Utilizing multiple models for accelerated model growth

Jonah Ghebremichael, Zongqing Qi

Increasing LLM performance

- As we increase layers, neurons, dataset size and computer power we have a direct increase in model performance.
- Therefore in order to increase the strength of LLM we must increase their size.

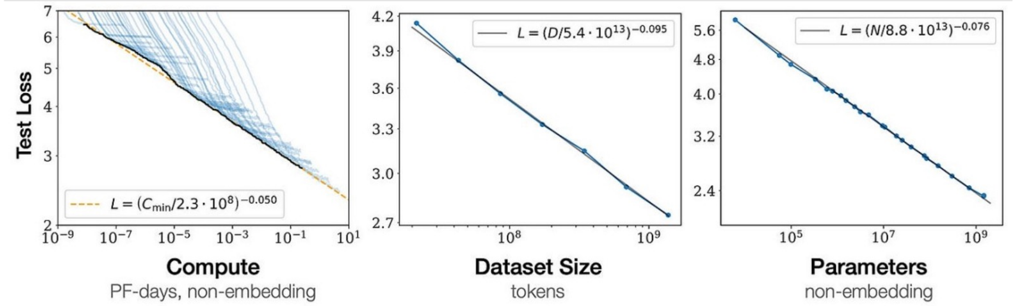
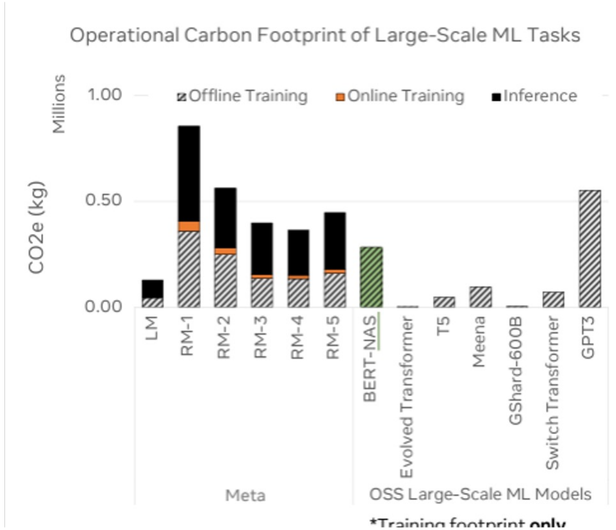


Figure 1 Language modeling performance improves smoothly as we increase the model size, datasetset size, and amount of compute[2] used for training. For optimal performance all three factors must be scaled up in tandem. Empirical performance has a power-law relationship with each individual factor when not bottlenecked by the other two.

Cost of LLM Training

- A significant problem with pre-training Larger Language Models is energy consumption and environmental impact.
- Efficiently pre-training LLMs is becoming more and more essential to ensure widespread growth and adoption of AI.



Sustainable AI: Environmental Implications, Challenges and Opportunities: arXiv:2111.00364

Model Growth

- Promising research direction is to utilize smaller (base) models to expedite the training of larger (target) models known as **model growth**
- Model growth studies how to leverage trained parameters from smaller models to initialize the larger model's parameters.
- Current methods show promising speedup results on models such as BERT
- Despite this it is not widely utilized for any current LLM pre-training except FLM-101B

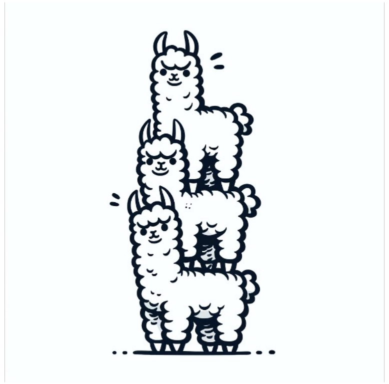
  [FLM-101B: An Open LLM and How to Train It with $100K Budget (2023)]

Related Work

- Pioneering work of Net2Net(2015) marks the first time an attempt to study model growth in the deep learning era. Expanded width and depth while keeping original functions via randomly splitting old neurons and injecting new identity layers.
- Many different implementations of expanding layers and neurons. Most utilize BERT.
  - Staged Grow(2022) doubles the width by concatenating two identical layers and halves final loss to keep function preserving.
  - StackedBert(2023) simply stacks duplicated layers to form a deeper model.
  - Bert2Bert(2021) BERT-based extension of the widthwise Net2Net.
  - Lemon(2023) suggests integrating a parameter into the splitting of neurons in Bert2Bert, aiming to break weight symmetry.
  - LiGO(2023) in contrast to the direct copy/split approaches of the rest, presents a learning-based method that initializes the larger model's parameters via learning a linear mapping from the smaller model's parameters.
  - MSG(2024) a multi-staged growing strategy that progressively expands transformer components, where the newly grown neurons are randomly initialized using a masking mechanism to ensure function preservation.
  - GradMax(2022) and Lemon(2023) assigned specific values, like zero, to the newly initialized neurons to negate their influence

Stacking Your Transformers(2024)

- Recent paper(May 2024) that provides a systematic review of model growth techniques.
- Currently under submission.
- Notes that despite reported gains in pre-training performance, practitioners have not yet adopted model growth techniques.
- Presents 3 obstacles that prevent practitioners from adopting model growth



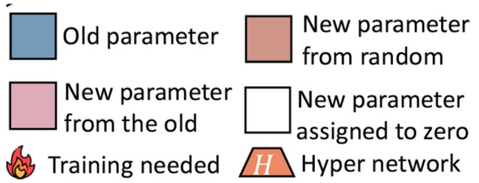
3 Obstacles

1. **Lack of comprehensive assessment**: Results on the effect of model growth on LLM pre-training lack a baseline comparison. In addition most approaches are evaluated on encoder-based BERT models and not decoder-based LLMs
2. **Untested scalability:** Two aspects.
    a. *Model size*: Existing approaches are only evaluated on smaller-scale BERT models, it is unclear whether these growth methods will continue accelerating training with Large-Scale LLMS.
    b. *Amount of Pretraining*: There are debates over whether certain efficient training strategies may initially converge faster but perform similarly or worse when given more training data.
3. **Lack of Empirical Guidelines**: While scaling laws give empirical guidelines on pre-training computational-optimized LLMs, there is a lack of empirical guidelines on growth techniques.

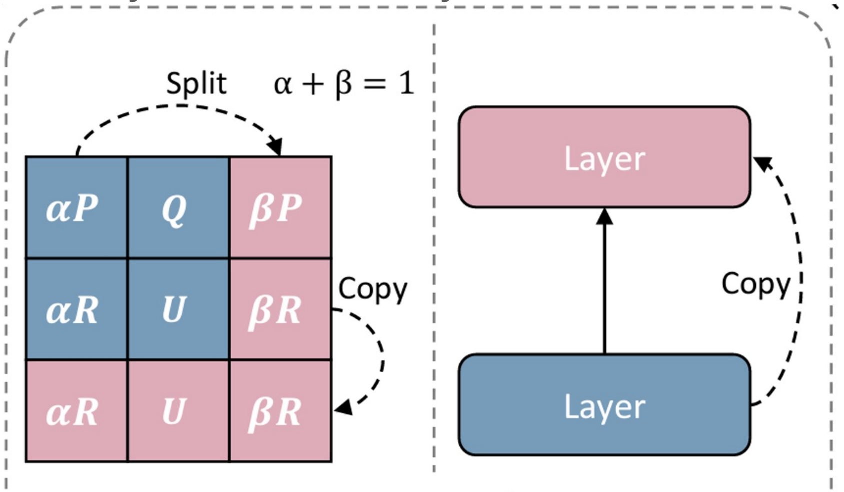
Growth Operators

Separated methods into 4 **atomic growth operators**

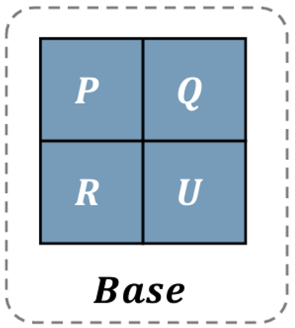
- Directly duplicating and stacking old layers in a depthwise manner or splitting neurons in the same layer width wisely denoted as *Gdirect*.



Old parameter
New parameter from random
New parameter from the old
New parameter assigned to zero
Training needed
H
Hyper network



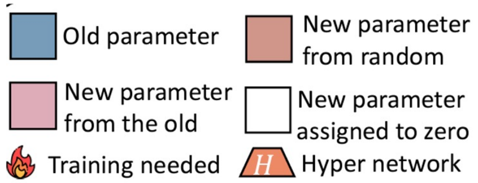
(a) $G^{\rightarrow}_{direct}$ and $G^{\uparrow}_{direct}$ (G_{stack})

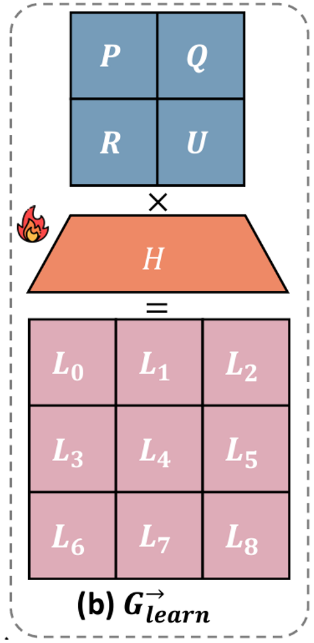


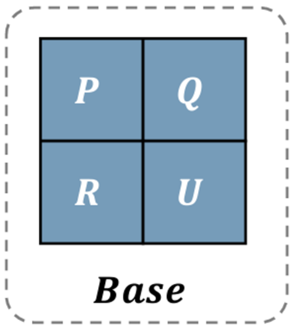
Growth Operators

Separated methods into 4 **atomic growth operators**

- Generating expanded parameters using a learnable mapping matrix to the existing parameters, denoted as *Glearn.*


Old parameter
New parameter from random
New parameter from the old
New parameter assigned to zero
Training needed
H
Hyper network

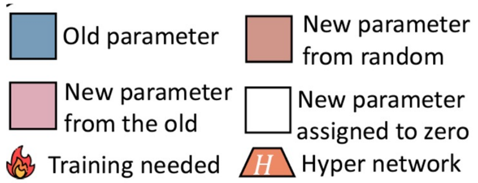




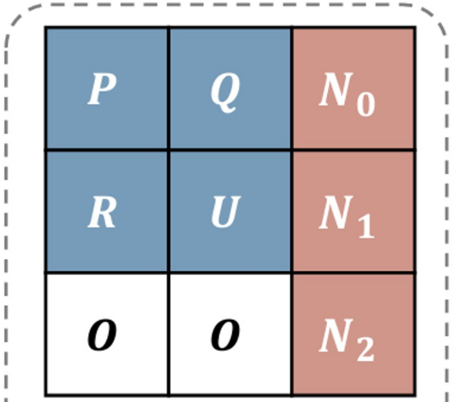
Growth Operators

Separated methods into 4 **atomic growth operators**

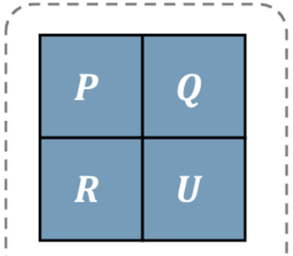
- Setting the new parameters to zero, denoted as *Gzero.*



Old parameter
New parameter from random
New parameter from the old
New parameter assigned to zero
Training needed
H
Hyper network



(c) $G^{\rightarrow}_{zero}$

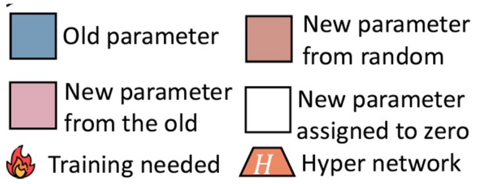


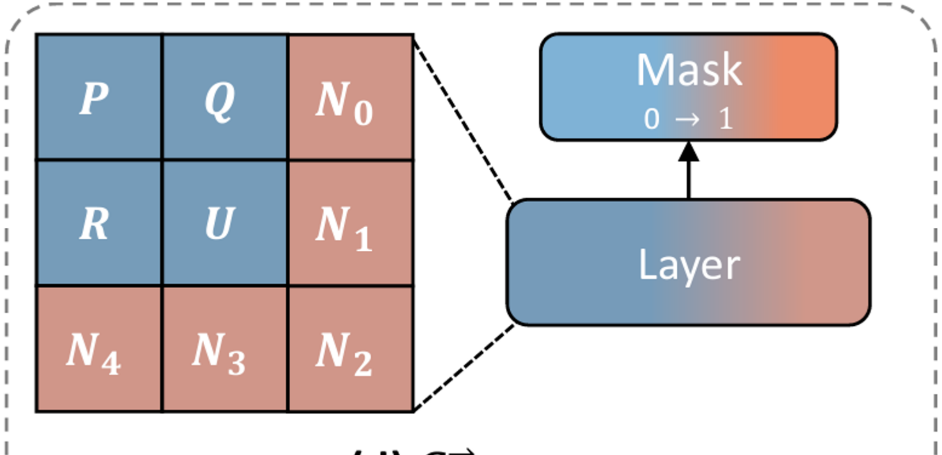
Base

Growth Operators

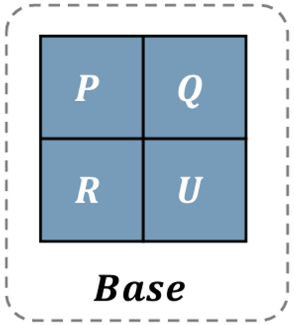
Separated methods into 4 **atomic growth operators**

- Randomly initializing the new parameters, denoted as *Grandom*.


Old parameter
New parameter from random
New parameter from the old
New parameter assigned to zero
Training needed
H
Hyper network

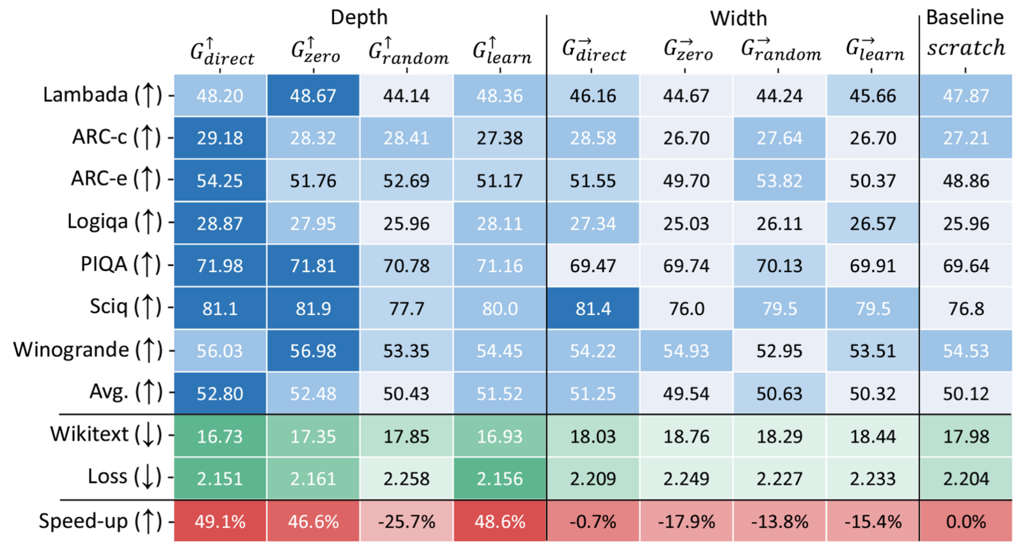


(d) $G^{\rightarrow}_{random}$



Results of Pre-Training 1.1B LLMs

- Report results on training loss, eight standard Harness NLP benchmarks along with the average accuracy and the speedup ratio
- Depthwise growth has significant over widthwise
- In fact widthwise does not seem to have any advantages
- **Depthwise stacking G↑ direct emerges as the clear winner among growth operators, surpassing its competitors in speedup, training loss and nearly every Harness evaluation metric.**



	Depth				Width				Baseline
	$G^{\uparrow}_{direct}$	$G^{\uparrow}_{zero}$	$G^{\uparrow}_{random}$	$G^{\uparrow}_{learn}$	$G^{\rightarrow}_{direct}$	$G^{\rightarrow}_{zero}$	$G^{\rightarrow}_{random}$	$G^{\rightarrow}_{learn}$	*scratch*
Lambada (↑)	48.20	48.67	44.14	48.36	46.16	44.67	44.24	45.66	47.87
ARC-c (↑)	29.18	28.32	28.41	27.38	28.58	26.70	27.64	26.70	27.21
ARC-e (↑)	54.25	51.76	52.69	51.17	51.55	49.70	53.82	50.37	48.86
Logiqa (↑)	28.87	27.95	25.96	28.11	27.34	25.03	26.11	26.57	25.96
PIQA (↑)	71.98	71.81	70.78	71.16	69.47	69.74	70.13	69.91	69.64
Sciq (↑)	81.1	81.9	77.7	80.0	81.4	76.0	79.5	79.5	76.8
Winogrande (↑)	56.03	56.98	53.35	54.45	54.22	54.93	52.95	53.51	54.53
Avg. (↑)	52.80	52.48	50.43	51.52	51.25	49.54	50.63	50.32	50.12
Wikitext (↓)	16.73	17.35	17.85	16.93	18.03	18.76	18.29	18.44	17.98
Loss (↓)	2.151	2.161	2.258	2.156	2.209	2.249	2.227	2.233	2.204
Speed-up (↑)	49.1%	46.6%	-25.7%	48.6%	-0.7%	-17.9%	-13.8%	-15.4%	0.0%

Figure 3: We evaluate operators using training loss and Lambada [30], ARC-c [31], ARC-e [31], Logiqa [32], PIQA [33], Sciq [34], Winogrande [35] and Wikitext PPL [36] totaling eight standard NLP benchmarks. After 8×10^{20} FLOPs of training, $G^{\uparrow}_{\text{direct}}$ demonstrates a significant speedup.

Our Approach

- To our knowledge all existing model growth approaches use one small model to initialize the weights for one larger model
- The idea behind model growth is that larger models can utilize the understanding stored in the parameters of smaller models
- The problem is in order to expand the model it requires duplicating, zeroing or randomizing the extra parameters
- If we could use multiple smaller models, trained on different subsets of data, we believe we could achieve greater pre-training results by incorporating the layers of each smaller model into the larger model

Model Structures

https://github.com/karpathy/minGPT Author: karpathy, openai co-founder

model name	layers	heads	emb dimension	language level
gpt-mini	6	6	192	lv0
gpt2	12	12	768	lv1
gpt2-large	36	20	1280	lv1/lv2

data example

Lv0:

Ducks lug and tug

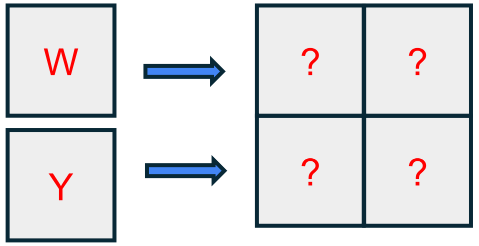
Ducks hug

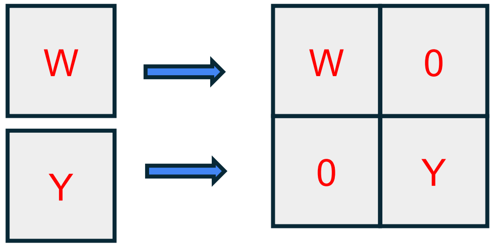
Lv1:

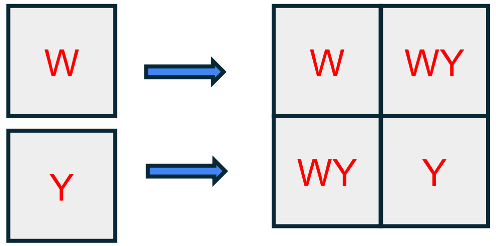
Her feet push the water so she can swim fast.

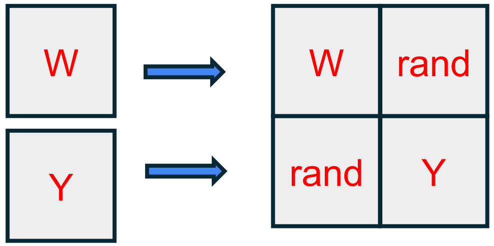
Before the kids knew it, they were up in a tree.

Model Design - Width Expansion



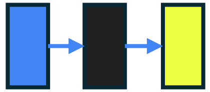




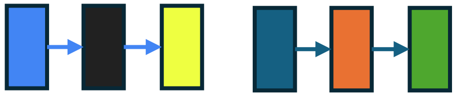


Model Design - Depth Expansion

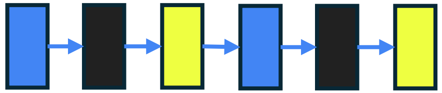
1 model

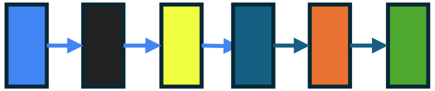


multiple models

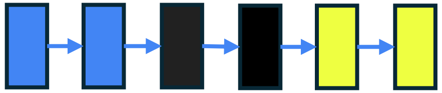


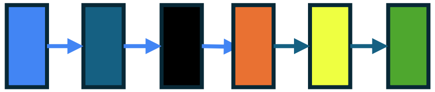
StackedBert(2023) Approach





Interweaving





Model Design - LiGO

- LiGO(2023) involves learning a linear mapping from the smaller model’s parameters to initialize the larger model’s parameters.
- This is achieved by factoring the linear transformation into width- and depth-growth operators.
- This is the most complicated combination method employed and we are still working on creating a learnable linear mapping for multiple models

Next Steps

- Recreate the results from Stacking Your Transformers(2024), they include their code here. It requires training a models with 627B tokens so this will take time.
- Implement Depth and Width expansion model growth for multiple base models in mingpt
- Design an approach for LiGO model expansion for multiple base models
- Implement LiGO approach
- Compare the training methods to the scratch approach(without model growth) in prediction accuracy and training cost, and conduct the profile-based analysis on the training latency, memory usage, and GPU core consumption with Nsight Systems/Compute.

Thank you!