DeblurRL: Image Deblurring with Deep Reinforcement Learning

Jai Singhal(✉) and Pratik Narang(✉)

Department of CSIS, BITS Pilani, Pilani, India
{h20190021,pratik.narang}@pilani.bits-pilani.ac.in

Abstract. Removing non-uniform blur from an image is a challenging computer vision problem. Blur can be introduced in an image by various possible ways like camera shake, no proper focus, scene depth variation, etc. Each pixel can have a different level of blurriness, which needs to be removed at a pixel level. Deep Q-network was one of the first breakthroughs in the success of Deep Reinforcement Learning (DRL). However, the applications of DRL for image processing are still emerging. DRL allows the model to go straight from raw pixel input to action, so it can be extended to several image processing tasks such as removing blurriness from an image. In this paper, we have introduced the application of deep reinforcement learning with pixel-wise rewards in which each pixel belongs to a particular agent. The agents try to manipulate each pixel value by taking a sequence of appropriate action, so as to maximize the total rewards. The proposed method achieves competitive results in terms of state-of-the-art.

1 Introduction

It is prevalent to adopt image deblurring techniques to recover the images from the blurry images. Blur in an image can be obtained in several ways, it may be introduced due to movement or shake of the camera (called motion blur), or by camera focus (called focused blur). It has been observed that the common type of blur found in the image is mainly due to motion and focus blur which therefore degrades the quality of the image [7].

Learning to control the agents from high-resolution images, or any signal like audio and video is one of the challenges of reinforcement learning. But, recent advances in deep learning made it possible to extract the features from high-resolution images, which is a breakthrough in fields like image processing, computer vision, speech recognition, etc. These methods utilize the power of neurons that makes a giant multi-level neural network architecture. These network techniques can be integrated with reinforcement learning (RL).

As deep Q-network (DQN) [2,9] has been introduced, many algorithms pertaining to RL were proposed that could play the Atari console games the same way a human would and beating professional poker players in the game of heads

S. K. Singh et al. (Eds.): CVIP 2020, CCIS 1377, pp. 445–454, 2021.
https://doi.org/10.1007/978-981-16-1092-9_37

up no-limit Texas hold'em, etc. This has attracted researchers to focus on deep reinforcement learning. However, these methods cannot easily be applied in applications such as image-deblurring [7] where pixel-wise manipulations are required. To deal with this problem, we have proposed a multi-agent Reinforcement Learning approach where an agent is assigned to each pixel to learn the optimal behavior i.e. to maximize the average expected total rewards of all pixels and update the pixel value iteratively. It is computationally not feasible to apply the existing techniques in a naive manner since the number of agents evaluated is huge (e.g., 1 million agents for 1000×1000 pixel images). To tackle this challenge we use a fully convolution network (FCN) so that all the parameters are shared and learning can be performed efficiently.

In this paper, we propose a deep reinforcement learning-based approach for image deblurring. We propose a reward map convolution which proves to be an effective learning method, wherein each agent considers not only the future states of its pixel but also those of their neighboring pixels. For setting up the deep reinforcement learning, the set of possible actions for a particular application should be pre-defined; this makes the proposed method interpretable, by which actions applied by the agents can be observable. The agent picks the best sequence of actions determined by the rewards provided by the environment. Our experimental result shows that the trained agents achieve better performance when compared to other state-of-art blinded/non-blinded deconvolutional and kernel estimation based fully CNN based approaches.

2 Related Work

Previously, [1,4,10–12,14] have employed CNN and other deep learning techniques for de-blurring. Xu et al. [14] proposed a non-blind setting whereas Schuler et al. [11] proposed a blind setting for deconvolutional scheme neural networks. In [14], the generative forward model has been used, where they have estimated the kernel by combining the locally extracted features from the image. Now, this information is used to reduce the difficulty of the problem. Sun et al. [12] has used an effective CNN based non-uniform motion deblurring approach to estimate the probabilistic distribution of motion kernels at the patch level. Kim et al. [4] proposed an approach that approximated the blur kernel such that the locally linear motion and the latent image are jointly estimated. Nah et al. [10] proposed a multi-scale convolutional neural network with multi-scale loss function, which avoids problems such as kernel-based estimation. Kupyn et al.[6] presented an end-to-end learning model using conditional Ad-versarial Networks for Blind motion deblurring, which is also a kernel free based deblurring approach and shows good result on GoPro and Kohler dataset.

On the other hand, Ryosuke et al. [3] proposed a deep reinforcement technique in which they have worked at a pixel level and have experimented with various image processing tasks such as image denoising, image restoration, color enhancement, and image editing; and have shown better performance with other state-of-the-art methods. They have used the deep reinforcement and pixel-based

rewards technique which certain discrete sets of actions which therefore makes different from other deep learning techniques.

3 Reinforcement Learning Background

In this paper, we have considered settings of the standard reinforcement learning in which over a discrete number of time steps, the agent interacts with an environment(E). Agent receives a state s^{τ} at time step τ, then the agent chooses an action from a set of possible actions "A" according to its policy obtained (π), where π is a mapping from states s^{τ} to actions a^{τ}. In return, the agent receives the next state $s^{\tau+1}$ and receives a scalar reward r^{τ}. This process continues until the agent reaches a terminal state after which the process restarts.

The R^{τ} is equals to

$$\begin{aligned} R^{\tau} = r^{\tau} + \gamma r^{\tau+1} + \gamma^2 r^{\tau+2} + \gamma^3 r^{\tau+3} \\ + ... + \gamma^{n-1} r^{\tau+n-1} + \gamma^{n} V(s^{\tau+n}) \end{aligned} \tag{1}$$

R^{τ} is the total accumulated reward return at time step τ with discount factor $\gamma \in (0, 1]$. The main objective of an agent is to maximize the expected return from each state s^{τ}.

In extension to the standard reinforcement learning, we have introduced pixel level agent, where each agent's policy is denoted as $\pi_i(a_i^{\tau}|s_i^{\tau})$ for each pixel ranging from $i \in [1, n]$

A3C [8] is a actor-critic method, which uses policy and value network both. We have denoted the parameter of policy and value network as θ_p and θ_v respectively. Both network uses the current state s^{τ} as input. Value network gives the expected total rewards from state s^{τ} which is nothing but the value $V(s^{\tau})$ which shows the goodness of the current state.

The gradient for value network is calculated as follows:

$$d\theta_v = \nabla_{\theta_v}(R^{\tau} - V(s^{\tau})^2) \tag{2}$$

The policy network results in the policy $\pi(a^{\tau}—s^{\tau})$, and uses a soft-max layer at the end to output the action to be applied to the pixel.

$$A(a^{\tau}, s^{\tau}) = (R^{\tau} - V(s^{\tau})) \tag{3}$$

$A(a^{\tau}, s^{\tau})$ is called advantage, and $V(s^{\tau})$ is subtracted to reduce the variance of the gradient [8]. The gradient for policy network is calculated as:

$$d\theta_p = -\nabla_{\theta_p} log\pi(a^{\tau}|s^{\tau}) A(a^{\tau}, s^{\tau}) \tag{4}$$

Table 1. Table denoting the filters, its size, the dilation factor, and number of output channels, respectively. (Dil. denotes Dilated, and Conv denotes Convulation)

Common network			
Conv + ReLU	Dil. Conv + ReLU	Dil. Conv + ReLU	Dil. Conv + ReLU
3X3, 1, 64	3 × 3, 2, 64	3 × 3, 3, 64	3 × 3,4,64
Policy network			
Dil. Conv + ReLU	Dil. Conv + ReLU	ConvGRU	Conv + Softmax
3 × 3, 3, 64	3 × 3, 2, 64	3 × 3, 1, 64	3 × 3, 1, $\|\|A\|\|$
Value network			
Dil. Conv + ReLU 3 × 3, 3, 64	Dil. Conv + ReLU 3 × 3, 2, 64	Conv 3 × 3, 1, 1	

3.1 Reinforcement Learning with Pixel-Wise Rewards

The network discussed above is obtained by combining the policy and value network. The network is fully convolutional A3C, and its specification of shared, policy, and value network are shown in Table 1. This network architecture is inspired by [15].

The objective of the problem is to learn the optimal policies $\pi = (\pi_1, \ldots \pi_{\mathrm{N}})$ which can maximize the overall mean of the expected rewards at each and every pixel.

$$\pi^* = argmax E_\pi(\sum_{\tau=0}^{\infty} \gamma^\tau \bar{r}_{\mathrm{i}}^\tau) \tag{5}$$

$$\bar{r}^\tau = \frac{1}{N}\sum_{i=0}^{N}(r^\tau) \tag{6}$$

Here, $\bar{r}^t$ is the mean of each reward at i^{th} pixel $\mathrm{r}^\tau{}_{\mathrm{i}}$.

This is taken to observe that, training N networks is computationally not practical when the image size is very huge, i.e., the number of pixels is huge. To solve this issue, this paper proposed the usage of the FCN instead of N networks, this will help the GPU to parallelize the computation, which makes the training efficient. This technique also makes sure that N agents can share their parameters. To boost the overall performance, we have proposed a powerful learning method known as reward-map convolution.

The gradients can be denoted in matrix form as follows [3]:

$$d\theta_{\mathrm{v}} = \nabla_{\theta_{\mathrm{v}}} \frac{1}{N} J^T\{(R^\tau - V(s^\tau)) \odot (R^\tau - V(s^\tau))\}J \tag{7}$$

$$A(a^\tau, s^\tau) = R^\tau - V(s^\tau) \tag{8}$$

$$d\theta_{\mathrm{p}} = \nabla_{\theta_{\mathrm{p}}} \frac{1}{N} J^{\mathrm{T}}\{\log \pi(a^\tau|s^\tau) \odot A(a^\tau, s^\tau)\}J \tag{9}$$

where (i_x, i_y) are the elements of matrices $A(a^\tau, s^\tau)$ and $\pi(a^\tau|s^\tau)$ respectively. J is ones-vector where every element is one, and $\odot$ denotes element-wise multiplication [3].

$$d\omega = -\nabla_\omega \frac{1}{N} \sum_{i=1}^{N} log\pi(a_i^\tau|s_i^\tau)(R_i^\tau - V(s_i^\tau)) + \nabla_\omega \frac{1}{N} \sum_{i=1}^{N} (R_i^\tau - V(s_i^\tau))^2 \tag{10}$$

$$= -\nabla_\omega \frac{1}{N} J^{\mathrm{T}} \{\log \pi(a^\tau|s^\tau) \odot A(a^\tau, s^\tau)\} J + \nabla_\omega \frac{1}{N} J^{\mathrm{T}} \{\{(R^\tau - V(s^\tau)) \odot (R^\tau - V(s^\tau))\} J \tag{11}$$

The first term in Eq. 10 outputs a higher total expected reward. And Eq. (11) operates as a regularizer such that R_i is not deviated from the prediction $V(s_i^t)$ by the convolution [3].

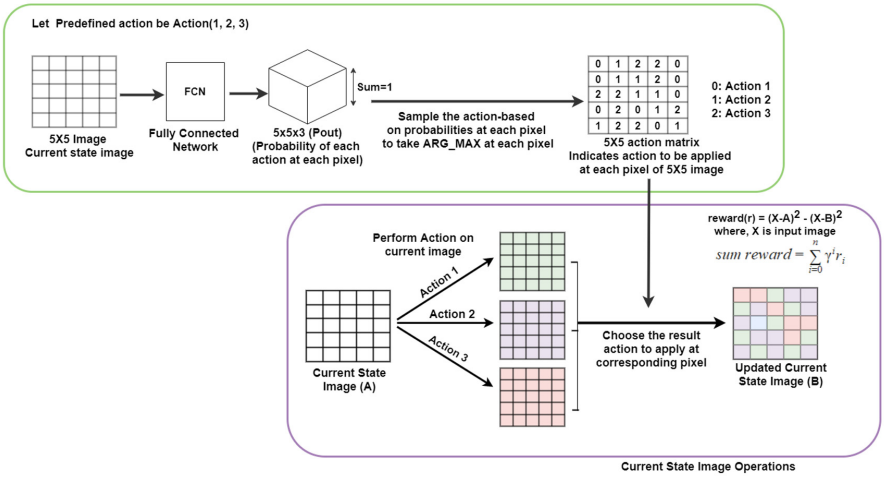

Fig. 1. Different actions (based on probability) applied on the pixels on the current image

3.2 Actions

The actions specified in Table 2 are applied depending on the pixel requirement.

Sharpening helps to sharpen the edges in the image. It increases the contrast between bright and dark regions which brings out the features in the given image. Blurriness causes the loss of the sharpness of most of the pixels.

Bilateral Filter is a non-linear, edge-preserving, smoothening, and noise removal filter from the image. It does so by replacing the intensity of each pixel with its weighted average of the neighboring pixels. It is applied as an action to smoothen the surroundings while preserving the edges of the image. It is applied as two actions with change in sigmaColor(σ_c) and sigmaSpace(σ_S). Sigma Color takes care of mixing the neighborhood color, whereas Sigma Space influences the farther pixels.

Table 2. Different actions applied to the pixel with its respective configurations and kernels.

Sno	Action	Kernel size	Filter/Conf
1	Sharpness	3 × 3	0 −1 0 −1 5 −1 0 −1 0
2	High pass filter	3 × 3	0 −5 0 −5 3 −5 0 −5 0
3	Low pass filter	3 × 3	1/9 1/9 1/9 1/9 1/9 1/9 1/9 1/9 1/9
4	Bilateral filter	–	$\sigma_c = 0.1$; $\sigma_s = 5.0$
5	Bilateral filter	–	$\sigma_c = 1.0$; $\sigma_s = 5.0$
6	Unsharp masking	–	Radius = 5, amount = 1
7	Unsharp masking	–	Radius = 5, amount = 2
8	Pix up	–	*= 1.05
9	Pix down	–	*= 0.95
10	No action	–	–

High Pass and Low Pass Filter are the frequency domain filter to smoothen and sharpen the image, by attenuating the particular(high/loss) component from the image.

- Low pass filter attenuates the high-frequency component, giving smoothness in the image, and also removes the noise from the image.

- A high pass filter attenuates the low-frequency component, giving sharpness to the image.

Unsharp masking is a linear image processing technique used to increase the sharpness in the image. The sharpness details are obtained by the difference

between the original and blurred images. The difference is calculated and added back to the original image.

$$enhanced\_img = img + amt * (img - blurred\_img) \tag{12}$$

Pix Up and Down helps to adjust the pixel level by increasing/decreasing the pixel value.

The following actions are subjected to each pixel on each state and try to maximize the total reward. Figure 1 shows how different actions are applied to the current state image. These actions are evaluated from the FCN [8] network. The actions are evaluated for each pixel value, and then applied to the current state image. After applying these actions to the pixels, total rewards are calculated of the current state, and checked for the reward, compared with the previous state, it will check the current state is how much better than the previous one.

Fig. 2. Qualitative comparison on GoPro dataset.

4 Experiments

In this paper, we have implemented the proposed method using Python3, chainerRL and Chainer [13] which are applied to the Deblurring application. We have experimented with two different sets of blurs, custom blurs using imgaug Motion Blur at a higher severity level and blurriness of GoPro dataset.

4.1 Input and State Actions

The input image is a blurred RGB image, the agents try to remove the blur from the photo, by applying several types of filters depending on the action required.

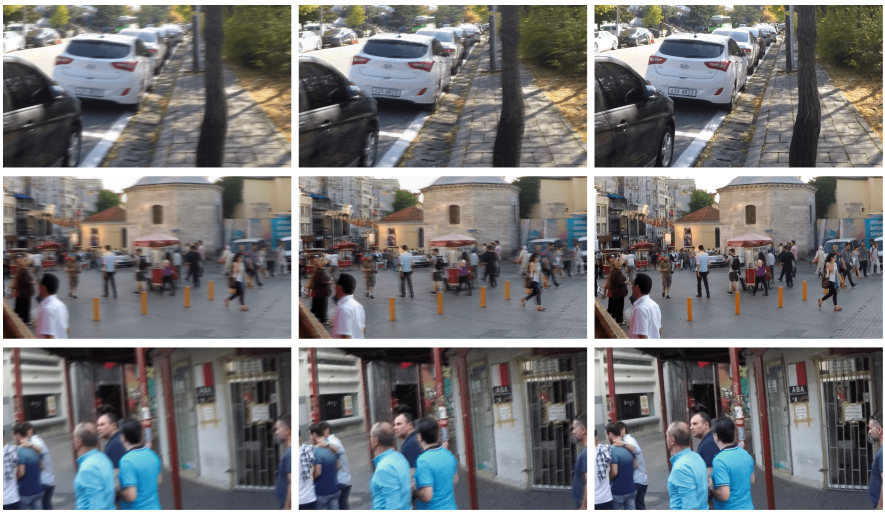

Fig. 3. Qualitative results after applying custom blur.

In Table 2, we have shown the various types of filters/actions applied to the input pixels which were empirically decided. One thing to note here is that we have only applied the classical image filtering techniques in our proposed method.

4.2 Implementation Details

We have used the GoPro dataset which has over 2103 different RGB images for training, and over 1111 RGB images for testing. The GoPro dataset for dynamic scene deblurring is publicly available[1] [10]. We set the mini-batch of random 50 images from the pool of training images with 70×70 random cropping. For the different experiments, we have added custom blur using imgaug blur and used Motion and Defocused blur at a severity level of 4.

For training, we have used Adam Optimizer [5], with the starting learning rate as 0.001. We have set the max episodes of 25,000, where the length of each episode is 5(t_max).

4.3 Results

We have implemented our model using Chainer and ChainerRL python library. All the experiments were performed on a workstation with Intel Xeon W-2123 CPU @3.6 GHz and NVIDIA Titan-V GPU.

We have evaluated the performance of our model for the GoPro dataset. We have tested the model for the 1111 test images available in the GoPro dataset

[1] https://seungjunnah.github.io/Datasets/gopro.

Table 3. Quantitative deblurring performance comparison on the GoPro dataset.

Measures	[12]	[4]	**DeblurRL**
SSIM	0.764	0.743	**0.763**
PSNR	31.573	31.965	**31.87**
Runtime	20 min	1 h	**5.5 s**

and compared the results with state-of-the-art methods of [4] and [12] in both qualitative and quantitative ways. In contrast, our results are free from kernel-estimation problems. Table 3, shows the PSNR (Peak signal-to-noise ratio) and SSIM (structural similarity index measure) scores. The SSIM score is perceptual metrics that quantify the degradation of the image after applying any image processing task. These are the average SSIM and PSNR score over testing all 1111 GoPro dataset test images.

The qualitative results obtained in the experiment are shown in Fig. 2. These results are compared with the results of [4] and [12]. Moreover, in Fig. 3, the qualitative results are shown for custom blur (using imgaug python library) which is compared with ground truth. The proposed approach is able to restore the blurred image close to the ground truth.

5 Conclusion

In this paper, we have proposed a new application of Deep reinforcement learning which operates the problem at a much granular level i.e., pixel-wise, and applies the given action at a pixel level. We have experimented with a technique to remove the blur from dynamic scene blurriness data-set (GoPro) from the given RGB image. Our experimental results show higher quantitative as well as qualitative performance when compared to other state-of-art methods of application.

This paper talks about how to maximize and focus on the pixel level. This paper also discusses how we can maximize each pixel reward, which makes our method different from other conventional convolutional neural networks based image processing methods. We believe that our method can be used for other image processing tasks where supervised learning can be difficult to apply.

References

1. Chakrabarti, A.: A neural approach to blind motion deblurring. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9907, pp. 221–235. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46487-9_14
2. François-Lavet, V., Henderson, P., Islam, R., Bellemare, M.G., Pineau, J.: An introduction to deep reinforcement learning. arXiv preprint arXiv:1811.12560 (2018)
3. Furuta, R., Inoue, N., Yamasaki, T.: Pixelrl: fully convolutional network with reinforcement learning for image processing. IEEE Trans. Multi. **22**(7), 1702–1719 (2019)

4. Hyun Kim, T., Mu Lee, K.: Segmentation-free dynamic scene deblurring. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2766–2773 (2014)
5. Kingma, D.P., Ba., J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
6. Kupyn, O., Budzan, V., Mykhailych, M., Mishkin, D., Matas, J.: Deblurgan: blind motion deblurring using conditional adversarial networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 8183–8192 (2018)
7. Li, D., Wu, H., Zhang, J., Huang., K.: A2-rl: aesthetics aware reinforcement learning for image cropping. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 8193–8201 (2018)
8. Mnih, V., et al.: Asynchronous methods for deep reinforcement learning. In: International Conference on Machine Learning, pp. 1928–1937 (2016)
9. Mnih, V., et al.: Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602 (2013)
10. Nah, S., Hyun Kim, T., Mu Lee, K.: Deep multi-scale convolutional neural network for dynamic scene deblurring. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3883–3891 (2017)
11. Schuler, C.J., Hirsch, M., Harmeling, S., Schölkopf, B.: Learning to deblur. IEEE Trans. Pattern Anal. Mach. Intell. **38**(7), 1439–1451 (2015)
12. Sun, J., Cao, W., Xu, Z., Ponce, J.: Learning a convolutional neural network for non-uniform motion blur removal. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 769–777 (2015)
13. Tokui, S., et al.: Chainer: a deep learning framework for accelerating the research cycle. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 2002–2011 (2019)
14. Xu, L., Jimmy, S.J., Ren, C.L., Jia, J.: Deep convolutional neural network for image deconvolution. Adv. Neural Inf. Process. Syst. **27**, 190–1798 (2014)
15. Zhang, K., Zuo, W., Gu, S., Zhang, L.: Learning deep CNN denoiser prior for image restoration. In: Proceedings of the IEEE Conference on Computer Vision Pattern Recognition, pp. 3929–3938 (2017)