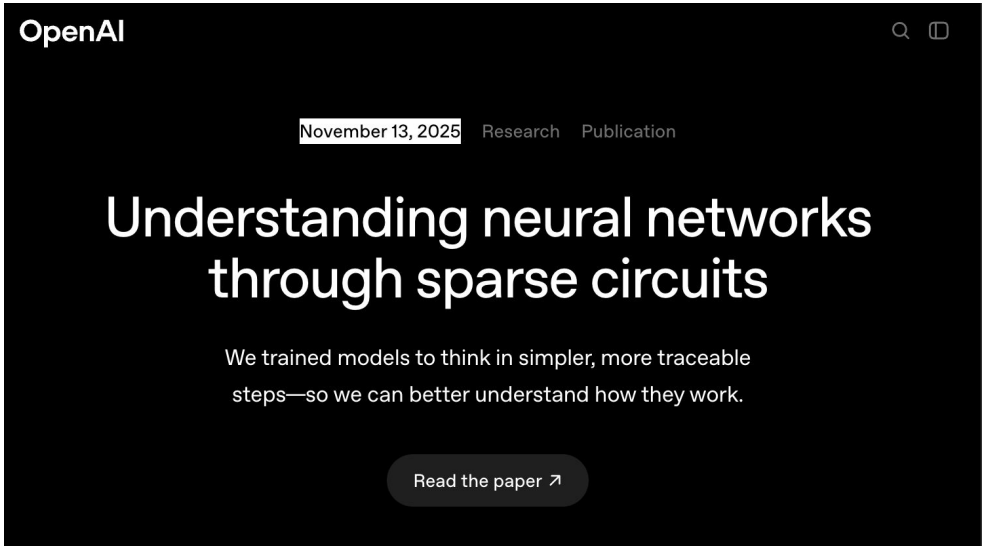
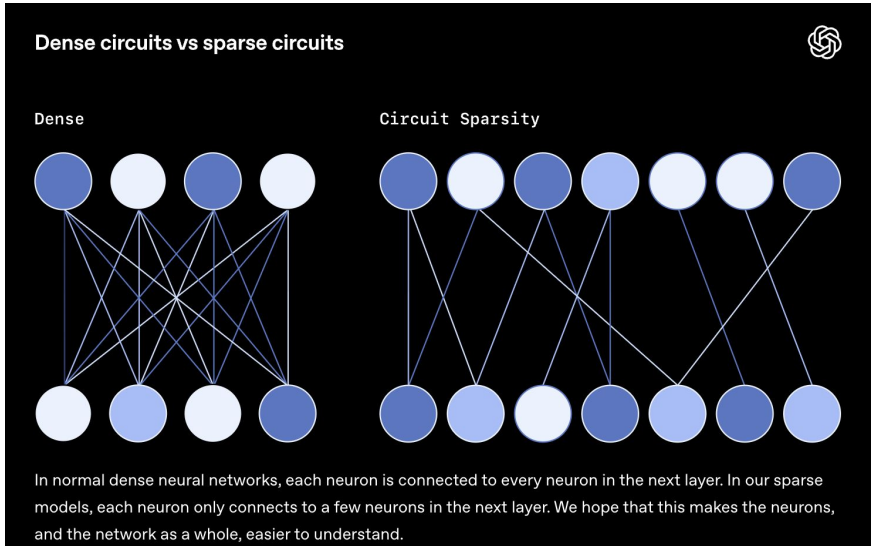
Project 1: Growing sparse large language models

Work done at my last internship. 

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!





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 Frantar [1] Dan Alistarh [1 2]

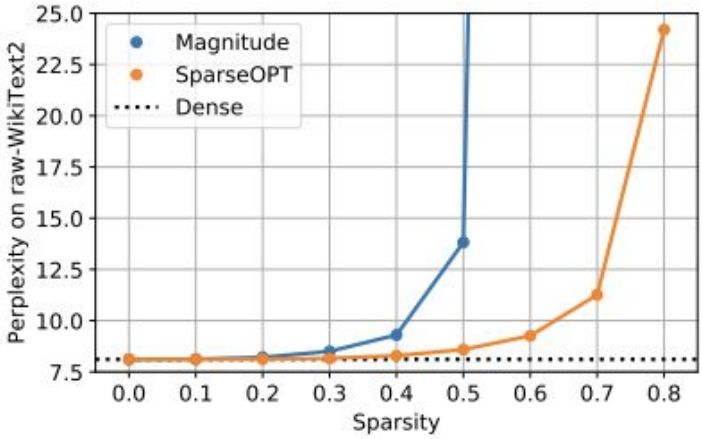
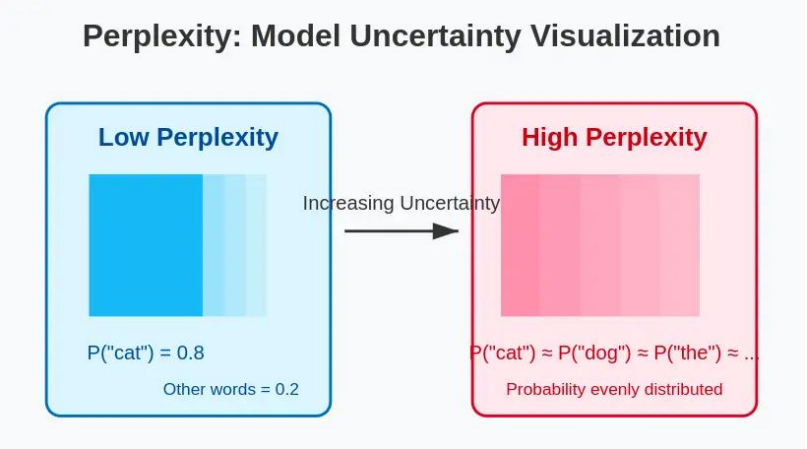
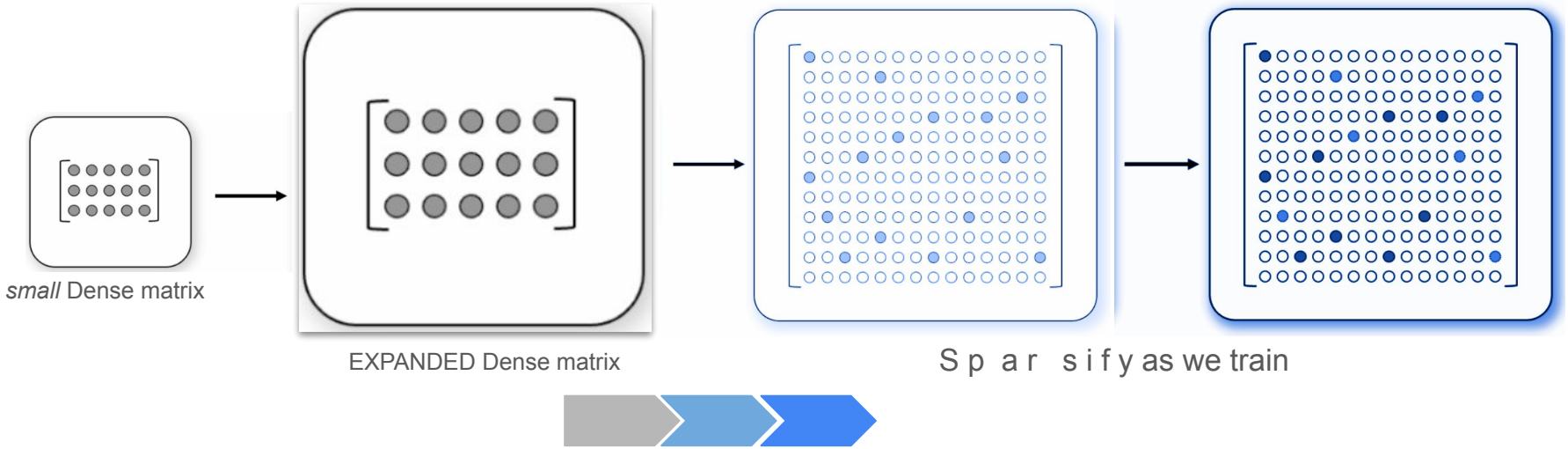


Figure 5. Uniform pruning BLOOM-176B.



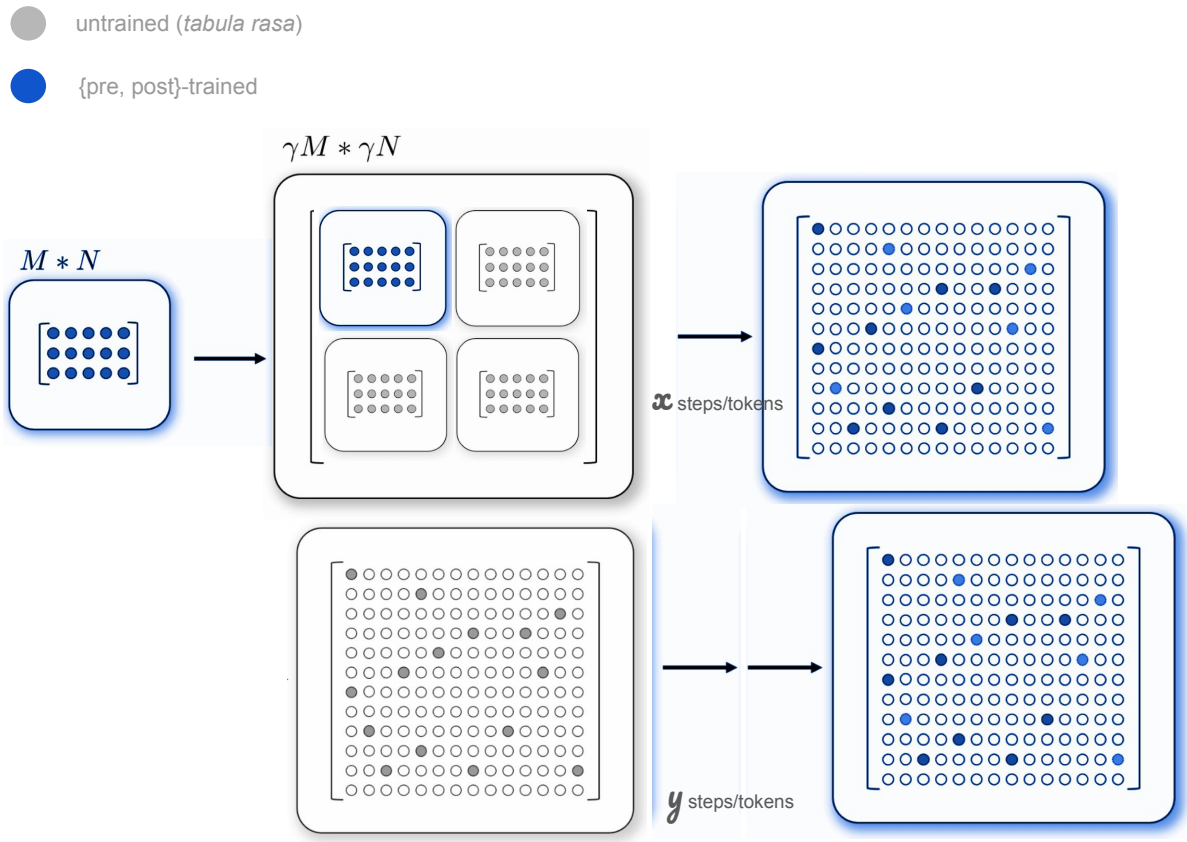
Another idea is to start from an untrained small dense model, expanding the dimension and sparsifying as we train. But this is often less efficient and does not perform as well as just training a large model from scratch.



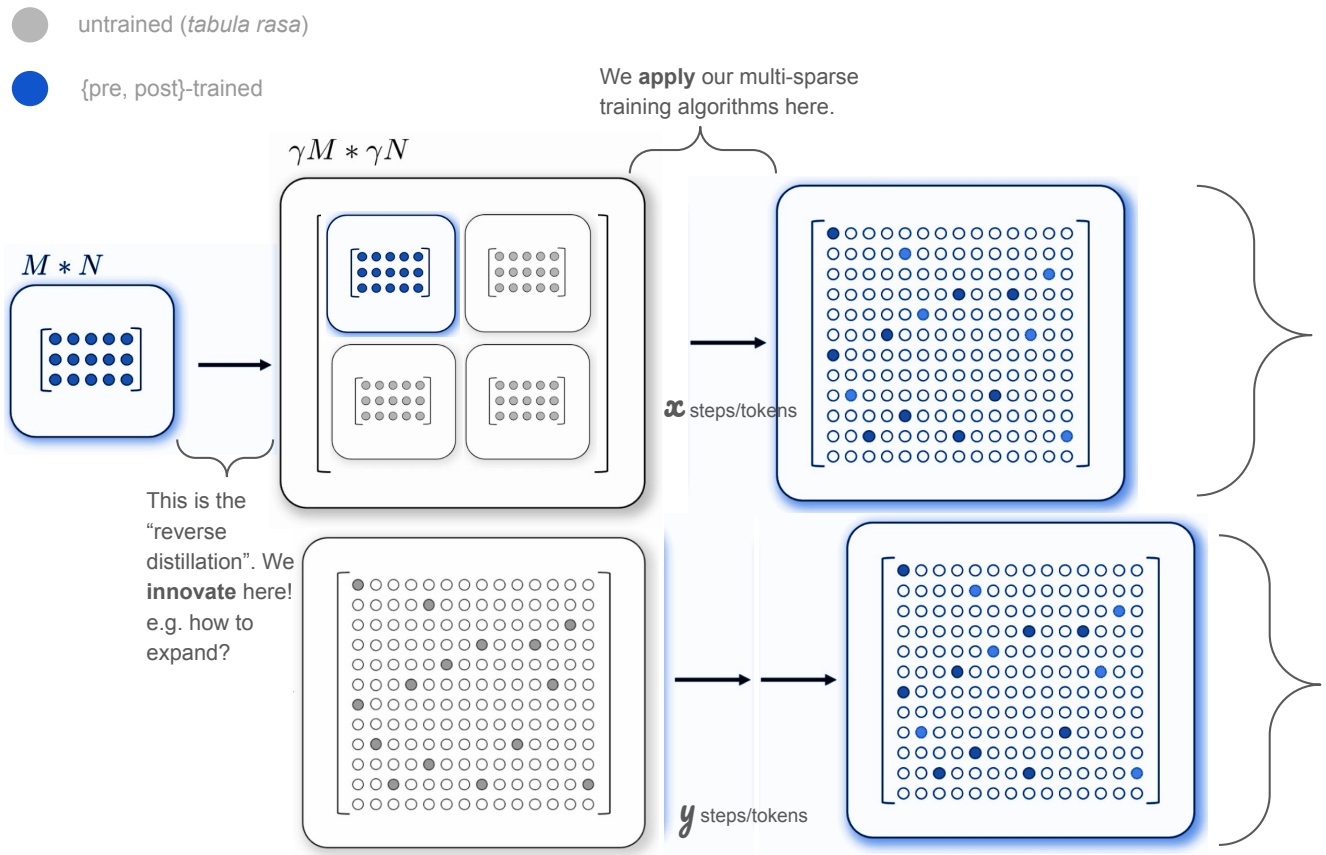
Can we “transplant” the weights of a **small pretrained** model into a **large untrained** one to cause that larger model to:

a. Start at a better <u>initialization</u>?
Transfer learning

b. <u>Train</u> more efficiently?
Continual learning



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



Recipe 2:

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

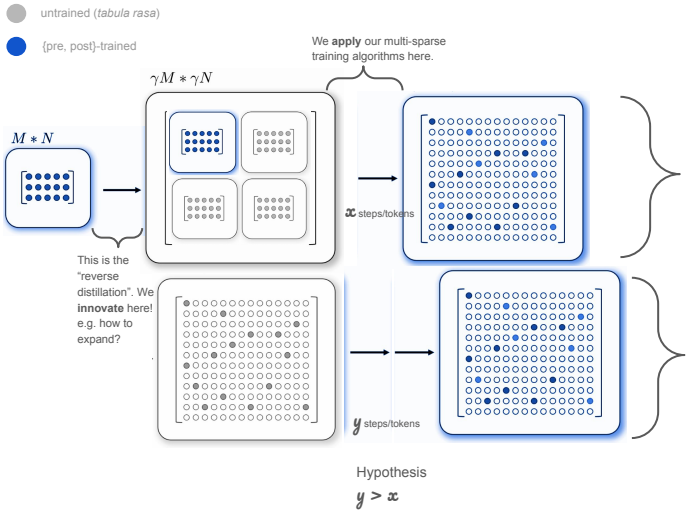
Recipe 1:

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

Hypothesis

$y > x$

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



Recipe 2:

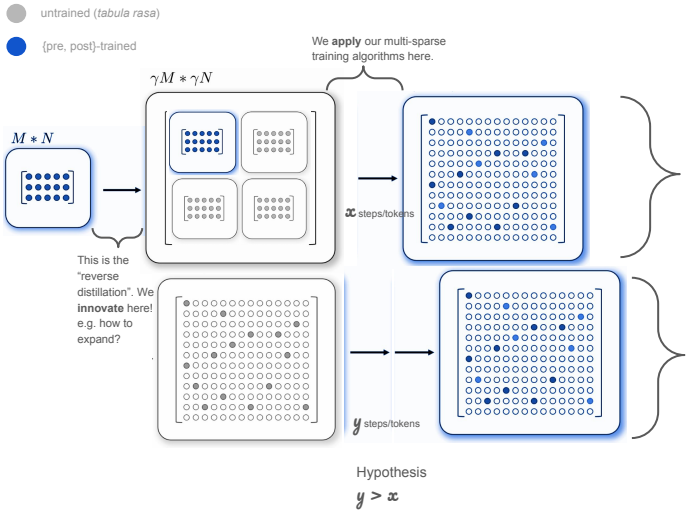
Somehow leverage/maintain the priors from a small (e.g. 410M) pre-trained model, reducing the amount of training needed.

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?

Hypothesis *plot*



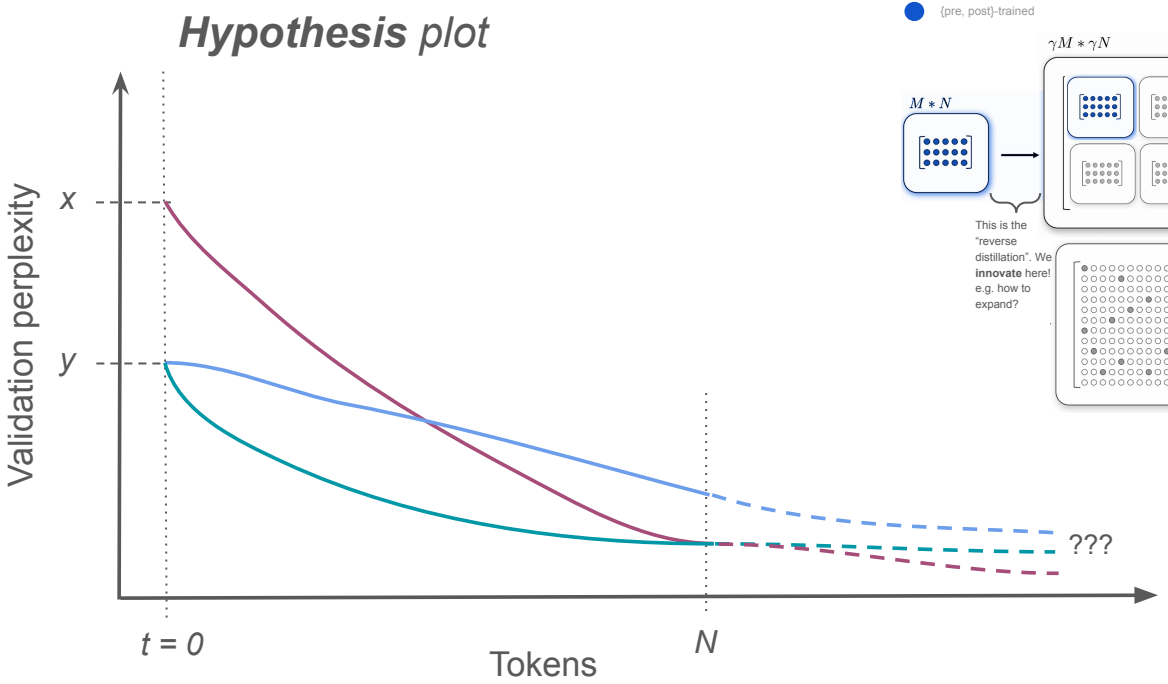
Recipe 2:

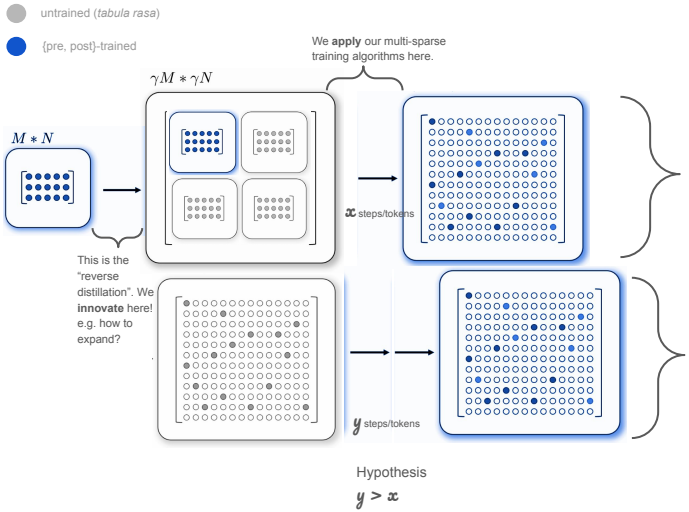
Somehow leverage/maintain the priors from a small (e.g. 410M) pre-trained model, reducing the amount of training needed.

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?





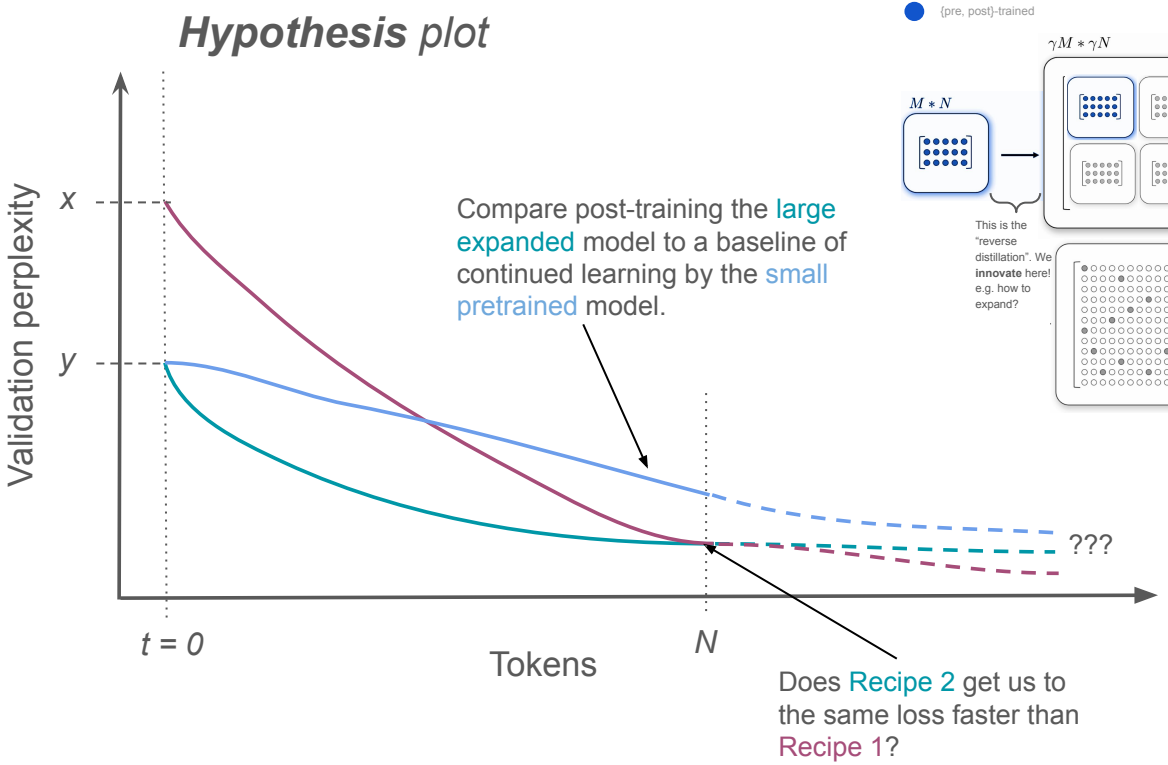
Recipe 2:

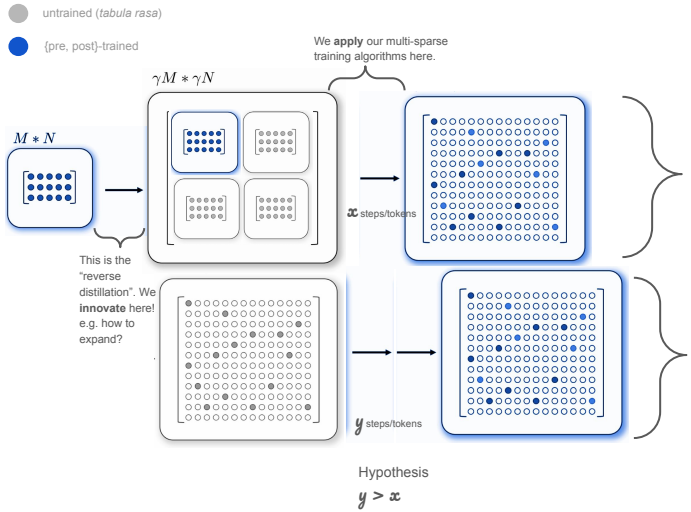
Somehow leverage/maintain the priors from a small (e.g. 410M) pre-trained model, reducing the amount of training needed.

Recipe 1:

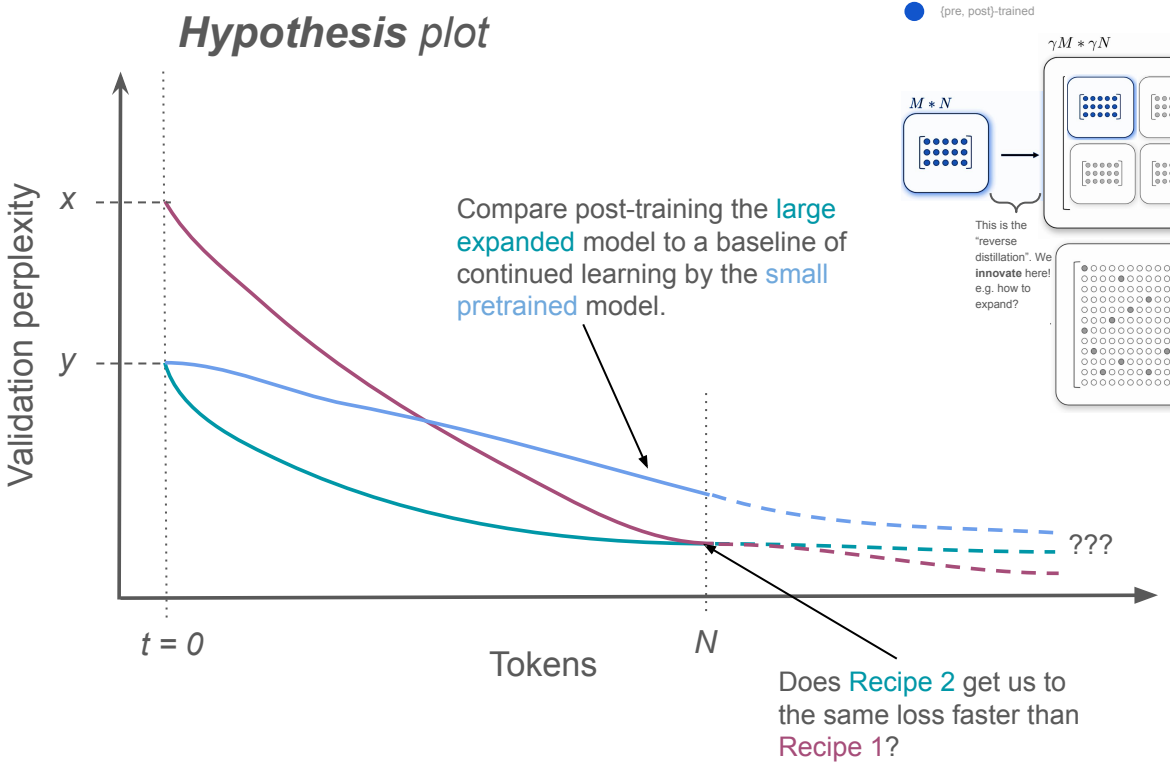
Train a large (e.g. 1.4B) sparse from scratch i.e from *tabula rasa* weights.

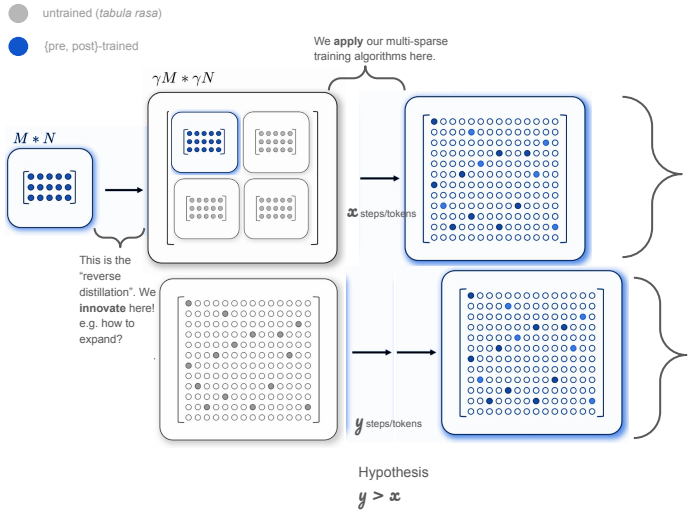
How can we use a pretrained model, but leverage higher dimensionality and multi-sparse training at the same time?





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





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?

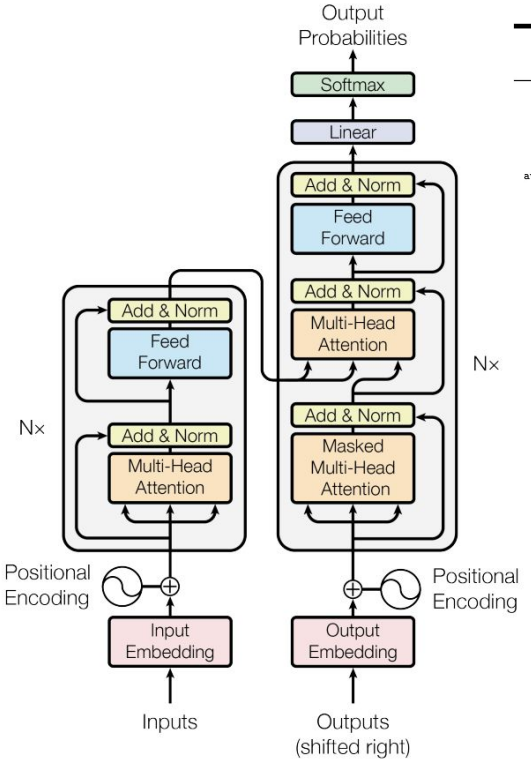
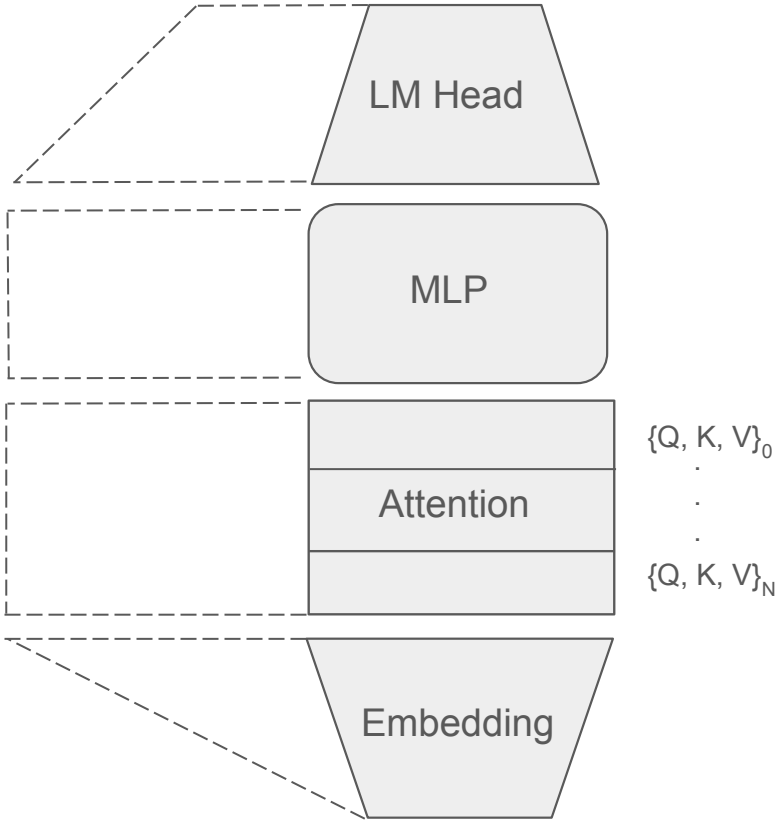


Figure 1: The Transformer - model architecture.

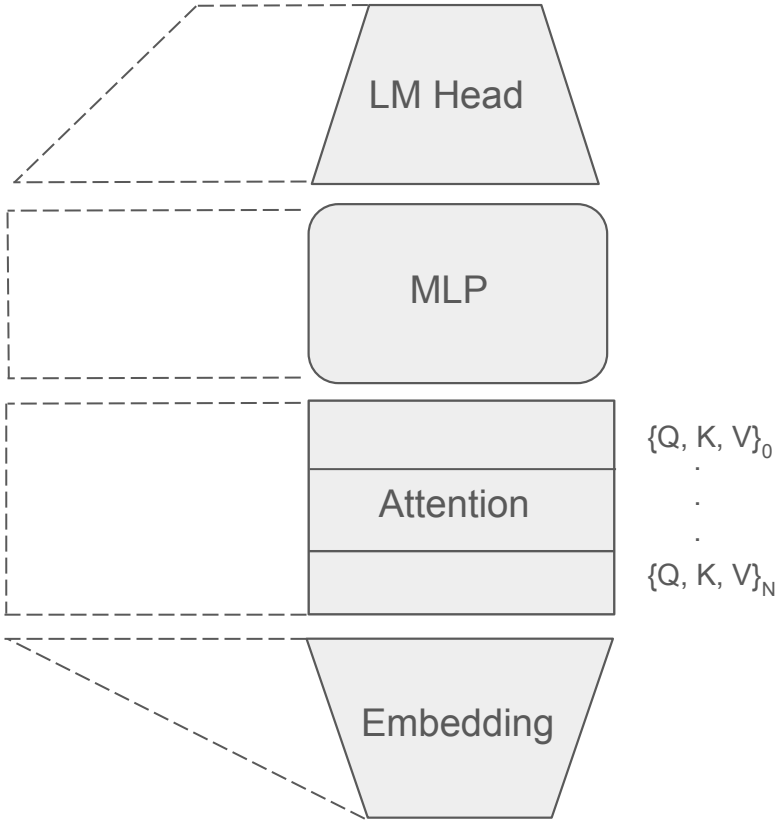
Attention Is All You Need

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
Łukasz Kaiser* Google Brain lukaszkaiser@google.com
Illia Polosukhin* ‡ illia.polosukhin@gmail.com

How to expand?



How to expand?

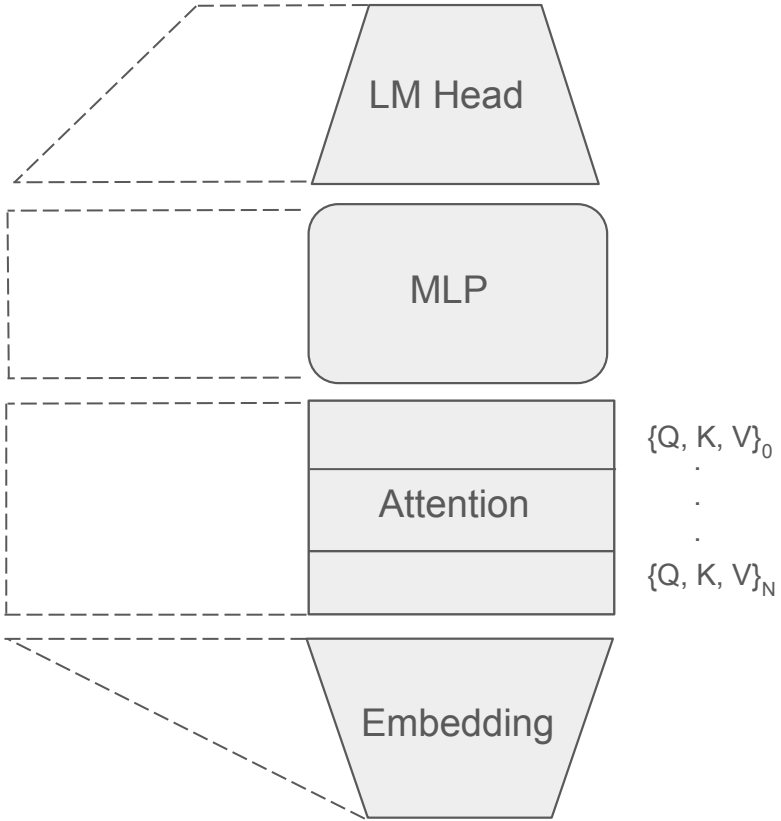


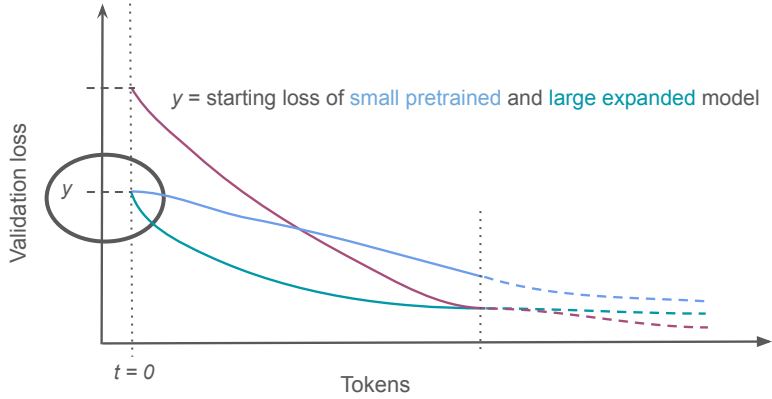
Fundamental **desiderata**:

- 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?





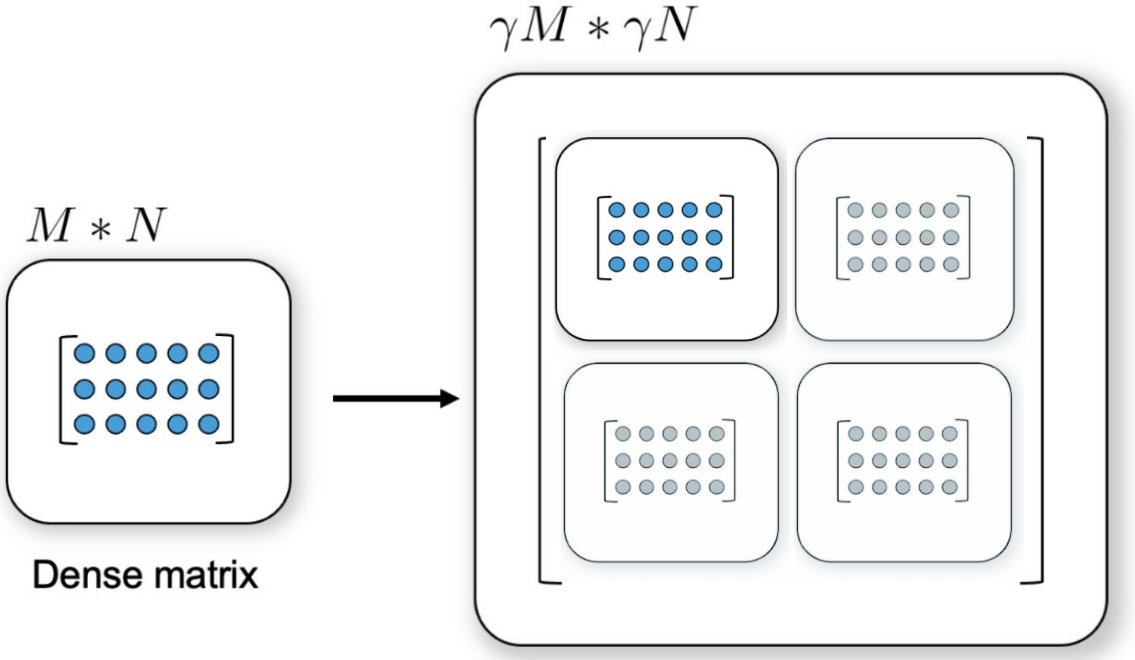
Fundamental **desiderata**:

- 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.

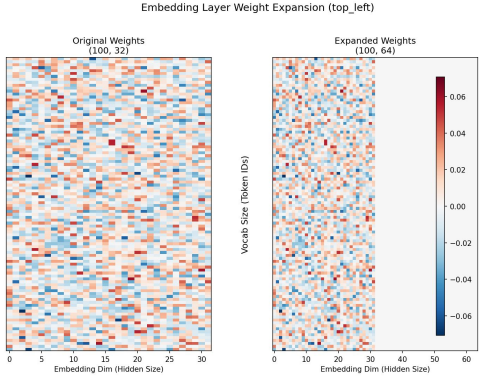
Expansion approaches

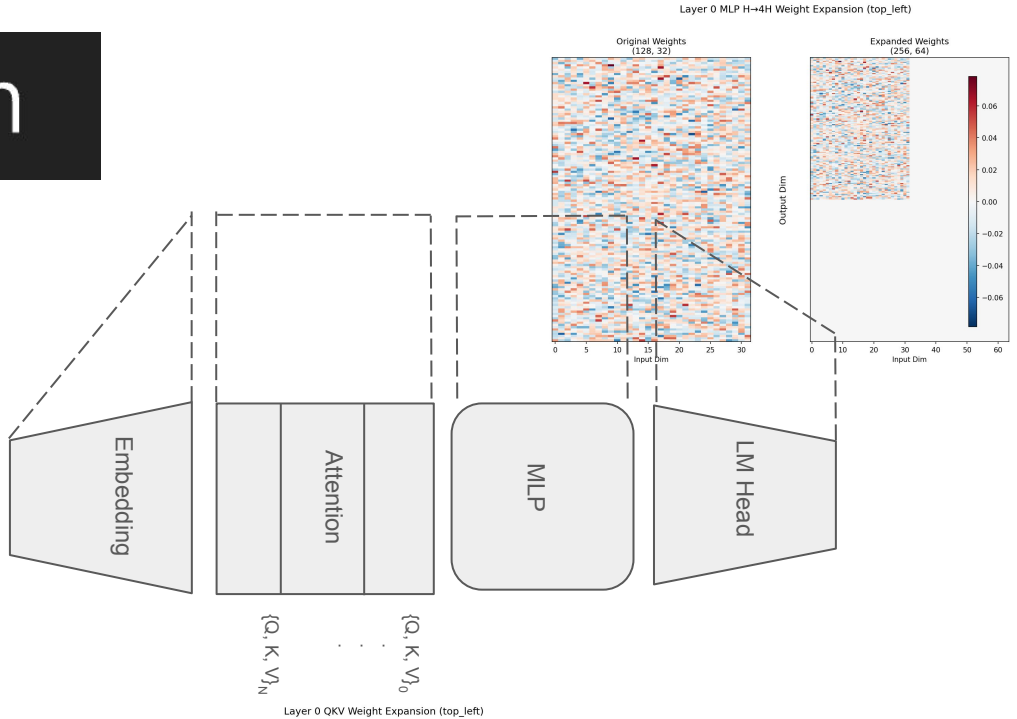
(1) Copy & zero-pad

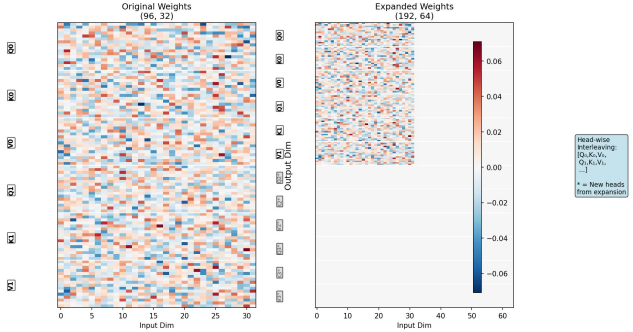


PyTorch

(1) Copy & zero-pad

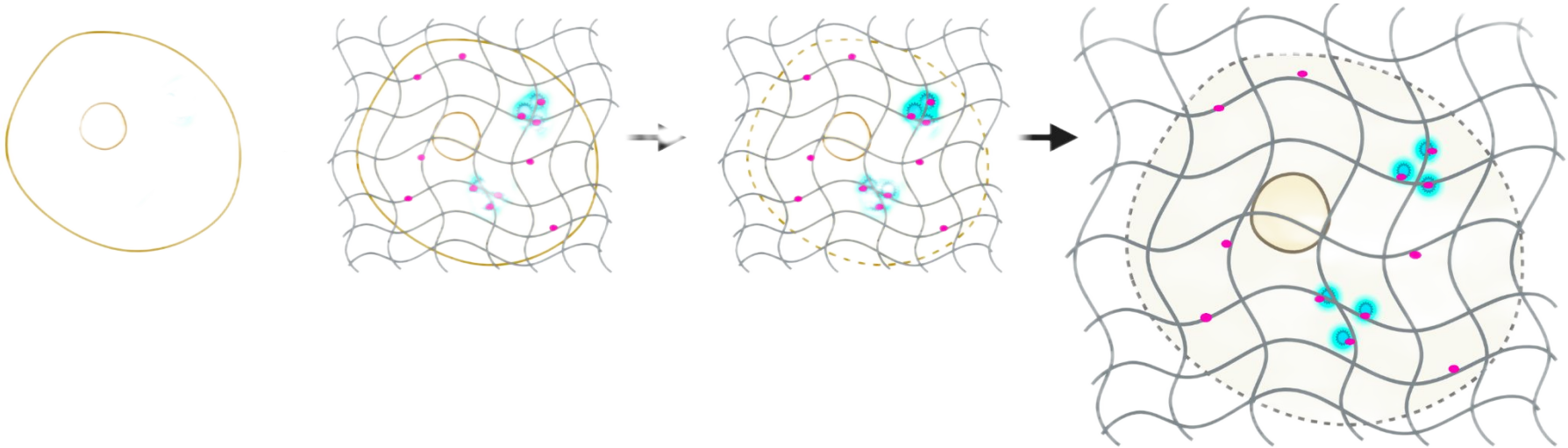






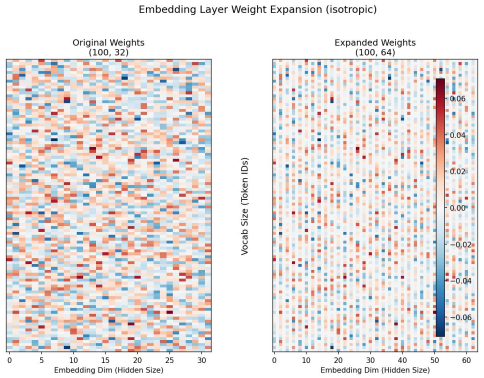
We call this expansion strategy
‘top-left’
because of how it prescribes for weights of intermediate module layers to be expanded .

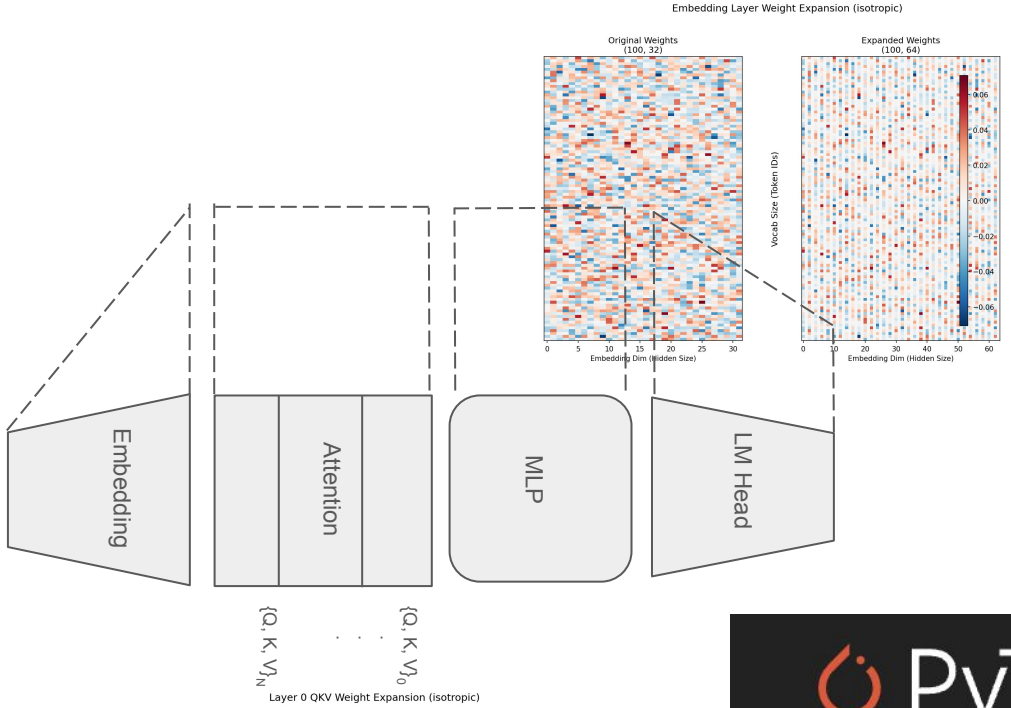
(2) Isotropic expansion

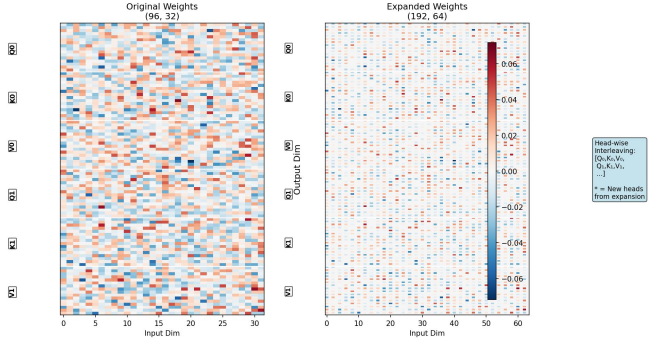


PS: My PhD lab at MIT actually works on expansion microscopy for neuroscience.

(2) Isotropic expansion

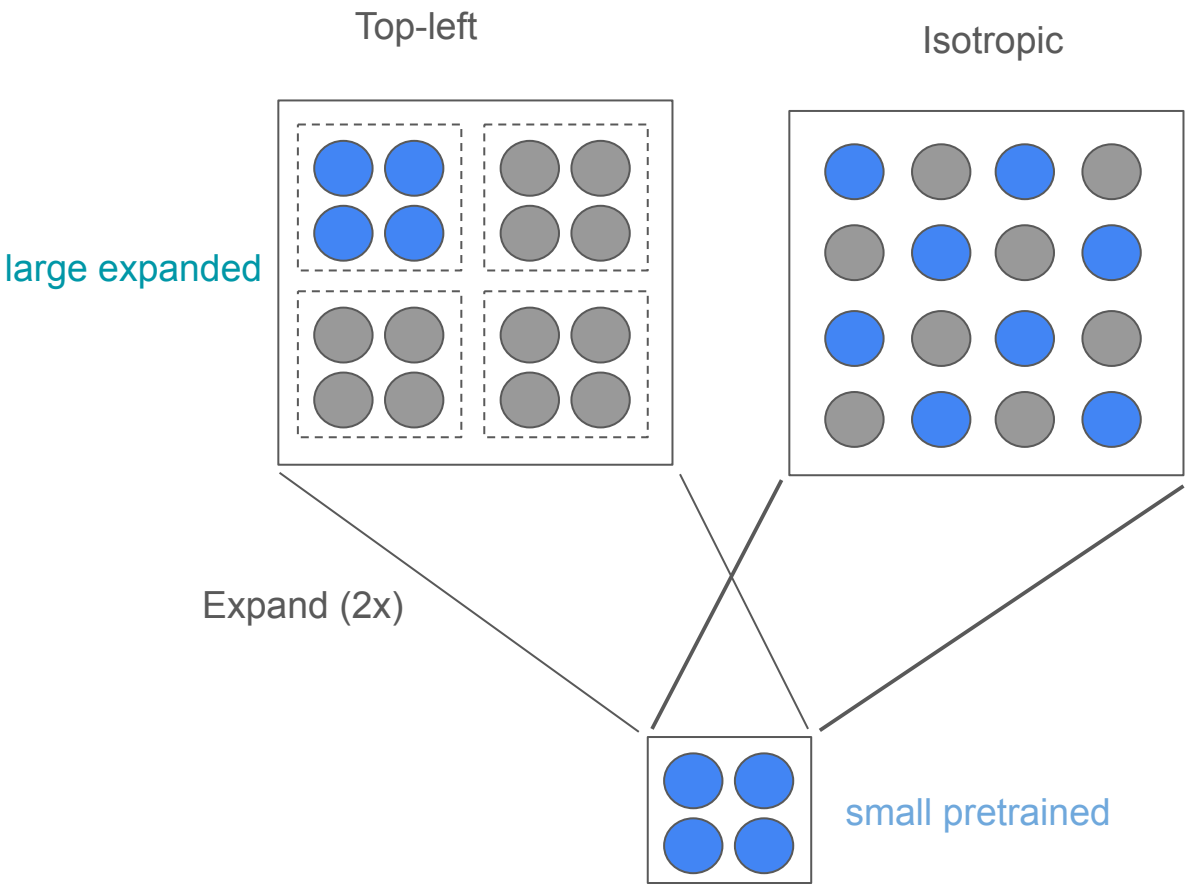


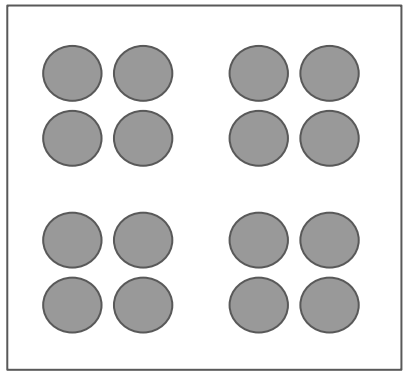




PyTorch

Comparisons for experiments





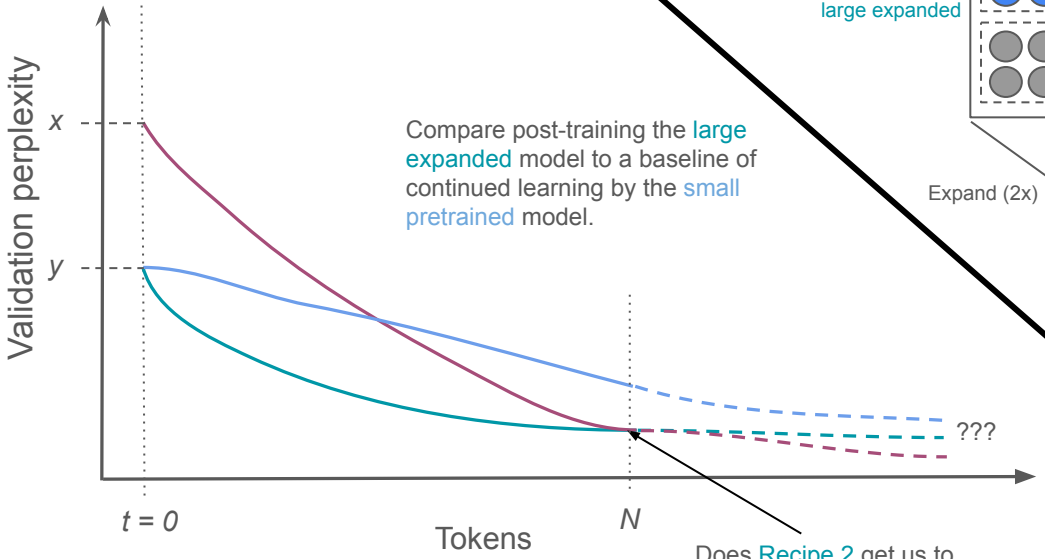
tabula rasa

Fair **baseline**:

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

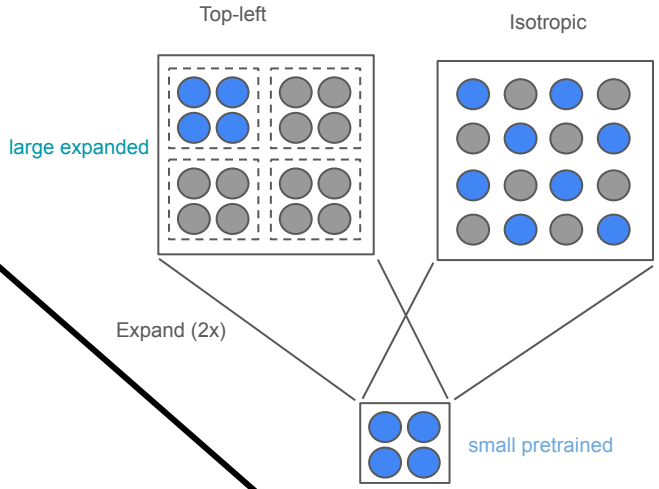
Comparisons for experiments

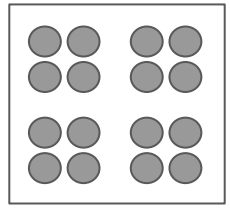
Always keep this plot in mind.



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?



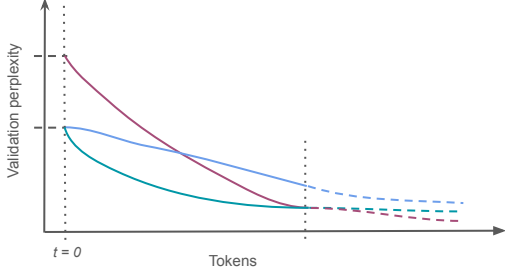


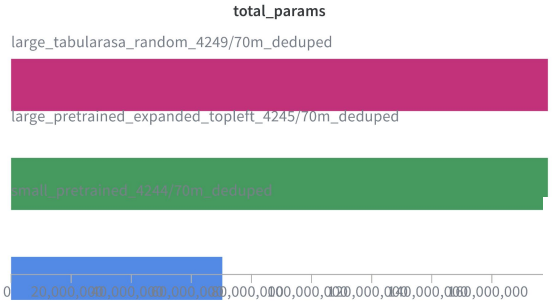
Fair **baseline**:

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

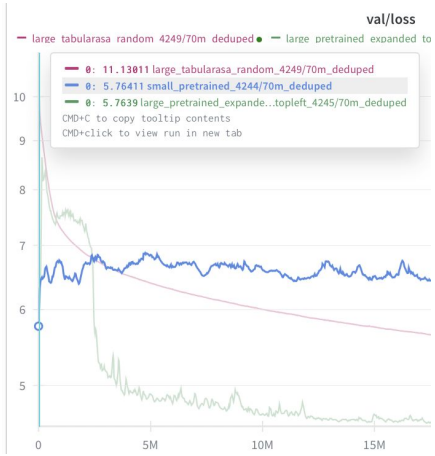
x = starting loss of *tabula rasa* model
y = starting loss of small pretrained and large expanded model
N = equivalence point where Recipe 1 and Recipe 2 cross

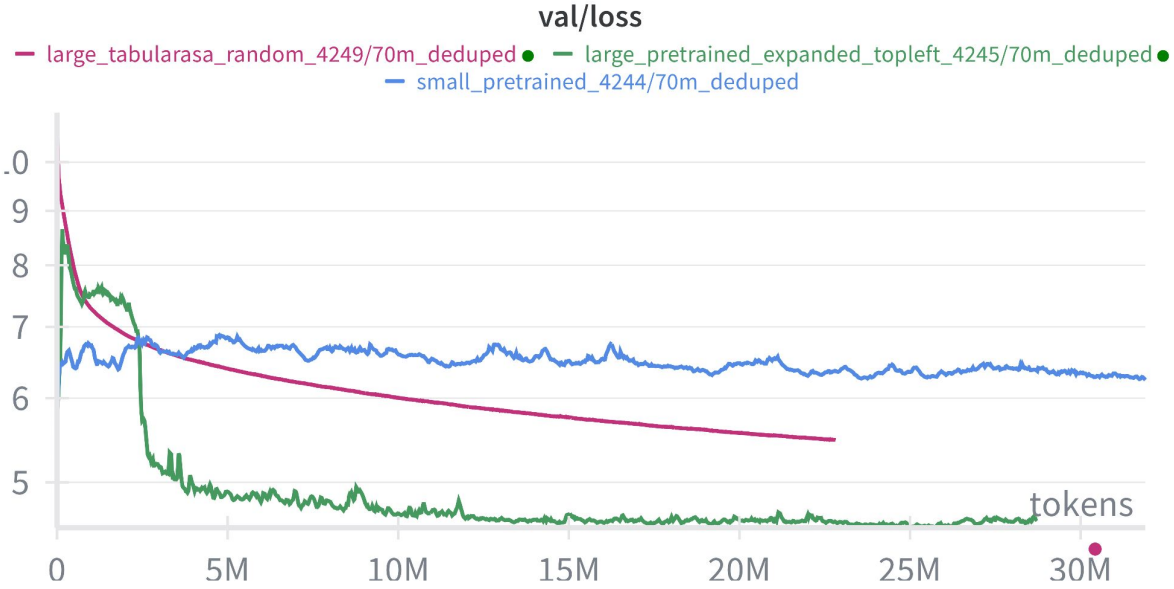
Preliminary result



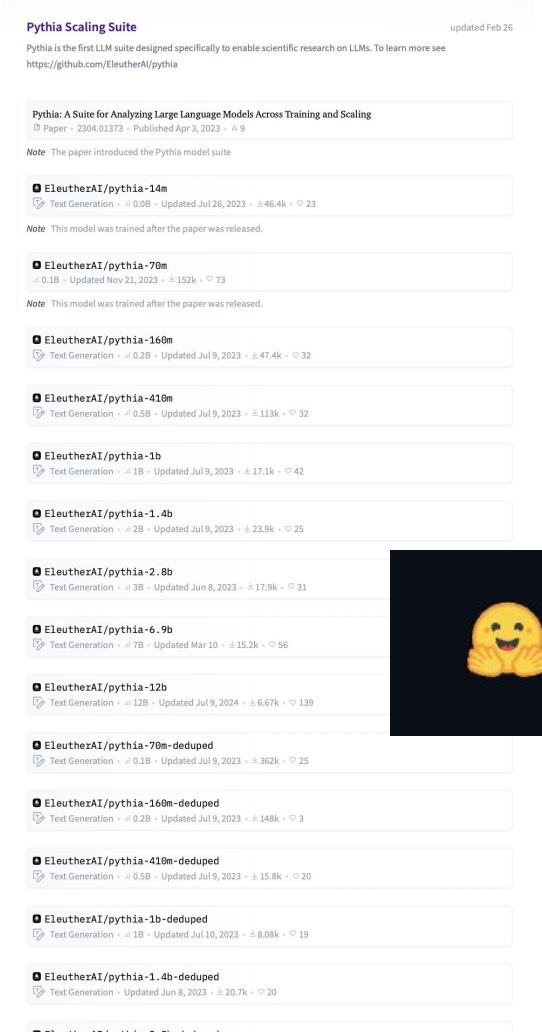


small pretrained : 70M params; init.val.loss 5.76
large expanded : 2x width expansion -> 180M; init.val.loss 5.76
tabula rasa : 180M params; init val.loss 11.13





But how will our results scale with model size?



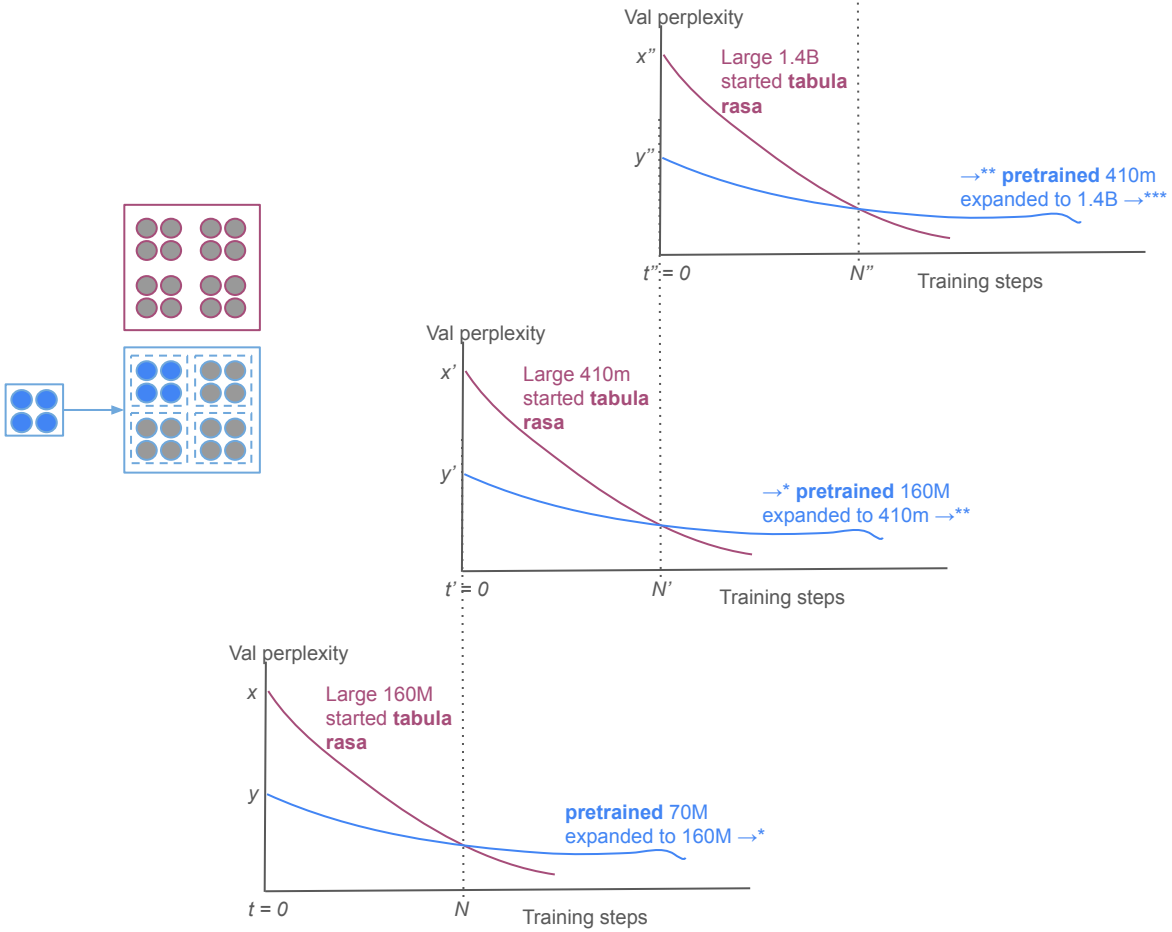
Pythia: A Suite for Analyzing Large Language Models Across Training and Scaling

Stella Biderman [*12] **Hailey Schoelkopf** [*13] **Quentin Anthony** [1] **Herbie Bradley** [14] **Kyle O'Brien** [1]
Eric Hallahan [1] **Mohammad Aflah Khan** [5] **Shivanshu Purohit** [61] **USVSN Sai Prashanth** [1] **Edward Raff** [2]
Aviya Skowron [1] **Lintang Sutawika** [17] **Oskar van der Wal** [8]

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

$x'' > x' >> x$

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

$y'' < y' < y$

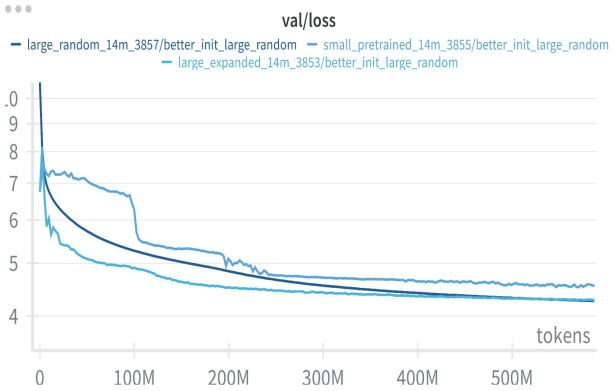
- classic scaling law result that larger models achieve lower loss

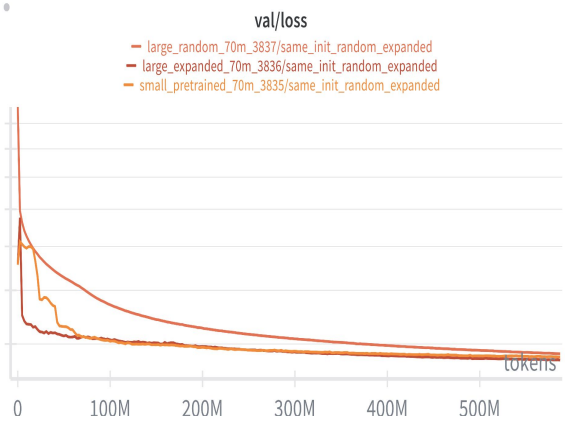
$N'' < N' < N$

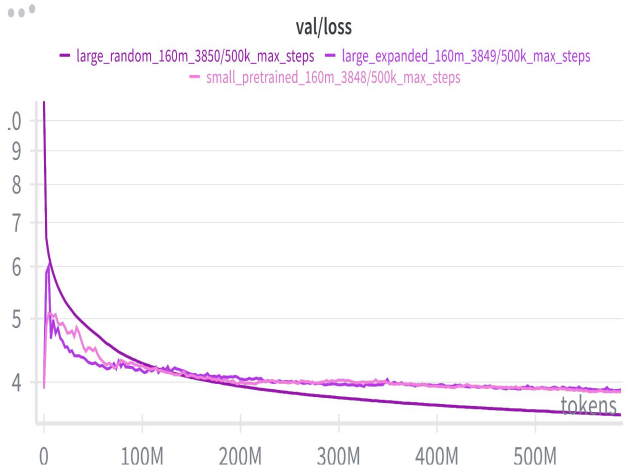
- for larger models, the equivalence point shifts to earlier.

Preliminary results

Weights & Biases
by CoreWeave







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…)