Image Toonification

Prajval Nakrani, 17D070014
Parth Shettiwar, 170070021
Harekrissna Rathod, 17D070001
Utkarsh Bhalode, 17D070006

Problem Definition

- We take upon the problem of transferring the style of real life images to the domain Anime style Images
- There are various styles of cartoon images and also vary based on quantization levels of artists
- We combined the latest new proved architectures and improve upon them by using state-of-the-art Digital processing techniques
- The problem is a subpart of bigger problem of Stylization

Previous Works

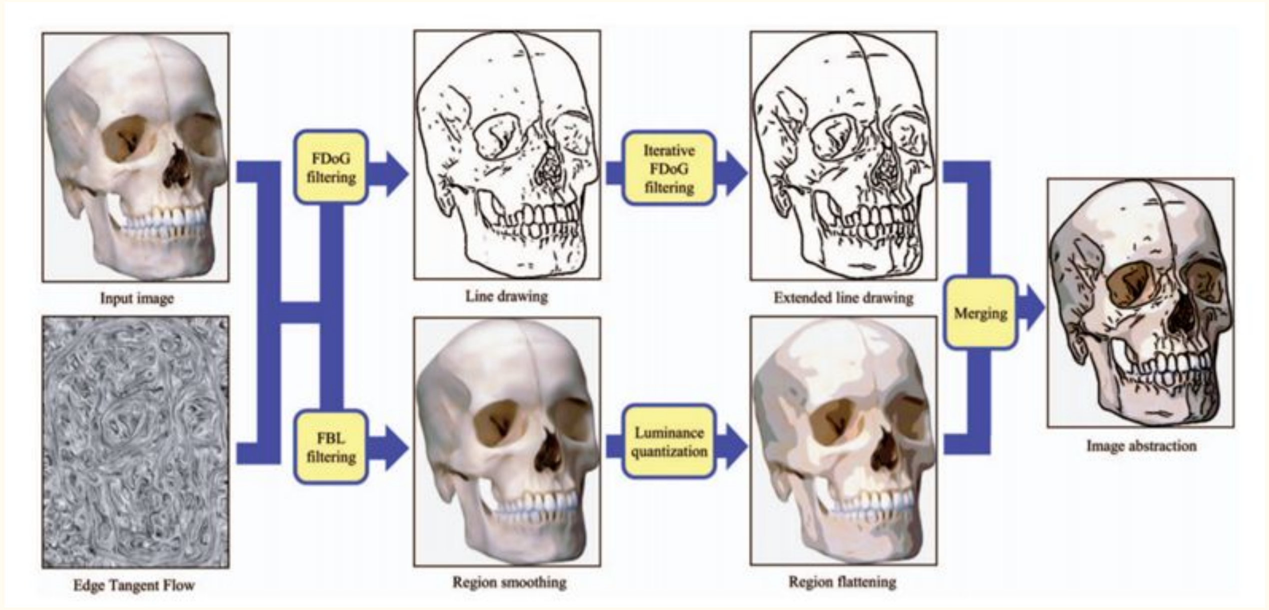
- Cartoonization has been a famous topic in stylization for a long time
- Many Dip based algorithms like Edge Preserving Bilateral Filtering & Flow based Image Abstraction have been implemented which are part of Non-Photorealistic rendering
- Approaches like Cycle Gan are able to transform appearance from real life to cartoon style but lack the quality of images, often missing semantic content of images
- There have also been implementations based on local matching of CNN features maps and using Markov random field fusion

Our Approach

- We take inspiration from the original implementation of Cartoon Gan where Generator is basically like an AutoEncoder with 3 convolution layer,8 residual Blocks, and 3 convolution layers
- Discriminator is a convolution only architecture which comprises of one flat convolution layer, followed by combinations of downsampling and strided convolution
- We have used Instance Normalization rather than Batch Normalization as it gave better results for stylization
- The initialization of Generator is done via training on supervised pairs of real and cartoon images produced using an off-the-shelf DIP algorithm

Initialization Of Generator

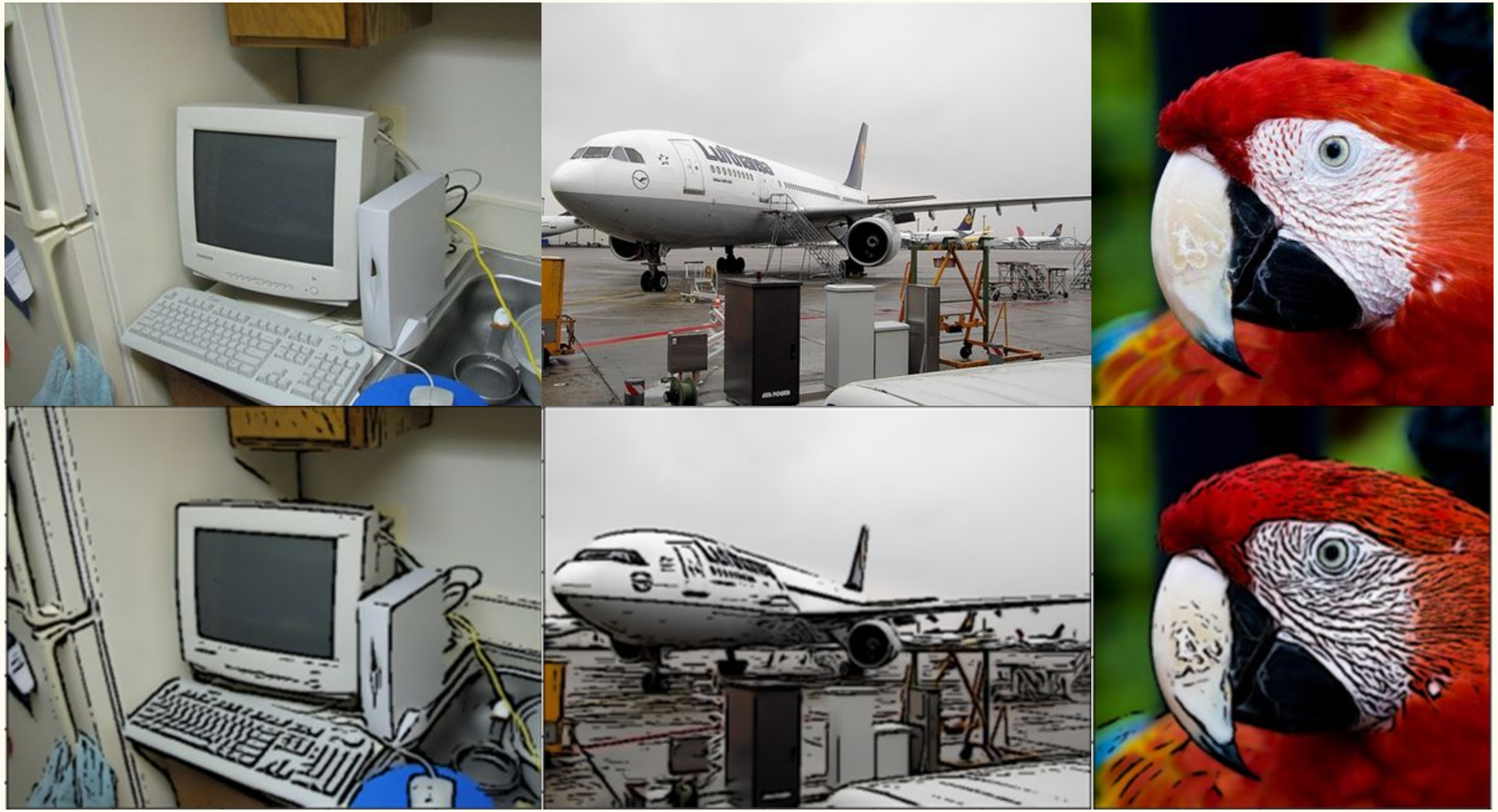
- Flow Based Image Abstraction: ETF, FDoG, FBL



Flow Construction

- A smooth, feature-preserving edge flow field constructed to guide the filters
- **Edge tangent** is a vector perpendicular to the image gradient or tangent to the local edge flow while the vector field is called **edge tangent flow (ETF)**
- Employ bilateral filter for constructing ETF to handle still images and video
- Update iteratively to get smooth ETF keeping gradient magnitude unchanged
- Time complexity can be reduced by separately applying 1D ETF filters in x and y directions too without noticeable quality degradation of the vector field

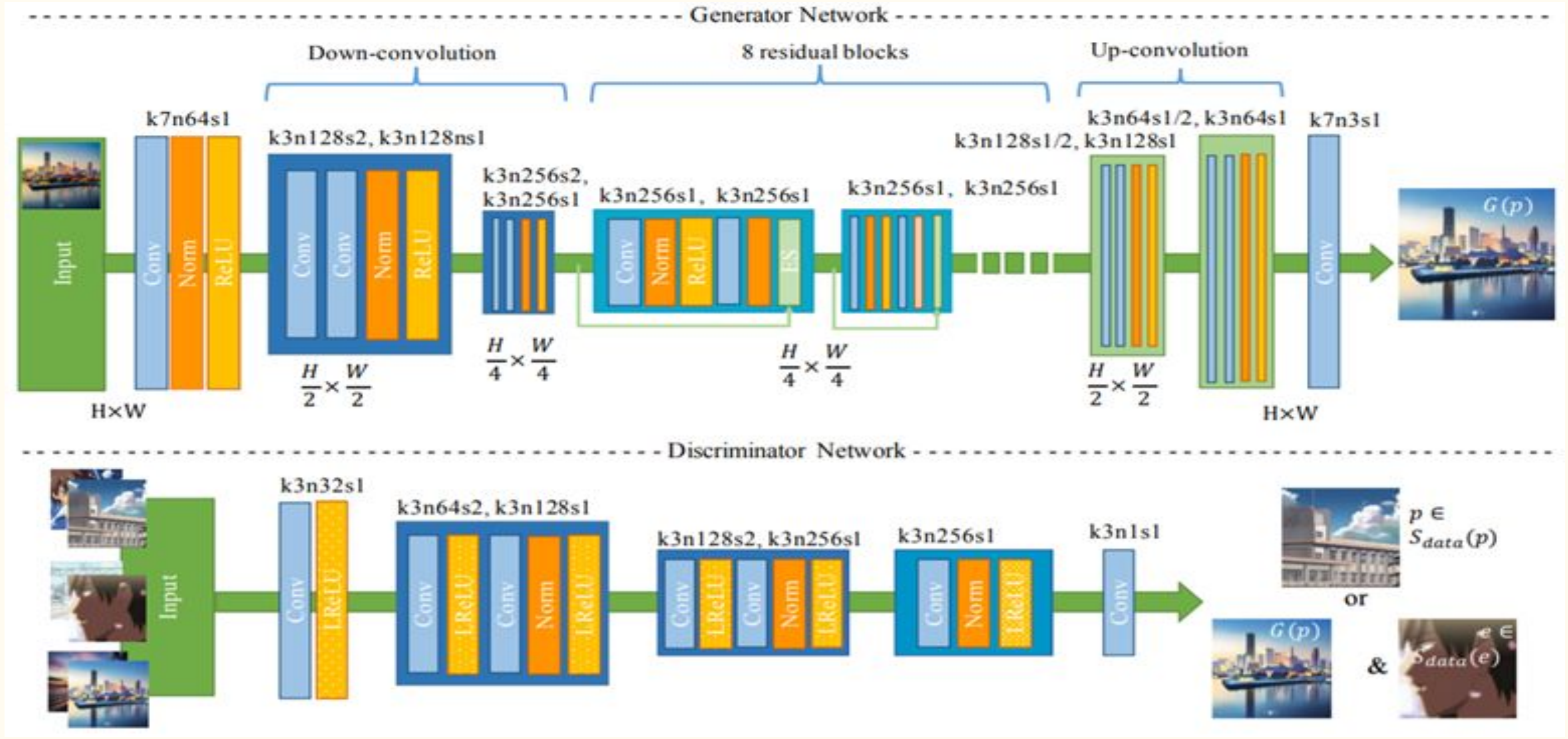
Initialization Phase Training Data



Network and Architecture Details

- Generator is an auto-encoder model with 8 Residual blocks
- Discriminator is a deep Convolutional network with sigmoid output
- BatchNorm is replaced with InstanceNorm for better stylization
- Accelerated performance using GPU hardware
- Initialization of generator is done by training on DIP based cartoon images
- Losses include semantic content loss , modified adversarial loss

Cartoon Gan: Architecture



Losses: Adversarial Loss

- The Discriminator is trained to distinguish between Cartoon manifold and Generated manifold
- Edges being an important aspects of a cartoon image, Discriminator thus should also distinguish between unclear edge images as non-cartoon images
- We use a series of edge smoothing algorithms so that discriminator can be trained

$$\begin{aligned}\mathcal{L}_{adv}(G, D) = \; & \mathbb{E}_{c_i \sim S_{data}(c)}[\log D(c_i)] \\ & + \mathbb{E}_{e_j \sim S_{data}(e)}[\log(1 - D(e_j))] \\ & + \mathbb{E}_{p_k \sim S_{data}(p)}[\log(1 - D(G(p_k)))].\end{aligned}$$

Losses: Content Loss

- For keeping the semantic features of the real images in the generated images, we use high-level features to ensure this
- To get the high level features map, we use a pre trained VGG network to extract features
- The content loss is given as:

$$\mathcal{L}_{con}(G, D) = \mathbb{E}_{p_i \sim S_{data}(p)}[||VGG_l(G(p_i)) - VGG_l(p_i)||_1]$$

Net loss is the summation of both losses: $\mathcal{L}(G, D) = \mathcal{L}_{adv}(G, D) + \omega \mathcal{L}_{con}(G, D),$

Losses & Hyper-parameters

- The Adversarial Loss helps to cartoonize the image, while the content loss preserves semantic information
- Using Edge smoothened images, and the loss component of edge smooth image, helps generator to do edge preserving
- Discriminator learns based on adversarial loss, while generator also uses content loss to learn
- Adam Optimiser helps for training, pre trained VGG are used for calculating content loss.
- We also used Resnet inplace of VGG and achieved similar performance

Results

Results

Noisy Image Stylization

Introduction

- The Image Toonification algorithm that we have described previously takes clean real real images as input and tries to produce corresponding cartoon versions of the input images.
- However, in real life we might not always have a clean version of the input real image. The input image might very well be corrupted with noise.
- We aim to solve such a problem of cartoon stylization of a noisy, real life input image.

Problem Statement

- For sake of simplicity, we assume a more specific problem of missing pixel blocks in the input real image like in the Image Inpainting task
- Basically, we have an input real image that has missing pixel information in form of blocks.
- The required output of the network is clean toonified image of the corrupted real input image.

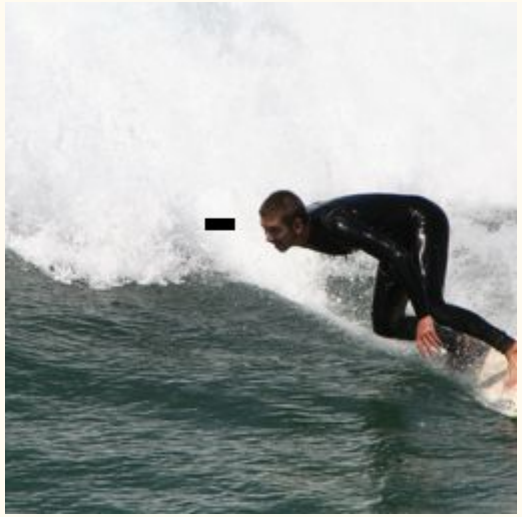
Cartoon Gan for Inpainting

- To solve this problem of creating clean cartoon images corresponding to noisy real input images.
- That is, we try to recreate masked patches in a cartoon image using our Cartoon Gan implementation
- This allows one to recreate the masked part of an output cartoon image which is corrupted and try to recreate it using the closest latent vector representation as described in the next slide.

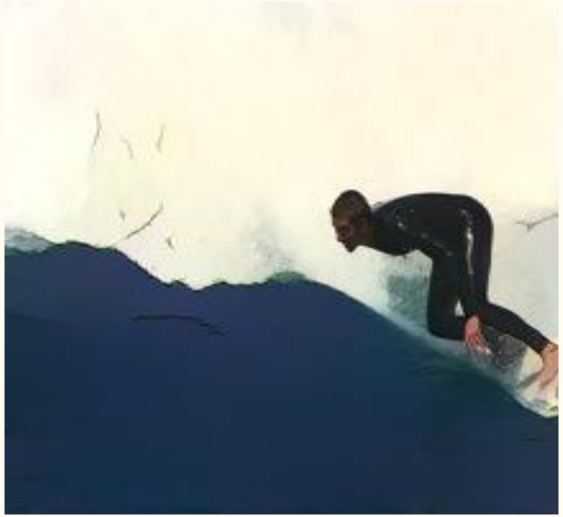
Cartoon Gan Inpainting: Implementation

- The way of using the latent vectors is along the lines of Semantic Inpainting with deep Generative models
- The first half (Real Image to Residual Blocks) of Cartoon gan allows one to generate latent features of multiple Real images, which can be obtained from the output of residual blocks
- We take the output of the 8th residual blocks as a substitute for the the latent vectors which are required in Inpainting
- The second half of Generator is used to recreate cartoon image using the appropriate substitute latent vector

Results of Cartoon Gan-Inpainting



Original Real Image



Output after cartoon inpainting
Corrupted Real Image

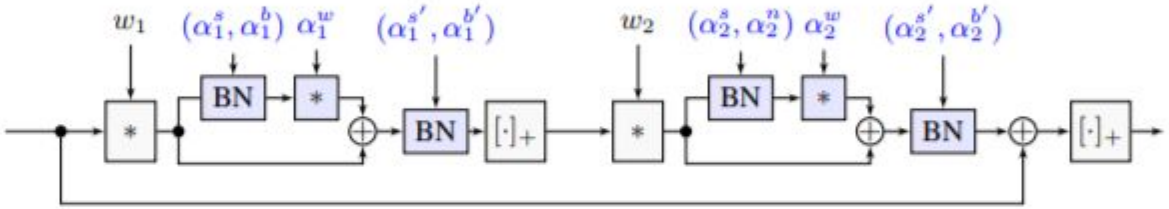
Latent Vector taken from original Cartoon Gan implementation

Learning multiple visual domains with residual adapters

Problem Definition: Multi-domain Learning with Residual Adapters

- We want to make a single network learn classification task of various Domains
- We want to minimize the Domain specific parameters while not compromising the accuracy of classification in each Domain
- Residual Adapters introduced in “Learning multiple visual domains with residual adapters” by Sylvestre.et.al performs very well.
- Introduce Domain specific parameters as Residual adapters in between the network while keeping majority of Domain-agnostic parameters.

Residual Adapters Model



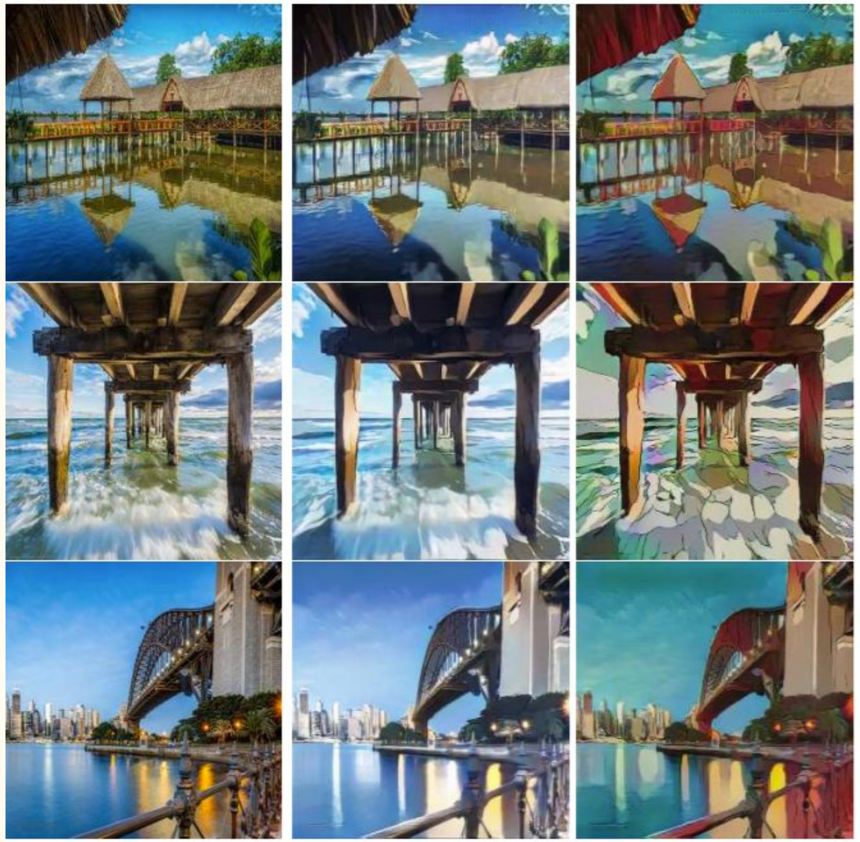
w_1
(α_1^s, α_1^b)
α_1^w
$(\alpha_1^{s'}, \alpha_1^{b'})$
w_2
(α_2^s, α_2^b)
α_2^w
$(\alpha_2^{s'}, \alpha_2^{b'})$
BN
$*$
$[\cdot]_+$

Figure 2: **Residual adapter modules.** The figure shows a standard residual module with the inclusion of adapter modules (in blue). The filter coefficients (w_1, w_2) are domain-agnostic and contains the vast majority of the model parameters; (α_1, α_2) contain instead a small number of domain-specific parameters.

Residual Adapters in CartoonGAN

- Currently CartoonGAN can only be trained on a specific artist style but we want to learn how to generate Cartoon Images of various styles together.
- We can use Residual adapters in Generator and Discriminator to learn how to generate Multiple domain images together
- Replace BatchNorm by InstanceNorm for image stylization task

Cartoon Gan Paper Results:



Thank You