Offset Allows for Extra Flexibility: Multitask System for Pose Estimation and Instance Segmentation

E6691.2021Spring.JYZM.report
Jiongxin Ye jy3114, Zhiyuan Ma zm2354
Columbia University

Abstract

Pose estimation and instance segmentation are two well-established computer vision tasks with broad application and prospect, whether a multi-task model on these two tasks would double the supervision remains questionable. To this end, a simple baseline as Faster-rcnn is applied to evaluate the gain of multi-task . Furthermore, this paper investigates a brand-new mechanism called refinement, it refines both feature maps and predictions, and only adds in a few modifications on the baseline model. Experiments are conducted on our proposals and result in noticeable performance gain.

1. Introduction

Pose estimation is a well established task with both scientific and engineering value. As a high level computer vision task, given a RGB image as input, it is designed to inference the locations of body key points and limbs to illustrate human poses. Such models serve as fundamental tools for tasks like action recognition, and implementations including human computer interaction and animation rendering.

Existed methods evolved from single-person pipelines [2, 3, 4, 5, 6] to ones for multiple people. Inherited from object detection, or human bounding-box detection in this case, a bunch of methods for multi-targets follow a single-person prototype, in the other words, the task is split into first detecting every single person and then estimating pose on each of them. They are named as top-down methods[7, 8, 9, 10]. Top-down methods are less sensitive to the complexity of image, especially when the scale of instances varies in a large range. The purposes and advantages of this method are: firstly the performance of instance detection is promised by conventional object detection models; secondly keypoint detection on cropped and resized feature maps is also robust to scale variance. Most SOTA models normally follow this method. However, given the amount of proposals which are candidates of instance bounding boxes is huge, these methods are computationally intensive.

In contrast, there is another cluster of motheds called bottom-up methods, where the keypoints are predicted first through heatmaps, then assigned to different body parts of different people[11, 12, 13, 22]. Normally bottom-up methods are able to achieve much higher inference speed than their counterpart, but both the precision of heatmap and correctness of clustering are challegable when the model faces small scale instances. Therefore a huge performance gap exists between bottom-up and top-down methods.

What challenges this field is the variance in human poses, and appearance including clothing, scale, occlusion, lighting. To tackle this, previous papers proposed single-stage methods by introducing mechanisms to accurately predict in one shot [3, 4], as well as multi-stage, in other words, recurrent architecture[2, 5, 6, 7] to refine predictions. However, this paper refers to multi-task architecture to boost robustness. It is used to aggregate supervision across multiple tasks, in order to generate more distinct representation and benefit each single task. Apart from previous efforts[47, 50], this paper is the first to incorporate pose estimation and instance segmentation. Because the supervision in the Region of Interest is doubled, it solidifies the part of the presentation of importance accordingly, which would outperform other combinations, such as detection+classification[47], detection+semantic segmentation[50], whose other supervision is introduced to the overall representation.

A major subset of methods incorporate prior knowledge into account[31, 37, 38]. They include manual prior knowledge like anchors, as well as adaptive ones like attention. Anchors provide initial status to regress then, while attentions hand over contextual clues on which the model should focus more. The latter is believed to provide extra information that the manual ones has no way to represent, thereby potential increase of performance is possible.

However, this paper proposes a novel adaptive prior knowledge named as refinement achieved by offset. If we regard attention as the information introduced to intermediate feature maps, then offset is able to help both intermediate representation and final prediction. In terms of intermediate representation, offset reshapes vanilla convolution to convolve on places it specifies. As for final prediction, a brand-new module is implemented to evaluate keypoint location precision, and perform adjustment on locations if needed.

The contribution of this paper is three-fold:

- An unsupervised prior-knowledge mechanism called offset is introduced to vanilla pose estimation model (R-CNN based). It is regarded as a refinement, which is helpful to both feature representation and prediction. For feature representation, conventional convolution and pooling can change their shapes and perform on flexible locations. For prediction, It turns to be a precision evaluation module which allows refinement on the inference of original output.
- The first multi-task system to perform instance segmentation and pose estimation is designed. The similar supervisions on features from RoI help the performance of each single task. This system also satisfies multiple needs in implementations.
- Detailed experiments and results are listed, showing that significant increase of performance exists in both proposals.

2. Related Work

2.1 Pose Estimation Methods

Solutions in Pose Estimation could be generally divided into regression-based and detection-based methods.

Regression-based methods map images directly to coordinates of joints [14]. It is widely used in the pose estimation system. Toshev *et al.* used AlexNet-like structure and multi-stage to refine coordinate localization [14]. Pfister *et al.* applied a similar structure on predicting sequence of images [15]. Carreira *et al.* iteratively refined predictions, the final output is refined from mean pose, which is the most probable pose skeleton according to the overall dataset [16]. Similarly, Sun *et al.* designed a body skeleton representation as the anchor, then regresses the offset to fit ground truth [17]. Another brunch just converts heatmap as another classical output to numerical joint coordinates. Luvizon group and Nibali group designed a softmax-like and differentiable spatial transformation to achieve this conversion [18, 19].

Bottom-up methods detect all the keypoints and then assemble groups of keypoints into skeletons for distinct objects [20]. The bottom-up methods are featured as fast processing. As the number of targets in an image goes larger, the computational cost remains unchanged. Cao *et al.* proposed a coarse-to-fine locating strategy called Convolutional Pose Machine, as well as a grouping strategy using Part Affinity Field to depict locations and directions of limbs [23]. Kreiss *et al.* proposed Part Intensity Field and Part Association Field to separately represent joints location and joints association, giving a boost on low-resolution performance [22]. Take note that all methods mentioned above follow the separation of detection and grouping, but there are methods that do both at once. Newell *et al.* achieved this by producing joints heatmap, together with associative embedding maps depicting clustering labels [24]. But key point clustering has still a long way to go, because chances are that keypoints of the same target would be grouped into other targets, which is overwhelmingly challengeable when targets overlap. Jin *et al.* proposed Hierarchical Graph Grouping, which is GNN-based grouping method, increases precision in complex scenarios [25].

Top-down Methods inherit ideas from object detection, in which instances are first picked up, then keypoints are predicted on each of them [20]. Although they are computationally inefficient with way more candidates to process, the points affiliation will not mess up, therefore the output precision is more competitive. Most top-down methods apply existing detectors to generate instance proposals, including Faster R-CNN[26], Mask R-CNN[27], FPN[28]. Major contributions are made on head block. Fang *et al.* combines Hourglass models together with a symmetric spatial transformer network [29]. George *et al*. implemented ResNet-101 and proposed to aggregate heatmap with offset map to make predictions [30]. Xiao *et al.* added deconvolutional layers to ResNet to generate heatmap, in the encoder-decoder fashion [32]. Chen *et al.* immediately used ResNet as backbone followed by up-sampling, then proposed PoseFix to refine prediction according to pose error distributions [31]. Sun *et al.* used HRNet to fuse multi-scale features and made predictions with stronger spatial invariance [34]. Li *et al.* proposed multi-scale fusion in a multi-stage manner [35].

2.2 Methods Using Prior Knowledge

Prior knowledge provides extra information, there are multiple implementations. Some methods focus on giving brand-new representation of keypoints to make training easier. Nie *et al.* saw the skeleton as rays projected from human center point, thus key point location is equivalent to center point location followed by rays fitting [36]. Wei *et al.* added in key point anchors to bounding box anchors, after bounding box proposals of each instance are located, it regresses key point locations from their anchors [37].

A majority of pose estimates are based on hierarchy, that is, attention is put on each key point in order to mine points that are hard to locate. Chen *et al.* proposed online hard point mining to emphasize location error on certain keypoints [31]. Jin *et al.* embed this

hierarchy in grouping keypoints prediction using GNN [38]. A common fashion of hierarchy is set as prior knowledge for the models, but one cannot tell how sufficient this knowledge is to boost training.

2.3 Multi-task methods

The main advantage of multi-task methods is that primal features are shared across branches of different domains, which means gradients from multiple outputs could work together to supervise shared layers. Feng *et al.* applied context segmentation in differentiating parts of the human body, it provides a cross-domain clue for inference, and is partly re-applied in this paper [39]. Georgia *et al.* synchronously performs human detection, pose estimation and action classification, using R-CNN architecture [47]. Luvizon et al. handled 2D and 3D pose estimation at the same time on image sequences [48]. Papandreou group incorporates instance segmentation [49], while Kocabas group takes in semantic segmentation, both report an increase in precision than a single branch [50].

2.4 Variations of Convolution

Apart from conventional convolution, there are variations that provide different feasibility. To achieve parallel and fast processing, a good method is group convolution, it is implemented by the very first CNN model to reserve computational cost and has been even popular for mobile platforms [46]. Features in different groups are not gathered until the final dense net. Zhang *et al.* shuffles these features before next group convolution to enhance model generality [42]. He *et al.* group comes up with a shortcut connection to get over gradience vanishment or explosion, it makes training of outstandingly deep models possible as others [40]. Under normal convolution, channels of features are collectively computed with kernels, while in Chollet's group Depth Wise convolution applies different kernels on different channels, the output of which are then collected by cross-channel convolution [41]. Hu *et al*. comes up with the idea of channel-wise attention, using max pooling to get the weight then paying different attention on channels of features [43]. Dilated Convolution, also known as Atrous Convolution, only takes pixels in certain poles to convolution, so as to increase receptive field without extra parameter to train [44]. Long *et al.* proposed deformable shapes of convolution kernels by computing offsets, hereby convolution evolves further in scaling, rotation robustness [45].

3. Methodology (of the Students' Project)

3.1. Architecture

The model architecture shows in Figure 1, it follows the two-stage style model construction as Mask R-CNN does. Mask RCNN adds another branch to Faster R-CNN so that prediction could be made regarding both object detection and segmentation.It is an outstanding multi-task architecture prototype. It's believed that by introducing keypoint head in the similar way, and hereby supervising over all the tasks, the model is able to learn an higher-level pattern, and push up performance on each individual task themselves. The model contains three heads which are keypoint head, segmentation head and detection head; they share the same RoI feature as input.

By applying Lagrangian, the final objective function is:

$$\mathcal{L} = \mathcal{L}_{KEY} + \mathcal{L}_{SEG} + \mathcal{L}_{OBJ}, \tag{1}$$

where $\mathcal{L}_{KEY}$, $\mathcal{L}_{SEG}$, $\mathcal{L}_{OBJ}$ are the supervisors for keypoint head, segmentation head, detection head, separately.

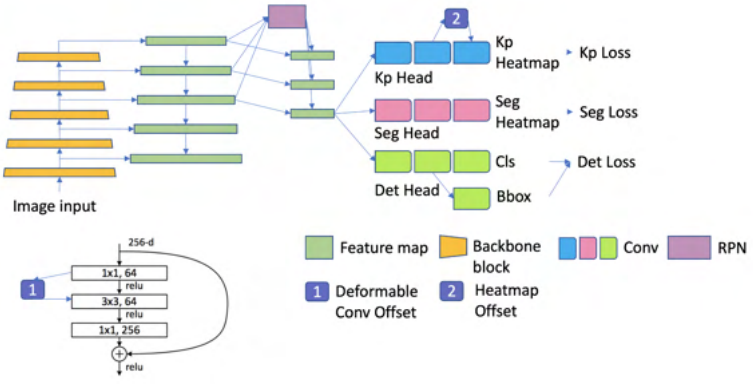

Figure 1: Multi-task system architecture

3.2. Refinement Module

The refinement mechanism can apply to both feature representation and heatmap prediction. Specifically, they are achieved by inserting the offsets to convolution, RoI pooling, and heatmap prediction, separately. The following are detailed elaborations for each of them.

3.2.1 Bilinear Interpolation

Bilinear interpolation is used to obtain the missing value located at the fractional index, it is necessary because deformable convolution is conducted on values there. The main idea of this method is to assign the weighted average of the nearest pixels on the boundary of the 4-neighboring integer index to the fractional index. Given p located at fractional index, we use equation (2) to find the value x(p):

$$\mathbf{x}(\mathbf{p}) = \sum_{\mathbf{q}} G(\mathbf{q}, \mathbf{p}) \cdot \mathbf{x}(\mathbf{q}) \tag{2}$$

where G(q, p) is for picking up 4 nearest values at integer index, in two dimensional matrices, it is achieved by

$$G(\mathbf{q}, \mathbf{p}) = g(q_x, p_x) \cdot g(q_y, p_y) \tag{3}$$

where g(a, b) = max(0, 1 − |a − b|).

3.2.2 Convolution Refinement

A deformable convolution module implements offset by an extra convolution [51]. Through this offset, the kernel would reshape and perform convolution with elements in corresponding areas. The concept of deformable convolution is that convolution itself becomes more scaling and rotation invariant. In a larger sense, more convolution would be made on regions that are critical for prediction. Visualizing the overall offsets would give a huge view on how the model makes the decision, which enhances the model interpretability. This method has been proven improvement on several vision task benchmarks [52].

Specifically, recalling conventional convolution at the point p_0, we have

$$\mathbf{y}(\mathbf{p}_0) = \sum_{\mathbf{p}_n \in \mathcal{R}} \mathbf{w}(\mathbf{p}_n) \cdot \mathbf{x}(\mathbf{p}_0 + \mathbf{p}_n), \tag{4}$$

where p_n denotes the locations on grid R. For a 3 × 3 kernel with dilation 1,

$$\mathcal{R} = \{(-1,-1), (-1,0), \ldots, (0,1), (1,1)\}.$$

When it comes to deformable convolution , define offsets $\{\ddot{Y}\mathfrak{B}_j \mid n=1,...,N\}$, where N is the number of elements in the convolutional kernel. Original convolution is transformed into:

$$\mathbf{y}(\mathbf{p}_0) = \sum_{\mathbf{p}_n \in \mathcal{R}} \mathbf{w}(\mathbf{p}_n) \cdot \mathbf{x}(\mathbf{p}_0 + \mathbf{p}_n + \Delta\mathbf{p}_n) \tag{5}$$

where the offset $\ddot{Y}\mathfrak{B}_j$ forms a matrix with the same shape as $\mathfrak{B}_j \in$ R. To obtain $\ddot{Y}\mathfrak{B}_j$, perform an extra conventional convolution on a given feature map x. To fit the value x(p) located at a fractional index, where $p = \mathfrak{B}_\psi + \mathfrak{B}_j + \ddot{Y}\mathfrak{B}_j$, use bilinear interpolation.

3.2.3 Heatmap Refinement

Given a heatmap x, to refine it at the point p_0, use the offset

$$\mathbf{y}(\mathbf{p}_0) = \mathbf{x}(\mathbf{p}_0 + \Delta\mathbf{p}), \tag{6}$$

where Δp denotes the offset at p_0. Similarly, to fit the value x(p) at fractional index, the bilinear interpolation in Equation 4 is applied.

3.3 Head

The heads of our multi-task architecture is conceptually simple: for every proposal, Faster R-CNN uses two outputs that are class labels and bounding-box regression offset; to this Mask R-CNN added the second branch for instance mask. Naturally then the other output for keypoint localization can be adopted here. Next, we introduce each head in detail.

Detection Head: Recall the detector in Faster R-CNN [26]. Take as input the feature which is the RoI, detection head performs classification and bounding-box regression, the same operations are done for all candidate boxes. For faster inference, both tasks share the same feature. The loss function is defined as:

$$\mathcal{L}_{OBJ} = \mathcal{L}_{cls} + \mathcal{L}_{reg} \tag{7}$$

For our case, the classification loss L_{cls} is binary cross-entropy loss over two classes (person / non-person), it depicts the probability difference between prediction and ground-truth. The regression loss L_{reg} is a smooth L1 loss between the prediction of 4 parameterized coordinate offsets and the corresponding ground-truths. For details including bounding-box regression, please refer to [26].

Segmentation Head: Review the mask head in [26]. Given the Region of Interest, several layers of convolution is applied to get the binary instance mask. Write ground truth heatmap for the s-th category as $\mathbb{Q}_{\rightarrow}^{\overline{\Sigma}}$ while the prediction is $\wp_{\rightarrow}^{\overline{\Sigma}}$, for the dataset with a total of S classes, the loss for mask head is given by:

$$\mathcal{L}_{SEG} = H_{s_0}^{seg} \log F_{s_0}^{seg} \tag{8}$$

Where $\rightarrow_{\text{Ɐ}}$ refers to the ground truth class. For each Region of Interest, the mask head outputs a $e\ddot{\imath}^{\acute{\alpha}}$-dimensional matrix, which encodes S binary masks with the resolution of m × m, one for each of the S classes. The pixel-level sigmoid is applied as the final activation function so that the outputs follow probabilistic distribution. Defining loss only on the class $\rightarrow_{\text{Ɐ}}$ avoids the competition among classes when the mask is about to generate and the gradients to back-propagate. On the inference mode, the dedicated classification branch will decide the output mask of which class to select.

Keypoint Head: In parallel to all heads in Mask-RCNN. The keypoint head is applied on the Region of Interest to compute both the heatmaps which are one channel per keypoint, and offsets which are two channels per keypoint for x/y directions, this follows the style of Mask R-CNN, where all three tasks are performed in parallel. For a dataset with K = 17 keypoints are annotated for each instance, there are a total of 3K channels. The offsets are introduced to heatmaps as are defined in **Equation 6.**

In terms of ground truth annotations, note that both that keypoint head and mask head use heatmap as output, given the Region of Interest, let $\text{Ɔ}_{\mathbb{d}}$ = 1 if the k-th keypoint is located at the position $\text{ϑ}_{\hat{\jmath}}$ or 0 otherwise, where k = 1, . . . , K is indexing the keypoint category and p is indexing the locations on the refined heatmap. Training on barely such ground truth is hard,because of the imbalance between positive and negative annotations. Hereby the soften mechanism is applied to increase the percentile of positive annotations, using the 2D gaussian with standard deviation of 1 px centered on the joint location. Chances are that joints are either truncated or occluded, in these cases a all-zero ground-truth is provided.

Define the MSE loss to measure the L2 distance between ground truths and predictions. When the ground truth heatmap for the k-th keypoint is written as $\mathbb{Q}_{\mathbb{d}}^{\overline{\mathbb{d}}\Sigma}$ while the output from keypoint head is $\wp_{\mathbb{d}}^{\overline{\mathbb{d}}\Sigma}$, the objective function is given by:

$$\mathcal{L}_{KEY} = \sum_{k=1}^{K} \|F_k^{key} - H_k^{key}\|_2^2 \tag{9}$$

Where $\mathbb{d}_{\psi}$ refers to the ground truth class. For each Region of Interest, the keypoint head outputs a $\mathbb{Z}\mathbb{j}^{\acute{\alpha}}$-dimensional matrix, which encodes K binary masks with the resolution of n × n, one for each of the K keypoints. The pixel-level sigmoid is applied so that the outputs follow probabilistic distribution.

Notice the difference of segmentation heads in Mask-RCNN between using ResNet C4 and FPN [27]. Similar modification is applied here, to both segmentation head and keypoint head.

4. Experiment

We implemented a multi-task system for 3 tasks: key point estimation task, instance segmentation task, and object detection task. The experiments results are mainly analyzed in 3 sides:

- The model performance of applying refinement in the backbone and keypoint head.
- The model performance of training with or without segmentation head.
- Differences between our result in instance segmentation and keypoint estimation with others' model results.

4.1. Experimental Setup

Dataset: Our model only trained on **COCO 2017 Train images** dataset (include 118K images, we only use 40K with annotations) and tested on **COCO 2017 Val images** (include 5K images) and **COCO 2017 Test images** datasets (include 41K images). For keypoint estimation, we trained each object with K = 17 keypoints.

Evaluation Metric: Since most of the experience in the COCO dataset for pose estimation and segmentation uses AP to evaluate the performance, we also use AP with OKS to maintain the consistency with other's results. Object keypoint similarity (OKS) shows the similarity between two different human poses. AP shows average precision under recall. We have $\hbar\text{K}\bar{\text{o}}\hbar\text{K}^{\psi\phi}$ $\bar{\text{o}}\hbar\text{K}^{\psi\phi\acute{\imath}}$ for different thresholds at t = 0.5 and t = 0.75 to get precision, while $\hbar$Kis the average of t from 0.5 to 0.95 increased in 0.05.

Architecture: Our backbone models were pre-trained on ImageNet dataset from MSRA. For refinement backbone, we only train the part which uses deformable convolution. 3 tasks are trained at the same time.

Training detail: Models are trained with SGD momentum, the initial learning rate is 0.03 and momentum is set to 0.9. The learning rate has divided by 10 when iteration = 60000, 80000. Each batch contains 12 images. The iteration number is 90000. We try 4 GPUs and 1 GPU in training the regular model to show the GPU influence in model performance.

4.2. Backbone

ResNet models insert shortcut connections in the network architecture, so that the model is in a residual architecture. We select to use 50 layers of ResNet in bottleneck as our backbone model. Instead of applying two ɻ $_{\chi}$ ɻ kernel convolution layers with shortcut, bottleneck use three layers and each layers has kernel size: ω $_{\chi}$ ω, ɻ $_{\chi}$ ɻ, ω $_{\chi}$ ω

This method reduces the size of parameters we should train for the deep layer ResNet model, and keeps the features of the shortcut to make sure we still have high performance. The difference of the ResNet model with or without bottleneck is in Figure 2 [40].

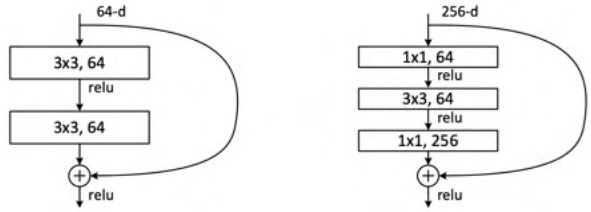

Figure 2: Residual networks description diagram

We have conv1, conv2, conv3, conv4, conv5 to make the ResNet-50. The conv1 block is a convolution layer with kernel size = ĩ $_{\chi}$ ĩ. The conv2 block started at ɻ $_{\chi}$ ɻ max pooling with 3 pairs of convolutional block which kernel size={ω $_{\chi}$ ω, ɻ $_{\chi}$ ɻ, ω $_{\chi}$ ω}. The Res3, Res4, Res5 are constructed with n pairs of convolutional block with kernel size={ω $_{\chi}$ ω, ɻ $_{\chi}$ ɻ, ω $_{\chi}$ ω} where n = {4, 6, 3}. Finally, we use average pooling and fully connected layers with softmax at the end.

4.2. ResNet-50-FPN

Feature Pyramid Network (FPN) is a feature exactor method to detect objects with different scales. We use ResNet-50 with 3 different heads: Detection head (Fast RCNN), Segmentation head (Mask RCNN) and keypoint head (Keypoint RCNN which is an alternative of Mask RCNN). Since they share the same RoI features as input, we can use FPN to predict the output. We generated 2 models based on the different convolution blocks FPN applied. For FPN only use conv5 as input, we called it single scale. For FPN use conv5, conv4, conv3 as inputs, we called it multi scale.

We also try applying refinement in different places. For refinement in backbone, we change the ResNet regular convolution layer in conv3, conv4 and conv5 which has kernel size = ɻ $_{\chi}$ ɻ to the deformable convolution. For refinement in keypoints, we replace the ɻ $_{\chi}$ ɻ kernel layers in keypoint RCNN with the deformable convolution.

Finally, we try to generate a model in single task and multi-task. For a single task model, we only train the keypoint head and detection head. This model can only be tested in keypoint estimation. For a multi-task model, we train all 3 heads. The purpose of this step is trying to figure out whether keypoint estimation performance will be better after adding the segmentation head.

The full model is ResNet-50-FPN in multi scale training in 3 heads and refinement in both backbone and keypoint head.

4.3. Software Design

Our code is built on Detectron2 which is a framework on top of pytorch in the domain of detection and estimation. The code of the multi task model can be found in the direction of configs/COCO-MultiTask/ in our github repository.

The link to the code in our project GitHub is:

https://github.com/ecbme6040/e6691-2021spring-project-jyzm-jy3114-zm2354

5. Results

5.1. Project Results

5.1.1. AP value on different models

We train the model and evaluate our method on COCO benchmark about keypoints challenge and instance segmentation only for the person category. We compared our model with other models. For a fair comparison with the state-of-the-art results, we only use the COCO 2017 training dataset for training and COCO 2017 validation dataset and test-dev split for evaluating. The AP for different model comparisons are shown in Table 1 and

Table 2. According to the experiment results, we have two findings:

1) Multi-task models with convolution refinement can improve the human image's challenge performance. Comparing the results between the full ResNet-50-FPN model with other person instance segmentation models, in general, our model has a higher AP value. We get the highest AP value in ħK^ψø. Comparing the results between the full ResNet-50-FPN model with other keypoint estimation models, higher AP value appeared in our model when we use ħK^ψø õħK^ψøí and ħK^ʒ.

2) Applying FPN in multi-scale or single-scale doesn't make a significant influence in model performance. Single-scale model achieve highest AP value when we use ħK^ψø and ħK^ψøí. Multi-scale model achieve highest AP value when we use ħK^ʒ. The overall highest AP value is 0.891 under ħK^ψø in the single-scale ResNet-50-FPN single-scale. The AP value gaps between single-scale and multi-scale in different settings aren't huge, and the multi-scale model also has the highest. Thus, we can not conclude that the single scale is better than the multi-scale. Although our model doesn't reach the highest AP in each setting, we can find that we got 3 over 5 highest AP values. Overall, our full model is still able to reach high performance in keypoint estimation tasks.

	AP	$AP^{0.50}$	$AP^{0.75}$	AP^{S}	AP^{M}	AP^{L}
FCIS [54] (baseline)	0.334	0.641	0.318	0.090	0.411	0.618
FCIS [54] (multi-scale)	0.386	0.693	0.410	0.164	0.481	0.621
PersonLab [55]:						
ResNet-101 (single-scale)	0.377	0.659	0.394	0.166	0.480	0.595
ResNet-101 (multi-scales)	0.411	0.686	**0.445**	0.215	0.496	0.626
Ours:						
ResNet-50-FPN	**0.477**	**0.765**	**0.445**	**0.239**	**0.515**	**0.649**

Table 1: Performance on COCO person instance segmentation test-dev split. Use the full model in this test. ħK^ψø õħK^ψøí is the different setting in threshold. ħK^e : pixels area under 32*32 detection box AP, ħK^U : pixels area between 32*32 and 96*96 detection box AP, ħK^ʒ : pixels area over 96*96 detection box AP.

	AP	$AP^{0.50}$	$AP^{0.75}$	AP^{M}	AP^{L}
Bottom-up methods:					
Openpose [57] (+refine)	0.618	0.849	0.675	0.571	0.682
Assoc. Embed [58] (multi-scale)	0.630	0.857	0.689	0.580	0.704
PersonLab:					
ResNet-101 [55] (single-scale)	0.655	0.871	0.714	0.613	0.715
Top-down methods:					
Mask R-CNN [26]	0.631	0.873	0.687	0.578	0.714
G-RMI [56] COCO-only(ResNet-101)	0.649	0.855	0.713	0.623	0.700
Ours:					
ResNet-50-FPN(single-scale)	0.644	**0.891**	**0.718**	0.612	0.739
ResNet-50-FPN(multi-scale)	0.648	0.887	0.711	0.605	**0.769**

Table 2: Performance on COCO keypoint test-dev split. Use the single scale and multi-scale FPN in the full model.

5.2. The Impact of Model Refinement

We modify the convolution layers in both backbone and keypoints head to figure out the influence from deformable convolution. We try three models: regular convolution layer, refinement in backbone, refinement in both backbone and keypoint heads. All three models use multi-scale FPN and test in COCO person validation split. We test the AP for each task, the AP value for each model's comparisons are shown in Table 3, Table 4 and Table 5.

As shown in 3 Tables, the result shows that refinement makes a positive influence in our 3 tasks. In object detection task, we get the highest AP value for the model with refinement in both backbone and keypoint head, especially for $\hbar K^{\bar{3}}$ and $\hbar K^{\psi\emptyset\acute{\imath}}$. In the keypoint estimation task, we get 4 over 5 highest AP values for the model with refinement in both than the regular convolution. In the Instance segmentation tasks, 4 high AP values appear in refinement only in backbone or refinement in both.

	AP	$AP^{0.50}$	$AP^{0.75}$	AP^S	AP^M	AP^L
Regular	53.61	82.12	58.41	35.98	61.47	71.04
Ours						
RB	54.55	82.74	58.74	35.94	61.29	73.54
RB+H	**54.72**	**82.97**	**59.48**	**36.02**	**61.63**	**73.37**

Table 3: Object detection performance on COCO person validation split.

	AP	$AP^{0.50}$	$AP^{0.75}$	AP^M	AP^L
Regular	63.90	86.04	69.73	59.77	71.85
Ours					
RB	63.61	85.88	69.28	58.91	72.79
RB+H	**64.10**	**86.52**	**69.82**	**59.21**	**73.24**

Table 4: Keypoint estimation performance on COCO person validation split.

	AP	$AP^{0.50}$	$AP^{0.75}$	AP^S	AP^M	AP^L
Regular	46.73	79.27	50.07	26.92	52.37	65.49
Ours						
RB	**47.56**	**79.81**	51.12	27.21	**52.31**	67.24
RB+H	**47.56**	79.76	**51.30**	**27.24**	52.12	**67.30**

Table 5: Instance segmentation performance on COCO person validation split.

5.3. Comparison between Multi-task and Single-task

We also want to know whether adding the segmentation head will influence the model performance. We trained models with or without the segmentation head in full model and analyzed the performance of keypoint estimation. The result is shown in Table 6. As shown in Table 6, if we do the task for segmentation, the AP of the keypoint task might increase. In the other words, adding segmentation head not hurt the system's performance in keypoint estimation.

Task	AP_{kps}	AP_{mask}
kps-only	0.649	–
kps & mask	**0.657**	0.465

Table 6: Result of Model training with/ without segmentation head

5.4. GPU Effect in Regular Model

We are also interested in different GPU numbers for our system. We use a full model without refinement. Results are shown in Table 7. According to this table, 4 GPUs improve the performance of our multi-task system in all 3 tasks.

	Detection	Segmentation	Keypoint
1 GPU	48.63	41.59	55.74
4 GPUs	**53.61**	**46.73**	**63.90**

Table 7: Training mode under 4 GPUs and 1 GPU.

5.5. Partial body Recognition

Sometimes, not whole human begging will be shown in the images. Our system shows a good performance in 3 tasks even if we only have some partial body parts such as hands in input images. Also, for images with lots of tiny people,our system can recognize them in high accuracy. Prediction examples are shown in Figure 3, Figure 4.

6. Conclusion

Our project design a multi-task system with refinement to predict human keypoints, person detection and person segmentation. Experiment conducted on the COCO 2017 datasets. The outstanding performance of our system is confirmed by comparing the result between our system with others' model, changing the refinement part in the full model, and comparing the result between model trained in single task and multi-task, the performance of our multi-task systems. Positive influences made by refinement are shown in our experiments. Our system achieves state-of-the-art results on the COCO dataset. In conclusion:

1) Refinement in convolution layers can increase the performance of human pose estimation systems.
2) It's feasible to generate a multi-task system by adding a head.
3) Our system works under different kinds of human images. The body shape in images can be

either covered by other items except only the whole body.

4) Since we use images with 17 keypoints to train the model, the predicted images may show at most 17 joints.
5) Also, increasing the setting of GPU makes the system more accurate.

7. Acknowledgement

Thanks for the detectron2 repository in Github helping us to build a clean, strong network: https://github.com/facebookresearch/detectron2/tree/72059968a2b2337ab34c86ddcbfc2f22e6914ff3

Figure 3: Prediction result for tiny people and covered body objects

Figure 4: Prediction result partial body and small

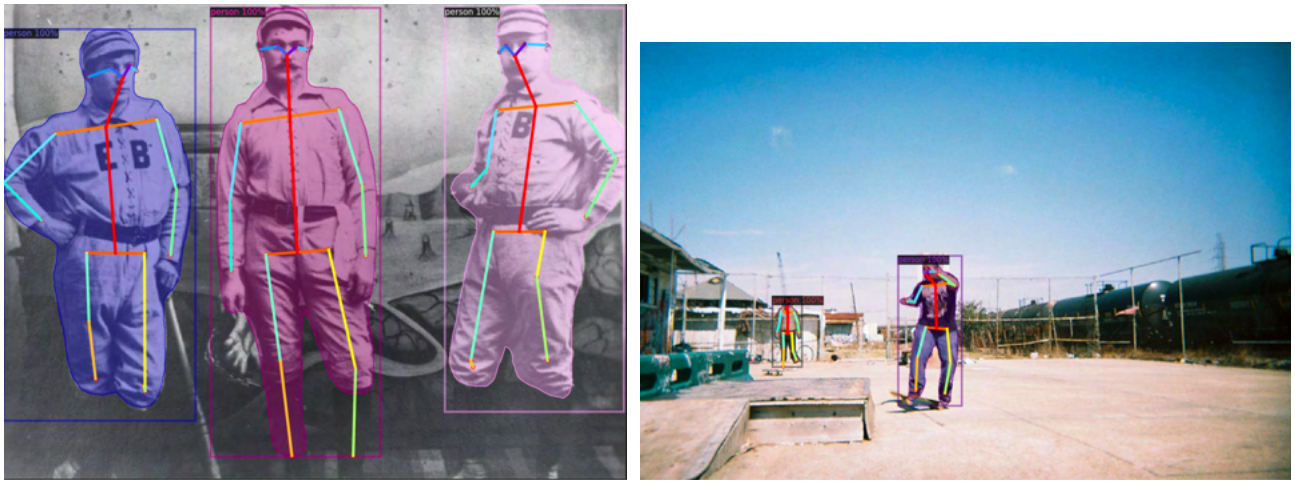

Figure 5: Some examples of our system

8. References

[1] Our Github repository link: https://github.com/ecbme6040/e6691-2021spring-project-jyzm-jy3114-zm2354

[2] Wei, Shih-En, et al. “Convolutional pose machines.” Proceedings of the IEEE conference on Com- puter Vision and Pattern Recognition. 2016.

[3] Chen, Xianjie, and Alan Yuille. “Articulated pose estimation by a graphical model with image dependent pairwise relations.” arXiv preprint arXiv:1407.3399 (2014).

[4] Chen, Yu, et al. “Adversarial posenet: A structure-aware convolutional network for human pose estimation.” Proceedings of the IEEE International Conference on Computer Vision. 2017.

[5] Newell, Alejandro, Kaiyu Yang, and Jia Deng. “Stacked hourglass networks for human pose estima- tion.” European conference on computer vision. Springer, Cham, 2016.

[6] Chu, Xiao, et al. “Multi-context attention for human pose estimation.” Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017.

[7] Toshev, Alexander, and Christian Szegedy. “Deeppose: Human pose estimation via deep neural networks.” Proceedings of the IEEE conference on computer vision and pattern recognition. 2014.

[8] Fang, Hao-Shu, et al. “Rmpe: Regional multi-person pose estimation.” Proceedings of the IEEE International Conference on Computer Vision. 2017.

[9] Papandreou, George, et al. “Towards accurate multi-person pose estimation in the wild.” Proceed- ings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017.

[10] Iqbal, Umar, and Juergen Gall. “Multi-person pose estimation with local joint-to-person associa- tions.” European Conference on Computer Vision. Springer, Cham, 2016.

[11] Pishchulin, Leonid, et al. “Deepcut: Joint subset partition and labeling for multi person pose estimation.” Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[12] Insafutdinov, Eldar, et al. “Deepercut: A deeper, stronger, and faster multi-person pose estimation model.” European Conference on Computer Vision. Springer, Cham, 2016.

[13] Insafutdinov, Eldar, et al. “Arttrack: Articulated multi-person tracking in the wild.” Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

[14] Toshev, Alexander, and Christian Szegedy. “Deeppose: Human pose estimation via deep neural networks.” Proceedings of the IEEE conference on computer vision and pattern recognition. 2014.

[15] Pfister, Tomas, et al. “Deep convolutional neural networks for efficient pose estimation in gesture videos.” Asian Conference on Computer Vision. Springer, Cham, 2014.

[16] Carreira, Joao, et al. “Human pose estimation with iterative error feedback.” Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[17] Sun, Xiao, et al. “Compositional human pose regression.” Proceedings of the IEEE International Conference on Computer Vision. 2017.

[18] Luvizon, Diogo C., Hedi Tabia, and David Picard. “Human pose regression by combining indirect part

detection and contextual information." Computers and Graphics 85 (2019): 15-22.
[19] Nibali, Aiden, et al. "Numerical coordinate regression with convolutional neural networks." arXiv preprint arXiv:1801.07372 (2018).
[20] Jin, Sheng et al. "Towards Multi-Person Pose Tracking: Bottom-up and Top-down Methods" (2017)
[21] Chen, Yilun, et al. "Cascaded pyramid network for multi-person pose estimation." Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.
[22] Kreiss, Sven, Lorenzo Bertoni, and Alexandre Alahi. "Pifpaf: Composite fields for human pose esti- mation." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019
[23] Cao, Zhe, et al. "Realtime multi-person 2d pose estimation using part affinity fields." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.
[24] Newell, Alejandro, Zhiao Huang, and Jia Deng. "Associative embedding: End-to-end learning for joint detection and grouping." arXiv preprint arXiv:1611.05424 (2016).
[25] Jin, Sheng, et al. "Differentiable hierarchical graph grouping for multi-person pose estimation." European Conference on Computer Vision. Springer, Cham, 2020.
[26] Ren, Shaoqing, et al. "Faster r-cnn: Towards real-time object detection with region proposal net- works." arXiv preprint arXiv:1506.01497 (2015).
[27] He, Kaiming, et al. "Mask r-cnn." Proceedings of the IEEE international conference on computer vision. 2017.
[28] Lin, Tsung-Yi, et al. "Feature pyramid networks for object detection." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.
[29] Fang, Hao-Shu, et al. "Rmpe: Regional multi-person pose estimation." Proceedings of the IEEE International Conference on Computer Vision. 2017.
[30] Papandreou, George, et al. "Towards accurate multi-person pose estimation in the wild." Proceed- ings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017.
[31] Chen, Yilun, et al. "Cascaded pyramid network for multi-person pose estimation." Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.
[32] Xiao, Bin, Haiping Wu, and Yichen Wei. "Simple baselines for human pose estimation and tracking." Proceedings of the European conference on computer vision (ECCV). 2018.
[33] Moon, Gyeongsik, Ju Yong Chang, and Kyoung Mu Lee. "Posefix: Model-agnostic general human pose refinement network." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019.
[34] Sun, Ke, et al. "Deep high-resolution representation learning for human pose estimation." Proceed- ings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019.
[35] Li, Wenbo, et al. "Rethinking on multi-stage networks for human pose estimation." arXiv preprint arXiv:1901.00148 (2019).
[36] Nie, Xuecheng, et al. "Single-stage multi-person pose machines." Proceedings of the IEEE/CVF International Conference on Computer Vision. 2019.
[37] Wei, Fangyun, et al. "Point-set anchors for object detection, instance segmentation and pose esti- mation." European Conference on Computer Vision. Springer, Cham, 2020.
[38] Jin, Sheng, et al. "Differentiable hierarchical graph grouping for multi-person pose estimation." European Conference on Computer Vision. Springer, Cham, 2020.
[39] Feng, Shenming, Xiying Li, and Haifeng Hu. "Combining Parsing Information With Joint Structure for Human Pose Estimation." IEEE Access 8 (2020): 123408-123418.
[40] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE con- ference on computer vision and pattern recognition. 2016.
[41] Chollet, Franc ̧ois. "Xception: Deep learning with depthwise separable convolutions." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.
[42] Zhang, Xiangyu, et al. "Shufflenet: An extremely efficient convolutional neural network for mobile devices." Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.
[43] Hu, Jie, Li Shen, and Gang Sun. "Squeeze-and-excitation networks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.
[44] Long, Jonathan, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.
[45] Dai, Jifeng, et al. "Deformable convolutional networks." Proceedings of the IEEE international conference on computer vision. 2017.
[46] Russakovsky, Olga, et al. "Imagenet large scale visual recognition challenge." International journal of computer vision 115.3 (2015): 211-252.
[47] Gkioxari, Georgia, et al. "R-cnns for pose estimation and action detection." arXiv preprint arXiv:1406.5212 (2014).
[48] Luvizon, Diogo C., David Picard, and Hedi Tabia. "2d/3d pose estimation and action recognition using multi task deep learning." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018.
[49] Papandreou, George, et al. "Personlab: Person pose estimation and instance segmentation with a bottom-up, part-based, geometric embedding model." Proceedings of

the European Conference on Computer Vision (ECCV). 2018.
[50] Kocabas, Muhammed, Salih Karagoz, and Emre Akbas. “Multiposenet: Fast multi-person pose estimation using pose residual network.” Proceedings of the European conference on computer vision (ECCV). 2018.
[51] Dai, Jifeng, et al. “Deformable convolutional networks.” Proceedings of the IEEE international conference on computer vision. 2017.
[52] Zhu, Xizhou, et al. “Deformable convnets v2: More deformable, better results.” Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019.

9. Appendix

9.1 Individual Student Contributions in Fractions

	jy3114	zm2354
Last Name	Ye	Ma
Fraction of (useful) total contribution	1/2	1/2
What I did 1	Paper Review	Network Implement
What I did 2	Model Training	Model Training
What I did 3	Report Writing	Report Writing

8.2 If/as needed: additional diagrams, source code listing, circuit schematics, relevant datasheets etc.