

# Human Pose Estimation

# Chapter Goals

**After completing this chapter, you should be able to understand :**

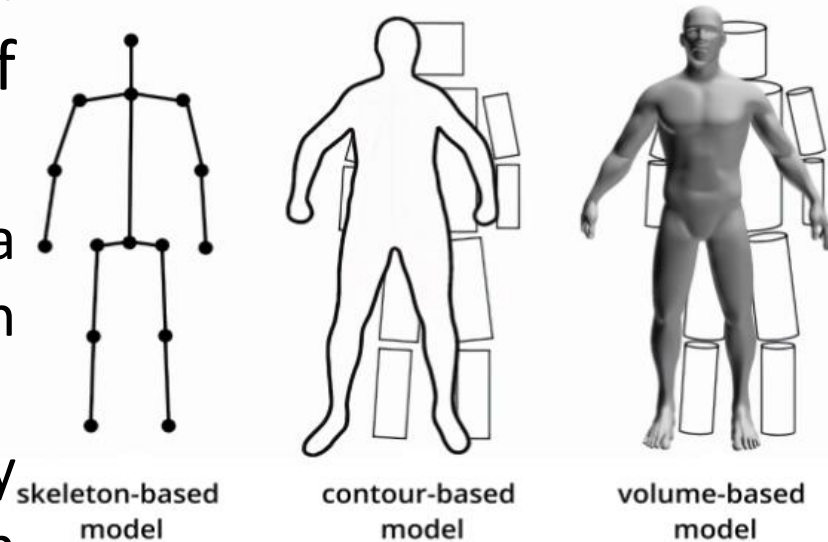
- what is human pose estimation
- Different approaches to solve pose estimation problem.
- OpenPose pipeline for 2D pose estimation
- OpenPose python implementation
- DensePose pipeline for 3D pose estimation
- DensePose python implementation
- MediaPipe pose estimation

# What is Human Pose Estimation

Detection and analysis of human posture. Based on modeling of the human body. There are three types of human body models:

- **Skeleton-based:** set of joints in the skeletal structure of a human body. This model is used both in 2D and 3D human pose estimation techniques because of its flexibility.
- **Contour-based:** the contour and rough width of the body torso and limbs, where body parts are presented with boundaries and rectangles of a person's silhouette.
- **Volume-based:** 3D human body shapes and poses represented by volume-based models with geometric meshes and shapes. Traditionally captured with 3D scans.

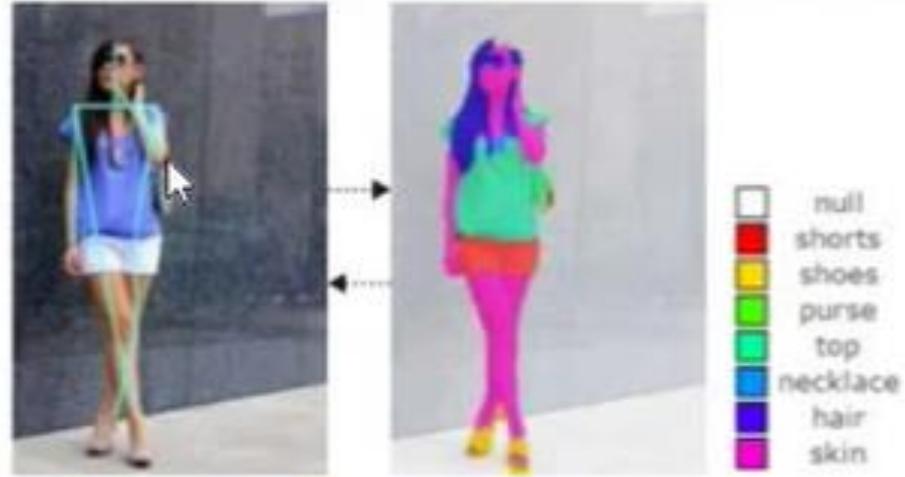
## HUMAN BODY MODELS



# Applications of Human Pose Estimation



Action recognition



Clothing Parsing



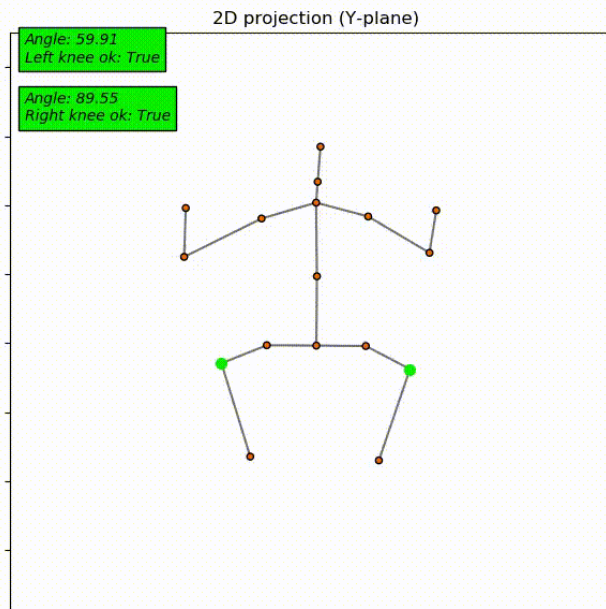
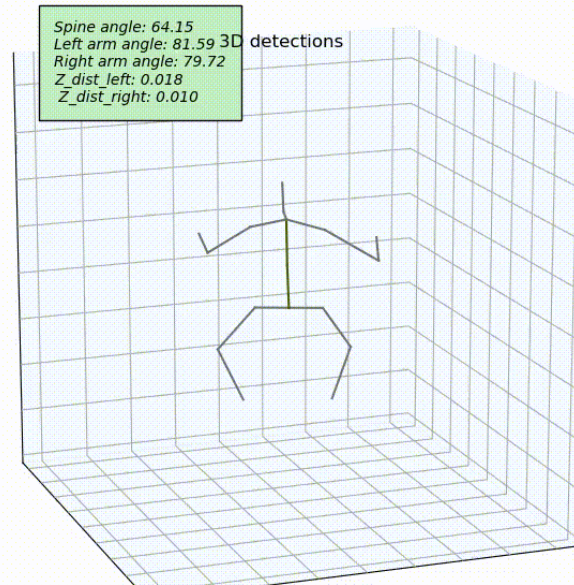
Human tracking



Gaming



# Applications of Human Pose Estimation



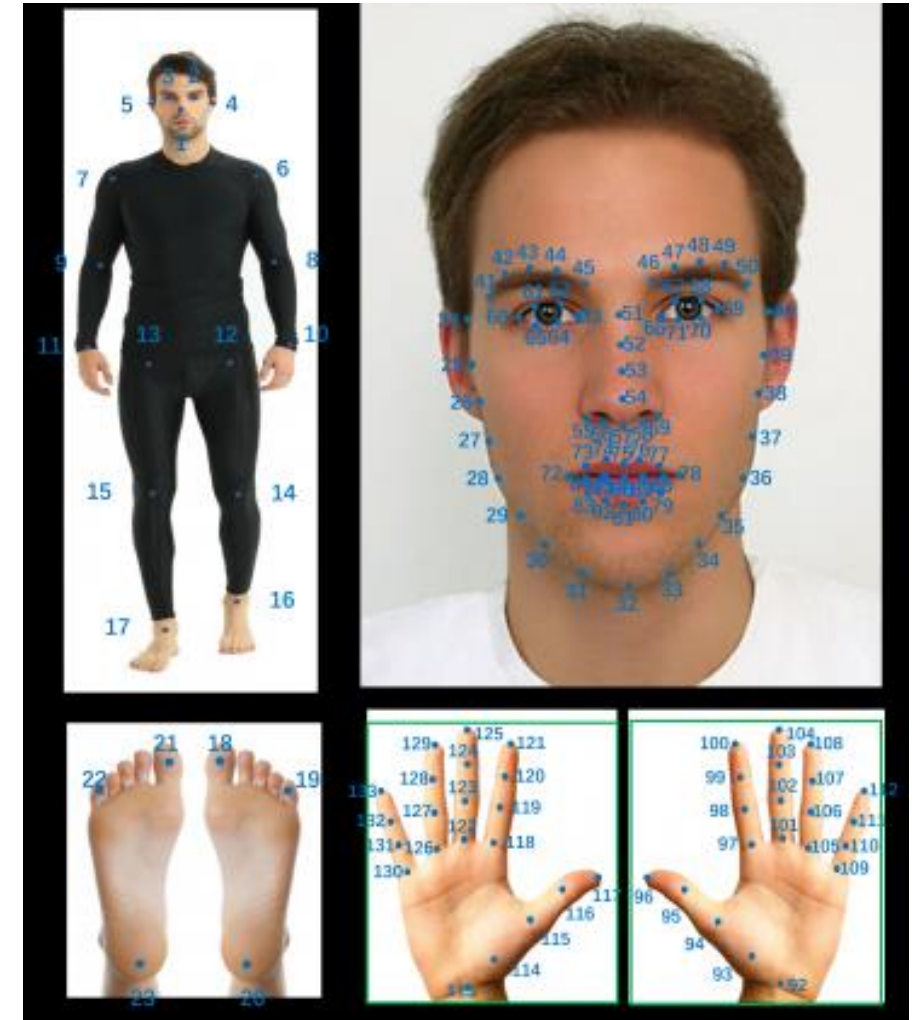
# Challenges for Human Pose Estimation





# Datasets for Human Pose Estimation: COCO

- **COCO WholeBody dataset.** is the first dataset for evaluating whole body posture. COCO-WholeBody is an extension of the COCO 2017 dataset with the same training and validation breakdowns as COCO. There are 4 types of object boundaries available for each person: person box, face box, left-hand box, and right-hand box. In addition, 133 key points: 17 for the body, 6 for the legs, 68 for the face, and 42 for the arms. The dataset is available exclusively for research purposes. **Commercial use is prohibited.**



# Datasets for Human Pose Estimation: MPII

- MPII dataset.** MPII Human Pose dataset is a state of the art benchmark for evaluation of articulated human pose estimation. The dataset includes around **25K images** containing over **40K people** with annotated body joints. The images were systematically collected using an established taxonomy of every day human activities. Overall the dataset covers **410 human activities** and each image is provided with an activity label. Each image was extracted from a YouTube video and provided with preceding and following un-annotated frames.



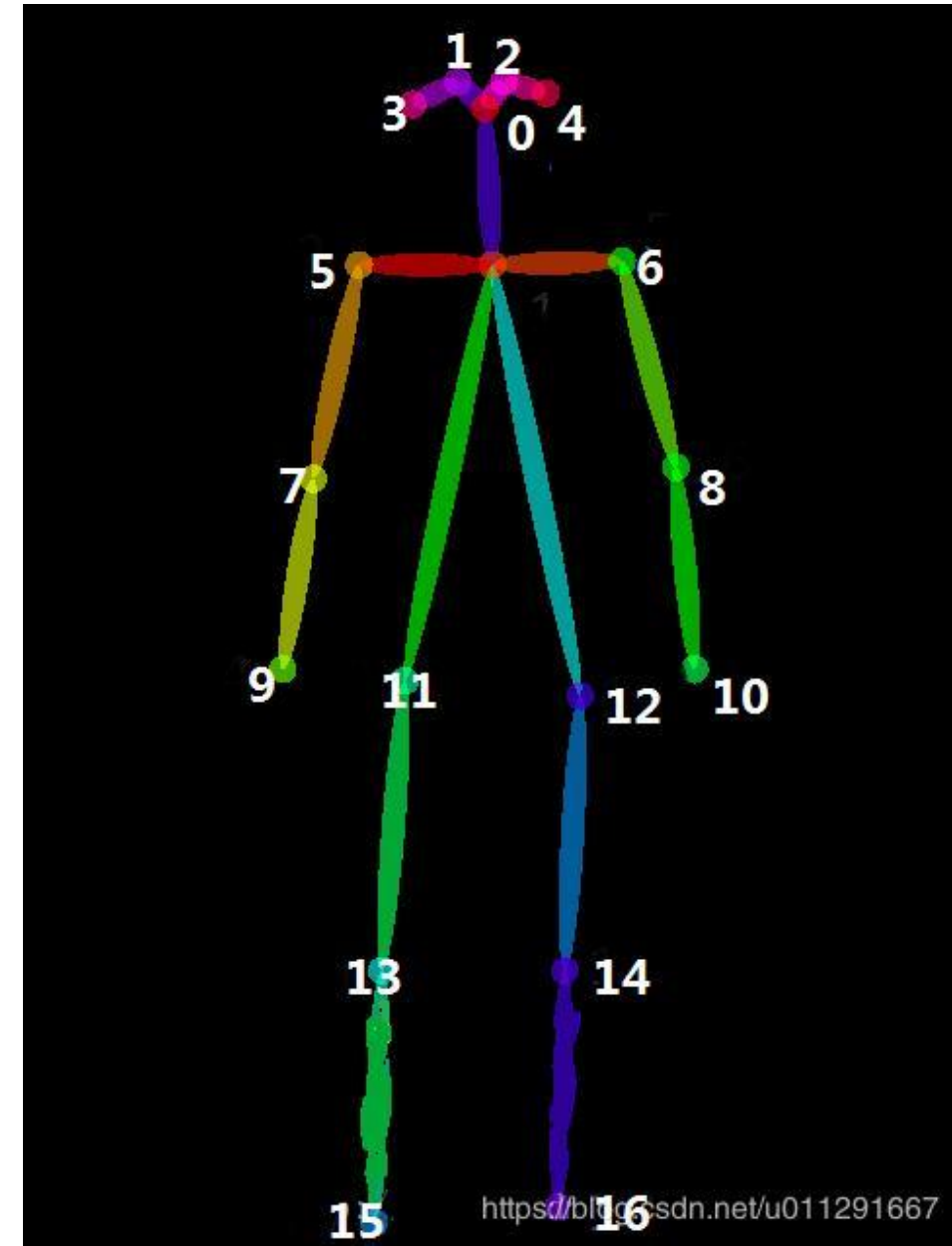


# Datasets for Human Pose Estimation

DataSet	Images	#Kpt	Wild	Body Box	Hand Box	Face Box	Body Kpt	Hand Kpt	Face Kpt	Total
MPII	25K	16	✓	✓		*	✓			40K
COCO	200K	17	✓	✓		*	✓			250K
COCO-WholeBody	200K	133	✓	✓	✓	✓	✓	✓	✓	250K

# Output of Human Pose Estimation

- **COCO Output Format** 0-‘nose’ 1-‘left\_eye’ 2-‘right\_eye’ 3-‘left\_ear’ 4-‘right\_ear’ 5-‘left\_shoulder’ 6-‘right\_shoulder’ 7-‘left\_elbow’ 8-‘right\_elbow’ 9-‘left\_wrist’ 10-‘right\_wrist’ 11-‘left\_hip’ 12-‘right\_hip’ 13-‘left\_knee’ 14-‘right\_knee’ 15-‘left\_ankle’ 16-‘right\_ankle’



# Output of Human Pose Estimation

- **MPII Output Format** 0 - r ankle, 1 - r knee, 2 - r hip, 3 - l hip, 4 - l knee, 5 - l ankle, 6 - **pelvis**, 7 - **thorax**, 8 - upper neck, 9 - **head top**, 10 - r wrist, 11 - r elbow, 12 - r shoulder, 13 - l shoulder, 14 - l elbow, 15 - l wrist



COCO Keypoints



MPII Keypoints



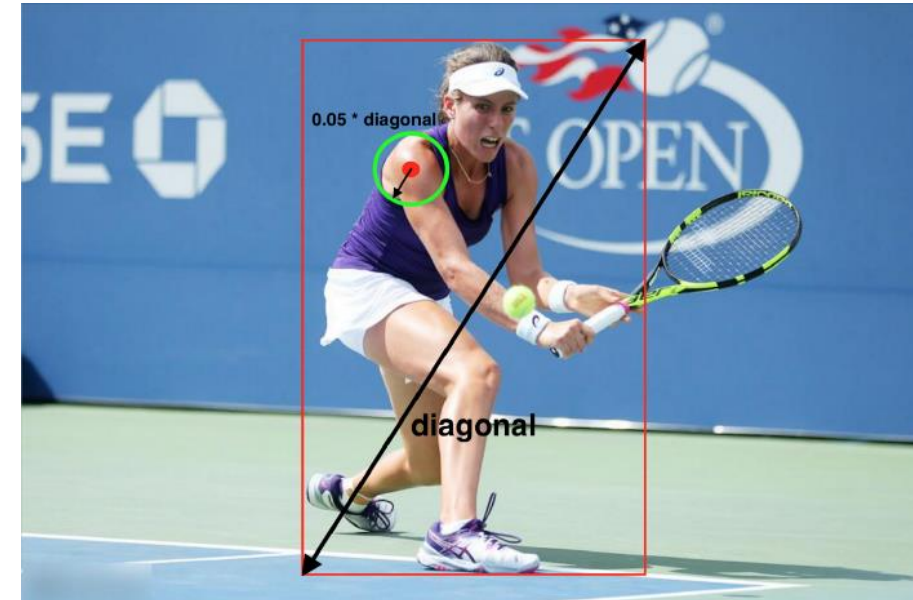
# Metrics for Evaluating Human Pose Estimation: Percentage of Detected Joints

**Percentage of Detected Joints (PDJ):** Detected Joint is considered correct if the distance between the predicted and the true joint is within a certain fraction of the bounding box diagonal.

The use of the PDJ metric implies that the accuracy of the determination of all joints is evaluated using the same error threshold.

$$PDJ = \frac{\sum_{i=1}^n \text{bool}(d_i < 0.05 * \text{diagonal})}{n}$$

- $d_i$  — the euclidian distance between ground truth keypoint and predicted keypoint;
- $\text{bool}(\text{condition})$  — a function that returns 1 if the condition is true, 0 if it is false;
- $n$  — the number of key points on the image.

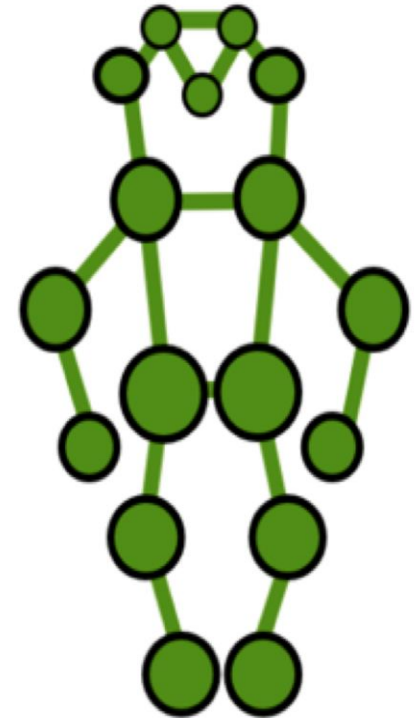


# Metrics for Evaluating Human Pose Estimation: Object Key Point Similarity

**Object Key Point Similarity (OKS):** It is calculated from the distance between predicted points and ground truth points normalized by the scale of the person. Scale and Keypoint constant needed to equalize the importance of each keypoint: neck location more precise than hip location

$$OKS = \exp\left(-\frac{d_i^2}{2s^2k_i^2}\right)$$

Keypoint	$k_i$
hips	0.107
ankles	0.089
knees	0.087
shoulders	0.079
elbows	0.072
wrists	0.062
ears	0.035
nose	0.026
eyes	0.025



- $d_i$  — the euclidian distance between ground truth keypoint and predicted keypoint;
- $s$  — scale: the square root of the object segment area;
- $k$  — per-keypoint constant that controls fall off;

# Human Pose: Top-Down vs Bottom-Up Approaches

- The simple approach is to incorporate a person detector first, followed by estimating the parts and then calculating the pose for each person. This method is known as the **top-down** approach.
- Another approach is to detect all parts in the image (parts of every person), followed by associating/grouping parts belonging to distinct persons. This method is known as the **bottom-up** approach.





# OpenPose



OpenPose

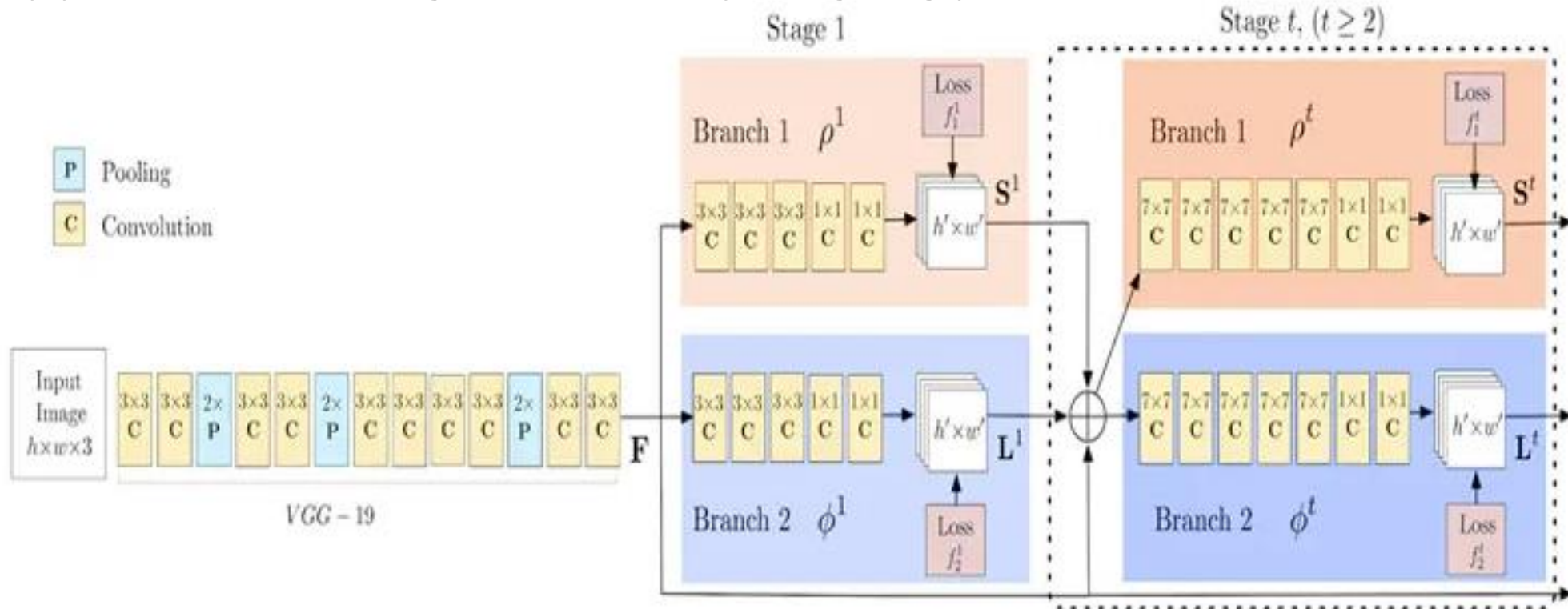
# OpenPose

- [OpenPose](#) is one of the most popular **bottom-up** approaches for multi-person human pose estimation, partly because of their well documented [GitHub](#) implementation.
- As with many bottom-up approaches, OpenPose first detects parts (keypoints) belonging to every person in the image, followed by assigning parts to distinct individuals.



# OpenPose Architecture

- [OpenPose](#) is one of the most popular **bottom-up** approaches for multi-person human pose estimation, partly because of their well documented [GitHub](#) implementation.
- As with many bottom-up approaches, OpenPose first detects parts (keypoints) belonging to every person in the image, followed by assigning parts to distinct individuals.





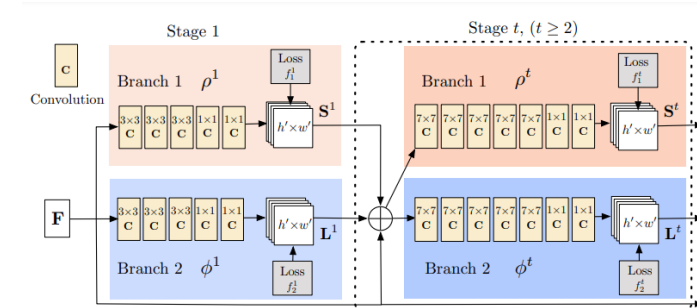
# Part Confidence Maps



The heat map over the Image denotes the probability of a body part in that map. For example, one on the left is the map for the left elbow and one on the right is for left shoulder. | Source:

<https://arxiv.org/pdf/1812.08008.pdf>

The model takes input and produces confidence maps as an intermediate output, one for each part of the body. Confidence maps denotes high probabilities where a body part might be in the Input Image.



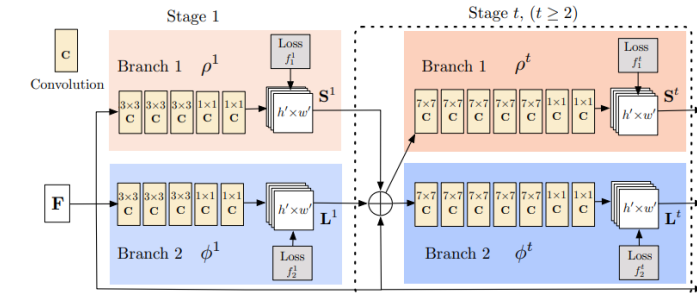
# PAFs: Part Affinity Fields



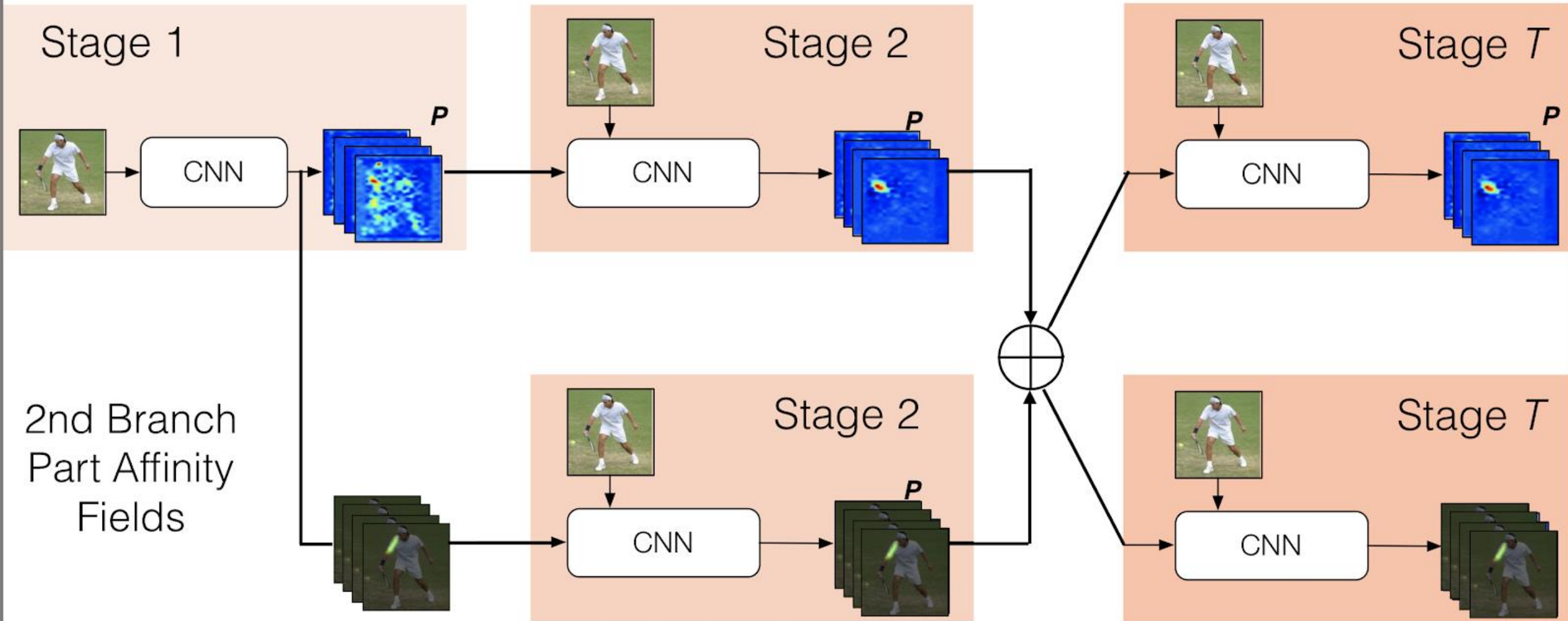
The PAF here above in the image is the association of the limb from left shoulder to the left elbow. | Source:

<https://arxiv.org/pdf/1812.08008.pdf>

PAFs are a set of 2D vector fields that encode the location and orientation of limbs over the image domain. PAFs are utilized in the network for part associations. We have a PAF for each limb or part pairs (in case of non living object). It encodes the direction that points from one part of the limb to the other.



# Network Architecture

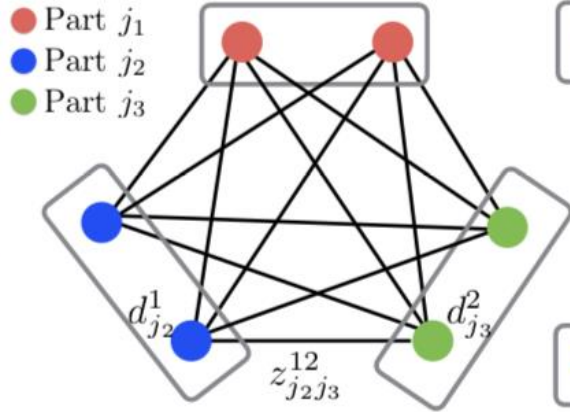




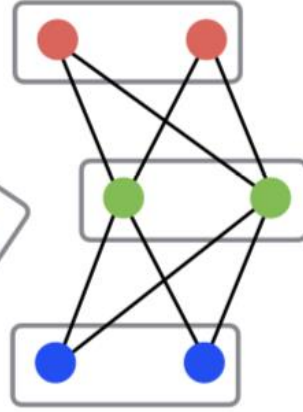
# Network Architecture



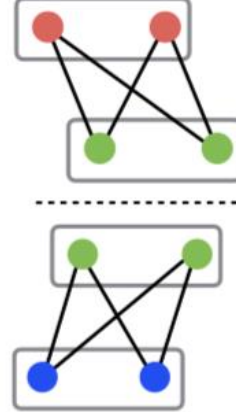
(a)



(b)



(c)



(d)

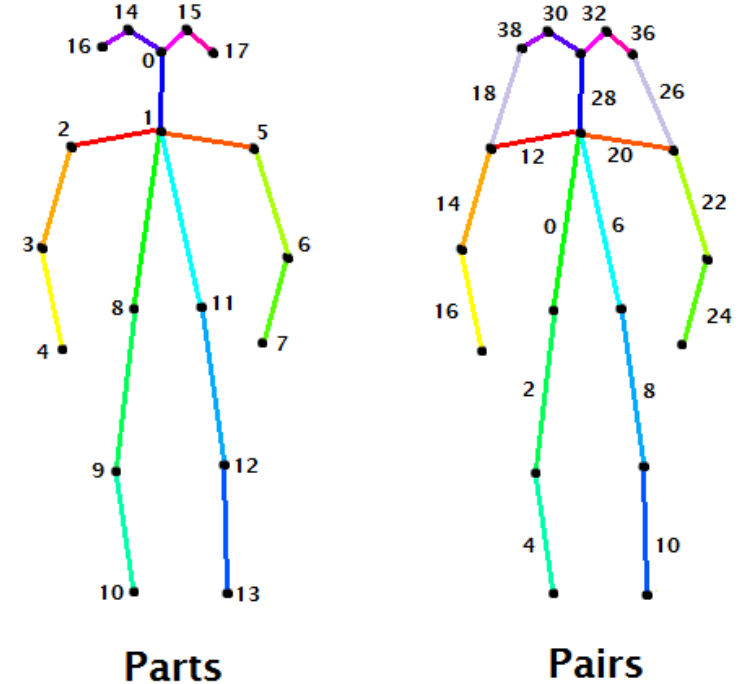
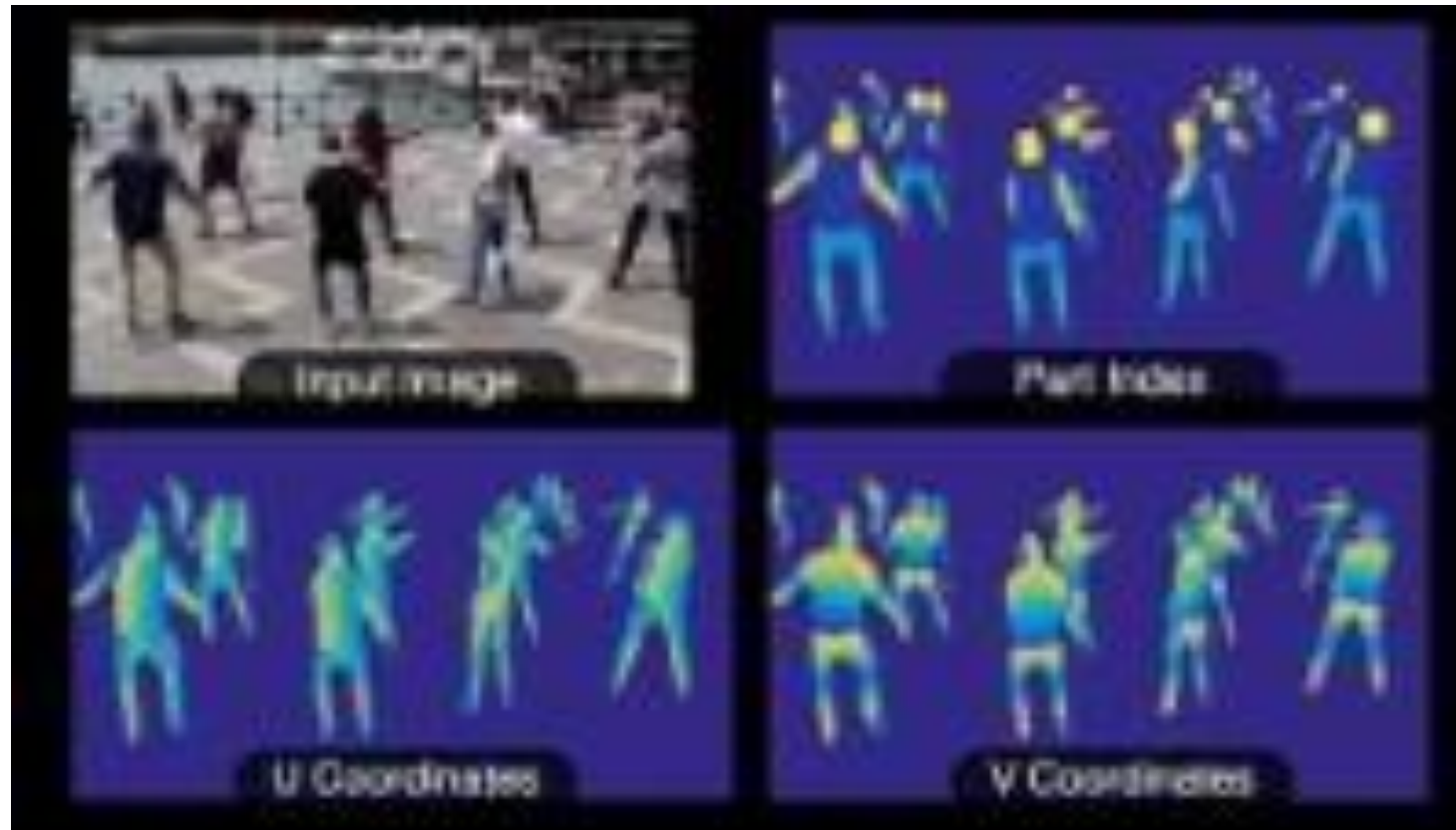


Figure 6. Graph matching. (a) Original image with part detections (b)  $K$ -partite graph (c) Tree structure (d) A set of bipartite graphs

this problem can be viewed as a **k-partite graph matching** problem, where nodes of the graph are body part detections, and edges are all possible connections between them (possible limbs).

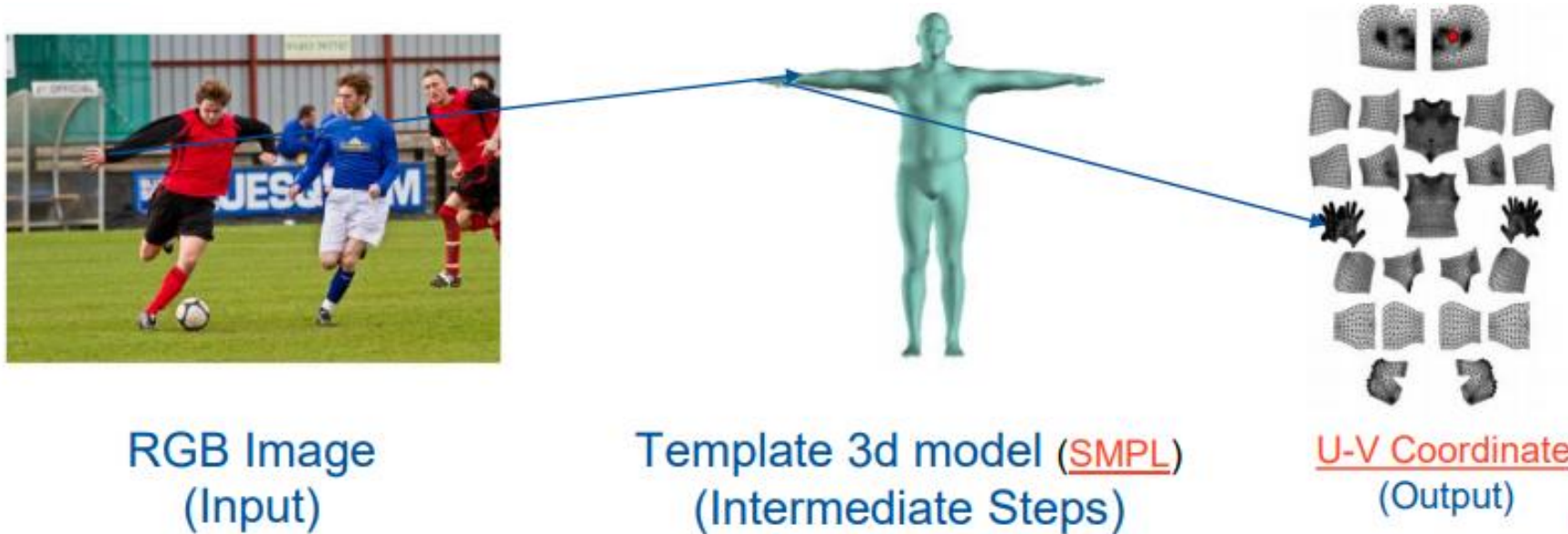
**k-partite graph** is a graph whose vertices are or can be partitioned into  $k$  different [independent sets](#).

# DensePose



# DensePose Estimation Motivation

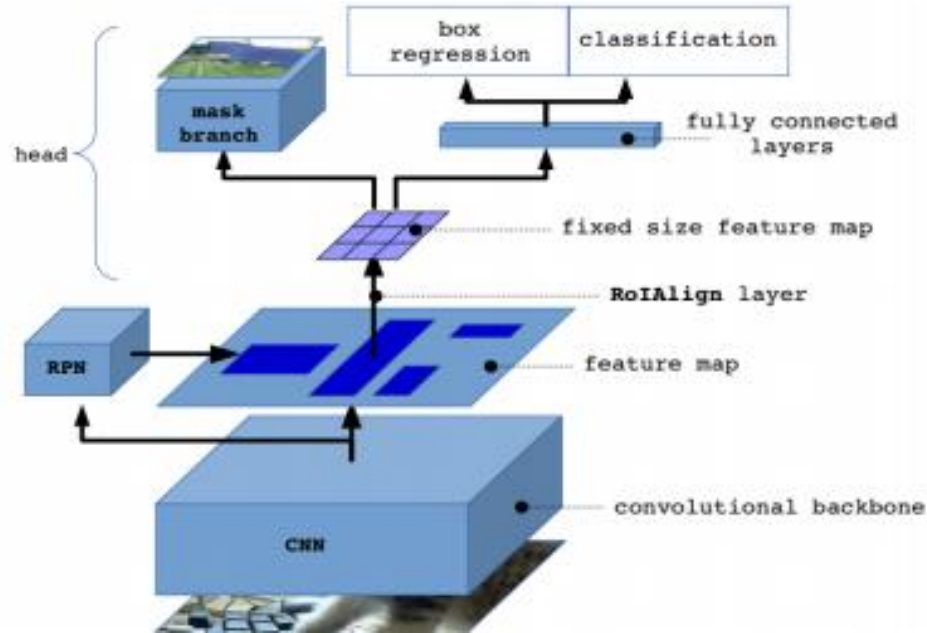
DensePose aims at pushing further the envelope of human understanding in images by establishing dense correspondences from a 2D image to a 3D, surface-based representation of the human body



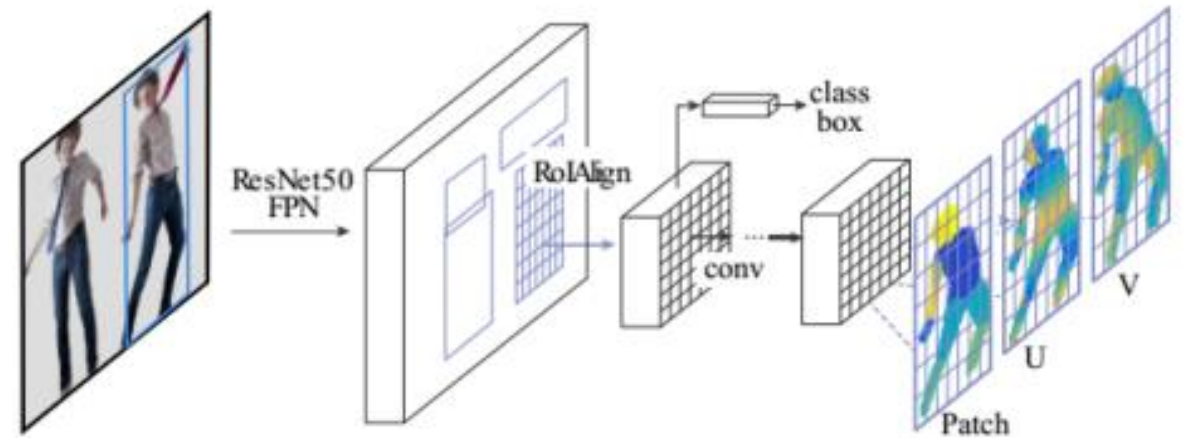
**UV mapping** is the [3D modeling](#) process of projecting a 2D image to a 3D model's surface for [texture mapping](#). The letters "U" and "V" denote the axis of the 2D texture

# Proposed Method

## Basic Model



Mask RCNN



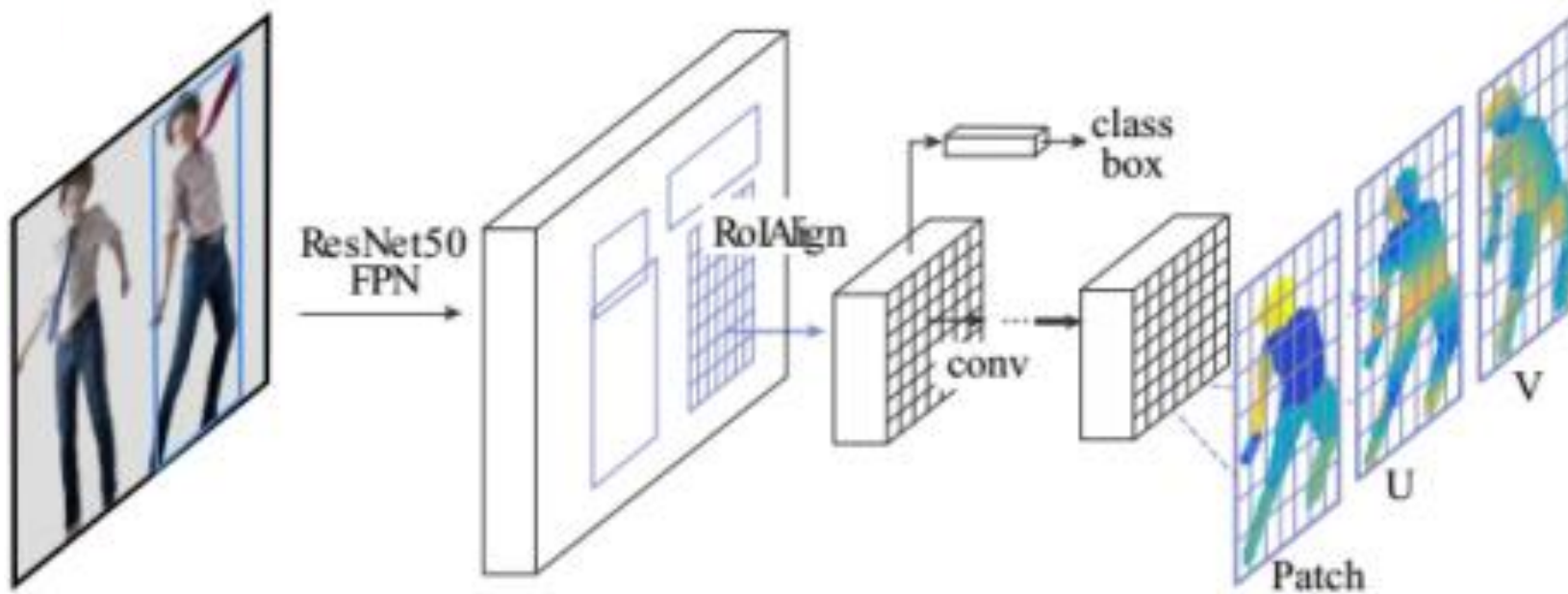
Region-based Dense Pose Regression

Replace the mask head with dense pose head. Such architectures decompose the complexity of the task into controllable modules.



# Proposed Method

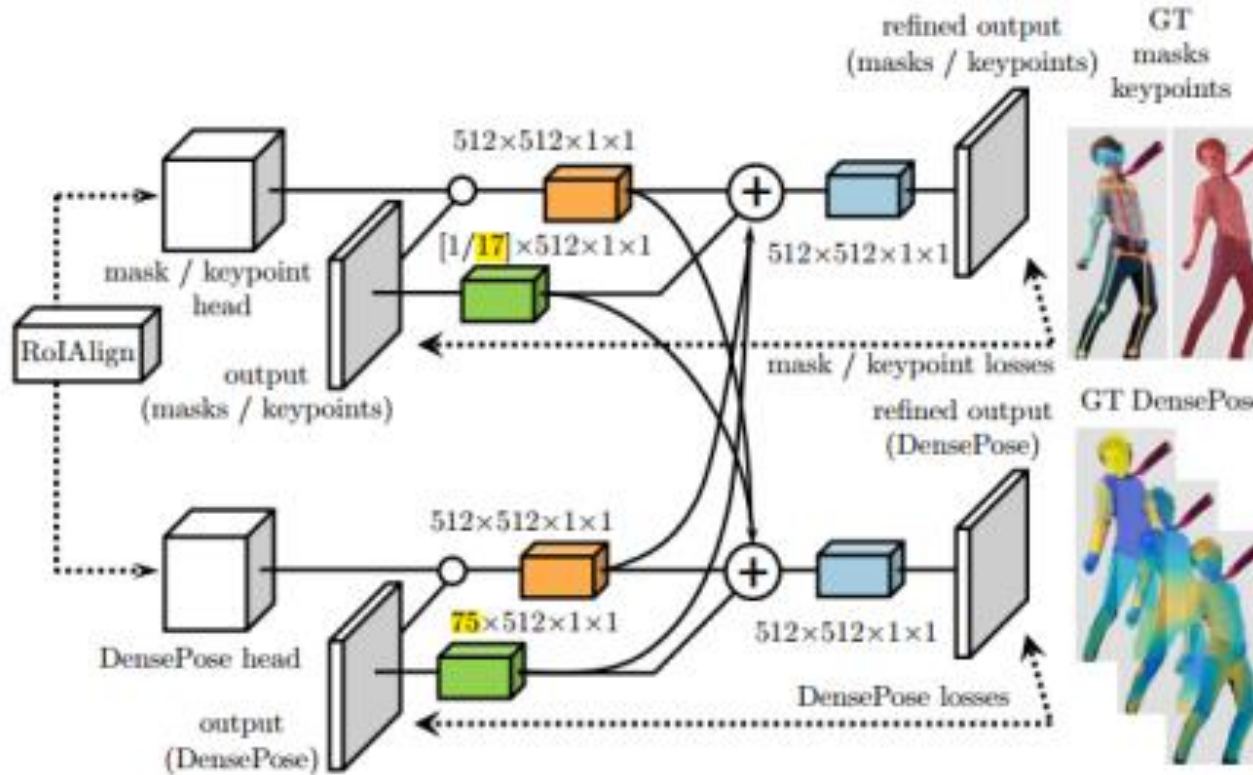
## Basic Model



- Patch: a classification that provide the part assignment. ( $25 \times H \times W$ )
- They classify a pixel as belonging to either background, or one among several body parts which provide a coarse estimate of surface coordinates.
- (U, V): a regression head that provide part coordinate predictions in each part. ( $25 \times H \times W \times 2$ ) Indicates the exact coordinates of the pixel within the part.

# Proposed Method

## Modification:



## Multi-task cascaded architectures:

- Inspired by the success of recent pose estimation models based on iterative refinement , so they provide the output of previous stage as the input of the next stage.
- exploit information from related tasks, such as keypoint estimation and instance segmentation, which have successfully been addressed by the Mask-RCNN architecture.

# Results















We observe that DensePose successfully estimates body pose regardless of skirts or dresses, while handling a large variability of scales, poses, and occlusions.

# MediaPipe for Pose Estimation



# Google's MediaPipe

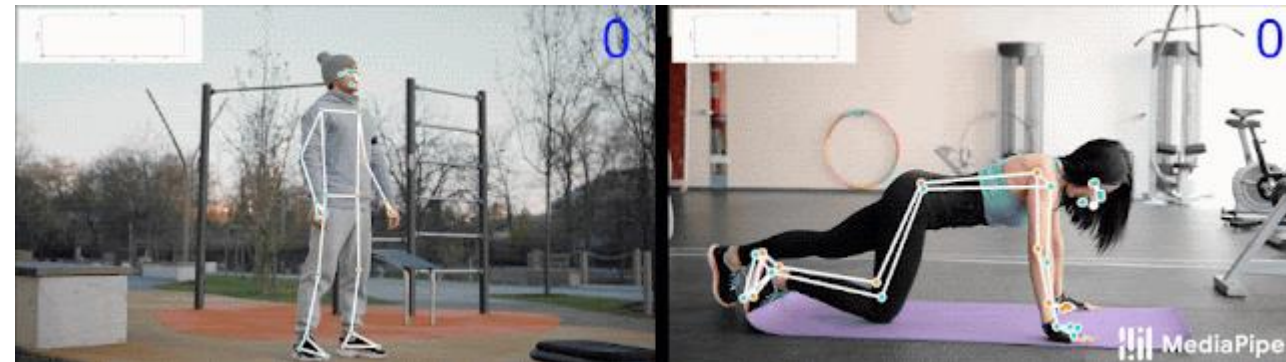
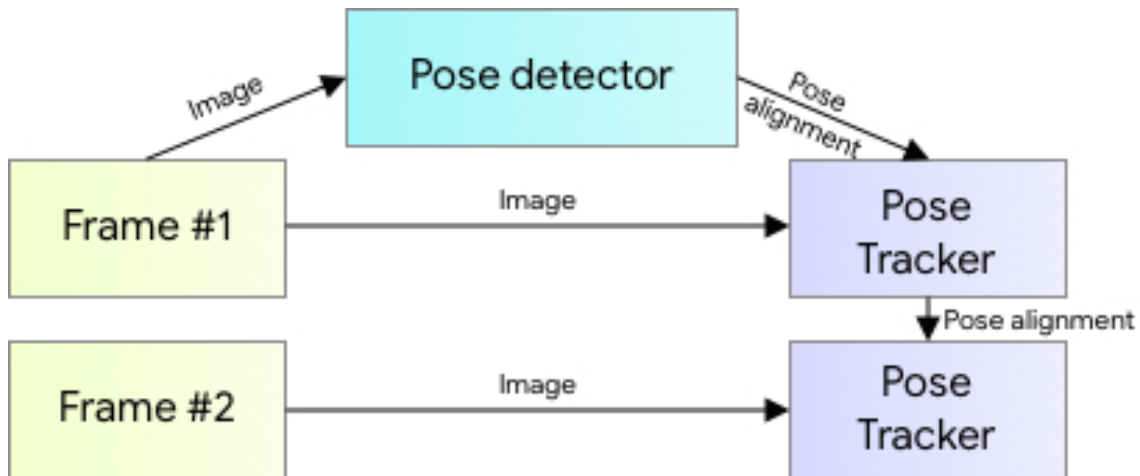
MediaPipe is a cross-platform, open-source library developed by Google providing cutting-edge ML solutions. The library is designed to work across platforms like iOS, Linux, Windows, and Android on high to low-powered hardware like Raspberry Pi. Some of the solutions provided by the library are Face Detection, Iris Detection, Human Pose Estimation, and Instant motion tracking. Refer to the [MediaPipe page](#) for solutions supported in different languages and platforms.

Face Detection	Face Mesh	Iris	Hands	Pose	Holistic
					
Hair Segmentation	Object Detection	Box Tracking	Instant Motion Tracking	Objectron	KNIFT
					

# Google's MediaPipe

MediaPipe Pose is a ML solution for high-fidelity body pose tracking, inferring 33 landmarks on the whole body from RGB video frames utilizing [BlazePose](#)

For pose estimation, we utilize our [proven](#) two-step [detector-tracker ML pipeline](#). Using a detector, this pipeline first locates the pose region-of-interest (ROI) within the frame. The tracker subsequently predicts all 33 pose keypoints from this ROI. Note that for video use cases, the detector is run only on the first frame. For subsequent frames we derive the ROI from the previous frame's pose keypoints as seen below.



## Similar Tasks:

- Hand Pose Estimation
- Facial Landmark Detection

# Similar Tasks: Hand Pose Estimation

- Hand Keypoint detection is the process of finding the joints on the fingers as well as the finger-tips in a given image

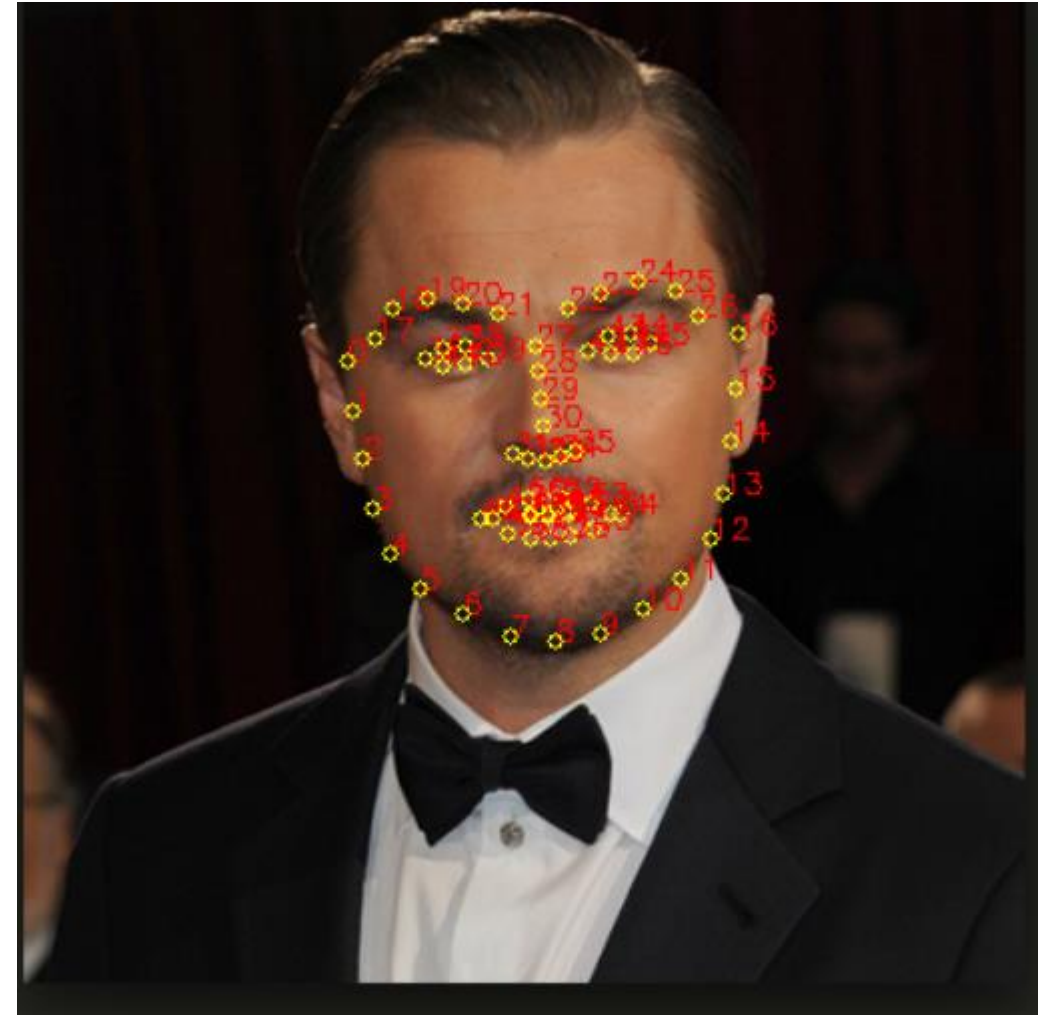




# Similar Tasks: Facial Landmark Detection

Facial landmarks can help predict:

- emotion detection
- distress detection
- driver is falling asleep. We are going to use a pre-trained model for detection of landmarks around the face .
- Dlib library is commonly used: offers different algorithms for face detection including CNN based face detector.



# Chapter Summary

- We saw what is human pose estimation
- Different approaches to solve pose estimation problem.
- OpenPose pipeline
- OpenPose python implementation
- DensePose pipeline
- DensePose python implementation
- Google MediaPipe