BRINGING OLD PHOTOS BACK TO LIFE

ABSTRACT:

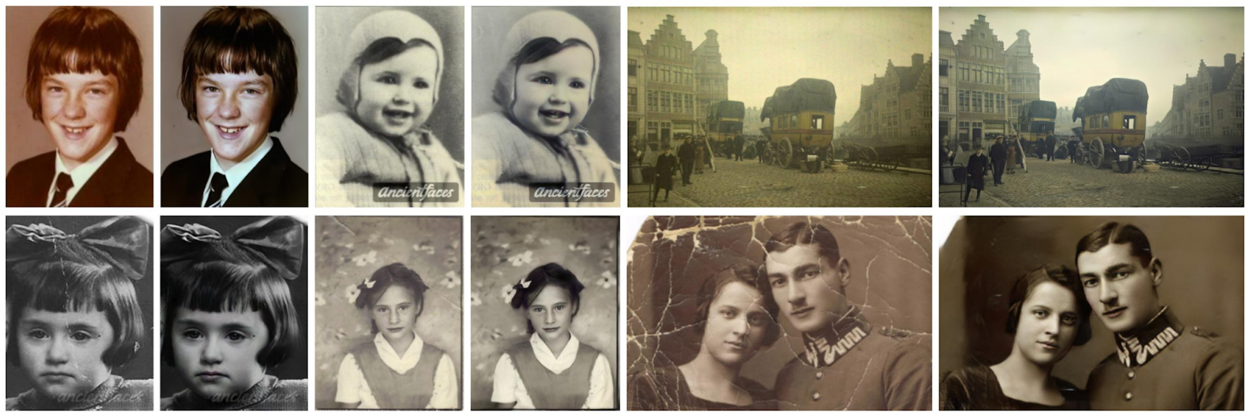


Figure 1: Image restoration done by base model on images from the dataset

Quality of old photo prints degrades when kept in poor environmental conditions. The photo content is permanently damaged due to fading and folding of photos. Manual retouching is usually laborious and time-consuming, which leaves piles of old photos impossible to get restored. Deep learning models usually perform poorly on photo restoration tasks, because the degradation of old photos is a complex process, and there exists no degradation model that can realistically render the old photo artifact. Therefore, the model learned from those synthetic data generalizes poorly on real photos. The base paper restores severely degraded photos through a deep learning approach using a triplet domain translation network consisting of two auto-encoders. This is to generalize the gap with real photos as the domain space is compact.

INTRODUCTION:

We click photos to save memories that we want to cherish. Today we can save them on our devices. However, up until just a couple of decades ago people used to save physical copies of photos on print-paper. Paper as we know deteriorates with time so do the pictures on them, but the memories associated with them don't. Many options are available to restore back to their original state but all of them are not cost efficient and the ones which are, usually provide a poor result.To solve this problem authors have used a deep learning approach using variational autoencoders to restore photos close to their original state. Thus, a new effective method has been devised to provide these restoration tasks in a cost efficient manner that provides results that closely resemble the original photo & eliminates degradation effectively.

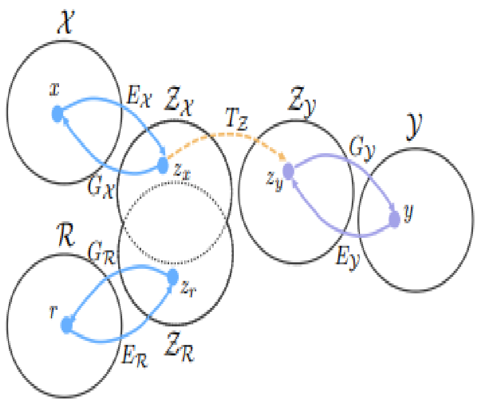
The models rectify the damage on the old photos and try to restore them to their original state. The models have been trained on the same dataset and their performance has been evaluated against one another and the results have been compared in the findings below.

IMPLEMENTATION DETAILS:

Baseline Algorithm

Technology & Tool
Python, GoogleColab

Network Architecture



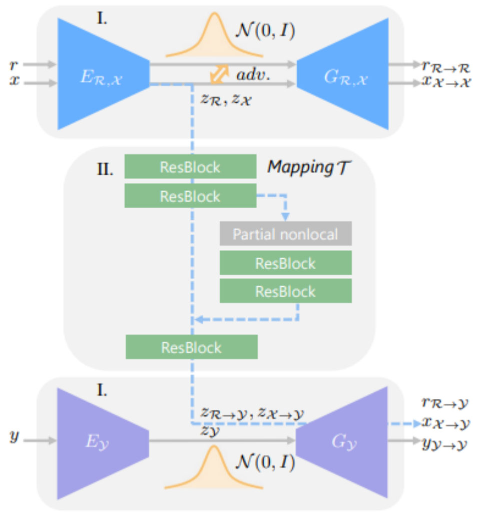

I.
$\mathcal{N}(0, I)$
r
x
$E_{\mathcal{R},\mathcal{X}}$
adv.
$G_{\mathcal{R},\mathcal{X}}$
$r_{\mathcal{R}\to\mathcal{R}}$
$x_{\mathcal{X}\to\mathcal{X}}$
$z_{\mathcal{R}}, z_{\mathcal{X}}$
II.
ResBlock
Mapping $\mathcal{T}$
ResBlock
Partial nonlocal
ResBlock
ResBlock
ResBlock
I.
$z_{\mathcal{R}\to\mathcal{Y}},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

Figure 2&3: Representation of Latent space by base model(left) Model architecture of the base model(right)

The base model makes use of images from 3 domains (Figure 2) - real/degraded(domain R), Ground truth(domain Y) and synthetic images(domain X) that are generated by adding structured/unstructured noise to the ground truth images. The architecture involves the following components :

Variational Autoencoder 1 (Figure 3): This combines the real(R) and synthetic domains(X) into a single latent space (Zr = Zx)

Variational Autoencoder 2 (Figure 3): This is trained on the clean images(Y), and converts them into a second latent space (Zy)

Then, the mapping that restores the corrupted images to clean ones in the latent space is learnt. T:Zx -> Zy.

Finally, generator Gy to restore the clean images from the latent space Zy.

Hyperparameters:
Adam solver [51] with β1 = 0.5 and β2 = 0.999.
learning rate is set to 0.0002 for the first 100 epochs, with linear decay to zero thereafter

Pix2pix

Technology & Tool
Python, GoogleColab, Keras, Tensorflow, Matplotlib, Numpy.

Network Architecture

Conditional GANs are usually used for Image to Image translation applications such as converting black and white images into a color image or converting aerial images into a map. Conditional GANs have similar architecture as baseline GANs. Instead of using a latent noise as an input to the model like GAN, Conditional GANs use some input image and some output image to give to the model. Essentially Conditional GANs not only use the network to learn mapping from input image to output image but also a learning function to train this mapping. This eliminates the need for a hand engineered loss function.

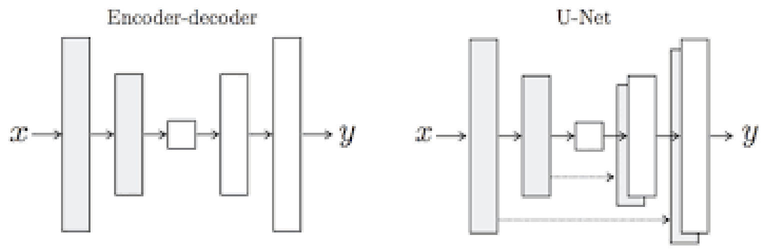


Figure 4: Model architecture of the pix2pix model

Similar to a normal GAN architecture, conditional GAN consists of a Generator and Discriminator model. The generator model is responsible for generating an image, conditional on an input image. This generator model is provided with an input image and it generates a translated version of the image. Whereas, in a discriminator model the generator output and ground truth image is given as an input and it must determine if the image is real or fake. In this project, an input image with scratches and noise was given to the generator model and the model is able to generate a clean image without scratches. In this project, for the generator model a U-net based architecture is implemented(shown in Fig 4) and for the discriminator model a PatchGAN is implemented.

The U-net model comprises two sections: Downsampling (encoder) and Upsampling (decoder). Downsampling involves converting a high-resolution image to a low-resolution image with the goal that the model interprets the object in the image but at the same time losing the spatial context of the object. This is achieved using 7 blocks of convolution operation and batch-normalization layers clubbed together. Since there is a loss of spatial information of the object in the image, upsampling the image is done. Due to upsampling, a low-resolution downsampled image is converted into a high-resolution image. This is done by 6 blocks of the transpose convolution operation.

In the PatchGAN model, it classifies patches of an input image as real or fake, rather than the entire image. This is implemented as a deep convolutional neural network, having 4 layers. The output of the network is a single feature map of real/fake predictions that can be averaged to give a single score.

Hyperparameter

Number of epochs = 6000
Patch size = 70x70
Lambda = 100
Kernel size (generator) = 4x4
Stride (generator) = 2
Batch Size = 10
Learning Rate = 0.0002
Activation Function = LeakReLu (Generator and Discriminator hidden layers), Sigmoid(final layer Discriminator), Tanh (final layer Generator)
Dropout = 0.5

CYCLEGAN:

Technology & Tool:

Google Colab, Python, Tensorflow, matplotlib

Network Architecture:

Generators - UNET (imported from `tensorflow_examples.models.pix2pix`), Discriminators - PatchGAN
The model makes use of 2 generators and 2 discriminators. The first generator (G) is responsible for converting a noisy image(domain - X) to a clean image(domain - Y). The first discriminator (Dy) is trained to distinguish the output image (G(X)) with a real value of Y.
This network contains three convolutions, several residual blocks , two fractionally-strided convolutions with stride 1 2 , and one convolution that maps features to RGB. There are 6 residual blocks for 128 × 128 images and 9 blocks for 256×256 and higher-resolution training images. For the discriminator networks a PatchGANs is used, which aims to classify whether overlapping image patches are real or fake.

Hyperparameter:
No. of epochs = 30
Lambda (controls the significance of the cycle loss) = 8
BUFFER_SIZE = 1000
BATCH_SIZE = 5
Optimizer function : Adam optimizer

Deep Image Prior:
Technology & Tool:
Google Colab, Python, Matplotlib, PyTorch, ResNet, future, OS,

Network Architecture:
Generators - UNET, Discriminator- ResNet

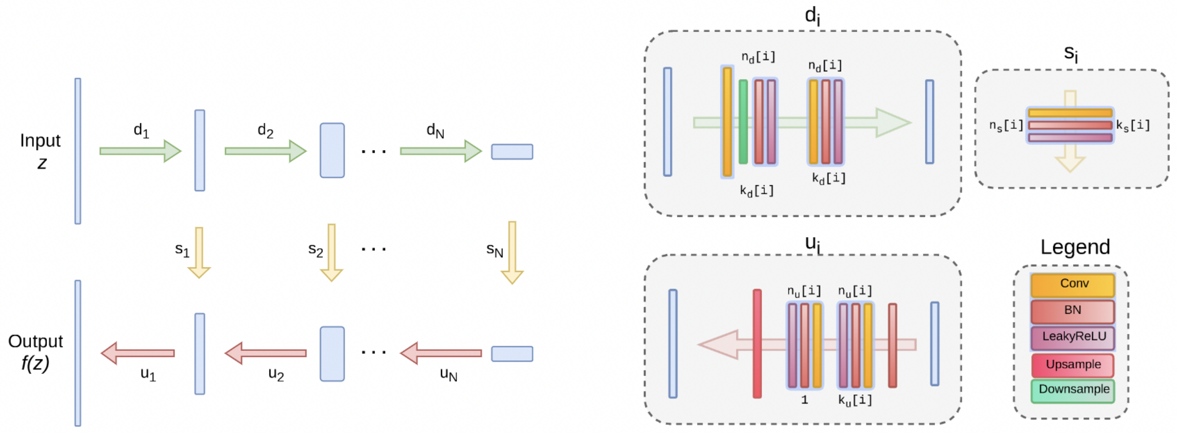


Figure 5: Model Architecture of the Deep Image Prior model

Similar to pix2pix, an hourglass architecture or an encoder decoder architecture with skip connections is used by the deep image prior algorithm. The connections that are skipped have been marked by yellow arrows in the image above. In the right figure, n(i) marks the number of filters at depth i for up sampling, down sampling and skip connections respectively. Each layer in the model has been labeled with a different color corresponding to the layer activation function used. ‘k(i)’ in the image denotes the kernel size of image at depth i respectively.

The encoder-decoder uses LeakyRelu as a non-linearity. Strides are implemented within the convolutional modules. The input image and the mask of the input image are used to train the model. The masks generated by the base paper were used to train the model used in this paper. The trained model was saved and was used to test the image in the results section. Adam optimizer is used for the inpainting tasks.

Hyperparameter

Number of Epochs: 3001
Learning rate: 0.01
Padding: 0
Input depth: 1
Network used: ResNet
Noise Parameter: True
Stride: 1
Activation Function = Conv, BN, LeakyReLu (Generator and Discriminator hidden layers), Sigmoid(final layer Discriminator)

<u>Experiments:</u>

Dataset:
The dataset considered for this project is from the Pascal VOC 2012 dataset. The dataset consisted of approximately 17,000 images. All the images were of different sizes. Out of this dataset we took a smaller subset of 500 images. For training the models we considered 400 images and for validation 100 images. Additional 50 images were taken from the original dataset for testing the models.
Pre-processing the dataset:
Since the authors of the paper did not share the original dataset along with scratches and patches we had to create our own dataset for this project. In order to achieve this, we applied three different methods to create our dataset.

1. Gaussian noise: we added random gaussian noise in the dataset to get the paper texture look in the input dataset.
2. Gray Scale: Random images were converted into gray scale images to consider black and white images.
3. Scratches: Added random lines to replicate real scratches of an image.

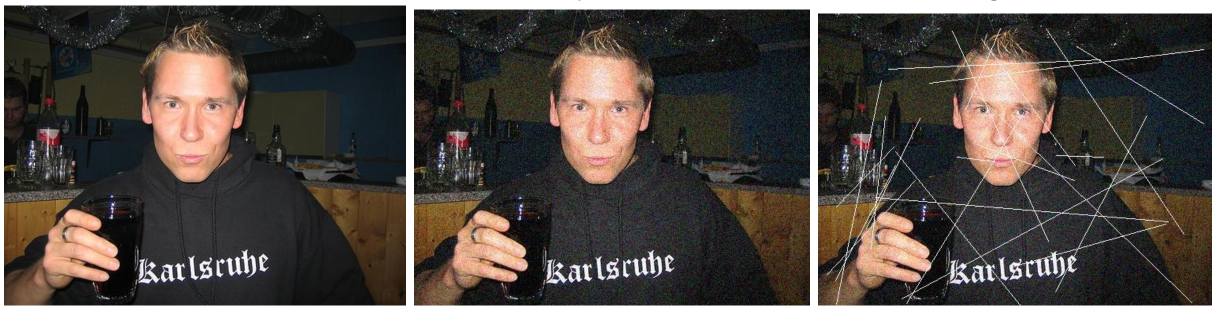


Figure 6: Image augmentation done on Pascal VOC dataset

Methodology:

<u>Pix2Pix:</u>
Two stages were followed to train a Pix2Pix model: training and inference. In the training stage, the synthetic image is given to the U-net model. The synthetic image is of size 256x256x3 with scratches. As it is observed in the figure x, U-net generates and should

generate an output image with no scratches and noise. This generated image is sent to the discriminator along with the ground truth image. The discriminator applies PatchGAN to these and gives a result if the image is real or fake. This particular training takes a lot of epochs to train but due to the simplicity of architecture trains faster. One advantage of cGAN learning faster is because of the use of PatchGAN as a discriminator.

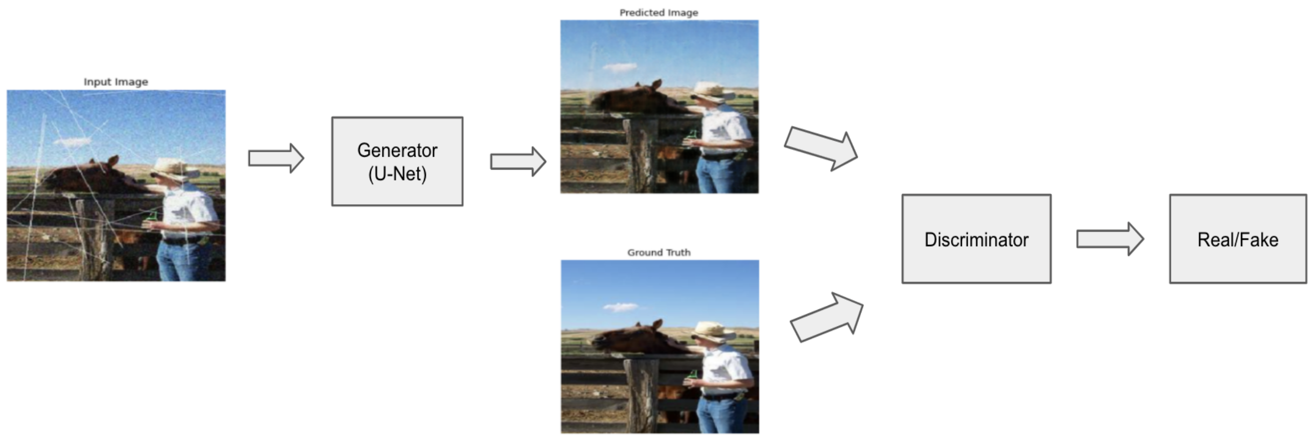


Figure 7: Training of the Pix2pix model

In the inference stage, a random test from the test dataset image is taken and sent through the generator to produce a clean and enhanced image as seen in the figure y.

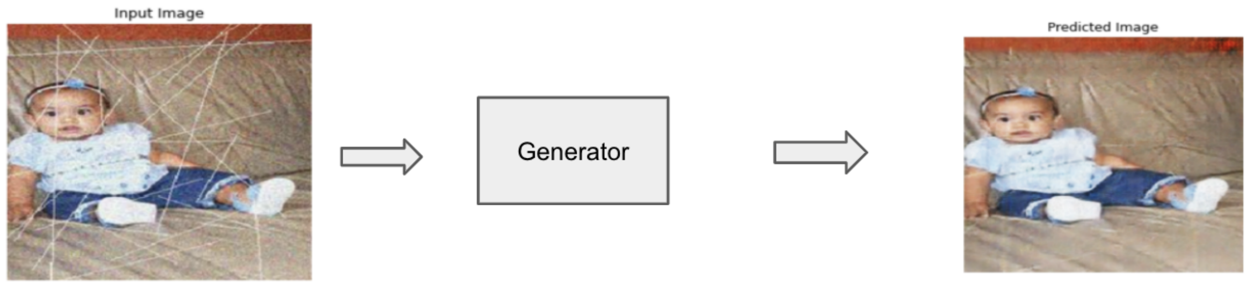
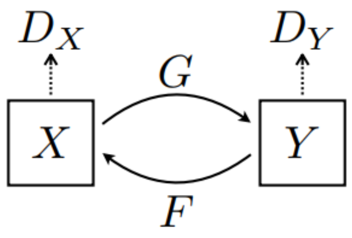


Figure 8: Inference of the pix2pix model

<u>CycleGAN:</u>
This leads us to calculating the first adversarial loss, which is - generator 1 loss + discriminator 1 loss. The second generator (F) is responsible for converting a clean image(domain - Y) to a noisy image(domain - X). The second discriminator (Dx) is trained to distinguish the (clean) output image (F(Y)) with a real value of X. This leads us to calculating the second adversarial loss, which is - generator 2 loss + discriminator 2 loss.



$$\begin{tikzcd} D_X & & D_Y \\ \boxed{X} \arrow[u, dotted] \arrow[rr, bend left, "G"] & & \boxed{Y} \arrow[u, dotted] \arrow[ll, bend left, "F"] \end{tikzcd}$$

Figure 9: Cyclic Gan working

Adversarial training alone can learn G and F that produce outputs identically distributed as target domains Y and X respectively However, with large enough capacity, a network can map the same set of input images to any random permutation of images in the target domain, where any of the learned mappings can induce an output distribution that matches the target distribution. Thus, adversarial losses alone cannot guarantee that the learned function can map an individual input xi to a desired output yi . To further reduce the space of possible mapping functions, the learned mappings should be cycle-consistent.

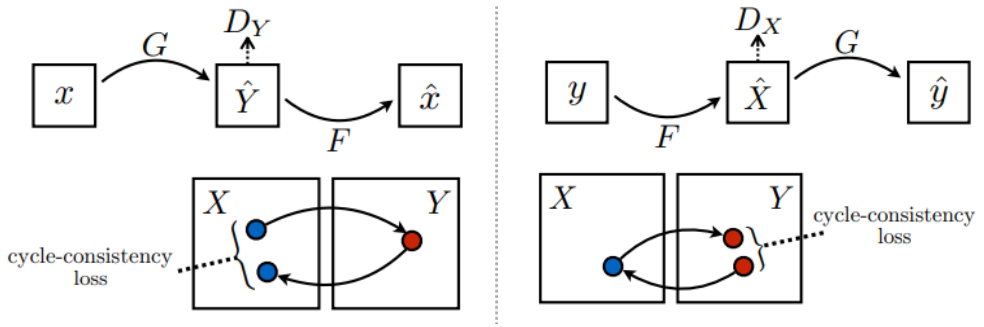


Figure 10: Generators and Discriminators of Cyclic GAN

This requires the need of a second loss function, also known as cycle loss. This is calculated by taking the sum of the forward reconstruction and backward reconstruction loss. Total cycle loss = Abs(F(G(x))-x) + Abs(G(F(y))-y)
Total loss = Adversarial loss + lambda * cycle loss

<u>Deep Image Prior:</u>

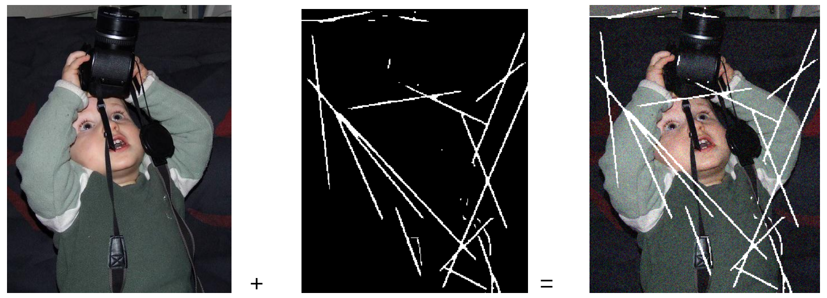


Figure 11: How deep Prior algorithm sees model.

We apply untrained randomly initialized CNN on a single degraded image. The model is capable of Denoising, Super resolution, Inpainting, and image restoration. Corrupted image can be seen as a combination of a pure image and a mask of noise added to it. The architecture of this network will affect the prior distribution in the space. It offers higher impedance to noise and low level image statistics before the unstructured noise.
It will first find a “good looking” local minimum first and then converge to the corrupted image because of over-fitting to the noise. If the optimization is restricted to a certain number of iterations, we can recover the plain image output before over-fitting noise.

$$\theta^* = \underset{\theta}{\operatorname{argmin}}\, E(f_\theta(z); x_0), \qquad x^* = f_{\theta^*}(z)\,.$$

$$E(x; x_0) = \|x - x_0\|^2$$

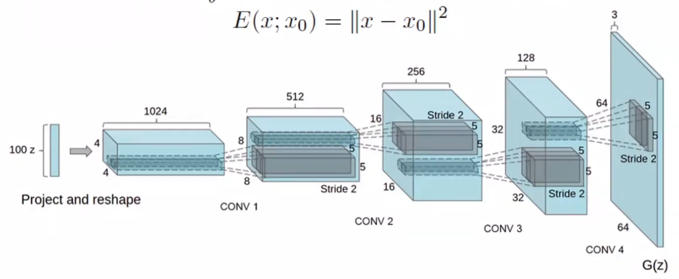


Figure 12: Architecture of DIP decoder

Results and Comparison:

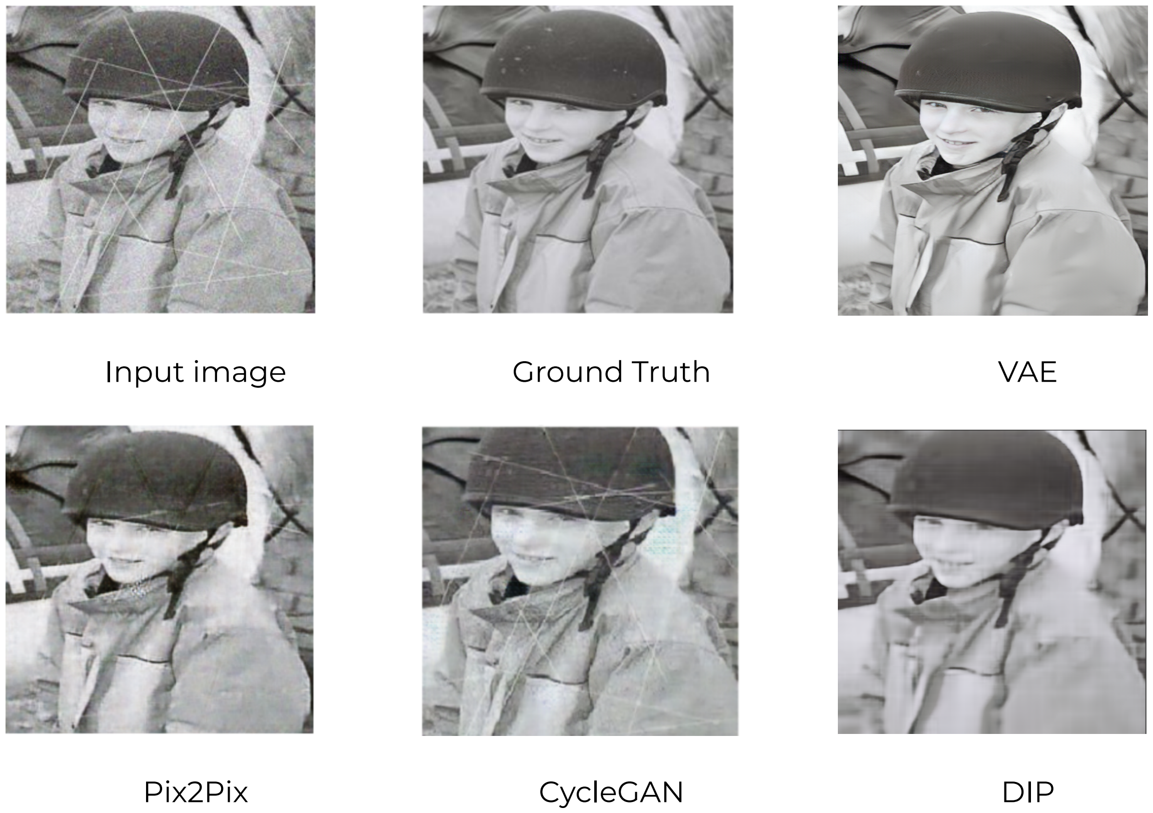


Figure 13: Results of models on the same test image.

CGANS and Pix2pix faired poorly on dataset. VAE performed the best out of models but there was some loss of information. DIP performed second best but wasn't able to upscale the image. CGAN faired the most poorly even though it is based on the pix2pix model.

DISCUSSIONS AND CONCLUSION:

Difficulties faced: The transfer learning documentation on the base paper was not interpretable. FIle structure required to train the Autoencoders was not given due to which we were unable to train the base model. The original dataset used to train the baseline model was not released to the public. Due to this we had to generate our own images. The data thus didn't replicate the tearing degradation that occurs in the images. The scratches are in a very linear manner and don't represent real time folds on the photographs.

Decisions taken: Initially we planned to train three different models mentioned in the paper. Since we were unable to replicate the training of the images due to poor documentation, we decided to train three different models which the base model is compared to. The decision to not add patches was taken after we failed to produce realistic effects.

Things that didn't work: Patches couldn't be added because the deterioration didn't represent natural degradation. Base models' hyperparameters couldn't be tuned because of poor documentation. A fifth sequential planned model couldn't be trained due to issue in the source code. The code was generating a blank data file despite of code being correct. Email has been sent to the repository's owner.

Conclusion: All models were able to restore the image close to ground truth. VAE and DIP models are capable of upscaling image resolution VAE outperforms Pix2Pix, CycleGAN, DIP in terms of scratch removal. VAE was better at detecting and removing unstructured defects from the input data.

REFERENCES:

[1] Ziyu Wan, Bo Zhang, Dongdong Chen, Pan Zhang, Dong Chen, Jing Liao, Fang Wen, et al. Bringing old photos back to life. *arXiv preprint* arXiv:2004.09484, 2020. 1
[2] Dimitry Ulyankov, Andrea Vedald, Victor Lempitsky, Deep Image Prior arXiv preprint arXiv:1711.10925, 2020
[3] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, Alexei A. Efros et al. Image-to-Image Translation with Conditional Adversarial Networks arXiv:1611.07004
[4] Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. https://arxiv.org/pdf/1703.10593.pdf