# 华为实验——wavenet

## 电脑环境

操作系统：Windows 10

CPU处理器规格：11th Gen Intel(R) Core(TM) i5-11400H @ 2.70GHz 2.69 GHz

机带RAM：16.0 GB (15.8 GB 可用)

GPU规格：NVIDIA GeForce RTX 3050 Ti Laptop GPU

显示内存VRAM：3990MB

共享内存：8073MB

Python 3.9

CUDA 11.7

## 配置环境

所有安装的包如下所示：

# Name Version Build Channel

asttokens 2.0.5 pyhd3eb1b0\_0

astunparse 1.6.3 py\_0

audioread 3.0.1 pypi\_0 pypi

blas 1.0 mkl

brotli 1.0.9 h2bbff1b\_7

brotli-bin 1.0.9 h2bbff1b\_7

brotli-python 1.0.9 py39hd77b12b\_7

ca-certificates 2023.12.12 haa95532\_0

certifi 2023.11.17 py39haa95532\_0

cffi 1.16.0 py39h2bbff1b\_0

charset-normalizer 3.3.2 pypi\_0 pypi

colorama 0.4.6 py39haa95532\_0

contourpy 1.2.0 py39h59b6b97\_0

cryptography 41.0.7 py39h89fc84f\_0

cuda-cccl 12.3.101 0 nvidia

cuda-cudart 11.7.99 0 nvidia

cuda-cudart-dev 11.7.99 0 nvidia

cuda-cupti 11.7.101 0 nvidia

cuda-libraries 11.7.1 0 nvidia

cuda-libraries-dev 11.7.1 0 nvidia

cuda-nvrtc 11.7.99 0 nvidia

cuda-nvrtc-dev 11.7.99 0 nvidia

cuda-nvtx 11.7.91 0 nvidia

cuda-runtime 11.7.1 0 nvidia

cycler 0.11.0 pyhd3eb1b0\_0

cython 3.0.8 pypi\_0 pypi

decorator 5.1.1 pypi\_0 pypi

docopt 0.6.2 py\_1 conda-forge

fastdtw 0.3.4 pypi\_0 pypi

filelock 3.13.1 py39haa95532\_0

fonttools 4.25.0 pyhd3eb1b0\_0

freetype 2.12.1 ha860e81\_0

giflib 5.2.1 h8cc25b3\_3

gmpy2 2.1.2 py39h7f96b67\_0

icc\_rt 2022.1.0 h6049295\_2

icu 73.1 h6c2663c\_0

idna 3.6 pypi\_0 pypi

importlib\_resources 6.1.1 py39haa95532\_1

intel-openmp 2023.1.0 h59b6b97\_46320

jinja2 3.1.2 py39haa95532\_0

joblib 1.3.2 pypi\_0 pypi

jpeg 9e h2bbff1b\_1

kiwisolver 1.4.4 py39hd77b12b\_0

krb5 1.20.1 h5b6d351\_0

lerc 3.0 hd77b12b\_0

libbrotlicommon 1.0.9 h2bbff1b\_7

libbrotlidec 1.0.9 h2bbff1b\_7

libbrotlienc 1.0.9 h2bbff1b\_7

libclang 14.0.6 default\_hb5a9fac\_1

libclang13 14.0.6 default\_h8e68704\_1

libcublas 11.10.3.66 0 nvidia

libcublas-dev 11.10.3.66 0 nvidia

libcufft 10.7.2.124 0 nvidia

libcufft-dev 10.7.2.124 0 nvidia

libcurand 10.3.4.107 0 nvidia

libcurand-dev 10.3.4.107 0 nvidia

libcusolver 11.4.0.1 0 nvidia

libcusolver-dev 11.4.0.1 0 nvidia

libcusparse 11.7.4.91 0 nvidia

libcusparse-dev 11.7.4.91 0 nvidia

libdeflate 1.17 h2bbff1b\_1

libjpeg-turbo 2.0.0 h196d8e1\_0

libnpp 11.7.4.75 0 nvidia

libnpp-dev 11.7.4.75 0 nvidia

libnvjpeg 11.8.0.2 0 nvidia

libnvjpeg-dev 11.8.0.2 0 nvidia

libpng 1.6.39 h8cc25b3\_0

libpq 12.15 h906ac69\_1

libprotobuf 3.20.3 h23ce68f\_0

librosa 0.9.1 pypi\_0 pypi

libtiff 4.5.1 hd77b12b\_0

libuv 1.44.2 h2bbff1b\_0

libwebp 1.3.2 hbc33d0d\_0

libwebp-base 1.3.2 h2bbff1b\_0

llvmlite 0.41.1 pypi\_0 pypi

lz4-c 1.9.4 h2bbff1b\_0

markupsafe 2.1.3 py39h2bbff1b\_0

matplotlib 3.8.0 py39haa95532\_0

matplotlib-base 3.8.0 py39h4ed8f06\_0

mindspore-cpu 2.2.10 py39\_0 mindspore

mkl 2023.1.0 h6b88ed4\_46358

mkl-service 2.4.0 py39h2bbff1b\_1

mkl\_fft 1.3.8 py39h2bbff1b\_0

mkl\_random 1.2.4 py39h59b6b97\_0

mpc 1.1.0 h7edee0f\_1

mpfr 4.0.2 h62dcd97\_1

mpir 3.0.0 hec2e145\_1

mpmath 1.3.0 py39haa95532\_0

munkres 1.1.4 py\_0

networkx 3.1 py39haa95532\_0

nnmnkwii 0.1.2 pypi\_0 pypi

numba 0.58.1 pypi\_0 pypi

numpy 1.26.3 py39h055cbcc\_0

numpy-base 1.26.3 py39h65a83cf\_0

openjpeg 2.4.0 h4fc8c34\_0

openssl 3.0.12 h2bbff1b\_0

packaging 23.1 py39haa95532\_0

pillow 10.0.1 py39h045eedc\_0

pip 23.3.1 py39haa95532\_0

platformdirs 4.1.0 pypi\_0 pypi

ply 3.11 py39haa95532\_0

pooch 1.8.0 pypi\_0 pypi

protobuf 3.20.3 py39hd77b12b\_0

psutil 5.9.0 py39h2bbff1b\_0

pycparser 2.21 pyhd3eb1b0\_0

pyopenssl 23.2.0 py39haa95532\_0

pyparsing 3.0.9 py39haa95532\_0

pyqt 5.15.10 py39hd77b12b\_0

pyqt5-sip 12.13.0 py39h2bbff1b\_0

pysocks 1.7.1 py39haa95532\_0

pysptk 0.2.2 pypi\_0 pypi

python 3.9.18 h1aa4202\_0

python-dateutil 2.8.2 pyhd3eb1b0\_0

python\_abi 3.9 2\_cp39 conda-forge

pytorch 2.1.2 py3.9\_cpu\_0 pytorch

pytorch-cuda 11.7 h16d0643\_5 pytorch

pytorch-mutex 1.0 cpu pytorch

pyyaml 6.0.1 py39h2bbff1b\_0

qt-main 5.15.2 h19c9488\_10

requests 2.31.0 py39haa95532\_0

resampy 0.4.2 pypi\_0 pypi

scikit-learn 1.3.2 pypi\_0 pypi

scipy 1.11.4 py39h309d312\_0

setuptools 68.2.2 py39haa95532\_0

sip 6.7.12 py39hd77b12b\_0

six 1.16.0 pyhd3eb1b0\_1

soundfile 0.12.1 pypi\_0 pypi

sqlite 3.41.2 h2bbff1b\_0

sympy 1.12 py39haa95532\_0

tbb 2021.8.0 h59b6b97\_0

tensorboardx 2.2 pyhd3eb1b0\_0

threadpoolctl 3.2.0 pypi\_0 pypi

tk 8.6.12 h2bbff1b\_0

tomli 2.0.1 py39haa95532\_0

torchaudio 2.1.2 py39\_cpu pytorch

torchvision 0.16.2 py39\_cpu pytorch

tornado 6.3.3 py39h2bbff1b\_0

tqdm 4.66.1 pyhd8ed1ab\_0 conda-forge

typing\_extensions 4.9.0 py39haa95532\_0

tzdata 2023d h04d1e81\_0

urllib3 2.1.0 pypi\_0 pypi

vc 14.2 h21ff451\_1

vs2015\_runtime 14.27.29016 h5e58377\_2

wheel 0.41.2 py39haa95532\_0

win\_inet\_pton 1.1.0 py39haa95532\_0

xz 5.4.5 h8cc25b3\_0

yaml 0.2.5 he774522\_0

zipp 3.17.0 py39haa95532\_0

zlib 1.2.13 h8cc25b3\_0

zstd 1.5.5 hd43e919\_0

## 文件结构

从<https://github.com/r9y9/wavenet_vocoder>处下载文件，并按照如下所示的方法设置文件的相对路径，另外注意在/datasets目录复制audio.py、train\_pytorch.py、wavenet\_vocoder文件夹，另外我将本次华为实验的数据集也放在/datasets目录下，而/src目录下也应该复制wavenet\_vocoder文件夹。

.

├── audio

└──wavenet

├── scripts

│ ├──run\_distribute\_train\_ascend.sh // launch distributed training with Ascend platform

│ ├──run\_distribute\_train\_gpu.sh // launch distributed training with GPU platform

│ ├──run\_eval\_ascend.sh // launch evaluation with Ascend platform

│ ├──run\_eval\_gpu.sh // launch evaluation with GPU platform

│ ├──run\_eval\_cpu.sh // launch evaluation with CPU platform

│ ├──run\_standalone\_train\_ascend.sh // launch standalone training with Ascend platform

│ ├──run\_standalone\_train\_gpu.sh // launch standalone training with GPU platform

│ └──run\_standalone\_train\_cpu.sh // launch standalone training with CPU platform

├── datasets // Process audio files for generating train/evaluate data

├── egs // Note the egs folder should be downloaded from the above link

├── utils // Note the utils folder should be downloaded from the above link

├── audio.py // Audio utils. Note this script should be downloaded from the above link

├── compute-meanvar-stats.py // Compute mean-variance Normalization stats. Note this script should be downloaded from the above link

├── evaluate.py // Evaluation

├── export.py // Convert mindspore model to air/mindir model

├── hparams.py // Hyper-parameter configuration. Note this script should be downloaded from the above link

├── mksubset.py // Make subset of dataset. Note this script should be downloaded from the above link

├── preprocess.py // Preprocess dataset. Note this script should be downloaded from the above link

├── preprocess\_normalize.py // Perform meanvar Normalization to preprocessed features. Note this script should be downloaded from the above link

├── README.md // Descriptions about WaveNet

├── train.py // Training scripts

├── train\_pytorch.py // Note this script should be downloaded from the above link. The initial name of this script is train.py in the project from the link

├── src

│ ├──\_\_init\_\_.py

│ ├──dataset.py // Generate dataloader and data processing entry

│ ├──callback.py // Callbacks to monitor the training

│ ├──lr\_generator.py // Learning rate generator

│ └──loss.py // Loss function definition

└── wavenet\_vocoder

├──\_\_init\_\_.py

├──conv.py // Extended 1D convolution

├──mixture.py // Loss function for training and sample function for testing

├──modules.py // Modules for Wavenet construction

├──upsample.py // Upsample layer definition

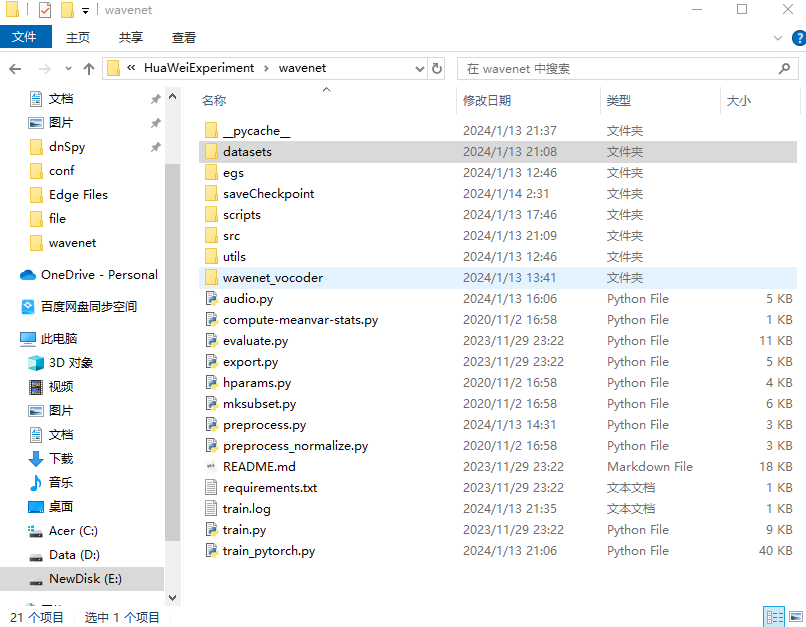
├──util.py // Utils. Note this script should be downloaded from the above link

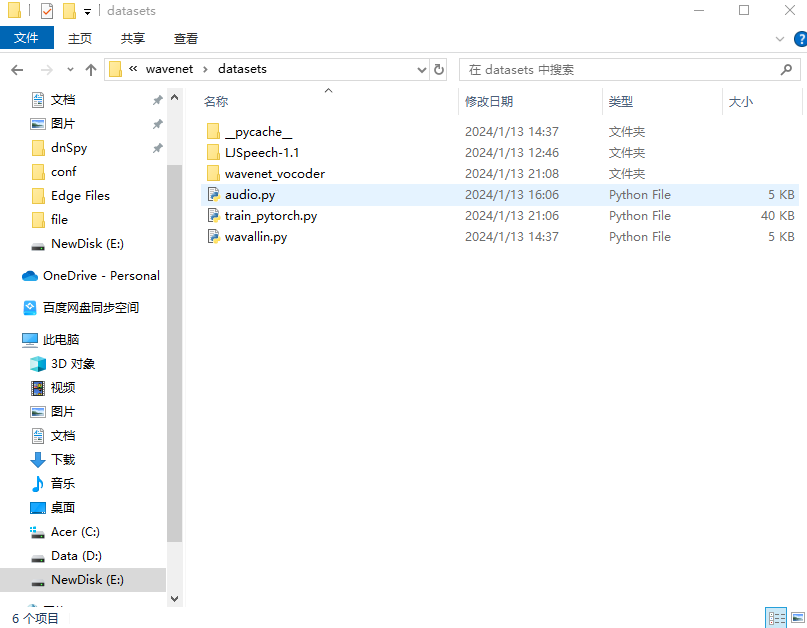
├──wavenet.py // WaveNet networks

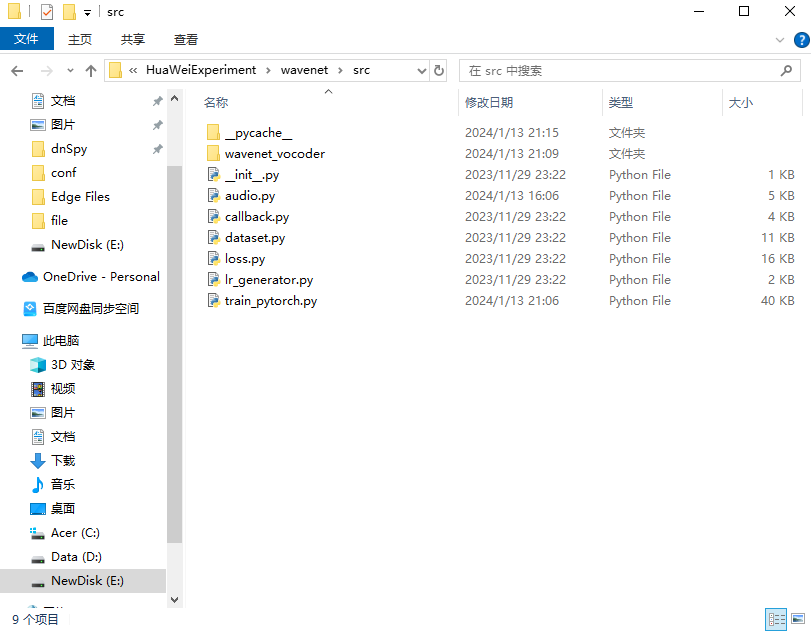
└──tfcompat // Note this script should be downloaded from the above link

├──\_\_init\_\_.py

└──hparam.py // Param management tools







如上的三张图片即为我所配置的文件结构。

## 数据预处理

参照模型说明，这里由于我是在Windows系统上，所以我在wavenet\egs\gaussian目录中使用Git bash来输入了如下命令，注意，这里需要将数据集提前解压至datasets，另外在输入第二条命令时，需要librosa库的版本<10.0，否则会产生报错。

bash run.sh --stage 0 --stop-stage 0 --db-root ../../datasets/LJSpeech-1.1/wavs

bash run.sh --stage 1 --stop-stage 1

预处理过程如下所示：

预处理后目录如下所示：

.

├── gaussian

├──conf

├──data

├──exp

└──dump

└──lj

└──logmelspectrogram

├──org

└──norm

├──train\_no\_dev

├──dev

└──eval

## 训练

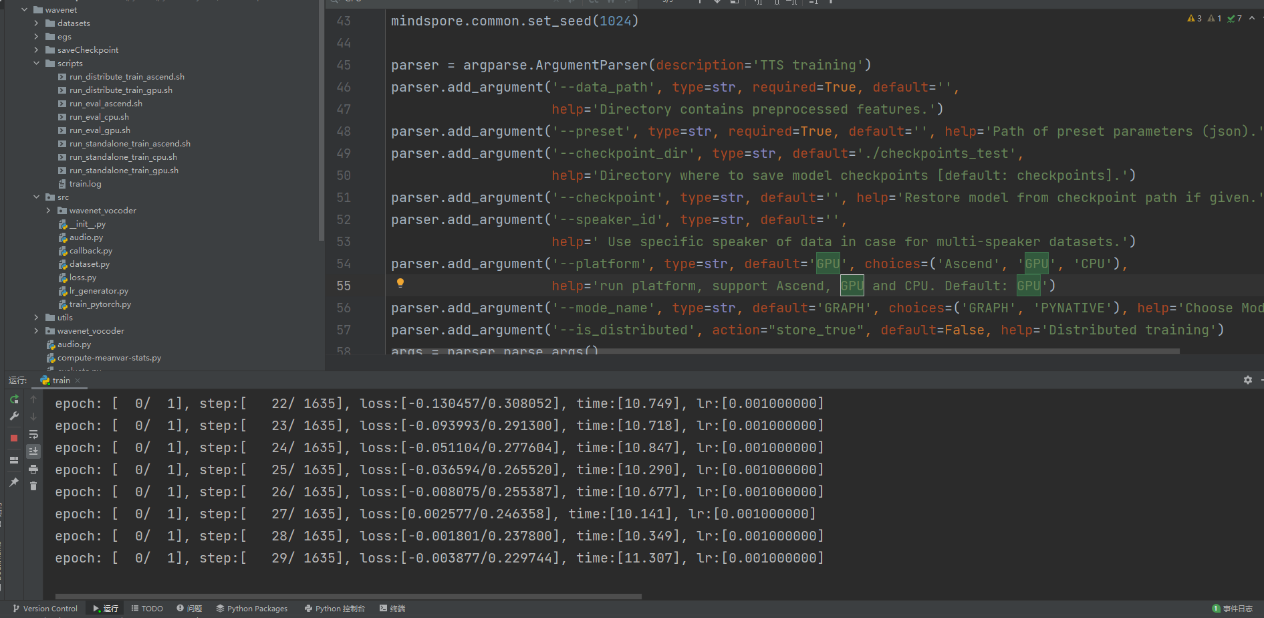
这里，由于硬件受限，我只能使用CPU进行训练，但是效率极其低下，考虑到此实验只要求跑通即可，为了节约时间，最初的epoch次数为2000，我将其修改为1。

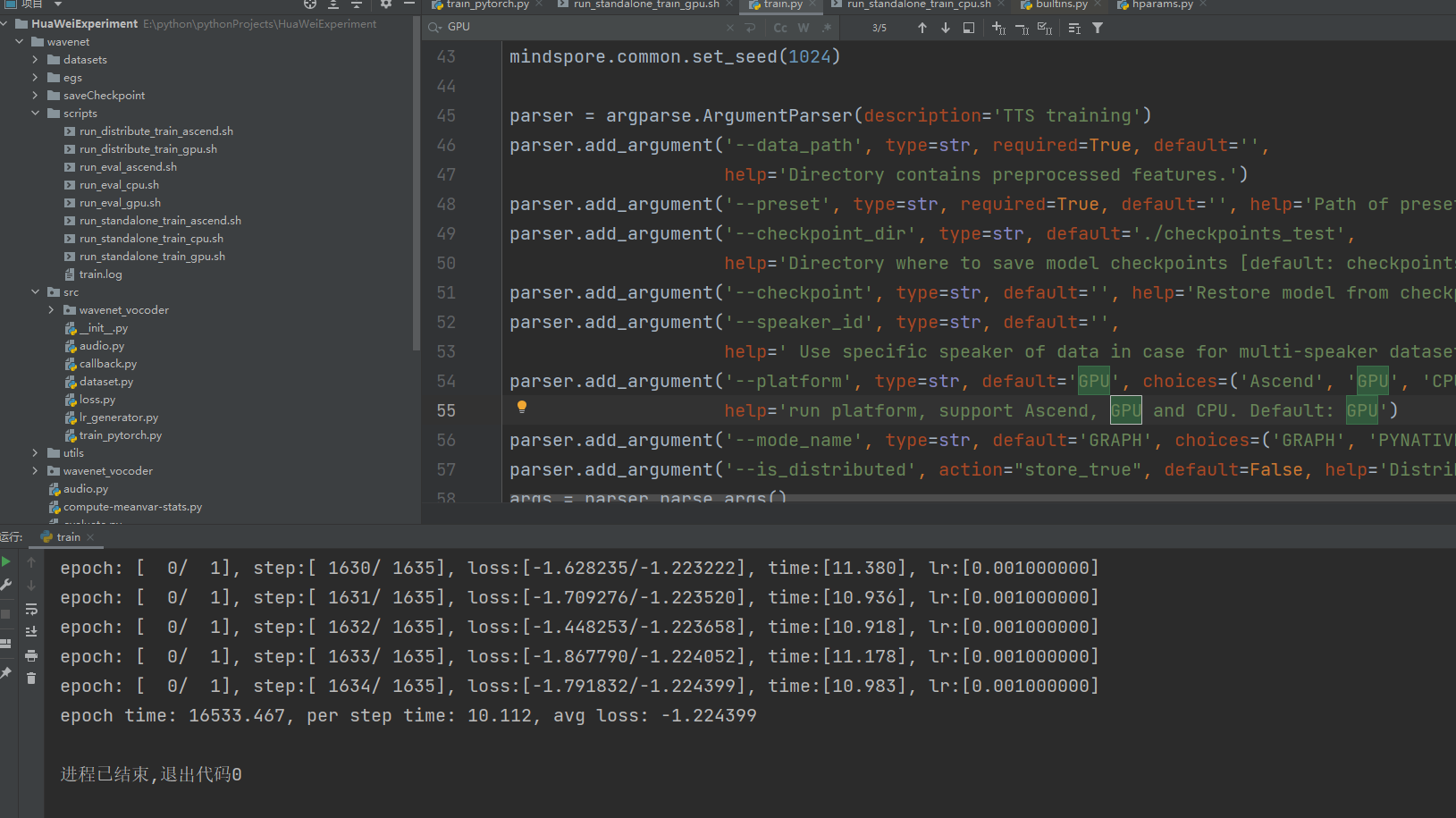
最后我通过Pycharm，设置如下参数：

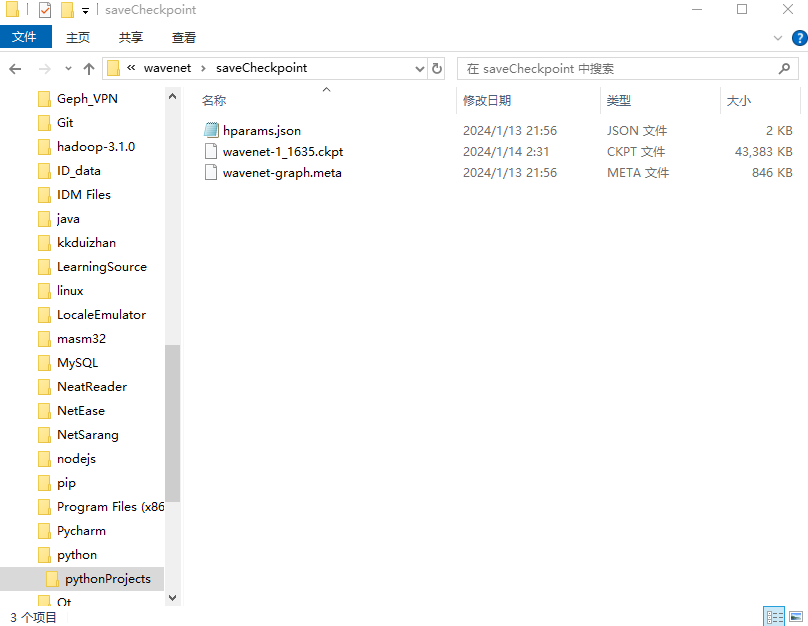
--data\_path ./egs/gaussian/dump/lj/logmelspectrogram/norm/ --preset ./egs/gaussian/conf/gaussian\_wavenet.json --checkpoint\_dir ./saveCheckpoint --platform CPU

然后运行，结果存在./saveCheckpoint目录下。

运行结果如下所示，总用时为16533.467秒，约4个小时35分钟。







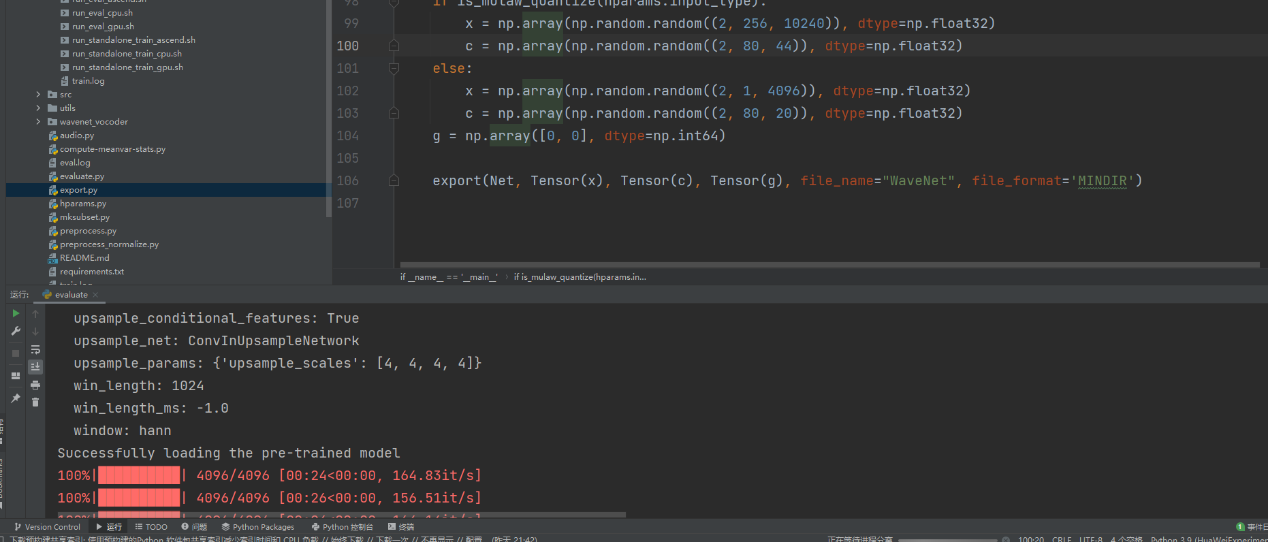
## 评估

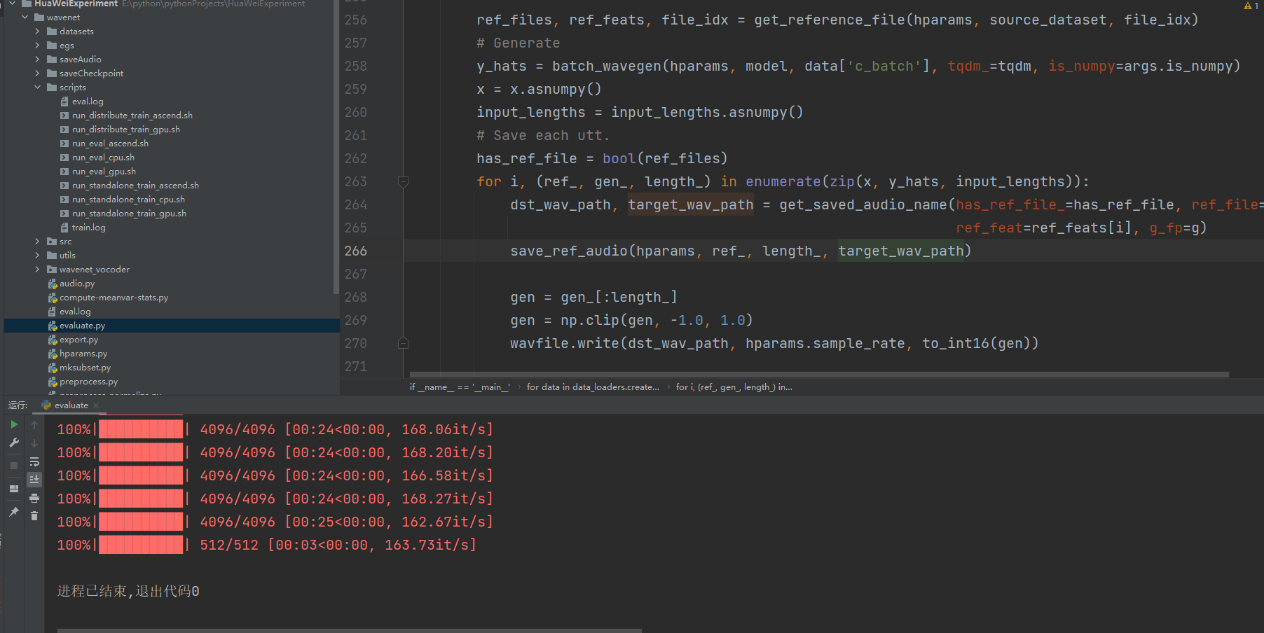
此部分评估主要是人工完成的，因为机器很难判断生成的音频是否是好的。评估的方式是让人听生成的音频和参考的音频，从而评估结果。

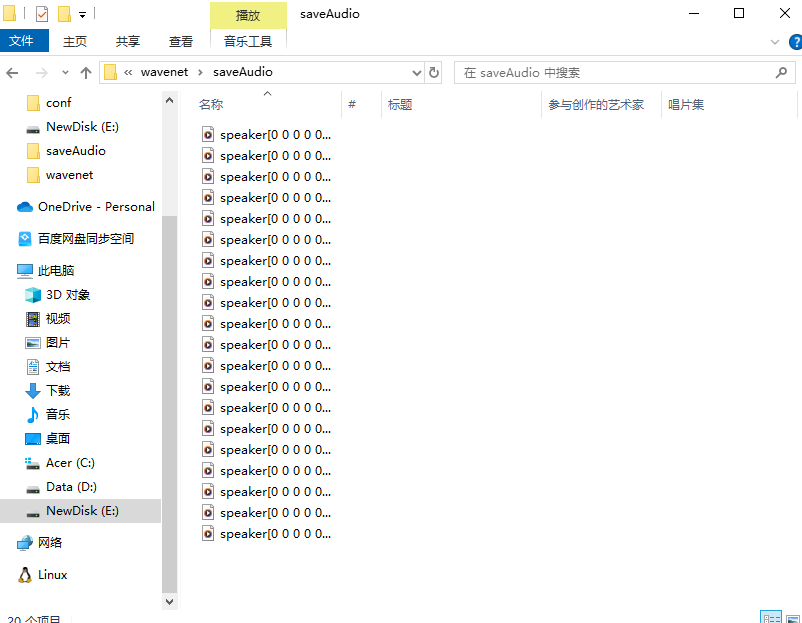
由于硬件限制，这里也只用CPU来进行评估，具体而言就是在Pycharm中参数设置如下

--data\_path ./egs/gaussian/dump/lj/logmelspectrogram/norm/eval/ --preset ./egs/gaussian/conf/gaussian\_wavenet.json --pretrain\_ckpt ./saveCheckpoint/wavenet-1\_1635.ckpt --is\_numpy --output\_path ./saveAudio/ --platform CPU

并且运行evaluate.py文件，结果如下：







## 评估结果

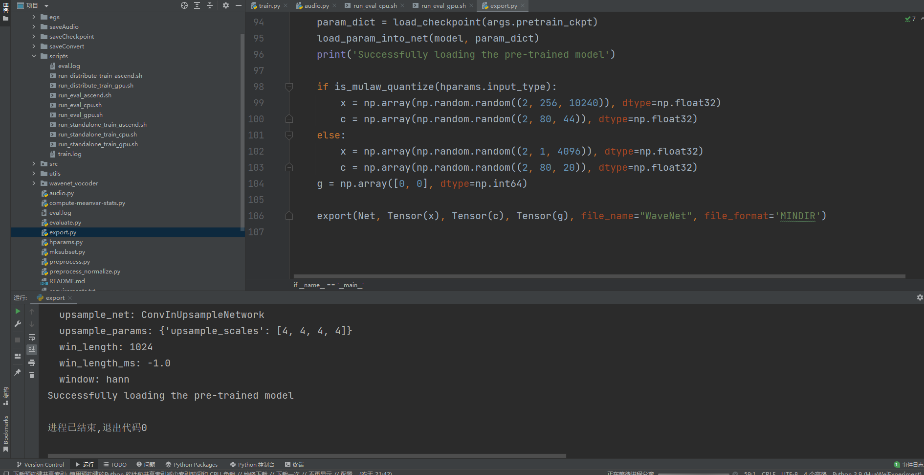
评估结果比较差，生成的音频几乎全是杂乱无章的噪音，可能是因为硬件受限导致训练时间过短，进而引发了这种情况。

## 转换

在Pycharm下设置参数如下，并运行export.py

--preset=.\egs\gaussian\conf\gaussian\_wavenet.json --checkpoint\_dir=.\saveCheckpoint --pretrain\_ckpt=.\saveCheckpoint\wavenet-1\_1635.ckpt --platform=CPU

运行结果如下所示。



作用就是将参数输出出来。