# HUMAN POSE DETECTION

Mohammad Nasser

ORANGE LABS  Machine Learning Internship Task

# 1 Introduction

Pose estimation is one of the long-standing problems of computer vision. It has interested researchers over the last several decades because not only is pose estimation an important class of problems itself, but it also serves as a preprocessing step for many even more interesting problems. If we know the pose of a human, we can further train machine learning models to automatically infer relative positions of the limbs and generate a pose model that can be used to perform smart surveillance with abnormal behavior detection, analyze pathologies in medical practices, control 3D model motion in realistic animations, and a lot more. And just for fun remember the scene where Tony stark wears the Iron Man suit using gestures? If such a suit is ever built, it would require human pose estimation!

# 2 Different Approaches

When you try to detect multiple people pose estimation it become even harder, humans occlude and interact with other humans. In this case, it is common practice to use a so-called *top-down approach*: apply a separately trained human detector (based on object detection techniques) find each person, and then run pose estimation on every detection. It sounds reasonable but actually the difficulties are almost insurmountable: if the detector fails to detect a person, or if limbs from several people appear in a single bounding box (which is almost guaranteed to happen in case of close interactions or crowded scenes), the whole algorithm will fail. Moreover, the computation time needed for this approach grows linearly with the number of people on the image, and that can be a big problem for real-time analysis of groups of people.

In contrast, *bottom-up* approaches recognize human poses from pixel-level image evidence directly. They can solve both problems above: when you have information from the entire picture you can distinguish between the people, and you can also decouple the runtime from the number of people on the frame… at least theoretically. However, you are still supposed to be able to analyze a crowded scene with lots of people, assigning body parts to different people, and even this task by itself could be NP-hard in the worst case.

And there are some deep learning methods using these approaches like OpenPose, DeepCut, RMPE (AlphaPose) and Mask RCNN.

And I decided to use OpenPose model for this task as it's an efficient method for multi-person pose estimation with state-of-the-art accuracy on multiple public benchmarks.

# 3 This Model

It is a bottom-up approach, and it uses the so-called Part Affinity Fields (PAFs) together with estimation of body-part confidence maps (keypoints of the body). A PAF is a set of 2D vector fields that encode the location and orientation of the limbs.

Suppose we have already detected all body parts (hands, elbows, feet, ankles etc.); how do we now generate poses from them? First, we must find out how to connect two points to form a limb. For each body part, there are several candidates to form a limb: there are multiple people on the image, and there also can be lots of false positives. We need some confidence measure for the association between each body part detection. That's when 'Part Affinity Fields' or PAF come in handy. It's a feature representation that contains information about location as well as orientation across the region of support of the limb.

*Figure 1. Affinity field visualization for right forehand. The color encodes limb's orientation*

If a point lies on the limb then its value in the PAF is a unit vector pointing from starting joint point to ending joint point of this limb; the value is zero if it is outside the limb. Thus, **PAF is a vector field that contains information about one specific limb for all the people on the image, and the entire set of PAFs encodes all the limbs for every person.**

## 3.1 Overall pipeline

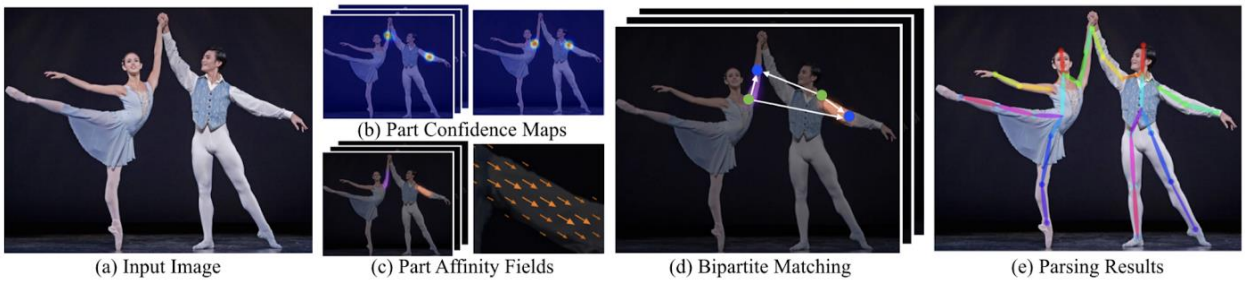



*Figure 2. Overall pipeline*

Figure 2 above illustrates all the steps from an input image to anatomical keypoints as an output. First, a feedforward neural network predicts a set of body part locations on the image in the

form of a confidence map and a set of PAFs that encode the degree of association between these body parts. Thus, the algorithm gets all information necessary for further matching of limbs and people. Next, confidence maps and affinity fields are parsed together to output the final positions of limbs for all people on the picture.

## 3.2 Architecture

The method uses a special feedforward network as a feature extractor. The model takes as input a color image of size h × w × 3 and produces, as output, the 2D locations of keypoints for each person in the image.

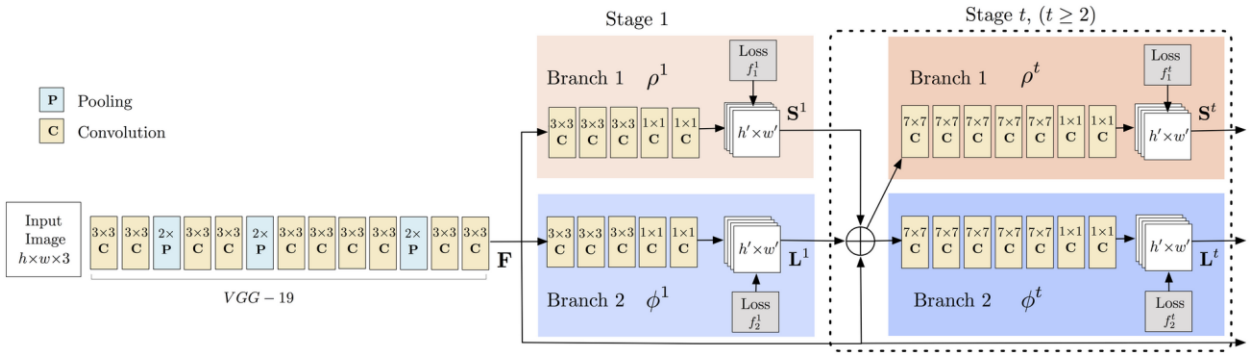



*Figure 3. Architecture of the two-branch multistage CNN*

Each stage in the top branch (beige) predicts a confidence map **S**, and each stage in the bottom branch (blue) predicts a PAF **L**. After every stage, predictions from both branches are concatenated with image features **F** (which come from a VGG-based architecture) and used as input for the next stage. Each branch performs multiple inferences, one per body part.

It is split into two branches: the top branch predicts the detection confidence maps and the bottom branch is for affinity fields. Both branches are organized as an iterative prediction architecture, which refines predictions over successive stages. The improvement of accuracy of predictions is controlled by intermediate supervision at each stage. Here is how it might look on a real image:

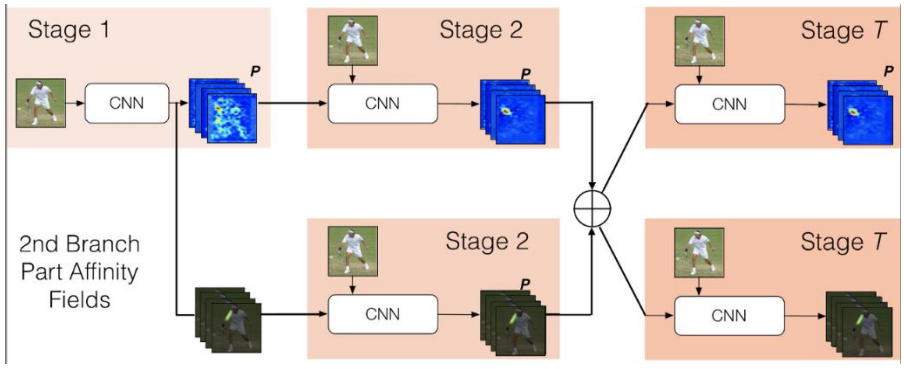



*Figure 4. Demonstration of real image inference by the two-branched architecture neural network.*

Before passing input to this two-branch network the method uses auxiliary CNN (first 10 layers of VGG-19) to extract an input feature map F. This prediction is processed by both branches, and their predictions concatenated with initial F are used as input for the next stage (as features).

## 3.3 From Body Parts to Limbs

To guide the network to iteratively predict confidence maps of body parts in the first branch and PAFs in the second branch (Fig. 3), we apply two loss functions at the end of each stage, one of each branch. We use an L2 loss between the estimated predictions and ground truth maps and fields.

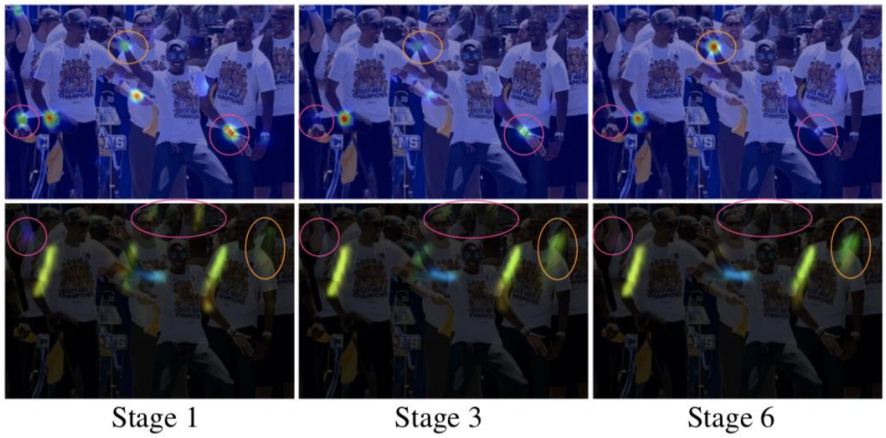



*Figure 5. Confidence maps (first row) and PAFs (second row) across stages.*

We can see that despite confusion on the first stage, the method can fix its mistakes on later stages.

Now consider the top branch; each confidence map is a 2D representation of our confidence that each pixel belongs to a particular body part ("body parts" here are "points" such as wrists and elbows, and, say, forearms are referred to as "limbs" rather than "body parts"). To get body part candidate regions, we aggregate confidence maps for different people. After that, the algorithm performs non-maximum suppression to obtain a discrete set of parts locations:

*Figure 6. How algorithm forms limbs from detections.*

During inference, algorithm computes line integrals over all the PAFs along the line segments between pairs of detected body-parts. If the candidate limb formed by connection of certain pair of points is aligned with corresponding PAF then it's considered as a true limb. This is exactly what the bottom branch does.

## 3.4 From Limps to Full Body Models

The algorithm now has found body part candidates and has scored pairs of these parts (integrating over PAFs), but we still cannot estimate poses because we need the full body model. The final goal is to find the optimal assignment for the set of all possible connections.

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). Here k-partite matching means that the vertices can be partitioned into k groups of nodes with no connections inside each group (i.e., vertices corresponding to the same body part). Edges of the graph are weighted with part affinities. Like this:

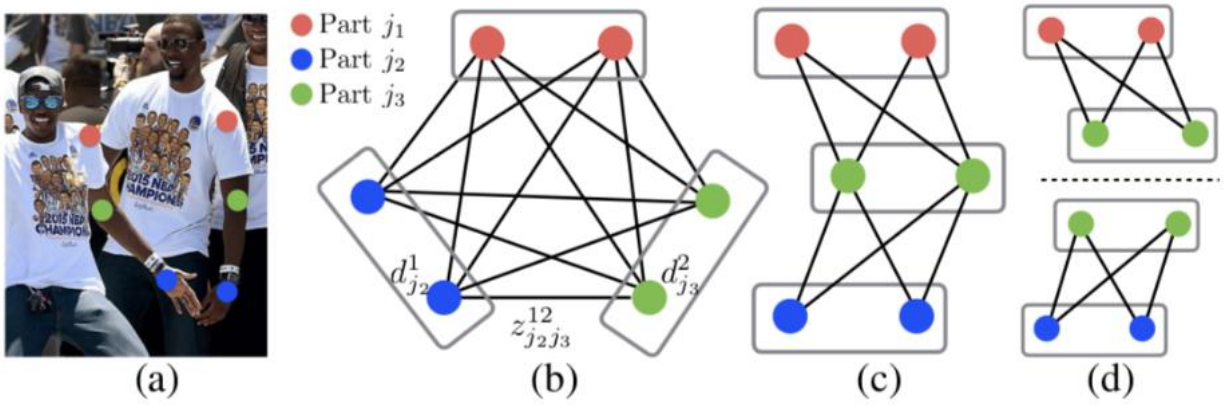



*Figure 6. Graph matching*

The Graph matching problem simplification. (Fig. 6) (a) Original image with part detections, (b) K-partite graph. (c) Tree structure implicitly includes human body model. (d) A set of bipartite graphs.

(Cao et al., 2017)* propose a relaxation where the initial k-partite graph is decomposed into a set of bipartite graphs where the matching task is much easier to solve. The decomposition is based on the problem domain: basically, you know how body parts can connect, and a hip cannot be connected to a foot directly, therefore we can first connect hip to knee and then knee to foot.

# 4 Results

The evaluation on this method goes on two benchmarks for multi-person pose estimation: the MPII human multi-person dataset and the COCO 2016 keypoints challenge dataset. These two datasets collect images in diverse scenarios that contain many real world challenges such as crowding, scale variation, occlusion, and contact. This approach set the state-of-the-art on the inaugural COCO 2016 keypoints challenge, and significantly exceeds the previous state-of-the-art result on the MPII multi-person bench-mark.

## 4.1 Results on the MPII Multi-Person Dataset

For comparison on the MPII dataset, we use the toolkit to measure mean Average Precision (mAP) of all body parts based on the PCKh threshold.

| Method | Hea | Sho | Elb | Wri | Hip | Kne | Ank | **mAP** | s/image |
|---|---|---|---|---|---|---|---|---|---|
| Subset of 288 images | | | | | | | | | |
| Deepcut | 73.4 | 71.8 | 57.9 | 39.9 | 56.7 | 44.0 | 32.0 | 54.1 | 57995 |
| Iqbal et al. | 70.0 | 65.2 | 56.4 | 46.1 | 52.7 | 47.9 | 44.5 | 54.7 | 10 |
| DeeperCut | 87.9 | 84.0 | 71.9 | 63.9 | 68.8 | 63.8 | 58.1 | 71.2 | 230 |
| Ours | **93.7** | **91.4** | **81.4** | **72.5** | **77.7** | **73.0** | **68.1** | **79.7** | **0.005** |
| Full testing set | | | | | | | | | |
| DeeperCut | 78.4 | 72.5 | 60.2 | 51.0 | 57.2 | 52.0 | 45.4 | 59.5 | 485 |
| Iqbal et al. | 58.4 | 53.9 | 44.5 | 35.0 | 42.2 | 36.7 | 31.1 | 43.1 | 10 |
| Ours (one scale) | 89.0 | 84.9 | 74.9 | 64.2 | 71.0 | 65.6 | 58.1 | 72.5 | 0.005 |
| Ours | **91.2** | **87.6** | **77.7** | **66.8** | **75.4** | **68.9** | **61.7** | **75.6** | **0.005** |

Besides these measures, we compare the average inference and optimization time per image in seconds. For the 288 images subset, this method outperforms previous state-of-the-art bottom-up methods by8.5%mAP.
For the entire MPII testing set, this method without scale search already outperforms previous state-of-the-art methods by a large margin, 13% absolute increase on mAP. The mAP comparison with previous bottom-up approaches indicate the effectiveness of our novel feature representation, PAFs, to associate body parts. Based on the tree structure, the greedy parsing method achieves better accuracy than a graphcut optimization formula based on a fully connected graph structure.

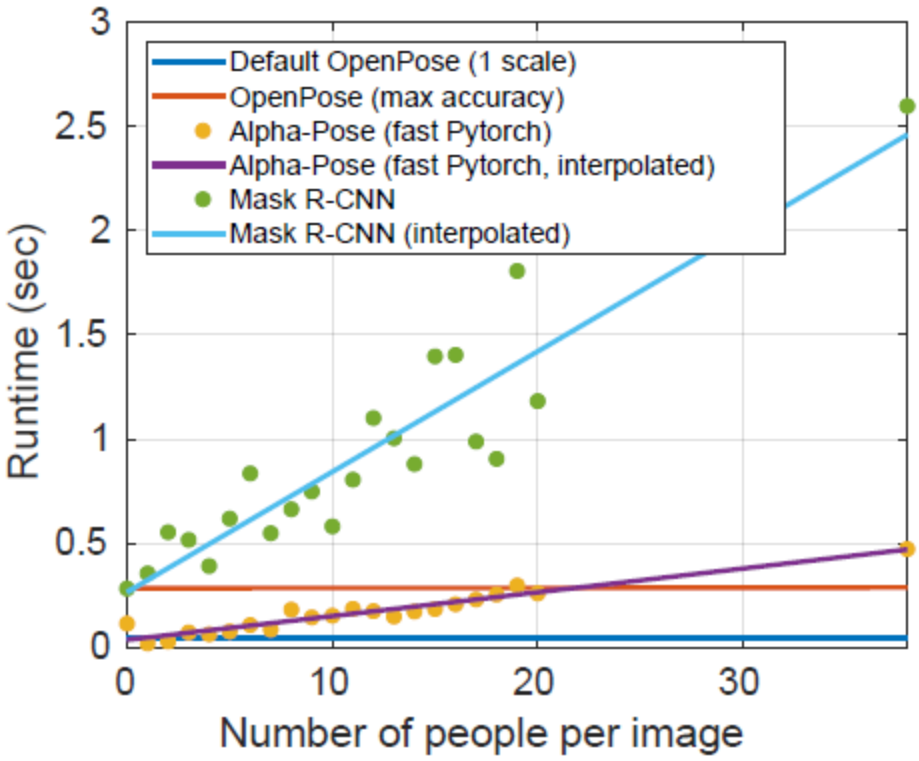
## 4.2 Results on the COCO keypoints challenge

The COCO training set consists of over 100K person in-stances labeled with over 1 million total keypoints. COCO uses a metric called Object Keypoint Similarity (OKS). It gives a measure of how close the predicted Keypoint is with the ground truth. Higher OKS means higher overlap between predicted Keypoints and the ground truth. The OKS plays the same role as the IoU in object detection. OpenPose performs not as well as the top-down methods, It is mainly because there is a higher drop in accuracy when considering only people of higher scales (AP^L).

| Team | **AP** | $AP^{50}$ | $AP^{75}$ | $AP^{M}$ | $AP^{L}$ |
|---|---|---|---|---|---|
| Test-challenge | | | | | |
| Ours | **60.5** | **83.4** | **66.4** | 55.1 | **68.1** |
| G-RMI | 59.8 | 81.0 | 65.1 | **56.7** | 66.7 |
| DL-61 | 53.3 | 75.1 | 48.5 | 55.5 | 54.8 |
| R4D | 49.7 | 74.3 | 54.5 | 45.6 | 55.6 |
| Test-dev | | | | | |
| Ours | **61.8** | **84.9** | **67.5** | 57.1 | **68.2** |
| G-RMI | 60.5 | 82.2 | 66.2 | **57.6** | 66.6 |
| DL-61 | 54.4 | 75.3 | 50.9 | 58.3 | 54.3 |
| R4D | 51.4 | 75.0 | 55.9 | 47.4 | 56.7 |

## 4.3 Inference time

OpenPose nearly remains the same runtime regardless the number of people per image. While top-down approach like Alpha-Pose and Mark R-CNN, the runtime is directly proportional to the number of people per image.




# 5 Resources

All information mentioned in this document is based on the "Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields" paper done by researchers from The Robotics Institute at Carnegie Mellon University (Cao et al., 2017).[link]