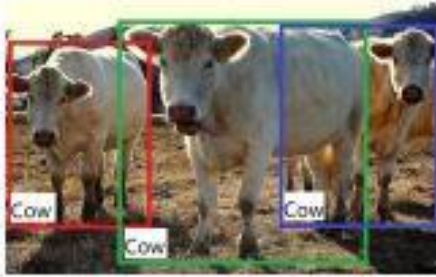


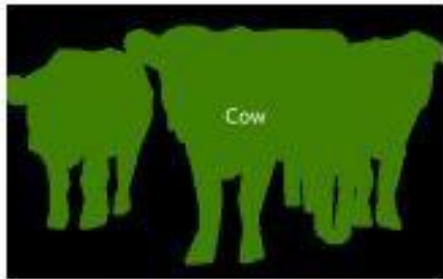
# Neural Networks and object detection



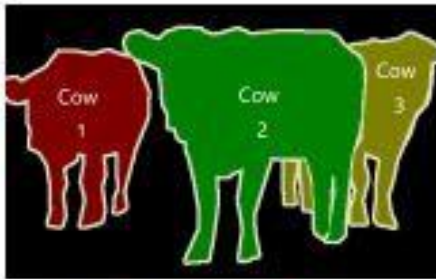
(a) Image Classification



(b) Object Detection



(c) Semantic Segmentation



(d) Instance Segmentation

IA&ML Module 4:  
Image and Video Processing

Sebastien Drouyer &  
Gabriele Facciolo

Lecture 3 - 10-12-2020

# Presentation of the course

**The objective** of this course is to present a panorama of the main modeling aspects and practical insights of neuronal networks (NN) for computer vision applications.

Page: [https://gfacciol.github.io/M1\\_IAML\\_image/](https://gfacciol.github.io/M1_IAML_image/)

## Lessons:

1. – Friday 23/10 : 16h00-18h30 - Intro NN, backprop and CNN for classification
2. – Thursday 12/11 : 13h00-15h30 - Semantic segmentation
3. – Tuesday 10/12 : 13h00-15h30 - Object detection
4. – Thursday 17/12 : 13h00-15h30 - Transfer learning and representation learning

# Plan

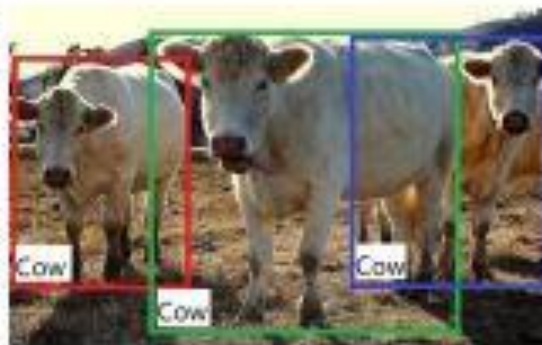
- Last week recap
- Object detection
- Instance detection
- Example of an advanced computer vision task
  - Human pose estimation

Last week recap

# Image classification



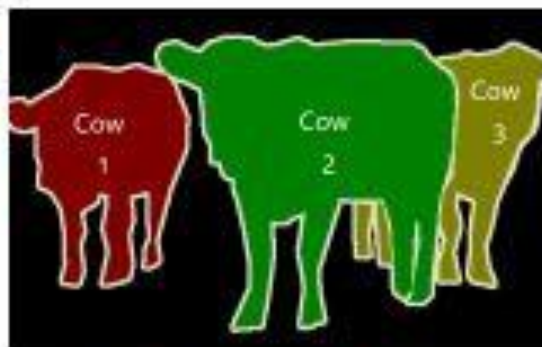
(a) Image Classification



(b) Object Detection

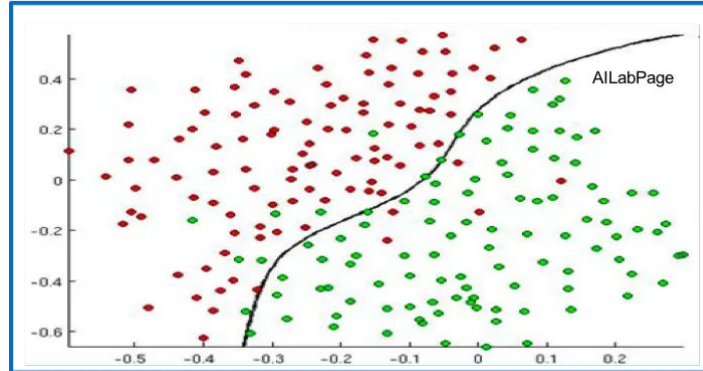
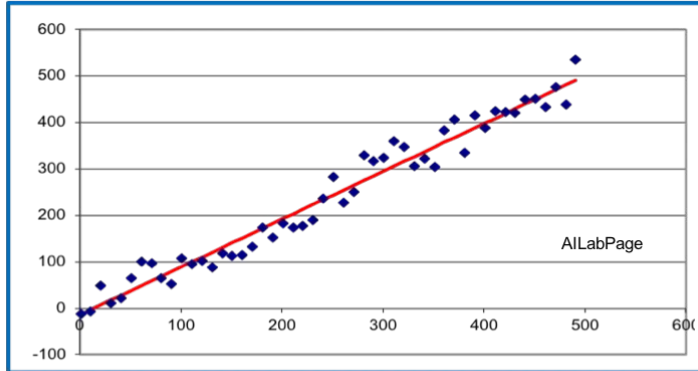


(c) Semantic Segmentation



(d) Instance Segmentation

# 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

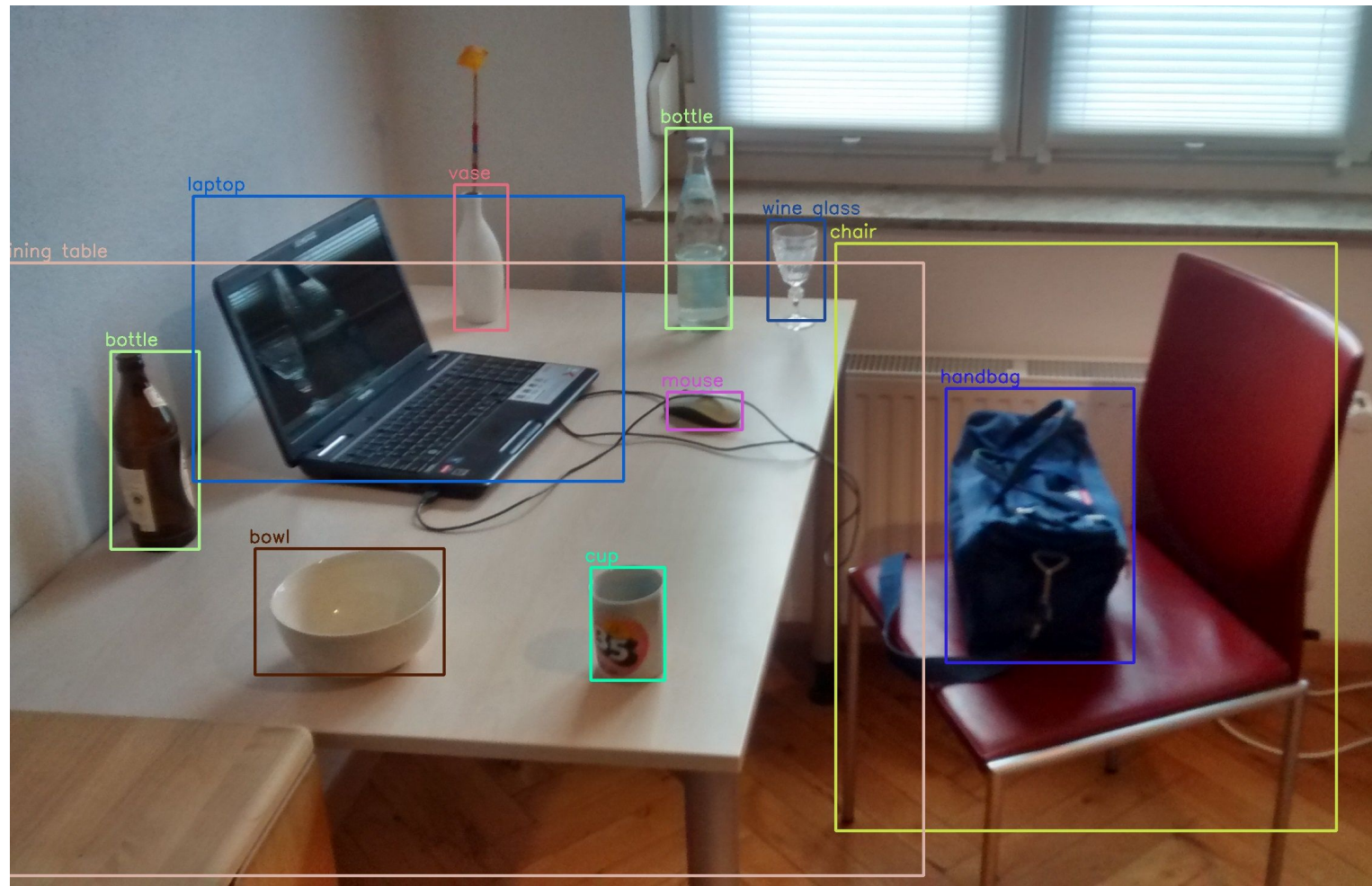


## Classification

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

Object detection

# Object detection

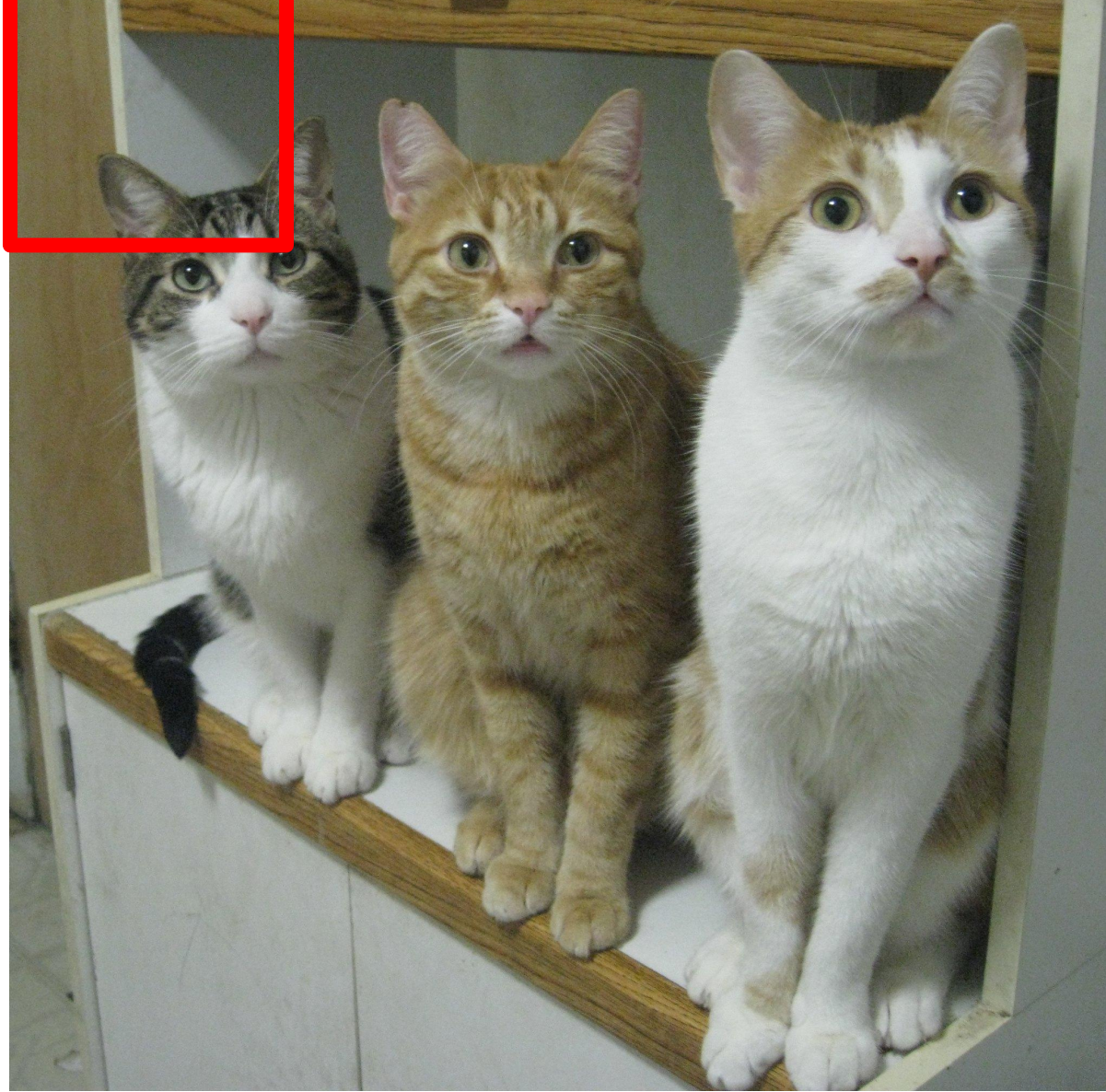




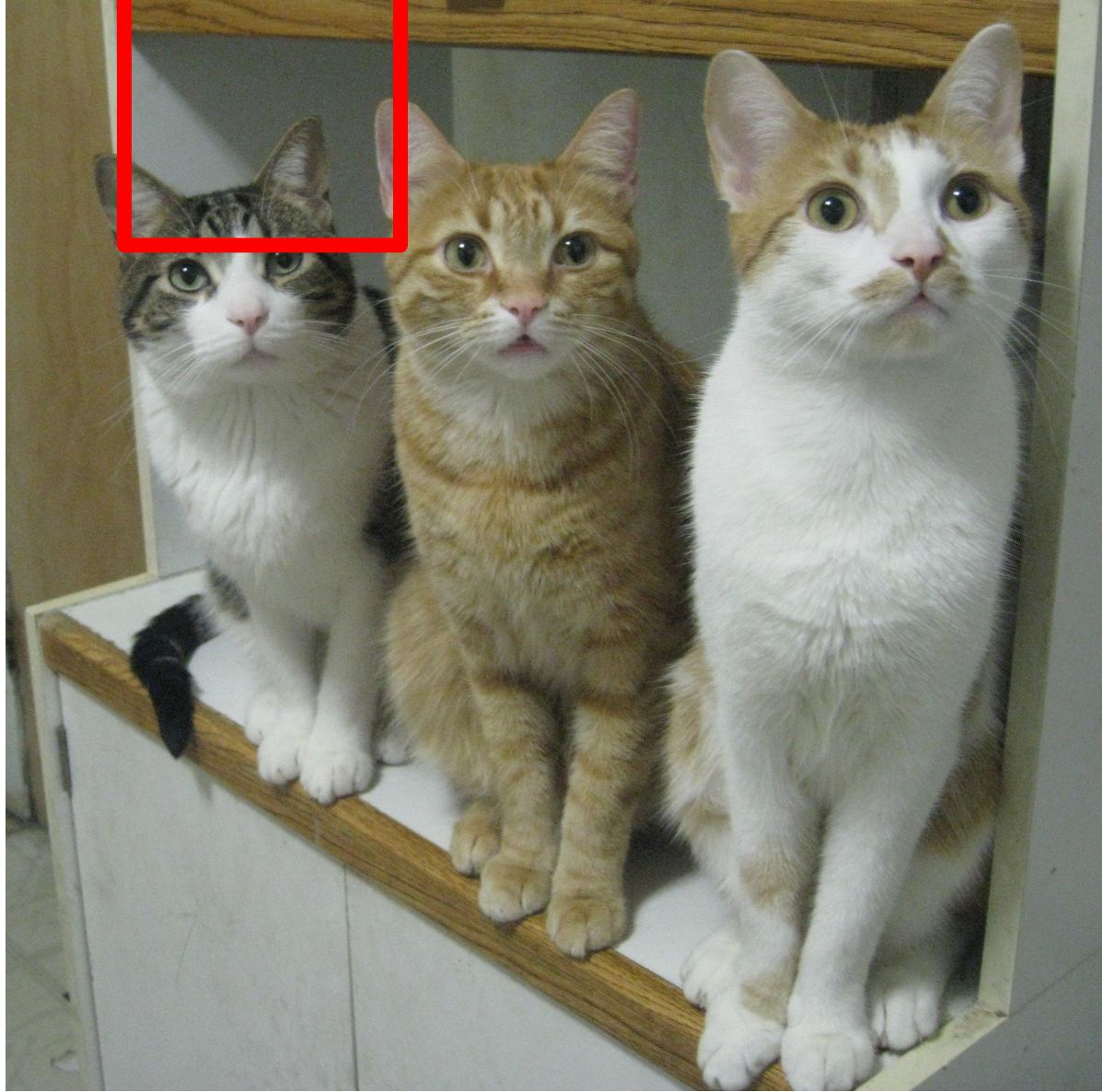
Let's detect cat heads



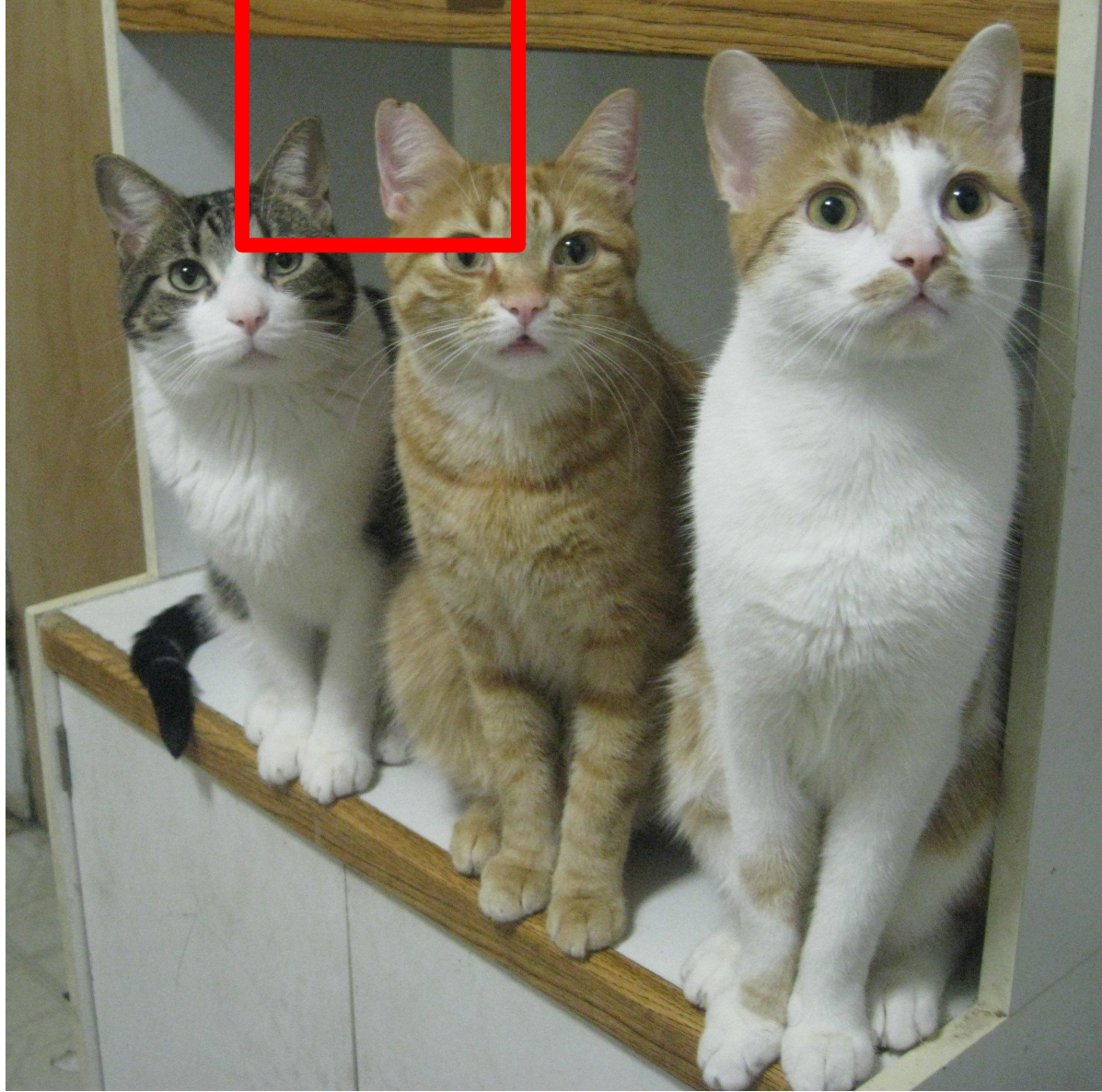
Let's detect cat heads



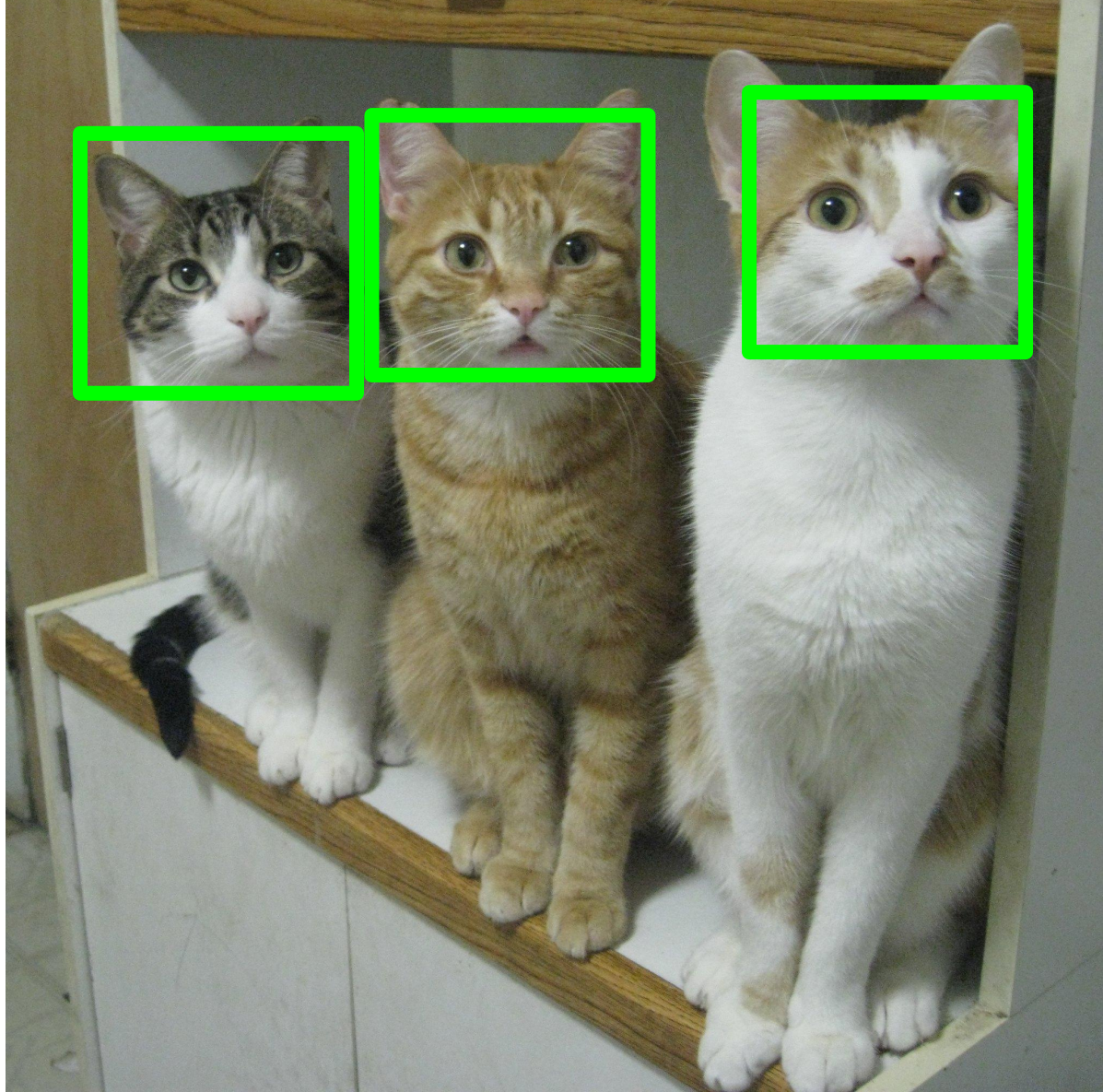
Let's detect cat heads



Let's detect cat heads



Let's detect cat heads



# Object detection

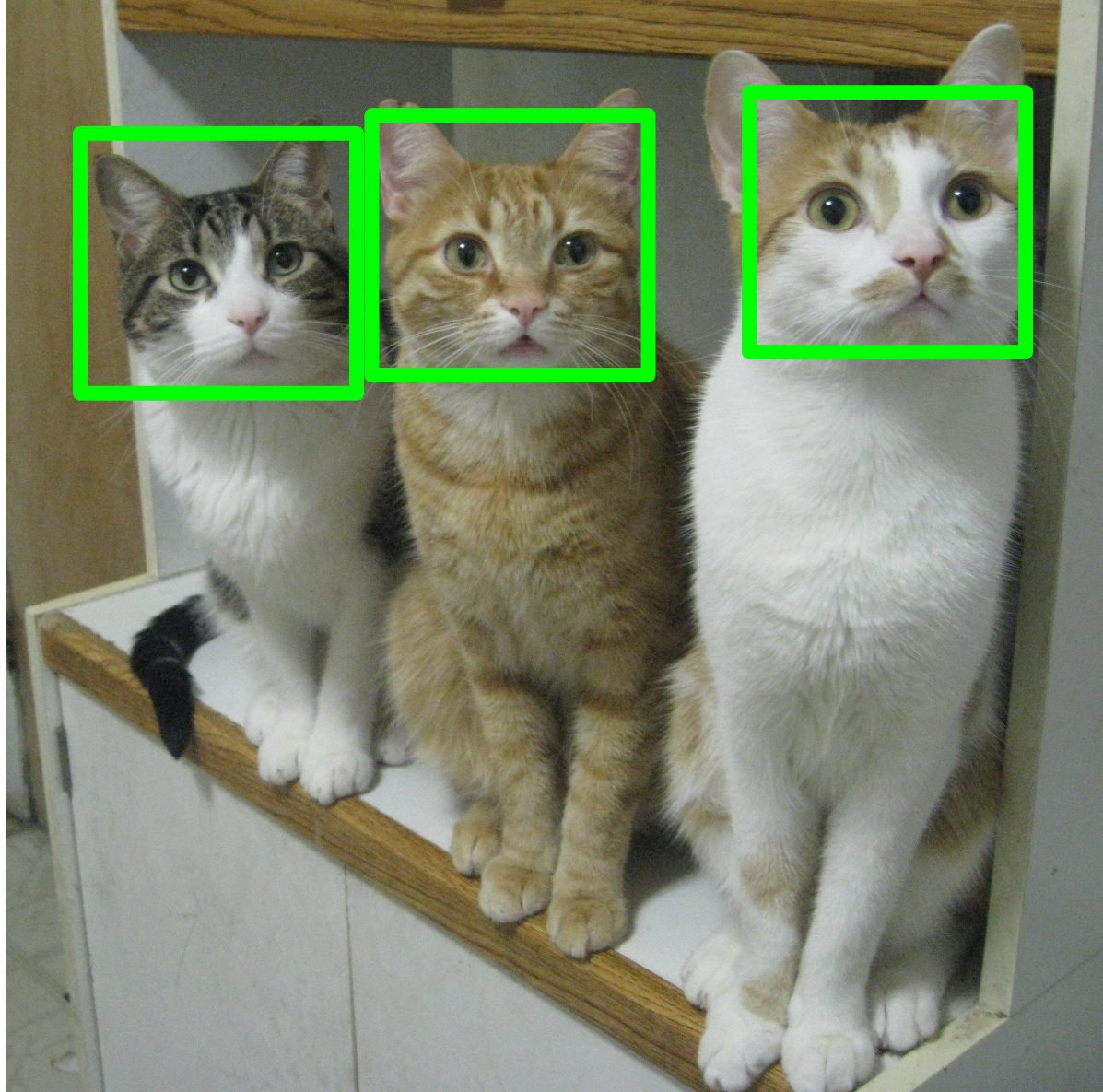
→ Object detection can again be seen as a classification problem.

**Several issues will appear.**

Let's detect cat heads.

First, more than one  
detection per cat is likely.

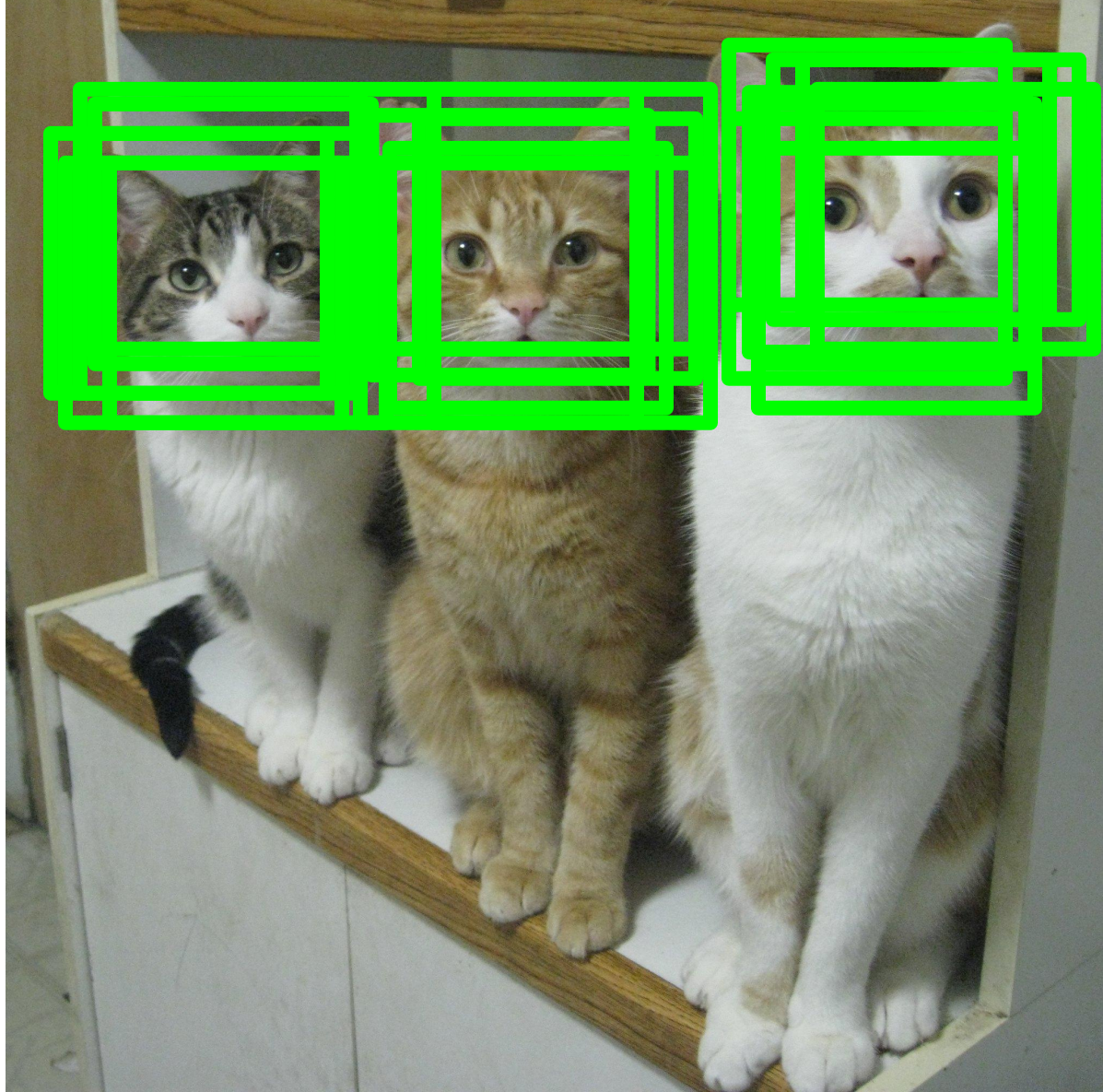
Dream →



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.

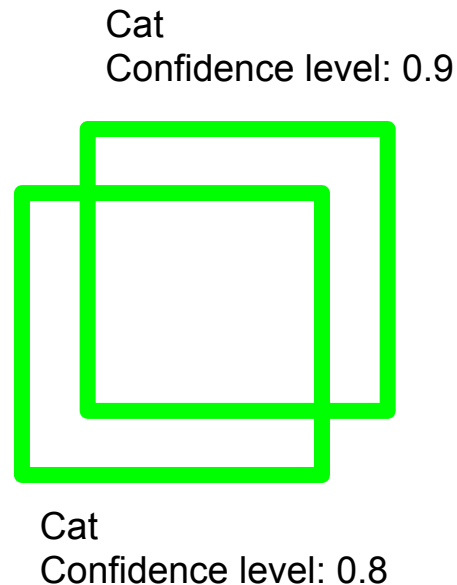
Cat head	Not cat head
0.8	0.2

# 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  $\text{IOU} > 0.5$  → keep BB with highest confidence level



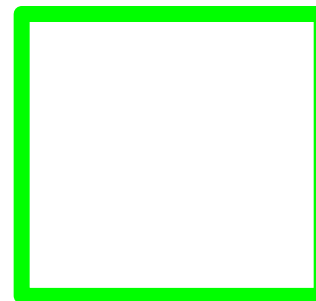
# Object detection

Solution: Non-Maximum Suppression (NMS)

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

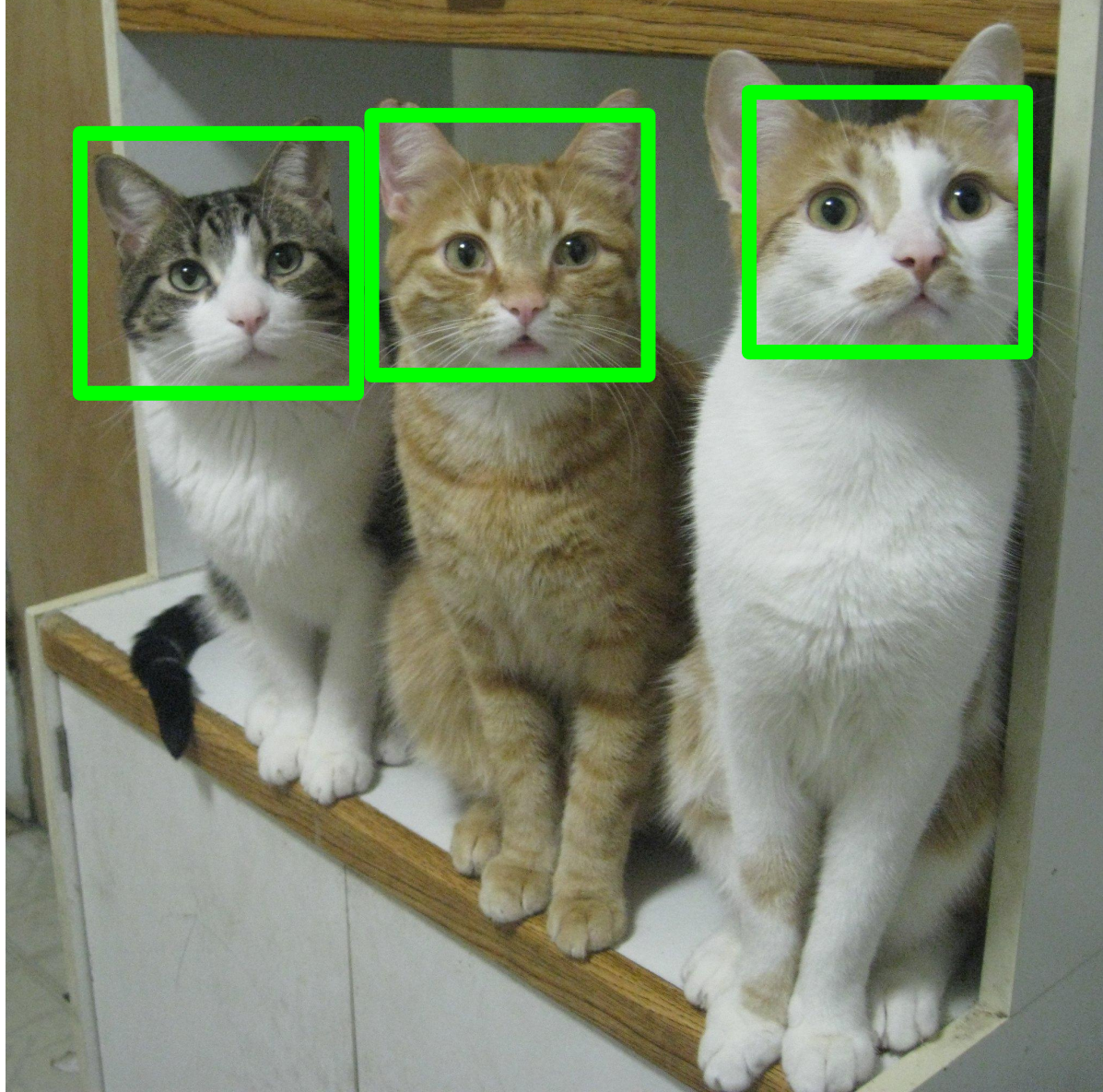
1. Compute Intersection Over Union (IOU)  
Measures how much the two BBs intersect
2. If  $\text{IOU} > 0.5$  → keep BB with highest confidence level

Cat  
Confidence level: 0.9



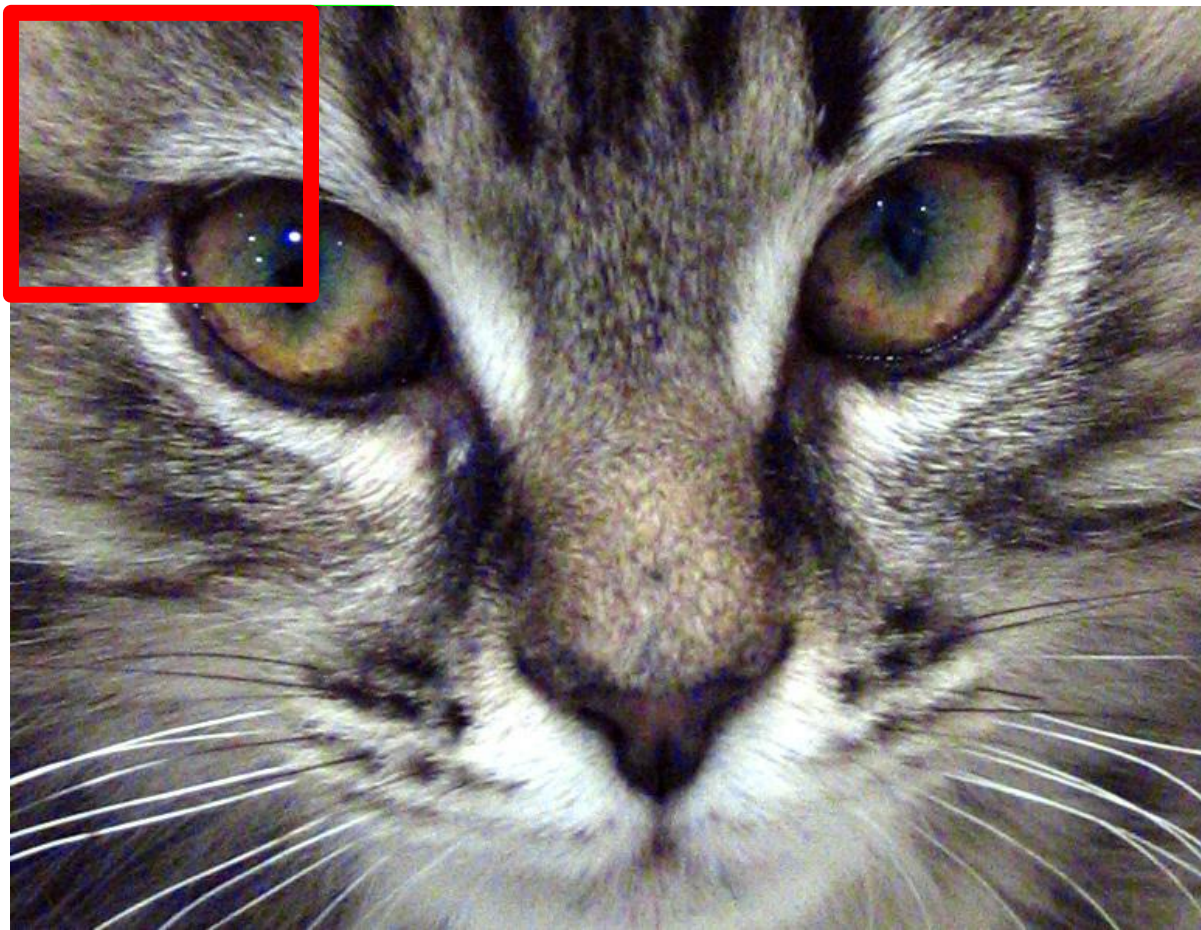
Let's detect cat heads.

Window size here is well adapted.



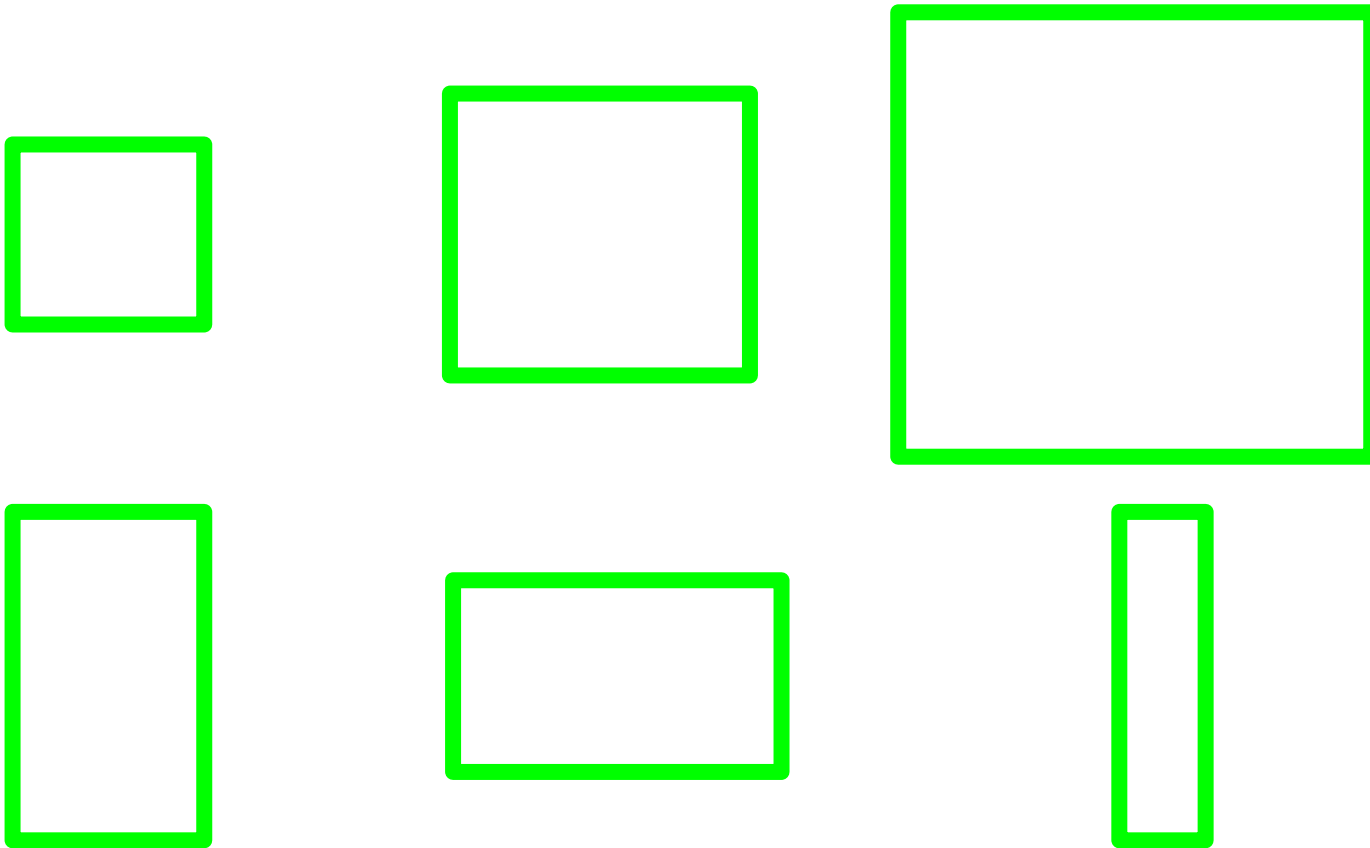
Let's detect cat heads.

Window size here is not well adapted.

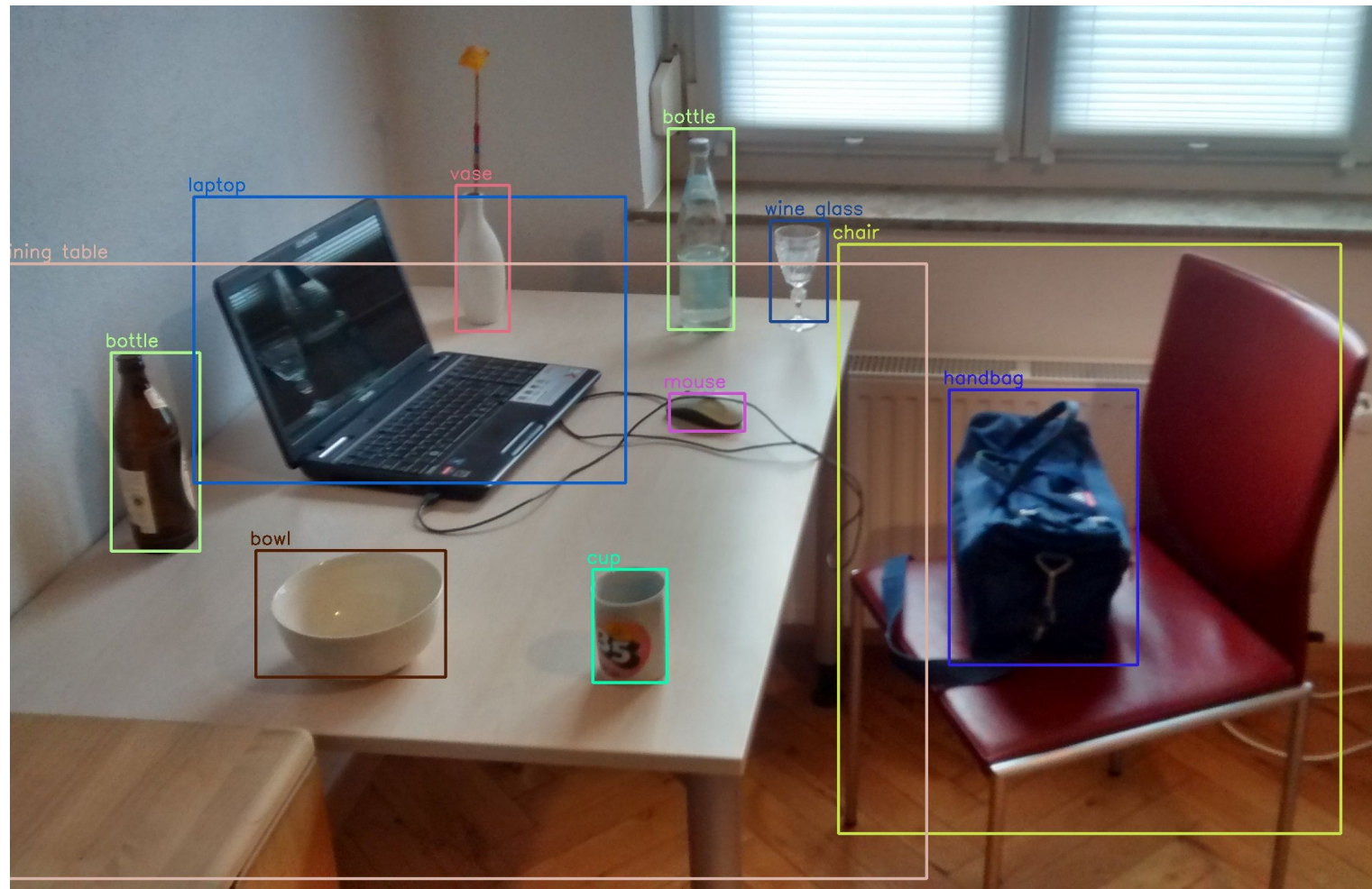


# Object detection

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:  $12288 \times 15 = 184320$  classifications for a single image!



# Object detection

Recap solution 1:

1. Classify sliding windows of different 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).

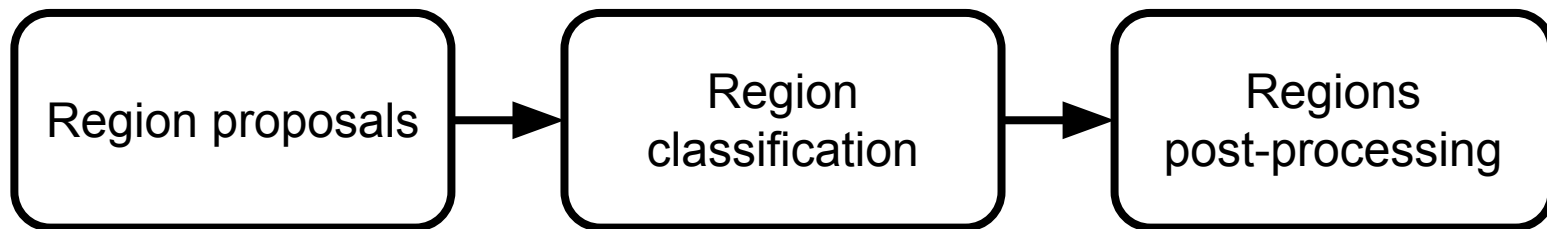
# Object detection

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



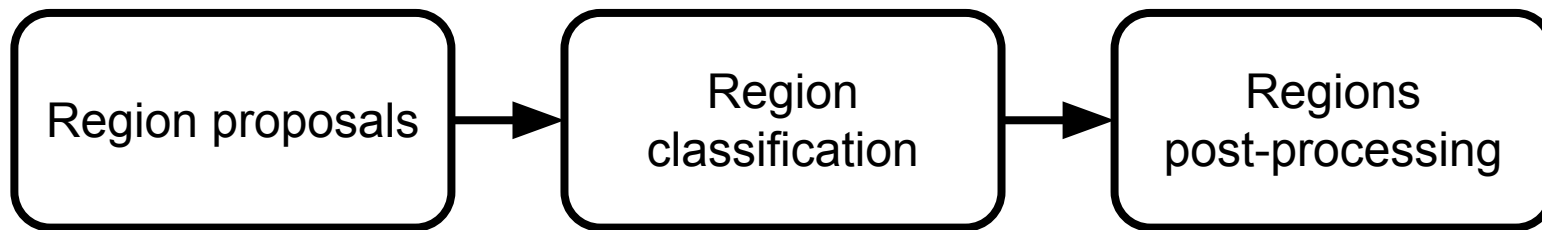
# Object detection

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



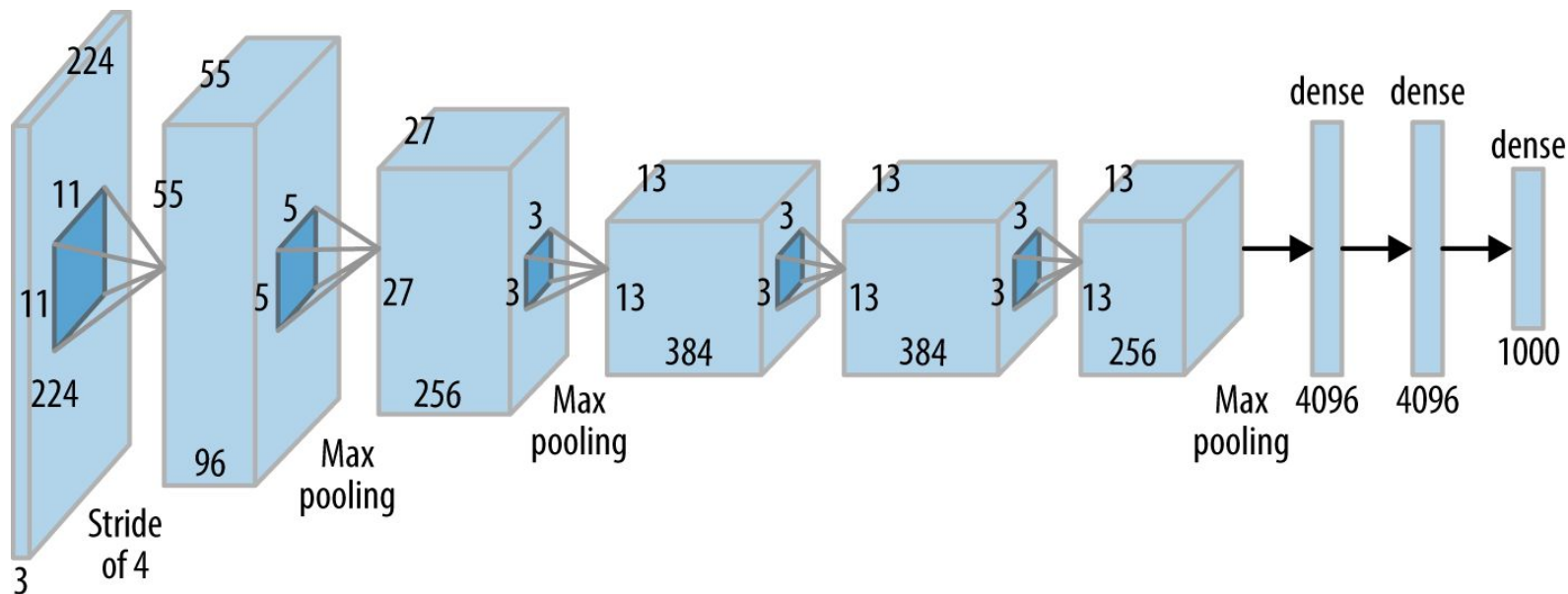
Solution 2: Use Selective Search for region proposals.

→ Main idea behind **R-CNN**

R-CNN has two additional “hacks”.

# Object detection

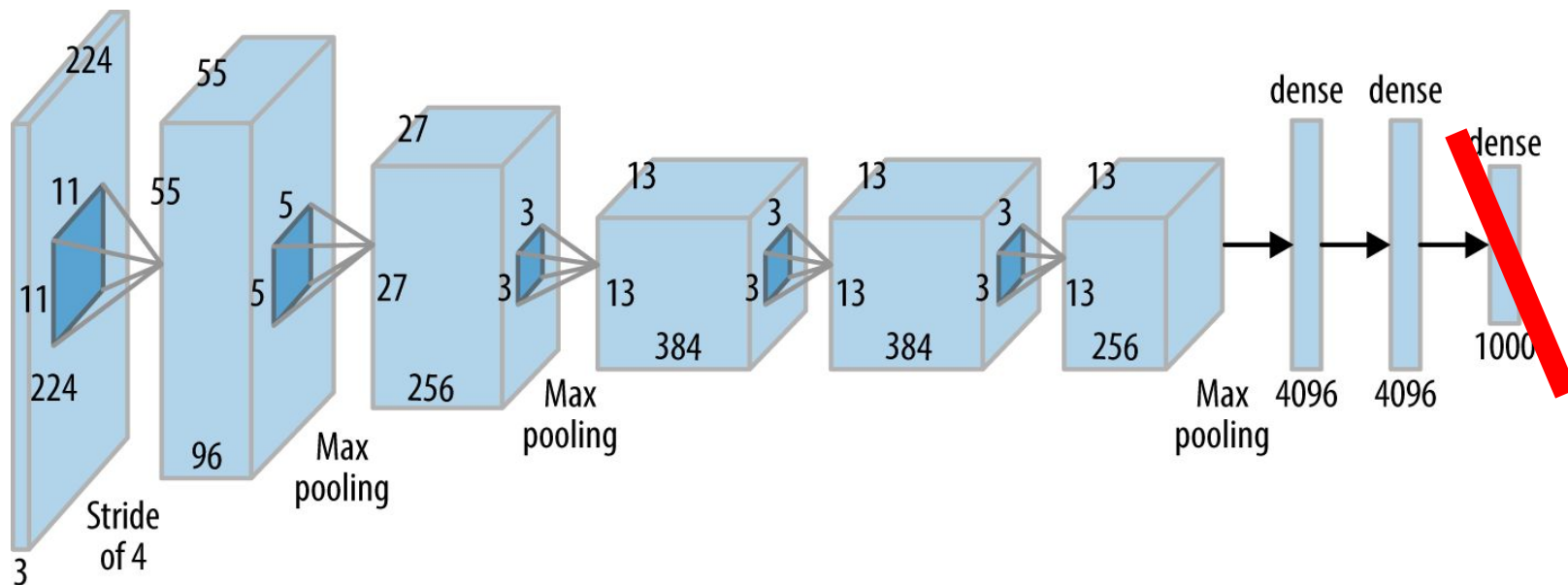
1st hack:



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

# Object detection

1st hack:



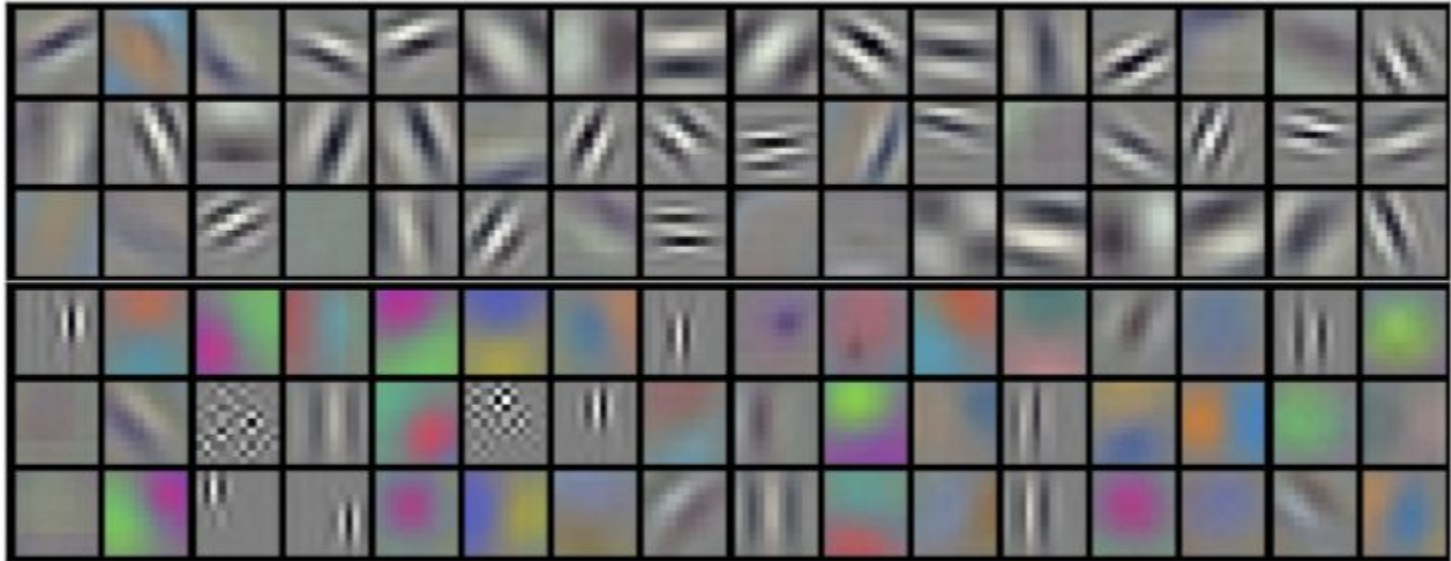
... where the final layer is removed and connected to a SVM.

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*

# Object detection

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

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

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

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

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

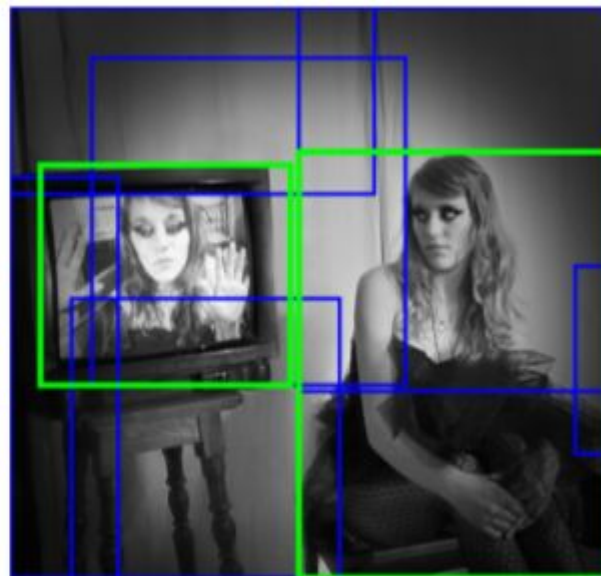
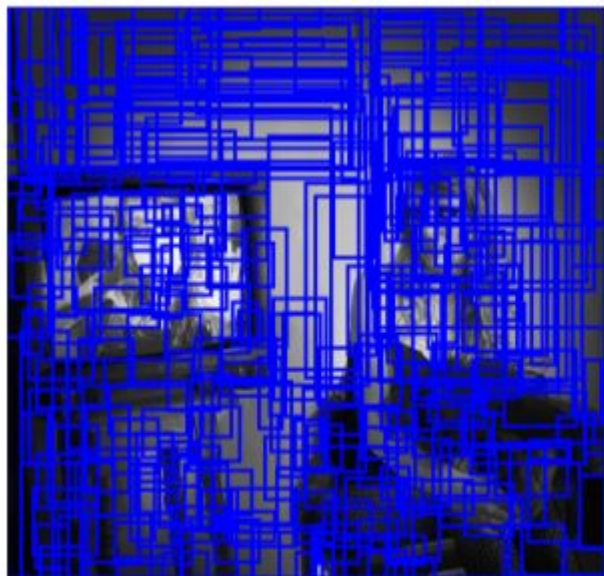
$P_x, P_y, P_w, P_h$ : x, y, width, height of detected window.

$G_x, G_y, G_w, G_h$ : x, y, width, height of target window.

# Object detection

Issues:

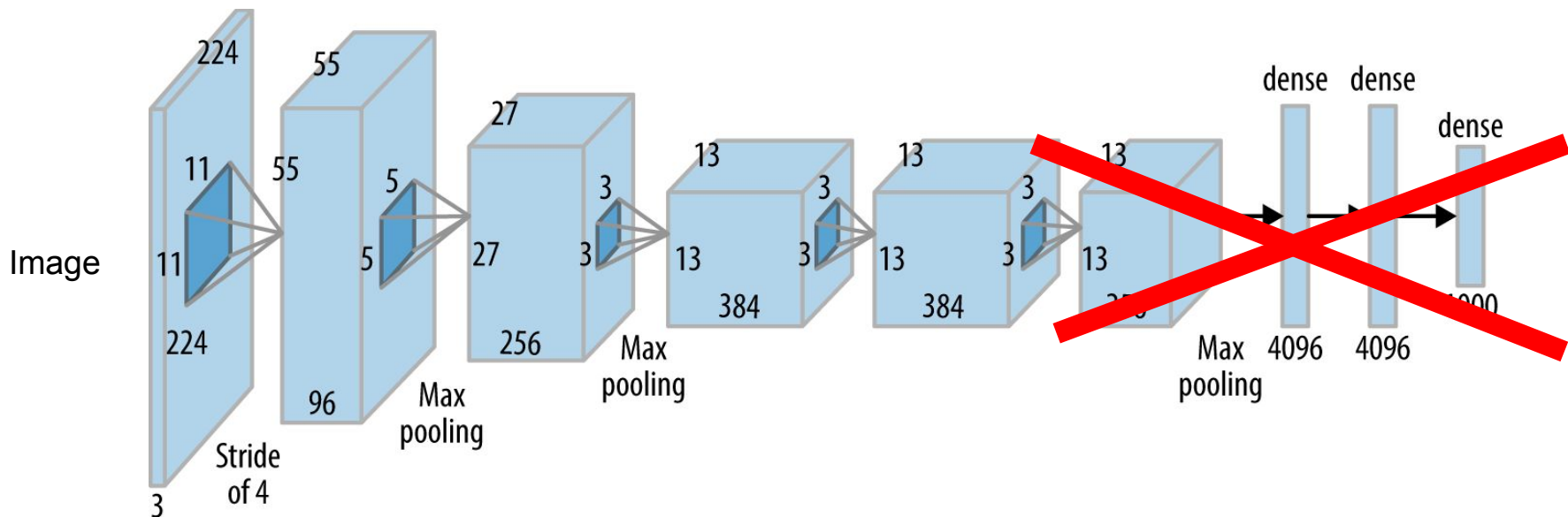
- Classifier in two parts, trained separately.
- Regressor trained separately.
- Still a lot of computation redundancies → still slow: 40-60s per image.



# Object detection

How can we reduce the computational redundancies?

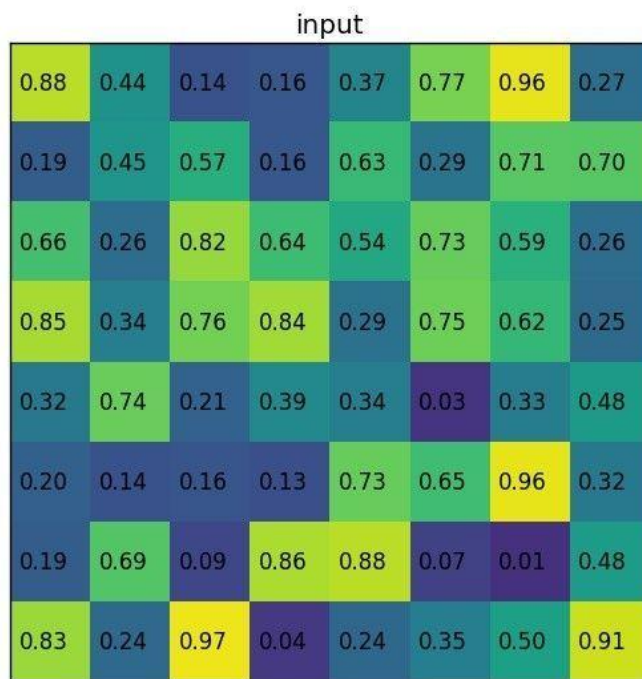
By applying the neural network on the whole image, and then extracting relevant information when testing a window. → Use Region of Interest pooling layer.



# Object detection

2x2 Region of Interest pooling layer

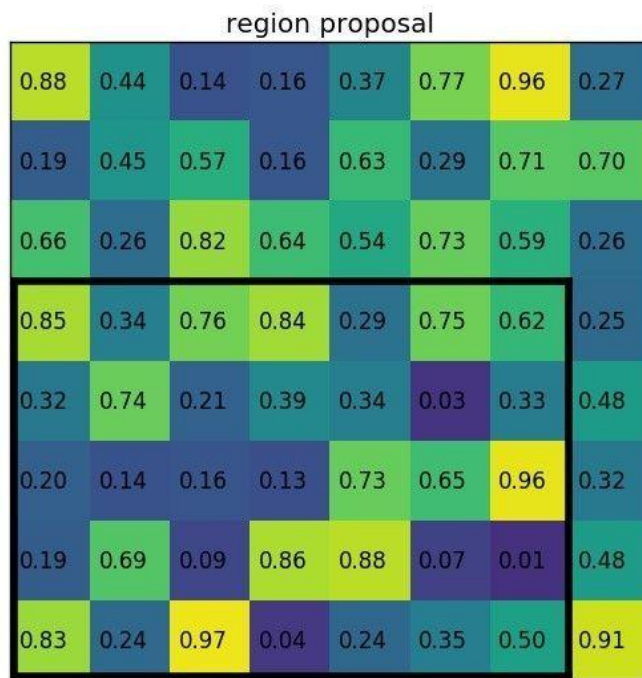
It allows us to reuse the feature map from the convolutional network





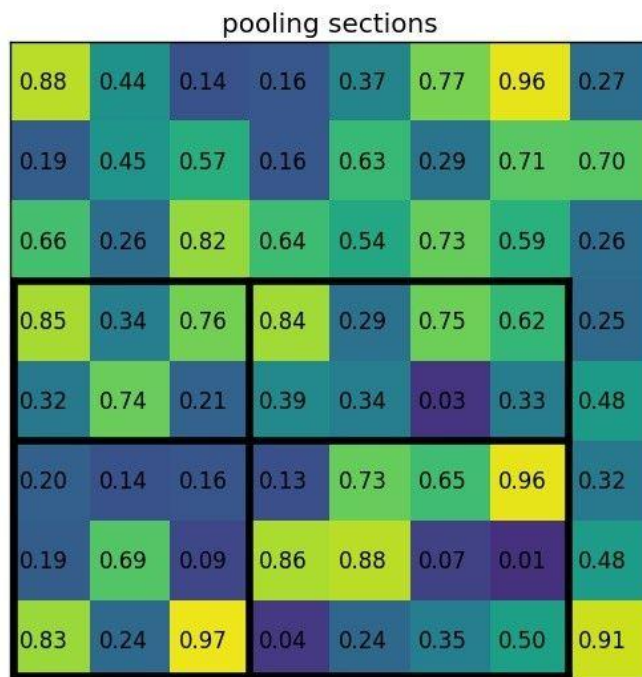
# Object detection

2x2 Region of Interest pooling layer



# Object detection

2x2 Region of Interest pooling layer



# Object detection

0.85	0.84
0.97	0.96

# Object detection

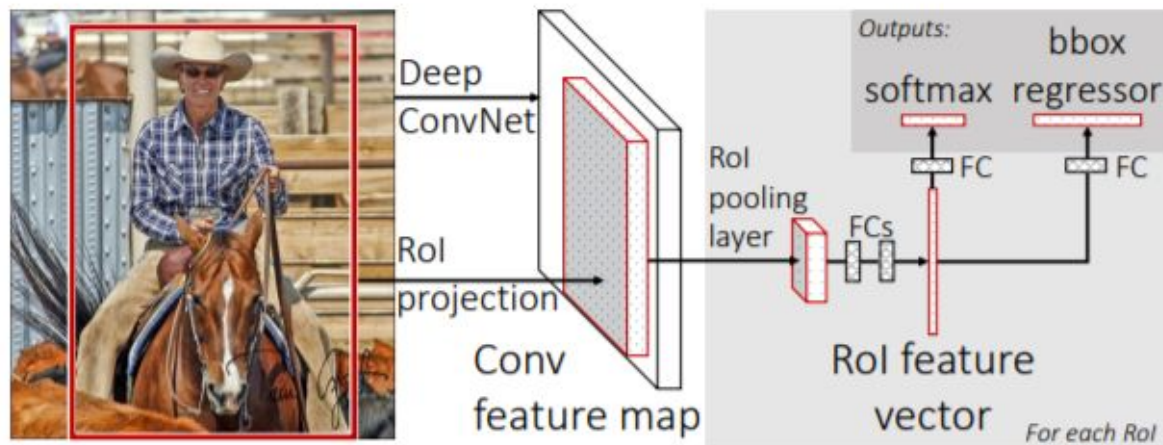


Figure 1. Fast R-CNN architecture. An input image and multiple regions of interest (RoIs) are input into a fully convolutional network. Each RoI is pooled into a fixed-size feature map and then mapped to a feature vector by fully connected layers (FCs). The network has two output vectors per RoI: softmax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



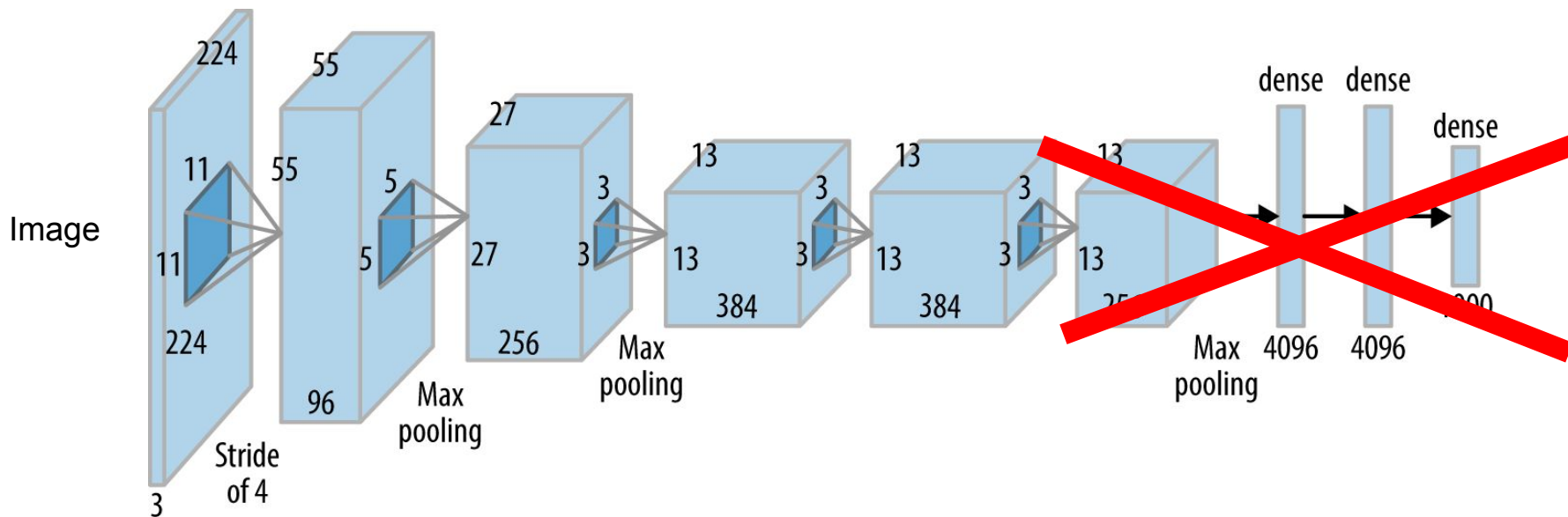
# Object detection

New bottleneck: region proposals (Selective Search)

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

# Object detection

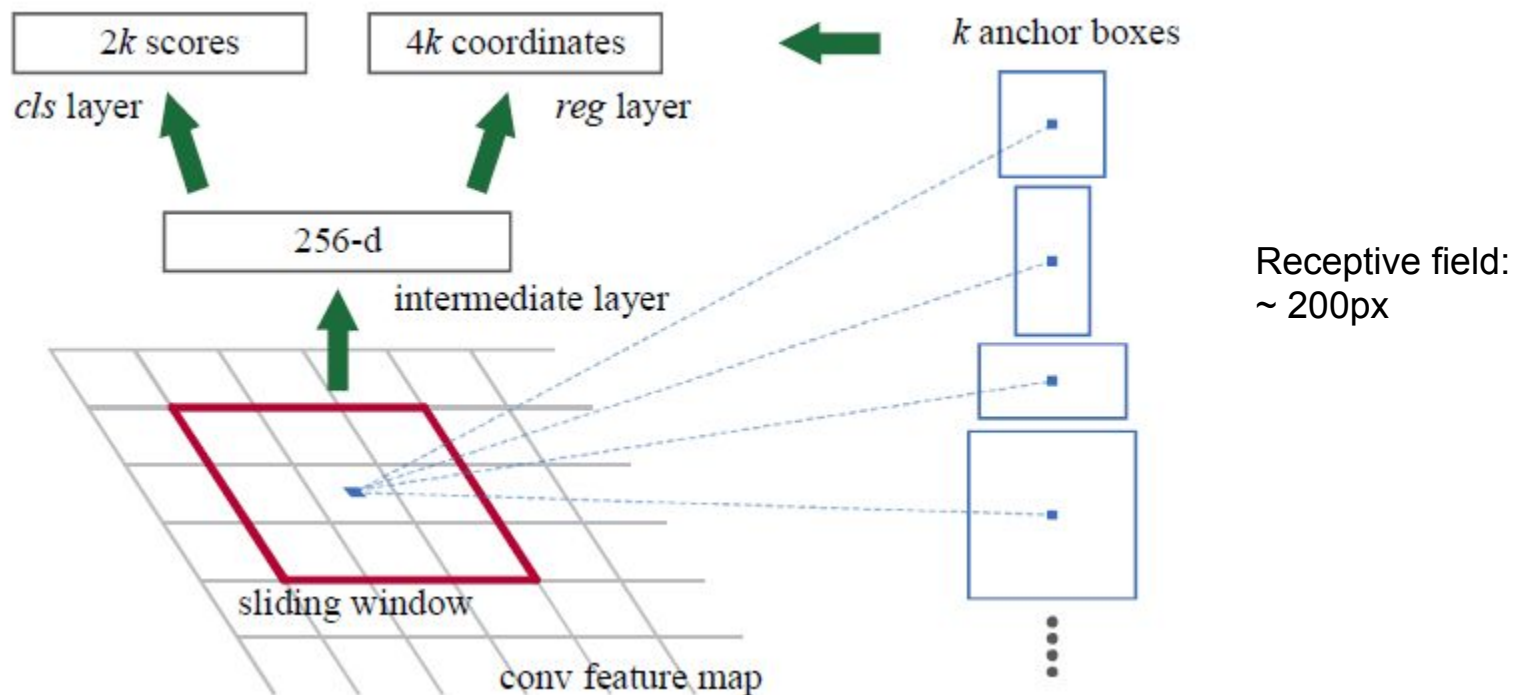
New concept: Region Proposal Network (RPN)





# Object detection

New concept: Region Proposal Network (RPN)



# Object detection

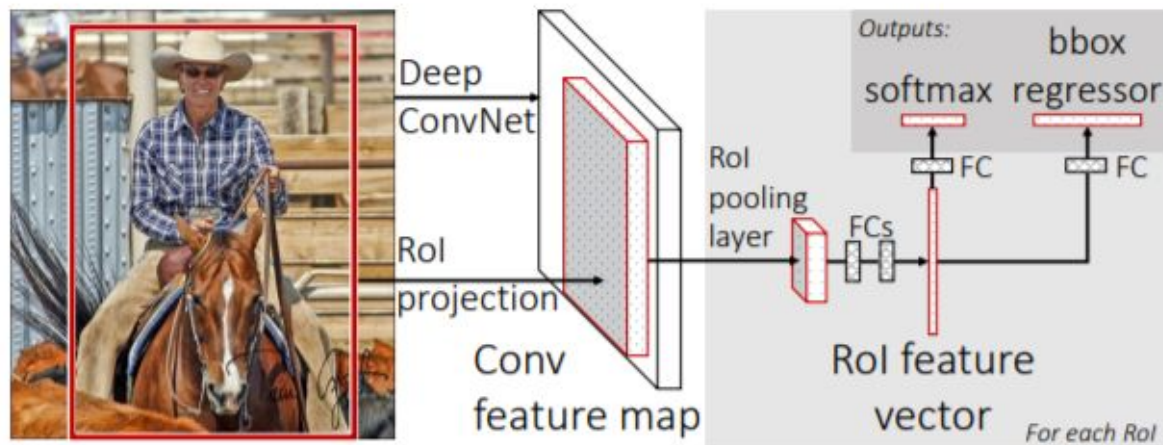


Figure 1. Fast R-CNN architecture. An input image and multiple regions of interest (RoIs) are input into a fully convolutional network. Each RoI is pooled into a fixed-size feature map and then mapped to a feature vector by fully connected layers (FCs). The network has two output vectors per RoI: softmax 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**

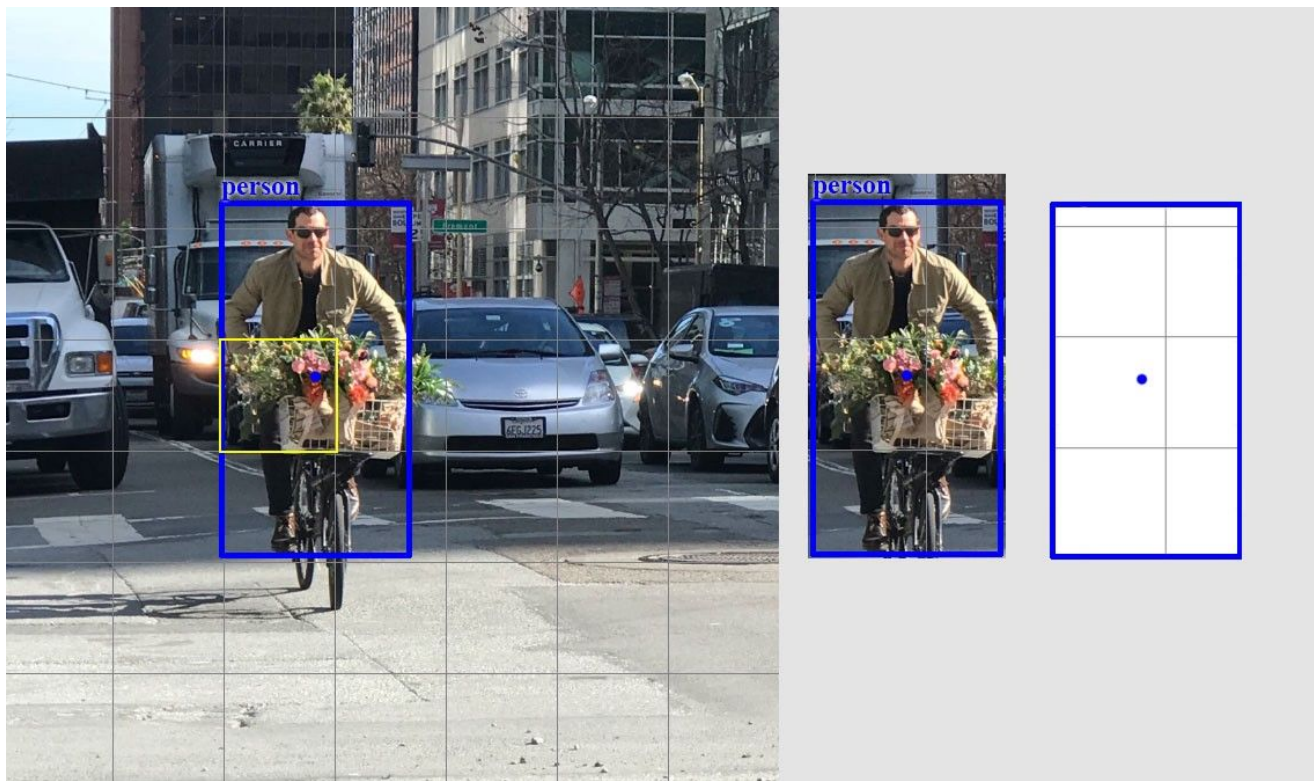
Results:

- Even better performance and speed-up

YOLO

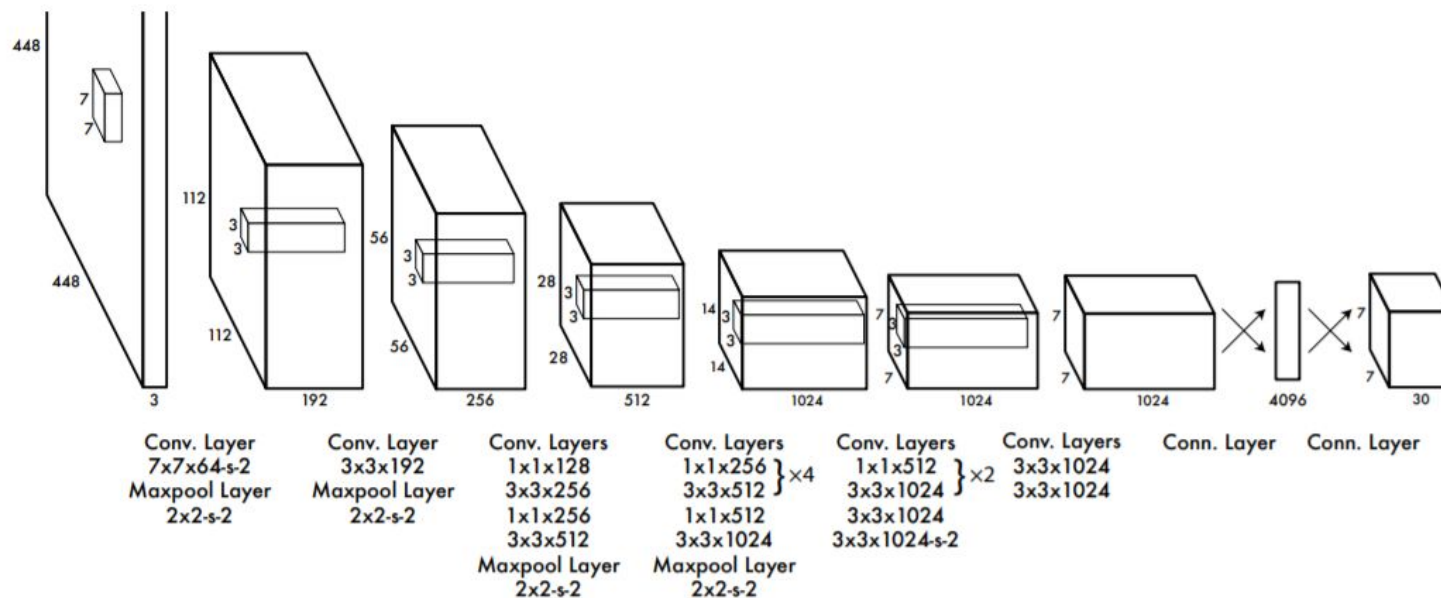
# Object detection

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  $1 \times 1$  convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution ( $224 \times 224$  input image) and then double the resolution for detection.

# Object detection

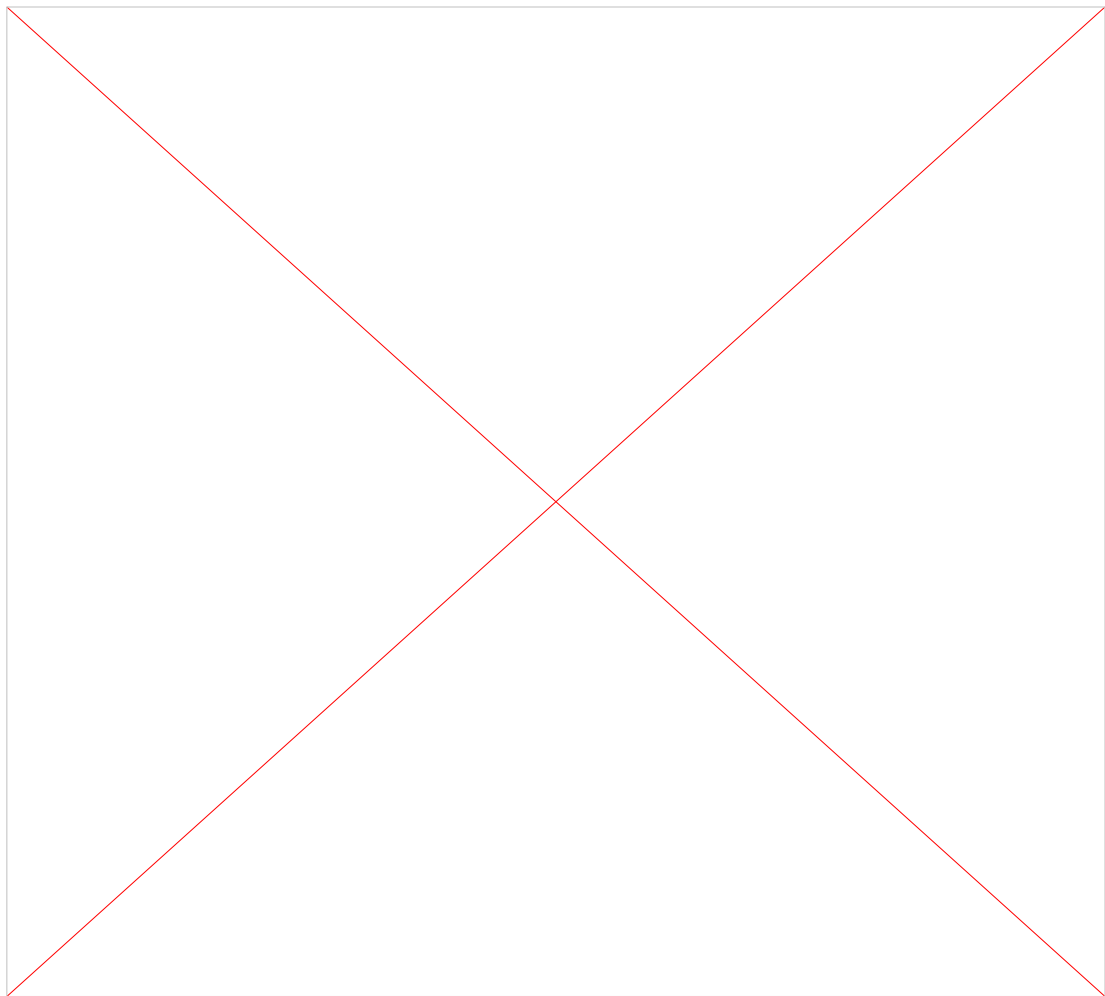
7x7 cells

Output for each cell:

2 bounding boxes + confidences

20 classes

→ 7x7x30



# Object detection

Much faster! (200 fps)

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

Localisation is less accurate.



# Object detection

YoloV2, YoloV3...

# Object detection

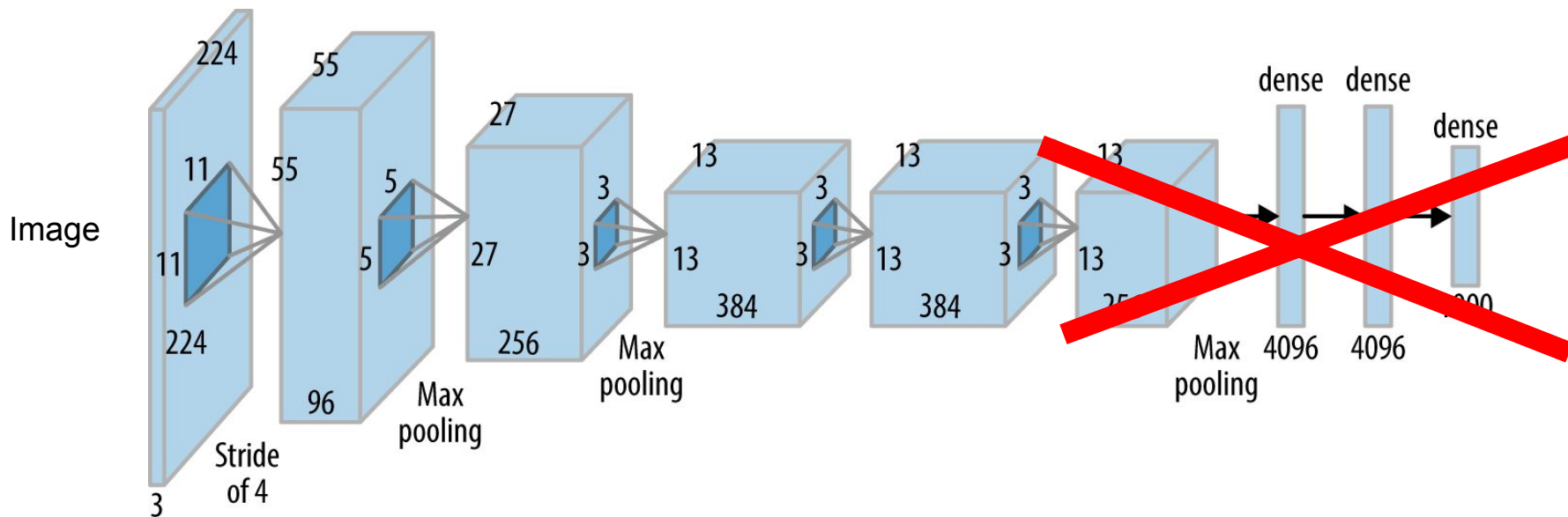
RetinaNet: state of the art (works well with dense and small scale objects)

Two main ideas...

- Feature Pyramid Network
- Focal Loss

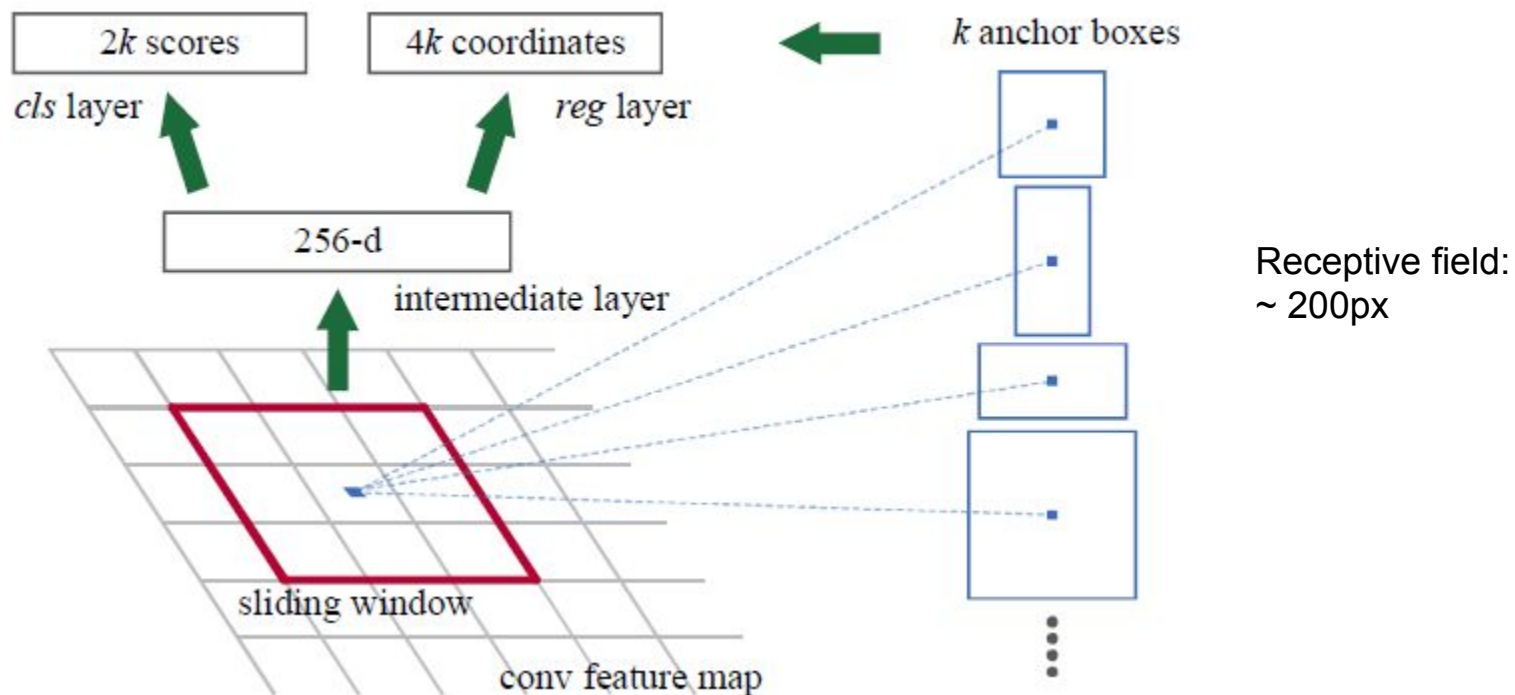
# Object detection

New concept: Region Proposal Network (RPN)

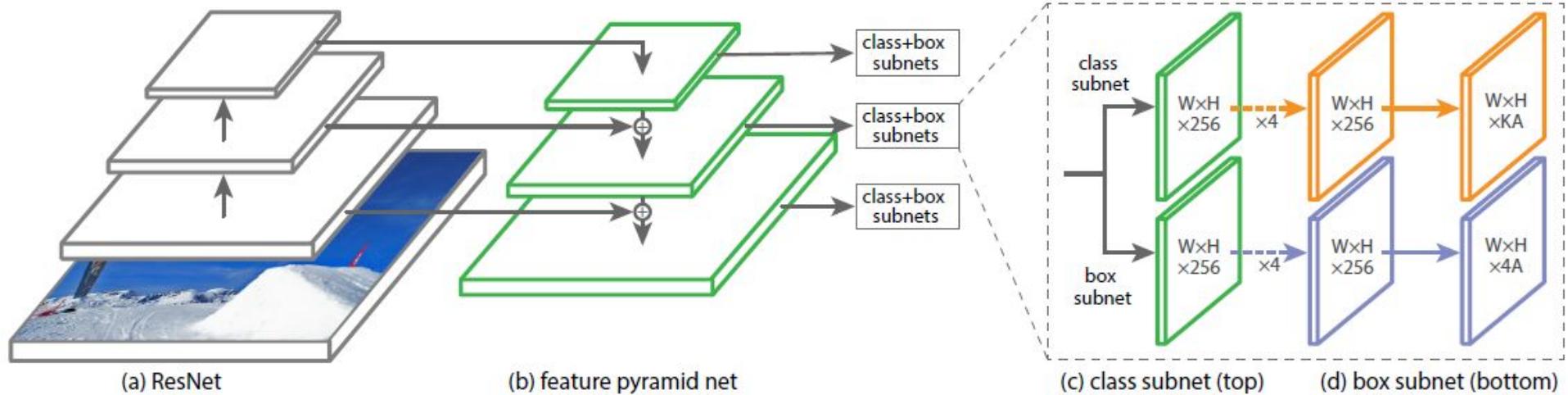


# Object detection

New concept: Region Proposal Network (RPN)



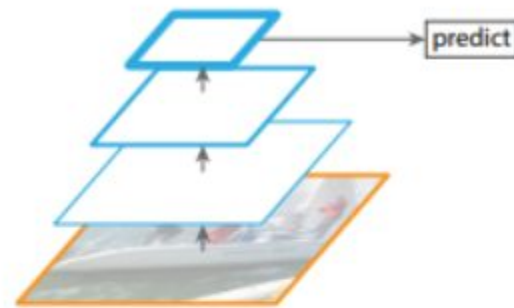
# Object detection



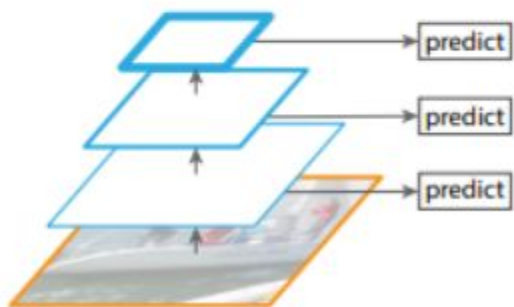
# Different types of pyramid architectures



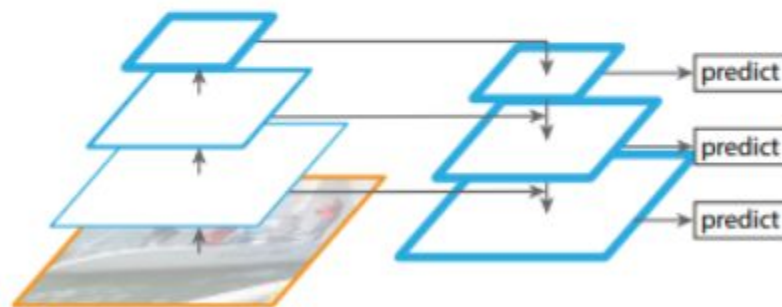
(a) Featurized image pyramid



(b) Single feature map



(c) Pyramidal feature hierarchy



(d) Feature Pyramid Network

# Object detection

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

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.

# Instance segmentation

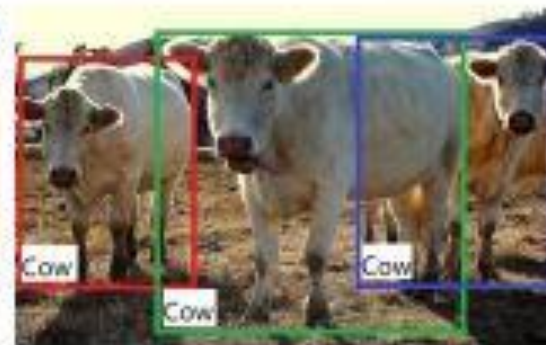


# Instance segmentation

Instance segmentation =  
Object detection +  
Semantic Segmentation



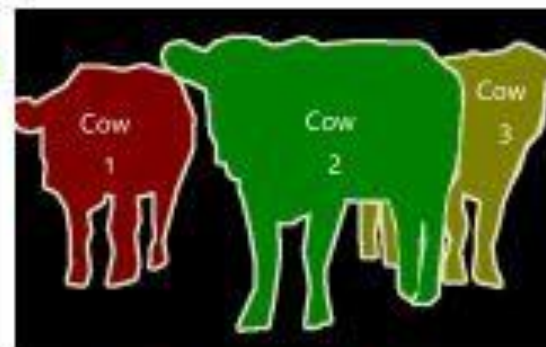
(a) Image Classification



(b) Object Detection



(c) Semantic Segmentation



(d) Instance Segmentation

# Instance segmentation

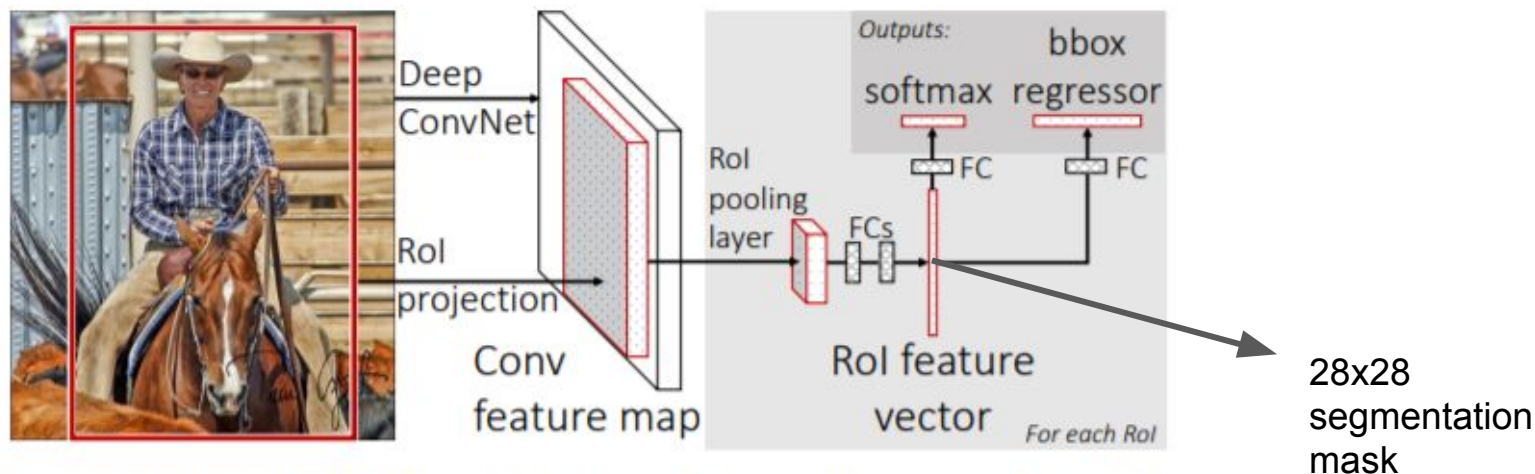


Figure 1. Fast R-CNN architecture. An input image and multiple regions of interest (RoIs) are input into a fully convolutional network. Each RoI is pooled into a fixed-size feature map and then mapped to a feature vector by fully connected layers (FCs). The network has two output vectors per RoI: softmax 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](https://github.com/matterport/Mask_RCNN/blob/master/samples/coco/inspect_model.ipynb)

# Today's plan

See in detail a solution for an advanced computer vision task

- Human pose estimation with OpenPose

Take a look at inverse problems with CNN

- CNN regression for image denoising
- Deep Image Prior: an atypical use of CNN

Missing here

- Generative Adversarial Networks
- Triplet losses
- ...

# 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](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>



Source: <http://www.youtube.com/watch?v=YGQ2lwAgng>

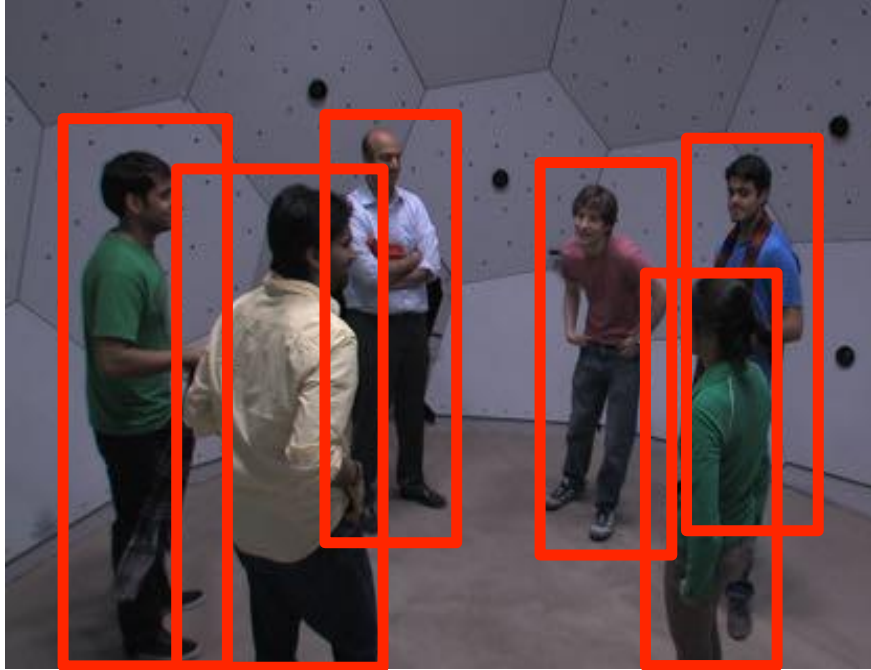
# Top-down Approach: Person Detection + Pose Estimation



Top-down



# Top-down Approach: Person Detection + Pose Estimation



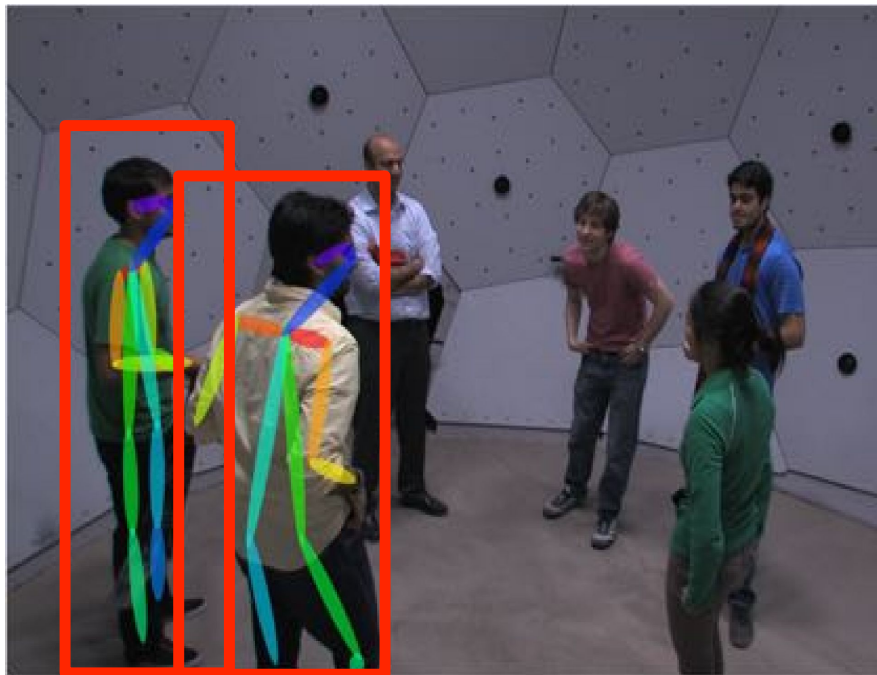
Top-down

# Top-down Approach: Person Detection + Pose Estimation



Top-down

# Top-down Approach: Person Detection + Pose Estimation



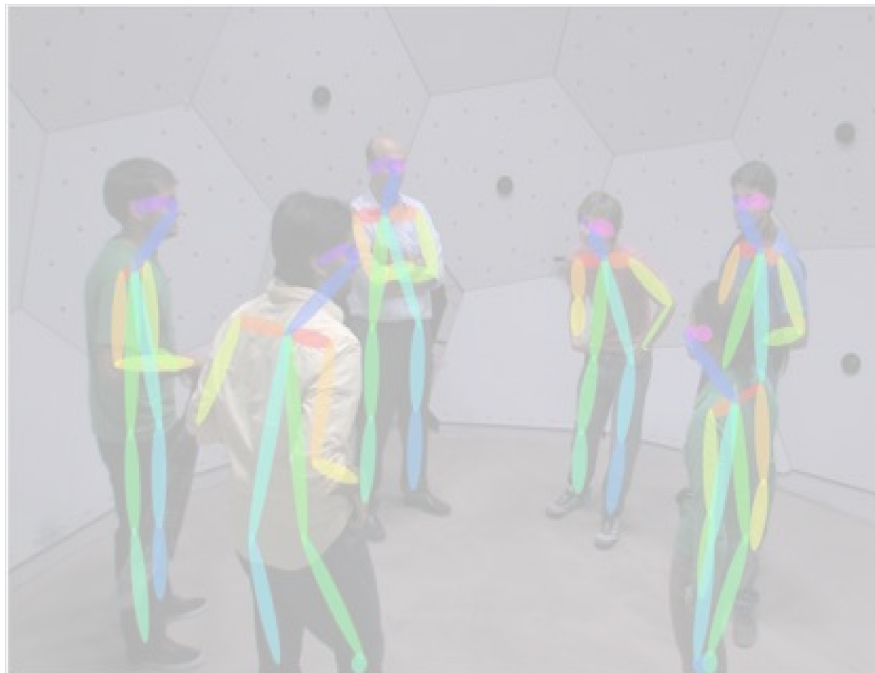
Top-down

# Top-down Approach: Person Detection + Pose Estimation



Top-down

# Openpose: Parts Detection + Parts Association

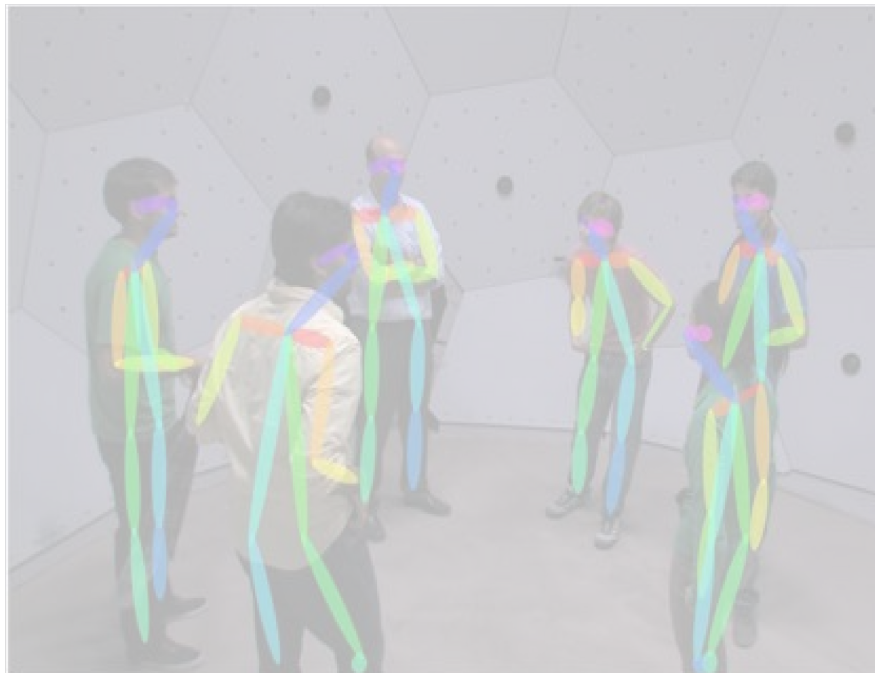


Top-down

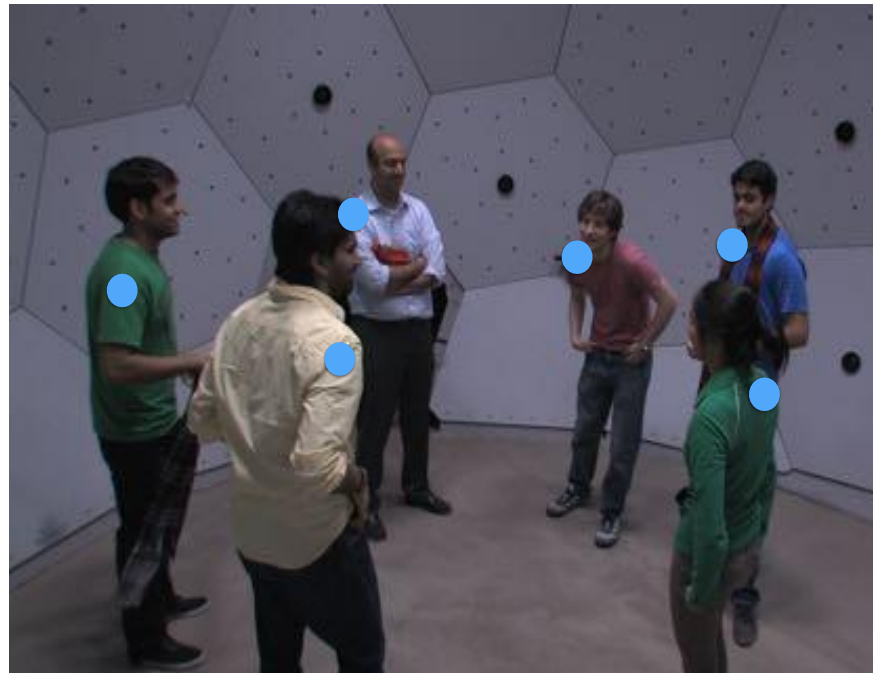


Openpose

# Openpose: Parts Detection + Parts Association

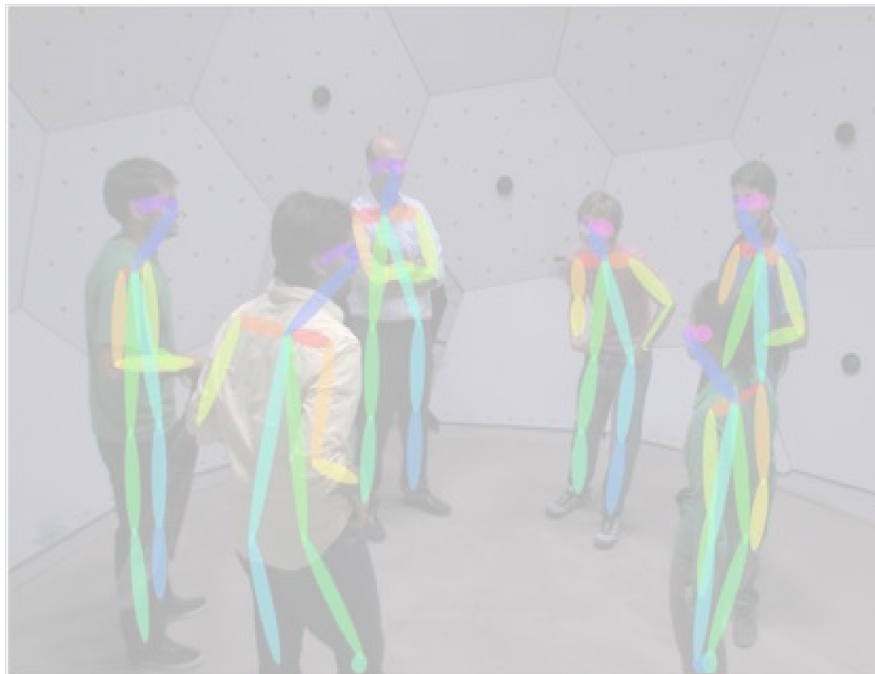


Top-down



Openpose

# Openpose: Parts Detection + Parts Association

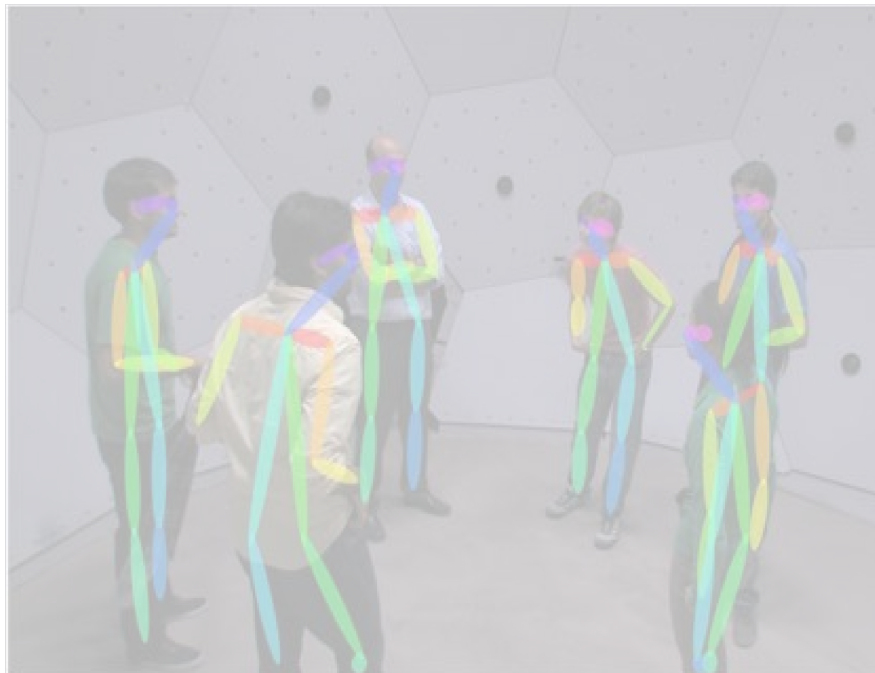


Top-down

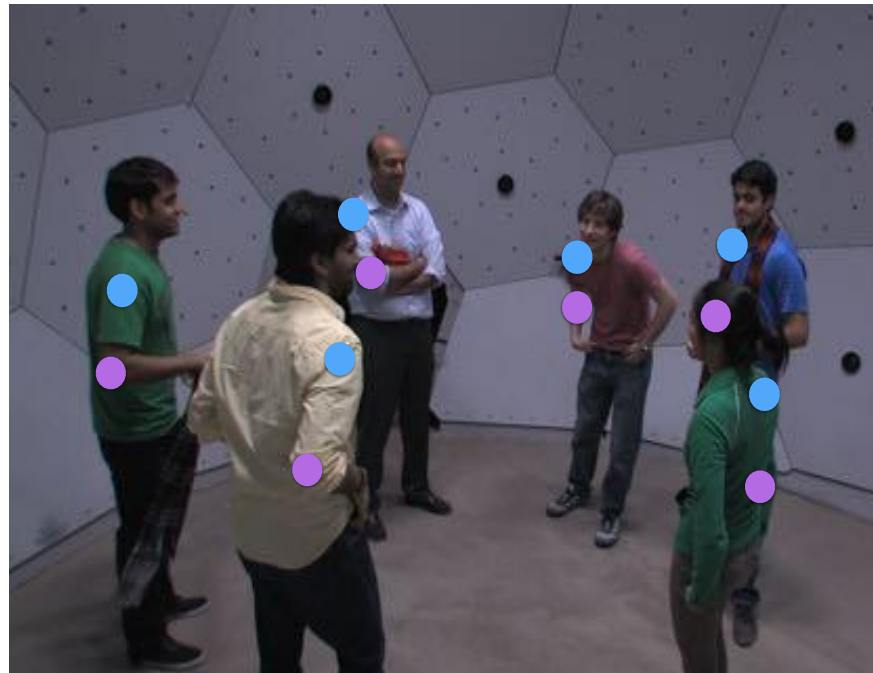


Openpose

# Openpose: Parts Detection + Parts Association



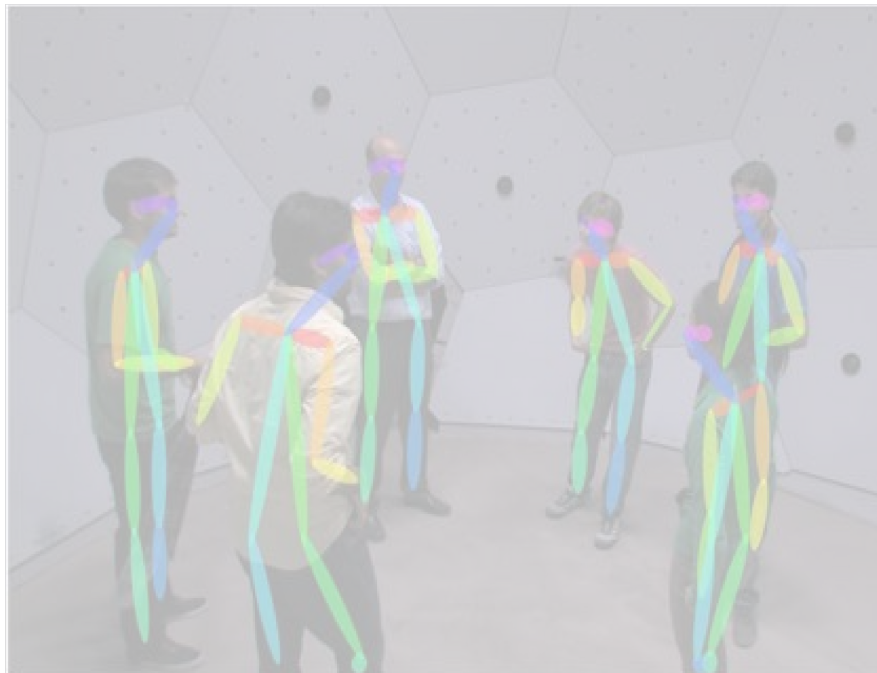
Top-down



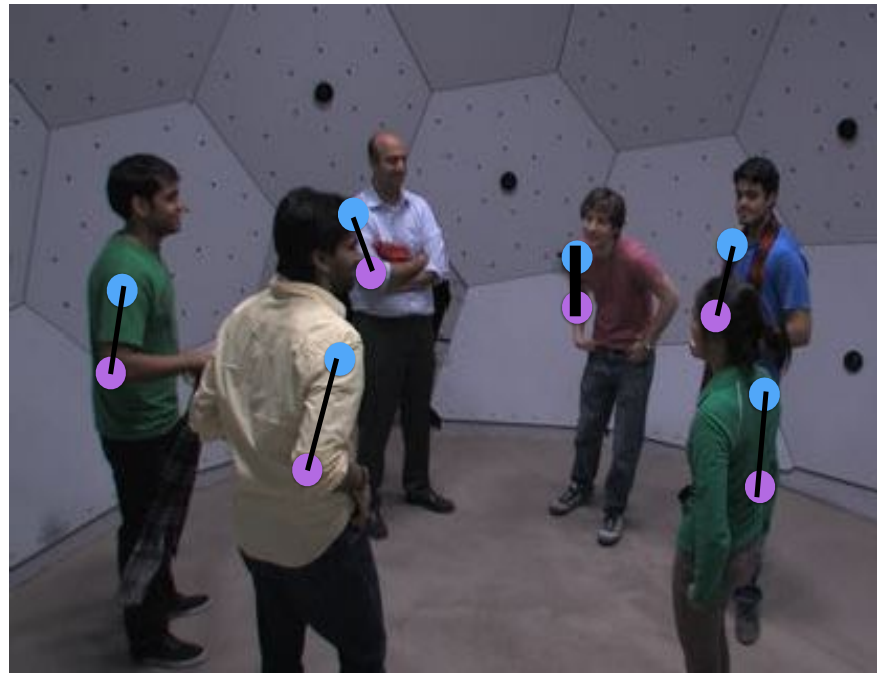
Openpose



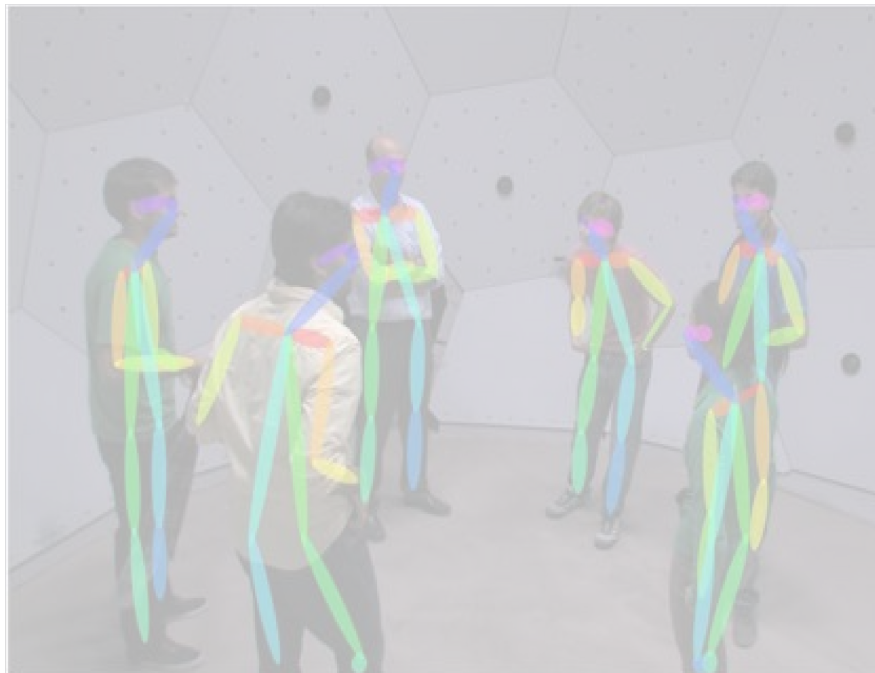
# Openpose: Parts Detection + Parts Association



Top-down



# Openpose: Parts Detection + Parts Association

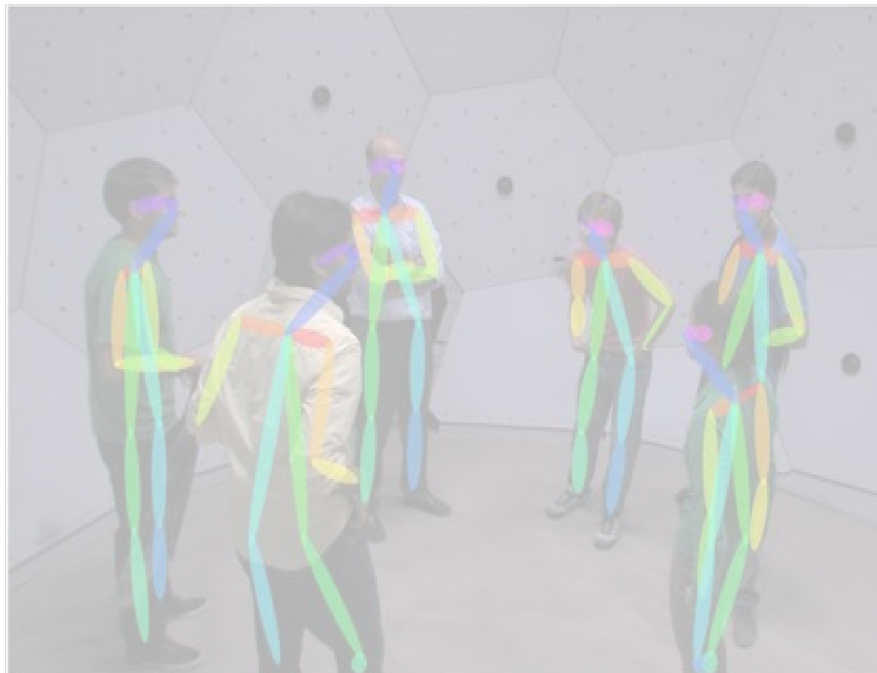


Top-down

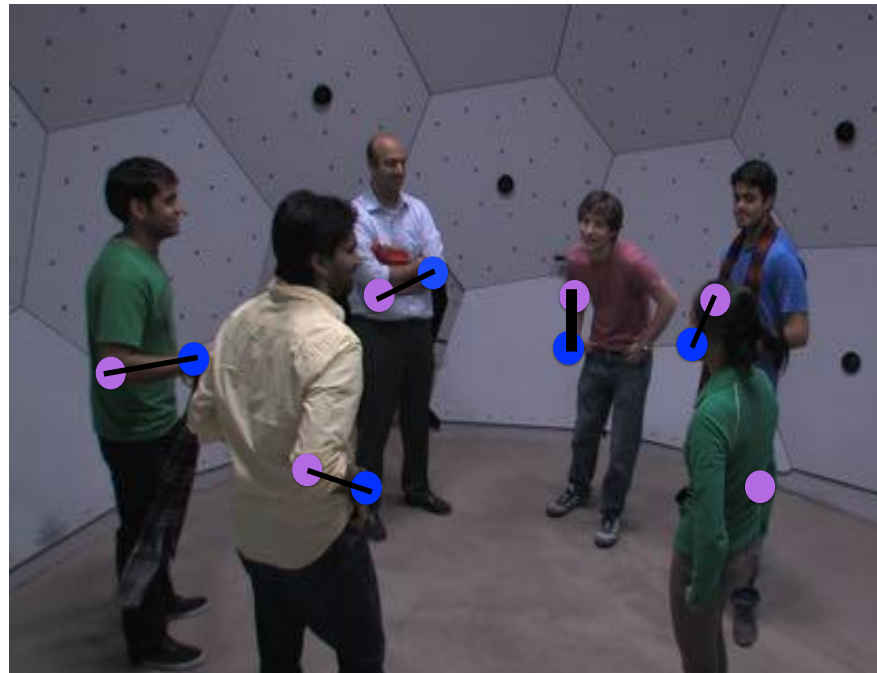


Part Affinity Fields

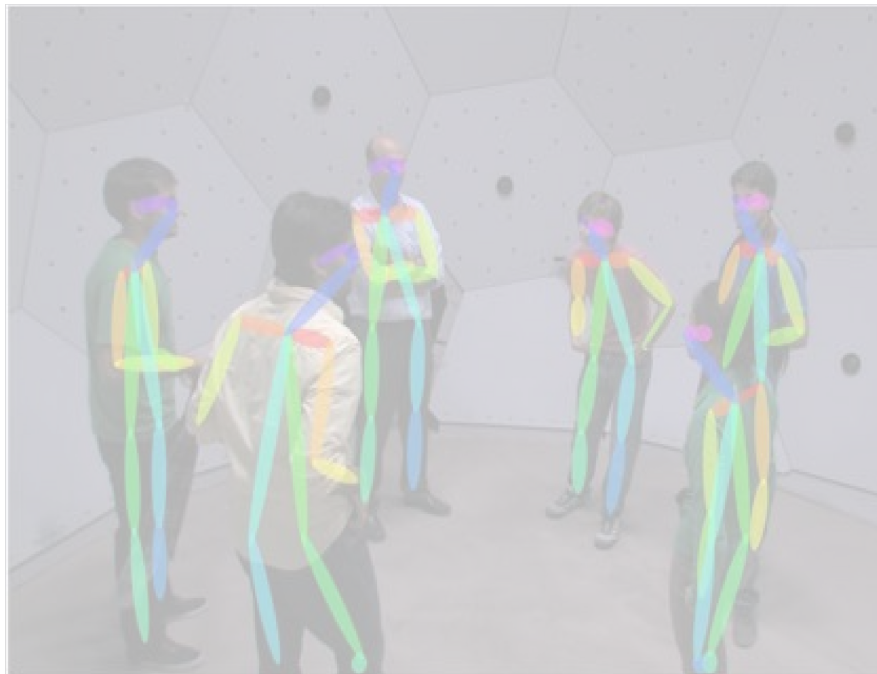
# Openpose: Parts Detection + Parts Association



Top-down



# Openpose: Parts Detection + Parts Association

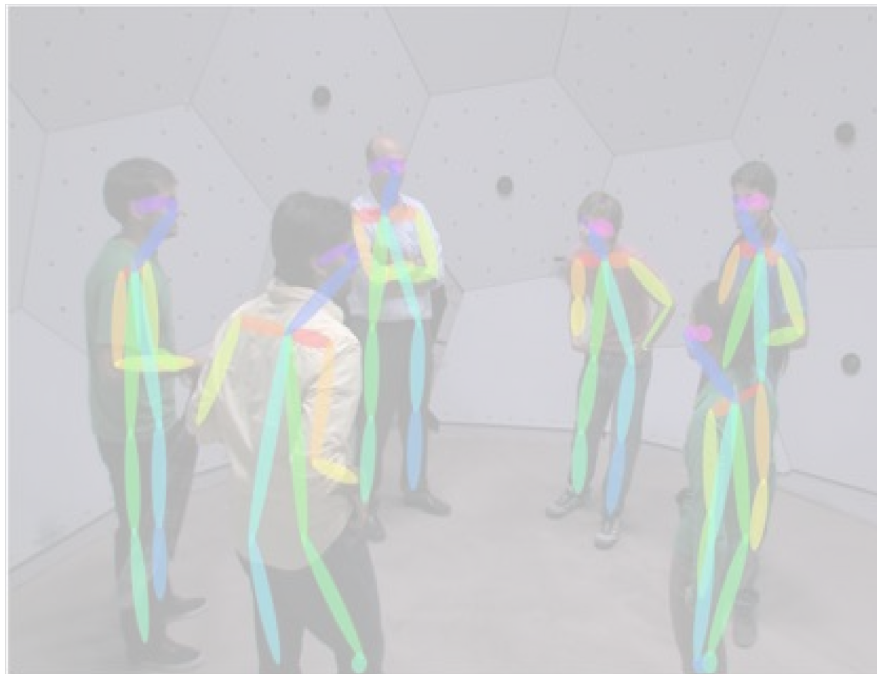


Top-down



Part Affinity Fields

# Openpose: Parts Detection + Parts Association

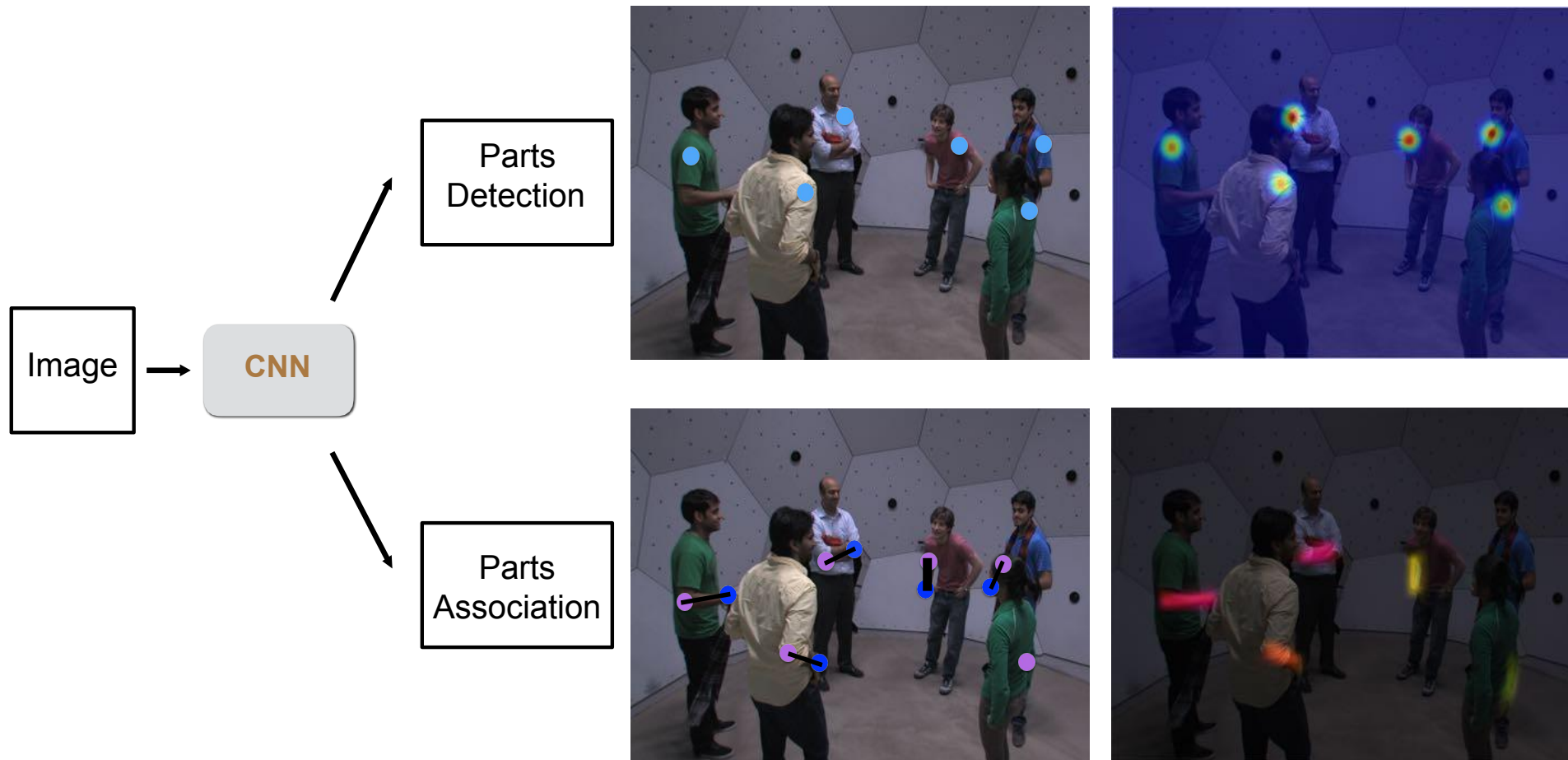


Top-down

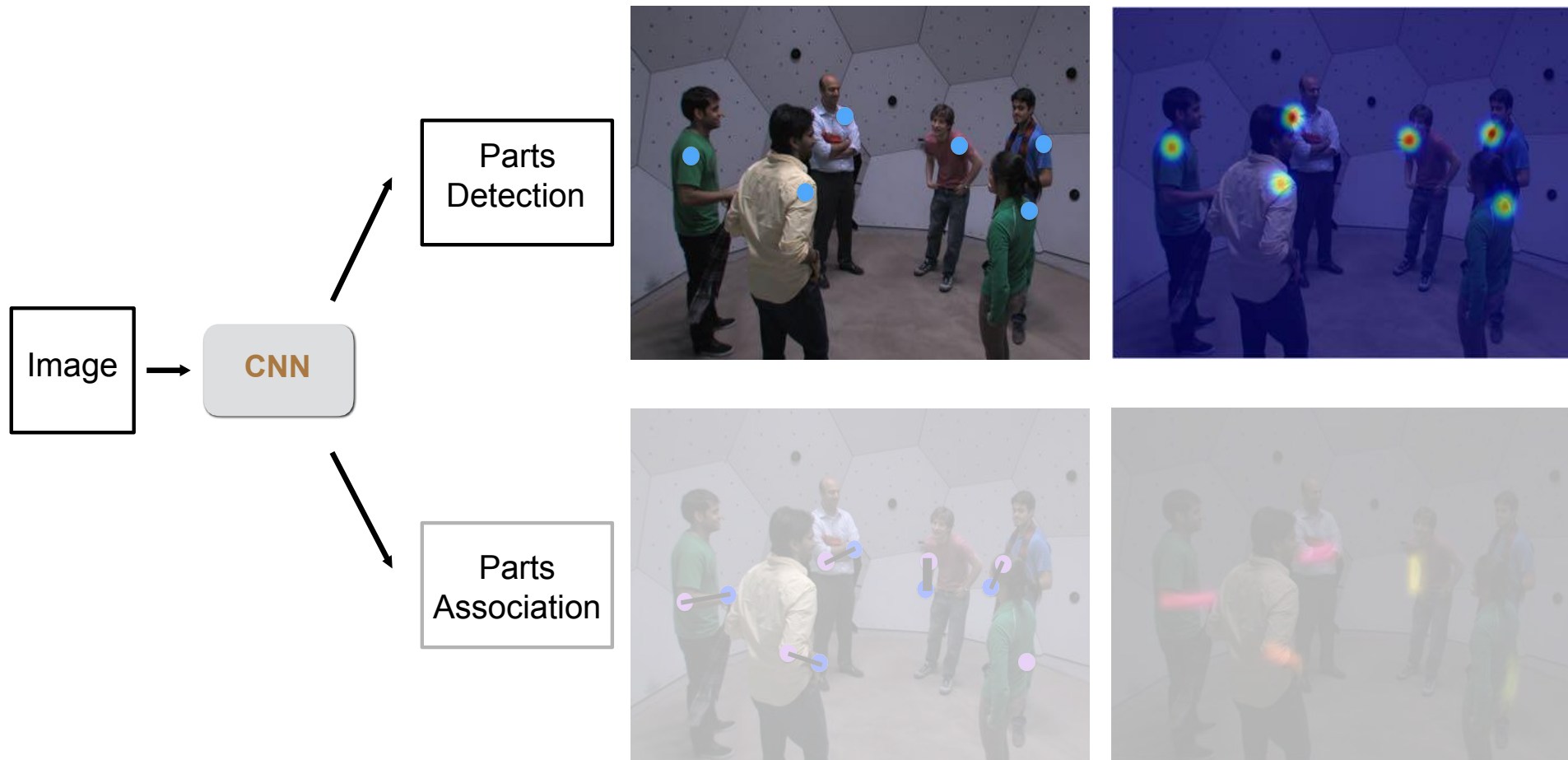


Openpose

# Novelty: Jointly Learning Parts Detection and Parts Association



# Novelty: Jointly Learning Parts Detection and Parts Association



# Sequential Prediction with Learned Spatial Context

Stage 1



CNN



*Right shoulder*



*Right wrist*



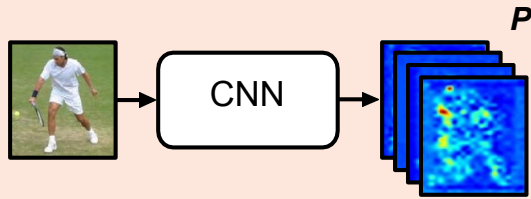
*Right knee*

⋮



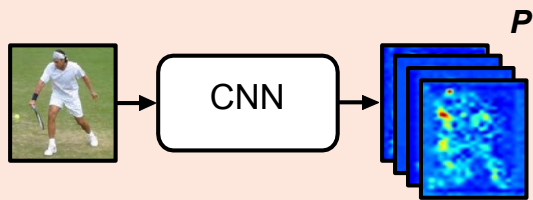
# Sequential Prediction with Learned Spatial Context

Stage 1



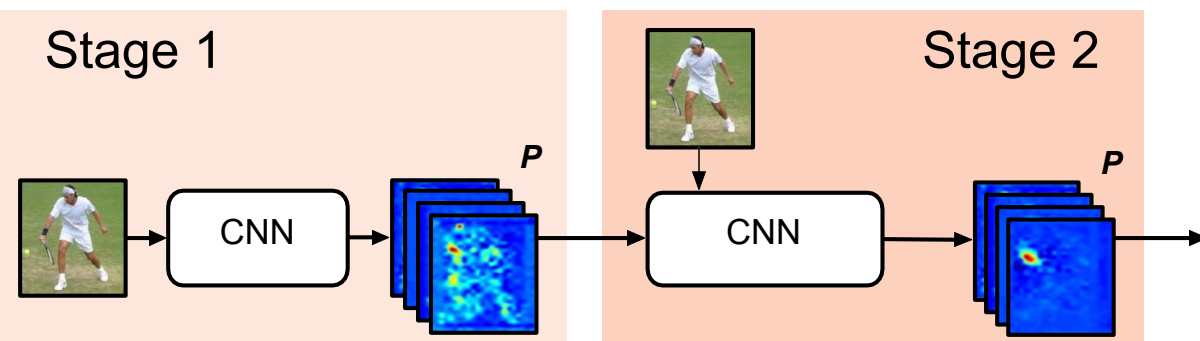
# Sequential Prediction with Learned Spatial Context

Stage 1



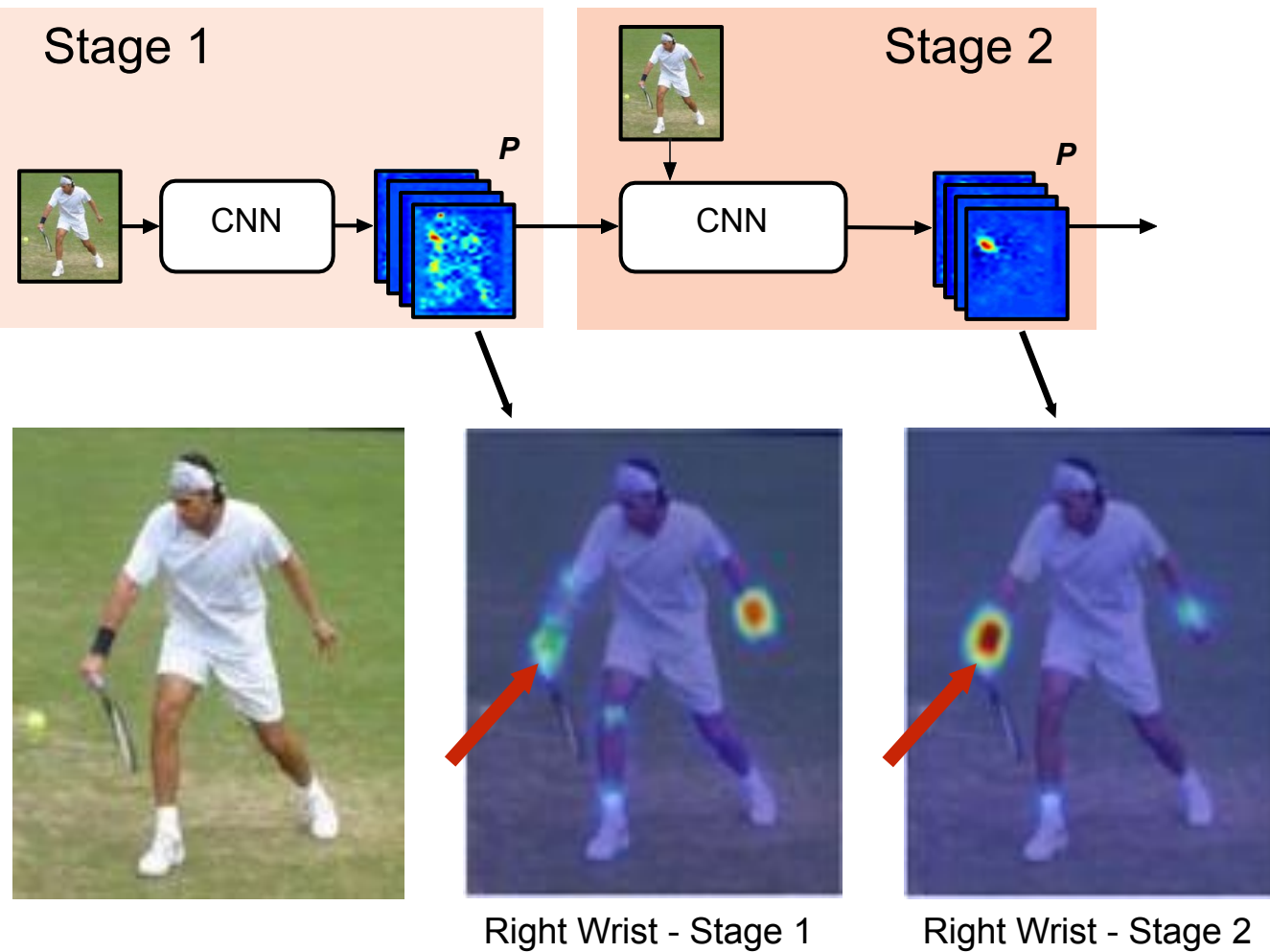
Right Wrist - Stage 1

# Sequential Prediction with Learned Spatial Context

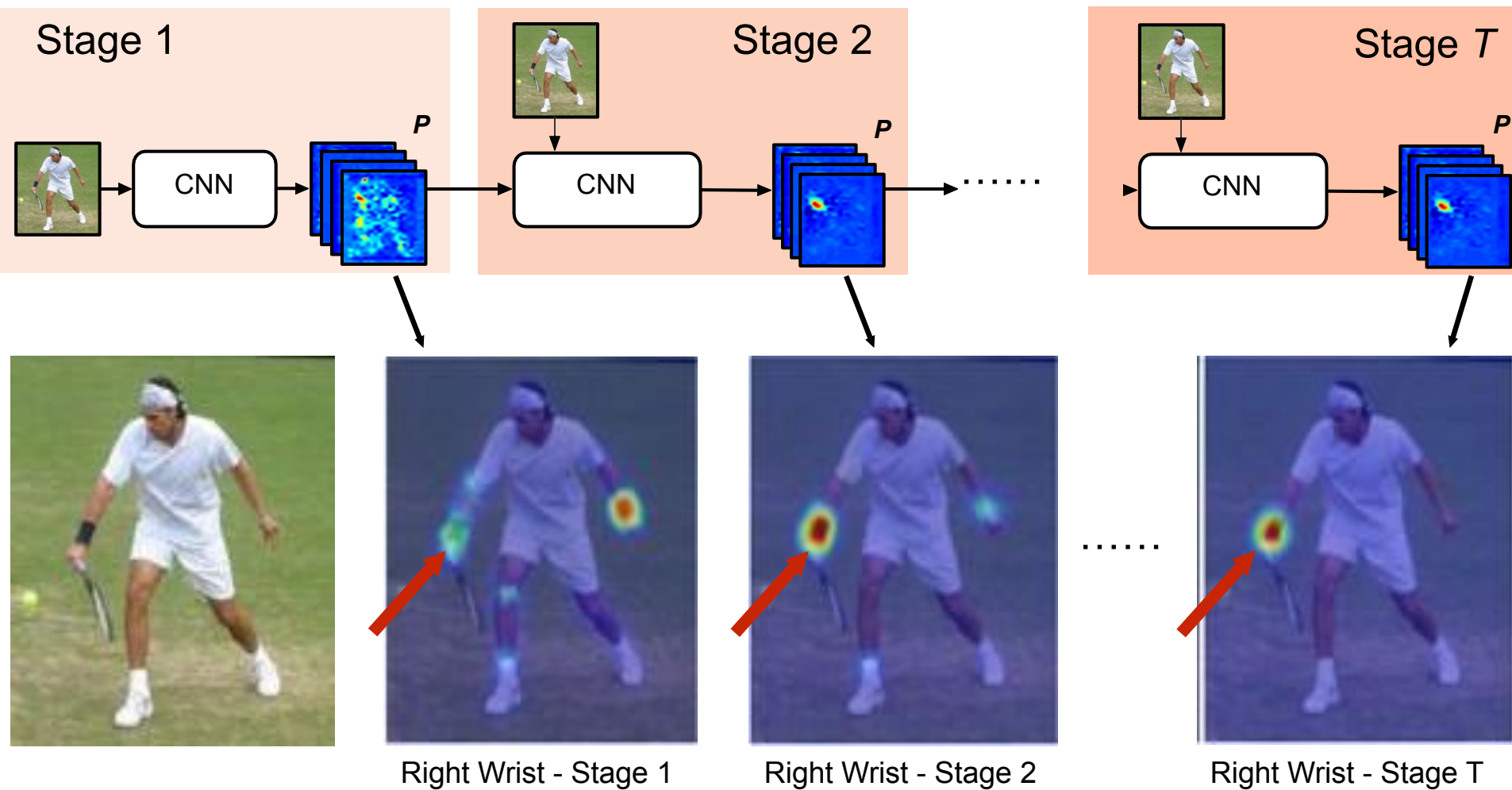


Right Wrist - Stage 1

# Sequential Prediction with Learned Spatial Context



# Sequential Prediction with Learned Spatial Context



## Parts Score Maps Prediction from Image Sequence

Input



Nose



Neck



Right Shoulder



Right Elbow

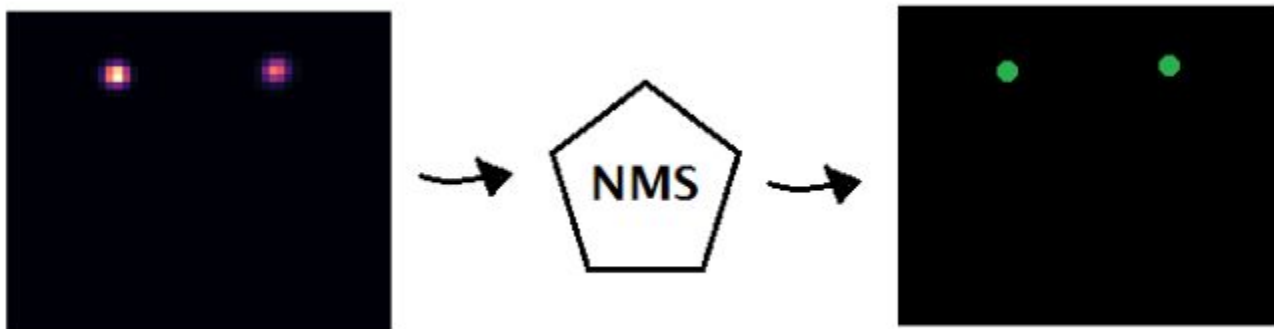


Right Wrist



## Exact part locations by non maximum suppression

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

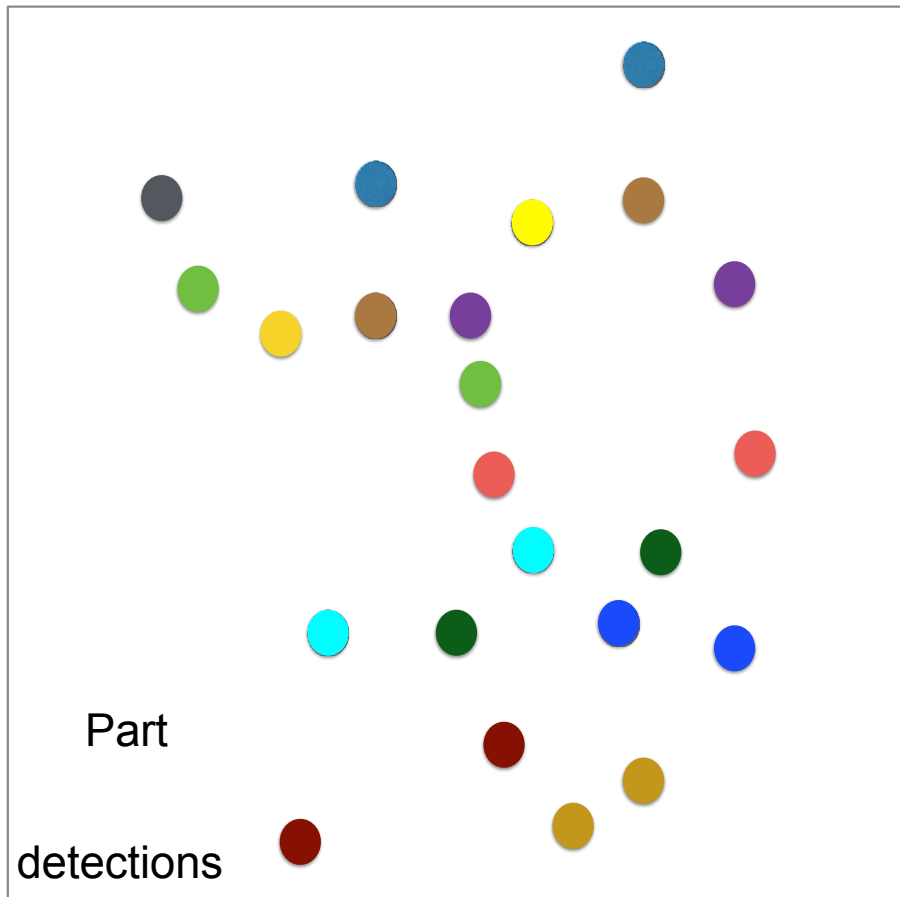


# Jointly Learning Parts Detection and Parts Association



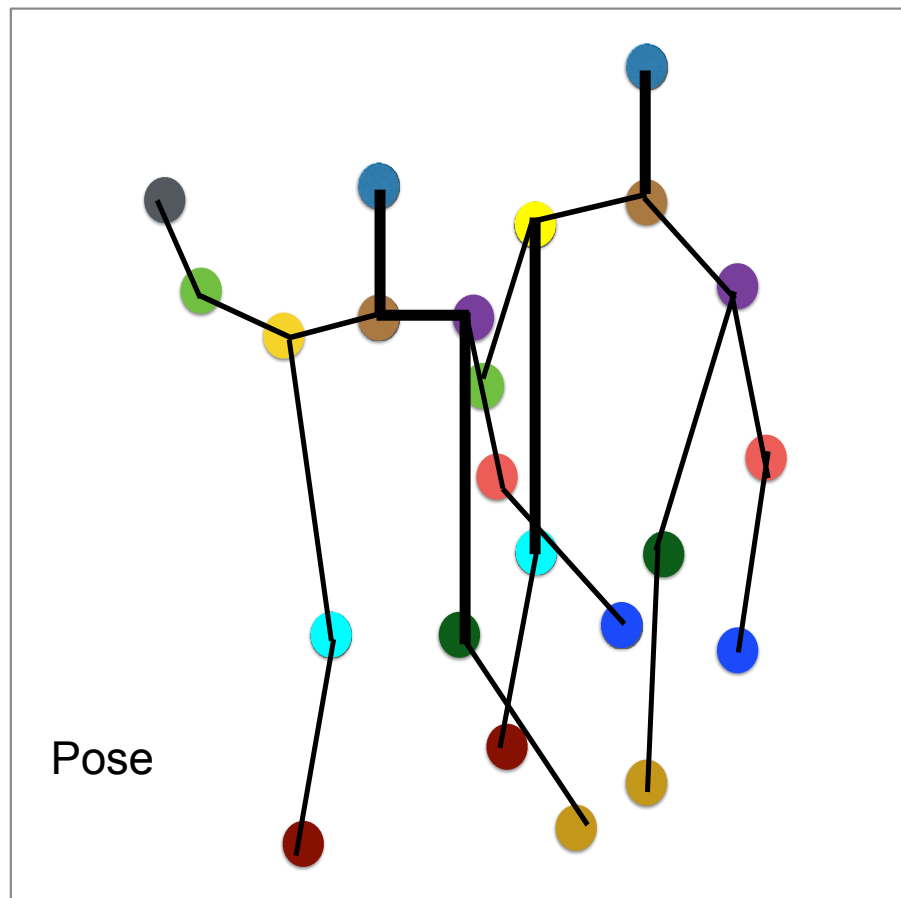
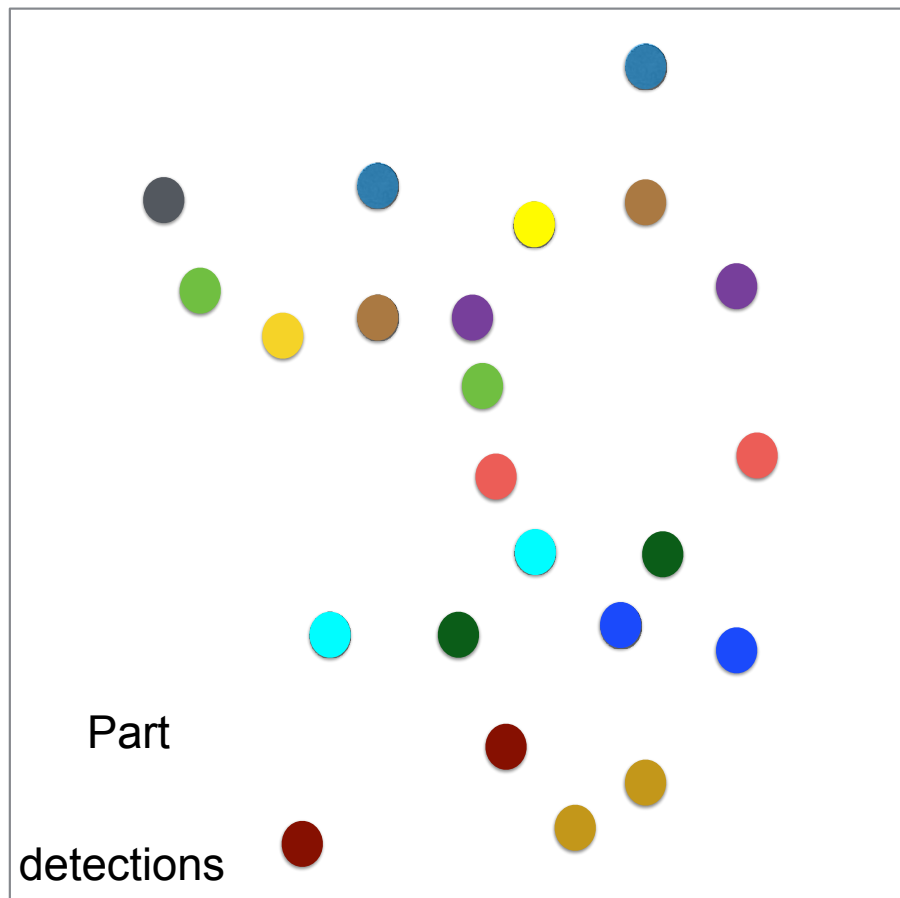


# Part-Person Association for Multi-Person Pose Estimation

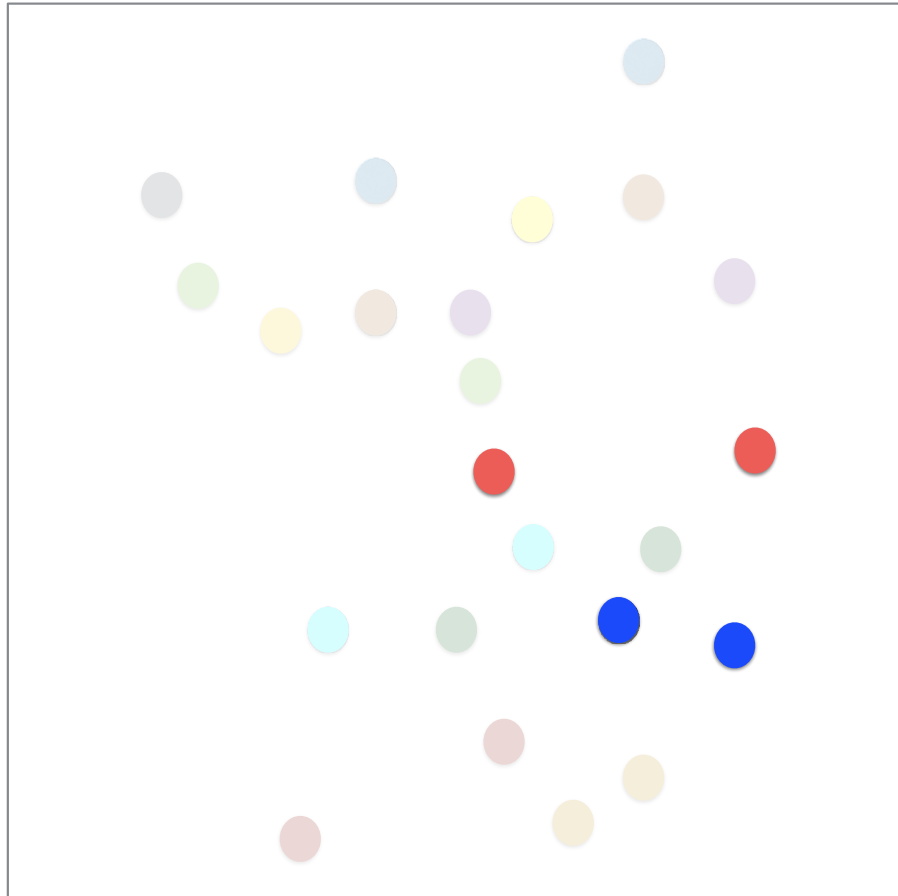


- Elbow
- Wrist
- ⋮

# Part-Person Association for Multi-Person Pose Estimation

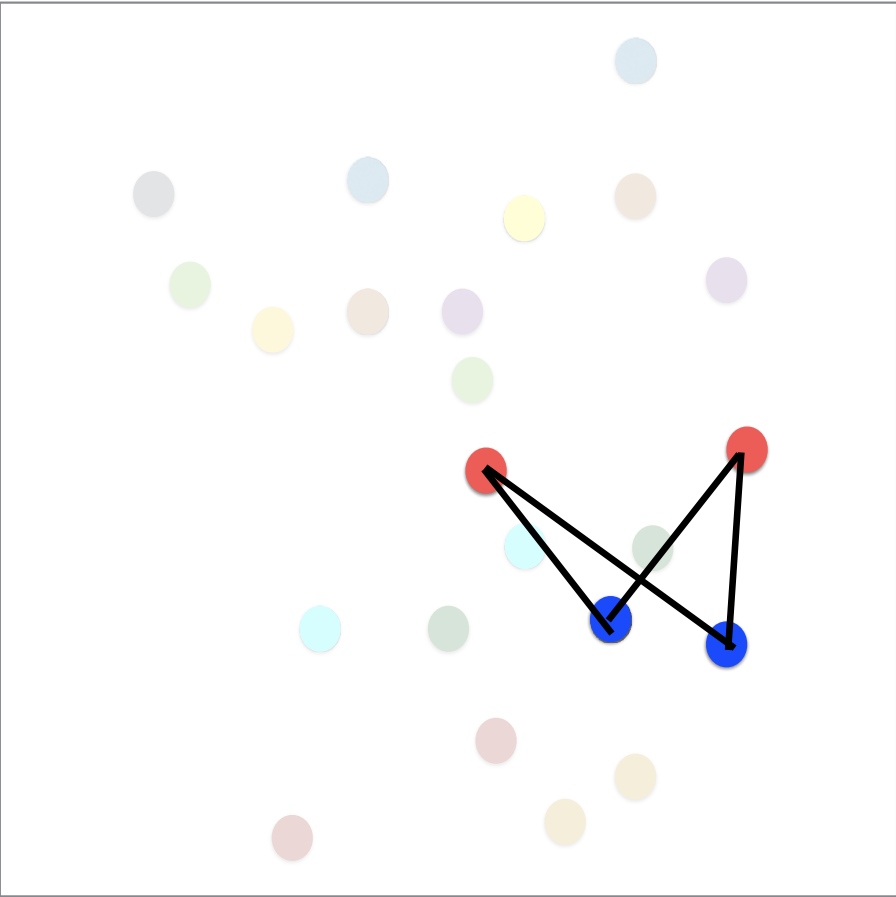


# Part-to-Part Association for Multi-Person Pose Estimation



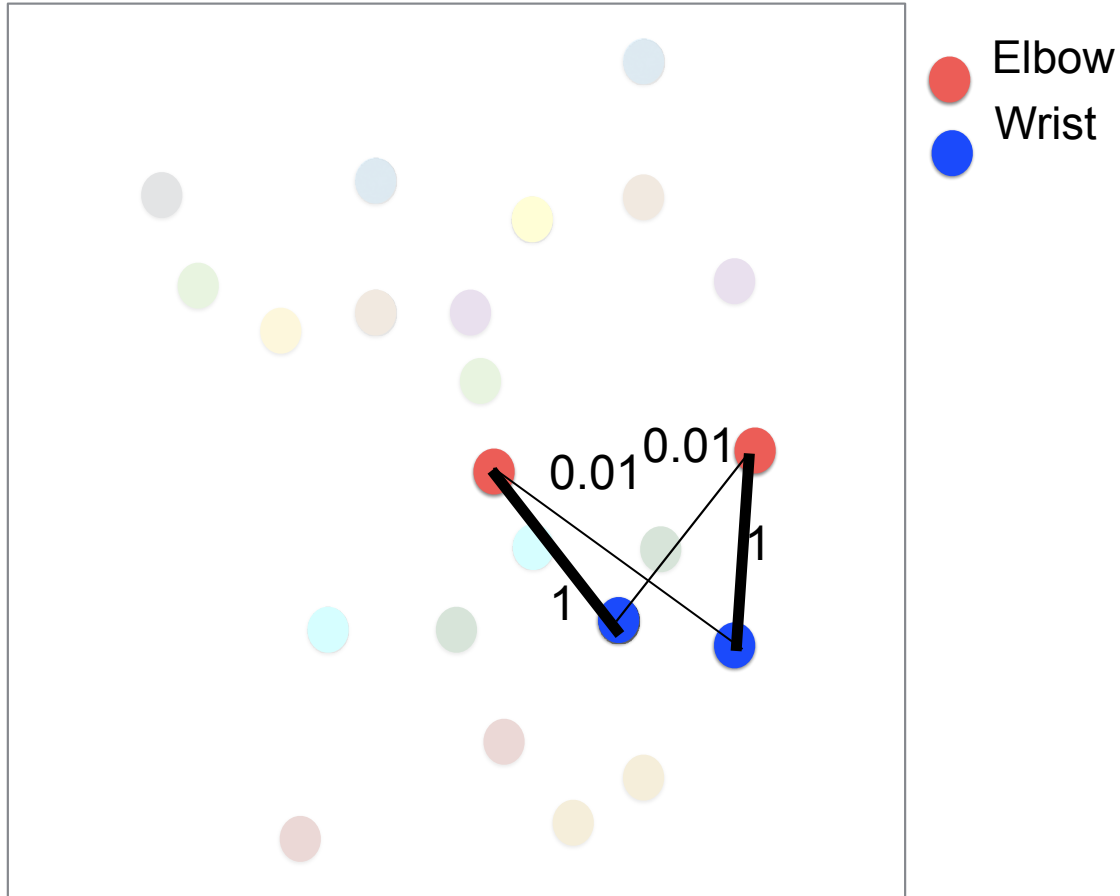
- Elbow
- Wrist

# Part Affinity Score Guides the Connection

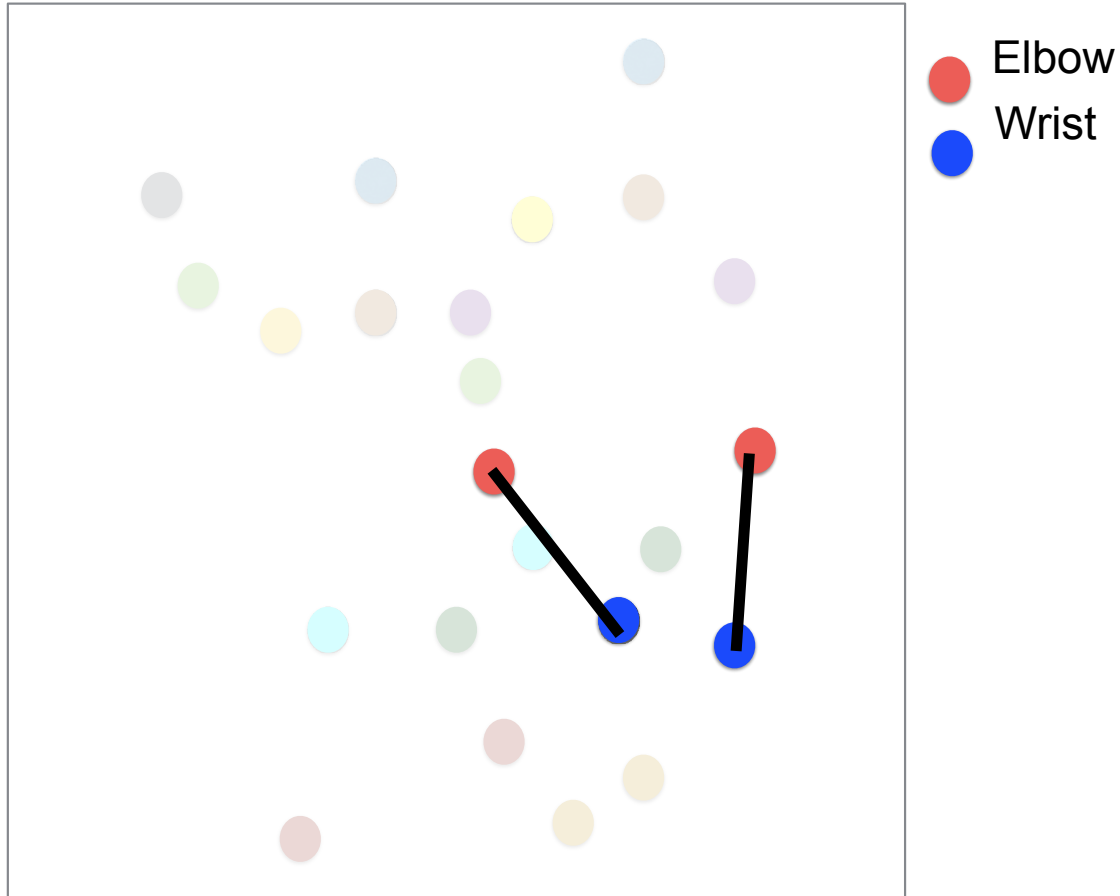


- Elbow
- Wrist

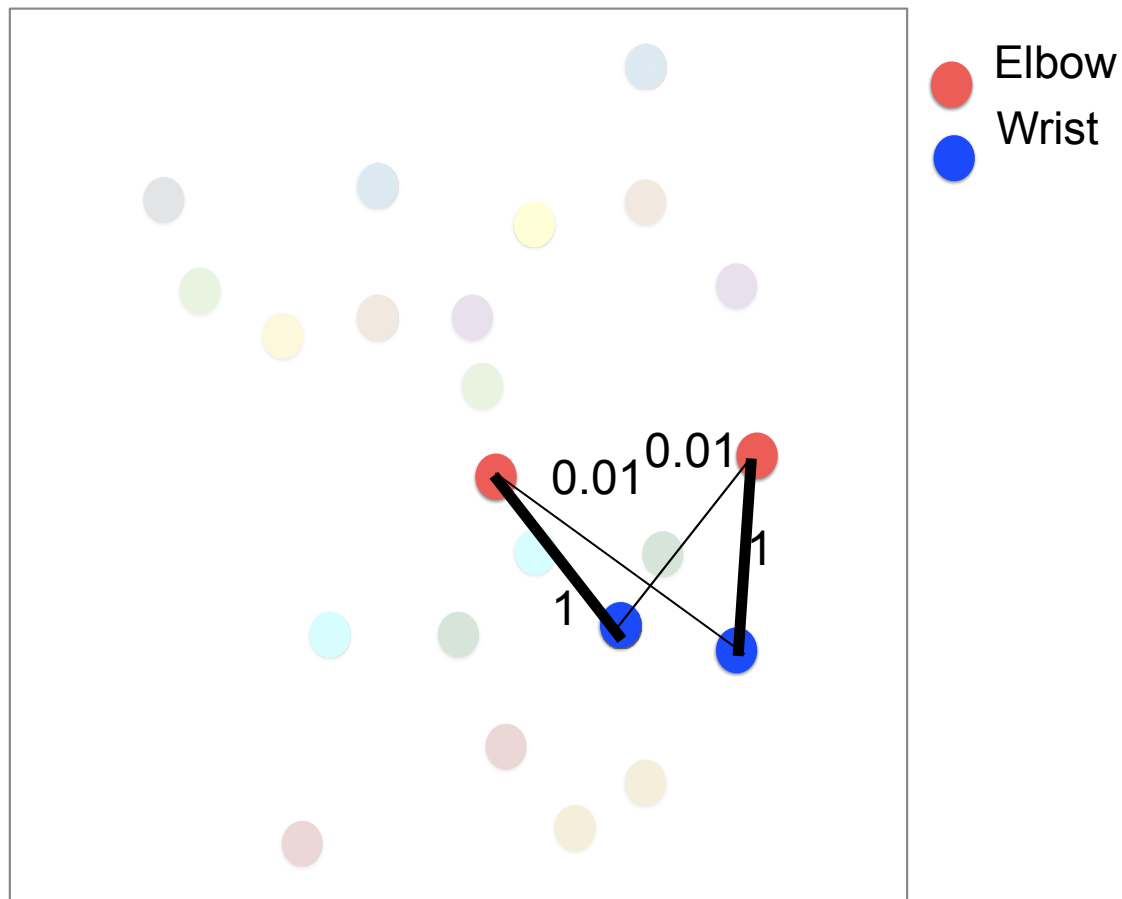
# Part Affinity Score Guides the Connection



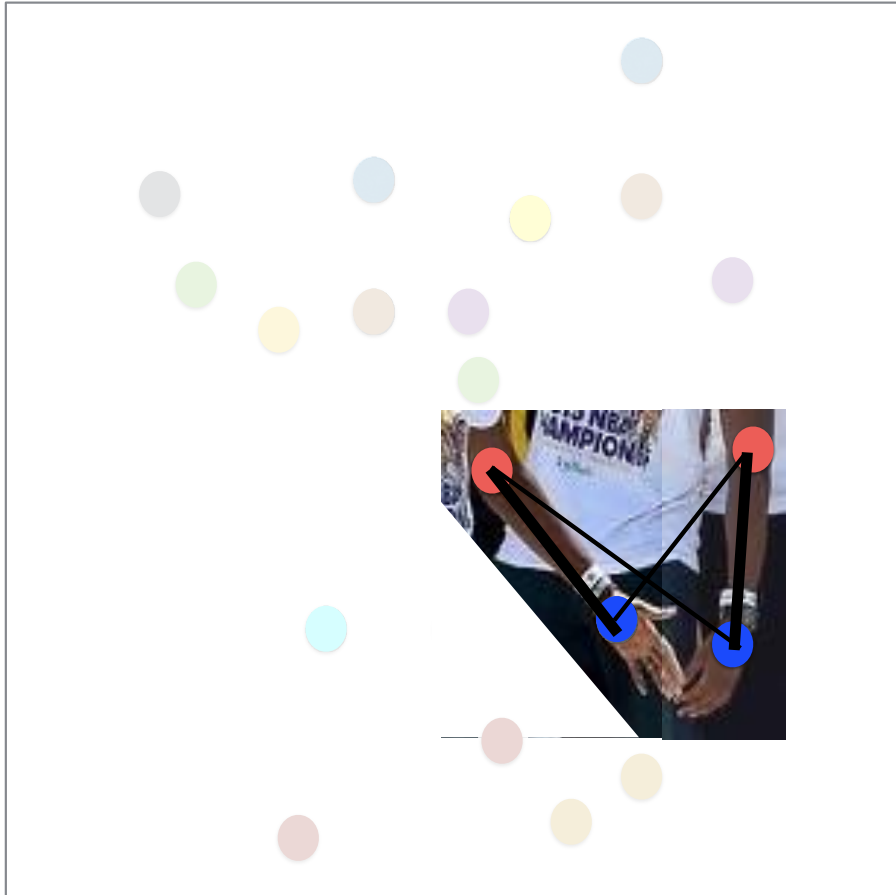
# Part Affinity Score Guides the Connection



# How to Obtain the Part Affinity Score

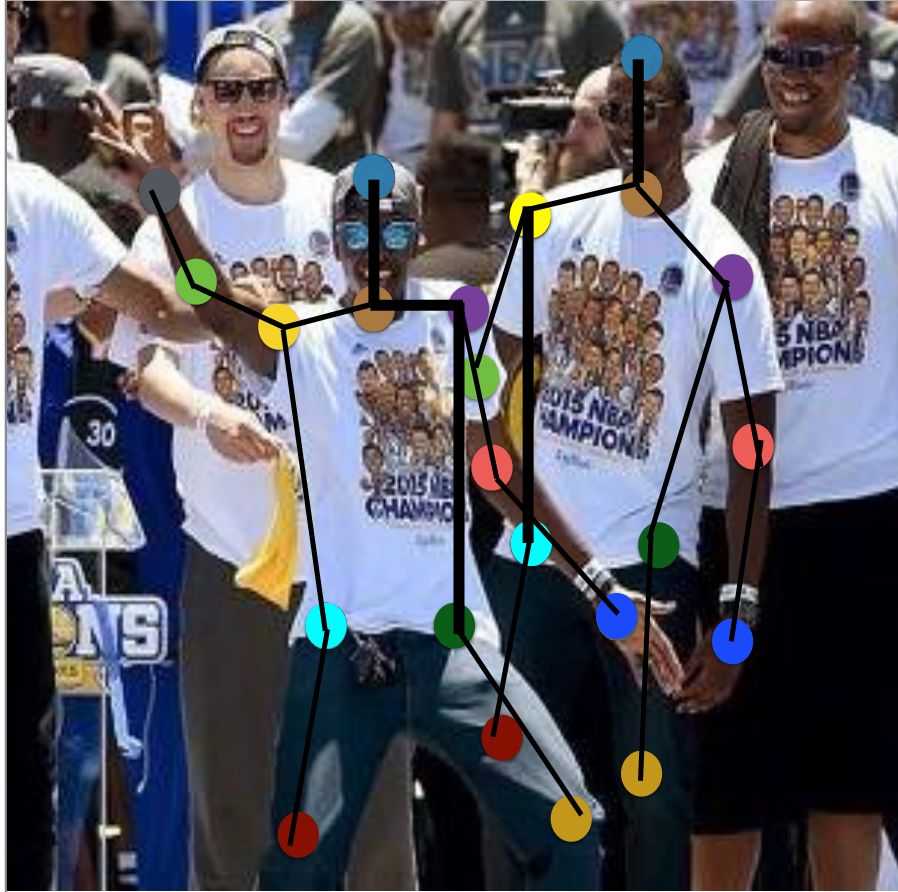


# Part Affinity Score is Dependent on Visual Appearance

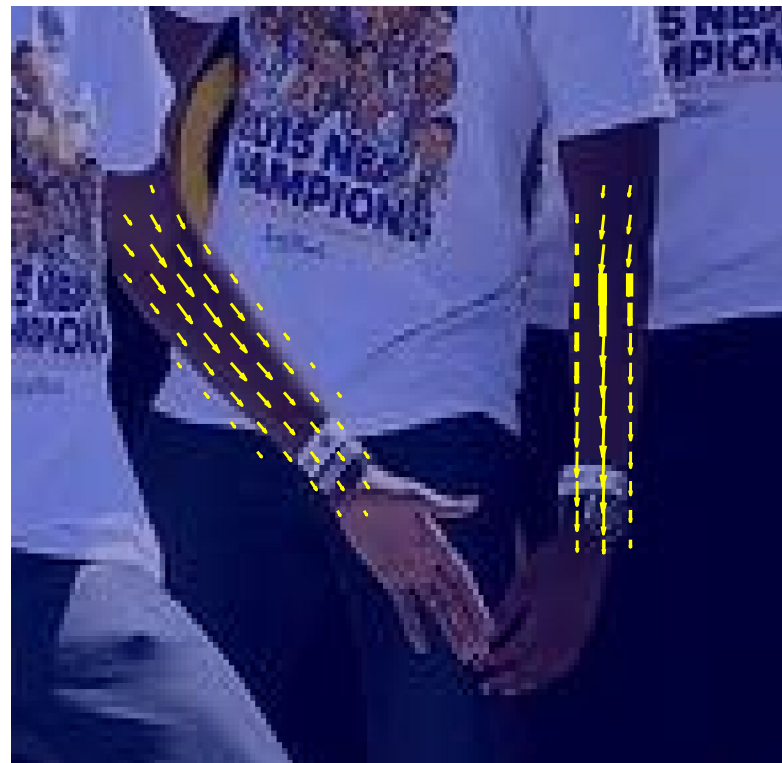
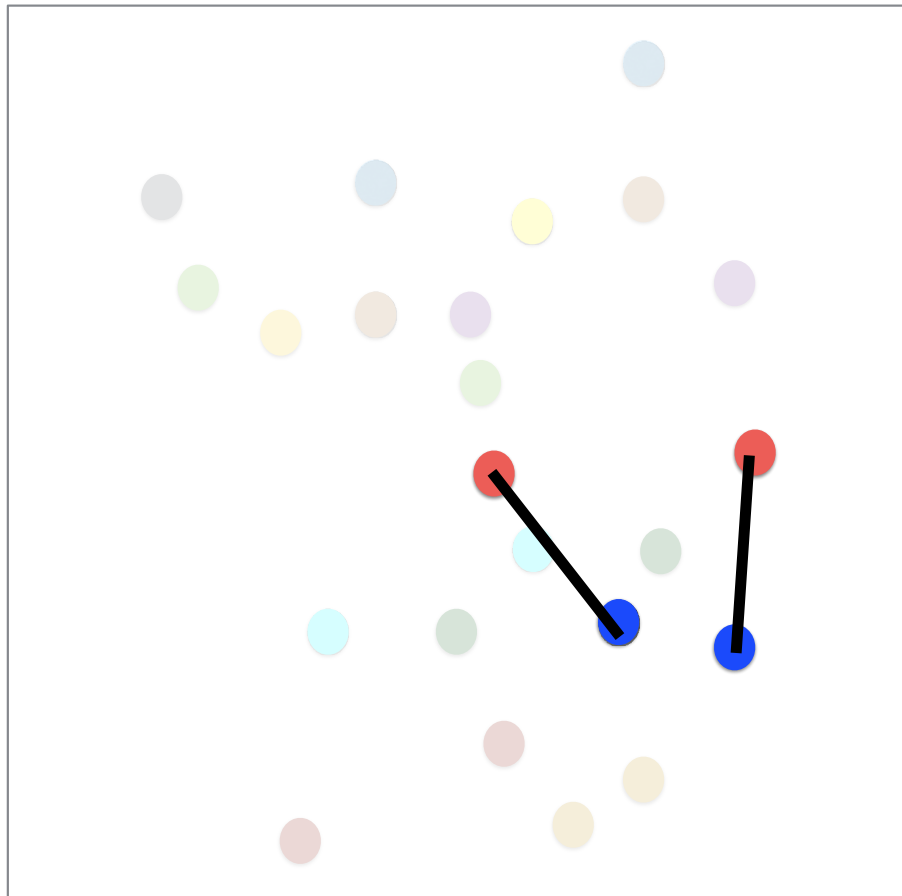




# Part Affinity Score is Dependent on Visual Appearance

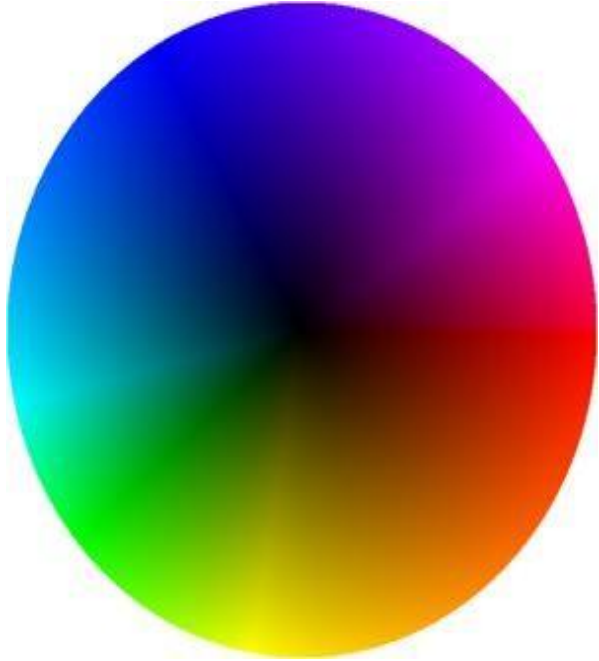


## Key Idea: Encode the Part Affinity Score on the Image Plane



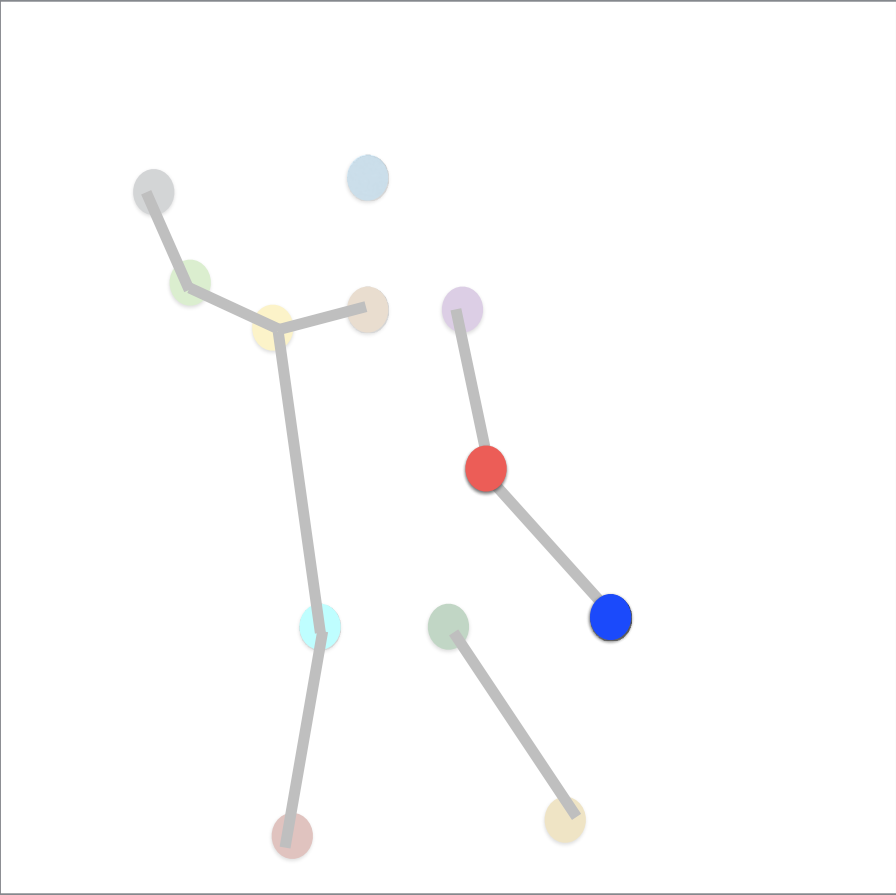
Part Affinity Fields  
encode **direction** and **position**

## 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



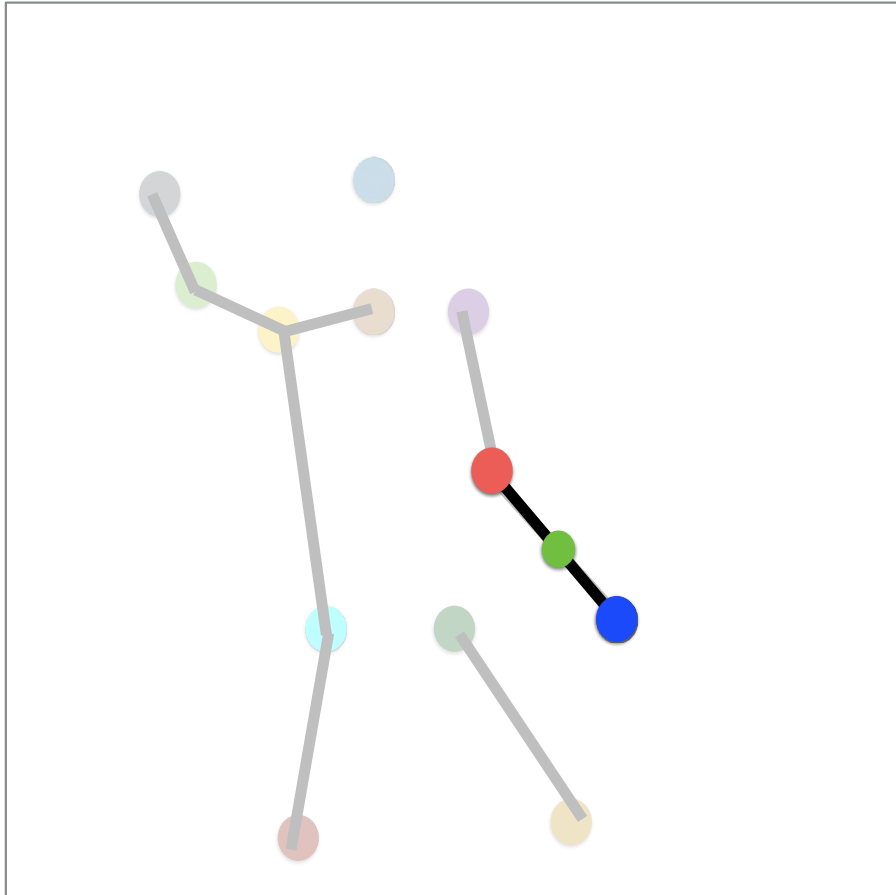
part1



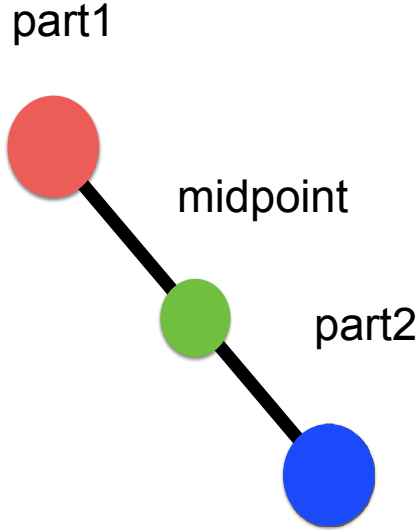
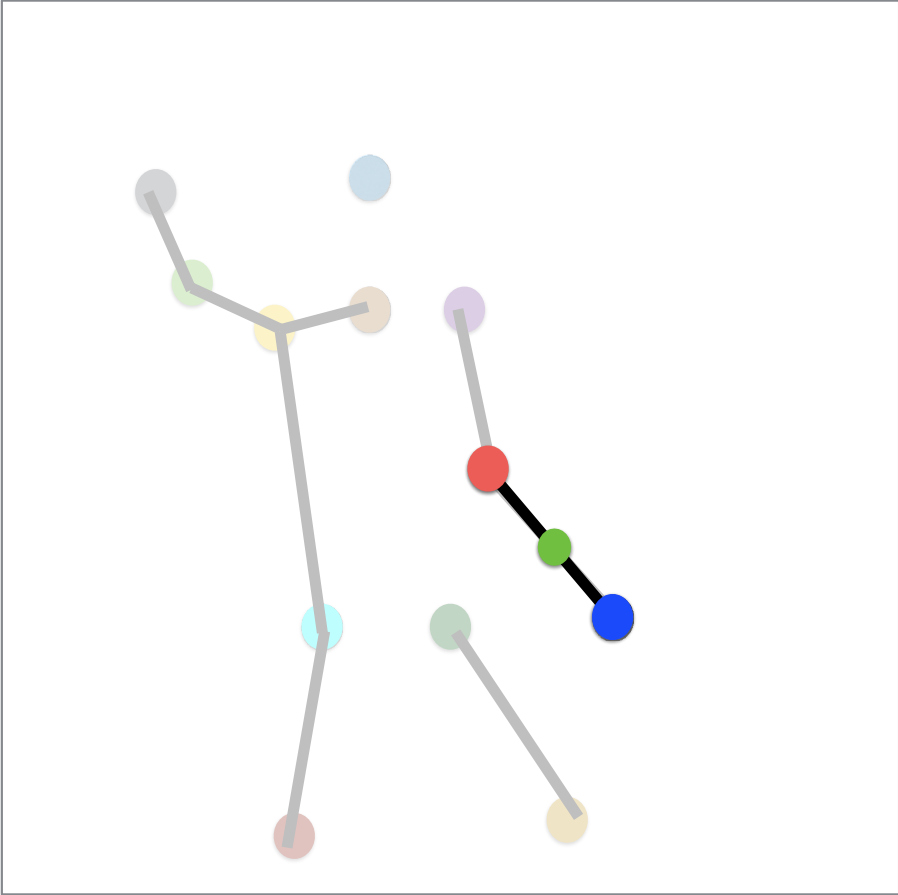
part2



# Midpoint Score Map for Part-to-Part Association

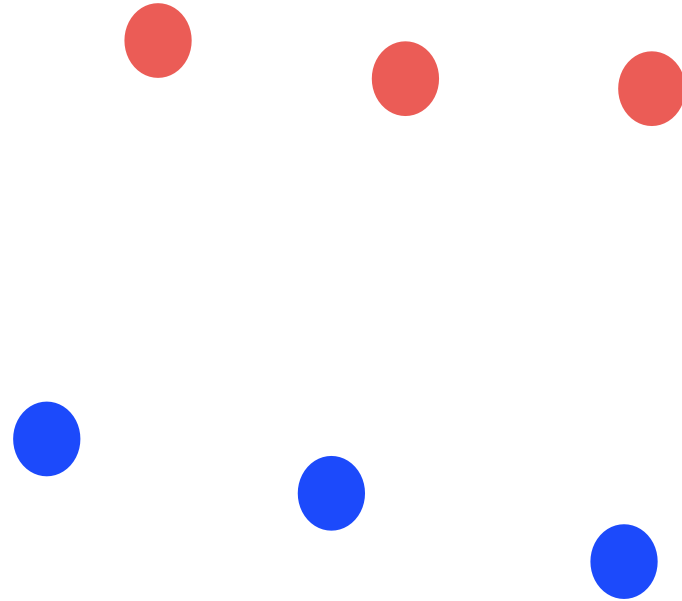


# Midpoint Score Map for Part-to-Part Association



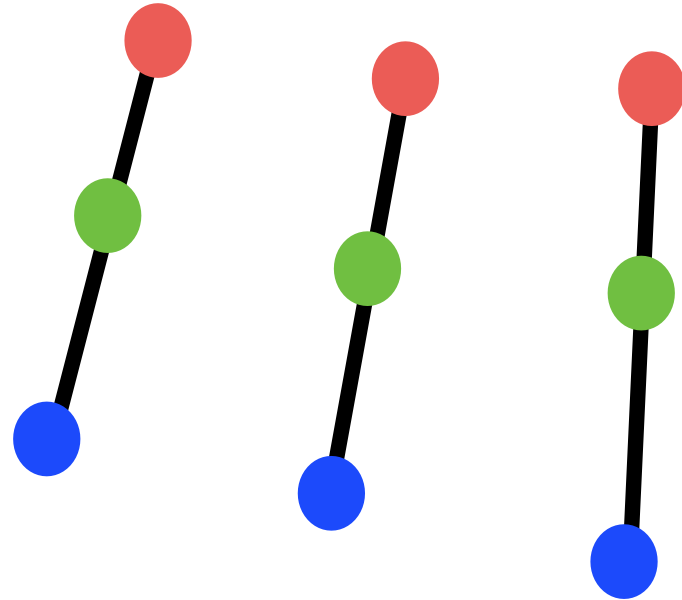
Affinity score between part1 and part2  
= confidence score of the midpoint

# Spatial Ambiguity of the Midpoint Representation



— Correct Connection

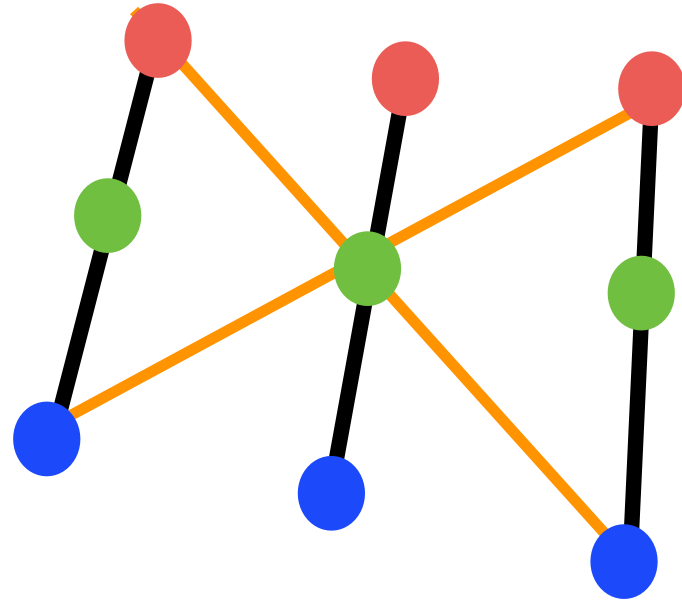
# Spatial Ambiguity of the Midpoint Representation



— Correct Connection  
— Wrong Connection

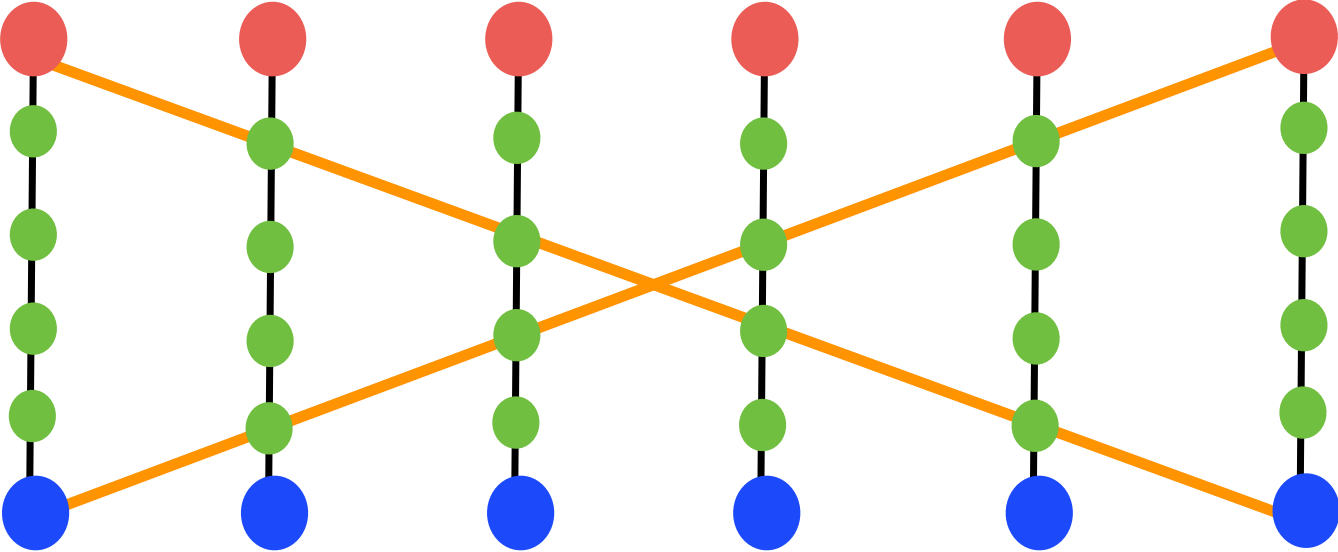


# Spatial Ambiguity of the Midpoint Representation



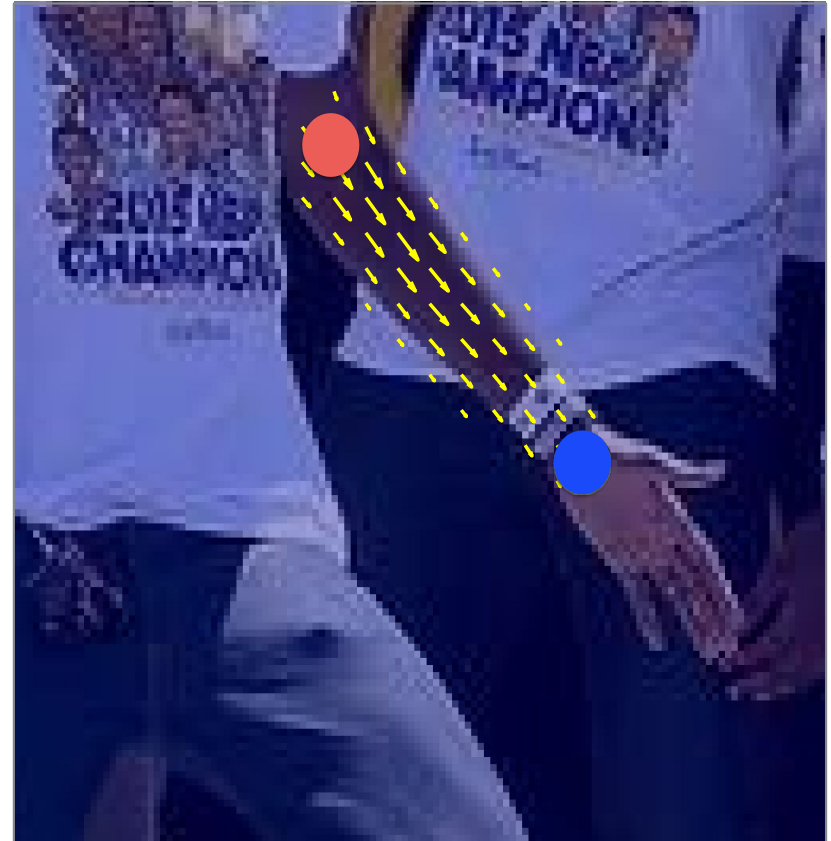
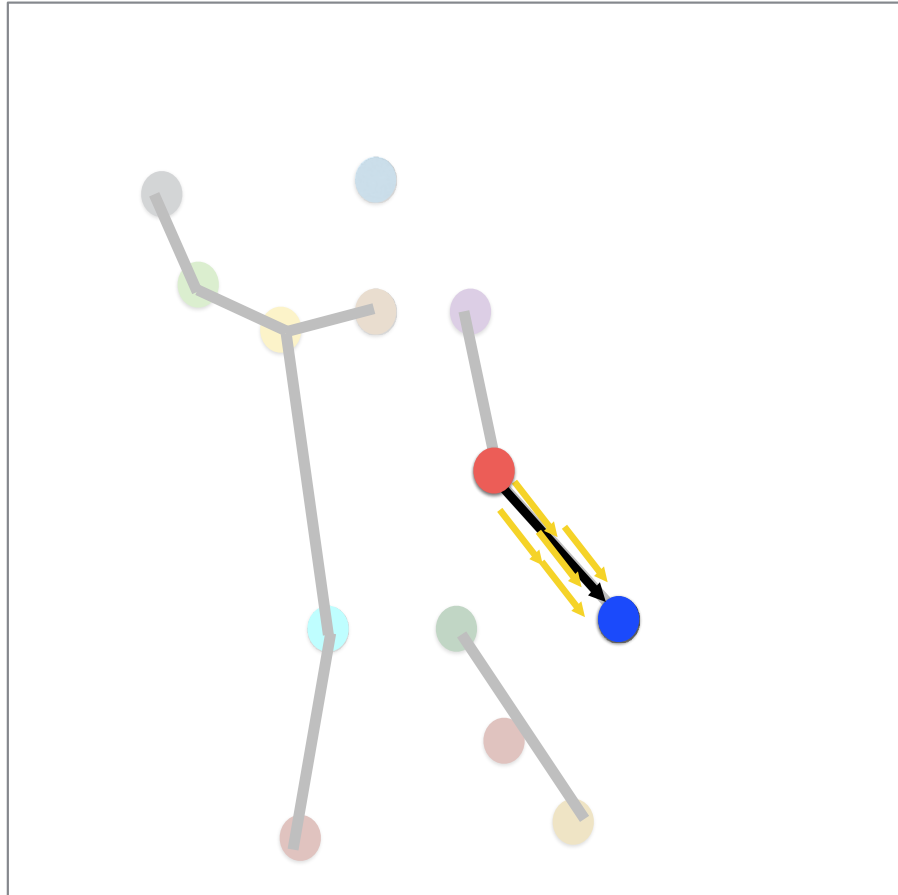
— Correct Connection  
— Wrong Connection

# Increasing Midpoint Number Cannot Solve The Problem

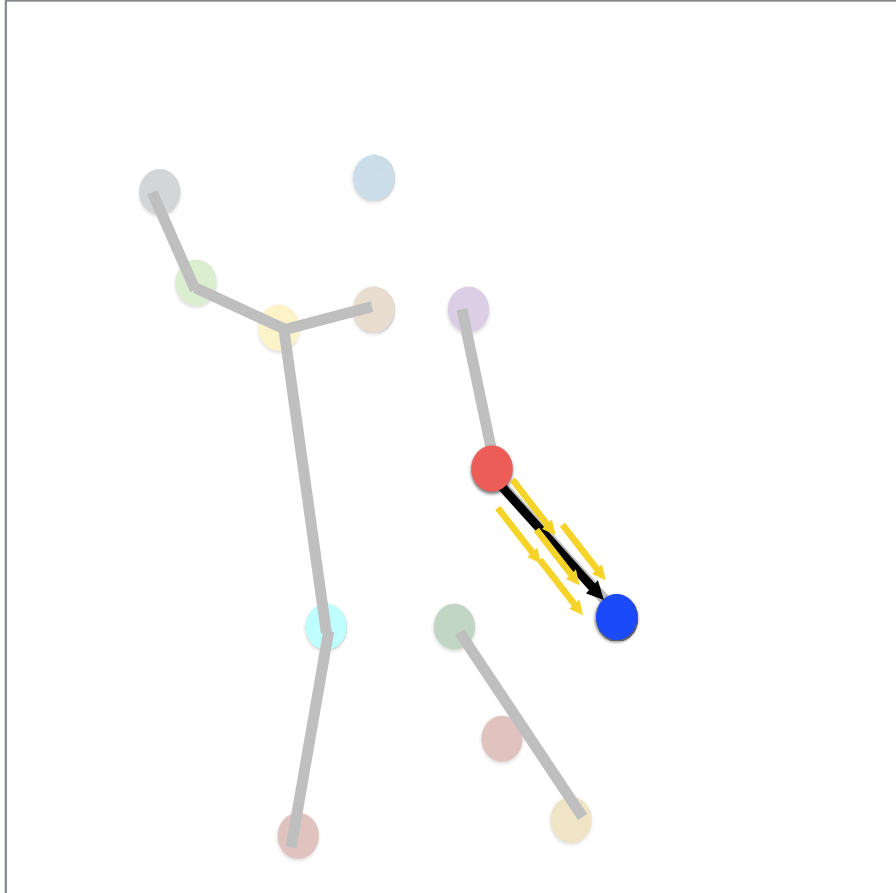


— Correct Connection  
— Wrong Connection

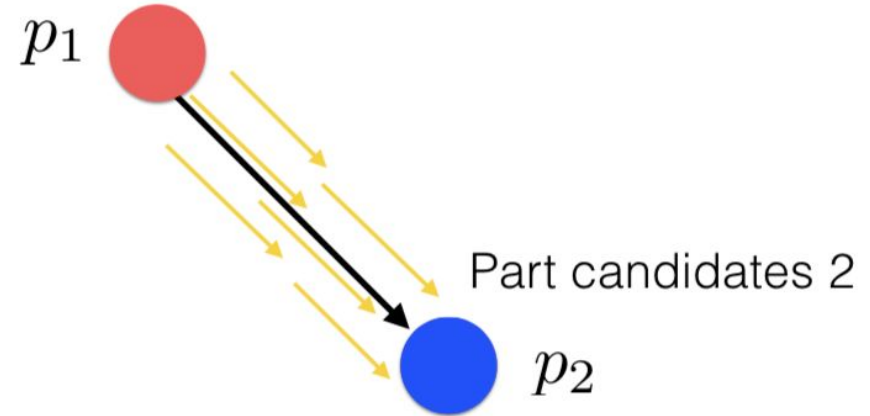
# Part Affinity Fields for Part-to-Part Association



# Part Affinity Fields for Part-to-Part Association



Part candidates 1

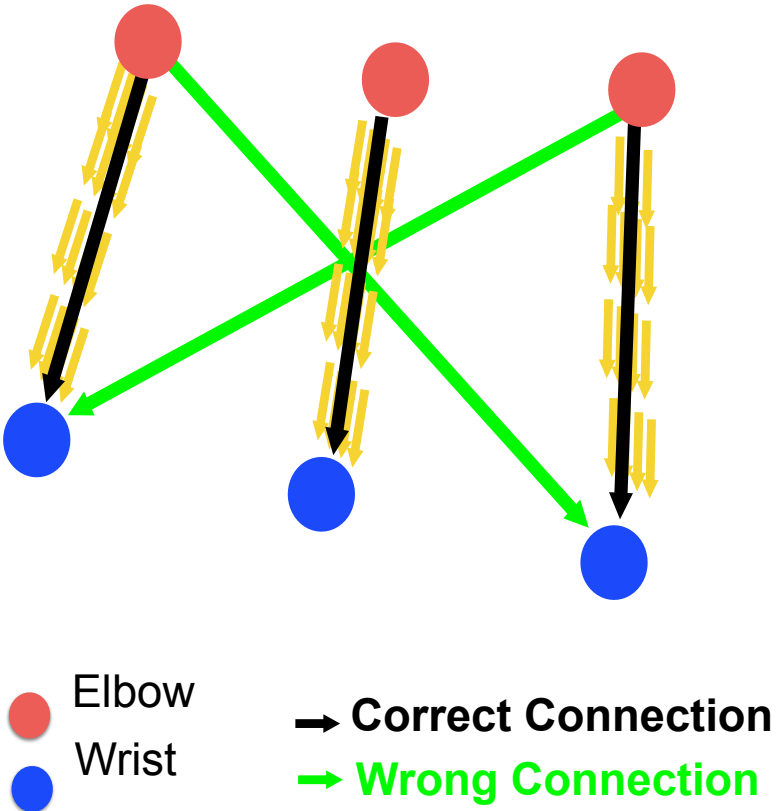


Affinity score between  $p_1$  and  $p_2$   
$$= \text{sum}(\vec{v} \cdot p_1 \vec{p}_2)$$

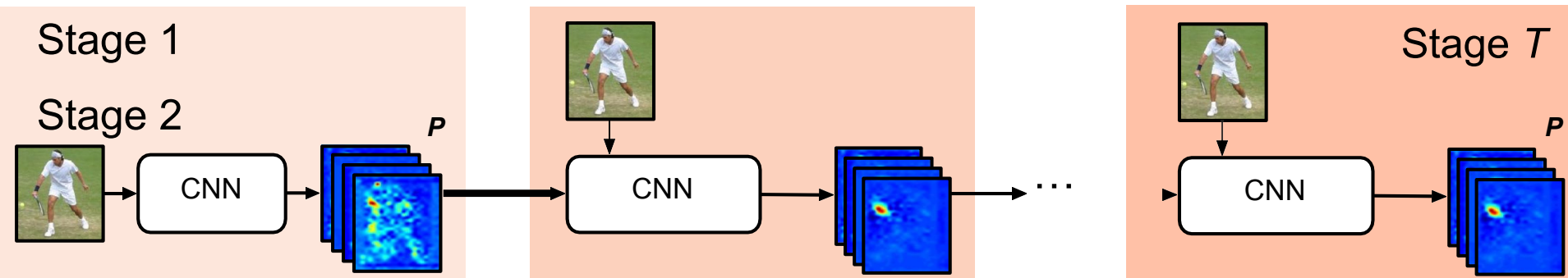
## Part Affinity Score Computation As Line Integral

$$\int_{y_1}^{y_2} \int_{x_1}^{x_2} \begin{bmatrix} \mathbf{PAF}_x(x, y) \\ \mathbf{PAF}_y(x, y) \end{bmatrix} \cdot \begin{bmatrix} \mathbf{v}_x \\ \mathbf{v}_y \end{bmatrix} dx dy$$

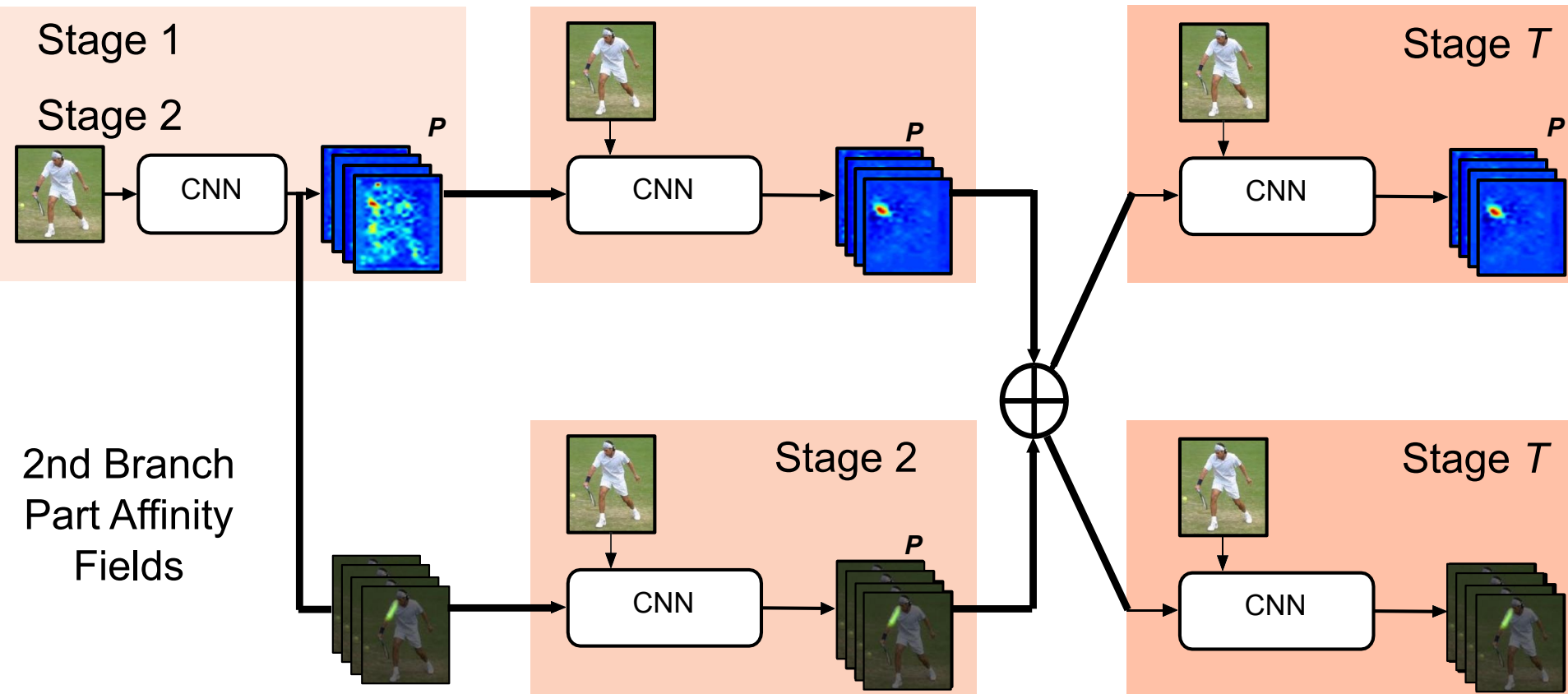
# Part Affinity Fields Avoid Spatial Ambiguity



# Jointly Learning Parts Detection and Parts Association



# Jointly Learning Parts Detection and Parts Association





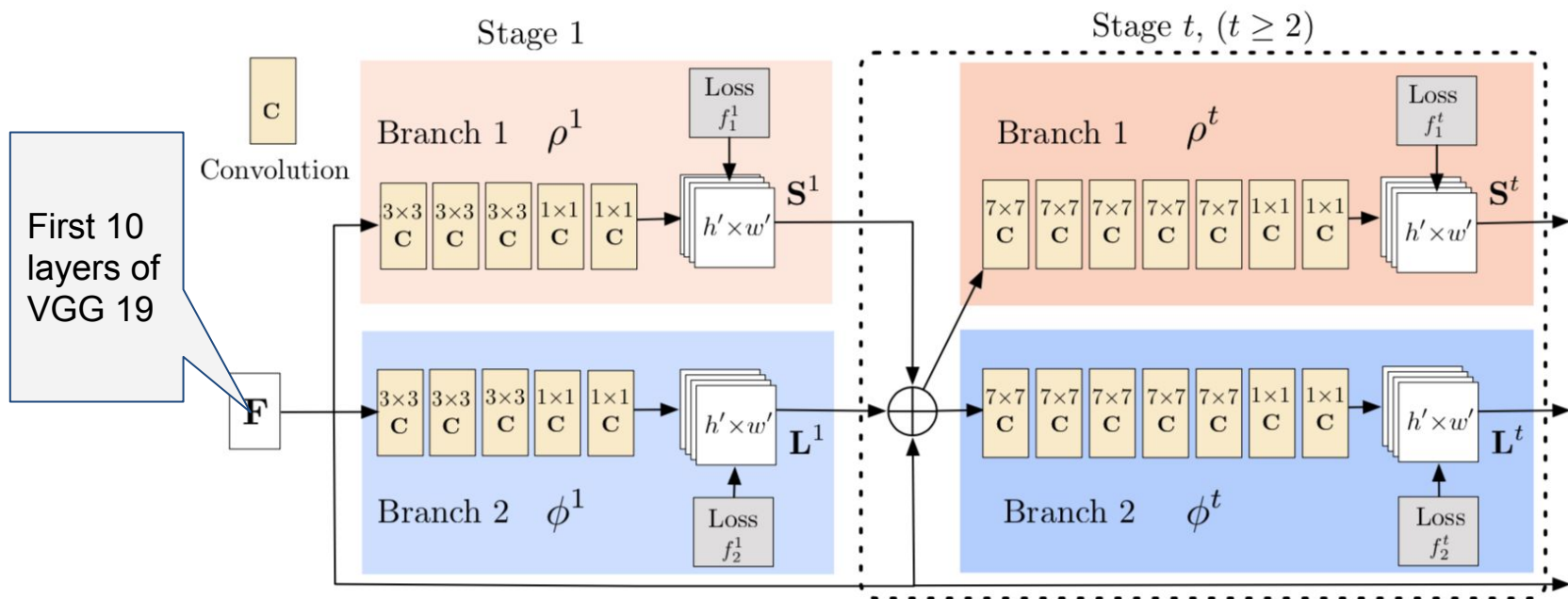
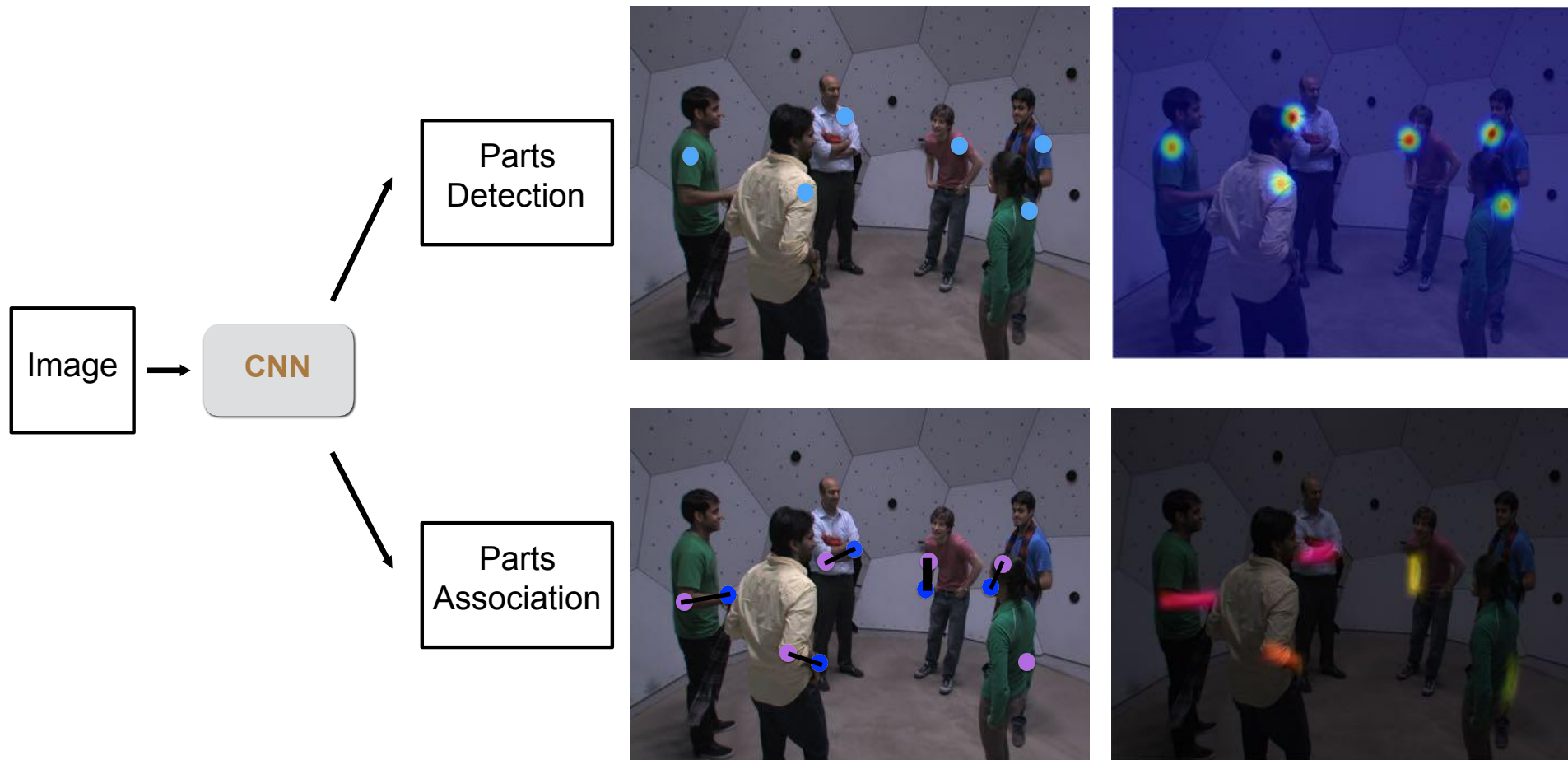


Figure 3. Architecture of the two-branch multi-stage CNN. Each stage in the first branch predicts confidence maps  $S^t$ , and each stage in the second branch predicts PAFs  $L^t$ . After each stage, the predictions from the two branches, along with the image features, are concatenated for next stage.

# 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

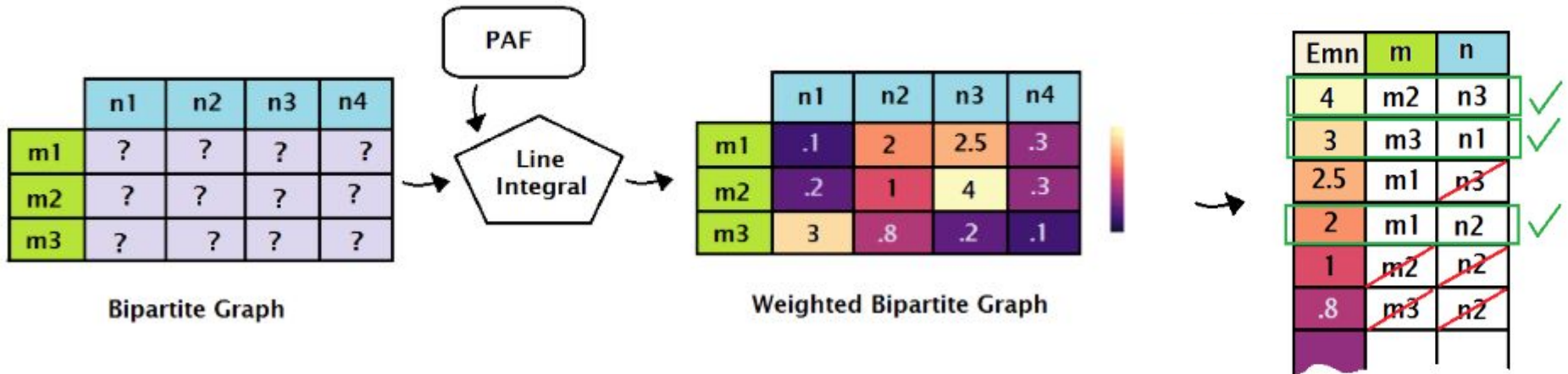
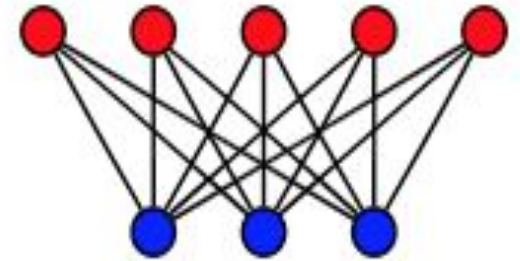




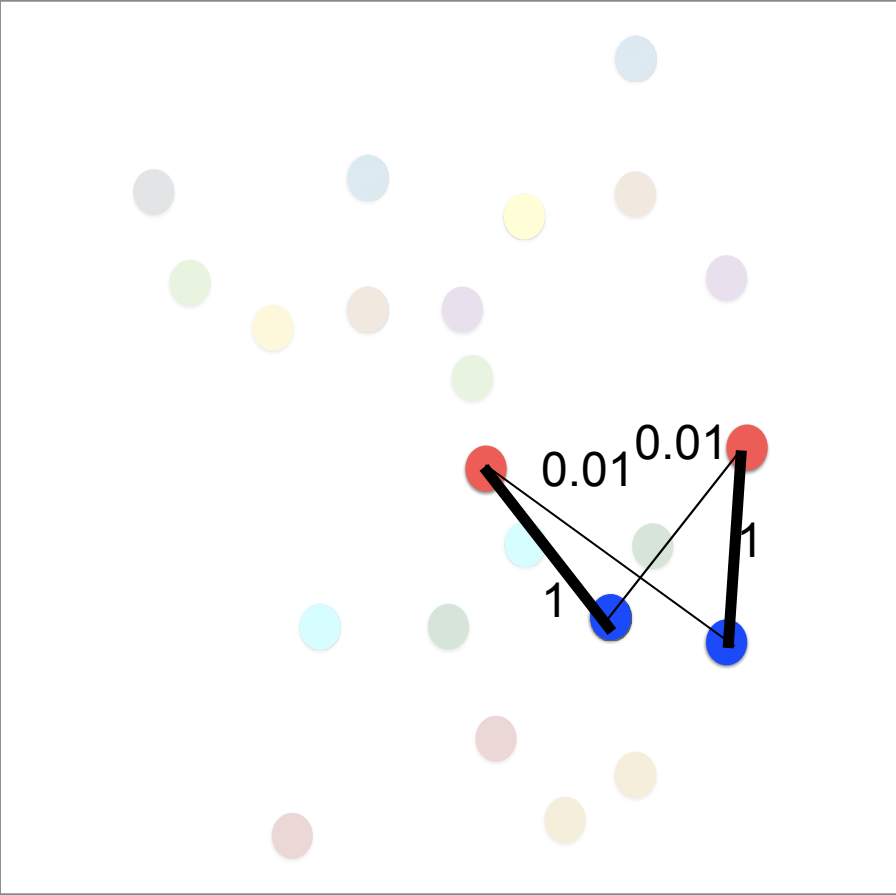


# Part matching overview

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

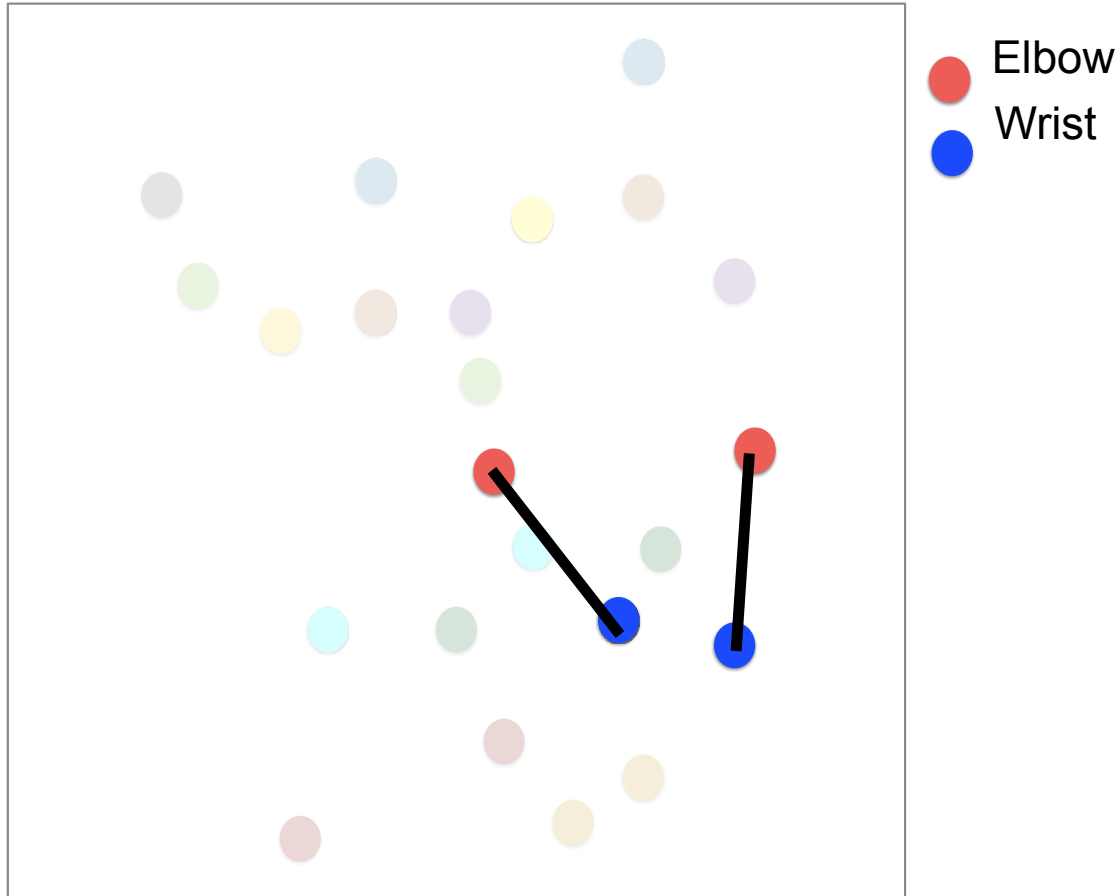


# Greedy Algorithm for Body Parts Association



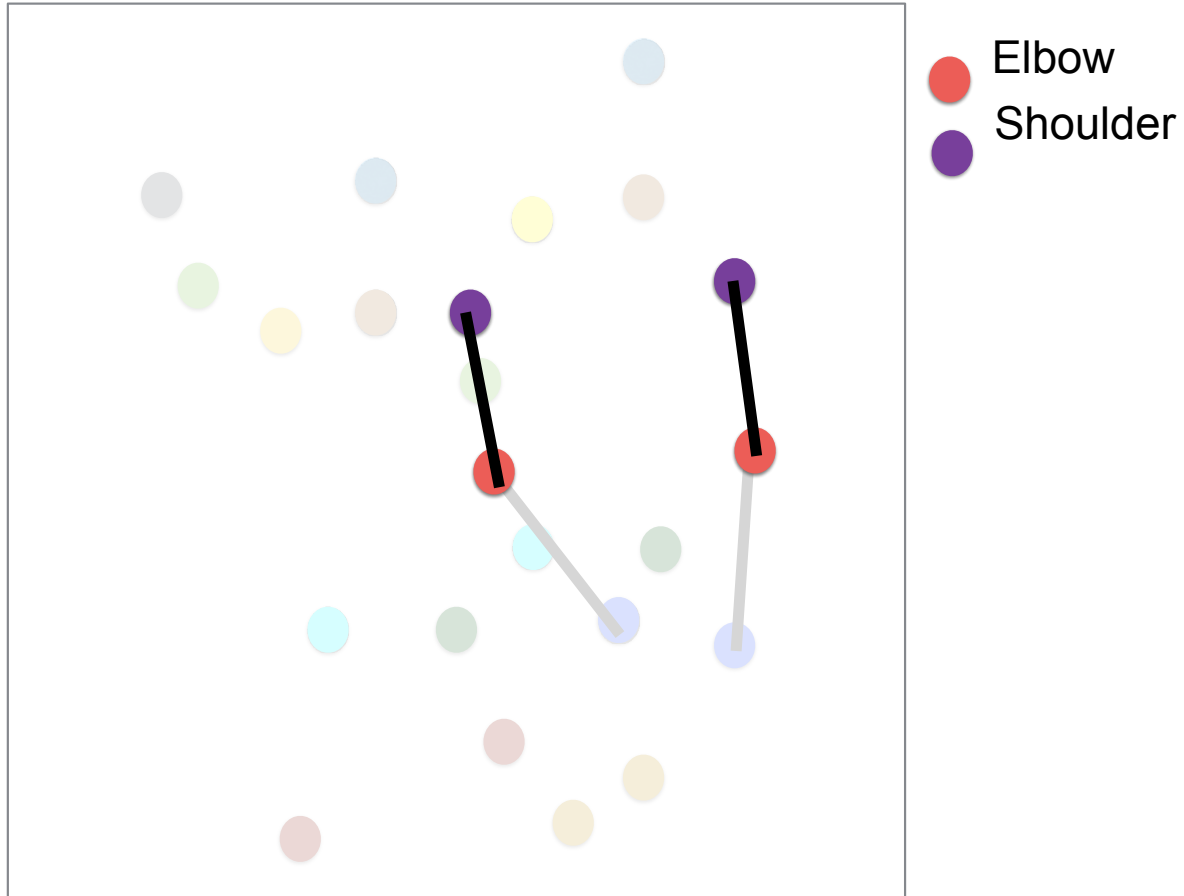
- Elbow
- Wrist

# Greedy Algorithm for Body Parts Association

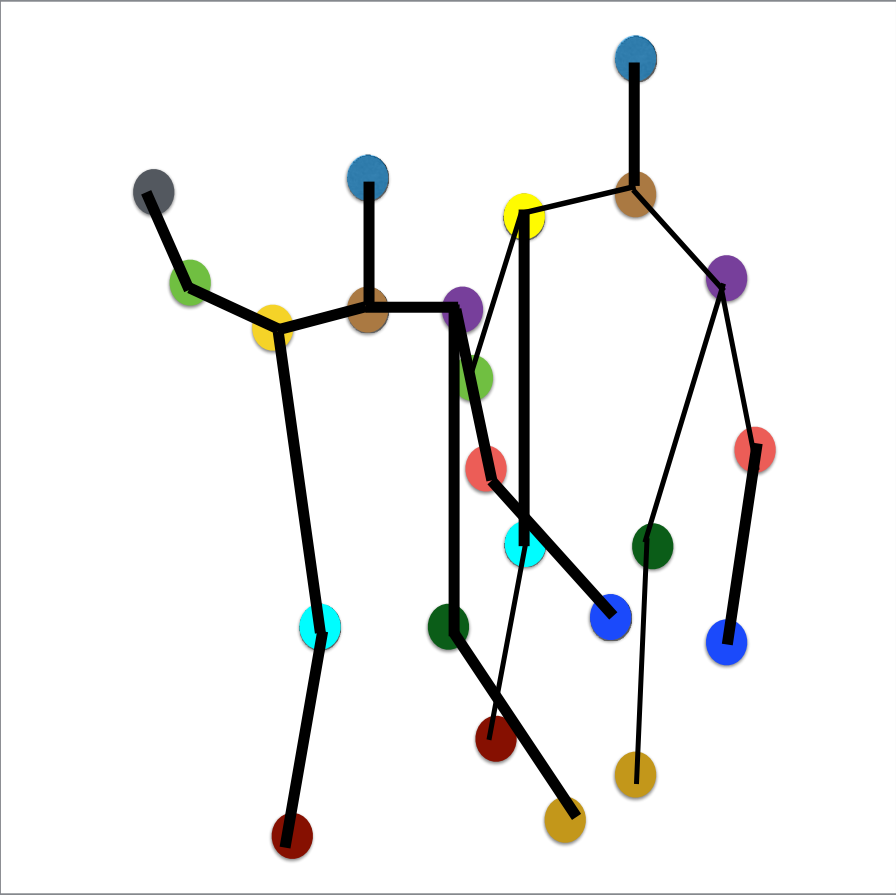




# Greedy Algorithm for Body Parts Association



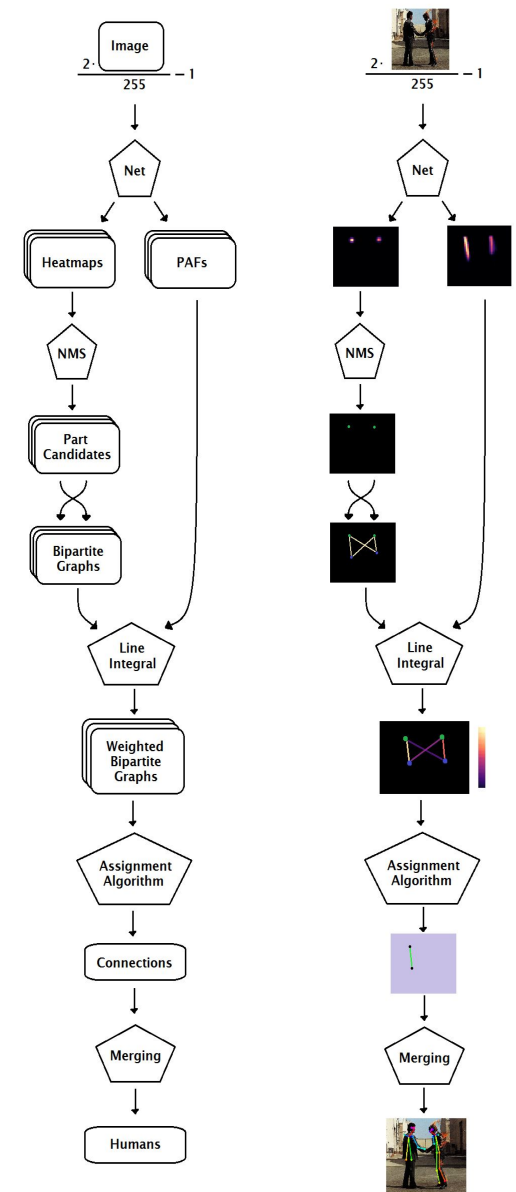
# Greedy Algorithm for Body Parts Association





# 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

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

1 Convolutional Pose Machines [Wei et al. 2016]

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