

Hello!

This is a two-part, take-home interview designed to test your ability to think through a fun **sparse-model architecture** problem in an AI systems context.

The metric we are using to measure sparsity is **memory accesses per next token prediction at inference time**. We have chosen this metric because it allows a comparison across dense models and standard approaches like mixture-of-experts, and even wacky things like Infinigrams. We hope you will enjoy sparsifying!

You'll work at the model / architecture level: compete with the target validation loss achieved by the provided starter dense transformer while aggressively reducing **memory accesses per next token prediction at inference time**.

Although this is an interview, we expect it to be educational—and hopefully pretty fun.

This take-home is scoped for a **couple hours per day across two days**, though you may not need all of that time. We'd prefer a small number of well-motivated experiments over a grab-bag of tweaks. Rather than flooding the space with incremental variants, pick a couple of promising ideas, state your hypothesis, design clean ablations, and use the results to iterate. We're evaluating clarity of reasoning and experimental discipline at least as much as the final metric (which we would also love for you to attempt to climb!).

Make sure to document what you tried, why you tried it, what you observed (including failures), and how you'd iterate next.

glhf!!!

weightless

How few memory accesses can we get away with, at inference time?

Your Task

Train a model that achieves **val_loss < 3.0** on a 1.3B sample of FineWeb (e.g. on [alkibijad/fineweb-edu-sample-10BT-gpt2tokenized](#)) while minimizing **memory accesses / bytes touched**. The baseline in the zipped code provided achieves this loss on the dataset fairly quickly—and you might want to run it. but we think that less tokens = more iteration speed!

We'll be treating all memory accesses equally for the sake of this project (even if moving things from e.g. HBM to SRAM can help, let's keep it simple!). The aim of the project is to minimize active bytes touched and to treat the memory hierarchy as flat for simplicity.

That said, unique parameter bytes and unique optimizer state bytes also matter beyond the total bytes touched at inference time, so if you have exciting proposals to improve those, we will also be excited to see them!

Dataset

This repo uses the tokenized [FineWeb-edu-gpt2](#) dataset (GPT-2 tokenizer, 513-token sequences, and we will be using a 1.31B sequence subset).

Setup

Using pixi (recommended):

```
Shell  
pixi install
```

Or with pip:

```
Shell  
pip install -r requirements.txt
```

Log in to wandb (optional but recommended):

Shell

```
wandb login
```

Files

- [data.py](#) - Data loading from HuggingFace streaming dataset
- [model.py](#) - Starter transformer model (modify this!)
- [train.py](#) - Training loop with wandb logging
- [eval.py](#) - Evaluation script

Training

Shell

Using pixi

```
pixi run train
```

Or directly with python

```
python train.py
```

Custom config

```
python train.py --batch_size 64 --max_lr 1e-4 --num_steps 20000 --d_model 128  
--n_layers 2
```

Without wandb

```
python train.py --no_wandb
```

You are welcome to add other evaluation metrics if you'd like!

Submission

We are looking for the following:

- Your code (zipped or a URL)
- Results, in a *Weights & Biases* report or a document. We will look at whichever metrics you recorded.
- Short writeup (1-2 pages): approach and what things you tried, how they worked or failed, and given more time, what things you would try next.

Once again, we care about demonstrated understanding, not just hitting a specific sparsity number!