Deep RL for Image Processing and Compressive Sensing

Gore Kao
Carnegie Mellon University
Pittsburgh, PA
gorek@andrew.cmu.edu

Abstract

Iterative as well unrolled learned methods have been applied to the domain of image processing and compressive sensing (CS). This work explores an alternative method to compressive sensing techniques for signal processing and reconstruction incorporating a deep RL framework. We maintain a deep network method, but with a mechanism to learn agents that can take actions for reconstruction. We first apply the framework to image denoising, a well knowing image processing topic and then try to apply the same principle to the compressive sensing problem. Using the established networks for image denoising, we add an extra step prior in order to map the compressed measurements to the space possible for image processing.

1 Introduction

Image processing include applications such as deblurring, inpainting, super resolution, denoising, etc. For simplicity, we first address the area of image denoising as a preliminary proof of concept. In image denoising a signal (or image) can be degraded by noise. The goal is therefore to remove the noise and recover the original signal as accurately as possible. This can be thought of as a simple formulation of $y = x + n$ where y is the observed noisy image, x is the original uncorrupted image, and n is additive white gaussian noise (AWGN) in this case. The generalized objective is then to find the minimized solution to the following equation.

$$\hat{x} = \arg\min_{x} \|y - x\|_2^2 + \lambda R(x) \tag{1}$$

Here $R(x)$ is a regularization term with regularization parameter λ. We follow a deep reinforcement learning (RL) method to address the denoising problem.

Compressive Sensing (CS) aims to reconstruct a signal by solving an underdetermined system of linear equations posed as $y = Ax$ where $x \in \mathbb{R}^N$ is the original signal of interest, $y \in \mathbb{R}^M$ is the observed signal, and $A \in \mathbb{R}^{M \times N}$ is the measurement operator. Traditional methods solve this formulation under a sparsity constraint of the signal x and utilize this prior to recover the original signal from far fewer samples required by the Nyquist–Shannon sampling theorem. Since we begin with an observation y in the space of $\mathbb{R}^M$, we construct a linear mapping matrix to map y to the space of $x \in \mathbb{R}^N$ which can then follow the same methodology used for image denoising (further described in Methods section). This work explores if a deep RL method utilizing a network and learned agents can also be used to solve a typical compressive sensing problem.

2 Related Work

There have been long standing classical methods to address image processing tasks such as image denoising. These methods include, but are not limited to, spatial domain filtering, variational denoising, and thresholding techniques (iterative soft/hard thresholding) [1]. More recently, other methods also include deep networks to learn the parameters for reconstruction in a supervised manner. One example includes work done by Mao et al. to utilize a deep fully convolutional auto-encoder network for image restoration, learning end-to-end mappings from corrupted images to the original ones [2]. Another previous work highlights using a deep image prior (Ulyanov et al.) through a generator network that maps a latent code z to the original image x [3].

Previous work by Furuta et al. tackles image processing tasks using reinforcement learning with pixel-wise rewards (**PixelRL**) through a Fully Convolutional Network (FCN) [4]. This deep RL method utilizes an agent for each pixel in the image and changes the value of the pixel by taking actions. The network consists of a shared network along with two separate FCN heads for the policy network and value network. In this way, the optimal policy can be learned through updating the network parameters and allows for pixel-wise manipulation in image processing tasks such as image denoising, restoration, local color enhancing, or saliency-driven image editing.

Zhang et al. propose a novel structured deep network, **ISTA-Net** [5], inspired by the Iterative Shrinkage-Thresholding Algorithm (ISTA) [6]. The authors cast ISTA into a deep convolutional network and solve the proximal mapping associated with the sparsity-inducing regularizer using nonlinear transforms. Using ISTA-Net, the parameters for reconstruction can be learned rather than handcrafted. The results show ISTA-Net can outperform state-of-the art optimization and network based compressive sensing methods. Notably, ISTA-Net also proposes a linear mapping matrix, $\mathbf{Q_{init}}$, in order to map the vector y to the space of x.

3 Method

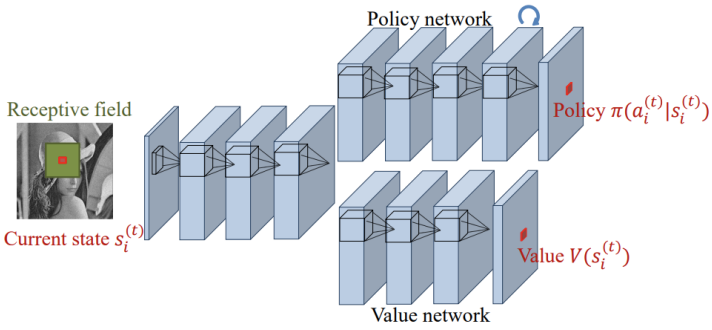
3.1 Formulation

We first consider the task of image denoising through a reinforcement learning formulation. On a high level, each pixel is in the image is treated as an agent that can take certain actions. Therefore this can be thought of in Multi-Agent Reinforcement Learning (MARL) setting [7]. Let I_i be the i-th pixel/agent in the input image that has N pixels and $s_i^{(t)}$ denotes the state of the i-th agent at time step t. Then the agent at each pixel has a policy denoted $\pi(a_i^{(t)}|s_i^{(t)})$ where $a_i^{(t)} \in \mathcal{A}$ (action space). We also define the reward function as follows in equation 2.

$$r_i^{(t)} = \left(I_i^{target} - s_i^{(t)}\right)^2 - \left(I_i^{target} - s_i^{(t+1)}\right)^2 \tag{2}$$

Agents obtain next states $\mathbf{s}_i^{(t+1)} = (s_i^{(t+1)}, \cdots, s_N^{(t+1)})$ and rewards $\mathbf{r}_i^{(t)} = (r_i^{(t)}, \cdots, r_N^{(t)})$ from taking actions $\mathbf{a}_i^{(t)} = (a_i^{(t)}, \cdots, a_N^{(t)})$ respectively. The objective is then to learn optimal policies $\pi = (\pi_1, \cdots, \pi_N)$.

Utilizing methods from both PixelRL and ISTA-Net [4, 5], we utilize a deep network along with an advantage actor critic method (A2C) similar to that used by Mnih et al [8]. We maintain a policy π and value function V for each agent through a fully convolutional network (FCN) with network heads for the policy network and value network respectively. The network then learns the parameters to determine the policy taken for each pixel in reconstruction. The architecture and configuration is shown in Figure 1.



Shared network				Policy network			
Conv +ReLU	Conv +ReLU	Conv +ReLU	Conv +ReLU	Conv +ReLU	Conv +ReLU	ConvGRU	Conv +Softmax
3x3, 1, 64	3x3, 2, 64	3x3, 3, 64	3x3, 4, 64	3x3, 3, 64	3x3, 2, 64	3x3, 1, 64	3x3, 1, $\|A\|$

Value network		
Conv +ReLU	Conv +ReLU	Conv
3x3, 3, 64	3x3, 2, 64	3x3, 1, 1

Figure 1: Architecture of Fully Convolutional Network from PixelRL [4]. Table denotes filter size, dilation size, and output filters respectively.

3.2 Reward Map Convolution

We follow the same added property utilized in PixelRL [4] which is the Reward Map denoted as $\mathbf{R}_i^{(t)}$. To understand the intuition for the reward map convolution, first take the scenario where each agent's actions are independent of each other. In this case we consider 1×1 receptive fields of the FCN. When each agent's action is independent, the reward map and derivative of the value network can be calculated from equations 3 and 4 respectively.

$$R_i^{(t)} = r_i^{(t)} + \gamma V(s_i^{(t+1)}) \tag{3}$$

$$d\theta_v = \nabla_{\theta_v} \frac{1}{N} \sum_{i=1}^{N} \left(R_i^{(t)} - V(s_i^{(t)}) \right)^2 \tag{4}$$

In addition, we follow the same definition for the advantage function used in the original work as well as the derivative of the policy network with respect to the network parameters shown in equations 5 and 6.

$$A(a_i^{(t)}, s_i^{(t)}) = R_i^{(t)} - V(s_i^{(t)}) \tag{5}$$

$$d\theta_p = -\nabla_{\theta_p} \frac{1}{N} \sum_{i=1}^{N} \log \pi(a_i^{(t)}|s_i^{(t)}) A(a_i^{(t)}, s_i^{(t)}) \tag{6}$$

The reward map formulation in equation 3 considers each agents' action as independent, however in this multi-agent RL setting, we should consider actions as dependent on neighboring agents. Therefore the formulation is adjusted to enlarge the receptive field of the reward map. Action $a_i^{(t)}$ affects not only state $s_i^{(t+1)}$ but also policy and values in $\mathcal{N}(i)$ (local window centered at i-th pixel). Following this we convert equation 3 to equation 7 with receptive field $\mathcal{N}(i)$.

$$R_i^{(t)} = r_i^{(t)} + \gamma \sum_{j \in \mathcal{N}(i)} w_{i-j} V(s_i^{(t+1)}) \tag{7}$$

Notice the summation in equation 7 can be neatly expressed as convolution, hence a Reward Map Convolution is taken [4].

3.3 Compressive Sensing Map

The formulation described in section 3.1 applies to the image denoising problem which takes as input an image with N pixels. For compressive sensing, we only observe a measurement $y \in \mathbb{R}^M$. Therefore to be able to apply the same formulation to the CS problem, we must first map y such that $\mathbb{R}^M \rightarrow \mathbb{R}^N$ to match the same dimension used in that for image denoising.

We follow the linear mapping formulation given in ISTA-Net [5]. Given training data pairs for the original image and its CS measurement, we have a dataset $\{\mathbf{y}_i, \mathbf{x}_i\}_{i=1}^{N_b}$ for N_b data pairs, $\mathbf{x}_i \in \mathbb{R}^N$, and $\mathbf{y}_i \in \mathbb{R}^M$. We can compute the linear mapping matrix, $\mathbf{Q}_{init}$, by solving a least squares problem which has a closed form solution shown in equation 8.

$$\mathbf{Q}_{init} = \arg\min_{\mathbf{Q}} \|\mathbf{Q}\mathbf{Y} - \mathbf{X}\|^2 = \mathbf{X}\mathbf{Y}^\top \left(\mathbf{Y}\mathbf{Y}^\top\right)^{-1} \tag{8}$$

where $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_{N_b}]$ and $\mathbf{Y} = [\mathbf{y}_1, ..., \mathbf{y}_{N_b}]$. Therefore input to the FCN can be computed as $\mathbf{x}^{(0)} = \mathbf{Q}_{init}\mathbf{y}$. The following methodology and training procedure then follows that described in section 3.1 and 3.5.

3.4 Action Space

We define the possible set of actions in the action space, $\mathcal{A}$ that the pixel agents can possibly take as follows.

1. Box Filter: kernel size = 5×5
2. Bilateral Filter: kernel size = 5×5, standard deviation in color space = 1.0
3. Bilateral Filter: kernel size = 5×5, standard deviation in color space = 0.1
4. Median Filter: kernel size = 5×5
5. Gaussian Filter: kernel size = 5×5, standard deviation = 1.5
6. Gaussian Filter: kernel size = 5×5, standard deviation = 0.5
7. Change Pixel Value: $+1$
8. Change Pixel Value: -1
9. Do Nothing

3.5 Training and Implementation

The dataset used is the Waterloo exploration database (4,774 images) taking 64×64 grayscale crops. For the denoising problem, additive white gaussian noise (AWGN) is added at a standard deviation of 25 (pixel values $0 - 255$). For the compressive sensing problem, we first construct a discrete fourier transform matrix (DFT) of size $N = 64 \times 64 = 4096$ and then randomly sample $M = 819$ (20% of N) rows to generate the measurement matrix A. We follow the A3C algorithm [8], adapting it to the same version used in PixelRL [4]. However, we set the number of threads to 1 which is equivalent to A2C.

We use an initial learning rate of 0.001 that is multiplied by $(1 - \frac{episode}{max\_episode})^{0.9}$ (poly learning rate). Batch size is set to 64 and the ADAM optimizer [9] is used. Maximum number of episodes is $20,000$ with $t_{max} = 5$ (length of each episode), which results in global counter $T_{max} = 100,000$. The reward map filter size is set to $\omega = 33 \times 33$. Implementation is done in PyTorch and is publicly available [1].

[1] `https://github.com/Gkao03/CS-Deep-RL`

4 Results

4.1 Image Denoising

Figure 2: Example 1 - Original, Noisy, Reconstructed

Figure 3: Example 2 - Original, Noisy, Reconstructed

The results of the denoising problem are not great and generally look like a blurred version of the original input, but it does do a decent job of filtering out some of the high frequency components.

4.2 Compressive Sensing

Following the mapping projection defined in section 3.3, the input $x^{(0)}$ follows the same algorithm used for the denoising problem. However, the experiments for the CS problem did not produce meaningful reconstruction results. A reason for failure is suspected to relate to how well the matrix, $\mathbf{Q}_{init}$, does in mapping y to the space of x such that it is close enough for taken actions to be meaningful. If the initialization is too far from the true result, the agents may not be able to take useful actions towards a good reconstruction. In addition on closer inspection, it is revealed that the singular values of $\mathbf{Y}\mathbf{Y}^\top$ are very small which leads to $\mathbf{Q}_{init}$ having very large values. This means that $\mathbf{Y}$ is too similar in some principal components (low variance).

5 Future Work

To improve the results of the compressive sensing problem, future work can consider a different intialization strategy to map y to x. In addition, the original action space used is more suited for the image denoising problem and thus the action space should be changed to better suit the compressed measurements. Another idea involves incorporating a gradient step in the algorithm similar to thresholding techniques like iterative soft thresholding.

References

[1] L. Fan, F. Zhang, H. Fan, and C. Zhang, "Brief review of image denoising techniques," *Visual Computing for Industry, Biomedicine, and Art*, vol. 2, no. 1, pp. 1–12, 2019.

[2] X.-J. Mao, C. Shen, and Y.-B. Yang, "Image restoration using convolutional auto-encoders with symmetric skip connections," *arXiv preprint arXiv:1606.08921*, 2016.

[3] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Deep image prior," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 9446–9454, 2018.

[4] R. Furuta, N. Inoue, and T. Yamasaki, "Pixelrl: Fully convolutional network with reinforcement learning for image processing," *IEEE Transactions on Multimedia*, vol. 22, no. 7, pp. 1704–1719, 2019.

[5] J. Zhang and B. Ghanem, "Ista-net: Interpretable optimization-inspired deep network for image compressive sensing," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1828–1837, 2018.

[6] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM journal on imaging sciences*, vol. 2, no. 1, pp. 183–202, 2009.

[7] K. Zhang, Z. Yang, and T. Başar, "Multi-agent reinforcement learning: A selective overview of theories and algorithms," *Handbook of reinforcement learning and control*, pp. 321–384, 2021.

[8] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*, pp. 1928–1937, PMLR, 2016.

[9] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.