SPDF: Sparse Pre-training and Dense Fine-tuning for Large Language Models (Supplementary Material)

Vithursan Thangarasa[1] **Abhay Gupta**[1] **William Marshall**[1] **Tianda Li**[*] **Kevin Leong**[1]
Dennis DeCoste[*] **Sean Lie**[1] **Shreyas Saxena**[1]

[1]Cerebras Systems Inc., Sunnyvale, California, USA

APPENDIX

A EXPERIMENTAL SETUP AND HYPERPARAMETER DETAILS

A.1 PRE-TRAINING ON PILE

To train all GPT models, we use AdamW optimizer [Loshchilov and Hutter, 2017] with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. The global norm is clipped at 1.0, and a weight decay of 0.1 is used. There is a learning rate warm-up over the first 375M tokens, followed by a cosine decay to 10% of the peak learning rate. In Table 1, we provide details on the size and architecture configurations of the models we pre-trained. Here, n_{params} is the total number of trainable parameters, n_{layers} is the number of decoder layers, and d_{model} is the base size of the model. The feedforward bottleneck is four times the base size, i.e., $d_{\text{ff}} = 4 \times d_{model}$. Finally, n_{heads} are the number of attention heads and d_{head} is the dimension of each attention head. The context window size is set to 2048 following [Brown et al., 2020].

Table 1: Sizes, architectures, and learning hyperparameters (batch size and learning rate) of the models, which are trained to Chinchilla optimal configurations ($\approx$ 20 tokens per parameter).

Model	n_{params}	n_{layers}	d_{model}	n_{heads}	d_{head}	Batch Size	Learning Rate	Training Tokens
GPT-2 Small	125M	12	768	12	64	256	6e-4	2.5B
GPT-3 XL	1.3B	24	2048	16	128	512	2e-4	26B

Sparsity Setup As mentioned in Section 2, we use random pruning (static sparsity) for training all our GPT models. The sparsity is distributed uniformly across all layers (i.e., all layers get the same specified sparsity level) irrespective of the number of parameters or FLOPs of a given layer. We only sparsify all dense linear layers including the two in the MLP module; W_I (intermediate) and W_O (MLP output projection), and the four weight matrices in the self-attention module; W_Q (query), W_K (key), W_V (value) and W_D (attention output projection). In this setting, we keep the embeddings (implemented as sparse lookups), LayerNorm [Ba et al., 2016], and biases dense.

A.2 NATURAL LANGUAGE GENERATION

Similar to Hu et al. [2022], we train all of our GPT-2 Small and GPT-3 XL models using AdamW [Loshchilov and Hutter, 2017] with a linear learning rate schedule for 5 epochs, and perform early-stopping when the models began to overfit. We perform a grid search to discover an appropriate learning rate that led to the best downstream BLEU score on each of the tasks for a given compute budget. More specifically, on the dense baseline and sparse variants, we select the best batch size among {8, 16, 32, 64} and select the best learning rate among {1e-4, 5e-5, 2.5e-5} on the validation set. The beam search beam size, length penalty, and no repeat ngram size remain the same as described in [Hu et al., 2022].

Accepted for the 39th Conference on Uncertainty in Artificial Intelligence (UAI 2023).

Table 2: Details on the total pre-training FLOPs for each pre-trained model.. We also report the relative FLOPs reduction of the sparse variants over the dense baseline. We note that the reported FLOPs per sequence (seq) includes both forward and backward passes.

Model	Pre-Train Sparsity	Total Seqs	Total FLOPs/ Seq	Total FLOPs	Toal exaFLOPs	FLOPs Reduction over Dense
GPT-2 Small	0%	1.22e6	1.99e12	2.43e18	2.43	1x
	50%	1.22e6	1.47e12	1.79e18	1.79	0.737x
	75%	1.22e6	1.20e12	1.46e18	1.46	0.601x
GPT-3 XL	0%	1.27e7	1.86e13	2.36e20	236.10	1x
	50%	1.27e7	1.12e13	1.42e20	141.87	0.601x
	75%	1.27e7	7.46e12	9.48e19	94.76	0.401x

A.3 CURATION CORPUS

We fine-tune the dense GPT-2 Small and its sparse variants on the Curation Corpus [Curation, 2020] following the setup presented in [Rae et al., 2021]. We fine-tune for 5 epochs on Curation Corpus, and perform early stopping once the models start to overfit. To discover good hyperparameters, we perform a grid search to discover an appropriate learning rate that led to the best perplexity for a given compute budget. More specifically, on the dense baseline and sparse variants, we use a batch size of 32 and select the best learning rate among {3e-3, 1e-3, 3e-4, 1e-4, 3e-5, 1e-5} on the validation set.

A.4 TRAINING FLOPS FOR PRE-TRAINING AND FINE-TUNING

We compute the total pre-training FLOPs for the dense and sparse variants of GPT-2 Small and GPT-3 XL, and report them in Table 2, along with their relative FLOPs reduction over the dense baseline. We also compute the total dense fine-tuning FLOPs for GPT-2 Small and GPT-3 XL on E2E, WebNLG, DART and Curation Corpus, and report them in Table 3. The total training FLOPs during the fine-tuning phase is a small fraction of the total pre-training FLOPs, even though fine-tuning is performed in a dense manner.

Table 3: Details on the total fine-tuning FLOPs for GPT-2 Small and GPT-3 XL on E2E, WebNLG, DART and Curation Corpus tasks. We note that the reported FLOPs per sequence (seq) includes both forward and backward passes.

Dataset	Model	Total Seq	Total FLOPs/ Seq	Total FLOPs	Toal exaFLOPs
E2E	GPT-2 Small	1.26e5	1.36e11	5.15e16	0.052
	GPT-3 XL		1.39e12	5.27e17	0.524
WebNLG	GPT-2 Small	0.54e5	1.36e11	2.21e16	0.022
	GPT-3 XL		1.39e12	2.26e17	0.226
DART	GPT-2 Small	1.25e5	1.36e11	5.12e16	0.051
	GPT-3 XL		1.39e12	5.24e17	0.524
Curation Corpus	GPT-2 Small	0.34e5	1.36e11	1.38e16	0.014
	GPT-3 XL		1.39e12	1.41e17	0.141

B DETAILED EXPERIMENTS ON NLG AND TEXT SUMMARIZATION

In Tables 4, 5 and 6, we provide detailed results on all official evaluation metrics for the E2E, WebNLG and DART tasks. Even across other metrics, we do not observe any significant drop in performance with sparse pre-trained GPT-2 Small and GPT-3 XL models.

Table 4: Downstream accuracy of GPT-2 Small and GPT-3 XL on E2E at different sparsity levels during pre-training. In the metric column, the direction of the arrow indicates better result (e.g., up indicates higher is better).

Model	Pre-Train Sparsity	E2E				
		BLEU↑	NIST↑	METEOR↑	ROUGE-L↑	CIDEr↑
GPT-2 Small	0%	$67.49_{\pm 0.60}$	$8.59_{\pm 0.03}$	$46.08_{\pm 0.22}$	$70.22_{\pm 0.42}$	$2.38_{\pm 0.03}$
	50%	$67.39_{\pm 0.38}$	$8.62_{\pm 0.03}$	$45.89_{\pm 0.18}$	$70.10_{\pm 0.26}$	$2.38_{\pm 0.01}$
	75%	$66.50_{\pm 0.01}$	$8.46_{\pm 0.13}$	$45.61_{\pm 0.32}$	$70.02_{\pm 0.26}$	$2.36_{\pm 0.02}$
GPT-3 XL	0%	$68.10_{\pm 0.46}$	$8.64_{\pm 0.06}$	$46.40_{\pm 0.01}$	$71.03_{\pm 0.22}$	$2.41_{\pm 0.02}$
	50%	$67.98_{\pm 0.63}$	$8.62_{\pm 0.07}$	$46.40_{\pm 0.07}$	$71.30_{\pm 0.20}$	$2.43_{\pm 0.02}$
	75%	$67.66_{\pm 0.59}$	$8.59_{\pm 0.09}$	$46.07_{\pm 0.26}$	$70.40_{\pm 0.26}$	$2.42_{\pm 0.03}$

Table 5: Downstream BLEU, NIST, METEOR, ROUGE-L and CIDEr scores of GPT-2 Small and GPT-3 XL on WebNLG at different sparsity levels during pre-training. In the metric column, the direction of the arrow indicates better result (e.g., down indicates lower is better).

Model	Pre-Train Sparsity	WebNLG		
		BLEU↑	METEOR↑	TER↓
GPT-2 Small	0%	$63.42_{\pm 0.26}$	0.44	0.34
	50%	$63.10_{\pm 0.13}$	0.44	0.34
	75%	$62.64_{\pm 0.22}$	0.43	0.34
GPT-3 XL	0%	$63.62_{\pm 0.23}$	0.45	0.32
	50%	$63.47_{\pm 0.21}$	0.45	0.33
	75%	$63.06_{\pm 0.11}$	0.45	0.33

Table 6: Downstream BLEU, MET and TER scores of GPT-2 Small and GPT-3 XL on DART at different sparsity levels during pre-training. In the metric column, the direction of the arrow indicates better result (e.g., down indicates lower is better).

Model	Pre-Train Sparsity	DART		
		BLEU↑	MET↑	TER↓
GPT-2 Small	0%	$46.30_{\pm 0.16}$	0.38	0.51
	50%	$45.74_{\pm 0.10}$	0.37	0.51
	75%	$44.97_{\pm 0.11}$	0.37	0.52
GPT-3 XL	0%	$47.71_{\pm 0.11}$	0.39	0.49
	50%	$47.10_{\pm 0.13}$	0.39	0.49
	75%	$46.96_{\pm 0.08}$	0.38	0.50

C UNSTRUCTURED SPARSITY ON SPECIALIZED HARDWARE ACCELERATORS

In Figure 1 we highlight the potential realized gains with unstructured weight sparsity on the Cerebras CS-2. This figure was regenerated based on the plot in [Lie, 2021].

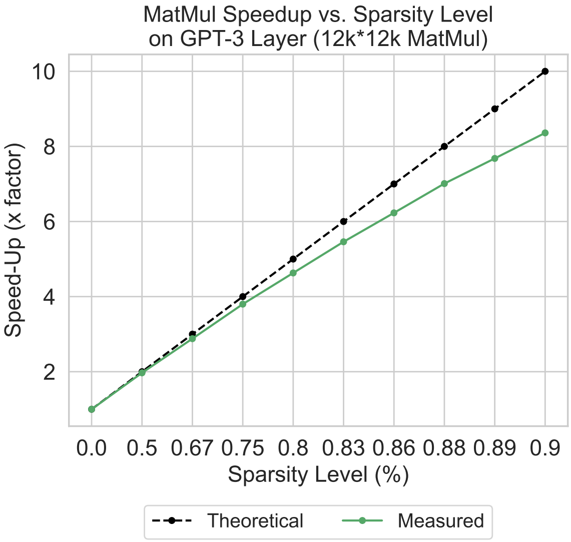

Figure 1: Measured speedup versus theoretical speedup at varying sparsity levels for a GPT-3 layer 12k × 12k matrix multiplication (MatMul) [Lie, 2021].

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *arXiv*, 2016.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *NeurIPS*, 2020.

Curation. Curation corpus base, 2020.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *ICLR*, 2022.

Sean Lie. Thinking outside the die: Architecting the ml accelerator of the future, Nov 2021. URL `https://www.microarch.org/micro54/media/lie-keynote.pdf`.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2017.

Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, et al. Scaling language models: Methods, analysis & insights from training gopher. *arXiv*, 2021.