Language Technologies Institute

Carnegie Mellon University

Multimodal Machine Learning

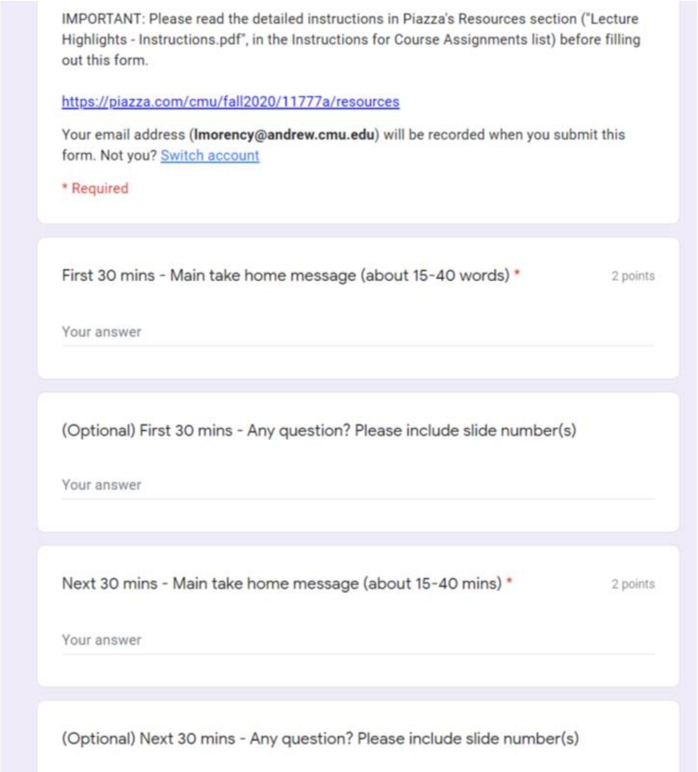
Lecture 2.2: Unimodal Representations

Louis-Philippe Morency

** Co-lecturer: Paul Liang. Original course co-developed with Tadas Baltrusaitis. Spring 2021 and 2022 editions taught by Yonatan Bisk*

Administrative Stuff

Lecture Highlight Form


IMPORTANT: Please read the detailed instructions in Piazza's Resources section ("Lecture Highlights - Instructions.pdf", in the Instructions for Course Assignments list) before filling out this form.

https://piazza.com/cmu/fall2020/11777a/resources

Your email address (**lmorency@andrew.cmu.edu**) will be recorded when you submit this form. Not you? Switch account

* Required

First 30 mins - Main take home message (about 15-40 words) * 2 points

Your answer

(Optional) First 30 mins - Any question? Please include slide number(s)

Your answer

Next 30 mins - Main take home message (about 15-40 mins) * 2 points

Your answer

(Optional) Next 30 mins - Any question? Please include slide number(s)

Deadline: Today, Thursday at 11:59pm ET

Use your Andrew CMU email

- You will need to login using this address

New form for each lecture

- Posted on Piazza’s Resources section

You should start taking as soon as the administrative stuss is over!

Contact us if you have any problem

Reading Assignments – Weekly Schedule

Four main steps for the reading assignments

1. Monday 8pm: Official start of the assignment
2. Wednesday 8pm: Select your paper
3. **Friday 8pm:** Post your summary
4. **Monday 8pm:** Post your extra comments (5 posts)

Team Matching Event – Today!

Today around 11am ET

(later part of the lecture)

- Detailed instructions will be shared during lecture
- Event optional for students who already have a full team

AWS Credits

New procedure this semester!

- We need your AWS account info (deadline: Tuesday 9/13)
- Max $150 credit for the whole semester. No exception.
- More details will be sent on Piazza

Alternative: Amazon SageMaker Studio Lab

- Similar to Google Colab (link)
- No cost, easy access to JupyterLab-based user interface
- Access to G4dn.xlarge instances

Multimodal Machine Learning

Lecture 2.2: Unimodal Representations

Louis-Philippe Morency

** Co-lecturer: Paul Liang. Original course co-developed with Tadas Baltrusaitis. Spring 2021 and 2022 editions taught by Yonatan Bisk*

Lecture Objectives

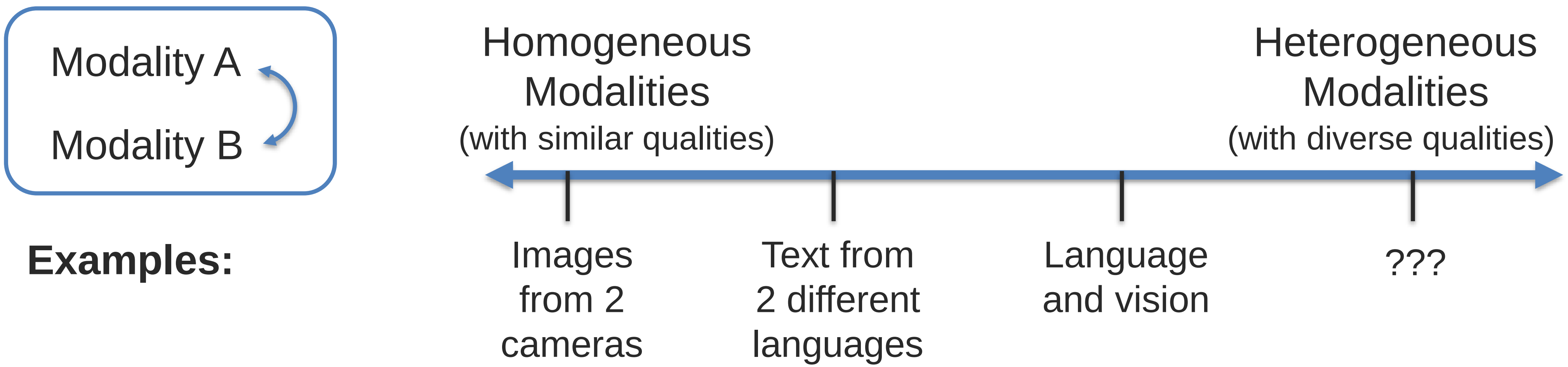
- Dimension of heterogeneity
- Image representations
  - Image gradients, edges, kernels
- Convolution neural network (CNN)
  - Convolution and pooling layers
- Visualizing CNNs
- Region-based CNNs
- Sequence modeling with convolution networks

- Team matching event

Dimensions of Heterogeneity

Heterogeneous Modalities

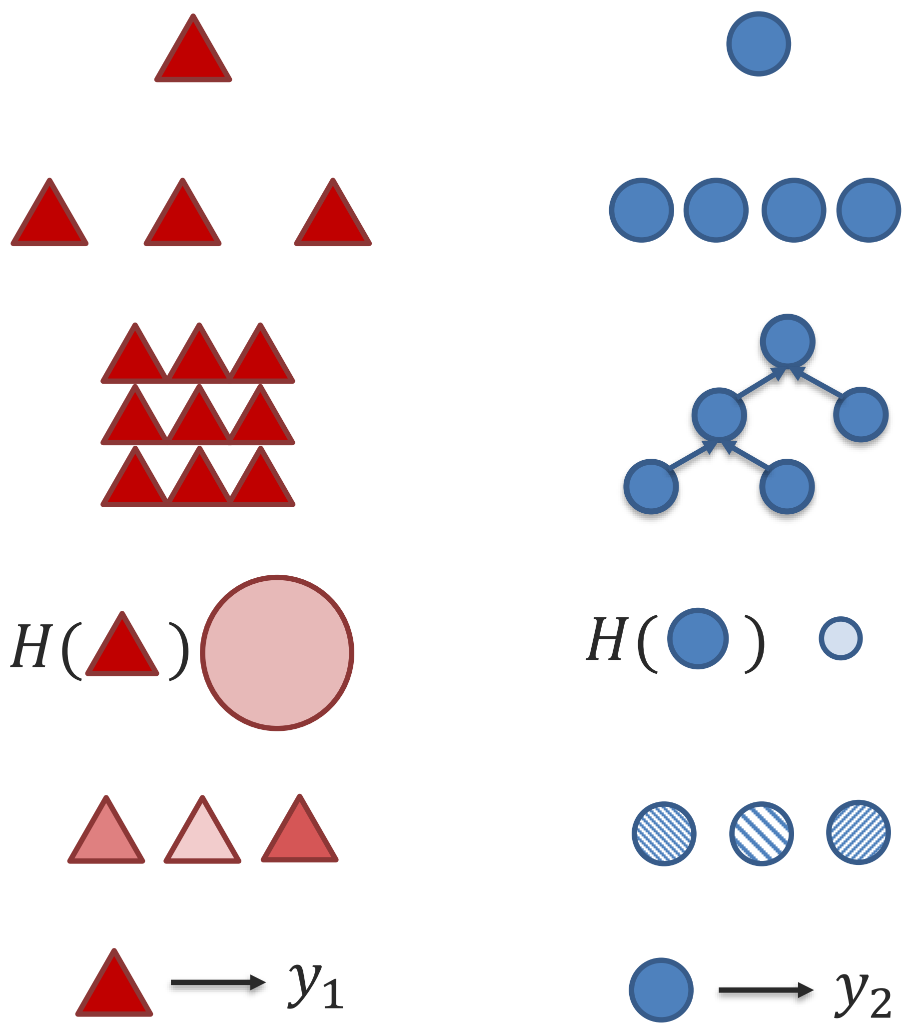
Information present in different modalities will often show diverse qualities, structures and representations.



Dimensions of Heterogeneity

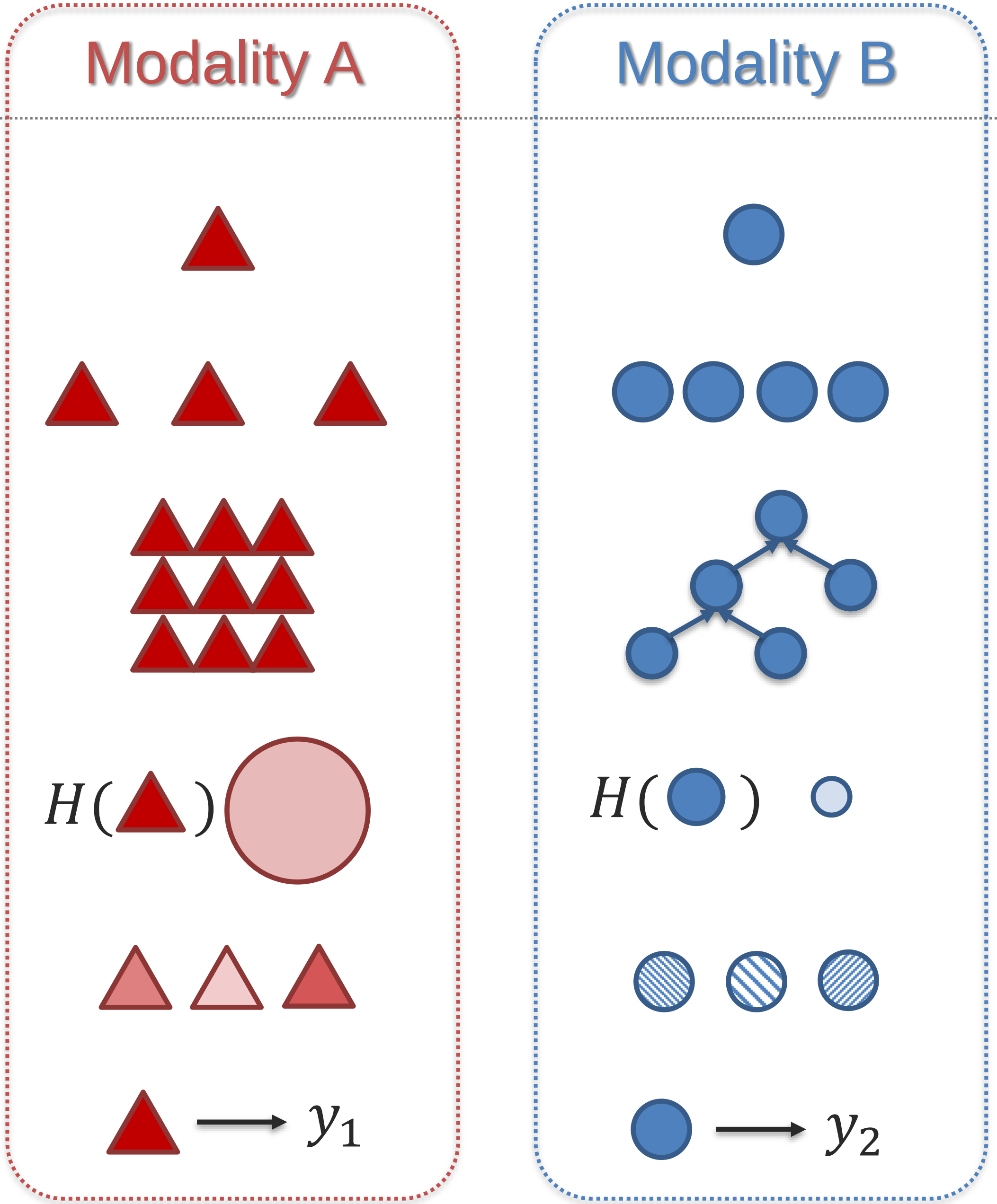
Modality A ⟷ Modality B

1. **Element representations:**
   Discrete, continuous, granularity
2. **Element distributions:**
   Density, frequency
3. **Structure:**
   Temporal, spatial, latent, explicit
4. **Information:**
   Abstraction, entropy
5. **Noise:**
   Uncertainty, noise, missing data
6. **Relevance:**
   Task, context dependence



Modality Profile

1. **Element representations:**
   Discrete, continuous, granularity
2. **Element distributions:**
   Density, frequency
3. **Structure:**
   Temporal, spatial, latent, explicit
4. **Information:**
   Abstraction, entropy
5. **Noise:**
   Uncertainty, noise, missing data
6. **Relevance:**
   Task, context dependence



Modality Profile

1. **Element representations:**
   Discrete, continuous, granularity
2. **Element distributions:**
   Density, frequency
3. **Structure:**
   Temporal, spatial, latent, explicit
4. **Information:**
   Abstraction, entropy
5. **Noise:**
   Uncertainty, noise, missing data
6. **Relevance:**
   Task, context dependence

Visual Image Modality

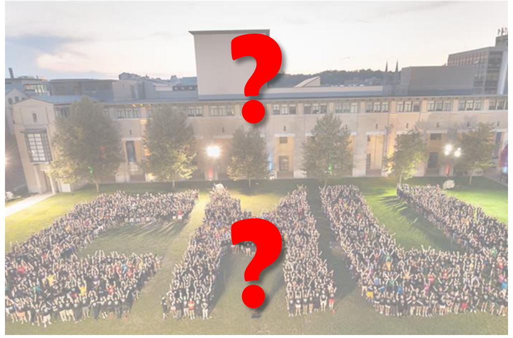
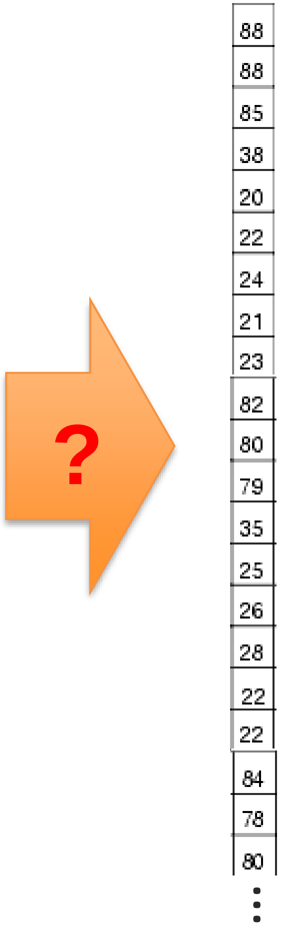
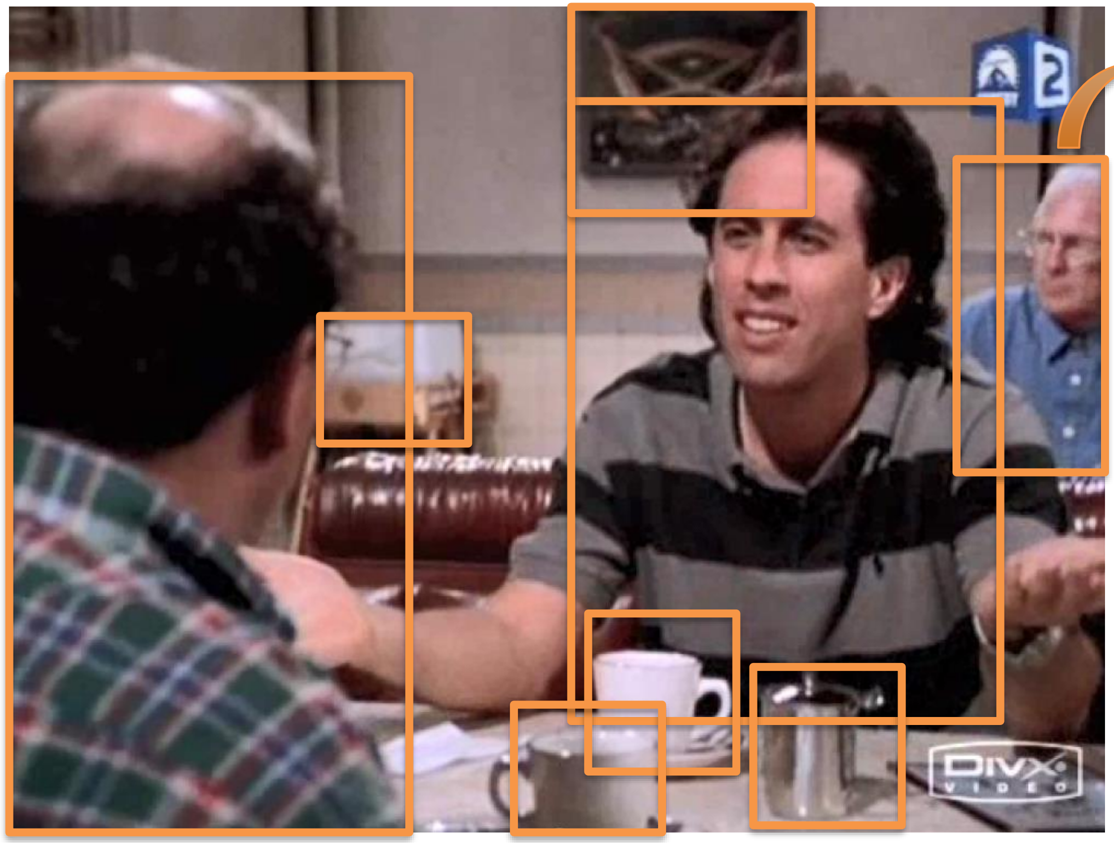


Image Representations

How Would You Describe This Image?



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…

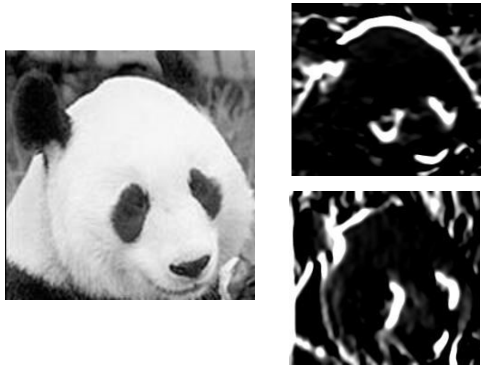
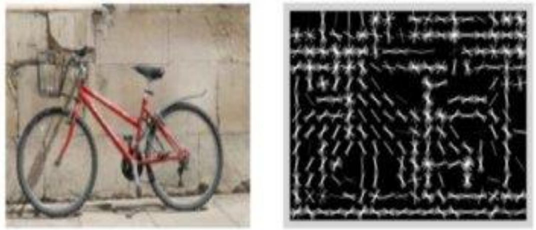


Image gradient

Edge detection



Histograms of Oriented Gradients

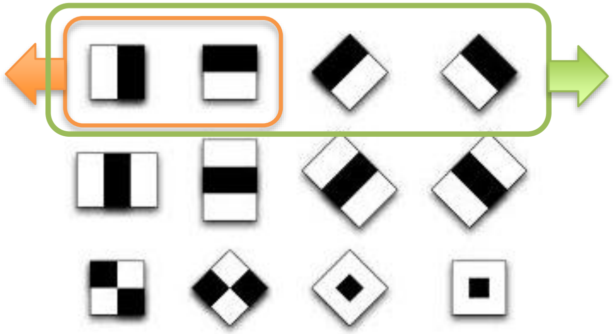
Optical Flow

Object Descriptors

How to represent and detect an object?

Many approaches over the years…

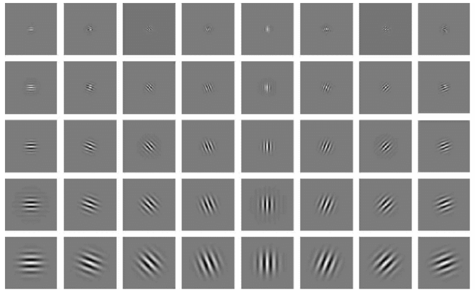
Horizontal and vertical gradients



Oriented gradients

Haar Wavelets

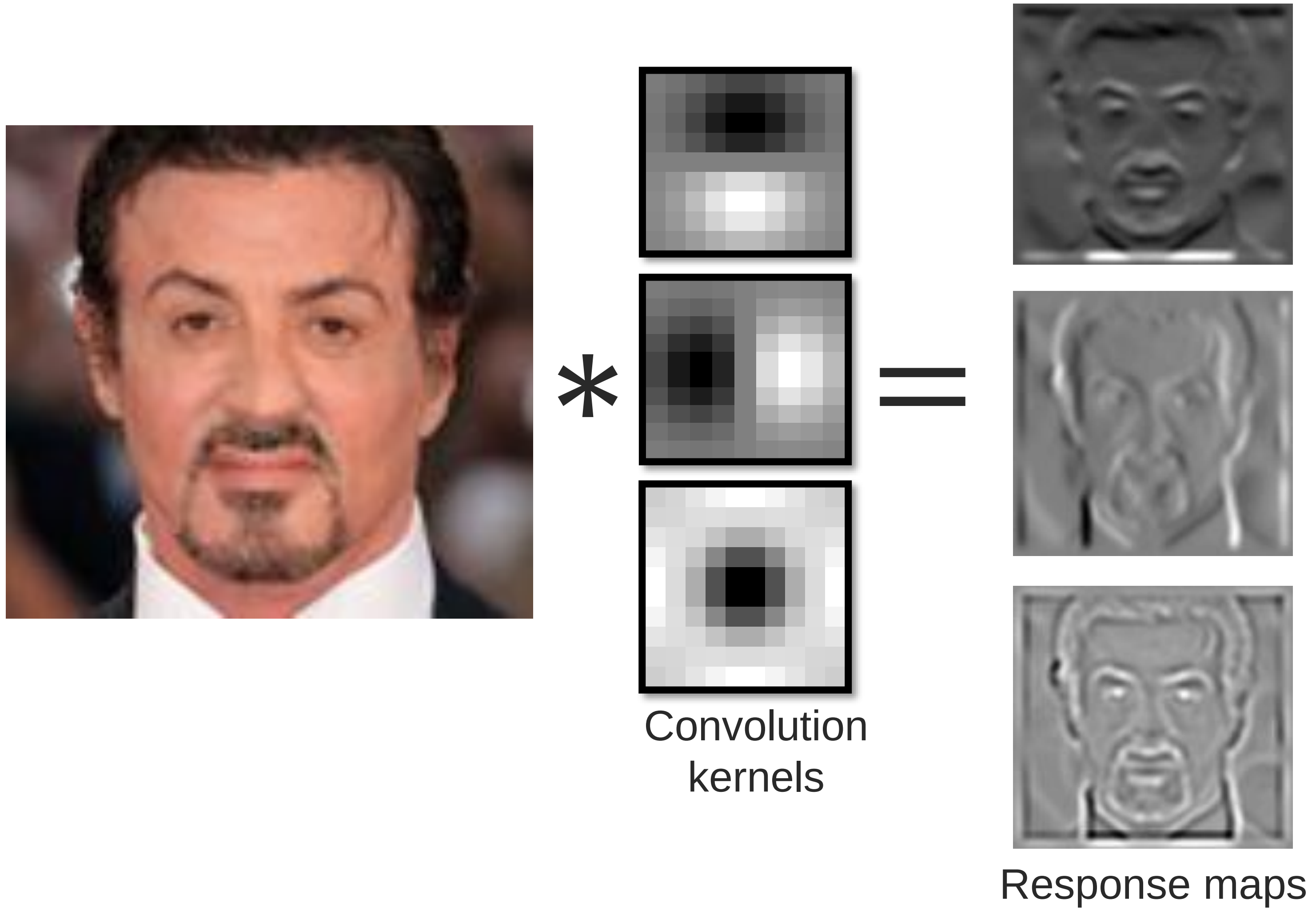
Templates tested on the image *(i.e., convolution kernels)*



Gabor filters

Inspired by visual cortex

Convolution Kernels

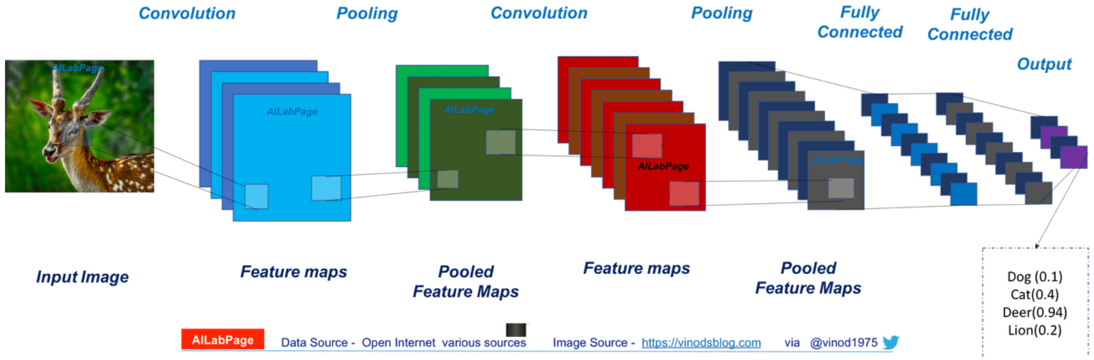


Object Descriptors

How to represent and detect an object?

Many approaches over the years…

Convolutional Neural Network (CNN)



More details about CNNs is coming…

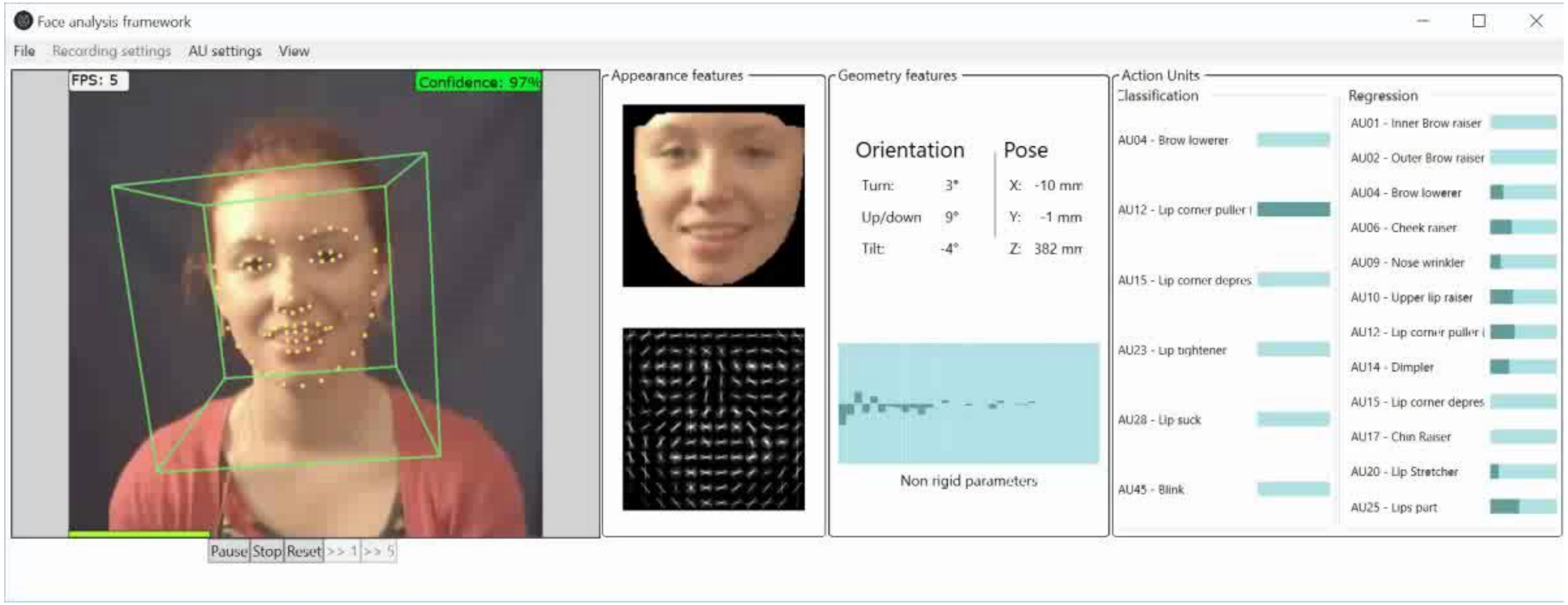
… and we will also talk about visual transformers in coming weeks…

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

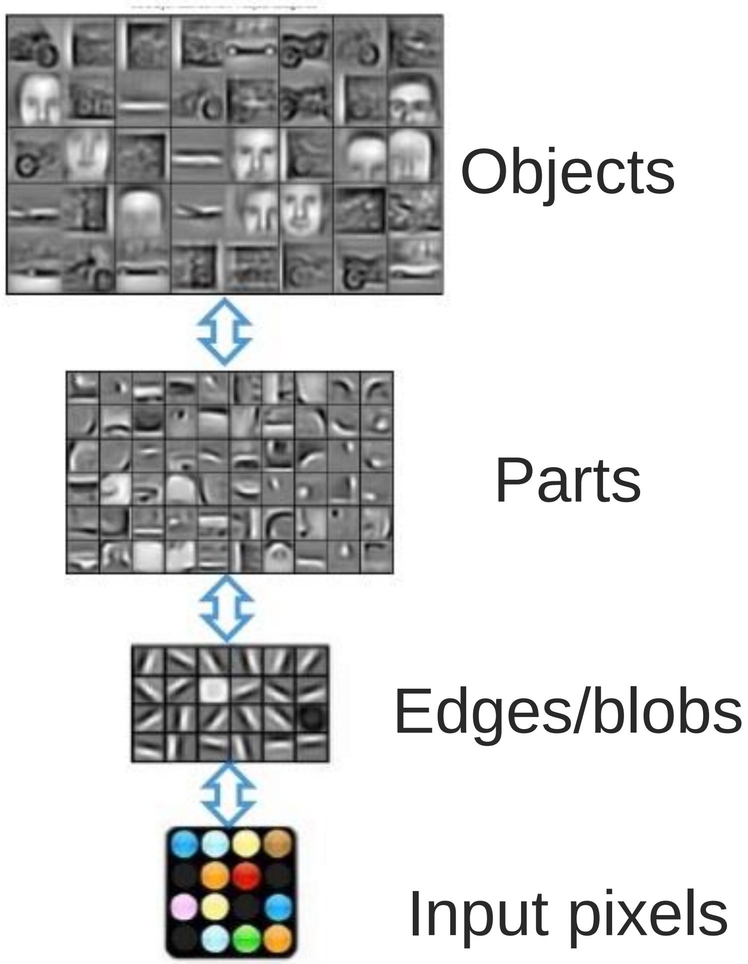
Convolutional Neural Networks

Why using Convolutional Neural Networks?

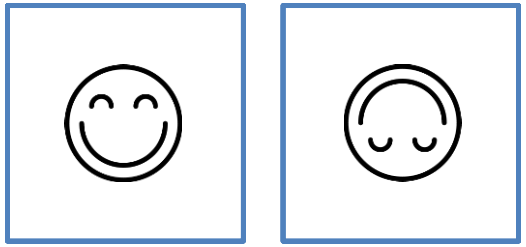
Goal: building more abstract, hierarchical visual representations

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?

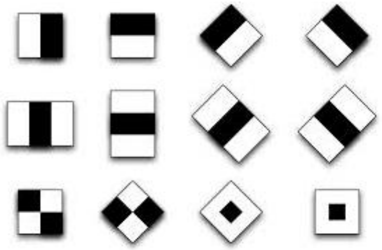
- ➢ The model should still be able to classify it

Conventional MLP models are not translation invariant!

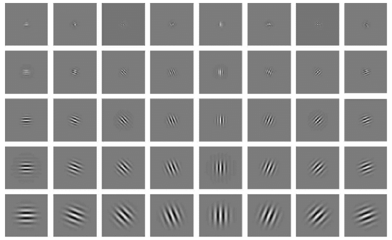
- ➢ But CNNs are kernel-based, which helps with translation invariance and reduce number of parameters

Predefined vs Learned Kernels

Predefined kernels



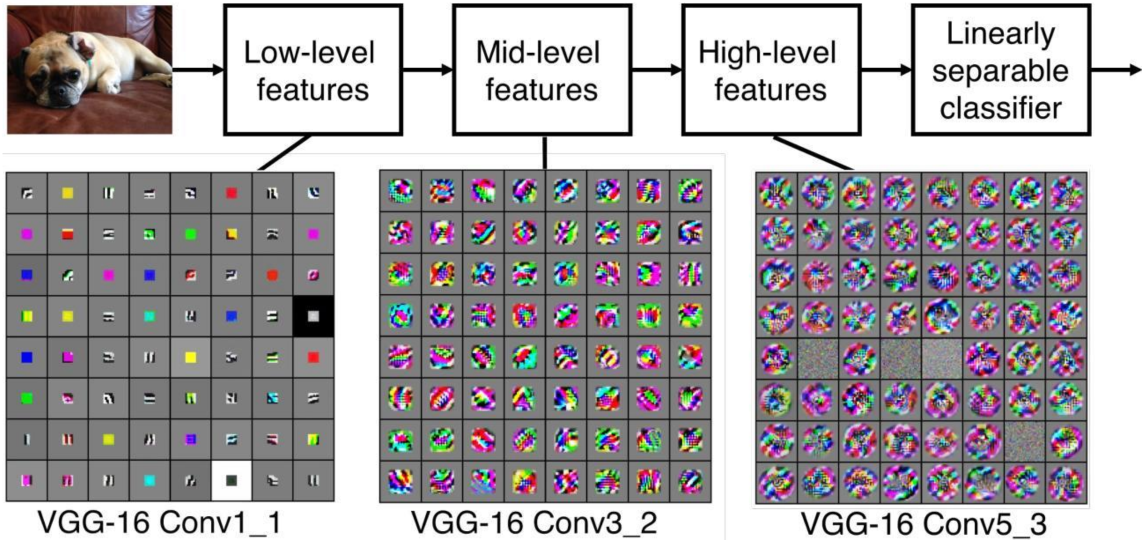
Haar Wavelets



Gabor filters

Learned kernels

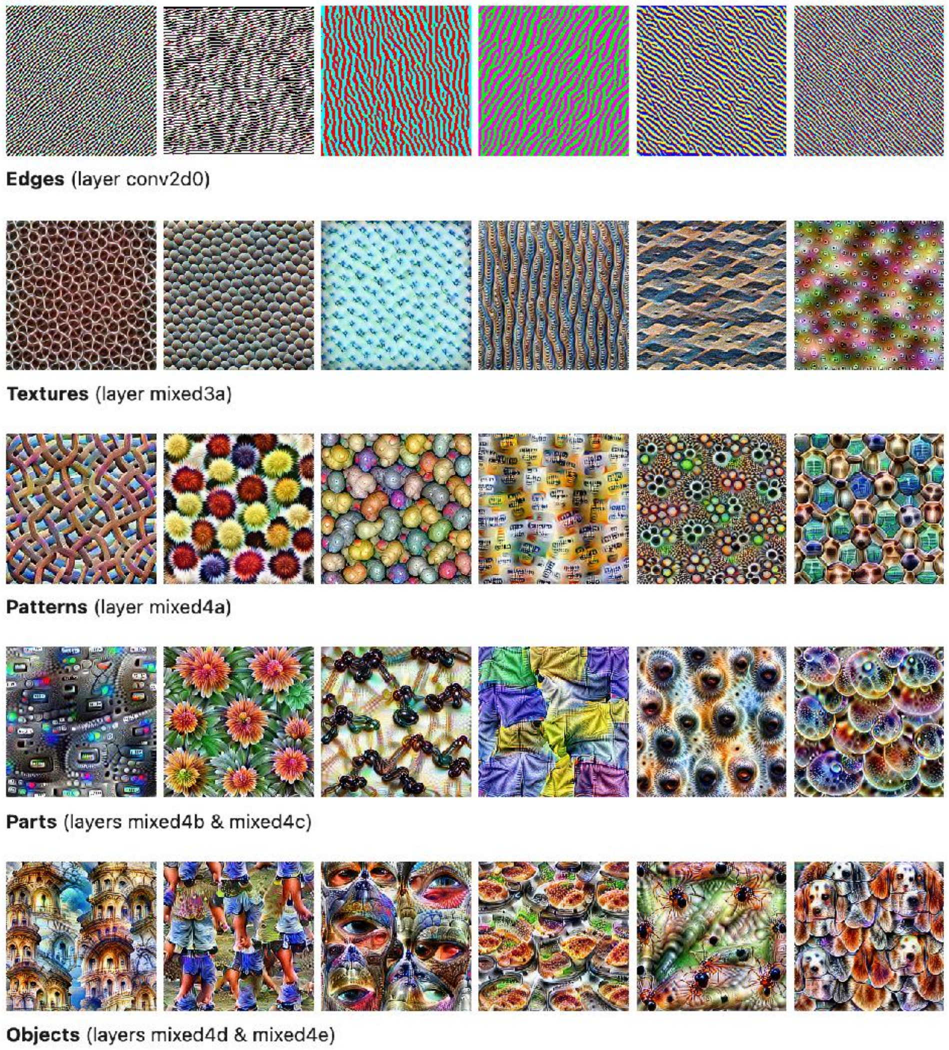
Convolutional Neural Network (CNN)



With CNNs, the kernel values are learned as model parameters

Learned Filters (aka Convolution Kernels)

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



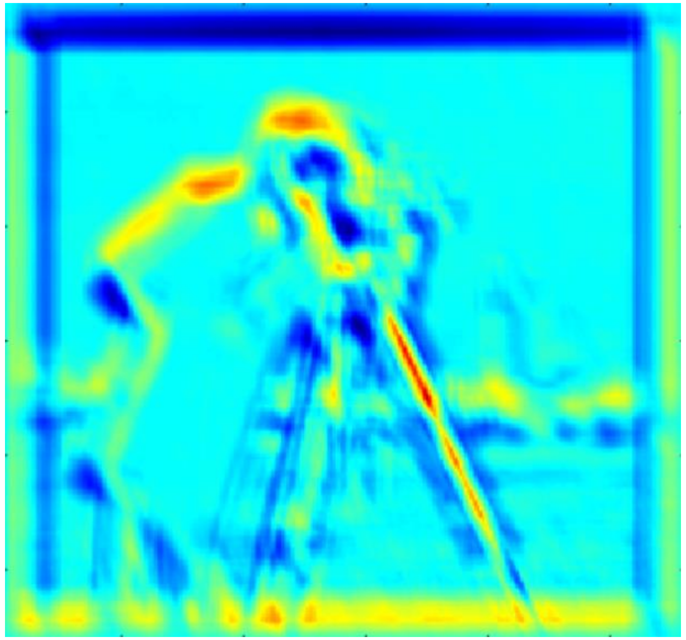
Convolution in 2D – Example

Input image

*

Convolution kernel

=



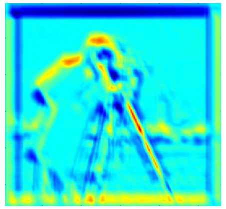
Response map

Convolution as a Fully-Connected Network

Input

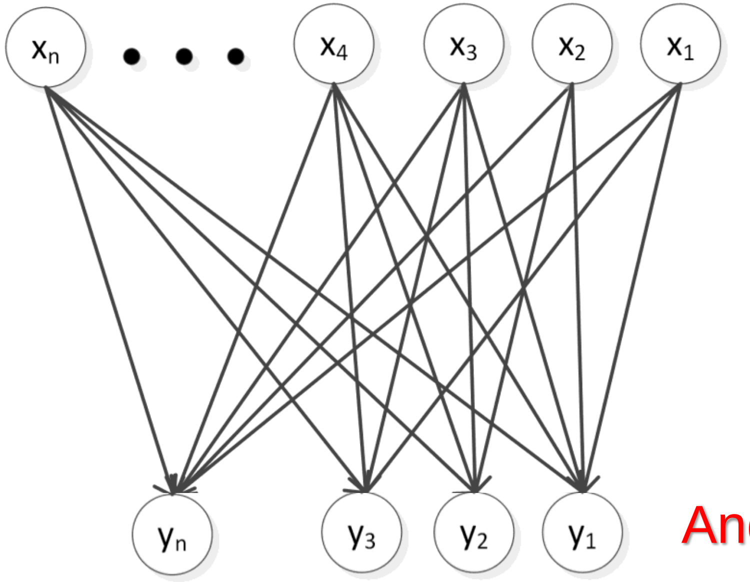
(image)

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

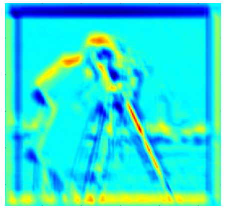
Convolutional Neural Layer

Input

(image)

Weighted sum
Wx

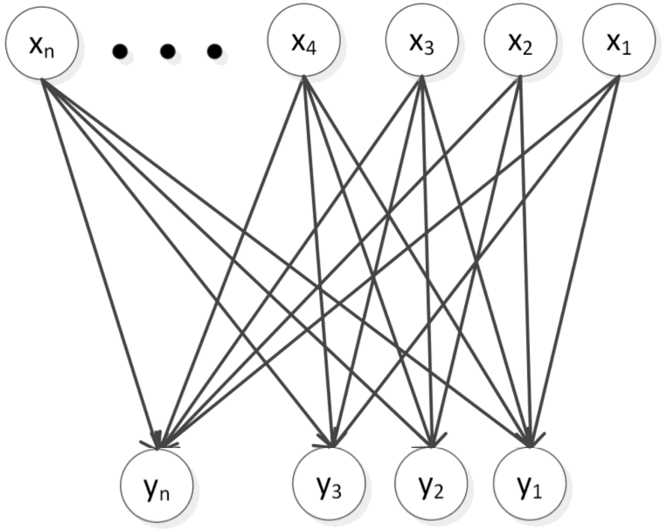
Output



(response map)

$$y = Wx$$

Input: all pixels



Output: kernel responses

Example with 1D kernel:

w_1	w_2	w_3

Convolution kernel

Convolutional Neural Layer

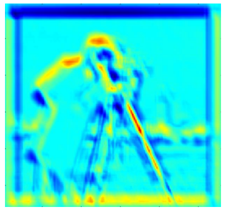
Modification 1: Sliding window – Only apply the kernel to a small region

Input

(image)

Weighted sum
Wx

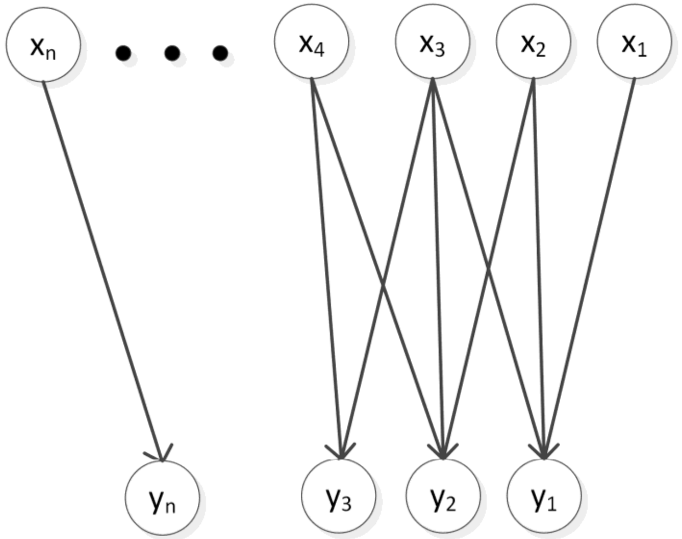
Output



(response map)

$y = Wx$

Input: all pixels



Output: kernel responses

Example with 1D kernel:

w_1	w_2	w_3

Convolutional Neural Layer

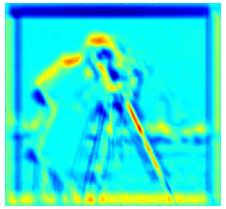
Modification 2: Same kernel applied to all sliding windows

Input

(image)

Weighted sum Wx

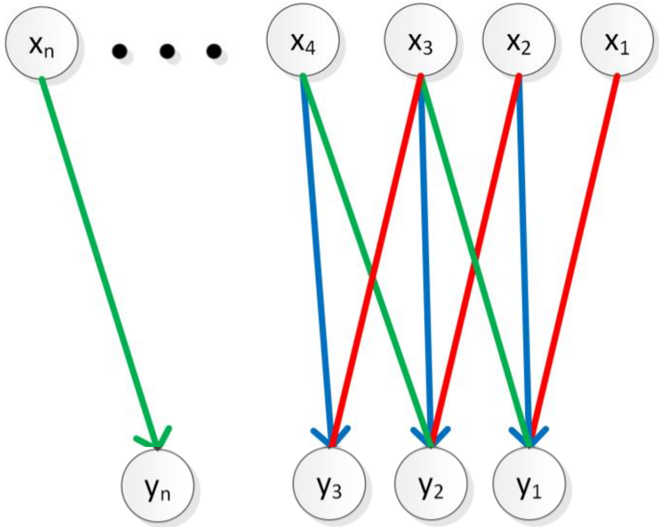
Output



(response map)

$$y = Wx$$

Input: all pixels



Output: kernel responses

Example with 1D kernel:

w_1	w_2	w_3

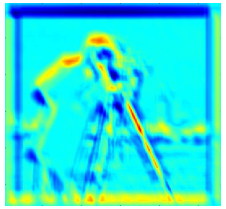
Convolutional Neural Layer

Modification 2: Same kernel applied to all sliding windows

Input

(image)

Output



(response map)

$y = Wx$

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

Example with 1D kernel:

w_1	w_2	w_3

- Can be implemented efficiently on GPUs
- *W* will be 3D: 3rd dimension allows for multiple kernels

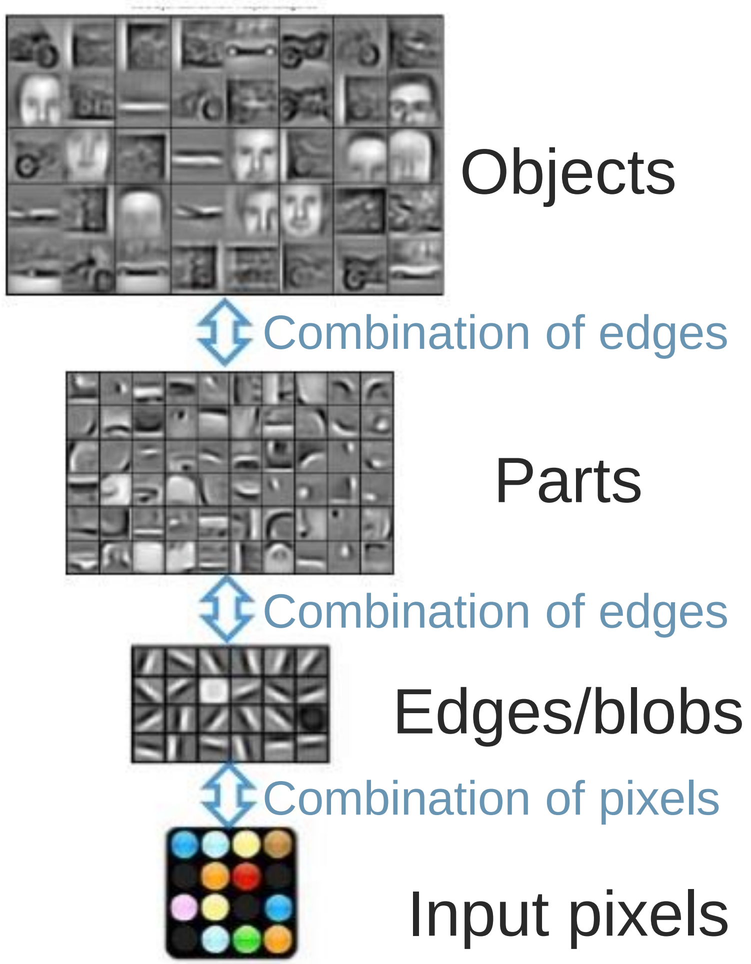
Convolutional Neural Network

Multiple convolutional layers

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

but how to encourage abstraction and summarization?

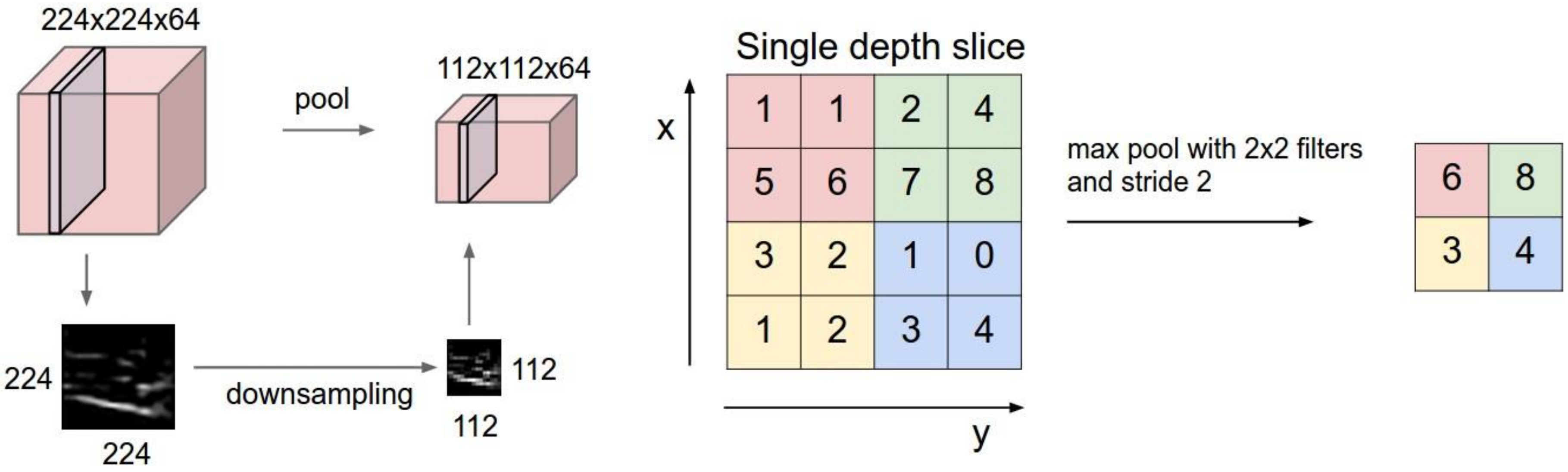
Answer: Pooling layers



Pooling Layer

Response map subsampling:

Allows summarization of the responses

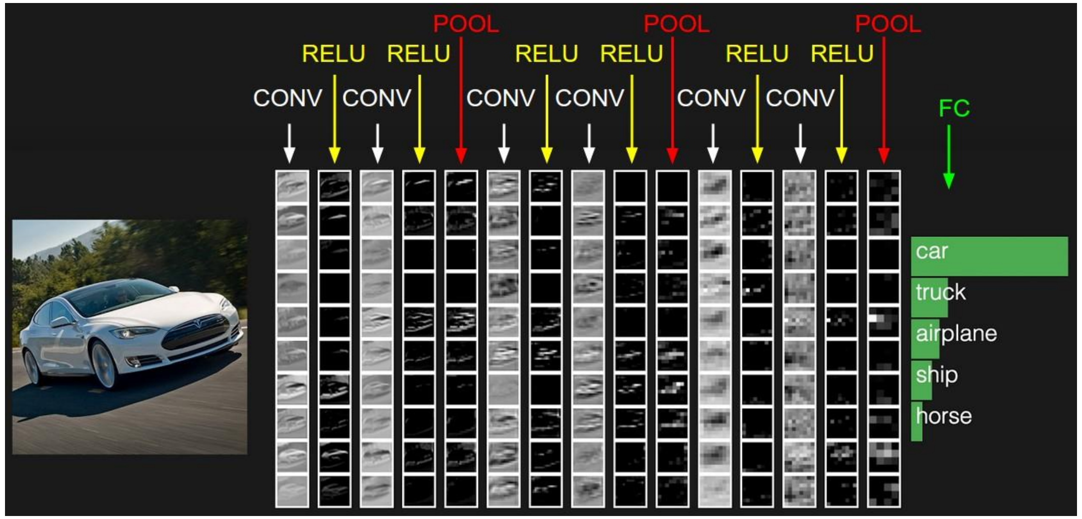


Common architectures

Repeat several times:

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

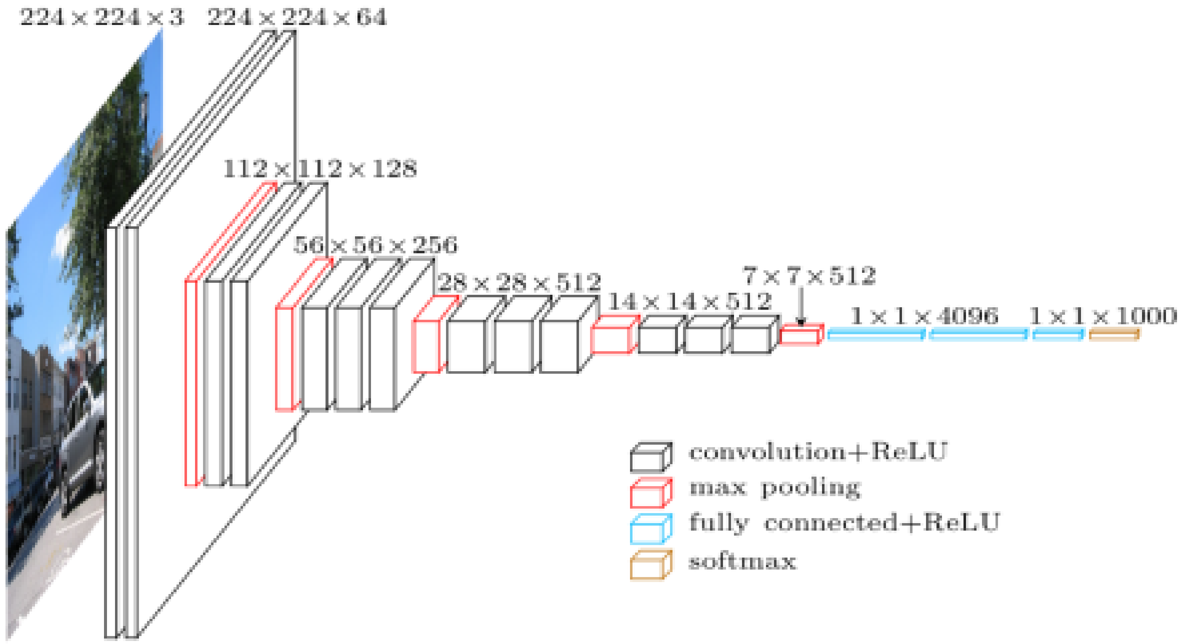
End with a fully connected (MLP) layer



Example: VGGNet model

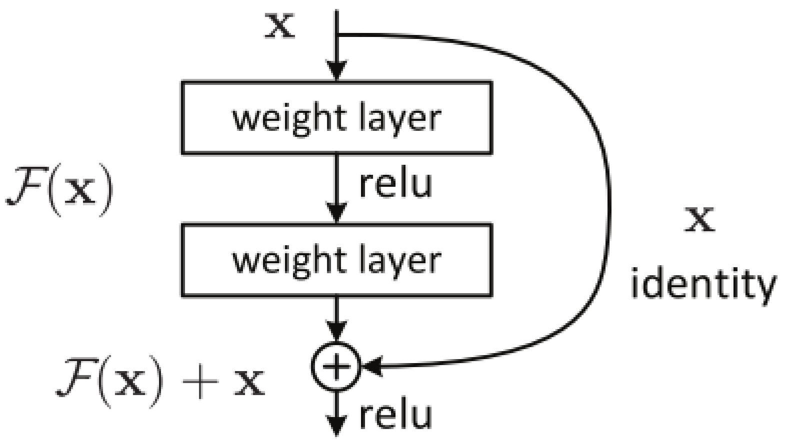
Used for object classification task

- 1000-way classification task
- 138 million parameters



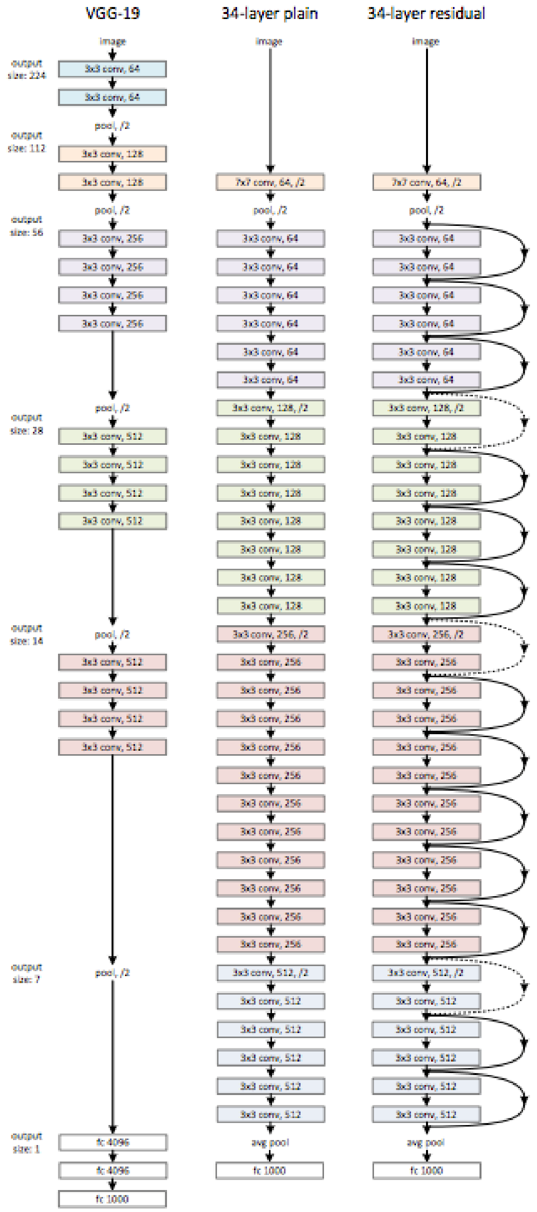
Residual Networks (ResNet)

Adding residual connections



ResNet (He et al., 2015)

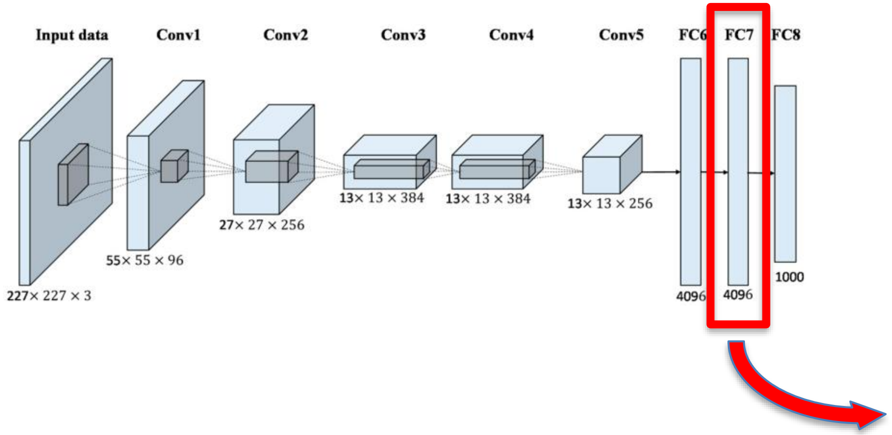
- Up to 152 layers!

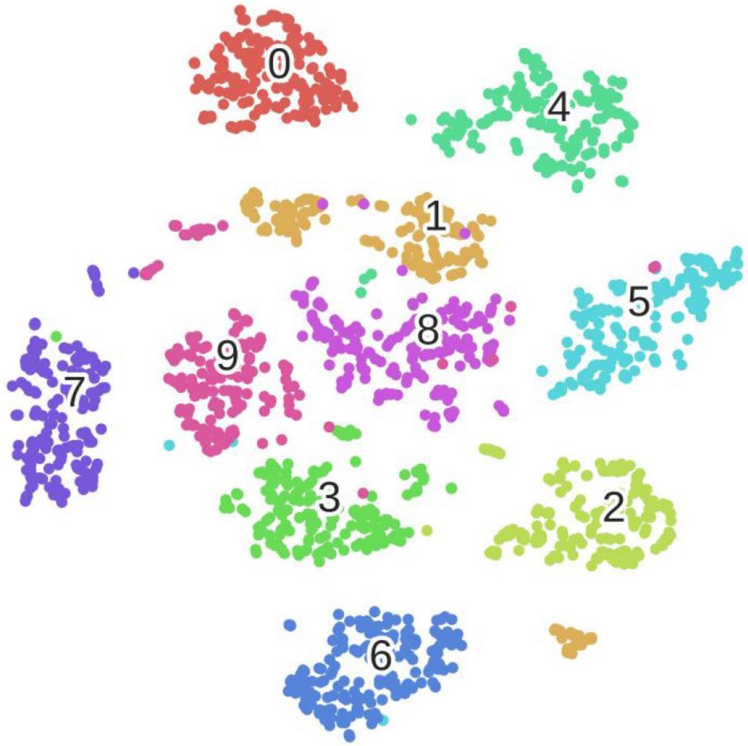


Visualizing CNNs

Visualizing the Last CNN Layer: t-sne

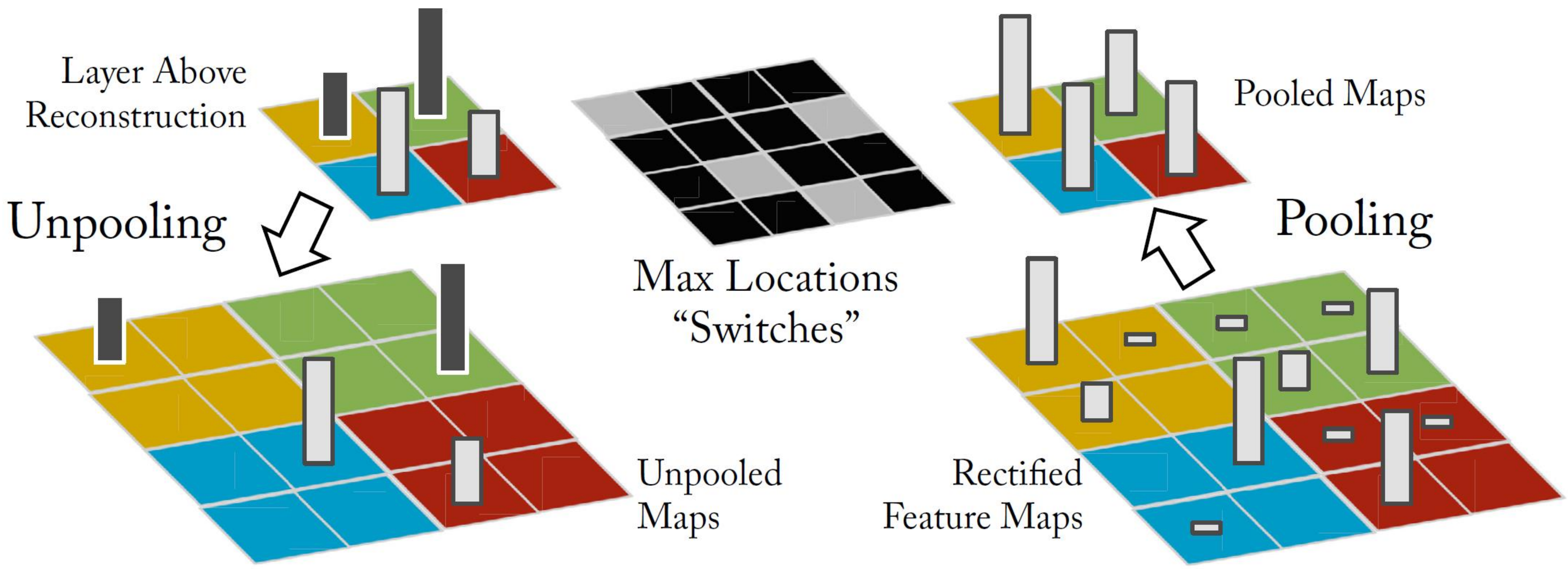
Alex Net



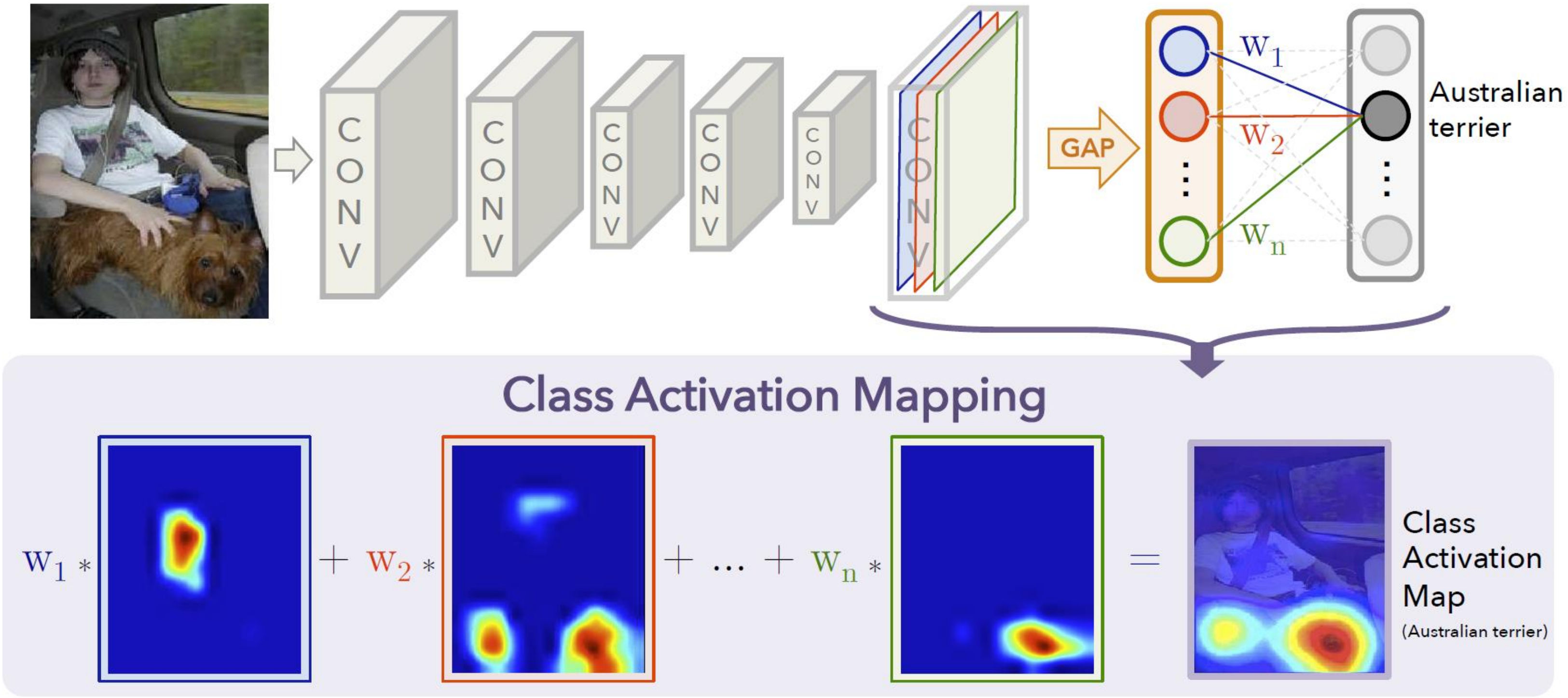


Embed high dimensional data points (i.e. feature codes) so that pairwise distances are conserved in local neighborhoods.

Deconvolution

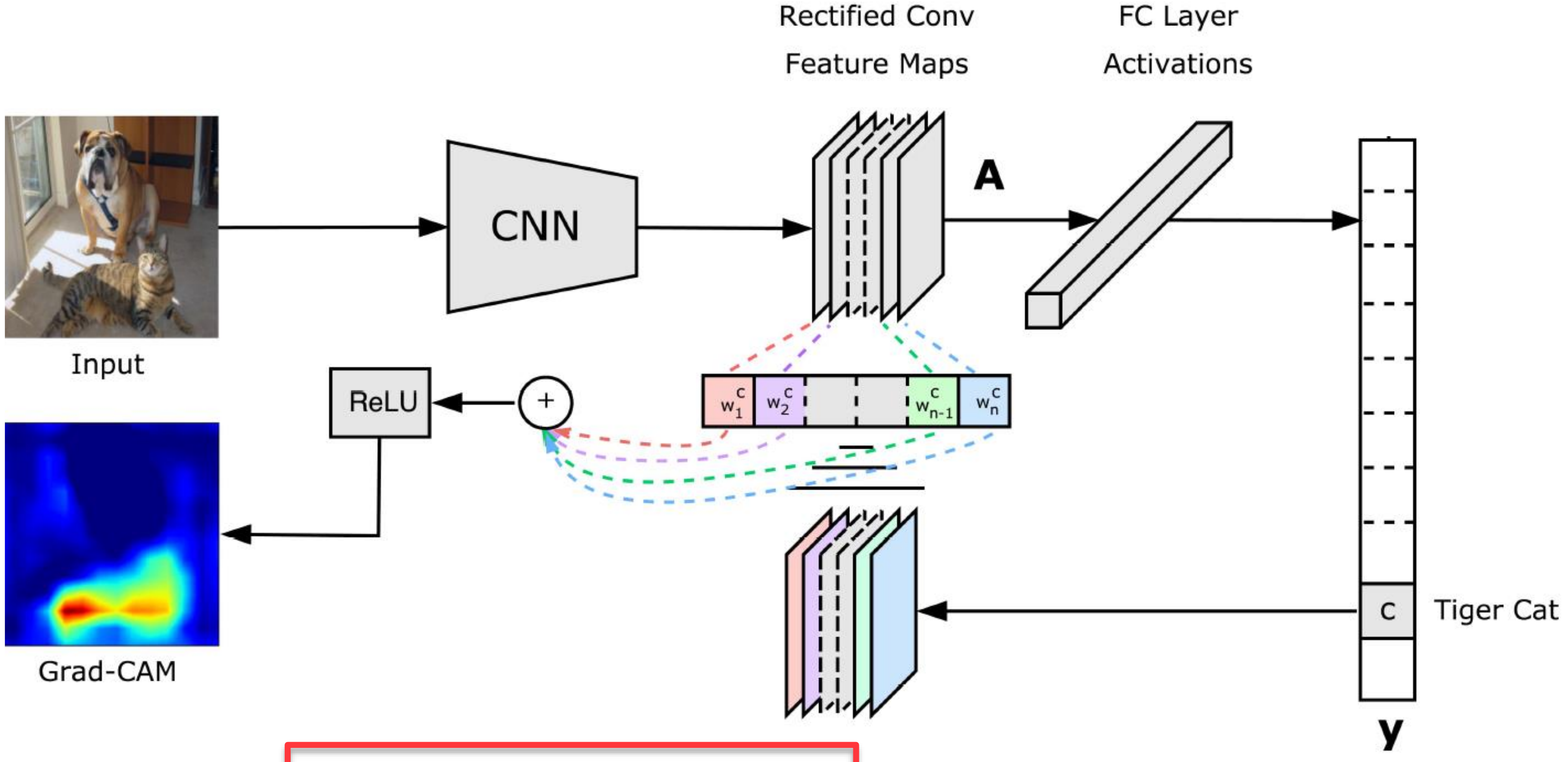


CAM: Class Activation Mapping [CVPR 2016]



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

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)

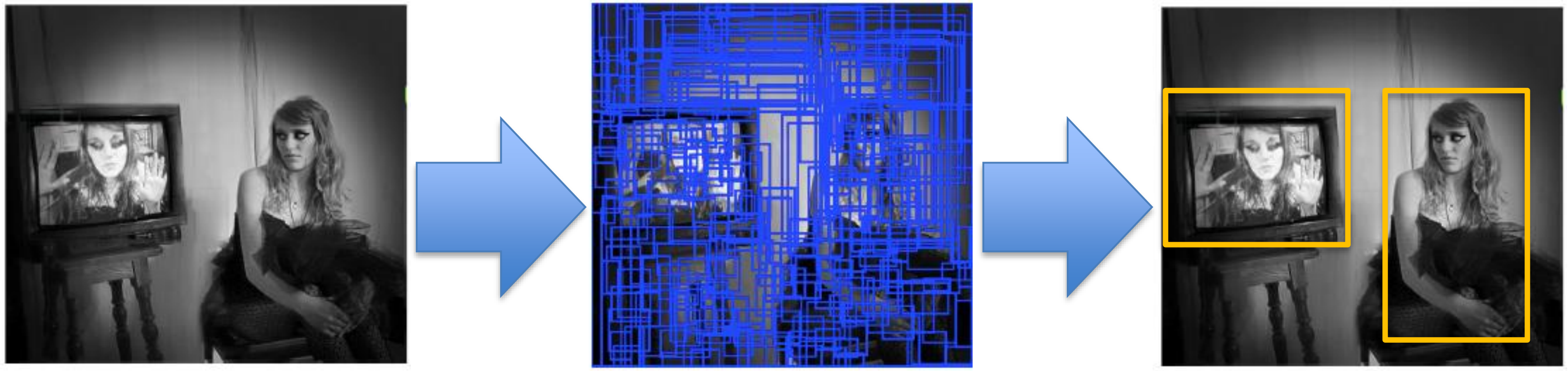
Input image

?

Detected Objects

One option: Sliding window

Object Detection (and Segmentation)



Input image | Region Proposals | Detected Objects

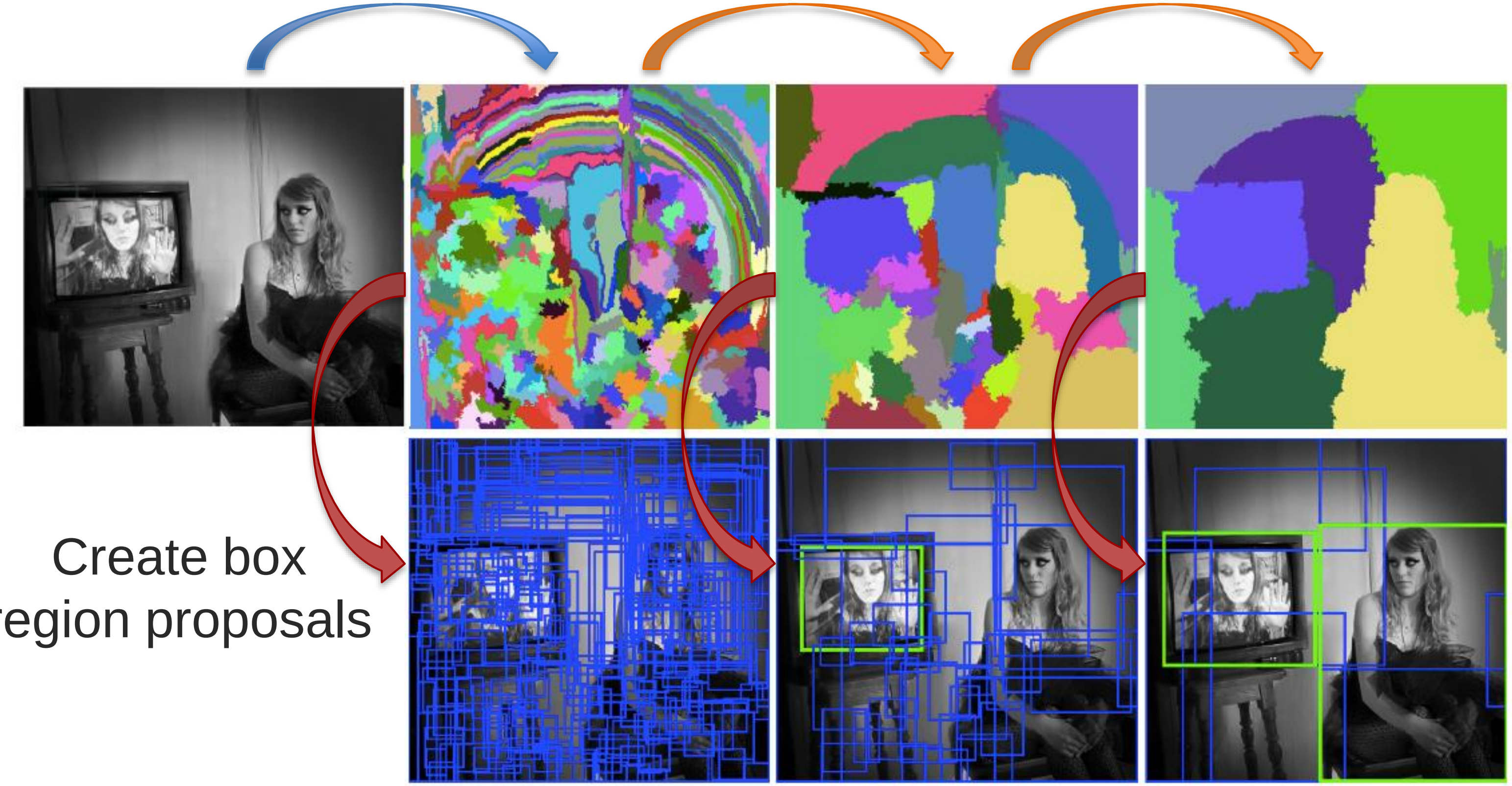
A better option: Start by Identifying hundreds of region proposals and then apply our CNN object detector

How to efficiently identify region proposals?

Selective Search [Uijlings et al., IJCV 2013]

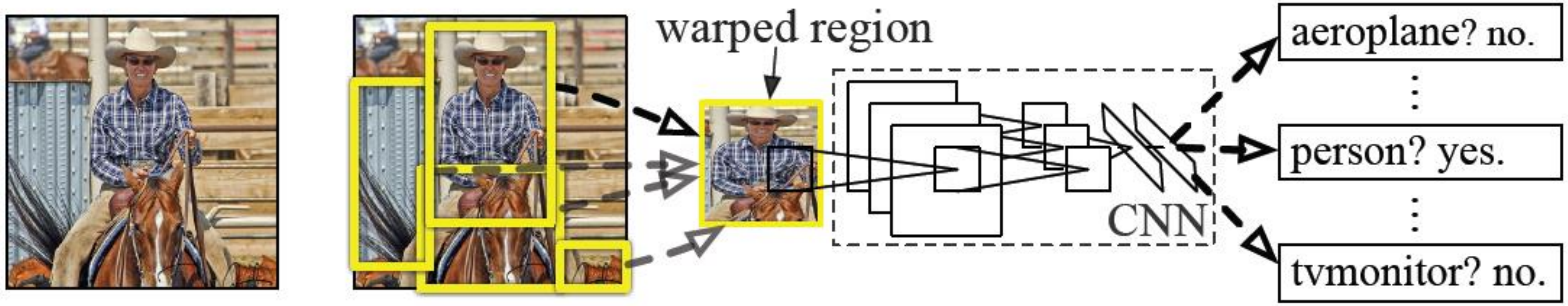
Image segmentation
(using *superpixels*)

And then merge
similar regions



Create box
region proposals

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



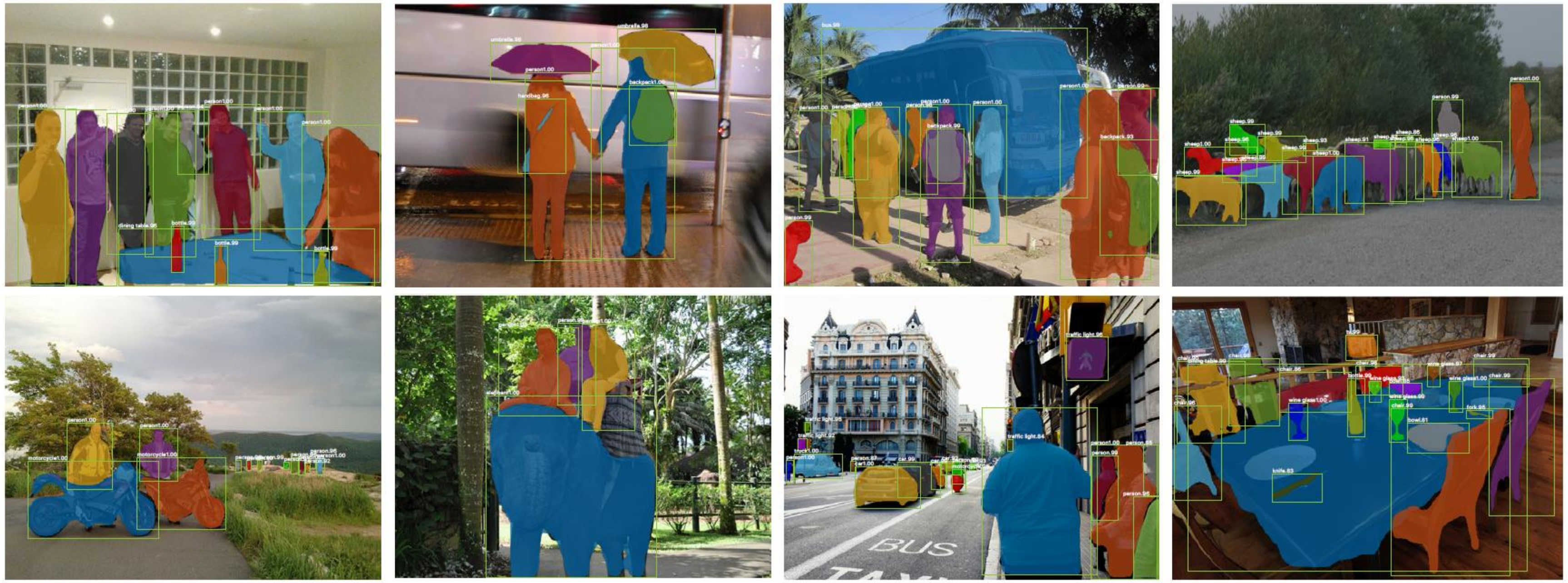
- Select ~2000 region proposals → Time consuming!
- Warp each region
- Apply CNN to each region → Time consuming!

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

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

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?

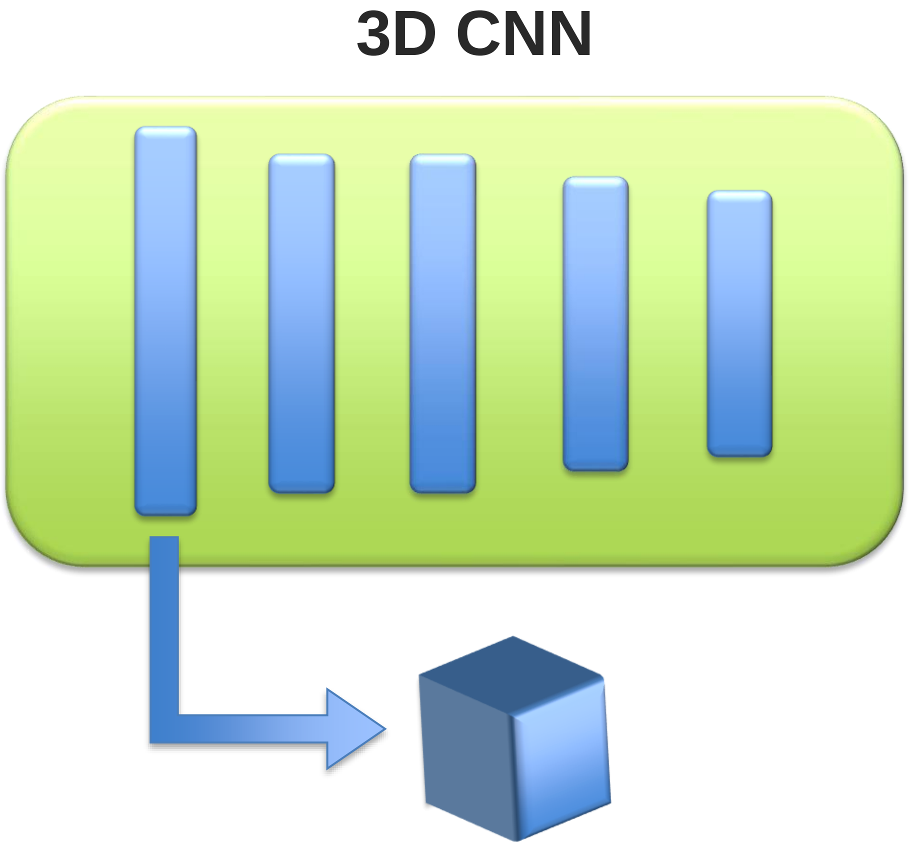
One option: Recurrent Neural Networks

(more about this next week)

3D CNN

Input as a 3D tensor

(stacking video images)



First layer with 3D kernels

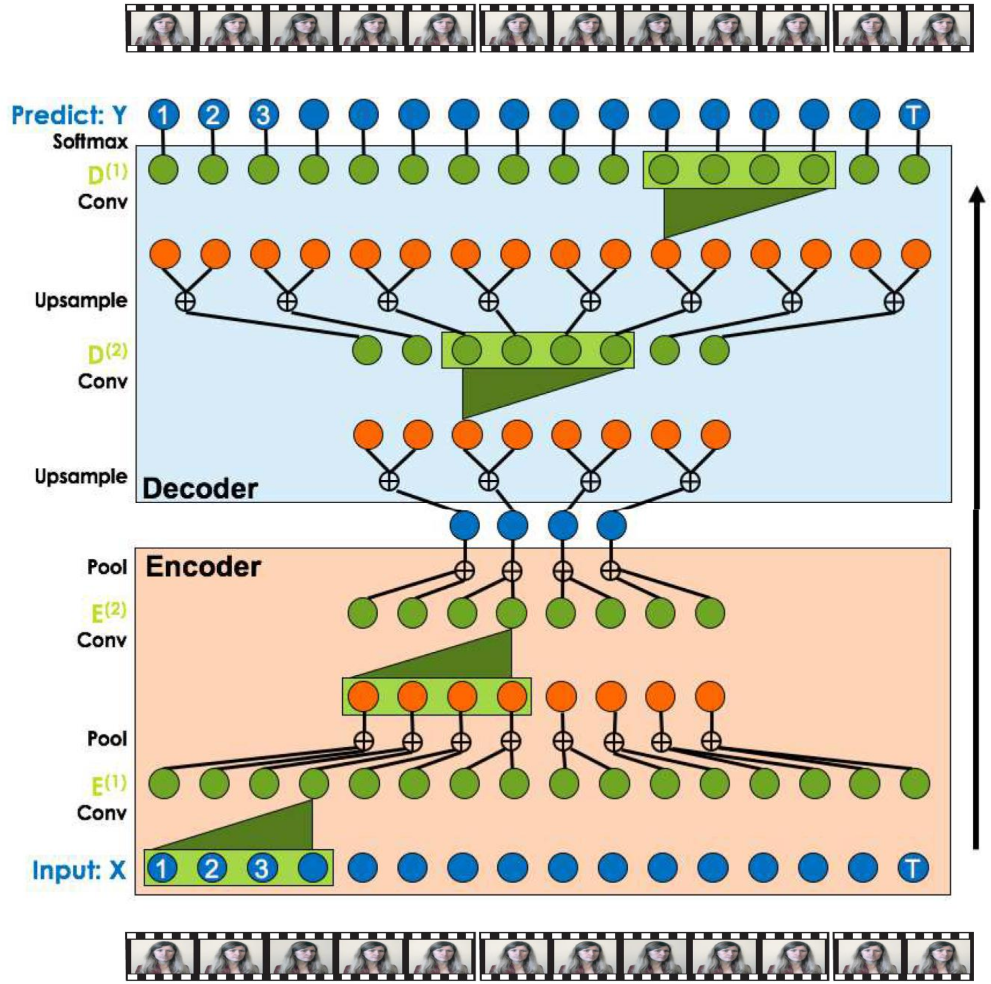
Time-Delay Neural Network

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



Team Matching Event

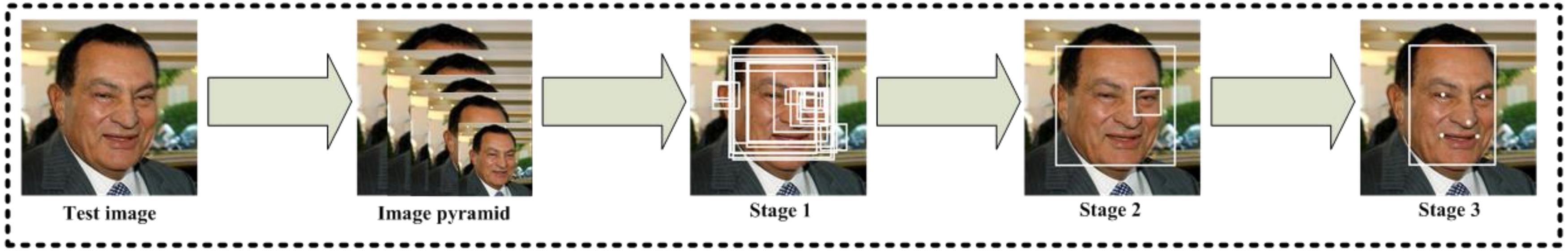
Appendix: Tools for Automatic visual behavior analysis

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

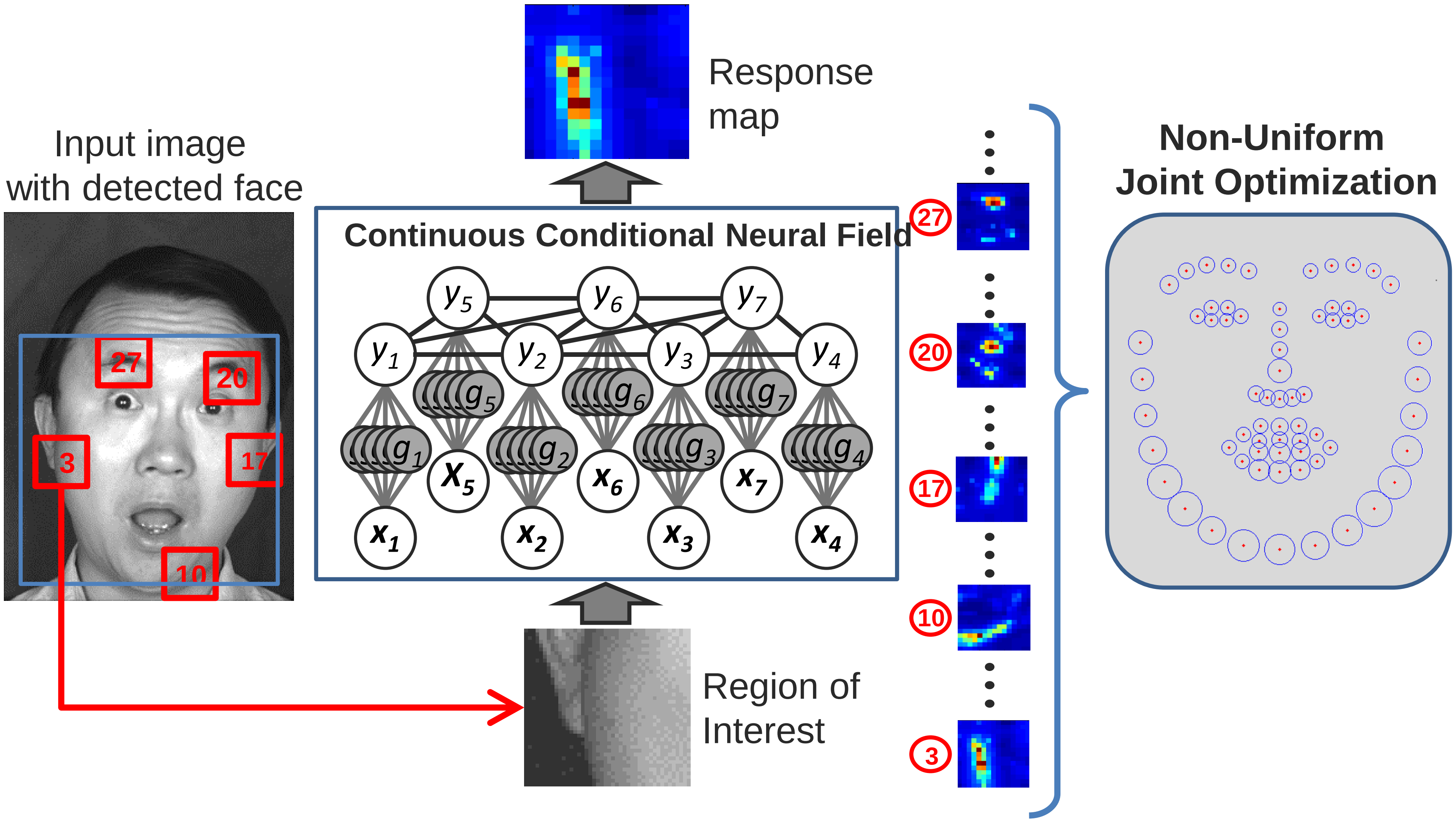
Stage 2: refine these candidates through a Refinement Network

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/

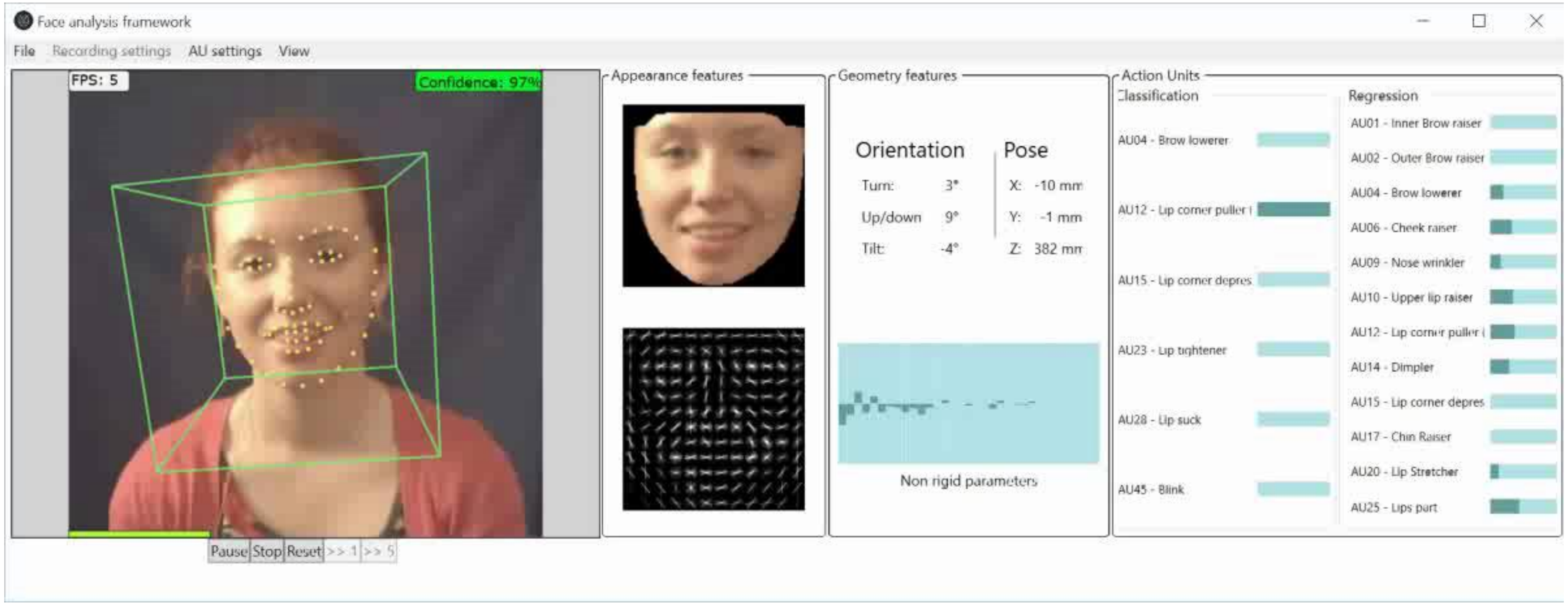
Facial Landmarks: Constrained Local Neural Field



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

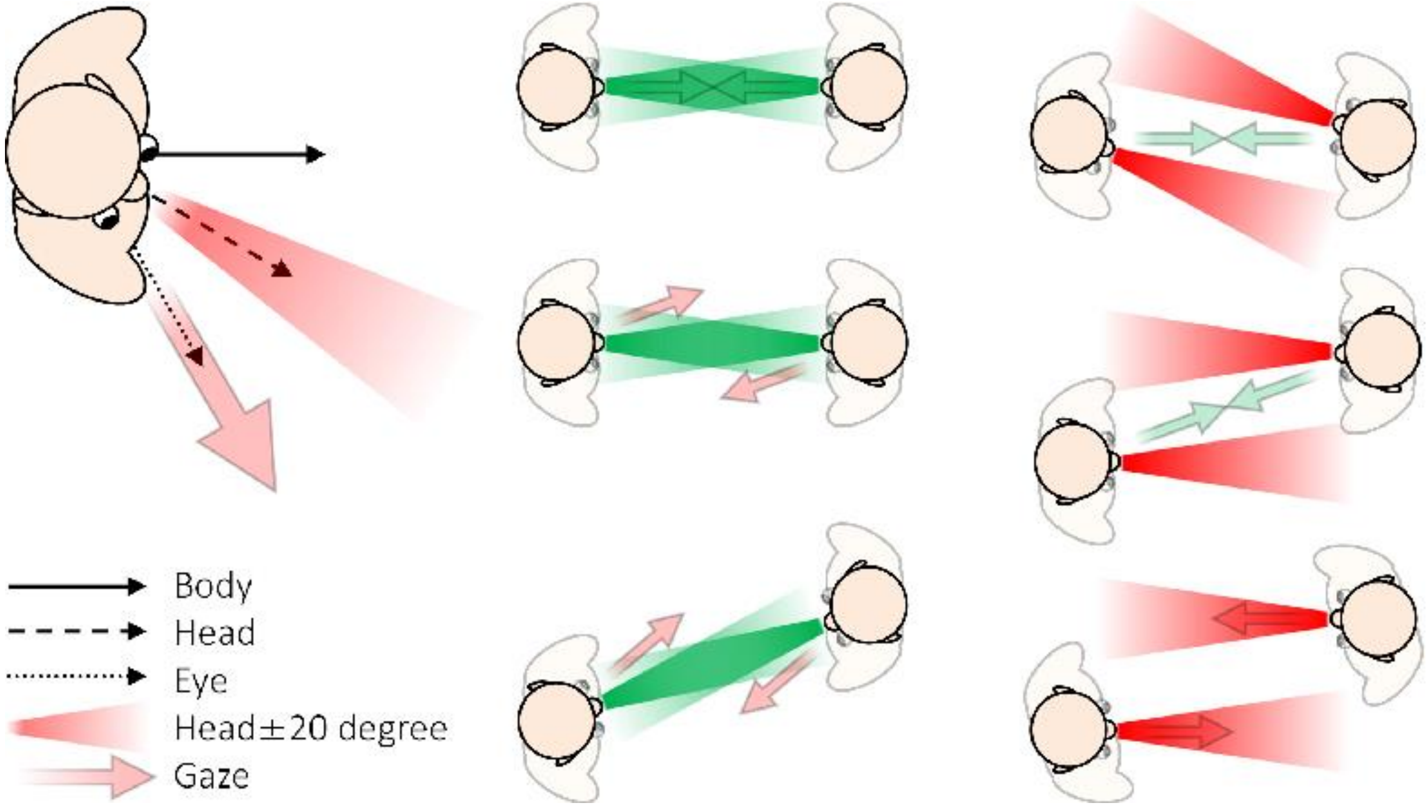
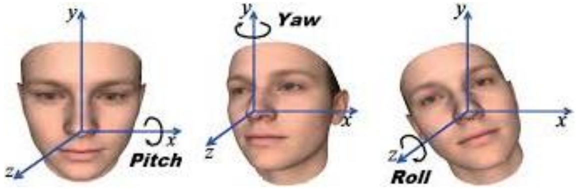


Image from Hachisu et al (2018). FaceLooks: A Smart Headband for Signaling Face-to-Face Behavior. *Sensors*.

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

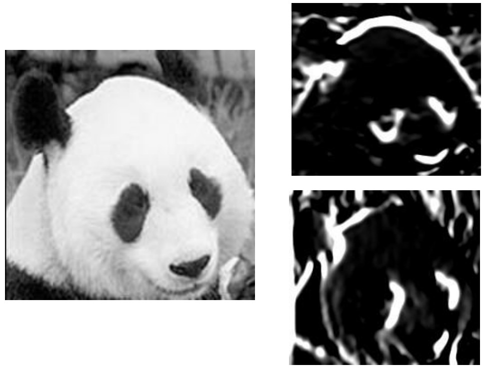
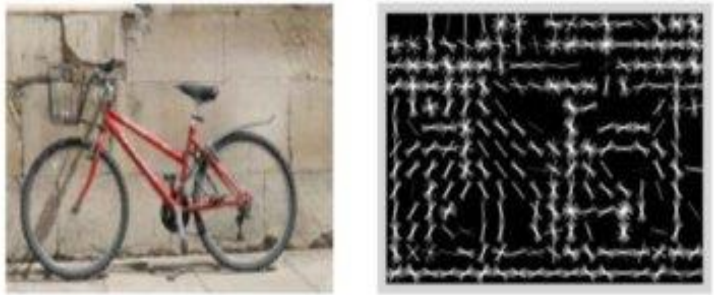


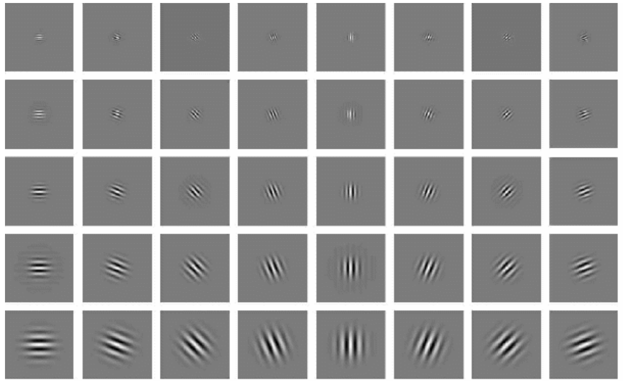
Image gradient

Edge detection

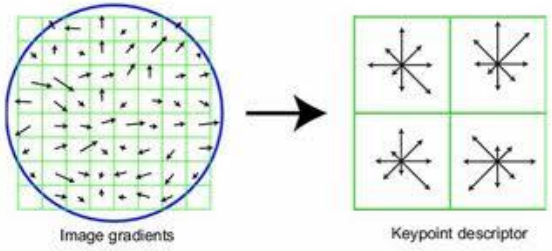


Histograms of Oriented Gradients

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