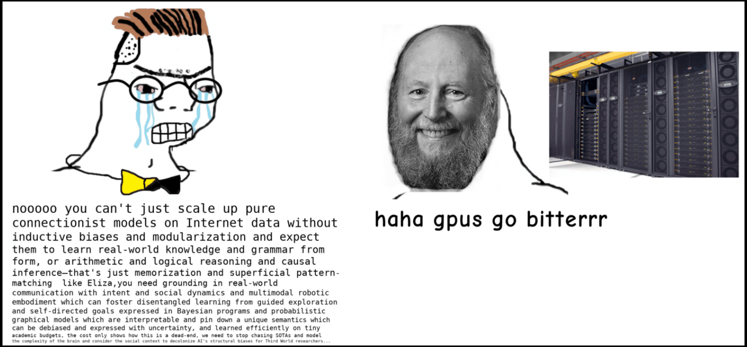
Syllabus day topics

1. what is a GPU kernel?
2. why Triton over CUDA?
3. prerequisites
4. why use this guide over others?
5. things to keep in mind during these lectures

what is a GPU kernel?

GPU: the types of computer processors that we use to make AI, videogames, scientific computing, etc. go BRRRRRR
GPU kernel: the function that defines exactly how to do a desired mathematical calculation using an awareness of how GPUs are structured in order to best take advantage of that structure to facilitate going BRRRRRR


nooooo you can't just scale up pure connectionist models on Internet data without inductive biases and modularization and expect them to learn real-world knowledge and grammar from form, or arithmetic and logical reasoning and causal inference—that's just memorization and superficial pattern-matching like Eliza,you need grounding in real-world communication with intent and social dynamics and multimodal robotic embodiment which can foster disentangled learning from guided exploration and self-directed goals expressed in Bayesian programs and probabilistic graphical models which are interpretable and pin down a unique semantics which can be debiased and expressed with uncertainty, and learned efficiently on tiny academic budgets, the cost only shows how this is a dead-end, we need to stop chasing SOTAs and model the complexity of the brain and consider the social context to decolonize AI's structural biases for Third World researchers...
haha gpus go bitterrr

why Triton over CUDA?

less popular	more popular
open-source	closed-source
both Nvidia & AMD	Nvidia GPUs only
Python	C
90% as fast	gold standard for speed
linux only	linux or windows
less to learn	more to learn

prerequisites

required

- Python
- basic computer hardware concepts (memory, processor, bits vs bytes, floating point operations)
- linear algebra
- calculus
- common deep learning operations (matmul, softmax, attention, etc.)
- PyTorch

preferred

- some basic but not-universal-among-python-programmer concepts *such as*
  - big O notation
  - compile-time vs run-time
- data-structures & algorithms (leetcode)

why use this guide over others?

guides I used were (links in repo readme)

- official triton documentation
- Umar Jamil's flash-attention tutorial
- GPU Mode's lecture series 14

they all some number of the following issues

- bugs (kernels straight up didn't pass tests)
- slower than PyTorch
- assumed you already know CUDA
- little to no attempt to explain what was happening
- unnecessarily confusing/overcomplicated
- primarily one format (text XOR video)

things to keep in mind

- i'm not an expert (but I can beat PyTorch)
- corrections and elaborations will go in the pinned comment on each video
- if you cannot build something you do not understand it! watching/reading is not good enough. you need to go build something sufficiently complex in order to claim comprehension