JOJOGAN FACE STYLIZATION REAL TIME

Thu Ya Hlaing - 6511238
Ye Myat Moe - 6511233

TABLE OF CONTENT

INTRODUCTION

- Transforms webcam video feed into an anime-style image

KEY FEATURES

- Real-time processing.
- Simple and user-friendly interface.
- High-quality stylization using neural networks.

METHODOLOGY

01.

Data Processing

- Input: Webcam feed or uploaded images.
- Preprocessing: Normalization, resizing, and format conversion.

02.

Stylization Process

- Model Inference: The input image is passed through a custom deep learning model to apply the anime style.
- Post-processing: The stylized image is denormalized and displayed in real-time.

03.

User Interface

- Built with Tkinter for an interactive GUI.
- Buttons for uploading images and controlling real-time video processing.

MODEL

01.

Architecture

- Base Network: Convolutional Neural Network (CNN) with Residual Blocks.
- Downsampling: DownBlock layers reduce the image resolution while increasing feature depth.
- Upsampling: UpBlock layers restore the resolution to the original size, applying the stylization.
- Residual Blocks: Enhance the quality of the stylization by preserving key features.

02.

Training

- The model is trained on a dataset of images paired with their stylized counterparts.
- Loss function: Mean Squared Error (MSE) to minimize the difference between the stylized output and the target style.

03.

Implementation

- Written in PyTorch for flexibility and efficiency.
- Utilizes pretrained weights for fast and reliable inference.

MODEL

01.

Architecture

- Base Network: Convolutional Neural Network (CNN) with Residual Blocks.
- Downsampling:DownBlocklayers reduce the image resolution while increasing feature depth.
- Upsampling: UpBlock layers restore the resolution to the original size, applying the stylization.
- Residual Blocks: Enhance the quality of stylization by preserving key features.

02.

Training

- The model is trained on a dataset of images paired with their stylized counterparts.
- Loss function: Mean Squared Error (MSE) to minimize the difference between the stylized output and the target style.

03.

Implementation

- Written in PyTorch for flexibility and efficiency.
- Utilizes pretrained weights for fast and reliable inference.

CODE OVERVIEW

01.

Main Components

- Model Definition: ResBlock, DownBlock, UpBlock, and SimpleGenerator classes.
- Stylization Logic: Forward pass through the model to generate stylized images.
- User Interface: Tkinter-based application to handle image input, display, and real-time processing.

02.

Key Functions

- **load_model:** Loads pretrained weights for the model.
- **stylize_image**: Handles image preprocessing, model inference, and post-processing.
- **update_frame**: Continuously captures webcam feed and applies the stylization in real-time.

CODE

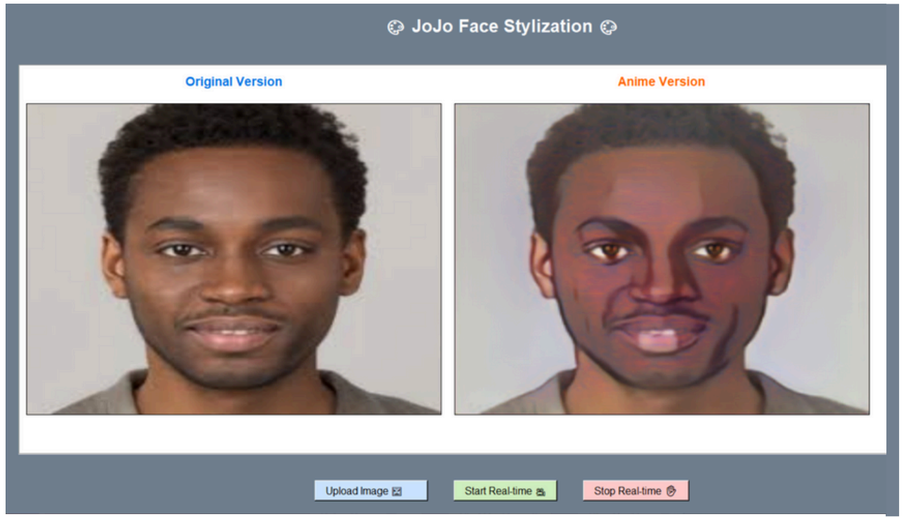
```python
class SimpleGenerator(nn.Module):
    def __init__(self, num_channel=32, num_blocks=4):
        super(SimpleGenerator, self).__init__()
        self.down1 = DownBlock(3, num_channel)
        self.down2 = DownBlock(num_channel, num_channel * 2)
        self.down3 = DownBlock(num_channel * 2, num_channel * 3)
        self.down4 = DownBlock(num_channel * 3, num_channel * 4)
        res_blocks = [ResBlock(num_channel * 4)] * num_blocks
        self.res_blocks = nn.Sequential(*res_blocks)
        self.up1 = UpBlock(num_channel * 4, num_channel * 3)
        self.up2 = UpBlock(num_channel * 3, num_channel * 2)
        self.up3 = UpBlock(num_channel * 2, num_channel)
        self.up4 = UpBlock(num_channel, 3, is_last=True)

    def forward(self, inputs):
        down1 = self.down1(inputs)
        down2 = self.down2(down1)
        down3 = self.down3(down2)
        down4 = self.down4(down3)
        down4 = self.res_blocks(down4)
        up1 = self.up1(down4)
        up2 = self.up2(up1 + down3)
        up3 = self.up3(up2 + down2)
        up4 = self.up4(up3 + down1)
        return up4
```

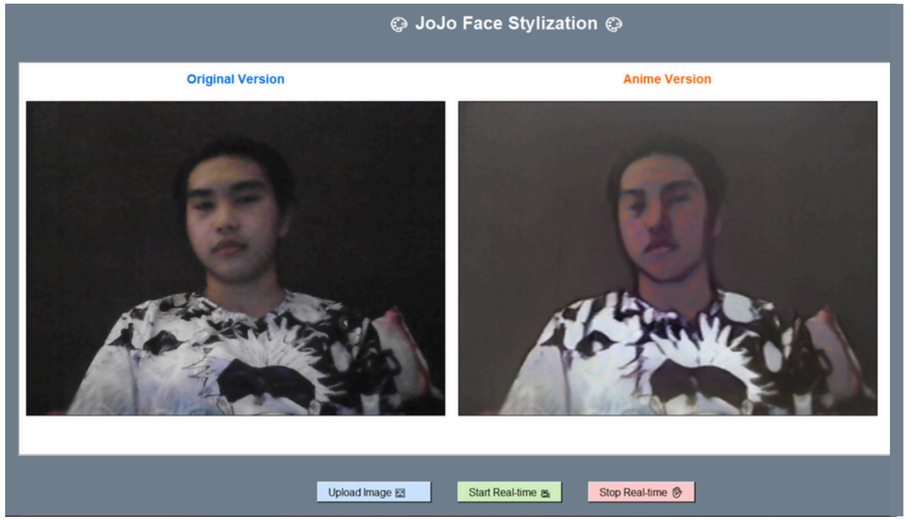
```python
def load_model(self):
    model = SimpleGenerator()
    if not os.path.exists('weight.pth'):
        print("Error: weight.pth file not found.")
        return None
    model.load_state_dict(torch.load('weight.pth', map_location='cpu'))
    model.eval()
    return model
```

RESULT

ONE-SHOT



REAL-TIME



REFERENCE

- https://blog.paperspace.com/one-shot-face-stylization-with-jojogan/
- https://github.com/mchong6/JoJoGAN
- https://research.google/blog/mediapipe-facestylizer-on-device-real-time-few-shot-face-stylization/

THANK YOU FOR LISTENING

https://github.com/YeMyat144/face-style