Multi-Step Reinforcement Learning for Single Image Super-Resolution

Kyle Vassilo, Cory Heatwole, Tarek Taha
University of Dayton
Dayton, OH
{vassilok1, heatwolec1, ttaha1}@udayton.edu

Asif Mehmood
Air Force Research Laboratory
Wright-Patterson AFB, OH
asif.mehmood.1@us.af.mil

Abstract

Deep Learning (DL) has become prevalent in today's image processing research due to its power and versatility. It has dominated the Single Image Super-Resolution (SISR) field with its ability to obtain High-Resolution (HR) images from their Low-Resolution (LR) counterparts, particularly using Generative Adversarial Networks (GANs). Interest in SISR comes from its potential to increase the performance of supplementary image processing tasks such as object detection, localization, and classification. This research applies a multi-agent Reinforcement Learning (RL) algorithm to SISR, creating an advanced ensemble approach for combining powerful GANs. In our implementation each agent chooses a particular action from a fixed action set comprised of results from existing GAN SISR algorithms to update its pixel values. The pixel-wise or patch-wise arrangement of agents and rewards encourages the algorithm to learn a strategy to increase the resolution of an image by choosing the best pixel values from each option.

1. INTRODUCTION

Single Image Super-Resolution (SISR) is a vague problem that poses challenges in many computer vision applications. In this expanding problem we attempt to reproduce High-Resolution (HR) images from their Low-Resolution (LR) counterparts. In past years, SISR research revolved around Deep Learning (DL) algorithms that attempted to directly map LR input images to HR output images with a single pass through the network. Hundreds of unique Generative Adversarial Networks (GANs) have been applied to SISR that differ in their architectures, loss functions, and up-sampling techniques. They all share underlying fundamentals such as adversarial learning, Convolutional Neural Networks (CNNs), and attempting to mimic a certain distribution.

Instead of mapping LR images to their HR counterparts or imitating a distribution, our method adopts a strategy to increase the resolution of an image by selecting a certain action for each pixel or patch of pixels, based on pixelRL [4]. PixelRL is a multi-agent Reinforcement Learning (RL) algorithm where each pixel is modified by an independent agent. Agents choose particular actions to update their own pixel values. Being a RL technique, pixelRL permits the ability to analyze the output of the network through multiple iterations, referred to as timesteps. At each timestep the agent can choose to change its decision or stay the same. Furuta et al. applied this algorithm to three image processing tasks: denoising, restoration, and local color enhancement [4]. In our research we explore the possibility of applying and modifying pixelRL for the SISR problem.

The contributions of this work are as follows:

- Proposing a SISR RL network, which employs multiple agents. The network is given a bicubicly upsampled image and learns a policy for each pixel in order to maximize the total reward.
- Proposing an agent-per-patch configuration, where agents replace square patches of pixels rather than individual pixels. We attempt to minimize a "noisy" effect left behind from the agent-per-pixel configuration.
- Proposing a novel action set consisting of a number of GAN outputs, essentially creating an advanced ensemble approach to SISR.
- Proposing an evaluation method to determine how well agents perform compared with the naive method of simply recreating the image from the overall "best" option ("best" from an image level perspective).

Other SISR works, pixelRL, and some of its uses are described in Section 2. Our proposed method is described in Section 3 and applications and results of the method are presented in Section 4. Finally, the paper is concluded in Section 5.

2. RELATED WORKS

GANs were first introduced by Ian Goodfellow et al., in 2014, and were applied to small grayscale and color images [6]. Consequently, GANs have become an extremely

hot topic in DL research due to their impressive results and wide application range including: image editing [16] and texture removal (such as rain [25]) , stenographic security [18], data generation for other DL applications [20], attention prediction [15], and more. In 2016, Christian Ledig et al. proposed the SRGAN algorithm where they applied GANs to SISR [9]. Their generator consisted of 16 fully-convolutional residual blocks, ending with two PixelShufflers to increase the image dimension by a factor of 4. Their discriminator network consisted of 8 fully-convolutional blocks with batch normalization (in blocks 2-8) and LeakyReLU activations, followed by dense layers to produce the final real/fake prediction. To create more perceptually plausible images, the standard mean squared error (MSE) loss was replaced with a VGG loss, based on feature map activations from particular layers of the 19-layer VGG network, and an adversarial loss. Although they did not achieve state-of-the-art PSNR or SSIM results, they set a new standard for photo-realistic quality by achieving the highest mean opinion score (MOS).

In the last few years, a number of image processing problems have been tackled with deep RL. Zhuopeng Li and Xiaoyan Zhang developed an image cropping algorithm which utilized collaborative deep RL [11]. Optimal cropping actions were chosen based on information from two different agents. Both agents were given a vector of all previous actions and an image. The first agent's image was the current crop window of the original image while the second agent's image was the current crop window of the emotional attention map associated with the original image. On two different datasets (FCD and HCD), Li and Zhang achieved state-of-the-art results. QingXing Cao et al. used deep RL for face hallucination, a specific case of SISR that applies only to face images [2]. A single agent determined the best patch of an image to enhance, via a custom CNN. The enhanced patch was then inserted back into the LR image and the agent selected a new patch. The process repeated until the entire image was enhanced. Their results showed that the order of patch enhancement affected the quality of the final image.

Ryosuke Furuta et al. used a novel multi-agent deep RL algorithm for image denoising, restoration, and color enhancement in which each pixel had its own agent, called pixelRL [4]. The action set for the denoising and restoration tasks included a number of standard image processing filters, pixel value increment and decrement, and "do nothing". For color enhancement the action set included adjusting contrast, brightness, saturation, color balance, and "do nothing". All tasks used the same CNN architecture, shown in Figure 1. Rewards were given to agents based on the MSE difference between the current image (state) and the target image and the previous image (state) and the target image (analogous to the L_2 loss). Their denoising results (PSNR) were state-of-the-art for salt-and-pepper noise with noise densities of 0.5 and 0.9. Their restoration results (PSNR and SSIM) were also state-of-the-art.

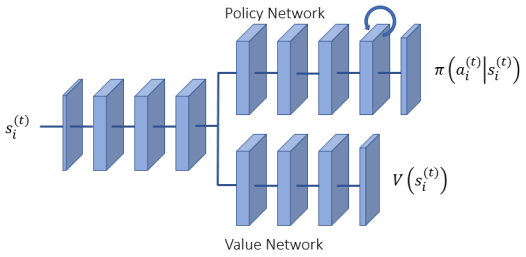


Figure 1. pixelRL FCN architecture [4].

As of this writing, pixelRL has been adapted to two additional problem settings: MRI reconstruction [10] and 3D medical image segmentation [12]. According to Wentian Li et al., the key benefit of applying pixelRL to MRI reconstruction is its interpretability. Typical DL methods create a complex mapping from corrupted images to reconstructed images, making it nearly impossible for people to understand exactly *how* the image is reconstructed. In contrast, pixelRL uses a well-defined action set, allowing people to see exactly how each pixel of the image has been modified from corrupted to reconstructed. Similar to the original pixelRL, Li et al. used standard filters in their action set, along with "do nothing" and a pixel value decrement. They included a number of sharpening filters (Laplace, Sobel, and unsharp mask) with learnable continuous parameters. The original pixelRL network architecture, shown in Figure 1, was modified to accommodate these parameters by adding a third branch to the network after the split. Rewards were given based on the absolute error between the original image (state) and the target image and the final image (state) and the target image (analogous to the L_1 loss). This method achieved state-of-the-art results for NMSE and SSIM on the fastMRI dataset with a random 40% mask [10].

Xuan Liao et al. adapted pixelRL to 3D medical image segmentation by using voxel-wise agents and rewards. They also included user interaction to aid in the segmentation process, where users provide hints (such as points or bounding boxes) to the model. The network architecture used was nearly identical to that in pixelRL, shown in Figure 1, with the main difference being that the input to the final layer in the policy and value branches was a concatenation of all previous layers' outputs. Inputs to the network included a 3D image, the previous segmentation probabilities, and a user hint map. The new segmentation probabilities were created by tweaking the previous ones. As such, the action set consisted of various values that agents could use to modify the previous probabilities. With an output consisting of probabilities, a cross-entropy gain-based function was used to calculate the rewards based on the improvement from the previous output

to the new output. This method achieved state-of-the-art results across three different datasets (BRATS2015, MM-WHS, AND NCI-ICBI2013) [12].

3. PROPOSED METHOD

PixelRL utilizes the asynchronous advantage actor-critic (A3C) algorithm [14]. A3C is a deep RL algorithm that uses multiple network instances that each have their own parameters and copy of the environment. This allows each instance to train in parallel and contribute to the shared learning of the global network. Actor-critic methods utilize two networks, a policy network and a value network, that simultaneously operate on the current state, $s^{(t)}$. The value network, known as the critic, outputs the expected total rewards, $V(s^{(t)})$, from the current state, which exhibits the quality of the current state [4]. The policy network, known as the actor, calculates the probabilities of the agent choosing each action, $\pi(a^{(t)}|s^{(t)})$, when in the current state [4].

PixelRL employs a Fully Convolutional Network (FCN) which rearranges the agents into a 2D space where they can share parameters [4]. This 2D representation facilitates the reward map convolution learning method proposed by Furuta et al [4]. In this method, the receptive field of an agent is treated as a weighted convolutional filter that influences the policies and values of neighboring agents. Figure 1 illustrates the pixelRL network architecture which applies a Gated Recurrent Unit (GRU) to the penultimate layer of the policy network. Cho et al. introduced the GRU in 2014 to implement a layer with memory that does not suffer from the vanishing gradient problem [3]. The last layer of the policy network produces a feature map for each action. A softmax function is applied across each of the feature maps to provide a probability distribution over the potential actions for each agent [4]. During training, the distribution is stochastically sampled to extract a single action for each pixel, while during testing the network always chooses the most probable action.

The pixelRL policy differs from traditional RL policies in that each pixel has an agent. Thus, the policy becomes $\pi_i(a_i^{(t)}|s_i^{(t)})$, where $a_i^{(t)}$ and $s_i^{(t)}$ are the action and state at timestep t of the i-th agent [4]. The number of actions come from the action set A predefined by the authors. In our case the action set consists of pixel value increment and decrement, "do nothing", and choosing pixel values from images upsampled by numerous GAN SISR algorithms. In the current implementation, the increment and decrement actions are applied to all channels of a pixel. PixelRL applied the filters during execution, however they could have used pre-filtered images, which inspired our actions. This idea is a novel contribution as we change the action set to a number of pre-computed GAN outputs. These actions are illustrated in Figure 2, where the colors will be used to create action maps for each timestep in subsequent figures (see Figure 3 for an example). Selecting the GAN algorithms began with a survey of current SISR methods. Their results were compared and ranked according to PSNR and SSIM values. The GANs selected for our action set [8, 13, 22] had the highest values amongst all of the algorithms with publicly available code.

	Action	Color
1	pixel value -= 1	
2	do nothing	
3	pixel value += 1	
4	edsr baseline	
5	esrgan	
6	esrgan-pnsr	
7	ppon	

Figure 2. SISR action set: 1. Subtract 1 from the value of all channels of the pixel, 2. Do not change the pixel value, 3. Add 1 to the value of all channels of the pixel, 4. Substitute pixel value with that of the Enhanced Deep Super-Resolution network (EDSR), 5. Substitute pixel value with that of the Enhanced Super-Resolution GAN (ESRGAN), 6. Substitute pixel value with that of the Enhanced Super-Resolution GAN, PSNR focused (ESRGAN-PSNR), 7. Substitute pixel value with that of the Progressive Perception-Oriented Network (PPON).

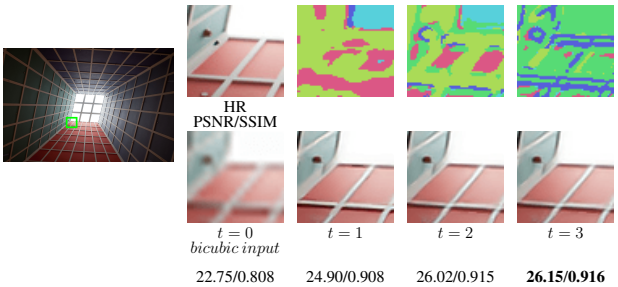


Figure 3. Super Resolution process and action map at each timestep for Urban100 image img_090 (RGB color space, pixel-wise agents, MSE-based rewards). The colors represent the agent's choice of options from Figure 2, essentially replacing the original image pixel with a different option.

Here the agent transitions from state to state by choosing an action and acquiring a reward to assist in learning an efficient policy, $\pi = [\pi_1, ..., \pi_N]$, where N is the total number of pixels/agents in an image [4]. The reward function compares the output and previous images at each timestep with the target image, described as:

$$r_i^{(t)} = (I_i^{target} - y_i^{(t-1)})^2 - (I_i^{target} - y_i^{(t)})^2, \quad (1)$$

where $r_i^{(t)}$ is the reward for each pixel at a given timestep, I_i^{target} is the original HR image, $y_i^{(t-1)}$ is the image from the previous timestep, and $y_i^{(t)}$ is the image from the current timestep [4]. Equation 1 reveals how the squared error between each pixel and its target has changed after taking a certain action [4]. If the given agent chooses an action that improves the state, the reward is positive. If the action

makes the state worse, the reward is negative. Herein, we attempt to maximize the total reward in Equation 1 by minimizing the squared error between each state and the original image. This forces the output image to resemble the original HR image.

It is important to note that the network does not alter the dimensionality of the input image. Therefore, the image being fed through the network has to be pre-processed to match the scale of the desired output. In our work we upsample by a scaling factor of 4 via bicubic interpolation, which contributes to the novelty of our algorithm as we apply RL to SISR. This initial upsampling functions as its own action that lays the foundation of the output image. Figure 2 illustrates the action of "do nothing" which, for timestep 1, in effect is choosing to preserve the bicubic pixel value instead of replacing it with the pixel value of another option in the action set.

4. EXPERIMENTS and RESULTS

The proposed network is trained on the DIV2K dataset [1], consisting of 800 high definition HR training images. For memory retention and speed, the network is trained on 60×60 random crops of the 800 training images. These images are initially blurred using a Gaussian filter, which approximates real-world distortions in the image capture process by simulating a camera's point spread function [19]. A standard deviation of $\sigma = 1.3$ is used in each of the images' spatial dimensions, as per the advice of [24]. After being blurred, they are bicubicly downsampled by a factor of 4 and immediately upsampled using the same interpolation method. Note that the results in Tables 1 and 2 for the four GAN algorithms do not match those in the original papers since the authors did not make any reference to using a Gaussian filter in the creation of their LR images. The bicubicly upsampled images are fed to the network in the initial timestep. This initial image acts as an action in and of itself because the network can choose to "do nothing", keeping the bicubic pixel value. The network requires approximately 48 hours to train for 10,000 epochs on a single NVIDIA RTX 2080Ti.

For testing and validating the network, the whole image is sent through the network. Each pixel has its own agent and chooses which action produces the best result. The same preprocessing approach is carried out: blurred, downsampled, and upsampled. The testing images come from commonly used testing datasets: Set5, DIV2K, and Urban100. Images 0855, 0878, 0879, and 0891 from DIV2K are excluded from the dataset due to memory issues resulting from their size. All results in Tables 1 and 2 are calculated without these four images to maintain a fair comparison. These datasets make it simple to quantitatively compare results with other SISR techniques. We compare results with the following metrics: Mean Squared Error (MSE), Peak Signal-to-Noise Ratio (PSNR), and Structural Similarity index (SSIM).

Table 1. Quantitative comparison between the proposed method and other competing SR algorithms in RGB color space (Best results in **bold**, second best in *italics*).

Method	Metric	Set5	Urban100	DIV2K Val.
bicubic	MSE	193.46	563.64	209.81
	PSNR	26.69	21.70	26.66
	SSIM	0.7961	0.6442	0.7581
edsr baseline	MSE	105.90	426.18	163.37
	PSNR	28.73	23.14	27.92
	SSIM	0.8631	0.7450	0.8100
esrgan	MSE	250.04	1031.3	507.44
	PSNR	24.61	18.89	22.87
	SSIM	0.7166	0.5569	0.6298
esrgan-psnr	MSE	107.99	457.53	170.42
	PSNR	28.61	22.57	27.67
	SSIM	0.8625	0.7482	*0.8139*
ppon	MSE	222.01	828.63	437.44
	PSNR	25.21	19.86	23.39
	SSIM	0.7335	0.6213	0.6622
Ours	MSE	**99.563**	**411.09**	**159.52**
	PSNR	**28.97**	**23.28**	**28.08**
	SSIM	**0.8664**	**0.7517**	**0.8140**
Ours VGG Rewards	MSE	107.83	428.32	165.35
	PSNR	28.65	23.10	27.85
	SSIM	0.8612	0.7417	0.8044
Ours Patches	MSE	*102.33*	*417.38*	*161.98*
	PSNR	*28.87*	*23.19*	*27.96*
	SSIM	*0.8645*	*0.7503*	0.8129

Table 2. Quantitative comparison between the proposed method and other competing SR algorithms in YCbCr color space (Best PSNR results in **bold**, second best in *italics*). Note: for SR option MSE and SSIM values, refer to Table 1.

Method	Metric	Set5	Urban100	DIV2K Val.
bicubic	PSNR	27.10	21.81	26.75
edsr baseline	PSNR	**29.19**	**23.25**	**28.01**
esrgan	PSNR	24.87	19.01	22.97
esrgan-psnr	PSNR	*29.03*	*22.66*	*27.75*
ppon	PSNR	25.43	19.94	23.45
Ours	MSE	192.37	560.08	208.28
	PSNR	27.18	21.84	26.83
	SSIM	0.7961	0.6447	0.7585

4.1. Patch-wise Agents

In some images, a "noisy" appearance can be observed where pixels are replaced by a number of different actions within a small area. We attempt to minimize this effect by converting our original network (RGB color space, MSE-based rewards) from an agent-per-pixel configuration to

HR | Ours RGB | Ours RGB Patches

Figure 4. DIV2K validation image 0804 showing the difference between pixel-wise and patch-wise agents.

agent-per-patch using square patches of pixels (tested with 3×3 patches). To accommodate this reduced number of agents, a max pooling layer is added immediately before the network (Figure 1) splits into its policy and value branches. The final policy is then upsampled to the image size via nearest neighbor interpolation, applying the single action to all pixels within the patch. Agent rewards are determined by modifying Equation 1 to calculate the average over their respective patches:

$$r_i^{(t)} = \frac{1}{p^2}\Sigma_{k=0}^{p-1}\Sigma_{j=0}^{p-1}(I_{i_x+j,i_y+k}^{target} - y_{i_x+j,i_y+k}^{(t-1)})^2 - (I_{i_x+j,i_y+k}^{target} - y_{i_x+j,i_y+k}^{(t)})^2, \quad (2)$$

where p is the size of the patch in each dimension and i_x and i_y are the x and y coordinates of the agents. In Figure 4, we can compare the "noisy" phenomenon of the agent-per-pixel configuration to the agent-per-patch. Although using patches makes the "noisy" appearance subjectively less noticeable, it is not entirely removed. Our patch-based method (see Table 1) achieved the second best MSE and PSNR values across all three datasets. For SSIM values, it achieved the second best results for Set5 and Urban100 datasets.

To verify that the patch-wise agents are sourcing pixel values from the best options, the PSNR value for each patch is calculated for all option images to determine which ones are correct on a patch-by-patch basis. A tolerance value is introduced to allow for multiple options to be considered correct if their PSNR values are sufficiently close. This gives us more insight into how much the image quality of each option varies between different regions. The tolerance values used are 0, 0.5, 1.0, 1.5, 2.0, and 2.5 dB. Figure 5 provides a simple example of this comparison in which four options are provided as sources for a four-patch image. In the example, only 0, 0.5, 1.0, and 1.5 dB tolerances are used to illustrate the premise.

With a 0 dB tolerance (Figure 5b), there can only be one correct option for each patch. Since option 2 is correct for the highest number of patches, it is the overall best option. With a 0.5 dB tolerance (Figure 5c), the highest PSNR values in patch 1 are close enough that options 1 and 2 are both considered correct, but option 2 is still the overall best option. With a 1.5 dB tolerance (Figure 5e), all patches have multiple correct options. Again, option 2 is still the overall best option. Looking at the 0 dB tolerance case in particular, one can see that there is the potential to create an image better than the overall best option, which is only correct for half of the patches.

Figure 6 shows the agent-chosen patch sources for the

Patch 1	Patch 2
27.0	20.3
Patch 3	Patch 4
17.9	23.0

Image from Option 1

Patch 1	Patch 2
26.7	21.0
Patch 3	Patch 4
22.0	23.7

Image from Option 2

Patch 1	Patch 2
25.2	19.0
Patch 3	Patch 4
16.5	25.0

Image from Option 3

Patch 1	Patch 2
26.0	18.0
Patch 3	Patch 4
20.5	22.5

Image from Option 4

a. Example PSNR Values [dB]

Patch 1 Img 1 27.0	Patch 2 **Img 2 21.0**
Patch 3 **Img 2 22.0**	Patch 4 Img 3 25.0

b. 0 dB Tolerance

Patch 1 Img 1, **Img 2** 27.0, **26.7**	Patch 2 **Img 2 21.0**
Patch 3 **Img 2 22.0**	Patch 4 Img 3 25.0

c. 0.5 dB Tolerance

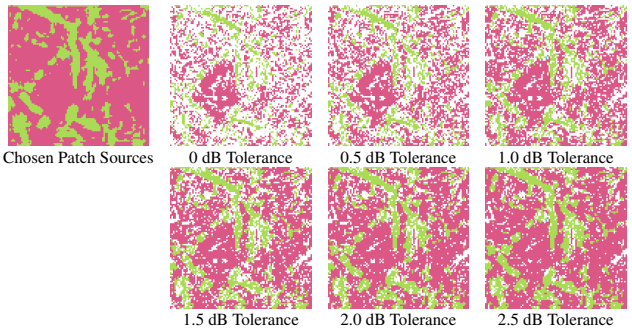
Patch 1 Img 1, **Img 2**, Img 4 27.0, **26.7,** 26.0	Patch 2 Img 1, **Img 2** 20.3, **21.0**
Patch 3 **Img 2 22.0**	Patch 4 Img 3 25.0

d. 1.0 dB Tolerance

Patch 1 Img 1, **Img 2**, Img 4 27.0, **26.7,** 26.0	Patch 2 Img 1, **Img 2** 20.3, **21.0**
Patch 3 **Img 2**, Img 4 **22.0,** 20.5	Patch 4 **Img 2**, Img 3 **23.7,** 25.0

e. 1.5 dB Tolerance

Figure 5. Simple example illustrating how we can determine the benefit of the patch-wise agents. For each tolerance, correct image options (highest PSNR values) are listed (abbreviated as "Img" for compactness) and the overall best option is shown in **bold**. If the number of correct patches chosen by agents is greater than the number of correct patches in the overall best option, then their use is beneficial. See Figures 6 and 9 for results with a real image.



Chosen Patch Sources | 0 dB Tolerance | 0.5 dB Tolerance | 1.0 dB Tolerance | 1.5 dB Tolerance | 2.0 dB Tolerance | 2.5 dB Tolerance

Figure 6. Options chosen by the patch-wise agents for Set5 bird (see Figure 2 for color key). For the various PSNR tolerances, incorrect patches are whited out.

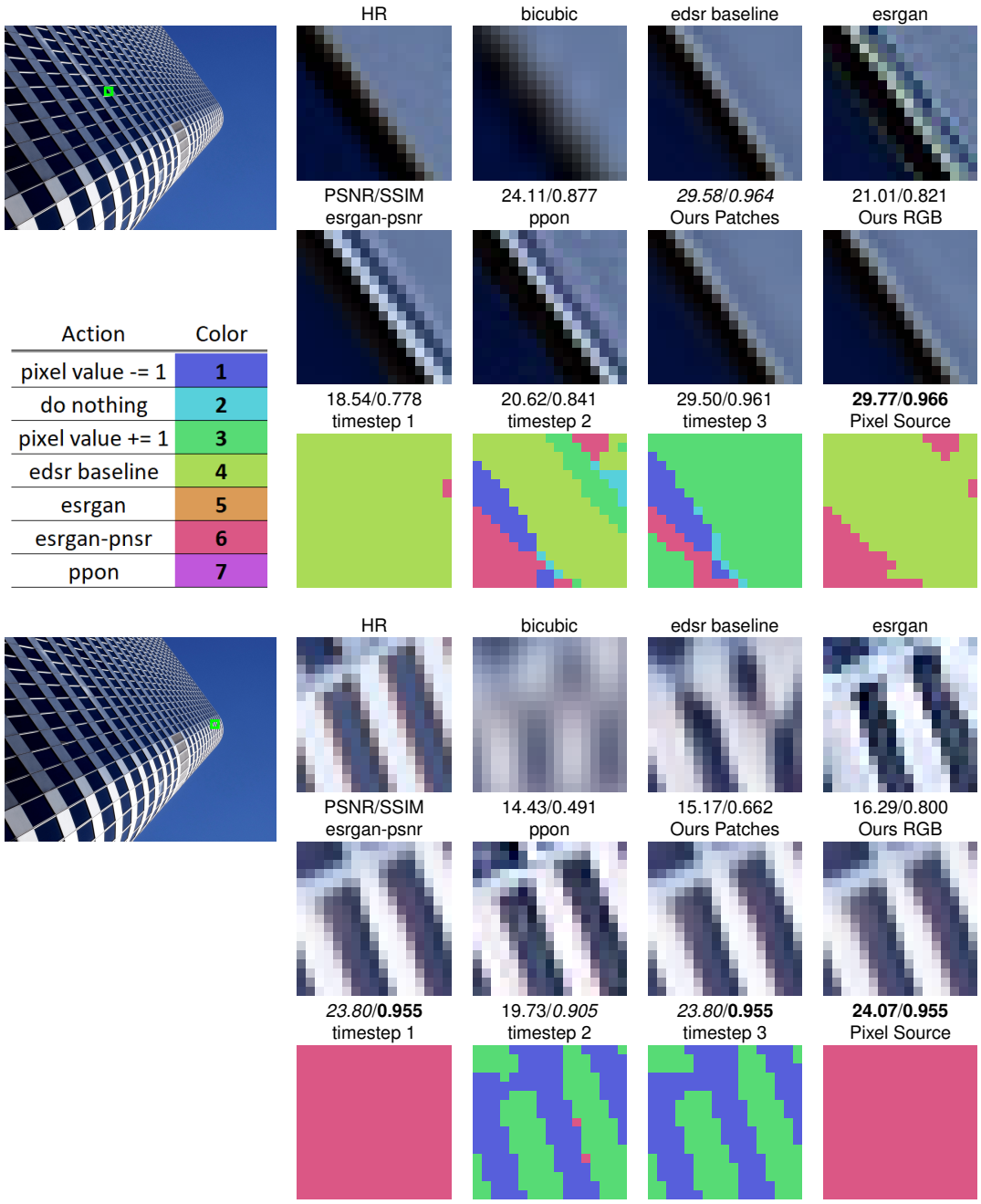


Figure 7. Qualitative comparison between the proposed methods and other competing SR algorithms on image img_005 from the Urban100 dataset (Best results in **bold**, second best in *italics*). The resulting image chooses between multiple options (seen in Pixel Source), where the chosen option is the best. The Pixel Source excludes pixel increment and decrement to show the source image of each pixel. The timestep and Pixel Source images are from our RGB pixel-wise method.

Set5 bird image. Patches may have been altered with increment or decrement actions; however, those are ignored here as only the original source of the pixel values is of interest. For each tolerance value, the patch is displayed in its corresponding source color (see Figure 2) if the algorithm chose a correct source or is whited out if it chose an incorrect source. The bar chart in Figure 9 plots the percentage of the patches that the algorithm chose from correct sources with the green bars (left) and the percentage of the patches that are correct in the overall best option image with the orange bars (right). The Set5 bird image, created by our RGB patch-wise agents, contains more correct patches than the overall best option image and thus is quantitatively better. This finding reiterates the results in Table 1, in which our

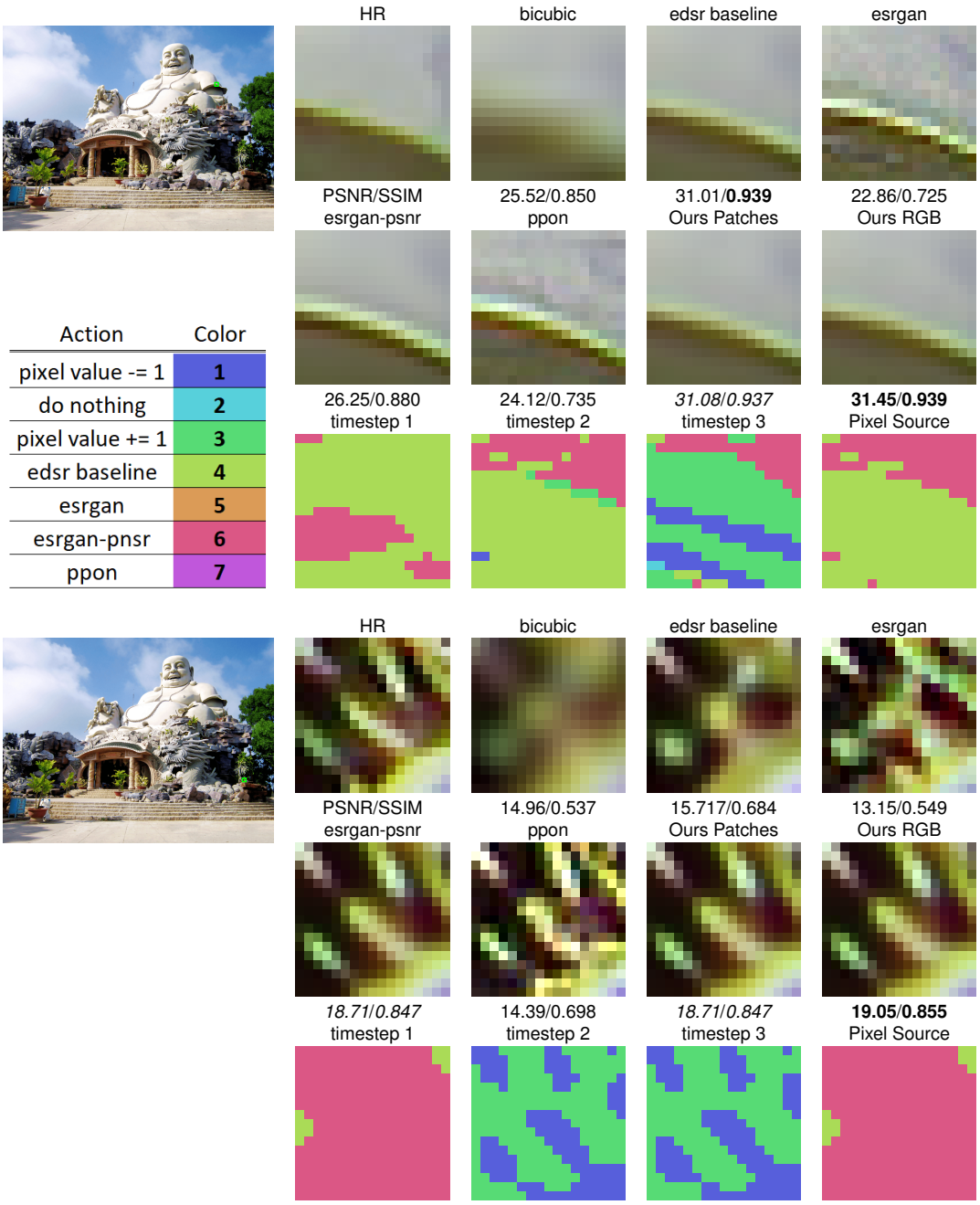


Figure 8. Qualitative comparison between the proposed methods and other competing SR algorithms on image 0818 from the DIV2K dataset (Best results in **bold**, second best in *italics*). The resulting image chooses between multiple options (seen in Pixel Source), where the chosen option is the best. The Pixel Source excludes pixel increment and decrement to show the source image of each pixel. The timestep and Pixel Source images are from our RGB pixel-wise method.

images have better PSNR values than any of the individual options.

4.2. YCbCr Color Space

Rather than focusing on a purely color-oriented color space, many recent SR papers explore images in a luminance-oriented color space. Rui Gong et al. tested SR reconstruction in a number of color spaces and recommended focusing on the L* channel in CIELAB color space or the Y channel in YIQ color space, both of which are luminance-oriented [5]. We follow in the footsteps of recent SR work [7, 8, 13, 22, 23] and also apply our method to images in YCbCr color space. All three channels are passed through the network and replaced by the values from the

images in the action space; however, as in recent SR work, only the Y channel is considered when calculating PSNR values. This method is explored using an agent for every pixel and an MSE-based reward function. Our results, in Table 2, are only marginally better than the bicubic images. This is because the policy learned by the agents kept all pixels from the bicubic images and only modified some of them with value increments and decrements.

4.3. VGG Rewards

Perceptual loss in the feature space has previously been explored by a number of sources. Instead of calculating the pixel-by-pixel error between the ground truth image and the generated image, the error is calculated in the feature space. This encourages the the network to generate images that have a similar feature representation to the ground truth images [17]. Sajjadi et al. extracts the feature representations by sending the HR and SR images through a pre-trained implementation of the VGG-19 network [17]. This network consists of a combination of convolution and max pooling layers that extract features as it decreases the spatial dimension of the image [21]. We extract features from the "conv2_2" layer of Chainer's VGG-19 implementation. The MSE is calculated between the ground truth and generated feature spaces before being upsampled via nearest neighbor interpolation to the original crop size of 60×60. This tensor is then added to the total reward calculated with Equation 1. The VGG results, in Table 1, are only slightly better than the ESRGAN-PSNR images, but present a good quantitative comparison.

4.4. Discussion

When GANs were first introduced, they aggressively exceeded existing SISR models. Since the development of so many GANs, their breakthroughs have started to level off. Instead of learning a complex mapping and neglecting the pixel level of an image, our method works on a pixel-by-pixel (or patch-by-patch) basis. Therefore, we must zoom in to the pixel level of an image to observe the full effect of our method. Figures 7 and 8 show a qualitative comparison between sections of two different images. Figure 7 shows image img_005 selected from the Urban100 dataset. This image demonstrates the advantage of using our method for SR. The timesteps manifest the algorithm's process through visual representations. Looking at these timesteps for the first section, we can see the algorithm chooses mostly EDSR and ESRGAN-PSNR (see Figure 2). The algorithm also chooses to increment and decrement the pixel values which helps to increase the PSNR and SSIM values. We can see that EDSR most accurately recreates the section, both qualitatively and quantitatively. All the other options either blur the image or add in artifacts not found in the HR image. Looking at the timesteps for the second section, we can see

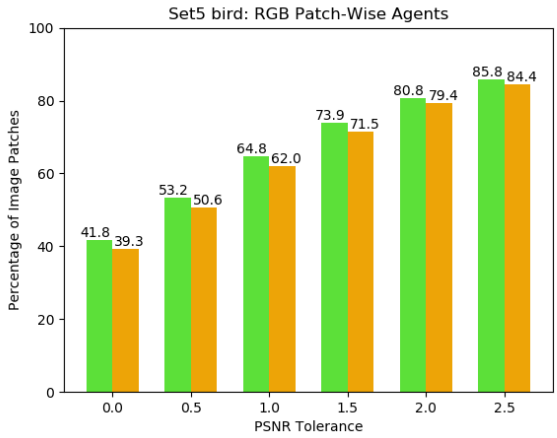


Figure 9. The green bar (left) shows the percentage of patches chosen by the patch-wise agents that are correct. The orange bar (right) shows the percentage of patches in the overall best option image that are correct.

the algorithm chooses exclusively ESRGAN-PSNR. The algorithm also chooses to increment and decrement the pixel values which helps it to surpass the PSNR and SSIM values of ESRGAN-PSNR. Here we can see that ESRGAN-PSNR most accurately recreates the section, both qualitatively and quantitatively. This illustrates that some SR algorithms are more adequate for different parts of an image. Similarly, Figure 8 shows sections of image 0818 from the DIV2K validation dataset. Again, we show that selecting from multiple sources can increase the qualitative and quantitative results of an image. In both figures, our method either obtains the highest metric values or scores the same as one of the options. Consequently, our method learns to quantitatively outperform the best SR algorithms over the entire datasets (see Table 1).

5. CONCLUSION

We modified the pixelRL problem setting to handle the SISR task. Our method learns a strategy to increase the resolution of an image by choosing the best pixel values from each of the GAN options provided. The qualitative and quantitative results support our theory: choosing between multiple GAN options at the pixel (or patch) level can increase the overall SISR performance. Our visual results give us the advantage of observing where each GAN option is most effective. Some options are more effective in low frequency portions of the image, while others are more effective in high frequency portions. This expression can be used to determine which GAN to use in certain scenarios. Other SISR applications can benefit greatly from this visual insight.

References

[1] Eirikur Agustsson and Radu Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017.

[2] Qingxing Cao, Liang Lin, Yukai Shi, Xiaodan Liang, and Guanbin Li. Attention-Aware Face Hallucination via Deep Reinforcement Learning. *arXiv:1708.03132 [cs]*, Aug. 2017. arXiv: 1708.03132.

[3] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014.

[4] Ryosuke Furuta, Naoto Inoue, and Toshihiko Yamasaki. Fully Convolutional Network with Multi-Step Reinforcement Learning for Image Processing. *arXiv:1811.04323 [cs]*, Nov. 2018. arXiv: 1811.04323.

[5] Rui Gong, Yi Wang, Yilin Cai, and Xiaopeng Shao. How to deal with color in super resolution reconstruction of images. *Optics Express*, 25(10):11144, May 2017.

[6] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[7] Jingwei Guan, Cheng Pan, Songnan Li, and Dahai Yu. SRDGAN: learning the noise prior for Super Resolution with Dual Generative Adversarial Networks. *arXiv:1903.11821 [cs]*, Mar. 2019. arXiv: 1903.11821.

[8] Zheng Hui, Jie Li, Xinbo Gao, and Xiumei Wang. Progressive Perception-Oriented Network for Single Image Super-Resolution. *arXiv:1907.10399 [cs, eess]*, July 2019. arXiv: 1907.10399.

[9] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4681–4690, 2017.

[10] Wentian Li, Xidong Feng, Haotian An, Xiang Yao Ng, and Yu-Jin Zhang. Mri reconstruction with interpretable pixel-wise operations using reinforcement learning. In *AAAI*, 2020.

[11] Z. Li and X. Zhang. Collaborative Deep Reinforcement Learning for Image Cropping. In *2019 IEEE International Conference on Multimedia and Expo (ICME)*, pages 254–259, July 2019.

[12] Xuan Liao, Wenhao Li, Qisen Xu, Xiangfeng Wang, Bo Jin, Xiaoyun Zhang, Ya Zhang, and Yanfeng Wang. Iteratively-Refined Interactive 3d Medical Image Segmentation with Multi-Agent Reinforcement Learning. *arXiv:1911.10334 [cs]*, Nov. 2019. arXiv: 1911.10334.

[13] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced Deep Residual Networks for Single Image Super-Resolution. *arXiv:1707.02921 [cs]*, July 2017. arXiv: 1707.02921.

[14] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous Methods for Deep Reinforcement Learning. *arXiv:1602.01783 [cs]*, Feb. 2016. arXiv: 1602.01783.

[15] Junting Pan, Cristian Canton-Ferrer, Kevin McGuinness, Noel E. O'Connor, Jordi Torres, Elisa Sayrol, and Xavier Gir'o i Nieto. Salgan: Visual saliency prediction with generative adversarial networks. *CoRR*, abs/1701.01081, 2017.

[16] Guim Perarnau, Joost van de Weijer, Bogdan Raducanu, and Jose M. Álvarez. Invertible conditional gans for image editing. *CoRR*, abs/1611.06355, 2016.

[17] Mehdi S. M. Sajjadi, Bernhard Schölkopf, and Michael Hirsch. Enhancenet: Single image super-resolution through automated texture synthesis. *CoRR*, abs/1612.07919, 2016.

[18] Haichao Shi, Jing Dong, Wei Wang, Yinlong Qian, and Xiaoyu Zhang. SSGAN: secure steganography based on generative adversarial networks. *CoRR*, abs/1707.01613, 2017.

[19] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P. Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1874–1883, June 2016. ISSN: 1063-6919.

[20] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Josh Susskind, Wenda Wang, and Russell Webb. Learning from simulated and unsupervised images through adversarial training. *CoRR*, abs/1612.07828, 2016.

[21] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[22] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Chen Change Loy, Yu Qiao, and Xiaoou Tang. ESRGAN: Enhanced Super-Resolution Generative Adversarial Networks. *arXiv:1809.00219 [cs]*, Sept. 2018. arXiv: 1809.00219.

[23] Yifan Wang, Federico Perazzi, Brian McWilliams, Alexander Sorkine-Hornung, Olga Sorkine-Hornung, and Christopher Schroers. A Fully Progressive Approach to Single-Image Super-Resolution. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 977–97709, June 2018. ISSN: 2160-7516, 2160-7508.

[24] Chih-Yuan Yang, Chao Ma, and Ming-Hsuan Yang. Single-Image Super-Resolution: A Benchmark. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, volume 8692, pages 372–386. Springer International Publishing, Cham, 2014.

[25] He Zhang, Vishwanath Sindagi, and Vishal M Patel. Image de-raining using a conditional generative adversarial network. *arXiv preprint arXiv:1701.05957*, 2017.