A Artifact Appendix

A.1 Abstract

This artifact includes the Radius for accelerating the pre-training of the Foundation Models. The example scripts for pre-training GPT-355M and GPT-2.0B model with OpenWebText dataset using dense *allreduce* and Radius are provided. Compared to the dense *allreduce* method, Radius with $d = 0.4$ and $T = 200$ achieved 20% speedup on pre-training GPT-2.0B model using 64 NVIDIA A100-80GB GPUs of Perlmutter supercomputer.

A.2 Artifact check-list (meta-information)

- **Algorithm:** Radius
- **Program:** Python
- **Model:** GPT-355M and GPT-2.0B
- **Data set:** OpenWebText
- **Run-time environment:**
- **Hardware:** A100-80GB GPUs
- **How much time is needed to complete experiments (approximately)?:** Depending on the status of Perlmutter, the whole experiment should take around two months, including the pre-training of GPT-355M and GPT-2.0B models and the corresponding evaluation on downstream tasks.
- **Publicly available?:** Yes
- **Workflow automation framework used?:** SLURM

A.3 Description

A.3.1 How to access. The code and instructions can be accessed through our github repository: link.

A.3.2 Hardware dependencies. Our experiment was carried out using 64 A100-80GB GPUs on Perlmutter supercomputer, where each compute node has 4 A100-80GB GPUs connected by NVLink3, providing an intra-node communication bandwidth of 600 GB/s, and the compute nodes are connected through HPE Slingshot 11, providing an aggregated internode communication bandwidth of 100 GB/s.

Other supercomputers or clusters with A100-80GB GPUs and the same intra- and inter-node communication bandwidth should observe similar speedups as reported in the paper.

A.3.3 Software dependencies. Some important Python packages: Apex, amp_C, and PyTorch.

A.3.4 Data sets. OpenWebText

A.3.5 Models. GPT-355M and GPT-2.0B from scratch. Please find the detailed model architecture in Table 1.

Table 1. Model architecture

Model	n_{layers}	n_{heads}	d_{model}	d_{head}
GPT-355M	24	16	1024	64
GPT-2.0B	24	32	2560	80

A.4 Installation

git clone https://github.com/mz687/radius

A.5 Experiment workflow

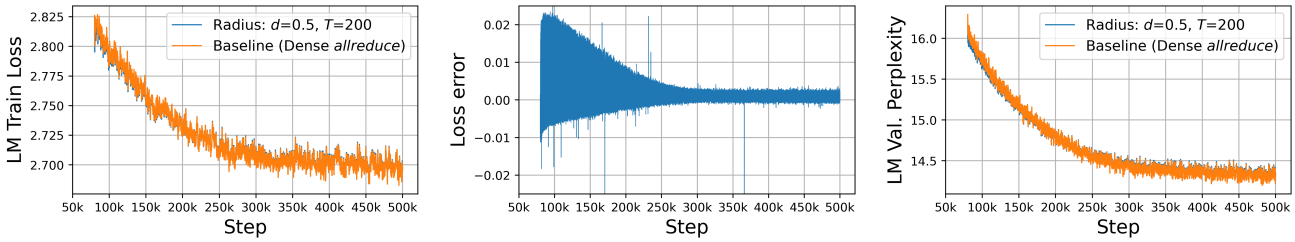
Please check our GitHub repository for the complete workflow and instructions.

A.6 Evaluation and expected results

Table 2, Figure 1, Figure 2 should be obtained after the complete workflow.

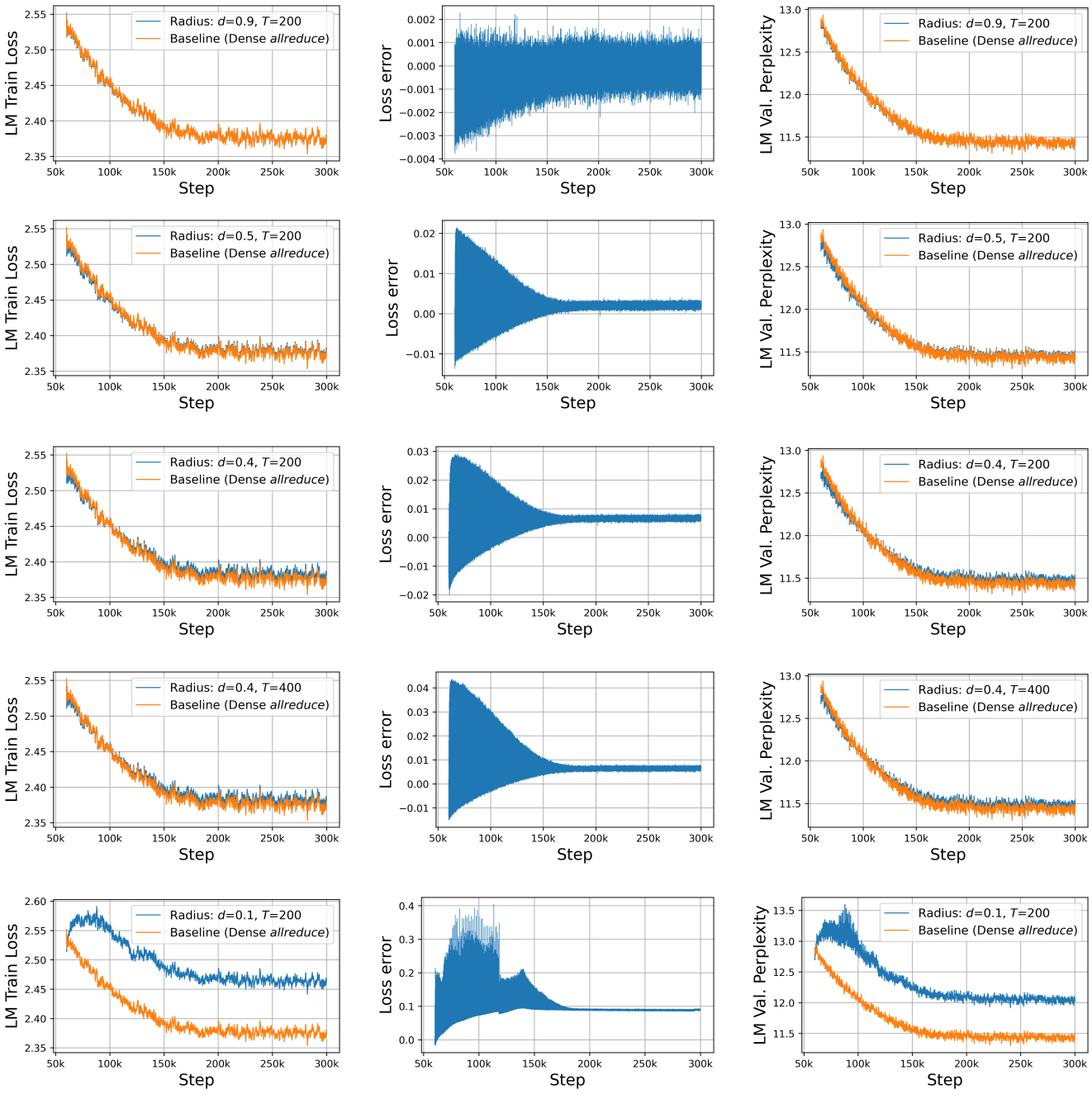
Table 2. Train, validation perplexity, wall-clock time, and overall training speedup.

		Baseline	Radius				
			d = 0.9, T = 200	d = 0.5, T = 200	d = 0.4, T = 200	d = 0.4, T = 400	d = 0.1, T = 200
GPT-355M	Train PPL	**14.64**	–	14.65	–	–	–
	Val. PPL	14.32	–	**14.31**	–	–	–
GPT-2.0B	Train PPL	**10.86**	10.87	10.89	10.94	10.94	11.85
	Val. PPL	11.42	**11.37**	11.39	11.41	11.46	11.98
	Wall-clock time (days)	6.44	6.82	5.73	5.43	5.41	4.78
	Speedup	–	-5.55%	12.40%	18.55%	19.03%	34.76%



(a) Train loss curve **(b)** Train loss error (baseline − Radius) **(c)** Validation PPL curve

Figure 1. Curves of pre-training GPT-355M with baseline (dense *allreduce*) and Radius



(a) Train loss curves **(b)** Train loss error curves **(c)** Validation PPL curves

Figure 2. Curves of pre-training GPT-2.0B with baseline (dense *allreduce*) and Radius