Lab 7 AI benchmark

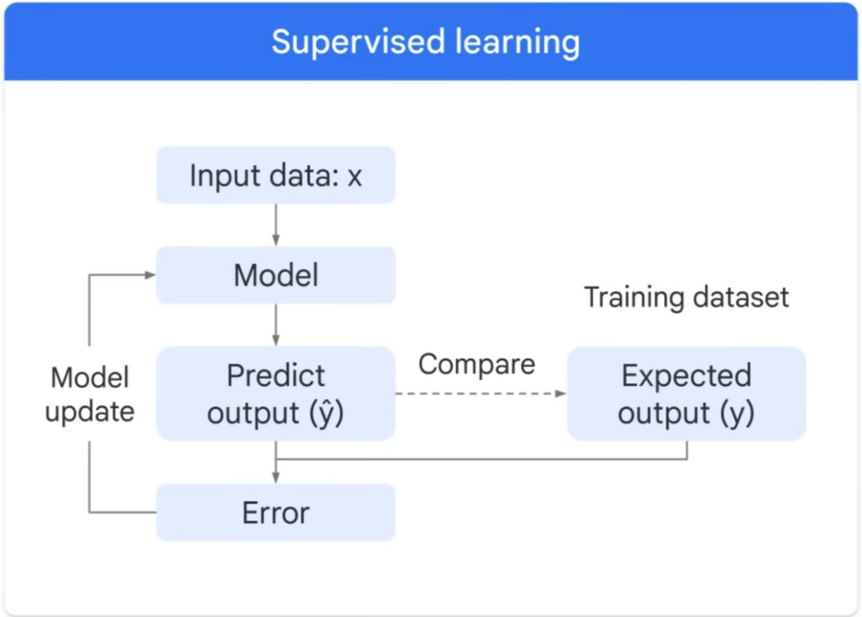
Dec, 2023 Parallel Programming

Overview

- AI recap
- Parallelization techniques
- Lab 7

AI recap

Supervised and unsupervised learning



Supervised learning

Input data: x

Model

Training dataset

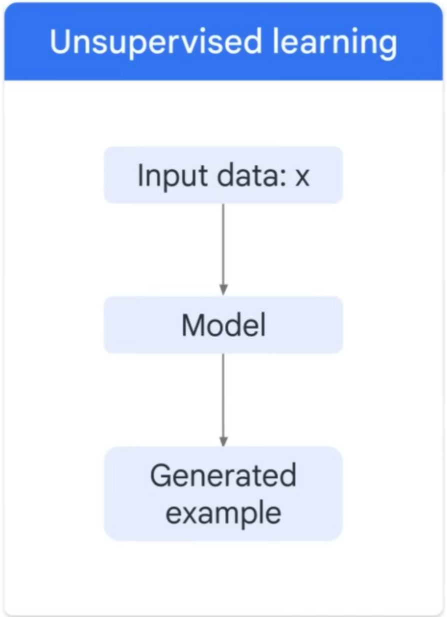
Model update

Predict output (ŷ)

Compare

Expected output (y)

Error



Unsupervised learning

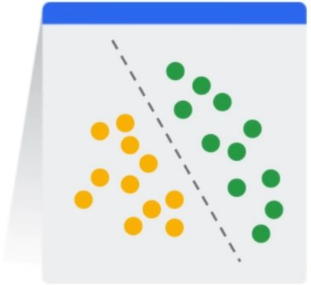
Input data: x

Model

Generated example

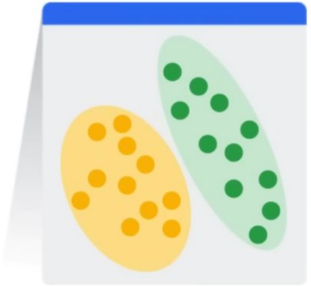
Generative AI

Deep Learning Model Types



Discriminative

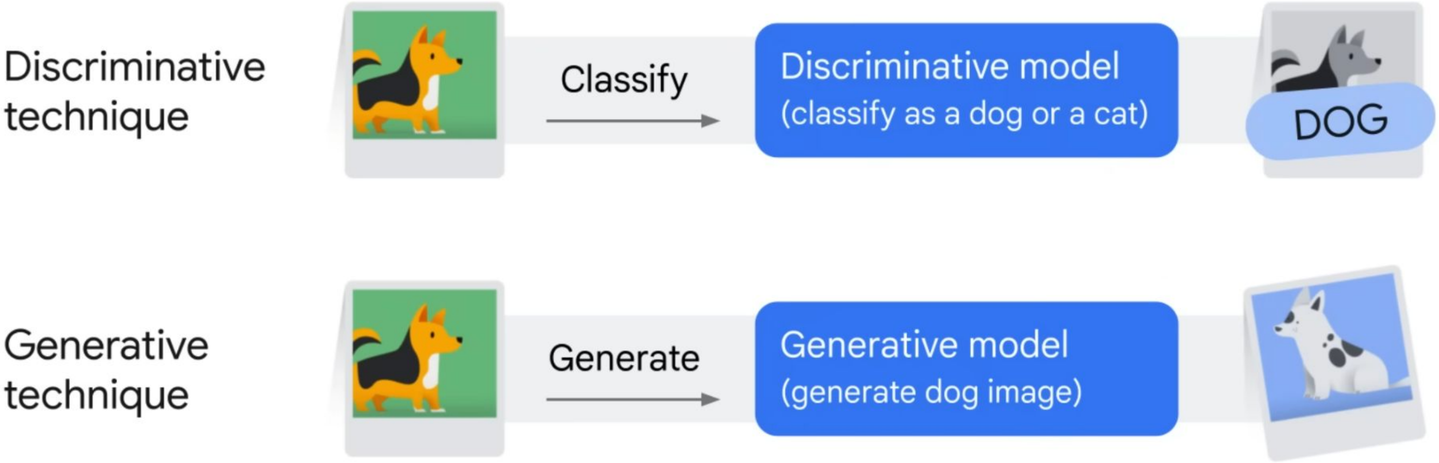
- Used to classify or predict
- Typically trained on a dataset of labeled data
- Learns the relationship between the features of the data points and the labels



Generative

- Generates new data that is similar to data it was trained on
- Understands distribution of data and how likely a given example is
- Predict next word in a sequence

Generative AI



Discriminative technique

Classify

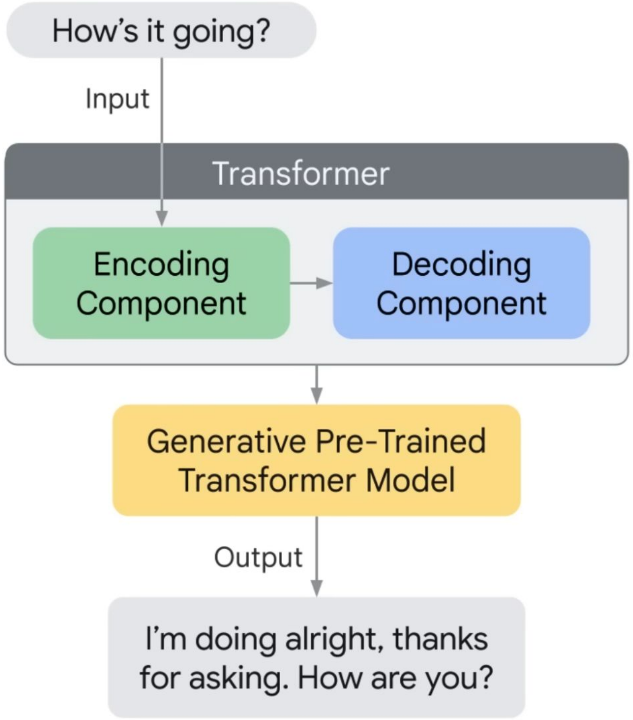
Discriminative model
(classify as a dog or a cat)

Generative technique

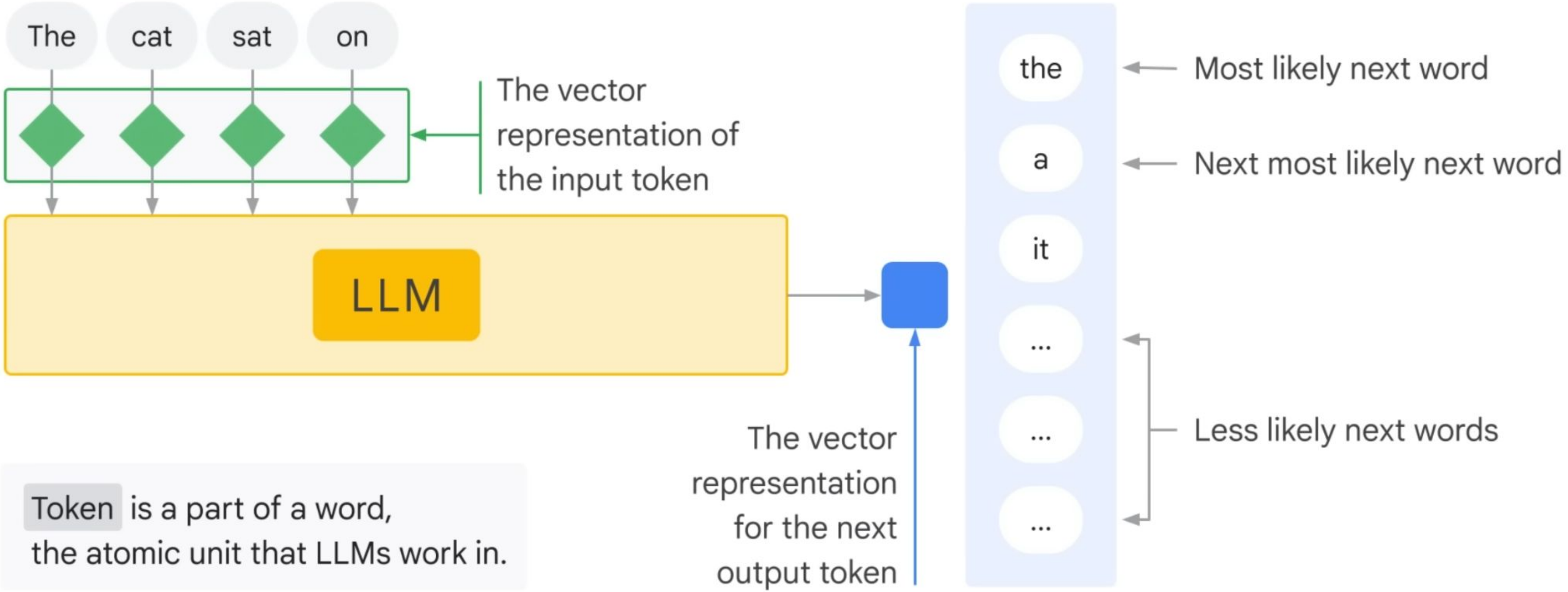
Generate

Generative model
(generate dog image)

Transformer


How's it going?
Input
Transformer
Encoding Component
Decoding Component
Generative Pre-Trained Transformer Model
Output
I'm doing alright, thanks for asking. How are you?

How to train Large Language Model(LLM) from scratch?


The
cat
sat
on
The vector representation of the input token
LLM
The vector representation for the next output token
the
a
it
...
...
...
Most likely next word
Next most likely next word
Less likely next words

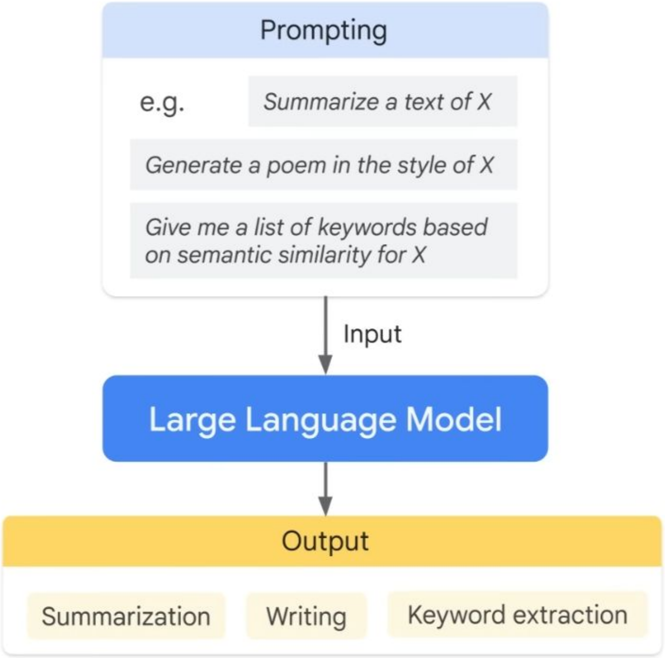
Token is a part of a word,
the atomic unit that LLMs work in.

Pretrained Language Model(PLM)

To train a LLM we need

- Large amount of Data
- Billions of parameters

Training a LLM from scratch is very expensive
Thus, we normally “finetune” a
Pretrained Language Model

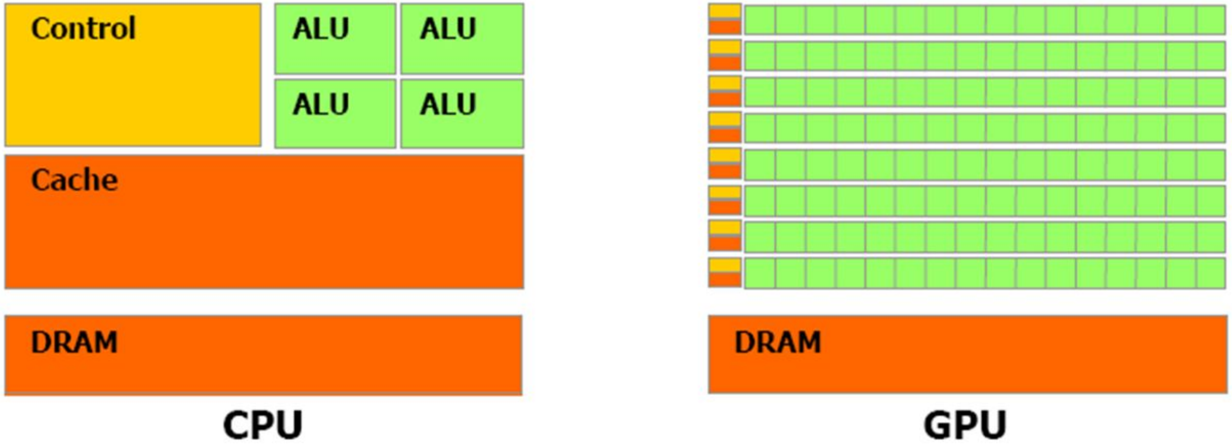


Parallelization Techniques

Run model on GPU!

Machine learning is mainly comprised by linear algebra arithmatics

GPUs have a much more stronger vector processing capability!

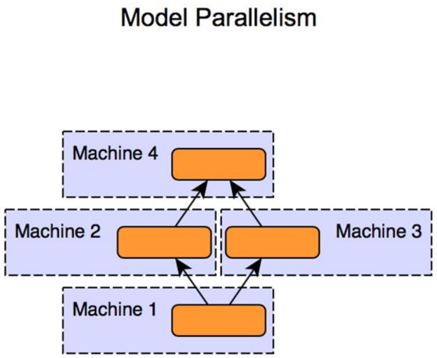


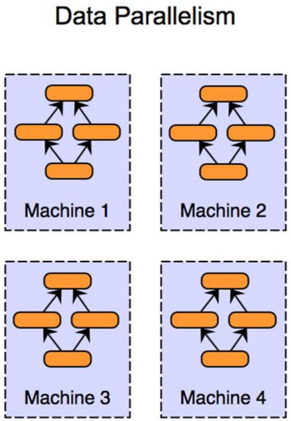
DEMO

Distributed Training

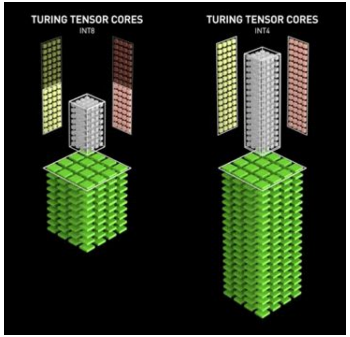
If model is too big, we do model parallelism!

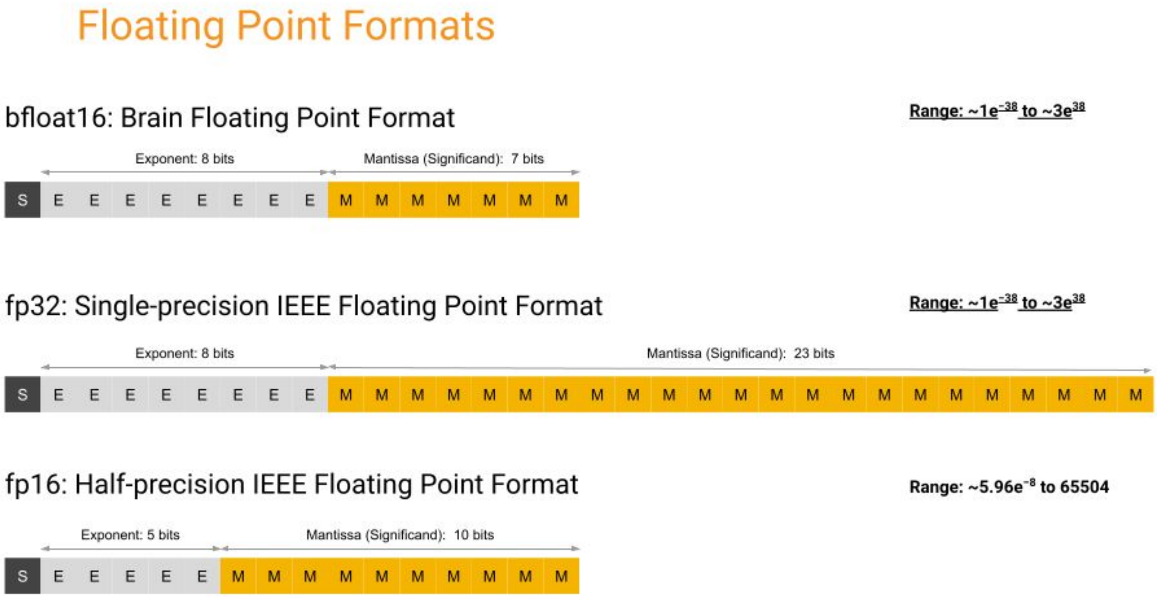
If data is too much, we do data parallelism!





Lower Precision





Floating Point Formats

bfloat16: Brain Floating Point Format

Range: ~1e^{-38} to ~3e^{38}

Exponent: 8 bits

Mantissa (Significand): 7 bits

S E E E E E E E E M M M M M M M

fp32: Single-precision IEEE Floating Point Format

Range: ~1e^{-38} to ~3e^{38}

Exponent: 8 bits

Mantissa (Significand): 23 bits

S E E E E E E E E M M M M M M M M M M M M M M M M M M M M M M M

fp16: Half-precision IEEE Floating Point Format

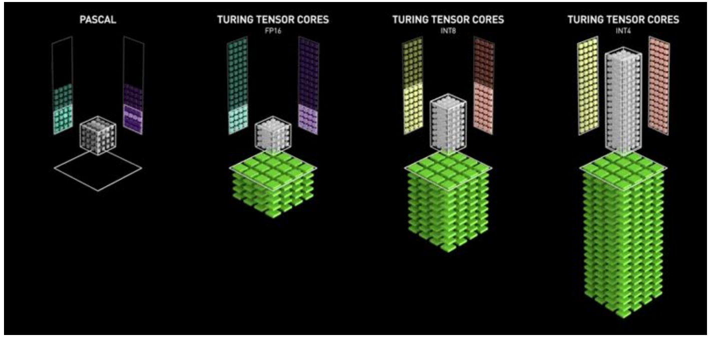
Range: ~5.96e^{-8} to 65504

Exponent: 5 bits

Mantissa (Significand): 10 bits

S E E E E E M M M M M M M M M M

Leverage the hardware



Tensor Core

For training

PyTorch

TensorFlow

For inference

TensorRT

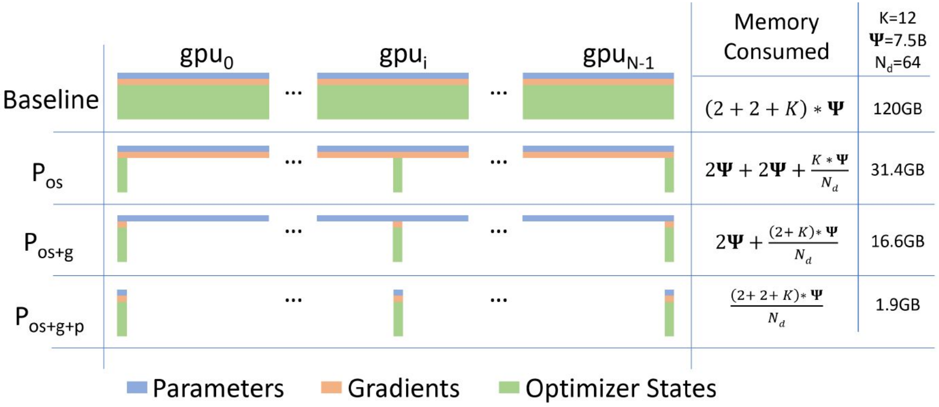
Deepspeed

ZeRO-DP optimize the memory consumption of model state by partition states on to devices

Stage 1

Stage 2

Stage 3



	Memory Consumed	K=12 Ψ=7.5B N_d=64
Baseline	$(2+2+K)*\Psi$	120GB
P_{os}	$2\Psi + 2\Psi + \frac{K*\Psi}{N_d}$	31.4GB
P_{os+g}	$2\Psi + \frac{(2+K)*\Psi}{N_d}$	16.6GB
P_{os+g+p}	$\frac{(2+2+K)*\Psi}{N_d}$	1.9GB

Figure 1: Comparing the per-device memory consumption of model states, with three stages of *ZeRO*-DP optimizations. Ψ denotes model size (number of parameters), K denotes the memory multiplier of optimizer states, and N_d denotes DP degree. In the example, we assume a model size of $\Psi = 7.5B$ and DP of $N_d = 64$ with $K = 12$ based on mixed-precision training with Adam optimizer.

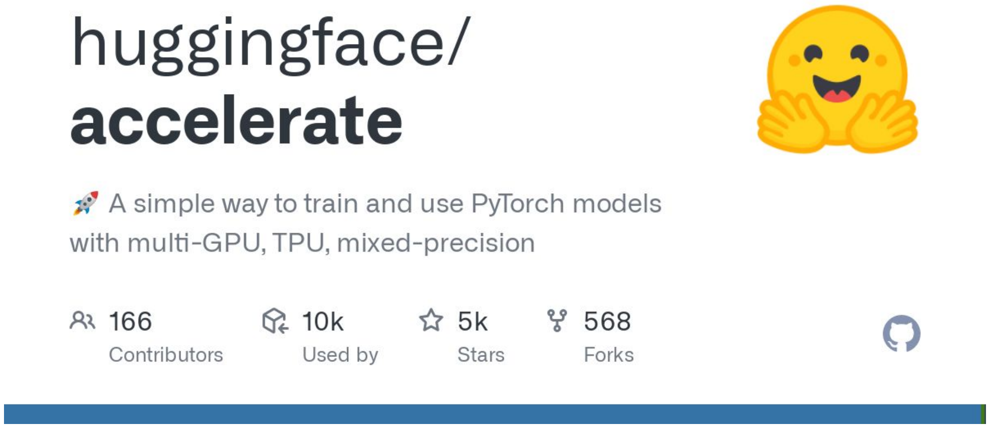
How to use those acceleration?

Most of frameworks already have them!

Just search “[frameworkname] [keyword]”!

PyTorch

TensorFlow



huggingface/
accelerate

🚀 A simple way to train and use PyTorch models with multi-GPU, TPU, mixed-precision

166 Contributors

10k Used by

5k Stars

568 Forks

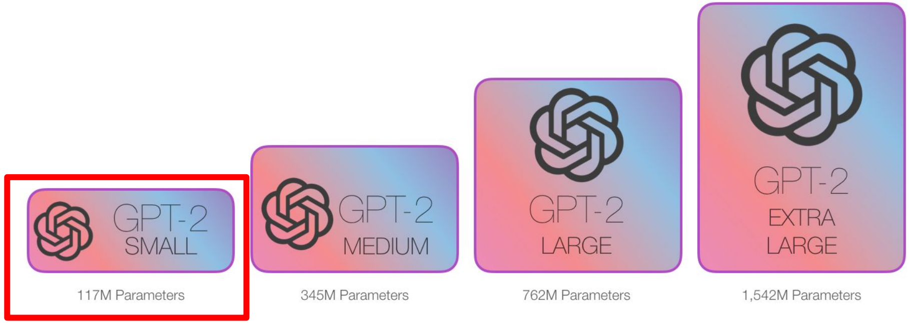
Lab 7 Assignment

(On hades)

Introduction

GPT-2 is a transformers model pretrained on a very large corpus of English data in a self-supervised fashion. This means it was pretrained on the raw texts only, with no humans labeling them in any way!

- It was revealed by openAI in 2019 and is the second in their foundational series of GPT models.
- It's the ancestor of ChatGPT!
- You are going to experience the process of fine-tuning the LLM model with the smallest version of GPT-2, which has 124M parameters (about ~1400x smaller than chatGPT).



Instructions

- Draw the strong scalability of data parallel training and explain why such observation and your experiment process (in less than 1 page)
  - Scale: from 1 GPU to 2 GPUs
  - Use “train_samples_per_second” as performance metric

- TAs provided sample scripts for data parallel training
  - Files are located at `/home/pp23/share/lab7`
  - `*.sh` are scripts for your experiment
  - `run_clm.py` is the python training script
  - `try_out.py` is the script to interact with the model

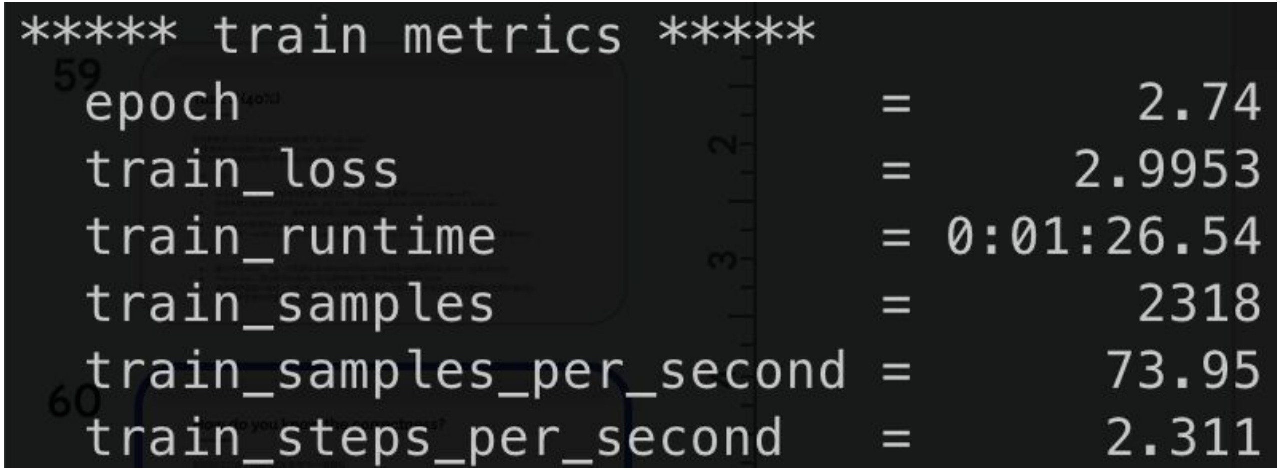
How to run?

- Normally, this script will take about 2 minutes to complete training
- Modify `--nproc_per_node` in the scripts according to the number of GPUs
- hades02 (script in `hades_2` folder)
  - bash run_DDP.sh

- hades[03-07] (scripts in `slurm_hades` folder)
  - sbatch run_DDP_{n}GPU.sh, {n} is the number of GPUs
  - Modify `--gres=gpu:{n}` in the script to allocate {n} GPUs
  - Check job status by `squeue -u $USER`
  - The scripts will dump
    stdout to [jobid].out, stderr to [jobid].err

How do you check the correctness?

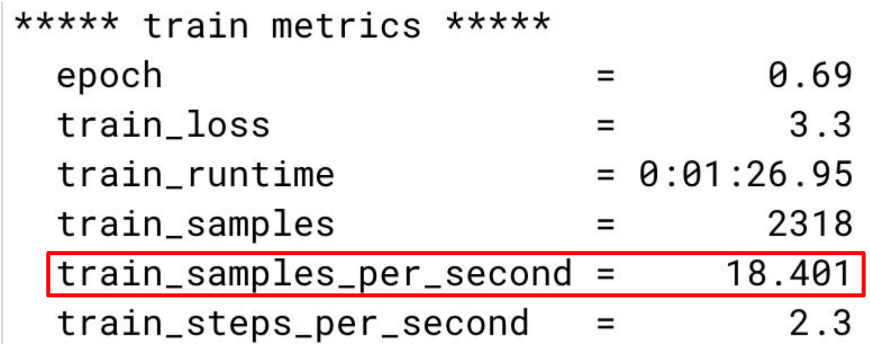
If the training completes successfully,
the following train metrics can be found at the end stdout.



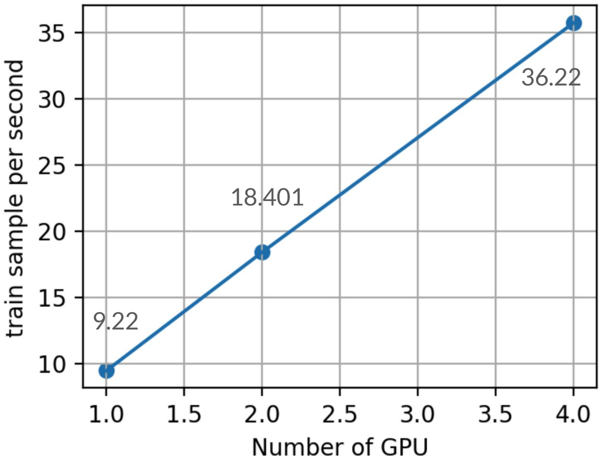
```
***** train metrics *****
  epoch                    =        2.74
  train_loss               =      2.9953
  train_runtime            = 0:01:26.54
  train_samples            =        2318
  train_samples_per_second =       73.95
  train_steps_per_second   =       2.311
```

Strong Scalability

You can use any tool to draw the strong scalability (even with pencil and paper)
x for `Number of GPUs`; y for `train_samples_per_second`



```
***** train metrics *****
  epoch                    =       0.69
  train_loss               =        3.3
  train_runtime            = 0:01:26.95
  train_samples            =       2318
  train_samples_per_second =     18.401
  train_steps_per_second   =        2.3
```

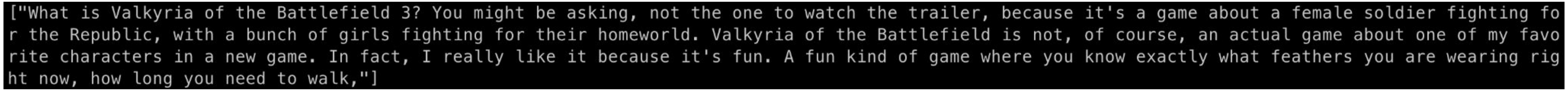


How to play with the model?

Use the provided “try_out.py” to have fun with your model.

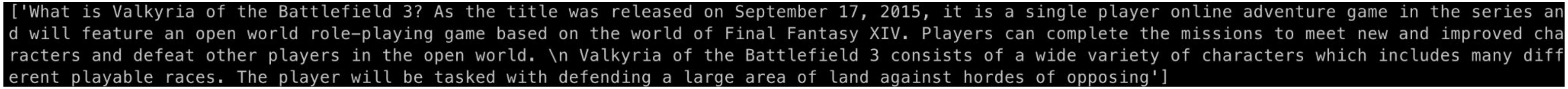
```
srun -n 1 --gres=gpu:1 python try_out.py
```

append `-m ${HOME}/GPT_DDP_weights` to assign finetuned model path
append `-p “[question]”` to ask different question to the model (don’t remove ”)



```
["What is Valkyria of the Battlefield 3? You might be asking, not the one to watch the trailer, because it's a game about a female soldier fighting fo
r the Republic, with a bunch of girls fighting for their homeworld. Valkyria of the Battlefield is not, of course, an actual game about one of my favo
rite characters in a new game. In fact, I really like it because it's fun. A fun kind of game where you know exactly what feathers you are wearing rig
ht now, how long you need to walk,"]
```

Completely nonsense
(No finetune)



```
['What is Valkyria of the Battlefield 3? As the title was released on September 17, 2015, it is a single player online adventure game in the series an
d will feature an open world role-playing game based on the world of Final Fantasy XIV. Players can complete the missions to meet new and improved cha
racters and defeat other players in the open world. \n Valkyria of the Battlefield 3 consists of a wide variety of characters which includes many diff
erent playable races. The player will be tasked with defending a large area of land against hordes of opposing']
```

More or less meaningful
(Finetuned)

Submission

- Submit your report(pdf), logs(err+out), scripts(if any modification) to eeclass before 1/4 23:59
- Your report should includes
  - Draw the strong scalability of data parallel training
  - Explain why such observation
  - Your experiment process
- Get started as soon as possible to avoid heavy queueing delay

Have fun!

Appendix

Q&A

Q:

I encountered "RuntimeError: The server socket has failed to listen on any local network address. The server socket has failed to bind to [::]:38788", what should I do?

A:

Reason: In very rare case, the port used for intercommunication might collided with someone other running on the same machine.

Solution: Wait s few seconds, try again!

Q&A

Q: What is the loss function？

A: negative log likelihood

Q: Why there’s nothing pop out after I type in `python run_clm.py -h`, TA lied us!?

A: No, wait few more seconds. Be patient!

Q&A

Q: How do I draw strong scalability

A:

1. Run training in 1 GPU and 2 GPUs

2. Check train_samples_per_second

3. Make a line plot with

   x for `Number of GPUs`; y for `train_samples_per_second`