

AI 大模型开发工程师之 大模型微调核心之微调实战

讲师：李希沅

目录

- 1 各大微调技术应用场景对比
- 2 微调实战之基于LoRA微调Llama2
- 3 微调实战之基于P-Tuning V2微调ChatGLM3
- 4 微调实战之基于多卡方案微调ChatGLM3
- 5 基于微调升级企业知识库

① 各大微调技术应用场景对比

01、各大模型对比

Causal Language Modeling

Model	LoRA	Prefix Tuning	P-Tuning	Prompt Tuning	IA3
GPT-2	✓	✓	✓	✓	✓
Bloom	✓	✓	✓	✓	✓
OPT	✓	✓	✓	✓	✓
GPT-Neo	✓	✓	✓	✓	✓
GPT-J	✓	✓	✓	✓	✓
GPT-NeoX-20B	✓	✓	✓	✓	✓
LLaMA	✓	✓	✓	✓	✓
ChatGLM	✓	✓	✓	✓	✓
Mistral	✓				

Causal Language Modeling（因果语言模型）**主要的任务是根据当前的上下文预测下一个单词**。它是一种生成模型，能够生成类似人类的文本。因果语言模型在处理序列数据时，会考虑前面的上下文信息，但是不会看到未来的信息。

例如，给定一句话的前半部分 "The quick brown fox jumps over ...", 因果语言模型的任务是预测接下来可能出现的词，比如 "the"。这个预测是基于给定上下文的条件概率分布来进行的。

这种模型在很多应用中都非常有用，比如**机器翻译、语音识别和文本生成**等。

Conditional Generation

Model	LoRA	Prefix Tuning	P-Tuning	Prompt Tuning	IA3
T5	✓	✓	✓	✓	✓
BART	✓	✓	✓	✓	✓

Conditional Generation（条件生成）的模型，主要是在**给定一定条件或上下文的情况下，生成特定的输出**。在自然语言处理中，条件生成模型常常用来生成符合特定条件的文本。

以**机器翻译**为例，给定一个源语言（例如英语）的句子，条件生成模型的任务就是生成目标语言（例如中文）的句子。这里的“条件”就是源语言的句子，而生成的目标就是目标语言的句子。

又如**文本摘要**任务，给定一个长篇文章，条件生成模型的任务就是生成该文章的摘要。这里的“条件”就是原始的长篇文章，生成的目标就是简短的摘要。

条件生成模型的一个关键特性是它能够根据不同的输入条件生成不同的输出，因此它在许多需要个性化输出的场景中有广泛的应用，如推荐系统、个性化新闻生成等。

02、各大模型对比

Sequence Classification

Model	LoRA	Prefix Tuning	P-Tuning	Prompt Tuning	IA3
BERT	✓	✓	✓	✓	✓
RoBERTa	✓	✓	✓	✓	✓
GPT-2	✓	✓	✓	✓	
Bloom	✓	✓	✓	✓	
OPT	✓	✓	✓	✓	
GPT-Neo	✓	✓	✓	✓	
GPT-J	✓	✓	✓	✓	
Deberta	✓		✓	✓	
Deberta-v2	✓		✓	✓	

Sequence Classification（序列分类）模型的主要任务是**对整个序列进行分类**。在这个上下文中，序列可以是一系列的单词（文本数据）、声音信号、时间序列数据等。模型的目标是将输入的序列映射到预定义的类别标签上。

一个典型的例子是情感分析任务。在这个任务中，给定一个文本序列（如一句话或一段评论），序列分类模型会判断这段文本的情感倾向，比如将其分类为“正面”、“负面”或“中立”。这里的序列就是文本中的单词序列，而分类的目标是情感倾向的类别。

其他序列分类的例子还包括：

垃圾邮件检测：将邮件序列分类为“垃圾邮件”或“非垃圾邮件”。

文本主题分类：将新闻文章、科学论文或其他文档分类到预设的主题类别，如“体育”、“政治”、“科技”等。

蛋白质功能分类：将蛋白质序列根据其生物学功能分类到不同的功能类别中。

Text-to-Image Generation

Model	LoRA	LoHa	LoKr	OFT	Prefix Tuning	P-Tuning	Prompt Tuning	IA3
Stable Diffusion	✓	✓	✓	✓				

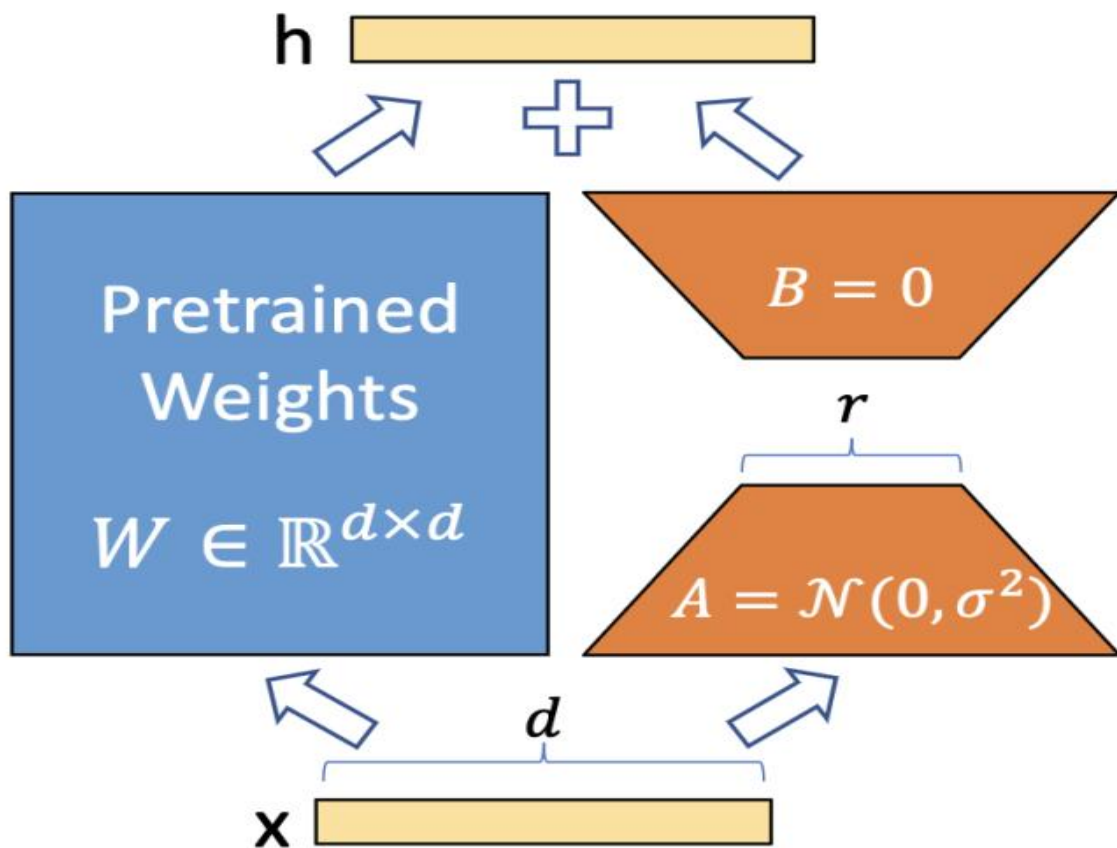
Image Classification

Model	LoRA	Prefix Tuning	P-Tuning	Prompt Tuning	IA3
ViT	✓				
Swin	✓				

Image to text (Multi-modal models)

Model	LoRA	Prefix Tuning	P-Tuning	Prompt Tuning	IA3
Blip-2	✓				

03、模型对比



优点:

①参数和成本上: 由于采用了低秩可分离矩阵, 相比于全参数微调而言, 参数量大幅度下降, 节省了内存, 并大幅度降低了训练成本和时间, 将硬件门槛降低了三倍。

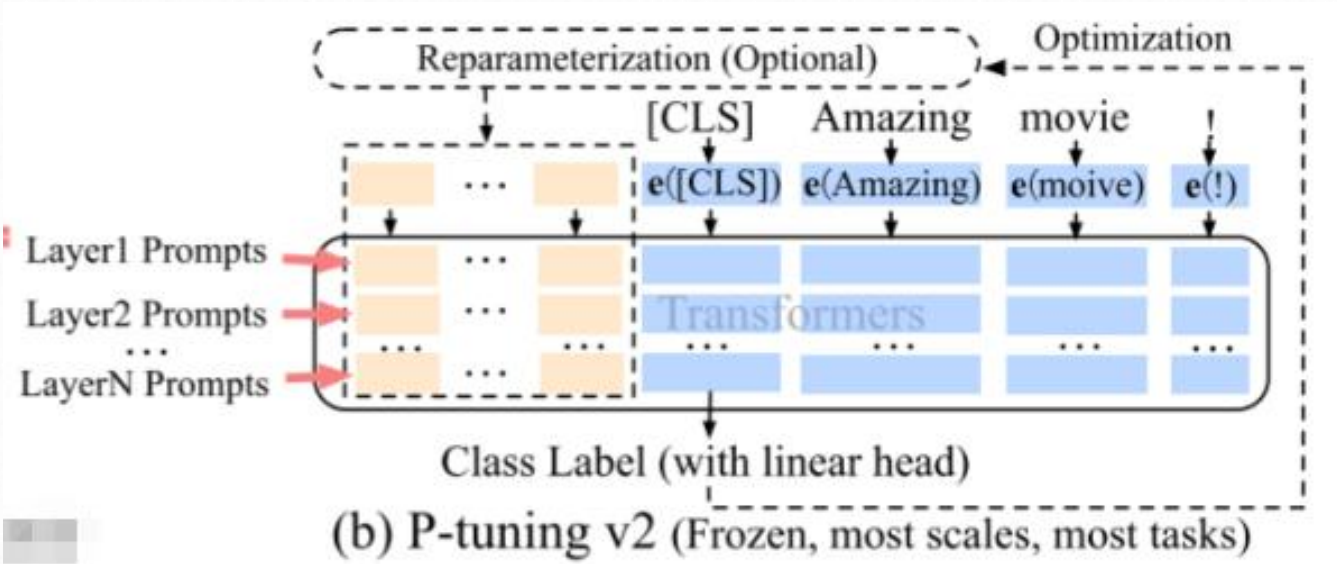
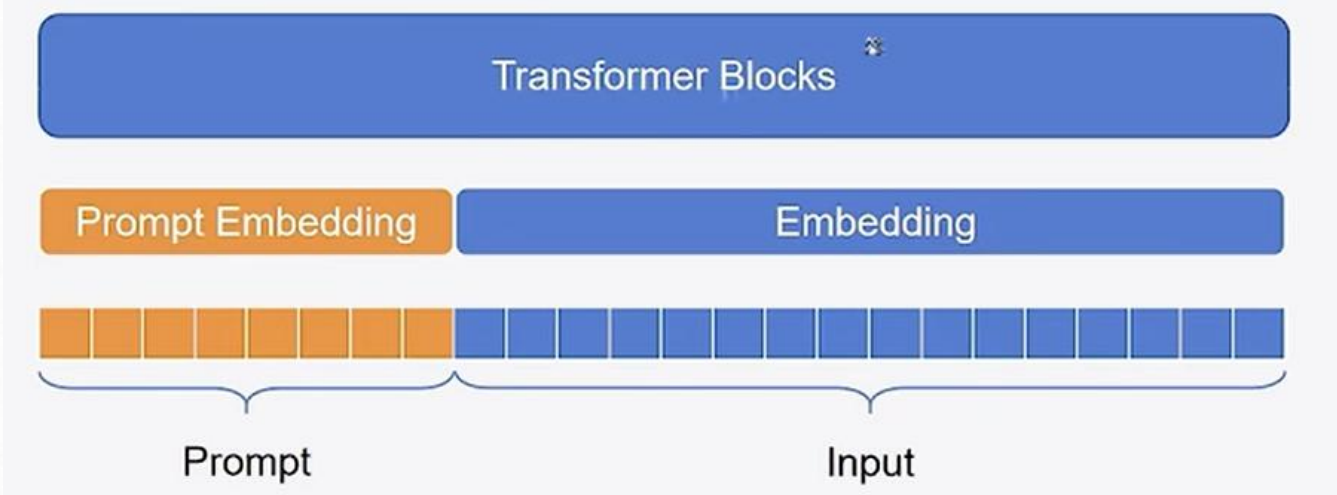
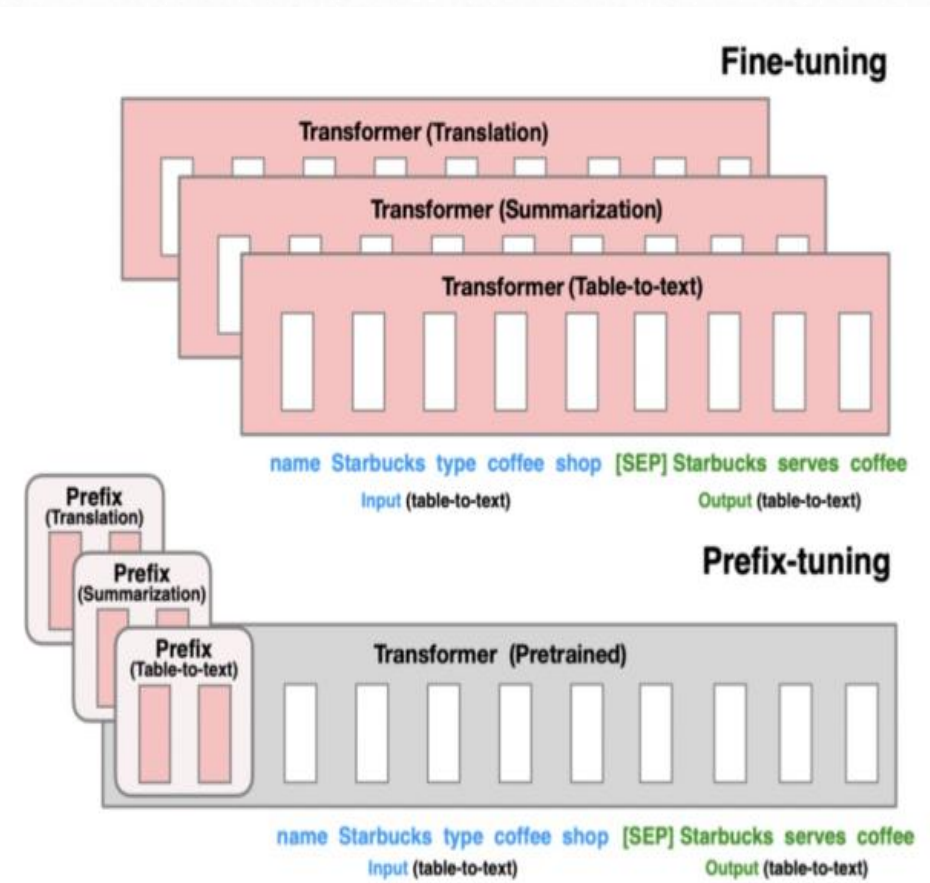
②推理上: 能够将训练的参数矩阵与冻结的参数矩阵合并, 不引入推理延迟。

③LoRA 与其它多种微调技术不冲突, 可以与其它微调技术结合。

缺点:

超参数秩 的最优值跟模型参数大小、微调数据大小、任务类型都有关系, 导致无法准确快速选择最优秩大小, 过小导致微调效果不佳, 过大导致过拟合, 最优秩可能需要多轮测试。

04、 Prompt Tuning



优点：相比于P-Tuning和Prefix-Tuning实现了改进，微调效果更稳定更好。

缺点：仍然存在规模不通用和任务不通用的问题，需要不同的Prompt长度和不同的Prompt模板。

② 微调实战之基于LoRA微调Llama2

01、Llama2



Github地址:

<https://github.com/facebookresearch/llama-recipes?tab=readme-ov-file>

Llama 2 was trained on 40% more data than Llama 1, and has double the context length.		
Llama 2		
MODEL SIZE (PARAMETERS)	PRETRAINED	FINE-TUNED FOR CHAT USE CASES
7B	Model architecture:	Data collection for helpfulness and safety:
13B	Pretraining Tokens: 2 Trillion	Supervised fine-tuning: Over 100,000
70B	Context Length: 4096	Human Preferences: Over 1,000,000

Meta(FaceBook)开源的预训练大模型

基于自回归架构的大模型

开源的尺寸有: 7B, 13B, 70B

丰富的中文扩展版本, 社区活跃

02、Llama2


模型地址：<https://www.modelscope.cn/models?name=llama2-7b-ms&page=1>

数据集：<https://huggingface.co/datasets/c-s-ale/alpaca-gpt4-data-zh>

Search this dataset		
instruction string · lengths	input string · lengths	output string · lengths
		
1 234	0 278	1 2.47k
辨认不同之处。	输入：Twitter, Instagram, Telegram。	Twitter, Instagram 和 Telegram 都是流行的社交媒体平台，但它们之间也有一些不同之处： - Twitter...
解释为什么以下分数等同于1/4	输入：4/16	4/16等于1/4是因为我们可以约分分子分母都除以他们的最大公约数4，得到 (4÷4) / (16÷4) =1/4。分数的约...
写一个关于主人公必须做出重要职业决定的第三人称叙述的短故事。		杰克是一位年轻有为的律师，他生活在纽约市，并在一家知名的法律事务所工作。杰克的生活似乎很完美，他有自...
渲染一座房子的三维模型		对不起，作为一个语言模型，我无法渲染三维模型。但我可以为您提供一些建议。 在制作三维模型时，您需要使...
朱利叶斯·凯撒是如何死亡的？		朱利叶斯·凯撒，又称尤利乌斯·恺撒（Julius Caesar）是古罗马的政治家、军事家和作家。他于公元...
法国的首都是什么？		法国的首都是巴黎。


#模型下载方式1

```
from modelscope import snapshot_download
snapshot_download(mode_id='modelscope/Llama-2-7b-ms',cache_dir='./')
```

 **ModelScope** [首页](#) [模型库](#) [数据集](#) [创空间](#) [文档中心](#)

模型库 / Llama-2-7b-ms

Llama-2-7b-ms











modelscope/Llama-2-7b-ms  (ModelScope 提供 | 9786 次下载 | 2023-08-09更新)

开源协议：Apache License 2.0 [facebook](#) [meta](#) [pytorch](#) [llama](#) [llama-2](#)

[模型介绍](#) [模型文件](#) [提交反馈](#)

master Llama-2-7b-ms

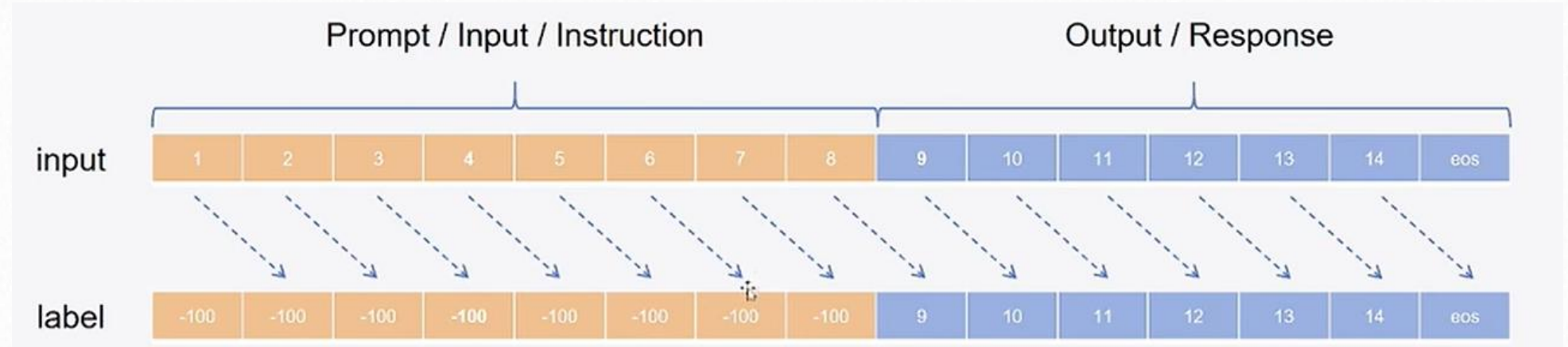
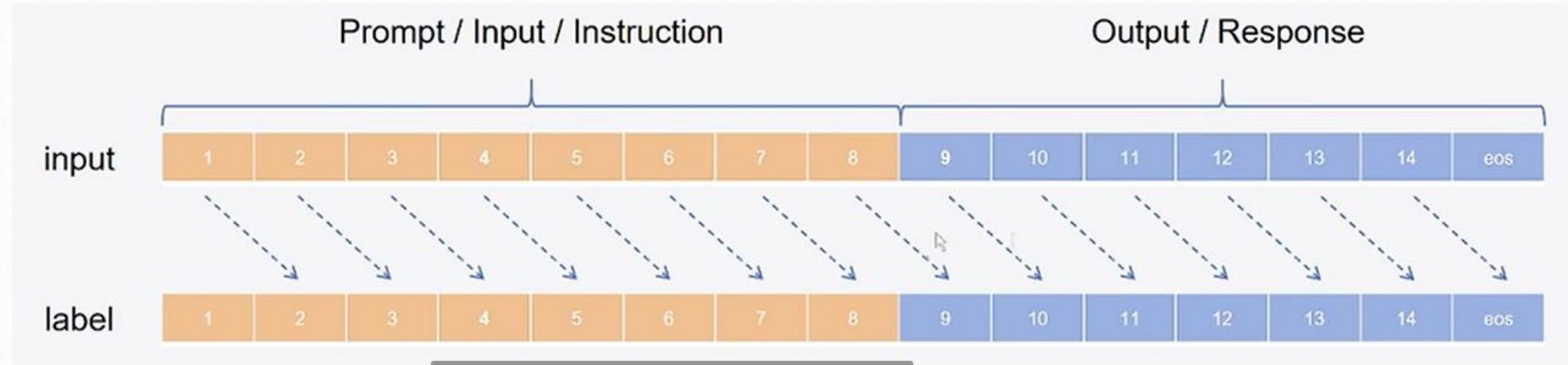
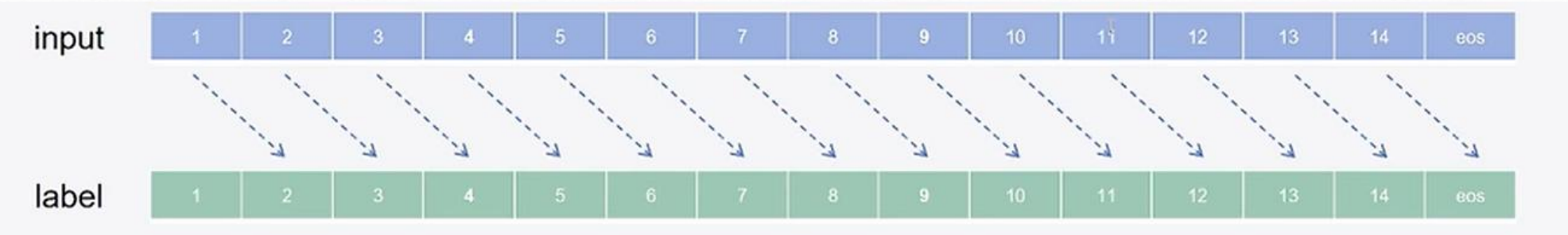
huangjintao Update README.md a0d1efc4

 .gitattributes	1.56KB
 added_tokens.json	21.00B
 config.json	583.00B
 configuration.json	183.00B
 generation_config.json	179.00B
 LICENSE.txt	6.86KB
 model-00001-of-00002.safetensors 	9.29GB
 model-00002-of-00002.safetensors 	9.26GB

#模型下载方式2

```
git clone https://www.modelscope.cn/modelscope/Llama-2-7b-ms.git
```

03、指令微调数据处理



04、训练流程演示

模型大小	数据集	微调技术	使用资源
Llama2-7B	alpaca_data_zh	LoRA	半精度训练

MAX_LENGTH设置

`torch_dtype=torch.half`

批处理注意点

默认往左对齐，设置成往右对齐

`eos_token`会被切碎

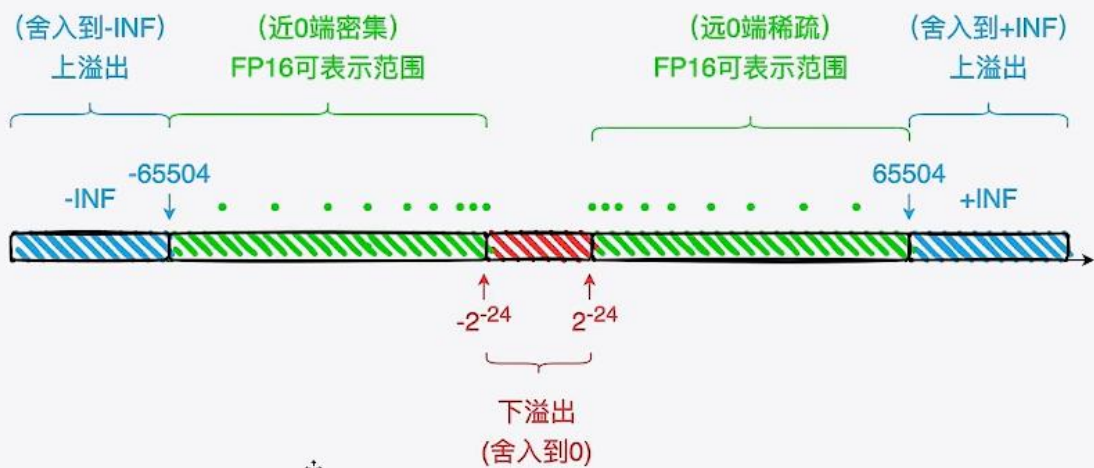
导致模型回答的时候不会停止

`pad_token_id`设置为`eos_token_id`

半精度训练时，需要把`pad_token_id`设置`eos_token_id` 否则无法收敛

05、训练流程演示

精度溢出



```

: ## adam_epsilon=1e-4
args = TrainingArguments(
    output_dir="/root/autodl-tmp/llama2output",
    per_device_train_batch_size=2,
    gradient_accumulation_steps=8,
    logging_steps=10,
    num_train_epochs=1,
    adam_epsilon=1e-4
)

```

③ 微调实战之ChatGLM3微调

01、经典的Transformer架构

掩码的方式不一样

训练数据不一样

训练目标不一样

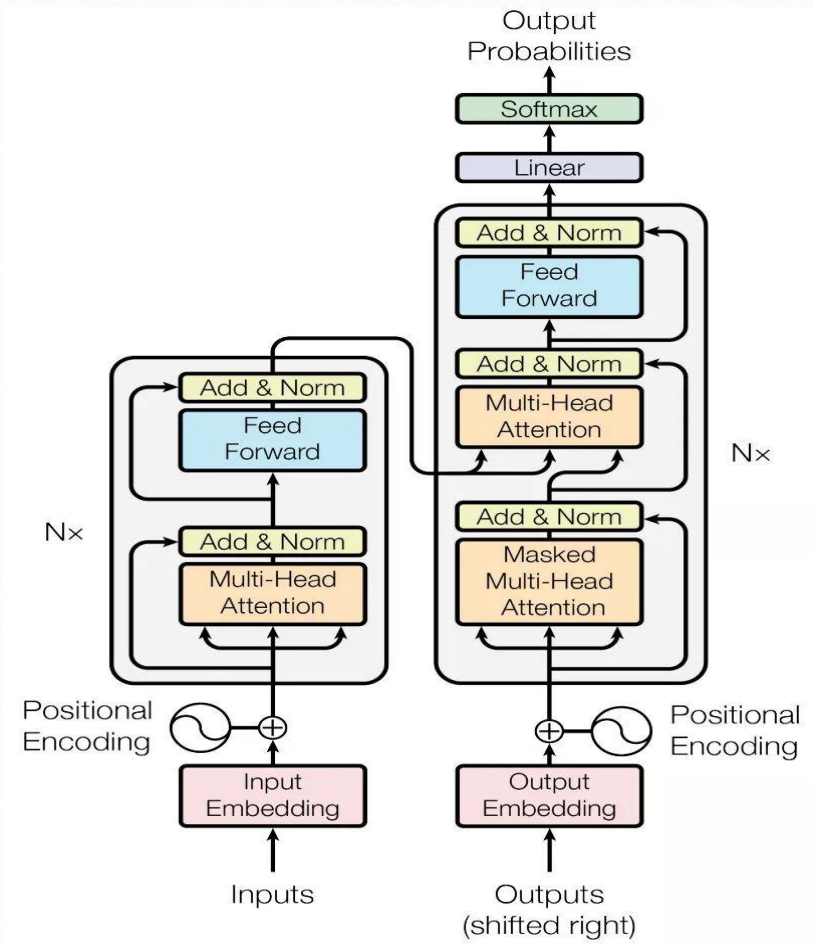


Figure 1: The Transformer - model architecture.

模型	结构	位置编码	激活函数	layer norm方法
原生 Transformer	Encoder-Decoder	Sinusoidal编码	ReLU	Post layer norm
BERT	Encoder	绝对位置编码	GeLU	Post layer norm
LLaMA	Casual decoder	RoPE	SwiGLU	Pre RMS Norm
ChatGLM-6B	Prefix decoder	RoPE	GeGLU	Post Deep Norm
Bloom	Casual decoder	ALiBi	GeLU	Pre Layer Norm

02、ChatGLM3架构设计

1、GLM 动机

目前大预训练语言模型分为三类:

1、自编码模型 (Auto Encoding): BERT, ROBERTa, DeBERTa, ALBERT等; attention是双向的。主要用于以下类型任务:

自然语言理解任务: 情感分类, 抽取式问答, 自然语言推理等。

2、自回归模型 (Auto Regressive): GPT系列, Llama, 百川, 等; attention是单向的。主要用于以下类型任务:

无条件生成任务: 语言建模等。

3、编码器-解码器模型 (Encoder-Decoder): T5, BART, MASS, PALM等; encoder的attention是双向的, decoder的attention是单向的。主要用于以下类型任务:

条件生成(seq2seq): 摘要, 机器翻译等。

但问题是, 这三大类模型, 没有一种能在这三方面都达到最佳的。那么, GLM的出发点就是, 设计一个模型能够兼具这3类模型的优点, 并在这3大类任务上取得最佳性能。

自回归填空

2D的PE

填空序列乱序

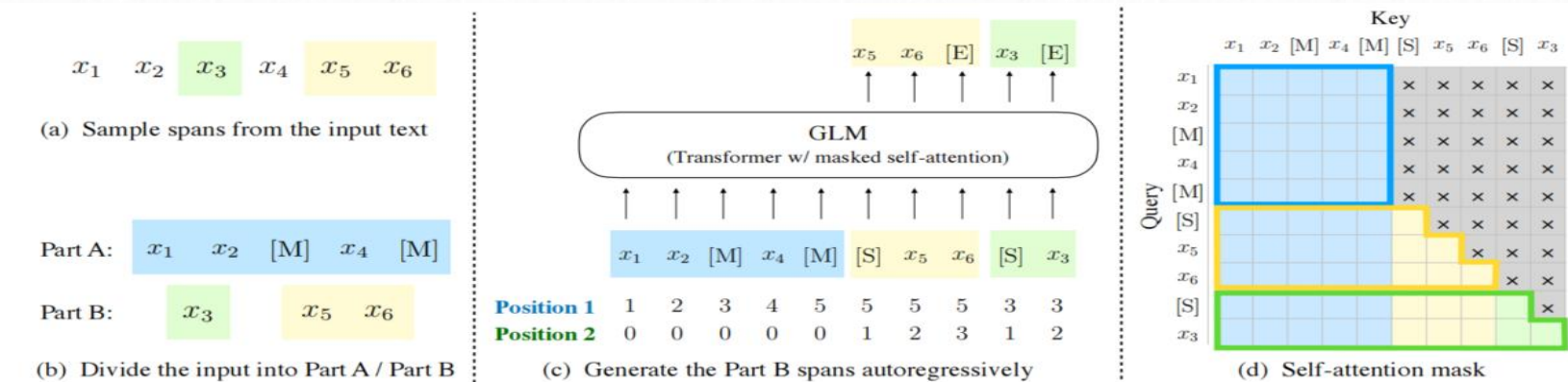
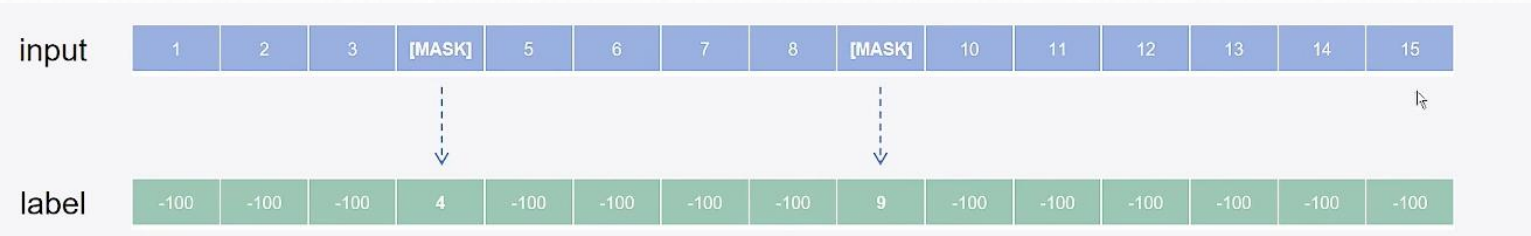


Figure 2: GLM pretraining. (a) The original text is $[x_1, x_2, x_3, x_4, x_5, x_6]$. Two spans $[x_3]$ and $[x_5, x_6]$ are sampled. (b) Replace the sampled spans with [M] in Part A, and shuffle the spans in Part B. (c) **GLM autoregressively generates Part B**. Each span is prepended with [S] as input and appended with [E] as output. 2D positional encoding represents inter- and intra-span positions. (d) Self-attention mask. Grey areas are masked out. Part A tokens can attend to themselves (blue frame) but not B. Part B tokens can attend to A and their antecedents in B (yellow and green frames correspond to the two spans). [M] := [MASK], [S] := [START], and [E] := [END].

03、预训练任务类型

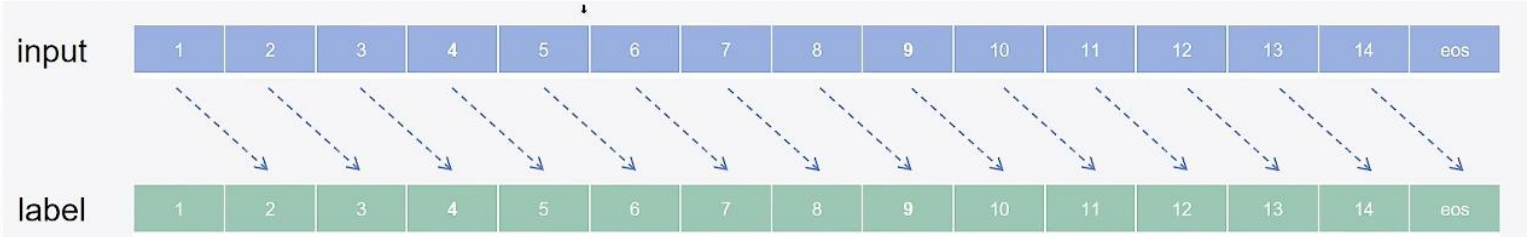
掩码语言模型，自编码模型

将一些位置的token替换成特殊[MASK]字符，预测被替换的字符



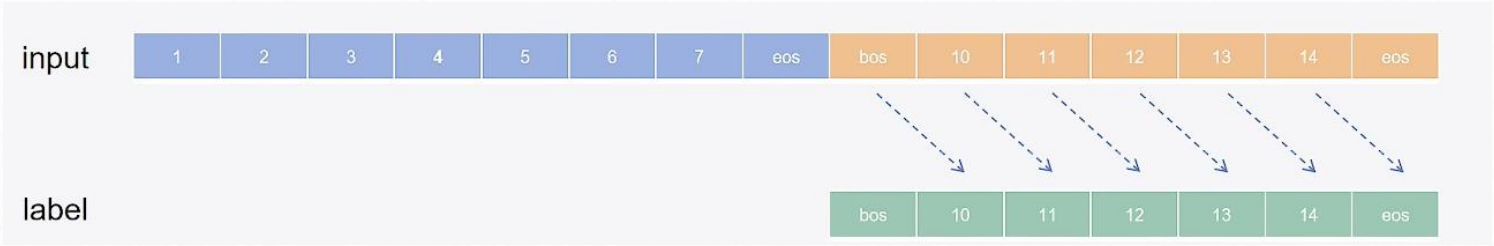
因果模型，自回归模型

将完整序列输入，基于上文的token预测下文的token



序列到序列模型

采用编码器解码器的方式，预测放在解码器部分



04、Prompt格式

```
<|system|>
You are ChatGLM3, a large language model trained by Zhipu.AI. Follow the user's instructions carefully. Respond using markdown.
<|user|>
Hello
<|assistant|>
Hello, I'm ChatGLM3. What can I assist you today?
```

工具调用

```
<|system|>
Answer the following questions as best as you can. You have access to the following tools:
[
  {
    "name": "get_current_weather",
    "description": "Get the current weather in a given location",
    "parameters": {
      "type": "object",
      "properties": {
        "location": {
          "type": "string",
          "description": "The city and state, e.g. San Francisco, CA",
        },
        "unit": {"type": "string"},
      },
      "required": ["location"],
    },
  },
]
<|user|>
今天北京的天气怎么样？
<|assistant|>
好的，让我们来看看今天的天气
<|assistant|>get_current_weather
```python
tool_call(location="beijing", unit="celsius")
```
<|observation|>
{"temperature": 22}
<|assistant|>
根据查询结果，今天北京的气温为 22 摄氏度。
```

```
tokenizer.build_chat_input("prompt", history=[], role="user")
tokenizer.decode([xxx])
```

v1 Prompt [gMASK][sop] response [eop]

v2 [gMASK][sop] Prompt Reponse [eos]

v3 [gMASK]sop<|user|> \n Prompt<|assistant|>
\n response eos_token

05、ChatGLM3基于LoRA微调

数据集：<https://huggingface.co/datasets/c-s-ale/alpaca-gpt4-data-zh>

模型：<https://www.modelscope.cn/models/ZhipuAI/chatglm3-6b-base/files>

| instruction
string · lengths | input
string · lengths | output
string · lengths |
|---|---|--|
|  |  |  |
| 1234 | 0278 | 12.47k |
| 辨认不同之处。 | 输入: Twitter, Instagram, Telegram。 | Twitter, Instagram 和 Telegram 都是流行的社交媒体平台, 但它们之间也有一些不同之处: - Twitter... |
| 解释为什么以下分数等同于1/4 | 输入: 4/16 | 4/16等于1/4是因为我们可以约分分子分母都除以他们的最大公约数4, 得到 (4÷4) / (16÷4) =1/4。分数的约... |
| 写一个关于主人公必须做出重要职业决定的第三人称叙述的短故事。 | | 杰克是一位年轻有为的律师, 他生活在纽约市, 并在一家知名的法律事务所工作。杰克的生活似乎很完美, 他有自... |
| 渲染一座房子的三维模型 | | 对不起, 作为一个语言模型, 我无法渲染三维模型。但我可以为您提供一些建议。 在制作三维模型时, 您需要使... |
| 朱利叶斯·凯撒是如何死亡的? | | 朱利叶斯·凯撒, 又称尤利乌斯·恺撒 (Julius Caesar) 是古罗马的政治家、军事家和作家。他于公元... |
| 法国的首都是什么? | | 法国的首都都是巴黎。 |

数据格式：

```
[gMASK]sop<|user|> \n query<|assistant|>
\n response eos_token
```

 ModelScope

首页 模型库 数据集 创空间 文档中心

模型库 / chatglm3-6b-base

chatglm3-6b-base

ZhipuAI/chatglm3-6b-base (智谱AI 提供 | 76103 次下载 | 2023-11-04更新)

glm

chatglm

thudm

模型介绍

模型文件

提交反馈

master chatglm3-6b-base

Cherrytest

Update configuration.json

5ede155b

.gitattributes

1.48KB

config.json

1.29KB

configuration.json

48.00B

configuration_chatglm.py

2.28KB

MODEL_LICENSE

4.04KB

modeling_chatglm.py

54.35KB

04、ChatGLM3/4基于P-Tuning V2微调

| 量化等级 | 最低 GPU 显存 (推理) | 最低 GPU 显存 (高效参数微调) |
|------------|----------------|--------------------|
| FP16 (无量化) | 13 GB | 14 GB |
| INT8 | 8 GB | 9 GB |
| INT4 | 6 GB | 7 GB |

资源准备

镜像 PyTorch 2.0.0 Python 3.8(ubuntu20.04) Cuda 11.8 [更换](#)

GPU RTX 4090(24GB) * 1 [升降配置](#)

CPU 12 vCPU Intel(R) Xeon(R) Platinum 8352V CPU @ 2.10GHz

内存 90GB

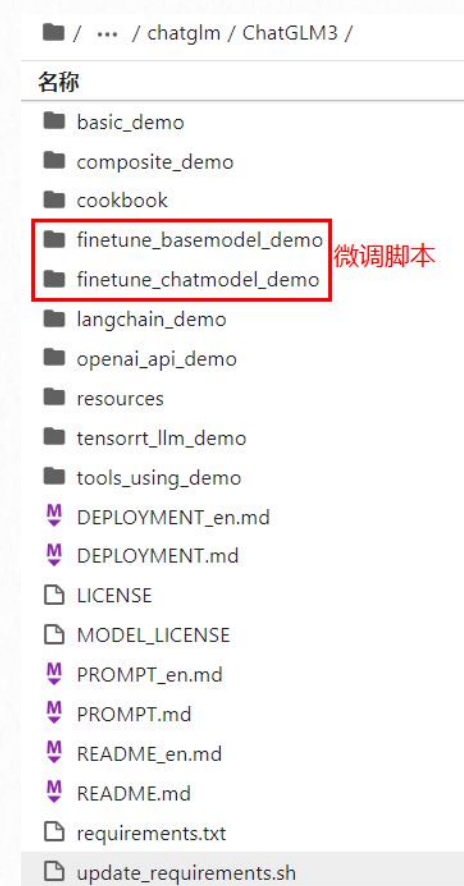
硬盘 系统盘: 30 GB

数据盘: 免费:50GB 付费:280GB [扩容](#) [缩容](#)

05、ChatGLM3/4基于P-Tuning V2微调

获取工程

<https://github.com/THUDM/ChatGLM3.git>



获取模型权重文件

进入魔塔下载: <https://www.modelscope.cn/>

使用如下代码:

```
from modelscope import snapshot_download
```

```
snapshot_download(mode_id='ZhipuAI/chatglm3-6b',cache_dir=".")
```



/ ... / model / chatglm3-6b /

名称

- config.json
- configuration_chatglm.py
- MODEL_LICENSE
- model-00001-of-00007.safetensors
- model-00002-of-00007.safetensors
- model-00003-of-00007.safetensors
- model-00004-of-00007.safetensors
- model-00005-of-00007.safetensors
- model-00006-of-00007.safetensors
- model-00007-of-00007.safetensors
- model.safetensors.index.json
- modeling_chatglm.py
- pytorch_model-00001-of-00007.bin
- pytorch_model-00002-of-00007.bin
- pytorch_model-00003-of-00007.bin
- pytorch_model-00004-of-00007.bin
- pytorch_model-00005-of-00007.bin
- pytorch_model-00006-of-00007.bin
- pytorch_model-00007-of-00007.bin
- pytorch_model.bin.index.json
- quantization.py
- README.md

06、ChatGLM3/4基于P-Tuning V2微调

进入 finetune_chatmodel_demo目录

```
drwxr-xr-x 5 root root 4096 Jan 10 16:54 ./
drwxr-xr-x 16 root root 4096 Jan 10 16:54 ../
-rw-r--r-- 1 root root 8753 Jan 10 16:39 README.md
drwxr-xr-x 2 root root 126 Jan 10 16:45 __pycache__/
-rw-r--r-- 1 root root 4997 Jan 10 16:39 arguments.py
drwxr-xr-x 2 root root 36 Jan 10 16:39 configs/
-rw-r--r-- 1 root root 7003 Jan 10 16:39 finetune.py
-rw-r--r-- 1 root root 1986 Jan 10 16:39 inference.py
-rw-r--r-- 1 root root 6252 Jan 10 16:39 preprocess_utils.py
-rw-r--r-- 1 root root 114 Jan 10 16:39 requirements.txt
drwxr-xr-x 3 root root 4096 Jan 10 16:48 scripts/
-rw-r--r-- 1 root root 3279 Jan 10 16:39 trainer.py
```

微调的说明文件

模型参数配置文件

DeepSpeed的配置文件

接口文件（主程序）

推理接口文件

逻辑处理代码

依赖环境

微调启动脚本

模型训练文件

微调脚本准备

进入 finetune_chatmodel_demo里的scripts目录

```
total 52
drwxr-xr-x 2 root root 4096 Jan 11 13:41 ./
drwxr-xr-x 4 root root 4096 Jan 11 13:41 ../
-rw-r--r-- 1 root root 1153 Jan 11 13:41 finetune_ds.sh
-rw-r--r-- 1 root root 1010 Jan 11 13:41 finetune_ds_multiturn.sh
-rw-r--r-- 1 root root 1124 Jan 11 13:41 finetune_pt.sh
-rw-r--r-- 1 root root 1048 Jan 11 13:41 finetune_pt_multiturn.sh
-rw-r--r-- 1 root root 585 Jan 11 13:41 format_advertise_gen.py
-rw-r--r-- 1 root root 1822 Jan 11 13:41 format_tool_alpaca.py
```

单轮对话的全量微调脚本

多轮对话的全量微调脚本

单轮对话的P-Tuning v2 微调脚本

多轮对话的P-Tuning V2 微调脚本

格式化单轮对话Advertise数据集

格式化多轮对话ToolAlpaca数据集

07、ChatGLM3/4基于P-Tuning V2微调

数据准备

安装必要的依赖
pip install -r requirements.txt

数据格式

```
[
  {
    "prompt": "<prompt text>",
    "response": "<response text>"
  }
  // ...
]
```

下载处理好的 AdvertiseGen 数据集，将解压后的 AdvertiseGen 目录放到本目录下。
下载地址：<https://cloud.tsinghua.edu.cn/f/b3f119a008264b1cabd1/?dl=1>



执行如下语句，对数据格式进行处理

```
./scripts/format_advertise_gen.py --path "AdvertiseGen/train.json"
```

```
{"content": "类型#上衣*版型#宽松*版型#显瘦*衣样式#外套*衣袖型#插肩袖*衣款式#拼接",
"summary": "宽松的版型，穿搭起来总是不挑身材；所以作为早春穿搭的话，有着插肩袖设计的这款外套，最是能展现出舒适和大方的感觉了。而比较宽松的外套款式，它在衣身上特别做了拼接的设计，你看那颜色独特的拼接，很是容易就能展现出独特和抢眼的效果；再加上直筒的版型设计，穿搭起来真的是一点也不挑身材，还能起到显瘦的效果。"}

{"content": "类型#上衣*风格#运动*风格#休闲*衣样式#外套*衣领型#立领*衣袖长#长袖*衣门襟#拉链*衣款式#拉链",
"summary": "基础的外套廓形，直筒，立领长袖，中间金属拉链穿脱，方便实用，带有浓浓的休闲运动味道。日常休闲上班或是去<UNK>等运动时都可以穿着，时尚前卫。"}

{"content": "类型#上衣*风格#街头*图案#创意*衣样式#卫衣",
"summary": "在这件卫衣上，BRAND-white集合了女性化的柔美还有不变的街头风采，<UNK><UNK>的向日葵花朵美丽的出现在胸前和背后，犹如暗<UNK>闪光的星星一般耀眼又充满着<UNK>的生命力，而后品牌标志性的logo<UNK>出现，呈现出将花束固定的效果，有趣极了，穿的不仅是服饰，更是新颖创意的载体。"}

```

```
{"prompt": "类型#裤*版型#宽松*风格#性感*图案#线条*裤型#阔腿裤",
"response": "宽松的阔腿裤这两年真的吸粉不少，明星时尚达人的心头爱。毕竟好穿时尚，谁都能穿出腿长2米的效果宽松的裤腿，当然是遮肉小能手啊。上身随性自然不拘束，面料亲肤舒适贴身体验感棒棒哒。系带部分增加设计看点，还让单品的设计感更强。腿部线条若隐若现的，性感撩人。颜色敲温柔的，与裤子本身所呈现的风格有点反差萌。"}

{"prompt": "类型#裙*风格#简约*图案#条纹*图案#线条*图案#撞色*裙型#鱼尾裙*裙袖长#无袖",
"response": "圆形领口修饰脖颈线条，适合各种脸型，耐看有气质。无袖设计，尤显清凉，简约横条纹装饰，使得整身人鱼造型更为生动立体。加之撞色的鱼尾下摆，深邃富有诗意。收腰包臀，修饰女性身体曲线，结合别出心裁的鱼尾裙摆设计，勾勒出自然流畅的身体轮廓，展现了婀娜多姿的迷人姿态。"}

```

08、ChatGLM3/4基于P-Tuning V2微调

模型推理

模型微调

```
./scripts/finetune_ds.sh # 全量微调  
./scripts/finetune_pt.sh # P-Tuning v2 微调
```

```
#!/usr/bin/env bash
```

```
set -ex
```

```
PRE_SEQ_LEN=128  
LR=2e-2  
NUM_GPUS=1  
MAX_SOURCE_LEN=1024  
MAX_TARGET_LEN=128  
DEV_BATCH_SIZE=1  
GRAD_ACCUMULATION_STEPS=32  
MAX_STEP=1000  
SAVE_INTERVAL=500
```

```
AUTORESUME_FROM_CHECKPOINT=True  
DATESTR=`date +%Y%m%d-%H%M%S`  
RUN_NAME=advertise_gen_pt
```

```
BASE_MODEL_PATH=THUDM/chatglm3-6b  
DATASET_PATH=formatted_data/advertise_gen.jsonl  
OUTPUT_DIR=output/${RUN_NAME}-${DATESTR}-${PRE_SEQ_LEN}-${LR}
```

```
mkdir -p $OUTPUT_DIR
```

对于输入输出格式的微调，可使用 `inference.py` 进行基本的推理验证。

```
python inference.py \  
    --pt-checkpoint "path to p-tuning checkpoint" \  
    --model THUDM/chatglm3-6b
```

```
python inference.py \  
    --tokenizer THUDM/chatglm3-6b \  
    --model "path to finetuned model checkpoint"
```

```
python inference.py \  
    --pt-checkpoint /root/autodl-  
tmp/chatglm/ChatGLM3/finetune_chatmodel_demo/output/advertise_gen_pt-20240118-  
135232-128-2e-2/ \  
    --model "/root/autodl-tmp/chatglm/model/chatglm3-6b/"
```

```
torchrun --standalone --nnodes=1 --nproc_per_node=$NUM_GPUS ../finetune.py \  
    --quantization bit 4 \   使用INT4量化等级  
    --train_format input-output \  
    --train_file $DATASET_PATH
```


09、核心参数介绍

`--train_format input-output` 在脚本里面指定数据格式，input-output是单轮对话，multi-turn是多轮对话

`--train_file $DATASET_PATH`: 指定训练数据集的文件路径。变量\$DATASET_PATH应该被替换为实际的数据集文件路径。

`--preprocessing_num_workers 1`: 指定在数据预处理阶段使用的工作进程数。这里设置为1，意味着使用一个工作线程进行数据预处理。

`--model_name_or_path $BASE_MODEL_PATH`: 指定预训练模型的名称或者路径。变量\$BASE_MODEL_PATH应该被替换为预训练模型所在的路径。

`--output_dir $OUTPUT_DIR`: 指定模型训练后的输出目录。变量\$OUTPUT_DIR应该被替换为实际的输出目录路径。

`--max_source_length $MAX_SOURCE_LEN` 和 `--max_target_length $MAX_TARGET_LEN`: 分别设置训练过程中输入（source）和输出（target）序列的最大长度。超过这个长度的序列将被截断。

`--per_device_train_batch_size $DEV_BATCH_SIZE`: 设置每个设备（如每块GPU）上进行训练的批量大小。

`--gradient_accumulation_steps $GRAD_ACCUMULATION_STEPS`: 梯度累积步骤数，用于在更新模型权重之前累积梯度，这有助于在内存有限的情况下模拟更大的批量大小。

`--max_steps $MAX_STEP`: 设置训练过程中最大步骤数，达到此数后训练将停止。

`--logging_steps 1`: 设置日志打印频率。这里表示每进行一步训练就记录一次日志。

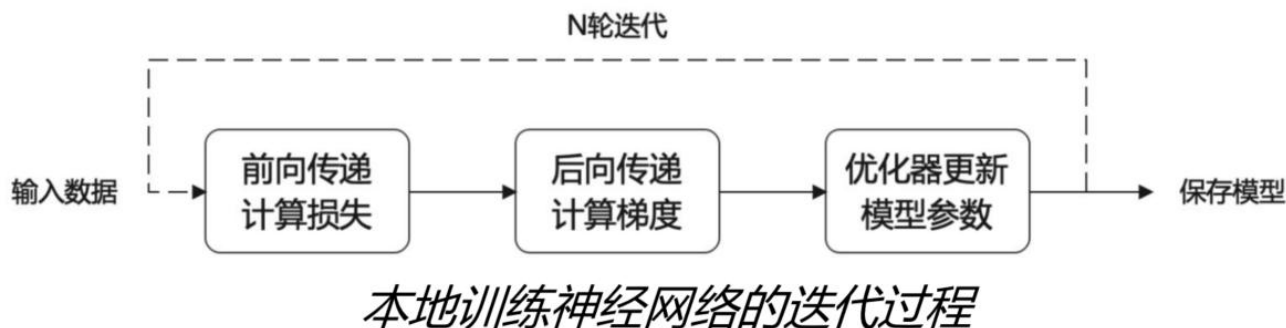
`--save_steps $SAVE_INTERVAL`: 设置模型保存的间隔步数。变量\$SAVE_INTERVAL应该被替换为实际的步数，每训练这么多步数后，模型的状态将被保存。

`--learning_rate $LR`: 设置训练过程中的学习率。变量\$LR应该被替换为实际使用的学习率值。学习率是一个非常重要的超参数。它决定了模型参数每次更新的步长。简单来说，如果学习率设置得太大，模型可能会在最优解附近震荡，无法收敛；反之，如果学习率设置得太小，虽然模型能够向最优解靠近，但是可能需要更多的时间和计算资源，甚至可能会陷入局部最优解

`--pre_seq_len $PRE_SEQ_LEN`: Prompts序列的长度

④ 微调实战之基于多卡方案微调ChatGLM3

01、分布式模型训练



1、**显存效率**：模型参数量太大，显存不够用

- 即使目前显存最大的 GPU 也放不下这些大模型的模型参数。
- 例如：175B 参数量的 GPT-3模型参数需要占用 700 GB ($175B * 4bytes$) 的显存。参数梯度是 700GB, Adam 优化器状态需要 1400 GB, 共计需要 **2.8 TB** 的显存。

2、**计算效率**：训练数据量多，模型参数量大，计算量大，单机训练时间久

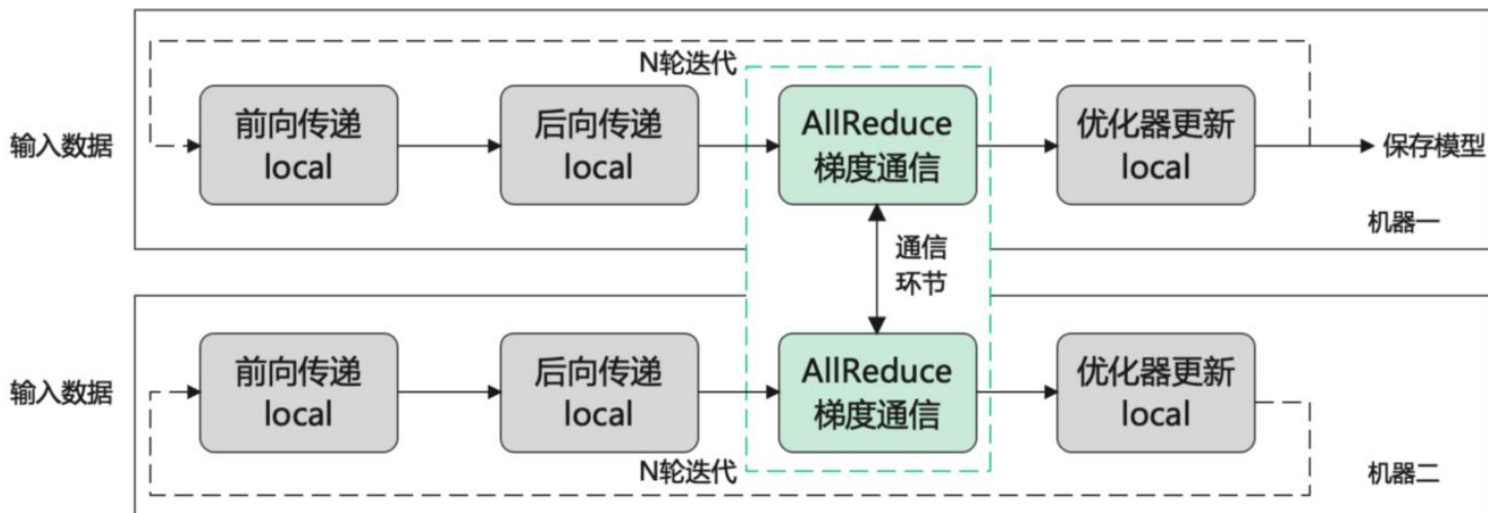
- 即使我们能将大模型放在一张 GPU 上，训练大模型需要的海量计算操作需要耗费很长时间。例如：用英伟达 A100 显卡训练 175B 参数量的 GPT-3 模型大约需要 **288 年**。

02、数据并行

1、**数据并行**：指的是将整个数据集切分为多份，每张 GPU 分配到不同的数据进行训练，每个进程都有一个完整的模型副本。



本地训练神经网络的迭代过程

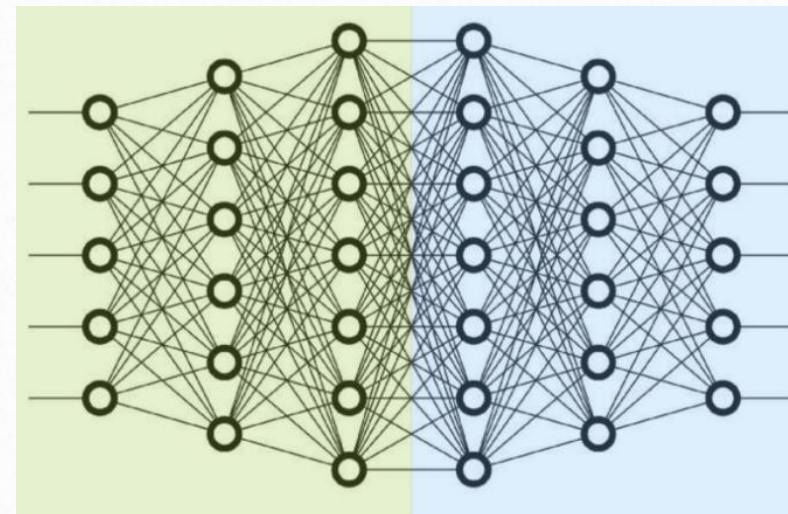
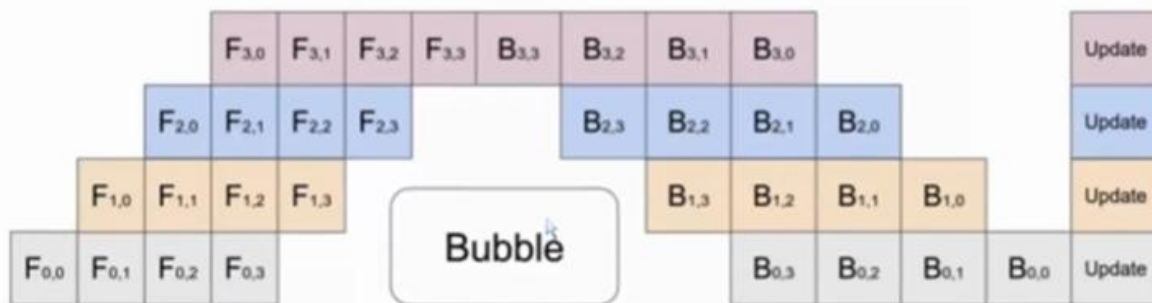


数据并行的迭代过程

03、模型并行

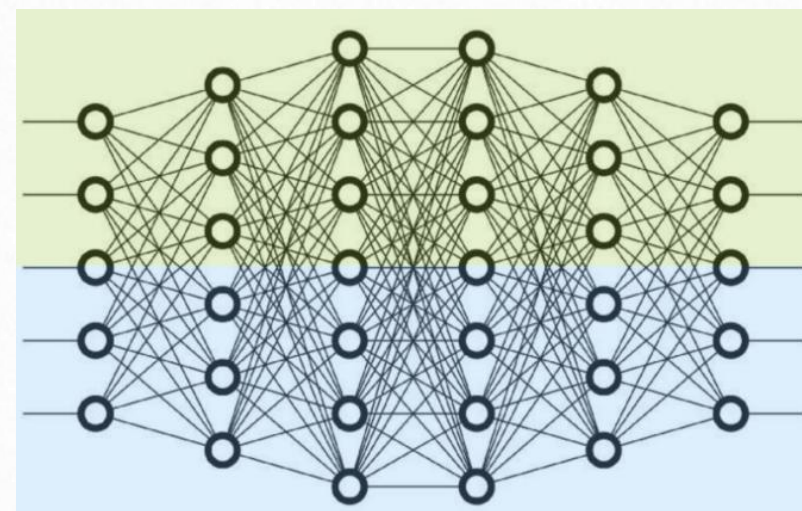
1、流水线并行 (inter-layer)

- 层间划分，将不同的层划分到不同的 GPU 上
- 前 3 层在 0 号卡上，后 3 层在 1 号卡上



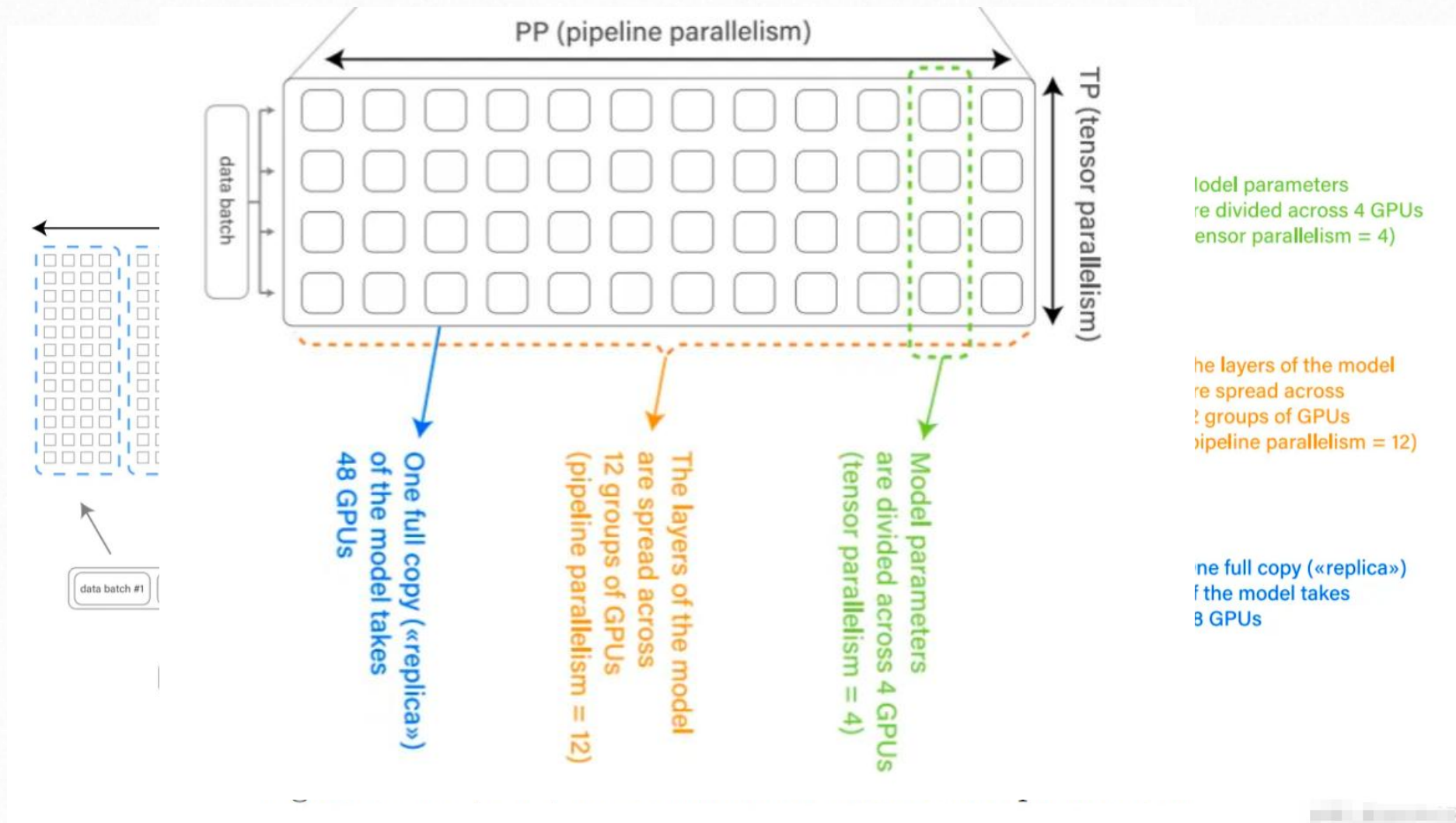
2、张量并行 (intra-layer)

- 层内划分，切分一个独立的层划分到不同的 GPU 上
- 0 号卡和 1 号卡分别计算某个层的不同部分



04、3D并行

Bloom-176B 进行预训练时，在 394 张 NVIDIA A100 80GB GPU (48 个节点) 上使用了 3D 并行（数据并行、流水线并行、张量并行）策略，针对 350B 个Token 训练了大约 3.5 个月



05、DeepSpeed

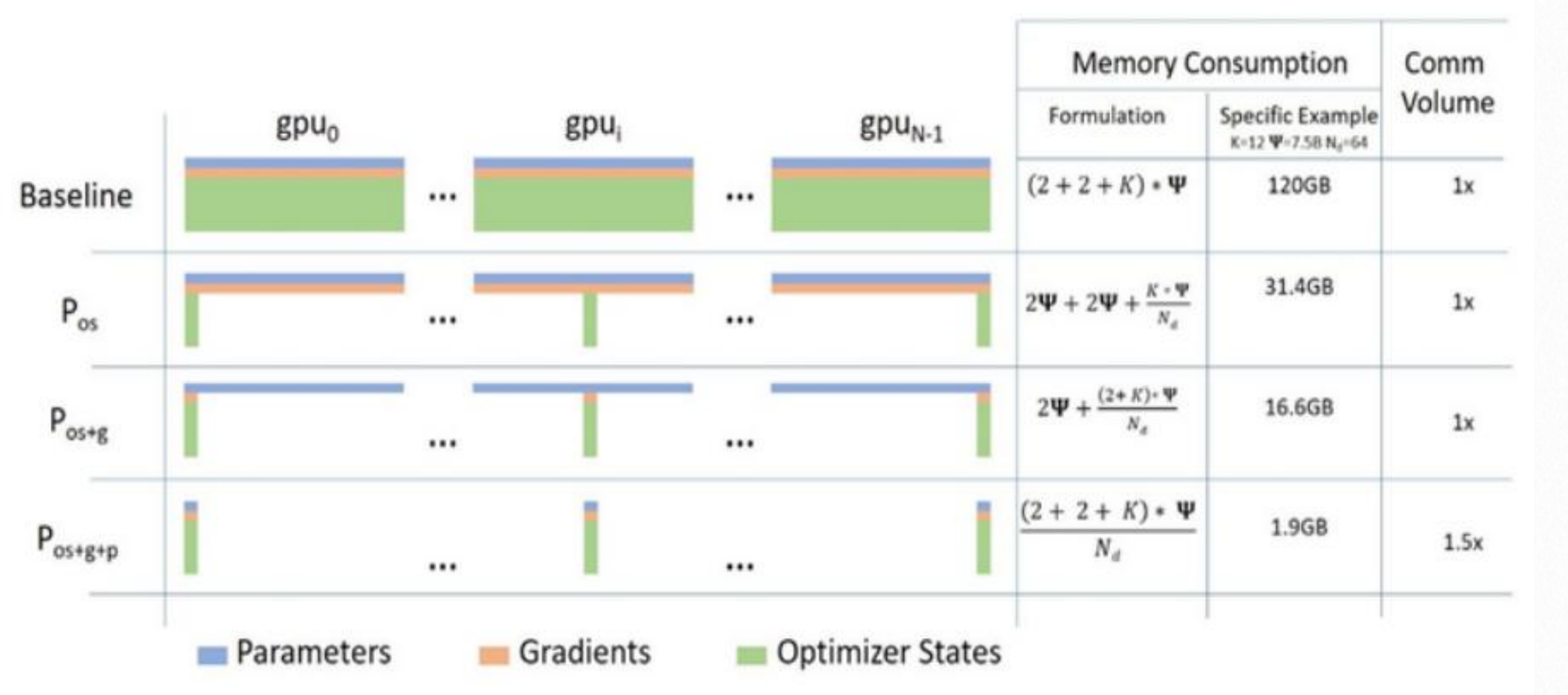
DeepSpeed是一个由微软开发的开源深度学习优化库，旨在提高大规模模型训练的效率和可扩展性。它通过多种技术手段来加速训练，包括模型并行化、梯度累积、动态精度缩放、本地模式混合精度等。DeepSpeed还提供了一些辅助工具，如分布式训练管理、内存优化和模型压缩等，以帮助开发者更好地管理和优化大规模深度学习训练任务。

DeepSpeed里的核心技术是：ZeRO(Zero Redundancy Optimizer) 零冗余优化器

| Memory Consumption | |
|--------------------|----------------------------|
| Model States | Residual States |
| Parameters | Activation（激活检查点） |
| Gradients | Temporary Buffers（临时缓冲区） |
| Optimizer States | Unusable Fragmented Memory |

| 论文 | 核心技术 | 备注 |
|-------|---------------|-----------------|
| 第一篇论文 | ZeRO | ZeRO-DP, ZeRO-R |
| 第二篇论文 | ZeRO-Offload | |
| 第三篇论文 | ZeRO-Infinity | |

06、第一篇论文：ZeRO-DP



07、第一篇论文-ZeRO-R

| ZeRO-R:Optimize Residual States | |
|---------------------------------|-------|
| Activation | 激活检查点 |
| Temporary Buffer | 临时缓冲区 |
| Unusable Fragmented Memory | 空间管理 |

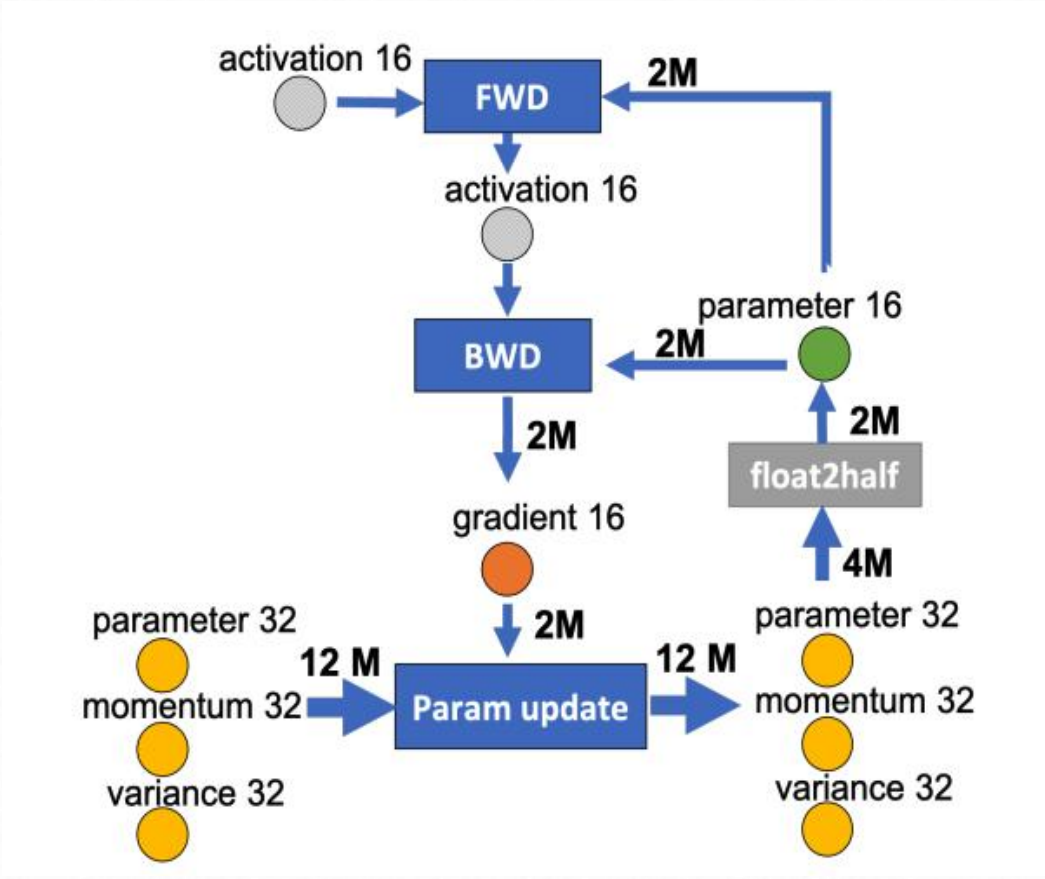
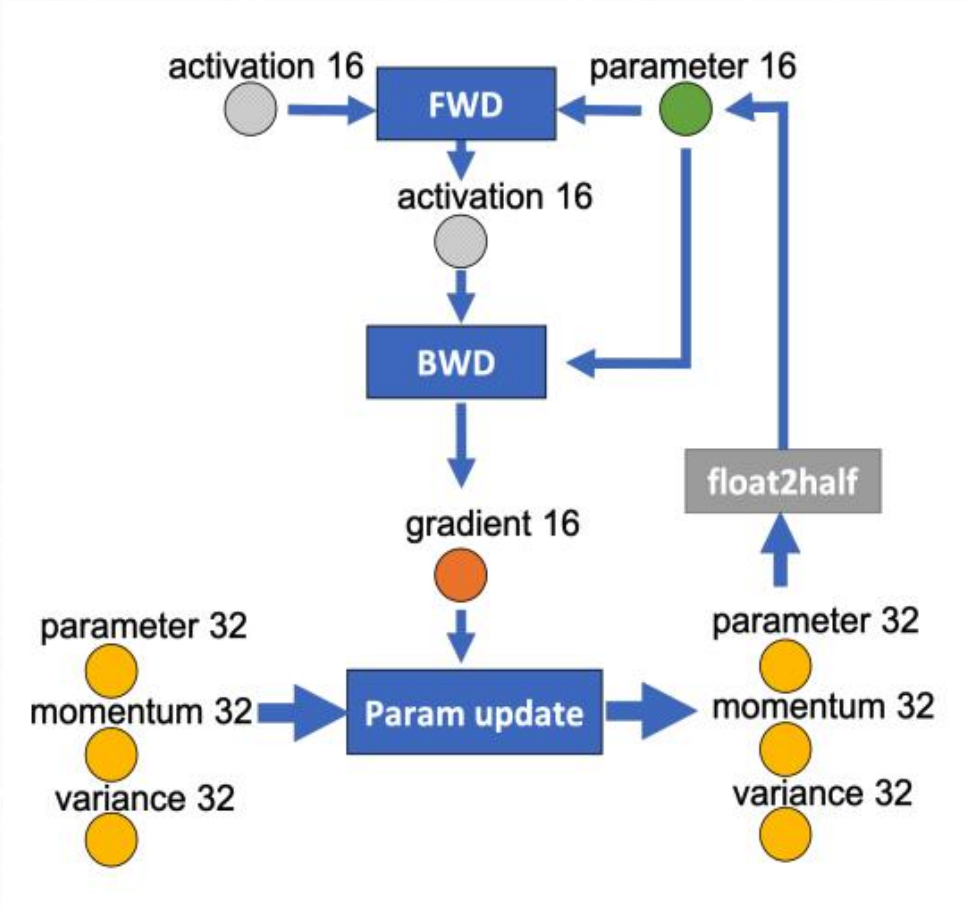
Activation的优化：前向传播的时候，需要存储一个激活值，因为后向传播的时候需要到。它的第一种优化方式就是：我不存储了，因为我知道是怎么计算的，用的时候重新计算一下就可以。第二种优化就是：这个激活值我计算出来了你也不是立马就用，那么我就先把存储到CPU里面，等用的时候再从CPU里面去加载回来。

Temporary Buffer的优化：模型训练过程中经常会创建一些大小不等的临时缓冲区，比如对梯度进行AllReduce啥的，解决办法就是预先创建一个固定的缓冲区，训练过程中不再动态创建，如果要传输的数据较小，则多组数据bucket后再一次性传输，提高效率。

Unusable Fragmented Memory的优化：显存出现碎片的一大原因是gradient checkpointing后，不断地创建和销毁那些不保存的激活值，解决方法是预先分配一块连续的显存，将常驻显存的模型状态和checkpointed activation存在里面，剩余显存用于动态创建和销毁 activation

08、第二篇论文：ZeRO-Offload

一张卡训不了大模型，根因是显存不足，ZeRO-Offload的想法很简单：显存不足，内存来补。



9、第二篇论文: ZeRO-Offload

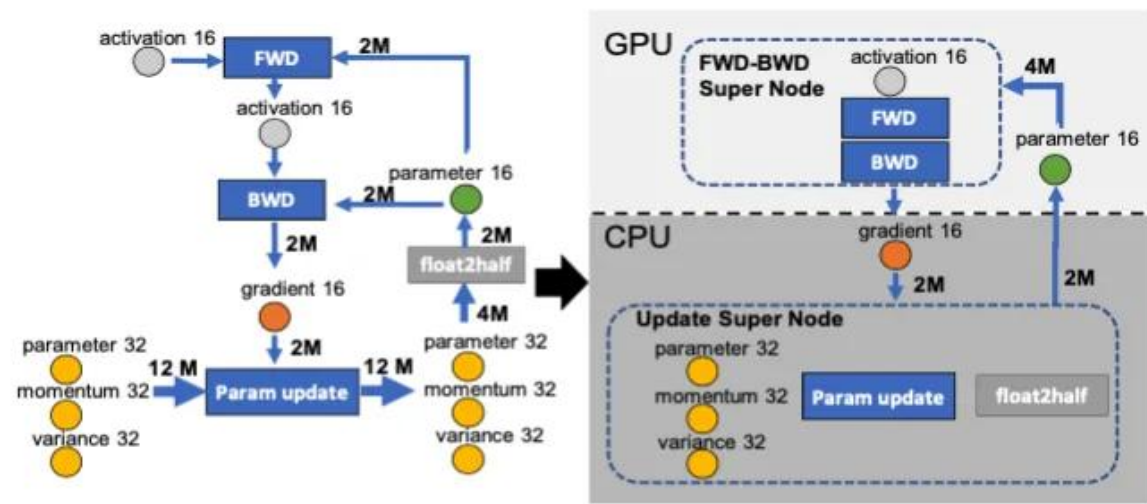


Figure 2: The dataflow of fully connected neural networks with M parameters. We use activation checkpoint to reduce activation memory to avoid activation migration between CPU and GPU.

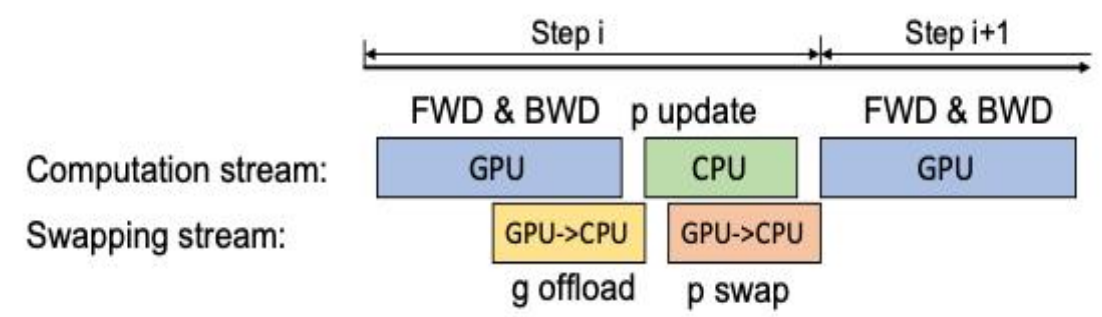
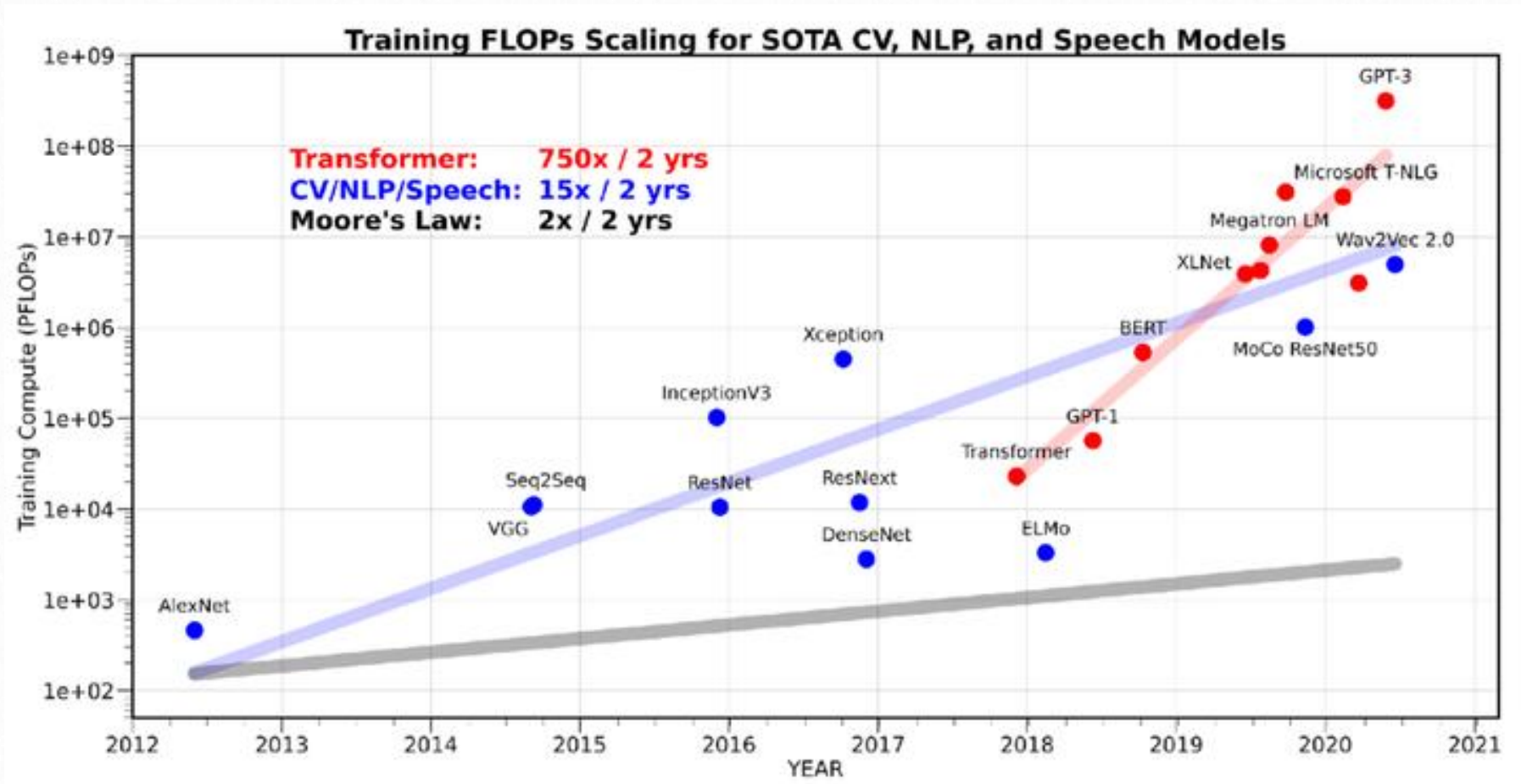


Figure 3: ZeRO-Offload training process on a single GPU.

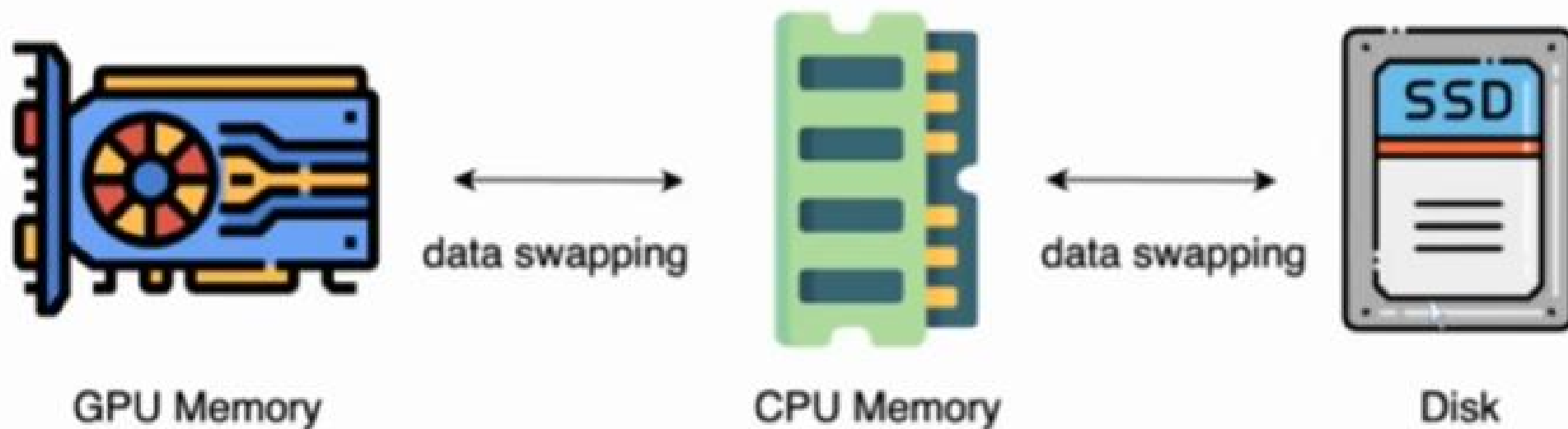
10、第三篇论文：ZeRO-Infinity



从GPT-1到GPT-3，两年时间内模型参数0.1B增加到175B，而同期，NVIDIA交出的成绩单是从V100的32GB显存增加A100的80GB

ZeRO-Infinity 允许通过使用 NVMe 固态硬盘扩展 GPU 和 CPU 内存来训练大型模型

11、DeepSpeed总结



论文地址:

<https://arxiv.org/pdf/2101.06840.pdf>

<https://arxiv.org/abs/1910.02054>

<https://arxiv.org/pdf/2104.07857.pdf>

01、资源不够怎么办？

微调量化版本

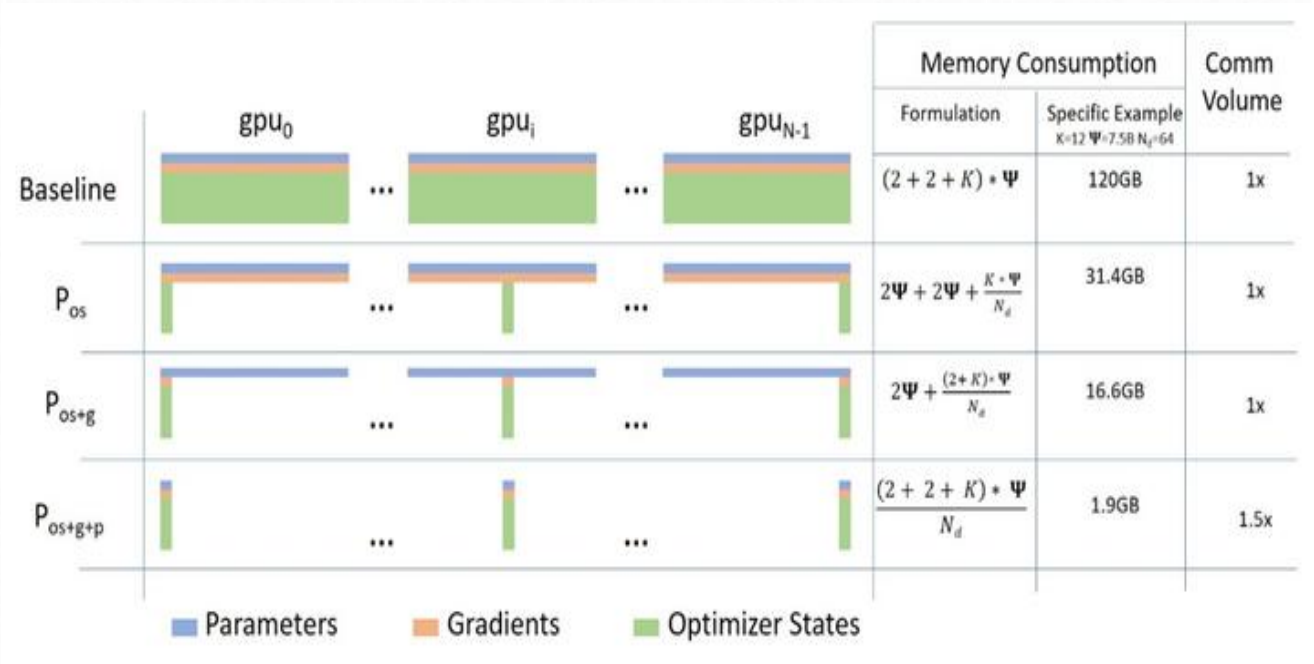
如果资源不够的，可以微调量化的版本，需要在微调脚本里添加如下参数即可：

--quantization_bit 8 或 --quantization_bit 4

多卡微调

```
7 set -ex
8
9 PRE_SEQ_LEN=128
10 LR=2e-2
11 NUM_GPUS=2 修改GPU数量
12 MAX_SOURCE_LEN=1024
13 MAX_TARGET_LEN=128
14 DEV_BATCH_SIZE=1
15 GRAD_ACCUMULARION_STEPS=32
16 MAX_STEP=500
17 SAVE_INTERVAL=500
```

DeepSpeed方案



02、资源准备

多卡微调+DeepSpeed方案

```
root@autodl-container-5e96488cf6-c1d638a7:~# nvidia-smi
Fri Jan 19 13:14:44 2024

+-----+
| NVIDIA-SMI 535.104.05                Driver Version: 535.104.05   CUDA Version: 12.2   |
+-----+-----+
| GPU Name                               Persistence-M   Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf    Pwr:Usage/Cap       Memory-Usage | GPU-Util  Compute M. |
|=====+=====+
| 0  NVIDIA GeForce RTX 4090             On          00000000:17:00.0 Off |          Off         |
| 0%   24C   P8             27W / 450W      2MiB / 24564MiB |      0%   Default   |
|                               MIG M. |
+-----+-----+
| 1  NVIDIA GeForce RTX 4090             On          00000000:32:00.0 Off |          Off         |
| 0%   33C   P8             14W / 450W      2MiB / 24564MiB |      0%   Default   |
|                               MIG M. |
+-----+-----+

+-----+
| Processes: |
| GPU   GI   CI        PID   Type   Process name                      GPU Memory |
| ID   ID                                     Usage      |
+-----+-----+
| No running processes found |
+-----+

root@autodl-container-5e96488cf6-c1d638a7:~#
```

```
root@autodl-container-5e96488cf6-c1d638a7:~/autodl-tmp/chatglm/ChatGLM3/finetune_chatmodel_demo/scripts# nvidia-smi
Fri Jan 19 13:45:56 2024

+-----+
| NVIDIA-SMI 535.104.05                Driver Version: 535.104.05   CUDA Version: 12.2   |
+-----+-----+
| GPU Name                               Persistence-M   Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf    Pwr:Usage/Cap       Memory-Usage | GPU-Util  Compute M. |
|=====+=====+
| 0  NVIDIA GeForce RTX 4090             On          00000000:17:00.0 Off |          Off         |
| 30%  51C   P2             340W / 450W    16863MiB / 24564MiB |    100%   Default   |
|                               MIG M. |
+-----+-----+
| 1  NVIDIA GeForce RTX 4090             On          00000000:32:00.0 Off |          Off         |
| 30%  46C   P2             332W / 450W    16863MiB / 24564MiB |     96%   Default   |
|                               MIG M. |
+-----+-----+

+-----+
| Processes: |
| GPU   GI   CI        PID   Type   Process name                      GPU Memory |
| ID   ID                                     Usage      |
+-----+-----+
| No running processes found |
+-----+
```

- 镜像 PyTorch 2.0.0 Python 3.8(ubuntu20.04) Cuda 11.8 [更换](#)
- GPU RTX 4090(24GB) * 2 [升降配置](#)
- CPU 24 vCPU Intel(R) Xeon(R) Platinum 8352V CPU @ 2.10GHz
- 内存 180GB
- 硬盘 系统盘: 30 GB
数据盘: 免费:50GB 付费:280GB [扩容](#) [缩容](#)
- 附加磁盘 无
- 端口映射 无
- 网络 同一地区实例共享带宽

03、多卡部署遇到的问题

NotImplementedError: Using RTX 3090 or 4000 series doesn't support faster communication broadband via P2P or IB. Please set `NCCL_P2P_DISABLE="1"` and `NCCL_IB_DISABLE="1"` or use `accelerate launch` which will do this automatically.

这个错误是因为在使用RTX 3090或4000系列显卡时，Nvidia Collective Communications Library (NCCL) 尝试使用点对点 (P2P) 或InfiniBand (IB) 进行通信，然而这些显卡不支持这些通信方式。解决这个问题的方法是禁用这些通信方式。

你可以设置以下的环境变量：

```
export NCCL_P2P_DISABLE=1
export NCCL_IB_DISABLE=1
```

bash 复制代码

你也可以通过加入上述行至你的bashrc文件来使这些设置持久化。你可以使用以下的命令打开bashrc文件：

```
nano ~/.bashrc
```

bash 复制代码

然后在文件尾部添加上述两行，保存并退出。然后，你还需要运行以下命令使更改生效：

```
source ~/.bashrc
```

bash 复制代码

执行这些步骤后，你应该就能够正常使用你的显卡了。

04、高效微调配置文件修改

```
finetune_pt.sh 终端 1 终端 2 +
1  #!/usr/bin/env bash
2
3  export NCCL_P2P_DISABLE=1
4  export NCCL_IB_DISABLE=1
5
6
7  set -ex
8
9  PRE_SEQ_LEN=128
10 LR=2e-2
11 NUM_GPUS=2
12 MAX_SOURCE_LEN=1024
13 MAX_TARGET_LEN=128
14 DEV_BATCH_SIZE=1
15 GRAD_ACCUMULATION_STEPS=32
16 MAX_STEP=500
17 SAVE_INTERVAL=500
18
19 DATESTR=`date +%Y%m%d-%H%M%S`
20 RUN_NAME=advertise_gen_pt
21
22 BASE_MODEL_PATH=/root/autodl-tmp/chatglm/model/chatglm3-6b/
23 DATASET_PATH=/root/autodl-tmp/chatglm/ChatGLM3/finetune_chatmodel_demo/formatted_data/advertise_gen.jsonl
24 OUTPUT_DIR=output2/${RUN_NAME}-${DATESTR}-${PRE_SEQ_LEN}-${LR}
25
26 mkdir -p $OUTPUT_DIR
27
28 torchrun --standalone --nnodes=1 --nproc_per_node=$NUM_GPUS finetune.py \
29     --deepspeed configs/deepspeed.json \
30     --train_format input-output \
31     --train_file $DATASET_PATH \
32     --preprocessing_num_workers 1 \
33     --model_name_or_path $BASE_MODEL_PATH \
34     --output_dir $OUTPUT_DIR \
35     --max_source_length $MAX_SOURCE_LEN \
36     --max_target_length $MAX_TARGET_LEN \
37     --per_device_train_batch_size $DEV_BATCH_SIZE \
```

添加参数

修改GPU数量

增加参数

05、deepspeed文件配置

```
{  
  "train_micro_batch_size_per_gpu": "auto",  
  "zero_allow_untested_optimizer": true,  
  "fp16": {  
    "enabled": "auto",  
    "loss_scale": 0,  
    "initial_scale_power": 16,  
    "loss_scale_window": 1000,  
    "hysteresis": 2,  
    "min_loss_scale": 1  
  },  
  "zero_optimization": {  
    "stage": 2,  
    "allgather_partitions": true,  
    "allgather_bucket_size": 5e8,  
    "overlap_comm": false,  
    "reduce_scatter": true,  
    "reduce_bucket_size": 5e8,  
    "contiguous_gradients": true  
  }  
}
```

"train_micro_batch_size_per_gpu": "auto" - 这个参数设置每个GPU的微型批处理大小。

"zero_allow_untested_optimizer": true - 这允许使用未经DeepSpeed测试的优化器。

"fp16": {...} - 这是用于半精度浮点数（FP16）训练的配置。FP16训练可以减少内存使用并提高计算效率。

"enabled": "auto" - 这设置了是否启用FP16训练。

"loss_scale": 0、"initial_scale_power": 16、"loss_scale_window": 1000、"hysteresis": 2、

"min_loss_scale": 1 - 这些参数用于控制FP16训练中的损失缩放。

"zero_optimization": {...} - 这是用于Zero Redundancy Optimizer (ZeRO)的配置。ZeRO是一种旨在减少模型训练过程中的内存占用的优化策略。

"stage": 2 - 这设置了ZeRO的阶段。可以大大降低内存需求。

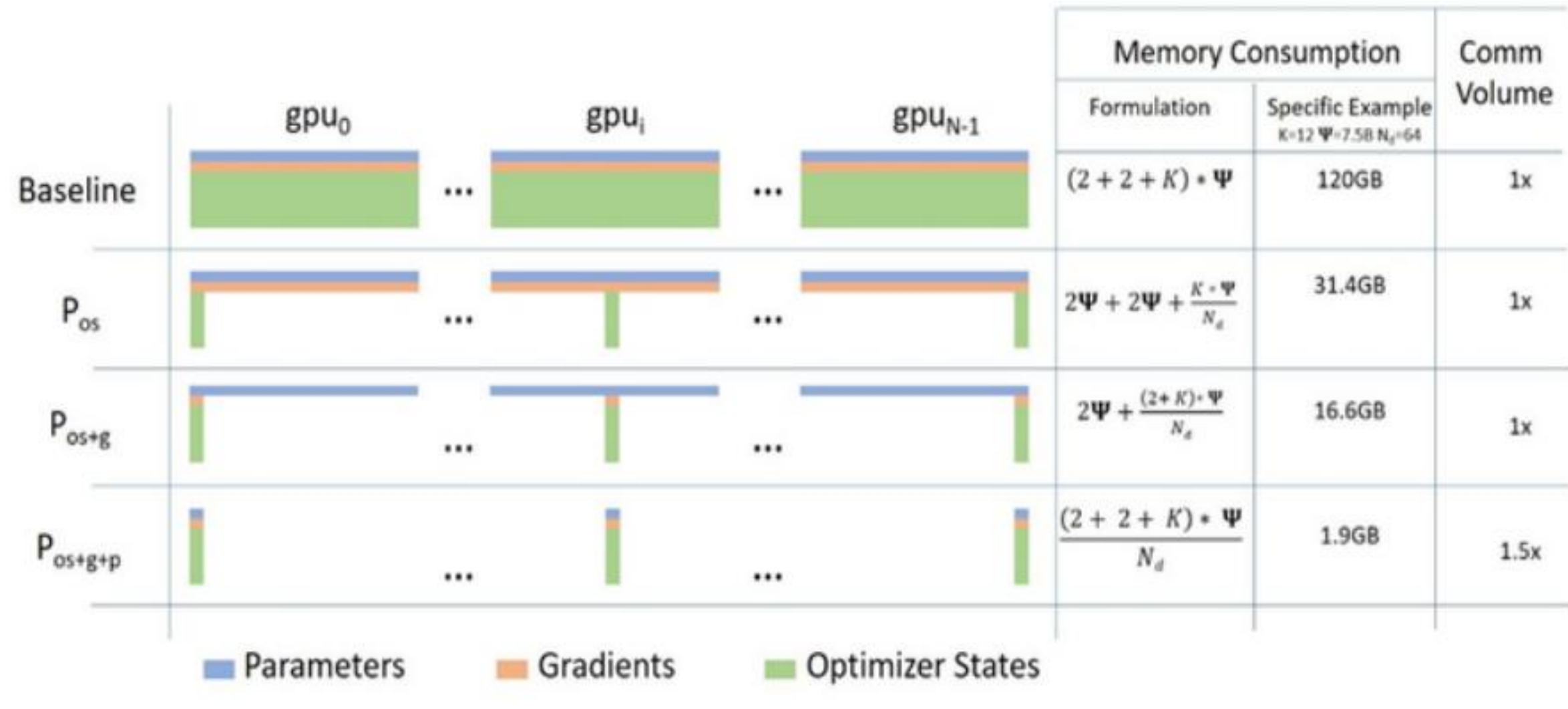
"allgather_partitions": true、"allgather_bucket_size": 5e8 - 这些参数控制了allgather操作，它在所有GPU上收集数据。

"overlap_comm": false - 这设置是否在计算和通信间重叠以提高效率。

"reduce_scatter": true、"reduce_bucket_size": 5e8 - 这些参数控制reduce-scatter操作，它在所有GPU上分散数据。

"contiguous_gradients": true - 这设置是否使梯度连续，以提高内存访问的效率。

06、 stage



07、多卡微调效果验证

```
python inference.py \
  --pt-checkpoint /root/autodl-tmp/chatglm/ChatGLM3/finetune_chatmodel_demo/output3/advertise_gen_pt-20240126-150806-128-2e-2 \
  --model "/root/autodl-tmp/chatglm/model/chatglm3-6b/"
```

```
root@autodl-container-cb2246bbbe-1e711273:~/autodl-tmp/chatglm/ChatGLM3/finetune_chatmodel_demo# python inference.py \
> --pt-checkpoint /root/autodl-tmp/chatglm/ChatGLM3/finetune_chatmodel_demo/output3/advertise_gen_pt-20240126-150806-128-2e-2 \
> --model "/root/autodl-tmp/chatglm/model/chatglm3-6b/"
Loading checkpoint shards: 100% |██████████████████████████████████████████████████████████████████████████████| 7/7 [00:02<00:00, 3.04it/s]
Some weights of ChatGLMForConditionalGeneration were not initialized from the model checkpoint at /root/autodl-tmp/chatglm/model/chatglm3-6b/ and are newly in
itialized: ['transformer.prefix_encoder.embedding.weight']
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
Prompt:裙型#鱼尾裙
Response: 裙摆处采用鱼尾裙摆设计，让裙子看起来更加飘逸，行走间更显灵动。裙摆处采用荷叶边设计，让裙子看起来更加飘逸，行走间更显灵动。裙身采用<UNK>设计，让裙子
看起来更加飘逸。
Prompt:
```

关注视频号：玄姐谈AGI
助力数字化人才提升 AIGC 能力



玄姐谈 AGI 



扫一扫二维码，关注我的视频号