Realistic Face Restoration with GFP-GAN and DFDNet

ALGORITHM ARTIFICIAL INTELLIGENCE INTERMEDIATE

This article was published as a part of the Data Science Blogathon.

Introduction

Artificial Intelligence has made it possible for machines to learn from experience and adjust to new inputs and perform human-like tasks. The rising popularity of AI apps that can apply cool filters to the human face, edit videos, and create funny deepfakes have become viral on social media. In this article, we will look at one specific neural network architecture that can take old, blurry, and distorted photos of human faces and restore them into near-perfect, realistic images. Several neural network architectures can be used to achieve this – we will look at two of them specifically – GFP-GAN and DFDNet.

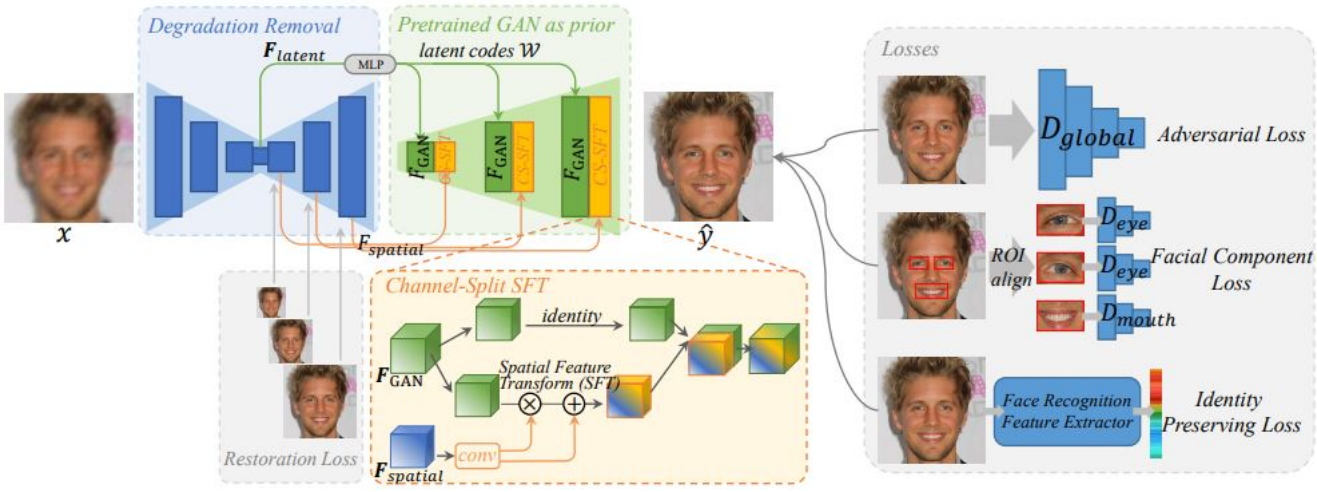
Impact of Blurry Image Restoration

Blurry image restoration can be useful in bringing old photos back to life. Most of us have old, distorted photos of our grandparents which have nostalgic or emotional value, and these images can be restored to high-quality clear images. We can also use recent photos that were blurry or distorted due to picture quality issues or camera movement and restore them into deblurred, high-quality images.

Videos taken with poor quality CCTV cameras can also make it difficult to identify faces in the video. GFP-GAN can be used to restore the person's face from still images captured from CCTV footage.

Let's look at how GFP-GAN works and the architecture behind it.

GFP-GAN Architecture



GFP-GAN Architecture

GFP-GAN consists of GAN pre-trained on Faces and a U-Net module for removing degradation. A latent code mapping and multiple layers of CS-SFT (Channel-Split Spatial Feature Transform) connect these two parts.

The degradation removal module extracts the latent features from the damaged/blurry photos and removes degradation. A pre-trained StyleGAN2 model (generative precursor) uses multi-layer perceptrons to generate style vectors to be used to produce intermediate convolutional features for modulating the final outputs further. The transforms are then predicted with Channel-Split for scaling and displacing feature maps. Finally, the losses are calculated and used to improve the training until the best quality results are achieved.

Some Examples

Let’s look at some example images restored with GFP-GAN and their comparison with other neural networks in the past.

Input
*From **real life***

HiFaceGAN
ACMMM 20

Images restored with GFP-GAN

The images restored by GFP-GAN are thus of the best accuracy and clarity compared to the other model results. Let’s dig deeper into the code for testing GFP-GAN with pre-trained models.

Code Deep Dive – GFP-GAN

Let’s look into the code step by step and try out image deblurring on some random blurred/distorted images.

Setup and Imports

Clone GPF-GAN git repo and install and import the necessary libraries.

```
# Clone GFPGAN %cd /content !rm -rf GFPGAN !git clone https://github.com/TencentARC/GFPGAN.git %cd GFPGAN # Set up the environment # Install basicsr - https://github.com/xinntao/BasicSR # BasicSR for both training and inference !pip install basicsr # Install facexlib - https://github.com/xinntao/facexlib # face detection pkg !pip install facexlib # Install other depencencies !pip install -r requirements.txt !python setup.py develop !pip install realesrgan # enhance background (non-face) regions # Download the pre-trained model # !wget https://github.com/TencentARC/GFPGAN/releases/download/v0.2.0/GFPGANCleanv1-NoCE-C2.pth -P experiments/pretrained_models # V1.3 model !wget https://github.com/TencentARC/GFPGAN/releases/download/v1.3.0/GFPGANv1.3.pth -P experiments/pretrained_models
```

Upload Input Images

If you are using Google Colab for running the code, upload the input images to session storage.

```
# upload images import os from google.colab import files import shutil upload_folder = 'inputs/upload' if os.path.isdir(upload_folder): shutil.rmtree(upload_folder) os.mkdir(upload_folder) # upload images uploaded = files.upload() for filename in uploaded.keys(): dst_path = os.path.join(upload_folder, filename) print(f'move {filename} to {dst_path}') shutil.move(filename, dst_path)
```

Input Image

Generate Deblurred Image

Generate the restored image pixels using the downloaded pre-trained model.

```
# [Real-ESRGAN](https://github.com/xinntao/Real-ESRGAN) improve the background (non-face) regions # different models in https://github.com/TencentARC/GFPGAN#european_castle-model-zoo !rm -rf results !python inference_gfpgan.py -i inputs/upload -o results -v 1.3 -s 2 --bg_upsampler realesrgan !ls results/cmp
```

Visualize the Results

Visualize the deblurred restored image with OpenCV. In Colab, the images are shown side by side with the original input and the restored image.

```
# visualize the restored image import cv2 import matplotlib.pyplot as plt def display(img1, img2): fig = plt.figure(figsize=(25, 10)) ax1 = fig.add_subplot(1, 2, 1) plt.title('Input image', fontsize=16)
```

```
ax1.axis('off') ax2 = fig.add_subplot(1, 2, 2) plt.title('GFPGAN output', fontsize=16) ax2.axis('off') ax1.imshow(img1) ax2.imshow(img2) def imread(img_path): img = cv2.imread(img_path) img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB) return img # displaying each image in the upload folder import os import glob input_folder = 'inputs/upload' result_folder = 'results/restored_imgs' input_list = sorted(glob.glob(os.path.join(input_folder, '*'))) output_list = sorted(glob.glob(os.path.join(result_folder, '*'))) for input_path, output_path in zip(input_list, output_list): img_input = imread(input_path) img_output = imread(output_path) display(img_input, img_output)
```

GFP-GAN Result

The result is highly accurate and the generated image has been restored with little to no blur.

Now let’s look at DFDNet architecture and the quality of results generated by it and compare it with GPF-GAN.

DFDNet Architecture

DFDNet Architecture

DFDNet can be divided into 2 parts – the breakdown and the restoration module.

The breakdown module trains on a certain number of HQ images with varying facial contours, shapes, and postures. K-means algorithm is used for clustering each part of the face (eyes, ears, etc.).

The restoration module uses the dictionary Feature Transform block to guide the restoration process and generate the result.

Some examples

The following are some examples of images restored with DFDNet:

DFDNet Results

Code Deep Dive – DFDNet

Now let's deep dive into the code and test the result on the same image we restored earlier using GFP-GAN.

Setup and Imports

Clone the DFDNet git repo and import the necessary libraries. Create some folders in Colab session storage and copy the checkpoints and weights from the mentioned link.

```
!git clone https://github.com/csxmli2016/DFDNet.git
```

```
#Link - https://drive.google.com/drive/folders/1bayYIUMCSGmoFPyd4Uu2Uwn347RW-vl5 !mkdir /content/DFDNet/checkpoints/ !mkdir /content/DFDNet/checkpoints/facefh_dictionary !mkdir /content/DFDNet/weights/ !mkdir /content/DFDNet/DictionaryCenter512/
```

```
from google.colab import drive drive.mount('/gdrive')
```

```
!cp '/gdrive/My Drive/DFDNet/checkpoints/facefh_dictionary/latest_net_G.pth' '/content/DFDNet/checkpoints/facefh_dictionary' !cp '/gdrive/My Drive/DFDNet/weights/vgg19.pth'
```

```
'/content/DFDNet/weights/' !cp -r '/gdrive/My Drive/DFDNet/DictionaryCenter512/' '/content/DFDNet/'
```

!pip install dominate

```
!python setup.py install
```

Load and Test DFDNet on Input Images

Upload the input images to session storage and generate the deblurred image with DFDNet.

```
import os os.chdir('/content/DFDNet/FaceLandmarkDetection/') !python test_FaceDict.py
```

Results

The result looks fairly accurate with minor blurs still visible on the extremities of the face. As we compare this to images generated with GFP-GAN, the GFP-GAN restored image was of much higher accuracy and contained little to no blur on the face and hence was more realistic.

DFDNet Result

Conclusion

GFP-GAN is one of the most accurate and realistic NN architectures for generating deblurred images and restoring old, damaged images. The results are fairly accurate and more realistic than those generated by DFDNet.

The key takeaways from what we learned:

- GFP-GAN aims at developing a practical Algorithm for Real-world Face Restoration
- It is a highly precise and accurate image restoration neural network
- Other neural networks like DFDNet perform the same job but with lower accuracy
- GFP-GAN is perfect for image restoration with negligible loss because of how its architecture is designed
- GFP-GAN can completely restore blurred, torn, wrinkled images with the highest degree of realism compared to other models

---

Article Url - https://www.analyticsvidhya.com/blog/2022/05/an-introduction-to-realistic-face-restoration-with-gfp-gan-and-dfdnet/

Suvojit Hore