

How to CS with visualization

Inhalt

Preconditions:	1
Remote visualization (recommended):	2
Connection without visualization (command line)	2
Setting up software environment:	2
Data management:.....	3
Remote offline structure	3
GitLab as code base management tool:	4
Setting up the first ML pipeline	4
GPU computing:	7
General remarks:.....	11
Versioning:.....	12
Open Questions:.....	12

Preconditions:

1. **Login** for the **CS system** is provided by the UniRZ. Application form and a guide to remote connection and visualization is provided here:

<https://www.tu-ilmenau.de/hpc>

Note: The Application form needs some time due to many signatures and personal fetching of your login credentials at the UniRZ terminal. You cannot start right away. Further you will get instructions via mail on the usage of the CS remote systems in terms of storage of data, file exchange and occupation of the machines. Read this carefully and stick to the guidelines.

2. Remote desktop can only be accessed from the intranet of the TU. So, if you want to access outside the TU network you should use a **VPN connection**. The university provides software and configuration instructions here: <https://intranet.tu-ilmenau.de/site/unirz/Freigegebene%20Dokumente/KommNetze/AnyConnect/Installation%20und%20Benutzung%20des%20Cisco%20AnyConnect%20Secure%20Mobility%20Clients%20unter%20Windows%2010.pdf>

3. In addition, the **NICE DCV client** (<https://cloud.tu-ilmenau.de/s/TmNBZ5ez5TFipJp>) must be installed on your local machine. In the later usage you do not need to actively start this client. It will be automatically called from the browser when you start your remote session with the right configuration.

Remote visualization (recommended):

Following the guide from the HPC page you should be able to connect to a remote desktop machine on the VIS page (<https://www.tu-ilmenau.de/vis>) with your credentials from the UniRZ.

- Click on the desktop session option on left column to initiate a new remote desktop instance
- Once the desktop session starts, click on the hyperlink under the “sessions” section, this will automatically download a NICE DCV interactive page
- Open this page and login with you CS credentials
- recommend: one session; maximum: one session
- Runtime: twelve days
- Please run in general one session!!!
- Hint: if useful You can share Your session with other users

Connection without visualization (command line)

Windows:

- Install putty on your local machine
- putty cslogin.tu-ilmenau.de
- file transfer via WinSCP (freeware)
- WinSCP server address: to csdata.tu-ilmenau.de (SFTP connection with personal credentials)

Linux:

- ssh cslogin.tu-ilmenau.de
- winscp|sftp|rsync to csdata.tu-ilmenau.de
- don't connect to cslogin.tu-ilmenau.de for file transfer purposes!!

Setting up software environment:

To execute a ML pipeline, train neural network models and evaluate them we need a software environment installed. Here we propose to use anaconda, a python container, that has useful packages pre-installed.

Install Your python distribution/packages in Your \$HOME located on the scratch4 file system!!!

Here you can find useful guides to install anaconda without root access on a CentOS environment:

- <https://arabelatso.github.io/2020/01/09/Install%20Anaconda%20on%20CentOS%20without%20root/>
- <https://linuxize.com/post/how-to-install-anaconda-on-centos-7/>
 - in short download the latest package from
 - in the terminal change to the download directory
 - run: “bash Anaconda3-5.3.1-Linux-x86_64.sh”
 - install and initialize anaconda
- **Attention:** When installation is finished **abort** the following **initialization**. Then initialize manually via “conda init tcsh” from the Anaconda root folder.
- To run anaconda navigator:
 - Click on applications>system tools> Terminal
 - Then run in Terminal: anaconda3/bin/anaconda-navigator
 - From there you have multiple options of creating code and running it. We recommend spyder as IDE since this adds great functionality in terms of debugging, plotting, inspection and many more.
 - For detailed information on the usage of spyder see the example in chapter “Setting up the first ML pipeline” also visit <https://docs.spyder-ide.org/current/installation.html> for first steps

Note: You might need different Conda environments for different projects since every project has its own toolchain with specific packages from specific versions. To manage your Conda environments you will find a useful guide here:

<https://docs.conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html>

Data management:

Remote offline structure

The primary working space is located at: `/usr/scratch4/[your_login_name]`. The max. available memory is limited to 1 Terabyte or 500.000 files. It is not backed up! A separate Backup location is reserved here: `/usr/wrk/people9/[your_login_name]`. The max. available memory there is limited to 100 GByte or 100.000 files, respectively. It is recommended to copy datasets, code, and repositories to the primary working directory which is your home directory. Only use the Backup location for important data to avoid “ghost folders” and “garbage collection”. Also please check your backup folder continuously and remove legacy content.

When connected to your virtual session you can use the command:

```
>> show_user_info
```

to check your current usage of memory as well as your limits.

GitLab as code base management tool:

The GitLab repository of our Lab is located at: https://avt10.rz.tu-ilmenau.de/git/users/sign_in

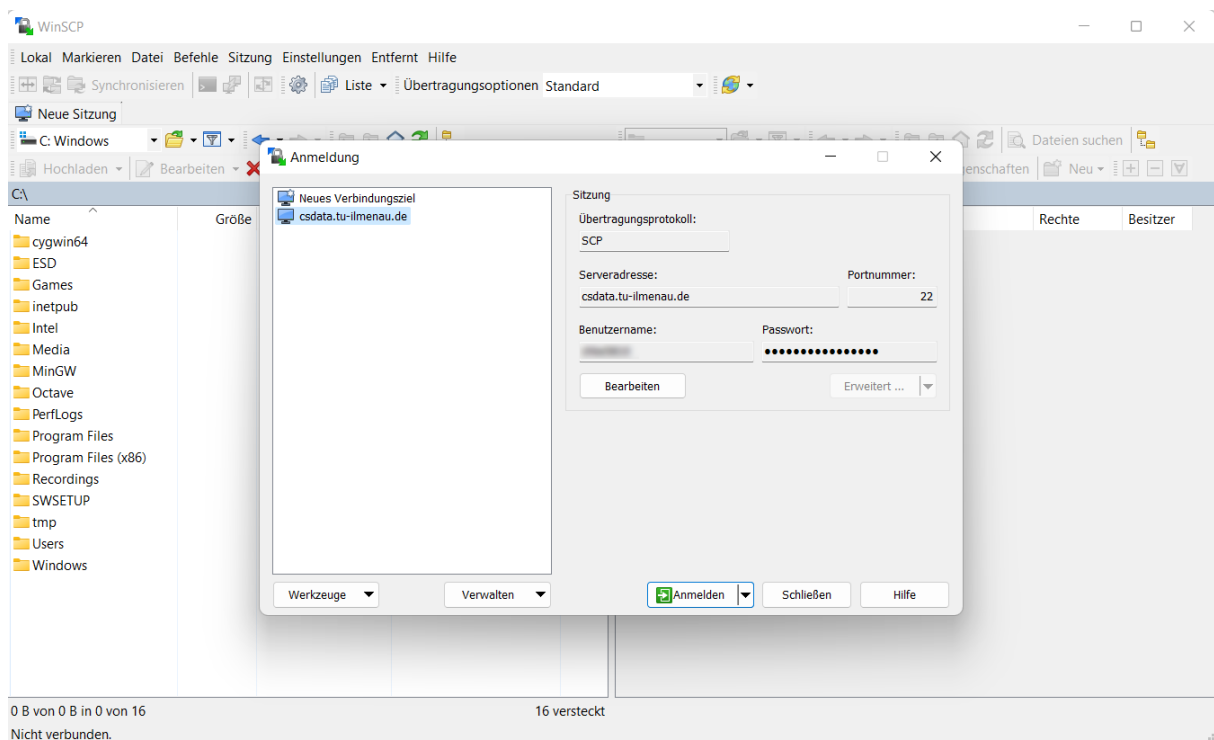
For access to the GitLab repository please contact Bernd Hildebrandt for Login: "Bernd Hildenbrandt" bernd.hildenbrandt@tu-ilmenau.de.

It is recommended to use git for backup and cloud storage of your code. If you are not familiar with the usage of git you can find several tutorials online. Git is a built-in functionality on CentOS.

Setting up the first ML pipeline

1. Copy data

For copying data i.e., training sets, use WinSCP or other ssh clients like PuTTY. The server address is `csdata.tu-ilmenau.de`. Login with your CS credentials. If you are not in the university network, remember to activate your VPN.



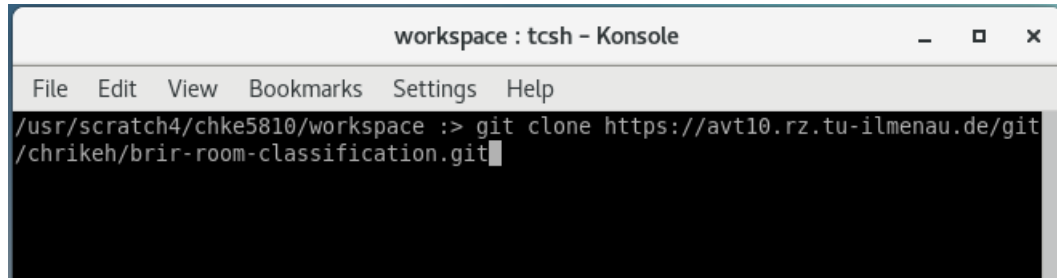
In this example a dataset is available at: <https://cloud.tu-ilmenau.de/s/tay6KRinPiG6Fik> Please download the data from there. Create a 'data' sub folder in your home directory and copy the folder there.

2. Checkout repository

It is recommended to create a 'workspace' subfolder to collect your work. The example repository should also be checked out in this folder. Use a command line tool (i.e., Terminal) to navigate to your

workspace folder and check out the git repository (<https://avt10.rz.tu-ilmenau.de/git/chrikeh/brir-room-classification.git>) using:

```
">> git clone [address_of_the_repository]"
```



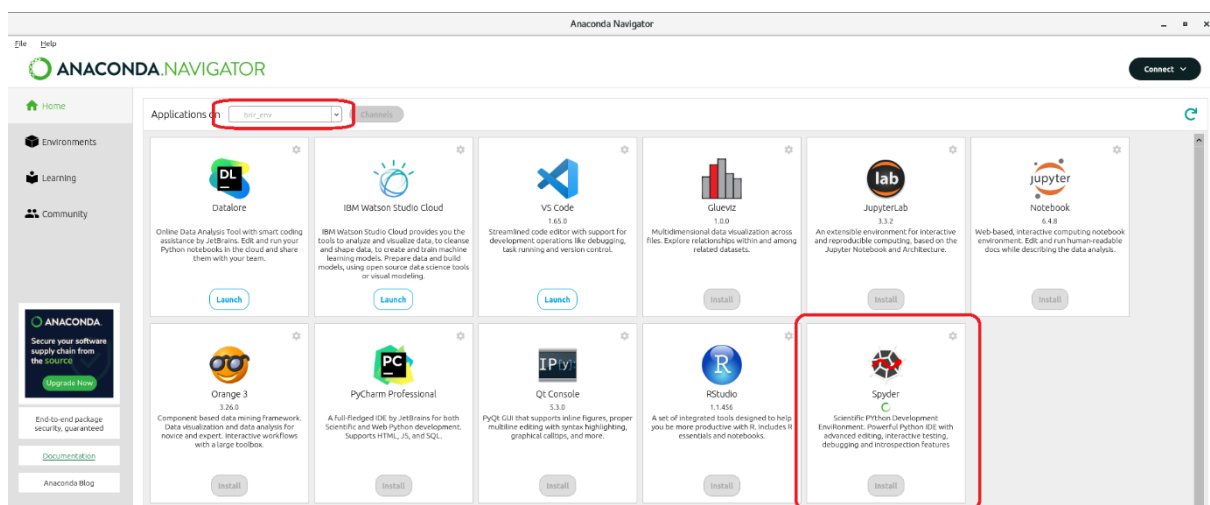
When running this for the first time you have to import the conda environment that sets up all needed python packages to run the code. To do so change directory in the command line tool to the repository and import the environment with:

```
>> conda env create -f brir_environment.yml
```

3. Start the IDE

a) Via conda navigator:

- Start the Conda Navigator in your command line tool from your anaconda installation folder:
- `>> /usr/scratch4/[usr_name]/anaconda3/bin/anaconda-navigator`
- When the GUI of the navigator has started select the environment from the top dropdown menu. Then install and launch spyder.



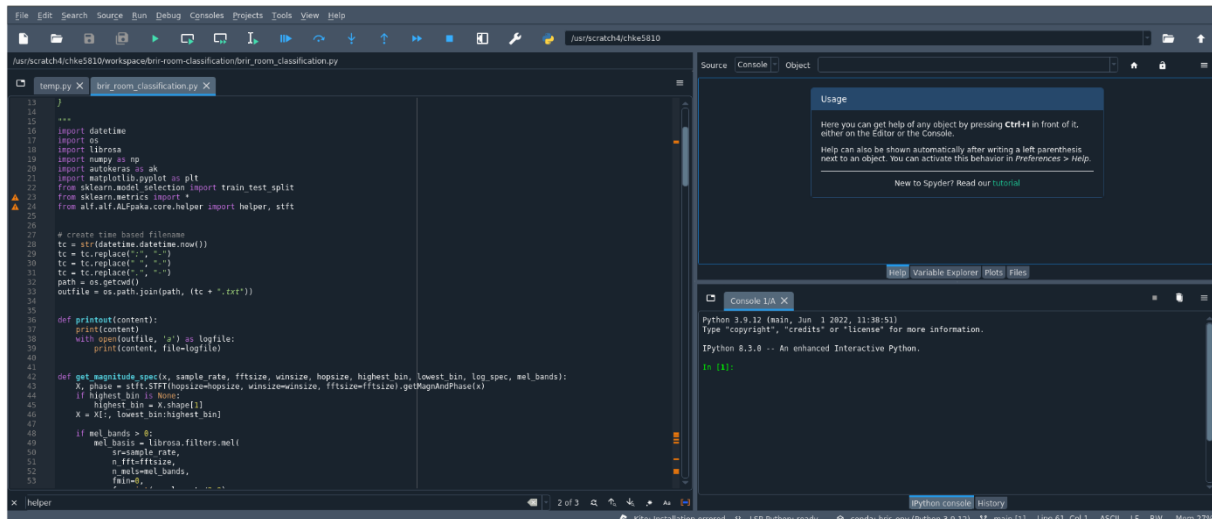
b) Via command line tool:

Make sure you already created the brir environment from the previous step then activate the environment in your command line tool with:

```
>> conda activate brir_env
```

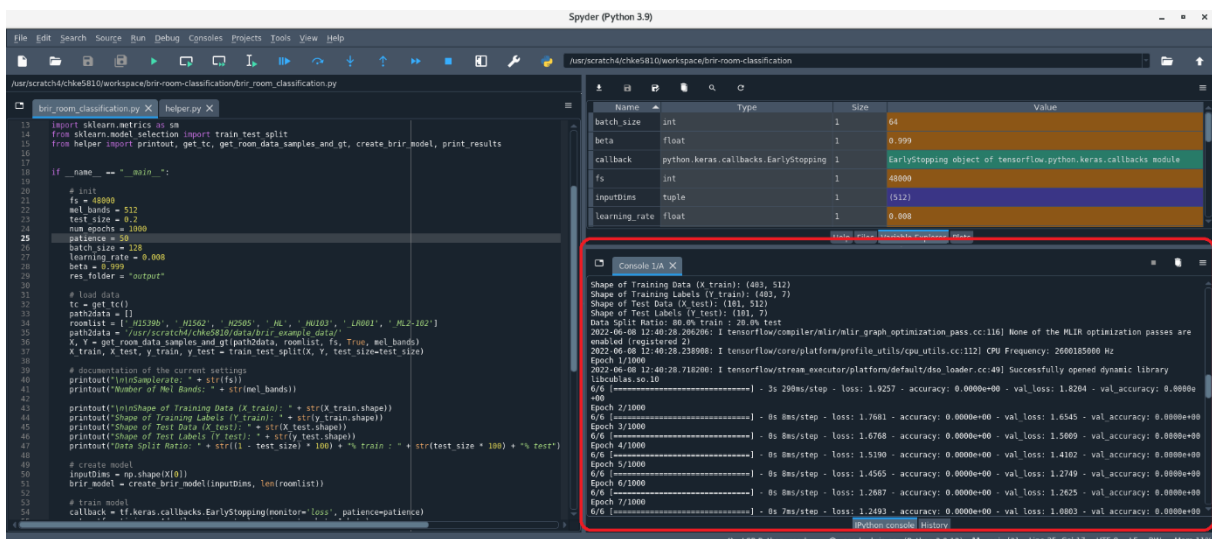
From the Anaconda installation directory start spyder with:

```
>> /usr/scratch4/[usr_name]/anaconda3/bin/spyder
```



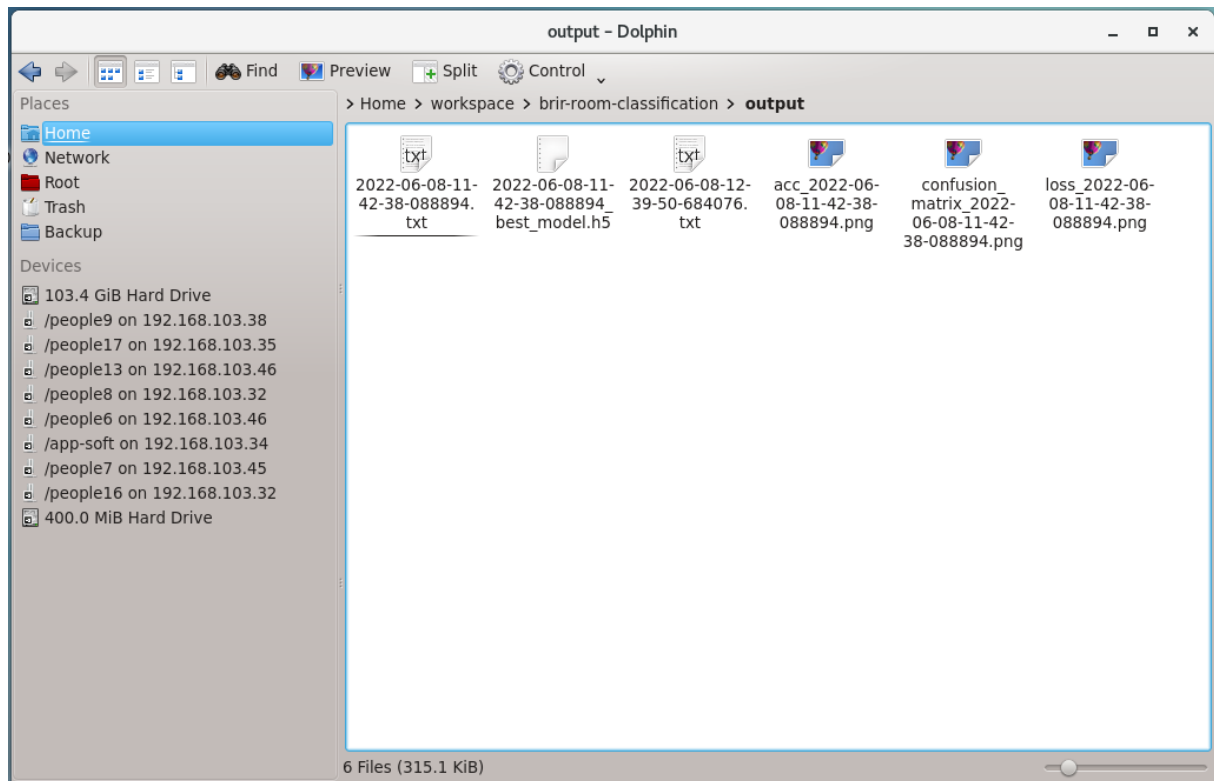
4. Run the script

- Open the script “brir_room_classification” in the editor
- Run code via green arrow or through the menu and check the output



- Check the working directory for output files
- In `./workspace/brir_room_classification/` a new folder ‘output’ should be visible
- The folder contains:
 - Log file `[timestamp].txt` of the most important training information

- The trained model stored in .h5 format
- 3 plots of the loss history, the accuracy history, and the confusion matrix of the test set



Troubleshooting: Some packages might not be imported due to different platforms. Please check the error output of python carefully and try to manually install missing packages with:

```
>> conda install [package_name], or
```

```
>> pip install [package_name]
```

You can find more general machine learning examples for audio here: <https://keras.io/examples/>

GPU computing:

The GPU visible in the interactive session is **NOT** the GPU device for GPU computing. This GPU is only in charge of visualizing the virtual session. Also avoid testing your scripts there since this is a shared GPU to visualize many virtual sessions at once. For PoC computations use “makalu86”:

To connect to makalu86 open the terminal and type:

```
>> ssh -X makalu86
```

Note: In some cases the connection fails due to conda installation issues (wrong config in `vi .tcshrc` script). If so, please contact Henning Schwanbeck (hpc@tu-ilmenau.de) for a fix.

Then login with your credentials, activate your conda environment, and browse to your script. When your scripts are running fine, switch to batch jobs for processing. The makalu86 unit is only for testing. It cannot be ensured that your job is running solemnly on the system. To ensure that and access a dedicated GPUs use a batch script and hand over the script you want to execute:

for certain software we offer generic batch job scripts:

- matlab.batch inputfile.m
- comsol.batch inputfile.mph
- feko.batch inputfile.cfx
- batch.1gpu scriptname (i.e., python)

Further GPUs can be accessed via batch jobs with the following scripts:

```
batch.1gpu user_script
```

```
batch.2gpu user_script
```

```
batch.4gpu user_script
```

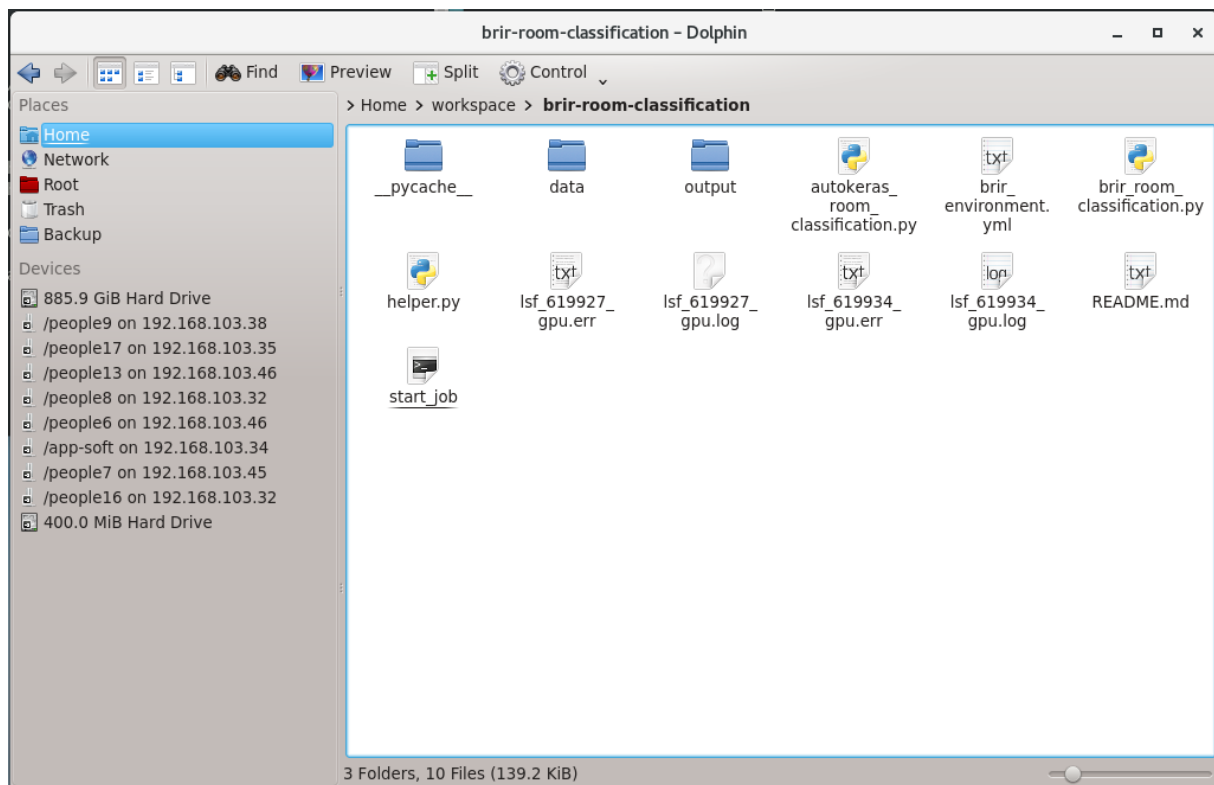
Replace “user_script” with the commands or scripts you want to execute.

Script „batch.1gpu“ starts a batch job on one GPU knot – und calls GPU.

Script „batch.2gpu“ starts a batch job on one GPU knot – und calls 2 GPUs.

And so on...

The output/return of the batchjobs is logged in the working directory with 2 files, a gpu.log and a gpu.err file. Their names inherit the pid of the batchjob, i.e.: `lsf_23423_gpu.log` and `lsf_23423_gpu.err`. In the example above the output folder looks like this:



The batchjobs can be spectated via:

```
>> bjobs
```

There are several useful commands to find, kill, or see the progress of your batch jobs, like:

```
>> bjobs -q BatchXL
>> BSUB -q BatchGPU
>> bqueues -l BatchGPU
>> bjobs -l 619927
```

Please check <https://www1.tu-ilmenau.de/hpcwiki/doku.php?id=commands> for a more detailed overview. A full documentation is available here: https://www1.tu-ilmenau.de/hpcwiki/lib/exe/fetch.php?media=wiki:lsf6.2_using.pdf

For fluent usage you can create your own script that works through some python commands that can be called in combination with the batch.Xgpu command. For example, create a start script called "start_job" in your working dir and edit the content, like:

```
date
pwd
conda env list
conda activate brir_env
which python

python brir_room_classification.py

date
```

Then change the permissions of the script with: “chmod 700 start_job” to make it executable.

You can monitor the usage of the GPU on the connected machines with:

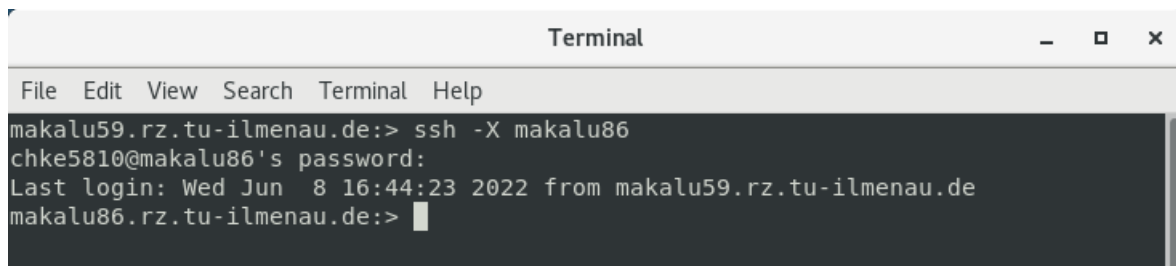
```
>> nvidia-smi
```

There you can also see the process IDs of the running processes. With these IDs you can access more info, using:

```
>> ps -eaf|grep pid
```

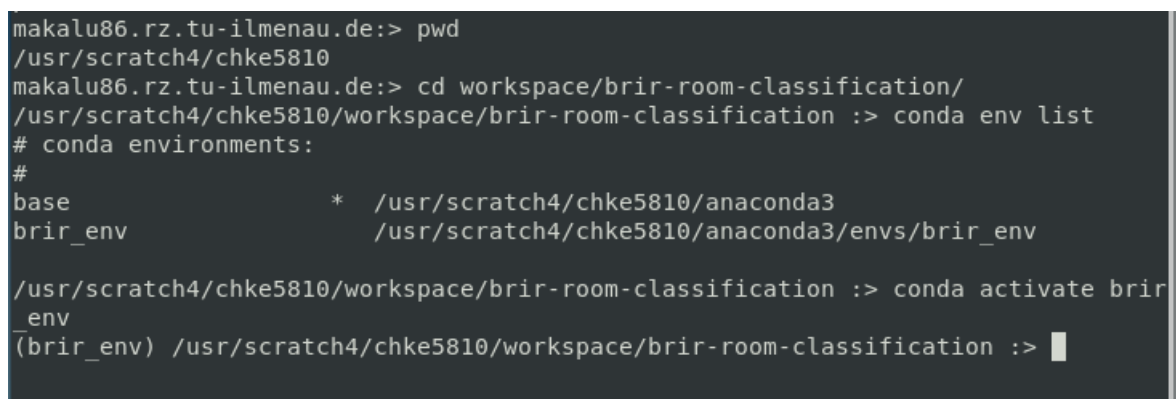
... continuing with the example code here:

In the case of our example, we can open a separate terminal and connect to makalu86 for testing the script on the GPU:



```
Terminal
File Edit View Search Terminal Help
makalu59.rz.tu-ilmenau.de:> ssh -X makalu86
chke5810@makalu86's password:
Last login: Wed Jun  8 16:44:23 2022 from makalu59.rz.tu-ilmenau.de
makalu86.rz.tu-ilmenau.de:>
```

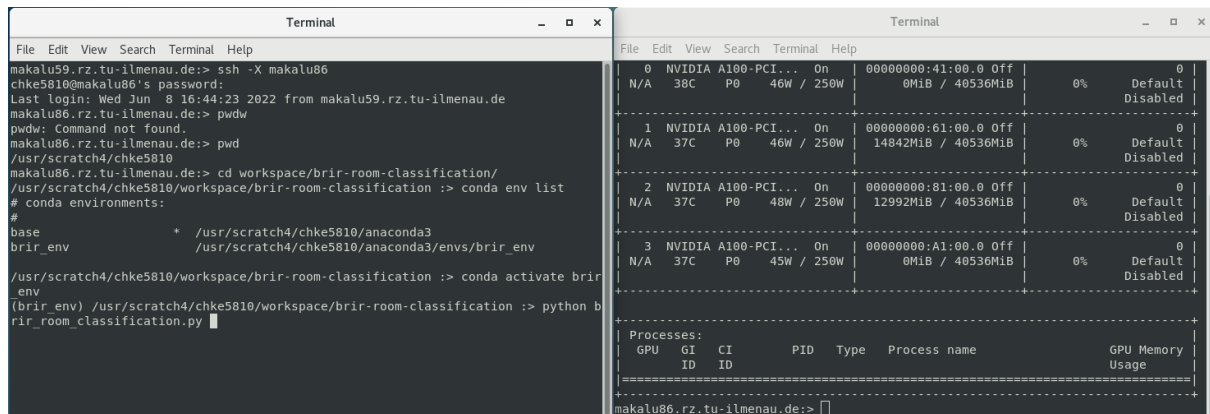
Here we navigate to the working directory of the script and activate the conda environment that shall be used for execution:



```
makalu86.rz.tu-ilmenau.de:> pwd
/usr/scratch4/chke5810
makalu86.rz.tu-ilmenau.de:> cd workspace/brir-room-classification/
/usr/scratch4/chke5810/workspace/brir-room-classification :> conda env list
# conda environments:
#
base                *  /usr/scratch4/chke5810/anaconda3
brir_env            /usr/scratch4/chke5810/anaconda3/envs/brir_env

/usr/scratch4/chke5810/workspace/brir-room-classification :> conda activate brir_env
(brir_env) /usr/scratch4/chke5810/workspace/brir-room-classification :>
```

From here you can simply execute the script by calling it from the terminal. In a second terminal you can also connect to Makalu86 and monitor the GPU usage with the “nvidia-smi” command.



When executing the as a batch script it is necessary to create a startup bash script (named here: `start_job`) to activate the correct environment:

```
# minimum startup script
conda activate brir_env
python brir_room_classification.py
```

Save this script and call it as an argument to the actual command for the GPU scripts:

```
batch.1gpu start_job
```

This will add the script to the queue and execute it in the next free scheduled slot on a free GPU. The output will be saved in the current working directory. Not console output is displayed during execution.

General remarks:

- Don't use special signs and "spaces"
- Setup Your own properly directory structure (\$HOME/work/...)
- Don't save data files in predefined directories like Documents Videos!!!
- Check Your quota frequently with the command: `show_user_info`
- Give Your application license "free" for other users!!! Close Your licensed application (ansys, matlab, comsol, other) before you disconnect from Your interactive session
- Clean up| Empty Your Trash Container frequently Goto: Applications >> Accessories >> Files >> Trash >> Empty ... Your trash content is included in Your file system limits
- Don't use tools like "top" or "htop" or similar over a long time period - it consumes cpu resources

Versioning:

Anaconda v. 3.5.3.1
Python v. 3.9.12
Spyder v. 5.1.5
Tensorflow v. 2.4.1
CUDA v. 11.2

Note: Some versions have dependencies to each other. Python, tensorflow, Cuda and Cudnn have to match to successfully work on the system. You can find a compatibility list of these components here:

<https://www.tensorflow.org/install/source#gpu>

https://docs.nvidia.com/deeplearning/frameworks/tensorflow-release-notes/rel_21-03.html

Open Questions:

hpc@tu-ilmenau.de

christian.kehling@tu-ilmenau.de