A Technical Overview

Romain Hermary

Friday 16th June, 2023

UNIVERSITÉ DU LUXEMBOURG

SnT

1.

Visual Studio Code

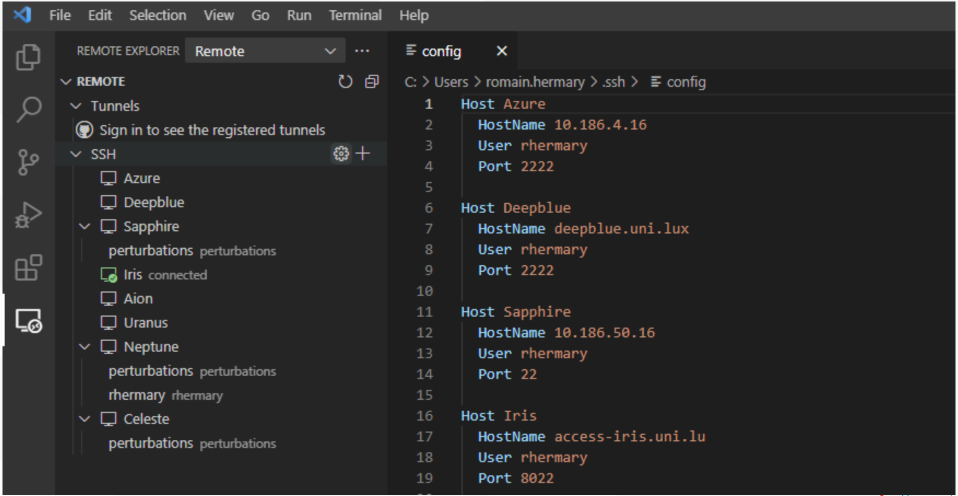
What Do I Use It For?

Everything

- Connect to a remote computer/server (and **WSL** on **Windows**)
- Open multiple terminals on the remote
- Automatize environment launching (can be different for each projects)
- Intuitive built-in debugger
- Efficient use of version control systems (**Git**)
- Very fast code navigation
- Awesome extensions
- ...

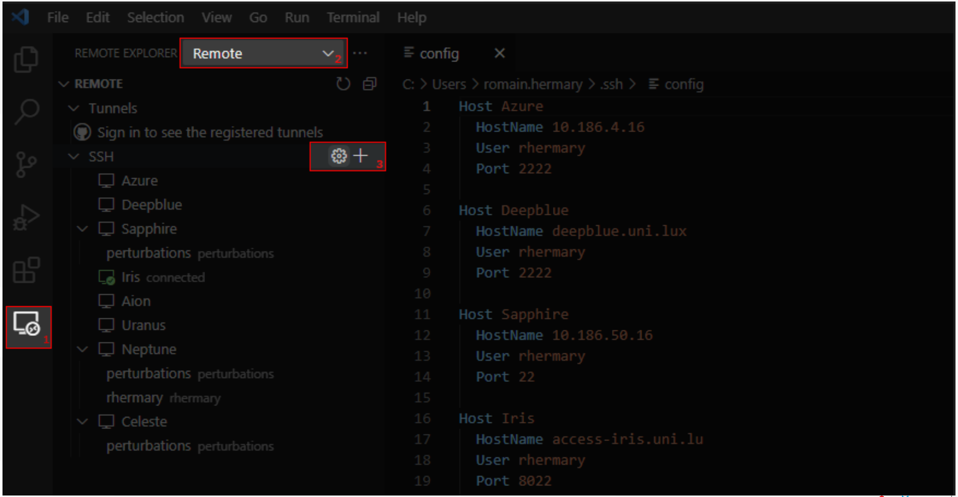
Note: **VSCode** is actually a nice text editor, not an *IDE*, ideal for development, debugging, etc.

SSH With VSCode


File
Edit
Selection
View
Go
Run
Terminal
Help
REMOTE EXPLORER
Remote
REMOTE
Tunnels
Sign in to see the registered tunnels
SSH
Azure
Deepblue
Sapphire
perturbations perturbations
Iris connected
Aion
Uranus
Neptune
perturbations perturbations
rhermary rhermary
Celeste
perturbations perturbations
config
C: > Users > romain.hermary > .ssh > config

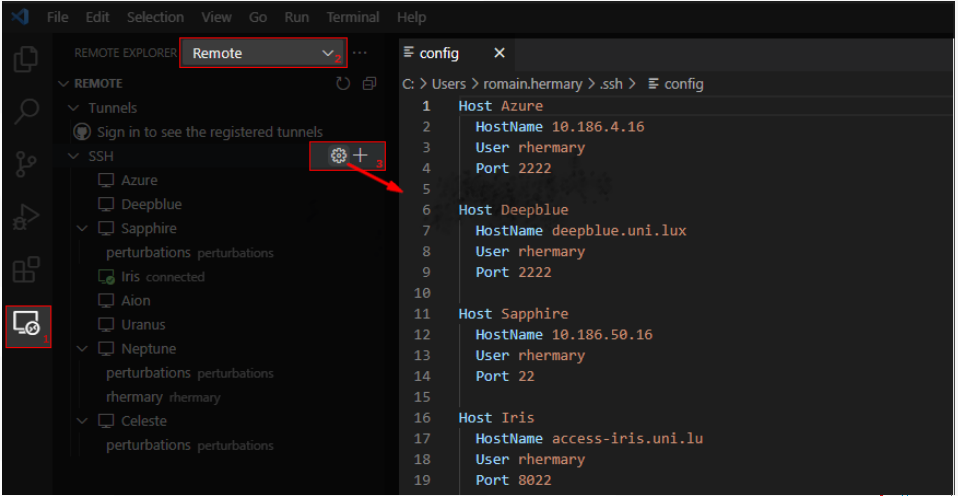
```
Host Azure
  HostName 10.186.4.16
  User rhermary
  Port 2222

Host Deepblue
  HostName deepblue.uni.lux
  User rhermary
  Port 2222

Host Sapphire
  HostName 10.186.50.16
  User rhermary
  Port 22

Host Iris
  HostName access-iris.uni.lu
  User rhermary
  Port 8022
```

SSH With VSCode


File
Edit
Selection
View
Go
Run
Terminal
Help
REMOTE EXPLORER
Remote
2
REMOTE
Tunnels
Sign in to see the registered tunnels
SSH
3
Azure
Deepblue
Sapphire
perturbations perturbations
Iris connected
Aion
Uranus
Neptune
perturbations perturbations
rhermary rhermary
Celeste
perturbations perturbations
1
config
C: > Users > romain.hermary > .ssh > config

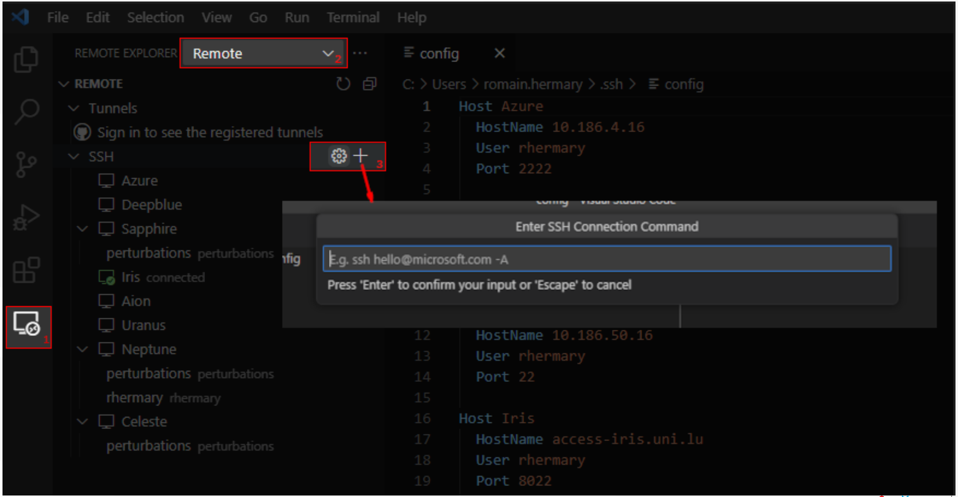
```
Host Azure
  HostName 10.186.4.16
  User rhermary
  Port 2222

Host Deepblue
  HostName deepblue.uni.lux
  User rhermary
  Port 2222

Host Sapphire
  HostName 10.186.50.16
  User rhermary
  Port 22

Host Iris
  HostName access-iris.uni.lu
  User rhermary
  Port 8022
```

SSH With VSCode


File
Edit
Selection
View
Go
Run
Terminal
Help
REMOTE EXPLORER
Remote
2
REMOTE
Tunnels
Sign in to see the registered tunnels
SSH
3
Azure
Deepblue
Sapphire
perturbations perturbations
Iris connected
Aion
Uranus
Neptune
perturbations perturbations
rhermary rhermary
Celeste
perturbations perturbations
1
config
C: > Users > romain.hermary > .ssh > config

```
Host Azure
  HostName 10.186.4.16
  User rhermary
  Port 2222

Host Deepblue
  HostName deepblue.uni.lux
  User rhermary
  Port 2222

Host Sapphire
  HostName 10.186.50.16
  User rhermary
  Port 22

Host Iris
  HostName access-iris.uni.lu
  User rhermary
  Port 8022
```

SSH With VSCode


File Edit Selection View Go Run Terminal Help
REMOTE EXPLORER
Remote
2
config
C: > Users > romain.hermary > .ssh > config
REMOTE
Tunnels
Sign in to see the registered tunnels
SSH
3
Azure
Deepblue
Sapphire
perturbations perturbations
Iris connected
Aion
Uranus
Neptune
perturbations perturbations
rhermary rhermary
Celeste
perturbations perturbations
1
config - Visual Studio Code

```
Host Azure
  HostName 10.186.4.16
  User rhermary
  Port 2222

  HostName 10.186.50.16
  User rhermary
  Port 22

Host Iris
  HostName access-iris.uni.lu
  User rhermary
  Port 8022
```

Enter SSH Connection Command

E.g. ssh hello@microsoft.com -A

Press 'Enter' to confirm your input or 'Escape' to cancel

SSH With VSCode

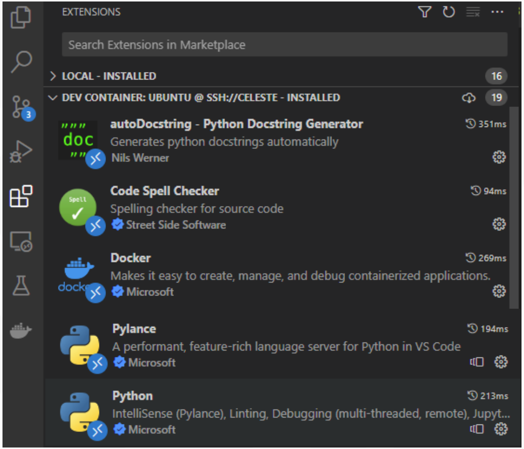
Tips:

- You might need to specify a private-key file or start an agent (more *here*)

```
Host name-of-ssh-host-here
  User your-user-name-on-host
  HostName host-fqdn-or-ip-goes-here
  IdentityFile ~/.ssh/id_ed25519
```

- I personally have:
  - A vault (with **1Password**), to manage/generate my keys – *and passwords!*
  - The extension for **VSCode** to automatically use and forward these keys

Market Place & Extensions



EXTENSIONS

Search Extensions in Marketplace

LOCAL - INSTALLED 16

DEV CONTAINER: UBUNTU @ SSH://CELESTE - INSTALLED 19

autoDocstring - Python Docstring Generator 351ms
Generates python docstrings automatically
Nils Werner

Code Spell Checker 94ms
Spelling checker for source code
Street Side Software

Docker 269ms
Makes it easy to create, manage, and debug containerized applications.
Microsoft

Pylance 194ms
A performant, feature-rich language server for Python in VS Code
Microsoft

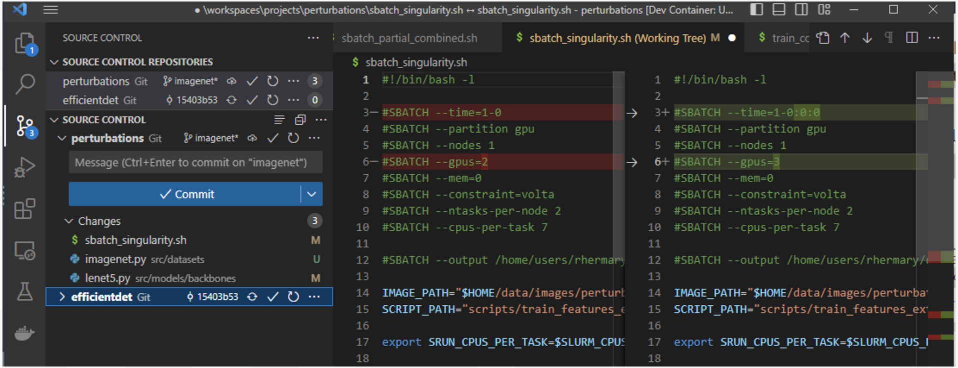
Python 213ms
IntelliSense (Pylance), Linting, Debugging (multi-threaded, remote), Jupyt...
Microsoft

Find your happiness:

- Spell checker
- Language specific highlighting/helpers
- Visualization/preview tools
- Automating tools
- API bridges (Azure, SQL databases, Rest Client, etc.)
- Vim
- ...

See all available extensions on the application or on the dedicated *web-page*

Git Without CLI

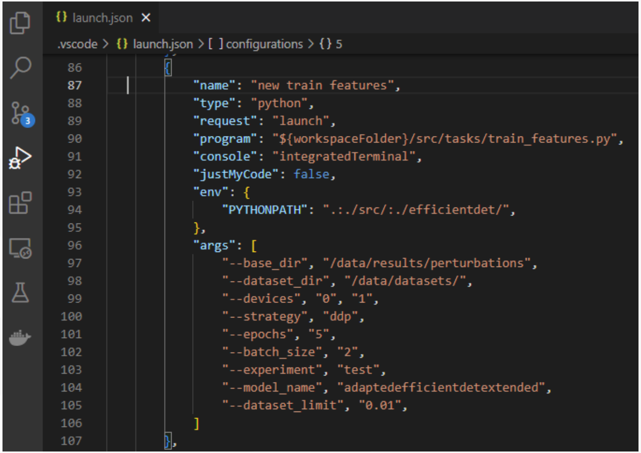

• \workspaces\projects\perturbations\sbatch_singularity.sh ↔ sbatch_singularity.sh - perturbations [Dev Container: U...
SOURCE CONTROL
SOURCE CONTROL REPOSITORIES
perturbations Git imagenet* 3
efficientdet Git 15403b53 0
SOURCE CONTROL
perturbations Git imagenet*
Message (Ctrl+Enter to commit on "imagenet")
✓ Commit
Changes 3
sbatch_singularity.sh M
imagenet.py src/datasets U
lenet5.py src/models/backbones M
efficientdet Git 15403b53
sbatch_partial_combined.sh
sbatch_singularity.sh (Working Tree) M
train_cc
sbatch_singularity.sh

```
#!/bin/bash -l

#SBATCH --time=1-0
#SBATCH --partition gpu
#SBATCH --nodes 1
#SBATCH --gpus=2
#SBATCH --mem=0
#SBATCH --constraint=volta
#SBATCH --ntasks-per-node 2
#SBATCH --cpus-per-task 7

#SBATCH --output /home/users/rhermary

IMAGE_PATH="$HOME/data/images/perturb
SCRIPT_PATH="scripts/train_features_e

export SRUN_CPUS_PER_TASK=$SLURM_CPUS

```

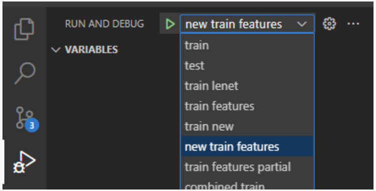
```
#!/bin/bash -l

#SBATCH --time=1-0:0:0
#SBATCH --partition gpu
#SBATCH --nodes 1
#SBATCH --gpus=3
#SBATCH --mem=0
#SBATCH --constraint=volta
#SBATCH --ntasks-per-node 2
#SBATCH --cpus-per-task 7

#SBATCH --output /home/users/rhermary/

IMAGE_PATH="$HOME/data/images/perturba
SCRIPT_PATH="scripts/train_features_ex

export SRUN_CPUS_PER_TASK=$SLURM_CPUS_

```

Disclaimer: Knowing how to use **Git** and other classic tools via the command line is still very important, do not rely solely on the GUI interface

The Debugger



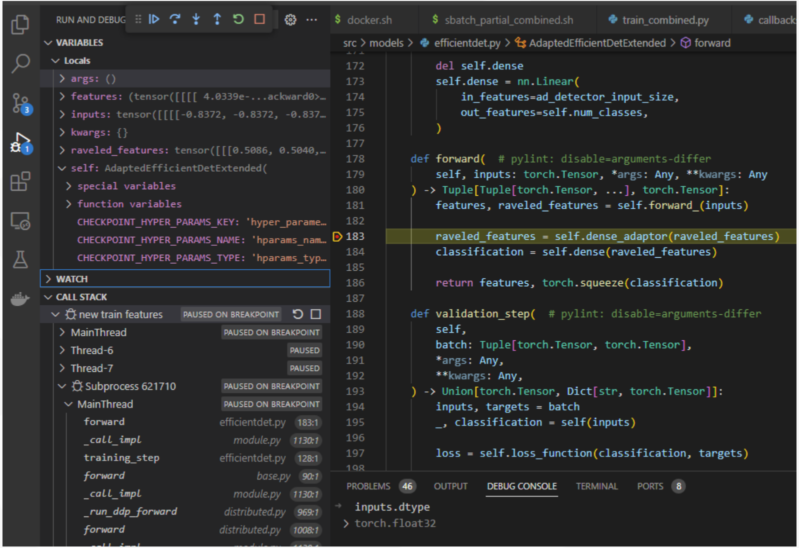
```
{
    "name": "new train features",
    "type": "python",
    "request": "launch",
    "program": "${workspaceFolder}/src/tasks/train_features.py",
    "console": "integratedTerminal",
    "justMyCode": false,
    "env": {
        "PYTHONPATH": ".:./src/:./efficientdet/",
    },
    "args": [
        "--base_dir", "/data/results/perturbations",
        "--dataset_dir", "/data/datasets/",
        "--devices", "0", "1",
        "--strategy", "ddp",
        "--epochs", "5",
        "--batch_size", "2",
        "--experiment", "test",
        "--model_name", "adaptedefficientdetextended",
        "--dataset_limit", "0.01",
    ]
},
```



1. Go on the debug tab, and create a `lauch.json` file
2. Create a new command setting
3. Select the experiment and start the debug session

More debug tips on *this* page

The Debugger



```
> args: ()
> features: (tensor([[[[ 4.0339e-...ackward0>…
> inputs: tensor([[[[-0.8372, -0.8372, -0.837…
> kwargs: {}
> raveled_features: tensor([[[0.5086, 0.5040,…
v self: AdaptedEfficientDetExtended(
  > special variables
  > function variables
    CHECKPOINT_HYPER_PARAMS_KEY: 'hyper_parame…
    CHECKPOINT_HYPER_PARAMS_NAME: 'hparams_nam…
    CHECKPOINT_HYPER_PARAMS_TYPE: 'hparams_typ…
```

```
v new train features          PAUSED ON BREAKPOINT
  > MainThread                PAUSED ON BREAKPOINT
  > Thread-6                  PAUSED
  > Thread-7                  PAUSED
  v Subprocess 621710         PAUSED ON BREAKPOINT
    v MainThread              PAUSED ON BREAKPOINT
        forward               efficientdet.py   183:1
        _call_impl            module.py         1130:1
        training_step         efficientdet.py   128:1
        forward               base.py           90:1
        _call_impl            module.py         1130:1
        _run_ddp_forward      distributed.py    969:1
        forward               distributed.py    1008:1
```

```python
        del self.dense
        self.dense = nn.Linear(
            in_features=ad_detector_input_size,
            out_features=self.num_classes,
        )

    def forward(  # pylint: disable=arguments-differ
        self, inputs: torch.Tensor, *args: Any, **kwargs: Any
    ) -> Tuple[Tuple[torch.Tensor, ...], torch.Tensor]:
        features, raveled_features = self.forward_(inputs)

        raveled_features = self.dense_adaptor(raveled_features)
        classification = self.dense(raveled_features)

        return features, torch.squeeze(classification)

    def validation_step(  # pylint: disable=arguments-differ
        self,
        batch: Tuple[torch.Tensor, torch.Tensor],
        *args: Any,
        **kwargs: Any,
    ) -> Union[torch.Tensor, Dict[str, torch.Tensor]]:
        inputs, targets = batch
        _, classification = self(inputs)

        loss = self.loss_function(classification, targets)
```

```
inputs.dtype
> torch.float32
```

- No more prints
- Stop where you want (`breakpoints`)
- Go backward in the call stack
- Test fixes
- View variables states
- Faster to use than `pdb` directly

Common Shortcuts

- `Ctrl` + `` ` `` : Open the terminal panel
- `Ctrl` + `Shift` + `P` : Open the command search bar (palette)
- `F5` : Launch the selected debug command (or relaunch the last one)
- `Ctrl` + `F5` : Launch the debug command without stopping at breakpoints
- `Ctrl` + `Left Click` : Go at the definition of the hovered object

Find more on the dedicated *web page*, or directly on **VSCode** by searching `Preferences: Open Keyboard Shortcuts` in the command palette

Useful Links

Specific Tutorials:

- **Python** Tips
- **Python** Environments
- **Jupyter** Notebooks debug mode

General Knowledge:

- *Complete documentation*
- Allowing breakpoints in external libraries (*disabling* `JustMyCode`)
- *Code Navigation*
- Use an `ssh-agent` and forward it (*avoid* creating a key/copypasting it on each server)
- **Intellisense**

NB: All the points above have clickable links!

2. ULHPC

Infrastructures

Aion

- `access-aion.uni.lu`
- 354 CPU nodes
  - $256GB$ of **RAM** per node
  - 128 cores per node

Iris

- `access-iris.uni.lu`
- 24 GPU nodes ($756GB$ of ***RAM***, 28 cores)
  - 18 nodes with $16GB$ GPUs
  - 6 nodes with $32GB$ GPUs
- 172 CPU nodes
  - 168 with $168GB$ of **RAM** & 28 cores
  - 4 with $3072GB$ of **RAM** & 112 cores

Benefit from $500GB$ personal homes directories (backed-up), shared projects folder of multiple terabytes[1] (access the *same* files/folders via the two different servers).

- The full documentation on how to use the clusters: `https://hpc-docs.uni.lu/`
- Very complete tutorials from the university on *everything*: `https://ulhpc-tutorials.readthedocs.io/`

[1] `https://hpc-docs.uni.lu/data/layout/`

Resource Managment with SLURM

slurm

workload manager

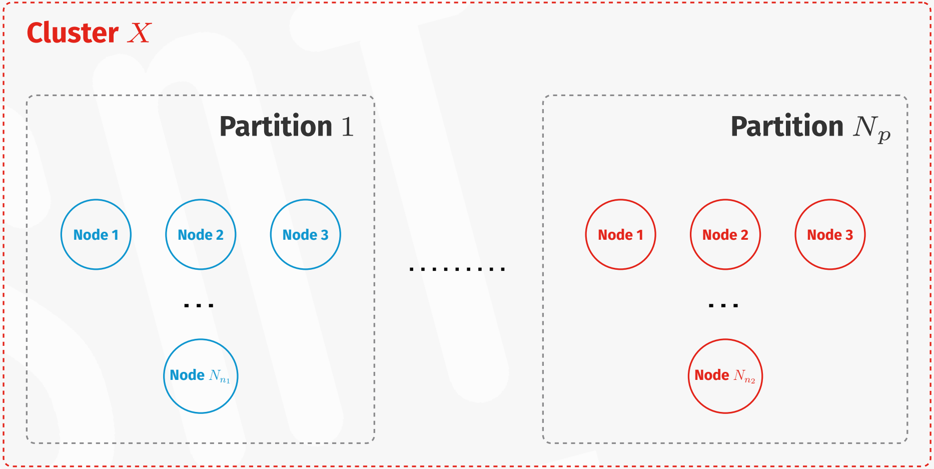
"Open source, fault-tolerant, and highly scalable cluster management and job scheduling system"[2]

- Software allowing resource management
- Interacting with this process is mandatory to access resources
- A resource request is called a *job*
- **SLURM** manages a queue of jobs, prioritise and schedule them

[2] https://ulhpc-tutorials.readthedocs.io/en/latest/basic/scheduling/

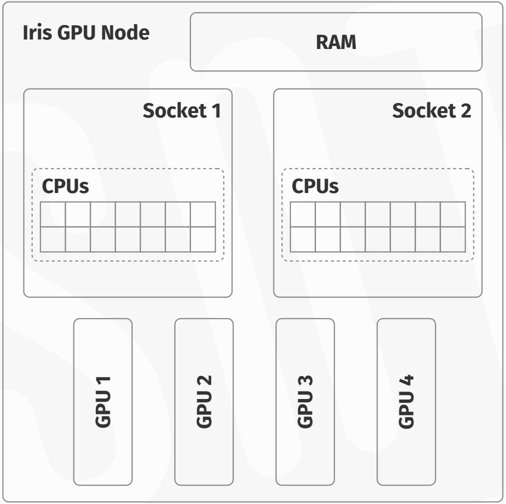
- More on **SLURM** on ULHPC in the *documentation*

Clusters



Cluster X
Partition 1
Node 1
Node 2
Node 3
...
Node N_{n_1}
....
Partition N_p
Node 1
Node 2
Node 3
...
Node N_{n_2}

Nodes Structure: An Example



When using a node, you can ask for a partial amount of the resources

- *e.g.* 14 CPUs, 2 GPUs and $100\,GB$ of RAM for 1 job

When demanding resources, try to ask for coherent amounts

- Use only what you need (beneficial for everyone)
- Adapt the number of threads and the settings to match the physical characteristics of the nodes[a]

[a] You can have a look at **NUMA** characteristics

Basic Commands

- `si` , `si-gpu`

  Custom commands which start an interactive session (allocate resource and attach your terminal to a node)[3]
- `srun`

  Used to submit a job for execution or initiate job steps in real time.

  -
    ```
    $ srun -p <partition> -c <nb_cpus> -G <nb_gpus> <command>
    ```
- `sbatch`

  Used to submit a job script for later execution. The script will typically contain one or more srun commands to launch parallel tasks..

  -
    ```
    $ sbatch <script_file>.sh
    ```

[3] More on interactive jobs *here*
More on the command options *here*

EasyBuild

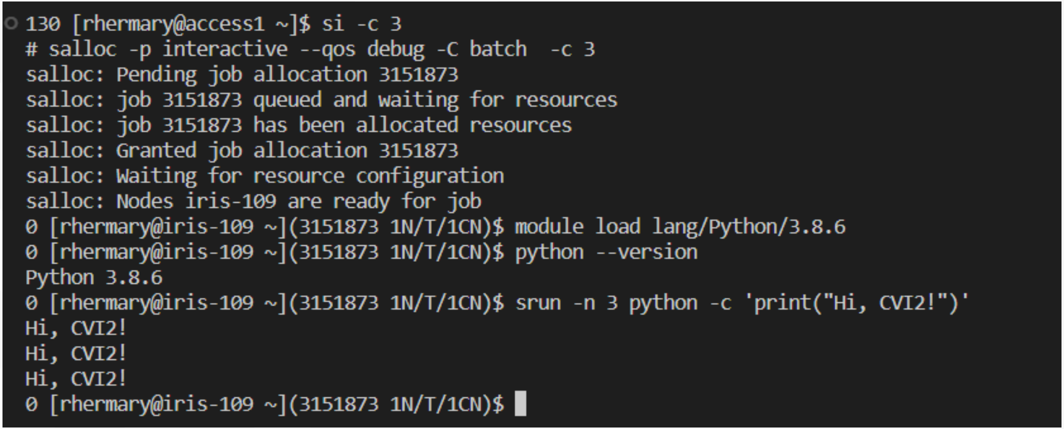
"EasyBuild is a software build and installation framework that allows you to manage software on HPC systems in an efficient way."

- Apps/packages are available as loadable modules
- A good amount of software is already available on servers
- You can build your own modules or request them
- ***Spack** is an alternative*

- All already available modules: `https://hpc-docs.uni.lu/`

- Full **EasyBuild** documentation: `https://docs.easybuild.io/`

Interactive Example



```
130 [rhermary@access1 ~]$ si -c 3
# salloc -p interactive --qos debug -C batch  -c 3
salloc: Pending job allocation 3151873
salloc: job 3151873 queued and waiting for resources
salloc: job 3151873 has been allocated resources
salloc: Granted job allocation 3151873
salloc: Waiting for resource configuration
salloc: Nodes iris-109 are ready for job
0 [rhermary@iris-109 ~](3151873 1N/T/1CN)$ module load lang/Python/3.8.6
0 [rhermary@iris-109 ~](3151873 1N/T/1CN)$ python --version
Python 3.8.6
0 [rhermary@iris-109 ~](3151873 1N/T/1CN)$ srun -n 3 python -c 'print("Hi, CVI2!")'
Hi, CVI2!
Hi, CVI2!
Hi, CVI2!
0 [rhermary@iris-109 ~](3151873 1N/T/1CN)$ 
```

Sbatch Script Example

```
#!/bin/bash -l

#SBATCH -N 2
#SBATCH --ntasks-per-node=1
#SBATCH -c 1
#SBATCH --time=0-00:05:00
#SBATCH -p batch

module purge
module load lang/Python

srun -n ${SLURM_NTASKS} python -c 'print("Hello World!")'
```

3.

Docker

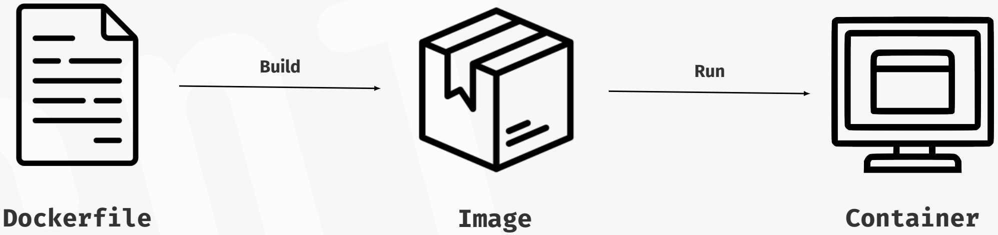
What is Docker?

"Docker is an open source platform that enables developers to build deploy, run, update and manage containers"[4]

- **Docker** packages software into standardized units that have everything the software needs to run (including libraries, system tools, code, and runtime)
- It provides a standard way of running your code
- It allows to quickly deploy and scale applications into any environment and know your code will run

[4] ibm.com

Using Docker



Build

Run

`Dockerfile`

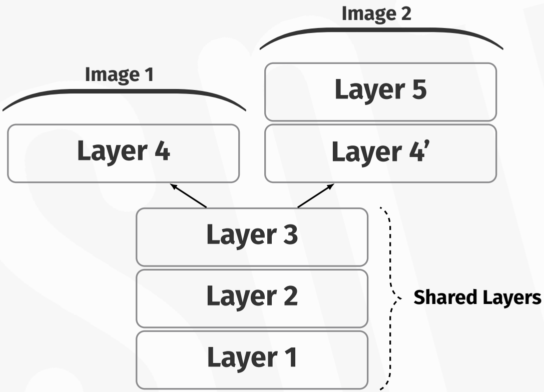
`Image`

`Container`

1. Describe your environment in a `Dockerfile`
2. Construct your **Docker** image – `$ docker build ...`
3. Launch a process with the image as a template – `$ docker run ...`

Images Sources: `iconpacks.net`, `pngwing.com`, `onlinewebfonts.com`

Some More About Images



Why are they cool?

- A layer is equivalent to a change recorded on the base image/files
- Sharing layers allows
  - Faster image building
  - Lower memory impact
- Images are not copied when launching a container![5]

[5] Rather, a *copy-on-write* layer is *created* – copy happens only when a change on shared data happens.

Why Docker? *(In Short)*

Mainly, consistency

- A more complete and stable environment than **Conda**
- Having the same environment everywhere (on different servers, different computers…)
- Sharing the environment with others (easy configuration sharing when working in teams, ship it with your article)

But also

- Possibility to modify your code while your are running experiments inside a container – *a downside of interpreted languages*
- One image for multiple users reduces the memory footprint
- `root` permissions[6], not to depend only on preinstalled packages

[6] Careful, use them *wisely*

Good Practices

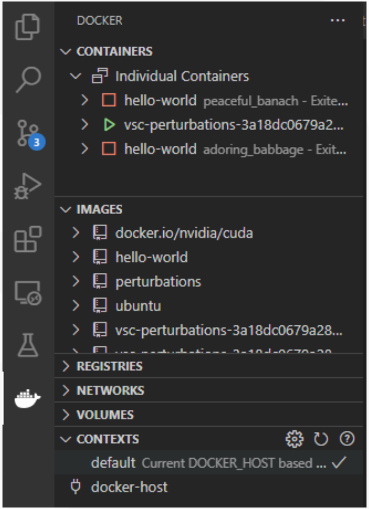
- **Tag** or **label** your images with your name
- Try to **organize** *optimally*[7][8] your building steps
  - Minimize the number of layers
  - While maximizing the number of shareable layers between different images
- Containers should be
  - Stopped when not using it
  - Removed when stopped[9]
- **Nothing** should be stored in the containers (use mount points)

[7] Some advice in *this* post

[8] More best practices for writing your `Dockerfile` in the *documentation*

[9] I would make an exception for *DevContainers*

Combining Docker & VSCode



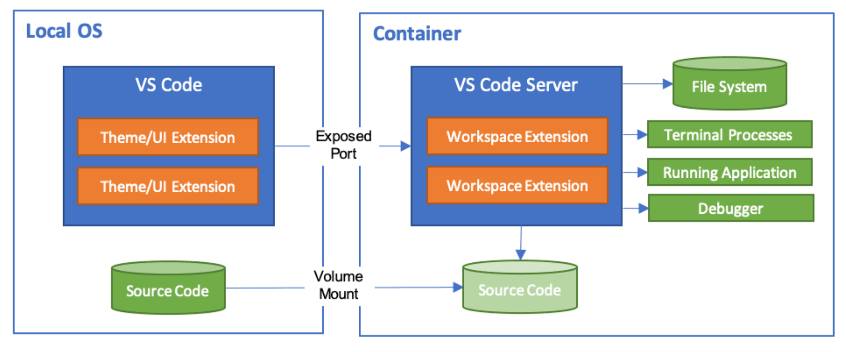
DOCKER ...

- CONTAINERS
  - Individual Containers
    - hello-world peaceful_banach - Exite...
    - vsc-perturbations-3a18dc0679a2...
    - hello-world adoring_babbage - Exit...
- IMAGES
  - docker.io/nvidia/cuda
  - hello-world
  - perturbations
  - ubuntu
  - vsc-perturbations-3a18dc0679a28...
- REGISTRIES
- NETWORKS
- VOLUMES
- CONTEXTS
  - default Current DOCKER_HOST based ... ✓
  - docker-host

- **Docker** extension is available on **VSCode**
- Nicely visualize:
  - Containers (and their states)
  - List locally available images
  - List available volumes, networks, contexts, registries...
  - *Build, deploy*

Combining Docker & VSCode: Dev Containers

What about developing inside a container?



Source: code.visualstudio.com

The extension is from **Microsoft**, and can be found in the *marketplace*

Using the Dev Containers

Automatic configuration setup

```
/project-folder
├── .devcontainer
│     └── devcontainer.json
└── [...]  # Your project files
```

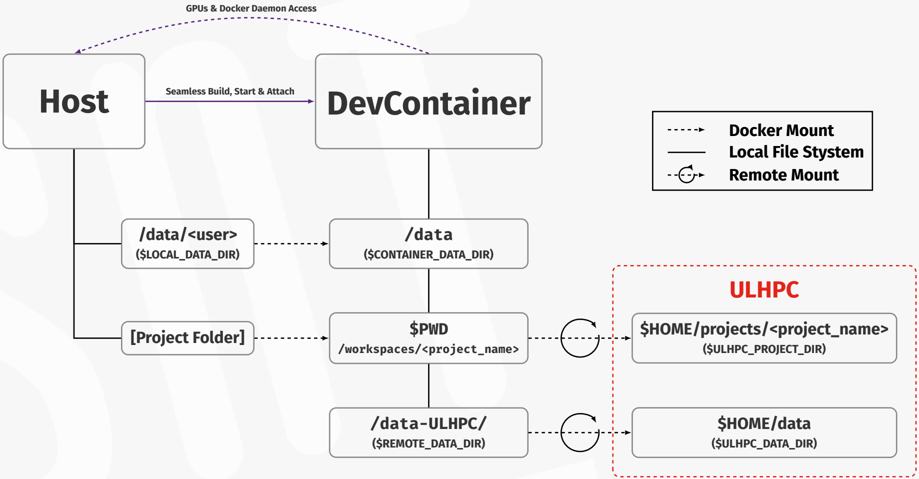
- Only adding a folder with a configuration file
- When opening your project folder, **VSCode** will detect this folder; you just need to *reopen* it in a container

My custom approach

```
/git-repo
├── project-folder
│   └── .devcontainer
│       ├── devcontainer.json
│       ├── Dockerfile
│       ├── .env
│       ├── postCreateCommand.sh
│       └── postStartCommand.sh
└── another-project
    └── .devcontainer
        └── [...]
```

But, where are my project files?

Combining Docker, VSCode & ULHPC Services



Find the template here: `https://gitlab.uni.lu/rhermary/workflow-template`

4.

Singularity

Yet Another Container Runtime

"Designed for ease-of-use on shared multi-user systems and in HPC environments"[10]

- Designed to be simple, fast, and secure
- Little memory/CPU overhead
- Fast instantiation and tear-down
- Compatible with **Docker** images
- Root elevation is needed for building
- Images are immutable, you can write only on mounted folders
- Default mounts: `$HOME` , `/tmp` , `/proc` , `/sys` , `/dev` , `$PWD`

[10] `https://hpc.auburn.edu/hpc/docs/hpcdocs/build/html/easley/containers.html`
Tutorial: `https://ulhpc-tutorials.readthedocs.io/en/latest/containers/singularity/`
Official Webpage: `https://sylabs.io/singularity/`

Some Commands

Build an image from a definition file

```
$ sudo singularity build image.sif Singularity.def
```

Pull and exec a command in a container

```
$ singularity pull docker://python:3.8.0b1-alpine3.9
$ singularity exec python_3.8.0b1-alpine3.9.sif python3
$ singularity shell python_3.8.0b1-alpine3.9.sif
```

Build an image from a compressed Docker image

```
$ sudo singularity build <image_name>.sif docker-archive://<archive_name>.tar
```

Run with GPU accessibility

```
$ singularity run --nv image.sif
```

Load Singularity module

```
$ module load tools/Singularity
```

5.

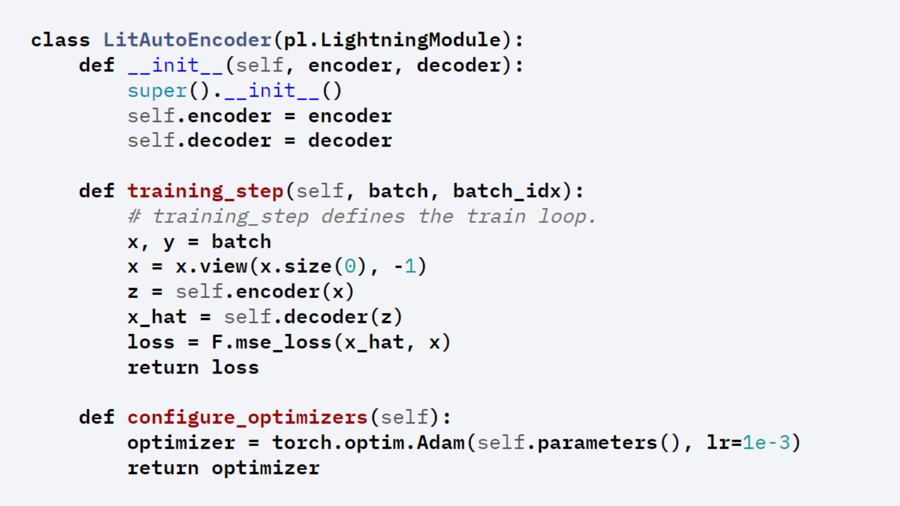
Miscellaneous

Libraries

- ***PyTorch Lightning*** – ***PyTorch*** *research framework*
  - Website – `https://www.pytorchlightning.ai/`
  - Documentation – `https://lightning.ai/docs/...`
  - Source Code – `https://github.com/Lightning-AI/...`
  - Colab Example – `https://colab.research.google.com/...`

NB: The displayed links might be shortened; click on them!

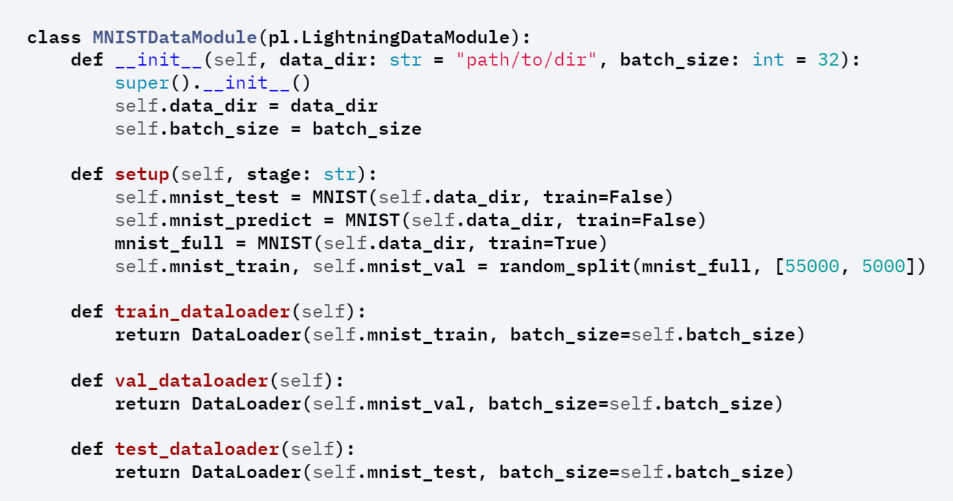
Libraries: Pytorch Lightning (Module)



```python
class LitAutoEncoder(pl.LightningModule):
    def __init__(self, encoder, decoder):
        super().__init__()
        self.encoder = encoder
        self.decoder = decoder

    def training_step(self, batch, batch_idx):
        # training_step defines the train loop.
        x, y = batch
        x = x.view(x.size(0), -1)
        z = self.encoder(x)
        x_hat = self.decoder(z)
        loss = F.mse_loss(x_hat, x)
        return loss

    def configure_optimizers(self):
        optimizer = torch.optim.Adam(self.parameters(), lr=1e-3)
        return optimizer
```

Source: lightning.ai

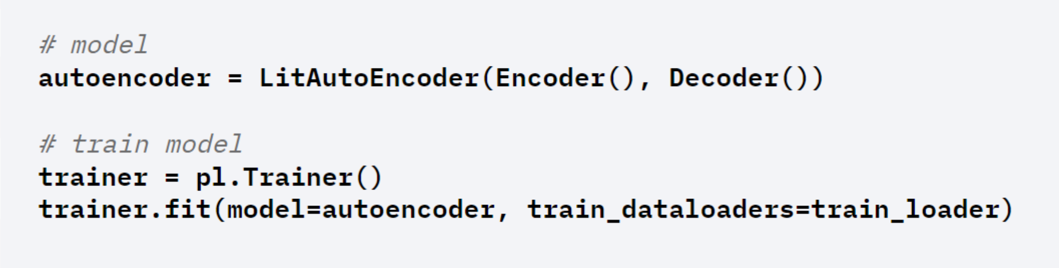
Libraries: Pytorch Lightning (DataModule)



```python
class MNISTDataModule(pl.LightningDataModule):
    def __init__(self, data_dir: str = "path/to/dir", batch_size: int = 32):
        super().__init__()
        self.data_dir = data_dir
        self.batch_size = batch_size

    def setup(self, stage: str):
        self.mnist_test = MNIST(self.data_dir, train=False)
        self.mnist_predict = MNIST(self.data_dir, train=False)
        mnist_full = MNIST(self.data_dir, train=True)
        self.mnist_train, self.mnist_val = random_split(mnist_full, [55000, 5000])

    def train_dataloader(self):
        return DataLoader(self.mnist_train, batch_size=self.batch_size)

    def val_dataloader(self):
        return DataLoader(self.mnist_val, batch_size=self.batch_size)

    def test_dataloader(self):
        return DataLoader(self.mnist_test, batch_size=self.batch_size)
```

Source: lightning.ai

Libraries: Pytorch Lightning (Trainer)



```
# model
autoencoder = LitAutoEncoder(Encoder(), Decoder())

# train model
trainer = pl.Trainer()
trainer.fit(model=autoencoder, train_dataloaders=train_loader)
```

Source: lightning.ai

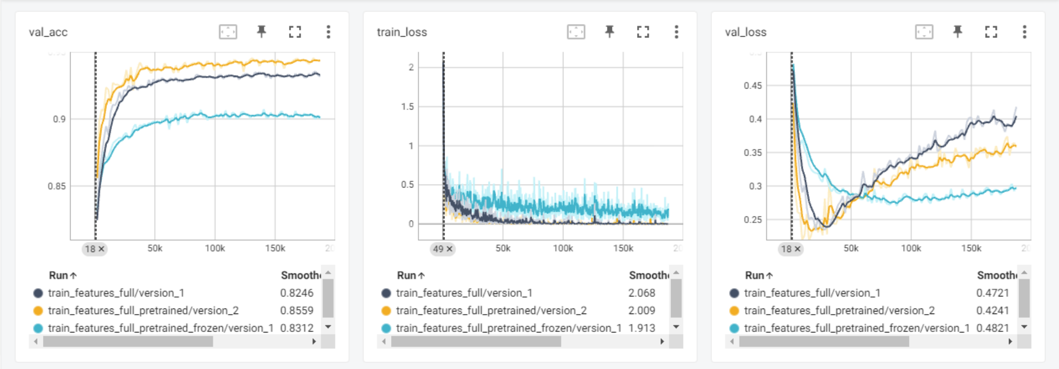
Libraries

- ***Black*** – ***Python*** *code formatter*
  - PyPi – `https://pypi.org/project/black/`
  - Documentation – `https://black.readthedocs.io/`
- ***MyPy*** – ***Python*** *static type checker*
  - PyPi – `https://pypi.org/project/mypy/`
  - Documentation – `https://mypy.readthedocs.io/`
  - Also read about typing – `https://docs.python.org/...`
- ***PyLint*** – ***Python*** *static code analyser*
  - PyPi – `https://pypi.org/project/pylint/`
  - Documentation – `https://pylint.readthedocs.io/`

Libraries

- ***TensorBoard*** – *Tracking and visualizing metrics*
    - Website – `https://www.tensorflow.org/tensorboard`
    - Guide – `https://www.tensorflow.org/tensorboard/get_started`
    - PyTorch Profiler – `https://pypi.org/project/torch-tb-profiler/`
- ***MLFlow*** – *Tracking ML experiments*
    - Website – `https://mlflow.org/`
    - Documentation – `https://mlflow.org/docs/`

Libraries: Tensorboard (Charts)



val_acc

Run ↑	Smooth
● train_features_full/version_1	0.8246
● train_features_full_pretrained/version_2	0.8559
● train_features_full_pretrained_frozen/version_1	0.8312

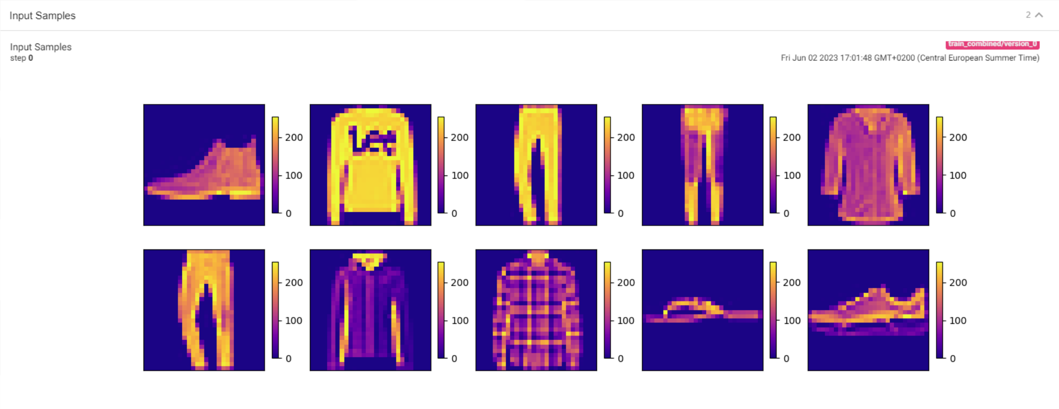
train_loss

Run ↑	Smooth
● train_features_full/version_1	2.068
● train_features_full_pretrained/version_2	2.009
● train_features_full_pretrained_frozen/version_1	1.913

val_loss

Run ↑	Smooth
● train_features_full/version_1	0.4721
● train_features_full_pretrained/version_2	0.4241
● train_features_full_pretrained_frozen/version_1	0.4821

Libraries: Tensorboard (Images)



Input Samples

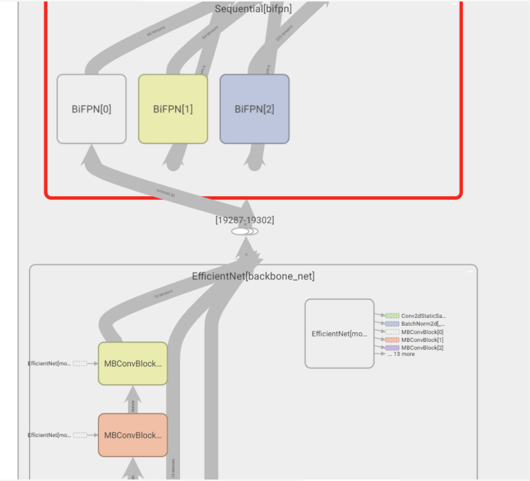
2

Input Samples
step **0**

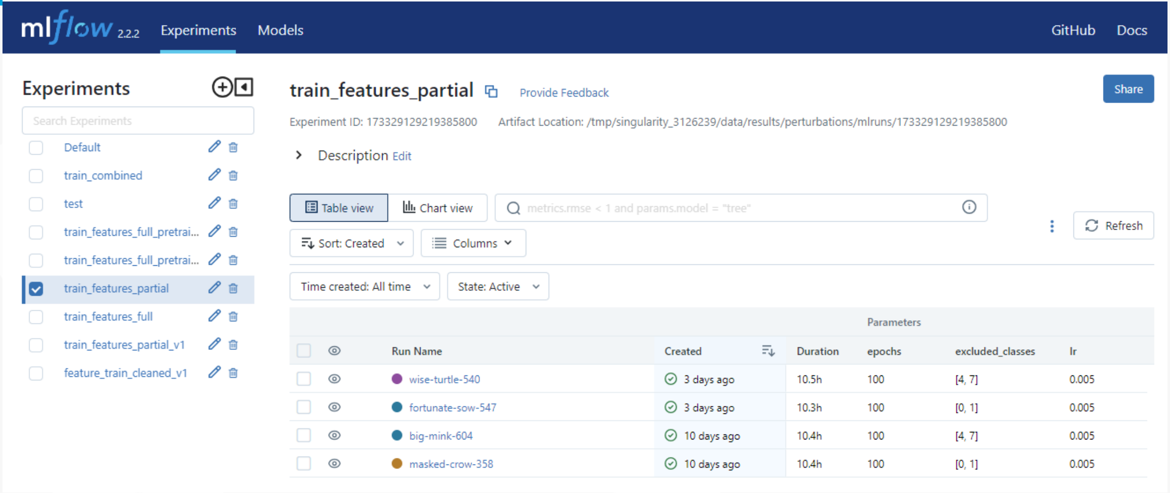
train_combined/version_0

Fri Jun 02 2023 17:01:48 GMT+0200 (Central European Summer Time)

Libraries: Tensorboard (Graphs)



Libraries: MLFlow (Experiments Table)



- □ Default
- □ train_combined
- □ test
- □ train_features_full_pretrai...
- □ train_features_full_pretrai...
- ☑ train_features_partial
- □ train_features_full
- □ train_features_partial_v1
- □ feature_train_cleaned_v1

train_features_partial

Experiment ID: 173329129219385800 Artifact Location: /tmp/singularity_3126239/data/results/perturbations/mlruns/173329129219385800

Description Edit

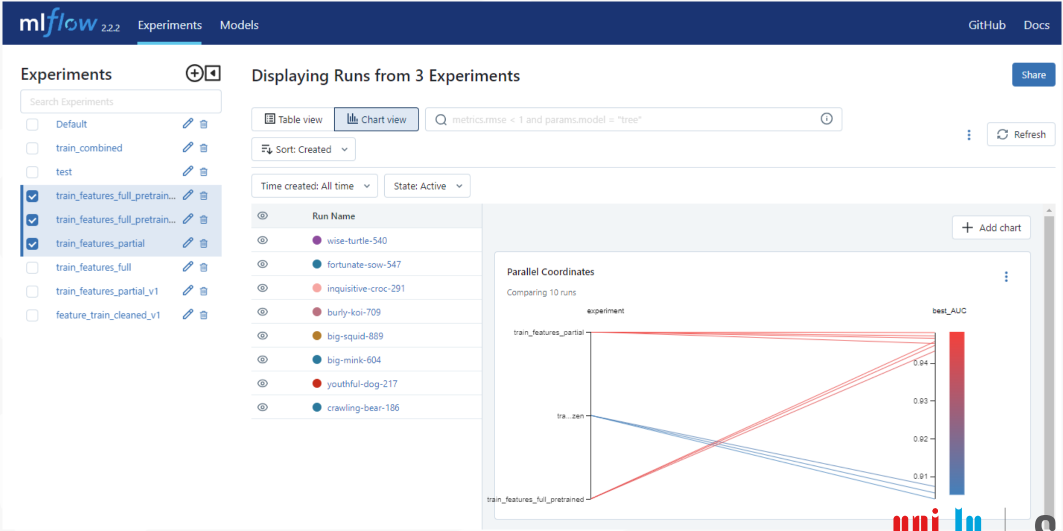
Sort: Created

Time created: All time

State: Active

					Parameters		
□		Run Name	Created	Duration	epochs	excluded_classes	lr
□		● wise-turtle-540	✓ 3 days ago	10.5h	100	[4, 7]	0.005
□		● fortunate-sow-547	✓ 3 days ago	10.3h	100	[0, 1]	0.005
□		● big-mink-604	✓ 10 days ago	10.4h	100	[4, 7]	0.005
□		● masked-crow-358	✓ 10 days ago	10.4h	100	[0, 1]	0.005

Libraries: MLFlow (Charts)



Experiments

- ☐ Default
- ☐ train_combined
- ☐ test
- ☑ train_features_full_pretrain...
- ☑ train_features_full_pretrain...
- ☑ train_features_partial
- ☐ train_features_full
- ☐ train_features_partial_v1
- ☐ feature_train_cleaned_v1

Displaying Runs from 3 Experiments

Run Name
wise-turtle-540
fortunate-sow-547
inquisitive-croc-291
burly-koi-709
big-squid-889
big-mink-604
youthful-dog-217
crawling-bear-186

Parallel Coordinates

Comparing 10 runs

Other Tips

- **ULHPC** *tutorials* website has a huge amount of resources
  - SSH, CUDA, deep learning, SLURM, Python, Singularity...
- Use **Version Control Systems (VCS)** – ***Git***
  - Push *before* running your experiments
  - Run your code on a specific commit, not on modified files
- Do not rush your experiments; running does not mean working
- **Jupyter** notebooks are handy for quick data analysis, but use **Python** scripts when running experiments[11]
- Start using ***Object Oriented Programming (OOP)***

[11] Some reasons why in *this* article

Other Tips

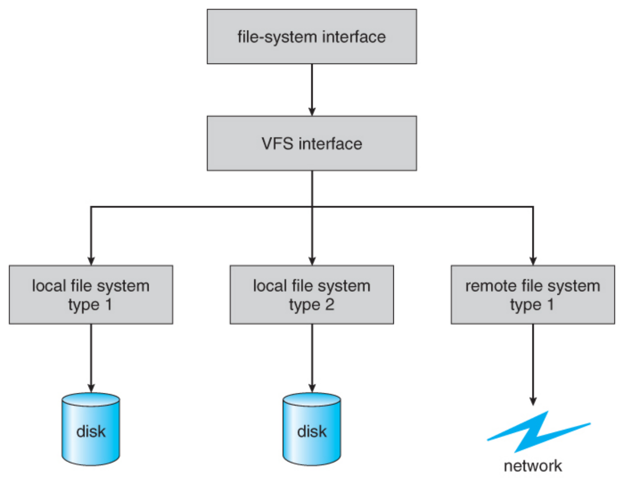
- Use LaTeX – `https://www.latex-project.org/`
  - ***Beamer*** – `https://en.wikipedia.org/wiki/`
  - Template – `https://gitlab.uni.lu/rhermary/snt-beamer-template`
- ***RTFM***
- Lock your computer (and turn it off)
  - ⊞ + `L`
  - `Ctrl` + ⌘ + `Q`
  - 🐧 + `L` or `Ctrl` + `Alt` + `L`
  - Instead of letting your computer run unlocked for some random task, look at `tmux` (terminal multiplexer)

Thanks for your attention!

Questions?

UNIVERSITÉ DU
LUXEMBOURG

Directory Mounting (File System Functioning)

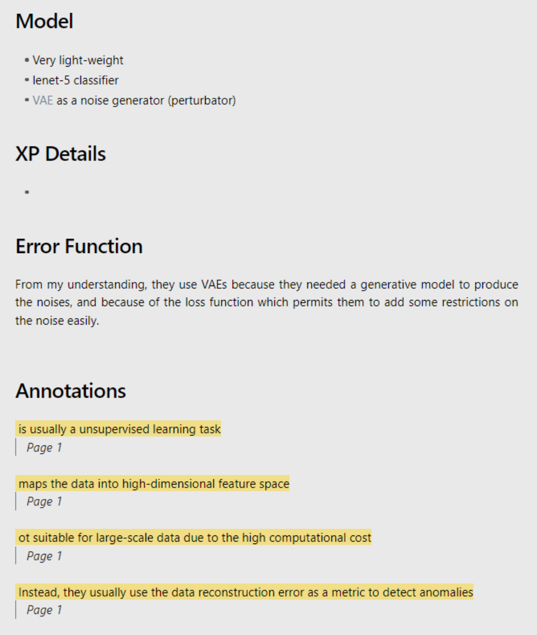


Source: `cs.uic.edu`

Dockerfile: Example

```
FROM ubuntu:20.04
LABEL maintainer="rhermary"

WORKDIR /app/
COPY requirements.txt requirements.txt
ENV PYTHONPATH src:efficientdet

ENV DEBIAN_FRONTEND noninteractive
RUN apt-get update -y\
   && apt-get install curl wget software-properties-common build-essential git -y\
   && add-apt-repository ppa:deadsnakes/ppa -y\
   && apt install python3.10 python3.10-distutils -y\
   && update-alternatives --install /usr/bin/python python /usr/bin/python3.10 1\
   && curl -sS https://bootstrap.pypa.io/get-pip.py | python\
   && python -m pip install --upgrade pip

RUN python -m pip install -r requirements.txt
CMD /bin/bash
```

Obsidian



Model

- Very light-weight
- lenet-5 classifier
- VAE as a noise generator (perturbator)

XP Details

-

Error Function

From my understanding, they use VAEs because they needed a generative model to produce the noises, and because of the loss function which permits them to add some restrictions on the noise easily.

Annotations

is usually a unsupervised learning task

> *Page 1*

maps the data into high-dimensional feature space

> *Page 1*

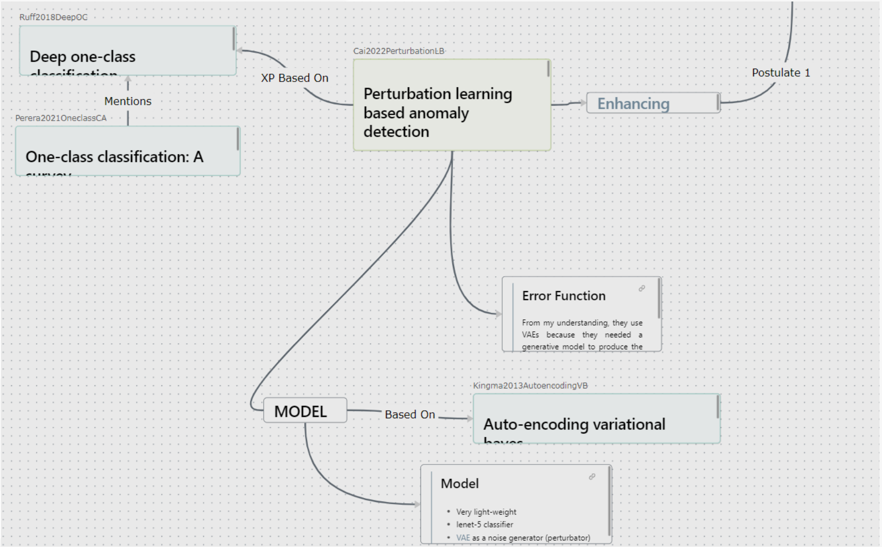
ot suitable for large-scale data due to the high computational cost

> *Page 1*

Instead, they usually use the data reconstruction error as a metric to detect anomalies

> *Page 1*

Obsidian


Ruff2018DeepOC
Deep one-class classification
Mentions
Perera2021OneclassCA
One-class classification: A survey
XP Based On
Cai2022PerturbationLB
Perturbation learning based anomaly detection
Enhancing
Postulate 1
MODEL
Based On
Kingma2013AutoencodingVB
Auto-encoding variational bayes

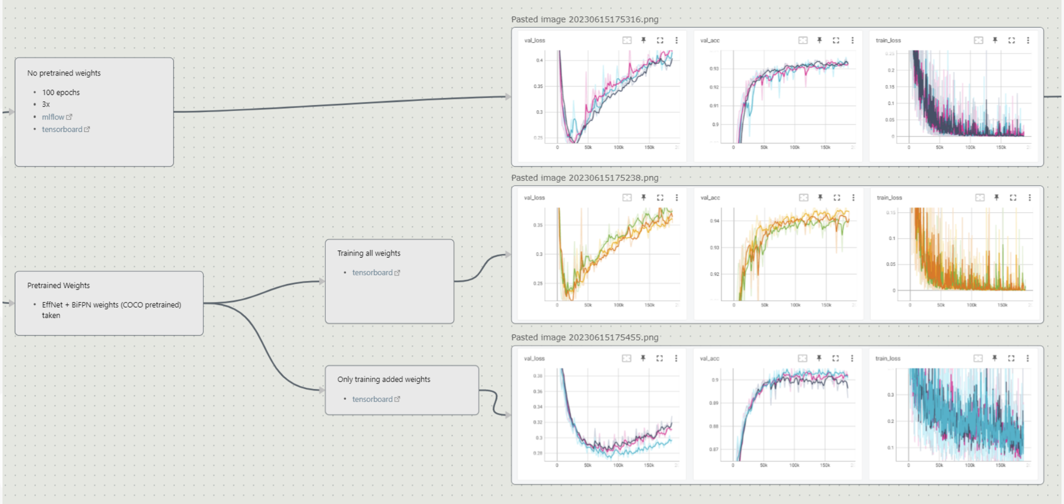
Error Function

From my understanding, they use VAEs because they needed a generative model to produce the

Model

- Very light-weight
- lenet-5 classifier
- VAE as a noise generator (perturbator)

Obsidian



No pretrained weights

- 100 epochs
- 3x
- mlflow
- tensorboard

Pretrained Weights

- EffNet + BiFPN weights (COCO pretrained) taken

Training all weights

- tensorboard

Only training added weights

- tensorboard

Pasted image 20230615175316.png

Pasted image 20230615175238.png

Pasted image 20230615175455.png