

Project 1: Growing sparse large language models

Work done at my last internship.  **Numenta**

Can we start with a small parameter language model and gradually expand it into a larger model that still performs well but has fewer parameters than very large models?

Sparse LLMs is an incredibly relevant topic right now!

OpenAI

November 13, 2025 Research Publication

Understanding neural networks through sparse circuits

We trained models to think in simpler, more traceable steps—so we can better understand how they work.

[Read the paper ↗](#)

Dense circuits vs sparse circuits

Dense

Circuit Sparsity

In normal dense neural networks, each neuron is connected to every neuron in the next layer. In our sparse models, each neuron only connects to a few neurons in the next layer. We hope that this makes the neurons, and the network as a whole, easier to understand.

Current techniques to pruning/sparsifying LLMs, involve training a large dense model and then pruning weights post-training to be sparse without loss in accuracy.

SparseGPT: Massive Language Models Can be Accurately Pruned in One-Shot

Elias Frantar¹ Dan Alistarh^{1,2}

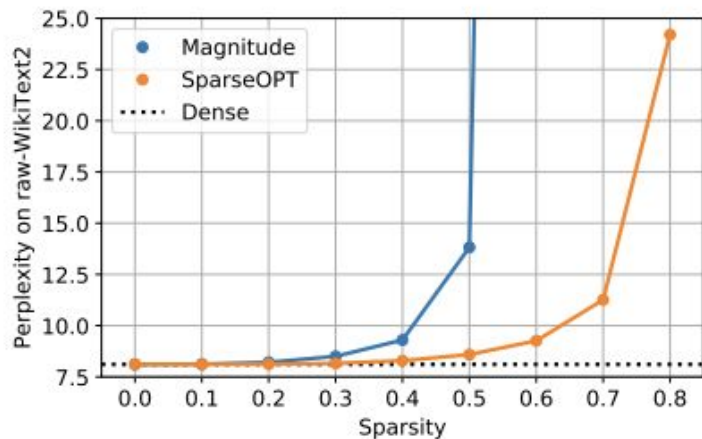
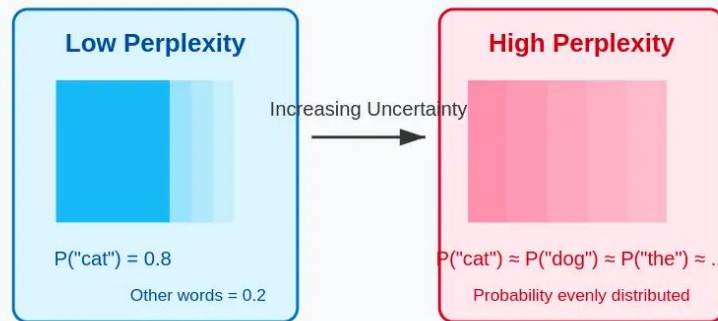


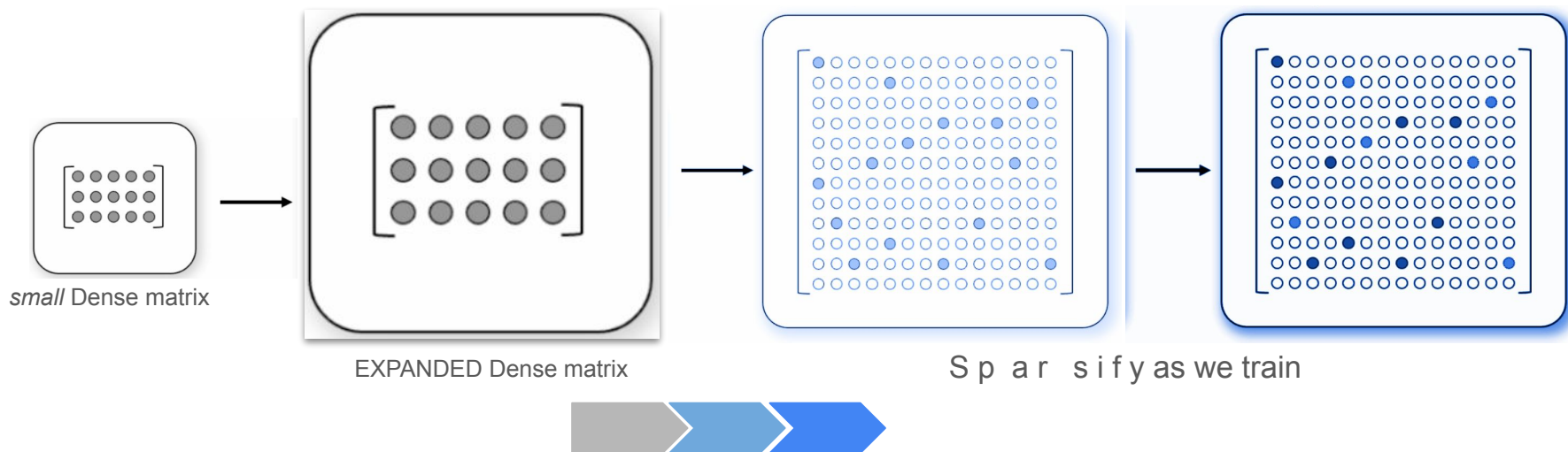
Figure 5. Uniform pruning BLOOM-176B.

Perplexity: Model Uncertainty Visualization



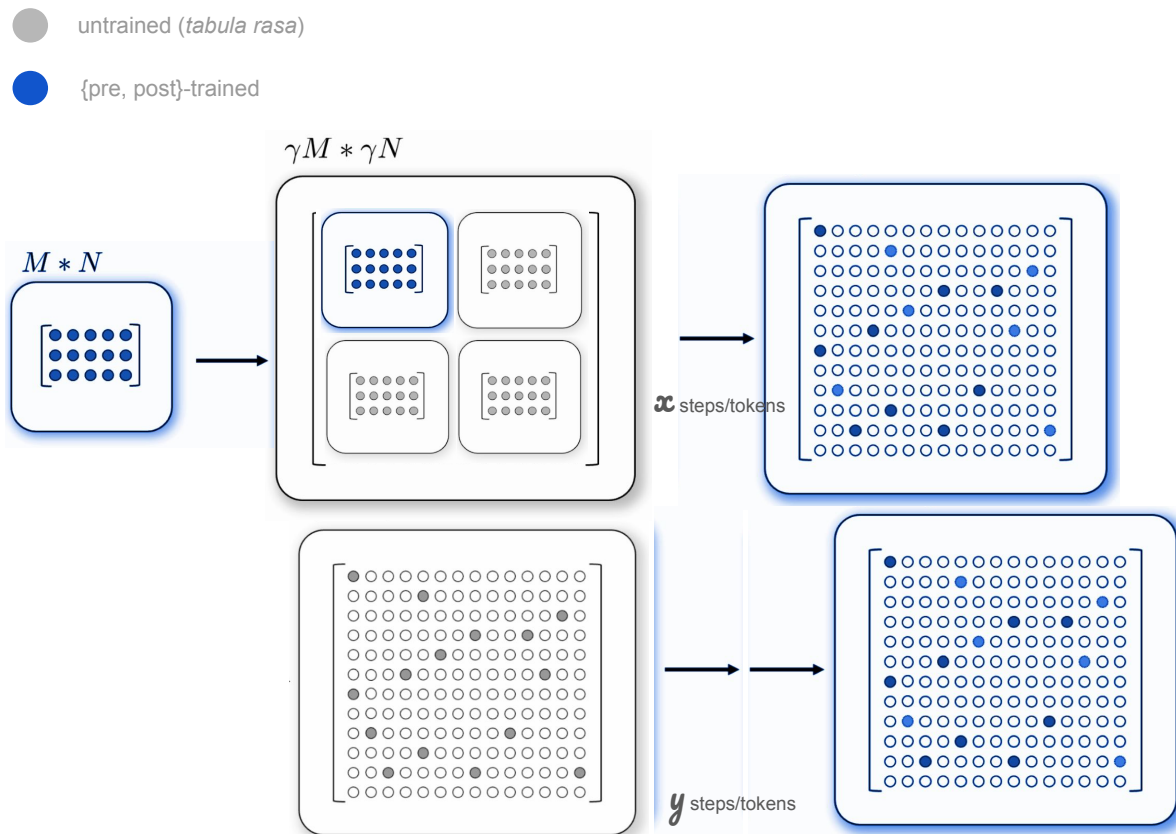
Another idea is to start from an untrained small dense model, expanding the dimension and sparsifying as we train. But this is often less efficient and does not perform as well as just training a large model from scratch.

- untrained
- some training
- trained training



Can we “transplant” the weights of a **small pretrained** model into a **large untrained** one to cause that larger model to:

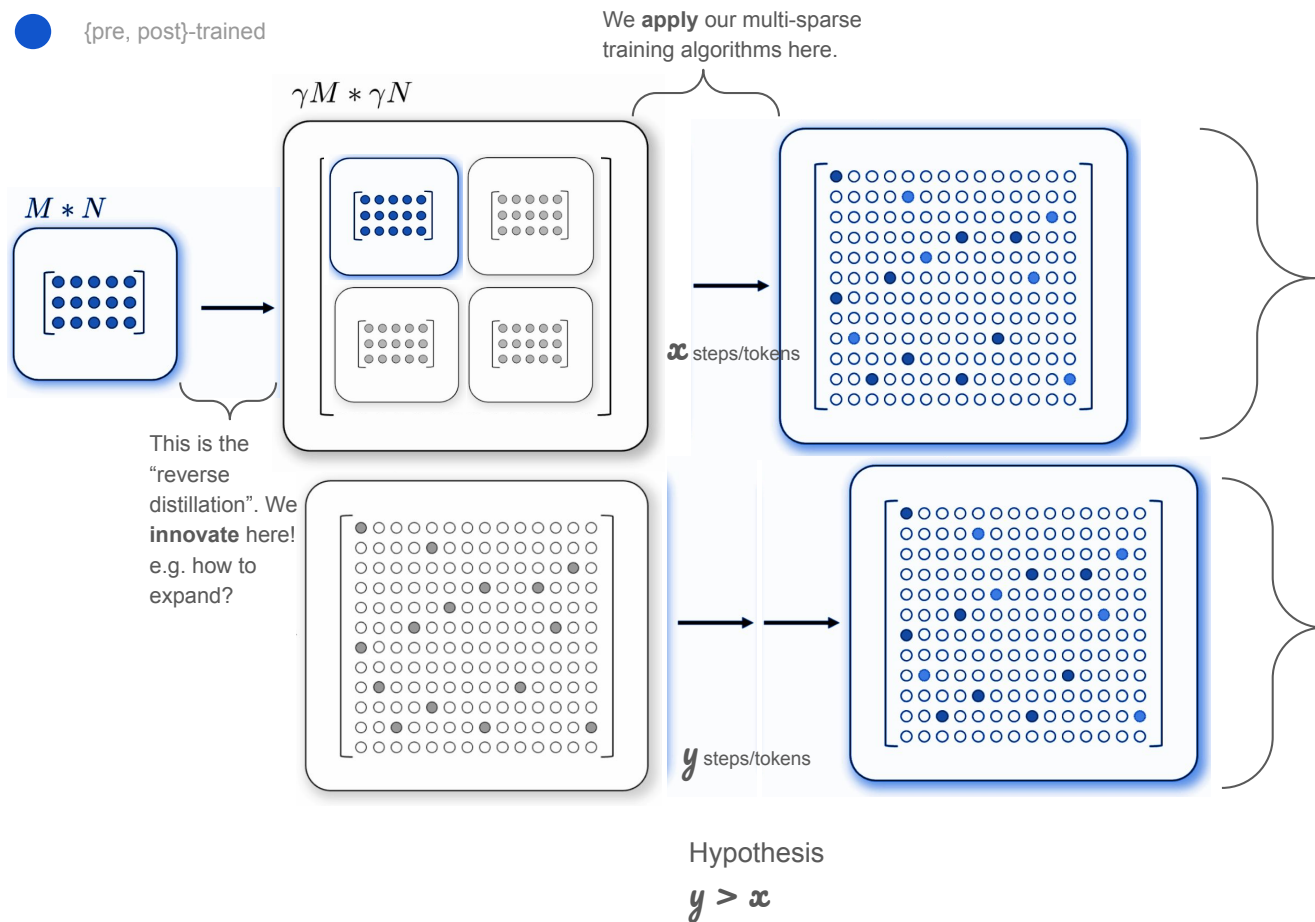
- Start at a better initialization?
Transfer learning
- Train more efficiently?
Continual learning



How can we use a pretrained model, but leverage higher dimensionality and multi-sparse training at the same time?

● untrained (*tabula rasa*)

● {pre, post}-trained



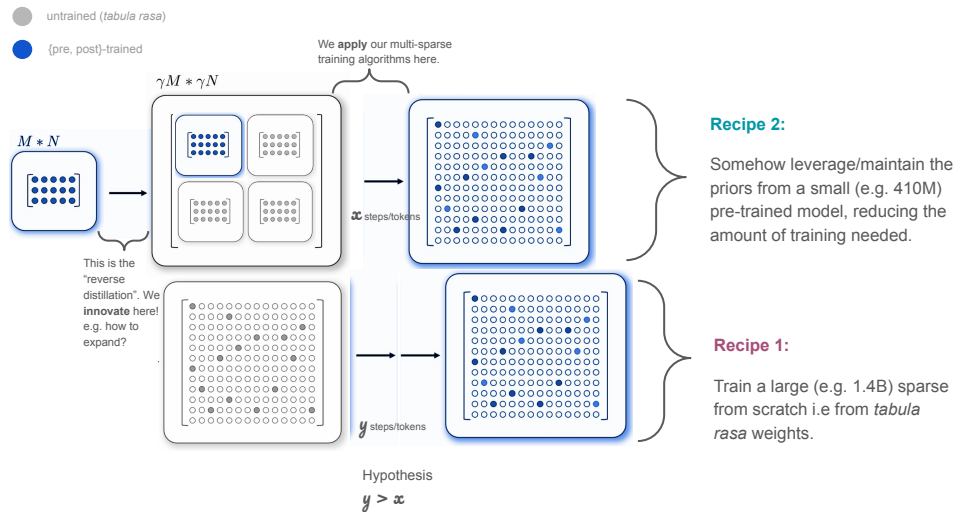
Recipe 2:

Somehow leverage/maintain the priors from a small (e.g. 410M) pre-trained model, reducing the amount of training needed.

Recipe 1:

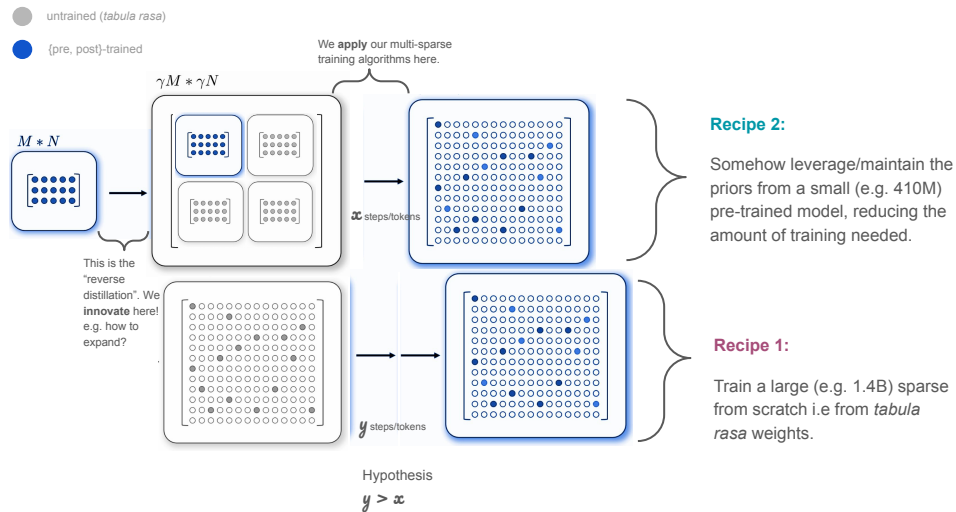
Train a large (e.g. 1.4B) sparse from scratch i.e from *tabula rasa* weights.

How can we use a pretrained model, but leverage higher dimensionality and multi-sparse training at the same time?



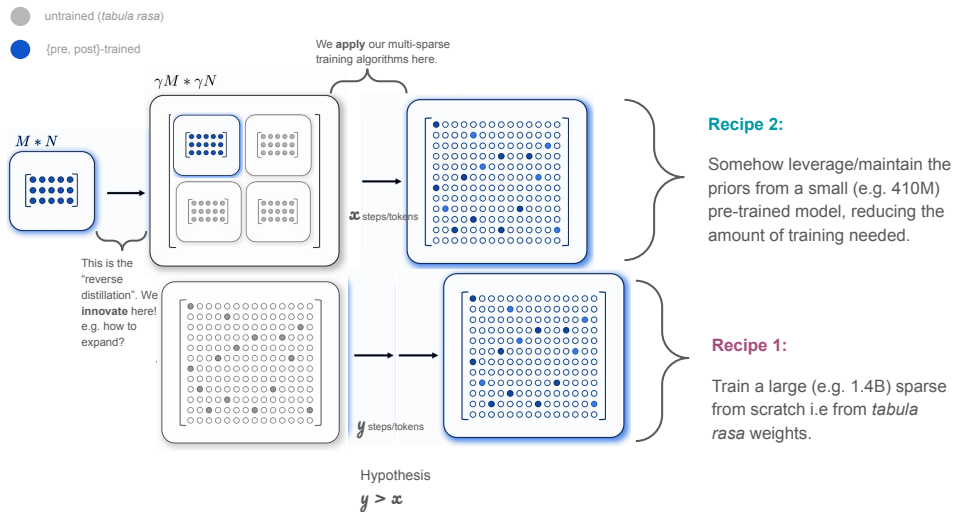
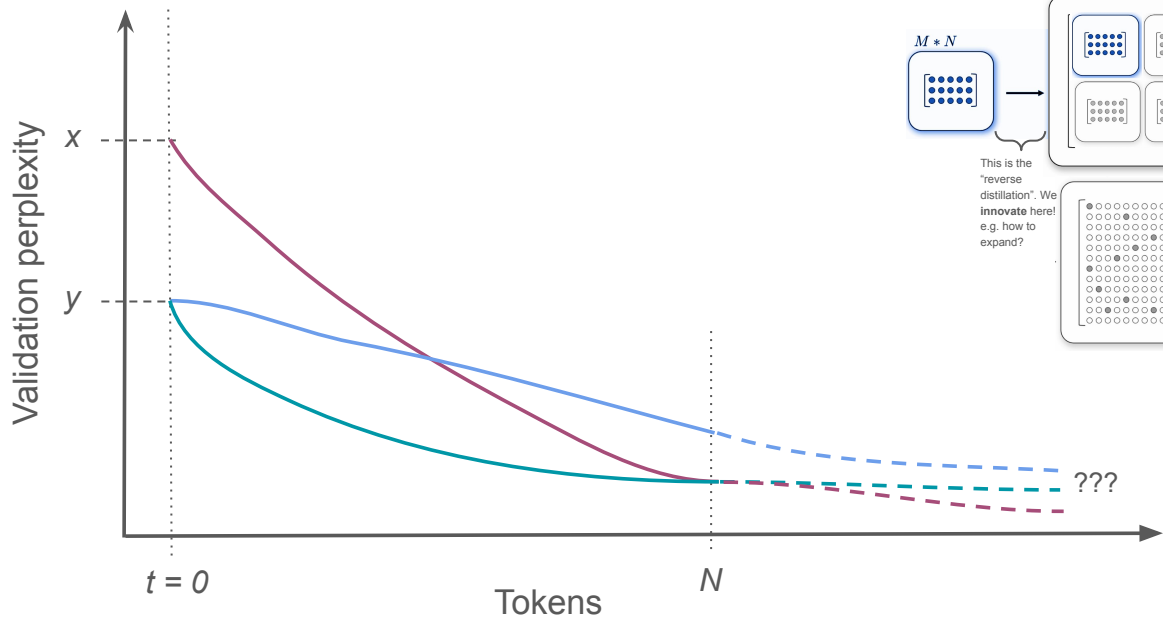
How can we use a pretrained model, but leverage higher dimensionality and multi-sparse training at the same time?

Hypothesis plot



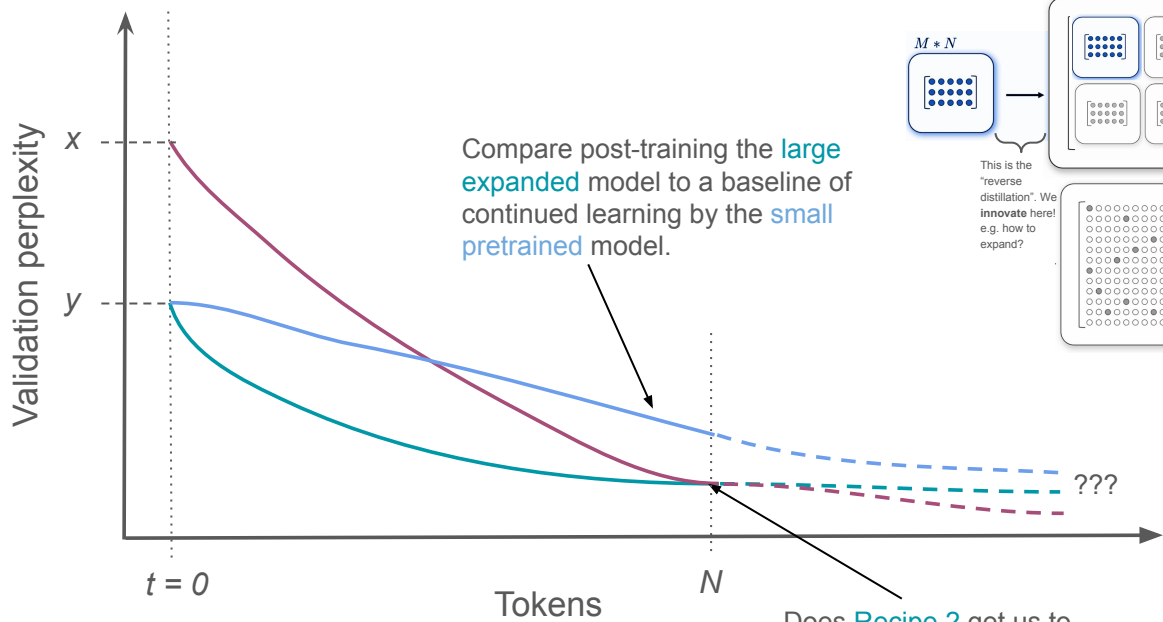
How can we use a pretrained model, but leverage higher dimensionality and multi-sparse training at the same time?

Hypothesis plot



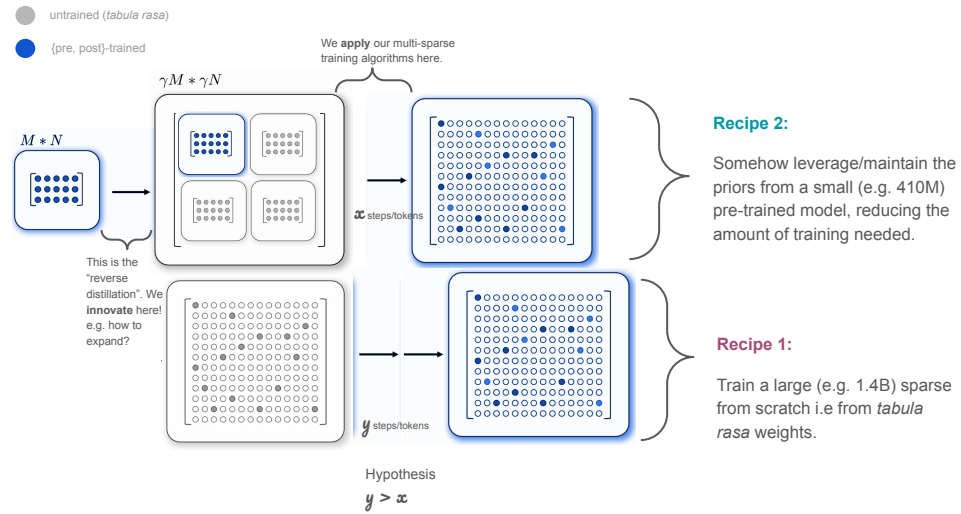
How can we use a pretrained model, but leverage higher dimensionality and multi-sparse training at the same time?

Hypothesis plot



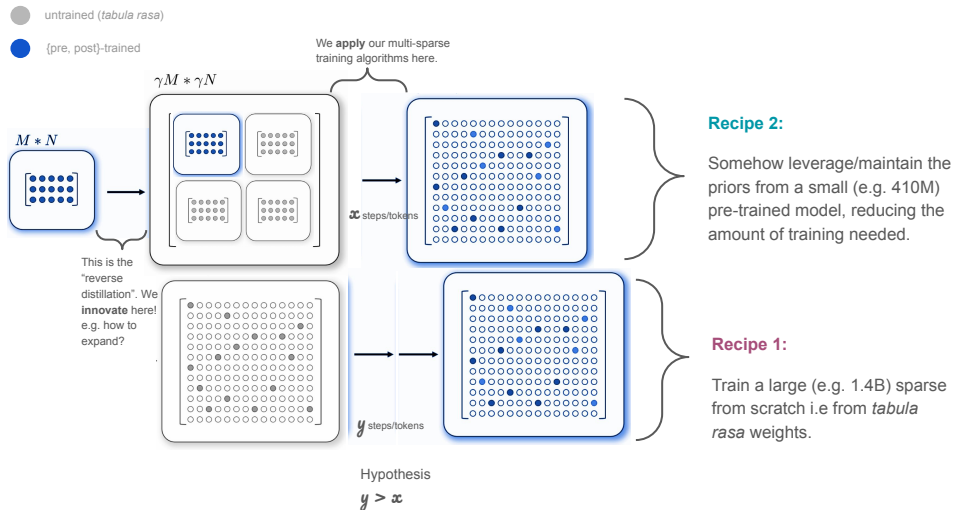
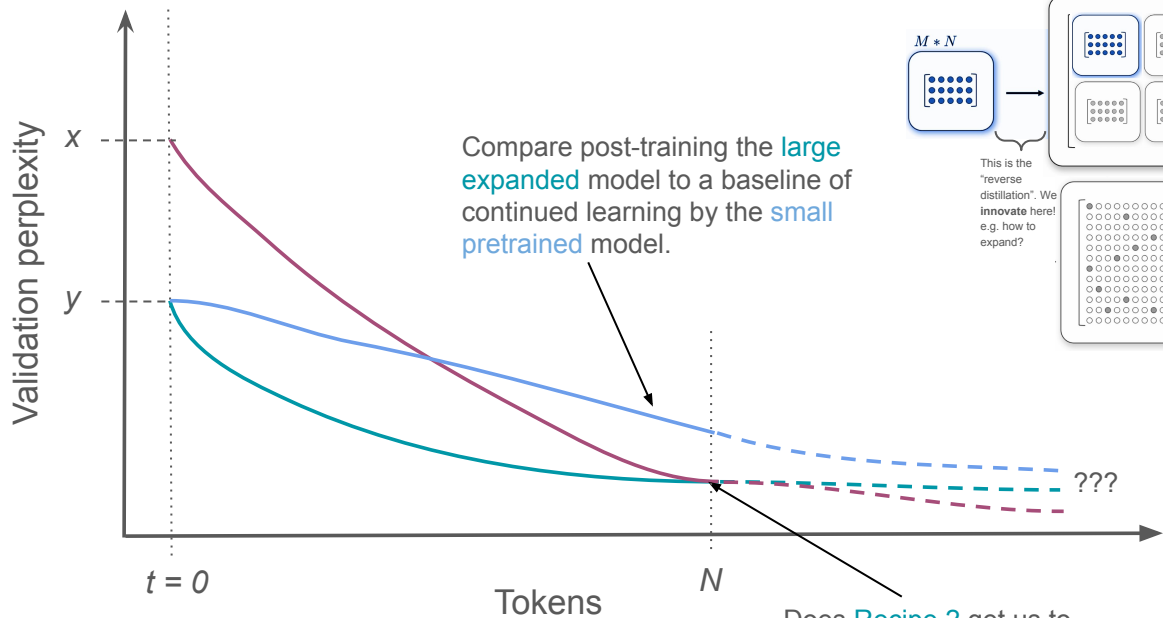
Compare post-training the **large expanded** model to a baseline of continued learning by the **small pretrained** model.

Does **Recipe 2** get us to the same loss faster than **Recipe 1**?



How can we use a pretrained model, but leverage higher dimensionality and multi-sparse training at the same time?

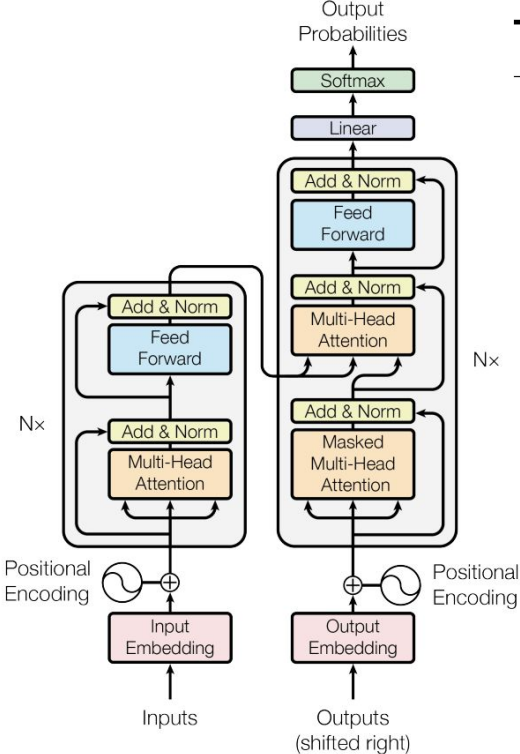
Hypothesis plot



x = starting loss of *tabula rasa* model
 y = starting loss of small pretrained and large expanded model
 N = equivalence point where Recipe 1 and Recipe 2 cross

Fair baseline:
 Size(*tabula rasa*) == Size(large expanded)

How to expand?



Attention Is All You Need

Ashish Vaswani*
Google Brain
avaswani@google.com

Noam Shazeer*
Google Brain
noam@google.com

Niki Parmar*
Google Research
nikip@google.com

Jakob Uszkoreit*
Google Research
usz@google.com

Llion Jones*
Google Research
llion@google.com

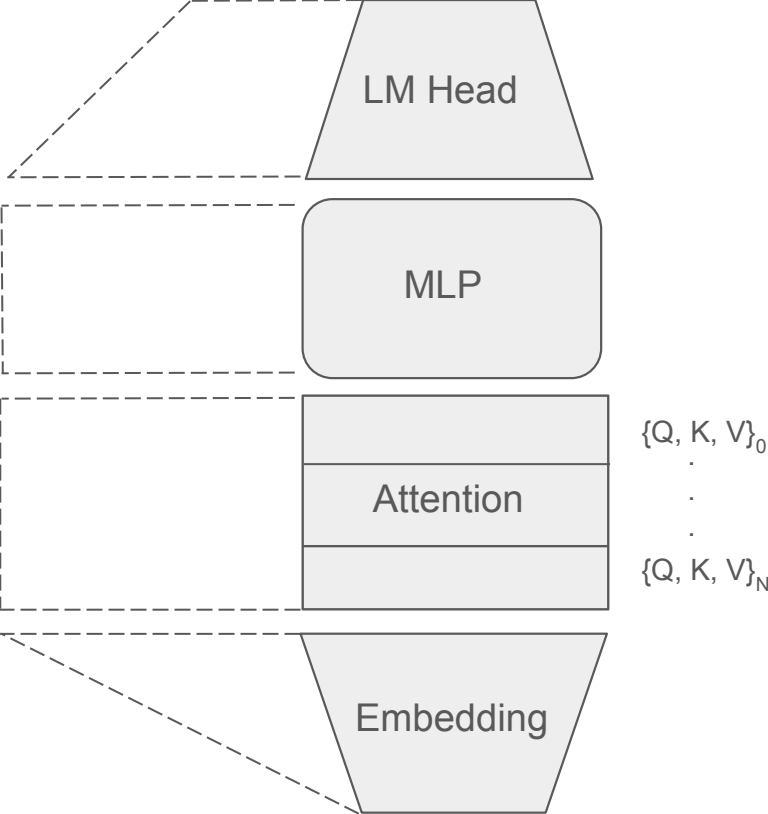
Aidan N. Gomez*[†]
University of Toronto
aidan@cs.toronto.edu

Lukas Kaiser*
Google Brain
lukaszkaizer@google.com

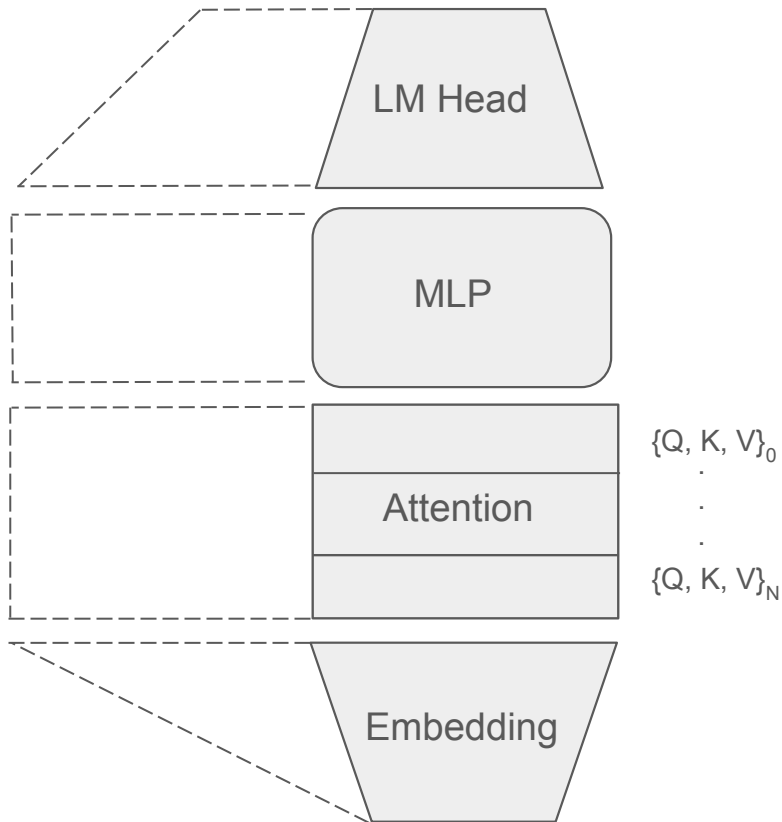
Ilya Polosukhin*[†]
ilya.polosukhin@gmail.com

Figure 1: The Transformer - model architecture.

How to expand?



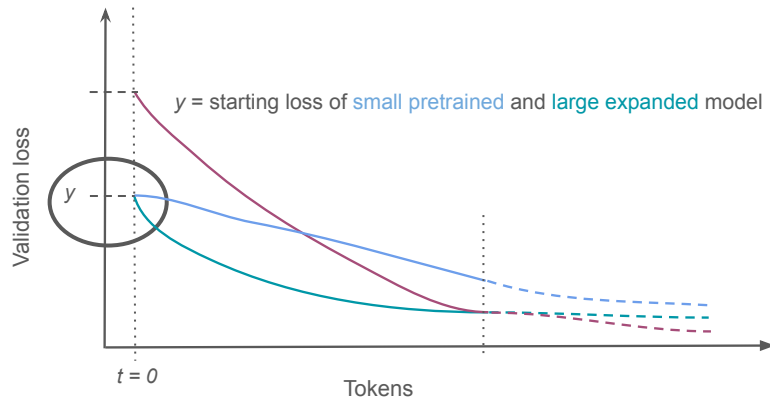
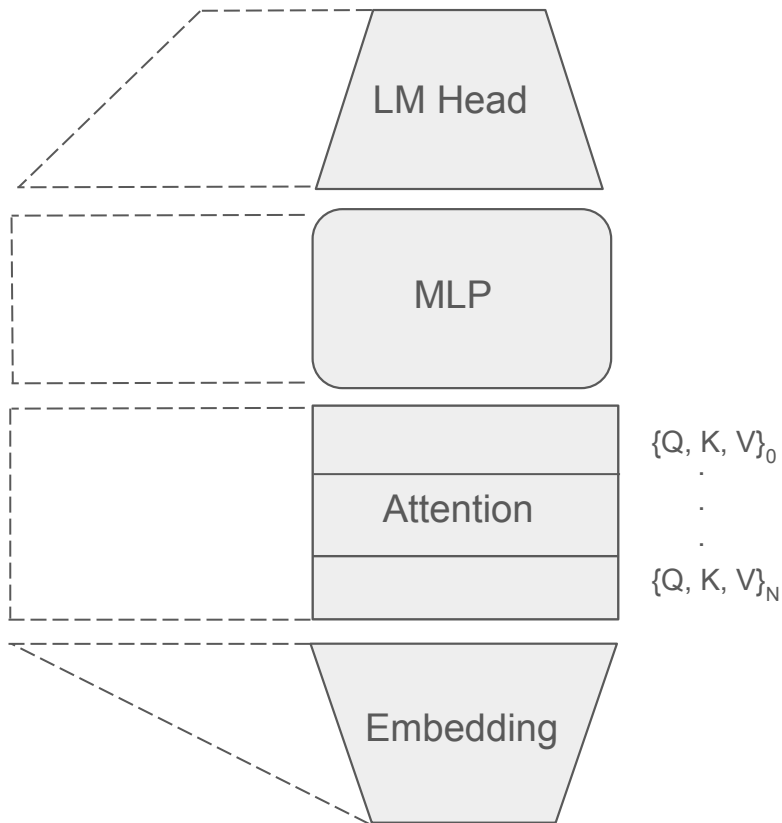
How to expand?



Fundamental **desiderata**:

- Input-output mapping must be preserved immediately after the expansion, before any post-training.
- In other words the expanded model must produce the same output logits as the source model given the same input tokens.

How to expand?

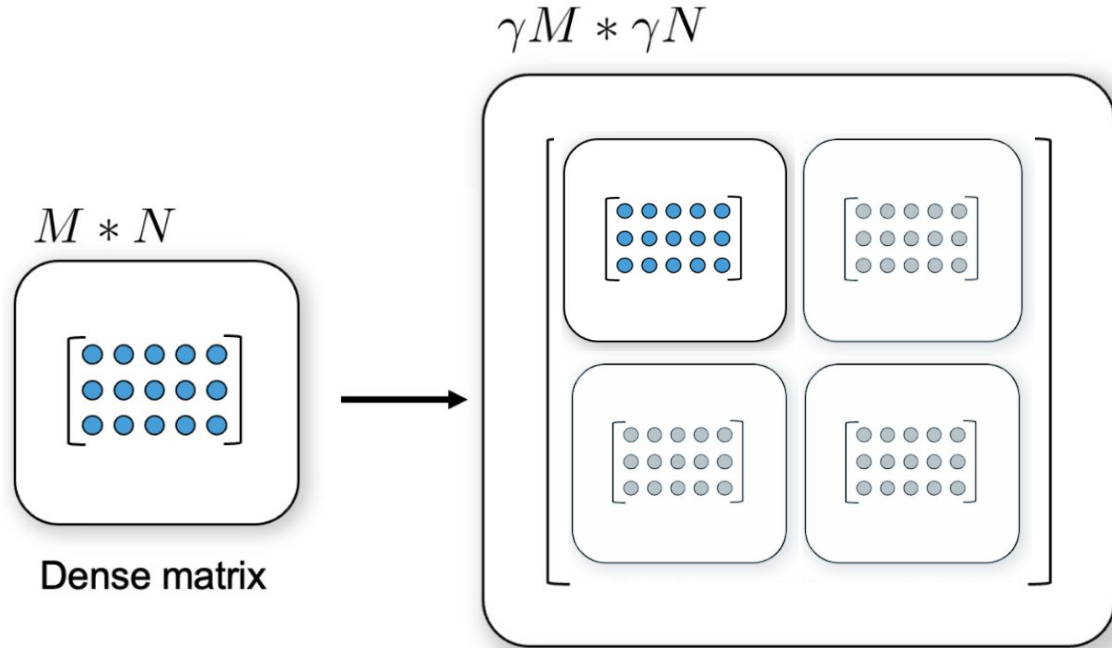


Fundamental **desiderata**:

- Input-output mapping must be preserved immediately after the expansion, before any post-training.
- In other words the expanded model must produce the same output logits as the source model given the same input tokens.

Expansion approaches

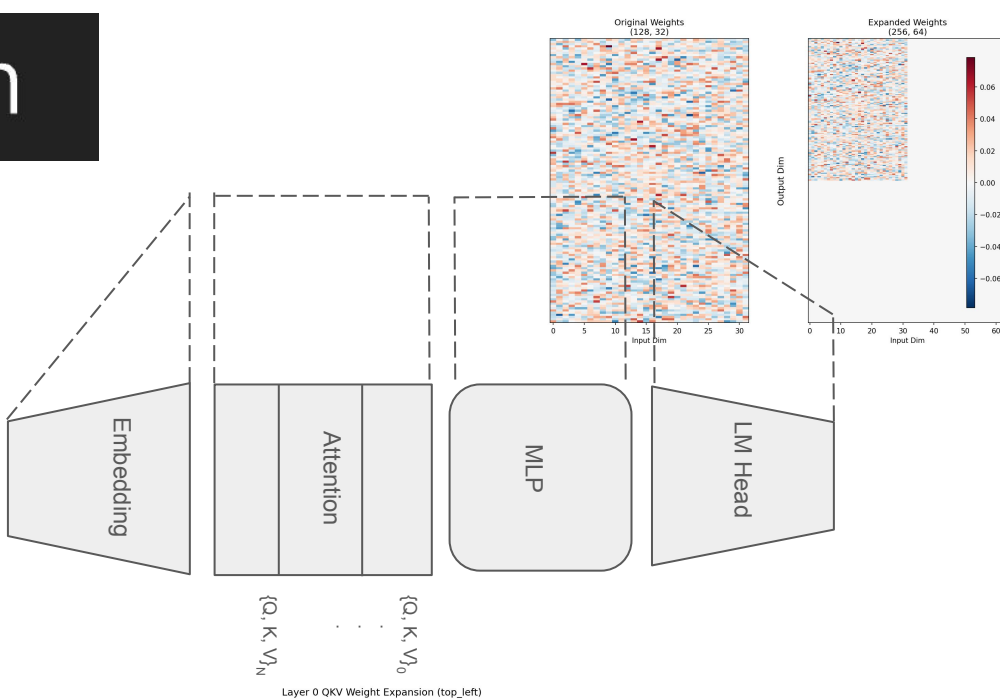
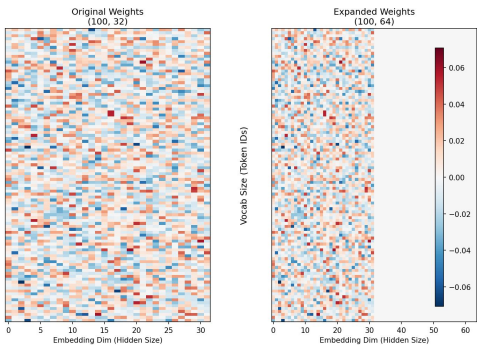
(1) Copy & zero-pad



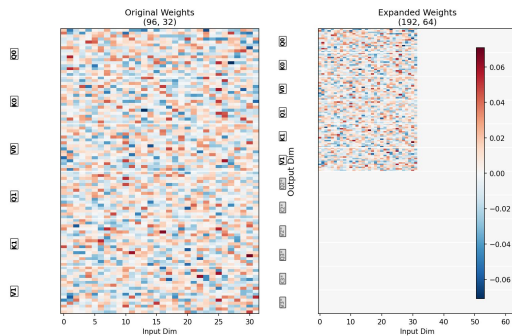


(1) Copy & zero-pad

Embedding Layer Weight Expansion (top_left)

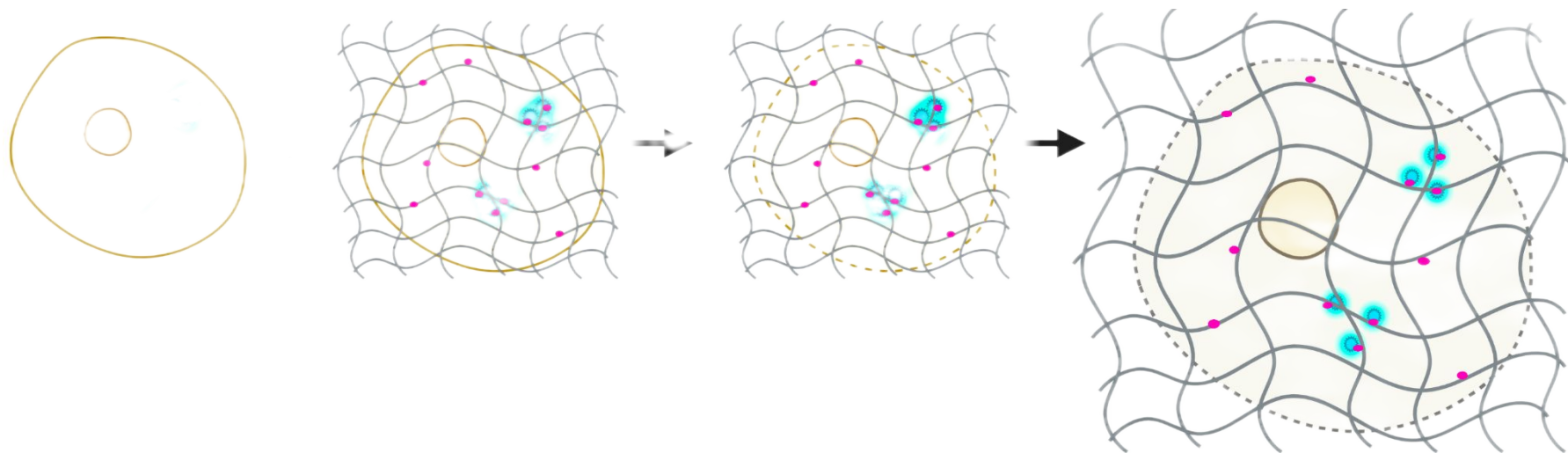


Layer 0 QKV Weight Expansion (top_left)



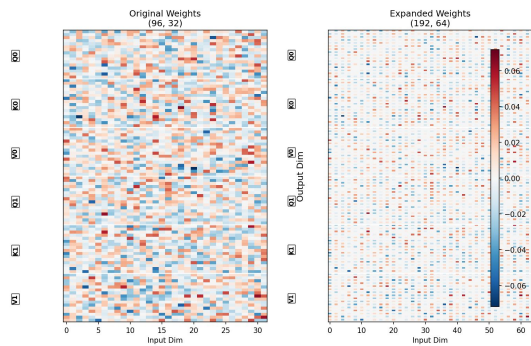
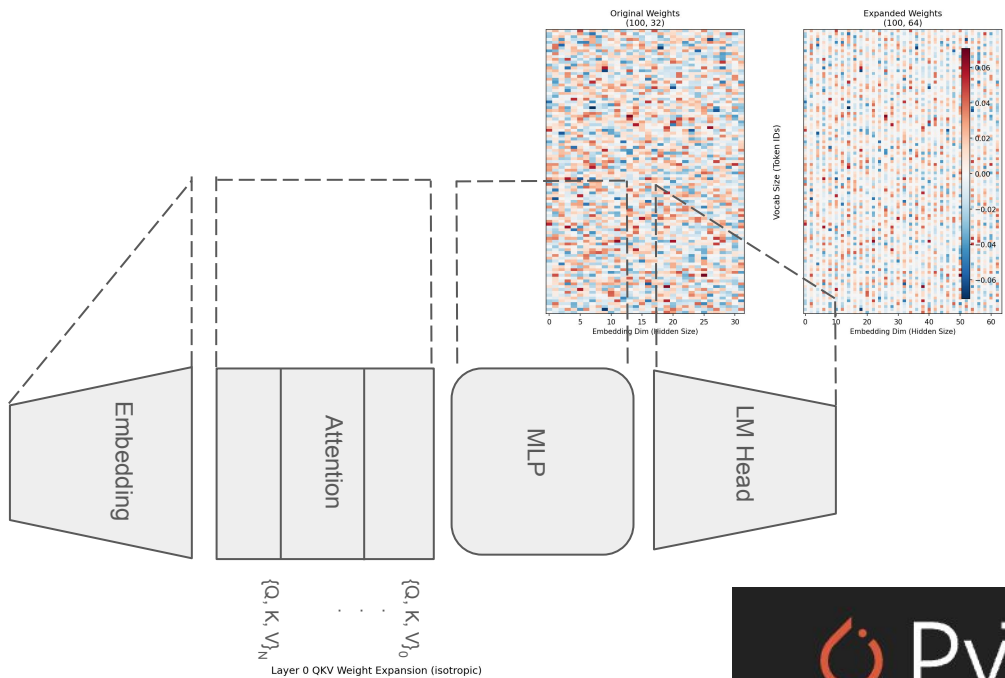
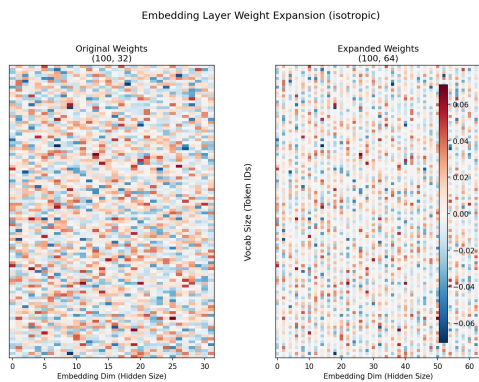
We call this expansion strategy 'top-left' because of how it prescribes for weights of intermediate module layers to be expanded .

(2) Isotropic expansion



PS: My PhD lab at MIT actually works on expansion microscopy for neuroscience.

(2) Isotropic expansion



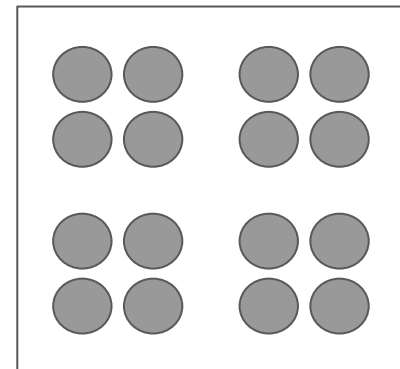
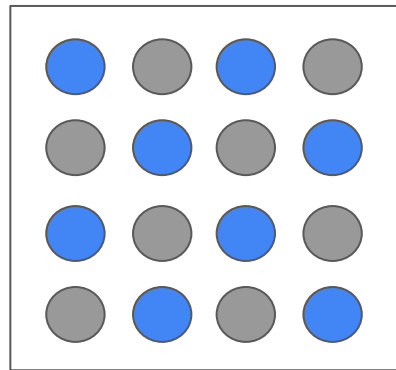
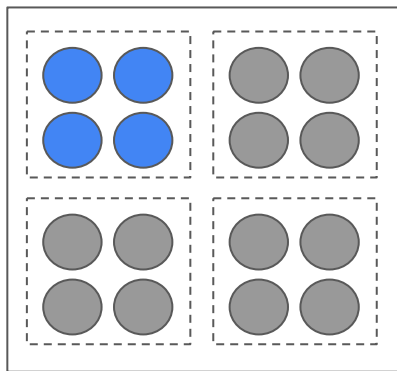
Head-wise interleaving: $\{Q, K, V\}_0, \dots, \{Q, K, V\}_N$
 * = New heads from expansion



Comparisons for experiments

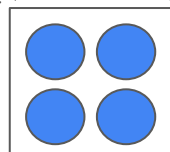
Top-left

Isotropic



large expanded

Expand (2x)



small pretrained

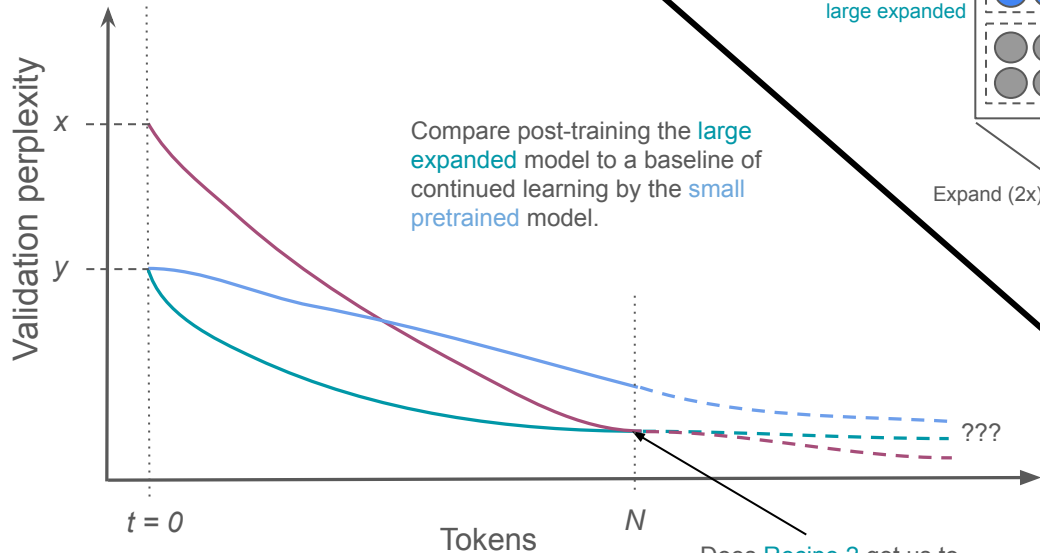
tabula rasa

Fair baseline:

Size(*tabula rasa*) == Size(large expanded)

Comparisons for experiments

Always keep this plot in mind.



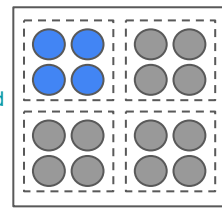
Compare post-training the **large expanded** model to a baseline of continued learning by the **small pretrained** model.

Does **Recipe 2** get us to same loss faster than **Recipe 1**?

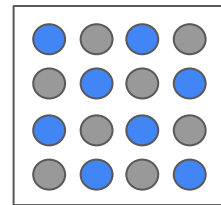
x = starting loss of *tabula rasa* model
y = starting loss of **small pretrained** and **large expanded** model
N = equivalence point where **Recipe 1** and **Recipe 2** cross

Top-left

large expanded



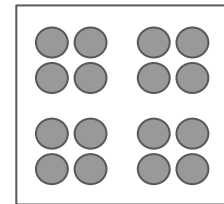
Isotropic



Expand (2x)



small pretrained



tabula rasa

Fair baseline:

Size(*tabula rasa*) == Size(**large expanded**)

Preliminary result

total_params

large_tabularasa_random_4249/70m_deduped



large_pretrained_expanded_topleft_4245/70m_deduped



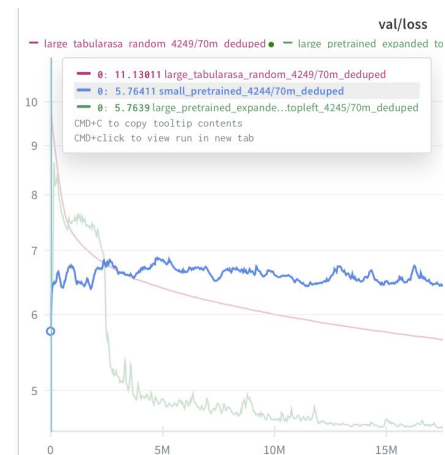
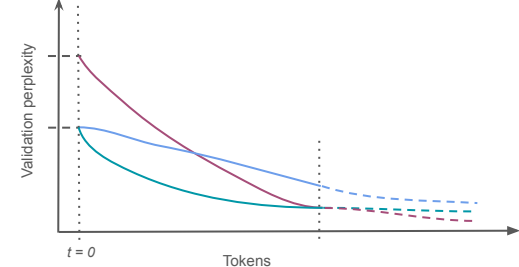
small_pretrained_4244/70m_deduped



small pretrained : 70M params; init.val.loss 5.76

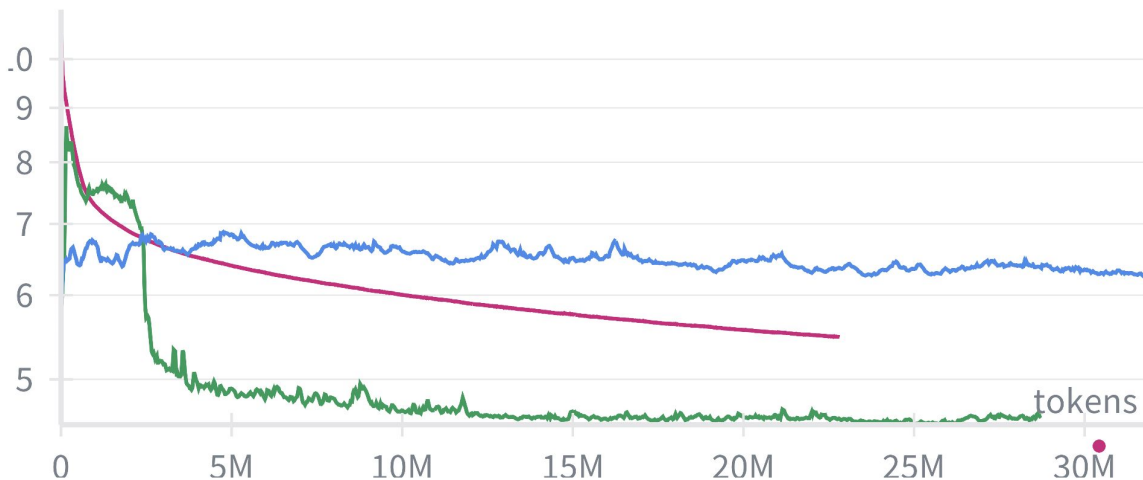
large expanded : 2x width expansion -> 180M; init.val.loss 5.76

tabula rasa : 180M params; init val.loss 11.13



val/loss

- large_tabularasa_random_4249/70m_deduped ● — large_pretrained_expanded_topleft_4245/70m_deduped ●
- small_pretrained_4244/70m_deduped



But how will our results scale with model size?

Pythia Scaling Suite

updated Feb 26

Pythia is the first LLM suite designed specifically to enable scientific research on LLMs. To learn more see <https://github.com/EleutherAI/pythia>

Pythia: A Suite for Analyzing Large Language Models Across Training and Scaling

📄 Paper - 2304.01373 · Published Apr 3, 2023 · 4 · 9

Note The paper introduced the Pythia model suite

EleutherAI/pythia-14m

📄 Text Generation · 0.08 · Updated Jul 26, 2023 · 46.4k · 23

Note This model was trained after the paper was released.

EleutherAI/pythia-70m

0.1B · Updated Nov 21, 2023 · 152k · 73

Note This model was trained after the paper was released.

EleutherAI/pythia-160m

Text Generation · 0.2B · Updated Jul 9, 2023 · 47.4k · 32

EleutherAI/pythia-410m

Text Generation · 0.5B · Updated Jul 9, 2023 · 113k · 32

EleutherAI/pythia-1b

Text Generation · 1B · Updated Jul 9, 2023 · 17.1k · 42

EleutherAI/pythia-1.4b

Text Generation · 2B · Updated Jul 9, 2023 · 23.9k · 25

EleutherAI/pythia-2.8b

Text Generation · 3B · Updated Jun 8, 2023 · 17.9k · 31

EleutherAI/pythia-6.9b

Text Generation · 7B · Updated Mar 10 · 15.2k · 56

EleutherAI/pythia-12b

Text Generation · 12B · Updated Jul 9, 2024 · 6.67k · 139

EleutherAI/pythia-70m-deduped

Text Generation · 0.1B · Updated Jul 9, 2023 · 362k · 25

EleutherAI/pythia-160m-deduped

Text Generation · 0.2B · Updated Jul 9, 2023 · 148k · 3

EleutherAI/pythia-410m-deduped

Text Generation · 0.5B · Updated Jul 9, 2023 · 15.8k · 20

EleutherAI/pythia-1b-deduped

Text Generation · 1B · Updated Jul 10, 2023 · 8.08k · 19

EleutherAI/pythia-1.4b-deduped

Text Generation · Updated Jun 8, 2023 · 20.7k · 20

Pythia: A Suite for Analyzing Large Language Models Across Training and Scaling

Stella Biderman^{*1,2} Hailey Schoelkopf^{*1,3} Quentin Anthony¹ Herbie Bradley^{1,4} Kyle O'Brien¹
Eric Hallahan¹ Mohammad Aflah Khan⁵ Shivanshu Purohit^{6,1} USVSN Sai Prashanth¹ Edward Raff²
Aviya Skowron¹ Lintang Sutawika^{1,7} Oskar van der Wal⁸

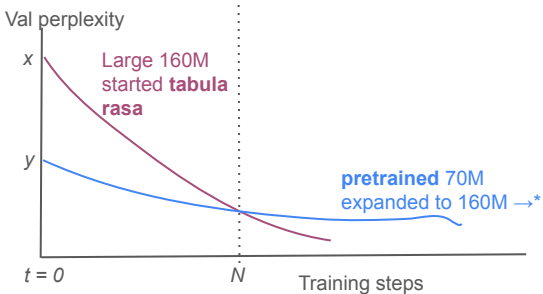
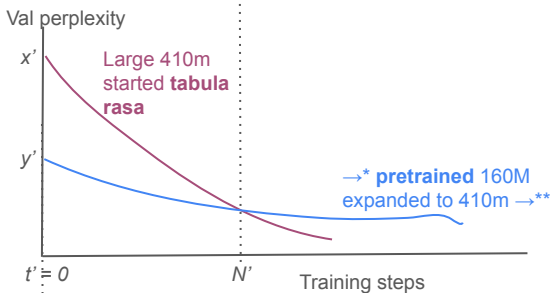
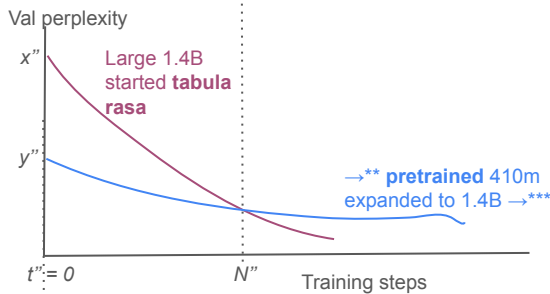
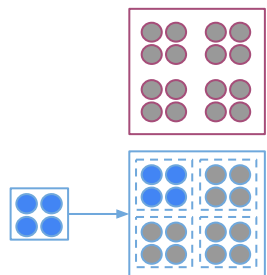
Abstract

How do large language models (LLMs) develop and evolve over the course of training? How do these patterns change as models scale? To answer these questions, we introduce *Pythia*, a suite of 16 LLMs all trained on public data seen in the exact same order and ranging in size from 70M to 12B parameters. We provide public access to 154 checkpoints for each one of the 16 models, alongside tools to download and reconstruct their exact training dataloaders for further study. We intend *Pythia* to facilitate research in many areas, and we present several case studies including novel results in memorization, term frequency effects on few-shot performance, and reducing gender bias. We demonstrate that this highly controlled setup can be used to yield novel insights toward LLMs and their training dynamics. Trained models, analysis code, training code, and training data can be found at <https://github.com/EleutherAI/pythia>.



Potentially novel model scaling recipe?

etcetera ... until final target model size (e.g. 12B) achieved.



Preliminary observations

$$x'' > x' \gg x$$

- larger random models start of worse, but they have better slopes so end up with better loss

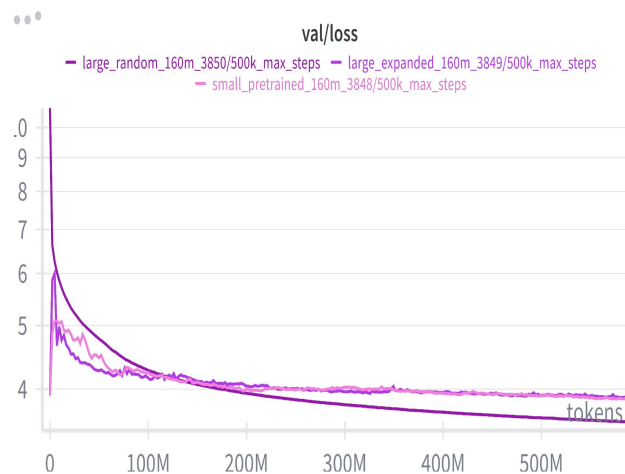
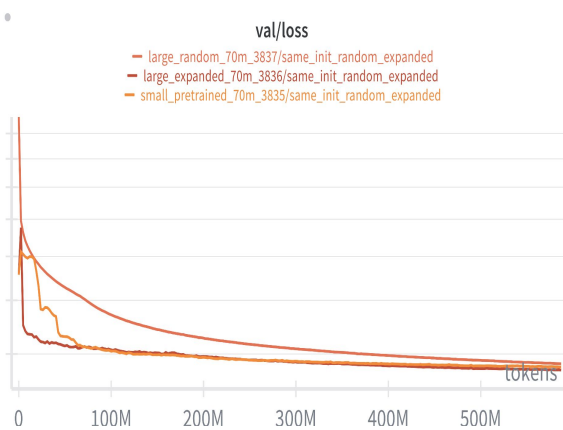
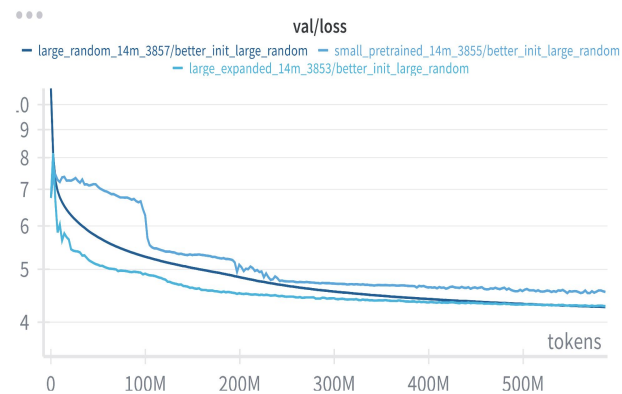
$$y'' < y' < y$$

- classic scaling law result that larger models achieve lower loss

$$N'' < N' < N$$

- for larger models, the equivalence point shifts to earlier.

Preliminary results



What I learned from Project 1

Working collaboratively with a small team.

Sharing resources. Contiguous integration testing.

Using and building upon existing infrastructure and codebases.

Lots of industry machine learning.

- Actually learned about and how to use git. Not quite mastered it, but who has?
- Importance of unit testing and continuous integration (CI) testing (CircleCI)
- Run experiments on cloud computing cloud computing clusters (Oracle)
- Track experiment progress with logging services (WandB)
- How to project plan and best practices like Agile for progress monitoring with (ShortCut)
- APIs for many close and open-source language models (OpenAI, Claude, Hugging Face...)

