

Practical Tips for Managing ML/DL Experiments

Iman Mirzadeh



Introduction



What is this talk about?

- Every research project, Has a a practical/development/engineering side.
- For our work, we need to implement and run our ideas and to check if they work!
- However, since we might not have engineering skills, we might not use software engineering disciplines or utilize the tools developed for our work.
- This might will come back and haunt us later!

What is this talk about?

- Many PhD students/researchers just want to have a working code and once they get the results, they think they are done!
- The reason? The way we work:
 - We start implementing a code and edit/debug the code until it works.
 - Once we get the result, we are done. No reason to continue.
- This is wrong:
 - You will work with this code later (rebuttal/when writing PhD dissertation, etc). You should be able to edit the code quickly and get new results.
 - You have a responsibility to publish your code so others can use.

What is this talk about?

- In this talk. We discuss:
 - What should we do during implementation/running/reporting phases?
 - Some guidelines/best practices for each phase
 - Tools that are helpful for our research.



Implementation Phase

What we do

- Write a working code locally.
- Debug until it works for a scenario.
- Backup: by copy & pasting.
- Clean the code, publish. Or worse, don't publish the code!

What we should do

- Design the pipeline/architecture of the code
- Write a CLEAN and MODULAR code.
- Version monitoring by git.
- You already have a publishable version!

Guidelines

- **Clean code:** can someone else understand your code? Or even yourself in a few months/years?
 - Checklist
 - No repeated code
 - Almost every function should be less than 10 lines
 - Is your code modular? Can you change your codebase so that you can run your method on new dataset in less than an hour?
- **Backup/version monitoring:**
 - Use git

Case study: A-GEM

<https://github.com/facebookresearch/agem/>



Running/Reporting Phase

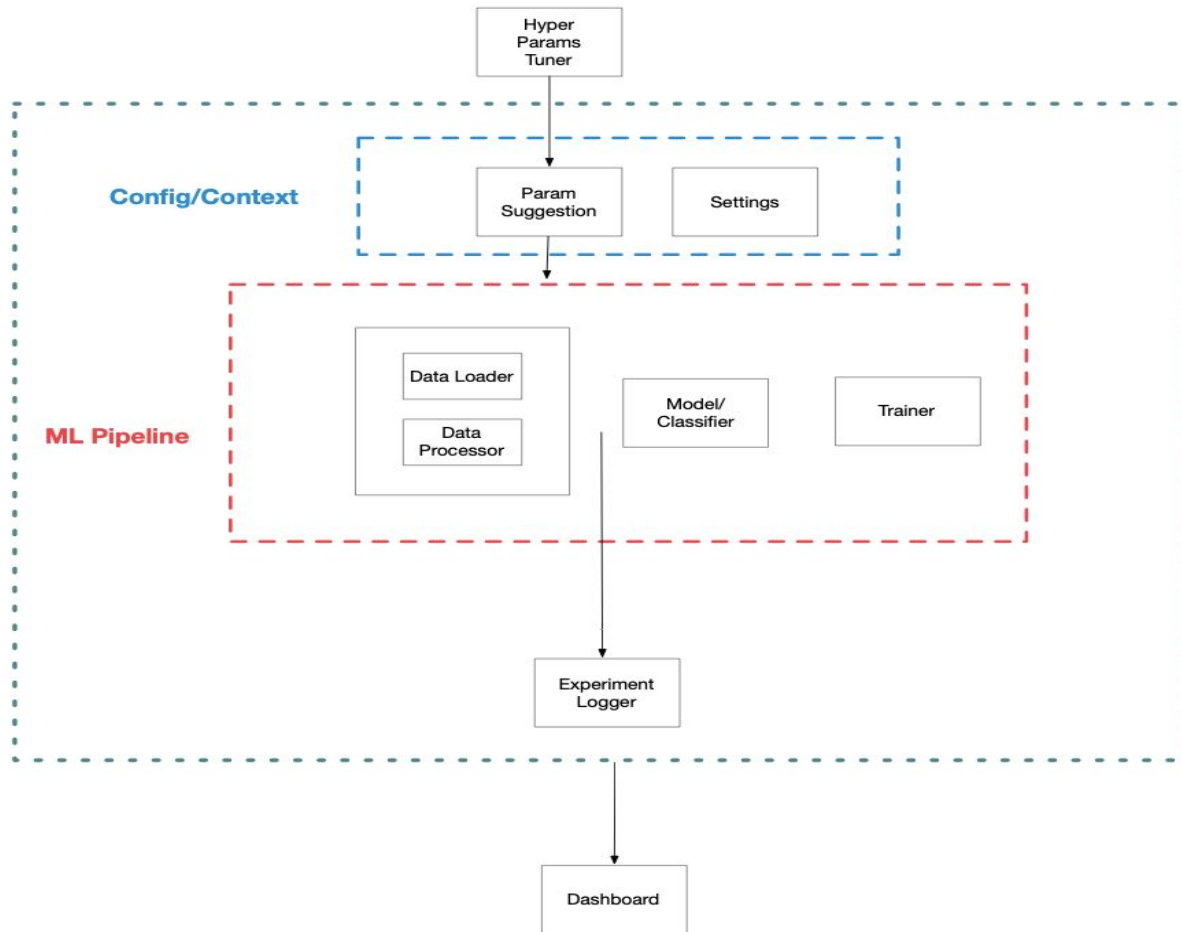
What we do

- Run with few hyper-params locally.
- Log the results in a csv sheet/file and calculate the metrics.

What we should do

- Use hyper-param tuner and run with various params on a cluster/gpu machine.
- Log the results using an experiment manager

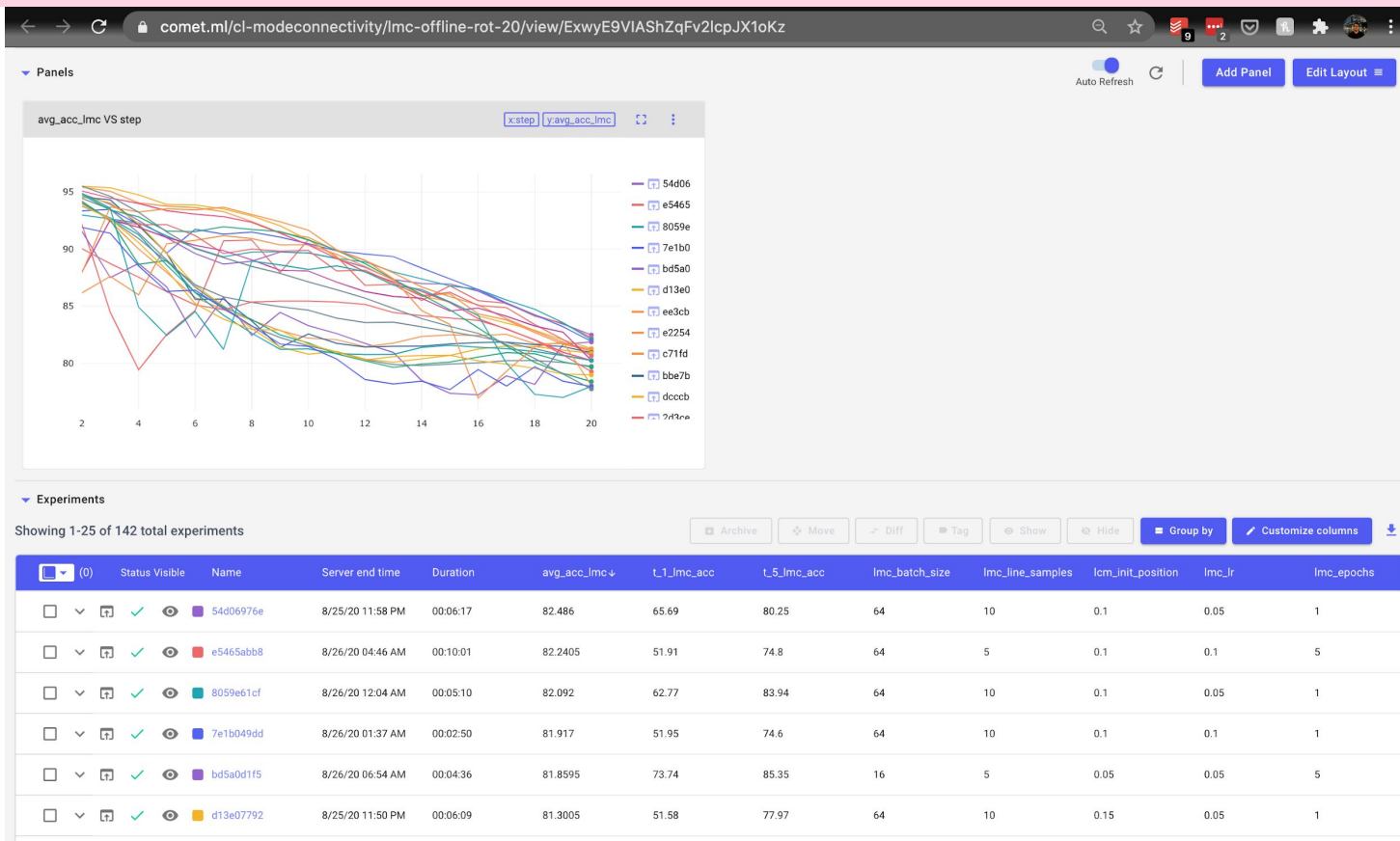
Experiment



Example: Hyper-param tuning

```
1  { "num_tasks": {"_type": "choice", "_value": [20]},
2    "memory_size": {"_type": "choice", "_value": [200]},
3    "momentum": {"_type": "choice", "_value": [0.8]},
4    "dropout": {"_type": "choice", "_value": [0.0, 0.25]},
5    "lr_decay": {"_type": "choice", "_value": [1.0, 0.7, 0.5]},
6
7    "seq_lr": {"_type": "choice", "_value": [0.1, 0.05, 0.01]},
8    "seq_batch_size": {"_type": "choice", "_value": [16, 32, 64]},
9    "seq_epochs": {"_type": "choice", "_value": [1]},
10
11    "lmc_lr": {"_type": "choice", "_value": [0.001, 0.005, 0.01]},
12    "lmc_batch_size": {"_type": "choice", "_value": [16, 32, 64]},
13    "lcm_init_position": {"_type": "choice", "_value": [0.25, 0.5, 0.75]},
14    "lmc_line_samples": {"_type": "choice", "_value": [5, 10]},
15    "lmc_epochs": {"_type": "choice", "_value": [1]}
16 }
```

Experiment management with comet.ml



Vast.ai for GPU machines

vast.ai



iman.kinder@gmail.com



Account

CLI

FAQ

CLIENT

Billing

Instances

Create

HOST

Dashboard

Machines

Create Job

Setup

EDIT IMAGE & CONFIG...

Disk Space To Allocate
15.76 GB

Filter offers

Availability

Host Reliability
90%

Max Instance Duration
1d 7h

☐ Include Unavailable Offers

☐ Include External Offers

☐ Include Unverified Machines

☐ Include Incompatible Machines

625605

2106

:



2x RTX 2080 Ti

36.6 TFLOPS
Max CUDA: 10.1

11.0 GB
477.3 GB/s

158B
PCIe 3.0, 16x 10.5 GB/s

Xeon® E5-2630 v2
24.0/24 cores 129/129 GB

KINGSTON
1382 MB/s 837.7 GB

↑23.4 Mbps
↓199.9 Mbps

38.0 DLPerf
98.6 DLP/\$/hr

Reliability
99.80%

\$0.385/hr

RENT

631555

125

:



1x RTX 2080 Ti

18.8 TFLOPS
Max CUDA: 10.1

11.0 GB
501.6 GB/s

EPYCD8
PCIe 3.0, 16x 12.4 GB/s

AMD EPYC 7251 ...
4.0/16 cores 16/64 GB

Storage
934 MB/s 174.7 GB

↑28.8 Mbps
↓88.0 Mbps

19.5 DLPerf
97.3 DLP/\$/hr

Reliability
99.85%

\$0.201/hr

RENT

627459

1817

:



4x RTX 2080 Ti

73.1 TFLOPS
Max CUDA: 10.1

11.0 GB
503.4 GB/s

Z10PE
PCIe 3.0, 16x 5.3 GB/s

Xeon® E5-2620 v3
24.0/24 cores 129/129 GB

Storage
1966 MB/s 1639.5 GB

↑36.4 Mbps
↓157.1 Mbps

77.3 DLPerf
96.3 DLP/\$/hr

Max Duration
1 mon, 21d

\$0.803/hr

RENT

621631

1453

:



2x RTX 2080 Ti

36.6 TFLOPS
Max CUDA: 10.1

11.0 GB
502.5 GB/s

Z10PE
PCIe 3.0, 16x 5.0 GB/s

Xeon® E5-2620 v3
12.0/24 cores 64/129 GB

Storage
681 MB/s 638.0 GB

↑17.1 Mbps
↓65.1 Mbps

38.9 DLPerf
96.6 DLP/\$/hr

Max Duration
1 mon, 11d

\$0.403/hr

RENT



Final Tips

Final Tips

- Every hyper-param/setting should be stored in a global object!
 - Learning rate, optimizer, batch size, dropout rate
 - Number of hidden layers, number of epochs
- Every step and every result, should be stored online!
 - Model weights/checkpoints
 - The config
 - Metrics
- Basically, you shouldn't lose anything if someone crushes your laptop with a hammer right now!

Thank You!

Contact: seyediman.mirzadeh@wsu.edu

