We need go ~~deeper~~ bigger

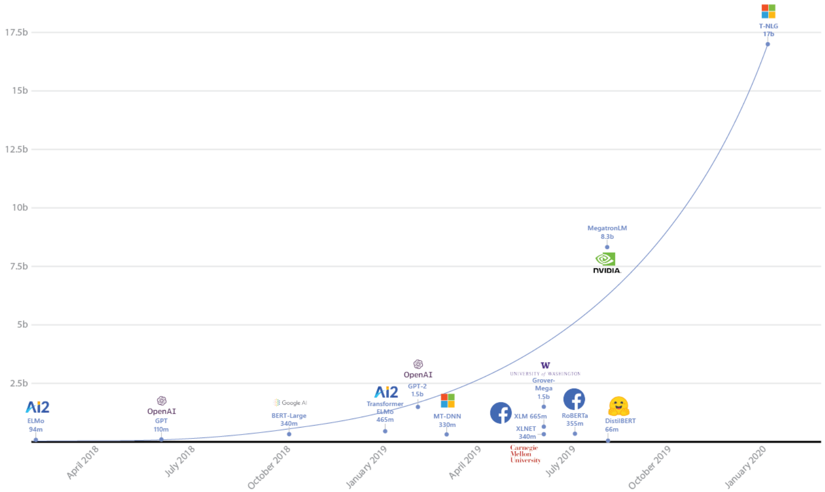
Karol Kaczmarek

Adam Mickiewicz University
Poznań

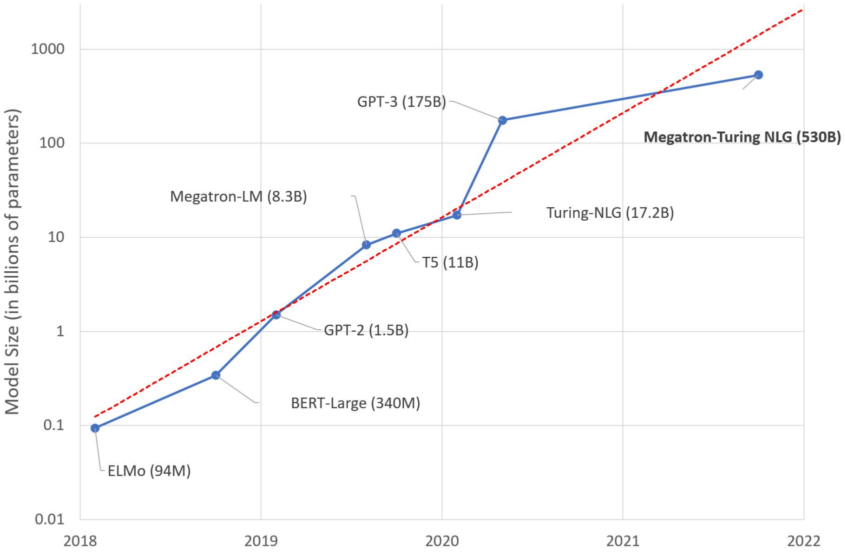
Applica.ai
Warsaw

2022


We need go
bigger
~~deeper~~



TuringNGL - Januray 2020



MT-NLG - October 2021

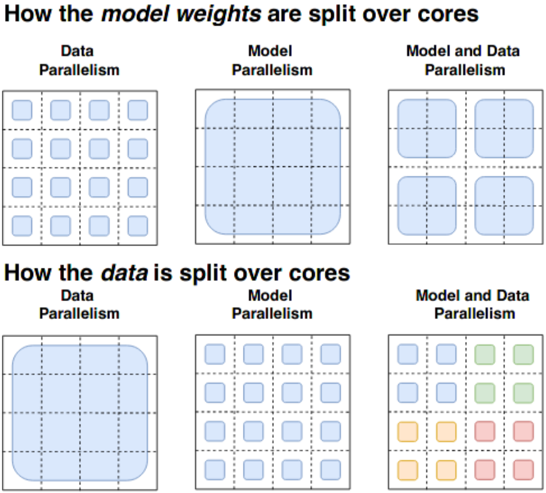
Data Parallelism (DP)

- model parameters are replicated on each device
- at each step, a mini-batch is divided evenly across all the data parallel processes
- each process executes the forward and backward propagation on a different subset of data samples
- use of averaged gradients across processes to update the model locally
- torch.nn.DataParallel, torch.nn.parallel.DistributedDataParallel

Model Parallelism (MP)

- splits the model vertically, partitioning the computation and parameters in each layer across multiple devices
  - **Tensor Model Parallelism** - partitions the individual layers of the model across workers
  - **Pipeline Model Parallelism** - divides the layers of the model into stages that can be processed in parallel
- requiring significant communication between each layer
- manual splitting, Gpipe [1], Pipedream [2], Mesh-Tensorflow [3], Megatron-LM [4] [5], L2L [6], Zero [7] [8] [9], JAX [10] + Haiku [11] (library for JAX), torch.distributed.fsdp.FullyShardedDataParallel [12]

Data Parallelism (DP) and Model Parallelism (MP)



Gopher [13]

- December 2021, Google/DeepMind - JAX and Haiku
- analysis of Transformer-based language model performance across a wide range of model scales and **152 diverse tasks**
- train 6 models: 44M, 117M, 417M, 1.4B, 7.1B, 280B (called **Gopher**)
- achieving state-of-the-art (SOTA) of 100/124 tasks (only 124 tasks has published LM performance)
- analysis of the training dataset and model behavior, covering the intersection of model scale with bias and toxicity, wide range of analyzes (120 page vs. 75 pages GPT-3)

Architecture

- 280B **autoregressive** Transformer (decoder) with modifications:
  - use **RMSNorm** (root mean square layer normalization) instead of LayerNorm
  - use the **relative positional encoding** (allows evaluate on longer sequences than was trained)
- 2048 tokens in sequence length
- **32k** byte-level SentencePiece vocabulary
- train for 300 billion tokens
- mixed precision (bfloat16) training with stochastic rounding
- using Adam optimizer (instead of Adafactor - instabilities of pre-training larger models)

Models

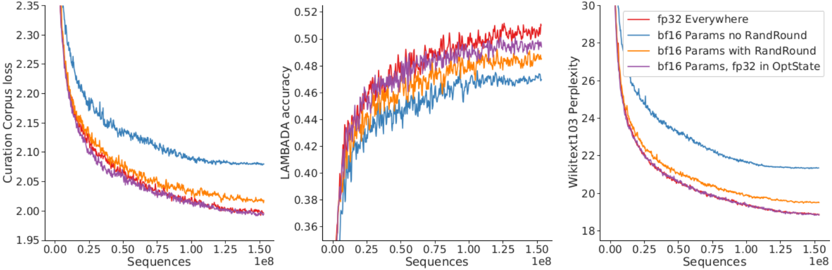
Model	Layers	Heads	d_{model}	Max LR	Batch Size
44M	8	16	512	6×10^{-4}	0.25M
117M	12	12	768	6×10^{-4}	0.25M
417M	12	12	1536	2×10^{-4}	0.25M
1.4B	24	16	2048	2×10^{-4}	0.25M
7.1B	32	32	4096	1.2×10^{-4}	2M
280B Gopher	80	128	16384	4×10^{-5}	3M $\rightarrow$ 6M[1]
175B GPT-3	96	96	12288	0.6×10^{-4}	3.2M

[1] - after warm up

Training

- use **JAX pmap** transformation to efficiently express both data and model parallelism
- trained and evaluated all models on TPUv3 chips
- **half-precision** parameters and **single-precision** Adam state for Gopher occupy **2.5 TiB** (16 GiB of memory in TPUv3 core)
- use **optimiser state partitioning, model parallelism** and **rematerialisation** (checkpointing) to partition the model state and reduce the activations
- create **MassiveText** (web pages, books, news articles and code)

Lower-Precision Training



- **fp32**: Significand precision: 24 bits, Exponent width: 8 bits
- **fp16**: Significand precision: 11 bits, Exponent width: 5 bits
- **bf16**: Significand precision: 8 bits, Exponent width: 8 bits

Lower-Precision Training

- **fp32 Everywhere** - both parameters and activations are stored in fp32
- **bf16 parameters without Random Rounding** - parameters and activations are cast to bp16, no random rounding during the parameter update
- **bf16 parameters with Random Rounding** - parameters and activations are cast to bf16, random rounding during the parameter update
- **bf16 parameters with a fp32 copy in the partitioned optimiser state** - parameters and activations are cast to bf16, copy of the parameters are stored in fp32 in the optimiser state and used for the update, parameters are randomly rounded to bf16 for the forward pass

Train data

	Disk Size	Documents	Tokens	Sampling proportion
MassiveWeb	1.9 TB	604M	506B	48%
Books	2.1 TB	4M	560B	27%
C4	0.75 TB	361M	182B	10%
News	2.7 TB	1.1B	676B	10%
GitHub	3.1 TB	142M	422B	3%
Wikipedia	0.001 TB	6M	4B	2%

Evaluation data

	# Tasks	Examples
Language Modelling	20	WikiText-103, The Pile: PG-19, arXiv, FreeLaw, ...
Reading Comprehension	3	RACE-m, RACE-h, LAMBADA
Fact Checking	3	FEVER (2-way & 3-way), MultiFC
Question Answering	3	Natural Questions, TriviaQA, TruthfulQA
Common Sense	4	HellaSwag, Winogrande, PIQA, SIQA
MMLU	57	High School Chemistry, Atronomy, Clinical Knowledge, ...
BIG-bench	62	Causal Judgement, Epistemic Reasoning, Temporal Sequences, ...

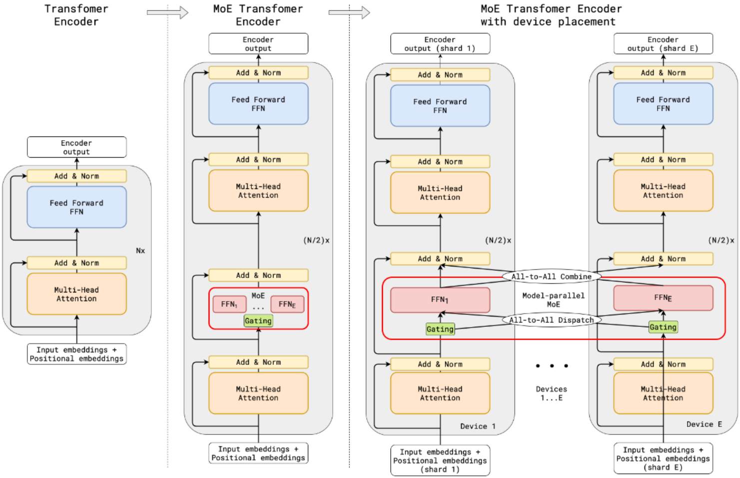
And much much more

About **60 pages** of additional analysis...

GShard [14] – MoE (Mixture-of-Experts)

- June/July 2020, Google
- parallel computation patterns with **minimal changes** to the existing model code GShard enabled us to scale up multilingual neural machine translation Transformer model with Sparsely-Gated Mixture-of-Experts beyond **600B parameters** using automatic sharding
- train 600B model on 2048 TPU v3 accelerators in 4 days to achieve far superior quality for translation from 100 languages to English
- even train 1 trillion model (**problem with stability training**, did not include the results)

Sparsely-Gated Mixture-of-Experts Transformer (MoE Transformer)



ZeRO [7] – DeepSpeed

- October 2019 (blogs - February 2020 [15]), Microsoft - PyTorch
- Turing-NLG - Turing Natural Language Generation - 17 billion parameters
- ZeRO (Zero Redundancy Optimizer) - optimize **memory**, vastly improving **training speed** while increasing **the model size** that can be efficiently trained, constains two sets of optimizations:
  - **ZeRO-DP** (ZeRO-powered data parallelism) aimed at reducing the memory footprint of the model states (removes the memory state redundancies across data-parallel processes by **partitioning** the model states instead of **replicating**)
  - **ZeRO-R** targeted towards reducing the residual memory consumption (**activation** partitioning with CPU offloading, appropriate size for **temporary buffers**, preventing **memory fragmentation**)

ZeRO-Offload [8] – DeepSpeed

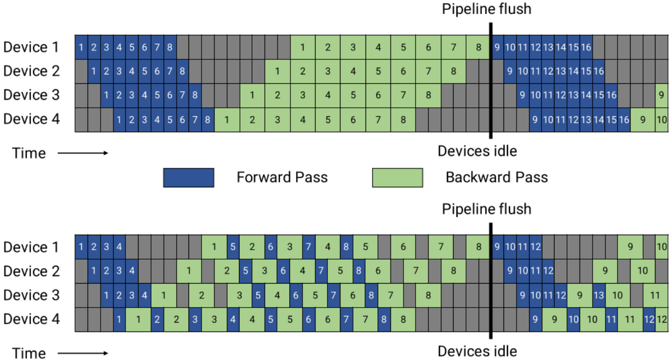
- January 2021, Microsoft - PyTorch
- train models with over 13 billion parameters on a single GPU (Nvidia V100 32GB)
- offload the **gradients, optimizer states and optimizer** computation to CPU, while keeping the parameters and forward and backward computation on GPU
- reduce CPU compute time while maximizing memory savings on GPU to their existing training pipeline (**Fast CPU Adam optimizer** and **One-Step Delayed Parameter Update**)

ZeRO-Infinity [9] – DeepSpeed

- April 2021, Microsoft - PyTorch
- **ZeRO-Infinity** - heterogeneous system technology that leverages GPU, CPU, and NVMe memory to allow for unprecedented model scale on limited resources without requiring model code refactoring
- available through DeepSpeed (ZeRO-Infinity extends the ZeRO with new innovations in heterogeneous memory access called the **infinity offload engine**)
- running 32 trillion parameters on 32 NVIDIA DGX-2 nodes (512 V100 GPUs)
- supports 1 trillion parameters per NVIDIA V100 DGX-2 node - 50x increase over 3D parallelism

Megatron-LM [5]

Megatron-LM (April 2021 - Nvidia) - train 1 trillion parameters on 3072 GPUs (Nvidia A100 80GB - 384 DGX A100 nodes):



and other optimizations (**activation checkpointing**, **CPU-offloading**, **adaptive optimization**).

Megatron-Turing NLG [17]

- January 2022 (blogs - October 2021 [16]), Microsoft - PyTorch
- combines **pipeline parallelism and data parallelism from DeepSpeed** with **tensor-slicing from Megatron** to create **3D-parallelism**
- train **530B** model (Megatron-Turing NLG – MT-NLG)
- use **280** Nvidia A100 GPUs (Nvidia Selene = 560 Nvidia A100):
  - 8-way tensor-slicing within a node
  - 35-way pipeline parallelism across nodes
- achieves SOTA zero-shot, one-shot, and few-shot learning on several NLP benchmarks

Architecture

- 530B **autoregressive** Transformer (decoder): **105** Transformer layers, **20480** model dimensions and **128** attention heads
- **2048** tokens in sequence length and **1920** batch size
- train for 339 billion tokens
- mixed precision (bfloat16) training
- using Adam optimizer
- better weight initialization, lower learning rate (higher learning rate increases the model instability), better Adam parameters

Models

Model	**Layers**	**Heads**	d_{model}	**Max LR**	**Batch Size**
280B Gopher	80	128	16384	4×10^{-5}	$3\text{M} \rightarrow 6\text{M}^1$
175B GPT-3	96	96	12288	0.6×10^{-4}	3.2M
530B MT-NLG	105	128	20480	5×10^{-5}	-

[1] - after warm up

Train data

Dataset	Tokens (billion)	Weights (%)	Epochs
Books3	25.7	14.3	1.5
OpenWebText2	14.8	19.3	3.6
Stack Exchange	11.6	5.7	1.4
PubMed Abstracts	4.4	2.9	1.8
Wikipedia	4.2	4.8	3.2
Gutenberg (PG-19)	2.7	0.9	0.9
BookCorpus2	1.5	1.0	1.8
NIH ExPorter	0.3	0.2	1.8
ArXiv	20.8	1.4	0.2
GitHub	24.3	1.6	0.2
Pile-CC	49.8	9.4	0.5
CC-2020-50	68.7	13.0	0.5
CC-2021-04	82.6	15.7	0.5
Realnews	21.9	9.0	1.1
CC-Stories	5.3	0.9	0.5

GPT-NeoX-20B [18] [19]

- Feburary 2022, EleutherAI - PyTorch
- the **largest publicly accessible** pretrained general-purpose autoregressive language model
- base on Megatron and DeepSpeed
- trained on the Pile (825GiB of raw text data)
- weights alone take up around **40GB** in GPU memory and, due to the tensor parallelism scheme as well as the high memory usage, you will need at **minimum 2 GPUs** with a total of **45GB** of GPU VRAM to run inference, and significantly more for training

Architecture

The same as GPT-3 with changes:

Component	**GPT-3**	**GPT-NeoX-20B**
Positional Embeddings	Absolute	Rotary
Parallel Attention + FF Layers	No	Yes
Sparse Layers	Yes	No

[1] Rotary Positional Embeddings [20] - twist embedding space so that the attention of a token at position m to token at position n is linearly dependent on $m - n$

Other

- **FLAN** [21] - 137B Transformer (like GPT-3), instruction tuning
- **XGLM** [22] - 7.5B multilingual autoregressive Transformer, few-shot learning
- **PanGu** [23] - 200B Chinese Transformer
- **LaMDA** [24] - 137B Transformer specialized for dialog
- **DeepNet** [25] - scale Transformers up to 1000 layers
- **PolyCoder** [26] - 2.7B Transformer based on the GPT-2 architecture, multi-lingual corpus of code, Codex alternative – GitHub Copilot

References I

[1] Y. Huang, C. Yonglong, D. Chen, H. Lee, J. Ngiam, Q. Le, and Z. Chen, "Gpipe: Efficient training of giant neural networks using pipeline parallelism," 2020.

[2] D. Narayanan, A. Harlap, A. Phanishayee, V. Seshadri, N. Devanur, G. Granger, P. Gibbons, and M. Zaharia, "Pipedream: Generalized pipeline parallelism for dnn training," 2019.

[3] N. Shazeer, Y. Cheng, N. Parmar, D. Tran, A. Vaswani, P. Koanantakool, P. Hawkins, H. Lee, M. Hong, C. Young, R. Sepassi, and B. Hechtman, "Mesh-tensorflow: deep learning for supercomputers," 2018.

[4] M. Shoeybi, M. Patwary, R. Puri, and et al., "Megatron-lm: Training multi-billion parameter language models using model parallelism," 2019.

[5] D. Narayanan, M. Shoeybi, J. Casper, P. LeGresley, and et al., "Efficient large-scale language model training on gpu clusters," 2021.

[6] B. Pudipeddi, M. Maral, J. Xi, and S. Bharadwaj, "Training large neural networks with constant memory using a new execution algorithm," 2020.

[7] S. Rajbhandari, J. Rasley, O. Ruwase, and Y. He, "Zero: Memory optimizations toward training trillion parameter models," 2019.

[8] J. Ren, S. Rajbhandari, R. Aminabadi, Yazdani, O. Ruwase, S. Yang, M. Zhang, D. Li, and Y. He, "Zero-offload: Democratizing billion-scale model training," 2021.

[9] S. Rajbhandari, O. Ruwase, J. Rasley, S. Smith, and Y. He, "Zero-infinity: Breaking the gpu memory wall for extreme scale deep learning," 2021.

[10] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, "Jax: composable transformations of python+numpy programs," 2018.

[11] T. Hennigan, T. Cai, T. Norman, and I. Babuschkin, "Haiku: Sonnet for jax," 2020.

References II

[12] “Introducing pytorch fully sharded data parallel (fsdp) api.” https://pytorch.org/blog/introducing-pytorch-fully-sharded-data-parallel-api/, March 14, 2022 (accessed March 29, 2021).

[13] W. J. Rae, S. Borgeaud, T. Cai, and et al., “Scaling language models: Methods, analysis & insights from training gopher,” 2021.

[14] D. Lepikhin, H. Lee, and Y. Xu, “Gshard: Scaling giant models with conditional computation and automatic sharding,” 2020.

[15] “Turing-nlg: A 17-billion-parameter language model by microsoft.” https://www.microsoft.com/en-us/research/blog/turing-nlg-a-17-billion-parameter-language-model-by-microsoft/, February 13, 2020 (accessed March 29, 2021).

[16] “Using deepspeed and megatron to train megatron-turing nlg 530b, the world’s largest and most powerful generative language model.” https://www.microsoft.com/en-us/research/blog/using-deepspeed-and-megatron-to-train-megatron-turing-nlg-530b-the-worlds-largest-and-most-powerful-ge October 11, 2021 (accessed March 29, 2021).

[17] S. Smith, M. Patwary, and et al., “Using deepspeed and megatron to train megatron-turing nlg 530b, a large-scale generative language model,” 2022.

[18] S. Black, S. Biderman, E. Hallahan, Q. Anthony, L. Gao, L. Golding, H. He, C. Leahy, K. McDonell, J. Phang, M. Pieler, U. S. Prashanth, S. Purohit, L. Reynolds, J. Tow, B. Wang, and S. Weinbach, “GPT-NeoX-20B: An open-source autoregressive language model,” 2022.

[19] “Announcing gpt-neox-20b.” https://blog.eleuther.ai/announcing-20b/, February 2, 2022 (accessed March 29, 2021).

[20] J. Su, Y. Lu, S. Pan, B. Wen, and Y. Liu, “Roformer: Enhanced transformer with rotary position embedding,” 2021.

[21] J. Wei, M. Bosma, V. Y. Zhao, K. Guu, and et al., “Finetuned language models are zero-shot learners,” 2021.

References III

[22] X. V. Lin, X. Li, and et al., "Few-shot learning with multilingual language models," 2021.

[23] W. Zeng, X. Ren, T. Su, H. Wang, Y. Liao, Z. Wang, X. Jiang, Z. Yang, K. M. Wang, X. Zhang, C. Li, Z. Gong, Y. Yao, X. Huang, J. Wang, J. Yu, Q. Guo, Y. Yu, Y. Zhang, J. Wang, H. Tao, D. Yan, Z. Yi, F. Peng, F. Jiang, H. Zhang, L. Deng, Y. Zhang, Z. Lin, C. Zhang, S. Zhang, M. Guo, S. Gu, G. Fan, Y. Wang, X. Jin, Q. Liu, and Y. Tian, "Pangu-α: Large-scale autoregressive pretrained chinese language models with auto-parallel computation," 2021.

[24] A. D. Cohen, A. Roberts, A. Molina, A. Butryna, A. Jin, A. Kulshreshtha, B. Hutchinson, B. Zevenbergen, B. H. Aguera-Arcas, C. ching Chang, C. Cui, C. Du, D. D. F. Adiwardana, D. Chen, D. D. Lepikhin, E. H. Chi, E. Hoffman-John, H.-T. Cheng, H. Lee, I. Krivokon, J. Qin, J. Hall, J. Fenton, J. Soraker, K. Meier-Hellstern, K. Olson, L. M. Aroyo, M. P. Bosma, M. J. Pickett, M. A. Menegali, M. Croak, M. Díaz, M. Lamm, M. Krikun, M. R. Morris, N. Shazeer, Q. V. Le, R. Bernstein, R. Rajakumar, R. Kurzweil, R. Thoppilan, S. Zheng, T. Bos, T. Duke, T. Doshi, V. Prabhakaran, W. Rusch, Y. Li, Y. Huang, Y. Zhou, Y. Xu, and Z. Chen, "Lamda: Language models for dialog applications," 2022.

[25] H. Wang, S. Ma, L. Dong, S. Huang, D. Zhang, and F. Wei, "Deepnet: Scaling transformers to 1, 000 layers," 2022.

[26] F. F. Xu, U. Alon, G. Neubig, and V. J. Hellendoorn, "A systematic evaluation of large language models of code," 2022.