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

Shuqin Xie¹, Zitian Chen²‡, Chao Xu¹, Cewu Lu¹*†
¹Shanghai Jiao Tong University, China ²Fudan University, China
¹{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 sin-

*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

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. $\Omega$ is the output of Stage 1. An agent observes the output of Stage 2 and choose an action to refine $\Omega$.

Figure 1: (a) Error case of 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. $\Omega$ is the output of Stage 1. An agent observes the output of Stage 2 and choose an action to refine $\Omega$.

gle person pose estimation algorithm [31] to each human instance, achieving state-of-the-art performance on multiple datasets.

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-
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 feedback 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 feedback 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 recognition algorithms 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 algorithm 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. 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.

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

![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 \( s^R_I \). The agent takes \((s^t_I, s^R_I)\) as input, samples an action \( a_t \) from policy \( \pi(a_t|s_t; \theta_p) \) to refine \( \Omega \). This changes the recognition network’s input at next step. This process is repeated until the agent decides to terminate the process.](image)

**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 \( \theta_r \) 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: \((s^t_I, s^R_I)\). \( s^t_I \) 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 \( \Omega \) from image \( I \). Then \( s^t_I \) is fed into the recognition network to generate a recognition output \( s^R_I = f(s^t_I; \theta_r) \). These two components are essential and complementary because \( s^R_I \) indicates how accurate the recognition output is, while \( s^t_I \) provides visual cues for the agent to tune \( \Omega \).

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

\[
A = \{ \text{move left, move right, move up, move down, scale up width, scale down width, scale up height, scale down height} \}
\]

\( A \) also includes another action \textit{terminate} that indicates the agent has finished the refinement process, leaving \( s^R_I \) 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 \( s^R_I \) and the ground truth label \( I_t \),
and formulate the reward function as

$$r_t = \begin{cases} \alpha \cdot \text{sign} [L(s^R_t, l_t) - L(s^R_{t+1}, l_{t+1})]; & a_t \neq \text{terminate} \\ \text{sign} [\epsilon - L(s^R_t, l_t)]; & a_t = \text{terminate} \end{cases}$$

(1)

where $\epsilon$ is a threshold and $\text{sign}[\cdot]$ returns 1 (-1) for positive (negative) value. Scale factor $\alpha$ 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 $\theta_p$, serving as the actor. The value network $V(s_t; \theta_v)$ approximates the total expected reward of state $s_t$ with parameters $\theta_v$, 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:

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

(2)

where $\gamma$ 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

1. Set $t \leftarrow 1$
2. Get initial RGB observation $s_t^I$
3. Get initial recognition prediction $s_t^R \leftarrow f(s_t^I; \theta_r)$
4. State $s_t \leftarrow (s_t^I, s_t^R)$
5. // Learning Agent
6. **repeat**
7. Sample $a_t$ from policy $\pi(a_t|s_t; \theta_p)$
8. Execute action $a_t$, get new RGB image observation $s_{t+1}^I$
9. Get new recognition prediction $s_{t+1}^R \leftarrow f(s_{t+1}^I; \theta_r)$
10. State $s_{t+1} \leftarrow (s_{t+1}^I, s_{t+1}^R)$
11. Receive reward $r_t$
12. // TD-learning
13. $E_t \leftarrow r_t + \gamma \cdot V(s_{t+1}; \theta_v) - V(s_t; \theta_v)$
14. $\theta_p \leftarrow \theta_p + \alpha \cdot \nabla_{\theta_p} \log \pi(a_t|s_t; \theta_p) \cdot E_t$
15. $\theta_v \leftarrow \theta_v + \alpha \cdot 2E_t \cdot \nabla_v V(s_t; \theta_v)$
16. $t \leftarrow t + 1$
17. **until** $t = t_{\max} + 1$ or $a_{t-1} = \text{terminate}$
18. // Upgrading Environment
19. $\theta_r \leftarrow \theta_r + \beta \cdot \frac{\partial L(s_{t-1}^R; l_{t-1})}{\partial \theta_r} \cdot \frac{\partial s_{t-1}^R}{\partial \theta_r}$
20. **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**

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

**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”. Lastly, 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”.

<table>
<thead>
<tr>
<th>Segmentation</th>
<th>Average IoU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finetune-Deepmask</td>
<td>23.4%</td>
</tr>
<tr>
<td>Agent-Deepmask</td>
<td>39.2%</td>
</tr>
<tr>
<td>Regress-Deepmask</td>
<td>38.5%</td>
</tr>
<tr>
<td>Regress-STN-Deepmask</td>
<td>41.7%</td>
</tr>
<tr>
<td>Regress-Agent-Deepmask</td>
<td>55.4%</td>
</tr>
<tr>
<td>GT Bbox-Deepmask</td>
<td>62.6%</td>
</tr>
</tbody>
</table>

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 $A$ 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 $A$, thus $A$ 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 pretrained on ImageNet. 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.

**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 vs. 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 $(200 \times scale)$...
<table>
<thead>
<tr>
<th></th>
<th>Head</th>
<th>Shoulder</th>
<th>Elbow</th>
<th>Wrist</th>
<th>Hip</th>
<th>Knee</th>
<th>Ankle</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iqbal&amp;Gall [18]</td>
<td>58.4</td>
<td>53.9</td>
<td>44.5</td>
<td>35.0</td>
<td>42.2</td>
<td>36.7</td>
<td>31.1</td>
<td>43.1</td>
</tr>
<tr>
<td>Insafutdinov et al. [17]</td>
<td>78.4</td>
<td>72.5</td>
<td>60.2</td>
<td>51.0</td>
<td>57.2</td>
<td>52.0</td>
<td>45.4</td>
<td>59.5</td>
</tr>
<tr>
<td>Levinkov et al. [23]</td>
<td>89.8</td>
<td>85.2</td>
<td>71.8</td>
<td>59.6</td>
<td>71.1</td>
<td>63.0</td>
<td>53.5</td>
<td>70.6</td>
</tr>
<tr>
<td>Insafutdinov et al. [16]</td>
<td>88.8</td>
<td>87.0</td>
<td>75.9</td>
<td>64.9</td>
<td>74.2</td>
<td>68.8</td>
<td>60.5</td>
<td>74.3</td>
</tr>
<tr>
<td>Cao et al. [4]</td>
<td>91.2</td>
<td>87.6</td>
<td>77.7</td>
<td>66.8</td>
<td>75.4</td>
<td>68.9</td>
<td>61.7</td>
<td>75.6</td>
</tr>
<tr>
<td>Fang et al. [11]</td>
<td>88.4</td>
<td>86.5</td>
<td>78.6</td>
<td>70.4</td>
<td>74.4</td>
<td>73.0</td>
<td>65.8</td>
<td>76.7</td>
</tr>
<tr>
<td>Levinkov et al. [23]</td>
<td>92.1</td>
<td>89.3</td>
<td>78.9</td>
<td>69.8</td>
<td>76.2</td>
<td>71.6</td>
<td>64.7</td>
<td>77.5</td>
</tr>
<tr>
<td>Cao et al. [4]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fang et al. [11]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Newell et al. [31]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ours</strong></td>
<td>91.4</td>
<td>88.3</td>
<td>79.8</td>
<td>71.7</td>
<td>75.8</td>
<td>75.2</td>
<td>67.2</td>
<td>78.5</td>
</tr>
</tbody>
</table>

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.

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

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

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 turn 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


