Playing with ML tools on a cluster: my experience using Hydra and Weights&Biases on Jean Zay

Zaccharie Ramzi, AI dev talk #4
(zaccharie.ramzi@gmail.com)

What I want to convey in this talk

JZ + W&B + Hydra => better ML experiments

- faster
- more diverse
- more thorough
- well logged / cataloged
- easily reproducible
- clearly configured
- less code / concerns separated

Jean Zay: faster, more diverse and thorough XPs

- 2696 GPU Nvidia V100 + 440 GPU Nvidia A100
- 30 Po
- relatively easy access procedure for AI projects: it’s FREE!

SLURM

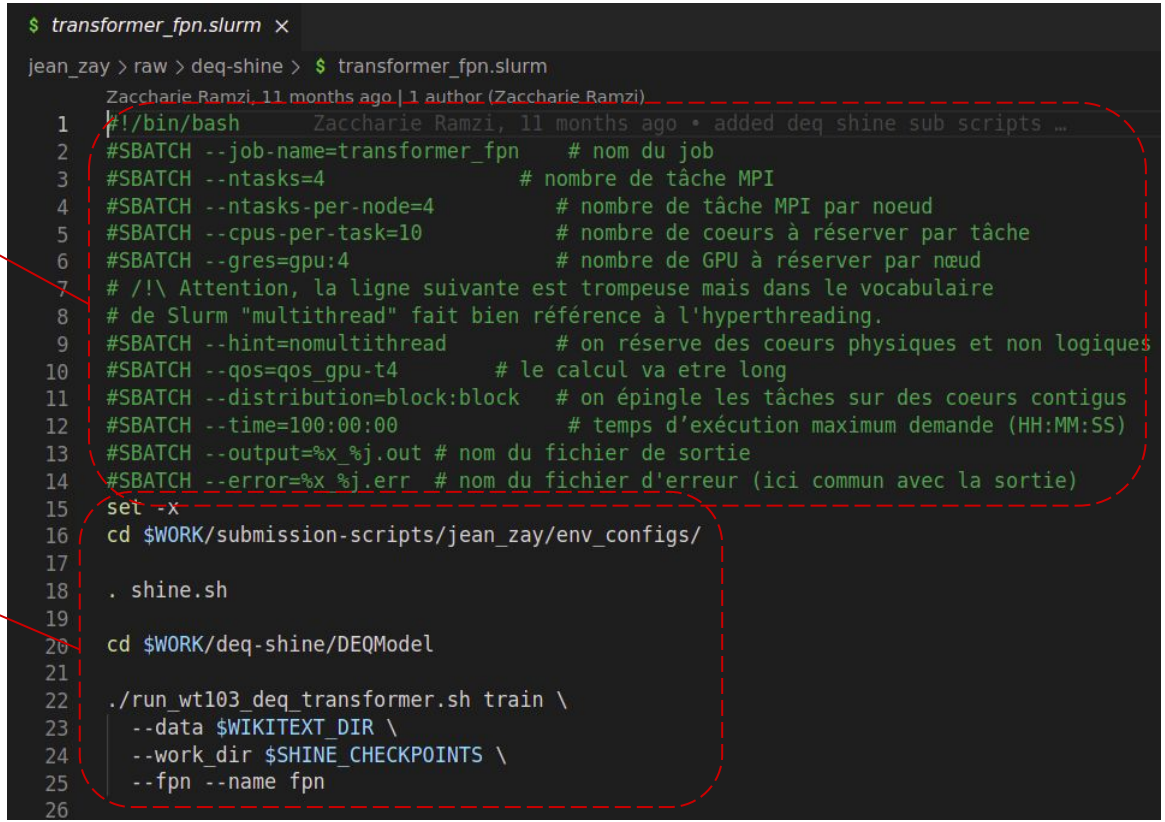
On JZ scripts are not run “directly” on the connexion node.

SLURM: workload management system => queue of jobs for all users.

Job example on JZ

Worker config

Env config + script running



$ transformer_fpn.slurm ×

jean_zay > raw > deq-shine > $ transformer_fpn.slurm

Zaccharie Ramzi, 11 months ago | 1 author (Zaccharie Ramzi)

```
1  #!/bin/bash      Zaccharie Ramzi, 11 months ago • added deq shine sub scripts …
2  #SBATCH --job-name=transformer_fpn    # nom du job
3  #SBATCH --ntasks=4                # nombre de tâche MPI
4  #SBATCH --ntasks-per-node=4          # nombre de tâche MPI par noeud
5  #SBATCH --cpus-per-task=10           # nombre de coeurs à réserver par tâche
6  #SBATCH --gres=gpu:4                 # nombre de GPU à réserver par nœud
7  # /!\ Attention, la ligne suivante est trompeuse mais dans le vocabulaire
8  # de Slurm "multithread" fait bien référence à l'hyperthreading.
9  #SBATCH --hint=nomultithread         # on réserve des coeurs physiques et non logiques
10 #SBATCH --qos=qos_gpu-t4        # le calcul va etre long
11 #SBATCH --distribution=block:block   # on épingle les tâches sur des coeurs contigus
12 #SBATCH --time=100:00:00             # temps d'exécution maximum demande (HH:MM:SS)
13 #SBATCH --output=%x_%j.out # nom du fichier de sortie
14 #SBATCH --error=%x_%j.err  # nom du fichier d'erreur (ici commun avec la sortie)
15 set -x
16 cd $WORK/submission-scripts/jean_zay/env_configs/
17
18 . shine.sh
19
20 cd $WORK/deq-shine/DEQModel
21
22 ./run_wt103_deq_transformer.sh train \
23   --data $WIKITEXT_DIR \
24   --work_dir $SHINE_CHECKPOINTS \
25   --fpn --name fpn
26
```

Job array example on JZ

Number of sweeps

Grid search config

Grid search call

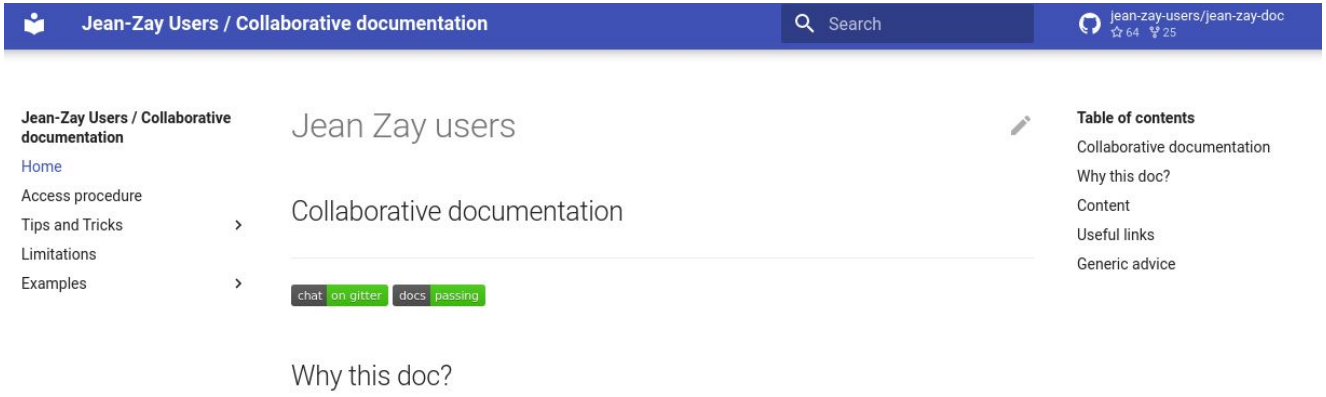
momentumvarnet_recon_big.slurm M

jean_zay > raw > fastmri-momentum > $ momentumvarnet_recon_big.slurm

```
#!/bin/bash
#SBATCH --job-name=momentum_varnet_big     # nom du job
#SBATCH --ntasks=1                 # nombre de tâche MPI
#SBATCH --ntasks-per-node=1           # nombre de tâche MPI par noeud
#SBATCH --cpus-per-task=24            # nombre de coeurs à réserver par tâche
#SBATCH --gres=gpu:8                 # nombre de GPU à réserver par nœud
#SBATCH --hint=nomultithread          # on réserve des coeurs physiques et non logiques
#SBATCH --qos=qos_gpu-t4        # le calcul va etre long
#SBATCH --distribution=block:block    # on épingle les tâches sur des coeurs contigus
#SBATCH --time=100:00:00               # temps d'exécution maximum demande (HH:MM:SS)
#SBATCH --output=%x_%j.out # nom du fichier de sortie
#SBATCH --error=%x_%j.err  # nom du fichier d'erreur (ici commun avec la sortie)
#SBATCH --array=0-1
cd $WORK/submission-scripts/jean_zay/env_configs/

. momentum.sh

cd $WORK/fastmri-momentum/

opt[0]='--use_momentum --gamma 0.2'
opt[1]='--use_momentum --gamma 0.2 --num_cascades 24'

python fastmri_momentum/knee_training.py \
    --data_path $ROOT_DIR \
    --gpus 8 \
    --max_epochs 50 \
    --default_root_dir $ROOT_DIR \
    ${opt[$SLURM_ARRAY_TASK_ID]}
```

Don’t be afraid

- Notebooks available for debugging (w. up to 8 GPUs)
- Hours deducted on actual use
- Assist is reponsive
- Community is active:
  - gitter
  - jean-zay-doc.readthedocs.io



More XPs ?

With more XPs comes some hardships:

- “I had this result once but the code changed so much...”
- “I don’t remember the parameters I used for this...”

W&B presentation

Pros:

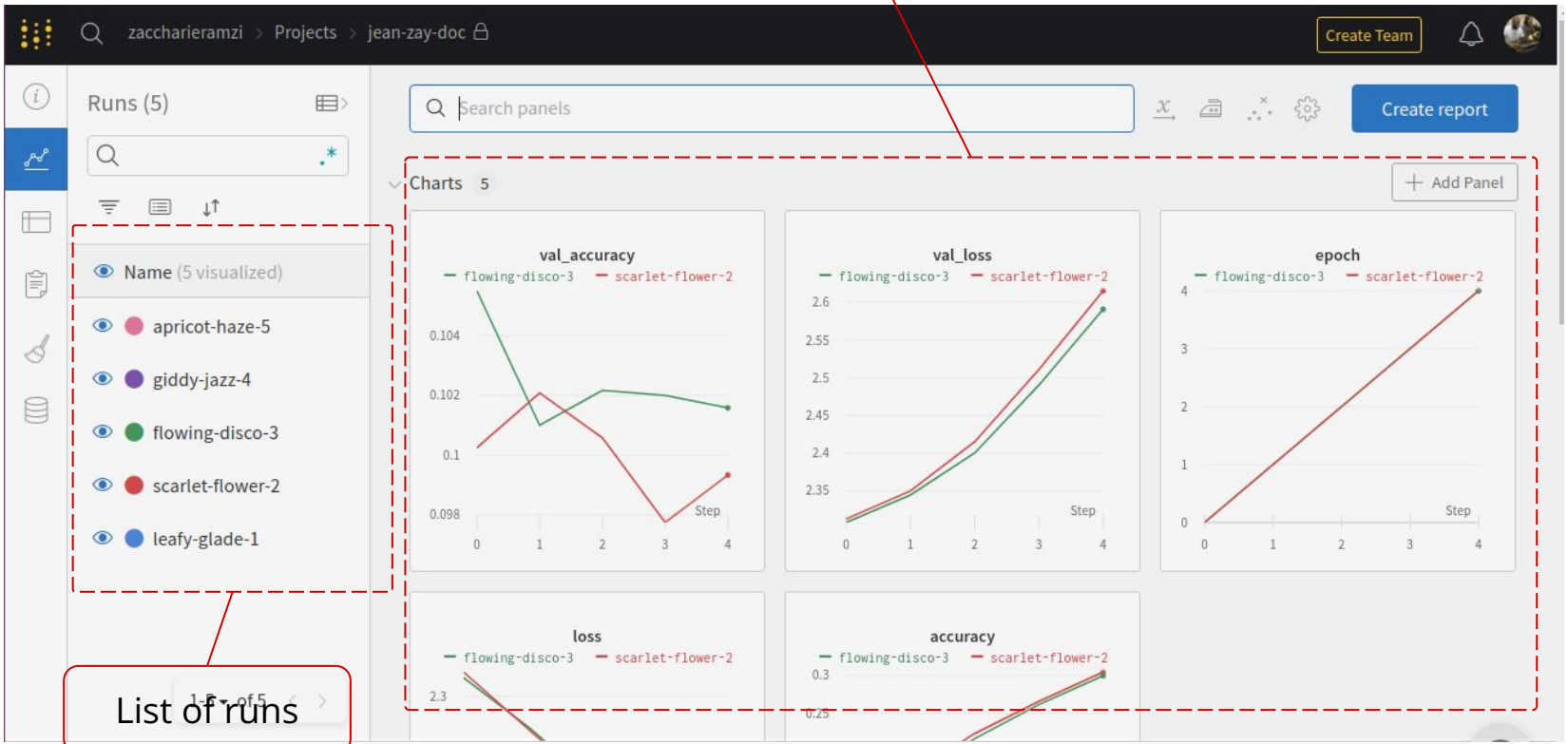
- stores all training related artifacts: memory consumption, images, metrics, weights
- logs configurations + code states
- client Open Source + good integration

Cons:

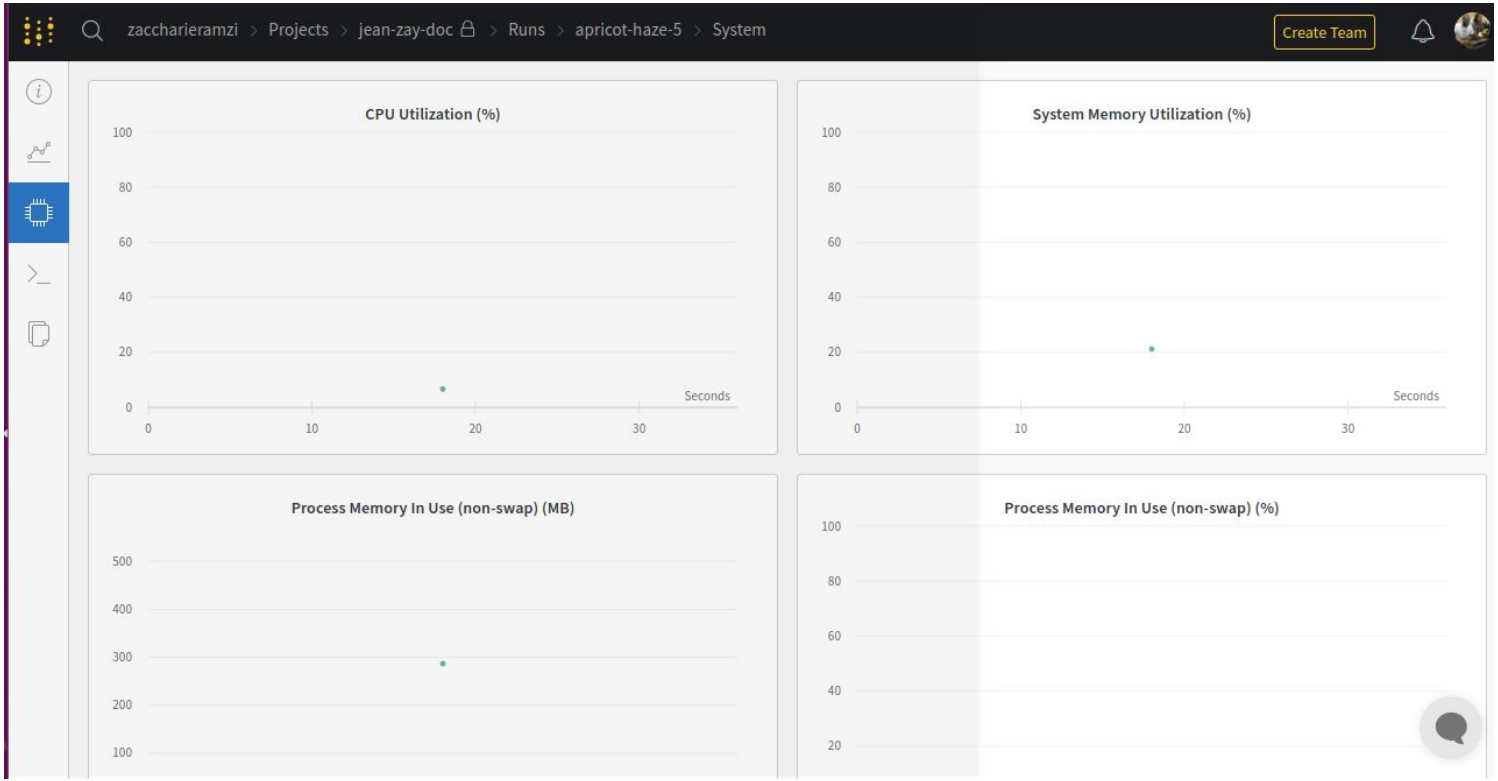
- UI not Open Source
- only free for under 100 GB of storage ($0.08 per GB)*

* Possibility to create academic teams

W&B: UI


Metrics
zaccharieramzi > Projects > jean-zay-doc
Create Team
Runs (5)
Search panels
Create report
Charts 5
Add Panel
Name (5 visualized)
apricot-haze-5
giddy-jazz-4
flowing-disco-3
scarlet-flower-2
leafy-glade-1
val_accuracy
flowing-disco-3
scarlet-flower-2
0.104
0.102
0.1
0.098
Step
0
1
2
3
4
val_loss
2.6
2.55
2.5
2.45
2.4
2.35
epoch
loss
2.3
accuracy
0.3
0.25
List of runs

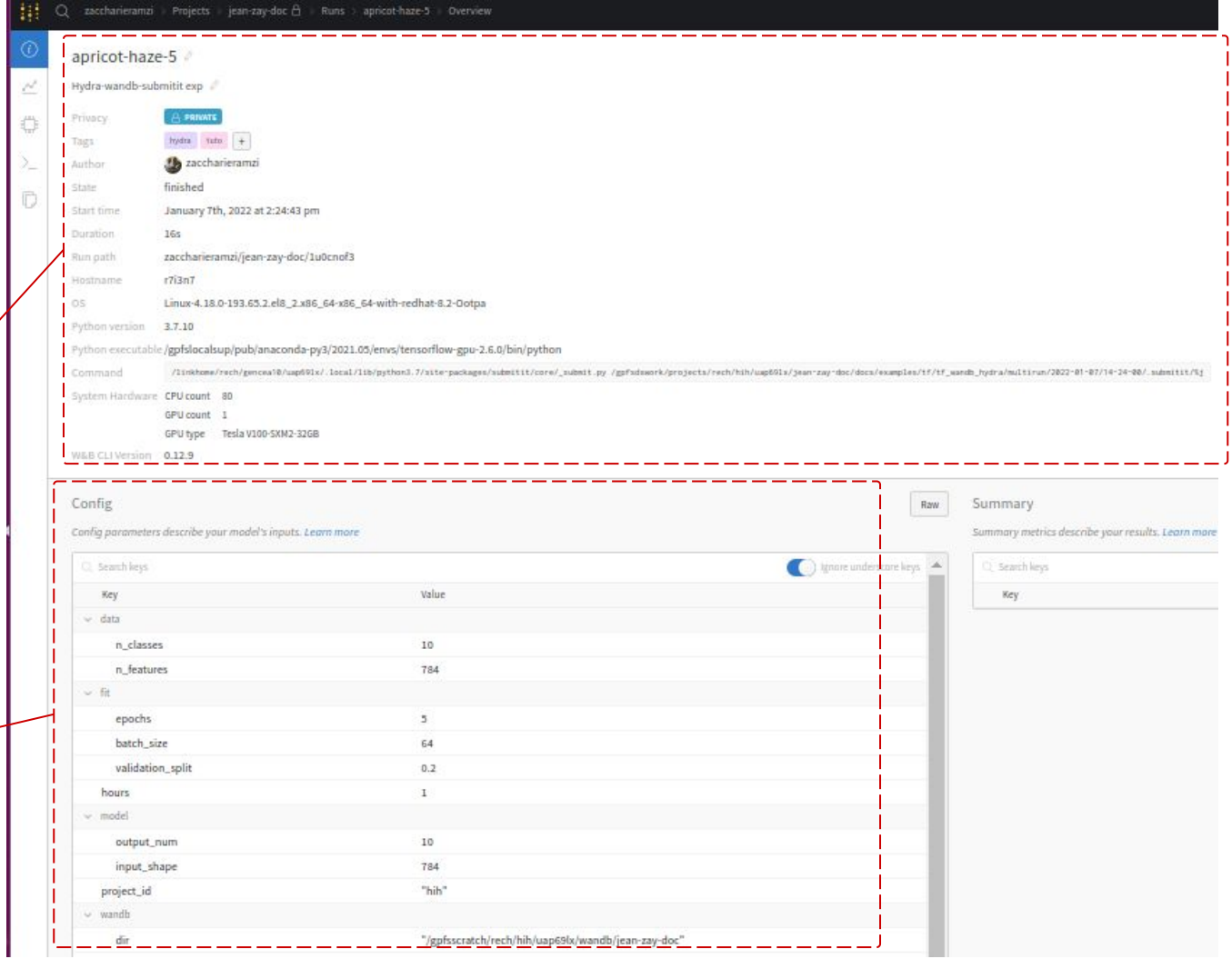
W&B: UI


zaccharieramzi > Projects > jean-zay-doc > Runs > apricot-haze-5 > System
Create Team
CPU Utilization (%)
100
80
60
40
20
0
Seconds
0
10
20
30
System Memory Utilization (%)
100
80
60
40
20
0
Seconds
0
10
20
30
Process Memory In Use (non-swap) (MB)
500
400
300
200
100
Process Memory In Use (non-swap) (%)
100
80
60
40
20

W&B: UI

Meta data

Configuration



W&B: in your code

W&B init

W&B cbacks (classic and custom)

```python
if __name__ == '__main__':
    config = dict(
        model_name='ifip_xl',
        n_samples=N_TRAIN,
        batch_size=90_000,
        epochs=2000,
        use_wd=False,
        seed=0,
        simple_compile=True,
        activation='siren',
        pos_enc_type='random',
        dropout=0.,
        latent_space_dim=512,
        model_kwargs=dict(
            D=16,
            skips=[4, 8, 12],
            W=512,
        )
    )
    wandb.init(
        project="ifip",
        notes="Maxing out the GPU",
        tags=["cifar"],
        config=config,
        dir=WANDB_PATH,
    )
    train_model_cifar(**config)
```

```python
callbacks = []
checkpoints_path = Path(checkpoints_path)
checkpoints_path.mkdir(exist_ok=True)
callbacks.append(ModelCheckpoint(checkpoints_path / f'{model_name}.h5'
callbacks.append(TensorBoard(log_dir=Path(logs_path) / f'{model_name}'
callbacks.append(MyWandbCallback(monitor='loss', save_weights_only=Tru
callbacks.append(HOLogger(model, use_wd, simple_compile))
callbacks.append(ReconstructionCallback(model, 0, recon_freq))
```

Reproducible experiments

W&B solves part of the problem.

You just saw a bad config example => this leads to more difficulty when reproducing large scale XPs.

Hydra presentation

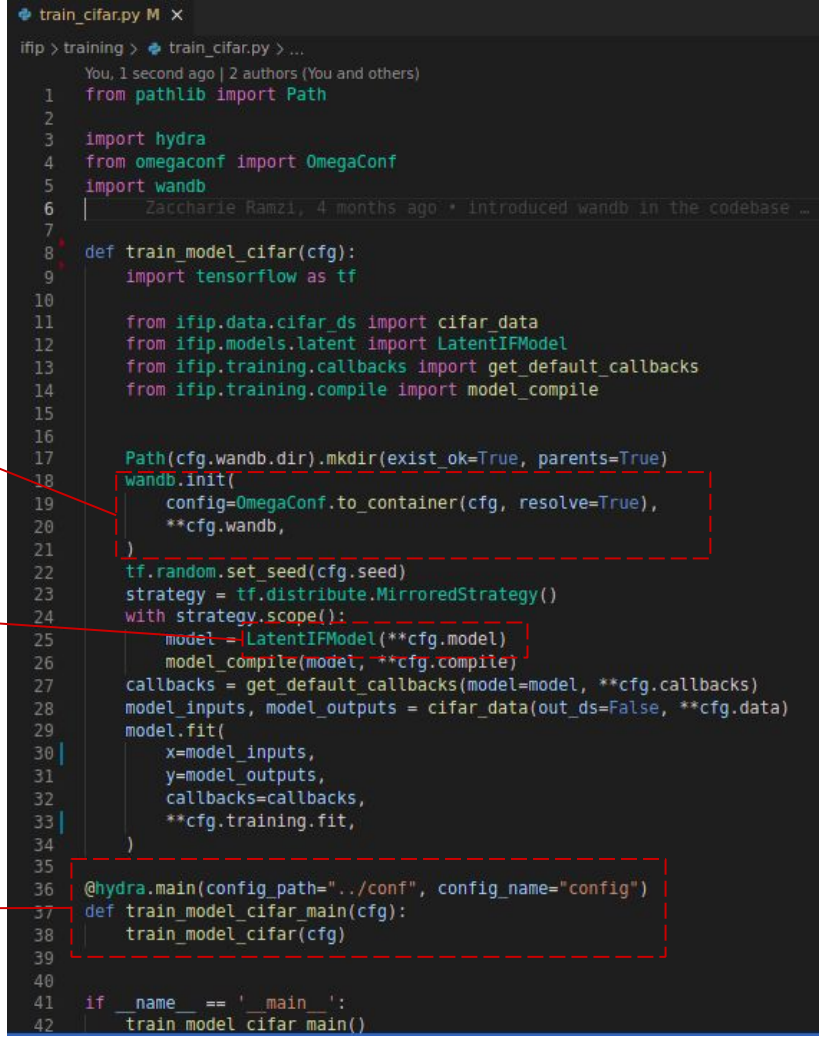
- configuration parsing + interpolation
- CLI for free

Hydra in practice

W&B integration

Config "parsing"

CLI definition



Hydra in practice

Interpolation

config.yaml

ifip > conf > config.yaml

```yaml
training:
  fit:
    epochs: 2000
    batch_size: 90000
  use_l2: False

compile:
  simple_compile: True
  use_wd: False
  n_train: ${data.n_samples}
  n_points: 1024
  batch_size: ${training.fit.batch_size}


callbacks:
  model_name: ${model.name}
  use_wd: ${compile.use_wd}
  simple_compile: ${compile.simple_compile}
  recon_freq: 10
  checkpoints_path: ${paths.checkpoints}
  logs_path: ${paths.logs}

model:
  name: null
  n_training_data: ${data.n_samples}
  pos_enc_type: logspace
  latent_space_dim: 128
  pos_emb_length: 10
  use_l2_reg: ${and:${not:${compile.use_wd}},${training.use_l2}}
  mlp:
    D: 8
    W: 256
    skips: 4
```

Putting it all together

Grid search from Jean Zay front node using Hydra multirun option, all in Python, and logged offline in Weights&Biases

`jz-hydra-submitit-launcher`

Example classical steps

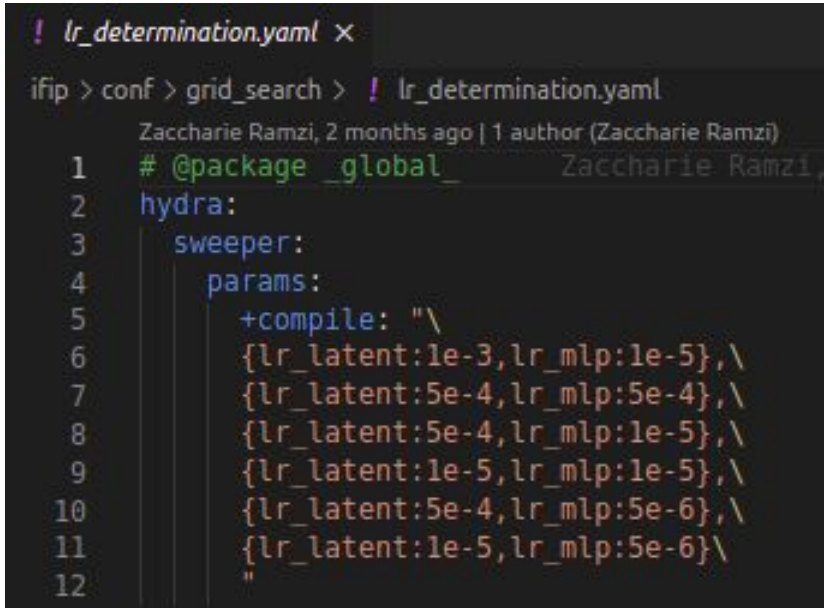
- `ssh jz`
- `cd $WORK/my-project`
- `sbatch jz_script/my_training_script.slurm`

Example new steps

- `ssh jz`
- `cd $WORK/my-project`
- `hydra-submitit-launch my_training_script.py dev`
  - `dev`
  - `t3`
  - `t4`
  - `4gpus_dev`
  - `4gpus_t3`
  - `4gpus_t4`

Combined advantages

- No more .slurm writing : all in Python!
- Easy CLI generation w. Hydra
  - no more `argparse` come and go
- A single script for all your grid searches!*
  - `hydra-submitit-launch my_training_script.py dev hydra/sweeper/params=grid_search`
  - => concerns separated!



* In Hydra 1.2, upcoming version

What about W&B?

A bit more complex on JZ:

- `screen -S wand_sync`
- `wandb sync run_dir`
- `ctrl A+D`

syncall_wandb.sh

```
#!/bin/bash

offline_runs="$1/offline-run*"
while :
do
    for ofrun in $offline_runs
    do
        wandb sync $ofrun;
    done
    sleep 5m
done
```

Conclusion

JZ + W&B + Hydra => better ML experiments

- faster
- more diverse
- more thorough
- well logged / cataloged
- easily reproducible
- clearly configured
- less code / concerns separated

Thank you!

Questions ?