High Performance LLMs From First Principles (2024)

Session 6
https://github.com/rwitten/HighPerfLLMs2024

rwitten@

Goal: learn how to achieve **high performance** for LLMs

This week:

- End-to-end Training, Measure Performance

Program (write code in Jax)
Predict (roofline on napkin or spreadsheet)
Profile (run code, compare to predictions)

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

Add ons to Session 1

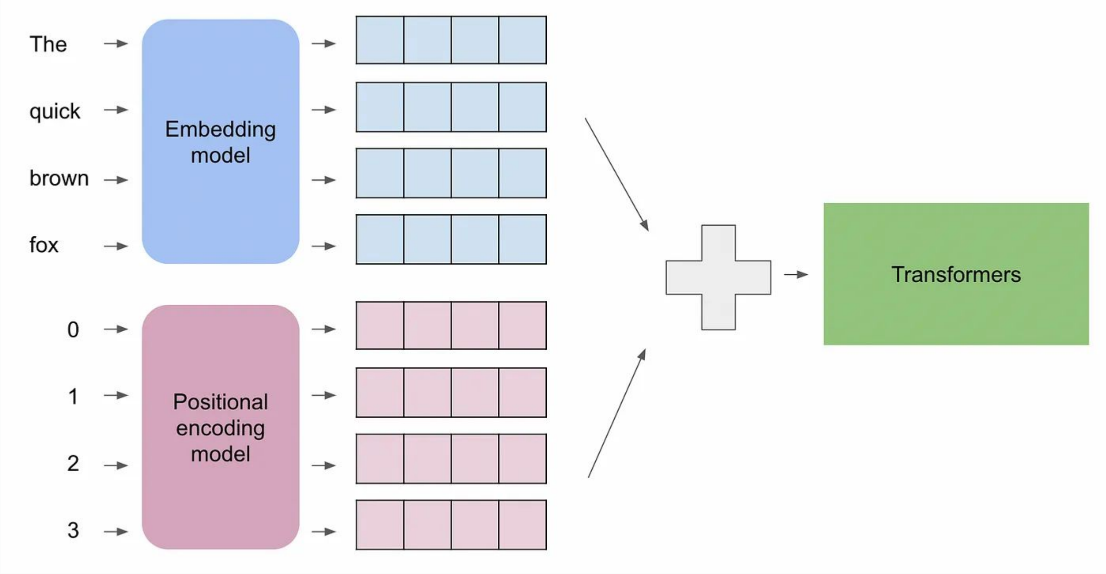
- Didn’t have Attention
- Didn’t do positional encoding
- Didn’t JIT
- Overlapping Data Loading
- Was single chip (not multichip)
- Never computed model flops per second since we didn’t yet know how

Add Attention

- Code

Add Positional Embedding

- The problem is that attention was order invariant
- “Brown” sees “the” and “quick” but doesn’t know the order!
- Solution – add embedding vector



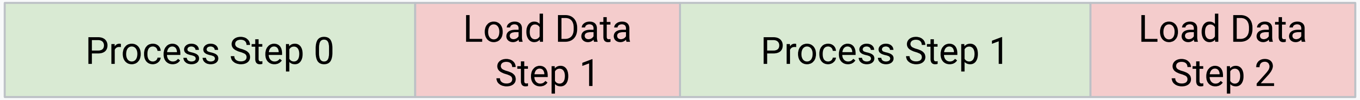
Thanks for the nice picture: https://medium.com/@xuer.chen.human/llm-study-notes-positional-encoding-0639a1002ec0

JIT

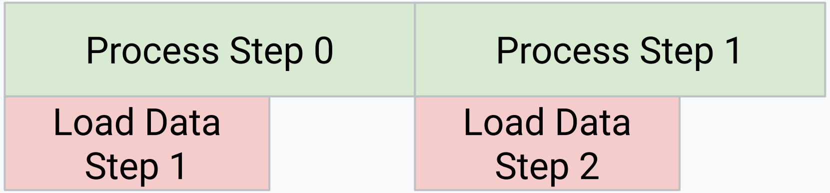
- Not just JITing forwards pass – want to JIT a whole computation traditionally!

Overlapping Data Loading (Lay Track Faster Than Train!)

Naive:

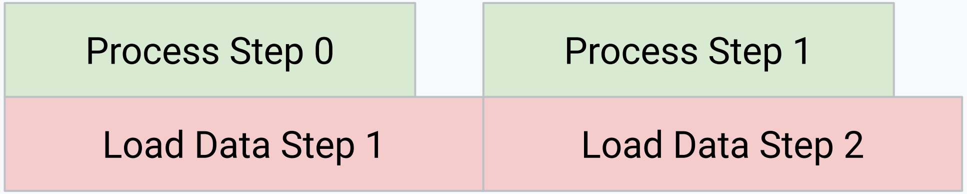


Overlapped:



- Easy to do it wrong by accident! (Also easy to get it right!)
- Assuming you do it right:
  - Data loading doesn’t matter unless it becomes the bottleneck
  - Then it matters a great deal because we’re in 1 for 1 slip.

Overlapped slow:



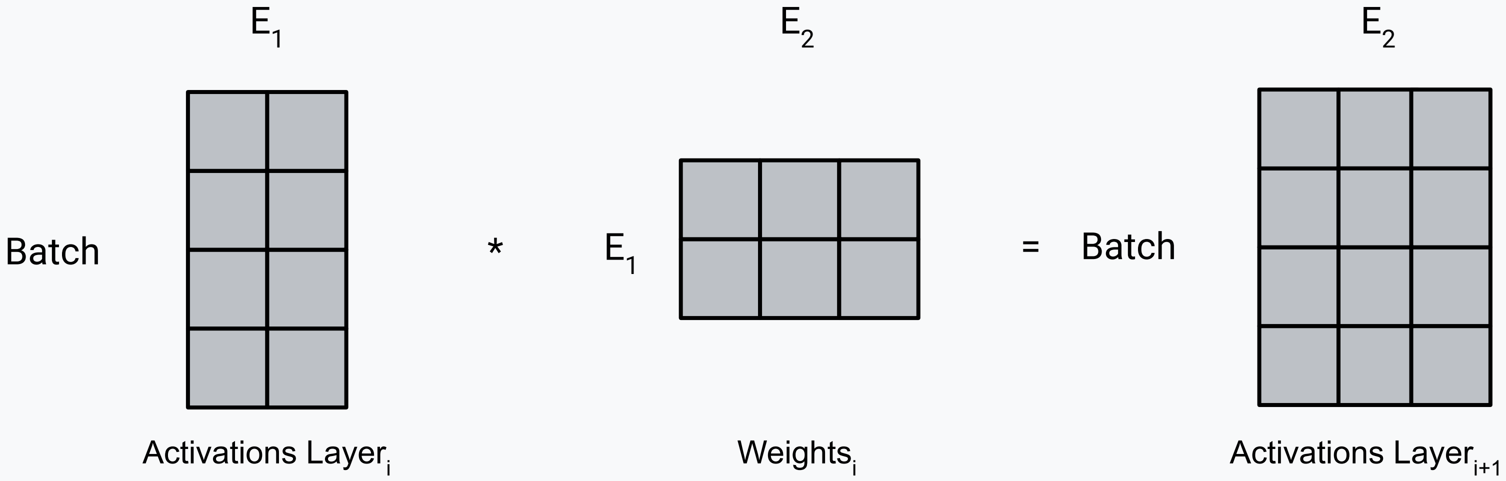
Multichip

- Need to generate params on multiple chips, generate data for multiple chips and then JIT it all together.
- Lots of options but actually a little trickier than it sounds:
    - The key issue is making sure to describe a distributed computation.
    - The anti-pattern is putting everything on one chip and then moving it around – this out-of-memories when things don’t fit on a single chip!
    - Useful: jax.ShapeDtypeStruct

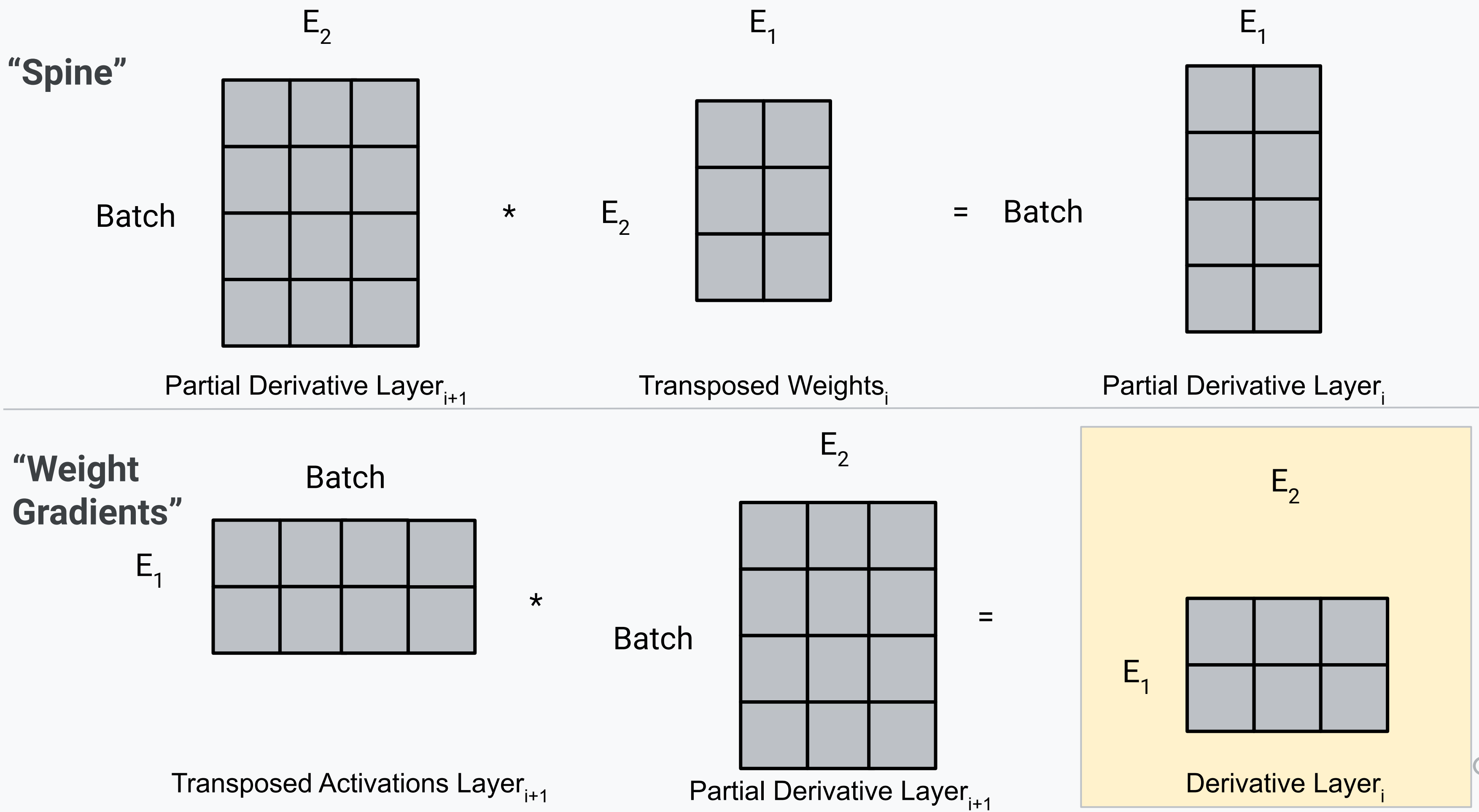
How long should training take?

- We discussed the forwards pass in the past.
- Training adds the backwards pass! Backwards pass is computing the derivatives!

Forwards Pass



Corresponding Backwards Pass



How long should training take?

- We discussed the forwards pass in the past.
- Training adds the backwards pass! Backwards pass is computing the derivatives!
- For every matmul in the forwards pass, two matmuls in the backwards pass:
    - “Spine” – generate the new partial derivations
    - “Weight gradients” – output the gradients we need!
    - All the matmuls take $2*B*E_1*E_2$ flops since they all have a B dimension, an E_1 and an E_2 dimensions! So $6*B*E_1*E_2$ total flops.
- Conveniently:
    - Parameters = E_1*E_2
    - So this matmul during training takes 6*B*P total flops.
    - And that is true for all matmuls – so the number of flops from matmuls is 6*B*P!
- Warning: this calculation ignores attention flops! Attention flops are typically a small fraction of the overall flops…

Next Week

- Get the model converging and do some inference.
- Lots of possible bonus topics, not sure what has interest, happy to cover a couple more if there is strong interest.
    - “Going a Level Deeper in shard_map and Pallas” (possibly with Sharad!)

Thanks!

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