Environment Upgrade Reinforcement Learning for Non-differentiable Multi-stage Pipelines

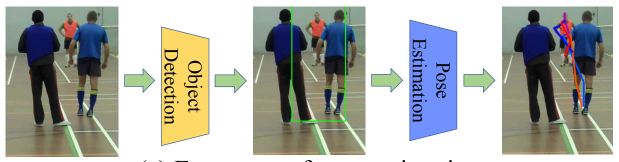
Shuqin Xie[1], Zitian Chen[2‡], Chao Xu[1], Cewu Lu[1*†]
[1]Shanghai Jiao Tong University, China [2] Fudan University, China
{qweasdshu, xuchao.19962007, lucewu}@sjtu.edu.cn, ztchen15@fudan.edu.cn

Abstract

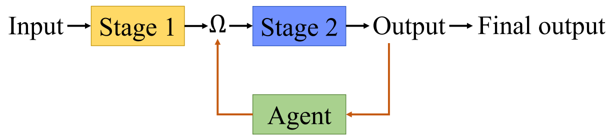
Recent advances in multi-stage algorithms have shown great promise, but two important problems still remain. First of all, at inference time, information can't feed back from downstream to upstream. Second, at training time, end-to-end training is not possible if the overall pipeline involves non-differentiable functions, and so different stages can't be jointly optimized. In this paper, we propose a novel environment upgrade reinforcement learning framework to solve the feedback and joint optimization problems. Our framework re-links the downstream stage to the upstream stage by a reinforcement learning agent. While training the agent to improve final performance by refining the upstream stage's output, we also upgrade the downstream stage (environment) according to the agent's policy. In this way, agent policy and environment are jointly optimized. We propose a training algorithm for this framework to address the different training demands of agent and environment. Experiments on instance segmentation and human pose estimation demonstrate the effectiveness of the proposed framework.

1. Introduction

Combining different building blocks of computer vision to perform more sophisticated tasks is an appealing idea, especially with the impressive advancement achieved in building blocks algorithms such as object detection [13, 12, 39, 27, 25, 20, 38], semantic segmentation[9, 24, 5, 14, 34, 35] and pose estimation [36, 43, 44, 2, 32, 8]. Some recent works [11, 33] attempt to utilize these improvements, making huge progress over previous methods[37, 17]. For example, Fang *et al.*[11] propose a framework for multi-person pose estimation. They first adopt a human detector[27] to detect human instances then apply single person pose estimation algorithm [31] to each human instance, achieving state-of-the-art performance on multiple datasets.



(a) Error case of pose estimation



(b) Our multi-stage feedback pipeline

Figure 1: (a) Error case of two-stage pose estimation. If the object detection generates a wrong bounding box, the pose estimation algorithm will fail too. (b) An illustration of our multi-stage feedback pipeline. Ω is the output of Stage 1. An agent observes the output of Stage 2 and choose an action to refine Ω.

However, two important problems exist in these multi-stage algorithms. First of all, information can't feed back from downstream (second stage) to upstream (first stage). Therefore, the overall performance heavily relies on the accurate output of the first stage. An inaccurate output from the first stage will result in the failure of the second stage, and there is no way to correct this error. For instance, in [11], if the human regions are wrong, the pose estimation will fail too. See Figure 1a. Nevertheless, the second stage's output can provide useful information to guide the first stage's output.

Second, not all multi-stage frameworks can be trained in an end-to-end manner. The connection among different stages often involves non-differentiable functions, such as cropping or distortion. In this situation, different stages can only be trained separately and fine-tuned to fit each other. This training setting fails to jointly optimize different stages, which is obviously less optimal. Also, we demon-

[*]Cewu Lu is the corresponding author
[†]Cewu Lu is a member of SJTU-SenseTime lab and a member of AI research institute at SJTU
[‡]This work was done when Zitian Chen was an intern at MVIG lab of Shanghai Jiao Tong University

strate in Section 4.2.2 that the finetuned model may not perform as well as the original model, again proving the shortcomings of this strategy.

To address the above problems, a novel environment upgrade reinforcement learning framework is proposed. As illustrated in Figure 1b, this framework uses a reinforcement learning agent to re-link the two separate stages. The agent observes the output of the second stage and chooses an action to refine the output of the first stage. In this way, information can feed back from the second stage to the first stage.

Specifically, the goal of this framework is to improve the final performance, which reduces to two subtasks: for the agent to learn a policy to refine the imperfect output of the first stage; and for the second stage to improve its recognition performance under the agent's policy. For the first task, by designing a reward function that encourages improvement of the final recognition result, we can train the agent to learn a good policy through reinforcement learning techniques[30, 29]. For the second task, we upgrade the second stage's network using data generated by the agent's policy under supervised learning. Different from traditional reinforcement learning frameworks which have fixed environments, our framework upgrades the environment with the agent policy. As the agent policy improves, the environment also improves; in turn, the enhanced environment can help the agent learn a more advantageous policy. In this way, we achieve joint optimization of environment and agent policy.

We present the training algorithm for this framework. Training this framework is a non-trivial problem because it involves the interaction of reinforcement learning and supervised learning. Care must be taken to combine them together. We propose an iterative training process to satisfy the different demands of training the agent and the second stage's network. Finally, we demonstrate the effectiveness of our framework in an instance segmentation task and a pose estimation task.

To summarize, the contribution of this paper is:

- We propose a novel environment upgrade reinforcement learning framework for non-differentiable multi-stage pipeline, enabling information to feed back from the second stage to the first stage as well as joint optimization of the environment and the agent.
- We present the training algorithm for this framework, which can address the different demands of training the agent and the second stage's network.

2. Related Work

2.1. Recognition

Since our ultimate goal is to improve the final recognition performance, some recent progress of recognition is investigated. Convolutional neural network (ConvNet, or CNN) has been the cornerstone in many image recognitions algorithm for its significant performance on feature extraction.

Human pose estimation. Compared to traditional methods[36] that used pictorial structures model to simulate the articulation connection of human body, ConvNet can accomplish better performance due to its higher model capacity and more sophisticated representation ability. DeepPose[43] was the first work that applied ConvNet to estimate human pose, and significantly improved the accuracy. After that, a large body of work [44, 2, 32] further promoted the performance by designing better network architectures as well as a more reasonable loss function. Nowadays, generative adversarial networks (GAN)[7, 6] are applied to predict poses by exploiting more geometric constraints of joint interconnectivity, achieving remarkable results.

Instance segmentation. [34] uses a kind of discriminative convolutional network which performs class-agnostic segmentation and then object classification in two stages. They go on to propose an augmented feed-forward net for a novel top-down refinement approach in [35]. [9] proposes a cascaded structure containing differentiating instances, estimating masks, and categorizing objects. [24] first presents a fully convolutional end-to-end solution with instance mask prediction and classification training jointly, winning the first prize in COCO 2016 segmentation challenge. [14] adds a branch for predicting an object mask in parallel with the existing branch for bounding box recognition in Faster R-CNN, defeating the state-of-the-art method.

2.2. Reinforcement learning for vision tasks

Reinforcement learning has attracted attention from researchers in computer vision with its general framework and the potential to be applied to non-differentiable tasks.

Active search. [3, 28, 21] apply reinforcement learning to active object search. [3] views finding an object in an image as a sequential decision-making process. It starts from the entire image and gradually narrows down the bounding box to an object. Different from [3], [28] trains an agent to predict where to look in each step and samples a set of proposals from there. Compared to sliding window methods, this achieves a speed-up of two orders in magnitude. [21] proposes a collaborative multi-agent reinforcement learning algorithm to utilize the contextual information and allows multiple agents to communicate with each other. Different from these works whose environments are fixed, we incorporate a set of parameters into the environment so that environment can also be improved.

Additional topics. [40] proposes an actor-critic framework to apply reinforcement learning in image captioning, achieving state-of-the-art results in MSCOCO dataset. [22]

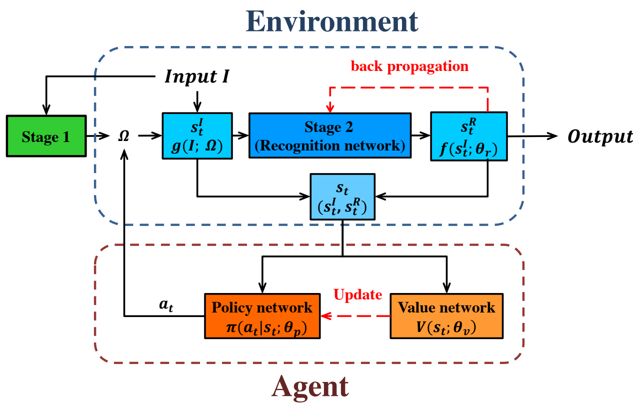
trains an agent to gradually refine 6D pose estimation. The agent directly views pose estimation algorithm as a policy network and its task is to choose which joint to refine next. Different from [22], we integrate the pose estimation as a recognition network in the environment, and the action operates on the previous stage's output rather than the order of joints to refine. [41] applies reinforcement learning techniques to the video tracking problem. Benefiting from the sparse reward of reinforcement learning, [41] is able to quickly train on massive datasets, achieving several orders of magnitude speed-up.

3. Approach

Our goal is to improve the final recognition performance, which reduces to two subtasks: training an agent to refine the imperfect output of the first stage; and upgrading the second stage's network to improve its performance under the agent's policy. We model these two objectives under a unified environment upgrade reinforcement learning framework (EU-RL).

We start with an overview of this framework and describe each component in detail. Lastly, we present the training and inference algorithm for this framework.

3.1. Overview



Environment
Input I
back propagation
Stage 1
Ω
s_t^I $g(I;\ \Omega)$
Stage 2 (Recognition network)
s_t^R $f(s_t^I;\theta_r)$
Output
s_t (s_t^I, s_t^R)
a_t
Policy network $\pi(a_t|s_t;\theta_p)$
Update
Value network $V(s_t;\theta_v)$
Agent

Figure 2: Pipeline of EU-RL framework. At each step, $g(I;\Omega)$ generates input s_t^I for the recognition network, which then generates a recognition prediction s_t^R. The agent takes (s_t^I, s_t^R) as input, samples an action a_t from policy $\pi(a_t|s_t;\theta_p)$ to refine Ω. This changes the recognition network's input at next step. This process is repeated until the agent decides to terminate the process.

Notation: We denote the input image as I and the output of the first stage to be a set of parameters Ω. A function $g(I;\Omega)$ then uses Ω to generate input for the second stage's network. Since the second stage's network is usually responsible for the recognition task, without loss of generality, we use 'recognition network' to represent the second stage's network.

Overview: The pipeline of this framework is illustrated in Figure 2. We view the process of refining the first stage's output to improve final recognition performance as a sequential decision-making process. At each step, the agent observes both the recognition network's input and output, and chooses an action to refine the first stage's output Ω. This interaction determines the input of the recognition network in the next step. A new recognition output is then generated and fed into the agent again, along with the new input. This process repeats until the agent decides to terminate the process.

3.2. Components

Environment: Typically, given a dataset of images $D = \{I_1, I_2, \cdots, I_N\}$, each image is viewed as an environment. Different from traditional reinforcement learning frameworks which maintain a fixed environment, our environment contains a recognition network $f(\cdot;\theta_r)$, where θ_r are the parameters of the recognition network. In this setting, the environment can be upgraded to help agent learn a better policy.

State: The state representation s_t in EU-RL contains two components: (s_t^I, s_t^R). s_t^I is the input of the recognition network, which is generated by a non-differentiable function $g(I;\Omega)$. For example, $g(\cdot;\Omega)$ is a cropping function which crops a rectangle region Ω from image I. Then s_t^I is fed into the recognition network to generate a recognition output $s_t^R = f(s_t^I;\theta_r)$. These two components are essential and complementary because s_t^R indicates how accurate the recognition output is, while s_t^I provides visual cues for the agent to tune Ω.

Action: The action set $\mathcal{A}$ is defined as a set of operations that change the parameters Ω, and thus change the state of the environment. For example, if $g(I;\Omega)$ is a cropping function, then action set $\mathcal{A}$ can be defined as the actions that change a bounding box's location, scale and aspect ratio, e.g.,

$\mathcal{A} = \{$ *move left, move right, move up, move down, scale up width, scale down width, scale up height, scale down height* $\}$.

$\mathcal{A}$ also includes another action *terminate* that indicates the agent has finished the refinement process, leaving s_t^R as the final recognition result.

Reward: At each step, after choosing an action a_t, the agent receives a reward r_t from the environment. Reward r_t directly relates to the quality of recognition prediction. We hope the agent can improve the recognition prediction at each step and keep the loss from the last step under a small threshold. Therefore, we introduce a loss function $L(\cdot,\cdot)$ for each recognition prediction s_t^R and the ground truth label l_t, and formulate the reward function as

$$r_t = \begin{cases} \alpha \cdot sign\left[L(s_t^R, l_t) - L(s_{t+1}^R, l_{t+1})\right]; & a_t \neq terminate \\ sign\left[\epsilon - L(s_t^R, l_t)\right]; & a_t = terminate \end{cases} \quad (1)$$

where ϵ is a threshold and $sign[\,\cdot\,]$ returns 1 (-1) for positive (negative) value. Scale factor α is set to 0.1 in this paper. Eq. (1) indicates the agent receives a positive reward if the chosen action improves the recognition quality and receives a penalty if it decreases the quality. If the agent chooses to terminate the process, the final recognition prediction must be good enough, otherwise it would receive a large penalty.

Agent: Our agent is an actor-critic agent [19] which consists of a policy network and a value network. The policy network parameterizes a policy $\pi(a_t|s_t;\theta_p)$ with parameters θ_p, serving as the actor. The value network $V(s_t;\theta_v)$ approximates the total expected reward of state s_t with parameters θ_v, serving as the critic. The policy $\pi(a_t|s_t)$ is the probability of choosing action a_t under state s_t.

3.3. Training and Inference

Training this framework involves two tasks: training the agent by reinforcement learning and upgrading the recognition network for the agent by supervised learning. We start by introducing the learning algorithm of the agent and then discuss how to upgrade the environment for the agent.

Training agent. Under the framework of reinforcement learning, the goal of the agent is to maximize its expected reward over all the images. The objective function for this goal is formulated as:

$$J = \max\left[\frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}^{\infty}\gamma^t r_t^i\right], \quad (2)$$

where γ is a discount factor.

At each step, an action a_t is sampled according to the policy $\pi(a_t|s_t;\theta_p)$ and the value network outputs a value $V(s_t;\theta_v)$ to estimate the expected total reward in current state s_t. We use Temporal Difference (TD) learning[42] to update the agent's parameters. The output of the value network at next state s_{t+1} is approximated as the real reward at s_{t+1} and is used to update the agent's parameters. Therefore, the TD error E_t is given by

$$E_t \leftarrow r_t + \gamma \cdot V(s_{t+1};\theta_v) - V(s_t;\theta_v).$$

According to [45], the formula for the policy gradient is

$$\nabla_{\theta_p} J = \nabla_{\theta_p} \log \pi(a_t|s_t;\theta_p) \cdot E_t.$$

In this way, we can train the policy network using standard back-propagation algorithms.

Upgrading environment. Next, we describe how to upgrade the recognition network for the agent. Intuitively, we would use all data generated by the agent to train the recognition network. However, in our experiment, this results in a decrease of the final accuracy. The reason is that we are only concerned about the recognition performance in the last step. If we use all the data to train the recognition network, the goal of the recognition network is then to minimize the loss function over the entire sequence, which may not result in the same accuracy as simply minimizing the loss function in the last step.

Note that since the policy network uses the recognition network's prediction as input, intuitively, we can back-propagate the gradient from the policy network to the recognition network. However, our experiment shows back-propagating the gradient from the policy network would in fact decrease the recognition network's performance. We argue that the goal of the policy network is not in line with the recognition network's, and the resulting information may be useless, or even harmful, for the recognition network's learning.

The details of the training algorithm are outlined in Algorithm 1.

Algorithm 1 EU-RL training

```
Initialize recognition network parameters $\theta_r$
Initialize policy network parameters $\theta_p$ and value network parameters $\theta_v$
Initialize agent learning rate $\alpha$, recognition network learning rate $\beta$
Initialize discount factor $\gamma$
for episode = 1,M do
    Set $t \leftarrow 1$
    Get initial RGB observation $s_t^I$
    Get initial recognition prediction $s_t^R \leftarrow f(s_t^I;\theta_r)$
    State $s_t \leftarrow (s_t^I, s_t^R)$
    // Learning Agent
    repeat
        Sample $a_t$ from policy $\pi(a_t|s_t;\theta_p)$
        Execute action $a_t$, get new RGB image observation $s_{t+1}^I$
        Get new recognition prediction $s_{t+1}^R \leftarrow f(s_{t+1}^I;\theta_r)$
        State $s_{t+1} \leftarrow (s_{t+1}^I, s_{t+1}^R)$
        Receive reward $r_t$
        // TD-learning
        $E_t \leftarrow r_t + \gamma \cdot V(s_{t+1};\theta_v) - V(s_t;\theta_v)$
        $\theta_p \leftarrow \theta_p + \alpha \cdot \nabla_{\theta_p} \log \pi(a_t|s_t;\theta_p) \cdot E_t$
        $\theta_v \leftarrow \theta_v + \alpha \cdot 2E_t \cdot \frac{\partial V(s_t;\theta_v)}{\partial \theta_v}$
        $t \leftarrow t + 1$
    until $t = t_{max} + 1$ or $a_{t-1} = terminate$
    // Upgrading Environment
    $\theta_r \leftarrow \theta_r + \beta \cdot \frac{\partial L(s_{t-1}^R, l_{t-1})}{\partial s_{t-1}^R} \frac{\partial s_{t-1}^R}{\partial \theta_r}$
end for
```

Inference. At inference stage, the agent keeps sampling an action a_t at each step by choosing the one with maximal probability in the policy $\pi(a_t|s_t;\theta_p)$ until the *terminate* action is sampled. The recognition output at the last step is returned as the final prediction. Please see Algorithm 2.

Algorithm 2 EU-RL inference

```
Initialize $\theta_r, \theta_p, \theta_v$ from pre-trained model
Set $t \leftarrow 1$
Get initial RGB image observation $s_t^I$
Get initial recognition prediction $s_t^R \leftarrow f(s_t^I;\theta_r)$
State $s_t \leftarrow (s_t^I, s_t^R)$
repeat
    $a_t \leftarrow \arg\max_{a_t} \pi(a_t|s_t;\theta_p)$
    Execute action $a_t$, get new RGB image observation $s_{t+1}^I$
    Get new recognition prediction $s_{t+1}^R \leftarrow f(s_{t+1}^I;\theta_r)$
    State $s_{t+1} \leftarrow (s_{t+1}^I, s_{t+1}^R)$
    $t \leftarrow t+1$
until $t = t_{max}+1$ or $a_{t-1} = terminate$
return $s_{t-1}^R$
```

4. Experiment

In this section, we design two experiments to verify the effectiveness of our framework. We begin with an instance segmentation experiment to examine the performance of our framework under different conditions. Then, we conduct a multi-person pose estimation experiment, where our framework achieves state-of-the-art performance in a challenging dataset. Finally, we explain that the finetuned model may not perform as well as the original model.

4.1. Instance Segmentation

In two-stage pipelines, because the different stages are trained separately, it's often the case that the output of the first stage doesn't quite match the training data of the second stage. In this situation, the input data distribution is different from the training data distribution and the second stage's model may not perform very well. This experiment is designed to explore how our framework can help improve the performance when the input data doesn't match the training data.

To satisfy our demand, we construct a subset from the MSCOCO dataset [26] and design several rules for all our baseline models. We first introduce dataset construction details and the rules for this task. Then, we set up several baselines to offer fair comparisons and report the results of the experiments.

Dataset construction: To satisfy our purpose, we extract a subset D from the MSCOCO dataset [26]. Specifically, we remove images that contain more than one object and images in which the object's size is too small (less than 32 x 32 pixels). Keeping images that only contain one object allows us to perform further processing without causing too many problems. The total number of images in D is 14028, 9334 of which are used for training and the rest for testing.

Design rules:

- A pretrained Deepmask model [34] is used for all baselines as the segmentation network. This model is trained on the tight-bounding-box cropped images.
- The characteristics of the data provided in this task don't match the training data of the Deepmask model. For example, not all images in D are tightly cropped and object-centric. This shows the difference in data distribution for our setting — a setting in which the pretrained Deepmask model does not work well.

Baselines setting: We design several baselines to offer a fair comparison. The first one is to finetune the pretrained Deepmask model to directly predict the mask for the images in D. We denote this as "Finetune-Deepmask". The second one is to train an agent under our framework to gradually narrow down the cropping area from the entire image to the object's location. This one is called "Agent-Deepmask". The third baseline is to train a network to directly regress a bounding box and then apply the pretrained Deepmask model on this bounding box. This one is called "Regress-Deepmask". The fourth one is to insert an agent between the regressing network and pretrained Deepmask model and jointly optimize the agent and Deepmask model under our EU-RL framework. This one is called "Regress-Agent-Deepmask". To compare with the Spatial Transformer Network (STN), we also insert a STN module between the regressing network and Deepmask module, calling this baseline "Regress-STN-Deepmask". Last, we directly apply the Deepmask model on the ground truth bounding box to test the upper bound of these baselines. This one is called "GT Bbox-Deepmask".

Segmentation

Model	Average IoU
Finetune-Deepmask	23.4%
Agent-Deepmask	39.2%
Regress-Deepmask	38.5%
Regress-STN-Deepmask	41.7%
Regress-Agent-Deepmask	55.4%
GT Bbox-Deepmask	62.6%

Table 1: Instance segmentation results on the MSCOCO subset D.

Results and analysis: The results for each baseline are reported in Table 1. As we can see, the "Agent-Deepmask" performs much better than the "Finetune-Deepmask" baseline and on par with the "Regress-Deepmask" baseline. This demonstrates the agent can learn a policy to narrow down the entire image to a smaller region where Deepmask can generate high activation. This has a similar effect to regressing a tight bounding box.

We also observe an improvement of the "Regress-Agent-Deepmask" over the "Agent-Deepmask" baseline. This is because "Regress-Agent-Deepmask" starts from the regressed bounding box, which has narrowed down the search space, so it's easier for the agent to find a better region than the "Agent-Deepmask" baseline.

Finally, we notice that "Regress-Agent-Deepmask" outperforms the "Regress-Deepmask" baseline by a large margin. The improvement is due to the fact that "Regress-Agent-Deepmask" can utilize the segmentation prediction to further refine the bounding box, while "Regress-Deepmask" cannot. Also, the "Regress-Agent-Deepmask" significantly outperforms "Regress-STN-Deepmask" baseline, suggesting that the advantage of RL-based method over the weak-supervised method. Some results of "Regress-Deepmask" and "Regress-Agent-Deepmask" are shown in Figure 3.

Implementation details. The action set $\mathcal{A}$ for this task changes the bounding box's location, scale and aspect ratio. The *Move* actions can shift the bbox center in four directions by $0.1 \times$ width/height. The *Scale* actions can scale up/down the width/height by a factor of 0.2. A *terminate* action is also included in the action set $\mathcal{A}$, thus $\mathcal{A}$ contains 9 actions. At intermediate steps, a reward of 0.1 is returned if the mask IoU is improved, otherwise, -0.1 is returned. At the last step, a reward of 1 is returned if the final mask IoU is greater than 0.5, otherwise, a reward of -1 is returned. The policy network is initialized with a Resnet-18 network[15] pretrained on ImageNet[10]. The last layer is replaced with a fully connected layer with 9 output units. The learning rate starts from 1e-4 and decreases by a factor of 5 whenever it reaches validation plateau.

4.2. Human Pose Estimation

We apply our algorithm to a challenging multi-person pose estimation dataset. In Section 4.2.1, we first introduce the dataset and briefly recap a two-stage framework named RMPE for multi-person pose estimation. Then, we conduct an experiment in Section 4.2.2 to demonstrate our argument that the finetuned model may not perform as well as the original model. In Section 4.2.3, we apply our framework to the multi-person pose estimation task and achieve state-of-the-art performance.

4.2.1 Dataset and baseline model

Dataset. MPII Human Pose[1] is a standard pose estimation benchmark that contains more than 28000 training samples for single person pose estimation and about 3800 training samples for multi-person pose estimation. Large variances of body shape, serious part occlusion and complex relationships among keypoints make it a very challenging task in computer vision.

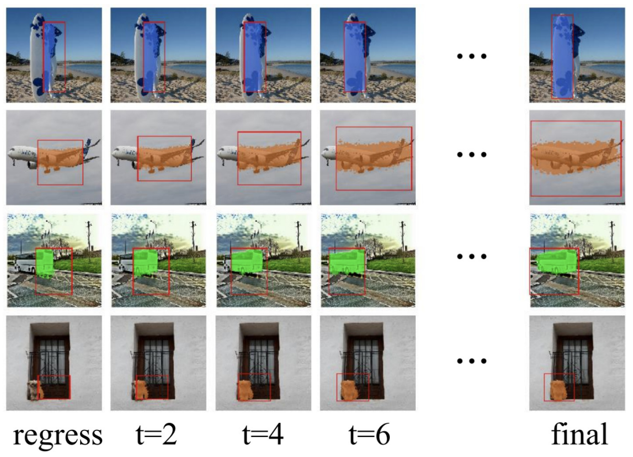


Figure 3: Results of "Regress-Deepmask" and "Regress-Agent-Deepmask". The first column is the output of "Regress-Deepmask". The second to the last columns depict the refinement process of "Regress-Agent-Deepmask". We can see the "Regress-Agent-Deepmask" gradually refines the imperfect bounding box to improve the mask prediction.

Baseline. Fang *et al.* [11] propose a two-stage framework named RMPE for multi-person pose estimation. The basic idea is to first apply an object detection algorithm to generate multiple human proposals and then apply single-person pose estimation algorithm on each proposal to generate keypoint detection. At last, a non-maximal suppression algorithm is performed to eliminate redundant pose estimations. Specifically, they adopt SSD [27] as their object detection algorithm and an 8-stacked hourglass model [32] as their pose estimation algorithm. SSD[27] is trained to generate tight bounding boxes. The 8-stacked hourglass model is pretrained on loose-bounding-box cropped images, then finetuned to fit tight-bounding-box cropped images.

4.2.2 Finetune v.s. Origin model

We demonstrate in this section that the finetuned model may not perform as well as the original model. Specifically, we compare the performance of hourglass model which is pretrained on loose-bbox cropped images then finetuned to tight-bbox cropped images.

Loose bbox. The annotation provided by MPII Human Pose [1] contains an item 'center' and an item 'scale' that indicate the location and size of a person. A square region centered in the location 'center' with length $(200 \times \text{scale})$

	Head	Shoulder	Elbow	Wrist	Hip	Knee	Ankle	Total
Iqbal&Gall [18]	58.4	53.9	44.5	35.0	42.2	36.7	31.1	43.1
Insafutdinov *et al.* [17]	78.4	72.5	60.2	51.0	57.2	52.0	45.4	59.5
Levinkov *et al.* [23]	89.8	85.2	71.8	59.6	71.1	63.0	53.5	70.6
Insafutdinov *et al.* [16]	88.8	87.0	75.9	64.9	74.2	68.8	60.5	74.3
Cao *et al.* [4]	91.2	87.6	77.7	66.8	75.4	68.9	61.7	75.6
Fang *et al.* [11]	88.4	86.5	78.6	70.4	74.4	73.0	65.8	76.7
Newell *et al.* [31]	**92.1**	**89.3**	78.9	69.8	**76.2**	71.6	64.7	77.5
ours	91.4	88.3	**79.8**	**71.7**	75.8	**75.2**	**67.2**	**78.5**

Table 2: Results on the MPII multi-person full test set (mAP).

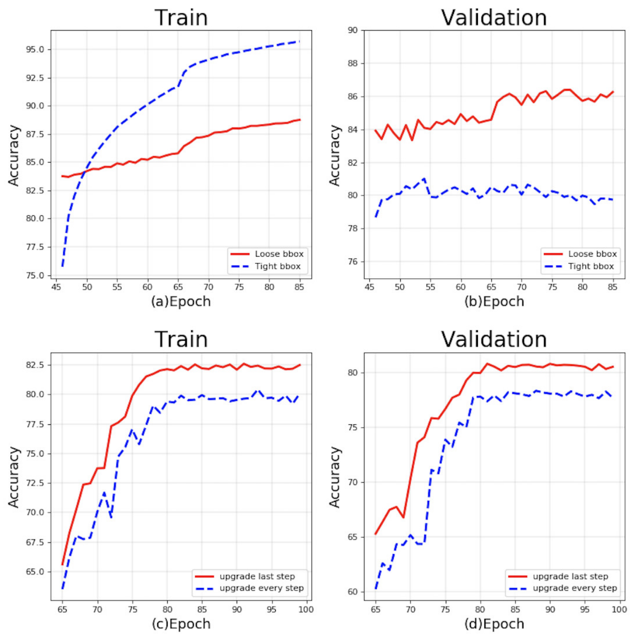


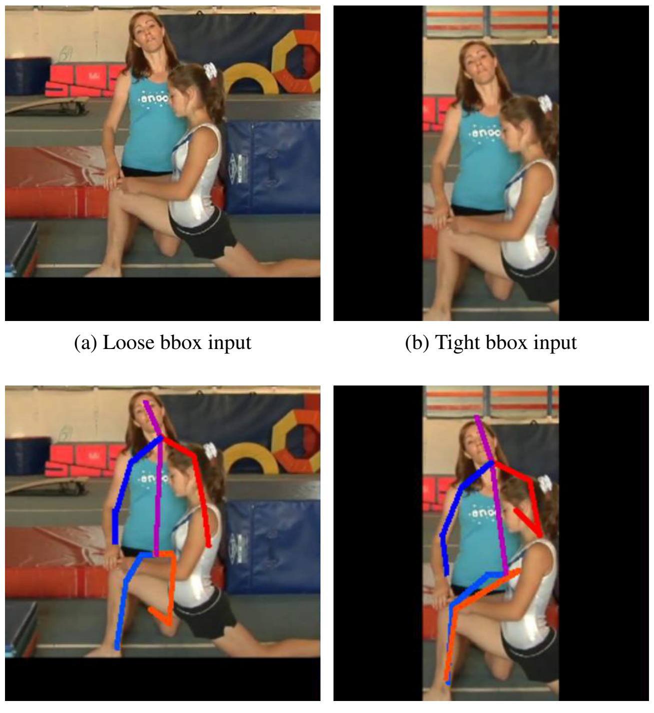
Figure 4: The first row are training and validation accuracy curves of "Tight bbox model" and "Loose bbox model". The second row are training and validation accuracy curves of upgrading the recognition network at each intermediate step and upgrading the recognition network at the last step.

pixels is cropped from the original image and used as the input for the network. See Figure 5a.

Tight bbox. Provided with the joints' locations, we can construct a tight bounding box by taking the minimal and maximal coordinates on the x and y axes of all the joints' coordinates. We further scale up the bounding box by a factor of 0.3 to make sure the entire human is included. This region is cropped from the original image and used as the input for the network. See Figure 5b.

Comparison. We separately train a two-stacked hourglass model on loose-bbox cropped images and tight-bbox cropped images. The training settings are the same (learning rate, train/valid split, etc.) The models are called "Loose bbox model" and "Tight bbox model", respectively.

The training and validation accuracy curves are shown in Figure 4a, Figure 4b . As we can see, the performance of "Tight bbox model" is lower than the "Loose bbox model". The reason is that in the loose bbox case, much contextual information is included and other people show up in the cropped area. This may help improve the robustness of the 'Loose bbox model'. In contrast, the training data for 'Tight bbox' contains less contextual information and only one person shows up in the cropped area. Therefore, the model gets confused when another person is also included in the image. See Figures 5c, 5d for comparison between the two predictions.



(a) Loose bbox input (b) Tight bbox input

(c) Loose bbox prediction (d) Tight bbox prediction

Figure 5: Different input format and prediction of pose estimation algorithms

4.2.3 Experiment setting and results

Since RMPE[11] has set up a general framework and achieves excellent performance, we adopt similar settings from their framework. Specifically, we maintain the same

Methods	Head	Shoulder	Elbow	Wrist	Hip	Knee	Ankle	Total
Ours, full	**87.3**	**85.6**	**80.5**	**71.2**	**75.4**	**70.8**	**62.3**	**76.2**
(a) w/o feedback	72.5	69.8	59.2	54.1	57.6	52.4	47.8	59.1
(b) w/o upgrading recognition	81.6	80.4	75.3	65.7	69.1	62.2	56.4	70.1
(c) back-propagate policy gradient	77.4	75.1	70.1	59.3	61.9	58.3	51.8	64.8

Table 3: Ablation studies on the MPII multi-person validation set (mAP). "w/o X" means without X in our framework.

object detection, pose estimation, and pose level NMS algorithms. Additionally, we insert an agent among the object detection and pose estimation algorithms. The agent's goal is to improve the pose estimation network's prediction by refining the bbox generated by the object detector. This new pipeline is then trained under our EU-RL framework.

Implementation detail. We use the same action set $\mathcal{A}$ as in Section 4.1, which contains 9 actions to change the bounding box. We initialize the policy network with a pre-trained Resnet-18 model [15] and change the number of output units of the last layer to 9. The value network shares most layers with the policy network, except in the last layer, where the number of output units is 1. The loss function for pose estimation is the mean-squared error. We give a positive reward of 0.1 when the agent improves the pose estimation at an intermediate step, and negative reward of -0.1 if the performance decreases. At the last step, a reward of 1 is returned if the final loss is less than 1e-3; otherwise, the reward is -1. The learning rate starts from 2.5e-4 and decreases by a factor of 5 whenever it reaches validation plateau.

Results. We report results on the test set in Table 2. As we can see, our method improves the RMPE model by 1.8 mAP, achieving state-of-the-art performance on MPII Human Pose dataset. We present some results in supplementary material.

5. Ablation studies

In this section, we study the influences of different components in our framework on the pose estimation task. All experiments are evaluated on our held out validation set. We use a two-stacked hourglass model as the pose estimation network.

We start by studying the influence of feeding the recognition network's output to the agent. We train an agent with only the RGB images as its input. The result is reported in Table 3(a). As we can see, removing the recognition network's output from the agent's input greatly decreases the overall performance. This demonstrates that the complementary nature of our state representation is essential in providing an accurate estimation.

We then study the influence of upgrading the recognition network. We train an agent without upgrading the recognition network and compare the overall performance to the environment-upgraded agent. The result is showed in Table 3(b). We can observe there is a decrease of performance for the non-upgraded agent, which verifies upgrading the environment helps the agent learn a better policy, as well as train the recognition network to perform better under that policy.

We also study the influence of back-propagating the policy gradient into the recognition network. Since the recognition network's output is fed into the policy network, we can back-propagate the policy gradient into the recognition network's output and jointly train the two networks. In fact, this setting results in a significant decrease of performance (Table 3(c)). We argue that the inconsistent goals of the two networks introduce the discrepancy: the policy network is for decision making while the recognition network is for perception. Another reason is that, compared to the recognition gradient generated by supervised learning, the policy gradient has much higher variance due to the trial-and-error nature in reinforcement learning. Back-propagating a high-variance gradient negatively affects the recognition network, and results in a decrease of performance.

At last, we test upgrading the environment at each intermediate step and report the training and validation accuracy curves in Figure 4c, Figure 4d. We notice that upgrading the environment at each intermediate step degrades the recognition network's performance by about 2 points. In this case, the recognition network's goal is to minimize the average loss through the entire sequence. This may differ from our original goal of improving the final performance, and so we suffer a loss of accuracy.

6. Conclusions and Future Work

In this paper, we proposed a novel environment upgrade reinforcement learning framework to re-link the separately trained stages in non-differentiable multi-stage pipelines. The framework utilizes an agent to feed back information from downstream to upstream. We upgrade the environment (downstream) to fit the agent's policy, which in term helps the agent learn a better policy. Extensive experiments verify the effectiveness of the proposed framework on both instance segmentation and pose estimation task. For the future work, since only two-stage framework is discussed, we will explore how to apply this framework into larger pipeline systems.

Acknowledgment: This work is supported in part by the National Natural Science Foundation of China under Grants 61772332 and in part by SenseTime Ltd.

References

[1] Mykhaylo Andriluka, Leonid Pishchulin, Peter Gehler, and Bernt Schiele. 2d human pose estimation: New benchmark and state of the art analysis. In *CVPR*, 2014. 6

[2] Adrian Bulat and Georgios Tzimiropoulos. Human pose estimation via convolutional part heatmap regression. In *European Conference on Computer Vision*, pages 717–732. Springer, 2016. 1, 2

[3] Juan C Caicedo and Svetlana Lazebnik. Active object localization with deep reinforcement learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2488–2496, 2015. 2

[4] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. *arXiv preprint arXiv:1611.08050*, 2016. 7

[5] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv preprint arXiv:1606.00915*, 2016. 1

[6] Yu Chen, Chunhua Shen, Xiu-Shen Wei, Lingqiao Liu, and Jian Yang. Adversarial posenet: A structure-aware convolutional network for human pose estimation. *arXiv preprint arXiv:1705.00389*, 2017. 2

[7] Hwann-Tzong Chen Chia-Jung Chou, Jui-Ting Chien. Self adversarial training for human pose estimation. *arXiv preprint arXiv: 1707.02439*, 2017. 2

[8] Xiao Chu, Wei Yang, Wanli Ouyang, Cheng Ma, Alan L Yuille, and Xiaogang Wang. Multi-context attention for human pose estimation. *arXiv preprint arXiv:1702.07432*, 2017. 1

[9] Jifeng Dai, Kaiming He, and Jian Sun. Instance-aware semantic segmentation via multi-task network cascades. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3150–3158, 2016. 1, 2

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009. 6

[11] Haoshu Fang, Shuqin Xie, and Cewu Lu. Rmpe: Regional multi-person pose estimation. *arXiv preprint arXiv:1612.00137*, 2016. 1, 6, 7

[12] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1440–1448, 2015. 1

[13] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014. 1

[14] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. *arXiv preprint arXiv:1703.06870*, 2017. 1, 2

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 6, 8

[16] Eldar Insafutdinov, Mykhaylo Andriluka, Leonid Pishchulin, Siyu Tang, Evgeny Levinkov, Bjoern Andres, Bernt Schiele, and Saarland Informatics Campus. Arttrack: Articulated multi-person tracking in the wild. In *Proc. of CVPR*, 2017. 7

[17] Eldar Insafutdinov, Leonid Pishchulin, Bjoern Andres, Mykhaylo Andriluka, and Bernt Schiele. Deepercut: A deeper, stronger, and faster multi-person pose estimation model. In *European Conference on Computer Vision*, pages 34–50. Springer, 2016. 1, 7

[18] Umar Iqbal and Juergen Gall. Multi-person pose estimation with local joint-to-person associations. In *European Conference on Computer Vision*, pages 627–642. Springer, 2016. 7

[19] Vijay R Konda and John N Tsitsiklis. Actor-critic algorithms. In *Advances in neural information processing systems*, pages 1008–1014, 2000. 4

[20] Tao Kong, Anbang Yao, Yurong Chen, and Fuchun Sun. Hypernet: Towards accurate region proposal generation and joint object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 845–853, 2016. 1

[21] Xiangyu Kong, Bo Xin, Yizhou Wang, and Gang Hua. Collaborative deep reinforcement learning for joint object search. *arXiv preprint arXiv:1702.05573*, 2017. 2

[22] Alexander Krull, Eric Brachmann, Sebastian Nowozin, Frank Michel, Jamie Shotton, and Carsten Rother. Poseagent: Budget-constrained 6d object pose estimation via reinforcement learning. *arXiv preprint arXiv:1612.03779*, 2016. 2, 3

[23] Evgeny Levinkov, Jonas Uhrig, Siyu Tang, Mohamed Omran, Eldar Insafutdinov, Alexander Kirillov, Carsten Rother, Thomas Brox, Bernt Schiele, and Bjoern Andres. Joint graph decomposition & node labeling: Problem, algorithms, applications. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 7

[24] Yi Li, Haozhi Qi, Jifeng Dai, Xiangyang Ji, and Yichen Wei. Fully convolutional instance-aware semantic segmentation. *arXiv preprint arXiv:1611.07709*, 2016. 1, 2

[25] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. *arXiv preprint arXiv:1612.03144*, 2016. 1

[26] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 5

[27] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016. 1, 6

[28] Stefan Mathe, Aleksis Pirinen, and Cristian Sminchisescu. Reinforcement learning for visual object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2894–2902, 2016. 2

[29] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pages 1928–1937, 2016. 2

[30] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015. 2

[31] Alejandro Newell and Jia Deng. Associative embedding: End-to-end learning for joint detection and grouping. *arXiv preprint arXiv:1611.05424*, 2016. 1, 7

[32] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *European Conference on Computer Vision*, pages 483–499. Springer, 2016. 1, 2, 6

[33] George Papandreou, Tyler Zhu, Nori Kanazawa, Alexander Toshev, Jonathan Tompson, Chris Bregler, and Kevin Murphy. Towards accurate multi-person pose estimation in the wild. *arXiv preprint arXiv:1701.01779*, 2017. 1

[34] Pedro O Pinheiro, Ronan Collobert, and Piotr Dollár. Learning to segment object candidates. In *Advances in Neural Information Processing Systems*, pages 1990–1998, 2015. 1, 2, 5

[35] Pedro O Pinheiro, Tsung-Yi Lin, Ronan Collobert, and Piotr Dollár. Learning to refine object segments. In *European Conference on Computer Vision*, pages 75–91. Springer, 2016. 1, 2

[36] Leonid Pishchulin, Mykhaylo Andriluka, Peter Gehler, and Bernt Schiele. Poselet conditioned pictorial structures. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 588–595, 2013. 1, 2

[37] Leonid Pishchulin, Eldar Insafutdinov, Siyu Tang, Bjoern Andres, Mykhaylo Andriluka, Peter V Gehler, and Bernt Schiele. Deepcut: Joint subset partition and labeling for multi person pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4929–4937, 2016. 1

[38] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788, 2016. 1

[39] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 1

[40] Zhou Ren, Xiaoyu Wang, Ning Zhang, Xutao Lv, and Li-Jia Li. Deep reinforcement learning-based image captioning with embedding reward. *arXiv preprint arXiv:1704.03899*, 2017. 2

[41] James Steven Supancic III and Deva Ramanan. Tracking as online decision-making: Learning a policy from streaming videos with reinforcement learning. *arXiv preprint arXiv:1707.04991*, 2017. 3

[42] Richard S Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3(1):9–44, 1988. 4

[43] Alexander Toshev and Christian Szegedy. Deeppose: Human pose estimation via deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1653–1660, 2014. 1, 2

[44] Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. Convolutional pose machines. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4724–4732, 2016. 1, 2

[45] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992. 4