Install packages and download models

```
%%shell
git clone https://github.com/yl4579/StyleTTS2.git
cd StyleTTS2
pip install SoundFile torchaudio munch torch pydub pyyaml librosa nltk matplotlib accelerate transformers phonemizer einops
sudo apt-get install espeak-ng
git-lfs clone https://huggingface.co/yl4579/StyleTTS2-LibriTTS
mv StyleTTS2-LibriTTS/Models .
```

```
Preparing to unpack .../libpcaudio0_1.1-6build2_amd64.deb ...
Unpacking libpcaudio0:amd64 (1.1-6build2) ...
Selecting previously unselected package libsonic0:amd64.
Preparing to unpack .../libsonic0_0.2.0-11build1_amd64.deb ...
Unpacking libsonic0:amd64 (0.2.0-11build1) ...
Selecting previously unselected package espeak-ng-data:amd64.
Preparing to unpack .../espeak-ng-data_1.50+dfsg-10ubuntu0.1_amd64.deb ...
Unpacking espeak-ng-data:amd64 (1.50+dfsg-10ubuntu0.1) ...
Selecting previously unselected package libespeak-ng1:amd64.
Preparing to unpack .../libespeak-ng1_1.50+dfsg-10ubuntu0.1_amd64.deb ...
Unpacking libespeak-ng1:amd64 (1.50+dfsg-10ubuntu0.1) ...
Selecting previously unselected package espeak-ng.
Preparing to unpack .../espeak-ng_1.50+dfsg-10ubuntu0.1_amd64.deb ...
Unpacking espeak-ng (1.50+dfsg-10ubuntu0.1) ...
Setting up libpcaudio0:amd64 (1.1-6build2) ...
Setting up libsonic0:amd64 (0.2.0-11build1) ...
Setting up espeak-ng-data:amd64 (1.50+dfsg-10ubuntu0.1) ...
Setting up libespeak-ng1:amd64 (1.50+dfsg-10ubuntu0.1) ...
Setting up espeak-ng (1.50+dfsg-10ubuntu0.1) ...
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for libc-bin (2.35-0ubuntu3.8) ...
/sbin/ldconfig.real: /usr/local/lib/libumf.so.0 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbb.so.12 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbbbind_2_0.so.3 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libur_loader.so.0 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtcm.so.1 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbbbind.so.3 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libur_adapter_level_zero.so.0 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbbmalloc.so.2 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libhwloc.so.15 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtcm_debug.so.1 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbbbind_2_5.so.3 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libur_adapter_opencl.so.0 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbbmalloc_proxy.so.2 is not a symbolic link

WARNING: 'git lfs clone' is deprecated and will not be updated
          with new flags from 'git clone'

'git clone' has been updated in upstream Git to have comparable
speeds to 'git lfs clone'.
Cloning into 'StyleTTS2-LibriTTS'...
remote: Enumerating objects: 25, done.
remote: Counting objects: 100% (12/12), done.
remote: Compressing objects: 100% (9/9), done.
remote: Total 25 (delta 0), reused 0 (delta 0), pack-reused 13 (from 1)
Unpacking objects: 100% (25/25), 3.73 KiB | 764.00 KiB/s, done.
```

Download dataset (LJSpeech, 200 samples, ~15 minutes of data)

You can definitely do it with fewer samples. This is just a proof of concept with 200 smaples.

```
%cd StyleTTS2
!rm -rf Data
```

```
/content/StyleTTS2
```

```
#!gdown --id 1vqz26D3yn7OXS2vbfYxfSnpLS6m6tOFP
!gdown --id 1UO_8yjJtPXWIYk6vXa1rOKhrqjpwEqZ8
!unzip Data.zip
```

```
/usr/local/lib/python3.11/dist-packages/gdown/__main__.py:140: FutureWarning: Option `--id` was deprecated in version 4.
  warnings.warn(
Downloading...
From (original): https://drive.google.com/uc?id=1UO_8yjJtPXWIYk6vXa1rOKhrqjpwEqZ8
From (redirected): https://drive.google.com/uc?id=1UO_8yjJtPXWIYk6vXa1rOKhrqjpwEqZ8&confirm=t&uuid=3a98c2c2-33d4-4ce4-a0
To: /content/StyleTTS2/Data.zip
100% 49.1M/49.1M [00:01<00:00, 33.4MB/s]
Archive:  Data.zip
   creating: Data/
  inflating: Data/LJ001-0048.wav
  inflating: Data/LJ001-0060.wav
  inflating: Data/LJ001-0074.wav
  inflating: Data/LJ001-0128.wav
  inflating: Data/LJ001-0114.wav
  inflating: Data/LJ001-0100.wav
  inflating: Data/LJ001-0101.wav
  inflating: Data/LJ001-0115.wav
  inflating: Data/LJ001-0129.wav
  inflating: Data/LJ001-0075.wav
  inflating: Data/LJ001-0061.wav
  inflating: Data/LJ001-0049.wav
  inflating: Data/LJ001-0077.wav
  inflating: Data/LJ001-0063.wav
  inflating: Data/LJ001-0088.wav
  inflating: Data/LJ001-0103.wav
  inflating: Data/LJ001-0117.wav
  inflating: Data/LJ001-0116.wav
  inflating: Data/LJ001-0102.wav
  inflating: Data/LJ001-0089.wav
  inflating: Data/LJ001-0062.wav
  inflating: Data/LJ001-0076.wav
  inflating: Data/LJ001-0072.wav
  inflating: Data/LJ001-0066.wav
  inflating: Data/LJ001-0099.wav
  inflating: Data/LJ001-0106.wav
  inflating: Data/LJ001-0112.wav
  inflating: Data/LJ001-0113.wav
  inflating: Data/LJ001-0107.wav
  inflating: Data/LJ001-0098.wav
  inflating: Data/LJ001-0067.wav
  inflating: Data/LJ001-0073.wav
  inflating: Data/LJ001-0065.wav
  inflating: Data/LJ001-0071.wav
  inflating: Data/LJ001-0059.wav
  inflating: Data/LJ001-0111.wav
  inflating: Data/LJ001-0105.wav
  inflating: Data/LJ001-0139.wav
  inflating: Data/LJ001-0138.wav
  inflating: Data/LJ001-0104.wav
  inflating: Data/LJ001-0110.wav
  inflating: Data/LJ001-0058.wav
  inflating: Data/LJ001-0070.wav
  inflating: Data/LJ001-0064.wav
  inflating: Data/LJ001-0003.wav
  inflating: Data/LJ001-0017.wav
  inflating: Data/LJ001-0177.wav
  inflating: Data/LJ001-0163.wav
  inflating: Data/LJ001-0162.wav
```

```
input_file_name = "200_sample.txt"
output_file_name = "phonemized_200.txt"
# load phonemizer
import phonemizer
global_phonemizer = phonemizer.backend.EspeakBackend(language='en-us', preserve_punctuation=True,  with_stress=True)

def text_to_phonemes(text):
    text = text.strip()
    ps = global_phonemizer.phonemize([text])
    return ps[0]

file_out = open(output_file_name, 'w')

with open(input_file_name, 'r', encoding='utf-8') as f:
    for line in f:
      #print(line)
      wave_file, text = line.split('|', 1)
      wave_file_name = wave_file + '.wav'
      #print(wave_file_name, text)
      phonemized = text_to_phonemes(text)
      #print(phonemized)
      #file_out.write(wave_file_name+ '.wav' + '|' + phonemized +'|'+'0'+'\n')
      file_out.write(f"{wave_file_name}|{phonemized}|{0}\n")
file_out.close()
```

```
WARNING:phonemizer:words count mismatch on 200.0% of the lines (2/1)
WARNING:phonemizer:words count mismatch on 200.0% of the lines (2/1)
WARNING:phonemizer:words count mismatch on 200.0% of the lines (2/1)
WARNING:phonemizer:words count mismatch on 200.0% of the lines (2/1)
WARNING:phonemizer:words count mismatch on 200.0% of the lines (2/1)
WARNING:phonemizer:words count mismatch on 200.0% of the lines (2/1)
WARNING:phonemizer:words count mismatch on 200.0% of the lines (2/1)
WARNING:phonemizer:words count mismatch on 200.0% of the lines (2/1)
WARNING:phonemizer:words count mismatch on 100.0% of the lines (1/1)
WARNING:phonemizer:words count mismatch on 200.0% of the lines (2/1)
WARNING:phonemizer:words count mismatch on 400.0% of the lines (4/1)
WARNING:phonemizer:words count mismatch on 100.0% of the lines (1/1)
WARNING:phonemizer:words count mismatch on 200.0% of the lines (2/1)
WARNING:phonemizer:words count mismatch on 200.0% of the lines (2/1)
WARNING:phonemizer:words count mismatch on 200.0% of the lines (2/1)
WARNING:phonemizer:words count mismatch on 400.0% of the lines (4/1)
WARNING:phonemizer:words count mismatch on 200.0% of the lines (2/1)
WARNING:phonemizer:words count mismatch on 200.0% of the lines (2/1)
WARNING:phonemizer:words count mismatch on 200.0% of the lines (2/1)
WARNING:phonemizer:words count mismatch on 100.0% of the lines (1/1)
WARNING:phonemizer:words count mismatch on 100.0% of the lines (1/1)
WARNING:phonemizer:words count mismatch on 200.0% of the lines (2/1)
WARNING:phonemizer:words count mismatch on 100.0% of the lines (1/1)
WARNING:phonemizer:words count mismatch on 100.0% of the lines (1/1)
WARNING:phonemizer:words count mismatch on 300.0% of the lines (3/1)
WARNING:phonemizer:words count mismatch on 100.0% of the lines (1/1)
WARNING:phonemizer:words count mismatch on 400.0% of the lines (4/1)
WARNING:phonemizer:words count mismatch on 200.0% of the lines (2/1)
WARNING:phonemizer:words count mismatch on 200.0% of the lines (2/1)
WARNING:phonemizer:words count mismatch on 200.0% of the lines (2/1)
WARNING:phonemizer:words count mismatch on 200.0% of the lines (2/1)
WARNING:phonemizer:words count mismatch on 200.0% of the lines (2/1)
WARNING:phonemizer:words count mismatch on 200.0% of the lines (2/1)
WARNING:phonemizer:words count mismatch on 300.0% of the lines (3/1)
WARNING:phonemizer:words count mismatch on 300.0% of the lines (3/1)
WARNING:phonemizer:words count mismatch on 200.0% of the lines (2/1)
WARNING:phonemizer:words count mismatch on 100.0% of the lines (1/1)
WARNING:phonemizer:words count mismatch on 200.0% of the lines (2/1)
WARNING:phonemizer:words count mismatch on 400.0% of the lines (4/1)
WARNING:phonemizer:words count mismatch on 200.0% of the lines (2/1)
WARNING:phonemizer:words count mismatch on 100.0% of the lines (1/1)
WARNING:phonemizer:words count mismatch on 200.0% of the lines (2/1)
WARNING:phonemizer:words count mismatch on 200.0% of the lines (2/1)
WARNING:phonemizer:words count mismatch on 200.0% of the lines (2/1)
WARNING:phonemizer:words count mismatch on 200.0% of the lines (2/1)
WARNING:phonemizer:words count mismatch on 200.0% of the lines (2/1)
WARNING:phonemizer:words count mismatch on 400.0% of the lines (4/1)
WARNING:phonemizer:words count mismatch on 200.0% of the lines (2/1)
WARNING:phonemizer:words count mismatch on 200.0% of the lines (2/1)
WARNING:phonemizer:words count mismatch on 100.0% of the lines (1/1)
WARNING:phonemizer:words count mismatch on 200.0% of the lines (2/1)
WARNING:phonemizer:words count mismatch on 400.0% of the lines (4/1)
WARNING:phonemizer:words count mismatch on 400.0% of the lines (4/1)
WARNING:phonemizer:words count mismatch on 100.0% of the lines (1/1)
WARNING:phonemizer:words count mismatch on 200.0% of the lines (2/1)
```

```
!head -n 150 phonemized_200.txt > Data/train_list.txt
!tail -n 50 phonemized_200.txt > Data/val_list.txt
!head -n 150 phonemized_200.txt > Data/OOD_texts.txt
```

Change the finetuning config

Depending on the GPU you got, you may want to change the bacth size, max audio length, epiochs and so on.

```
config_path = "Configs/config_ft.yml"

import yaml
config = yaml.safe_load(open(config_path))
!cat Configs/config_ft.yml
!head Data/train_list.txt
!head 200_sample.txt
```

```
{ASR_config: Utils/ASR/config.yml, ASR_path: Utils/ASR/epoch_00080.pth, F0_path: Utils/JDC/bst.t7,
  PLBERT_dir: Utils/PLBERT/, batch_size: 2, data_params: {OOD_data: Data/OOD_texts.txt,
    min_length: 50, root_path: Data/, train_data: Data/train_list.txt, val_data: Data/val_list.txt},
  device: cuda, epochs: 1, load_only_params: true, log_dir: Models/LJSpeech, log_interval: 10,
  loss_params: {diff_epoch: 10, joint_epoch: 110, lambda_F0: 1.0, lambda_ce: 20.0,
    lambda_diff: 1.0, lambda_dur: 1.0, lambda_gen: 1.0, lambda_mel: 5.0, lambda_mono: 1.0,
    lambda_norm: 1.0, lambda_s2s: 1.0, lambda_slm: 1.0, lambda_sty: 1.0}, max_len: 100,
  model_params: {decoder: {resblock_dilation_sizes: [[1, 3, 5], [1, 3, 5], [1, 3,
          5]], resblock_kernel_sizes: [3, 7, 11], type: hifigan, upsample_initial_channel: 512,
      upsample_kernel_sizes: [20, 10, 6, 4], upsample_rates: [10, 5, 3, 2]}, diffusion: {
      dist: {estimate_sigma_data: true, mean: -3.0, sigma_data: 0.2, std: 1.0}, embedding_mask_proba: 0.1,
      transformer: {head_features: 64, multiplier: 2, num_heads: 8, num_layers: 3}},
    dim_in: 64, dropout: 0.2, hidden_dim: 512, max_conv_dim: 512, max_dur: 50, multispeaker: true,
    n_layer: 3, n_mels: 80, n_token: 178, slm: {hidden: 768, initial_channel: 64,
```

```
        model: microsoft/wavlm-base-plus, nlayers: 13, sr: 16000}, style_dim: 128},
    optimizer_params: {bert_lr: 1.0e-05, ft_lr: 0.0001, lr: 0.0001}, preprocess_params: {
      spect_params: {hop_length: 300, n_fft: 2048, win_length: 1200}, sr: 24000}, pretrained_model: Models/LibriTTS/epochs
    save_freq: 1, second_stage_load_pretrained: true, slmadv_params: {batch_percentage: 0.5,
      iter: 10, max_len: 500, min_len: 400, scale: 0.01, sig: 1.5, thresh: 5}}
LJ001-0001.wav|pɹˈɪntɪŋ, ɪnðɪ ˈoʊnli sˈɛns wɪð wˌɪtʃ wiː ɑːɹ æt pɹˈɛzənt kənsˈɜːnd, dˈɪfɚz fɹʌm mˈoʊst ɪf nˌɑːt fɹʌm ˈɔː
LJ001-0002.wav|ɪn bˌiːɪŋ kəmpˈæɹətˌɪvli mˈɑːdɚn.ɪn bˌiːɪŋ kəmpˈæɹətˌɪvli mˈɑːdɚn. |0
LJ001-0003.wav|fɔːɹ ɔːlðˈoʊ ðə tʃaɪnˈiːz tˈʊk ɪmpɹˈɛʃənz fɹʌm wˈʊd blˈɑːks ɛŋgɹˈeɪvd ɪn ɹɨlˈiːf fɔːɹ sˈɛntʃɚɹiz bɨfˌoːɹ
LJ001-0004.wav|pɹədˈuːst ðə blˈɑːk bˈʊks, wˌɪtʃ wɜː ðɪ ɪmˈiːdɪət pɹˈɛdɨsˌɛsɚz ʌvðə tɹˈuː pɹˈɪntɨd bˈʊk,pɹədˈuːst ðə blˈɑ
LJ001-0005.wav|ðɪ ɪnvˈɛnʃən ʌv mˈuːvəbəl mˈɛɾəl lˈɛɾɚz ɪnðə mˈɪdəl ʌvðə fˈɪfti:nθ sˈɛntʃɚɹi mˈeɪ dʒˈʌstli biː kənsˈɪdɚd
LJ001-0006.wav|ænd ɪɾ ɪz wˈɜːθ mˈɛnʃən ɪn pˈæsɪŋ ðˈæt, æz ɐn ɛgzˈæmpəl ʌv fˈaɪn taɪpˈɑːgɹəfi,ænd ɪɾ ɪz wˈɜːθ mˈɛnʃən ɪn
LJ001-0007.wav|ðɪ ˈɜːlɪɪst bˈʊk pɹˈɪntɨd wɪð mˈuːvəbəl tˈaɪps, ðə gjˈuːtənbˌɜːg, ɔːɹ "fˈɔːɹɾit'uː lˈaɪn bˈaɪbəl" ʌv ɛbˌa
LJ001-0008.wav|hɛz nˈɛvɚ bˌɪn sɚpˈæst.hɛz nˈɛvɚ bˌɪn sɚpˈæst. |0
LJ001-0009.wav|pɹˈɪntɪŋ, ðˈɛn, fɔːɹ ˌaʊɚ pˈɜːpəs, mˈeɪ biː kənsˈɪdɚd æz ðɪ ˈɑːɹt ʌv mˌeɪkɪŋ bˈʊks baɪ mˈiːnz ʌv mˈuːvəbə
LJ001-0010.wav|nˈaʊ, æz ˈɔːl bˈʊks nˌɑːt pɹaɪmˈɛɹɨli ɪntˈɛndɨd æz pˈɪktʃɚbˈʊks kənsˈɪst pɹˈɪnsɪpəli ʌv tˈaɪps kəmpˈoʊzd
LJ001-0001|Printing, in the only sense with which we are at present concerned, differs from most if not from all the art
LJ001-0002|in being comparatively modern.|in being comparatively modern.
LJ001-0003|For although the Chinese took impressions from wood blocks engraved in relief for centuries before the woodcu
LJ001-0004|produced the block books, which were the immediate predecessors of the true printed book,|produced the block
LJ001-0005|the invention of movable metal letters in the middle of the fifteenth century may justly be considered as the
LJ001-0006|And it is worth mention in passing that, as an example of fine typography,|And it is worth mention in passing
LJ001-0007|the earliest book printed with movable types, the Gutenberg, or "forty-two line Bible" of about 1455,|the ear
LJ001-0008|has never been surpassed.|has never been surpassed.
LJ001-0009|Printing, then, for our purpose, may be considered as the art of making books by means of movable types.|Prin
LJ001-0010|Now, as all books not primarily intended as picture-books consist principally of types composed to form lette
```

```
config['data_params']['root_path'] = "Data/"

config['batch_size'] = 2 # not enough RAM
config['save_freq'] = 1
config['max_len'] = 100 # not enough RAM
config['epochs'] = 50
config['loss_params']['diff_epoch'] = 10
config['loss_params']['joint_epoch'] = 110 # we do not do SLM adversarial training due to not enough RAM

with open(config_path, 'w') as outfile:
  yaml.dump(config, outfile, default_flow_style=True)
```

Start finetuning

```
!python train_finetune.py --config_path ./Configs/config_ft.yml
```

```
ɪn 'ɪŋɡlənd ɐb,aʊt ðɪs t'aɪm, ɐn ɐt'ɛmpt wʌz m'eɪd (n'oʊɾəbli baɪ k'æslɑːn, h,uː st'ɑːɹɾᵻd b'ɪznəs ɪn l'ʌndən æz ɐ t'aɪp
ɪn 'ɪŋɡlənd ɐb,aʊt ðɪs t'aɪm, ɐn ɐt'ɛmpt wʌz m'eɪd (n'oʊɾəbli baɪ k'æslɑːn, h,uː st'ɑːɹɾᵻd b'ɪznəs ɪn l'ʌndən æz ɐ t'aɪp
ɪn 'ɪŋɡlənd ɐb,aʊt ðɪs t'aɪm, ɐn ɐt'ɛmpt wʌz m'eɪd (n'oʊɾəbli baɪ k'æslɑːn, h,uː st'ɑːɹɾᵻd b'ɪznəs ɪn l'ʌndən æz ɐ t'aɪp
ɪn 'ɪŋɡlənd ɐb,aʊt ðɪs t'aɪm, ɐn ɐt'ɛmpt wʌz m'eɪd (n'oʊɾəbli baɪ k'æslɑːn, h,uː st'ɑːɹɾᵻd b'ɪznəs ɪn l'ʌndən æz ɐ t'aɪp
Epochs: 1
Validation loss: 0.728, Dur loss: 0.224, F0 loss: 5.695







Saving..
```

Test the model quality

Note that this mainly serves as a proof of concept due to RAM limitation of free Colab instances. A lot of settings are suboptimal. In the future when DDP works for train_second.py, we will also add mixed precision finetuning to save time and RAM. You can also add SLM adversarial training run if you have paid Colab services (such as A100 with 40G of RAM).

```python
import nltk
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
True
```

```python
import torch
torch.manual_seed(0)
torch.backends.cudnn.benchmark = False
torch.backends.cudnn.deterministic = True

import random
random.seed(0)

import numpy as np
np.random.seed(0)

# load packages
import time
import random
import yaml
from munch import Munch
import numpy as np
import torch
from torch import nn
import torch.nn.functional as F
import torchaudio
import librosa
from nltk.tokenize import word_tokenize

from models import *
from utils import *
from text_utils import TextCleaner
textclenaer = TextCleaner()

%matplotlib inline

to_mel = torchaudio.transforms.MelSpectrogram(
    n_mels=80, n_fft=2048, win_length=1200, hop_length=300)
mean, std = -4, 4

def length_to_mask(lengths):
    mask = torch.arange(lengths.max()).unsqueeze(0).expand(lengths.shape[0], -1).type_as(lengths)
    mask = torch.gt(mask+1, lengths.unsqueeze(1))
    return mask

def preprocess(wave):
    wave_tensor = torch.from_numpy(wave).float()
    mel_tensor = to_mel(wave_tensor)
    mel_tensor = (torch.log(1e-5 + mel_tensor.unsqueeze(0)) - mean) / std
    return mel_tensor

def compute_style(path):
    wave, sr = librosa.load(path, sr=24000)
    audio, index = librosa.effects.trim(wave, top_db=30)
    if sr != 24000:
        audio = librosa.resample(audio, sr, 24000)
    mel_tensor = preprocess(audio).to(device)
```

```
    with torch.no_grad():
        ref_s = model.style_encoder(mel_tensor.unsqueeze(1))
        ref_p = model.predictor_encoder(mel_tensor.unsqueeze(1))

    return torch.cat([ref_s, ref_p], dim=1)

device = 'cuda' if torch.cuda.is_available() else 'cpu'

# load phonemizer
import phonemizer
global_phonemizer = phonemizer.backend.EspeakBackend(language='en-us', preserve_punctuation=True,  with_stress=True)

config = yaml.safe_load(open("Models/LJSpeech/config_ft.yml"))

# load pretrained ASR model
ASR_config = config.get('ASR_config', False)
ASR_path = config.get('ASR_path', False)
text_aligner = load_ASR_models(ASR_path, ASR_config)

# load pretrained F0 model
F0_path = config.get('F0_path', False)
pitch_extractor = load_F0_models(F0_path)

# load BERT model
from Utils.PLBERT.util import load_plbert
BERT_path = config.get('PLBERT_dir', False)
plbert = load_plbert(BERT_path)

model_params = recursive_munch(config['model_params'])
model = build_model(model_params, text_aligner, pitch_extractor, plbert)
_ = [model[key].eval() for key in model]
_ = [model[key].to(device) for key in model]
```

```
177
```

```
files = [f for f in os.listdir("Models/LJSpeech/") if f.endswith('.pth')]
sorted_files = sorted(files, key=lambda x: int(x.split('_')[-1].split('.')[0]))


params_whole = torch.load("Models/LJSpeech/" + sorted_files[-1], map_location='cpu')
params = params_whole['net']
```

```
<ipython-input-40-febab44deea8>:1: FutureWarning: You are using `torch.load` with `weights_only=False` (the current defa
  params_whole = torch.load("Models/LJSpeech/" + sorted_files[-1], map_location='cpu')
```

```
for key in model:
    if key in params:
        print('%s loaded' % key)
        try:
            model[key].load_state_dict(params[key])
        except:
            from collections import OrderedDict
            state_dict = params[key]
            new_state_dict = OrderedDict()
            for k, v in state_dict.items():
                name = k[7:] # remove `module.`
                new_state_dict[name] = v
            # load params
            model[key].load_state_dict(new_state_dict, strict=False)
#             except:
#                 _load(params[key], model[key])
_ = [model[key].eval() for key in model]
```

```
bert loaded
bert_encoder loaded
predictor loaded
decoder loaded
text_encoder loaded
predictor_encoder loaded
style_encoder loaded
diffusion loaded
text_aligner loaded
pitch_extractor loaded
mpd loaded
msd loaded
wd loaded
```

```
from Modules.diffusion.sampler import DiffusionSampler, ADPM2Sampler, KarrasSchedule


sampler = DiffusionSampler(
    model.diffusion.diffusion,
```

```
        sampler=ADPM2Sampler(),
        sigma_schedule=KarrasSchedule(sigma_min=0.0001, sigma_max=3.0, rho=9.0), # empirical parameters
        clamp=False
    )


    def inference(text, ref_s, alpha = 0.3, beta = 0.7, diffusion_steps=5, embedding_scale=1):
        text = text.strip()
        ps = global_phonemizer.phonemize([text])
        #ps = word_tokenize(ps[0])
        ps = ' '.join(ps)
        print(ps)
        tokens = textclenaer(ps)
        tokens.insert(0, 0)
        tokens = torch.LongTensor(tokens).to(device).unsqueeze(0)

        with torch.no_grad():
            input_lengths = torch.LongTensor([tokens.shape[-1]]).to(device)
            text_mask = length_to_mask(input_lengths).to(device)

            t_en = model.text_encoder(tokens, input_lengths, text_mask)
            bert_dur = model.bert(tokens, attention_mask=(~text_mask).int())
            d_en = model.bert_encoder(bert_dur).transpose(-1, -2)

            s_pred = sampler(noise = torch.randn((1, 256)).unsqueeze(1).to(device),
                                             embedding=bert_dur,
                                             embedding_scale=embedding_scale,
                                               features=ref_s, # reference from the same speaker as the embedding
                                                num_steps=diffusion_steps).squeeze(1)


            s = s_pred[:, 128:]
            ref = s_pred[:, :128]

            ref = alpha * ref + (1 - alpha)  * ref_s[:, :128]
            s = beta * s + (1 - beta)  * ref_s[:, 128:]

            d = model.predictor.text_encoder(d_en,
                                            s, input_lengths, text_mask)

            x, _ = model.predictor.lstm(d)
            duration = model.predictor.duration_proj(x)

            duration = torch.sigmoid(duration).sum(axis=-1)
            # if torch.isnan(duration).any():
            #    raise ValueError("NaN detected in duration!")

            #pred_dur = torch.round(duration.squeeze()).clamp(min=1)
            pred_dur = torch.nan_to_num(torch.round(duration.squeeze()), nan=1.0).clamp(min=1)


            pred_aln_trg = torch.zeros(input_lengths, int(pred_dur.sum().data))
            c_frame = 0
            for i in range(pred_aln_trg.size(0)):
                pred_aln_trg[i, c_frame:c_frame + int(pred_dur[i].data)] = 1
                c_frame += int(pred_dur[i].data)

            # encode prosody
            en = (d.transpose(-1, -2) @ pred_aln_trg.unsqueeze(0).to(device))
            if model_params.decoder.type == "hifigan":
                asr_new = torch.zeros_like(en)
                asr_new[:, :, 0] = en[:, :, 0]
                asr_new[:, :, 1:] = en[:, :, 0:-1]
                en = asr_new

            F0_pred, N_pred = model.predictor.F0Ntrain(en, s)

            asr = (t_en @ pred_aln_trg.unsqueeze(0).to(device))
            if model_params.decoder.type == "hifigan":
                asr_new = torch.zeros_like(asr)
                asr_new[:, :, 0] = asr[:, :, 0]
                asr_new[:, :, 1:] = asr[:, :, 0:-1]
                asr = asr_new

            out = model.decoder(asr,
                                F0_pred, N_pred, ref.squeeze().unsqueeze(0))


        return out.squeeze().cpu().numpy()[..., :-50] # weird pulse at the end of the model, need to be fixed later
```

Synthesize speech

```
text = '''Maltby and Company would issue warrants on them deliverable to the importer, and the goods were then passed to be
'''
text = "text to speech synthesis from Machine."
# text = "Speech synthesis is the artificial production of human speech."


# get a random reference in the training set, note that it doesn't matter which one you use
path = "Data/LJ001-0101.wav"
# this style vector ref_s can be saved as a parameter together with the model weights
ref_s = compute_style(path)


start = time.time()
print(ref_s)
print(text)
wav = inference(text, ref_s, alpha=0, beta=0, diffusion_steps=10, embedding_scale=1)
rtf = (time.time() - start) / (len(wav) / 24000)
print(f"RTF = {rtf:5f}")
import IPython.display as ipd
display(ipd.Audio(wav, rate=24000, normalize=False))

display(ipd.Audio(path, rate=24000, normalize=False))
```

```
tensor([[ 0.0230, -0.1614, -0.0089,  0.0936, -0.0667, -0.0143,  0.0162, -0.0366,
          0.2725,  0.2923, -0.0045,  0.0017, -0.1045, -0.0276,  0.0275,  0.0012,
          0.0211, -0.1088,  0.0611,  0.0539,  0.1216,  0.0707, -0.0469, -0.0400,
          0.0426, -0.1814,  0.0665, -0.2311,  0.0576,  0.0044, -0.2770,  0.1583,
          0.0492, -0.1014, -0.0786,  0.1362,  0.0787, -0.1787, -0.1223, -0.0484,
         -0.0787,  0.0236, -0.1216, -0.1894, -0.1876, -0.2552,  0.2542,  0.0839,
          0.1114,  0.0302,  0.0627,  0.1352,  0.0344,  0.0179,  0.0844, -0.0985,
         -0.0265, -0.2179,  0.0357,  0.0587,  0.4057, -0.0599, -0.1162, -0.1824,
         -0.1900,  0.1794, -0.1033,  0.3379, -0.0047, -0.1541, -0.0626, -0.0746,
          0.1323, -0.0219, -0.0335, -0.0861, -0.0786,  0.0584, -0.1455,  0.0352,
         -0.2429,  0.0640, -0.0404,  0.0126,  0.2362,  0.0240,  0.1099, -0.0698,
          0.1150,  0.2502, -0.3255, -0.0184, -0.3179, -0.1842,  0.1843, -0.0238,
          0.0191, -0.2090, -0.3168, -0.0383,  0.0017, -0.0431,  0.1877,  0.0248,
          0.0517,  0.1304,  0.3227,  0.1507, -0.1555, -0.1549,  0.2278, -0.1100,
          0.3189, -0.1968, -0.0099,  0.0088,  0.0436, -0.0377,  0.0609,  0.0788,
         -0.0845, -0.1078,  0.0197,  0.1407,  0.2768, -0.0979, -0.0925,  0.0059,
          0.1885, -0.1549,  0.2506, -0.0048, -0.0137, -0.2540, -0.2044,  0.0987,
          0.5234,  0.2017, -0.1850,  0.3116, -0.0817,  0.2002,  0.1932,  0.0824,
         -0.0045, -0.0127,  0.4878,  0.1704, -0.0252, -0.0830,  0.0587,  0.6587,
         -0.0385, -0.1336, -0.1389, -0.4228, -0.0039,  0.3074, -0.2779,  0.0376,
         -0.2588, -0.3194, -0.5228,  0.2407, -0.1212,  0.0755,  0.0649,  0.1347,
         -0.4262,  0.1032, -0.2839, -0.1088, -0.0602, -0.1684, -0.0619,  0.0932,
         -0.1153, -0.1015,  0.3091, -0.0445,  0.3807,  0.1317, -0.1842,  0.1968,
          0.8657, -0.6194, -0.0036, -0.2980,  0.0095, -0.1377, -0.1548, -0.2327,
          0.0545, -0.1558, -0.1396,  0.0767, -0.0949, -0.0355, -0.0154,  0.1067,
          0.1954,  0.2138,  0.0370, -0.2069, -0.2167,  0.5180, -0.0355,  0.1505,
          0.3411, -0.2258, -0.1712,  0.2349,  0.1879,  0.2549,  0.2606, -0.0897,
         -0.1445,  0.3909, -0.1514, -0.3054,  0.1242, -0.1270, -0.1929, -0.2005,
         -0.0504, -0.0955, -0.3228, -0.0363, -0.0089,  0.0115,  0.1339, -0.0033,
         -0.1462, -0.0679, -0.0148, -0.0776,  0.0200, -0.3018,  0.1916,  0.2651,
          0.2457, -0.1651,  0.2363, -0.0373,  0.0821, -0.2314,  0.2360,  0.6982,
         -0.0785,  0.1640,  0.0838,  0.1179,  0.4384,  0.1361, -0.3215, -0.2652]],
        device='cuda:0')
text to speech synthesis from Machine.
tˈɛkst tə spˈiːtʃ sˈɪnθəsˌɪs fɹʌm məʃˈiːn.
RTF = 0.245347
/usr/local/lib/python3.11/dist-packages/IPython/lib/display.py:175: RuntimeWarning: invalid value encountered in cast
  return scaled.astype("<h").tobytes(), nchan
```

0:01 / 0:01

0:00 / 0:07