Supplementary Material - Bringing Old Photos Back to Life

Ziyu Wan[1*], Bo Zhang[2], Dongdong Chen[3], Pan Zhang[4], Dong Chen[2], Jing Liao[1†], Fang Wen[2]
[1]City University of Hong Kong [2]Microsoft Research Asia [3]Microsoft Cloud + AI
[4]University of Science and Technology of China

1. Overview

In this supplemental material, additional experimental details and results are provided, including:

- more details about data synthesis (Section 2);
- more details and qualitative results about scratch detection (Section 3);
- the detailed network architecture (Section 4);
- qualitative comparison during the user study (Section 5).

2. Degradation Model

2.1. Unstructured Degradation

We use the following operations to simulate the unstructured degradation. Specifically, 1) Gaussian white noise with $\sigma \in [5, 50]$. 2) Gaussian blur with kernel size $k \in [3, 5, 7]$ and standard deviation $\sigma \in [1.0, 5.0]$. 3) JPEG compression whose level is in the range of $[40, 100]$. 4) Box blur to mimic the lens defocus. We perform these synthesis defects with varying parameters in random order. In order to achieve more variations, we stochastically drop the operation with a probability of 30%. However, the synthesis cannot exactly match the appearance of real photo defects, and thus requires the proposed network to further reduce the domain gap.

2.2. Structured Degradation

Figure 4 in the main text shows that a realistic synthesis helps the network generalize to real scratch detection. To this end, we collect 62 scratch texture images and 55 paper texture images, which are further augmented with elastic distortions. We use layer addition, lighten-only and screen modes with random level of opacity to blend the scratch textures over the natural images from the dataset. Besides, in order to simulate large-area photo damage, we generate holes with feathering and random shape where the underneath paper texture is unveiled. Note that we also introduce film grain noise and blur with random kernel to simulate the global defects at this stage so that the synthetic data has a similar global style as the real old photos. These injected noises are beneficial in that they make the distribution of synthetic and real data become more overlapped. Examples of synthesized scratched old photos are shown in Figure 1.

3. Scratch Detection

Now we have the synthetic image set $\mathcal{S} \subset \mathbb{R}^{H\times W\times 3}$ with the associated segmentation maps $\mathcal{Y} \subset [0,1]^{H\times W}$, where H and W denote height and width respectively. Let $\{s_i, y_i | s_i \in \mathcal{S}, y_i \in \mathcal{Y}\}$ denote the training pairs for supervised learning. We train a network $\mathcal{F}_\theta$ parameterized by θ, to predict the probability of local defects at each location, thus obtaining the predicted segmentation map $\hat{y}_i = \mathcal{F}_\theta(s_i)$. We minimize the cross-entropy loss between the prediction and the ground truth,

$$\mathcal{L}_{CE} = \mathbb{E}_{(s_i,y_i)\sim(\mathcal{S},\mathcal{Y})} \left\{ \alpha \sum_{h=1}^{H}\sum_{w=1}^{W} -y_i^{(h,w)} \log \hat{y}_i^{(h,w)} - (1-\alpha) \sum_{h=1}^{H}\sum_{w=1}^{W} (1-y_i^{(h,w)}) \log(1-\hat{y}_i^{(h,w)}) \right\}. \quad (1)$$

* Work done during the internship at Microsoft Research Asia
† Corresponding author

Figure 1. **Synthetic photos with scratches.**

Here, we use α_i as image-wise weight to remedy the imbalance of positive and negative detections, and α_i is computed according to the proportion of defect areas in the synthetic image s_i, specifically,

$$\alpha_i = \frac{[y_i = 1]}{[y_i = 1] + [y_i = 0]]}. \tag{2}$$

Besides, we also introduce the focal loss to focus on the hard samples,

$$\mathcal{L}_{FL} = \mathbb{E}_{(s_i,y_i)\sim(\mathcal{S},\mathcal{Y})}\left\{\sum_{h=1}^{H}\sum_{w=1}^{W} -(1-p_i^{(h,w)})^{\gamma}\log p_i^{(h,w)}\right\}, \tag{3}$$

where,

$$p_i^{(h,w)} = \begin{cases} \hat{y_i}^{(h,w)} & \text{if } y_i^{(h,w)} = 1 \\ 1-\hat{y_i}^{(h,w)} & \text{otherwise} \end{cases} \tag{4}$$

The whole detection objective is

$$\mathcal{L}_{Seg} = \mathcal{L}_{CE} + \beta\mathcal{L}_{FL}. \tag{5}$$

We set the parameters in Equations (3) and (5) with $\gamma = 0.2$ and $\beta = 10$. And the detection network adopts Unet architecture which reuses low-level features through extensive skip connection.

To further improve the detection performance on real old photos, we annotate the local defects for 783 collected old photos, among which 400 images are used to finetune the network. Since the number of labeled real samples is limited, in order to make full use of these data, we train the temporally ensembled network, whose parameters is moving averaged during training, with the same loss as Equation 1. During inference, we apply multiscale testing and the segmentation map after binary thresholding is denoted by M_i. Some sampled scratch detection masks and restoration results of test dataset are shown in Figure 2.

Figure 2. **Scratch detection results.** GT masks are labeled by hand.

4. Network Architectures

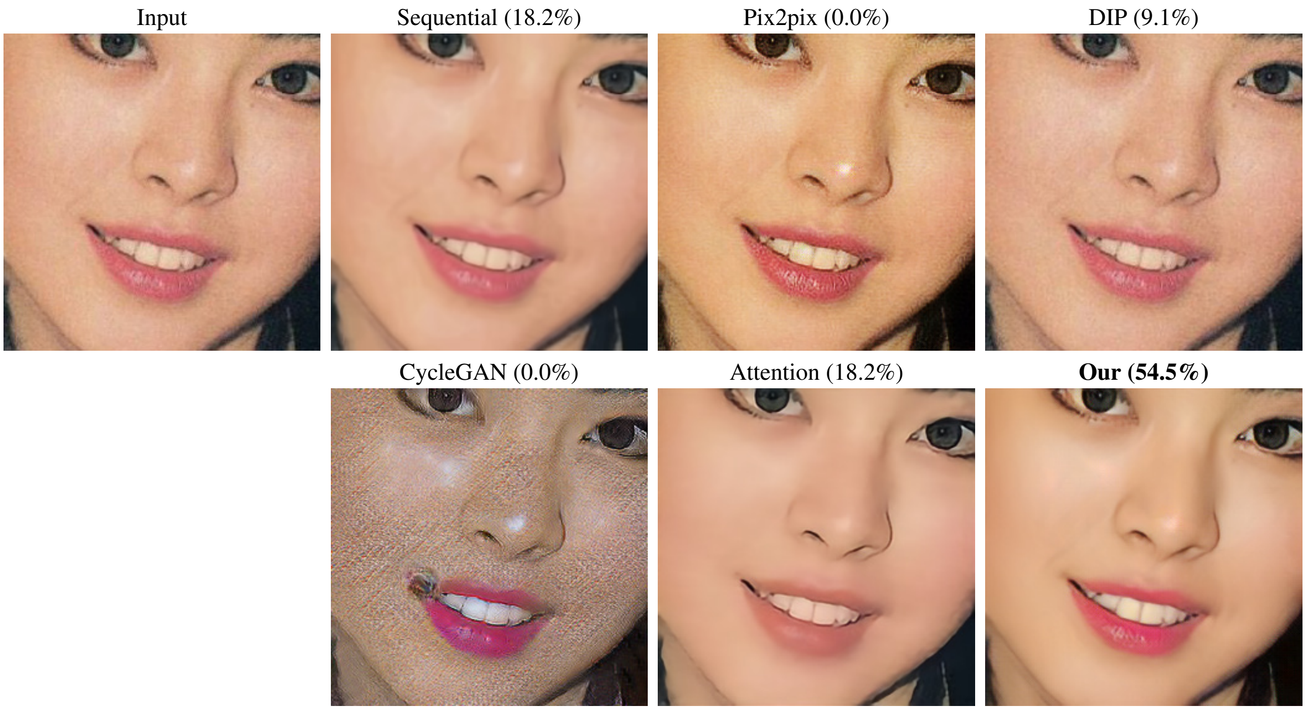
Table 1 shows the detailed network structure.

Module	Layer	Kernel size / stride	Output size
Encoder E	Conv	$7 \times 7/1$	$256 \times 256 \times 64$
	Conv	$4 \times 4/2$	$128 \times 128 \times 64$
	Conv	$4 \times 4/2$	$64 \times 64 \times 64$
	ResBlock×4	$3 \times 3/1$	$64 \times 64 \times 64$
Generator G	ResBlock×4	$3 \times 3/1$	$64 \times 64 \times 64$
	Deconv	$4 \times 4/2$	$128 \times 128 \times 64$
	Deconv	$4 \times 4/2$	$256 \times 256 \times 64$
	Conv	$7 \times 7/1$	$256 \times 256 \times 3$
Mapping $\mathcal{T}$	Conv	$3 \times 3/1$	$64 \times 64 \times 128$
	Conv	$3 \times 3/1$	$64 \times 64 \times 256$
	Conv	$3 \times 3/1$	$64 \times 64 \times 512$
	Partial Non-local	$1 \times 1/1$	$64 \times 64 \times 512$
	Resblock×2	$3 \times 3/1$	$64 \times 64 \times 512$
	ResBlock×6	$3 \times 3/1$	$64 \times 64 \times 512$
	Conv	$3 \times 3/1$	$64 \times 64 \times 256$
	Conv	$3 \times 3/1$	$64 \times 64 \times 128$
	Conv	$3 \times 3/1$	$64 \times 64 \times 64$

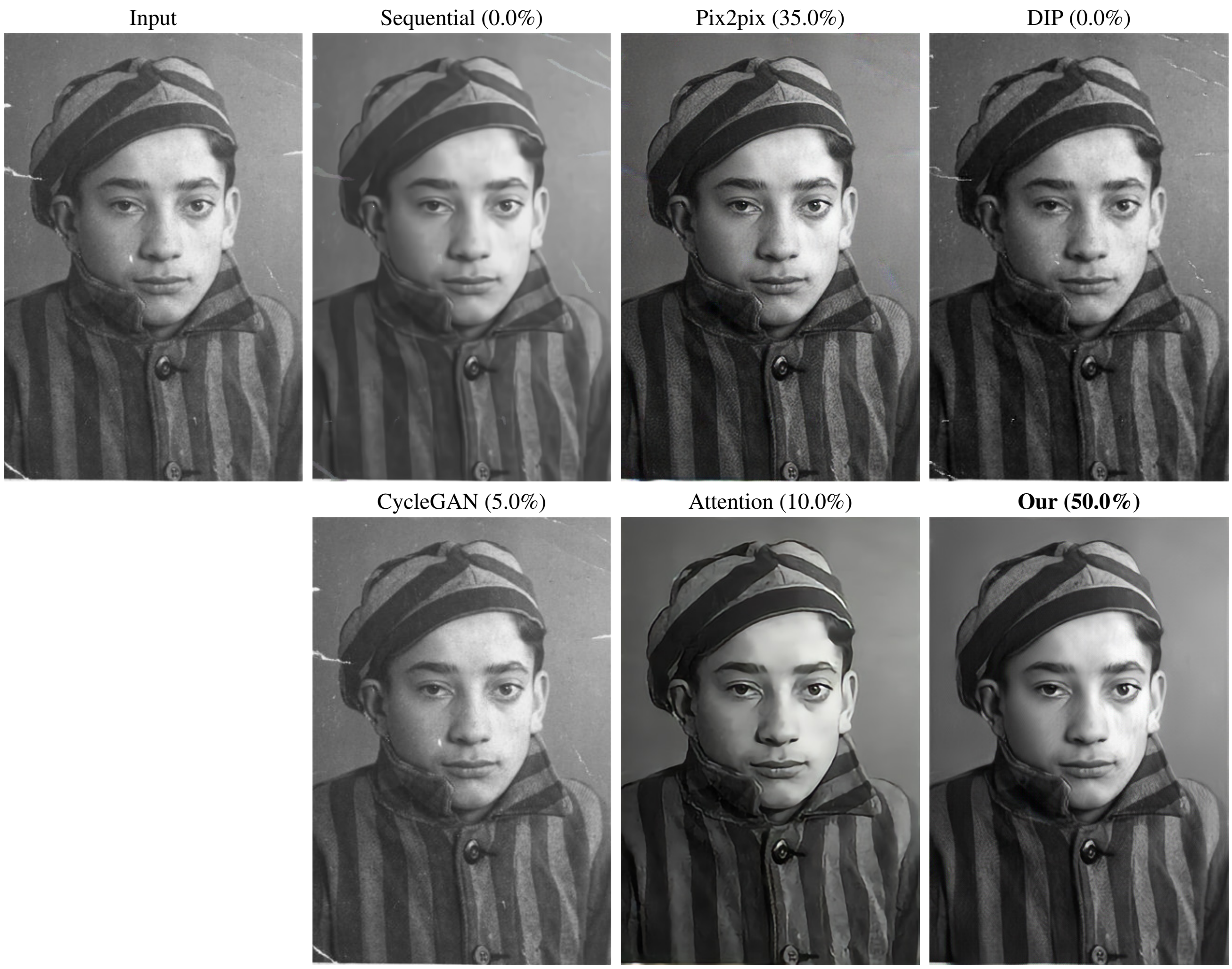
Table 1. **Detailed network structure.** The modules in the global branch of the mapping network are highlighted in gray.

5. User Study Results

We next show the comparisons conducted during the user study. The percentage of user voting is provided as well. Our compares favorably to state-of-the-art methods in most cases.


Input
Sequential (18.2%)
Pix2pix (0.0%)
DIP (9.1%)
CycleGAN (0.0%)
Attention (18.2%)
Our (54.5%)

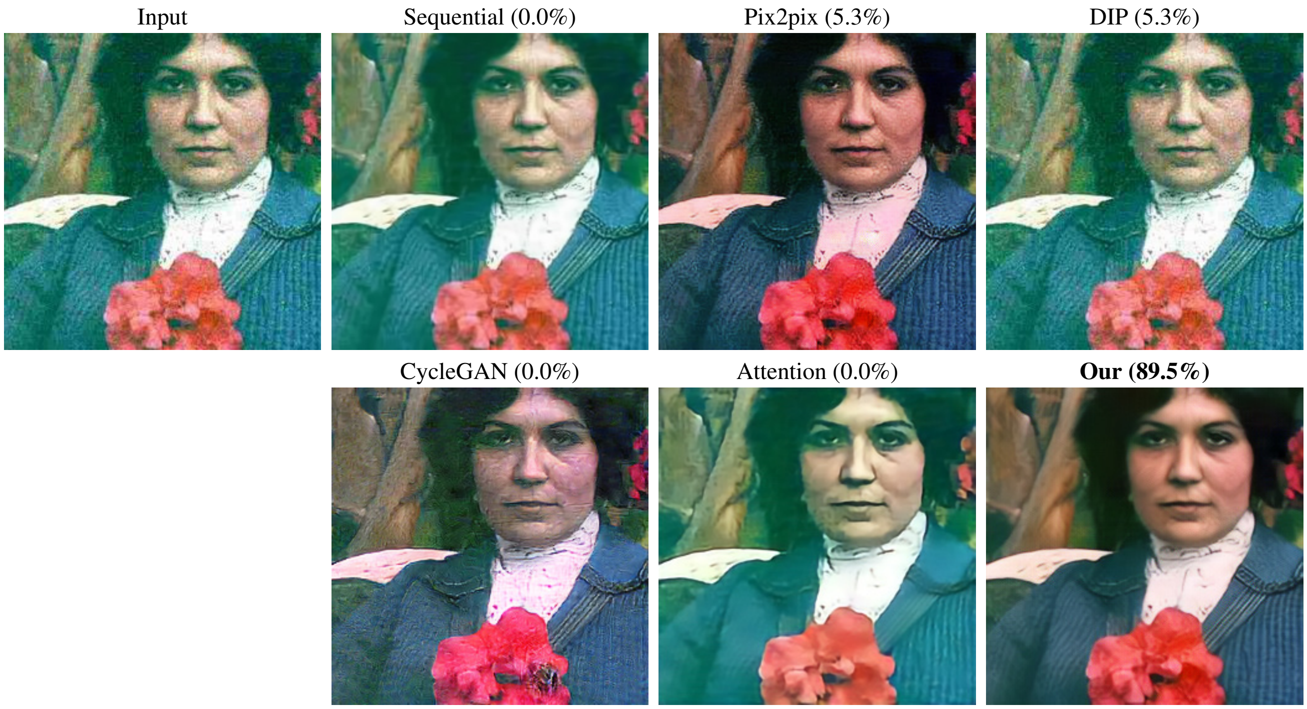

Input
Sequential (0.0%)
Pix2pix (25.0%)
DIP (0.0%)
CycleGAN (6.2%)
Attention (6.2%)
Our (62.5%)


Input
Sequential (0.0%)
Pix2pix (35.0%)
DIP (0.0%)
CycleGAN (5.0%)
Attention (10.0%)
Our (50.0%)


Input
Sequential (0.0%)
Pix2pix (50.0%)
DIP (0.0%)
CycleGAN (0.0%)
Attention (0.0%)
Our (50.0%)

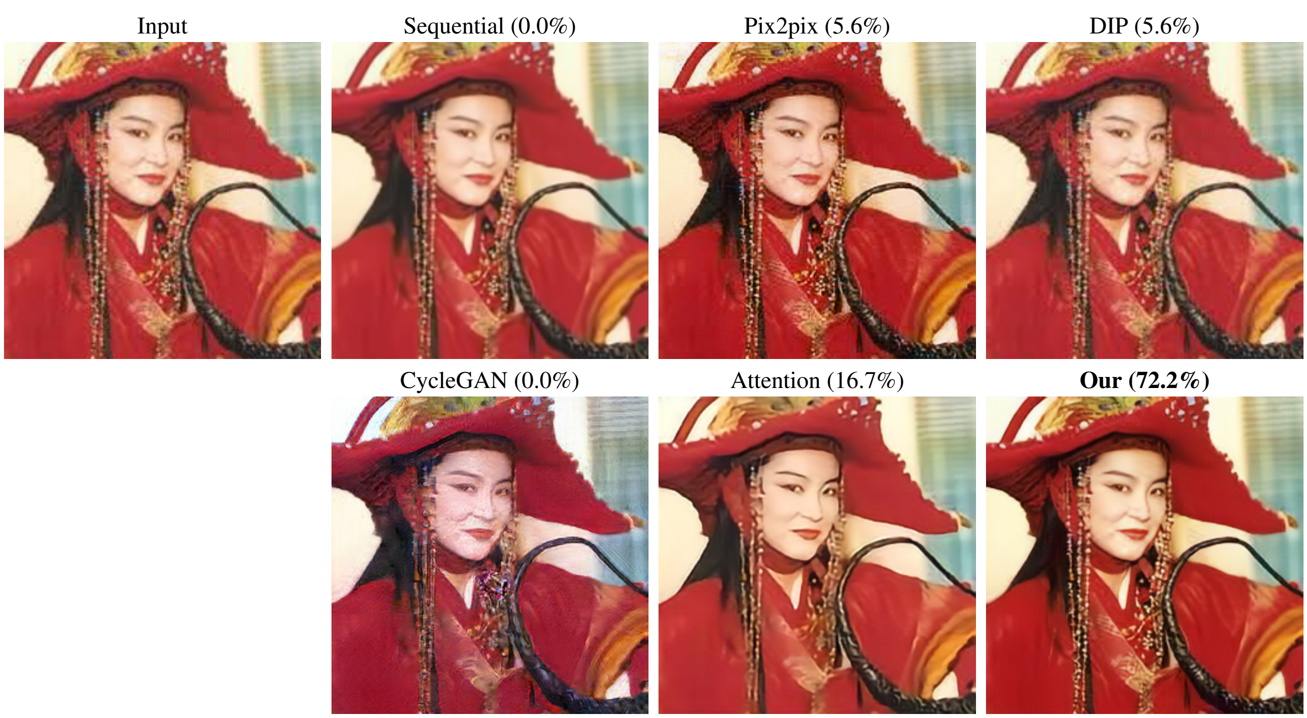

Input
Sequential (5.9%)
Pix2pix (29.4%)
DIP (5.9%)
CycleGAN (0.0%)
Attention (5.9%)
Our (52.9%)

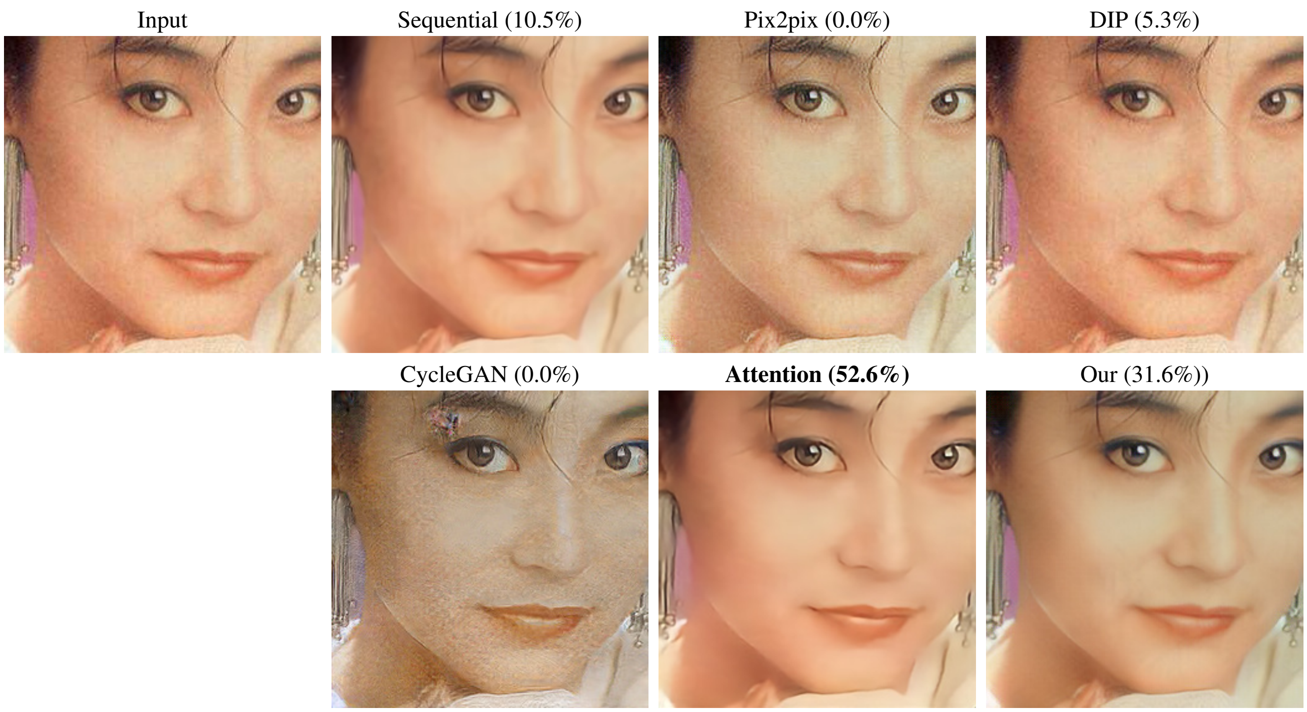

Input
Sequential (0.0%)
Pix2pix (29.4%)
DIP (0.0%)
CycleGAN (5.3%)
Attention (5.9%)
Our (78.9%)

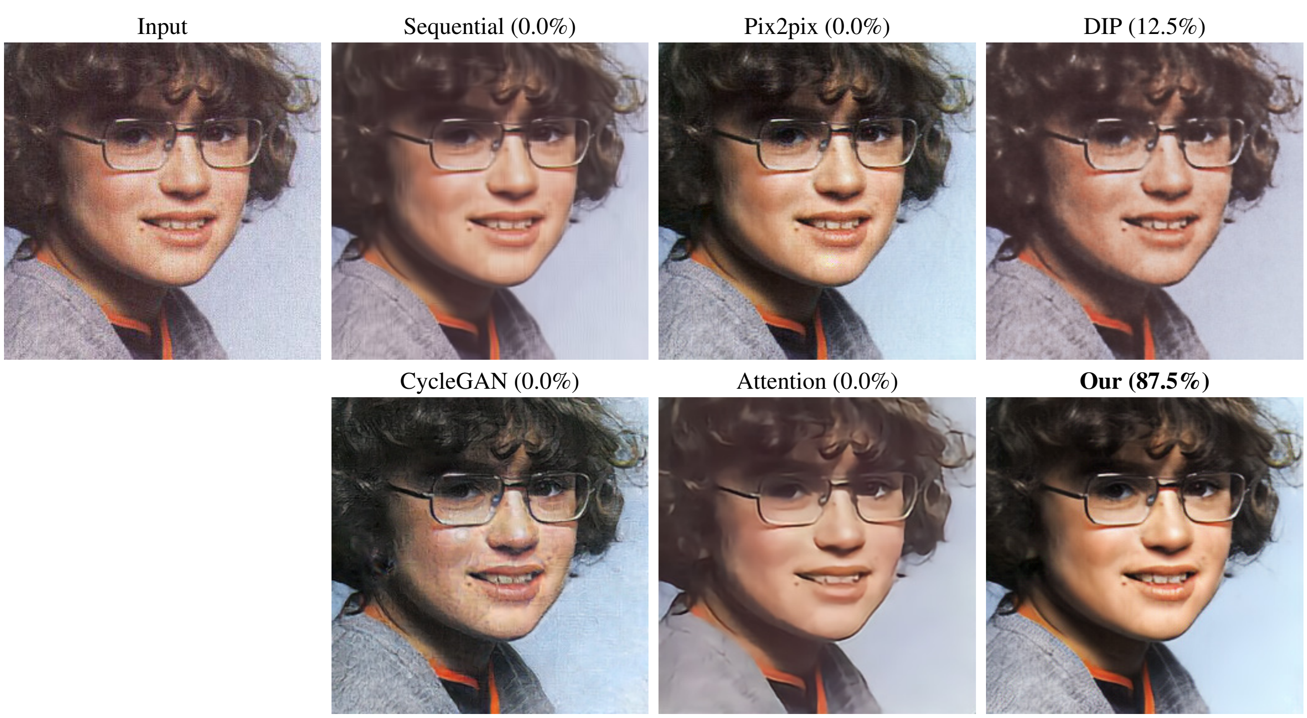

Input
Sequential (0.0%)
Pix2pix (5.3%)
DIP (5.3%)
CycleGAN (0.0%)
Attention (0.0%)
Our (89.5%)


Input
Sequential (0.0%)
Pix2pix (14.3%)
DIP (0.0%)
CycleGAN (0.0%)
Attention (28.6%)
Our (57.1%)


Input
Sequential (0.0%)
Pix2pix (4.8%)
DIP (0.0%)
CycleGAN (9.5%)
Attention (14.3%)
Our (71.4%)


Input
Sequential (0.0%)
Pix2pix (5.6%)
DIP (5.6%)
CycleGAN (0.0%)
Attention (16.7%)
Our (72.2%)


Input
Sequential (10.5%)
Pix2pix (0.0%)
DIP (5.3%)
CycleGAN (0.0%)
Attention (52.6%)
Our (31.6%))


Input
Sequential (0.0%)
Pix2pix (0.0%)
DIP (12.5%)
CycleGAN (0.0%)
Attention (0.0%)
Our (87.5%)

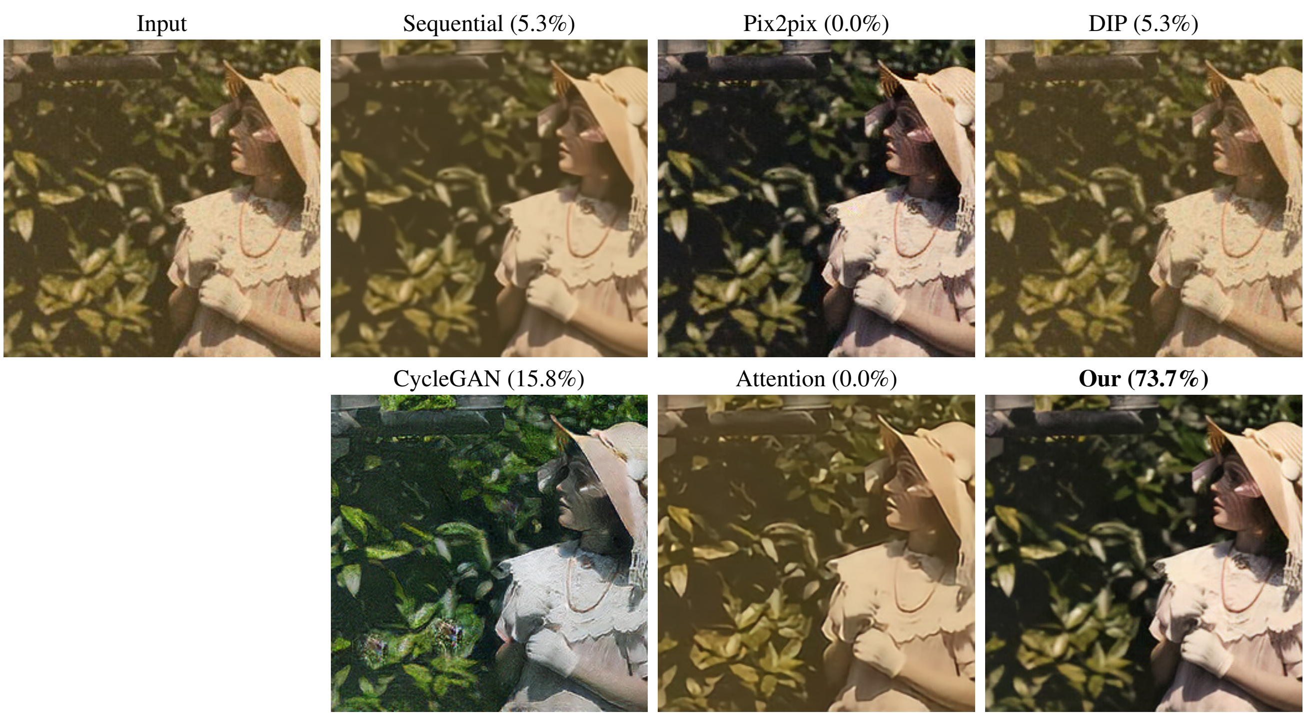

Input
Sequential (0.0%)
Pix2pix (16.7%)
DIP (5.6%)
CycleGAN (0.0%)
Attention (11.1%)
Our (66.7%)


Input
Sequential (4.5%)
Pix2pix (0.0%)
DIP (4.5%)
CycleGAN (0.0%)
Attention (9.1%)
Our (81.8%)


Input
Sequential (0.0%)
Pix2pix (10.5%)
DIP (0.0%)
CycleGAN (5.3%)
Attention (5.3%)
Our (78.9%)


Input
Sequential (16.7%)
Pix2pix (0.0%)
DIP (0.0%)
CycleGAN (0.0%)
Attention (58.3%)
Our (25.0%)


Input
Sequential (4.8%)
Pix2pix (9.5%)
DIP (4.8%)
CycleGAN (0.0%)
Attention (0.0%)
Our (81.0%)


Input
Sequential (5.3%)
Pix2pix (0.0%)
DIP (5.3%)
CycleGAN (15.8%)
Attention (0.0%)
Our (73.7%)


Input
Sequential (0.0%)
Pix2pix (16.7%)
DIP (0.0%)
CycleGAN (5.6%)
Attention (0.0%)
Our (77.8%)