Language model pretraining

CS 5624: Natural Language Processing

Fall 2026

https://tuvllms.github.io/nlp-fall-2026/

Tu Vu

VIRGINIA TECH

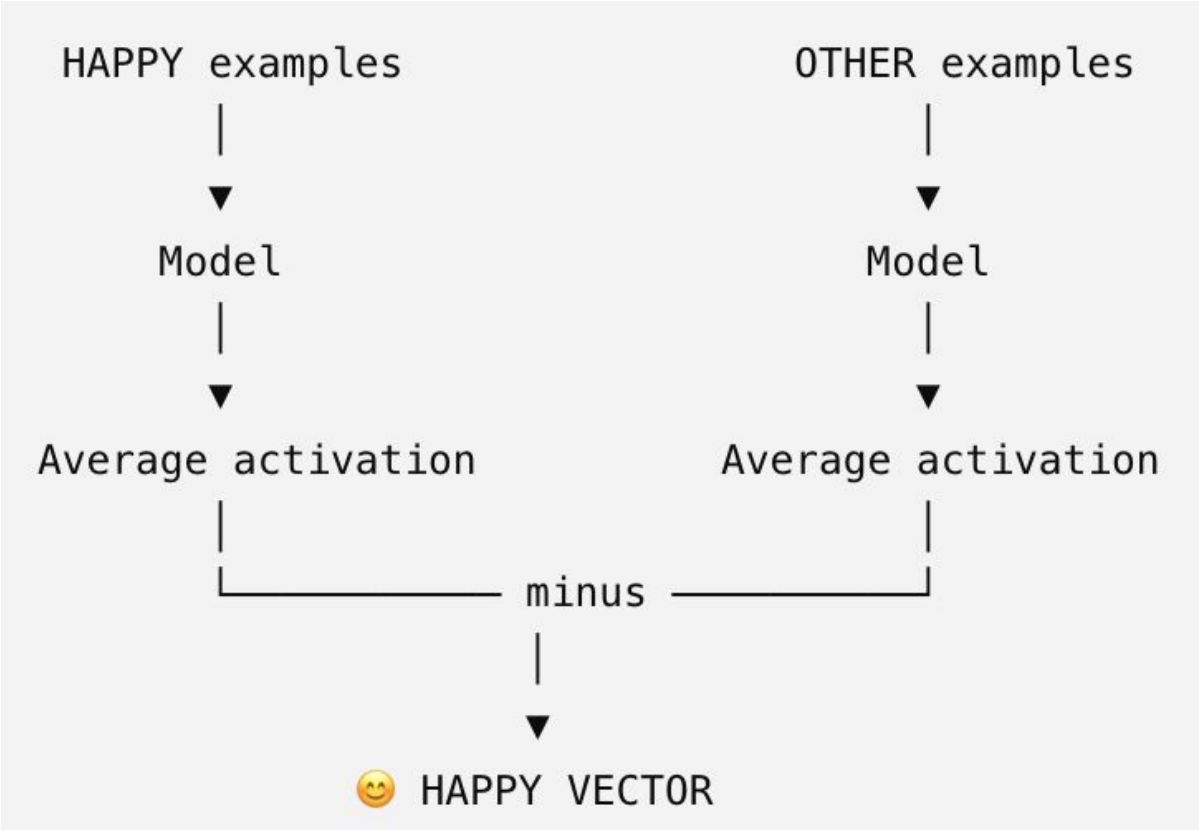
Logistics

- Final projects
  - Final group confirmed next week
- Quiz0 due tomorrow
- HW0 due next Friday

Studying internal representations

- Latent Planning Emerges with Scale // Transformers plan ahead
- Anthropic's Emergent Introspective Awareness in Large Language Models // the model settles on its rhyme before writing toward a full line of verse
- Anthropic's Verbalizable Representations Form a Global Workspace in Language Models // Models form a global workspace
- Anthropic's Emotion Concepts and their Function in a Large Language Model // Models form emotional concepts before responding, which shape what they say

Steering vectors



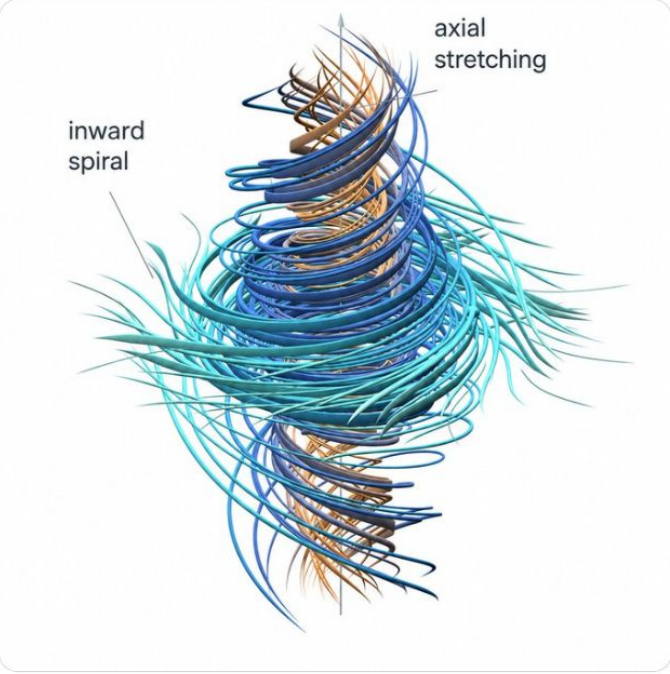
AI news

OpenAI ✓
@OpenAI

We're sharing a solution to the Navier-Stokes Millennium Prize Problem, one of the deepest problems at the frontier of mathematics.

The proof was produced by a group of agents, using an OpenAI next-generation model significantly more capable than GPT-6 Astra.

The problem concerns whether the description of smooth three-dimensional fluid motion modeled by the Navier-Stokes equations can break down. It has remained unresolved for roughly 90 years.



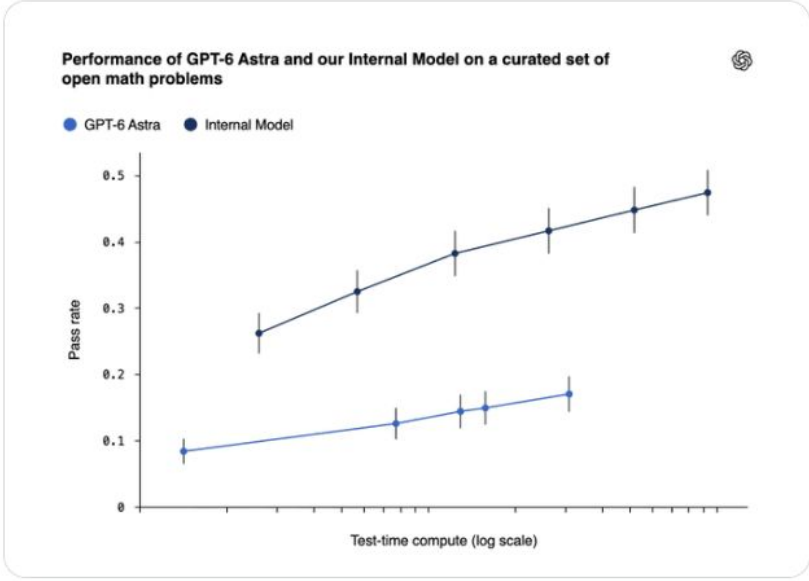
1:20 PM · Sep 8, 2026 · **71.9M** Views

OpenAI ✓
@OpenAI

This model represents a step-function improvement on many benchmarks, and its training is ongoing.

Our internal model group arrived at the Navier–Stokes solution in 88 hours, using around 10,000 coordinating AI agents.

Throughout the effort, we maintained the strict safeguards—including monitoring and isolation—that we apply to all our frontier evaluations.



1:20 PM · Sep 8, 2026 · **4.1M** Views

OpenAI
@OpenAI

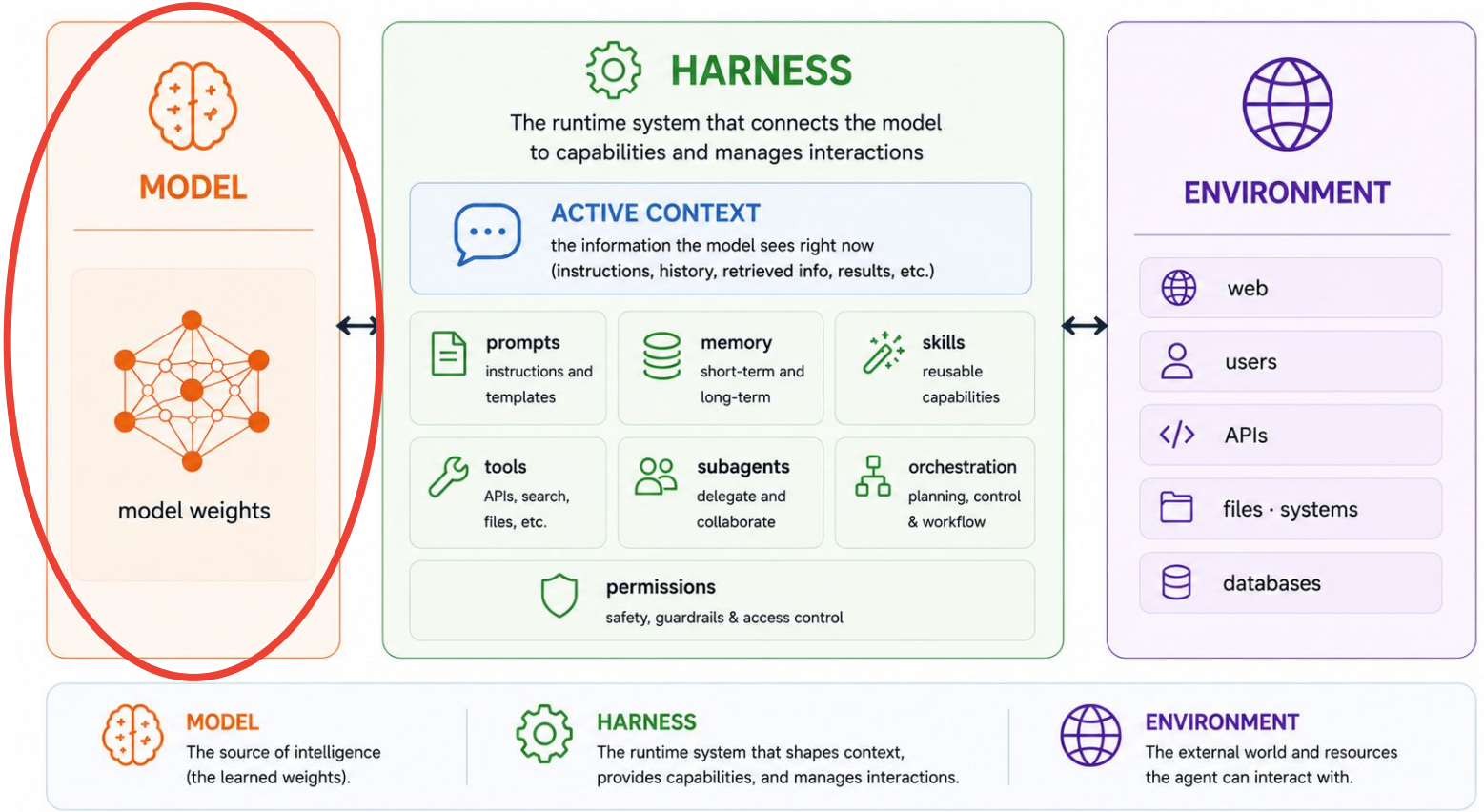
We congratulate Levent Alpöge and Tristan Buckmaster on their remarkable mathematical work.

We (the researchers and the agents) did not see any of their work through any means until they released it publicly — in particular, no specific user data was accessed in order to solve this problem.

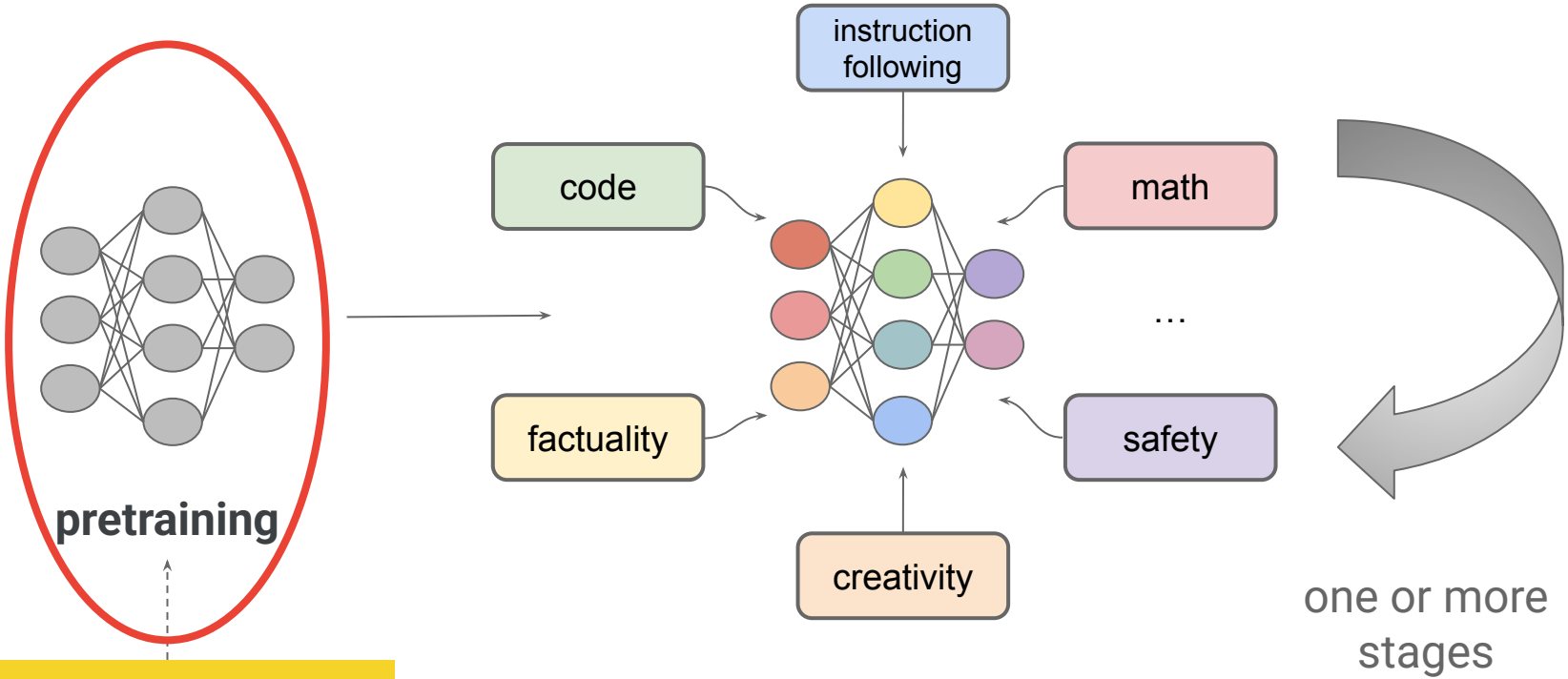
While unlikely, we cannot rule out that de-identified data derived from their usage of our products helped improve our models.

However, our proofs differ significantly and even the precise results proved are different in the Euler case (forced vs. unforced).

Where are we?

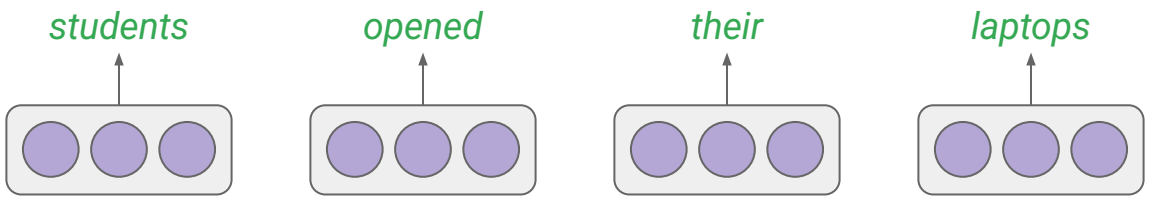

MODEL
model weights
HARNESS
The runtime system that connects the model to capabilities and manages interactions
ACTIVE CONTEXT
the information the model sees right now (instructions, history, retrieved info, results, etc.)
prompts
instructions and templates
memory
short-term and long-term
skills
reusable capabilities
tools
APIs, search, files, etc.
subagents
delegate and collaborate
orchestration
planning, control & workflow
permissions
safety, guardrails & access control
ENVIRONMENT
web
users
APIs
files · systems
databases
MODEL
The source of intelligence (the learned weights).
HARNESS
The runtime system that shapes context, provides capabilities, and manages interactions.
ENVIRONMENT
The external world and resources the agent can interact with.

The development of modern LLMs



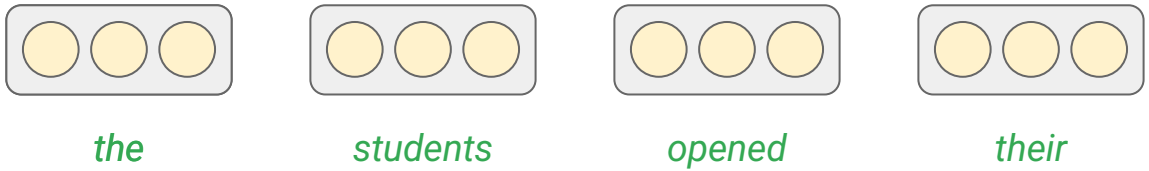
How to evaluate language models during pretraining?

Decoder-only Transformer

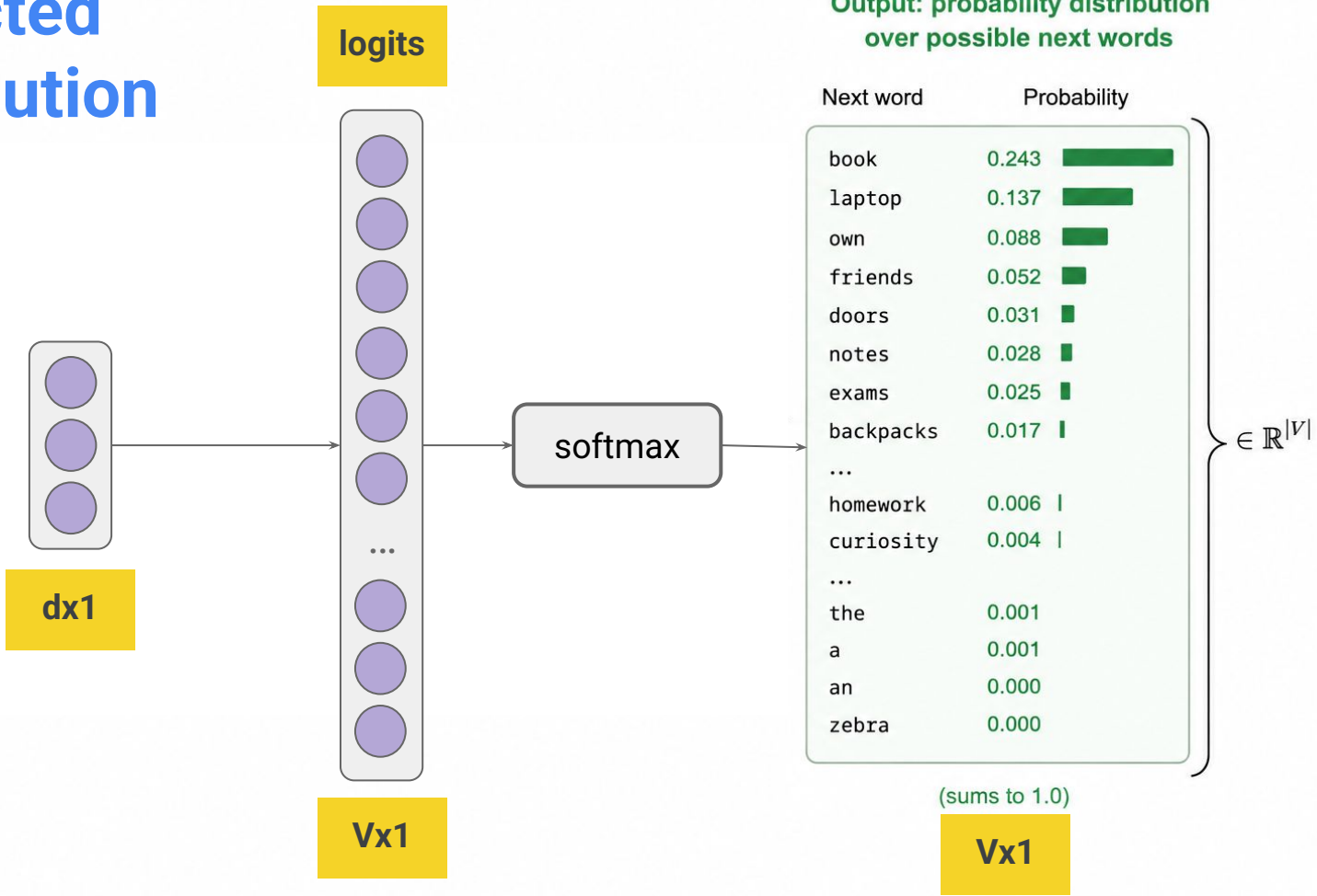


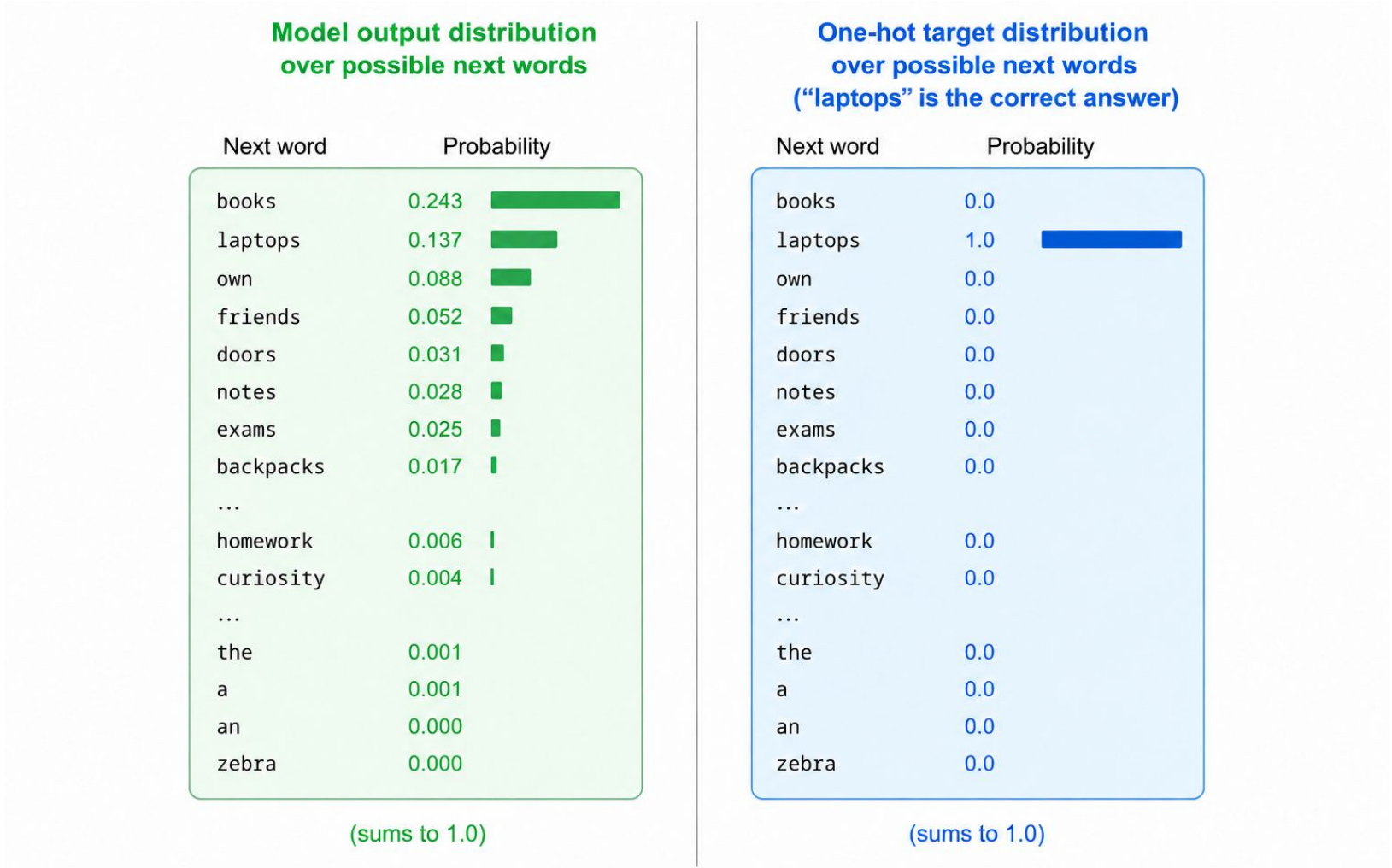
the architecture used in frontier LLMs

Transformer decoder
(causal mask)



Predicted distribution




Model output distribution
over possible next words
Next word
Probability
books 0.243
laptops 0.137
own 0.088
friends 0.052
doors 0.031
notes 0.028
exams 0.025
backpacks 0.017
...
homework 0.006
curiosity 0.004
...
the 0.001
a 0.001
an 0.000
zebra 0.000
(sums to 1.0)
One-hot target distribution
over possible next words
("laptops" is the correct answer)
Next word
Probability
books 0.0
laptops 1.0
own 0.0
friends 0.0
doors 0.0
notes 0.0
exams 0.0
backpacks 0.0
...
homework 0.0
curiosity 0.0
...
the 0.0
a 0.0
an 0.0
zebra 0.0
(sums to 1.0)

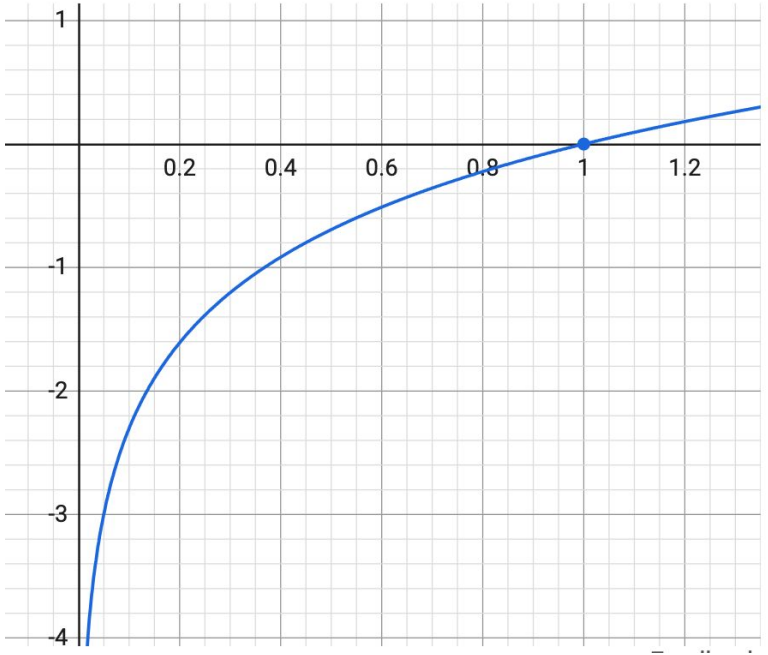
Negative log likelihood (NLL) loss

$$\mathcal{L}_{\text{NLL}} = -\log p(\text{laptops})$$

logs to avoid numerical underflow

Negative log likelihood (NLL) loss (cont'd)

$$\mathcal{L}_{\text{NLL}}(\hat{y}, y) = -\log \hat{y}_c$$



$$\hat{y}_c \to 0, \log \hat{y}_c \to -\infty$$

- If $\hat{y}_c = 0.9$, then $\log(0.9) \approx -0.105$, and the loss will be small.
- If $\hat{y}_c = 0.1$, then $\log(0.1) \approx -2.302$, and the loss will be much larger.

Cross-entropy

For probability distributions p, q:

$$H(p, q) = -\sum_i p_i \log q_i.$$

Gibbs' inequality states:

$$H(p, q) \geq H(p) = -\sum_i p_i \log p_i,$$

with equality if and only if $p = q$.

Cross-entropy is **defined** as

$$H(q, p) = -\sum_i q_i \log p_i,$$

where q is the true distribution and p is the predicted distribution.

The reason for the $\log p_i$ comes from **information theory**. The amount of information, or "surprise," associated with observing an event that has probability p is defined as

$$I(p) = -\log p.$$

This definition has an important property. If two independent events occur with probabilities p_1 and p_2, their joint probability is

$$p_1 p_2.$$

We want their information to **add**:

$$I(p_1 p_2) = I(p_1) + I(p_2).$$

The logarithm gives exactly this property:

$$-\log(p_1 p_2) = -\log p_1 - \log p_2.$$

Cross-entropy then asks:

> **On average, how surprised would I be by data generated from q, if I use p to assign probabilities to it?**

Since the surprise for outcome i under the prediction p is

$$-\log p_i,$$

and outcome i actually occurs with probability q_i, its expected surprise is

$$\boxed{H(q, p) = \sum_i q_i(-\log p_i) = -\sum_i q_i \log p_i}$$

So the **log is there because cross-entropy measures expected information/surprise**, and information is logarithmic in probability.

Cross-entropy loss

$$L_{CE}(\hat{y}, y) = -\sum_{i=1}^{V} y_i \log \hat{y}_i$$

$$L_{CE}(\hat{y}, y) = -\left(y_1 \log \hat{y}_1 + y_2 \log \hat{y}_2 + \cdots + y_V \log \hat{y}_V\right)$$

Since the true label y is one-hot encoded, only one term in the sum is nonzero, corresponding to the correct class c, where $y_c = 1$ and $y_i = 0$ for all $i \neq c$. This simplifies the sum to:

$$L_{CE}(\hat{y}, y) = -y_c \log \hat{y}_c$$

Since $y_c = 1$, this further reduces to:

$$L_{CE}(\hat{y}, y) = -\log \hat{y}_c$$

The loss models the distance between the system output and the gold output —lower is better

Perplexity

For a language model, **perplexity is just the exponential of the average negative log-likelihood (NLL)**, assuming the NLL uses natural logarithms.

Suppose the model assigns probabilities to a sequence of T tokens:

$$p(x_1, \ldots, x_T) = \prod_{t=1}^{T} p(x_t \mid x_{<t}).$$

The total negative log-likelihood is

$$\mathrm{NLL} = -\sum_{t=1}^{T} \log p(x_t \mid x_{<t}).$$

First, average the NLL over the tokens:

$$\overline{\mathrm{NLL}} = -\frac{1}{T} \sum_{t=1}^{T} \log p(x_t \mid x_{<t}).$$

Then exponentiate:

$$\boxed{\mathrm{PPL} = \exp(\overline{\mathrm{NLL}})}$$

So, if your reported language-model loss is already **mean cross-entropy/NLL per token**, as it usually is during training, then simply

$$\boxed{\mathrm{PPL} = e^{\mathrm{loss}}}.$$

Perplexity (cont'd)

If perplexity is

$$\mathrm{PPL} = 10,$$

then average cross-entropy is

$$H = \log 10.$$

Since cross-entropy for the correct token is

$$H = -\log p,$$

we get

$$-\log p = \log 10 \quad \Rightarrow \quad p = \frac{1}{10}.$$

So **PPL = 10** means the model has the same average uncertainty as choosing uniformly among **10 options**.

Backpropagation

The partial derivative of the loss function

The partial derivative of the loss function L with respect to the parameter w represents how much the loss changes as w changes.

$$\frac{dL}{dw}$$

The gradient

A **derivative** applies to a function with a single variable. It measures the rate of change of the function with respect to that variable. For example, if $f(x) = x^2$, then the derivative $f'(x) = 2x$ shows how fast $f(x)$ changes as x changes.

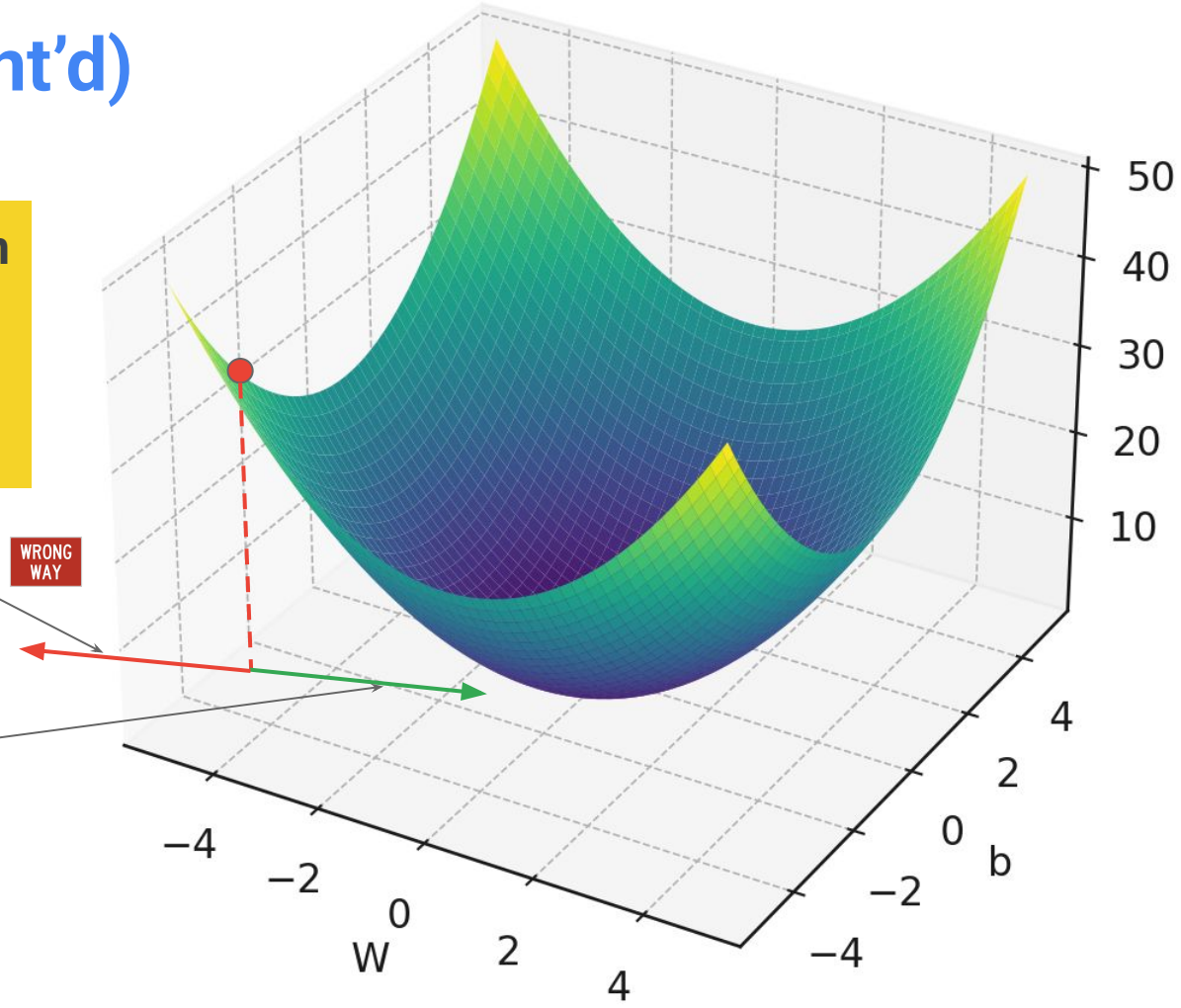
A **gradient** applies to a function with multiple variables. It is a vector that contains all of the partial derivatives of the function with respect to each variable. For example, if $f(x, y) = x^2 + y^2$, then the gradient is

$$\nabla f(x, y) = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right) = (2x, 2y).$$

The gradient (cont'd)

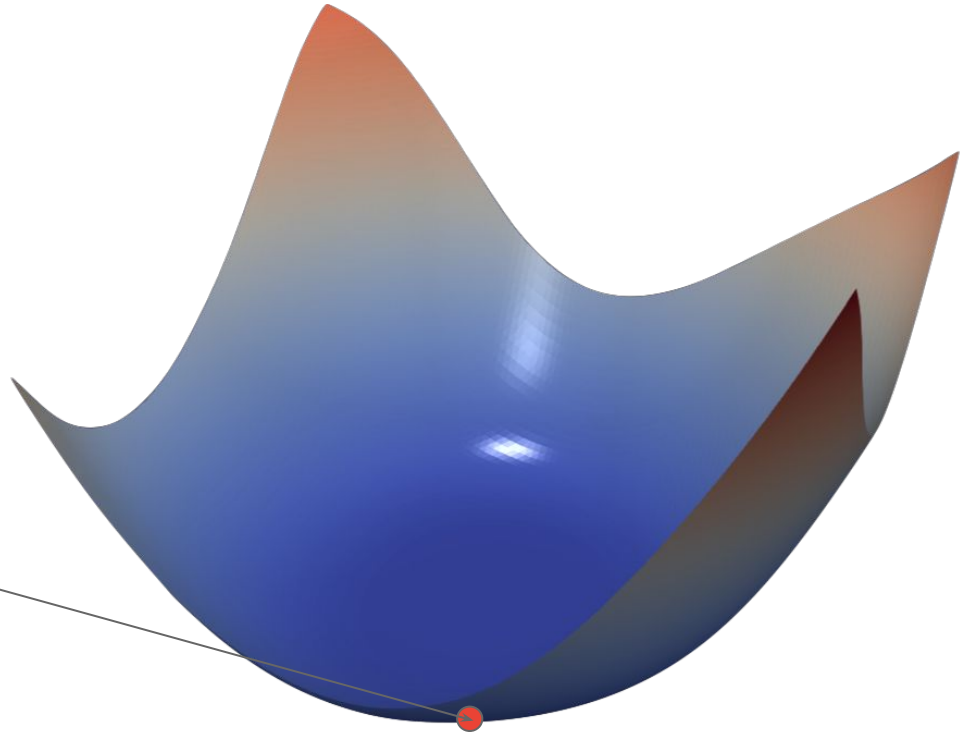
the gradient points in the direction of the steepest increase in the loss

negative gradient



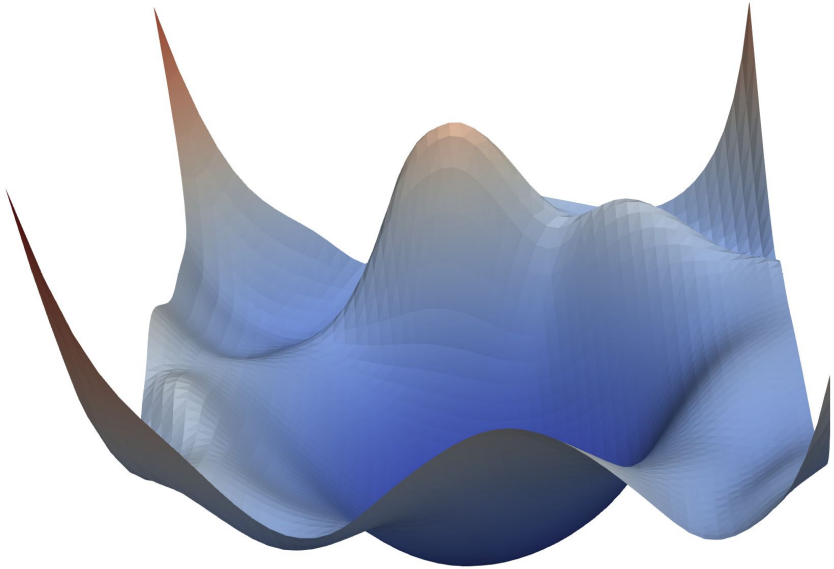
The loss landscape of neural nets

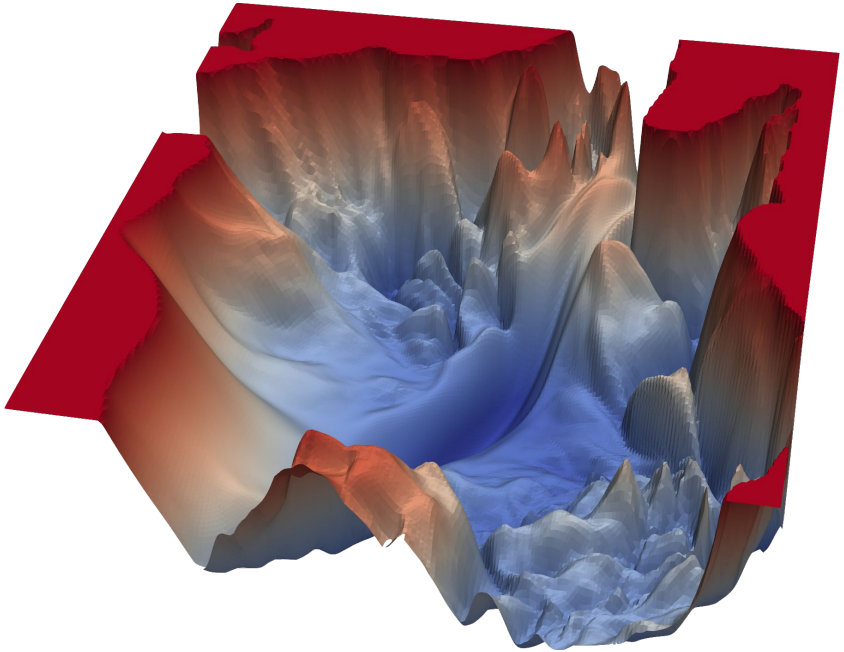
A convex function has at most one minimum; there are no local minima to get stuck in.



https://www.cs.umd.edu/~tomg/project/landscapes/

The loss landscape of neural nets (cont'd)

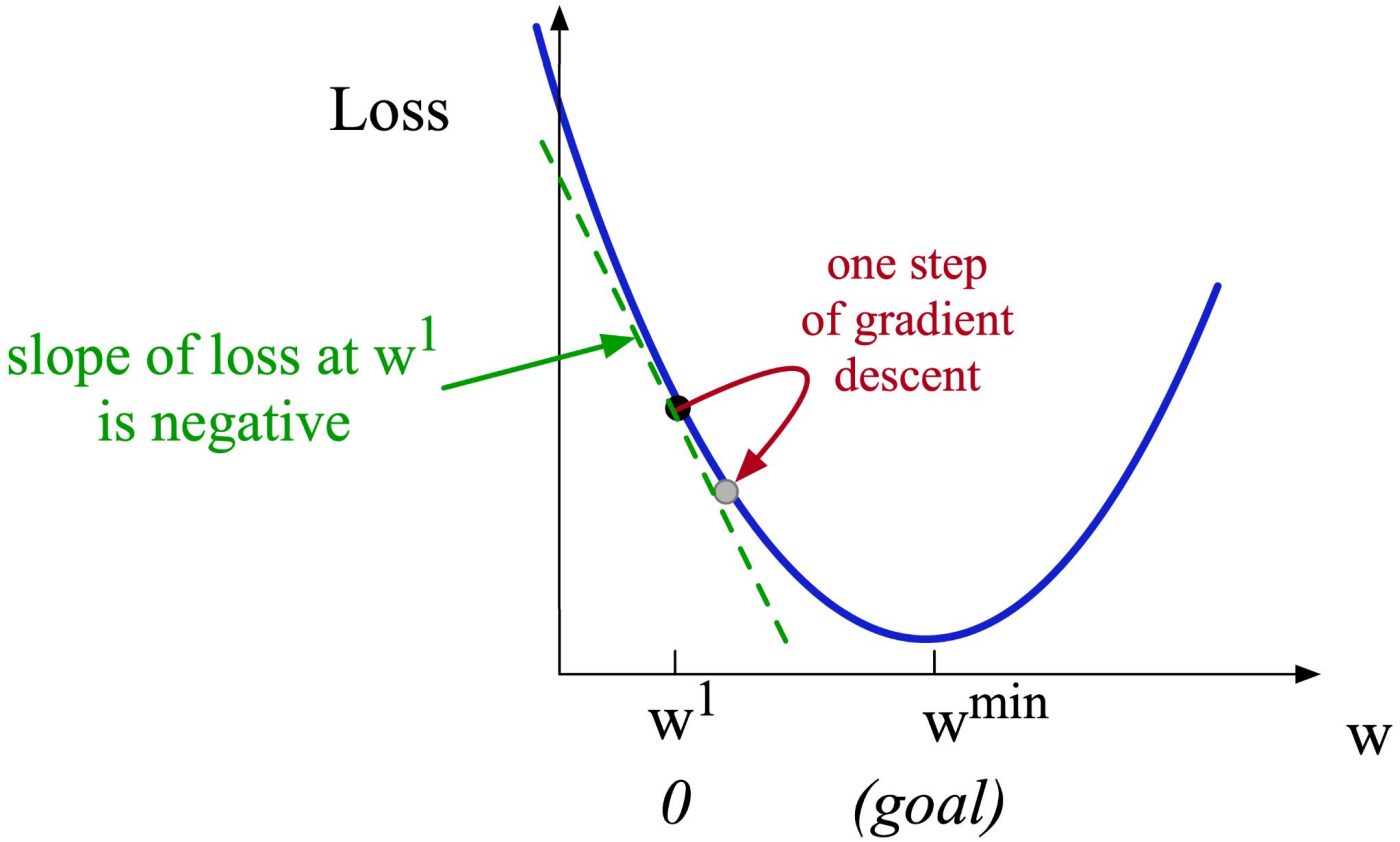




The loss for multi-layer neural networks is non-convex, and gradient descent may get stuck in local minima and never find the global optimum

https://www.cs.umd.edu/~tomg/project/landscapes/

Gradient descent



Gradient descent (cont'd)

$$w_{t+1} = w_t - \eta \cdot \frac{\partial L}{\partial w_t}$$

Where:

- w_t is the parameter at the current time step.
- w_{t+1} is the updated parameter after applying the gradient.
- η is the learning rate, which controls the step size.
- $\frac{\partial L}{\partial w_t}$ is the gradient of the loss function L with respect to the parameter w_t, representing how the loss changes as the parameter changes.

For example, let the loss be

$$L(w) = (w - 3)^2$$

and suppose the current weight is $w = 1$.

1. Current loss:

$$L(1) = (1 - 3)^2 = 4$$

2. Compute the slope:

$$\frac{dL}{dw} = 2(w - 3)$$

so at $w = 1$,

$$\frac{dL}{dw} = 2(1 - 3) = -4.$$

The slope is negative, which means **increasing w decreases the loss**.

3. Gradient descent update, with learning rate $\eta = 0.1$:

$$w_{\text{new}} = w - \eta \frac{dL}{dw} = 1 - 0.1(-4) = \boxed{1.4}.$$

So:

$$\boxed{w : 1 \to 1.4, \qquad L : 4 \to 2.56}$$

The weight moves toward the minimum at $w = 3$.

Backpropagation

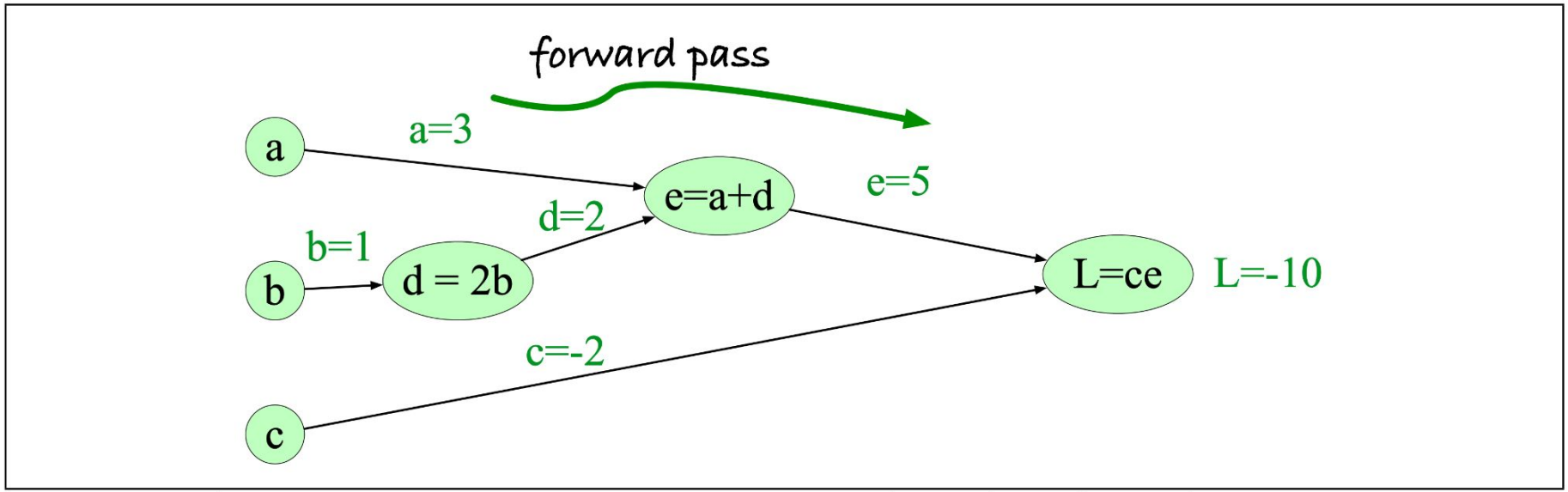


Figure 7.12 Computation graph for the function $L(a,b,c) = c(a+2b)$, with values for input nodes $a = 3$, $b = 1$, $c = -2$, showing the forward pass computation of L.

Backpropagation (cont'd)

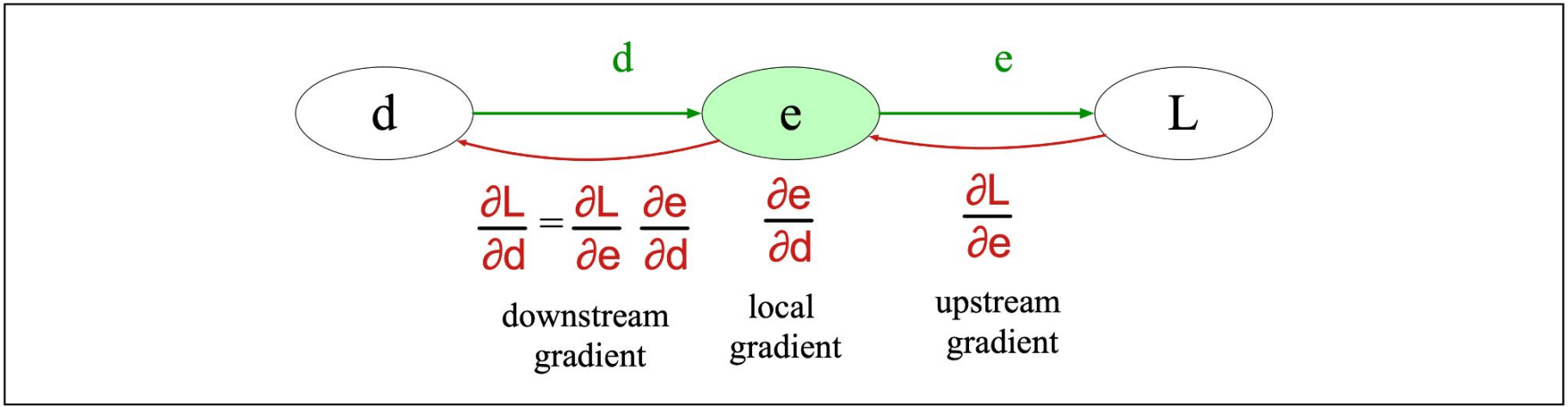
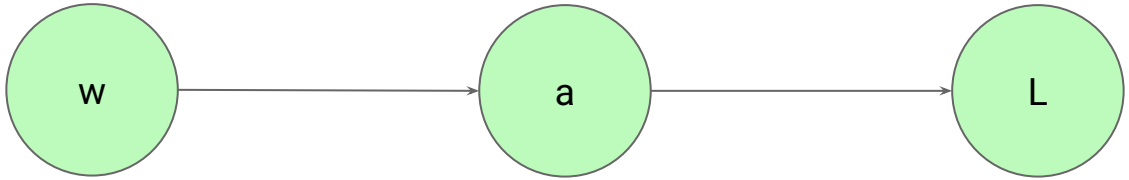


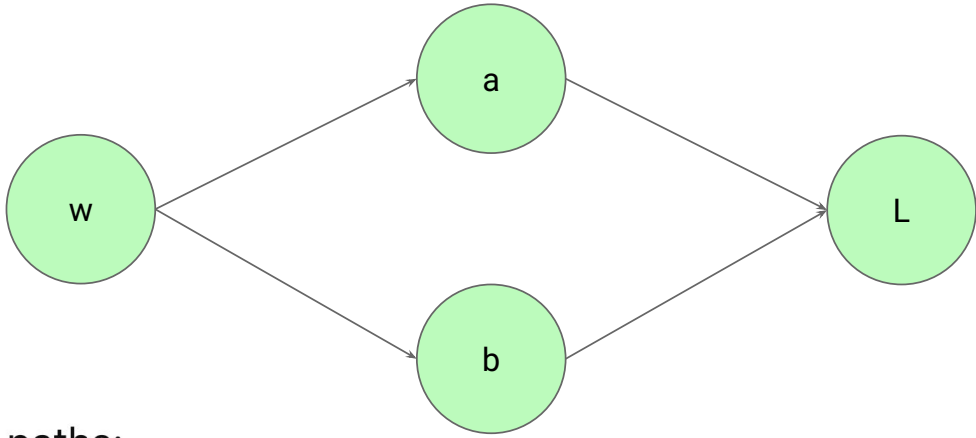
Figure 7.13 Each node (like *e* here) takes an upstream gradient, multiplies it by the local gradient (the gradient of its output with respect to its input), and uses the chain rule to compute a downstream gradient to be passed on to a prior node. A node may have multiple local gradients if it has multiple inputs.



$$\frac{dL}{dw} = \frac{dL}{da} \cdot \frac{da}{dw}.$$

Step-by-step

1. Identify the local derivative from a to L: $\frac{dL}{da}$.
2. Identify the local derivative from w to a: $\frac{da}{dw}$.
3. Multiply them: $\frac{dL}{dw} = \frac{dL}{da} \cdot \frac{da}{dw}$.



Gradient Calculation

By the multivariable chain rule, we sum over all paths:

$$\frac{dL}{dw} = \frac{dL}{da}\frac{da}{dw} + \frac{dL}{db}\frac{db}{dw}$$

Step-by-step

1. Compute $\frac{dL}{da}$ and $\frac{da}{dw}$ for the $w \to a \to L$ path.
2. Compute $\frac{dL}{db}$ and $\frac{db}{dw}$ for the $w \to b \to L$ path.
3. Add them together.

Updating parameters

$$w_{t+1} = w_t - \eta \cdot \frac{\partial L}{\partial w_t}$$

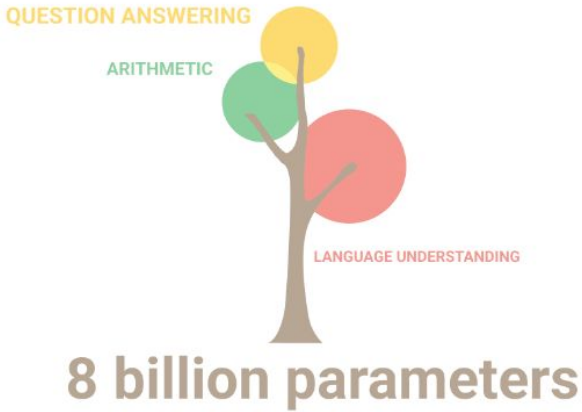
Hyperparameters of gradient descent

- Learning rate
- Batch size
- …

Optimizer

- SGD (Stochastic gradient descent)
- Adam (Adaptive moment estimation)
- Muon (MomentUm Orthogonalized by Newton-Schulz)

Scaling model size unlocks new capabilities



From "PaLM: Scaling Language Modeling with Pathways" by Chowdhery et al. (2022)

Open-weight models are getting bigger

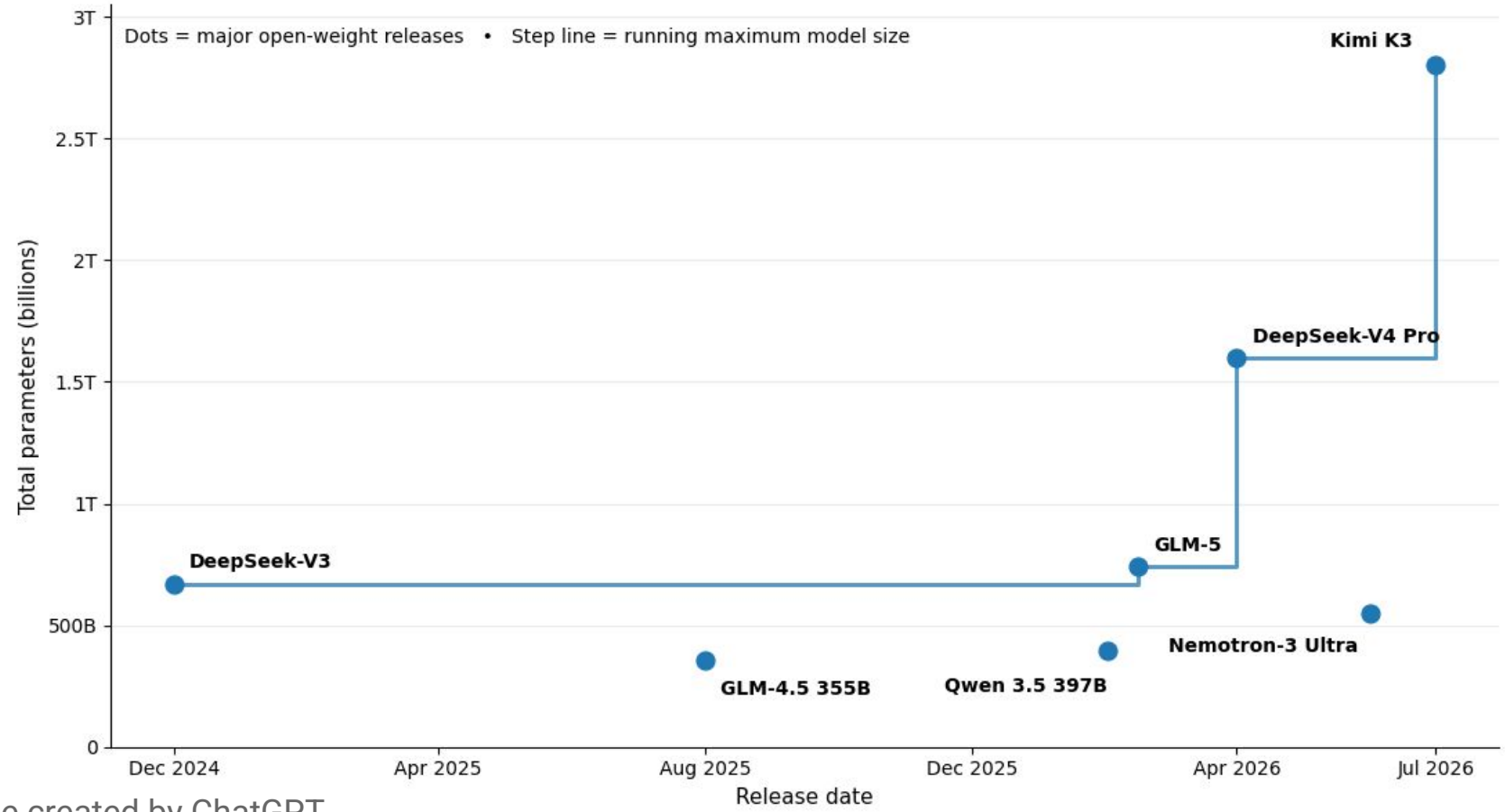


Image created by ChatGPT

Why do LLMs work so well? Pretraining = Massively multi-task learning?

Prefix {choice_1, choice_2}	Task
In my free time, I like to {run, banana}	Grammar
I went to the zoo to see giraffes, lions, and {zebras, spoon}	Lexical semantics
The capital of Denmark is {Copenhagen, London}	World knowledge
I was laughing the entire time, the movie was {good, bad}	Sentiment analysis
The word for "pretty" in Spanish is {bonita, hola}	Translation
First grade arithmetic exam: 3 + 8 + 4 = {15, 11}	Math question

https://www.jasonwei.net/blog/some-intuitions-about-large-language-models

Why do LLMs work so well? Pretraining = Massively multi-task learning? (cont'd)

Prefix	Next word [task]
A transformer is a deep learning architecture, initially proposed in	2017 [factual recall]
A transformer is a deep learning architecture, initially proposed in 2017	, [comma prediction]
A transformer is a deep learning architecture, initially proposed in 2017,	that [grammar]
A transformer is a deep learning architecture, initially proposed in 2017, that	relies [impossible task?]

https://www.jasonwei.net/blog/some-intuitions-about-large-language-models

Thank you!