# High Performance LLMs From First Principles (2024)

Session 9
https://github.com/rwitten/HighPerfLLMs2024

rwitten@

This session:

- Implement efficient inference!
- What else to learn? (lots!)
- One deep dive: talk numerics

Next/last session:

- Pallas deep dive with Sharad Vikram!

# My Asks

Please ask lots of questions! Just raise your hand or speak up!

If there are topics you're interested in, message me between sessions.

Join the discord! https://discord.gg/2AWcVatVAw

Do the exercises! Give feedback, ask questions!

Website: https://github.com/rwitten/HighPerfLLMs2024

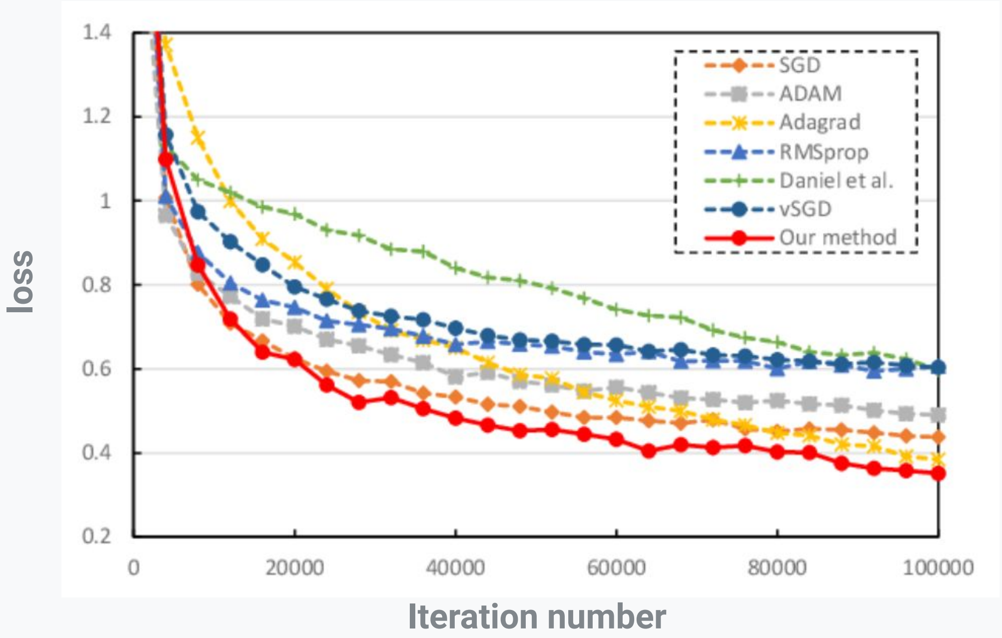
# High Performance Inference (Generate, Batch=1)

- Pass in single token, not multiple tokens.
  - Deal with embedding lookup (jax.lax.dynamic_slice_in_dim)
  - Now NN is silly.
- KV Cache
  - Create KV Cache
  - Read/write from KV cache (jax.lax.dynamic_update_index_in_dim)
  - Fix Attention to work with KV cache
- Time it and compare to memory bandwidth limit.
- Todo (not in class):
  - Quantization!
  - Prefill
  - Batch > 1 (as we saw on rooflines, batch > 1 can give more throughput at the same latency)

# Much Much More To Learn!

- Thinking about Performance goes much deeper!
  - Next week, Sharad Vikram, co-inventor of the kernel language Pallas, will explain Pallas!
  - Pallas lets you more directly program the GPUs and TPUs to achieve high performance in cases where pure XLA can't.
- Broadly the other category is ML Modeling (which we haven't covered at all)! Many many questions!
  - How should we design the network? What types of layers, what order, what mix and what sizes?
  - How should we quantize it?
  - What data should we use?
  - What optimizer should we use?

# ML Modeling – try things, measure loss




- Typical graph! "Our method" looks good!

Thanks Xu et Al, 2017 for the randomly selected example graph.

# ML Modeling – minimizing loss is useful!

TEXT

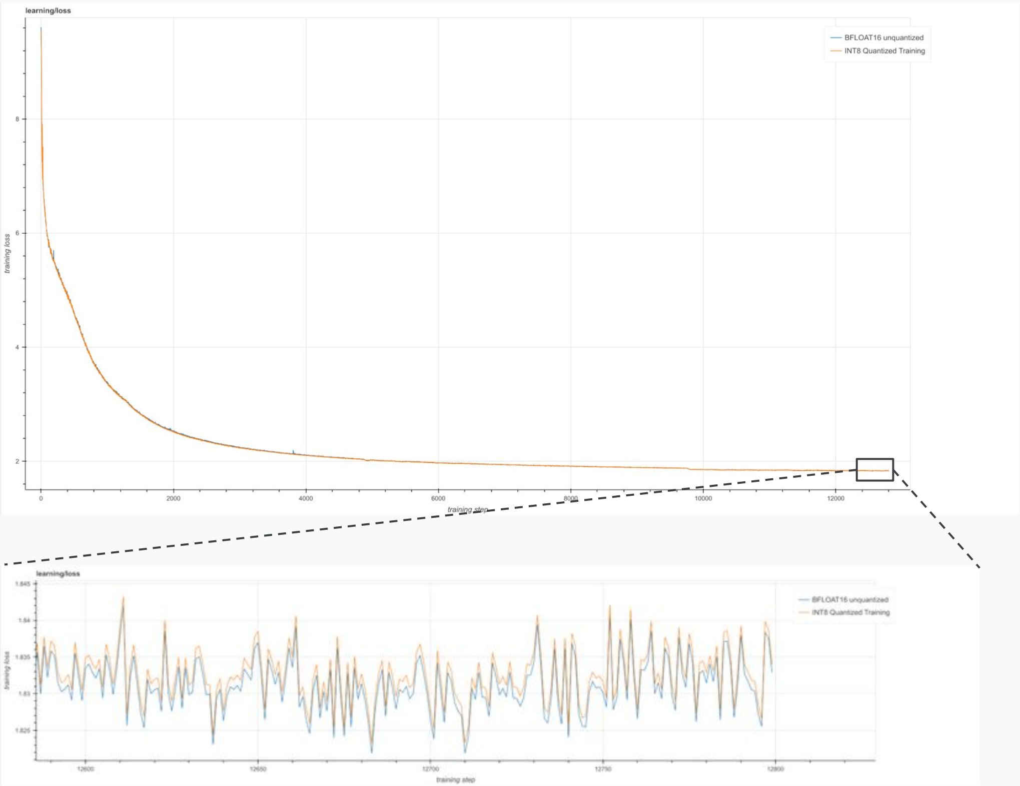
| Capability | Benchmark<br>Higher is better | Description | Gemini Ultra | GPT-4<br>API numbers calculated where reported numbers were missing |
|---|---|---|---|---|
| General | MMLU | Representation of questions in 57 subjects (incl. STEM, humanities, and others) | 90.0%<br>CoT@32* | 86.4%<br>5-shot**<br>(reported) |
| Reasoning | Big-Bench Hard | Diverse set of challenging tasks requiring multi-step reasoning | 83.6%<br>3-shot | 83.1%<br>3-shot<br>(API) |
| | DROP | Reading comprehension<br>(F1 Score) | 82.4<br>Variable shots | 80.9<br>3-shot<br>(reported) |
| | HellaSwag | Commonsense reasoning for everyday tasks | 87.8%<br>10-shot* | 95.3%<br>10-shot*<br>(reported) |
| Math | GSM8K | Basic arithmetic manipulations (incl. Grade School math problems) | 94.4%<br>maj1@32 | 92.0%<br>5-shot CoT<br>(reported) |

# Deep dive into “Numerics” – typical mix of Performance and Modeling!

- Generally, it is much faster for computer chips to do math on lower bit representations of numbers than on higher bit representations!
- But lower precision introduces errors so the models learn less quickly (or maybe not at all).
- The first big change was going from float32 (32-bit floats) to bfloat16 (16-bit floats).
  - (Introduced in 2019 by Google!)
- As an example, let’s consider summing up a vector.
  - Coding exercise…

# Numerics go deep!

- Nowadays, we're training and inferring on models with 8-bit matrix multiplies!
  - For training a ~1.5x speedup, very small quality gap!
- Typically int8 on TPU and fp8 on GPU.
- ([Source Link](Source Link))



# Next/last session:

- Pallas deep dive with Sharad Vikram!

# Thanks!

# Ping me (rwitten@google.com) with feedback, suggested topics, etc!