Language Technologies Institute

Carnegie Mellon University

Multimodal Machine Learning

Lecture 3.1: Visual Representations and CNNs

Louis-Philippe Morency

** Original course co-developed with Tadas Baltrusaitis.*
Spring 2021 edition taught by Yonatan Bisk

Administrative Stuff

Pre-proposals – Due tomorrow 9/15

- Dataset and research problem
- Input modalities and multimodal challenges
- Initial research ideas
- Teammates and resources

Submit via Gradescope before 8PM ET

If you are still looking for teammates, you should still submit a pre-proposals. We will help you!

Upcoming Deadlines

Week 3 reading assignment was posted

1. **Wednesday 8pm:** Select your paper
2. **Friday 8pm:** Post your summary
3. **Monday 8pm:** End of the reading assignment

Preproposal deadline: **Wednesday 8pm**

Language Technologies Institute

Carnegie Mellon University

Multimodal Machine Learning

Lecture 3.1: Visual Representations and CNNs

Louis-Philippe Morency

** Original course co-developed with Tadas Baltrusaitis.*
Spring 2021 edition taught by Yonatan Bisk

Lecture Objectives

- Image representations
  - Object descriptors
- Convolutional Neural networks
  - Convolution kernels
  - Convolution neural layers
  - Pooling layers
- Convolutional architectures
  - VGGNet and residual networks
  - Visualizing CNNs
  - Region-based CNNs
  - Sequential Modeling with convolutional networks
- Appendix: Tools for visual behavior analysis

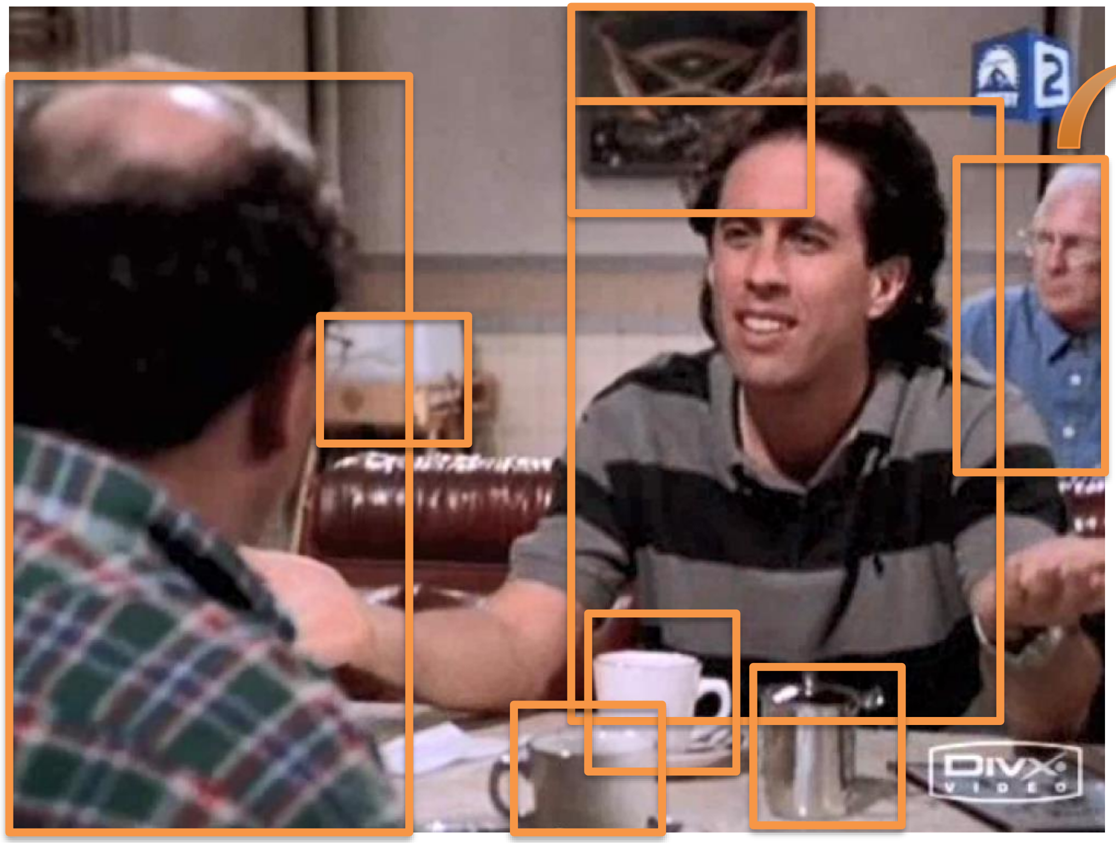
Image Representations

How Would You Describe This Image?

88
88
85
38
20
22
24
21
23
82
80
79
35
25
26
28
22
22
84
78
80

⋮

Object-Based Visual Representation



"person" label

+

Appearance descriptor

- ❑ Age
- ❑ Expression
- ❑ Clothes
- ❑ …

Feature vector

88
88
85
38
20
22
24
21
23
82
80
79
35
25
26
28
22
22
84
78
80

Object Descriptors

How to represent and detect an object?

Many approaches over the years…

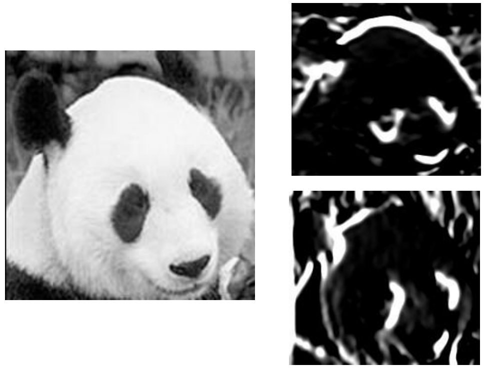
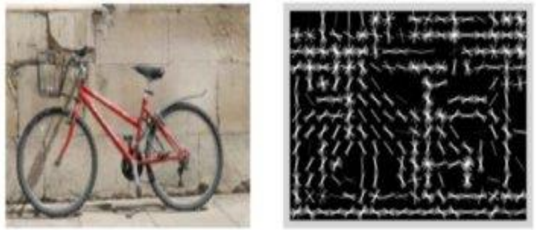


Image gradient

Edge detection



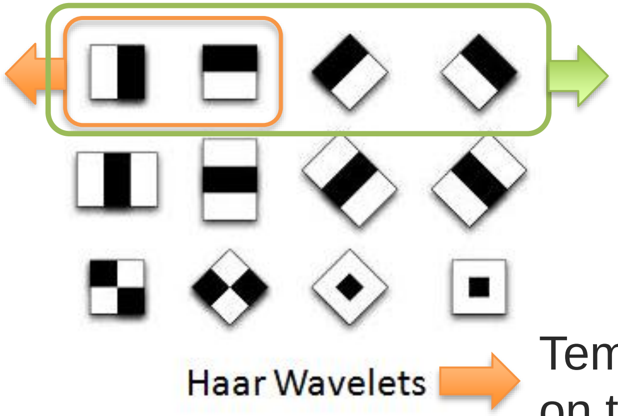
Histograms of Oriented Gradients

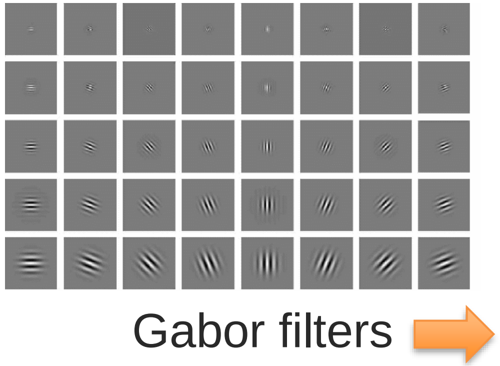
Optical Flow

Object Descriptors

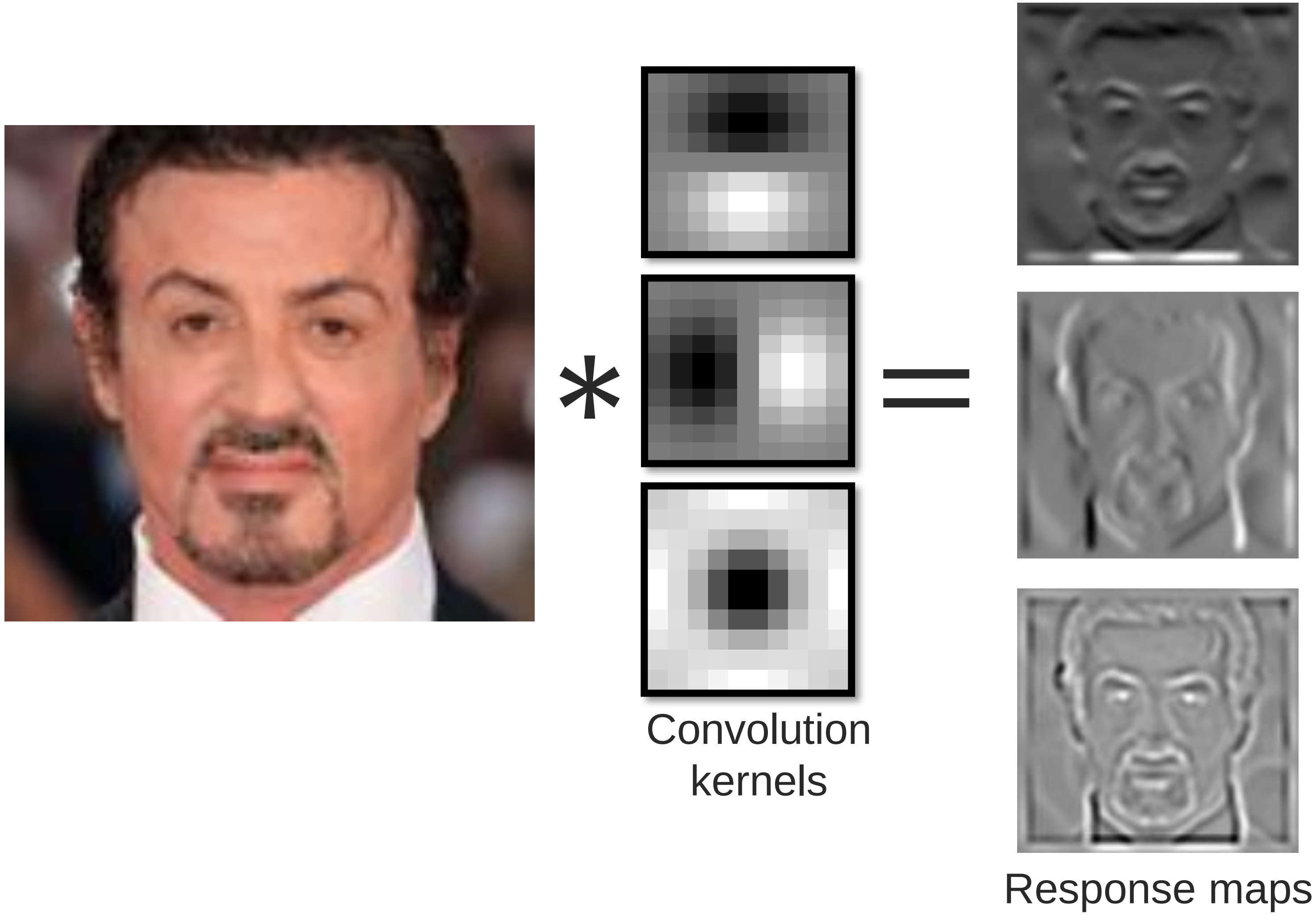
How to represent and detect an object?

Many approaches over the years…





Convolution Kernels

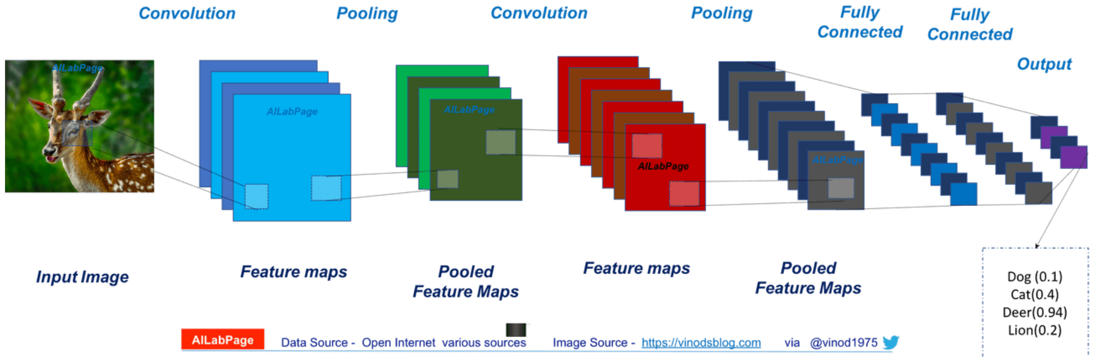


Object Descriptors

How to represent and detect an object?

Many approaches over the years…

Convolutional Neural Network (CNN)



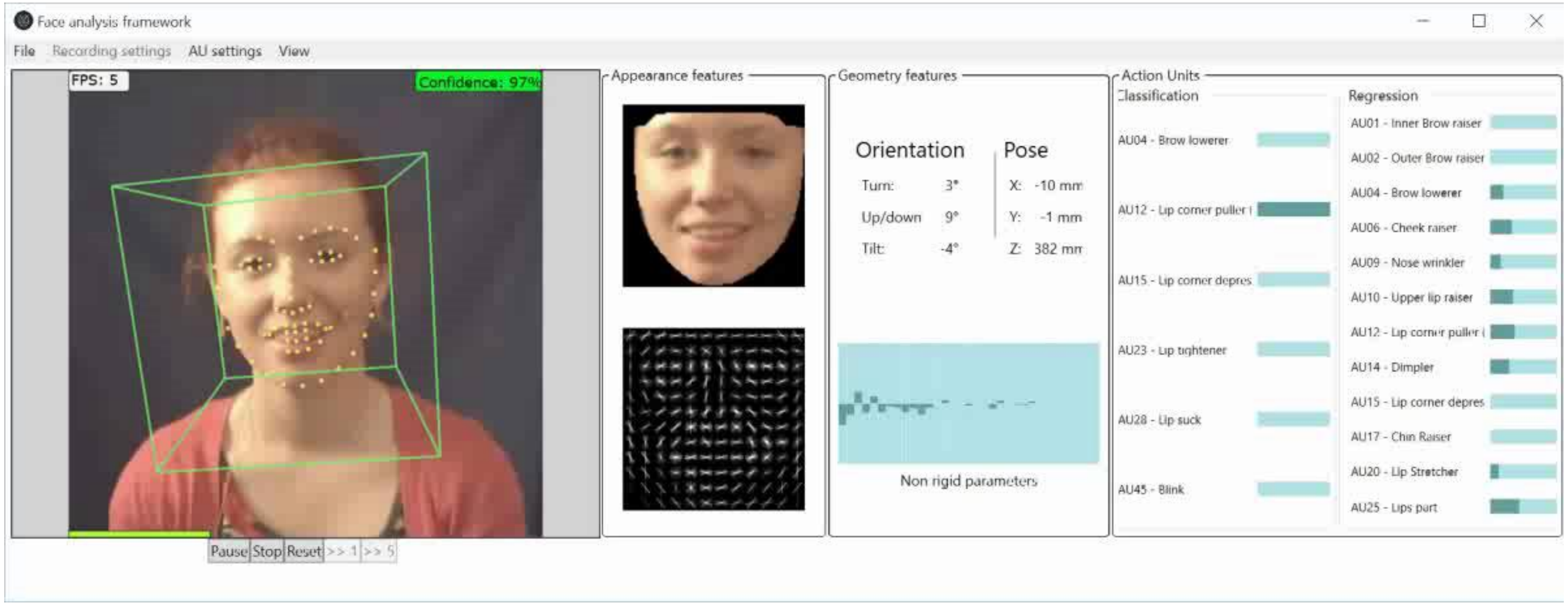
More details about CNNs is coming…

And images are more than a list of objects!

One representation, lots of tasks

https://github.com/facebookresearch/detectron2

Facial expression analysis



[**OpenFace: an open source facial behavior analysis toolkit**, T. Baltrušaitis et al., 2016]

Articulated Body Tracking: OpenPose

https://github.com/CMU-Perceptual-Computing-Lab/openpose

See appendix for list of available tools for automatic visual behavior analysis

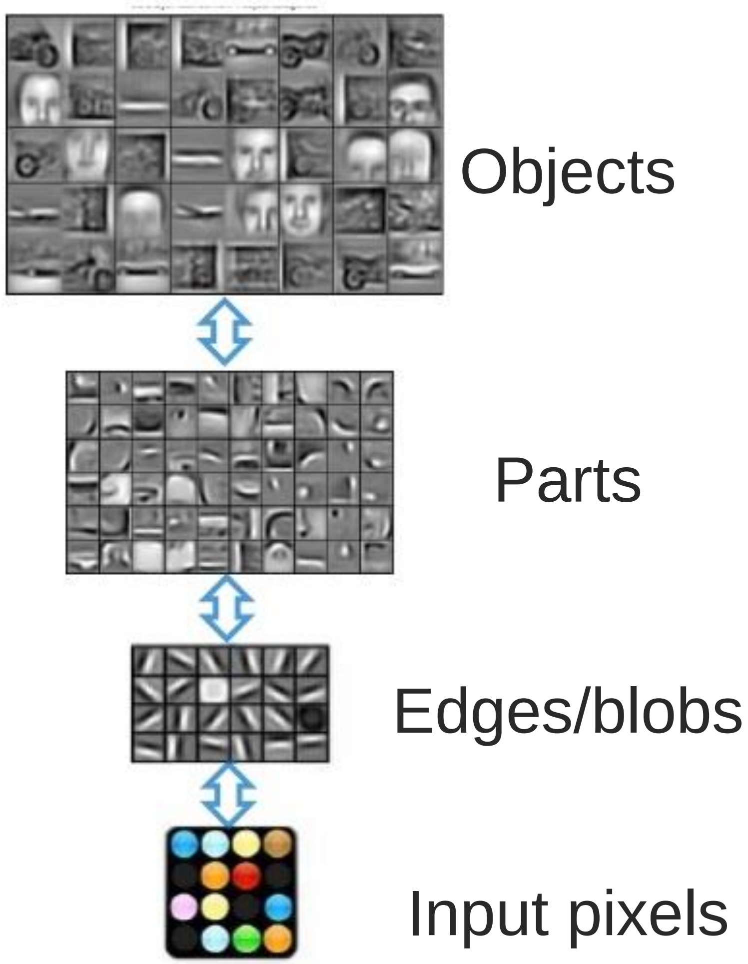
Convolutional Neural Networks

Why using Convolutional Neural Networks?

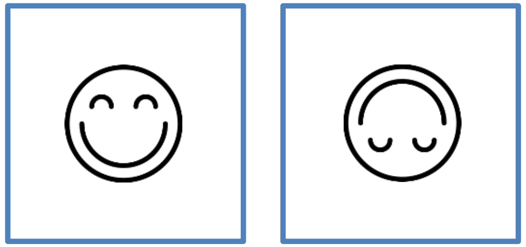
Goal: building more abstract, hierarchical visual representations

Key advantages:

1) Inspired from visual cortex
2) Encourages visual abstraction
3) Exploits *translation invariance*
4) Kernels/templates are learned
5) Fewer parameters than MLP



Translation Invariance



2 Data Points – Which one is up?

- ➢ MLP can easily learn this task (possibly with only 1 neuron!)

What happens if the face is slightly translated?

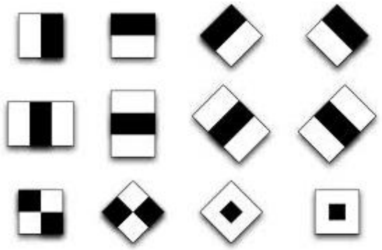
- ➢ The model should still be able to classify it

Conventional MLP models are not translation invariant!

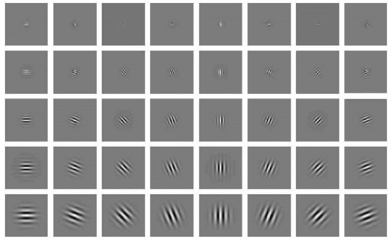
- ➢ But CNNs are kernel-based, which helps with translation invariance and reduce number of parameters

Learned vs Predefined Kernels

Predefined kernels



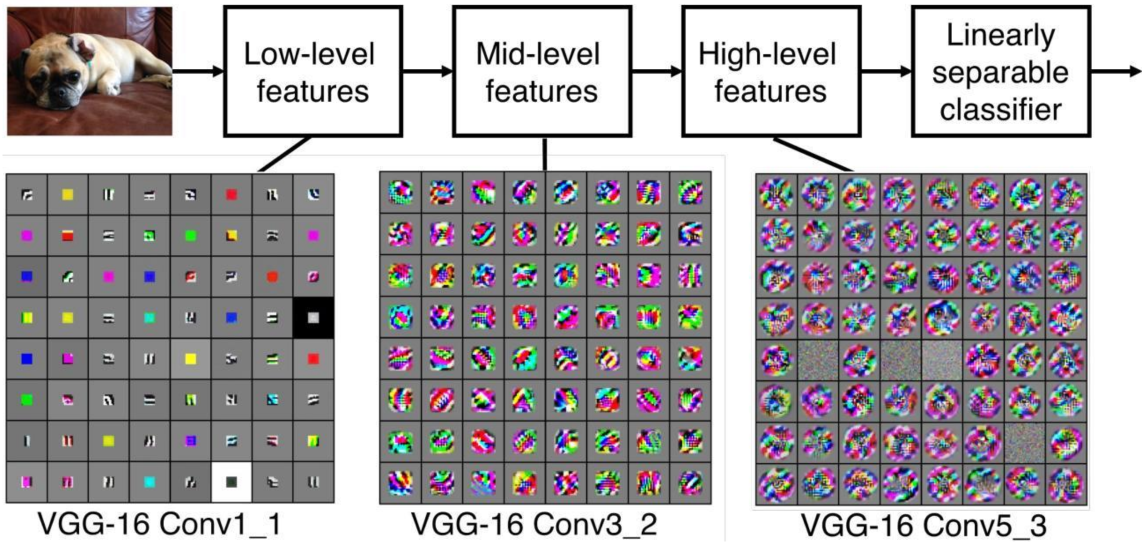
Haar Wavelets



Gabor filters

Learned kernels

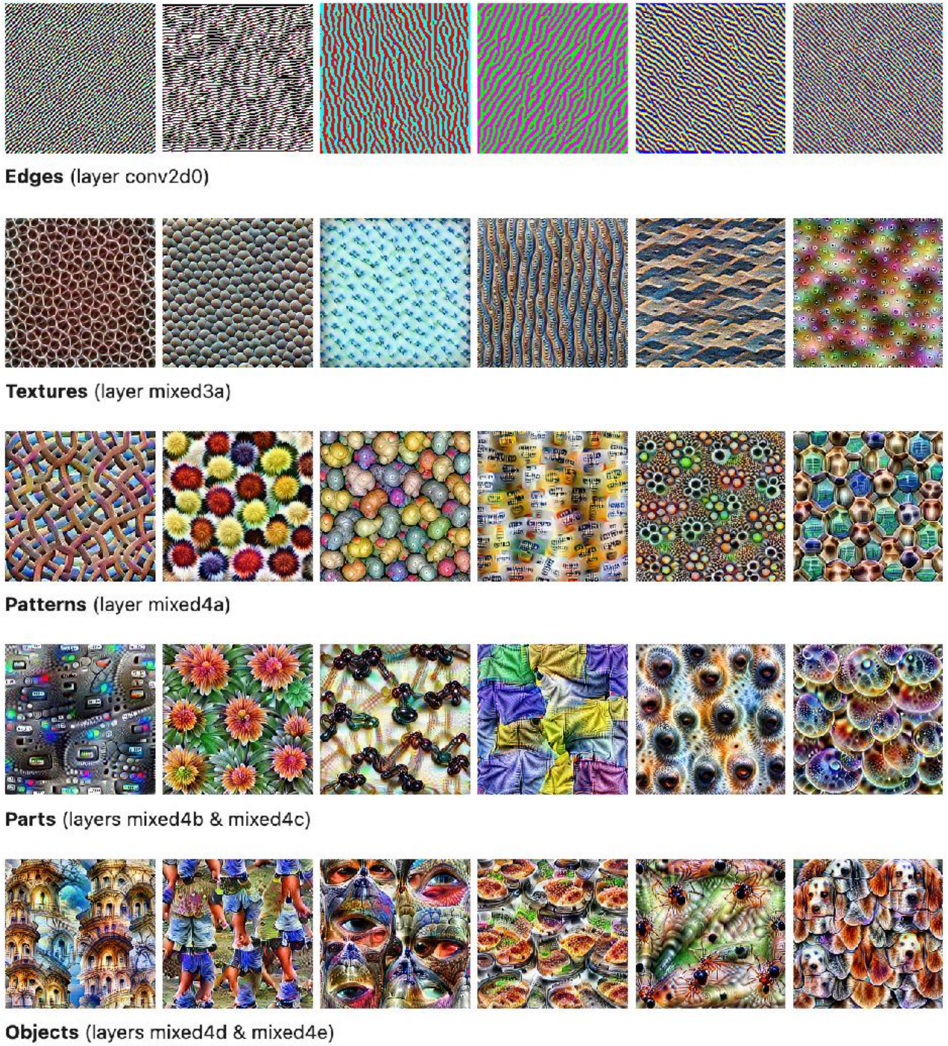
Convolutional Neural Network (CNN)



With CNNs, the kernel values are learned as model parameters

Learned Filters

https://distill.pub/2017/feature-visualization/



Convolution

Convolution: Mathematical Definition

A basic mathematical operation (that given two functions returns a function)

$$(f * g)[n] \stackrel{\text{def}}{=} \sum_{m=-\infty}^{\infty} f[m]g[n-m]$$

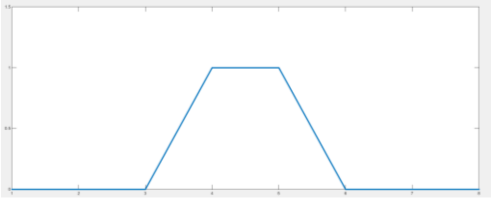
Two versions: continuous and discrete

(we will focus on the latter)

Convolution in 1D – Example

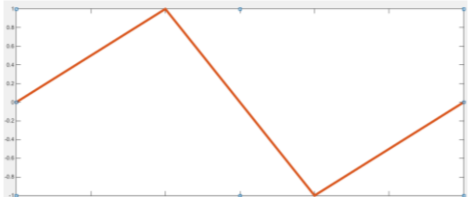
Input:

$$f = [\dots, 0,1,1,1,0,0, \dots]$$



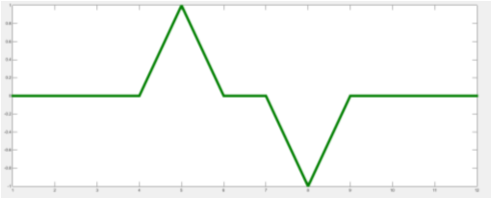
Kernel:

$$g = [\dots, 0,1,-1,0 \dots]$$



Convolution:

$$f * g = [\dots, 0,1,0,0,-1,0,0, \dots]$$



$$(f * g)[n] \stackrel{\text{def}}{=} \sum_{m=-\infty}^{\infty} f[m]g[n-m]$$

Convolution in practice

In CNN we only consider functions with limited domain (not from $-\infty$ to ∞)

CNN considers fully defined (valid) version:

- We have a signal of length N
- Kernel of length K
- Output will be length $\mathrm{N}-\mathrm{K}+1$

Example: $f=[1,2,1]$, $g=[1,-1]$, $f*g=[1,-1]$

Convolution in practice

If we want output to be different sizes, we can add padding to the signal:

- Just add 0s at the beginning and end

$$f = [0,0,1,2,1,0,0],\ g = [1,-1],\ f * g = [0,1,1,-1,-1,0]$$

We can perform *strided (aka, dilated)* convolution: the filter jumps over pixels or samples

- Example with stride 2:

$$f = [0,0,1,2,1,0,0],\ g = [1,-1],\ f * g = [0,1,-1,0]$$

When would it be a good idea?

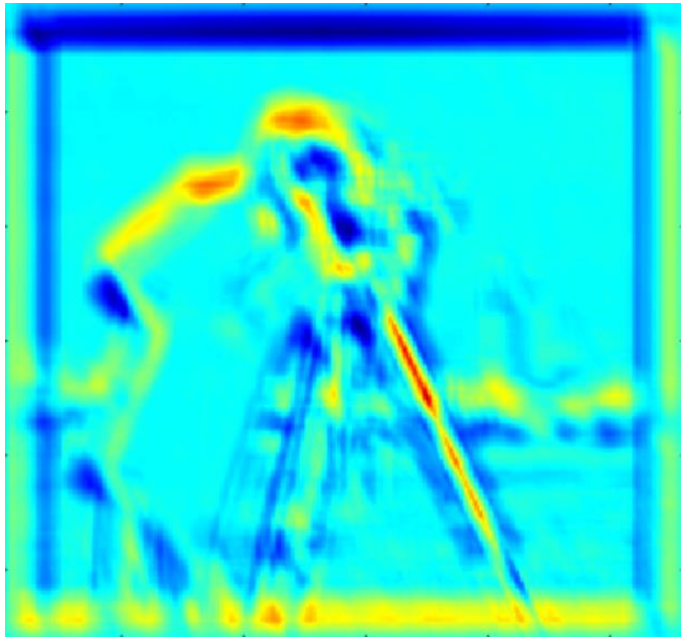
Convolution in 2D – Example

Input image

$*$

Convolution kernel

$=$

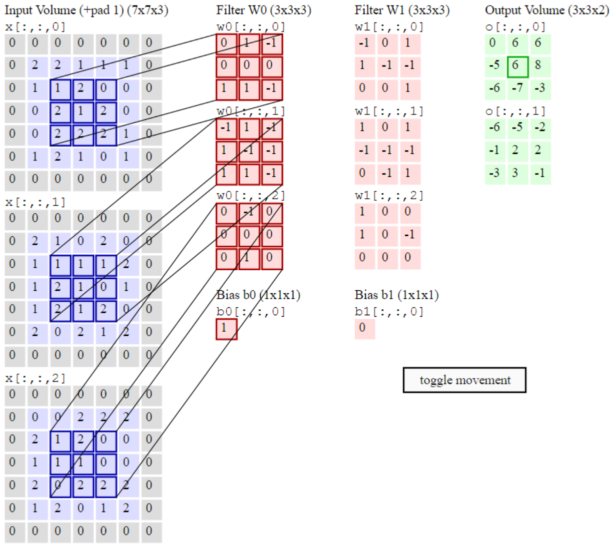


Response map

Sample CNN convolution

Great animated visualization of 2D convolution:

http://cs231n.github.io/convolutional-networks/



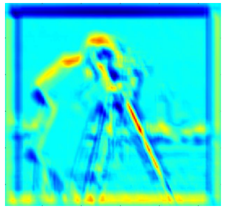
Convolutional Neural Layer

Convolution as a Fully-Connected Network

Input

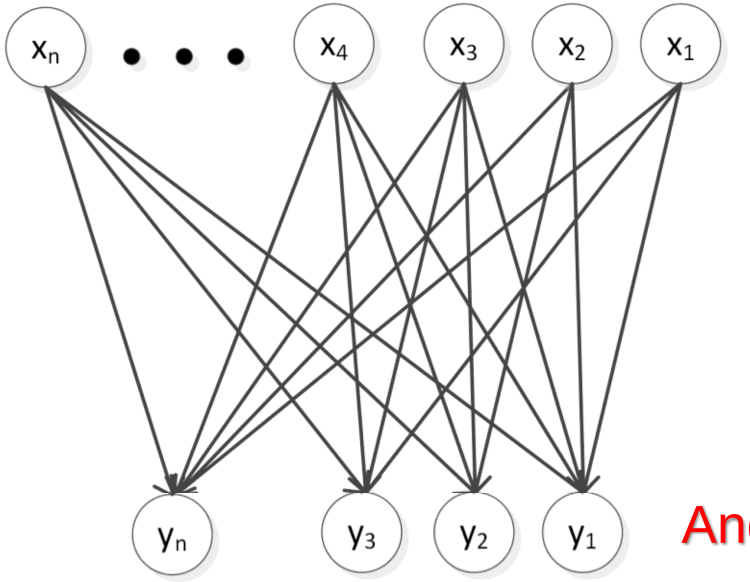
(image)

Output



(response map)

Input: all pixels



Output: kernel responses

Not efficient!

200×200 image requires $40{,}000 \times n$ parameters

(where n is size of kernel)

And it may learn different kernels for different pixel positions

➡ Not translation invariant

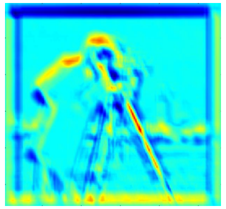
Convolutional Neural Layer

Input

(image)

Weighted sum
Wx

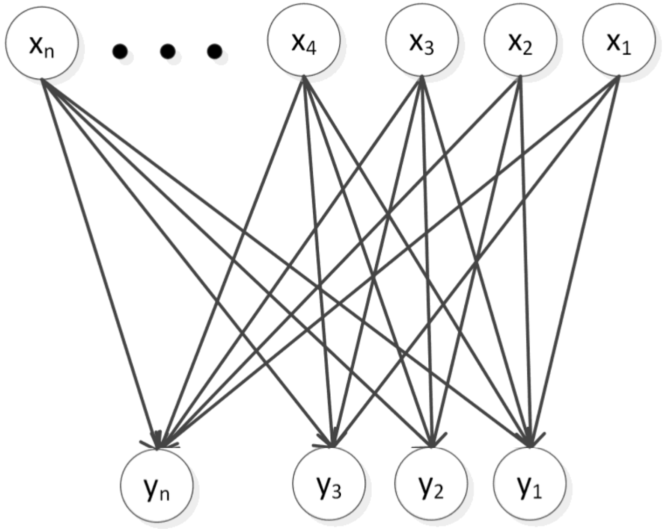
Output



(response map)

$$y = Wx$$

Input: all pixels



Output: kernel responses

Example with 1D kernel:

w_1	w_2	w_3

Convolutional Neural Layer

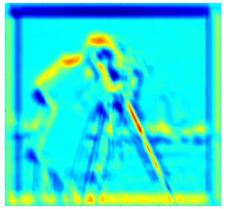
Modification 1: Remove redundant links making the matrix W sparse

Input

(image)

Weighted sum
Wx

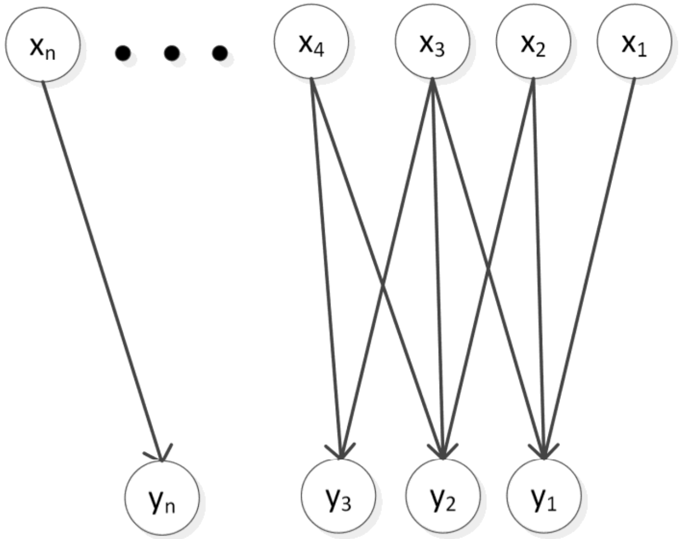
Output



(response map)

$y = Wx$

Input: all pixels



Output: kernel responses

Example with 1D kernel:

w_1	w_2	w_3

Convolutional Neural Layer

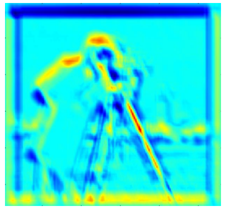
Modification 2: share the weights in matrix W
not to do redundant computation

Input

(image)

Weighted sum
Wx

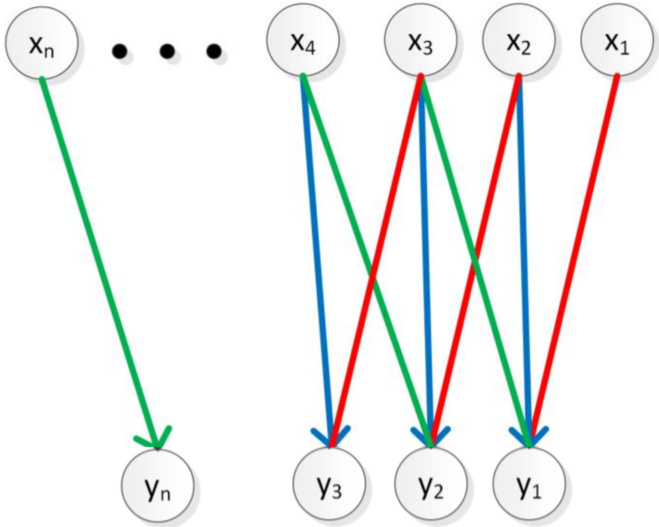
Output



(response map)

$y = Wx$

Input: all pixels



Output: kernel responses

Example with
1D kernel:

w_1	w_2	w_3

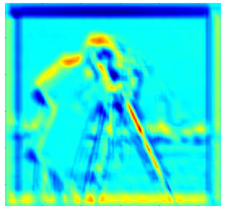
Convolutional Neural Layer

Modification 2: share the weights in matrix W
not to do redundant computation

Input

(image)

Output



(response map)

$$y = Wx$$

$$W = \begin{pmatrix} w_1 & w_2 & w_3 & & 0 & 0 & 0 \\ 0 & w_1 & w_2 & \cdots & 0 & 0 & 0 \\ 0 & 0 & w_1 & & 0 & 0 & 0 \\ & \vdots & & \ddots & & \vdots & \\ 0 & 0 & 0 & & w_3 & 0 & 0 \\ 0 & 0 & 0 & \cdots & w_2 & w_3 & 0 \\ 0 & 0 & 0 & & w_1 & w_2 & w_3 \end{pmatrix}$$

Example with 1D kernel:

w_1	w_2	w_3

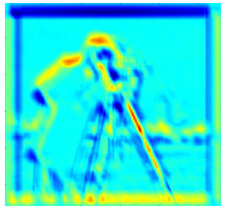
Can be implemented efficiently on GPUs

Convolutional Neural Layer

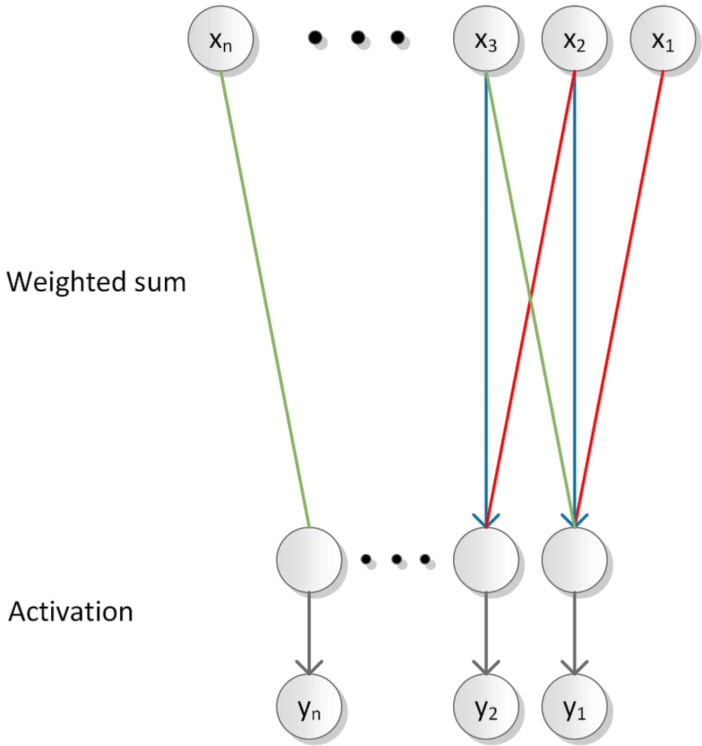
Input

(image)

Output



(response map)



Example with 1D kernel:

w_1	w_2	w_3

$y = \sigma(Wx + b)$ with the activation function, and bias terms

Convolutional Neural Layer

Can expand this to 2D (or even 3D!)

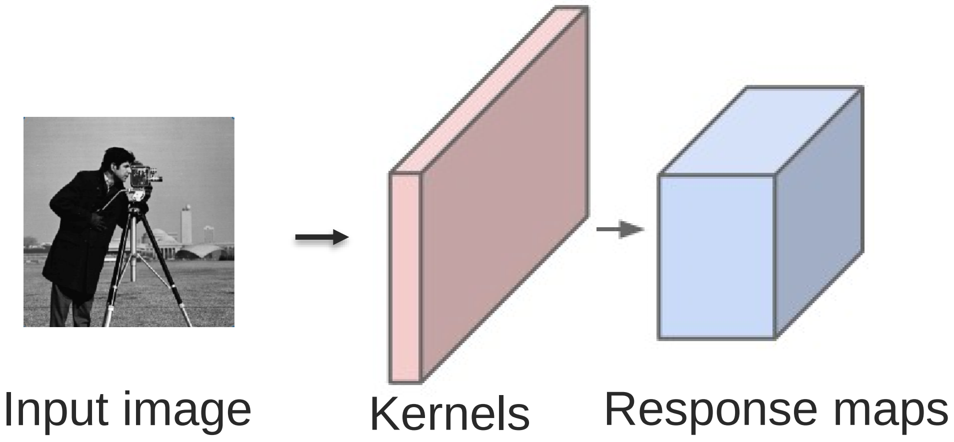
- Just need to make sure to link the right pixel with the right weight

Can expand to multi-channel 2D

- e.g., for RGB images

Can expand to multiple kernels/filters

- Output is not a single image anymore, but a tensor (a 3D matrix)



Convolutional Neural Network

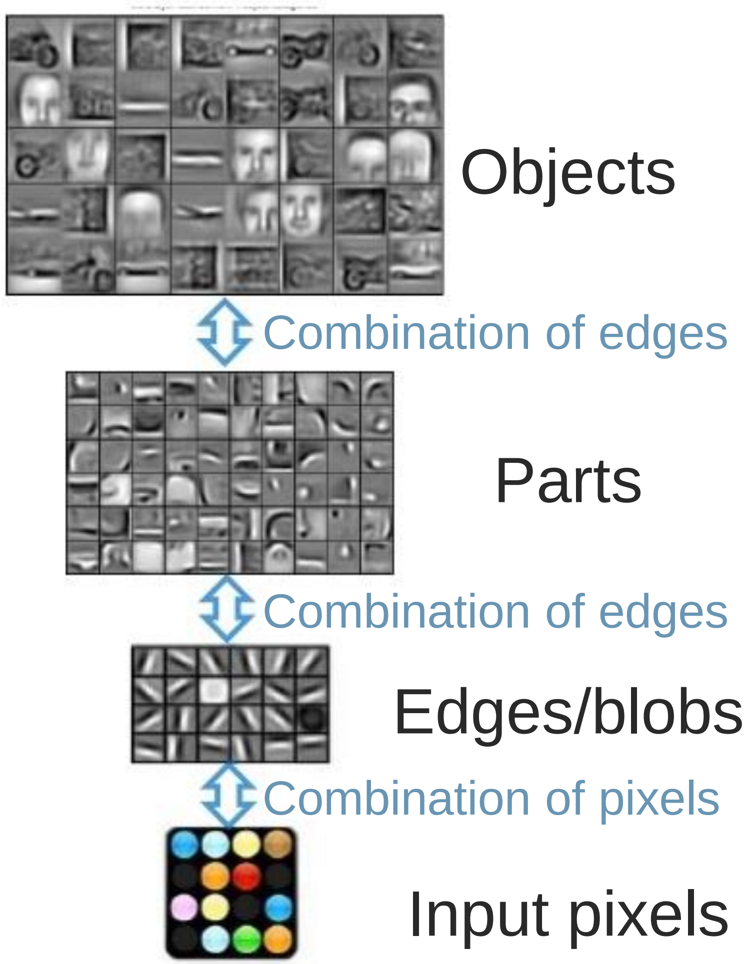
Convolutional Neural Network

Multiple convolutional layers

- Allows the network to learn combinations of sub-parts, to increase complexity

but how to encourage abstraction and summarization?

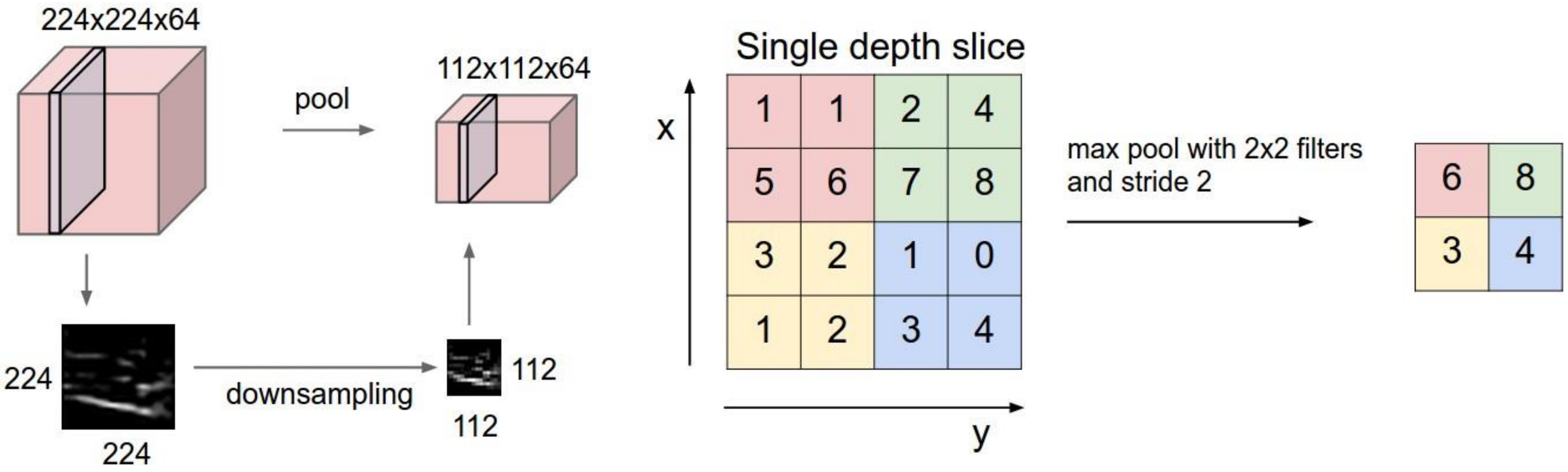
Answer: Pooling layers



Pooling Layer

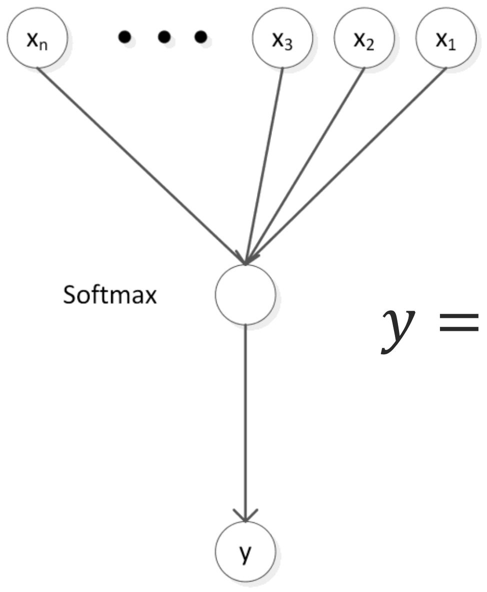
Response map subsampling:

Allows summarization of the responses



Pooling Layer Gradient

1. Record during forward pass which pixel was picked and use the same in backward pass
2. Pick the maximum value from input using a smooth and differentiable approximation



$$y = \frac{\sum_{i=1}^{n} x_i e^{\alpha x_i}}{\sum_{i=1}^{n} e^{\alpha x_i}}$$

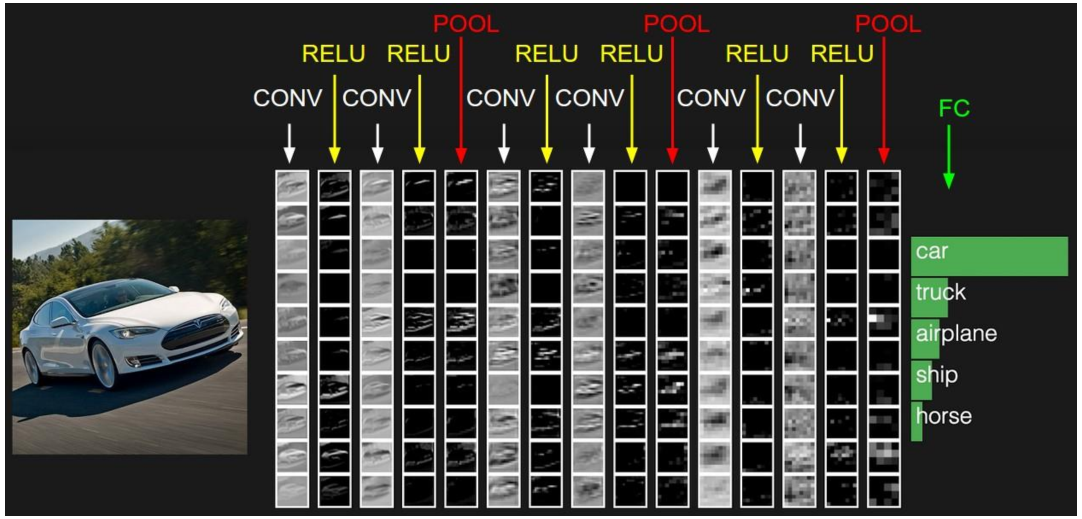
Example of CNN Architectures

Common architectures

Start with a convolutional layer follow by non-linear activation and pooling

Repeat this several times

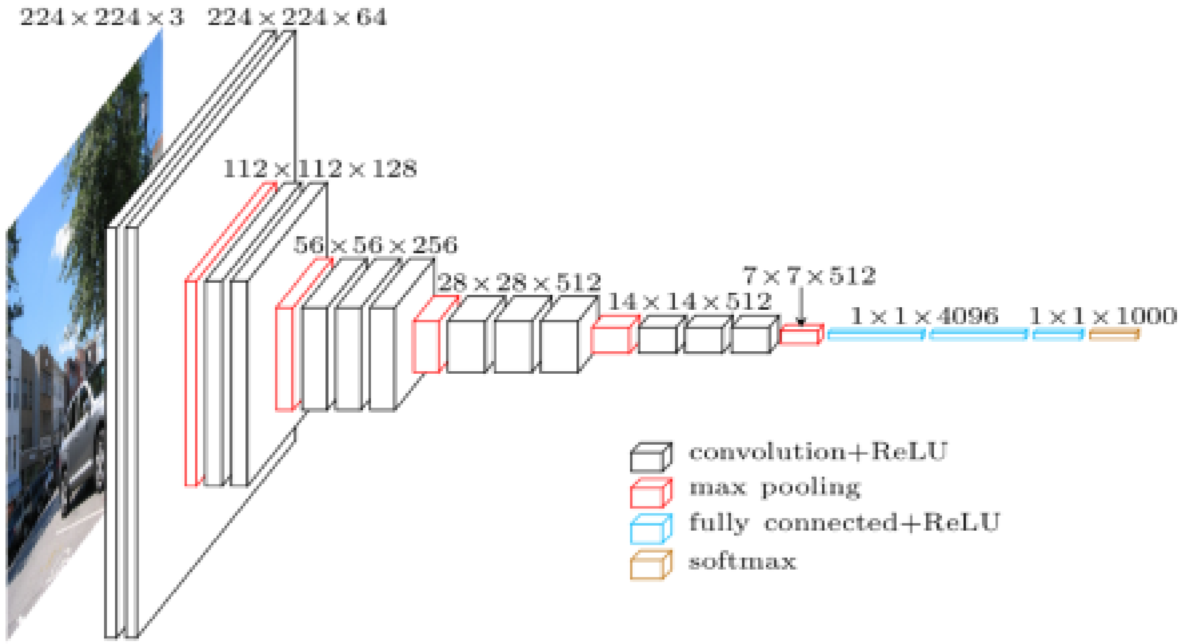
Ends with a fully connected (MLP) layer



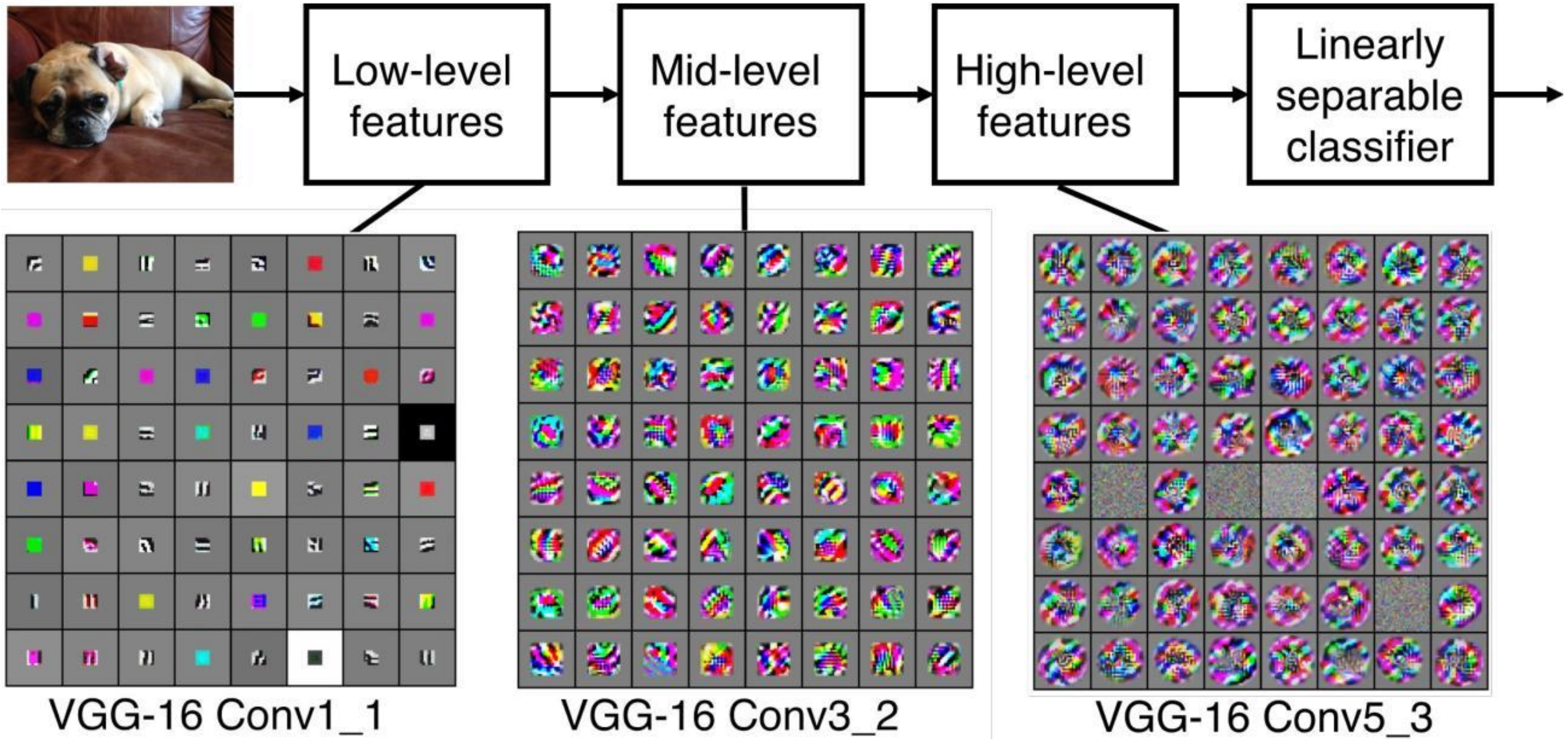
VGGNet model

Used for object classification task

- 1000-way classification task
- 138 million parameters

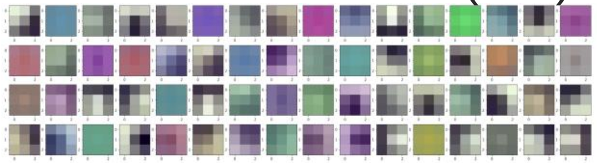


VGGNet Convolution Kernels

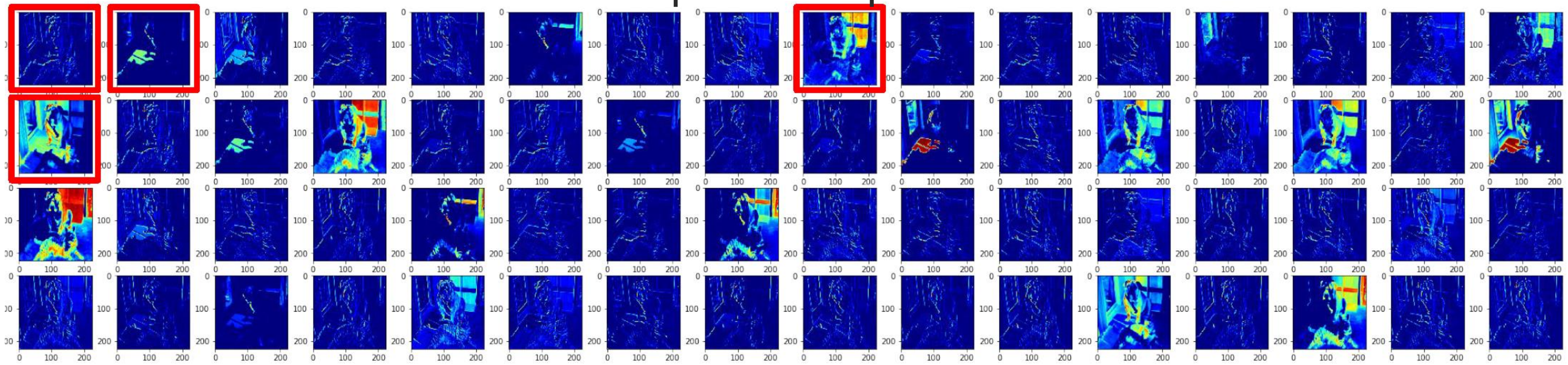


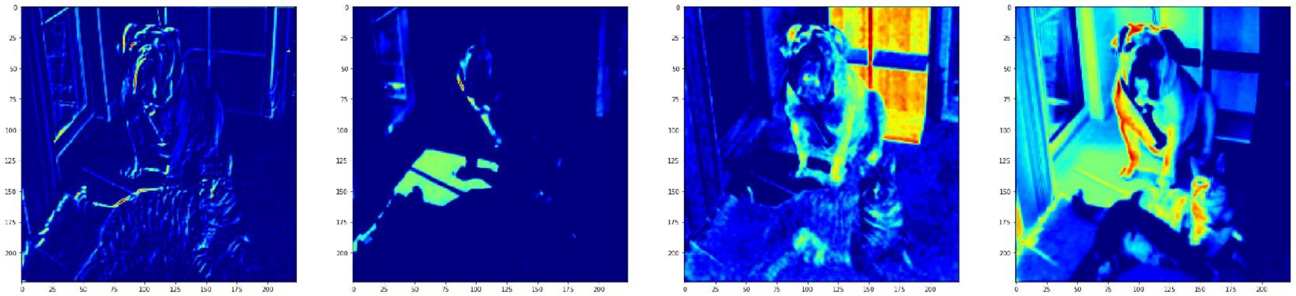
VGGNet Response Maps (aka Activation Maps)

Convolution kernels (3x3)



Response Maps





Other architectures

LeNet – an early 5 layer architecture for handwritten digit recognition

DeepFace – Facebook's face recognition CNN

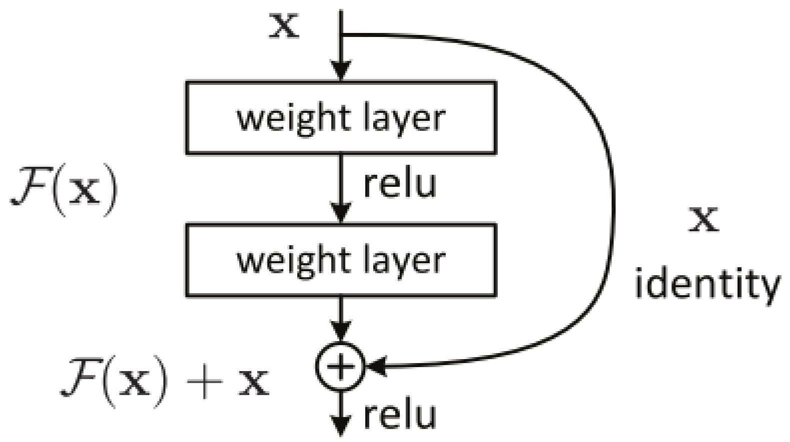
VGGFace – For face recognition (from VGG folks)

AlexNet – Object Recognition

Already trained models for object recognition can be found online

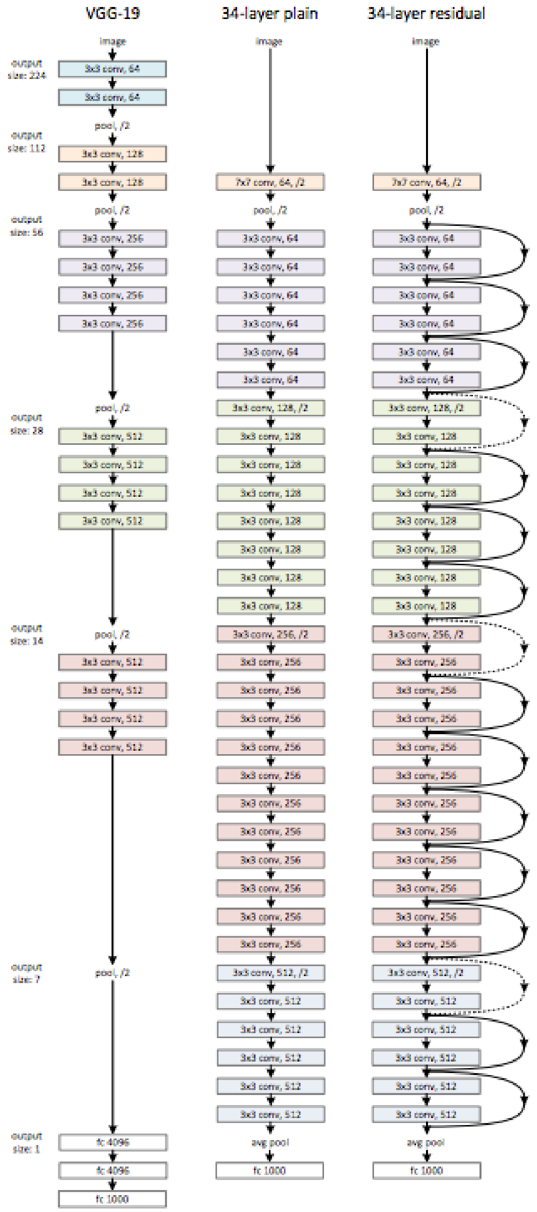
Residual Networks

Adding residual connections



ResNet (He et al., 2015)

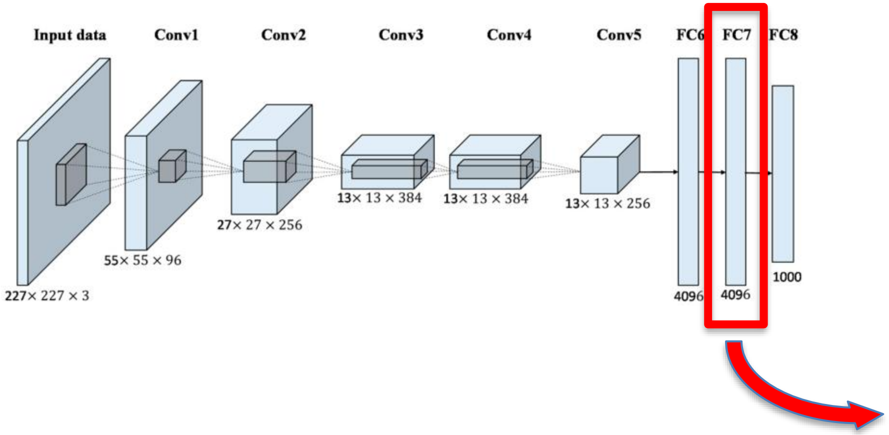
- Up to 152 layers!

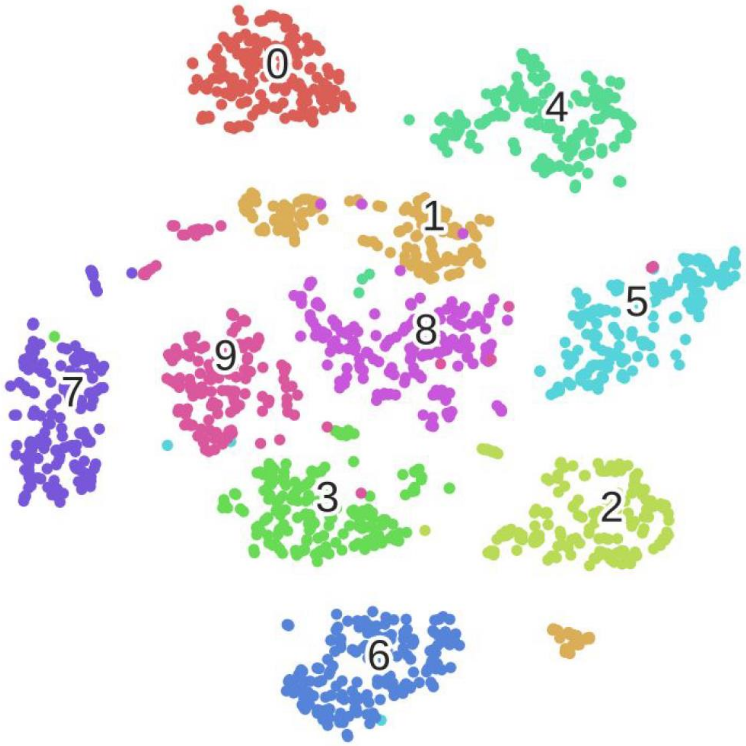


Visualizing CNNs

Visualizing the Last CNN Layer: t-sne

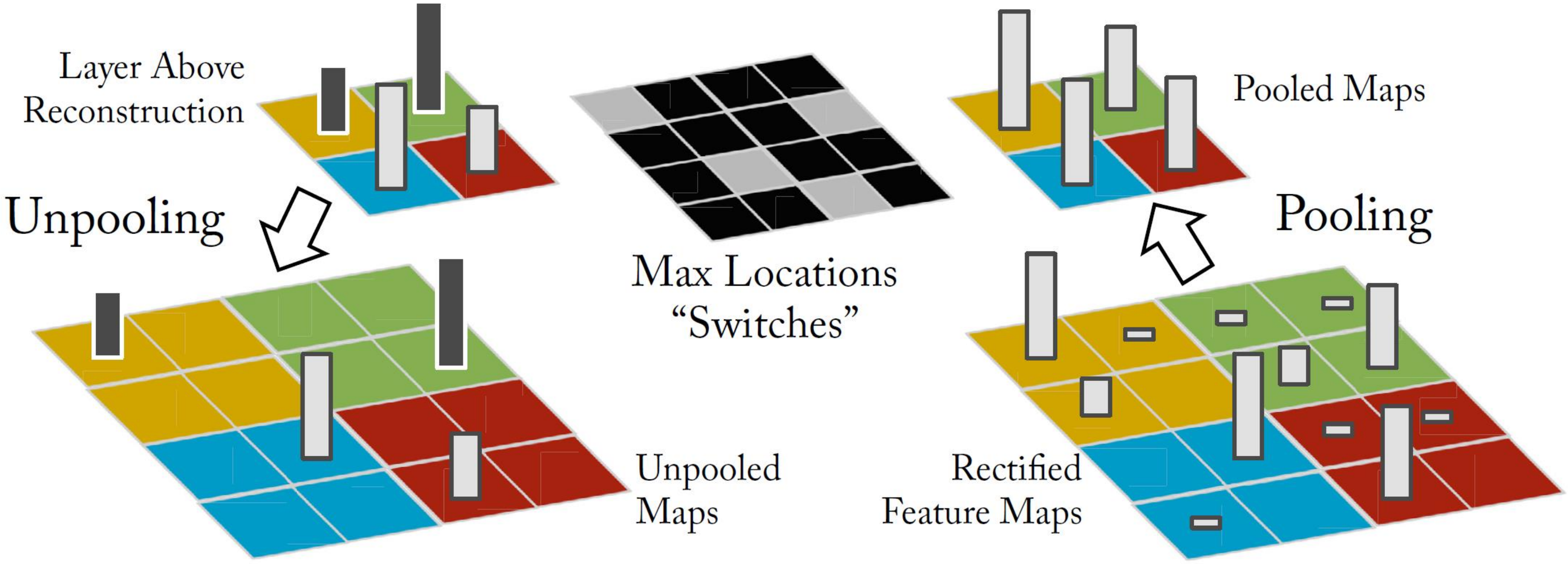
Alex Net



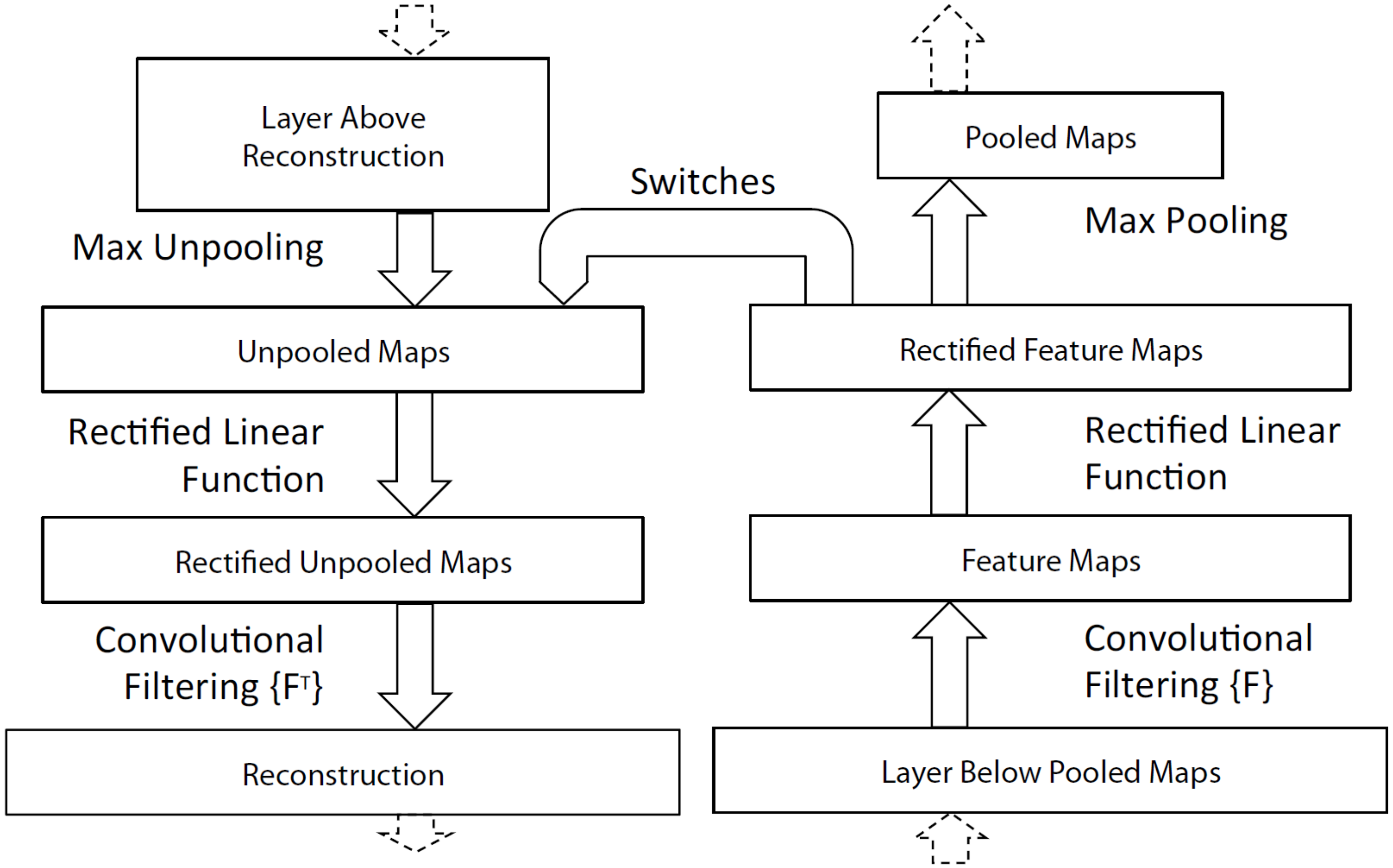


Embed high dimensional data points (i.e. feature codes) so that pairwise distances are conserved in local neighborhoods.

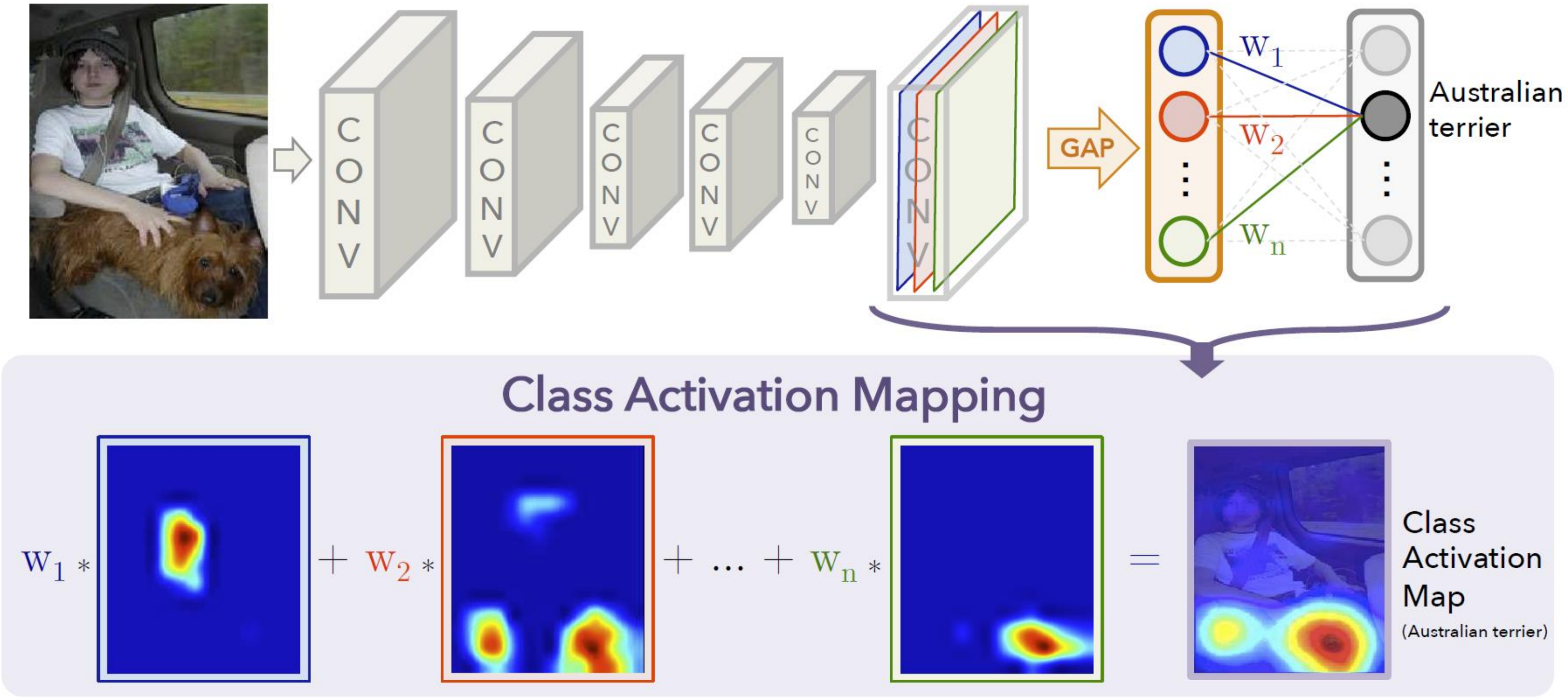
Deconvolution



Deconvolution

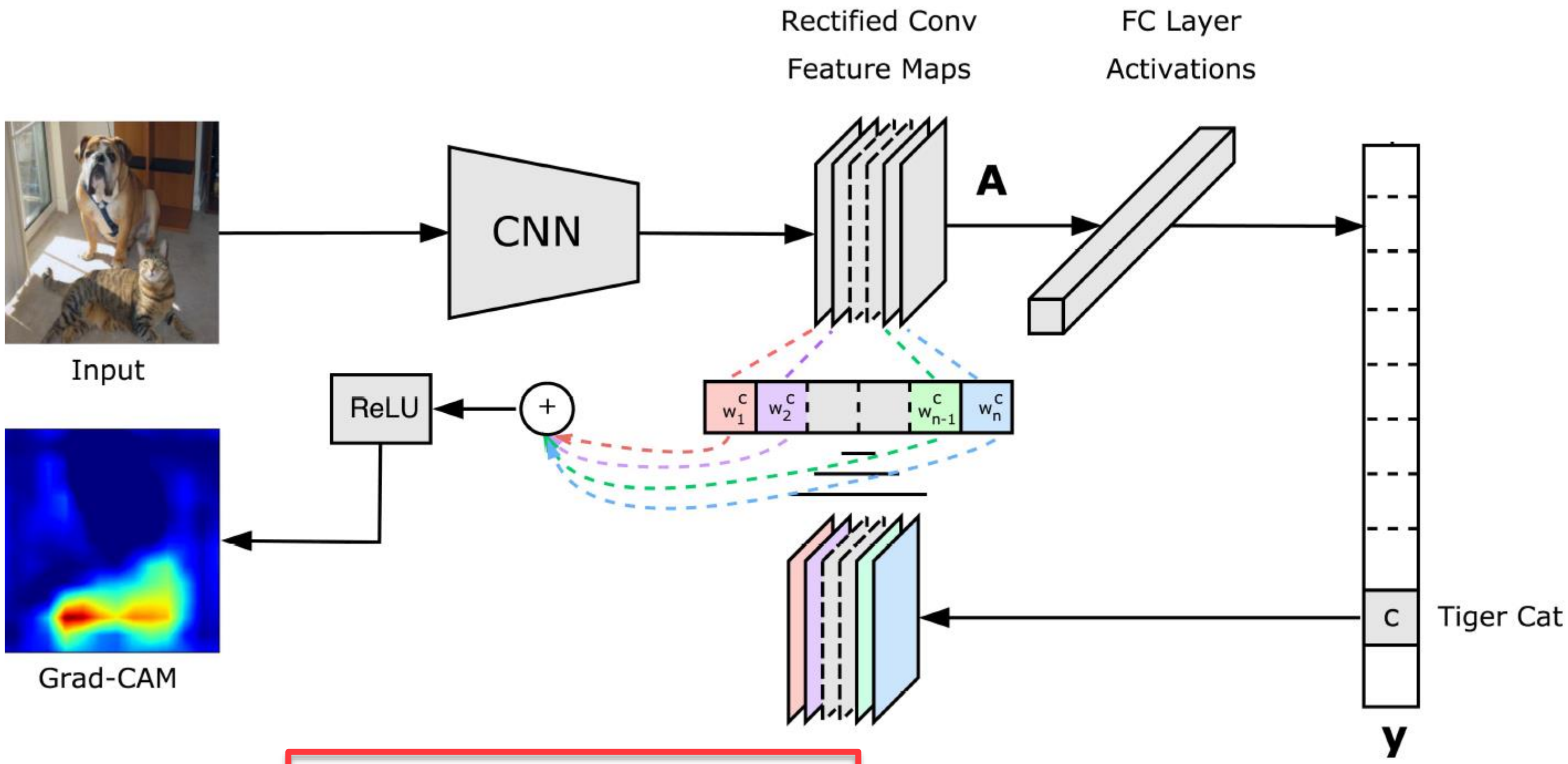


CAM: Class Activation Mapping [CVPR 2016]



$$L^c_{\text{CAM}} = \underbrace{\sum_k w^c_k A^k}_{\text{linear combination}}$$

Grad-CAM [ICCV 2017]



$$\alpha_k^c = \overbrace{\frac{1}{Z}\sum_i \sum_j}^{\text{global average pooling}} \underbrace{\frac{\partial y^c}{\partial A_{ij}^k}}_{\text{gradients via backprop}}$$

$$L^c_{\text{Grad-CAM}} = ReLU \underbrace{\left(\sum_k \alpha_k^c A^k\right)}_{\text{linear combination}}$$

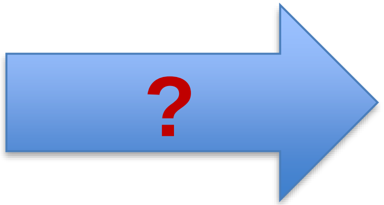
Region-based CNNs

Object recognition

Input Image

Object Detection (and Segmentation)

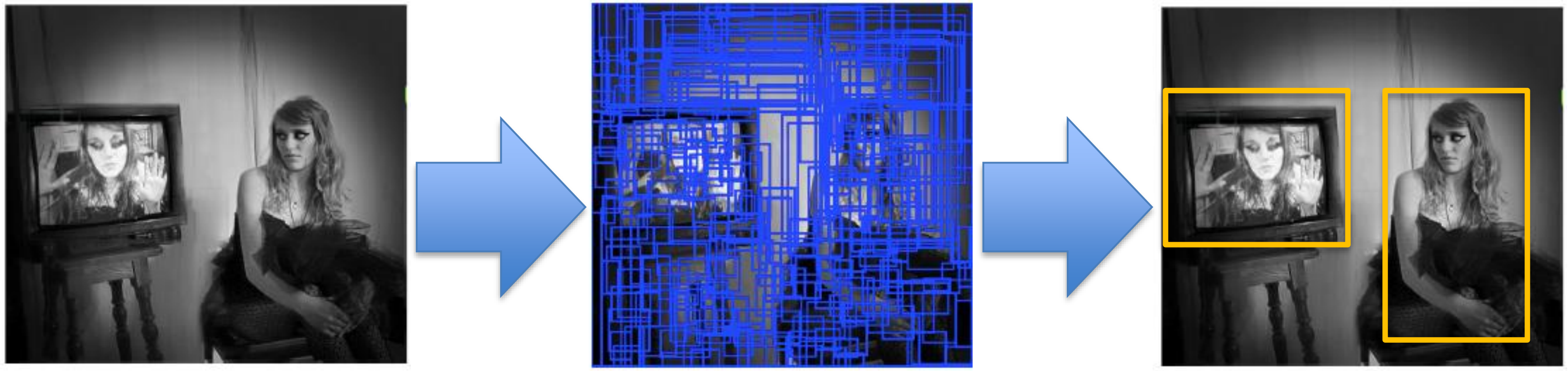
Input image



Detected Objects

One option: Sliding window

Object Detection (and Segmentation)



Input image Region Proposals Detected Objects

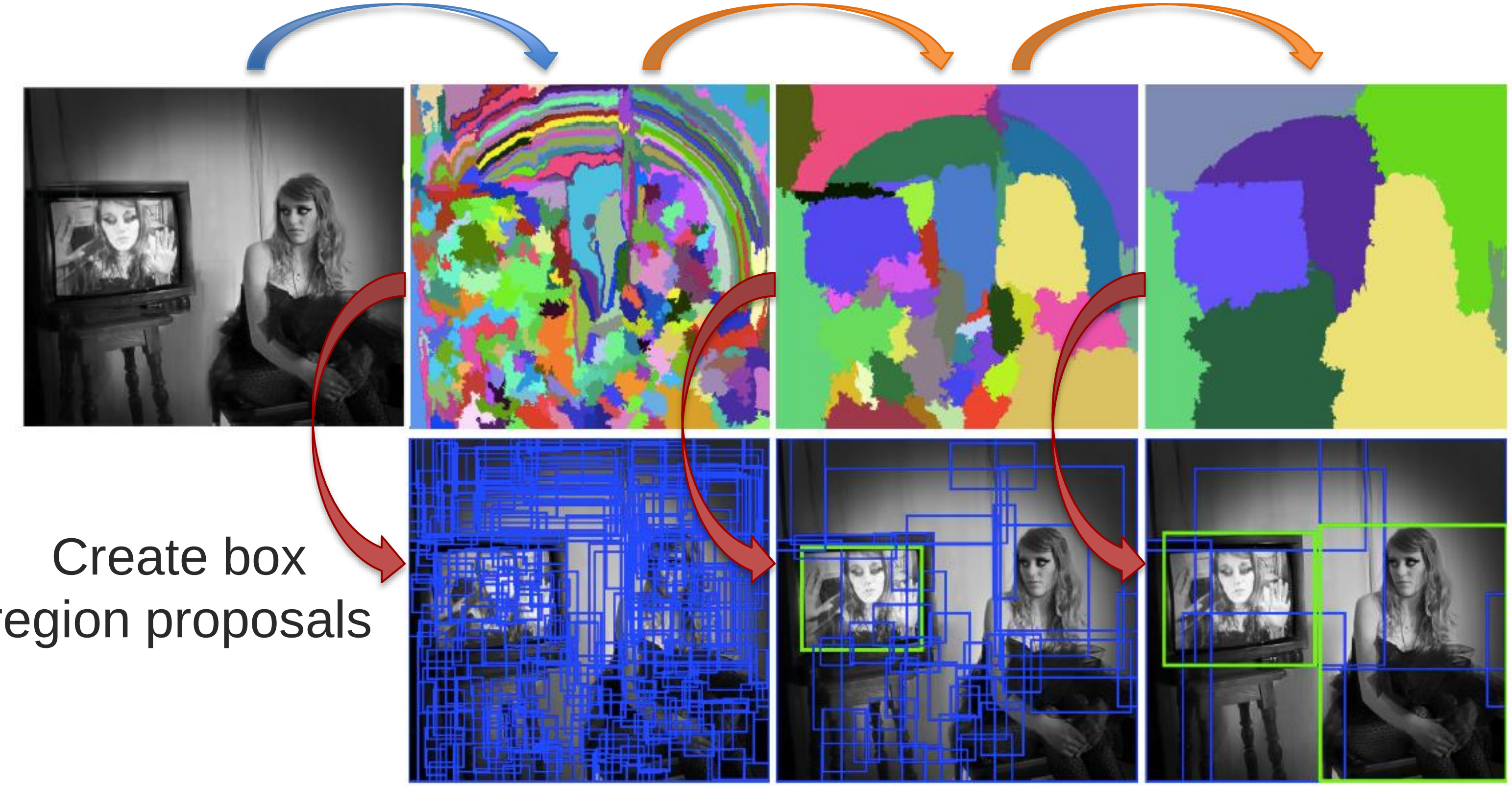
A better option: Start by Identifying hundreds of region proposals and then apply our CNN object detector

How to efficiently identify region proposals?

Selective Search [Uijlings et al., IJCV 2013]

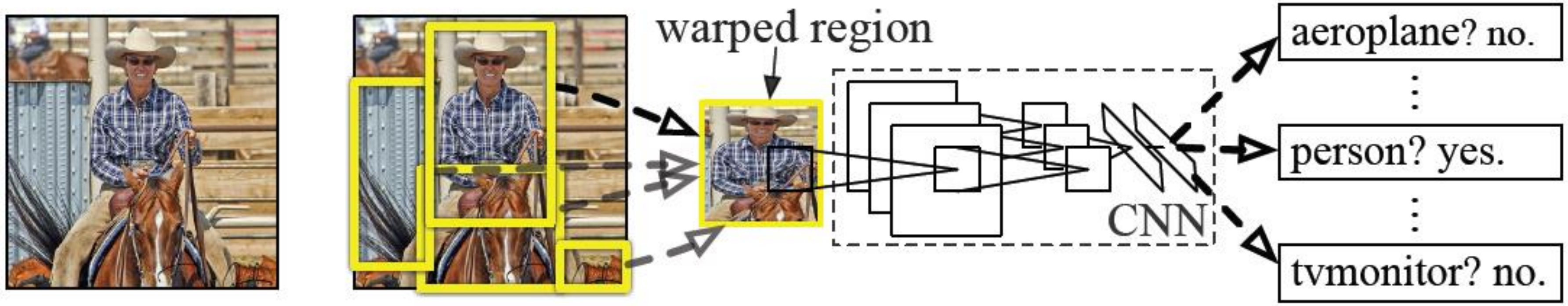
Image segmentation
(using *superpixels*)

And then merge
similar regions



Create box
region proposals

R-CNN [Girshick et al., CVPR 2014]

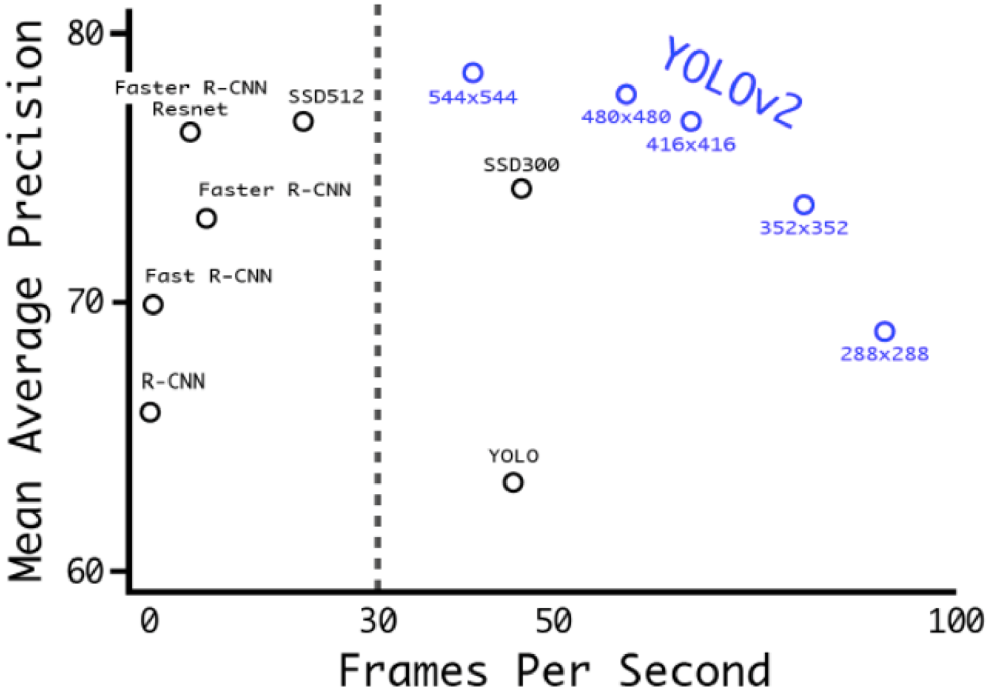


- Select ~2000 region proposals → Time consuming!
- Warp each region
- Apply CNN to each region → Time consuming!

Fast R-CNN: Applies CNN only once, and then extracts regions

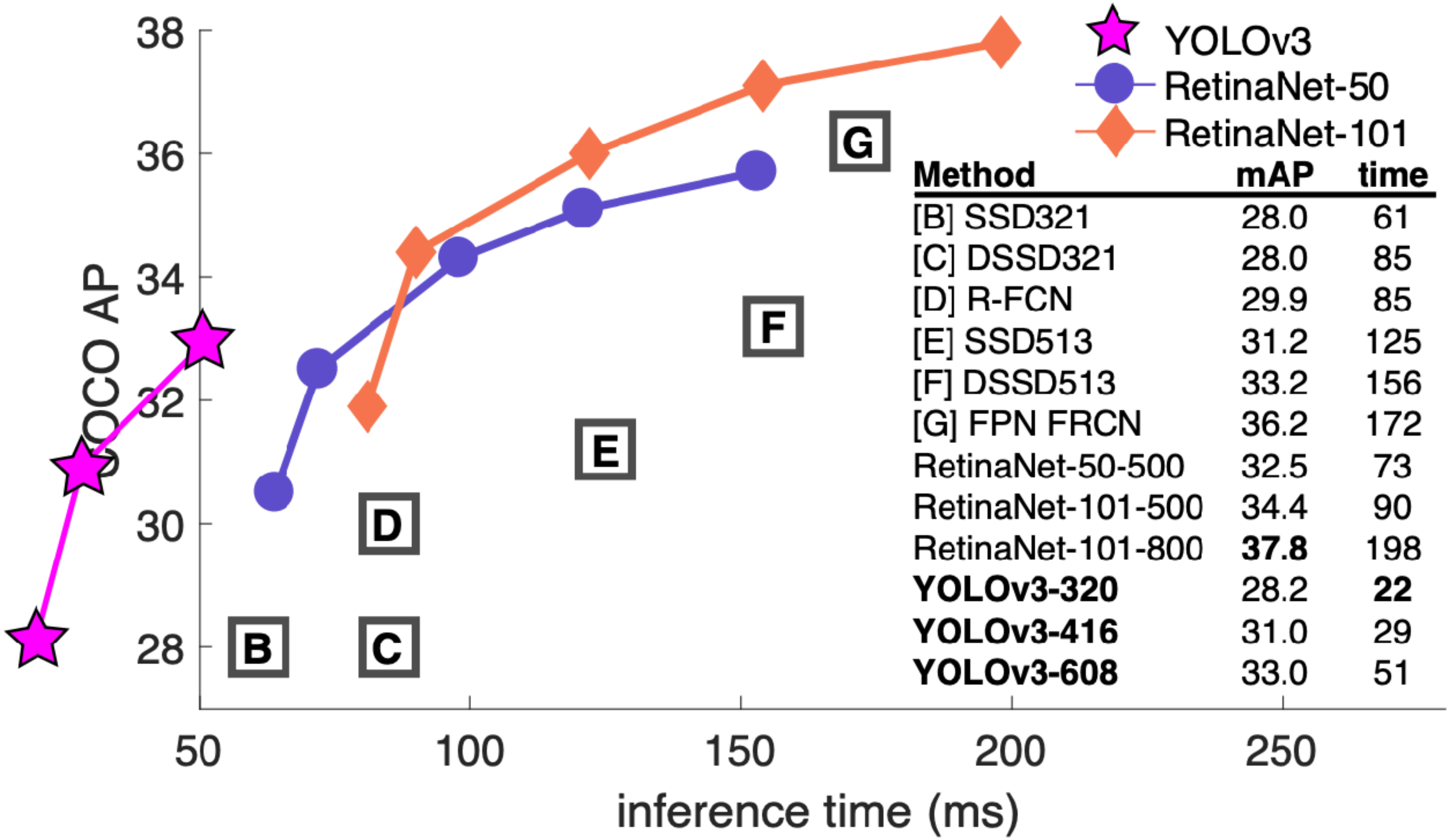
Faster R-CNN: Region selection on the Conv5 response map

Trade-off Between Speed and Accuracy



YOLO: You Only Look Once (CVPR 2016, 2017)
SSD: Single Shot MultiBox Detector (ECCV 2016)

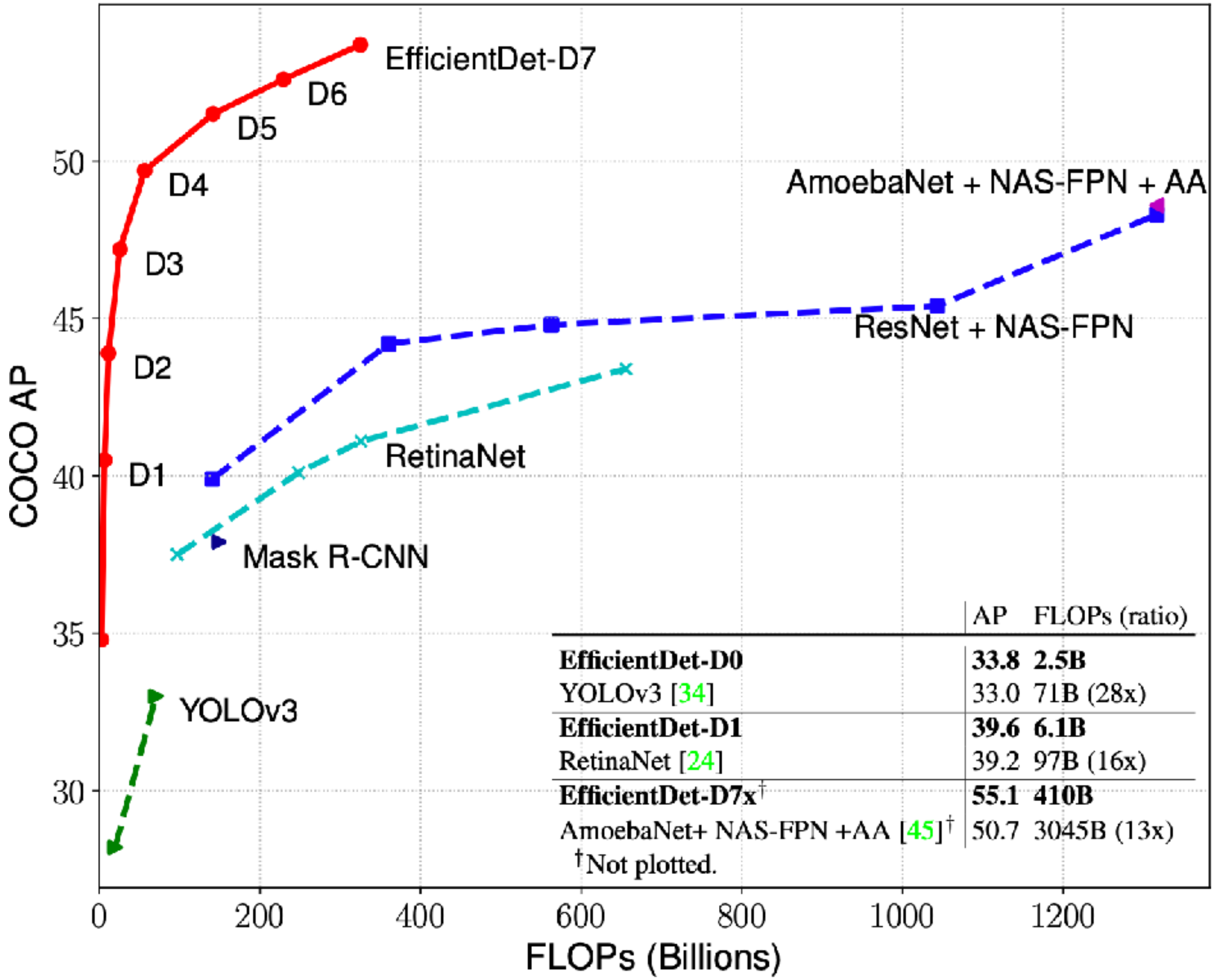
Trade-off Between Speed and Accuracy



Method	mAP	time
[B] SSD321	28.0	61
[C] DSSD321	28.0	85
[D] R-FCN	29.9	85
[E] SSD513	31.2	125
[F] DSSD513	33.2	156
[G] FPN FRCN	36.2	172
RetinaNet-50-500	32.5	73
RetinaNet-101-500	34.4	90
RetinaNet-101-800	**37.8**	198
YOLOv3-320	28.2	**22**
YOLOv3-416	31.0	29
YOLOv3-608	33.0	51

https://arxiv.org/abs/1804.02767

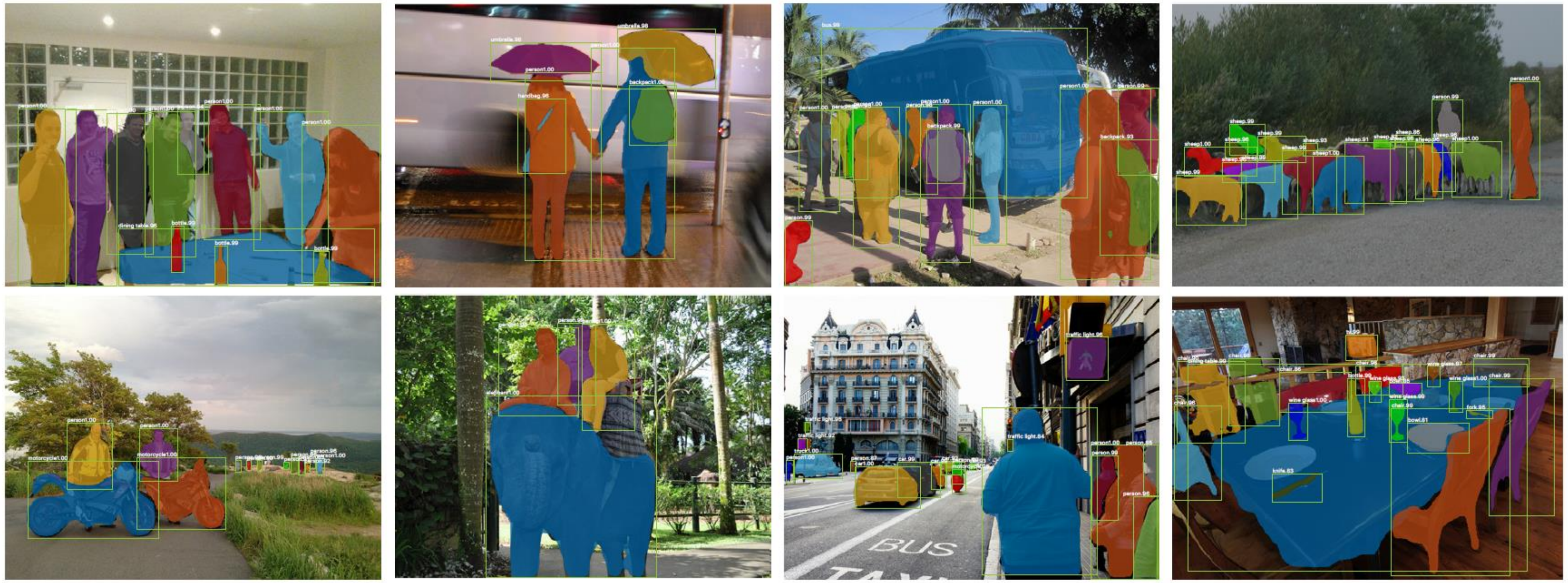
Trade-off Between Speed and Accuracy



https://arxiv.org/abs/1911.09070

Mask R-CNN: Detection and Segmentation

(He et al., 2018)



Sequential Modeling with Convolutional Networks

Modeling Temporal and Sequential Data

How to represent a video sequence?

One option: Recurrent Neural Networks

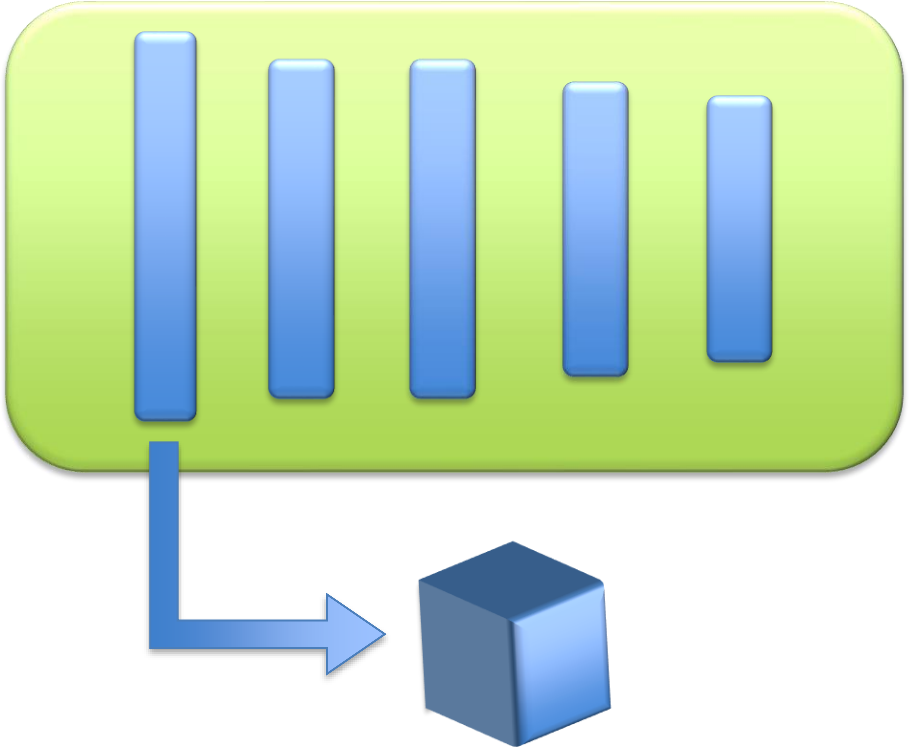
(more about this on Thursday)

3D CNN

Input as a 3D tensor

(stacking video images)

3D CNN



First layer with 3D kernels

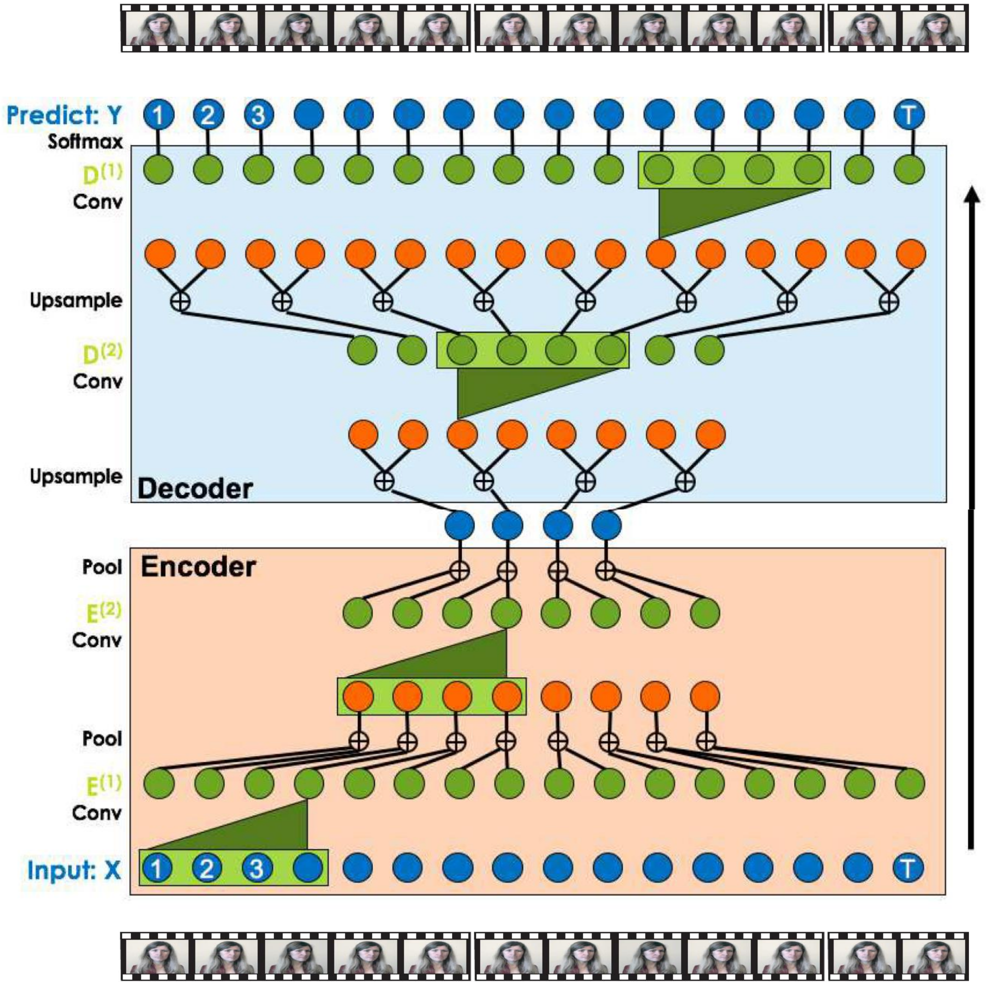
Time-Delay Neural Network

Alexander Waibel, Phoneme Recognition Using Time-Delay Neural Networks, SP87-100, Meeting of the Institute of Electrical, Information and Communication Engineers (IEICE), December, 1987,Tokyo, Japan.

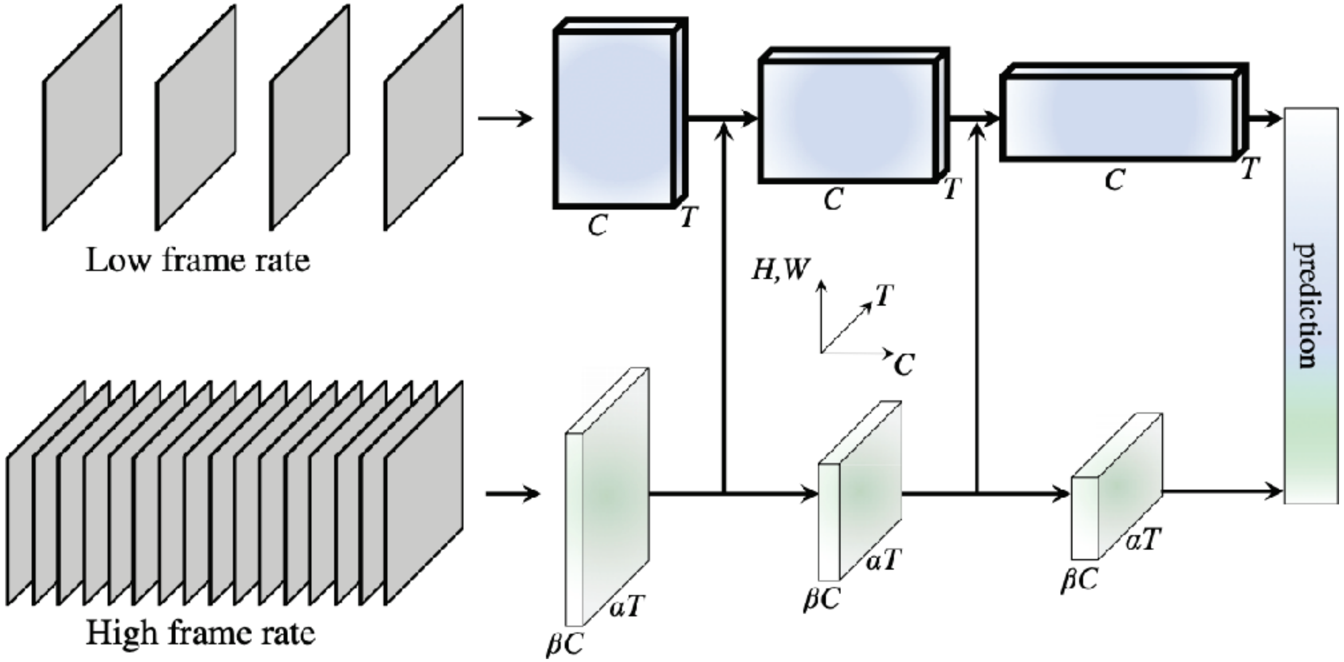
Temporal Convolution Network (TCN) [Lea et al., CVPR 2017]

Decoder

Encoder



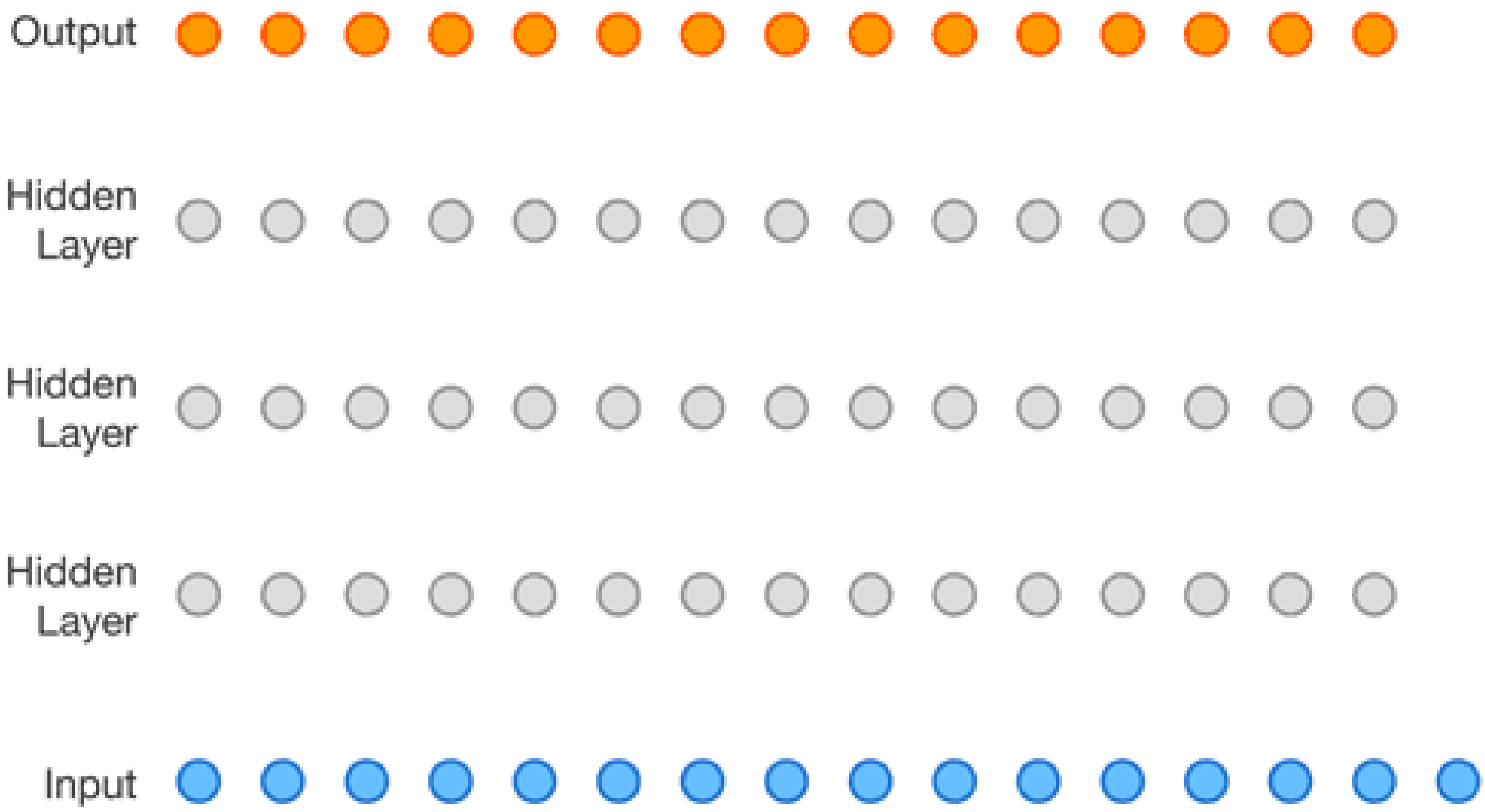
SlowFast Networks for Video Recognition



stage	*Slow* pathway	*Fast* pathway	output sizes $T\times S^2$
raw clip	-	-	64×224^2
data layer	stride 16, 1^2	stride **2**, 1^2	*Slow* : 4×224^2 *Fast* : 32×224^2
conv_1	1×7^2, 64 stride 1, 2^2	$\underline{5\times 7^2}$, 8 stride 1, 2^2	*Slow* : 4×112^2 *Fast* : 32×112^2
pool_1	1×3^2 max stride 1, 2^2	1×3^2 max stride 1, 2^2	*Slow* : 4×56^2 *Fast* : 32×56^2
res_2	$\begin{bmatrix} 1\times 1^2, 64 \\ 1\times 3^2, 64 \\ 1\times 1^2, 256 \end{bmatrix}\times 3$	$\begin{bmatrix} \underline{3\times 1^2}, 8 \\ 1\times 3^2, 8 \\ 1\times 1^2, 32 \end{bmatrix}\times 3$	*Slow* : 4×56^2 *Fast* : 32×56^2
res_3	$\begin{bmatrix} 1\times 1^2, 128 \\ 1\times 3^2, 128 \\ 1\times 1^2, 512 \end{bmatrix}\times 4$	$\begin{bmatrix} \underline{3\times 1^2}, 16 \\ 1\times 3^2, 16 \\ 1\times 1^2, 64 \end{bmatrix}\times 4$	*Slow* : 4×28^2 *Fast* : 32×28^2
res_4	$\begin{bmatrix} \underline{3\times 1^2}, 256 \\ 1\times 3^2, 256 \\ 1\times 1^2, 1024 \end{bmatrix}\times 6$	$\begin{bmatrix} \underline{3\times 1^2}, 32 \\ 1\times 3^2, 32 \\ 1\times 1^2, 128 \end{bmatrix}\times 6$	*Slow* : 4×14^2 *Fast* : 32×14^2
res_5	$\begin{bmatrix} \underline{3\times 1^2}, 512 \\ 1\times 3^2, 512 \\ 1\times 1^2, 2048 \end{bmatrix}\times 3$	$\begin{bmatrix} \underline{3\times 1^2}, 64 \\ 1\times 3^2, 64 \\ 1\times 1^2, 256 \end{bmatrix}\times 3$	*Slow* : 4×7^2 *Fast* : 32×7^2
global average pool, concate, fc			# classes

https://arxiv.org/abs/1812.03982

WaveNet: A Generative Model for Raw Audio



https://deepmind.com/blog/article/wavenet-generative-model-raw-audio

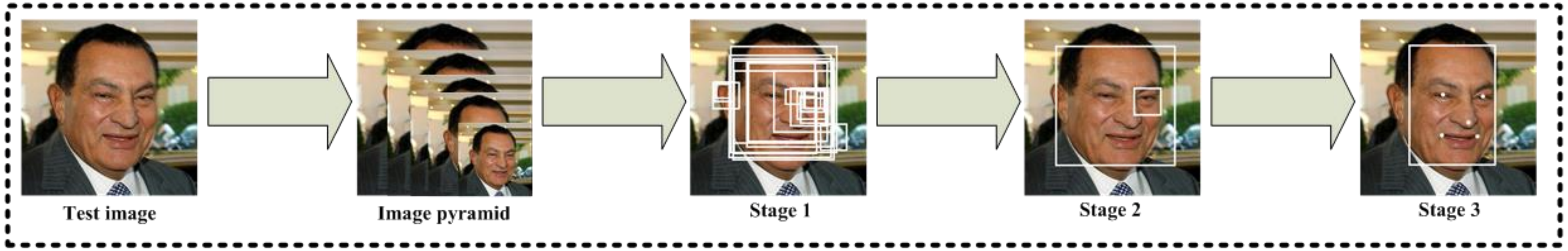
Appendix: Tools for Automatic visual behavior analysis

Automatic analysis of visual behavior

- Face detection
- Face tracking
    - Facial landmark detecion
- Head pose
- Eye gaze tracking
- Facial expression analysis
- Body pose tracking

Face Detection – Multi-Task CNN

[SPL 2016]



Stage 1: candidate windows are produced through a fast Proposal Network

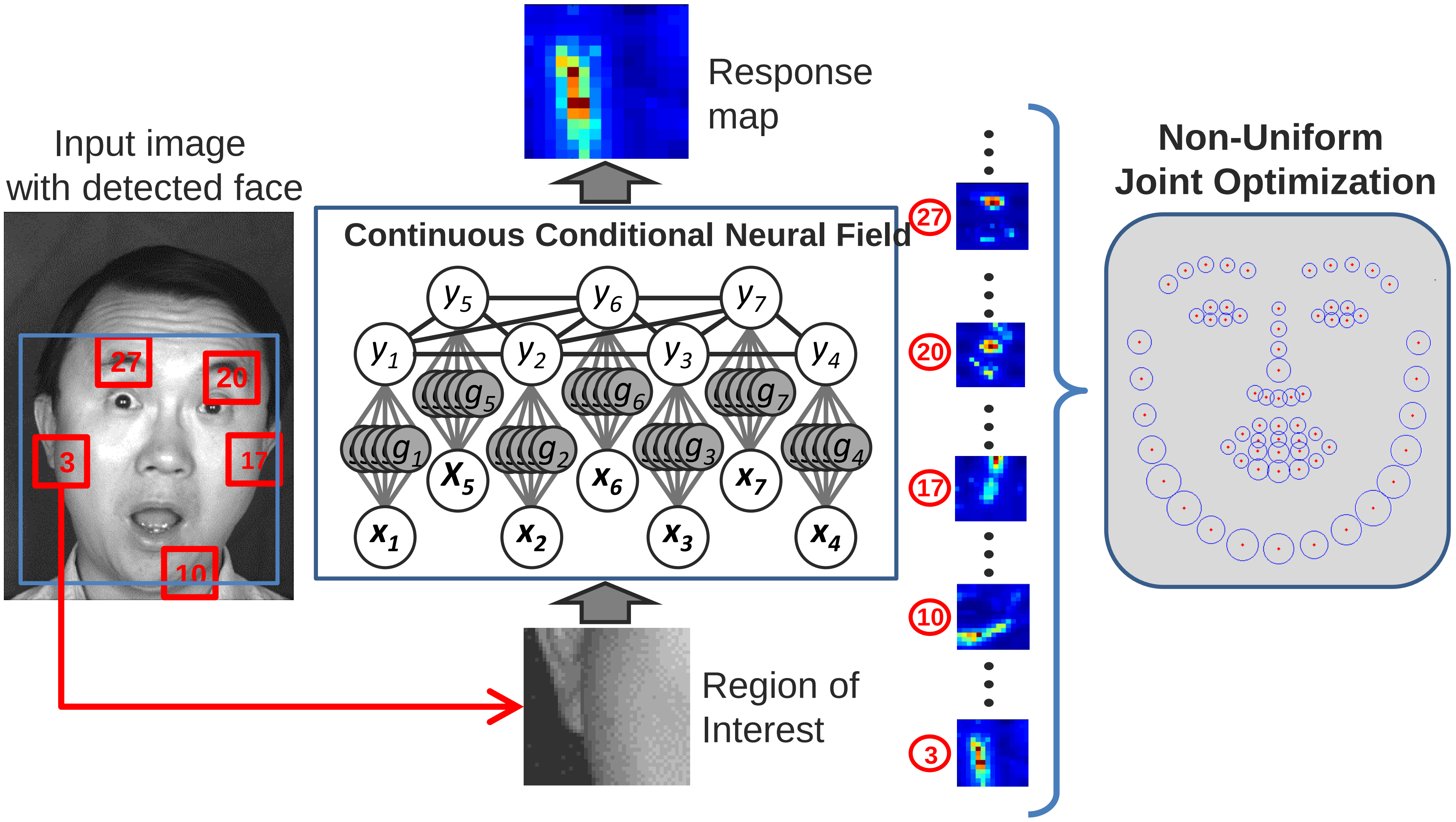
Stage 2: refine these candidates through a Refinement Network

Stage 3: produces final bounding box and facial landmarks position

Existing software (face detection)

- Multi-Task CNN face detector
  - https://kpzhang93.github.io/MTCNN_face_detection_alignment/index.html
- OpenCV (Viola-Jones detector)
- dlib (HOG + SVM)
  - http://dlib.net/
- Tree based model (accurate but very slow)
  - http://www.ics.uci.edu/~xzhu/face/
- HeadHunter (accurate but slow)
  - http://markusmathias.bitbucket.org/2014_eccv_face_detection/
- NPD
  - http://www.cbsr.ia.ac.cn/users/scliao/projects/npdface/

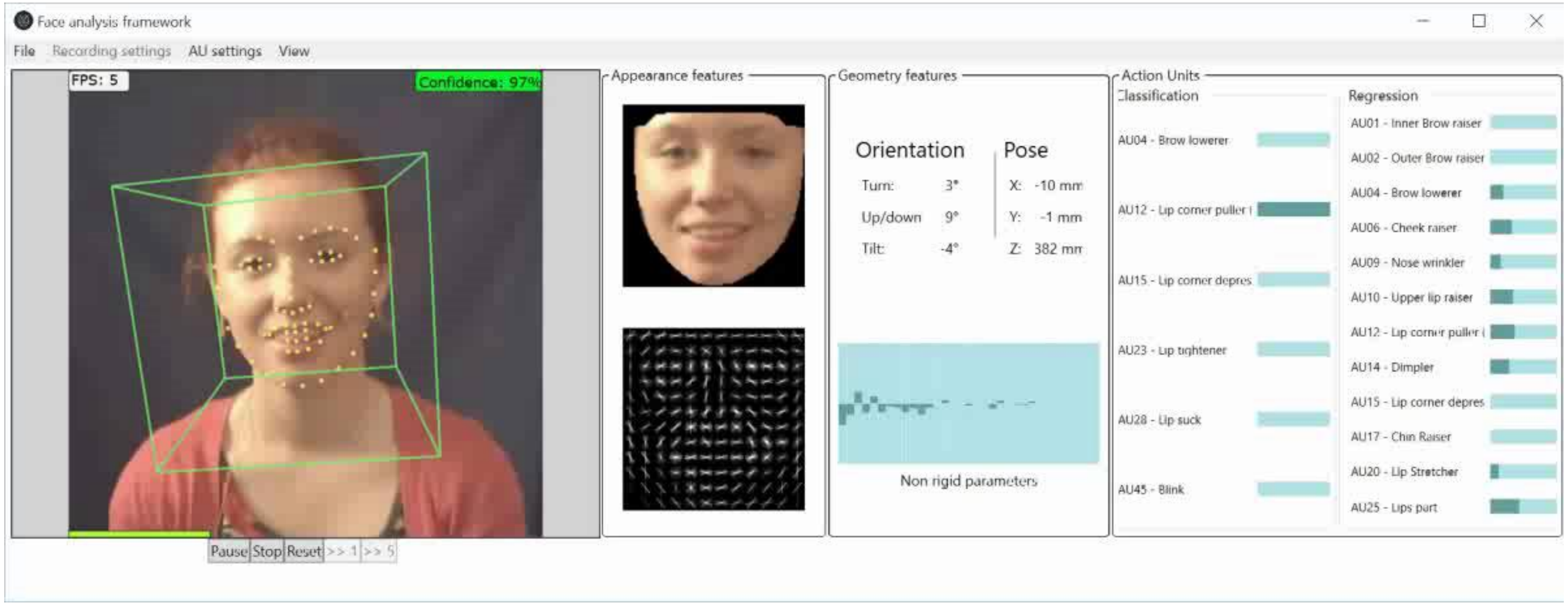
Facial Landmarks: Constrained Local Neural Field



Existing software (facial landmarks)

- OpenFace: facial features
  - https://github.com/TadasBaltrusaitis/OpenFace
- Chehra face tracking
  - https://sites.google.com/site/chehrahome/
- Menpo project (good AAM, CLM learning tool)
  - http://www.menpo.org/
- IntraFace: Facial attributes, facial expression analysis
  - http://www.humansensing.cs.cmu.edu/intraface/
- OKAO Vision: Gaze estimation, facial expression
  - http://www.omron.com/ecb/products/mobile/okao03.html (Commercial software)
- VisageSDK
  - http://www.visagetechnologies.com/products/visagesdk/
  - (Commercial software)

Facial expression analysis



[**OpenFace: an open source facial behavior analysis toolkit**, T. Baltrušaitis et al., 2016]

Existing Software (expression analysis)

- OpenFace: Action Units
  - https://github.com/TadasBaltrusaitis/OpenFace
- Shore: facial tracking, smile detection, age and gender detection
  - http://www.iis.fraunhofer.de/en/bf/bsy/fue/isyst/detektion/
- FACET/CERT (Emotient API): Facial expression recognition
  - http://imotionsglobal.com/software/add-on-modules/attention-tool-facet-module-facial-action-coding-system-facs/ (Commercial software)
- Affdex
  - http://www.affectiva.com/solutions/apis-sdks/
  - (commercial software)

Gaze Estimation – Eye, Head and Body

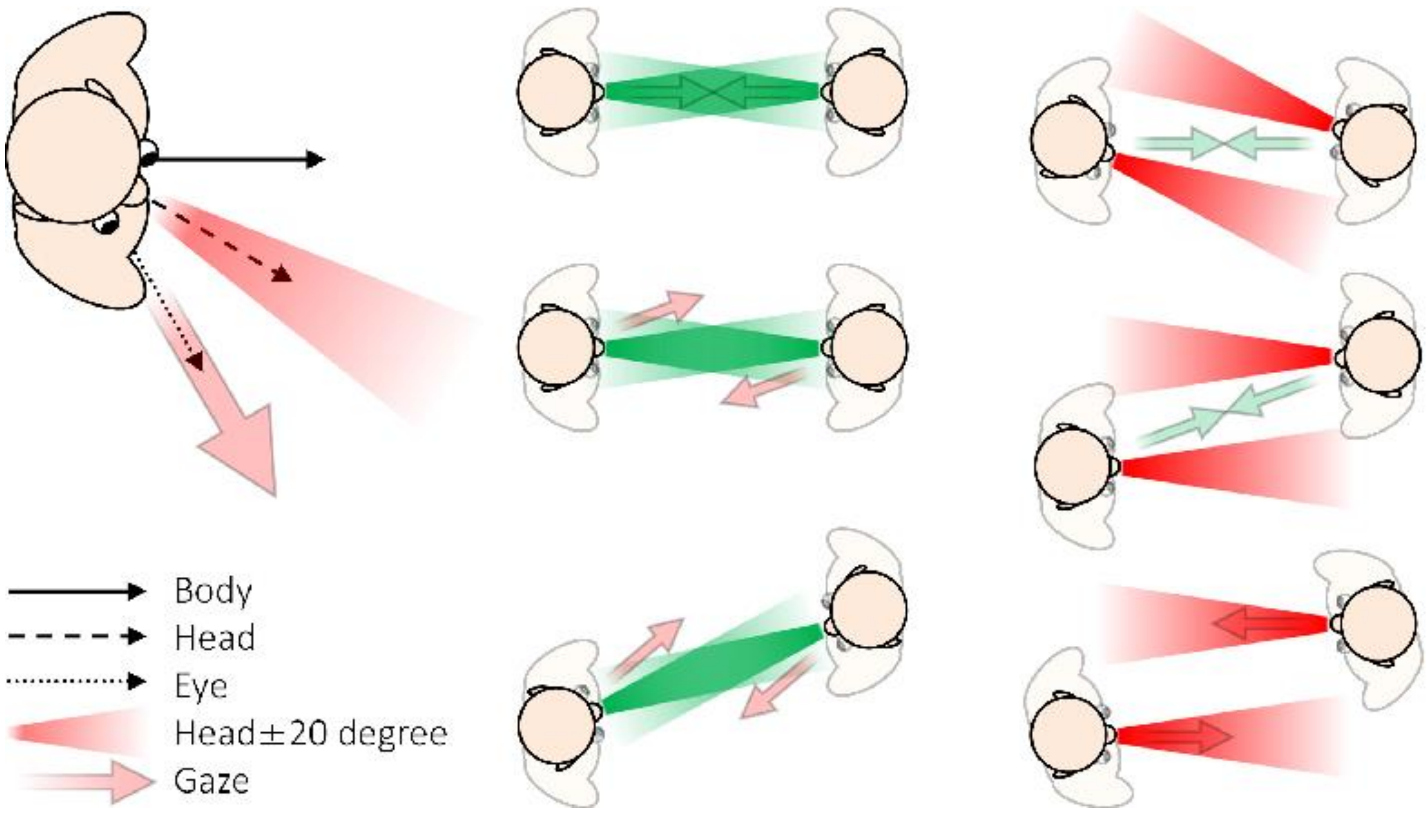
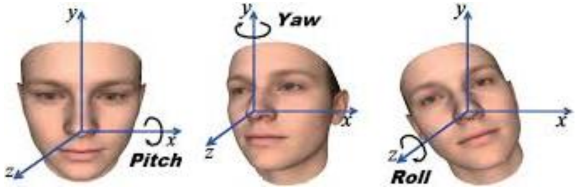


Image from Hachisu et al (2018). FaceLooks: A Smart Headband for Signaling Face-to-Face Behavior. *Sensors*.

Existing Software (head gaze)



- OpenFace
  - https://github.com/TadasBaltrusaitis/OpenFace
- Chehra face tracking
  - https://sites.google.com/site/chehrahome/
- Watson: head pose estimation
  - http://sourceforge.net/projects/watson/
- Random forests
  - http://www.vision.ee.ethz.ch/~gfanelli/head_pose/head_forest.html
  - (requires a Kinect)
- IntraFace
  - http://www.humansensing.cs.cmu.edu/intraface/

Existing Software (eye gaze)

- OpenFace: gaze from a webcam
  - https://github.com/TadasBaltrusaitis/OpenFace
- EyeAPI: eye pupil detection
  - http://staff.science.uva.nl/~rvalenti/
- EyeTab
  - https://www.cl.cam.ac.uk/research/rainbow/projects/eyetab/
- OKAO Vision: Gaze estimation, facial expression
  - http://www.omron.com/ecb/products/mobile/okao03.html (Commercial software)

Articulated Body Tracking: OpenPose

Existing Software (body tracking)

- OpenPose
  - https://github.com/CMU-Perceptual-Computing-Lab/openpose
- Microsoft Kinect
  - http://www.microsoft.com/en-us/kinectforwindows/
- OpenNI
  - http://openni.org/
- Convolutional Pose Machines
  - https://github.com/shihenw/convolutional-pose-machines-release

Visual Descriptors

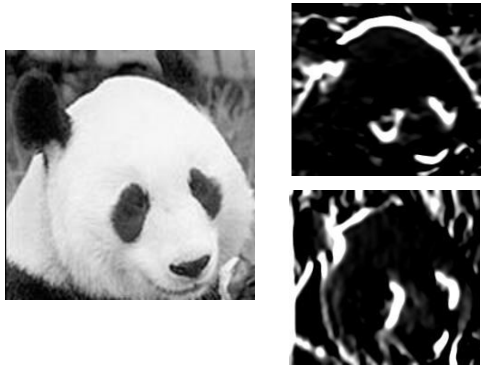
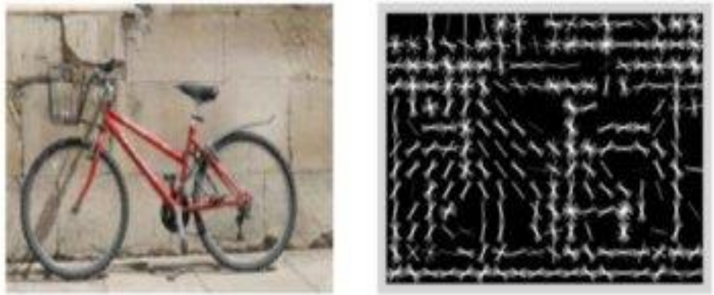


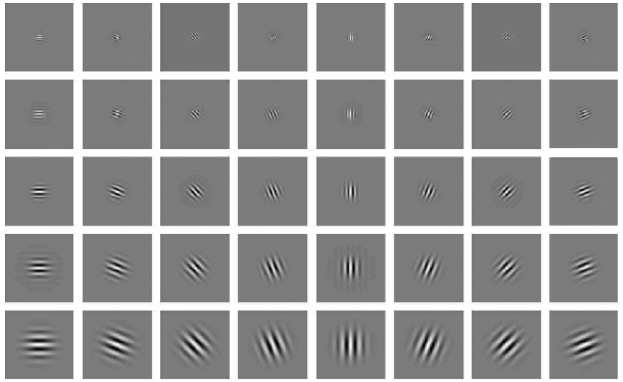
Image gradient

Edge detection

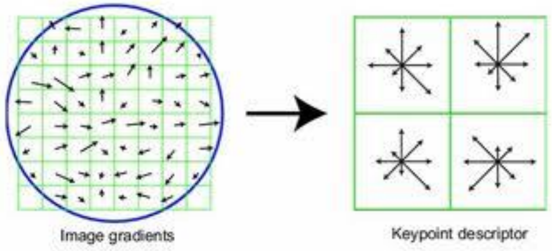


Histograms of Oriented Gradients

Optical Flow



Gabor Jets



SIFT descriptors

Existing Software (visual descriptors)

- OpenCV: optical flow, gradient, Haar filters…
- SIFT descriptors
  - http://blogs.oregonstate.edu/hess/code/sift/
- dlib – HoG
  - http://dlib.net/
- OpenFace: Aligned HoG for faces
  - https://github.com/TadasBaltrusaitis/CLM-framework