M6-10T: A SHARING-DELINKING PARADIGM FOR EFFICIENT MULTI-TRILLION PARAMETER PRETRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent expeditious developments in deep learning algorithms, distributed training, and even hardware design for large models have enabled training extreme-scale models, say GPT-3 and Switch Transformer possessing hundreds of billions or even trillions of parameters. However, under limited resources, extreme-scale model training that requires enormous amounts of computes and memory footprint suffers from frustratingly low efficiency in model convergence. In this paper, we propose a simple training strategy called "Pseudo-to-Real" for high-memory-footprint-required large models. Pseudo-to-Real is compatible with large models with architecture of sequential layers. We demonstrate a practice of pretraining unprecedented 10-trillion-parameter model, an order of magnitude larger than the state-of-the-art, on solely 512 GPUs within 10 days. Besides demonstrating the application of Pseudo-to-Real, we also provide a technique, Granular CPU offloading, to manage CPU memory for training large model and maintain high GPU utilities. Fast training of extreme-scale models on a decent amount of resources can bring much smaller carbon footprint and contribute to greener AI.

1 INTRODUCTION

Foundation models with self-supervised learning on big data have become an emerging paradigm of artificial intelligence systems [2], as they mostly possess high transferability to a wide range of downstream tasks and even multiple modalities. The scale of foundation models across domains, including natural language processing, computer vision, and cross-modality representation learning, have been growing tremendously from millions to trillions of parameters [7; 23; 24; 26; 33; 37; 3; 29; 14; 11; 16; 47; 46; 42; 44; 38] thanks to the concurrent advancement in distributed training framework [20; 27; 31; 28; 1; 12; 30] and hardware design, and these studies have made a demonstration of the neural scaling law [13]. However, the training of these transformer-based models incurs high financial costs and even environmental damage due to the massive carbon footprint and thus training extreme-scale models under a decent amount of resources but with high efficiency should be a fundamental goal for both the research and industrial communities to achieve, which promotes the progress of greener AI [22; 34].

Generally there are two tracks of research in large-scale pretraining, dense models and sparse expert models respectively. A typical case of large-scale dense models is GPT-3 [3], a 175-billion-parameter transformer model trained with $10,000$ GPUs for months, incurring striking financial and environmental costs. Researchers have been searching for methods to training large-scale models with a decent amount of costs. Solutions include effective management of memory with gradient and optimizer state partitioning [27] or more efficient model parallelism and pipeline parallelism [37; 20; 12]. A series of following studies apply those techniques to realize fast training of 10-billion-parameter transformers with hundreds of GPUs in $1-2$ months. [16; 46; 44; 38] Sparse expert models with large model capacity are capable of fast training owing to the combination of data parallelism and expert parallelism [35; 14; 11; 32], and it is even accessible to train a 1-trillion-parameter transformer with no more than 500 GPUs [42].

Be there as it may, a question emerges in our mind: is it possible to train an extreme-scale model with only a decent amount of resources, e.g., training a 10-trillion-parameter model with around 500 GPUs? Tackling the problem requires the utilization of external memory except for GPU memory, for instance, CPU memory or even NVMe storage [31; 28]. These methods resolve the problem of

high memory footprint, but instead, their extra cost is low training efficiency caused by the frequent swap in-and-out between memories.

In this paper, we provide a solution to training large models that require high memory footprint, and we demonstrate a successful practice of pretraining an unprecedented extreme-scale model with over 10 trillion parameters, an order of magnitude larger than the previous state-of-the-arts [11; 42]. The whole pretraining was conducted on solely 512 NVIDIA-V100 GPUs and lasted around 10 days. A simple and effective training strategy called "Pseudo-to-Real" enables the sharing and delinking of parameters. This training strategy is compatible with architectures built by stacking layers of an identical structure, including dense models like GPT [23; 24; 3], BERT [7], or sparse expert models like M6 [16; 42]. It is essentially a two-stage training. It first trains a relatively small model but with a computation graph of a large one with the utilization of cross-layer parameter sharing, and we name it "Pseudo Giant". Then it builds a correspondingly large model and delinks the parameters of the shared layer for second-stage model initialization. In this way, we achieve fast convergence in the first stage as the training costs much less memory and speeds up with large batches. Parameter sharing that addresses the communication overhead improves training speed as well. The second-stage training is responsible for the final convergence for better performance.

We unlock the secret of pretraining an unprecedented extreme-scale model with over 10 trillion parameters on limited resources of 512 GPUs. Compared with the previous M6-T on around 500 GPUs, we do not have a significant increase in computation resources but level up the model scale by an order of magnitude. Besides the application of the "Pseudo-to-Real" training strategy, we provide a faster offloading mechanism for both management of CPU memory for parameter storage and utility of GPUs. We successfully train the M6-10T within 10 days to reach strong performance in log perplexity evaluation and outperform the baseline M6-T.

Contributions at a glance are below:

- We illustrate the training difficulty of extreme-scale models on limited resources and provide a simple but effective solution called "Pseudo-to-Real". Upstream and downstream evaluation demonstrates the effectiveness of the strategy.
- We further demonstrate a successful practice of pretraining a 10-trillion-parameter model on 512 GPUs and reach an outstanding performance within 10 days.

2 Related Work

Large-Scale Pretraining In recent years, pretrained language models with growing magnitudes of parameters have been proposed, keeping to raise the validated upper limit of scaling law for model capacity w.r.t the number of parameters [13]. Earlier milestones of extreme-large models come from GPT-2 [24] and Megatron-LM [37], which demonstrates that scaling the transformer model up to billions of parameters can result in improvement on language modeling benchmarks [19; 21]. Turing-NLG [33], as a successor, implements a 17-billion-parameter transformer and achieves further lower perplexity. Similar phenomena are also observed on classification language tasks by T5 model [26]. The GPT-3 [3] pushes the boundary of model scale to 1,750 billion parameters and demonstrates its striking effectiveness on downstream tasks in even zero-shot settings. Furthermore, large-scale pretraining has recently demonstrated success in other fields, including pretraining on other languages or cross-lingual pretraining [41; 46; 44; 6], cross-modal pretraining [25; 29; 16; 8; 48] and code generation [4]. Along with the benefit from increasing the model scale, the concern of unaffordable pretraining cost in time, computation resource and energy keeps emerging [22; 2], resulting in the strong demand for more efficient and greener large-scale pretraining [34].

Methods to Train Larger Models Faster Our work is about designing algorithm to train extreme-scale models efficiently. Researchers have been demonstrating different types of methods to reach this objective. To reduce the computational cost during training, some studies introduce sparsity to the model. Mixture-of-Experts (MoE) [35] was proposed and was reintroduced in Mesh Tensorflow [36]. It shows that MoE with sparsity can significantly improve the model scale efficiently without increasing computation, and the models achieved state-of-the-art performance in language modeling and machine translation. GShard [14] extended it to a tremendously large scale of 600B parameters with the sophisticated collaborated design of model architecture and demonstrated its effectiveness

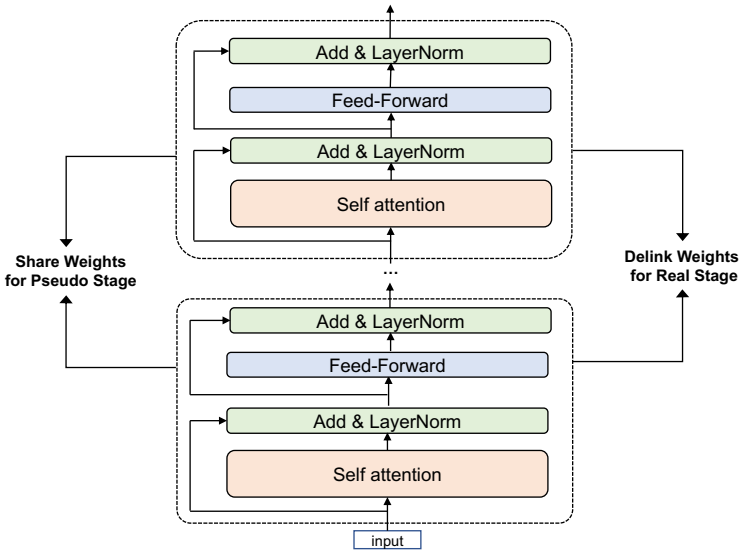


Figure 1: A demonstration of the Pseudo-to-Real training paradigm. It first shares parameters across layers at the Pseudo stage, and then delinks the parameters at the Real stage.

across over 100 languages. Similarly, Switch Transformer reached 1.6 trillion parameters and showed its strong performance on NLU tasks. Those models with high sparsity are computationally efficient, and therefore though they possess large model capacity they still can be trained with high efficiency. Most other studies still focus on training dense models to validate the scaling law, and thus the emerged problem is the distributed training of large models. The most influential distributed framework should be DeepSpeed that proposed ZERO [27] that partitions optimizer states and gradients to multiple GPU devices, and ZERO-offload [31] as well as ZERO-Infinity [28] can even offload parameters to CPU memory and NVMe storage. The memory footprint management makes training extremely large models on limited resources possible. However, such offloading mechanisms still have some defects that they may fail to fully utilize the fast hardware. For example, when offloading all parameters to the CPU, the GPU memory can be idle. In this work, we tackle this issue by proposing a granular offloading mechanism that can determine which parameters to be offloaded.

In addition to reducing the amount of computation in a single iteration, another route to speedup training is to reduce the needed iterations for model convergence. Child et al. [5] and Xiong et al. [40] propose to put forward the layernorm operations in transformer blocks for more stabilized and faster convergence. You et al. [43] employs a layer-wise adaptive optimizer to enable super-large pretraining batches. Zhang & He [45] progressively increases the layer dropping rate in a stochastic manner which significantly speedups pretraining.

3 Approach

3.1 Model Architecture

Choice in model architecture depends on several factors. First, the architecture should contain a sequence of stacking layers, as the sequential structure enables cross-layer parameter sharing. We prefer a simple encoder or decoder architecture, instead of an encoder-decoder framework where there are cross attentions that bring extra parameters and incur difficulties in activation checkpointing. Second, a model of such architecture should be compatible with different types of downstream tasks, including understanding and generation, and it is more preferable that it can be compatible with multiple modalities. Third, as we mention that dense models and sparse expert models are two main tracks of large-scale pretraining, we prefer the model that can flexibly become whether dense or sparse expert models. Therefore, we select M6 [16] as an option, and we evaluate the effects of our method on M6 of different scales and types.

M6 is built with stacking transformer layers, which includes self attention and feed-forward neural nets (FFN). For the transformation from dense models to sparse expert models, we should only replace FFN layers with the Mixture-of-Expert (MoE) layers. MoE consists of multiple experts, which are usually FFNs distributed on different devices. A gating network decides the dispatching and combining behaviors of each token and thus tokens can be processed in diverse devices. Such mechanism is a combination of data parallelism and expert parallelism, and thus it is highly efficient though with large model capacity. For the training, to realize the learning of both understanding and generation, the model is trained with text denoising and language modeling on plain text data and with image-based text denoising and image captioning on multimodal data. The model is compatible with different types of downstream tasks and can process information of multiple modalities.

3.2 PSEUDO-TO-REAL

This section demonstrates the details of "Pseudo-to-Real" two-stage training strategy that enables fast training of high-memory-footprint-required transformer models. The strategy consists of two stages. The first stage trains a model with many fewer parameters but with a large computation graph ("Pseudo Giant"), and the second stage trains a corresponding large model ("Real Giant") initialized with the delinked weights of the shared layer. Thus we name the strategy "Pseudo-to-Real", and the general idea is demonstrated in Figure 1.

3.2.1 "PSEUDO" STAGE: LAYER-WISE PARAMETER SHARING

The core of "Pseudo" stage is to train a Pseudo-Giant that shares parameters across layers. Cross-layer parameter sharing has proved successful in maintaining satisfactory performance while keeping a much smaller amount of parameters. We introduce it to training an extreme-scale model, and we hypothesize that the first-stage training can gain benefits from cross-parameter sharing as it can address communication overhead and it consumes much less memory footprint. Also, as Pseudo Giant with much fewer parameters is not bounded by memory, it can be trained with large batches for acceleration.

Suppose we build an M6 model with L layers that share parameters across all layers. The Pseudo Giant though consists of a computation graph of a L-layer transformer, its number of weight parameters and optimizer states should be $1/L$ of those of the original one. As to the gradients, we can accumulate the gradients of each layer in the backward computation process, and therefore the amount of gradients becomes $2/L$ of the original one. Such saving in memory enables much faster training with larger batches. Also, it is capable to use fewer resources even due to lower memory consumption.

This can also be applied to sparse expert models, as their architecture is stacking transformer layers. However, different from dense models, the MoE partition their weights to all devices, and each token is distributed to a selected expert by a routing function. Thus the models can trained with data parallelism across all devices and expert parallelism where parameters at each device are activated. This mechanism though enlarges model capacity significantly without much efficiency loss, yet it limits the flexible usage of GPU resources in different stages, say training and inference. Due to the application of data parallelism, we can implement dense models on different numbers of GPU devices at the training and inference stage. However, as mentioned above, the combination of data parallelism and expert parallelism requires the identical amount of devices at different stages.

To tackle this problem, we specifically design methods for expert merging and partitioning. Due to cross-layer parameter sharing, at the Pseudo stage it is capable of implementing more experts on each device, and it is available to use fewer GPUs while keeping the total number of experts. We name it expert merging for simplicity. At the Real stage, the delinking of parameters with the same amount o devices will cause out-of-memory errors. We delink the parameters and partition experts to more devices, e.g., from 256 to 512, and thus there are fewer experts on each device where memory is sufficient. Thus we name it expert partitioning. Therefore, now it is available to use different number of GPU devices at different stages for training sparse expert models.

3.2.2 "REAL" STAGE: DELINKING THE SHARED PARAMETERS

We name a large model without cross-layer parameter sharing "Real Giant", in comparison with Pseudo Giant. Both models share an identical computation graph, but possess different numbers

Table 1: Experimental results on downstream evaluation of natural language understanding. We evaluate the performance of models on 8 tasks of GLUE dev set except for WNLI following Devlin et al. [7].

Model	#Params	SST-2	CoLA	MNLI	QNLI	QQP	MRPC	RTE	STS-B	Avg
BERT	345M	93.7	60.6	86.6	92.3	91.3	88.0	70.4	90.0	84.1
M6	350M	94.2	61.3	86.3	92.3	91.4	91.4	83.4	89.0	86.2
M6 (P)	65M	90.5	46.2	80.8	89.4	89.9	90.1	70.7	85.6	80.4
M6 (P2R)	65M/350M	94.4	58.6	86.6	93.3	91.5	91.5	84.5	89.1	86.2

of parameters. Given a Pseudo Giant fully trained until convergence, we apply the delinking of cross-layer shared parameters to accelerate Real Giant training. There is no need to train a large model from scratch. The model can start its convergence from low perplexity.

Embedding initialization can be directly restored, but the layer weights should be treated specially. In practice, there is only one layer of weights θ_{shared} in Pseudo Giant, and there are L layers of weights $\{\theta_1, \theta_2, \cdots, \theta_L\}$ in Real Giant. Thanks to their identical structure, each layer of Real Giant can be initialized with θ_{shared}. Without further training, this model is equivalent to a fully-trained Pseudo Giant.

This extremely simple training strategy is highly beneficial for the high-memory-footprint-required large models, especially extreme-scale models like the 10-trillion-parameter M6. While the first stage of training saves much time for faster convergence, we can use a decent amount of computational resources in this stage as lower efficiency in this stage becomes acceptable. Therefore, in the practice of training an extreme-scale M6, we apply CPU offloading to utilize CPU memory. Therefore, we can use a limited amount of resources, e.g., 512 GPUs, to train an unprecedented 10-trillion-parameter model efficiently, which is an order of magnitude larger than the state-of-the-arts.

3.2.3 Timing for Switching

A question naturally emerges: when should we switch from the Pseudo stage to the Real stage? As mentioned above, the greatest advantage of Pseudo stage for training is the significantly faster convergence speed. Yet the performance of Pseudo Giant is bounded by its limitation in the number of parameters. Training Pseudo Giant until convergence apparently incurs much waste of time.

In practice, we present a simple strategy to determine the training step to switch from Pseudo to Real based on the convergence speed. During the training of the Pseudo stage, we evaluate a training step in a fixed interval by attempting to transfer it into the Real stage and training for a small while (e.g., 30 minutes). After that, we will revert the model parameters to the evaluated step and continue the training of the Pseudo stage for the same training time as the Real stage. If the decreasing speed of loss in the Real stage surpasses that of the Pseudo stage, we determine the evaluated training step as the best switching point for the next-stage training.

3.3 Experiments

In this section, we demonstrate experiments to evaluate the effectiveness of the training strategy by evaluating the model quality and observing the performance improvement.

3.3.1 Performance Evaluation

We aim to validate two hypotheses: 1. Pseudo-to-Real training paradigm can help the model reach competitive performance with the one trained from scratch without parameter sharing; 2. Pseudo-to-Real can effectively accelerate convergence in training of large-scale models.

We aim at discovering the model quality brought by different pretraining strategies. "Pseudo (P)" refers to training with parameter sharing across layers, and "Pseudo-to-Real (P2R)" refers to the proposed two-stage training of a sharing-delinking paradigm. We pretrain the model on BookCorpus [49] and English Wikipedia [7], which are corpora with around 16GB of plain texts. Following Radford et al. [23] and Lewis et al. [15], we use a vocabulary of around $50,000$ subwords. Each

Table 2: Experimental results on downstream task evaluation. "#Params" refers to the number of parameters. We report the PPL evaluation on WikiText-103 and the ROUGE1, ROUGE-2, and ROUGE-L evaluation on Gigaword.

Model	#Params	WikiText-103	Gigaword
Megatron-LM	350M	16.69	-
UniLM	340M	-	38.5/19.5/35.4
M6	350M	16.59	38.8/20.1/36.0
M6 (P)	65M	28.60	36.9/18.1/34.3
M6 (P2R)	65M/350M	16.60	38.3/19.3/35.7

Table 3: Model refers to the types of model. d_{model} and d_{ff} refer to the hidden size and intermediate size. L refers to the number of layers in the computation graph, and l refers to the number of transformer layers with parameters. #Heads refers to the number of heads in self attention. #Params refers to the total number of model parameters. We also report their training speed on 48 GPU devices by the number of consumed samples per second.

Model	d_{model}	d_{ff}	l/L	#Heads	#Params	Speed
Base	1024	4096	24/24	16	350M	650
Pseudo	1024	16384	1/36	16	1.4B	248
Real	1024	16384	36/36	16	1.4B	48

sample consists of sentences from an identical passage, and we use a sequence length of 512 and correspondingly truncate or pad the sequence. Following the common practice in pretraining [7; 17], we apply AdamW optimizer [18] for optimization. To determine the most suitable learning rate of the two stages for fast convergence, we have made some preliminary tests and finally used the peak learning rate of 2e-4 for "Pseudo" stage and 8e-5 for "Real" stage, respectively. We use a cosine decaying mechanism for learning rate scheduling and a warm-up ratio of 0.001. We pretrain the models until convergence and transfer them to downstream tasks.

For comprehensive analysis, the experiments include natural language understanding and generation tasks. Notably, for NLU tasks, we follow Devlin et al. [7] and validate transfer effects on GLUE [39]. For natural language generation, we specify zero-shot language modeling and text summarization to evaluate both models' upstream and downstream quality on generation. Specifically, we conduct experiments on WikiText-103 and Gigaword for both tasks.

Experimental results on NLU tasks are demonstrated in Table 1. For better comparison, we also present the experimental results of BERT with a similar amount of parameters, and it shows that M6 can achieve a better performance over the baseline. From Table 1, we find that Pseudo-only training brings a significant performance downgrade in the 8 tasks. It is fair to believe that the limited amount of parameters hinders downstream performance, though with a large computation graph. However, in comparison, we find that P2R can help the model perform similarly to the one trained from scratch without parameter sharing. Specifically, M6 with P2R can even outperform the baseline on 6 tasks, including SST-2, MNLI, QQP, MRPC, RTE, and STS-B. We further validate their performance on zero-shot language modeling and text sumarization tasks. Table 2 demonstrate the model performance on both tasks, and we additionally add Megatron-LM [37] and [10] for better comparison. Similarly, Pseudo-only training leads to far worse performance, and P2R is also able to help the model achieve similar performance with Real training.

3.3.2 Efficiency Evaluation

Next, we evaluate the efficiency of training strategies by observing model performance under the condition of identical computation budget.

Experimental Setup To satisfy the requirements of high memory footprint where the two-staged training can make a difference in training efficiency, we conduct experiments on a 1.4B-parameter model. We present the details of model configuration in Table 3.

Table 4: Experimental results of large models trained with limited budget on downstream task evaluation. We report the PPL on WikiText-103 and the ROUGE scores on Gigaword.

Model	#Params	WikiText-103	Gigaword
	limited budget		
M6-1B (P)	90M	56.79	36.8/17.9/34.1
M6-1B (R)	1.4B	26.80	36.9/18.2/34.2
M6-1B (P2R)	90M/1.4B	23.60	37.3/18.3/34.5

For both Pseudo and Real Giant, we train them with a total batch size of $6,144$. In practice, we use a micro-batch size of 32 and a gradient accumulation step of 4, and we train our models on 48 NVIDIA-V100 GPU devices. We compare the model performance with a limited budget, where models have been trained for the same duration of time. Correspondingly, "Pseudo" has been trained for around $25,000$ steps, Real has been trained for around $4,700$ steps, and "P2R" has been trained for $12,000$ steps.

Model Performance Table 3 demonstrates the training speed of the models. We report their training speed on 48 NVIDIA-V100 GPU devices with their consumed samples per second. Training Real Giant model from scratch is highly time-consuming. Pseudo Giant training has an advantage of around 5 times of convergence speed over Real Giant training. We also observe the loss convergence of both P2R and Real Giant trained from scratch. In Figure 2 we present the development of pretraining language modeling loss, which is the log perplexity, on the time basis. The log perplexity of P2R decreases much faster than that of the Real Giant, with an advantage larger than 0.3.

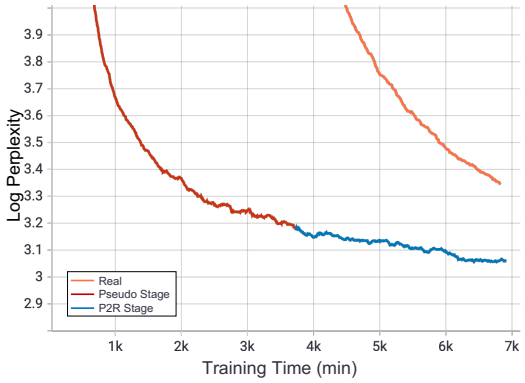


Figure 2: Comparison of pretraining language modeling loss of M6-1B P2R and Real on time-basis.

Experimental results on downstream tasks presented in Table 4 are consistent with our hypothesis that Pseudo-to-Real training can speed up training effectively. In the setup of limited budget, the M6 model trained with Pseudo-to-Real can outperform the Pseudo Giant and Real Giant in both language modeling and text generation.

4 Towards a 10-Trillion-Parameter Model

Previously, training large-scale models brings tons of challenges to the collaboration algorithm design, distributed training, as well as hardware design, etc. Training a GPT-3 of 175B parameters with a combination of data parallelism on over 500GB of data should cost around 300 GPU-years. Later with the emergence of partitioning on optimizer states, gradients, and even weights, GPU memory can be fully utilized without performance degradation. Now we can even use the CPU memory or even NVMe storage to store the parameters, but we have to bear the costs of efficiency. Therefore, we attempt to tackle the difficulty of extreme-scale model training from the perspective of algorithm design and thus we apply the aforementioned Pseudo-to-Real training strategy to train an extreme-scale model.

4.1 Model Setup

We design a 10-trillion-parameter M6 model with the combination of existing methods and proposed strategies to demonstrate a case of how to train an extreme-scale model efficiently. In comparison with the previous studies of trillion-parameter models [11; 42], this one is almost 10 times larger. To efficiently utilize the memory, we adopt Mixture-of-Experts and we replace every FFN layer with the memory-efficient MoE layers. Notably, we remove the auxiliary loss that consumes memory and

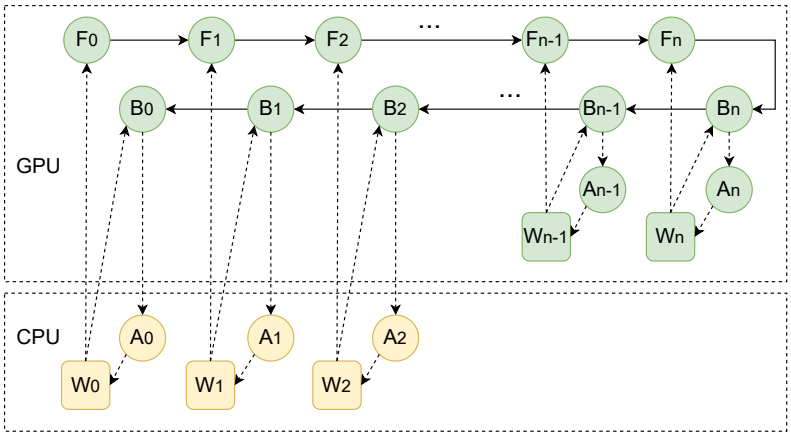


Figure 3: A demonstration of Granular CPU Offloading mechanism.

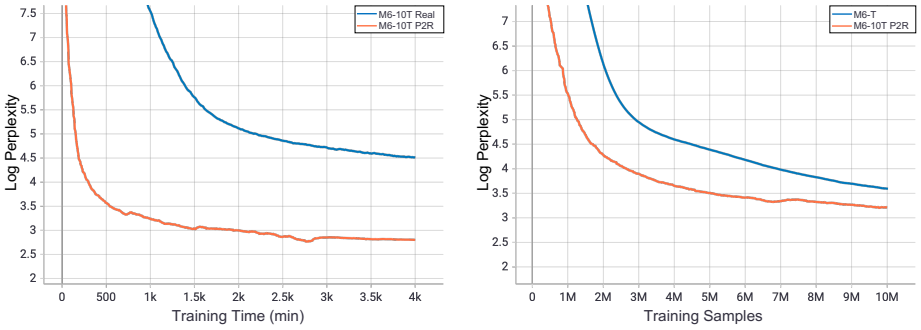
demonstrates little effects on model quality, and we follow Yang et al. [42] to apply expert prototyping for improved model quality and training stability. To be more specific, the hidden size d_{model} is 1024 and the intermediate size d_{ff} is 9984. The number of model layers is 48. For each MoE layer, there are 10240 experts distributed on multiple devices. We use 80 prototypes of experts based on our experience in preliminary experiments. The training batch size per GPU device is 8. We implement models on EFLOPs, a distributed training platform with an advanced server architecture and a new network topology [9]. Specifically our models are trained on a cluster of 8-GPU workers connected by RDMA networks with a bandwidth of 100Gb. The CPU memory of each worker is around 750GB. The expert distribution and Granular Offloading strategies are supported by Whale framework[12].

4.2 Granular CPU offloading

To utilize CPU memory for larger models with fewer resources, we apply the Granular CPU offloading which has a higher efficiency compared with the conventional CPU offloading. Previously we note that conventional offloading mechanisms offload all parameters, which may fail to effectively utilize GPU. To improve the efficiency of CPU offloading, we propose a new CPU offloading mechanism called Granular CPU offloading. The training process is composed of phases including "Forward (Fn)", "Backward(Bn)" and "Apply (An)". Offloading all model parameters to CPU in Fn and Bn requires loading parameters from CPU to GPU memory twice. Activation checkpointing that brings recomputation needs the parameters loaded in Bn. An requires the gradients offloaded from GPU memory to CPU memory. Assume the model parameter size is $\mathbf{W}$, the above processes bring parameter movement of size $\mathbf{4} * \mathbf{W}$.

In offloading, PCIE is the bottleneck of the whole training process. We observe that when offloading all parameters with recomputation, the GPU memory is idle. We can fill up the GPU memory by selective offloading, leaving part of the model in GPU memory. In this way, the model can be accelerated by reducing across-device memory copy. In our preliminary experiment, with the setting of training a 48-layer 78B-parameter M6 model on 8 NVIDIA-V100 GPU devices, the step-time costs 89 seconds when fully offloading the parameters into CPU memory. In comparison, granularly offloading the first 24 layers into CPU memory and leaving the remaining 24 layers on GPU reduces the step-time to only 45 seconds. The significant difference in training step-time indicates that the time-cost of parameter movement between CPU and GPU will dominate the training step-time when offloading is employed, thus the granularity of offloading and the utilization of GPU should be seriously considered in extreme-scale pretraining. In addition, offloading the whole model can result in OOM error in the CPU when the model is extremely large.

With Granular CPU offloading, we successfully implement a 10-trillion-parameter M6 model on solely 512 NVIDIA-V100 GPUs. Furthermore, at the Pseudo stage, we can train a Pseudo Giant with a computation graph of 10 trillion parameters only with 256 GPU devices without the utilization of CPU memory for offloading. Thus in our practice, we train a Pseudo Giant with only 256 GPUs, and



(a) M6-10T P2R vs. Real on time-basis (b) M6-10T P2R vs. M6-T on sample-basis

Figure 4: The log perplexity of M6-10T Pesudo-to-Real (P2R) compared with baselines on time-basis and sample-basis, respectively. (a) Compared with M6-10T Real on time-basis, M6-10T P2R converges much faster due to the significant reduction of time-cost on offloading in the Pseudo stage. (b) Compared with 1T-parameter M6-T model on sample-basis, though undergoing the Pseudo stage which limits the model capacity, M6-10T still has an advantage in training sample efficiency.

then partition the experts and redistribute them to 512 GPUs. This saves the usage of GPU resources, which is more resource-efficient and also more environmentally friendly.

4.3 Analysis

Training Efficiency We pretrain the M6-10T with Pseudo-to-Real training strategy for around 10 days, and we additionally train an M6-10T from scratch without the strategy for around 3 days for comparison. The Real stage training can be facilitated without Out-of-Memory errors with the help of Granular CPU offloading, but its step-time is only around 180s. In contrast, the step-time of Pseudo stage is only 14s without the cost of offloading, which greatly boosts the training efficiency of M6-10T P2R. The M6-10T with P2R has been trained for $15k$ steps, but the one from scratch has been trained for solely $1.3k$ steps. We record the log perplexity of both models trained on the M6-Corpus on the time basis in Figure 4(a). Results show that within the same length of time M6-10T with P2R can outperform the one trained from scratch by a large margin.

Convergence Analysis We have trained M6-10T with Pseudo-to-Real strategy for around 10 days, and the model converges to a low level of log perplexity based on the upstream evaluation on the M6-Corpus. For better comparison, we also show the convergence performance of the 1T-parameter M6-T model proposed in the previous work. As shown in Figure 4(b), the observation is consistent with our intuition that the model with a larger capacity can converge faster on the sample basis, and it should achieve the best performance on language modeling. What leaves open is whether it can positively lead to better downstream performance concerning different types of downstream tasks. Finetuning extreme-scale models should be difficult and there is still much room for us to discover the potential of extreme-scale models.

5 Conclusion and Future Work

Pseudo-to-Real training strategy is a simple and effective way to train large-scale models that are highly memory consuming, and it is also essential to training extremely large models with limited resources with significantly higher training efficiency. We unlock pretraining unprecedented extreme-scale models with 10 trillion parameters with limited resources of 512 GPUs in 10 days. Besides the application of Pseudo-to-Real training strategy, we further provide Granular CPU offloading to enhance GPU utility while breaking the GPU memory wall with a cost in efficiency. The advances take a leap towards extreme-scale model training beyond implementation on limited resources. With only a few GPU cards, training large models with tens or hundreds of parameters has become accessible to many researchers. We believe that this can motivate low carbon dioxide production and encourages the progress of green AI.

ETHICS STATEMENT

This work is highly concerned with large-scale language models and multimodal pretrained models. These models have been pretrained on broad data of plain texts and image-text pairs, which might contain harmful information, such as hate speech, terrorism, pornography, etc. We have put much efforts to remove these kinds of data in our datasets by quality evaluation on texts and images. However, this problem cannot be eliminated and ignored, and it is common in the pretraining community. For those models that are not trained on commonly-used public datasets, we will carefully release the model checkpoints before careful evaluation, and also limit the access to avoid misconduct.

REPRODUCIBILITY STATEMENT

This work is generally reproducible. Following the description in Section 3, researchers can easily implement the training strategy on the codebases for pretraining, including Huggingface Transformer [1], Fairseq [2], etc.

REFERENCES

[1] Mandeep Baines, Shruti Bhosale, Vittorio Caggiano, Naman Goyal, Siddharth Goyal, Myle Ott, Benjamin Lefaudeux, Vitaliy Liptchinsky, Mike Rabbat, Sam Sheiffer, Anjali Sridhar, and Min Xu. Fairscale: A general purpose modular pytorch library for high performance and large scale training. https://github.com/facebookresearch/fairscale, 2021.

[2] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. arXiv preprint arXiv:2108.07258, 2021.

[3] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual, 2020.

[4] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde, Jared Kaplan, Harri Edwards, Yura Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. arXiv preprint arXiv:2107.03374, 2021.

[5] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. arXiv preprint arXiv:1904.10509, 2019.

[6] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised cross-lingual representation learning at scale. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, pp. 8440–8451, 2020.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pp. 4171–4186, Minneapolis, Minnesota, 2019. Association for Computational Linguistics.

[1] https://github.com/huggingface/transformers

[2] https://github.com/pytorch/fairseq

[8] Ming Ding, Zhuoyi Yang, Wenyi Hong, Wendi Zheng, Chang Zhou, Da Yin, Junyang Lin, Xu Zou, Zhou Shao, Hongxia Yang, and Jie Tang. Cogview: Mastering text-to-image generation via transformers. CoRR, abs/2105.13290, 2021.

[9] Jianbo Dong, Zheng Cao, Tao Zhang, Jianxi Ye, Shaochuang Wang, Fei Feng, Li Zhao, Xiaoyong Liu, Liuyihan Song, Liwei Peng, Yiqun Guo, Xiaowei Jiang, Lingbo Tang, Yin Du, Yingya Zhang, Pan Pan, and Yuan Xie. EFLOPS: algorithm and system co-design for a high performance distributed training platform. In IEEE International Symposium on High Performance Computer Architecture, HPCA 2020, pp. 610–622, 2020.

[10] Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. Unified language model pre-training for natural language understanding and generation. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada, pp. 13042–13054, 2019.

[11] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. arXiv preprint arXiv:2101.03961, 2021.

[12] Xianyan Jia, Le Jiang, Ang Wang, Jie Zhang, Xinyuan Li, Wencong Xiao, Yong Li, Zhen Zheng, Xiaoyong Liu, and Wei Lin. Whale: Scaling deep learning model training to the trillions. arXiv preprint arXiv:2011.09208, 2020.

[13] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. arXiv preprint arXiv:2001.08361, 2020.

[14] Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. Gshard: Scaling giant models with conditional computation and automatic sharding. arXiv preprint arXiv:2006.16668, 2020.

[15] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pp. 7871–7880, Online, 2020. Association for Computational Linguistics.

[16] Junyang Lin, Rui Men, An Yang, Chang Zhou, Ming Ding, Yichang Zhang, Peng Wang, Ang Wang, Le Jiang, Xianyan Jia, et al. M6: A chinese multimodal pretrainer. arXiv preprint arXiv:2103.00823, 2021.

[17] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. arXiv preprint arXiv:1907.11692, 2019.

[18] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019. OpenReview.net, 2019.

[19] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings. OpenReview.net, 2017.

[20] Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Anand Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, et al. Efficient large-scale language model training on gpu clusters. arXiv preprint arXiv:2104.04473, 2021.

[21] Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The LAMBADA dataset: Word prediction requiring a broad discourse context. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 1525–1534, Berlin, Germany, August 2016. Association for Computational Linguistics.

[22] David A. Patterson, Joseph Gonzalez, Quoc V. Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David R. So, Maud Texier, and Jeff Dean. Carbon emissions and large neural network training. arXiv preprint arXiv:2104.10350, 2021.

[23] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. URL https://s3-us-west-2.amazonaws.com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf, 2018.

[24] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners, 2019.

[25] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In Proceedings of the 38th International Conference on Machine Learning, ICML 2021, pp. 8748–8763, 2021.

[26] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. Journal of Machine Learning Research, 21:1–67, 2020.

[27] Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: Memory optimizations toward training trillion parameter models. In SC20: International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 1–16. IEEE, 2020.

[28] Samyam Rajbhandari, Olatunji Ruwase, Jeff Rasley, Shaden Smith, and Yuxiong He. Zero-infinity: Breaking the gpu memory wall for extreme scale deep learning. arXiv preprint arXiv:2104.07857, 2021.

[29] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. arXiv preprint arXiv:2102.12092, 2021.

[30] Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 3505–3506, 2020.

[31] Jie Ren, Samyam Rajbhandari, Reza Yazdani Aminabadi, Olatunji Ruwase, Shuangyan Yang, Minjia Zhang, Dong Li, and Yuxiong He. Zero-offload: Democratizing billion-scale model training. arXiv preprint arXiv:2101.06840, 2021.

[32] Stephen Roller, Sainbayar Sukhbaatar, Arthur Szlam, and Jason Weston. Hash layers for large sparse models. arXiv preprint arXiv:2106.04426, 2021.

[33] Corby Rosset. Turing-nlg: A 17-billion parameter language model by microsoft. Microsoft Research Blog, 2020.

[34] Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. Green AI. Commun. ACM, 63 (12):54–63, 2020.

[35] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey E. Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In 5th International Conference on Learning Representations, ICLR 2017, 2017.

[36] Noam Shazeer, Youlong Cheng, Niki Parmar, Dustin Tran, Ashish Vaswani, Penporn Koanantakool, Peter Hawkins, HyoukJoong Lee, Mingsheng Hong, Cliff Young, Ryan Sepassi, and Blake A. Hechtman. Mesh-tensorflow: Deep learning for supercomputers. In Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, pp. 10435–10444, 2018.

[37] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism. arXiv preprint arXiv:1909.08053, 2019.

[38] Yu Sun, Shuohuan Wang, Shikun Feng, Siyu Ding, Chao Pang, Junyuan Shang, Jiaxiang Liu, Xuyi Chen, Yanbin Zhao, Yuxiang Lu, et al. Ernie 3.0: Large-scale knowledge enhanced pre-training for language understanding and generation. arXiv preprint arXiv:2107.02137, 2021.

[39] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019. OpenReview.net, 2019.

[40] Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tie-Yan Liu. On layer normalization in the transformer architecture. In Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event, volume 119 of Proceedings of Machine Learning Research, pp. 10524–10533. PMLR, 2020.

[41] Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. mT5: A massively multilingual pre-trained text-to-text transformer. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 483–498, Online, June 2021. Association for Computational Linguistics.

[42] An Yang, Junyang Lin, Rui Men, Chang Zhou, Le Jiang, Xianyan Jia, Ang Wang, Jie Zhang, Jiamang Wang, Yong Li, et al. M6-t: Exploring sparse expert models and beyond. arXiv preprint arXiv:2105.15082, 2021.

[43] Yang You, Jing Li, Sashank J. Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training BERT in 76 minutes. In 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020. OpenReview.net, 2020.

[44] Wei Zeng, Xiaozhe Ren, Teng Su, Hui Wang, Yi Liao, Zhiwei Wang, Xin Jiang, ZhenZhang Yang, Kaisheng Wang, Xiaoda Zhang, et al. Pangu-α: Large-scale autoregressive pretrained chinese language models with auto-parallel computation. arXiv preprint arXiv:2104.12369, 2021.

[45] Minjia Zhang and Yuxiong He. Accelerating training of transformer-based language models with progressive layer dropping. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual, 2020.

[46] Zhengyan Zhang, Yuxian Gu, Xu Han, Shengqi Chen, Chaojun Xiao, Zhenbo Sun, Yuan Yao, Fanchao Qi, Jian Guan, Pei Ke, et al. Cpm-2: Large-scale cost-effective pre-trained language models. arXiv preprint arXiv:2106.10715, 2021.

[47] Zhengyan Zhang, Xu Han, Hao Zhou, Pei Ke, Yuxian Gu, Deming Ye, Yujia Qin, Yusheng Su, Haozhe Ji, Jian Guan, et al. Cpm: A large-scale generative chinese pre-trained language model. AI Open, 2:93–99, 2021.

[48] Zhu Zhang, Jianxin Ma, Chang Zhou, Rui Men, Zhikang Li, Ming Ding, Jie Tang, Jingren Zhou, and Hongxia Yang. M6-ufc: Unifying multi-modal controls for conditional image synthesis. arXiv preprint arXiv:2105.14211, 2021.

[49] Yukun Zhu, Ryan Kiros, Richard S. Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In 2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015, pp. 19–27. IEEE Computer Society, 2015.