from args import parser

import os

from prepare\_data import create\_data

from train\_model import training

from prediction\_denoise import prediction

if \_\_name\_\_ == '\_\_main\_\_':

args = parser.parse\_args()

mode = args.mode

# Initialize all modes to zero

data\_mode = False

training\_mode = False

prediction\_mode = False

# Update with the mode the user is asking

if mode == 'prediction':

prediction\_mode = True

elif mode == 'training':

training\_mode = True

elif mode == 'data\_creation':

data\_mode = True

if data\_mode:

#Example: python main.py --mode='data\_creation'

#folder containing noises

noise\_dir = args.noise\_dir

#folder containing clean voices

voice\_dir = args.voice\_dir

#path to save time series

path\_save\_time\_serie = args.path\_save\_time\_serie

#path to save sounds

path\_save\_sound = args.path\_save\_sound

#path to save spectrograms

path\_save\_spectrogram = args.path\_save\_spectrogram

# Sample rate to read audio

sample\_rate = args.sample\_rate

# Minimum duration of audio files to consider

min\_duration = args.min\_duration

#Frame length for training data

frame\_length = args.frame\_length

# hop length for clean voice files

hop\_length\_frame = args.hop\_length\_frame

# hop length for noise files

hop\_length\_frame\_noise = args.hop\_length\_frame\_noise

# How much frame to create for training

nb\_samples = args.nb\_samples

#nb of points for fft(for spectrogram computation)

n\_fft = args.n\_fft

#hop length for fft

hop\_length\_fft = args.hop\_length\_fft

create\_data(noise\_dir, voice\_dir, path\_save\_time\_serie, path\_save\_sound, path\_save\_spectrogram, sample\_rate,

min\_duration, frame\_length, hop\_length\_frame, hop\_length\_frame\_noise, nb\_samples, n\_fft, hop\_length\_fft)

elif training\_mode:

#Example: python main.py --mode="training"

#Path were to read spectrograms of noisy voice and clean voice

path\_save\_spectrogram = args.path\_save\_spectrogram

#path to find pre-trained weights / save models

weights\_path = args.weights\_folder

#pre trained model

name\_model = args.name\_model

#Training from scratch vs training from pre-trained weights

training\_from\_scratch = args.training\_from\_scratch

#epochs for training

epochs = args.epochs

#batch size for training

batch\_size = args.batch\_size

training(path\_save\_spectrogram, weights\_path, name\_model, training\_from\_scratch, epochs, batch\_size)

elif prediction\_mode:

#Example: python main.py --mode="prediction"

#path to find pre-trained weights / save models

weights\_path = args.weights\_folder

#pre trained model

name\_model = args.name\_model

#directory where read noisy sound to denoise

audio\_dir\_prediction = args.audio\_dir\_prediction

#directory to save the denoise sound

dir\_save\_prediction = args.dir\_save\_prediction

#Name noisy sound file to denoise

audio\_input\_prediction = args.audio\_input\_prediction

#Name of denoised sound file to save

audio\_output\_prediction = args.audio\_output\_prediction

# Sample rate to read audio

sample\_rate = args.sample\_rate

# Minimum duration of audio files to consider

min\_duration = args.min\_duration

#Frame length for training data

frame\_length = args.frame\_length

# hop length for sound files

hop\_length\_frame = args.hop\_length\_frame

#nb of points for fft(for spectrogram computation)

n\_fft = args.n\_fft

#hop length for fft

hop\_length\_fft = args.hop\_length\_fft

prediction(weights\_path, name\_model, audio\_dir\_prediction, dir\_save\_prediction, audio\_input\_prediction,

audio\_output\_prediction, sample\_rate, min\_duration, frame\_length, hop\_length\_frame, n\_fft, hop\_length\_fft)