CVHCI Onboarding

Account and Infrastructure

Your advisor should have already provided you with an account for using the CVHCI infrastructure, e.g. Max Mueller → mmueller.

On your first log-in, you should change your password, using the **passwd** command.

The first thing you can do is checkout the CVHCI-Wiki at https://cvhci.anthropomatik.kit.edu/wiki/.

You have access to a home directory for the account with a certain quota (50GB or so), the home directory is backed up regularly. Therefore you should not put big datasets there nor save your trained models into the home directory.

For datasets, you can sync with your advisor, chances are the dataset is already available in /cvhci/data/, regarding saving trained models you can save them to **/cvhci/temp/username** (e.g. /cvhci/temp/mmueller).
Regularly keep track of your folder size and e.g. do not save models of every epoch but just the latest and the best performing model for each of your experiments (rule: be reasonable and aware about disk space).

Basic commands (substitute mmueller with your username):

We have a couple GPU servers students can use, to connect to them use:

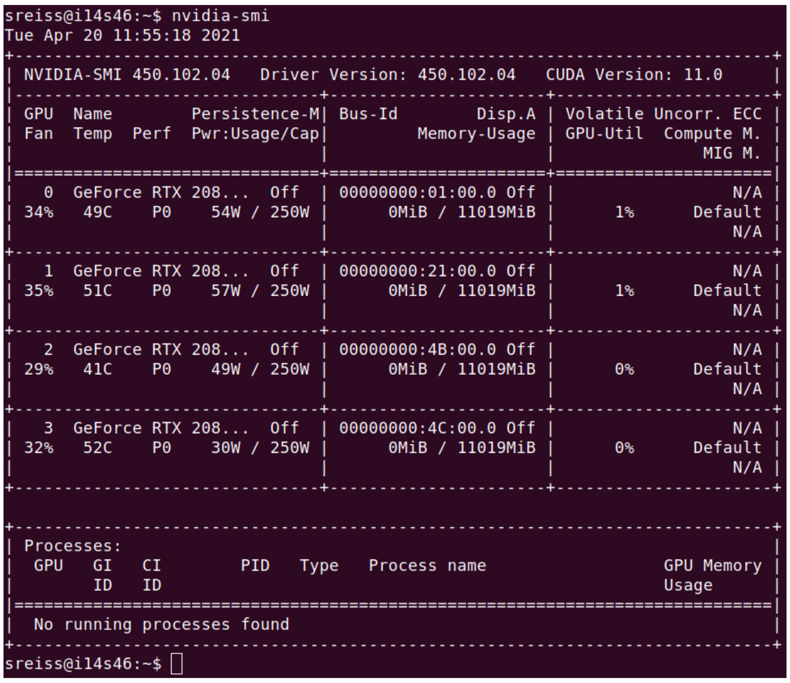
ssh mmueller@i14s34.anthropomatik.kit.edu
ssh mmueller@i14s35.anthropomatik.kit.edu
ssh mmueller@i14s36.anthropomatik.kit.edu

These are all servers that can be used by students, therefore before starting a script you need to check which GPUs are already taken.

You can check this via the command **nvidia-smi** (see screenshot).

The server in the screenshot has 4 GPUs and no processes are running.

If all GPUs on the student servers are taken, you can synchronize with your advisor whether you can have access to his or her server.



```
sreiss@i14s46:~$ nvidia-smi
Tue Apr 20 11:55:18 2021
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 450.102.04   Driver Version: 450.102.04   CUDA Version: 11.0     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  GeForce RTX 208...  Off  | 00000000:01:00.0 Off |                  N/A |
| 34%   49C    P0    54W / 250W |      0MiB / 11019MiB |      1%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   1  GeForce RTX 208...  Off  | 00000000:21:00.0 Off |                  N/A |
| 35%   51C    P0    57W / 250W |      0MiB / 11019MiB |      1%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   2  GeForce RTX 208...  Off  | 00000000:4B:00.0 Off |                  N/A |
| 29%   41C    P0    49W / 250W |      0MiB / 11019MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   3  GeForce RTX 208...  Off  | 00000000:4C:00.0 Off |                  N/A |
| 32%   52C    P0    30W / 250W |      0MiB / 11019MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
sreiss@i14s46:~$
```

The rule stays the same: If a GPU is taken, don’t start a script on it, just use free GPUs, otherwise you might crash the script of someone else.

You can specify which GPU your script (in this case train.py) uses via the command:
CUDA_VISIBLE_DEVICES=0 python3 train.py
This will run the python-script train.py on the GPU 0. After starting your script you can check whether it runs on the correct GPU via **nvidia-smi**.

Also, be aware that you are not the only one using these servers, so don’t take too many GPUs if you need **more** than 2 or 3 simultaneously, discuss with your advisor if you can use his or her server for those experiments.

To log out and keep your models training, you should use a terminal-multiplexer such as **tmux** (some people also use screen). This will enable you to (1) open multiple terminal-tabs to work in and (2) detach from the terminal (and exit the ssh connection) without stopping your jobs.

The most important commands for tmux you need are:
tmux, this will open a new tmux-session, within which you can:

- **Ctrl+b** then **c** to open a new tab
- **Ctrl+b** then **n** to switch to the next tab
- **Ctrl+b** then **&** to delete the current tab
- **Ctrl+b** then **,** to rename the current tab
- **Ctrl+b** then **d** to detach from the tmux session (to reattach just type **tmux a**, or specify the tmux-session if you have multiple running)
- More commands e.g. how to navigate in a tab can be found on tmux cheat sheets: https://duckduckgo.com/?q=tmux+cheat+sheet&ia=cheatsheet&iax=cheatsheet

What I like to do is having one tab in tmux where I run **watch nvidia-smi**, with this you can easily monitor which GPUs are already taken and which are free.

Maybe sometimes you want to use a graphical interface. To do that, you can connect to a pc in the pool-room with a command like **ssh -X mmueller@i14pc188.anthropomatik.kit.edu**
Here, pc188 is just an example, you can check the pc number of the pc you are working on in the pool-room and connect to that one (numbers are between pc184 and pc190).
If you connect with this command and type e.g. **nautilus** or **firefox** the file-viewer or web-browser will open respectively. It is quite slow, but maybe it is useful sometimes.

Of course you can use general commands like
scp mueller@i14s30.anthropomatik.kit.edu:/home/mmueller/VeryImportantImage.png [INSERT PATH ON YOUR MACHINE]
to remotely copy a file to your local machine or anything the like.

You should set up your own virtual environment for python and install your needed packages there (https://docs.python.org/3/library/venv.html).
If you decide to use venv, you can activate the virtual environment via the command:
source [PATH TO VIRTUAL ENVIRONMENT]/bin/activate

and start installing your packages with commands like **pip3 install --user packageName**.

Via your student KIT account you should have access to your own Gitlab workspace (https://git.scc.kit.edu), you can set up your project there and invite your advisor.

Startingpoints if you never used Pytorch

There are a lot of great tutorials for getting comfortable with pytorch at https://pytorch.org/tutorials/, for example getting started with vision tasks you can do the tutorial on CIFAR10 https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html.

You could also check out how custom dataloaders can be implemented https://pytorch.org/tutorials/beginner/data_loading_tutorial.html.

Tensorboard

A powerful tool to visualize everything from losses to performance metrics and sample images in training is tensorboard, I would recommend using it https://www.tensorflow.org/tensorboard/ (it is also available in the pytorch reference https://pytorch.org/docs/master/tensorboard.html#torch-utils-tensorboard). But be aware that the logging files can get quite big if you log a lot of stuff, so just keep that in mind.

You can install tensorboard to your virtual environment via pip (see above).
If your code produces a tensorboard-log file in training and logs things like loss, metrics or the learning rate you can inspect it in your browser by:

Run the command: **tensorboard logdir=[PATH TO TENSORBOARD LOG DIRECTORY]**
(or: **python3 -m tensorboard.main logdir=[PATH TO TENSORBOARD LOG DIRECTORY]**)
Then tensorboard will output a link (e.g. http://localhost:6006/) in the console which you can open in your browser to access the logged information:

```
>(venv) mmueller@i14pc185:~$ tensorboard --logdir=/cvhci/temp/mmueller/workspace/...
>TensorFlow installation not found - running with reduced feature set.
>Serving TensorBoard on localhost; to expose to the network, use a proxy or pass --bind_all
>TensorBoard 2.4.1 at http://localhost:6006/ (Press CTRL+C to quit)
```

If you work remotely, you can connect via ssh to a server and start tensorboard on the server as shown above. If tensorboard runs on the server, e.g. s34, you can then connect from your terminal via:

ssh -L 16006:127.0.0.1:6006 mmueller@i14s34.anthropomatik.kit.edu

In this example you will have access to your tensorboard logs in your local browser if you type in http://localhost:16006/ . Note you have to specify the correct ports.

Alright, that was a lot! But I hope you can establish a productive workflow with all this Info!