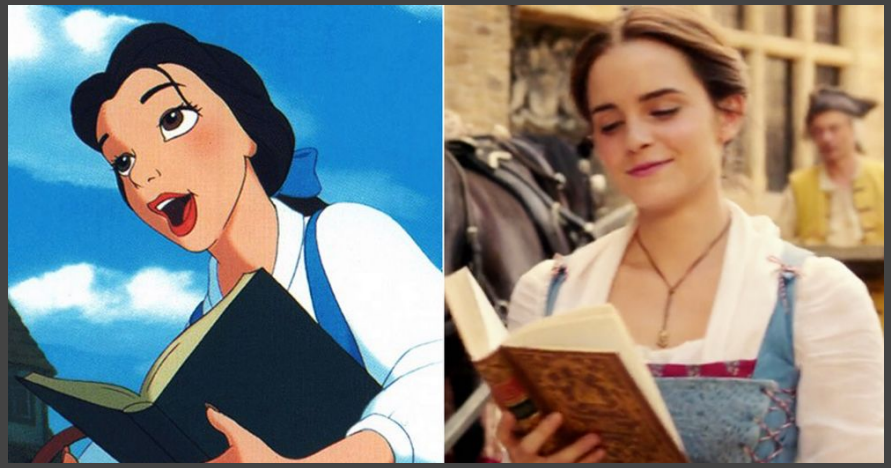
Generating animated variants of real images using GANs

Neural Networks & Applications

Indian Institute of Technology Kharagpur

Problem Statement

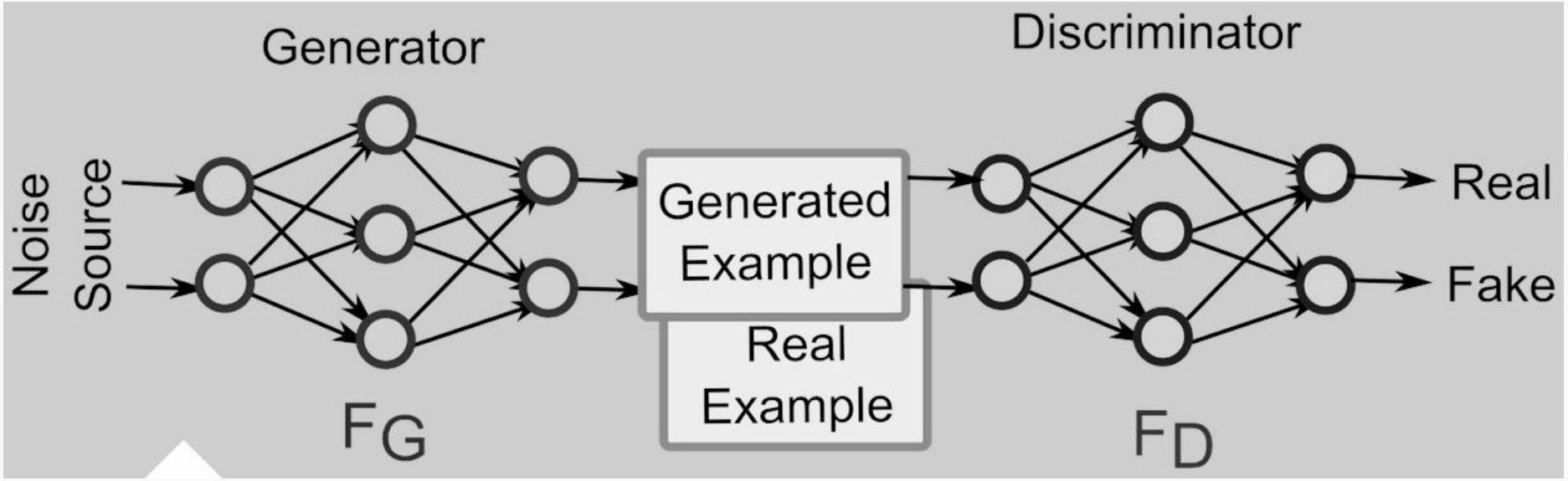
- Generate animated variants of real-world images.
- Possible applications: Use in Computer Graphics rendering for animated characters, etc.



Motivation

- Application in comical printed media and storytelling for education.
- Rendering characters in animated movies take up a lot of time.
- Artists can take ideas from the images generated .

Generative Adversarial Nets



Training a standard GAN is seen as a two-player minimax game with the following value function:

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

Value (Loss) functions in Conditional GANs

- **Conditional GANs:** Both the Generator and Discriminator are conditioned on some extra information. In our case, these are the images from the *CelebA* dataset. The value function is reformulated as:

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x}|\boldsymbol{y})] + \mathbb{E}_{\boldsymbol{z}\sim p_z(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z}|\boldsymbol{y})))].$$

- The minimax game can be visualized as just a **minimization problem** by defining separate losses for *Generator* and *Discriminator*.

Discriminator Loss:

$$\zeta_D = -V(G,D)$$

Generator Loss:

$$\zeta_G = V(G,D)$$

- **Not just fool the Discriminator**. Take into account subtle nuances of animated images (*such as predominant edges*) as well as ensure than the generated animated image is “close” to the conditioning real-world image.

Variants of the Vanilla Loss Function

- **Feature/Content (Coarse) Loss:** To retain high-level information. High-Level information extracted using VGGNet. *(Contributes to Generator Loss).*

$$\zeta_f = E_{x,y\sim p_{data}(x,y),z\sim p_Z(z)}[\| VGG_l(y) - VGG_l(G(x,z)) \|_1]$$

- Note: ***x*** is a sample image from the animated dataset. ***y*** is a sample image from real-world dataset. ***z*** is random noise.

- **Edge-enhancing (Fine) Loss:** To ensure the predominance of sharp boundaries in animated images. *(Contributes to Discriminator Loss).*

$$\zeta_E = E_{x\sim p_{data}(x)}[log(1 - D(e(x)))]$$

e(x) is the edge-smoothened version of animated image ***x***.

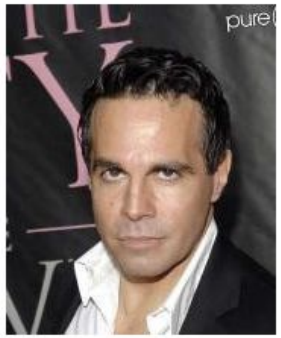
Datasets

Real Images:

Celebrity Attributes Dataset (CelebA)

- Large-scale face attributes dataset (more than 200K images)
- Dimension: 178 x 218

http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html



Animated Images:

Danbooru 2017 Dataset

- Large scale crowd-sourced data (around 300K)
- Dimension: 512 x 512

https://www.gwern.net/Danbooru2017



Training: 6K Images Test: 500 Images

Smoothening the boundaries of Animated Images for Edge Enhancement (Fine) Loss

- Boundaries detected with **Canny Edge** filter on grayscale image.
- Dilate the edge regions.
- Gaussian smoothing in the dilated edge regions.

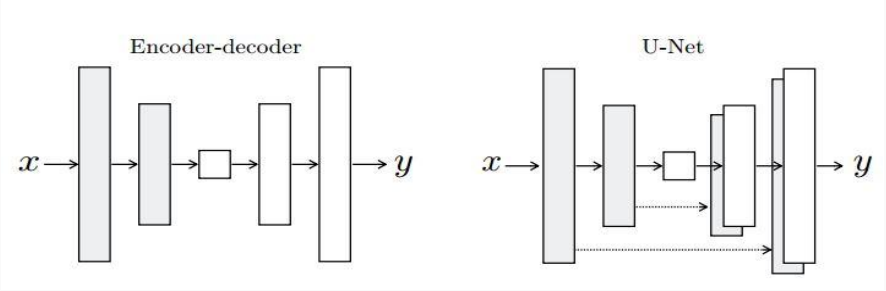
Fig. (a) Original Animated Image

Fig. (b) Smoothened Animated Image

Motivations for the Network Architecture

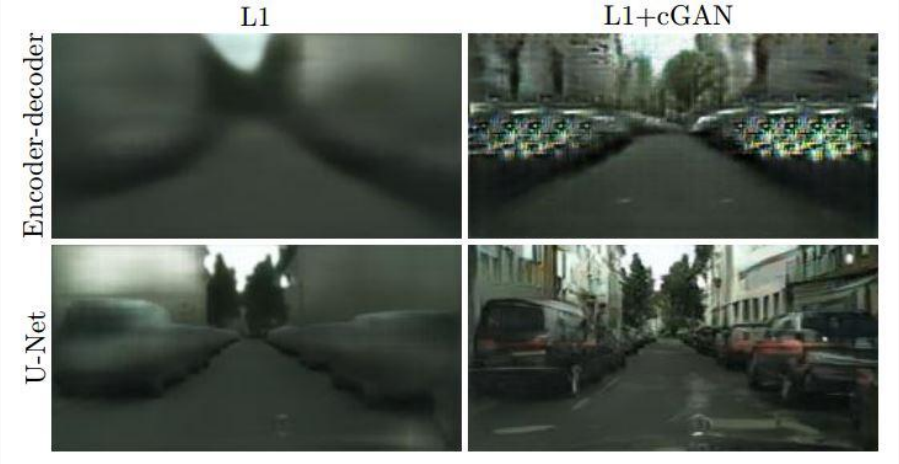
Generator Architecture: **U-Net**

- **cGANs**: Structure of the output is roughly aligned with the structure of the output.
- Standard Encoder-Decoder: Low level information lost. **Skip Connections** directly shuttle it across.



Discriminator Architecture:

- "Markovian" PatchGAN discriminator.
- Instead of classifying the whole image as "real" or "fake", we classify image patches and average the results.
- Patch size: A new hyperparameter!

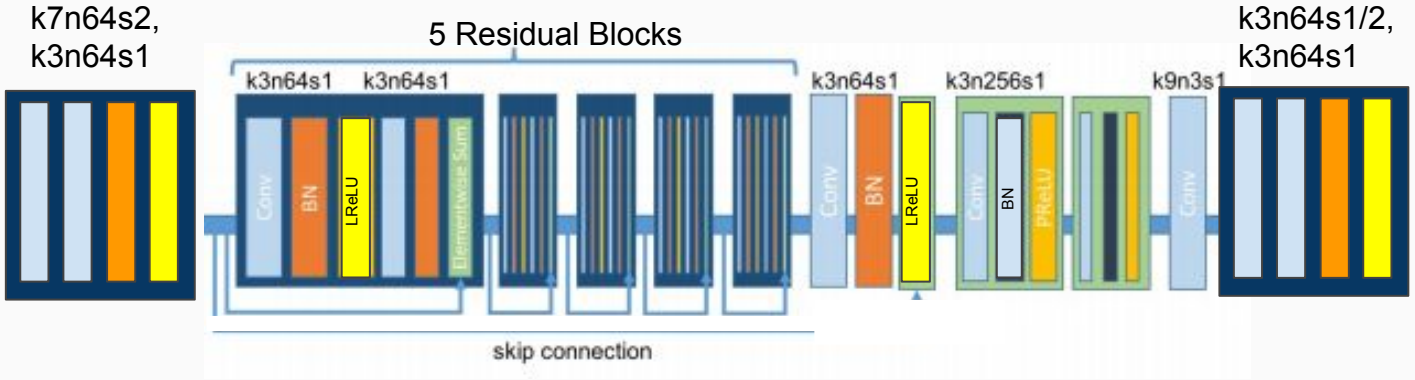


From *Isola et al. Image-to-Image Translation with Conditional Adversarial Networks*

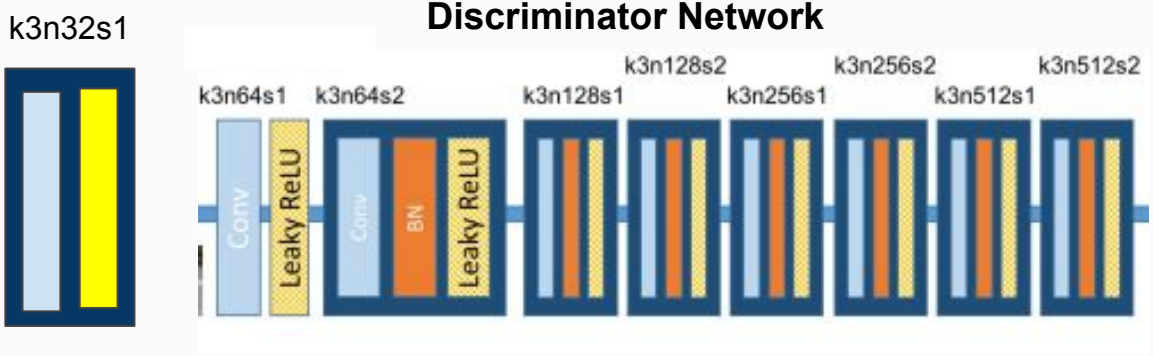
Final GAN Architecture

Generator Network

Input Image



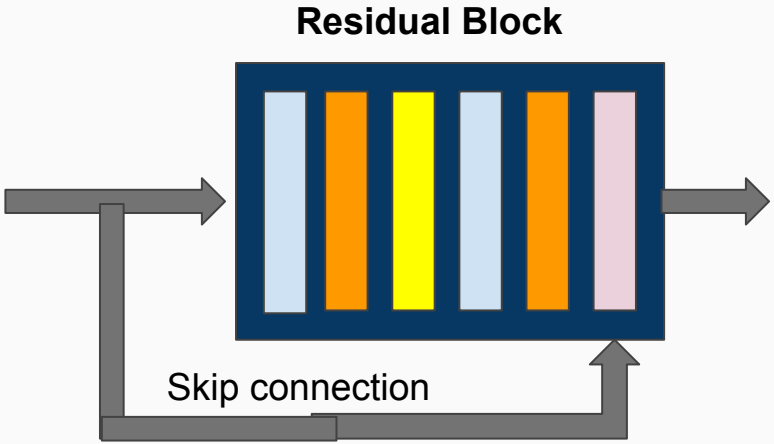
Discriminator Network



$p \in S_{data}(p)$

Network motivation: Image Super-resolution GAN, *https://arxiv.org/pdf/1609.04802.pdf*

Residual Block



- 5 Residual Blocks in Generator Network
- Function recasted to F(x) + x form by skip connections
- Counter degradation problem of deep networks as adding more layers leads to high training error.

Residual Block motivation: Deep Residual Learning, Microsoft 2015, https://arxiv.org/pdf/1512.03385.pdf

Training Details & Parameters

Trained on Nvidia Quadro M4000 in Computer Vision Lab server
Training of model occupied 7821 Mb / 8125 Mb GPU memory
Deep Learning Library: Pytorch

- Epochs = 100 (each epoch took around 2 hrs, total ~ 6.5 days of training)
- Training iterations = 6000 per epoch
- Adam Optimizer with learning rate = 0.0001, beta1=0.5, beta2=1
- Fine loss weight = 10
- Input Image dimensions = 256 x 256
- Batch Size = 1 (due to limited GPU memory)
- Checkpoint save frequency = 1000

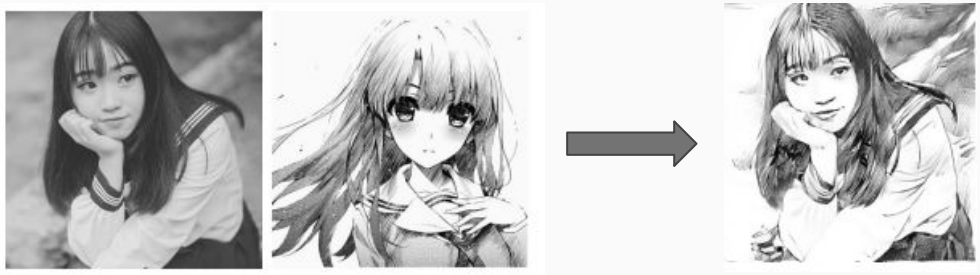
Previous Result: Generated Images for natural image on same CelebA trained GAN

Original Real Image

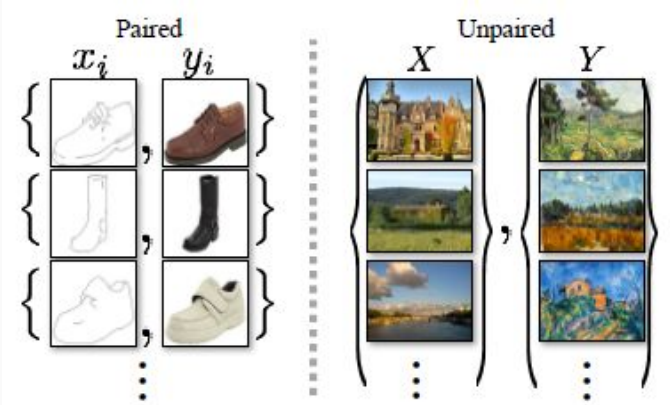
Generated
Transformed Image
Epoch No. = 10

Generated
Transformed Image
Epoch No. = 30

Using Unpaired Images Dataset: Learn distributions and not specific image style



Content Image | Style Image | Transferred Style



- Initial methods of Neural Style Transfer combines style of one image with another.
- Unpairing of dataset in GANs learn mapping between two image collections (distributions) rather than between two specific images.
- Currently limited paired datasets freely available
- Pairing requires manual work which is expensive

Neural Style Transfer: Transforming photos to comics using CNN, ICIP 2017
https://orca.cf.ac.uk/100937/1/transforming-photos-comics-ICIP2017.pdf
Unpaired Data motivation: Unpaired Image to Image Translation using CycleGAN
https://arxiv.org/pdf/1703.10593.pdf

Final Results: Generated Images

Testing on Portrait Image

Original Real Image

Generated Transformed Image
(Epoch = 100)

Final Results: Generated Images

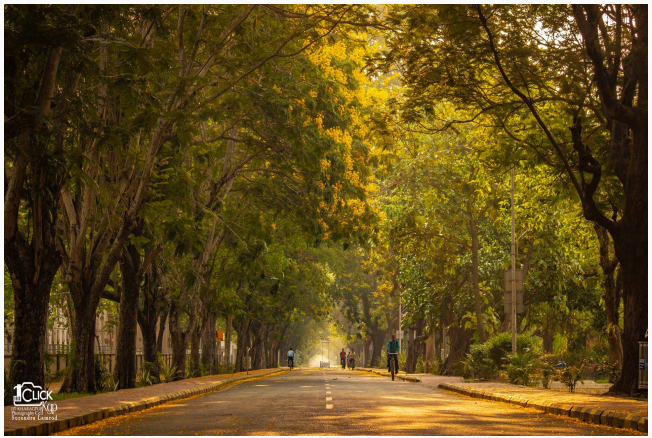
Testing on Indoor Image

Original Real Image

Generated Transformed Image
(Epoch = 100)

Final Results: Generated Images

Testing on Outdoor Image



Original Real Image

Generated Transformed Image
(Epoch = 100)

Final Results: Collective Generated Images

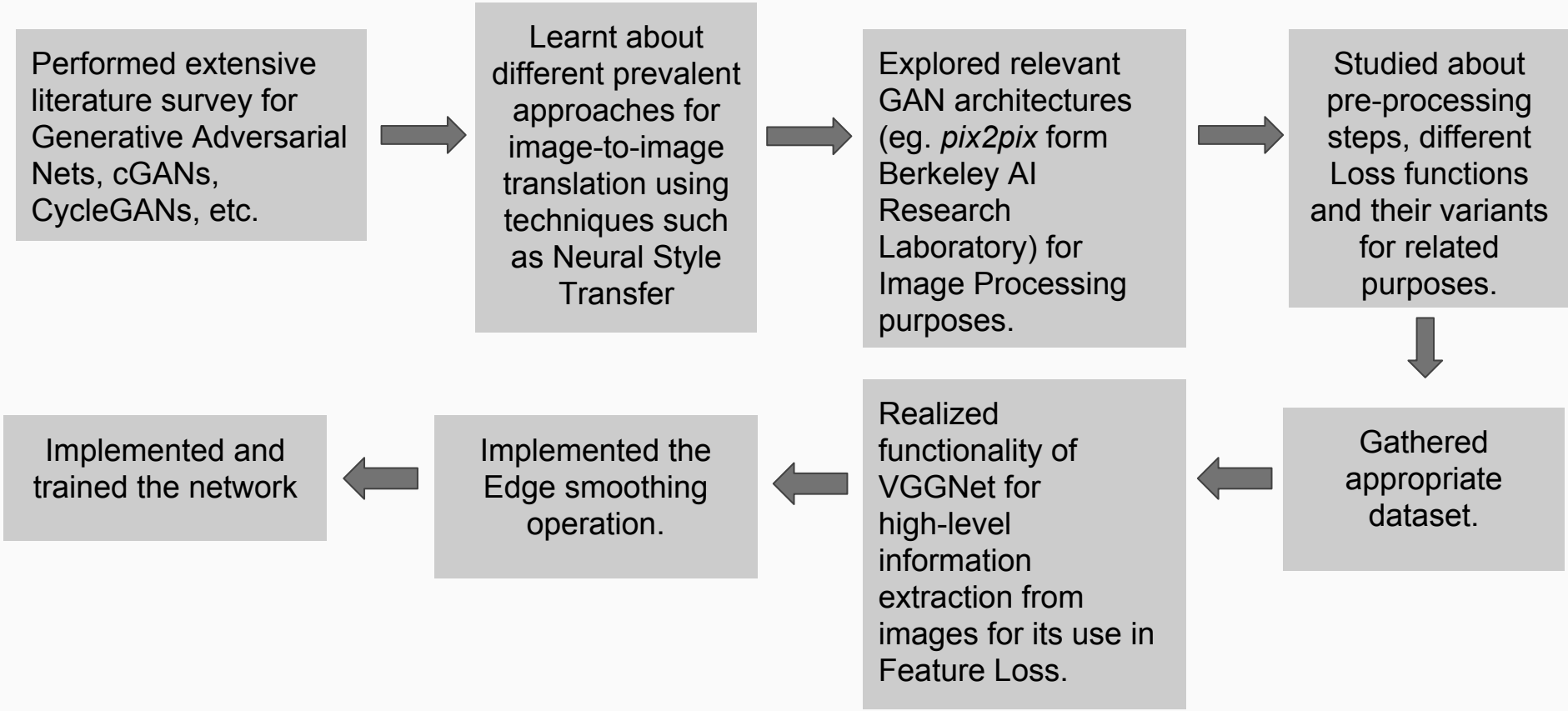
Original Real Images

Generated Transformed Images

Conclusion

- Training of GAN requires a lot of hyper-parameter tuning for the GAN to learn.
- GANs can be successfully used to transform real life images to their animated versions and thus, can serve a great potential in the animated film industry.
- Loss function plays a significant role in guiding the output of generated images in GAN. Here, edge-enhancing loss (fine) to discriminator and content loss (course) to generator give best output.

Roadmap


Performed extensive literature survey for Generative Adversarial Nets, cGANs, CycleGANs, etc.
Learnt about different prevalent approaches for image-to-image translation using techniques such as Neural Style Transfer
Explored relevant GAN architectures (eg. *pix2pix* form Berkeley AI Research Laboratory) for Image Processing purposes.
Studied about pre-processing steps, different Loss functions and their variants for related purposes.
Gathered appropriate dataset.
Realized functionality of VGGNet for high-level information extraction from images for its use in Feature Loss.
Implemented the Edge smoothing operation.
Implemented and trained the network

Individual Contributions

- **Anjali Yadav (14EC35022):** Smoothened the boundaries of animated images on Canny edge detected boundaries for edge enhancement loss, selected & scaled the real image dataset (CelebA) and animated images dataset (Danbooru-2017)

- **Prerit Gupta (14EC35001):** Selected and finalized the architecture of GAN by making modifications in the initial layers of super-resolution network (Chrisitan et al.), coded in PyTorch, studied the effect of different loss functions in the output, saved initial model of stabilized generator & discriminator

- **Rajarshi Saha (14EC35030):** Literature Survey: Image Generation using cGANs (Yifan Liu et al.), Image-to-Image Translation using cGANs (Philip Isola et al.); Introduced *Skip Connections, Content Loss and Edge-Enhancing Loss.* Preprocessing: Smoothed edges in cartoon images using *Canny Edge Detection*.

- **Ritwika Chowdhury (14EC35006):** Designed the problem statement and did literature survey, analyzed different loss functions, and network architectures, finalized the GAN architecture and loss functions. Implemented the network in PyTorch.

- **Yash Agarwal (16IE10032):** Performed the training of GAN and tuning of the hyper-parameters like discriminator critical iterations, learning rate, etc., found final transformed generated images of different scenes at IIT Kgp, performed installations and setup for the code.

References

[1] *Transforming photos to comics using Convolutional Neural Network, IEEE ICIP 2017*
https://orca.cf.ac.uk/100937/1/transforming-photos-comics-ICIP2017.pdf

[2] *CycleGAN:Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, arXiv 2018*
https://arxiv.org/pdf/1703.10593.pdf

[3] *Photo-Realistic Single Image Super Resolution using a Generative Adversarial Network, arXiv 2017*
https://arxiv.org/pdf/1609.04802.pdf

[4] *Auto-painter: Cartoon Image Generation from Sketch by Using Conditional Generative Adversarial Networks, arXiv 2017*
https://arxiv.org/pdf/1705.01908.pdf

[5] *Towards the High-quality Anime Characters Generation with Generative Adversarial Networks, NIPS 2017*
https://nips2017creativity.github.io/doc/High_Quality_Anime.pdf

- Anjali Yadav (14EC35022)
- Prerit Gupta (14EC35001)
- Rajarshi Saha (14EC35030)
- Ritwika Chowdhury (14EC35006)
- Yash Agarwal (16IE10032)