VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JNANA SANGAMA", MACHHE, BELAGAVI-590018

ML Mini Project Report
on

Cartoon Image Generator

Submitted in partial fulfillment of the requirements for the VI semester
Bachelor of Engineering
in
Artificial Intelligence & Machine Learning
of
Visvesvaraya Technological University, Belagavi
by

Keerthana RS
(1CD21AI024)
Ramya P
(1CD21AI045)

Under the Guidance of
Dr. Varalatchoumy. M
Prof. Syed Hayath
Dept. of AI&ML

Department of Artificial Intelligence & Machine Learning
CAMBRIDGE INSTITUTE OF TECHNOLOGY, BANGALORE-560 036

2023-2024

CAMBRIDGE INSTITUTE OF TECHNOLOGY

K.R. Puram, Bangalore-560 036

DEPARTMENT OF ARTIFICIAL INTELLIGENCE & MACHINE LEARNING

CERTIFICATE

Certified that **Ms. Keerthana RS** bearing USN **1CD21AI024** and **Ms. Ramya P** bearing USN **1CD21AI045,** a bonafide student of **Cambridge Institute of Technology,** has successfully completed the ML Mini Project entitled "**Cartoon Image Generator"** in partial fulfillment of the requirements for VI semester **Bachelor of Engineering** in **Artificial Intelligence & Machine Learning** of **Visvesvaraya Technological University, Belagavi** during academic year 2023-24. It is certified that all Corrections/Suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The ML Mini Project report has been approved as it satisfies the academic requirements prescribed for the Bachelor of Engineering degree.

Mini Project Guides,

1. Dr. Varalatchoumy. M,

2.Asst Prof. Syed Hayath
Dept. of AI&ML, CITech

Head of the Department,
Dr. Varalatchoumy. M
Dept. of AI&ML, CITech

DECLARATION

We, Keerthana RS and **Ramya P** of VI semester BE, Artificial Intelligence & Machine Learning, Cambridge Institute of Technology, hereby declare that the ML Mini Project entitled "**Cartoon Image Generator"** has been carried out by us and submitted in partial fulfilment of the course requirements of VI semester **Bachelor of Engineering** in **Artificial Intelligence & Machine Learning** as prescribed b**y Visvesvaraya Technological University, Belagavi**, during the academic year 2023-2024.

We also declare that, to the best of our knowledge and belief, the work reported here does not form part of any other report on the basis of which a degree or award was conferred on an earlier occasion on this by any other student.

Date:
Place: Bangalore

Keerthana RS
1CD21AI024

Ramya P
1CD21AI045

ACKNOWLEDGEMENT

We would like to place on record our deep sense of gratitude to **Shri. D. K. Mohan,** Chairman, Cambridge Group of Institutions, Bangalore, India for providing excellent Infrastructure and Academic Environment at CITech without which this work would not have been possible**.**

We are extremely thankful to **Dr. G. Indumathi,** Principal, CITech, Bangalore, for providing us the academic ambience and everlasting motivation to carry out this work and shaping our careers.

We express our sincere gratitude to **Dr. Varalatchoumy.M,** Prof. & Head, Dept. of Artificial Intelligence & Machine Learning, CITech, Bangalore, for her stimulating guidance, continuous encouragement and motivation throughout the course of present work.

We also wish to extend our thanks to Mini Project Guides, **Dr. Varalatchoumy.M,** Prof. & Head, and **Prof. Syed Hayath**, Assistant Professor, Dept. of AI&ML, CITech, Bangalore for the critical, insightful comments, guidance and constructive suggestions to improve the quality of this work.

Finally, to all our friends, classmates who always stood by us in difficult situations also helped us in some technical aspects and last but not the least, we wish to express deepest sense of gratitude to our parents who were a constant source of encouragement and stood by us as pillar of strength for completing this work successfully.

Keerthana RS

Ramya P

CONTENTS

List of Figures

ABSTRACT

Cartoon image generation has gained significant attention in computer vision and digital art due to its applications in entertainment, virtual reality, and personalized content creation. AnimeGAN, a specialized variant of Generative Adversarial Networks (GANs), has emerged as a powerful tool for synthesizing anime-style images from real-world photographs. This project explores the capabilities and advancements of AnimeGAN in generating high-quality anime-style images, contributing to the field's ongoing evolution in stylized image synthesis.

The proposed framework utilizes AnimeGAN's architecture, which includes a generator network trained to convert input photographs into anime-style images and a discriminator network trained to distinguish between generated and real anime artworks. The generator employs convolutional layers and residual blocks optimized for capturing the intricate details, vibrant colours, and expressive features characteristic of anime art. This approach ensures that the generated images exhibit stylistic coherence and visual appeal akin to hand-drawn animations.

Key enhancements to AnimeGAN include the integration of perceptual loss functions and style transfer techniques, which improve the fidelity and diversity of generated outputs. These additions enable the preservation of both content and style from the input photographs, achieving realistic and artistically pleasing results.

Evaluation of the proposed AnimeGAN model encompasses quantitative metrics measuring image quality, diversity, and style consistency, alongside qualitative assessments through human perceptual studies and expert evaluations.

The potential applications of AnimeGAN span a wide range of fields, including character design, gaming, virtual reality, and interactive entertainment, fostering new opportunities for artistic expression and technological innovation in the digital age.

CHAPTER 1

INTRODUCTION

Cartoons are commonly used in various kinds of applications. As we know cartoons are made artistically it requires elegant and fine human artistic skills. While portraying Cartoons in humongous numbers for any animated movies it gets time-consuming for the artist as they need to define the sketch of the cartoon properly to get a good result. Cartoons are not only done in film industry but also done in different fields like people try to convert their own image into cartoons to know their appearance in Cartoon effect. But this is all time taking process and requires more resources which might be difficult to collect all resources and execute accordingly.This technique of cartoonifying image reduces the time consumption for Artists in Films. They can get the same output as before with this technique. But compared to sketching an image, cartoonifying an image is efficient and quick. Python programming language is used for writing a code to this technique. And different python libraries are used to get most accurate output. Mainly OpenCV, one of the libraries in python, is used in this method of cartoonifying.

1.1 Background

A cartoon image generator uses various techniques in artificial intelligence, computer graphics, and image processing to create cartoon-like representations of images. Here's a background on how these systems typically work:

Machine Learning Models: The core technology behind many modern cartoon image generators is machine learning, particularly deep learning models such as convolutional neural networks (CNNs). These models are trained on large datasets of images and their cartoon equivalents.

Style Transfer: A popular method used in cartoon image generation is neural style transfer. This technique involves training a neural network to separate and recombine content and style from two images. The content of one image is combined with the style of another (e.g., a cartoon style) to create a new image.

Generative Adversarial Networks (GANs): GANs are another powerful approach used in this field. They consist of two neural networks, a generator and a discriminator, that are trained together. The generator creates images, and the discriminator evaluates them. Over time, the generator becomes skilled at producing cartoon-like images that are indistinguishable from those created by human artists.

Edge Detection and Simplification: Traditional image processing techniques like edge detection and color simplification are also employed. These methods involve detecting the edges within an image and simplifying the color palette to create a flat, cartoon-like appearance.

Software Tools and Libraries: There are numerous software tools and libraries available for creating cartoon images. These range from standalone applications to plugins for popular photo editing software. Examples include Adobe Photoshop with cartoon filters, and specialized software like Cartoonify.

Applications: Cartoon image generators are used in a variety of applications, including social media filters, video games, digital art, and animation. They provide a way for users to transform photos into artistic renditions with minimal effort

Ethical Considerations: As with many AI technologies, there are ethical considerations around the use of cartoon image generators. Issues such as the potential for misuse, the impact on professional artists, and the need for transparency in AI-generated content are important to address.

By leveraging these techniques and tools, cartoon image generators are able to produce stylized and creative renditions of photographs, offering users a fun and often artistic way to transform their images.

1.2 Why

Choosing to use a cartoon image generator can be beneficial for several reasons:

Creativity and Artistic Expression: Cartoon image generators allow users to transform ordinary photos into unique and artistic renditions. This can be a fun way to explore creativity and add a personal touch to images.

Visual Appeal: Cartoon images often have a charming and whimsical quality that can make them more visually appealing than regular photos. This can be especially useful for social media, marketing, and personal branding.

Simplification and Focus: Cartoon images can simplify complex scenes, removing unnecessary details and focusing on essential elements. This can make the main subject stand out more clearly and be easier to comprehend.

Versatility in Applications: Cartoon images are versatile and can be used in various contexts such as avatars, profile pictures, greeting cards, digital art, illustrations, and more. They can be a fun addition to personal projects or professional designs.

Enhanced Engagement: On social media and other platforms, cartoon images can enhance engagement by attracting more attention and encouraging interaction. People may find cartoon images more shareable and relatable

Cost and Time Efficiency: Hiring a professional artist to create custom cartoons can be expensive and time-consuming. Cartoon image generators provide a quick and cost-effective alternative, allowing users to produce cartoon-style images instantly.

Accessibility for Non-Artists: Cartoon image generators make it possible for individuals who may not have artistic skills to create cartoon-style images. This democratizes the process of creating art and allows more people to participate.

Fun and Entertainment: Using a cartoon image generator can simply be enjoyable. It provides a playful way to experiment with photos and see them transformed into something new and exciting.

Privacy and Anonymity: For those who prefer not to share real photos online, cartoon images can provide a level of privacy and anonymity while still allowing for visual representation.

Consistency in Style: For businesses or content creators, using a cartoon image generator can help maintain a consistent visual style across different media. This can be important for branding and creating a cohesive look.

1.3 Problem Statement

In today's digital age, the demand for visually appealing and engaging content has skyrocketed. Whether for social media, marketing, digital art, or personal use, users are constantly seeking creative ways to transform ordinary photos into unique and captivating images. Traditional methods of creating cartoon images, such as manual drawing or hiring professional artists, can be time-consuming, costly, and inaccessible for many individuals. Therefore, there is a need for an automated solution that can quickly and easily generate high-quality cartoon images from photos.

1.4 Objectives

To develop an automated cartoon image generator that can transform digital photos into cartoon-style images with high accuracy and aesthetic appeal. The solution should be user-friendly, efficient, and capable of producing consistent results across a wide range of input images.

Key Requirements:

1. **User-Friendly Interface:**
    - Design an intuitive interface that allows users to upload photos and customize cartoonization settings with minimal effort.
    - Provide real-time previews of the cartoonized image to help users make adjustments before finalizing.
2. **High-Quality Output:**
    - Ensure that the cartoonized images retain key facial features and details while exhibiting the distinctive characteristics of cartoon art, such as simplified lines and vibrant colors.
    - Support multiple cartoon styles to cater to different user preferences (e.g., comic book style, anime style, sketch style).
3. **Efficiency:**
    - Develop the system to process and generate cartoon images quickly, minimizing wait times for users.
    - Optimize the solution for various device capabilities, ensuring it runs smoothly on both desktop and mobile platforms.
4. **Customization Options:**
    - Provide users with customization options such as adjusting line thickness, colour palettes, and background effects.
    - Allow for additional artistic effects like adding text bubbles, stickers, and filters.
5. **Scalability:**
    - Design the system to handle a high volume of users and image processing requests simultaneously without degradation in performance.
    - Ensure that the solution can be easily integrated into existing applications and platforms through APIs.

6. **Privacy and Security:**

    - Implement robust measures to ensure the privacy and security of user-uploaded photos.
    - Ensure that no user data is stored without explicit consent and that all processed images are securely handled.

CHAPTER 2

LITERATURE SURVEY

2.1 MD.Salar Mohammad, Bollepalli Pranitha,Shivani Goud Pandula, Pulakanti Teja Sree.(2021). "Cartoonizing the Image" an International Journal of Advanced Trends in Computer Science and Engineering, Volume 10, No.4, July – August 2021.

In this paper, Processing technique is used to cartoonize an image.Image Processing-In the field of the research processing of an image consisting of identifying an object in an image identify the dimensions, no of objects, changing the images to blur effect and such effects are highly appreciated in the modern era of media and communication. There are multiple properties in the Image Processing.Each of the property estimates the image to be produced more with essence and sharper image.Each image is examined to various grid. Each picture element together is viewed as a 2-D Matrix.With each of the cell store different pixel values corresponding to the each of the picture element.

2.2 S.Rajatha, Anusha Shrikant Makkigadde, Neha L. Kanchan, Sapna, K. Janardhana Bhat,"Cartoonizer: Convert Images and Videos to Cartoon-Style Images and Videos" an International Journal of Research in Engineering, Science and Management Volume 4, Issue 7, July

In this paper, Cartoon GAN, where GAN stands for Generative Adversarial Network is used to transform images (snapshots) to the finest cartooned image (animated image). With the help of the loss function and of two types named as Adversarial loss and content loss, we got a flexible as well as a clear edge defined images. Also, with the help of CV2 which is Computer Vision, we have transformed video into animation (cartoonized video).

2.3 Zengchang Qin, Zhenbo Luo, Hua Wang, " Auto-painter: Cartoon Image Generation from Sketch by Using Conditional Generative Adversarial Networks", International Conference on Image Processing, 2017.

Recently, realistic image generation using deep neural network has become a hot topic in machine learning and computer vision. In this paper, the sketch-to-image synthesis problem is investigated by using Conditional Generative Adversarial Networks (CGAN). The new model is not only capable of painting hand-draw sketch with proper colors, but also allowing users to indicate preferred colours. Experimental results on two sketch datasets show that the auto-painter performs better that existing image-to-image methods. Auto-painter is a supervised learning model, given a black-and-white sketch, the model can generate a painted colorful image based on given sketchimage pairs in the training data.

CHAPTER 3

METHODOLOGY

3.1 Model Training

1. Model Architecture:

Generator and Discriminator:

- Generator: Creates anime-style images from real-world photos. It typically uses a U-Net structure or similar, with skip connections to preserve details across different layers.
- Discriminator: Distinguishes between real anime images and generated images. It is usually a convolutional neural network (CNN) that classifies images as real or fake, providing feedback to the generator.
- Network Design: The generator network transforms real-world photos into anime-style images by learning to map the input images to the target style, while the discriminator network improves the generator's output quality by identifying discrepancies between real and generated images.

2. Loss Functions:

- Adversarial Loss: Encourages the generator to produce images that the discriminator cannot distinguish from real anime images. This loss function drives the generator to improve continuously, as it tries to "fool" the discriminator.
- Content Loss: Ensures the generated images retain the essential content and structure of the input photos. This loss compares the generated image with the original photo to ensure that key features and details are preserved.
- Style Loss: Captures the artistic elements unique to anime by comparing feature representations of the generated images with those of real anime images. This loss ensures that the generated images adopt the desired stylistic elements, such as color schemes and line styles.

3. Training Procedure:

- Initialization: Initialize model parameters using techniques like Xavier or He initialization. Proper initialization helps the model start with reasonable weight values, facilitating faster convergence.

- Training Loop:

  i. Alternating Updates: Train the discriminator and generator alternately. For each iteration, update the discriminator using real and generated images, then update the generator using the gradients from the discriminator.

  ii. Stabilization: Use techniques like gradient clipping or feature matching to stabilize training. These techniques prevent the gradients from exploding or vanishing, ensuring smooth training progress.

4. Learning Rate:

- Schedule: Start with a relatively high learning rate (e.g., 0.0002) and progressively decrease it using learning rate decay or a scheduler. This approach helps the model converge more quickly initially and fine-tunes it as training progresses.

5. Batch Size:

- Memory Consideration: Choose a batch size that fits your GPU memory while providing sufficient gradients for training. Common batch sizes range from 8 to 64, depending on the available hardware. Larger batch sizes can improve gradient estimates but require more memory.

6. Epochs:

- Sufficient Training: Train the model for several thousand epochs until the loss stabilizes and the quality of the generated images is satisfactory. Monitor the training process to avoid overfitting or underfitting, adjusting the number of epochs as needed.

3.2 System Architecture:

The system architecture for a project focused on cartoon image generation using AnimeGAN involves several key components, each playing a specific role in the process. Here's a brief overview of the typical architecture:

1. Input Layer

- Image Data: High-resolution images are collected as input for the model. These images can be of various scenes, objects, or character

2. Preprocessing Layer

- Normalization: Input images are normalized to a specific range, often between -1 and 1, to standardize the input data for the model.

- Resizing: Images are resized to a fixed dimension to ensure uniformity, typically required by the neural network architecture.

3. Generator Network (G)

- Architecture: The generator network is a deep convolutional neural network (DCNN) designed to convert input images to anime-style images.
- Layers: It includes convolutional layers, residual blocks, and transposed convolutional layers. Residual blocks help in maintaining image features and style transfer.

4. Discriminator Network (D)

- Architecture: The discriminator network is another DCNN that distinguishes between real anime images and generated images.
- Layers: It consists of convolutional layers and fully connected layers that output a probability score indicating the authenticity of the input image.

5. Loss Functions

- Adversarial Loss: Used to train the generator and discriminator in a competitive manner. The generator aims to fool the discriminator, while the discriminator aims to correctly identify real vs. generated images.
- Content Loss: Ensures that the generated image maintains the content structure of the input image.
- Style Loss: Ensures that the generated image adheres to the desired anime style.

6. Training Phase

- Dataset: A large dataset of anime images and their corresponding real-world images is used.
- Training Loop: The generator and discriminator are trained iteratively. The generator's goal is to minimize the adversarial loss, while the discriminator's goal is to maximize it.
- Optimizers: Typically, Adam or other stochastic gradient descent-based optimizers are used to update the model weights.

7. Post-Processing Layer

- Image Enhancement: After generation, the images may undergo additional processing such as sharpening, contrast adjustment, or filtering to enhance visual quality.

8. Output Layer

- Generated Images: The final output is a set of anime-styled images that are visually similar to the input images but stylized according to the learned anime features.

System Flow

1. Input Image Collection: Gather and preprocess input images.
2. Model Inference: Use the trained generator to convert input images to anime style.
3. Post-Processing: Enhance the quality of the generated images if needed.
4. Output: Save and display the anime-styled images.

This architecture ensures that the system can efficiently convert real-world images into high-quality anime-styled images by leveraging deep learning techniques and adversarial training.

Flowchart explanation:

Here is a flowchart outlining the system architecture for a cartoon image generation project using AnimeGAN:

Description of Flowchart:

1. Start: The process begins.

2. Collect Input Images: Gather images to be converted into anime style.

3. Preprocess Images: Normalize and resize images to the required input format for the model.

4. Load Pre-trained Model: Load a pre-trained AnimeGAN model.

5. Generate Anime Images using Generator Network (G): Use the generator network to convert the input images into anime style.

6. Discriminator Network (D): Use the discriminator to evaluate the authenticity of the generated images.

7. Compute Loss: Calculate the adversarial, content, and style loss to guide the training process.

8. Optimize Networks: Update the weights of the generator and discriminator networks using gradient descent.

9. Post-Process Generated Images: Enhance the visual quality of the generated anime images.

10. Save and Display Images: Save the final anime-styled images and display them.

11. End: The process concludes.

This flowchart visually represents the step-by-step process of generating cartoon images using the AnimeGAN framework, emphasizing key stages such as data preprocessing, model inference, and post-processing.

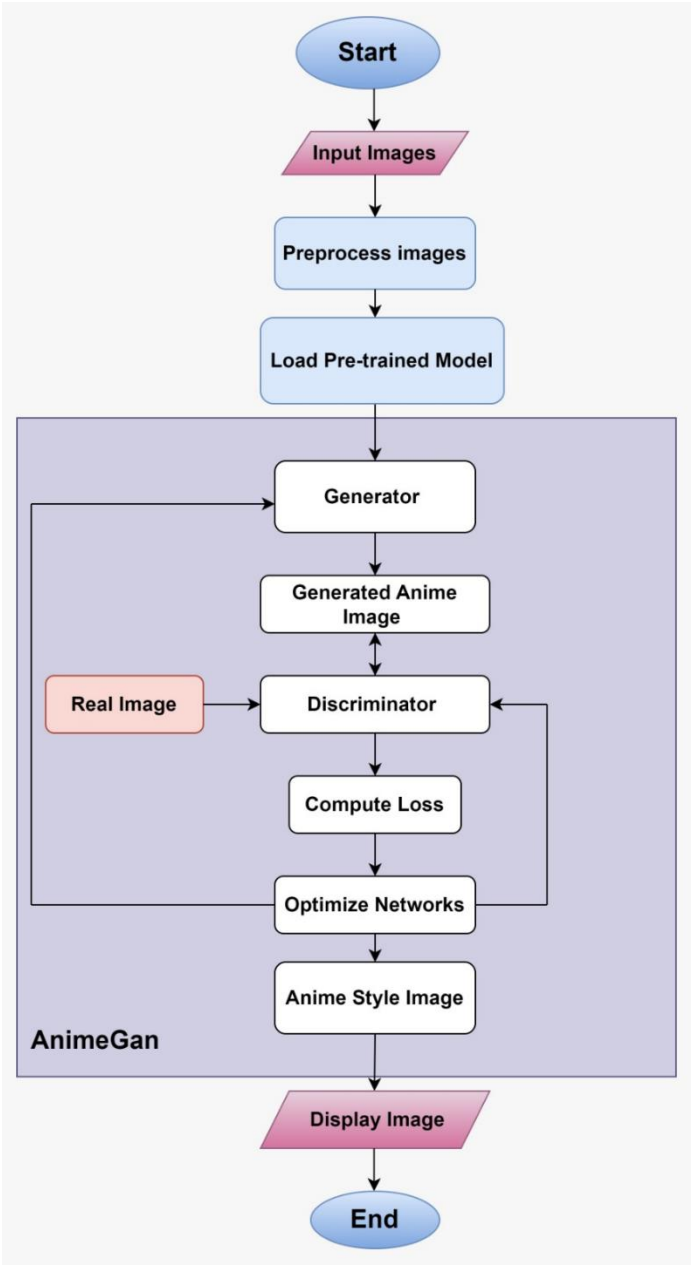

Figure: 3.1 System Architecture

3.3 Tools and Technologies

Hardware Requirements:

1. **Computer with a GPU**: A powerful graphics processing unit (GPU) is essential for handling the intensive computations required by many cartoon generation models.

2. **Sufficient RAM**: At least 16 GB of RAM is recommended, though more may be necessary for handling large datasets.

3. **Storage Space**: Adequate storage (SSD preferred) for datasets, model weights, and generated images. Aim for at least 500 GB.
4. **High-Resolution Monitor**: Useful for visualizing and editing images.

Software Requirements:

1. **Operating System**: Linux (Ubuntu is popular), Windows, or macOS.
2. **Programming Languages**: Python is the most commonly used language due to its extensive libraries and frameworks.
3. **Libraries and Frameworks**:
    - **TensorFlow or PyTorch**: Popular frameworks for building and training neural networks.
    - **OpenCV**: For image processing tasks.
    - **Numpy**: For numerical computations.
    - **Matplotlib or Seaborn**: For plotting and visualizing data.
4. **Deep Learning Models**:
    - **Generative Adversarial Networks (GANs):** Such as DCGAN, CycleGAN, or StyleGAN, which are often used for image generation tasks.
    - **Pre-trained Models:** Models like VGG, ResNet for feature extraction or as a backbone for transfer learning.
5. **Image Processing Tools**: Software like Adobe Photoshop or GIMP for manual touch-ups and adjustments.
6. **Dataset**: A large and diverse dataset of images to train the model. Datasets like CIFAR-10, CelebA, or custom datasets specific to the style you want to generate.

CHAPTER 4

IMPLEMENTATION

4.1 Steps Followed

1. Set Up the Environment: Install required libraries.

2. Load the ONNX Model: Load the pre-trained ONNX model using onnxruntime.

3. Preprocess Input Image: Prepare the input image by resizing, normalizing, and converting it to the format expected by the model.

4. Run the Model: Pass the preprocessed image through the model to get the generated output.

5. Postprocess Output Image: Convert the model's output tensor back to a displayable image format.

6. Display or Save the Generated Image: Show or save the cartoon image generated by the model.

4.2 Code Snippet

Back-end:

```
# pip install flask onnxruntime opencv-python-headless pillow numpy tqdm
# run this in terminal before running the script

from flask import Flask, request, render_template, send_file, redirect, url_for
import onnxruntime as ort
import cv2
import numpy as np
import os
from PIL import Image
app = Flask (__name__, static_url_path='/outputs', static_folder='outputs')


pic_form = ['.jpeg', '.jpg', '.png', '.JPEG', '.JPG', '.PNG']
device_name = ort.get_device()
if device_name == 'CPU':
    providers = ['CPUExecutionProvider']
elif device_name == 'GPU':
    providers = ['CUDAExecutionProvider', 'CPUExecutionProvider']
```

```
# model = 'Paprika_54'
# session = ort.InferenceSession(f'./{model}.onnx', providers=providers)

def process_image(img, x32=True):
    h, w = img.shape[:2]
    if x32:
        def to_32s(x):
            return 256 if x < 256 else x - x % 32
        img = cv2.resize(img, (to_32s(w), to_32s(h)))
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB).astype(np.float32) / 127.5 - 1.0
    return img

def load_test_data(image_path):
    img0 = cv2.imread(image_path).astype(np.float32)
    img = process_image(img0)
    img = np.expand_dims(img, axis=0)
    return img, img0.shape[:2]

def Convert(img, scale,model):
    session = ort.InferenceSession(f'./{model}.onnx', providers=providers)
    x = session.get_inputs()[0].name
    y = session.get_outputs()[0].name
    fake_img = session.run(None, {x: img})[0]
    images = (np.squeeze(fake_img) + 1.) / 2 * 255
    images = np.clip(images, 0, 255).astype(np.uint8)
    output_image = cv2.resize(images, (scale[1], scale[0]))
    return cv2.cvtColor(output_image, cv2.COLOR_RGB2BGR)

@app.route('/')
def upload_form():
    return render_template('index.html')

@app.route('/', methods=['POST'])
def upload_file():
    if 'file' not in request.files:
        return 'No file part'
    model=request.form['model']
    file = request.files['file']
    if file.filename == '':
        return 'No selected file'
```

```
    if file and allowed_file(file.filename):
        filename = file.filename
        filepath = os.path.join('uploads', filename)
        file.save(filepath)

        img, scale = load_test_data(filepath)
        output_img = Convert(img, scale, model)
        output_path = os.path.join('outputs', filename)
        cv2.imwrite(output_path, output_img)

        return render_template('index.html',filename=filename)

def allowed_file(filename):
    return any(filename.endswith(ext) for ext in pic_form)

if __name__ == "__main__":
    os.makedirs('uploads', exist_ok=True)
    os.makedirs('outputs', exist_ok=True)
    app.run(host='0.0.0.0', port=5000, debug=True)
```

Front-end:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>AnimeGANv2 Converter</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f0f0f5;
            margin: 0;
            padding: 20px;
            display: flex;
            justify-content: center;
            align-items: center;
            flex-direction: column;
        }
        h1 {
            color: #333;
```

```
        }
        form {
            margin: 20px 0;
            padding: 20px;
            border-radius: 10px;
            background-color: #fff;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
        }
        input[type="file"], select {
            margin: 10px 0;
            padding: 10px;
            border: 1px solid #ccc;
            border-radius: 5px;
            width: 100%;
            box-sizing: border-box;
        }
        input[type="submit"] {
            background-color: #007bff;
            color: #fff;
            border: none;
            padding: 10px 20px;
            border-radius: 5px;
            cursor: pointer;
        }
        input[type="submit"]:hover {
            background-color: #0056b3;
        }
        .image-container {
            margin: 20px 0;
            text-align: center;
        }
        .image-container img {
            max-width: 90%;
            height: 300px;
            border-radius: 20px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
        }
    </style>
</head>
<body>
```

```
    <h1>AnimeGANv2 Converter</h1>

    <form action="/" method="post" enctype="multipart/form-data">
        select a your anime generation model
        <select name ="model">
            <option value="Hayao_64">Hayo-64 AnimeGan</option>
            <option value="Hayao-60">Hayo-60 AnimeGan</option>
            <option value="Paprika_54">PaprikaAnimeGan</option>
            <option value="Shinkai_53">ShinkaiAnimeGan</option>
        </select>
        <input type="file" name="file" required>

        <input type="submit" value="Upload and Convert">
    </form>

    <div class="image-container">
    <h2>Converted Image</h2>
    <img src="{{ url_for('static', filename=filename) }}" alt="Converted Image">

</div>
</body>
</html>
```

CHAPTER 5

RESULTS AND DISCUSSION

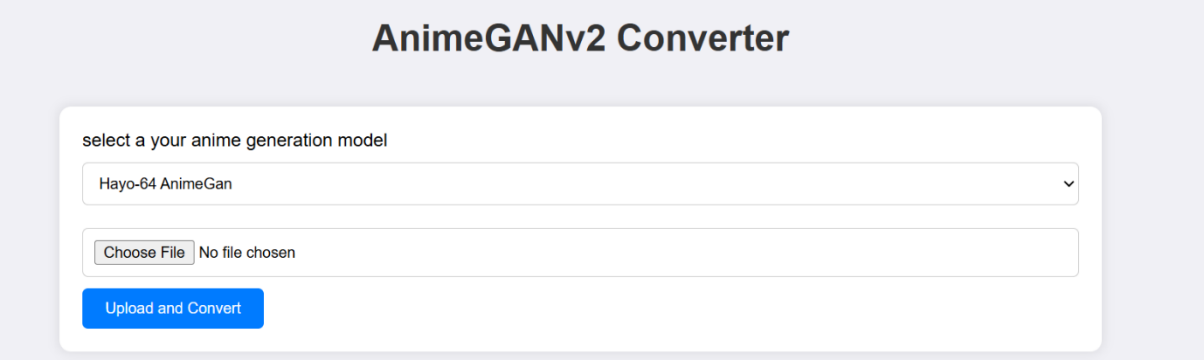

Figure: 5.1 Display Screen

The image shows a user interface for an AnimeGANv2 Converter, which is a tool used to convert images into anime-style images. The interface includes:

1. A dropdown menu to select the anime generation model, currently set to "Hayo-64 AnimeGan."
2. An option to choose a file for conversion.
3. A button labelled "Upload and Convert" to start the conversion process.

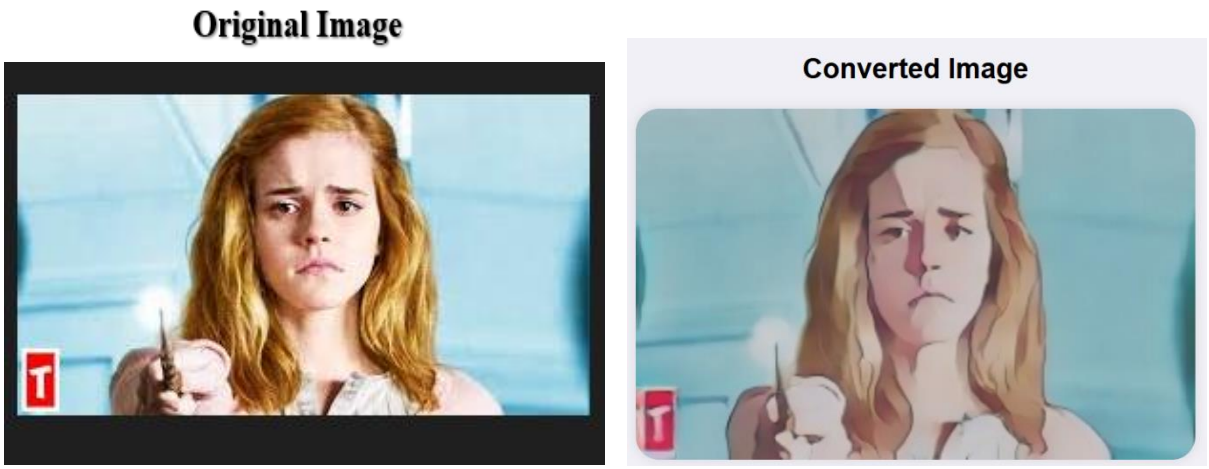

Figure: 5.2 Original Image VS Cartoonized Image

An original image refers to the real-world photograph or artwork in its natural form. A cartoonized image is the same image transformed using a cartoon generation model, such as AnimeGANv2, to appear in an anime or cartoon style.

CONCLUSION AND FUTURE WORK

The Main theme of this paper is to get high quality cartoonized image from the normal image. And this can be done using different techniques and methods as discussed in above papers. Generally, to convert any general image into cartoon, it requires more skills, technique and also consumes much time to get a proper cartoon but now a days with developed technology it became easy. The time required in this method is comparatively very less. Hence the latest methods became popular for getting cartoon images. There are different techniques for converting an image to cartoon. Based on User requirements, suitable technique is used to get an efficient output. And with developing technology many new methods/techniques are introduced for better implementation and best results. Even Videos can be animated using these techniques.So with this different algorithms are implemented. Finally, it can be concluded that an image can be converted into a cartoonized image with suitable techniques which gives efficient output and less time consumption. Though many applications are developed for Cartoonizing, implementation using Python or any other language gives better understanding and better outputs.

Future Work

The future work for cartoon image generators will likely focus on several key areas:

1. Enhanced Realism and Detail: Advances in AI and machine learning will enable cartoon image generators to create highly detailed and realistic images, pushing the boundaries of what is traditionally considered a "cartoon."

2. Greater Customization: Future generators will offer more sophisticated customization options, allowing users to specify intricate details such as character emotions, clothing, backgrounds, and artistic styles.

3. Dynamic Animations: Beyond static images, future work will likely include the ability to generate dynamic, animated sequences, enabling users to create full-length animations and short films with minimal effort.

4. User-Friendly Interfaces: Development will focus on making these tools more intuitive and accessible to a broader audience, including those without technical expertise, fostering creativity across all skill levels.

REFERENCES

[1]. MD. Salar Mohammad, Bollepalli Pranitha,Shivani Goud Pandula, Pulakanti Teja Sree.(2021). “Cartoonizing the Image” an International Journal of Advanced Trends in Computer Science and Engineering, Volume 10, No.4, July – August 2021.

[2]. S. Rajatha, Anusha Shrikant Makkigadde, Neha L. Kanchan, Sapna, K. Janardhana Bhat,” Cartoonizer: Convert Images and Videos to Cartoon-Style Images and Videos” an International Journal of Research in Engineering, Science and Management Volume 4, Issue 7, July 2021.

[3]. Zengchang Qin, Zhenbo Luo, Hua Wang, " Auto-painter: Cartoon Image Generation from Sketch by Using Conditional Generative Adversarial Networks”, International Journal on Image Processing, 2017.

[4]. Harshitha R,Kavya S Muttur,Prof.Jyothi Shetty Dept. Computer Science,RV College of Engineering,Bangalore . “Cartooniation Using White-box Technique in Machine Learning” (2020).

[5]. Akanksha Apte, Ashwathy Unnikrishnan, Navjeevan Bomble, Prof. Sachin Gavhane, “Transformation of Realistic Images and Videos into Cartoon Imagesand Video using GAN” an International Journal of Research in Engineering, Science and Management Volume 4, Issue 7, July2021.

[6]. Silviya D’monte, Aakash Varma, Ricky Mhatre, Rahul Vanmali, Yukta sharma.

[7]. Chinmay Joshi, Devendra Jaiswal, Akshata Patil Department Name of Organization: Information Technology Name of Organization: Kc College of Engineering Management Studies and Research City: Kopari Thane (East) Country: India.