Track 2 on Domain-specialized TinyLM

- Topic: Make Small LM better as a domain expert 👩‍🍳
  - Considerations: *Efficiency* & *in-Domain Performance*
- How to approach? 🤔
  - Basic training script ready to use ☑️
  - Sufficient GPUs for experiments (100 GPU hours free per team) ☑️
- What can we do?
  - almost everything sparkling idea is allowed 🤗 *(minus some caveats on next slide)*

🌟 Change the neural network architecture (number of layers, LORA, etc.)

🌟 Better data scheme (selection, mixture)

🌟 Balance resource allocation for pretrain and fine-tuning

🌟 Better optimizer ......

EPFL

Competition Rules

- Resources & Constraints
  - Training codebase: https://github.com/epfml/llm-baselines
  - Training data: ***Must*** *use only the dataset and tokenizer we provide*
    - **SlimPajama**: 6B tokens, fixed data seed, fixed sequence length for evaluation set
    - **MathQA** train split: in-domain used for fine-tuning
    - ⚠️ *External data or extra models are not allowed*
  - GPU Resources: 18x A100 40GB GPUs + 10GB storage per team
- Evaluation criterion
  - **Validation loss** & **5-shot Accuracy** on a held-out set of **Math Reasoning** after **4h** on 1 A100 40GB GPU. See provided example code - 15pts
  - **Innovativeness** of your approach (as described in code readme, by sunday midnight) - 15pts

→ innovation is as important as pure scores! :)

EPFL

Competition Rules

- Demo
  - Saturday ~3:30pm: Present 3min on the ideas you're trying
- Deliverables
  - **Only Code** with bash script for training. Do not submit any models checkpoints and datasets.
  - If you work on data selection: submit **script used to select data**
- Final submission
  - Deadline: Saturday 3:30pm (before Demo)
  - submitting the link to your github repository. The README of your repo needs:
    1) a brief and clear description of your idea, approach and final results
    2) a link to the final model weights of your trained model
    3) the **training script** and **config** we can use to reproduce your results
  - **we will verify submissions and run to check reproducibility**

Example Ideas

- **Architecture**
  - Transformer architecture (width, layers, ...)
  - Mixture of Experts (MoE)
  - LoRA for fine-tuning ...
- **Optimization**
  - Training hyperparameters (learning rate, ...)
  - Different optimizers
  - Low precision training
- **Training Scheme**
  - Optimal resource allocation on Pretraining and Fine-tuning (e.g. 1h pretrain+2h FT)
- **Data Selection**
  - Better mixture of general pretraining data (SlimPajama) and in-domain data (MathQA)
  - Select a subset with higher quality
  - From heuristics
- ....

Agenda

(Fri Apr 19, 6pm): Tutorials/workshops

(Fri Apr 19, 10pm): end of day (can continue working at home)

(Sat Apr 20, 10am): breakfast

(Sat Apr 20, till 3:30pm): hack, hack, hack

(Sat Apr 20, 4pm): demos of both tracks

Sun Apr 21, midnight: **final submissions** for LLM architectures track #2

Tue Apr 23, 4pm: results announced of LLM architectures track #2

Overview: GPU Cluster and LLM-Baselines

→ How to set up your EPFL cluster access

- 1 access per group (needs EPFL Gaspar)
  - Collaborate and experiment with GPUs together
- Main concept:
  - “pods” that reserve you a GPU that you can connect to and run code

→ How to get started on the baselines code for training a TinyLM

- Create a fork of our modular LLM framework *llm-baselines*
- Run a simple Python script and track the results on *Weights and Biases*

Tutorial: GPU on RCP Cluster

How to set up your EPFL cluster access

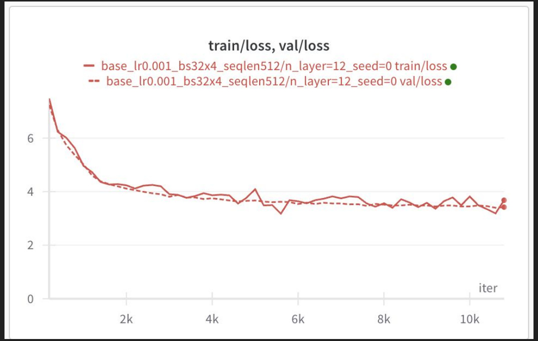
- We provide a step-by-step guide: https://drive.google.com/drive/folders/1lQ1geB2Vge7GKJ8iQedwMSHmWmcRQwWp?usp=drive_link
- The experiments will be easily launched on the web-browser :)

Setup Account → launch jupyter → develop and run python

Tutorial: LLM-Baselines

How to get started on the baselines code for training a TinyLM

- With the guide we provide, as simple as:
  - Create your fork of https://github.com/epfml/llm-baselines (also needed for submission)
  - On the cluster, run python src/main.py
  - Track your experiments by WanDB
  - Easy start: play with some provided parameters (see README), e.g.




```
parser.add_argument('--n_head', default=12, type=int)
parser.add_argument('--n_layer', default=12, type=int) # depth in (att + ff) blocks
parser.add_argument('--n_embd', default=768, type=int) # hidden size ...
```

```
parser.add_argument('--lr', default=1e-3, type=float)
parser.add_argument('--warmup_percent', default=0.05, type=float) # the total number of warmup
parser.add_argument('--weight_decay', default=0.1, type=float) # I recommend you keep this val
parser.add_argument('--beta1', default=0.9, type=float) # adam parameter
parser.add_argument('--beta2', default=0.95, type=float) # adam parameter
parser.add_argument('--scheduler', default='cos', choices=['linear', 'cos', 'none'])
parser.add_argument('--opt', default='adamw', choices=['adamw', 'sgd'])
```