Computer Vision 1

THEO GEVERS

MASTER AI

UNIVERSITY OF AMSTERDAM

Lectures/Theory

- 06-02-2018, 17:00-19:00, C0.05, **Introduction** (*Szeliski 1*)
- 13-02-2018, 17:00-19:00, C0.05, **Image Formation** (*Szeliski: 2.1.1 + 2.1.2 + 2.2 + 2.3.2 + 2.3.3*)
- 20-02-2018, 17:00-19:00, C0.05, **Color and Image Processing** (*Szeliski: 3.1 + 3.2 + 3.3*)
- 27-02-2018, 17:00-19:00, C0.05, **Feature Detection, Motion and Classification** (*Szeliski: 4, 8.1.1 + 8.1.3 + 8.2.1 + 8.4; Bengio: 4 + 5.1 + 5.2 + 5.3 + 5.7 + 5.8 + 5.9*)
- 06-03-2018, 17:00-19:00, C0.05, **Object Recognition: BoW and Deep Learning** (*Szeliski: 5.1.1 + 5.1.4 + 5.1.5 + 5.2 + 5.3 + 5.4, 6.1 + 6.3, 14.1 + 14.2.1 + 14.3 + 14.4.1; Bengio: 7.2 + 7.4 + 9.1 + 9.2 + 9.3*)
- 13-03-2018, 17:00-19:00, C0.05, **ConvNets, Stereo and 3D Reconstruction** (*Szeliski: 11.1 + 11.2 + 11.3 + 11.4, 12.1 + 12.2: Bengio: 12.1 + 12.2*)
- 20-03-2018, 17:00-19:00, C0.05, **Applications** (*Szeliski: 12.6.2 + 12.6.3 + 12.2.4)*
- 26-03-2018, Monday, 9:00-12:00, **Written Exam**

Today’s class

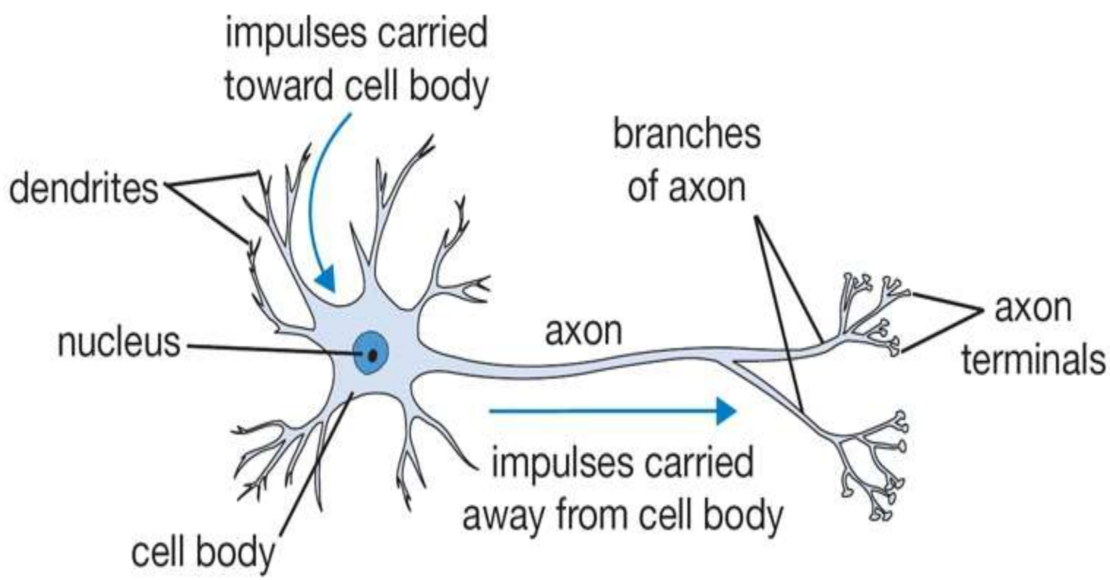
Object Recognition (ConvNets)

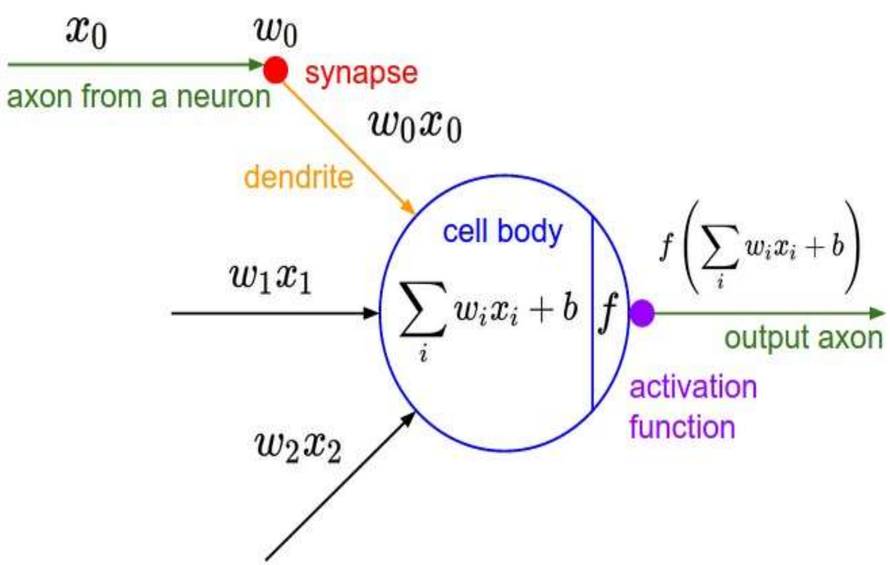
Object Detection

Stereo Vision

Object Recognition using Deep Learning

Artifical Model (1943 McCulloch/Pitts)





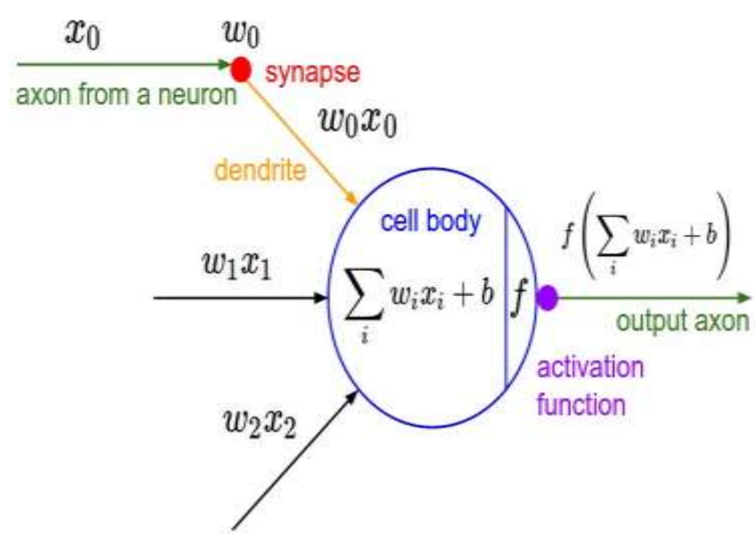
The Neuron

Inputs: $\vec{x} = x_1, x_1, \ldots, x_n$

Weights: $\vec{w} = w_1, w_1, \ldots, w_n$

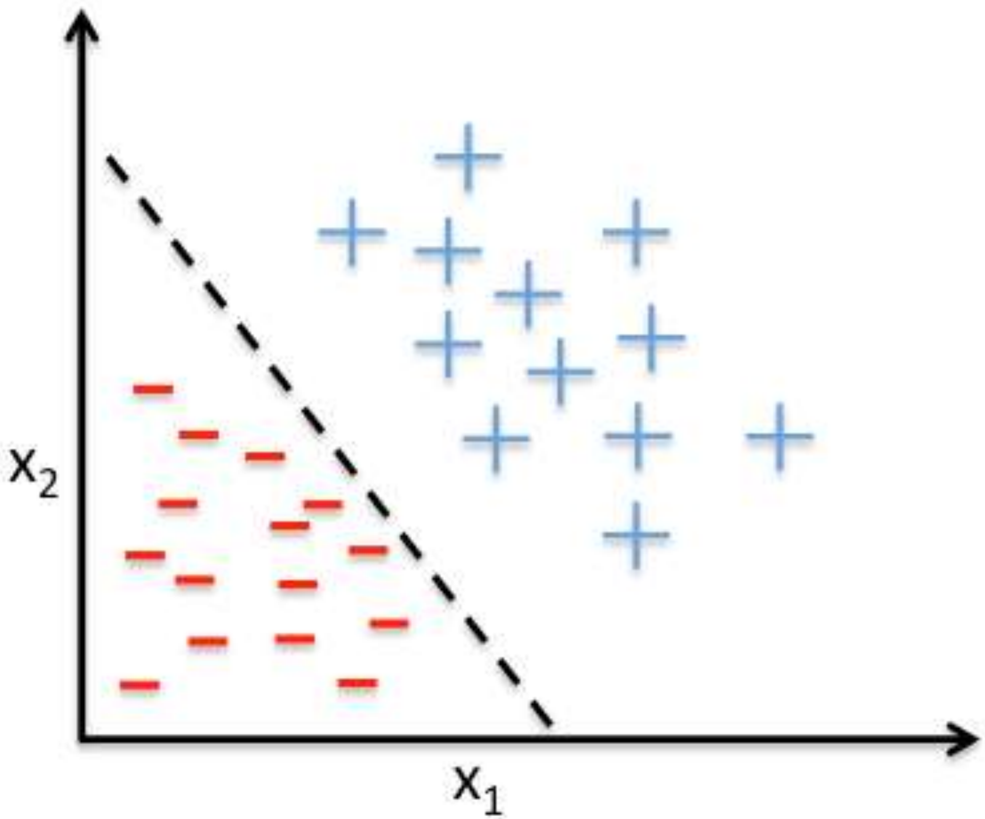
Logit: $$z = \sum_{i=1}^{n} w_i x_i + b$$

Output: $$y = f(z)$$



Output: $$y = f(\vec{x} \cdot \vec{w} + b)$$

Linear Perceptron



Example of a linear decision boundary for binary classification.

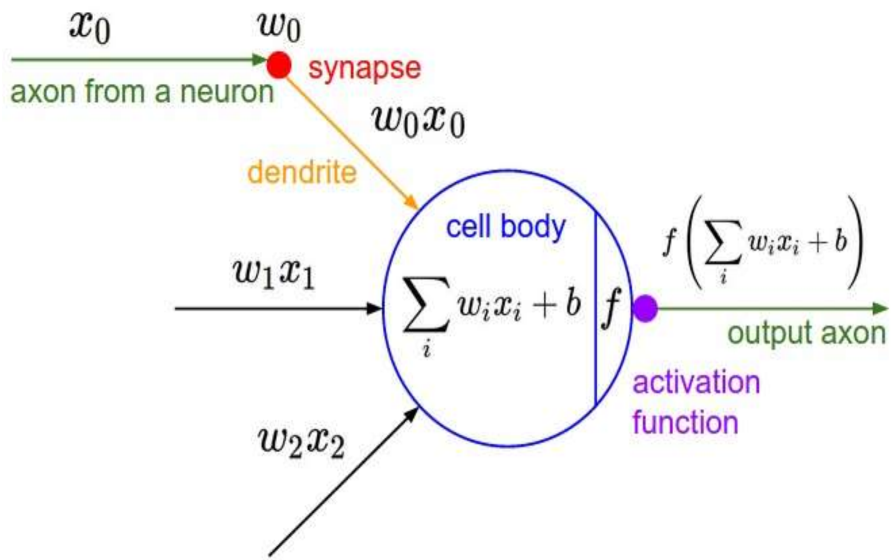
Inputs $x_1, x_1, \ldots, x_n$

Weights $w_1, w_1, \ldots, w_n$

Logit $z = \sum_{i=1}^{n} w_i x_i + b$

$$f(z) = \begin{cases} -1 & \text{if } z < 0 \\ 1 & \text{if } z \geq 0 \end{cases}$$

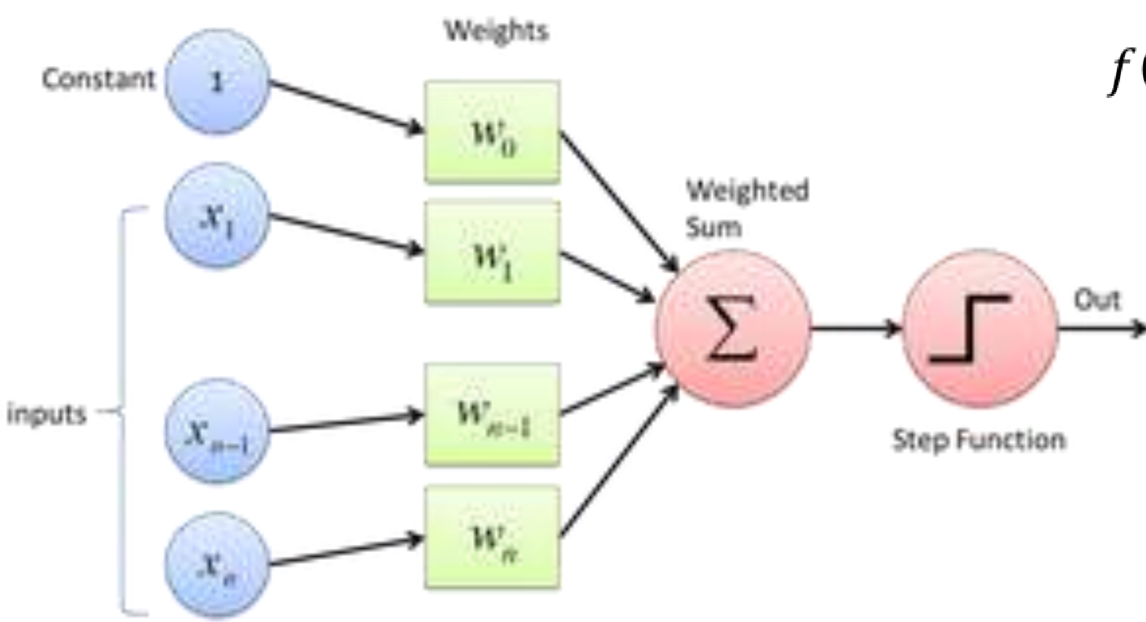
Linear Perceptron



Inputs $x_1, x_1, \ldots, x_n$

Weights $w_1, w_1, \ldots, w_n$

Logit $z = \sum_{i=1}^{n} w_i x_i + b$

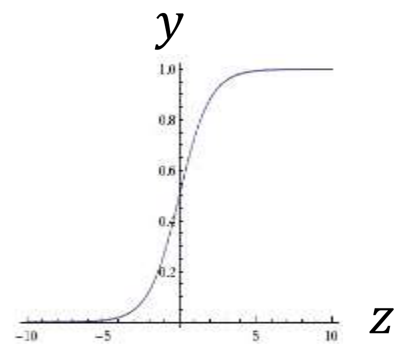


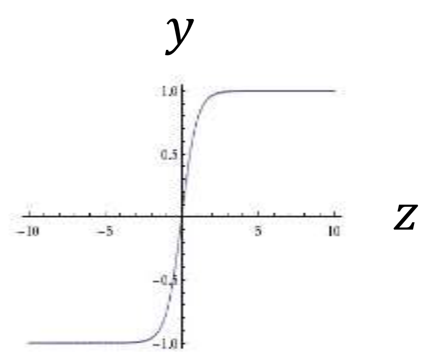
$$f(z) = \begin{cases} -1 & \text{if } z < 0 \\ 1 & \text{if } z \geq 0 \end{cases}$$

Sigmoid, Tanh, and ReLU Neurons

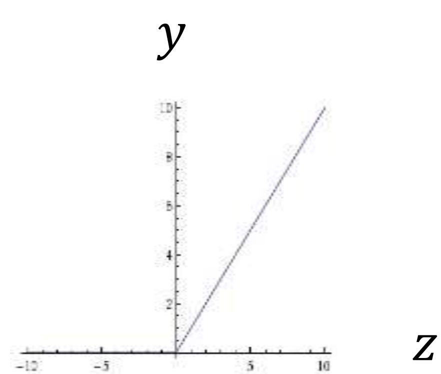
Sigmoid

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





tanh tanh(z)



ReLU max(0,z)

Feed-forward Neural Networks

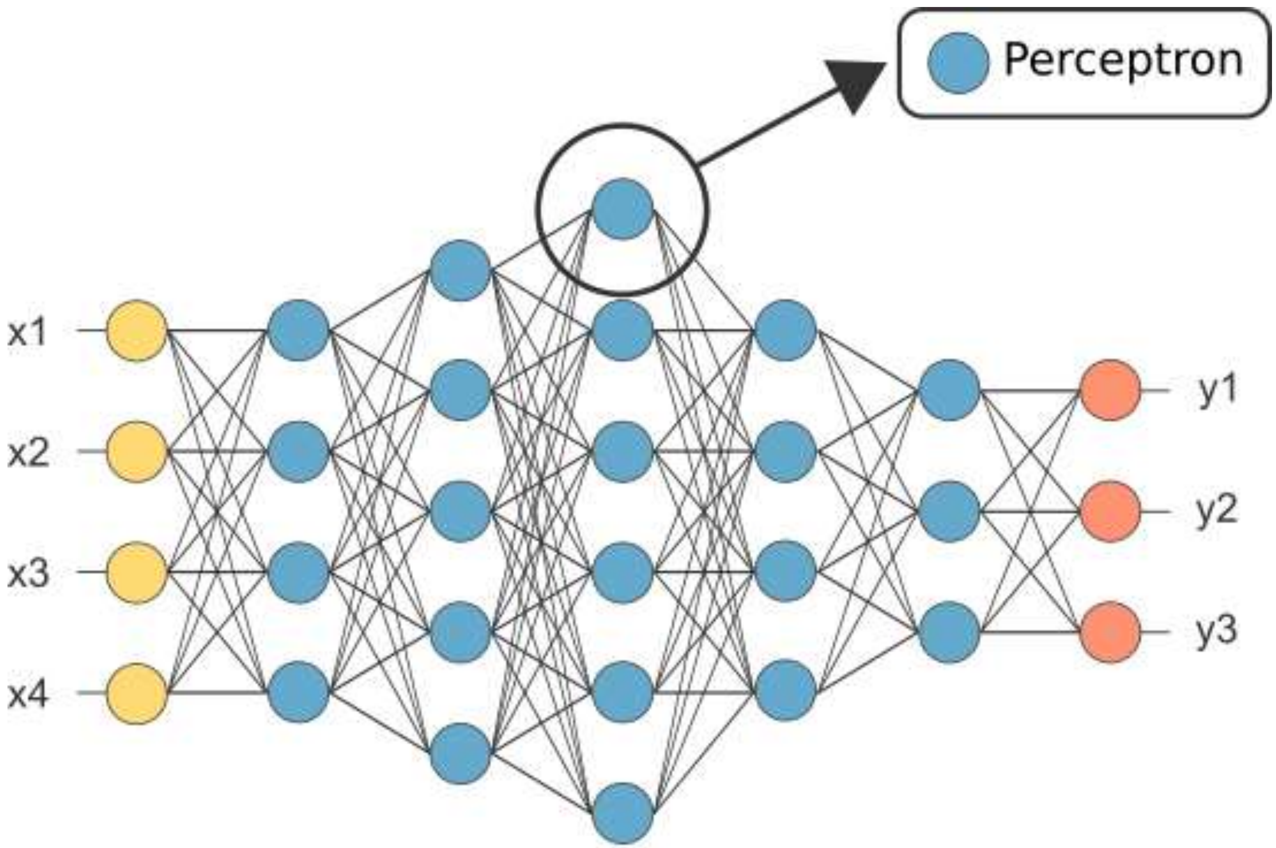
$$\vec{y} = f(\overrightarrow{W} \cdot \vec{x} + \vec{b})$$

Weight matrix nxm: $\overrightarrow{W}$

Inputs: $\vec{x} = x_1, x_1, \ldots, x_n$

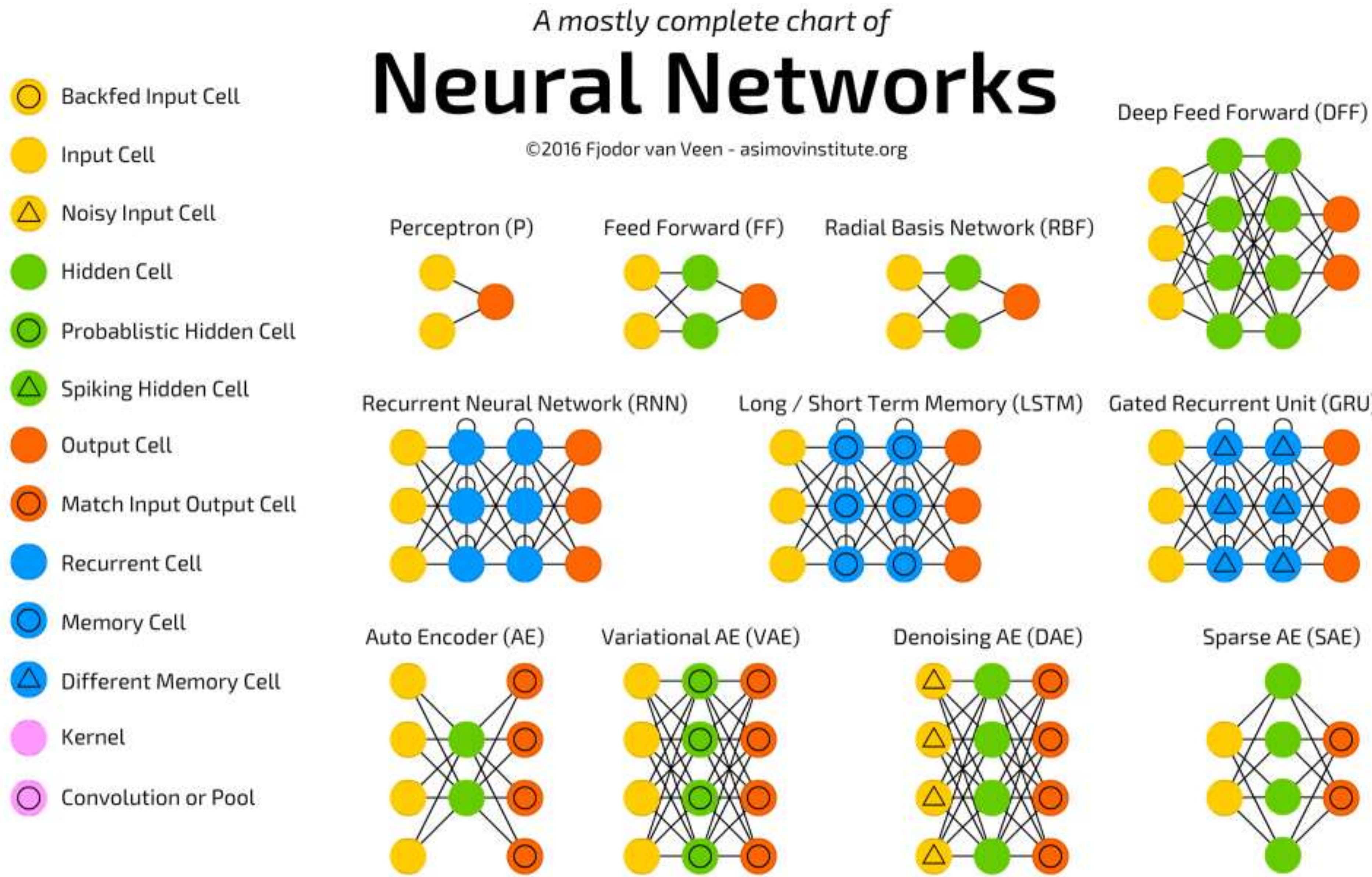
Ouputs: $\vec{y} = y_1, y_1, \ldots, y_m$

Bias: $\vec{b}$



Softmax Output Layer

$$y_i = \frac{e^{z_i}}{\sum_j e^{z_j}}$$



A mostly complete chart of
Neural Networks
©2016 Fjodor van Veen - asimovinstitute.org
Backfed Input Cell
Input Cell
Noisy Input Cell
Hidden Cell
Probablistic Hidden Cell
Spiking Hidden Cell
Output Cell
Match Input Output Cell
Recurrent Cell
Memory Cell
Different Memory Cell
Kernel
Convolution or Pool
Perceptron (P)
Feed Forward (FF)
Radial Basis Network (RBF)
Deep Feed Forward (DFF)
Recurrent Neural Network (RNN)
Long / Short Term Memory (LSTM)
Gated Recurrent Unit (GRU)
Auto Encoder (AE)
Variational AE (VAE)
Denoising AE (DAE)
Sparse AE (SAE)

Single layer feed-forward NN

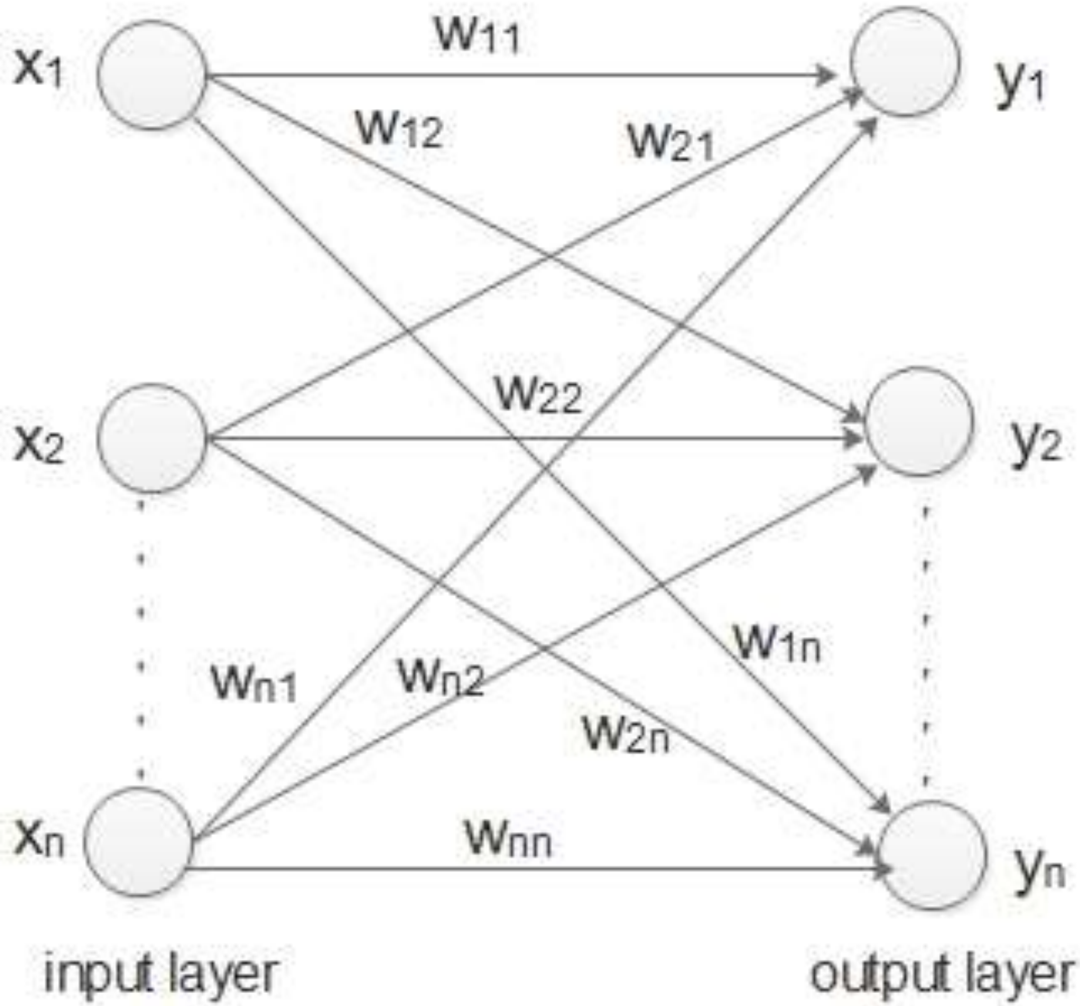


Figure: Single layer feed forward NN

CIFAR-10

10 labels
50,000 training images, each image is tiny: 32x32
10,000 test images.

$$\vec{y} = f(\overrightarrow{W} \cdot \vec{x} + \vec{b})$$

[32x32x3]
array of numbers
i.e. 3072 numbers total

10 numbers,
indicating class
scores

Andrej Karpathy, Fei-Fei Li, Justin Johnson

Output=10x1 **Weights=10x3072** **Bias=10x1**

$$\vec{y} = f(\overrightarrow{W} \cdot \vec{x} + \vec{b})$$

Input=3072x1

10 numbers, indicating class scores

[32x32x3]
array of numbers 0...1
(3072 numbers total)

Andrej Karpathy, Fei-Fei Li, Justin Johnson

Output=3x1 **Weights=3x4** **Bias=4x1**

$$\vec{y} = f(\vec{W} \cdot \vec{x} + \vec{b})$$

Input=4x1

(cat/dog/ship)

stretch pixels into single column

input image

$$\underbrace{\begin{bmatrix} 0.2 & -0.5 & 0.1 & 2.0 \\ 1.5 & 1.3 & 2.1 & 0.0 \\ 0 & 0.25 & 0.2 & -0.3 \end{bmatrix}}_{\vec{W}} \underbrace{\begin{bmatrix} 56 \\ 231 \\ 24 \\ 2 \end{bmatrix}}_{\vec{x}} + \underbrace{\begin{bmatrix} 1.1 \\ 3.2 \\ -1.2 \end{bmatrix}}_{\vec{b}} \longrightarrow \underbrace{\begin{bmatrix} -96.8 \\ 437.9 \\ 61.95 \end{bmatrix}}_{\vec{y}} \begin{array}{l} \text{cat score} \\ \text{dog score} \\ \text{ship score} \end{array}$$

Weights=3x4 **Bias=4x1** **Output=3x1**

Input=4x1

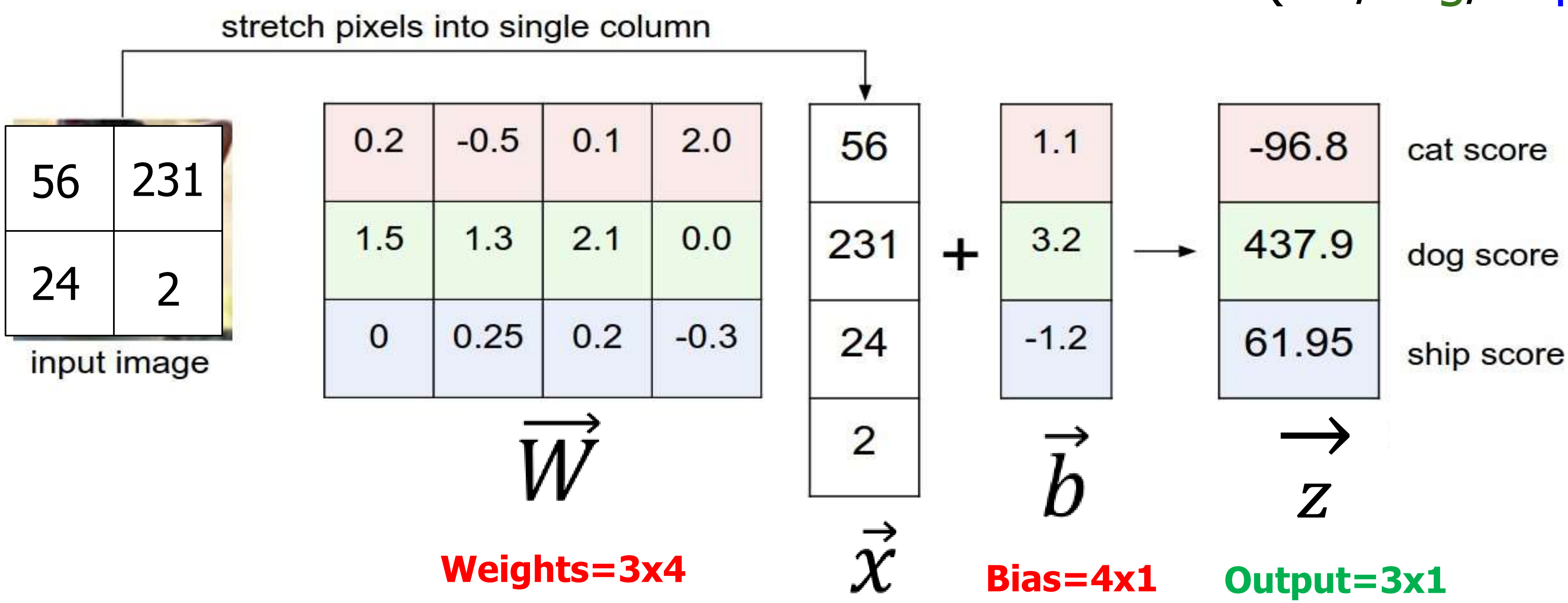
Andrej Karpathy, Fei-Fei Li, Justin Johnson

Output=3x1 Weights=3x4 Bias=4x1

$$\vec{y} = f(\vec{W} \cdot \vec{x} + \vec{b})$$

Input=4x1

(cat/dog/ship)

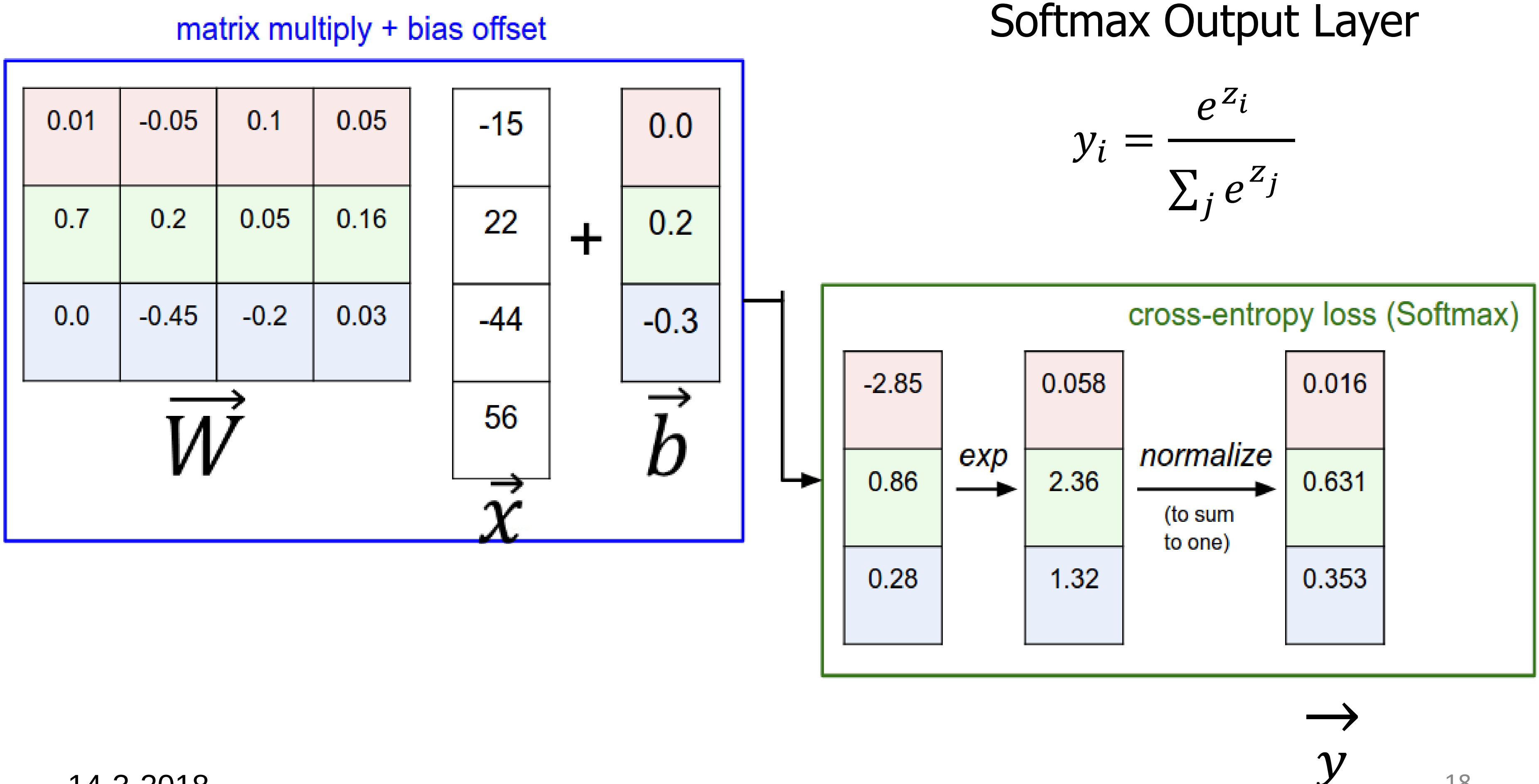


Input=4x1

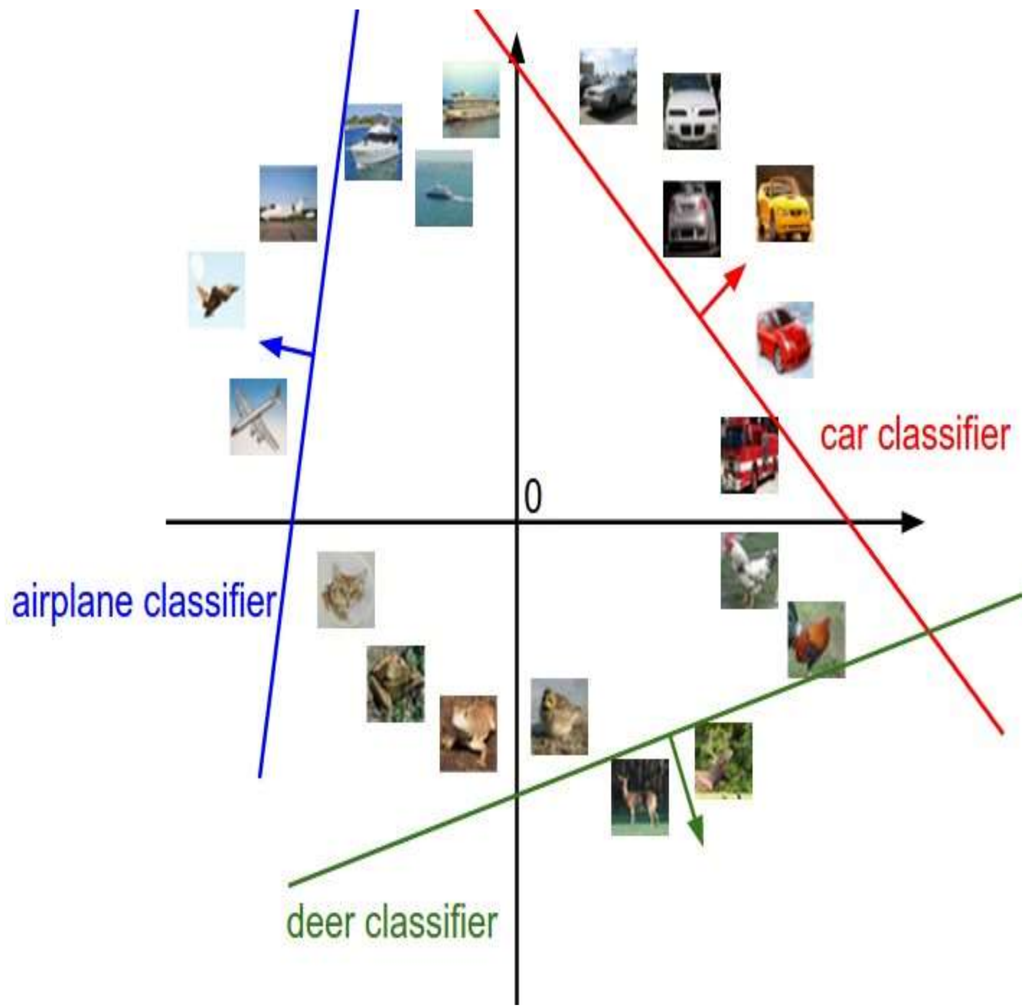
Andrej Karpathy, Fei-Fei Li, Justin Johnson

Softmax Output Layer

$$y_i = \frac{e^{z_i}}{\sum_j e^{z_j}}$$



Andrej Karpathy, Fei-Fei Li, Justin Johnson



$$\vec{y} = f(\overrightarrow{W} \cdot \vec{x} + \vec{b})$$

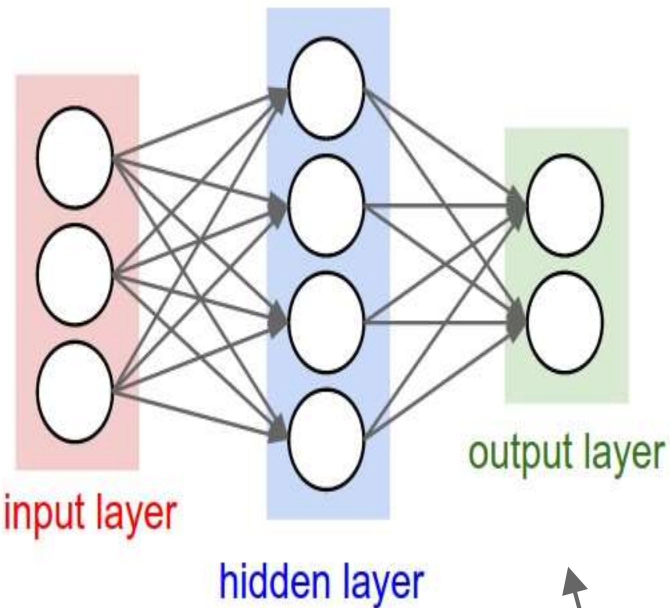
[32x32x3]
array of numbers 0...1
(3072 numbers total)

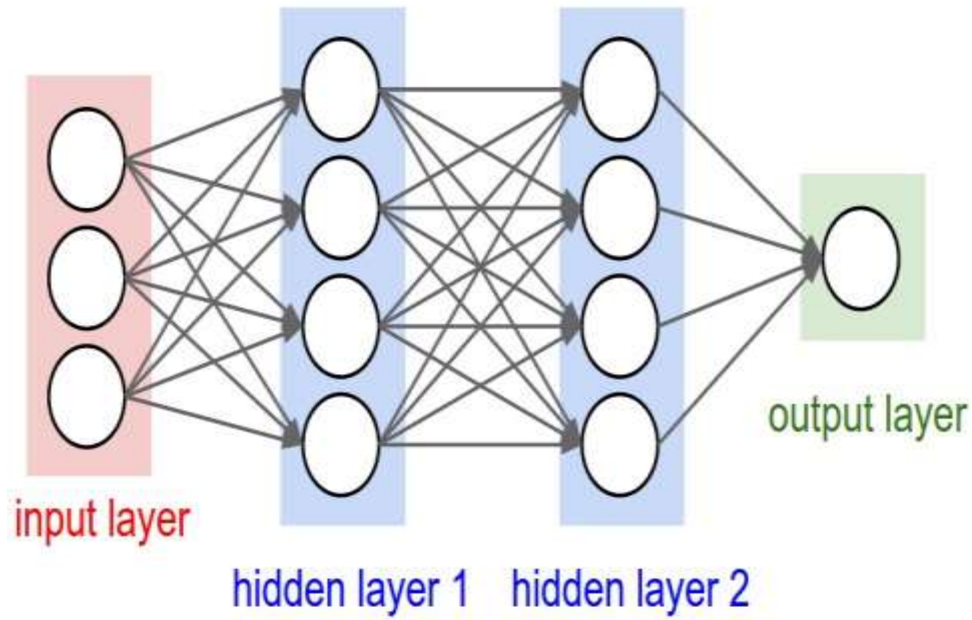
Andrej Karpathy, Fei-Fei Li, Justin Johnson

$$f(x_i, W, b) = Wx_i + b$$

Example trained weights of a linear classifier trained on CIFAR-10:

Andrej Karpathy, Fei-Fei Li, Justin Johnson

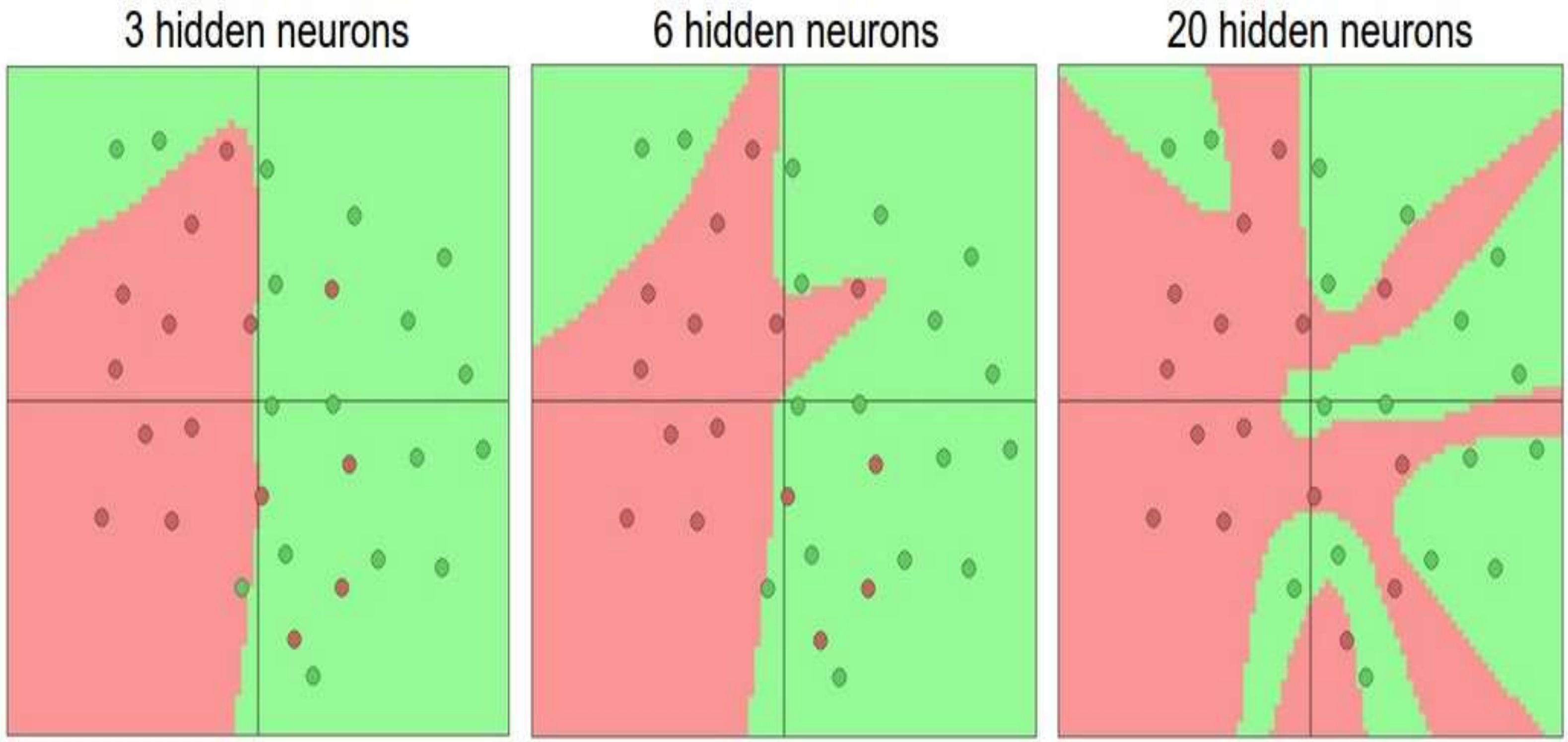




"3-layer Neural Net", or
"2-hidden-layer Neural Net"

"2-layer Neural Net", or
"1-hidden-layer Neural Net"

"Fully-connected" layers

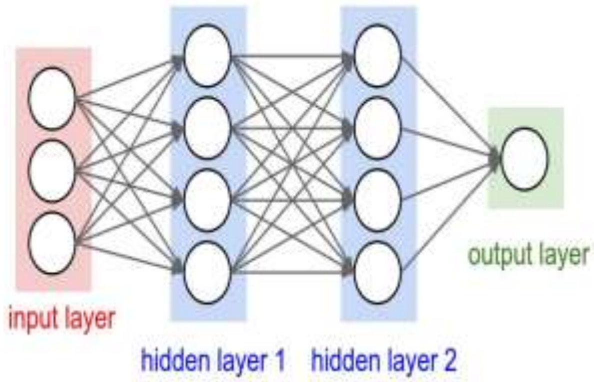


more neurons = more capacity

Object Recognition using ConvNets

Convolutional Neural Networks (CNN, ConvNet, DCN)

- Fully connected NN's: e.g. input layer is 200x200x3 color image = 120,000 weights!!
- CNN = a multi-layer neural network with
  - **Local** connectivity
  - **Share** weight parameters across spatial positions



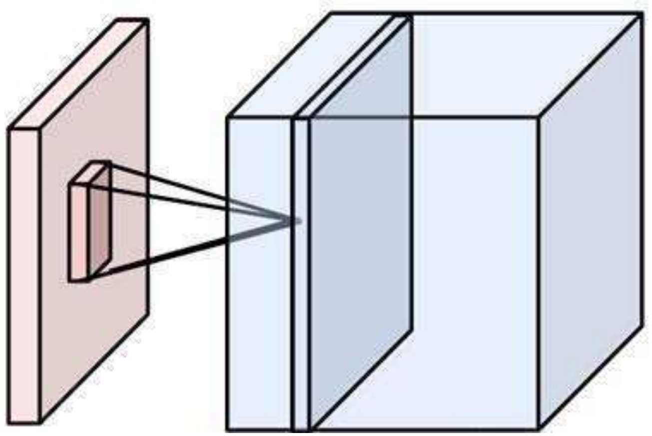


Image credit: A. Karpathy

Convolutional Layer (1)

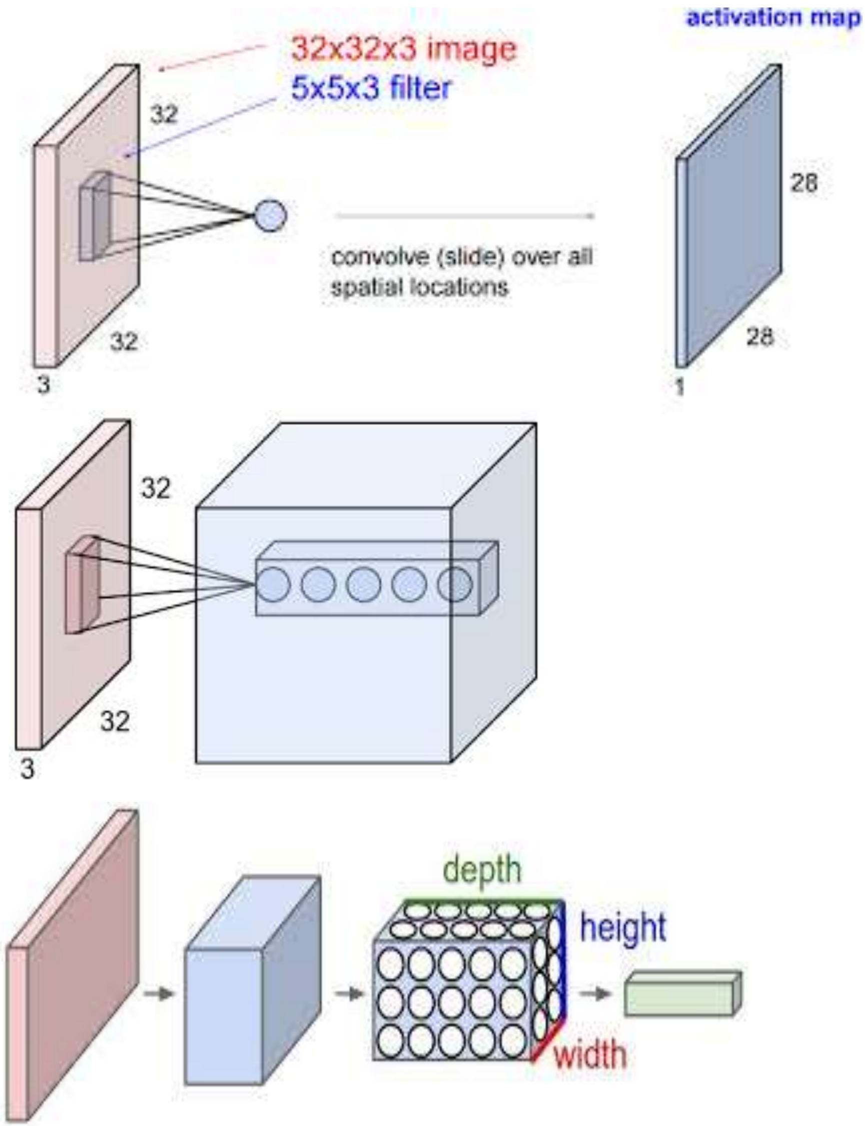
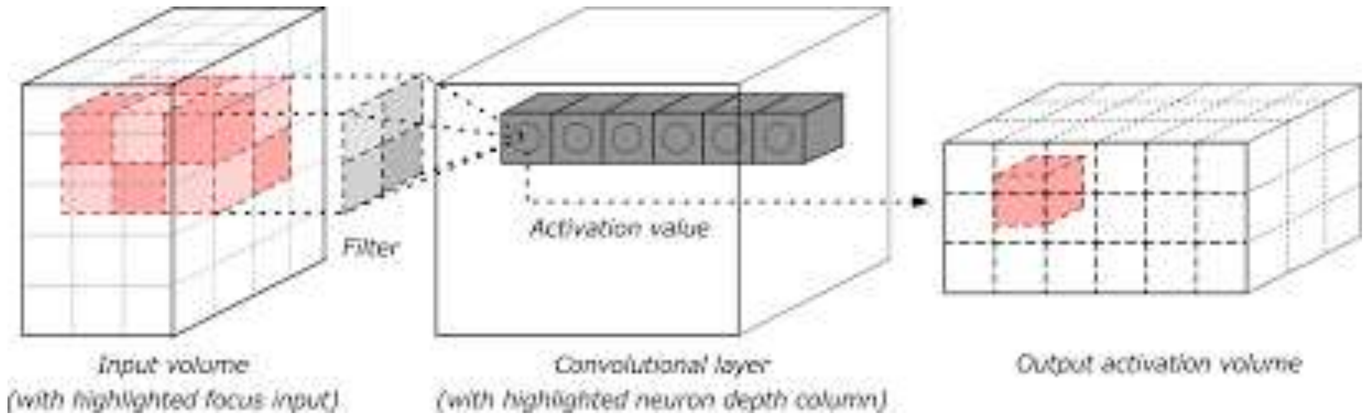


Image credit: A. Karpathy

Convolutional Layer (1)



Input volume of convolutional layer:

width w_{in}, height h_{in}, depth d_{in}, zero padding p

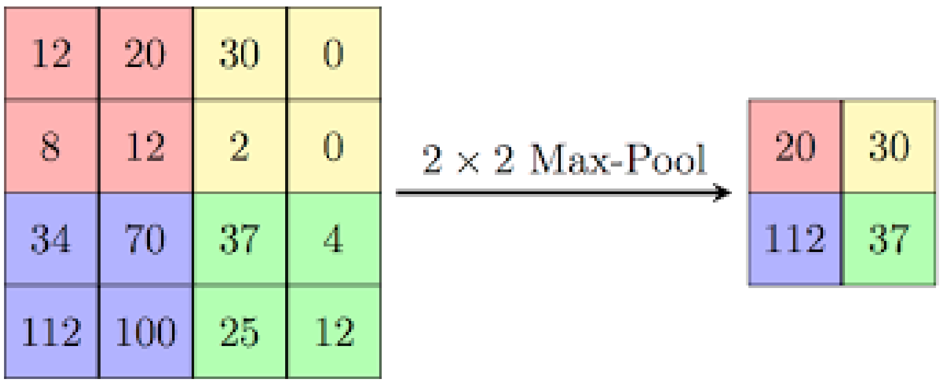
This input volume is processed by k filters. Filters are defined by:

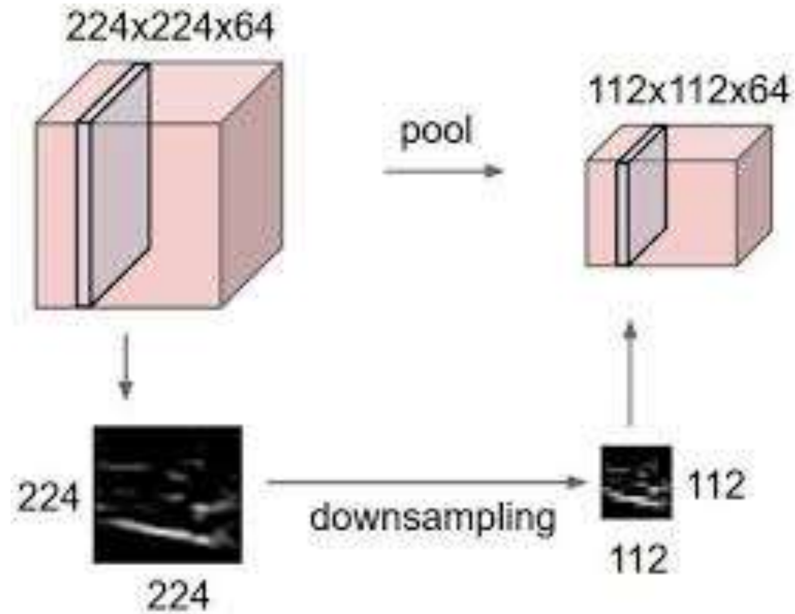
spatial extent e, stride s

Output volume parameters:

width $w_{out} = \left(\frac{w_{in}-e+2p}{s}\right) + 1$, height $h_{out} = \left(\frac{h_{in}-e+2p}{s}\right) + 1$, depth $d_{out} = \mathrm{k}$

Max Pooling





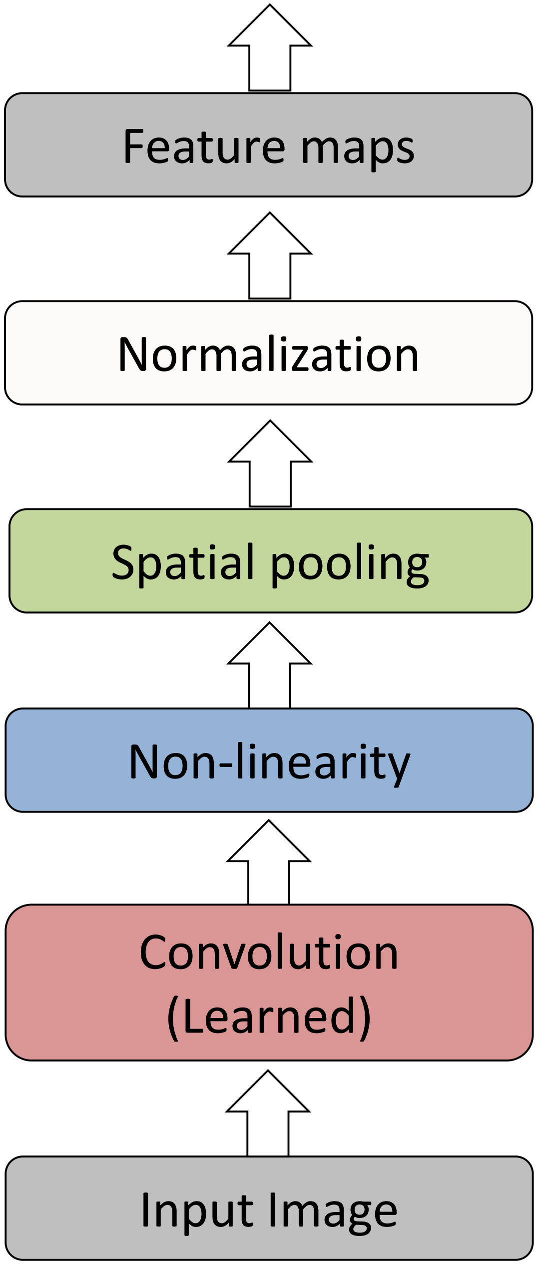
Pooling layer parameters:

spatial extent e, stride s

Resulting dimensions:

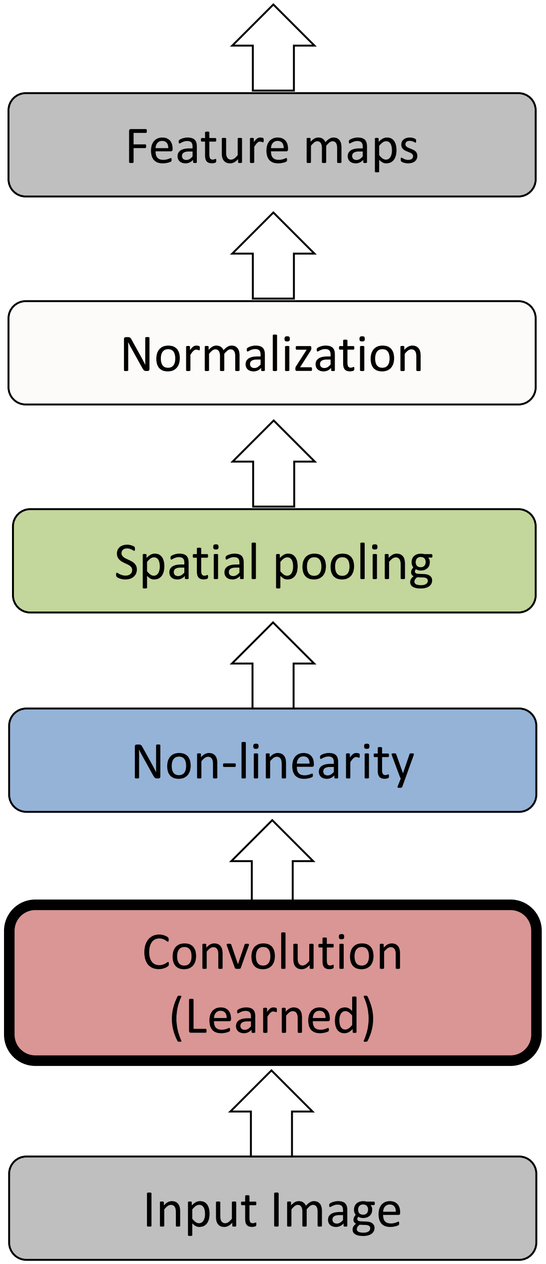
width $w_{out} = \left(\frac{w_{in} - e}{s}\right) + 1$, height $h_{out} = \left(\frac{h_{in} - e}{s}\right) + 1$

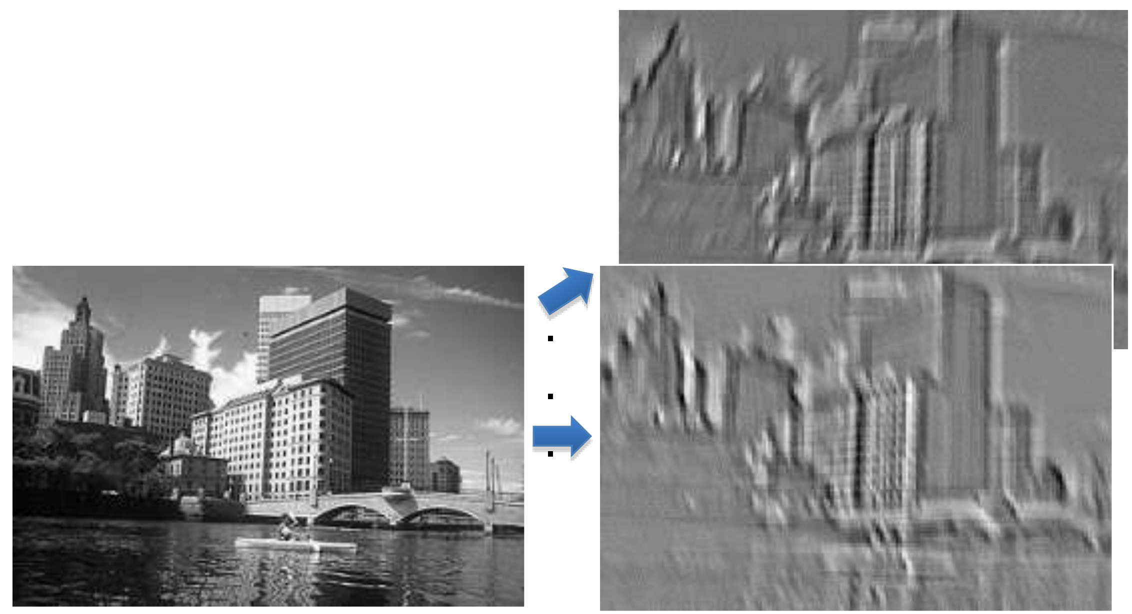
Convolutional Neural Networks



slide credit: S. Lazebnik

Convolutional Neural Networks



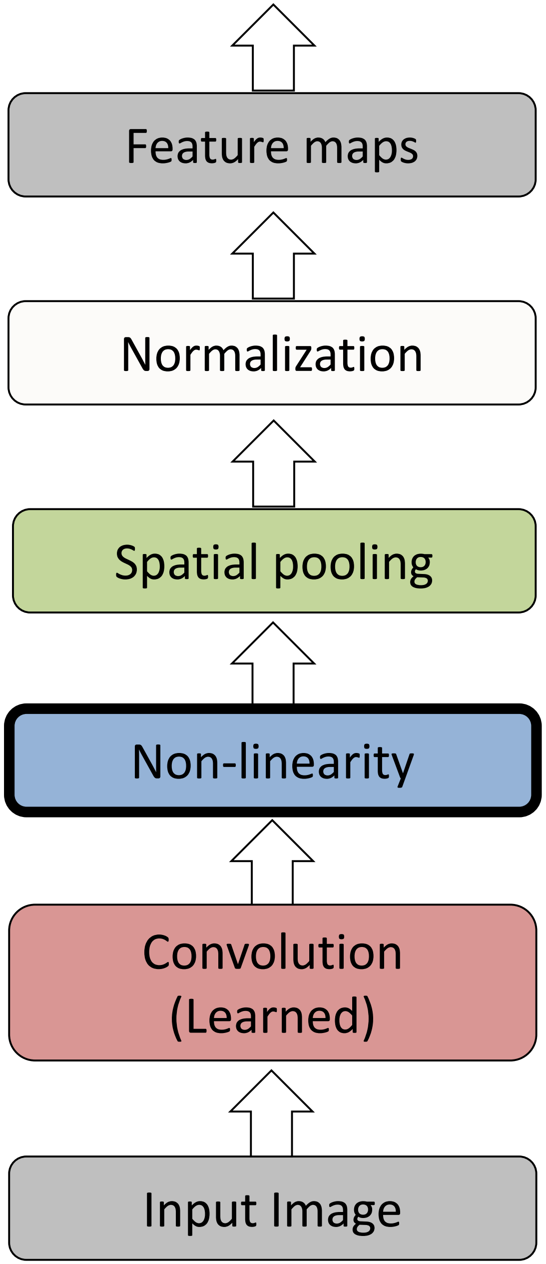


Input

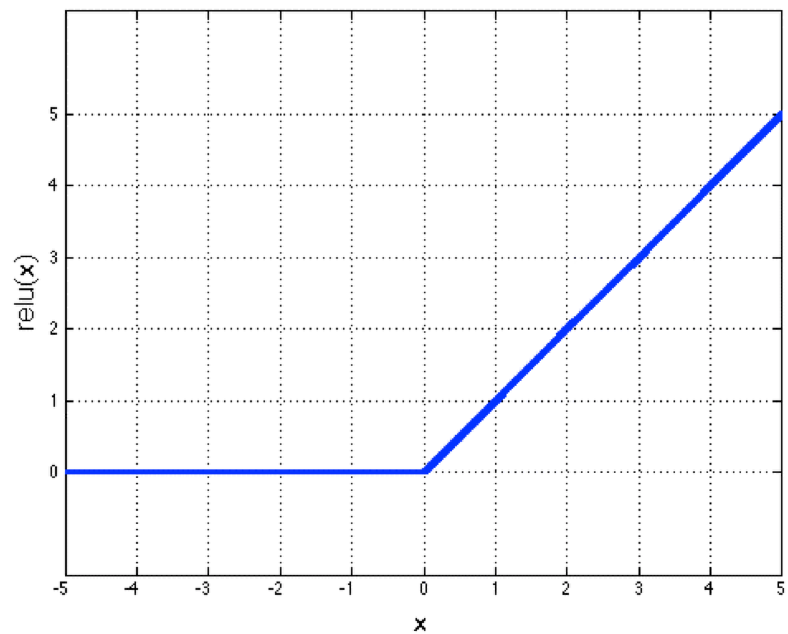
Feature Map

slide credit: S. Lazebnik

Convolutional Neural Networks

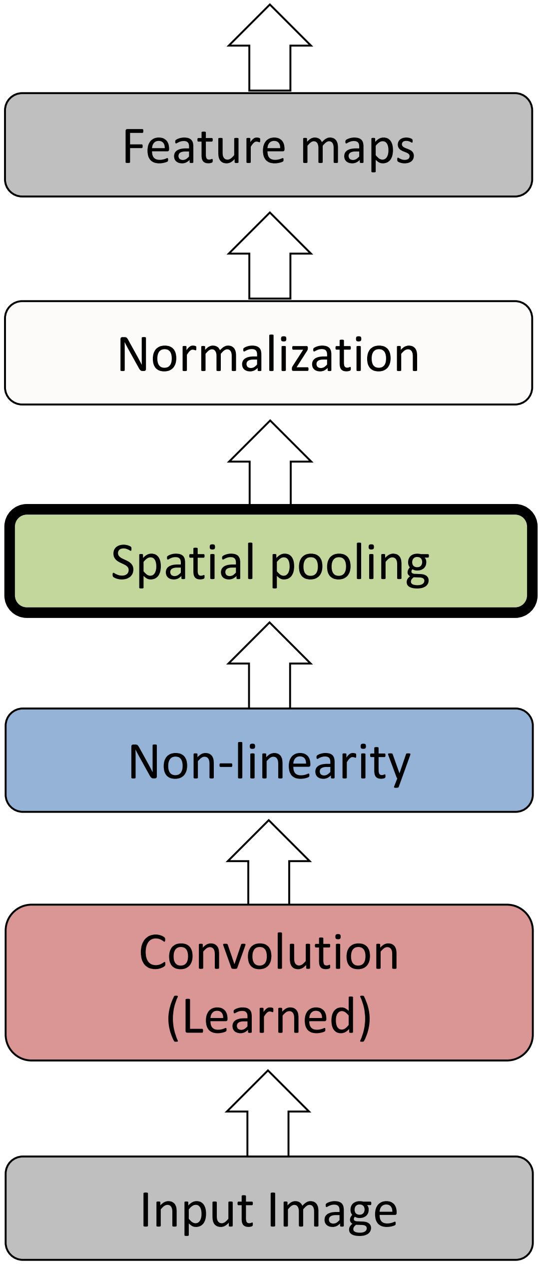


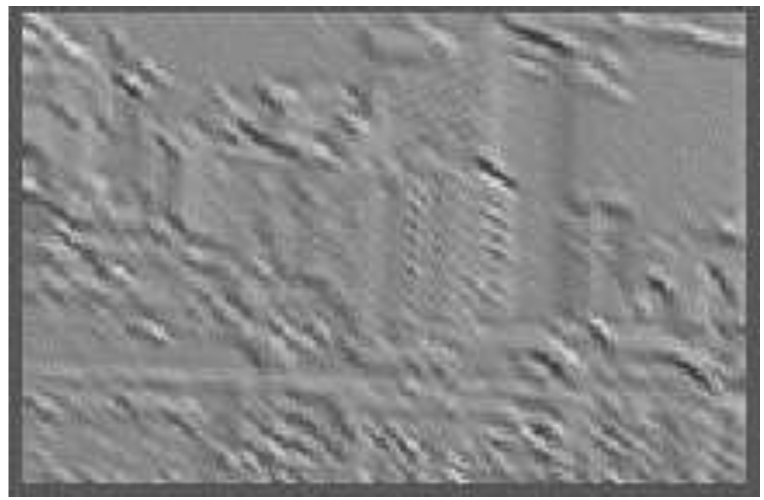
Rectified Linear Unit (ReLU)



slide credit: S. Lazebnik

Convolutional Neural Networks



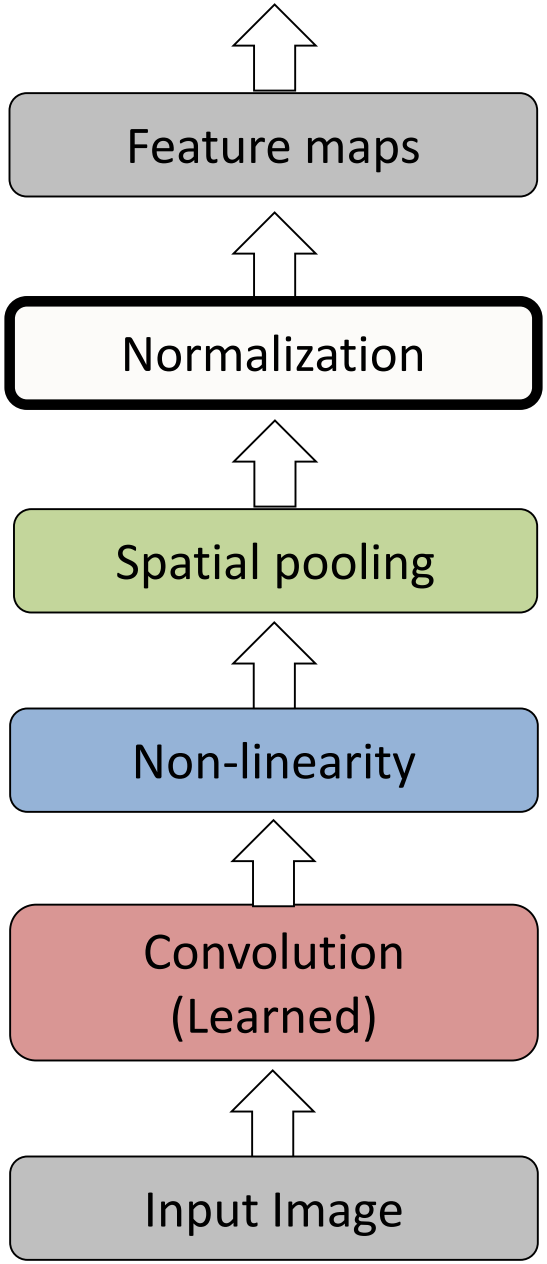


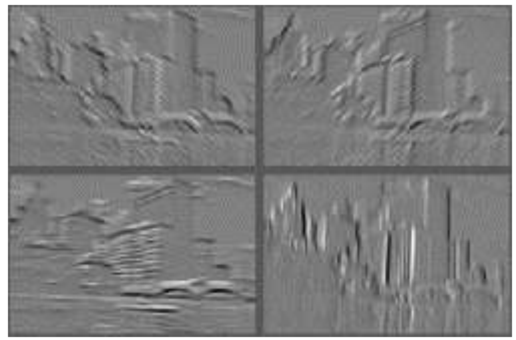
Max pooling

Max-pooling: a non-linear down-sampling

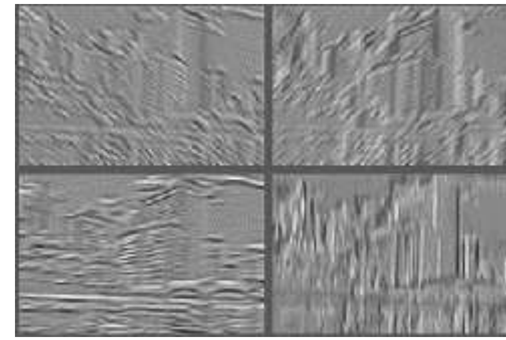
Provide *local invariance*

Convolutional Neural Networks





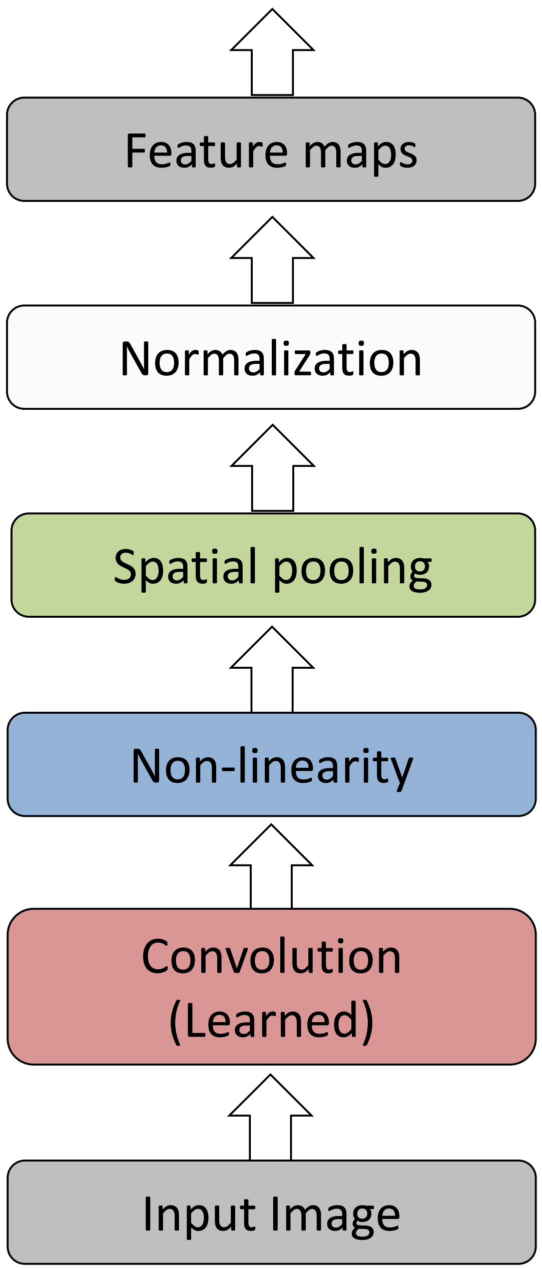
Feature Maps



Feature Maps
After Contrast
Normalization

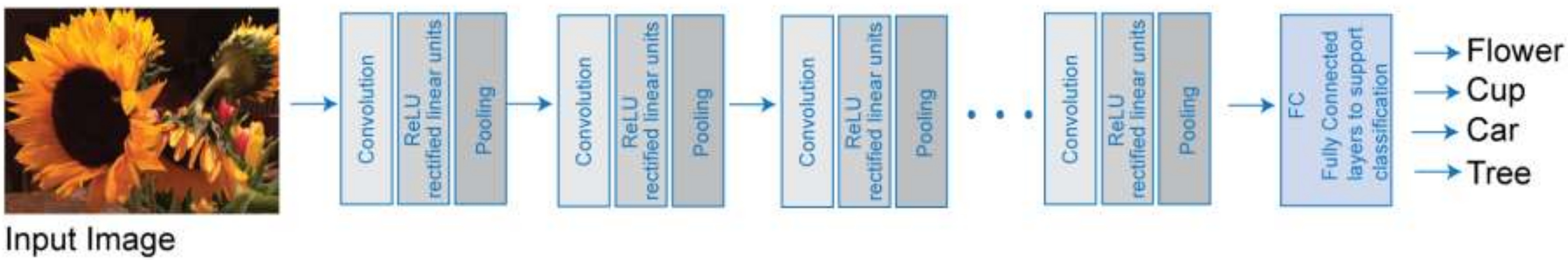
slide credit: S. Lazebnik

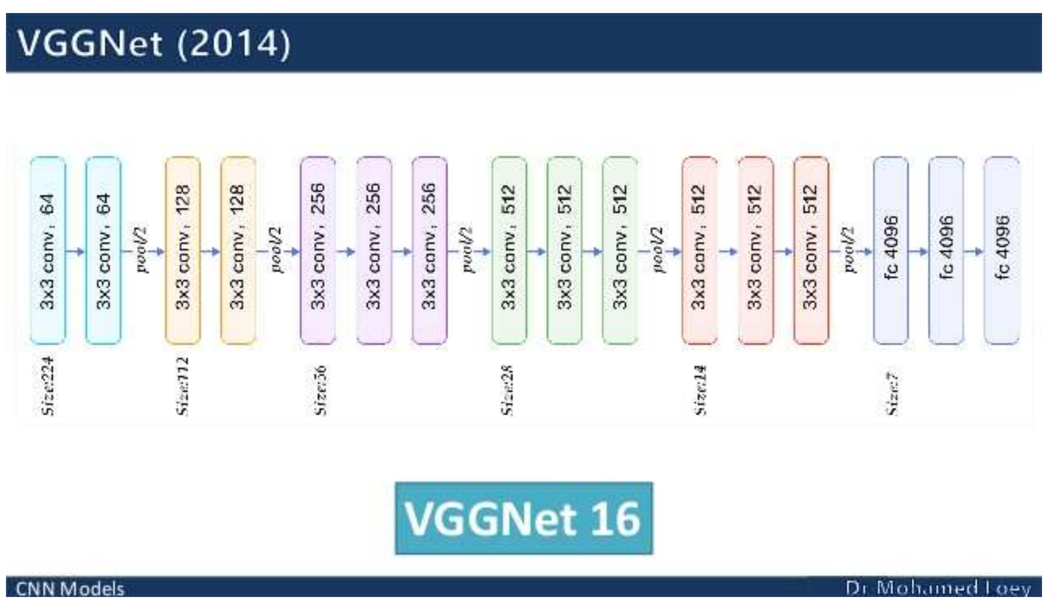
Convolutional Neural Networks



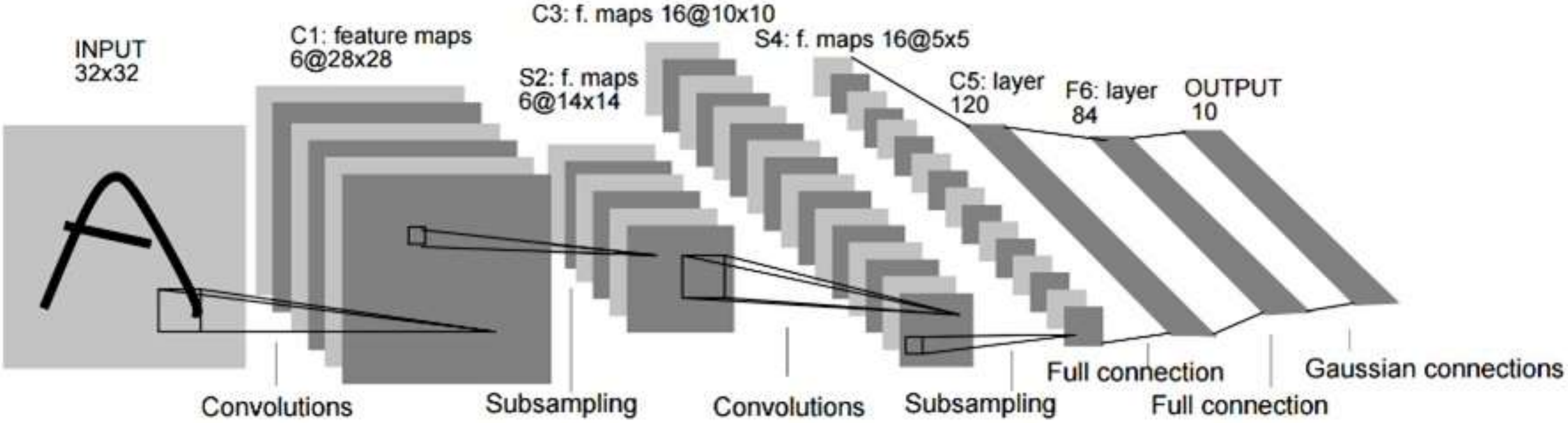
slide credit: S. Lazebnik

Standard ConvNets Architecture





LeNet [LeCun et al. 1998]

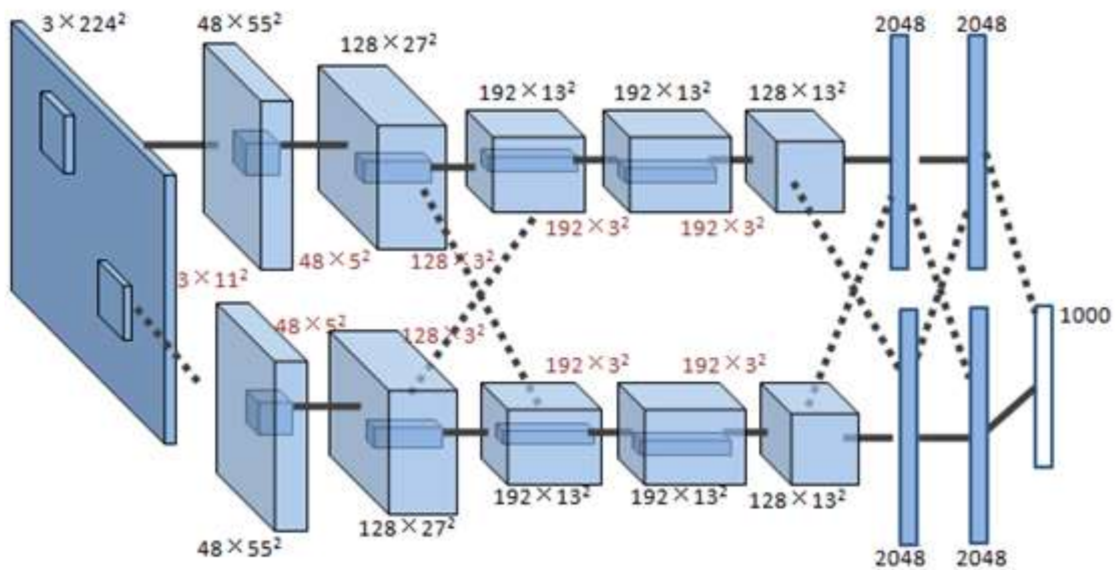


LeNet-1 from 1993

Gradient-based learning applied to document recognition [[LeCun, Bottou, Bengio, Haffner 1998]]

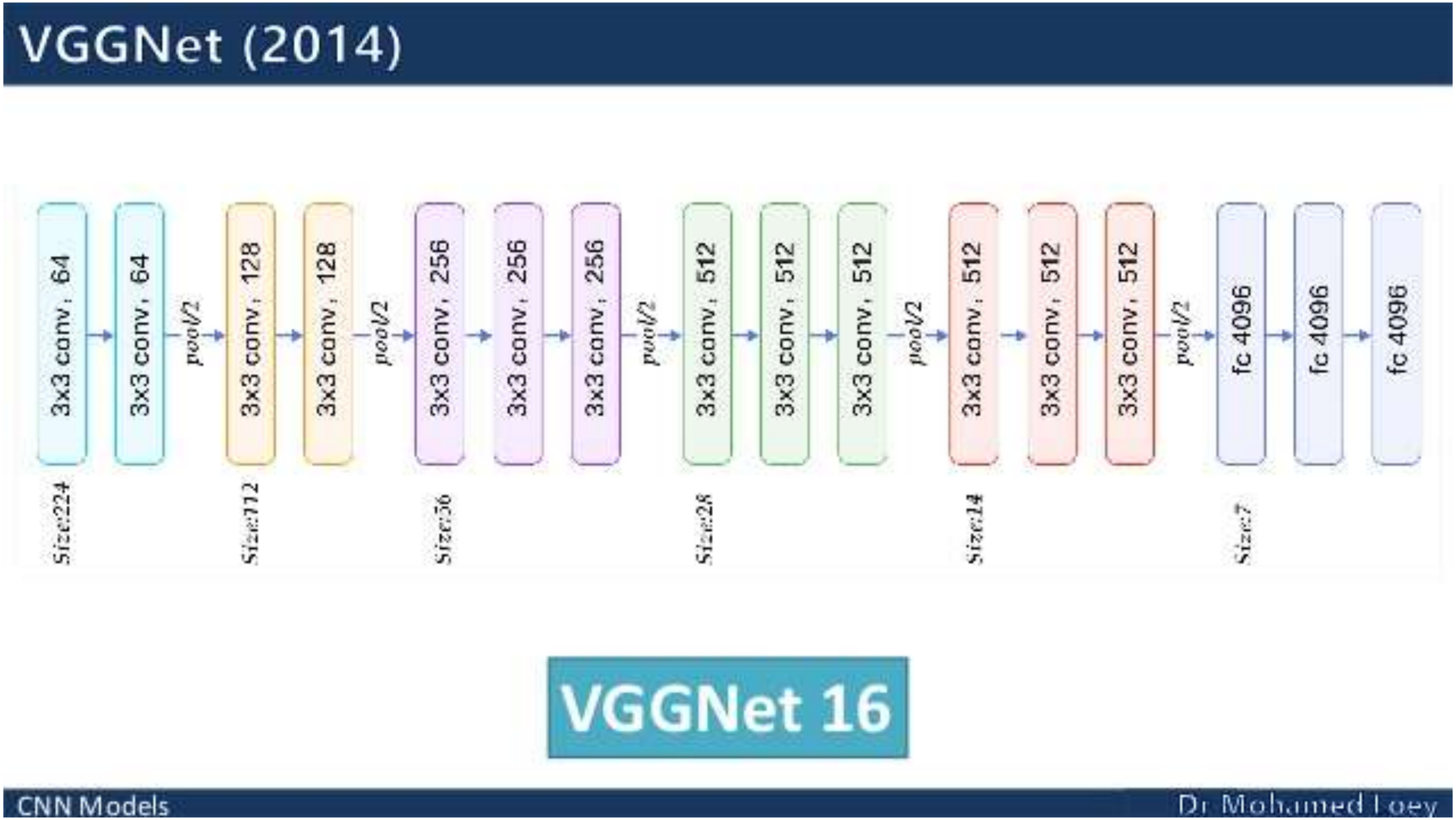
AlexNet (2013 ImageNet Winner)

- Similar framework to LeCun'98 but:
  - Bigger model (8 hidden layers, 650,000 units, 60,000,000 params)
  - More data (10^6 vs. 10^3 images)
  - GPU implementation (50x speedup over CPU)
    - Trained on two GPUs for a week

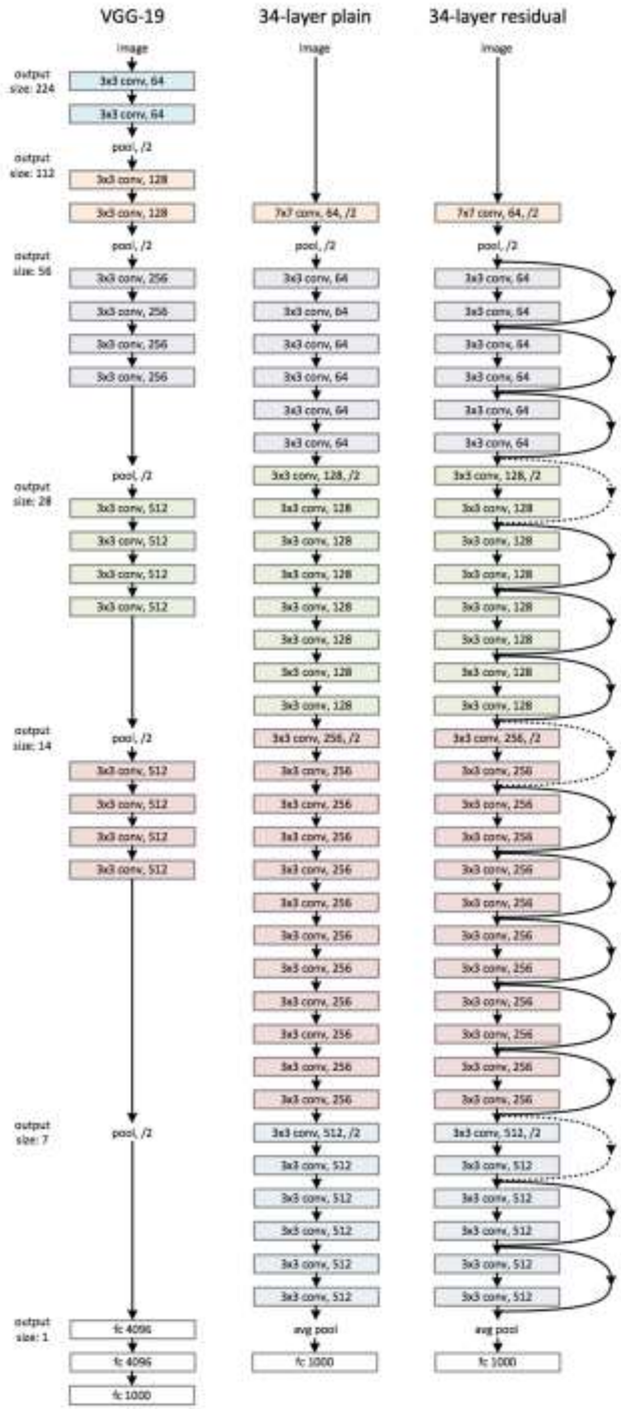


A. Krizhevsky, I. Sutskever, and G. Hinton,
[ImageNet Classification with Deep Convolutional Neural Networks](), NIPS 2012

VGG Net (2014 ImageNet Winner)

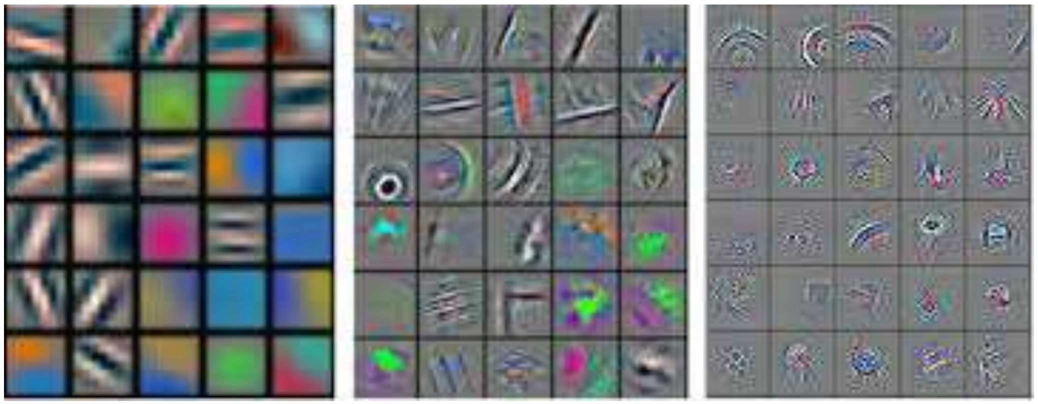


AlexNet and VGGNet



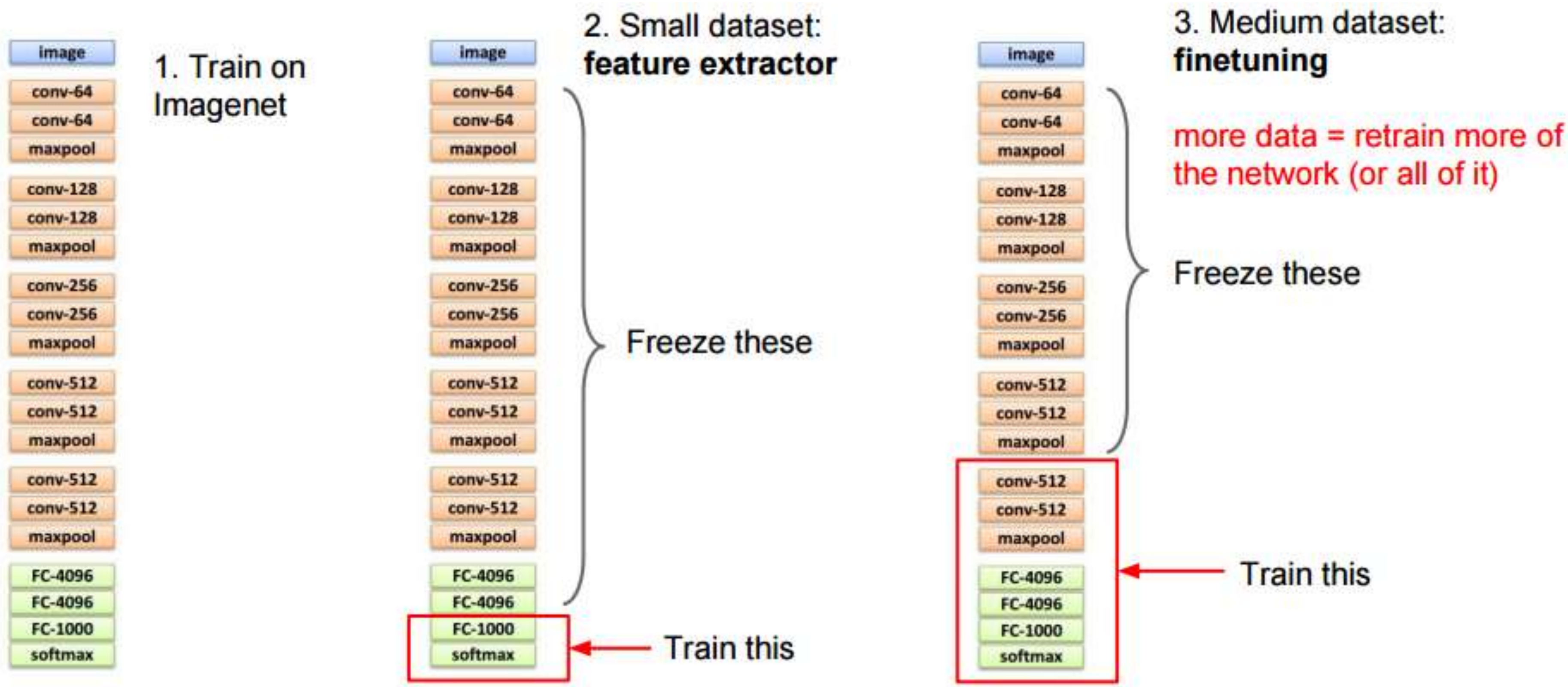
VGG-19
34-layer plain
34-layer residual
image
output size: 224
3x3 conv, 64
3x3 conv, 64
pool, /2
output size: 112
3x3 conv, 128
3x3 conv, 128
pool, /2
output size: 56
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
pool, /2
output size: 28
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
output size: 14
pool, /2
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
output size: 7
pool, /2
output size: 1
fc 4096
fc 4096
fc 1000
image
7x7 conv, 64, /2
pool, /2
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 128, /2
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 256, /2
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 512, /2
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
avg pool
fc 1000
image
7x7 conv, 64, /2
pool, /2
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 128, /2
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 256, /2
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 512, /2
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
avg pool
fc 1000

Features



Transfer Learning

- Improvement of learning in a **new** task through the *transfer of knowledge* from a **related** task that has already been learned.
-



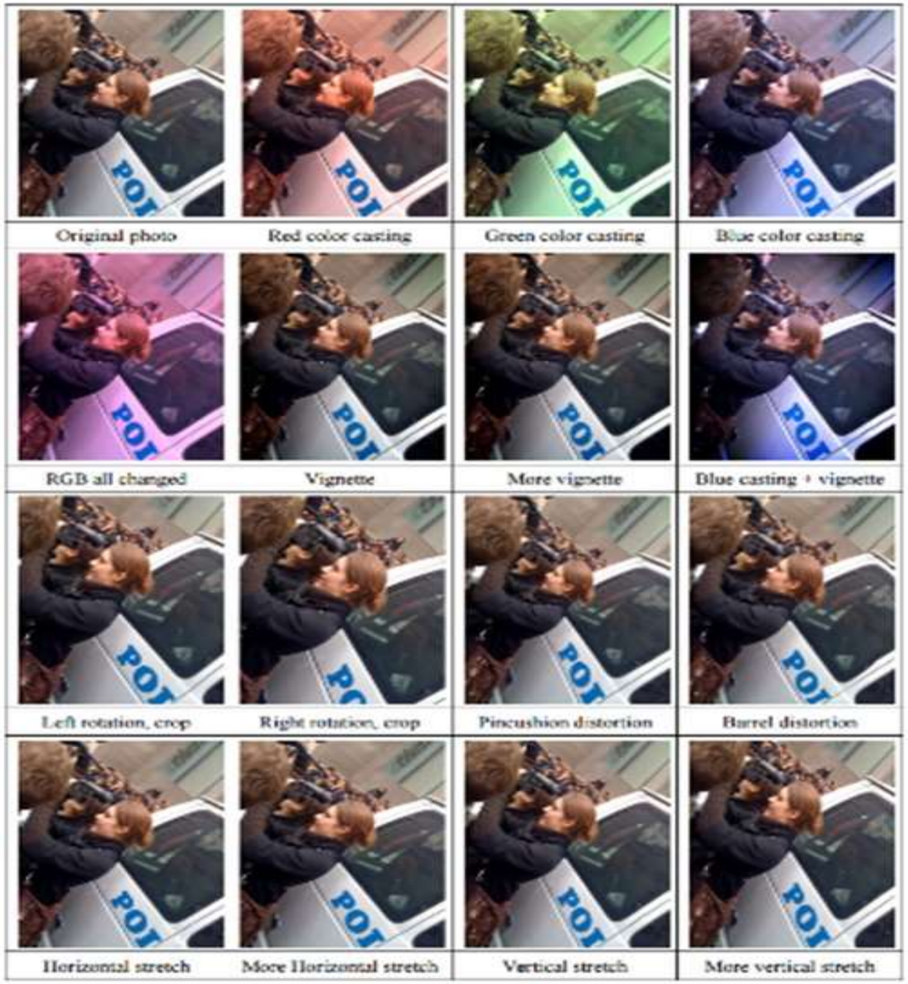
Learning and Transferring Mid-Level Image Representations using Convolutional Neural Networks [Oquab et al. CVPR 2014]

Training Convolutional Neural Networks

- Backpropagation + stochastic gradient descent with momentum
  - Neural Networks: Tricks of the Trade
- Data augmentation
- Dropout
- Batch normalization
- Initialization
  - Transfer learning

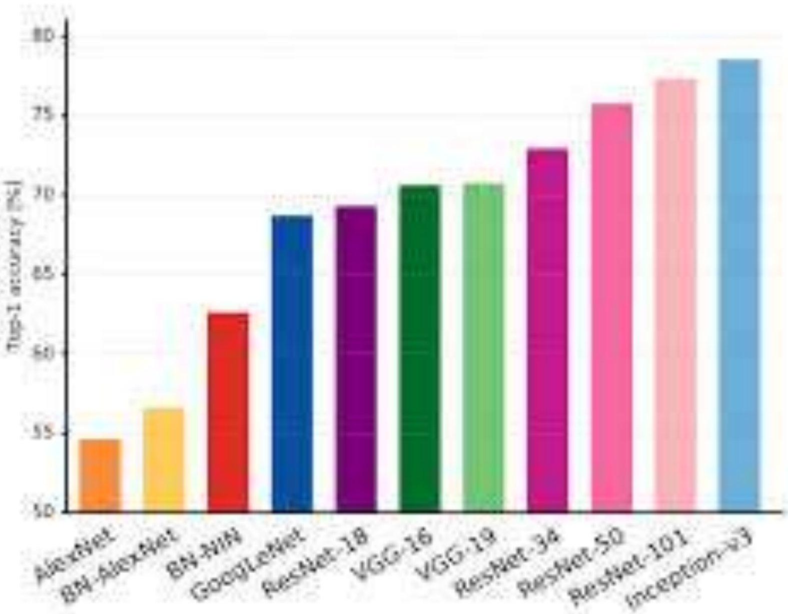
Data Augmentation (Jittering)

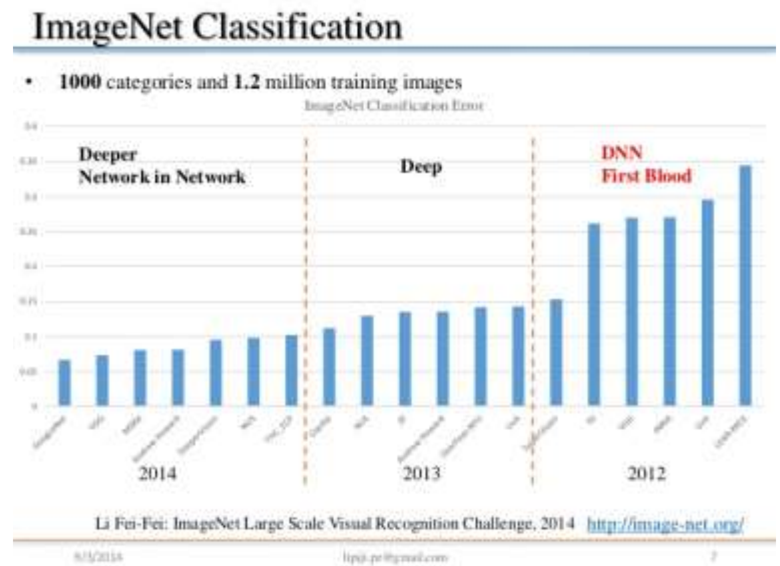
- Create *virtual* training samples
  - Horizontal flip
  - Random crop
  - Color casting
  - Geometric distortion

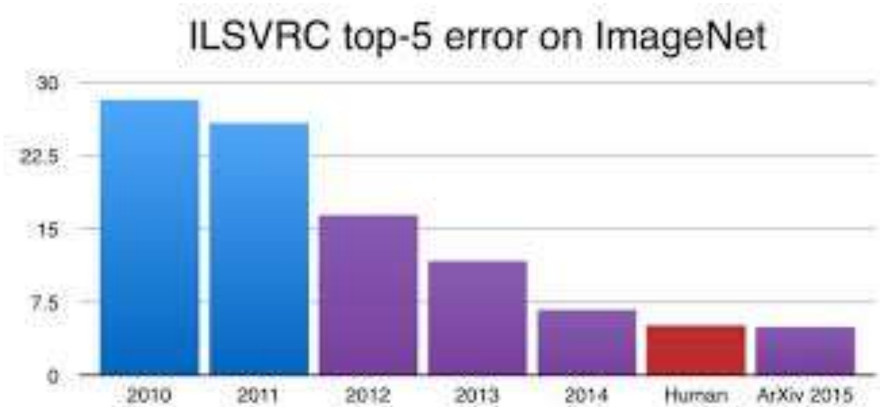


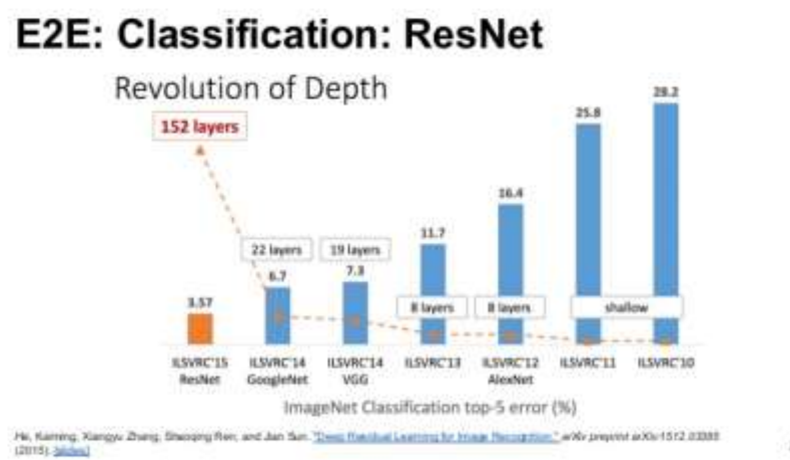
Deep Image [Wu et al. 2015]

ImageNet Challenge 2012-2016









Tools

- Caffe
- cuda-convnet2
- Torch
- MatConvNet
- Pylearn2
- TensorFlow

Today's class

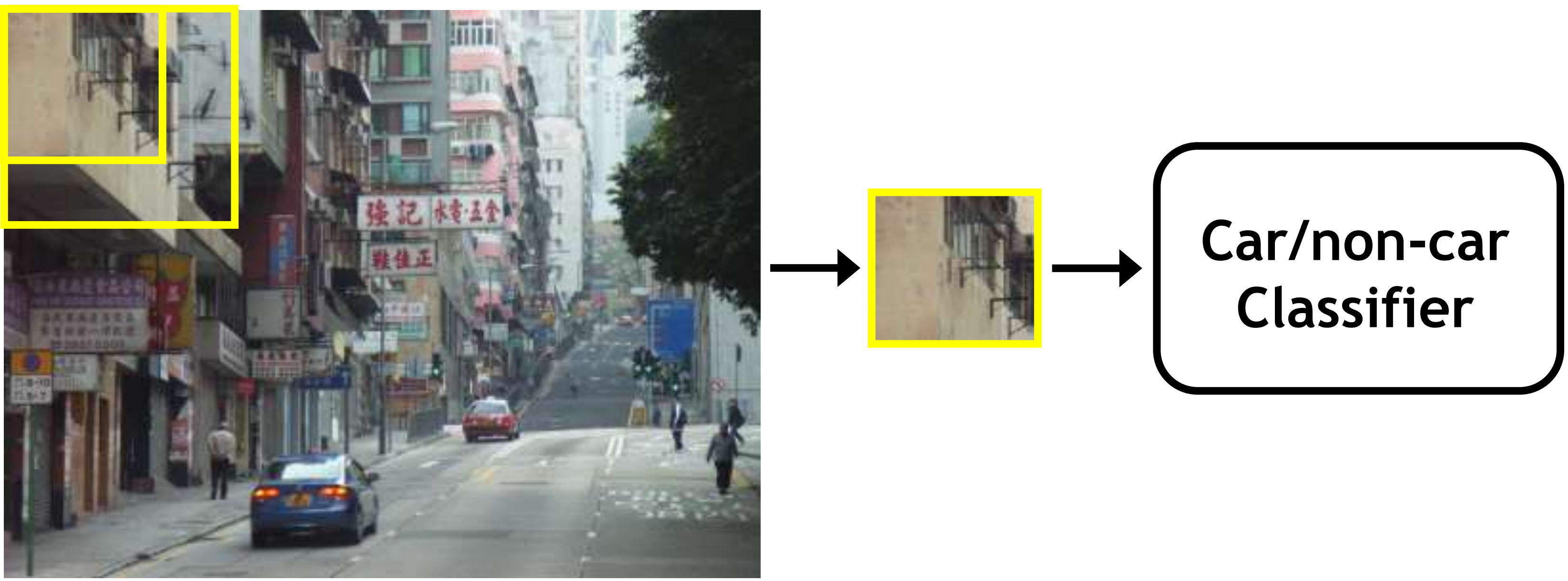
Object Detection

Stereo

3D Reconstruction

Window-based models

Generating and scoring candidates



Kristen Grauman

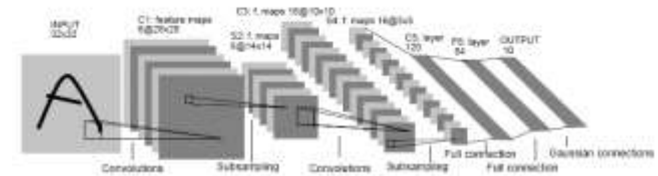
Discriminative Classifier Construction

Nearest neighbor

10^6 examples

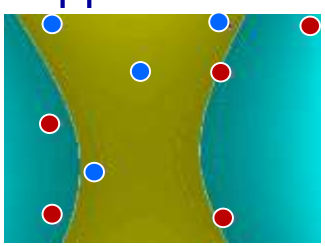
Shakhnarovich, Viola, Darrell 2003
Berg, Berg, Malik 2005...

Neural networks



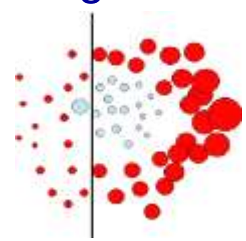
AlexNet 2013
VGGNet 2014
…

Support Vector Machines



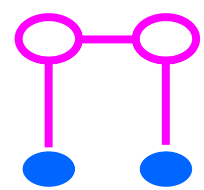
Guyon, Vapnik
Heisele, Serre, Poggio, 2001,…

Boosting



Viola, Jones 2001,
Torralba et al. 2004,
Opelt et al. 2006,…

Conditional Random Fields



McCallum, Freitag, Pereira 2000; Kumar, Hebert 2003
…

Slide adapted from Antonio Torralba

The Viola/Jones Face Detector

- A seminal approach to real-time object detection
- Training is slow, but detection is very fast
- Key ideas
  - *Integral images* for fast feature evaluation
  - *Boosting* for feature selection
  - *Attentional cascade* for fast rejection of non-face windows

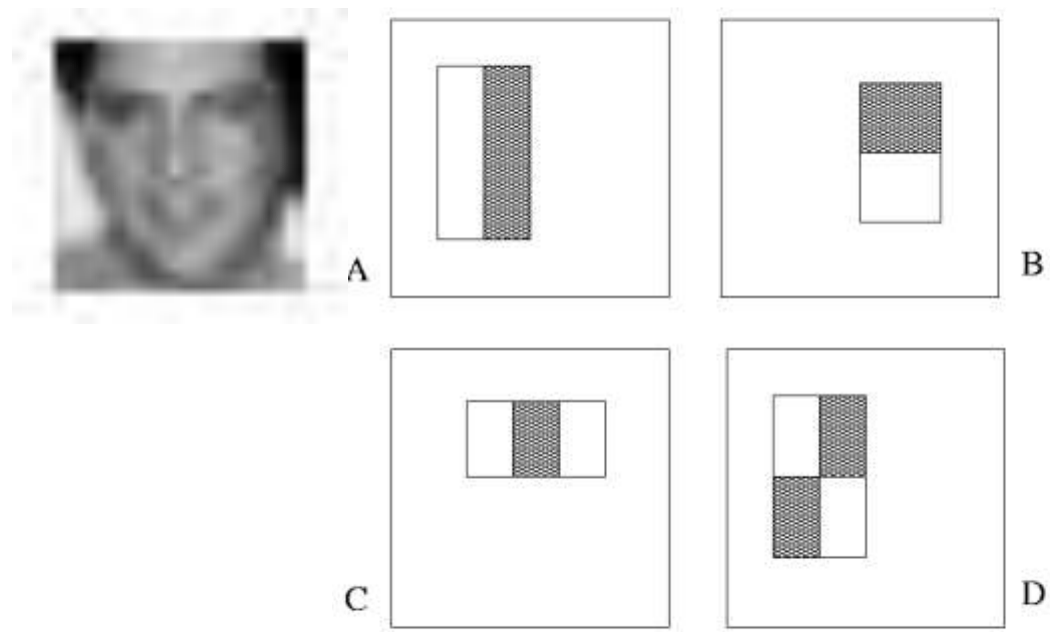
P. Viola and M. Jones. *Rapid object detection using a boosted cascade of simple features.* CVPR 2001.

P. Viola and M. Jones. *Robust real-time face detection.* IJCV 57(2), 2004.

~14000 citations!

Image Features

"Rectangle filters"

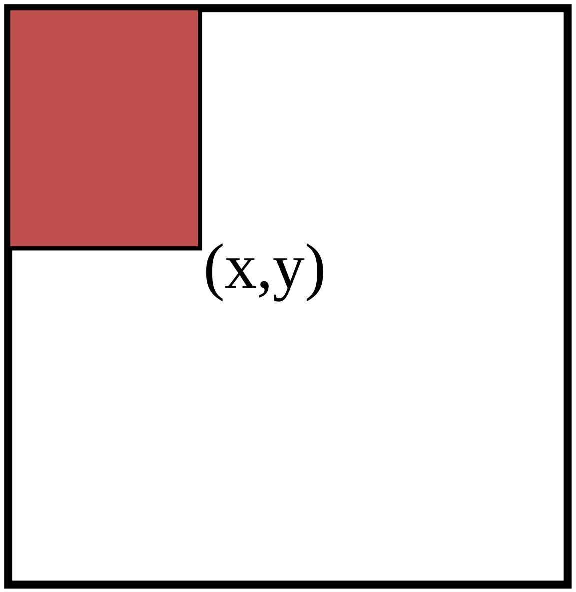


Value =

∑ (pixels in white area) –
∑ (pixels in black area)

Fast computation with integral images

- The *integral image* computes a value at each pixel (*x*,*y*) that is the sum of the pixel values above and to the left of (*x*,*y*), inclusive
- This can quickly be computed in one pass through the image

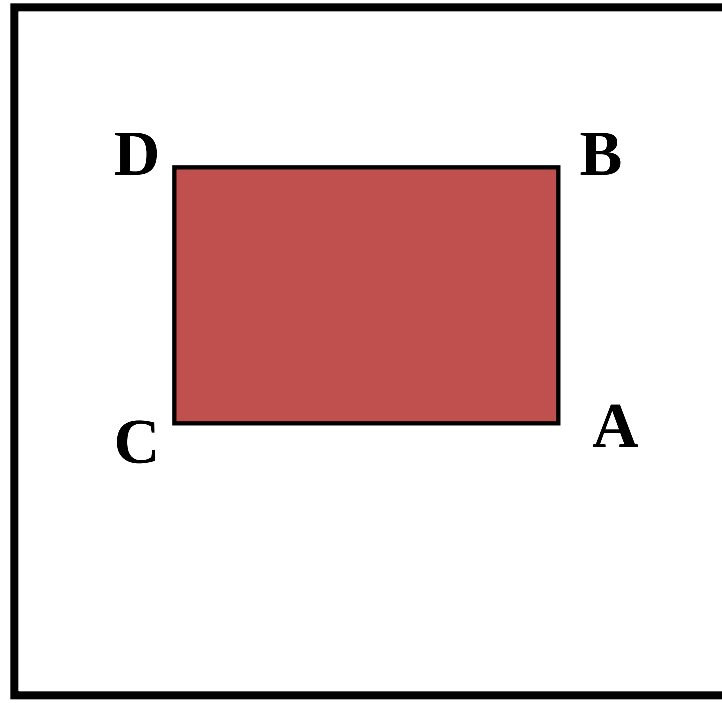


Computing Sum within a Rectangle

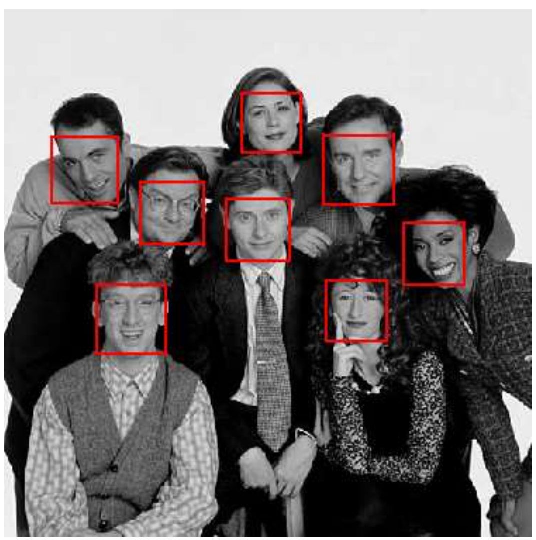
- Let A,B,C,D be the values of the integral image at the corners of a rectangle
- Then the sum of original image values within the rectangle can be computed as:

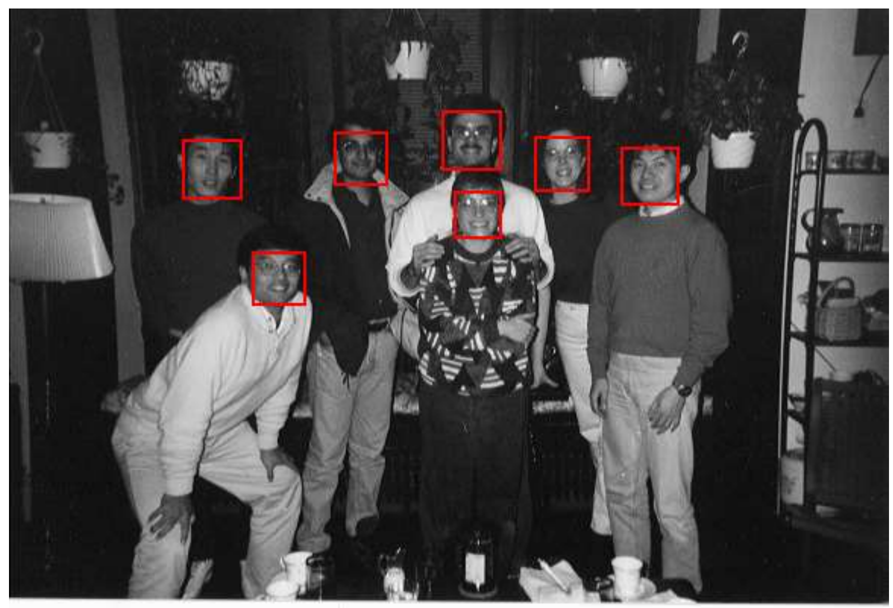
  sum = A – B – C + D

- Only 3 additions are required for any size of rectangle!



Output of Face Detector on Images





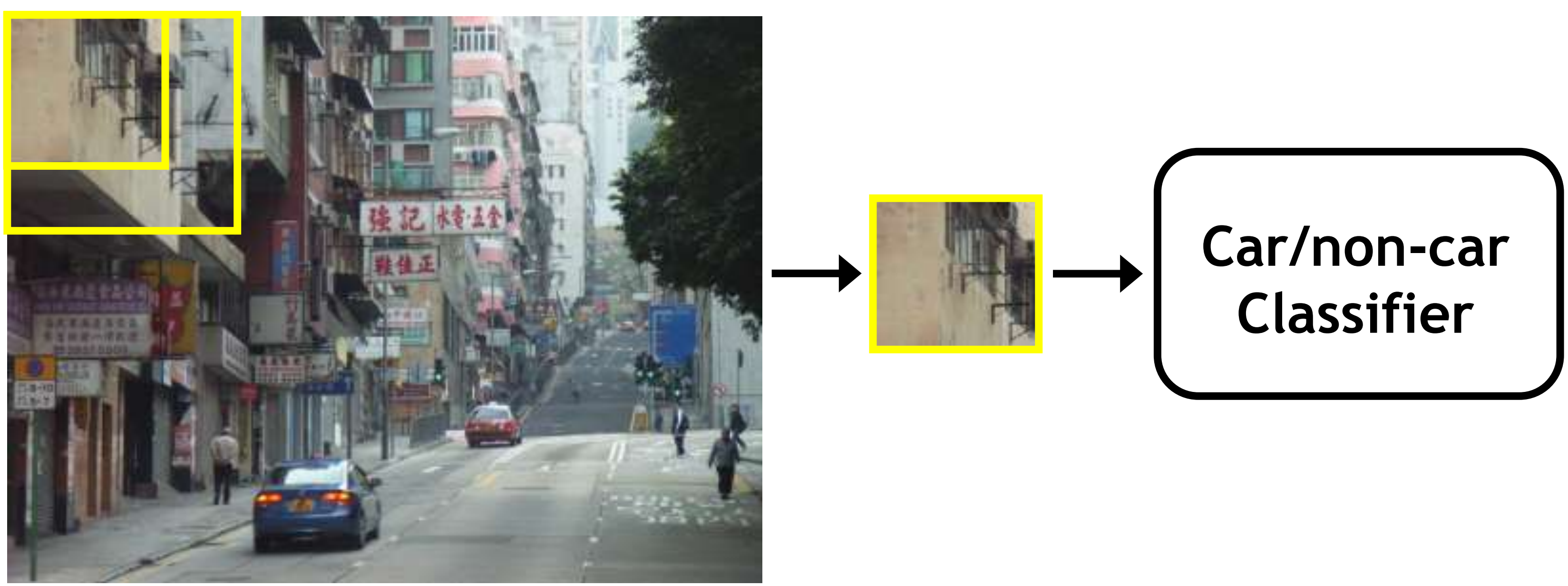
Profile Detection

Summary: Viola/Jones Detector

- Rectangle features
- Integral images for fast computation
- Boosting for feature selection
- Attentional cascade for fast rejection of negative windows

Window-based Detection: Strengths

- Sliding window detection and global appearance descriptors:
  - Simple detection protocol to implement
  - Good feature choices critical
  - Past successes for certain classes



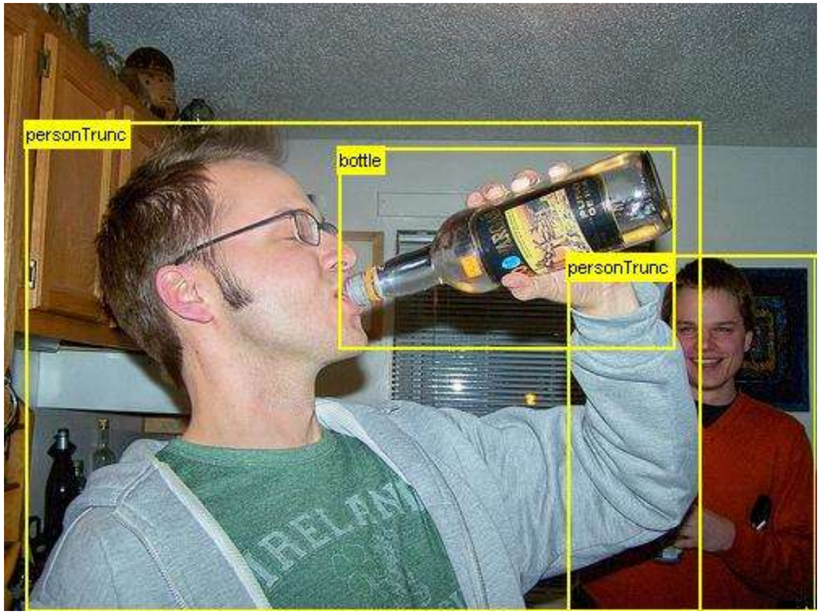
Window-based Detection: Limitations

- High computational complexity
  - For example: 250,000 locations x 30 orientations x 4 scales = 30,000,000 evaluations!
  - If training binary detectors independently, means cost increases linearly with number of classes
- With so many windows, false positive rate better be low

Kristen Grauman

Limitations (continued)

- Not all objects are “box” shaped



Kristen Grauman

Segmentation as Selective Search for Object Detection

Jasper Uijlings, Koen van de Sande, Theo Gevers, Arnold Smeulders:

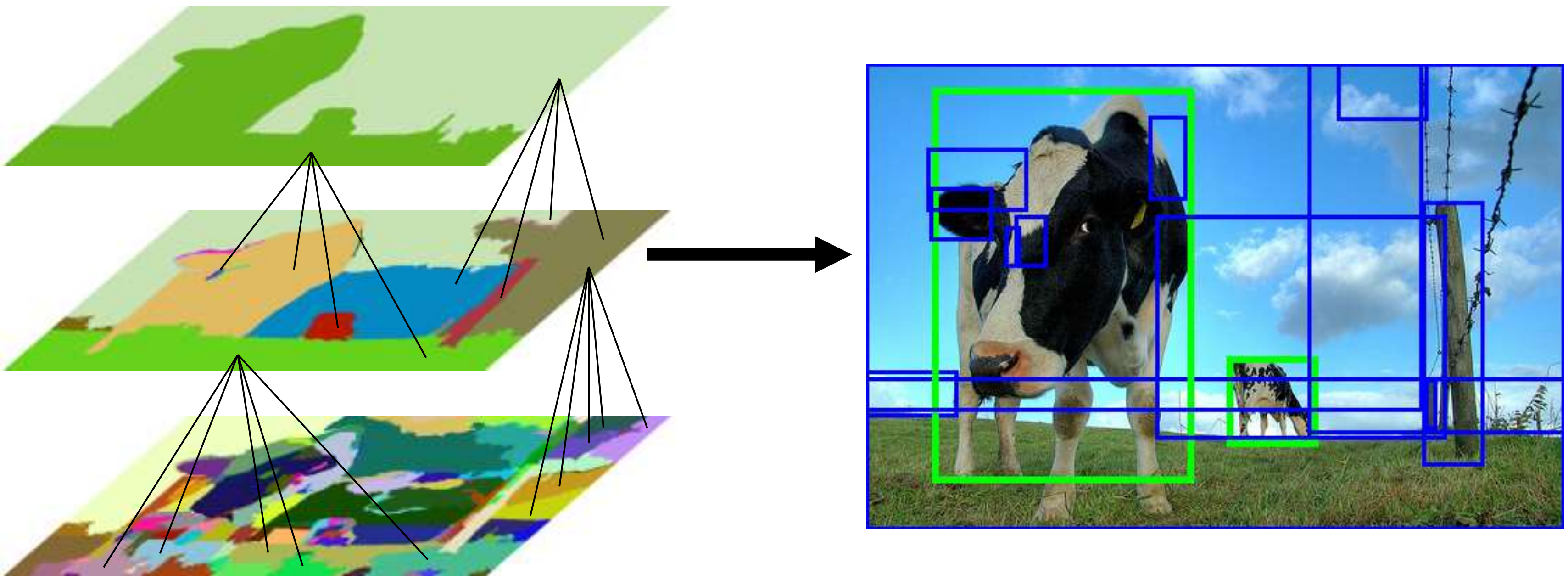
Selective Search for Object Recognition. International Journal of Computer Vision 104(2): 154-171 (2013)

Selective Search for Recognition

- Design criteria
  - High recall
  - Coarse locations are sufficient
    - ⇨ Bounding boxes
  - Fast to compute
    - ⇨ Efficient low-level features
    - ⇨ <10s per image

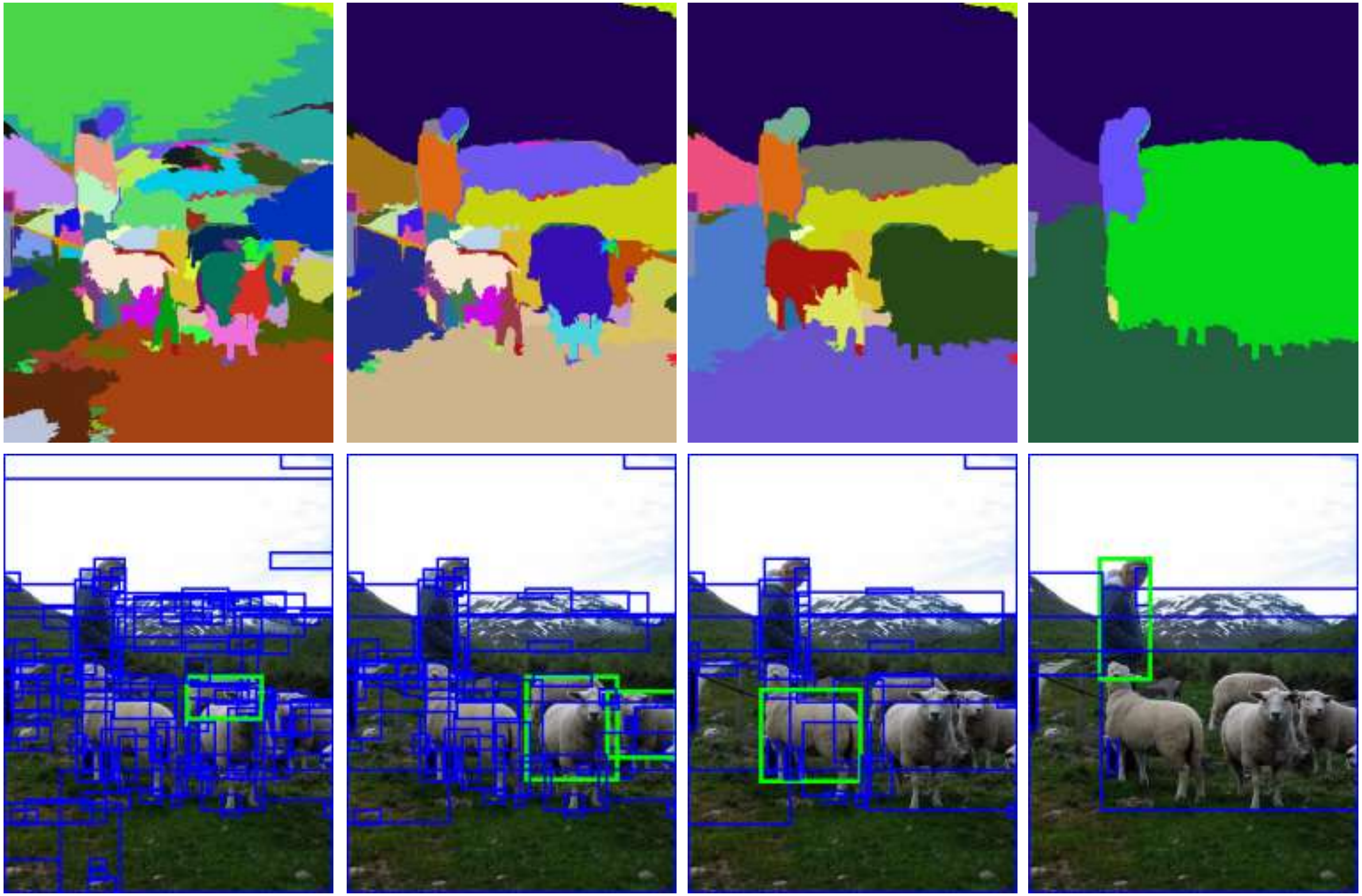
Selective Search: Approach

- Hypotheses based on hierarchical grouping



Group adjacent regions on color/texture cues

Example 1



Example 2

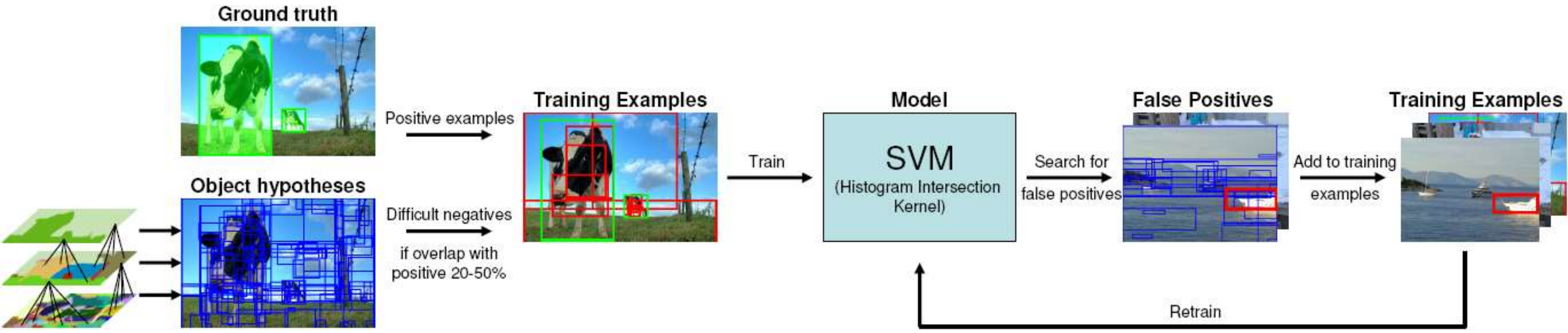
Selective Search on ILSVRC2011

- Apply to ILSVRC2011 train set
- Object hypotheses are class-independent

	ILSVRC2011 train
With bounding box annotations	315,525 images
Average #boxes/image	1,565
Average recall	98.5%

Localisation System Training

- Use positives and mirrored positives
- Use object hypotheses to create difficult initial negatives (at most 7,500)
- Add 2 iterations of false positives (from 4,000 images)



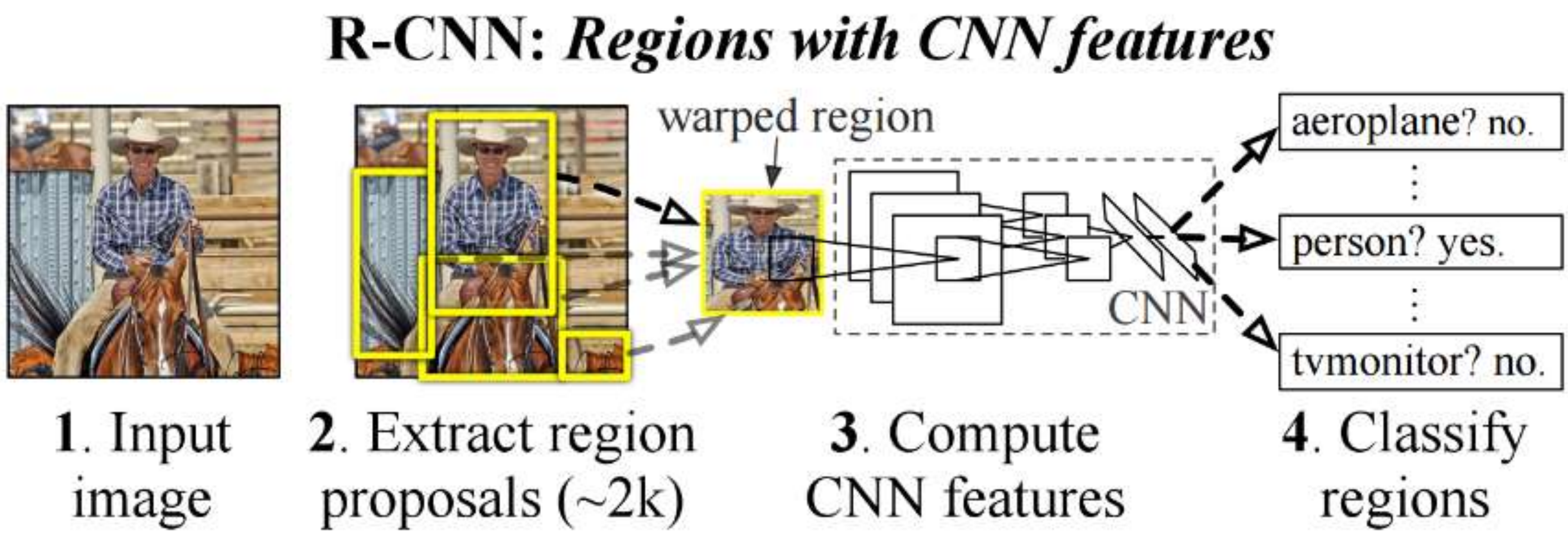
- Features: Bag-of-words, sample every pixel, SIFT, “ColorSIFT” and RGB-SIFT, pyramid up to level 3, codebook size 4096
- Histogram Intersection Kernel with Fast Approximation

Summary

- Adopted segmentation as selective search strategy for object localisation:
  - High recall: >96% with ~1,500 locations
  - Coarse locations are sufficient: bounding boxes
  - Fast to compute: <10s per image
  - Class-independent
  - Enables the use of bag-of-words features

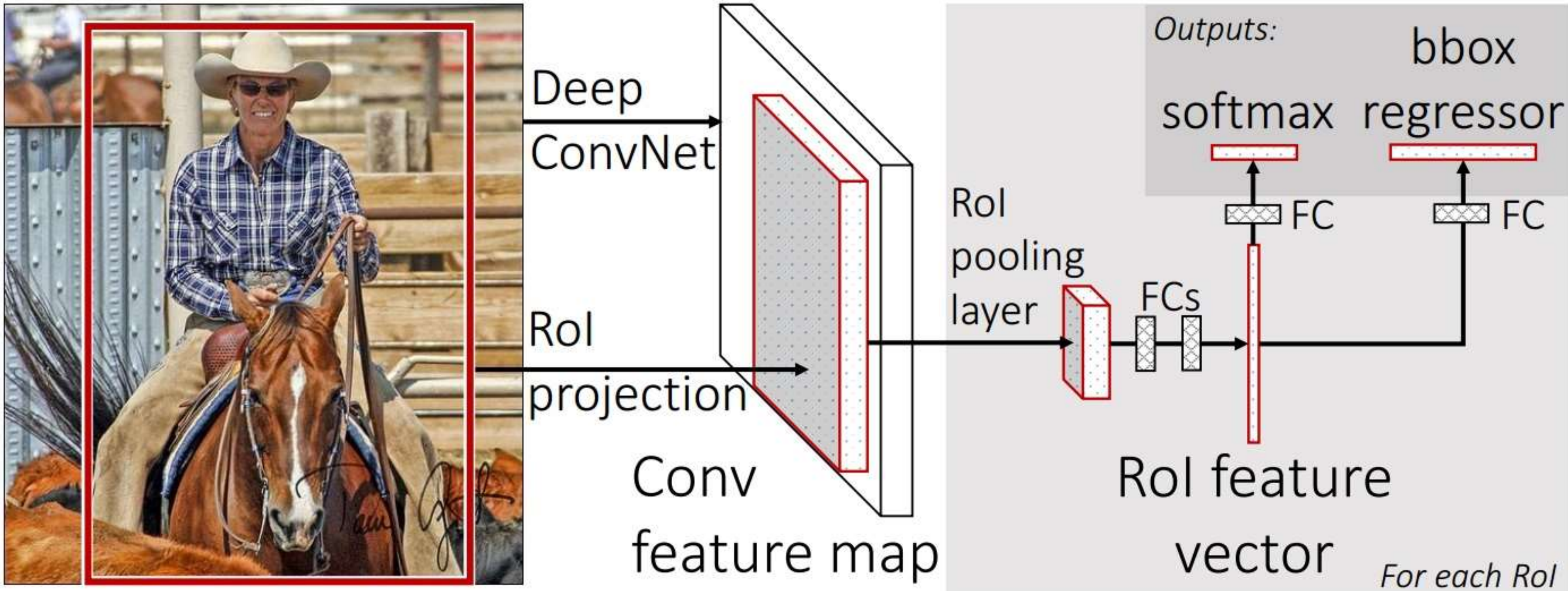
R-CNN: Regions with CNN features

- Trained on ImageNet classification
- Finetune CNN on PASCAL



RCNN [Girshick et al. CVPR 2014]

Fast R-CNN



Fast RCNN [[Girshick, R 2015]()]
https://github.com/rbgirshick/fast-rcnn

Faster R-CNN

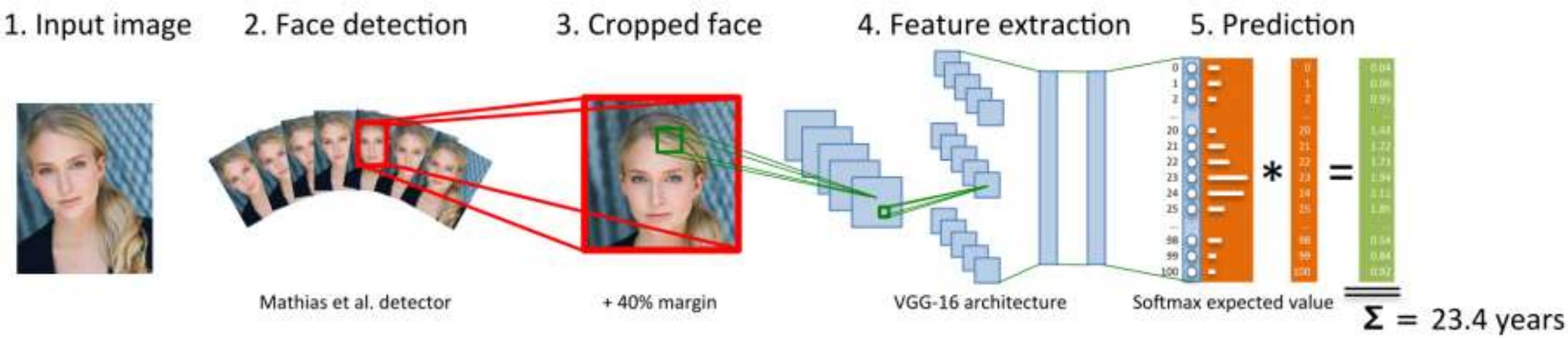


Figure 5. Sample detection results on the FDDB dataset, where green bounding boxes are ground-truth annotations and red bounding boxes

Application:
Deap Learning Image Intrinsics

Recovering Lightness: Retinex

- Image Intensity: $I = e\rho$

- Take Logarithm: $\log I = \log e + \log \rho$

  OR $I = e + \rho$

- Use Laplacian:

$$d = \nabla^2 I = \nabla^2 e + \nabla^2 \rho \qquad \nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$$

- Sharp changes in reflectance ρ

  $\nabla^2 \rho$ has 2 infinite spikes near edges and $\nabla^2 \rho = 0$ elsewhere

- Smooth changes in illumination e

  $\nabla^2 e \approx 0$ everywhere

Solving the Inverse Problem

Image	Laplacian	Thresholding	Lightness
$I = e + \rho$	$d = \nabla^2 I = \nabla^2 e + \nabla^2 \rho$	$t = T(d) \approx \nabla^2 \rho$	$l(x, y)$

Image $\rightarrow$ Laplacian $\rightarrow$ Thresholding $\rightarrow$ Lightness

Find lightness *l(x,y)* from *t(x,y):*

Poissonʼs Equation

$$\nabla^2 l = t$$

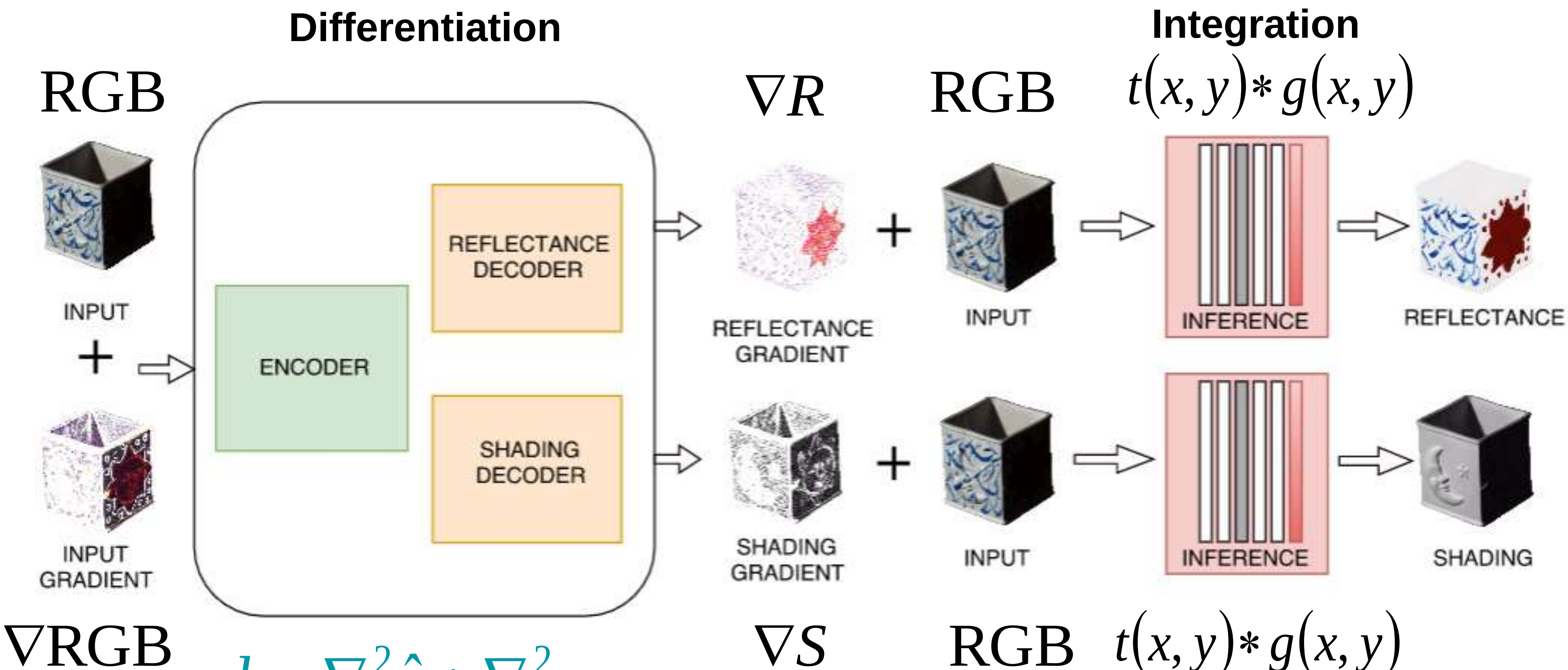
$$\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right) l(x, y) = t(x, y)$$

We have to find *g(x,y)* which satisfies

$$l(x, y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} t(u, v) g(x - u, y - v) du dv$$

$$\boxed{l(x, y) = t(x, y) * g(x, y)}$$

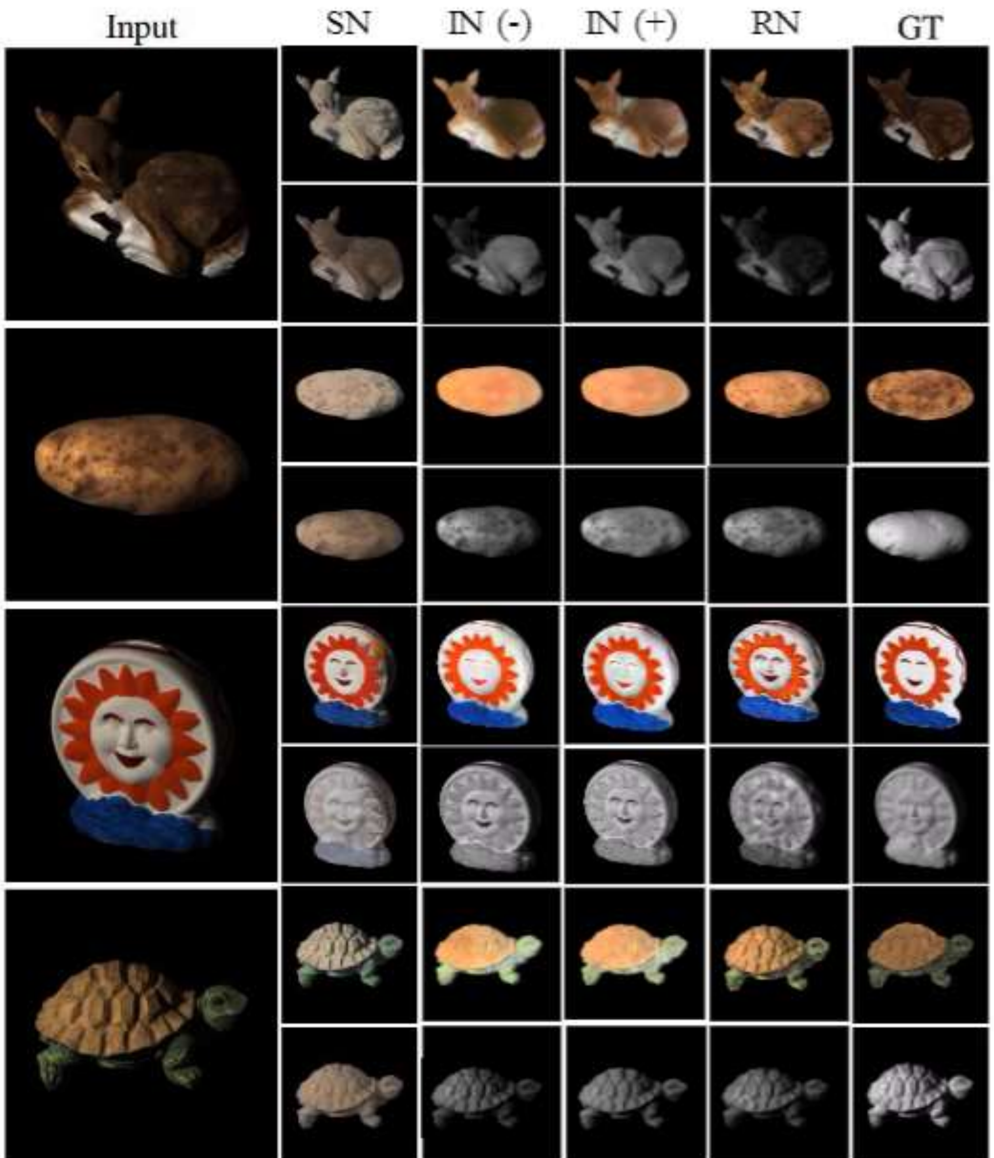
RetiNet: Retinex-Inspired ConvNet

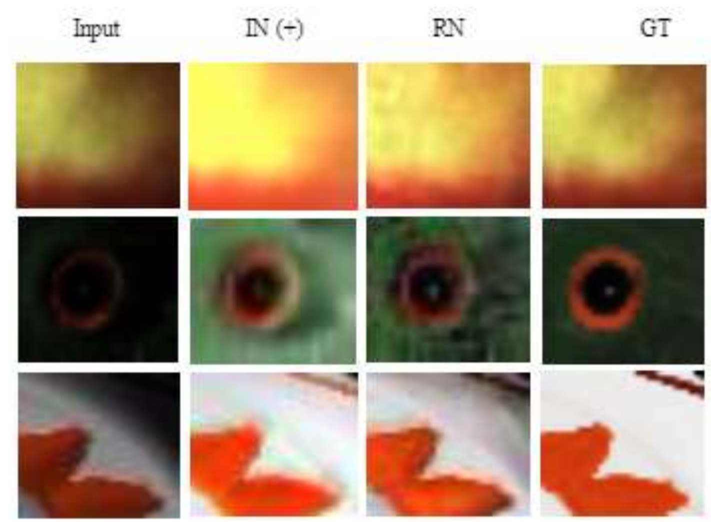


$$d = \nabla^2 \hat{e} + \nabla^2 \rho$$

$$d = \nabla R + \nabla S$$

Qualitative Results: MIT Dataset





RetiNet: Conclusion

Advantage

- Can capture most of the color information & shading
- Colors are more vivid and eliminates most of the color artifacts
- Fast and memory efficient
- Sharp Edges

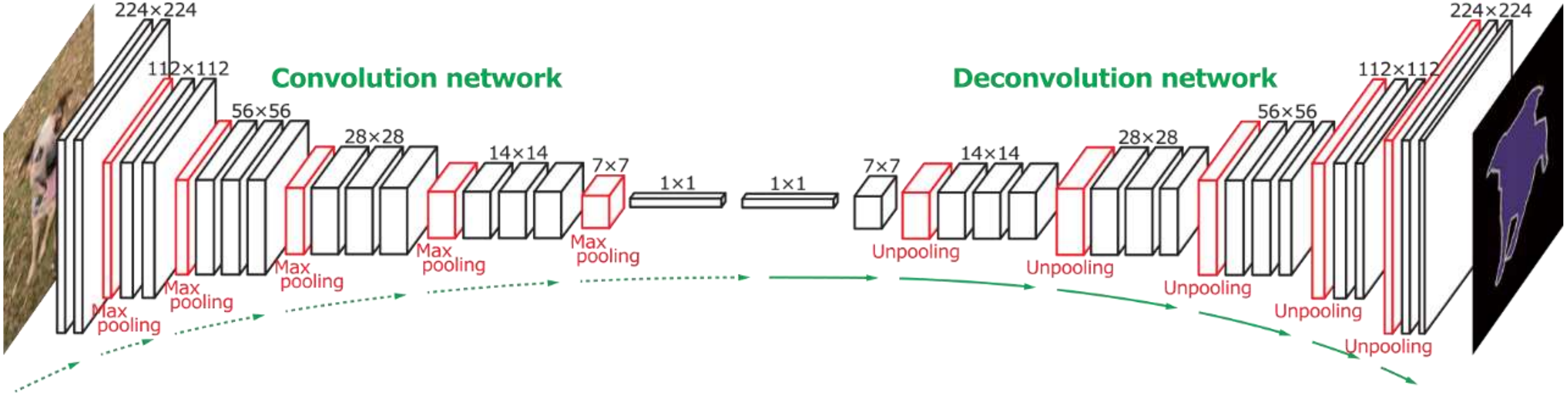
Application

- Semantic Image Segmenation

Application:
Semantic Segmentation and Style Transfer

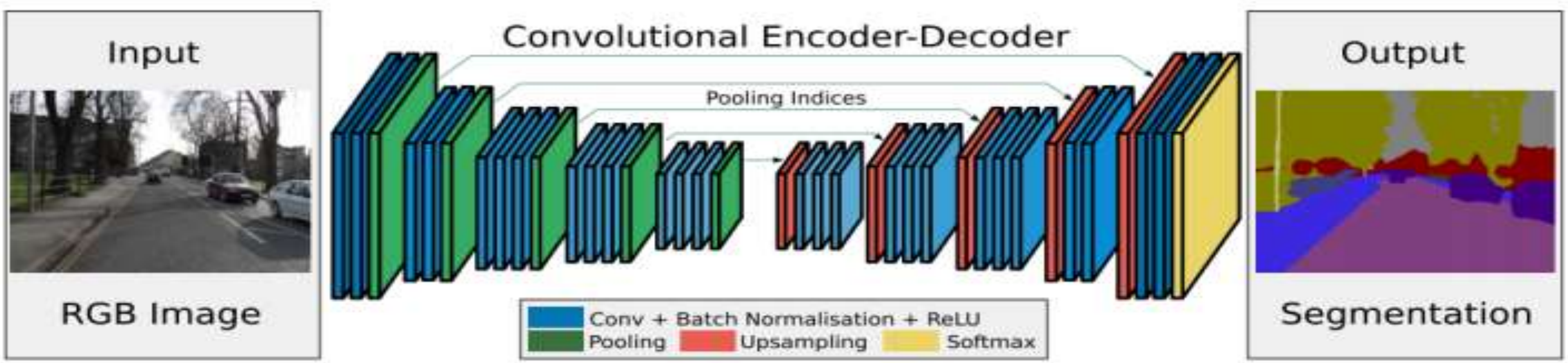
Semantic Segmentation

- Deep fully convolution network: SegNet
  - Encoder: learn low-level features in images
  - Decoder: reconstruct the image structure with labels



Badrinarayanan, et al., “SegNet: A deep convolutional Encoder-Decoder Architecture for Image Segmentation

Semantic Segmentation: SegNet



Badrinarayanan, et al., “SegNet: A deep convolutional Encoder-Decoder Architecture for Image Segmentation.

Style Transfer

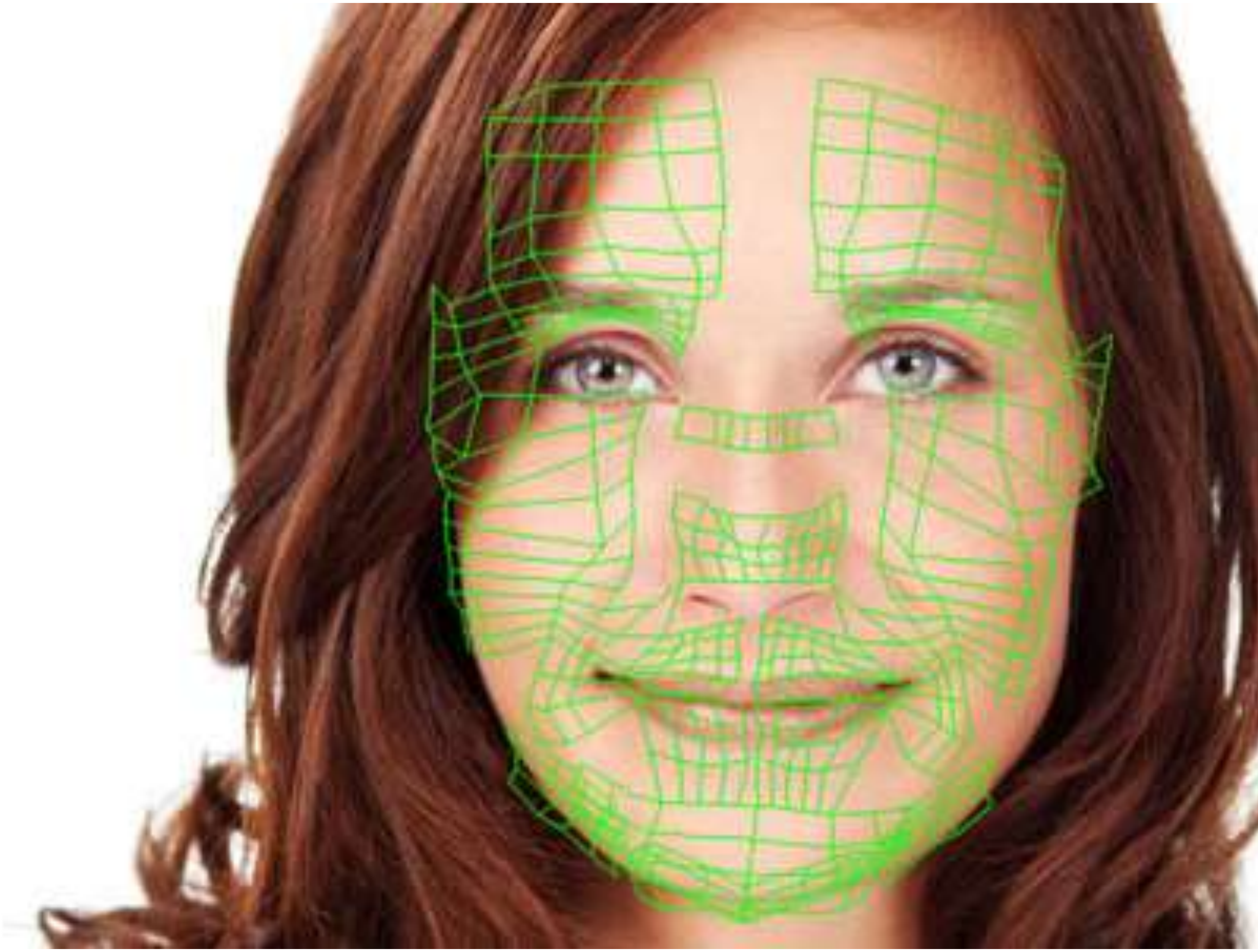
Application:
Face Analysis

Emotions

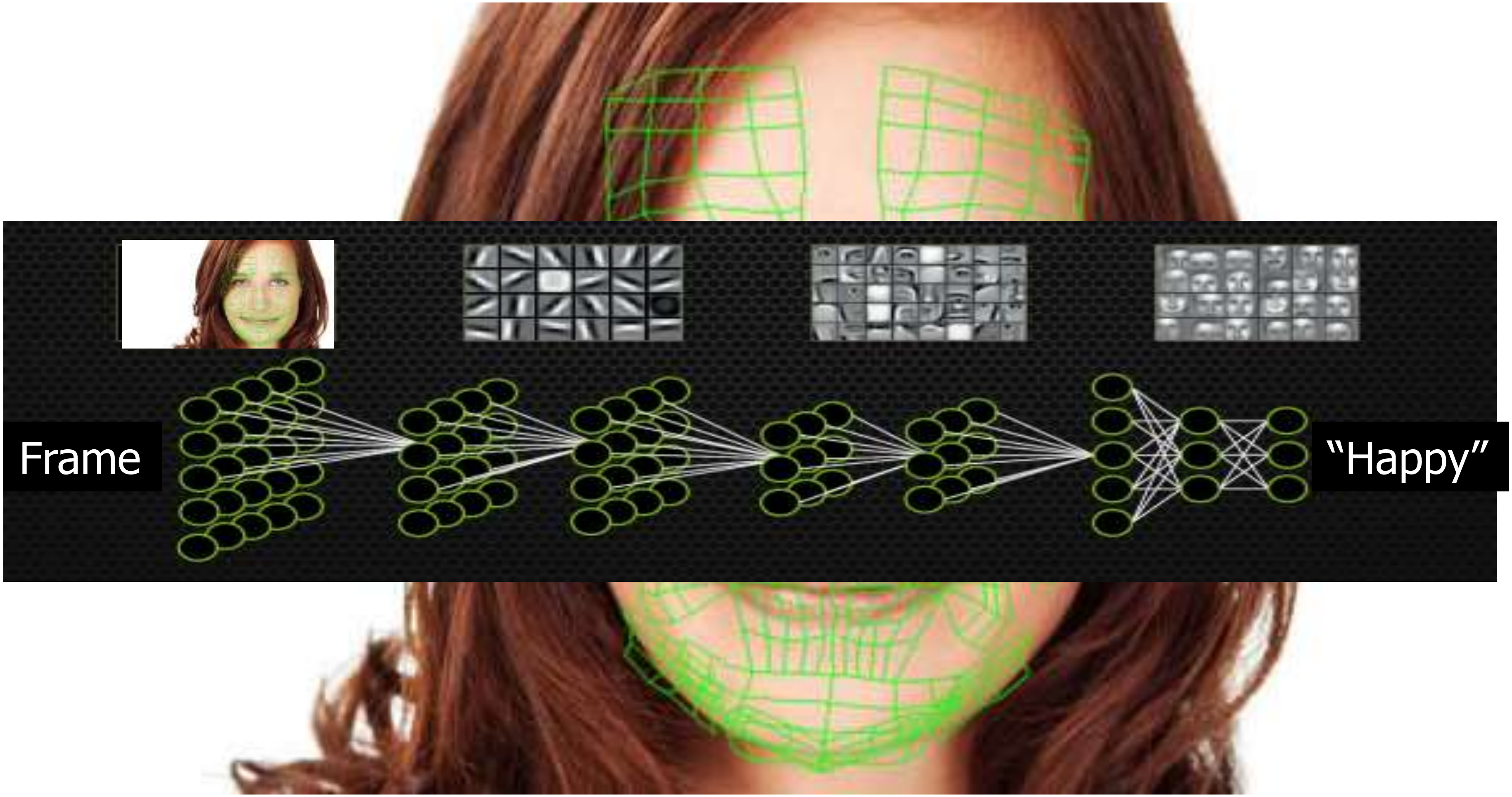
Facial Action Units

- Charles Darwin, 1872
- Ekman & Friesen, 1978

Face Analysis

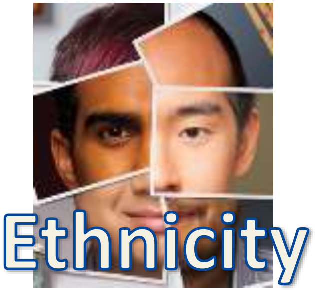


Face Analysis by Deep Learning



Sightcorp


Gender

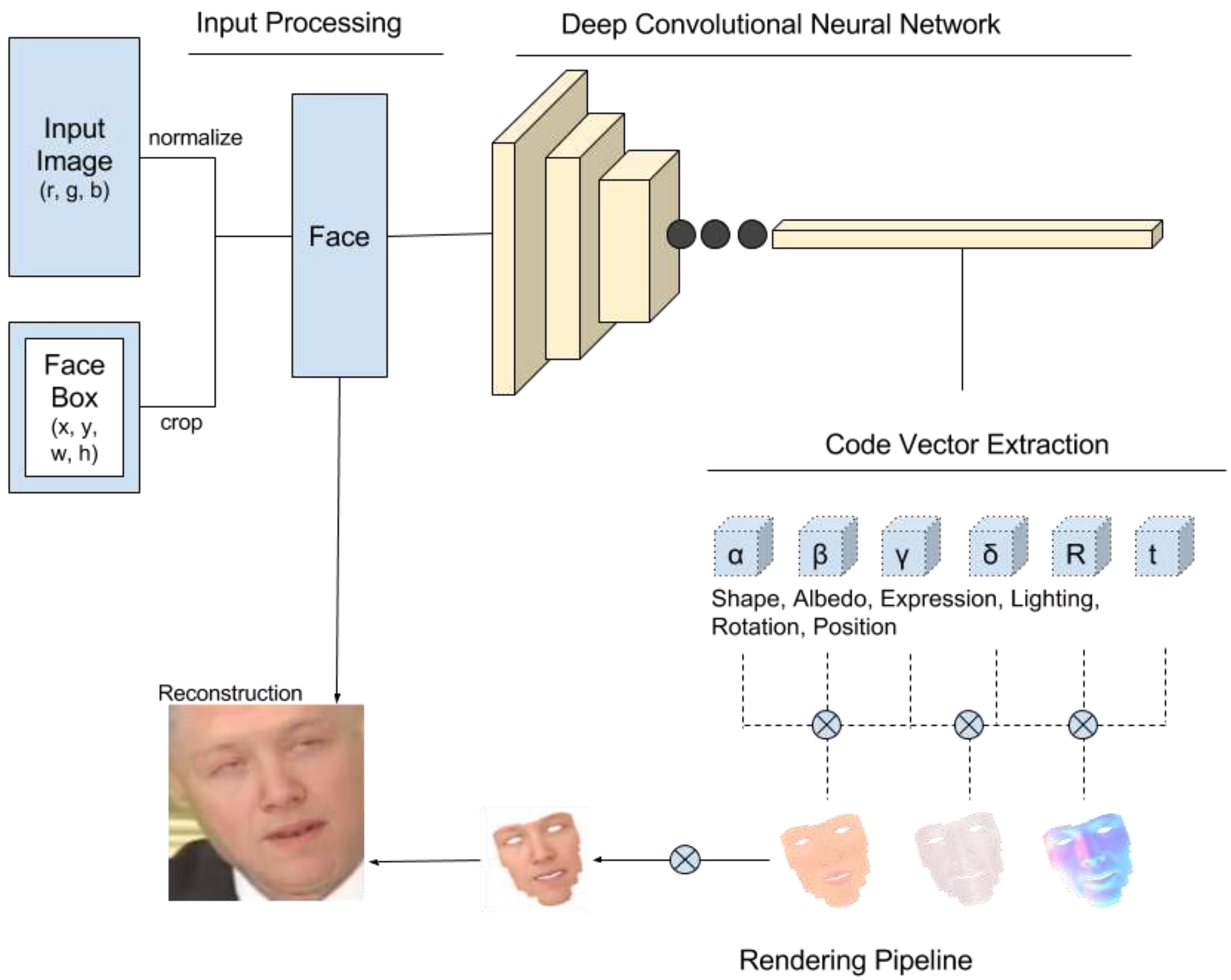

Ethnicity


Age

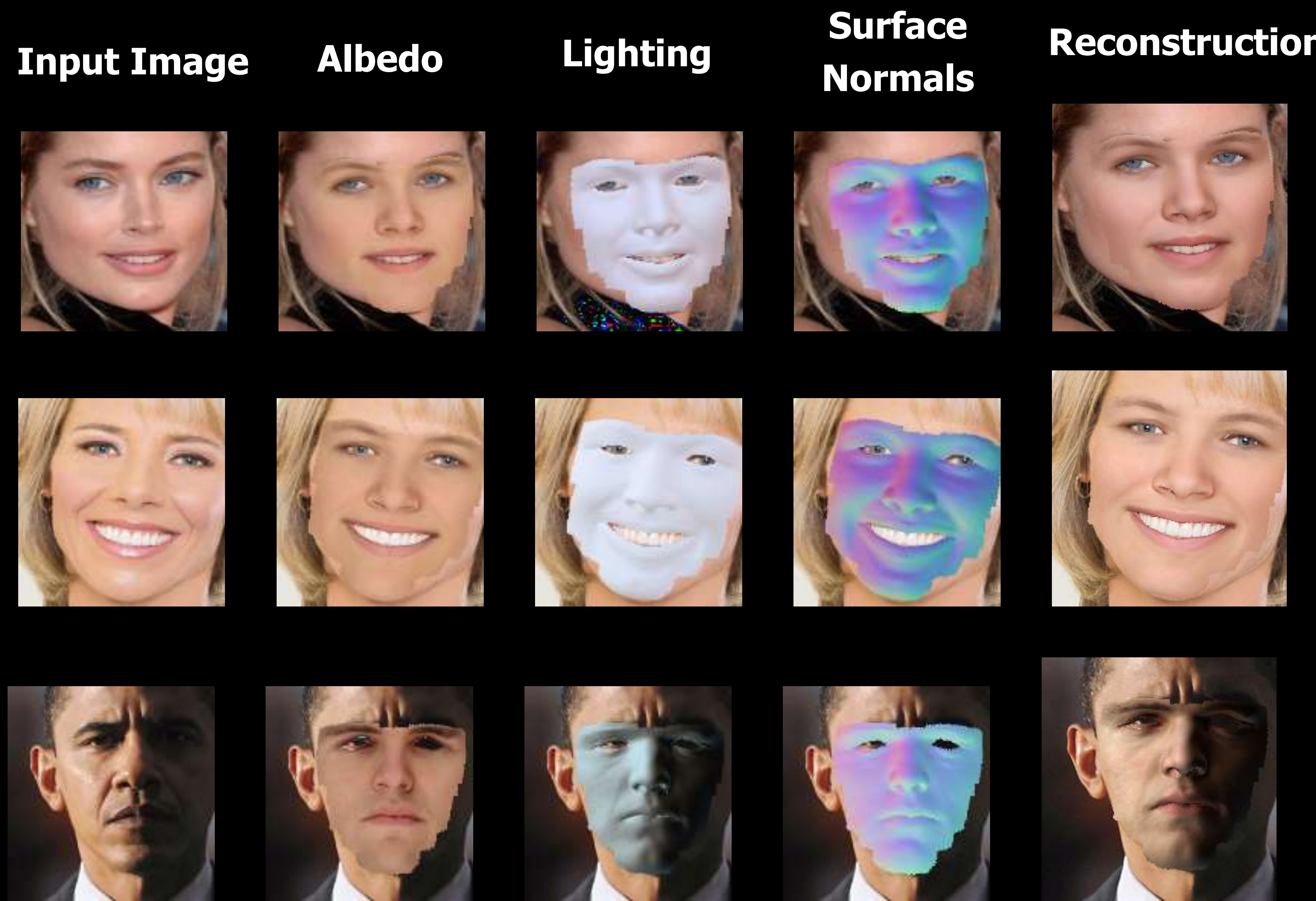

Yaw
Pitch
Roll
Interest


Facial Expressions

CNN Model: Multi-Task Supervised



CNN Model: Results



Conclusion:
Deep Learning in Computer Vision

- Object classification, detection and segmentation
- Optical flow
- Color constancy
- Intrinsic image decomposition
- 3D and slam
- Human behavior analysis
- Etc…

Computer Vision 1 (total #slides 89| Lecture 6)

Summary

1. **Object Recognition (ConvNets)**
2. **Object Detection**
3. **Stereo Vision (Next Week)**