Practical Tips for Managing ML/DL Experiments

Iman Mirzadeh

Disclaimer: The master theme for this presentation is borrowed and edited from DM theme.
Introduction
What is this talk about?

- Every research project, has 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.
Example: Hyper-param tuning

```json
{
  "num_tasks": {
    "_type": "choice",
    "_value": [20]
  },
  "memory_size": {
    "_type": "choice",
    "_value": [200]
  },
  "momentum": {
    "_type": "choice",
    "_value": [0.8]
  },
  "dropout": {
    "_type": "choice",
    "_value": [0.0, 0.25]
  },
  "lr_decay": {
    "_type": "choice",
    "_value": [1.0, 0.7, 0.5]
  },
  "seq_lr": {
    "_type": "choice",
    "_value": [0.1, 0.05, 0.01]
  },
  "seq_batch_size": {
    "_type": "choice",
    "_value": [16, 32, 64]
  },
  "seq_epochs": {
    "_type": "choice",
    "_value": [1]
  },
  "lmc_lr": {
    "_type": "choice",
    "_value": [0.001, 0.005, 0.01]
  },
  "lmc_batch_size": {
    "_type": "choice",
    "_value": [16, 32, 64]
  },
  "lcm_init_position": {
    "_type": "choice",
    "_value": [0.25, 0.5, 0.75]
  },
  "lmc_line_samples": {
    "_type": "choice",
    "_value": [5, 10]
  },
  "lmc_epochs": {
    "_type": "choice",
    "_value": [1]
  }
}
```
<table>
<thead>
<tr>
<th>ID</th>
<th>Type</th>
<th>GPU Model</th>
<th>CPU Model</th>
<th>Memory (GB)</th>
<th>Bandwidth</th>
<th>Price/hr</th>
</tr>
</thead>
<tbody>
<tr>
<td>625605</td>
<td>2x RTX 2080 Ti</td>
<td>36.6 TFLOPS Max CUDA: 10.1</td>
<td>Xeon® E5-2630 v2 24/24 cores 120/120 GB</td>
<td>11.0</td>
<td>23.4 Mbps</td>
<td>$0.385/hr</td>
</tr>
<tr>
<td>627459</td>
<td>4x RTX 2080 Ti</td>
<td>73.1 TFLOPS Max CUDA: 10.1</td>
<td>Xeon® E5-2520 v3 24/24 cores 120/120 GB</td>
<td>11.0</td>
<td>36.4 Mbps</td>
<td>$0.803/hr</td>
</tr>
<tr>
<td>621631</td>
<td>2x RTX 2080 Ti</td>
<td>36.6 TFLOPS Max CUDA: 10.1</td>
<td>Xeon® E5-2520 v3 12/24 cores 64/129 GB</td>
<td>11.0</td>
<td>17.1 Mbps</td>
<td>$0.403/hr</td>
</tr>
</tbody>
</table>
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