Klinikum rechts der Isar
Technische Universität München

Practical: Applied Deep Learning in Medicine

Chair for AI in Medicine and Healthcare

Technische Universität München

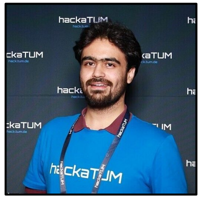
Team

- PhDs here at the chair
- Working on different aspects of AI in Medicine
- All topics are related to our own research
- Practical organisation: Sarah, Alex, Hendrik

Alex

Hendrik

Jojo


Maulik

Johannes

Dima

Sarah

Anna

Robert

Wenke

Matan

Organisational

- Meetings
  - Graded presentations:
    - 7 mins of presentation, 2-4 minutes questions from other groups
    - Open questions and feedback welcome
    - You can miss one (joker) without explanation
  - Team meetings:
    - Meeting in small groups with project supervisors
    - Format of these is agreed upon between groups and supervisors

	Date	Grade Count	Expectations
1st Presentation	15.05.25	10%	Introduction to your topic, first ideas
2nd Presentation	12.06.25	10%	First approaches, encountered issues
3rd Presentation	10.07.25	10%	Current status and results
Final Poster Session	06.08.25 (14:00-16:00)	20%	Summary of your whole project and work, answering questions

Organisational

- Submissions
  - Code
    - Please comment your code
    - Easy-to-follow and execute (requirements…), Reproducibility!
    - Good practices
      - Configuration files
      - No hard coding
      - No jupyter notebooks for pre-processing or training
  - Report
    - ~6-8 pages
    - Summarizing your work, decisions, and reasoning behind it
    - MICCAI Template, our guideline is here: link

	Date	Grade Count
Code Submission	01.08.25 11:59 CEST (noon!)	20%
Report Submission	28.08.25 11:59 CEST (noon!)	30%

Organisational

- Recommendations:
  - Keep a log of each week
    - We recommend notion for this (notion.so)
    - This will help you for final presentations
  - You have control over this project
    - We will guide you but you steer it
    - It is part of the practical that you set goals
    - In the end you must be able to reason why you made certain decisions
- Communication
  - ADLM Slack
  - Communicate within the group (this is a group project)
  - Communicate with us (don’t be afraid)

Organisational

Questions?

Compute cluster

1. General remarks
2. VPN access to Galileo and CIT
3. Connect via ssh
4. Storage
5. Run jobs with SLURM as scheduling system
   a. Example sbatch script
   b. More on sbatch
   c. Debugging within slurm

#server-cluster channel for issues/ problems

General remarks

- LDAP by cit.tum.de -> VPN with cit.tum.de credentials
- All compute resources are distributed by SLURM as scheduling system
- One user can launch up to two (non-interruptible) jobs at the same time
- Setup and get familiar until the first presentation!
- #server-cluster channel for issues/problems
- No data in home directory -> put everything under /vol/miltank/[...]

1. VPN access to Galileo and CIT

To access the workstations or servers via ssh, you need a VPN. Please follow the guidelines:
https://wiki.ito.cit.tum.de/bin/view/Informatik/Helpdesk/Vpn/

Only works with the RBG IN/MA VPN https://vpn.rbg.tum.de. A cit.tum.de account is required. If you don't know your username ask us.

Download the configuration file for our chair called il31 at https://vpn.rbg.tum.de and add

```
route 131.159.110.0 255.255.255.0 # fmi (servers)
route 131.159.128.0 255.255.255.0 # galileo (workstations)
```

Alternatively use already modified versions: vpn-il31-2.4-linux.ovpn / vpn-il31-standard.ovpn (for Windows, see known issue below).

Known Issue on Windows

Using the routes from the code block conflicts with using the Windows-specific ovpn file.

The Windows file suggests using the experimental driver "*windows-driver wintun*" which does not work with specific routes. Therefore, use the standard ovpn file or remove the line "*windows-driver wintun*" which results basically in the standard file.

Once you've downloaded the file, start the connection on Linux with

```
sudo openvpn vpn-il31-2.4-linux.ovpn
```

2. Connecting via ssh to selene (131.159.110.9)

Connect to the workstation via ssh

```
ssh <your-CIT-username>@131.159.110.9

ssh <your-CIT-username>@selene
```

Use selene only as ssh node to request resources. Do not run jobs on selene!*

*exception: creating conda envs or moving files but monitor the usage

3. Storage

Storage possibilities

There are three filesystems:

- `~/` or `/u/home/<in-tum-username>`
  - Your home directory
  - Do not use! It must remain empty.
- `/vol/miltank/users/<username>/`
  - Do not use more than 500 GB space per group
  - Check your usage with `du-sh`
  - Talk to your supervisor if you feel like you need more
- `/meta/users/<username>/`
  - This is fast storage for metadata (conda / vscode / etc.)
  - Move all of these directories from ~/ to meta storage and symlink them:

First thing to do when on the cluster:

```
mv ~/.conda /meta/users/$USER/ && ln -s /meta/users/$USER/.conda /u/home/$USER/.conda
mv ~/.cache /meta/users/$USER/ && ln -s /meta/users/$USER/.cache /u/home/$USER/.cache
mv ~/.local /meta/users/$USER/ && ln -s /meta/users/$USER/.local /u/home/$USER/.local
mv ~/.vscode-server /meta/users/$USER/ && ln -s /meta/users/$USER/.vscode-server /u/home/$USER/.vscode-server
```

Organization of storage

- Use your personal directory in `/vol/miltank/users/<userid>`
- Shared files can be located in `/vol/miltank/projects/practical_sose25`

4. Run jobs with SLURM

Usually, you submit a job through the Slurm command `sbatch` together with a small script. In this section, we demonstrate how to do this with a simple base script as seen below. A submission script consists of two parts at its core:

- **Resource requests**: Directives specifying options preceded by `#SBATCH`.
- **Job steps**: Any executable command, e.g. for setting up an environment or running a program.

Most submission scripts are fairly straight forward. On the next slide is a simple example, followed by a more detailed description.

Submitting the job

Once your sbatch script is ready, you can submit the job to Slurm by running `sbatch <script-name>`. Your job will then be *queued* until the requested resources are available. As soon as they are, your job will start to run on some compute node and do whatever you defined below the `#SBATCH` directives.

a. Example sbatch script

```
#!/bin/bash

#SBATCH --job-name=<name>
#SBATCH --output=<name>-%A.out
#SBATCH --error=<name>-%A.err
#SBATCH --mail-user=<your-email-address>
#SBATCH --mail-type=ALL
##SBATCH --partition=universe,asteroids
##SBATCH --qos=master-queuesave
#SBATCH --time =0-24:00:00
#SBATCH --gres=gpu:1
#SBATCH --cpus-per-task=8
#SBATCH --mem=16G

# Load python module
ml python/anaconda3

# Activate corresponding environment
# If you launch your script from a terminal where your environment is already loaded, conda won't activate the environment. The following guards against that. Not necessary if you
always run this script from a clean terminal
conda deactivate

# If the following does not work, try 'source activate <env-name>'
conda activate <env-name>


# Run the program
python main.py --dataset cifar10 --dataroot data --cuda
```

b. More on sbatch

The `#SBATCH` directives followed by some argument are your means to control Slurm's behaviour in regard to your specific job. They allow you to e.g. request resources as you desire or define where to write outputs to, etc. Most of these directives are quite self-explanatory. Let's still go through them here one by one.

General settings:

- job-name: Give your job a descriptive name. This will help you identify your run later when viewing a (possibly very long) list of running jobs.
- output: May be an absolute or relative path defining where to write standard output to, e.g. anything that would normally be echoed to your terminal. %A adds your job's ID to the filename such that you can launch the same script multiple times without simply overwriting previous outputs.
- error: Same as *output*, just for error messages.
- mail-user: Provide your email address if you want to receive notifications about your job's status.
- mail-type: Define for which events to send you an email. May be one of: BEGIN, END, FAIL, REQUEUE, ALL
- qos: The *quality-of-service* (QOS) to use for your submission. See the QOS section below for more details.

Resource requests:

- time: Limit on the job's total runtime with the format days-hours:minutes:seconds.
- gres: Requests so-called *generic resources* which for us really boils down to GPUs. For example, `--gres=gpu:1` tells Slurm to allocate one GPU for your job.
- cpu-per-task: Request a certain number CPU cores. Request **fewer than 32 cores** unless you **absolutely** need more.
- mem: Request a specific amount of RAM for your run. Stay **well below 128 Gb** unless you **absolutely** need more.

Preceding a line with `#` as in above script's `--qos` statement, defines it as a comment and will cause it to be ignored. There are many more options for such sbatch scripts to choose from. If you are interested in learning more, please refer to the official manual. After the last `#SBATCH` directive, you add *job steps,* i.e. actual commands to run on the cluster. In above demo script, we load an environment and then run our main python program. But really, you may have any command here, just like you would in a *normal* terminal.

Two `--qos` types: master and master-queuesave

- master: you can run as many jobs as you like, but if a higher prio job has no resources yours is stopped and started again once resources are free again
- master-queuesave: you can start two jobs at most, but once they're running they will not be interrupted

c. Debugging within slurm

A former practical student developed this handy tool: https://gitlab.lrz.de/christian.beischl/shs

This allows you with little to no overhead to connect a vscode session / jupyter notebook with a slurm job.

1. Go to selene and install the above tool. At the end of the installation script are instructions to copy for the third step.
2. In the shs directory is a user.config file. Go into that file and see if everything is set to your liking
   a. Specifically pay attention to the VSC_KEY_PATH argument. This should link to a public ssh key
   b. If this key does not yet exist generate one using ssh-keygen Then set the above argument.
3. On your local machine go to the ssh config file add the lines from the first step. They should look similar to this:

```
Host aim-tunnel
     ProxyCommand ssh selene "bash /vol/miltank/users/zillera/shs/svscode.sh"
     StrictHostKeyChecking no
     ConnectTimeout 60
     User zillera
```

Troubleshooting

- Instead of doing it in vscode just copy paste the ssh command (ssh selene "bash /vol/miltank/users/<username>/.vscode/shs/svscode.sh") and see if this starts a slurm job that keeps running. If the job appears for a second and disappears then there is an error in the settings.
- To investigate which error this could be it is very helpful to look at the slurm output. To see this you have to add a variable called VSC_OUTPUT to your user.config file and set it to an arbitrary path where the slurm output should be stored. In there you can see any error message that comes along.

5. Dos and Don’ts

- Keep resources limited to what you actually need
- Only request a GPU if you are training with a GPU
- Run preprocessing and data downloads also with jobs not on selene
- The cluster is limited and shared between the entire chair
- Do not use the home directory /u/home/ for ANYTHING
- All longer running jobs (training, preprocessing, …) should be run as a script within a slurm job.
- Do not run trainings within a jupyter notebook, use jupyter only for exploration purposes
- All data, scripts, outputs, checkpoints, … must be on /vol/miltank/…

Additional remarks

- Conda is preinstalled: `ml python/anaconda3`
- Our cluster supports the use of uv: `ml python/uv`
- Welcome messages to the cluster can be really helpful
  - Watch out! Anyone can add messages. Sometimes they are evil!
- For very fast storage you can use /vol/cuttlefish/
  - However, storage capability is limited
  - System is currently still in a test phase
- Actually there are two QoS you can use: master and master-queuesave
  - master-queuesave:
    - Limited to two jobs
    - Can not be interrupted
  - master:
    - As many jobs as you like
    - Jobs are interrupted once a higher prio job comes in.
    - If cluster is fully utilised, master jobs are not scheduled at all

Compute cluster

Questions?

Recommendations

- Projects are very diverse
- Some recommendations might not be applicable to your project
- But a lot will be
- (If unsure, talk to your supervisor)

How to do Research

1. Literature Review: Find approaches for your data
2. Goal Setting: Decide on your project goals
3. Exploratory data analysis: Understand your data
4. Outline the project: Write an abstract / diagram. Set a timeline.

Presentation 1 (after 3 weeks)

1. Implement a simple baseline model
2. Analyze potential issues

Presentation 2 (4 weeks)

1. Implement, iterate, evaluate, iterate

Presentation 3 (4 weeks)

Poster Presentation (4 weeks)

Recommendations: Literature Review

- Goal:
  - You must **understand the field** to tackle your project
  - What problem are you trying to solve? What is state of the art to solve that problem?
  - What is missing from current solutions? How can you add something of **value** to our current understanding of the problem?
  - What unique angle do you have? How can you best leverage the data you have?
- Where to find Papers
  - https://scholar.google.de/
  - Cited papers or related works (https://www.connectedpapers.com/)
  - https://pubmed.ncbi.nlm.nih.gov/
  - Note: Focus on more recent works. The field moves fast.
- How to skim papers
  - Read titles and abstracts of many papers and identify interesting ones
  - Skim important parts (e.g. figures, methods, results) of interesting papers
  - Only read very few papers in detail
- Manage your literature
  - Using notes (notion.so, G-Docs) or using tools like Mendeley, EndNote, …

Recommendations: Goal Setting

1. Come up with potential goals
   - Understand the literature and then come up with your own ideas
   - Supervisors can give you guidance/directions
   - Consider which tasks would be possible and **interesting** given your data
   - Good science **creates value**
2. Discuss with us, we give feedback
   - Realistic within the given project scope
   - Relevant from a medical perspective (we will involve medical experts to answer this question)
   - Interesting / relevant to the scientific community
3. How can goals be evaluated
   - How can we measure the results (do we have a ground-truth)?
   - What can be used as baselines?
   - What are appropriate metrics?
   - What would demonstrate success i.e. new knowledge gained?
     - **Success != high accuracy**

Recommendations: Exploratory data analysis

- Understand dataset structure
  - Patients and sample structure
  - Multiple modalities
  - Missing information
  - Image level splits / patient level splits
- What information do we have
  - Class labels?
  - Metadata?
- Compute and plot dataset statistics
  - Number of samples (per patient, per modality, …)
  - Label distribution - balanced or unbalanced
  - (Makes great figures for presentation 1)
- Understand the modalities
  - Load and view some images / sequences / graphs / text
  - **Look at your data!**
  - Analyse properties of modalities (image sizes / sequence length / graph properties / …)

Recommendations I

Questions?

Recommendations: Simple Baseline Model

- Implement a simple model
  - Check the whole training pipeline: pre-processing, dataloading, …
- Start very simple
  - Use existing methods without major modifications
  - Use libraries. Do not reinvent the wheel. You will make mistakes.
  - Keep the implementation as simple as possible. Follow the original paper.
  - This simple model can later be used as a baseline
- Extend the model incrementally
  - One change at a time
  - Allows you to identify which changes actually improve the results
  - Integrate ideas from literature
  - Implement more complex algorithms or integrate more modalities
  - Helps to understand if the issue is the data or the algorithm
- Start with small subsets of the training data
  - Allows you to do hyperparameter tuning more efficiently
  - Try which subset sizes still give reasonable results
  - Later you can train the best models on the full dataset

Recommendations: How to organize and keep reproducibility

- Track your results
  - E.g. using tensorboard or wandb (be careful of data protection)
  - Also save the hyperparameters/configs of runs
  - Keep (best) model checkpoints for plotting results later
- Keep shared notes of ideas, discussions, tasks, experiments
  - We recommend Notion
- Create a roadmap
  - What are the milestones in your project?
  - When should the milestones be done?
  - Make sure you are hitting your milestones and if not, adjust your approach
- Define and assign tasks
  - Keep team TODO-lists
  - Split tasks across team members
  - Some tasks (like problem analysis or discussion of ideas) are often better done by the whole team
  - But… don’t have 3 people working on one thing

Recommendations: Coding practices

- Use a project repository
  - E.g. https://gitlab.lrz.de/
  - Commit regularly (never start training without a committed version)
- Use an IDE
  - We recommend **VSCode** or PyCharm
- Jupyter Notebooks are good for **explorative** tasks
- Python Scripts are good for training and data-processing
  - **DO NOT** use Jupyter notebooks for long-running scripts (i.e. training, multiple pre-processing steps, etc.)
  - Use command-line arguments (argparse, click) or configs (Hydra, …)
- Keep it simple
  - You are writing research code not production code

Recommendations: Presentation

- First slide - Names, Supervisor, Project
  - It’s often helpful to introduce yourselves briefly
- Second slide - Brief project summary
  - What is your project about? What is the goal?
- Third slide - Goals of the Presentation (Agenda)
  - Tell us what you want to accomplish with this presentation. Put the audience in the right frame of mind.
  - Are you presenting new findings or looking for advice? What are the key takeaways?
  - Don’t use a useless, time consuming “table of contents” slide
  - Try and stay away from generic points like “Intro” and “Results”.
- Rest - Update on what you did
  - What worked. Tips for others and sanity check by us
  - What didn’t work. Where do you need input / help
  - What you are planning on doing next
  - Be precise and give details even if you don’t talk about it (e.g. augmentations)

Recommendations: Project Management

- Keep your goal in mind
- How did you define a successful project at the start? Try and spend as much time as possible on the most impactful section
- Try not to spend two months optimizing hyperparameters of the baseline
- You want to create something of **value**
- Start early on novel ideas once the **simple** baseline works
- Effectively split the work to optimize your time; the semester always goes faster than you think

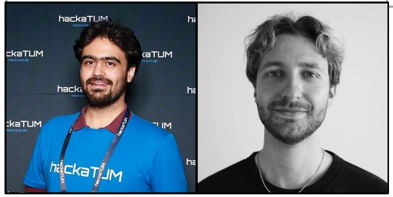
Recommendations: Group - Supervisor Relationship

- Our expertise is research. We can help you improve your scientific thinking, project conception, and critical analysis.
- Debugging problems: start by consulting stack overflow, ChatGPT, and teammates
- In the group meetings:
  - Summarize what worked, what did not work, and where you are blocked.
  - Presentations are great for this (ugly is fine). Please come prepared.
  - Please do not start searching for tensorboard runs during the meeting.
- Especially if you are unsure or encounter issues, **please be open and honest.**
- If you are stuck, don't be afraid to reach out (instead of waiting for next weeks meeting)

Recommendations II

Questions?

Team Matching

Evaluating Pretraining strategies for Lung Cancer Risk Prediction (Maulik, Johannes) 	Multi-Modal Deep Learning for Predicting Oncology Outcomes (Anna) 	Analysis of Resolution Differences on a Segmentation Task (Hendrik) 	Atlas-Based Registration for Torso and Legs (Robert)
Rotational Equivariance of Medical Foundation Models (Jojo) 	Text-Guided Medical Image Editing with Cross-Attention Control (Wenke) 	Multimodal Learning for Vertebral Tumor Classification Using CT and MRI (Matan) 	In-context Learning for image-derived features in the medical domain. (Dima)