Conversion of Videos From SD to HD Resolution Using Diffusion

A project by Suraj Pradeepkumar

Introduction

The aim is to develop a prototype of a video conversion process that can convert SD resolution (640 x 480px) videos to HD resolution (1280 x 720px) videos using a diffusion type model. The tool should be able to fill in the blank areas on the left and right sides of the video with relevant pixels or image parts while preserving the context of the video.

To do the same, we use the model `stabilityai/stable-diffusion-x4-upscaler` for this project. Let's dive deeper into the approach. We use both CPU and GPU to complete this process so that the system memory is taken into consideration.

The approach

Step1: Environment Setup

Here we import the libraries after we install the necessary libraries into ther jupyter notebook environment.The code below shows the libraries that we imported.

```
import torch
from diffusers import StableDiffusionUpscalePipeline
from moviepy.editor import VideoFileClip, ImageClip,
concatenate_videoclips
from PIL import Image
import os
```

Step2: Model Loading

In this step, we import the pre-trained `**stabilityai/stable-diffusion-x4-upscaler**`model. The model is loaded with mixed precision to optimize memory usage and computation speed. The code is as follows:

```
# Load the pre-trained pipeline with mixed precision
pipeline = StableDiffusionUpscalePipeline.from_pretrained(
    "stabilityai/stable-diffusion-x4-upscaler",
    torch_dtype=torch.float16
)
```

Step3: Frame Upscaling Function

Next, a function is defined to upscale individual frames of the video. Each frame is resized to 640x480 pixels (assuming the input is in SD resolution) before being passed through the upscaling pipeline. The upscaled frames are then saved in a directory for later use. The function is as follows:

```
# Function to upscale a single frame
def upscale_frame(frame, pipeline, prompt):
    img = Image.fromarray(frame).convert("RGB")
    low_res_img = img.resize((640, 480))  # assuming the input frame
is in SD resolution
    upscaled_image = pipeline(prompt=prompt,
image=low_res_img).images[0]
    return upscaled_image
```

Step4: Video Upscaling Function

The video upscaling function iterates over the input video, processing frames in batches for memory optimization. Each batch of frames is passed through the **`upscale_frame`** function, and the upscaled frames are saved to disk. After all frames are processed, the upscaled frames are concatenated into a new video file. The function is as follows:

```
# Function to upscale a video
def upscale_video(input_path, output_path, pipeline, prompt,
batch_size=2):
    video = VideoFileClip(input_path)
    upscaled_frames = []

    # Create directory for upscaled frames
    if not os.path.exists("frames"):
        os.makedirs("frames")

    for batch_start in range(0, int(video.duration * video.fps),
batch_size):
        batch_end = min(batch_start + batch_size, int(video.duration *
video.fps))
        for i in range(batch_start, batch_end):
            frame = video.get_frame(i / video.fps)
            upscaled_frame = upscale_frame(frame, pipeline, prompt)
            upscaled_frames.append(upscaled_frame)
            upscaled_frame.save(f"frames/frame_{i:04d}.png")
        torch.cuda.empty_cache()  # Clear cache after each batch
```

```
    # Create video from upscaled frames
    out_video = concatenate_videoclips(
        [ImageClip(f"frames/frame_{i:04d}.png").set_duration(1 /
video.fps) for i in range(len(upscaled_frames))]
    )
    out_video.write_videofile(output_path, fps=video.fps)
```

Step5: Upscaling the video

Finally, the `upscale_video` function is called with the input and output paths of the video, along with the loaded upscaling pipeline and a prompt if needed. This function processes the entire video, upscales each frame, and generates an upscaled video file. The final code is as follows:

```
# Upscale the video
upscale_video(input_video_path, output_video_path, pipeline, prompt)
```

Results and Analysis

GPU Utilization

The use of a GPU significantly enhances the processing speed. By offloading computationally intensive tasks to the GPU, we achieve faster frame processing and overall reduced time for video upscaling. The GPU's parallel processing capabilities are ideal for handling the complex computations required by the stable diffusion model.

Quality of Upscaled Video

The `stabilityai/stable-diffusion-x4-upscaler` model is designed to enhance the resolution and detail of images. In this project, each frame of the video is processed individually, ensuring consistent quality across the entire video. The upscaled video demonstrates a notable improvement in sharpness, clarity, and overall visual quality compared to the original SD video.

Frame-by-Frame Processing

Each frame of the input video is resized to 480p before being fed into the model. This resizing step is crucial as it prepares the frames for the model's input requirements. The stable diffusion model then upscales these frames to 720p, effectively increasing the resolution by a factor of four. This upscaling process preserves the original content while enhancing the detail and quality.

Advantages of Using Stable Diffusion Model

- **High-Quality Upscaling**: The model produces high-quality frames with enhanced detail and resolution.
- **Efficiency**: Leveraging GPU capabilities significantly reduces the time required for the upscaling process.
- **Consistency**: The frame-by-frame processing ensures consistent quality throughout the video.

Potential Improvements

- **Batch Processing**: Implementing batch processing of frames could further enhance efficiency.
- **Optimization**: Fine-tuning model parameters and exploring different upscaling techniques could lead to even better results.
- **Post-Processing**: Additional post-processing steps, such as color correction and noise reduction, could further improve the final video quality.

Future Work

Future work could explore the integration of additional machine learning models for even higher resolution upscaling. Additionally, expanding the pipeline to support different video formats and resolutions would make the tool more versatile. Investigating real-time upscaling capabilities could also open up new applications for live video streaming and broadcasting.

By leveraging the advanced capabilities of the `stabilityai/stable-diffusion-x4-upscaler` model and the computational power of GPUs, this project showcases an effective method for enhancing video resolution, transforming SD videos into high-quality HD content.

Conclusion

This project successfully demonstrates the use of the `stabilityai/stable-diffusion-x4-upscaler` model to convert an SD video (480p) to HD (720p). The process involves extracting frames from the input video, upscaling each frame using the stable diffusion model, and compiling the upscaled frames into a new video file.

Important Links

Project Github: https://github.com/Svrajj/SD2HD-Convertor

Project Google Drive:
https://drive.google.com/drive/folders/13GL5p2c1FSgU4QOhemHB0NMLvxE8iM68?usp=sharing

My Resume: https://drive.google.com/file/d/1zsdMPzqLEE977HO7l7asES6cGiGM75dP/view?usp=sharing

My LinkedIn: https://www.linkedin.com/in/svraj/