Natural Language Processing

Berkeley NLP

Scaling and Systems

Kevin Lin – UC Berkeley

March 22, 2023

Scaling and Systems

Announcements

- Spring Break Next Week
- Panels Week After Spring Break
- HW4 Bug
- HW5 Testing
- Today
  - What to scale?
  - How to scale?

Scaling With Fixed Compute

ed CS 288 – Ed Discussion

hahhah

I made the network with 4096 hidden units. It finally achieved 66% accuracy!

I guess brute force really works.

Reply Edit Delete ···

Scaling With Fixed Compute

Model	Size (# Parameters)	Training Tokens
LaMDA (Thoppilan et al., 2022)	137 Billion	168 Billion
GPT-3 (Brown et al., 2020)	175 Billion	300 Billion
Jurassic (Lieber et al., 2021)	178 Billion	300 Billion
Gopher (Rae et al., 2021)	280 Billion	300 Billion
MT-NLG 530B (Smith et al., 2022)	530 Billion	270 Billion
Chinchilla	70 Billion	1.4 Trillion

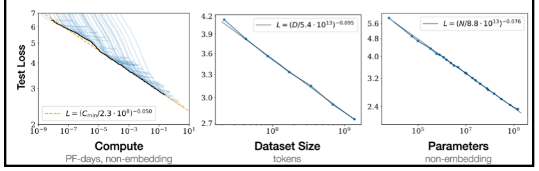
Fixed Compute

- Scaling Laws for Neural Language Models (Kaplan et al., 2020)

- N – the number of model parameters, *excluding all vocabulary and positional embeddings*
- $C \approx 6NBS$ – an estimate of the total non-embedding training compute, where B is the batch size, and S is the number of training steps (ie parameter updates). We quote numerical values in PF-days, where one PF-day $= 10^{15} \times 24 \times 3600 = 8.64 \times 10^{19}$ floating point operations.

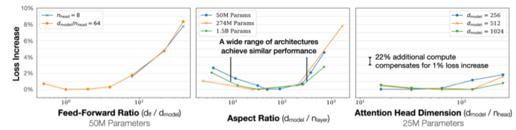
Scaling “Laws”

- N number of parameters
- D size of dataset
- C amount of compute

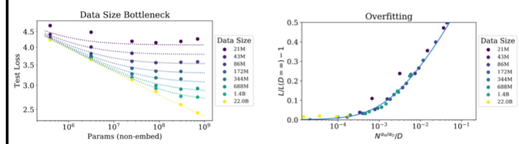


Model Shape

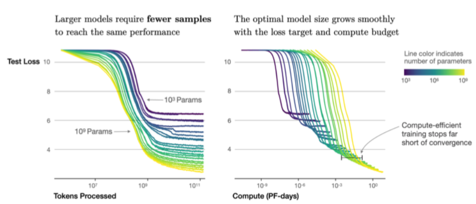
- Strong dependence on scale, weak dependence on shape



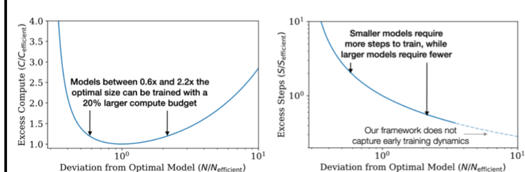
Data Size and Overfitting



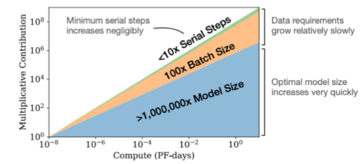
Sample Efficiency



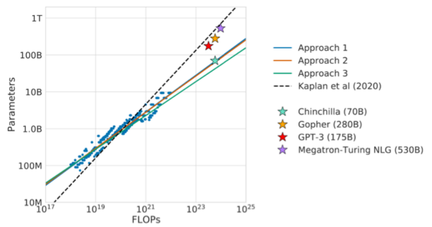
Convergence Inefficient



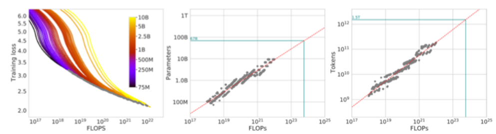
Optimal Scaling



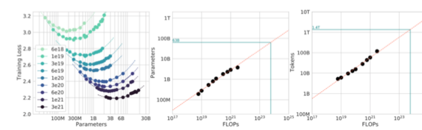
Chinchilla Scaling



Fix Model Size, Vary Tokens

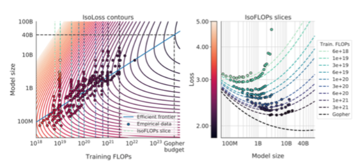


IsoFLOPs



Fit Parametric Loss Function

$$\hat{L}(N, D) \triangleq E + \frac{A}{N^{\alpha}} + \frac{B}{D^{\beta}}$$



BIG-bench

1.Step Inference. Given a prompt goal and four candidate steps, choose the correct *step* that helps achieve the goal. For example:

Which step is likely to help achieve the goal “prevent coronavirus”?
A. wash your hands B. wash your cat C. clap your hands D. eat your protein

2.Goal Inference. Given a prompt step and four candidate goals, choose the correct *goal* that the step helps achieve. For example:

Which is the most likely goal of “choose a color of lipstick”?
A. get pink lips B. read one’s lips C. lip sync D. draw lips

3.Step Ordering. Given a prompt goal and two steps, determine which step temporally precedes the other. For example:

In order to "clean silver," which step should be done first?
A. dry the silver B. handwash the silver

Task: goal_step_wikihow

BIG-bench

1.When Max was applying for a new job after he got fired, he wrote his resume and cover letters so creatively that he got offered a copywriter job. Which of the following proverbs best apply to this situation?,

2.Where there is a will, there is a way.

3.Where one door shuts, another opens

4.You can catch more flies with honey than with vinegar

Task: English_proverbs

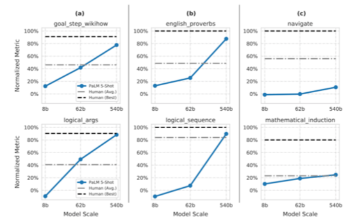
BIG-bench

Turn right. Take 1 step. Turn right. Take 6 steps. Turn right. Take 1 step. Turn right. Take 2 steps. Take 4 steps.

Are you back at the start?

Task: navigate

“Emergence”



BIG-bench

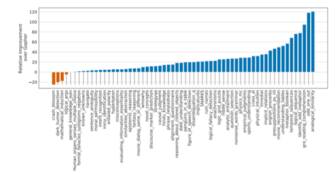


Figure 7 | **BIG-bench results compared to *Gopher*** *Chinchilla* out performs *Gopher* on all but four BIG-bench tasks considered. Full results are in Table A7.

Are We Optimal?

	Compute budget (C)	Notes	alpha	beta	a	b	A (fitted)	B (fitted)	G (derived from eq 4)	N_opt(C): optimal model size	D_opt(C): optimal dataset size
OPT-175B	4.30E+23	From eq 10 in Appendix D.2.	0.34	0.28	0.452	0.548	406.4	410.7	1.3447	28.2B	2540.6B
	4.30E+23	Approach 1 (min over training curves)	0.34	0.34	0.5	0.5	406.4	410.7	0.9846	263.6B	271.9B
	4.30E+23	Approach 1 (min over training curves)	0.28	0.28	0.5	0.5	406.4	410.7	0.9814	262.7B	272.8B
	4.30E+23	Approach 2 (IsoFLOP profiles)	0.34	0.33	0.49	0.51	406.4	410.7	1.0452	165.3B	433.5B
	4.30E+23	Approach 2 (IsoFLOP profiles)	0.28	0.27	0.49	0.51	406.4	410.7	1.0552	166.9B	429.4B
	4.30E+23	Approach 3 (Parametric loss)	0.34	0.29	0.46	0.54	406.4	410.7	1.2686	41.4B	1731.9B
	4.30E+23	Approach 3 (Parametric loss)	0.28	0.24	0.46	0.54	406.4	410.7	1.3350	43.5B	1645.8B

Source: Susan Zhang

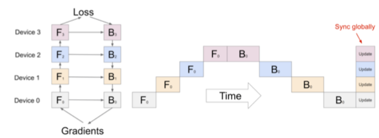
Systems

- Parallelism
  - Data
  - Model
  - Tensor
- Memory Optimization

Data Parallel

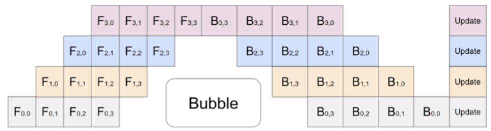
- Bulk synchronous parallel: sync at end of every minibatch
  - Pros: higher learning efficiency
  - Cons: wait for all machines
- Asynchronous parallel: apply updates when ready
  - Pros: No wait
  - Cons: lower learning efficiency

Model Parallel



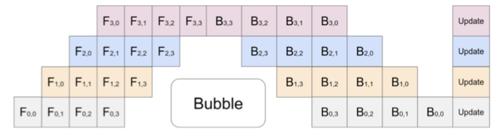
Pipeline Parallel

- GPipe (Huang et al., 2019)



Pipeline Parallel

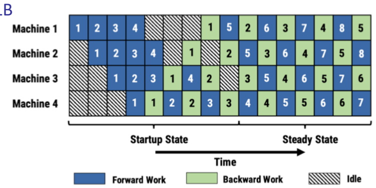
- GPipe (Huang et al., 2019)



$$1 - \frac{2md}{(2m + 2(d-1))d} = \frac{d-1}{m+d-1}$$

Pipeline Parallel

- Pipedream (Narayanan et al., 2019)
- 1F1B



Pipeline Parallel

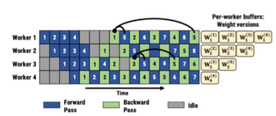


Figure 9: Weight stashing as minibatch 5 flows across stages. Arrows point to weight versions used for forward and backward passes for minibatch 5 at the first and third stages.

Pipeline Parallel

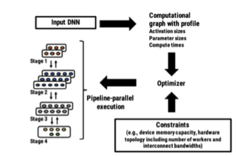


Figure 6: PipeDream's automated mechanism to partition DNN layers into stages. PipeDream first profiles the input DNN, to get estimates for each layer's compute time and output size. Using these estimates, PipeDream's optimizer partitions layers across available machines, which is then executed by PipeDream's runtime.

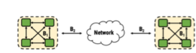


Figure 7: An example 2-level hardware topology. Solid green boxes represent GPUs. Each server (dashed yellow boxes) has 4 GPUs connected internally by links of bandwidth B_1; each server is connected by links of bandwidth B_2. In real systems, $B_1 > B_2$. Figure best seen in color.

Pipeline Parallel

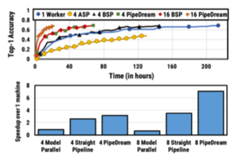


Fig. 5. Results for VGG16 on ILSVRC12. (Top) Accuracy vs time. The integer marks the number of stage workers. ASP = Asynchronous parallel & BSP = Bulk synchronous parallels. (Bottom) Training time speedup for different parallelism configurations. Straight pipeline refers to pipeline parallelism without data parallelism. (Image source: Harlap et al. 2018)

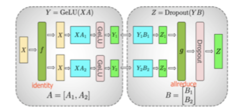
Pipeline Parallel

Fig. 6. Illustration of pipeline scheduling in PipeDream-flush. (Image source: (Narayanan et al. 2021)

Fig. 7. Illustration of pipeline scheduling in PipeDream-2BW. (Image source: (Narayanan et al. 2021)

Tensor Parallel

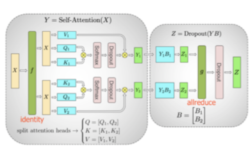
- Megatron-LM (Shoeybi et al., 2020)



(a) MLP

$$\text{Split } A = [A_1, A_2]$$
$$Y = \text{GeLU}(XA)$$
$$[Y_1, Y_2] = [\text{GeLU}(XA_1), \text{GeLU}(XA_2)]$$

Tensor Parallel



(b) Self-Attention

$$\text{Attention}(X, Q, K, V) = \text{softmax}\left(\frac{(XQ)(XK)^\top}{\sqrt{d_k}}\right)XV$$

Putting It Together

- PTD-P (Narayanan et al., 2021)

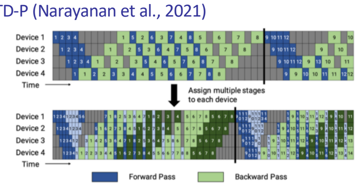


Fig. 9. (Top) Default `1F1B` pipeline schedule as in PipeDream-flush. (Bottom) Interleaved 1F1B pipeline schedule. First model chunks are in dark colors and second chunks are in light colors. (Image source: Narayanan et al. 202))

Where did the memory go?

- GPT-2 (1.5B) 3GB weight
- **Model states:** momentum and variance in Adam, gradients, parameters
- **Residual states:** activation, temporary buffer, unusable fragmented memory

Model States

- *Example.* Transformer architecture trained with Adam
- Ψ parameters with mixed precision training (use F16 and F32)
- F16 copies of **params (2Ψ bytes)** and **gradients (2Ψ bytes)**
- F32 copies of **params (4Ψ bytes)**, **momentum (4Ψ bytes)** and **variance (4Ψ bytes)**
- **= 16Ψ bytes**, at least **24GB**

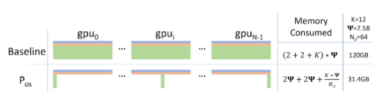
Residual States

- **Activations:** 1.5B transformer, around 60 GB even with activation checkpointing
- **Temporary buffers**: gradient all-reduce, norm computation, etc. around 5GB
- **Memory fragmentation:** 30% of memory still available when OOM

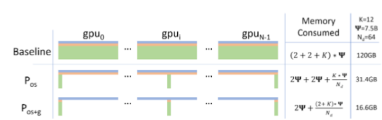
ZeRO Redundancy Optimizer

- Memory Optimizations Towards Training Trillion Parameter Models (Rajbandari et al., 2019)
- **ZeRO-DP** optimizes model states
- **ZeRO-R** optimizes residual states

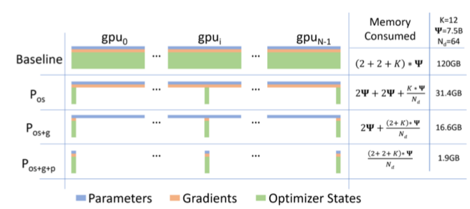
ZeRO-DP



ZeRO-DP



Memory Optimization



ZeRO-R

- **Partitioned Activation Checkpointing:** Once forward prop for a layer is computed, partition the input activations until needed for backprop
- **Constant size buffer:** computational efficiency can depend on input size, eg. All-reduce achieves higher bandwidth than a smaller one
- **Memory Defragmentation:** pre-allocate contiguous memory chunks for activation checkpoints

Scaling for Varying Model Sizes

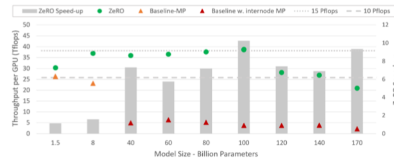


Figure 2: *ZeRO* training throughput and speedup w.r.t SOTA baseline for varying model sizes. For *ZeRO*, the MP always fit in a node, while for baseline, models larger than 40B require MP across nodes.

Superlinear Scaling for Increasing GPUs

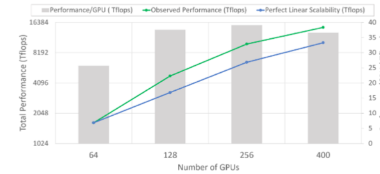


Figure 3: Superlinear scalability and per GPU training throughput of a 60B parameter model using *ZeRO*-100B.

Turing-NLG

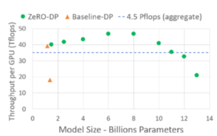


Figure 4: Max model throughput with *ZeRO*-DP.

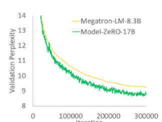


Figure 5: SOTA Turing-NLG enabled by *ZeRO*.