# Lecture 8: Convolutional Neural Networks 1

## CS109B Data Science 2

Pavlos Protopapas and Mark Glickman

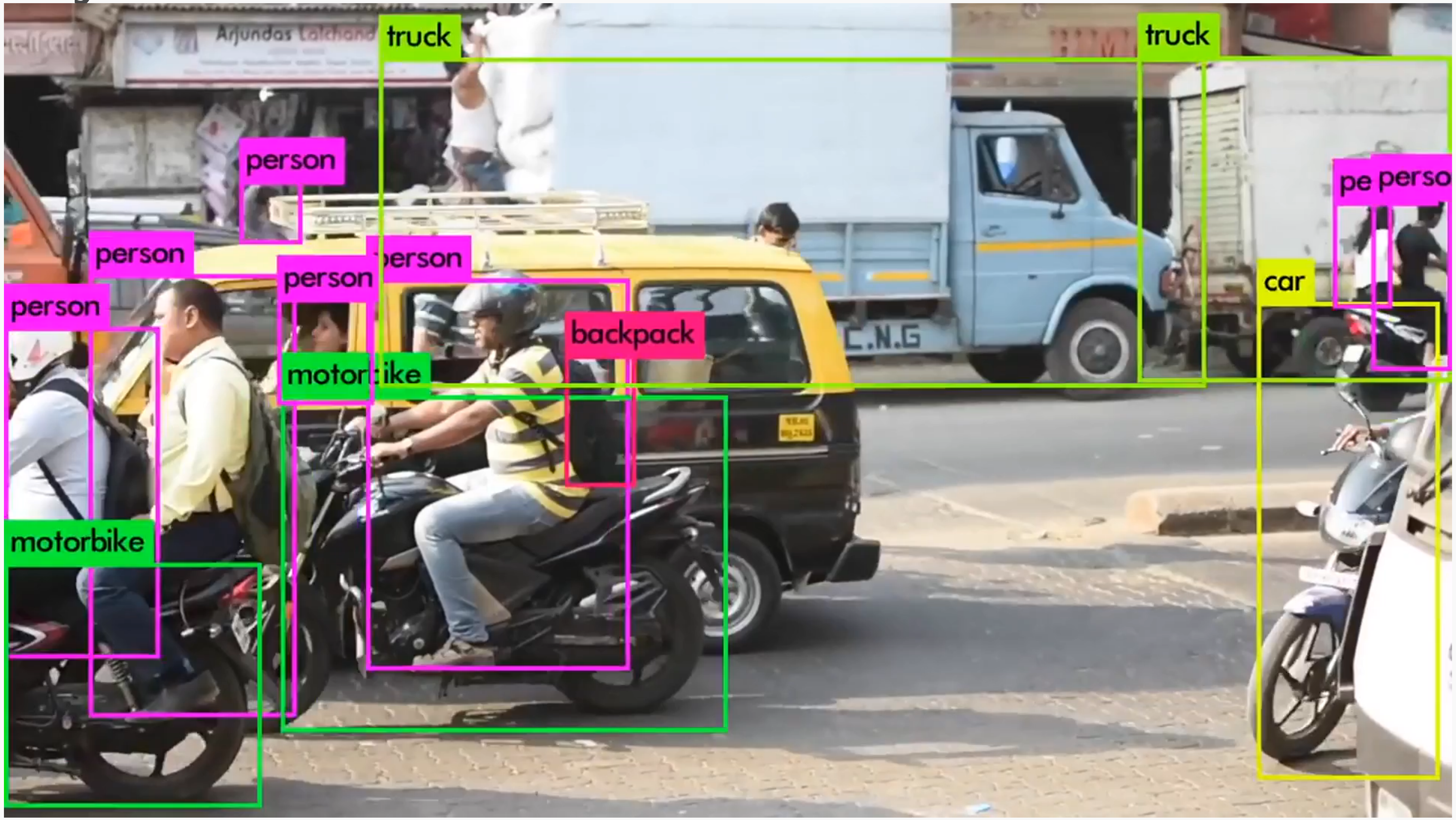
# Outline

# Main **drawbacks** of MLPs

- MLPs use one perceptron for each input (e.g. pixel in an image, multiplied by 3 in RGB case). The amount of weights rapidly becomes unmanageable for large images.
- Training difficulties arise, overfitting can appear.
- MLPs react differently to an input (images) and its shifted version – they are not translation invariant.

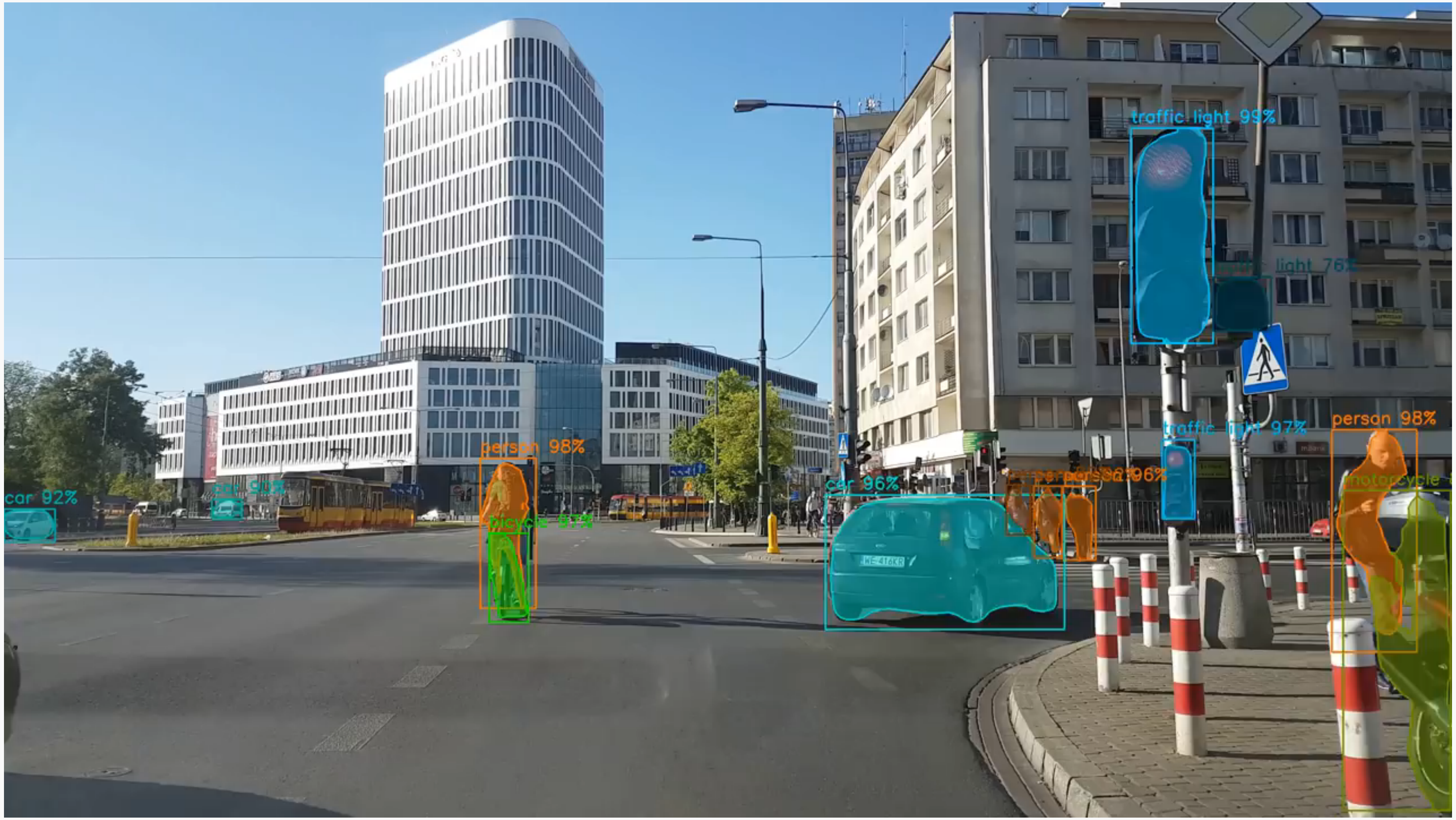
# Latest events on Image Recognition

## You Only Look Once (YOLO) - 2016



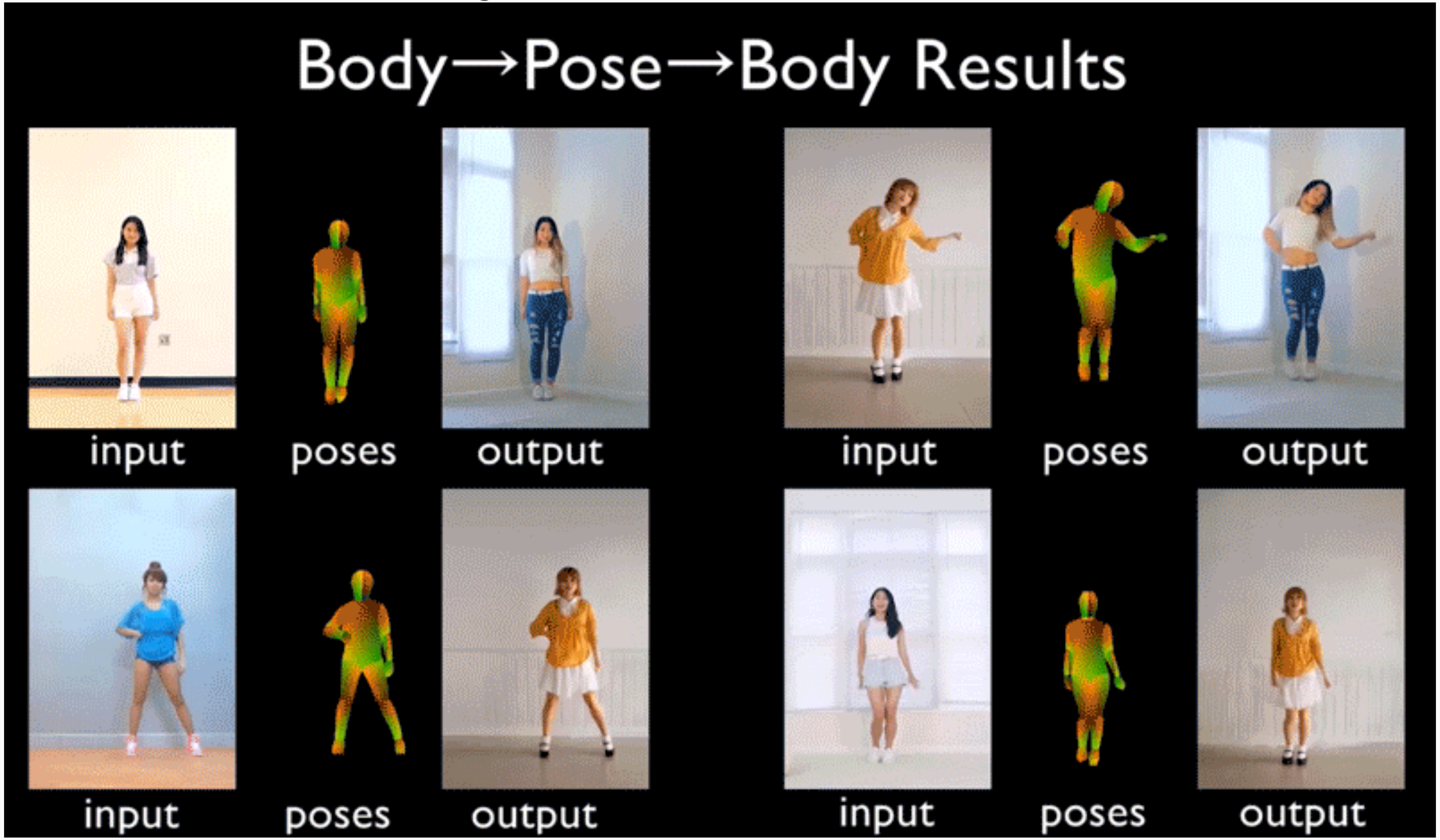
# Latest events on Image Recognition

## Mask- RCNN - 2017



# Latest events on Image Recognition

## NVIDIA Video to Video Synthesis - 2018



# Image analysis

Imagine that we want to recognize swans in an image:

Oval-shaped white blob (body)

Round, elongated oval with orange protuberance

Long white rectangular shape (neck)

# Cases can be a bit more complex…

Round, elongated head with orange or black beak

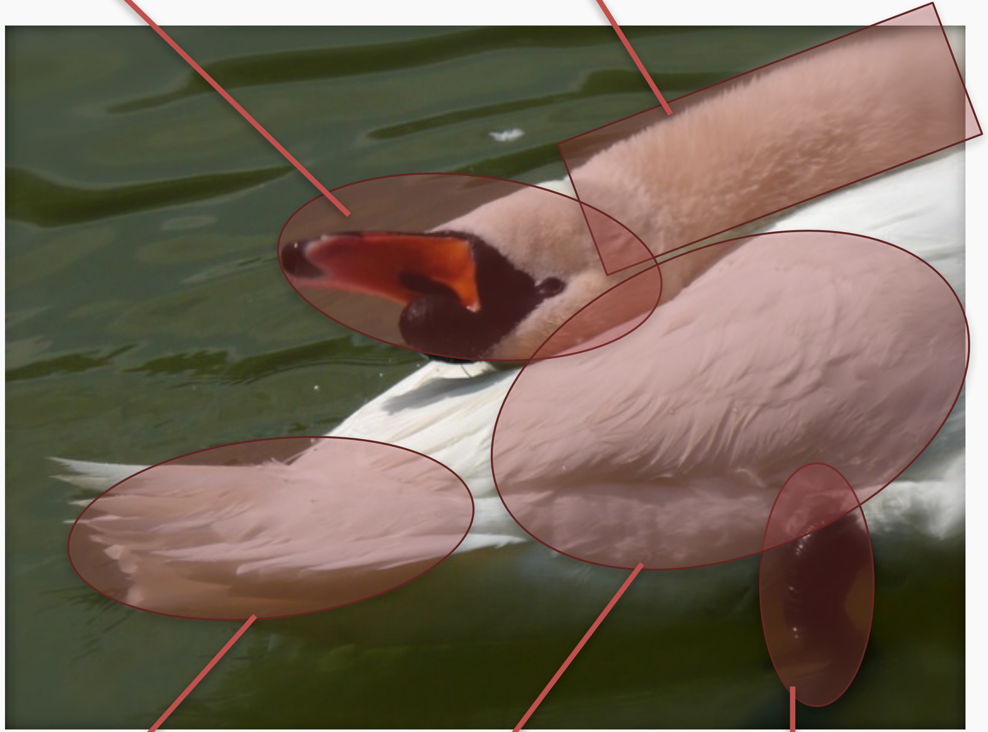
Oval-shaped white body with or without large white symmetric blobs (wings)

Long white neck, square shape

# Now what?

Round, elongated head with orange or black beak, can be turned backwards

Long white neck, can bend around, not necessarily straight



White tail, generally far from the head, looks feathery

White, oval shaped body, with or without wings visible

Black feet, under body, can have different shapes

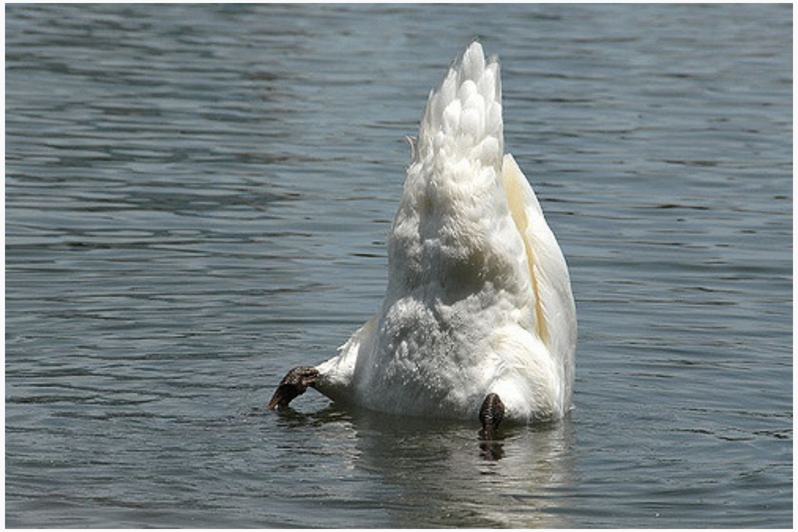
Small black circles, can be facing the camera, sometimes can see both

Black triangular shaped form, on the head, can have different sizes

White elongated piece, can be squared or more triangular, can be obstructed sometimes

Luckily, the color is consistent…

# We need to be able to deal with these cases.

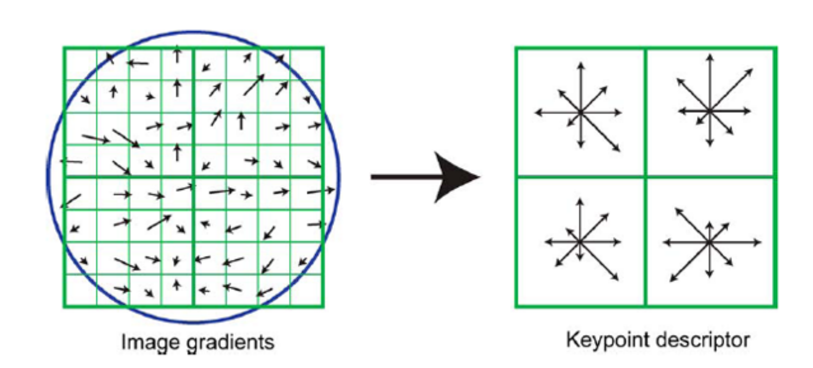


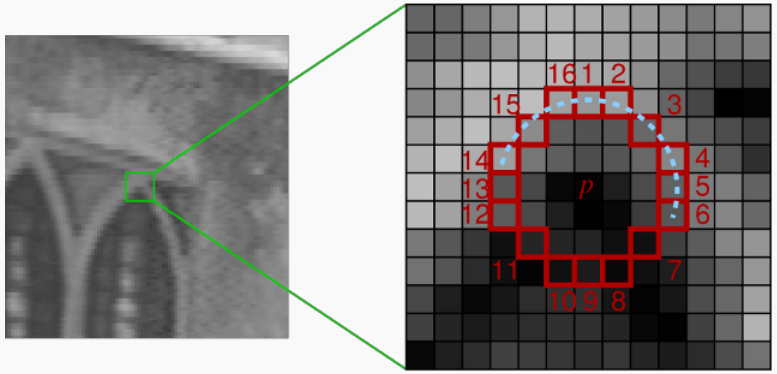
Man in swan tent photographing swans

# Image features

- We've been basically talking about detecting features in images, in a very naïve way.
- Researchers built multiple computer vision techniques to deal with these issues: SIFT, FAST, SURF, BRIEF, etc.
- However, similar problems arose: the detectors where either too general or too over-engineered. Humans were designing these feature detectors, and that made them either too simple or hard to generalize.

SIFT feature descriptor




FAST corner detection algorithm

# Image features (cont)

- What if we learned the features to detect?
- We need a system that can do Representation Learning (or Feature Learning).

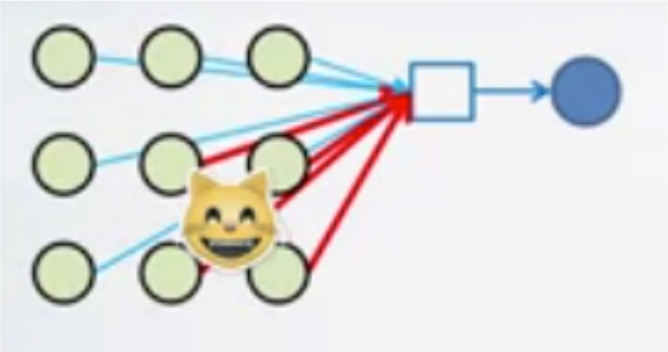
Representation Learning: technique that allows a system to automatically find relevant features for a given task. Replaces manual feature engineering.

Multiple techniques for this:

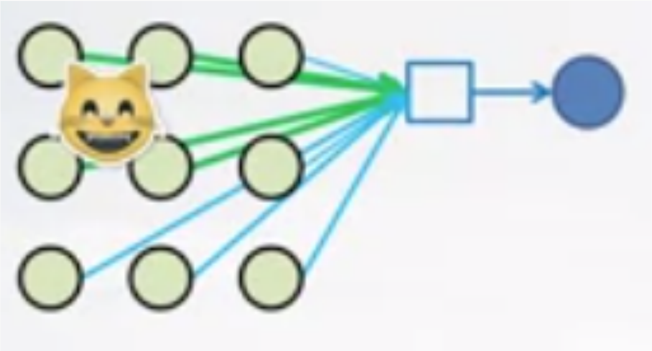
- Unsupervised (K-means, PCA, ...).
- Supervised (Sup. Dictionary learning, Neural Networks!)

# Drawbacks

Imagine we want to build a cat detector with an MLP.



In this case, the red weights will be modified to better recognize cats



In this case, the green weights will be modified.

We are learning redundant features. Approach is not robust, as cats could appear in yet another position.

# Drawbacks

Example: CIFAR10

Simple 32x32 color images (3 channels)

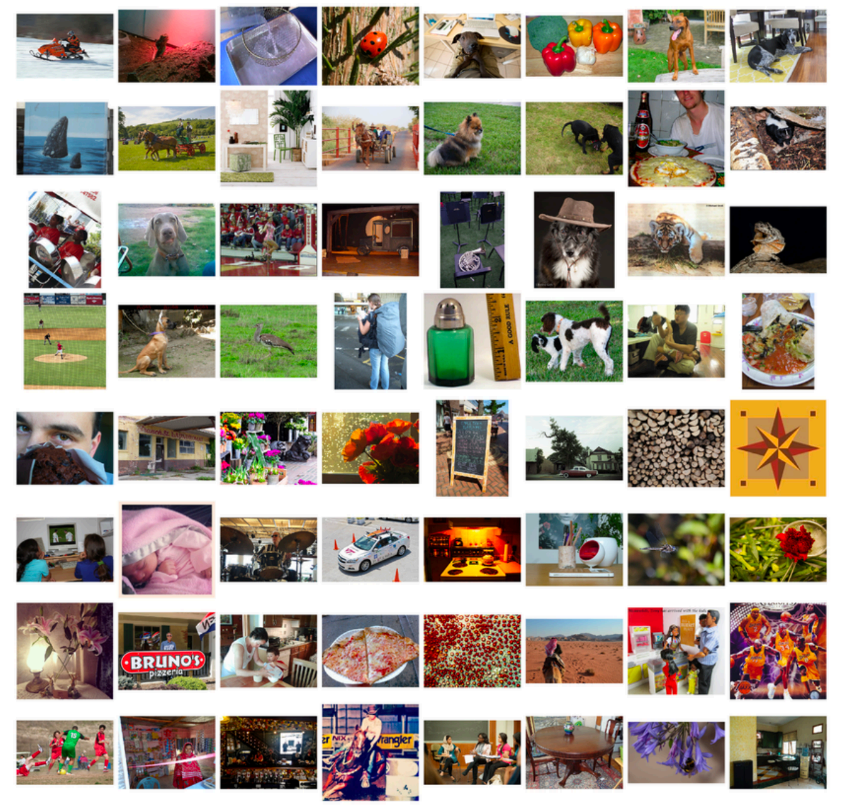
Each pixel is a feature: an MLP would have 32x32x3+1 = 3073 weights per neuron!

# Drawbacks

Example: ImageNet

Images are usually 224x224x3: an MLP would have 150129 weights per neuron. If the first layer of the MLP is around 128 nodes, which is small, this already becomes very heavy to calculate.

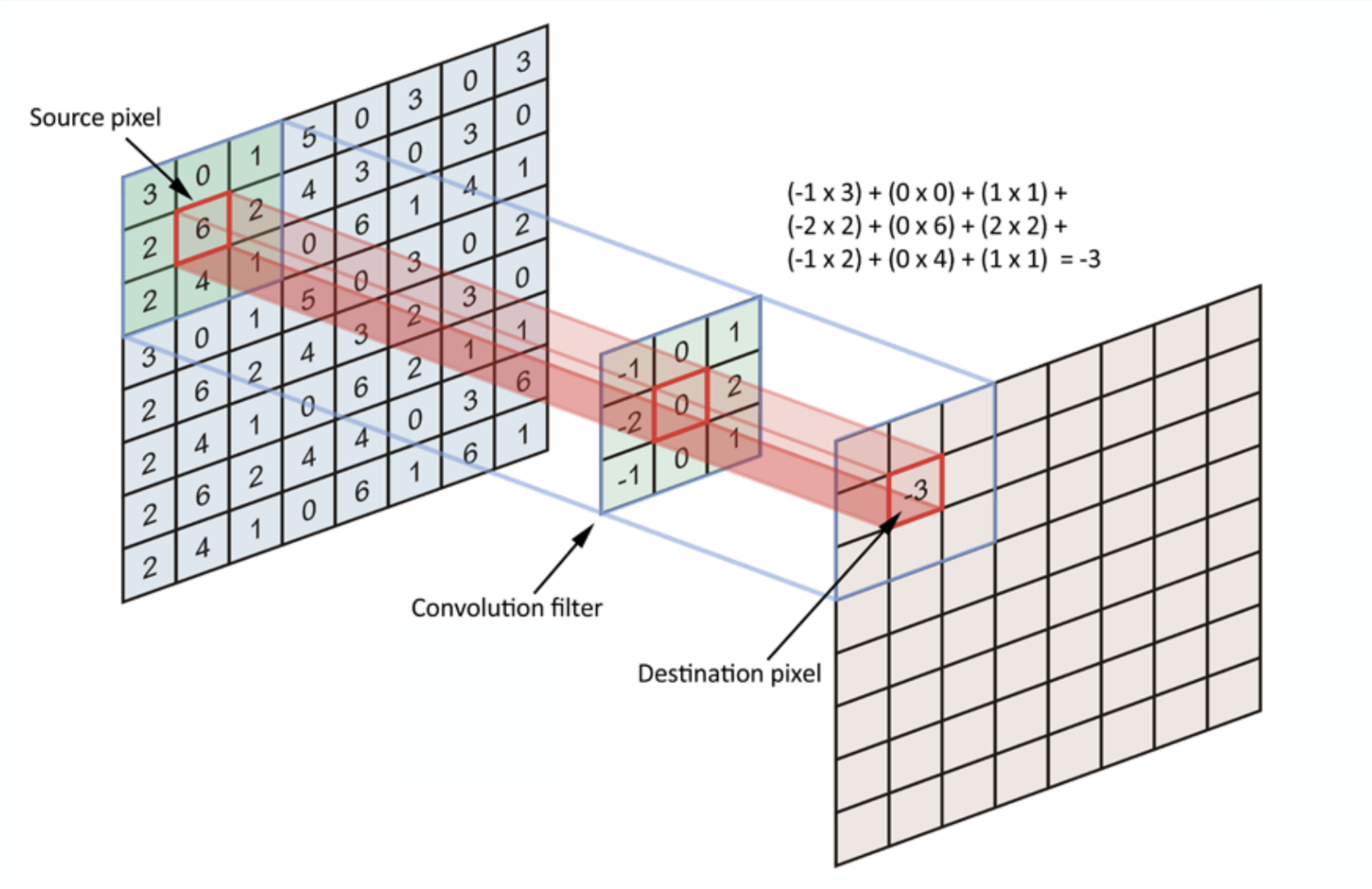
Model complexity is extremely high: overfitting.




Nearby pixels are more strongly related than distant ones.

Objects are built up out of smaller parts.

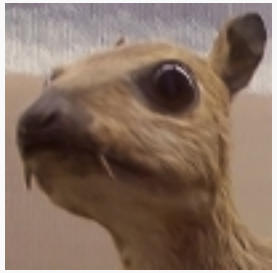
# “Convolution” Operation



# "Convolution" Operation

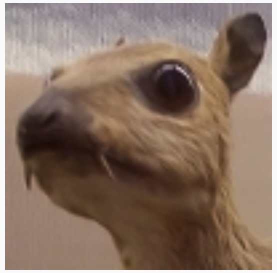
Kernel

*Edge detection*

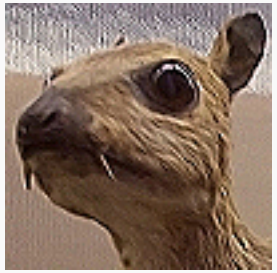


$$* \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} =$$

*Sharpen*



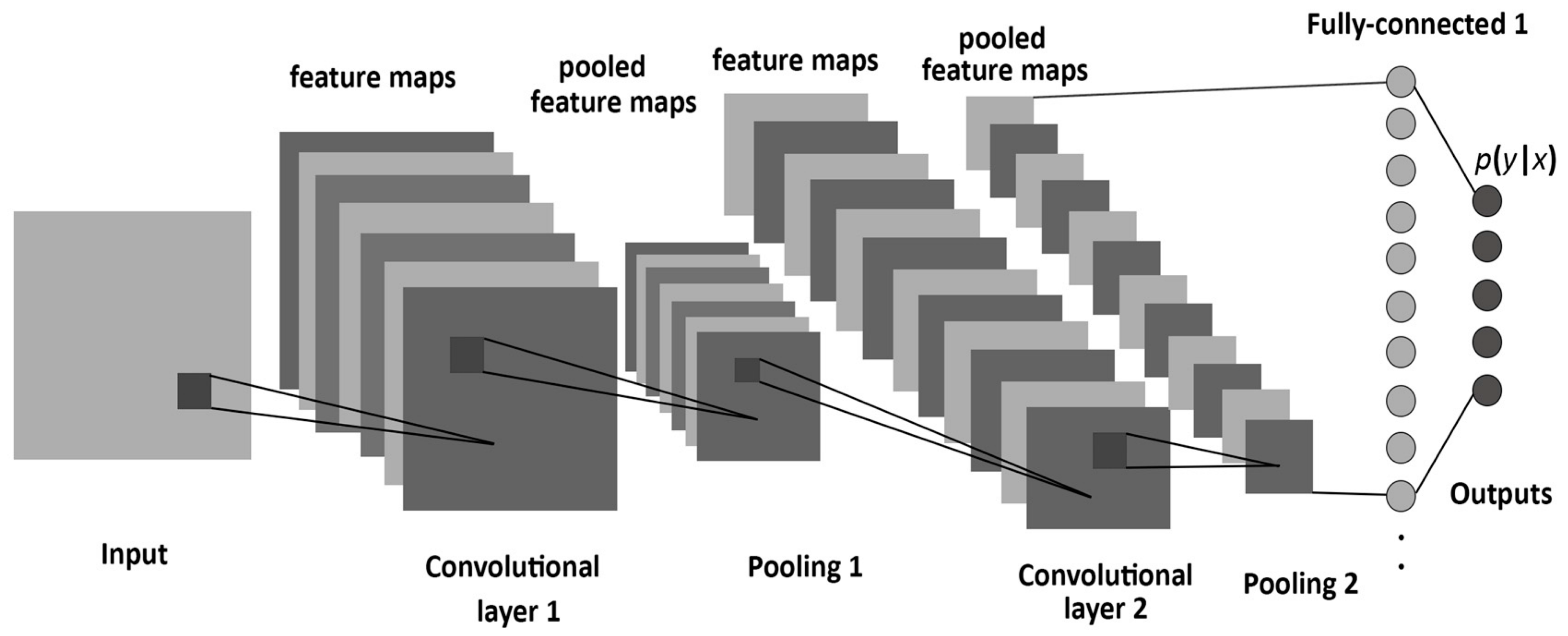
$$* \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix} =$$





wikipedia.org

# A Convolutional Network




+ ReLU

+ ReLU

# Basics of CNNs

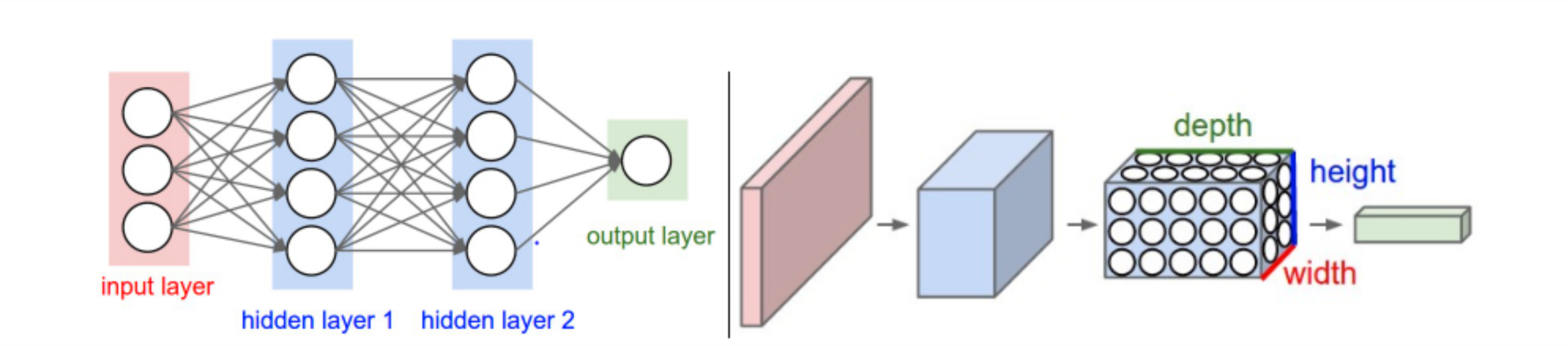
We know that MLPs:

- Do not scale well for images
- Ignore the information brought by pixel position and correlation with neighbors
- Cannot handle translations

The general idea of CNNs is to intelligently adapt to properties of images:

- Pixel position and neighborhood have semantic meanings.
- Elements of interest can appear anywhere in the image.

# Basics of CNNs




MLP CNN

CNNs are also composed of layers, but those layers are not fully connected: they have filters, sets of cube-shaped weights that are applied throughout the image. Each 2D slice of the filters are called kernels.

These filters introduce translation invariance and parameter sharing.

How are they applied? Convolutions!

# Convolution and cross-correlation

- A **convolution** of f and g $(f * g)$ is defined as the integral of the product, having one of the functions inverted and shifted:

$$(f * g)(t) = \int_a f(a)g(t - a)da$$

Function is inverted and shifted left by t

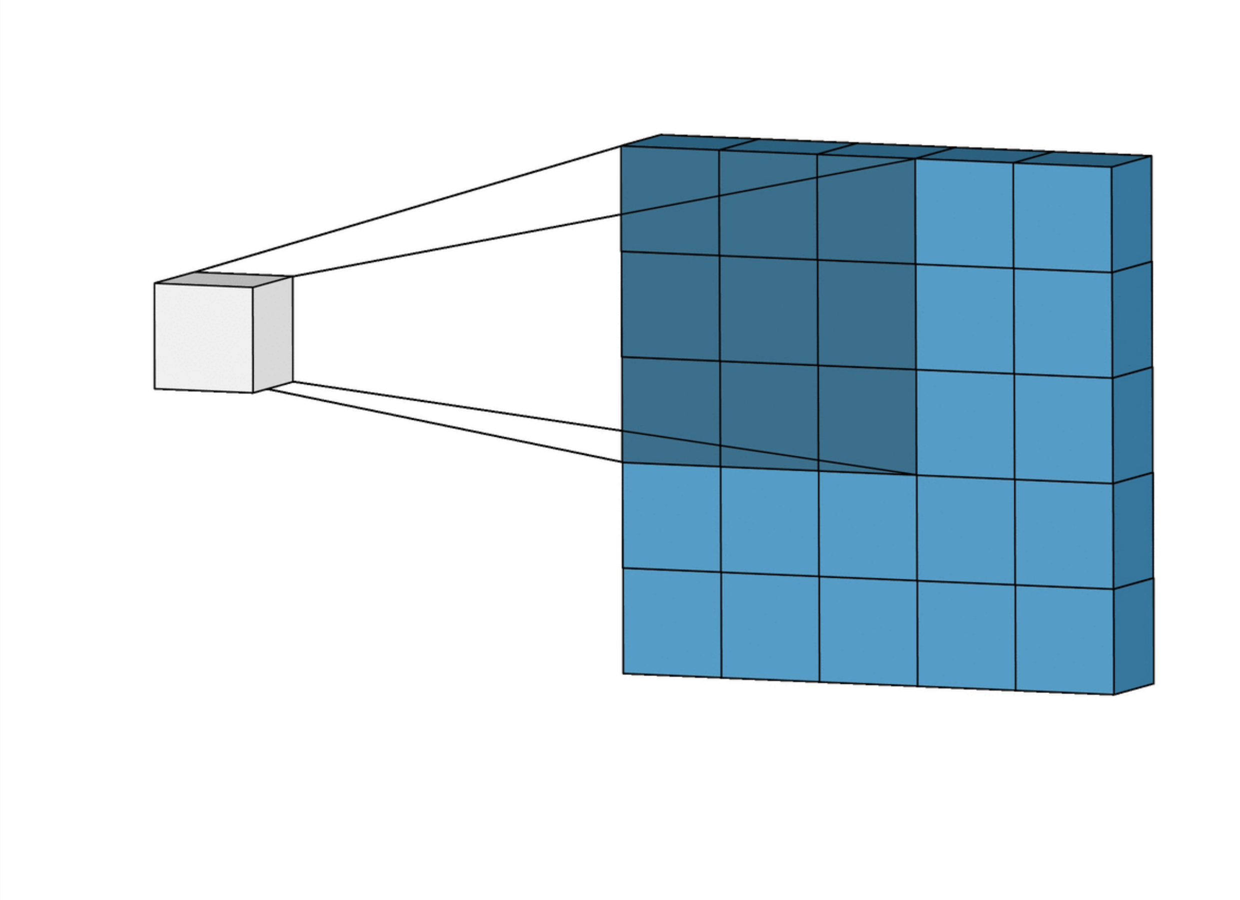
- Discrete convolution:

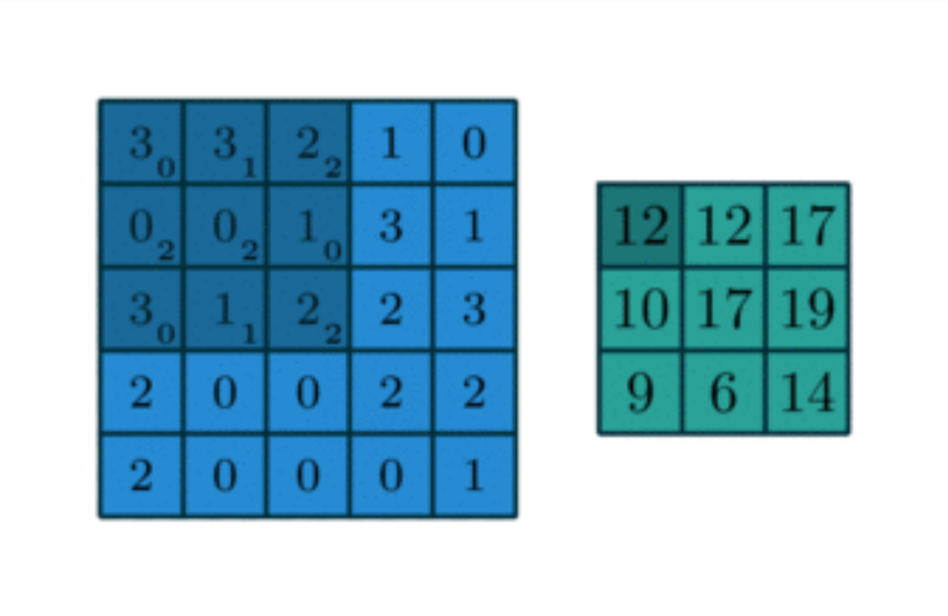
$$(f * g)(t) = \sum_{a=-\infty}^{\infty} f(a)g(t - a)$$

- Discrete cross-correlation:

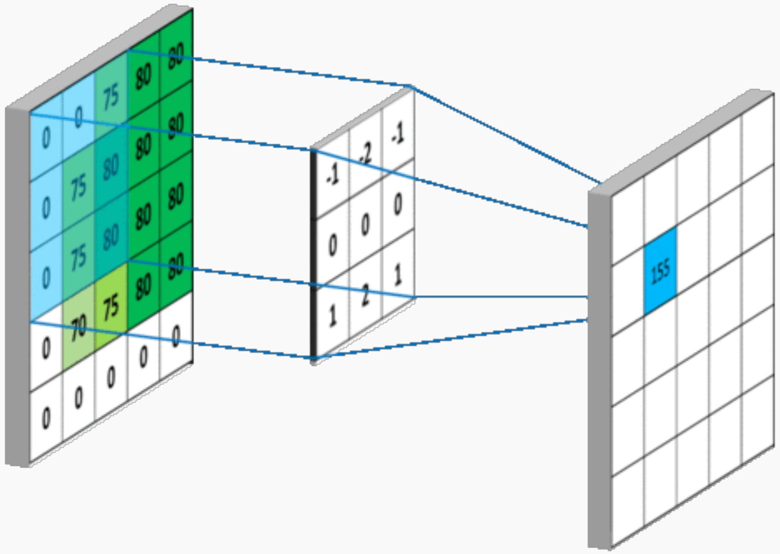
$$(f \star g)(t) = \sum_{a=-\infty}^{\infty} f(a)g(t + a)$$

# Convolutions – step by step

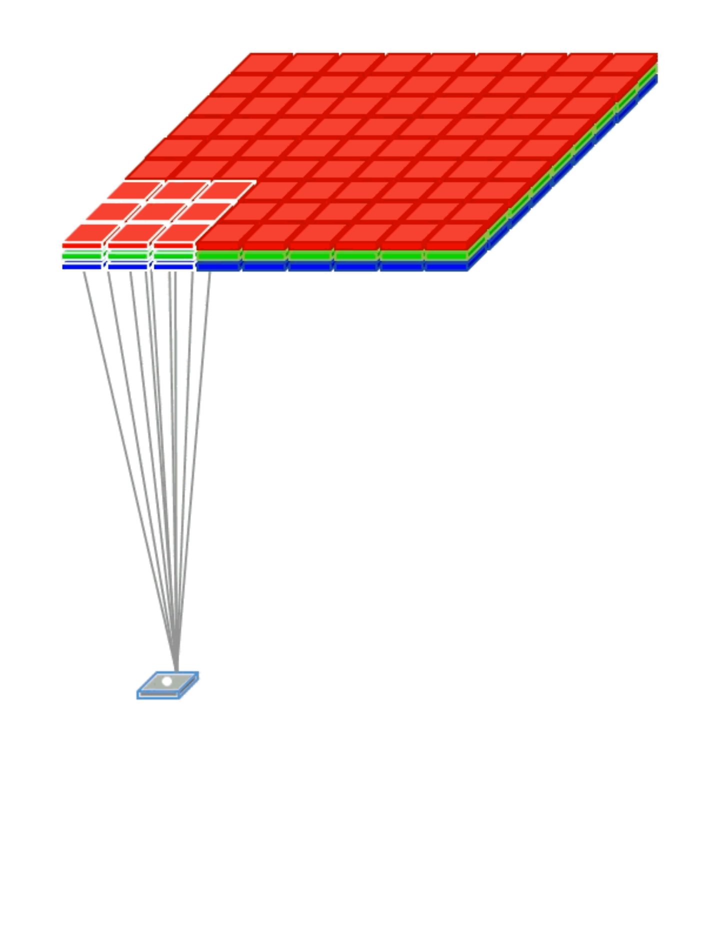




# Convolutions – another example

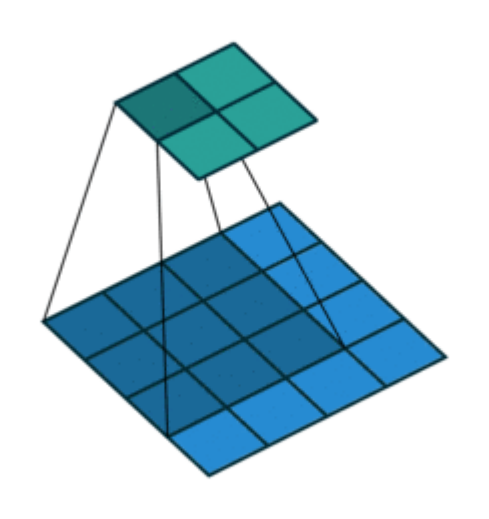


# Convolutions – 3D input



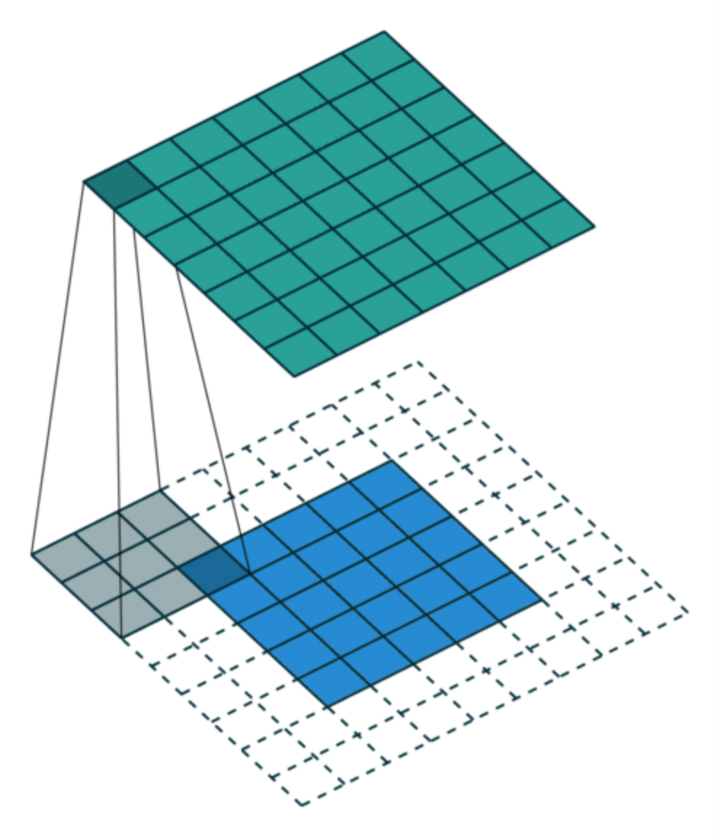
# Convolutions – what happens at the edges?

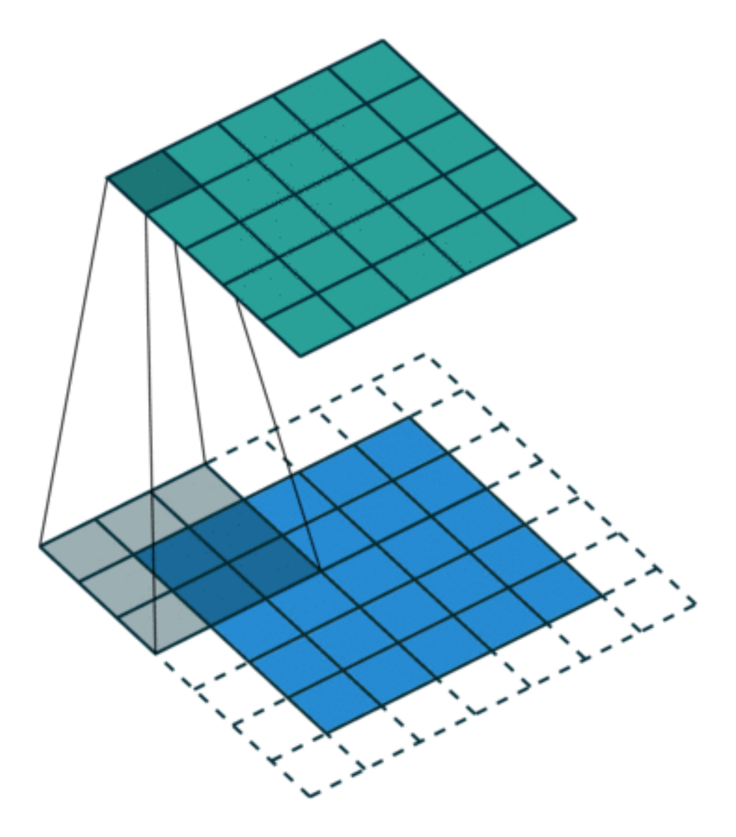
If we apply convolutions on a normal image, the result will be down-sampled by an amount depending on the size of the filter.



We can avoid this by padding the edges in different ways.

# Padding

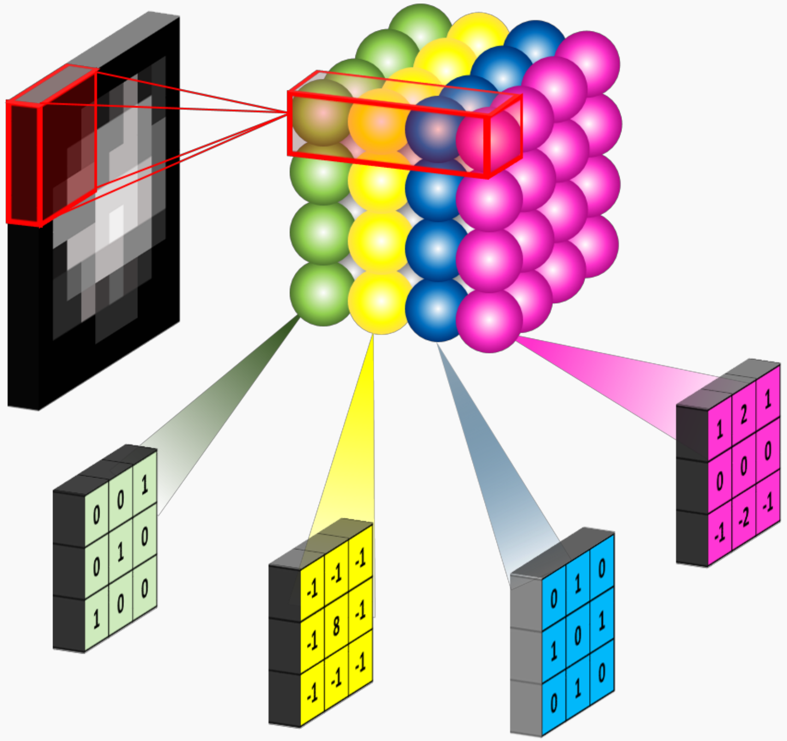




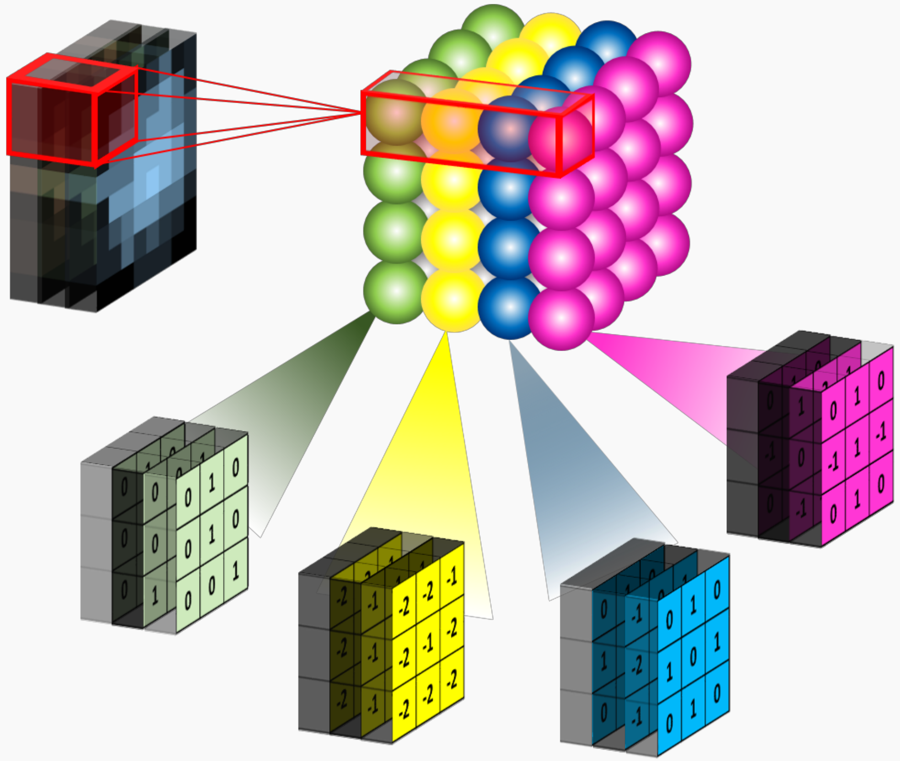
Full padding. Introduces zeros such that all pixels are visited the same amount of times by the filter. Increases size of output.

Same padding. Ensures that the output has the same size as the input.

# Convolutional layers



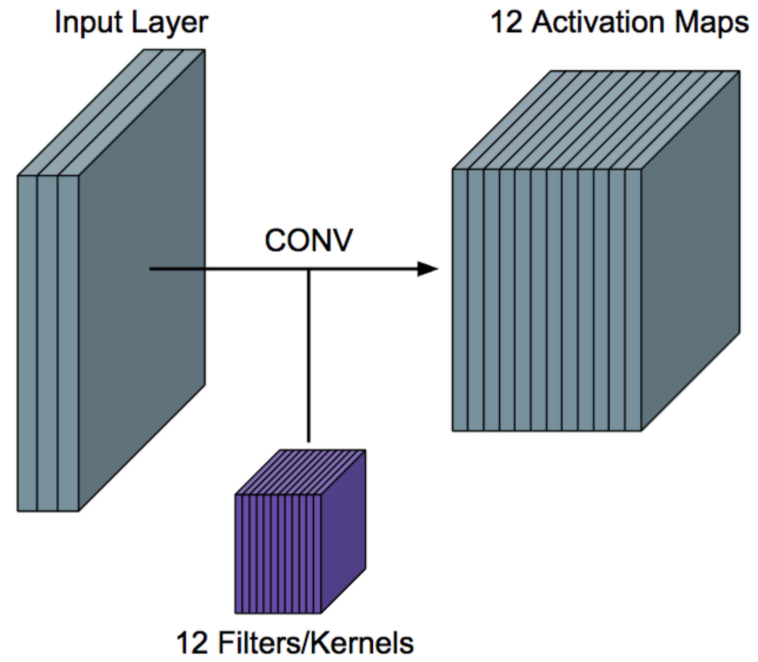
Convolutional layer with four 3x3 filters on a black and white image (just one channel)



Convolutional layer with four 3x3 filters on an RGB image. As you can see, the filters are now cubes, and they are applied on the full depth of the image..

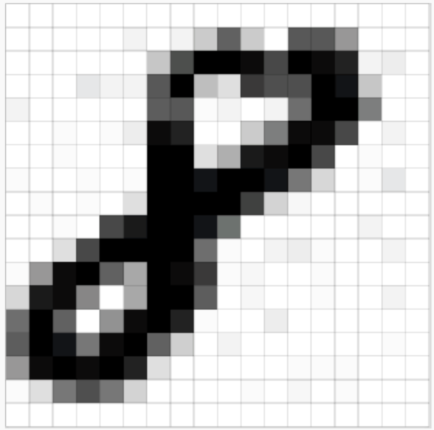
# Convolutional layers (cont)

- To be clear: each filter is convolved with the entirety of the 3D input cube, but generates a 2D feature map.
- Because we have multiple filters, we end up with a 3D output: one 2D feature map per filter.
- The feature map dimension can change drastically from one conv layer to the next: we can enter a layer with a 32x32x16 input and exit with a 32x32x128 output if that layer has 128 filters.



# Why does this make sense?

In image is just a matrix of pixels.



Convolving the image with a filter produces a feature map that highlights the presence of a given feature in the image.

| Operation | Filter | Convolved Image |
| --- | --- | --- |
| Identity | $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ |  |
| 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 | $\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$ |  |


Input

# Learning CNN

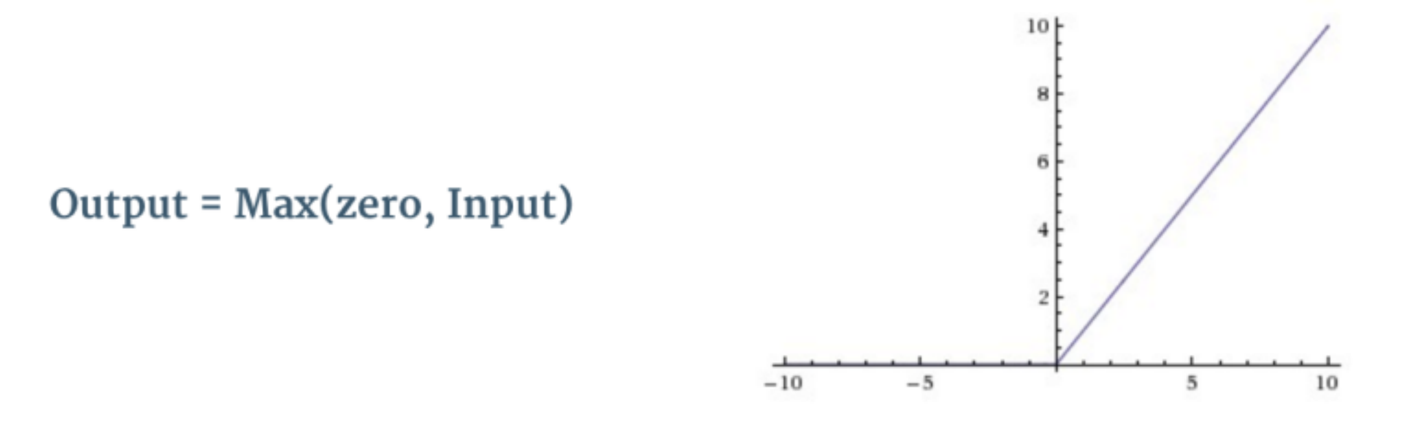
In a convolutional layer, we are basically applying multiple filters at over the image to extract different features.

But most importantly, we are learning those filters!

One thing we're missing: non-linearity.

# Introducing ReLU

The most successful non-linearity for CNNs is the Rectified Non-Linear unit (ReLU):




Combats the vanishing gradient problem occurring in sigmoids, is easier to compute, generates sparsity (not always beneficial)

# Convolutional layers so far

- A convolutional layer convolves each of its filters with the input.
- Input: a 3D tensor, where the dimensions are Width, Height and Channels (or Feature Maps)
- Output: a 3D tensor, with dimensions Width, Height and Feature Maps (one for each filter)
- Applies non-linear activation function (usually ReLU) over each value of the output.
- Multiple parameters to define: number of filters, size of filters, stride, padding, activation function to use, regularization.

# Building a CNN

A convolutional neural network is built by stacking layers, typically of 3 types:

Convolutional Layers

Pooling Layers

Fully connected Layers

# Building a CNN

A c
ty

## Convolutional Layers

### Action

- Apply filters to extract features
- Filters are composed of small kernels, learned.
- One bias per filter.
- Apply activation function on every value of feature map

### Parameters

- Number of kernels
- Size of kernels (W and H only, D is defined by input cube)
- Activation function
- Stride
- Padding
- Regularization type and value

### I/O

- Input: 3D cube, previous set of feature maps
- Output: 3D cube, one 2D map per filter

# Building a CNN

A convolutional neural network is built by stacking layers, typically of 3 types:

- Convolutional Layers
- Pooling Layers
- Fully connected Layers

# Building a CNN

A c
ty

## Pooling Layers

**Action**

- Reduce dimensionality
- Extract maximum of average of a region
- Sliding window approach

**Parameters**

- Stride
- Size of window

**I/O**

- Input: 3D cube, previous set of feature maps
- Output: 3D cube, one 2D map per filte, reduced spatial dimensions

# Building a CNN

A convolutional neural network is built by stacking layers, typically of 3 types:

- Convolutional Layers
- Pooling Layers
- Fully connected Layers

# Building a CNN

A c
ty

## Fully connected Layers

### Action

- Aggregate information from final feature maps
- Generate final classification

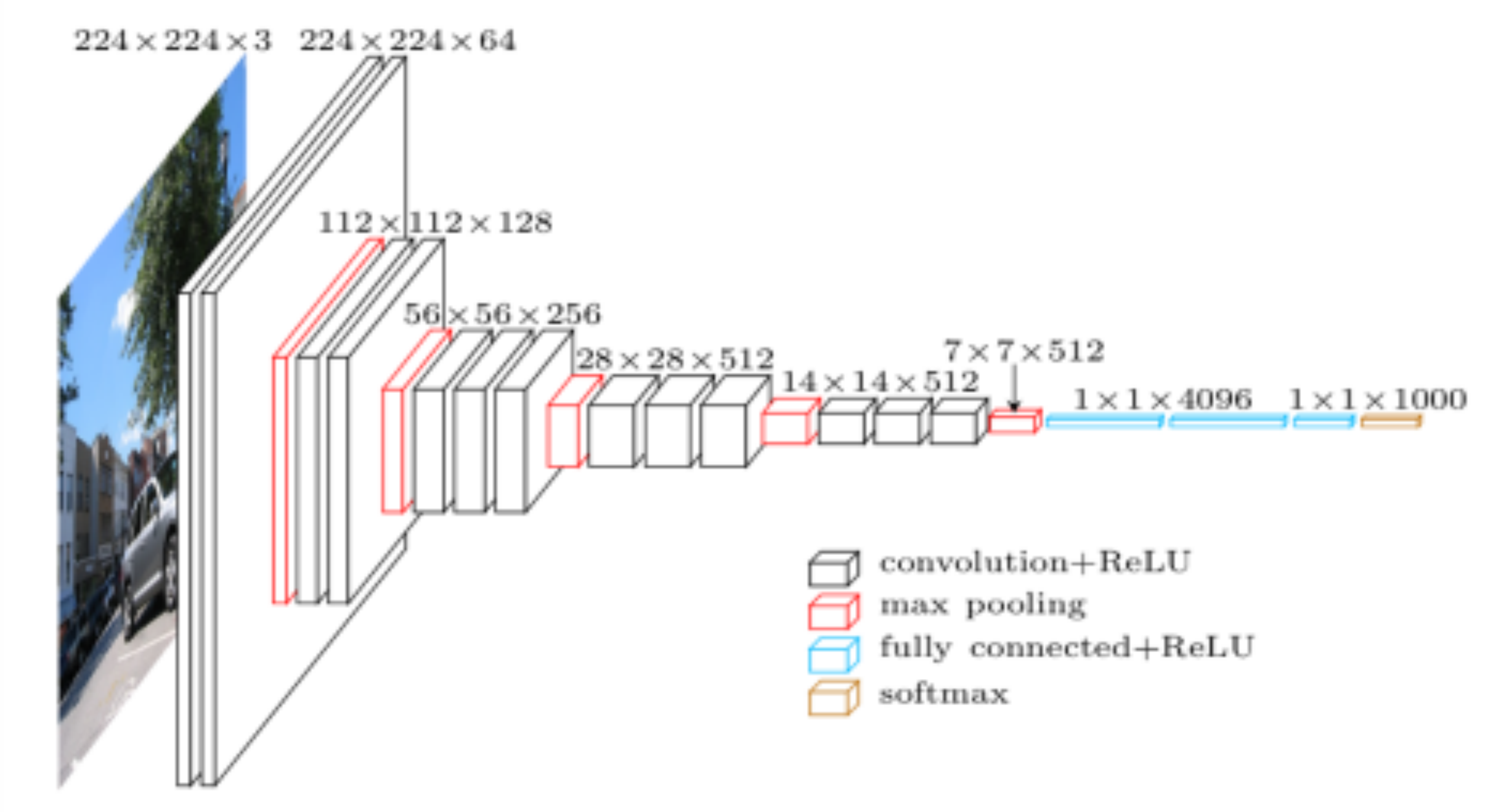
### Parameters

- Number of nodes
- Activation function: usually changes depending on role of layer. If aggregating info, use ReLU. If producing final classification, use Softmax.

### I/O

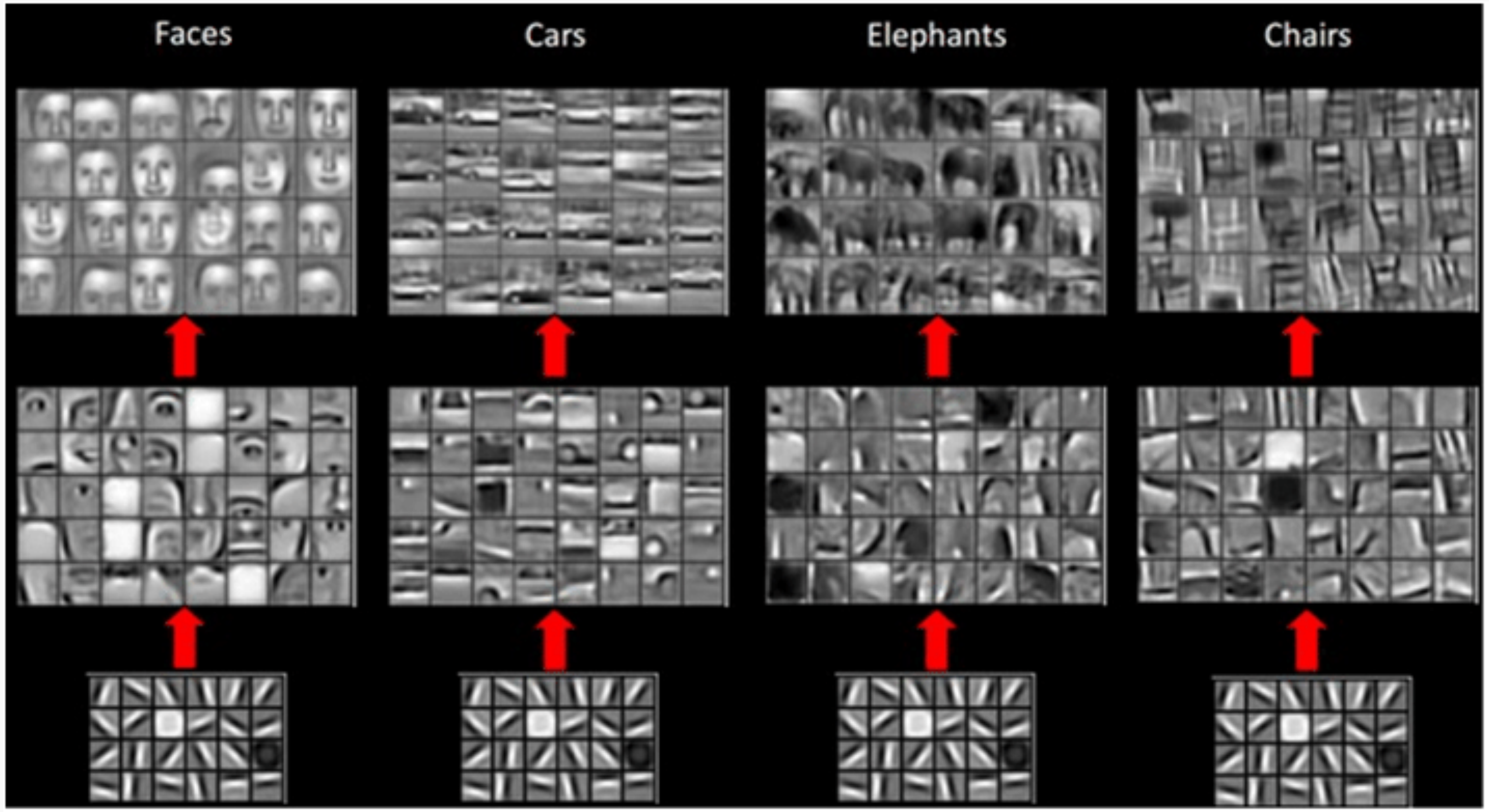
- Input: FLATTENED 3D cube, previous set of feature maps
- Output: 3D cube, one 2D map per filter

# Fully built CNN (VGG)



# What do CNN layers learn?

- Each CNN layer learns filters of increasing complexity.
- The first layers learn basic feature detection filters: edges, corners, etc.
- The middle layers learn filters that detect parts of objects. For faces, they might learn to respond to eyes, noses, etc.
- The last layers have higher representations: they learn to recognize full objects, in different shapes and positions.


Faces
Cars
Elephants
Chairs

# Examples

- I have a convolutional layer with 16 3x3 filters that takes an RGB image as input.
  - What else can we define about this layer?
    - Activation function
    - Stride
    - Padding type
  - How many parameters does the layer have?

$$16 \times 3 \times 3 \times 3 + 16 = 448$$

Number of filters

Size of Filters

Number of channels of prev layer

Biases (one per filter)

# Examples

- Let C be a CNN with the following disposition:
  - Input: 32x32x3 images
  - Conv1: 8 3x3 filters, stride 1, padding=same
  - Conv2: 16 5x5 filters, stride 2, padding=same
  - Flatten layer
  - Dense1: 512 nodes
  - Dense2: 4 nodes
- How many parameters does this network have?

(8 x 3 x 3 x 3 + 8) + (16 x 5 x 5 x 8 + 16) + (16 x 16 x 16 x 512 + 512) + (512 x 4 + 4)

Conv1 Conv2 Dense1 Dense2

## 3D visualization of networks in action

http://scs.ryerson.ca/~aharley/vis/conv/

https://www.youtube.com/watch?v=3JQ3hYko51Y

# EVOLUTION OF CNNs

A bit of history

# Initial ideas

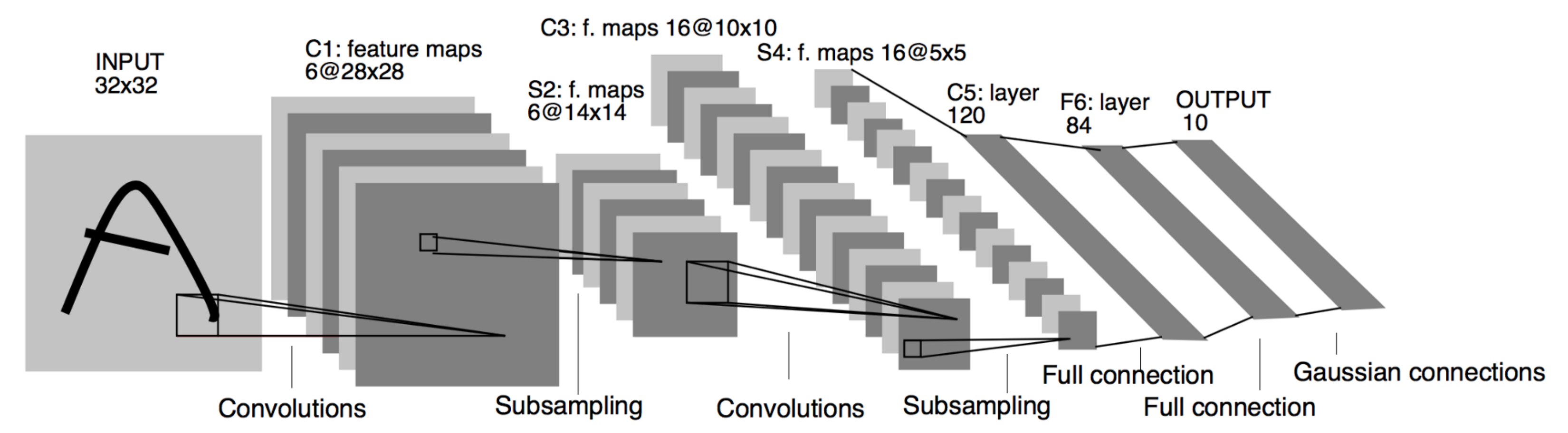
- The first piece of research proposing something similar to a Convolutional Neural Network was authored by Kunihiko Fukushima in 1980, and was called the NeoCognitron[1].
- Inspired by discoveries on visual cortex of mammals.
- Fukushima applied the NeoCognitron to hand-written character recognition.
- End of the 80's: several papers advanced the field
  - Backpropagation published in French by Yann LeCun in 1985 (independently discovered by other researchers as well)
  - TDNN by Waiber et al., 1989 - Convolutional-like network trained with backprop.
  - Backpropagation applied to handwritten zip code recognition by LeCun et al., 1989

---

[1] K. Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biological Cybernetics, 36(4): 93-202, 1980.

# LeNet

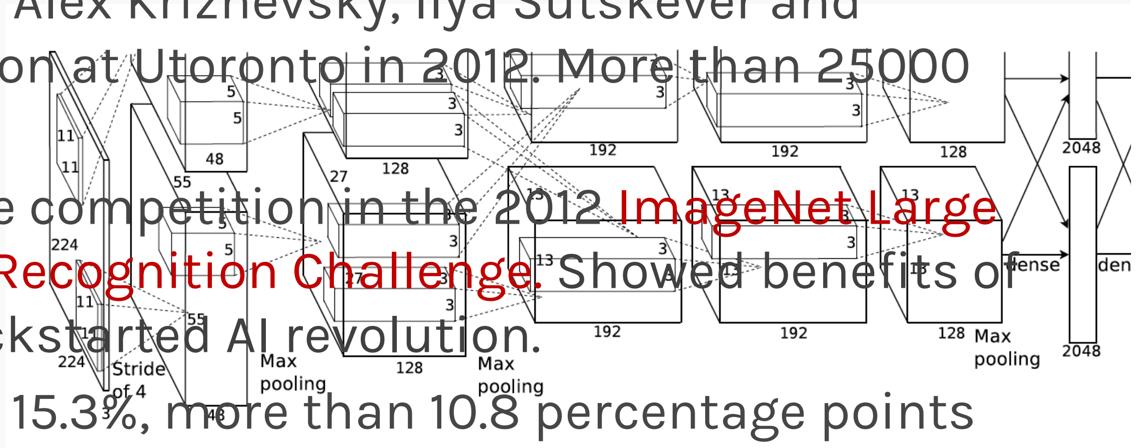
- November 1998: LeCun publishes one of his most recognized papers describing a “modern” CNN architecture for document recognition, called LeNet[1].
- Not his first iteration, this was in fact LeNet-5, but this paper is the commonly cited publication when talking about LeNet.



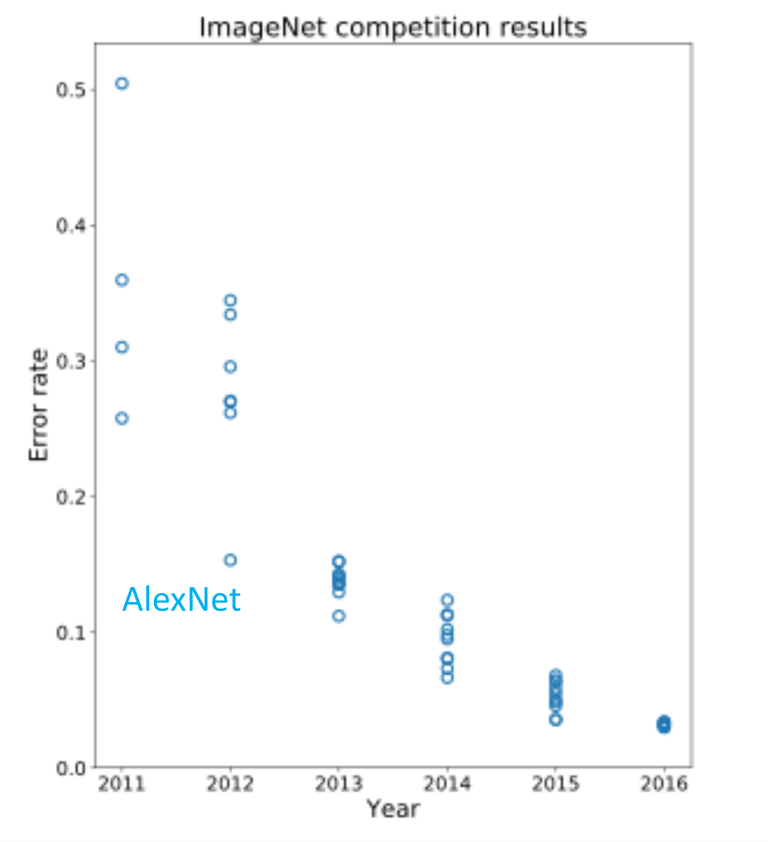

[1] LeCun, Yann, et al. "Gradient-based learning applied to document recognition." *Proceedings of the IEEE* 86.11 (1998): 2278-2324.

# AlexNet

- Developed by Alex Krizhevsky, Ilya Sutskever and Geoffrey Hinton at Utoronto in 2012. More than 25000 citations.
- Destroyed the competition in the 2012 ImageNet Large Scale Visual Recognition Challenge. Showed benefits of CNNs and kickstarted AI revolution.
- top-5 error of 15.3%, more than 10.8 percentage points lower than runner-up.

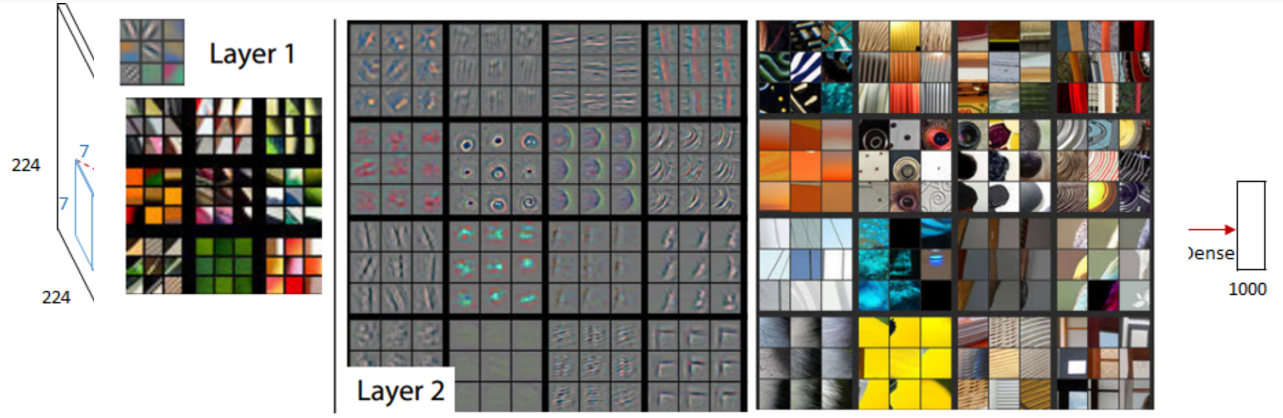



- Main contributions:
  - Trained on ImageNet with data augmentation
  - Increased depth of model, GPU training (*five to six days*)
  - Smart optimizer and Dropout layers
  - ReLU activation!



# ZFNet

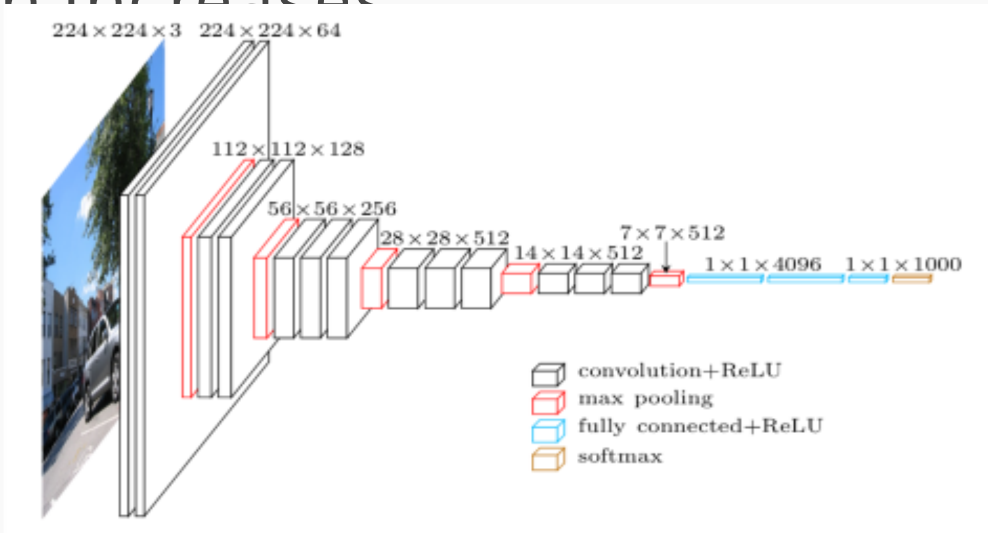
- Introduced by Matthew Zeiler and Rob Fergus from NYU, won ILSVRC 2013 with 11.2% error rate. Decreased sizes of filters.
- Trained for 12 days.
- Paper presented a visualization technique named Deconvolutional Network, which helps to examine different feature ac ut space.




Visualizations of Layer 1 and 2. Each layer illustrates 2 pictures, one which shows the filters themselves and one that shows what part of the image are most strongly activated by the given filter. For example, in the space labled Layer 2, we have representations of the 16 different filters (on the left)

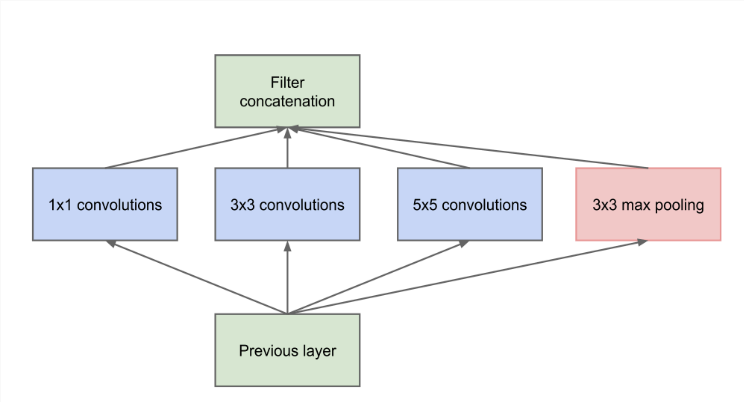
# VGG

- Introduced by Simonyan and Zisserman (Oxford) in 2014
- Simplicity and depth as main points. Used 3x3 filters exclusively and 2x2 MaxPool layers with stride 2.
- Showed that two 3x3 filters have an effective receptive field of 5x5.
- As spatial size decreases, depth increases.
- Trained for *two to three weeks.*
- Still used as of today.



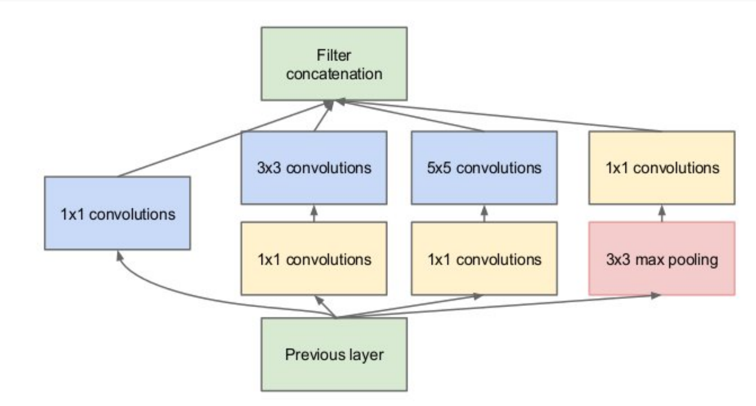
# GoogLeNet (Inception-v1)

- Introduced by Szegedy et al. (Google), 2014. Winners of ILSVRC 2014.
- Introduces inception module: parallel conv. layers with different filter sizes. Motivation: we don't know which filter size is best – let the network decide. Key idea for future archs.
- No fully connected layer at the end. AvgPool instead. 12x fewer params than AlexNet.

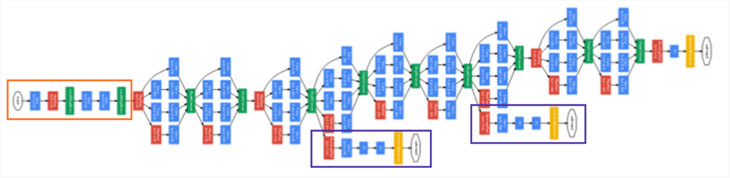



Proto Inception module

1x1 convs to Reduce number of parameters

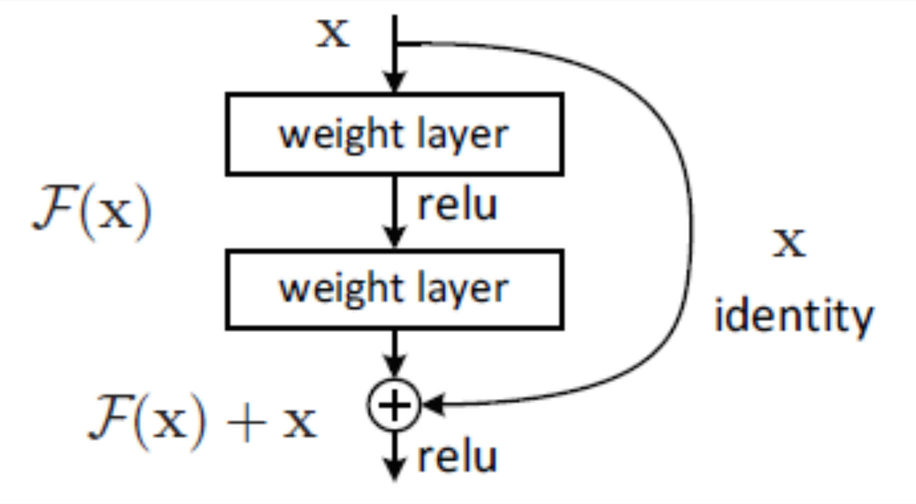



Inception module



# ResNet

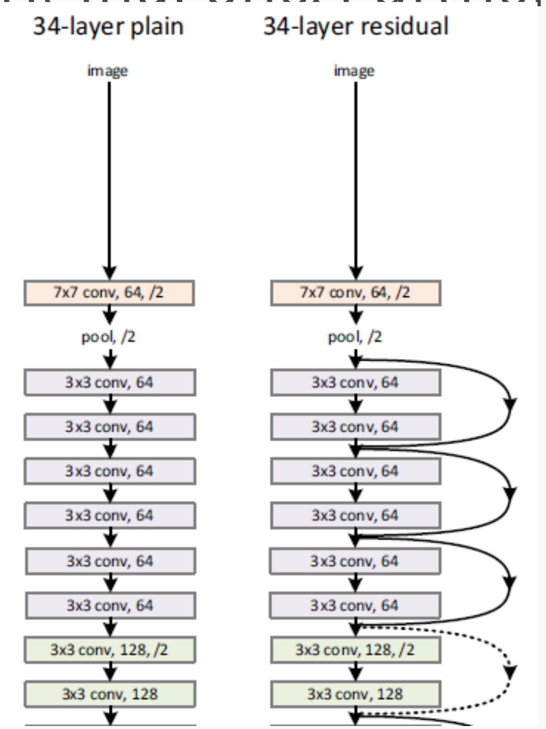
- Presented by He et al. (Microsoft), 2015. Won ILSVRC 2015 in multiple categories.
- Main idea: Residual block. Allows for extremely deep networks.
- Authors believe that it is easier to optimize the residual mapping than the original one. Furthermore, residual block can decide to “shut itself down” if needed.

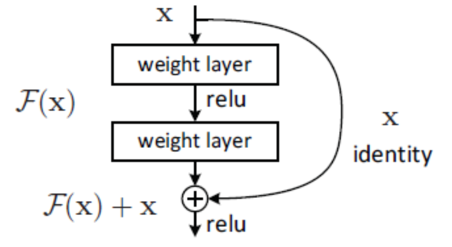



Residual Block

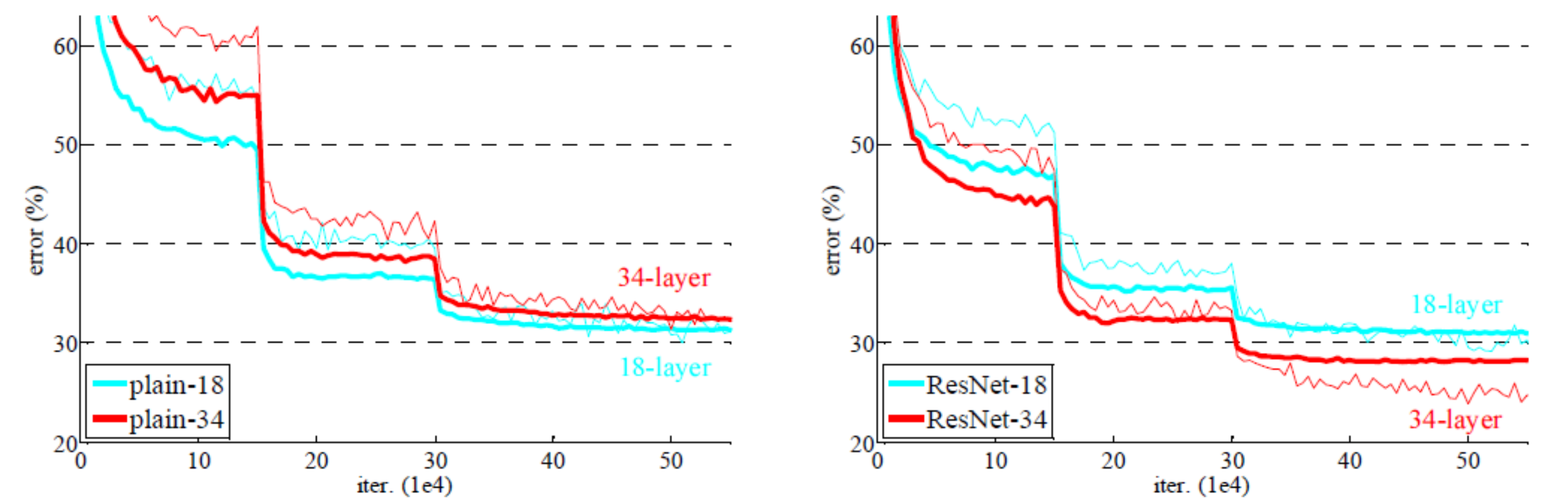
# ResNet

- Presented by He et al. (Microsoft), 2015. Won ILSVRC 2015 in multiple categories.
- Main idea: Residual block. Allows for extremely deep networks.
- Authors believe that it is easier to optimize the residual mapping than the original one. Furthermore, residual … de to “shut itself down” if n…




Residual Block



# DenseNet

- Proposed by Huang et al., 2016. Radical extension of ResNet idea.
- Each block uses every previous feature map as input.
- Idea: n computation of redundant features. All the previous information is available at each point.
- Counter-intuitively, it reduces the number of parameters needed.

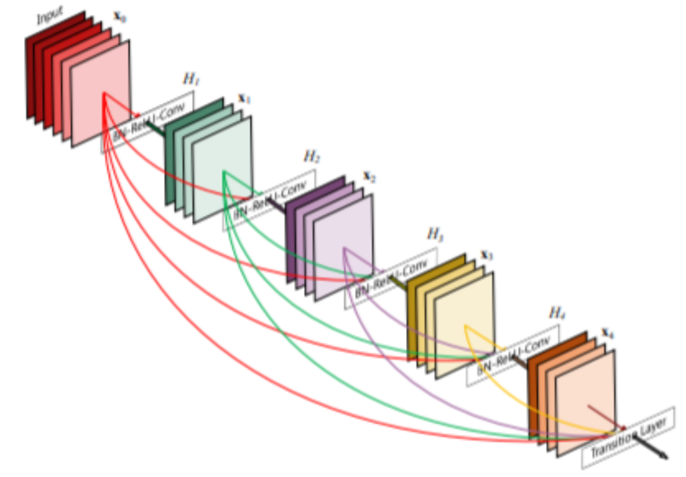


**Figure 1:** A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.

# DenseNet

- Proposed by Huang et al., 2016. Radical extension of ResNet idea.
- Each block uses every previous feature map as input.
- Idea: n computation of redundant features. All the previous information is available at each point.
- Counter-intuitively, it reduces the number of parameters needed.

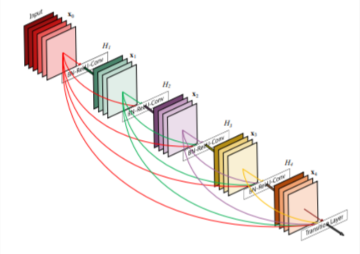
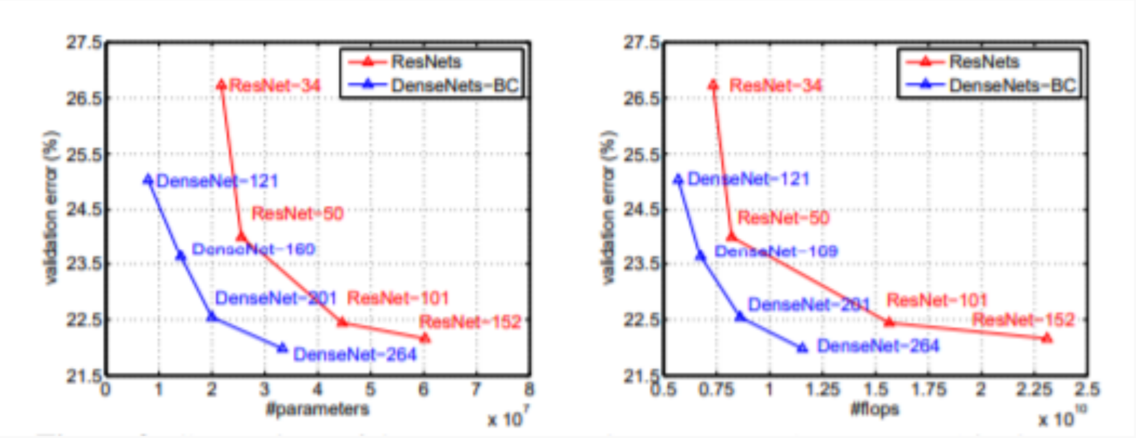
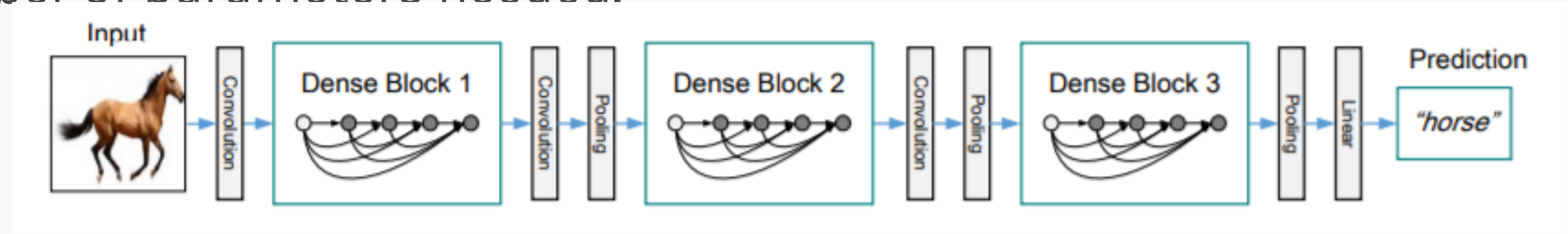


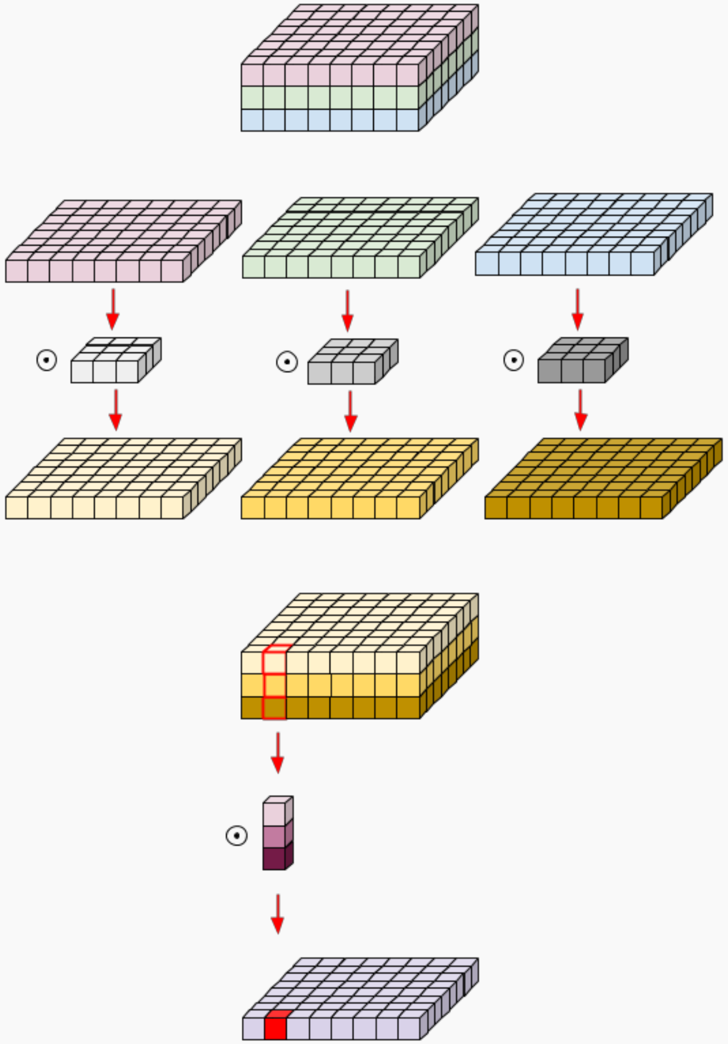
**Figure 1:** A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.



# MobileNet

- Published by Howard et al., 2017.
- Extremely efficient network with decent accuracy.
- Main concept: depthwise-separable convolutions. Convolve each feature maps with a kernel, then use a 1x1 convolution to aggregate the result.
- This approximates vanilla convolutions without having to convolve large kernels through channels.



# Beyond

- MobileNetV2 (https://arxiv.org/abs/1801.04381)
- Inception-Resnet, v1 and v2 (https://arxiv.org/abs/1602.07261)
- Wide-Resnet (https://arxiv.org/abs/1605.07146)
- Xception (https://arxiv.org/abs/1610.02357)
- ResNeXt (https://arxiv.org/pdf/1611.05431)
- ShuffleNet, v1 and v2 (https://arxiv.org/abs/1707.01083)
- Squeeze and Excitation Nets (https://arxiv.org/abs/1709.01507 )