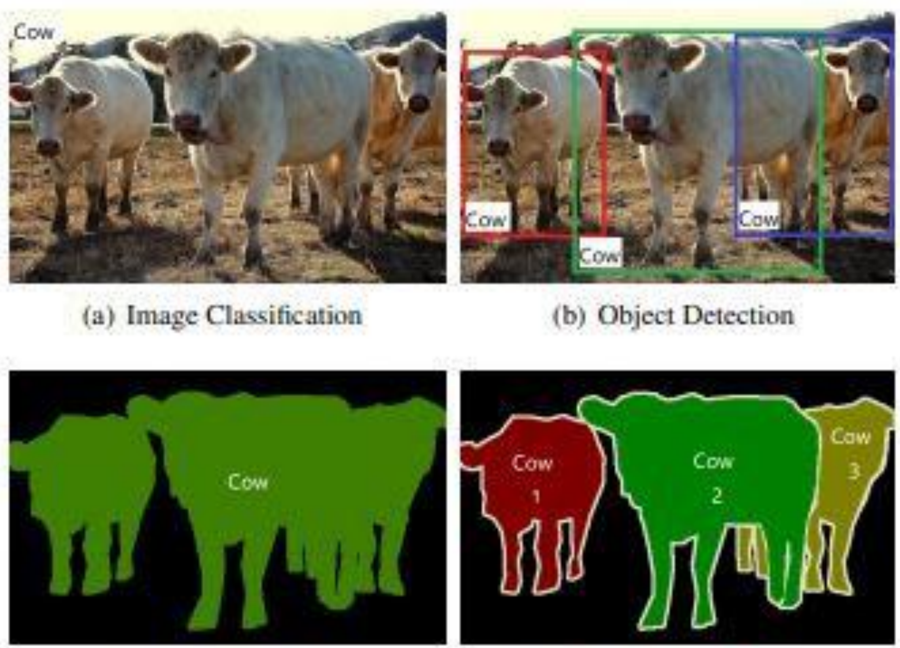


(a) Image Classification (b) Object Detection (c) Semantic Segmentation (d) Instance Segmentation

Neural Networks and object detection

M1 ARIA
Image and Video Processing

Gabriele Facciolo

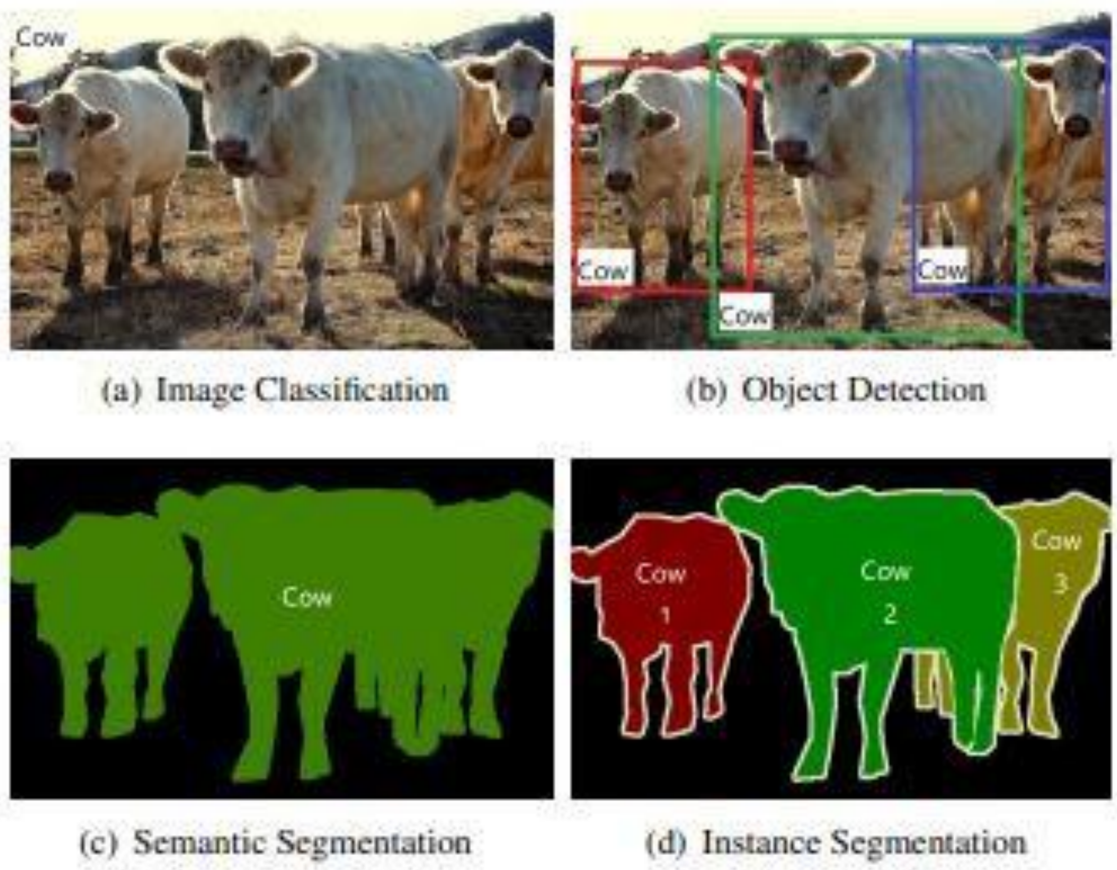
Lecture 3 - 23-09-2024

Plan

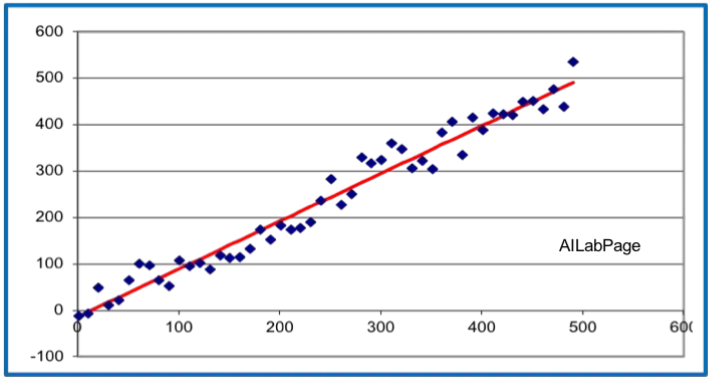
- Last week recap
- Object detection
- Instance detection
- Example of an advanced computer vision task
    - Human pose estimation

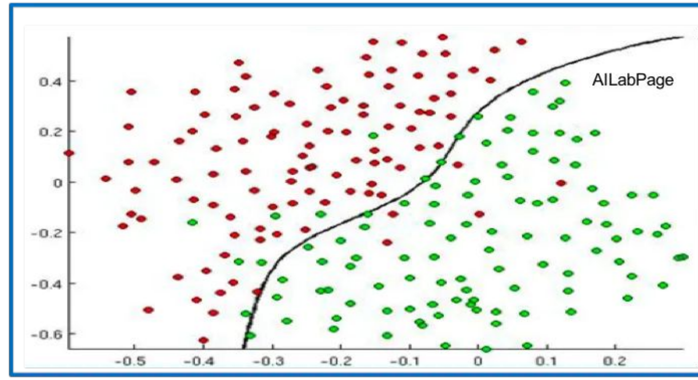
Last week recap

Image classification



Reminder: classification vs regression





Regression

1. The system attempts to predict a value for an input based on past data.
2. Real number / Continuous numbers – Regression problem
3. Example – 1. Temperature for tomorrow

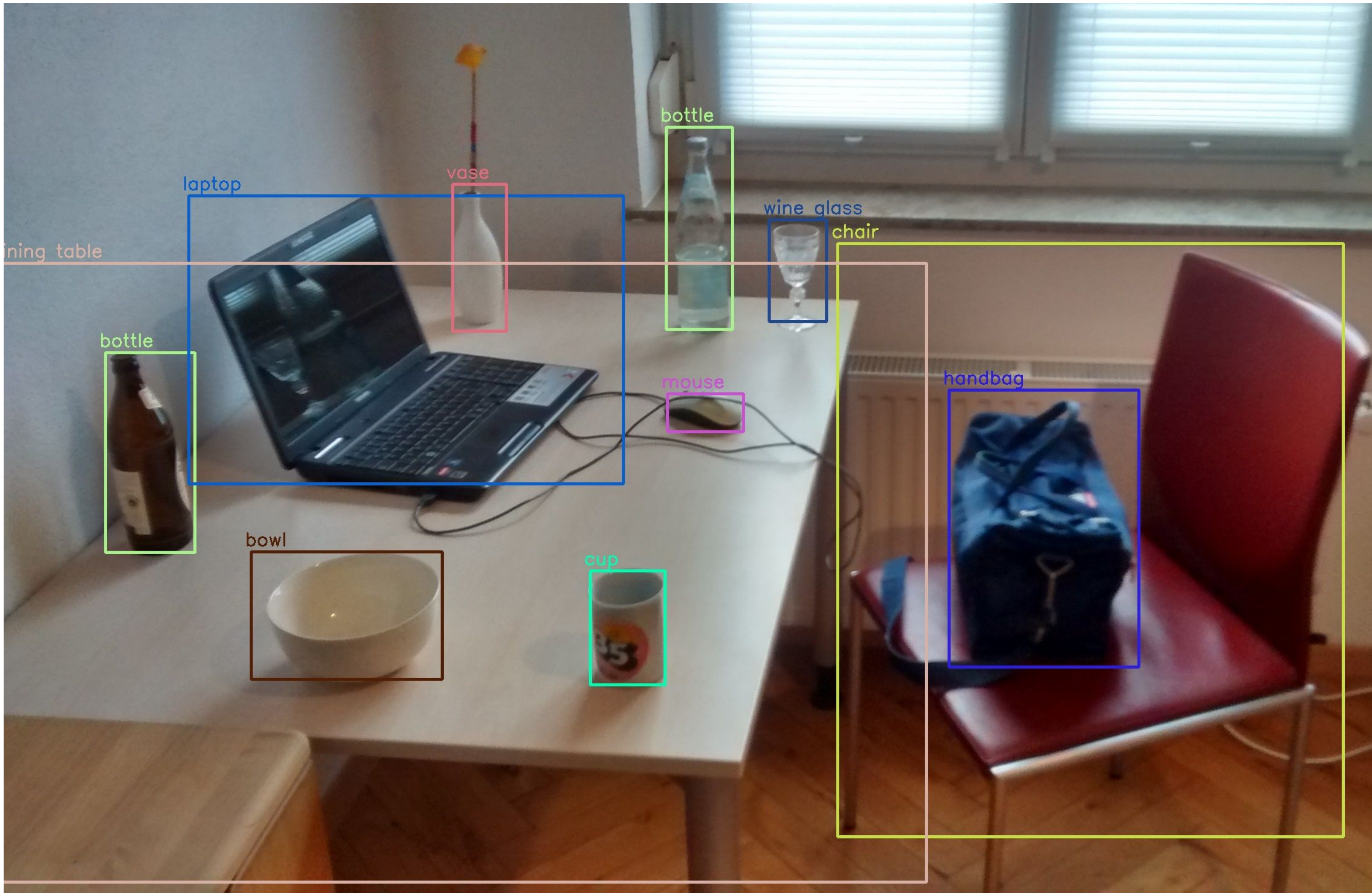
Classification

1. In classification, predictions are made by classifying them into different categories.
2. Discreate / categorical variable – Classification problem
3. Example – 1. Type of cancer 2. Cancer Y/N

AILabPage

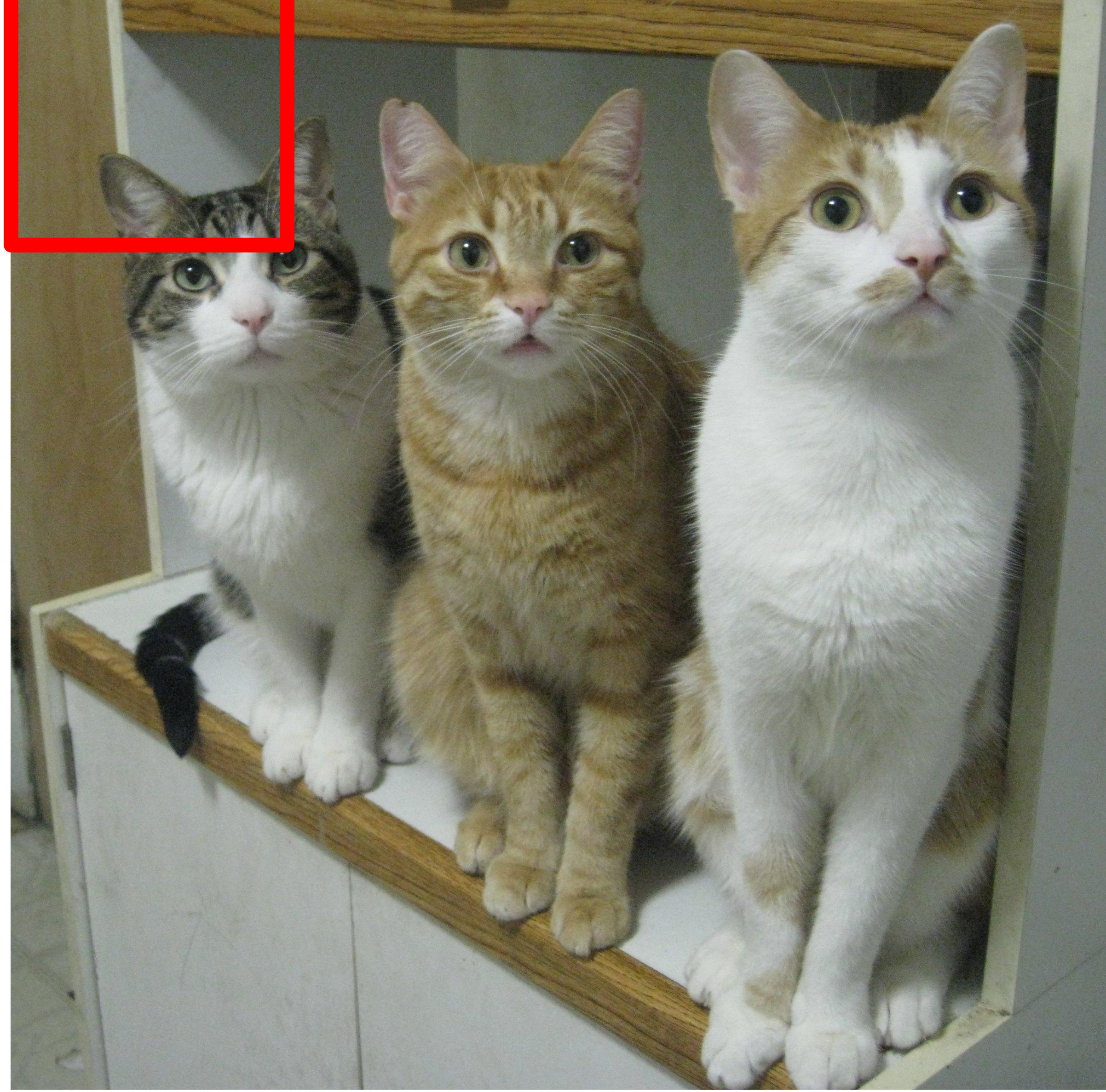
Object detection

Object detection

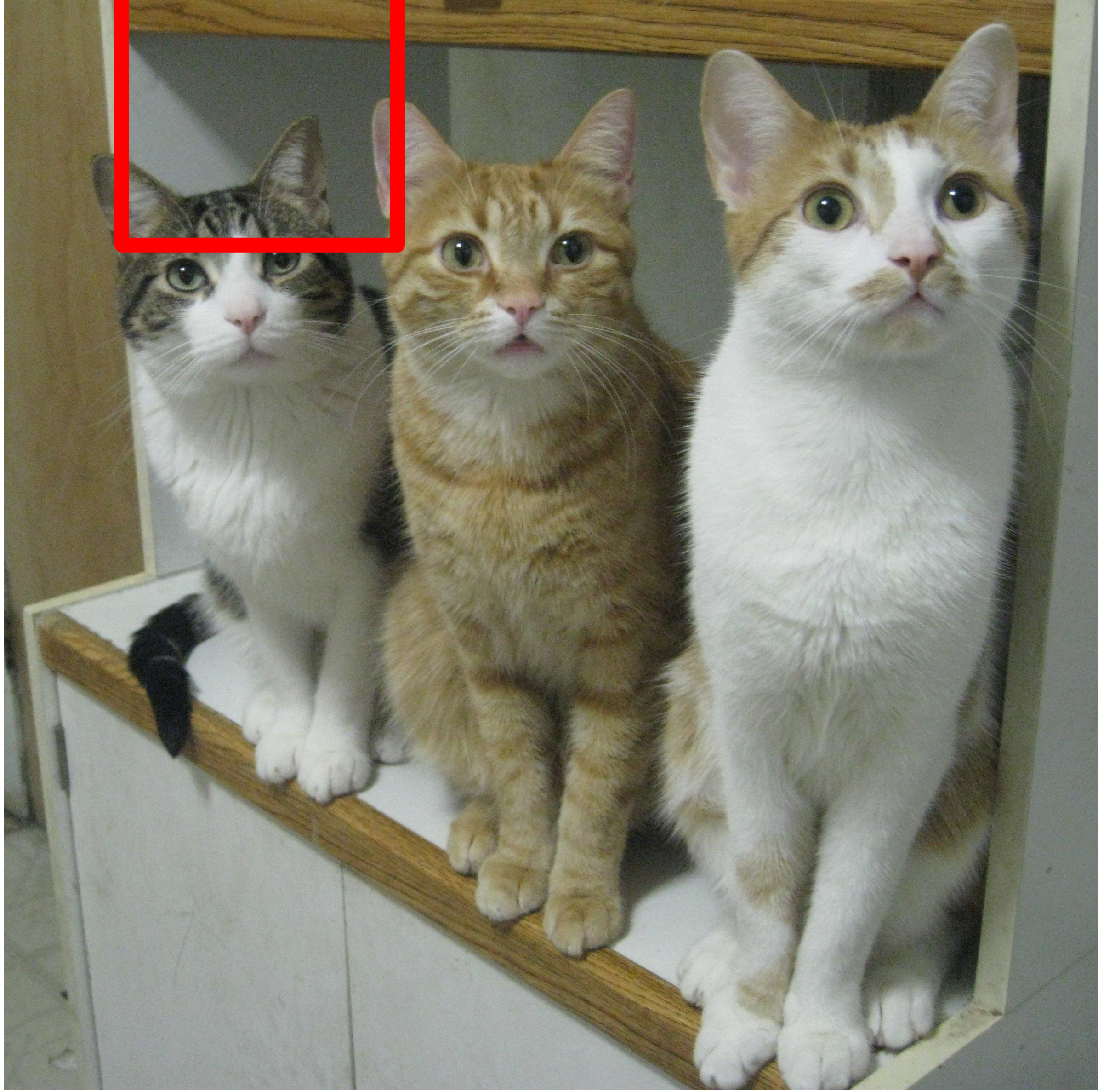


Let’s detect cat heads

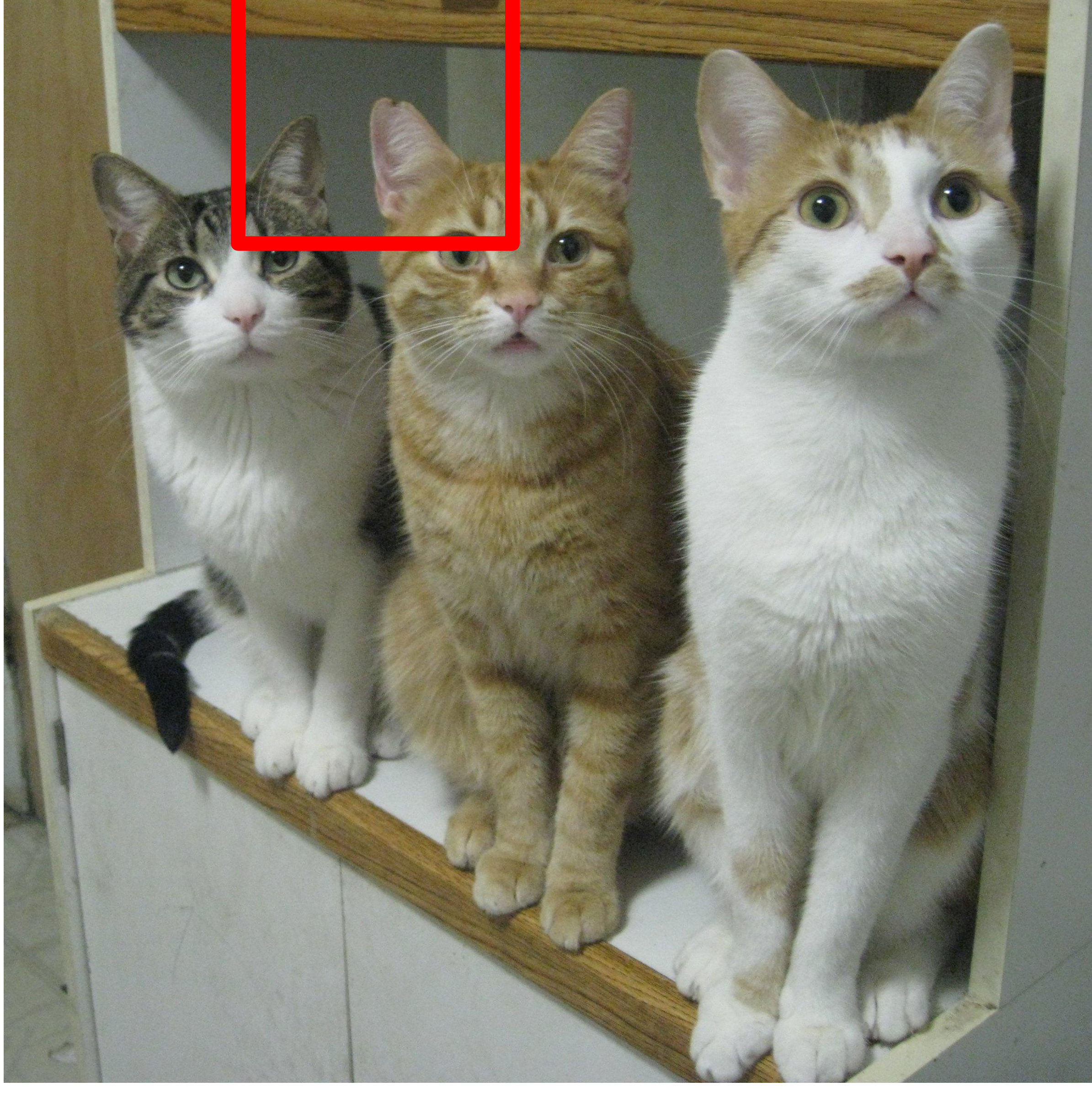
Let’s detect cat heads



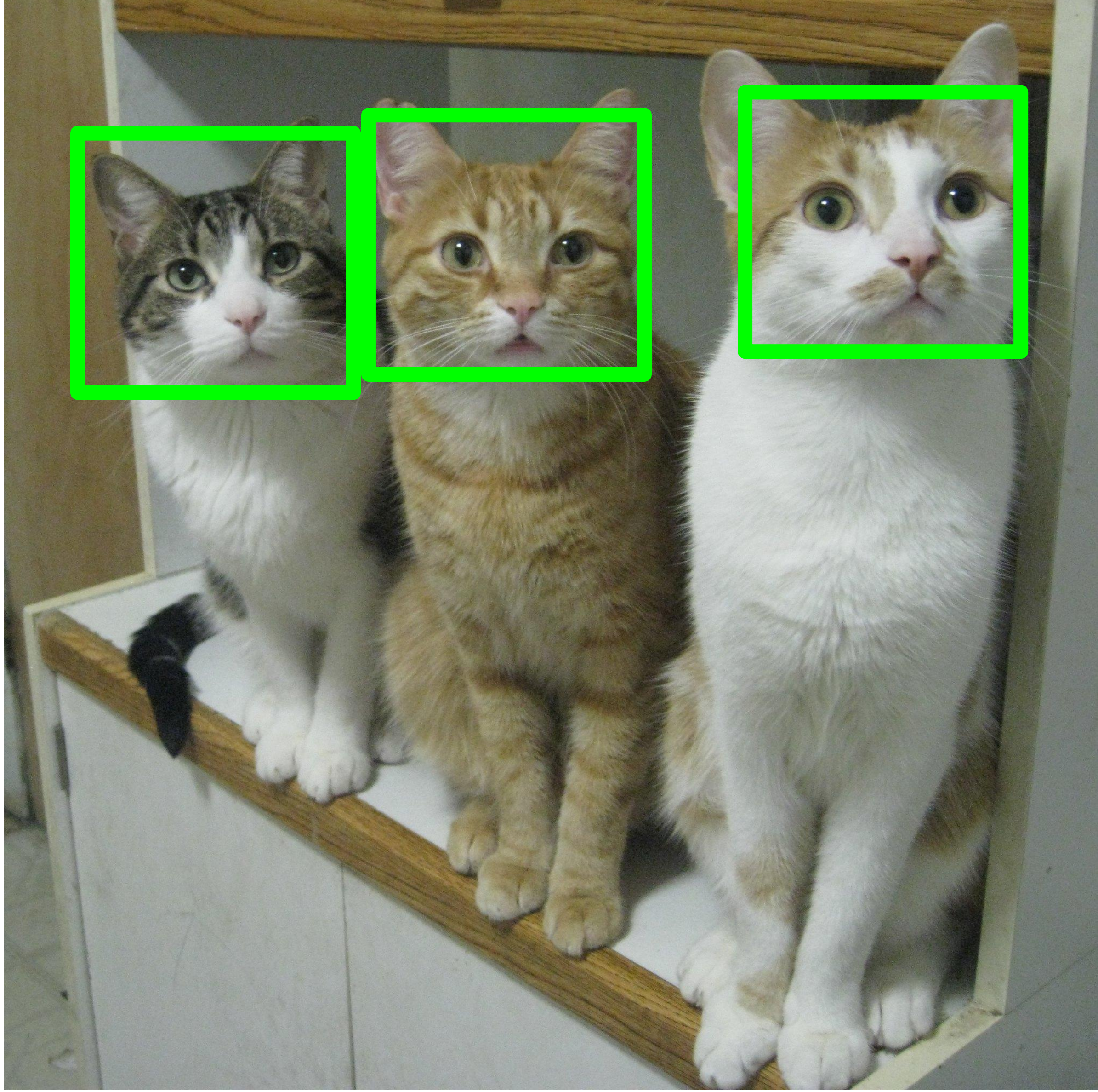
Let’s detect cat heads



Let's detect cat heads



Let’s detect cat heads



Object detection

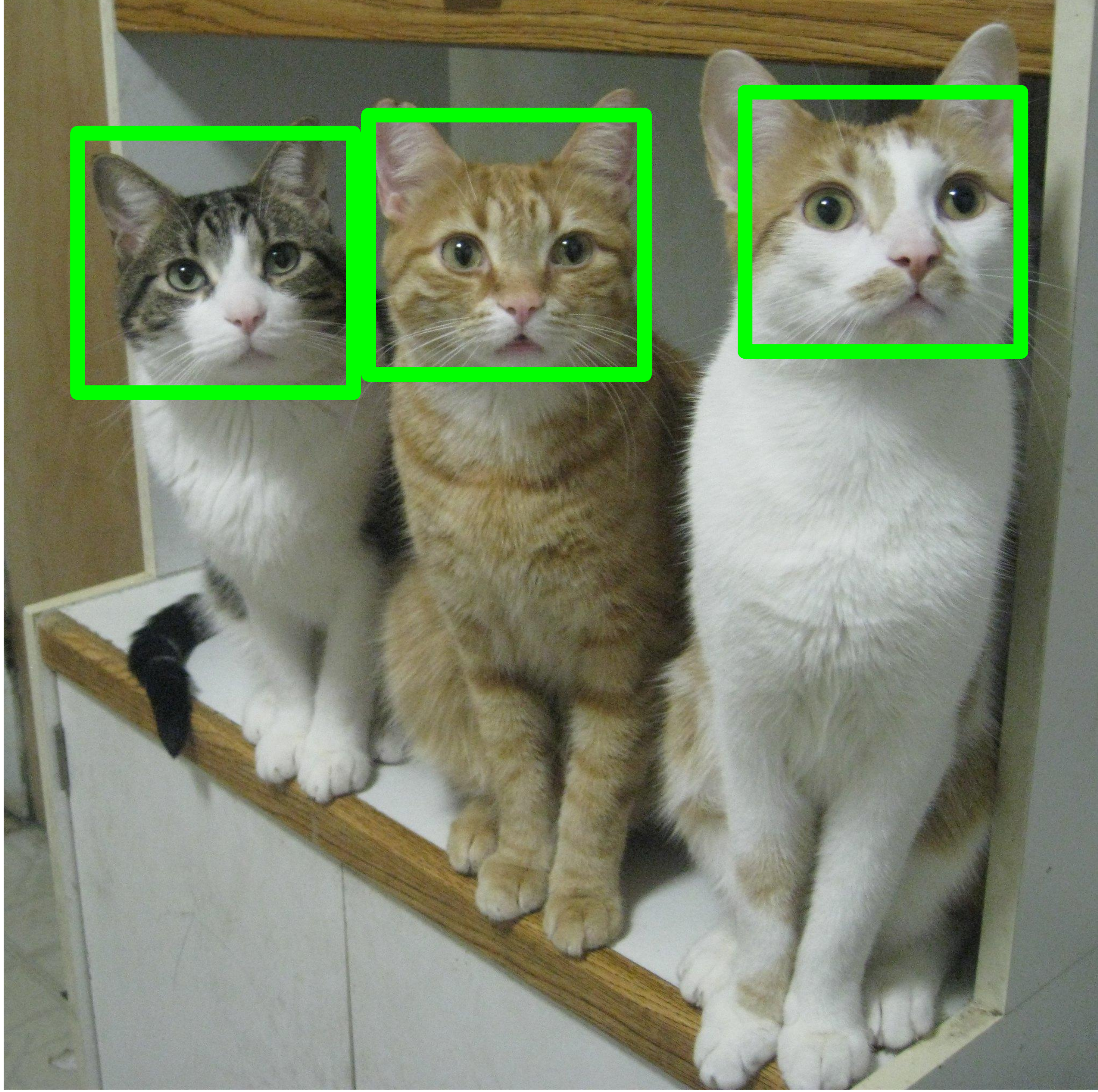
→ Object detection can again be seen as a classification problem.

Several issues will appear.

Let’s detect cat heads.

First, more than one detection per cat is likely.

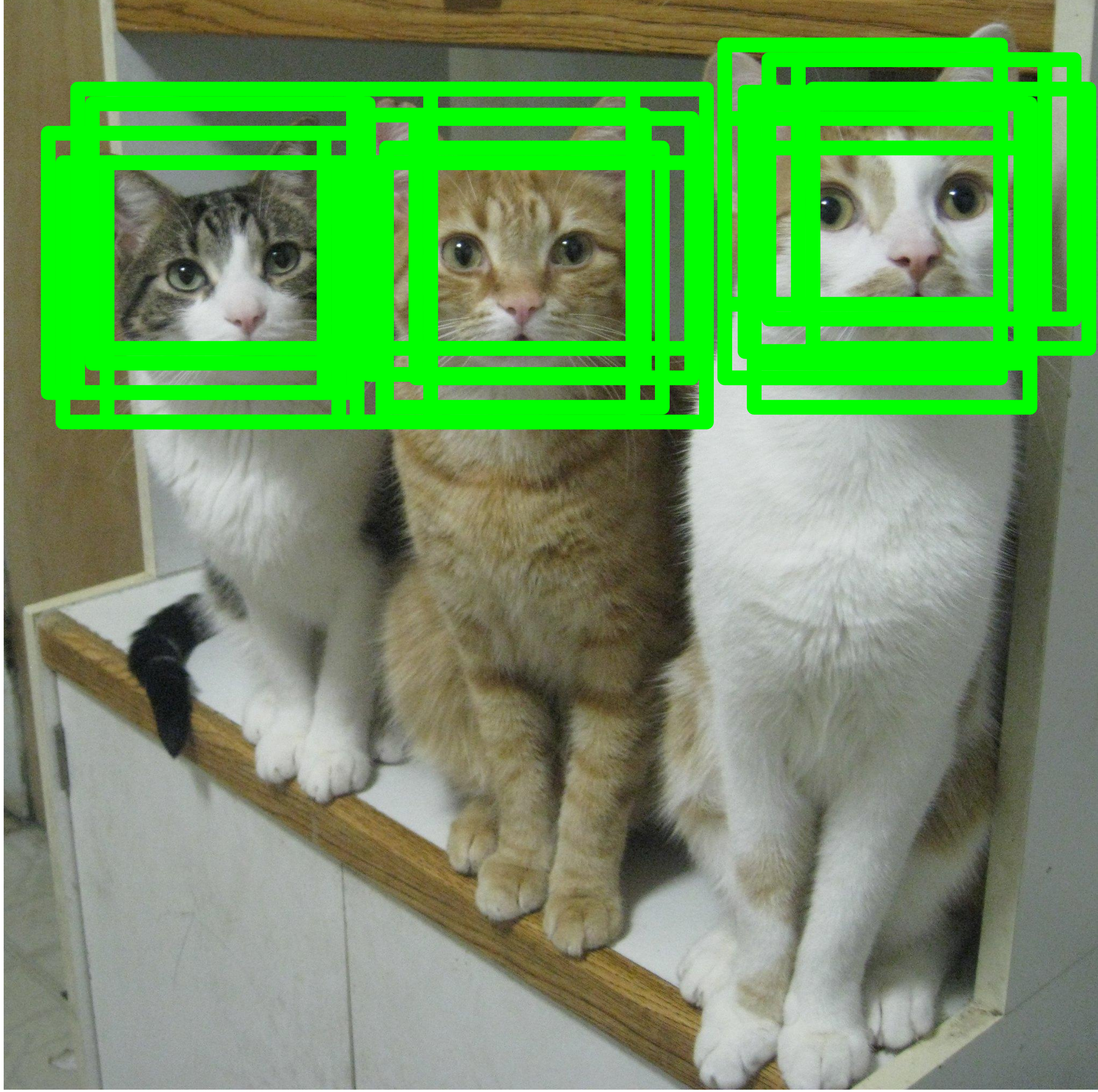
Dream →



Let’s detect cat heads.

First, more than one detection per cat is likely.

Reality →



Object detection

Solution: Non-Maximum Suppression (NMS)

Reminder: a classification is represented by a vector of size N if there are N classes.

Cat head	Not cat head
0.8	0.2

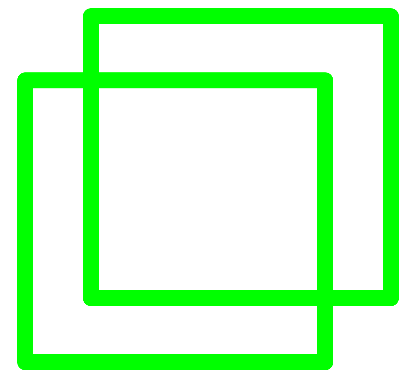
Object detection

Solution: Non-Maximum Suppression (NMS)

Reminder: a classification is represented by a vector of size N if there are N classes.

1. Compute Intersection Over Union (IOU)
   Measures how much the two BBs intersect.

2. If IOU > 0.5 → keep BB with highest confidence level



Object detection

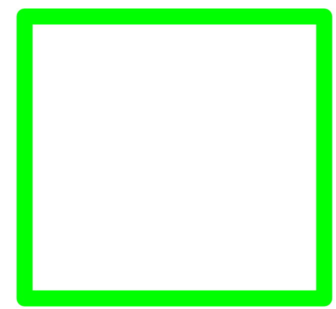
Solution: Non-Maximum Suppression (NMS)

Reminder: a classification is represented by a vector of size N if there are N classes.

1. Compute Intersection Over Union (IOU)
   Measures how much the two BBs intersect

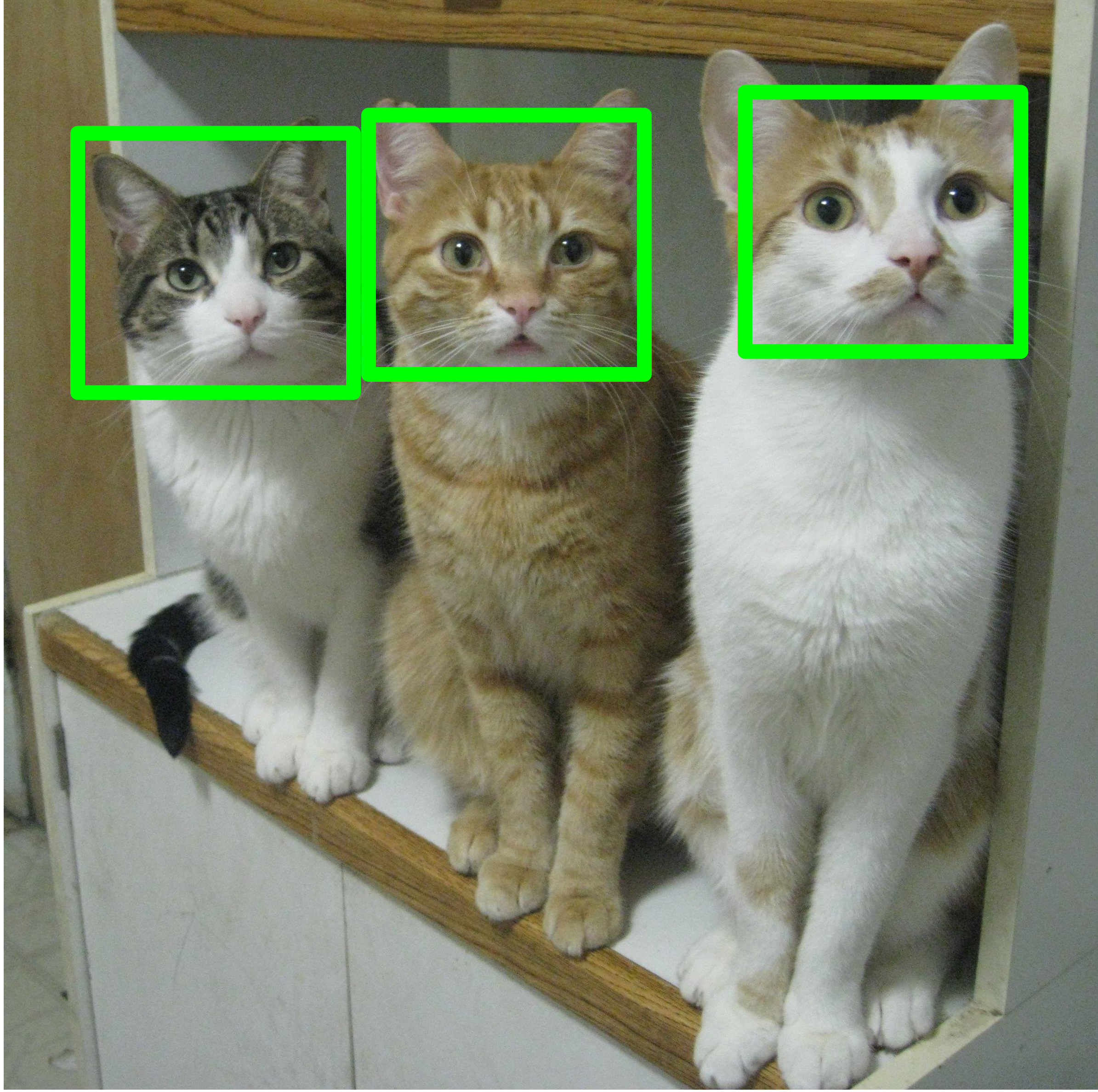
2. If IOU > 0.5 → keep BB with highest confidence level

Cat
Confidence level: 0.9



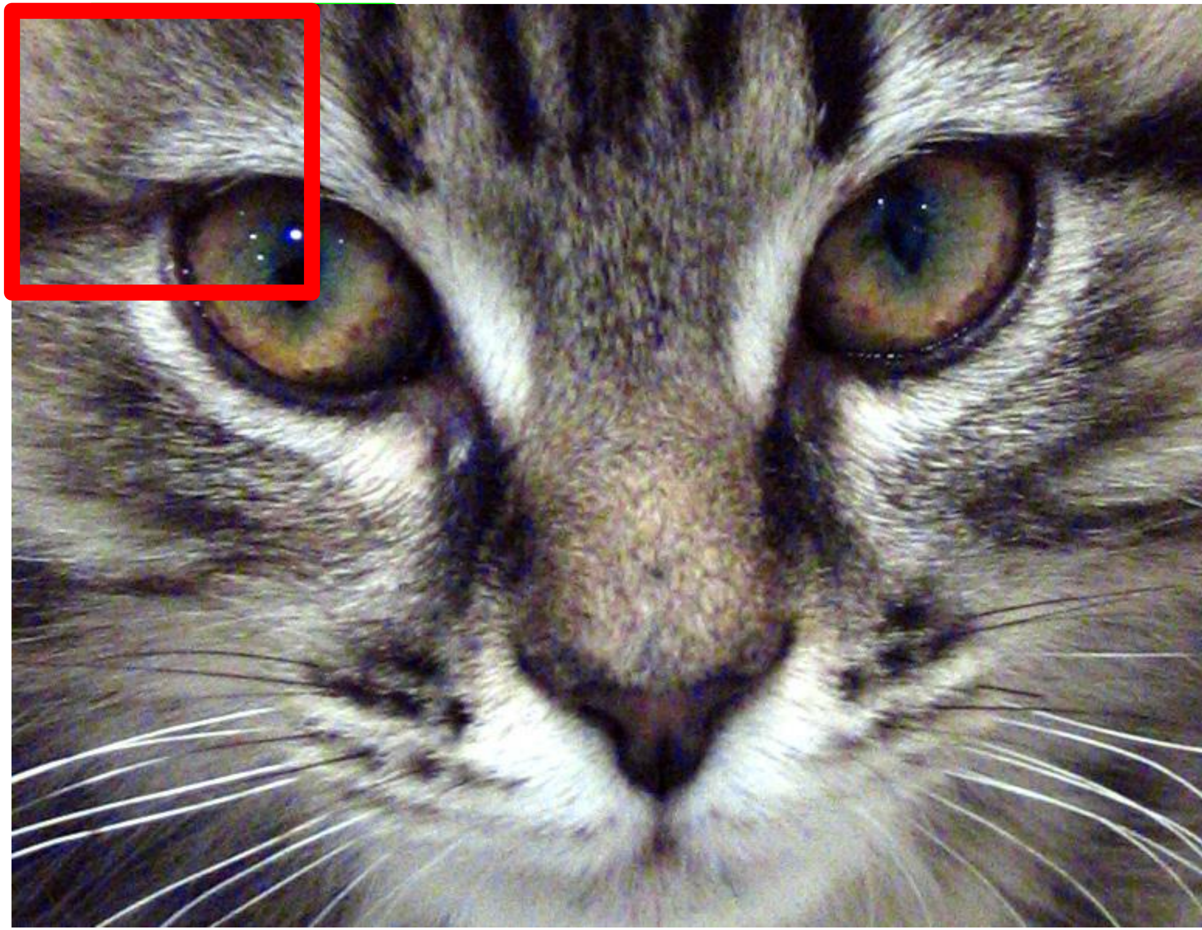
Let's detect cat heads.

Window size here is well adapted.



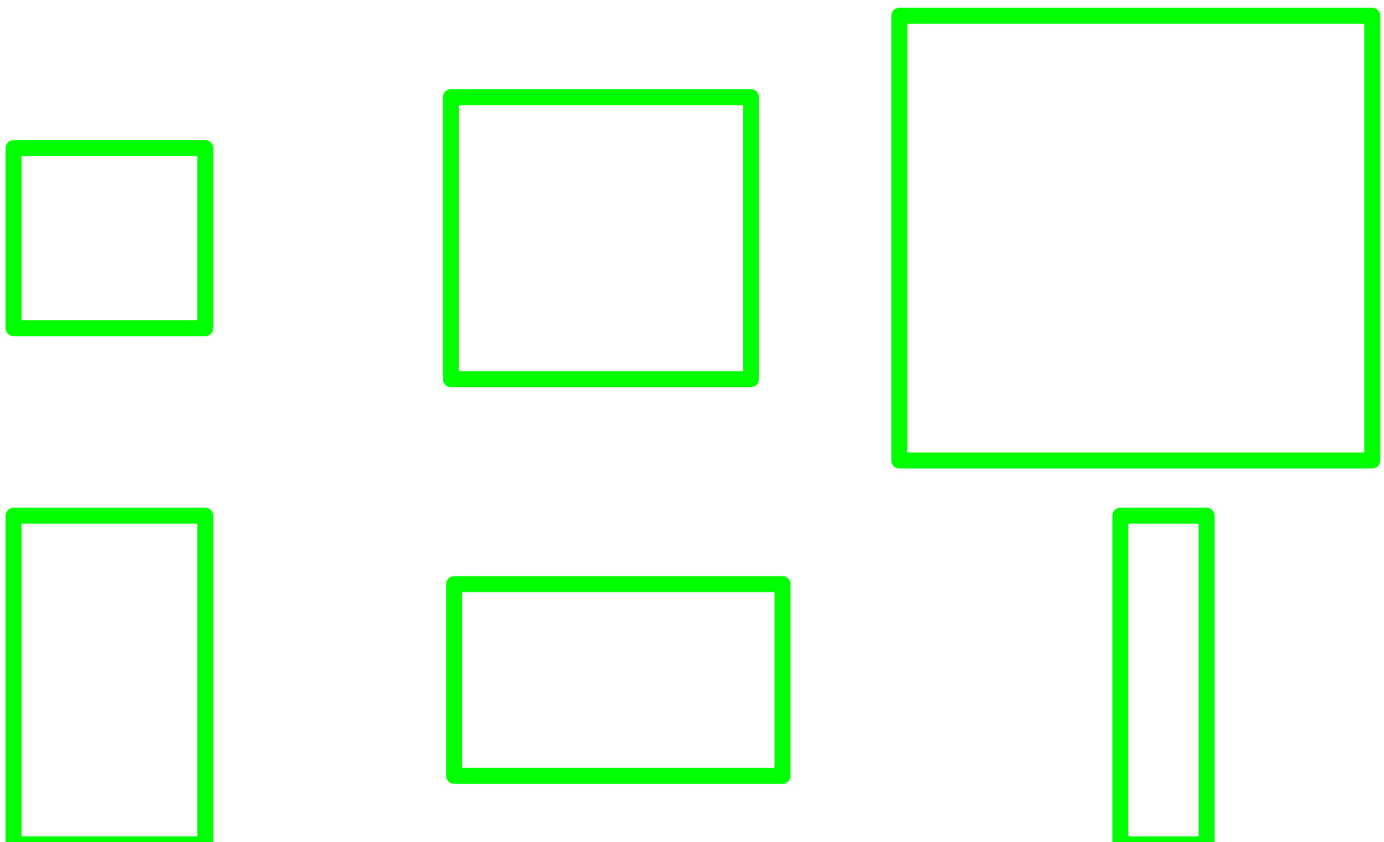
Let’s detect cat heads.

Window size here is not well adapted.

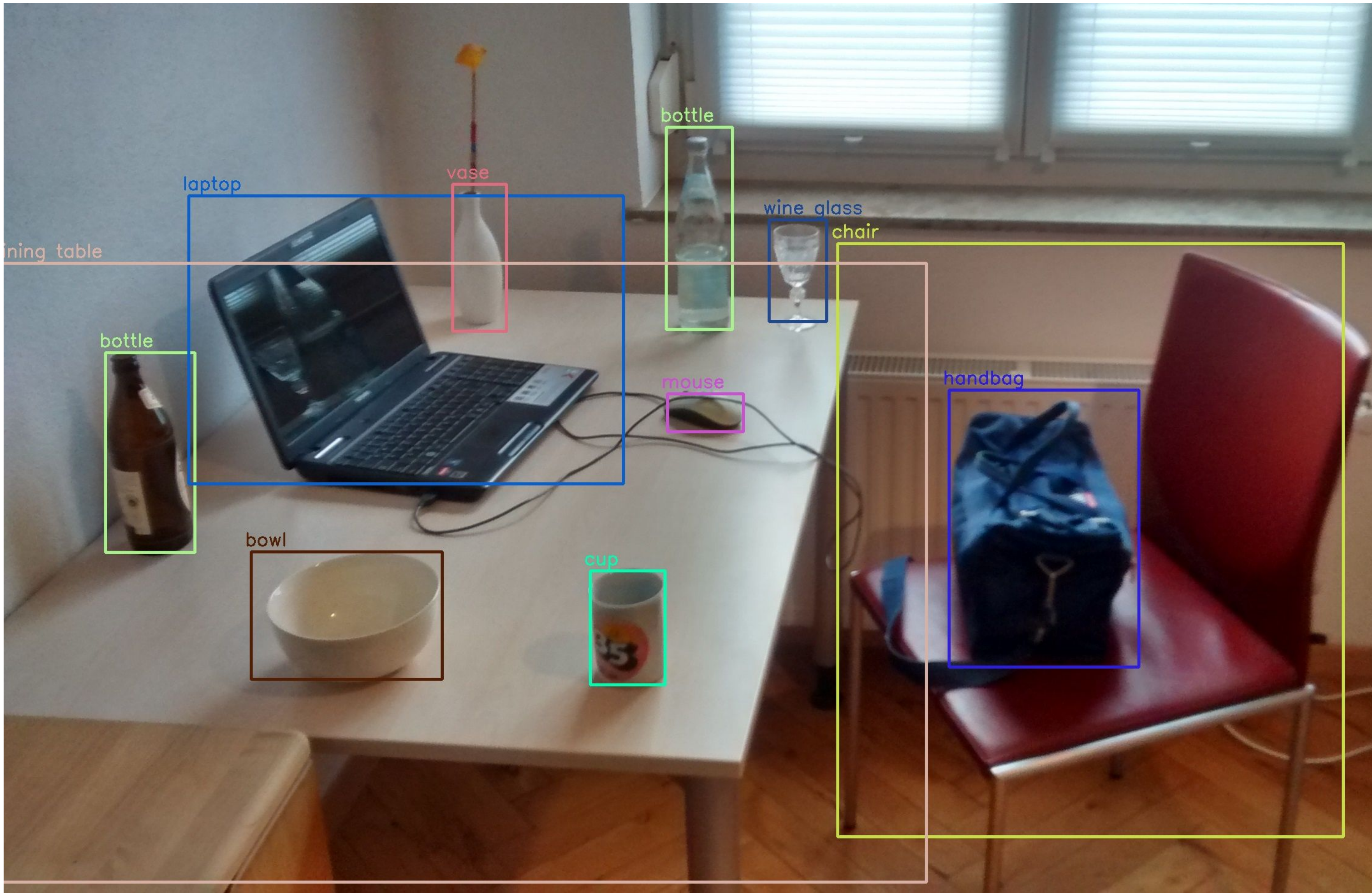


Object detection

Solution: Must test several window size, and potentially several ratios.



Object detection



Object detection

Underlying problem: **Computationally expensive**

In a 1024x768px image, if we want to test sliding window each 8 pixel, we have to estimate 12288 classifications, for a single window size.

If we test 5 window size with 3 different ratios: 12288 x 15 = 184320 classifications for a single image!

Object detection

Recap solution 1:

1. Classify sliding windows of difference size and ratios in the image.
2. Merge intersecting detections with Non-Maximum Suppression.

Limitation: very computationally expensive (tens of thousands of tests for each image).

Object detection

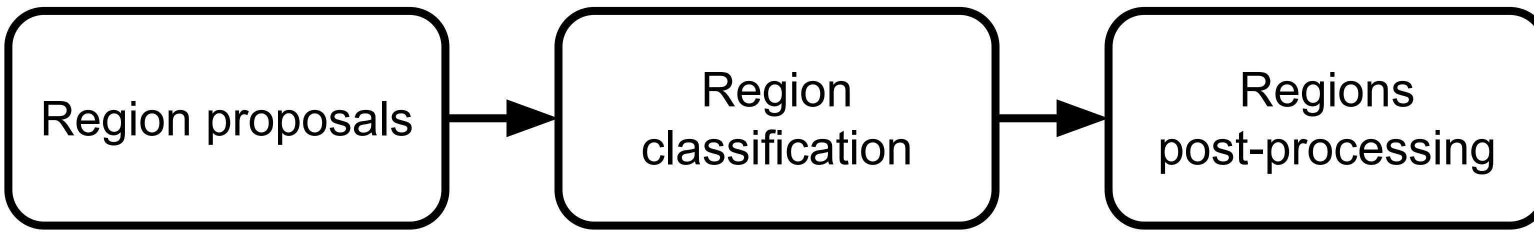
Side question: how do we train this?

1. Dataset creation where each image is associated to bounding boxes that have been manually labelled.

2. Ground truth for each window corresponds to the class of the largest intersection...

Object detection

A more general representation of object detectors.



In 1st solution:

- Region proposals: sliding windows
- Region classification: historically SVM, neural networks…
- Regions post-processing: Non-Maximum Suppression.

R-CNN

Object detection

Observation of solution 1: too many windows to tests, too many region propositions.

→ We need to reduce it.

Object detection

Instead of naively looking for every possible windows, we can look for regions with similar colors, and texture.

However, using this method, objects are generally comprised of different colors, textures, so we need also to test merges of regions.

Object detection

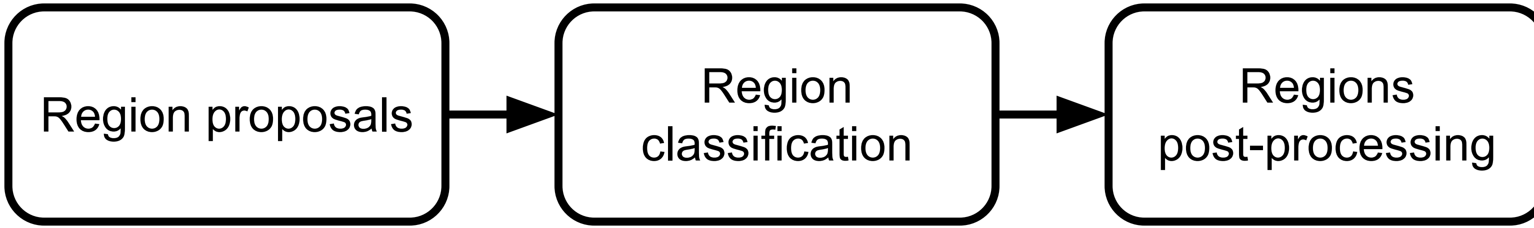
→ This is called Selective Search

Object detection

Selective Search [1] reduces number of regions to approximately 2000 per image.

[1] Segmentation as Selective Search for Object Recognition. Sande et al. 2011

Object detection



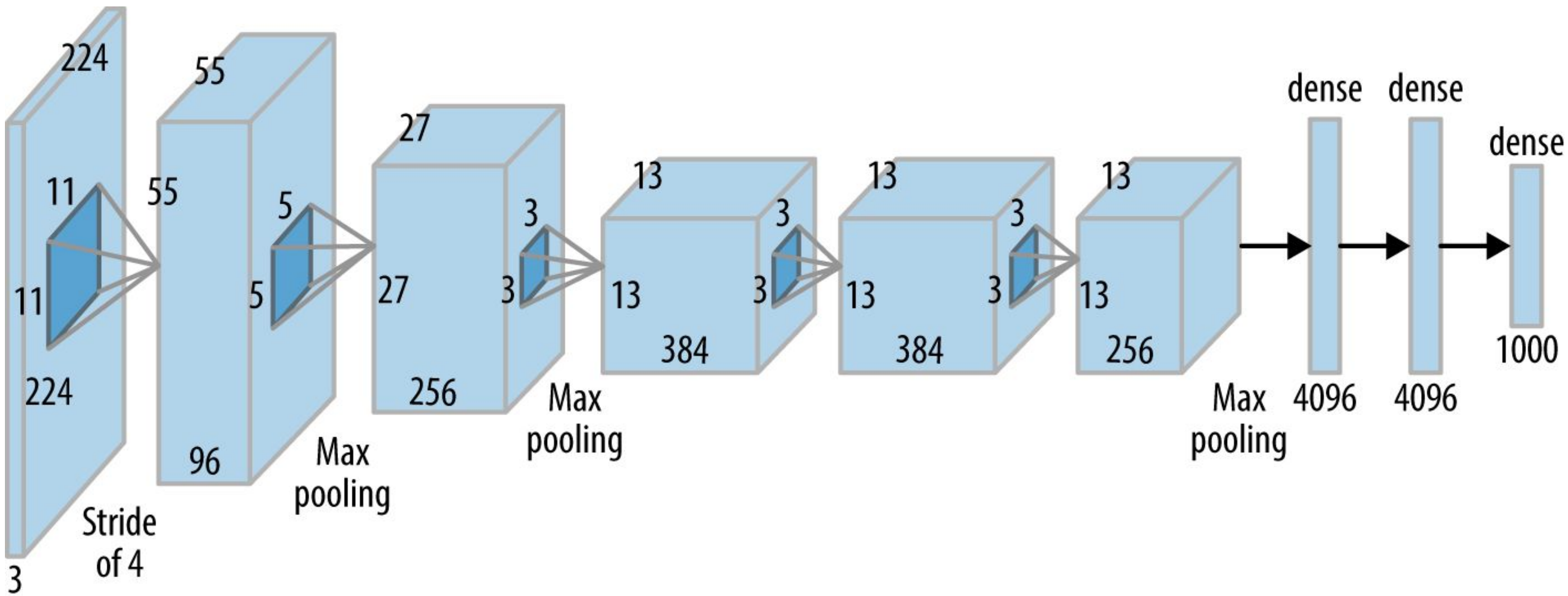
Solution 2: Use Selective Search for region proposals.

→ Main idea behind **R-CNN**

R-CNN has two additional “hacks”.

Object detection

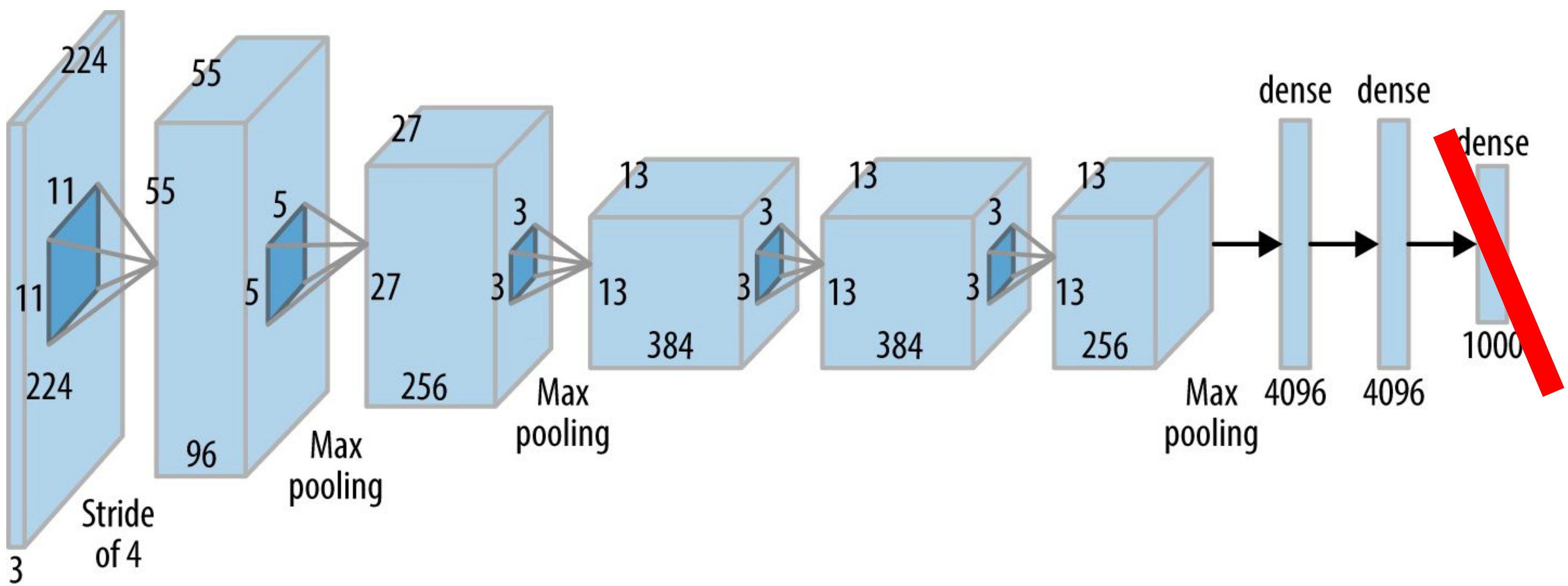
1st hack:



For region classification, it uses a pretrained AlexNet (on another classification task, such as imagenet)...

Object detection

1st hack:



… where the final layer is removed and connected to a SVM.

It allows to reduce the training time, and profit from NN trained on larger dataset.

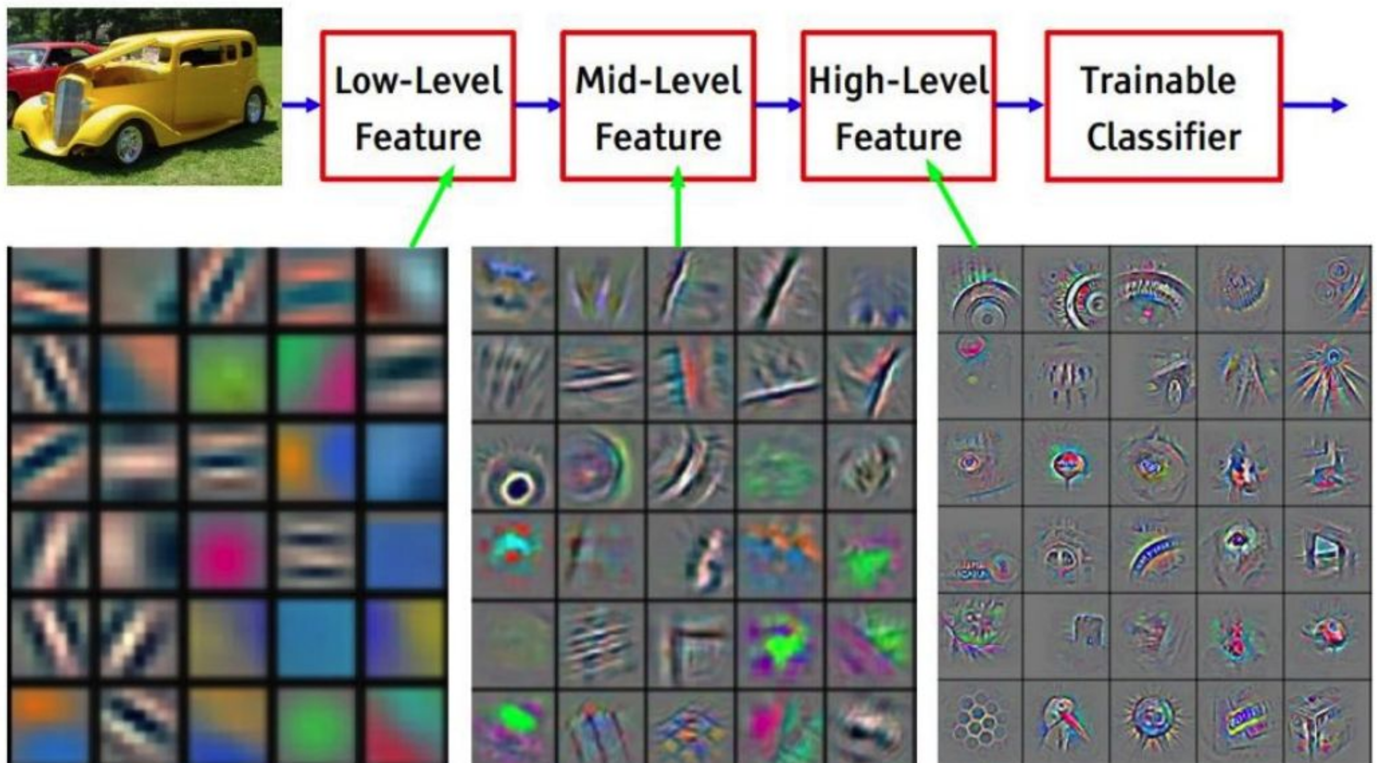
Object detection

AlexNet is used as a Feature Extractor

SVM use these features to classify the target classes (this is called **transfer learning**)

Learnings of First convolution layer on image of size 224X224X3

The hierarchical layer structure allows to learn hierarchical filters (features)



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Slide credit: Yann LeCun

Object detection

2nd hack: in the regions post-processing step, the window positions and dimensions are corrected using a regression.

$$\hat{G}_x = P_w d_x(P) + P_x \tag{1}$$

$$\hat{G}_y = P_h d_y(P) + P_y \tag{2}$$

$$\hat{G}_w = P_w \exp(d_w(P)) \tag{3}$$

$$\hat{G}_h = P_h \exp(d_h(P)). \tag{4}$$

Px, Py, Pw, Ph: x, y, width, height of detected window.

Gx, Gy, Gw, Gh: x, y, width, height of target window.

Object detection

Issues:

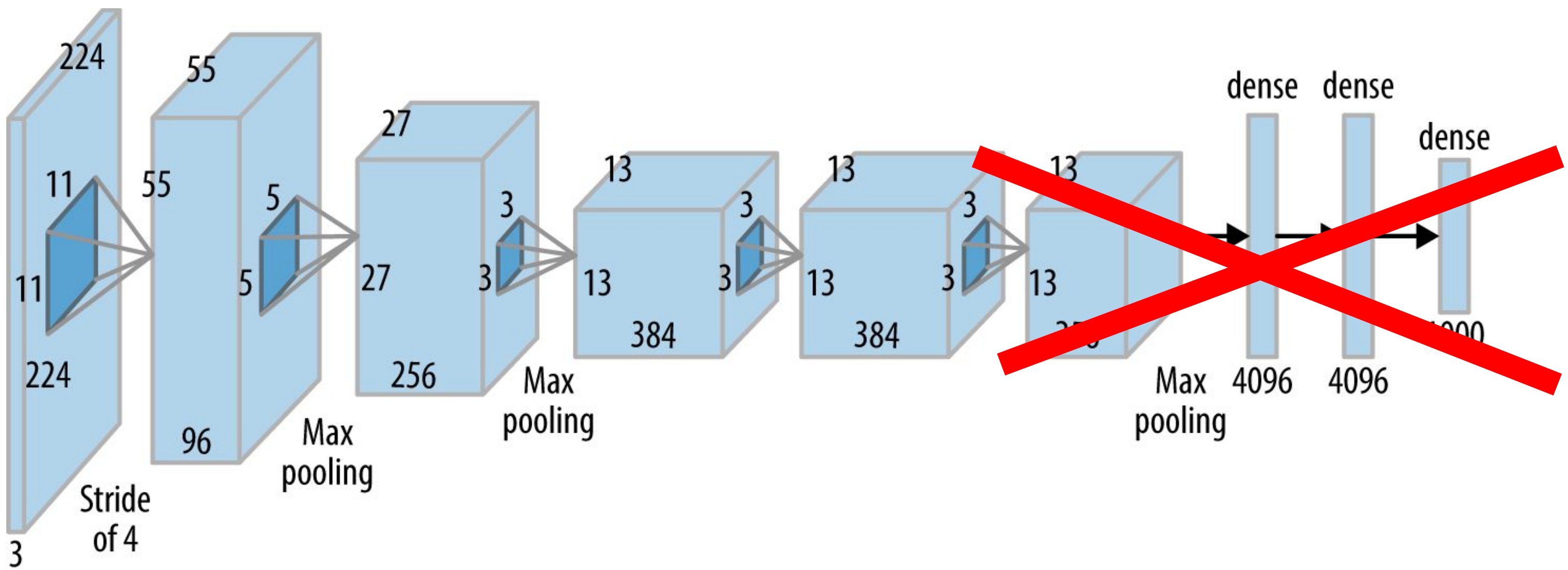
- Classifier in two parts, trained separately.
- Regressor trained separately.
- Still a lot of computation redundancies → still slow: 40-60s per image.

Object detection

How can we reduce the computational redundancies?

By applying the neural network on the whole image, and then extracting relevant information when testing a window. → Use Region of Interest pooling layer.

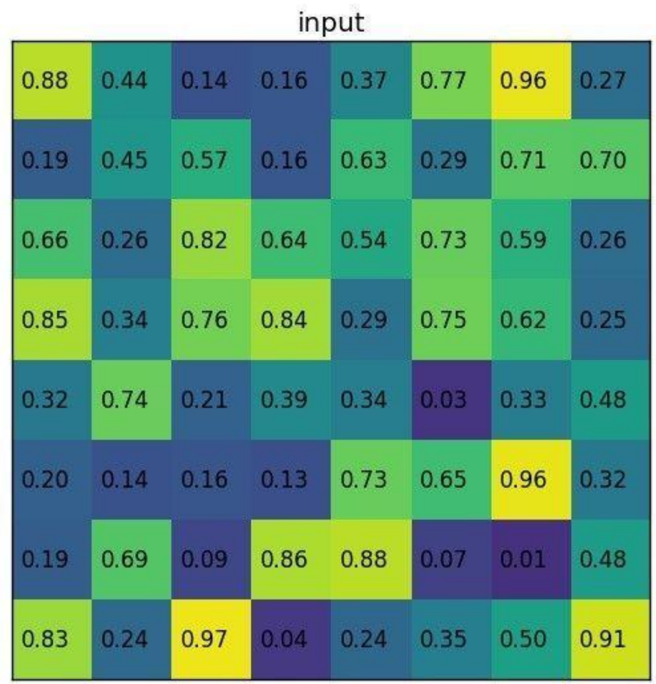
Image



Object detection

2x2 Region of Interest pooling layer

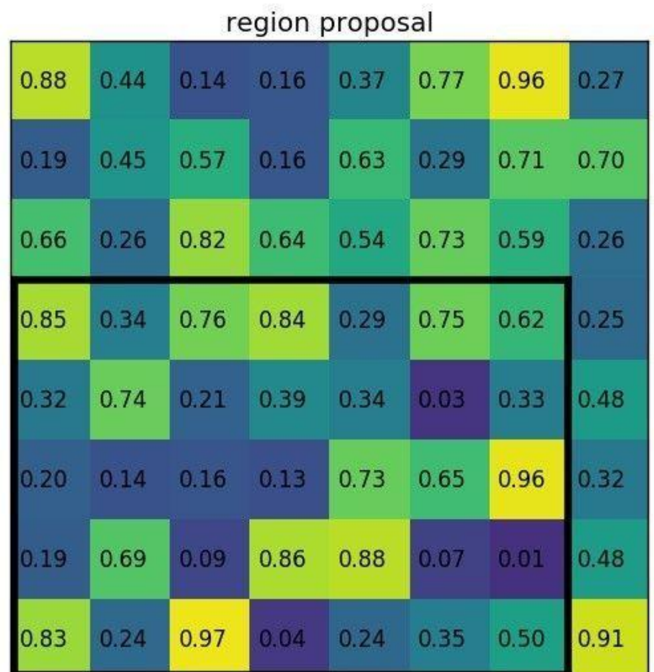
It allows us to reuse the feature map from the convolutional network



https://deepsense.ai/region-of-interest-pooling-explained/

Object detection

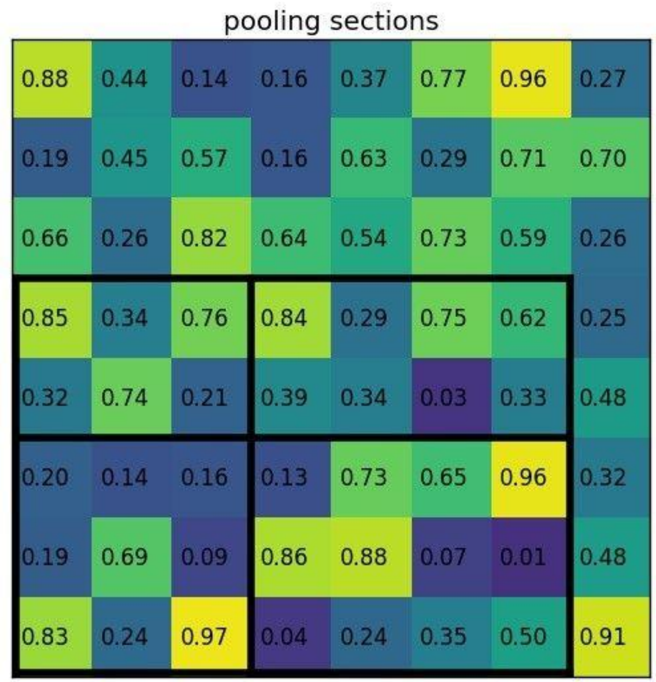
2x2 Region of Interest pooling layer



https://deepsense.ai/region-of-interest-pooling-explained/

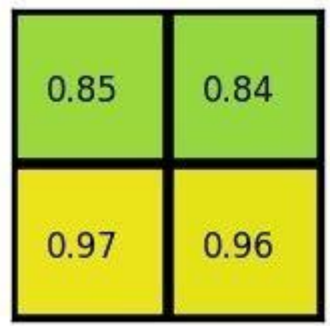
Object detection

2x2 Region of Interest pooling layer



https://deepsense.ai/region-of-interest-pooling-explained/

Object detection



Object detection

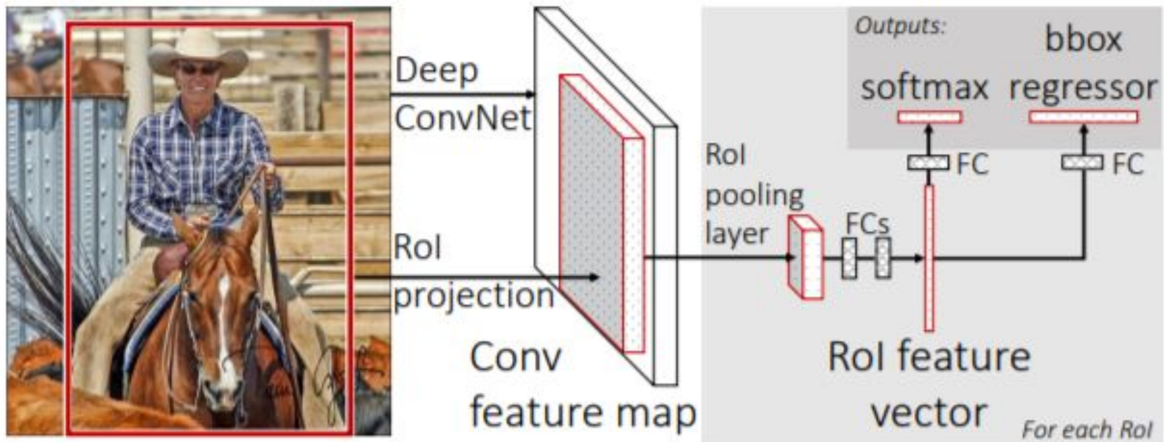


Figure 1. Fast R-CNN architecture. An input image and multiple regions of interest (RoIs) are input into a fully convolutional network. Each RoI is pooled into a fixed-size feature map and then mapped to a feature vector by fully connected layers (FCs). The network has two output vectors per RoI: softmax probabilities and per-class bounding-box regression offsets. The architecture is trained end-to-end with a multi-task loss.

Object detection

→ Main idea of **Fast R-CNN**

Results:

- Slightly better performance than R-CNN
- Train speedup: 8-20x
- Test speedup: 150-213x (3-10 images per seconds)

Object detection

New bottleneck: region proposals (Selective Search)

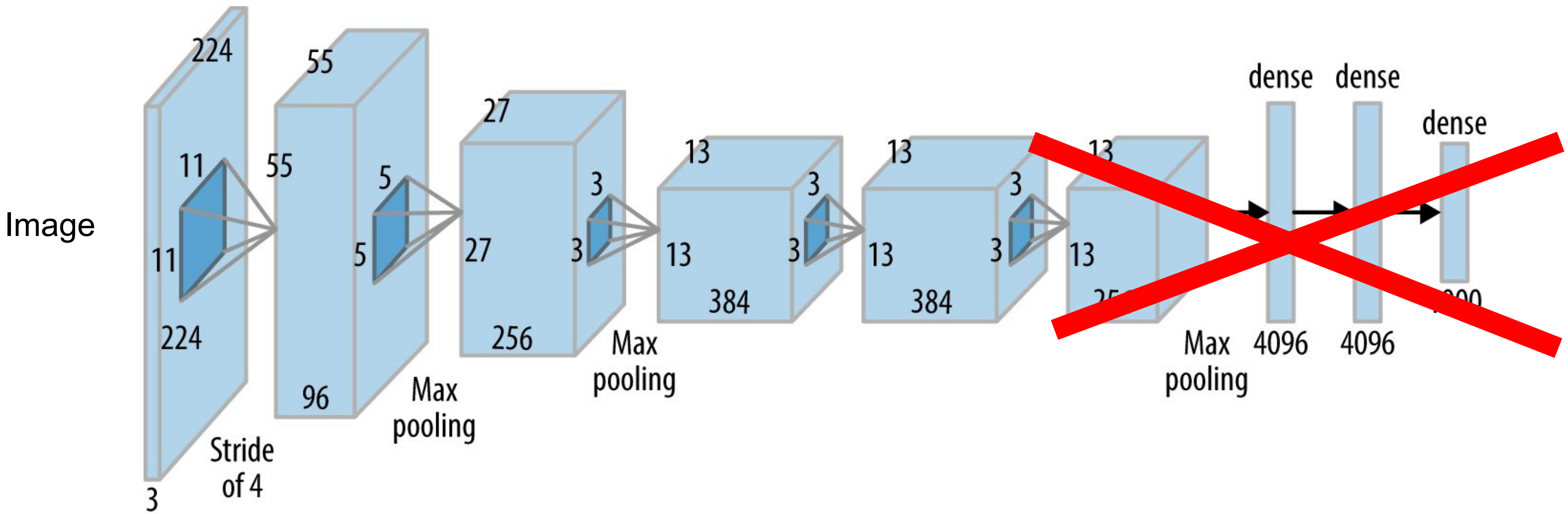
Object detection

New bottleneck: region proposals (Selective Search)

- Regions proposed not always adapted (missed regions)
- Now a bit slow compared to the rest

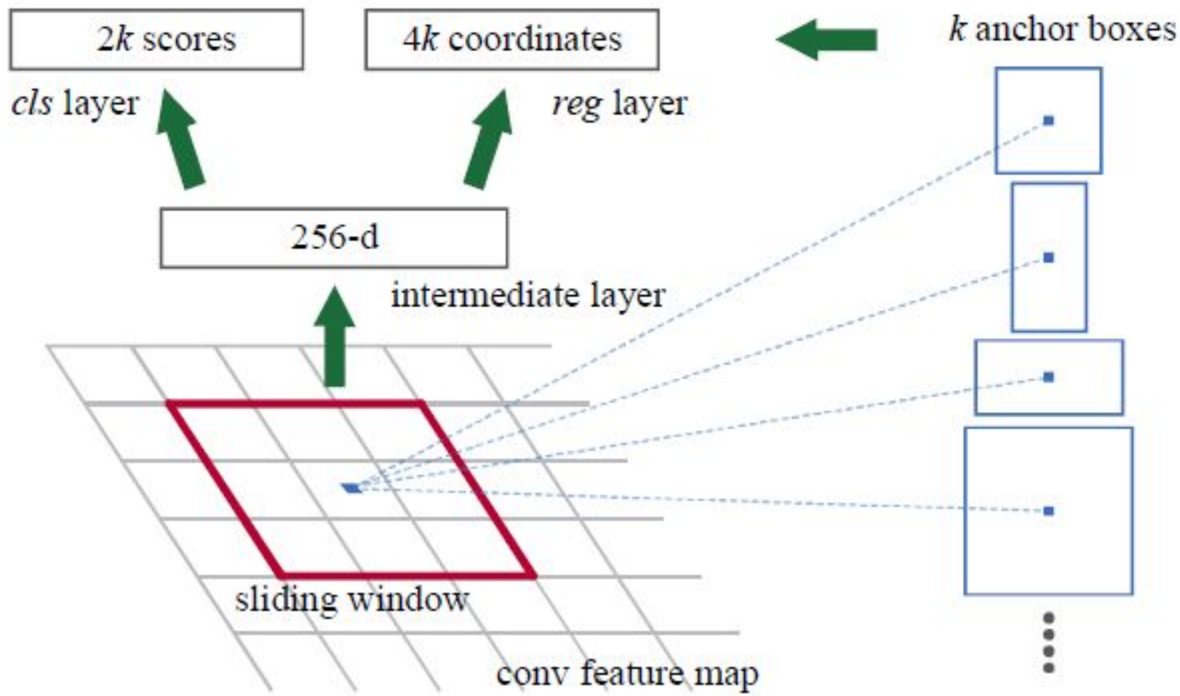
Object detection

New concept: Region Proposal Network (RPN)



Object detection

New concept: Region Proposal Network (RPN)



Receptive field:
~ 200px

Object detection

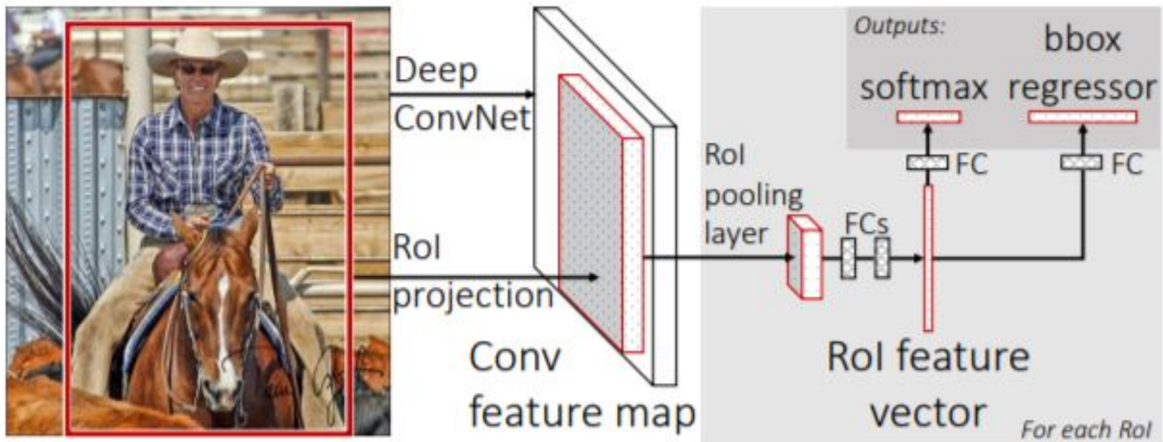


Figure 1. Fast R-CNN architecture. An input image and multiple regions of interest (RoIs) are input into a fully convolutional network. Each RoI is pooled into a fixed-size feature map and then mapped to a feature vector by fully connected layers (FCs). The network has two output vectors per RoI: softmax probabilities and per-class bounding-box regression offsets. The architecture is trained end-to-end with a multi-task loss.

Object detection

→ Main idea of **Faster R-CNN**

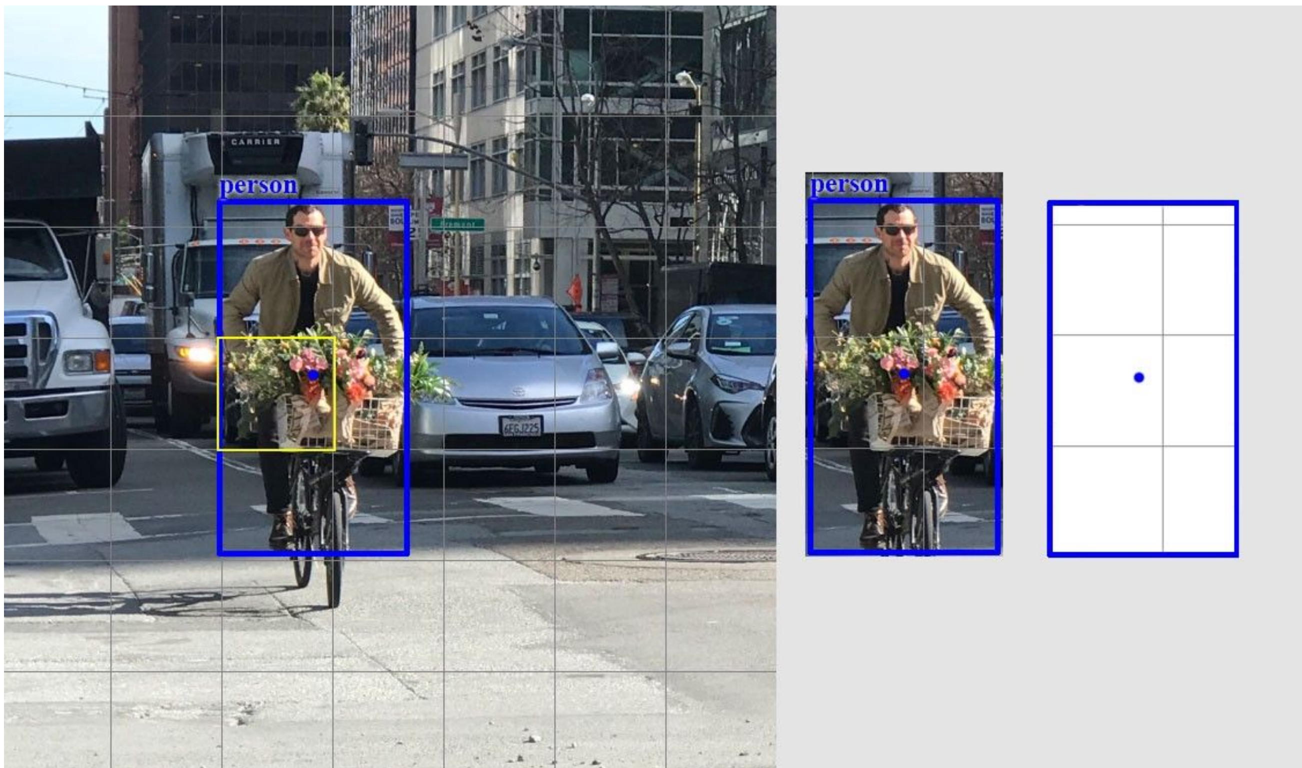
Results:

- Even better performance and speed-up

YOLO

Object detection

YOLO: You Only Look Once



Object detection

YOLO: You Only Look Once

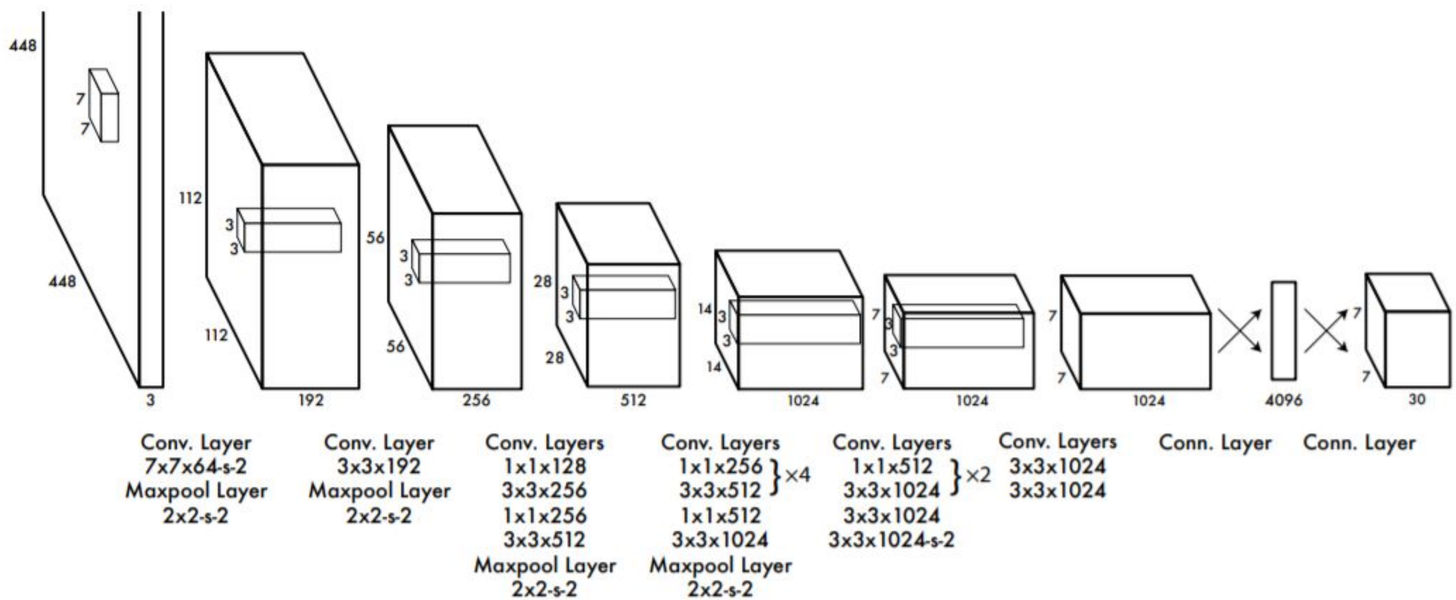


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.

Object detection

7x7 cells

Output for each cell:

2 bounding boxes + confidences : 2 * (4 +1)

20 classes

→ 7x7x30

Object detection

Much faster! (200 fps)

Problems if object is too small and for groups of small objects.

Localization is less accurate.

Object detection

YoloV2, YoloV3...

Object detection

RetinaNet: state of the art (works well with dense and small scale objects)

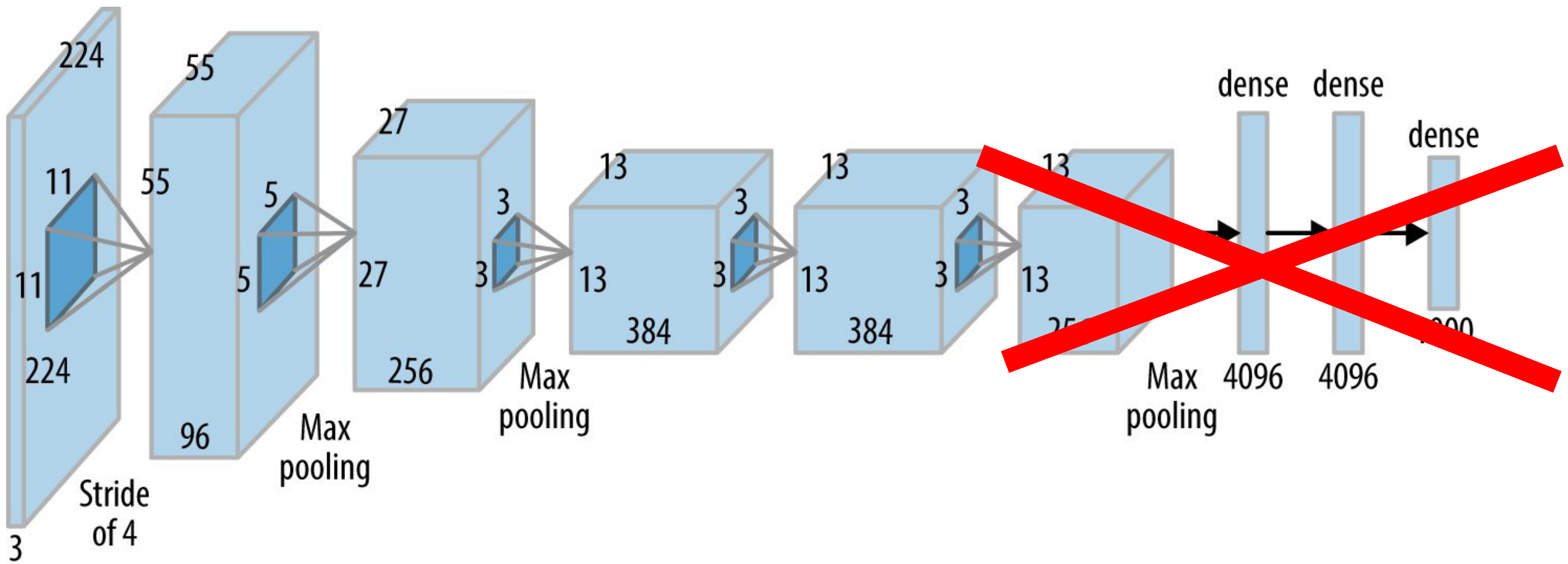
Two main ideas…

- Feature Pyramid Network
- Focal Loss

Object detection

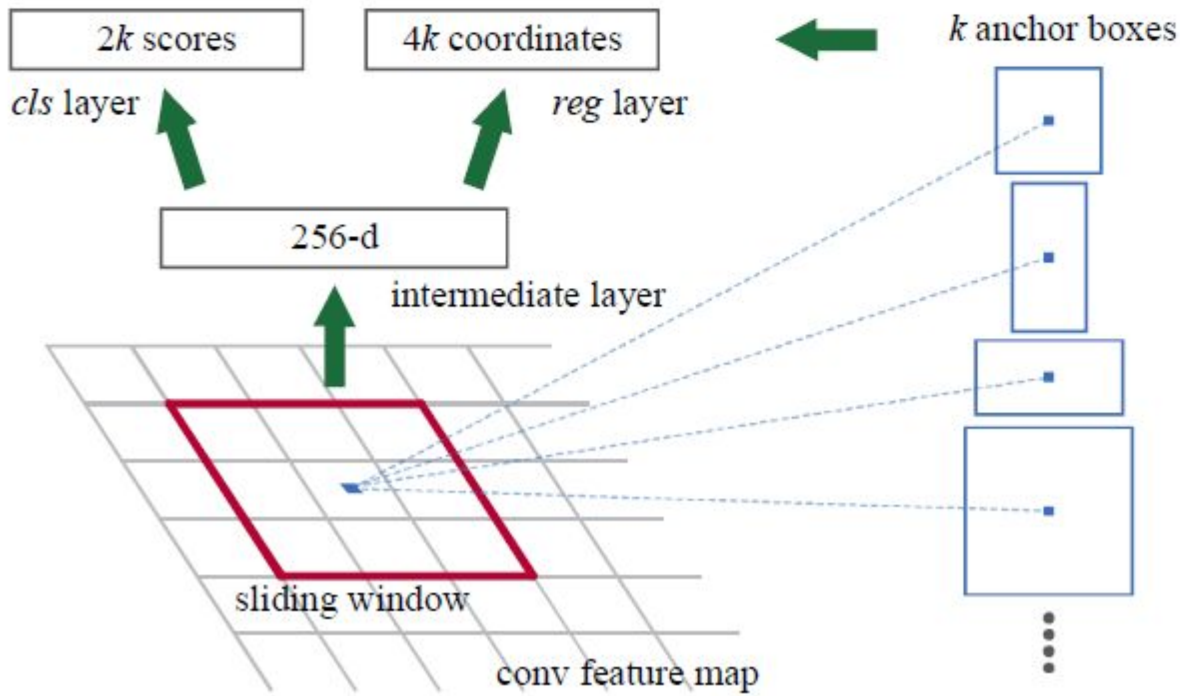
New concept: Region Proposal Network (RPN)

Image



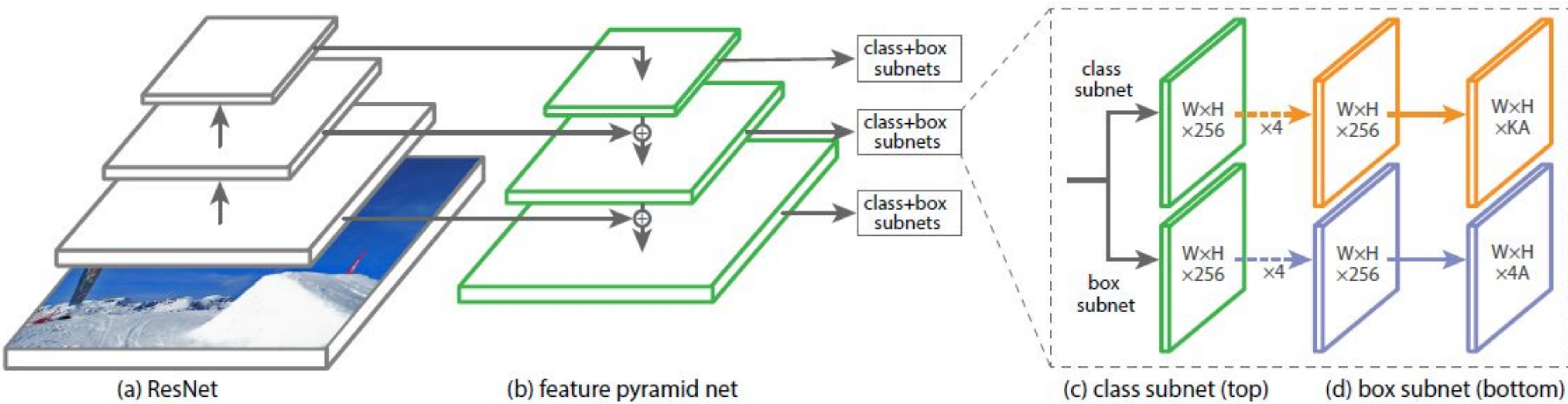
Object detection

New concept: Region Proposal Network (RPN)

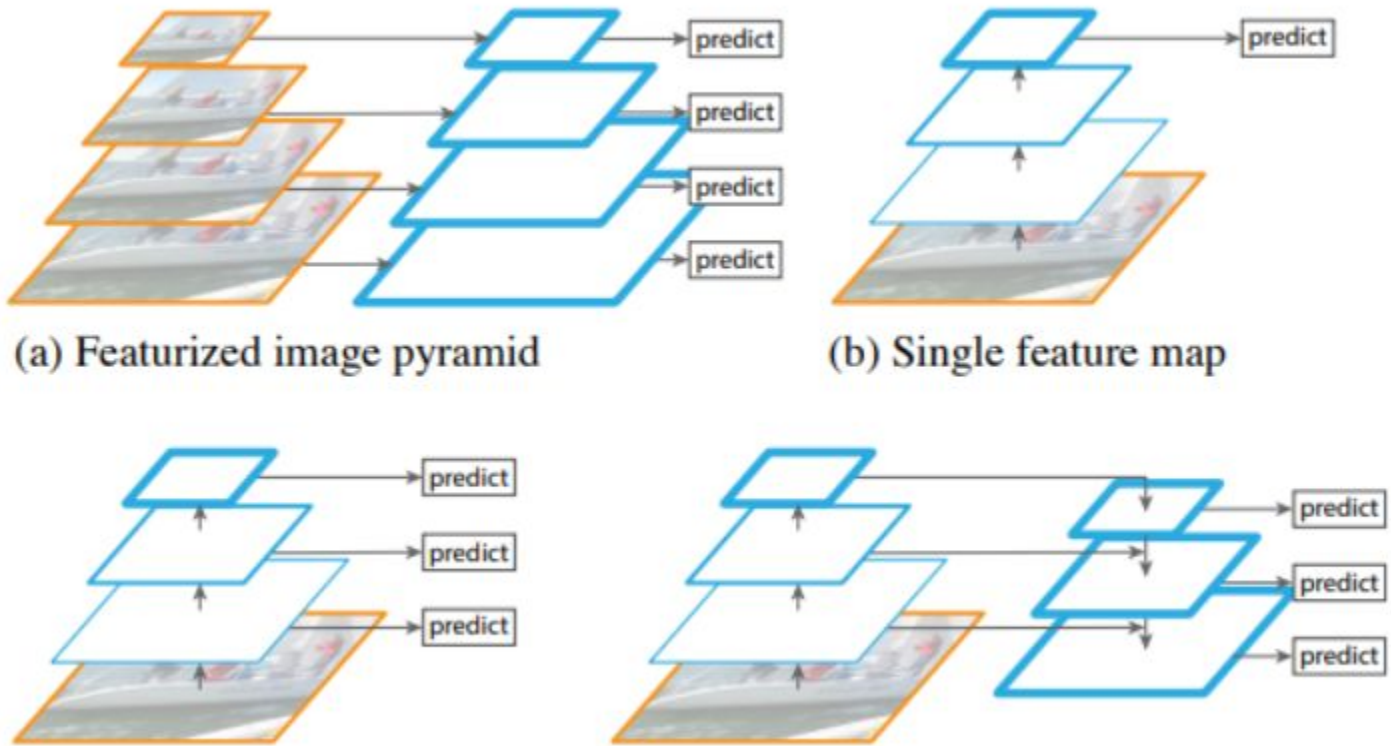


Receptive field:
~ 200px

Object detection



Different types of pyramid architectures



(a) Featurized image pyramid

(b) Single feature map

(c) Pyramidal feature hierarchy

(d) Feature Pyramid Network

Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan: “Feature Pyramid Networks for Object Detection”, 2016

Object detection: Focal loss

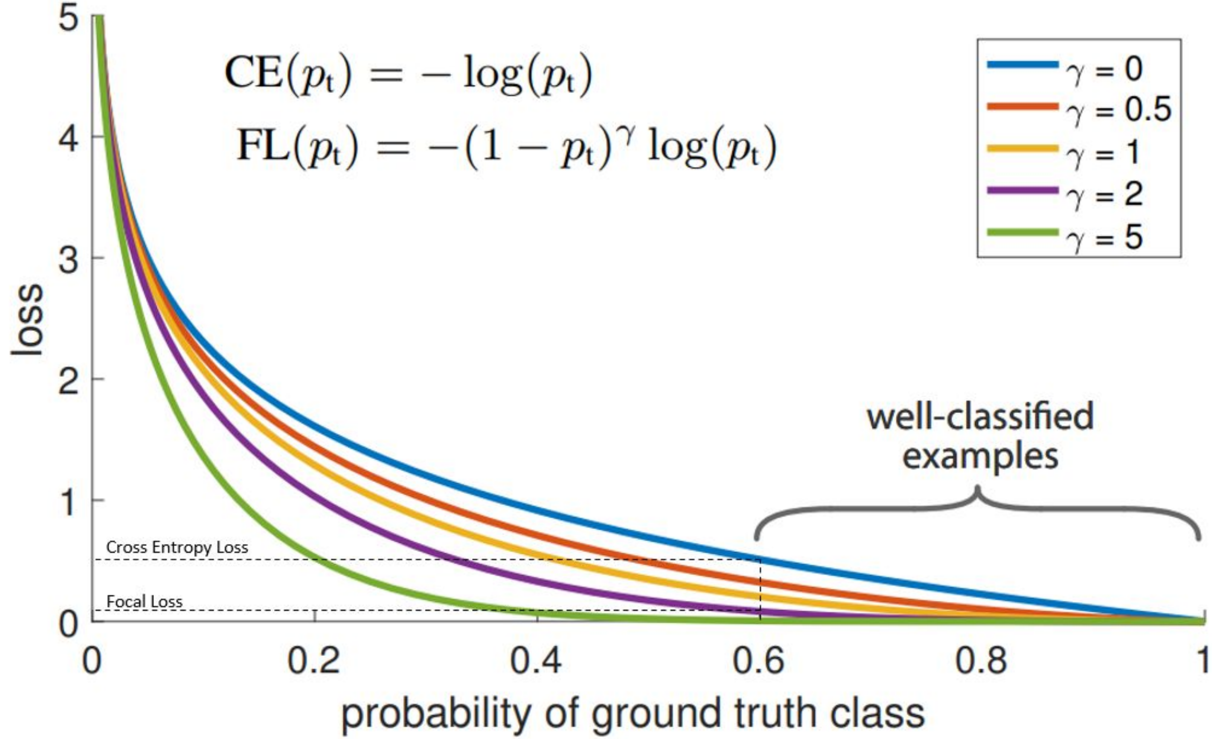
Focal loss: $$\mathrm{FL}(p_t) = -(1 - p_t)^{\gamma} \log(p_t).$$

Focal Loss reduces the loss contribution from easy examples and increases the importance of correcting misclassified examples.

Addresses class imbalance by focusing on samples with lower probability.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He: “Focal Loss for Dense Object Detection”, 2017;

Object detection: Focal loss

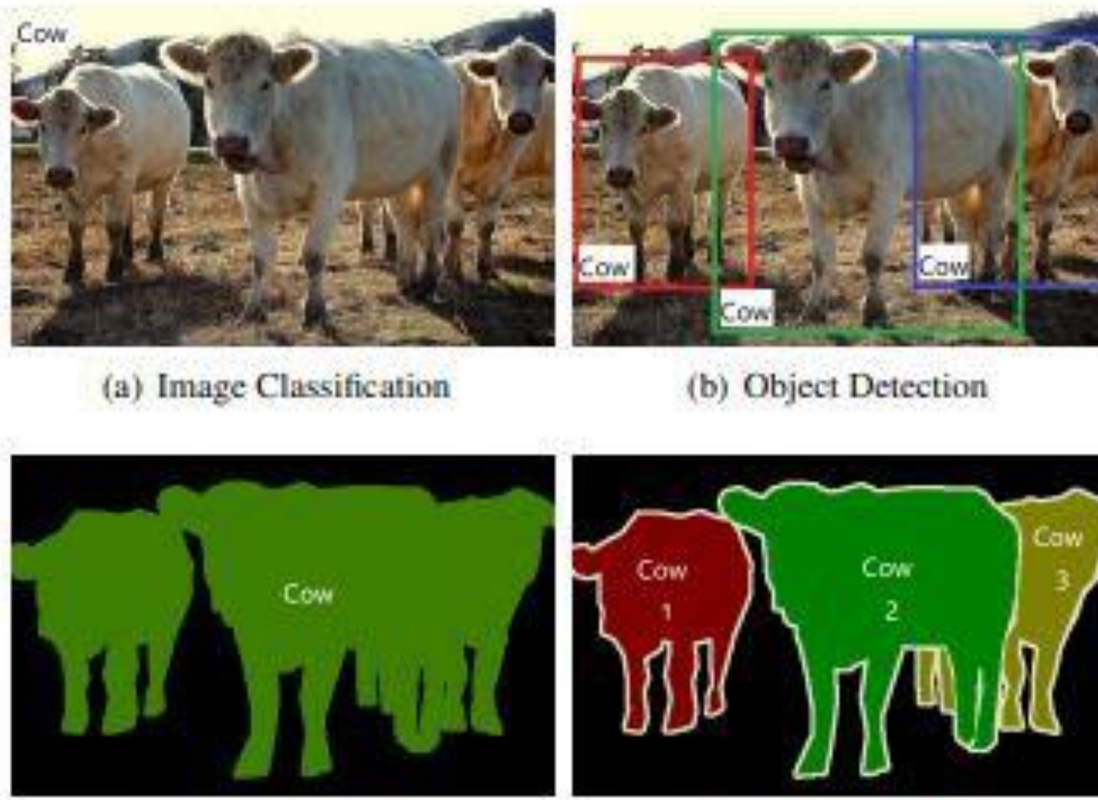


Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He: “Focal Loss for Dense Object Detection”, 2017;

Instance segmentation

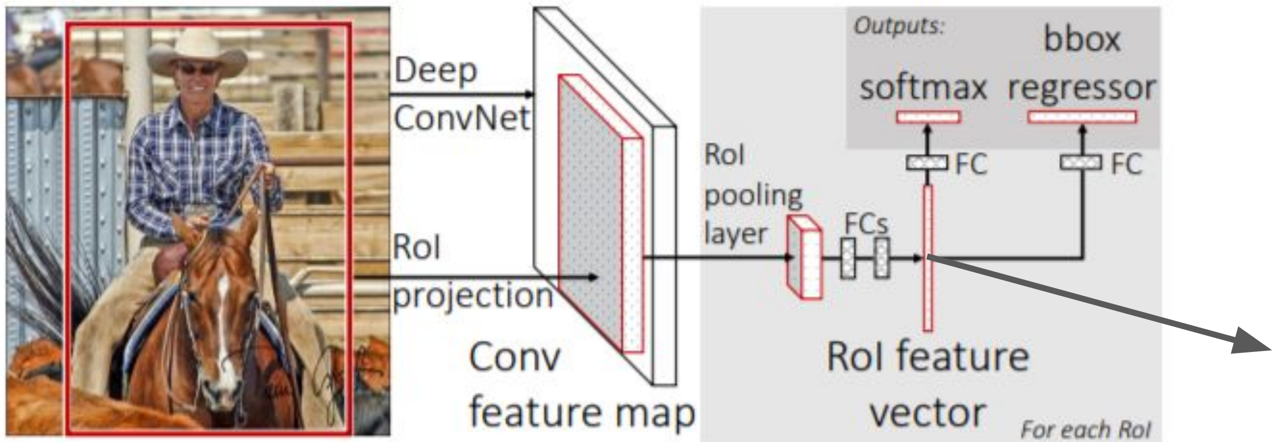
Instance segmentation

Instance segmentation =
Object detection +
Semantic Segmentation



(a) Image Classification (b) Object Detection
(c) Semantic Segmentation (d) Instance Segmentation

Instance segmentation



28x28
segmentation
mask

Figure 1. Fast R-CNN architecture. An input image and multiple regions of interest (RoIs) are input into a fully convolutional network. Each RoI is pooled into a fixed-size feature map and then mapped to a feature vector by fully connected layers (FCs). The network has two output vectors per RoI: softmax probabilities and per-class bounding-box regression offsets. The architecture is trained end-to-end with a multi-task loss.

Instance segmentation

Mask-RCNN

Instance segmentation

https://github.com/matterport/Mask_RCNN/blob/master/samples/coco/inspect_model.ipynb

Human Pose Estimation with Openpose

Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields
Zhe Cao, Shih-En Wei, Tomas Simon, Yaser Sheikh (2017)

Slides taken mainly from https://github.com/ZheC/Realtime_Multi-Person_Pose_Estimation
Some images from: https://arvrjourney.com/human-pose-estimation-using-openpose-with-tensorflow-part-2-e78ab9104fc8

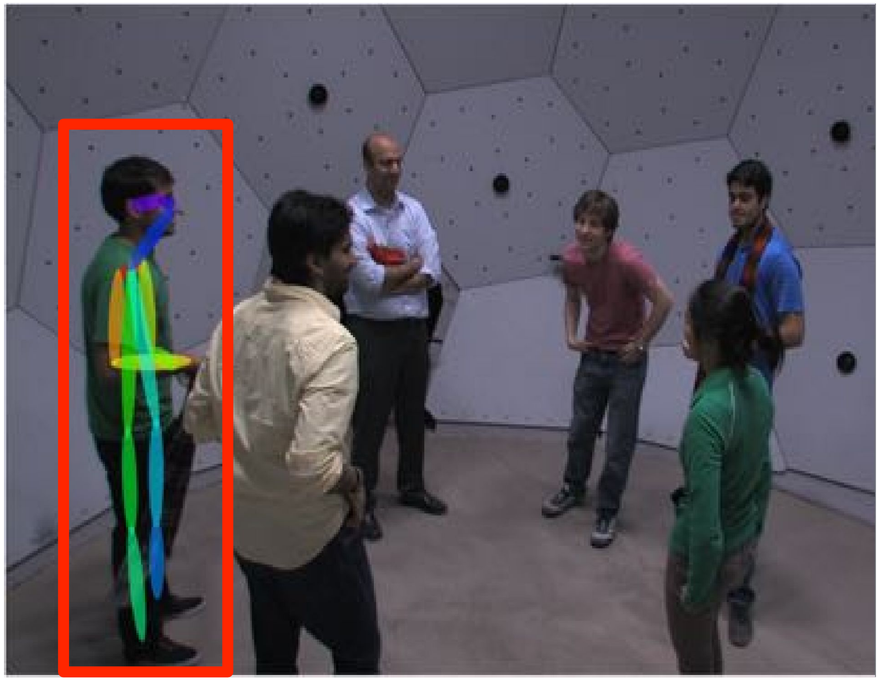
Top-down Approach: Person Detection + Pose Estimation

Top-down

Top-down Approach: Person Detection + Pose Estimation

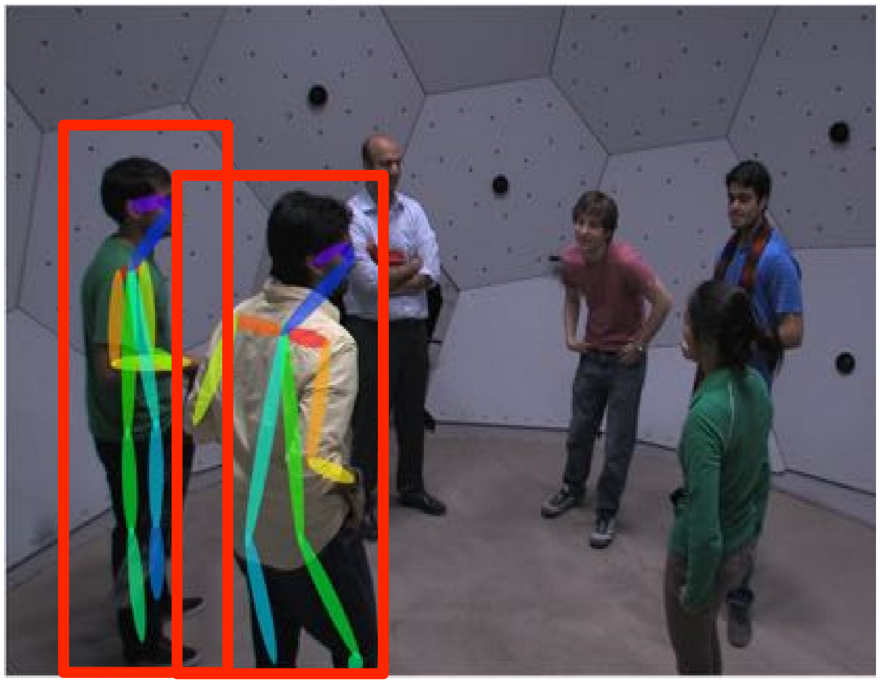
Top-down

Top-down Approach: Person Detection + Pose Estimation



Top-down

Top-down Approach: Person Detection + Pose Estimation

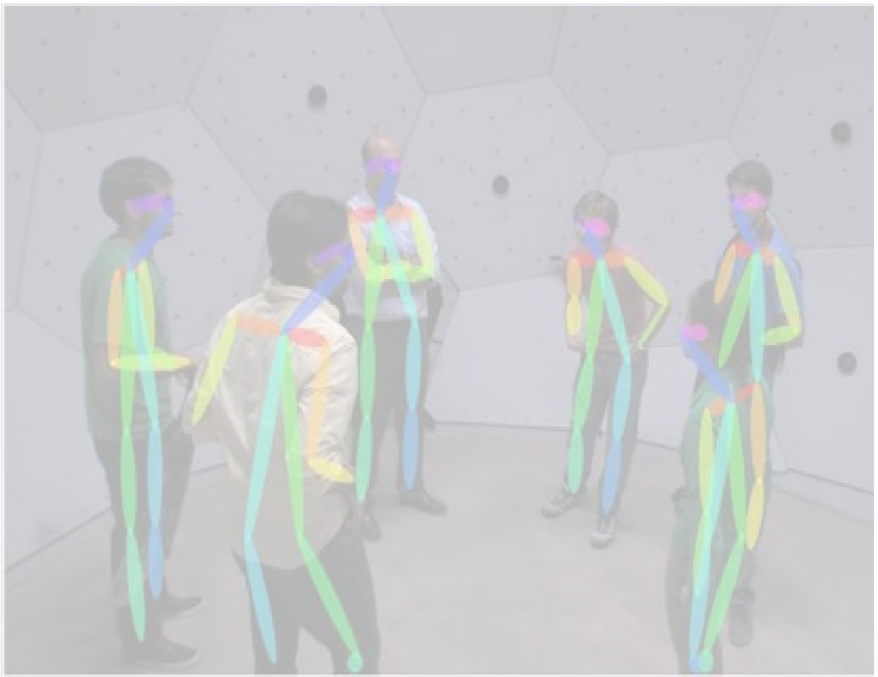


Top-down

Top-down Approach: Person Detection + Pose Estimation

Top-down

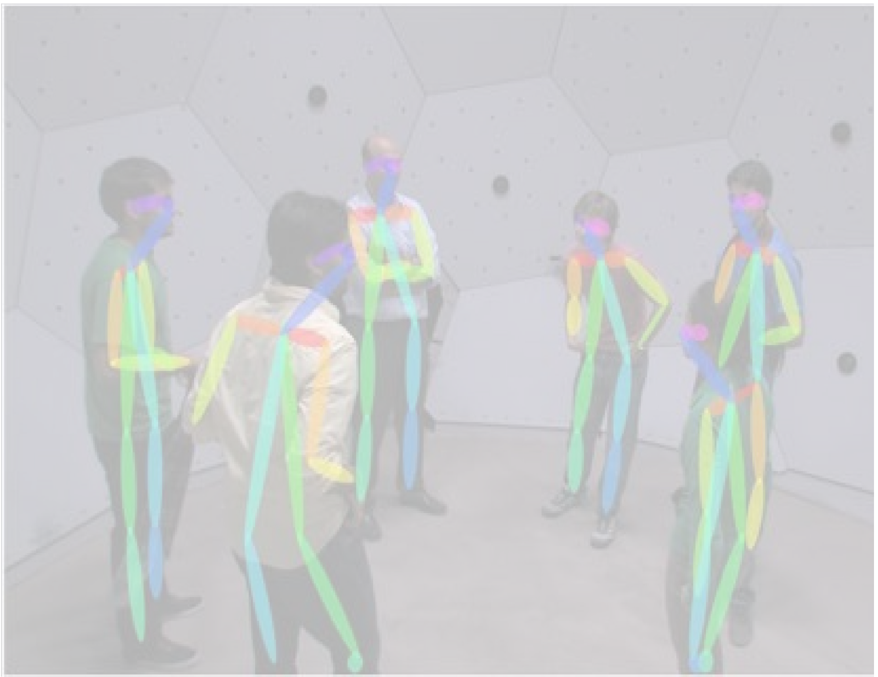
Openpose: Parts Detection + Parts Association



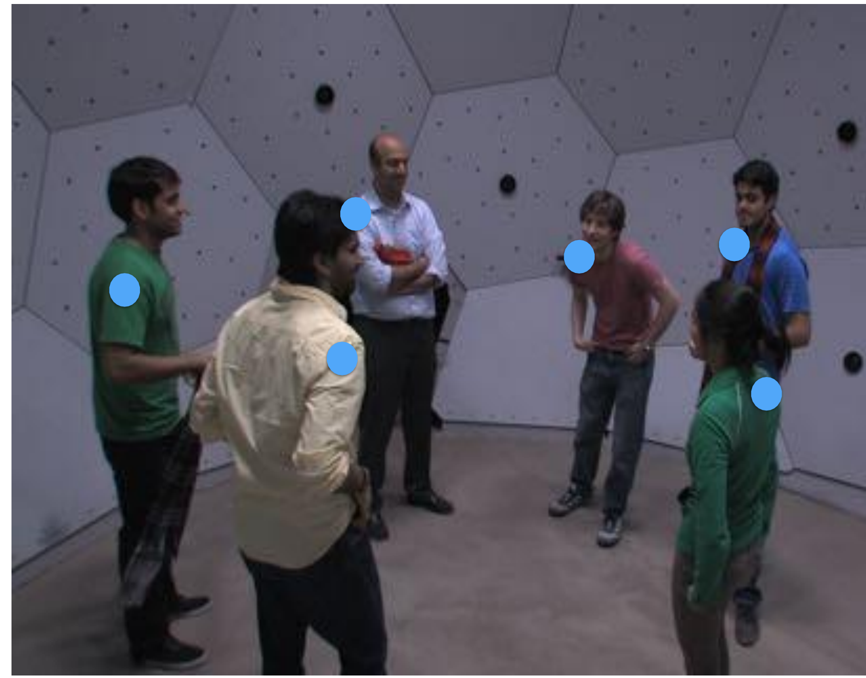
Top-down

Openpose

Openpose: Parts Detection + Parts Association

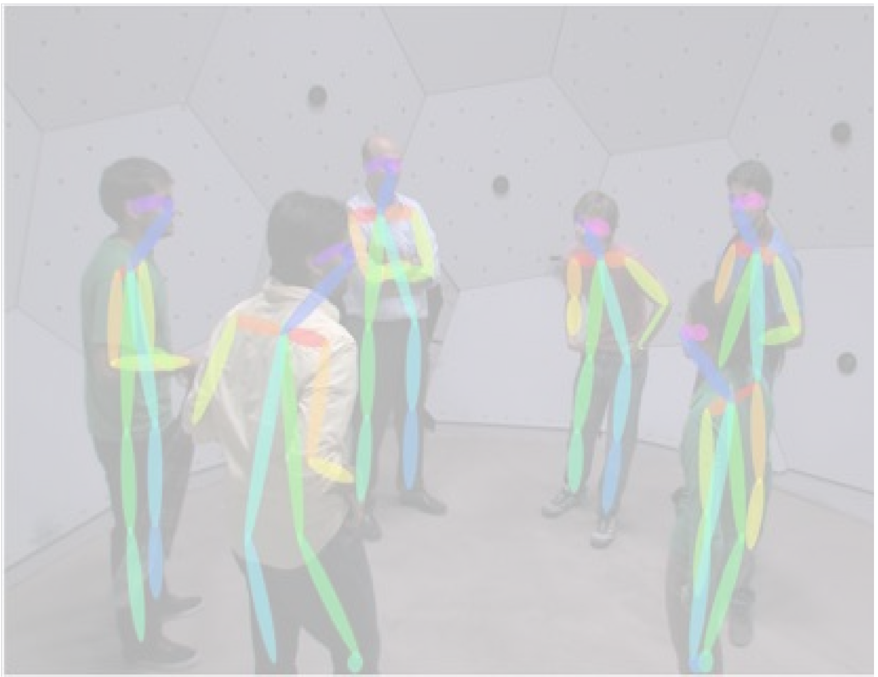


Top-down



Openpose

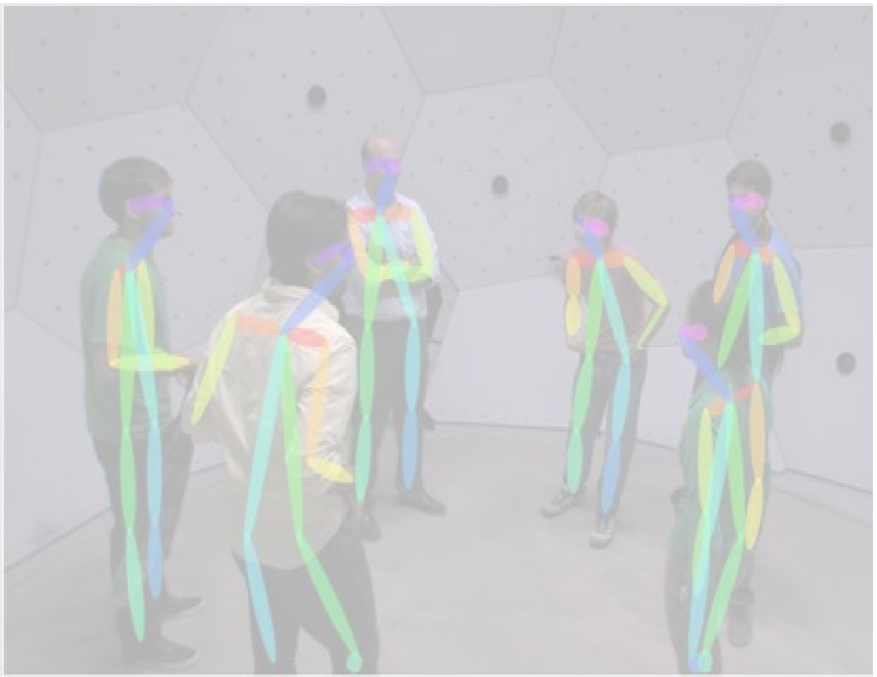
Openpose: Parts Detection + Parts Association



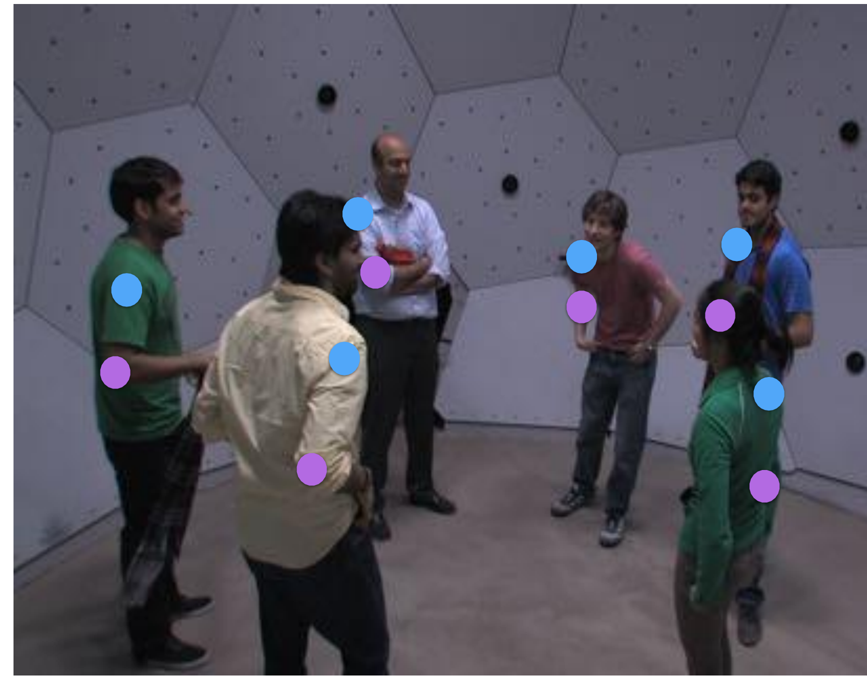
Top-down

Openpose

Openpose: Parts Detection + Parts Association

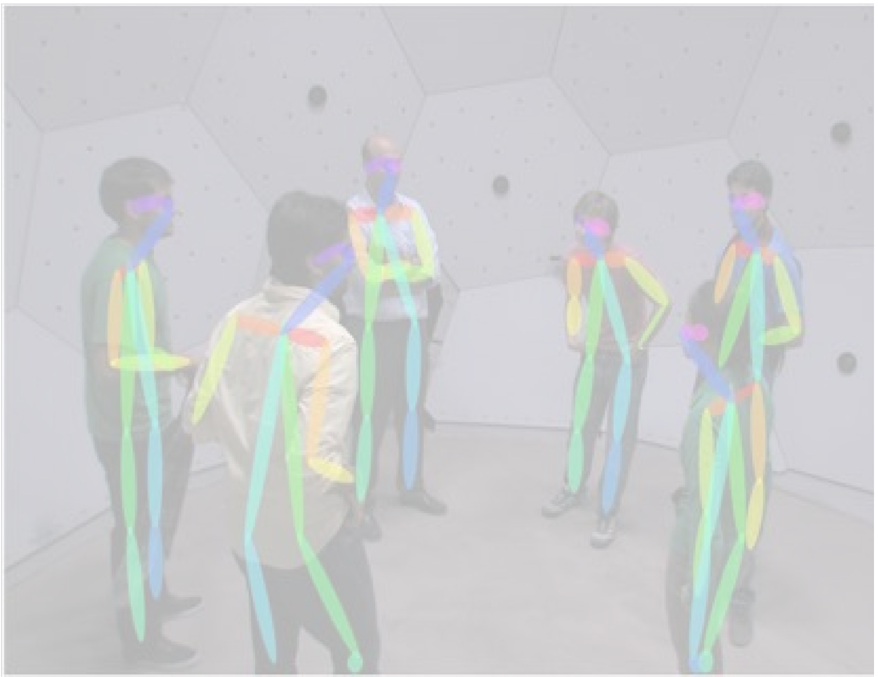


Top-down

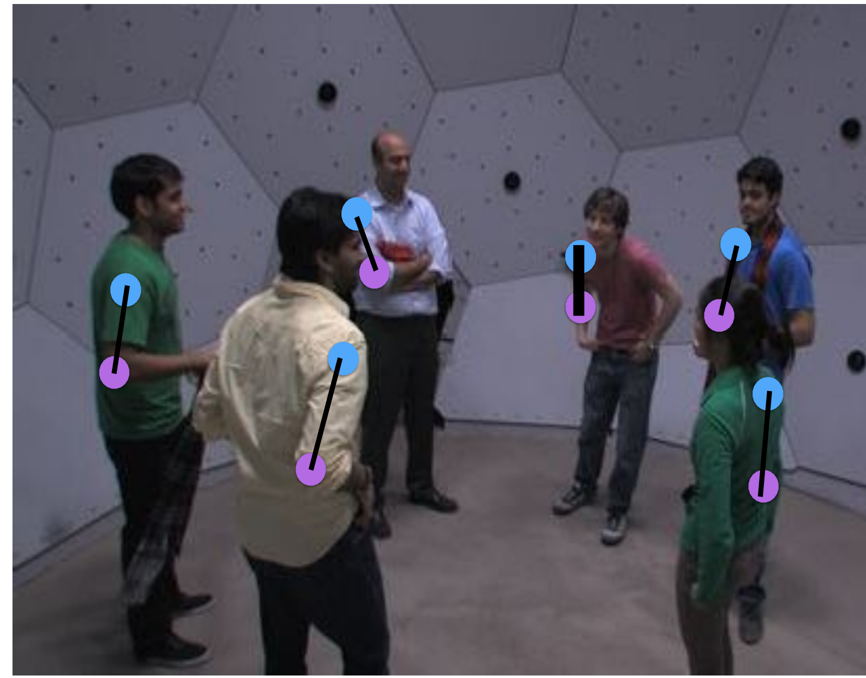


Openpose

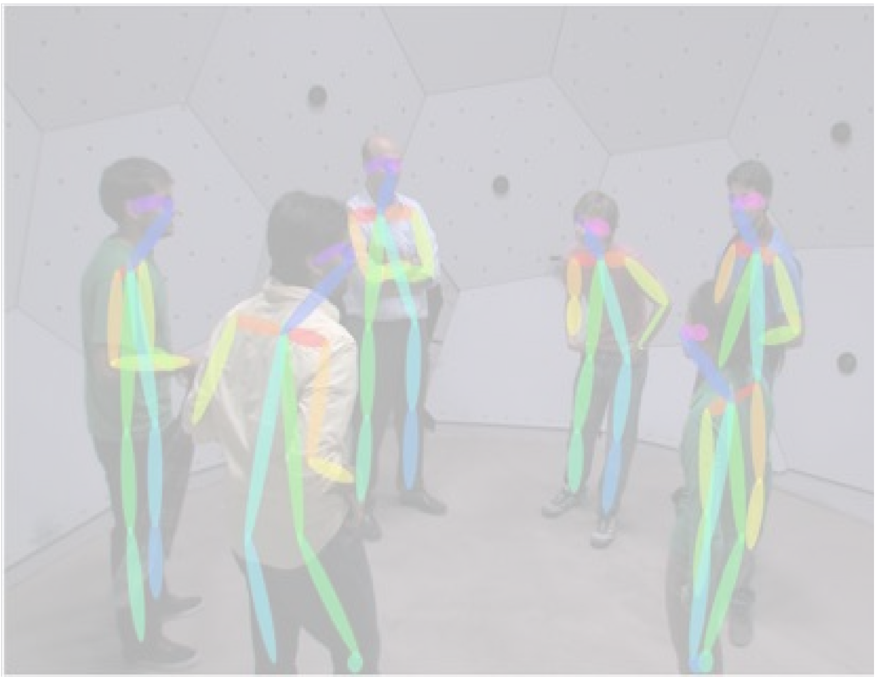
Openpose: Parts Detection + Parts Association



Top-down



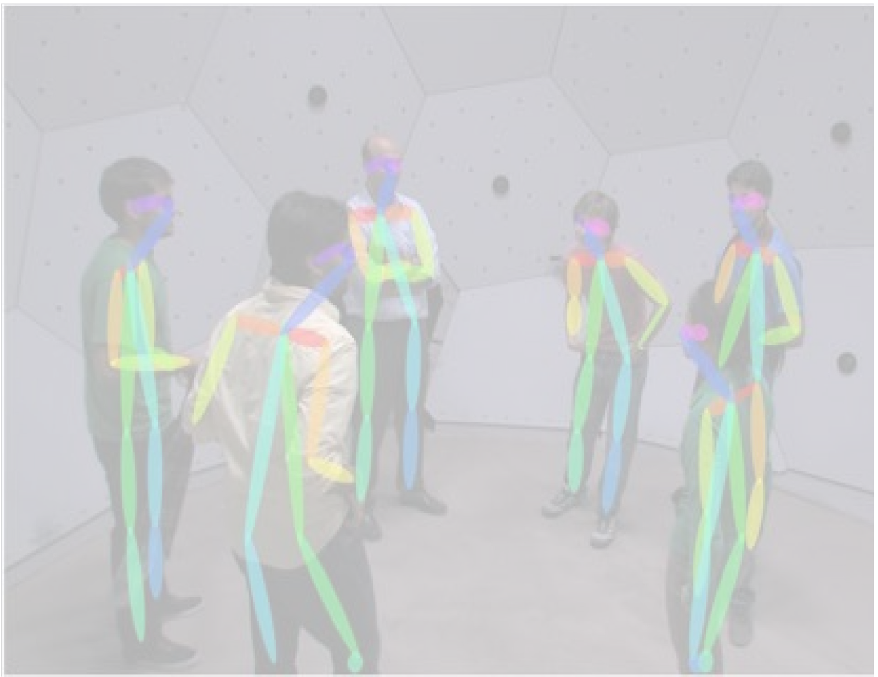
Openpose: Parts Detection + Parts Association



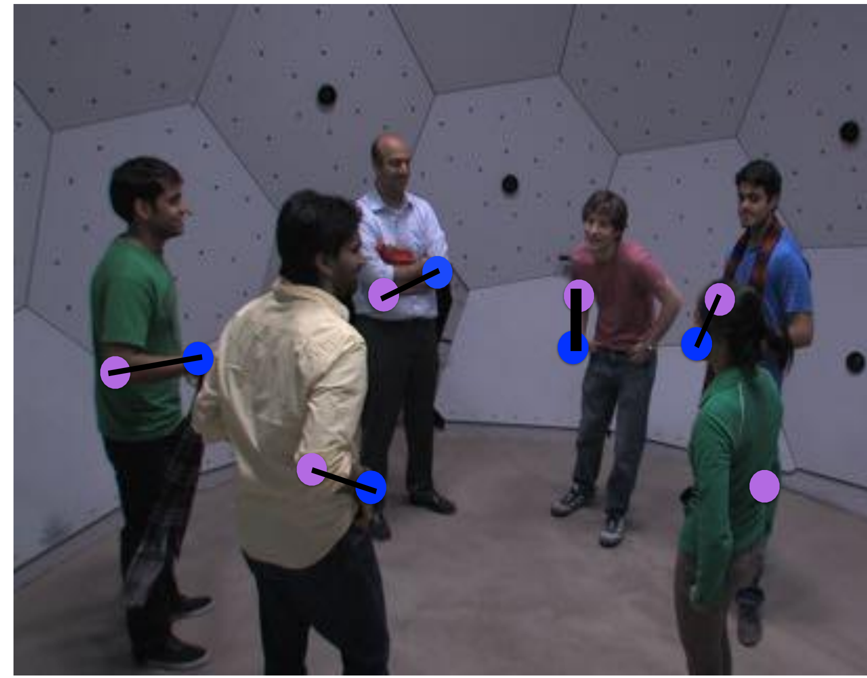
Top-down

Part Affinity Fields

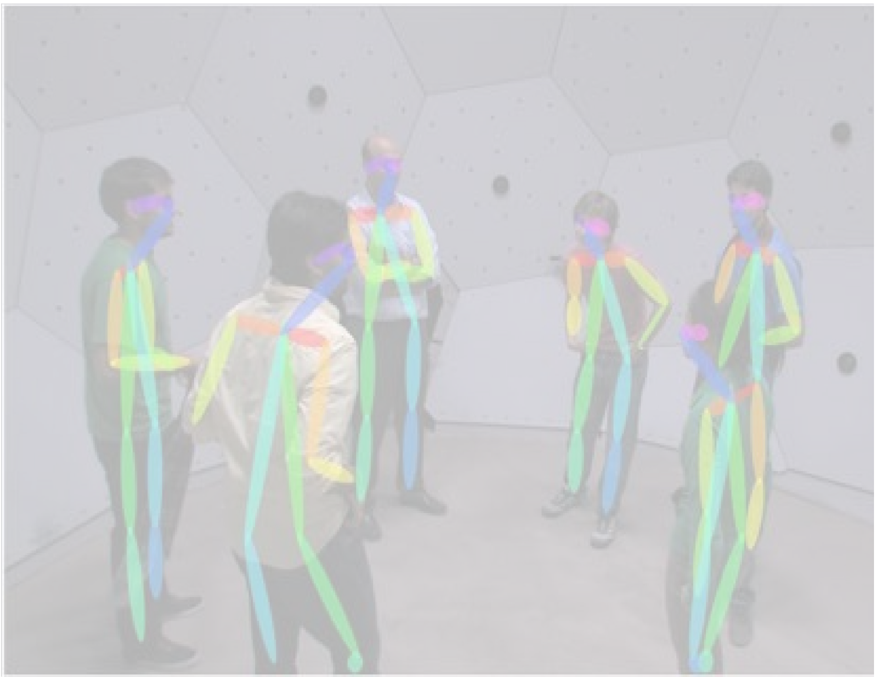
Openpose: Parts Detection + Parts Association



Top-down



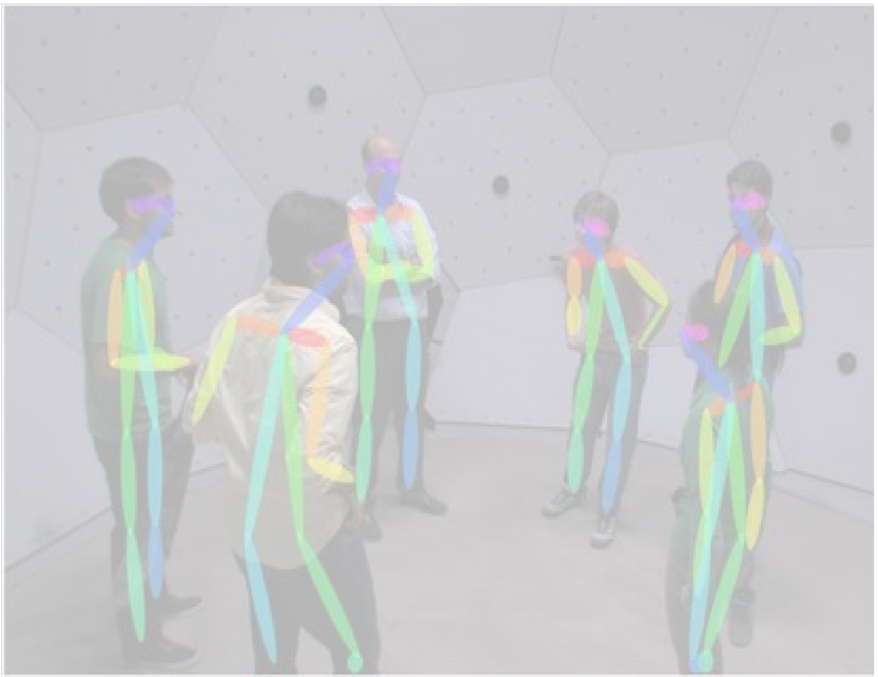
Openpose: Parts Detection + Parts Association



Top-down

Part Affinity Fields

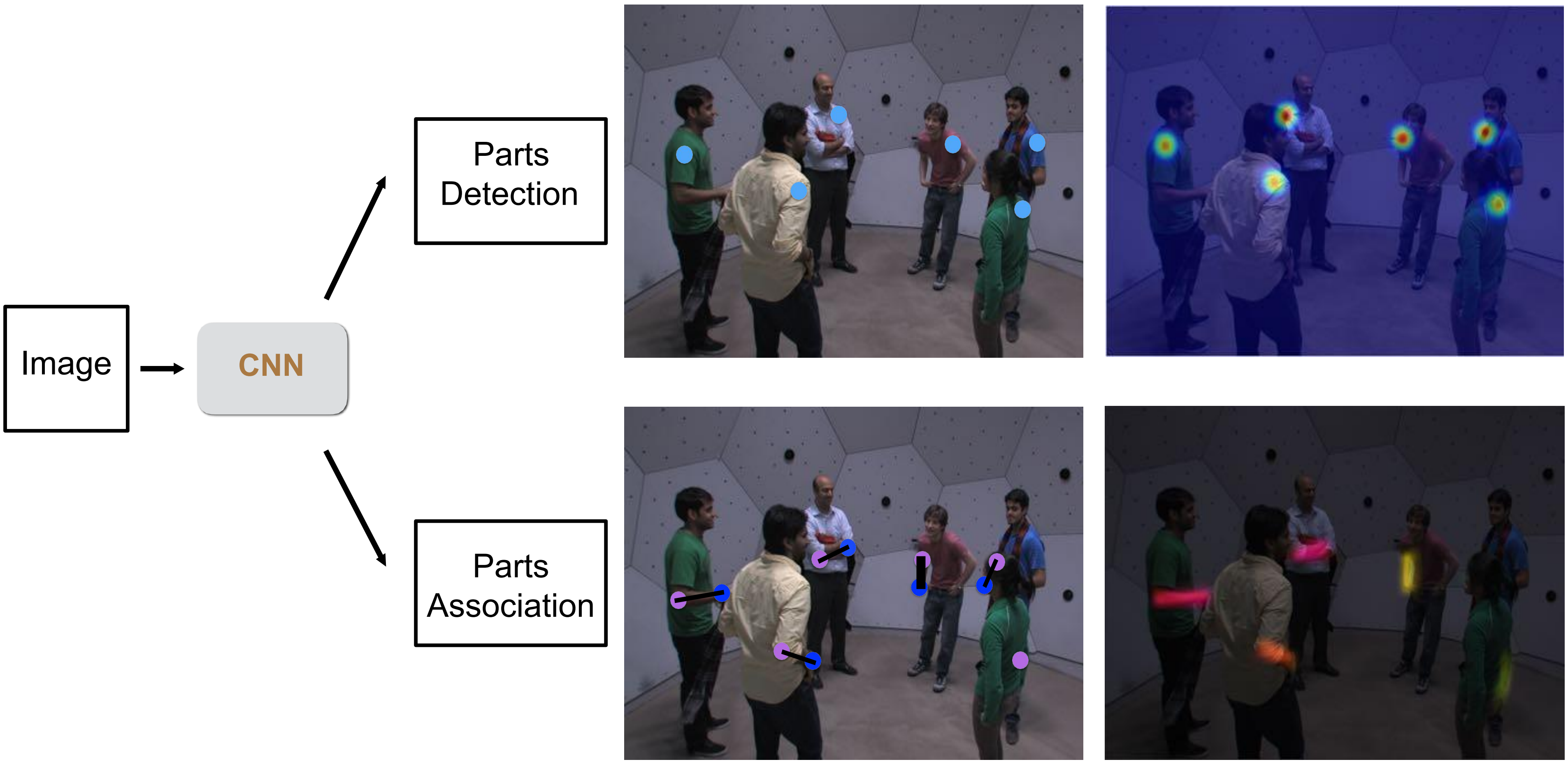
Openpose: Parts Detection + Parts Association



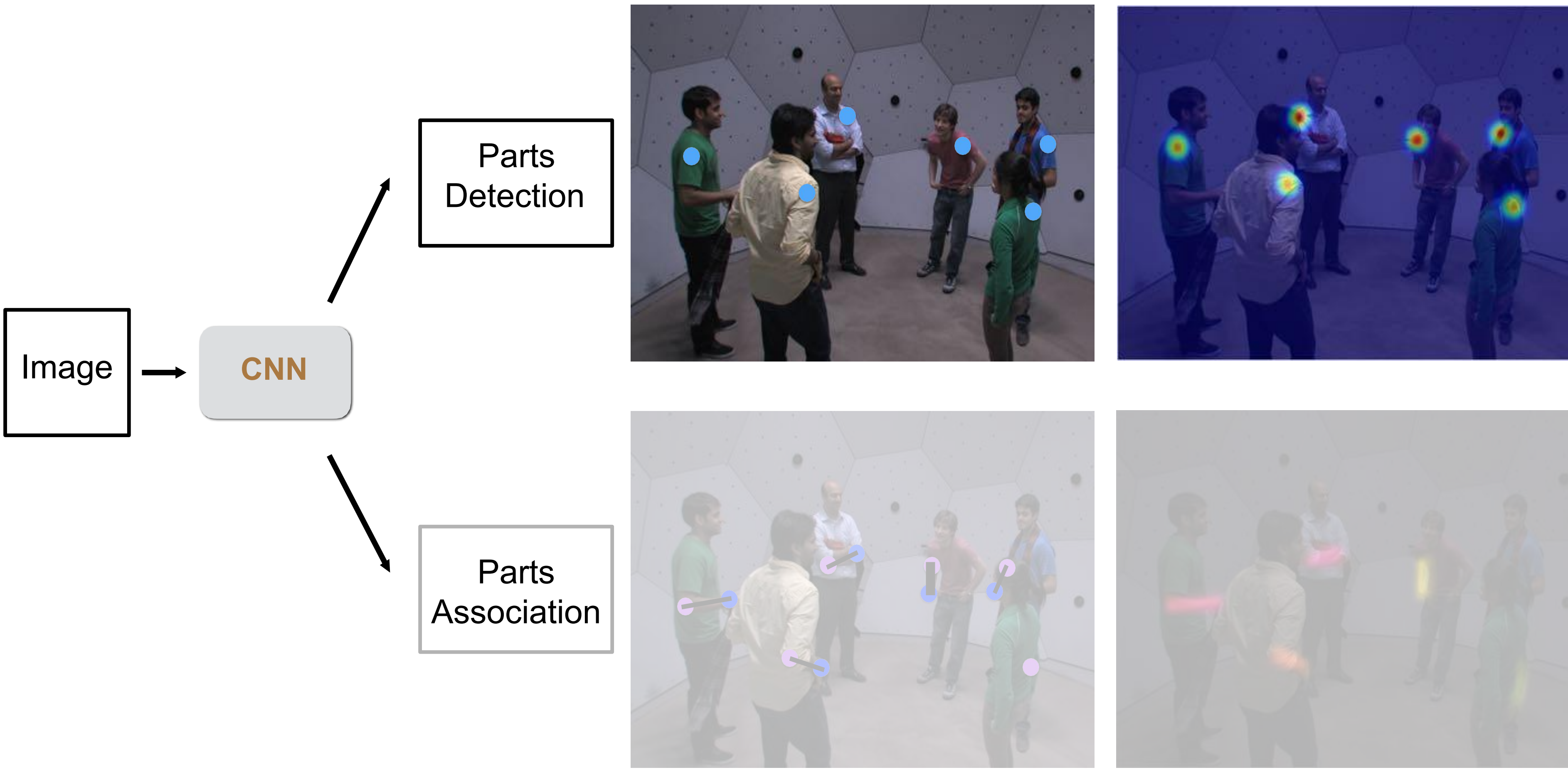
Top-down

Openpose

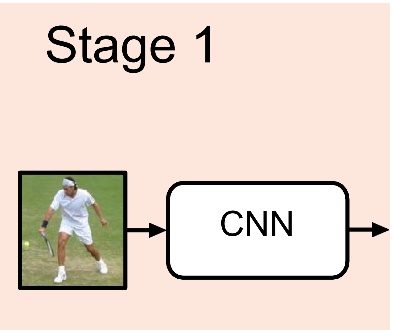
Novelty: Jointly Learning Parts Detection and Parts Association



Novelty: Jointly Learning Parts Detection and Parts Association



Sequential Prediction with Learned Spatial Context



Right shoulder

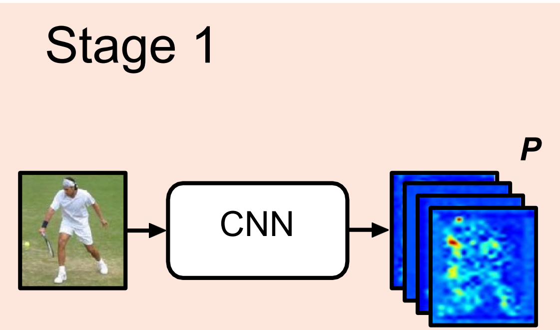
Right wrist

Right knee

⋮

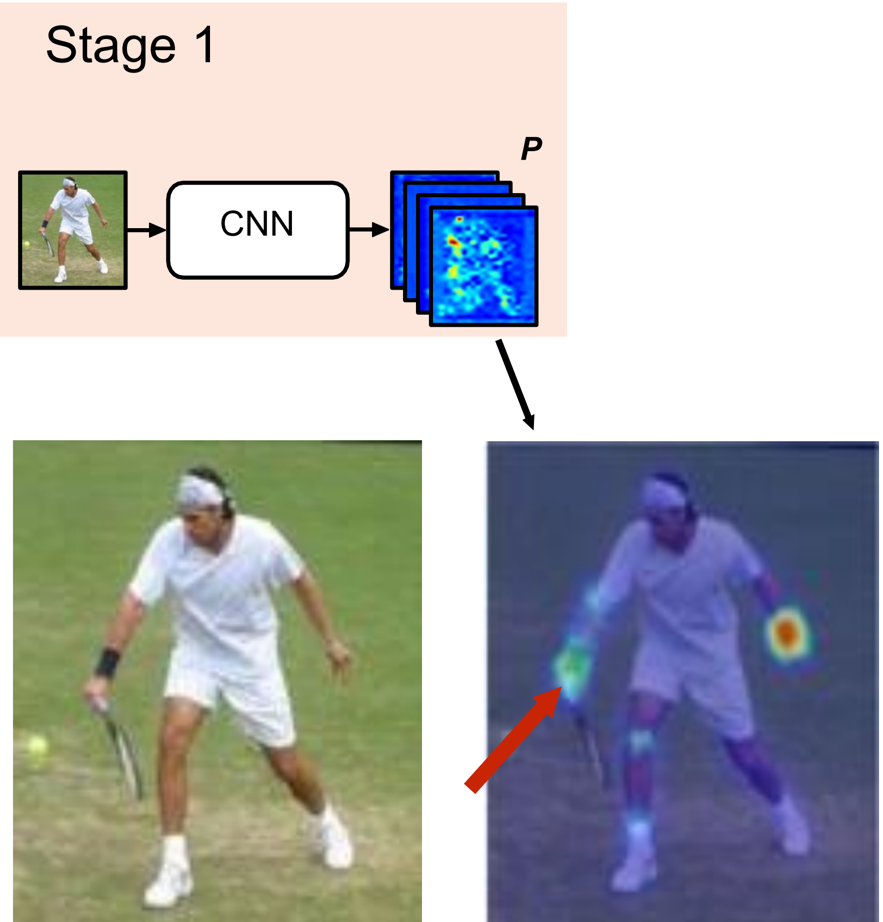
Convolutional Pose Machines, Wei, Ramakrishna, Kanade, Sheikh, CVPR 2016

Sequential Prediction with Learned Spatial Context



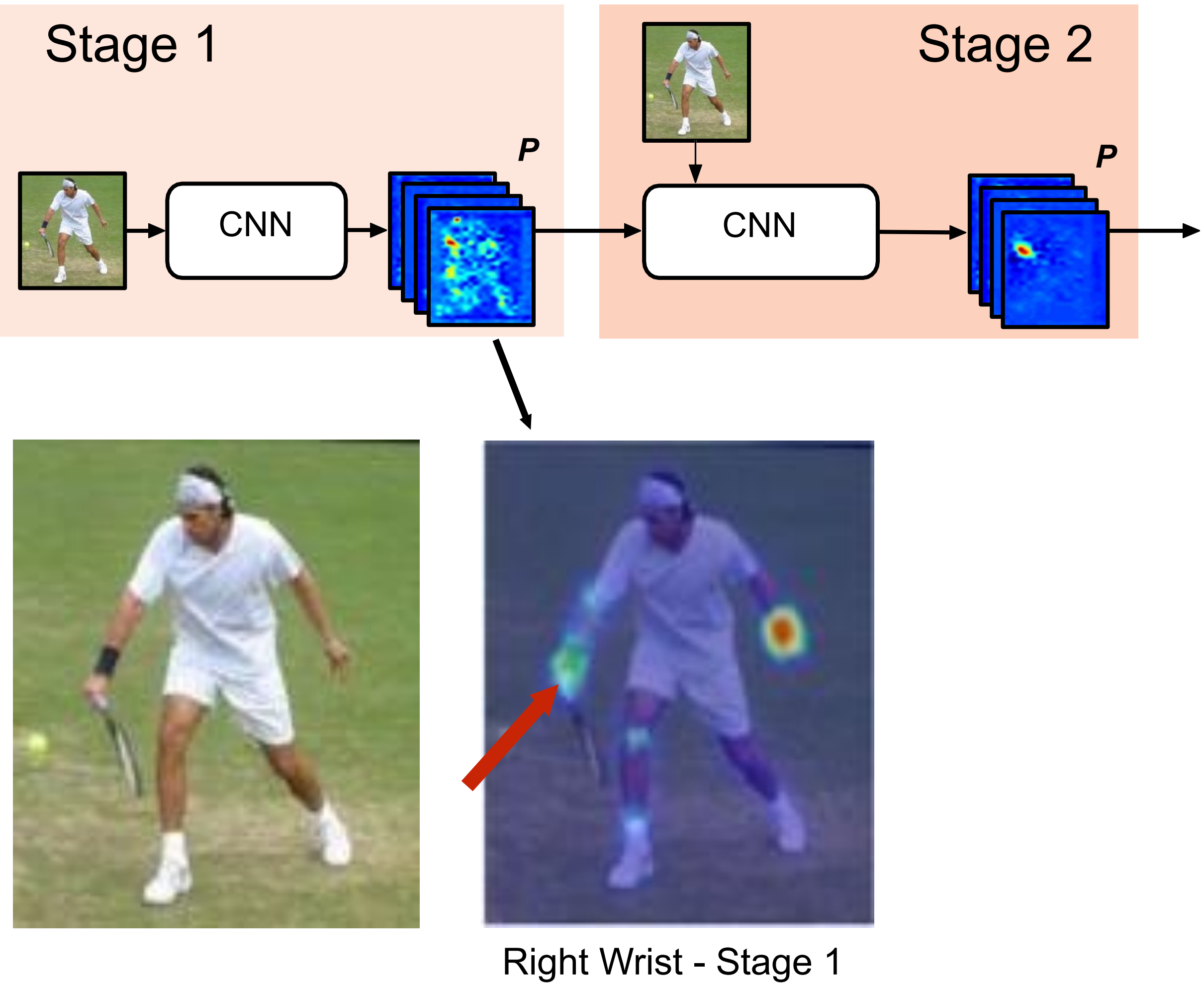
Convolutional Pose Machines, Wei, Ramakrishna, Kanade, Sheikh, CVPR 2016

Sequential Prediction with Learned Spatial Context

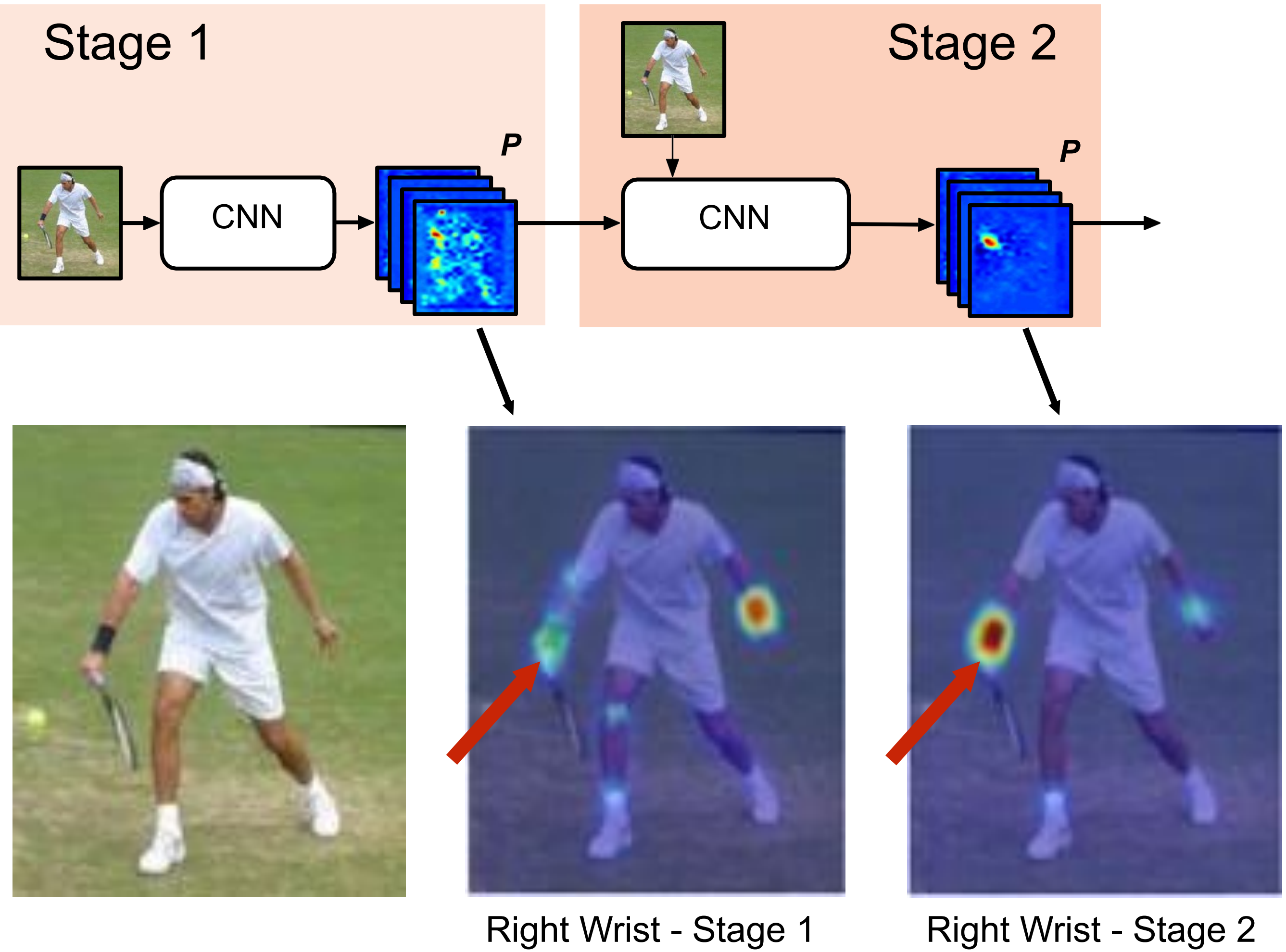


Right Wrist - Stage 1

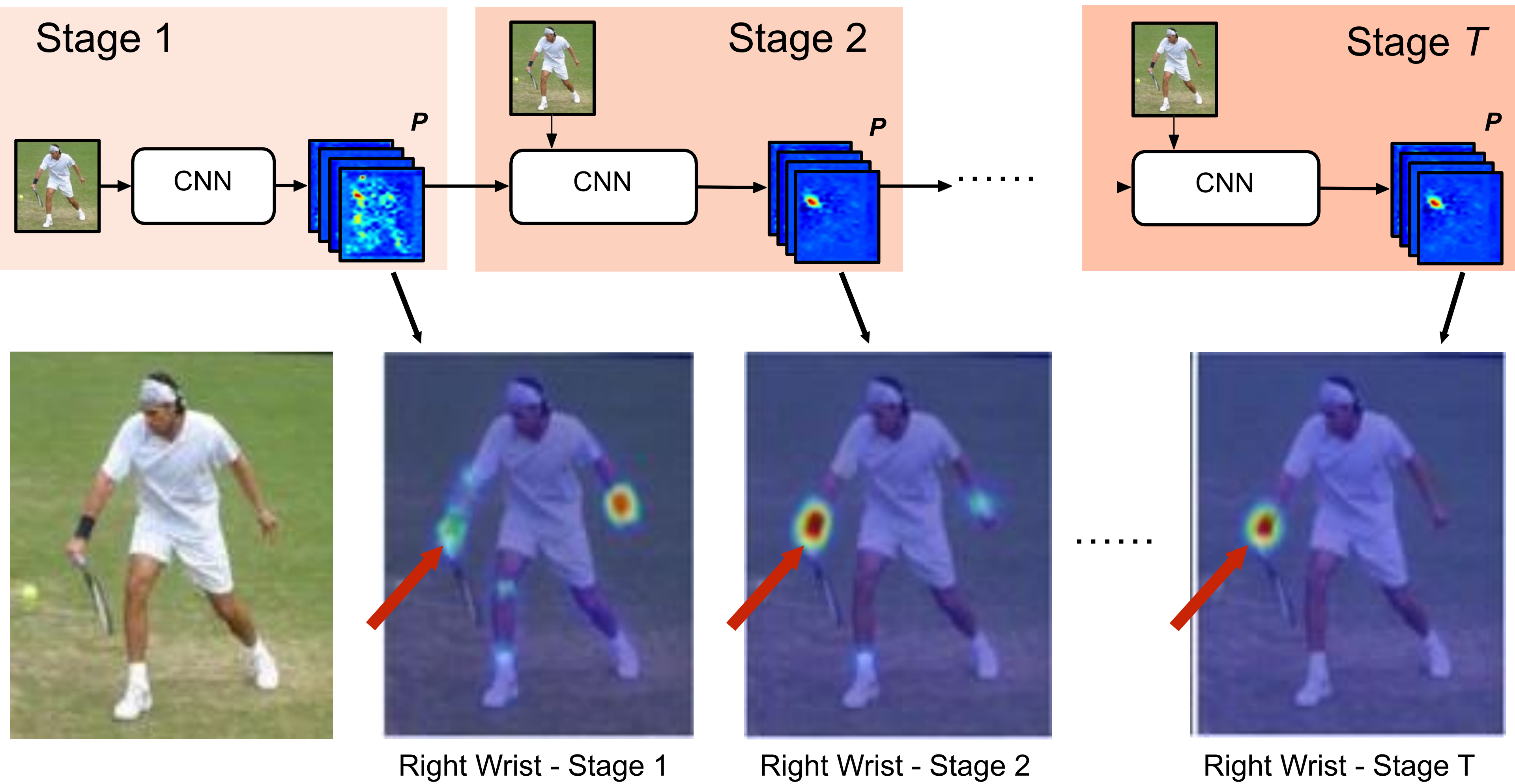
Sequential Prediction with Learned Spatial Context



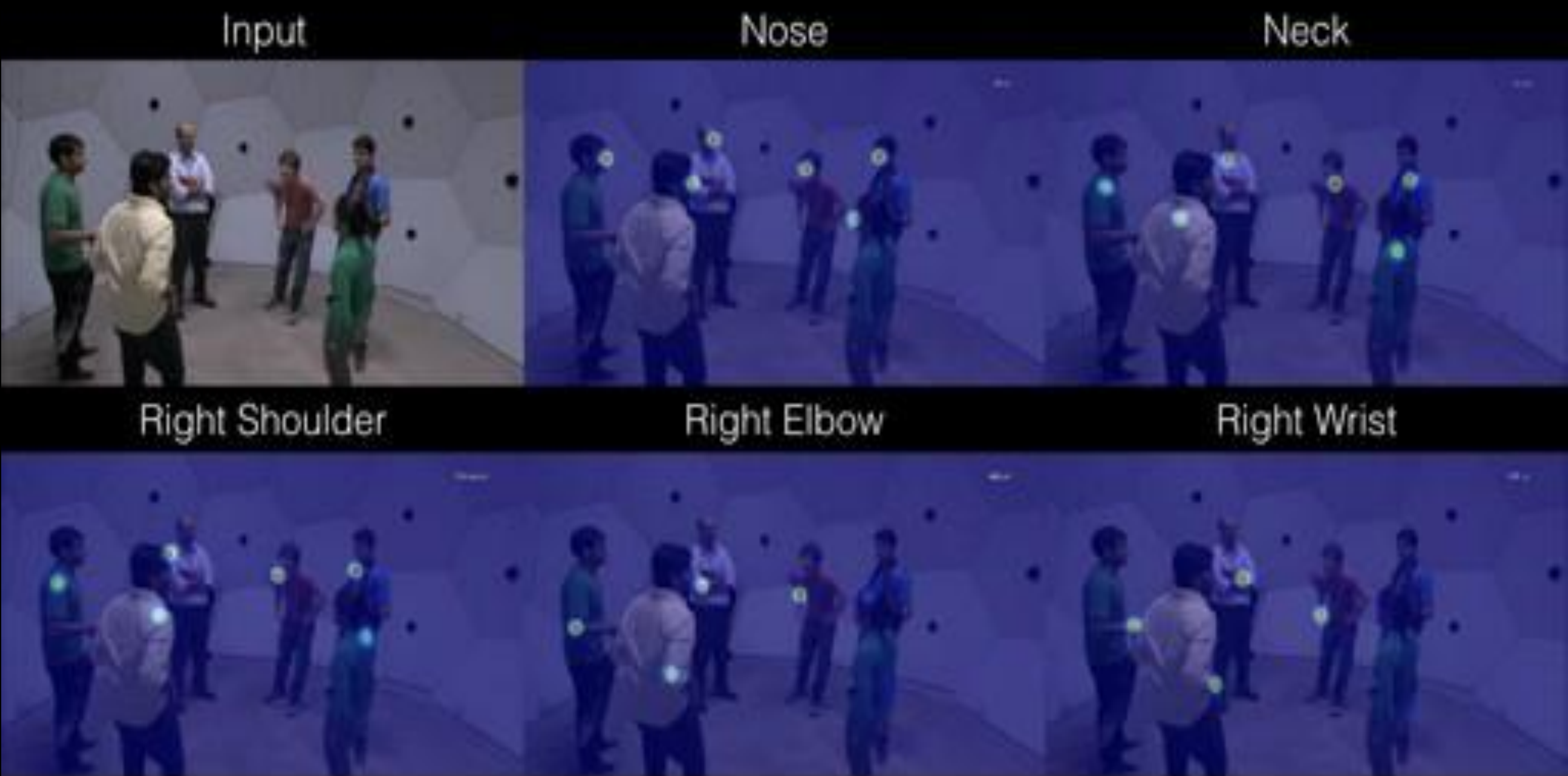
Sequential Prediction with Learned Spatial Context



Sequential Prediction with Learned Spatial Context

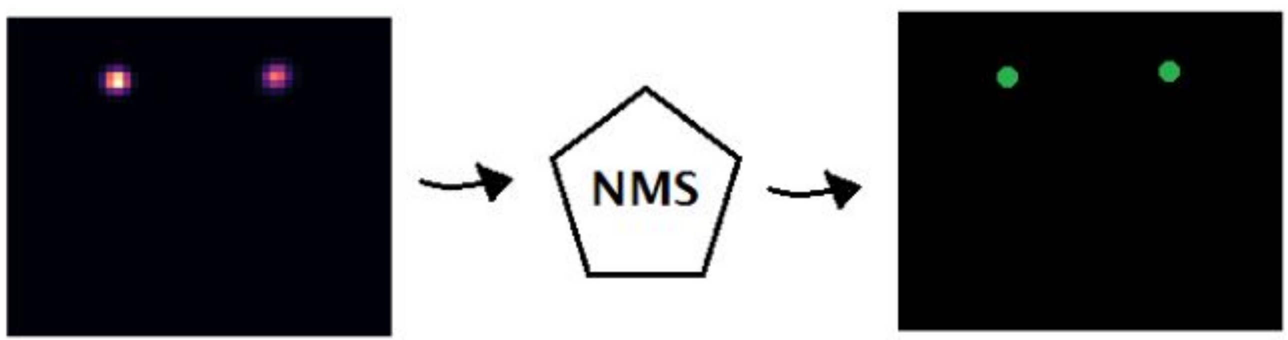


Parts Score Maps Prediction from Image Sequence

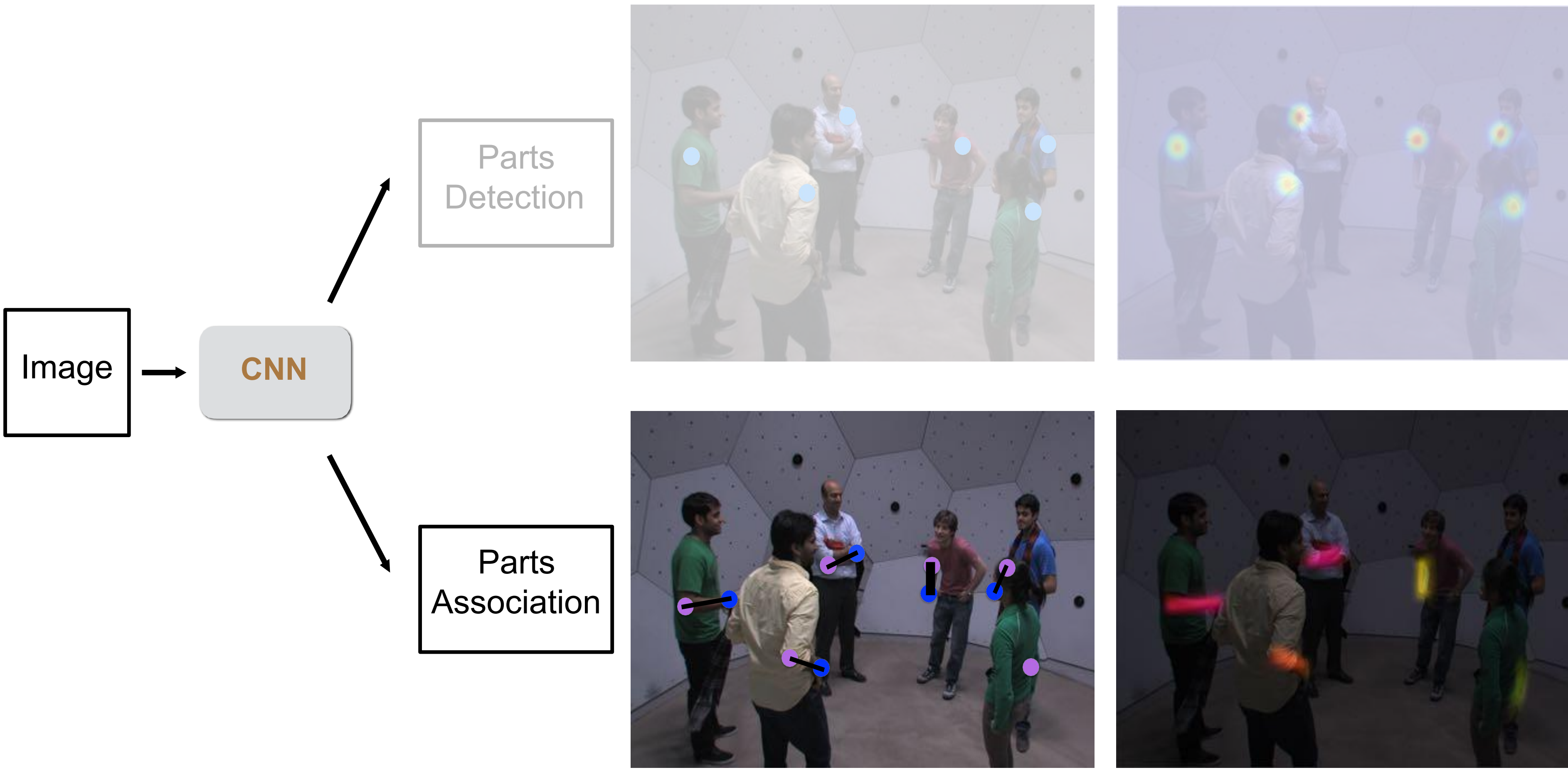


Exact part locations by non maximum suppression

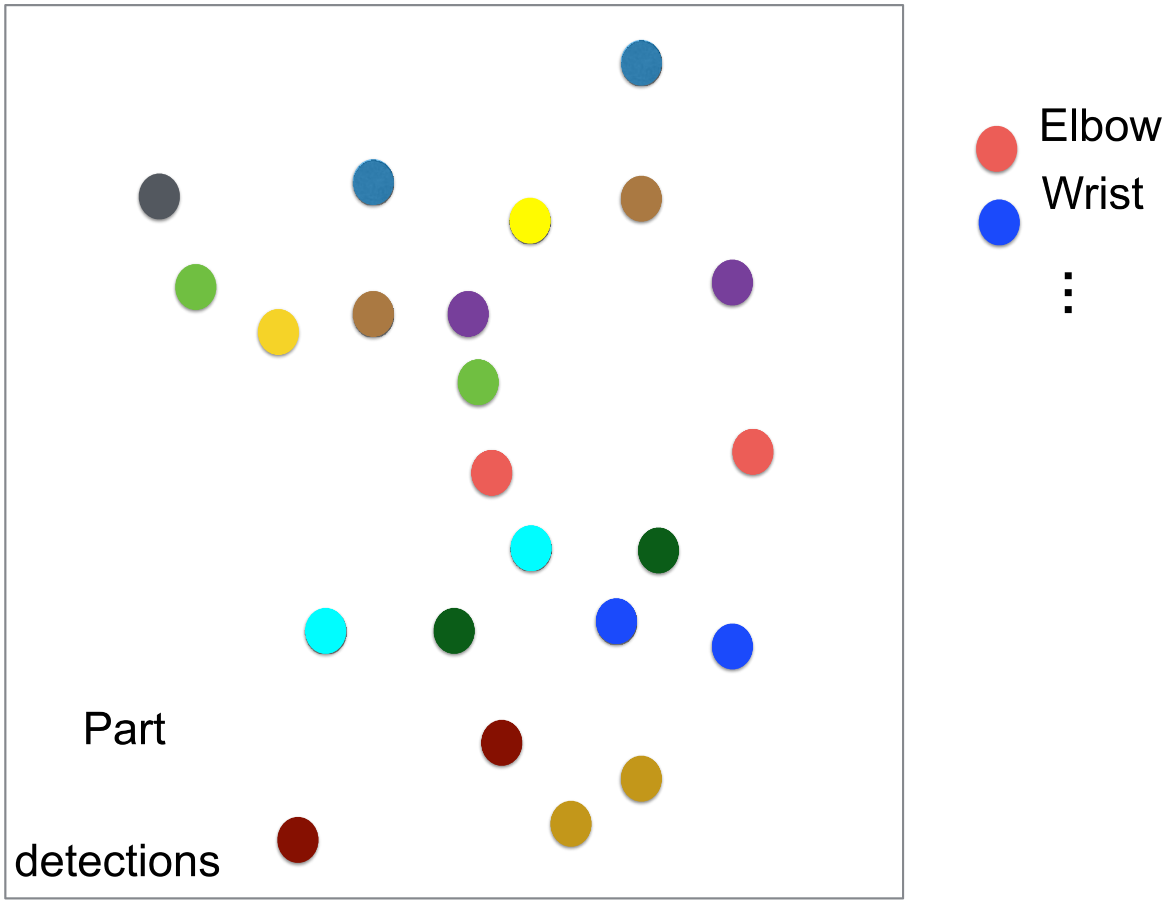
- Pick only local maximums of the heat maps
- Computed by max convolution



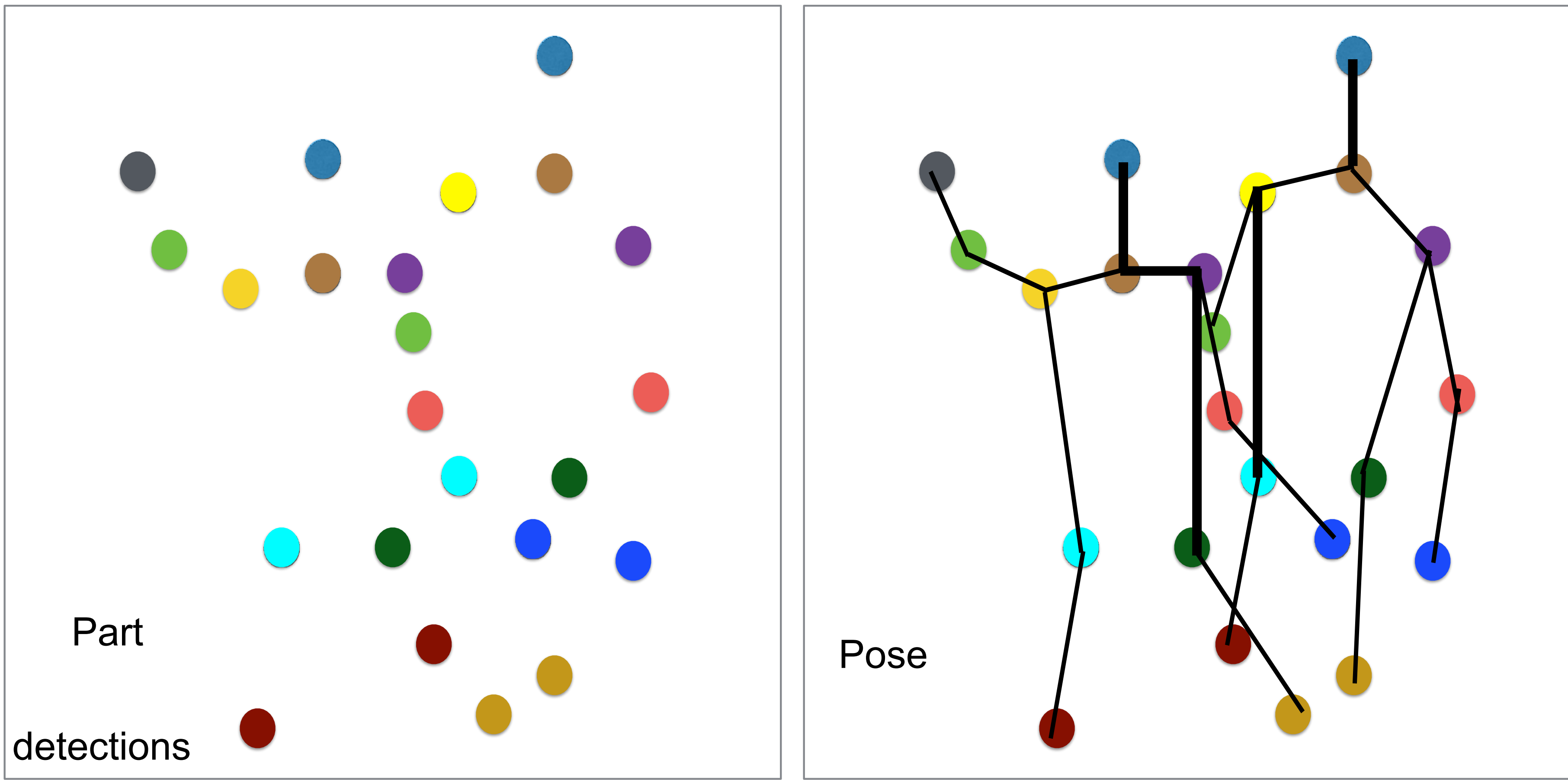
Jointly Learning Parts Detection and Parts Association



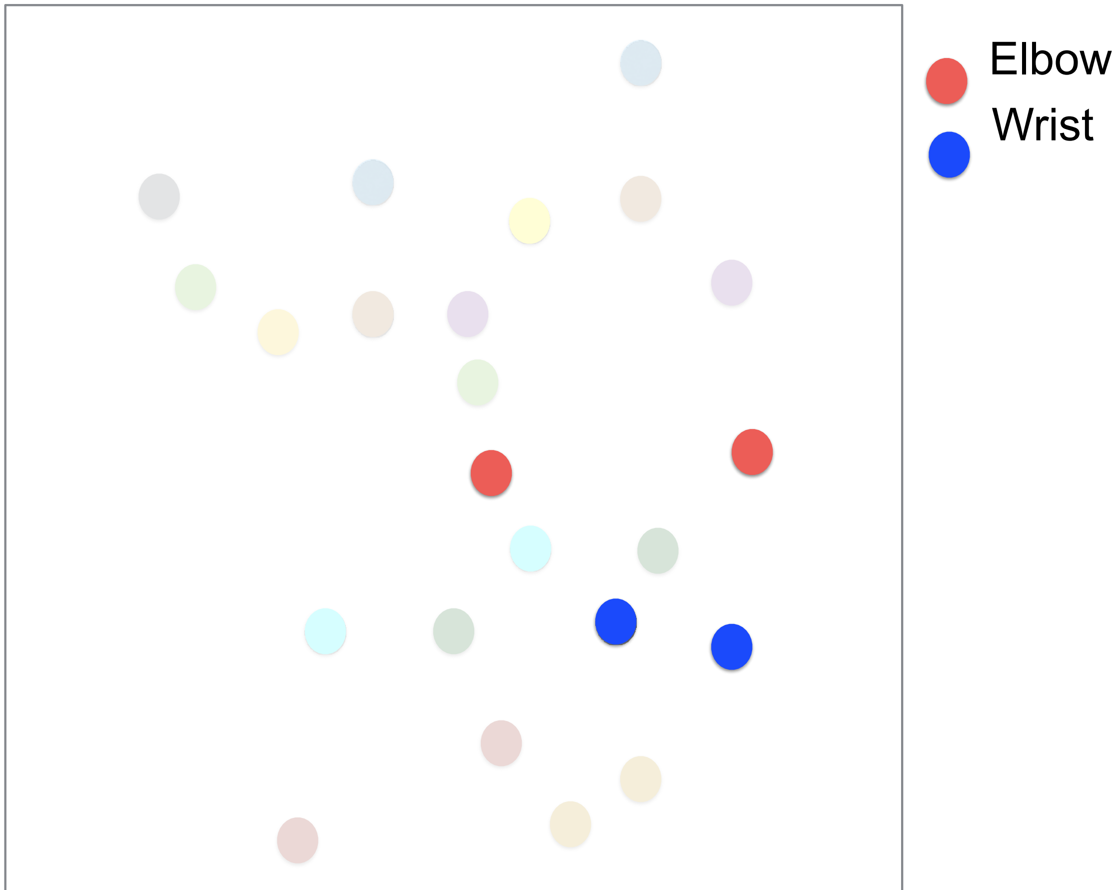
Part-Person Association for Multi-Person Pose Estimation



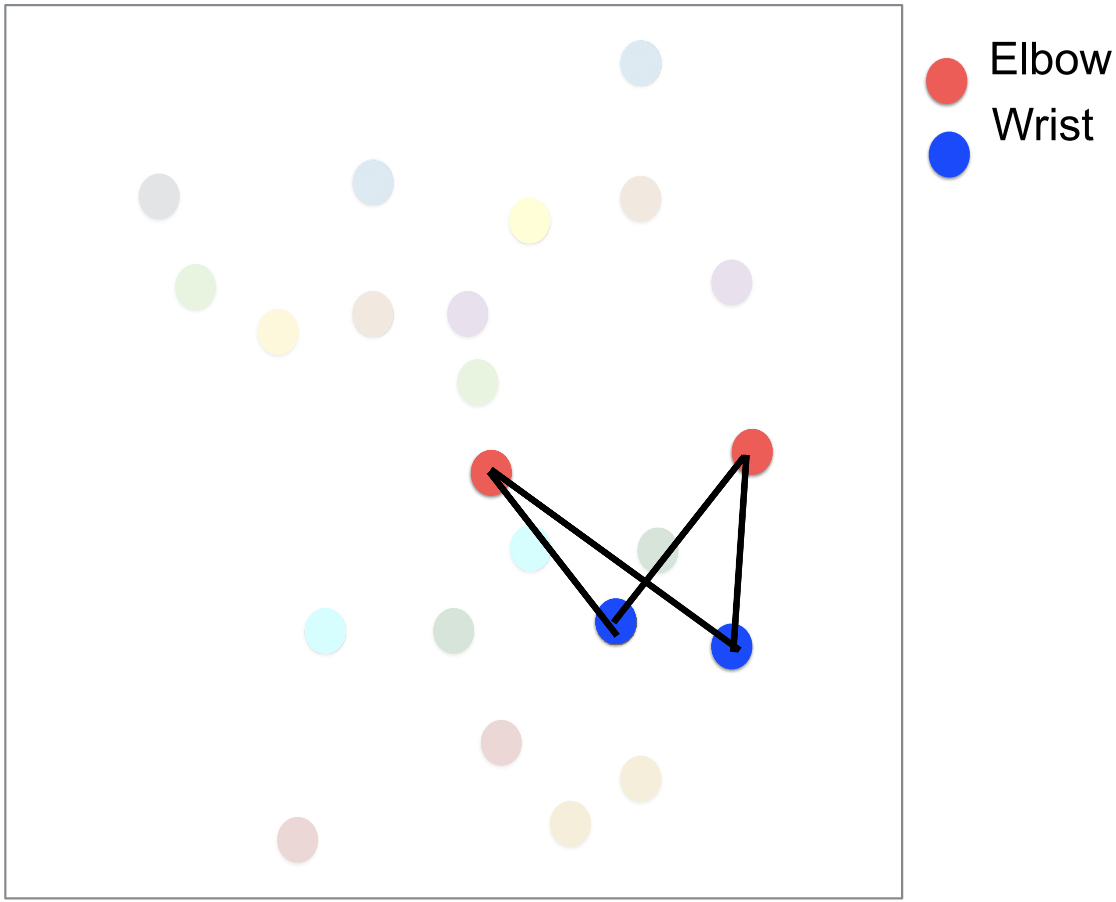
Part-Person Association for Multi-Person Pose Estimation



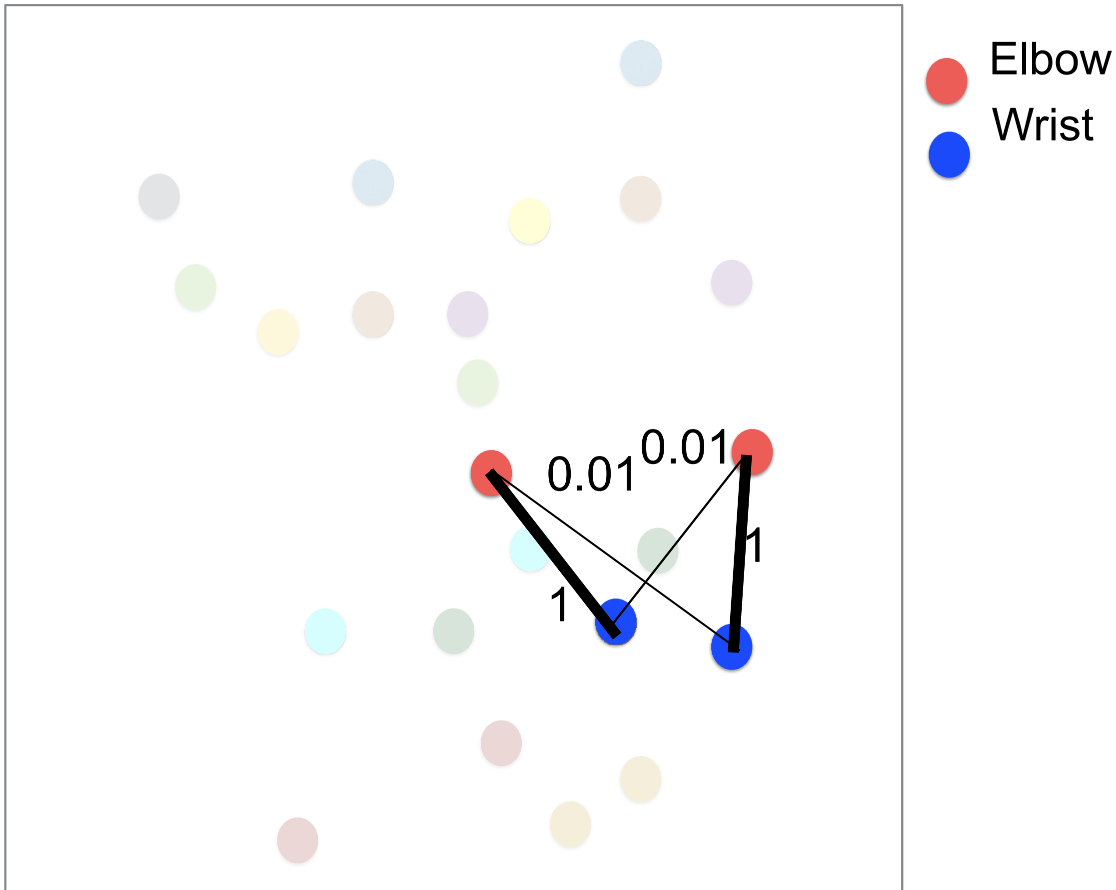
Part-to-Part Association for Multi-Person Pose Estimation



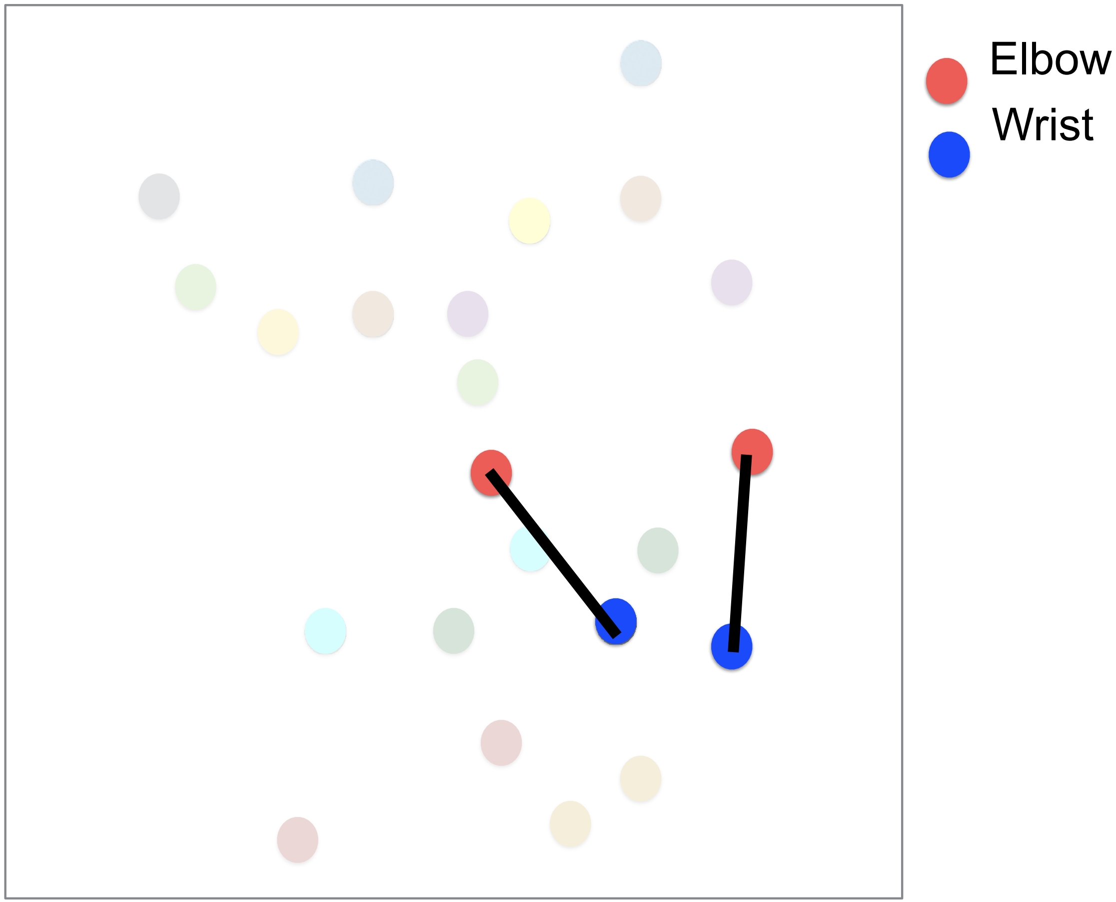
Part Affinity Score Guides the Connection



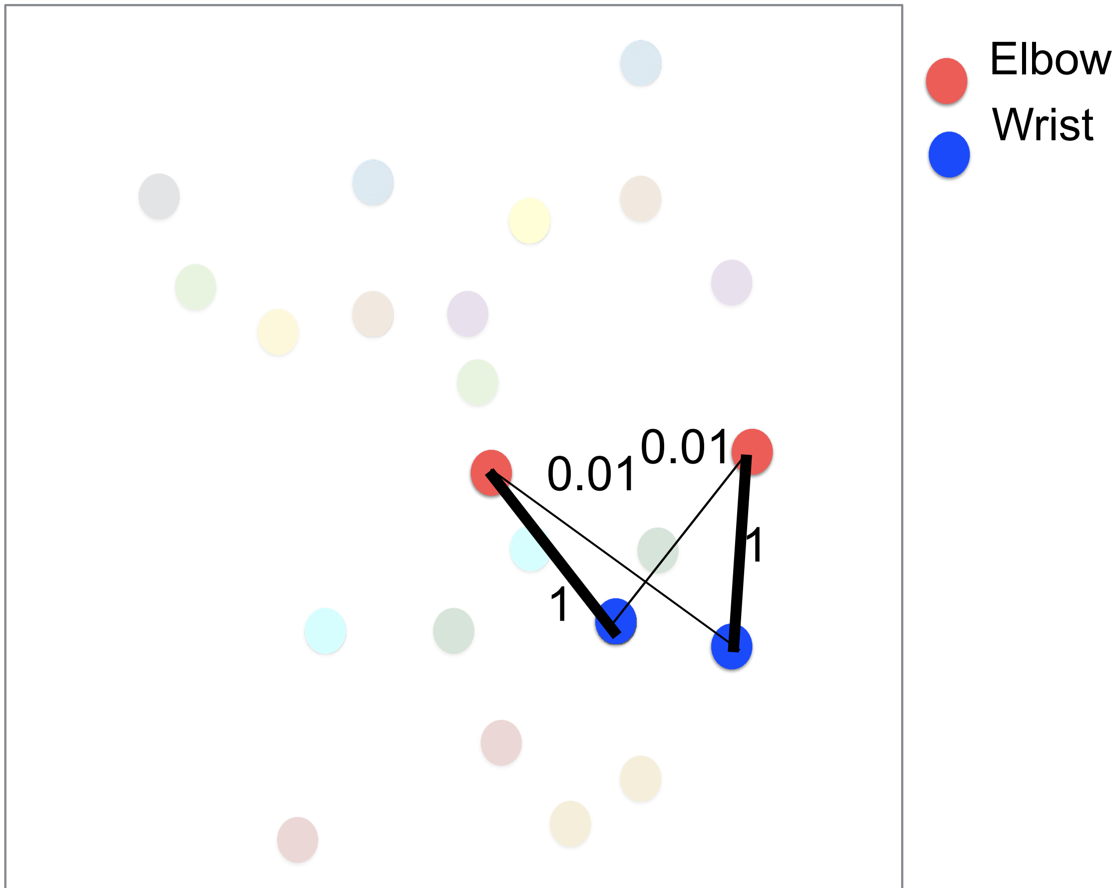
Part Affinity Score Guides the Connection



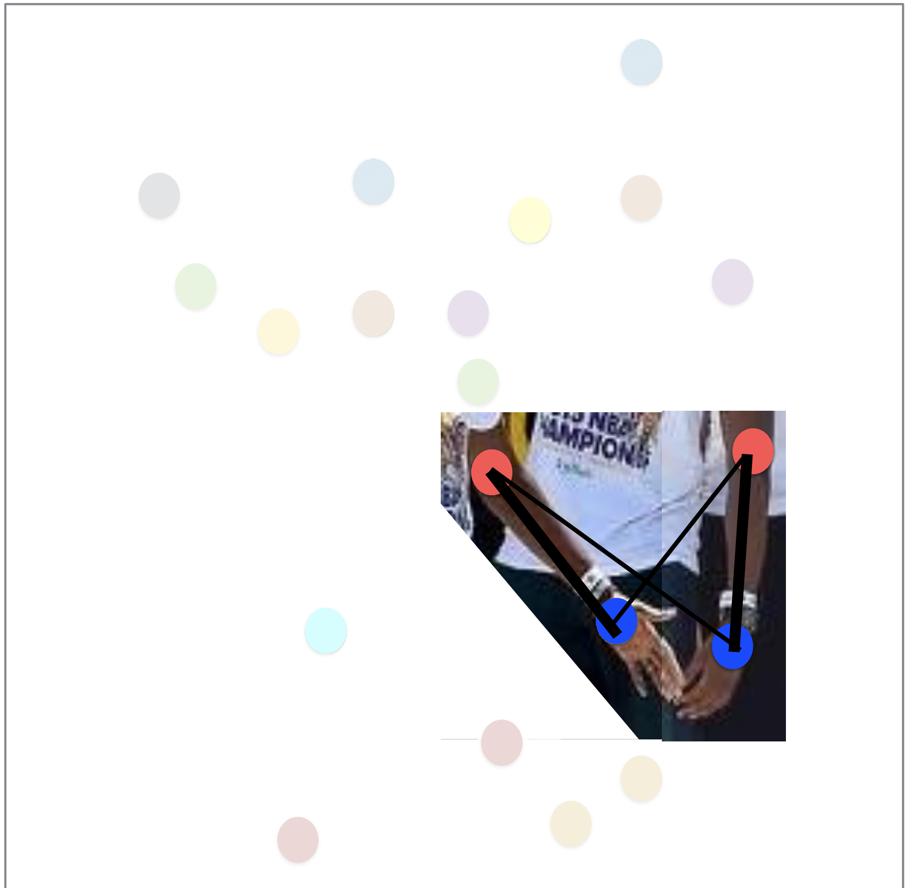
Part Affinity Score Guides the Connection



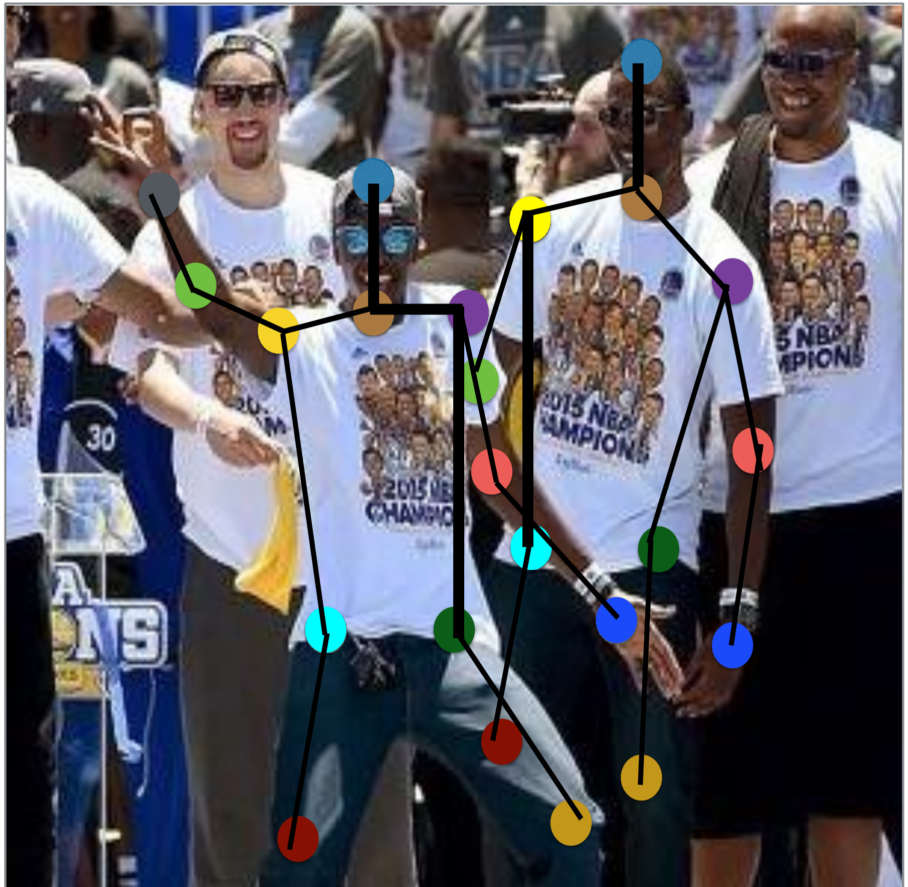
How to Obtain the Part Affinity Score



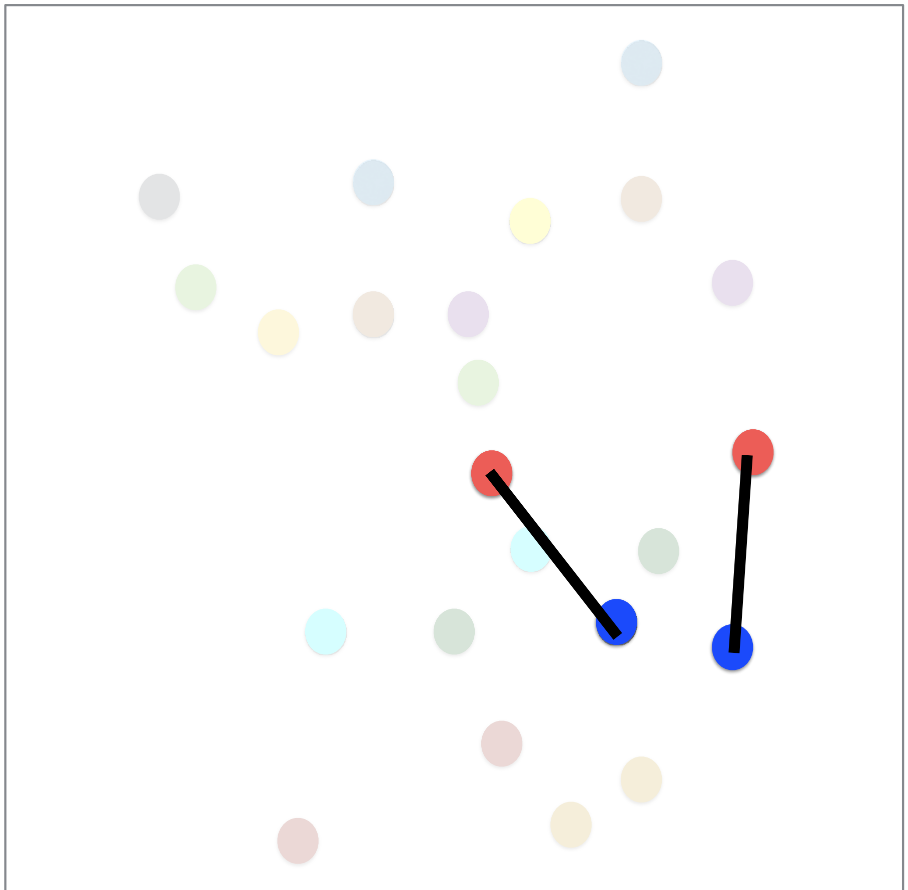
Part Affinity Score is Dependent on Visual Appearance

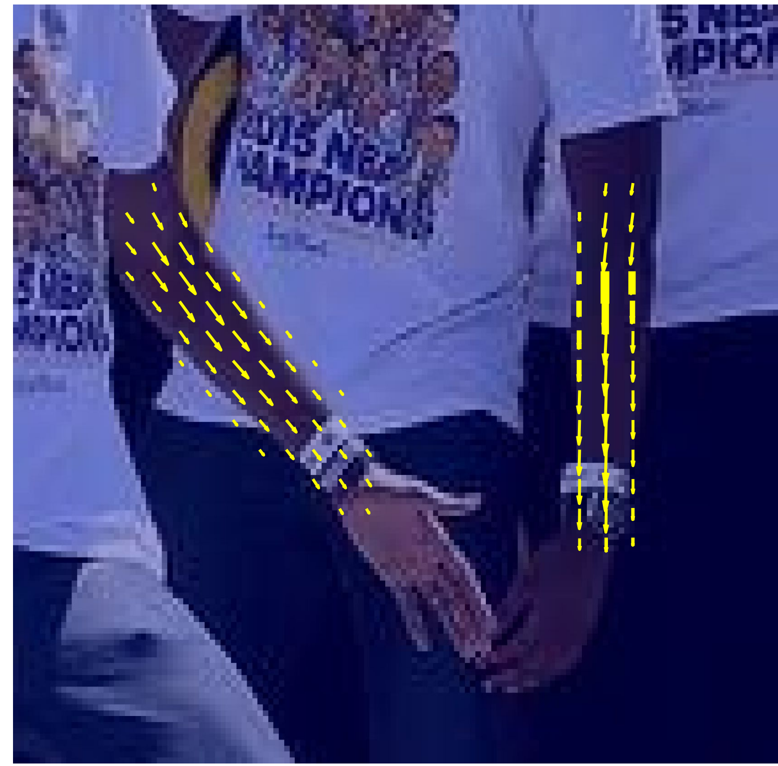


Part Affinity Score is Dependent on Visual Appearance



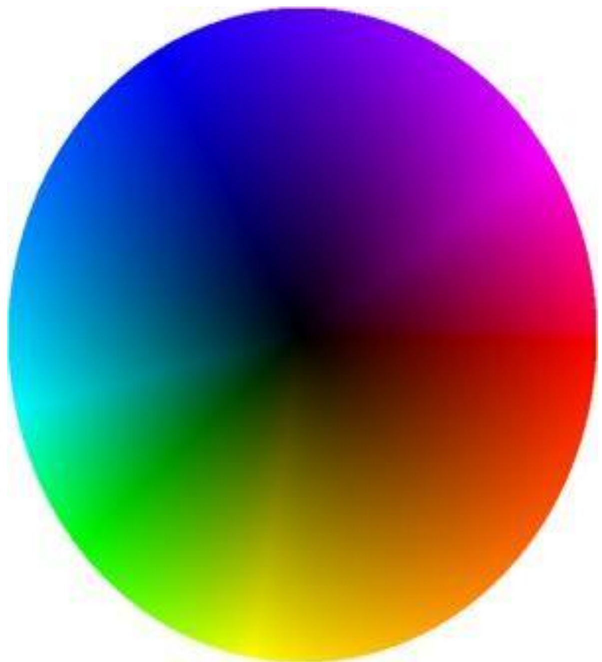
Key Idea: Encode the Part Affinity Score on the Image Plane





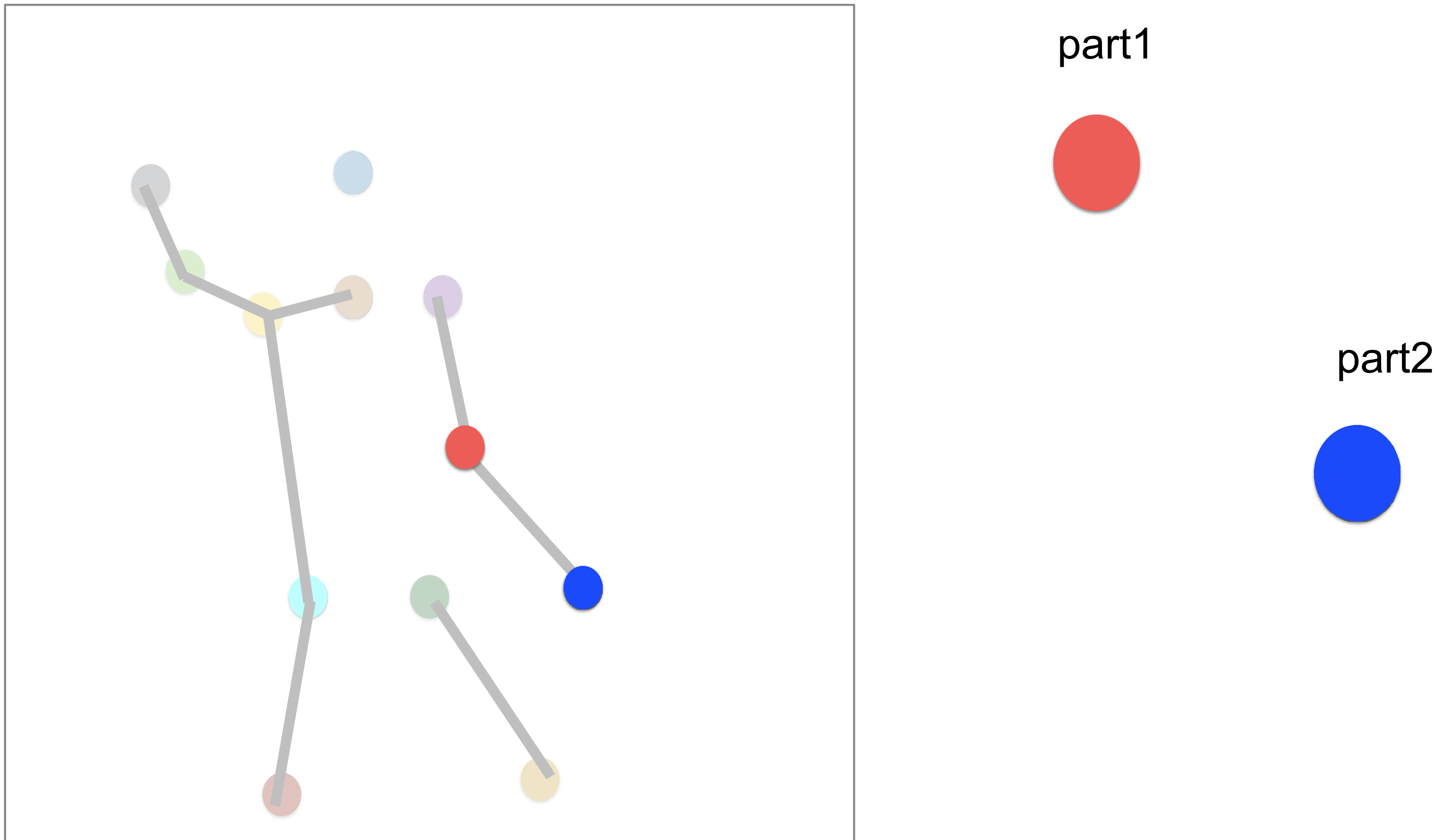
Part Affinity Fields
encode **direction** and **position**

Key Idea: Encode the Part Affinity Score on the Image Plane

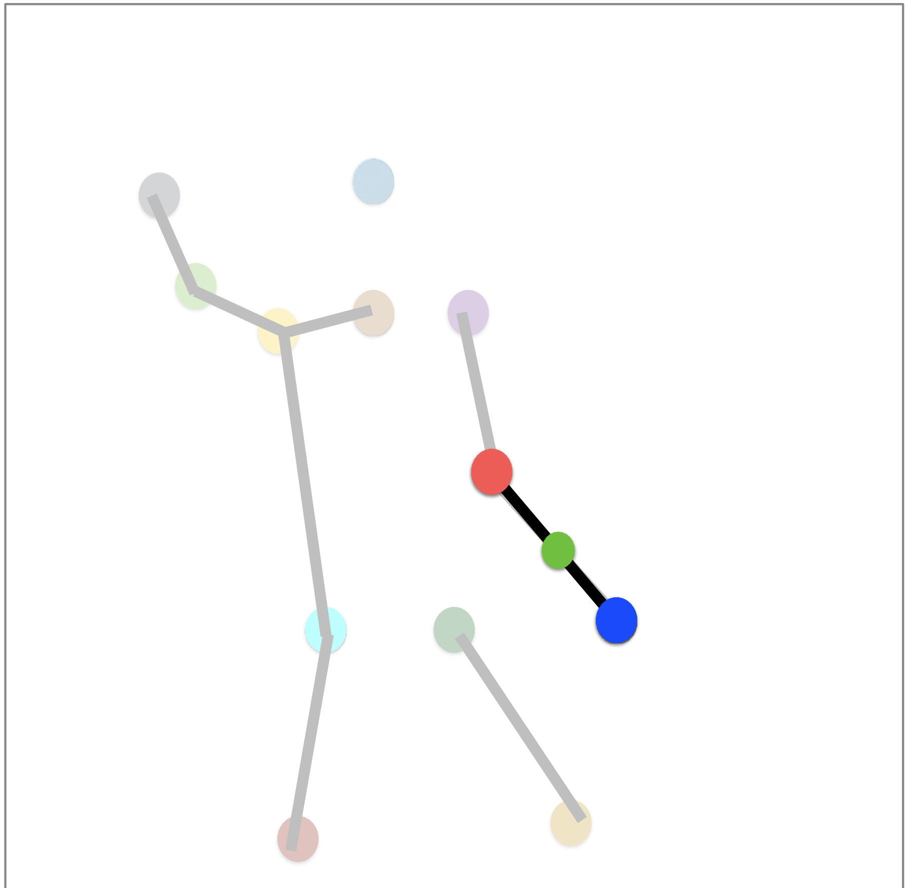


Part Affinity Fields
encode **direction** and **position**

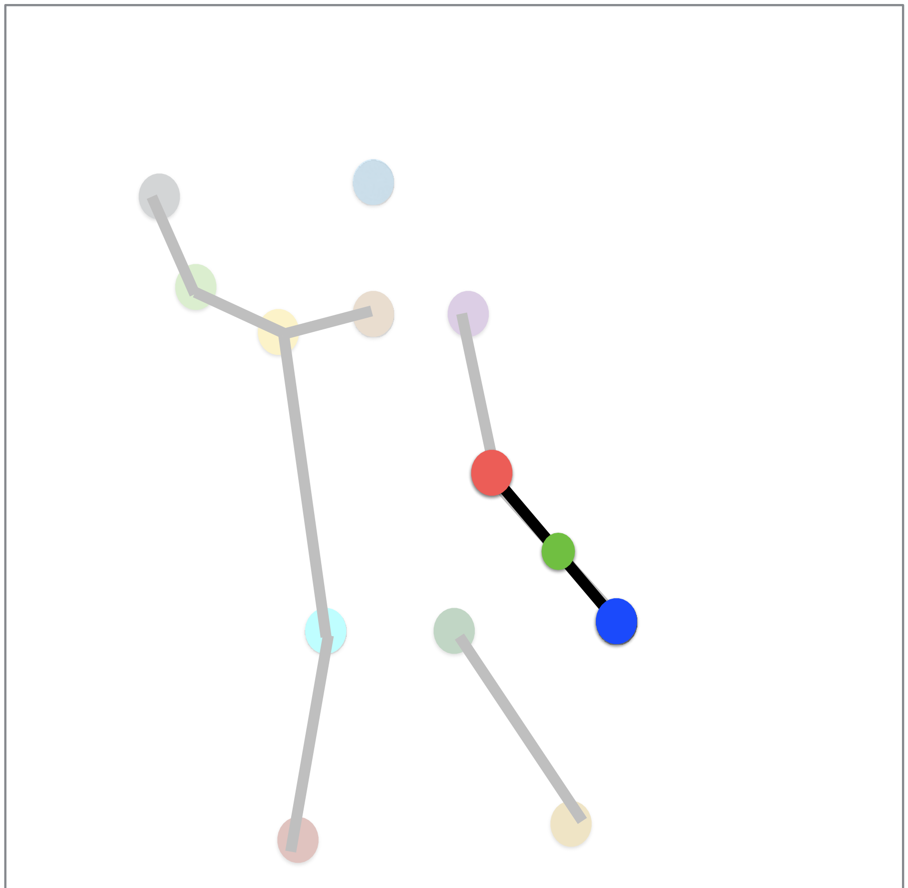
Midpoint Score Map for Part-to-Part Association

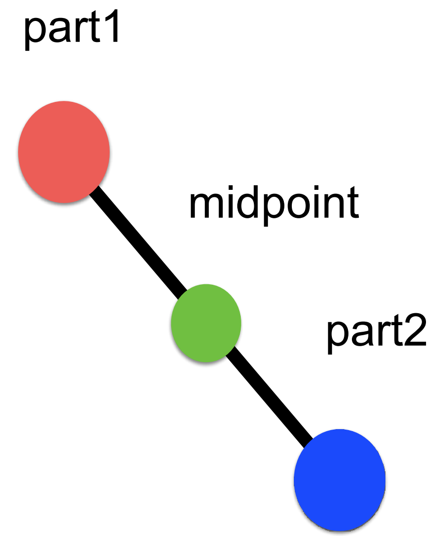


Midpoint Score Map for Part-to-Part Association



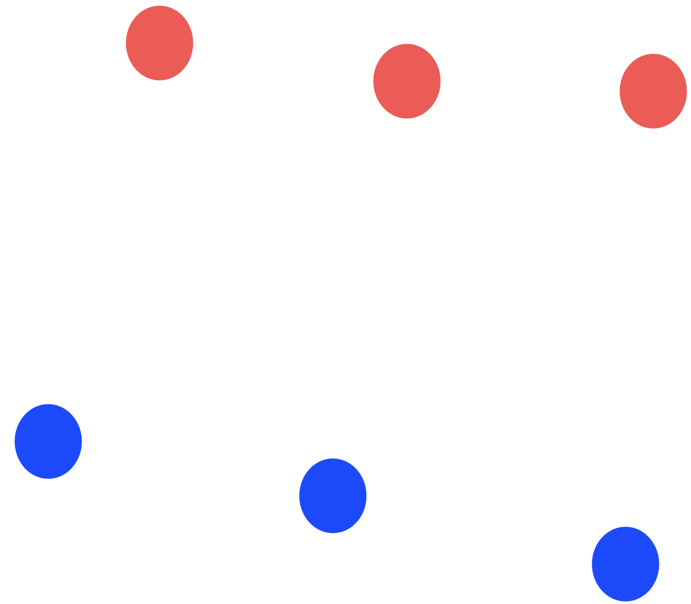
Midpoint Score Map for Part-to-Part Association





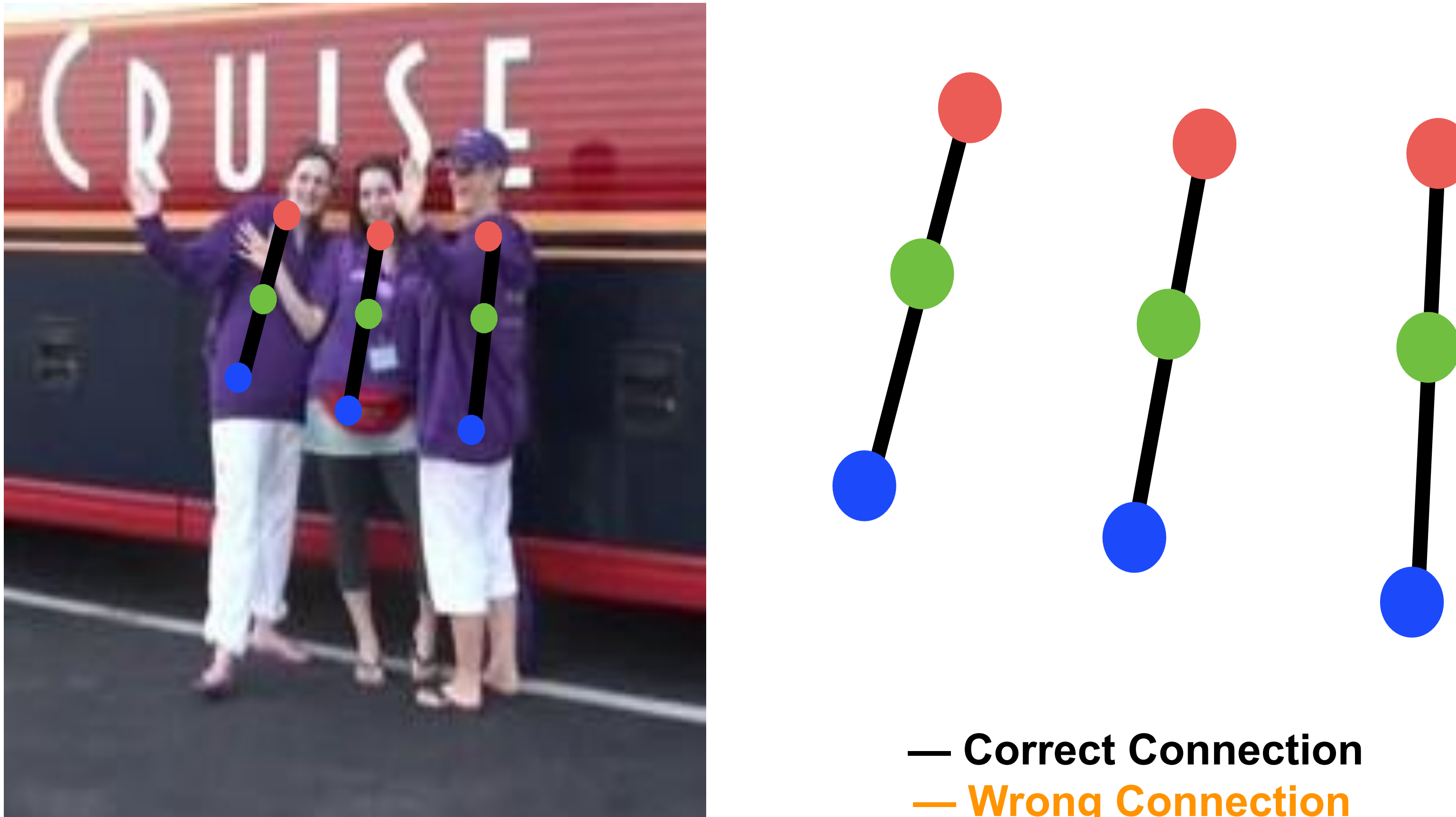
Affinity score between part1 and part2
= confidence score of the midpoint

Spatial Ambiguity of the Midpoint Representation

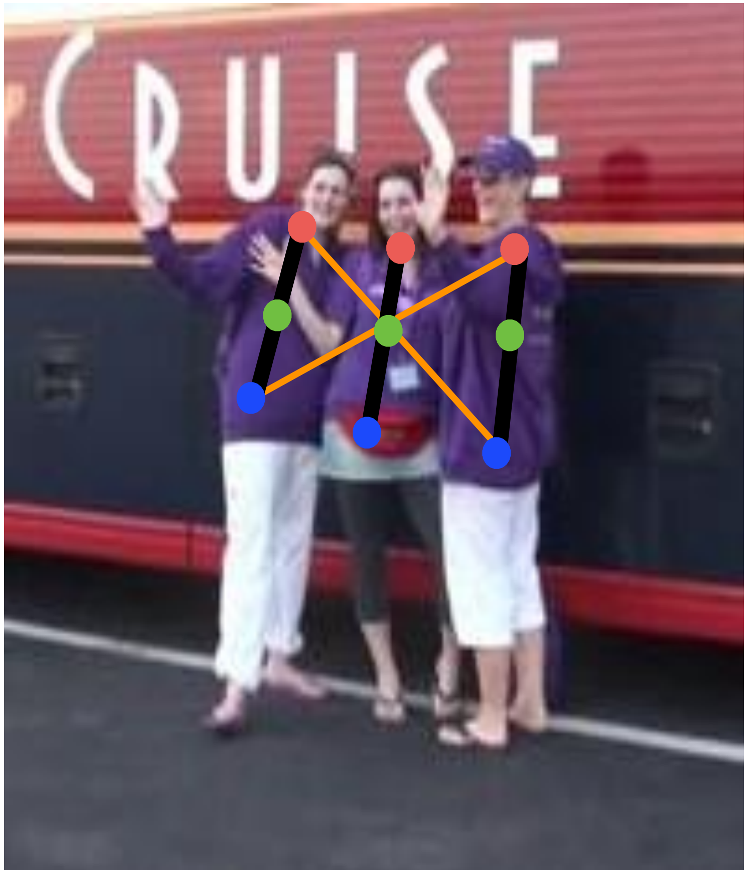


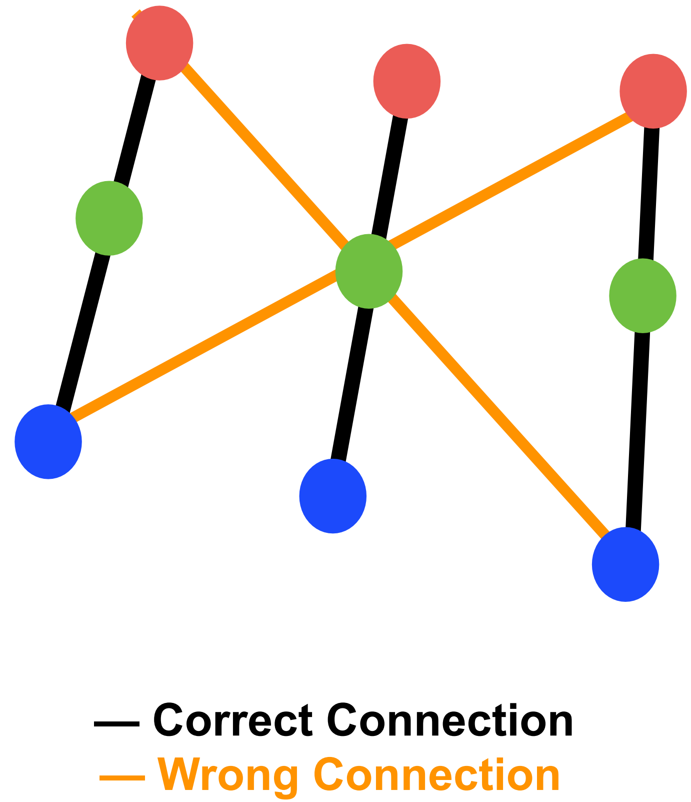
— Correct Connection

Spatial Ambiguity of the Midpoint Representation

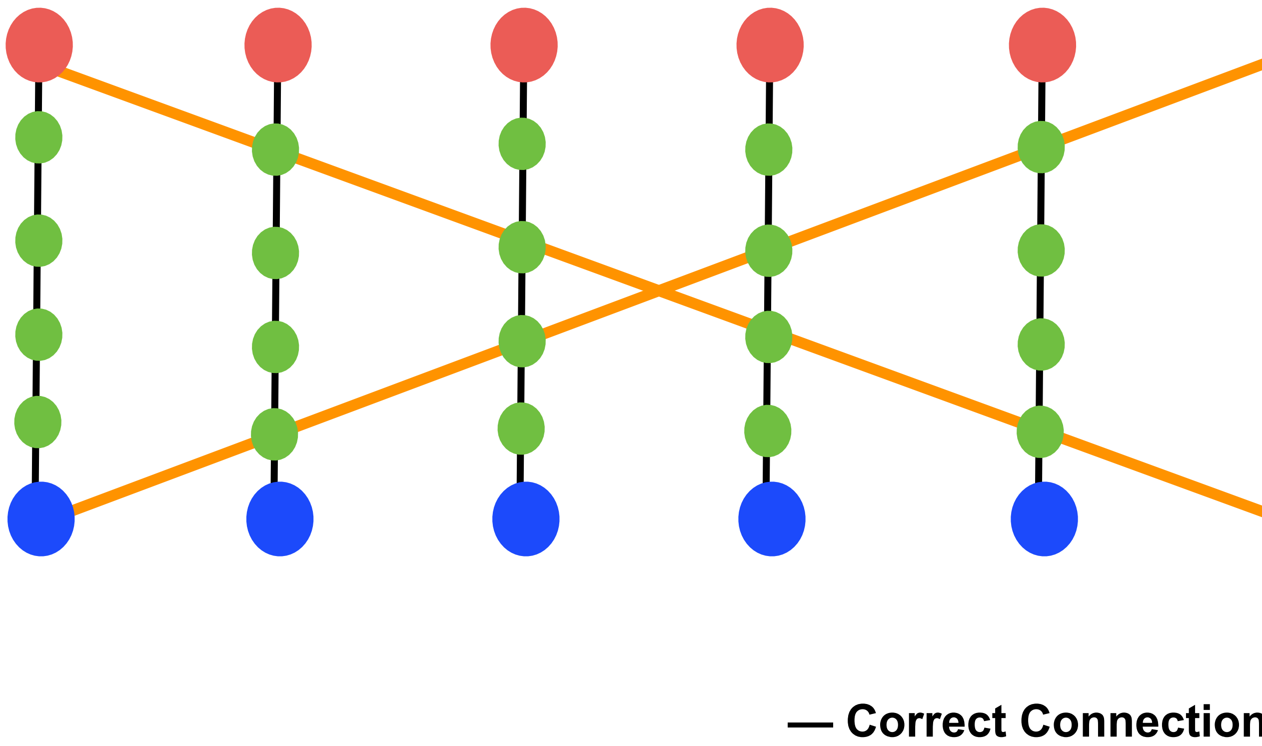


Spatial Ambiguity of the Midpoint Representation

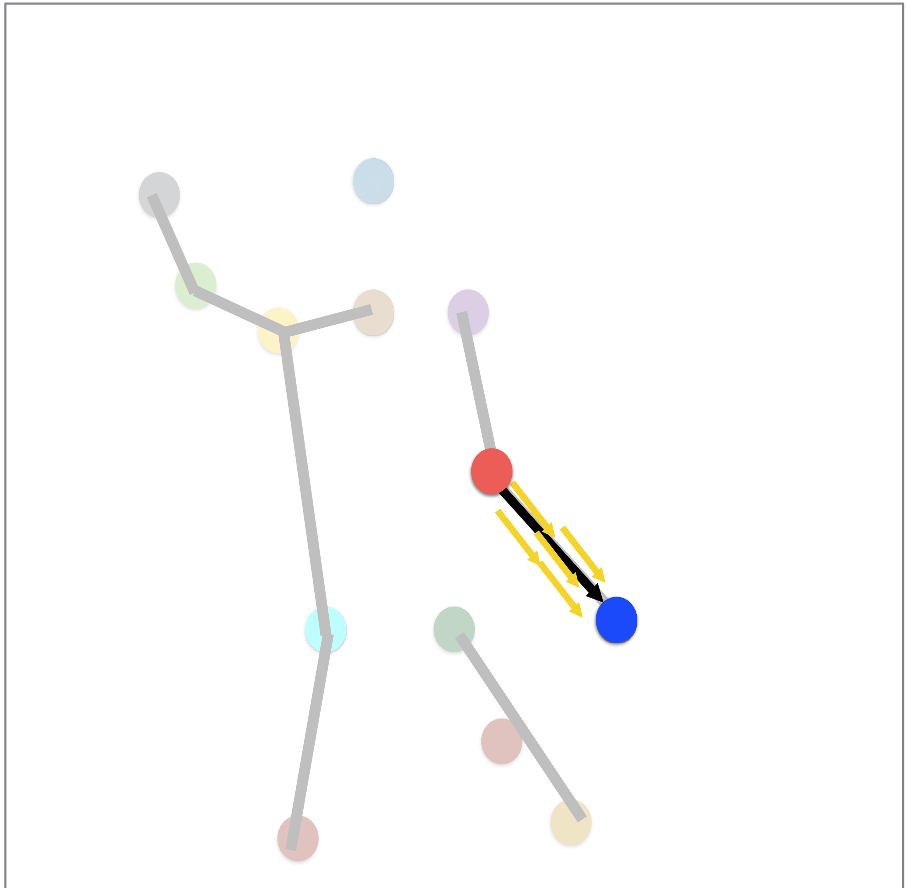


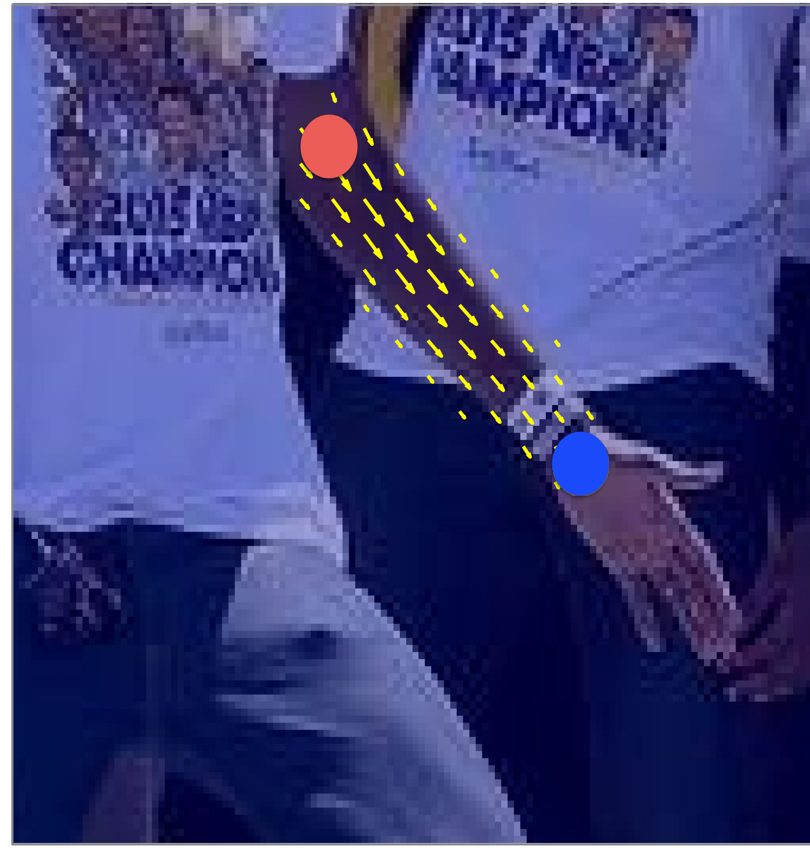


Increasing Midpoint Number Cannot Solve The Problem

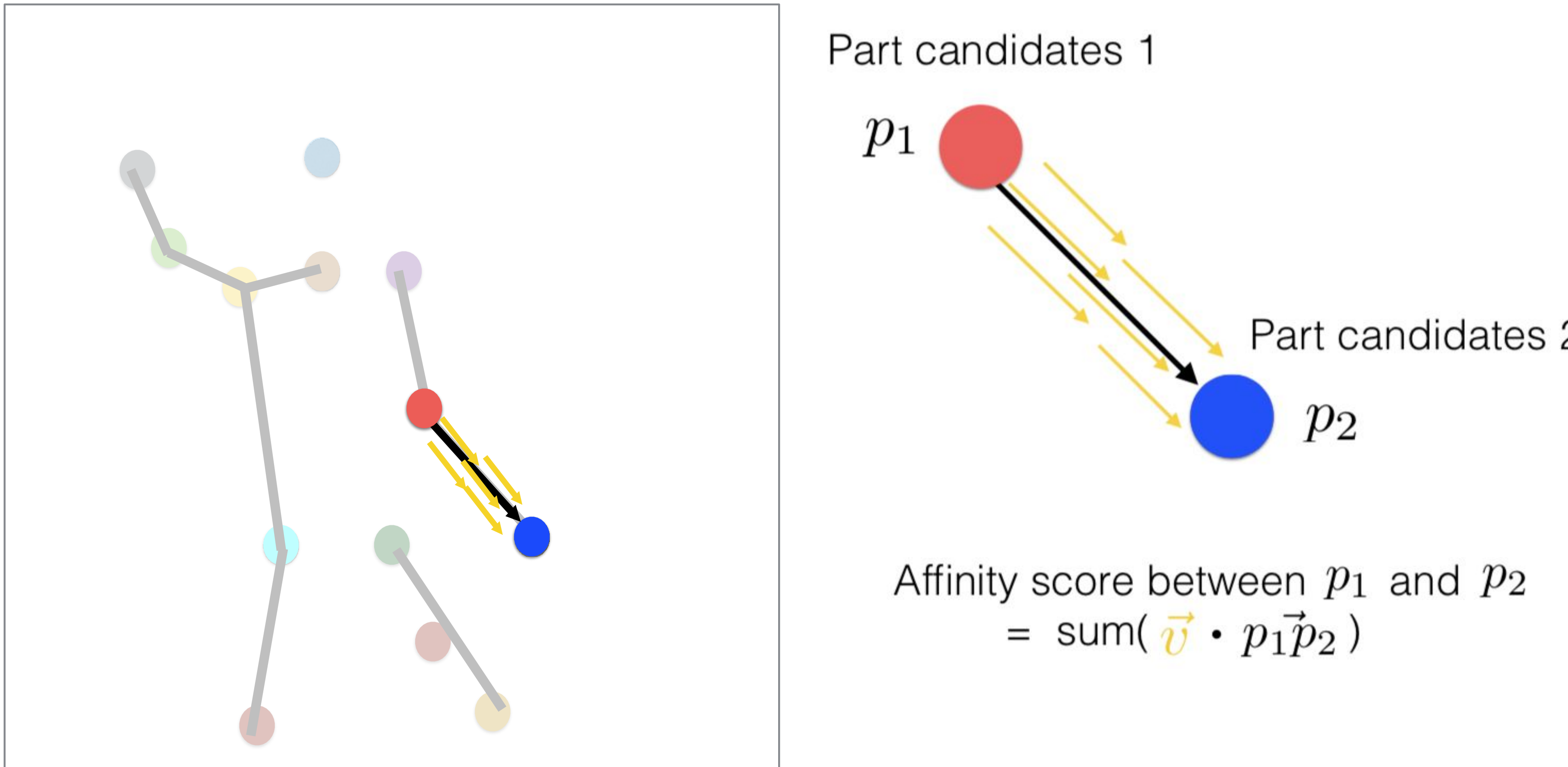


Part Affinity Fields for Part-to-Part Association





Part Affinity Fields for Part-to-Part Association

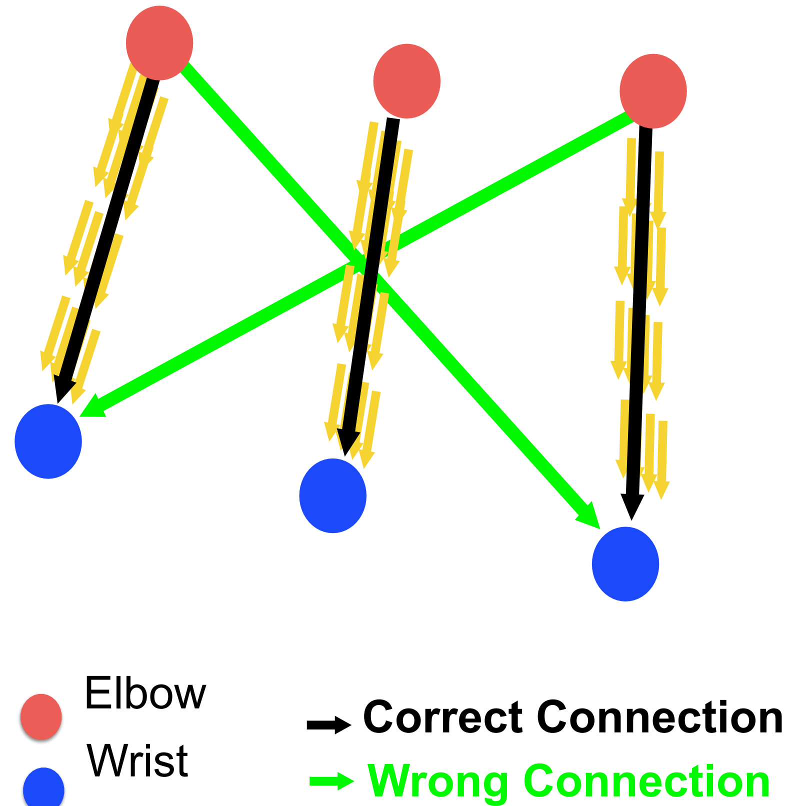


Affinity score between p_1 and p_2
= sum($\vec{v} \cdot \vec{p_1 p_2}$)

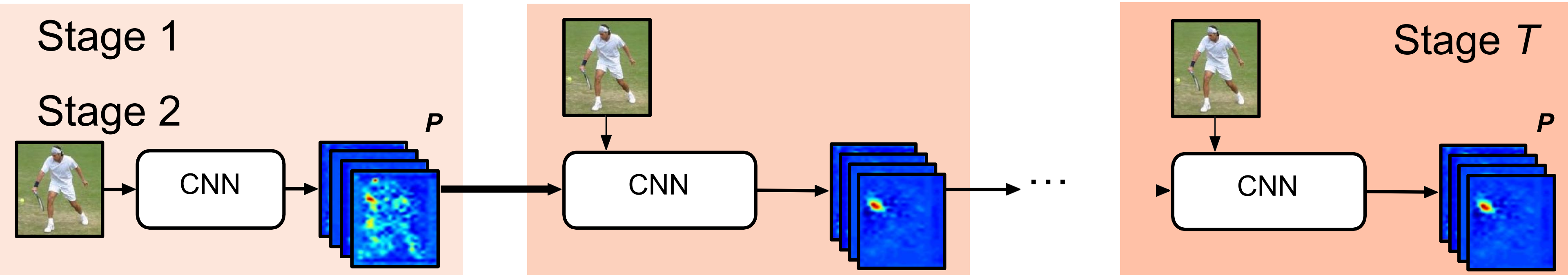
Part Affinity Score Computation As Line Integral

$$\int_{y_1}^{y_2} \int_{x_1}^{x_2} \begin{bmatrix} \mathbf{PAF_x}(x,y) \\ \mathbf{PAF_y}(x,y) \end{bmatrix} \cdot \begin{bmatrix} \mathbf{v_x} \\ \mathbf{v_y} \end{bmatrix} dx\,dy$$

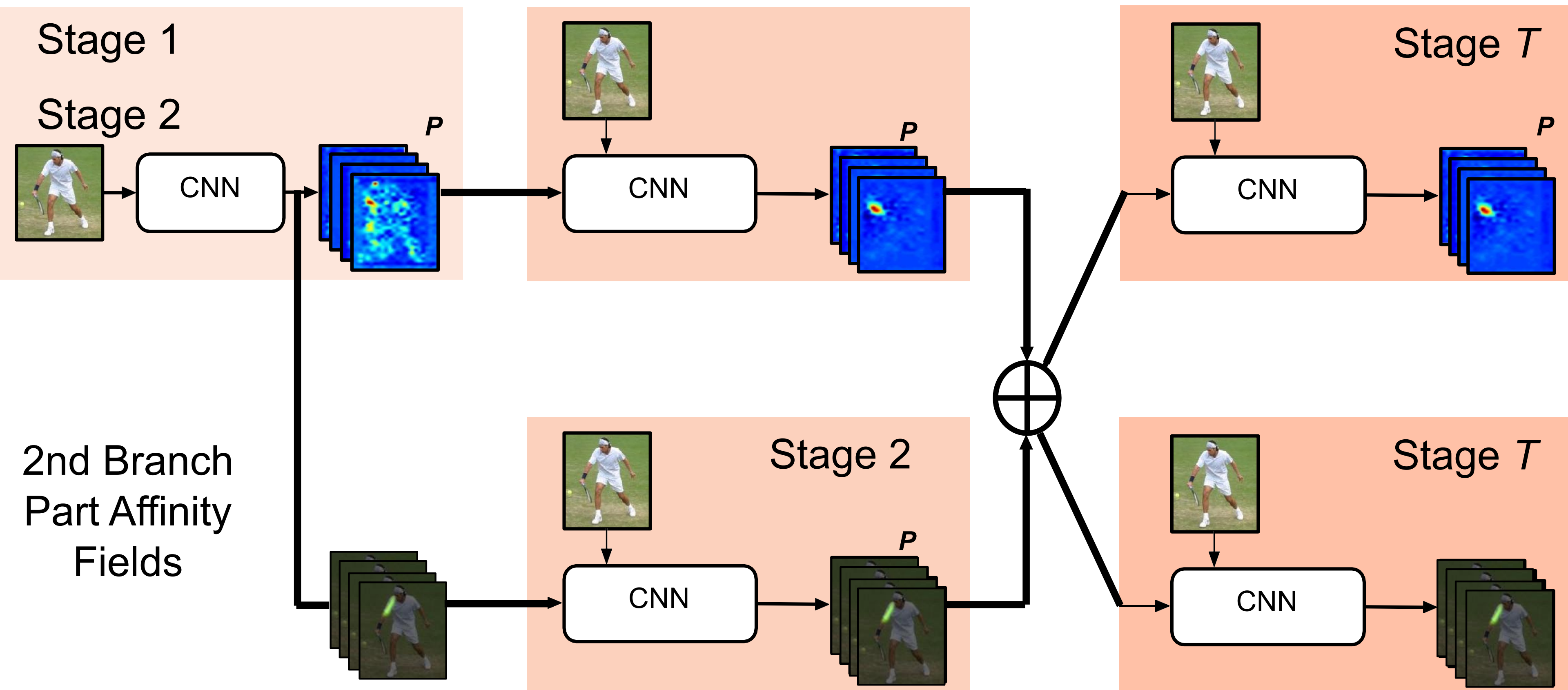
Part Affinity Fields Avoid Spatial Ambiguity



Jointly Learning Parts Detection and Parts Association



Jointly Learning Parts Detection and Parts Association



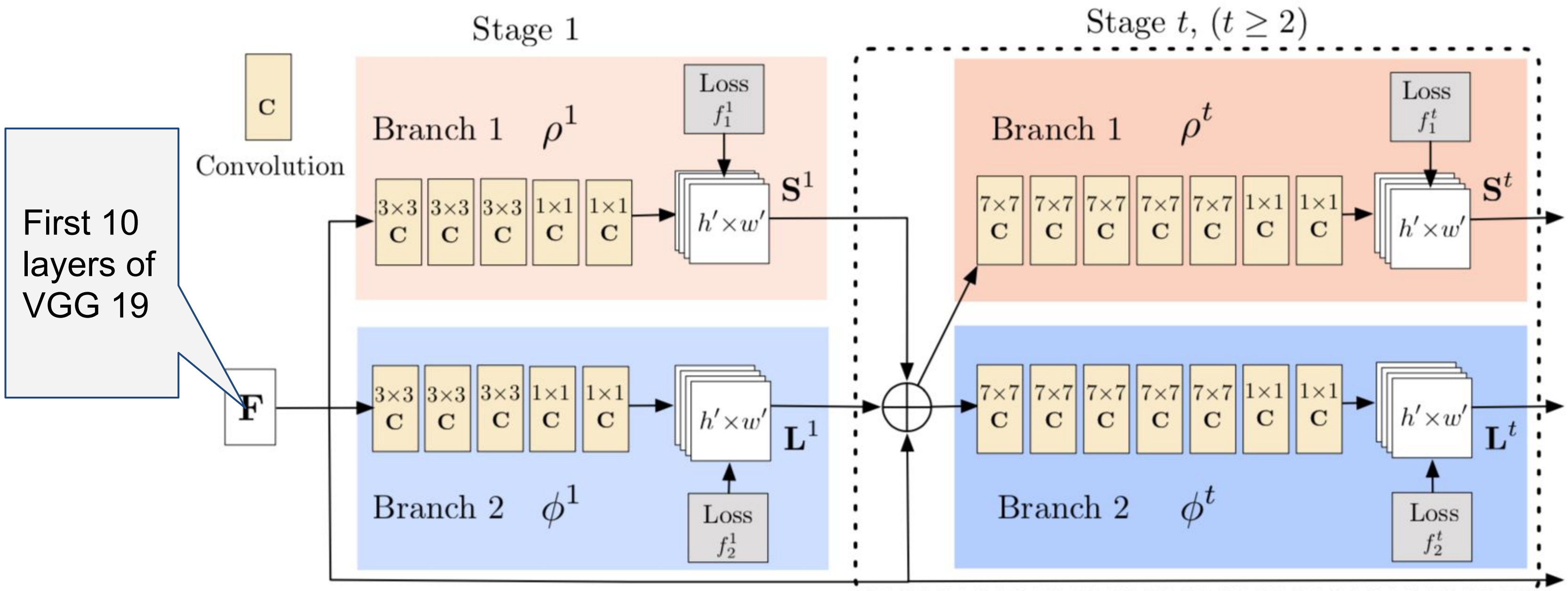
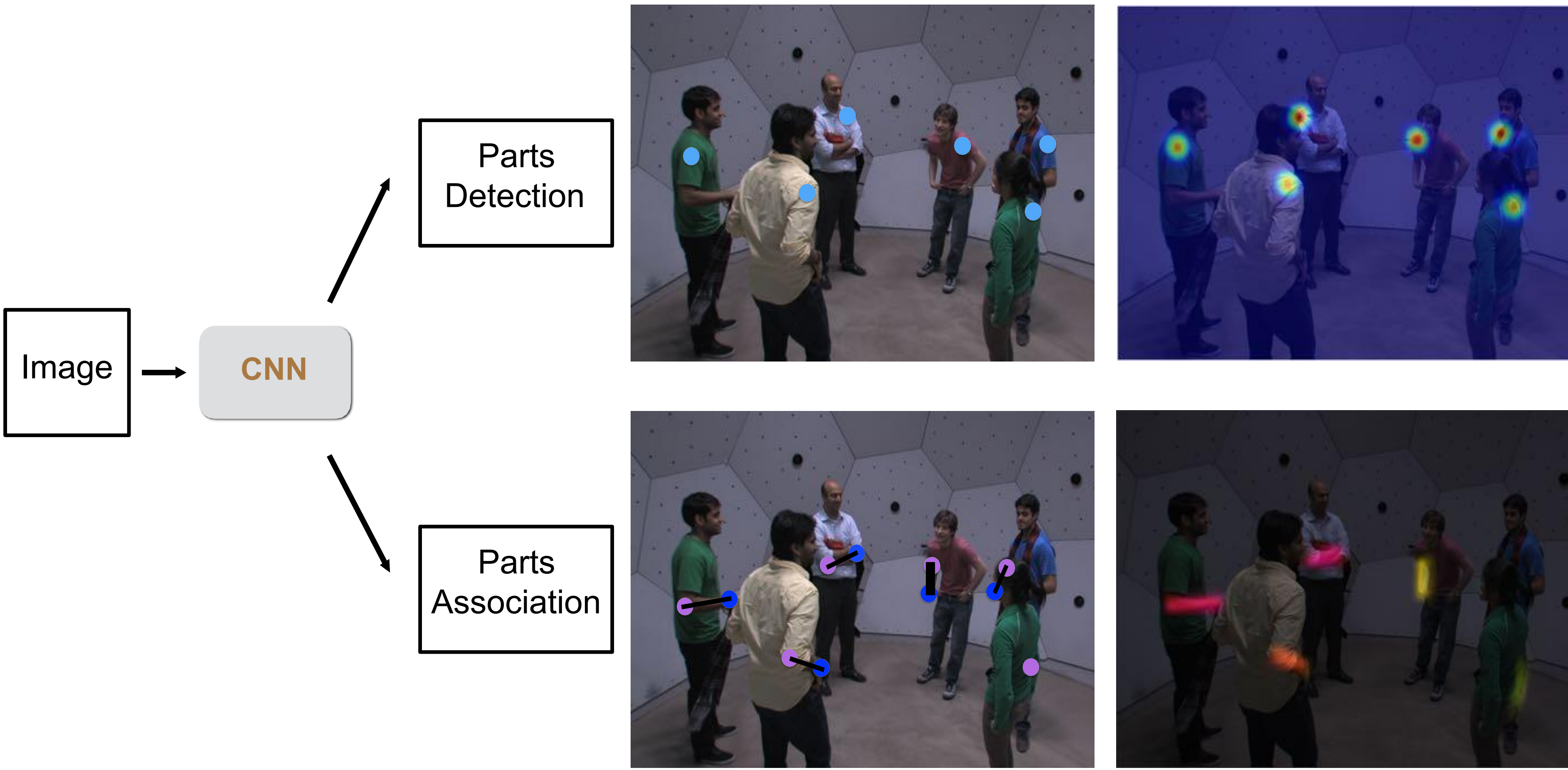


Figure 3. Architecture of the two-branch multi-stage CNN. Each stage in the first branch predicts confidence maps $\mathbf{S}^t$, and each stage in the second branch predicts PAFs $\mathbf{L}^t$. After each stage, the predictions from the two branches, along with the image features, are concatenated for next stage.

Jointly Learning Parts Detection and Parts Association



Datasets

COCO 2016 keypoints challenge dataset (189 Gb)

- 100K person instances labeled with over 1 million total keypoints

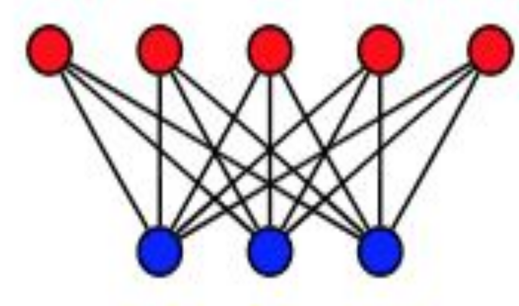
MPII human multi-person dataset

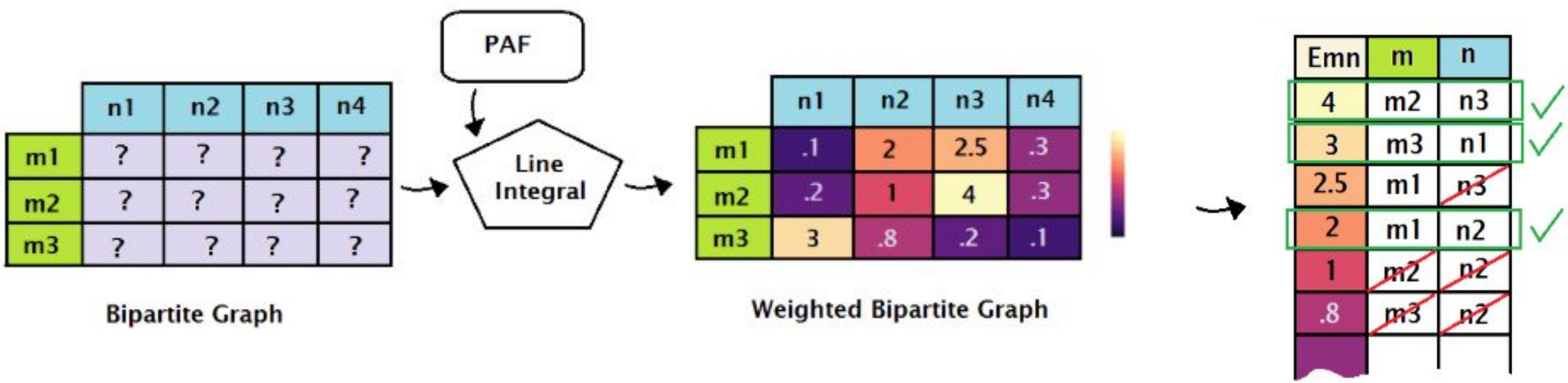

Bkg


PAFs

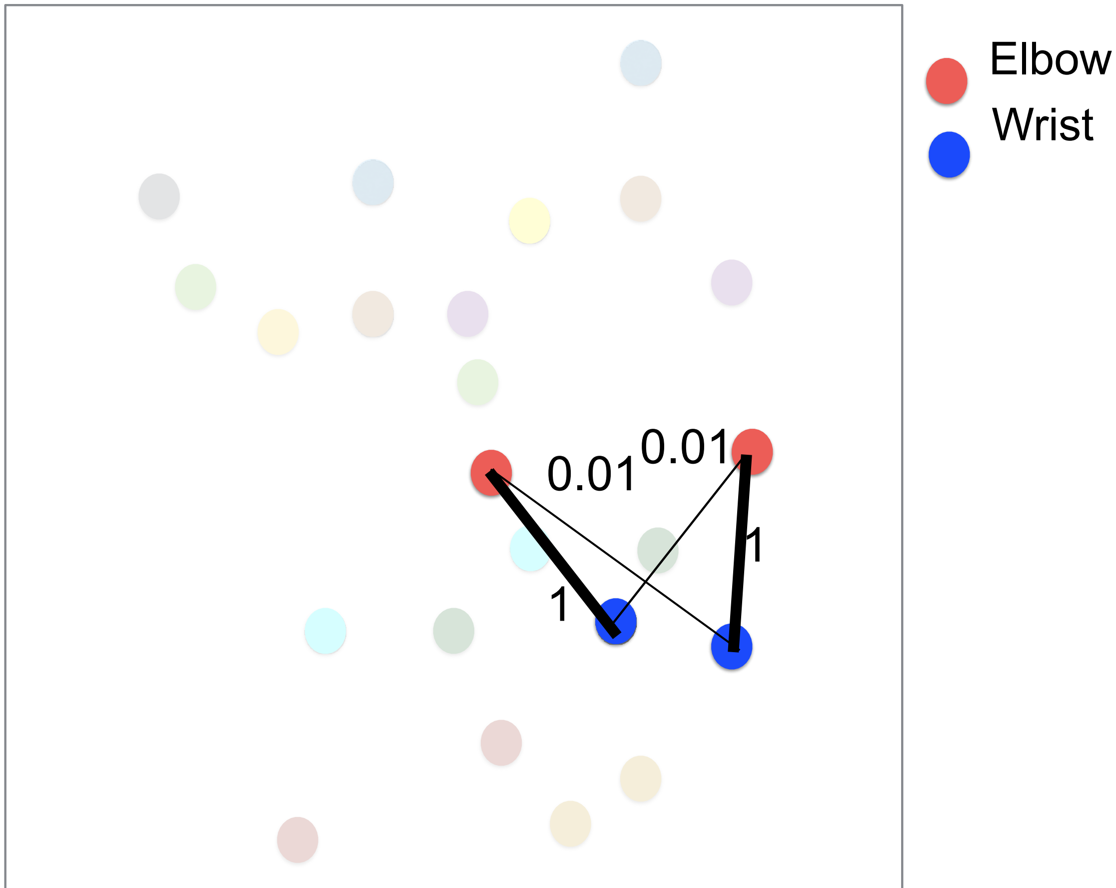
Part matching overview

- For a pair of parts (L.elbow-L.hand)
- Build a weighted bipartite graph
- Solve the assignment problem

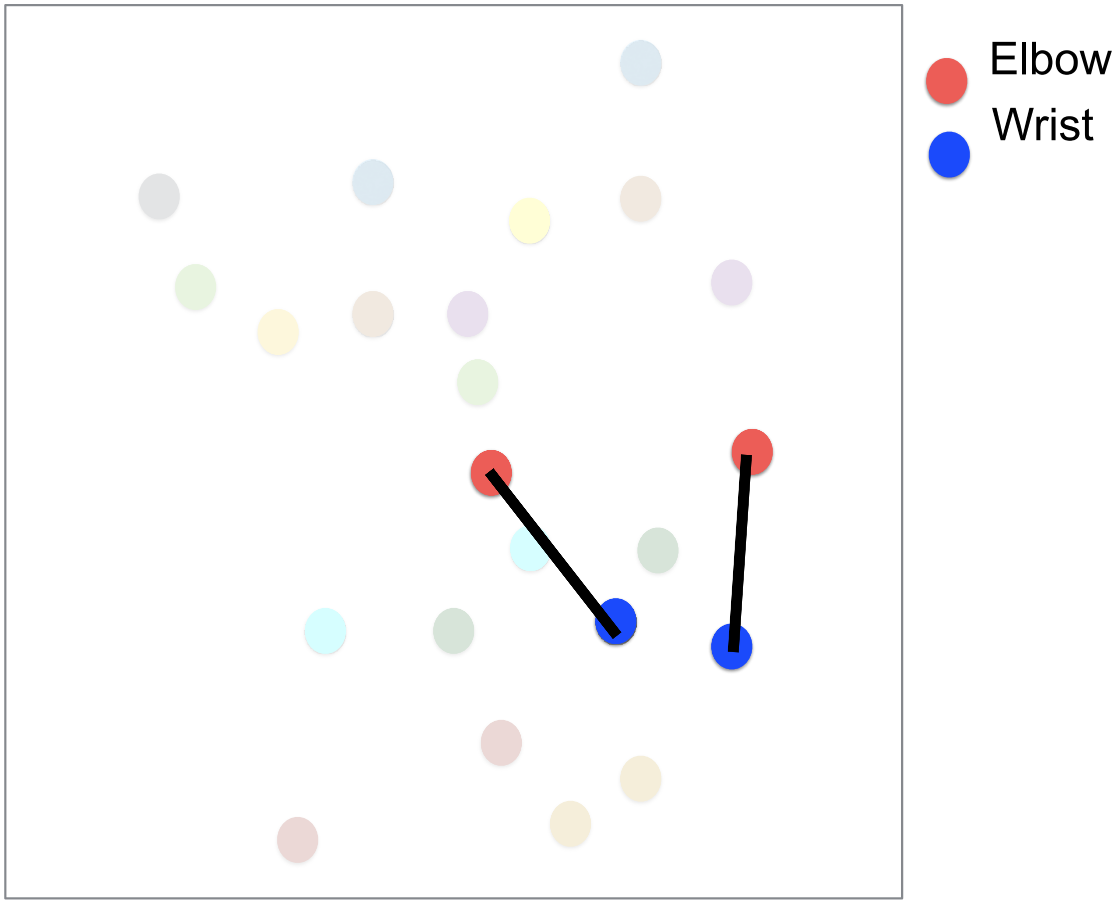




Greedy Algorithm for Body Parts Association

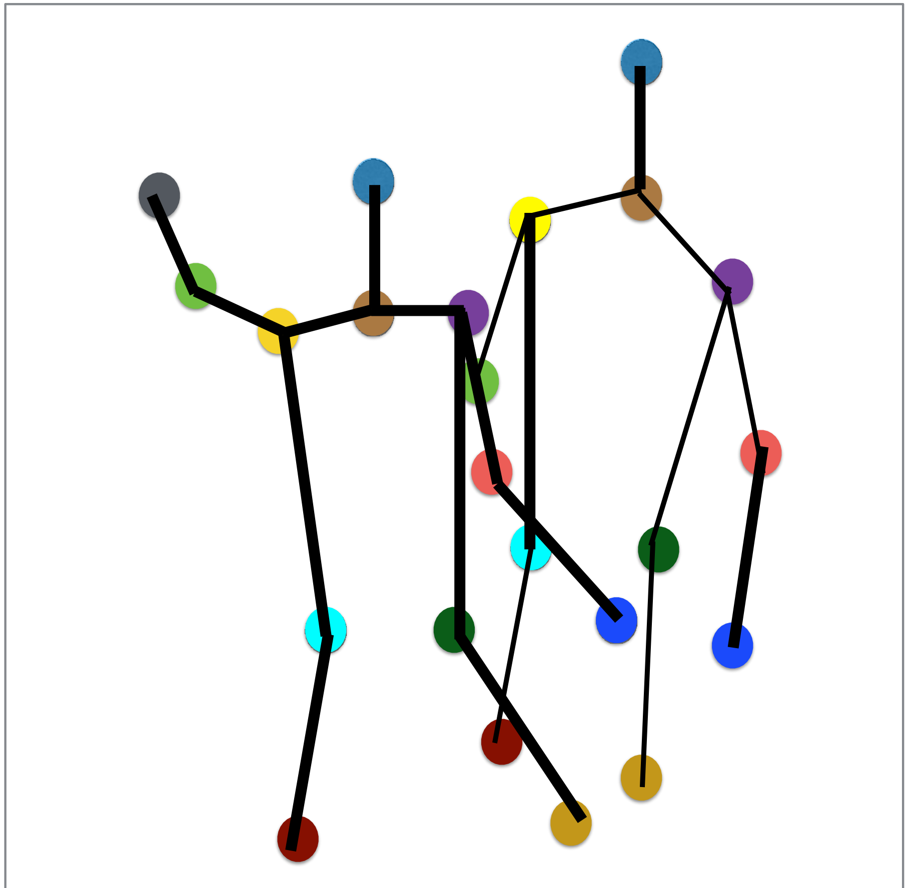


Greedy Algorithm for Body Parts Association



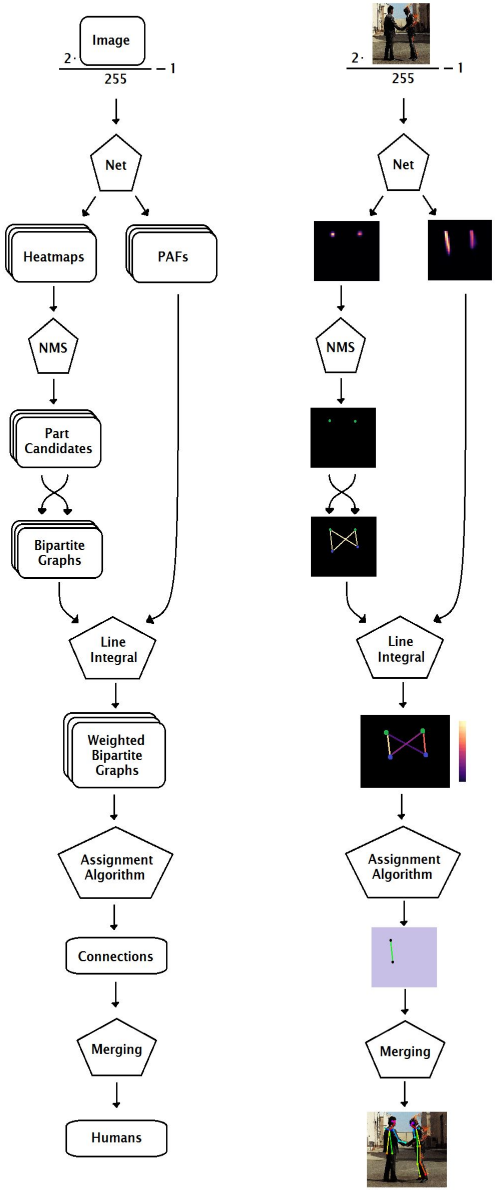
Greedy Algorithm for Body Parts Association

Greedy Algorithm for Body Parts Association



Pipeline Summary

- CNN computes PAFs and heatmaps
- Non Maximum suppression on heatmaps
- Compute assignment confidence between two parts by line integral of the PAF
- Build weighted bipartite graphs
- Solve assignment problem for all pairs of parts
- Merging of connections



Results on COCO Challenge Validation Set

	Method	AP on val
Top-down	**GT bbox + CPM** [1]	63
	SSD [2] **+ CPM** [1]	53
	Our Method	58.5
	Ours + Refinement	61

1 Convolutional Pose Machines [Wei et al. 2016]

2 SSD: Single Shot MultiBox Detector [Liu et al. 2015]

Online demo

Posenet live demo available @ https://github.com/tensorflow/tfjs-models