How to Train Your BeatSaber Automapper with InfernoSaber

System Spec

Recommended setup for training:  
- CPU: -  
- GPU: NVIDIA GPU with >= 8GB VRAM  
- RAM: >= 24GB  
- OS: Linux, (or WSL2 with Windows)  
- (Minimal spec for around 50-150 songs, depending on the variety of the songs)

WSL2

If you do not have a Linux/dual boot system available, you can follow the internet guides to setup WSL2 on Windows and increase the RAM allowance for WSL2 to the maximum (minus a bit for Windows itself). This is tested to work with NVIDIA 30th generation.

How to install

Download/Clone the “main” repository from <https://github.com/fred-brenner/InfernoSaber---BeatSaber-Automapper>.

I recommend having PyCharm or VSCode available to view and edit Project files.

To install the dependencies:

* <https://learn.microsoft.com/en-us/windows/python/web-frameworks#install-windows-subsystem-for-linux>
* sudo apt update && sudo apt upgrade
* Download Python 3.10 (in theory, 3.9 to 3.12 should work as well)
* You will need those dependencies:
  + libswresample-dev
  + libsamplerate-dev
  + libsndfile-dev
  + txt2man
  + doxygen
  + if available you can also try to install aubio via linux first:
  + python3-aubio
  + aubio-tools
* Other commands you may try in case of errors:
  + sudo apt-get install ffmpeg libavcodec-extra
  + sudo apt-get install libavcodec-dev libavformat-dev libavutil-dev libswresample-dev
  + pip uninstall -yv aubio
  + pip install --force-reinstall --no-cache-dir --verbose aubio
* wsl --mount C:\Users\frede\Desktop\BS\_Automapper (C:/ Drive will be reachable via /mnt/c/
* pip install -r requirements.txt (inside the InfernoSaber project folder) and it ran fine, and I then configured the /tools/config/paths.py to point to my desired folders

Change the config

*config.py*

What you will like to change for training:

* use\_mapper\_selection: select your new model name (lowercase, no spaces etc.)
* use\_bpm\_selection: set to false if you want advanced sorting by mapper etc. (recommended to leave true)
* min\_bps\_limit: minimum bps for songs to consider for training. Should be > 1 to make sure no empty/lightshow songs are loaded.
* max\_bps\_limit: maximum bps for songs to consider for training.
* training\_songs\_diff: Default difficulty to load for training
* allow\_training\_diff2: If default difficulty is not available search for second one
* training\_songs\_diff2: second difficulty to load for training if first one not available
* vram limit: approximate limiting of song imports based on GPU VRAM in GB. Depends on the variety of the songs, adjust based on trial n error. Required for lighting/event model.
* autoenc\_song\_limit: should be fine, in case of error reduce
* mapper\_song\_limit: adapt based on your RAM usage. As initial guess, assume 200 per 30GB
* beat\_song\_limit: same as above

If more songs (than \_song\_limit) are available during training, they will be randomly discarded.

Run the training

Folder architecture:

Before starting the scripts, best take a look at the folder structure which comes along with the Pinokio app. Copy the same architecture to your “Data” folder for Linux or reference the config.py to the pinokio “Data” folder.

**Prepare the Input songs:**

Make sure you have a copy/backup of all your songs, or better, copy the beatsaber song folder somewhere and paste this directory into the config (bs\_input\_path). It will notice you before if it makes any changes but working on the copy only is recommended.

If you want to load your favorite maps:

*hashtest.py* – adapt paths at bottom and run to find your favorite songs

*copyfavorites.py* – adapt paths at bottom and run to copy your favorite songs

Once you have your songs set, clean them with:

*bs\_shift/cleanup\_n\_format.py* - **NEEDS to be run from the project folder, not bs\_shift**

Lots of maps have different naming conventions, e.g., for naming the ExpertPlus.dat file. So after you have copied them, you can run the bs\_shift/cleanup\_n\_format.py on the copy folder which will be able to format most v2 maps into the default format expected by the later script main\_training.py

*main\_training.py*– actual training script, each training can be repeated in case of errors.

add for every of the 5 training runs an 'y', so what you want first is to type yyyyy for all of them running consecutively. This is just so if one fails you can restart from this point by typing for example nnnyy.

Test your environment

*python3 -c "import tensorflow as tf; print(tf.config.list\_physical\_devices('GPU'))"*

…should result in …

[PhysicalDevice(name='/physical\_device:GPU:0', device\_type='GPU')]

*free*

… will show you the amount of memory available to Linux

Troubleshooting

In case of any errors and questions, let me know via Discord. The improvements channel also holds the chat of one successful implementation.