



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

```
$ transformer_fpn.slurm x
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 noeud
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
```

Worker config

Env config +
script running

Job array example on JZ

Number of sweeps

Grid search config

Grid search call

```
momentumvarnet_recon_big.slurm M x
jean_zay > raw > fastmri-momentum > $ momentumvarnet_recon_big.slurm
1  #!/bin/bash
2  #SBATCH --job-name=momentum_varnet_big # nom du job
3  #SBATCH --ntasks=1 # nombre de tâche MPI
4  #SBATCH --ntasks-per-node=1 # nombre de tâche MPI par noeud
5  #SBATCH --cpus-per-task=24 # nombre de coeurs à réserver par tâche
6  #SBATCH --gres=gpu:8 # nombre de GPU à réserver par noeud
7  #SBATCH --hint=nomultithread # on réserve des coeurs physiques et non logiques
8  #SBATCH --qos=qos_gpu-t4 # le calcul va etre long
9  #SBATCH --distribution=block:block # on épingle les tâches sur des coeurs contigus
10 #SBATCH --time=100:00:00 # temps d'exécution maximum demande (HH:MM:SS)
11 #SBATCH --output=%x_%j.out # nom du fichier de sortie
12 #SBATCH --error=%x_%j.err # nom du fichier d'erreur (ici commun avec la sortie)
13 #SBATCH --array=0-1
14 cd $WORK/submission-scripts/jean_zay/env_configs/
15
16 . momentum.sh
17
18 cd $WORK/fastmri-momentum/
19
20 opt[0]='--use_momentum --gamma 0.2'
21 opt[1]='--use_momentum --gamma 0.2 --num_cascades 24'
22
23 python fastmri_momentum/knee_training.py \
24     --data_path $ROOT_DIR \
25     --gpus 8 \
26     --max_epochs 50 \
27     --default_root_dir $ROOT_DIR \
28     ${opt[$SLURM_ARRAY_TASK_ID]}
```

Don't be afraid

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

The screenshot shows the top navigation bar of the 'Jean-Zay Users / Collaborative documentation' website. The header is dark blue with a search bar and repository statistics (64 stars, 25 forks). The main content area has a light blue background. On the left is a sidebar with a table of contents. The main content area displays the title 'Jean Zay users' and a sub-section 'Collaborative documentation' with a 'chat on gitter' and 'docs passing' status. Below this is the heading 'Why this doc?'.

Jean-Zay Users / Collaborative documentation

Home

Access procedure

Tips and Tricks >

Limitations

Examples >

Jean Zay users

Collaborative documentation

chat on gitter docs passing

Why this doc?

Table of contents

- Collaborative documentation
- Why this doc?
- Content
- Useful links
- Generic advice

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:

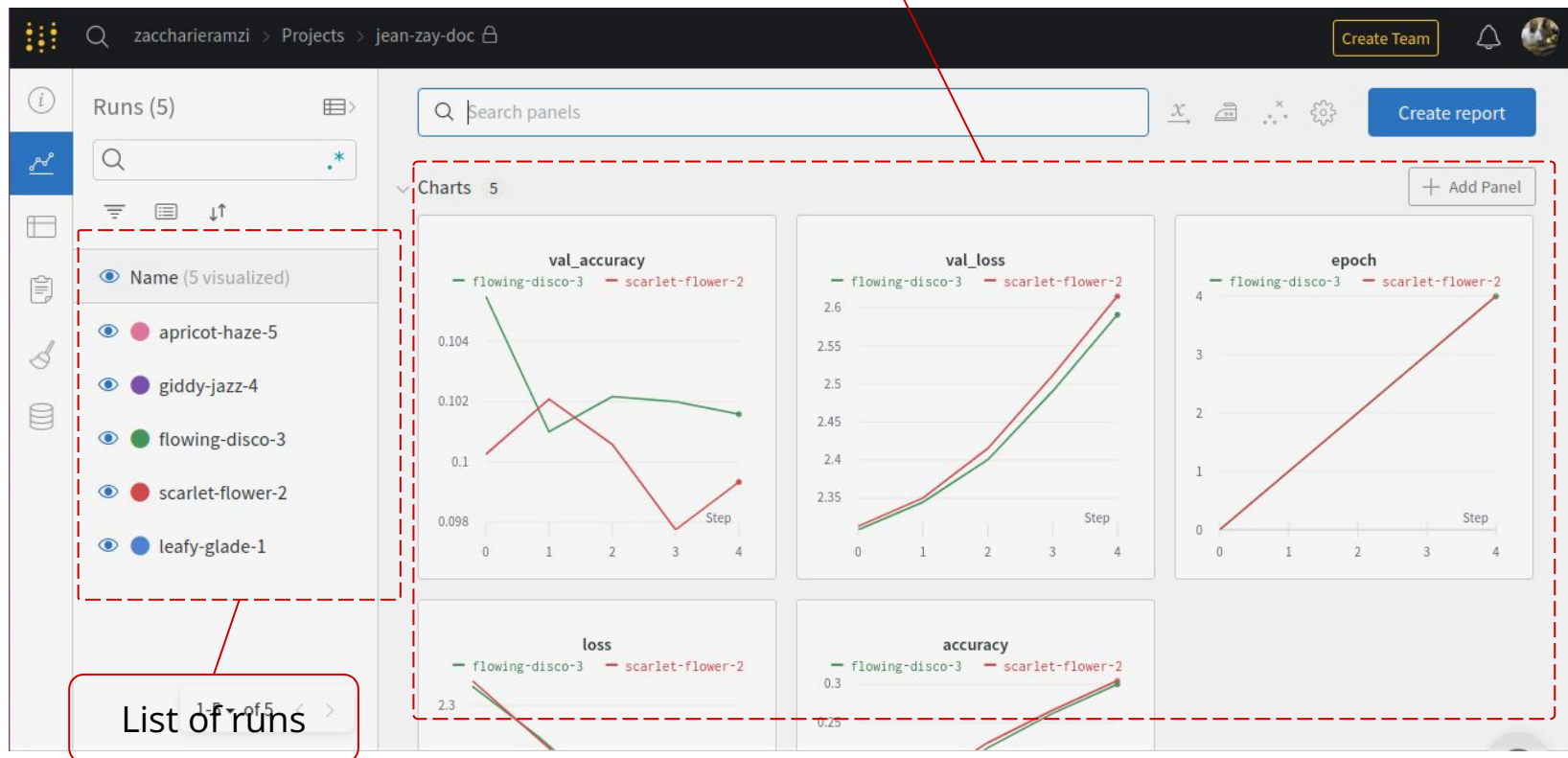
- UI not Open Source
- only free for under 100 GB of storage (\$0.08 per GB)*



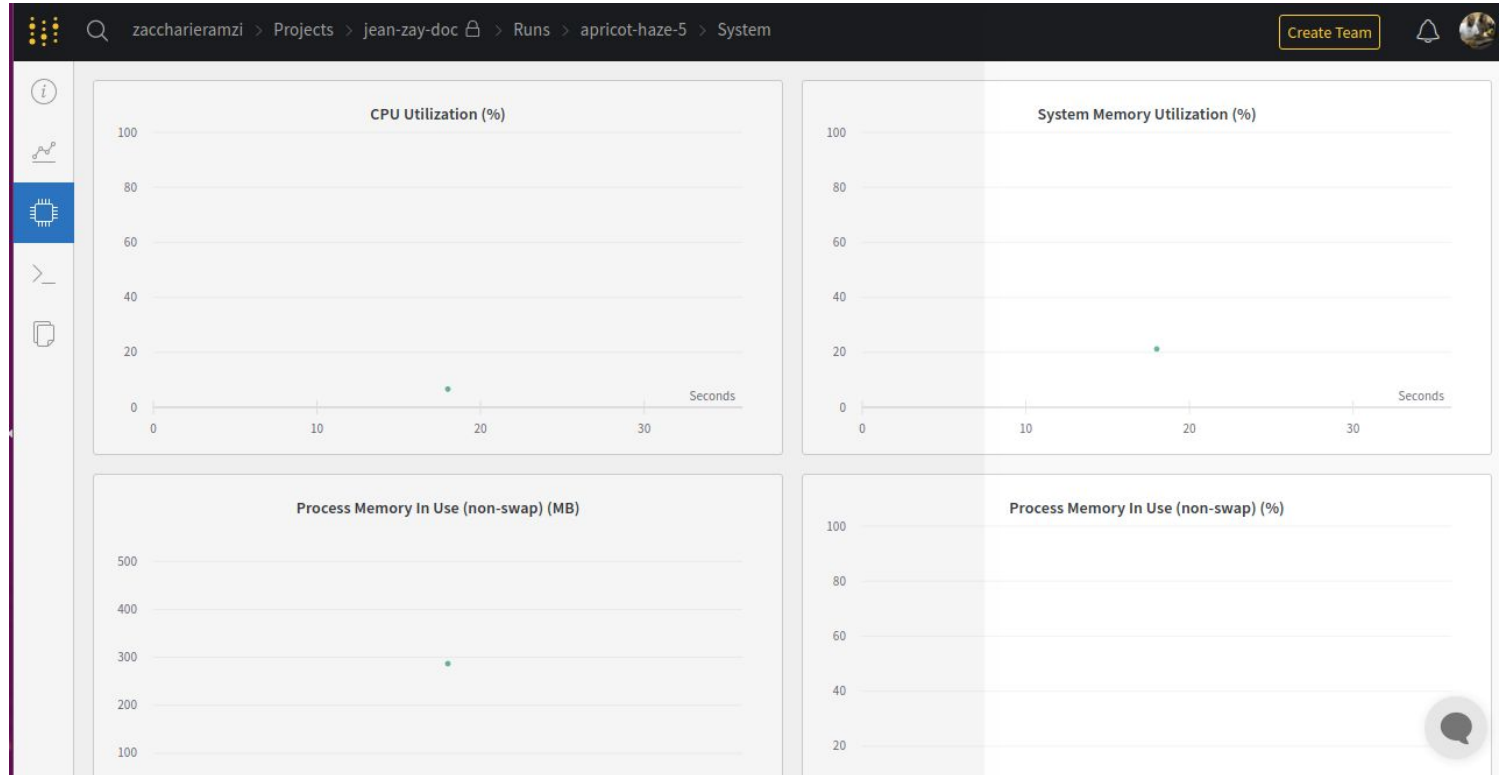
* Possibility to create academic teams

W&B: UI

Metrics



W&B: UI



W&B: UI

Meta data

apricot-haze-5

Hydra-wandb-submitit exp

Privacy

PRIVATE

Tags

hydra yolo +

Author

zaccharieramzi

State

finished

Start time

January 7th, 2022 at 2:24:43 pm

Duration

16s

Run path

zaccharieramzi/jean-zay-doc/1u0cnoF3

Hostname

r7i3nT

OS

Linux-4.18.0-193.65.2.el8_2.x86_64-x86_64-with-redhat-8.2-Optpa

Python version

3.7.10

Python executable

/gpfslocalsup/pub/anaconda-py3/2021.05/envs/tensorflow-gpu-2.6.0/bin/python

Command

```
/11nktome/rech/gencea/8/uap69ix/.local/lib/python3.7/site-packages/submitit/core/_submit.py /gpfsdwork/projects/rech/hih/uap69ix/jean-zay-doc/docs/examples/1f/1f_wandb_hydra/MultiRun/2022-01-07/14-24-00/_submitit/1j
```

System Hardware

CPU count 80

GPU count 1

GPU type Tesla V100-SXM2-32GB

W&B CLI Version

0.12.9

Config

Config parameters describe your model's inputs. [Learn more](#)

Search keys

ignore under score keys

Key	Value
data	
n_classes	10
n_features	784
fit	
epochs	3
batch_size	64
validation_split	0.2
hours	1
model	
output_num	10
input_shape	784
project_id	"hih"
wandb	
dir	"/gpfsscratch/rech/hih/uap69ix/wandb/jean-zay-doc"

Raw

Summary

Summary metrics describe your results. [Learn more](#)

Search keys

Key

Configuration

W&B: in your code

W&B init

W&B cbacks (classic and custom)

```
75 if __name__ == '__main__':
76     ...config = dict(
77         ...model_name='ifip_x1',
78         ...n_samples=N_TRAIN,
79         ...batch_size=90_000,
80         ...epochs=2000,
81         ...use_wd=False,
82         ...seed=0,
83         ...simple_compile=True,
84         ...activation='siren',
85         ...pos_enc_type='random',
86         ...dropout=0.,
87         ...latent_space_dim=512,
88         ...model_kwargs=dict(
89             ...D=16,
90             ...skips=[4, 8, 12],
91             ...W=512,
92         ...))
93     ...
94     ...wandb.init(
95         ...project="ifip",
96         ...notes="Maxing out the GPU",
97         ...tags=["cifar"],
98         ...config=config,
99         ...dir=WANDB_PATH,
100     ...))
101     ...train_model_cifar(**config)
```

```
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=True))
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

```
train_cifar.py M X
ifip > training > train_cifar.py > ...
  You, 1 second ago | 2 authors (You and others)
  1 from pathlib import Path
  2
  3 import hydra
  4 from omegaconf import OmegaConf
  5 import wandb
  6 | Zaccharie Ramzi, 4 months ago • introduced wandb in the codebase ...
  7
  8 def train_model_cifar(cfg):
  9     import tensorflow as tf
  10
  11     from ifip.data.cifar_ds import cifar_data
  12     from ifip.models.latent import LatentIFModel
  13     from ifip.training.callbacks import get_default_callbacks
  14     from ifip.training.compile import model_compile
  15
  16
  17     Path(cfg.wandb.dir).mkdir(exist_ok=True, parents=True)
  18     wandb.init(
  19         config=OmegaConf.to_container(cfg, resolve=True),
  20         **cfg.wandb,
  21     )
  22     tf.random.set_seed(cfg.seed)
  23     strategy = tf.distribute.MirroredStrategy()
  24     with strategy.scope():
  25         model = LatentIFModel(**cfg.model)
  26         model_compile(model, **cfg.compile)
  27     callbacks = get_default_callbacks(model=model, **cfg.callbacks)
  28     model_inputs, model_outputs = cifar_data(out_ds=False, **cfg.data)
  29     model.fit(
  30         x=model_inputs,
  31         y=model_outputs,
  32         callbacks=callbacks,
  33         **cfg.training.fit,
  34     )
  35
  36 @hydra.main(config_path="../conf", config_name="config")
  37 def train_model_cifar_main(cfg):
  38     train_model_cifar(cfg)
  39
  40
  41 if __name__ == '__main__':
  42     train_model_cifar_main()
```


Hydra in practice

Interpolation

```
! config.yaml x
ifip > conf > ! config.yaml
19 training:
20   fit:
21     epochs: 2000
22     batch_size: 90000
23     use_l2: False
24
25 compile:
26   simple_compile: True
27   use_wd: False
28   n_train: ${data.n_samples}
29   n_points: 1024
30   batch_size: ${training.fit.batch_size}
31
32
33 callbacks:
34   model_name: ${model.name}
35   use_wd: ${compile.use_wd}
36   simple_compile: ${compile.simple_compile}
37   recon_freq: 10
38   checkpoints_path: ${paths.checkpoints}
39   logs_path: ${paths.logs}
40
41 model:
42   name: null
43   n_training_data: ${data.n_samples}
44   pos_enc_type: logspace
45   latent_space_dim: 128
46   pos_emb_length: 10
47   use_l2_reg: ${and:${not:${compile.use_wd}},${training.use_l2}}
48   mlp:
49     D: 8
50     W: 256
51     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

```
! lr_determination.yaml X
ifip > conf > grid_search > ! lr_determination.yaml
Zaccharie Ramzi, 2 months ago | 1 author (Zaccharie Ramzi)
1 # @package _global_ Zaccharie Ramzi
2 hydra:
3   sweeper:
4     params:
5       +compile: "\
6         {lr_latent:1e-3,lr_mlp:1e-5},\
7         {lr_latent:5e-4,lr_mlp:5e-4},\
8         {lr_latent:5e-4,lr_mlp:1e-5},\
9         {lr_latent:1e-5,lr_mlp:1e-5},\
10        {lr_latent:5e-4,lr_mlp:5e-6},\
11        {lr_latent:1e-5,lr_mlp:5e-6}\
12        "
```

What about W&B?

A bit more complex on JZ:

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

```
syncall_wandb.sh
```

```
1  #!/bin/bash
2
3  offline_runs="$1/offline-run*"
4  while :
5  do
6    ...for ofrun in $offline_runs
7    ...do
8      ....wandb sync $ofrun;
9    ...done
10   ...sleep 5m
11  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 ?
