Abstract

By James Camacho (teammate: Joseph Camacho)

Neural networks are currently bottle-necked by memory access. The actual computations can be offloaded to a GPU or TPU, but transferring the weights still takes time. That begs the question: why not put the weights directly in your circuit? In fact, a startup is already working on this: Etched.

We're unrelated to them, we just think it'd be cool to run neural networks on our FPGAs. Since they're pretty constrained by size---around 3Mb of BRAM and 50k logic cells---we'll use single bits for our parameters and a majority function as our nonlinearity (only takes four gates). Our first goal is to get MNIST working. After that, we might:

- Make a small latent denoiser (some call them diffusers),
- a small LLM, or
- a chess bot.

The issue with the first two is you don't really get good results until about 100Mb. There is a tiny BERT model that's ~6Mb, and it's possible to turn that into a next-token-predictor, and given transformers are incredibly inefficient, a tree structure *might* be able to further compress them, but all of that would be a lot of work not really related to this class.

So, we'll probably make a chess engine. It turns out that current engines use pretty small neural networks---most of their prowess actually comes from beam searching 15+ deep in a couple milliseconds.

To divide up work evenly, we'll talk to each other. We're roommates, so the latency is pretty low.