COS 597k: Systems for Serving Generative AI

Instructor: Ravi Netravali

Fall 2024

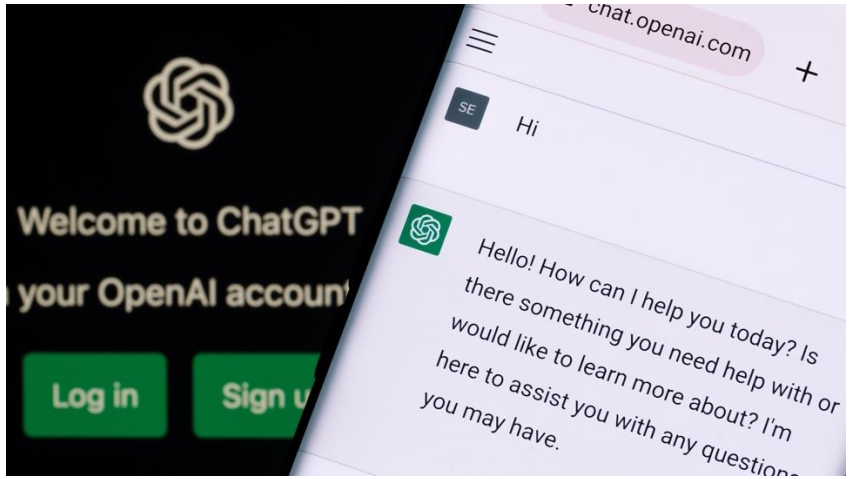
https://www.cs.princeton.edu/~ravian/COS597_F24/

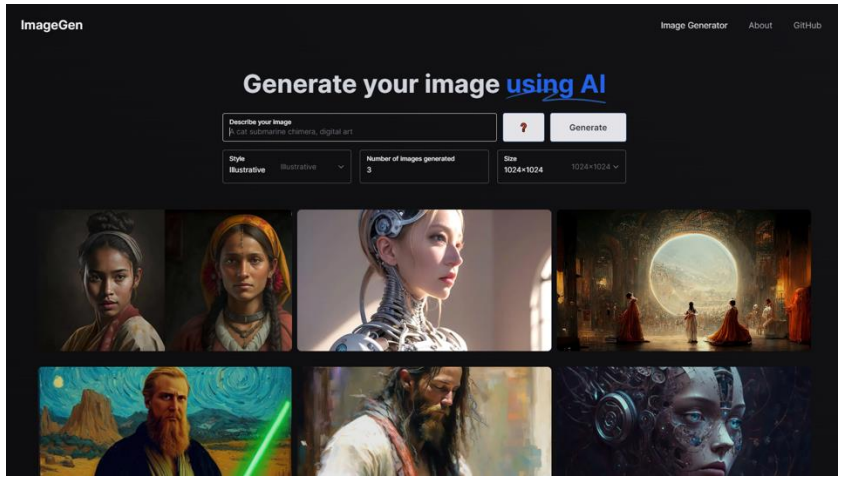
About me

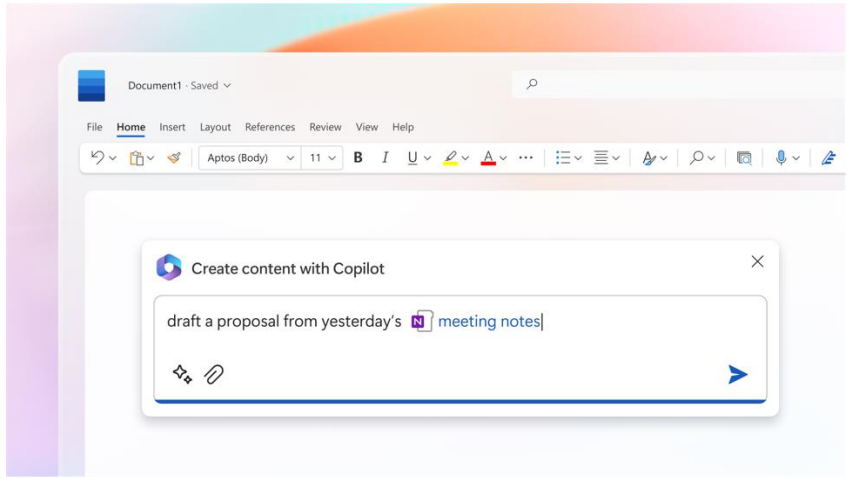
- PhD MIT in 2018
- Professor at UCLA CS 2018-2021; Professor at Princeton CS since 2021
- Research: systems and networking (mostly systems-ML these days…)
- Co-Founder at BreezeML: startup focused on AI governance and LLM safety

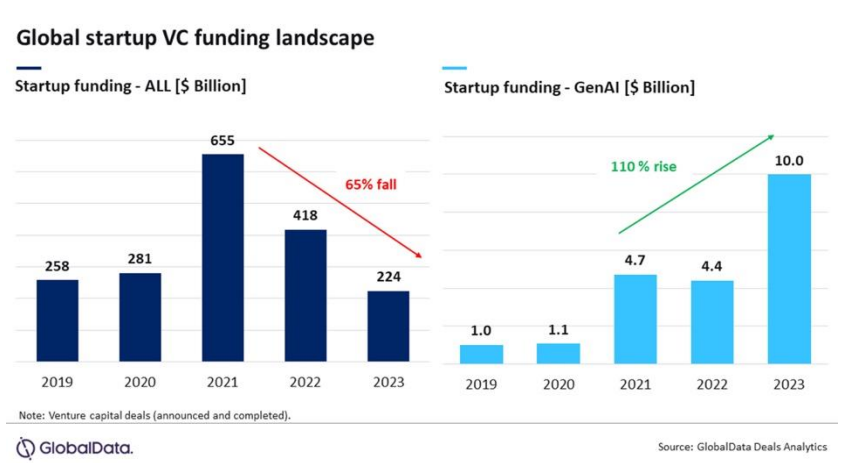
AI is everywhere

And now it’s all about GenAI









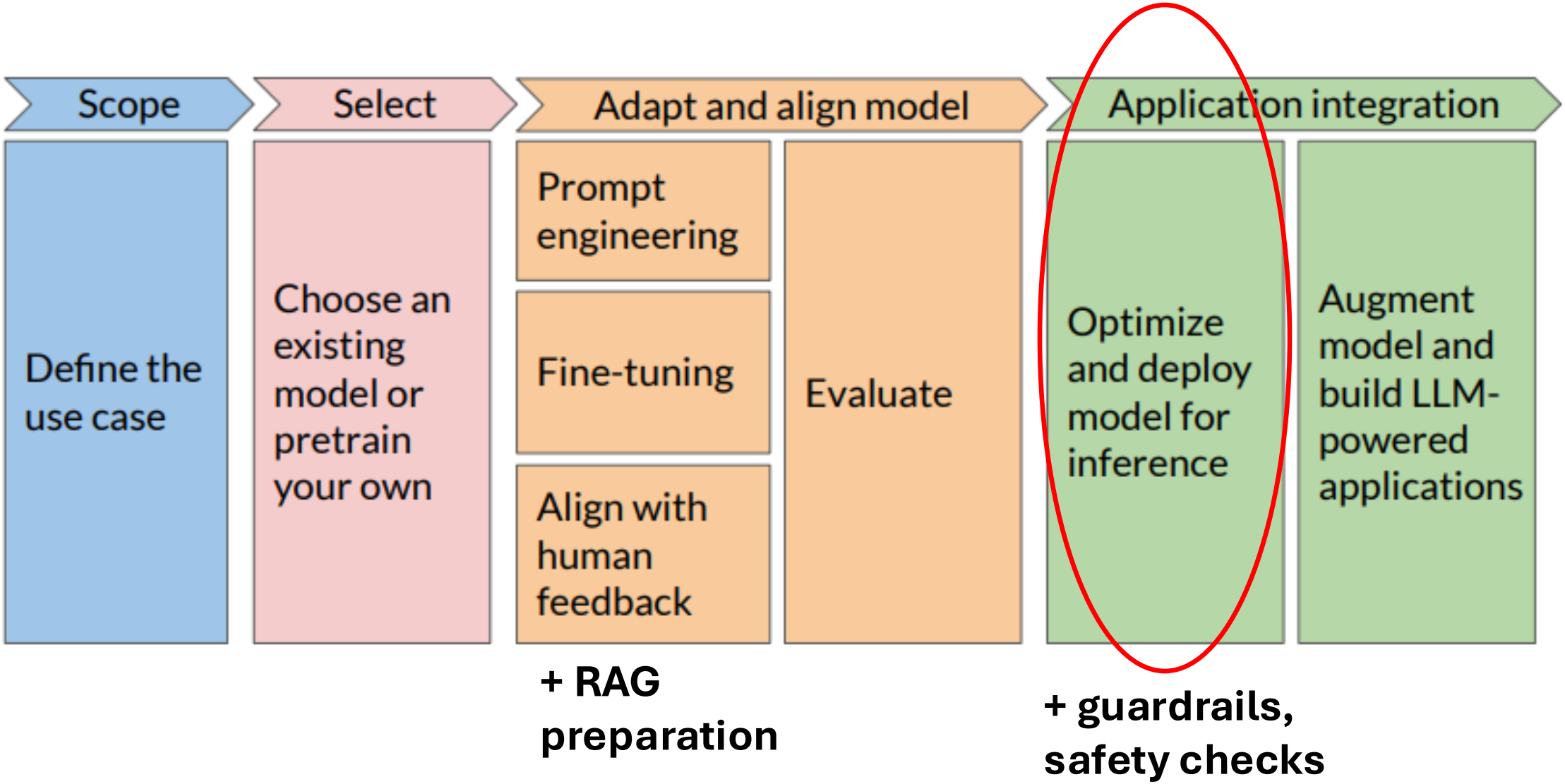
Why has GenAI taken off?

- Generative AI has been around since the 1960's with early chatbots

- What's different now?
  - Improved models and ML algorithms (e.g., transformers/attention in LLMs)
  - More data
  - More compute (especially cloud computing)
  - Better infra (lower latency)

- Predominant case of this: ChatGPT!

Stages of Generative AI



Why focus on serving and not training?

- Fewer people are really training models these days (mostly hyperscalers)
- Serving is becoming the real bottleneck – can't even take advantage of the models we already have!
  - Each new innovation (often) brings different resource requirements and serving complexity
- Serving pipelines are getting more complex –recent innovation in surrounding generative models with other components (compound AI systems)
- Very strict requirements (online, user-facing!)

So what's hard about serving at scale?

- Often have to make do with limited resources (GPUs are expensive, shortages)

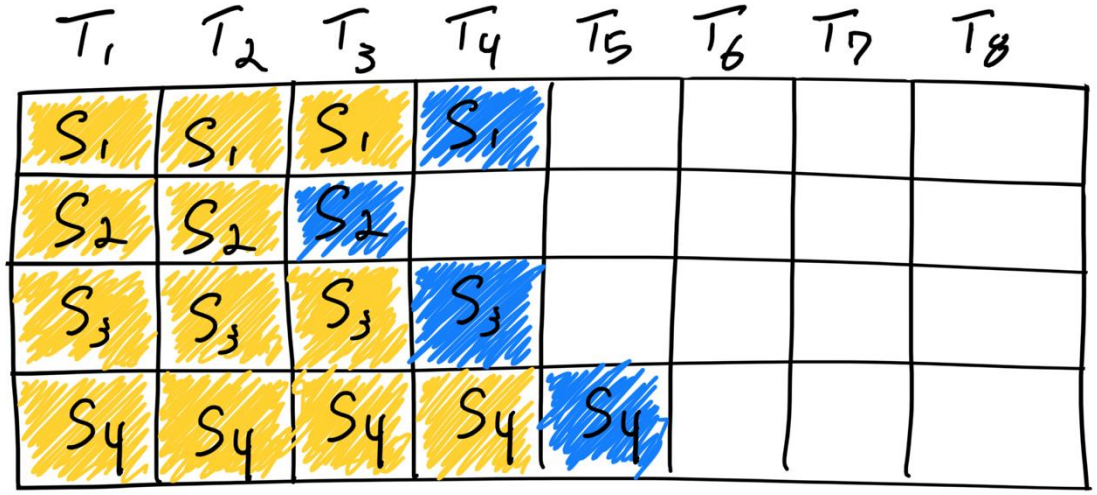
The right answer is constantly changing with new paradigms/optimizations!

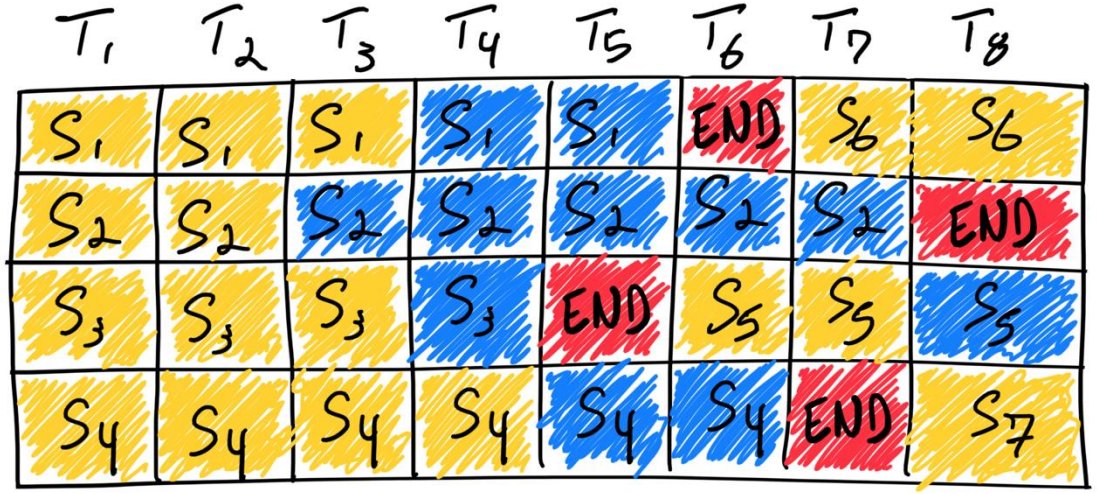
- Where do you place models (or model components)? What requests to schedule and when?

Key course topics

- Core systems decisions for LLM serving
- LLM serving on consumer-grade GPUs
- Compound AI systems (RAGs, Agents)
- Mixture of Experts
- LoRA
- Image generation

Core systems decisions for LLM serving



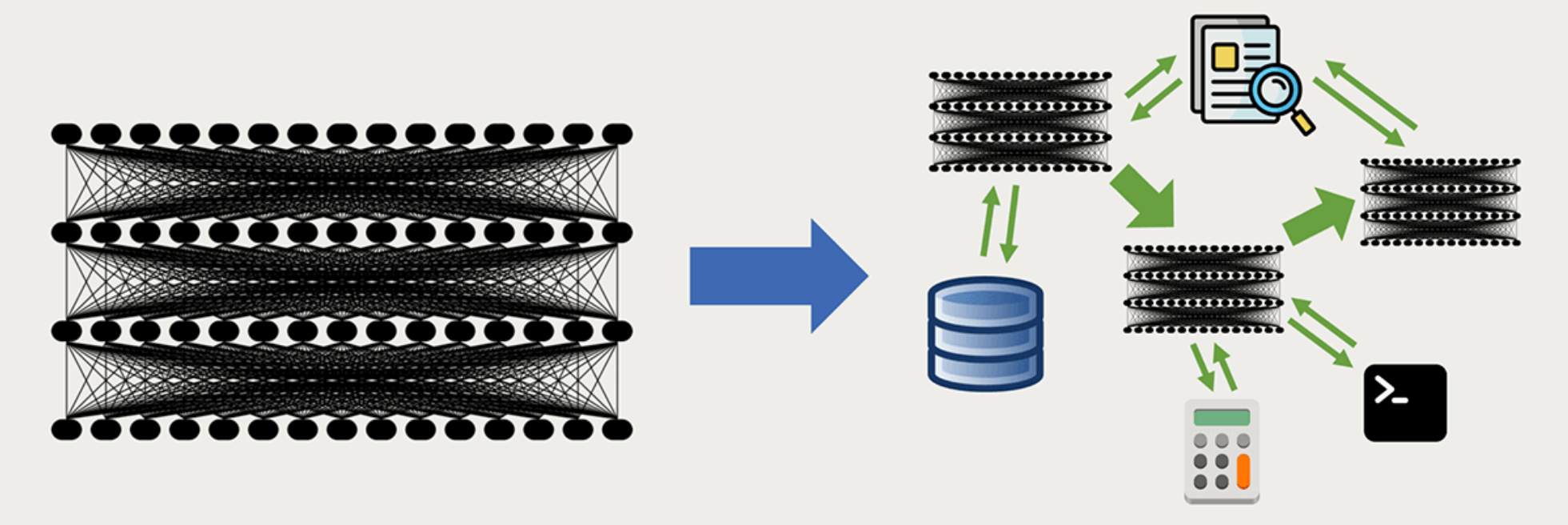


- Batching, throughput vs. latency
- How to deal with auto-regressive nature and variable output lengths
- Scheduling across generation phases
- State/memory management

LLM serving on consumer-grade GPUs

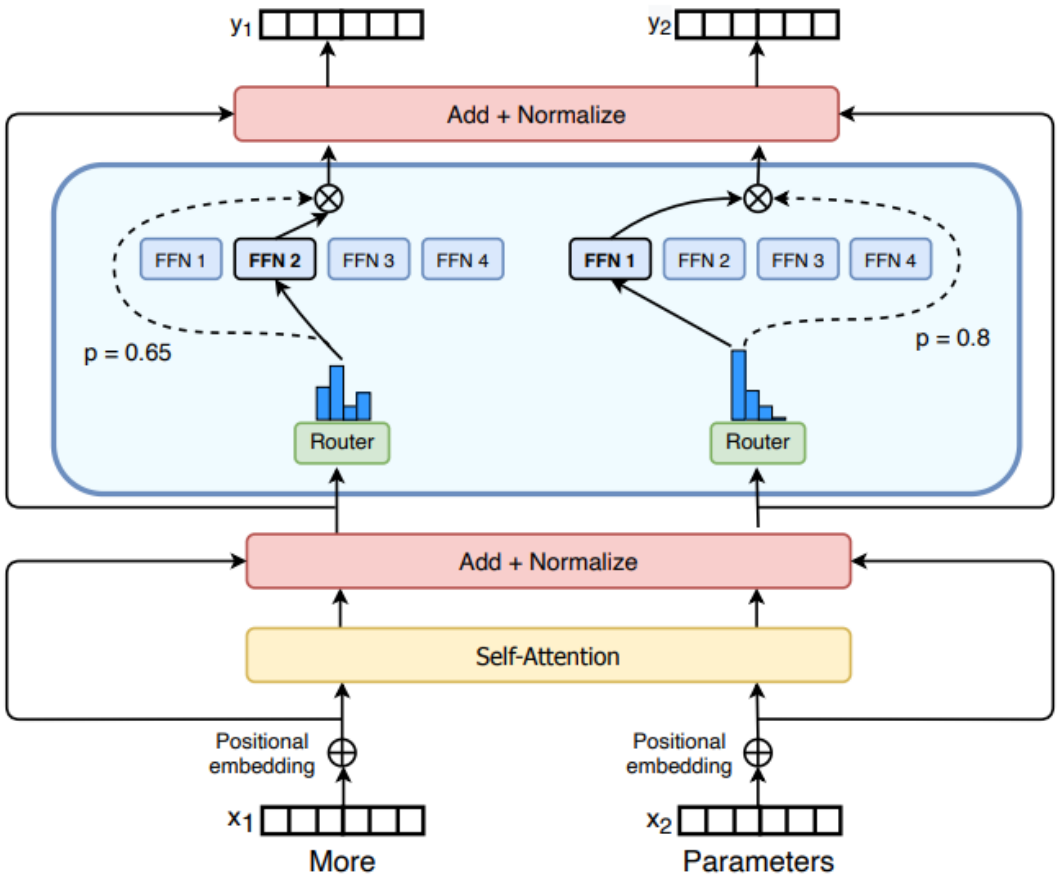
- How to deal with limited GPU memory (each token needs the whole LLM!)
- CPUs? GPUs? A mix?
  - How to deal with communication delays/overheads?

Compound AI Systems



- How to schedule and evaluate requests with multiple steps, e.g., batching
- How to generate content quickly despite pauses to fetch context (RAG)
- How to schedule across different tasks for different requests (agents)

Mixture of Experts

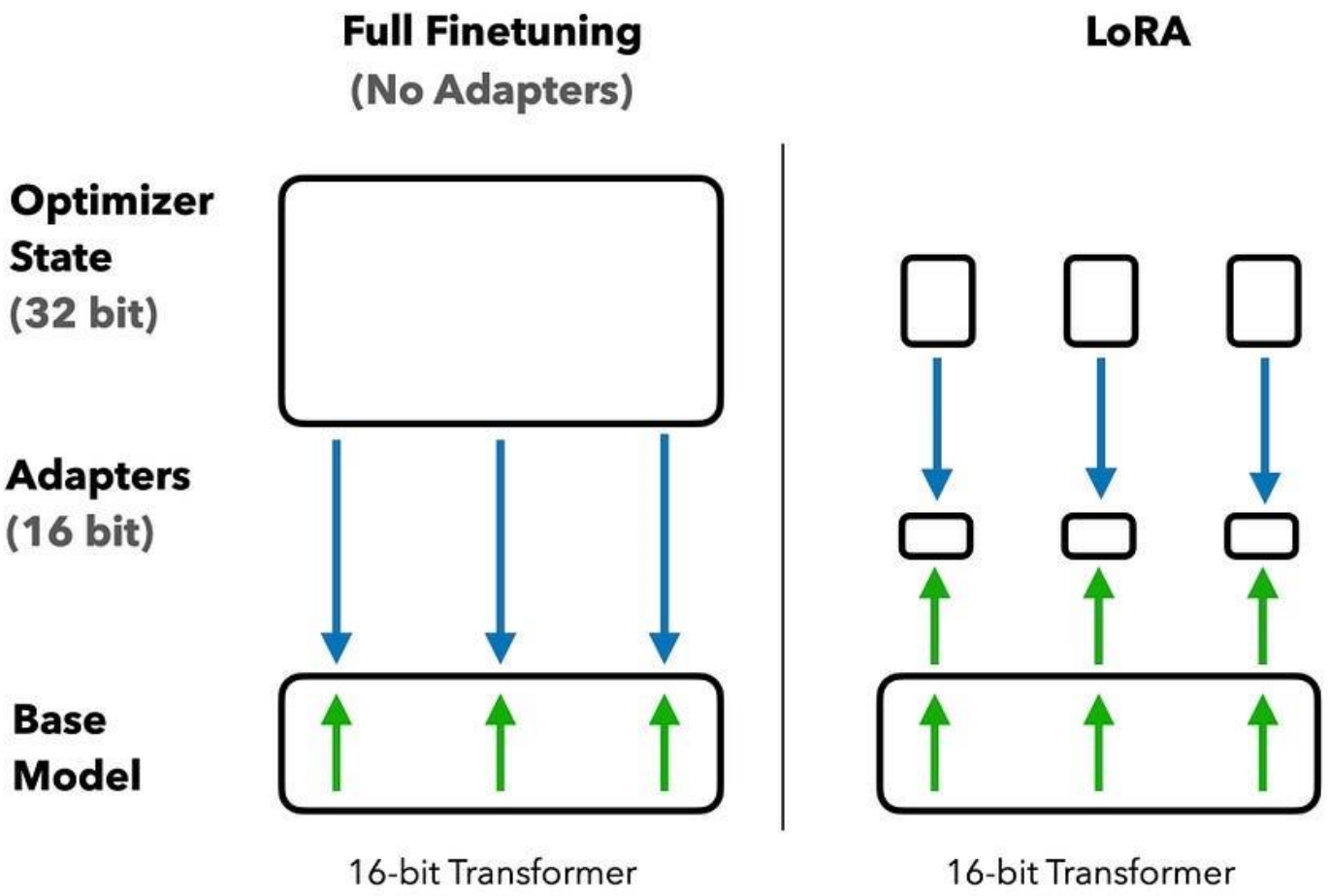


How to deal with higher memory needs to house experts?

How to manage uncertainty in how execution will unfold?

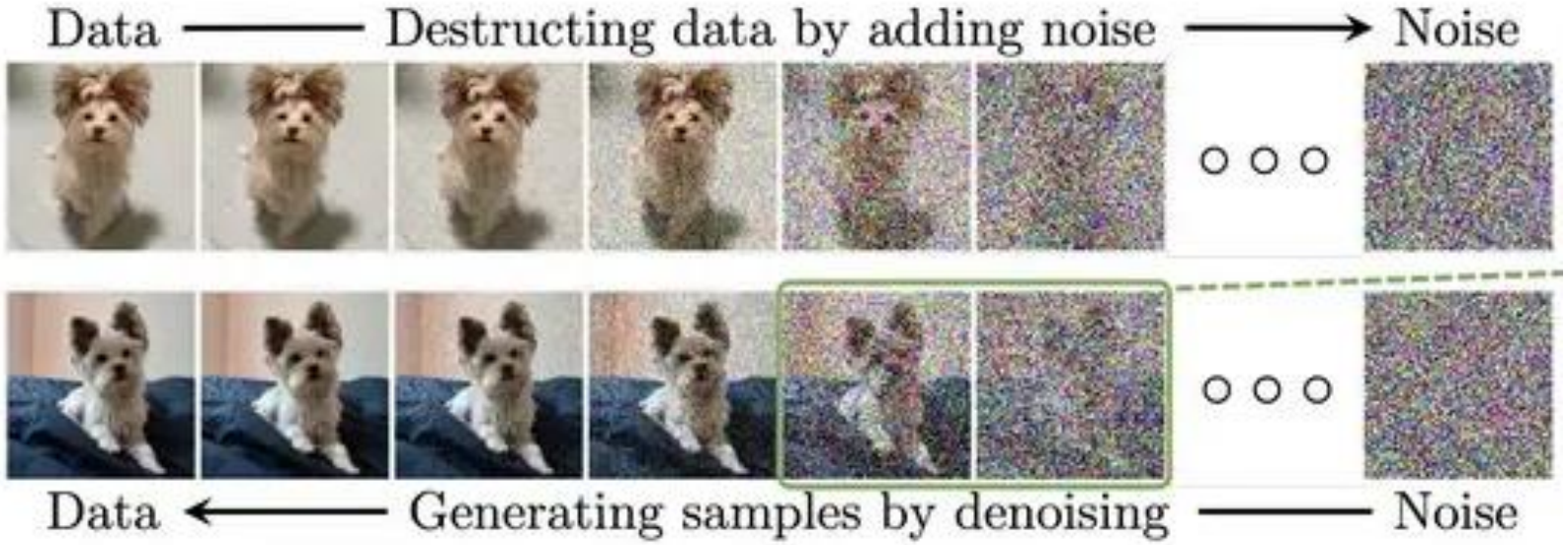
How to manage communication overheads between experts?

Low-rank adaptation (LoRA)



- How to efficiently serve when you have multiple models, some with different base and some with different adapters?
- How to deal with uncertainty in generation time/length when scheduling tasks per adapter/model?

Image generation



- Many rounds to denoise and generate polished image → resource intensive and slow!

- High resolution images(lots of state) with spatial dependencies → hard to parallelize!

Why take this class?

- Interest in systems more so than ML/AI
- Have taken core systems courses (e.g., OS, distributed systems) and/or performed systems research
- Are comfortable learning via research papers and discussion, as opposed to lectures
- Want to understand what's going on *under the hood* of serving platforms, e.g., not just Python frameworks

What is this course *not* about?

- Not studying ML algorithms; instead, given ML paradigm, how do we optimize serving platforms for latency, throughput, resource efficiency

- Not going to cover things like how to use TensorFlow, PyTorch, etc.

- Exclusively inference and generative models; for more general sysml class, consider COS 598D “Systems and Machine Learning” with Prof. Kai Li

Course structure

- Paper to read and discuss each week
- Present 1-2 papers and lead the discussions for them
- Course project

Paper presentations

- Sign up by next Wednesday, 9/11; 1-2 people per paper
  - https://shorturl.at/gjlGC
- Thoroughly present the paper *and* lead discussion
- Presentation (~30 mins): background for area, problem description, motivational results, *detailed* overview of solution, and evaluation
  - Much more detailed than typical conference talk!
  - Background may require reading and summarizing other work (prior papers, blogs, etc.)
- Discussion: come prepared with key questions or ideas to spark discussion
  - *Everyone* should come prepared with at least 2 discussion prompts and also actively contribute to discussion

Project

- Goal: *motivate* a systems problem/optimization for serving generative models (other serving scenarios may be acceptable)
  - Often requires motivational measurements and design sketches
  - Do not need to implement large-scale system so thing big!

- Project elements
  - 1-2 page proposal due Friday, 10/11
  - In-class presentation on Wednesday, 12/4
  - 5-6 page writeup due Dean's date

- 1-2 people

- Start thinking early!!!

Grading

- 40% in-class participation in paper discussions
- 25% paper presentation
- 35% course project

Next week

- Orca – dynamic batching for LLMs

- Presenter: Yinwei