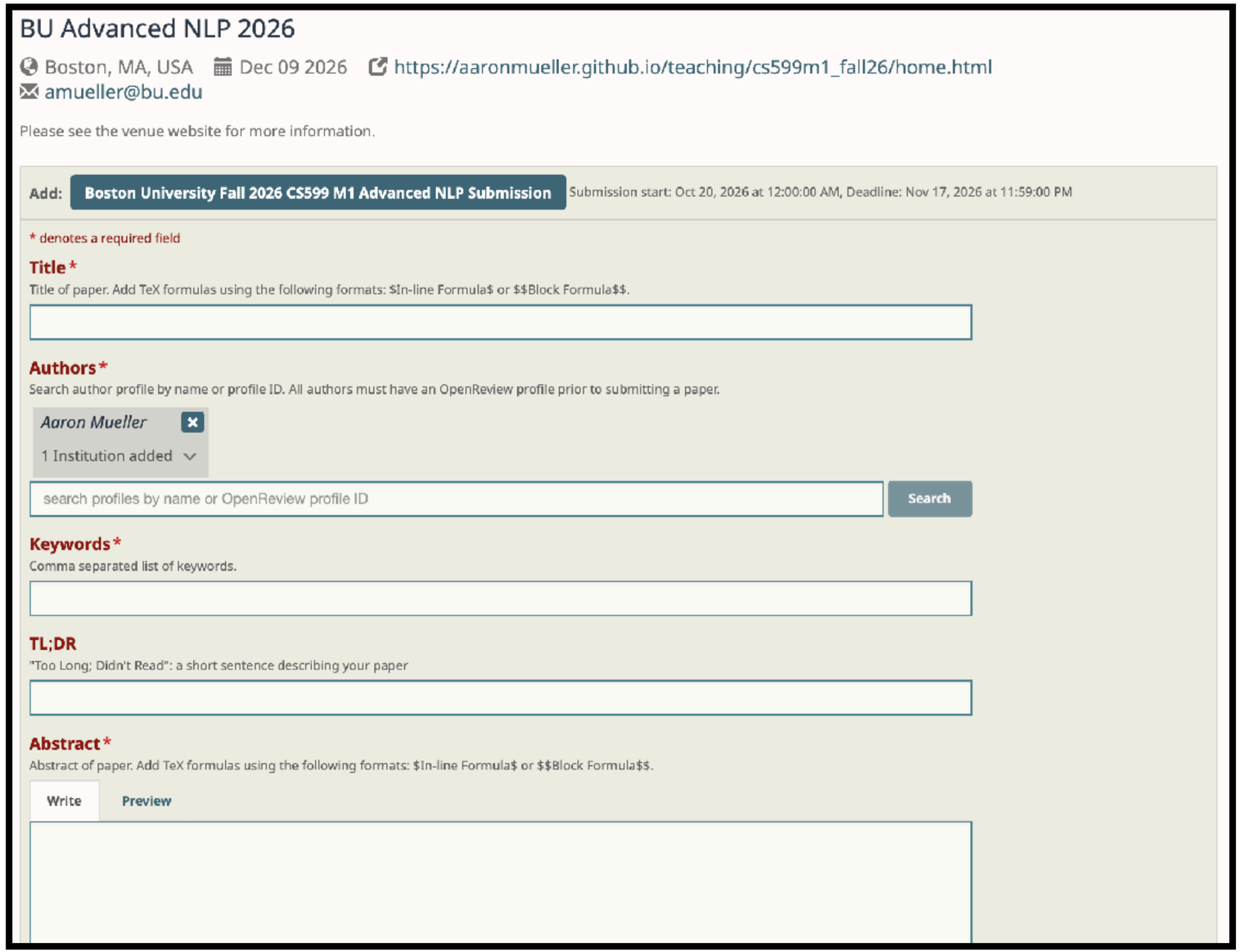
Training Dynamics I

Aaron Mueller
CS599 B1: Advanced Natural Language Processing
Nov. 12, 2026

Midway Report Submission



https://openreview.net/group?id=bu.edu/Boston_University/Fall_2026/CS599_M1_Advanced_NLP

Due **tonight!**

Make sure your PDFs are anonymized!

Also, be sure you've made your OpenReview accounts!

Peer Review Guide

- See the final project instructions doc for details on how this will go and how you should structure your review
- The goal is to help the authors improve their work
- It’s ok to give praise or to be critical, but all comments should be constructive
- It’s ok to say that you don’t understand something! The target audience is you all, so if you don’t get what’s happening, this is a sign that the authors could clarify some things. This will help them improve their final report
- Please do not submit completely AI-generated reviews; this is highly frowned upon in peer review.
    - Plus the authors can get AI reviews themselves. This doesn’t help them much.

Why study training dynamics?

Position: Don't Just "Fix it in Post":
A Science of AI Must Study Training Dynamics

Stella Biderman [1] Mohammad Aflah Khan [2] Niloofar Mireshghallah [3] Catherine Arnett [1] Fazl Barez [4]
Naomi Saphra [5 6]

Neural networks are primarily functions of data.

It's nice to fix things post hoc, but the most robust solution is always to have better data and learn the right kinds of things in the first place.

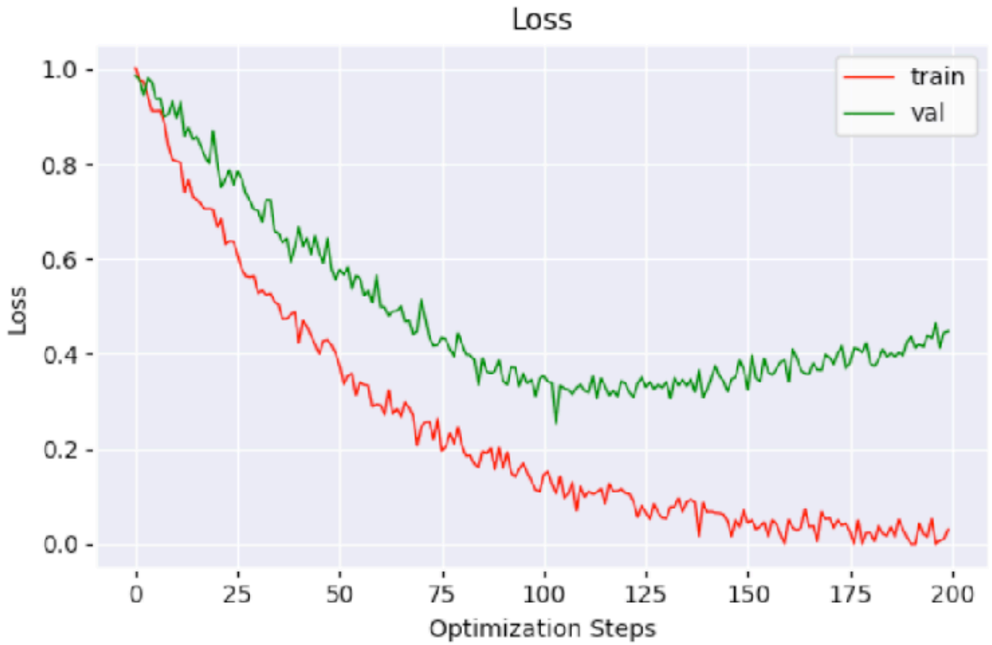
Example: Band-Aids for Racism

When DALL-E 2 generated mostly white men for prompts like "CEO," OpenAI's fix was to secretly appending words like "Black" or "female" to user prompts (Baio, 2022; Offert & Phan, 2022). This produces superficially diverse outputs while leaving the underlying bias intact. This is highly reminiscent of issues with Google Photos, which still cannot recognize gorillas over a decade after Black users were infamously misclassified as apes (Grant & Hill, 2023). After years of unsuccessful remediation, Google ultimately disabled primate recognition altogether rather than figure out how to fix the root cause (Hern, 2018).

The Classic View of Learning

- The longer you train, the lower the training loss will get

- Validation loss decreases mostly monotonically until overfitting starts

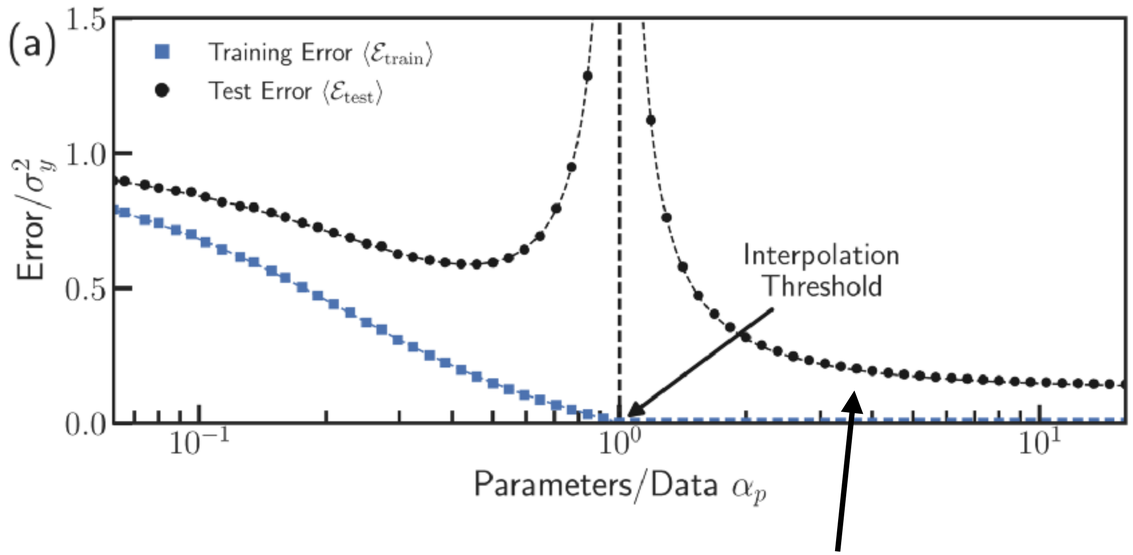


Phase Transitions

- As it turns out, there are quite a few *non*-monotonic trends when we scale learning up to very large datasets and parameters!

- We sometimes see what looks like **phase transitions**, or sudden non-continuous shifts in behavior that were not predictable from previous steps

- For example: **double descent**, where if we keep adding parameters far past overfitting, test error does *not* keep getting worse, and in fact begins a second descent



What's happening here?

Learning Modular Addition (via Language Modeling)

- Dataset: binary operations $x \circ y = z$, or modular addition
  - E.g., $a + b \bmod 97 = ?$

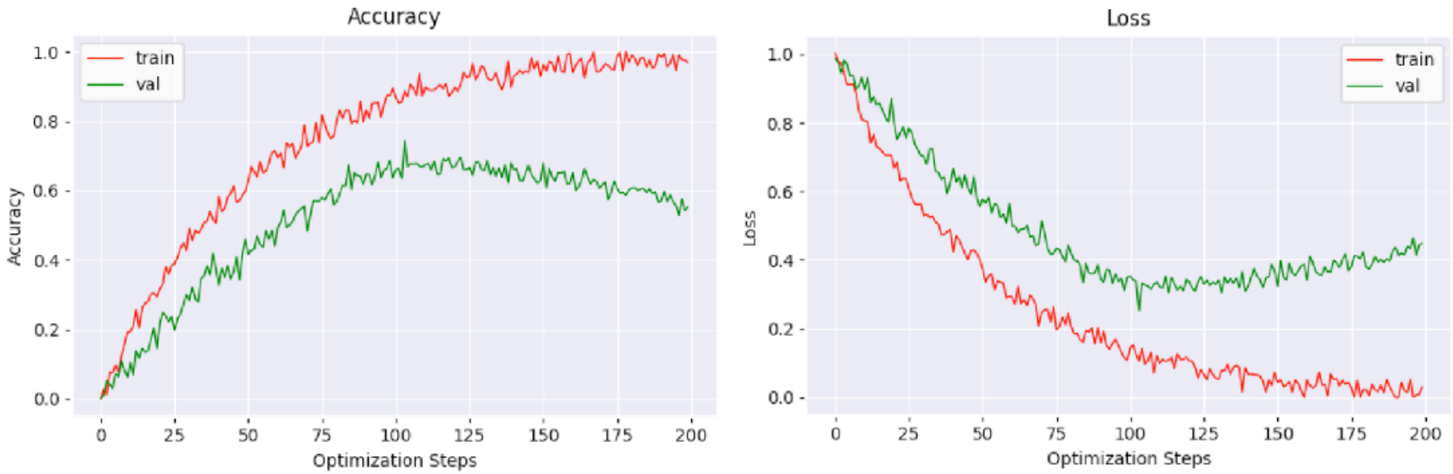
- We can expose language models to many instances of this task, and have it predict the correct answer via next token prediction
  - Same cross-entropy loss as usual

Grokking

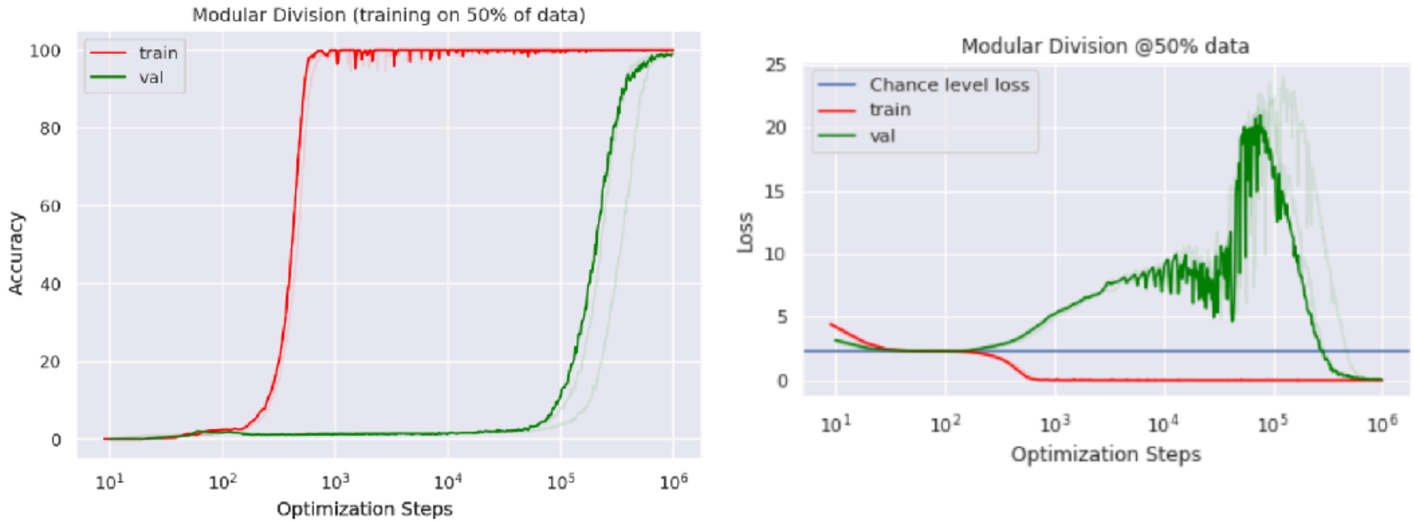
[Power et al., 2022]

- We see classic overfitting for many thousands of steps...
- ...until suddenly, test accuracy becomes near-perfect
- This suddenly improved generalization is known as **grokking**

Standard overfitting

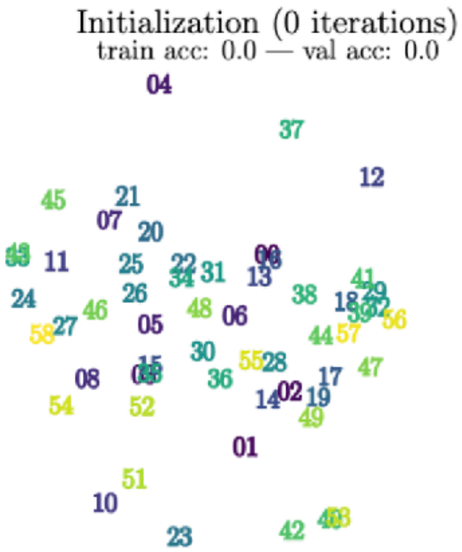


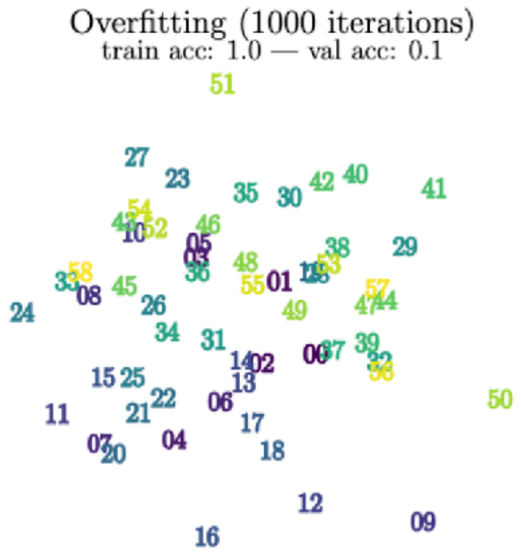
Grokking



Grokking

[Liu et al., 2022]

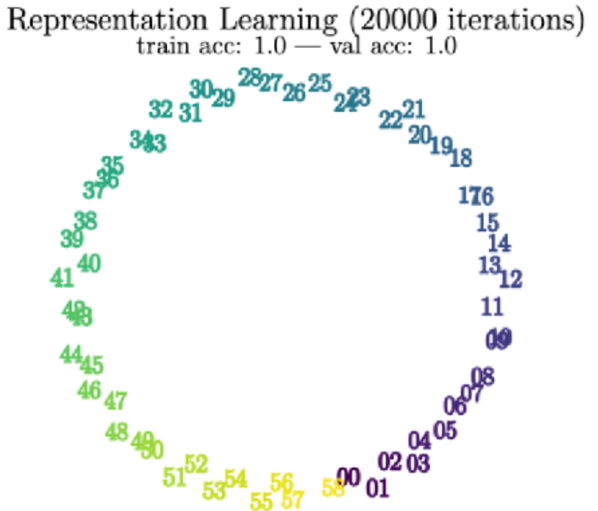




- Right around when validation performance increases, a periodic representation structure emerges:

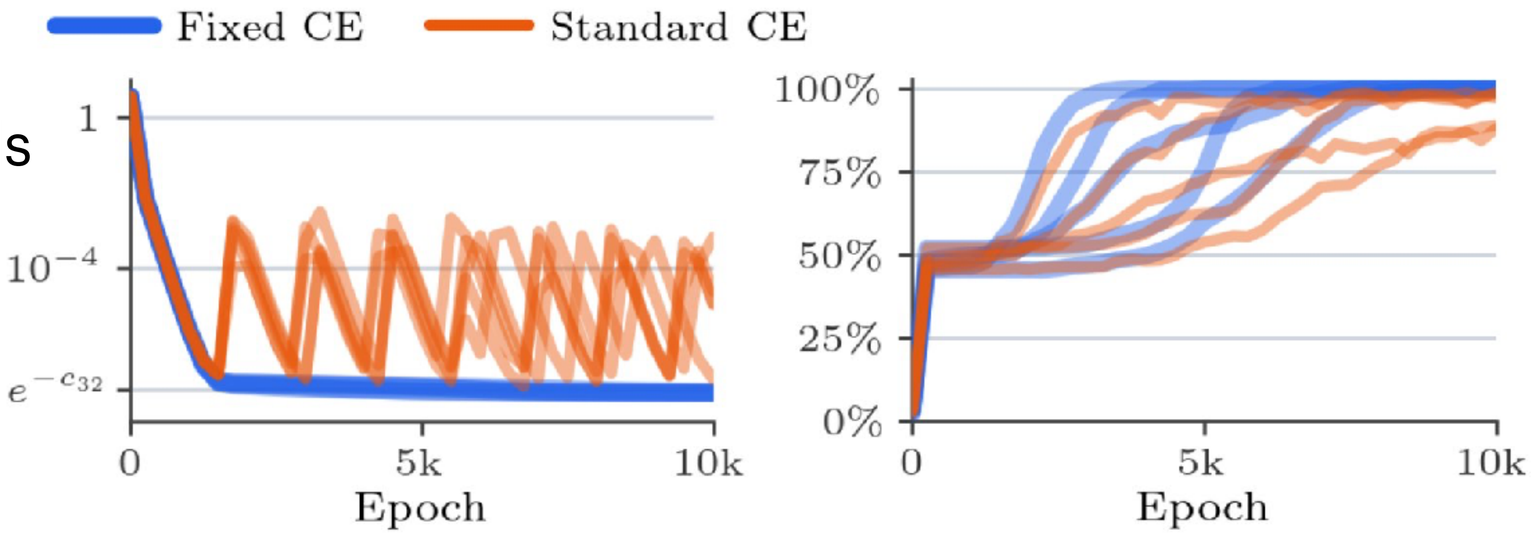
- Is representation structure driving high performance?

- If tokens are embedded as $\cos(\omega_k a), \sin(\omega_k a)$, trig identities give $\cos(\omega_k(a+b-c))$, which peaks at $c = a + b \bmod p$
  - Nanda et al. posit that models learn this in three phases: memorization, circuit formation, and cleanup via weight decay

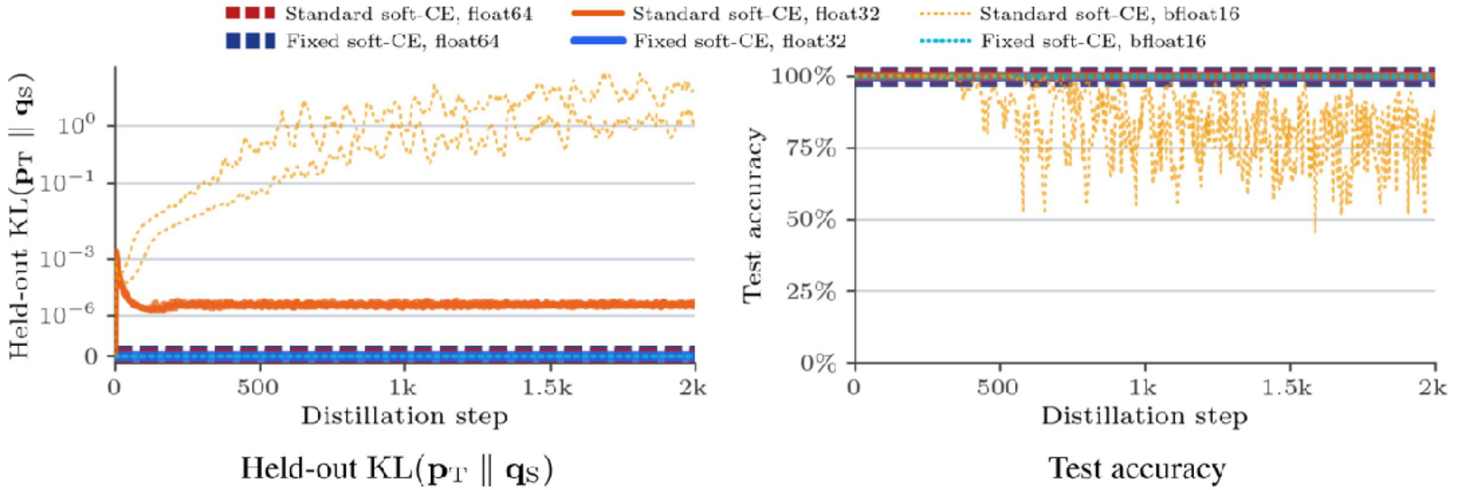


Oops, bugs in softmax implementations delayed grokking

When one logit is much larger than others, its gradient rounds to 0, but others don't, which leads gradients to not sum to 0. Fixing this removes spikes and accelerates grokking.

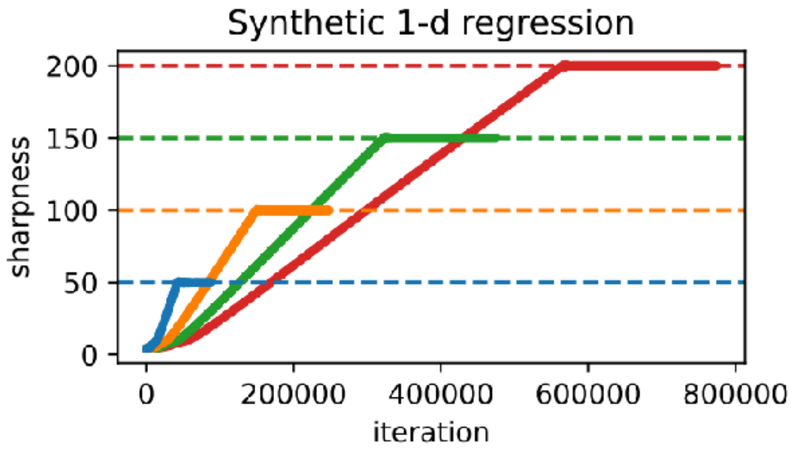


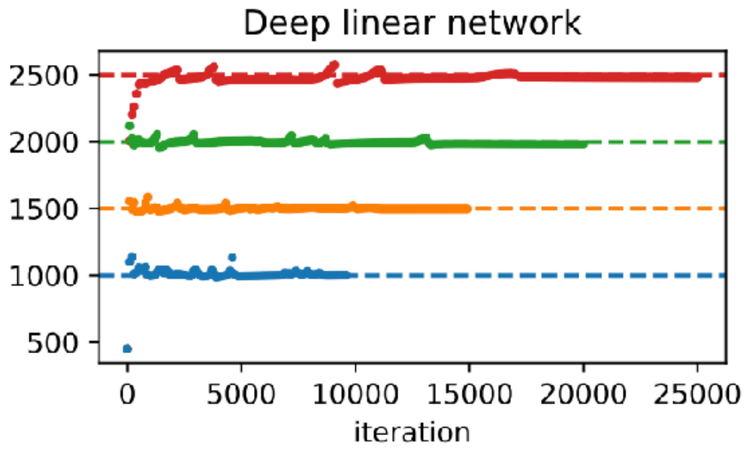
When output logits are close to target logits, we get non-zero gradients even when the output is perfectly on target. Fixing this again accelerates learning.

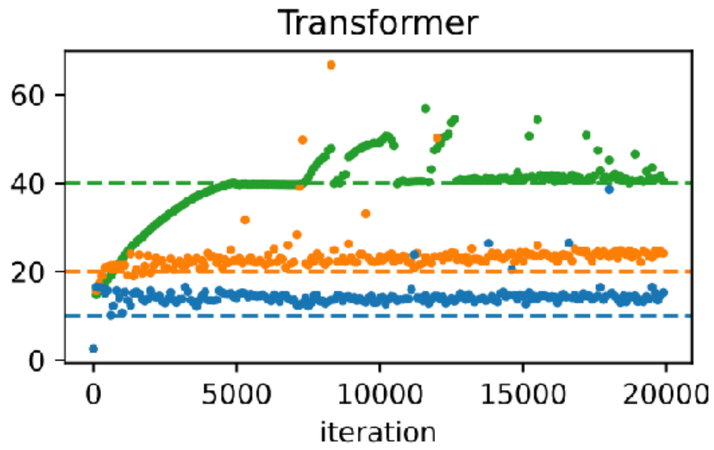


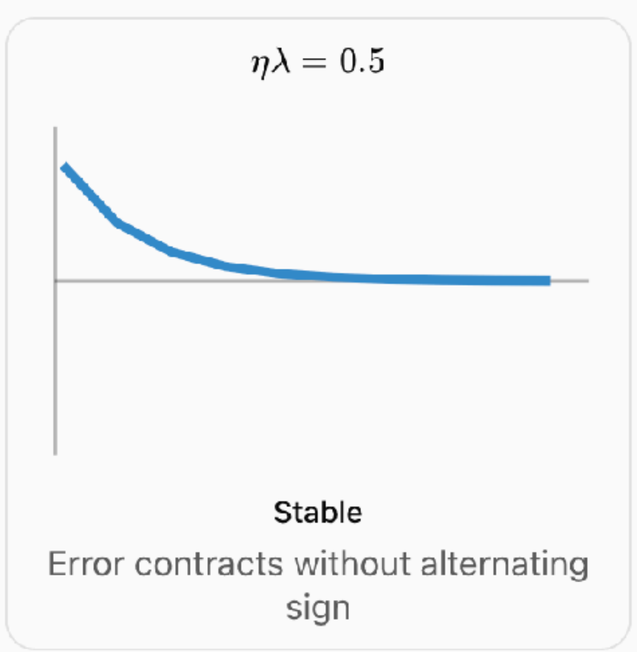
[Barron et al., 2026]

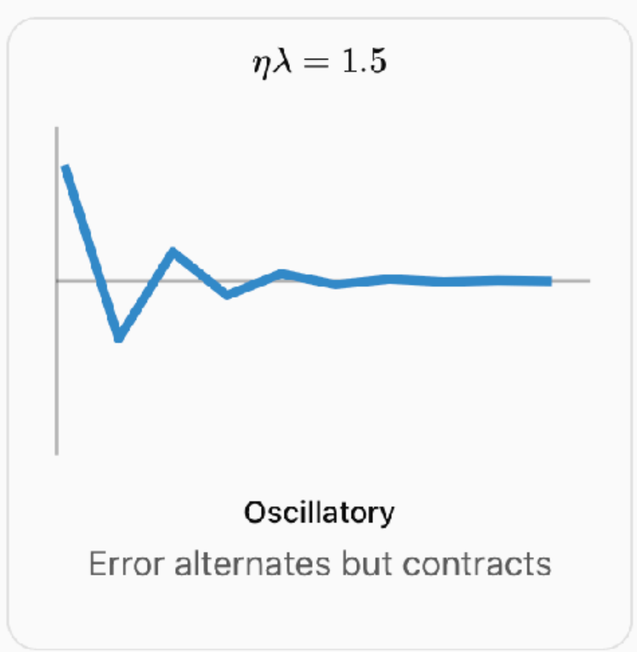
Edge of Stability

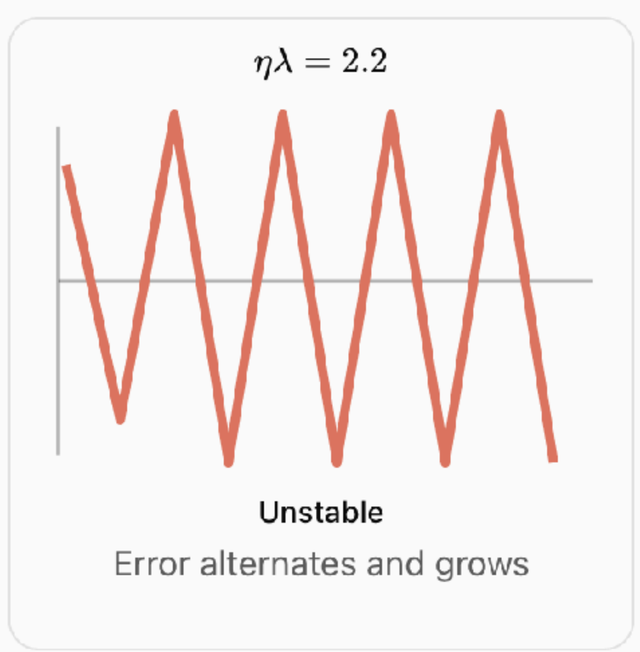












$$L(\theta) = \frac{1}{2}\lambda\theta^2$$

$$\theta_{t+1} = (1 - \eta\lambda)\theta_t$$

ead hovers right

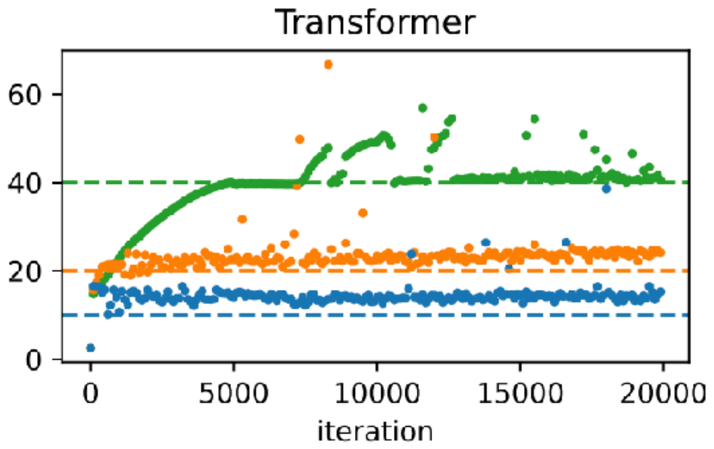
e while no longer

Edge of Stability

- In this regime, we have frequent short-term oscillations, but still see long-term improvement in performance

- Why does a neural network learn successfully even when conventional assumptions about gradient descent are violated?

- Can these optimization dynamics explain which solutions a network will discover or use?



Simplicity Bias

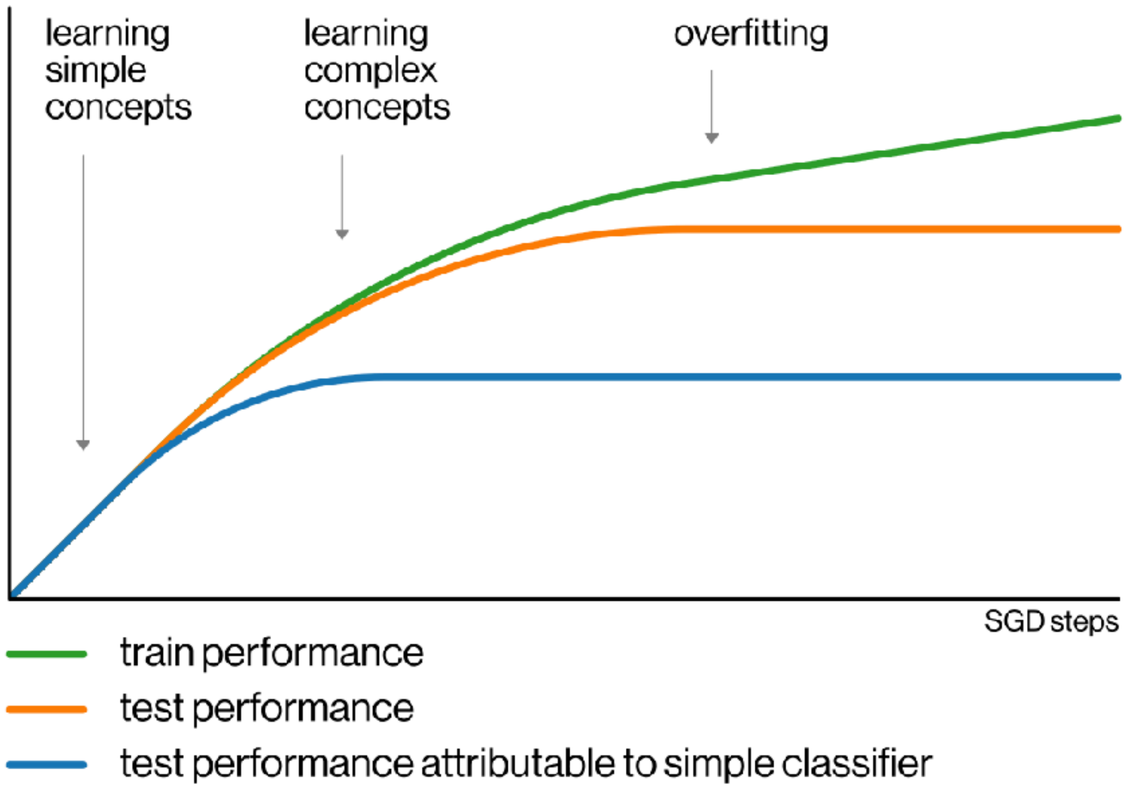
- Progress in learning with SGD can be attributed to a simple classifier at first. Then, SGD continues to learn more complex classifiers
- Finally, the classifier will interpolate the training data
- Why?

$$\frac{\partial \hat{y}}{\partial t} = -K(\hat{y} - y)$$

Neural tangent kernel of a linearized network

$$K = U\Lambda U$$

Diagonalize the kernel



$$e_i(t) = e_i(0)e^{-\lambda_i t}$$

Prediction error evolves like this. So directions w/ larger kernel eigenvalues are learned earlier

Types of Simplicity Bias

- A preference for simple solutions shows up in many ways:
  - Neural nets learn linear functions before they learn highly nonlinear functions
  - Neural nets exploit means and pairwise relationships before higher-order correlations
  - Neural nets learn lower-frequency periodic features before higher-frequency periodic features

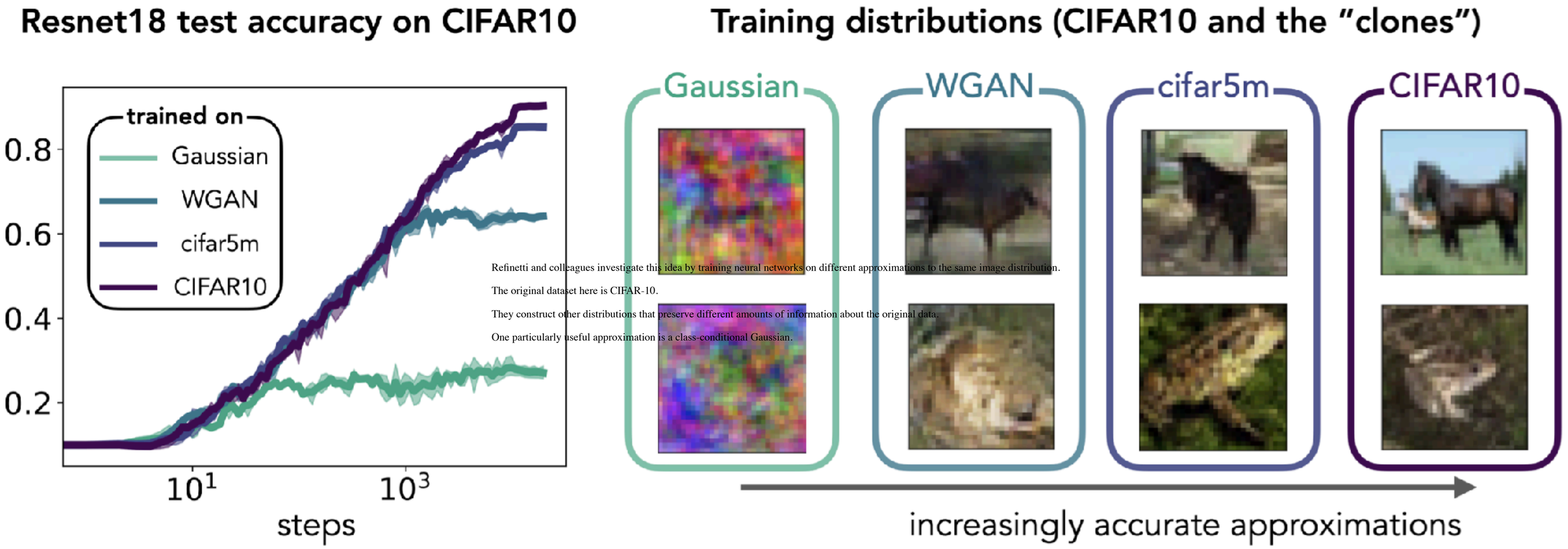
Distributional Simplicity Bias

- Maybe models learn statistics of increasing orders:

- **First-order statistics:** means, such as the average value of a feature
- **Second-order statistics:** covariances, which describe how pairs of features vary together
- **Higher-order statistics:** interactions involving three or more variables, which may not be captured directly by means and covariances

- Hypothesis: neural networks initially exploit lower-order statistics, then increasingly rely on higher-order statistics as training progresses

Neural networks learn <u>distributions</u> of increasing complexity



Refinetti and colleagues investigate this idea by training neural networks on different approximations to the same image distribution.

The original dataset here is CIFAR-10.

They construct other distributions that preserve different amounts of information about the original data.

One particularly useful approximation is a class-conditional Gaussian.

- We move from modeling average pixel values to correlations between pixels in particular locations. We then move to more complex patterns such as spatial arrangements that involve many features at once

[Refinetti et al., 2023]

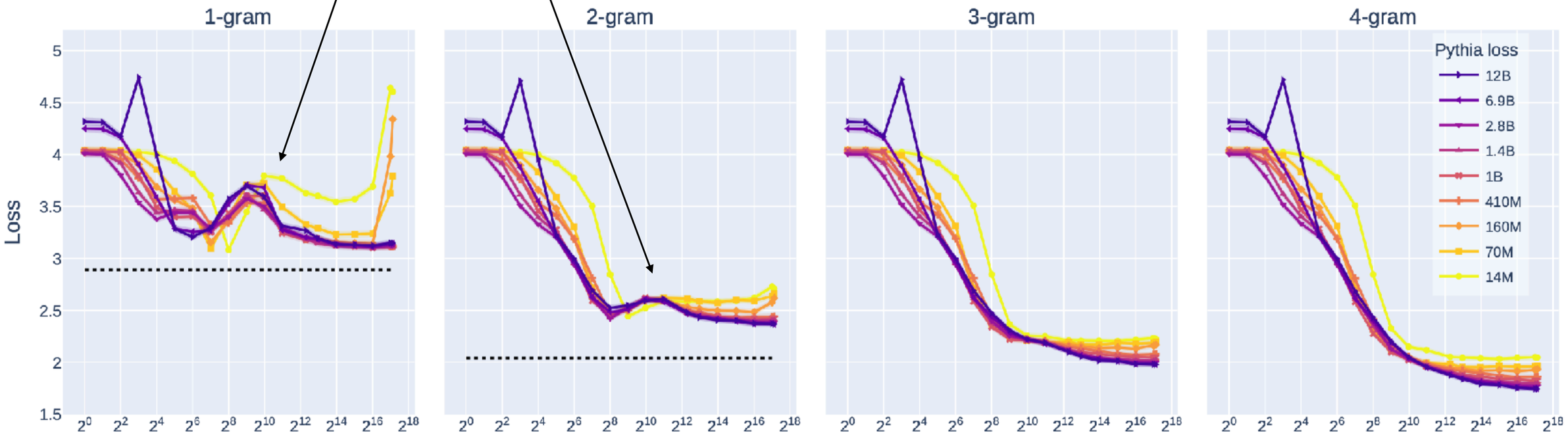
Neural networks learn statistics of increasing complexity

Fit n-gram models to the Pile, sample sequences from them, measure Pythia's loss on them.

Double descent!

In language models, later checkpoints increasingly exploit higher-order dependencies.

[Belrose et al., 2024]

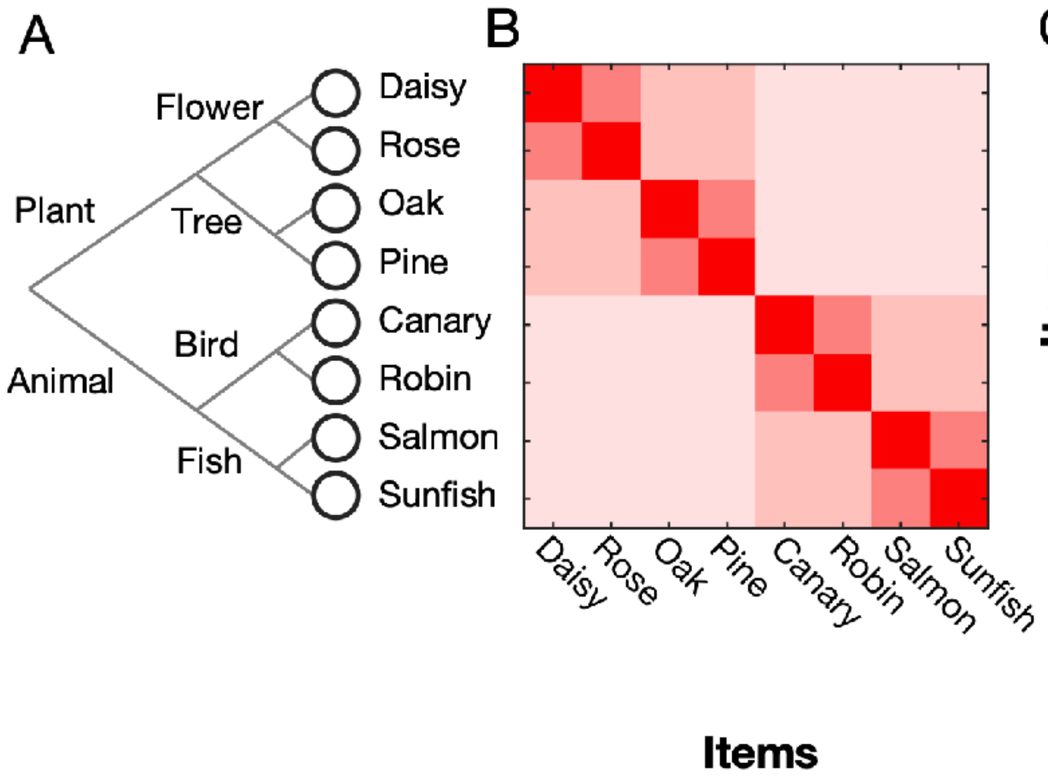


Token frequencies

Token pair co-occurrences

Longer patterns

Neural networks learn progressively finer-grained distinctions



Saxe et al. [2018] show that each statistical mode's effective singular value is a strong predictor of when/in what order that mode will be learned.

Simplicity Bias

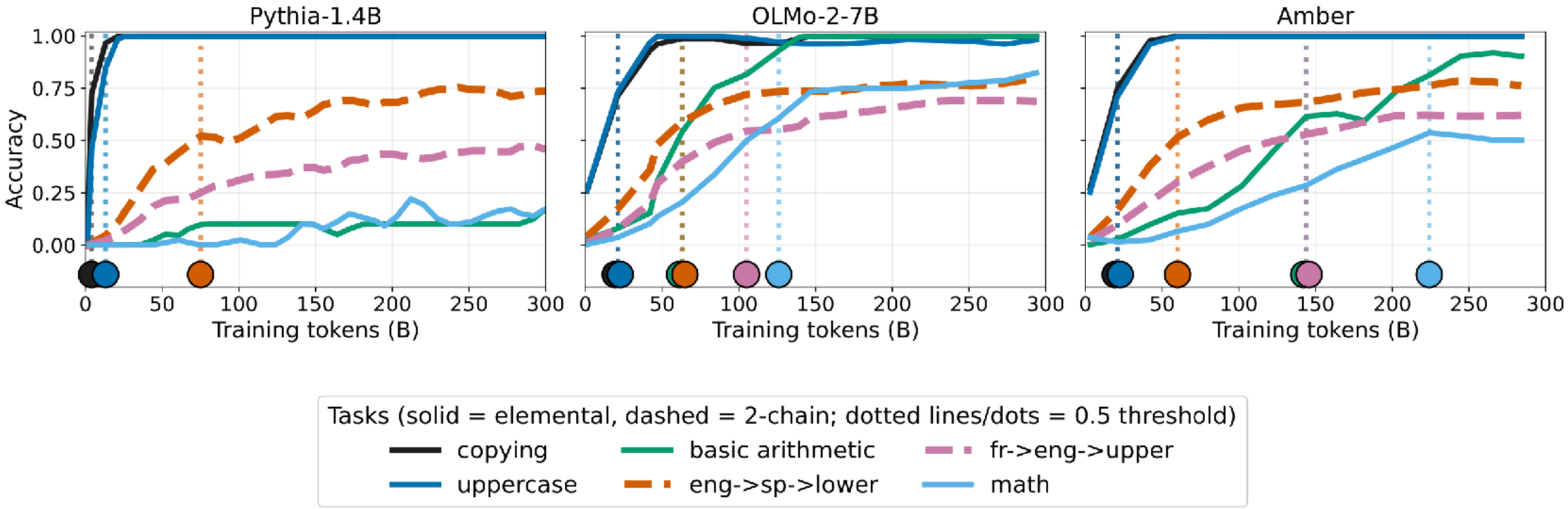
- All of this suggests a general model of learning. Maybe whether a model learns something is directly dependent on how complex the solution is:

$$L = L_{\text{prediction}} + \lambda L_{\text{complexity}}$$

- Even if we have no such explicit term in the loss, we can conceptualize simplicity bias as an *implicit* preference for simpler solutions
- At first, a model can quickly reduce loss by learning simple things. But to further decrease loss, it will need to learn progressively more complex phenomena

Implicit Curricula

[Liu et al., 2026]



- Language models appear to acquire certain capabilities in a particular order across architectures and sizes

Implicit Curricula

	OLMo2			OLMo3	LLM360		Pythia			
	1B	7B	13B	7B	Amber	Crystal	410M	1.4B	2.8B	12B
O2-1B	—	.80	.72	.75	.88	.76	.84	.92	.78	.84
O2-7B		—	.93	.93	.88	.92	.71	.72	.77	.84
O2-13B			—	.93	.82	.89	.64	.66	.76	.83
O3-7B				—	.83	.92	.70	.72	.79	.85
Amber					—	.87	.77	.84	.87	.90
Crystal						—	.70	.75	.83	.87
P-410M							—	.88	.72	.79
P-1.4B								—	.77	.82
P-2.8B									—	.91
P-12B										—



The ordering of capabilities is highly correlated across models

Aside: Do Real Curricula Work?

- What if we sort the training data such that the model sees easier examples earlier on, and harder examples later on?
  - This is known as **curriculum learning**

- Very early work in the 2000s found some success with this, but in language modeling, it does not seem to work
  - The first 3 BabyLM Challenges saw many submissions based on curriculum learning, and very few of them consistently improved performance!

- If models learn easier patterns first anyway, why doesn't presenting easy examples first help?

Lazy-to-rich Learning

- The previous simplicity bias explanations assumed that the NTK remains fixed
- But Kumar et al. [2024] propose that grokking can arise from a transition between two regimes:
  - **Lazy regime:** in overparameterized models, networks behave approximately like a linearized model around initialization, relying on features that change relatively little
  - **Rich regime:** feature representations evolve substantially, allowing the model to discover solutions that would have been difficult to learn with its initial features

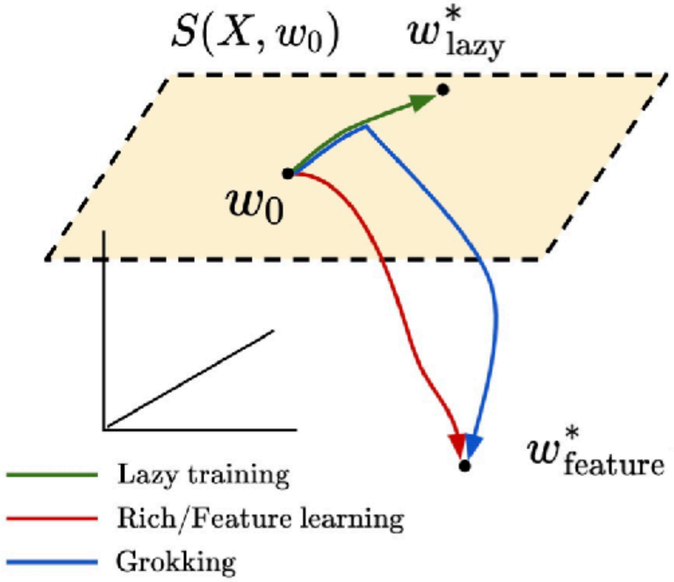
$$f_\theta(x) \approx f_{\theta_0}(x) + J_{\theta_0}(x)(\theta - \theta_0)$$

Linear update in the lazy regime

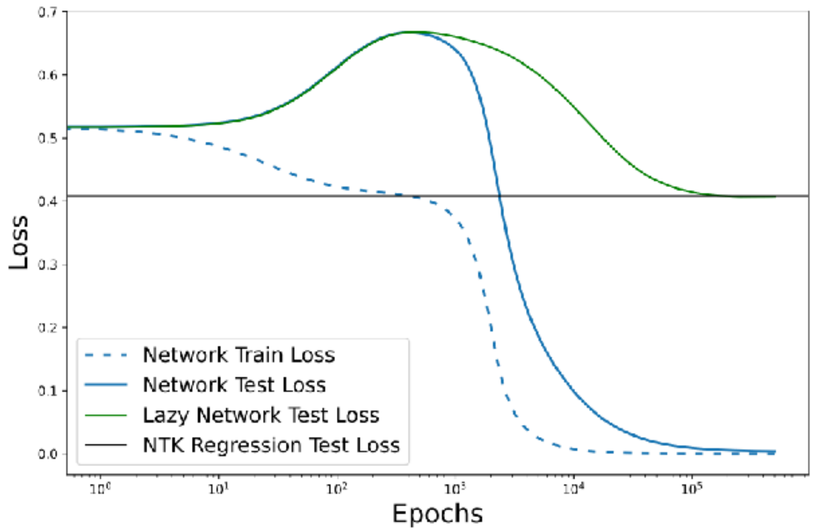
In the rich regime,

$$J_{\theta_t}(x) \not\approx J_{\theta_0}(x)$$

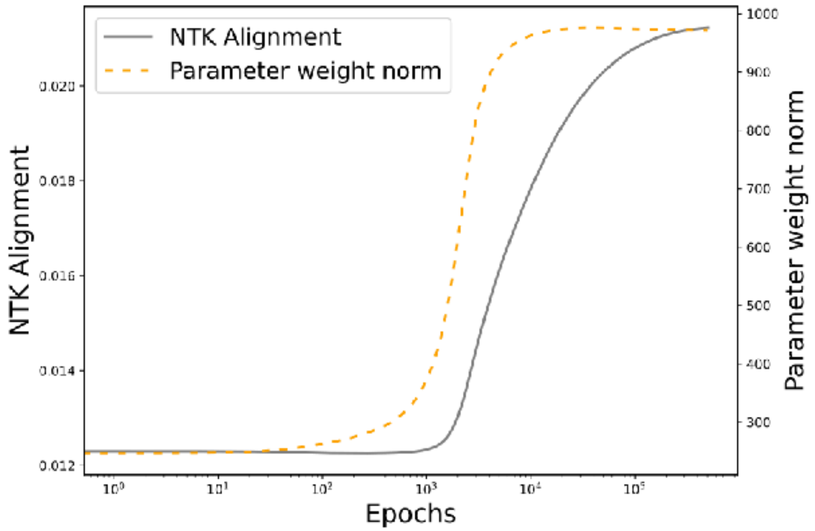
Lazy-to-rich Learning



(a)



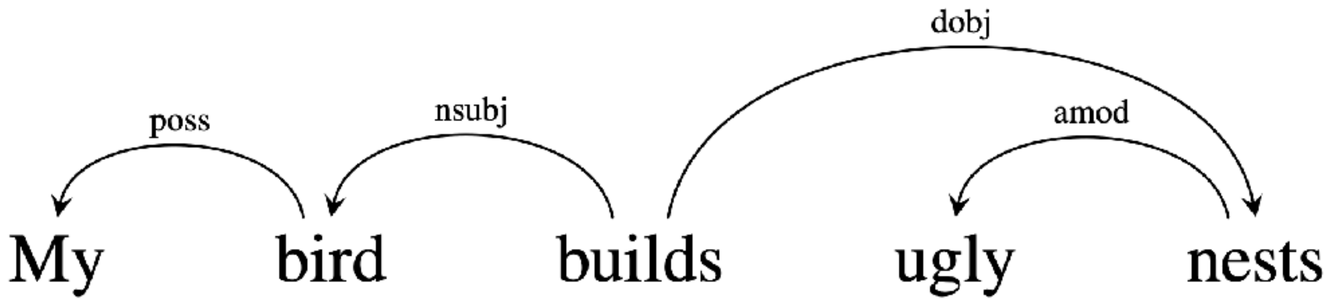
(b)



(c)

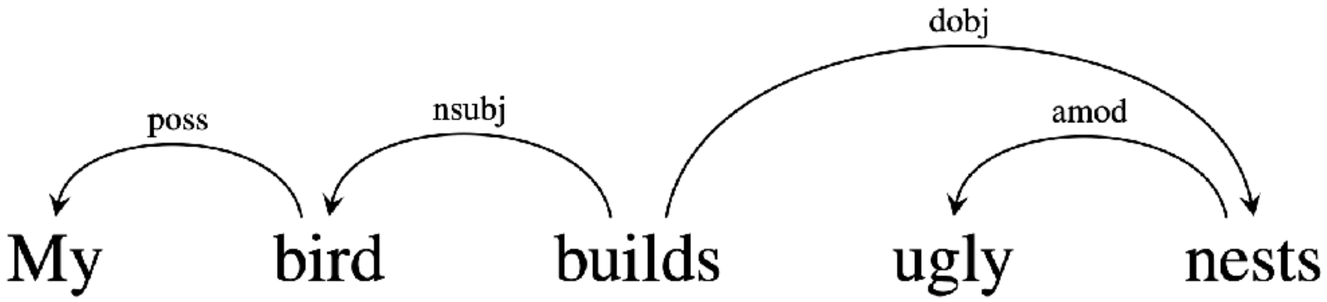
- In the lazy regime, models don't adapt their features to the data, leading to suboptimal solutions
- In the rich regime, models learn the proper features
- In grokking, early training follows lazy dynamics. The model later adapts its features to enable a solution that generalizes far better

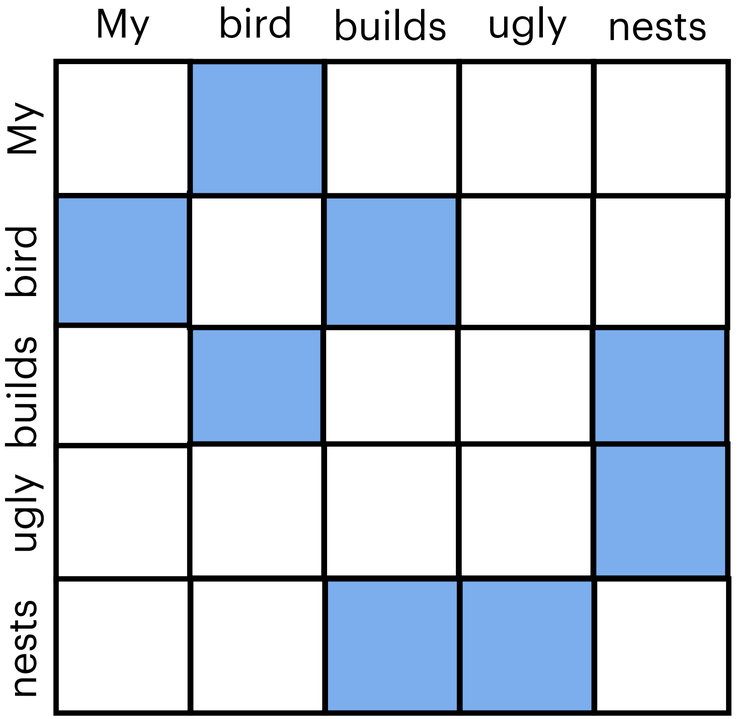
Dependency Parsing



- Language has hierarchical structure
- We commonly use **dependency parses** to represent this structure
- In the above example, we have *dependencies* (the arrows) from heads to their dependents
    - All dependencies proceed from one head to one dependent
    - A word can only be the dependent of one other word (but can be the head of multiple dependents)

Dependency Structure in Neural Networks





If we inspect the attention weights in a trained language model, they often correlate with this dependency structure.

We can measure the alignment of attention with syntactic structure as:

$$\gamma(x_i, x_j) = \max_{h,l} \alpha_{ij}^{(h,l)} + \max_{h,l} \alpha_{ji}^{(h,l)}$$

We'll call this syntactic attention structure (SAS)

Syntactic Evaluation

- We'll use the BLiMP dataset to measure the syntactic capabilities of models.
- Models are scored according to whether they assign higher probability to the grammatical sequence over the ungrammatical sequence.

Phenomenon	N	Acceptable Example	Unacceptable Example
ANAPHOR AGR.	2	*Many girls insulted themselves.*	*Many girls insulted herself.*
ARG. STRUCTURE	9	*Rose wasn't disturbing Mark.*	*Rose wasn't boasting Mark.*
BINDING	7	*Carlos said that Lori helped him.*	*Carlos said that Lori helped himself.*
CONTROL/RAISING	5	*There was bound to be a fish escaping.*	*There was unable to be a fish escaping.*
DET.-NOUN AGR.	8	*Rachelle had bought that chair.*	*Rachelle had bought that chairs.*
ELLIPSIS	2	*Anne's doctor cleans one important book and Stacey cleans a few.*	*Anne's doctor cleans one book and Stacey cleans a few important.*
FILLER-GAP	7	*Brett knew what many waiters find.*	*Brett knew that many waiters find.*
IRREGULAR FORMS	2	*Aaron broke the unicycle.*	*Aaron broken the unicycle.*
ISLAND EFFECTS	8	*Which bikes is John fixing?*	*Which is John fixing bikes?*
NPI LICENSING	7	*The truck has clearly tipped over.*	*The truck has ever tipped over.*
QUANTIFIERS	4	*No boy knew fewer than six guys.*	*No boy knew at most six guys.*
SUBJECT-VERB AGR.	6	*These casseroles disgust Kayla.*	*These casseroles disgusts Kayla.*

Each of these pairs is known as a **minimal pair**: they differ only according to a single feature.

$$p(\text{grammatical}) >^{?} p(\text{ungrammatical})$$

Do Syntactic Representations Drive Syntactic Performance?

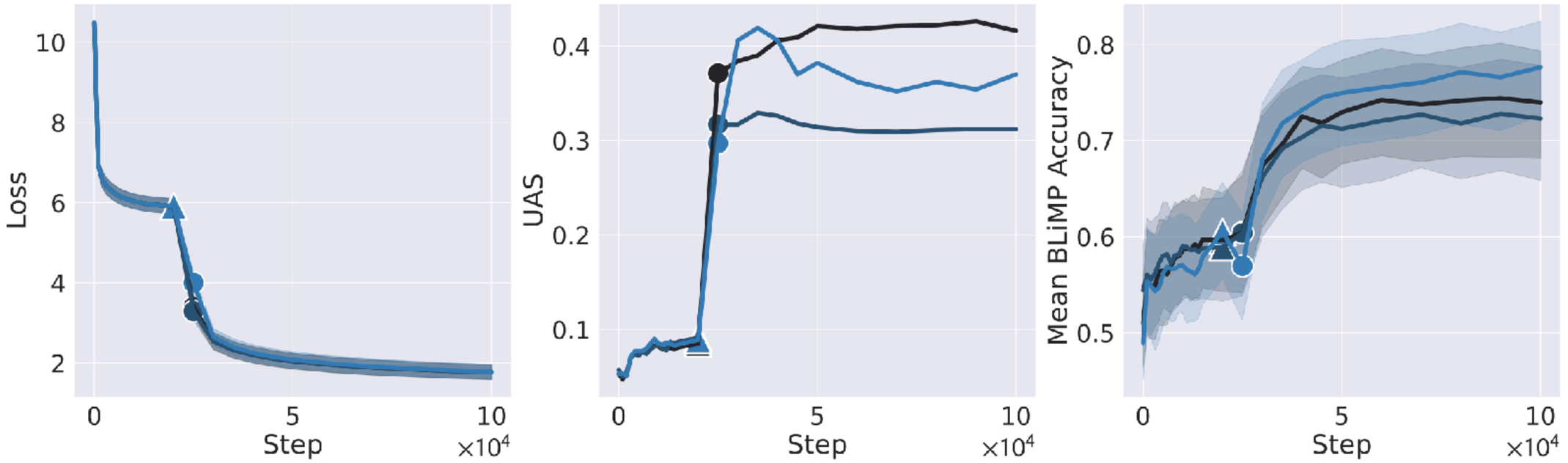


Figure 1: **BERT first learns to focus on syntactic neighbors with specialized attention heads, and then exhibits grammatical capabilities in its MLM objective**. The former (internal) and the latter (external) model behaviors both emerge abruptly, at moments we respectively call the **structure onset** (▲) and **capabilities onset** (●) (quantified as described in Section 2.3). We separately visualize

We can first observe that syntactic representations precede syntactic capabilities

Do Syntactic Representations Drive Syntactic Performance?

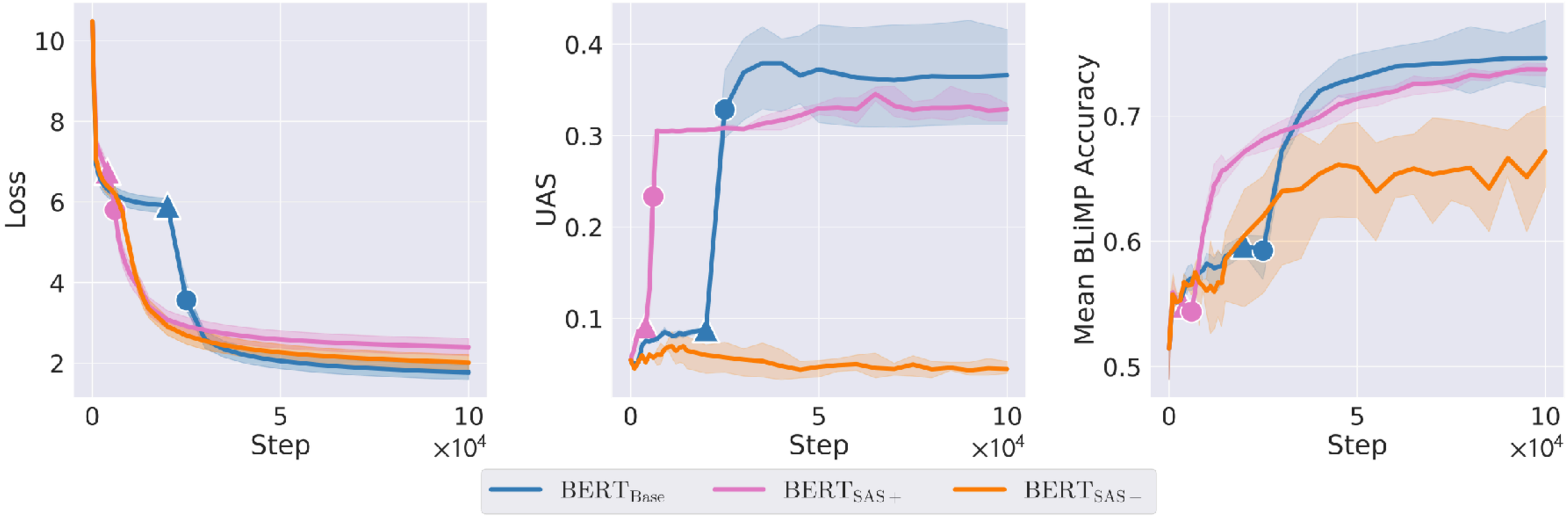
- Now, we can define a syntactic regularization term in our training loss:

$$L(x) = L_{MLM}(x) + \lambda \sum_{i=1}^{|x|} \sum_{x_j \in D(x_i)} \gamma(x_i, x_j)$$

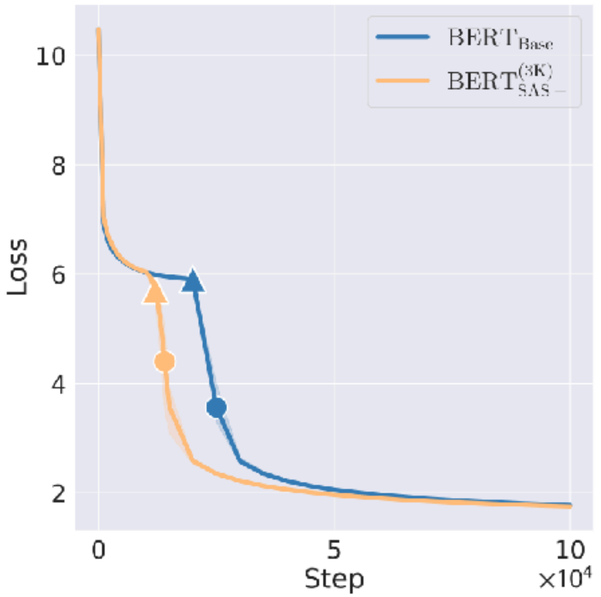
Language modeling loss

Syntactic regularization

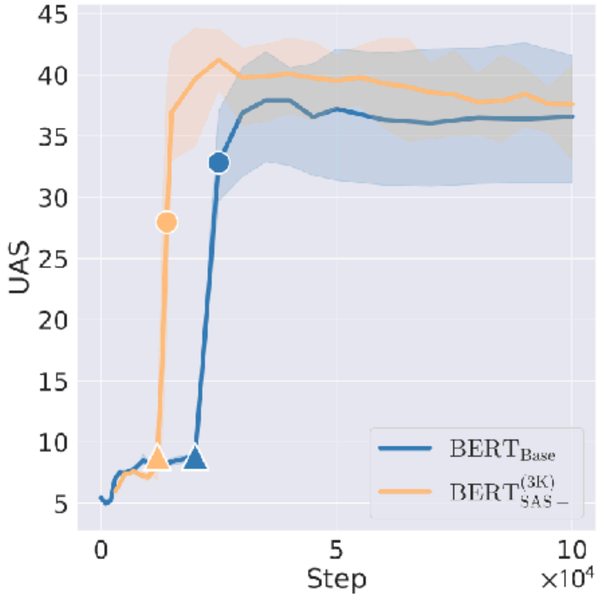
- The syntactic regularization term penalizes the model for having attention weights that correlate with dependency structure
- We can directly control how much the model is discouraged or incentivized to learn this dependency structure by modifying the λ coefficient



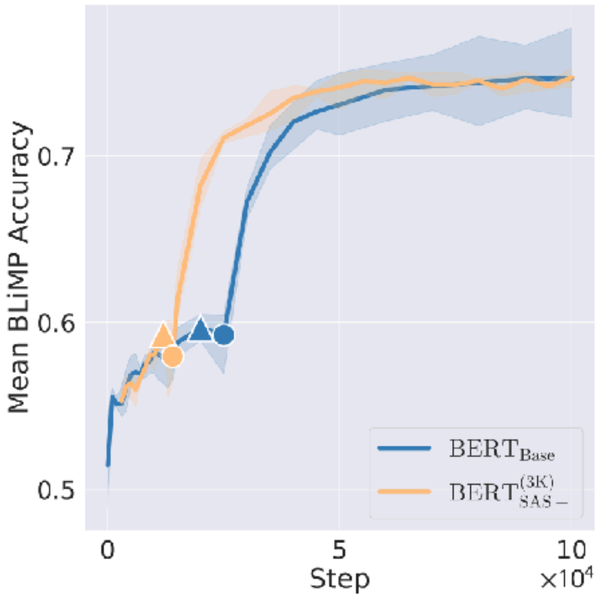
- The blue line is the model as it naturally learns (without the regularizer)
- The pink line is when we encourage the model to learn this syntactic structure by subtracting the regularizer
- The orange line is when we discourage the model from learning this syntactic structure by adding the regularizer
- Encouraging this structure does accelerate syntactic capability acquisition, but also *reduces* overall final performance
- The natural model achieves the lowest final loss



(a) MLM validation loss.



(b) Implicit parse accuracy.



(c) Mean BLiMP challenge accuracy.

- *Temporarily* suppressing syntactic structure can accelerate acquisition of syntactic structure in the attention, and sometimes accelerate the onset of capabilities *without loss in final performance*!
- Here, the syntactic structure is regularized for the first 3K steps

Competing Learning Strategies

- SAS is a relatively simple learning strategy, but learning it too early may interfere with the acquisition of other useful patterns
- Temporarily suppressing it changes the learning trajectory
- If multiple strategies can reduce training loss, what determines which ones the model learns first? Can learning a useful strategy too early ultimately hurt us?

Next Time

- When multiple solutions explain a training set equally well, what drives the selection of one of these solutions?
- Induction heads
- The learning of in-context learning