Inspiration Day Report

How to Train the Really Large Model on Many GPUs?

DeepSpeed: Extreme-scale model training for everyone

2022-03-17 15:00-15:30

主讲人：孟令辉　指导教师：徐波

简介：**DeepSpeed框架：由微软和英伟达联合推出的，在万亿参数预训练大模型目标下的高效训练框架**

融合多种优化手段，包括课程学习、高效通信的优化器以及MoE等关键技术构建适用于大规模数据和模型参数的训练框架DeepSpeed。该方法可以将Bert-large从原有训练需要的64GPU的代价，降低到只需要单卡V100即可训练收敛。

----大规模数据和集群如何高效训练“大模型”

How to Train the Really Large Model on Many GPUs?

DeepSpeed: Extreme-scale model training for everyone

Linghui Meng

Institute of Automation, Chinese Academy of Sciences, CASIA

2022.03.17

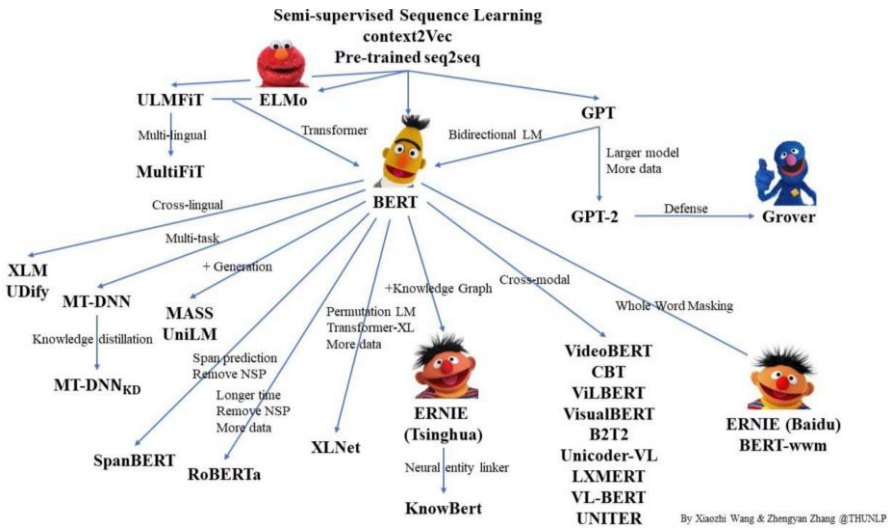
Outline

- Background & Motivation
- Critical Techniques
  - Distributed Training Parallelism （Data/Model/Pipeline Parallelism）
  - Optimizer Design for Saving Memory
  - Curriculum Learning for stabilizing the training process
  - Mixture-of-Experts, MoE
- References

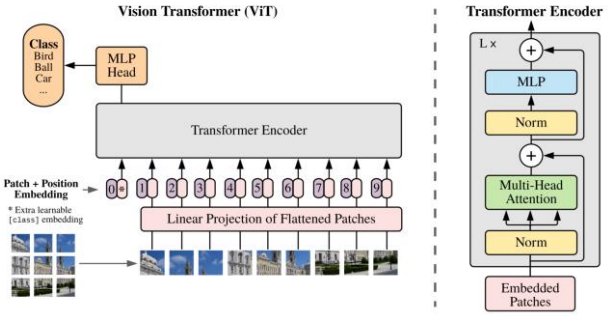
Background & Motivation

中国科学院 自动化研究所 INSTITUTE OF AUTOMATION CHINESE ACADEMY OF SCIENCES

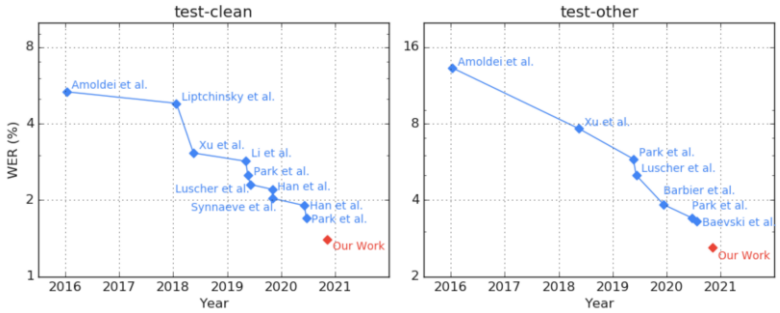
The pre-trained large model based on Transformer has been investigated much among Natural Language, Speech and Computer Vision, even Decision Making



(a) Pre-trained models in NLP



(b) Pre-trained models in CV



(c) Pre-trained models in Speech

Background & Motivation

中国科学院
自动化研究所
INSTITUTE OF AUTOMATION
CHINESE ACADEMY OF SCIENCES

The pre-trained large model based on Transformer has been investigated much among Natural Language, Speech and Computer Vision

What is pre-training?

- The training in advance of standard training

Why we need super-large model with pre-training?

- The standard training (data/model size) is not enough

What to learn in pre-training from the large model

- Representation learning: more general, self-supervised
- Task learning: more task specific, supervised

What problems the super-huge model use?

- Computing resource requirements
- Low efficiency across multiple GPUs to update the large model
- Data loading with low efficiency

So How to train a really large model on many GPUs with huge datasets

Critical Techniques – Distributed Training Parallelism

中国科学院 自动化研究所 INSTITUTE OF AUTOMATION CHINESE ACADEMY OF SCIENCES

Data parallelism (DP):

The simplest method is to copy the model on many workers and pass partial data to each worker. **But the problem emerges when the model overhead the memory of a single GPU.**

Model parallelism (MP):

MP aims to solve the case when the model weights cannot fit into a single node.

They use gradient accumulation for the model parallelism and only allocates a fraction of model parameters on one worker and thus both the memory usage and the computation are reduced.

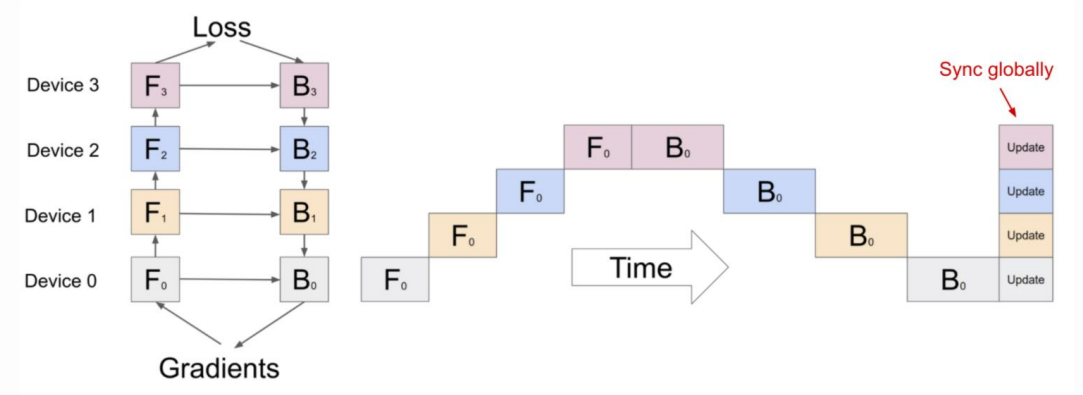


Figure: A naive model parallelism setup where the model is vertically split into 4 partitions.

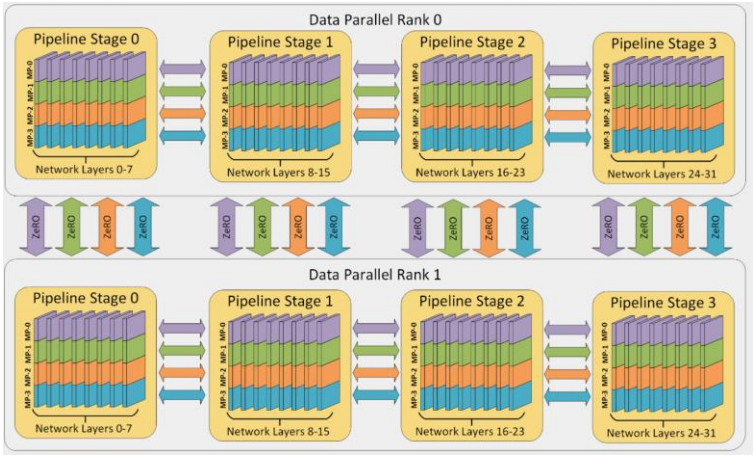
中国科学院 自动化研究所 INSTITUTE OF AUTOMATION CHINESE ACADEMY OF SCIENCES

Pipeline parallelism (PP):

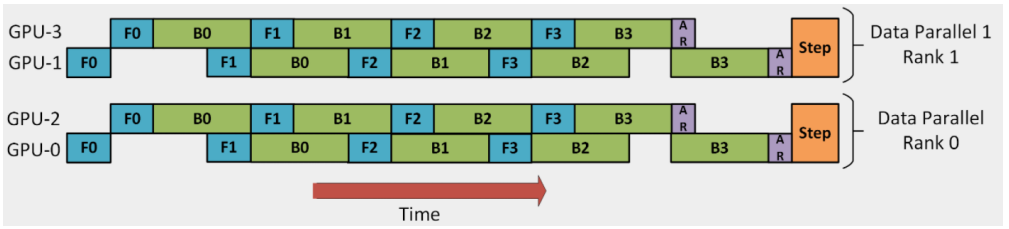
PP combines model parallelism with data parallelism to reduce inefficient time "bubbles''. The main idea is to split one minibatch into multiple micro batches and enable each stage worker to process one micro batch simultaneously

PP in DeepSpeed

DeepSpeed uses *gradient accumulation* to extract pipeline parallelism. Once a stage completes the forward pass for a micro-batch, the activation memory is communicated to the next stage in the pipeline



(a) 3D pipeline in DeepSpeed



(b) An illustration of how DeepSpeed will train a batch with eight micro-batches using hybrid two-way data parallelism and two-stage pipeline parallelism.

Critical Techniques – Optimizer Design for Saving Memory

中国科学院 自动化研究所 INSTITUTE OF AUTOMATION CHINESE ACADEMY OF SCIENCES

1-bit LAMB in DeepSpeed

Comparing with Adam optimizer, LAMB owns higher computing efficiency under the large batch size but non-frequent communication across workers. 1-bit Adam can reduce the communication cost but cannot be directly applied on LAMB.

They proposed a two stage algorithm (warmup & compression stage) shown as below:

LAMB optimizer can be viewed as Adam with adaptive layerwise learning rates, is an example of large batch optimization.

$$\begin{aligned}
\boldsymbol{m}_t^{(l)} =& \beta_1 \boldsymbol{m}_{t-1}^{(l)} + (1-\beta_1)\boldsymbol{g}_t^{(l)}, \boldsymbol{v}_t^{(l)} = \beta_2 \boldsymbol{v}_{t-1}^{(l)} + (1-\beta_2)(\boldsymbol{g}_t^{(l)})^2 \\
\boldsymbol{u}_t^{(l)} =& \frac{\boldsymbol{m}_t^{(l)}}{\sqrt{\boldsymbol{v}_t^{(l)}} + \eta}, c_t^{(l)} = \text{clip}\left(\frac{\|\boldsymbol{x}_{t-1}^{(l)}\|}{\|\boldsymbol{u}_t^{(l)}\|}, c_{min}, c_{max}\right) \\
\boldsymbol{x}_t^{(l)} =& \boldsymbol{x}_{t-1}^{(l)} - \gamma c_t^{(l)} \boldsymbol{u}_t^{(l)}.
\end{aligned}$$

Algorithm 1 1-bit LAMB

Initialize: $\boldsymbol{x}_0^{(l)}$, $\boldsymbol{m}_0^{(l)} = \mathbf{0}$, $\boldsymbol{v}_0^{(l)} = \mathbf{0}$, $c_{avg}^{(l)} = 0$ for each layer. Learning rate γ, initial error $\boldsymbol{\delta} = \mathbf{0}$, number of total iterations T, warm-up steps T_w, three decaying factor $\beta_1, \beta_2, \beta_3$ for LAMB's momentum, variance, and scaling coefficient. $r^{(l)} = 1$, $r_{min}, r_{max}, r_{threshold}$ for 1-bit LAMB.
Running the original LAMB in (1) for T_w steps, and at each step $c_{avg}^{(l)} = \beta_3 c_{avg}^{(l)} + (1-\beta_3)c_t^{(l)}$.
At the end of step T_w, for each layer store the variance term (defined as $\boldsymbol{v}_t^{(l)}$ in (1)) $\boldsymbol{v}_{T_w}^{(l)}$ while still keep updating $\boldsymbol{v}_t^{(l)}$ in the future steps. Also stop updating $c_{avg}^{(l)}$.
for $t = T_w, \ldots, T$ **do**
  (On i-th node)
  Randomly sample $\boldsymbol{\xi}_t^{(i)}$ and compute local stochastic gradient $\boldsymbol{g}_t^{(i)} := \nabla F_i(\boldsymbol{x}_t^{(i)}, \boldsymbol{\xi}_t^{(i)})$, and update the local momentum $\boldsymbol{m}_t^{(i)}$ according to $\boldsymbol{m}_t^{(i)} = \beta_1 \boldsymbol{m}_{t-1}^{(i)} + (1-\beta_1)\boldsymbol{g}_t^{(i)}$.
  Compress the fused momentum $\boldsymbol{m}_t^{(i)}$ into $\hat{\boldsymbol{m}}_t^{(i)} = \boldsymbol{C}_\omega\left[\boldsymbol{m}_t^{(i)} + \boldsymbol{\delta}_{t-1}^{(i)}\right]$, and update the compression error by $\boldsymbol{\delta}_t^{(i)} = \boldsymbol{m}_t^{(i)} + \boldsymbol{\delta}_{t-1}^{(i)} - \hat{\boldsymbol{m}}_t^{(i)}$.
  Send the $\hat{\boldsymbol{m}}_t^{(i)}$ to the server.
  (On server)
  Take the average over all $\hat{\boldsymbol{m}}_t^{(i)}$ it receives and compress it into $\overline{\boldsymbol{m}}_t = \boldsymbol{C}_\omega\left[\frac{1}{n}\sum_{j=1}^n \hat{\boldsymbol{m}}_t^{(i)} + \overline{\boldsymbol{\delta}}_{t-1}\right]$, and update the compression error accordingly by $\overline{\boldsymbol{\delta}}_t = \frac{1}{n}\sum_{j=1}^n \hat{\boldsymbol{m}}_t^{(i)} + \overline{\boldsymbol{\delta}}_{t-1} - \overline{\boldsymbol{m}}_t$.
  Send $\overline{\boldsymbol{m}}_t$ to all the workers.
  (On j-th node)
  Set $\boldsymbol{m}_t = \overline{\boldsymbol{m}}_t$.
  for the l-th layer **do**
    Reconstruct global gradient $\boldsymbol{g}_t^{(l)} = (\boldsymbol{m}_t^{(l)} - \beta_1 \boldsymbol{m}_{t-1}^{(l)})/(1-\beta_1)$.
    $\boldsymbol{v}_t^{(l)} = \beta_2 \boldsymbol{v}_{t-1}^{(l)} + (1-\beta_2)\left(\boldsymbol{g}_t^{(l)}\right)^2$.
    $r_t^{(l)} = \left\|\boldsymbol{v}_{T_w}^{(l)} / \boldsymbol{v}_t^{(l)}\right\|_\infty$.
    $r_t^{(l)} = \text{clip}\left(r_t^{(l)}, (1-r_{threshold}) \times r_{t-1}^{(l)}, (1+r_{threshold}) \times r_{t-1}^{(l)}\right)$.
    $r_t^{(l)} = \text{clip}\left(r_t^{(l)}, r_{min}, r_{max}\right)$.
    $c_t^{(l)} = r_t^{(l)} c_{avg}^{(l)}$.
    Update model of the l-th layer $\boldsymbol{x}_t^{(l)} = \boldsymbol{x}_{t-1}^{(l)} - \gamma c_t^{(l)} \frac{\boldsymbol{m}_t^{(l)}}{\sqrt{\boldsymbol{v}_{T_w}^{(l)}}}$.
  end for
end for
Output: $\boldsymbol{x}$.

Critical Techniques – Optimizer Design for Saving Memory

中国科学院 自动化研究所 INSTITUTE OF AUTOMATION CHINESE ACADEMY OF SCIENCES

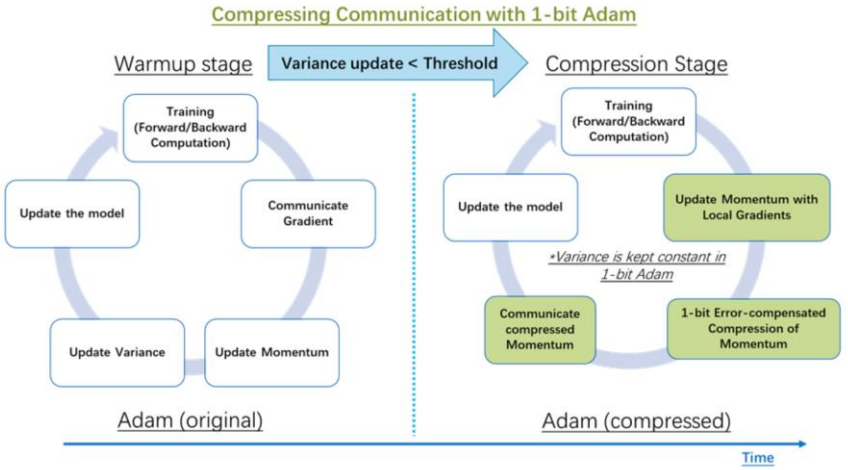
1-bit LAMB in DeepSpeed

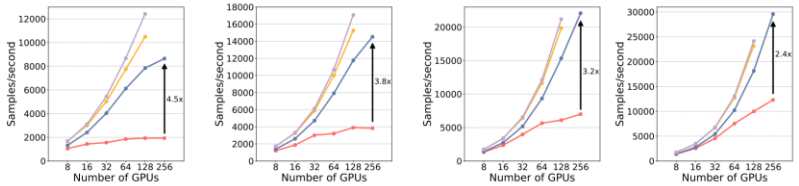
Comparing with Adam optimizer, LAMB owns higher computing efficiency under the large batch size but non-frequent communication across workers. 1-bit Adam can reduce the communication cost but cannot be directly applied on LAMB.

They proposed a two-stage algorithm (warmup & compression stage) shown as below:

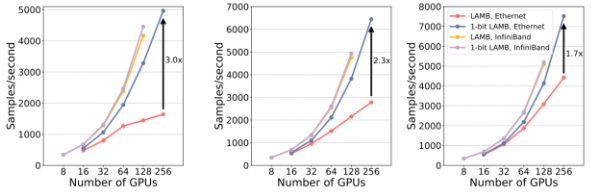
LAMB optimizer can be viewed as Adam with adaptive layerwise learning rates, is an example of large batch optimization.

$$
\begin{aligned}
\boldsymbol{m}_t^{(l)} &= \beta_1 \boldsymbol{m}_{t-1}^{(l)} + (1-\beta_1)\boldsymbol{g}_t^{(l)}, \boldsymbol{v}_t^{(l)} = \beta_2 \boldsymbol{v}_{t-1}^{(l)} + (1-\beta_2)(\boldsymbol{g}_t^{(l)})^2 \\
\boldsymbol{u}_t^{(l)} &= \frac{\boldsymbol{m}_t^{(l)}}{\sqrt{\boldsymbol{v}_t^{(l)} + \eta}}, c_t^{(l)} = \text{clip}\left(\frac{\|\boldsymbol{x}_{t-1}^{(l)}\|}{\|\boldsymbol{u}_t^{(l)}\|}, c_{min}, c_{max}\right) \\
\boldsymbol{x}_t^{(l)} &= \boldsymbol{x}_{t-1}^{(l)} - \gamma c_t^{(l)} \boldsymbol{u}_t^{(l)}.
\end{aligned}
$$





(a) seqlen 128, batch size 8K (b) seqlen 128, batch size 16K (c) seqlen 128, batch size 32K (d) seqlen 128, batch size 64K



(e) seqlen 512, batch size 8K (f) seqlen 512, batch size 16K (g) seqlen 512, batch size 32K

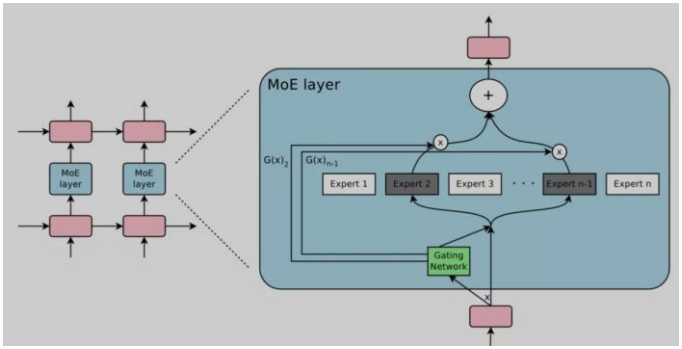
中国科学院 自动化研究所 INSTITUTE OF AUTOMATION CHINESE ACADEMY OF SCIENCES

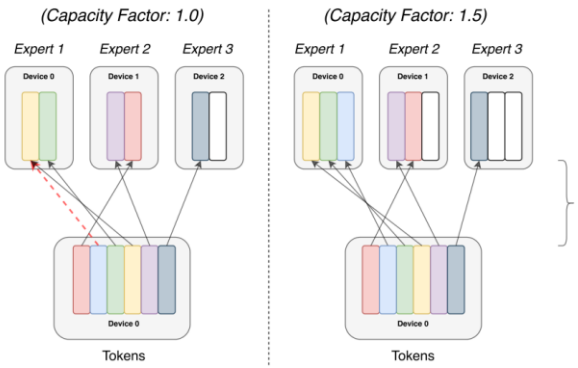
Mixture-of-Expert Routing, **MoE**

Shazeer et al. proposed a natural language Mixture-of-Experts (MoE) layer which takes as an input a token representation x and then routes this to the best determined top- k experts, selected from a set $\{E_i(x)\}_i^N$ of N experts.

MoE in DeepSpeed

Referring to Switch Transformer, depending on the gating outputs, not every expert has to be evaluated. When the number of experts is too large, we can consider using a two-level hierarchical MoE. For the trade-off between the performance and the capacity





$$\text{expert capacity} = \left(\frac{\text{tokens per batch}}{\text{number of experts}}\right) \times \text{capacity factor}$$

$$\text{loss} = \alpha N \cdot \sum_{i=1}^{N} f_i \cdot P_i$$

$$f_i = \frac{1}{T} \sum_{x \in \mathcal{B}} \mathbb{1}\{\text{argmax } p(x), i\}$$

$$P_i = \frac{1}{T} \sum_{x \in \mathcal{B}} p_i(x) \qquad p_i(x) = \frac{e^{h(x)_i}}{\sum_j^N e^{h(x)_j}}$$

Critical Techniques – Curriculum Learning for stabilizing

中国科学院 自动化研究所 INSTITUTE OF AUTOMATION CHINESE ACADEMY OF SCIENCES

Take the sequence length as the difficulty metrics

They start from shorter sequence training data, then gradually increase the sequence length) for two reasons.

(1) Sequence length as the curriculum difficulty metric has been proven to be effective in NLP;

(2) It can reduce the time complexity

But supporting the curriculum requires reordering the whole datasets causing the additional cost.

They truncate the raw text into sequences with the same length to form a mini-batch with a pacing function as below

Pacing function: step-wise linear

Given a starting sequence length $seqlen_1$, an ending sequence length $seqlen_2$ (baseline full sequence length), and a curriculum duration T (number of steps), the sequence length used for the training batch at step t is

$$seqlen_t = seqlen_1 + (seqlen_2 - seqlen_1) \times \min(\frac{t}{T}, 1)$$

Critical Techniques – Curriculum Learning for stabilizing

中国科学院 自动化研究所 INSTITUTE OF AUTOMATION CHINESE ACADEMY OF SCIENCES

Take the sequence length as the difficulty metrics

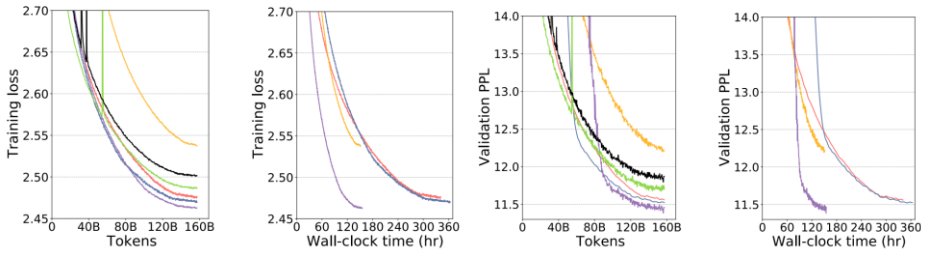
But supporting the curriculum requires reordering the whole datasets causing the additional cost.

They truncate the raw text into sequences with the same length to form a mini-batch with a pacing function as below

Pacing function: step-wise linear

Given a starting sequence length $seqlen_1$, an ending sequence length $seqlen_2$ (baseline full sequence length), and a curriculum duration T (number of steps), the sequence length used for the training batch at step t is

$$seqlen_t = seqlen_1 + (seqlen_2 - seqlen_1) \times \min(\frac{t}{T}, 1)$$



(a) Token-wise training loss (b) Time-wise training loss (c) Token-wise validation perplexity (d) Time-wise validation perplexity

(e) Step-wise Adam variance norm

(f) Step-wise Adam variance max element

Figure: Training GPT-2 1.5B Seqlen 1k pre-training

Brainstorm Discussion

中国科学院
自动化研究所
INSTITUTE OF AUTOMATION
CHINESE ACADEMY OF SCIENCES

What techniques we can borrow?

MoE、Curriculum Learning

How DeepSpeed help us to train a model?

Pre-trained Model with Large scale

When the pre-trained model fails?

References

[1] Li C, Zhang M, He Y. Curriculum Learning: A Regularization Method for Efficient and Stable Billion-Scale GPT Model Pre-Training[J]. arXiv preprint arXiv:2108.06084, 2021.

[2] Li C, Awan A A, Tang H, et al. 1-bit LAMB: Communication Efficient Large-Scale Large-Batch Training with LAMB's Convergence Speed[J]. arXiv preprint arXiv:2104.06069, 2021.

[3] Fedus W, Zoph B, Shazeer N. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity[J]. arXiv preprint arXiv:2101.03961, 2021.

[4] Masoudnia S, Ebrahimpour R. Mixture of experts: a literature survey[J]. Artificial Intelligence Review, 2014, 42(2): 275-293.

[5] Shazeer N, Mirhoseini A, Maziarz K, et al. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer[J]. arXiv preprint arXiv:1701.06538, 2017.

[6] Shoeybi M, Patwary M, Puri R, et al. Megatron-lm: Training multi-billion parameter language models using model parallelism[J]. arXiv preprint arXiv:1909.08053, 2019.

[7] Rajbhandari S, Rasley J, Ruwase O, et al. Zero: Memory optimizations toward training trillion parameter models[C]//SC20: International Conference for High Performance Computing, Networking, Storage and Analysis. IEEE, 2020: 1-16.

Thank You!

Linghui Meng

Institute of Automation, Chinese Academy of Sciences, CASIA

menglinghui2019@ia.ac.cn