGNet

- *Very Deep Convolutional Networks For Large Scale Image Recognition - Karen Simonyan and Andrew Zisserman; 2015*
- The runner-up at the ILSVRC 2014 competition
- Significantly deeper than AlexNet
- 140 million parameters

Carnegie Mellon University

[Simonyan and Zisserman, 2014]

VGGNet

Input
3x3 conv, 64
3x3 conv, 64
Pool 1/2
3x3 conv, 128
3x3 conv, 128
Pool 1/2
3x3 conv, 256
3x3 conv, 256
Pool 1/2
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
Pool 1/2
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
Pool 1/2
FC 4096
FC 4096
FC 1000
Softmax

SMALLER FILTERS

- ONLY 3X3 CONV FILTERS, STRIDE 1, PAD 1
- AND 2X2 MAX POOL , STRIDE 2

DEEPER NETWORK

- ALEXNET: 8 LAYERS
- VGGNET: 16 - 19 LAYERS

ZFNET: 11.7% TOP 5 ERROR IN ILSVRC'13

VGGNET: 7.3% TOP 5 ERROR IN ILSVRC'14

Slide taken from Fei-Fei & Justin Johnson & Serena Yeung. Lecture 9.

[Simonyan and Zisserman, 2014]

VGGNet

WHY USE SMALLER FILTERS? (3X3 CONV)

STACK OF THREE 3X3 CONV (STRIDE 1) LAYERS HAS THE SAME EFFECTIVE RECEPTIVE FIELD AS ONE 7X7 CONV LAYER.

WHAT IS THE EFFECTIVE RECEPTIVE FIELD OF THREE 3X3 CONV (STRIDE 1) LAYERS?

7X7

BUT DEEPER, MORE NON-LINEARITIES

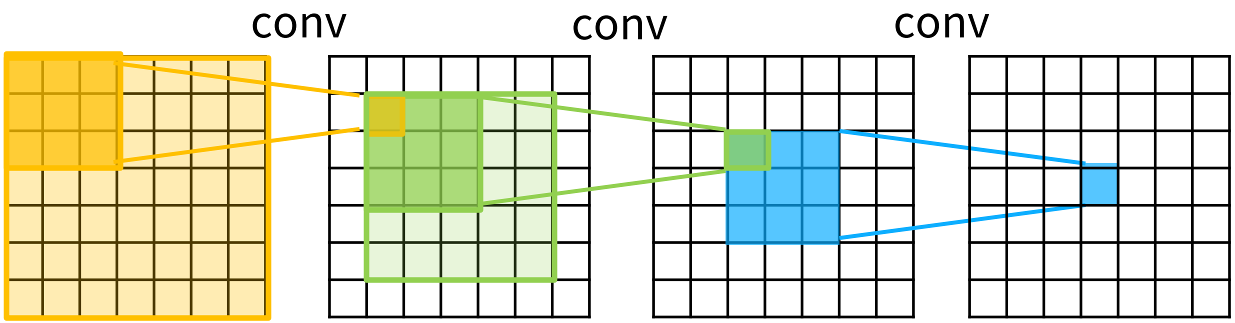
AND FEWER PARAMETERS: $3 * (3^2C^2)$ VS. 7^2C^2 FOR C CHANNELS PER LAYER

Carnegie Mellon University

Slide taken from Fei-Fei & Justin Johnson & Serena Yeung. Lecture 9.

[Simonyan and Zisserman, 2014]

Reminder: Receptive Field



Carnegie Mellon University

Input	memory: 224*224*3=150K	params: 0
3x3 conv, 64	memory: 224*224*64=3.2M	params: (3*3*3)*64 = 1,728
3x3 conv, 64	memory: 224*224*64=3.2M	params: (3*3*64)*64 = 36,864
Pool	memory: 112*112*64=800K	params: 0
3x3 conv, 128	memory: 112*112*128=1.6M	params: (3*3*64)*128 = 73,728
3x3 conv, 128	memory: 112*112*128=1.6M	params: (3*3*128)*128 = 147,456
Pool	memory: 56*56*128=400K	params: 0
3x3 conv, 256	memory: 56*56*256=800K	params: (3*3*128)*256 = 294,912
3x3 conv, 256	memory: 56*56*256=800K	params: (3*3*256)*256 = 589,824
3x3 conv, 256	memory: 56*56*256=800K	params: (3*3*256)*256 = 589,824
Pool	memory: 28*28*256=200K	params: 0
3x3 conv, 512	memory: 28*28*512=400K	params: (3*3*256)*512 = 1,179,648
3x3 conv, 512	memory: 28*28*512=400K	params: (3*3*512)*512 = 2,359,296
3x3 conv, 512	memory: 28*28*512=400K	params: (3*3*512)*512 = 2,359,296
Pool	memory: 14*14*512=100K	params: 0
3x3 conv, 512	memory: 14*14*512=100K	params: (3*3*512)*512 = 2,359,296
3x3 conv, 512	memory: 14*14*512=100K	params: (3*3*512)*512 = 2,359,296
3x3 conv, 512	memory: 14*14*512=100K	params: (3*3*512)*512 = 2,359,296
Pool	memory: 7*7*512=25K	params: 0
FC 4096	memory: 4096	params: 7*7*512*4096 = 102,760,448
FC 4096	memory: 4096	params: 4096*4096 = 16,777,216
FC 1000	memory: 1000	params: 4096*1000 = 4,096,000

Carnegie Mellon University

Slide taken from Fei-Fei & Justin Johnson & Serena Yeung. Lecture 9.

[Simonyan and Zisserman, 2014]

VGGNet

Input
3x3 conv, 64
3x3 conv, 64
Pool
3x3 conv, 128
3x3 conv, 128
Pool
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
Pool
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
Pool
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
Pool
FC 4096
FC 4096
FC 1000
Softmax

VGG16:

TOTAL MEMORY: 24M * 4 BYTES ~= 96MB / IMAGE

TOTAL PARAMS: 138M PARAMETERS

[Simonyan and Zisserman, 2014]

Input	memory: 224*224*3=150K	params: 0
3x3 conv, 64	memory: $\boxed{224*224*64=3.2M}$	params: (3*3*3)*64 = 1,728
3x3 conv, 64	memory: 224*224*64=3.2M	params: (3*3*64)*64 = 36,864
Pool	memory: 112*112*64=800K	params: 0
3x3 conv, 128	memory: 112*112*128=1.6M	params: (3*3*64)*128 = 73,728
3x3 conv, 128	memory: 112*112*128=1.6M	params: (3*3*128)*128 = 147,456
Pool	memory: 56*56*128=400K	params: 0
3x3 conv, 256	memory: 56*56*256=800K	params: (3*3*128)*256 = 294,912
3x3 conv, 256	memory: 56*56*256=800K	params: (3*3*256)*256 = 589,824
3x3 conv, 256	memory: 56*56*256=800K	params: (3*3*256)*256 = 589,824
Pool	memory: 28*28*256=200K	params: 0
3x3 conv, 512	memory: 28*28*512=400K	params: (3*3*256)*512 = 1,179,648
3x3 conv, 512	memory: 28*28*512=400K	params: (3*3*512)*512 = 2,359,296
3x3 conv, 512	memory: 28*28*512=400K	params: (3*3*512)*512 = 2,359,296
Pool	memory: 14*14*512=100K	params: 0
3x3 conv, 512	memory: 14*14*512=100K	params: (3*3*512)*512 = 2,359,296
3x3 conv, 512	memory: 14*14*512=100K	params: (3*3*512)*512 = 2,359,296
3x3 conv, 512	memory: 14*14*512=100K	params: (3*3*512)*512 = 2,359,296
Pool	memory: 7*7*512=25K	params: 0
FC 4096	memory: 4096	$\boxed{\text{params: 7*7*512*4096 = 102,760,448}}$
FC 4096	memory: 4096	params: 4096*4096 = 16,777,216
FC 1000	memory: 1000	params: 4096*1000 = 4,096,000

Carnegie Mellon University

Slide taken from Fei-Fei & Justin Johnson & Serena Yeung. Lecture 9.

[Simonyan and Zisserman, 2014]

VGGNet

Details/Retrospectives :

ILSVRC'14 2nd in classification, 1st in localization

Similar training procedure as AlexNet

No Local Response Normalisation (LRN)

Use VGG16 or VGG19 (VGG19 only slightly better, more memory)

Use ensembles for best results

FC7 features generalize well to other tasks

Trained on 4 Nvidia Titan Black GPUs for **two to three weeks.**

Carnegie Mellon University

[Simonyan and Zisserman, 2014]

VGGNet

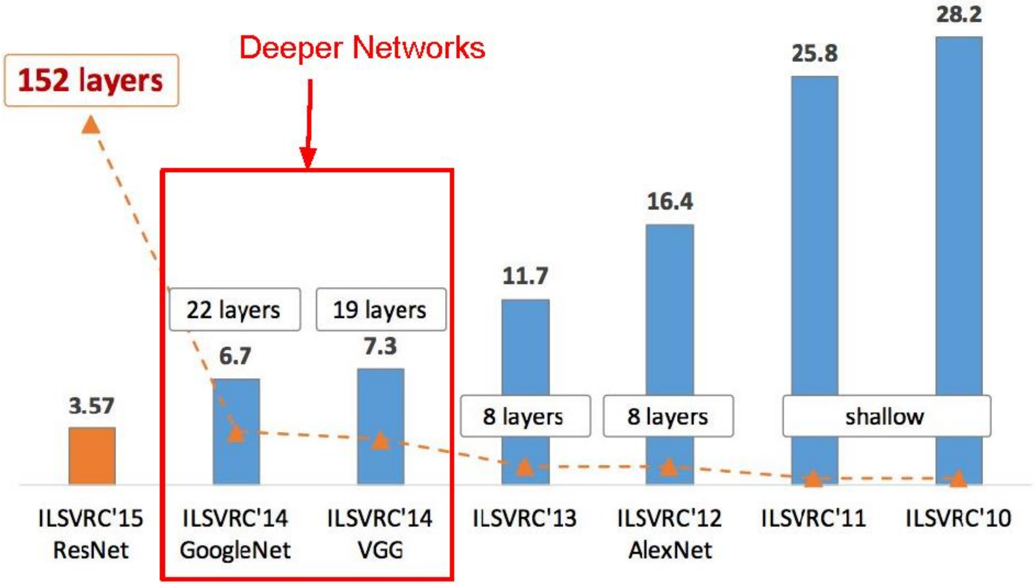
VGG Net reinforced the notion that **convolutional neural networks have to have a deep network of layers in order for this hierarchical representation of visual data to work.**

Keep it deep.

Keep it simple.

[Simonyan and Zisserman, 2014]

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



Slide taken from Fei-Fei & Justin Johnson & Serena Yeung. Lecture 9.

GoogleNet

Going Deeper with Convolutions - Christian Szegedy et al.; 2015

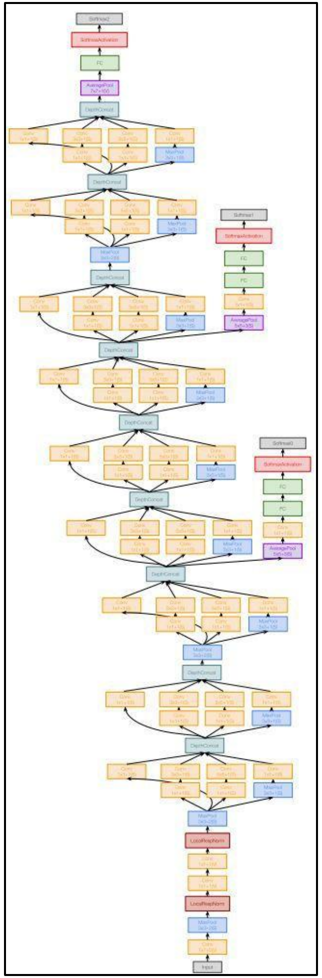
ILSVRC 2014 competition winner

Also significantly deeper than AlexNet

x12 less parameters than AlexNet

Focused on computational efficiency

[Szegedy et al., 2014]



GoogleNet

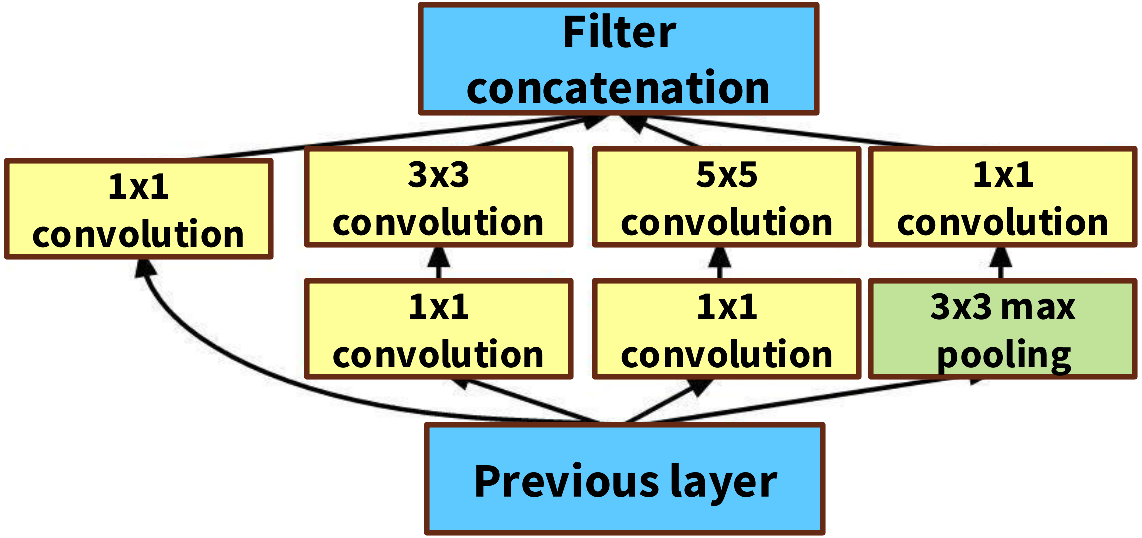
- 22 layers
- Efficient **“Inception” module** - strayed from the general approach of simply stacking conv and pooling layers on top of each other in a sequential structure
- No FC layers
- Only 5 million parameters!
- ILSVRC’14 classification winner (6.7% top 5 error)

Carnegie Mellon University

[Szegedy et al., 2014]

GoogleNet

"INCEPTION MODULE": DESIGN A GOOD LOCAL NETWORK TOPOLOGY (NETWORK WITHIN A NETWORK) AND THEN STACK THESE MODULES ON TOP OF EACH OTHER



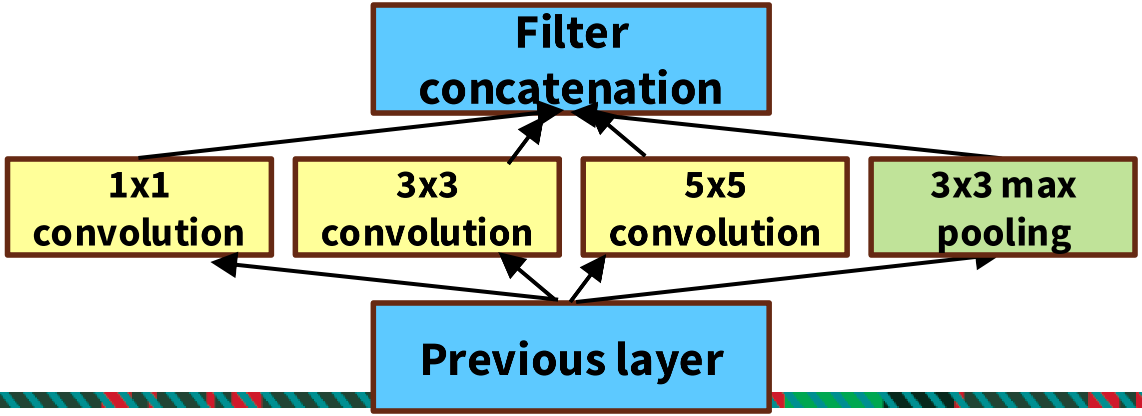
Carnegie Mellon University

[Szegedy et al., 2014]

GoogleNet

Naïve Inception Model

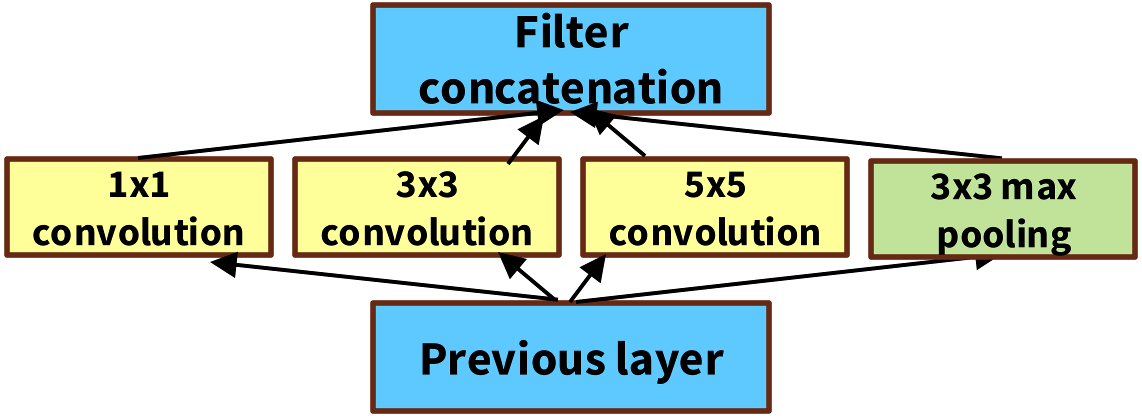
- Apply parallel filter operations on the input :
  - Multiple receptive field sizes for convolution (1x1, 3x3, 5x5)
  - Pooling operation (3x3)
- Concatenate all filter outputs together depth-wise



[Szegedy et al., 2014]

GoogleNet

- **What's the problem with this?**
  High computational complexity



[Szegedy et al., 2014]

GoogleNet

OUTPUT VOLUME SIZES:

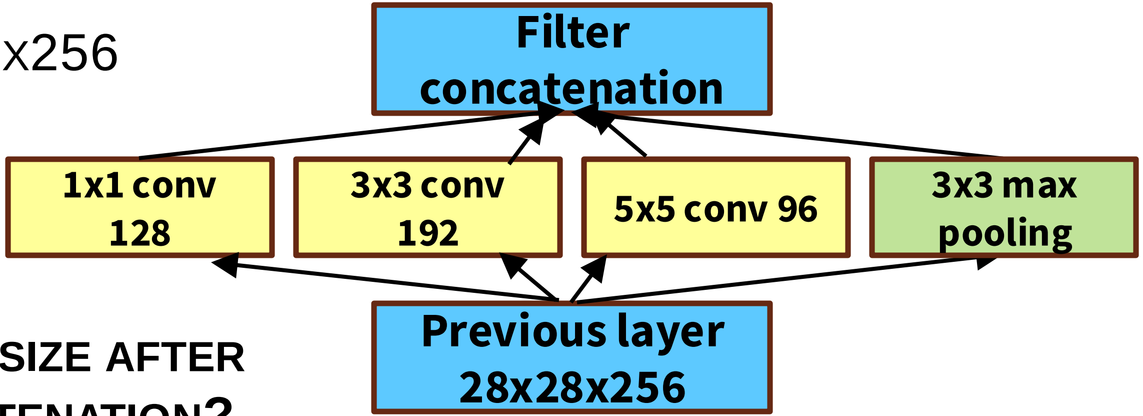
1X1 CONV, 128: 28X28X128

3X3 CONV, 192: 28X28X192

5X5 CONV, 96: 28X28X96

3X3 POOL: 28X28X256

Example:



WHAT IS OUTPUT SIZE AFTER FILTER CONCATENATION?

28x28x(128+192+96+256) = 28x28x672

Carnegie Mellon University

Slide taken from Fei-Fei & Justin Johnson & Serena Yeung. Lecture 9.

[Szegedy et al., 2014]

GoogleNet

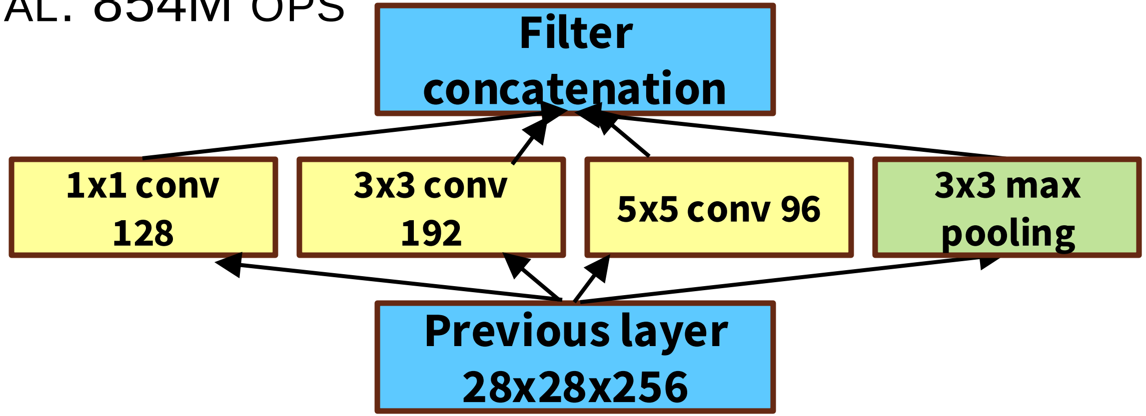
NUMBER OF CONVOLUTION OPERATIONS:

1X1 CONV, 128: 28X28X128X1X1X256

3X3 CONV, 192: 28X28X192X3X3X256

5X5 CONV, 96: 28X28X96X5X5X256

TOTAL: 854M OPS



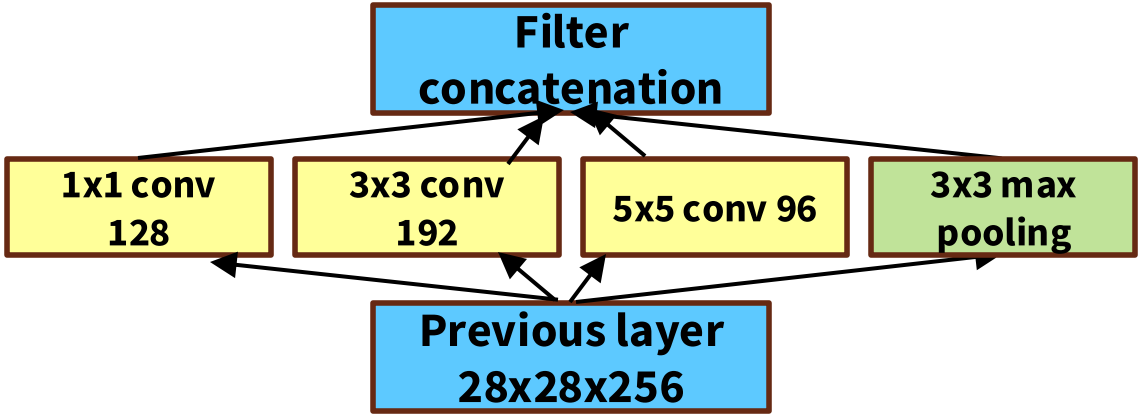
Slide taken from Fei-Fei & Justin Johnson & Serena Yeung. Lecture 9.

[Szegedy et al., 2014]

GoogleNet

VERY EXPENSIVE COMPUTE!

POOLING LAYER ALSO PRESERVES FEATURE DEPTH, WHICH MEANS TOTAL DEPTH AFTER CONCATENATION CAN ONLY GROW AT EVERY LAYER.

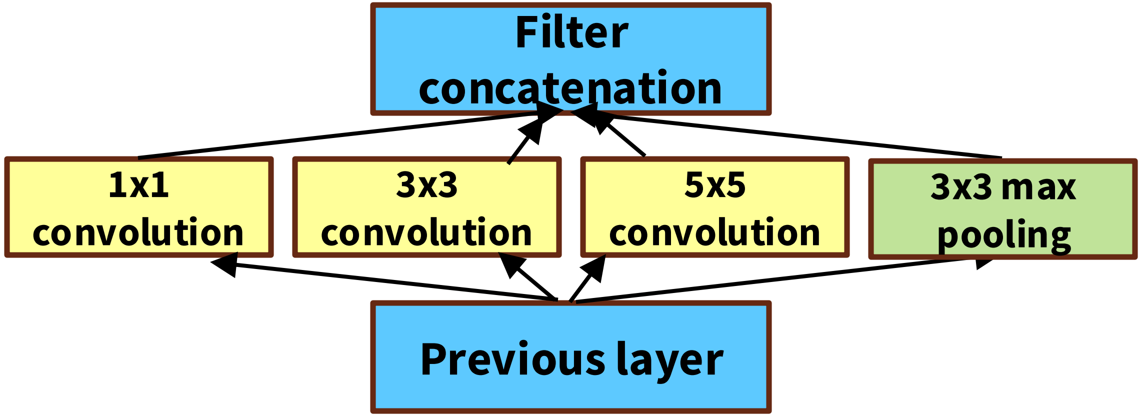


Slide taken from Fei-Fei & Justin Johnson & Serena Yeung. Lecture 9.

[Szegedy et al., 2014]

GoogleNet

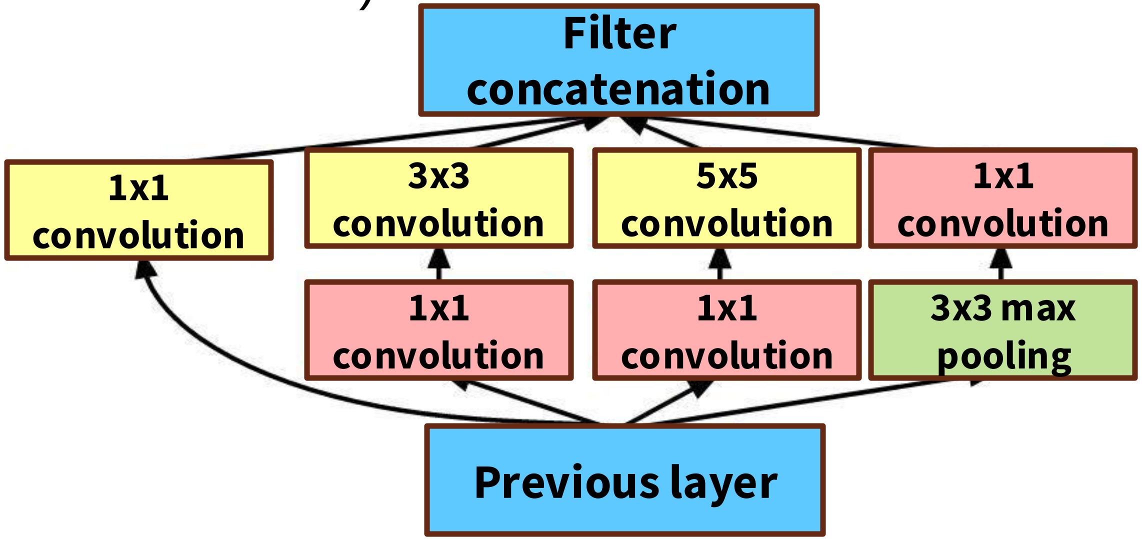
SOLUTION: "BOTTLENECK" LAYERS THAT USE 1X1 CONVOLUTIONS TO REDUCE FEATURE DEPTH (FROM PREVIOUS HOUR).



[Szegedy et al., 2014]

GoogleNet

SOLUTION: "BOTTLENECK" LAYERS THAT USE 1X1 CONVOLUTIONS TO REDUCE FEATURE DEPTH (FROM PREVIOUS HOUR).

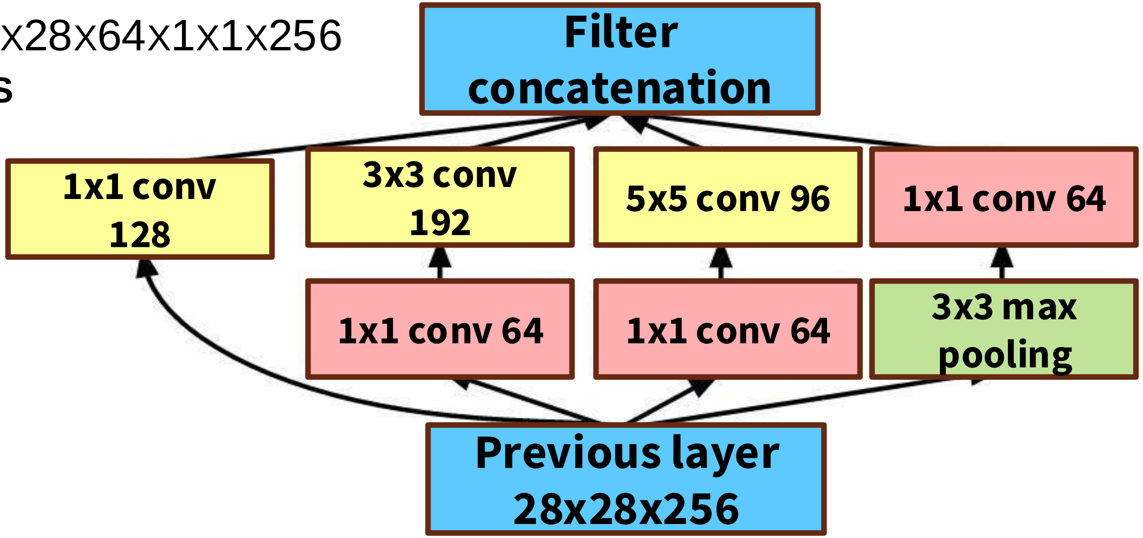


Slide taken from Fei-Fei & Justin Johnson & Serena Yeung. Lecture 9.

[Szegedy et al., 2014]

Number of convolution operations:

1x1 conv, 64: 28x28x64x1x1x256
1x1 conv, 64: 28x28x64x1x1x256
1x1 conv, 128: 28x28x128x1x1x256
3x3 conv, 192: 28x28x192x3x3x64
5x5 conv, 96: 28x28x96x5x5x264
1x1 conv, 64: 28x28x64x1x1x256
Total: 353M ops

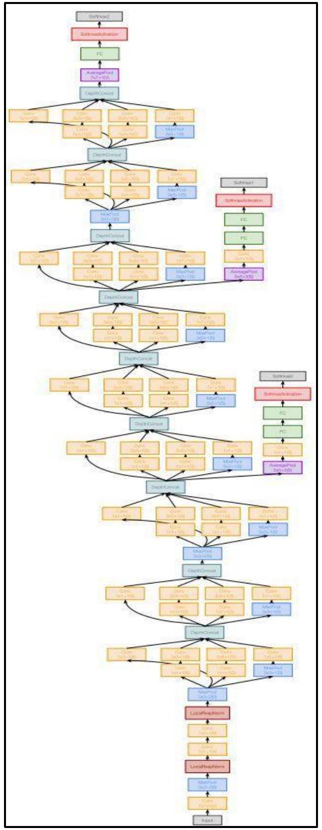


Compared to 854M ops for naive version

Slide taken from Fei-Fei & Justin Johnson & Serena Yeung. Lecture 9.

[Szegedy et al., 2014]

GoogleNet



Details/Retrospectives :

- Deeper networks, with computational efficiency
- 22 layers
- Efficient “Inception” module
- No FC layers
- 12x less params than AlexNet
- ILSVRC’14 classification winner (6.7% top 5 error)

Carnegie Mellon University

Slide taken from Fei-Fei & Justin Johnson & Serena Yeung. Lecture 9.

[Szegedy et al., 2014]

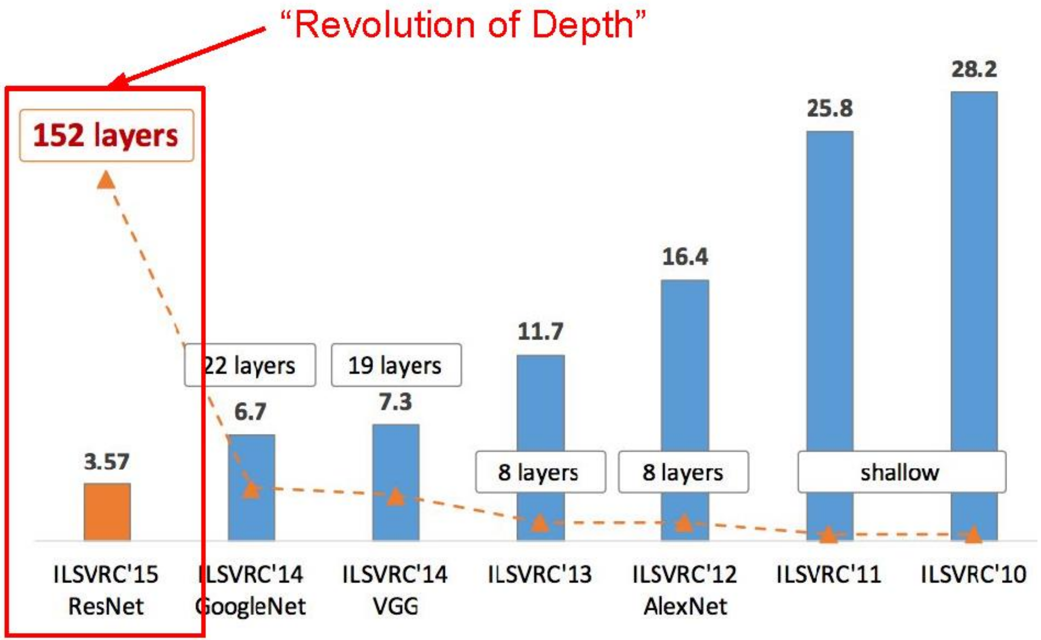
GoogleNet

INTRODUCED THE IDEA THAT CNN LAYERS **DIDN'T ALWAYS HAVE TO BE STACKED UP SEQUENTIALLY.** COMING UP WITH THE INCEPTION MODULE, THE AUTHORS SHOWED THAT A CREATIVE STRUCTURING OF LAYERS CAN LEAD TO IMPROVED PERFORMANCE AND **COMPUTATIONALLY EFFICIENCY**.

Carnegie Mellon University

[Szegedy et al., 2014]

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



Slide taken from Fei-Fei & Justin Johnson & Serena Yeung. Lecture 9.

ResNet

Deep Residual Learning for Image Recognition - Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun; 2015

Extremely deep network – 152 layers

Deeper neural networks are more difficult to train.

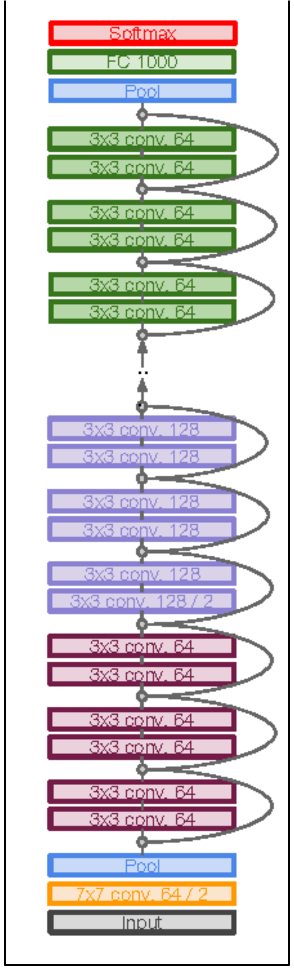
Deep networks suffer from vanishing and exploding gradients.

Present a residual learning framework to ease the training of networks that are substantially deeper than those used previously.

Carnegie Mellon University

[He et al., 2015]

ResNet

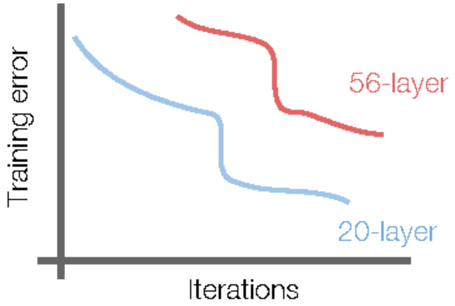


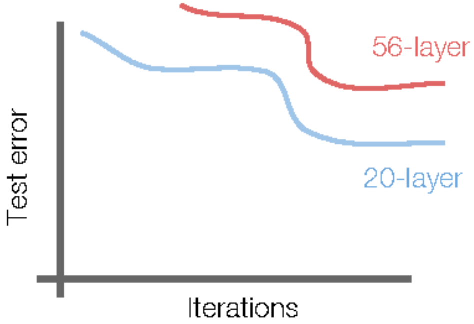
- ILSVRC'15 classification winner (3.57% top 5 error, humans generally hover around a 5-10% error rate)
  Swept all classification and detection competitions in ILSVRC'15 and COCO'15!

[He et al., 2015]

ResNet

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





- 56-layer model performs worse on both training and test error

-> The deeper model performs worse (not caused by overfitting)!

[He et al., 2015]

ResNet

- **Hypothesis**: The problem is an optimization problem. Very deep networks are harder to optimize.
- **Solution**: Use network layers to fit residual mapping instead of directly trying to fit a desired underlying mapping.

- We will use **skip connections** allowing us to take the activation from one layer and feed it into another layer, much deeper into the network.
- Use layers to fit residual F(x) = H(x) – x instead of H(x) directly

Carnegie Mellon University

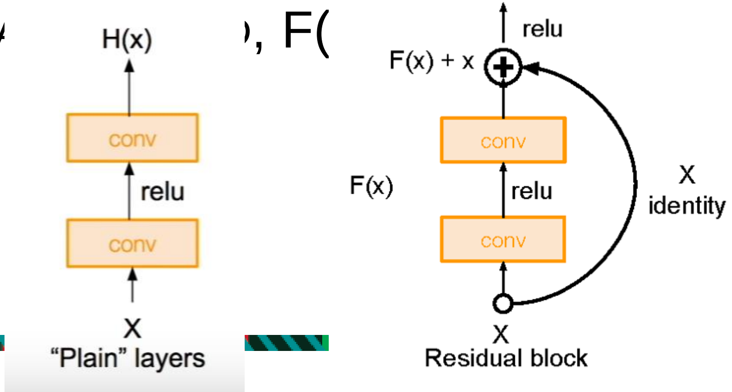
[He et al., 2015]

ResNet

RESIDUAL BLOCK

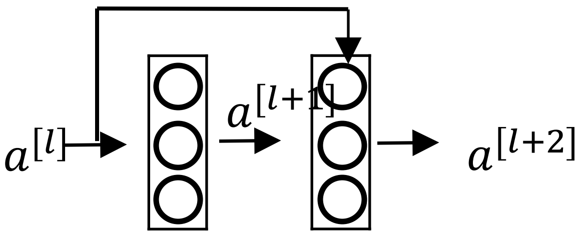
INPUT X GOES THROUGH CONV-RELU-CONV SERIES AND GIVES US F(X). THAT RESULT IS THEN ADDED TO THE ORIGINAL INPUT X. LET'S CALL THAT H(X) = F(X) + X.

IN TRADITIONAL CNNS, H(X) WOULD JUST BE EQUAL TO F(X). SO, INSTEAD OF JUST COMPUTING THAT TRANSFORMATION (STRAIGHT FROM X TO F(X)). WE'RE COMPUTING THE TERM THAT WE HA … , F(… .



[He et al., 2015]

ResNet



Short cut/ skip connection

$a^{[l]}$ → **Linear** → **ReLU** → **Linear** → **ReLU** → $a^{[l+2]}$

$a^{[l+1]}$

$\mathbf{z^{[l+1]} = W^{[l+1]}\, a^{[l]} + b^{[l+1]}}$ $\mathbf{z^{[l+2]} = W^{[l+2]} a^{[l+1]} + b^{[l+2]}}$

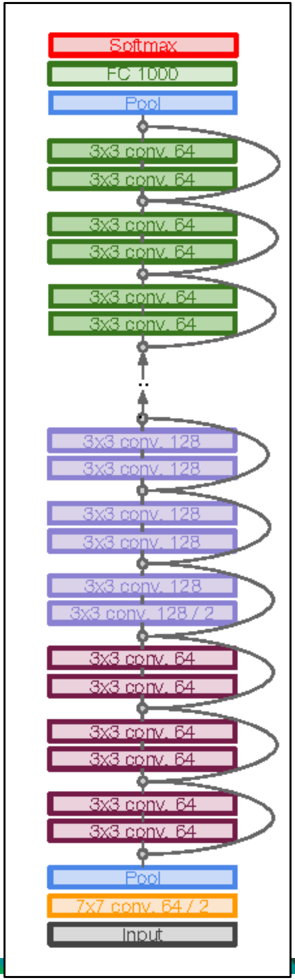
$\mathbf{a^{[l+1]} = g(z^{[l+1]})}$ $\mathbf{a^{[l+2]} = g(z^{[l+2]})}$

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

Carnegie Mellon University

[He et al., 2015]

ResNet



FULL RESNET ARCHITECTURE:

STACK RESIDUAL BLOCKS

EVERY RESIDUAL BLOCK HAS TWO 3X3 CONV LAYERS

PERIODICALLY, DOUBLE # OF FILTERS AND DOWNSAMPLE SPATIALLY USING STRIDE 2 (IN EACH DIMENSION)

ADDITIONAL CONV LAYER AT THE BEGINNING

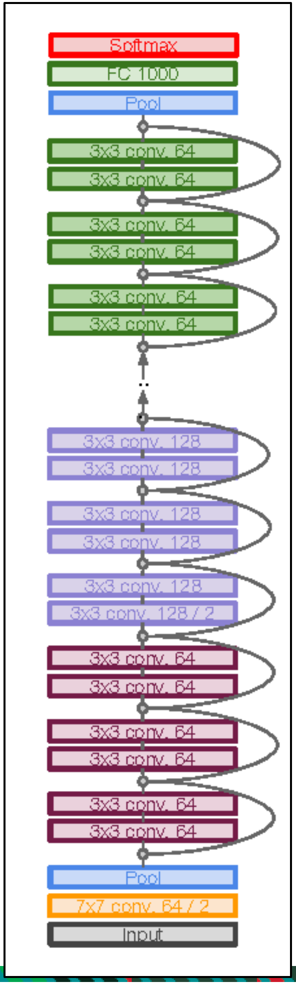
NO FC LAYERS AT THE END (ONLY FC 1000 TO OUTPUT CLASSES)

Carnegie Mellon University

[He et al., 2015]

Slide taken from Fei-Fei & Justin Johnson & Serena Yeung. Lecture 9.

ResNet



- Total depths of 34, 50, 101, or 152 layers for ImageNet
- For deeper networks (ResNet-50+), use "bottleneck" layer to improve efficiency (similar to GoogLeNet)

Carnegie Mellon University

[He et al., 2015]

ResNet

EXPERIMENTAL RESULTS:

ABLE TO TRAIN VERY DEEP NETWORKS WITHOUT DEGRADING

DEEPER NETWORKS NOW ACHIEVE LOWER TRAINING ERRORS
AS EXPECTED

[He et al., 2015]

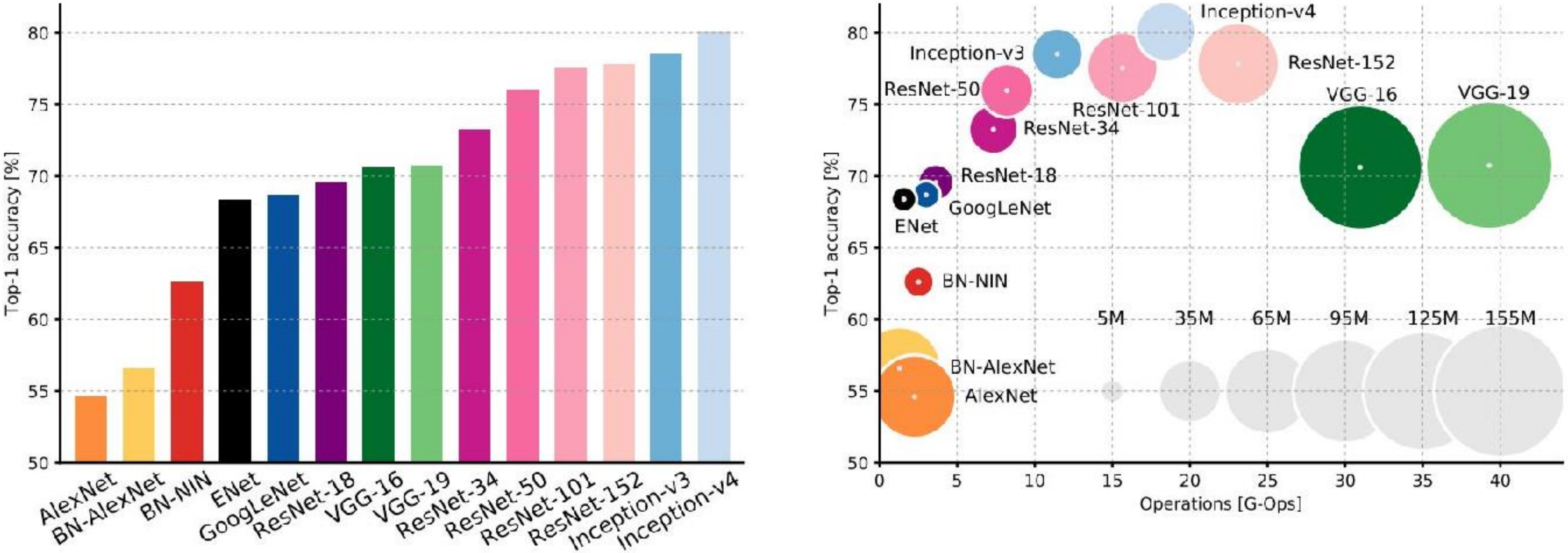
ResNet

THE **BEST** CNN ARCHITECTURE THAT WE CURRENTLY HAVE AND IS A GREAT INNOVATION FOR THE IDEA OF RESIDUAL LEARNING.

EVEN BETTER THAN HUMAN PERFORMANCE!

[He et al., 2015]

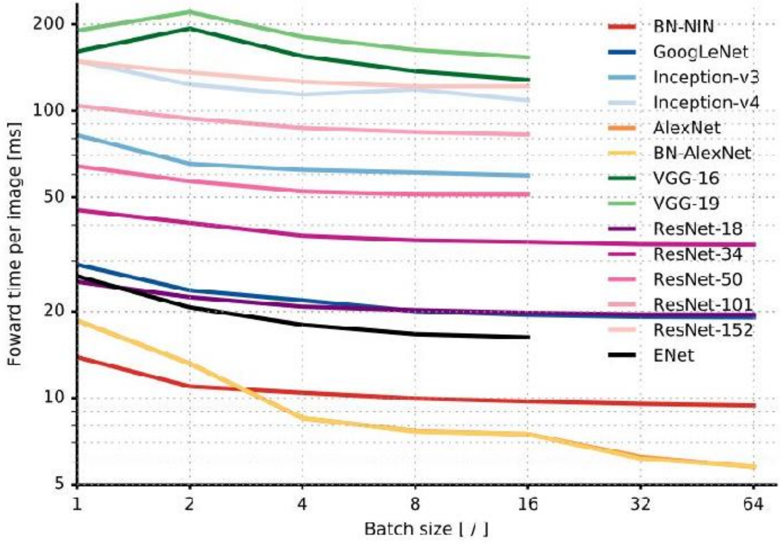
Accuracy comparison

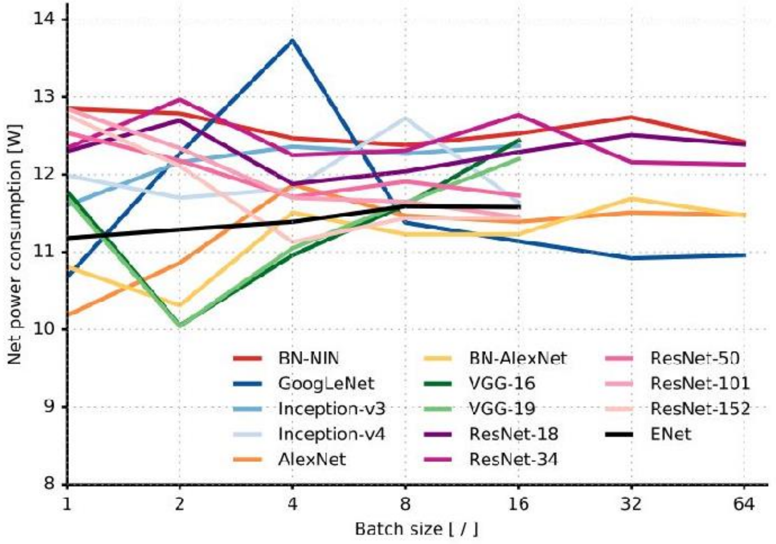


Carnegie Mellon University

Slide taken from Fei-Fei & Justin Johnson & Serena Yeung. Lecture 9.

Forward pass time and power consumption





Carnegie Mellon University

Slide taken from Fei-Fei & Justin Johnson & Serena Yeung. Lecture 9.

Summary

LeNet-5

AlexNet

VGG

GoogLeNet – Inception module

ResNet – Residual block

References

Gradient-based learning applied to document recognition; Ann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner; 1998

ImageNet Classification with Deep Convolutional Neural Networks - Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton; 2012

Very Deep Convolutional Networks For Large Scale Image Recognition - Karen Simonyan and Andrew Zisserman; 2015

Going Deeper with Convolutions - Christian Szegedy et al.; 2015

Deep Residual Learning for Image Recognition - Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun; 2015

Stanford CS231- Fei-Fei & Justin Johnson & Serena Yeung. Lecture 9

Coursera, Machine Learning course by Andrew Ng.

References

THE 9 DEEP LEARNING PAPERS YOU NEED TO KNOW ABOUT (UNDERSTANDING CNNS PART 3) BY ADIT DESHPANDE HTTPS://ADESHPANDE3.GITHUB.IO/ADESHPANDE3.GITHUB.IO/THE-9-DEEP-LEARNING-PAPERS-YOU-NEED-TO-KNOW-ABOUT.HTML

CNNS ARCHITECTURES: LENET, ALEXNET, VGG, GOOGLENET, RESNET AND MORE … BY SIDDHARTH DAS HTTPS://MEDIUM.COM/@SIDDHARTHDAS_32104/CNNS-ARCHITECTURES-LENET-ALEXNET-VGG-GOOGLENET-RESNET-AND-MORE-666091488DF5

SLIDE TAKEN FROM FORWARD AND BACKPROPAGATION IN CONVOLUTIONAL NEURAL NETWORK. – MEDIUM , BY SUJIT RAI HTTPS://MEDIUM.COM/@2017CSM1006/FORWARD-AND-BACKPROPAGATION-IN-CONVOLUTIONAL-NEURAL-NETWORK-4DFA96D7B37E

Carnegie Mellon University