10-423/623: Generative AI Lecture 1 – Recurrent Neural Networks and Language Modelling

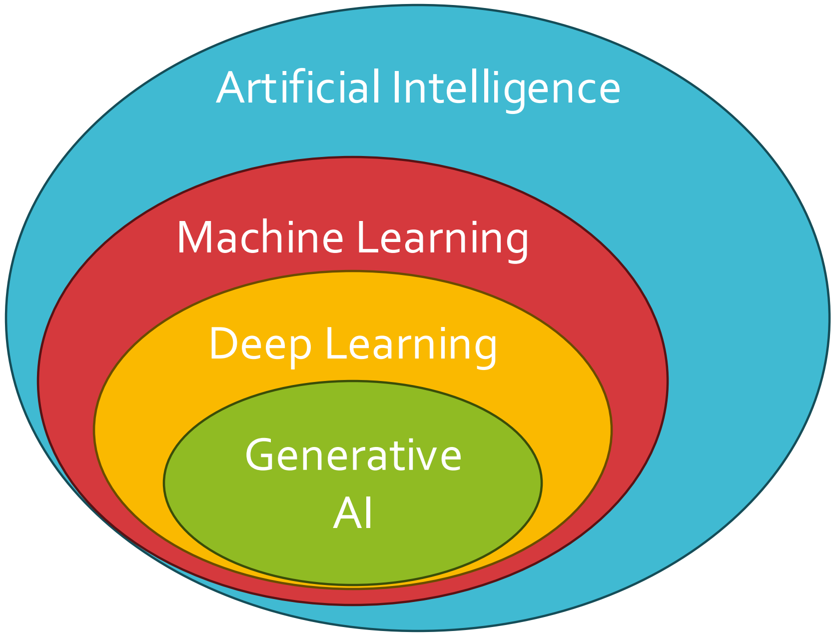
Henry Chai & Matt Gormley

8/26/24

What is Generative AI?

What is AI?

- The field of artificial intelligence is concerned with developing computer systems that mimic (or exceed) human behavior
- Sub-goals:
  - Perception
  - Reasoning
  - Planning
  - Communication
  - Creativity
  - Control / Motion / Manipulation
  - Learning



What is AI?

- The field of artificial intelligence is concerned with developing computer systems that mimic (or exceed) human behavior
- Sub-goals:
  - Perception
  - Reasoning
  - Planning
  - Communication
  - Creativity
  - Control / Motion / Manipulation
  - Learning

- Large language models are capable (to varying degrees) or comprehending and responding to natural language

What is AI?

- The field of artificial intelligence is concerned with developing computer systems that mimic (or exceed) human behavior
- Sub-goals:
  - Perception
  - Reasoning
  - Planning
  - Communication
  - Creativity
  - Control / Motion / Manipulation
  - Learning

- Text-to-image models (e.g., Midjourney) or text-to-music models (MusicGen)

What is AI?

- The field of artificial intelligence is concerned with developing computer systems that mimic (or exceed) human behavior
- Sub-goals:
  - Perception
  - Reasoning
  - Planning
  - Communication
  - Creativity
  - Control / Motion / Manipulation
  - Learning

- Multimodal models can answer natural language questions about images and perform zero-shot classification

What is AI?

- The field of artificial intelligence is concerned with developing computer systems that mimic (or exceed) human behavior
- Sub-goals:
  - Perception
  - Reasoning
  - Planning
  - Communication
  - Creativity
  - Control / Motion / Manipulation
  - Learning

- LLMs have opened new possibilities in how we approach the learning problem e.g., in-context learning

What is AI?

- The field of artificial intelligence is concerned with developing computer systems that mimic (or exceed) human behavior
- Sub-goals:
  - Perception
  - Reasoning
  - Planning
  - Communication
  - Creativity
  - Control / Motion / Manipulation
  - Learning

- LLMs are decent at certain reasoning tasks using e.g., chain-of-thought prompting

What is AI?

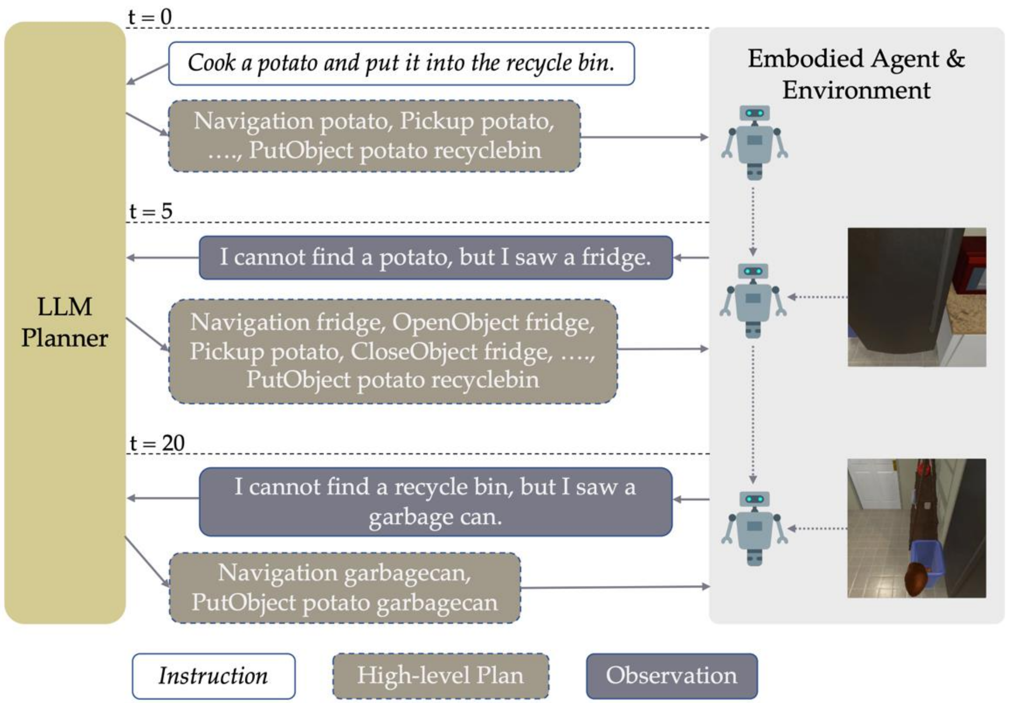
- The field of artificial intelligence is concerned with developing computer systems that mimic (or exceed) human behavior
- Sub-goals:
  - Perception
  - Reasoning
  - Planning
  - Communication
  - Creativity
  - Control / Motion / Manipulation
  - Learning

- LLM-Planner: a modified LLM that can help embodied agents (i.e., robots) perform complex, multi-step tasks by generating high-level plans

Source: https://arxiv.org/pdf/2212.04088

What is AI?

- The field of artificial intelligence is concerned with developing computer systems that mimic (or exceed) human behavior
- Sub-goals:
  - Perception
  - Reasoning
  - Planning
  - Communication
  - Creativity
  - Control / Motion / Manipulation
  - Learning



Source: https://arxiv.org/pdf/2212.04088

What is AI?

- The field of artificial intelligence is concerned with developing computer systems that mimic (or exceed) human behavior
- Sub-goals:
  - Perception
  - Reasoning
  - Planning
  - Communication
  - Creativity
  - Control / Motion / Manipulation
  - Learning

- DayDreamer: learns a ”World Model” for reinforcement learning without the need for simulated data; allows for rapid training of RL agents

Source: https://arxiv.org/pdf/2206.14176

What is AI?

- The field of artificial intelligence is concerned with developing computer systems that mimic (or exceed) human behavior
- Sub-goals:
  - Perception
  - Reasoning
  - Planning
  - Communication
  - Creativity
  - Control / Motion / Manipulation
  - Learning

The ability to create or generate novel ideas/concepts (to “imagine”) is central to how humans successfully perform these tasks!

What is Generative AI capable of?

What is Generative AI capable of?

Teach me about narwhals in the style of a Shakespearean sonnet

Behold the narwhal, monarch of the sea,
With spiral tusk that doth the waves adorn,
A unicorn of ocean's mystery,
By nature's hand so wondrously born.

Beneath the icy realms where cold winds blow,
This creature glides through waters dark and deep,
Its ivory lance, a silent, graceful show,
A secret kept where ancient legends sleep.

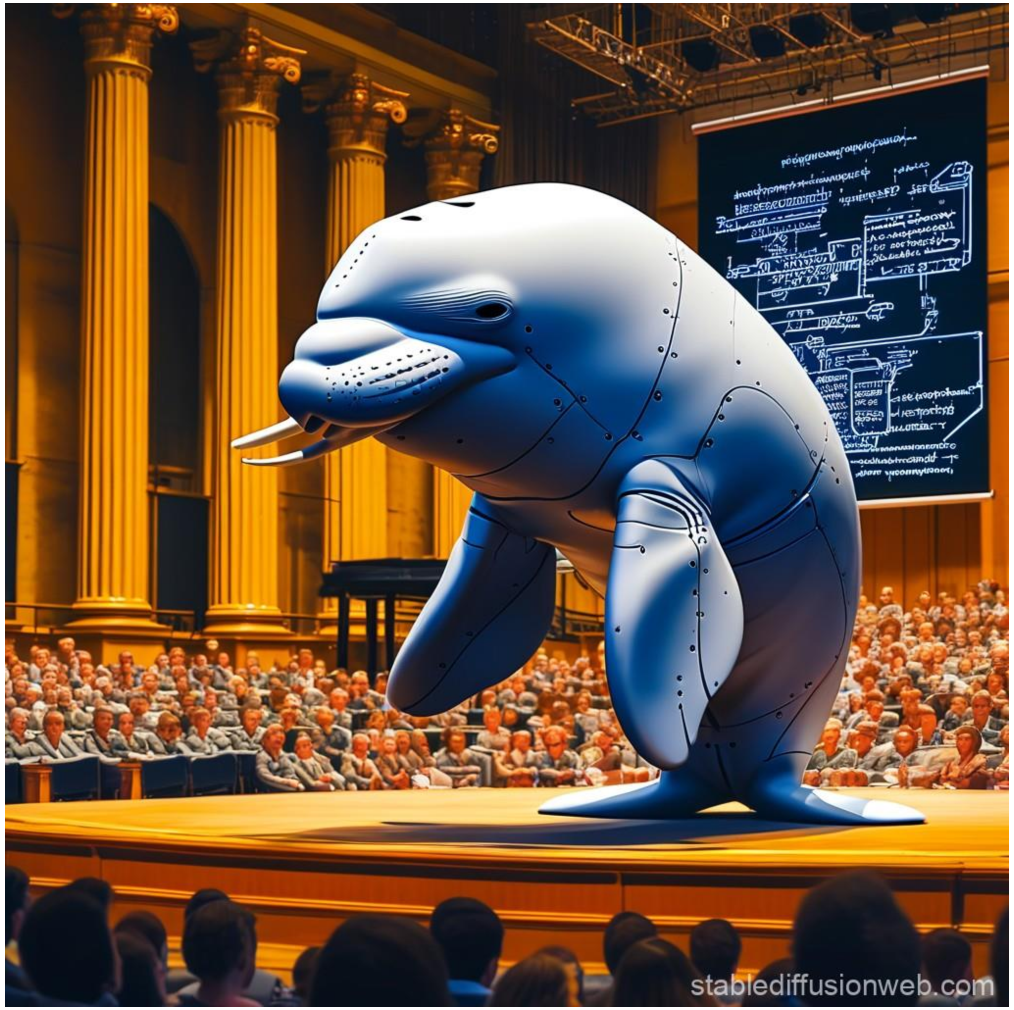
Yet not for battle is this horn designed,
But rather, like a herald in the night,
It senses all that swims or stirs behind,
Guiding its path through realms devoid of light.

O narwhal, gentle giant of the brine,
In thee, both strength and beauty intertwine.

Source: https://chat.openai.com/

What is Generative AI capable of?

Prompt: a picture of a narwhal giving a lecture on generative AI. Put the narwhal on a stage in front of a large audience. The narwhal should appear scholarly and is presenting off of a slide deck, gesturing at some dense equations on the slide



Source: https://stablediffusionweb.com/app/image-generator

What is Generative AI capable of?

Source: https://huggingface.co/stabilityai/stable-video-diffusion-img2vid-xt

What is Generative AI capable of?

Whimsical Indie Folk acoustic guitar and ukulele accompaniment, playful and quirky, with a laid-back tempo around 90 BPM.

Epic cinematic score with a heavy, pulsing bassline and energetic trap beats, blending elements of electronic and orchestral music, in a 140-150 BPM tempo.

Source: https://deepai.org/music

What is Generative AI capable of?

GPT-4 vs ChatGPT

Prompt:

```
I want to write a customized optimizer in pytorch, the optimizer should do:
1. the parameters for the optimizer is learning rate, momentum ,weigth decay, d_dim, k and factor alpha
2. the optimizer goes through each parameter in the network, and then
-- Get the gradient of this parameter, reshape it to 2D by looping through the dimensions, and keeping the dimension whose actual
dimension (shape) is equal to d_dim, do an SVD decomposition of the parameter so W = U\Sigma V.
-- Keep the top k singular vectors of W, so we get a new matrix W_k = U \Sigma_k V.
-- Keep the top 2k singular vectors of W, so we get a new matrix W_{2k} = U \Sigma_{2k} V.
-- Normalize W_k according to the momentum of the F-norm of W_{2k}.
-- Truncate each coordinate of W_k to the top alpha  percentile of all the coordinates, in absolute value.
-- Apply momentum on W_k, and then update the network parameters using this momentum.
-- Apply weight decay.
```

GPT-4:

```
def step(self):
    for group in self.param_groups:
        for p in group['params']:
            if p.grad is None:
                continue
            grad = p.grad.data
            # reshape grad to 2D
            grad_2d = grad.reshape(-1, group['d_dim'])
            # SVD decomposition
            U, S, V = torch.svd(grad_2d)
            # keep top k singular vectors
            G_k = U[:, :group['k']] @ torch.diag(S[:group['k']]) @
            V[:, :group['k']].T
            # keep top 2k singular vectors
            G_2k = U[:, :2*group['k']] @ torch.diag(S[:2*group['k']])
            @ V[:, :2*group['k']].T
            # normalize G_k
            G_k /= torch.norm(G_2k, p=2)
            # truncate G_k
            threshold = torch.quantile(torch.abs(G_k),
            group['alpha'])
            G_k[torch.abs(G_k) < threshold] = 0
            # apply momentum
            if 'momentum_buffer' not in group:
                group['momentum_buffer'] = torch.zeros_like(G_k)
            group['momentum_buffer'] = group['momentum'] *
            group['momentum_buffer'] + G_k
            # update parameters
            p.data.add_(-group['lr'] * group['momentum_buffer'])
            # apply weight decay
            if group['weight_decay'] != 0:
                p.data.add_(-group['weight_decay'] * group['lr'],
                p.data)
```

ChatGPT:

```
def step(self):
    for group in self.param_groups:
        for p in group['params']:
            if p.grad is None:
                continue
            grad = p.grad.data
            grad = grad.reshape(-1, grad.shape[-1]) # reshape to 2D

            # SVD decomposition
            u, s, v = torch.svd(grad)

            # Keep top k and 2k singular vectors
            k = group['k']
            u_k = u[:,:k]
            s_k = s[:k]
            v_k = v[:,:k]
            u_2k = u[:,:2*k]
            s_2k = s[:2*k]
            v_2k = v[:,:2*k]

            # Normalize according to momentum of F-norm of G_2k
            grad_k = u_k @ torch.diag(s_k) @ v_k.t()
            norm_2k = s_2k.norm()
            grad_k = grad_k * group['momentum'] / norm_2k

            # Truncate to top alpha percentile
            alpha = group['alpha']
            percentile = grad_k.abs().flatten().kthvalue(int(alpha *
            grad_k.numel()))
            grad_k[grad_k.abs() < percentile] = 0

            # Apply weight decay
            p.data.add_(-group['lr'], grad_k + group['weight_decay']
            * p.data)
```

Figure 3.4: GPT-4 vs ChatGPT at implementing a customized Pytorch optimizer module. We highlight the astuteness of GPT-4 vs the mistake by ChatGPT at 'applying momentum'.

Source: https://arxiv.org/pdf/2303.12712.pdf

What is Generative AI capable of?

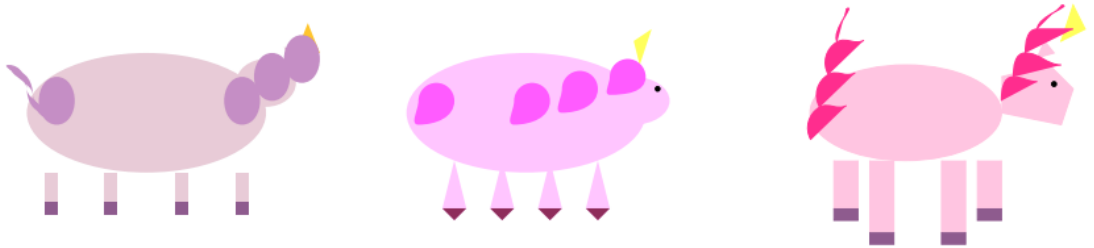


Figure 1.3: We queried GPT-4 three times, at roughly equal time intervals over the span of a month while the system was being refined, with the prompt "Draw a unicorn in TikZ". We can see a clear evolution in the sophistication of GPT-4's drawings.

GPT-4 vs ChatGPT

Prompt:

```
A function f is called {increasing}_{emphasize!} if f(s) <= f(t) holds for every s <= t. Consider the formula:
limit_{x goes to 0} {limit when n approaches infinite} underscore{
\matrix[ [ sum of i from 1 to n (f(nx) - integral from 0 to x sin(nx) dx ),
((x + 2/8)^2 - 5)/((4x-1/x^2) /(n xor 3)), \\ , ..., partial f(x/n)
over partial x * (e^{-i*omega n x} + 3(x#7) ) , f inverse of (x mod n) ]]
multiply \matrix{1, 1, ..., 1 \\ 1, 1, ..., 1} }
{grey_color{this gives us a good impression about the function}}

Convert the above paragraph to LaTeX.
```

GPT-4:

A function f is called *increasing* if $f(s) \leq f(t)$ holds for every $s \leq t$. Consider the formula:

$$\underbrace{\lim_{x\to 0}\lim_{n\to\infty}\begin{bmatrix}\sum_{i=1}^{n}\left(f(nx)-\int_0^x \sin(nx)\,dx\right), & \frac{\frac{(x+\frac{2}{8})^2-5}{4x-1/x^2}}{n\oplus 3}, \\ \vdots & \vdots \\ \frac{\partial f(x/n)}{\partial x}\cdot\left(e^{-i\omega n x}+3(x\#7)\right), & f^{-1}(x \bmod n)\end{bmatrix}\times\begin{bmatrix}1 & 1 & \cdots & 1\\ 1 & 1 & \cdots & 1\end{bmatrix}}_{\text{this gives us a good impression about the function}}$$

Source: https://arxiv.org/pdf/2303.12712.pdf

Why is Generative AI worth understanding deeply?

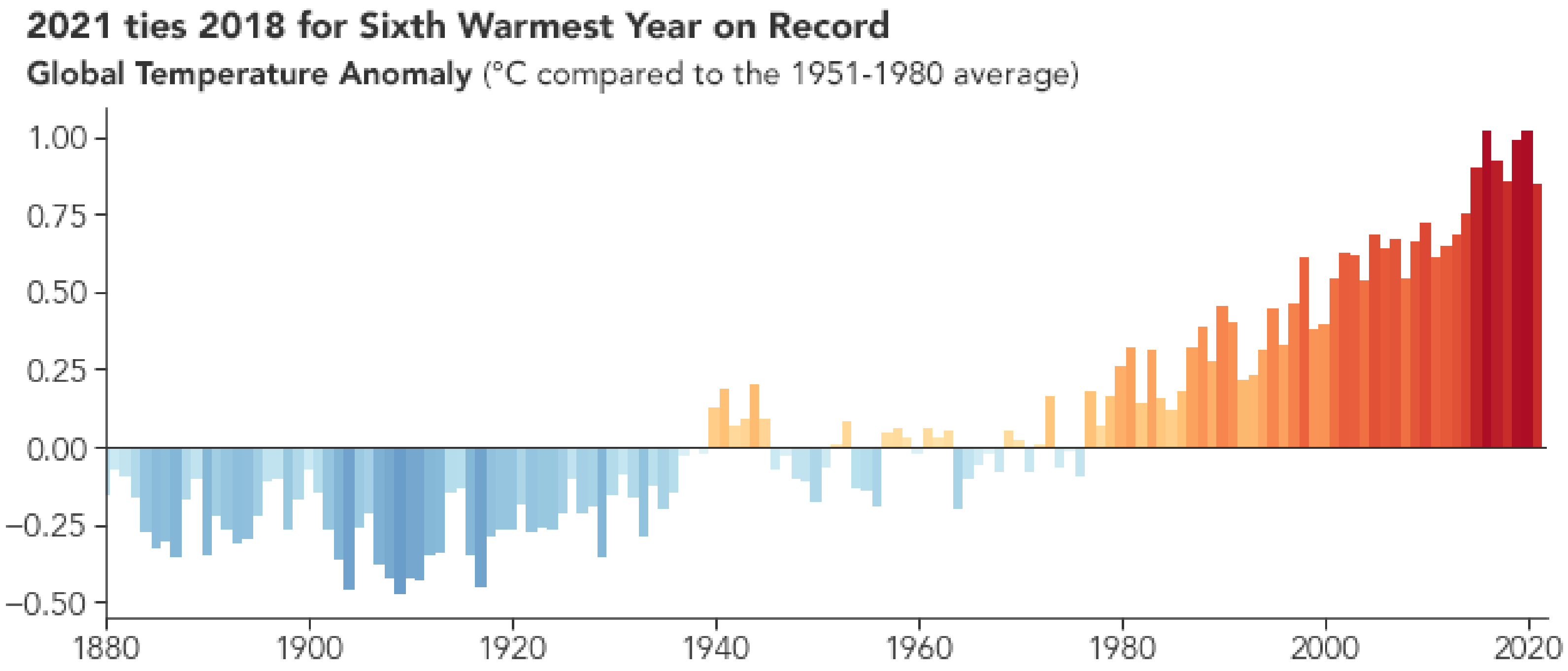
Why is Generative AI worth understanding deeply?

Source: https://wallpaperaccess.com/vw-bus

Why is Generative AI worth understanding deeply?

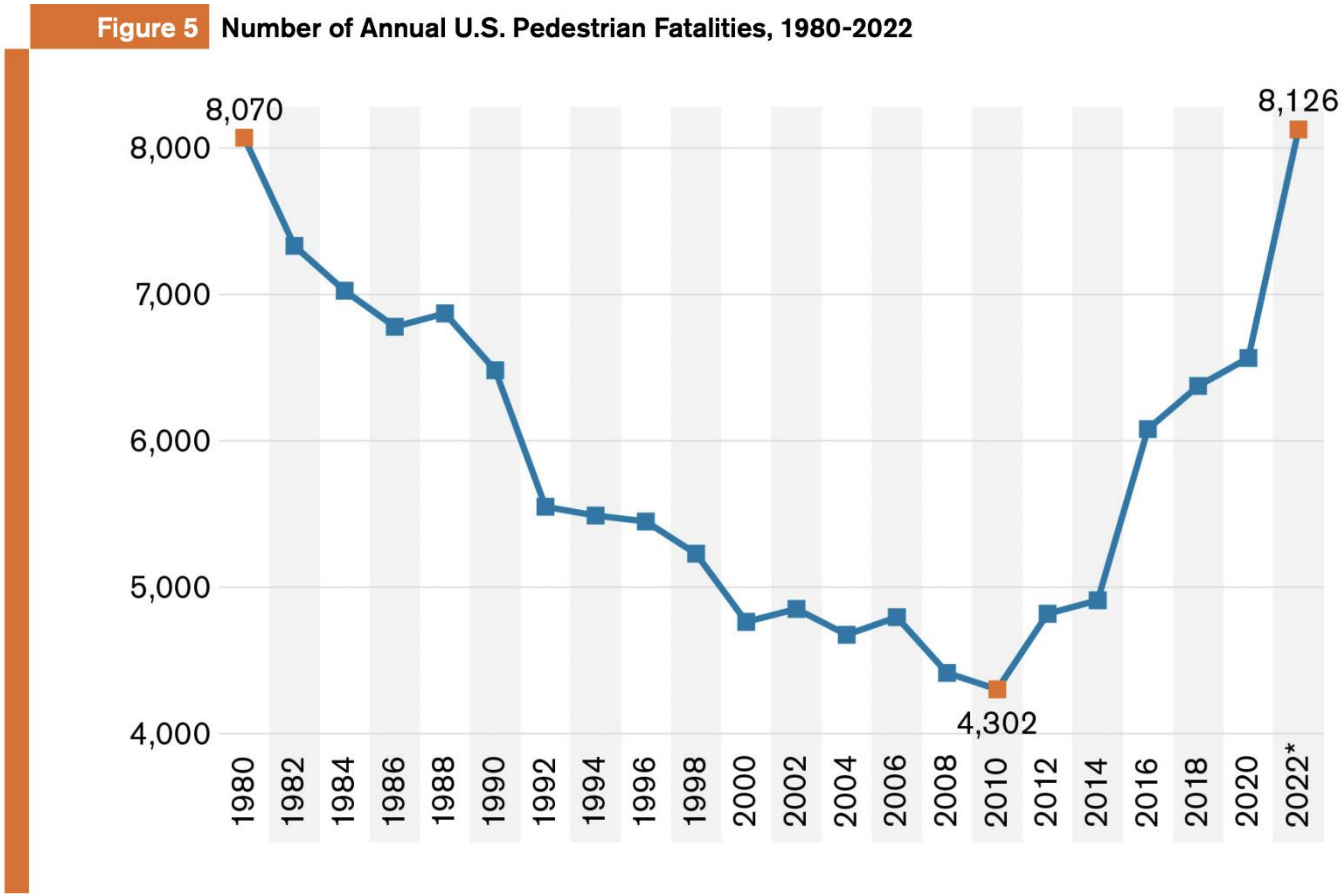
Source: https://daily.jstor.org/the-science-of-traffic/

Why is Generative AI worth understanding deeply?



Source: https://earthobservatory.nasa.gov/images/149321/2021-continued-earths-warming-trend

Why is Generative AI worth understanding deeply?



Source: https://www.ghsa.org/sites/default/files/2023-06/GHSA%20-%20Pedestrian%20Traffic%20Fatalities%20by%20State%2C%202022%20Preliminary%20Data%20%28January-December%29.pdf

Why is Generative AI worth understanding deeply?

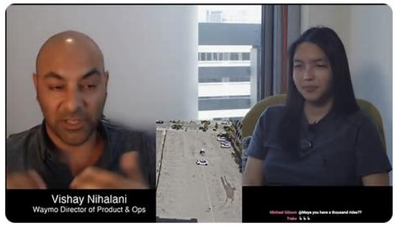
Source: https://www.bing.com/images/create/futuristic-self-driving-electric-cars2c-cinematic/1-65a81fced86249d186602d396671a605?id=giV09dC3LyVTiXb86uHpug%3d%3d&view=detailv2&idpp=genimg&FORM=GCRIDP&mode=overlay

Why is Generative AI worth understanding deeply?

engadget · 5d · on MSN

Waymo director says the company's cars won't honk at each other anymore

Waymo's self-driving cars no longer honk when near each other, Waymo's Director of Product and Operations Vishay Nihalani ...



ars TECHNICA · 12d

Self-driving Waymo cars keep SF resident honking at each other

Silicon Valley's latest disruption? Your sleep schedule. On reported that San Francisco's South of ...

+11 relevant news

NEWS · 13d · on MSN

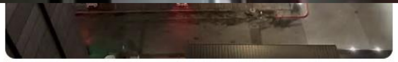
Waymo cars honk at each other throughou SF neighbors

San Francisco resident Randol White says he heard the n weeks ago -- he was woken up around ...

Los Angeles Times · 13d · on MSN

Driverless Waymo taxis disturb peace, sle honking — at each other

Waymo's robotaxis have started occupying a new San Fra residents are finding the self-driving cars ...



yahoo! · 5d

Driverless Waymo cars still honking despite software fix

Source: https://www.bing.com/news/search?q=waymo+cars+honking+at+each+other&FORM=HDRSC7

What is ~~Generative AI~~ 10-423/623: Topics

- Generative models of text
  - RNN LMs / Autodiff
  - Transformer LMs
  - Pre-training, fine-tuning, evaluation, decoding
- Generative models of images
  - CNNs / Transformers for vision
  - GANs, Conditional GANs
  - VAEs and Diffusion models
- Applying and adapting foundation models
  - Reinforcement learning with human feedback (RLHF)
  - Parameter-efficient fine tuning
  - In-context learning for text
  - In-context learning for vision
- Multimodal foundation models
  - Text-to-image generation
  - Aligning multimodal representations
  - Visual-language foundation models
- Scaling models
  - Efficient decoding strategies
  - Distributed training
  - Scaling laws and data
- What can go wrong?
  - Safety/bias/fairness
  - Hallucinations
  - Adversarial attacks
  - Cheating – how to watermark, Legal issues, e.g., copyright,...
  - Drift in performance
  - Data contamination
- Advanced Topics
  - State space models
  - Code generation
  - Audio understanding and synthesis
  - Video synthesis

What is ~~Generative AI~~ 10-423/623: Website

- https://www.cs.cmu.edu/~mgormley/courses/10423/
- Also http://423.mlcourse.org or http://623.mlcourse.org

What is ~~Generative AI~~ 10-423/623: Syllabus

- https://www.cs.cmu.edu/~mgormley/courses/10423/syllabus.html
- This entire page is **required** reading

What is ~~Generative AI~~ 10-423/623: Syllabus

- https://www.cs.cmu.edu/~mgormley/courses/10423/syllabus.html
- Assessments:
  - 40% homework
  - 10% quizzes
  - 20% exam
  - 25% project
  - 5% participation

What is ~~Generative AI~~ 10-423/623: Homework

- https://www.cs.cmu.edu/~mgormley/courses/10423/coursework.html

HW #	Topic	Model	Application	Type
0	PyTorch Primer	image classifier + text classifier	Vision + language	written + programming
1	LLMs	Transformer with RoPE and GQA	Charater-level text generation	written + programming
2	Image Generation	Diffusion model	Infilling	written + programming
3	Adapters for LLMs	Llama with LoRA	Code	written + programming
4	Multimodal Foundation Models	Text-to-image model	Vision *and* language	written + programming
623 **(623 students only)**	Recent Research	N/A	Generative AI	video presentation

What is ~~Generative AI~~ 10-423/623: Homework

- https://www.cs.cmu.edu/~mgormley/courses/10423/coursework.html

Home
FAQ
Syllabus
People
Schedule
Office Hours
Coursework
Links

Generative AI

10-423 + 10-623, Fall 2024
School of Computer Science
Carnegie Mellon University

Assignments

There will be 5 homework assignments (and a special extra assignment for 10-623 only). The assignments will consist of both theoretical and programming problems. Homework assignments will be released via a Piazza announcement explaining where to find the handout, LaTeX template, etc.

- Homework 0: PyTorch Primer
- Homework 1: Large Language Models
- Homework 2: Image Generation
- Homework 3: Applying and Adapting LLMs
- Homework 4: Multimodal Foundation Models
- Homework 623: (10-623 only)

What is ~~Generative~~ ~~AI~~ 10-423/623: Quizzes

- https://www.cs.cmu.edu/~mgormley/courses/10423/coursework.html

Home
FAQ
Syllabus
People
Schedule
Office Hours
Coursework
Links

Quizzes

There will be 5 quizzes.

- Quiz 1 (Lectures 1 - 3)
- Quiz 2 (Lectures 4 - 8)
- Quiz 3 (Lectures 9 - 11)
- Quiz 4 (Lectures 12 - 15)
- Quiz 5 (Lectures 16 - 20)

- Quizzes will be held in-class, before the start of lecture

What is ~~Generative~~ ~~AI~~ 10-423/623: Quizzes

- https://www.cs.cmu.edu/~mgormley/courses/10423/schedule.html

- Home
- FAQ
- Syllabus
- People
- Schedule
- Office Hours
- Coursework
- Links

Important Notes

This schedule is **tentative** and subject to change. Please check back often.

Tentative Schedule

Date	Lecture	Readings	Announcements
	Generative models of text		
Mon, 26-Aug	Lecture 1 : RNN LMs / Autodiff [slides]	• *Sequence Modeling: Recurrent and Recursive Nets*. Ian Goodfellow, Yoshua Bengio, & Aaron Courville (2016). Deep Learning, Chapter 10.1-10.5. • *A Framework for the Cooperation of Learning Algorithms*. Léon Bottou, Patrick Gallinari (1991). Advances in Neural Information Processing Systems. • *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. Paszke et al. (2019). Advances in Neural Information Processing Systems.	
Wed, 28-Aug	Lecture 2 : Transformer LMs	• *The Long Short-Term Memory and Other GatedRNNs*. Ian Goodfellow, Yoshua Bengio, & Aaron Courville (2016). Deep Learning, Chapter 10.10-10.12. • *Attention Is All You Need*. Vaswani et al. (2017). NeurIPS. • *The Illustrated Transformer*. Alammar (2018).	HW0 out
Fri, 30-Aug	Recitation: HW0		
Mon, 2-Sep	(Labor Day - No Class)		

What is ~~Generative AI~~ 10-423/623: Quizzes

- https://www.cs.cmu.edu/~mgormley/courses/10423/schedule.html

Home
FAQ
Syllabus
People
Schedule
Office Hours
Coursework
Links▾

Wed, 11-Sep	Lecture 5 : Encoder-only Transformers / Vision Transformers	• *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. Devlin et al. (2018). NAACL. • *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. Dosovitskiy et al. (2021). ICLR.	Quiz 1 (in-class, L1-L4)
Fri, 13-Sep	Recitation: HW1		
Mon, 16-Sep	Lecture 6 : Generative Adversarial Networks (GANs) / PGM	• *Generative Adversarial Nets*. Goodfellow et al. (2014). NeurIPS. • *NuerIPS 2016 Tutorial: Generative Adversarial Networks*. Goodfellow (2017). NeurIPS Tutorials.	
Wed, 18-Sep	Lecture 7 : Variational Autoencoders (VAEs) / Diffusion models (Part I)	• *Variational Inference: A Review for Statisticians*. Blei, Kucukelbir, and McAuliffe (2018). • *High-Level Explanation of Variational Inference*. Jason Eisner (2011). • *Tutorial on Variational Autoencoders*. Carl Doersch (2016).	
Fri, 20-Sep	(No Recitation)		
Mon, 23-Sep	Lecture 8 : Diffusion models (Part II)	• *Deep Unsupervised Learning using Nonequilibrium Thermodynamics*. Sohl-Dickstein et al. (2015). MLR. • *Denoising Diffusion Probabilistic Models*. Ho et al. (2020). NeurIPS.	HW1 due HW2 out (L4-L8)
Applying and adapting foundation models			
Wed, 25-Sep	Lecture 9 : In-context learning for text & for vision		Quiz 2 (in-class, L4-L8)

What is ~~Generative~~ ~~AI~~ 10-423/623: Exam

- https://www.cs.cmu.edu/~mgormley/courses/10423/schedule.html

Home
FAQ
Syllabus
People
Schedule
Office Hours
Coursework
Links

Mon, 4-Nov	Lecture 18 : Efficient decoding strategies		HW4 due HW623 out
Advanced Topics			
Wed, 6-Nov	Lecture 19 : Long Context in LLM		
Fri, 8-Nov	(No Recitation)		
Mon, 11-Nov	In-Class Exam		Pass/no pass & withdrawal deadline
Wed, 13-Nov	Lecture 20 : State Space Models		
Fri, 15-Nov	(No Recitation)		Project proposal due

What is ~~Generative AI~~ 10-423/623: Project

- https://www.cs.cmu.edu/~mgormley/courses/10423/schedule.html

Home
FAQ
Syllabus
People
Schedule
Office Hours
Coursework
Links ▾

Mon, 4-Nov	Lecture 18 : Efficient decoding strategies		HW4 due HW623 out
	Advanced Topics		
Wed, 6-Nov	Lecture 19 : Long Context in LLM		
Fri, 8-Nov	(No Recitation)		
Mon, 11-Nov	In-Class Exam		Pass/no pass & withdrawal deadline
Wed, 13-Nov	Lecture 20 : State Space Models		
Fri, 15-Nov	(No Recitation)		Project proposal due

What is ~~Generative AI~~ 10-423/623: Project

- https://www.cs.cmu.edu/~mgormley/courses/10423/schedule.html
- Apply generative AI to a problem / setting of your choice
- Report back to the class during a *poster session* to be held **sometime over finals period**
- You must complete the project in **groups of 3**

Prompt to ChatGPT-4o: *Create an image of three Scottish terriers in traditional Scottish outfits working collaboratively on a project for a generative AI course*

What is ~~Generative AI~~ 10-423/623: Syllabus

- https://www.cs.cmu.edu/~mgormley/courses/10423/syllabus.html
- Assessments:
  - 40% homework
  - 10% quizzes
  - 20% exam
  - 25% project
  - 5% participation
    - Participation will be based on a few (out-of-class) surveys/polls and some project deliverables

What is ~~Generative AI~~ 10-423/623: Prerequisites

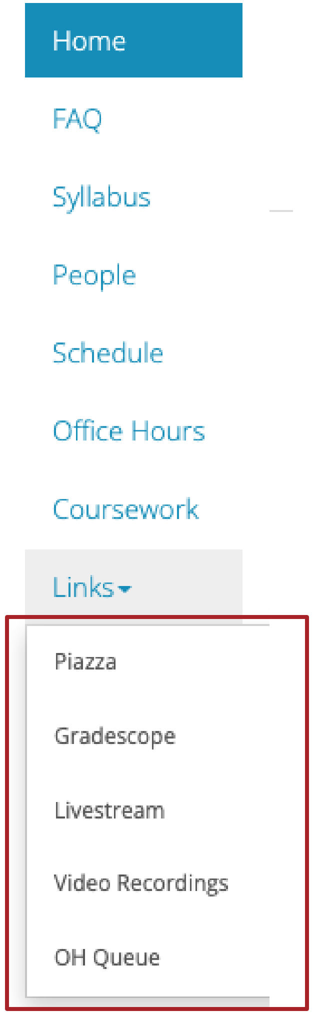
- https://www.cs.cmu.edu/~mgormley/courses/10423/syllabus.html
- You must have taken either:
  - an introduction to machine learning course (e.g., 10-301 / 601 or 10-315 or 10-701 or 10-715) *or*
  - an introduction to deep learning course (e.g., 11-485 / 685 / 785)
- You **do not** need to have familiarity with deep learning or PyTorch

What is ~~Generative AI~~ 10-423/623: Collaboration

- https://www.cs.cmu.edu/~mgormley/courses/10423/syllabus.html
- Collaboration is strongly encouraged in this course!
- But collaboration must always be documented
- You must always write your own code/solutions
  - No re-use of found code or code written for previous assignments
- Please see the syllabus for
  - recommended collaboration practices and
  - penalties for violations
- Note the 10-423/623 collaboration policy is different (more lenient) than our 10-301/601 policies

What is ~~Generative AI~~ 10-423/623: Technologies

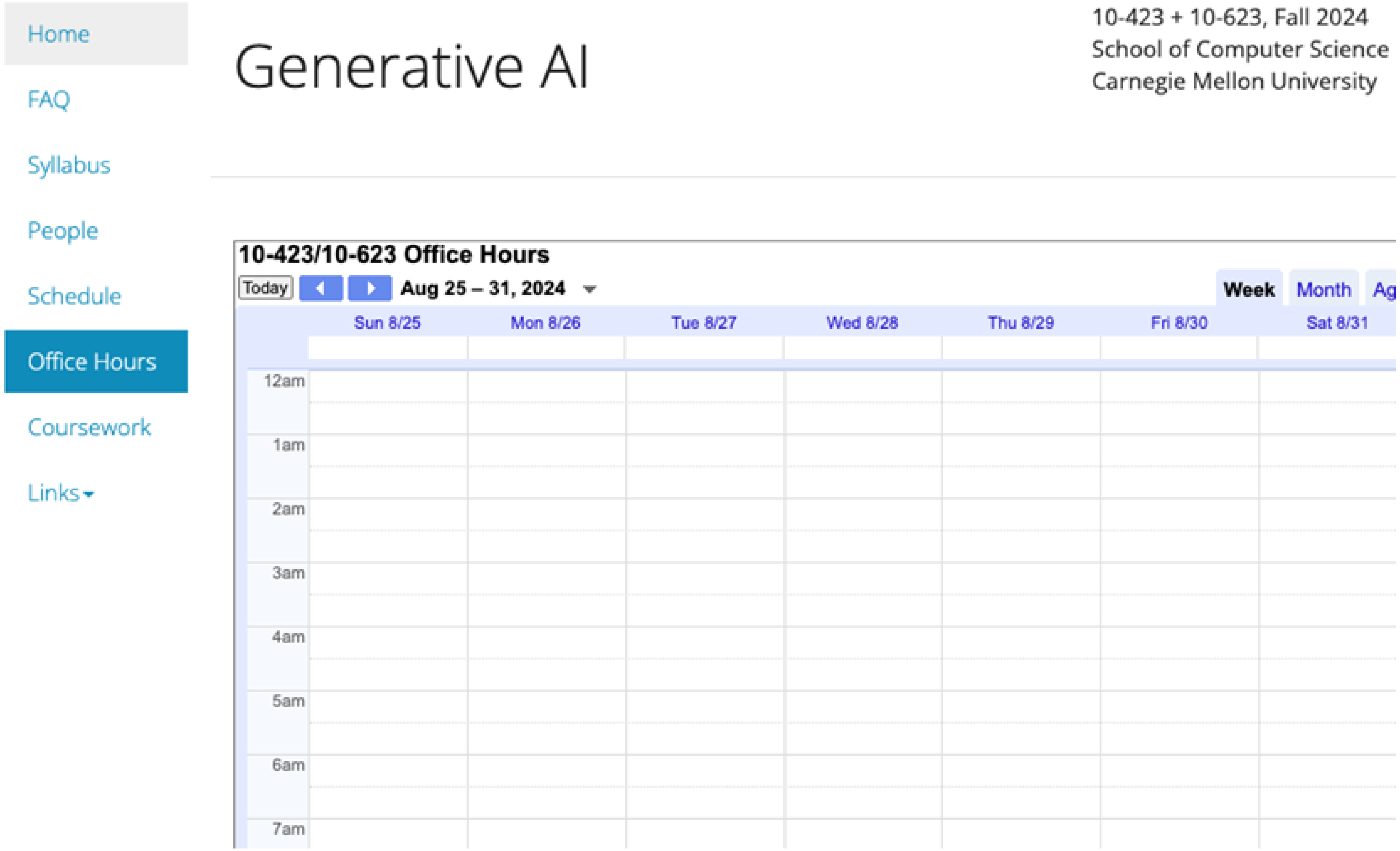
- https://www.cs.cmu.edu/~mgormley/courses/10423/



- Piazza (for Q&A)
- Gradescope (for submitting homework)
- Zoom (for livestreams)
- Panopto (for recordings)
- Google forms (for surveys/polls)

What is ~~Generative AI~~ 10-423/623: Office Hours

- https://www.cs.cmu.edu/~mgormley/courses/10423/officehours.html



What is ~~Generative AI~~ 10-423/623: Learning Objectives

You should be able to…

1. Differentiate between different mechanisms of learning such as parameter tuning and in-context learning.
2. Implement the foundational models underlying modern approaches to generative modeling, such as transformers and diffusion models.
3. Apply existing models to real-world generation problems for text, code, images, audio, and video.
4. Employ techniques for adapting foundation models to tasks such as fine-tuning, adapters, and in-context learning.
5. Enable methods for generative modeling to scale-up to large datasets of text, code, or images.
6. Use existing generative models to solve real-world discriminative problems and for other everyday use cases.
7. Analyze the theoretical properties of foundation models at scale.
8. Identify potential pitfalls of generative modeling for different modalities.
9. Describe societal impacts of large-scale generative AI systems.

What is Generative AI?

- Generative AI is ~~just~~ *modelling* and *sampling* from probability distributions that look like

$$p(x_{t+1}, x_t, \dots, x_1)$$

or

$$p(x_{t+1} | x_t, \dots, x_1)$$

Language Models

1. Convert raw text into a sequence of words (or “tokens”)

$$\boldsymbol{x} = [x_1, x_2, \dots, x_t]$$

2. Learn (or approximate) a joint probability distribution over sequences

$$p(\boldsymbol{x}) = p(x_1, x_2, \dots, x_t)$$

3. Sample from the implied conditional distribution to generate new words (or “tokens”)

$$p(x_{t+1} | x_1, x_2, \dots, x_t) = \frac{p(x_1, x_2, \dots, x_t, x_{t+1})}{p(x_1, x_2, \dots, x_t)}$$

   - In practice, it often makes more sense to learn (or approximate) this conditional distribution directly

Language Models

1. Convert raw text into a sequence of words (or "tokens")

$$\boldsymbol{x} = [x_1, x_2, \ldots, x_t]$$

2. Learn (or approximate) a joint probability distribution over sequences

$$p(\boldsymbol{x}) = p(x_1, x_2, \ldots, x_t)$$

- Use the chain rule of probability: predict the next word based on the previous words in the sequence

$$p(x) = p(x_1) \prod_{i=2}^{t} p(x_i \mid x_{i-1}, \ldots, x_1)$$

$$p(x) = p(x_1) \cdot p(x_2 \mid x_1) \cdot p(x_3 \mid x_2, x_1) \cdots p(x_t \mid x_{t-1}, \ldots, x_1)$$

$$p(x) = p(x_t) \cdot p(x_{t-1} \mid x_t) \cdots p(x_1 \mid x_t, \ldots, x_2)$$

Language Models: Example

1. Convert raw text into a sequence of words (or "tokens")

Narwhals	are	way	cooler	than	parrots
x_1	x_2	x_3	x_4	x_5	x_6

2. Learn (or approximate) a joint probability distribution over sequences

$$p(x_1, x_2, x_3, x_4, x_5, x_6) =$$

Narwhals — $p(x_1)$

Narwhals are — $p(x_2|x_1)$

Narwhals are way — $p(x_3|x_2, x_1)$

Narwhals are way cooler — $p(x_4|x_3, x_2, x_1)$

Narwhals are way cooler than — $p(x_5|x_4, x_3, x_2, x_1)$

Narwhals are way cooler than parrots — $p(x_6|x_5, x_4, x_3, x_2, x_1)$

n-Gram Language Models

1. Convert raw text into a sequence of words (or "tokens")

$$\boldsymbol{x} = [x_1, x_2, \dots, x_t]$$

2. Idea: approximate the chain rule using *conditional independence* assumptions

bigram model ($n = 2$): $p(\boldsymbol{x}) \approx p(x_1) \prod_{i=2}^{t} p(x_i | x_{i-1})$

n-Gram Language Models

1. Convert raw text into a sequence of words (or "tokens")

$$\boldsymbol{x} = [x_1, x_2, \dots, x_t]$$

2. Idea: approximate the chain rule using *conditional independence* assumptions

trigram model: $p(\boldsymbol{x}) \approx p(x_1)p(x_2|x_1)\prod_{i=3}^{t} p(x_i|x_{i-1}, x_{i-2})$

Bigram Language Models: Example

1. Convert raw text into a sequence of words (or "tokens")

Narwhals	are	way	cooler	than	parrots
x_1	x_2	x_3	x_4	x_5	x_6

2. Idea: approximate the chain rule using *conditional independence* assumptions

$$p(x_1, x_2, x_3, x_4, x_5, x_6) \approx$$

Narwhals	$p(x_1)$
Narwhals are	$p(x_2 \| x_1)$
are way	$p(x_3 \| x_2)$
way cooler	$p(x_4 \| x_3)$
cooler than	$p(x_5 \| x_4)$
than parrots	$p(x_6 \| x_5)$

Trigram Language Models: Example

1. Convert raw text into a sequence of words (or "tokens")

Narwhals	are	way	cooler	than	parrots
x_1	x_2	x_3	x_4	x_5	x_6

2. Idea: approximate the chain rule using *conditional independence* assumptions

$$p(x_1, x_2, x_3, x_4, x_5, x_6) \approx$$

Narwhals						$p(x_1)$
Narwhals	are					$p(x_2 \vert x_1)$
Narwhals	are	way				$p(x_3 \vert x_2, x_1)$
	are	way	cooler			$p(x_4 \vert x_3, x_2)$
		way	cooler	than		$p(x_5 \vert x_4, x_3)$
			cooler	than	parrots	$p(x_6 \vert x_5, x_4)$

n-gram Language Models: Training

- What exactly are we learning when we train an n-gram language model?

n-gram Language Models: Training

- What exactly are we learning when we train an n-gram language model?
- A *lot* of probability distributions…

x_t	$p(x_t \mid \text{narwhals}, \text{are})$
aquatic	0.04
awesome	
⋮	
fish	

x_t	$p(x_t \mid \text{cooler}, \text{than})$
ice	0.01
cool	
⋮	
Henry	

x_t	$p(x_t \mid \text{axolotls}, \text{eat})$
shrimp	0.1
narwhals	0.00005
⋮	⋮
Henry	0.0001

⋱

n-gram Language Models: Training

- How do we train an n-gram language model?
- Using training data! Simply count frequency of next words

Narwhals are big aquatic mammals that…

Who knows what **narwhals are** hiding?

Watch out, the **narwhals are** coming!

These **narwhals are** friendly!

Narwhals are a surprisingly large p

The **narwhals are** a punk rock banc

Narwhals are big fans of machine

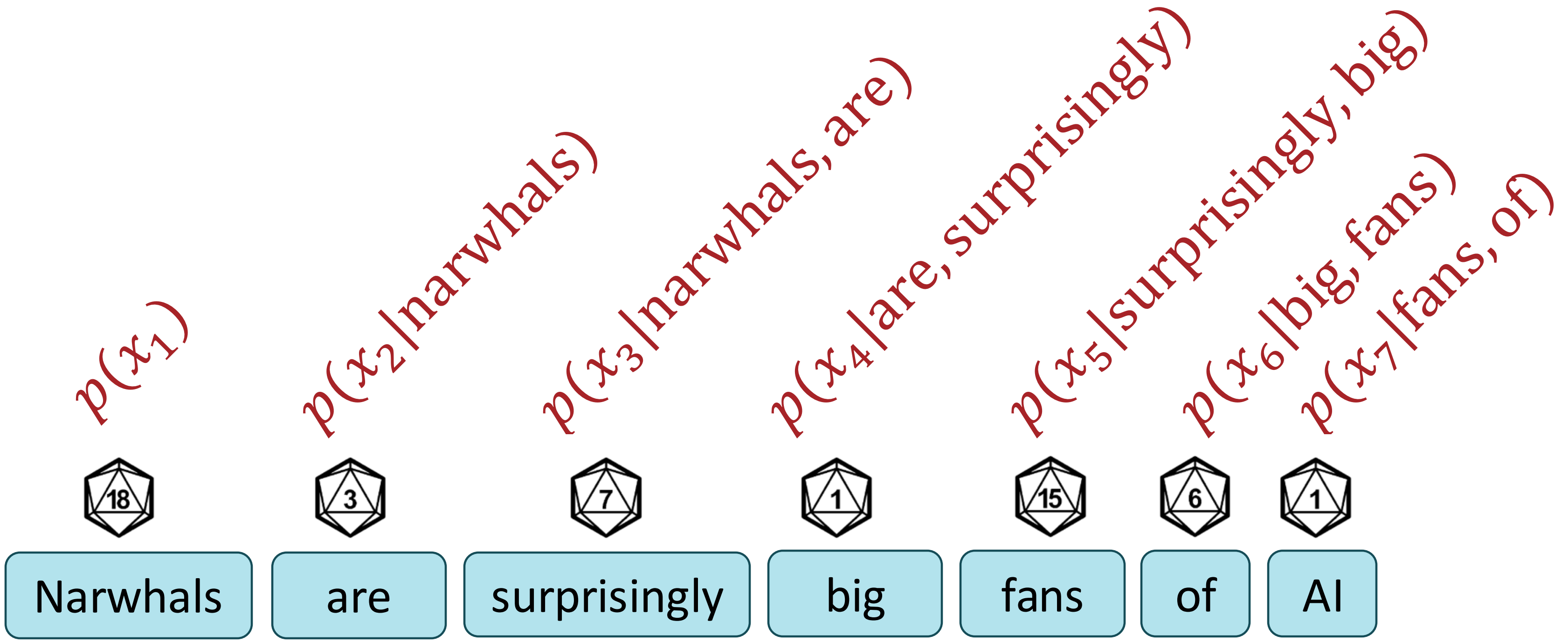
Narwhals are generated by AI.

x_t	$p(x_t \mid \text{narwhals, are})$
big	2/8
hiding	1/8
coming	1/8
friendly	1/8
a	2/8
generated	1/8

- Why are we counting?

n-gram Language Models: Generation

- How do we generate new sequences using an n-gram language model?
- Sample from the learned distributions and use the sampled words to determine what distribution to use next



n-gram Language Models: Generation?

- How do we generate new sequences using an n-gram language model?
- Sample from the learned distributions and use the sampled words to determine what distribution to use next

Training Data (Shakespeare)	5-Gram Model
I tell you, friends, most charitable care ave the patricians of you. For your wants, Your suffering in this dearth, you may as well Strike at the heaven with your staves as lift them Against the Roman state, whose course will on The way it takes, cracking ten thousand curbs Of more strong link asunder than can ever Appear in your impediment.	`Approacheth, denay. dungy Thither! Julius think: grant,--O Yead linens, sheep's Ancient, Agreed: Petrarch plaguy Resolved pear! observingly honourest adulteries wherever scabbard guess; affirmation--his monsieur; died. jealousy, chequins me.`

Language Models

1. Convert raw text into a sequence of words (or "tokens")

Narwhals	are	way	cooler	than	parrots
x_1	x_2	x_3	x_4	x_5	x_6

2. Idea: approximate the chain rule using a *parametric function* to compute *fixed-length* representations of prior words and condition on those to compute next words

Narwhals

Narwhals are

Narwhals are way

Narwhals are way cooler

Narwhals are way cooler than

Narwhals are way cooler than parrots

$$p(x_1, x_2, x_3, x_4, x_5, x_6) = p(x_1)$$

$$p(x_2 | f_\theta(x_1))$$

$$p(x_3 | f_\theta(x_2, x_1))$$

$$p(x_4 | f_\theta(x_3, x_2, x_1))$$

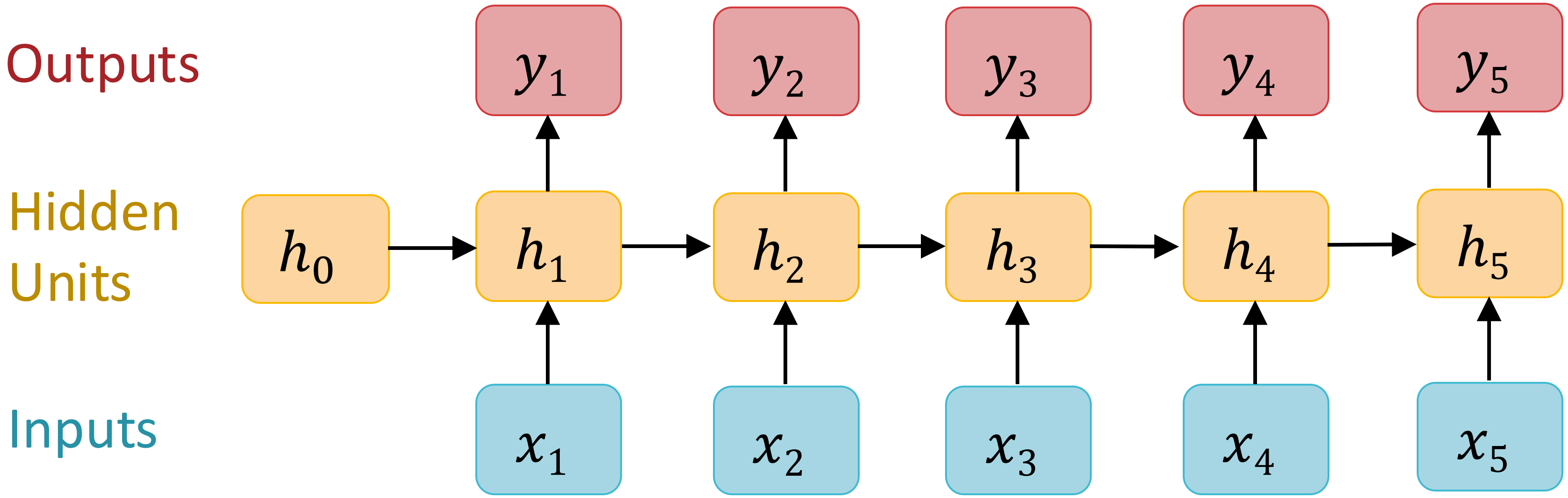
$$p(x_5 | f_\theta(x_4, x_3, x_2, x_1))$$

$$p(x_6 | f_\theta(x_5, x_4, x_3, x_2, x_1))$$

Recurrent Neural Networks

$$h_t = \phi(W_{xh}x_t + W_{hh}h_{t-1} + b_h)$$

$$y_t = \psi(W_{hy}h_t + b_y)$$

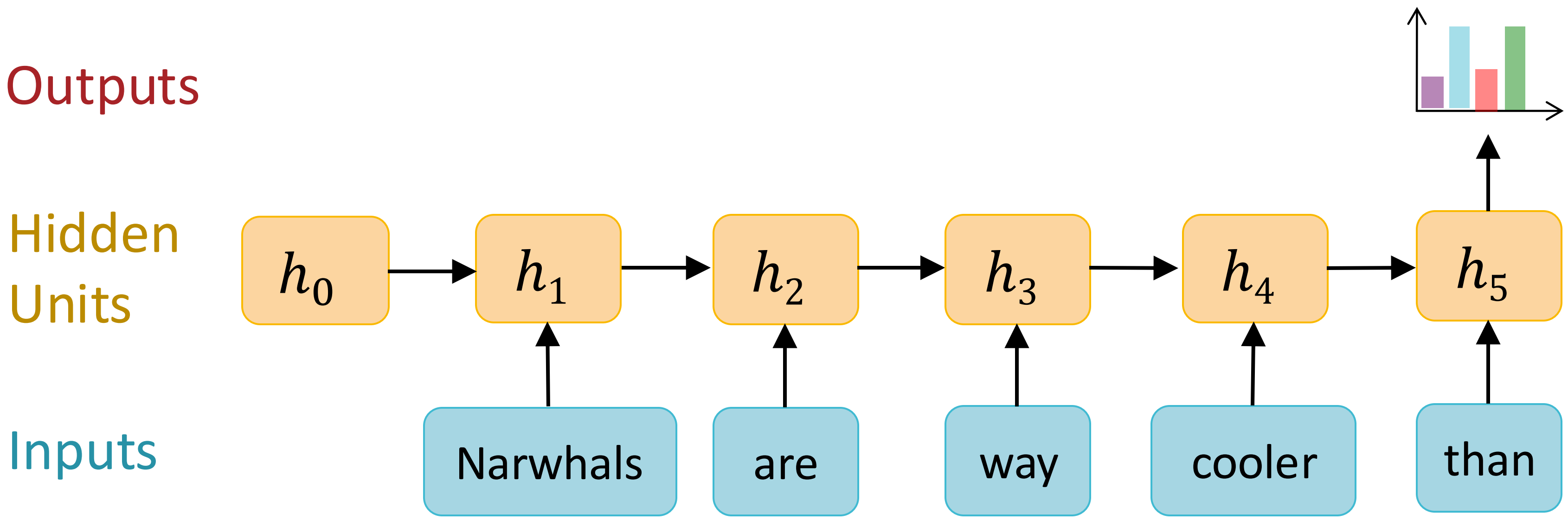


Recurrent Neural Network Language Models

- How can we get a distribution $p(x_t | h_{t-1} = f_\theta(x_1, \ldots, x_{t-1}))$ from some fixed-length vector h_{t-1}?

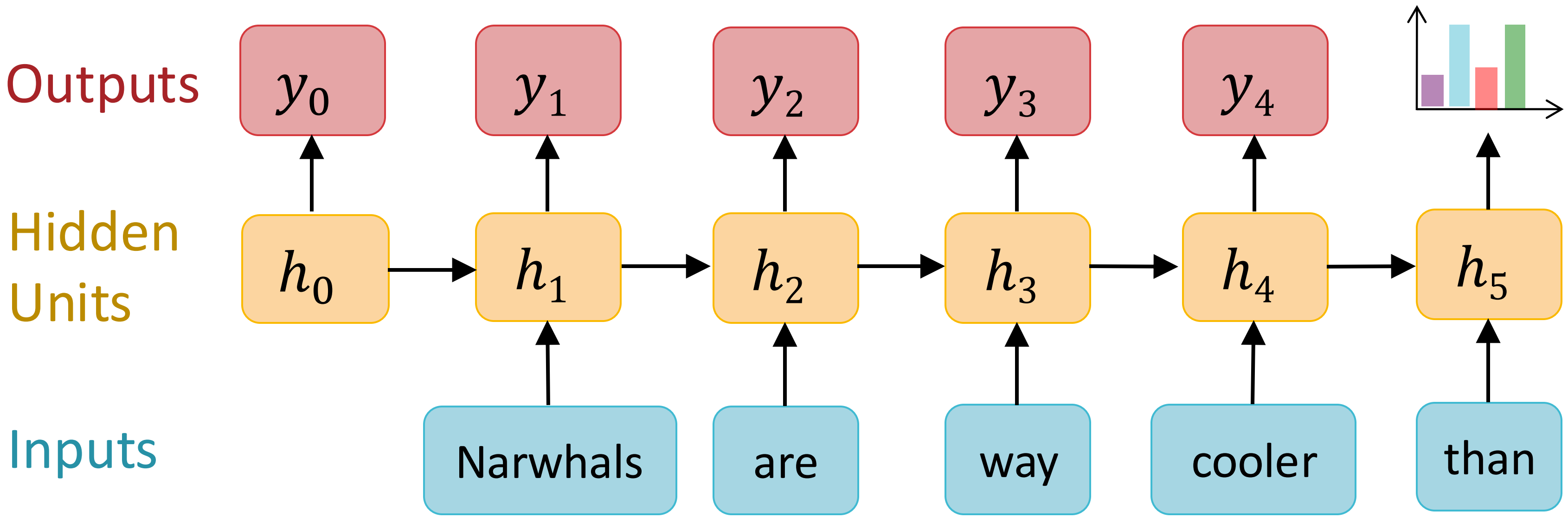
$y_t = \text{softmax}(\text{linear}(h_t))$

$p(x_6 | h_5 = f_\theta(x_1, \ldots, x_5))$

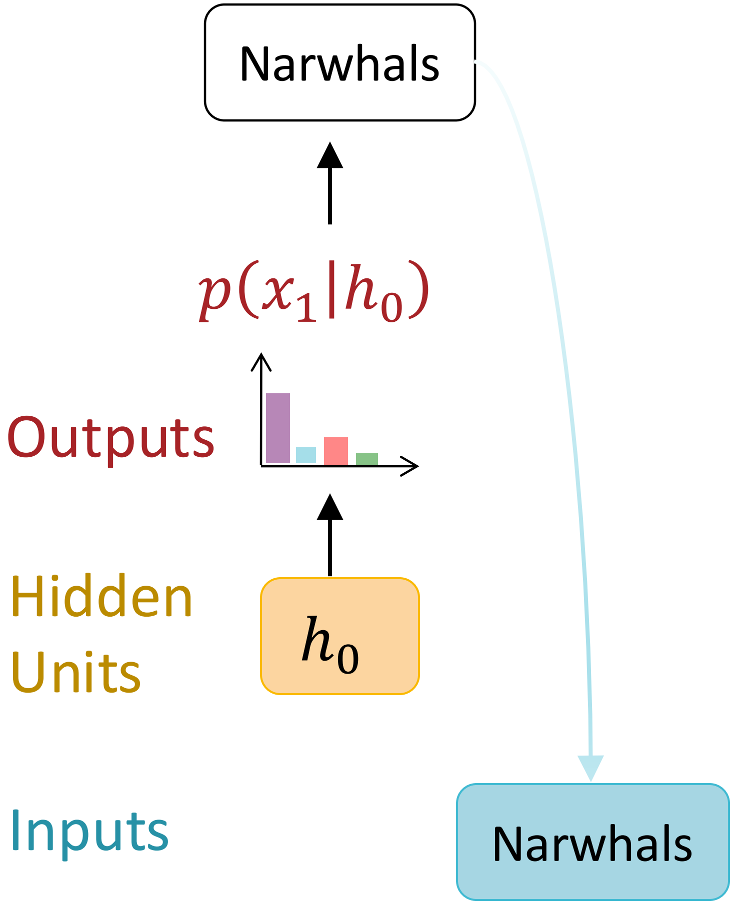


Recurrent Neural Network Language Models

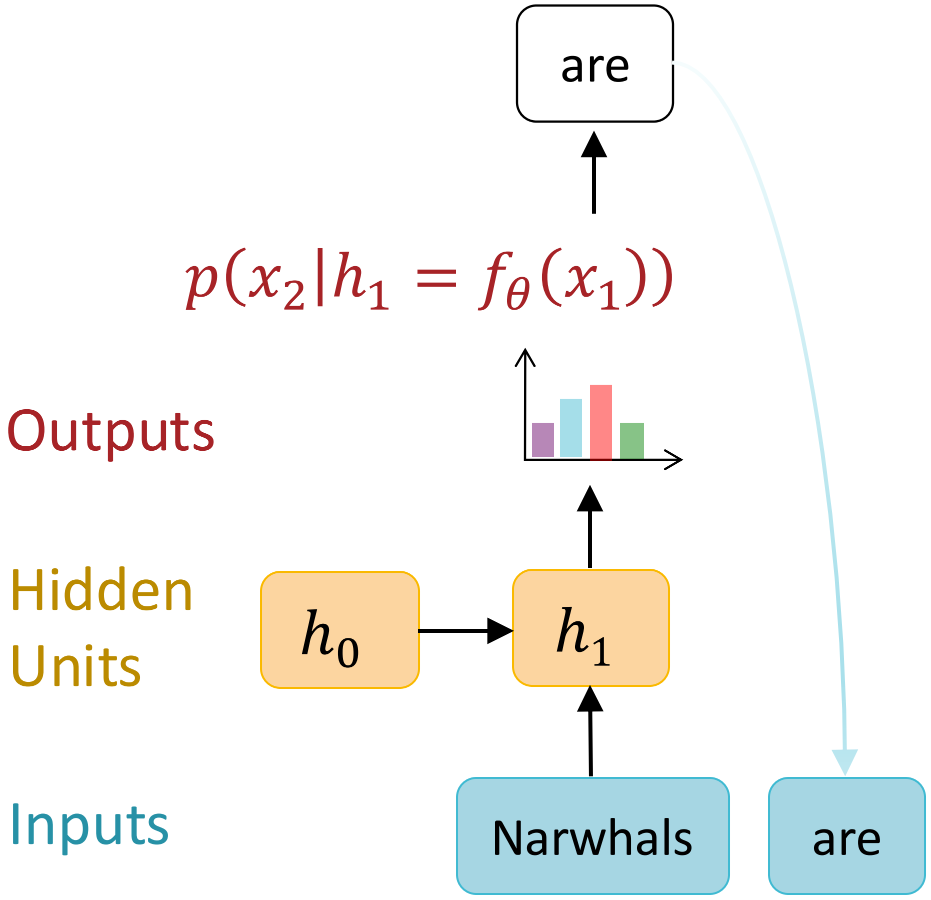
$$p(x_6 | h_5 = f_\theta(x_1, \dots, x_5))$$



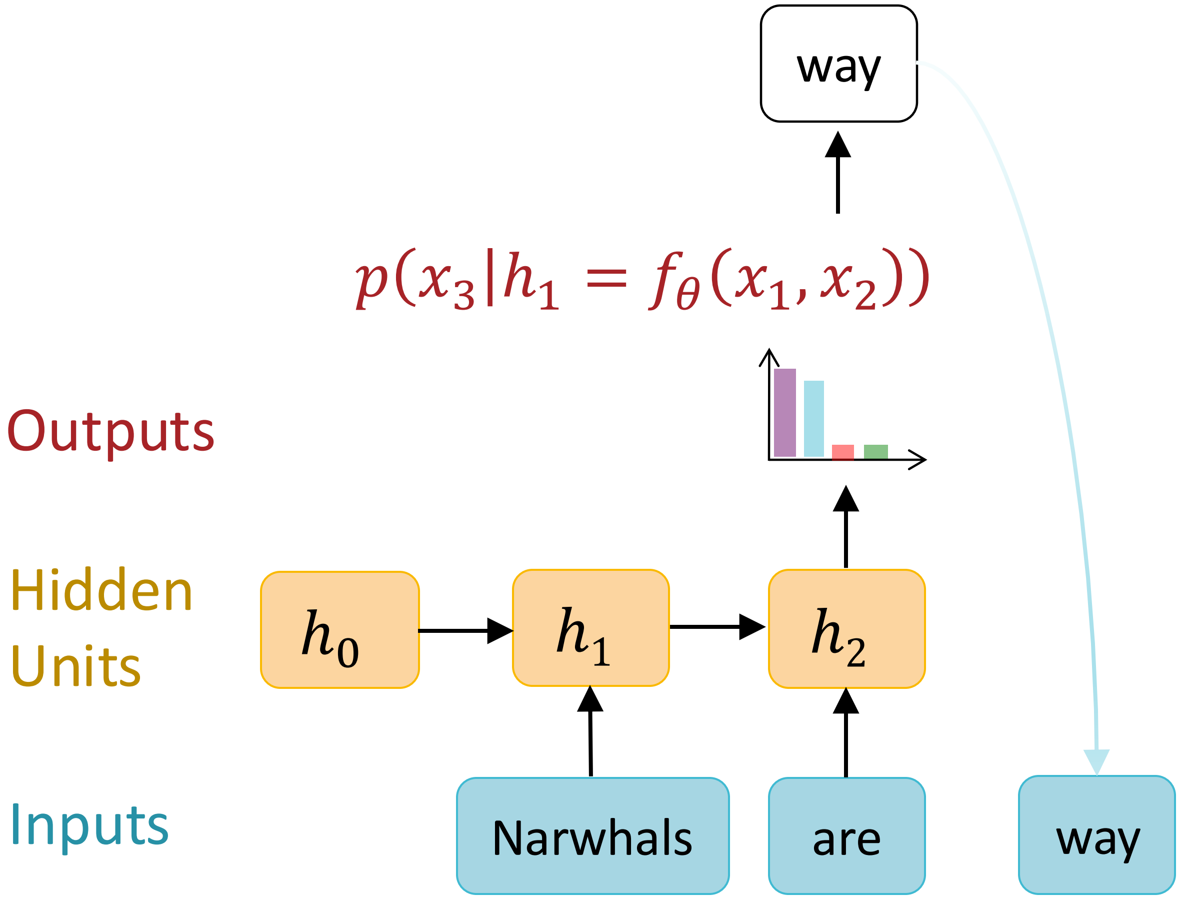
Recurrent Neural Network Language Models: Generation



Recurrent Neural Network Language Models: Generation

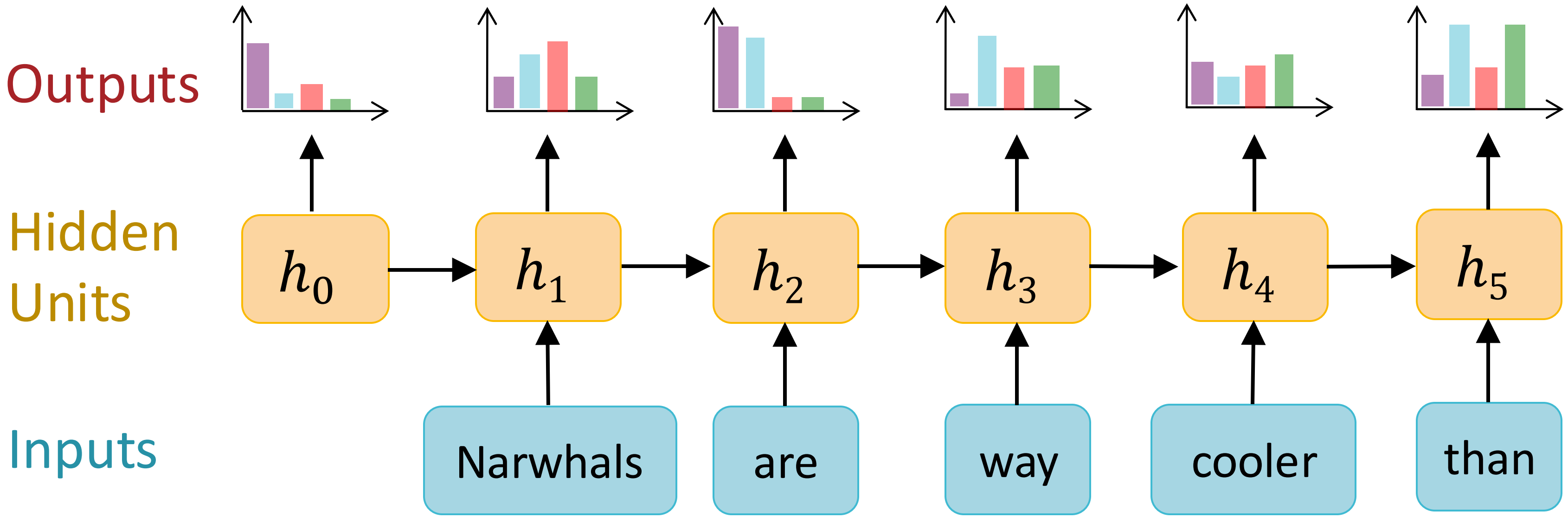


Recurrent Neural Network Language Models: Generation



Recurrent Neural Network Language Models: Generation

- How do we generate new sequences using an RNN language model?
- Exactly the same way we did for an n-gram language model, by sampling from some learned probability distributions over next words!



RNN Language Models: Generation

Shakespeare's As You Like It

VIOLA: Why, Salisbury must find his flesh and thought That which I am not aps, not a man and in fire, To show the reining of the raven and the wars To grace my hand reproach within, and not a fair are hand, That Caesar and my goodly father's world; When I was heaven of presence and our fleets, We spare with hours, but cut thy council I am great, Murdered and by thy master's ready there My power to give thee but so much as hell: Some service in the noble bondman here, Would show him to her wine.

KING LEAR: O, if you were a feeble sight, the courtesy of your law, Your sight and several breath, will wear the gods With his heads, and my hands are wonder'd at the deeds, So drop upon your lordship's head, and your opinion Shall be against your honour.

RNN-LM Sample

CHARLES: Marry, do I, sir; and I came to acquaint you with a matter. I am given, sir, secretly to understand that your younger brother Orlando hath a disposition to come in disguised against me to try a fall. To-morrow, sir, I wrestle for my credit; and he that escapes me without some broken limb shall acquit him well. Your brother is but young and tender; and, for your love, I would be loath to foil him, as I must, for my own honour, if he come in: therefore, out of my love to you, I came hither to acquaint you withal, that either you might stay him from his intendment or brook such disgrace well as he shall run into, in that it is a thing of his own search and altogether against my will.

TOUCHSTONE: For my part, I had rather bear with you than bear you; yet I should bear no cross if I did bear you, for I think you have no money in your purse.

Source: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

RNN-LM Sample?

VIOLA: Why, Salisbury must find his flesh and thought That which I am not aps, not a man and in fire, To show the reining of the raven and the wars To grace my hand reproach within, and not a fair are hand, That Caesar and my goodly father's world; When I was heaven of presence and our fleets, We spare with hours, but cut thy council I am great, Murdered and by thy master's ready there My power to give thee but so much as hell: Some service in the noble bondman here, Would show him to her wine.

KING LEAR: O, if you were a feeble sight, the courtesy of your law, Your sight and several breath, will wear the gods With his heads, and my hands are wonder'd at the deeds, So drop upon your lordship's head, and your opinion Shall be against your honour.

Shakespeare's As You Like It?

CHARLES: Marry, do I, sir; and I came to acquaint you with a matter. I am given, sir, secretly to understand that your younger brother Orlando hath a disposition to come in disguised against me to try a fall. To-morrow, sir, I wrestle for my credit; and he that escapes me without some broken limb shall acquit him well. Your brother is but young and tender; and, for your love, I would be loath to foil him, as I must, for my own honour, if he come in: therefore, out of my love to you, I came hither to acquaint you withal, that either you might stay him from his intendment or brook such disgrace well as he shall run into, in that it is a thing of his own search and altogether against my will.

TOUCHSTONE: For my part, I had rather bear with you than bear you; yet I should bear no cross if I did bear you, for I think you have no money in your purse.

RNN Language Models: Generation

Source: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

RNN-LM Sample?

VIOLA: Why, Salisbury must find his flesh and thought That which I am not aps, not a man and in fire, To show the reining of the raven and the wars To grace my hand reproach within, and not a fair are hand, That Caesar and my goodly father's world; When I was heaven of presence and our fleets, We spare with hours, but cut thy council I am great, Murdered and by thy master's ready there My power to give thee but so much as hell: Some service in the noble bondman here, Would show him to her wine.

KING LEAR: O, if you were a feeble sight, the courtesy of your law, Your sight and several breath, will wear the gods With his heads, and my hands are wonder'd at the deeds, So drop upon your lordship's head, and your opinion Shall be against your honour.

Shakespeare's As You Like It?

CHARLES: Marry, do I, sir; and I came to acquaint you with a matter. I am given, sir, secretly to understand that your younger brother Orlando hath a disposition to come in disguised against me to try a fall. To-morrow, sir, I wrestle for my credit; and he that escapes me without some broken limb shall acquit him well. Your brother is but young and tender; and, for your love, I would be loath to foil him, as I must, for my own honour, if he come in: therefore, out of my love to you, I came hither to acquaint you withal, that either you might stay him from his intendment or brook such disgrace well as he shall run into, in that it is a thing of his own search and altogether against my will.

TOUCHSTONE: For my part, I had rather bear with you than bear you; yet I should bear no cross if I did bear you, for I think you have no money in your purse.

RNN Language Models: Training?

Source: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Backpropagation

Forward Computation

1. Write an **algorithm** for evaluating the function $y = f(\boldsymbol{x})$.
   - The algorithm defines the **computation graph**, a **directed acyclic graph**, where each variable is a node
2. Visit each node in **topological order**:
   For variable u_i with inputs $v_1, \ldots, v_N$ compute
   $u_i = g_i(v_1, \ldots, v_N)$ and store the result at that node

Backward Computation (Version A)

1. **Initialize** $dy/dy = 1$.

2. Visit each node v_j in **reverse topological order**:
   - Let $u_1, \ldots, u_M$ denote all the nodes with v_j as an input
     Assume $y = h(\boldsymbol{u}) = h(u_1, \ldots, u_M)$ and
     $$u_i = g(\boldsymbol{v}) = g_i(v_1, \ldots, v_j, \ldots, v_N)\ \forall\, i$$
3. Compute $\dfrac{dy}{dv_j} = \displaystyle\sum_{i=1}^{M} \frac{dy}{du_i}\frac{du_i}{dv_j}$
   - We already know $dy/du_i\ \forall\, i$ and du_i/dv_j is (typically) easy to compute

4. **Return** partial derivatives dy/du_i for all variables

Backpropagation

Forward Computation

1. Write an **algorithm** for evaluating the function $y = f(\boldsymbol{x})$.
   - The algorithm defines the **computation graph**, a **directed acyclic graph**, where each variable is a node
2. Visit each node in **topological order**:
   For variable u_i with inputs $v_1, \ldots, v_N$ compute
   $u_i = g_i(v_1, \ldots, v_N)$ and store the result at that node

Backward Computation (Version B)

1. Initialize $dy/dy = 1$.
2. Visit each node v_j in **reverse topological order**
3. For each variable $u_i = g(\boldsymbol{v}) = g_i(v_1, \ldots, v_j, \ldots, v_N)$, increment dy/dv_j by $(dy/du_i)(du_i/dv_j)$
   a) We already know $dy/du_i\ \forall\ i$ and du_i/dv_j is (typically) easy to compute
4. Return partial derivatives dy/du_i for all variables

- **Key takeaway**: backpropagation makes efficient reuse of intermediate quantities when computing the necessary gradients to train a neural network

Backpropagation: Procedural Method

Algorithm 1 Forward Computation

1: **procedure** NNFORWARD(Training example $(\mathbf{x}, \mathbf{y})$, Params $\boldsymbol{\alpha}, \boldsymbol{\beta}$)
2: $\mathbf{a} = \boldsymbol{\alpha}\mathbf{x}$
3: $\mathbf{z} = \sigma(\mathbf{a})$
4: $\mathbf{b} = \boldsymbol{\beta}\mathbf{z}$
5: $\hat{\mathbf{y}} = \text{softmax}(\mathbf{b})$
6: $J = -\mathbf{y}^T \log \hat{\mathbf{y}}$
7: $\mathbf{o} = \texttt{object}(\mathbf{x}, \mathbf{a}, \mathbf{z}, \mathbf{b}, \hat{\mathbf{y}}, J)$
8: **return** intermediate quantities $\mathbf{o}$

Algorithm 2 Backpropagation

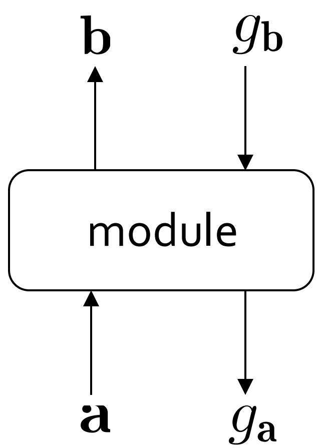
1: **procedure** NNBACKWARD(Training example $(\mathbf{x}, \mathbf{y})$, Params $\boldsymbol{\alpha}, \boldsymbol{\beta}$, Intermediates $\mathbf{o}$)
2: Place intermediate quantities $\mathbf{x}, \mathbf{a}, \mathbf{z}, \mathbf{b}, \hat{\mathbf{y}}, J$ in $\mathbf{o}$ in scope
3: $\mathbf{g}_{\hat{\mathbf{y}}} = -\mathbf{y} \div \hat{\mathbf{y}}$
4: $\mathbf{g}_{\mathbf{b}} = \mathbf{g}_{\hat{\mathbf{y}}}^T \left(\text{diag}(\hat{\mathbf{y}}) - \hat{\mathbf{y}}\hat{\mathbf{y}}^T\right)$
5: $\mathbf{g}_{\boldsymbol{\beta}} = \mathbf{g}_{\mathbf{b}}^T \mathbf{z}^T$
6: $\mathbf{g}_{\mathbf{z}} = \boldsymbol{\beta}^T \mathbf{g}_{\mathbf{b}}^T$
7: $\mathbf{g}_{\mathbf{a}} = \mathbf{g}_{\mathbf{z}} \odot \mathbf{z} \odot (1 - \mathbf{z})$
8: $\mathbf{g}_{\boldsymbol{\alpha}} = \mathbf{g}_{\mathbf{a}} \mathbf{x}^T$
9: **return** parameter gradients $\mathbf{g}_{\boldsymbol{\alpha}}, \mathbf{g}_{\boldsymbol{\beta}}$

Issues:

1. Hard to reuse / adapt for other models
2. Hard to optimize individual steps
3. Hard to debug using the finite-difference check

Module-based AutoDiff

- Key Idea:
  - componentize the computation of the neural-network into layers
  - each layer consolidates multiple **real-valued nodes** in the computation graph (a subset of them) into one **vector-valued node** (aka. a **module**)
- Each **module** is capable of two actions:
  - **Forward computation of the output given some input**
  - **Backward computation of the gradient with respect to the input given the gradient with respect to the output**



Module-based AutoDiff

Linear Module The linear layer has two inputs: a vector $\mathbf{a}$ and parameters $\omega \in \mathbb{R}^{B \times A}$. The output $\mathbf{b}$ is not used by LINEARBACKWARD, but we pass it in for consistency of form.

```
procedure LINEARFORWARD(a, ω)
    b = ωa
    return b
procedure LINEARBACKWARD(a, ω, b, g_b)
    g_ω = g_b a^T
    g_a = ω^T g_b
    return g_ω, g_a
```

Softmax Module The softmax layer has only one input vector $\mathbf{a}$. For any vector $\mathbf{v} \in \mathbb{R}^D$, we have that $\text{diag}(\mathbf{v})$ returns a $D \times D$ diagonal matrix whose diagonal entries are $v_1, v_2, \ldots, v_D$ and whose non-diagonal entries are zero.

```
procedure SOFTMAXFORWARD(a)
    b = softmax(a)
    return b
procedure SOFTMAXBACKWARD(a, b, g_b)
    g_a = g_b^T (diag(b) − b b^T)
    return g_a
```

Sigmoid Module The sigmoid layer has only one input vector $\mathbf{a}$. Below σ is the sigmoid applied element-wise, and $\odot$ is element-wise multiplication s.t. $\mathbf{u} \odot \mathbf{v} = [u_1 v_1, \ldots, u_M v_M]$.

```
procedure SIGMOIDFORWARD(a)
    b = σ(a)
    return b
procedure SIGMOIDBACKWARD(a, b, g_b)
    g_a = g_b ⊙ b ⊙ (1 − b)
    return g_a
```

Cross-Entropy Module The cross-entropy layer has two inputs: a gold one-hot vector $\mathbf{a}$ and a predicted probability distribution $\hat{\mathbf{a}}$. It's output $b \in \mathbb{R}$ is a scalar. Below $\div$ is element-wise division. The output b is not used by CROSSENTROPYBACKWARD, but we pass it in for consistency of form.

```
procedure CROSSENTROPYFORWARD(a, â)
    b = −a^T log â
    return b
procedure CROSSENTROPYBACKWARD(a, â, b, g_b)
    g_â = −g_b (a ÷ â)
    return g_a
```

Module-based AutoDiff

Algorithm 1 Forward Computation

procedure NNFORWARD(Training example $(\mathbf{x}, \mathbf{y})$, Parameters $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$)
  $\mathbf{a} = \text{LINEARFORWARD}(\mathbf{x}, \boldsymbol{\alpha})$
  $\mathbf{z} = \text{SIGMOIDFORWARD}(\mathbf{a})$
  $\mathbf{b} = \text{LINEARFORWARD}(\mathbf{z}, \boldsymbol{\beta})$
  $\hat{\mathbf{y}} = \text{SOFTMAXFORWARD}(\mathbf{b})$
  $J = \text{CROSSENTROPYFORWARD}(\mathbf{y}, \hat{\mathbf{y}})$
  $\mathbf{o} = \texttt{object}(\mathbf{x}, \mathbf{a}, \mathbf{z}, \mathbf{b}, \hat{\mathbf{y}}, J)$
  return intermediate quantities $\mathbf{o}$

Algorithm 2 Backpropagation

procedure NNBACKWARD(Training example $(\mathbf{x}, \mathbf{y})$, Parameters $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, Intermediates $\mathbf{o}$)
  Place intermediate quantities $\mathbf{x}, \mathbf{a}, \mathbf{z}, \mathbf{b}, \hat{\mathbf{y}}, J$ in $\mathbf{o}$ in scope
  $g_J = \frac{dJ}{dJ} = 1$ ▷ Base case
  $\mathbf{g}_{\hat{\mathbf{y}}} = \text{CROSSENTROPYBACKWARD}(\mathbf{y}, \hat{\mathbf{y}}, J, g_J)$
  $\mathbf{g}_{\mathbf{b}} = \text{SOFTMAXBACKWARD}(\mathbf{b}, \hat{\mathbf{y}}, \mathbf{g}_{\hat{\mathbf{y}}})$
  $\mathbf{g}_{\boldsymbol{\beta}}, \mathbf{g}_{\mathbf{z}} = \text{LINEARBACKWARD}(\mathbf{z}, \mathbf{b}, \mathbf{g}_{\mathbf{b}})$
  $\mathbf{g}_{\mathbf{a}} = \text{SIGMOIDBACKWARD}(\mathbf{a}, \mathbf{z}, \mathbf{g}_{\mathbf{z}})$
  $\mathbf{g}_{\boldsymbol{\alpha}}, \mathbf{g}_{\mathbf{x}} = \text{LINEARBACKWARD}(\mathbf{x}, \mathbf{a}, \mathbf{g}_{\mathbf{a}})$ ▷ We discard $\mathbf{g}_{\mathbf{x}}$
  return parameter gradients $\mathbf{g}_{\boldsymbol{\alpha}}, \mathbf{g}_{\boldsymbol{\beta}}$

1. Easy to reuse / adapt for other models
2. Individual layers are easier to optimize
3. Simple to debug: just run a finite-difference check on each layer separately

Module-based AutoDiff (OOP Version)

Object-Oriented Implementation:

- Let each module be an object and allow the **control flow** of the program to define the computation graph
- **No longer need to implement NNBackward(·)**, just follow the computation graph in reverse topological order

```
class Sigmoid(Module)
    method forward(a)
        b = σ(a)
        return b
    method backward(a, b, g_b)
        g_a = g_b ⊙ b ⊙ (1 − b)
        return g_a
```

```
class Softmax(Module)
    method forward(a)
        b = softmax(a)
        return b
    method backward(a, b, g_b)
        g_a = g_b^T (diag(b) − b b^T)
        return g_a
```

```
class Linear(Module)
    method forward(a, ω)
        b = ω a
        return b
    method backward(a, ω, b, g_b)
        g_ω = g_b a^T
        g_a = ω^T g_b
        return g_ω, g_a
```

```
class CrossEntropy(Module)
    method forward(a, â)
        b = −a^T log â
        return b
    method backward(a, â, b, g_b)
        g_â = −g_b (a ÷ â)
        return g_a
```

Module-based AutoDiff (OOP Version)

```
class NeuralNetwork(Module):

    method init()
        lin1_layer = Linear()
        sig_layer = Sigmoid()
        lin2_layer = Linear()
        soft_layer = Softmax()
        ce_layer = CrossEntropy()

    method forward(Tensor x, Tensor y, Tensor α, Tensor β)
        a = lin1_layer.apply_fwd(x, α)
        z = sig_layer.apply_fwd(a)
        b = lin2_layer.apply_fwd(z, β)
        ŷ = soft_layer.apply_fwd(b)
        J = ce_layer.apply_fwd(y, ŷ)
        return J.out_tensor

    method backward(Tensor x, Tensor y, Tensor α, Tensor β)
        tape_bwd()
        return lin1_layer.in_gradients[1], lin2_layer.in_gradients[1]
```

Module-based AutoDiff (OOP Version)

```
global tape = stack()

class Module:

    method init()
        out_tensor = null
        out_gradient = 1

    method apply_fwd(List in_modules)
        in_tensors = [x.out_tensor for x in in_modules]
        out_tensor = forward(in_tensors)
        tape.push(self)
        return self

    method apply_bwd():
        in_gradients = backward(in_tensors, out_tensor, out_gradient)
        for i in 1,...,len(in_modules):
            in_modules[i].out_gradient += in_gradients[i]
        return self

function tape_bwd():
    while len(tape) > 0
        m = tape.pop()
        m.apply_bwd()
```

PyTorch

```
class NeuralNetwork(nn.Module):
    def __init__(self):
        super().__init__()
        self.flatten = nn.Flatten()
        self.linear1 = nn.Linear(28*28, 512)
        self.sigmoid = nn.Sigmoid()
        self.linear2 = nn.Linear(512, 512)

    def forward(self, x):
        x = self.flatten(x)
        a = self.linear1(x)
        z = self.sigmoid(a)
        b = self.linear2(z)
        return b

def one_step_of_sgd(X, y):
    model = NeuralNetwork()
    loss_fn = nn.CrossEntropyLoss()
    optimizer = torch.optim.SGD(model.parameters(), lr=1e-2)

    pred = model(X)
    loss = loss_fn(pred, y)

    optimizer.zero_grad()
    loss.backward()
    optimizer.step()
```

PyTorch: FAQ

- Q: Why don't we call `linear.forward()` in PyTorch?
- A: There's a special method in Python `__call__` that allows you to define what happens when you treat an object as if it were a function so `linear(x)` is equivalent `linear.__call__(x)` and PyTorch defines every Module's `__call__` method to be something like

```
def __call__(self):
    self.forward()
```

PyTorch: FAQ

- Q: Why don't we pass in the parameters to a Module?
- A: This is just a design choice; in PyTorch, you store the parameters inside the Module and "mark" them as parameters that should contribute to the eventual gradient used by an optimizer