"WHITE BOX CARTOONIZATION USING AN EXTENDED GAN FRAMEWORK"

Mini- Project

(Third Year/ Sem VI)

Submitted in fulfilment of the requirement of
University of Mumbai For the Degree of

Bachelor Of Engineering
(Computer Engineering)

By

1.	**AMEY THAKUR**	**TE-COMPS B-50**	**TU3F1819127**
2.	**HASAN RIZVI**	**TE-COMPS B-51**	**TU3F1819130**
3.	**MEGA SATISH**	**TE-COMPS B-58**	**TU3F1819139**

Under the Guidance of
Prof. Rohini Palve

Department of Computer Engineering
TERNA ENGINEERING COLLEGE
Plot no.12, Sector-22, Opp. Nerul Railway station,
Phase-11, Nerul (w), Navi Mumbai 400706
UNIVERSITY OF MUMBAI

2020-2021

Internal Approval Sheet

Terna Engineering College

NERUL, NAVI MUMBAI

CERTIFICATE

This is to certify that

1. **AMEY MAHENDRA THAKUR**
2. **HASAN MEHDI RIZVI**
3. **MEGA SATISH**

Has satisfactorily completed the requirements of the
Mini Project (Third year/Sem VI)
entitled

**"WHITE BOX CARTOONIZATION
USING AN EXTENDED GAN FRAMEWORK"**

As prescribed by the University of Mumbai Under the guidance of

Prof. Rohini Palve

GUIDE **APC** **HOD**

Approval Sheet

Project Report Approval

This Mini Project Report - I entitled

“WHITE BOX CARTOONIZATION USING AN EXTENDED GAN FRAMEWORK”

by the following students is approved for the degree of Bachelor in **"Computer Engineering(Sem-VI)"**.

Submitted by:

AMEY THAKUR	**TU3F1819127**
HASAN RIZVI	**TU3F1819130**
MEGA SATISH	**TU3F1819139**

Examiners Name & Signature:

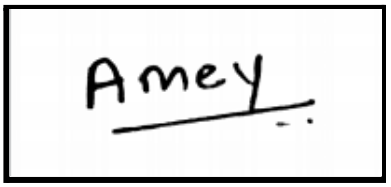
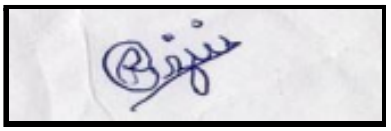
1. ----------

2. ----------

Date: 01-05-2021
Place: MUMBAI

DECLARATION

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced The cartoon sources task-specific. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

AMEY THAKUR	**TU3F1819127**	
HASAN RIZVI	**TU3F1819130**	
MEGA SATISH	**TU3F1819139**	

Date: 01-05-2021
Place: MUMBAI

ACKNOWLEDGEMENT

We would like to express our sincere gratitude towards our guide **Prof. Priyanka Sherkhane**, Mini Project Mentors **Prof. Rohini Palve, Prof. Nayana Vaitty** for their help, guidance and encouragement, they provided during the project development. This work would have not been possible without their valuable time, patience and motivation. We thank them for making our stint thoroughly pleasant and enriching. It was a great learning and an honour being their student.

We are deeply thankful to **Dr Archana Mire** (H.O.D Computer Department) and the entire team in the Computer Department. They supported us with scientific guidance, advice and encouragement, they were always helpful and enthusiastic and this inspired us in our work.

We take the privilege to express our sincere thanks to **Dr L. K. Ragha** our Principal for providing encouragement and much support throughout our work.

ABSTRACT

We propose to implement a new framework for estimating generative models via an adversarial process to extend the existing GAN framework and develop a white-box controllable image cartoonization, which can generate high-quality cartoonized images from real-world photos. Images are decomposed into three cartoon representations. The surface representation that contains a smooth surface of cartoon images, the structure representation that refers to the sparse colour-blocks and flattens global content in the celluloid style workflow, and the texture representation that reflects high-frequency texture, contours, and details in cartoon images. The learning objectives of our method are separately based on each extracted representations, making our framework controllable and adjustable. This project demonstrates the potential of the framework through qualitative and quantitative evaluation of the generated samples. This project will be based on the model proposed by Xinrui Wang and Jinze Yu - Learning to Cartoonize Using White-Box Cartoon Representations.

LIST OF FIGURES

LIST OF ABBREVIATIONS

Acronym	Abbreviation
WBC	White Box Cartoonization
GAN	Generative Adversarial Network
DCGAN	Deep Convolutional Generative Adversarial Network
LReLU	Leaky Rectified Linear Activation
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short Term Memory
FFHQ	Flickr-Faces-HQ
DIV2K	DIVerse 2K
FID	Frechet Inception Distance
GPU	Graphics Processing Unit
GANILLA	Generative Adversarial Networks for Image to Illustration Translation

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Cartoons are a very popular art form that has been widely applied in diverse scenes, from publication in printed media to storytelling for children. Some cartoon artwork was created based on real-world scenes. However, manually re-creating real-life based scenes can be very laborious and requires refined skills.

The evolution in the field of Machine Learning has expanded the possibilities of creating visual arts. Some famous products have been created by turning real-world photography into usable cartoon scene materials, where the process is called image cartoonization.

White box cartoonization is a method that reconstructs high-quality real-life pictures into exceptional cartoon images using the GAN framework.

1.1 AIM AND OBJECTIVE OF THE PROJECT

The main objective of this project is to develop modern cartoon animation workflows which allow artists to use a variety of sources to create content. A cartoon is a popular art form that has been widely applied in diverse scenes. Some famous products have been created by turning real-world photography into usable cartoon scene materials, where the process is called image cartoonization.

The motivation to create this project has many sources:

- Interest to take in-depth knowledge of Generative Adversarial Networks (GANs) Framework.
- Interest to develop a good user-friendly model of image cartoonization.
- To increase our knowledge horizon in newly developed technologies like CartoonGAN - Tensorflow, W-B Cartoonization.
- To gain good experience we will attempt to deploy it as a web app using HTML/CSS and Tensorflow.js.

1.2 SCOPE OF THE PROJECT

Generative Adversarial Networks (GAN) have attained remarkable results in image enhancement and augmentation. Currently, our project focuses on the cartoonization of high-quality images. This may prove to be an asset in the entertainment industry. GAN framework can furthermore be used in other fields such as Video Games, Animes, Animated Movies.

1.3 ORGANIZATION OF THE REPORT

- Chapter 1 gives a brief overview of the aim of developing this project. Also, it defines the scope of the project.
- Chapter 2 of the report includes the literature survey on the existing system, a brief description of the existing system is given, and comparisons between CartoonGAN system and White Box Cartoonization using GAN are explained.
- Chapter 3 defines the problem statement and proposes a new system as a solution.
- Chapter 4 shows the UML diagrams that give us an abstract view of the system. This chapter gives a detailed explanation of the technologies used and the techniques used in the development of the project. Along with this the Hardware and software requirements and software components are described.
- Chapter 5 elaborates the methodology used in our proposed system as well as the distinct architectures of the components. It also defines the mathematical concepts and algorithms used for the implementation of the project.
- Chapter 6 is about the implementation details as well the description of the dataset used for training the model. It also shows the working of the project.
- Chapter 7 evaluates the performance of the project. We have set up an experimental setup to do the same. Also, Quantitative Evaluation and Qualitative Comparisons are given.
- Chapter 8 depicts the project timeline and the Gantt chart of the project.
- Chapter 9 contains the snapshots of the web app created and also the results of the White Box Cartoonization model.
- Chapter 10 is the conclusion. This chapter gives a summary of the entire project.

CHAPTER 2

LITERATURE SURVEY

2.1 BRIEF HISTORY OF AN EXISTING SYSTEM SIMILAR TO OUR CHOSEN SYSTEM

Cartoonization is itself a classic art but the evolution in the field of Artificial Intelligence has opened many opportunities. Many models have been developed to generate cartoon images from pictures, but have many drawbacks.

CartoonGAN is one of the technologies to generate cartoonized images but it adds noise and reduces the quality of an image. On the other hand, White Box Cartoonization overcomes these problems and results in more precise and sharp images.

The challenges in CartoonGAN are described as follows:

1. The problem of stability between generator and discriminator.
2. Problem to determine the positioning of the object.
3. The problem in understanding the perspective of images i.e. 2D or 3D.
4. The problem in understanding global objects i.e. trees, flowers, etc.

This project will be based on the model proposed by Xinrui Wang and Jinze Yu - Learning to Cartoonize Using White-Box Cartoon Representations. The main objective of this project is to develop modern cartoon animation workflows which allow artists to use a variety of sources to create content.

2.2 COMPARISON OF WHITE BOX CARTOONIZATION USING GAN FRAMEWORK WITH THE PREVIOUS SYSTEM

White Box Cartoonization is based on Black Box Cartoonization and has overcome the drawbacks found in the latter.

For example, in some cartoon workflows by analysing the processed dataset pay more attention to global palette themes and treat the sharpness of lines as a secondary issue. For others, the sharpness of objects and persons hold a great value. Black box cartoonization models struggle to effectively deal with such diverse workflow requirements and using a black-box model to directly fit the training data can negatively affect generality and stylization quality, resulting in poor-quality outputs.

To get a better idea of Cartoonization, researchers consulted artists and observed cartoon painting behaviour to identify three separate cartoon image representations: a surface representation that contains smooth surfaces, a structure representation that refers to sparse colour-blocks and flattens global content in the workflow, and a texture representation that reflects high-frequency texture, contours, and details in cartoon images.

Image processing modules extract each representation, and a generative adversarial network (GAN) framework is used to learn the extracted representations and cartoonize the input images. Output styles can be controlled by adjusting the weight of each representation in the loss function.

In a user study with 10 respondents and 30 images, the proposed approach outperformed three existing cartoonization methods in both cartoon quality (similarity between the input images and cartoon images) and overall quality (identifying undesirable colour shifts, texture distortions, high-frequency noise or other artefacts in the images).

CHAPTER 3

PROBLEM STATEMENT

AI-powered cartoonization has many practical applications these days — from personalized anime-style avatars to video and even fine art. Many black-box cartoonization frameworks however provide users with limited control or adjustability when rendering real-world photography into cartoon scenes.

Now, researchers have introduced a framework that can generate high-quality cartoonized images with much-improved controllability to meet artists' requirements across a wider range of styles and use cases. We aim to provide a web application in which users can feed images to get high-quality cartoonized outputs.

CHAPTER 4

DESIGN AND IMPLEMENTATION

4.1 USE CASE DIAGRAM

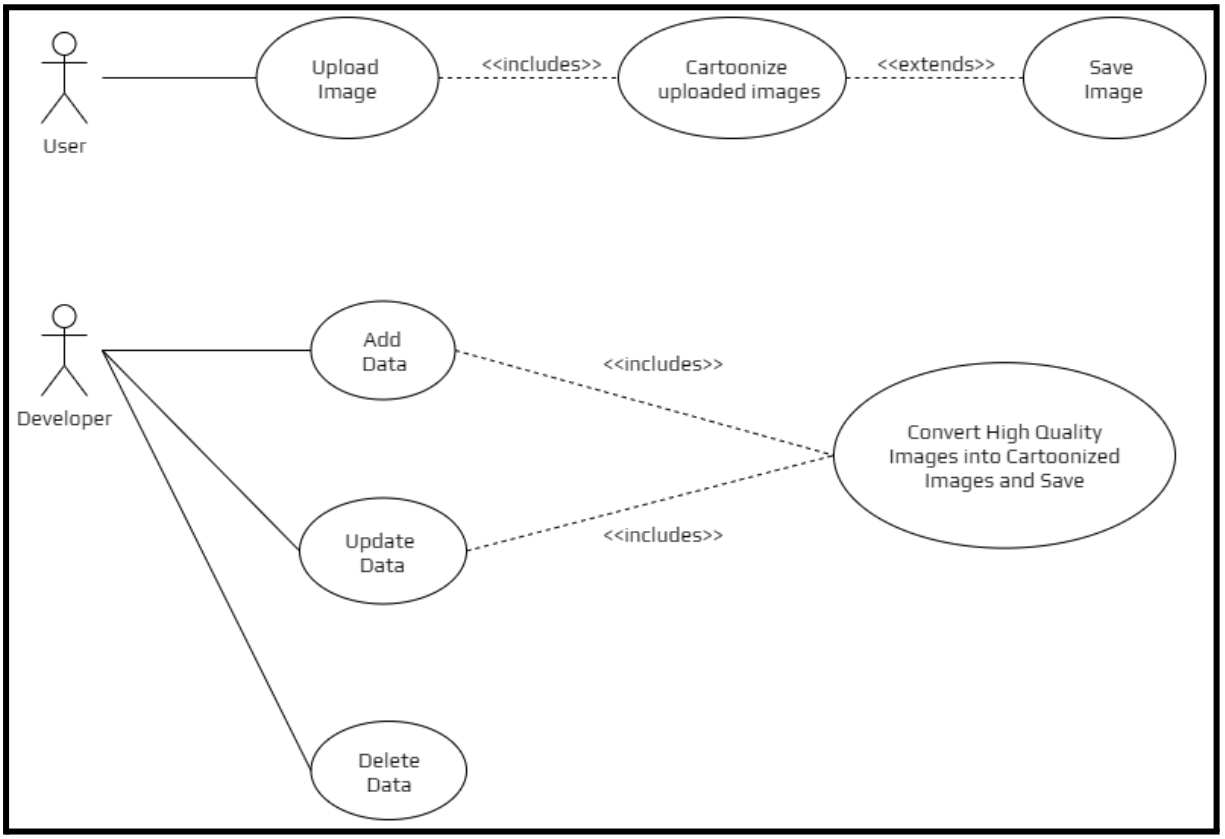


Figure 4.1: Use Case Diagram

4.2 FLOWCHART OF WHITE-BOX-CARTOONIZATION MODEL

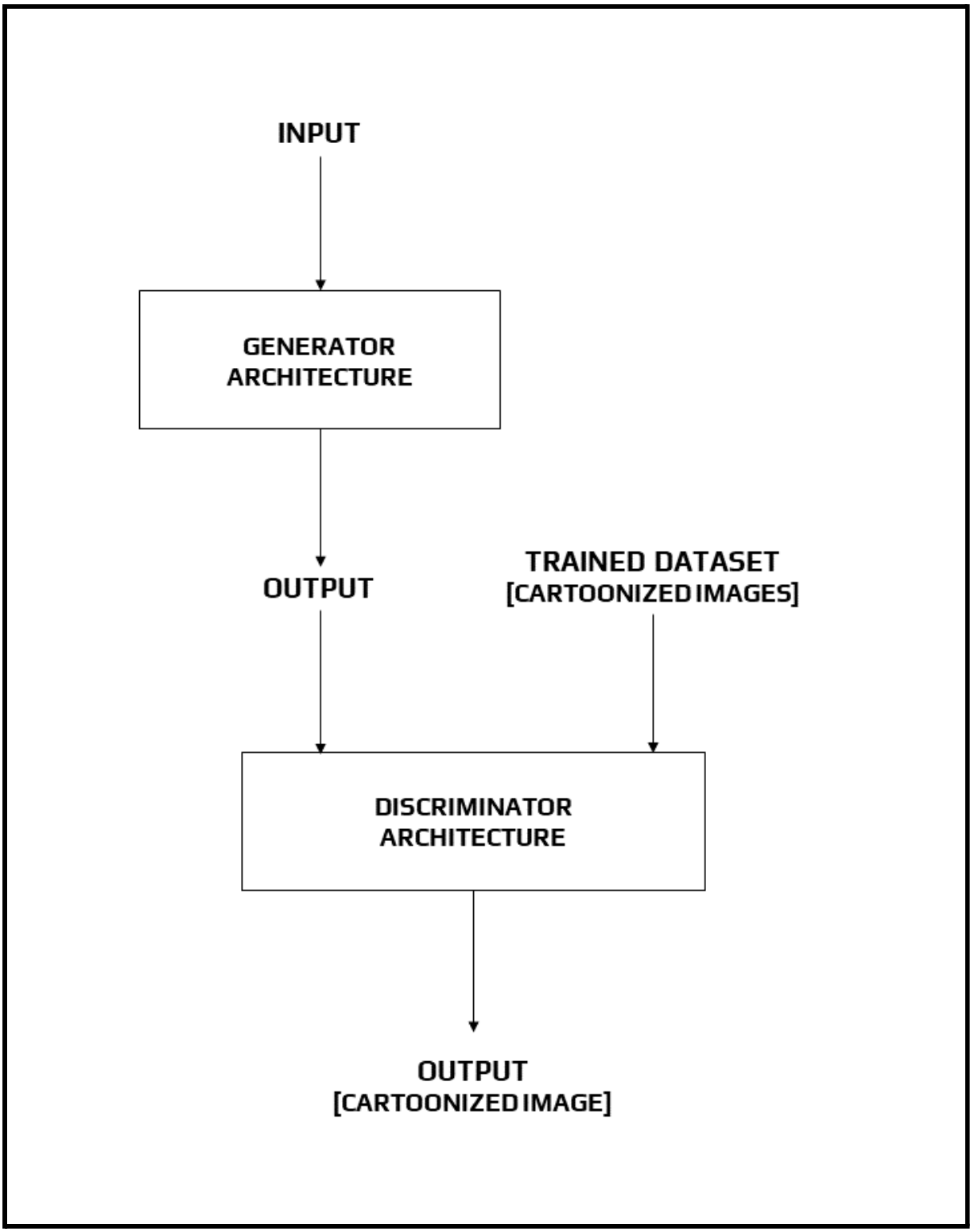


Figure 4.2: Flowchart

4.3 ARCHITECTURE OF WBC MODEL

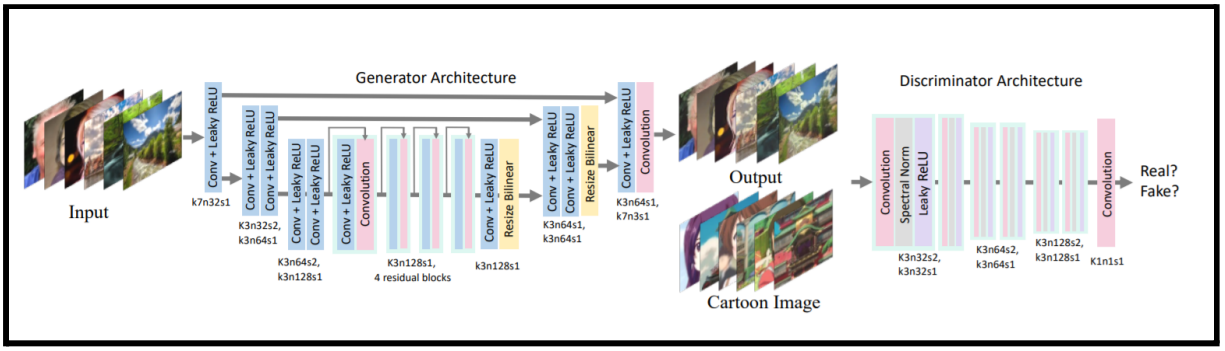


Figure 4.3: Architecture of generator and discriminator

We show the architecture of the generator network and discriminator network in the above figure. The generator network is a fully convolutional U-Net-like network. We use convolution layers with stride2 for down-sample and bilinear interpolation layers for upsampling to avoid checkerboard artefacts. The network consists of only three kinds of layers: convolution, Leaky ReLU (LReLU) and bilinear resize layers. This enables it to be easily embedded in edge devices such as mobile phones. PatchGAN is adopted in the discriminator network, where the last layer is a convolution layer. Each pixel in the output feature map corresponds to a patch in the input image, with the patch size equals to the perceptive field, and is used to judge whether the patch belongs to cartoon images or generated images. PatchGAN enhances the discriminative ability of details and accelerates training. Spectral normalization is placed after every convolution layer (except the last one) to enforce Lipschitz constrain on the network and stabilize training.

4.4 REQUIREMENTS

4.4.1 SOFTWARE REQUIREMENTS

- Web Browser

4.4.2 HARDWARE REQUIREMENTS

- No Specific Hardware requirement.
- NVIDIA GPU + CUDA CuDNN
- If a GPU is available, then the web app will give faster results.

4.5 TOOLS USED

- Anaconda
- Jupyter Notebook
- TensorFlow.js
- Bootstrap
- Flexbox
- GitHub Pages
- Colab

4.6 TECHNOLOGIES USED

- HTML
- CSS
- JavaScript
- Python

CHAPTER 5

METHODOLOGY

5.1 INTRODUCTION TO GENERATIVE ADVERSARIAL NETWORKS (GANs)

Generative adversarial networks (GANs) are an exciting recent innovation in machine learning. GANs are generative models: they create new data instances that resemble your training data. For example, GANs can create images that look like photographs of human faces, even though the faces don't belong to any real person.

Generative Adversarial Networks, or GANs for short, are an approach to generative modelling using deep learning methods, such as convolutional neural networks. Generative modelling is an unsupervised learning task in machine learning that involves automatically discovering and learning the regularities or patterns in input data in such a way that the model can be used to generate or output new examples that plausibly could have been drawn from the original dataset.

GANs are a clever way of training a generative model by framing the problem as a supervised learning problem with two sub-models: the generator model that we train to generate new examples, and the discriminator model that tries to classify examples as either real (from the domain) or fake (generated). The two models are trained together in a zero-sum game, adversarial, until the discriminator model is fooled about half the time, meaning the generator model is generating plausible examples.

GANs are an exciting and rapidly changing field, delivering on the promise of generative models in their ability to generate realistic examples across a range of problem domains, most notably in image-to-image translation tasks such as translating photos of summer to winter or day to night, and in generating photorealistic photos of objects, scenes, and people that even humans cannot tell are fake.

5.2 DEEP CONVOLUTIONAL GENERATIVE ADVERSARIAL NETWORKS (DCGAN)

The deep convolutional generative adversarial network, or DCGAN for short, is an extension of the GAN architecture for using deep convolutional neural networks for both the generator and discriminator models and configurations for the models and training that result in the stable training of a generator model.

The DCGAN is important because it suggested the constraints on the model required to effectively develop high-quality generator models in practice. This architecture, in turn, provided the basis for the rapid development of a large number of GAN extensions and applications.

5.3 ARCHITECTURE OF GAN

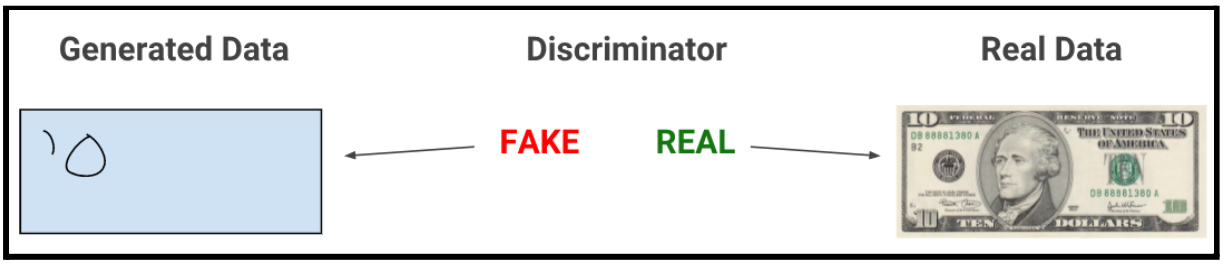
The architecture of a GAN has two basic elements: the generator network and the discriminator network. Each network can be any neural network, such as an **Artificial Neural Network** (**ANN**), a **Convolutional Neural Network** (**CNN**), a **Recurrent Neural Network** (**RNN**), or a **Long Short Term Memory** (**LSTM**). The discriminator has to have fully connected layers with a classifier at the end.

A generative adversarial network (GAN) has two parts:

- The **generator** learns to generate plausible data. The generated instances become negative training examples for the discriminator.

- The **discriminator** learns to distinguish the generator's fake data from real data. The discriminator penalizes the generator for producing implausible results.

When training begins, the generator produces fake data, and the discriminator quickly learns to tell that it's fake:



As training progresses, the generator gets closer to producing output that can fool the discriminator:

Finally, if generator training goes well, the discriminator gets worse at telling the difference between real and fake. It starts to classify fake data as real, and its accuracy decreases.

Here's a picture of the whole system:

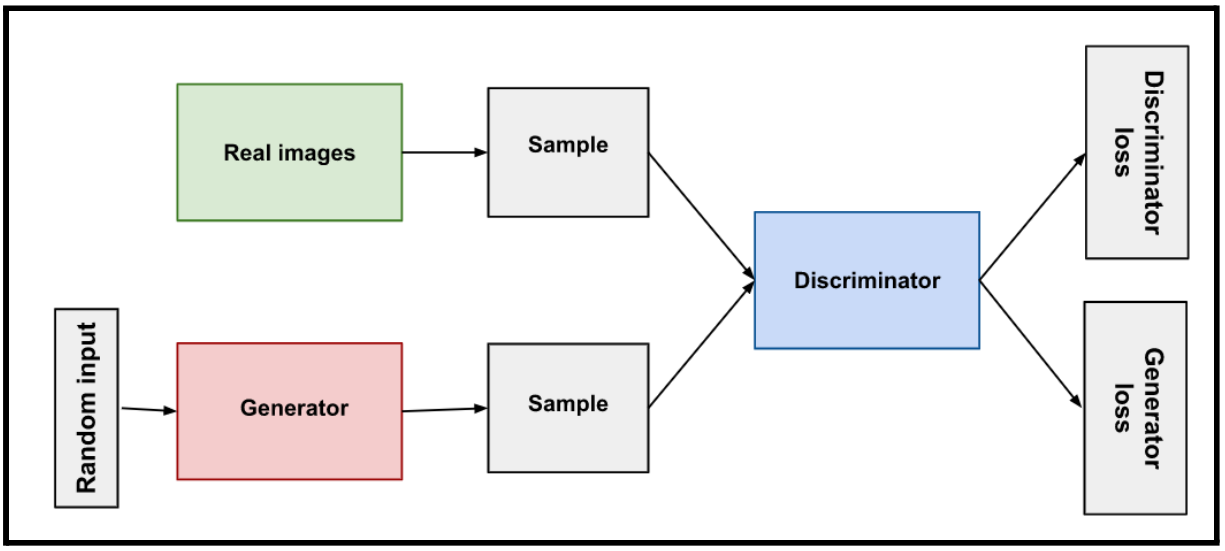


Figure 5.1: Architecture of GAN

Both the generator and the discriminator are neural networks. The generator output is connected directly to the discriminator input. Through backpropagation, the discriminator's classification provides a signal that the generator uses to update its weights.

5.3.1 THE DISCRIMINATOR

The discriminator in a GAN is simply a classifier. It tries to distinguish real data from the data created by the generator. It could use any network architecture appropriate to the type of data it's classifying.

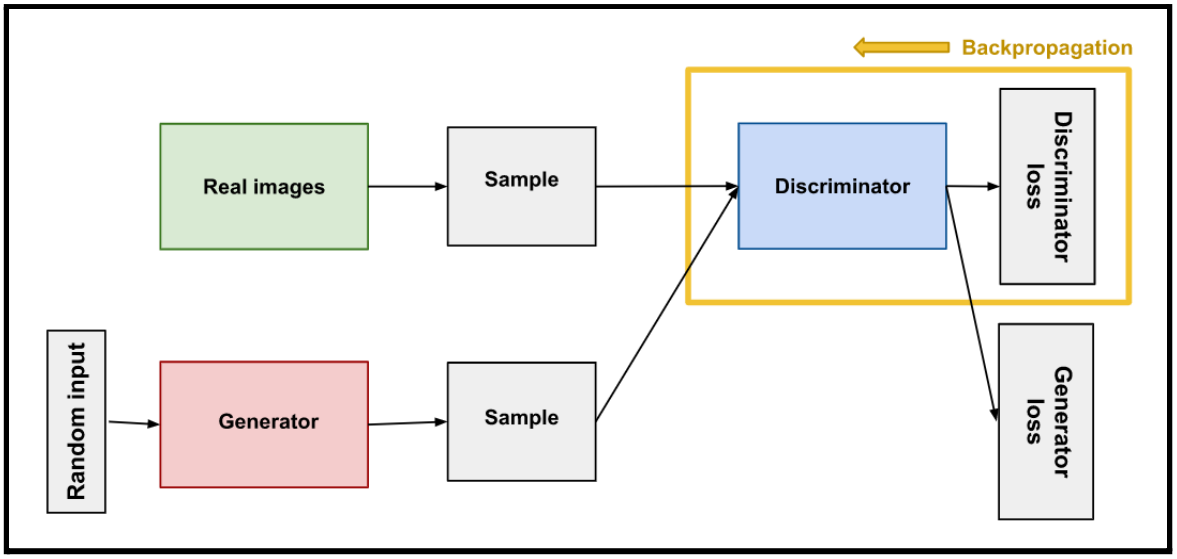


Figure 5.2: Backpropagation in discriminator training

5.3.1.1 DISCRIMINATOR TRAINING DATA

The discriminator's training data comes from two sources:

- Real data instances, such as real pictures of people. The discriminator uses these instances as positive examples during training.
- Fake data instances created by the generator. The discriminator uses these instances as negative examples during training.

In Figure 1, the two "Sample" boxes represent these two data sources feeding into the discriminator. During discriminator training, the generator does not train. Its weights remain constant while it produces examples for the discriminator to train on.

5.3.1.2 TRAINING THE DISCRIMINATOR

The discriminator connects to two loss functions. During discriminator training, the discriminator ignores the generator loss and just uses the discriminator loss. We use the generator loss during generator training, as described in the next section.

During discriminator training:

1. The discriminator classifies both real data and fake data from the generator.
2. The discriminator loss penalizes the discriminator for misclassifying a real instance as fake or a fake instance as real.
3. The discriminator updates its weights through backpropagation from the discriminator loss through the discriminator network.

5.3.2 THE GENERATOR

The generator part of a GAN learns to create fake data by incorporating feedback from the discriminator. It learns to make the discriminator classify its output as real.

Generator training requires tighter integration between the generator and the discriminator than discriminator training requires. The portion of the GAN that trains the generator includes:

- random input
- generator network, which transforms the random input into a data instance
- discriminator network, which classifies the generated data
- discriminator output
- generator loss, which penalizes the generator for failing to fool the discriminator

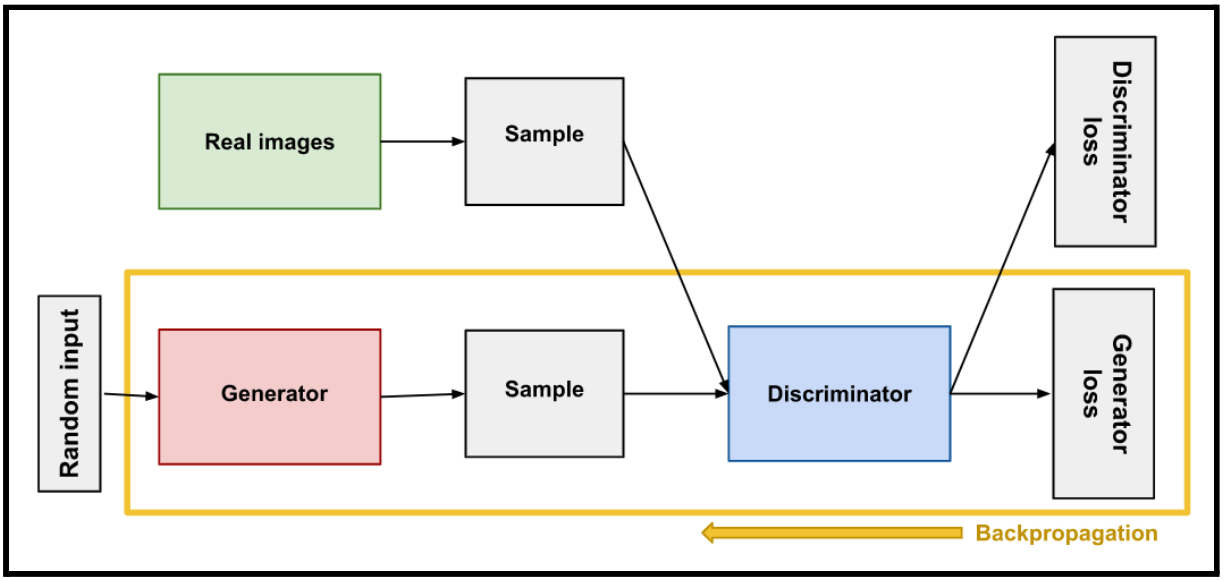


Figure 5.3: Backpropagation in generator training

5.3.2.1 USING THE DISCRIMINATOR TO TRAIN THE GENERATOR

To train a neural net, we alter the net's weights to reduce the error or loss of its output. In our GAN, however, the generator is not directly connected to the loss that we're trying to affect. The generator feeds into the discriminator net, and the discriminator produces the output we're trying to affect. The generator loss penalizes the generator for producing a sample that the discriminator network classifies as fake.

Backpropagation adjusts each weight in the right direction by calculating the weight's impact on the output — how the output would change if you changed the weight. But the impact of a generator weight depends on the impact of the discriminator weights it feeds into. So backpropagation starts at the output and flows back through the discriminator into the generator.

At the same time, we don't want the discriminator to change during generator training. Trying to hit a moving target would make a hard problem even harder for the generator.
So we train the generator with the following procedure:

1. Sample random noise.
2. Produce generator output from sampled random noise.
3. Get discriminator "Real" or "Fake" classification for generator output.
4. Calculate loss from discriminator classification.
5. Backpropagate through both the discriminator and generator to obtain gradients.
6. Use gradients to change only the generator weights.

5.4 PROS OF GAN FRAMEWORK

- Very impressive result related to content generated, and realistic visual content.

- Also can be modelled for other applications like image restoration, etc.

5.5 CONS OF GAN FRAMEWORK

- Despite the success of GAN, the potential of such a framework has certain limitations.

- Difficult for training: huge computation.

- “Realistic" but not Real. A fake pattern can be created, especially for small scale structure or non-nature objects like text ---> problem of the loss function.

- Some physical effects in the picture like shadow will become unrealistic.

5.6 WHITE-BOX-CARTOONIZATION

We propose to separately identify three white-box representations from images:

1. The surface representation
2. The structure representation
3. The texture representation

- The surface representation contains a smooth surface of cartoon images.
- The structure representation refers to the sparse colour blocks and flattens global content in the celluloid style workflow.
- The texture representation reflects high-frequency texture, contours, and details in cartoon images.
- A Generative Adversarial Network (GAN) framework is used to learn the extracted representations and to cartoonize images.

5.6.1 SURFACE REPRESENTATION

- The surface representation imitates the cartoon painting style where artists roughly draw drafts with coarse brushes and have smooth surfaces similar to cartoon images.
- To smooth images and meanwhile keep the global semantic structure, a differentiable guided filter is adopted for edge-preserving filtering.
- Edge-preserving filtering is an image processing technique that smooths away noise or textures while retaining sharp edges. Examples are the median, bilateral, guided, and anisotropic diffusion filters.

5.6.1.1 SURFACE LOSS FORMULA

$$L_{surface}(G, D_s) = \log D_s(F_{dgf}(I_c, I_c)) + \log(1 - D_s(F_{dgf}(G(I_p), G(I_p))))$$

Where,

G = Generator,

Ds = Discriminator,

Ic = Reference Cartoon Image,

Ip = Input Photo,

Fdgf = It takes an image I as input and itself as a guide map, returns extracted surface representation Fdgf (I, I) with textures and details removed.

Note: A discriminator Ds is introduced to judge whether model outputs and reference cartoon images have similar surfaces, and guide generator G to learn the information stored in the extracted surface representation.

5.6.2 STRUCTURE REPRESENTATION

- We at first used the felzenszwalb algorithm to segment images into separate regions. As superpixel algorithms only consider the similarity of pixels and ignore semantic information, we further introduce selective search to merge segmented regions and extract a sparse segmentation map.

- Standard superpixel algorithms colour each segmented region with an average of the pixel value. We found this lowers global contrast, darkens images, and causes a hazing effect on the final results by analysing the processed dataset. We thus propose an adaptive colouring algorithm

- Adaptive coloring formula,

$$S_{i,j} = (\theta 1 * \bar{S} + \theta 2 * \check{S})^{\wedge}\mu$$

$$(\theta 1, \theta 2) = \begin{cases} (0, 1) & \sigma(S) < \gamma 1, \\ (0.5, 0.5) & \gamma 1 < \sigma(S) < \gamma 2, \\ (1, 0) & \gamma 2 < \sigma(S). \end{cases}$$

Where we find $\gamma 1 = 20$, $\gamma 2 = 40$ and $\mu = 1.2$ generate good results.

This effectively enhances the contrast of images and reduces the hazing effect.

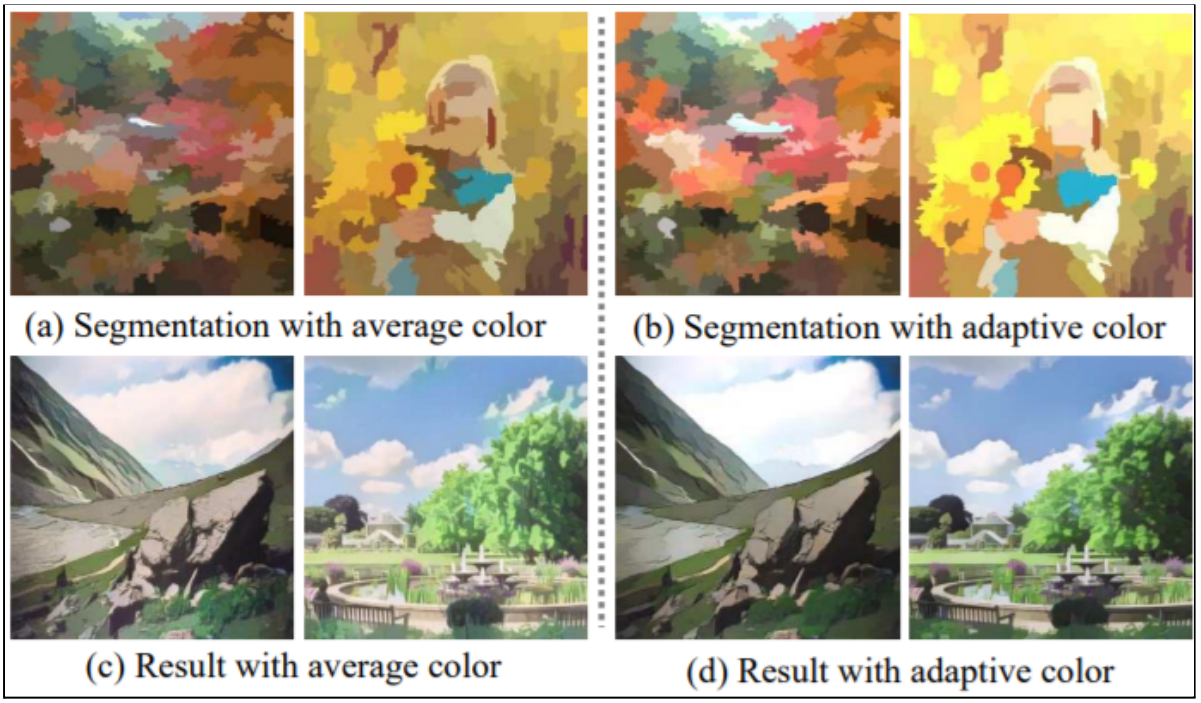


Figure 5.4: Adaptive colouring algorithm.

(a) and (b) show segmentation maps with different colouring methods, while (c) and (d) shows results generated with different colouring methods. Adaptive colouring generates results that are brighter and free from hazing effects.

5.6.2.1 STRUCTURE LOSS FORMULA

$$L_{structure} = \| VGG_n(G(I_p)) - VGG_n(F_{st}(G(I_p))) \|$$

Where,

G = Generator,

Ip = Input Photo,

Fst = Structure Representation Extraction.

Note: We use high-level features extracted by a pre-trained VGG16 network to enforce spatial constraints between our results and extracted structure representation.

5.6.3 TEXTURE REPRESENTATION

- The high-frequency features of cartoon images are key learning objectives, but luminance and colour information make it easy to distinguish between cartoon images and real-world photos. We thus propose a random colour shift algorithm. The random colour shift can generate random intensity maps with luminance and colour information removed.

- Frcs extract single-channel texture representation from colour images, which retains high-frequency textures and decreases the influence of colour and luminance.

$$F_{rcs}(I_{rgb}) = (1 - \alpha)(\beta_1 * I_r + \beta_2 * I_g + \beta_3 * I_b) + \alpha * Y$$

Where,

Irgb represents 3-channel RGB colour images, Ir, Ig and Ib represent three colour channels, and Y represents standard grayscale image converted from RGB colour image.

Note: We set $\alpha = 0.8$, β_1, β_2 and $\beta_3 \sim U(-1, 1)$.

5.6.3.1 TEXTURAL REPRESENTATION FORMULA

$$L_{texture}(G, Dt) = \log Dt(Frcs(Ic)) + \log(1 - Dt(Frcs(G(Ip))))$$

Where,

G = Generator,

Dt = Discriminator,

Ic = Reference Cartoon Image,

Ip = Input Photo,

Frcs = Extract single-channel texture representation from colour images, which retains high-frequency textures and decreases the influence of colour and luminance.

5.7 AGILE METHODOLOGY

- Agile methodology is a practice that promotes continuous iteration of development and testing throughout the software development life cycle of the project.

- In the Agile model, both development and testing activities are concurrent, unlike the Waterfall model.

- It is a new approach to software development that involves iterative development and incremental release of the product. Agile methodology is dynamic, context-specific, aggressively change embracing and growth-oriented.

- Agile software development emphasizes four core values.
    1. Individual and team interactions over processes and tools
    2. Working software over comprehensive documentation
    3. Customer collaboration over contract negotiation
    4. Responding to change over following a plan

- Benefits of Agile Methodology are described as follows:
    1. Faster delivery of high quality and more reliable products and early identification of defects
    2. Better customer satisfaction
    3. Collaborative approach

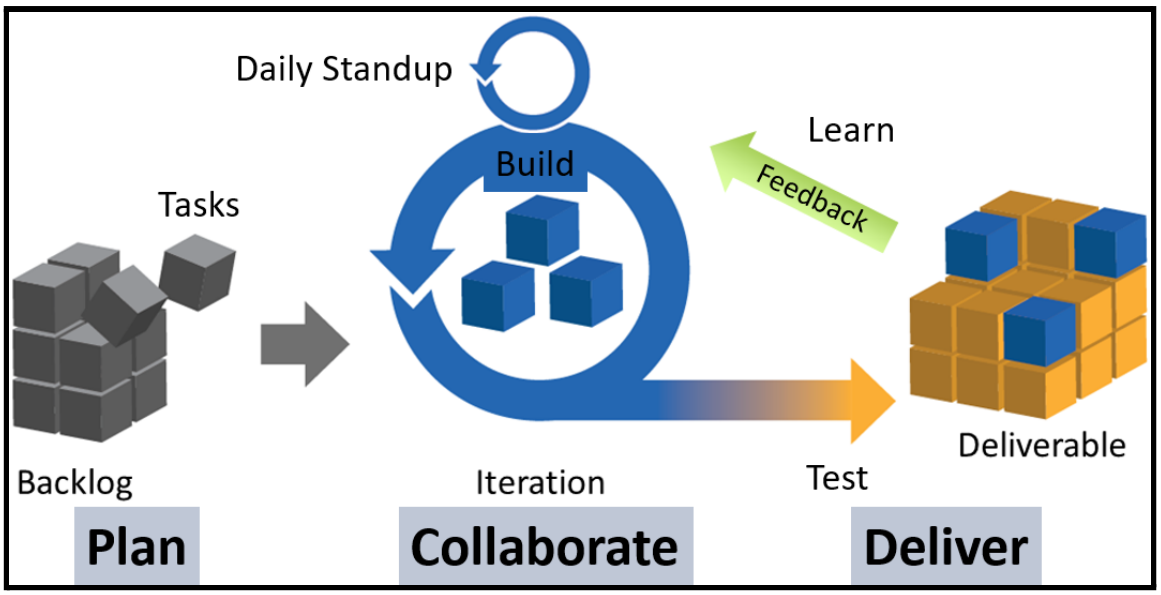


Figure 5.5: Agile way of working

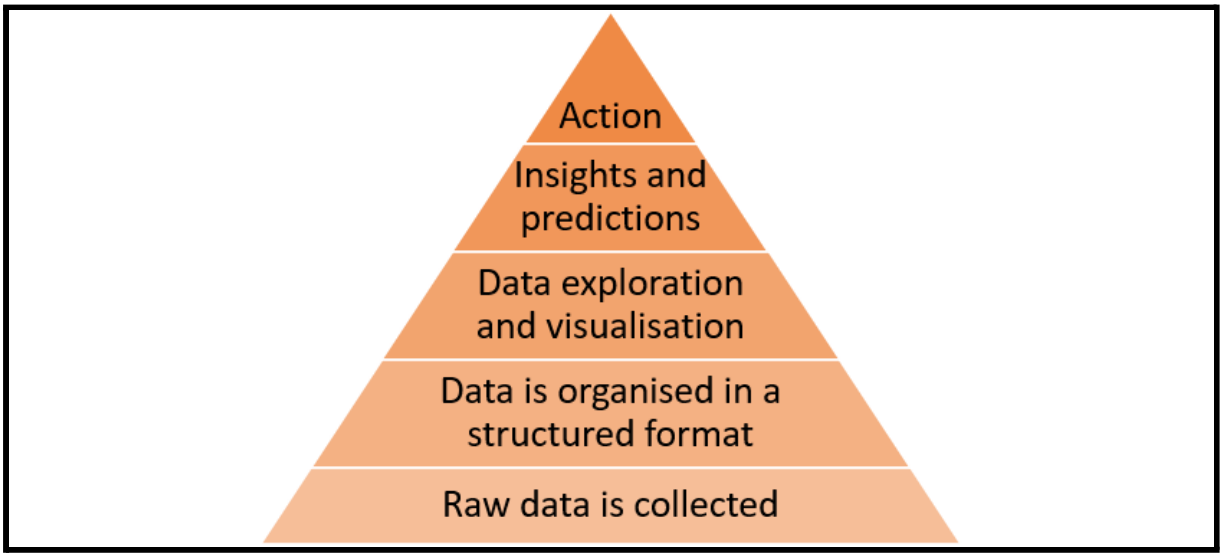


Figure 5.6: Data Value Pyramid

CHAPTER 6

IMPLEMENTATION DETAILS

Project Repository -
https://github.com/rizvihasan/whitebox_cartoonisation-webapp

6.1 IMPLEMENTATION

- We implement our GAN method with TensorFlow.
- Patch discriminator is adopted to simplify the calculation and enhance discriminative capacity.
- We use the Adam algorithm to optimize both networks. Learning rate and batch size are set to $2 * 10^{-4}$ and 16 during training.
- We at first pre-train the generator with the content loss for 50000 iterations, and then jointly optimize the GAN based framework. Training is stopped after 100000 iterations or on convergence.

6.2 DATASET

- Human face and landscape data are collected for generalization on diverse scenes.
- For real-world photos, we collect 10000 images from the FFHQ dataset for the human face and 5000 images from the dataset for landscape.
- For cartoon images, we collect 10000 images from animations for the human face and 10000 images for landscape.
- For the validation set, we collect 3011 animation images and 1978 real-world photos.
- Images shown in the main paper are collected from the DIV2K dataset, and images in the user study are collected from the Internet and Microsoft COCO dataset.
- During training, all images are resized to 256*256 resolution, and face images are fed only once in every five iterations.

6.3 LEARNING RATE

When tuning the hyperparameters of our model, we first performed a grid search to find an optimal learning rate of 0.001. Because we were testing locally due to GCloud resource shortages, our mini-batch size was limited to 2.

6.4 WORKING OF THE SYSTEM

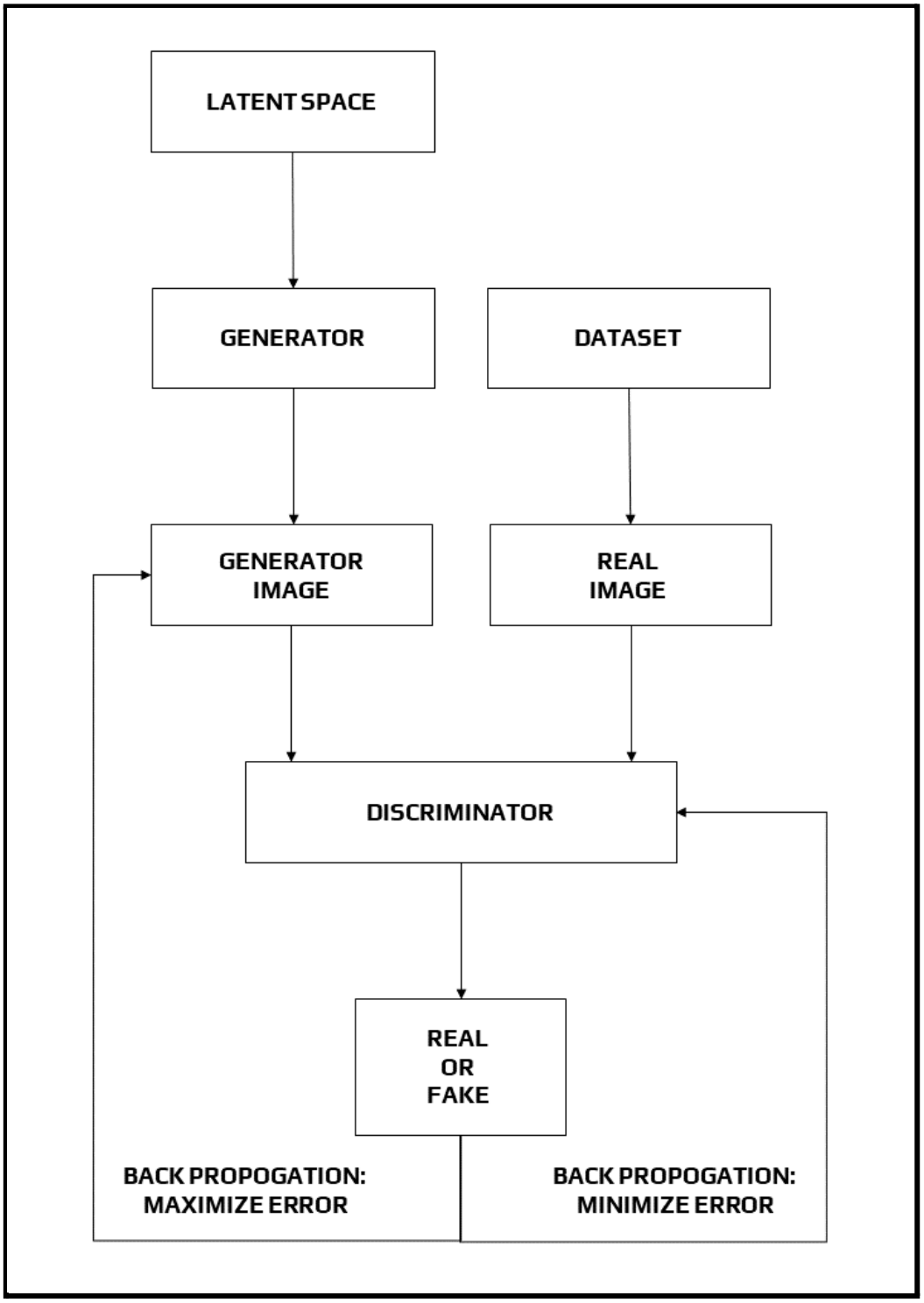


Figure 6.1: Working of the system

CHAPTER 7

PERFORMANCE EVALUATION

7.1 PREVIOUS METHODS

We compare our method with four algorithms that represent:

1. Neural Style Transfer,
2. Image-to-Image Translation,
3. Image Abstraction,
4. Image Cartoonization.

7.2 EVALUATION METRICS

In qualitative experiments, we present results with details of four different methods and original images, as well as qualitative analysis. In quantitative experiments, we use Frechet Inception Distance (FID) to evaluate the performance by calculating the distance between source image distribution and target image distribution. In the user study, candidates are asked to rate the results of different methods between 1 to 5 in cartoon quality and overall quality. Higher scores mean better quality.

7.3 TIME PERFORMANCE AND MODEL SIZE

- Our model is the fastest among four methods on all devices and all resolutions and has the smallest model size.
- Especially, our model can process a 720*1280 image on GPU within only 17.23ms, which enables it for real-time High-Resolution video processing tasks.
- Generality to diverse use cases: We apply our model to diverse real-world scenes, including natural landscape, city views, people, animals, and plants.

7.4 VALIDATION OF CARTOON REPRESENTATIONS

To validate our proposed cartoon representations reasonable and effective, a classification experiment and a quantitative experiment based on FID are conducted. We train a binary classifier on our training dataset to distinguish between real-world photos and cartoon images. The classifier is designed by adding a fully connected layer to the discriminator in our framework. The trained classifier is then evaluated on the validation set to validate the influence of each cartoon representation.

We find the extracted representations successfully fool the trained classifier, as it achieves lower accuracy in all three extracted cartoon representations compared to the original images. The calculated FID metrics also support our proposal that cartoon representations help close the gap between real-world photos and cartoon images, as all three extracted cartoon representations have smaller FID compared to the original images.

7.5 QUALITATIVE COMPARISON

Figure 7.1: Qualitative comparison
The Second row shows 4 different styles of CartoonGAN

Comparisons between our method and previous methods are shown in Figure 7.1.

The white-box framework helps generate clean contours. Image abstraction causes noisy and messy contours, and other previous methods fail to generate clear borderlines, while our method has clear boundaries, such as human face and clouds. Cartoon representations also help keep colour harmonious. CycleGAN generates darkened images and Fast Neural Style causes over smoothed colour, and CartoonGAN distorts colours like human faces and ships. Our method, on the contrary, prevents improper colour modifications such as faces and ships. Lastly, our method effectively reduces artefacts while preserving fine details, such as the man sitting on the stone, but all other methods cause over-smoothed features or distortions. Also, methods like CycleGAN, image abstraction and some styles of CartoonGAN cause high-frequency artefacts. To conclude, our method outperforms previous methods in generating images with harmonious colour, clean boundaries, fine details, and fewer noises.

7.6 QUANTITATIVE EVALUATION

Frechet Inception Distance (FID) is widely used to quantitatively evaluate the quality of synthesized images. The pre-trained Inception-V3 model is used to extract high-level features of images and calculate the distance between two image distributions. We use FID to evaluate the performance of previous methods and our method. As CartoonGAN models have not been trained on human face data, for fair comparisons, we only calculate FID on the scenery dataset. Our method generates images with the smallest FID to cartoon image distribution, which proves it generates results most similar to cartoon images. The output of our method also has the smallest FID to real-world photo distribution, indicating that our method loyally preserves image content information.

7.7 ANALYSIS OF EACH COMPONENT

Results of ablation studies in Figure 7.2. Ablating the texture representation causes messy details.

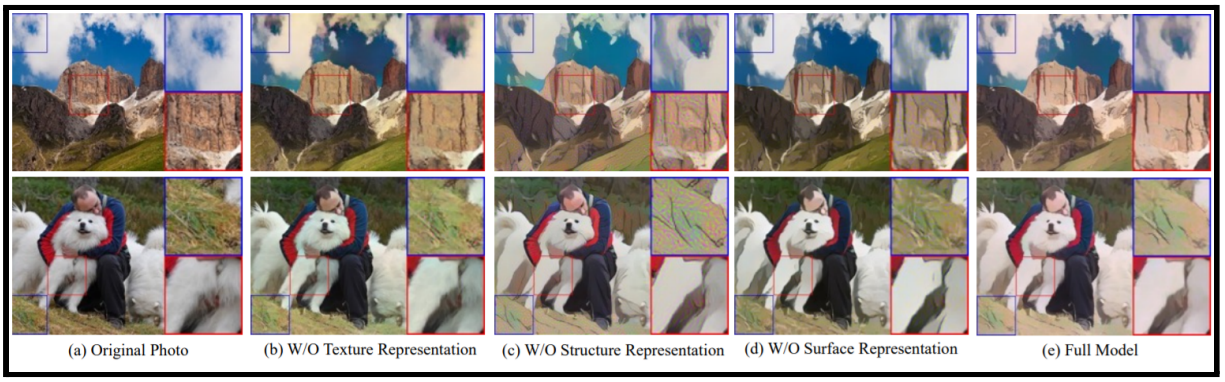


Figure 7.2: Ablation study by removing each component

- As shown in Figure 7.2(a), irregular textures on the grassland and the dog's leg remain. This is due to the lack of high-frequency stored in the surface representation, which deteriorates the model's cartoonization ability.

- Ablating the structure representation causes high-frequency noises in Figure 7.2(b). Severe pepper-and-salt appears on the grassland and the mountain. This is because the structure representation flattened images and removed high-frequency information. Ablating the surface representation causes both noise and messy details.

- Unclear edges of the cloud and noises on the grassland appear in Figure 7.2(c). The reason is that guided filtering suppresses high-frequency information and preserves smooth surfaces.

- As a comparison, the results of our full model are shown in Figure 7.2(d), which have smooth features, clear boundaries, and much less noise. In conclusion, all three representations help improve the cartoonization ability of our method.

CHAPTER 8

TIMELINE

8.1 PROBLEM TIMELINE

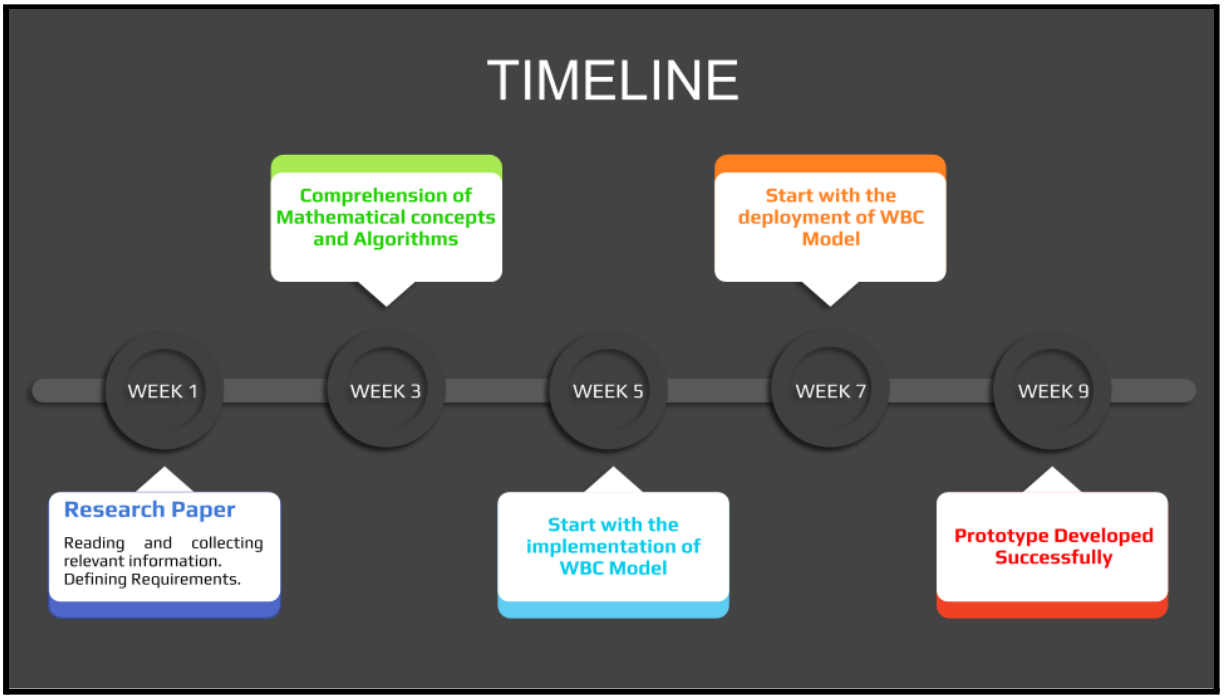
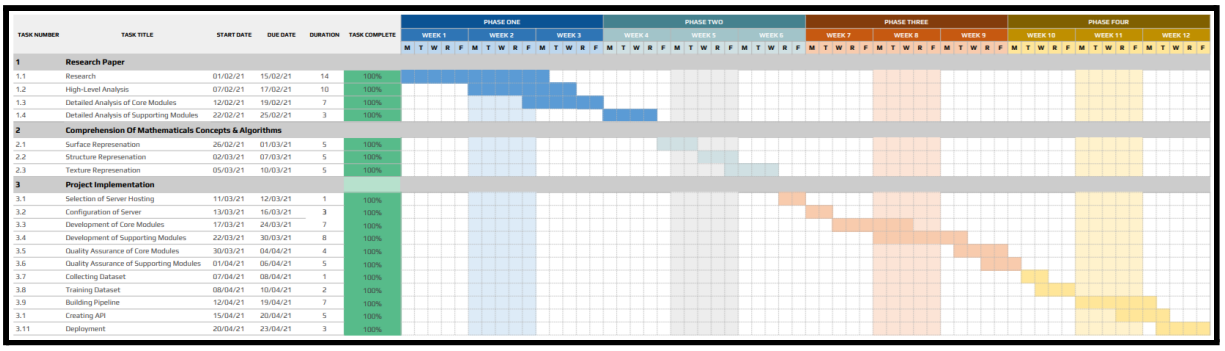


Figure 8.1: Timeline

8.2 GANTT CHART



TASK NUMBER	TASK TITLE	START DATE	DUE DATE	DURATION	TASK COMPLETE
1	**Research Paper**				
1.1	Research	01/02/21	15/02/21	14	100%
1.2	High-Level Analysis	07/02/21	17/02/21	10	100%
1.3	Detailed Analysis of Core Modules	12/02/21	19/02/21	7	100%
1.4	Detailed Analysis of Supporting Modules	22/02/21	25/02/21	3	100%
2	**Comprehension Of Mathematicals Concepts & Algorithms**				
2.1	Surface Represenation	26/02/21	01/03/21	5	100%
2.2	Structure Represenation	02/03/21	07/03/21	5	100%
2.3	Texture Represenation	05/03/21	10/03/21	5	100%
3	**Project Implementation**				
3.1	Selection of Server Hosting	11/03/21	12/03/21	1	100%
3.2	Configuration of Server	13/03/21	16/03/21	3	100%
3.3	Development of Core Modules	17/03/21	24/03/21	7	100%
3.4	Development of Supporting Modules	22/03/21	30/03/21	8	100%
3.5	Quality Assurance of Core Modules	30/03/21	04/04/21	4	100%
3.6	Quality Assurance of Supporting Modules	01/04/21	06/04/21	5	100%
3.7	Collecting Dataset	07/04/21	08/04/21	1	100%
3.8	Training Dataset	08/04/21	10/04/21	2	100%
3.9	Building Pipeline	12/04/21	19/04/21	7	100%
3.1	Creating API	15/04/21	20/04/21	5	100%
3.11	Deployment	20/04/21	23/04/21	3	100%

Figure 8.2: Gantt Chart

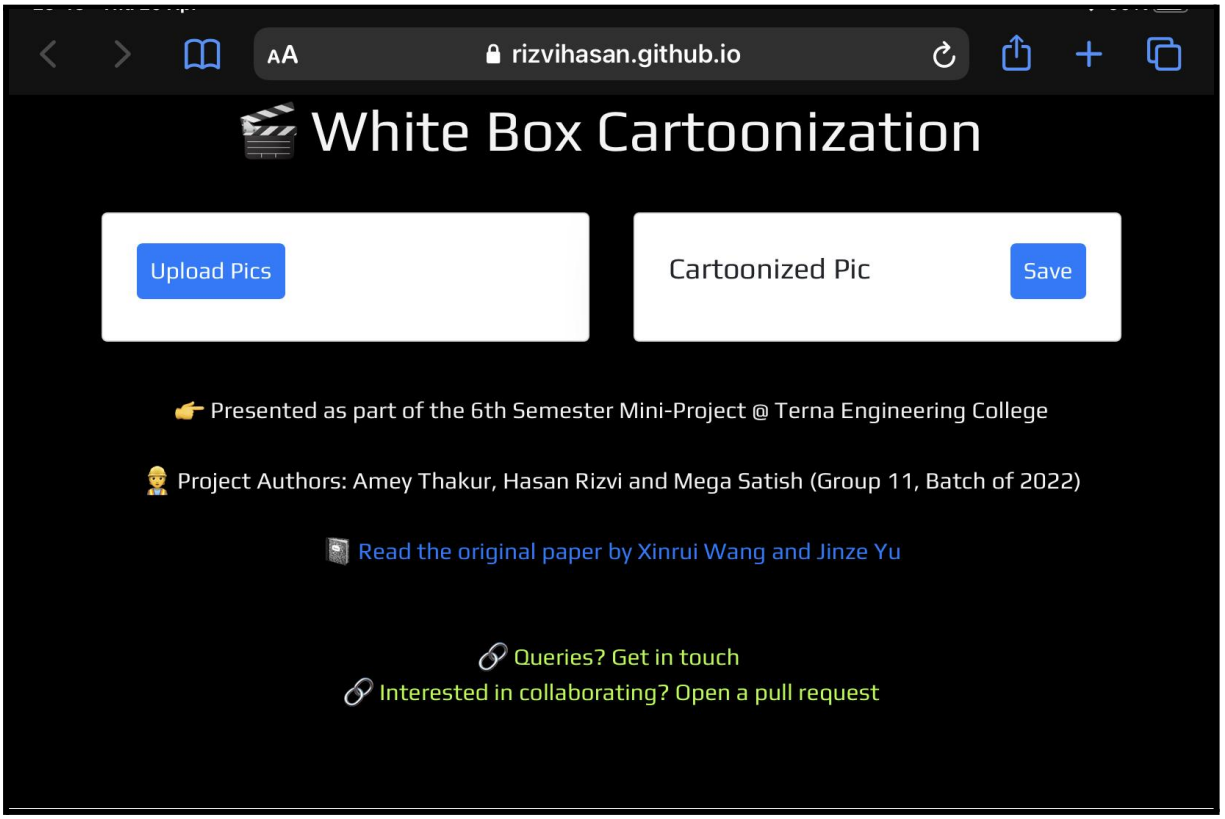
CHAPTER 9

RESULTS

9.1 WEB APP

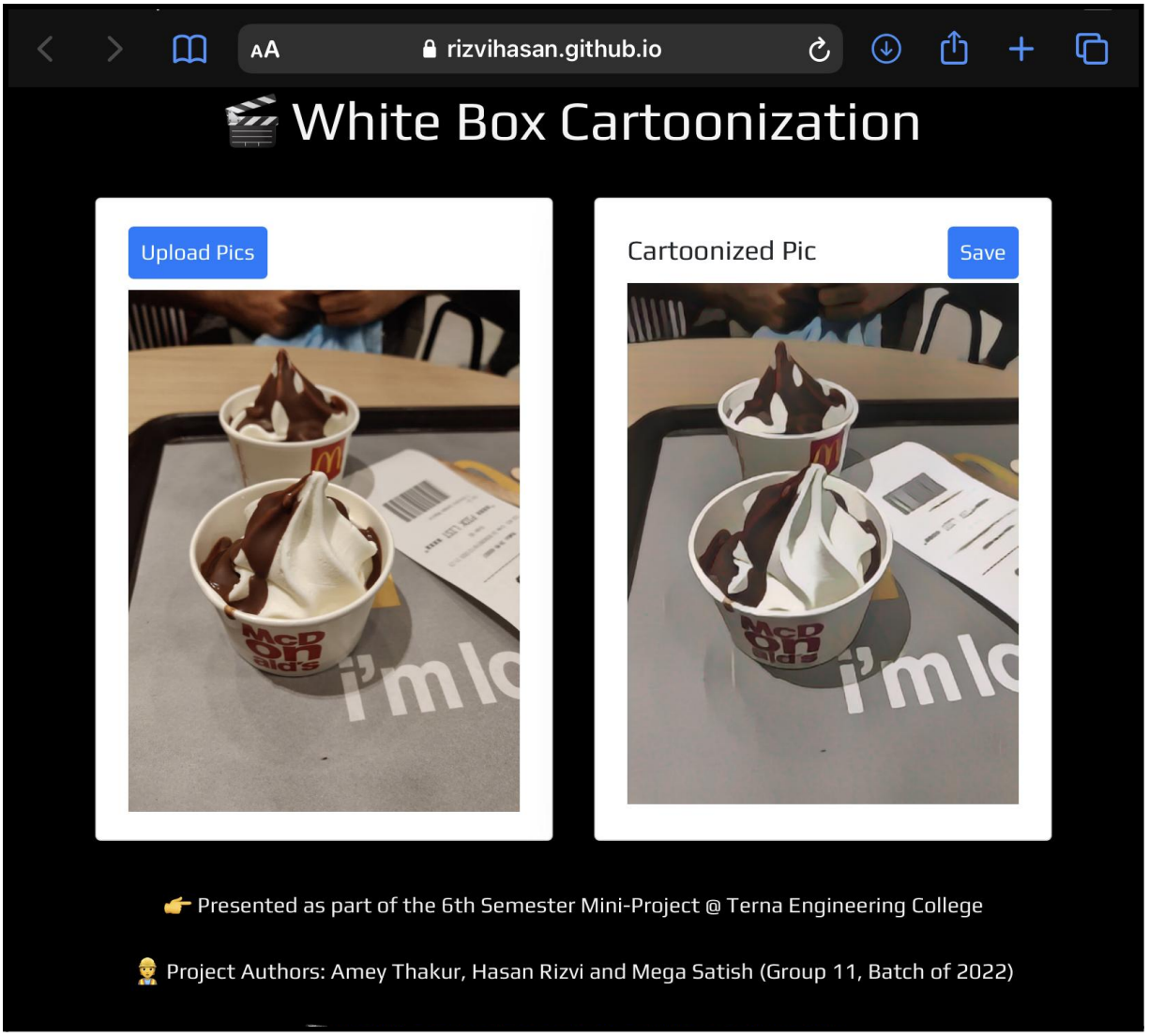
Link: https://rizvihasan.github.io/whitebox_cartoonisation-webapp

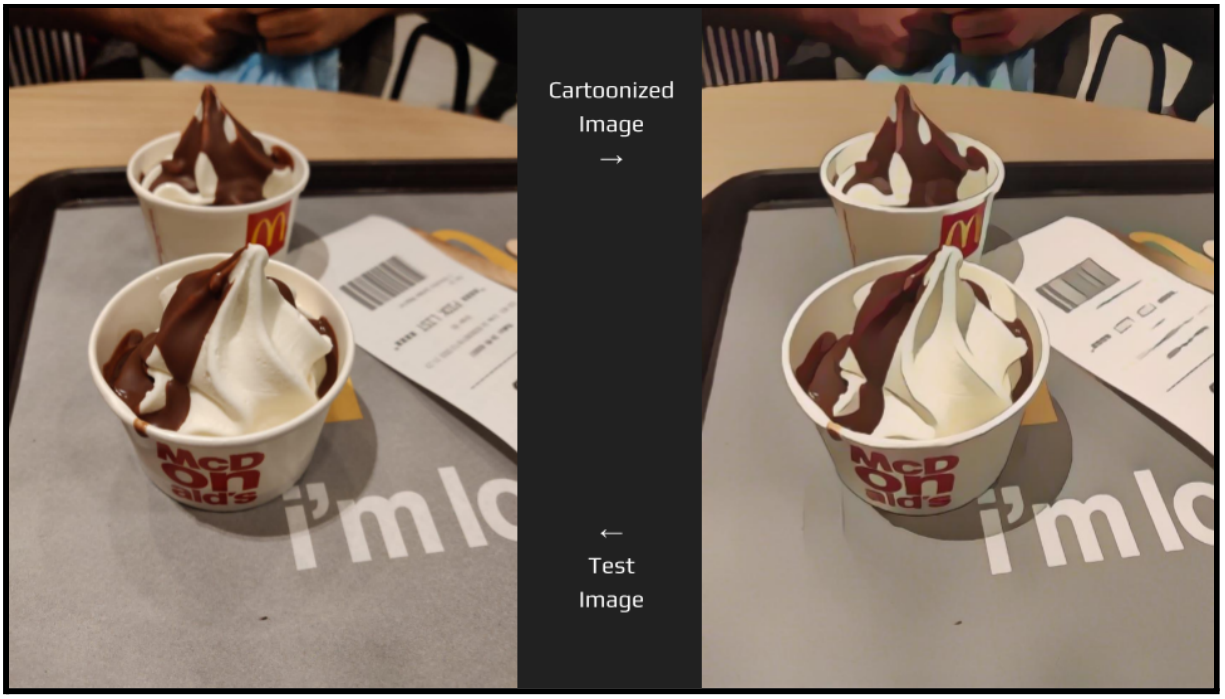
Sample Implementation: MINI-PROJECT_TE-COMPS_B-50,51,58.mp4

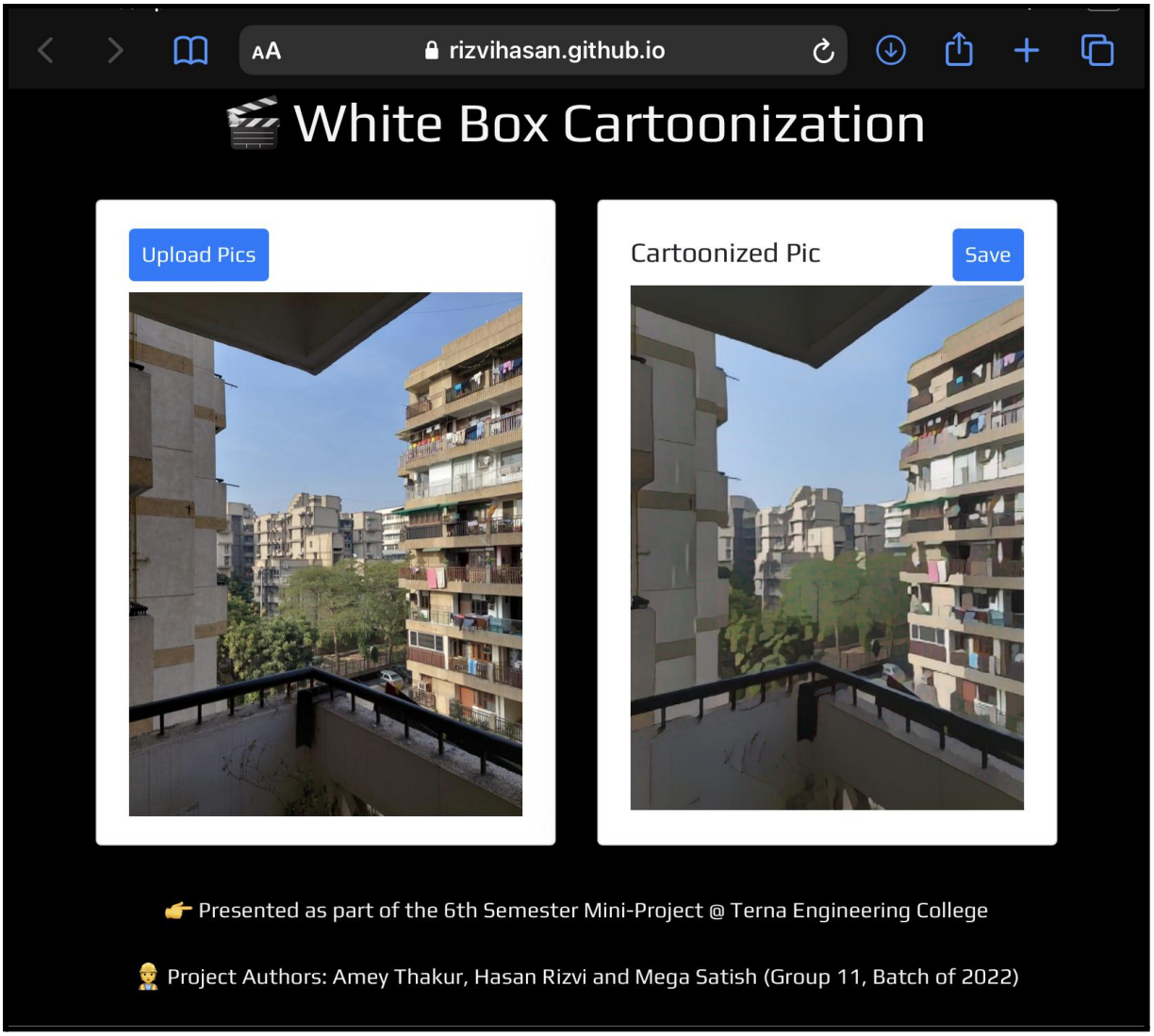


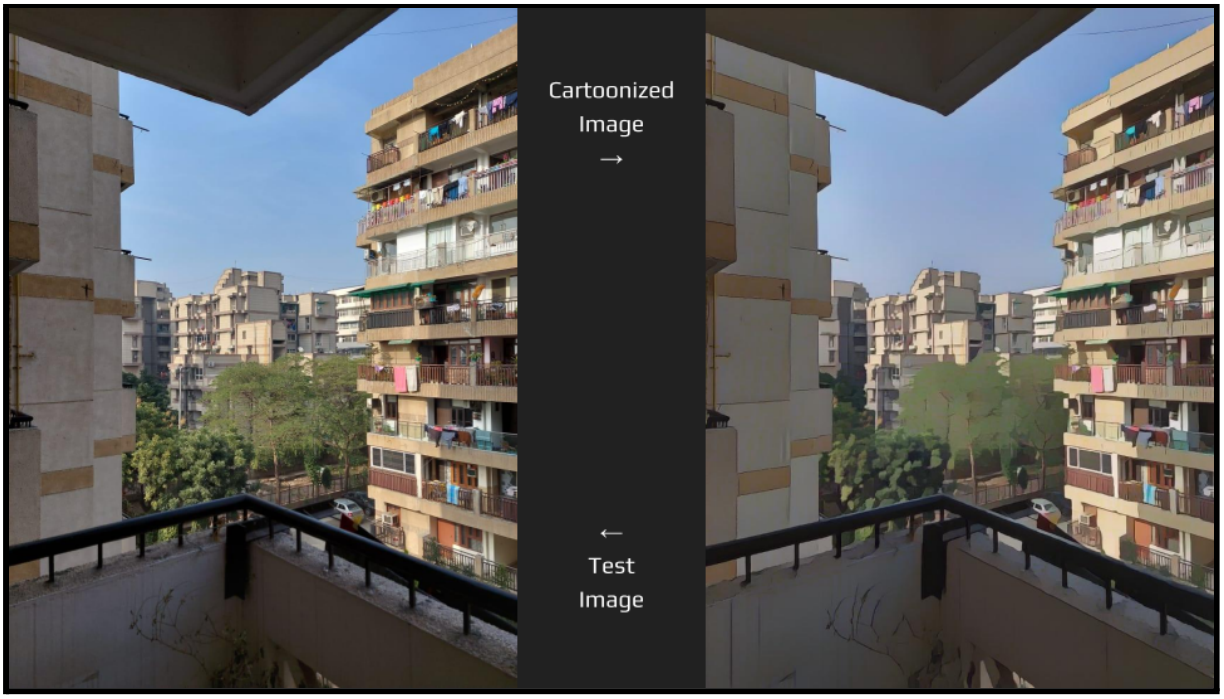
As we can see in the cartoonized pictures below, they are very similar when it comes to the sharpness of the object and the presence of different colours in the pictures.

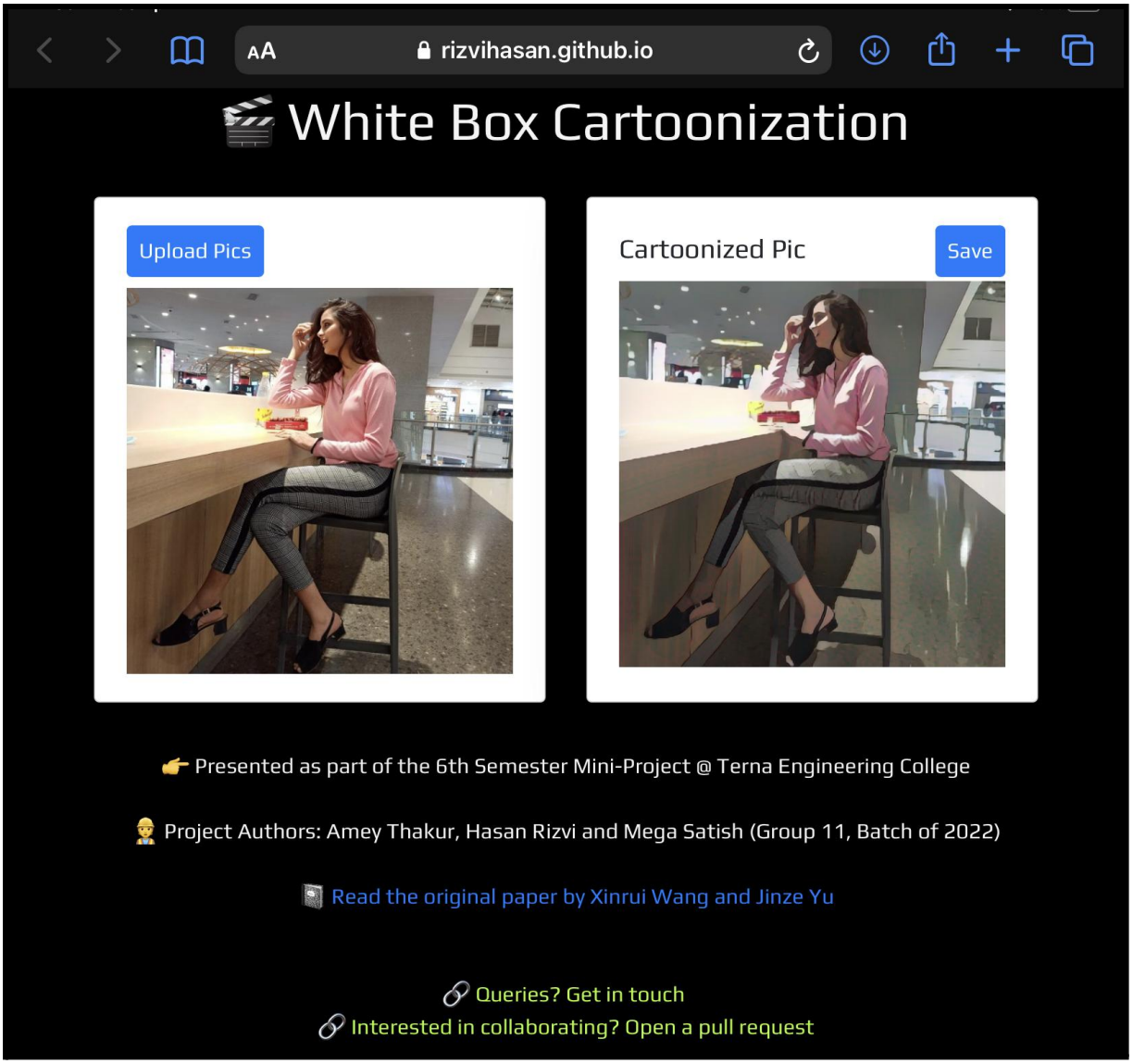
Additionally, elements like reflection, shadows are depicted with precision.

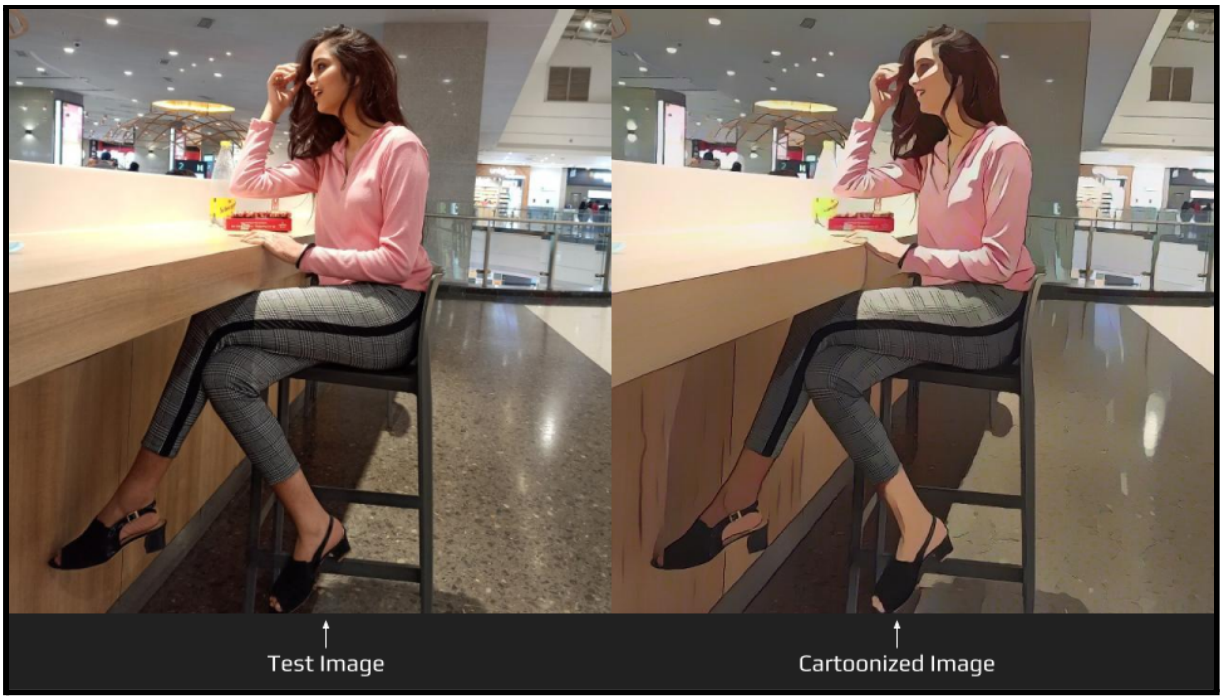

AA
rizvihasan.github.io
White Box Cartoonization
Upload Pics
Cartoonized Pic
Save
Presented as part of the 6th Semester Mini-Project @ Terna Engineering College
Project Authors: Amey Thakur, Hasan Rizvi and Mega Satish (Group 11, Batch of 2022)

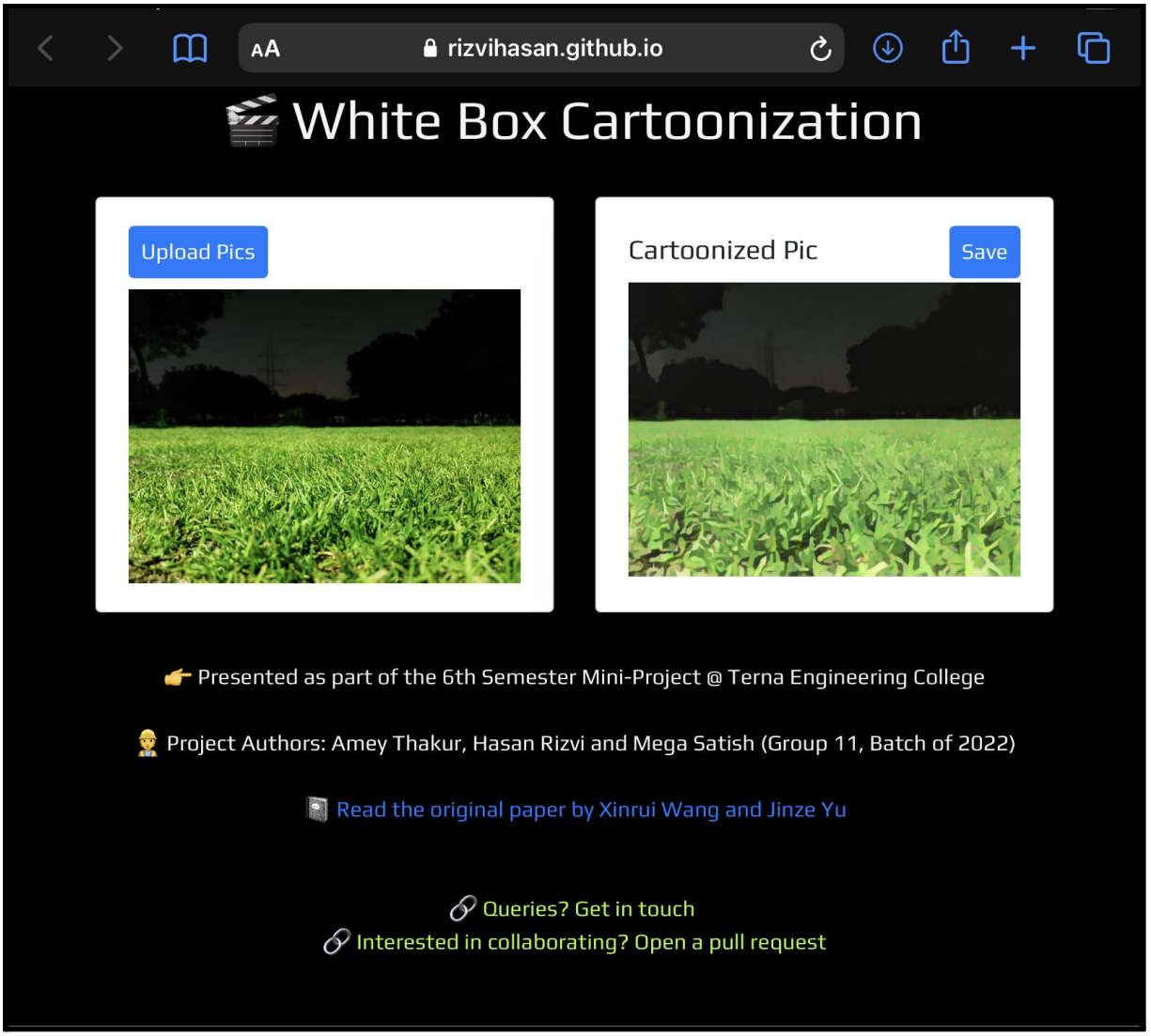

Cartoonized Image →
← Test Image


AA
rizvihasan.github.io
White Box Cartoonization
Upload Pics
Cartoonized Pic
Save
Presented as part of the 6th Semester Mini-Project @ Terna Engineering College
Project Authors: Amey Thakur, Hasan Rizvi and Mega Satish (Group 11, Batch of 2022)


Cartoonized Image
→
←
Test Image


AA
rizvihasan.github.io
White Box Cartoonization
Upload Pics
Cartoonized Pic
Save
Presented as part of the 6th Semester Mini-Project @ Terna Engineering College
Project Authors: Amey Thakur, Hasan Rizvi and Mega Satish (Group 11, Batch of 2022)
Read the original paper by Xinrui Wang and Jinze Yu
Queries? Get in touch
Interested in collaborating? Open a pull request

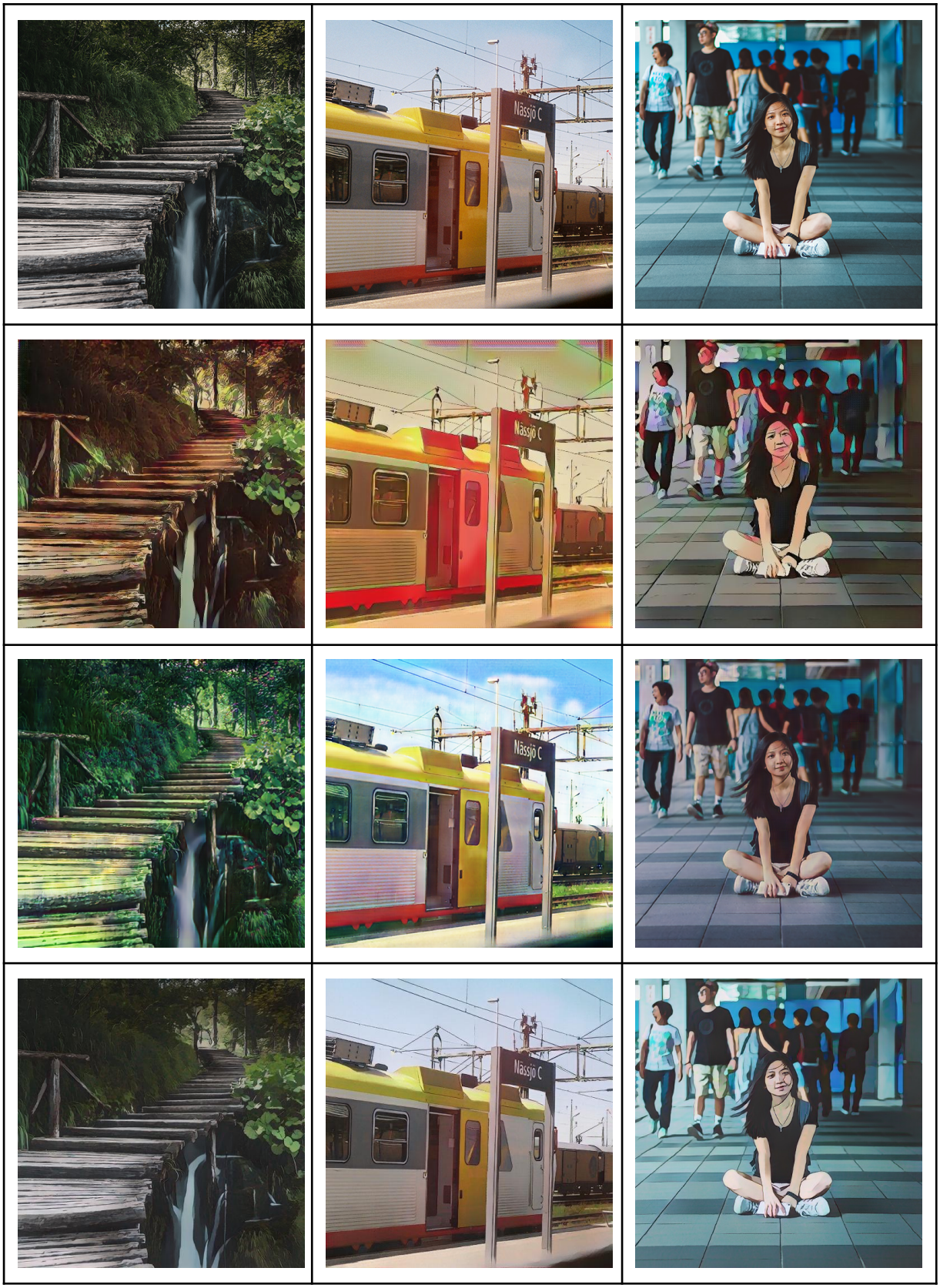

Test Image
Cartoonized Image


AA
rizvihasan.github.io
White Box Cartoonization
Upload Pics
Cartoonized Pic
Save
Presented as part of the 6th Semester Mini-Project @ Terna Engineering College
Project Authors: Amey Thakur, Hasan Rizvi and Mega Satish (Group 11, Batch of 2022)
Read the original paper by Xinrui Wang and Jinze Yu
Queries? Get in touch
Interested in collaborating? Open a pull request


Test Image
Cartoonized Image

9.2 COMPARISON BETWEEN CARTOONGAN, GANILLA AND WBC

The four rows contain original images, CartoonGAN Results, GANILLA Results and WBC Results respectively.



CHAPTER 10

CONCLUSION

In this paper, we proposed a deployed white-box controllable image cartoonization framework based on GAN, which can generate high-quality cartoonized images from real-world photos. Images are decomposed into three cartoon representations: the surface representation, the structure representation, and the texture representation. Corresponding image processing modules are used to extract three representations for network training. Extensive quantitative and qualitative experiments have been conducted to validate the performance of our method. Ablation studies are also conducted to demonstrate the influence of each representation.

Further, deep work and development in this white box cartoonization can lead to many various applications:

- Generates quick prototypes or sprites for anime, cartoons and games.
- Since it subdues facial features and information in general, it can be used to generate minimal art.
- Games can import shortcut scenes very easily without using motion-capture.
- Can be modelled as an assistant to graphic designers or animators.

REFERENCES

- https://arxiv.org/abs/2002.05638
- https://arxiv.org/abs/1406.2661
- https://systemerrorwang.github.io/White-box-Cartoonization
- https://www.researchgate.net/publication/329748755_CartoonGAN_Generative_Adversarial_Networks_for_Photo_Cartoonization
- https://www.researchgate.net/publication/343457093_Learning_to_Cartoonize_Using_White-Box_Cartoon_Representations
- https://www.tensorflow.org/js/models
- https://developer.mozilla.org/en-US/docs/Web/API/CanvasRenderingContext2D
- https://getbootstrap.com/docs/4.0/components/card
- https://alibabatech.medium.com/run-machine-learning-models-in-your-browser-with-tensorflow-js-e5f78a840818