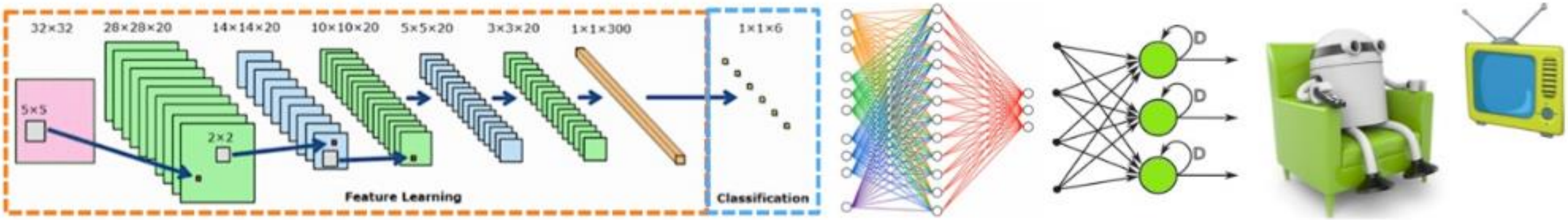


Introduction to Deep Learning

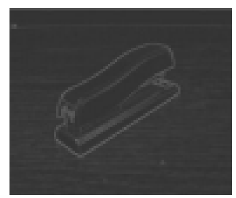
Traditional machine perception

State-of-the-art:
“hand-crafting”

Input data → Feature representation → Learning algorithm

Object detection

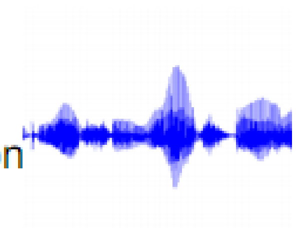
Image



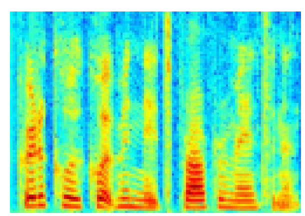
Low-level
vision features
(SIFT, HOG, etc.)

Object detection
/ classification

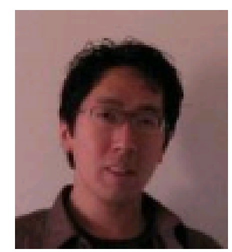
Audio classification



Audio

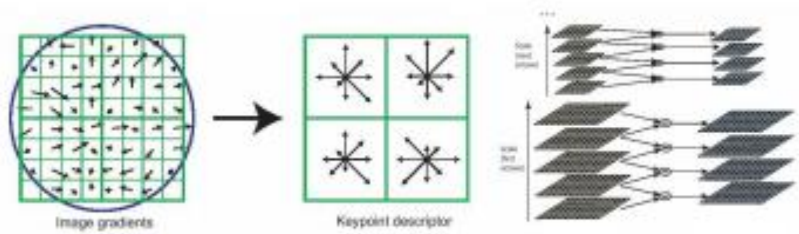


Low-level
audio features
(spectrogram, MFCC, etc.)

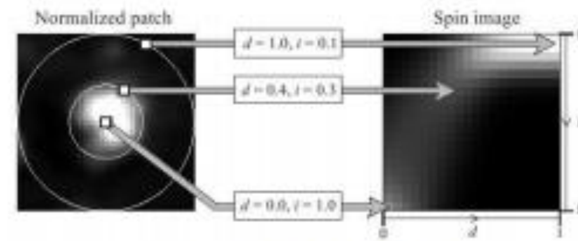


Speaker
identification

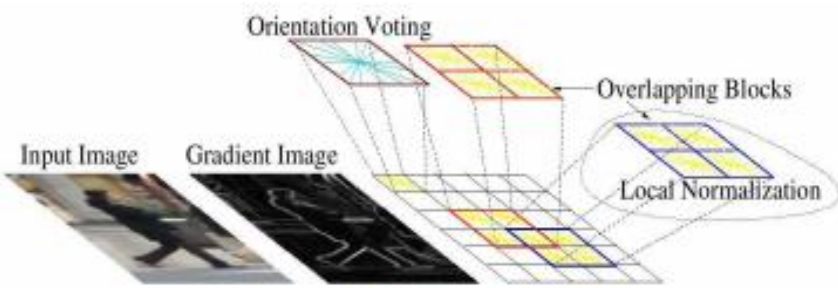
Computer vision features



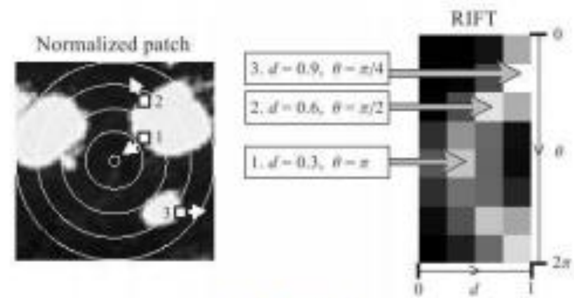
SIFT



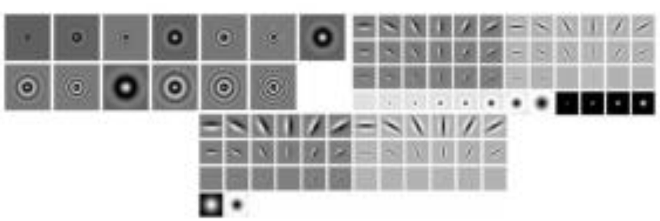
Spin image



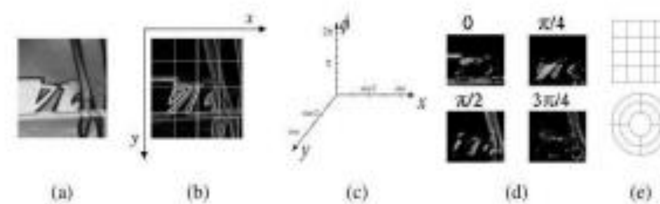
HoG



RIFT



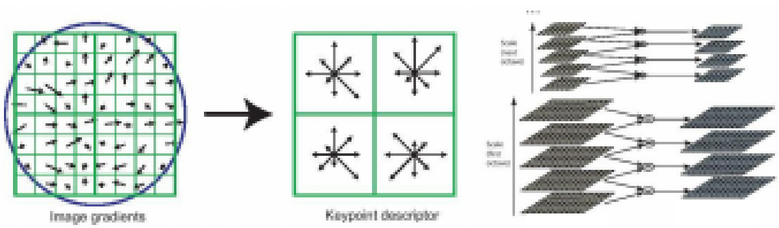
Textons



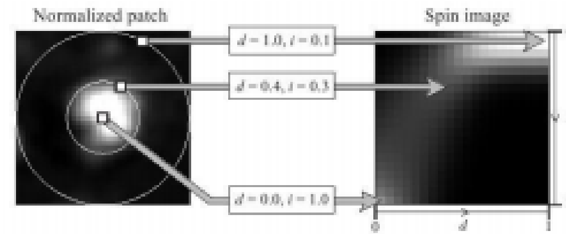
GLOH

Computer Vision

Computer vision features



SIFT



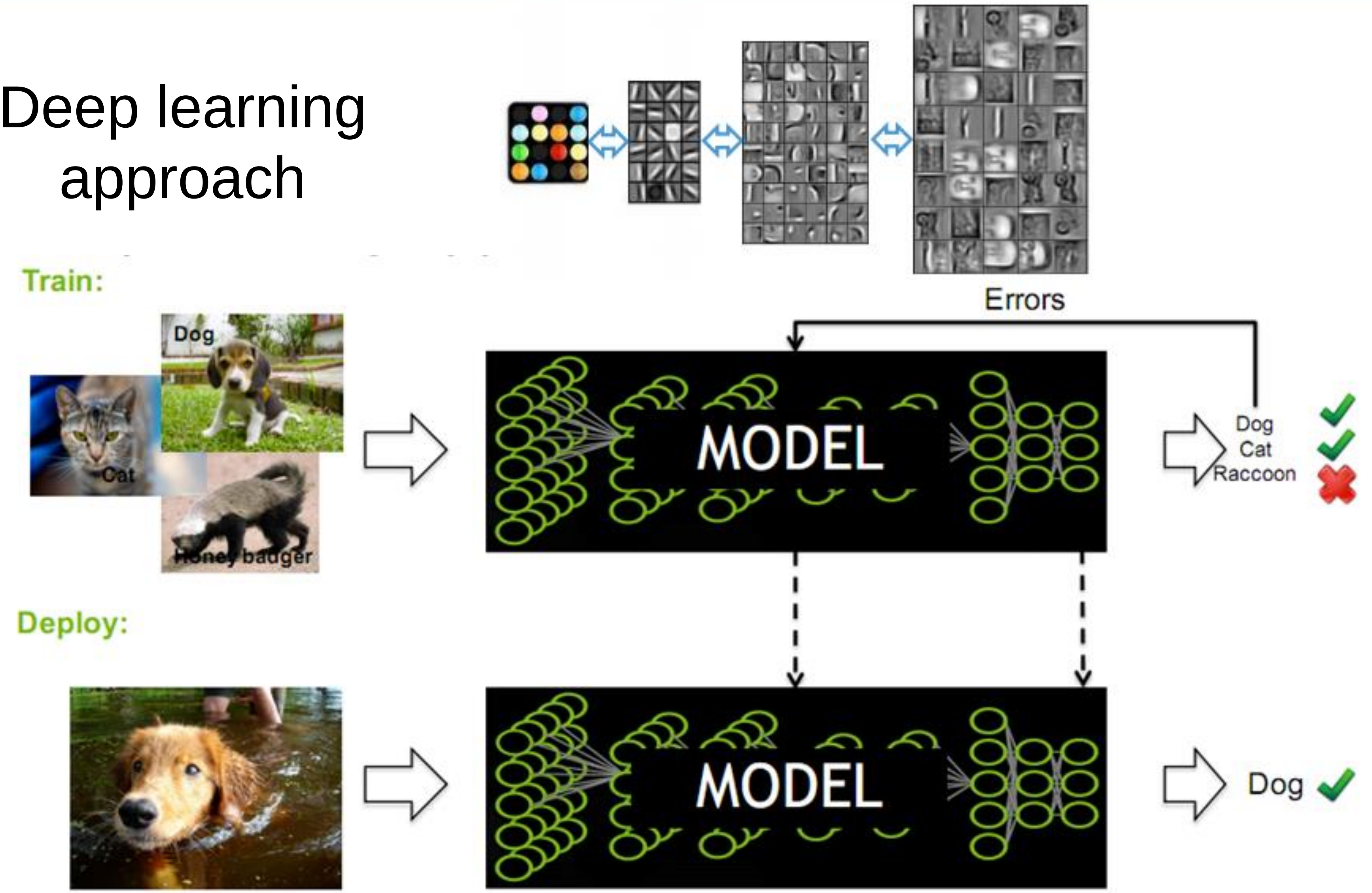
Spin image

Hand-crafted features:

1. Needs expert knowledge
2. Requires time-consuming hand-tuning
3. (Arguably) one of the limiting factors of computer vision systems

Deep Learning

Deep learning approach



Deep Learning

Deep Learning

MIT Technology Review

10 BREAKTHROUGH TECHNOLOGIES 2013

Introduction | The 10 Technologies | Past Years

Deep Learning

With massive amounts of computational power, machines can now recognize objects and translate speech in real time. Artificial intelligence is finally getting smart.

Temporary Social Media

Messages that quickly self-destruct could enhance the privacy of online communications and make people freer to be spontaneous.

Prenatal DNA Sequencing

Reading the DNA of fetuses will be the next frontier of the genomic revolution. But do you really want to know about the genetic problems or musical aptitude of your unborn child?

Additive Manufacturing

Skeptical about 3-D printing? GE, the world's largest manufacturer, is on the verge of using the technology to make jet parts.

Baxter: The Blue-Collar Robot

Rodney Brooks's newest creation is easy to interact with, but the complex innovations behind the robot show just how hard it is to get along with people.

Memory Implants

A maverick neuroscientist believes he has deciphered the code by which the brain forms long-term memories. Next: testing a prosthetic implant for people suffering from long-term memory loss.

Smart Watches

The designers of the Pebble watch realized that a mobile phone is more useful if you don't have to take it out of your pocket.

Ultra-Efficient Solar Power

Doubling the efficiency of a solar cell would completely change the economics of renewable energy. Nanotechnology just might make it possible.

Big Data from Cheap Phones

Collecting and analyzing information from simple cell phones can provide surprising insights into how people move about and behave – and even help us understand the spread of diseases.

Supergrids

A new high-power circuit breaker could finally make highly efficient DC power grids practical.

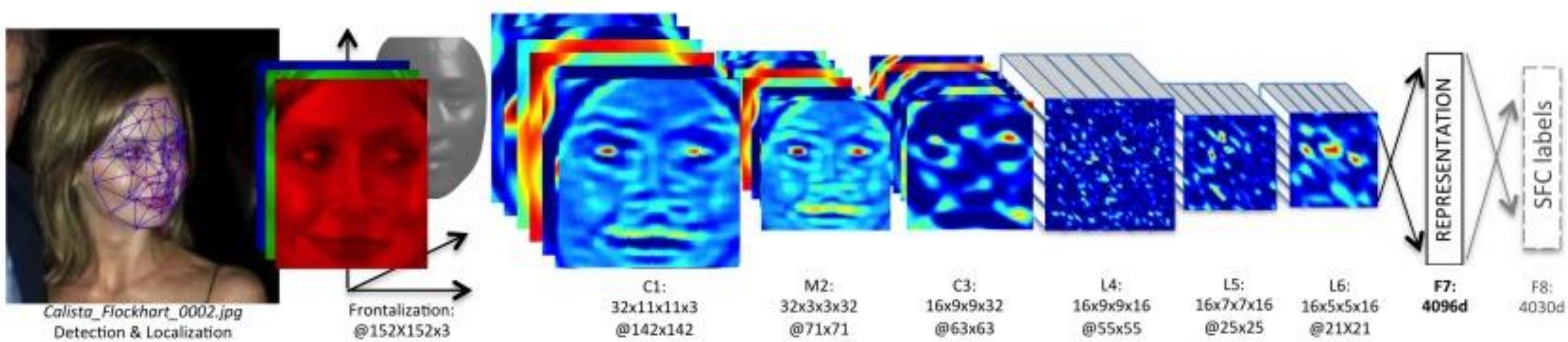
Deep Learning vs Human

DeepFace: Closing the Gap to Human-Level Performance in Face Verification

Yaniv Taigman Ming Yang Marc'Aurelio Ranzato Lior Wolf

Facebook AI Research
Menlo Park, CA, USA
{yaniv, mingyang, ranzato}@fb.com

Tel Aviv University
Tel Aviv, Israel
wolf@cs.tau.ac.il



DeepFace 2014

Closing the Gap to **Human** Level **Performance** in **Face** Verification

Accuracy

DeepFace: 97.35%
Human: 97.5%

Deep Learning vs Human

News & Analysis

Microsoft, Google Beat Humans at Image Recognition

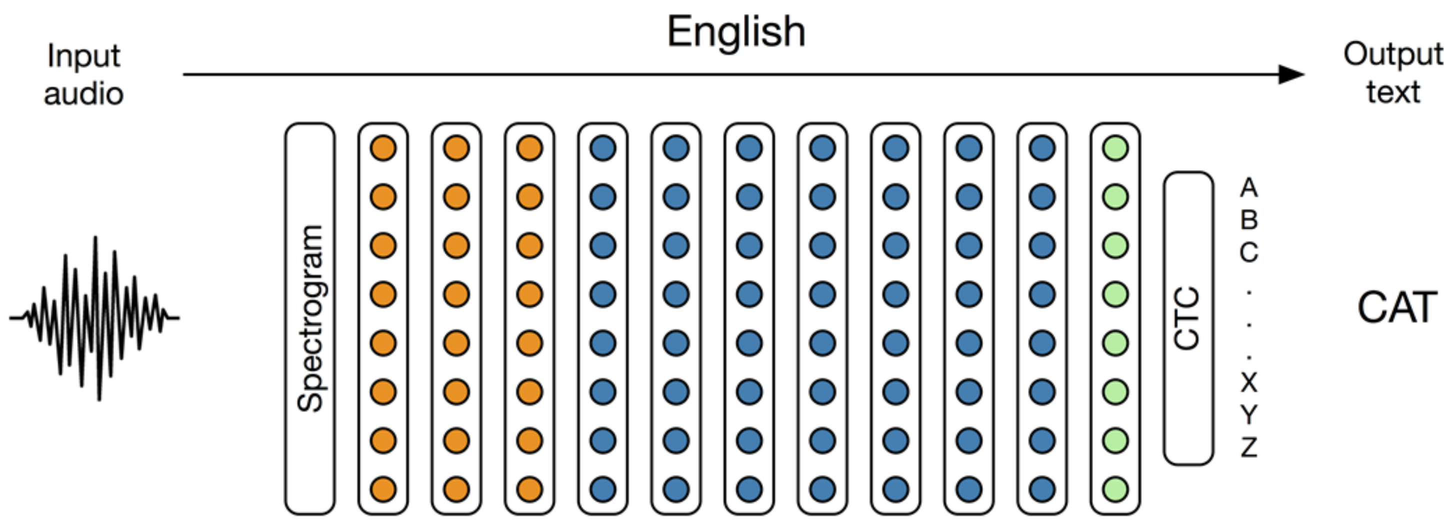
Deep learning algorithms compete at ImageNet challenge

R. Colin Johnson
2/18/2015 08:15 AM EST
14 comments

NO RATINGS
1 saves
LOGIN TO RATE

The competition is fierce, with the ImageNet Large Scale Visual Recognition Challenge doing the judging for the 2015 championship on December 17. Between now and then expect to see a stream of papers claiming they have one-upped humans too. For instance, only 5 days after **Microsoft announced it had beat the human benchmark of 5.1% errors with a 4.94%** error grabbing neural network, **Google announced it had one-upped Microsoft by 0.04%**.

Deep Learning vs Human

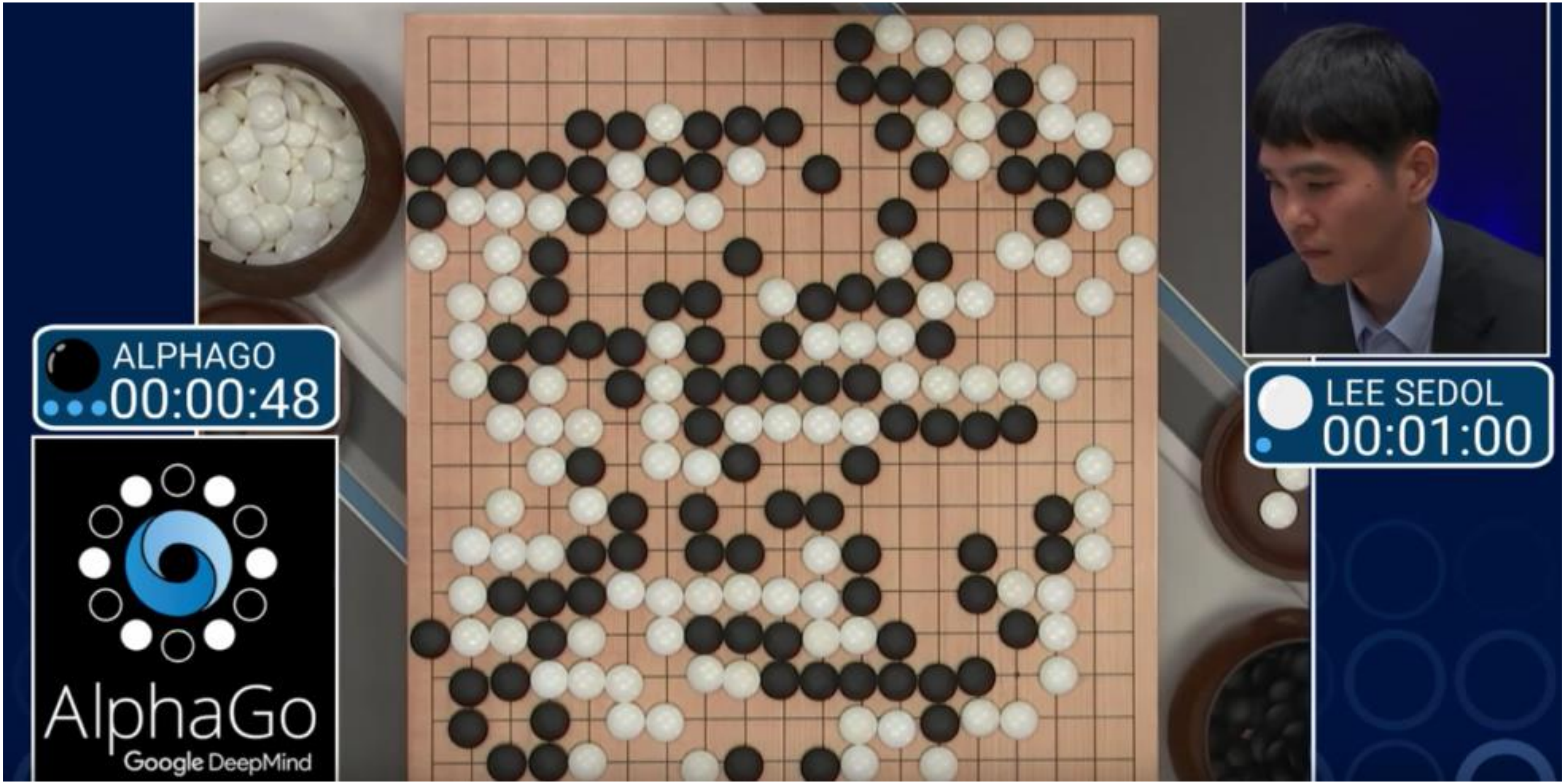


Deep Speech 2015

Baidu has developed a voice system that can recognize English and Mandarin speech **better than people**, in some cases.

"For short phrases, out of context, we seem to be **surpassing human levels of recognition**"- Andrew Ng

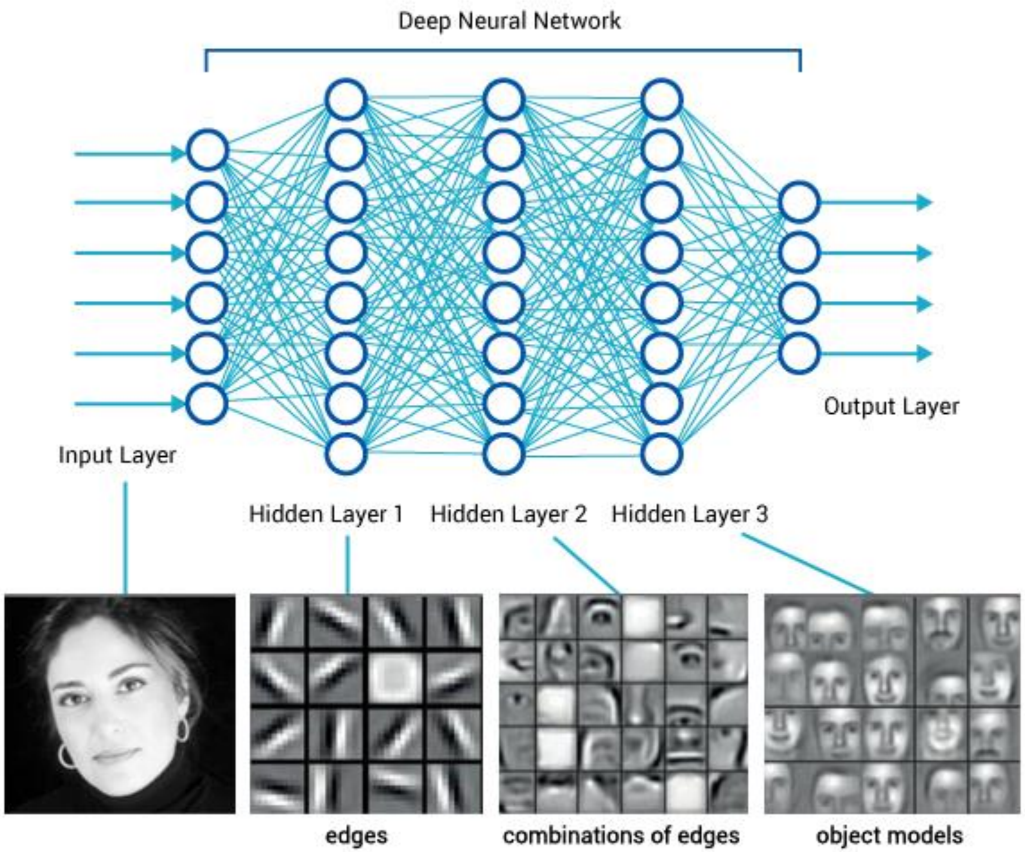
Deep Learning vs Human



AlphaGo vs Human: 9-1

What is Deep Learning?

Deep learning is a branch of machine learning based on a set of algorithms that attempt to model high-level abstractions in data by using a deep graph with multiple processing layers, composed of multiple non-linear transformations.



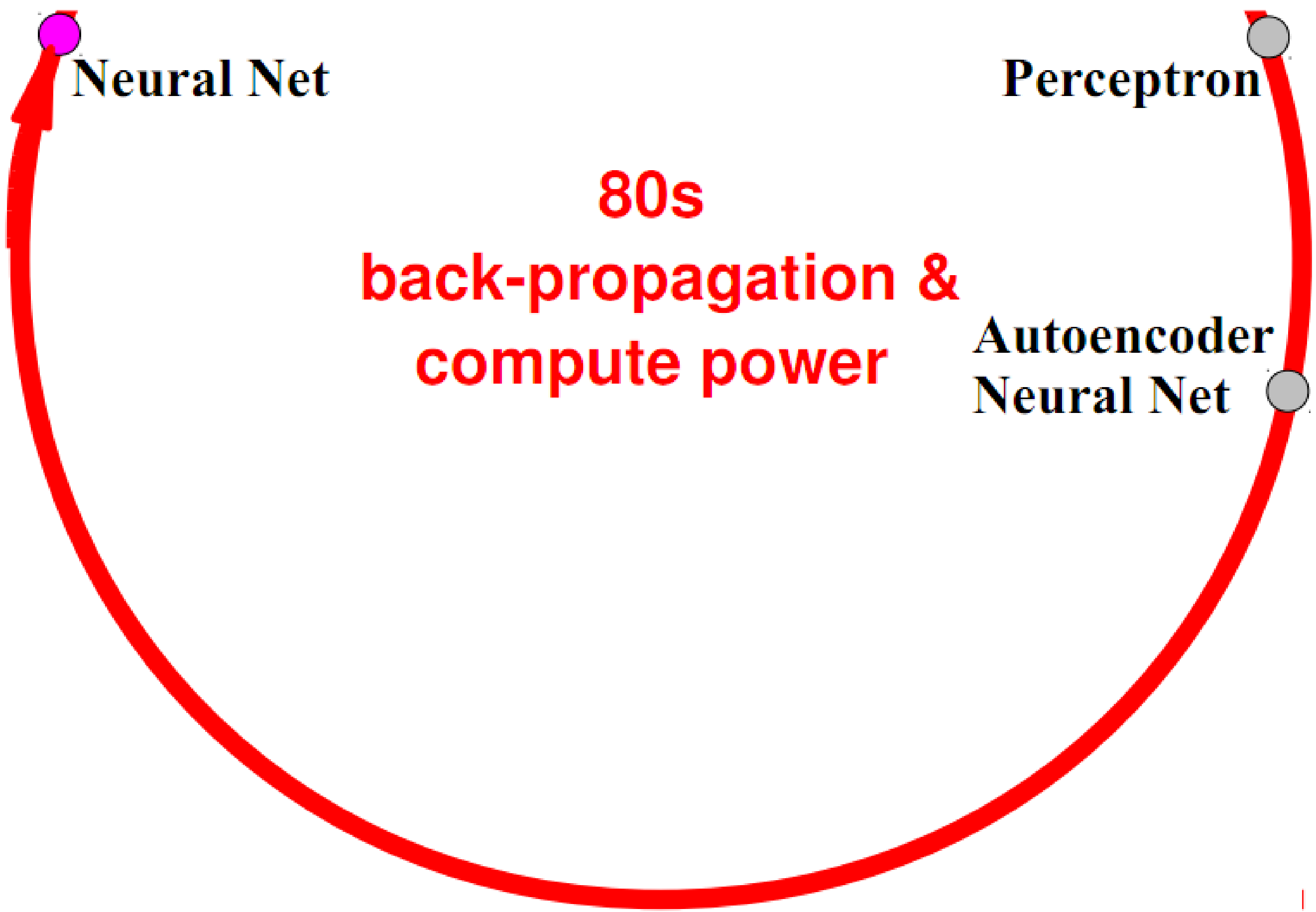
History of DL

Perceptron

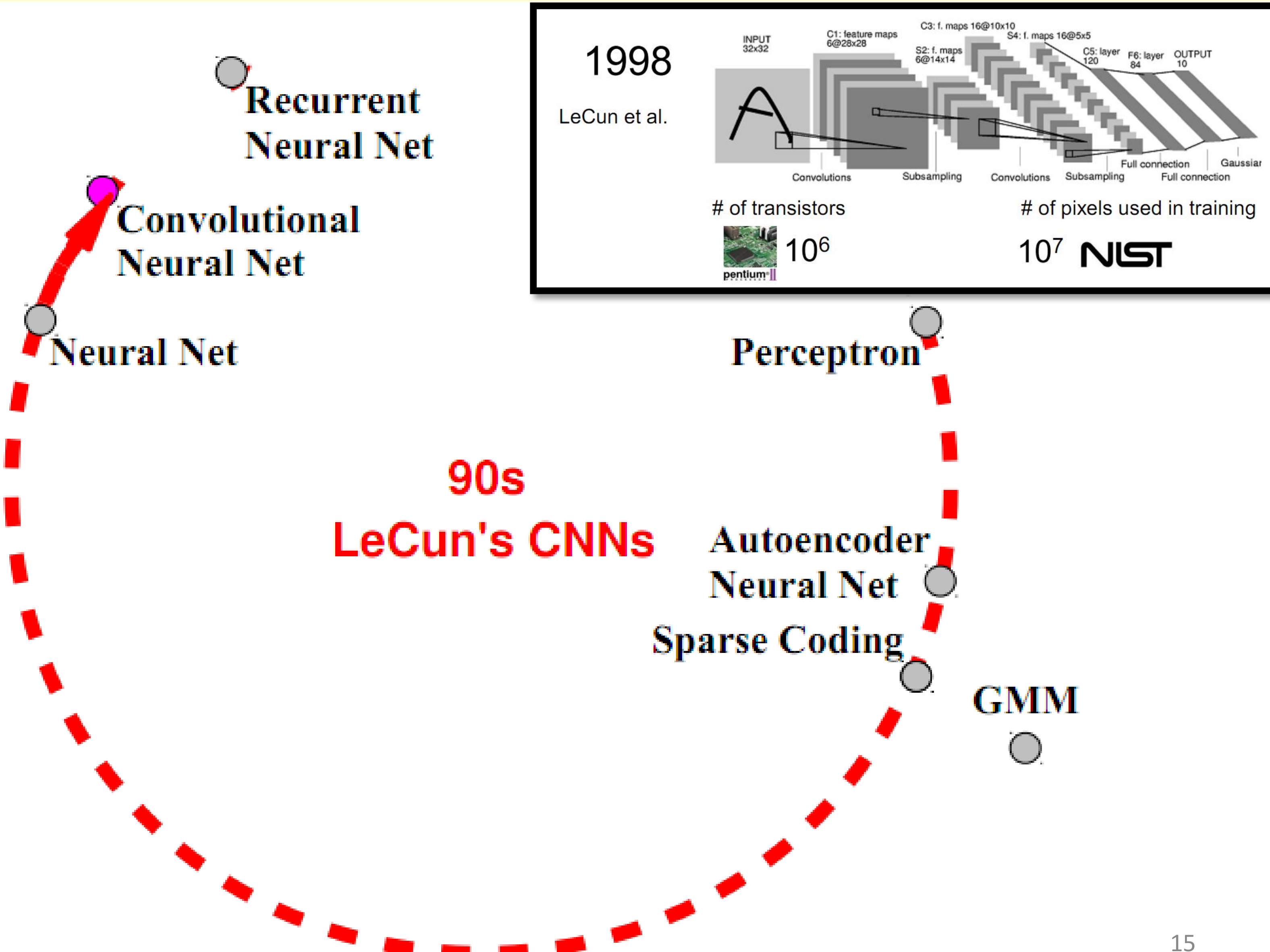
1957
Rosenblatt

THE SPACE OF
MACHINE LEARNING METHODS

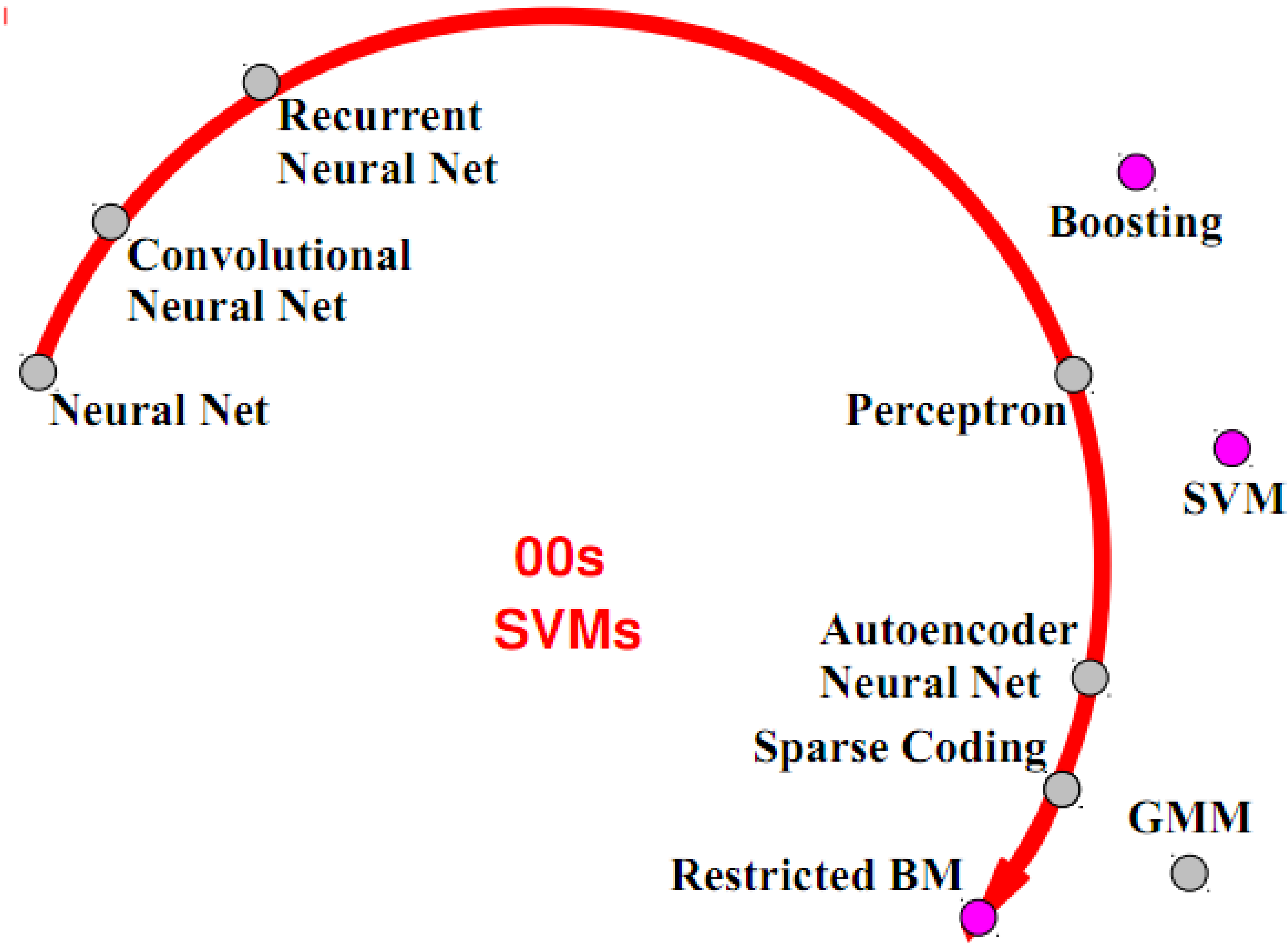
History of DL



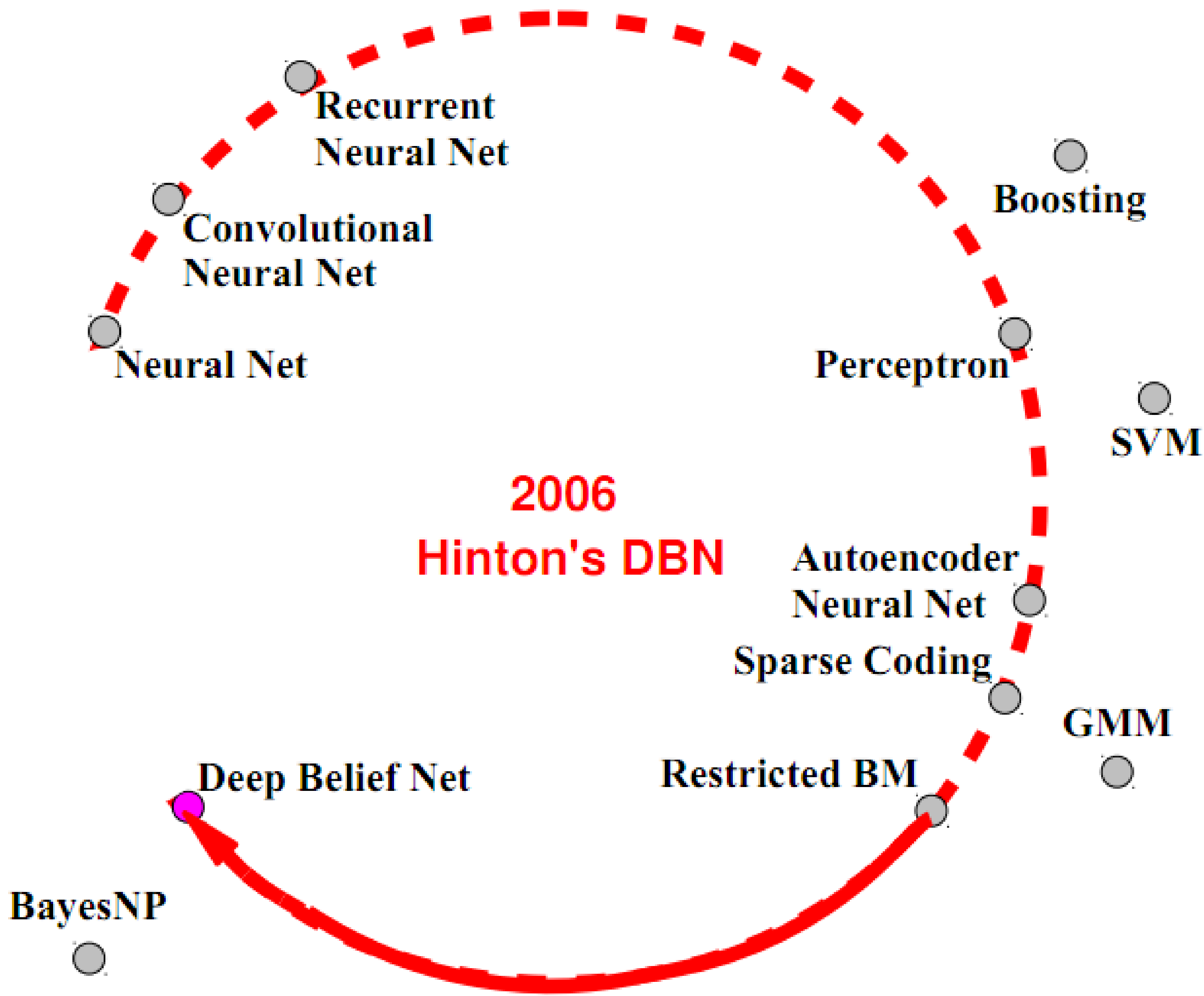
History of DL



History of DL

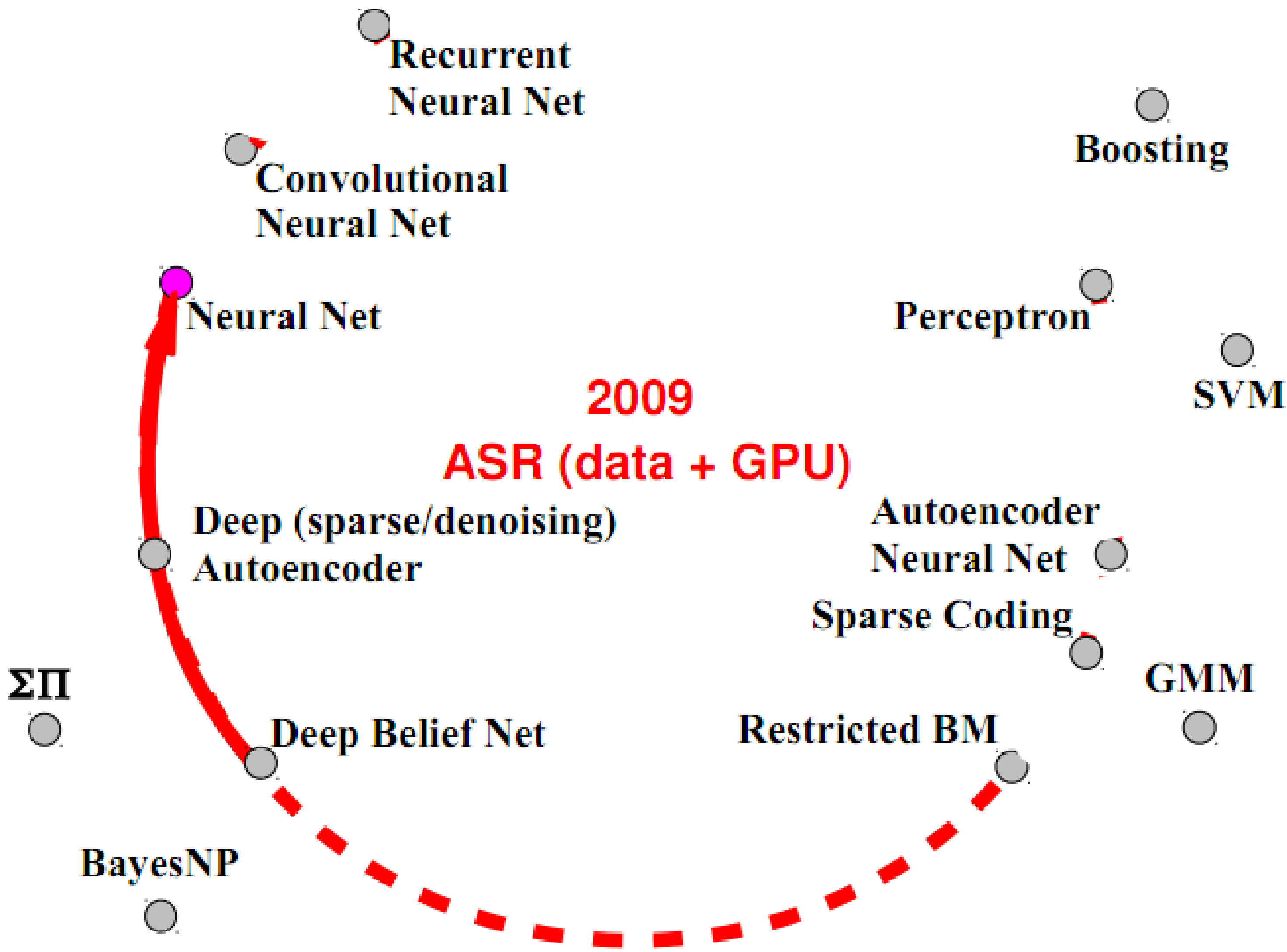


History of DL

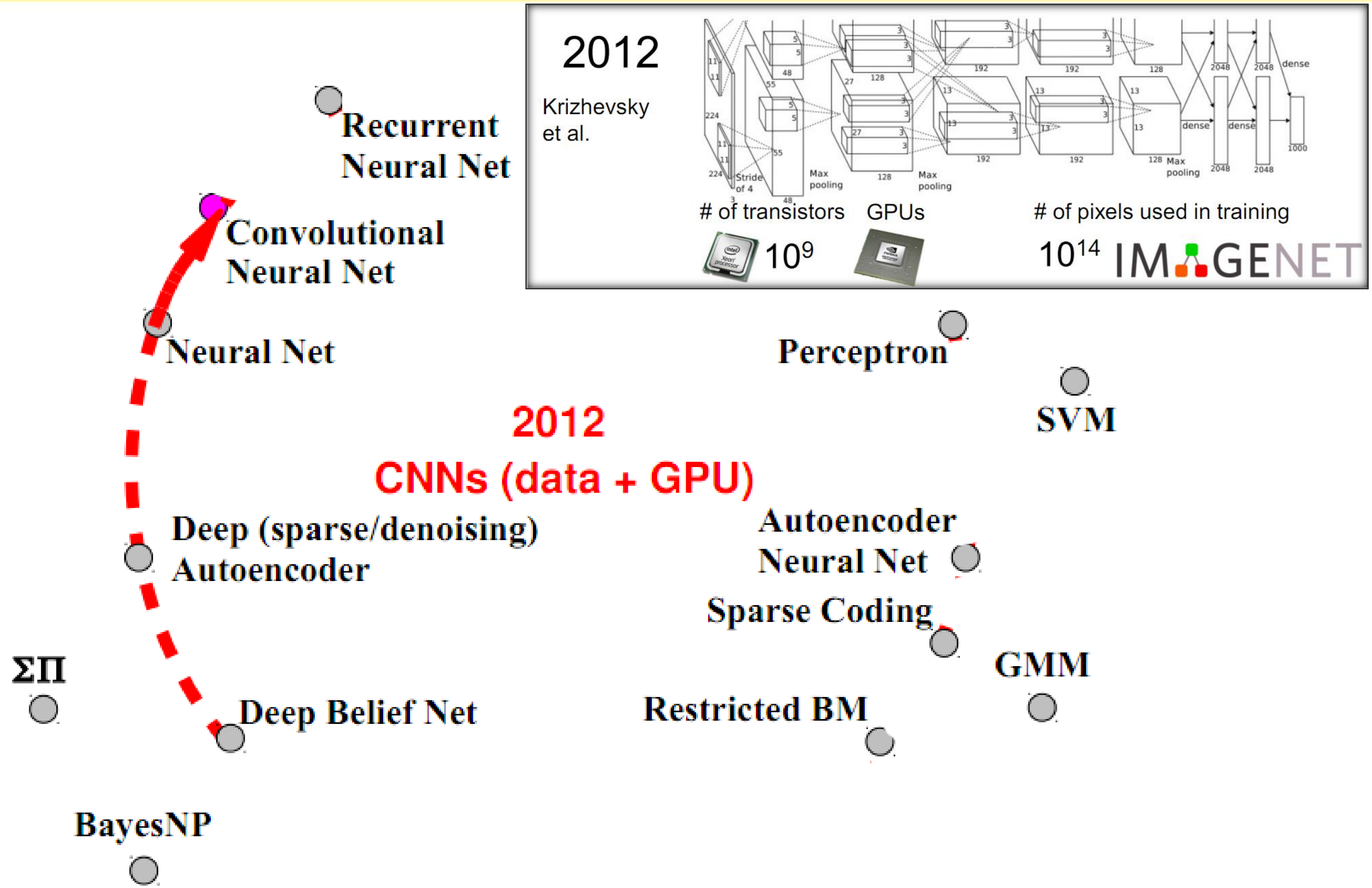


History of DL

2009



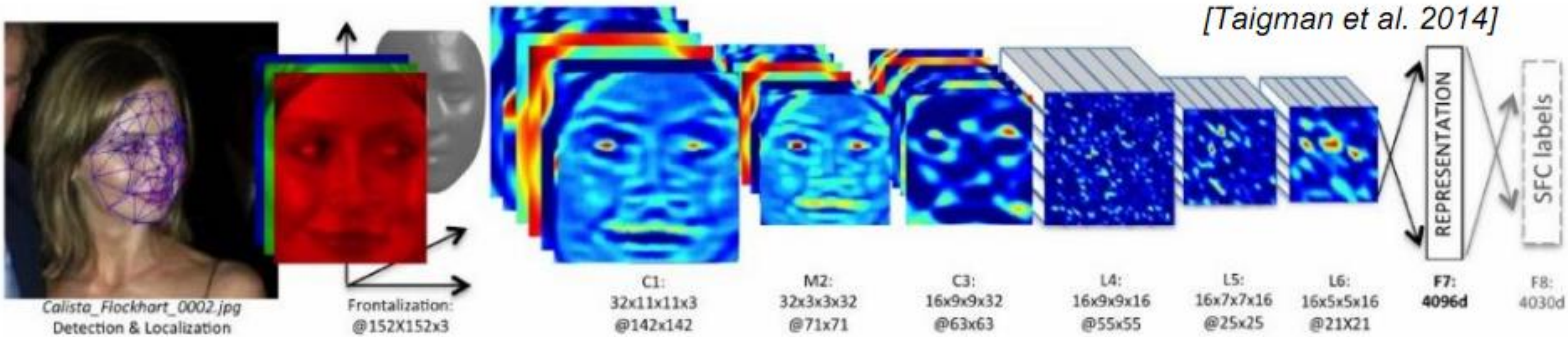
History of DL

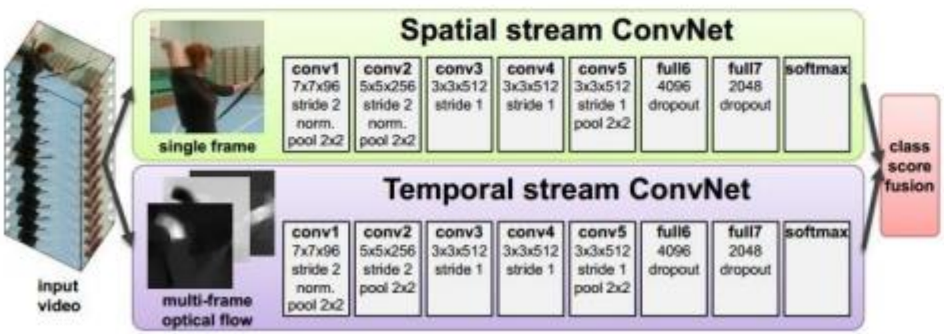


ConvNets

Fast-forward to today: ConvNets are everywhere

[Taigman et al. 2014]





[Simonyan et al. 2014]

[Goodfellow 2014]

ConvNets

Fast-forward to today: ConvNets are everywhere

Classification

Retrieval

[Krizhevsky 2012]

ConvNets

Fast-forward to today: ConvNets are everywhere

[Toshev, Szegedy 2014]

[Mnih 2013]

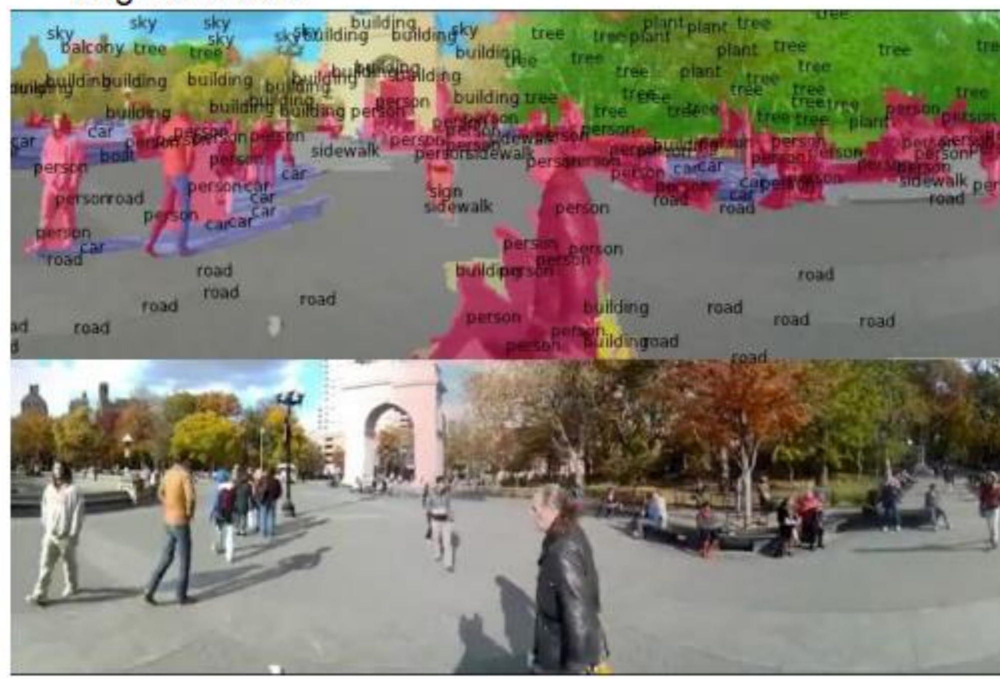
ConvNets

Fast-forward to today: ConvNets are everywhere

Detection

[Faster R-CNN: Ren, He, Girshick, Sun 2015]

Segmentation



[Farabet et al., 2012]

ConvNets

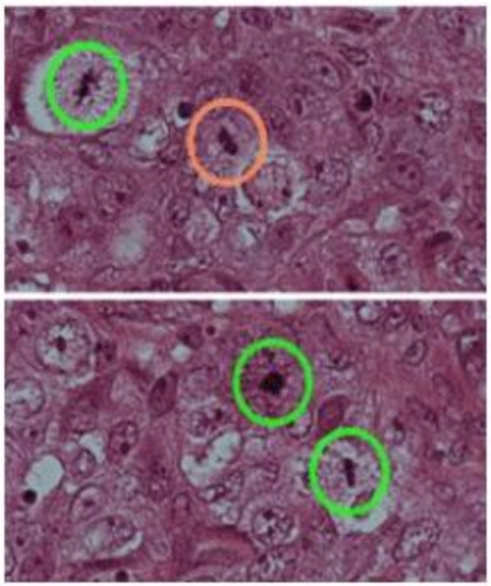
Fast-forward to today: ConvNets are everywhere

self-driving cars

NVIDIA Tegra X1

ConvNets

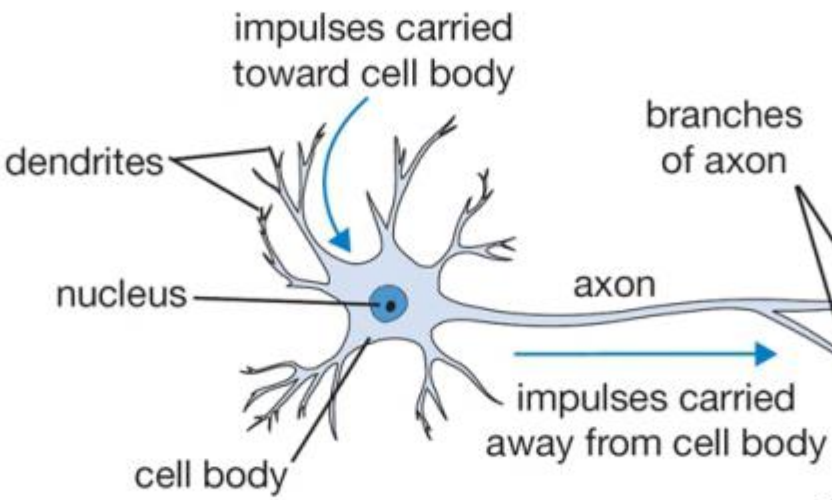
Fast-forward to today: ConvNets are everywhere

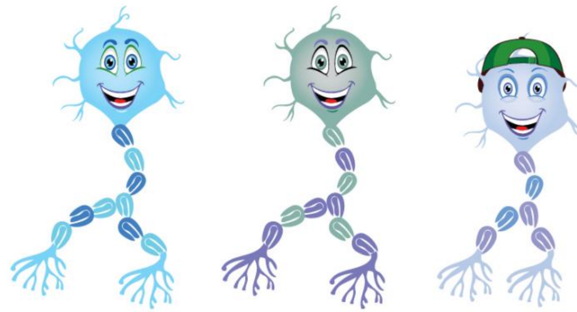


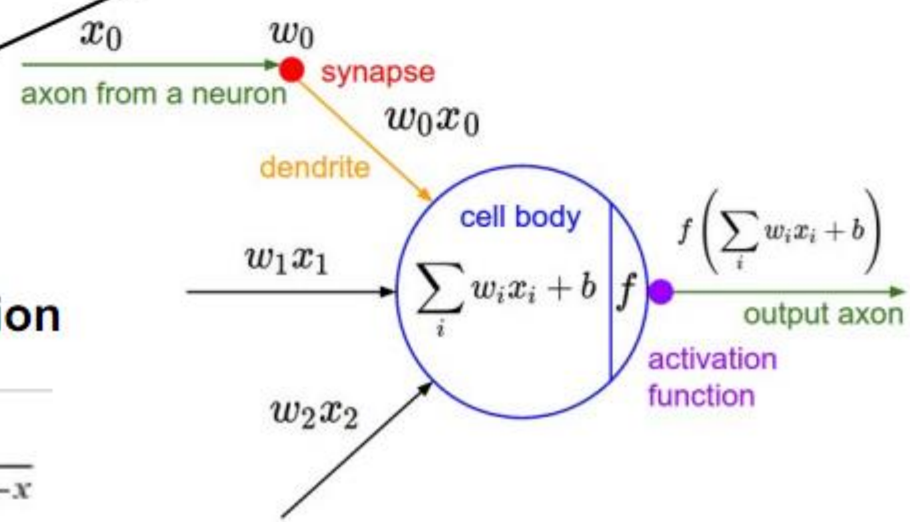
[Ciresan et al. 2013]

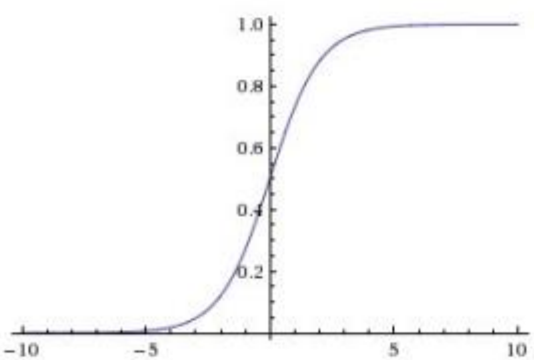
[Sermanet et al. 2011]
[Ciresan et al.]

Neuron







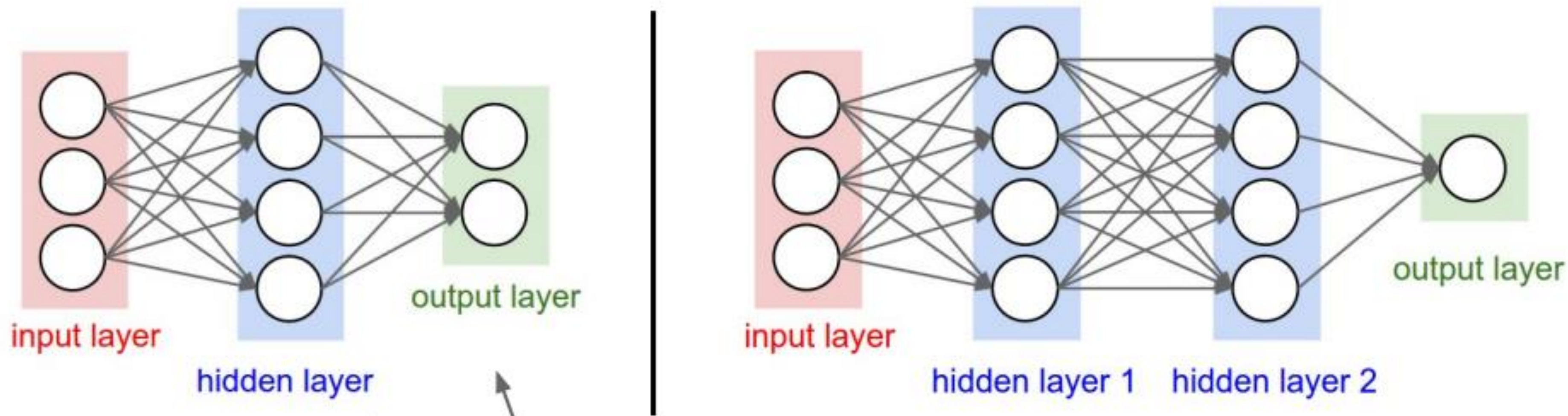


sigmoid activation function

$$\frac{1}{1+e^{-x}}$$

Neural Networks

Neural Networks: Architectures



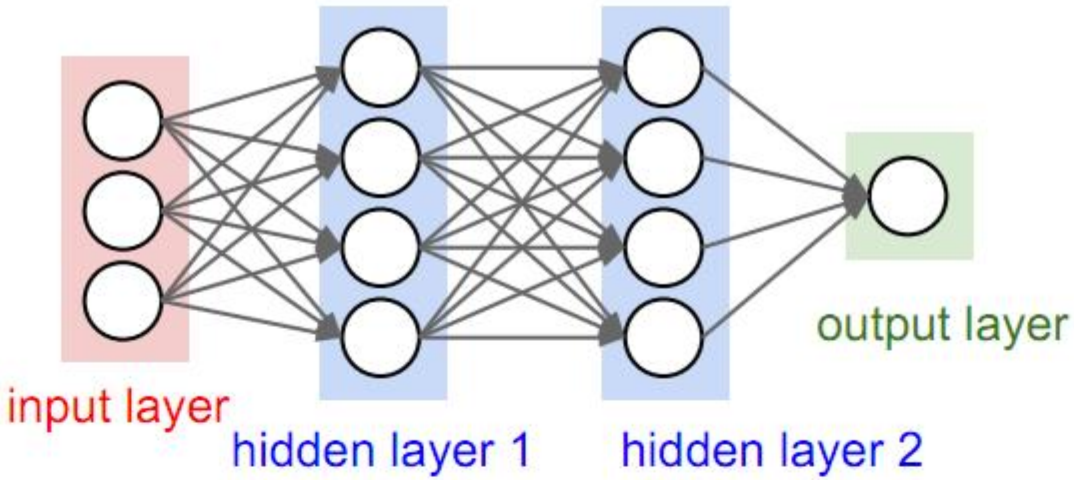
“2-layer Neural Net”, or
“1-hidden-layer Neural Net”

“Fully-connected” layers

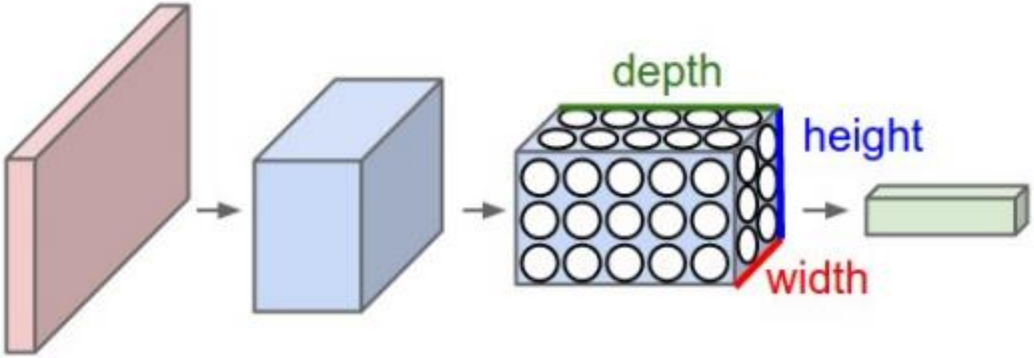
“3-layer Neural Net”, or
“2-hidden-layer Neural Net”

ConvNets

before:



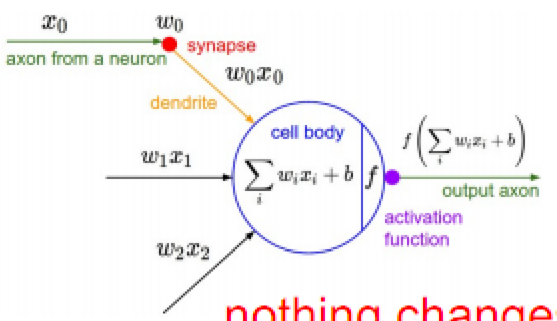
now:



ConvNets

Convolutional Neural Networks
are just Neural Networks BUT:

1. **Local connectivity**



nothing changes

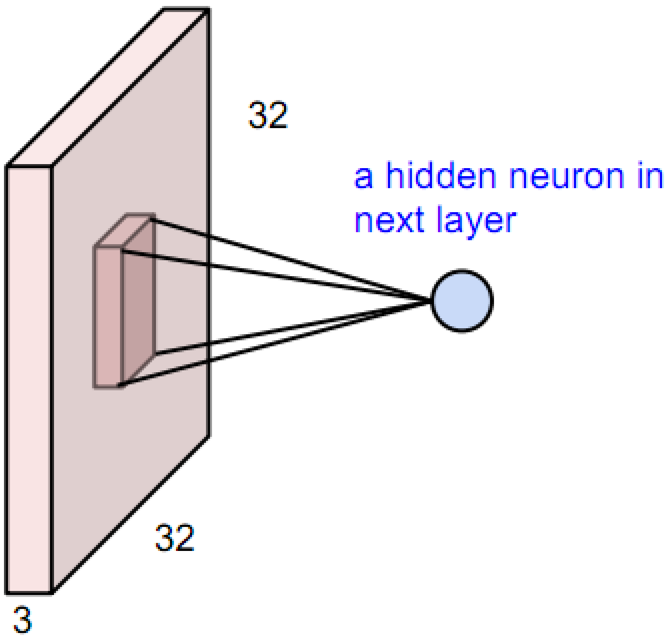


image: 32x32x3 volume
before: full connectivity: 32x32x3 weights

now: one neuron will connect to, e.g. 5x5x3 chunk and only have 5x5x3 weights.

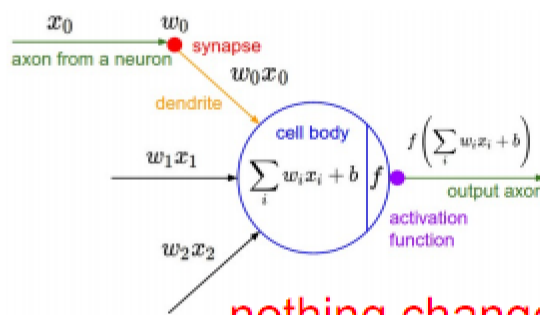
note that connectivity is:

- <u>local in space</u> (5x5 inside 32x32)
- <u>but full in depth</u> (all 3 depth channels)

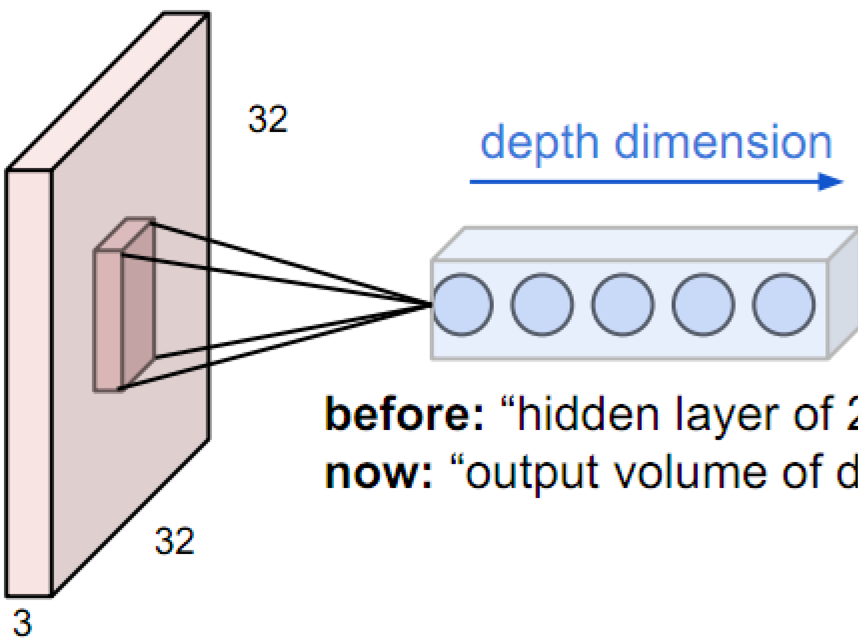
ConvNets

Convolutional Neural Networks are just Neural Networks BUT:

1. **Local connectivity**



nothing changes



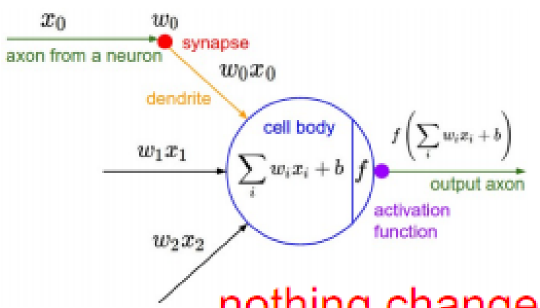
before: "hidden layer of 200 neurons"
now: "output volume of depth 200"

Multiple neurons all looking at the same region of the input volume, stacked along depth.

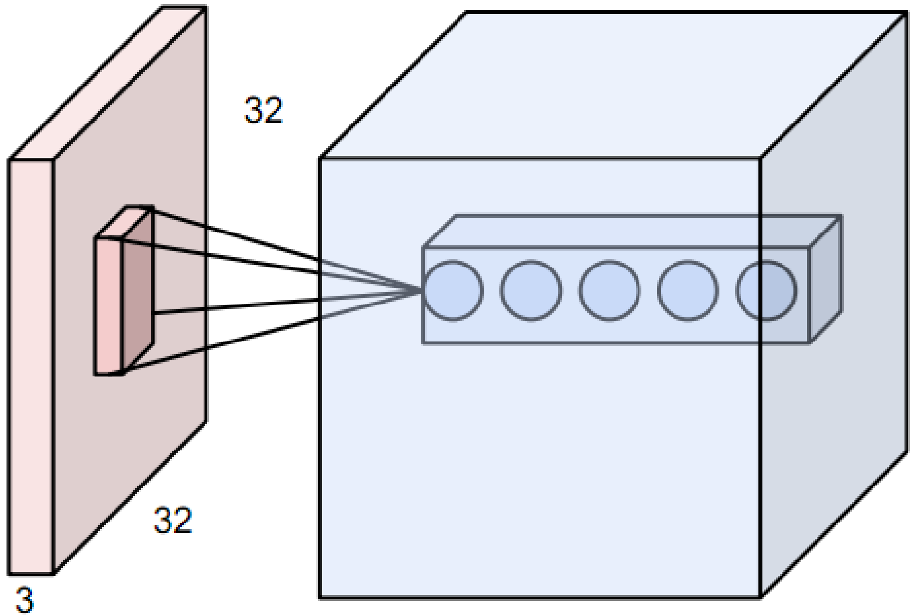
ConvNets

Convolutional Neural Networks
are just Neural Networks BUT:

1. **Local connectivity**



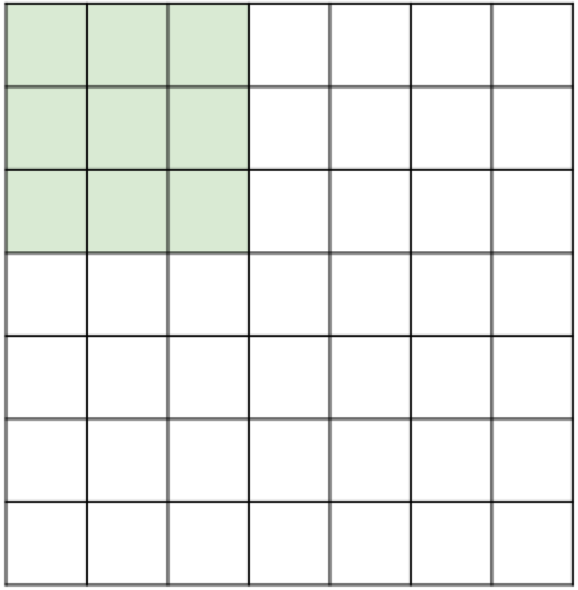
nothing changes



These form a single
[1 x 1 x depth]
“depth column” in the
output volume

ConvNets

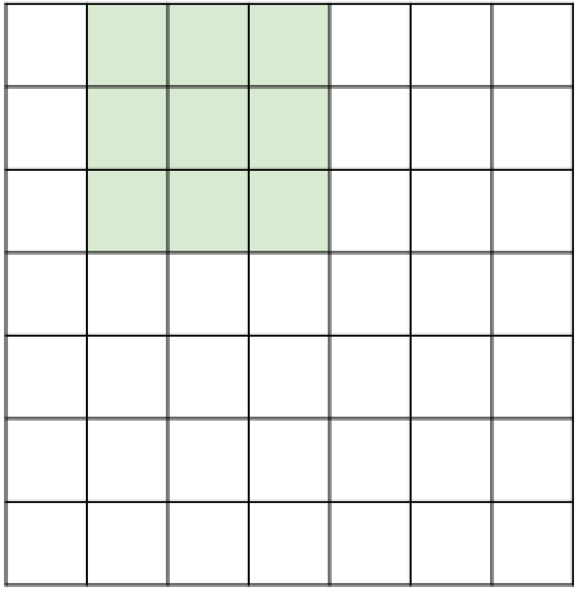
Replicate this column of hidden neurons across space, with some **stride.**



7x7 input
assume 3x3 connectivity, stride 1

ConvNets

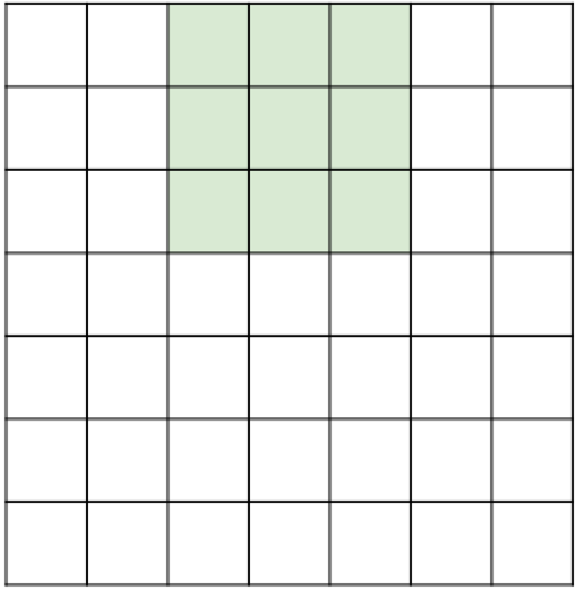
Replicate this column of hidden neurons across space, with some **stride.**



7x7 input
assume 3x3 connectivity, stride 1

ConvNets

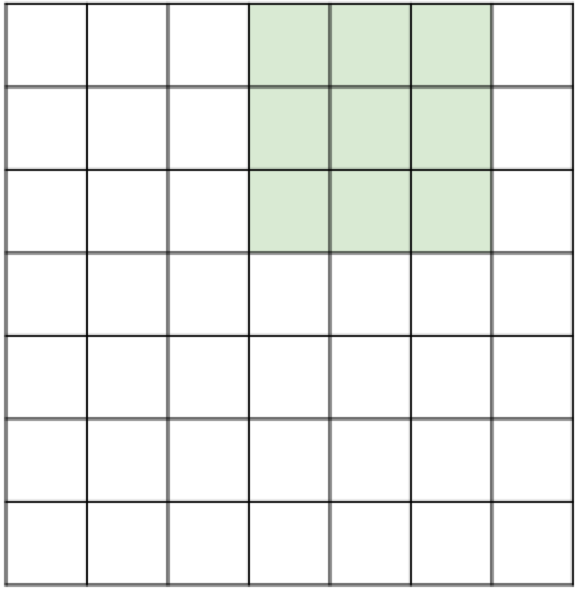
Replicate this column of hidden neurons across space, with some **stride.**



7x7 input
assume 3x3 connectivity, stride 1

ConvNets

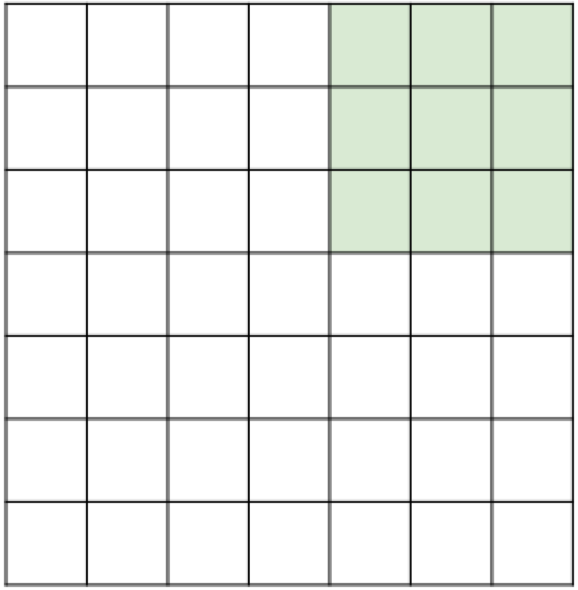
Replicate this column of hidden neurons across space, with some **stride.**



7x7 input
assume 3x3 connectivity, stride 1

ConvNets

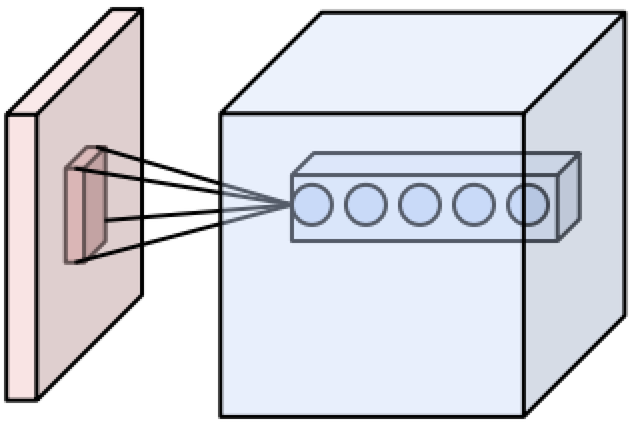
Replicate this column of hidden neurons across space, with some **stride.**



7x7 input
assume 3x3 connectivity, stride 1
=> **5x5 output**

ConvNets

Examples time:



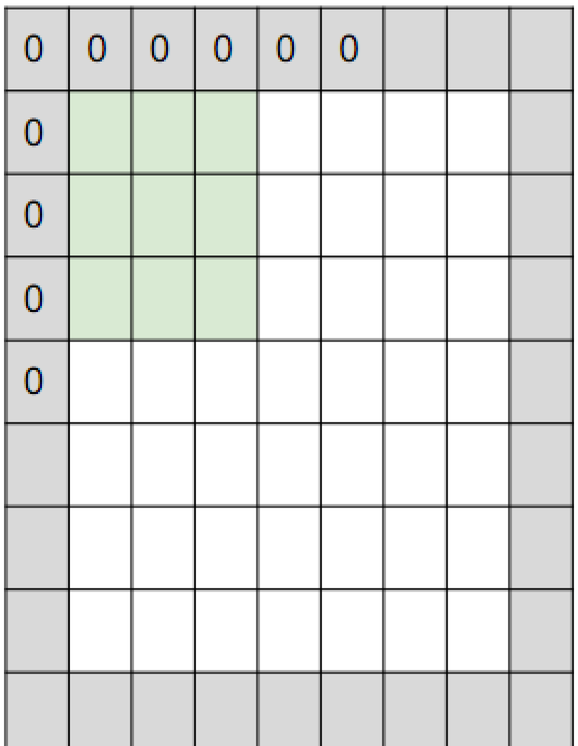
Input volume: **32x32x3**
Receptive fields: **5x5**, **stride 1**
Number of neurons: **5**

Output volume: (32 - 5) / 1 + 1 = 28, so: **28x28x5**
How many weights for each of the 28x28x5 neurons?

ConvNets

In practice: Common to zero pad the border

(in each channel)



e.g. input 7x7
neuron with receptive field 3x3, stride 1
pad with 1 pixel border => what is the output?

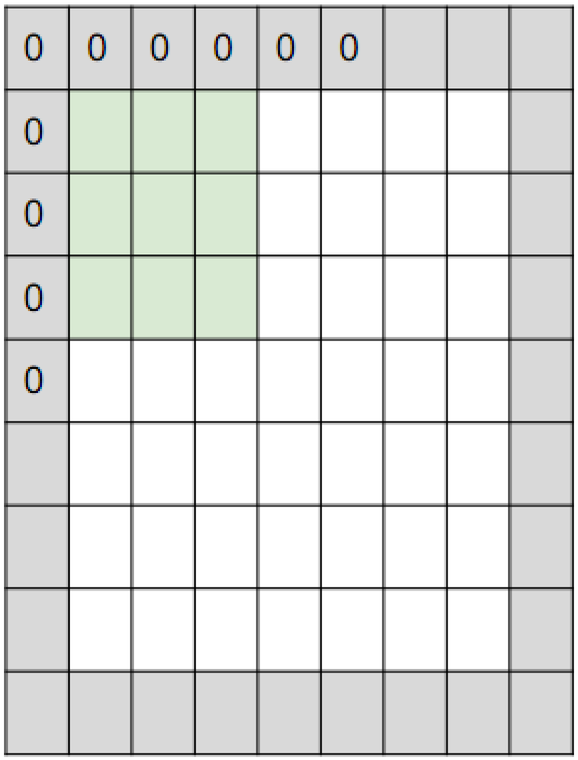
7x7 => preserved size!

in general, common to see stride 1, size F, and zero-padding with (F-1)/2.
(Will preserve input size spatially)

ConvNets

In practice: Common to zero pad the border

(in each channel)



e.g. input 7x7
neuron with receptive field 3x3, stride 1
pad with 1 pixel border => what is the output?

7x7 => preserved size!

in general, common to see stride 1, size F, and zero-padding with (F-1)/2.
(Will preserve input size spatially)

ConvNets

There’s one more problem…

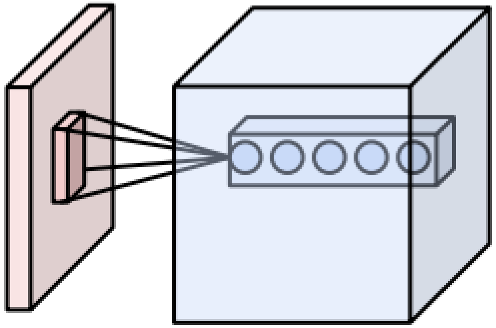
Assume input **[32 x 32 x3]**

30 neurons with receptive fields **5x5**, applied at **stride 1/pad1:**

=> Output volume: [32 x 32 x 30]   (32*32*30 = 30720 neurons)

Each neuron has 5*5*3 (=75) weights

=> Number of weights in such layer: 30720 * 75 ~= **3 million :**



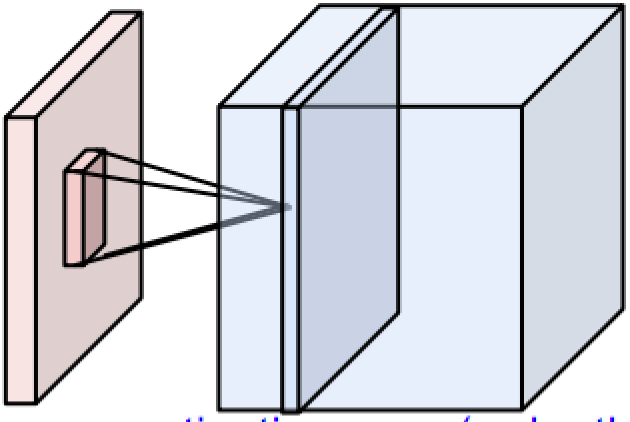
Example trained weights

IDEA: lets not learn the same thing across all spatial locations

ConvNets

These layers are called **Convolutional Layers**

1. Connect neurons only to local receptive fields
2. Use the same neuron weight parameters for neurons in each “depth slice” (i.e. across spatial positions)

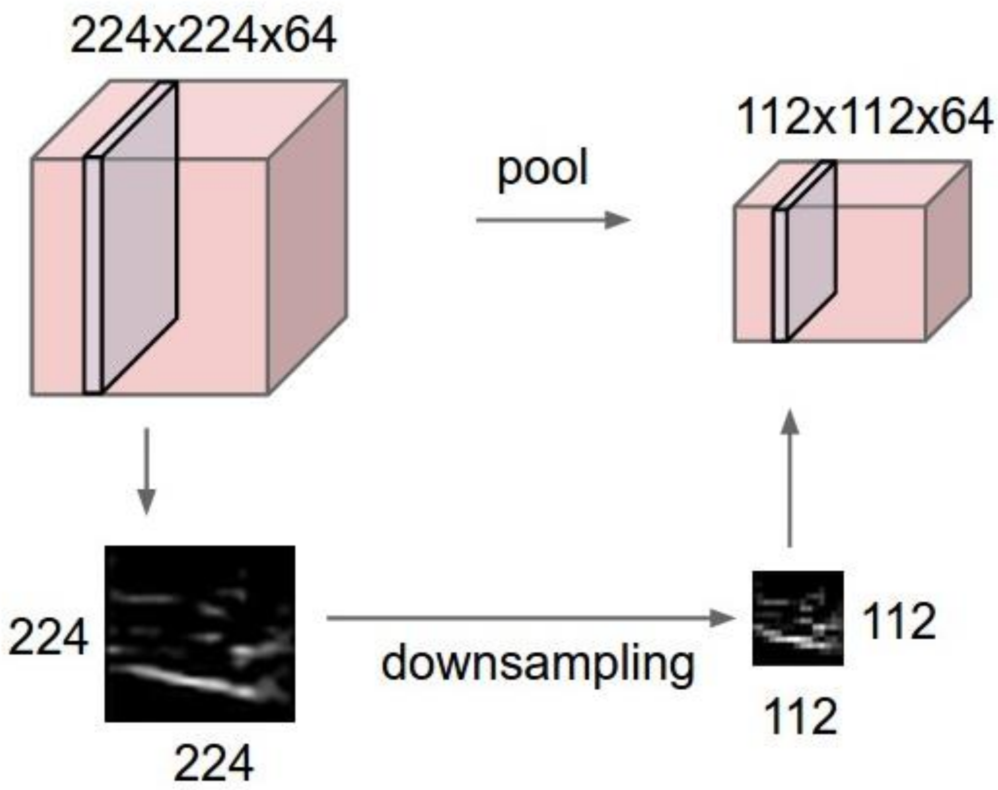


one activation map (a depth slice), computed with one set of weights

ConvNets

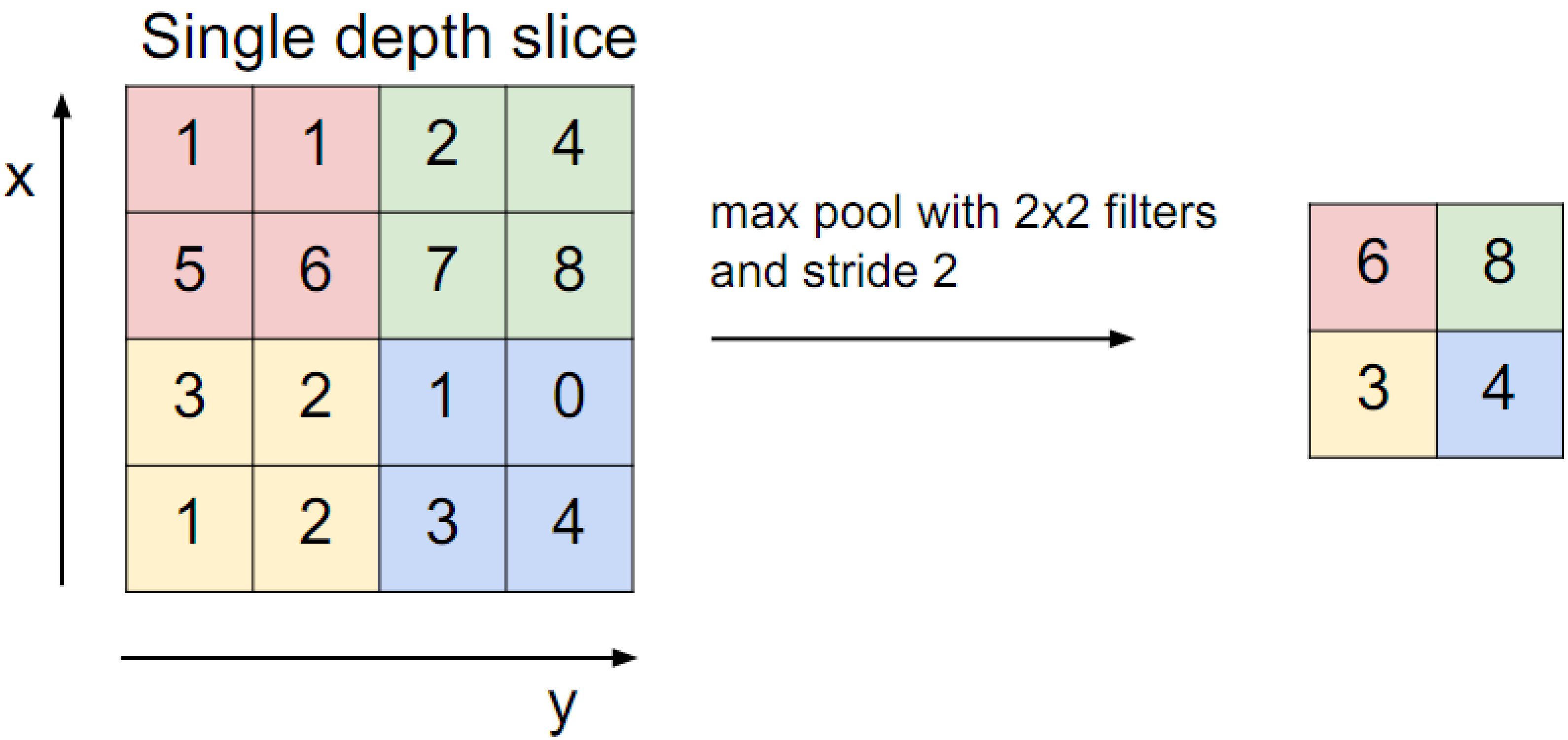
In ConvNet architectures, **Conv** layers are often followed by **Pool** layers

- convenience layer: makes the representations smaller and more manageable without losing too much information. Computes MAX operation (most common)

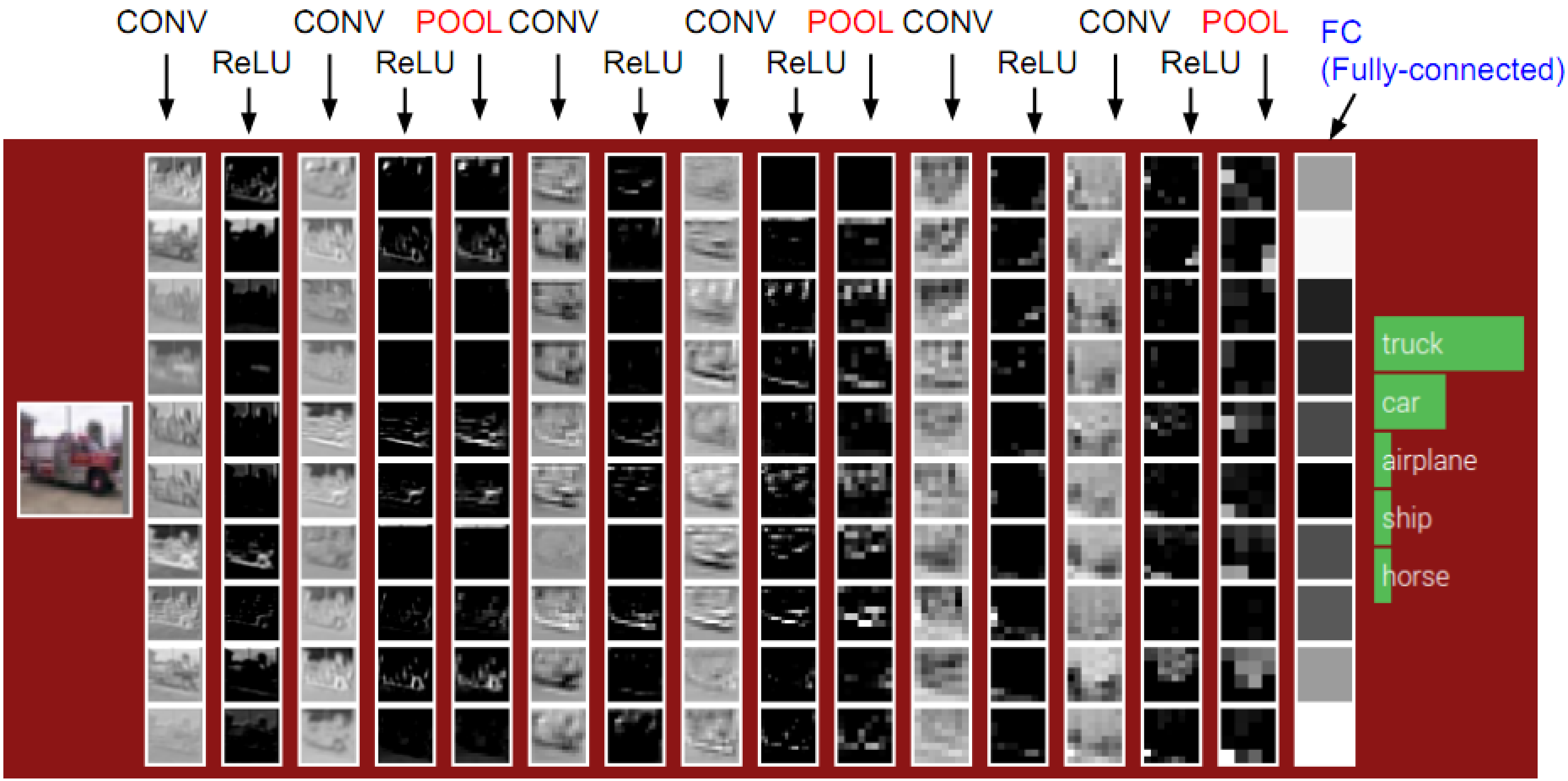


ConvNets

MAX POOLING



ConvNets



CONV
ReLU
CONV
ReLU
POOL
CONV
ReLU
CONV
ReLU
POOL
CONV
ReLU
CONV
ReLU
POOL
FC
(Fully-connected)
truck
car
airplane
ship
horse

ConvNets

Modern CNNs:

- use **filter sizes of 3x3** (maybe even 2x2 or 1x1!)
- use **pooling sizes of 2x2** (maybe even less - e.g. fractional pooling!)
- **stride 1**
- **very deep**

INPUT -> [[CONV -> RELU]*N -> POOL?]*M -> [FC -> RELU]*K -> FC
where the * indicates repetition, and the POOL? indicates an optional pooling layer.
N >= 0 (and usually N <= 3), M >= 0, K >= 0 (and usually K < 3).

ConvNets

Case study: VGGNet / OxfordNet
(runner-up winner of ILSVRC 2014)
[Simonyan and Zisserman]

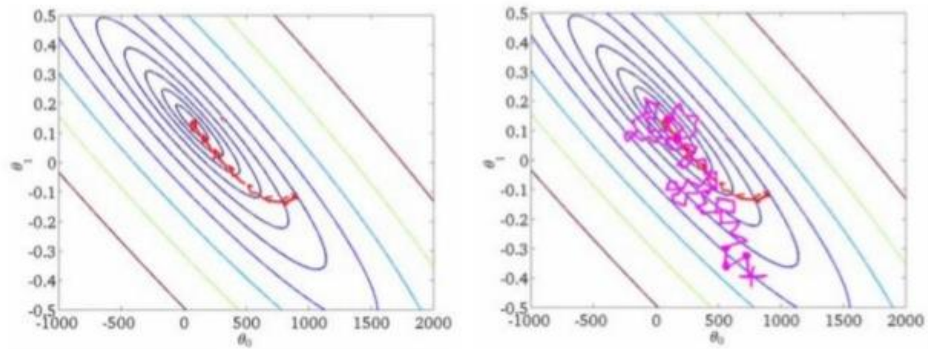
best model

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Table 2: **Number of parameters** (in millions).

Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

Training ConvNets



Batch: gradient

$$x \leftarrow x - \eta \nabla F(x)$$

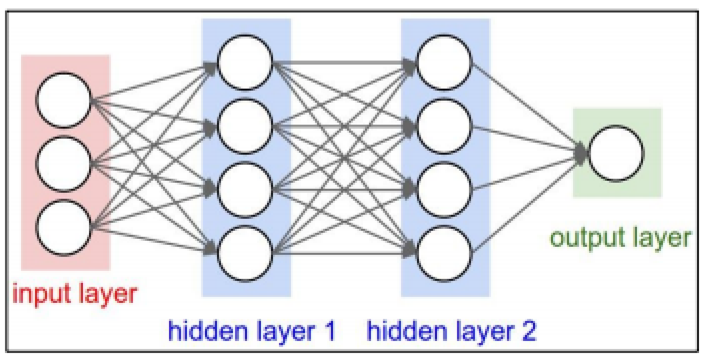
Stochastic: single-example gradient

$$x \leftarrow x - \eta \nabla F_i(x)$$

Mini-batch SGD

Loop:

1. **Sample** a batch of data
2. **Forward** prop it through the graph, get loss
3. **Backprop** to calculate the gradients
4. **Update** the parameters using the gradient

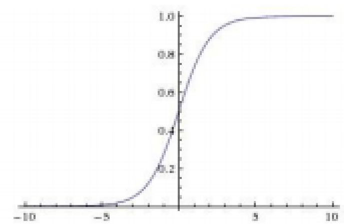


Training ConvNets

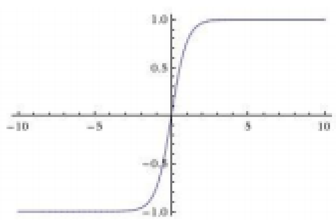
Activation Functions

Sigmoid

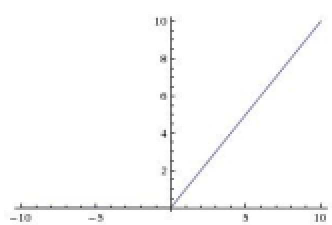
$$\sigma(x) = 1/(1+e^{-x})$$



tanh tanh(x)

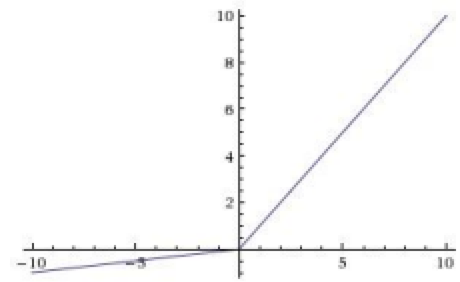


ReLU max(0,x)



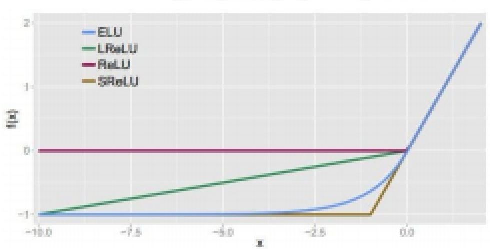
Leaky ReLU

max(0.1x, x)



Maxout $\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU $f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha\,(\exp(x) - 1) & \text{if } x \le 0 \end{cases}$

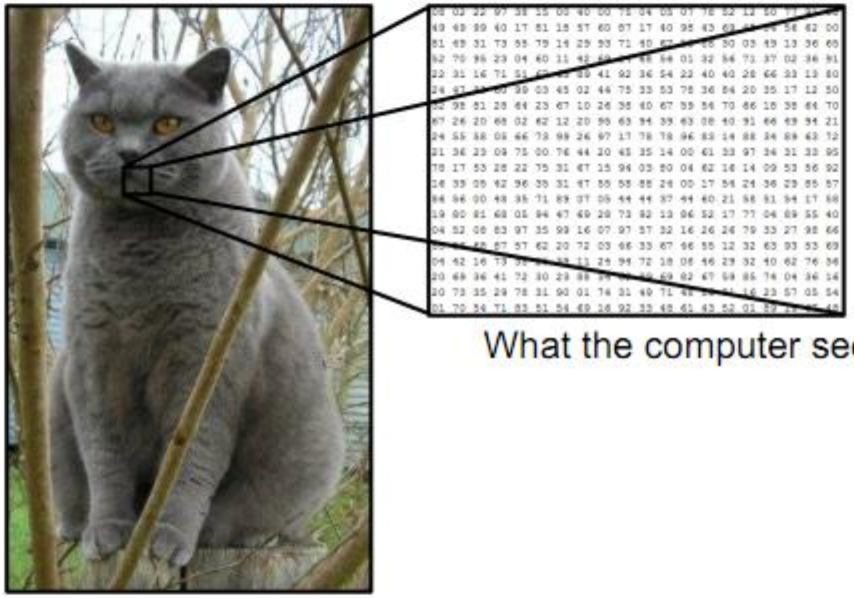


Data Augmentation

Data Augmentation

- i.e. simulating “fake” data

- explicitly encoding image transformations that shouldn’t change object identity.

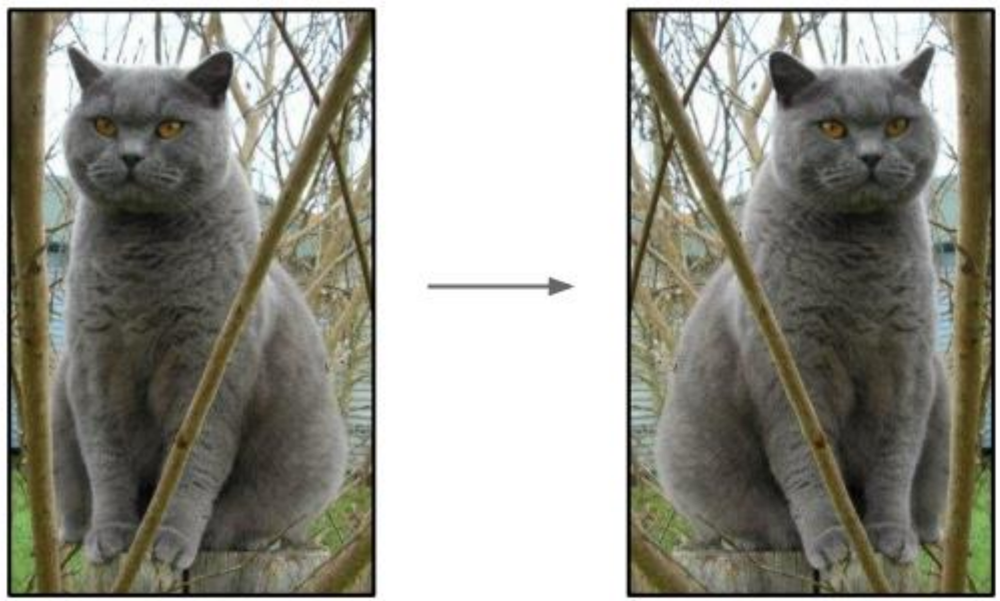


What the computer sees

Data Augmentation

Data Augmentation

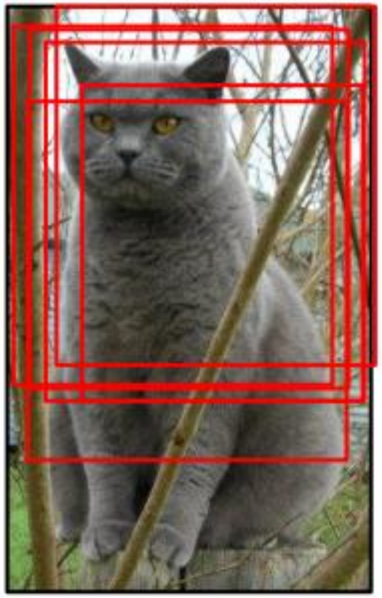
1. Flip horizontally



Data Augmentation

Data Augmentation

2. Random crops/scales



Sample these during training
(also helps a lot during test time)

e.g. common to see even up to 150 crops used

Data Augmentation

Data Augmentation

3.

Random mix/combinations of :

- translation
- rotation
- stretching
- shearing,
- lens distortions, … (go crazy)

Data Augmentation

Data Augmentation

4. Color jittering

(maybe even contrast jittering, etc.)

- Simple: Change contrast small amounts, jitter the color distributions, etc.

- Vignette,... (go crazy)

Transfer learning

"You need a lot of a data if you want to train/use CNNs"

Transfer learning

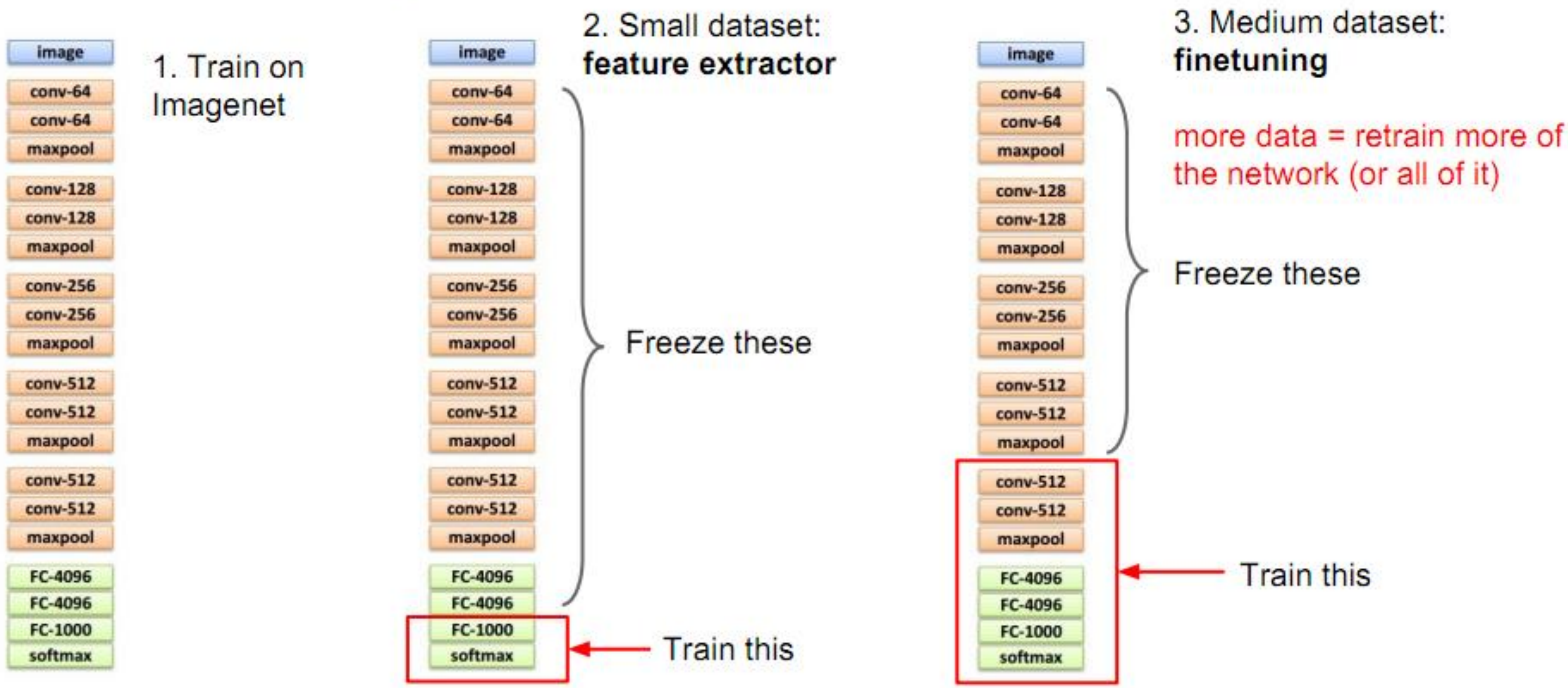
Transfer Learning

"You need a lot of a data if you want to train/use CNNs"

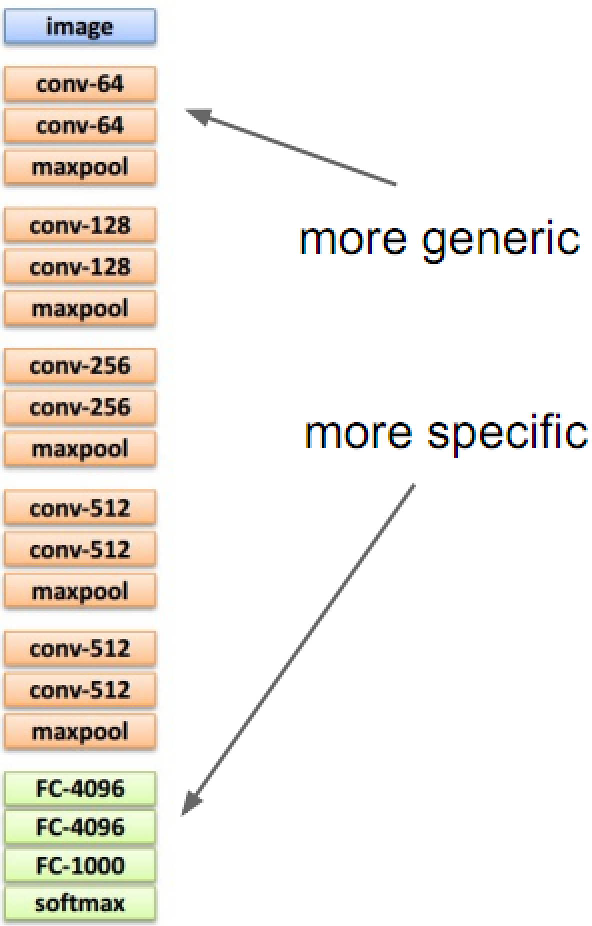
BUSTED

Transfer learning

Transfer Learning with CNNs



Transfer learning

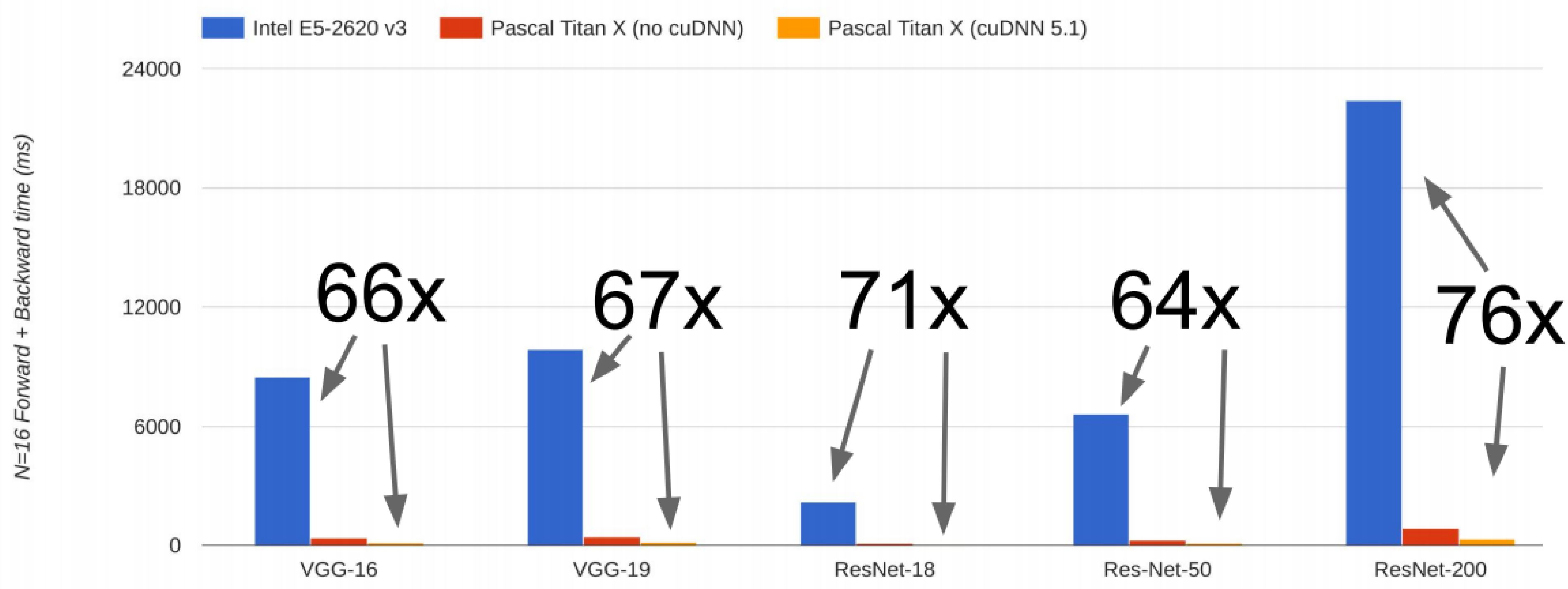


	very similar dataset	**very different dataset**
very little data	Use Linear Classifier on top layer	You're in trouble… Try linear classifier from different stages
quite a lot of data	Finetune a few layers	Finetune a larger number of layers

CPU vs GPU

CPU vs GPU in practice

(CPU performance not well-optimized, a little unfair)



CPU vs GPU

	Cores	Clock Speed	Memory	Price	Speed
CPU (Intel Core i7-7700k)	4 (8 threads with hyperthreading)	4.2 GHz	System RAM	$339	~540 GFLOPs FP32
GPU (NVIDIA GTX 1080 Ti)	3584	1.6 GHz	11 GB GDDR5 X	$699	~11.4 TFLOPs FP32
TPU NVIDIA TITAN V	5120 CUDA, 640 Tensor	1.5 GHz	12GB HBM2	$2999	~14 TFLOPs FP32 ~112 TFLOP FP16
TPU Google Cloud TPU	?	?	64 GB HBM	$6.50 per hour	~180 TFLOP

CPU: Fewer cores, but each core is much faster and much more capable; great at sequential tasks

GPU: More cores, but each core is much slower and "dumber"; great for parallel tasks

TPU: Specialized hardware for deep learning

A zoo of frameworks

Caffe
(UC Berkeley) → Caffe2
(Facebook)

Torch
(NYU / Facebook) → PyTorch
(Facebook)

Theano
(U Montreal) → TensorFlow
(Google)

PaddlePaddle
(Baidu)

Chainer

MXNet
(Amazon)
Developed by U Washington, CMU, MIT, Hong Kong U, etc but main framework of choice at AWS

CNTK
(Microsoft)

Deeplearning4j

And others...