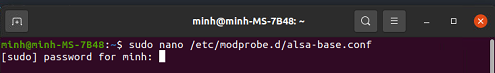
**Instruction for transfer learning with VQ-VAE**

**on PC**

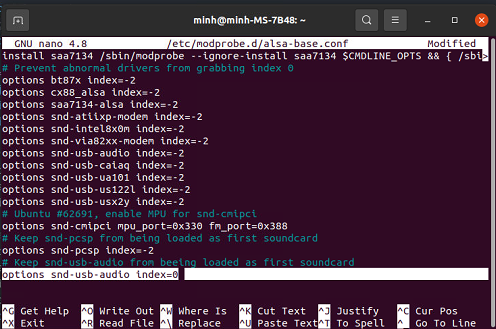
## Setting up the device

Since the experiment is carried out on PC, there are two ways to set up:

* **Using the built-in microphone:** for this choice, there is no need for further configuration.
* **Using an external device** (microphone): to be able to record with external mic, the priority of the device must be the highest one. To do this, first go to the configuration file by entering the below command into a terminal:



Once the configuration file has been opened, go to the last line and **change options snd-usb-audio index** **from -2 to 0.**



Press CTRL C then Y to confirm the change. Now the external mic can be used to record audio.

## Environment and dependencies

* Operating system: Ubuntu 20.04.3 LTS 64 bit
* Python version: 3.8 or higher
* Dependencies: details in the requirement.txt (to install, use: **pip3 install -r requirement.txt**

## Directory structure

In the code folder, there are 6 sub-folders, each with its own use:

* **Data**: this folder is for storing the audio features and their ids.
* **Models**: includes all the definition of different models.
* **Postprocessing**: contains a file defining a helper class that helps with drawing graphs and storing metrics. This is normally used with training and testing process.
* **Preprocessing**: contains all helper functions for preparing the data. These include:
* **extracting\_feature.py**: defines a class that extract feature directly from audio files and store them in npz format (numpy format).
* **import\_data.py**: helps with import data from .npz file into numpy arrays.
* **trunking\_audio.py**: cuts down the long audio file into smaller, desirable trunks.
* **utils.py**: includes some small functions that might be useful.
* **Results**: contains graphs and metrics from training and testing procedures. The models from training are also saved here.
* **Trainer**: defines a helper function for training.

For training and evaluating, these following files can be modify to suit the purpose of the user:

* **b\_training.py**: defines the base training process (setting up => loading data => training model => saving the results). The trained model will be saved in

**./training\_on\_pc/Results/Saved\_models/vq\_vae/base**

and the graphs and metrics will be in

**./training\_on\_pc/Results/Graphs/vq\_vae/base**

* **tl\_training.py**: just like base training process but this time with transfer learning.The trained model will be saved in

**./training\_on\_pc/Results/Saved\_models/vq\_vae/tl**

and the graphs and metrics will be in

**./training\_on\_pc/Results/Graphs/vq\_vae/tl**

* **evaluating.py**: when the test set is cleaned up but the model doesn’t need re-train, use function defined in the file to have all the performance evaluated. Graphs and metrics will be stored in

**./training\_on\_pc/Results/Updated\_Graphs/vq\_vae/tl**

or

**./training\_on\_pc/Results/Updated\_Graphs/vq\_vae/base**

based on the ‘--transferLearning’ flag

* **inferencing.py**: performs running test.
* **monitoring.py**: monitors the CPU, RAM usage and the temperature of the running device (most likely Raspberry Pi 4)
* **rt\_test.py**: runs a real-time test
* **micUSB\_cmd.py**: records the audio from microphone to

**./training\_on\_pc/test\_samples/test**

* **audio\_cleanup.py**: remove all the 2 second recorded audio files from

**./training\_on\_pc/test\_samples/temp**

and save them as a concatenated audio file in

**./training\_on\_pc/Data/history**

with the date\_time as file name.

And lastly, to perform these processes with ease, main\_program.py should be run in terminal with suitable flags. The details of running this file will be discussed in the next part.

## Data preparation

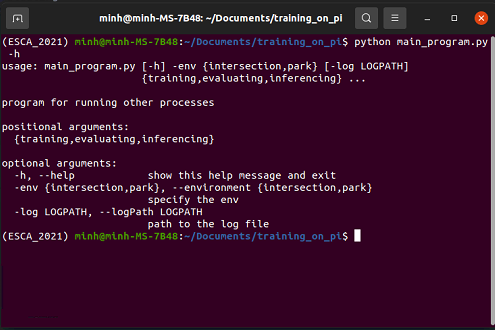
### Cutting up audio

To get a smaller size audio, use the **split\_audio.py** script. Change the source folder location (**src**), the destination folder location (**dst**) and the duration of each file (**duration**) to get the desire results.

### Extracting features

To prepare data for the training process, the gammatone features must be acquired. This can be done by the script features.py. Again, before running the script, the audio files location (**norm\_audio** for normal audio **and abnorm\_audio** for anomaly) must be set correctly. Another important variable is **feature\_folder**. This variable defines the location of the feature, which should be under the Data directory (different locations is also acceptable but some changes in loading up data in b\_traiining.py and tl\_training.py must be done). To set the audio length over which the gammatone feature can be captured, use the variable **length**. Note that a 60-second audio file can be used directly to extract 30 features (without having to cut up the file first, then extracting the features).

## Running instructions



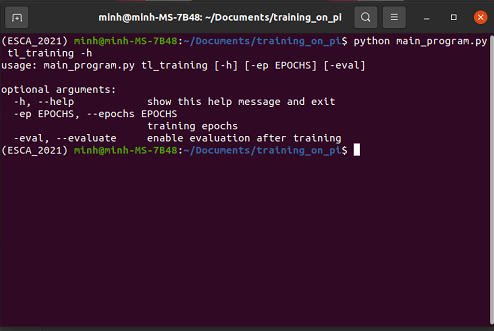
There are 3 base arguments when running main\_program.py:

* **--environment (shorthand: -env, required)**: specify the environment for the upcoming process
* --logPath (shorthand: -log, optional): specify the location of log file (saved in json format)
* process to carry out: choose among training, evaluating and inferencing

### For transfer learning training

When choosing transfer learning training process (tl\_training), there are 2 additional arguments:

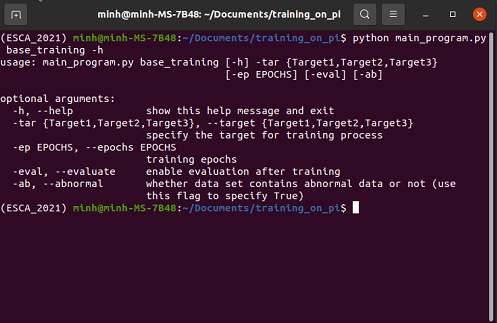
* --epochs (shorthand: -ep, optional): set the epochs for training (default to 60)
* --evaluate (shorthand: -eval, optional): choose to perform evaluating on test set (default to False)



### For base training

When choosing base training process (base\_training), there are 4 additional arguments:

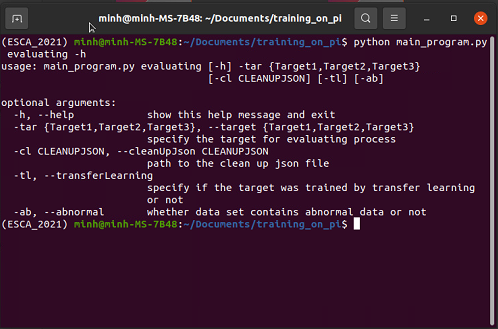
* --epochs (shorthand: -ep, optional): set the epochs for training (default to 60)
* --evaluate (shorthand: -eval, optional): choose to whether perform evaluating on test set (default to False)
* **--target** (shorthand: -tar, **required**): specify the target for base training process
* **--abnormal** (shorthand: -ab, optional): **call this flag if there are anomaly samples in the target’s data set**



### For evaluating

When choosing evaluating process, there is 4 additional arguments:

* **--target** (shorthand: -tar, **required**): specify the target to perform evaluation on
* --cleanUpJson (shorthand: -cl, optional): often the test set need to be cleaned before being re-evaluated again. This argument specifies the path to the clean-up file.
* --transferLearning (shorthand: -tl, optional): whether the target was trained by transfer learning scheme (default to False).
* **--abnormal** (shorthand: -ab, optional): **call this flag if there is no anomaly samples in the target’s data set**

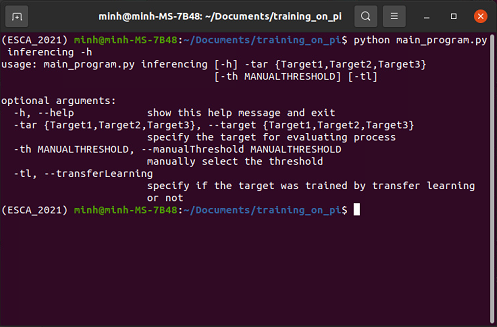


### For inferencing

When choosing inferencing process, there is 3 additional argument:

* --manualThreshold (shorthand: -th, optional): specify the hand-picked threshold value for a certain inferencing process (default to the threshold with highest combination of precision and recall score when evaluating with clean-up set)
* **--target** (shorthand: -tar, **required**): specify the target to perform evaluation on
* --transferLearning (shorthand: -tl, optional): whether the target was trained by transfer learning scheme (default to False).

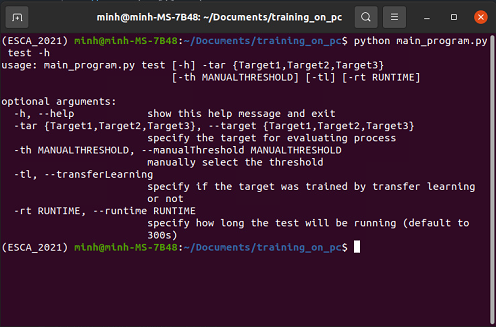
**Note:** the inferencing test here is only loading the file in and perform prediction.



### For real time test

When choosing real time testing process (test), there is 4 additional argument:

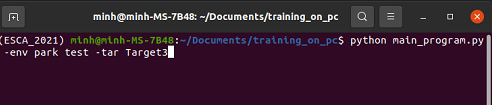
* --manualThreshold (shorthand: -th, optional): specify the hand-picked threshold value for a certain inferencing process (default to the threshold with highest combination of precision and recall score when evaluating with clean-up set)
* **--target** (shorthand: -tar, **required**): specify the target to perform evaluation on
* --transferLearning (shorthand: -tl, optional): whether the target was trained by transfer learning scheme (default to False).
* --runtime (shorthand; -rt, optional): set a timer for running the process (default to 5 minutes)



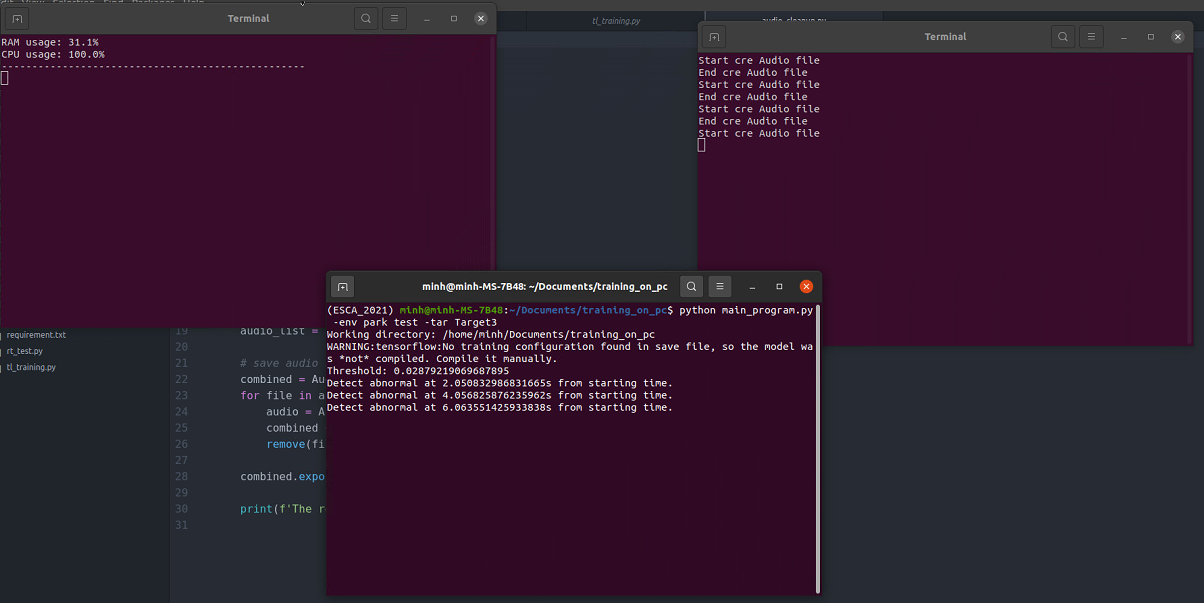
### Example

This example showed the input to a real time testing process:

* Run the **main\_program.py** with test positional argument (for this test the park environment was chosen). Also, specify a target with **-tar flag**, the choices are among source, Target1, Target2 and Target3. Note that for a target to use, that target **must have been re-evaluated** or a **manual threshold must be set**. Optionally, the runtime could be set by -rt flag (default to 5 minutes) and base model could be use by calling the -tl flag (using transfer learning model by default)



Results:



**Note**: the program can be stopped by keyboard interrupt CTRL C. The testing results should still be saved