Advanced Machine Learning and Deep Learning

Dr. Mohammad Pourhomayoun
Assistant Professor
Computer Science Department
California State University, Los Angeles

Deep Learning

Convolutional Neural Networks (CNN)

Convolutional Neural Networks (CNN, or ConvNet)

- **Convolutional Neural Networks (CNN, aka ConvNet)** is a type of deep neural networks inspired by the human brain **visual cortex**.
- The response of an individual cortical neuron within its receptive field can be approximated mathematically by a **convolution** operation.
- CNN has been the most popular deep learning method for image classification and object recognition. **In 2015, CNN based algorithms could beat human in object recognition for the first time!!!**

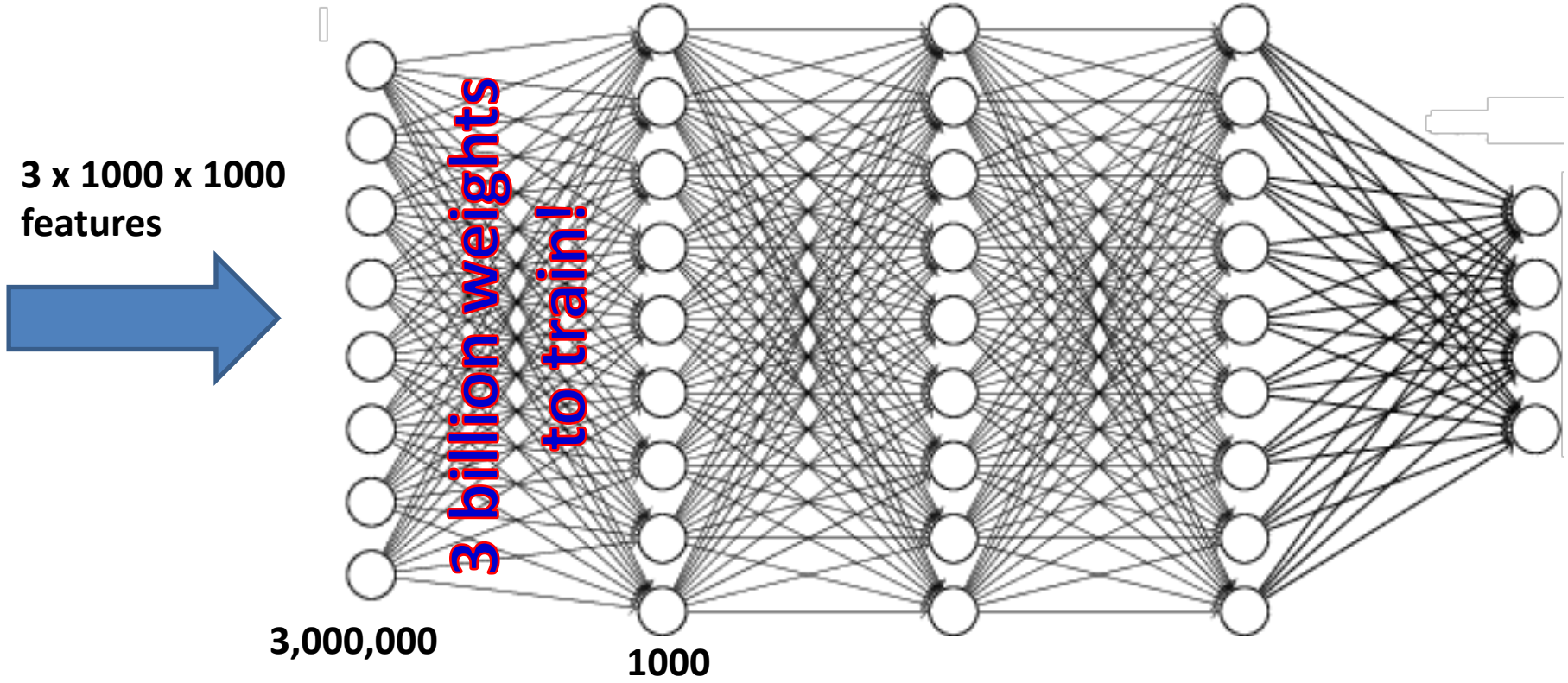
Convolutional Neural Networks (CNN, or ConvNet)

- The **dimensionality** of the data is very high when we work with images (each pixel of an image can be a feature for our machine learning algorithm!). Thus, we usually need a **smart compact feature set** along with **dimensionality reduction** methods to represent an image.
- Moreover, in classic machine learning, **extracting and using meaningful patterns** from image (such as edges or basic shapes) can be more efficient than using raw pixels. Thus, many different techniques have been proposed by image processing community to extract **features that can represent an image** for machine learning process (e.g. HOG, SIFT, SURF, …).
- However, in **deep learning**, we would like to use **feature learning methods** to automatically extract (learn) the best set of features that are much more efficient than **handcrafted features** in representing the image.

Computational Complexity

1000 x 1000 pixels

3 x 1000 x 1000 features

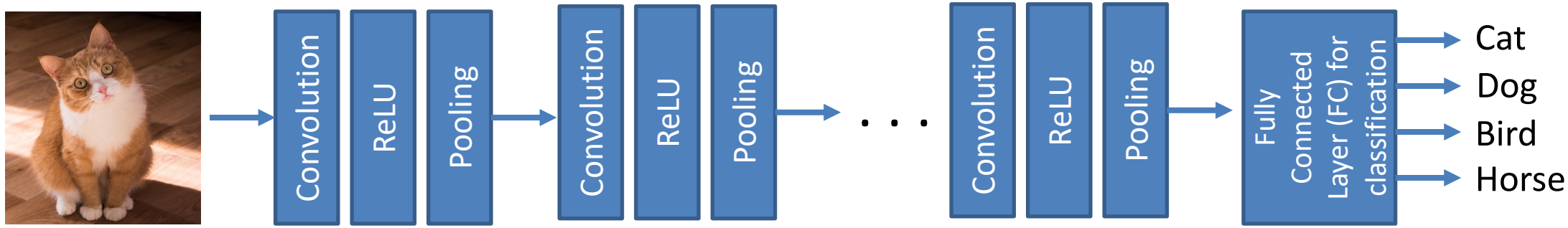


If we have 1000 neurons in the first hidden layer, then the number of weights only in the first hidden layer is: 3,000,000,000 x 1000 = 3 billion weights!!!!

[Image source]: Inception movie

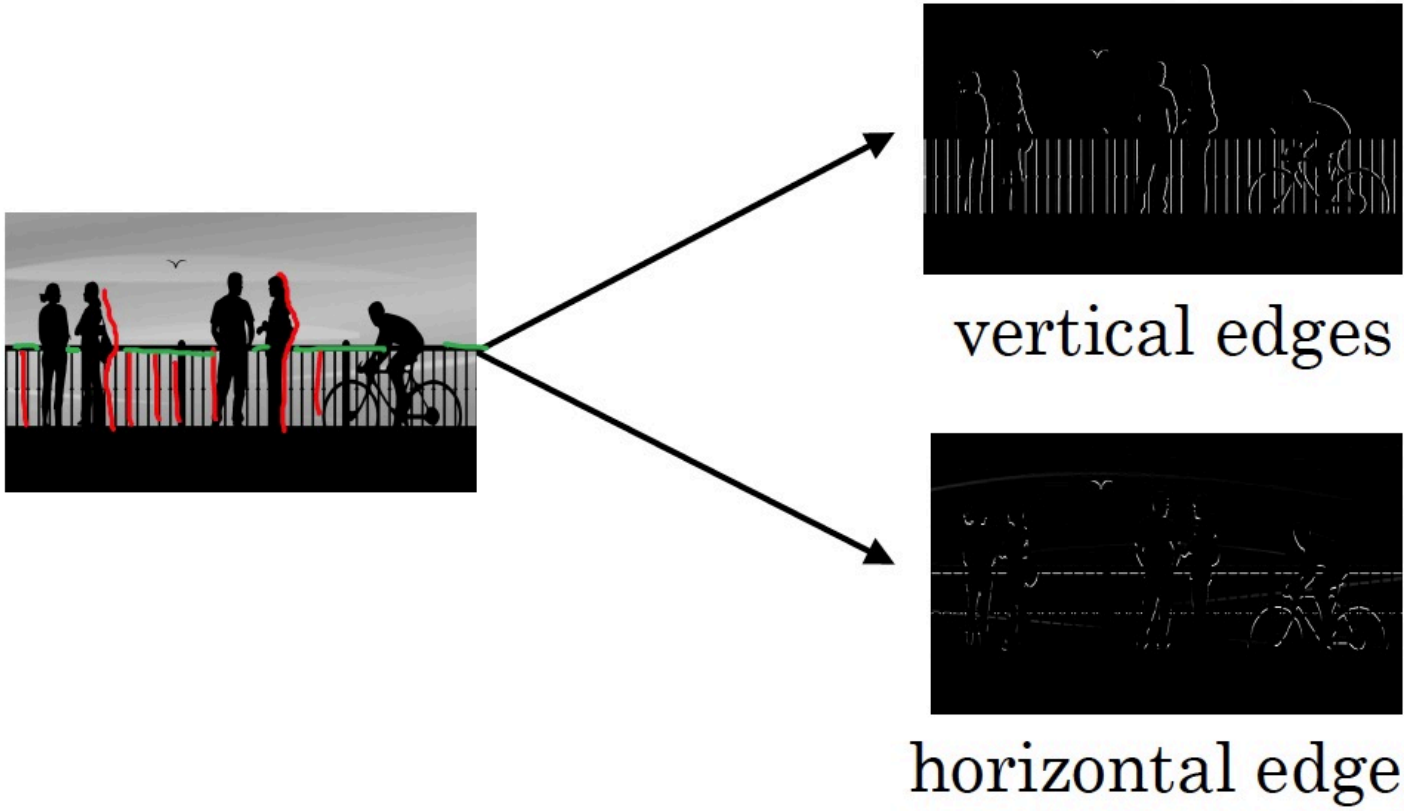
Convolutional Neural Networks (CNN, aka ConvNet)

- **Convolutional Neural Networks** consist of a series of layers including **Convolutional Layers** (for feature extraction), **ReLU Layers** (for non-linearity), and **Pooling Layers** (for further dimensionality reduction), followed by some **Fully Connected** Layers (for final classification).



First Thought: Extracting Edge Information as Image Features

- Edge detection is a fundamental tool in computer vision, particularly in image feature extraction and representation.

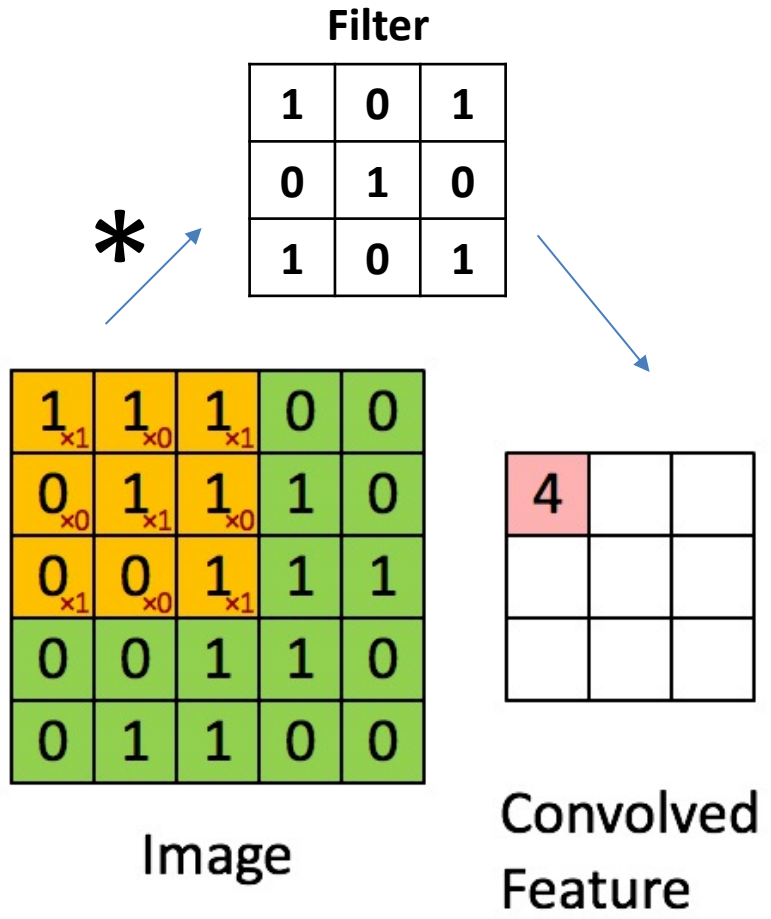


[Figure Source]: Andrew Ng

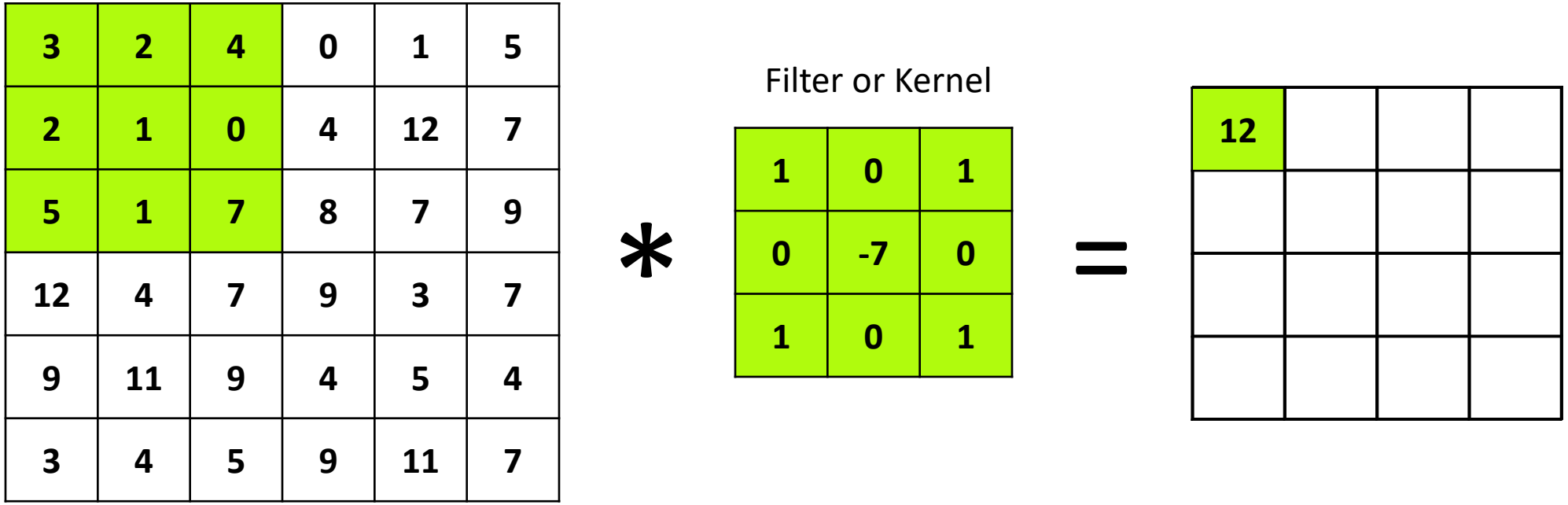
Convolution for Edge Detection

- **Convolution** in image processing is the process of weighting the pixels of an image by multiplying them to a **filter** (aka **kernel**), and then add up the local neighbors (it is like sliding a filter all over the original image).
- **The output will be a new image with the size of:**

  (OriginalSize – FilterSize + 1)



Convolution



$$1 \times 3 + 0 \times 2 + 1 \times 4 + 0 \times 2 - 7 \times 1 + 0 \times 0 + 1 * 5 + 0 \times 1 + 1 \times 7 = 12$$

Convolution

3	2	4	0	1	5
2	1	0	4	12	7
5	1	7	8	7	9
12	4	7	9	3	7
9	11	9	4	5	4
3	4	5	9	11	7

*

Filter or Kernel

1	0	1
0	-7	0
1	0	1

=

12	11	-9	-62
14	-31	-34	-22
2	-25	-35	4
-50	-37	-2	-3

$$1 \times 2 + 0 \times 4 + 1 \times 0 + 0 \times 1 - 7 \times 0 + 0 \times 4 + 1 * 1 + 0 \times 7 + 1 \times 8 = 11$$

Vertical Edge Detection Using Convolution

5	5	5	5	0	0	0	0
5	5	5	5	0	0	0	0
5	5	5	5	0	0	0	0
5	5	5	5	0	0	0	0
5	5	5	5	0	0	0	0
5	5	5	5	0	0	0	0

$*$

Filter or Kernel

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

$=$

Vertical Edge Detection Using Convolution

5	5	5	5	0	0	0	0
5	5	5	5	0	0	0	0
5	5	5	5	0	0	0	0
5	5	5	5	0	0	0	0
5	5	5	5	0	0	0	0
5	5	5	5	0	0	0	0

$*$

Filter or Kernel

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

$=$

Vertical Edge Detection Using Convolution

5	5	5	5	0	0	0	0
5	5	5	5	0	0	0	0
5	5	5	5	0	0	0	0
5	5	5	5	0	0	0	0
5	5	5	5	0	0	0	0
5	5	5	5	0	0	0	0

*

Filter or Kernel

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

=

0					

Vertical Edge Detection Using Convolution

5	5	5	5	0	0	0	0
5	5	5	5	0	0	0	0
5	5	5	5	0	0	0	0
5	5	5	5	0	0	0	0
5	5	5	5	0	0	0	0
5	5	5	5	0	0	0	0

*

Filter or Kernel

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

=

0					

Vertical Edge Detection Using Convolution

5	5	5	5	0	0	0	0
5	5	5	5	0	0	0	0
5	5	5	5	0	0	0	0
5	5	5	5	0	0	0	0
5	5	5	5	0	0	0	0
5	5	5	5	0	0	0	0

*

Filter or Kernel

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

=

0	0				

Vertical Edge Detection Using Convolution

5	5	5	5	0	0	0	0
5	5	5	5	0	0	0	0
5	5	5	5	0	0	0	0
5	5	5	5	0	0	0	0
5	5	5	5	0	0	0	0
5	5	5	5	0	0	0	0

$*$

Filter or Kernel

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

$=$

0	0				

Vertical Edge Detection Using Convolution

5	5	5	5	0	0	0	0
5	5	5	5	0	0	0	0
5	5	5	5	0	0	0	0
5	5	5	5	0	0	0	0
5	5	5	5	0	0	0	0
5	5	5	5	0	0	0	0

$*$

Filter or Kernel

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

$=$

0	0	15			

Vertical Edge Detection Using Convolution

5	5	5	5	0	0	0	0
5	5	5	5	0	0	0	0
5	5	5	5	0	0	0	0
5	5	5	5	0	0	0	0
5	5	5	5	0	0	0	0
5	5	5	5	0	0	0	0

$*$

Filter or Kernel

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

$=$

0	0	15			

Vertical Edge Detection Using Convolution

5	5	5	5	0	0	0	0
5	5	5	5	0	0	0	0
5	5	5	5	0	0	0	0
5	5	5	5	0	0	0	0
5	5	5	5	0	0	0	0
5	5	5	5	0	0	0	0

*

Filter or Kernel

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

=

0	0	15	15		

Vertical Edge Detection Using Convolution

5	5	5	5	0	0	0	0
5	5	5	5	0	0	0	0
5	5	5	5	0	0	0	0
5	5	5	5	0	0	0	0
5	5	5	5	0	0	0	0
5	5	5	5	0	0	0	0

*

Filter or Kernel

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

=

0	0	15	15		

Vertical Edge Detection Using Convolution

5	5	5	5	0	0	0	0
5	5	5	5	0	0	0	0
5	5	5	5	0	0	0	0
5	5	5	5	0	0	0	0
5	5	5	5	0	0	0	0
5	5	5	5	0	0	0	0

*

Filter or Kernel

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

=

0	0	15	15	0	

Vertical Edge Detection Using Convolution

5	5	5	5	0	0	0	0
5	5	5	5	0	0	0	0
5	5	5	5	0	0	0	0
5	5	5	5	0	0	0	0
5	5	5	5	0	0	0	0
5	5	5	5	0	0	0	0

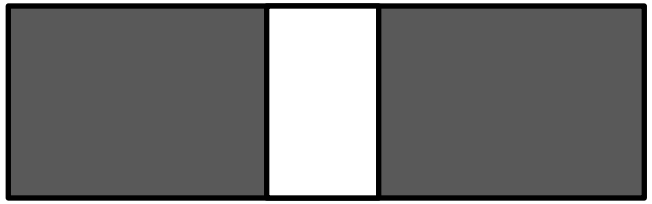
*

Filter or Kernel

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

=

0	0	15	15	0	0
0	0	15	15	0	0
0	0	15	15	0	0
0	0	15	15	0	0



Vertical Edge Detection Using Convolution

5	5	5	5	0	0	0	0
5	5	5	5	0	0	0	0
5	5	5	5	0	0	0	0
5	5	5	5	0	0	0	0
5	5	5	5	0	0	0	0
5	5	5	5	0	0	0	0

*

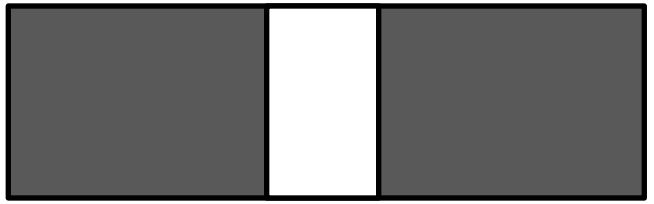
Filter or Kernel

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

=

Vertical Edge location & intensity is detected

0	0	15	15	0	0
0	0	15	15	0	0
0	0	15	15	0	0
0	0	15	15	0	0



Edge Detection Using Convolution

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

Vertical Edge Filter

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

Horizontal Edge Filter

0	1	0
1	0	-1
0	-1	0

0	1	0
-1	0	1
0	-1	0

Diagonal Edge Filter

1	0	-1
2	0	-2
1	0	-1

Vertical Sobel Edge Filter

3	0	-3
10	0	-10
3	0	-3

Vertical Scharr Edge Filter

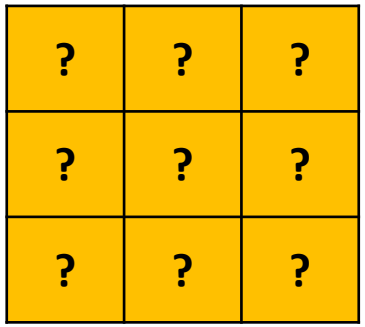
?	?	?
?	?	?
?	?	?

What are the best Filters?

What are the best set of filters?

Question:

What are the best set of filters that can be used to detect the most important edges?



What are the best Filters?

Answer:

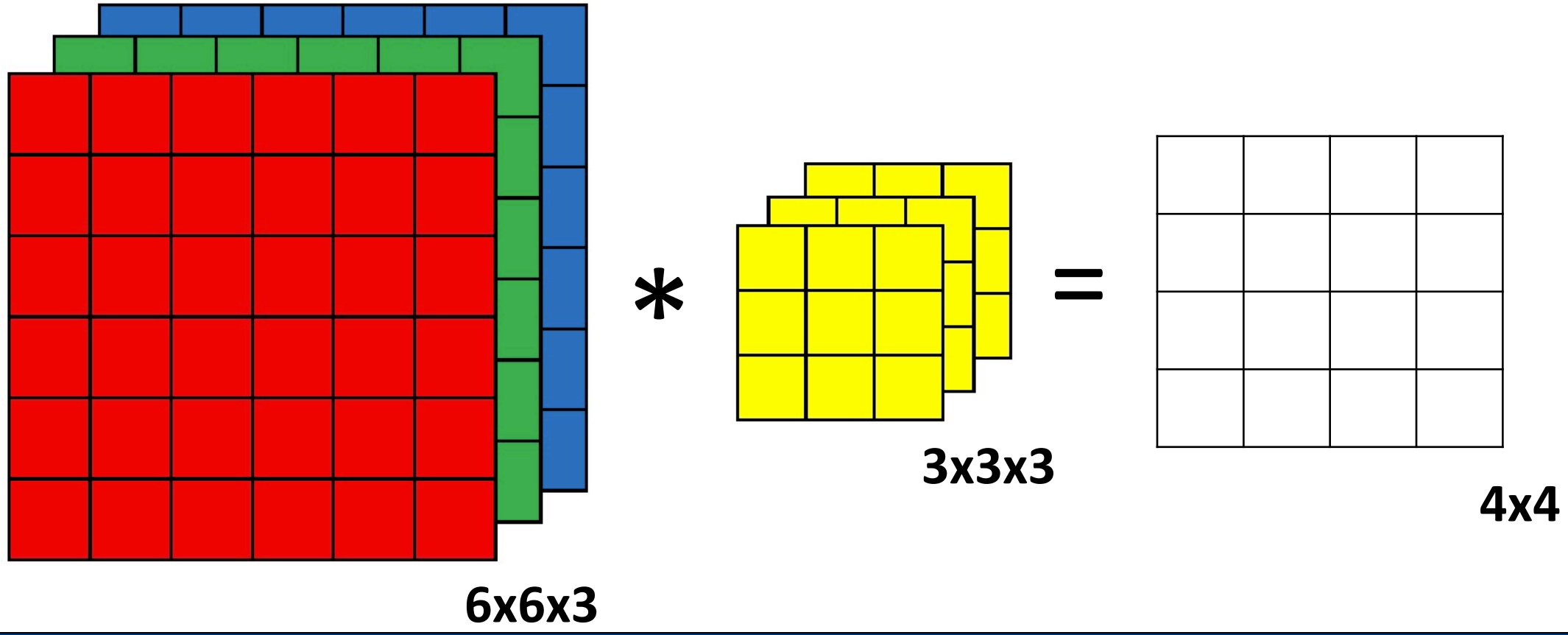
The best set of filters in a machine learning scenario are the filters that we can learn from data!

We can treat them as weights of a Neural Network that can be learned in a backpropagation process!

Let's find them in a training process!

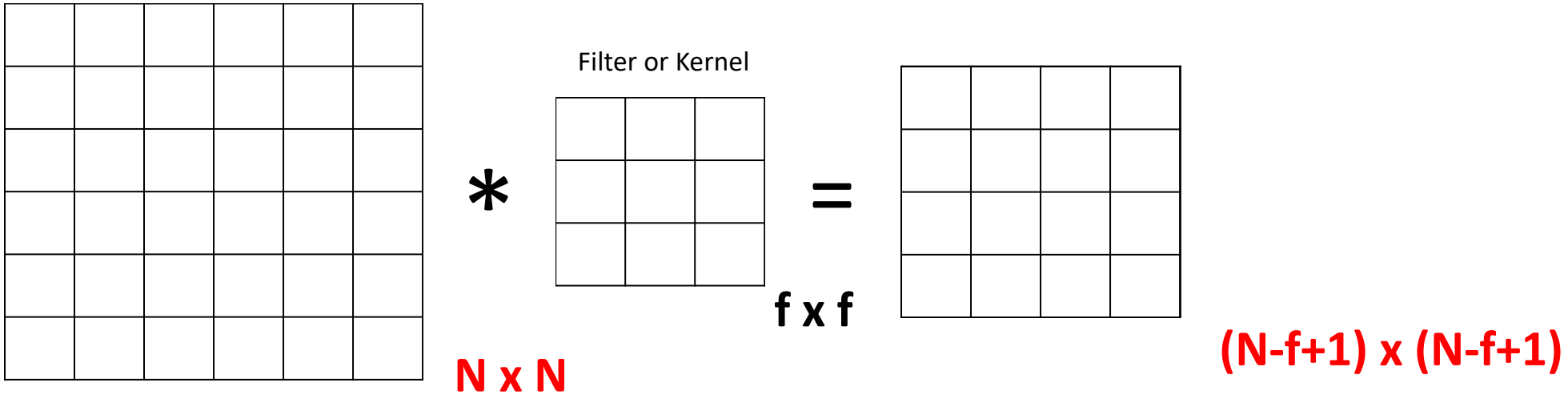
Convolution for RGB Images

- For RGB images, our filter is also 3D. However, our output is still 2D.



Some Problems with Regular Convolution:

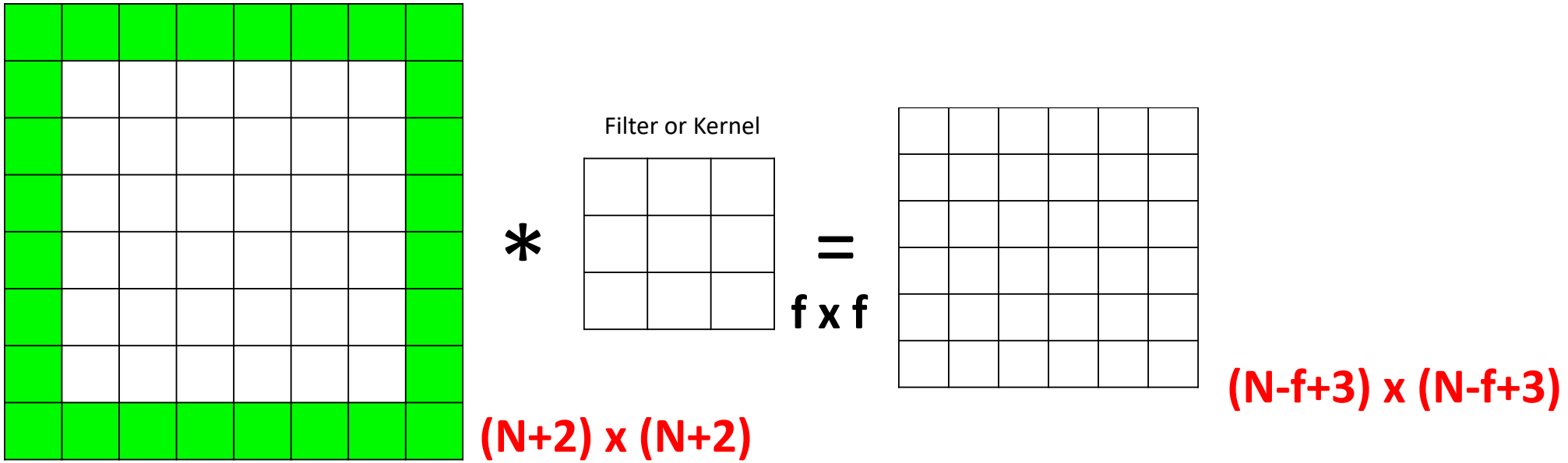
- **Problem1:** Notice the **size reduction** in each round of convolution (this is called ***Valid Convolution***):



- **Problem2:** There will be more focus on **middle pixels** (because middle pixels are involved in many convolution operations) and **less focus on edge pixels**.

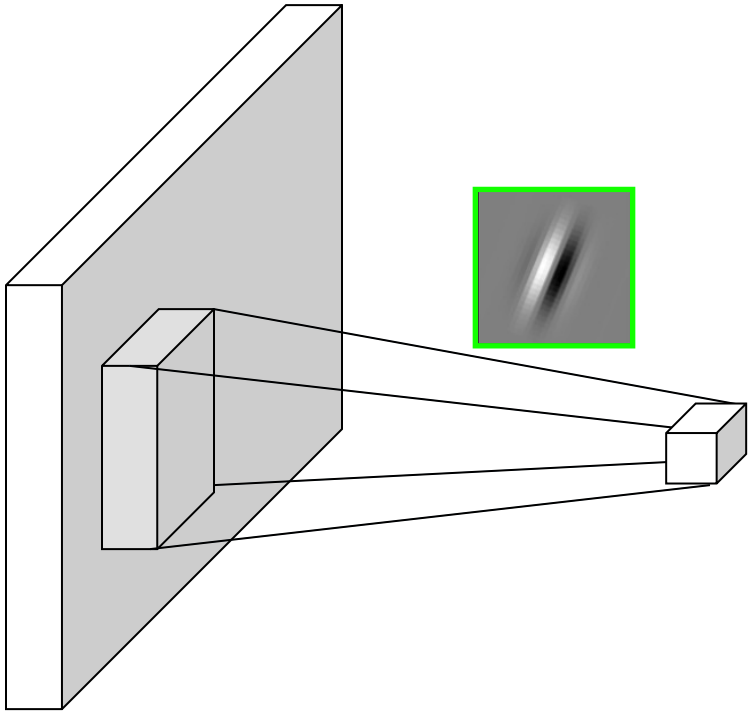
Zero Padding

- To resolve these problems, we can add **zeros** to the borders of the original image (zero padding) before convolution (**note**: if your filter is 3x3 (i.e. f=3), then the original size will be preserved after convolution with only one row of zero padding!).



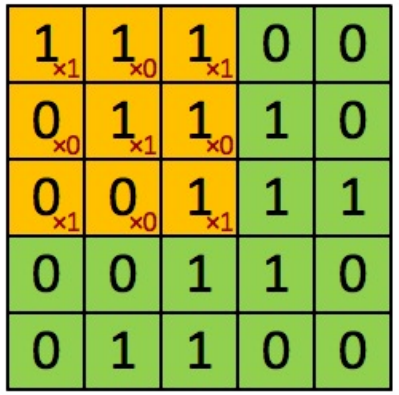
It is called ***Same Convolution*** when **Padding_Size = (f-1)/2** and consequently, **output size = input size**.

Convolutional Neural Networks

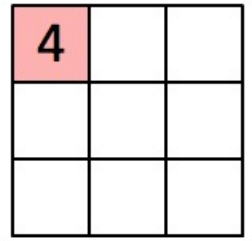


Image

Convolutional Layer



Image



Convolved Feature

[Figure Source]: Svetlana Lazebnik

Convolutional Neural Networks

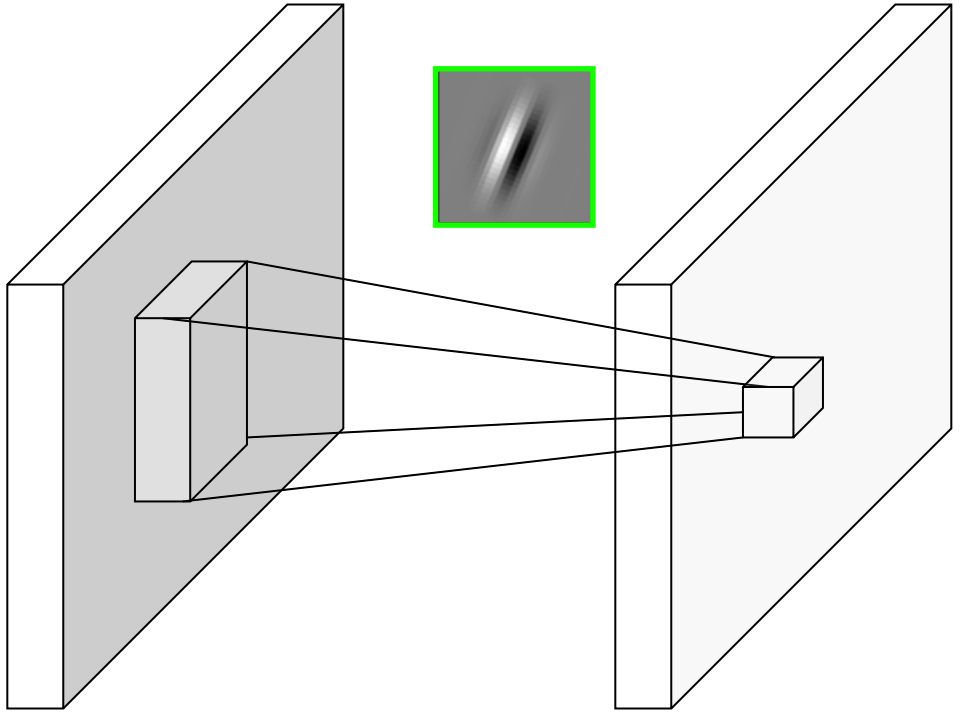
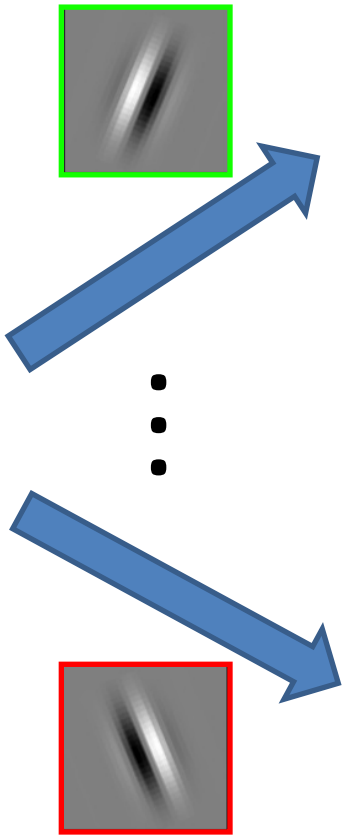
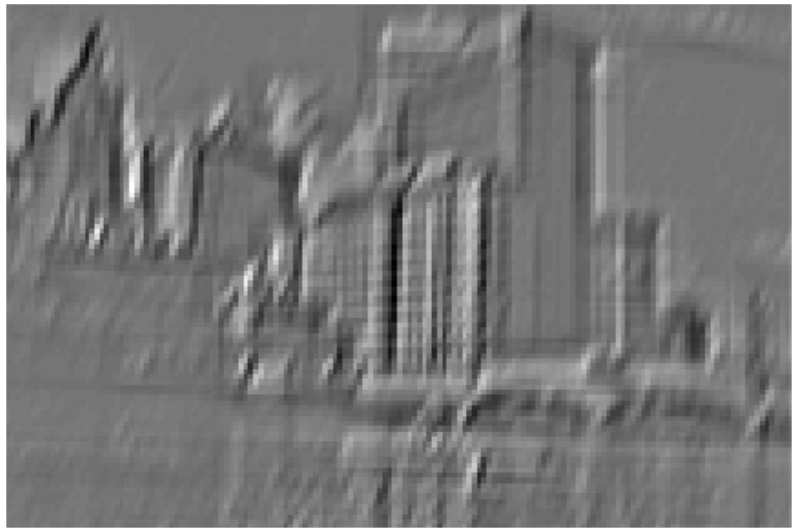


Image **Convolutional Layer**

[Figure Source]: Svetlana Lazebnik

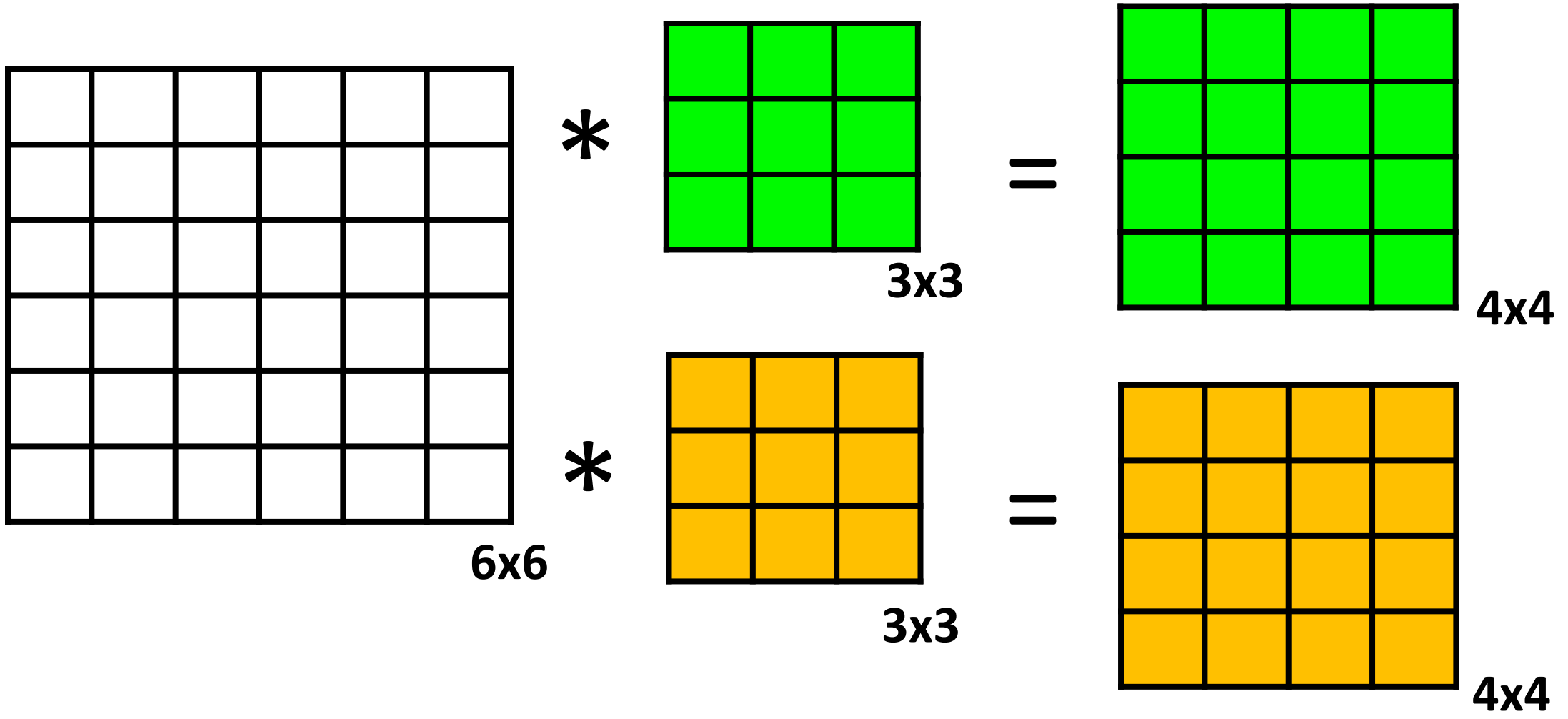
We always use more than one filter to detect different edges!



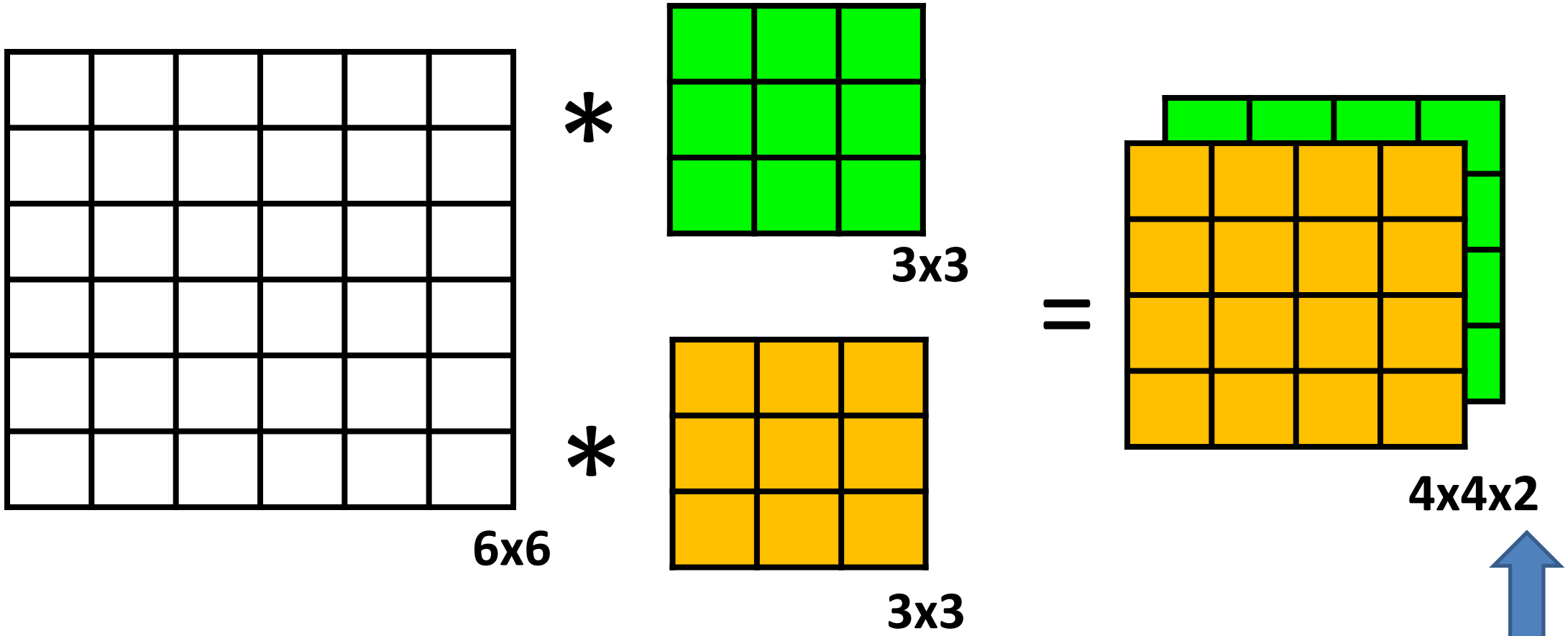


[image Source]: Svetlana Lazebnik

We always use several filters to detect different edges!

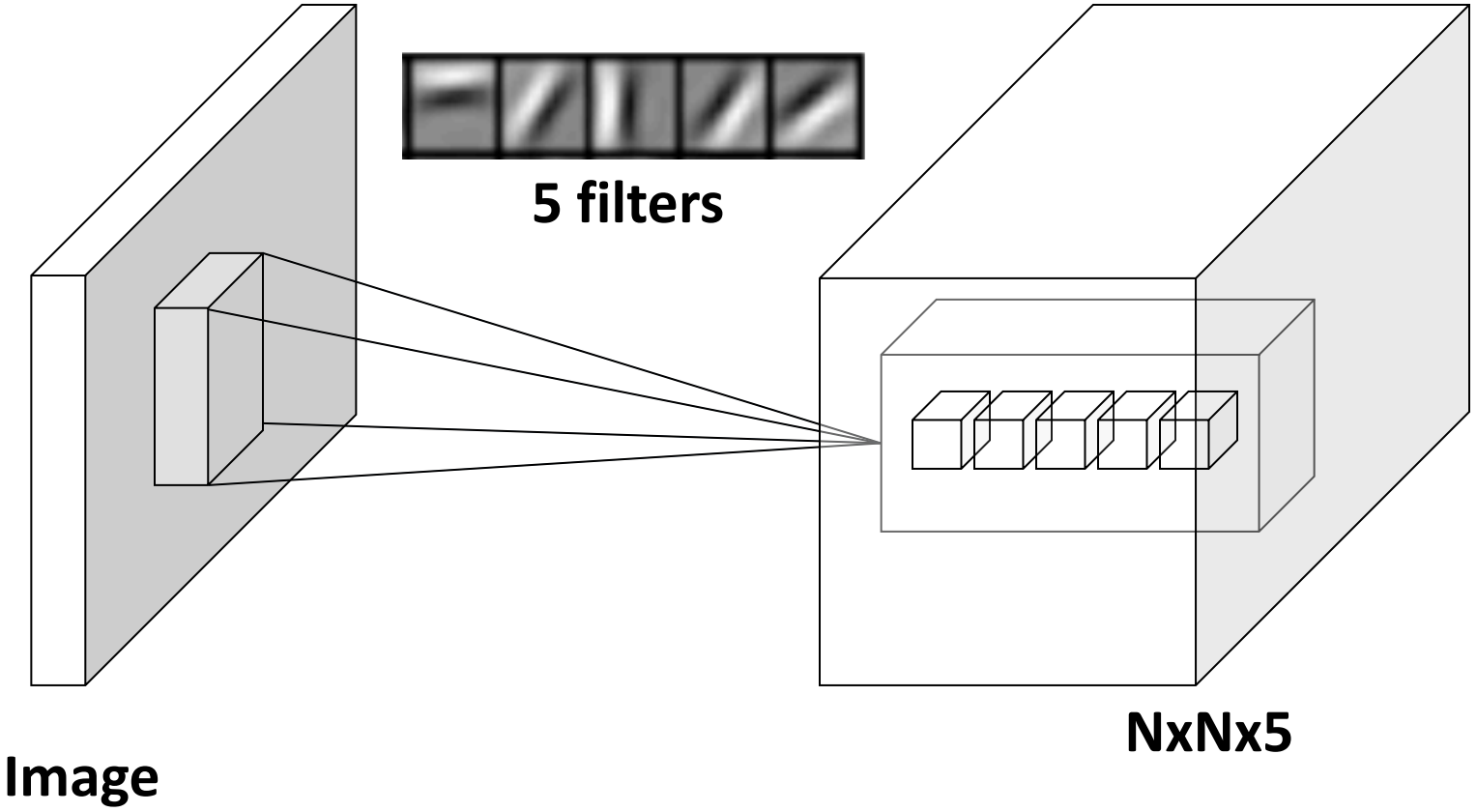


We always use several filters to detect different edges!



NOTICE: This “2” represents the convolution results with 2 filters

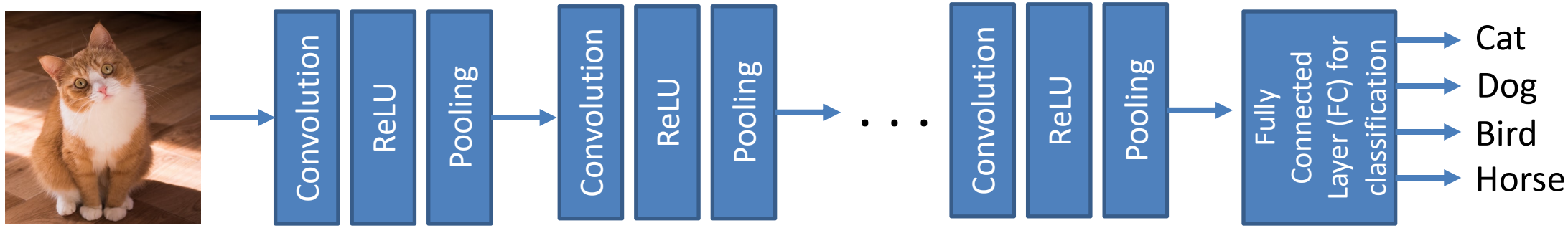
Convolutional Neural Networks



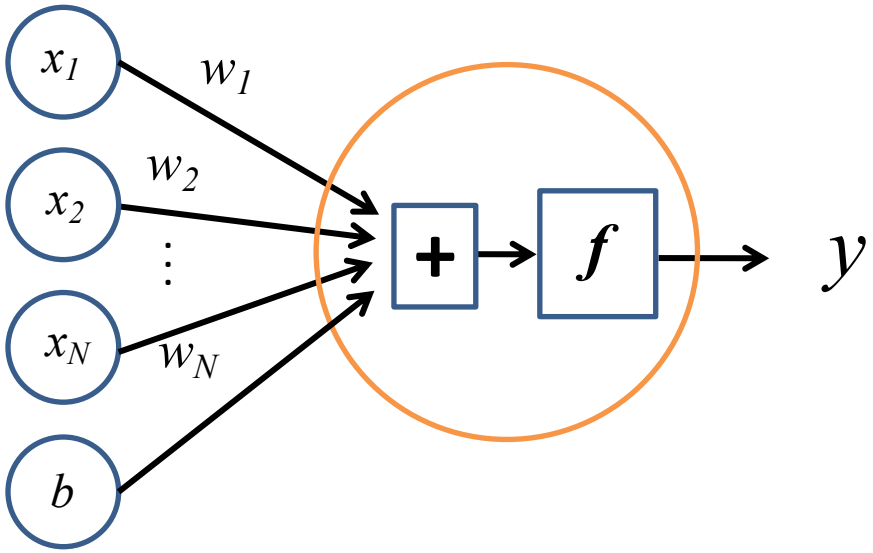
[Figure Source]: Svetlana Lazebnik

Convolutional Neural Networks (CNN, aka ConvNet)

- **Convolutional Neural Networks** consist of a series of layers including **Convolutional Layers** (for feature extraction), **ReLU Layers** (for non-linearity), and **Pooling Layers** (for further dimensionality reduction), followed by some **Fully Connected** Layers (for final classification).

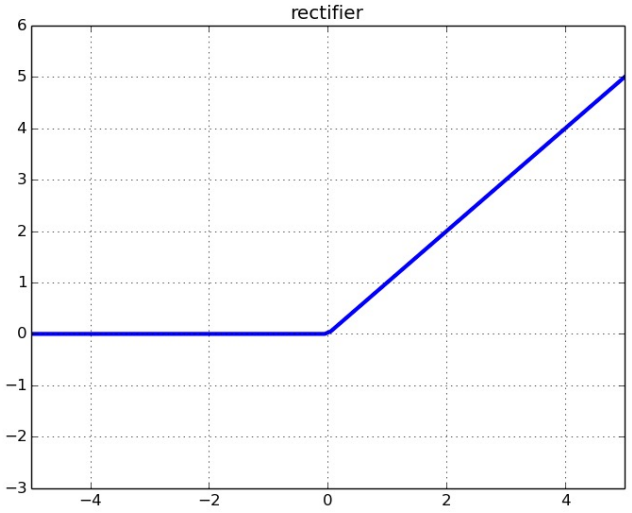


Adding Bias and ReLU Non-linear Activation Function

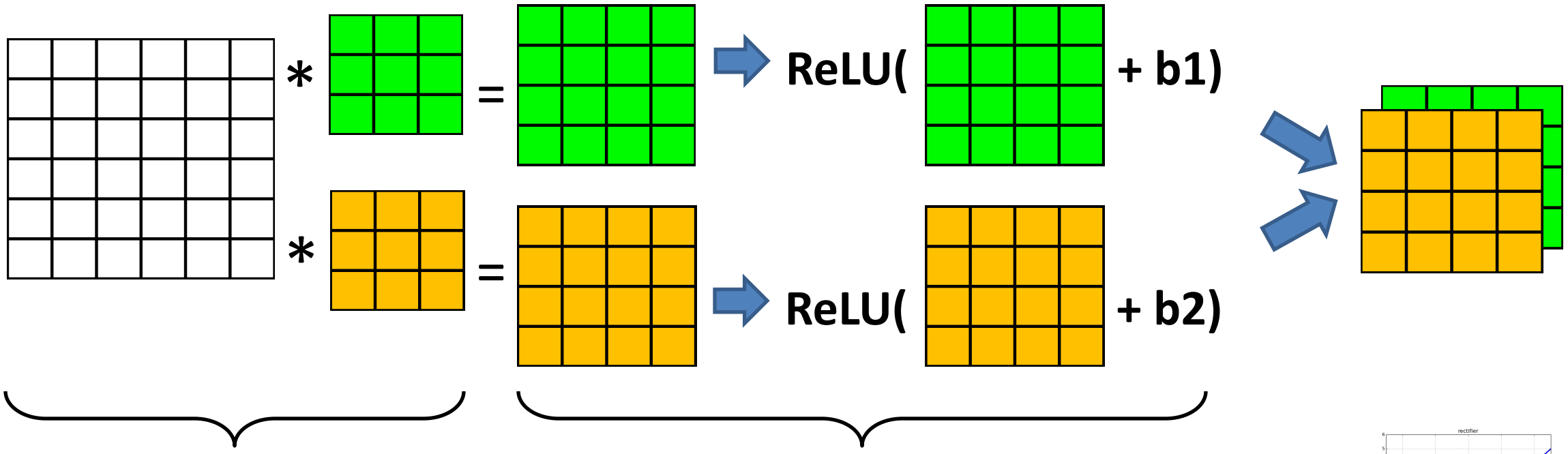


$$y = f(w_0 b + \sum_{i=1}^{N} x_i w_i)$$

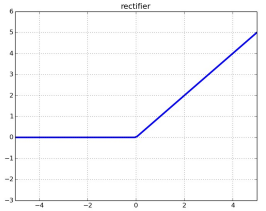
ReLU as activation function f:

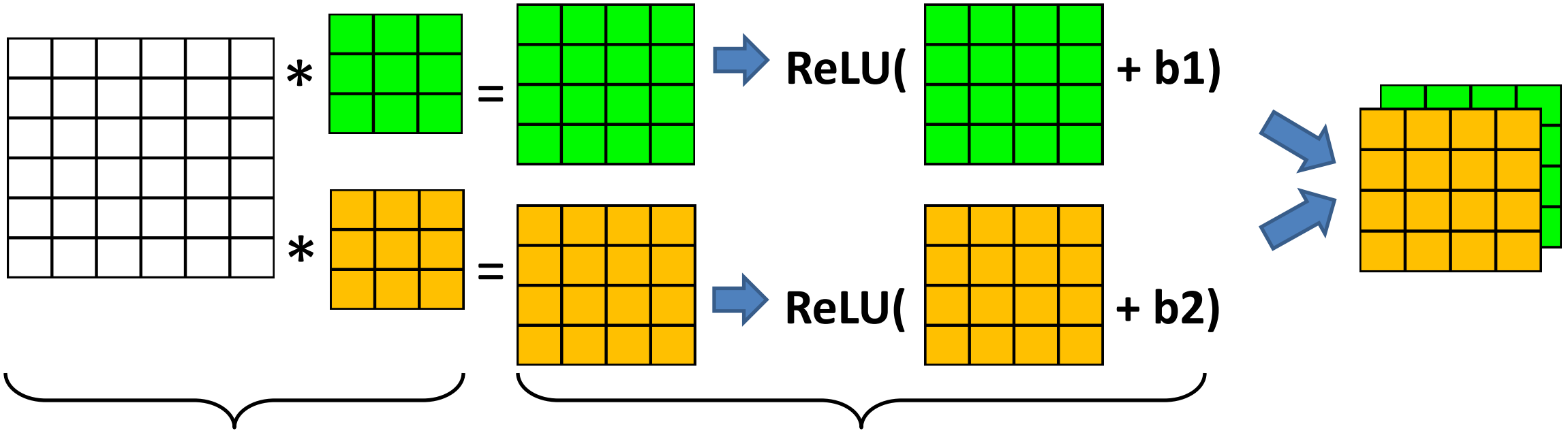


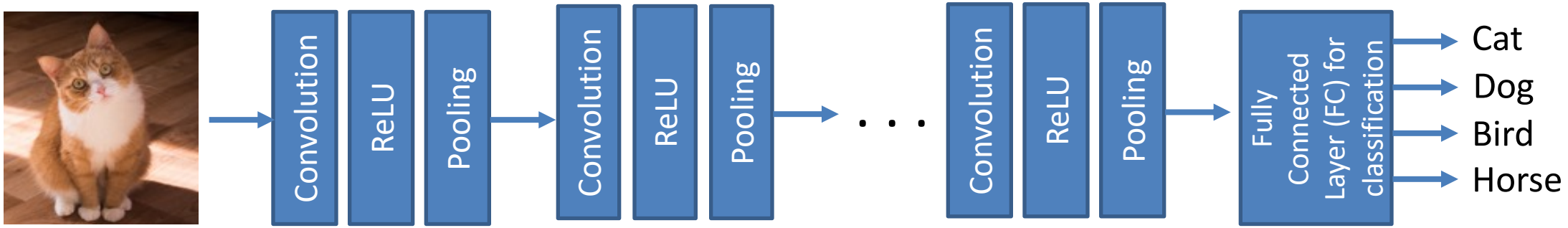
Adding Bias and ReLU Non-linear Activation Function



ReLU:

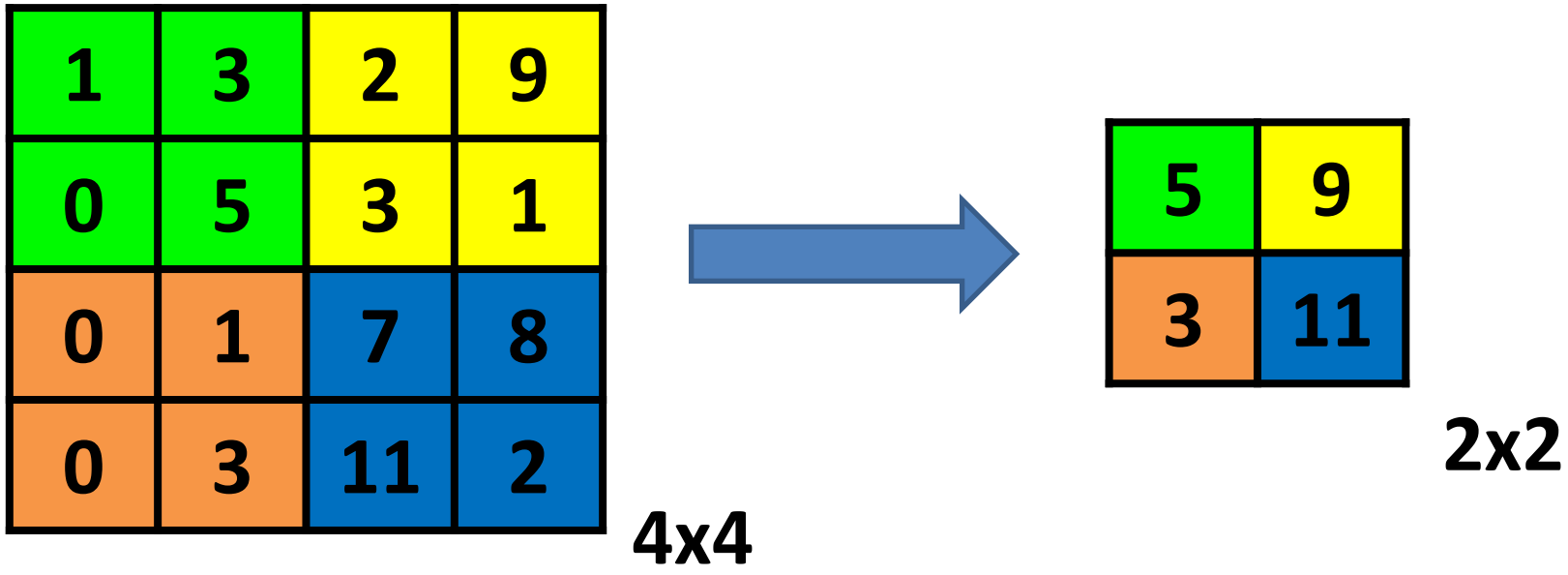



ReLU(
+ b1)
ReLU(
+ b2)
Convolution Layer
Activation Function
(ReLU) Layer


Convolution
ReLU
Pooling
Convolution
ReLU
Pooling
. . .
Convolution
ReLU
Pooling
Fully Connected Layer (FC) for classification
Cat
Dog
Bird
Horse

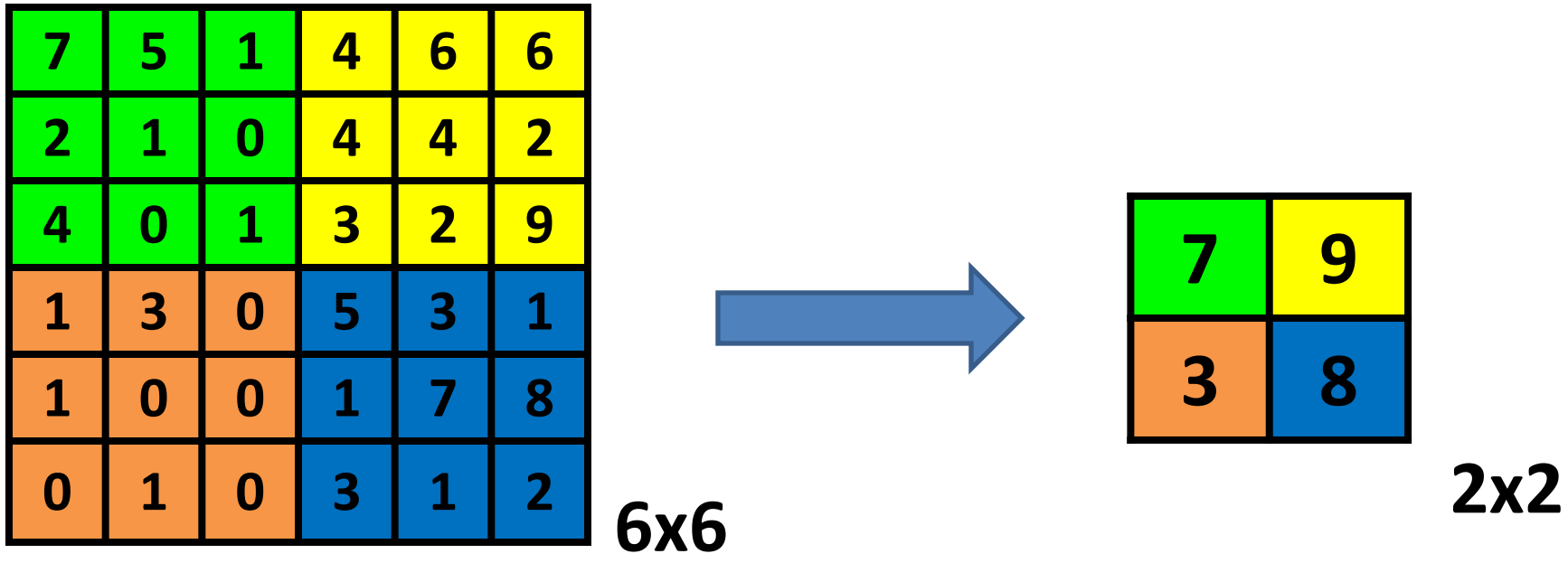
Pooling Layer (Subsampling): Max Pooling

- **Pooling** is usually used for dimensionality reduction in each layer of CNN. Here is an example of 2x2 **Max Pooling:** the algorithm selects the **maximum value** in each 2x2 block of the original matrix:

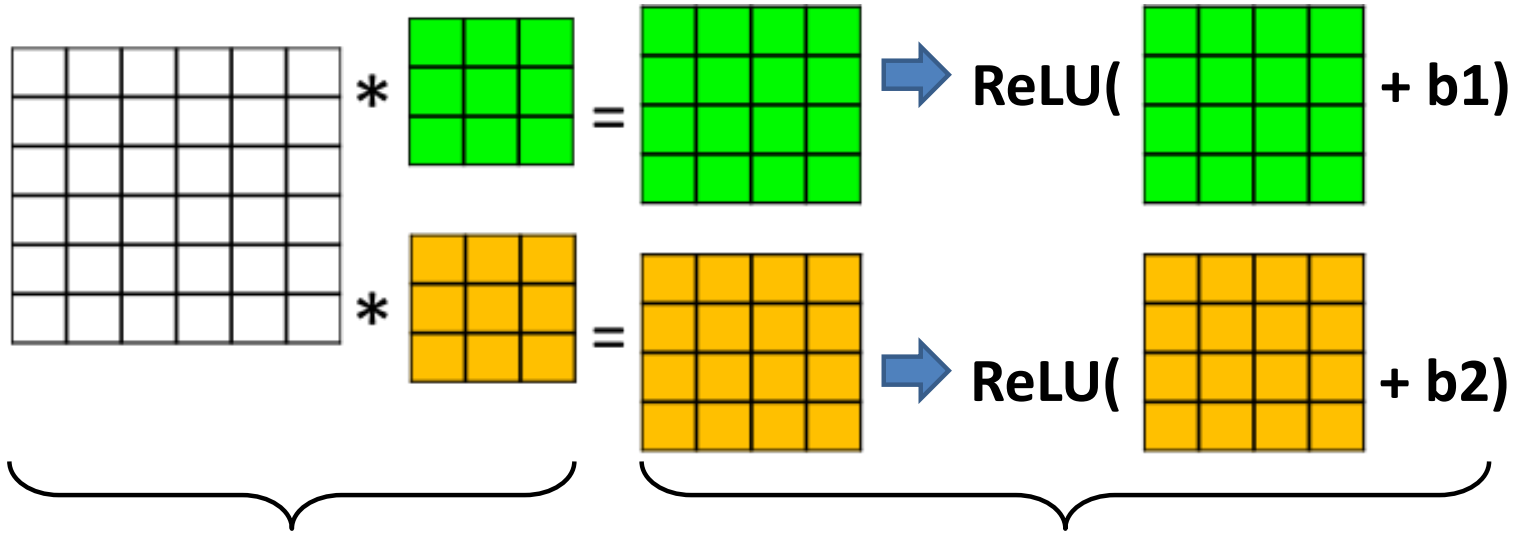


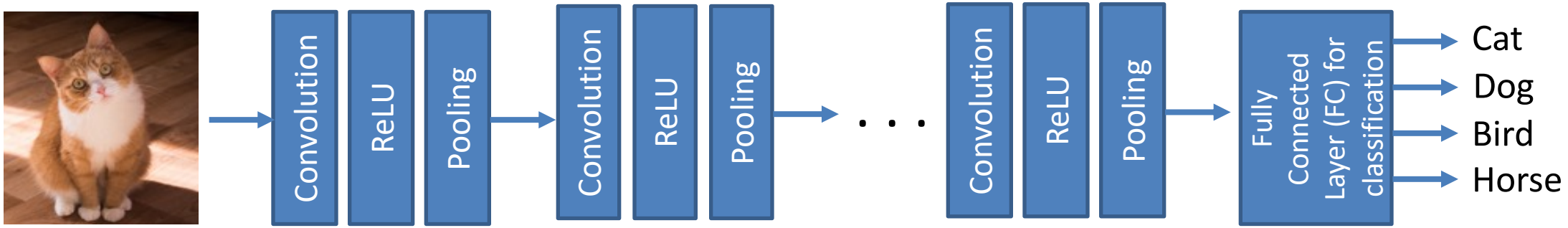
Pooling Layer (Subsampling): Max Pooling

- **Example of 3x3 Max Pooling:**

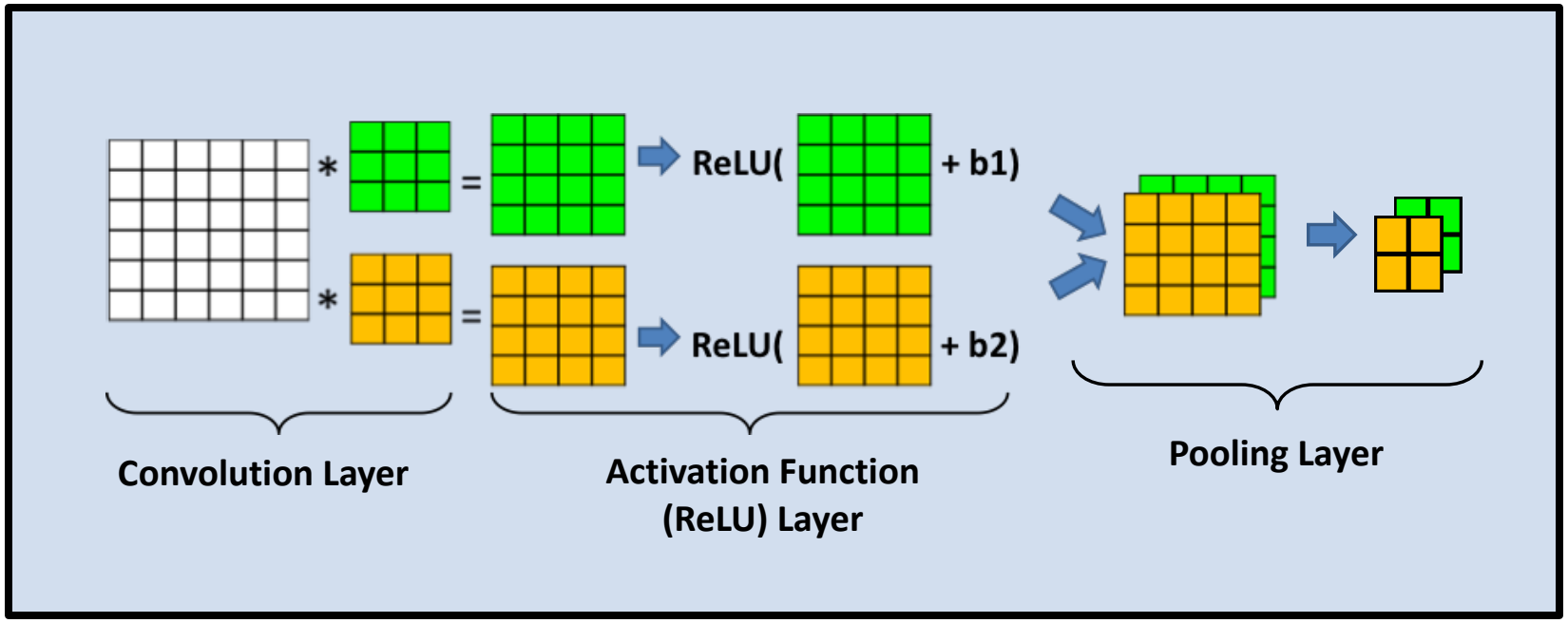


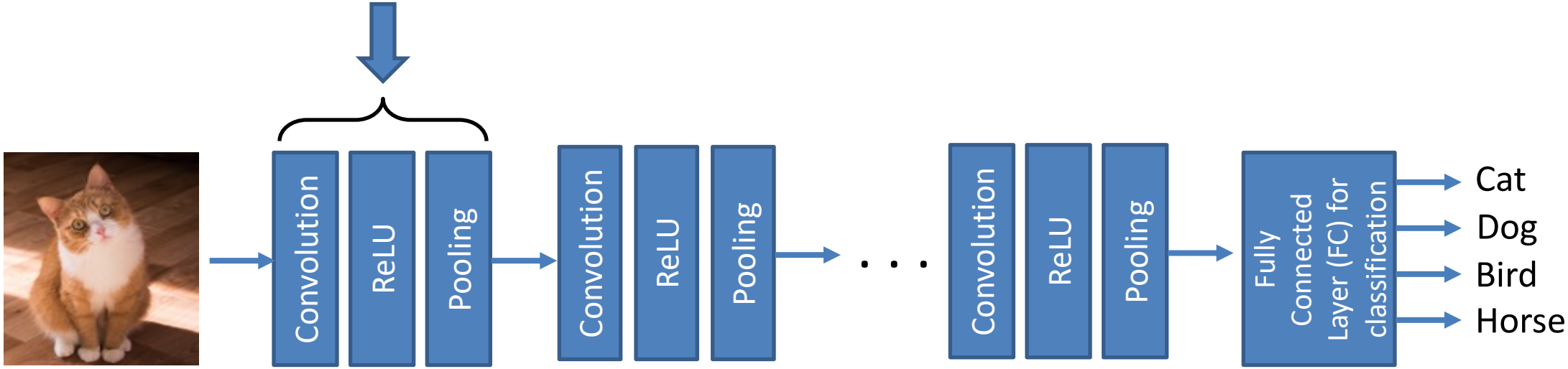
- **Notice**: Pooling applies on every channel of the input **independently**. So, if the input size is 6x6x10 (10 filters), then by applying 3x3 pooling, the output will be 2x2x10. So, it only shrinks the height and width.


ReLU(+ b1)
ReLU(+ b2)
Convolution Layer
Activation Function (ReLU) Layer
Pooling Layer


Convolution
ReLU
Pooling
Convolution
ReLU
Pooling
. . .
Convolution
ReLU
Pooling
Fully Connected Layer (FC) for classification
Cat
Dog
Bird
Horse

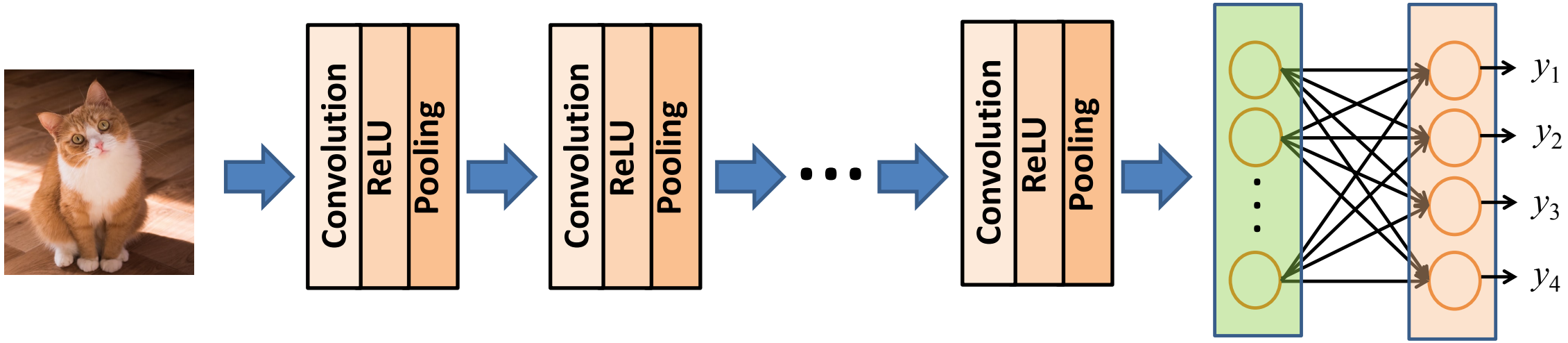
One Layer of CNN:



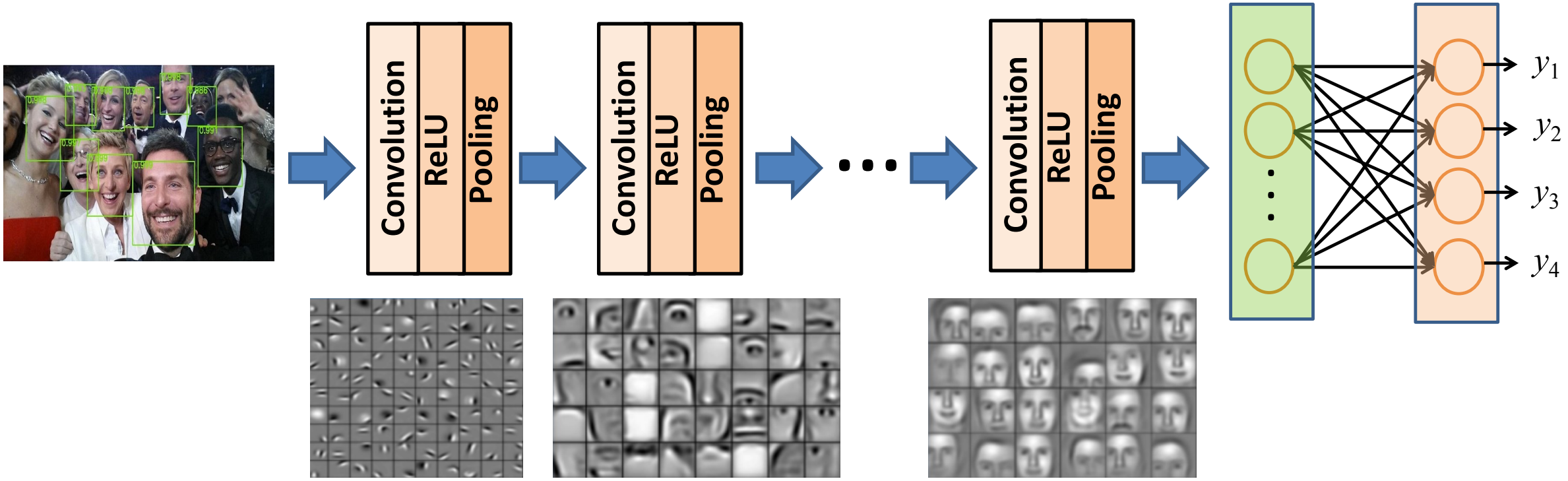


Last Layer: Fully Connected Layer

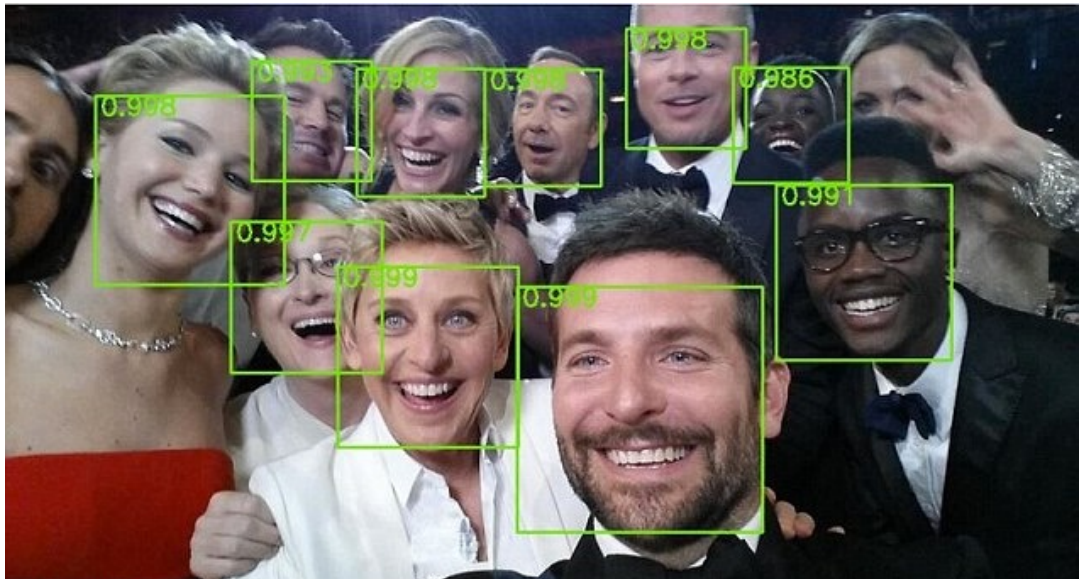
- After **Several rounds of Convolutional Layers**, we flatten all generated values into a final **row of features**, and then use a regular **Fully Connected Network** including a logistic/Sigmoid or Softmax layer for final classification:



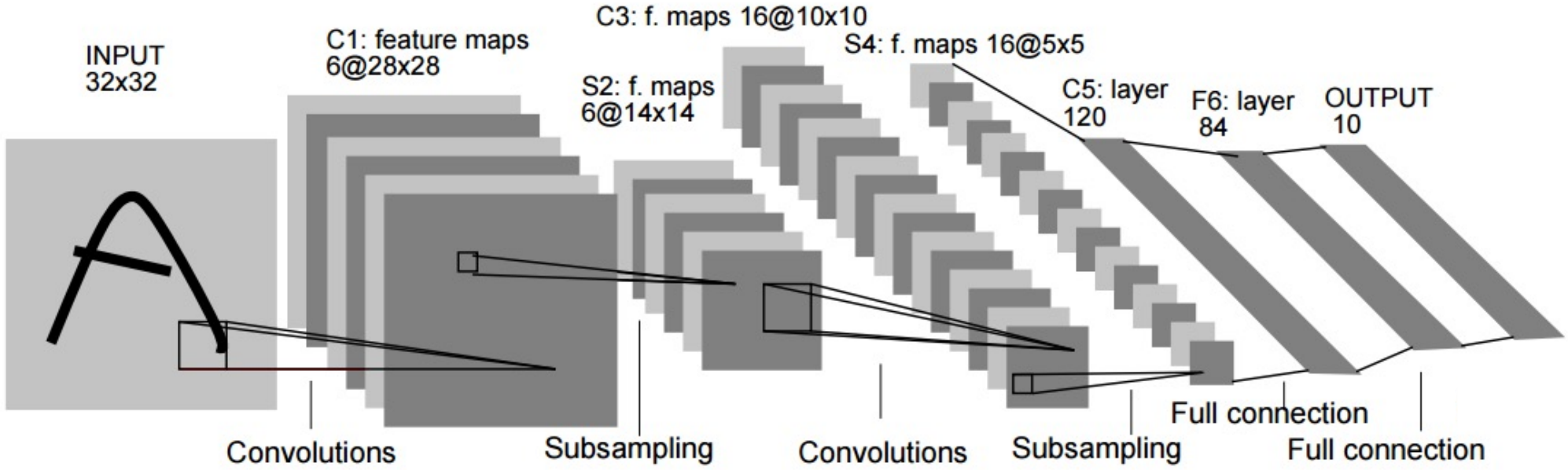
- As mentioned before, each layer progressively extracts higher and higher-level features. Early layers look for very simple patterns (e.g. edges), middle layers combine simple patterns to find more complex patterns (e.g. eye, nose in a face recognition model), and final layers try to combine previous patterns to find and recognize high level patterns (e.g. a face)! Finally, the last layer makes a final decision based on the most advanced patterns find in the data!



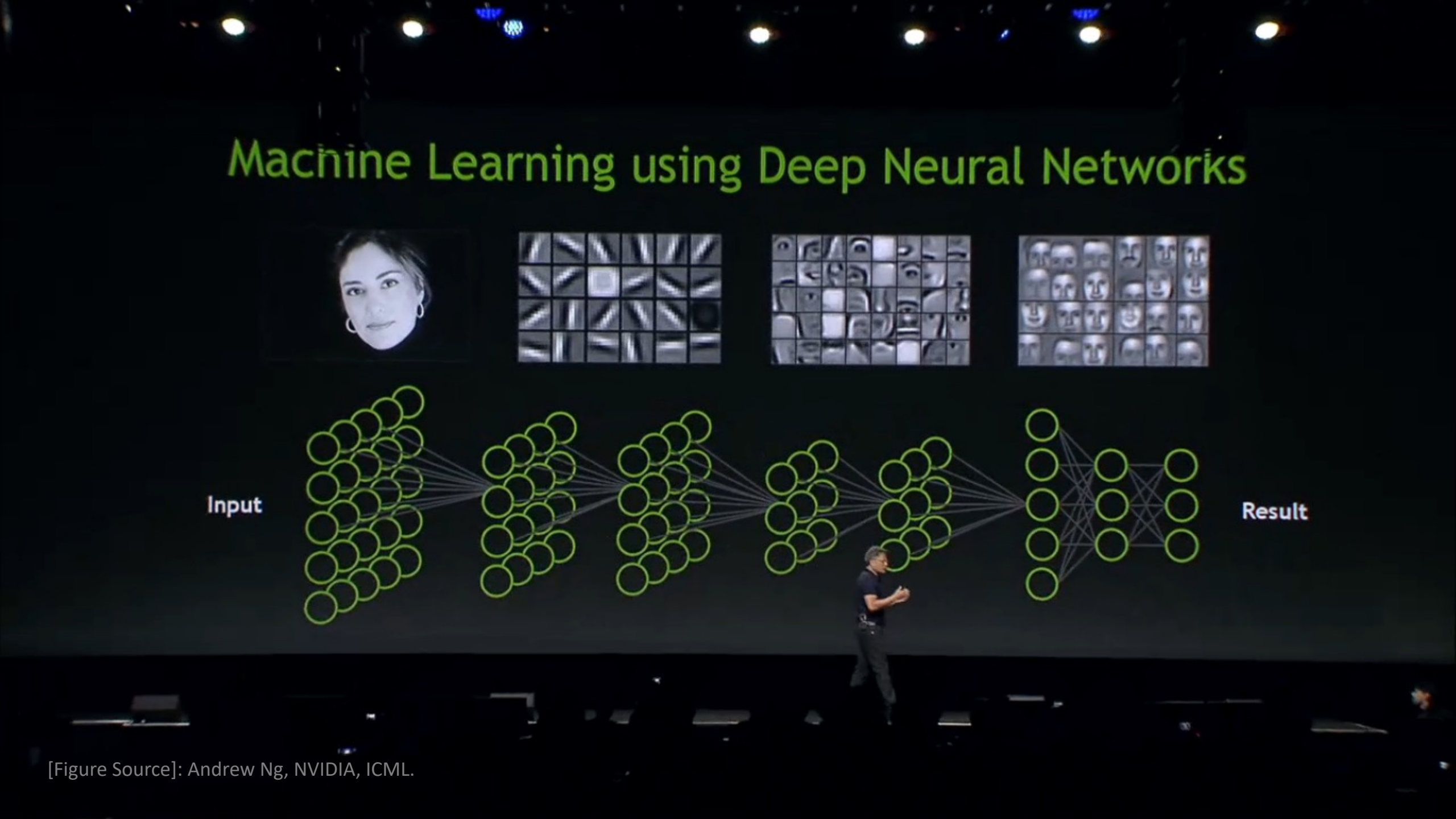
Example: Face Recognition



Convolutional Neural Networks



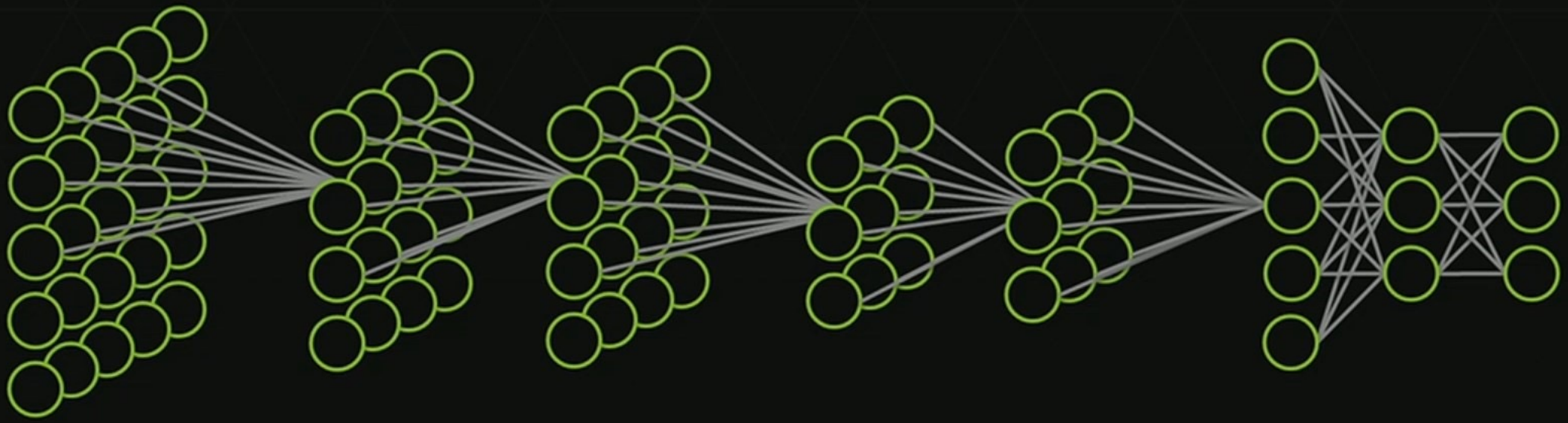
[Ref]: Yann LeCun et al., Gradient-based learning applied to document recognition



[Figure Source]: Andrew Ng, NVIDIA, ICML.

HOW A DEEP NEURAL NETWORK SEES

Image



"Audi A7"

Image source: "Unsupervised Learning of Hierarchical Representations with Convolutional Deep Belief Networks" ICML 2009 & Comm. ACM 2011. Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Ng.

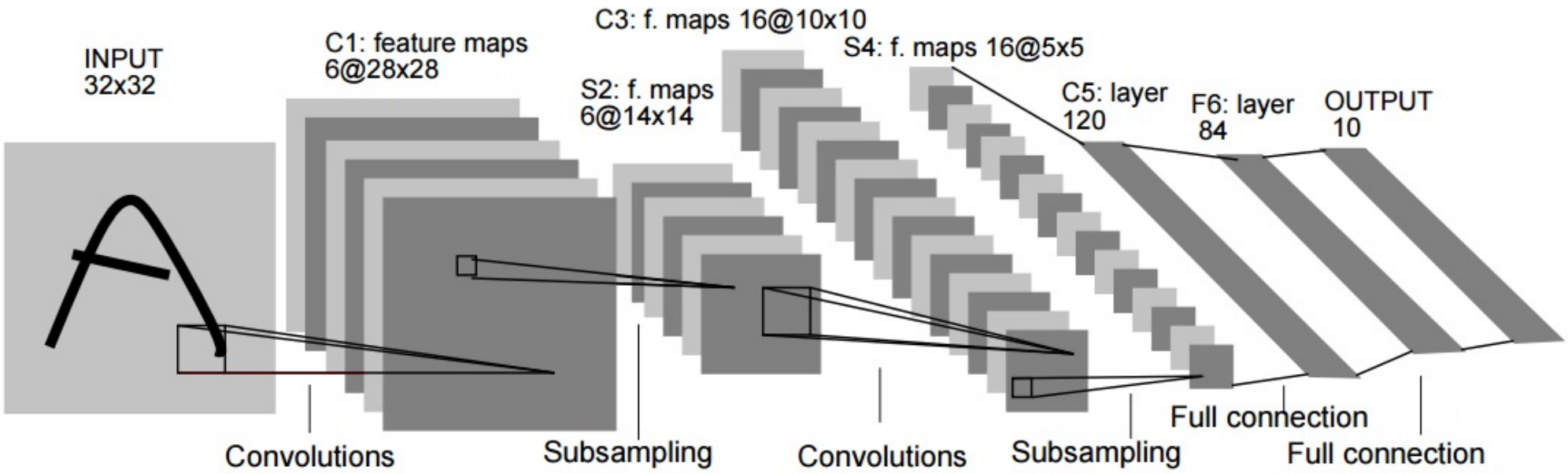
Thanks to …

And many others …

Some Well known CNN Models

Some Well known CNN Models

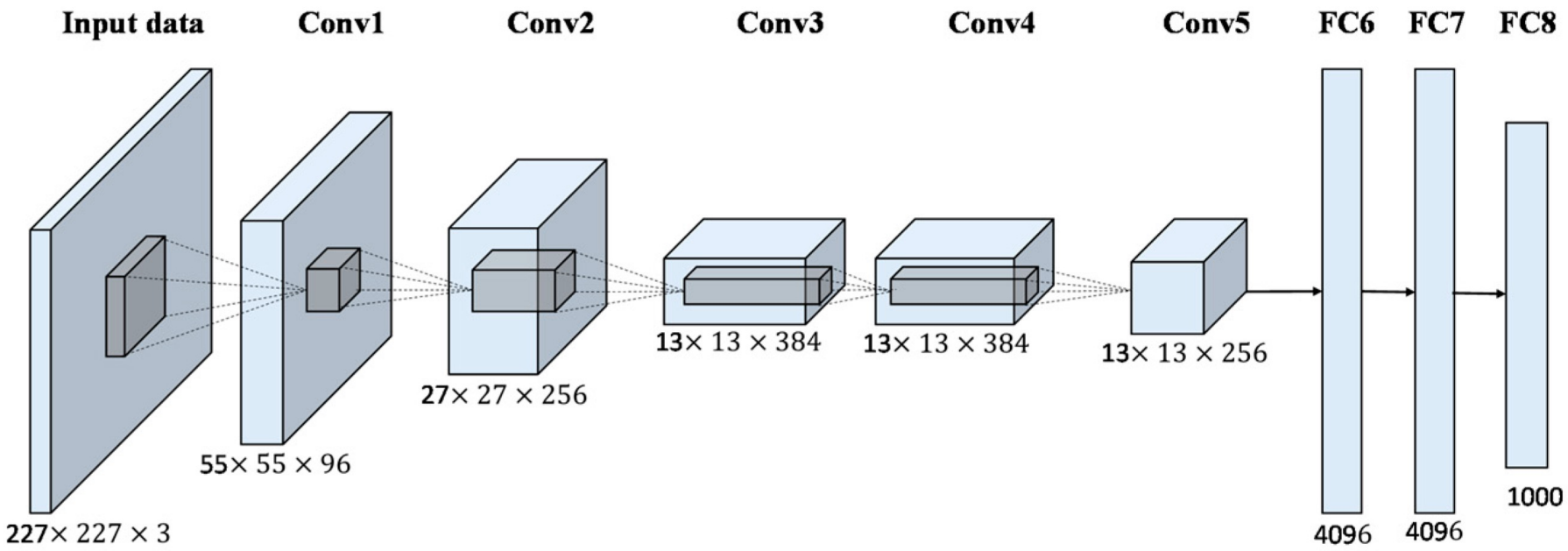
- **LeNet-5** proposed by Yann LeCun:



[Ref]: Yann LeCun et al., Gradient-based learning applied to document recognition

Some Well known CNN Models

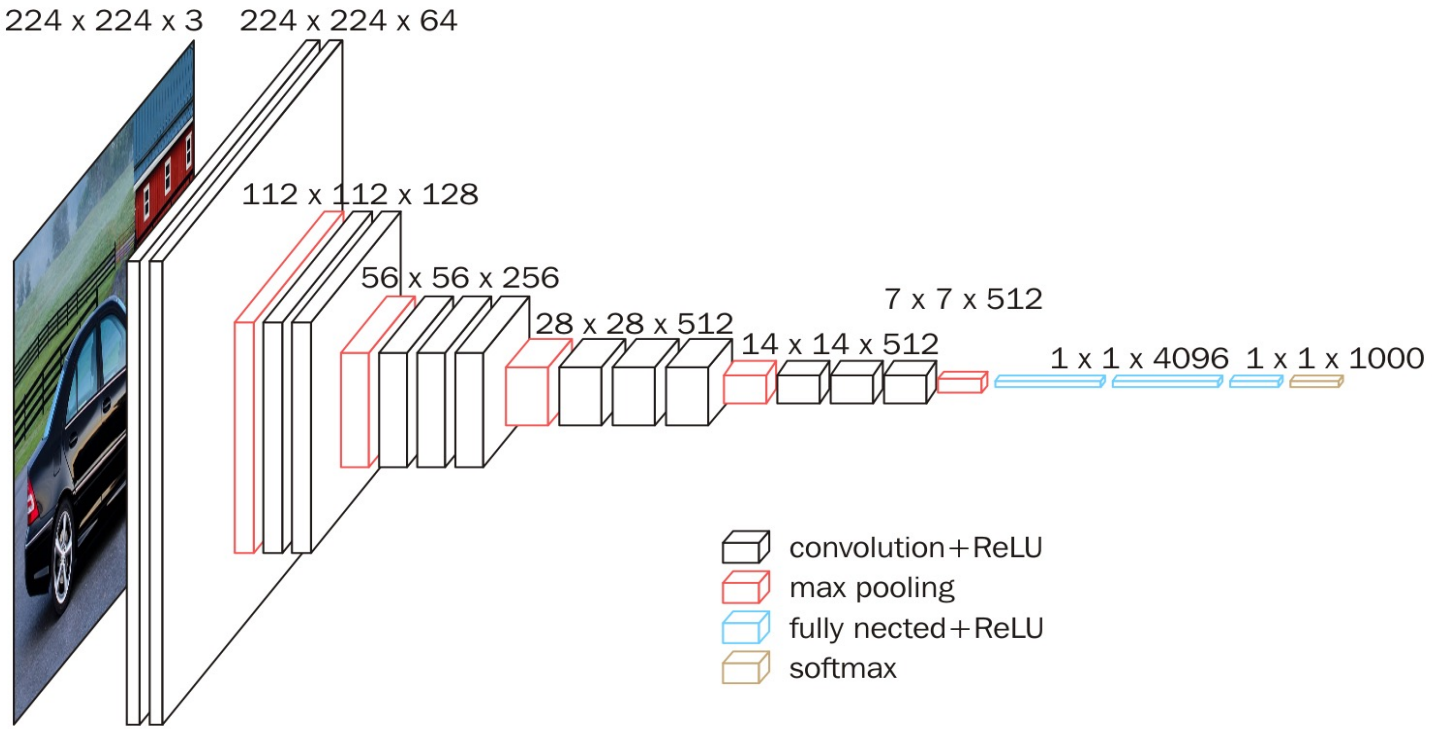
- **AlexNet** proposed by Alex Krizhevsky and Jef Hinton:



[Ref]: *Alex Krizhevsky, Ilya Sutskever, Geff Hinton*. "ImageNet classification with deep convolutional neural networks"

Some Well known CNN Models

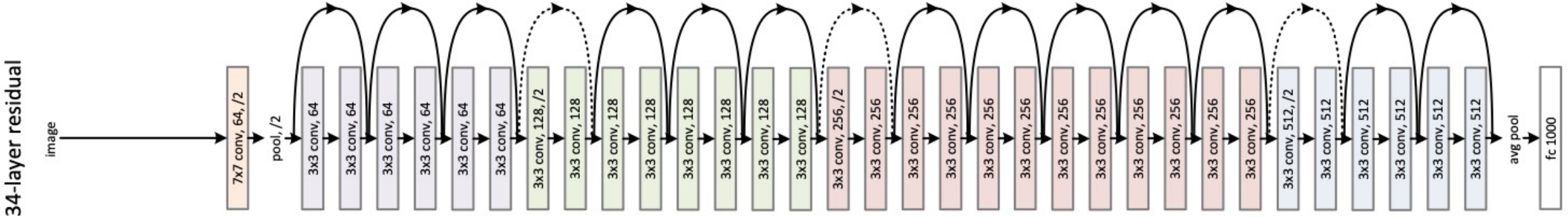
- **VGG-16** and **VGG-19** proposed by Simonyan and Zisserman:



[Ref]: Simonyan & Zisserman 2015. "Very deep convolutional networks for large-scale image recognition," Visual Geometry Group, Univ of Oxford

Some Well known CNN Models

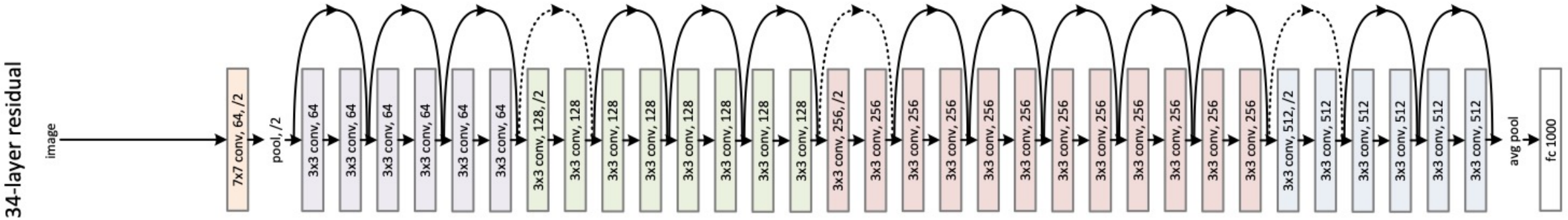
- **ResNet-(18,34,50,101, 152): Residual CNNs.**
  - *Includes "**skip connections**" ("**shortcuts**")* to jump over some layers.
  - Significantly improves the Vanishing Gradients and the Degradation (accuracy saturation) problems (where adding more layers leads to higher training error).



[Ref]: Kaiming He, et al. "Deep Residual Learning for Image Recognition", Microsoft Research team.

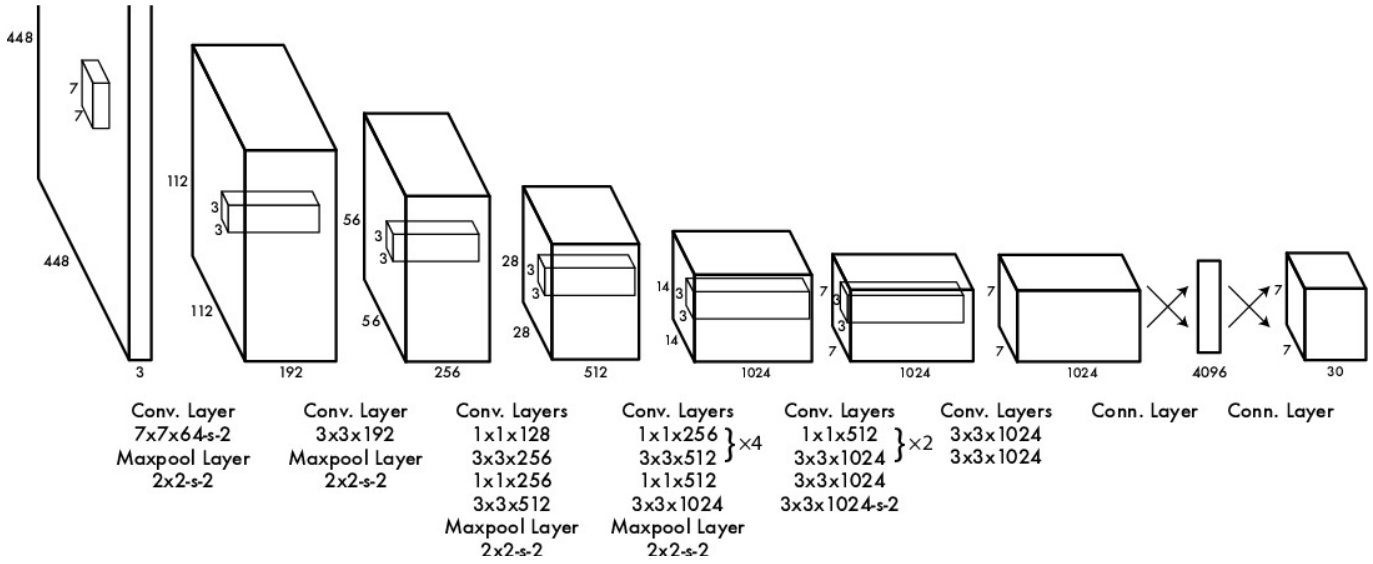
Some Well known CNN Models

- **ResNet-(18,34,50,101, 152)**: Residual CNNs.
  - ***Shortcuts*** simplifies the network in the initial training rounds. This speeds up the training process by reducing the impact of vanishing gradients. The network then gradually restores the skipped layers as it learns more and tries to achieve better accuracy in next rounds.
  - **Some parts of the brain in cerebral cortex has the same structure!**



Some Well known CNN Models

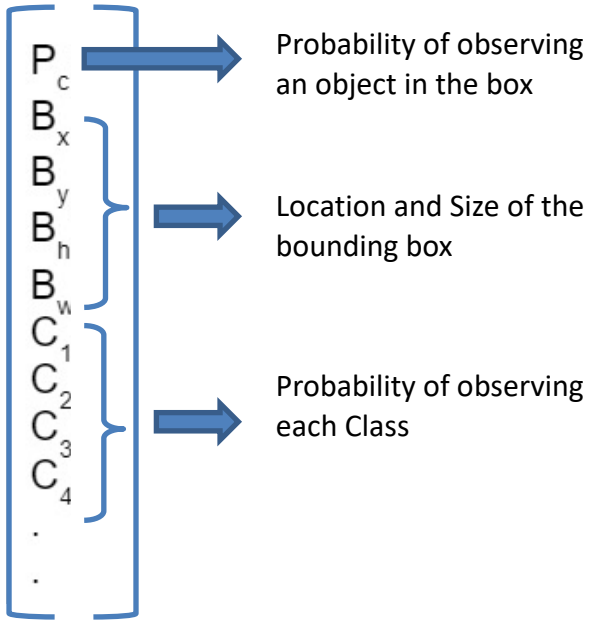
- **YOLO: You Only Look Once**
  - Yolo considers object detection as a regression problem to assign class probabilities to spatially separated bounding boxes. YOLO originally includes 24 convolutional layers.

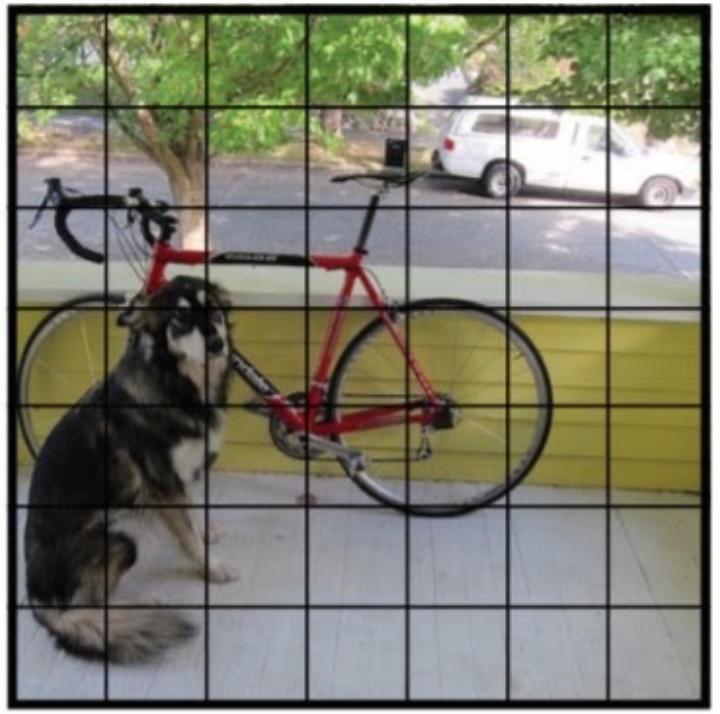


[Ref]: J. Redmon, S. Divvala, R. Girshick, A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection"
https://pjreddie.com/darknet/yolo/

Some Well known CNN Models

- **YOLO: You Only Look Once**
  - YOLO can detect objects and their locations/sizes simultaneously. YOLO sees the entire image during training and testing.

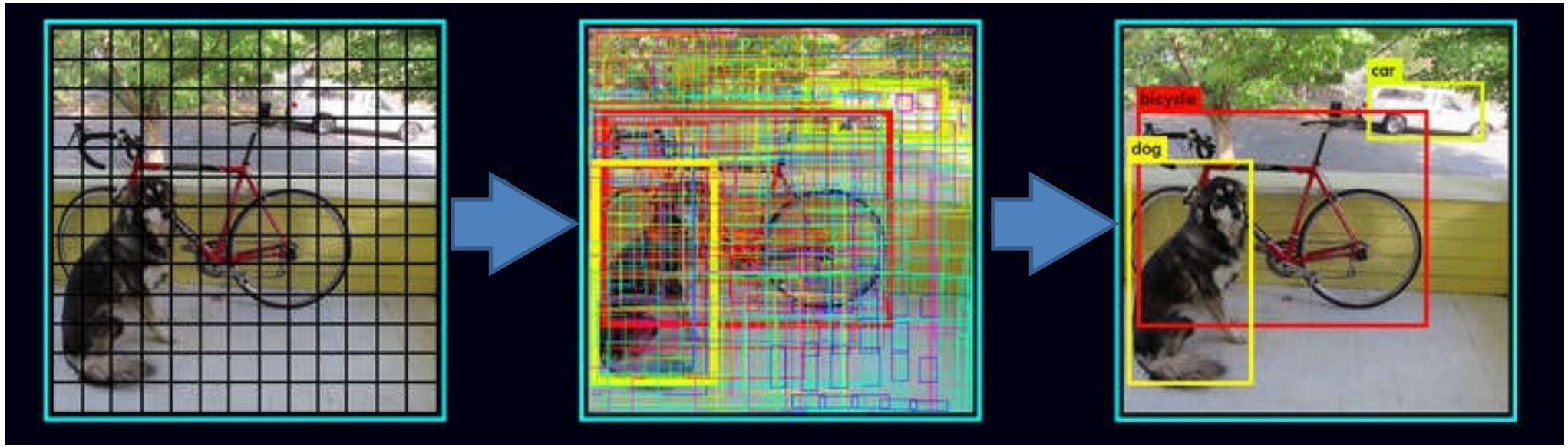




[Ref]: J. Redmon, S. Divvala, R. Girshick, A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection"

Some Well known CNN Models

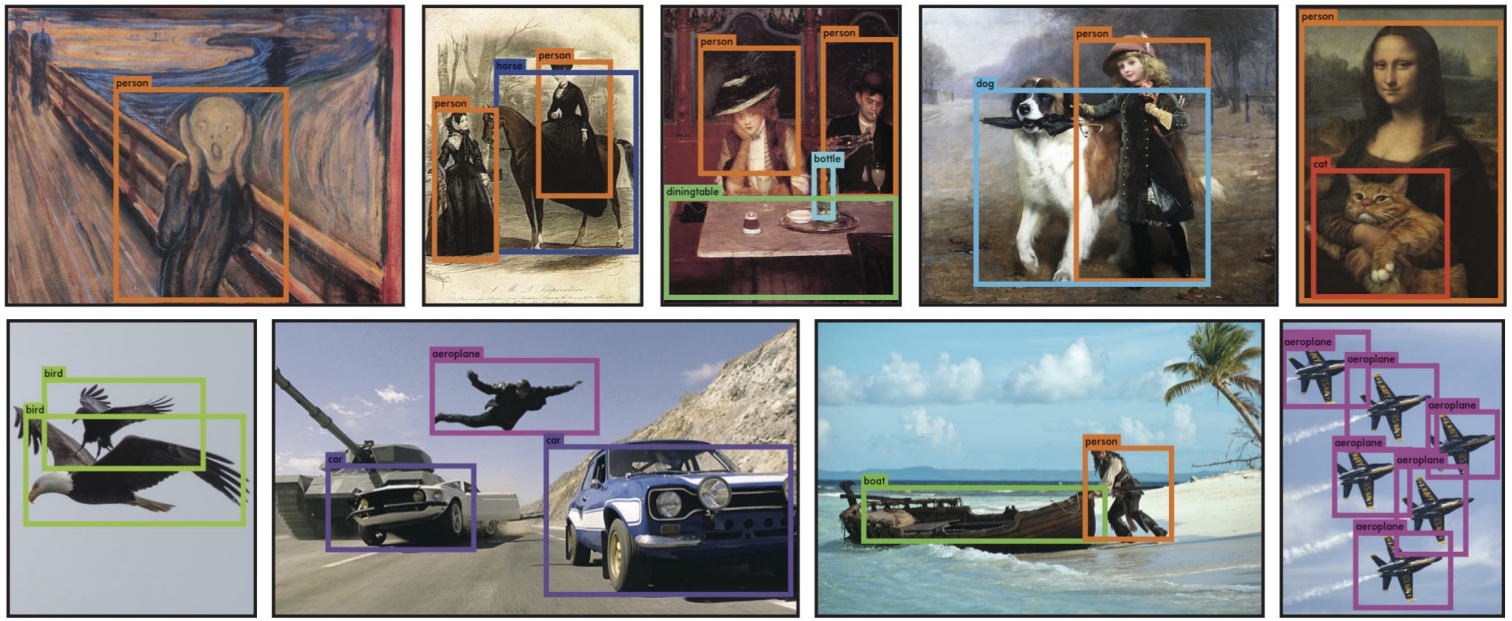
- **YOLO: You Only Look Once**
  - YOLO can detect objects and their locations/sizes simultaneously. YOLO sees the entire image during training and testing.



[Ref]: J. Redmon, S. Divvala, R. Girshick, A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection"

Some Well known CNN Models

- **YOLO: You Only Look Once**
  - It lets the algorithm to be fast. So, It is an ideal algorithm for real-time object detection.



[Ref]: J. Redmon, S. Divvala, R. Girshick, A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection"

My Important Advice

- **My last Advice: Use Machine Learning, Data Science, and AI to help and benefit people and society for good!**

[Figure Ref]: UN.

Thank You!

Questions?