GPU usage for ML at CC-IN2P3

Introduction

Goal of this presentation :

- Discuss practical usage of GPU for ML in ATLAS
- Overview of many aspects/difficulties in ML projects
  - environements setup
  - libraries
  - file format & processing
  - application in Athena
- NOT meant to be a best-practice recommendation
  - other usage / workflow might be better
  - personnal choice/preference/experience in purple
  - discussions welcome !

Environment – Jupyter or scripts ?

Jupyter notebook

- Python development interface in a web browser
- Used in most (all?) ML tutorials
- Many advantages/nice features
  - integrated graphics, equation, markdown doc..
  - easily shareable
  - etc…
- Available at CC with GPUs

Or use **scripts & python command line**

- Intense usage of command line (history, shortcuts, auto-completion)
- Jupyter editor limited (or I'm not used to it)
- any important project will anyway need to
  - develop full libraries beyond simple notebook/scripts
  - write elaborated batch scripts

Environment - facilities

CC-IN2P3

- GPU farm available to all ATLAS users
- Batch: jobs submitted with SLURM (same as CPU)
- Interactive: `srun -p gpu_interactive … (see doc)`
  - necessary to develop, debug, prototype
- Custom software environment possible
  - details next slides
- Excellent, direct support from CC !
  - the issue tracking system is an issue though...
- Batch slots availability : usually good… but sometimes completely full !
  - personal : jobs stuck in queue since several days at critical period => decided to ask for an account at IDRIS/JeanZay

Environment - facilities

IDRIS Jean Zay

- Need to submit a project to have an account
  - not difficult for small project : ~1 page description+signature from DU
  - took a few days to obtain an acount + 5000h GPU
- Slurm system for batch/interactive jobs
  - options/settings might differ
- Disk quota : ~50Tb
- Reactive support
- Batch slots availability: excellent, jobs never queued more than a few mins.

Environment – setting up ML libraries

CC-IN2P3

- load conda then activate an environment
- Or prepare a custom environment

```
module load conda
conda activate <environment name>
```

```
conda env create -f ~/myEnvConfig.yaml -p /pbs/throng/atlas/delsart/myenv
# activate with :
conda activate /pbs/throng/atlas/delsart/myenv
```

- Install additional packages ?
  - use "conda install …"
  - … or "pip install" (works well but might conflict with conda packages)

JeanZay

- Use "module load <env>" + "pip install"

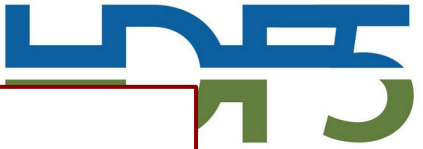
Data preparation for ML

- Typical case: how to extract data from DAOD to feed ML training algorithms ?
  - Athena job to write ROOT ntuple or HDF5 file
  - Many Athena frameworks exist, ex: easyjet

- Which format to use ?
  - HDF5 : popular in ML community, many tools and interface with ML libraries
  - ROOT: familiar to HEP users, can be read in pure python with uproot (very easy)

Data preparation for ML

- Typical case: how to extract data from DAOD to feed ML training algorithms ?
  - Athena job to write ROOT ntuple or HDF5 file
  - N
- Wh
  - h
    v
  - F
    uproot (very easy)



Note on ROOT's future :

- **RNTupe** will replace TTree: smaller and faster to read than all other format
- v6.34 will provide solution to feed pytorch tensorflow directly from **RDataFrame**
  - potentially perfect solution for loading and preprocessing !

ce

with

ML coding : Tensorflow or pyTorch ?

- Both very mature & performant frameworks
  - No obvious choice...
- Wrote the same project with both tensorflow/keras and pytorch
- preferred pyTorch
  - easier to understand the training mechanics (keras simplicity is "hiding" many details), more flexible (tensors are mutable)
  - lower memory usage in my case
  - seem more popular in academia (more package based on pytorch)

Data flow and preprocessing

Can be a very important and time consuming development

- Large datasets do not all fit in GPU memory
- Need to stream from file to GPU
  - includes loading, filtering, preprocessing
- Wrote custom code to read-in ROOT ntuple
  - using uproot to load TTree from files
  - Did not find good data streaming solution for HDF5 files ??

Input (output) normalization suggestion:

- Do it as the 1st layer of your model
  - normalization code won't have to be ported when using the model in an other environment (Athena….)

Model architecture and development

Architecture

- Very project specific
  - found pyTorch extremely flexible & easy for complex NN architecture

Developing & debugging

- Use python interactive sessions a lot
  - auto-completion + history
  - inspect objects, quick tests…
- Hyper parameter optimization
  - possible if clear criteria for stopping training exist…
  - wasn't my case → couldn't test !

Common issues during training

- NaN or Inf
  - the "gradient descent" used in training can easily produce numerical errors
  - ruins and stops the training
  - Often due to 1/x of √x operations (ex: inside loss function)
    - protect with $x \rightarrow x+\varepsilon$ (where ε=small constant)
- memory issues
  - memory usage grows fast with model size : can exhaust memory during "back-propagation"
  - solutions:
    - reduce batch size
    - request higher memory jobs (SLURM option)

Validation of model prediction

- Training and validation loss are often not enough
- Other metrics might be long to evaluate
  - plan in advance when designing data flow & preprocessing
  - sometimes it imply going back to Athena
    - very slow validation !

Other points

- Train on multiple GPU for very complex model
  - not tested
  - was problematic at CC: are issues solved ?
- Choice of loss function
  - Mathematically determines what the model learns → make sure it is appropriate
- Input feature importance

Conclusions

Completing a large ML project involve lot of time and efforts

- Imply understanding many different technical aspects
- Similar as completing a physics analysis

Many workflows/approaches/tools are possible

- Finding the appropriate ones is not obvious
- ...but can help to save a lot of time !
- discussing & sharing info is important