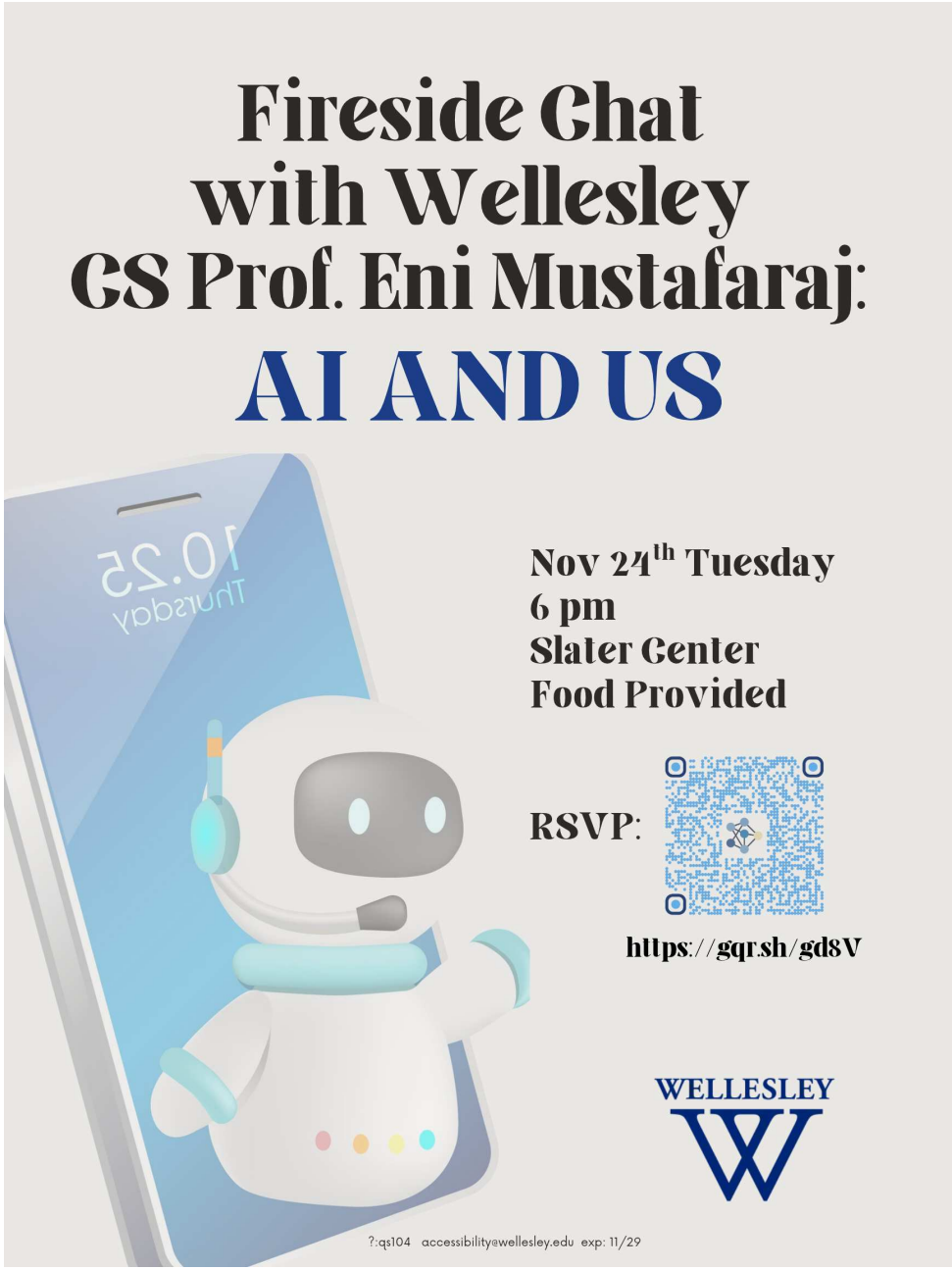
CS 232: Artificial Intelligence

Fall 2026

Prof. Carolyn Anderson
Wellesley College

Reminders

- Homework 8 is released
- My help hours:
  - Friday 3-4 in W423
  - Monday 3-4 in W323



Homework 8

In HW 8, you will *evaluate* a dataset that you might use for your final project:

- Provenance and Intended Uses
- Composition and Representativeness
- Accuracy and Annotation Quality
- Bias, Privacy, and Potential Harms
- Fitness for Purpose

Recap: Transformers

What is attention?

We'll measure the "**importance**" of each input vector $\boldsymbol{x}$ by its similarity (dot product) to $\boldsymbol{v}$

$\boldsymbol{r_1 = v \cdot x_1}$ $\boldsymbol{r_2 = v \cdot x_2}$ $\boldsymbol{r_3 = v \cdot x_3}$ $\boldsymbol{r_4 = v \cdot x_4}$ $\boldsymbol{r_5 = v \cdot x_5}$



x_1: I **x_2: loved** **x_3: the** **x_4: movie** **x_5: !**

Adapted from the slides by Bamman (2021)

Why dot product?

- Dot product provides a measure of similarity between keys and queries.
- But you might be wondering: *why do we want to pay attention to words that are similar to the current word*?

Consider:

My brother, a chemist, was late yesterday because he missed the bus. When he arrived, he was surprised to find that his lab _____

lab



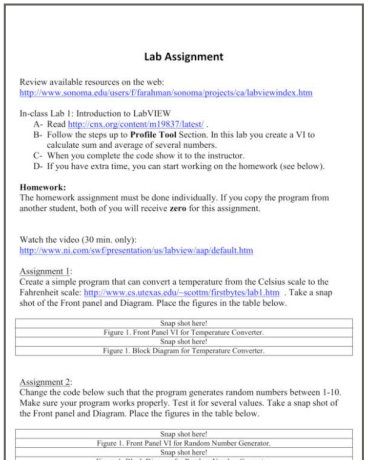
lab



lab



lab



Lab Assignment

Review available resources on the web:
http://www.sonoma.edu/users/f/farahman/sonoma/projects/ca/labviewindex.htm

In-class Lab 1: Introduction to LabVIEW
A- Read http://cnx.org/content/m19837/latest/ .
B- Follow the steps up to **Profile Tool** Section. In this lab you create a VI to calculate sum and average of several numbers.
C- When you complete the code show it to the instructor.
D- If you have extra time, you can start working on the homework (see below).

Homework:
The homework assignment must be done individually. If you copy the program from another student, both of you will receive **zero** for this assignment.

Watch the video (30 min. only):
http://www.ni.com/swf/presentation/us/labview/aap/default.htm

Assignment 1:
Create a simple program that can convert a temperature from the Celsius scale to the Fahrenheit scale: http://www.cs.utexas.edu/~scottm/firstbytes/lab1.htm . Take a snap shot of the Front panel and Diagram. Place the figures in the table below.

Snap shot here!
Figure 1. Front Panel VI for Temperature Converter.
Snap shot here!
Figure 1. Block Diagram for Temperature Converter.

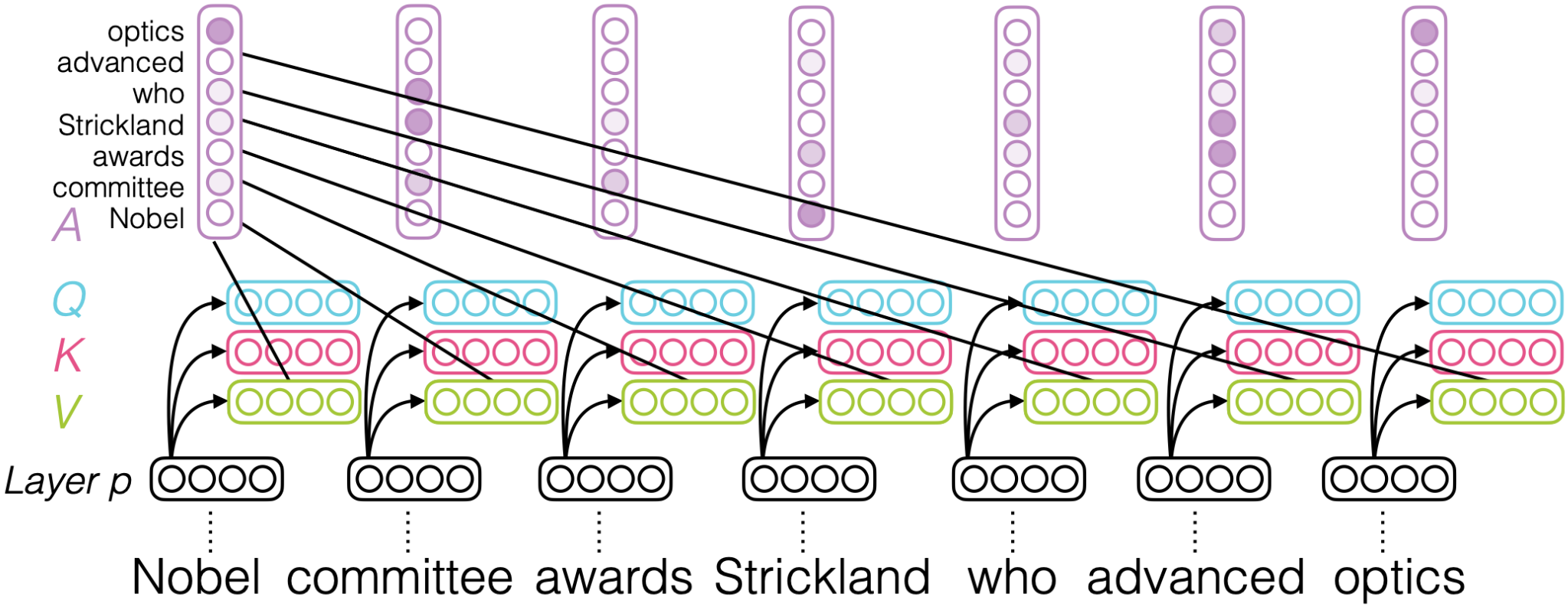
Assignment 2:
Change the code below such that the program generates random numbers between 1-10. Make sure your program works properly. Test it for several values. Take a snap shot of the Front panel and Diagram. Place the figures in the table below.

Snap shot here!
Figure 1. Front Panel VI for Random Number Generator.
Snap shot here!

Slides by Emma Strubell!

Self-attention

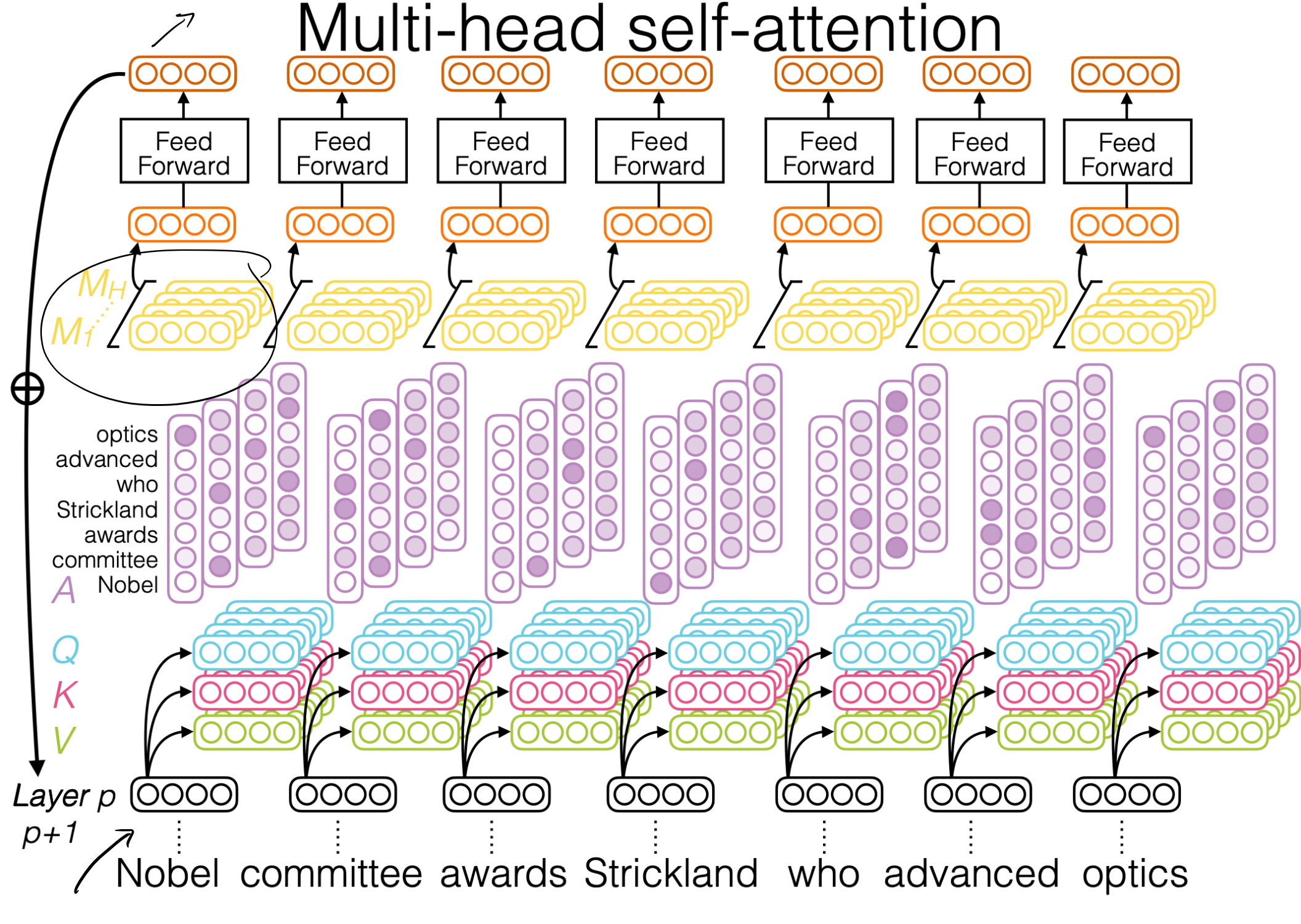
[Vaswani et al. 2017]



Slides by Emma Strubell!

[Vaswani et al. 2017]

Multi-head self-attention



Text Generation

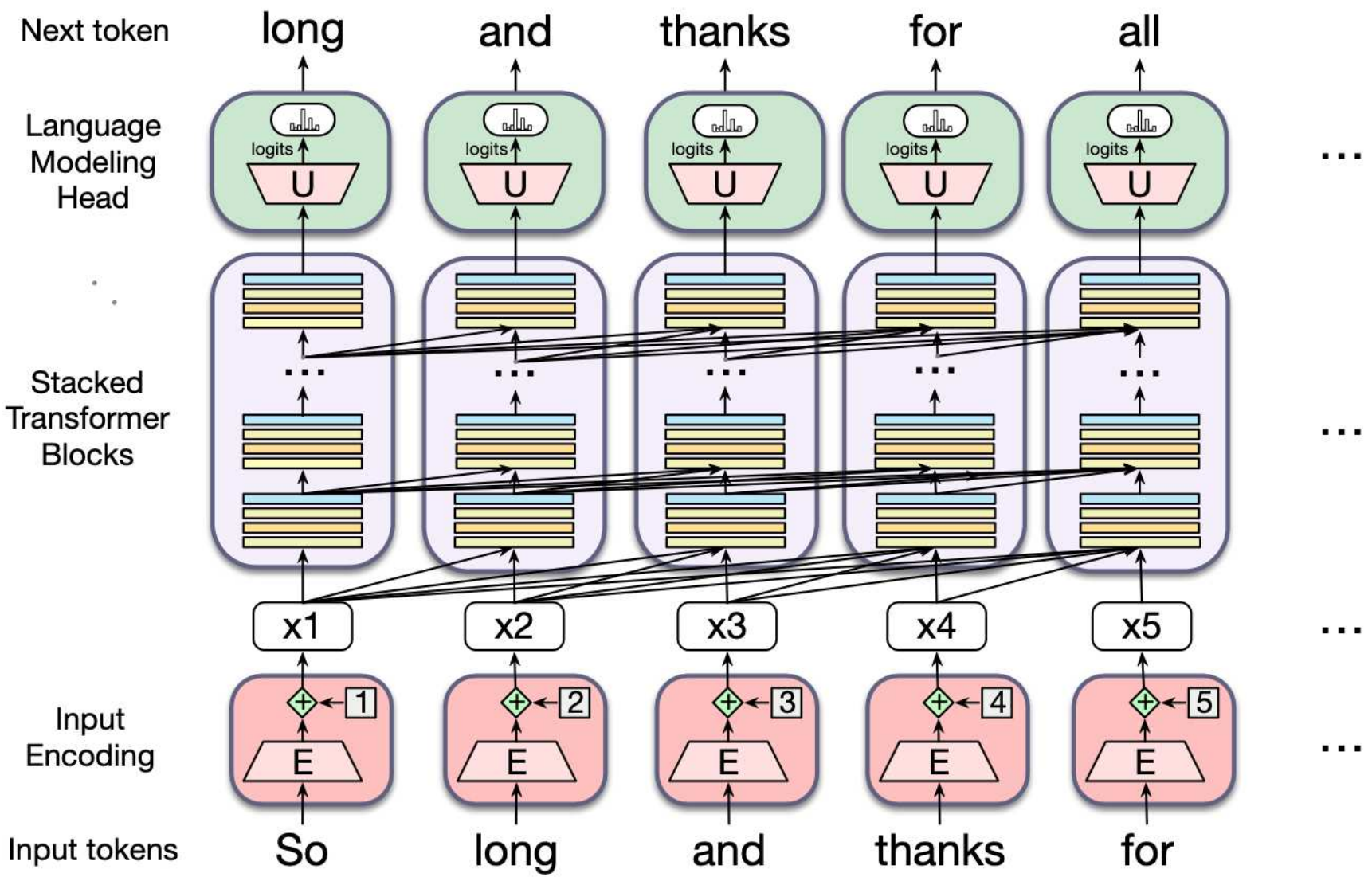


Image from Jurafsky & Martin, Chapt. 7

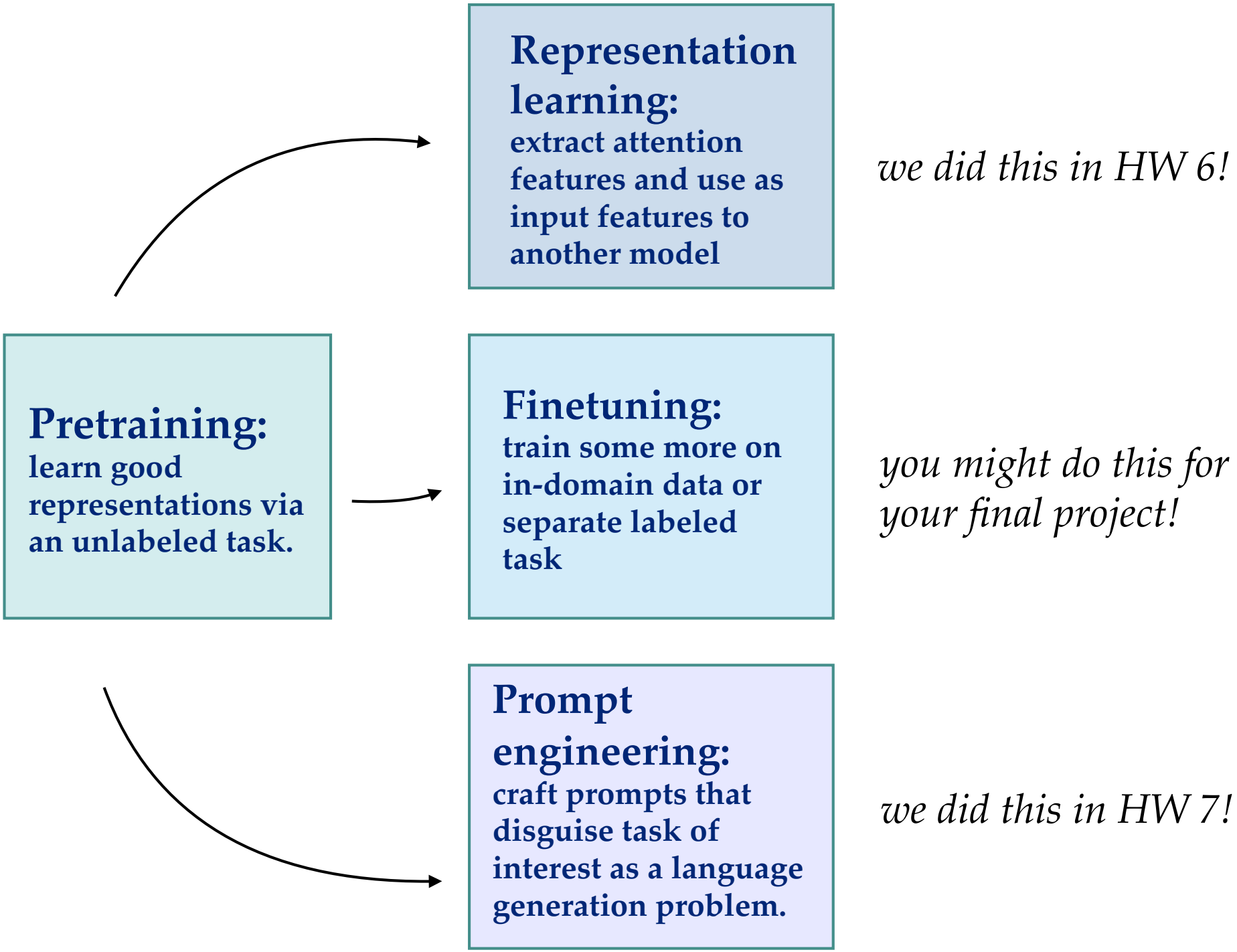
AI Needs You: Conclusion

- A theme running throughout this book is that you don't need to be an expert to have an opinion about AI's impact on your life. Discuss this passage:

  You don't need to be an AI expert to have an informed opinion about AI. In fact, those who are not AI experts may have more experience living and working with AI than those building it.

- What did you think of this book overall?

Using LLMs


Representation learning:
extract attention features and use as input features to another model
we did this in HW 6!
Pretraining:
learn good representations via an unlabeled task.
Finetuning:
train some more on in-domain data or separate labeled task
you might do this for your final project!
Prompt engineering:
craft prompts that disguise task of interest as a language generation problem.
we did this in HW 7!

Prompt Engineering

Few-Shot Prompting

Often called *in context learning*.

Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.

```
1  Translate English to French:        ← task description
2  cheese =>                           ← prompt
```

One-shot

In addition to the task description, the model sees a single example of the task. No gradient updates are performed.

```
1  Translate English to French:        ← task description
2  sea otter => loutre de mer          ← example
3  cheese =>                           ← prompt
```

Few-shot

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.

```
1  Translate English to French:        ← task description
2  sea otter => loutre de mer          ← examples
3  peppermint => menthe poivrée        ←
4  plush girafe => girafe peluche      ←
5  cheese =>                           ← prompt
```

Fine-tuning

The model is trained via repeated gradient updates using a large corpus of example tasks.

```
1  sea otter => loutre de mer          ← example #1
```

↓ gradient update ↓

```
1  peppermint => menthe poivrée        ← example #2
```

↓ gradient update ↓

...

```
1  plush giraffe => girafe peluche     ← example #N
```

↓ gradient update ↓

```
1  cheese =>                           ← prompt
```

Few-Shot Prompting

Few-shot prompting usually helps performance for text tasks.

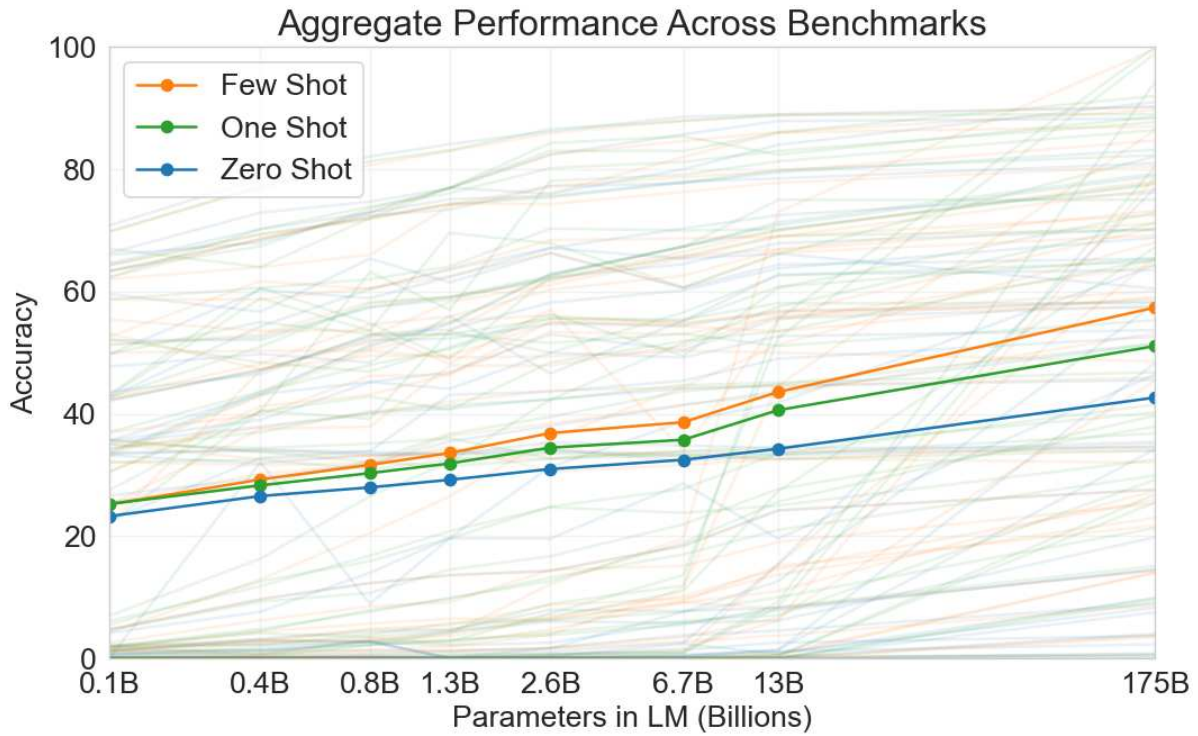


Figure 1.3: Aggregate performance for all 42 accuracy-denominated benchmarks While zero-shot performance improves steadily with model size, few-shot performance increases more rapidly, demonstrating that larger models are more proficient at in-context learning. See Figure 3.8 for a more detailed analysis on SuperGLUE, a standard NLP benchmark suite.

[Brown et al. 2020]

Few-Shot Prompting

However, it can be highly sensitive to the choice of examples, their ordering, and the format of the prompt.

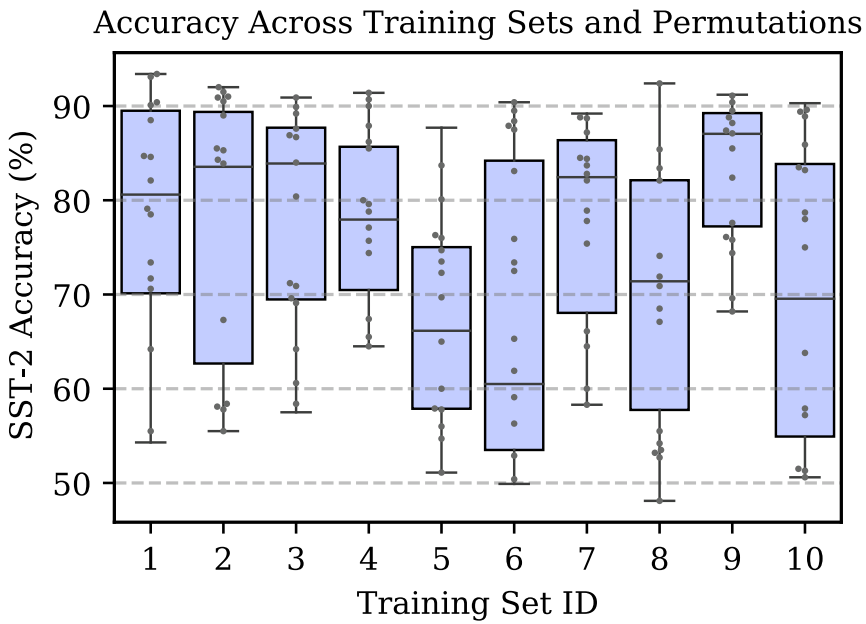


Figure 2. There is high variance in GPT-3's accuracy as we change the prompt's **training examples**, as well as the **permutation** of the examples. Here, we select ten different sets of four SST-2 training examples. For each set of examples, we vary their permutation and plot GPT-3 2.7B's accuracy for each permutation (and its quartiles).

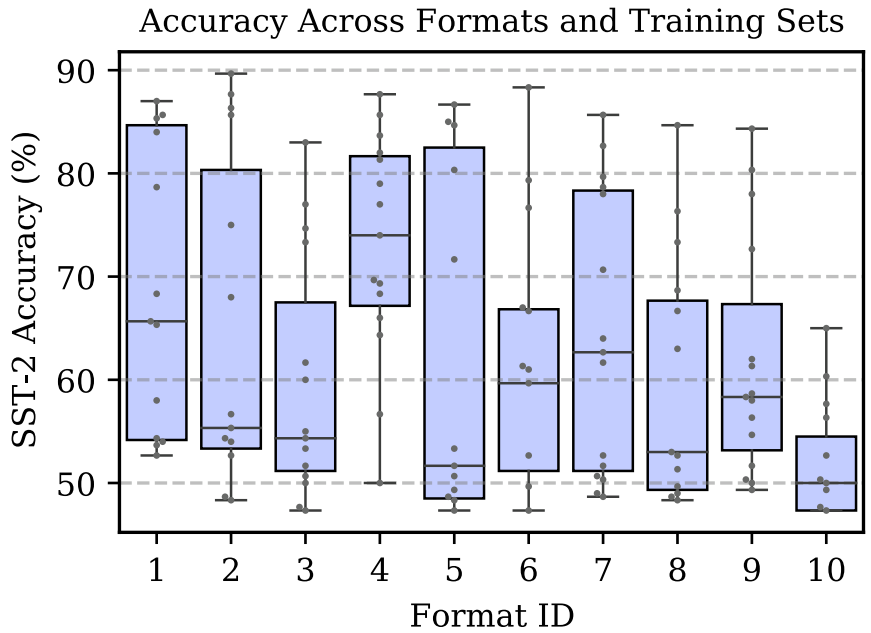


Figure 3. There is high variance in GPT-3's accuracy as we change the **prompt format**. In this figure, we use ten different prompt formats for SST-2. For each format, we plot GPT-3 2.7B's accuracy for different sets of four training examples, along with the quartiles.

[Zhao et al. 2021]

Few-Shot Prompting

Also, it seems not to work as well for vision-language models.

GlyphPattern: An Abstract Pattern Recognition for Vision-Language Models

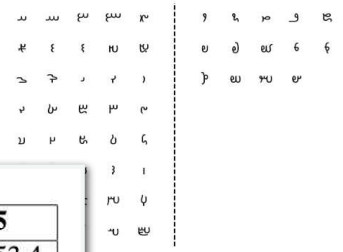
Zixuan Wu
Northeastern University
zi.wu@northeastern.edu

Yoolim Kim
Wellesley College
ykim6@wellesley.edu

Carolyn Jane Anderson
Wellesley College
carolyn.anderson@wellesley.edu

Abstract

Vision-Language Models (VLMs) have made rapid progress in reasoning across visual and textual data. While VLMs perform well on vision tasks that they are trained on, our results highlight key challenges in abstract pat-



rs on the right side in the image are
s that contain a loop or circular

Example GlyphPattern item in the le
. The writing system is Avestan.

Model	0-shot	1	3	5
Gemini-1.5	47.1	52.0	**53.7**	53.4
Gemini-1.5 CoT	46.6	-	39.3	-
GPT-4o	**55.6**	53.4	55.4	52.9
GPT-4o CoT	50.0	-	49.1	-
Idefics2	**31.5**	30.1	31.2	30.0
Idefics3	**34.3**	33.5	32.7	33.3
InstructBLIP	24.1	-	-	-
LLaVA-NeXT	**32.9**	27.0	28.0	26.8
Molmo-O	42.6	-	-	-
Molmo-D	**46.0**	-	-	-

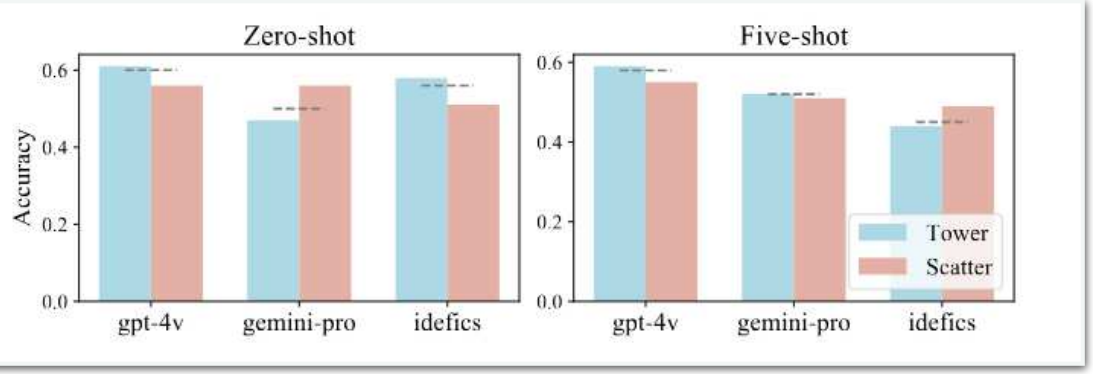
Table 1: Zero-shot and few-shot accuracy by model.

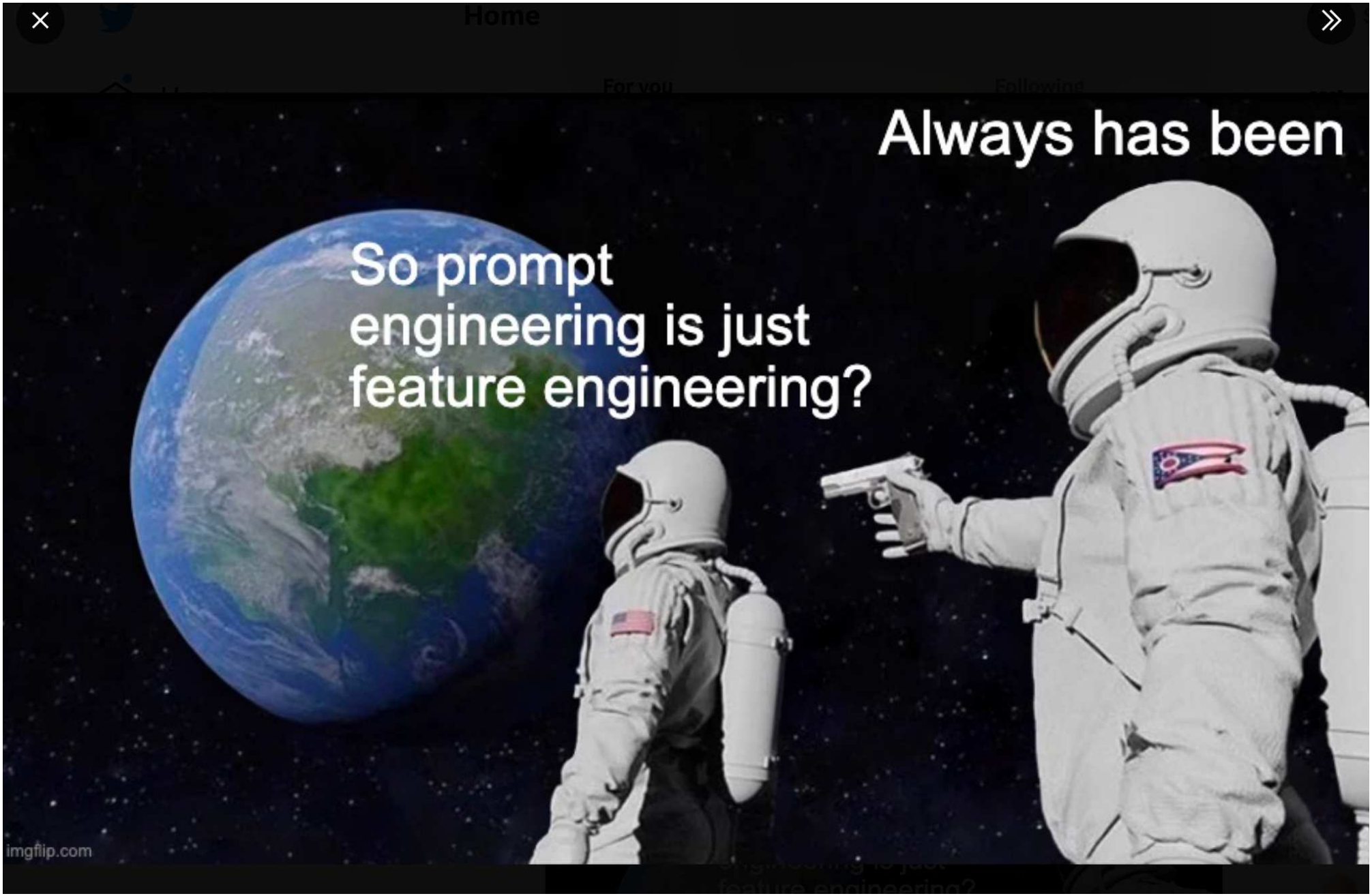
A Surprising Failure?
Multimodal LLMs and the NLVR Challenge

Anne Wu, Kianté Brantley, and Yoav Artzi
Department of Computer Science and Cornell Tech, Cornell University
{annewu, yoav}@cs.cornell.edu, kdb82@cornell.edu

Abstract

This study evaluates three state-of-the-art MLLMs — GPT-4V, Gemini Pro, and the open-source model IDEFICS — on the compositional natural language vision reasoning task NLVR. Given a human-written sentence paired with a synthetic image, this task requires the model to determine the truth value of the sentence with respect to the image. Despite the strong performance demonstrated by these models, we observe they perform poorly on NLVR, which was constructed to require compositional and spatial reasoning, and to be robust for semantic and systematic biases.







Mark Dredze
@mdredze

Fine-tuning = Further training

Fine-tuning:
train some more on in-domain data or separate labeled task

Task Fine-tuning

Train on more labeled data for the target task

Instruction Tuning

Train the model to follow different kinds of instructions

Reinforcement Learning

Tune the model with a reward function

Finetuning

Fine-tuning = Further training

Fine-tuning means further training a model. For instance, if we want to turn a base pretrained LLM into a sentiment analysis model, we might further train it on labeled sentiment data (like our NYT Cooking reviews).

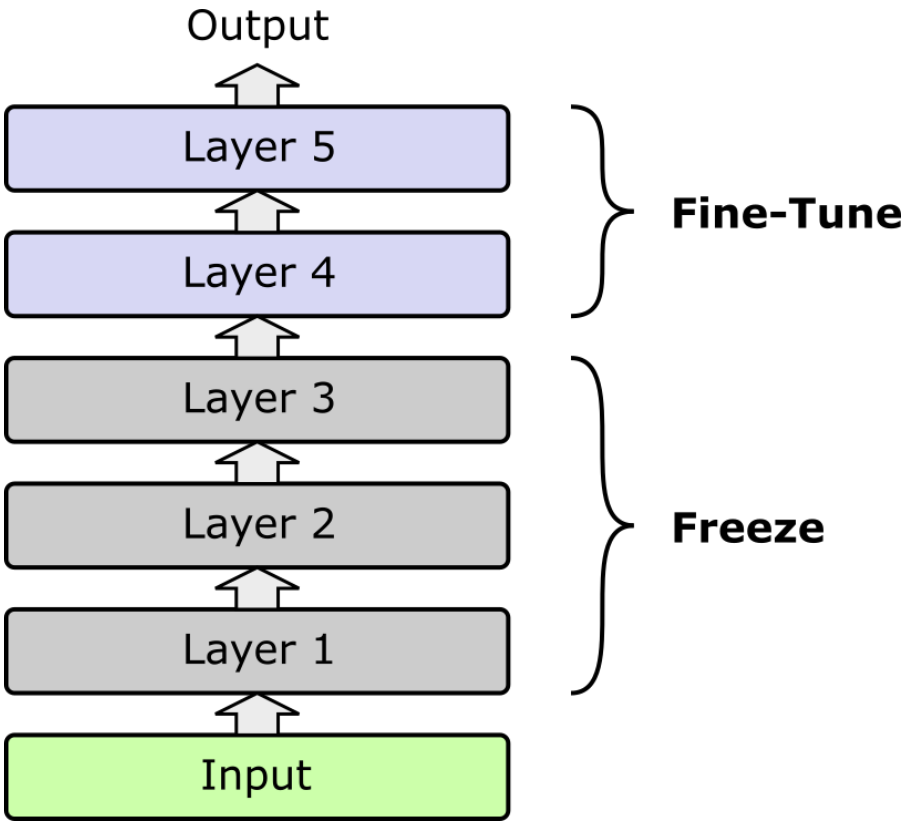
This is useful both to teach it **task-specific information** and to teach it about the **task format**.

Fine-tuning = Further training

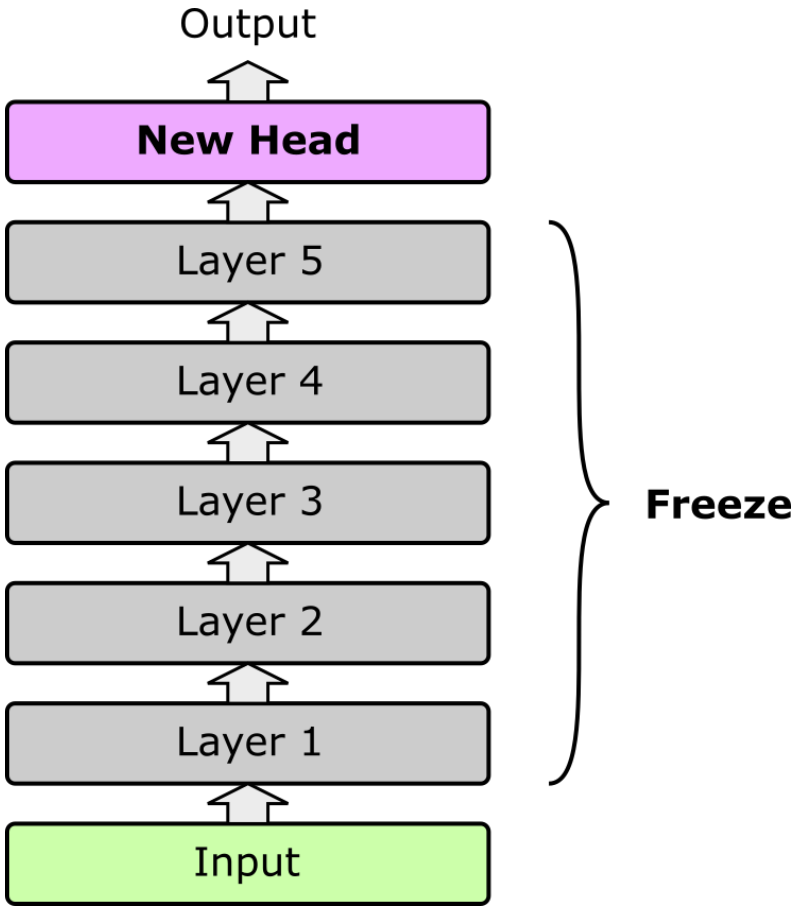
We have options about how much of the base model we further train:

- Full fine-tuning: all parameters of the base model continue to get updated
- Partial fine-tuning: we **freeze** some of the parameters of the model and train the rest
- Parameter-efficient fine-tuning: we **insert** some small number of new weights into the model to train while freezing the rest.

Partial Fine-tuning

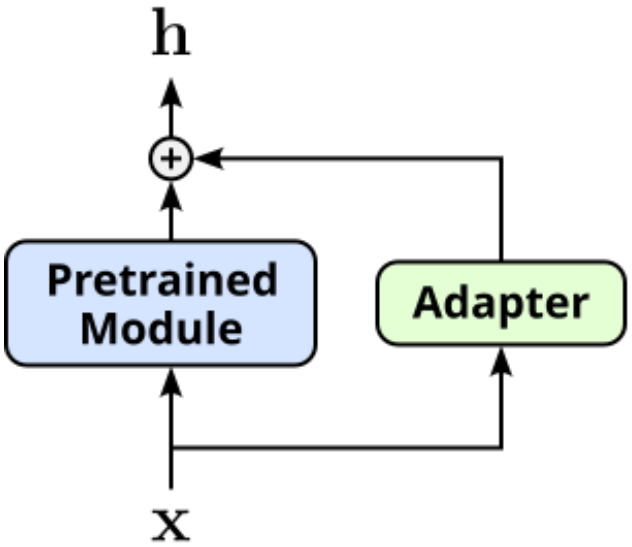


partial

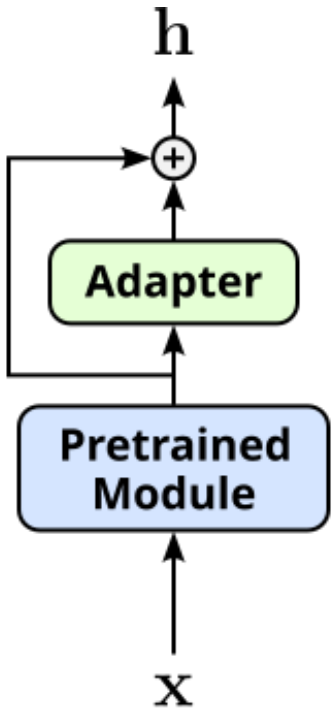


partial + new head

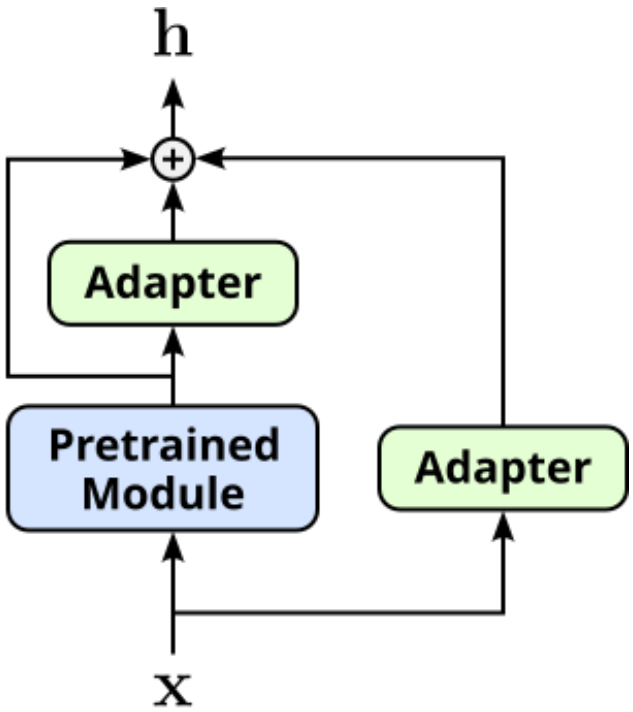
Parameter-Efficient Fine-tuning



Parallel Adapter



Sequential Adapter



Hybrid Adapter

Credit: https://github.com/chrisvdweth/selene/blob/master/notebooks/llm_model_fine_tuning_overview.ipynb

Reinforcement Learning

We'll cover this next class!

Evaluating LLMs

Evaluating LLMs

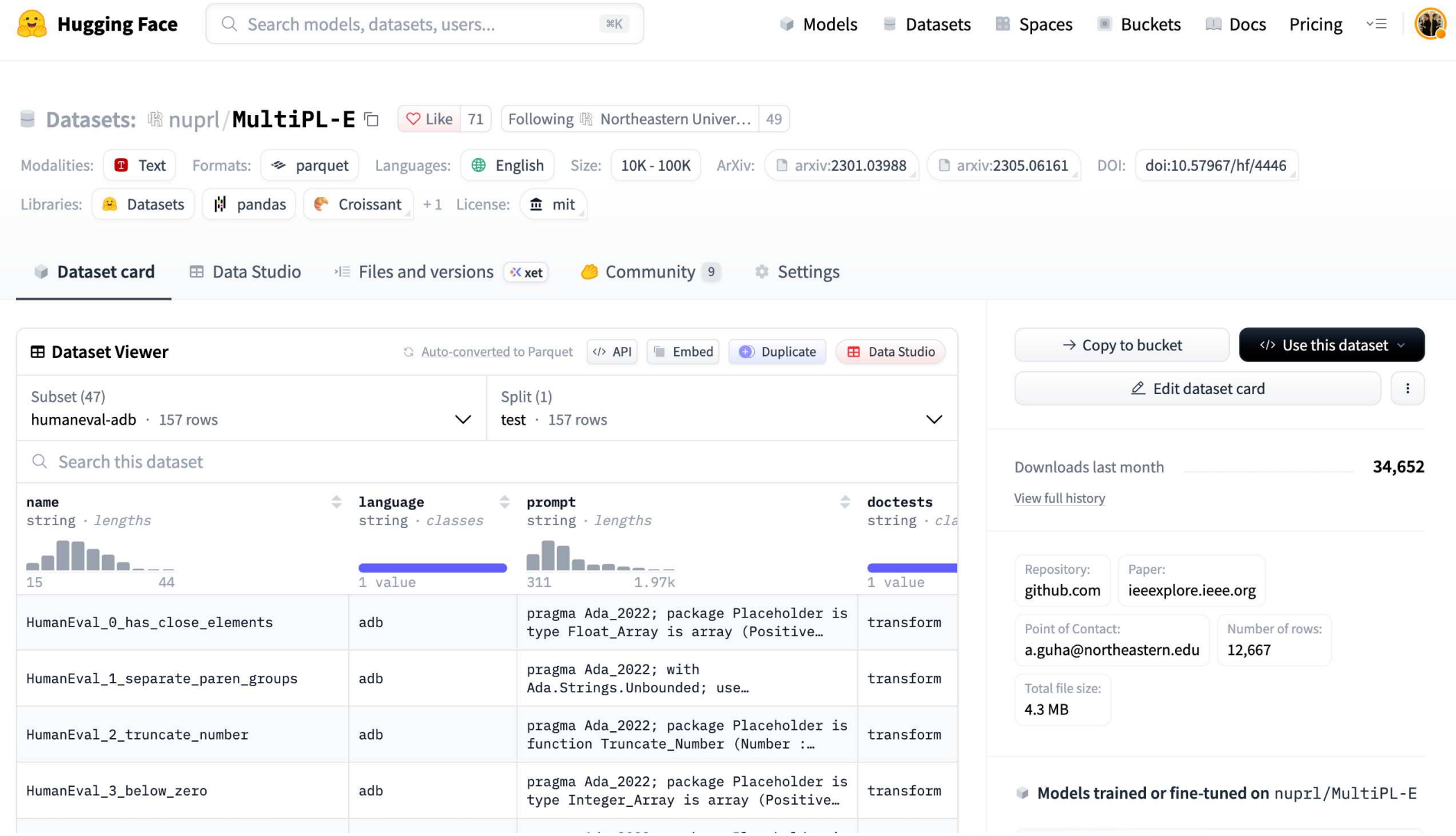
Contemporary LLMs are general text-processing machines that can be used to solve a variety of tasks, so long as they can be described in text.

How do we evaluate their abilities?

Benchmark Evaluations

- The most common way to evaluate an LLM is to run it on a specially designed evaluation dataset called a ***benchmark***.
- Usually, a benchmark targets a specific task or set of related tasks.
- Often, the same benchmark is used to test many LLMs, and the results are presented as a ***leaderboard*** ranking models by their performance.

Example: MultiPL-E



Example: Big Code Models Leaderboard

⭐ Big Code Models Leaderboard

Inspired from the 🤗 Open LLM Leaderboard and 🤗 Open LLM-Perf Leaderboard 🏋️, we compare performance of base multilingual code generation models on HumanEval benchmark and MultiPL-E. We also measure throughput and provide information about the models. We only compare open pre-trained multilingual code models, that people can start from as base models for their trainings.

🔍 Evaluation table | 📊 Performance Plot | 📝 About | Submit results 🚀

➡️ See All Columns ▼

☑ Win Rate ☐ Average score ☑ humaneval-python ☑ java ☑ javascript ☐ Throughput (tokens/s) ☑ cpp ☐ php ☐ rust ☐ swift
☐ r ☐ lua ☐ d ☐ racket ☐ julia ☐ #Languages ☐ Throughput (tokens/s) bs=50 ☐ Peak Memory (MB) ☐ Seq_length ☐ Links
☐ Submission PR

🔍 Search for your model and press ENTER...

⏚ Filter model types

◉ all ○ 🟢 base ○ 🔶 instruction-tuned ○ EXT external-evaluation

T	Model	Win Rate	humaneval-python	java	javascript	cpp
🔶 EXT	Qwen2.5-Coder-32B-Instruct	59.17	83.2	73.69	76.05	81.95
🟢 EXT	Qwen2.5-Coder-32B	56.67	57.1	65.49	65.07	64.35
🔶 EXT	OpenCodeInterpreter-DS-33B	56.25	75.23	54.8	69.06	64.47
🔶 EXT	Nxcode-CQ-7B-orpo	55.92	87.23	60.91	71.69	68.04

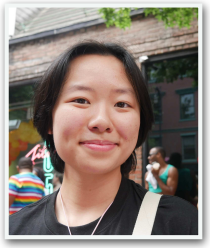
Benchmark Evaluations

Benchmarks typically use a single, expert-written prompt to describe the task to the model. Success is measured as the ability to complete the task.

Benchmark Evaluations

However, some of my work has argued that good models should:

- Be robust to differences in how the task is described: StudentEval (ACL Findings 2024)
- Be efficient with their compute when solving tasks: The Best Programming Language for Tokenmaxxing (under review)



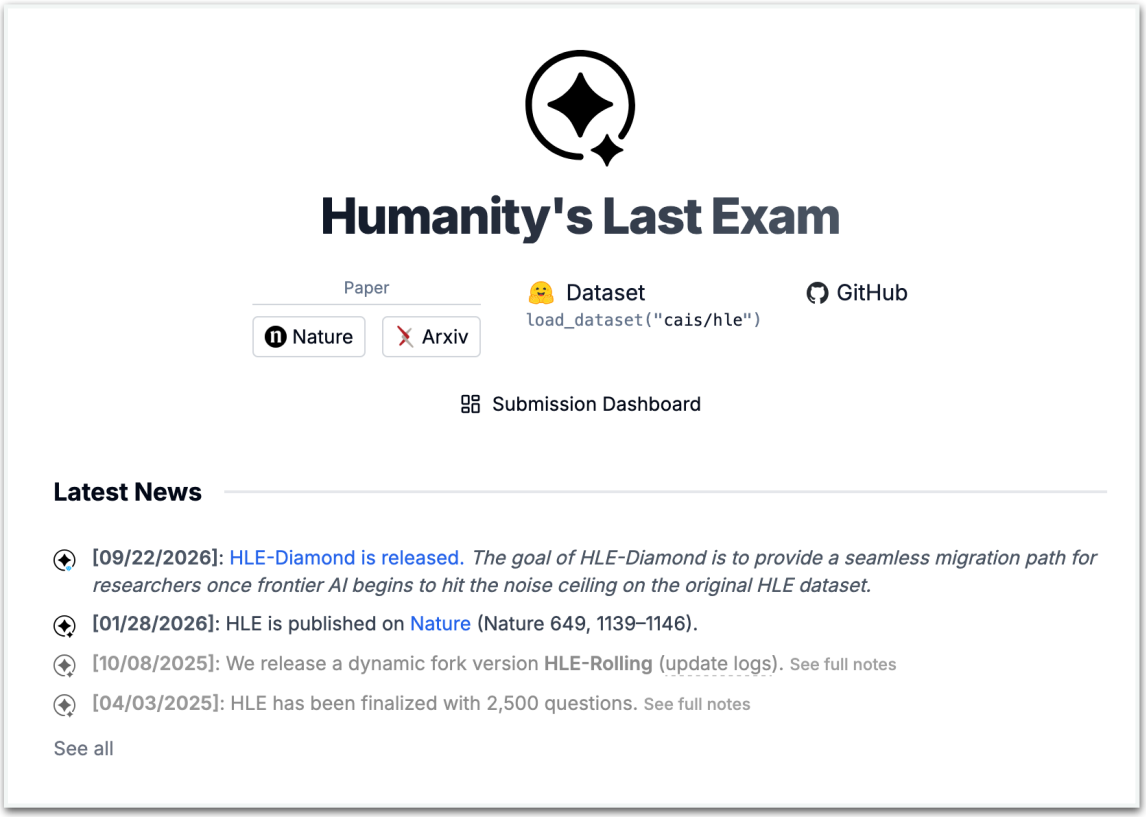
Sydney Nguyen
Class of 2024



Zixuan Wu
Class of 2025

Benchmark Evaluations

As LLM abilities have improved, benchmark tasks have needed to become more challenging. This has led to an increased focus on very challenging tasks posed by experts in a particular field.



Benchmark Evaluations

I've argued that we should also have benchmarks that are challenging for models but easy for humans to verify:

ReasoningWeekly: A General Knowledge and Verbal Reasoning Challenge for Large Language Models (AACL 2025)

Challenge:

These four words have a very interesting and unusual property in common — something hidden in them. What is it?

- *NEANDERTHAL*
- *EMBARRASS*
- *SATURATION*
- *CONTEMPTUOUSNESS*



Zixuan Wu
Class of 2025



Jingmiao Zhao
Class of 2023

Evaluating LLMs

- Benchmark evaluations only capture so much about a model's performance. It's also important to do human evaluation and user testing.
- A popular way to do this is through ***model arenas***. This is similar to what you did in HW 7!

Example: Chatbot Arena

