1. Overall design

The project is based on a paper from CVPR2021 and the code is reproduced.

It enables free-gesture control when driving arbitrary speaking faces with audio. Instead of learning pose movements from audio, I utilize another pose source video to compensate for head movements only. The key is to design an implicit low-dimensional gesture code that does not contain mouth shape or identity information.

In this way, audiovisual representations are modularized into a space of three key factors: speech content, head posture, and identity information.

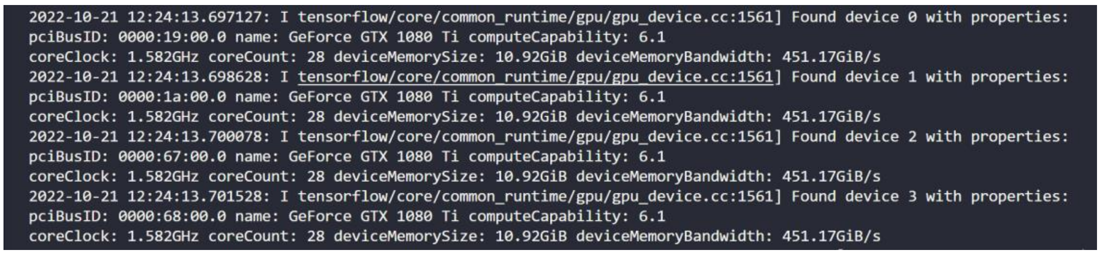
The program is mainly divided into model part, test part and auxiliary code part. The auxiliary part integrates functions through sequential calls to the program. The following mainly focuses on the process of using the trained model to generate audio and video driver speaking videos.

2. Development environment

The program uses Python 3.6 and Pytorch 1.3.0.

The other libraries mainly used and their minimum versions are as follows:
torch>=1.2.0
torchvision
dominate>=2.3.1
dill
scikit-image
numpy>=1.15.4
scipy>=1.1.0
matplotlib
opencv-python>=3.4.3.18
tensorboard==1.14.0
tqdm
librosa

The computing resources used for the program training are: GeForce GTX 1080 Ti. The testing process is performed using the CPU.



```
2022-10-21 12:24:13.697127: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1561] Found device 0 with properties:
pciBusID: 0000:19:00.0 name: GeForce GTX 1080 Ti computeCapability: 6.1
coreClock: 1.582GHz coreCount: 28 deviceMemorySize: 10.92GiB deviceMemoryBandwidth: 451.17GiB/s
2022-10-21 12:24:13.698628: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1561] Found device 1 with properties:
pciBusID: 0000:1a:00.0 name: GeForce GTX 1080 Ti computeCapability: 6.1
coreClock: 1.582GHz coreCount: 28 deviceMemorySize: 10.92GiB deviceMemoryBandwidth: 451.17GiB/s
2022-10-21 12:24:13.700078: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1561] Found device 2 with properties:
pciBusID: 0000:67:00.0 name: GeForce GTX 1080 Ti computeCapability: 6.1
coreClock: 1.582GHz coreCount: 28 deviceMemorySize: 10.92GiB deviceMemoryBandwidth: 451.17GiB/s
2022-10-21 12:24:13.701528: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1561] Found device 3 with properties:
pciBusID: 0000:68:00.0 name: GeForce GTX 1080 Ti computeCapability: 6.1
coreClock: 1.582GHz coreCount: 28 deviceMemorySize: 10.92GiB deviceMemoryBandwidth: 451.17GiB/s
```

3. Code Operation Process

This model can only process cropped data of VoxCeleb2 type, so the data needs to be preprocessed. This process requires the face-alignment library.

First, the three videos need to first generate the demo.csv file used by the test code through scripts/prepare_testing_files.py. The execution instruction is "python scripts/prepare_testing_files.py". The parameters in the py file are:

1). --src_pose_path: Indicates the drive pose source path. It can be an mp4 file or a folder containing frames in the format %06d.jpg starting at 0.

2). --src_audio_path: indicates the path of the audio source. It can be an "mp3" audio file or an "mp4" video file. If a video is given, frames will automatically be saved in /misc/Mouth_Source/video_name.

3). --src_mouth_frame_path: When --src_audio_path is not a video path, this flag can provide a folder containing video frames synchronized with the source audio.

4). --src_input_path: Enter the path of the reference image. When the path is a video file, we convert it into frames.
5). --csv_path: The path of the original data to be saved. When the video has passed the previous steps and is processed into the [name] folder through prepare_testing_files.py, by running "align_68.py --folder_path [name]" in the terminal, the three video images will be cropped and saved. In another "[name_cropped]" folder. Then manually change the demo.csv file or change the directory folder path and run the preprocessed file again.

Next, the first line in the demo_vox.sh script file modifies the final demo.csv file path and name, and runs experiments/demo_vox.sh. The execution command is "bash experiments/demo_vox.sh". The image generation results are finally saved in the results folder.

Finally, video splicing and voice fusion are performed on the three result picture folders through concave.ipynb.

4. Technical Difficulties

Although GPUs are used to process data and train models, model training is actually difficult. In addition, during the test process, the video of just a few seconds needs to be segmented into pose, mouth and input video frames through the cv2 library first, then the model is used to generate a new matching mouth shape for the input, and finally the three generated pictures are spliced together through ffmpeg. into a video, and then spliced into a comparative video with sound.