# 使用deepspeed训练案例

|  |  |  |  |
| --- | --- | --- | --- |
| **案例** | **功能** | **代码位置** | **未完成/暂不支持的功能** |
| OpenLLaMA2: DeepSpeed+Ray based LLaMA2 RLHF/RS training framework | Develop a fast LLaMA2 SFT/PPO Training Framework based on DeepSpeed. 尚未支持rlhf | <https://github.com/OpenLLMAI/OpenLLaMA2/tree/main> | 1.train a [chinese-llama2 ↗](https://github.com/OpenLLMAI/chinese-llama2) RLHF model  2.QLora  3.Multi-nodes RLHF based on Ray  4.Multiple RM models |
| DeepSpeed框架与ChatGLM微调模型相结合 |  | <https://mp.weixin.qq.com/s?__biz=MzkzNTEzNzA3NA==&mid=2247484328&idx=1&sn=97d4729a832c0a11b36ccfa0b89c4162&chksm=c2b3d9f5f5c450e313ccd690a569637f1a8b674b3cb77c0c4c36f6cf16b9f68afbcc58f56b61&scene=21#wechat_redirect> |  |
| 基于transformers和deepspeed对llama（7-65B）模型进行微调，并提供简单方便的训练脚本 |  | <https://github.com/HuangLK/transpeeder> |  |
| llama-7b-hf Tuning with [Stanford Alpaca](https://github.com/tatsu-lab/stanford_alpaca) Dataset using Deepspeed and Transformers |  | <https://github.com/lxe/llama-tune> |  |
| [alpaca-rlhf](https://github.com/l294265421/alpaca-rlhf)  Finetuning LLaMA with RLHF (Reinforcement Learning with Human Feedback) based on DeepSpeed Chat | RLHF ，DeepSpeed | <https://github.com/l294265421/alpaca-rlhf> |  |
| deepspeed+trainer简单高效实现多卡微调大模型ChatGLM | 实现multi\_gpu + zero 微调ChatGLM, 目前已实现LoRA, Ptuning-v2, Freeze三种微调方式 | <https://github.com/CSHaitao/ChatGLM_mutli_gpu_tuning> |  |
| LexiLaw - 中文法律大模型 | 三种方式对 ChatGLM-6B 进行了深度微调,所有的模型都是在 7 张40G A100上训练模型，训练代码使用DeepSpeed和Trainer | <https://github.com/CSHaitao/LexiLaw> | 使用RLHF进行训练 |
| 多模态预训练框架 TencentPretrain 现已支持 LoRA 和 DeepSpeed ZeRO stage1和2、ZeRO-3 并行。 | 单机 8 \* A100 40G 训练 LLaMA 30B 在 512 seq length 时能达到 256 batch size | <https://github.com/Tencent/TencentPretrain> |  |
| TencentPretrain基于 ZeRO-3，训练了中文 LLaMA 基础模型和 ChatLLaMA 对话模型，数据、代码和模型权重均公开 |  | <https://github.com/CVI-SZU/Linly> |  |
| [LLaMA-Efficient-Tuning](https://github.com/hiyouga/LLaMA-Efficient-Tuning/blob/main/README_zh.md)。Easy-to-use LLM fine-tuning framework (LLaMA-2, BLOOM, Falcon, Baichuan, Qwen, ChatGLM2) | 支持预训练、指令监督微调、奖励模型训练、PPO 训练、DPO 训练、LoRA、QLoRA  暂不支持RLHF | <https://github.com/hiyouga/LLaMA-Efficient-Tuning/blob/main/README_zh.md> |  |
| Baichuan-7B |  | <https://github.com/baichuan-inc/Baichuan-7B> |  |
| [Baichuan-13B](https://github.com/baichuan-inc/Baichuan-13B) | deepspeed 的环境下进行了全量微调sft | <https://github.com/baichuan-inc/Baichuan-13B> |  |
| [deepspeed chat替换模型和数据](https://link.zhihu.com/?target=https%3A//techdiylife.github.io/big-model-training/deepspeed/deepspeed-chat-model-data.html)成BLOOMZ |  | <https://techdiylife.github.io/big-model-training/deepspeed/deepspeed-chat-model-data.html#2-%E6%9B%BF%E6%8D%A2%E6%A8%A1%E5%9E%8B> |  |

DeepSpeed has been used to train many different large-scale models, below is a list of several examples that we are aware of :  
DeepSpeed 已被用来训练许多不同的大型模型，下面列出了已知道的几个示例：

* [Megatron-Turing NLG (530B) 威震天-图灵 NLG (530B)](https://www.microsoft.com/en-us/research/blog/using-deepspeed-and-megatron-to-train-megatron-turing-nlg-530b-the-worlds-largest-and-most-powerful-generative-language-model/)
* [Jurassic-1 (178B)](https://uploads-ssl.webflow.com/60fd4503684b466578c0d307/61138924626a6981ee09caf6_jurassic_tech_paper.pdf)
* [BLOOM (176B)](https://huggingface.co/blog/bloom-megatron-deepspeed)
* [GLM (130B)](https://github.com/THUDM/GLM-130B)
* [xTrimoPGLM (100B)](https://www.biorxiv.org/content/10.1101/2023.07.05.547496v2)
* [YaLM (100B)](https://github.com/yandex/YaLM-100B)
* [GPT-NeoX (20B)](https://github.com/EleutherAI/gpt-neox)
* [AlexaTM (20B)](https://www.amazon.science/blog/20b-parameter-alexa-model-sets-new-marks-in-few-shot-learning)
* [Turing NLG (17B)  图灵 NLG (17B)](https://www.microsoft.com/en-us/research/blog/turing-nlg-a-17-billion-parameter-language-model-by-microsoft/)
* [METRO-LM (5.4B)](https://arxiv.org/pdf/2204.06644.pdf)

deepspeed 官方使用文档 <https://www.deepspeed.ai/>

[hugging face 使用文档 https://huggingface.co/docs/transformers/main/main\_classes/deepspeed](https://huggingface.co/docs/transformers/main/main_classes/deepspeed#troubleshooting)

[DeepSpeed](https://github.com/microsoft/DeepSpeed) implements everything described in the [ZeRO paper](https://arxiv.org/abs/1910.02054). Currently it provides full support for:

1. Optimizer state partitioning (ZeRO stage 1)
2. Gradient partitioning (ZeRO stage 2)
3. Parameter partitioning (ZeRO stage 3)
4. Custom mixed precision training handling
5. A range of fast CUDA-extension-based optimizers
6. ZeRO-Offload to CPU and NVMe

DeepSpeed ZeRO-2 is primarily used only for training, as its features are of no use to inference.

DeepSpeed ZeRO-3 can be used for inference as well, since it allows huge models to be loaded on multiple GPUs, which won’t be possible on a single GPU.

# 经验

DeepSpeed，通过将模型参数拆散分布到各个GPU上，以实现大型模型的计算，弥补了DDP的缺点，非常方便，这也就意味着我们能用更少的GPU训练更大的模型，而且不受限于显存。

使用DeepSpeed其实和写一个pytorch模型只有部分区别，一开始的流程是一样的

1. pip install deepspeed

2. 初始化Deepspeed：DeepSpeed 通过输入参数来启动训练。关键在于配置好config.json文件

DeepSpeed has direct integrations with [HuggingFace Transformers](https://github.com/huggingface/transformers) and [PyTorch Lightning](https://github.com/PyTorchLightning/pytorch-lightning). HuggingFace Transformers users can now easily accelerate their models with DeepSpeed through a simple --deepspeed flag + config file [See more details](https://huggingface.co/docs/transformers/main_classes/deepspeed). PyTorch Lightning provides easy access to DeepSpeed through the Lightning Trainer [See more details](https://pytorch-lightning.readthedocs.io/en/stable/advanced/multi_gpu.html?highlight=deepspeed#deepspeed).

Transformers integrates [DeepSpeed](https://github.com/microsoft/DeepSpeed) via 2 options:

1. Integration of the core DeepSpeed features via [Trainer](https://huggingface.co/docs/transformers/main/en/main_classes/trainer#transformers.Trainer). This is an everything-done-for-you type of integration - just supply your custom config file or use our template and you have nothing else to do. Most of this document is focused on this feature.
2. If you don’t use [Trainer](https://huggingface.co/docs/transformers/main/en/main_classes/trainer#transformers.Trainer) and want to use your own Trainer where you integrated DeepSpeed yourself, core functionality functions like from\_pretrained and from\_config include integration of essential parts of DeepSpeed like zero.Init for ZeRO stage 3 and higher. To tap into this feature read the docs on [non-Trainer DeepSpeed Integration](https://huggingface.co/docs/transformers/main/main_classes/deepspeed#nontrainer-deepspeed-integration).

also. You don’t have to use the [Trainer](https://huggingface.co/docs/transformers/main/en/main_classes/trainer#transformers.Trainer) to use DeepSpeed with Transformers - you can use any model with your own trainer, and you will have to adapt the latter according to [the DeepSpeed integration instructions](https://www.deepspeed.ai/getting-started/#writing-deepspeed-models).

可能的坑：

1. 多机多卡跑不起来，多机间不能通信，报错如下 Call to connect returned Connection refused, retrying

解决方案：

deepspeed环境依赖问题，重新装pytorch依赖；

Deepspeed多卡多机ZeRo-3训练踩的坑

2. AttributeError: 'DeepSpeedCPUAdam' object has no attribute 'ds\_opt\_adam'

解决方法：在训练文件头部加入

import deepspeed

deepspeed.ops.op\_builder.CPUAdamBuilder().load()

3. Error building extension 'cpu\_adam'

解决方法：[https://github.com/microsoft/DeepSpeed/issues/2268#issuecomment-1230830048](https://link.zhihu.com/?target=https%3A//github.com/microsoft/DeepSpeed/issues/2268%23issuecomment-1230830048)

注意：如果你是python3.x版本的最好安装python3.x版本的python-dev

sudo apt-get install python3.9-dev

4. Ninja is required to load C++ extension

解决方法：在文件头部加入

import os

local\_env = os.environ.copy()

local\_env["PATH"]="/home/xxxx/.conda/envs/xxx/bin:" + local\_env["PATH"]

os.environ.update(local\_env)

5. DeepSpeed会通过占用CPU内存来减缓GPU的开销，当系统CPU不够的时候，DeepSpeed进程就会自动被系统停止，造成没有任何报错，DeepSpeed无法启动的现象。建议先用上文介绍的estimation估计一下CPU内存占用，然后用free -h查看一下机器的CPU内存空余量，来判断能否使用DeepSpeed

the deepspeed process gets killed at startup without a traceback

solution:

If the deepspeed process gets killed at launch time without a traceback, that usually means that the program tried to allocate more CPU memory than your system has or your process is allowed to allocate and the OS kernel killed that process. This is because your configuration file most likely has either offload\_optimizer or offload\_param or both configured to offload to cpu. If you have NVMe, experiment with offloading to NVMe if you’re running under ZeRO-3. Here is how you can [estimate how much memory is needed for a specific model](https://deepspeed.readthedocs.io/en/latest/memory.html).

6. training and/or eval/predict loss is NaN

solution:

This often happens when one takes a model pre-trained in bf16 mixed precision mode and tries to use it under fp16 (with or without mixed precision). Most models trained on TPU and often the ones released by Google are in this category (e.g. almost all t5-based models). Here the solution is to either use fp32 or bf16 if your hardware supports it (TPU, Ampere GPUs or newer).

The other problem may have to do with using fp16. When you configure this section:

{

"fp16": {

"enabled": "auto",

"loss\_scale": 0,

"loss\_scale\_window": 1000,

"initial\_scale\_power": 16,

"hysteresis": 2,

"min\_loss\_scale": 1

}

}

and you see in your log that Deepspeed reports OVERFLOW!

that means that the Deepspeed loss scaler can’t figure out a scaling co-efficient that overcomes loss overflow.

In this case you usually need to raise the value of initial\_scale\_power. Setting it to "initial\_scale\_power": 32 will typically resolve the problem.

7. While DeepSpeed has a pip installable PyPI package, it is highly recommended that it gets installed from [source](https://github.com/microsoft/deepspeed#installation) to best match your hardware and also if you need to enable certain features, like 1-bit Adam, which aren’t available in the pypi distribution.

8. DeepSpeed works with the PyTorch [Trainer](https://huggingface.co/docs/transformers/main/en/main_classes/trainer#transformers.Trainer) but not TF TFTrainer. You don’t have to use the [Trainer](https://huggingface.co/docs/transformers/main/en/main_classes/trainer#transformers.Trainer) to use DeepSpeed with Transformers - you can use any model with your own trainer, and you will have to adapt the latter according to [the DeepSpeed integration instructions](https://www.deepspeed.ai/getting-started/#writing-deepspeed-models).

# 可行性

业界调研，现有的分布式训练框架对比

Megatron-LM: https://[github.com/NVIDIA/Megatron-LM](http://github.com/NVIDIA/Megatron-LM)

DeepSpeed: https://[github.com/microsoft/DeepSpeed](http://github.com/microsoft/DeepSpeed)

mesh-tensorflow: https://[github.com/tensorflow/mesh](http://github.com/tensorflow/mesh)

fairscale: https://[github.com/facebookresearch/fairscale](http://github.com/facebookresearch/fairscale)

选择微软开发的Deepspeed框架作为训练框架的主要原因是，它具备高效、有效、易使用等三个方面的特点。该框架是基于PyTorch构建的，因此可以简单修改以便进行迁移使用。

与其他 RLHF 系统（如 Colossal-AI 或由原生 PyTorch 提供支持的 HuggingFace）相比，DeepSpeed-RLHF 在系统性能和模型可扩展性方面表现出色

RLHF在性能方面是最具挑战性的部分。每次迭代都需要高效处理两个阶段：a) 生成回答的推理阶段，为训练提供输入；b) 更新 actor 和 reward 模型权重的训练阶段，以及它们之间的交互和调度。这引入了两个主要困难：（1）内存成本，因为在第三阶段的整个过程中需要运行多个 SFT 和 RW 模型；（2）生成回答阶段的速度较慢，如果没有正确加速，将显著拖慢整个第三阶段。

将 DeepSpeed 训练和推理的系统功能整合为一个统一的基础设施，称为混合引擎（Hybrid Engine）。它利用原始 DeepSpeed 引擎进行高速训练模式，同时轻松应用 DeepSpeed 推理引擎进行生成 / 评估模式，DeepSpeed 训练和推理引擎之间的过渡是无缝的。

在训练执行过程中，混合引擎使用了多种内存优化技术，如 DeepSpeed 的 ZeRO 系列技术和的 LoRA 方法（DeepSpeed 和 ZeRO（Zero Redundancy Optimizer，零冗余优化器），前者是一个开源的深度学习训练优化库，后者是该库中的一种新型内存优化技术。ZeRO 支持的数据并行、流水线并行和张量切片模型并行。为具有超过 **一万亿** 个参数的 **超大型模型** 提供支持）