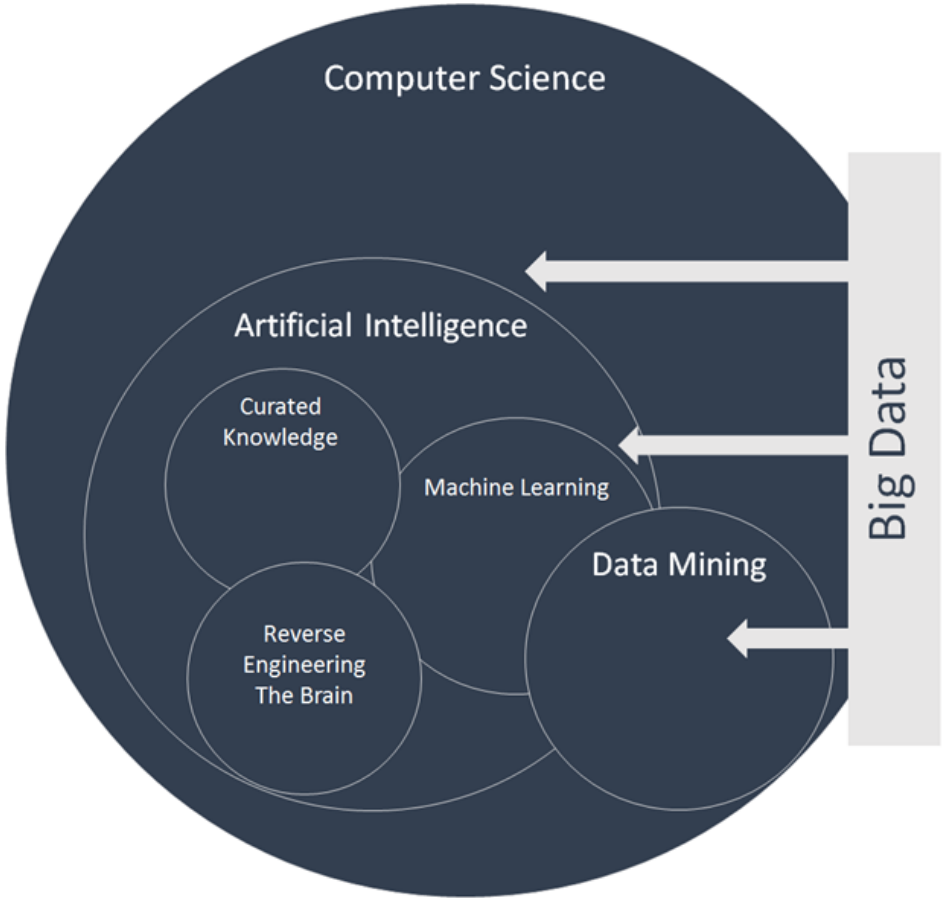
Deep Learning for Visual Recognition
- Part 1

Kyu-Hwan Jung, Ph.D
kyuhwanjung@gmail.com

Class 1
Overview of Deep Learning

Artificial Intelligence Diagram



~~Knowledge-based Approach~~

⇩

Data(Learning)-driven Approach

~~Computationalism~~

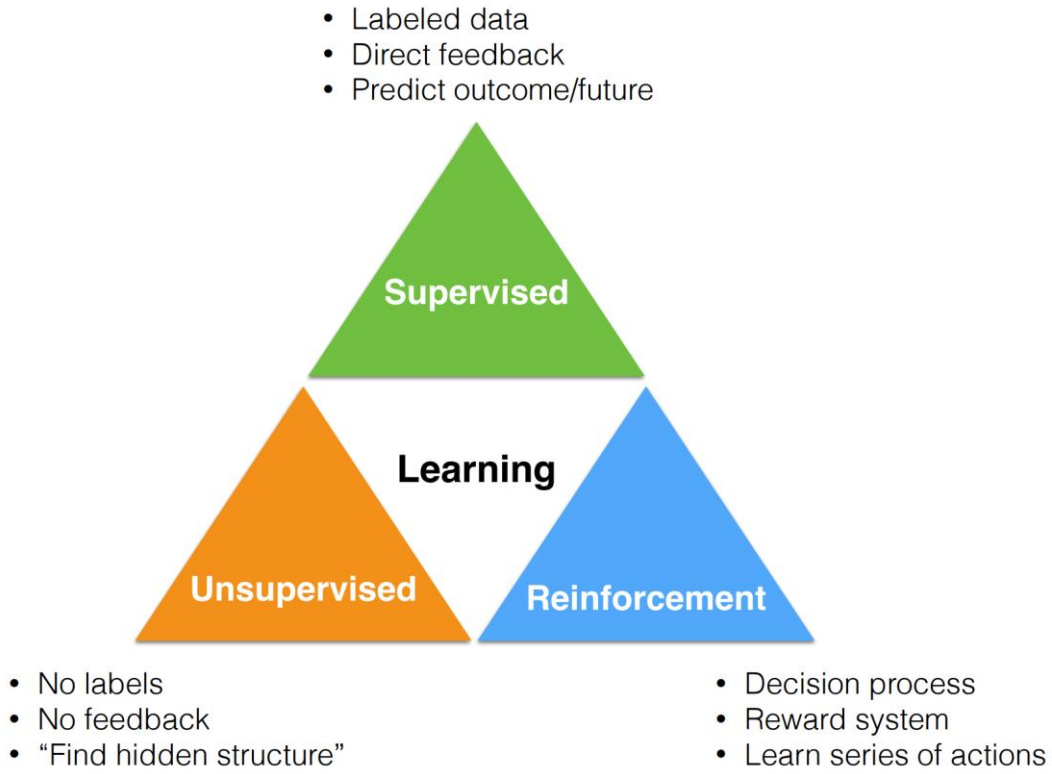
⇩

Connectionism

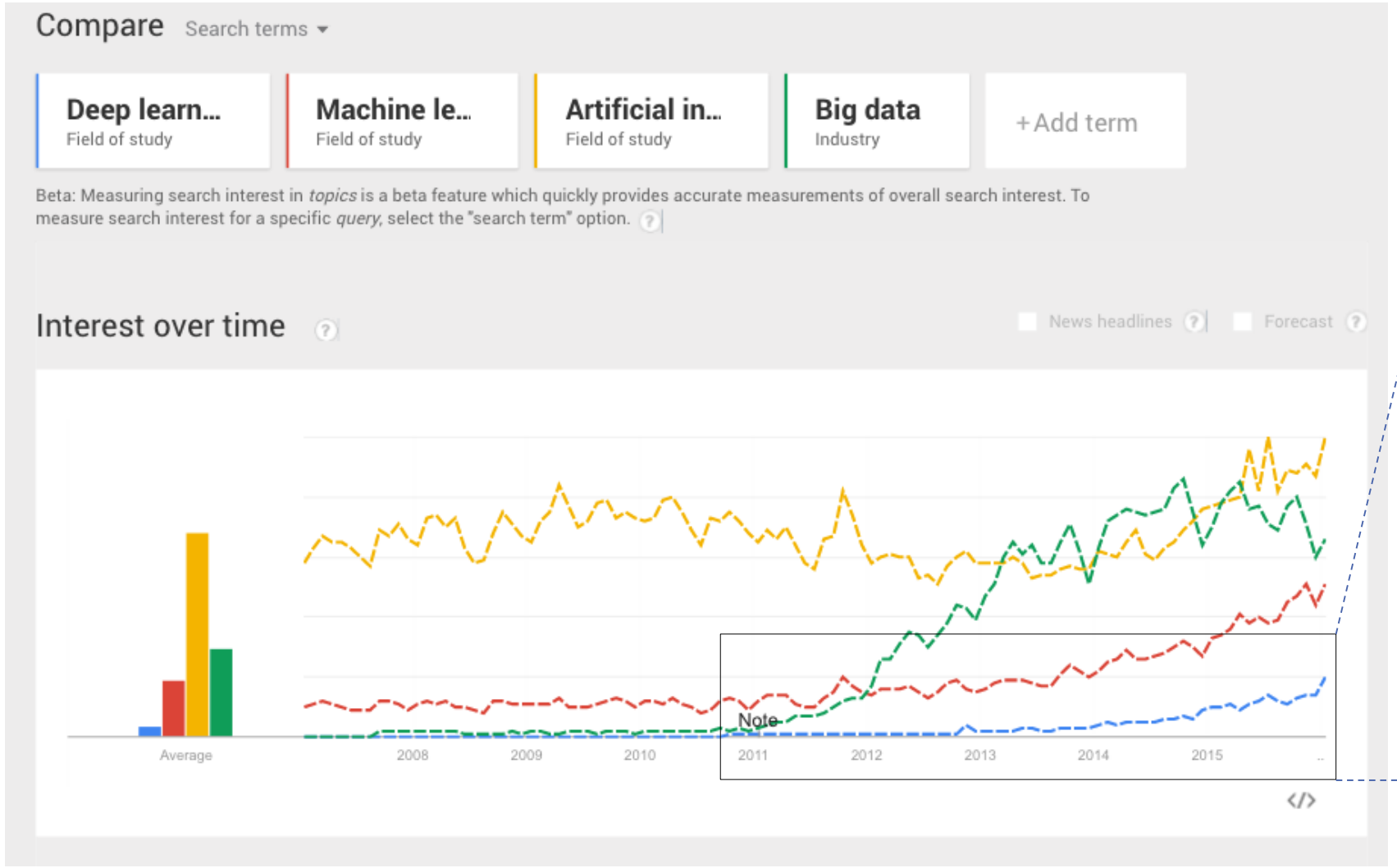
Definition of Machine Learning

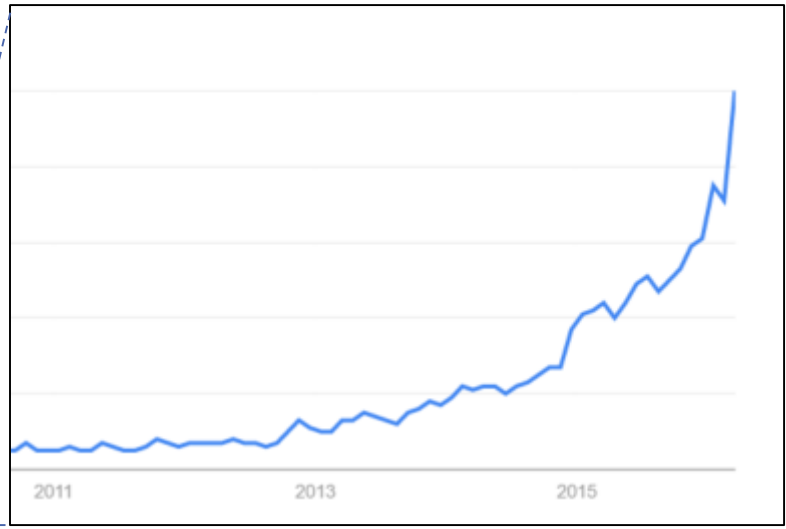
- "Field of study that gives computers the ability to learn without being explicitly programmed”

> A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.
>
> -Tom M. Mitchell

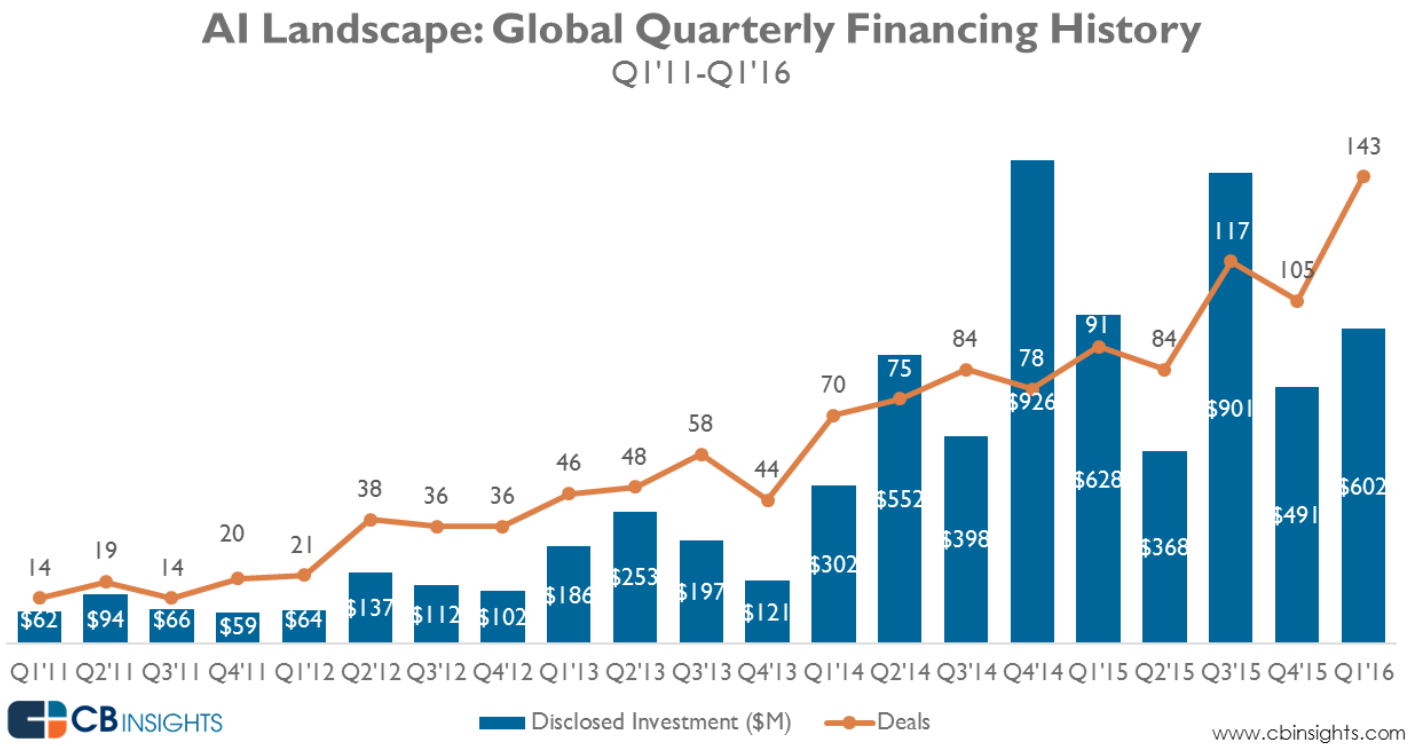


Google Trends – Machines are Learning





Smart Money is Moving to AI



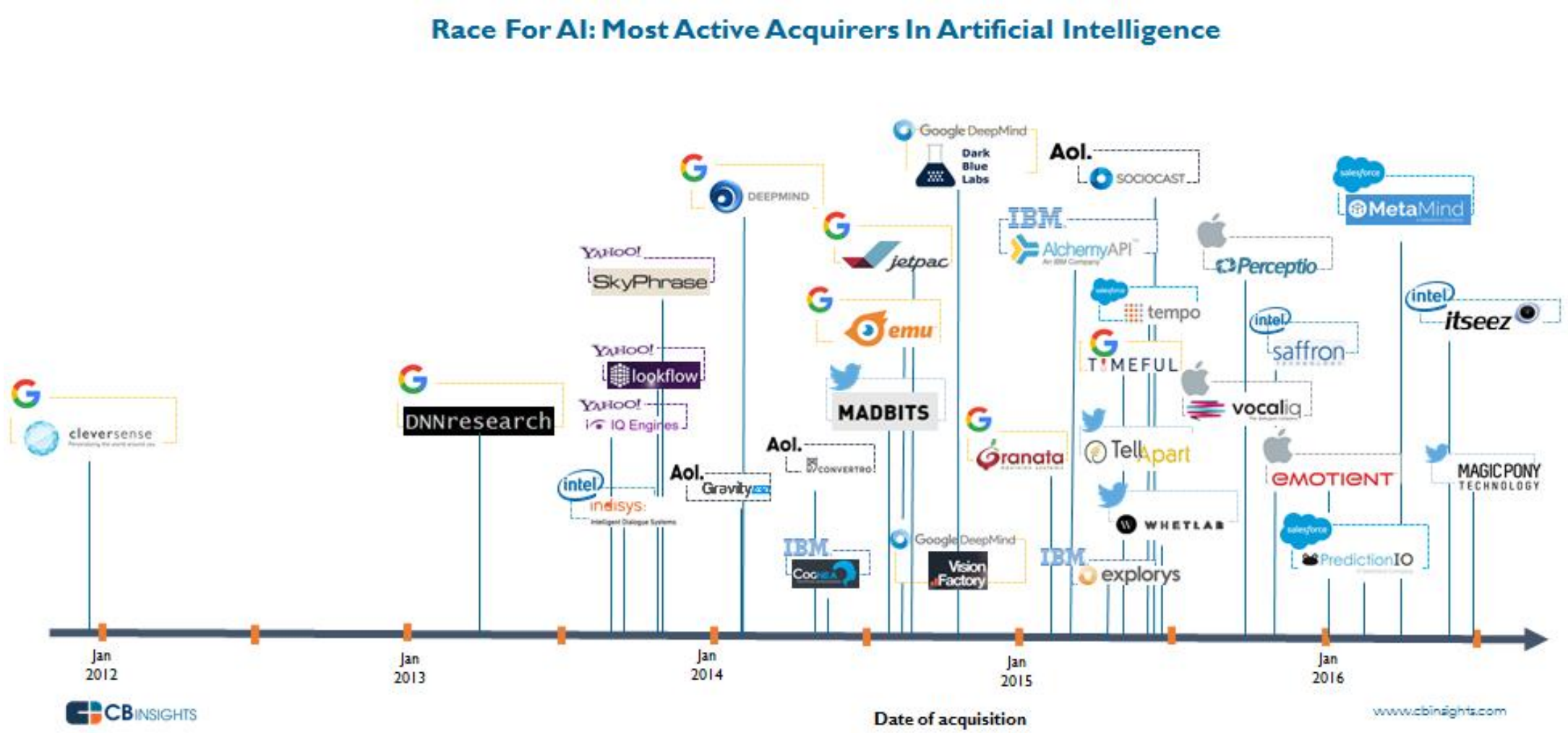
Artificial Intelligence: Most Active Corporate Investors

2011-2016YTD (as of 6/15/2016)

Investor	Rank	Select Investments
Intel Capital	1	DataRobot, prelert, lumiata, MAANA, incoming, saffron, PERFANT, EMOTIENT, Reflektion, Parallel Machines, MindMeld, smartzip, api.ai, indisys, COLDLIGHT
Google Ventures	2	BUILDING ROBOTICS, clarifai, KENSHO, FRAMED, ZEPHYR HEALTH, Unbabel, tamr, MindMeld, Orbital Insight, Predilytics
GE Ventures	3	SIGHT MACHINE, ARTERYS, AYASDI, BITSTEW SYSTEMS, MedAware, PingThings, stem, Predixion, MAANA
Samsung Ventures	4	vicarious, sentiance, MaluubA, iDIBON, jibo, MindMeld, AUTOMATED INSIGHTS
Bloomberg Beta	4	deep genomics, context relevant, AVISO, howdy., DOMINO, Orbital Insight, DigitalGenius, DIFFBOT
In-Q-Tel	6	MindMeld, CYLANCE, celect, INTERSET, Digital Reasoning, DOMINO
Tencent	7	DIFFBOT, CLOUDMEDX, SI, SCALED INFERENCE, iCarbonX, skymind
Nokia Growth Partners	8	rocketfuel, WorkFusion, rapidminer, indix.
Microsoft Ventures	8	BUILDING ROBOTICS, NEURA, insidesales, CrowdFlower
Qualcomm Ventures	8	clarifai, Predilytics, Welltok., tempo
Salesforce Ventures	8	DigitalGenius, insidesales, 6sense
AXA Strategic Ventures	8	NEURA, BEATS, medlanes, pricemethod
New York Life Insurance Company	8	context relevant, DataRobot, Skycure, captricity

Source : CB Insights

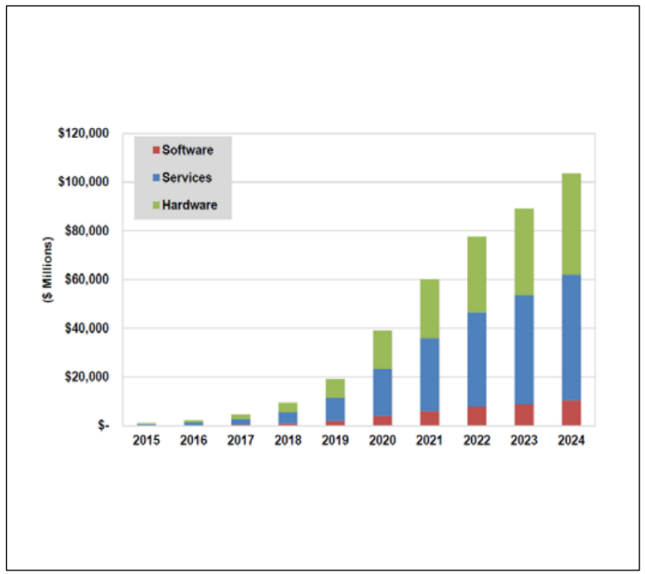
Tech Giants are Snapping Up AI



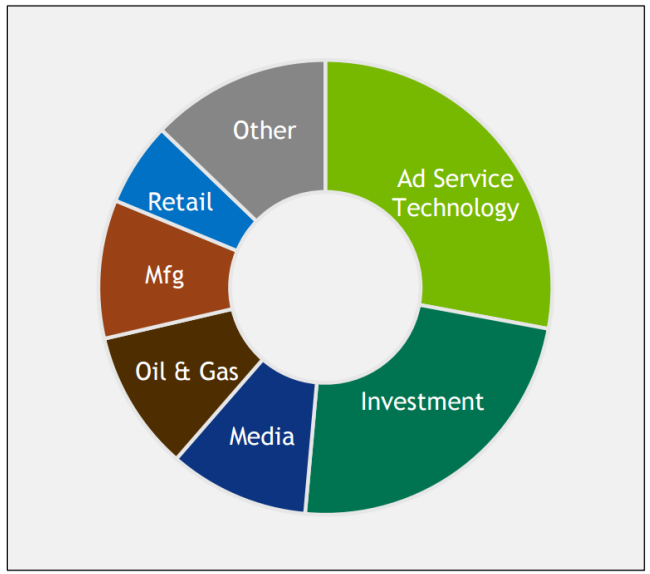
Source : CB Insights

Huge Market Opportunity

$500B OPPORTUNITY OVER 10 YRS



Deep Learning Total Revenue by Segment

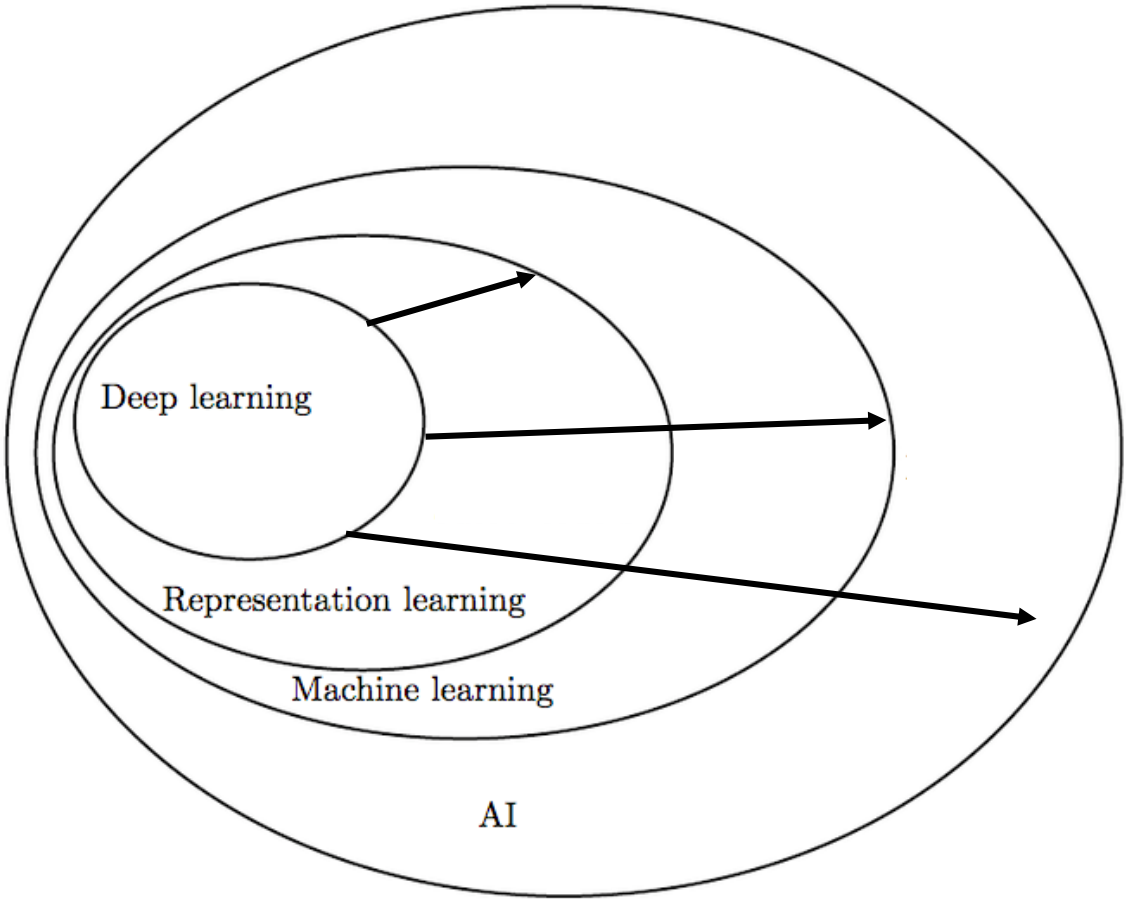


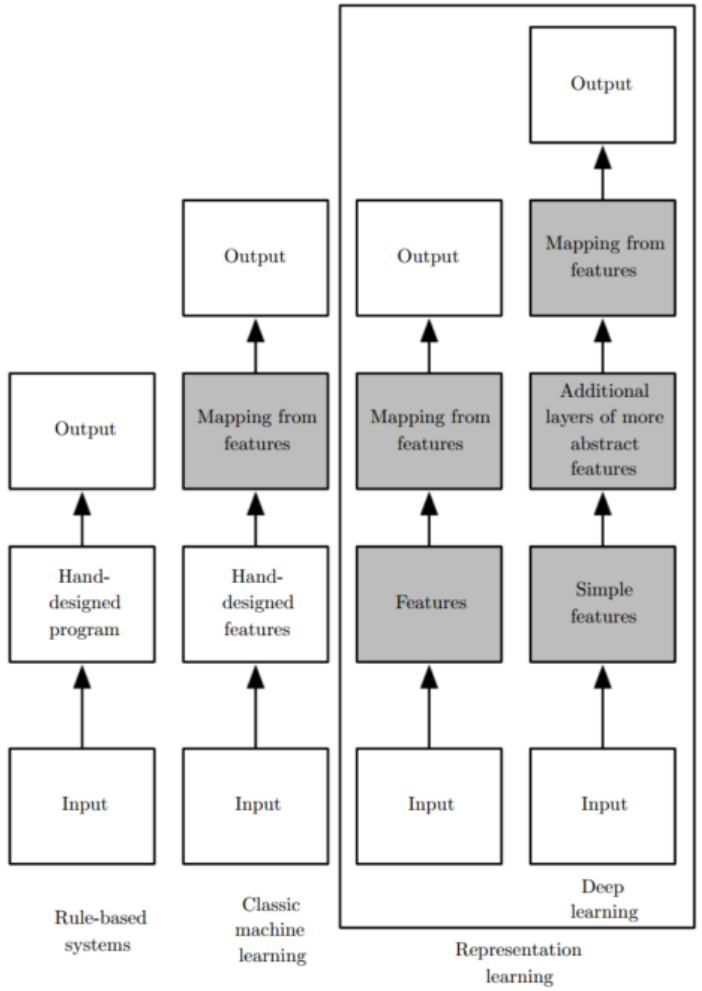
Deep Learning Software Revenue by Industry

IBM: "Cognitive business represents a $2T opportunity"

SOURCE: "Deep Learning for Enterprise Applications," 4Q 2015, Tractica

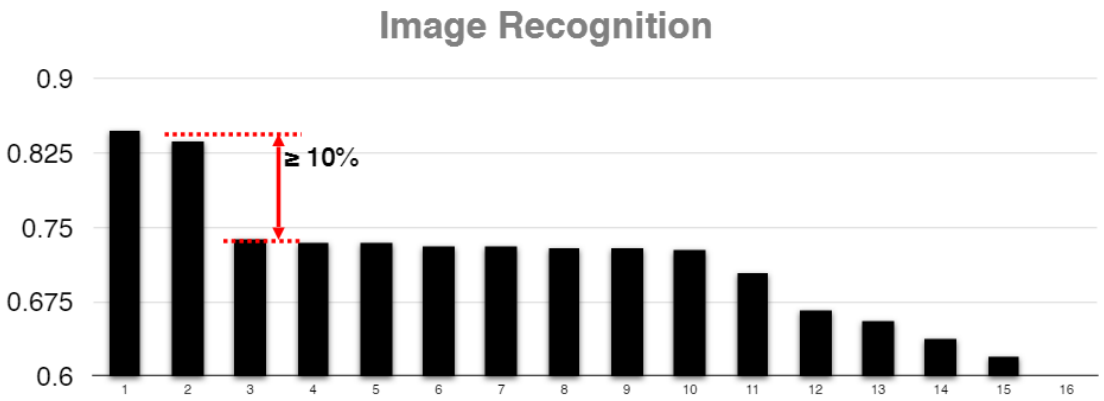
Deep Learning as the Core of 'AI' Engine



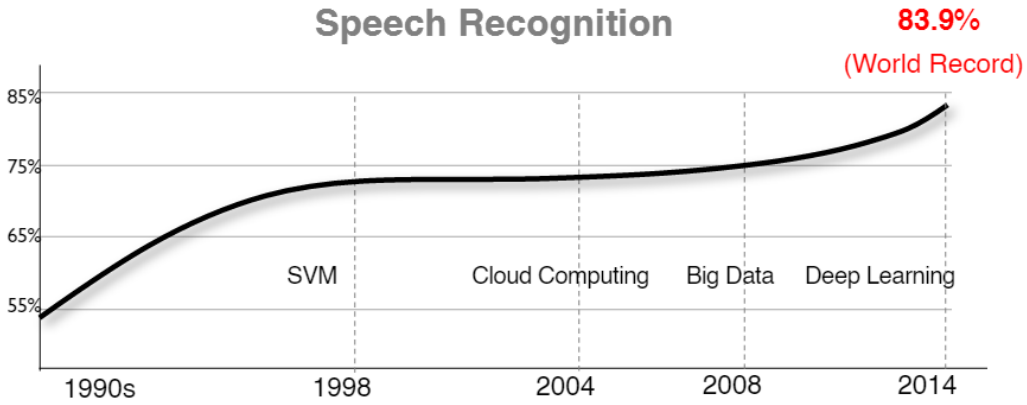


"Deep Learning", Ian Goodfellow et al. 2016

Deep Learning Make Machines to have 'Perception'



ILSVRC-2012 Task 1 (Classification) Result



rank	method	accuracy (%)
1	Deep RNN	82.3
2	DNN	79.3
3	Augmented CRF	77
4	MLPHMM	74.2

Phoneme Recognition Accuracy (TIMIT Data)

Now, Machines Learn Deep

- Now, Machines Beat Human in Tasks Once Considered Impossible

AlphaGo

VS

5:0
vs Fan Hui
(Oct. 2015)

4:1
vs Sedol Lee
(Mar. 2016)

Explaining Deep Learning

You could think of Deep Learning as the building of learning machines, say pattern recognition systems or whatever, by assembling lots of modules or elements that all train the same way.

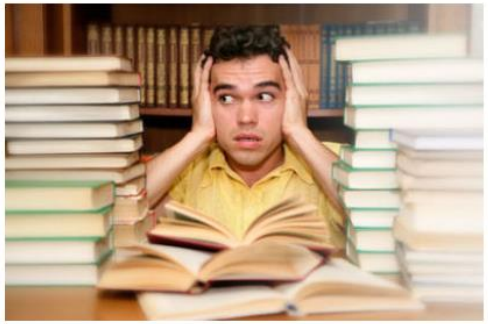
So there is a **Single Principle** to train everything.

IEEE Spectrum, Feb. 2015

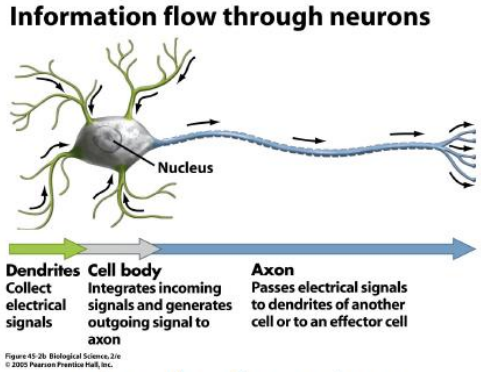
WIKIPEDIA
The Free Encyclopedia

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 linear and non-linear transformations.

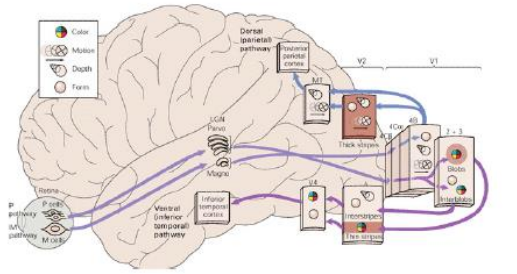
Brain-inspired Learning



Learn massive data

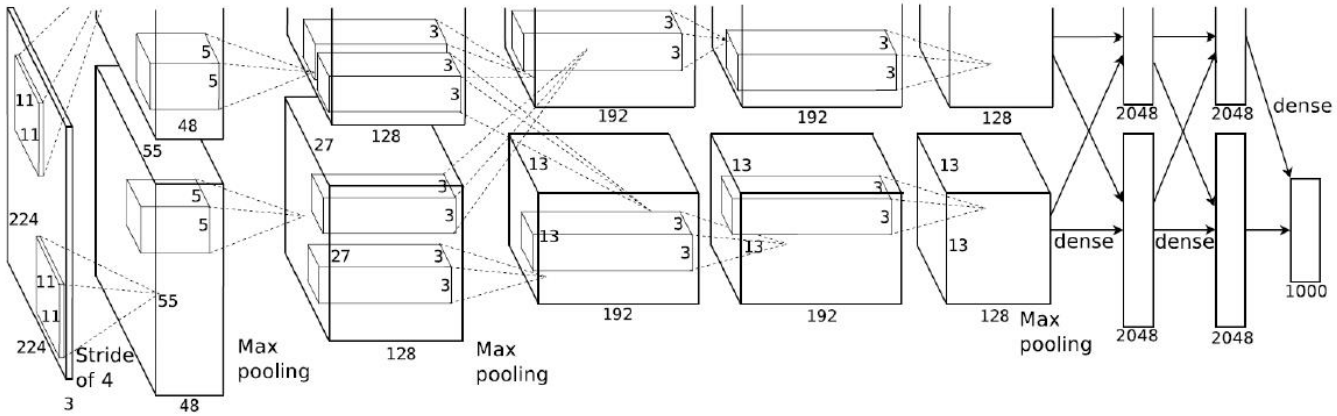


simple functions

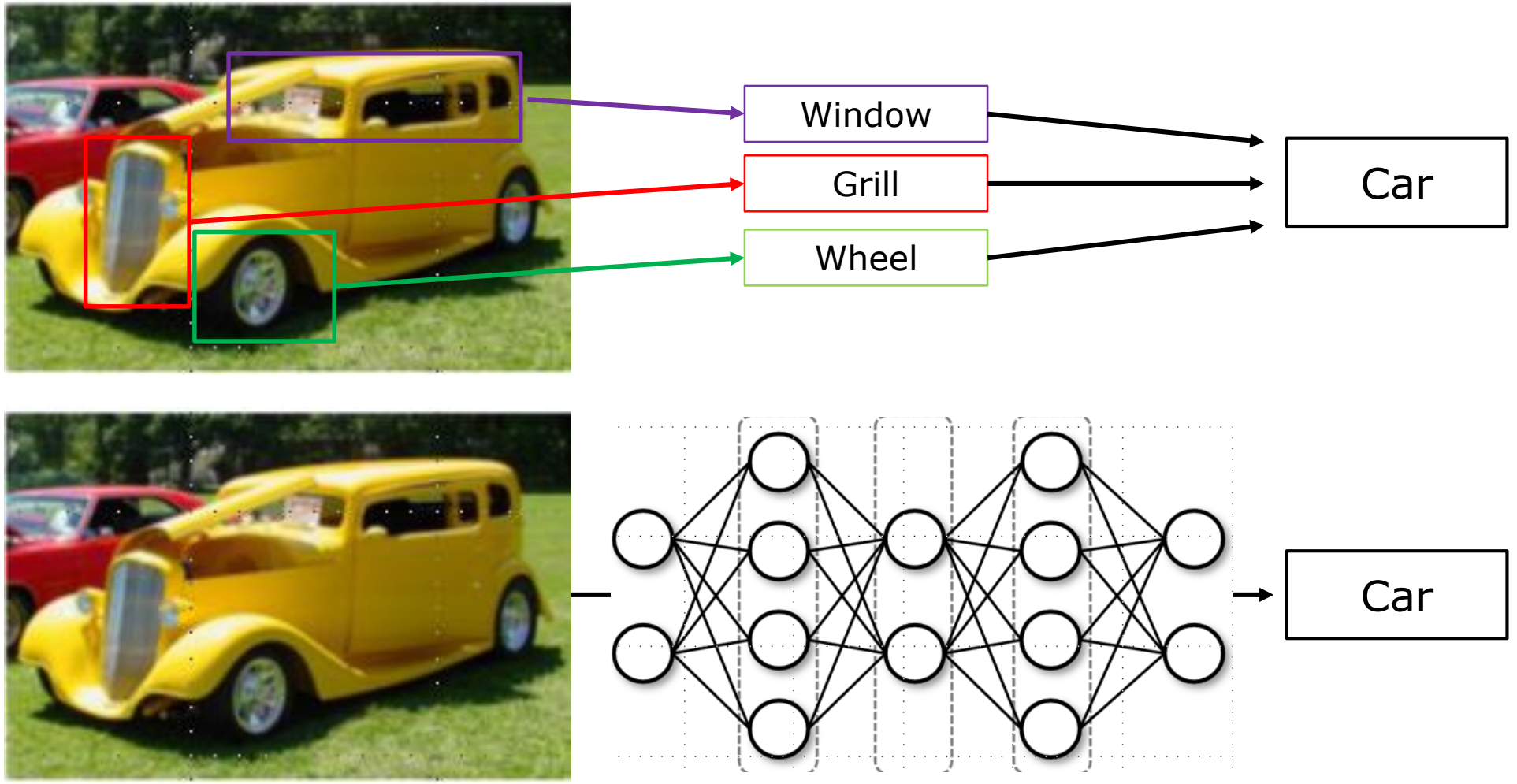


(Van Essen&Gallant, 1994)

Multi-layered

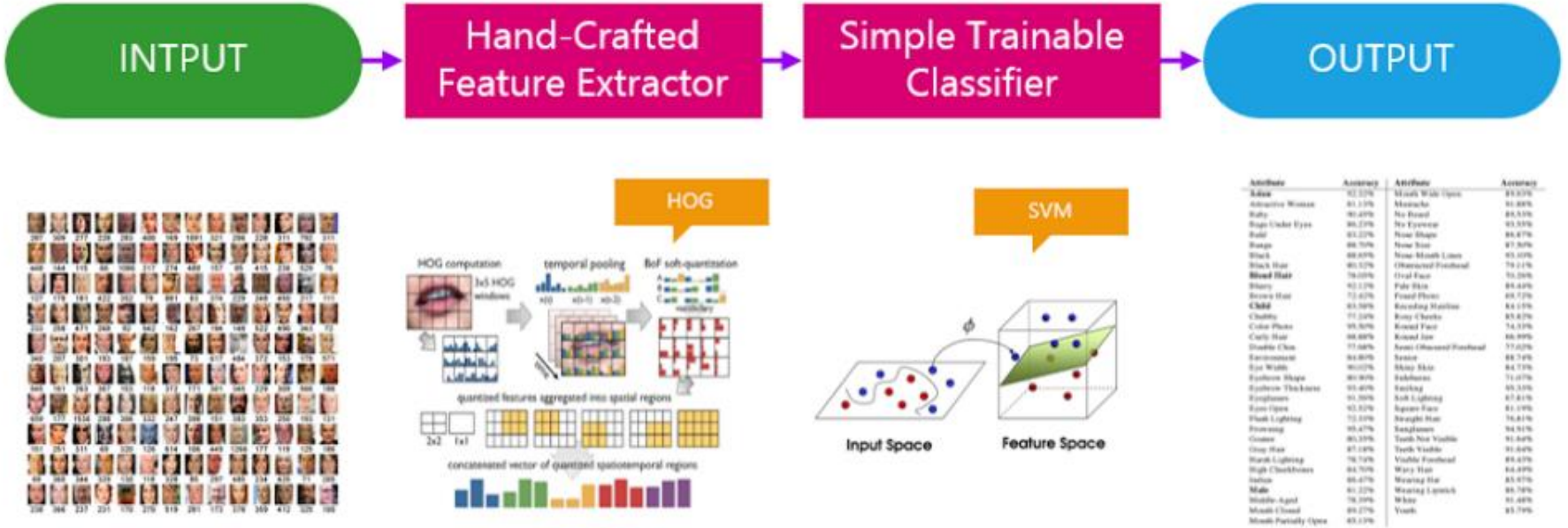


Deep Learning as a Feature Learning



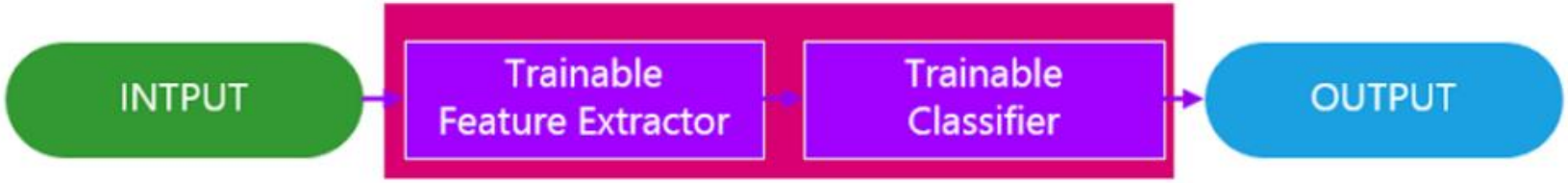
Traditional Machine Learning vs Deep Learning

Traditional Machine Learning

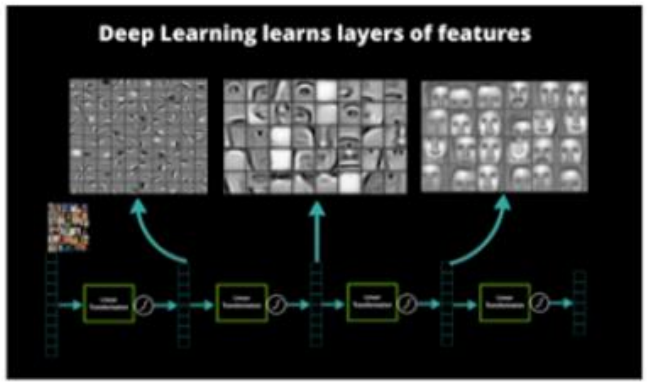


Traditional Machine Learning vs Deep Learning

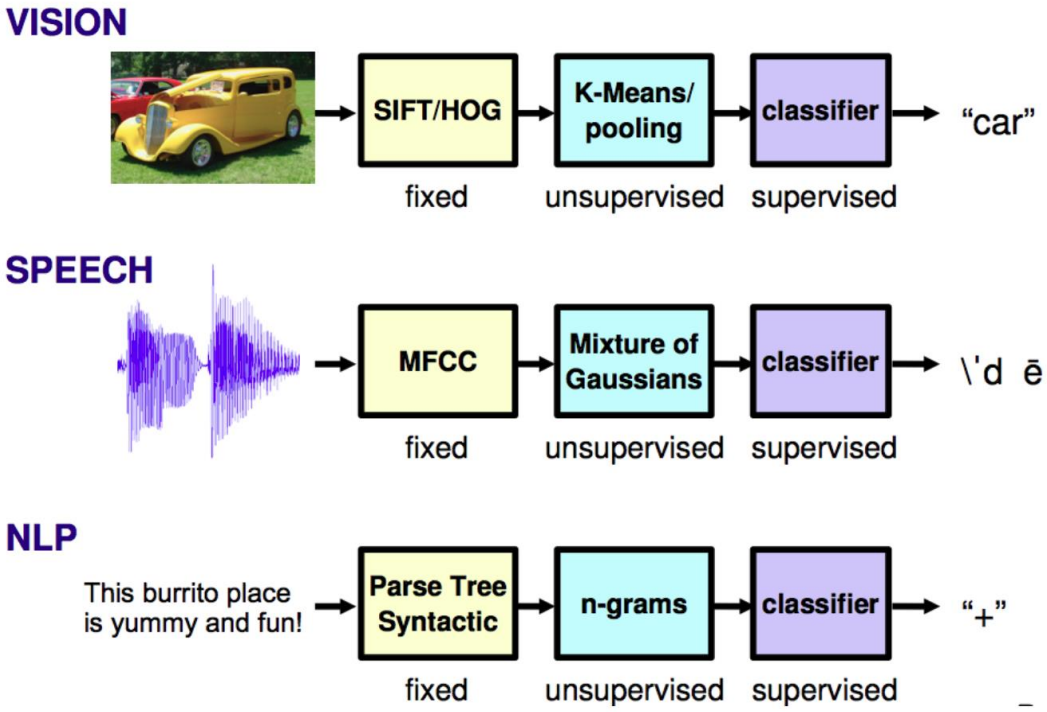
Deep Learning

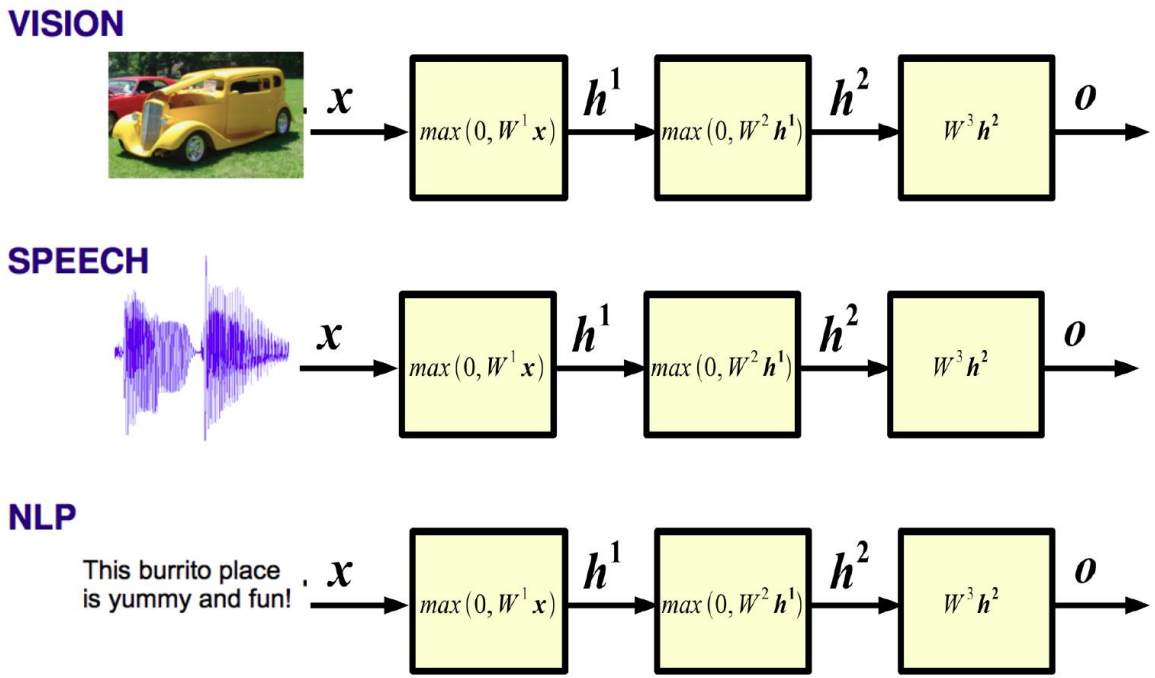


END-TO-END



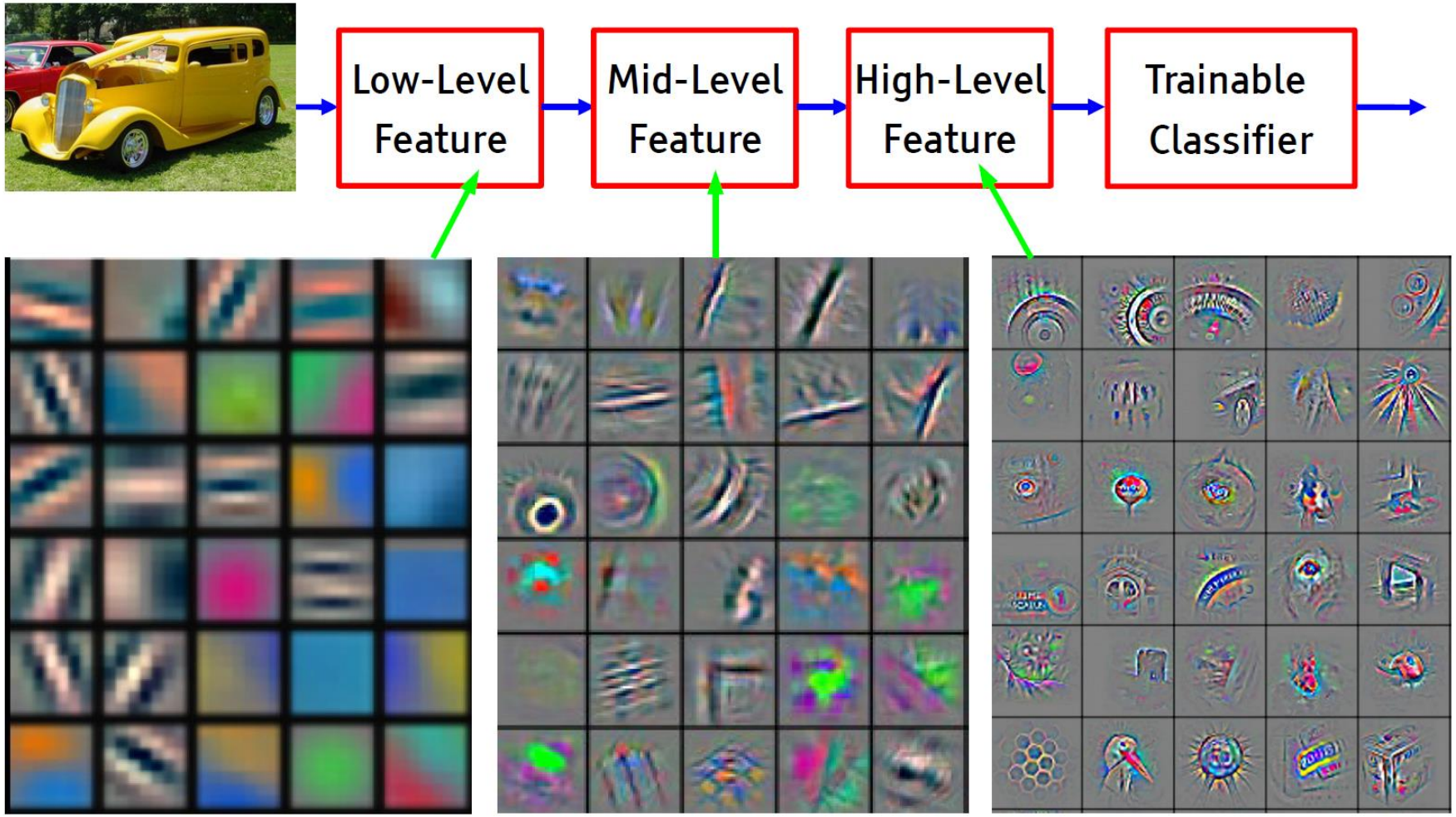
Feature Engineering vs Feature Learning





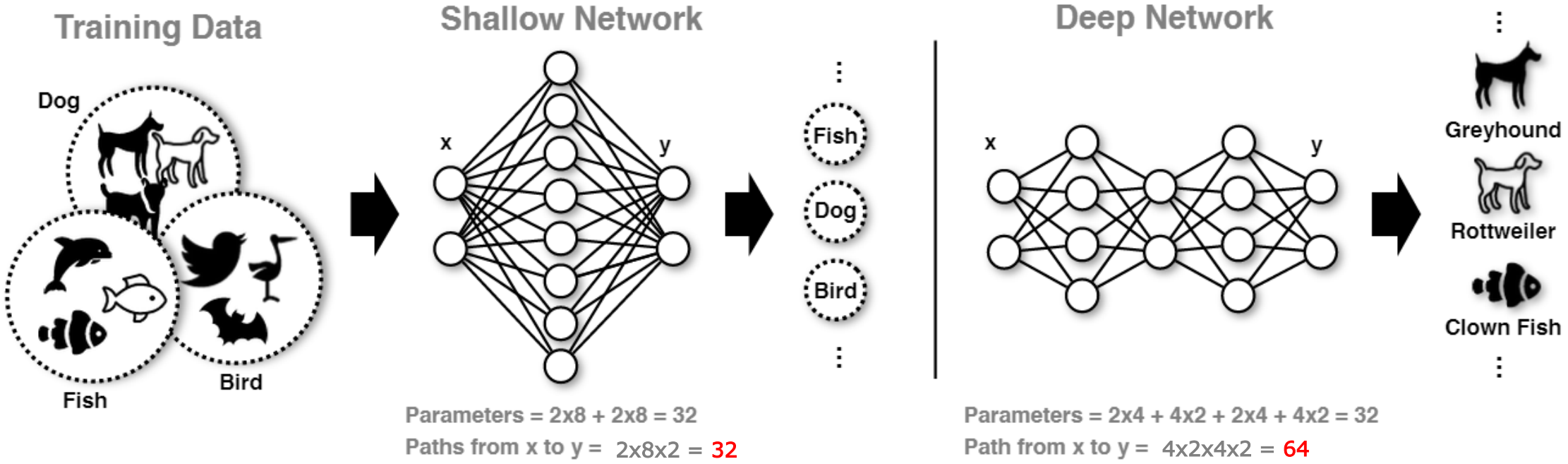
From Yann LeCun

Hierarchical Learned Representation



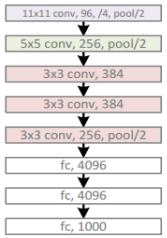
From Yann LeCun and Zeiler(2013)

Why Depth Matters?

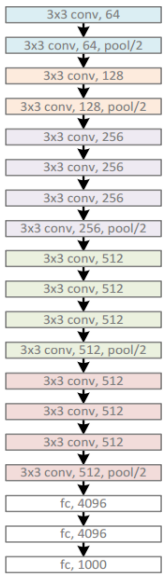


How Deep is Deep?

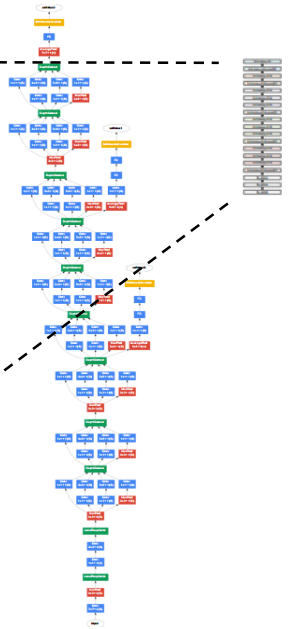
AlexNet, 8 layers
(ILSVRC 2012)



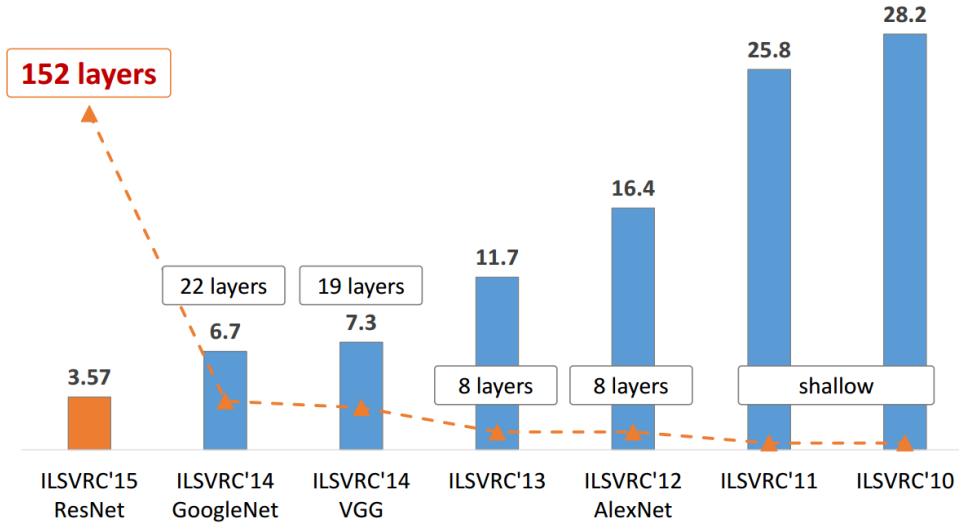
VGG, 19 layers
(ILSVRC 2014)



GoogleNet, 22 layers
(ILSVRC 2014)



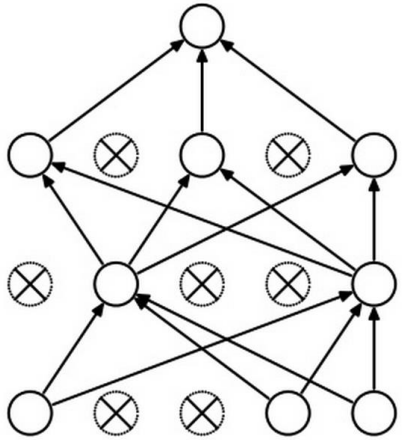
ResNet, 152 layers
(ILSVRC 2015)



Why Now?

Big Data

Computational Power

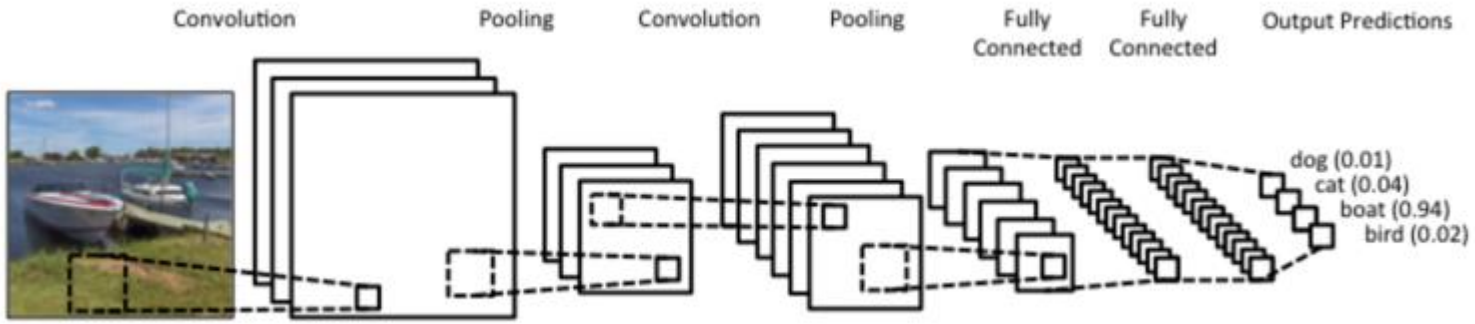


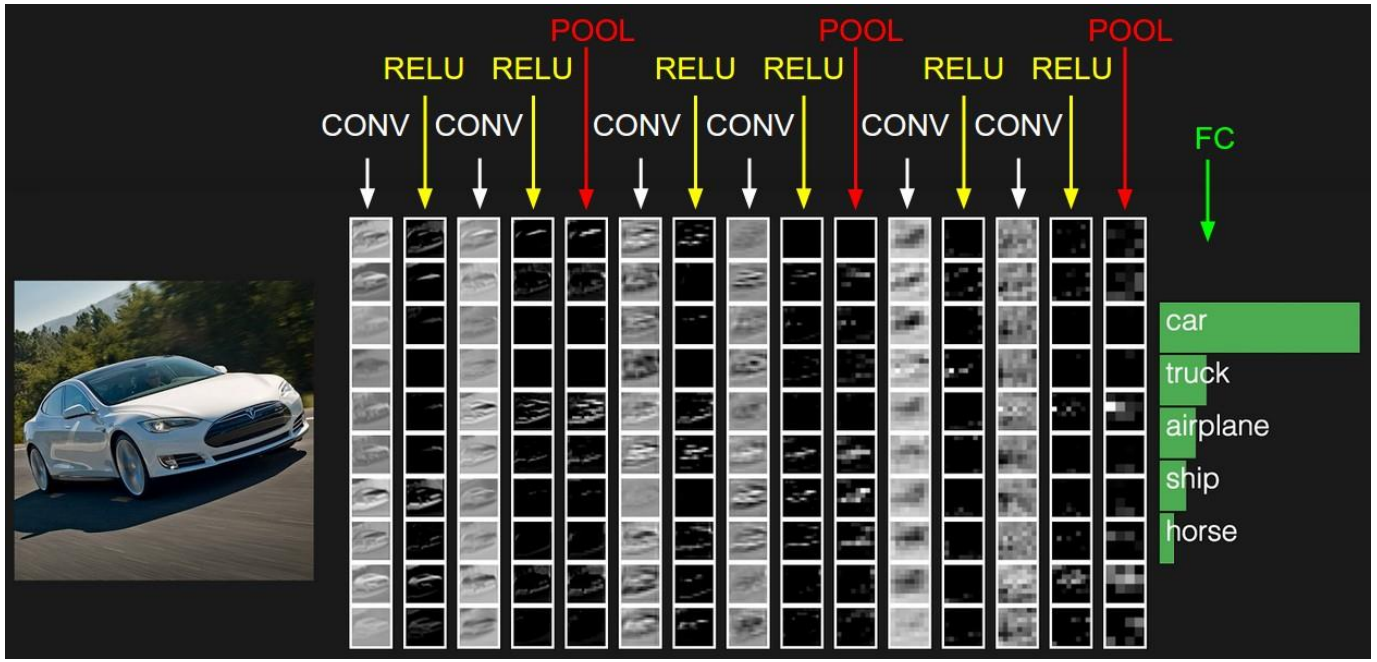
Algorithm

Deep Learning for Visual Recognition

Basics for Convolutional Neural Networks

Convolutional Neural Network (CNN)





Elements of Convolutional Neural Network

- Local Connectivity
- Parameter Sharing

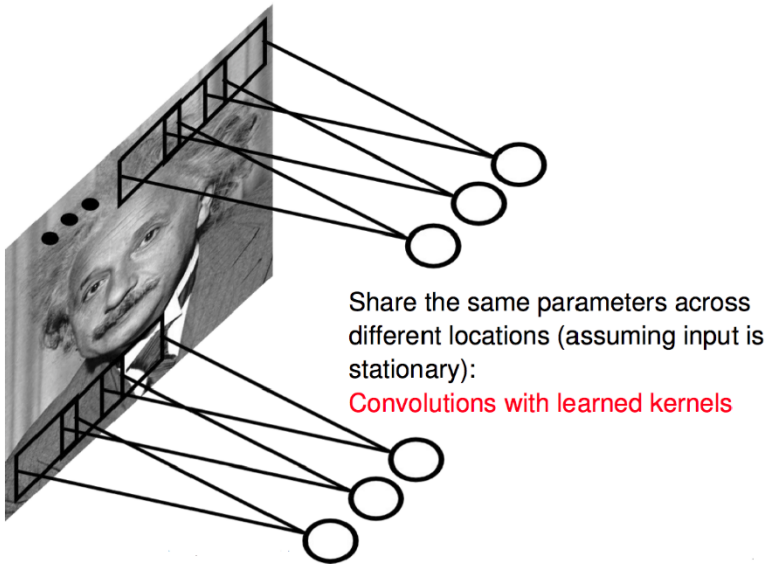
→ Convolution Layer

- Pooling/Subsampling

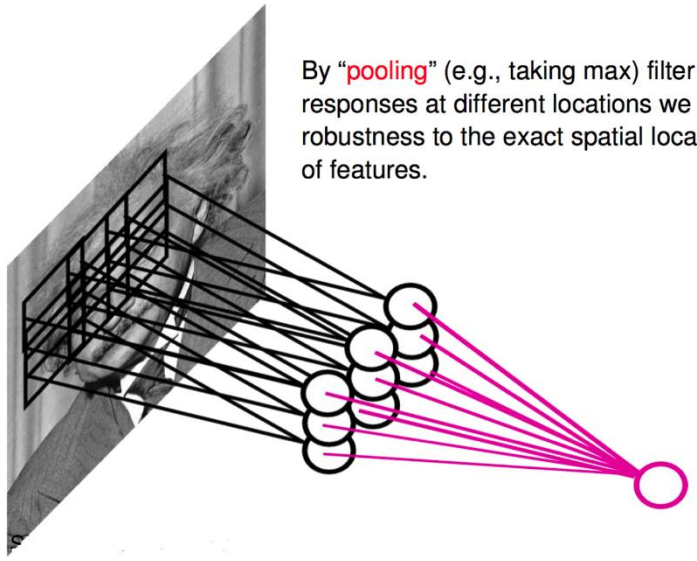
→ Pooling Layer

- Nonlinearity

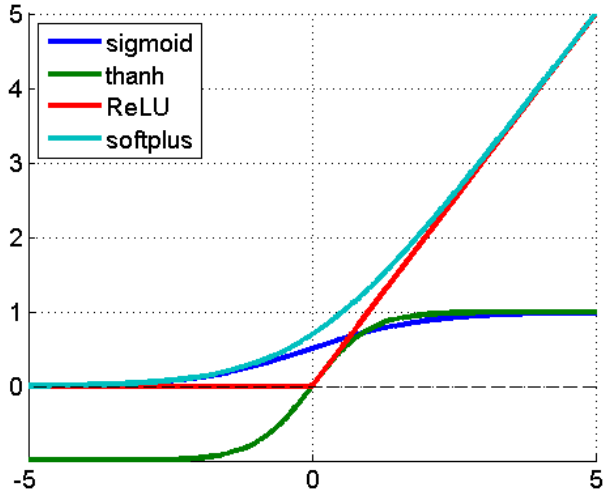
→ Activation Function



Convolution



Pooling



Activation Function

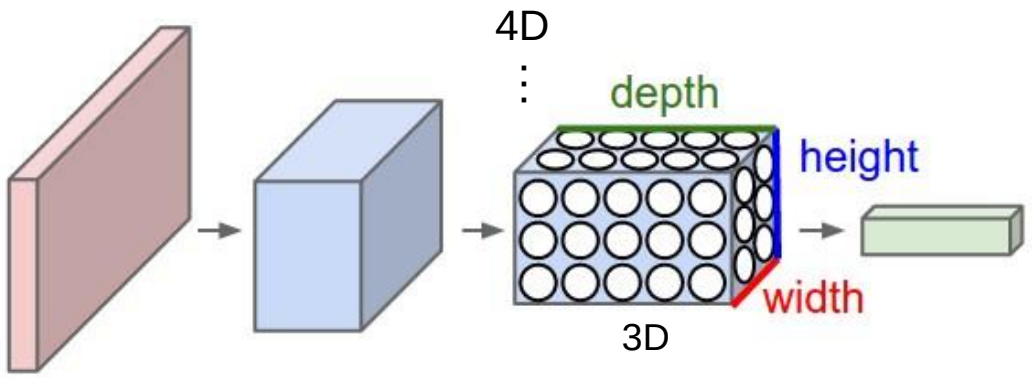
Data Structure

- 4D Tensor
  - (Sample, Channel, Height, Width)
  - Height and width for feature map size
  - Channel for number of feature map(or number of filter)
  - Sample is mini-batch size

cuDNN Example :
cudnnSetTensor4dDescriptor(outputDesc,CUDNN_TENSOR_NCHW,
CUDNN_FLOAT,sampleCnt,channels,height,width)

4D Tensor (2, 2, 3, 3)

Sample 1			Sample 2			
1	2	3	19	20	21	Channel 1
4	5	6	22	23	24	
7	8	9	25	26	27	
10	11	12	28	29	30	Channel 2
13	14	15	31	32	33	
16	17	18	34	35	36	

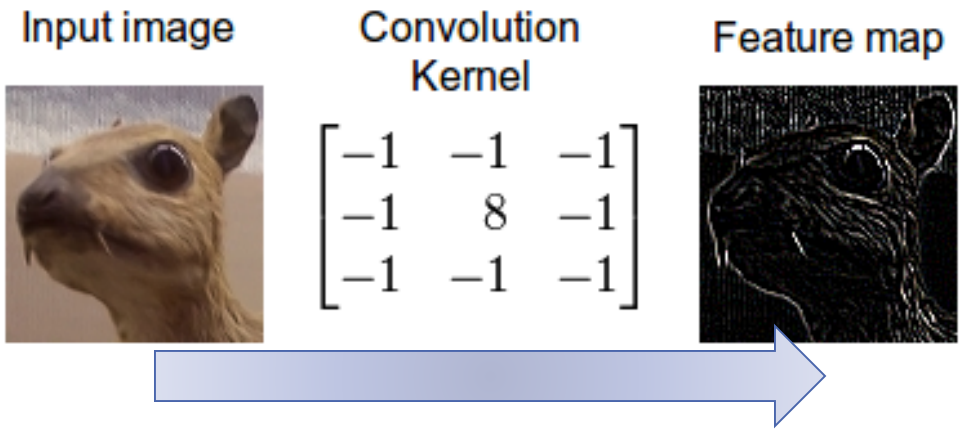


Convolution

- Basic Usage of Convolution
  - To remove unimportant information from the image : Filter
  - To capture necessary information for given task : Feature
  - To map input image patch of same size into single number : Kernel

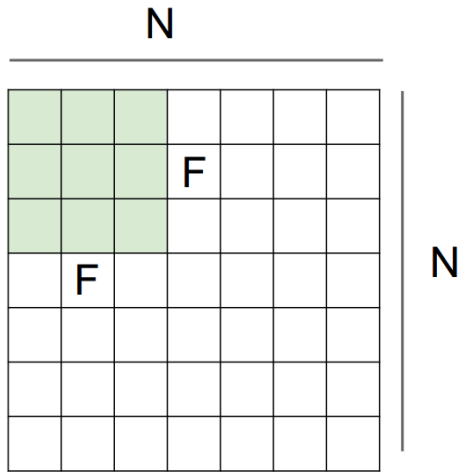
  - Choosing right convolution kernel is crucial for the success of analysis

    => How can we choose the best kernel?



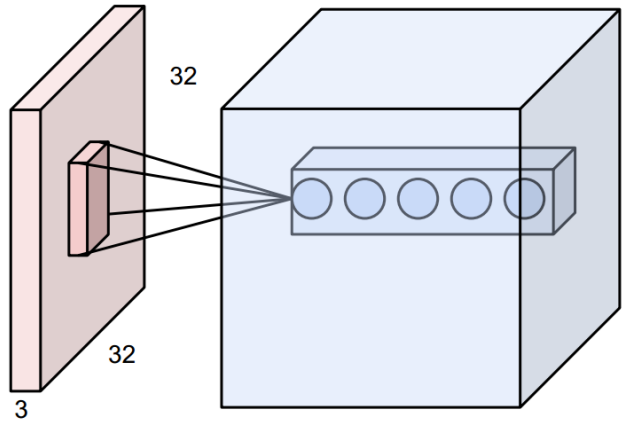
Convolution

- Local Connectivity
  - Multiple neurons all looking at the same region(receptive field) of the input volume
    stacked along the depth
  - The size of receptive field for each dimension, stride and padding size and method are all hyper-parameters.



Output size:
(N - F) / stride + 1

e.g. N = 7, F = 3:
stride 1 => (7 - 3)/1 + 1 = 5
stride 2 => (7 - 3)/2 + 1 = 3
stride 3 => (7 - 3)/3 + 1 = … :\

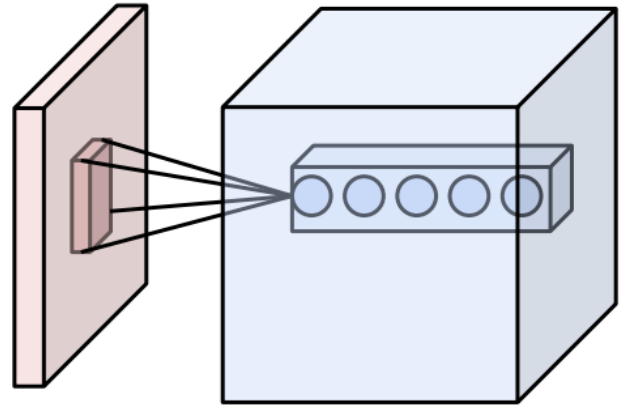


Convolution

- Need for Weight Sharing

Input volume: **32x32x3**
Receptive fields: **5x5**, **stride 1**
Number of neurons: **30**

Output volume: (32 - 5) / 1 + 1 = 28, so: **28x28x 30**
How many weights for each of the 28x28x 30 neurons? **5x5x3 = 75**



Before:
#weights in such layer: (32*32*30) * 75 = **3 million :**

Now: (paramater sharing)
#weights in the layer: 30 * 75 = **2250**.

Example trained weights

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

Convolution

- Zero-padding
  - Adding zero-valued(most common) pixels surrounding the input feature map.
  - Provide some useful characteristics as below :

“Same convolution” (preserves size)

Input [9x9]

3x3 neurons, stride 1, pad **1** => [9x9]
3x3 neurons, stride 1, pad **1** => [9x9]

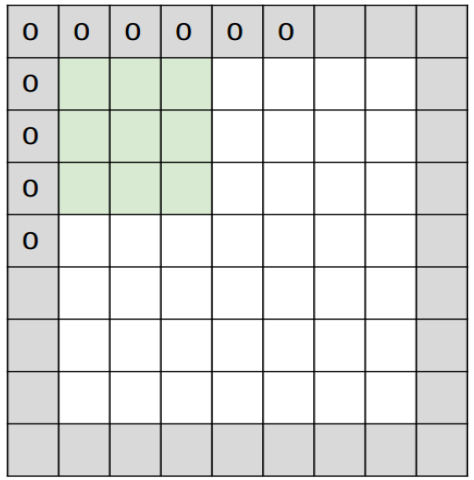
- No headaches when sizing architectures
- Works well

“Valid convolution” (shrinks size)

Input [9x9]

3x3 neurons, stride 1, pad **0** => [7x7]
3x3 neurons, stride 1, pad **0** => [5x5]

- **Headaches** with sizing the full architecture
- **Works Worse!** Border information will “wash away”, since those values are only used once in the forward function



Convolution vs Cross-correlation

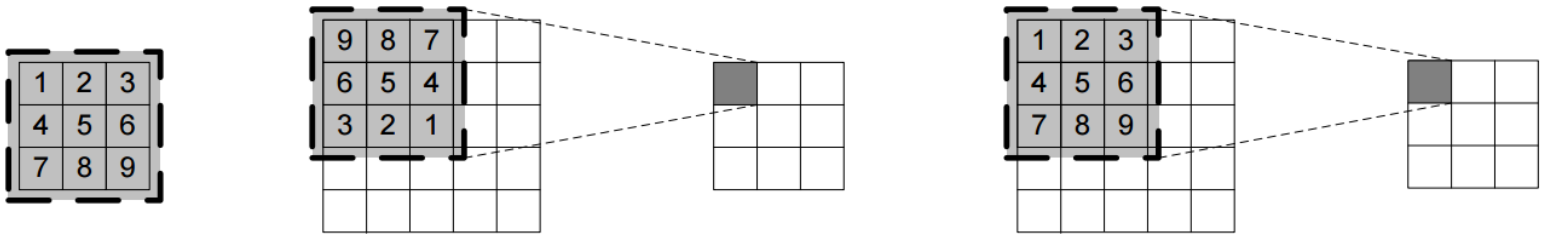
- In practice, we use cross-correlation instead of 2d convolution operation.
  - Because the filter weights are randomly initialized and learned, both convolution and cross-correlation results in similar result.
  - To perform convolution operation, we need to flip the weights in both dimension or rotate the matrix 180 degree.
  - Using cross-correlation is easy to debug and faster

cudnnConvolutionMode_t

Value	Meaning
CUDNN_CONVOLUTION	In this mode, a convolution operation will be done when applying the filter to the images.
CUDNN_CROSS_CORRELATION	In this mode, a cross-correlation operation will be done when applying the filter to the images.

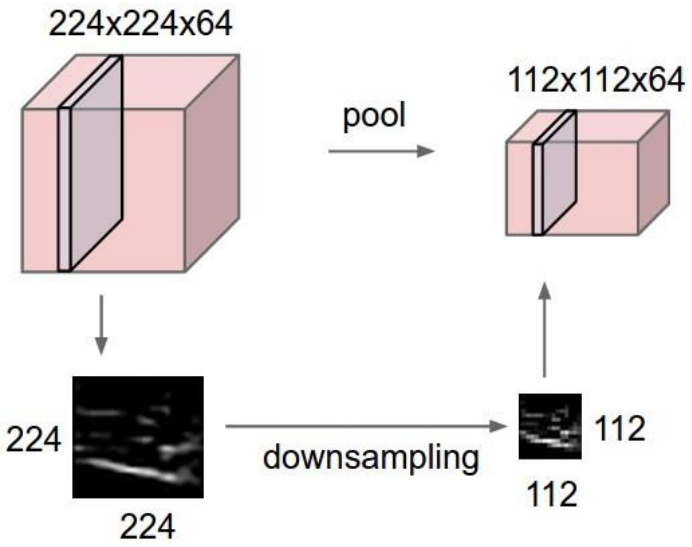
$$Y(x,y) = \sum_{u=0}^{K_x} \sum_{v=0}^{K_y} X(x-u, y-v) w(u,v)$$

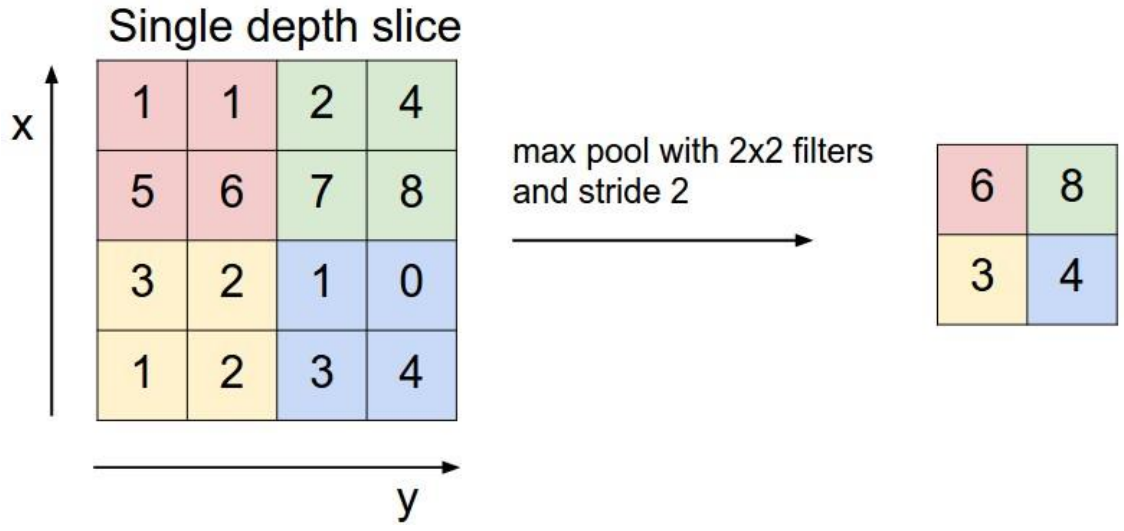
$$Y(x,y) = \sum_{u=0}^{K_x} \sum_{v=0}^{K_y} X(x+u, y+v) w(u,v)$$



Pooling

- Pooling or Subsampling Feature Map
  - Introduces invariance to local translations
  - Reduces the number of hidden units and enlarge the receptive field
  - Max(most popular) or average pooling
  - Need to save the 'switch' index for backpropagation

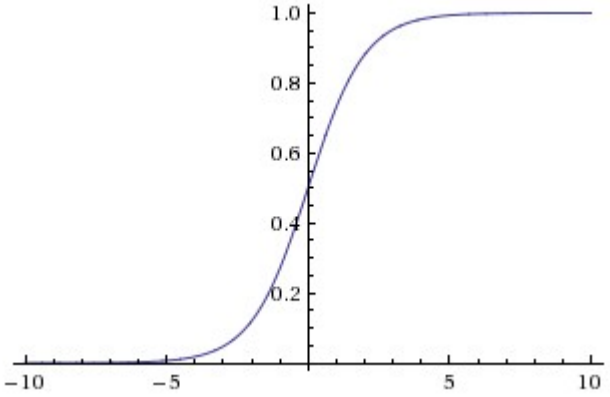




Basic Activation Functions

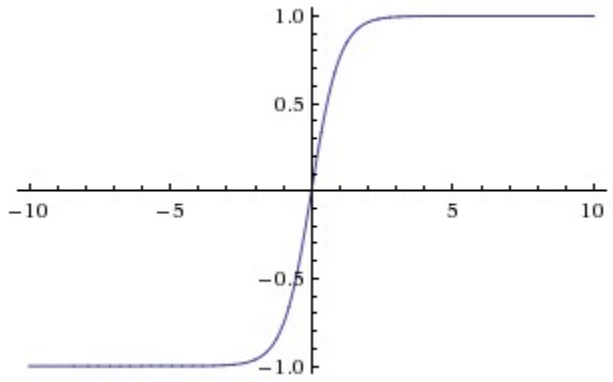
- Activation Function
  - Differentiable nonlinear function which controls the firing rate of neurons.
  - Squashes real-value input to the output of specific range.
  - The output range and gradient of activation function have significant impact to training neural network

Sigmoid



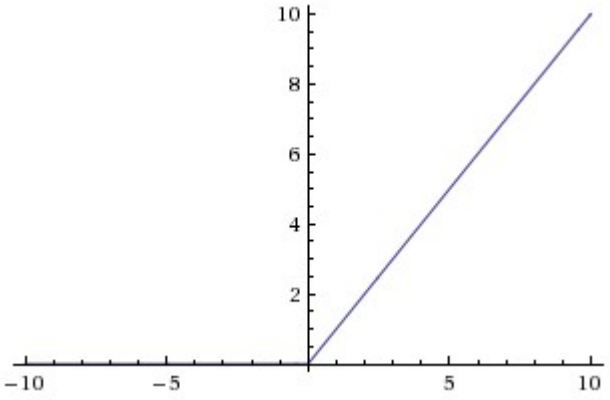
$$g(z) = \frac{1}{1 + e^{-z}}$$

Tanh



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

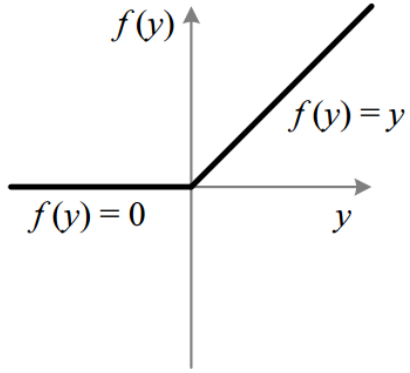
Rectified Linear Unit

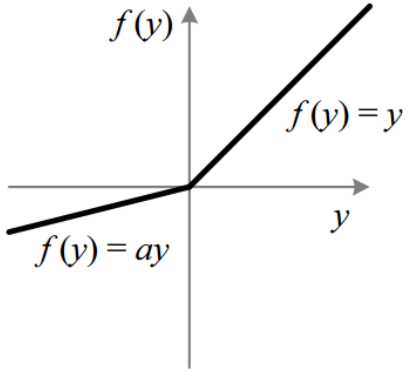


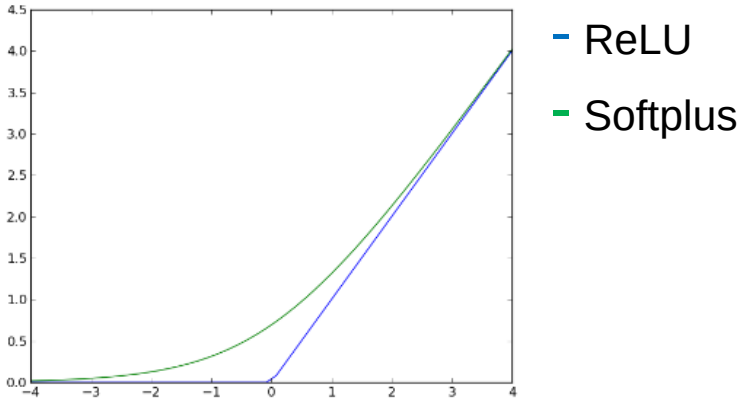
$$g(z) = \max(0, z)$$

Other Activation Functions

- To reduce 'dying' neurons
  - Leaky ReLU : Allow fixed small slope in the negative input range
  - Parametric ReLU : Allow 'parameterized' slope in the negative input range
  - Softplus : Smooth function which is approximately linear function in the positive input range and allow some gradient in the negative input range







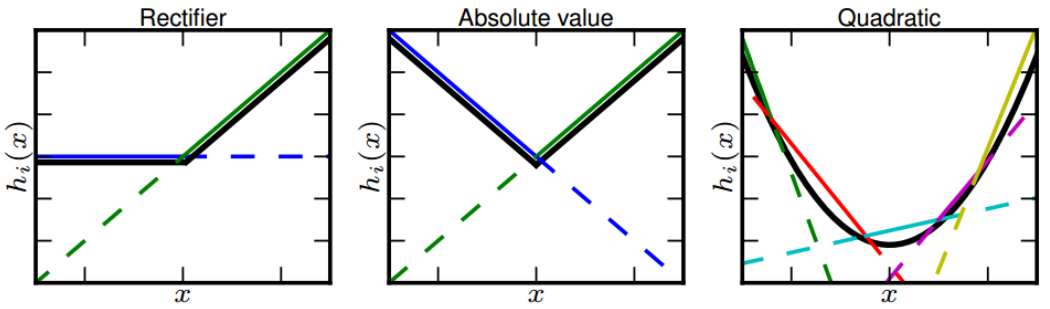
$$g(z) = \ln(1 + e^z)$$

More Complex Activation Function

- Maxout Network(I. Goodfellow, 2013)
  - Generalization of ReLU
  - Take maximum of set of piecewise linear function
  - Can approximate any convex function

$$h_i(x) = \max_{j \in [1,k]} z_{ij}$$

$$z_{ij} = x^T W_{\ldots ij} + b_{ij}$$

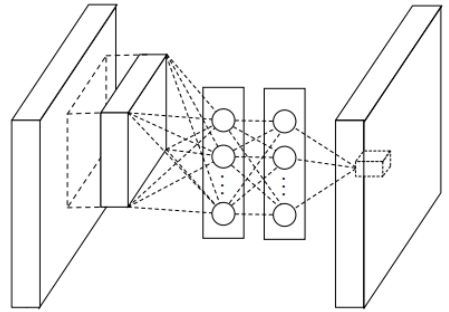


- Network-in-Network(M. Lin, 2013)
  - Universal approximator(MLP) as an activation function

$$f^1_{i,j,k_1} = \max({w^1_{k_1}}^T x_{i,j} + b_{k_1}, 0).$$

$$\vdots$$

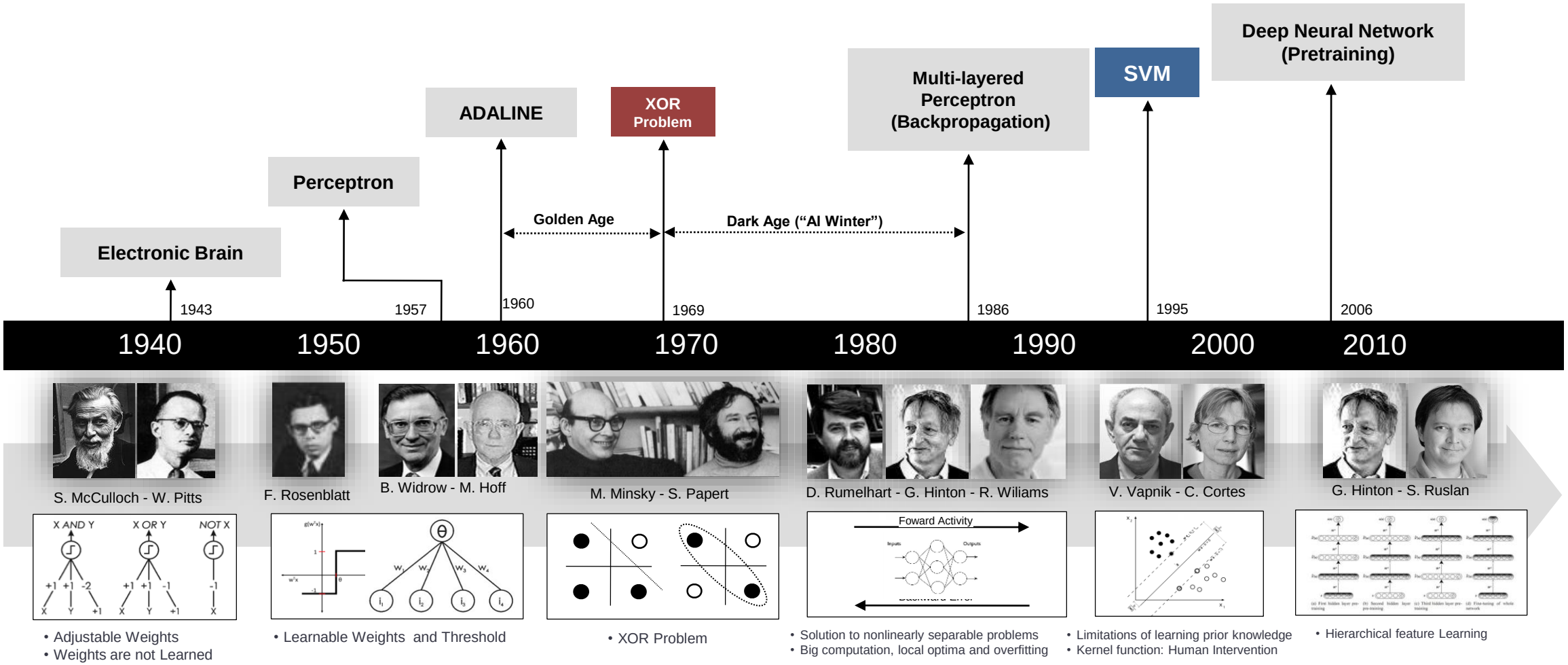
$$f^n_{i,j,k_n} = \max({w^n_{k_n}}^T f^{n-1}_{i,j} + b_{k_n}, 0).$$



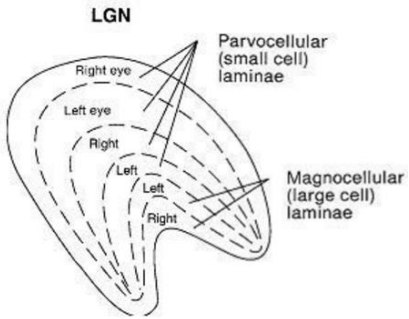
mlpconv

History of Convolutional Neural Networks

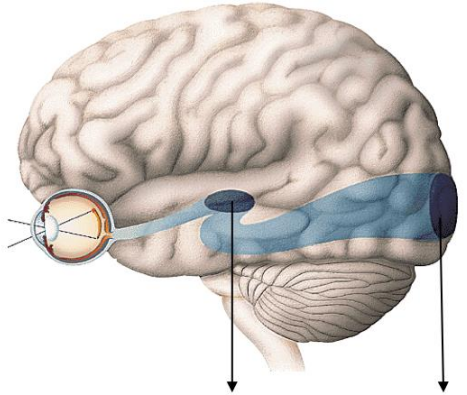
Neural Network Timeline



Receptive Field

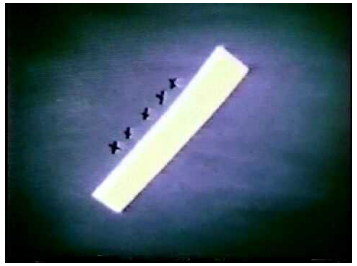


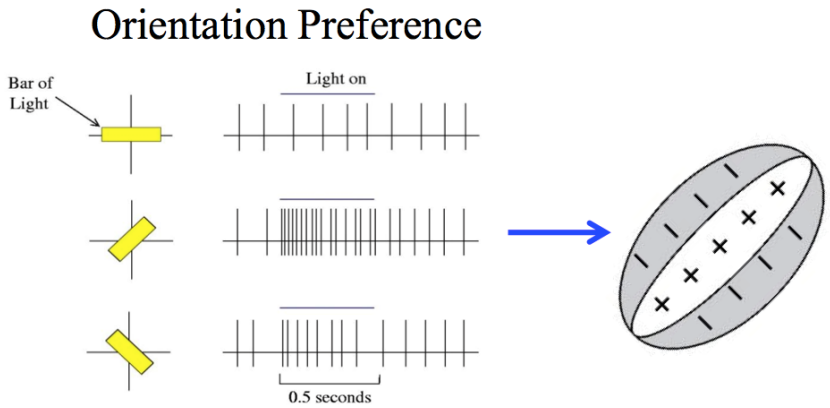
- Hubel and Wiesel(1959, 1962)
  - Receptive Field
    - Particular region of sensory space that will trigger the activation of corresponding neuron.
  - Simple Cell and Complex Cell
    - Simple Cell : Neurons that respond to stimuli with specific angle.
    - Complex Cell : Neurons that respond to stimuli regardless the position of stimuli



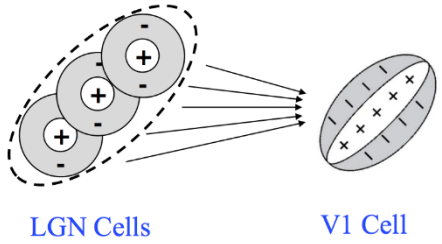
LGN (Lateral Geniculate Nucleus)

V1 (Primary Visual Cortex)



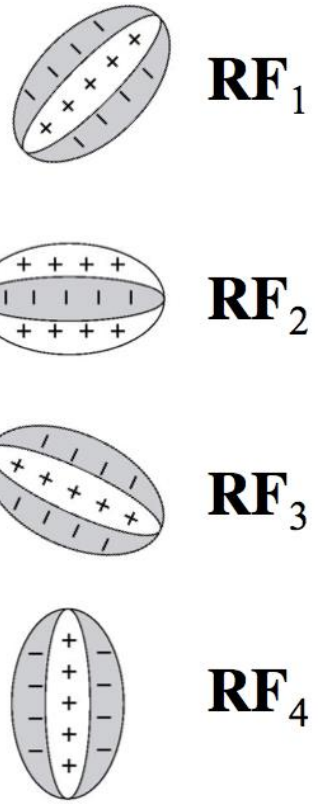


Oriented receptive field of a neuron in primary visual cortex (V1)



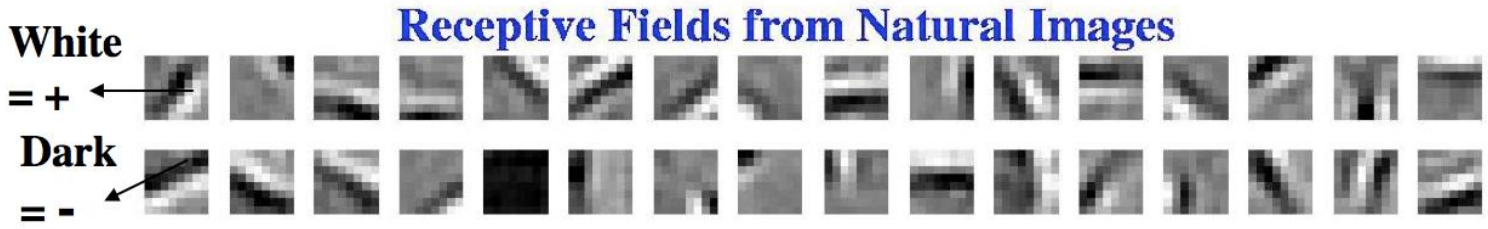
Efficient Coding Hypothesis

- How do we represent natural images using these RFs?
  - Linearly combine the responses from RFs.
  - The combination is the 'code' of image that represent it.
  - One efficient coding method algorithm is 'sparse coding' (Olshausen and Field, *Nature*, 1997)



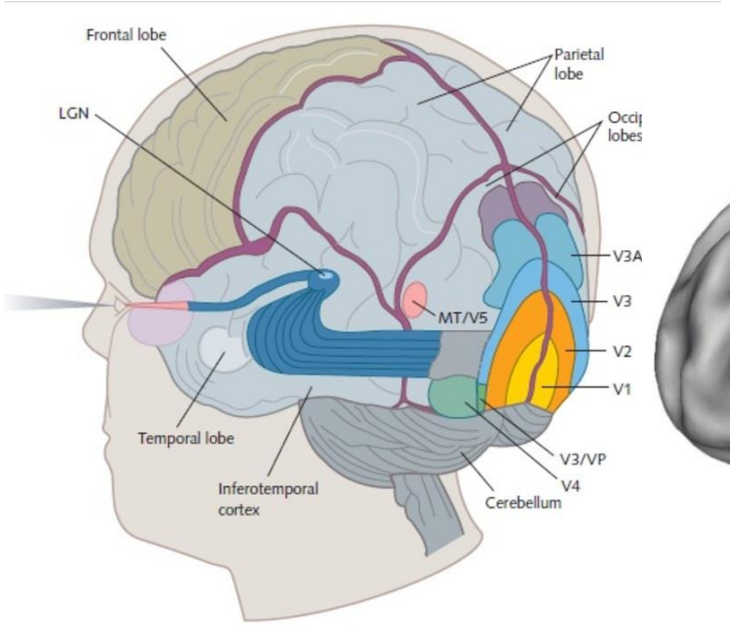
Given image **I**, we can reconstruct **I** using neural responses $r_1, r_2 \ldots$:

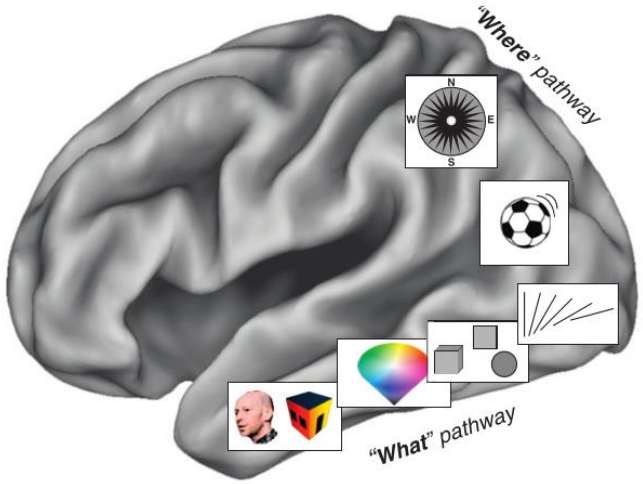
$$\hat{\mathbf{I}} = \sum_i \mathbf{RF}_i r_i$$

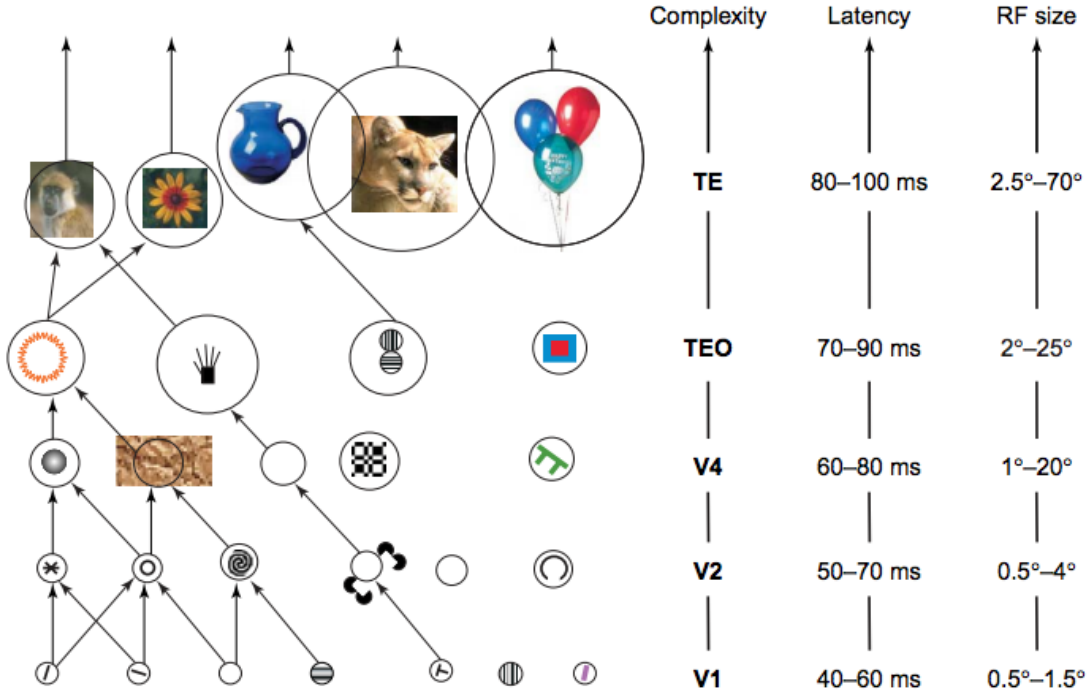


Beyond V1

- Gradual More Complex Preferred Stimulus
  - The receptive fields are extended by complex cells
  - The stimulus that active neurons are higher concepts

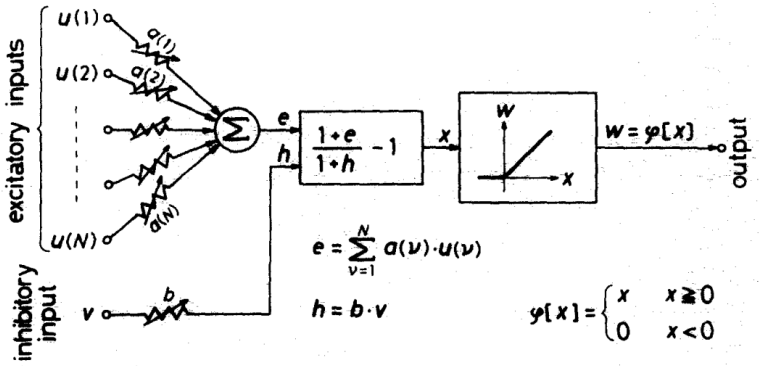


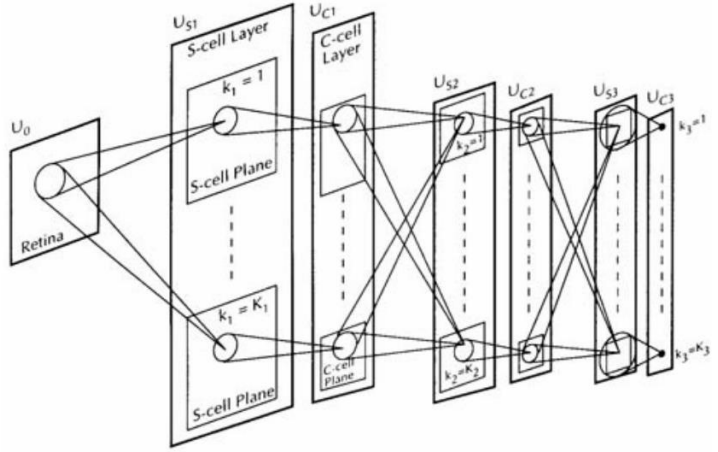




Neocognitron

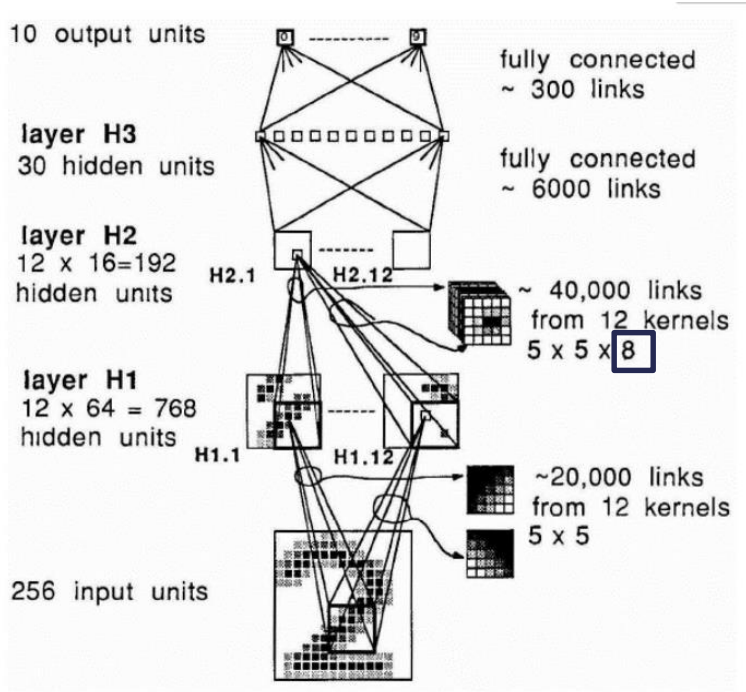
- Hierarchical Multilayered Neural Network(K. Fukushima, 1980)
  - Alternating layers of simple cell(S-cell layer) and complex cell(C-cell layer).
  - S-cell plane and C-cell plane are paired with sparse and localized connectivity.
  - S-cell detects patterns while C-cell integrate their activation
  - S-cells are trained unsupervised way, similar to self-organizing map : 'Winner takes all'
  - A pattern recognition model which is not affected by deformation, scaling and shift





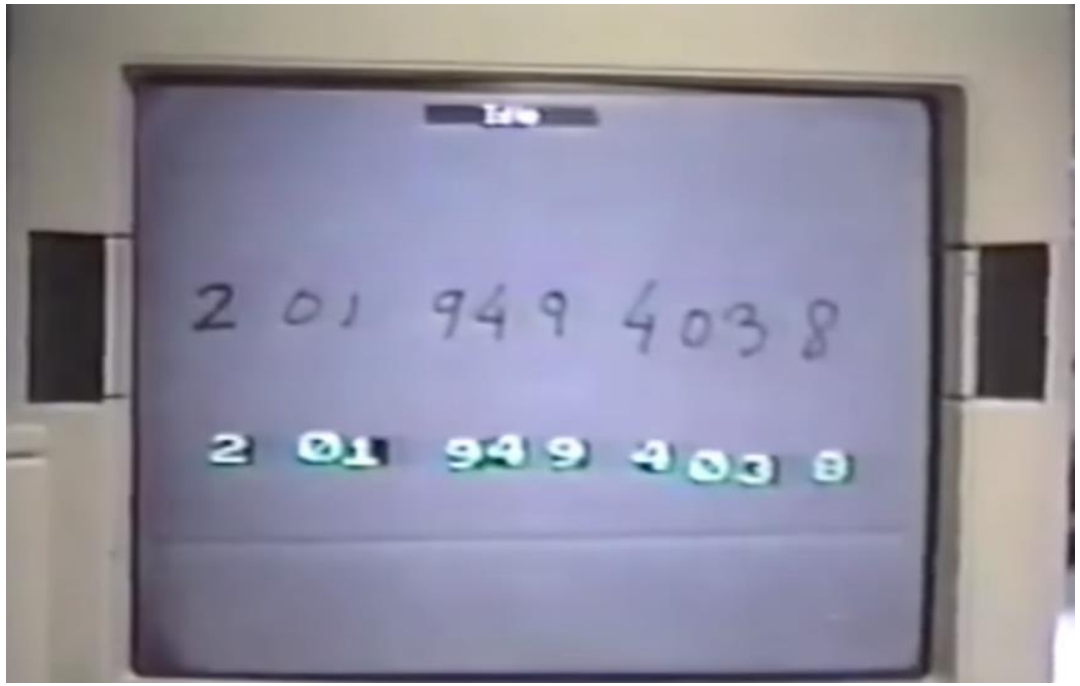
LeNet

- LeNet(LeCun, 1989)
  - Visual recognition system for handwritten zip code
  - 2 Convolution layer and 1 fully connected layer.
  - Weight sharing : All units in the same convolution plane share weights.
  - Supervised : Trained with backpropagation algorithm with stochastic gradient decent
  - No pooling layer.



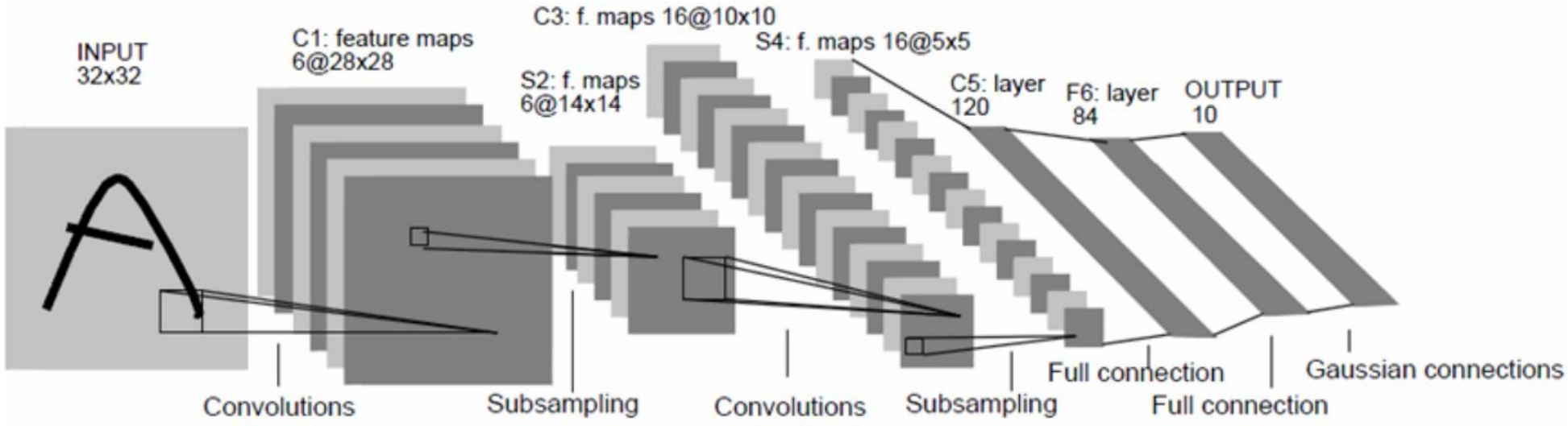
LeNet Demo

- LeNet Demo by LeCun
  - At the times he was at AT&T Bell Lab.

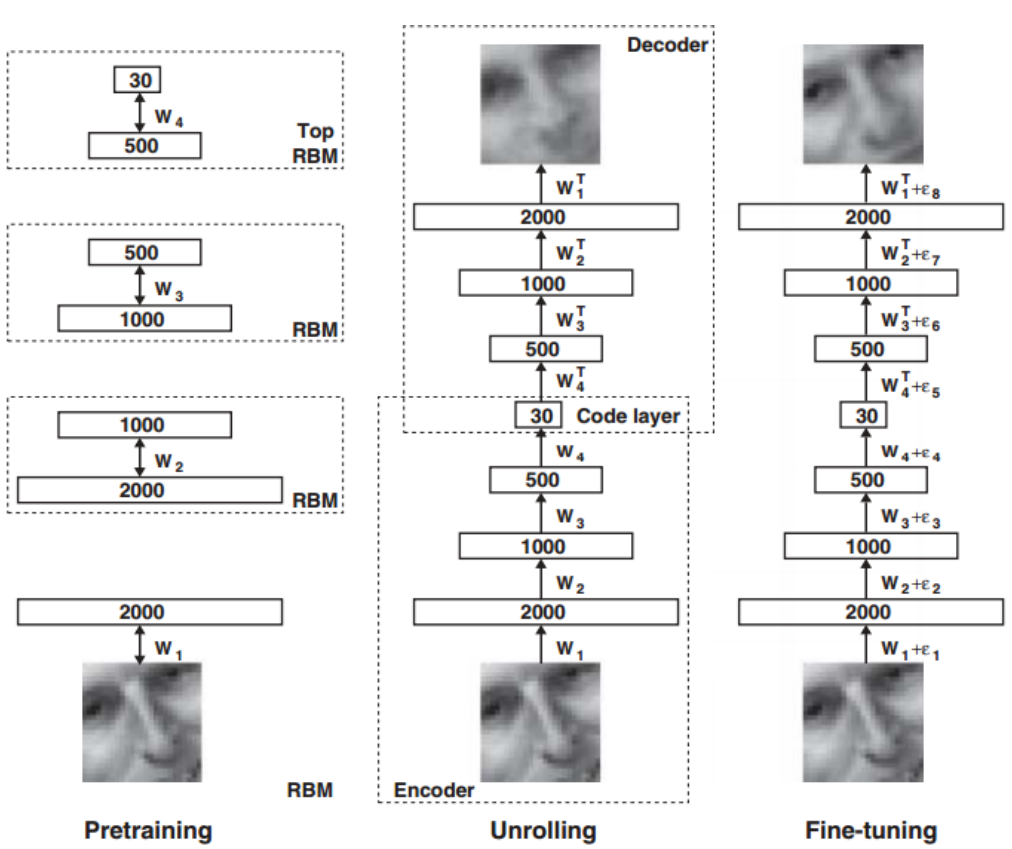


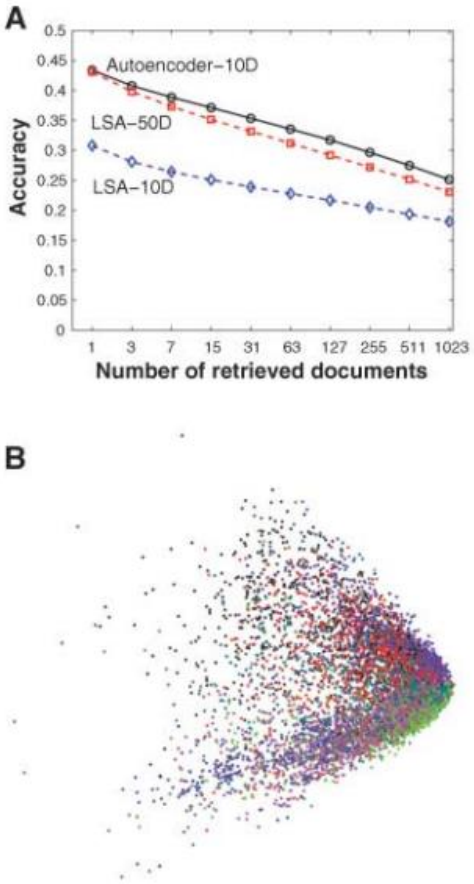
LeNet-5

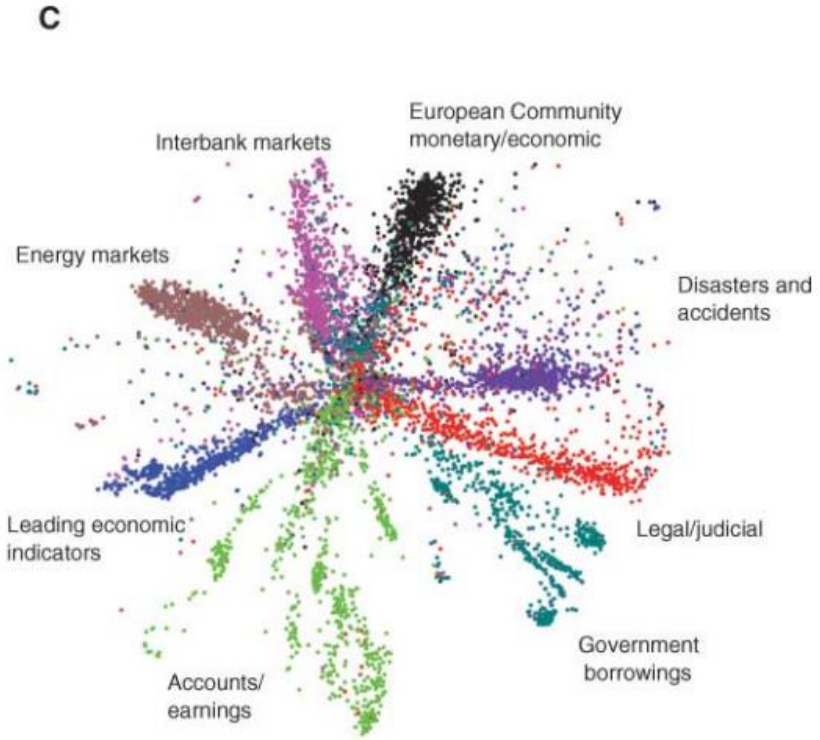
- LeNet-5(LeCun, 1998)
  - Commercially used CNN for reading bank checks.
  - Pooling or sub-sampling layer is introduced for transition invariance.
  - Standard structure of modern CNNs



Stacking RBMs for Deep Auto-Encoder (2006)

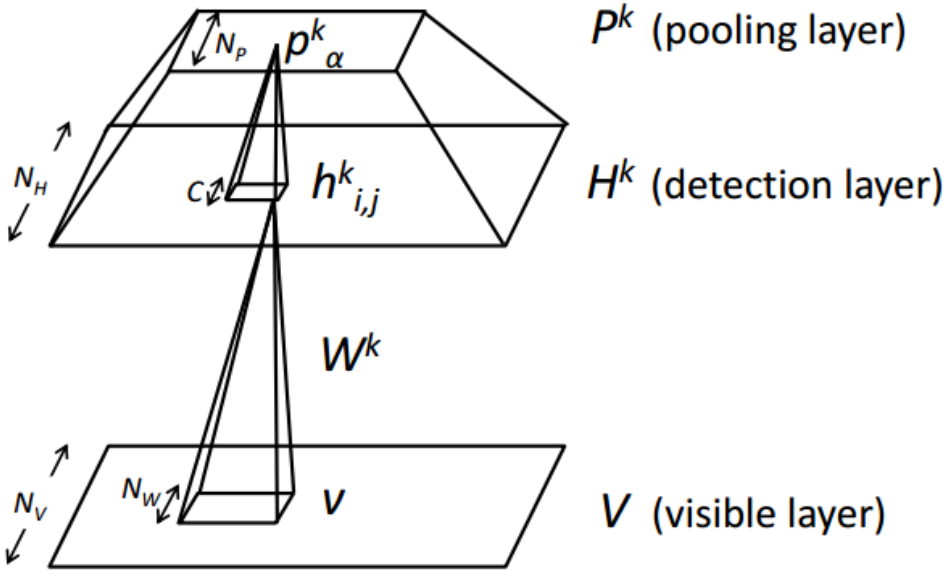


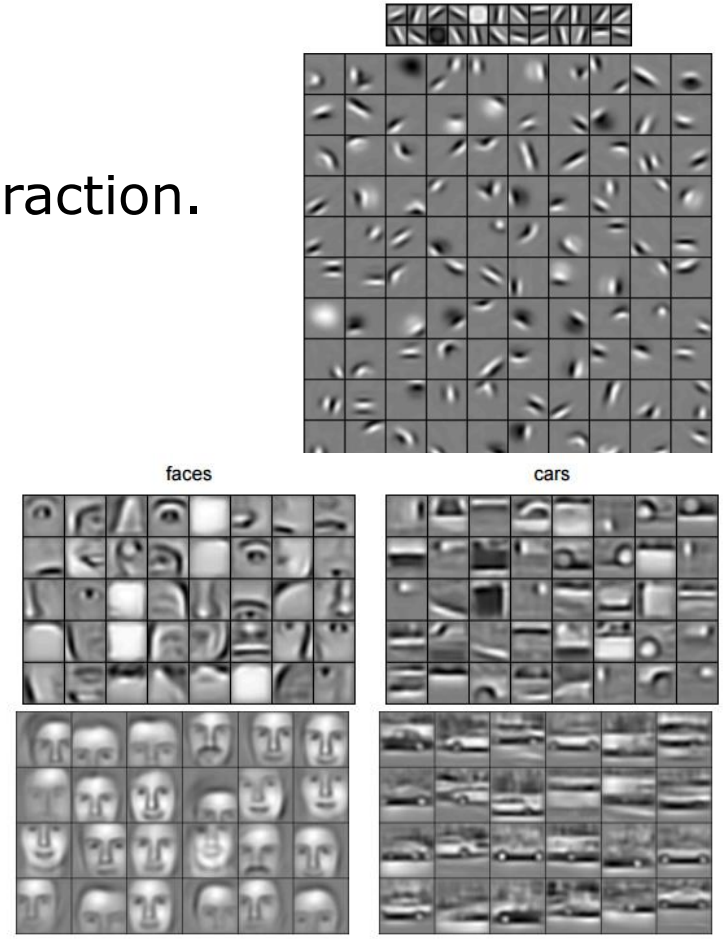




Convolutional Deep Belief Network (2009)

- Learning Hierarchical Representation
  - Unsupervised learning using convolution and probabilistic pooling for hierarchical feature extraction.

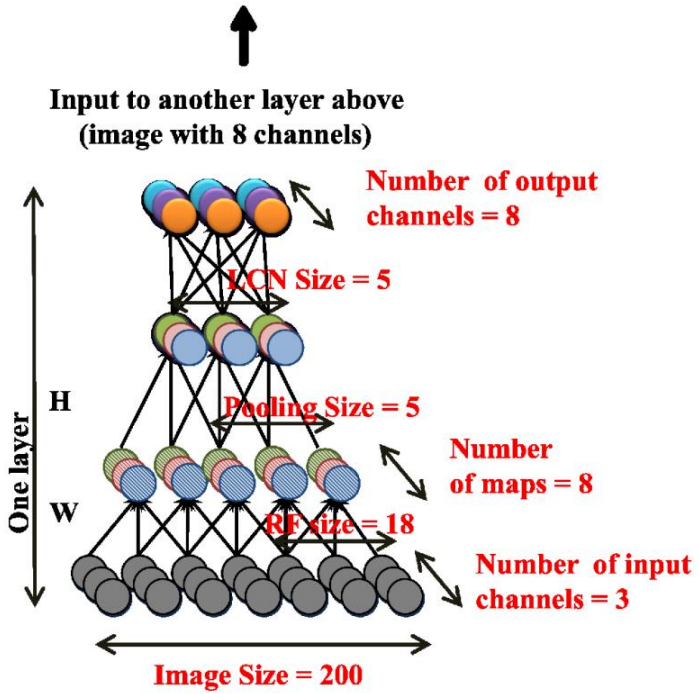


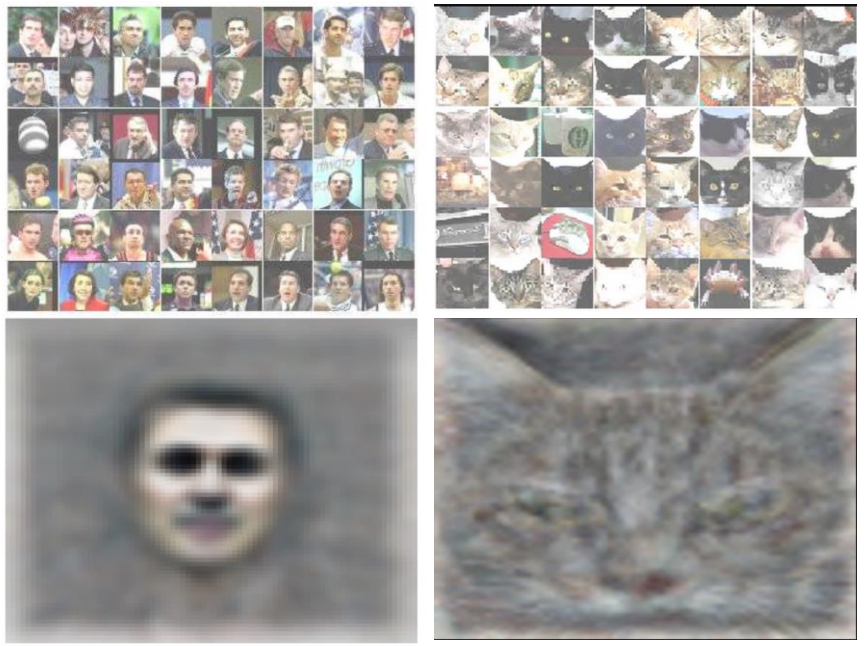


H. Lee et al, 2009

Deep Sparse Auto Encoder (2012)

- Fully-unsupervised Learning for 'Concept' Learning
  - Using locally connected layers, pooling, local contrast normalization, simulating high-level class-specific neuron is possible.



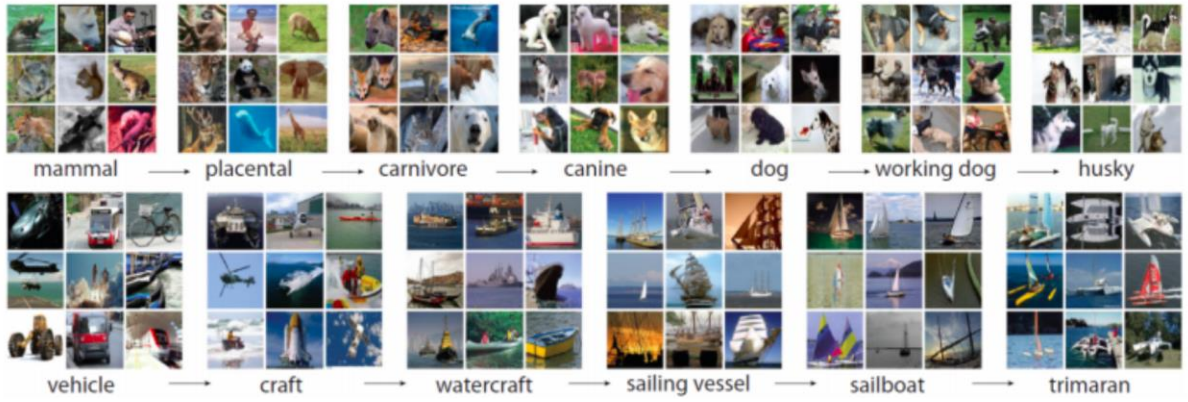


Q. Le et al, 2012

Modern CNNs

ImageNet Large Scale Visual Recognition Challenge

- ImageNet Dataset
  - 22K Categories
  - 14M annotated images



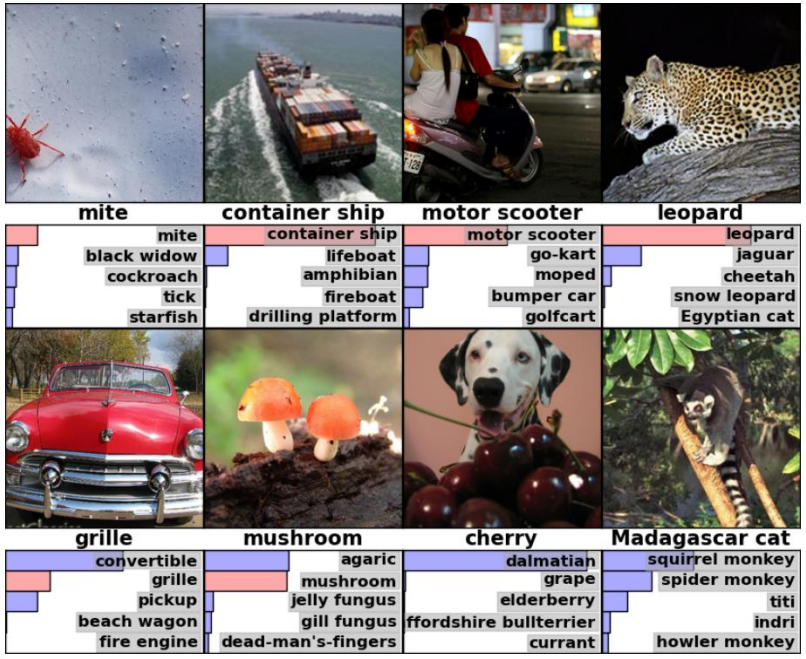
→ Fungus: 134 classes, 90K images

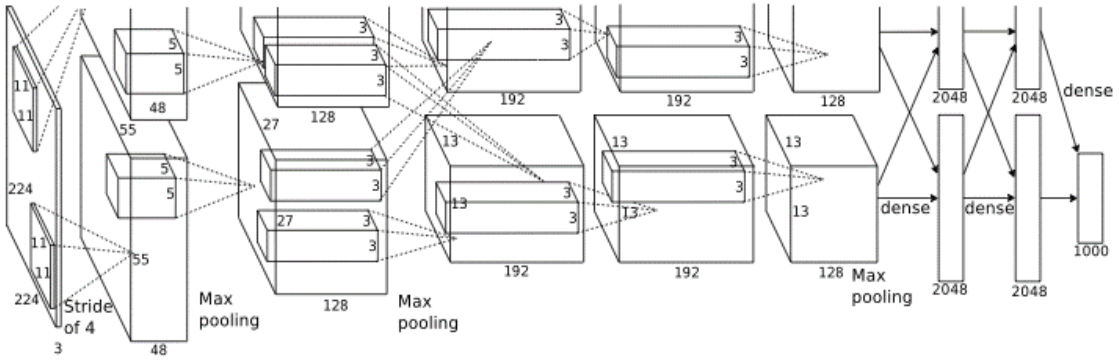
→ Vehicle: 262 classes, 226K images

Must Watch : https://www.youtube.com/watch?v=40riCqvRoMs

Convolutional Neural Network in ILSVRC (2012)

IMAGENET





ILSVRC 2012

SuperVision	15.3%	Deep CNN
ISI	26.1%	FV + PA
OXFORD_VGG	26.7%	FV + SVM
XRCE/INRIA	27.1%	FV + SVM
Univ. of Amsterdam	29.6%	FV + SVM
LEAR-XRCE	34.5%	FV + NCM

ILSVRC 2013

Clarifai	11.7%	Deep CNN
NUS	13.0%	SVM based + Deep CNN
ZF	13.5%	Deep CNN
Andrew Howard	13.6%	Deep CNN
OverFeat-NYU	14.1%	Deep CNN
UvA-Euvison	14.2%	Deep CNN

Convolutional Neural Network in ILSVRC (2012~)

- Performance

Team	Year	Place	Top-5 test error
SuperVision	2012	1	16.42%
ISI	2012	2	26.17%
VGG	2012	3	26.98%
Clarifai	2013	1	11.74%
NUS	2013	2	12.95%
ZF	2013	3	13.51%
GoogLeNet	2014	1	6.66%
VGG	2014	2	7.32%
MSRA	2014	3	8.06%
Andrew Howard	2014	4	8.11%
DeeperVision	2014	5	9.51%

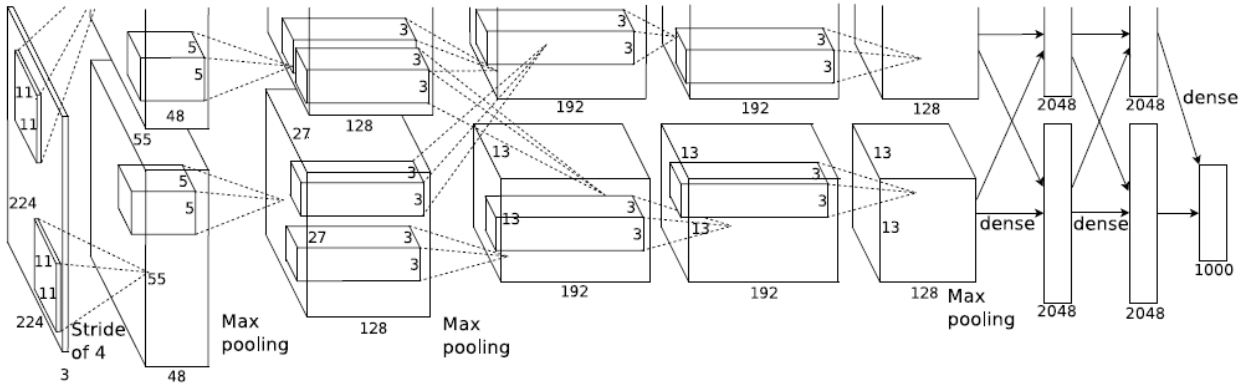
Team	Date	Top-5 test error
GoogLeNet	2014	6.66%
Deep Image	01/12/2015	5.98%
Deep Image	02/05/2015	5.33%
Microsoft	02/05/2015	4.94%
Google	03/02/2015	**4.82%**

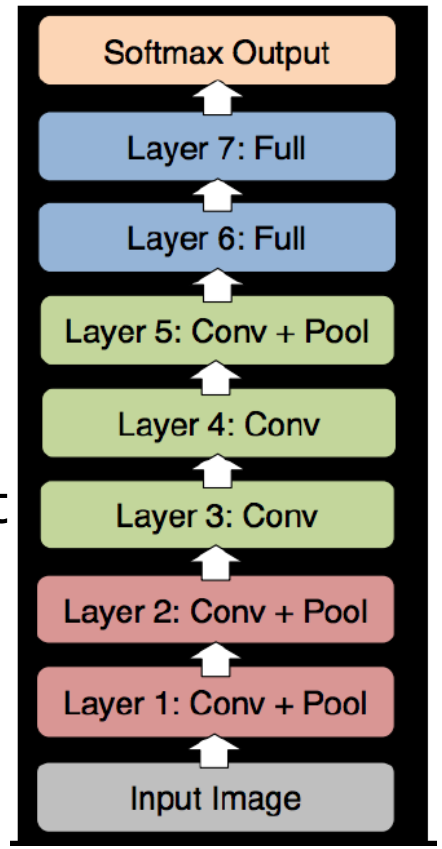
Alex Net

- Winner of ILSVRC 2012(16.4%)
  - Multi-GPU Implementation
  - Local Response Normalization(+1.2%)

$$b^i_{x,y} = a^i_{x,y} / \left(k + \alpha \sum_{j=\max(0,i-n/2)}^{\min(N-1,i+n/2)} (a^j_{x,y})^2 \right)^{\beta}$$

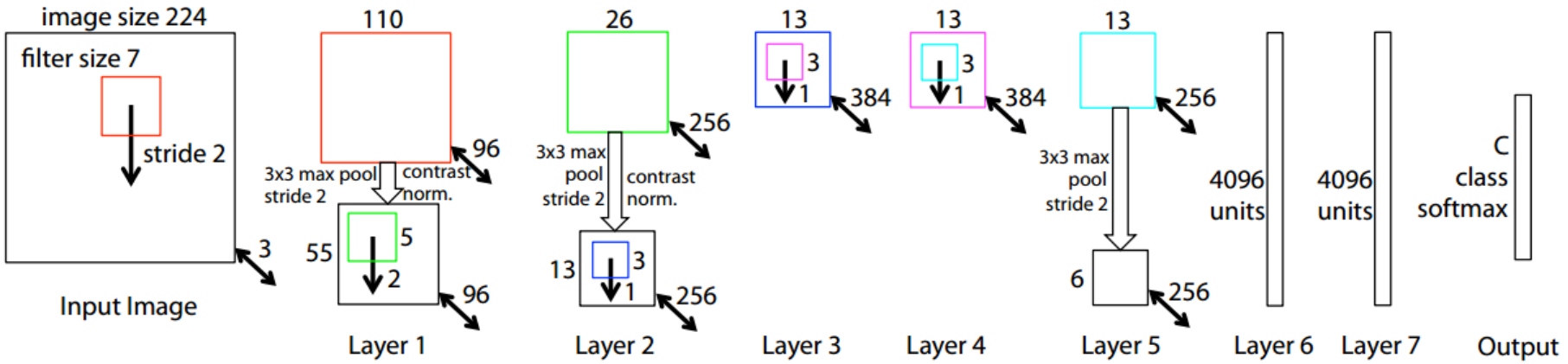
  - Overlapping pooling(+0.3%)
  - Dropout and ReLU
  - Data augmentation(+1%) : Random Crop and Color Shift





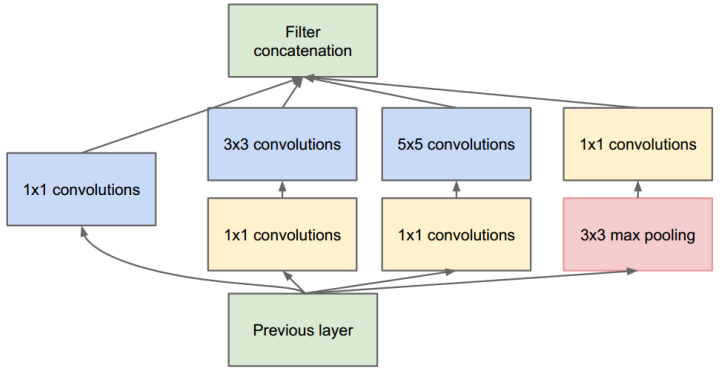
ZF Net

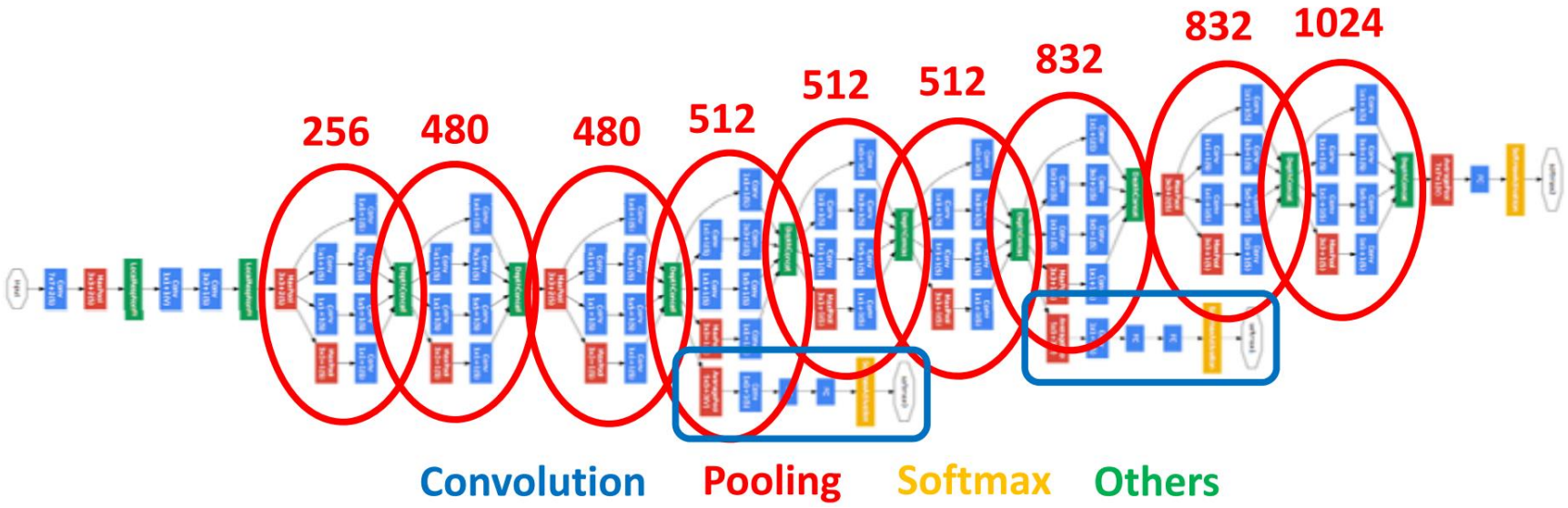
- Winner of ILSVRC 2013(11.74%)
  - Smaller filter in the input layer(11->7)
  - Smaller stride in the input layer(4->2)
  - All other details are similar to Alex Net



GoogLeNet(Inception v1)

- Winner of ILSVRC 2014(6.67%)
  - 9 Inception module : 22 conv layers
  - Auxiliary classifier to facilitate training
  - 1x1 convolutions for dimension reduction
  - Far less parameters than previous nets (Alex Net : 60M, GoogLeNet : 5M)





VGG Net

- The Runner-up of ILSVRC 2014(7.32%)
  - Very deep structure with small filters
  - Memory intensive(140M parameters)
  - Useful for transfer learning

Suppose input has depth C & we want output depth C as well

1x CONV with 7x7 filters	3x CONV with 3x3 filters
Number of weights:	Number of weights:
C*(7*7*C) = **49 C^2**	C*(3*3*C) + C*(3*3*C) + C*(3*3*C) = 3 * 9 * C^2 = **27 C^2**

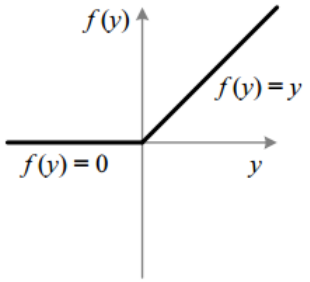
Fewer parameters and more nonlinearities = GOOD.

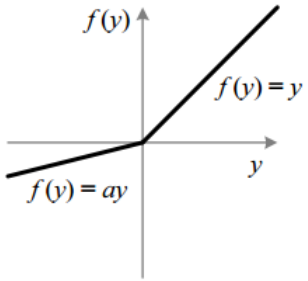
ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	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					

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

MS Net

- First CNN Surpassing Human Performance(4.94%)
  - Parametric ReLU
  - Smart weight initialization
  - Train deeper network





$$f(y_i) = \begin{cases} y_i, & \text{if } y_i > 0 \\ a_i y_i, & \text{if } y_i \leq 0 \end{cases}$$

$$\frac{1}{2} n_l Var[w_l] = 1, \quad \forall l. \quad \Rightarrow \quad \sqrt{2/n_l} \qquad n_l = k_l^2 c_l$$

input size	VGG-19 [25]	model A	model B	model C
224	3×3, 64 3×3, 64 2×2 maxpool, /2	7×7, 96, /2	7×7, 96, /2	7×7, 96, /2
112	3×3, 128 3×3, 128 2×2 maxpool, /2	2×2 maxpool, /2	2×2 maxpool, /2	2×2 maxpool, /2
56	3×3, 256 3×3, 256 3×3, 256 3×3, 256 2×2 maxpool, /2	3×3, 256 3×3, 256 3×3, 256 3×3, 256 3×3, 256 2×2 maxpool, /2	3×3, 256 3×3, 256 3×3, 256 3×3, 256 3×3, 256 3×3, 256 2×2 maxpool, /2	3×3, 384 3×3, 384 3×3, 384 3×3, 384 3×3, 384 3×3, 384 2×2 maxpool, /2
28	3×3, 512 3×3, 512 3×3, 512 3×3, 512 2×2 maxpool, /2	3×3, 512 3×3, 512 3×3, 512 3×3, 512 3×3, 512 2×2 maxpool, /2	3×3, 512 3×3, 512 3×3, 512 3×3, 512 3×3, 512 3×3, 512 2×2 maxpool, /2	3×3, 768 3×3, 768 3×3, 768 3×3, 768 3×3, 768 3×3, 768 2×2 maxpool, /2
14	3×3, 512 3×3, 512 3×3, 512 3×3, 512 2×2 maxpool, /2	3×3, 512 3×3, 512 3×3, 512 3×3, 512 3×3, 512 spp, {7, 3, 2, 1}	3×3, 512 3×3, 512 3×3, 512 3×3, 512 3×3, 512 3×3, 512 spp, {7, 3, 2, 1}	3×3, 896 3×3, 896 3×3, 896 3×3, 896 3×3, 896 3×3, 896 spp, {7, 3, 2, 1}
fc_1 fc_2 fc_3	4096 4096 1000			
depth (conv+fc)	19	19	22	22
complexity (ops., $\times 10^{10}$)	1.96	1.90	2.32	5.30

Baidu Net

- Brute-force Approach with State-of-the-Art Performance(4.58%?)
  - Intensive data augmentation with high resolution models
  - High performance computing

- Scale up, up to 100 nodes
  - High bandwidth low latency ⟹ **Infiniband**
- 36 nodes, 144 GPUs, 6.9TB Host, 1.7TB Device
  - **0.6 PFLOPS**
- **Highly Optimized software stack**
  - RDMA/GPU Direct
  - New data partition and communication strategies

Possible variations

Augmentation	The number of possible changes
Color casting	68920
Vignetting	1960
Lens distortion	260
Rotation	20
Flipping	2
Cropping	82944(crop size is 224x224, input image size is 512x512)

The Deep Image system learned from **~2 billion** examples, out of **90 billion** possible candidates.

Inception-v2

- Batch Normalization
  - Removing covariate shift using batch-wise normalization
  - Facilitates faster and more stable learning
  - Top-5 error reduced to 4.82%

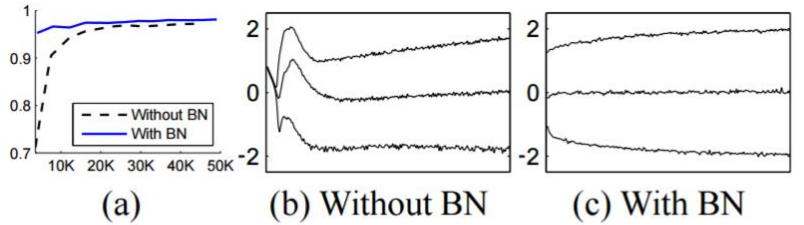
Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

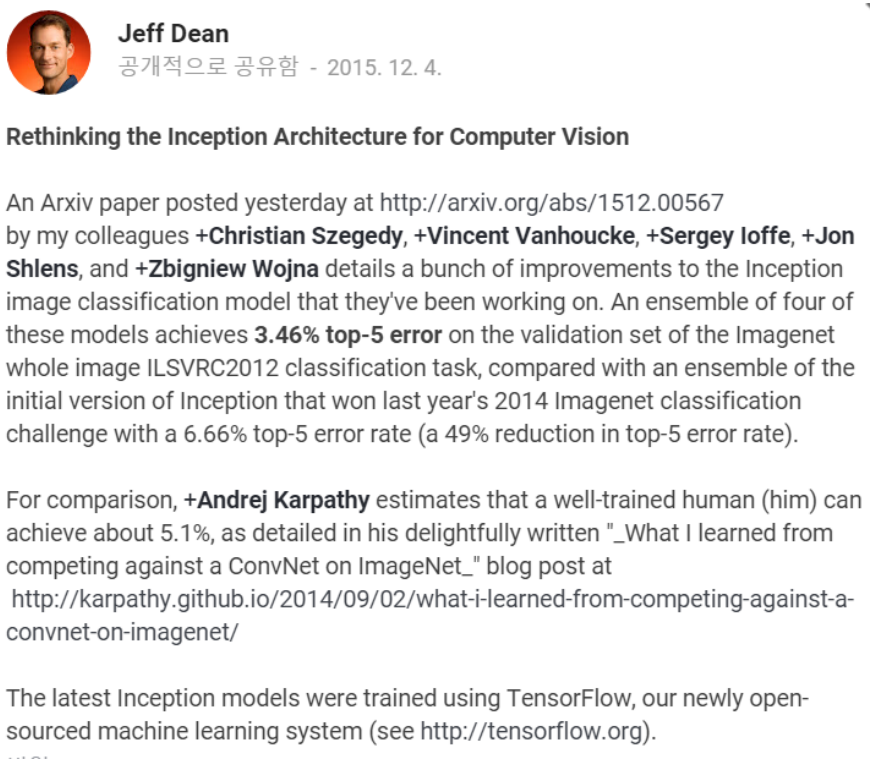


(a) (b) Without BN (c) With BN

Figure 1: (a) *The test accuracy of the MNIST network trained with and without Batch Normalization, vs. the number of training steps. Batch Normalization helps the network train faster and achieve higher accuracy.* (b, c) *The evolution of input distributions to a typical sigmoid, over the course of training, shown as* $\{15, 50, 85\}$*th percentiles. Batch Normalization makes the distribution more stable and reduces the internal covariate shift.*

New ILSVRC Classification Record

- GoogLeNet Again.



Jeff Dean
공개적으로 공유함 - 2015. 12. 4.

Rethinking the Inception Architecture for Computer Vision

An Arxiv paper posted yesterday at http://arxiv.org/abs/1512.00567 by my colleagues **+Christian Szegedy**, **+Vincent Vanhoucke**, **+Sergey Ioffe**, **+Jon Shlens**, and **+Zbigniew Wojna** details a bunch of improvements to the Inception image classification model that they've been working on. An ensemble of four of these models achieves **3.46% top-5 error** on the validation set of the Imagenet whole image ILSVRC2012 classification task, compared with an ensemble of the initial version of Inception that won last year's 2014 Imagenet classification challenge with a 6.66% top-5 error rate (a 49% reduction in top-5 error rate).

For comparison, **+Andrej Karpathy** estimates that a well-trained human (him) can achieve about 5.1%, as detailed in his delightfully written "_What I learned from competing against a ConvNet on ImageNet_" blog post at http://karpathy.github.io/2014/09/02/what-i-learned-from-competing-against-a-convnet-on-imagenet/

The latest Inception models were trained using TensorFlow, our newly open-sourced machine learning system (see http://tensorflow.org).

Team	Date	Top-5 test error
GoogLeNet	2014	6.66%
Deep Image	01/12/2015	5.98%
Deep Image	02/05/2015	5.33%
Microsoft	02/05/2015	4.94%
Google	03/02/2015	**4.82%**

“Inception”

“Batch Normalization”

3.46%

Inception-v3

- Scalability is the Key
  - But naïve scaling up kills efficiency which is of growing importance.
- Observations and Lessons Learned
  - **Principle 1** : Avoid representational bottlenecks especially early in the network.
    - The representation size should be gently decrease from the input to outputs
  - **Principle 2** : Higher dimensional representation are easier to process.
    - Increasing activation will allow disentangled features thus result in faster training.
  - **Principle 3** : Spatial aggregation after reducing the dimension of input is beneficial.
    - Strong correlation between adjacent unit leads to much less information loss during dimension reduction and even it promotes faster learning.
  - **Principle 4** : Balance the width and depth of the network
    - Optimal improvement of the network can be achieved when both are increased in parallel.

Inception-v3

- Factorization into Smaller Convolutions

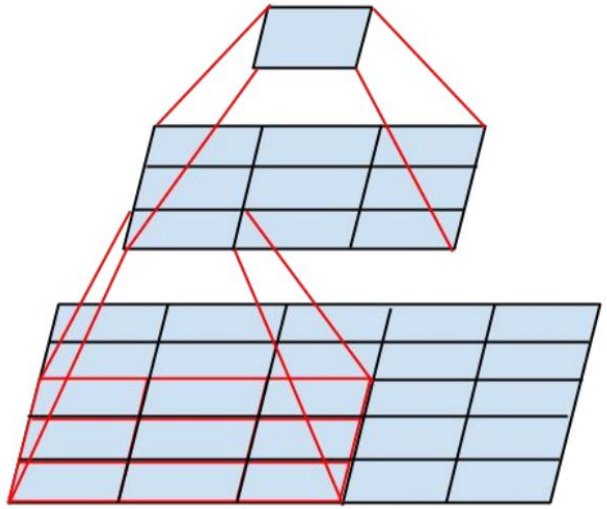


Fig.4

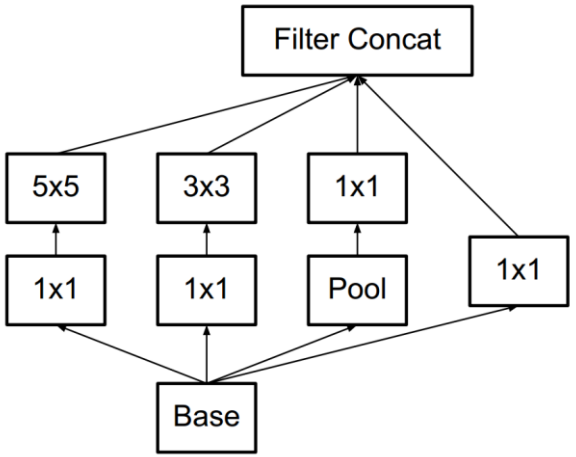
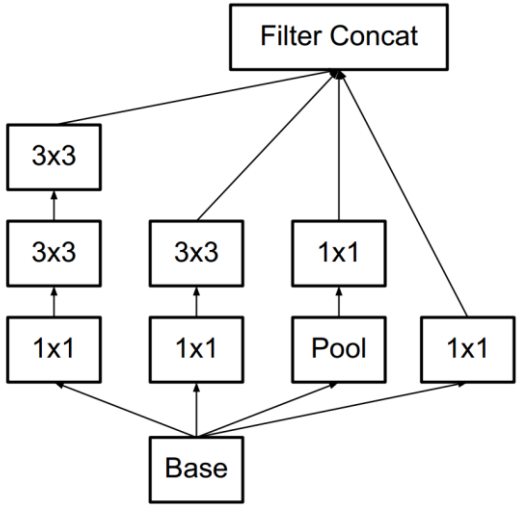


Fig.5

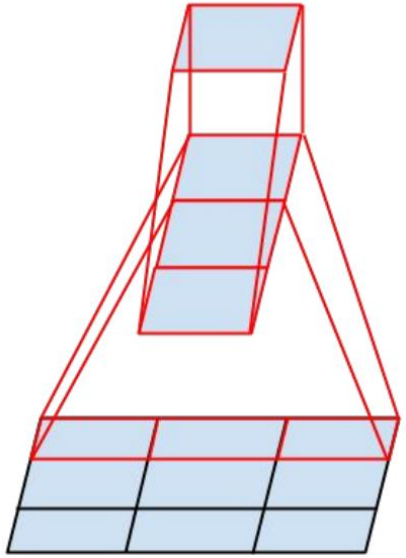


Replacing 5x5 convolutions with two-layered 3x3 convolutions

[Num of parameters]
5x5 convolution = 5x5 = 25
Two layered 3x3 convolution = 3x3 + 3x3 = 18

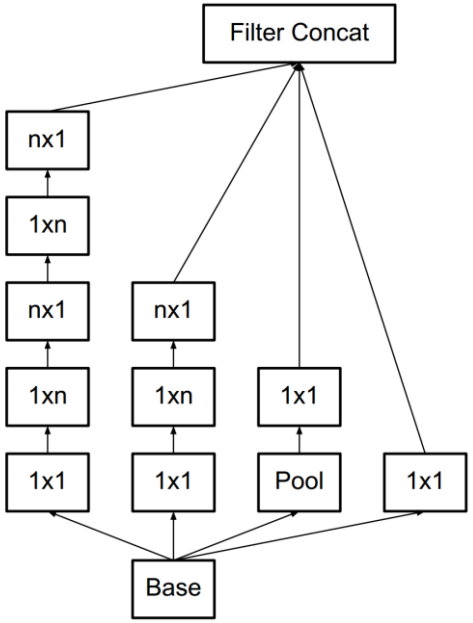
Inception-v3

- Spatial Factorization into Asymmetric Convolutions



Replacing 3x3 convolutions with
3x1 and 1x3 convolutions

Fig.6



"In practice, employing asymmetric factorization does not work well on early layers, but on medium grid-sizes (12 and 20) with using 1 × 7 convolutions followed by 7 × 1 convolutions."

[Num of parameters]
3x3 convolution = 3x3 = 9
3x1 and 1x3 convolution = 3x1 + 1x3 = 6

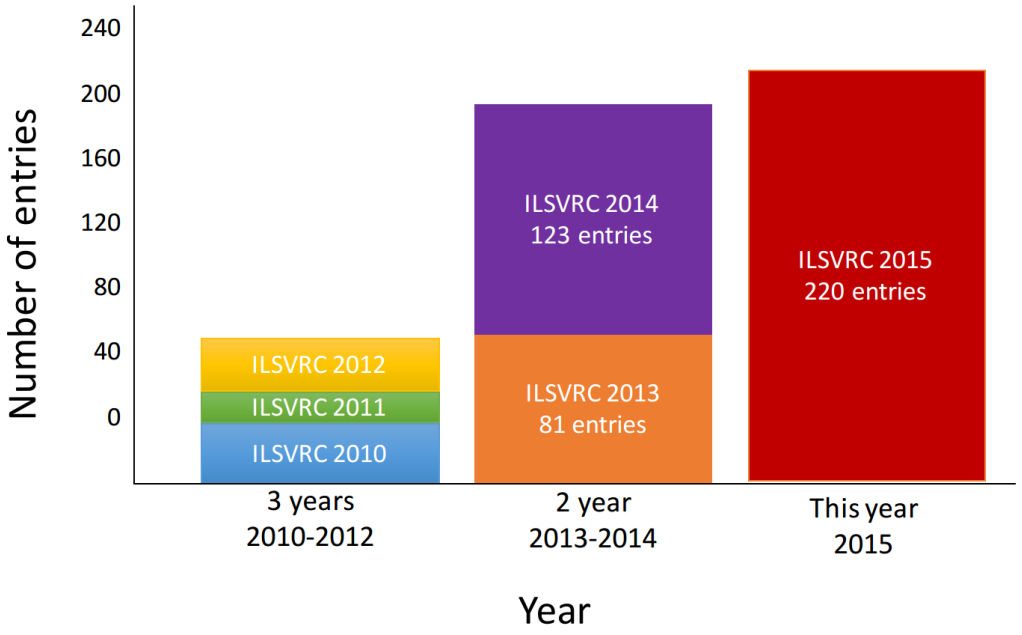
Inception-v3

- Combining All Modules based on the Principles
  - Three 3x3 convolutions instead of 7x7 convolution layer
  - Three traditional Inception module followed by grid reduction in Fig. 10
  - Five factorized Inception in Fig. 5 followed by grid reduction in Fig. 10
  - Two Inception module in Fig. 6
  - Pooling + FC + Softmax
  - 42 Layer deep, 2.5 times computational cost than original GoogLeNet(22 layers)
  - Label smoothing
  - Modifications to auxiliary classifier

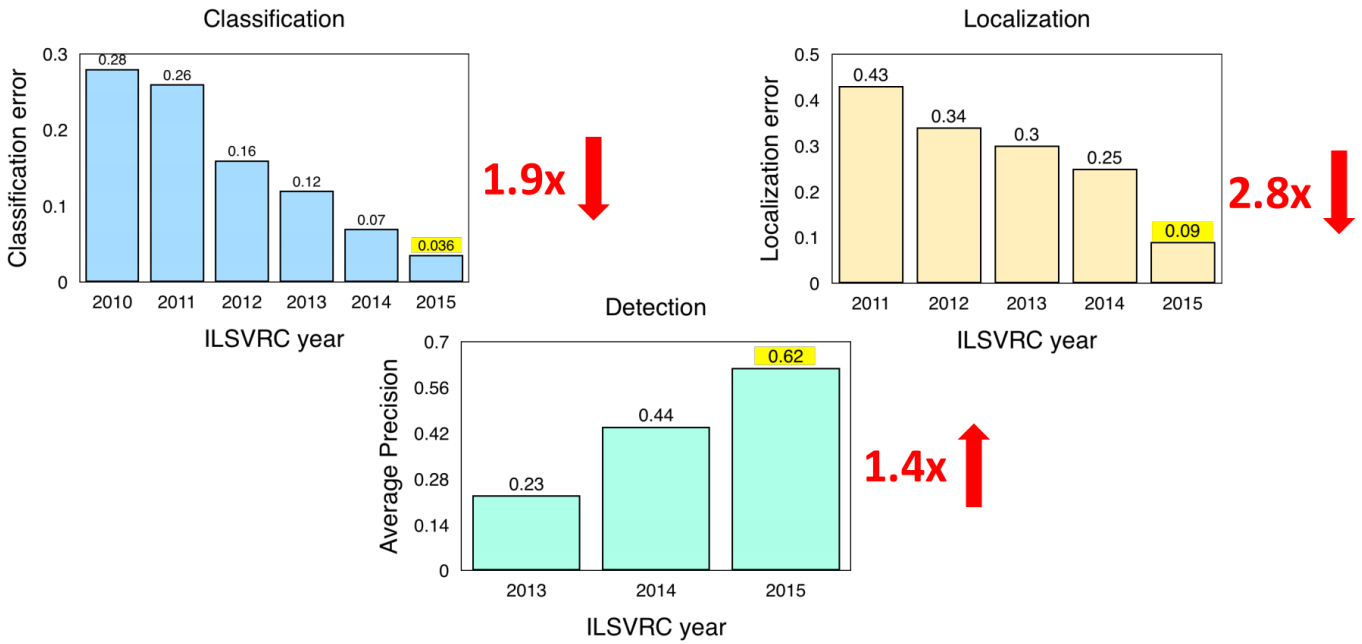
type	patch size/stride or remarks	input size	
conv	3×3/2	299×299×3	
conv	3×3/1	149×149×32	
conv padded	3×3/1	147×147×32	
pool	3×3/2	147×147×64	
conv	3×3/1	73×73×64	+Fig.10
conv	3×3/2	71×71×80	+Fig.10
conv	3×3/1	35×35×192	
3×Inception	As in figure 4	35×35×288	
5×Inception	As in figure 5	17×17×768	
2×Inception	As in figure 6	8×8×1280	
pool	8 × 8	8 × 8 × 2048	
linear	logits	1 × 1 × 2048	
softmax	classifier	1 × 1 × 1000	

ILSVRC 2015

Participation in ILSVRC over the years

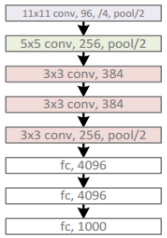


Result in ILSVRC over the years

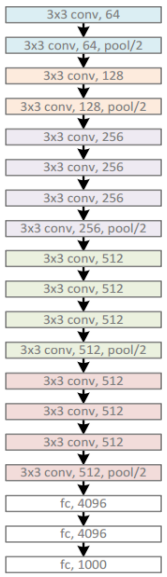


Deeper and Deeper

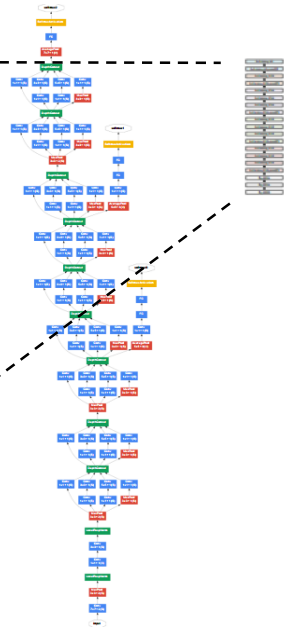
AlexNet, 8 layers
(ILSVRC 2012)



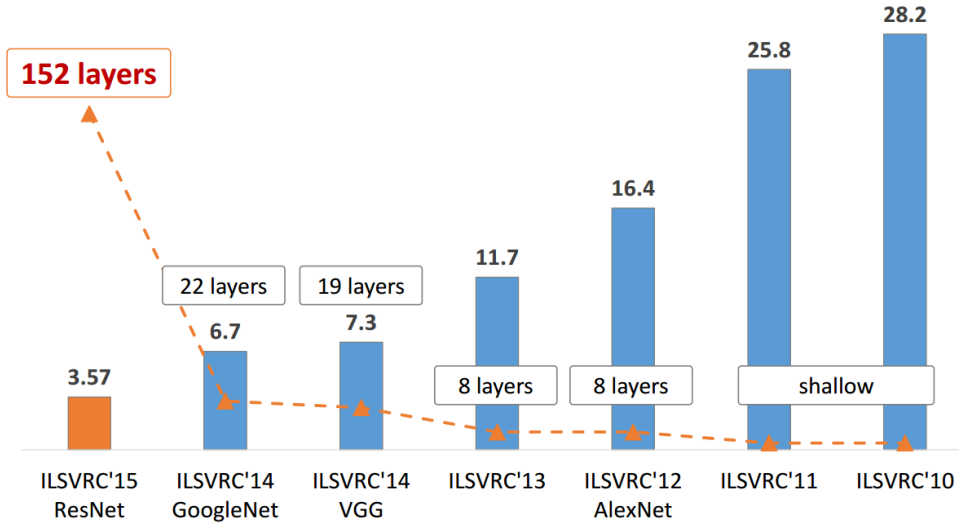
VGG, 19 layers
(ILSVRC 2014)



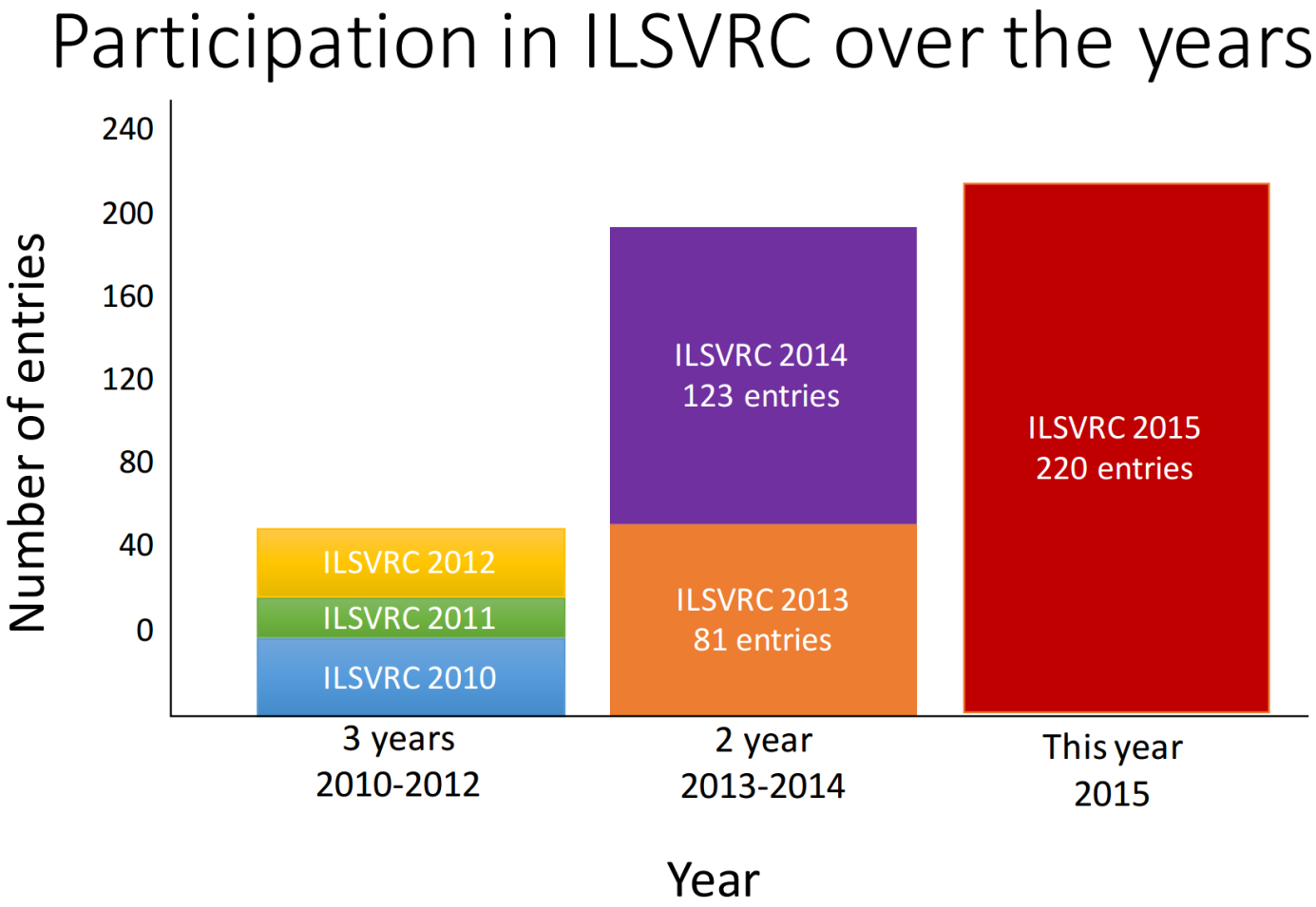
GoogleNet, 22 layers
(ILSVRC 2014)



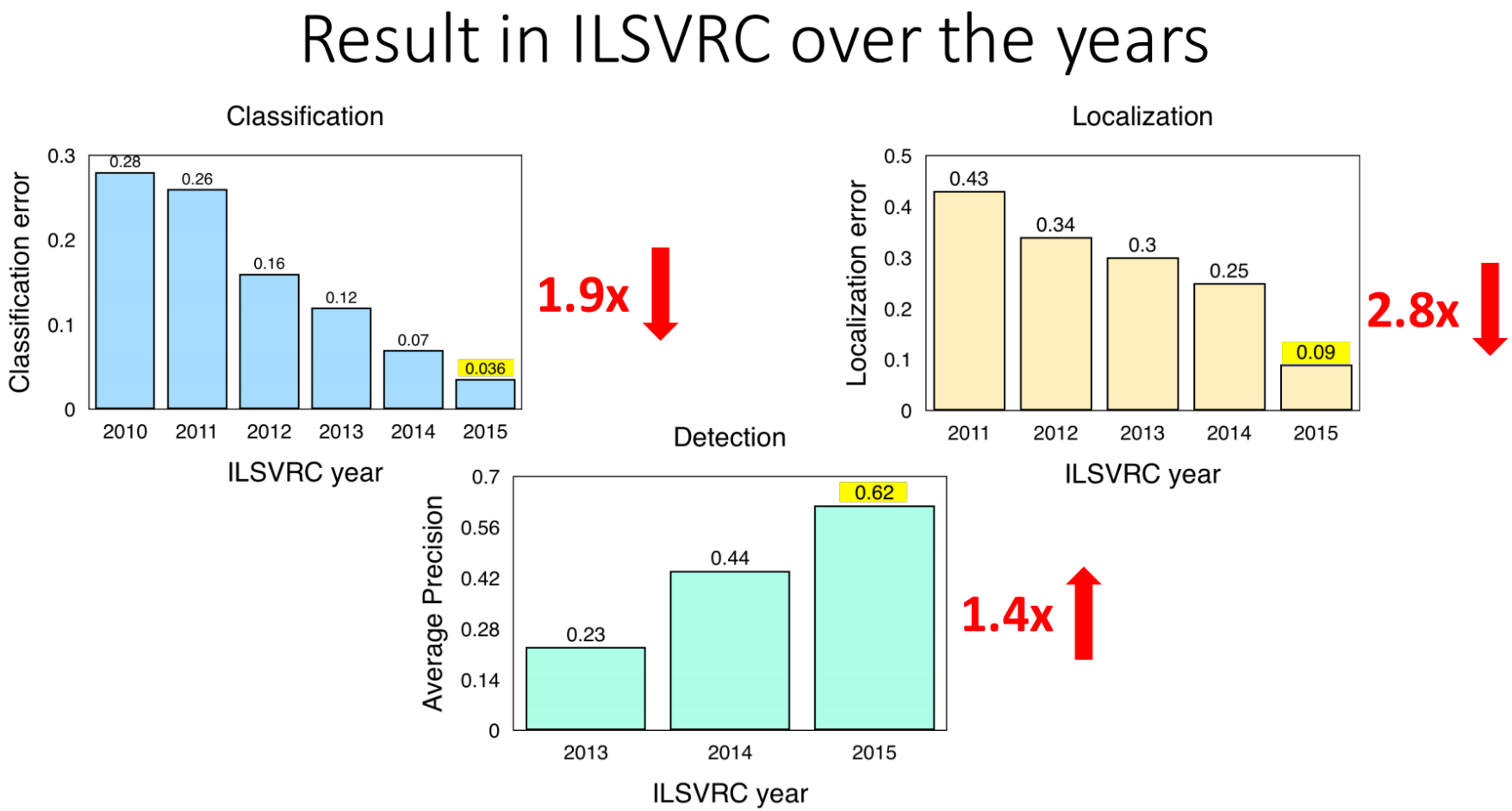
ResNet, 152 layers
(ILSVRC 2015)



Ongoing Popularity of ILSVRC



Ongoing Breakthrough of ILSVRC



ILSVRC 2015 - LOC Result

Team Name	Error (%)
MSRA	9.0
Trimps-Soushen	12.3
Qualcomm Research	12.6
MCG-ICT-CAS	14.7
Lunit-KAIST	14.7
Tencent-Bestimage	15.5
ReCeption	19.6
CUimage	21.2
CIL	23.2
VUNO	25.3

MSRA:
Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun
Microsoft

Trimps-Soushen:
Jie Shao*, Xiaoteng Zhang*, Jianying Zhou*, Zhengyan Ding*, Wenfei Wang, Lin Mei, Chuanping Hu
(* indicates equal contribution)
The Third Research Institute of the Ministry of Public Security, P.R. China.

ILSVRC 2015 - CLS Result

Team Name	Error (%)
MSRA	3.57
ReCeption	3.58
Trimps-Soushen	4.58
Qualcomm Research	4.87
VUNO	5.03
CIL	5.48
CUimage	5.86
MCG-ICT-CAS	6.31
HiVision	6.48

VUNO at ILSVRC 2015

- VUNO 2015
  - CLS 5th
  - LOC 10th
  - Both exceeds the performance of ILSVRC 2014 winners

VUNO	A combination of CNN and CLSTM (averaging)	0.252951	0.05034
VUNO	A combination of two CNNs (averaging)	0.254922	0.05034

- CLS 2014 Winner

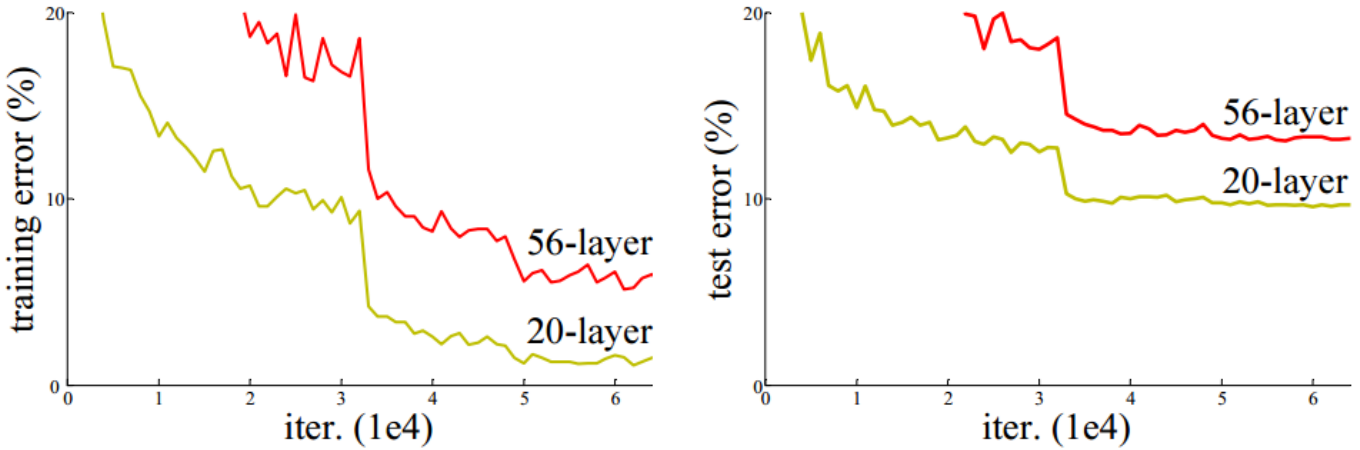
Team name	Entry description	**Classification error**	Localization error
GoogLeNet	No localization. Top5 val score is 6.66% error.	0.06656	0.606257

- LOC 2014 Winner

Team name	Entry description	**Localization error**	Classification error
VGG	a combination of multiple ConvNets (by averaging)	0.253231	0.07405

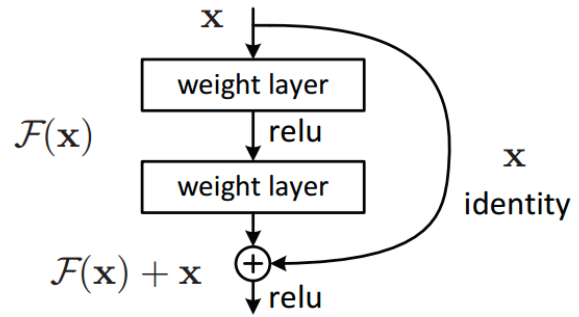
ResNet

- Simply Stacking More Layers is Not the Answer
  - Training and testing error on CIFAR-10 with 20-layer and 56-layer networks.
  - This is not overfitting : both training and test error increases with the depth.
  - By construction, there exists a deeper model with exactly same performance by adding layers with identity mapping.
  - But we cannot find deep network the performance which is comparable to this constructed network



ResNet

- Explicitly Making Layers to Learn Residual Mapping
  - Denoting underlying desired mapping as $\mathcal{H}(\mathbf{x})$, we let the stacked nonlinear layers fit another mapping $\mathcal{F}(\mathbf{x}) := \mathcal{H}(\mathbf{x}) - \mathbf{x}$
  - If the optimal mapping is identity, it would be easier to push residual to zero than learning identity with stack of nonlinear layers.
  - The original formulation $\mathcal{H}(\mathbf{x}) := \mathcal{F}(\mathbf{x}) + \mathbf{x}$ can be achieved with 'shortcut connection' which is just an identity mapping.
  - This identity shortcut connection do not add any parameters or complexity.
  - The identity connection plays role of 'preconditioning' which make the layers easy to approximate the underlying mapping.

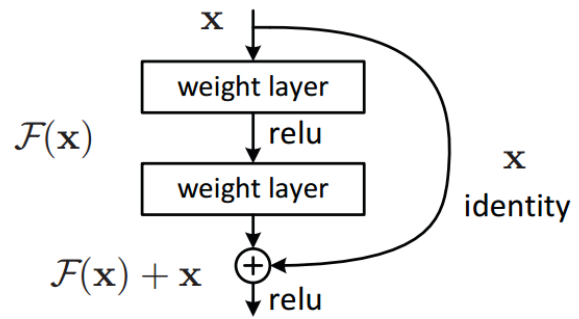


ResNet

- Identity Mapping by Shortcuts
  - Building block is defined by $\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + \mathbf{x}$. where $\mathcal{F}(\mathbf{x}, \{W_i\})$ represent the residual mapping to be learned.
  - For the building block below that has two layers, $\mathcal{F} = W_2\sigma(W_1\mathbf{x})$ where σ is ReLU.
  - We adopt second nonlinearity after element-wise addition $\mathcal{F} + \mathbf{x}$
  - When input-output dimension is not same, we perform linear projection W_s to make the shortcut connections to match the dimensions as

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + W_s\mathbf{x}.$$

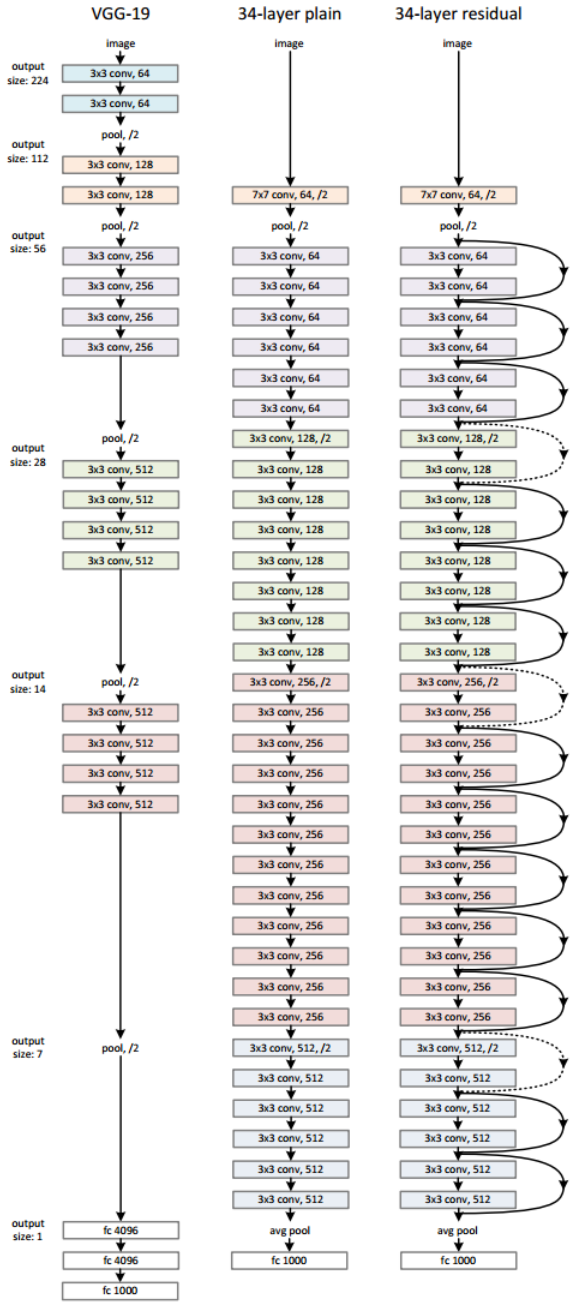
  - When the function $\mathcal{F}(\mathbf{x}, \{W_i\})$ is multiple convolutional layers, the element-wise addition is performed channel by channel. Conv layers with depth 2 or 3 are used here.
  - The form of $\mathcal{F}(\mathbf{x}, \{W_i\})$ is flexible but single layer is not beneficial since it leads to $\mathbf{y} = W_1\mathbf{x} + \mathbf{x}$



ResNet

- Residual Network
  - Add shortcut connections
    - When input-output dimension matches, add identity connection.
    - When input-output dimension dismatches,
      a)zero-padding, b)add linear projection(1x1 conv.)

- Implementation Detail
  - Training
    - Scale jittering in [256, 480] for shorter side and crop 224x224.
    - Random crop and random flip, perf-pixel mean substraction.
    - Color augmentation and batch normalization, ms-init
    - SGD with mini-batch 256, learning rate starts from 0.1 and decayed by 10 when plateau. Weight decay 0.0001 with momentum 0.9
  - Testing
    - 10-crop testing with fully-convolutional from.
    - Averaging scores from multiple scales(224,256,384,480,640)



ResNet

- ImageNet Performance
  - Comparison with other models
    - Time complexity of 50-layer ResNet : 3.8 billion FLOPs,
      152-layer ResNet : 11.3 billion, VGG-16/19 : 15.3/19.6 billion FLOPs

method	top-1 err.	top-5 err.
VGG [41] (ILSVRC'14)	-	8.43†
GoogLeNet [44] (ILSVRC'14)	-	7.89
VGG [41] (v5)	24.4	7.1
PReLU-net [13]	21.59	5.71
BN-inception [16]	21.99	5.81
ResNet-34 B	21.84	5.71
ResNet-34 C	21.53	5.60
ResNet-50	20.74	5.25
ResNet-101	19.87	4.60
ResNet-152	**19.38**	**4.49**

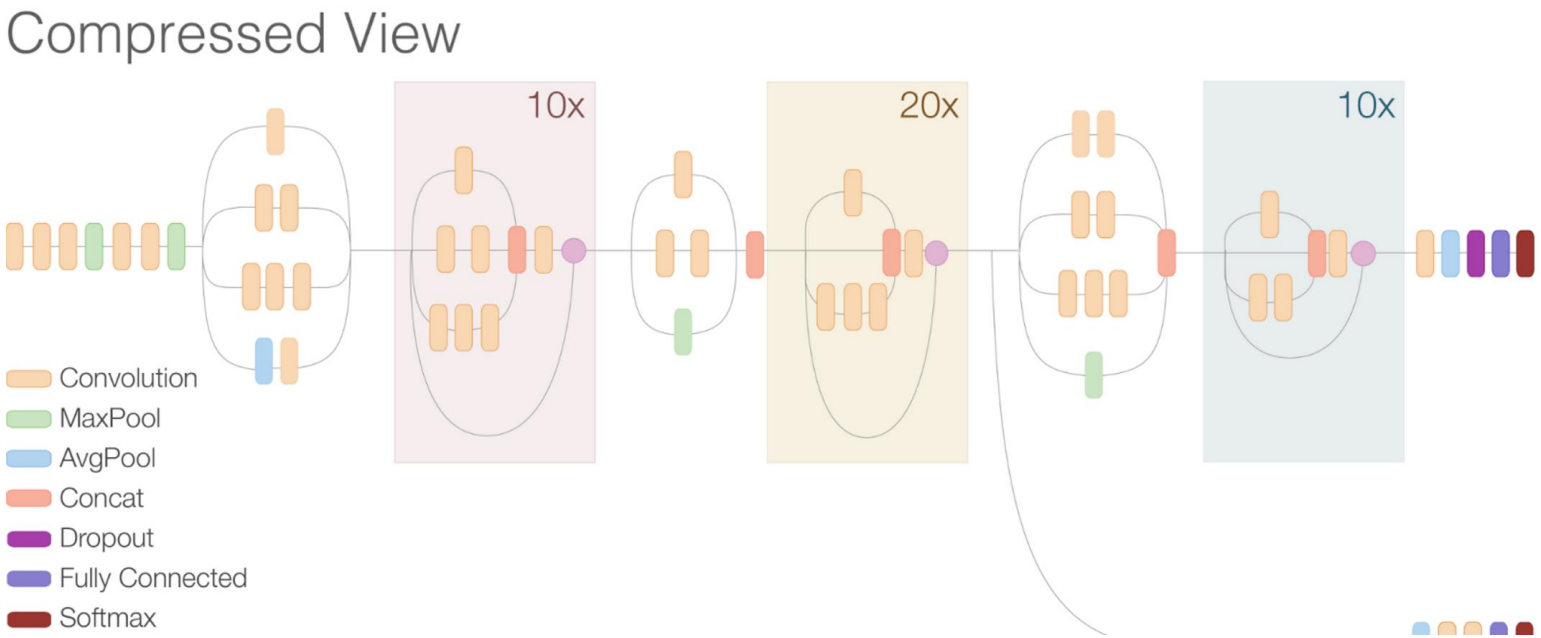
Single model on validation

method	top-5 err. (**test**)
VGG [41] (ILSVRC'14)	7.32
GoogLeNet [44] (ILSVRC'14)	6.66
VGG [41] (v5)	6.8
PReLU-net [13]	4.94
BN-inception [16]	4.82
ResNet (ILSVRC'15)	**3.57**

Ensemble on test

Inception v4 and Inception-ResNet

- Adopting Residual Connection to Inception
  - Inception + Residual Learning
  - Top-5 performance : 3.08%

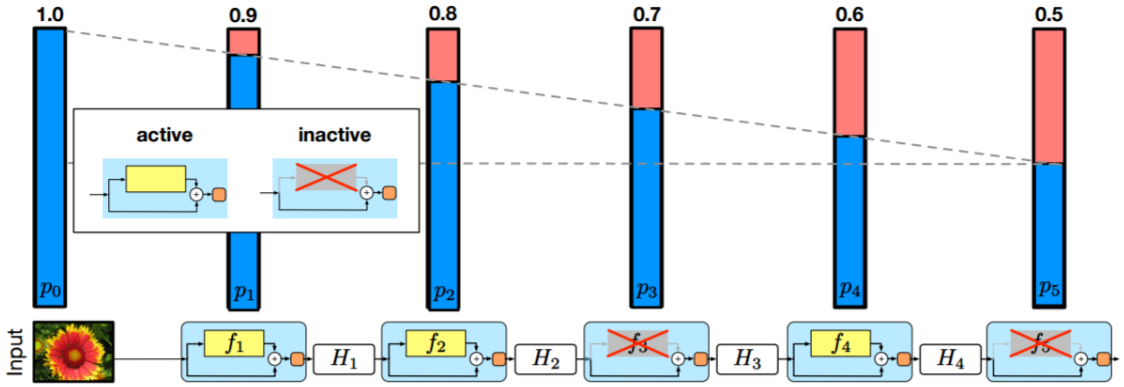


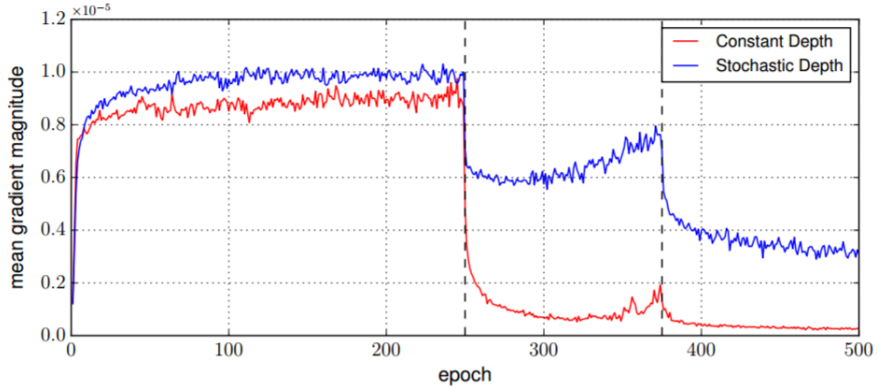
Inception v1-v4

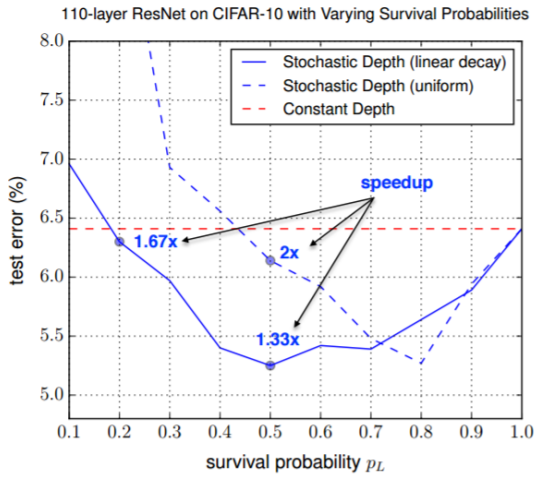
- [v1] Going Deeper with Convolutions, 6.67% test error, http://arxiv.org/abs/1409.4842
- [v2] Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, 4.8% test error, http://arxiv.org/abs/1502.03167
- [v3] Rethinking the Inception Architecture for Computer Vision, 3.5% test error, http://arxiv.org/abs/1512.00567
- [v4] Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning, 3.08% test error, http://arxiv.org/abs/1602.07261

Stochastic Depth

- Training with effectively shallower network, testing with ensemble of variable depth network





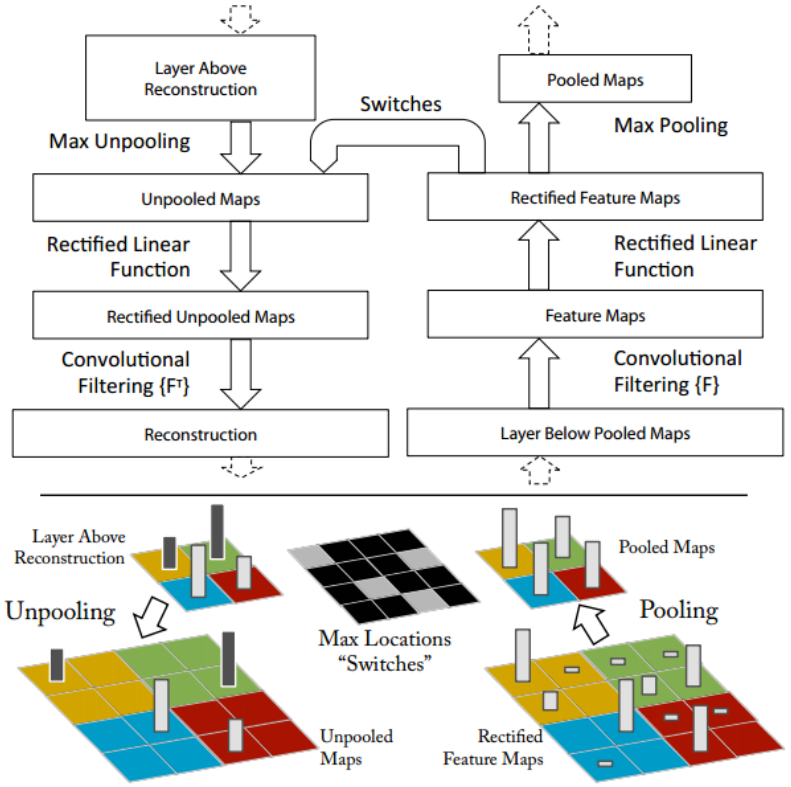


Train : $H_\ell = \texttt{ReLU}(b_\ell f_\ell(H_{\ell-1}) + \text{id}(H_{\ell-1})) \quad b_\ell \in \{0, 1\}$

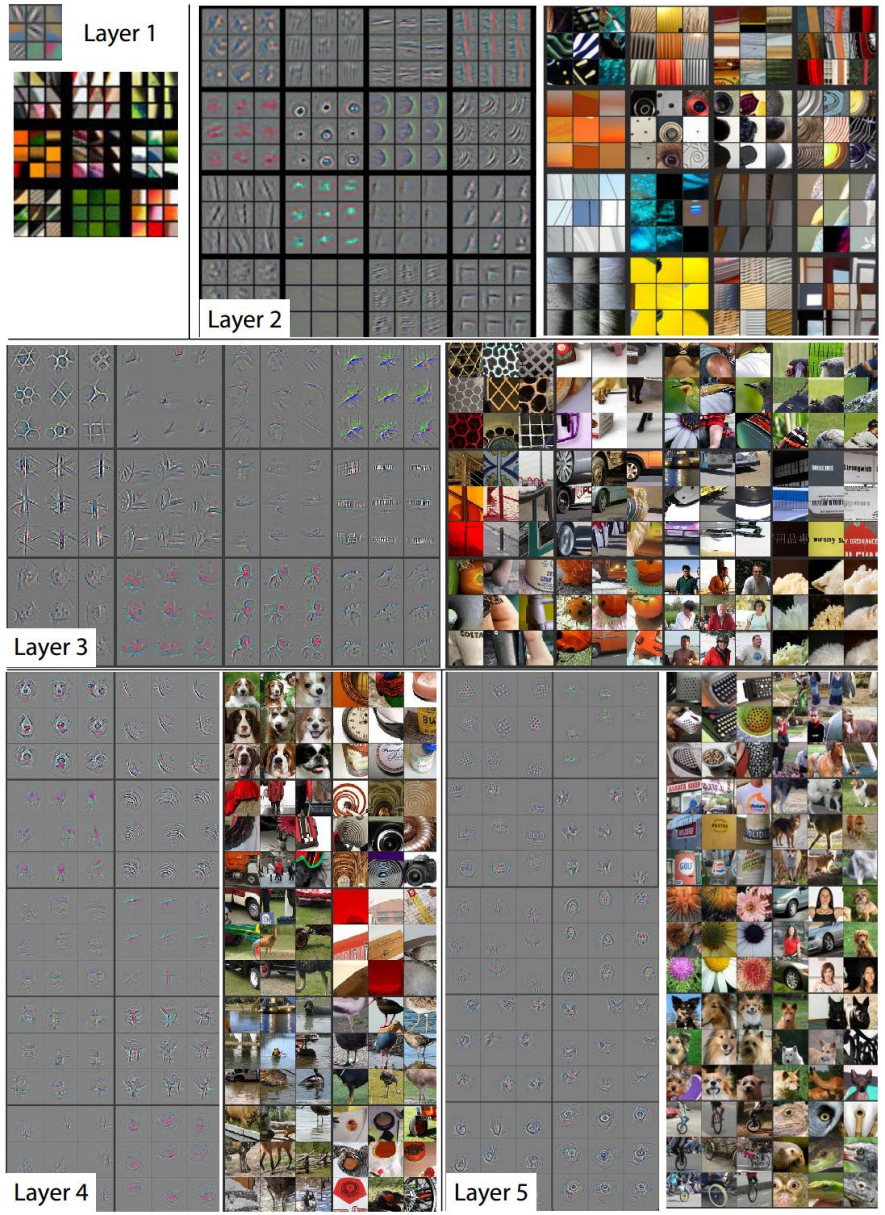
Test : $H_\ell^{\text{Test}} = \texttt{ReLU}(p_\ell f_\ell(H_{\ell-1}^{\text{Test}}; W_\ell) + H_{\ell-1}^{\text{Test}}) \quad p_\ell = 1 - \frac{\ell}{L}(1 - p_L)$

Visualizing and Understanding CNNs

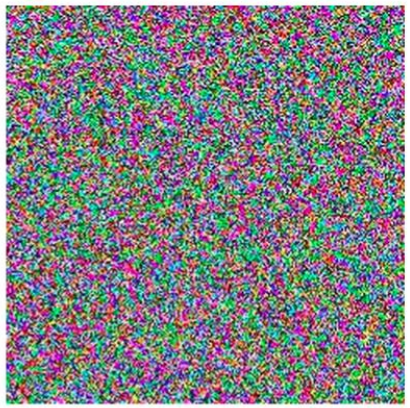
Visualizing CNNs



DeConvnet (Zeiler and Fergus, 2013)

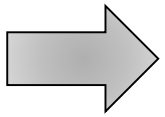


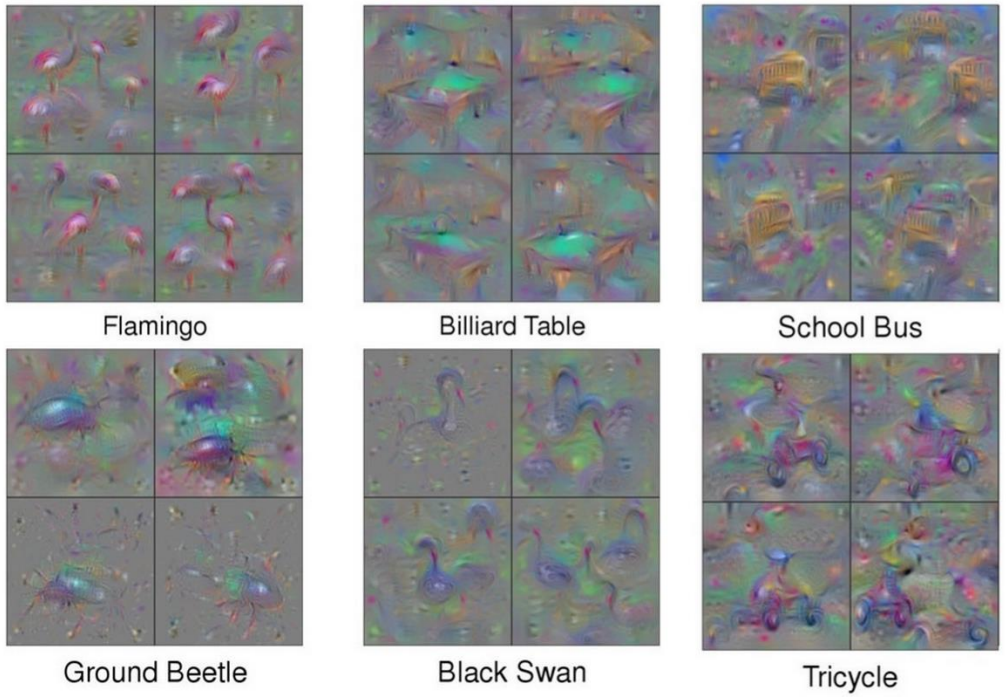
Visualizing CNNs



$$\arg\max_I \boxed{S_c(I)} - \lambda \|I\|_2^2$$

Score for class c
(before Softmax)





Yosinski et al.(2015)

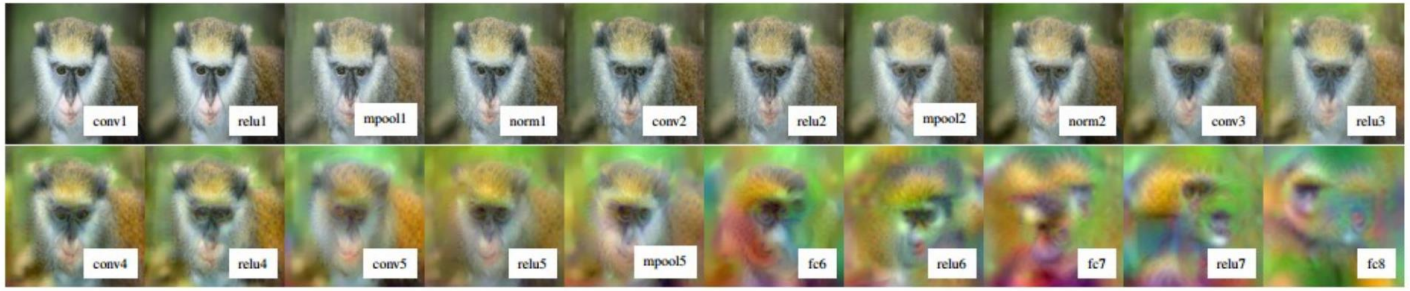
Reconstruction from Features

- Find an image such that
  - Its code is similar to a given code and it looks natural

$$\mathbf{x}^* = \underset{\mathbf{x} \in \mathbb{R}^{H \times W \times C}}{\operatorname{argmin}} \ell(\Phi(\mathbf{x}), \Phi_0) + \lambda \mathcal{R}(\mathbf{x})$$

$$\ell(\Phi(\mathbf{x}), \Phi_0) = \|\Phi(\mathbf{x}) - \Phi_0\|^2$$

Reconstructions from intermediate layers



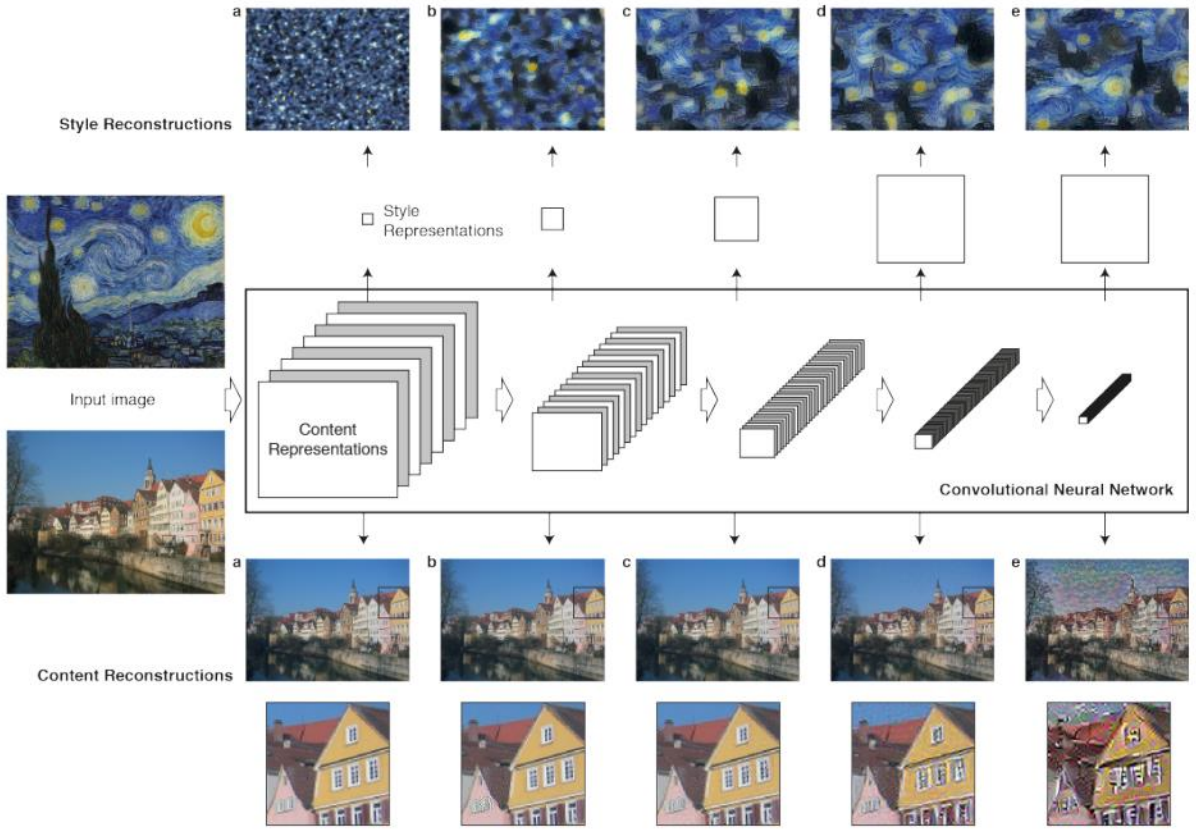
Deep Dream

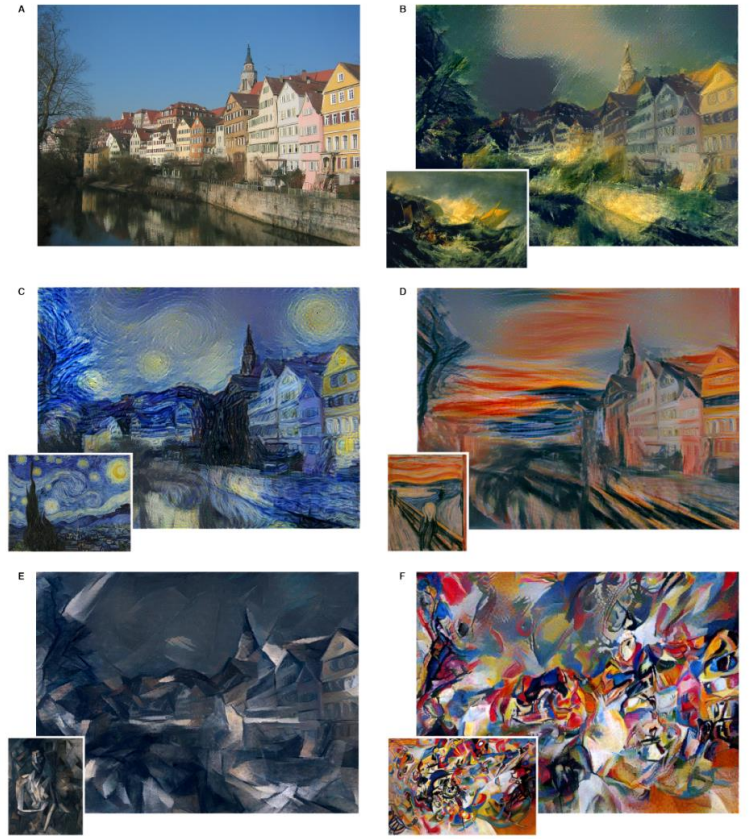
Deep Dream

Inceptionism(Google, 2015)

Neural Style

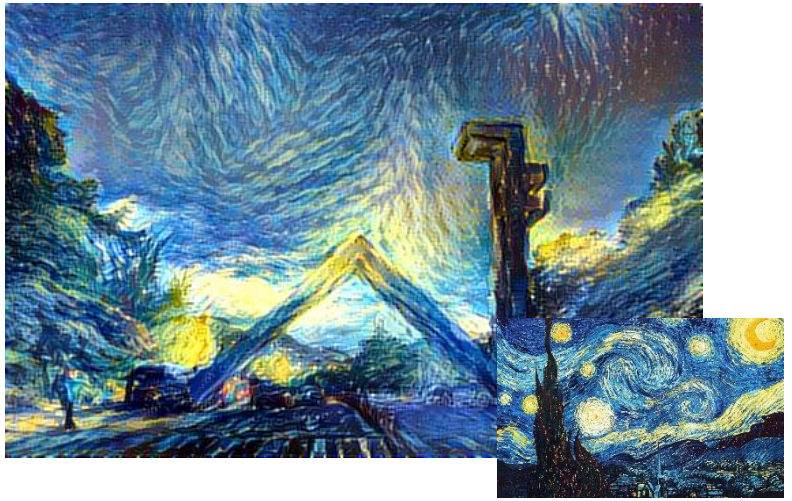
- Separating ‘Style’ from ‘Contents’ of the image.

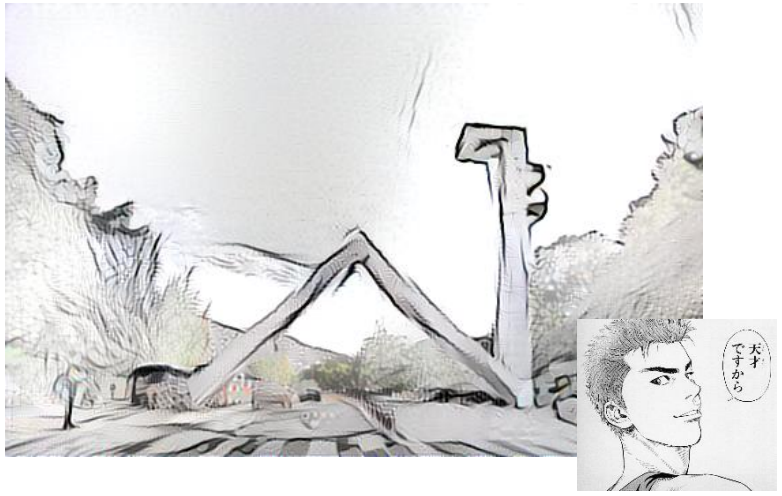


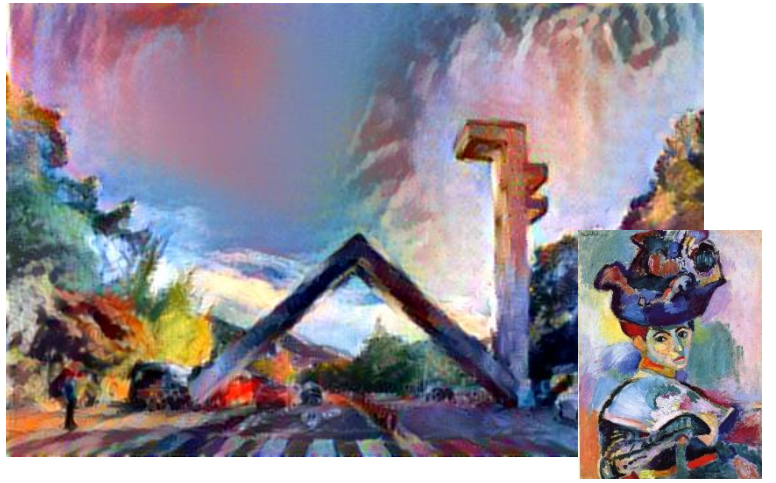


A Neural Algorithm of Artistic Style(Gatys el al., 2015)

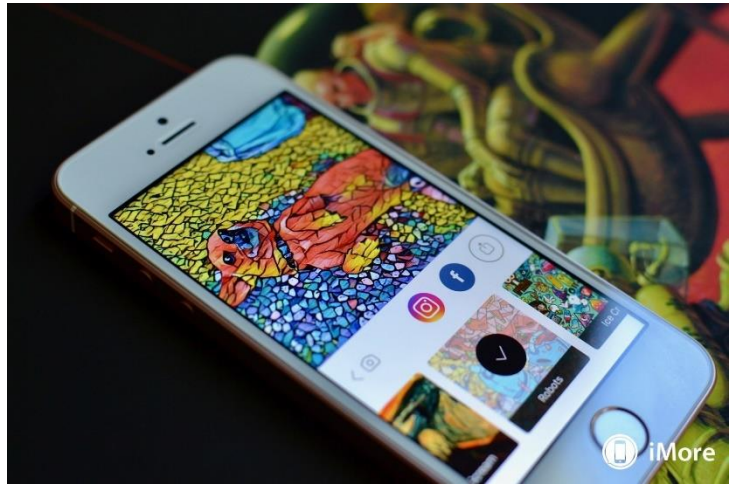
Neural Style







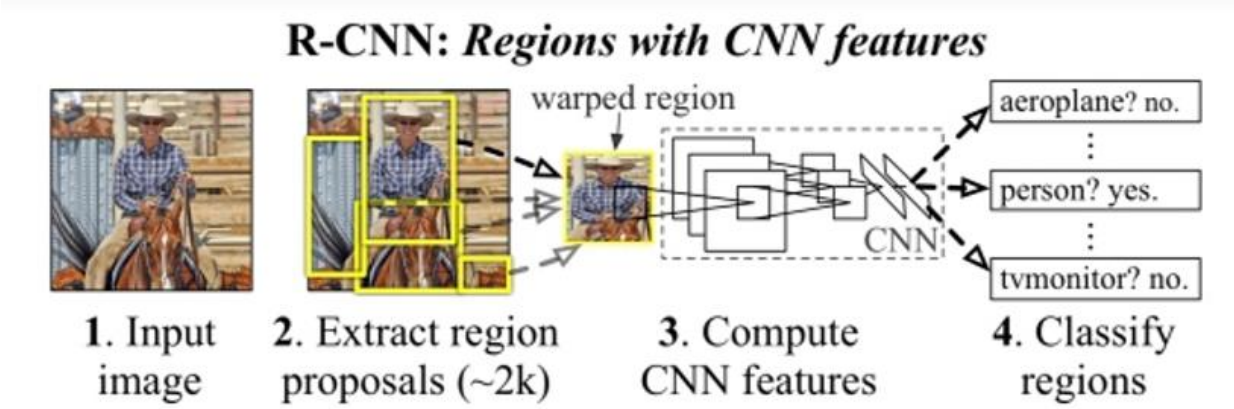
Deepart.io and Prisma

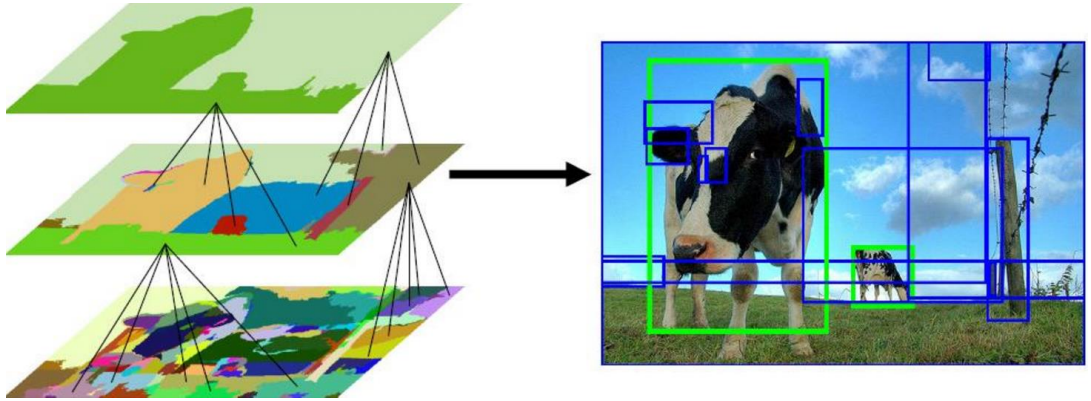


Convolutional Neural Network for Object Detection

R-CNN

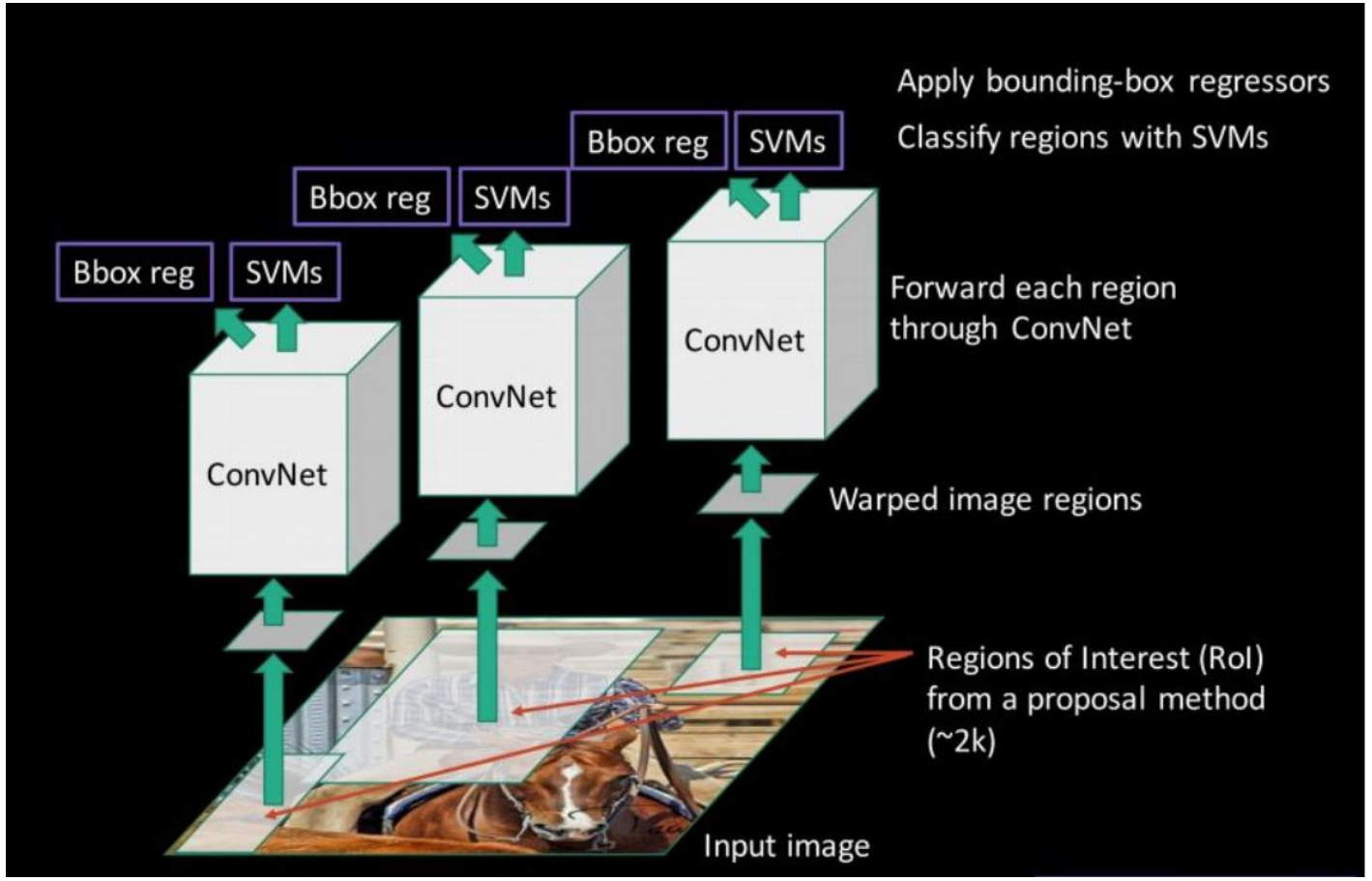
- Multi-phase Architecture for Object Detection
  - 1)Generate category-independent region proposals
  - 2)Extract a fixed length feature vector from CNN
  - 3)Classify region using class-specific linear SVM





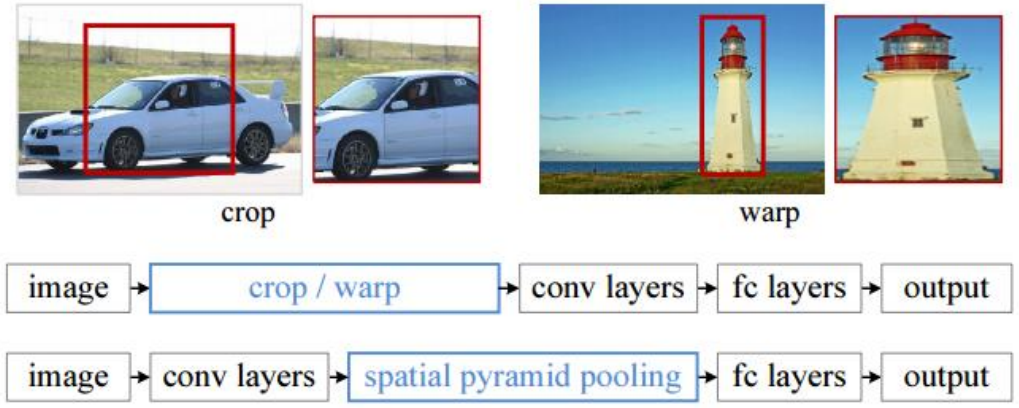
R-CNN

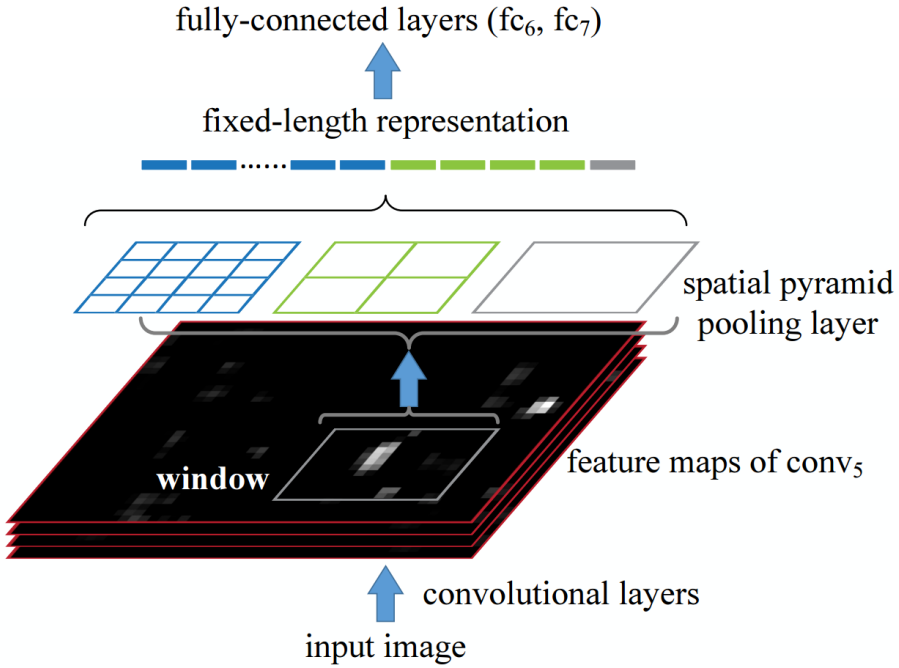
- Multi-phase Architecture for Object Detection



SPP-net

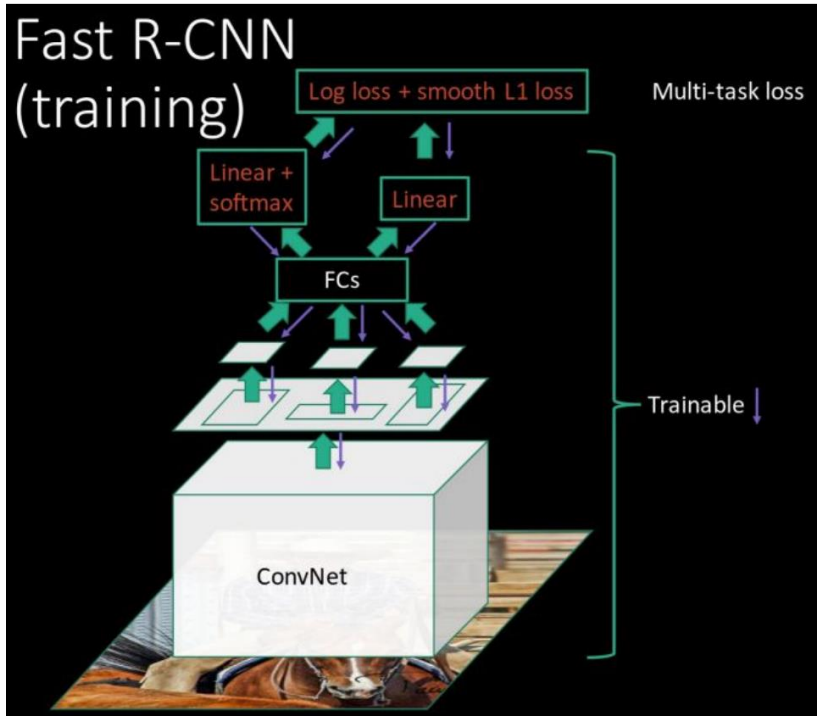
- Extraction of Features of From Candidate Window
  - Not applying convent for every candidate window in the input image, extract fixed representation from the feature map using SPP
  - It is still multi-phase architecture

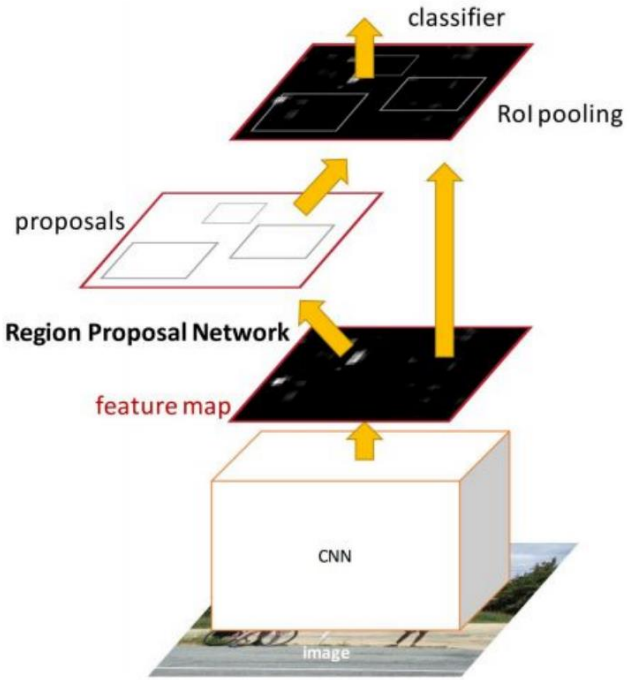




Fast R-CNN and Faster R-CNN

- Using Shared Feature Maps
  - Generate region proposal from either image or feature maps





	R-CNN	Fast R-CNN	Faster R-CNN
Test time per image (with proposals)	50 seconds	2 seconds	**0.2 seconds**
(Speedup)	1x	25x	**250x**
mAP (VOC 2007)	66.0	**66.9**	**66.9**

Convolutional Neural Network for Object Segmentation

Problem of Semantic Segmentation

- Level of Problem Difficulty

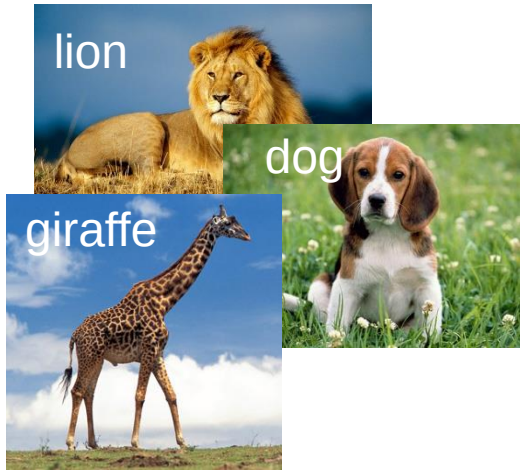
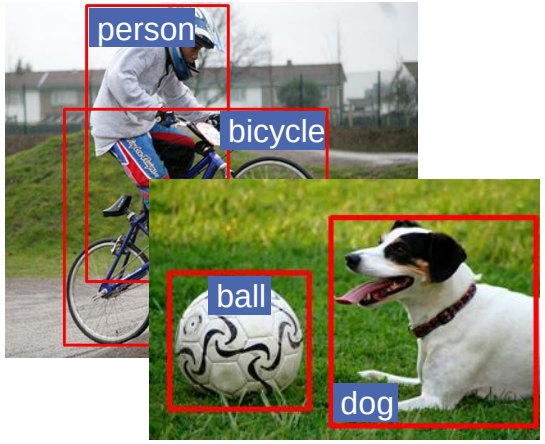


Image Classification



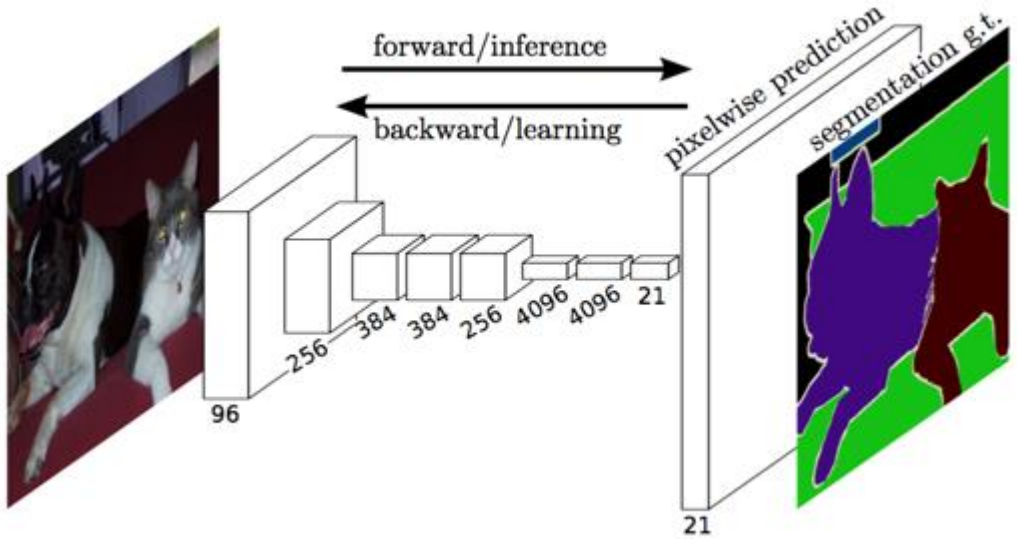
Object Detection

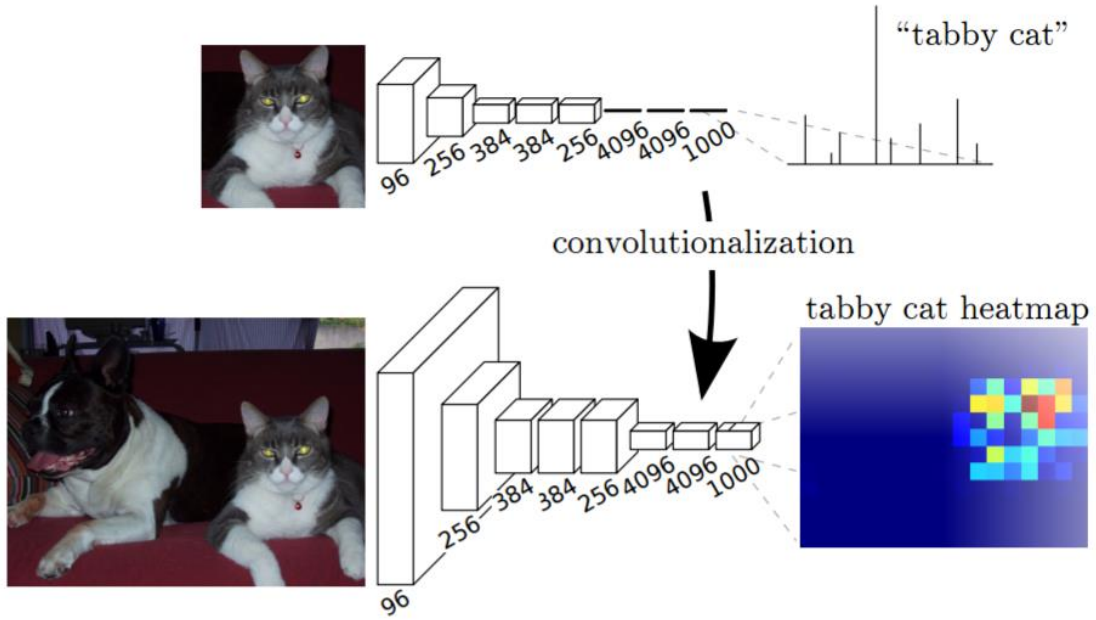
Semantic Segmentation

(a) Image (b) G.T.

Fully Convolutional Neural Network (FCN)

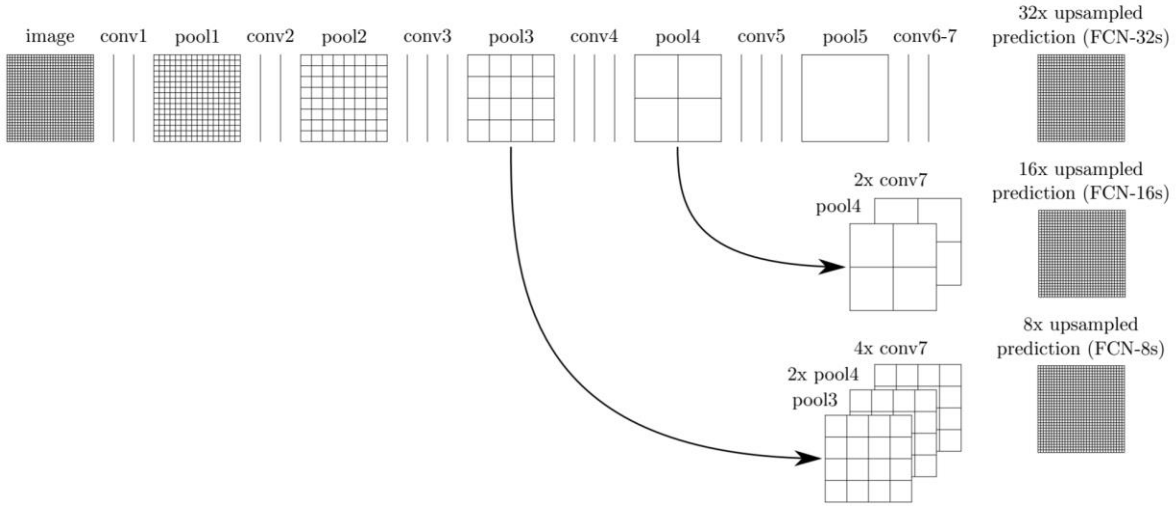
- End-to-end CNN for Semantic Segmentation

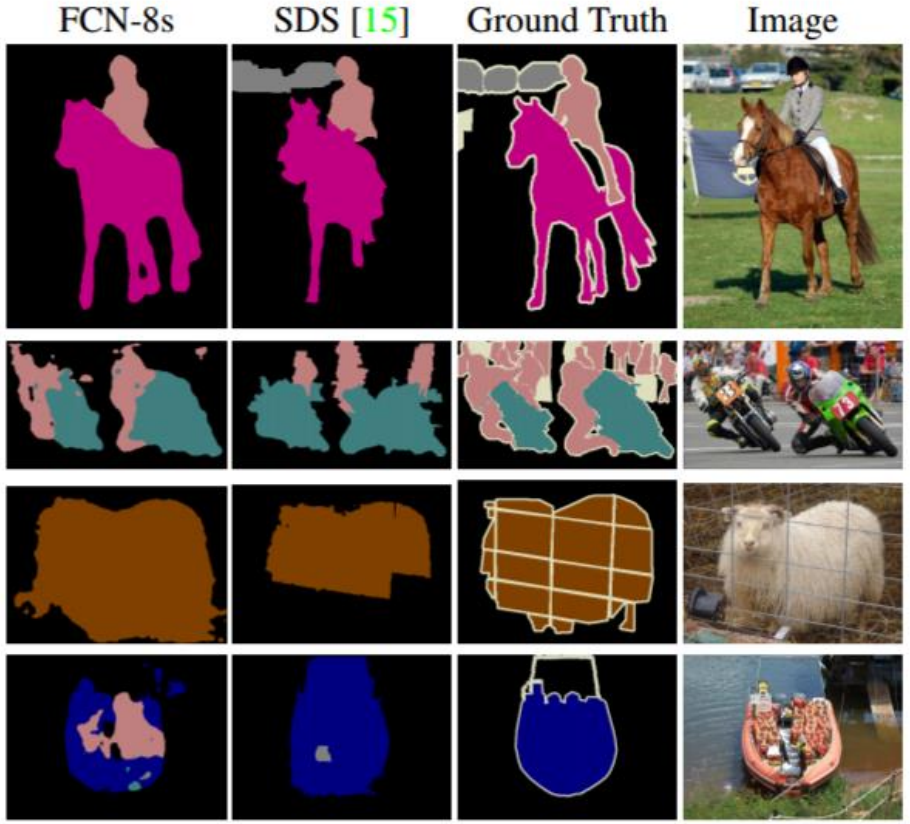




Fully Convolutional Neural Network (FCN)

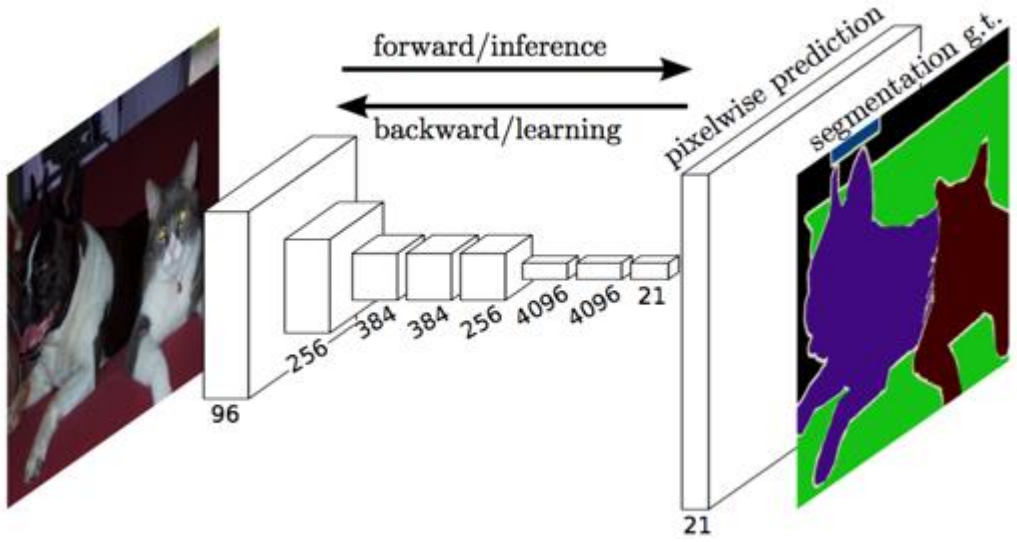
- End-to-end CNN for Semantic Segmentation

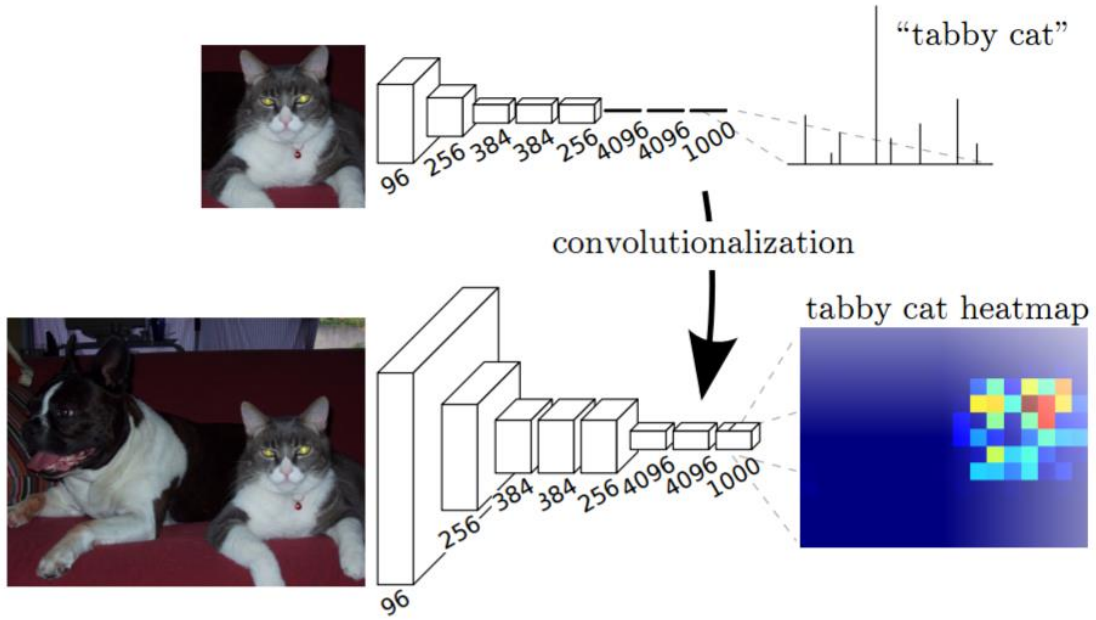




Fully Convolutional Neural Network (FCN)

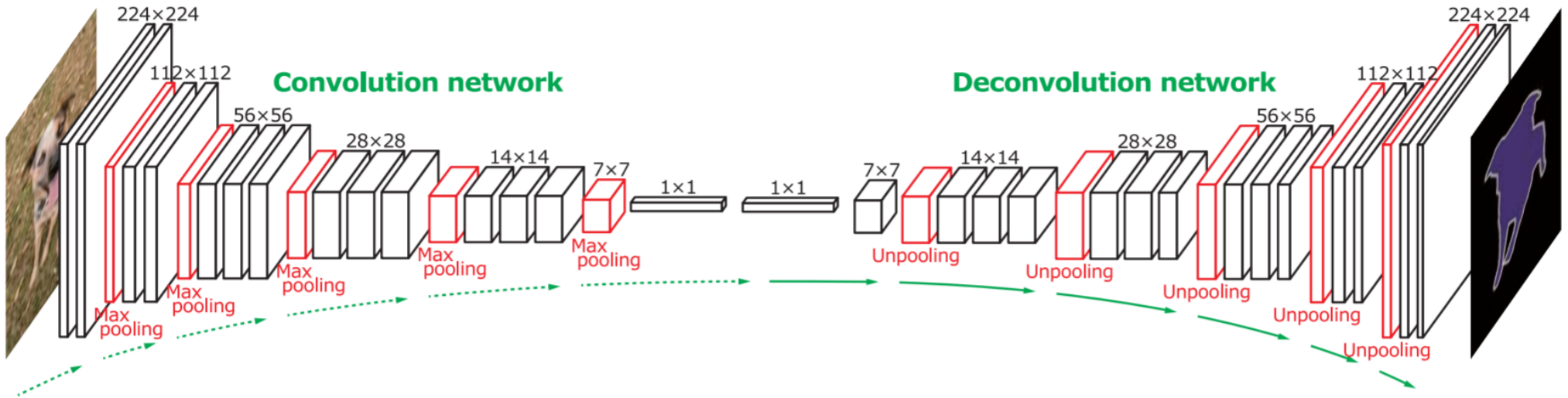
- End-to-end CNN for Semantic Segmentation





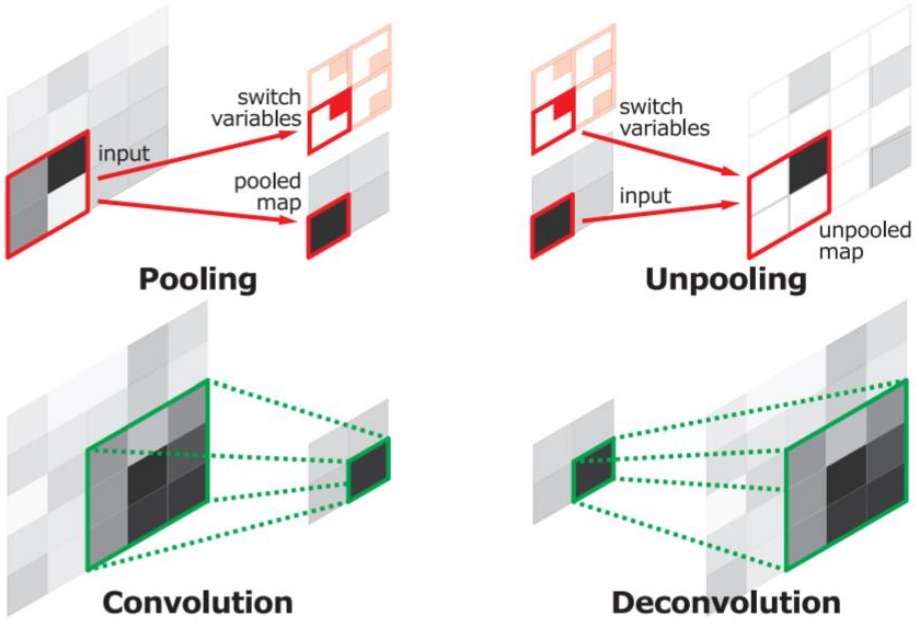
DeconvNet

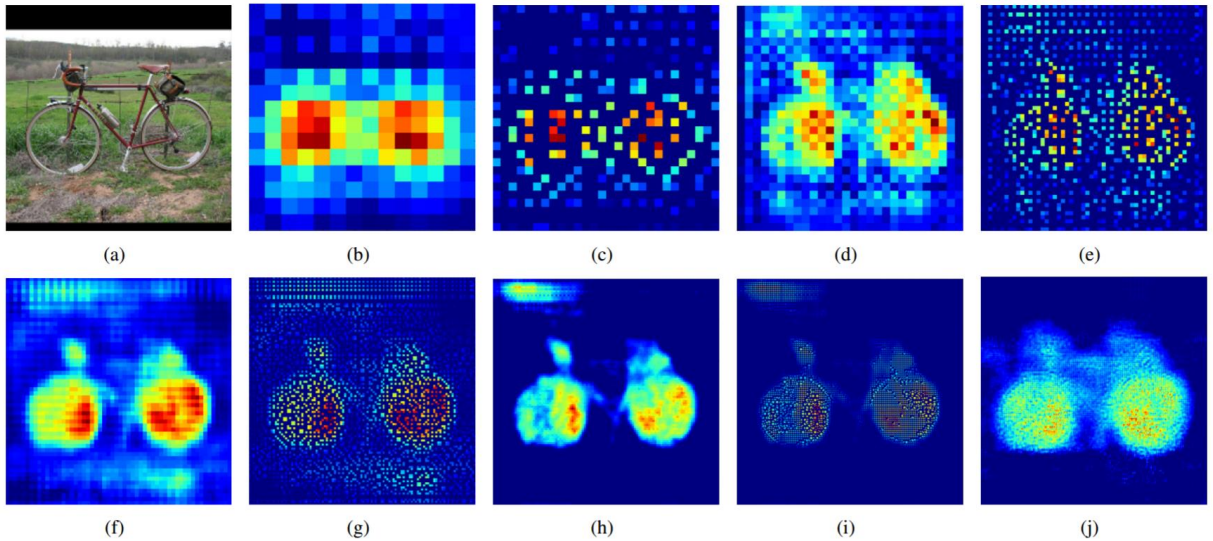
- Deconvolutional Neural Network with UnPooling Operation



DeconvNet

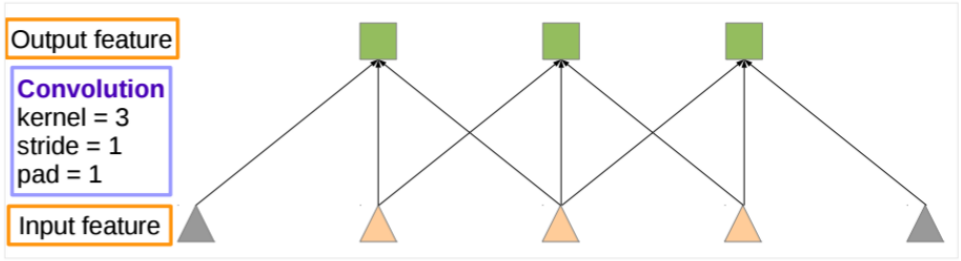
- Deconvolutional Neural Network with UnPooling Operation



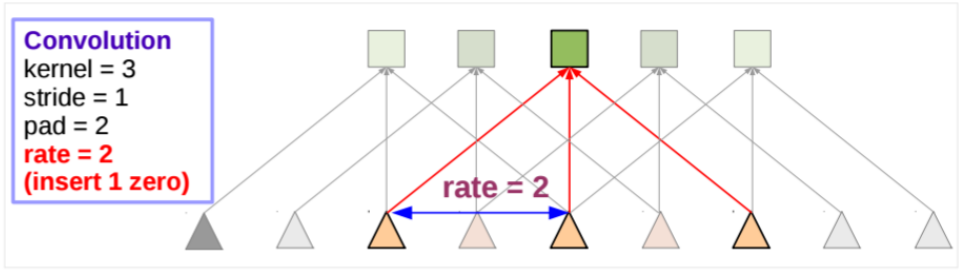


DeepLab

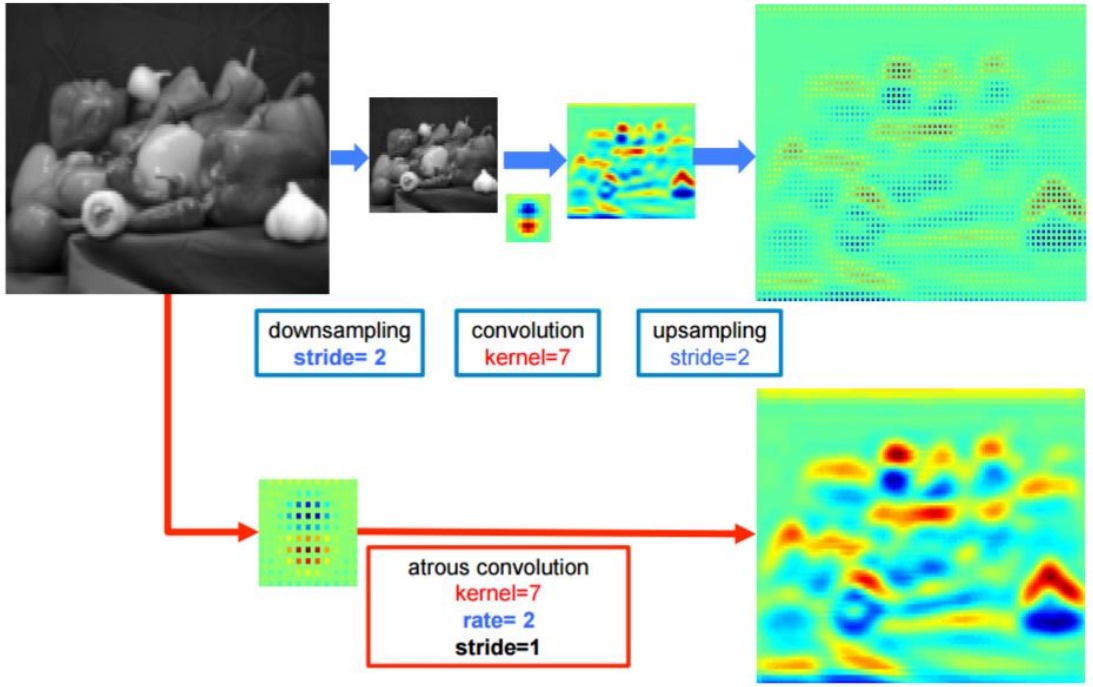
- Enlarging Field of View using Atrous Convolution



(a) Sparse feature extraction

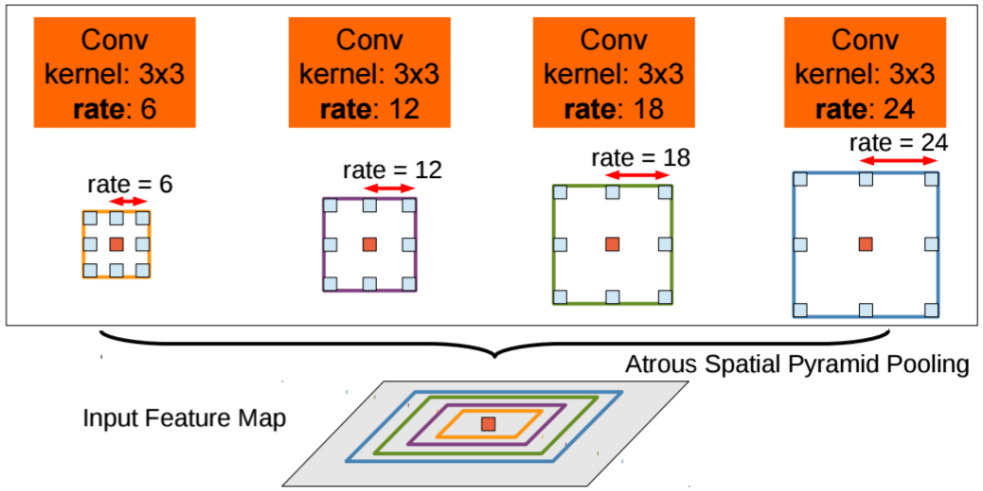


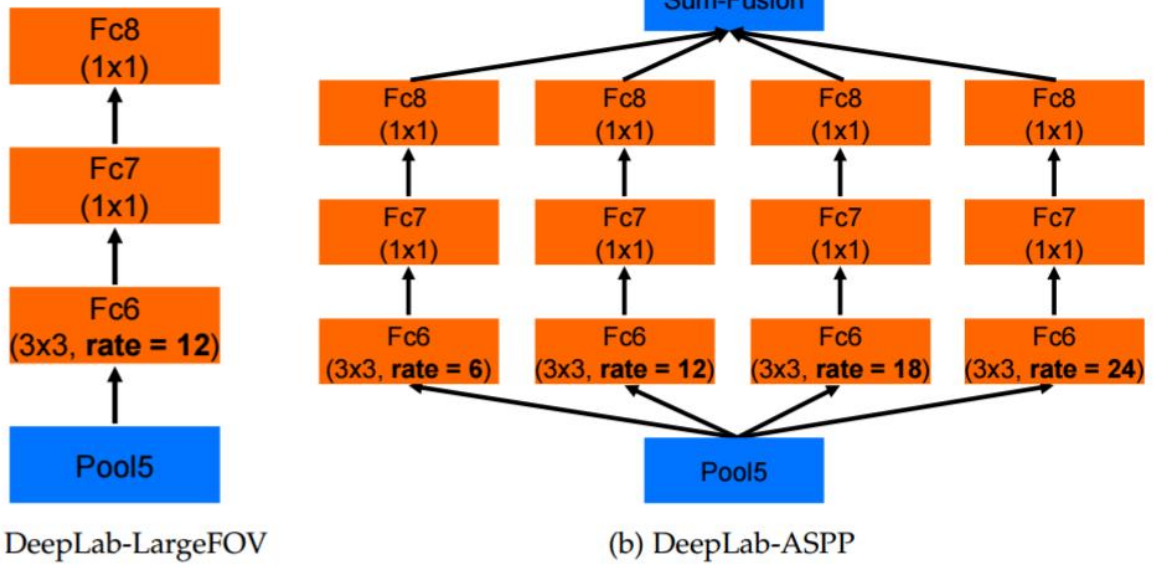
(b) Dense feature extraction



DeepLab

- Using Multiple 'Rate' for Multi-scale Object Segmentation

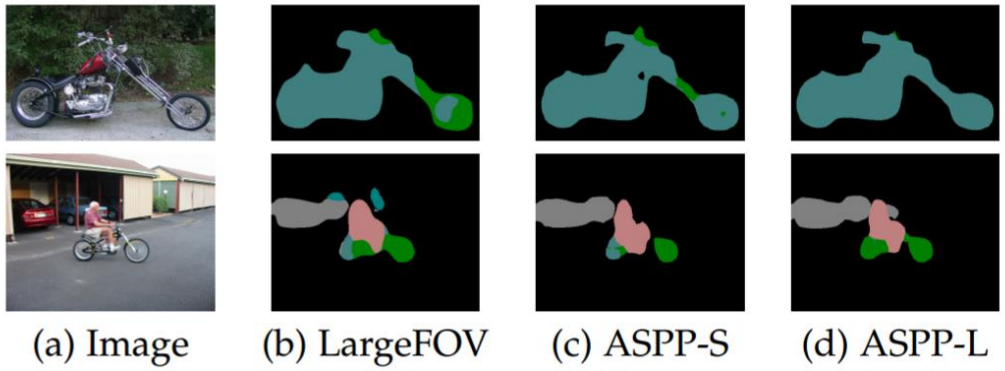


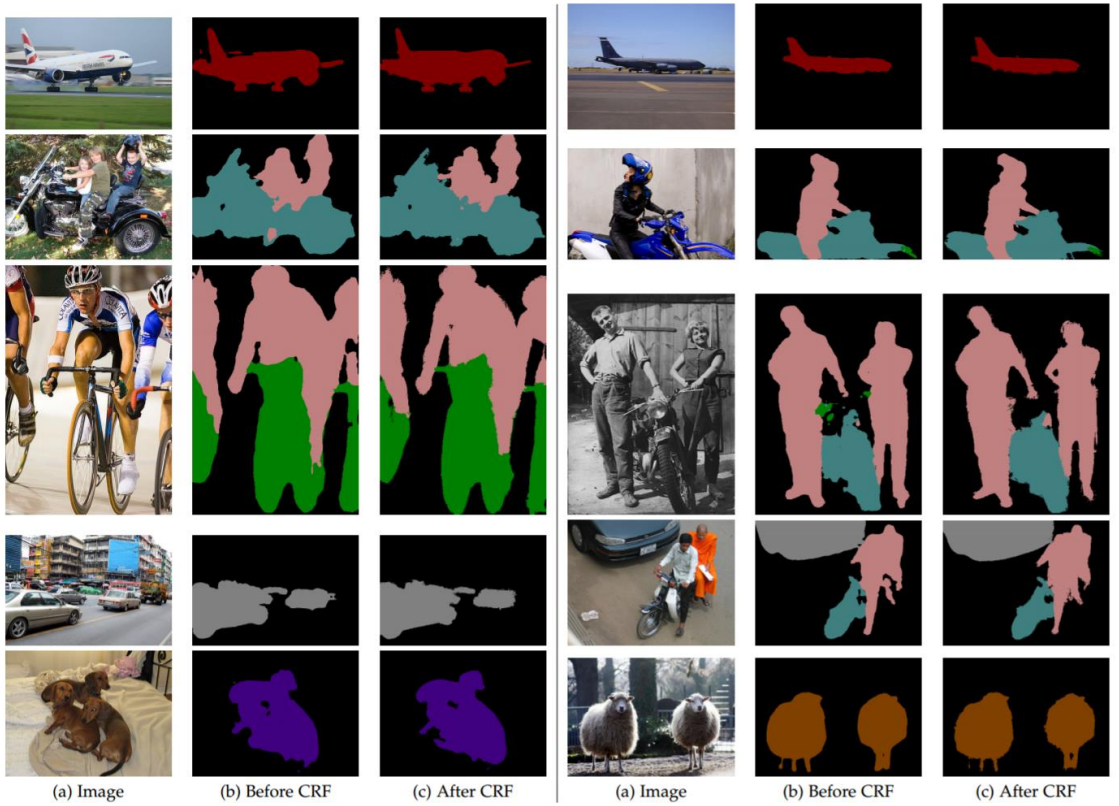


(a) DeepLab-LargeFOV (b) DeepLab-ASPP

DeepLab

- Shows State-of-the-art Performance on VOC 2012 after CRF

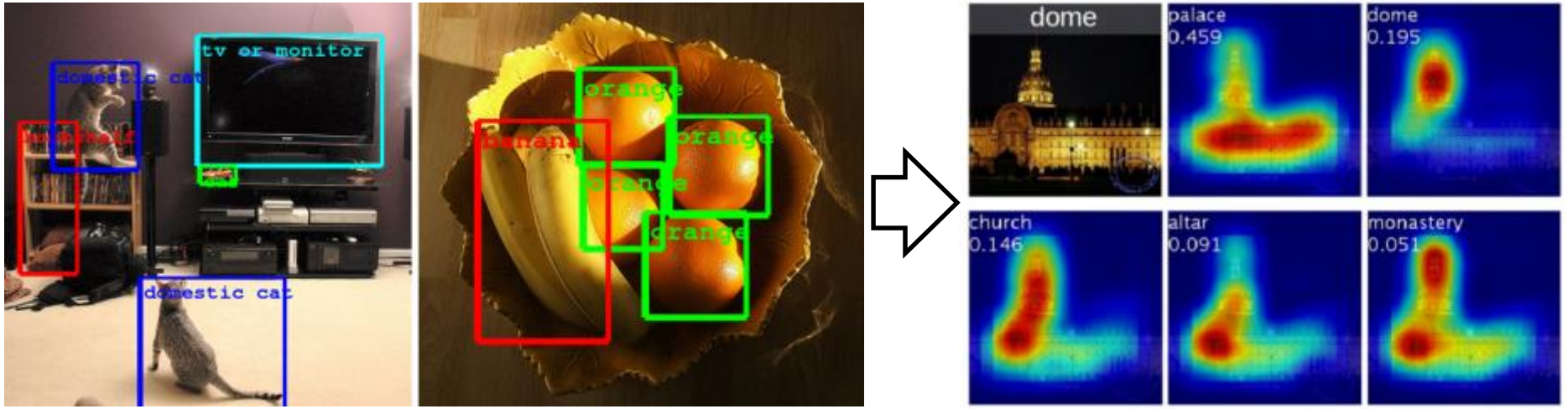




Convolutional Neural Network for Weakly-supervised Localization

Weakly Supervised Learing

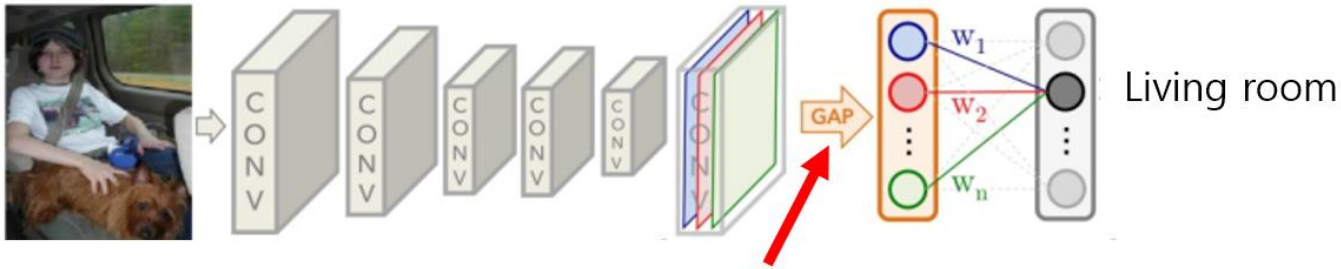
- Problem Definition
  - What if we have only image-level annotations while we need to precisely locate the region of target object?



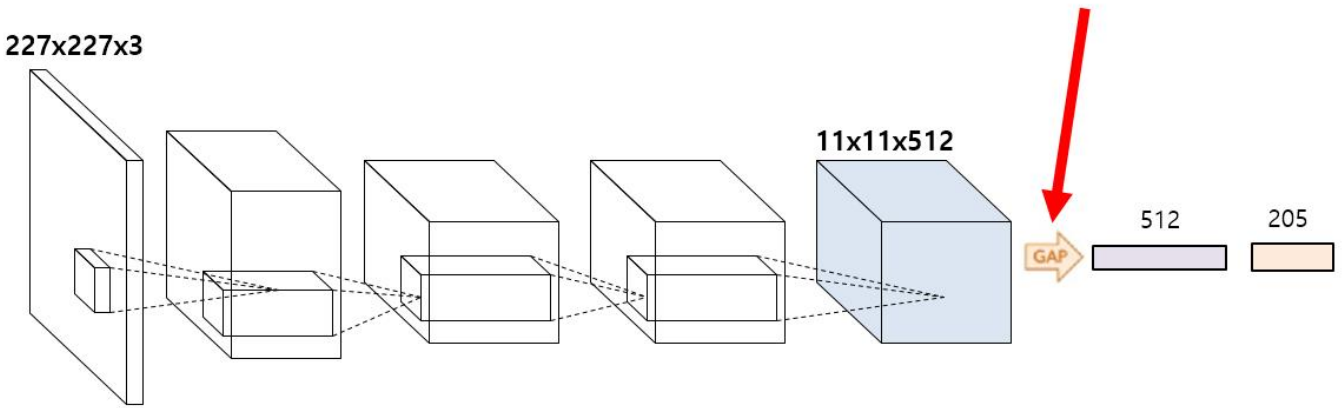
Weakly Supervised Learning

- Class Activation Map Approach

AlexNet+GAP+places205



11x11 Avg Pooling: Global Average Pooling (GAP)




```
('data', (10, 3, 227, 227))
('conv1', (10, 96, 55, 55))
('pool1', (10, 96, 27, 27))
('norm1', (10, 96, 27, 27))
('conv2', (10, 256, 27, 27))
('pool2', (10, 256, 13, 13))
('norm2', (10, 256, 13, 13))
('conv3', (10, 384, 13, 13))
('conv4', (10, 384, 13, 13))
('conv5', (10, 384, 13, 13))
('pool5', (10, 384, 11, 11))
('conv6', (10, 512, 11, 11))
('conv7', (10, 512, 11, 11))
('pool8_global', (10, 512, 1, 1))
('fc9', (10, 205))
('prob', (10, 205))
→ alexnetplusCAM_places205
```

Weakly Supervised Learning

- Class Activation Map Approach
  - Identify important image regions by projecting back the weights of output layer to convolutional feature maps.
  - CAMs can be generated for each class in single image.
  - Regions for each categories are different in given image.

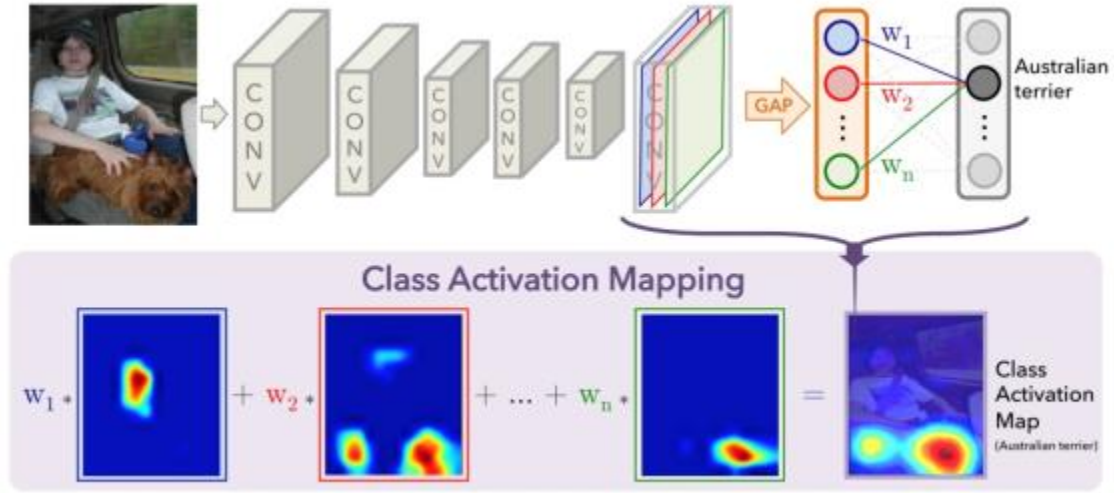


Figure 2. Class Activation Mapping: the predicted class score is mapped back to the previous convolutional layer to generate the class activation maps (CAMs). The CAM highlights the class-specific discriminative regions.

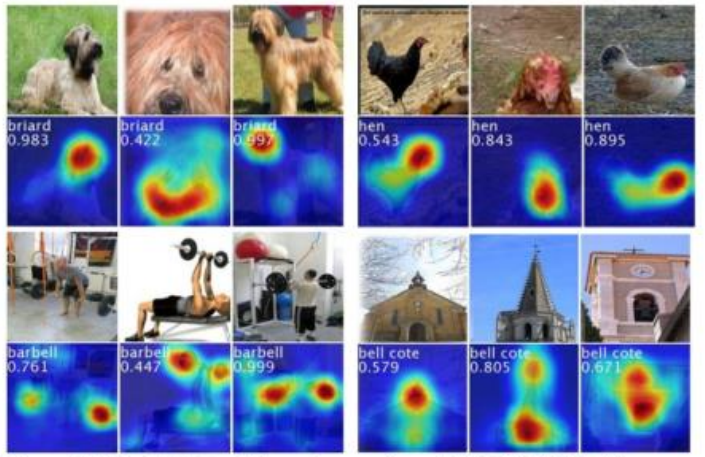
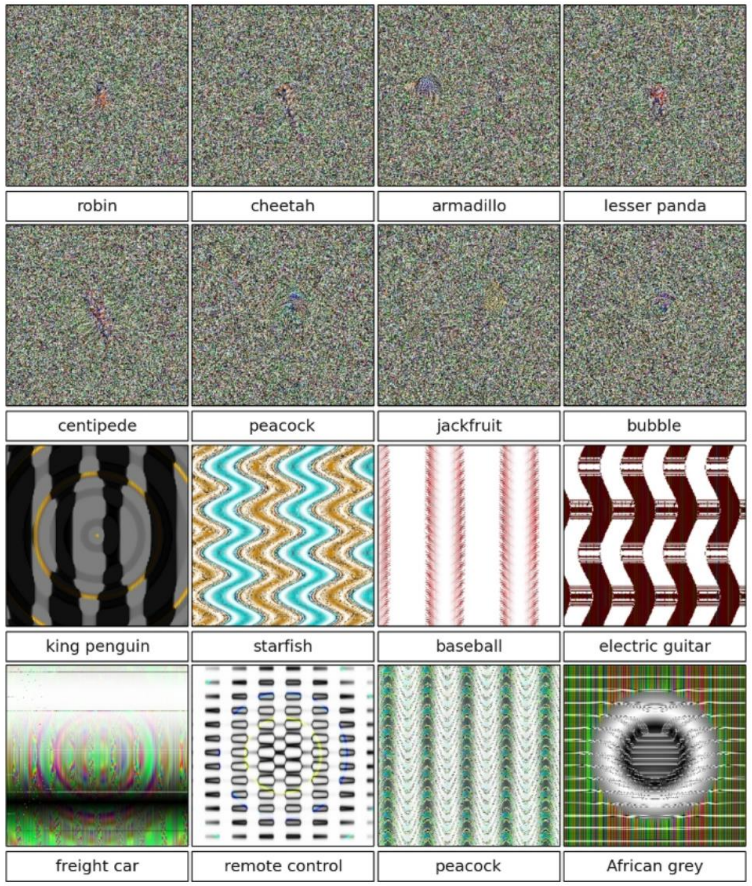


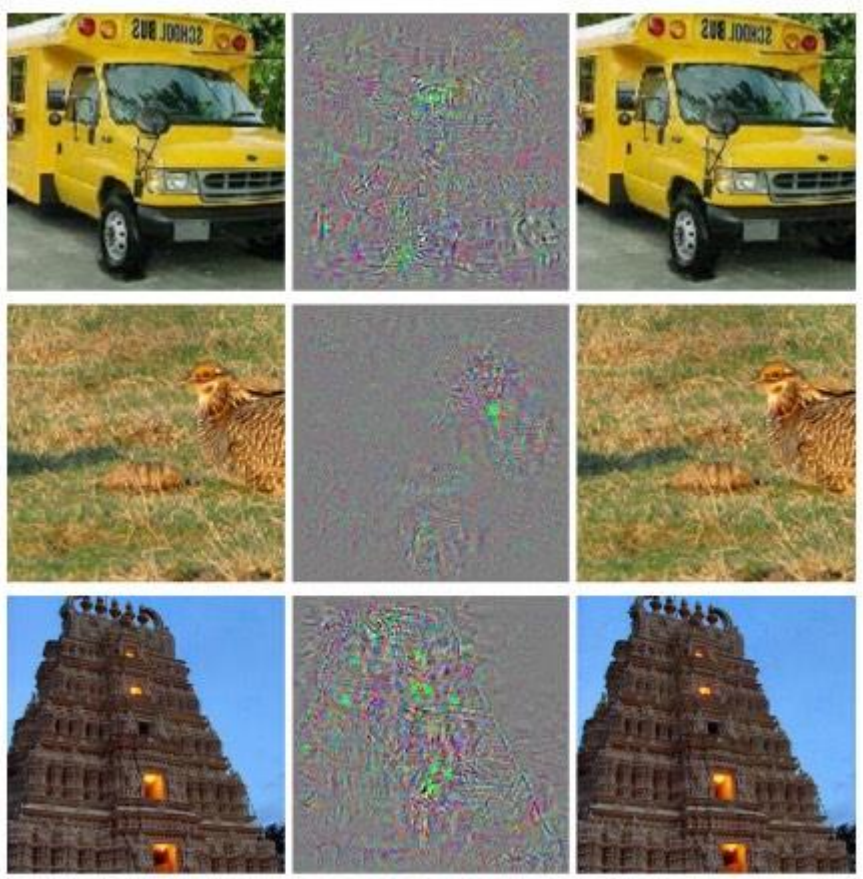
Figure 3. The CAMs of four classes from ILSVRC [20]. The maps highlight the discriminative image regions used for image classification e.g., the head of the animal for *briard* and *hen*, the plates in *barbell*, and the bell in *bell cote*.

Unsupervised Learning for Visual Understanding

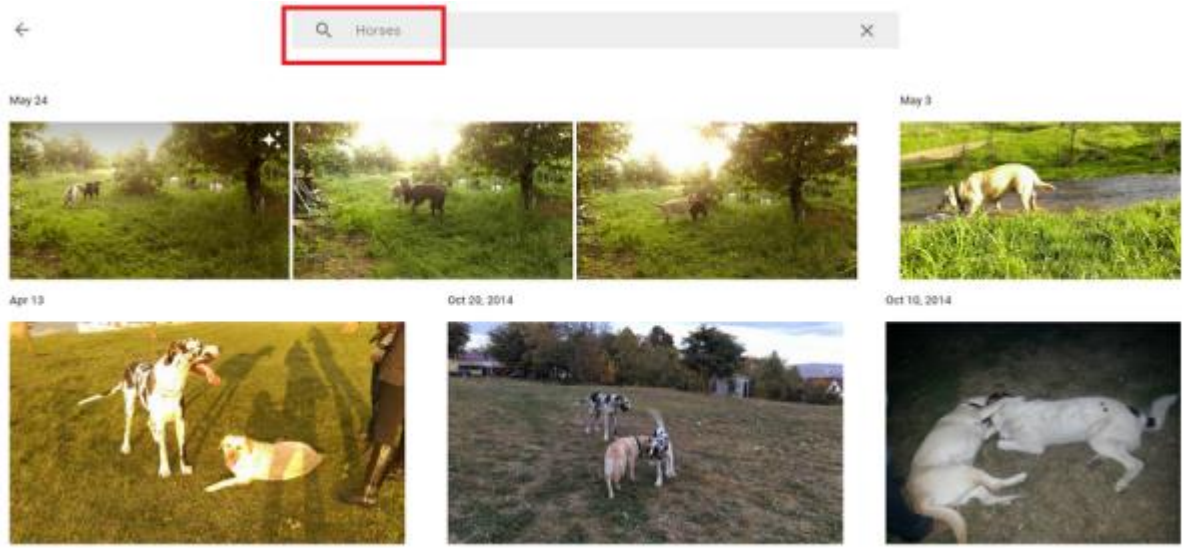
Does DNN Really See the World as We Do?

- DNNs are easily fooled





Does DNN Really See the World as We Do?



Puppy or Bagel?

Puppy or Bagel? – Computer Guess

Generative Models

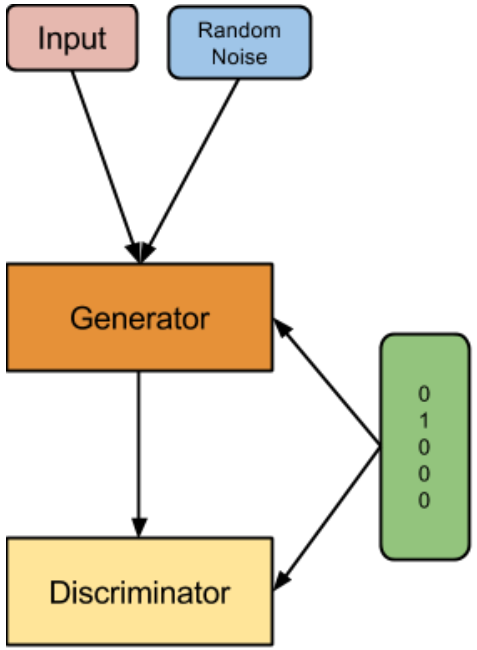
- Definition
  - A model for randomly generating observable data values, typically given some hidden parameters
  - “What I cannot created, I do not understand.”

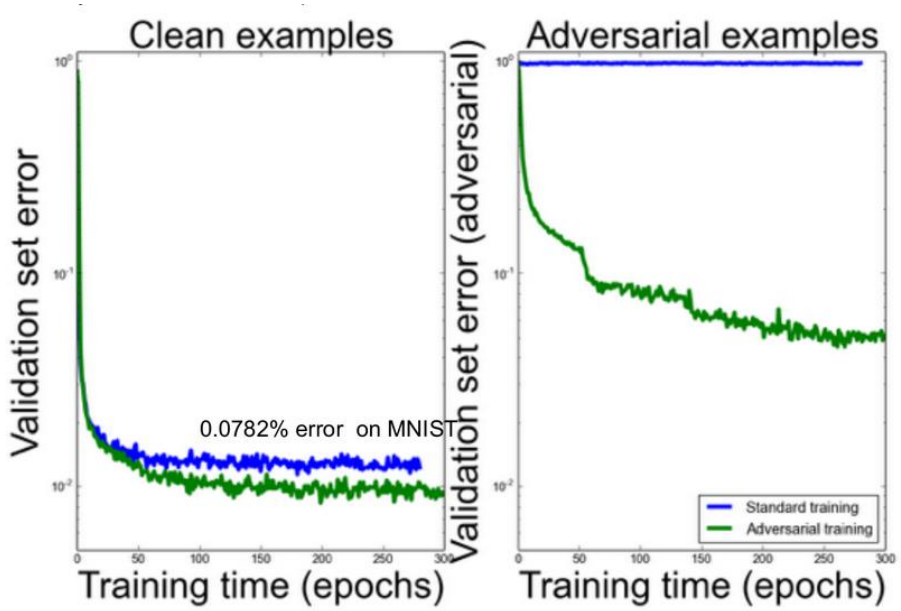
- Examples
  - We can distinguish Chinese from Japanese but we cannot speak them.
  - Kids can read numbers and characters but cannot write down them.

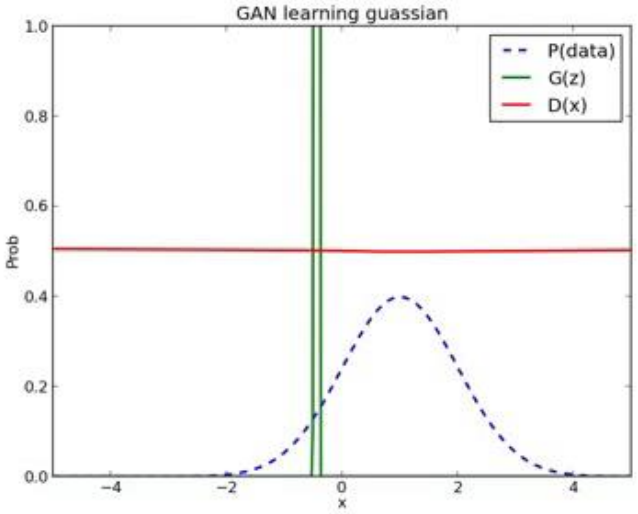
- Deep Generative Models
  - Auto-Encoders : AE, Denoising AE, Sparse AE, Variational AE
  - Generative Adversarial Networks : GAN, DCGAN, InfoGAN,

Generative Adversarial Networks (NIPS 2014)

- Generator vs Discriminator
  - Simpler generative model than MCMC-based or variational inference-based models.

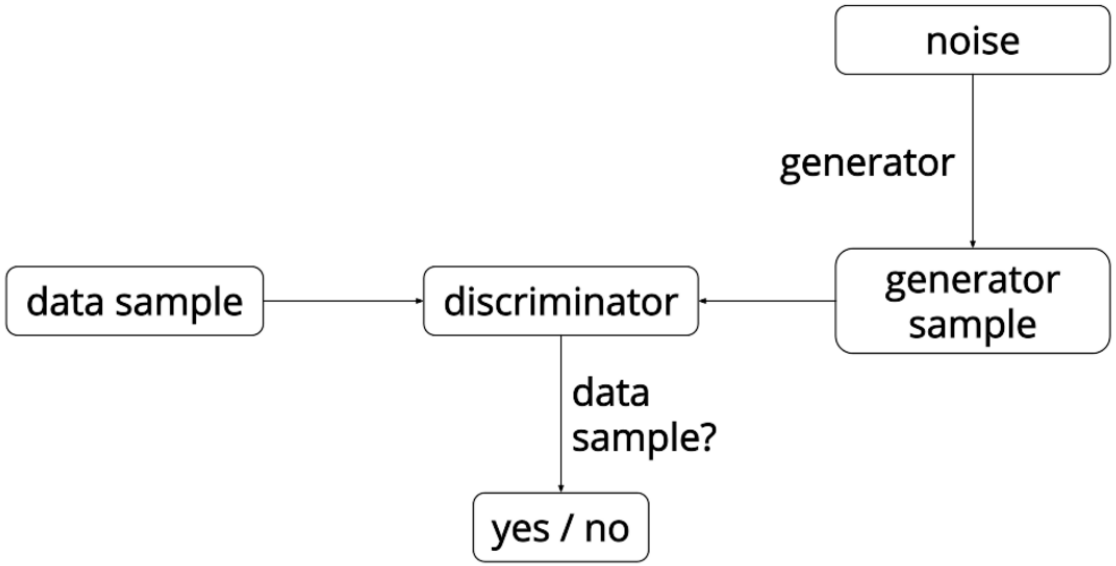






Generative Adversarial Networks (NIPS 2014)

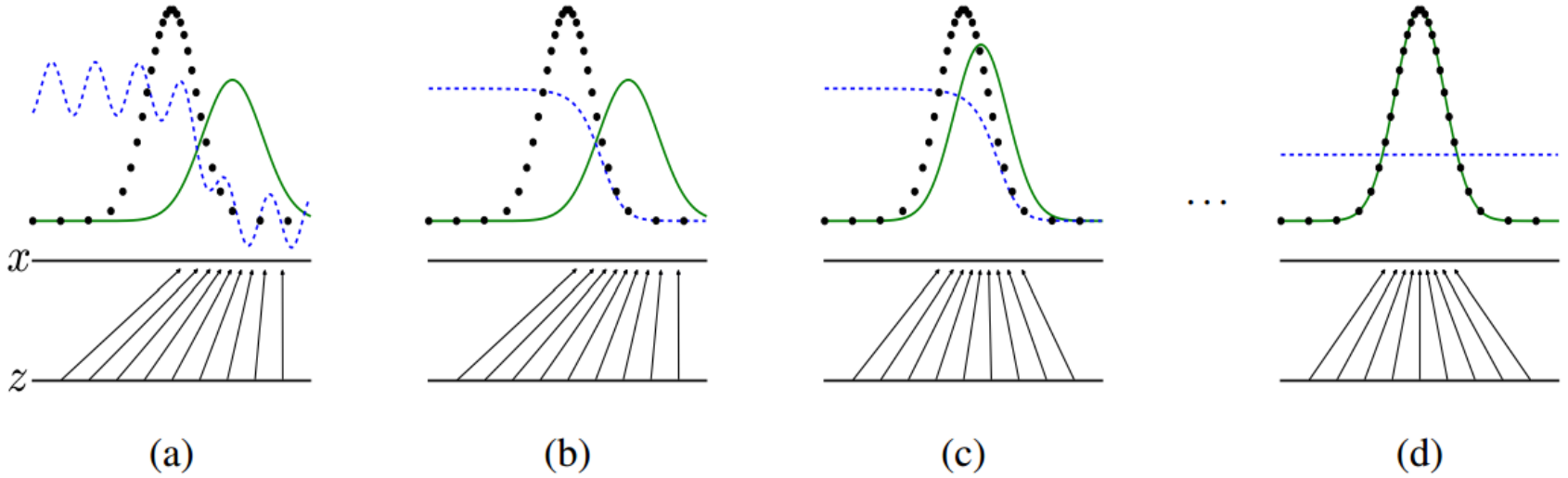
- 1D Example



Generative Adversarial Networks (NIPS 2014)

- MinMax Problem for Generator and Discriminator
  - Minimize for G, maxize for D.
  - D discriminate whether the input is from data or generative model.
  - G generate sample from

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$



Generative Adversarial Networks (NIPS 2014)

- The Algorithm

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

for k steps **do**

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

end for

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

Training D to convergence is expensive.
Therefore, train D k step and train G slowly.

In the early phase D is confident and gradient is small
Instead we can max D(G(z))

Generative Adversarial Networks (NIPS 2014)

- Theoretical Results
  - Optimality for D

Proposition 1. *For G fixed, the optimal discriminator D is*

$$D_G^*(\boldsymbol{x}) = \frac{p_{data}(\boldsymbol{x})}{p_{data}(\boldsymbol{x}) + p_g(\boldsymbol{x})}$$

  - Optimality for G

Theorem 1. *The global minimum of the virtual training criterion $C(G)$ is achieved if and only if $p_g = p_{data}$. At that point, $C(G)$ achieves the value $-\log 4$.*

  - Convergence Analysis

Proposition 2. *If G and D have enough capacity, and at each step of Algorithm 1, the discriminator is allowed to reach its optimum given G, and p_g is updated so as to improve the criterion*

$$\mathbb{E}_{\boldsymbol{x}\sim p_{data}}[\log D_G^*(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x}\sim p_g}[\log(1 - D_G^*(\boldsymbol{x}))]$$

then p_g converges to p_{data}

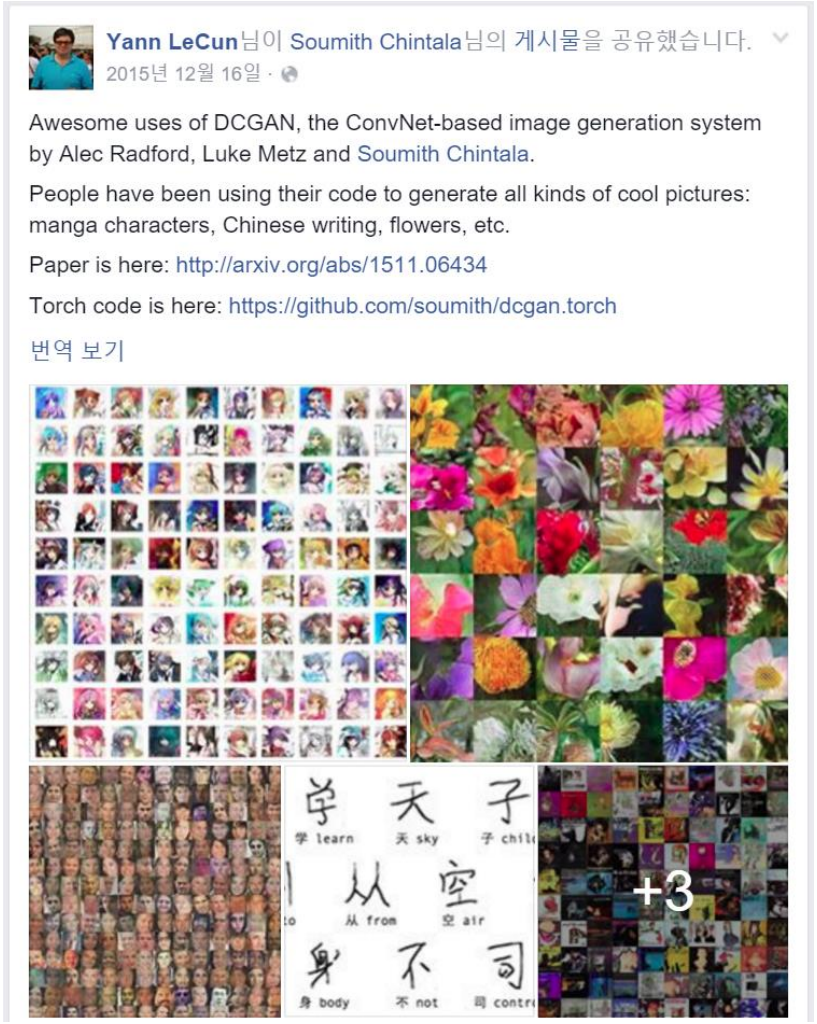
Generative Adversarial Networks (NIPS 2014)

- Experimental Results
  - D : MLP with maxout activation and dropout
  - G : MLP with ReLU and sigmoid activation

Deep Convolutional GAN (ICLR 2016)

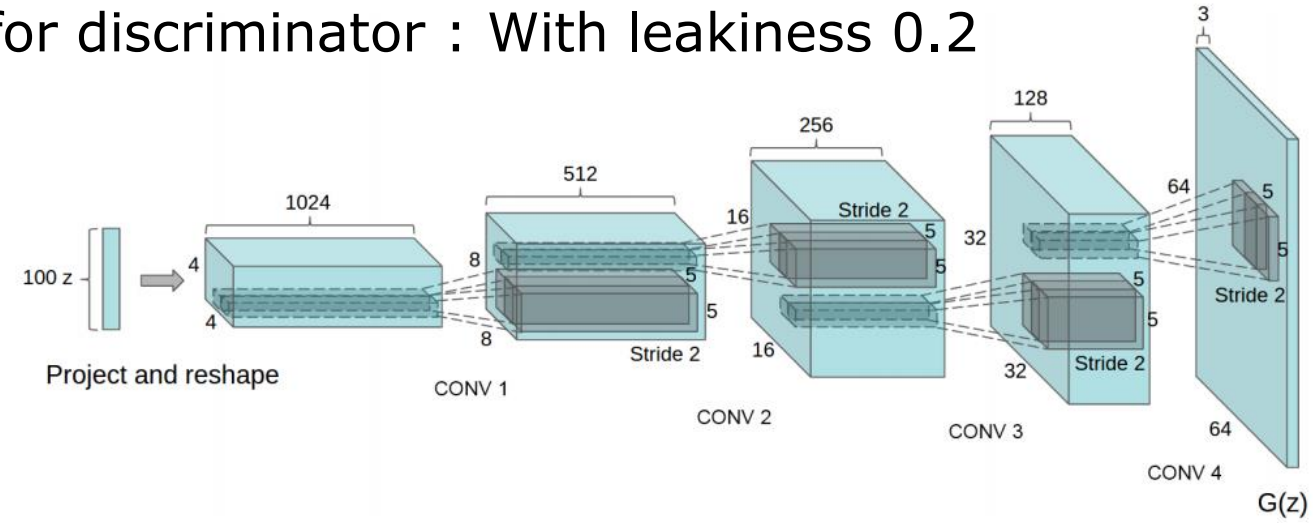
- Bridging Gap between Supervised and Unsupervised Feature Learning
  - CNNs have mainly used for supervised tasks.
  - There are unlimited images without labels.
  - CNNs are good at dealing with Images.
  - How can we used CNNs to extract good representation of images without supervision?

- There have been some image generators
  - Yes. For example GAN and its laplacian pyramid extension, VAE, RNN, DeConvNet, …
  - However, training network was unstable and the generated images were blurry or noisy.



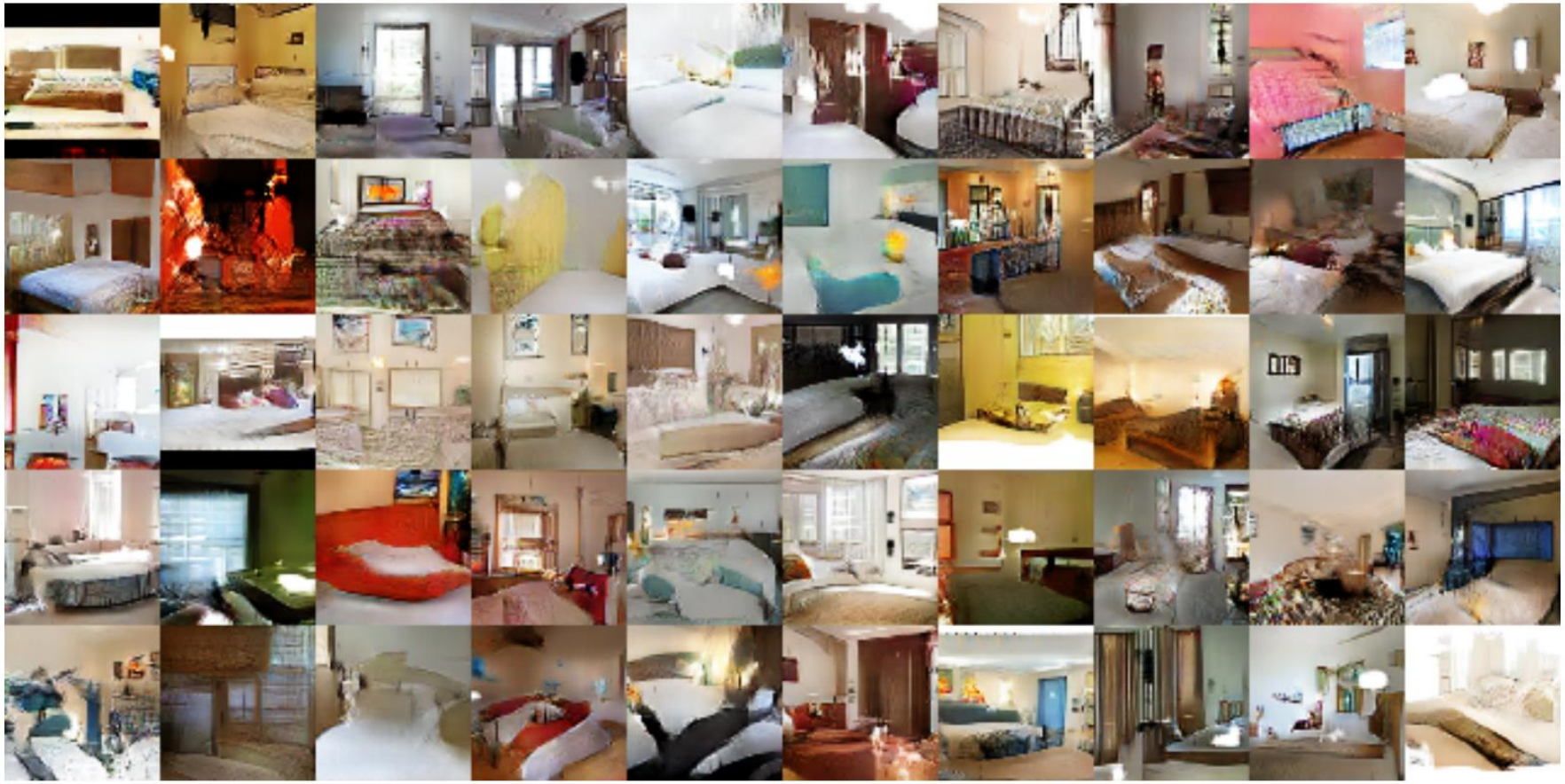
Deep Convolutional GAN (ICLR 2016)

- Recipes Making CNNs Work for GAN
  - All ConvNet : Replace pooling layer with strided convolutions.
  - No FC layers : Directly connecting last conv features to the input of generator and output of discriminator.
  - Batch normalization : Stabilizing learning process and help training deeper generator.
  - ReLU activation for generator : Except the output layer which uses Tanh
  - LeakyReLU for discriminator : With leakiness 0.2



Deep Convolutional GAN (ICLR 2016)

- Generated Samples – Large-scale Scene Understanding

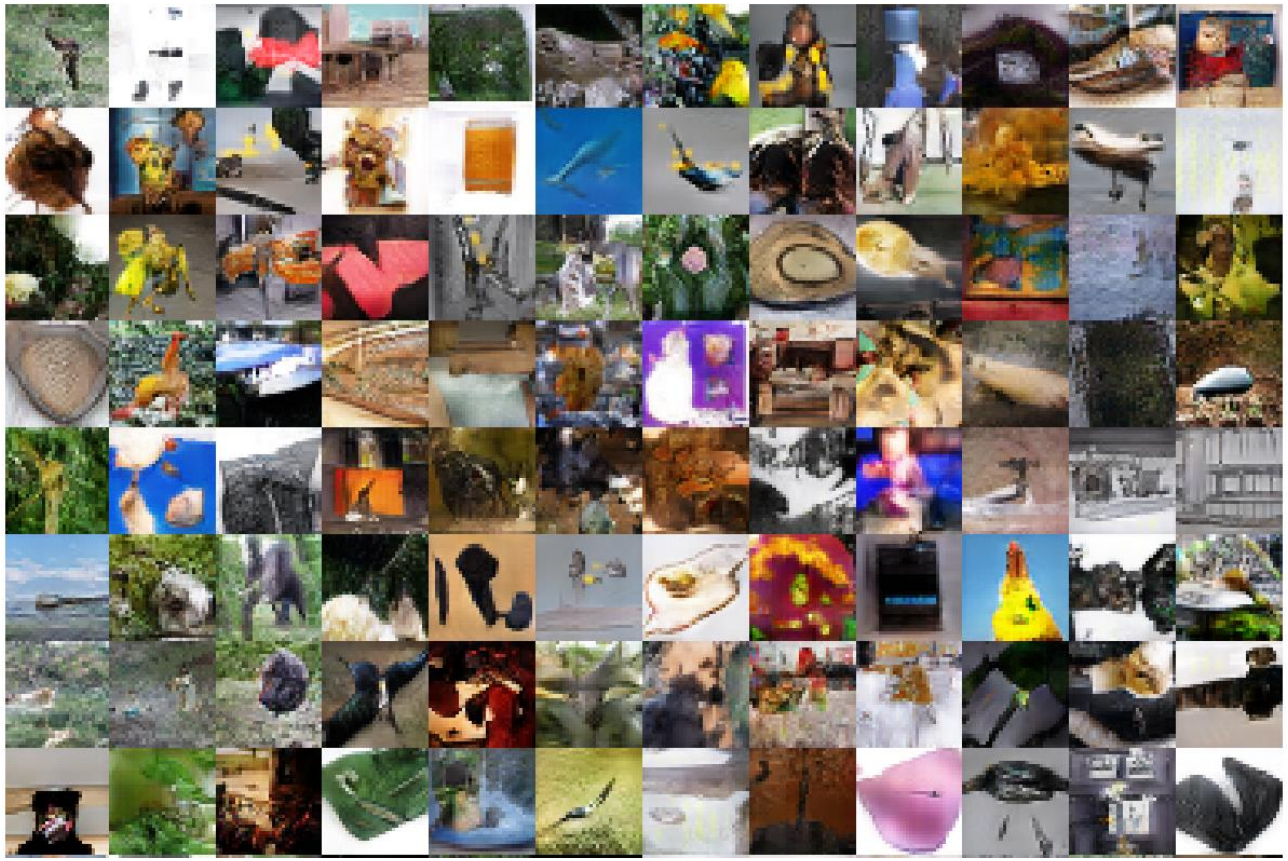


Deep Convolutional GAN (ICLR 2016)

- Generated Samples – Faces

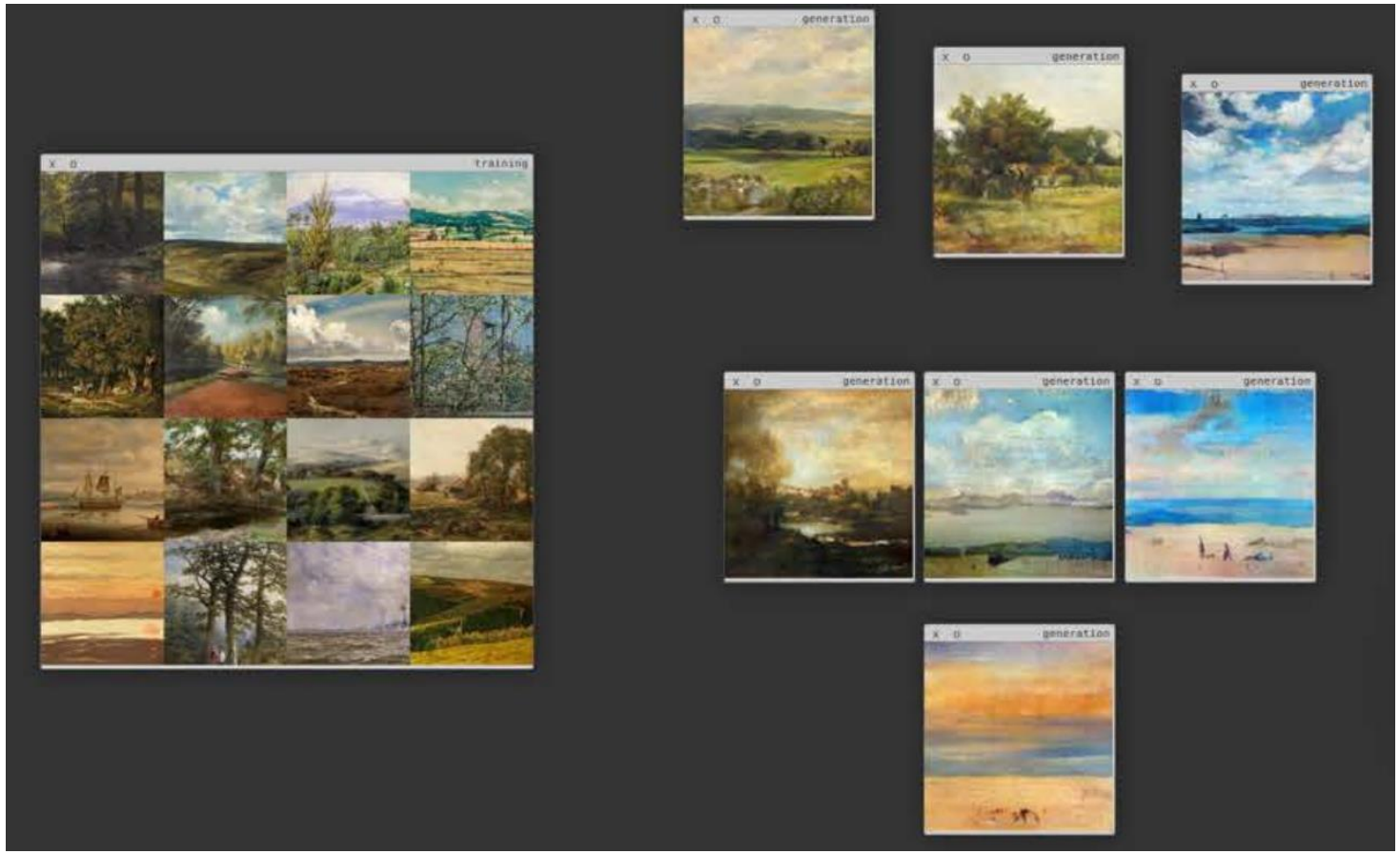
Deep Convolutional GAN (ICLR 2016)

- Generated Samples – ImageNet 1k(Trained with 32x32 center crop)



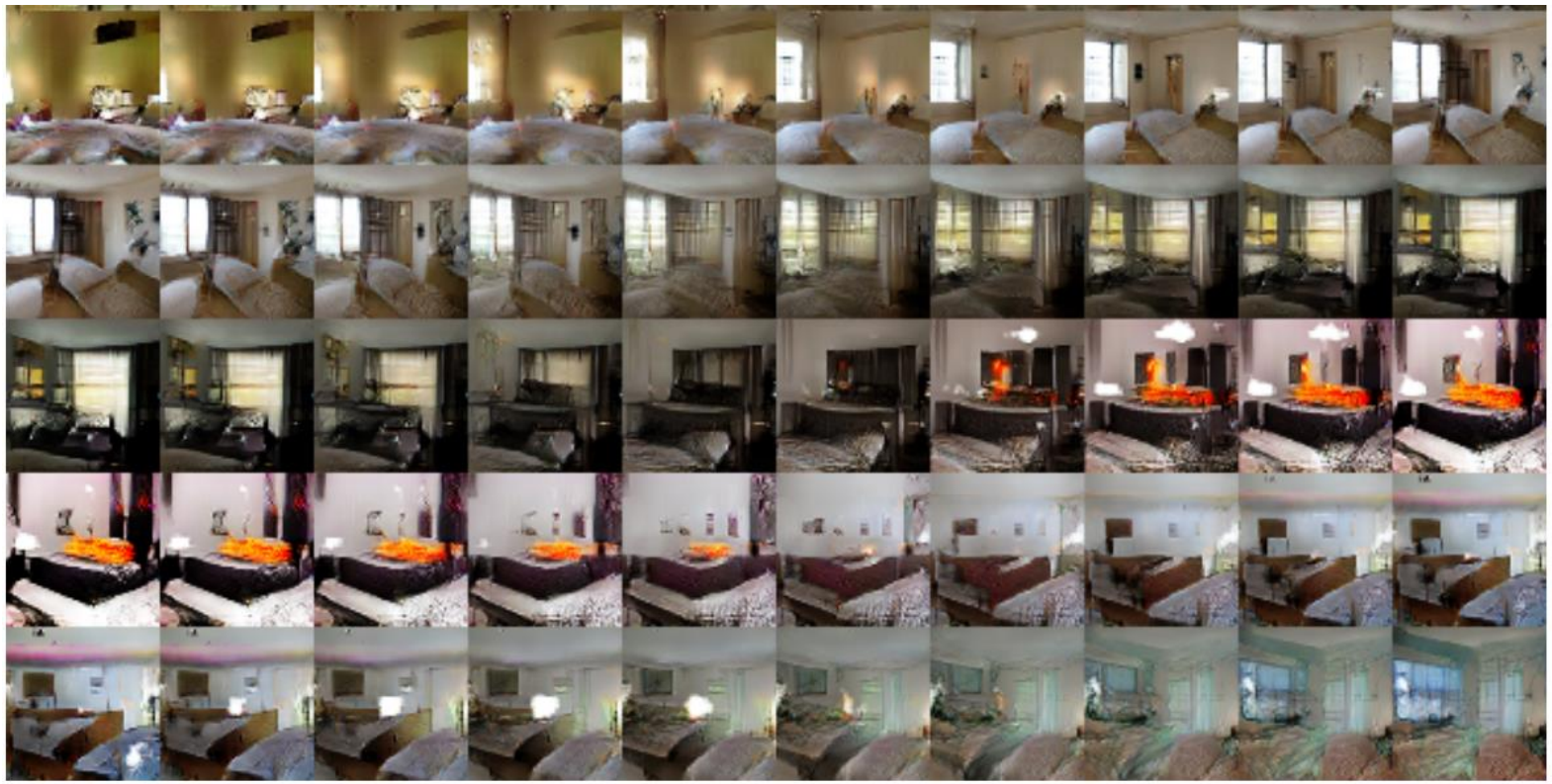
Deep Convolutional GAN (GTC 2016)

- Generated Samples – Natural Scene



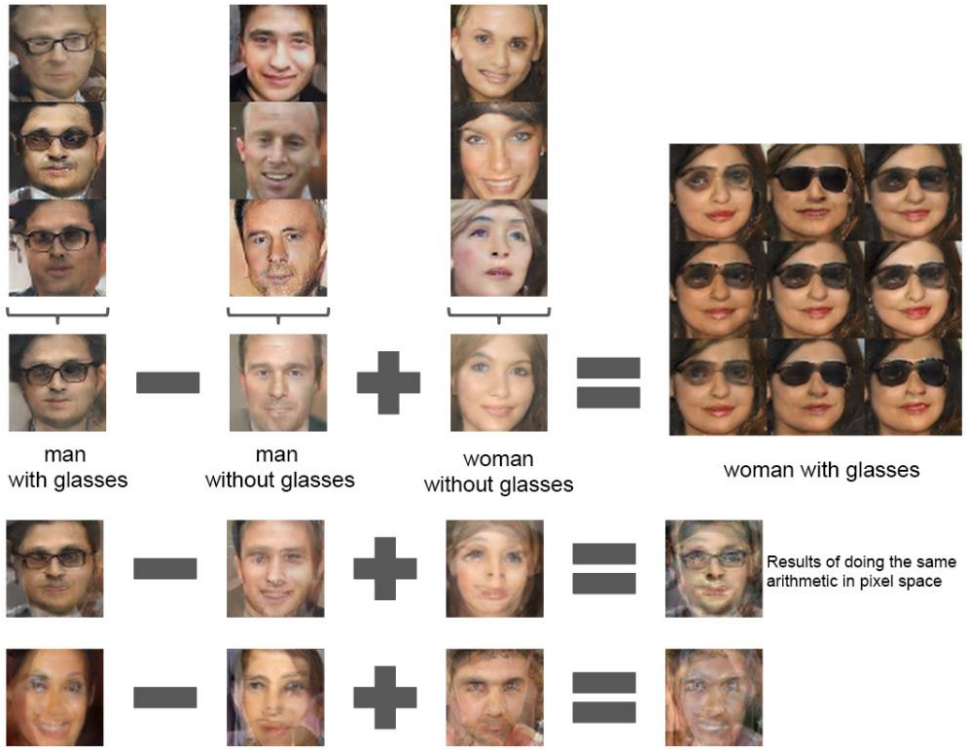
Deep Convolutional GAN (ICLR 2016)

- Walking in the Latent Space
  - If sharp transition is not observed and moving in the latent space result in semantic changes, it can be considered relevant representation has been learned.



Deep Convolutional GAN (ICLR 2016)

- Vector Arithmetic for Visual Concepts
  - Arithmetic on averaged Z representation of generator shows rich linear structure in the representation space.

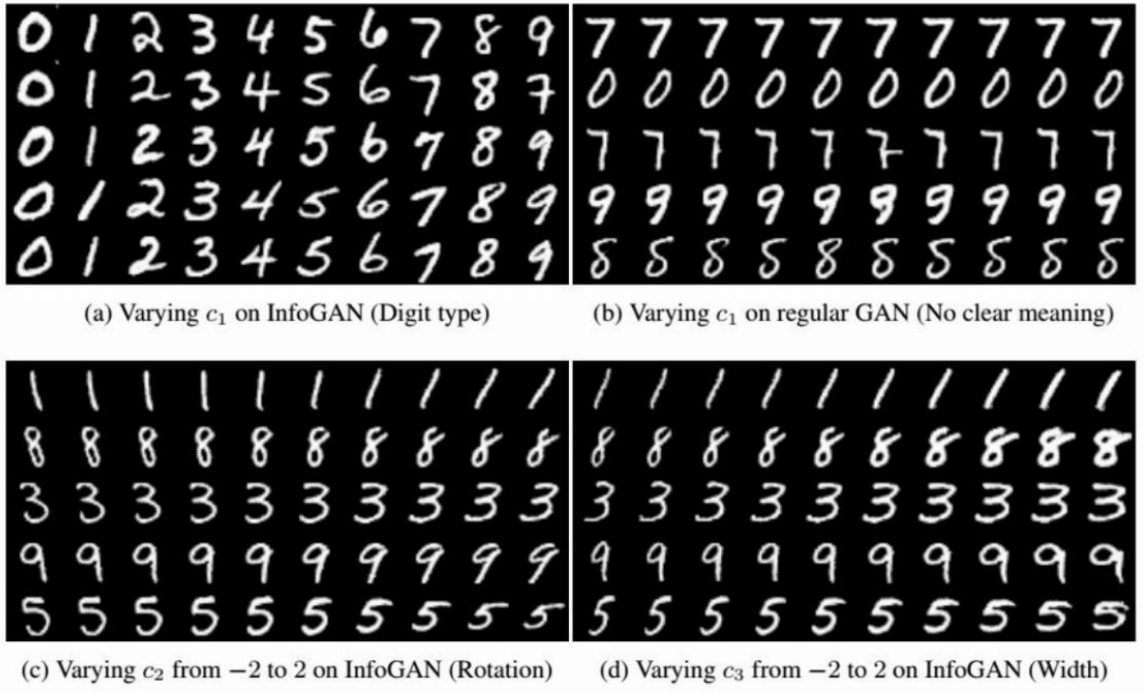


Variants of GANs

- InfoGAN
  - Add mutual information regularizer to induce meaningful latent code in GAN.

$$I(X;Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)$$

$$\min_G \max_D V_I(D,G) = V(D,G) - \lambda I(c;G(z,c))$$



(a) Varying c_1 on InfoGAN (Digit type)

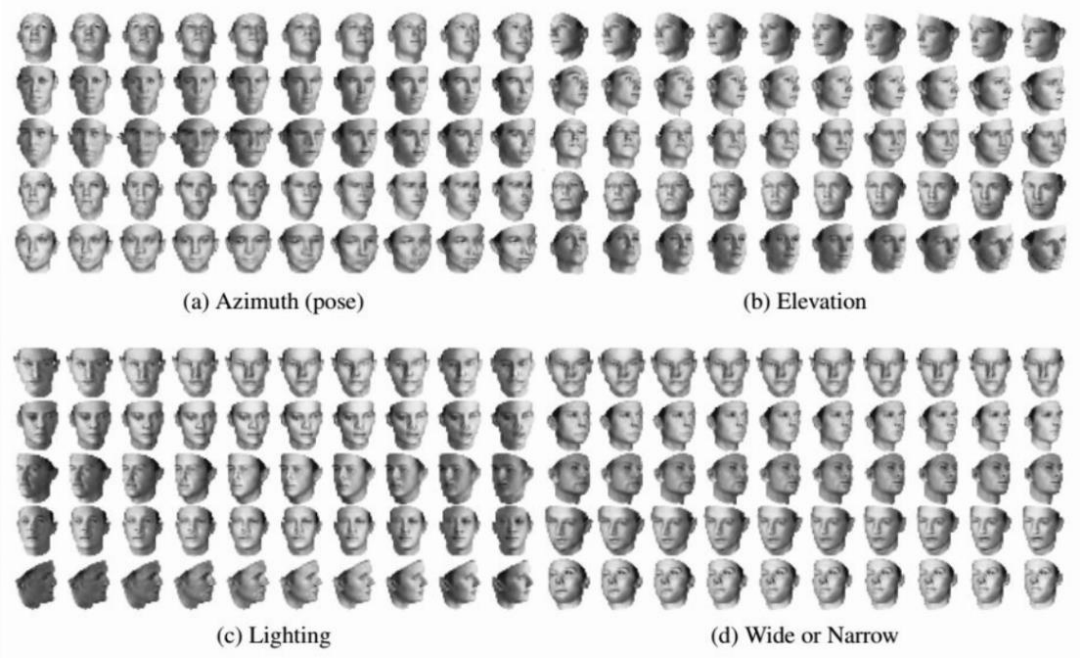
(b) Varying c_1 on regular GAN (No clear meaning)

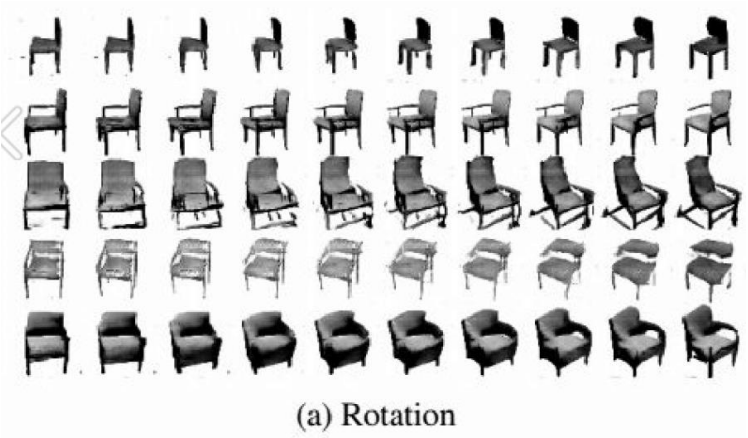
(c) Varying c_2 from -2 to 2 on InfoGAN (Rotation)

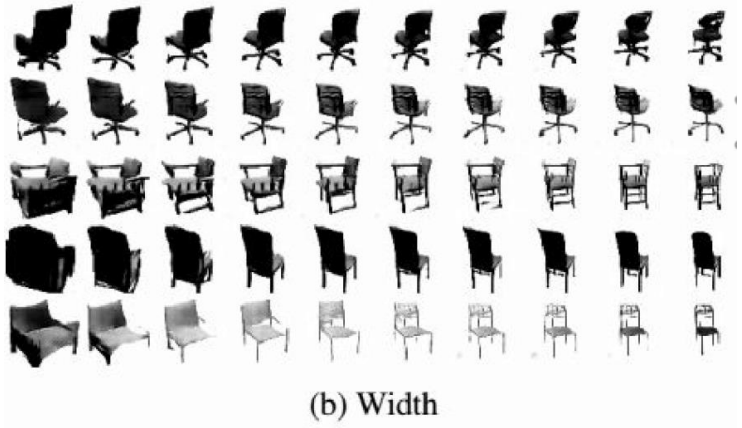
(d) Varying c_3 from -2 to 2 on InfoGAN (Width)

Variants of GANs

- InfoGAN







TensorFlow Tutorial

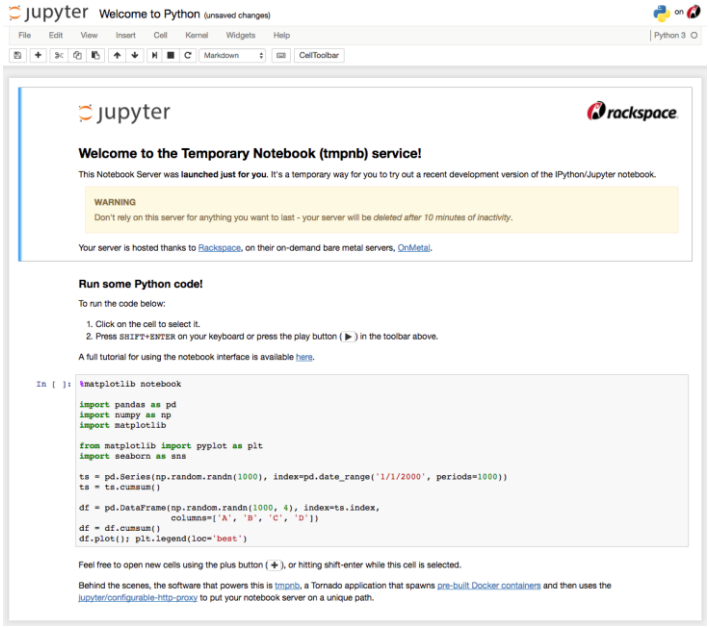
Jupyter Notebook

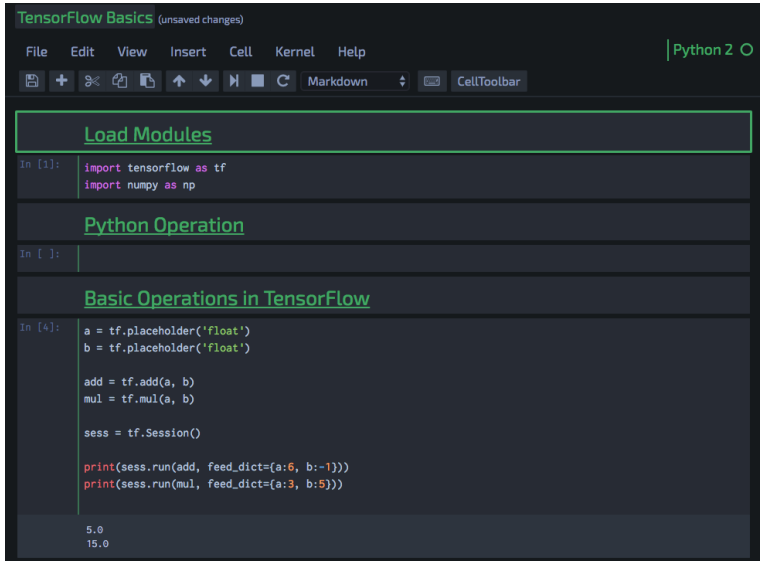
- Jypter Notebook
  - http://jupyter.org/
  - Formerly known as IPython notebook
  - Open source, web-based application for interactive scientific computing.
  - Support +40 languages including python.
  - run with jupyter notebook

- Installation
  - For Linux and Mac
    - pip3 install –upgrade pip
    - pip3 install jupyter
  - For Windows
    - Install Anaconda
    - https://www.continuum.io/downloads

Jupyter Notebook

- Themes
  - https://github.com/dunovank/jupyter-themes
  - Installation
    - pip install jupyterthemes





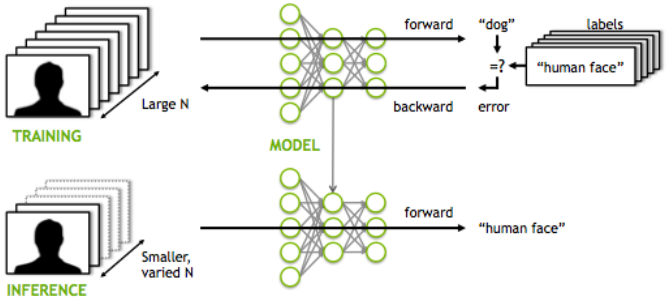
Nvidia : Getting Started with Deep Learning

Step1
Select your Software

Step2
Choose a GPU

Step3
Get Up and Running

Caffe
Chainer
DL4J
Deeplearning4j
MINERVA
mxnet
Purine
KERAS
Microsoft CNTK
MatConvNet
TensorFlow
theano
torch



Software : Deep Learning Frameworks

Name	Description
TensorFlow – New!	Deep learning backend based on data flow graph by Google
Veles – New!	Distributed deep learning platform by Samsung. OpenCL supported.
Caffe	Deep learning framework made for C++ by Berkeley Vision and Learning Center
Theano	Python library developed by universite de Montreal mostly by LISA group
Torch7	Scientific computing framework based on Lua maintained by former Yann LeCun's group
cuda-convnet2	CNN libarary based on C++ developed by Alex Krizhevsky(Google)
MatConvNet	MATLAB Toolbox implementing CNNs developed by Oxford VGG team
DeepLearning4J	Java and Scala based distributed deep learning framework developed by Skymind
DIGITS2	Web interface for training CNN based on Caffe. Developed by Nvidia
Keras	Theano-based Python deep library inspired by Torch
Chainer	Python-based flexible frame for deep leaning framework developed by Preferred Networks
Lasagne	Lightweight library to build neural networks in Theano developed by Sander Dieleman(DeepMind)
Mocha	Deep learning framework for Julia, inspired by Caffe
RNNLIB	RNN library for sequence labelling problems based on C++ developed by Alex Graves
CURRENNT	CUDA-enabled machine learning library for recurrent neural networks by TU Munchen
pyBrain	Python-Based Reinforcement Learning, Artificial Intelligence and Neural Network Libarary by IDSIA
pyLearn2	Python machine learning library built on top of Theano developed by LISA group
Neon	Highly configurable deep learning framework developed by Nervana Systems
Mozi	Deep learning package based on Theano with clean and sharp design
And more …	deepnet, deeppy, convnetjs, DeepLearingToolbox, char-rnn,

Software : Deep Learning Frameworks

Aggregate popularity (30•contrib + 10•issues + 5•forks)•10e-3		
#1:	77.88	tensorflow/tensorflow
#2:	65.16	BVLC/caffe
#3:	36.70	fchollet/keras
#4:	30.05	Theano/Theano
#5:	25.45	dmlc/mxnet
#6:	16.73	deeplearning4j/deeplearning4j
#7:	13.10	Microsoft/CNTK
#8:	12.83	torch/torch7
#9:	8.19	Lasagne/Lasagne
#10:	7.65	mila-udem/blocks
#11:	7.63	pfnet/chainer
#12:	6.86	NVIDIA/DIGITS
#13:	5.63	karpathy/convnetjs
#14:	5.17	NervanaSystems/neon
#15:	2.56	tflearn/tflearn
#16:	1.75	IDSIA/brainstorm

https://twitter.com/fchollet/status/765212287531495424

Software : Deep Learning Frameworks

Framework	Base Language	multi-GPU?	Execution Speed*	Research Areas of Applicability	Pros	Cons
TensorFlow	Python and C++	yes	slower than theano & torch	general	1) Biggest user base 2) Maintained by Google	1) Not perfect support for GPU on Windows 2) Relatively slow
Torch	Lua	yes	competitive with Theano	general	1.) easy to set up 2.) helpful error messages 3.) large amount of sample code and tutorials	1)Can be somewhat difficult to set up in CentOS 2) Written in Lua
Caffe	C++	yes	slower than theano & torch	Image Classification		1)Vision specific modules 2)Maintained by Academy
Theano	Python	By default, no.	competitive with Torch	general	1.) expressive Python syntax 2.) higher-level spin-off frameworks 3.) large amount of sample code and tutorials	1) Hard to debug 2) Maintained by Academy

TensorFlow

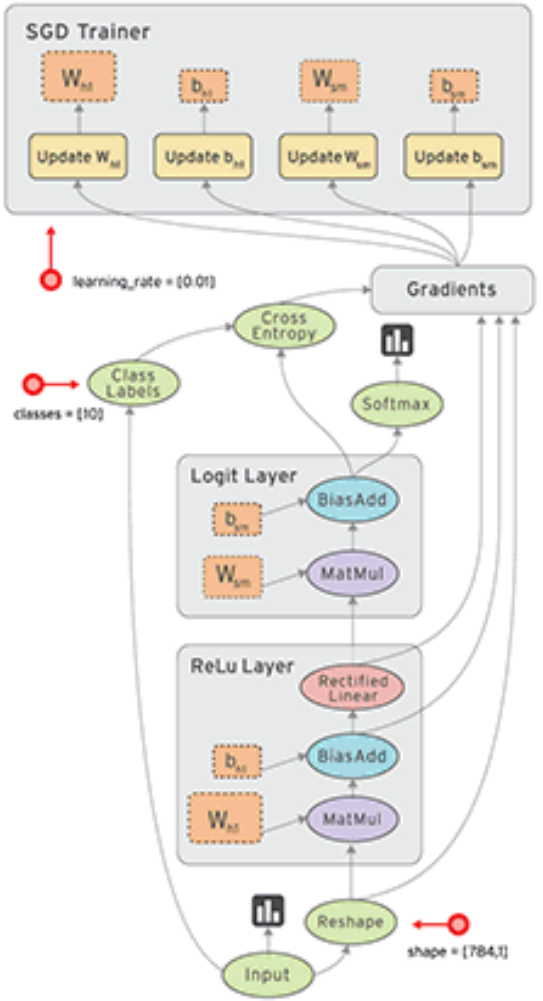
- TensorFlow was originally developed by the Google Brain Team within Google's Machine Intelligence research organization for the purposes of conducting machine learning and deep neural networks research

- Open source software library for numerical computation
using data flow graphs.

- The flexible architecture allows you to deploy computation
to one or more CPUs or GPUs in a desktop, server, or
mobile device with a single API.

TensorFlow

- Describe mathematical computation with a directed graph of nodes and edges.
  - Nodes in the graph represent mathematical operations
  - Edges describe the i/o relationships between nodes.
  - Data edges carry dynamically-sized multidimensional data arrays, or tensors.
- The flow of tensors through the graph is where TensorFlow gets its name.
- Nodes are assigned to computational devices and execute asynchronously and in parallel once all the tensors on their incoming edges becomes available



TensorFlow

- Installation
  - Linux Installation
    - pip install tensorflow
    - pip install tensorflow-gpu
  - Windows Installation
    - http://goodtogreate.tistory.com/entry/GPU-TensorFlow-on-Window-10-TensorFlow-GPU%EB%B2%84%EC%A0%84-%EC%9C%88%EB%8F%84%EC%9A%B010-%EC%84%A4%EC%B9%98

- Getting Started
  - Python Basics(https://github.com/KyuhwanJung/tensorflow_tutorial/blob/master/Python%20Basics.ipynb)
  - TensorFlow Basics(https://github.com/KyuhwanJung/tensorflow_tutorial/blob/master/TensorFlow%20Basics.ipynb)

Deep Learning and TensorFlow Resources

- https://github.com/ChristosChristofidis/awesome-deep-learning
- https://github.com/kjw0612/awesome-rnn
- https://github.com/kjw0612/awesome-deep-vision
- https://github.com/sjchoi86/Deep-Learning-101
- https://github.com/sjchoi86/Tensorflow-101
- https://github.com/jtoy/awesome-tensorflow
- https://github.com/nlintz/TensorFlow-Tutorials
- https://github.com/aymericdamien/TensorFlow-Examples
- http://www.jorditorres.org/first-contact-with-tensorflow/
- http://hunkim.github.io/ml/