Bringing Old Photos Back to Life

Abstract

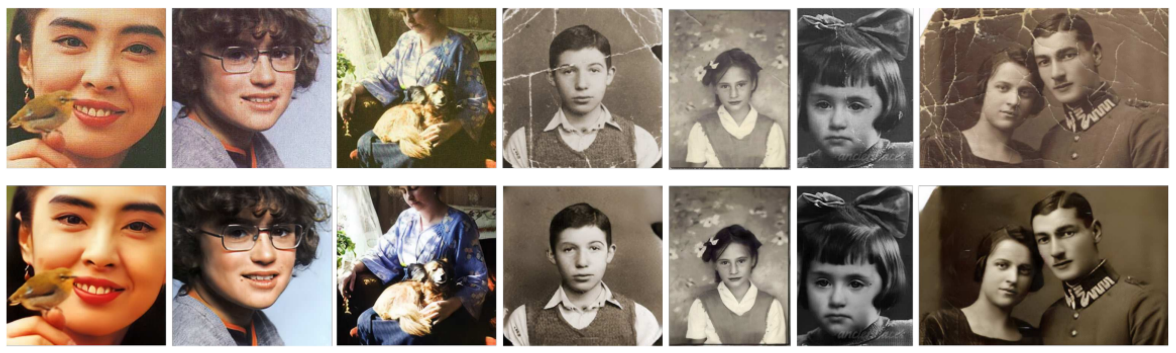


Figure 1: **Old image restoration results produced by our method.** Our method can handle the complex degradation mixed by both unstructured and structured defects in real old photos.

Core Content in this paper

> **Restore old photos that suffer from severe degradation through a deep learning approach.**

- What's the existing method for restoring old photos?
  Orginally, the existing method for restroing is supervised learning method, but the degradation in real photos is complex and the domain gap between synthetic images and real old photos makes the network fail to generalize
- What's the method for deep learning approach?
  - A novel triplet domain translation network by leveraging real photos along with massive synthetic image pairs.
  - Specifically, it trains **two VAEs(variational autoencoders)**
    - Transform old photos and clean photos into two latent spaces
    - The translation between these two latent spaces is learned with synthetic paired data
    - This translation generalizes well to real photos because the domain gap is closed in the compact latent space
  - The mixtures of mutliple degradations in a old photo is solved by
    - **global branch with a partial nonlocal block** targeting to the structured defects

      > scratches and dust spot

    - **a local branch targeting** to the unstructured defects

      > noises and blurriness

Introduction

There are many attempts for restoring old photos

1. Prior to the deep learning era, there are some attempts that restore photos by automatically detecting the localized defects such as scratches and blemishes, and filling in the damaged areas with inpainting techniques.
    - It's still outdated compared with modern photographic
2. With the emergence of deep learning, one can address a variety of low-level image restoration problems by exploiting the powerful representation capability of convolutional neural networks,

But there doesn't apply to old photo restoration.

1. The degradation process of old photos is rather complex
   There exists no degradation model that can realistically render the old photo artifact

2. Old photos are plagued with a compound of degradations
   So, it inherently requires different strategies for repair

    - **unstructured defects**
    - **structured defects**

So formulate the old photo restoration as a triplet domain translation problem

We leverage data from three domains and the translation is performed in latent space

The domains is below:

- real old photos
- synthetic images
- the corresponding ground truth

Synthetic images and the real photos are first transformed to the same latent space with a shared variational autoencoder

Meanwhile, another VAE is trained to project ground truth clean images into the corresponding latent space

The mapping between the two latent spaces is then learned with the synthetic image pairs, which restores the corrupted images to clean ones.

Advantages

1. the learned latent restoration can generalize well to real photos because of the domain alignment within the first variational autoencoder(VAE)
2. we differentiate the mixed degradation, and propose a partial nonlocal block that considers the long-range dependencies of latent features to specifically address the structured defects during the latent translation

Background

Degradations

- **structured defects**

  > scratches and dust spot

- **unstructured defects**

  > noises and blurriness

Variational AutoEncoder (VAE)

Method

In the following, we propose solutions to address the aforementioned generalization issue and mixed degradation issue respectively

Restoration via latent space translation

In order to mitigate the domain gap, we formulate the old photo restoration as an image translation problem, where we treat clean images and old photos as images from distinct domains and we wish to learn the mapping in between.

- the real photo domain **R**
- the synthetic domain **X** where images suffer from artificial degradation
- the corresponding ground truth domain **Y** that comprises images without degradation.

We denote images from three domains respectively with $r \in R$, $x \in X$ and $y \in Y$, where x and y are paired by data synthesizing, i.e., x is degraded from y

Directly learning the mapping from real photos $\{r\}_{i=1}^{N}$ to clean images $\{y\}_{i=1}^{N}$ is hard since they are not paired and thus unsuitable for supervised learning. decomposet he translation with two stages, which are illustrated in Figure 2.

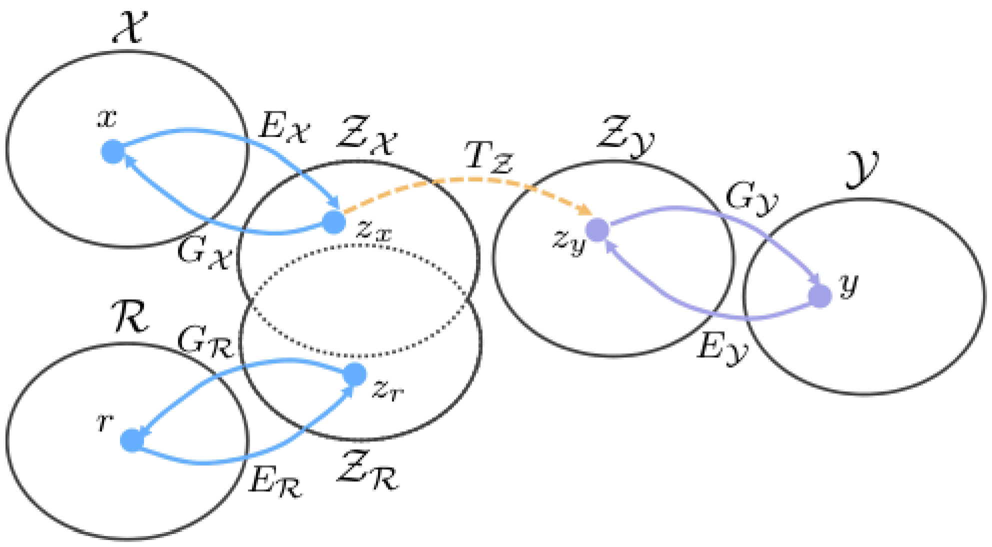


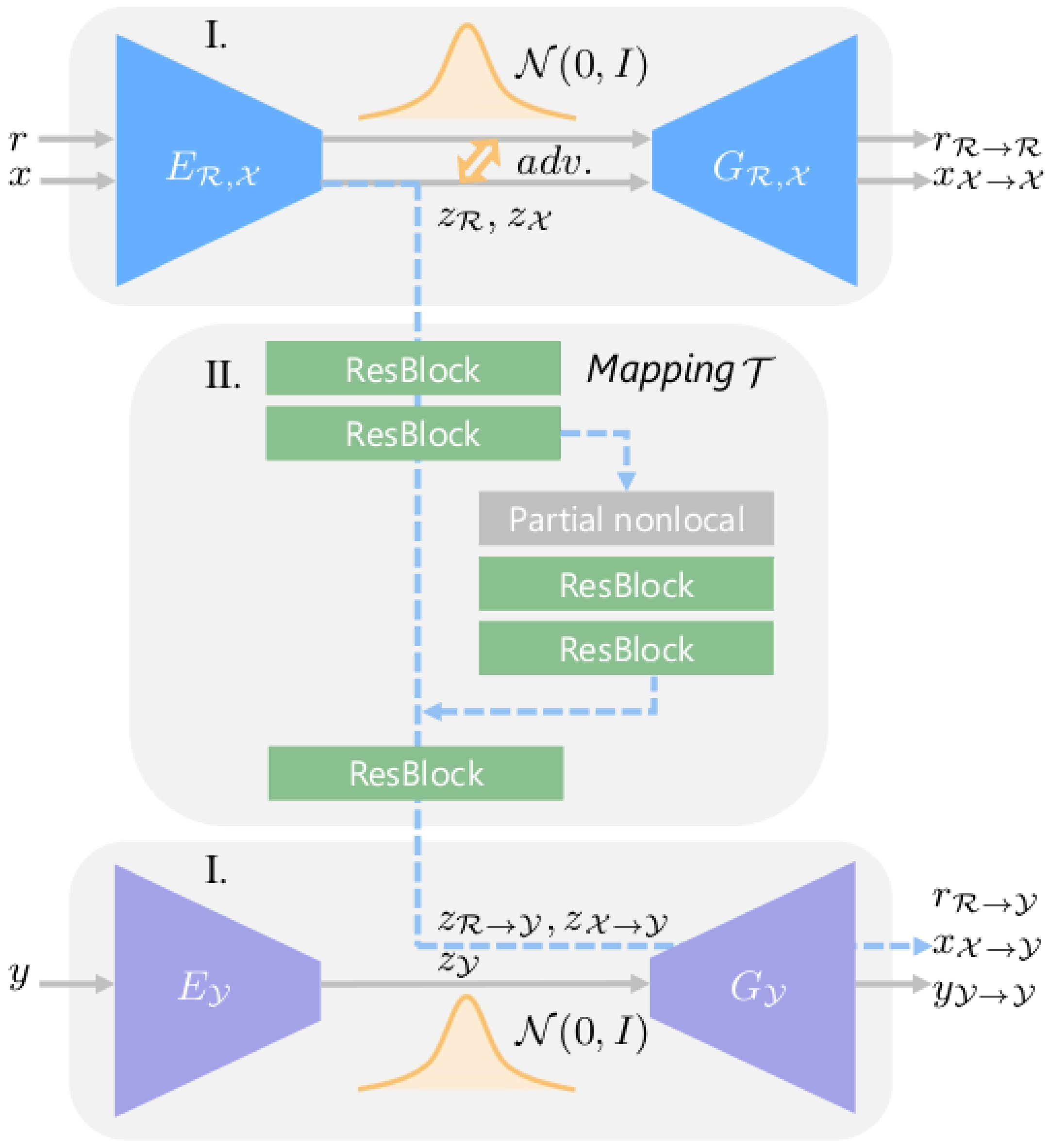
Figure 2: **Illustration of our translation method with three domains.**

1. E R : R 7→ Z R , E X : X 7→ Z X , and E Y : Y 7→ Z Y
2. Z R ≈ Z X
3. This aligned latent space encodes features for all the corrupted images, either synthetic or real ones
4. T Z : Z X 7→ Z Y
5. G Y : Z Y 7→ Y

Result: r R→Y = G Y ◦ T Z ◦ E R (r).

Domain alignment in the variational autoencoder(VAE) latent space

One key of our method is to meet the assumption that R and X are encoded into the same latent space. To this end, we propose to utilize variational autoencoder (VAE) to encode images with compact representation, whose domain gap is further examined by an adversarial discriminator



I.
$\mathcal{N}(0, I)$
r
x
$E_{\mathcal{R},\mathcal{X}}$
adv.
$z_{\mathcal{R}}, z_{\mathcal{X}}$
$G_{\mathcal{R},\mathcal{X}}$
$r_{\mathcal{R}\to\mathcal{R}}$
$x_{\mathcal{X}\to\mathcal{X}}$
II.
ResBlock
ResBlock
Mapping $\mathcal{T}$
Partial nonlocal
ResBlock
ResBlock
ResBlock
I.
y
$E_{\mathcal{Y}}$
$z_{\mathcal{R}\to\mathcal{Y}}, z_{\mathcal{X}\to\mathcal{Y}}$
$z_{\mathcal{Y}}$
$\mathcal{N}(0, I)$
$G_{\mathcal{Y}}$
$r_{\mathcal{R}\to\mathcal{Y}}$
$x_{\mathcal{X}\to\mathcal{Y}}$
$y_{\mathcal{Y}\to\mathcal{Y}}$

Multiple degradation restoration

two VAEs are learned for the latent representation.

- Old photos {r} and synthetic images {x} share the first one termed VAE 1 , with the encoder E R,X
  and generator G R,X
- The ground true images {y} are fed into the second one, VAE 2 with the encoder-generator pair {E Y , G Y }

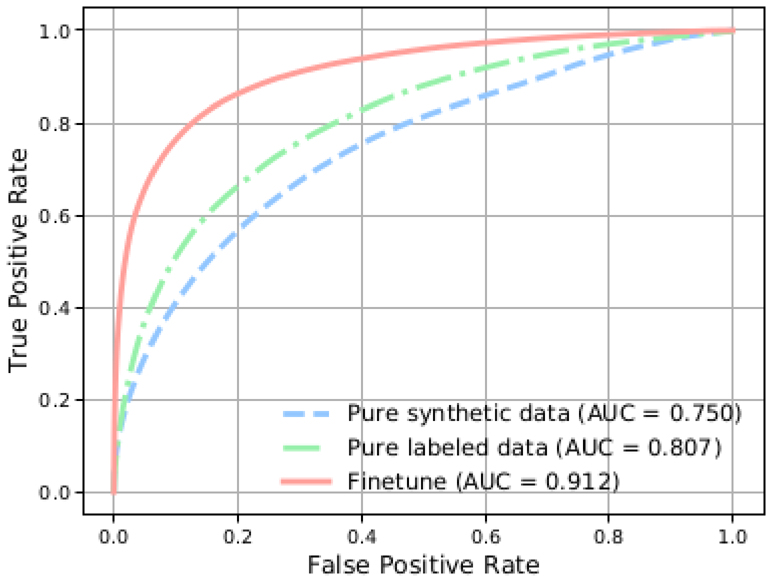
Experiments

Implementation

Training Dataset

- The Pascal VOC dataset
- Collect scratch and paper textures in order to render realistic defects
- use layer addition, lighten-only and screen modes with random level of opacity to blend the scratch textures over the real images from the dataset

Scratch Detection



- To detect structured area for the parital nonlocal block, We train another network with Unet architecture
- The ROC curves on the validation set in Figure 4 show the effectiveness of finetuning. The area under the curve (AUC) after finetuning reaches 0.91

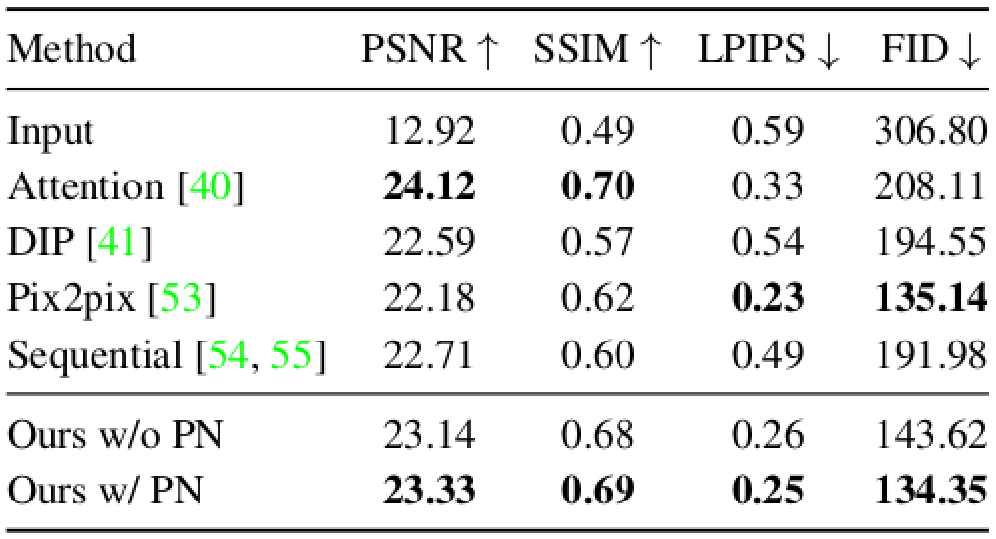
Training Details

- Adopt Adam solver [51] with β 1 = 0.5 and β 2 = 0.999
- The learning rate is set to 0.0002 for the first 100 epochs, with linear decay to zero thereafter
- During training, we randomly crop images to 256×256
- Set in Equations (2) and (5) with α = 10, λ 1 = 60 and λ 2 = 10 respectively.

Comparison

- For fair comparison, train all the methods with the same training dataset (Pascal VOC)
- Test them on the corrupted images synthesized from DIV2K dataset and the test set of our old photo dataset
- The following methods are included for comparison:
    - Operation-wise **Attention**
    - Deep image prior(**DIP**)
    - **Pix2Pix**
    - **CycleGAN**
    - **sequential**ly perform BM3D a classical denoising method, and EdgeConnect

Quantitative Comparison



Method	PSNR ↑	SSIM ↑	LPIPS ↓	FID ↓
Input	12.92	0.49	0.59	306.80
Attention [40]	**24.12**	**0.70**	0.33	208.11
DIP [41]	22.59	0.57	0.54	194.55
Pix2pix [53]	22.18	0.62	**0.23**	**135.14**
Sequential [54, 55]	22.71	0.60	0.49	191.98
Ours w/o PN	23.14	0.68	0.26	143.62
Ours w/ PN	**23.33**	**0.69**	**0.25**	**134.35**

Table 1: **Quantitative results on the DIV2K dataset.** Upward arrows indicate that a higher score denotes a good image quality. We highlight the best two scores for each measure. In the table, PN stands for partial nonlocal block.

- Test different models on the synthetic images from DIV2K dataset
- Adopt four metrics for comparison:
    - peak signal-to-noise ratio (PSNR)
    - the structural similarity index (SSIM)
        - to compare the low-level differences between the restored output and the ground truth
        - these two metrics characterizing low-level discrepancy, usually do not correlate well with human judgment, especially for complex unknown distortions
    - perceptual image patch similarity (LPIPS)
        - which calculates the distance of multi-level activations of a pretrained network and is deemed to better correlate with human perception
    - Fréchet Inception Distance (FID)
        - evaluating the quality of generative models
        - the distance between the feature distributions of the final outputs and the real images

Qualitative Comparison

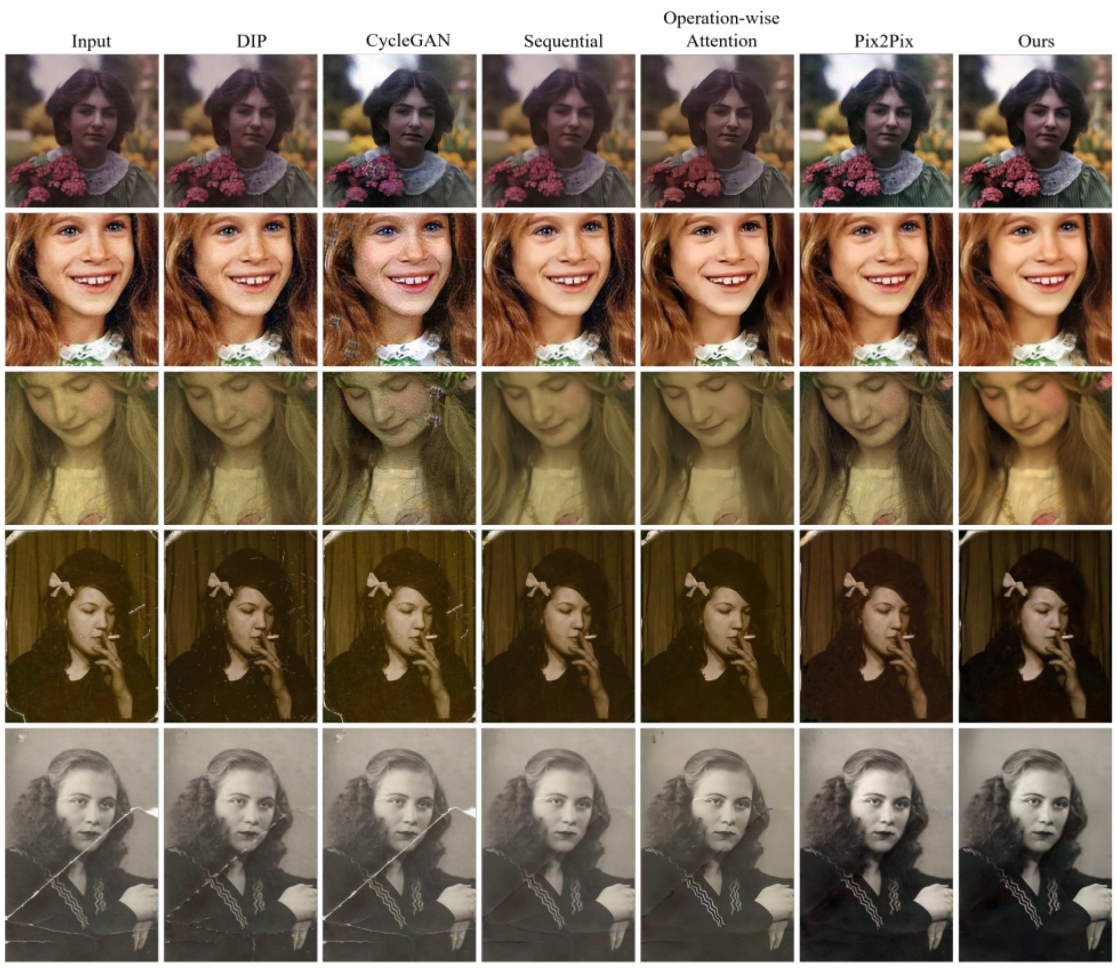


Figure 5: **Qualitative comparison against state-of-the-art methods.** It shows that our method can restore both unstructured and structured degradation and our recovered results are significantly better than other methods.

User study

- Randomly select 25 old photos from the test set
- Let users to sort the results according to the restoration quality

Method	Top 1	Top 2	Top 3	Top 4	Top 5
DIP [41]	2.75	6.99	12.92	32.63	69.70
CycleGAN [42]	3.39	8.26	15.68	24.79	52.12
Sequential [54, 55]	3.60	20.97	51.48	83.47	93.64
Attention [40]	11.22	28.18	56.99	75.85	89.19
Pix2Pix [53]	14.19	54.24	72.25	86.86	96.61
Ours	**64.83**	**81.35**	**90.68**	**96.40**	**98.72**

Table 2: **User study results.** The percentage (%) of user selection is shown.

- It means that this method is about 65% more likely to be chosen as the first rank result.
- But I think it's not turstable data, becuase they conducted it only for 22 users

Ablation Study

In order to prove the effectiveness of individual technical contributions, we perform the following ablation study

Latent translation with VAEs

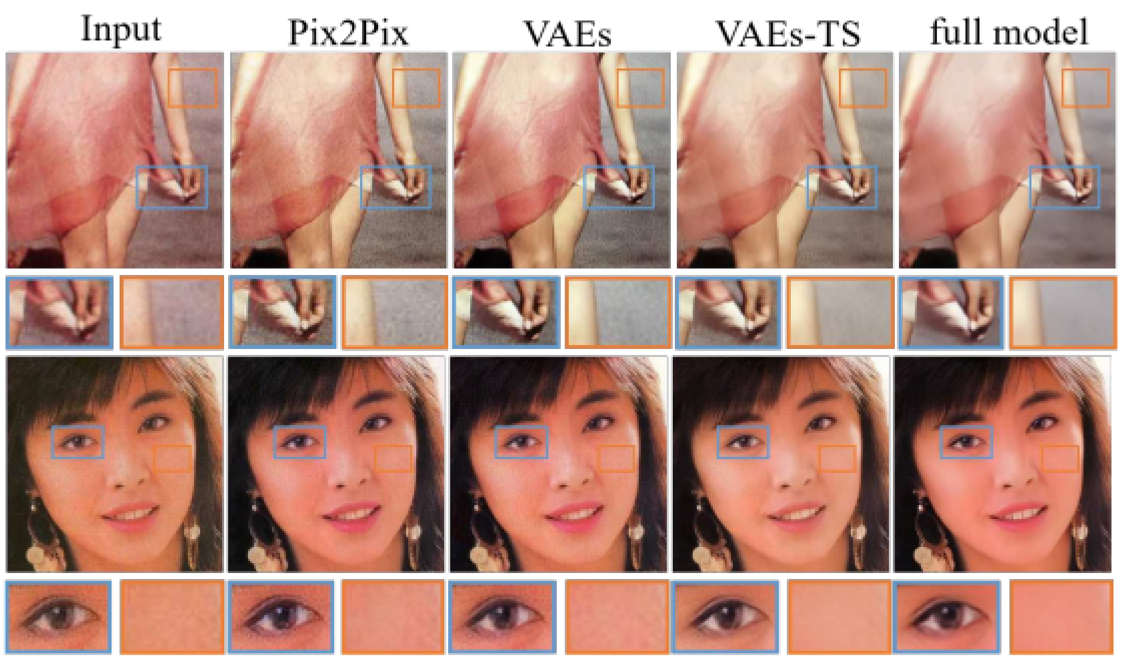


Figure 6: **Ablation study for two-stage VAE translation.**

Method	Pix2Pix	VAEs	VAEs-TS	full model
Wasserstein ↓	1.837	1.048	0.765	**0.581**
BRISQUE ↓	25.549	23.949	23.396	**23.016**

Table 3: **Ablation study of latent translation with VAEs.**

Partial nonlocal block

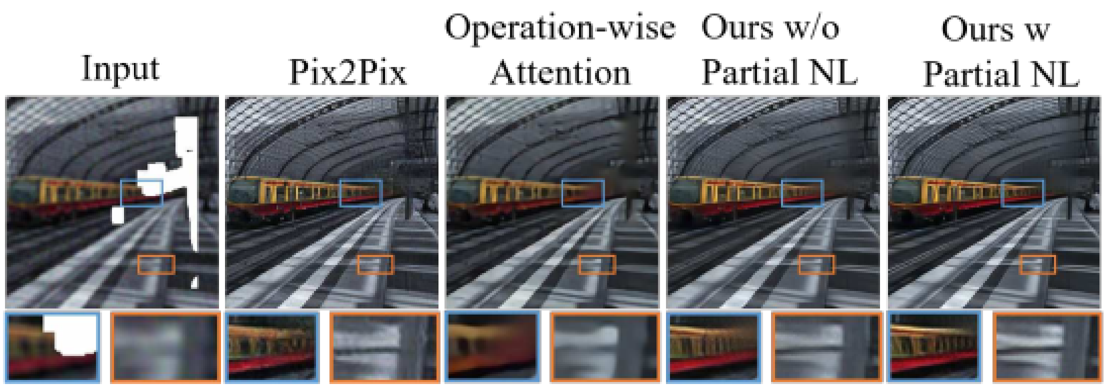


Figure 7: **Ablation study of partial nonlocal block.** Partial nonlocal better inpaints the structured defects.

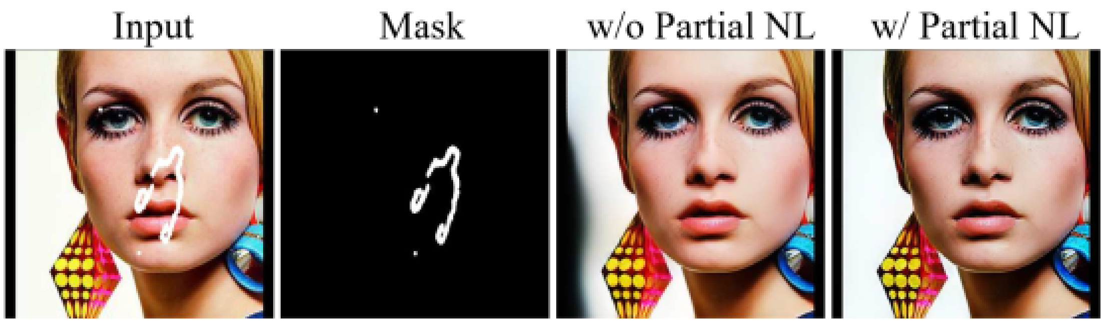


Figure 8: **Ablation study of partial nonlocal block.** Partial nonlocal does not touch the non-hole regions.

Conclusion

- propose a novel triplet domain translation network to restore the mixed degradation in old photos
- The domain gap is reduced between old photos and synthetic images, and the translation to clean images is learned in latent space
- Our method suffers less from generalization issue compared with prior methods
- Furthermore, we propose a partial nonlocal block which restores the latent features by leveraging the global context, so the scratches can be inpainted with better structural consistency.
- Our method demonstrates good performance in restoring severe degraded old photos

limitation

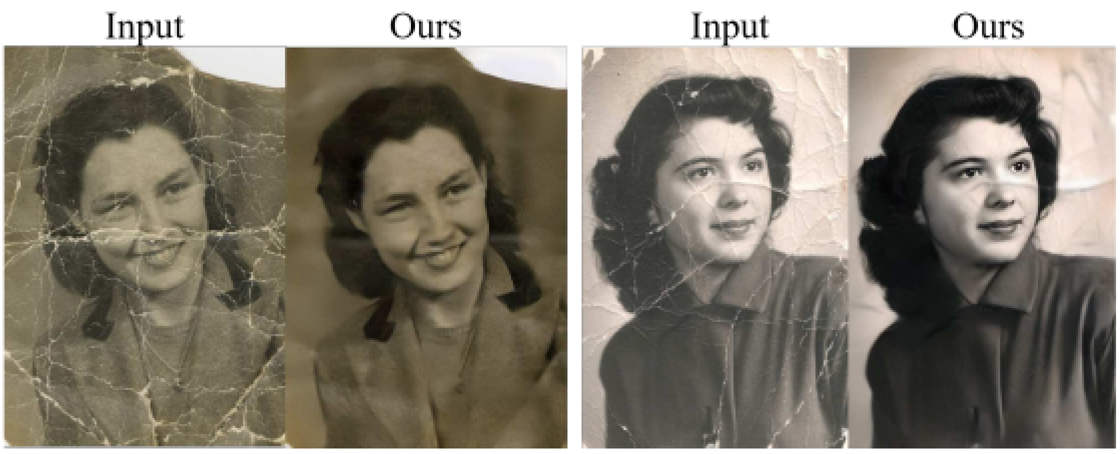


Figure 9: **Limitation.** Our method cannot handle complex shading artifacts.

- our method cannot handle complex shading because our dataset contains few old photos with such defects
- One could possibly address this limitation using our framework by explicitly considering the shading effects during synthesis or adding more such photos as training data

Code

```
## Stage 1: Overall Quality Improve
    print("Running Stage 1: Overall restoration")
    os.chdir("./Global")
    stage_1_input_dir = opts.input_folder
    stage_1_output_dir = os.path.join(opts.output_folder,
"stage_1_restore_output")
    if not os.path.exists(stage_1_output_dir):
        os.makedirs(stage_1_output_dir)

    if not opts.with_scratch:
        stage_1_command = (
            "python test.py --test_mode Full --Quality_restore --test_input
"
            + stage_1_input_dir
            + " --outputs_dir "
            + stage_1_output_dir
            + " --gpu_ids "
            + gpu1
        )
        run_cmd(stage_1_command)
    else:

        mask_dir = os.path.join(stage_1_output_dir, "masks")
        new_input = os.path.join(mask_dir, "input")
        new_mask = os.path.join(mask_dir, "mask")
        stage_1_command_1 = (
            "python detection.py --test_path "
            + stage_1_input_dir
            + " --output_dir "
            + mask_dir
            + " --input_size full_size"
            + " --GPU "
            + gpu1
```

```python
        )
        stage_1_command_2 = (
            "python test.py --Scratch_and_Quality_restore --test_input "
            + new_input
            + " --test_mask "
            + new_mask
            + " --outputs_dir "
            + stage_1_output_dir
            + " --gpu_ids "
            + gpu1
        )

        run_cmd(stage_1_command_1)
        run_cmd(stage_1_command_2)

    ## Solve the case when there is no face in the old photo
    stage_1_results = os.path.join(stage_1_output_dir, "restored_image")
    stage_4_output_dir = os.path.join(opts.output_folder, "final_output")
    if not os.path.exists(stage_4_output_dir):
        os.makedirs(stage_4_output_dir)
    for x in os.listdir(stage_1_results):
        img_dir = os.path.join(stage_1_results, x)
        shutil.copy(img_dir, stage_4_output_dir)

    print("Finish Stage 1 ...")
    print("\n")

    ## Stage 2: Face Detection

    print("Running Stage 2: Face Detection")
    os.chdir("../Face_Detection")
    stage_2_input_dir = os.path.join(stage_1_output_dir, "restored_image")
    stage_2_output_dir = os.path.join(opts.output_folder, "stage_2_detection_output")
    if not os.path.exists(stage_2_output_dir):
        os.makedirs(stage_2_output_dir)
    stage_2_command = (
        "python detect_all_dlib.py --url " + stage_2_input_dir + " --save_url " + stage_2_output_dir
    )
    run_cmd(stage_2_command)
    print("Finish Stage 2 ...")
    print("\n")

    ## Stage 3: Face Restore
    print("Running Stage 3: Face Enhancement")
    os.chdir("../Face_Enhancement")
    stage_3_input_mask = "./"
    stage_3_input_face = stage_2_output_dir
    stage_3_output_dir = os.path.join(opts.output_folder, "stage_3_face_output")
    if not os.path.exists(stage_3_output_dir):
        os.makedirs(stage_3_output_dir)
    stage_3_command = (
        "python test_face.py --old_face_folder "
        + stage_3_input_face
        + " --old_face_label_folder "
        + stage_3_input_mask
        + " --tensorboard_log --name "
        + opts.checkpoint_name
        + " --gpu_ids "
        + gpu1
        + " --load_size 256 --label_nc 18 --no_instance --preprocess_mode resize --batchSize 4 --results_dir "
        + stage_3_output_dir
        + " --no_parsing_map"
    )
    run_cmd(stage_3_command)
    print("Finish Stage 3 ...")
    print("\n")

    ## Stage 4: Warp back
    print("Running Stage 4: Blending")
    os.chdir("../Face_Detection")
    stage_4_input_image_dir = os.path.join(stage_1_output_dir, "restored_image")
    stage_4_input_face_dir = os.path.join(stage_3_output_dir, "each_img")
    stage_4_output_dir = os.path.join(opts.output_folder, "final_output")
    if not os.path.exists(stage_4_output_dir):
        os.makedirs(stage_4_output_dir)
    stage_4_command = (
        "python align_warp_back_multiple_dlib.py --origin_url "
        + stage_4_input_image_dir
        + " --replace_url "
        + stage_4_input_face_dir
        + " --save_url "
        + stage_4_output_dir
    )
    run_cmd(stage_4_command)
    print("Finish Stage 4 ...")
    print("\n")

    print("All the processing is done. Please check the results.")
```