Jawer Echeverry & Rodrigo Nogueira

Draft Paper

Due Date: 04/18/2023

Machine Learning Project Part 2:

So-Vits-SVC

Using AI for Voice Synthesis

Voice synthesis with AI often involves the use of neural networks, particularly deep learning models. These neural networks are designed to learn and generate speech patterns by processing and understanding large amounts of audio data, from a dataset. Our example of a neural network used for voice synthesis will be a custom So-Vits-SVC model that we trained ourselves. It learns to generate human-like speech from raw reference vocals.

Training Process

https://colab.research.google.com/drive/1laRNiMSgSw_SxSnuti8oWIuC--RHzAGp#scrollTo=LS0OPRkL4Pme

Commands and Python Code:

```
#@title Clone github repo
!git clone https://github.com/effusiveperiscope/so-vits-svc -b eff-4.0

#@title Install dependencies
%cd /content/so-vits-svc
!pip install pyworld praat-parselmouth
!python -m pip install --upgrade pip
!pip install fairseq==0.12.2 librosa==0.8.1

#@title Download ContentVec model file
!wget -P hubert/ https://huggingface.co/therealvul/so-vits-svc-4.0-init/resolve/main/checkpoint_best_legacy_500.pt

#@title Mount google drive
#@markdown Mount google drive, which will be used to hold your dataset files.
from google.colab import drive
drive.mount('/content/drive')

#@title Load the dataset from .zip in Google Drive for preprocessing
```

```
#@markdown Name of the zip folder
DATASETNAME = "Chrysalis"  #@param {type:"string"}
#@markdown Zip path (usually do not need to change this unless you
uploaded to a different directory)
ZIP_PATH = "/content/drive/MyDrive/dataset/"  #@param {type:"string"}
ZIP_NAME = ZIP_PATH + DATASETNAME

!unzip -d /content/so-vits-svc/dataset_raw {ZIP_NAME}.zip
#@title Resample to 44.1k
!python resample.py
#@title Segment training set and generate configuration files
!python preprocess_flist_config.py

#@title Generate hubert and f0
!python preprocess_hubert_f0.py

#@title Save preprocessed dataset files to Google Drive (for training
convenience)
#Compress dataset folder
!zip -r dataset.zip /content/so-vits-svc/dataset
#@markdown Customize name of preprocessed dataset folder to avoid
conflicts.
dataset_name_drive = "44k_dataset"  #@param {type:"string"}
DATASET_PATH_DRIVE = "/content/drive/MyDrive/dataset/" +
dataset_name_drive
!mkdir -p {DATASET_PATH_DRIVE}

!cp /content/so-vits-svc/dataset.zip "{DATASET_PATH_DRIVE}"
!cp configs/config.json "{DATASET_PATH_DRIVE}"
!cp filelists/train.txt "{DATASET_PATH_DRIVE}"
!cp filelists/val.txt "{DATASET_PATH_DRIVE}"

#@title Load preprocessed dataset files from Google Drive (no need to run
first round)
#@markdown If you already have preprocessed dataset files on Google Drive,
you can load them here instead of re-running the preprocessing steps.
back_up_name = "three_dataset"  #@param {type:"string"}
BACK_UP_DATASET_PATH = "/content/drive/MyDrive/dataset/" + back_up_name
!unzip {BACK_UP_DATASET_PATH}/dataset.zip -d /
!cp {BACK_UP_DATASET_PATH}/config.json /content/so-vits-
svc/configs/config.json
!cp {BACK_UP_DATASET_PATH}/val.txt filelists/val.txt
!cp {BACK_UP_DATASET_PATH}/train.txt filelists/train.txt

#@title  Model saving/pretrained model preferences
```

```
Clone = "44k"
%cd /content/so-vits-svc

#@markdown **Save the trained model files to Google Drive instead of the colab runtime filesystem. You also need to check and execute this when resuming training. If you are short on Drive space you may want to uncheck this (in which case you must use the file menu to manually download the model checkpoints in so-vits-svc/logs/ yourself and copy the latest checkpoint back into so-vits-svc/logs/ to resume training).**
Save_to_drive = False #@param {type:"boolean"}
if Save_to_drive:
  !rm -rf /content/so-vits-svc/logs/"{Clone}"
  !mkdir -p /content/drive/MyDrive/"{Clone}"
  !ln -s /content/drive/MyDrive/"{Clone}" /content/so-vits-svc/logs/"{Clone}"

#@markdown **Download the pre-trained model for the first training. For continuing training, use the checkpoints saved by yourself after continuing to train.**
pre_pth = True #@param {type:"boolean"}
if pre_pth:
  !wget -O logs/"{Clone}"/G_0.pth -P logs/"{Clone}"/ https://huggingface.co/therealvul/so-vits-svc-4.0-init/resolve/main/G_0.pth
  !wget -O logs/"{Clone}"/D_0.pth -P logs/"{Clone}"/ https://huggingface.co/therealvul/so-vits-svc-4.0-init/resolve/main/D_0.pth

#@title  Start training
#@markdown **Start training**
Clone = "44k"

#@markdown **Enable tensorboard for data visualization**
tensorboard_on = True #@param {type:"boolean"}
if tensorboard_on:
  %load_ext tensorboard
  %tensorboard --logdir logs/"{Clone}"

!python train.py -c configs/config.json -m "{Clone}"
```

What code does:

Downloads ContentVec model file ##########

Connects to your Google drive to store new copies of trained model

Loads and unzips compressed folder of uploaded dataset. This is what the model trains off of.

Resamples dataset to 44.1k ##########

Generates new config file for current training session. This tells training progress, data and model values, and saves speaker name (name of person you are synthesizing).

Generates hubert and f0. Hubert is basically the pre-trained model that your new model uses as a reference.

Saves and loads the preprocessed dataset files for training convenience.

Saves new trained model files to mounted Google Drive fro backup purposes. Automatically downloads pre-trained model to start training (G and D files).

Starts training and turns on tensor board. Training basically means the AI model goes through the dataset a certain number of times, known as steps. Once the model reaches a milestone of 800 steps, it saves the new model (G and D files) into the mounted Google Drive. The tensor board is used to keep track of training progress in real-time, being able to hear inferences of how the model's voice is sounding during the training process. Users must manually delete old versions of model files from their Google Drives to maintain enough storage space. Once model reaches a certain amount of step intervals (typically 20,000 minimum) the AI voice model will start sounding more realistic, and ready to make inferences.

Inference Process

https://colab.research.google.com/drive/1xUK-bdsrGoV5PUJz3z1cArTH0of1CWON

Commands and Python Code:

```
#@title Clone Github Repository
!git clone https://github.com/effusiveperiscope/so-vits-svc -b eff-4.0

#@title Install Dependencies
%cd /content/so-vits-svc
!pip install pip==23.0.1
!pip install pyworld==0.3.1 praat-parselmouth
!pip install fairseq==0.12.2 librosa==0.8.1

#@title Download the necessary model files
# Source Repository Address：
[contentvec](https://github.com/auspicious3000/contentvec)
# Model original download link：
[checkpoint_best_legacy_500.pt](https://ibm.box.com/s/z1wgl1stco8ffooyatzdwsqn2psd9lrr)
# Since the source network disk can not provide http direct link, according to the mit protocol, the model is distributed twice to provide download direct link
!wget -P hubert/ https://huggingface.co/datasets/makiligon/required-stuff-for-things/resolve/main/checkpoint_best_legacy_500.pt

#@markdown ## Mount your Google Drive
#@markdown (This is an essential step if you want to load your own trained model)
from google.colab import drive
drive.mount('/content/drive')

#@title Import model and config file from Google Drive
#@markdown example of whats inside the zip file https://cdn.discordapp.com/attachments/749847915660443678/1080738982654132234/image.png remember to add a slash at the end of ZIP_PATH
ZIP_FILE_NAME = "arianagrande"  #@param {type:"string"}
ZIP_PATH = "/content/drive/MyDrive/models/"  #@param {type:"string"}
CONFIG_NAME = "config"  #@param {type:"string"}
ZIP_NAME = ZIP_PATH + ZIP_FILE_NAME

!unzip -d /content/so-vits-svc/logs/44k {ZIP_NAME}.zip
!mv "/content/so-vits-svc/logs/44k/{CONFIG_NAME}.json" "/content/so-vits-svc/configs"
```

```
#@markdown ## Upload your reference audio or zip file of reference audios
#@markdown alternatively you can just drag and drop from google drive to
so-vits-svc/raw if you bothered mounting to google drive. Google Drive
method is faster

%cd "/content/so-vits-svc/raw/"

from google.colab import files
uploaded = files.upload()

%cd "/content/so-vits-svc/"
print("\n\033[32m\033[1mdone")

#@title Synthesize audio (inference)
#@markdown SPK_NAME is the name of the folder you gave to the training
dataset. If you dont remember the SPK_NAME check config.json
G_MODEL_PATH = "logs/44k/G_69420.pth"  #@param {type:"string"}
CONFIG_PATH = "configs/config.json"  #@param {type:"string"}
SPK_NAME = "beberexha"  #@param {type:"string"}
RAW_AUDIO = "Name_of_audio_file.wav"  #@param {type:"string"}
TRANSPOSE = "0"  #@param {type:"string"}

!python inference_main.py -m "{G_MODEL_PATH}" -c "{CONFIG_PATH}" -n
"{RAW_AUDIO}" -t {TRANSPOSE} -s {SPK_NAME}
```

What code does:

Downloads the Hubert model file. Used for reference for own model.

Mounts Google Drive to access uploaded trained model.

Unzips uploaded compressed model folder, and locates config file.

Uploaded raw audio file (reference audio, or audio file that you want the AI model to imitate).

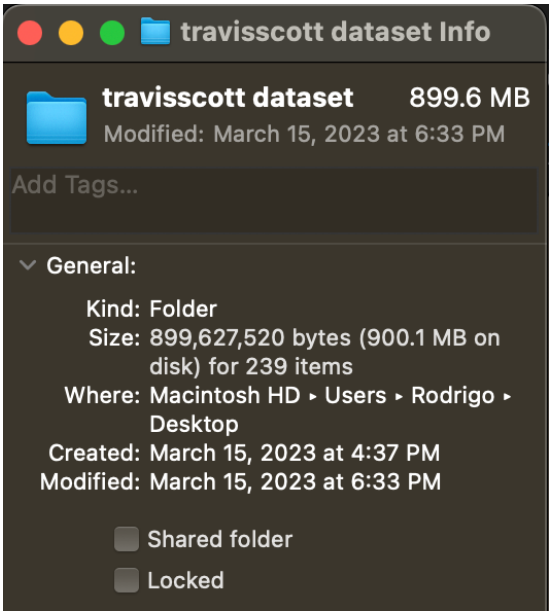
Script then locates G file path (G file is required for making inferences, D file is only needed for training), config file (so that the script knows what speaker to use), raw audio, and transpose (pitch of inference voice, typically stays at 0).

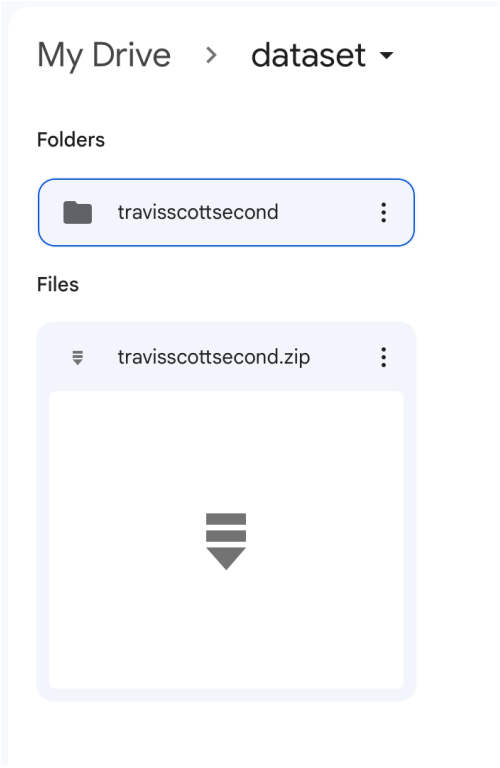
An inference of the AI model is created based on the raw audio, and it output into a “results” folder. File outputs as a .flac file.

Personal Training and Inference

I collected my own dataset, trained my own model using the explained method, and created inferences of my own reference vocals in order to recreate low quality music and remaster into high quality sound.

Dataset



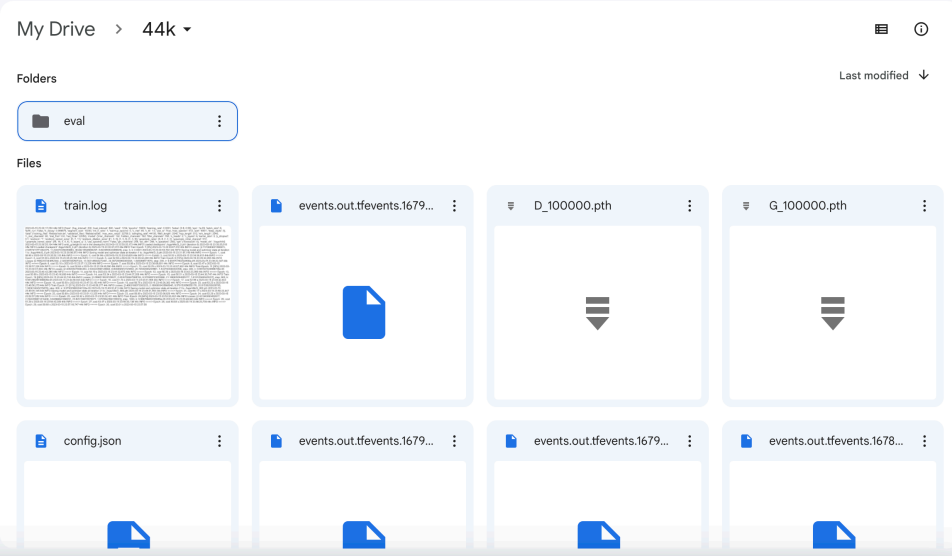


The speaker I am making a model of is Travis Scott, well known and famous rapper. I came up with a dataset of 239 .wav files, with the size of almost a gigabyte of just raw audio pulled from and cut up from various interviews. The interviewer had to be cut out from all clips, leaving only the speaker's unaltered voice. I had also included some acappella stem files from Travis Scott's studio sessions, though these only make up a very small portion of the dataset.

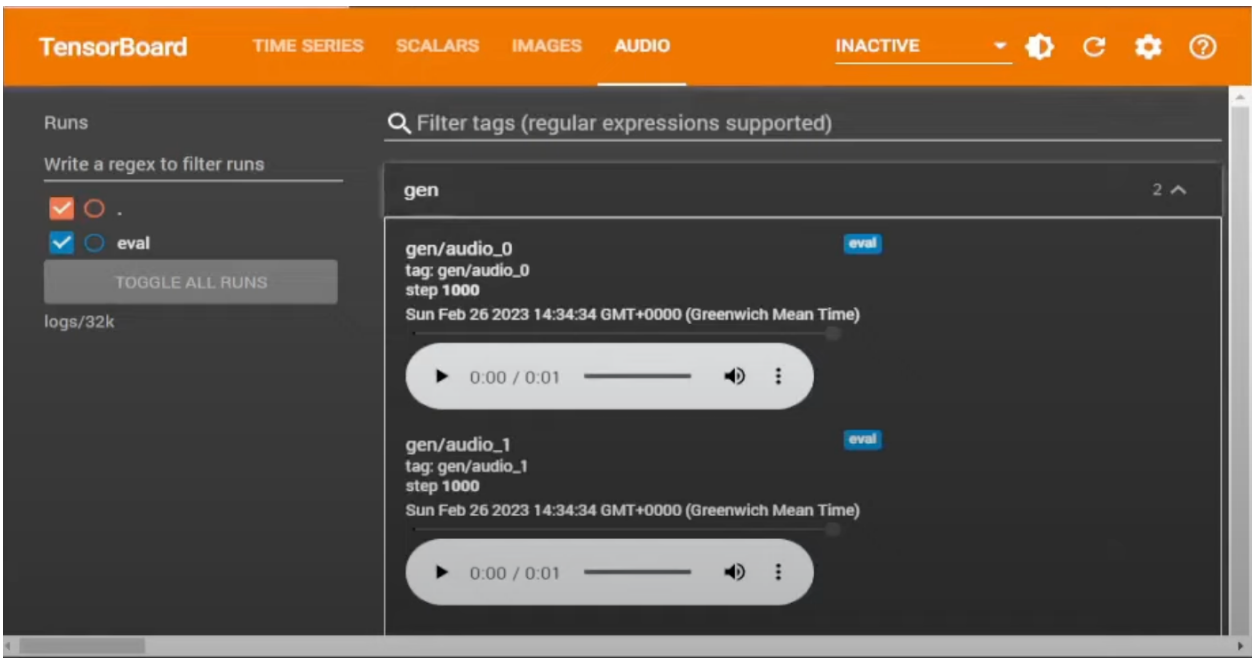
This folder of .wav files is then compressed and uploaded into my personal Google Drive, where the Google Colab script can access from mounting.

Training Process

I then used the above script to begin and go through the training process of my brand new AI model. Mounting my Google Drive, giving access to my dataset, then monitoring the new model files that are created in my Google Drive's 44k folder.



After much moderating and many days later, I managed to get my model trained up to 100,000 steps. This ended up being a tedious process because I could not just turn on the training process and leave it over night while I was away. I had to constantly monitor my Drive and delete old model files which took up about 1 GB each. The training process would stop when my Drive had run out of storage space.

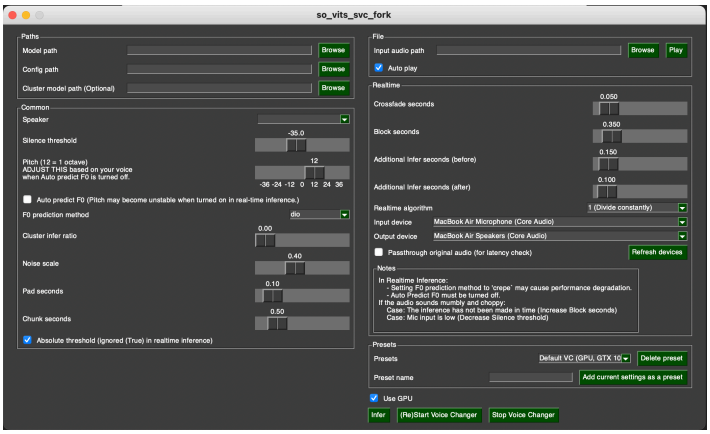


At this point, the model was sounding pretty good, and the training process was finished.

Inference Process

Making inferences would be a painful process. Going through the entire script and commands to get the process started was tedious and took a few minutes at a time. But loading up raw audio and created inferences was easy at that point.

What made making inferences easier, was downloading a graphic user interface that was compatible with my Apple Silicon CPU. All I had to do was load up my reference vocals, open my preset with my trained model loaded, set the transpose, and infer.



Recreating the Song

At this point, I had shown you the process I went through to collect my own dataset, train my own model, and inference my own reference vocals to create an AI-synthesized vocal take for a song remake.

The next part of the project will take place entirely during my presentation, as I can’t showcase audio files in a document.

References

Bansal, S. (2021). Deep Learning for Speech Synthesis: A Review. IEEE Access, 9, 56224-56243. https://doi.org/10.1109/ACCESS.2021.3072378.

Google. (2021, September 29). G Suite Learning Center: Google Docs. https://docs.google.com/document/d/1y1pfS0LCrwbbvxdn3ZksH25BKaf0LaO13uYppxIQnac/edit#heading=h.dc64xv18v6pq.

svc-develop-team. (2022, January 10). so-vits-svc. GitHub. https://github.com/svc-develop-team/so-vits-svc.

Van Den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., & Kavukcuoglu, K. (2016). WaveNet: A generative model for raw audio. arXiv preprint arXiv:1609.03499. Retrieved from https://arxiv.org/abs/1609.03499.

34j. (2022, February 15). so-vits-svc-fork [Forked from svc-develop-team/so-vits-svc]. GitHub. https://github.com/34j/so-vits-svc-fork.