Towards an Artificial Neuroscience: Analytics for Language Model Interpretability

by

Robert Wesley Gurnee

Submitted to the Sloan School of Management
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY IN OPERATIONS RESEARCH

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2025

Authored by: Robert Wesley Gurnee
Operations Research Center
November 13, 2024

Certified by: Dimitris J. Bertsimas
Boeing Professor of Operations Research, Thesis Supervisor

Accepted by: Patrick Jaillet
Dugald C. Jackson Professor of Electrical Engineering and Computer Science
Co-Director, Operations Research Center

Towards an Artificial Neuroscience: Analytics for Language Model Interpretability

by

Robert Wesley Gurnee

Submitted to the Sloan School of Management
on November 13, 2024 in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY IN OPERATIONS RESEARCH

ABSTRACT

The growing deployment of neural language models demands greater understanding of their internal mechanisms. The goal of this thesis is to make progress on understanding the latent computations within large language models (LLMs) to lay the groundwork for monitoring, controlling, and aligning future powerful AI systems. We explore four areas using open source language models: concept encoding across neurons, universality of learned features and components across model initializations, presence of spatial and temporal representations, and basic dynamical systems modeling.

In Chapter 2, we adapt optimal sparse classification methods to neural network probing, allowing us to study how concepts are represented across multiple neurons. This sparse probing technique reveals both monosemantic neurons (dedicated to single concepts) and polysemantic neurons (representing multiple concepts in superposition) in full-scale LLMs confirming predictions from toy models. In Chapter 3, we identify and exhaustively catalog universal neurons across different model initializations by computing pairwise correlations of neuron activations over large datasets. Our findings show that 1-5% of neurons are universal, often with clear interpretations, and we taxonomize them into distinct neuron families.

To investigate spatial and temporal representations, we analyze LLM activations on carefully curated datasets of real-world entities in Chapter 4. We discover that models learn linear representations of space and time across multiple scales, which are robust to prompting variations and unified across different entity types. We identify individual "space neurons" and "time neurons" that reliably encode spatial and temporal coordinates. In Chapter 5, we use optimal sprase regression techniques to improve the sparse identification of nonlinear dynamics (SINDy) framework, demonstrating improved sample efficiency and support recovery in canonical differential systems. We then leverage this improvement to study the ability of LLMs to in-context learn dynamical systems and find internal representations which track the underlying system state.

Thesis supervisor: Dimitris J. Bertsimas
Title: Boeing Professor of Operations Research

Acknowledgments

There is a long chain of people that I have to thank for getting to me where I am today. First and foremost, I want to express my immense gratitude for my advisor, Dimitris Bertsimas. Dimitris gave me unusual freedom in what I chose to work on, both accommodating my frequent changes in interest in the beginning of my PhD and my laser focus on interpretability after my first year. Dimitris gave me the space and support to let me explore and find something I love. He was always excited to discuss important ideas with me—constantly curious about the latest advances in AI and the industry updates. It was clear he had my best interest at heart, allowing me the pursue my passions, where ever they might take me.

I would also like to thank Neel Nanda and Max Tegmark who were important advisors during the second half of my PhD. Neel took me under his wing in the "early" days of interp (way back in late '22 early '23), giving me lots of important hands on advice. I learned tons from our frequent pairing sessions and check-ins and benefited tremendously from all our subsequent collaborations. Neel also introduced me to Max, who welcomed me into his group and has been a source of great scientific and safety inspiration.

I would like to thank Open Philanthropy for financially supporting the second half of my PhD, which significantly supported my freedom in my research and collaborations.

I would like to thank the other members of my committee, Alex Jacquillat and Rama Ramakrishnan for their time and attention in learning about my work, as well as being accommodating of the short timelines and offering important advice for smoothly finishing my degree.

Before my PhD, the people most responsible for my success were David Shmoys and Daniel Freund. With David, I published my first two papers [106, 116], learned the full research lifecycle, and learned the ways of academia. David was my closest confidant and advisor in my grad school deliberations and helped me land my team at Google. Daniel introduced me to operations research, introduced me to David, and was instrumental in me getting my first two summer jobs: (1) with Samitha Samaranayake–who I greatly appreciate taking a chance on me as a freshman and (2) with David.

Before even undergrad, I have to most thank Matthew Hurray for introducing me to my love of computer science, which started me down this whole path. Also in high school, my pole vaulting couch Jack Scott was an important mentor to me, who taught me a lot about life and the pursuit of excellence.

In addition to my advisors and funders, I am deeply grateful for all of the wonderful collaborators I have had. Within my own papers, I thank Nikhil Garg, Matthew Pauly, Katherine Harvey, Dmitrii Troitskii, Theo Horsley, Carl Guo, Tara Kherikhah, Wendy Sun, and Will Hathaway. For those papers which I have collaborated or advised, I thank Lucia

Quirke, Lovis Heindrich, Josh Engels, Isaac Liao, Eric Michaud, Vedang Lad, Andy Arditi, Oscar Balcells, Daniel Paleka, Ben Wu, and Alessandro Stolfo who did the actual work of putting together some great papers [11, 100, 159, 225, 257].

Beyond research I was apart of, I would like to thank those scientists which inspired me. The most credit here goes to Chris Olah and the rest of the original circuits team, and now the team at Anthropic, which I am ecstatic to join.

I would like to thank those that made up and ran the Operations Research Center: Laura Rose, Andrew Carvalho, Georgia Perakis, and Patrick Jaillet. Laura in particular I greatly appreciate for helping me out with slightly complicated funding and graduation logistics. In addition to the faculty and stuff, I appreciate the hard work put in by my fellow INFORMS officers and student REFS groups to make the ORC and more lively place.

In addition to my department, I befitted from many other programs during my time at grad school. In particular I would like to thank SERI MATS for supporting my initial and subsequent collaborations with Neel, to Constellation for housing me as a visiting researcher for a month, and to MAIA (and HAIST) for concentrating so many talented and like minded individuals together.

Of course, I also am deeply grateful for my friends and family who helped so much along the way. I would like to especially thank my parents for their constant love, support, and encouragement.

Throughout my PhD, I had the joy to hang out with many lovely groups of people: "the Americans" Brian, Angela, and Becca; my fellow informs officers: Becca, Pericles, and Manu; my fellow Dimitris students: Leonard, Yu, Kim, Nick, Pericles, George, Lisa, Matt; other ORC friends: Jacob, Sean, Emily, Yuan, Houssam, Joyce, and Bonnie; the MATS late night crew Stefan, Aryan, and Marius; EA friends Jake, Juan, Robi, and Daniel; and the MAIA grad students Eric, Tony, and Cas and other AIS friends Sam and Josh!

Last but certainly not least, I would like to thank Janice Yang who has been my partner is every sense of the word throughout the second half.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

As large language models (LLMs) become more widely deployed in high-stakes settings, our lack of understanding of how or why models make decisions creates many potential vulnerabilities and risks [25]. While some claim deep learning based systems are fundamentally inscrutable, artificial neural networks seem unusually amenable to empirical science compared to other complex systems: they are fully observable, (mostly) deterministic, created by processes we control, admit complete mathematical descriptions of their form and function, can be run on any input with arbitrary modifications made to their internals, all at low cost and on computational timescales. The goal of this thesis is to make progress on understanding the latent computations within LLMs to lay the groundwork for monitoring, controlling, and aligning future powerful AI systems.

1.1 Outline

1.1.1 Sparse Probing

The first part of this thesis asks basic scientific questions regarding how concepts are computed and represented in the neurons in the feedforward layers of LLMs (accounting for $\frac{2}{3}$ of the parameters). In the simplest case, one might hope there exists a 1:1 correspondence between features of the input and neurons in a network, i.e., each neuron activates for one coherent stimulus. Although the literature contains many examples of such seemingly *monosemantic neurons* [19, 58, 97, 111, 214, 227], an obvious problem arises when a network has to represent more features than it has neurons. To accomplish this, a model must employ some form of compression to embed n features in $d < n$ dimensions. While this *superposition* of features enables more representational power, it also causes loss-increasing "interference" between non-orthogonal features [98]. Recent works in toy models demonstrate that this

tension manifests in a spectrum of representations: dedicated dimensions or neurons for the most prevalent and important features with increasing levels of superposition—and hence decreasing levels of sparsity—for the long tail of rarer or less important features [98, 248].

As with any conceptual insight, these results raise more questions than they answer: To what extent do these observations transfer to full scale language models? What kinds of features do or do not appear in superposition? At what scale? How do we reliably find and verify such (feature, neuron) pairs? In Chapter 2, we introduce and apply sparse probing to systematically study such questions, finding clean examples of both monosemantic neurons and superposition "in the wild."

1.1.2 Universality

In Chapter 3, we study the *universality* of neurons. Specifically, we run experiments to determine to what extent the "same" neurons are learned across a suite of GPT2 model trained on the same data but from a different random initialization. We do so by computing pairwise correlations of neuron activations over 100 million tokens for every neuron pair across five different seeds and find that 1-5% of neurons are universal, that is, pairs of neurons which consistently activate on the same inputs. We then study these universal neurons in detail, finding that they usually have clear interpretations and taxonomize them into a small number of neuron families. We conclude by studying patterns in neuron weights to establish several universal functional roles of neurons in simple circuits: deactivating attention heads, changing the entropy of the next token distribution, and predicting the next token to (not) be within a particular set.

1.1.3 Space and Time

The second part of this thesis attempts to use interpretability tools to study whether LLMs possess necessary ingredients for a world model: (1) static spatial and temporal representations and (2) ability to model basic dynamical systems. The capabilities of LLMs have sparked debate over whether such systems just learn an enormous collection of superficial statistics or a set of more coherent and grounded representations that reflect the real world. In Chapter 4 we find evidence for the latter by analyzing the learned representations of three spatial datasets (world, US, NYC places) and three temporal datasets (historical figures, artworks, news headlines) in the Llama-2 family of models. We discover that LLMs learn *linear* representations of space and time across multiple scales. These representations are robust to prompting variations and unified across different entity types (e.g. cities and landmarks). In addition, we identify individual "space neurons" and "time neurons" that reliably encode

spatial and temporal coordinates. While further investigation is needed, our results suggest modern LLMs learn rich spatiotemporal representations of the real world and possess basic ingredients of a world model.

1.1.4 Dynamics

Finally in Chapter 5, we move from studying static world model primitives to dynamic ones: basic physical systems modeling. We begin by introducing a refinement of the popular sparse identification of nonlinear dynamics (SINDy) framework [52] that leverages mixed-integer optimization to solve the sparse regression subproblem to optimality. After demonstrating that our method improves sample efficiency and support recovery in a number of canonical ordinary and partial differential systems, we explore an application of our technique to language models. In particular, motivated by recent work observing LLMs can in-context [172], we use our system identification method to trace this capability to the internal representations of LLMs.

1.2 Contributions

1.2.1 Findings

As a relatively nascent field, interpretability lacks established paradigms and well-understood quantitative metrics. In such "pre-paradigmatic" disciplines, qualitative results often play a crucial role in guiding research and uncovering fundamental insights. This motivates our radically empirical approach which seeks to (1) gain greater "empirical surface area" on LLMs, that is, gather lots of observations to constrain the hypothesis space for further theorizing and (2) find "signal of structure"[212]—complex patterns that clearly indicate underlying phenomena. Through careful qualitative analysis, we strive to understand the basic abstractions, mechanisms, and higher-level structures underlying neural networks to contribute to the development of a more robust interpretability paradigm.

To this end, this thesis establishes a number of important empirical findings. In Chapter 2, we provide the cleanest evidence (at the time of writing) of computational superposition in full scale LLMs. We then demonstrate the existence of both polysemantic and monosemantic neurons in LLMs, confirming predictions made in toy models [98]. We also show how sparsity of concepts changes with model scale and discuss many challenges and subtleties with applying and interpreting sparse probing experiments.

In Chapter 3, we find most neurons are not universal, in the sense that for otherwise

identical models trained from different random initalizations, there usually does not exist a pair of neurons which consistently activate on the same inputs. However, those that are universal are more likely to be monosemantic. By studying patterns in weights instead of activations, we discover several important functional categories of neurons like prediction neurons and entropy neurons as well as motifs like redundancy, antipodal pairs, and depth specialization.. Many of the insights and motifs studied in Chapter 2 and 3 also generalize to sparse autoencoder features [48, 77], so will continue to be relevant as less attention is paid to individual neurons.

In Chapters 4 and 5, we empirically show that frontier open source LLMs have basic ingredients of a world model. Specifically, in Chapter 4, we find that LLMs have linear, integrated, and multi-scale representations of space and time that are robust to prompting. Finally, in Chapter 5, we show that in addition to static temporal representations, LLMs can also represent and learn dynamical systems in context.

1.2.2 Data

To enable these findings, this thesis required the construction of many new large scale datasets that we have open sourced and are proving useful to others. In Chapter 2, we generated datasets 10 categorical datasets which yielded over 100 binary features to probe for, with such datasets going on to be used by others to evaluate sparse autoencoders [105]. In Chapter 3, we created several hundred token level annotations that have been used as covariates in other studies. Finally, in Chapter 4, we aggregated several hundred thousand diverse real-world entities at many spatiotemporal scales from a variety of sources.

1.2.3 Methods

The main methodological contributions of this thesis are adapting tools from optimal sparse machine learning [29] to novel settings. In Chapter 2, we apply optimal sparse classification methods to neural network probing, to better understand how concepts are represented across multiple neurons. In Chapter 5, we reformulate the sparse system identification problem as a mixed-integer optimization problem and solve the resulting sparse regression problem to optimality resulting in greater sample efficiency and noise robustness.

Chapter 2

Neurons in a Haystack: Case Studies with Sparse Probing

2.1 Introduction

Neural networks are often conceptualized as being flexible "feature extractors" that learn to iteratively develop and refine suitable representations from raw inputs [24, 87]. This begs the question: what features are being represented, and how? Probing is a standard technique used to study if and where a neural network represents a specific feature by training a simple classifier (a probe) on the internal activations of a model to predict a property of the input [20] (e.g., classifying the tense of a verb based on the activations of a specific layer). Because models are parameterized as a series of dense matrix multiplications and elementwise nonlinearities, a natural intuition is that features are represented as linear directions in activation space [98] and are iteratively combined to synthesize increasingly abstract features using linear combinations of previously computed features [214]. To zoom in on these dynamics in modern transformer language models, we propose *sparse probing*, where we constrain the probing classifier to use at most k neurons in its prediction; we probe for over 100 features to precisely localize relevant neurons and elucidate broader principles of how to study and interpret the rich structure within LLMs.

In the simplest case, one might hope there exists a 1:1 correspondence between features of the input and neurons in a network—that for the correct feature definition, a 1-sparse probe would be sufficient. Although the literature contains many examples of such seemingly *monosemantic neurons* [19, 58, 97, 111, 214, 227], an obvious problem arises when a network has to represent more features than it has neurons. To accomplish this, a model must employ some form of compression to embed n features in $d < n$ dimensions. While this *superposition*

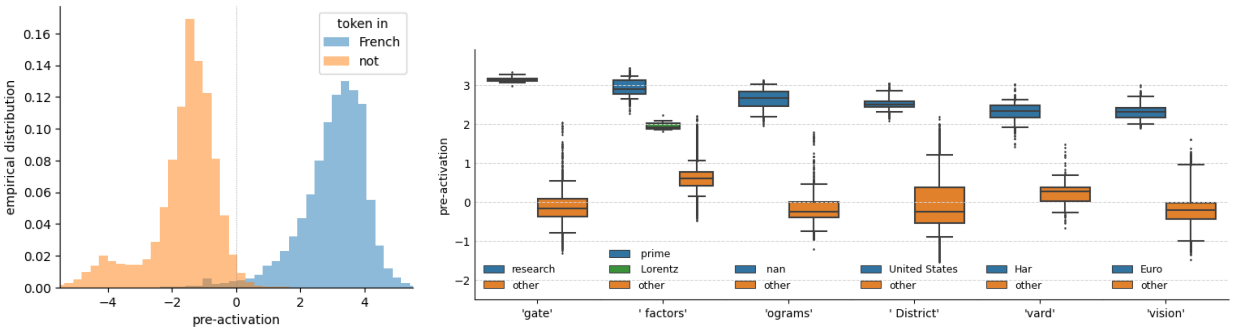
of features enables more representational power, it also causes loss-increasing "interference" between non-orthogonal features [98]. Recent works in toy models demonstrate that this tension manifests in a spectrum of representations: dedicated dimensions or neurons for the most prevalent and important features with increasing levels of superposition—and hence decreasing levels of sparsity—for the long tail of rarer or less important features [98, 248].

As with any conceptual insight, these results raise more questions than they answer: To what extent do these observations transfer to full scale language models? What kinds of features do or do not appear in superposition? At what scale? How do we reliably find and verify such (feature, neuron) pairs? We leverage sparse probing to systematically study such questions, finding clean examples of both monosemantic neurons and superposition "in the wild." Better understanding—and potentially resolving—superposition is on the critical path to ambitious full-model interpretability because a logical consequence of superposition is the presence of *polysemantic* neurons that activate for a large collection of seemingly unrelated stimuli (Figure 2.1) [97, 197]. Such tangled representations undermine our ability to decompose networks into independently meaningful and composable components, thwarting existing approaches to reverse-engineer networks [214].

In addition to being especially well suited to studying superposition, our sparse probing methodology addresses several shortcomings identified in the probing literature [10, 20, 135, 234]. By leveraging recent advances in optimal sparse prediction [34, 36], we are able to prove optimality of the k-sparse feature selection subproblem (for small k), addressing the conflation of ranking quality and classification quality raised in [10]. Second, by employing sparsity as an inductive bias, our probes maintain a strong simplicity prior and are more capable of precise localization of important neurons. This precision enables a more granular level of analysis illustrated throughout our case studies. Finally, the lack of capacity inhibits the probe from memorizing correlation patterns associated with the feature of interest [135], offering a more reliable signal of whether this feature is explicitly represented and used downstream [234].

In the first part of the paper, we outline several variants of sparse probing, discuss the various subtleties of applying sparse probing, and run a large number of probing experiments. In particular, we probe for over 100 unique features comprising 10 different categories in 7 different models spanning 2 orders of magnitude in parameter count (up to 6.9 billion). The majority of the paper then focuses on zooming in [214] on specific examples of general phenomena in a series of more detailed case studies to demonstrate:

- There is a tremendous amount of interpretable structure within the neurons of LLMs, and sparse probing is an effective methodology to locate such neurons (even in superposition), but requires careful use and follow-up analysis to draw rigorous conclusions.



(a) A monosemantic French neuron (b) A polysemantic neuron activating on six unrelated n-grams

Figure 2.1: Neurons that respond to single features (left) can be understood independently, in contrast to polysemantic neurons (right) that activate for many unrelated features (in this case, the occurrence of specific multi-token spans).

- Many early layer neurons are in superposition, where features are represented as sparse linear combinations of polysemantic neurons, each of which activates for a large collection of unrelated n-grams and local patterns. Moreover, weight statistics combined with insights from toy models suggest that the first 25% of fully connected layers employ substantially more superposition than the rest.
- Higher-level contextual and linguistic features (e.g., `is_python_code`) are seemingly encoded by monosemantic neurons, predominantly in middle layers, though conclusive statements about monosemanticity remain methodologically out of reach.
- As models increase in size, representation sparsity increases on average, but different features obey different dynamics: some features with dedicated neurons emerge with scale, others split into finer grained features with scale, and many remain unchanged or appear somewhat randomly.

2.2 Related Work

Probing Originally introduced by [5], probing is a standard technique used to determine whether models represent specific features or concepts [20]. For language models in particular, there exist many probing studies showing the rich linguistic representations learned by models [75, 260, 261], contributing to the broader field of "BERTology" [238]. Particularly relevant to our work are investigations on the role of individual neurons [19, 244, 258, 277] and the identification of sparsely represented features [79, 93]. While probing in general has a number of limitations [88, 95, 135, 184, 234], our work seeks to address, at least partially,

the shortcomings most associated with probing individual neurons [10]. Methodologically, sparse probing is situated within the broader category of probing methods [59, 133, 223, 272], which itself is just one paradigm among other localization techniques [7, 18, 82, 85, 94, 112].

Mechanistic Interpretability Philosophically, our work is motivated by the growing field of mechanistic interpretability (MI) [64, 96, 214]. MI concerns itself with rigorously understanding the learned algorithms (circuits) utilized by neural networks [72, 202, 217, 274] in the hope of maintaining oversight and diagnosing failures of increasingly capable models [204]. Although research on reverse-engineering specific neurons in LLMs is limited, [107] proposed—and later refined [108]—the hypothesis that feed-forward layers function as key-value memories, responding to specific input features (keys) and updating the output vocabulary distribution accordingly [81].

Superposition Perhaps the most significant obstacle to interpreting neurons in LLMs, and consequently, the success of MI as a whole, is the phenomenon of superposition. As first observed by [12], superposition involves compressing multiple features into a smaller number of dimensions [98]. Recent research has investigated when superposition occurs [98, 248], how to design models to have less of it [97, 142], and how to extract features in spite of it [254]. Many of the underlying mathematical intuitions rely on prior work on compressed sensing [89]. Moreover, similar questions have been studied elsewhere in machine learning within the disentanglement literature [24, 70, 151].

Connections to Neuroscience In a promising demonstration of consilience, our previous discussion of superposition has striking analogues with coding theory from biological neuroscience—the study of how neurons in the brain map to sensory stimulus [17, 216, 237]. On one extreme, local coding theory posits the existence of "monosemantic" biological neurons which respond to a very specific stimulus (e.g., pictures of Jennifer Aniston [226]). Superposition is then analogous to sparse coding where a subset of neurons encode some feature about the input [216, 271]. Finally population coding theorizes the responses of a whole brain region are relevant, analogous to the computations of a full layer being required to compute or represent a feature [220, 224].

2.3 Sparse Probing

2.3.1 Preliminaries

Transformers We restrict our scope to transformer-based generative pre-trained (GPT) language models [228] that currently power the most capable AI systems [55]. Given an input sequence of tokens $x = [x_1, \ldots, x_t] \in \mathcal{X}$ from the vocabulary, a model $\mathcal{M} : \mathcal{X} \to \mathcal{Y}$ outputs a probability distribution over the token vocabulary $\mathcal{V}$ to predict the next token in the sequence. $\mathcal{M}$ is parameterized by interleaving L multi-head attention (MHA) layers and multi-layer perceptron (MLP) layers. The central object of the transformer is the residual stream, the sum of the output of all previous layers. Each MHA and MLP layer reads its input from the residual stream and writes its output by adding it to the stream. MHA layers are responsible for moving information between token positions, MLP layers apply pointwise nonlinearities to each token independently and therefore perform the majority of the feature extraction, and the residual stream acts as the communication channel between layers. In this study, we are primarily concerned with the MLP layers (accounting for $\approx \frac{2}{3}$ of total parameters). Ignoring the MHA layer, the value of the residual stream (also referred to as the hidden state) for token t at layer ℓ after applying the MLP is

$$h_t^{(\ell)} = h_t^{(\ell-1)} + W_{\text{proj}}^{(l)} \, \sigma \left(W_{fc}^{(l)} \gamma \left(h_t^{(l-1)} \right) + b_{fc}^{(l)} \right) + b_{\text{proj}}^{(l)}$$

where W_{proj} , W_{fc}, $b_{\text{proj}}^{(l)}$, $b_{\text{fc}}^{(l)}$ are learned weight matrices and biases, γ is a normalizing nonlinearity, and σ is a pointwise rectifying nonlinearity (in our case GeLU). For a more detailed mathematical explanation of the transformer architecture we refer the reader to [96].

Probing is a technique to localize where specific information resides in a model by training a simple classifier (a probe) to predict a labeled feature of the input using the internal activations of the model [20]. More formally, we require a tokenized text dataset $X \in \mathcal{V}^{n \times T}$ (where n is the number of sequences and T is the context length) and an associated labeled dataset $D_{\text{probe}} = \{x_{jt}, z_{jt}\}$ which provides labels for a subset of the tokens (e.g., the tense of every verb). In our setting, we focus on the MLP neuron activations $a^{(\ell)} = \sigma(W_{fc}^{(l)} \gamma(h_t^{(l-1)}))$—the set of representations immediately after the elementwise nonlinearity—and train a linear binary classifier $g_\ell(a_{jt}^{(\ell)}) = \hat{z}_{jt}$ to minimize a classification loss $\mathcal{L}(z_{jt}, \hat{z}_{jt})$ for each layer of the network.

Key Concepts We make frequent reference to the concept of a *feature*. The field has not yet settled on a consensus definition [98], but for our purposes, we mean an interpretable property of the input that would be recognizable to most humans. A *monosemantic* neuron is a neuron which activates for exactly one feature, though this can also be subtle depending on what one is willing to count as one feature versus a composition of related features. In contrast, a *polysemantic* neuron is a neuron which activates for multiple unrelated features. Polysemanticity is a consequence of *superposition*, the phenomenon of representing n features with $d < n$ dimensions. We focus on applying sparse probing to the activations of the MLP layers because these activations form a *privileged basis* [96]. That is, because applying an elementwise nonlinearity breaks a rotational invariance of the representations, it is more likely that the basis dimensions (each neuron) are independently meaningful. Without a privileged basis, as in the residual stream, there is no reason for features to be basis aligned, and therefore no reason to expect sparsity (modulo optimization quirks [99]). Finally, superposition manifests differently in the residual stream than in the neuron activations because of this privileged basis; we discuss this further in A.1.9.

2.3.2 Sparse Feature Selection Methods

Rather than just localize a feature to a particular layer, we attempt to identify a single neuron or a sparse subset of neurons which fire if (and ideally only if) the feature is present in the input. This more granular localization can be accomplished by training a k-sparse probe, a linear classifier with at most k non-zero coefficients. The problem becomes, which of the $\binom{d}{k}$ possible neuron subsets are most predictive? While this is an $\mathcal{NP}$-hard combinatorial optimization problem, there exist a number of fast heuristics and tractable optimal algorithms [264].

In probing, this is typically formalized as a ranking problem [10], where neurons are individually scored by some importance measure and then rank ordered to include the top k. Given a binary classification dataset, natural scoring rules that have been explored are the absolute difference between class means for each neuron [10], the mutual information between each neuron and the labels [223, 240, 272], or the absolute magnitude of the coefficients of a dense probe trained with l_1 regularization [79].

We propose two additional techniques: adaptive thresholding and optimal sparse probing (OSP). OSP leverages recent advantages in sparse prediction solution techniques to train a k-sparse classifier to provable optimality using a tractable cutting plane algorithm [36], though this is only feasible for smaller values of k. For larger ranges of k, we use adaptive thresholding to train a series of classifiers that iteratively decrease the value of k, where

at each step t we retrain a probe to only use the top k_t neurons with highest coefficient magnitude from the previous k_{t-1}-sparse probe. For the sake of computational efficiency, in both methods we perform an initial filtering step to take the top neurons with maximum mean difference.

For all methods, we retrain a logistic regression probe for the k neurons selected. Additionally, while we focus on classification, all of these methods have a straightforward generalization to continuous targets.

2.3.3 Probing in Practice

Constructing Probe Datasets An informative probing experiment begins with designing a suitable probing dataset. To illustrate a common issue, take the case of probing for the `is_politician` feature. If the dataset just includes the names of people labeled as politicians or not, the trained probe will not be able to distinguish the model's `is_politician` feature from the `is_political` feature which fires for all political content (e.g., "The Democratic Party"). On the other extreme, if the dataset contains just the names of politicians and random tokens, the probe won't distinguish `is_politician` from `is_person` as very few random tokens concern non-politician people. In other words, there is a general tension in how to shape the negative examples in the dataset to create the most *conceptual separation* between the true feature of interest and all possible correlates.

Another issue arises when the feature of interest is a property of multiple tokens or a full sequence of tokens (e.g., `is_python_comment` or `is_french`). If the relevant neuron fires on every token in the feature sequence, one could just sample tokens randomly, but there might be more specific conditions on when a neuron fires that cannot be known *a priori*[1]. A solution to this is to perform an elementwise aggregation (e.g., mean or max) of the activations over the token span of each occurrence of the feature. For example, for a dataset of sequences in many different languages, the input to our probe could be the average activation vector for each sequence, with the target being the language of the sequence. Unfortunately, this gives a somewhat weaker result, as this process is not able to distinguish the `is_french_noun` feature from the `is_french_verb` feature and thus requires further analysis to interpret correctly.

There are additional subtleties having to do with feature granularity, rare features, and overlapping features [234] which we explore in Section 2.5.4. For all of these issues, the most appropriate recourse will depend on the researcher's intent and the nature of the feature being probed for.

[1] As an example, in the case of factual recall, the results from [188] suggest that MLP neurons are especially important for the last token of a multi-token subject.

Evaluation and Interpretation To evaluate the performance of a probe, we compute the number of true positives (TP), false positives (FP), false negatives (FN), and true negatives (TN) of our binary classifier on an out-of-sample test set. We then calculate the precision (PR), recall (RE), and F1 score (F1):

$$\text{PR} = \frac{\text{TP}}{\text{TP} + \text{FP}} \qquad \text{RE} = \frac{\text{TP}}{\text{TP} + \text{FN}} \qquad \text{F1} = \frac{2\text{PR} \times \text{RE}}{\text{PR} + \text{RE}}$$

We use the F1 score as our primary evaluation metrics to determine which neurons are most likely associated with the target feature given the asymmetric importance of the positive class[2]. Precision and recall give insight into how the selected neurons' implicit feature granularity compares to the feature being probed for. Low precision and high recall indicates either selected neurons are highly polysemantic or the model represents a more general feature than is being probed for. High precision and low recall of the probing classifier may indicate that the identified submodule represents a more specific feature than the feature being probed for (e.g. `is_french_noun` instead of `is_french`). We can adjust our preference for classifiers with high precision or high recall by modifying the threshold of the classifier or by tuning the class weights in the probe loss function; a higher class weight for the positive class (assuming it has fewer samples than the negative class) will prioritize recall over precision.

2.4 Empirical Overview

Models We study EleutherAI's Pythia suite of autoregressive transformer language models [37] [3]. These models are fairly standard GPT variants that use parallel attention and rotary positional encodings and were trained on The Pile [104]. In particular, we run experiments on the 7 models ranging from 70M to 6.9B parameters; full model hyperparameters are given in Table A.1.

Data We study ten different feature collections: the natural language of Europarl documents, the programming language of Github source files, the data source of documents from The Pile, the part-of-speech and grammatical dependency of individual tokens, morphological features of tokens (e.g., verb tense), plain text features of tokens (e.g., whitespace or capitalization), the presence of specific compound words, LaTeX features in ArXiv docu-

[2] We also measure Matthew's Correlation Coefficient (MCC), a balanced accuracy metric, and the results are largely the same.

[3] https://github.com/EleutherAI/pythia; unfortunately, our experiments were performed with the `V0` suite of models which have recently been updated.

ments, and a number of factual features associated with people (e.g., gender, occupation). Full descriptions of datasets, their construction process, and summary statistics are available in A.2.2. All categorical features are converted into separate one-versus-all binary features for over 100 total binary classification tasks.

Experiments For each combination of model, feature, layer, and sparse feature selection methods, we train probes for a range of values for k and report classification performance on a held-out test set. We compare the feature selection methods in A.2.5 and illustrate the relationship between model size and sparsity in Figure A.2. For neurons identified as being especially relevant, we save the activations over larger text data sets and perform further analyses described throughout Section 2.5. All code and data are available at https://github.com/wesg52/sparse-probing-paper.

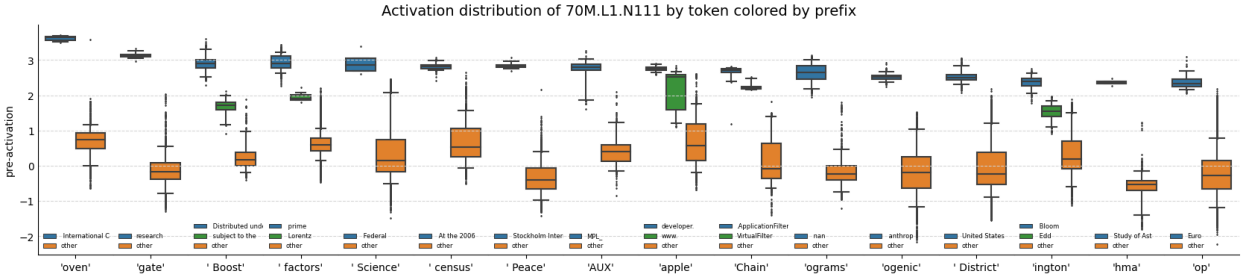
2.5 Case Studies

We conduct a series of more detailed case studies to carefully study the behavior of individual neurons, while also illustrating the challenges that pose barriers to further progress. Although we zoom in on very narrow examples, we found many neurons of the same category in our probing experiments and believe these examples are representative of broader neuron families [214] that exist in all LLMs.

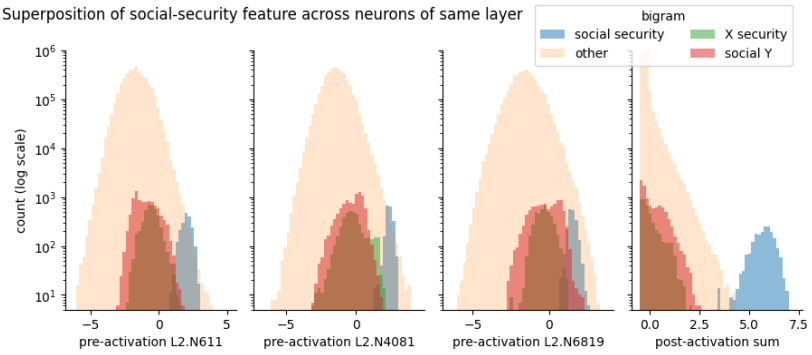
2.5.1 Superposition in the Wild: Compound Word Neurons

The token vocabulary is a fairly unnatural symbolic language to perform all of the linguistic and conceptual processing required of a language model. For instance, compound words like "social security" are treated as two separate tokens, despite meaning something very different when the tokens appear together versus apart. Moreover, quirks of spacing and capitalization frequently cause words to be broken up (e.g., despite " Harvard" being a single token, when not preceded by a space "Harvard" gets tokenized into "Har" and "vard"). Based on many such examples, [97] hypothesize that a primary function of the early layer neurons is to "de-tokenize" the raw tokens into a sequence of more useful abstractions. However, this *pseudo-vocabulary* might be extremely large (e.g., all common n-grams) making it a natural candidate for being represented in superposition. To investigate this, we probe for neurons which respond to 21 different compound words, where the first and second words mean something quite different when appearing separately versus together (e.g., "prime factors").

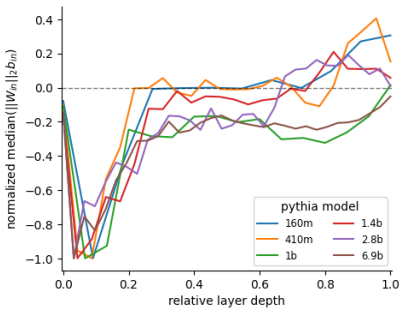
After probing for neurons which activate for specific compound words `XY` while not firing



(a) Activations of a single polysemantic neuron on different tokens when preceded by specific stimuli.



(b) Example of superposition in representing the compound word "social security."



(c) Evidence of superposition in early layers

Figure 2.2: Summary of our key results of superposition. Polysemantic neurons firing on many diverse stimuli (a) are necessary to implement superposition (b) to adequately represent many more possible n-grams than dimensions. The natural mechanism for implementing superposition in toy models leverages large weight norms and negative biases[98] (see Section A.3 for additional motivation and discussion), which we observe to be most associated with early layers (c).

on any bigrams of the form XZ or WY $\forall$ W,Z$\in \mathcal{V}$, Z$\neq$Y, W$\neq$X for each compound word we found *many* individual neurons which were almost perfectly discriminating. However, after inspecting the activations across a much wider text corpus, we observe these neurons activate for a huge variety of unrelated n-grams (see Figure 2.2a), a classic example of the well known phenomena of polysemanticity [97, 197, 214]. Superposition implies polysemanticity by the pigeonhole principle—if a layer of n neurons reacts to a set of $m \gg n$ features, then neurons must (on average) respond to more than one feature. This example also underscores the dangers of "interpretability illusions" caused by interpreting neurons using just the maximum activating dataset examples [44]. A researcher who just looked at the top 20 activating examples would be blind to all of the additional complexity of neuron 70M.L1.N111, with Figure 2.2a still only scratching the surface[4]. While inconvenient for interpretability researchers, polysemanticity is also problematic for the model, as it causes interference between different features [98]. That is, if 70M.L1.N111 fires, the model gets mixed signals that both the "prime **factors**" feature and the "International C**oven**" feature are present.

However, detecting n-grams in particular (and hence constructing the hypothesized pseudo-vocabulary), turns out to be a task particularly well suited for superposition (see A.1.6). In short, the model can take advantage of the fact that (for fixed n) exactly one possible n-gram out of $|\mathcal{V}|^n$ can occur at any time—that is, n-grams are mutually-exclusive binary features with respect to the current token of the input, despite coming from a massive set of possible features. While it would be impossible to dedicate a unique neuron to all possible n-grams, by leveraging the activations of multiple polysemantic neurons, each of which react to a bigram XY but with no other overlapping stimuli, the magnitude of the "true" feature gets boosted above all of the possible interfering features. As an example, in Figure 2.2b, we show three neurons from the same layer which activate for the "social security" bigram while (mostly) not activating for bigrams with just one of the words (see the blue histograms being significantly to the right of the red and green histograms). However, because these neurons are polysemantic, there are many other inputs which cause them to activate, potentially even more so than the "social security" feature (see the orange histogram having a longer tail than the blue histograms). Despite this polysemanticity, by summing the activations across the three neurons, we achieve nearly perfect separation between the total activation magnitude of the "social security" feature and *all* other observed token combinations! We believe this to be first example of neuron superposition exhibited "in the wild" in real LLMs. We include

[4]This figure was generated with 300 million tokens from the Pile Test set but when run on 10 billion tokens of the Pile train set 16 of the top 20 dataset examples show the neuron activating on the "he" of German words starting with "Schönhe". These maximum activating dataset examples can be viewed on Neuroscope[198]

examples for all 21 compounds words from layers 1-3 of Pythia 1B in Figures A.4, A.5, and A.6, and further results on basis alignment in A.1.10 and Figures A.8 and A.9.

We believe that this computational motif—merging individual tokens into more semantically meaningful n-grams in the pseudo-vocabulary via a linear combination of massively polysemantic neurons—is one of the primary function of early layers. As a supporting line of evidence, we observe that a natural idealized model of n-gram recovery in superposition predicts a mechanistic fingerprint of superposition: the presence of large input weight norms and large negative input biases. We provide an idealised construction in Section A.3 for how an arbitrary number of features can be compressed into two linear dimensions, similar to the construction in [98]. When we measure the product of the input weight norm and bias for each neuron in every layer and every model, we observe a striking difference in the early[5] layers, exactly in line with our conceptual argument (see Figure 2.2c for a summary and Figure A.3 for the full distributions, and Section A.3 for additional motivation of the metric and discussion).

We conclude by noting that, in another encouraging display of consilience, the phenomenon of different regions of the model employing different coding strategies for different functions bears resemblance to the diverse coding strategies employed by biological neural networks in different brain regions with different functional roles [137, 143, 162, 215]. We expect further research connecting local mechanisms to macroscopic structure to be particularly fruitful.

2.5.2 Context Neurons: a Monosemantic Neuron Family

Given the potential benefits of superposition, it is reasonable to ask if *any* features are represented monosemantically, and if so, why? We hypothesized that a likely candidate would be *context features*—high-level descriptions of all (or most) tokens within a sequence (e.g. `is_french` or `is_python_code`). Such features seem quite important, to the point that avoiding interference is worth a full neuron, while also being a higher-level property that may or may not be mutually exclusive or binary, making it harder to represent in superposition.

In particular, we probe on the language of natural language sequences from the Europarl dataset [153], the programming language of different Github source files, and the data subset of The Pile from which a sequence originated. We probe for these features using mean aggregation—training a classifier on the averaged activations over each sequence to predict the sequence label (e.g., `is_french` or `is_python`). Figure 2.3 depicts a sample of our results, illustrating the existence of highly specialized context neurons that activate approximately

[5] Pythia models use parallel attention so layer 0 is purely a function of the current token, hence not subject to the n-gram analysis.

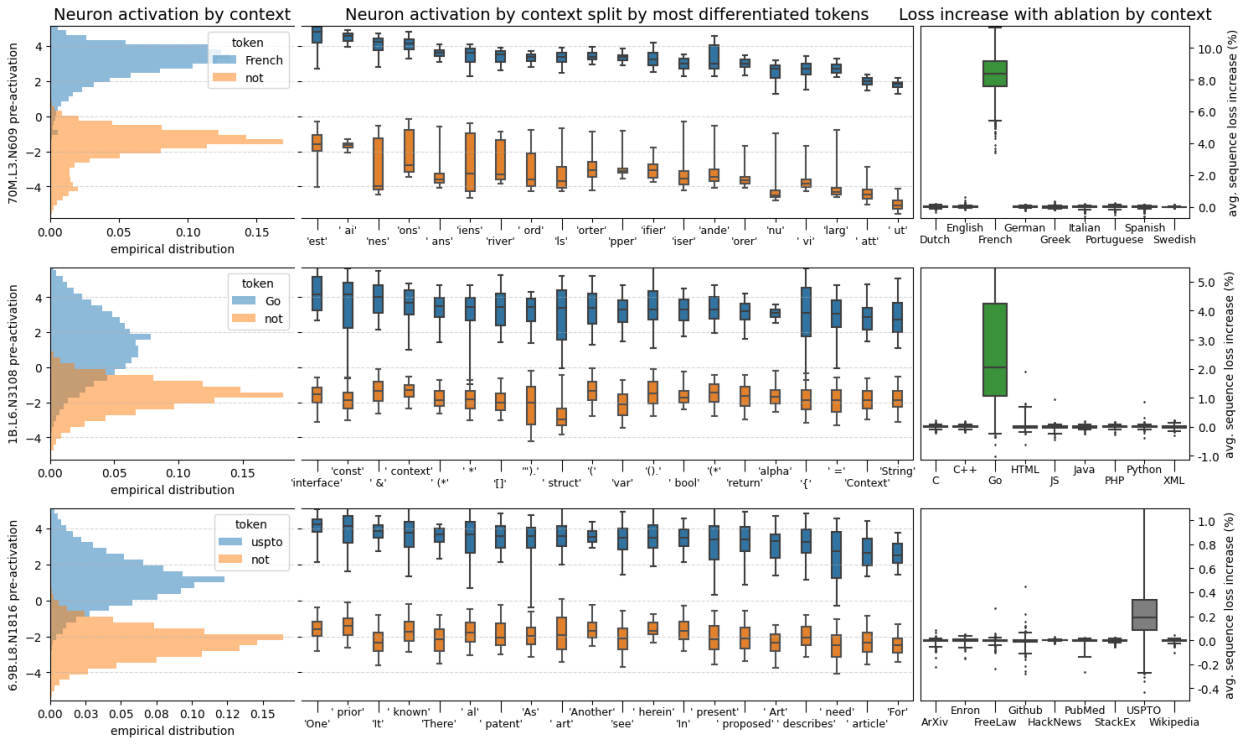
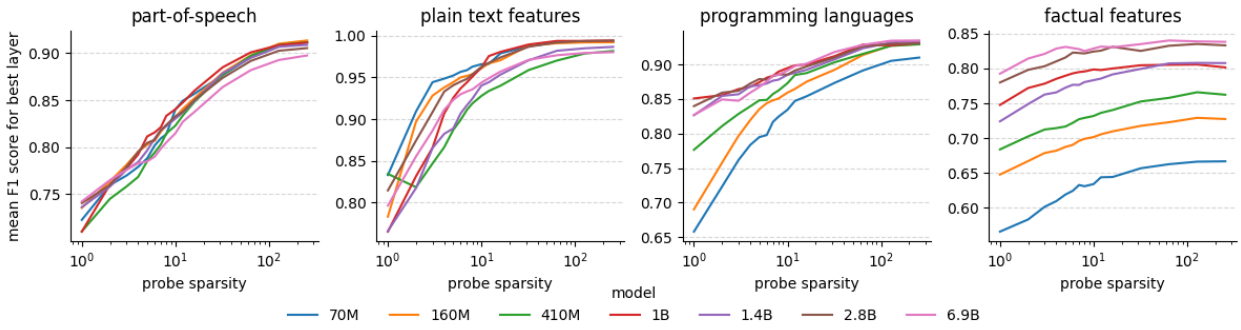


Figure 2.3: Analysis of individual context neurons that activate on tokens in French (top), in Go code (middle), and US Patent office documents (bottom). We show the distribution of activation when in and not in the target context (left) in addition to the tokens with average activation most distinguished by the neuron (middle). Finally we report the increase in sequence loss across contexts when the neuron is ablated (right).

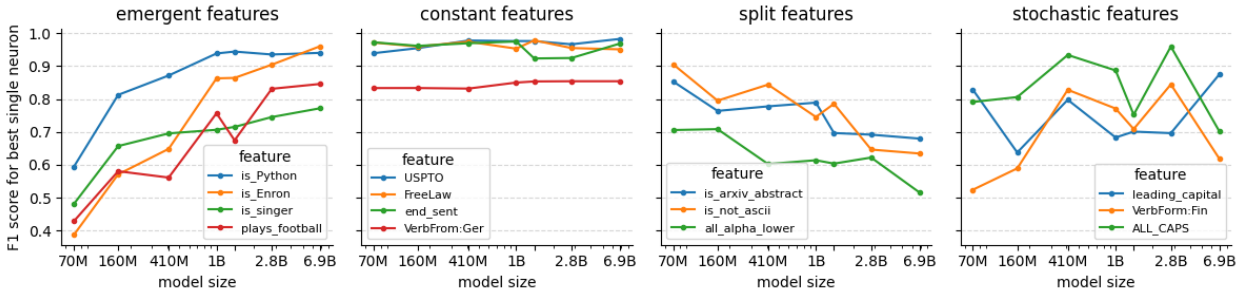
only when a token is in a specific context[6] (with 21 additional examples in Figures A.10, A.11, and A.12). To better understand the function of these neurons, for each token, we study the distribution of activations broken down by context (e.g., the activation of "return" in every programming language). Sorting by the largest differences (middle panel), we observe that one explanation for these neurons' roles is token disambiguation. For example, many programming languages have a `return` keyword, but neuron `1B.L6.N3108` activates on `return` if and only if it is in the context of Go code.

To further support our interpretation and gain more insight into function, we conduct ablation experiments comparing the language modeling loss of the base model to the loss of the model with the identified neuron fixed to 0 for all tokens in every sequence (right panel). As anticipated, ablating the context neurons significantly degrades language modeling performance within the relevant context, while leaving other contexts nearly unaffected. The impact, however, heavily depends on the model size—in the 70M parameter model ($\approx$ 12k neurons), ablating a single neuron causes an average loss increase of 8% per French sequence, while in the 6.9B model ($\approx$524k neurons), ablating one neuron results in only a 0.2% increase in loss.

While these neurons appear to be genuinely monosemantic, we emphasize that it is extremely difficult to prove this. Doing so requires answering thorny ontological questions (what *is* French?) and then efficiently searching through a dataset of billions of tokens to verify the neuron implements the answer. Moreover, the "true" feature the neuron responds to might be quite subtle. For instance, many of the code neurons do not fire when in a code comment, whereas the French neuron will fire in a non-French context on a French name or other token strongly associated to France[7]. In some sense, these exceptions prove the rule of the primary role of the neuron, but highlight the difficulty of mapping human features to the ontology of the network. Perhaps most challenging though is ruling out the existence of very rare features the neuron also responds to. Sufficiently rare features will likely be undetectable from random samples or summary statistics—even when broken down by token—and therefore require manually[8] examining max activating dataset examples in subdistributions where we don't expect to find much French text.



(a) Probe sparsity versus classification performance for different feature types and model sizes.



(b) Classification performance of most feature-aligned neuron by model size for different feature classes.

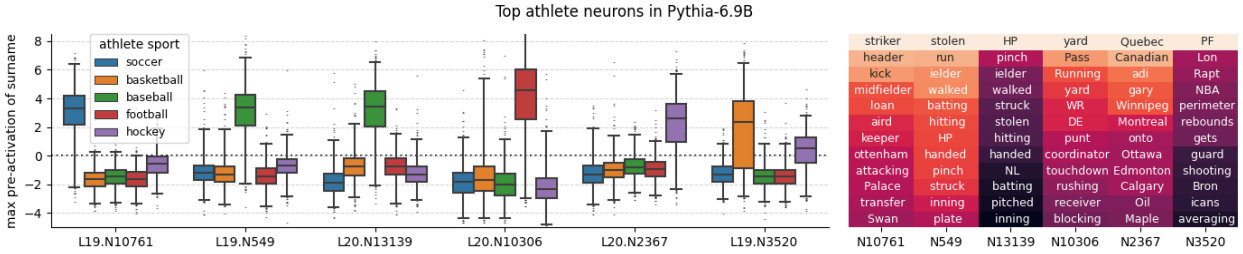
Figure 2.4: Summary of results on model scale. Representational sparsity increases on average with scale (top) but individual features obey different scaling dynamics (bottom).

2.5.3 Effects of Scale: Quantization and Splitting

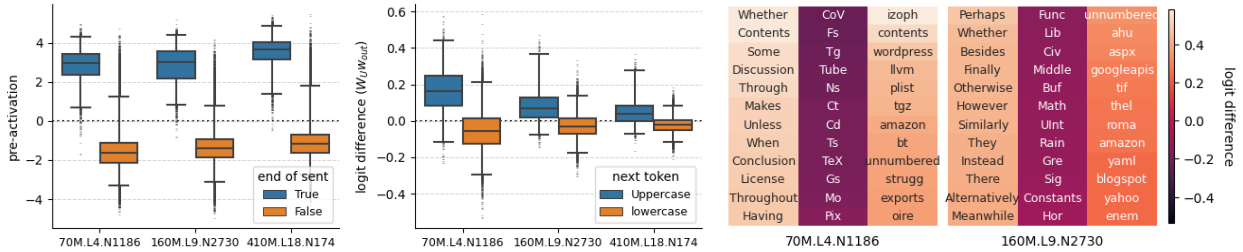
Given the importance of scale in LLMs, we turn our attention to studying the relationship between model size and the sparsity of representations, and the dynamics that drive this relationship. For all of our feature datasets described in A.2.2, we train a series of probes sweeping the value of k from 256 to 1 using adaptive thresholding. For sake of summarization, we report the maximum out-of-sample F1 value over the layers for each model, while averaging over the features within the collection (see Figure 2.4a for a sample and Figure A.2 for full results with random baselines).

While some features appear to be more sparsely represented with scale (e.g., programming languages and factual features), we were surprised by how consistent others were (e.g., part-of-speech and compound words). In fact, for some of the simplest plain-text features, the smallest models seem to actually implement greater sparsity. When analyzing individual features on a per neuron basis, several general patterns become more apparent (Figure 2.4b). We believe there are two main dynamics driving these results: the quantization model of scaling [193] and neuron splitting [97].

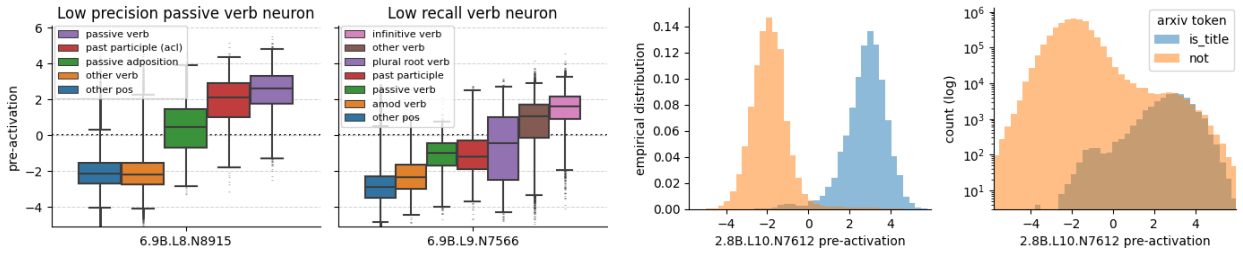
In particular, the quantization hypothesis posits that there exists a natural ordering of features learned by a model with increasing scale based on how loss reducing they are, with larger models able to learn a longer tail of increasingly rare features [193]. In our context, this suggests that features like part-of-speech or compound words are within the feature set learned by even small models, and are represented with very similar sparsity patterns. In contrast, factual features and some (but not all) contextual features only get represented by single neurons at sufficient scales. As a countervailing force, with increasing scale the model can dedicate multiple neurons to represent more granular features that previously comprised one coarser feature. Consider the `ALL_CAPS` feature; while it would likely be advantageous to have dedicated circuitry for representing this feature in all models, a larger model might have dedicated neurons for all of the particular reasons a token might be in all capital letters (e.g., an abbreviation, a constant in python, shouting on the internet, etc.), eliminating the need to have just one more coarse grained representation. The result, as viewed from a probing experiment, is less sparsity, not more.



(a) Neurons which activate for the names of specific types of athletes (left) turn out be more general sport neurons when analyzing the top per-token average activations (right).



(b) End of sentence neurons: analyzing the effect on the output vocabulary can corroborate and refine neuron interpretations. For token heatmaps (right), the columns correspond to capital tokens with max increased logits, capital tokens with max decrease, and lowercase tokens with max increase, respectively.



(c) Precision and recall metrics can guide further analysis.

(d) Distributional statistics can hide rare features.

Figure 2.5: Examples of analyses used to refine and support neuron interpretations.

2.5.4 Refining and Verifying Interpretations

The result of a sparse probing experiment is a set of k neurons with a collection of classification performance metrics, usually for a range of values of k. We illustrate what conclusions might follow, simple techniques to refine and corroborate such conclusions, and confounding factors to be wary of.

One confounding factor is an ambiguity between superposition and *composition* [197]. If a (layer, feature) pair has a 1-sparse probe with poor accuracy and a k-sparse probe with high accuracy, it is temping to take this as evidence of superposition, but it is also consistent with a feature being coarse or compositional—being either the union or intersection of multiple independent features. As an example, consider an `is_athlete` feature. Such a feature could either be represented by a single neuron, a superposition of polysemantic neurons, or as the union of neurons for specific types of athletes, as those shown in Figure 2.5a. While we can distinguish a feature union from superposition by analyzing the activation co-occurrence (only one athlete type neuron activates for each athlete), this reasoning does not work for a true compositional feature. For example, we also found individual neurons which coded for a person's gender and whether they are alive or not (Figure A.15). It is likely more natural for a model to represent the `is_living_female_soccer_player` feature as simply a composition of three different neuron aligned-features `is_living`, `is_female`, and `plays_soccer`, which would nevertheless have low 1-sparse accuracy but high 3-sparse accuracy. Of course, this also assumes that these neurons are all in the same layer. However, models can, and almost certainly do, leverage superposition and composition across multiple layers, a subtlety which we blissfully ignore and leave for future work.

For the remaining discussion, we consider individual neurons identified by 1-sparse probes—how do we interpret such neurons? The most basic analysis is to simply inspect the maximum activating dataset examples [198], though this is subject to interpretability illusions [44]. A slightly more robust analysis involves computing the average activation for each token in the vocabulary. Doing so for our athlete neurons in Figure 2.5a reveals these neurons are perhaps better thought of as generalized sport neurons which activate for all tokens generally having to do with a particular sport, including the names of athletes (with the notable exception of the hockey neuron which appears to be a Canadian neuron).

In addition to analyzing a neuron via the input, one can also gain insight by analyzing

[6] By recording the activation of every token, we eliminate the concern of finding an `is_french_noun` neuron instead of an `is_french` neuron.

[7] To further explore this, we encourage the reader to explore max activating dataset examples in Neuroscope[198]

[8] This is an area where we expect AI assisted interpretability to be particularly useful.

the output; in particular, one can compute a neuron's effect on the output logits by simply considering the product of the unembedding matrix and the neuron output weight [81]. Doing so for neurons which always activate on punctuation ending a sentence, we find that the output probability of most uppercase tokens gets increased while the probability of most lowercase tokens gets decreased (Figure 2.5b). Inspecting specific examples is often constructive; for instance, the highest increasing lowercase tokens are words like "amazon" or "wordpress," indicating such neurons are also useful in constructing URLs. For uppercase tokens, those with the highest increase in probability correspond to words like "Finally" or "However" while the most decreased tokens are those representing atomic elements, proper nouns, or CamelCase strings.

Most of the neurons discussed thus far have had excellent overall classification performance; what happens if there does not exist such a crisp fit? As discussed in Section 2.3.3, classification performance can be further decomposed into precision (sensitive to false positives) and recall (sensitive false negatives). In our context, high recall and low precision with respect to a specific feature potentially suggests that a neuron represents a less granular feature than the feature manifest in the probe dataset [9]; low recall and high precision suggest a neuron represents a more specific feature. As an example, Figure 2.5c depicts the activations for a high-recall-low-precision neuron identified when probing for `is_passive_verb` and a low-recall-high-precision neuron identified when probing for `is_verb` in Pythia 6.9B. When we analyzed these neurons in more detail, we find that in addition to activating on all occurrences of passive verbs, `L8.N8915` also activates on adjacent adposition tokens, and past participles when in an adnominal clause. Almost symmetrically, `L9.N7566` activates on most verbs, especially infinitive verbs, but not passive verbs, adjectival modifiers, plural root verbs, or past participles in certain dependency roles. These examples illustrate the more general point that it should not be assumed that a model will learn to represent features in an ontology convenient or familiar to humans.

Of course, precision and recall are only defined with respect to the labels of a probing dataset, which may themselves contain imbalances or spurious correlations that confound the results. Perhaps most common, especially when constructing a dataset from scratch, is the problem of asymmetric sampling and rare features. Consider a probing dataset for an `is_arxiv_paper_title` feature (Figure 2.5d). Without a more specific hypothesis on the relevant negative examples, one is likely to simply sample random non-title tokens (or at least weight all non-title examples the same). However, when the space of negatives is large (e.g., all non-title tokens), at a distribution level, all rare features get drowned out.

[9] It is also consistent with superposition. Indeed, all of the compound word neurons had higher recall than precision.

Hence, when looking at summary or distributional statistics, the results can look impressive (Figure 2.5d; the neuron had F1 $>$ 0.95 for `is_title`), but when looking at raw counts (Figure 2.5d right) one observes that the `is_title` feature actually explains less than half of the activations. Though, upon inspection, we find that the rest of the activations are for *section* titles!

2.6 Discussion

2.6.1 Strengths and Weaknesses of Sparse Probing

The primary use case of sparse probing is to quickly and precisely localize neurons relevant to a specific feature or concept, while naturally accounting for superposition and composition. The speed and precision of recovery is in contrast to gradient-based [82] and causal-intervention based methods [188], which are too slow or coarse-grained to be used for features requiring precise localization within large models. The feature specificity contrasts with sparse autoencoding based methods which aim to recover *all* features stored in the model, but in an unsupervised manner and without attributing semantic meaning to each feature [254]. Having probes with optimality guarantees further addresses the pitfall raised by [10] regarding the conflation of classification quality and ranking quality when analyzing individual neurons with probes. Moreover, sparse probes are designed to be of minimum capacity, mitigating the concern that the probe is powerful enough to construct its own representation to learn the task [135]. While probing requires a supervised dataset, once constructed, it can be reused for interpreting any model (modulo features regarding tokenization), enabling investigation into the universality of learned circuits [72, 214] and the natural abstractions hypothesis [68]. By design, sparse probing is particularly well suited for studying superposition, and can be used to automatically test the effect of architectural changes on the frequency of polysemanticty and superposition, as opposed to relying on human evaluations as in [97].

However, sparse probing also inherits many of the weaknesses of the general probing paradigm [88, 95, 135, 184, 234]. In short, the results from a probing experiment do not allow for drawing especially strong conclusions without more detailed secondary analysis on the identified neurons. Probing offers limited insight into causation, and is highly sensitive to implementation details, anomalies, misspecifications, and spurious correlations within the probing dataset. For interpretability specifically, sparse probes cannot detect features built up over multiple layers or easily distinguish between features in superposition versus features represented as the union of multiple independent and more granular features [197].

In seeking the sparsest classifier, sparse probing may also fail to select important neurons that are redundant within the probing dataset, requiring the use of iterative pruning to enumerate all important neurons. Multi-token features require special processing, often in the form of aggregations that can further weaken the specificity of the result. Finally, depending on how much representations change with scale [193], larger models may utilize more specific features [97], hampering the transferability of datasets between different model scales.

2.6.2 Strengths and Weaknesses of Empirical Findings

In light of the limitations of probing, we attempted to corroborate every case study with independent evidence, including theoretical predictions, ablations, and vocabulary analyses. We find our evidence compelling, although not always conclusive. In particular, we believe we have provided the clearest evidence to date of superposition, monosemantic neurons, and polysemantic neurons in full scale language models. Furthermore, by demonstrating this behavior in seven different models spanning two orders of magnitude in size while exploring over a hundred features, we believe our basic insights are likely to be general and to transfer to current frontier models like GPT-4.

However, much of our analysis is *ad hoc*, tailored to the specific feature being investigated, and requires substantial researcher effort to draw conclusions. While we explored models of varying size, they were all from the same model family and trained with the same data. We think it is unlikely our results are specific to the implementation details of the Pythia model suite, but we do not rule this out. Additionally, the largest model we studied is 6.9 billion parameters which is still more than order-of-magnitude off the frontier. Given the emergent abilities of LLMs with scale [279], it is possible our analysis misses a key dynamic underlying the success of the largest models. Moreover, our results are restricted to binary features and categorical features converted to binary features, and we are not confident that our insights will cleanly transfer to continuous features or cleverly encoded categorical features.

2.6.3 Implications

For interpretability researchers, our results support the conclusion that superposition is important for the success of models. Hence, attempts to remove it [97] are likely either hiding it or are unlikely to be competitive, but that the highest leverage interventions are perhaps best aimed at the early layers. For AI ethicists and legal scholars, our study of factual neurons point to an important avenue of further inquiry—understanding how neurons encoding protected attributes compute this feature and affect downstream predictions of the model. For AI alignment scholars, our work highlights the potential of identifying safety-

critical features and perhaps even manually intervening in computations to enhance our ability to steer models. While none of the features we probed for were especially critical, the Pile subset context neurons could be considered a minimal precursor for situational awareness [63, 204], as these could be used to detect whether an input is from the training or test distribution.

2.6.4 Future Directions

We only scratched the surface of possible applications and experiments involving sparse probing. Scientifically, further understanding superposition—and how to cope with it—seems central to making progress on the ambitious version of mechanistic interpretability. As outlined in [98], this requires either designing architectures which do not use superposition or to take features out of superposition using a sparse coding like technique—both of which can be assisted with automatic sparse probing as a form of validation. While not explored here, it would also be possible to apply sparse probing to predicting properties of the *output* (e.g. `next_token_is_verb`), as opposed to properties of the input, to better understand the neurons most implicated in making specific types of predictions. With neurons identified in each experiment, it would be potentially insightful to also track how these neurons develop and change through time [171] as well as more carefully analyze how the set of neurons change with scale [193], which is naturally enabled by the Pythia suite's [37] checkpoints. In particular, sparse probing (or similar) is well suited to study neuron splitting in more detail by training probes to predict the value of a less granular neuron from the sum of a sparse set of more granular neurons.

Our primary motivation however, is in ensuring the development of safe AI systems. To further this goal, we envision the development of a large library of probing datasets—potentially with AI assistance—that capture features of particular relevance to bias, fairness, safety, and high-stakes decision making. In addition to automating evaluations of new models, having large and diverse supervised datasets will enable better evaluations of the next generation of unsupervised interpretability techniques [56, 254] that will be needed to keep pace with AI progress.

2.7 Conclusion

Guided by sparse probing, we have found some of the cleanest examples of monosemanticty, polysemanticity, and superposition in language models "in the wild," and contributed practical guidance and conceptual clarification for how to interpret neurons in greater detail.

More than any specific technical contribution, we hope to contribute to the general sense that ambitious interpretability *is* possible—that LLMs have a tremendous amount of rich structure that can and should be understood by humans. We believe this is most productively accomplished with an empirical approach more reminiscent of the natural sciences, such as biology or neuroscience, than the traditional experimental loop of ML. While such research can be hard to finish, it is easy to start, so we encourage the curious researcher to just start looking!

Acknowledgements

This chapter is based on joint work [118] with Neel Nanda, Matthew Pauly, Katherine Harvey, Dmitrii Troitski, and Dimitris Bertsimas.

Our research benefited from discussions, feedback, and support from many people, including Marius Hobbhahn, Stefan Heimersheim, Jett Janiak, Eric Purdy, Aryan Bhatt, Eric Michaud, Janice Yang, Laker Newhouse, and Yonatan Belinkov. We particularly thank Trenton Bricken, Adam Jermyn, and Chris Olah for extremely generous and detailed feedback, and Chris Olah for help clarifying concepts in the FAQ. We would also like to thank the SERI MATS program for facilitating the collaborations that started the project. Our work would also not have been possible without the excellent TransformerLens library [199] and the computing resources provided by the MIT supercloud [236].

Chapter 3

Universal Neurons in GPT2 Language Models

3.1 Introduction

As large language models (LLMs) become more widely deployed in high-stakes settings, our lack of understanding of why or how models make decisions creates many potential vulnerabilities and risks [25, 45, 131]. While some claim deep learning based systems are fundamentally inscrutable, artificial neural networks seem unusually amenable to empirical science compared to other complex systems: they are fully observable, (mostly) deterministic, created by processes we control, admit complete mathematical descriptions of their form and function, can be run on any input with arbitrary modifications made to their internals, all at low cost and on computational timescales [209]. An advanced science of interpretability enables a more informed discussion of the risks posed by advanced AI systems and lays firmer ground to engineer systems less likely to cause harm [23, 62, 90, 204, 280].

Olah et al. [214] propose three speculative claims regarding the interpretation of artificial neural networks: that features—directions in activation space representing properites of the input—are the fundamental unit of analysis, that features are connected into circuits via network weights, and that features and circuits are universal across models. That is, analogous features and circuits form in a diverse array of models and that different training trajectories converge on similar solutions [165]. Taken seriously, these hypotheses suggest a strategy for discovering important features and circuits: look for that which is universal. This line of reasoning motivates our work, where we leverage different notions of universality to identify and study individual neurons that represent features or underlie circuits.

Beyond discovery, the degree to which neural mechanisms are universal is a basic open

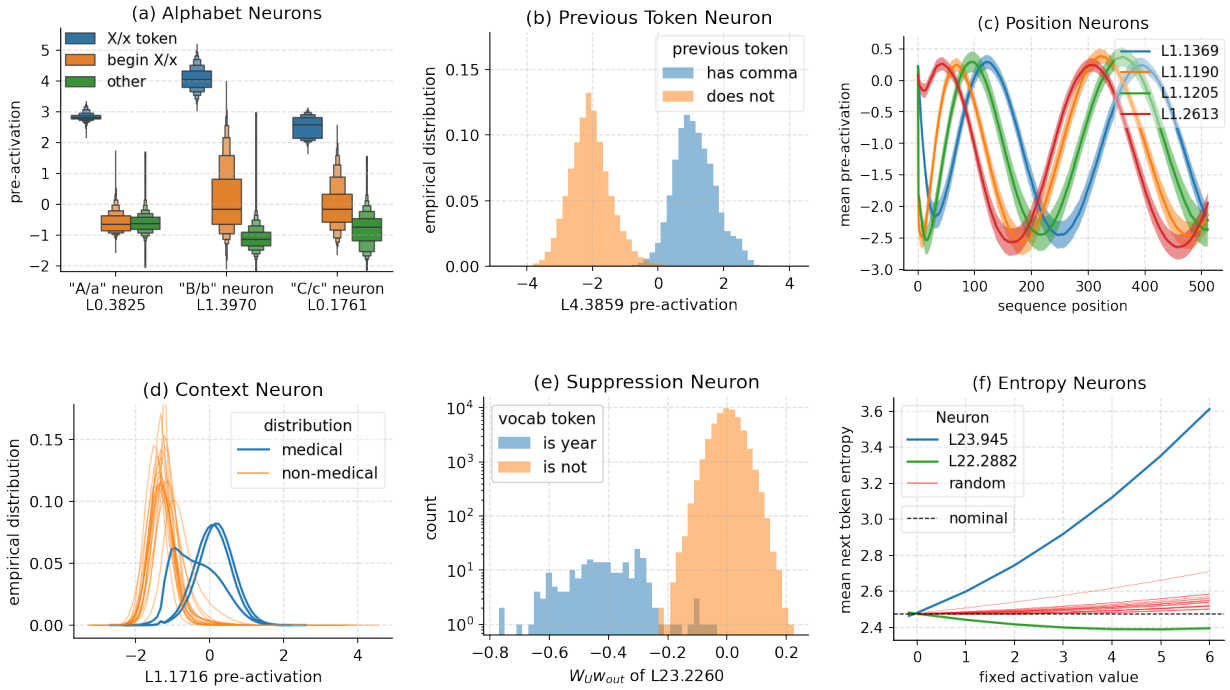


Figure 3.1: Universal neurons in GPT2 models, interpreted via their activations (a-d), weights (e), and causal interventions (f). (a) Neurons which activate primarily on a specific individual letter and secondarily on tokens which begin with the letter; (b) Neuron which activates approximately if and only if the previous token contains a comma; (c) Neurons which activate as a function of absolute token position in the context (shaded area denotes standard deviation around the mean); (d) A neuron which activates in medical contexts (e.g. pubmed abstracts) but not in non-medical distributions; (e) a neuron which decreases the probability of predicting any integer tokens between 1700 and 2050 (i.e., years); (f) Neurons which change the entropy of the next token distribution when causally intervened.

question that informs what kinds of interpretability research are most likely to be tractable and important. If the universality hypothesis is largely true in practice, we would expect detailed mechanistic analyses [58, 185, 202, 217, 274] to generalize across models such that it might be possible to develop a periodic table of neural circuits which can be automatically referenced when interpreting new models [214]. Conversely, it becomes less sensible to dedicate substantial manual labor to understand low-level details of circuits if they are completely different in every model, and instead more efficient to allocate effort to engineering scalable and automated methods that can aid in understanding and monitoring higher-level representations of particular interest [38, 48, 56, 74, 290]. However, even in the case that not all features or circuits are universal, those which are common across models are likely to be more fundamental [18, 217], and studying them should be prioritized accordingly.

In this work, we study the universality of individual neurons across GPT2 language models [229] trained from five different random initializations [148]. While it is well known that individual neurons are often polysemantic [98, 118, 205, 214] i.e., represent multiple unrelated concepts, we hypothesized that universal neurons were more likely to be monosemantic (see §B.1.1), potentially giving an approximation on the number of independently meaningful neurons. We choose to study models of the same architecture trained on the same data to have the most favorable experimental conditions for measuring universality to establish a rough bound for the universality over larger changes. We begin by operationalizing neuron universality in terms of activation correlations, that is, whether there exist pairs of neurons across different models which consistently activate on the same inputs. We compute pairwise correlations of neuron activations over 100 million tokens for every neuron pair across the different seeds and find that only 1-5% of neurons pass a target threshold of universality compared to random baselines (§ 3.4.1). We then study these universal neurons in detail, analyzing various statistical properties of both weights and activations (§ 3.4.2), and find that they usually have clear interpretations and taxonomize them into a small number of neuron families (§ 3.4.3).

In Section 3.5 we study a more abstract form of universality in terms of neuron weights rather than activations. That is, rather than understand a neuron in terms of the inputs which cause it to activate, understand a neuron in terms of the effects the neuron has on later model components or directly on the final prediction. Specifically, we analyze patterns in the compositional structure of the weights and find consistent outliers in how neurons affect other network components, constituting very simple circuits. In Section 3.5.1, we show there exists a large family of late layer neurons which have clear roles in predicting or suppressing a coherent set of tokens (e.g., second-person pronouns or single digit numbers), where the suppression neurons typically come in later layers than the prediction neurons. We then

investigate a small set of neurons that leverage the final layer-norm operation to modulate the overall entropy of the next token prediction distribution (§ 3.5.2). We conclude with an analysis of neurons which control the extent to which an attention head attends to the first token, which empirically controls the output norm of the head, effectively turning a head on or off (§ 3.5.3).

3.2 Related Work

Universal Neural Mechanisms Features and circuits like high-low frequency detectors [252] and curve circuits [58] have been found to reoccur in vision models, with some features even reappearing in biological neural networks [111]. In language models, recent research has found similarly universal circuits and components like induction heads [217] and successor heads [113] and that models reuse certain circuit components to implement different tasks [190]. There has also been a flurry of recent work on studying more abstract universal mechanisms in language models like function vectors [129, 265], variable binding mechanisms [102], and long context retrieval [270]. Studying universality in toy models has provided "mixed evidence" on the universality hypothesis [72] and shown that multiple algorithms exist to implement the same tasks [167, 289].

Representational Similarity Preceding the statement of the universality hypothesis in mechanistic interpretability, there has been substantial work measuring representational similarity [152]. Common methods include canonical correlation analysis-based measures [196, 230], alignment-based measures [86, 92, 123, 281], matrix-based measures [43, 110, 155, 170, 253, 259], neighborhood-based measures [121, 138], topology-based measures [16, 150], and descriptive statistics [160, 178, 276]. Previous work, mostly in vision models, has yielded mixed conclusions on whether networks with the same architecture learn similar representations. Some studies have found that networks with different initializations "exhibit very low similarity" [275] and "do not converge to a unique basis" [50], while others have shown that networks learn the same low-dimensional subspaces but not identical basis vectors [166] and that different models can be linearly stitched together with minimal loss suggesting they learn similar representations [14].

Analyzing Individual Neurons Many prior interpretability studies have analyzed individual neurons. In vision models, researchers have found neurons which activate for specific objects [19], curves at specific orientations [58], high frequency boundaries [251], multimodal concepts [111], as well as for facets [205] and compositions [197] thereof. Moreover, many

of these neurons seem universal across models [91]. In language models, neurons have been found to correspond to sentiment [88, 227], knowledge [78], skills [277], de-/re-tokenization [97], contexts [38, 118], position [273], space and time [117], and many other linguistic and grammatical features [18, 79, 80, 94, 245, 284]. More generally, it is hypothesized that neurons in language models form key-value stores [107] that facilitate next token prediction by promoting concepts in the vocabulary space [108]. However, many challenges exist in studying individual neurons, especially in drawing causal conclusions [10, 139].

3.3 Conceptual and Empirical Preliminaries

3.3.1 Universality

Notions of Universality Universality can refer to many different notions of similarity, each at a different level of abstraction and with differing measures and methodologies. Similar to Marr's levels of analysis in neuroscience [124, 182], relevant notions of universality are: *computational* or *functional* universality regarding whether a (sub)network implements a particular input-output-behavior (e.g., whether the next token predictions for two different networks are the same); *algorithmic* universality regarding whether or not a particular function is implemented using the same computational steps (e.g., whether a transformer trained to sort strings always learns the same sorting algorithm); *representational* universality, or the degree of similarity of the information contained within different representations [155] (e.g., whether every network represents absolute position in the context); and finally *implementation* universality, i.e., whether individual model components learned by different models implement the same specialized computations (e.g., induction heads [217], successor heads [113], French neurons [118], *inter alia*). None of these notions of universality are usually binary, and the universality between components or computations can range from being formally isomorphic to simply sharing a common high-level conceptual or statistical motif.

In this work, we are primarily concerned with implementation universality in the form of whether individual neurons learn to specialize and activate for the same inputs across models. If such universal neurons do exist, then this is also a simple form of functional universality, as the distinct neurons constitute the final node of distinct subnetworks which compute the same output.

Dimensions of Variations Universality must be measured over some independent dimension of variation, i.e., some change in the model, data or, training. For example, model variables include random seed, model size, hyperparameters, and architectural changes; data

variables include the data size, ordering, and distribution contents; training variables include loss function, optimizer, regularization, finetuning, and hyperparameters thereof. Assuming that changing random seed is the smallest change, this work primarily focuses on initialization universality in an attempt to bound the expected similarity of larger changes.

3.3.2 Models

We restrict our scope to transformer-based auto-regressive language models [228] that currently power the most capable AI systems [55]. Given an input sequence of tokens $x = [x_1, \ldots, x_t] \in \mathcal{X} \subseteq \mathcal{V}^t$ from the vocabulary $\mathcal{V}$, a language model $\mathcal{M} : \mathcal{X} \to \mathcal{Y}$ outputs a probability distribution over the vocabulary to predict the next token in the sequence.

We focus on a replication of the GPT2 series of models [229] with some supporting experiments on the Pythia family [37]. For a GPT2-small and GPT2-medium architecture (see § B.2.3 for hyperparameters) we study five models trained from different random seeds, referred to as GPT2-{small, medium}-[a-e] [148].

Anatomy of a Neuron Of particular importance to this investigation is the functional form of the neurons in the feed forward (also known as multi-layer perceptron (MLP)) layers in the transformer. The output of an MLP layer given a normalized hidden state $\mathbf{x} \in \mathbb{R}^{d_{\text{model}}}$ is

$$\text{MLP}(\mathbf{x}) = \mathbf{W}_{\text{out}}\sigma(\mathbf{W}_{\text{in}}\mathbf{x} + \mathbf{b}_{\text{in}}) + \mathbf{b}_{\text{out}} \tag{3.1}$$

where $\mathbf{W}_{\text{out}}^T, \mathbf{W}_{\text{in}} \in \mathbb{R}^{d_{\text{mlp}} \times d_{\text{model}}}$ are learned weight matrices, $\mathbf{b}_{\text{in}}$ and $\mathbf{b}_{\text{out}}$ are learned biases, and σ is an elementwise nonlinear activation function. For all models we study, σ is the GeLU activation function $\sigma(\mathbf{x}) = \mathbf{x}\Phi(\mathbf{x})$ [130]. One can analyze an individual neuron j in terms of its activation $\sigma(\mathbf{w}_{\text{in}}^j\mathbf{x} + b_{\text{in}}^j)$ for different inputs $\mathbf{x}$, or its weights—row j of $\mathbf{W}_{\text{in}}$ or $\mathbf{W}_{\text{out}}^T$ which respectively dictate for what inputs a neuron activates and what effects it has downstream.

We refer the reader to [96] for a full description of the transformer architecture. We employ standard weight preprocessing techniques described further in B.2.1.

3.4 The Search for Universal Neurons

3.4.1 How Universal are Individual Neurons?

Experiment Inspired by prior work studying common neurons in neural networks [18, 91, 165], we compute maximum pairwise correlations of neuron activations across five different

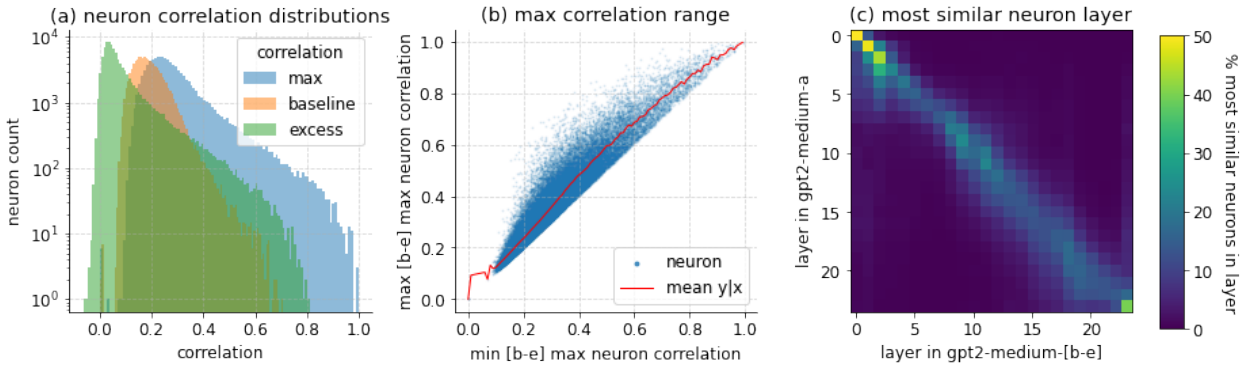


Figure 3.2: Summary of neuron correlation experiments in GPT2-medium-a. (a) Distribution of the mean (over models b-e) max (over neurons) correlation, the mean baseline correlation, and the difference (excess). (b) The max (over models) max (over neurons) correlation compared to the min (over models) max (over neuron) correlation for each neuron. (c) Percentage of layer pairs with most similar neuron pairs.

models GPT2-{a, b, c, d, e} to find pairs of neurons across models which activate on the same inputs. Let $N(a)$ be the set of neurons in model a. For each neuron $i \in N(a)$, we compute the Pearson correlation

$$\rho_{i,j}^{a,m} = \frac{\mathbb{E}\left[(\mathbf{v}^i - \mu_i)(\mathbf{v}^j - \mu_j)\right]}{\sigma_i \sigma_j} \tag{3.2}$$

with all neurons $j \in N(m)$ in every model $m \in${b, c, d, e}, where μ_i and σ_i are the mean and standard deviation of a vector of neuron activations $\mathbf{v}^i$ computed across a dataset of 100 million tokens from the Pile test set [104]. For a baseline, we also compute $\bar{\rho}_{i,j}^{a,m}$, where instead of taking the correlation of $\rho(\mathbf{v}^i, \mathbf{v}^j)$, we compute $\rho(\mathbf{v}^i, (\mathbf{RV})_j)$ for a random $d_{\text{mlp}} \times d_{\text{mlp}}$ Gaussian matrix $\mathbf{R}$ and the matrix of activations $\mathbf{V}$ for all neurons in a particular layer $N_\ell(m)$. In other words, we compute the correlation between neurons and elements within a random (approximate) rotation of a layer of neurons to establish a baseline correlation for the case where there does not exist a privileged basis [50, 96] to verify the importance of the neuron basis.

For a set of models M we define the *excess correlation* of neuron i as the difference between the mean maximum correlation across models and the mean maximum baseline correlation in the rotated basis:

$$\varrho_i = \frac{1}{|M|} \sum_{m \in M} \left(\max_{j \in N(m)} \rho_{i,j}^{a,m} - \max_{j \in N_R(m)} \bar{\rho}_{i,j}^{a,m} \right) \tag{3.3}$$

Results Figure 3.2 summarizes our results. In Figure 3.2a, we depict the average of the maximum neuron correlations across models [b-e], the average of the baseline correlations, and the excess correlation i.e., the left term, the right term, and the difference in (3.3). While there is no principled threshold at which a neuron should be deemed universal, only 1253 out of the 98304 neurons in GPT2-medium-a have an excess correlation greater than 0.5. In Figure B.6, we report the (complement) cumulative distribution of these correlation metrics to show how the number of universal neurons changes with threshold.

To understand if high (low) correlation in one model implies high (low) correlation in all the models, in Figure 3.2b we report $\max_m \max_{j \in N(x)} \rho_{i,j}^{a,m}$ compared to $\min_m \max_{j \in N(m)} \rho_{i,j}^{a,m}$ for every neuron $i \in N(a)$. Figure 3.2b suggests there is relatively little variation in the correlations, as the mean difference between the max-max and min-max correlation is 0.049 for all neurons and 0.105 for neurons with $\varrho > 0.5$. Another natural hypothesis is that neurons specialize into roles based on how deep they are within the network (as suggested by [97, 214]). In 3.2c, for each layer l of model a, we compute the fraction of neurons in layer l that have their most correlated neuron in layer l' for all ℓ' in models [b-e]. Averaging across the different models, we observe significant *depth specialization*, suggesting that neurons do perform depth specific computations, which we explore further in § 3.4.3.

We repeat these experiments on GPT2-small and Pythia-160m displayed in Figures B.4 and B.5 respectively. A rather surprising finding is that while the percentage of universal neurons ($\varrho_i > 0.5$) within GPT2-medium and Pythia-160M are quite consistent (1.23% and 1.26% respectively), the number in GPT2-small-a is far higher at 4.16%. We offer additional results and speculations in § B.3.3.

3.4.2 Properties of Universal Neurons

We now seek to understand whether there are statistical proprieties associated with whether a neuron is universal or not, defined as having an excess correlation $\varrho_i > 0.5$. For all neurons in GPT2-medium-a, GPT2-small-a, and Pythia-160m, we compute various summary statistics of their weights and activations. For activations, we compute the mean, skew, and kurtosis of the pre-activation distribution over 100 million tokens, as well as the fraction of activations greater than zero, termed activation sparsity. For weights, we record the input bias $\mathbf{b}_{\text{in}}$, the cosine similarity between the input and output weight $\cos(\mathbf{w}_{\text{in}}, \mathbf{w}_{\text{out}})$, the weight decay penalty $\|\mathbf{w}_{\text{in}}\|_2^2 + \|\mathbf{w}_{\text{out}}\|_2^2$, and the kurtosis of the neuron output weights with the unembedding $\text{kurt}(\cos(\mathbf{w}_{\text{out}}, \mathbf{w}_U))$ to measure the composition with the unembedding [81, 108].

In Figure 3.3, we report these statistics for universal neurons as a percentile compared to

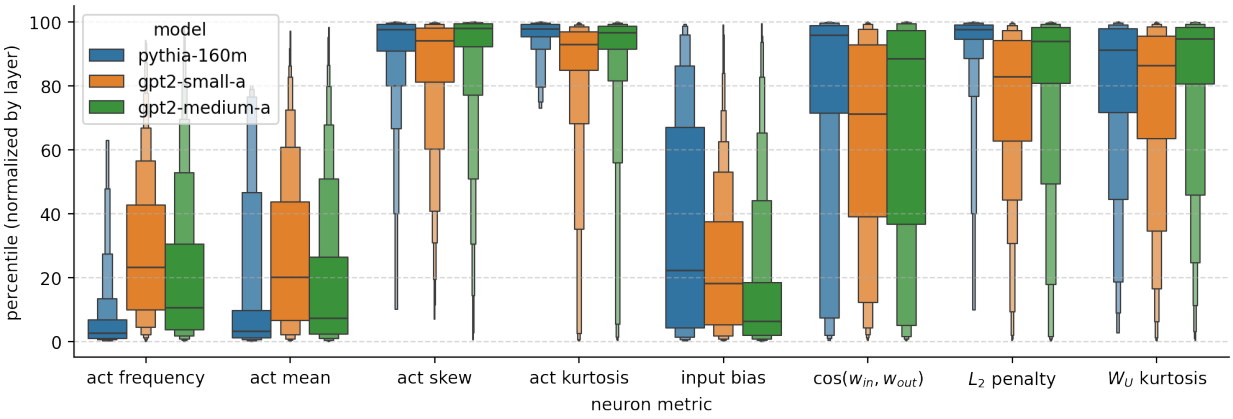
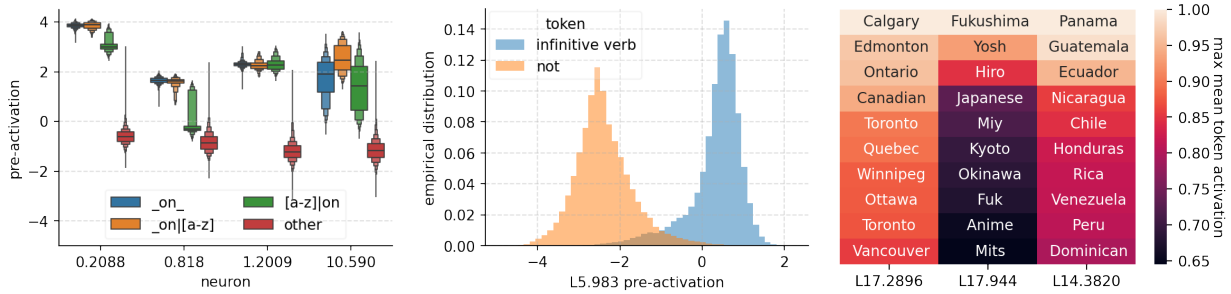


Figure 3.3: Properties of activations and weights of universal neurons for three different models, plotted as a percentile compared to neurons in the same layer.

all neurons within the same layer; we choose this normalization to enable comparison across different layers, models, and metrics (a breakdown per metric and layer for GPT2-medium-a is given in Figure B.7). Our results show that universal neurons do stand out compared to non-universal neurons. Specifically, universal neurons typically have large weight norm (implying they are important because the model was trained with weight decay) and have a large negative input bias, resulting in a large negative pre-activation mean and hence lower activation frequency. Furthermore, universal neurons have very high pre-activation skew and kurtosis, implying they usually have negative activation, but occasionally have very positive activation, proprieties we would expect of monosemantic neurons [98, 118, 214] which only activate when a specific feature is present in the input. In contrast, non-universal neurons usually have skew approximately 0 and kurtosis approximately 3, identical to a Gaussian distribution. We will discuss the meaning of high $\mathbf{W}_U$ kurtosis in § 3.5.1 and high $\cos(\mathbf{w}_{\text{in}}, \mathbf{w}_{\text{out}})$ in § B.3.

3.4.3 Universal Neuron Families

Motivated by the observation that universal neurons have distributional statistics suggestive of monosemanticity, we zoom-in on individual neurons with $\varrho > 0.5$ and attempt to group them into a partial taxonimization of neuron families [58, 213]. After manually inspecting many such neurons, we developed several hundred automated tests to classify neurons using algorithmically generated labels derived from elements of the vocabulary (e.g., whether a token `is_all_caps` or `contains_digit`) and from the NLP package spaCy [136]. Specifically, for each neuron with activation vector $\mathbf{v}$, and each test explanation which is a binary vector $\mathbf{y}$ over all tokens in the input, we compute the reduction in variance when conditioned on



(a) Similar "on" unigram neurons

(b) Syntax neuron

(c) Place Neurons

Figure 3.4: Additional examples of universal neuron families in GPT2-medium.

the explanation:

$$1 - \frac{(1-\beta)\sigma^2(\mathbf{v}|\mathbf{y}=0) + \beta\sigma^2(\mathbf{v}|\mathbf{y}=1)}{\sigma^2(\mathbf{v})} \tag{3.4}$$

where β is the fraction of positive labels and $\sigma^2(\cdot)$ is the variance of a vector or subset thereof. In words, Eq 3.4 measures the change in variance between the original activation distribution, and the weighted (by β) variance of the distribution slices where $y_i = 1$ and $y_i = 0$. As a useful intuition, this is the same metric used to decide how to split in a regression tree, where the goal is to find a split which most reduces the variance in the prediction target. Below, we qualitatively describe the most common families, and find our results replicate many findings previously documented in the literature.

Unigram Neurons The most common type of neuron we found were *unigram* neurons, which simply activate approximately if and only if the current token is a particular word or part of a word. These neurons often have many near duplicate neurons activating for the same unigram (Figure B.8) and appear predominately in the first two layers (Figure B.9). We breakdown activations of neurons responding to alphabetical unigrams based on the unigram's position in a word, as common words often have four tokenizations, and find that duplicate neurons can respond differently to unigram variations (Figures 3.4a and B.8). Such neurons illustrate that the token (un)embeddings may not contain all of the relevant token-level information, and that the model uses neurons to create an "extended" embedding of higher capacity.

Alphabet Neurons A particularly fun subclass of unigram neurons are *alphabet* neurons (Figure 3.1a), which activate most strongly on tokens corresponding to an individual letter, and secondarily on tokens which begin with the respective letter. For 18 of 26 English letters there exist alphabet neurons with $\varrho > 0.5$ (Figure B.10), with some letters also having several

near duplicate neurons.

Previous Token Neurons After finding an example of one neuron which seemed to activate purely as a function of the *previous* token (e.g., if it contains a comma; Figure 3.1b), we decided to rerun our unigram tests with the labels shifted by one—that is, with the label given by the previous token. These tests surfaced many more previous token neurons occurring most often in layers 4-6 (see Figure B.11 for an additional 25 universal previous token neurons). Such neurons illustrate the many potentially redundant paths of computations that can occur which complicates ablation based interpretability studies.

Position Neurons Inspired by the recent work of [273], we also run evaluations for *position neurons*, neurons which activate as a function of absolute position rather than token or context (Figure 3.1c). We follow the procedure of [273] (who run their experiments on OPT models with ReLU activation [287]) by computing the mutual information between activation and context position, and find similar results, with neurons that have a variety of positional patterns concentrated in layers 0-2 (see Figure B.12 for 20 more neurons). Similar to the unigram neurons, the presence of these neurons is potentially unexpected given their outputs could be learned directly by the positional embedding at the beginning of the model with less variance in activation.

Syntax Neurons Using the NLP package spaCy [136], we label our input data with part-of-speech, dependency role, and morphological data. We find many individual neurons that selectively activate for basic linguistic features like negation, plurals, and verb forms (Figure 3.4b) which are not concentrated to any part of the network and resemble past findings on linguistic properties [79, 94]. Figure B.13 includes 25 more examples.

Semantic Neurons Finally, we found a large number of neurons which activate for semantic features corresponding to coherent topics [169], concepts [97], or contexts [118]. Such features are naturally much harder to algorithmically supervise. We use the subdistribution label from the Pile dataset [104] and manually labeled topics from an SVD based topic model as a best attempt, but this leaves many interpretable neurons undiscovered and uncategorized. In Figure 3.4c, we show three regions neurons which activate most strongly on tokens corresponding to places in Canada, Japan, or Latin America respectively. Figure B.14 depicts 30 additional context neurons which activate on specific subdistributions, with many neurons which always activate for non-english text.

3.5 Universal Functional Roles of Neurons

While the previous discussion was primarily focused on analyzing the *activations* of neurons, and by extension the features they represent, this section is dedicated to studying the *weights* of neurons to better understand their downstream effects. The neurons in this section are examples of *action* mechanisms [9]—model components that are better thought of as implementing an action rather than purely extracting or representing a feature, analogous to motor neurons in neuroscience.

3.5.1 Prediction Neurons

A simple but effective method to understand weights is through logit attribution techniques [81, 108, 206]. Because the final residual stream is the sum of all previous layers, we can approximate a neuron's effect on the final prediction logits by simply computing the product between the unembedding matrix and a neuron output weight $\mathbf{W}_U\mathbf{w}_{\text{out}}$ and hence interpret the neuron based on how it promotes concepts in the vocabulary space [108].

When we apply our automated tests from § 3.4.3 on $\mathbf{W}_U\mathbf{w}_{\text{out}}$ rather than the activations for our universal neurons, we found several general patterns (Figure 3.5), many individual neurons with extremely clear interpretations (Figure B.16), and clusters of neurons which all affect the same tokens (Figure B.17). Specifically, we find many examples of *prediction* neurons that positively increase the predicted probability of a coherent set of tokens while leaving most others approximately unchanged (Fig 3.5a); *suppression* neurons that are similar, except decrease the probability of a group of related tokens (Fig 3.5b); and *partition* neurons that partition the vocabulary into two groups, increasing the probability of one while decreasing the probability of the other (Fig 3.5c). The prediction, suppression, and partition motifs can be automatically detected by studying the moments of the distribution of vocabulary effects given by $\mathbf{W}_U\mathbf{w}_{\text{out}}$. In particular, both prediction and suppression neurons will have high kurtosis (the fourth moment—a measure of how much mass is in the tails of a distribution), but prediction neurons will have positive skew and suppression neurons will have negative skew. Partition neurons will shift the probability of most tokens and have high variance in overall logit effect. From this, we see almost all universal neurons ($\varrho > 0.5$) in later layers are one of these prediction neuron variants (Figure B.7).

To better understand the number and location of these prediction neurons, we compute the moment metrics of $\cos(\mathbf{W}_U, \mathbf{w}_{\text{out}})$ for all neurons in all five GPT2-medium models, and show how these statistics vary over model depth in Figure 3.6. We find a striking pattern which is quite consistent across the different seeds: after about the halfway point in the

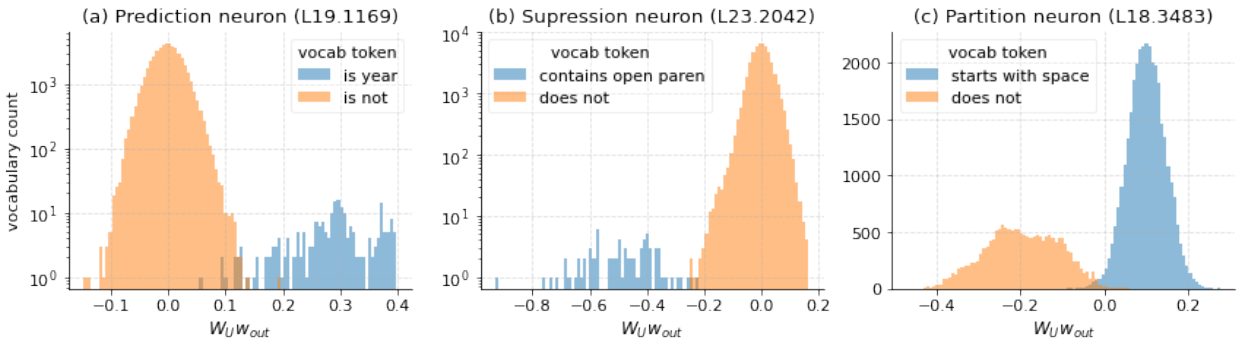


Figure 3.5: Example prediction neurons in GPT2-medium-a. Depicts the distribution of logit effects on the output vocabulary ($\mathbf{W}_U\mathbf{w}_{\text{out}}$) split by token property for 3 different neurons. (a) Prediction neuron increasing logits of integer tokens between 1700 and 2050 (i.e. years; high kurtosis), (b) Suppression neuron decreasing logits for tokens containing an open parenthesis (high kurtosis and negative skew), and (c) Partition neuron boosting tokens beginning with a space and suppressing tokens which do not (high variance).

model, prediction neurons become increasingly prevalent until the very end of the network where there is a sudden shift towards suppression neurons. To ensure this is not just an artifact of the tied embeddings ($\mathbf{W}_E = \mathbf{W}_U^T$) in the GPT2 models, we also run this analysis on five Pythia models ranging from 410M to 6.9B parameters and find the results are largely the same (Figure B.15).

We observed an interesting pattern when examining the activations of suppression neurons: they activate much more frequently when the next token actually belongs to the set of tokens they suppress the probability of predicting. In other words, neurons which *decrease* the probability that the next token is a year (e.g. "1970"), activate much more often when the next token is actually a year compared to not. We intuit that these suppression neurons fire when it is plausible but not certain that the next token is from the relevant set. Combined with the observation that there exist many suppression and prediction neurons for the same token class (Figure B.17), we take this as evidence of an ensemble hypothesis where the model uses multiple neurons with some independent error that combine to form a more robust and calibrated estimate of whether the next token is in fact a year.

In addition to being a clean example of an action mechanism [9], these results are interesting as they refine a conjecture made by [108]. Specifically, rather than "feed-forward layers build predictions by promoting concepts in the vocabulary space," we claim *late* feed-forward (MLP) layers build predictions by both promoting *and* suppressing concepts in the vocabulary space. Moreover, it suggests there are different stages in the iterative inference pipeline [21, 140], where first affirmative predictions are made, and then the distribution is sharpened or made more calibrated by suppression neurons at the very end. The existence

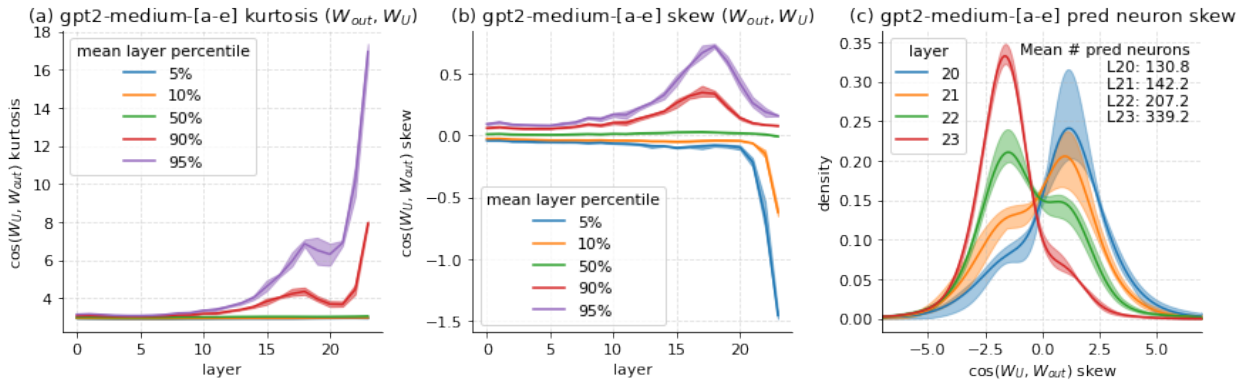


Figure 3.6: Summary statistics of cosine similarity between neuron output weights ($\mathbf{W}_{\text{out}}$) and token unembedding ($\mathbf{W}_U$) for GPT2-medium-[a-e]. (a,b) Percentiles of kurtosis and skew by layer averaged over [a-e]. (c) Distribution of skews for neurons with kurtosis greater than 10 in last four layers. Shaded area denotes range across all five models.

of suppression neurons also sheds light on recent observations of individual neurons [38] and MLP layers [186] suppressing the maximum likelihood token and being a mechanism for self-repair.

3.5.2 Entropy Neurons

Because models are trained with weight decay (ℓ_2 regularization) we hypothesized that neurons with large weight norms would be more interesting or important because they come at a higher cost. While most turned out to be relatively uninteresting (mostly neurons which activate for the beginning of sequence token), the 15^{th} largest norm neuron in GP2-medium-a (L23.945) had an especially interesting property: it had the lowest variance logit effect $\mathbf{W}_U\mathbf{w}_{\text{out}}$ of any neuron in the model; i.e., it only has a tiny effect on the logits. To understand why a final layer neuron, which can only affect the final logit distribution, has high weight norm while performing an approximate no-op on the logits, recall the final decoding formula for the probability of the next token given a final residual stream vector $\mathbf{x}$

$$p(\mathbf{y}|\mathbf{x}) = \text{Softmax}(\mathbf{W}_U\text{LayerNorm}(\mathbf{x})), \qquad \text{LayerNorm}(\mathbf{x}) = \frac{\mathbf{x} - \mathbb{E}[\mathbf{x}]}{\sqrt{\text{Var}[\mathbf{x}] + \epsilon}}. \tag{3.5}$$

We hypothesize that the function of this neuron is to modulate the model's uncertainty over the next token by using the layer norm to squeeze the logit distribution, in a manner quite similar to manually increasing the temperature when performing inference. To support this hypothesis, we perform a causal intervention, fixing the neuron in question to a particular value and studying the effect compared to 20 random neurons from the last two layers that

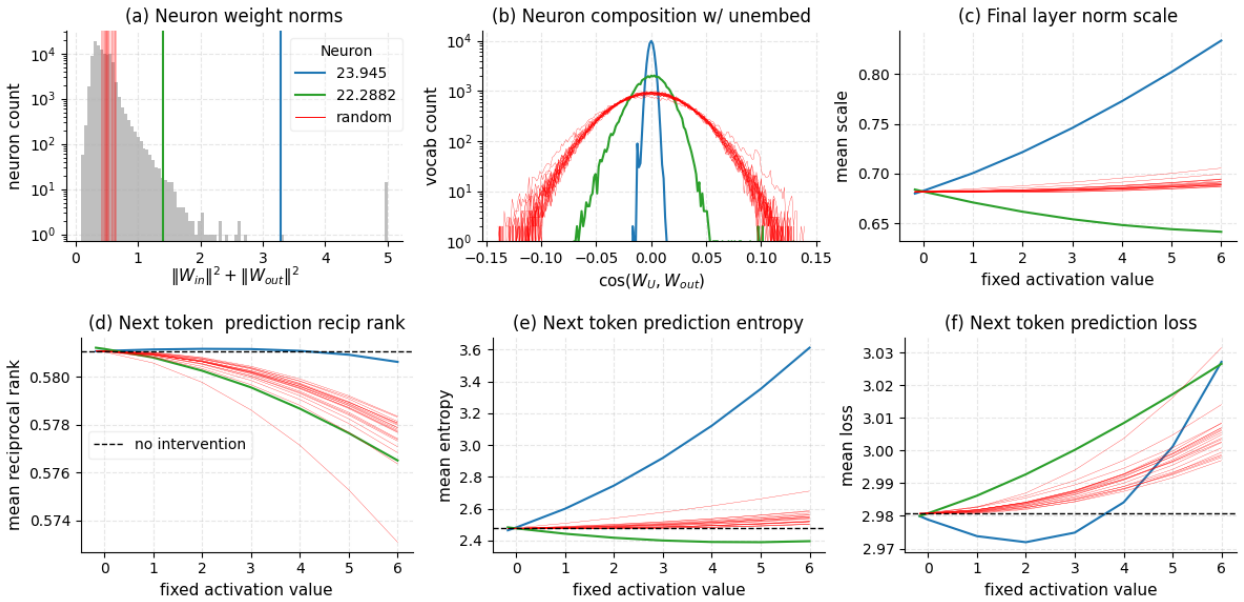


Figure 3.7: Summary of (anti-)entropy neurons in GPT2-medium-a compared to 20 random neurons from final two layers. Entropy neurons have high weight norm (a) with output weights mostly orthogonal to the unembedding matrix (b). Fixing the activation to larger values causes the final layer norm scale to increase dramatically (c) while leaving the ranking of the true next token prediction mostly unchanged (d). Increased layer norm scale squeezes the logit distribution, causing a large increase in the prediction entropy (e; or decrease for anti-entropy neuron) and an increase or decrease in the loss depending on the model's baseline level of under- or over-confidence (f). Legend applies to all subplots.

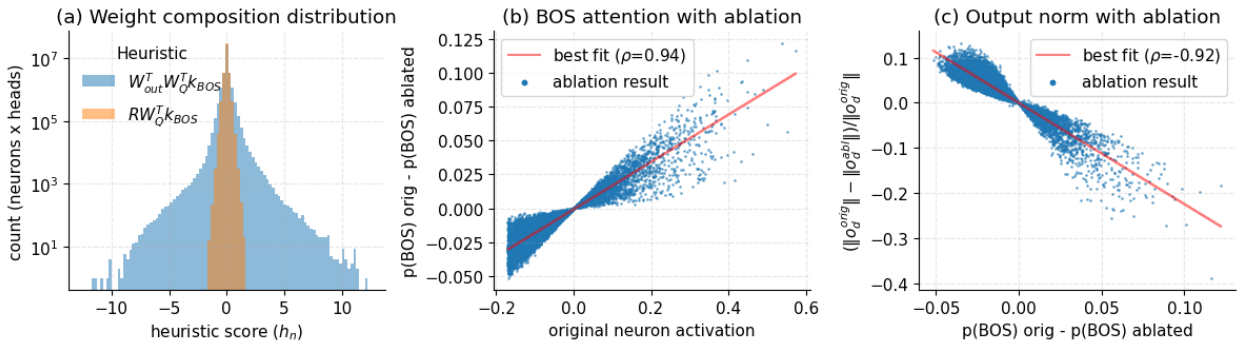


Figure 3.8: Summary of attention (de-)activation neuron results in GPT2-medium-a. (a) Distribution of heuristic score h_n for every pair of neurons and heads compared to random neuron directions $\mathbf{R}$. (b;c) path ablations effect of neuron L4.3594 on head L5.H0: ablating positive activation reduces attention to BOS (b) causing the norm to increase (c).

are not in the top decile of norm or in the bottom decile of logit variance (Figure 3.7). We find that intervening on this *entropy* neuron indeed causes the layer norm scale to increase dramatically (because of the large weight norm) while largely not affecting the relative ordering of the vocabulary (because of the low composition), having the effect of increasing overall entropy by dampening the post-layer norm component of $\mathbf{x}$ in the row space of $\mathbf{W}_U$.

Additionally, we observed a neuron (L22.2882) with $\cos(\mathbf{w}_{\text{out}}^{23.945}, \mathbf{w}_{\text{out}}^{22.2882}) = -0.886$ (i.e., a neuron that writes in the opposite direction forming an antipodal pair [98]) that also has high weight norm. Repeating the intervention experiment, we find this neuron *decreases* the layer norm scale and decreases the mean next token entropy, forming an anti-entropy neuron. These results suggest there may be one or more global uncertainty directions that the model maintains to modulate its overall confidence in its prediction. However, our experiments with fixed activation value do not necessarily imply the model uses these neurons to increase the entropy as a general uncertainty mechanism, and we did notice cases in which increasing the activation of the entropy neuron decreased entropy, suggesting the true mechanism may be more complicated.

We repeat these experiments on GPT2-small-a and find an even more dramatic antipodal pair of (anti-)entropy neurons in Figure B.18. To our knowledge, this is the first documented mechanism for uncertainty quantification in language models and the second example of a mechanism involving layer norm [49].

3.5.3 Attention Deactivation Neurons

In autoregressive models, attention heads frequently place all of their attention on the beginning of sequence (BOS) token [283]. We hypothesise that the model uses the attention to the BOS token as a kind of (de-)activation for the head, where fully attending to BOS implies the head is deactivated and has minimal effect. Moreover, we hypothesize that there are individual neurons which control the extent to which heads attend to BOS.

Recall the output of an attention head $\mathbf{o}_d$ for a destination token d from source tokens s is given by

$$\mathbf{q}_d = \mathbf{W}_Q \mathbf{r}_d, \quad \mathbf{k}_s = \mathbf{W}_K \mathbf{r}_s, \quad \mathbf{S}_{ds} = \mathbf{q}_d^T \mathbf{k}_s, \quad \mathbf{A}_{ds} = \text{softmax}_s(\frac{M(\mathbf{S}_{ds})}{\sqrt{d_h}}),$$

$$\mathbf{v}_s = \mathbf{W}_V \mathbf{r}_s, \quad \mathbf{o}_d = \mathbf{W}_O \sum_s \mathbf{A}_{ds} \mathbf{v}_s$$

where $\mathbf{r}_{s/d}$ is the residual stream at the source / destination token, d_h is the bottleneck dimension of the head, and $M(\cdot)$ applies the causal attention mask to the attention scores. The calculation of the attention pattern $\mathbf{A}_{ds}$ via a softmax across the source positions means that the attention given to the source tokens by a given destination token sums to one.

Assuming the BOS token is always the first token in the context, the vector $\mathbf{W}_O \mathbf{v}_{BOS}$ is constant for all prompts and contains no semantic information. If it has a low norm, attending to BOS scales down the outputs of attending to other source positions while maintaining their relative attention because the attention scores must sum to one. If the BOS output norm is near zero, the head can effectively turn off by only attending to the BOS token. In practice, the median head in GPT-2-medium-a has a $\mathbf{W}_O \mathbf{v}_{BOS}$ with norm 19.4 times smaller than the average for other tokens.

We can identify neurons which may use this mechanism by a heuristic score $h_n = \mathbf{W}_{out}^T \mathbf{W}_Q^T \mathbf{k}_{BOS}$ for unit normalized $\mathbf{W}_{out}$. Positive scores suggest activation of the neuron will increase the attention placed on BOS, decreasing the output norm of the head, and the opposite for negative scores. Figure 3.8a shows the distribution of the scores for all heads in GPT2-medium-a compared to a unit normalized Gaussian matrix $\mathbf{R}$.

For a given neuron, we can measure the effect of activation on the attention to BOS and output norm of a given head by path ablation [274] of the neuron at a particular destination token. Specifically, we can measure the difference in BOS attention and norm of the output of the head between the original run and a forward pass where the contribution of a neuron is deleted (i.e, zero path ablated) from the input of a particular head at the current token position. We perform this procedure over a random subset of tokens in the second half of

the context to avoid spurious effects stemming from short contexts. Figure 3.8b and 3.8c depict the results of these path ablations for the highest scoring neuron in layer 4 for head 0 in attention layer 5. This is an example of an attention deactivation neuron—increasing the activation of the neuron increases the attention to BOS reducing the output norm of the head $\|\mathbf{o}_d\|$. See Figure B.19 for 5 additional examples of attention (de-)activating neurons.

3.6 Discussion and Conclusion

Findings In this work, we explore the universality of individual neurons in GPT2 language models, and find that only about 1-5% of neurons pass a certain threshold of universality across models. We have shown that leveraging universality is an effective unsupervised approach to identify interpretable model components and important motifs. In particular, those few neurons which are universal are often interpretable, can be grouped into a smaller number of neuron families, and often develop with near duplicate neurons in the same model. Some universal neurons also have clear functional roles, like modulating the next token prediction entropy, controlling the output norm of an attention head, and predicting or suppressing elements of the vocabulary in the prediction. Moreover, these functional neurons often form antipodal pairs, potentially enabling collections of neurons to ensemble to improve robustness and calibration. These findings raise useful lessons and motifs for further interpretability research (§B.1.2).

Limitations Compared to frontier LLMs, we study small models of only hundreds of million parameters and tens of thousands of neurons due to the expense of training multiple large scale language models from different random initializations. We also study a relatively narrow form of universality: neuron universality over random seeds within the same model family. Studying universality across different model families is made difficult by tokenization discrepancies, and studying models across larger sizes is difficult due to the expense of computing all pairwise neuron correlations over a sufficiently sized text corpus. Additionally, many of our interpretations rely on manual analysis or algorithmic supervision which restricts the scope and generality of our methods. Moreover, our narrow focus on a subset of individual elements of the neuron basis potentially obscures important details and ignores the vast majority of overall network computation.

Future Work Each of these limitations suggest avenues for future work. Instead of studying the neuron basis, our experiments could be replicated on an overcomplete dictionary basis that is more likely to contain the true model features [48, 77]. Motivated by the finding that

the most correlated neurons occur in similar network depths, our experiments could be re-run on larger models where pairwise correlations are only computed between adjacent layers to improve scalability. Additionally, the interpretation of common units could be further automated using LLMs to provide explanations [38]. Finally, by uncovering interpretable footholds within the internals of the network, our findings can form the basis of deeper investigations into how these components respond to stimulus or perturbation, develop over training [225], and affect downstream components to further elucidate general motifs and specific circuits within language models.

Acknowledgments

This chapter was based off of joint work [119] with Theo Horsley, Zifan Carl Guo, Tara Rezaei Kheirkhah, Qinyi Sun, Will Hathaway, Neel Nanda, and Dimitris Bertsimas.

We would like to thank Yossi Gandelsman, Lovis Heindrich, and Lucia Quirke for useful discussions and comments on our work. We made extensive use of the TransformerLens library [199] and the MIT Supercloud [236] for our experiments and computational resources. WG was partially supported by an Open Philanthropy early career grant.

Chapter 4

Language Models Represent Space and Time

4.1 Introduction

Despite being trained to just predict the next token, modern large language models (LLMs) have demonstrated an impressive set of capabilities [55, 279], raising questions and concerns about what such models have actually learned. One hypothesis is that LLMs learn a massive collection of correlations but lack any coherent model or "understanding" of the underlying data generating process given text-only training [22, 40]. An alternative hypothesis is that LLMs, in the course of compressing the data, learn more compact, coherent, and interpretable models of the generative process underlying the training data, i.e., a *world model*. For instance, [164] have shown that transformers trained with next token prediction to play the board game Othello learn explicit representations of the game state, with [203] subsequently showing these representations are linear. Others have shown that LLMs track boolean states of subjects within the context [163] and have representations that reflect perceptual and conceptual structure in spatial and color domains [1, 221]. Better understanding of if and how LLMs model the world is critical for reasoning about the robustness, fairness, and safety of current and future AI systems [23, 45, 131, 204, 280].

In this work, we take the question of whether LLMs form world (and temporal) models as literally as possible—we attempt to extract an actual map of the world! While such spatiotemporal representations do not constitute a dynamic causal world model in their own right, having coherent multi-scale representations of space and time are basic ingredients required in a more comprehensive model.

Specifically, we construct six datasets containing the names of places or events with

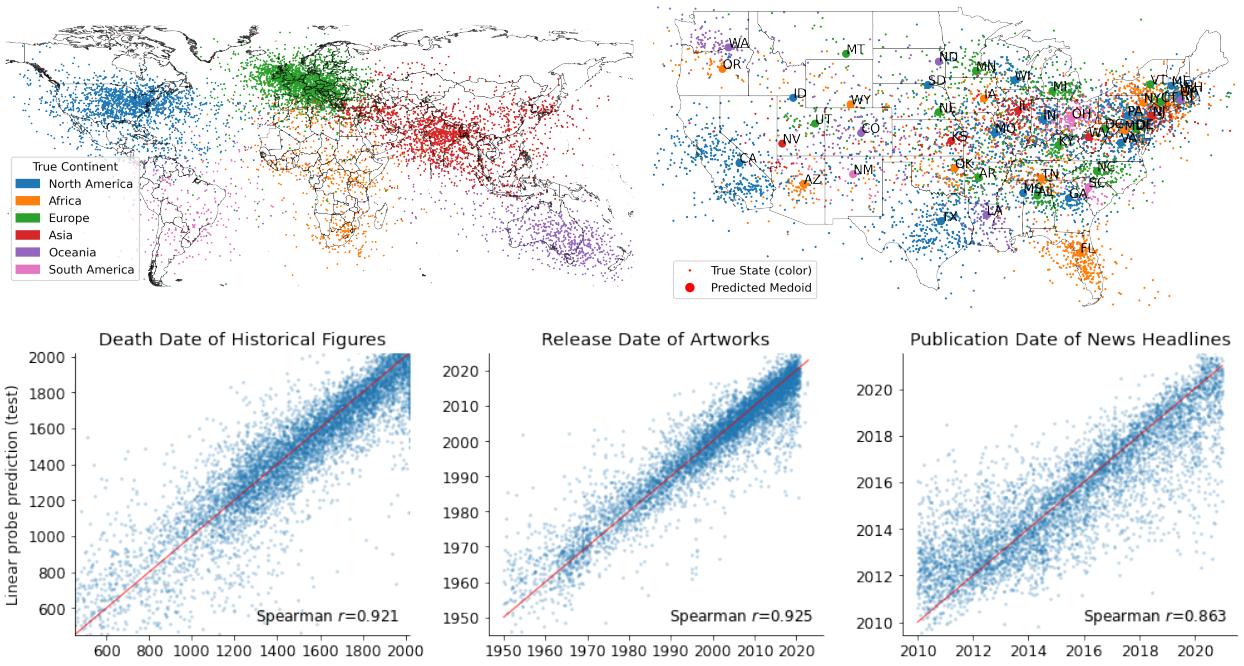


Figure 4.1: Spatial and temporal representations of Llama-2-70b. Each point corresponds to the layer 50 activations of the last token of a place (top) or event (bottom) projected on to a learned linear probe direction. All points depicted are from the test set.

corresponding space or time coordinates that span multiple spatiotemporal scales: locations within the whole world, the United States, and New York City in addition to the death year of historical figures from the past 3000 years, the release date of art and entertainment from 1950s onward, and the publication date of news headlines from 2010 to 2020. Using the Llama-2 [267] and Pythia [37] family of models, we train linear regression probes [5, 20] on the internal activations of the names of these places and events at each layer to predict their real-world location (i.e., latitude/longitude) or time (numeric timestamp).

These probing experiments reveal evidence that models build spatial and temporal representations throughout the early layers before plateauing at around the model halfway point with larger models consistently outperforming smaller ones (§ 4.3.1). We then show these representations are (1) linear, given that nonlinear probes do not perform better (§ 4.3.2), (2) fairly robust to changes in prompting (§ 4.3.3), and (3) unified across different kinds of entities (e.g. cities and natural landmarks). We then conduct a series of robustness checks to understand how our probes generalize across different data distributions (§ 4.4.1) and how probes trained on the PCA components perform (§ 4.4.2). Finally, we use our probes to find individual neurons which activate as a function of space or time and use basic causal interventions to verify their importance in spatiotemporal modeling, providing strong evidence that the model is truly using these features (§ 4.5).

4.2 Empirical Overview

4.2.1 Space and Time Raw Datasets

To enable our investigation, we construct six datasets of names of *entities* (people, places, events, etc.) with their respective location or occurrence in time, each at a different order of magnitude of scale. For each dataset, we included multiple types of entities, e.g., both populated places like cities and natural landmarks like lakes, to study how unified representations are across different object types. Furthermore, we maintain or enrich relevant metadata to enable analyzing the data with more detailed breakdowns, identify sources of train-test leakage, and support future work on factual recall within LLMs. We also attempt to deduplicate and filter out obscure or otherwise noisy data.

Space We constructed three datasets of place names within the world, the United States, and New York City. Our world dataset is built from raw data queried from DBpedia [161]. In particular, we query for populated places, natural places, and structures (e.g. buildings or infrastructure). We then match these against Wikipedia articles, and filter out entities which do not have at least 5,000 page views over a three year period. Our United States dataset is constructed from DBPedia and a census data aggregator, and includes the names of cities, counties, zipcodes, colleges, natural places, and structures where sparsely populated or viewed locations were similarly filtered out. Finally, our New York City dataset is adapted from the NYC OpenData points of interest dataset [207] containing locations such as schools, churches, transportation facilities, and public housing within the city.

Time Our three temporal datasets consist of (1) the names and occupations of historical figures who died between 1000BC and 2000AD adapted from [8]; (2) the titles and creators of songs, movies, and books from 1950 to 2020 constructed from DBpedia with the Wikipedia page views filtering technique; and (3) New York Times news headlines from 2010-2020 from news desks that write about current events, adapted from [13].

4.2.2 Models and Methods

Data Preparation All of our experiments are run with the base Llama-2 [267] series of auto-regressive transformer language models, spanning 7 billion to 70 billion parameters. For each dataset, we run every entity name through the model, potentially prepended with a short prompt, and save the activations of the hidden state (residual stream) on the last

Table 4.1: Entity count and representative examples for each of our datasets.

Dataset	Count	Examples
World	39585	"Los Angeles", "St. Peter's Basilica", "Caspian Sea", "Canary Islands"
USA	29997	"Fenway Park", "Columbia University", "Riverside County"
NYC	19838	"Borden Avenue Bridge", "Trump International Hotel"
Figures	37539	"Cleopatra", "Dante Alighieri", "Carl Sagan", "Blanche of Castile"
Art	31321	"Stephen King's It", "Queen's Bohemian Rhapsody"
Headlines	28389	"Pilgrims, Fewer and Socially Distanced, Arrive in Mecca for Annual Hajj"

entity token for each layer. For a set of n entities, this yields an $n \times d_{model}$ activation dataset for each layer.

Probing To find evidence of spatial and temporal representations in LLMs, we use the standard technique of probing [5, 20], which fits a simple model on the network activations to predict some target label associated with labeled input data. In particular, given an activation dataset $\boldsymbol{A} \in \mathbb{R}^{n \times d_{model}}$, and a target $\boldsymbol{Y}$ containing either the time or two-dimensional latitude and longitude coordinates, we fit linear ridge regression probes

$$\hat{\boldsymbol{W}} = \arg\min_{\boldsymbol{W}} \|\boldsymbol{Y} - \boldsymbol{A}\boldsymbol{W}\|_2^2 + \lambda \|\boldsymbol{W}\|_2^2 = (\boldsymbol{A}^T\boldsymbol{A} + \lambda \boldsymbol{I})^{-1}\boldsymbol{A}^T\boldsymbol{Y}$$

yielding a linear predictor $\hat{\boldsymbol{Y}} = \boldsymbol{A}\hat{\boldsymbol{W}}$. High predictive performance on out-of-sample data indicates that the base model has temporal and spatial information linearly decodable in its representations, although this does not imply that the model actually uses these representations [234]. In all experiments, we tune λ using efficient leave-out-out cross validation [127] on the probe training set.

4.2.3 Evaluation

To evaluate the performance of our probes we report standard regression metrics such as R^2 and Spearman rank correlation on our test data (correlations averaged over latitude and longitude for spatial features). An additional metric we compute is the *proximity error* for each prediction, defined as the fraction of entities predicted to be closer to the target point than the prediction of the target entity. The intuition is that for spatial data, absolute error metrics can be misleading (a 500km error for a city on the East Coast of the United States is far more significant than a 500km error in Siberia), so when analyzing errors per prediction, we often report this metric to account for the local differences in desired precision.

4.3 Linear Models of Space and Time

4.3.1 Existence

We first investigate the following empirical questions: do models represent time and space at all? If so, where internally in the model? Does the representation quality change substantially with model scale? In our first experiment, we train probes for every layer of Llama-2-{7B, 13B, 70B} and Pythia-{160M, 410M, 1B, 1.4B, 2.8B, 6.9B} for each of our space and time datasets. Our main results, depicted in Figure 4.2, show fairly consistent patterns across datasets. In particular, both spatial and temporal features can be recovered with a linear probe, these representations smoothly increase in quality throughout the first half of the layers of the model before reaching a plateau, and the representations are more accurate with increasing model scale. The gap between the Llama and Pythia models is especially striking, and we suspect is due to the large difference in pre-training corpus size (2T and 300B tokens respectively). For this reason, we report the rest of our results on just the Llama models.

The dataset with the worst performance is the New York City dataset. This was expected given the relative obscurity of most of the entities compared with other datasets. However, this is also the dataset where the largest model has the best relative performance, suggesting that sufficiently large LLMs could eventually form detailed spatial models of individual cities.

4.3.2 Linear Representations

Within the interpretability literature, there is a growing body of evidence supporting the *linear representation hypothesis* that features within neural networks are represented linearly, that is, the presence or strength of a feature can be read out by projecting the relevant activation on to some feature vector [98, 195, 214]. However, these results are almost always for binary or categorical features, unlike the continuous features of space or time.

To test whether spatial and temporal features are represented linearly, we compare the performance of our linear ridge regression probes with that of substantially more expressive nonlinear MLP probes of the form $\boldsymbol{W}_2\text{ReLU}(\boldsymbol{W}_1\boldsymbol{x} + \boldsymbol{b}_1) + \boldsymbol{b}_2$ with 256 neurons. Table 4.2 reports our results and shows that using nonlinear probes results in minimal improvement to R^2 for any dataset or model. We take this as strong evidence that space and time are also represented linearly (or at the very least are linearly decodable), despite being continuous.

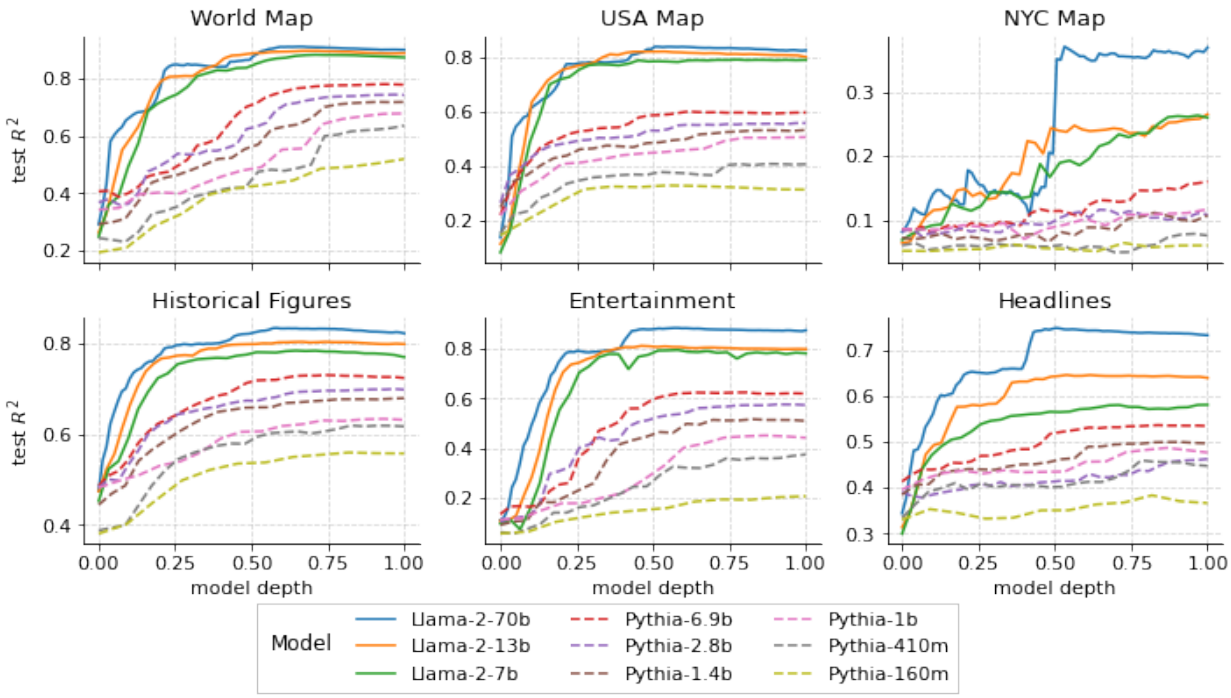


Figure 4.2: Out-of-sample R^2 for linear probes trained on every model, dataset, and layer.

4.3.3 Sensitivity to Prompting

Another natural question is if these spatial or temporal features are sensitive to prompting, that is, can the context induce or suppress the recall of these facts? Intuitively, for any entity token, an autoregressive model is incentivized to produce a representation suitable for addressing any future possible context or question.

To study this, we create new activation datasets where we prepend different prompts to each of the entity tokens, following a few basic themes. In all cases, we include an "empty"

Table 4.2: Out-of-sample R^2 of linear and nonlinear (one layer MLP) probes for all models and features at 60% layer depth.

		Dataset					
Model	Probe	World	USA	NYC	Historical	Entertainment	Headlines
Llama-2-7b	Linear	0.881	0.799	0.219	0.785	0.788	0.564
	MLP	0.897	0.819	0.204	0.775	0.746	0.467
Llama-2-13b	Linear	0.896	0.825	0.237	0.804	0.806	0.645
	MLP	0.916	0.824	0.230	0.818	0.808	0.656
Llama-2-70b	Linear	0.911	0.864	0.359	0.835	0.885	0.746
	MLP	0.926	0.869	0.312	0.839	0.884	0.739

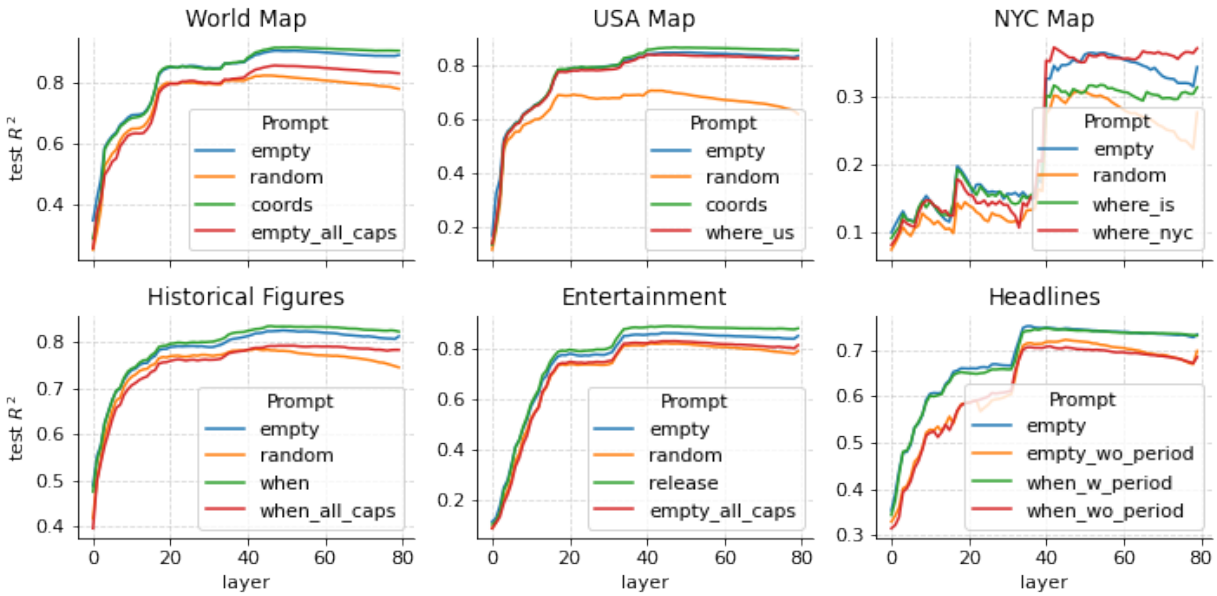


Figure 4.3: Out-of-sample R^2 when entity names are included in different prompts for Llama-2-70b.

prompt containing nothing other than the entity tokens (and a beginning of sequence token). We then include a prompt which asks the model to recall the relevant fact, e.g., "What is the latitude and longitude of `<place>`" or "What was the release date of `<author>`'s `<book>`." For the United States and NYC datasets we also include versions of these prompts asking where in the US or NYC this location is, in an attempt to disambiguate common names of places (e.g. City Hall). As a baseline we include a prompt of 10 random tokens (sampled for each entity). To determine if we can obfuscate the subject, for some datasets we fully capitalize the names of all entities. Lastly, for the headlines dataset, we try probing on both the last token and on a period token appended to the headline.

We report results for the 70B model in Figure 4.3 and all models in Figure C.3. We find that explicitly prompting the model for the information, or giving disambiguation hints like that a place is in the US or NYC, makes little to no difference in performance. However, we were surprised by the degree to which random distracting tokens degrades performance. Capitalizing the entities also degrades performance, though less severely and less surprisingly, as this likely interferes with "detokenizing" the entity [97, 109, 118]. The one modification that did notably improve performance is probing on the period token following a headline, suggesting that periods are used to contain some summary information of the sentences they end.

4.4 Robustness Checks

The previous section has shown that the true point in time or space of diverse types of events or locations can be linearly recovered from the internal activations of the mid-to-late layers of LLMs. However, this does not imply if (or how) a model actually uses the feature direction learned by the probe, as the probe itself could be learning some linear combination of simpler features which are actually used by the model.

4.4.1 Verification via Generalization

Block holdout generalization To illustrate a potential issue with our results, consider the task of representing the full world map. If the model has, as we expect it does, an almost orthogonal binary feature for `is_in_country_X`, then one could construct a high quality latitude (longitude) probe by summing these orthogonal feature vectors for each country with coefficient equal to the latitude (longitude) of that country. Assuming a place is in only one country, such a probe would place each entity at its country centroid. However, in this case, the model does not actually represent space, only country membership, and it is only the probe which learns the geometry of the different countries from the explicit supervision.

To better distinguish these cases, we analyze how the probes generalize when holding out specific blocks of data. In particular, we train a series of probes, where for each one, we hold out one country, state, borough, century, decade, or year for the world, USA, NYC, historical figure, entertainment, and headlines dataset respectively. We then evaluate the probes on the held out block of data. In Table 4.3, we report the average proximity error for the block of data when completely held out, compared to the error of the test points from that block in the default train-test split, averaged over all held out blocks.

We find that while generalization performance suffers, especially for the spatial datasets, it is clearly better than random. By plotting the predictions of the held out states or countries in Figures C.6 and C.7, a qualitatively clearer picture emerges. That is, the probe correctly generalizes by placing the points in the correct relative position (as measured by the angle between the true and predicted centroid) but not in their absolute position. We take this as weak evidence that the probes are extracting explicitly learned features by the model, but are memorizing the transformation from model coordinates to human coordinates. However, this does not fully rule out the underlying binary features hypothesis, as there could be a hierarchy of such features that do not follow country or decade boundaries.

Table 4.3: Average proximity error across blocks of data (e.g., countries, states, decades) when included in the training data compared to completely held out. Random performance is 0.5.

		Dataset					
Model	Block	World	USA	NYC	Historical	Entertainment	Headlines
Llama-2-7b	nominal	0.071	0.144	0.331	0.129	0.147	0.258
	held out	0.170	0.192	0.473	0.133	0.158	0.264
Llama-2-13b	nominal	0.068	0.144	0.319	0.121	0.141	0.223
	held out	0.156	0.189	0.470	0.126	0.152	0.235
Llama-2-70b	nominal	0.071	0.121	0.262	0.115	0.105	0.182
	held out	0.164	0.188	0.433	0.119	0.122	0.200

Cross entity generalization Implicit in our discussion so far is the claim that the model represents the space or time coordinates of different types of entities (like cities or natural landmarks) in a unified manner. However, similar to the concern that a latitude probe could be a weighted sum of membership features, a latitude probe could also be the sum of different (orthogonal) directions for the latitudes of cities and for the latitudes of natural landmarks.

Similar to the above, we distinguish these hypotheses by training a series of probes where the train-test split is performed to hold out all points of a particular entity class.[1] Table 4.4 reports the proximity error for the entities in the default test split compared to when heldout, averaged over all such splits as before. The results suggest that the probes largely generalize across entity types, with the main exception of the entertainment dataset.[2]

Table 4.4: Average proximity error across entity subtypes (e.g. books and movies) when included in the training data compared to being fully held out. Random performance is 0.5.

		Dataset					
Model	Entity	World	USA	NYC	Historical	Entertainment	Headlines
Llama-2-7b	nominal	0.120	0.206	0.313	0.164	0.224	0.199
	held out	0.151	0.262	0.367	0.168	0.305	0.289
Llama-2-13b	nominal	0.117	0.197	0.310	0.153	0.207	0.171
	held out	0.147	0.259	0.377	0.159	0.283	0.266
Llama-2-70b	nominal	0.113	0.173	0.266	0.149	0.159	0.144
	held out	0.147	0.203	0.322	0.149	0.271	0.219

[1]We only do this entities which do not make up the majority of the training data (e.g., as is the case with populated places for the world dataset and songs for the entertainment dataset) which is partially responsible for the discrepancies in the nominal cases for Tables 4.3 and 4.4.

[2]We note in this case the Spearman correlation is still high, suggesting this is an issue with bias generalization, as the different entity types are not uniformly distributed in time.

4.4.2 Dimensionality Reduction

Despite being linear, our probes still have d_{model} learnable parameters (ranging from 4096 to 8192 for the 7B to 70B models), enabling it to engage in substantial memorization. As a complementary form of evidence to the generalization experiments, we train probes with 2 to 3 orders of magnitude fewer parameters by projecting the activation datasets onto their k largest principal components.

Figure 4.4 illustrates the test R^2 for probes trained on each model and dataset over a range of k values, as compared to the performance of the full d_{model}-dimensional probe. We also report the test Spearman correlation in Figure C.8 which increases much more rapidly with increasing k than the R^2. Notably, the Spearman correlation only depends on the rank order of the predictions while R^2 also depends on their actual value. We view this gap as further evidence that the model explicitly represents space and time as these features must account for enough variance to be in the top dozen principal components, but that the probe requires more parameters to convert from the model's coordinate system to literal spatial coordinates or timestamps. We also observed that the first several principal components clustered the different entity types within the dataset, explaining why more than a few are needed.

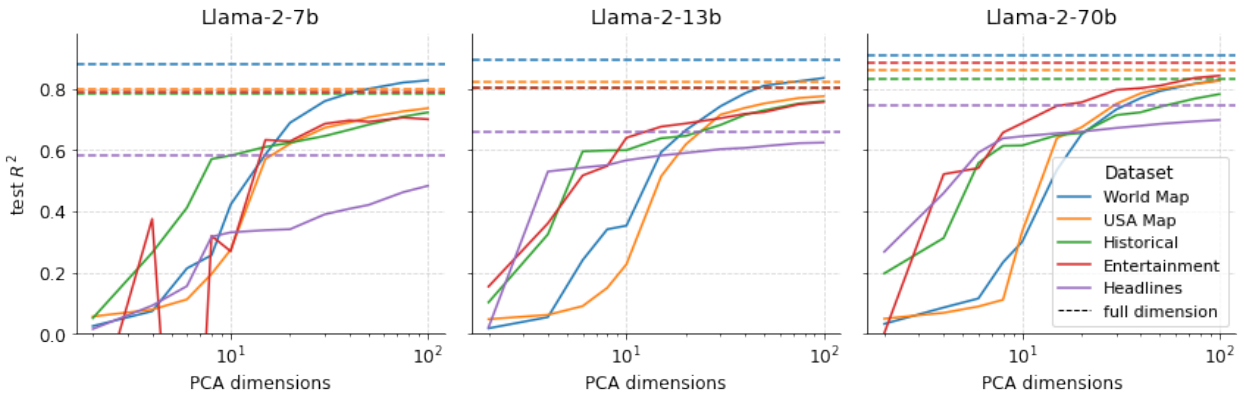


Figure 4.4: Test R^2 for probes trained on activations projected onto k largest principal components for each dataset and model compared to training on the full activations.

4.5 Space and Time Neurons

While the previous results are suggestive, none of our evidence directly shows that the model *uses* the features learned by the probe. To address this, we search for individual neurons with input or output weights that have high cosine similarity with the learned probe direction.

That is, we search for neurons which read from or write to a direction similar to the one learned by the probe.

We find that when we project the activation datasets on to the weights of the most similar neurons, these neurons are indeed highly sensitive to the true location of entities in space or time (see Figure 4.5). In other words, there exist individual neurons within the model that are themselves fairly predictive feature probes. Moreover, these neurons are sensitive to all of the entity types within our datasets, providing stronger evidence for the claim these representations are unified.

If probes trained with explicit supervision are an approximate upper bound on the extent to which a model represents these spatial and temporal features, then the performance of individual neurons is a lower bound. In particular, we generally expect features to be distributed in superposition [98], making individual neurons the wrong level of analysis. Nevertheless, the existence of these individual neurons, which received no supervision other than from next-token prediction, is very strong evidence that the model has learned and makes use of spatial and temporal features.

We also perform a series of neuron ablation and intervention experiments in Appendix C.2 to verify the importance of these neurons in spatial and temporal modeling.

4.6 Related Work

Linguistic Spatial Models Prior work has shown that natural language encodes geographic information [176, 177] and that relative coordinates can be approximately recovered with simple techniques like multidimensional scaling, co-occurrence statistics, or probing word embeddings [114, 154, 177, 194]. However, these studies only consider a few hundred well known cities and obtain fairly weak correlations. Most similar to our work is [168] who probe word embeddings and small language models for the coordinates of global cities and whether countries share a border, but conclude the amount of geographic information learned is "limited," likely because the largest model they study was 345M parameters (500x smaller than Llama 70B).

Neural World Models We consider a spatiotemporal model to be a necessary ingredient within a larger world model. The clearest evidence that such models are learnable from next-token prediction comes from GPT-style models trained on chess [266] and Othello games [164] which were shown to have explicit representations of the board and game state, with further work showing these representations are linear [203]. In true LLMs, [163] show that an entity's dynamic properties or relations can be linearly read out from representations

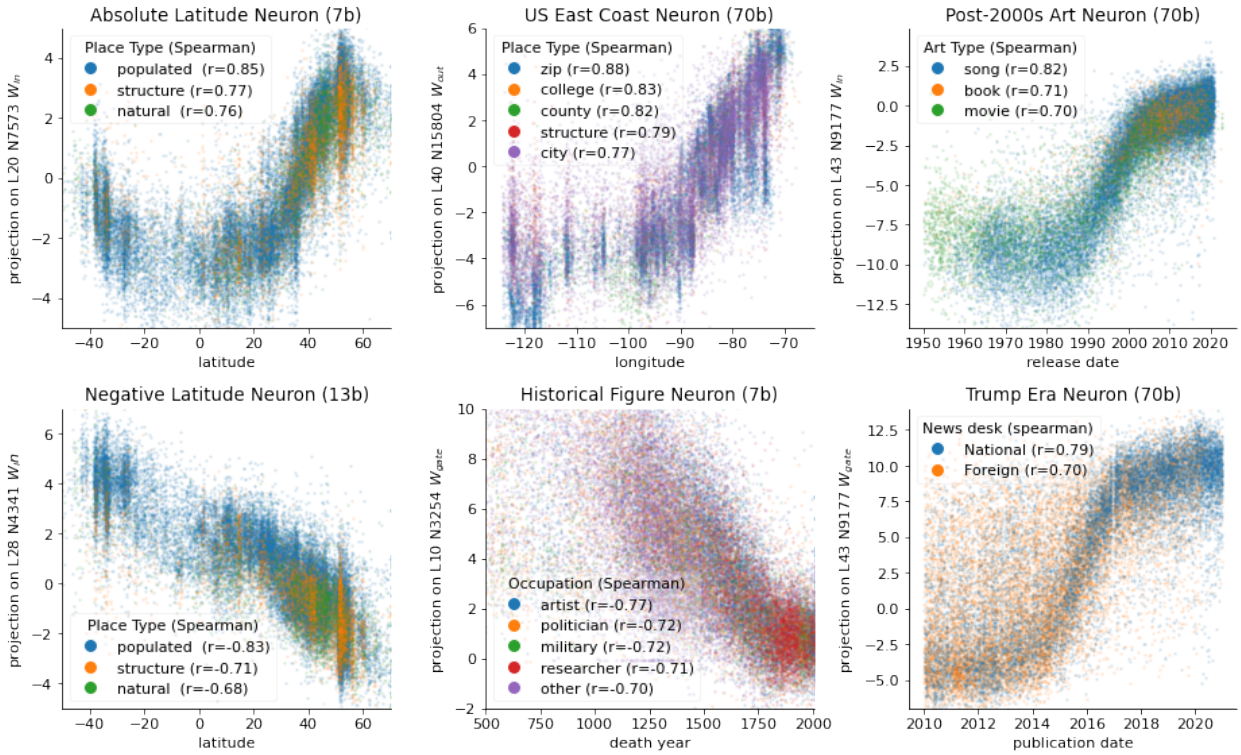


Figure 4.5: Space and time neurons in Llama-2 models. Depicts the result of projecting activation datasets onto neuron weights compared to true space or time coordinates with Spearman correlation by entity type.

at different points in the context. [1] and [221] show LLMs have representations that reflect perceptual and conceptual structure in color and spatial domains.

Factual Recall The point in time or space of an event or place is a particular kind of fact. Our investigation is informed by prior work on the mechanisms of factual recall in LLMs [109, 188, 189] indicating that early-to-mid MLP layers are responsible for outputting information about factual subjects, typically on the last token of the subject. Many of these works also show linear structure, for example in the factuality of a statement [56] or in the structure of subject-object relations [134]. To our knowledge, our work is unique in considering continuous facts.

Interpretability More broadly, our work draws upon many results and ideas from the interpretability literature [233], especially in topics related to probing [20], BERTology [239], the linearity hypothesis and superposition [98], and mechanistic interpretability [214]. More specific results related to our work include [125] who find a circuit implementing greater-than in the context of years, and [111] who find "region" neurons in multimodal models that resemble our space neurons.

4.7 Discussion

We have demonstrated that LLMs learn linear representations of space and time that are unified across entity types and fairly robust to prompting, and that there exists individual neurons that are highly sensitive to these features. We conjecture, but do not show, these basic primitives underlie a more comprehensive causal world model used for inference and prediction.

Our analysis raises many interesting questions for future work. While we showed that it is possible to linearly reconstruct a sample's absolute position in space or time, and that some neurons use these probe directions, the true extent and structure of spatial and temporal representations remain unclear. We conjecture that the most canonical form of this structure is a discretized hierarchical mesh, where any sample is represented as a linear combination of its nearest basis points at each level of granularity. Moreover, the model can and does use this coordinate system to represent absolute position using the correct linear combination of basis directions in the same way a linear probe would. We expect that as models scale, this mesh is enhanced with more basis points, more scales of granularity (e.g. neighborhoods in cities), and more accurate mapping of entities to model coordinates [193]. This suggests future work on extracting representations in the model's coordinate system rather than trying

to reconstruct human interpretable coordinates, perhaps with sparse autoencoders [77].

We also barely scratched the surface of understanding how these spatial and temporal models are learned, recalled, and used internally, or to what extent these representations exist within a more comprehensive world model. By looking across training checkpoints, it may be possible to localize a point in training when a model organizes constituent `is_in_place_X` features into a coherent geometry or else conclude this process is gradual [171]. We expect that the model components which construct these representations are similar or identical to those for factual recall [109, 188].

Finally, we note that the representation of space and time has received much more attention in biological neural networks than artificial ones [57, 250]. Place and grid cells [122, 208] in particular are among the most well-studied in the brain and may be a fruitful source of inspiration for future work on LLMs.

Acknowledgements

This chapter was based off of joint work [117] with Max Tegmark.

The authors would like to thank Sam Marks, Eric Michaud, Ziming Liu, Janice Yang, and especially Neel Nanda for their helpful discussions and feedback. W.G. was supported by Dimitris Bertsimas and an Open Philanthropy early career grant through the course of this work.

Chapter 5

Learning Sparse Nonlinear Dynamics via Mixed-Integer Optimization

5.1 Introduction

Advances in machine learning (ML) combined with the exponential growth of data and computing power are enabling new paradigms of data-driven science and engineering. In particular, emerging techniques for learning dynamic patterns directly from data are poised to revamp fields where data is abundant but traditional static analysis methods have failed to generate useful models. While accurate dynamical models are important, the ultimate goal is to advance scientific understanding by discovering interpretable models that are as simple as possible, but no simpler.

The modern era of data-driven system discovery started in earnest with the work of [46, 249] on symbolic regression. Since then, probabilistic methods [103, 219, 242, 286] and deep neural networks [15, 66, 179, 231, 232, 278, 285] have proven to be effective tools for modeling high-dimensional complex dynamical systems. However, it is the seminal work of [54] on the Sparse Identification of Nonlinear Dynamics (SINDy) framework that serves as the foundation for our approach. SINDy casts system identification as a sparse regression problem over a large set of nonlinear library functions to find the fewest active terms which accurately reconstruct the system dynamics. Such a technique is especially useful for finding highly interpretable models, and performs well even with limited training data (in particular, much less than what a neural network would require). Its success has inspired a large number of extensions and variants tailored for more specific problems [53, 101, 144, 145, 243].

The enabling technology underlying all of these methods is the optimization algorithm that selects and fits the best set of library terms to reconstruct the dynamics. The original

SINDy paper [54] used the sequential threshold least squares algorithm which finds a sparse solution by iteratively fitting a least squares regression on the candidate library and removing terms with coefficients below a specified threshold. However, thresholding is problematic for recovering small model coefficients and does not easily allow adding additional structure on the coefficients as might be needed for enforcing an arbitrary physical constraint. These limitations motivated the development of algorithms with different sparse regularizers that could incorporate constraints such as the sparse relaxed regularized regression (SR3) [67, 288] and Conditional Gradients based approaches [61]. There is also a long history of greedy algorithms and convex relaxations designed to solve sparse regression and related subset selection problem [222, 263, 264]. For system identification problems, we claim that such relaxations and heuristics are unnecessary, and ultimately, inadequate.

Concurrent with the developments in system identification and increase in scientific data, advances in hardware and software have led to over a one trillion times speed up in mixed-integer optimization (MIO) solvers since 1990 [2, 41, 157]. This fact necessitates researchers revisit their preconceptions about the tractability of MIO in machine learning contexts. Indeed, there has been substantial work on viewing the full slate of classical machine learning algorithms under a modern optimization lens [29], often yielding state-of-the-art results with practical computational budgets. Most relevant to system identification is the recent progress on high-dimensional sparse regression which solves the NP-hard feature selection problem exactly [28, 31, 33–35, 128, 264]. In addition to superior performance, these modern formulations inherit the full generality of MIO, empowering domain experts with a rich modeling language to express a vast range of model desiderata as arbitrary linear, quadratic, and semidefinite constraints on both the coefficients and sparsity structure.

The objective of this work is to bridge the gap between the system identification and discrete optimization literatures and demonstrate the effectiveness of learning sparse nonlinear dynamics via MIO-SINDy. We begin by reviewing MIO for sparse regression and then adapt a formulation utilizing specially ordered sets to the basic SINDy framework and its relevant extensions. We then systematically illustrate the contrast in performance between heuristic and MIO sparse regression (MIOSR) methods on a wide range of canonical dynamical systems, including both ordinary and scalar partial differential equations. Our main contribution is the optimal MIO-SINDy formulation utilizing a MIOSR optimizer for which we establish the following results:

1. **Tractable and provably optimal.** MIOSR terminates when the objective of the incumbent solution matches the dual lower bound. That is, when the gap between the objective upper bound and lower bound vanishes, yielding both an optimal solution and a proof of optimality. Despite solving the NP-hard subset selection problem exactly,

in Section 5.3.3 we show that the extra computational cost of MIOSR is minimal, and scales favorably with additional data and compute.

2. **Sample efficient and noise robust.** This theoretical optimality buys practical performance in more challenging statistical regimes. In particular, we study the low data limit in Section 5.3.2 and the high noise setting in Section 5.3.5 where we find MIOSR outperforms heuristic methods, especially in learning the true sparse form of the dynamics.

3. **Customizable.** Due to the flexibility of MIO as a modeling framework, MIOSR can be endlessly customized to impose additional structure on the learning problem to further improve sample efficiency, enforce physically realistic models, or incorporate other domain tailored model requirements. We discuss this flexibility and a few relevant extensions in Section 5.2.3 and then demonstrate the benefits of incorporating known physics as constraints in Section 5.3.4.

4. **Consistent interface.** We provide an implementation of our algorithm which adheres to the PySINDy interface [83, 147], both computationally and conceptually. Therefore, our algorithm is compatible with other advancements in the SINDy framework (e.g., preprocessing, library construction, outer loop algorithms) and seamlessly integrates into existing tools and workflows.

We conclude this chapter with a case study on Large Language Models (LLMs) that (1) further demonstrates the increased sample efficiency and robustness of over heuristic baselines and (2) provides additional evidence that LLMs possess important world modeling primitives.

5.2 Methods

We begin by reviewing the SINDy problem, its extension to partial differential equations (PDEs), and the weak form of SINDy for noisy data. We then discuss the most appropriate MIO formulations for sparse regression that solve the SINDy problem and some relevant extensions.

5.2.1 SINDy

The original SINDy framework [54] was designed to recover systems of ordinary differential equations (ODEs) of the form

$$\frac{d}{dt}x(t) = f(x(t)) \tag{5.1}$$

where $x(t) \in \mathbb{R}^d$ is the state of the system at time t, and the function $f(x(t))$ encodes the dynamics of the system. Implicitly, $f(\cdot)$ is also assumed to be sparse. This is justified because most physical systems are known to have sparse dynamics when represented in suitable coordinates. Sparsity also acts as a natural and effective regularizer [158] while yielding more interpretable models.

Given n measurements of a system of interest, the three required inputs are a state-time matrix $\mathbf{X} \in \mathbb{R}^{n\times d}$ where x_{ij} is the state of system variable j at time i, the measured or numerically approximated time derivatives of the state variables $\dot{\mathbf{X}} \in \mathbb{R}^{n\times d}$, and a candidate library of nonlinear functions $\boldsymbol{\Theta}(\boldsymbol{X}) = [\boldsymbol{\theta}_1(\boldsymbol{X}), \ldots, \boldsymbol{\theta}_L(\boldsymbol{X})] \in \mathbb{R}^{n\times D}$. As an example, we could consider second order polynomials $\boldsymbol{\theta}_\ell(\boldsymbol{X}) = \boldsymbol{X}^2$ which would yield $d(d-1)/2$ candidate terms of elementwise products for every pair $\boldsymbol{X}_i \odot \boldsymbol{X}_j$. In all of our experiments, we will use a library of low order polynomials, but other natural choices are trigonometric, logarithmic, or exponential functions.

With these ingredients, we seek a solution to

$$\dot{\boldsymbol{X}} = \boldsymbol{\Theta}(\boldsymbol{X})\boldsymbol{\Xi} \tag{5.2}$$

for $\boldsymbol{\Xi} = \begin{bmatrix} \boldsymbol{\xi}^{(1)} & \boldsymbol{\xi}^{(2)} & \ldots & \boldsymbol{\xi}^{(d)} \end{bmatrix} \in \mathbb{R}^{D\times d}$ to learn the dynamics of each state variable $\dot{\boldsymbol{X}}_i = \boldsymbol{f}_i(\boldsymbol{X}) = \boldsymbol{\Theta}\left(\boldsymbol{X}\right)\boldsymbol{\xi}^{(i)}$. Framed as an optimization problem, the standard objective is to

$$\min_{\boldsymbol{\Xi}} \|\dot{\boldsymbol{X}} - \boldsymbol{\Theta}(\boldsymbol{X})\boldsymbol{\Xi}\|^2 + \lambda R(\boldsymbol{\Xi}) \tag{5.3}$$

where $R(\cdot)$ is a sparsity promoting regularization function which may also include an l_2 ridge regularization term to improve the conditioning and add robustness [27].

Given the rapid growth of the number of library functions with regard to the input data dimensionality (e.g., $D = O(d^p)$ for an order p polynomial library), SINDy is best suited for analysis of low-dimensional data sets. Therefore, to learn the dynamics of high dimensional systems, a dimensionality reduction technique such as proper orthogonal decomposition (POD) is first applied to the data [26], and then SINDy is applied to the reduced data space. SINDy also makes the implicit assumptions that the data contain the relevant governing variables, are represented in coordinates which allow for sparsely representing the

dynamics as the sum of only a few elementary functions, and that the library contains these elementary functions.

SINDy for PDEs This core framework can be further extended to the automatic discovery of PDEs by including partial derivatives in the candidate library [243, 246]. Concretely, spatiotemporal data of m spatial locations measured over n time slices is arranged into a length mn column vector $\mathbf{U}$. Then a candidate library $\mathbf{\Theta}(\mathbf{U}) \in \mathbb{R}^{mn \times D}$ is constructed as before except here we consider functions of both the system state and system spatial derivatives (which have to be numerically approximated). That is, in addition to library functions like $\boldsymbol{U}^2$, we also might include $\boldsymbol{U}\boldsymbol{U}_x$, and potentially higher order derivatives like $\boldsymbol{U}_{xxx}$. Finally, as before, we seek coefficients $\mathbf{\Xi}$ which accurately reconstruct the temporal dynamics $\boldsymbol{U}_t = \mathbf{\Theta}(\boldsymbol{U})\mathbf{\Xi}$.

Weak Form One core drawback with SINDy, especially when applied to PDEs or noisy data, is the need to numerically estimate derivatives because numerical differentiation compounds any noise present in the underlying measurement data. When differentiating multiple times, as is necessary for higher order PDEs, the estimates can become unusable, even while using more robust differentiation techniques like smoothed finite difference or polynomial interpolation.

This drawback motivates the weak form of SINDy [191, 235, 247] (which also generalizes to the PDE case [115, 192]), where both sides of Equation 5.2 are integrated over a random collection of K temporal subdomains (spatiotemporal for PDEs). That is, for a random subdomain Ω_k, a candidate library function $\boldsymbol{\theta}_i$, and a weight vector $\boldsymbol{w}$, we compute

$$q_i^k = \int_{\Omega_k} \boldsymbol{w}^T \boldsymbol{\theta}_i d\Omega \tag{5.4}$$

for every library term and subdomain. Each of the elements are then organized into a data matrix $\mathbf{Q} \in \mathbb{R}^{K \times D}$. By also integrating the left hand side of Equation 5.1 over the same subdomains, we get a linear system $\mathbf{q_0} = \mathbf{Q\Xi}$ amenable to sparse regression without needing to differentiate noisy data.

5.2.2 Mixed-Integer Sparse Regression

The seminal paper on best subset selection using MIO [33] proposed two primal formulations for sparse regression. Both of these formulations use binary variables to encode the support of the coefficients, one using big-M constraints, and the other using type-1 specially ordered sets (SOS-1). Because the system identification problem is coordinate separable, we can

fit each dimension independently, resulting in d smaller subproblems which can be solved directly using these techniques.

For system dimension j, with user-specified target sparsity k_j, the big-M formulation with ridge regularization for the SINDy problem is

$$\min_{\boldsymbol{\xi},\boldsymbol{z}} \quad \|\dot{\boldsymbol{X}}_j - \boldsymbol{\Theta}(\boldsymbol{X})\boldsymbol{\xi}\|_2^2 + \lambda\|\boldsymbol{\xi}\|_2^2 \tag{5.5}$$

$$\text{s.t.} \quad M_i^\ell z_i \le \xi_i \le M_i^U z_i \qquad i = 1, \dots D \tag{5.6}$$

$$\sum_{i=1}^{D} z_i \le k_j \tag{5.7}$$

$$\xi_i \in \mathbb{R}, \quad z_i \in \{0,1\} \qquad i = 1, \dots, D \tag{5.8}$$

where M_i^ℓ, M_i^U are lower and upper bounds on the coefficients. This formulation is solved for each $j \in [1, d]$, where we simply stack the coefficient vectors to recover the full system dynamics $\boldsymbol{\Xi} = \begin{bmatrix} \boldsymbol{\xi}^{(1)} & \boldsymbol{\xi}^{(2)} & \dots & \boldsymbol{\xi}^{(d)} \end{bmatrix}$.

While the theory and practice of solving MIO problems is deep [32], the basic solution technique relies on the linear-programming (LP) based branch-and-bound algorithm to avoid performing the full combinatorial search. The hard part of MIO problems are the discrete variables which make the problem non-convex and NP-hard. Hence, branch-and-bound relies on solving the polynomial-time linear relaxation of the problem by allowing integer variables to take continuous values, to obtain bounds on the optimal integral solution. Branch-and-bound maintains a tree of solutions to the LP relaxation where, at each branch of the tree, an integer variable is fixed to an integer value, while maintaining global upper and lower bounds on the optimal objective value of an integral solution. The algorithm continues expanding the tree, by branching on integer variables with fractional LP optimal values in promising partial solutions, while pruning other branches outside the solution bounds, until the lower and upper bounds converge, yielding an optimal integral solution. We rely on modern optimization solvers such as Gurobi [120] or CPLEX [76] to both solve the problem and present this certificate of optimality.

The effectiveness of big-M modeling relies on the tightness of coefficient bounds as otherwise the linear relaxations are too weak to efficiently prune the branch-and-bound tree. [33] derive a number of ways to obtain such bounds, however these approaches can add significant overhead to the solution times, and don't generalize well in the presence of arbitrary constraints. This motivates a nonlinear approach which circumvents the need to calculate these $2D$ different bounds. [33] also proposed adding the cardinality constraint via type-1 specially ordered sets (SOS-1) [32]. An SOS-1 constraint on a set of variables enforces that

no more than one variable within the set is nonzero enabling branching on multiple variables for each branch of the branch-and-bound tree. In this case

$$(1 - z_i)\xi_i = 0 \iff \{\xi_i, 1 - z_i\} : \text{SOS-1}$$

correctly captures the support of ξ. By replacing the support constraint, we get

$$\min_{\boldsymbol{\xi}, \boldsymbol{z}} \quad \|\dot{\boldsymbol{X}}_j - \boldsymbol{\Theta}(\boldsymbol{X})\boldsymbol{\xi}\|_2^2 + \lambda\|\boldsymbol{\xi}\|_2^2 \tag{5.9}$$

$$\text{s.t.} \quad \{\xi_i, 1 - z_i\} : \text{SOS-1} \qquad i = 1, \dots D \tag{5.10}$$

$$\sum_{i=1}^{D} z_i \leq k_j \tag{5.11}$$

$$\xi_i \in \mathbb{R}, \quad z_i \in \{0, 1\} \qquad i = 1, \dots, D. \tag{5.12}$$

More explicitly, the main objective term is

$$\|\dot{\boldsymbol{X}}_j - \boldsymbol{\Theta}(\boldsymbol{X})\boldsymbol{\xi}\|_2^2 \tag{5.13}$$

$$= \boldsymbol{\xi}^T \boldsymbol{\Theta}(\boldsymbol{X})^T \boldsymbol{\Theta}(\boldsymbol{X})\boldsymbol{\xi} - 2\langle \boldsymbol{\Theta}(\boldsymbol{X})^T \dot{\boldsymbol{X}}_j, \boldsymbol{\xi}\rangle + \dot{\boldsymbol{X}}_j{}^T \dot{\boldsymbol{X}}_j. \tag{5.14}$$

If we remove the constant term $\dot{\boldsymbol{X}}_j{}^T \dot{\boldsymbol{X}}_j$ and add the regularization term we get our final objective

$$\boldsymbol{\xi}^T \boldsymbol{\Theta}(\boldsymbol{X})^T \boldsymbol{\Theta}(\boldsymbol{X})\boldsymbol{\xi} - 2\langle \boldsymbol{\Theta}(\boldsymbol{X})^T \dot{\boldsymbol{X}}_j,\ \boldsymbol{\xi}\rangle + \lambda \boldsymbol{\xi}^T \boldsymbol{\xi} \tag{5.15}$$

$$= \boldsymbol{\xi}^T \left(\boldsymbol{\Theta}(\boldsymbol{X})^T \boldsymbol{\Theta}(\boldsymbol{X}) + \lambda \boldsymbol{I}\right) \boldsymbol{\xi} - 2\langle \boldsymbol{\Theta}(\boldsymbol{X})^T \dot{\boldsymbol{X}}_j,\ \boldsymbol{\xi}\rangle. \tag{5.16}$$

For a problem with n temporal observations and library size D, our final formulation has D continuous variables, D binary variables, D corresponding SOS-1 constraints, and one knapsack constraint. The objective has $\frac{D(D+1)}{2}$ quadratic terms and D linear terms. Notably, the formulation size is independent of n which yields very favorable scaling properties as we will discuss in Section 5.3.3.

A question that naturally arises is whether to enforce sparsity as a hard constraint or promote sparsity as an objective penalty. While it may be tempting to use a penalty term and let the model balance sparsity, it is known that constrained problems enjoy more favorable statistical properties [255]. In particular, while an optimal solution to the sparsity regularized problem is always obtainable by the constrained problem, the converse is not true in general (see [156] Section 2.2). This is especially true when the data matrix exhibits high multicollinearity which is common in system discovery because the library terms are

usually strongly correlated.

Finally, we note that we adopt the SOS-1 formulation because we found it to be the most numerically stable and most flexible in including other potential model desiderata (e.g., satisfaction of physical constraints). However, it is not the most scalable. Modern general purpose optimization solvers can only handle sparse regression problems with up to a few thousand SOS-1 constraints. This is in stark contrast to very recent tailored sparse regression solution techniques such as the outer approximation method [31], coordinate-descent based branch-and-bound [128], and the backbone method [28] which can scale to the high-dimensional regime with dimension $O(10^7)$. Given that even a six-dimensional system with a fifth-order polynomial library is only of dimension 462, the more stable and flexible general purpose solvers are preferable for our circumstances.

5.2.3 Extensions

In many physical systems where something is known about the underlying physics, we can incorporate this knowledge as constraints on the model coefficients [67, 146, 173]. However, these constraints generally apply to the system as a whole (e.g., conservation of energy), so it is no longer possible to fit one coordinate at a time. Therefore, we fit all coordinates jointly using objective

$$\min_{\boldsymbol{\xi}} \left\| \begin{bmatrix} \dot{\boldsymbol{X}}_1 \\ \dot{\boldsymbol{X}}_2 \\ \vdots \\ \dot{\boldsymbol{X}}_d \end{bmatrix} - \begin{bmatrix} \boldsymbol{\Theta}(\boldsymbol{X}) & 0 & \dots & 0 \\ 0 & \boldsymbol{\Theta}(\boldsymbol{X}) & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \dots & \boldsymbol{\Theta}(\boldsymbol{X}) \end{bmatrix} \begin{bmatrix} \boldsymbol{\xi}^{(1)} \\ \boldsymbol{\xi}^{(2)} \\ \vdots \\ \boldsymbol{\xi}^{(d)} \end{bmatrix} \right\|_2^2 \tag{5.17}$$

$$= \min_{\boldsymbol{\xi}} \sum_{i=1}^{d} \boldsymbol{\xi}^{(i)T} \boldsymbol{\Theta}(\boldsymbol{X})^T \boldsymbol{\Theta}(\boldsymbol{X}) \boldsymbol{\xi}^{(i)} - 2\langle \boldsymbol{\Theta}(\boldsymbol{X})^T \dot{\boldsymbol{X}}_i, \boldsymbol{\xi}^{(i)} \rangle. \tag{5.18}$$

Now in addition to the sparsity constraint, we can add arbitrary constraints $\boldsymbol{A}\bar{\boldsymbol{\xi}} \leq \boldsymbol{b}$ where $\bar{\boldsymbol{\xi}}$ is the vectorized coefficient matrix of length Dd. Of course, because we inherit the full generality of MIO, these constraints can be anything that modern optimization solvers can handle (e.g., linear, quadratic, semidefinite, equality, inequality).

While such a formulation increases the dimension of the regression problem by a factor of d (which is potentially quite costly, both in terms of computational time and sample efficiency), it is sometimes more natural and offers several additional benefits. The first is based on the fact that the coordinates used in SINDy often represent the spatial modes of a high-dimensional discretized PDE simulation computed using a dimensionality reduction

technique like proper orthogonal decomposition [69]. Each of these spatial modes have an associated energy λ_i designated by their singular values. Consequently, the quality of the high-dimensional reconstruction does not depend uniformly on the accuracy of each of the individual spatial modes, but in proportion to their energies. In this setting, we can weight each inner term in the sum of the objective function (5.17) by the energy of the respective spatial mode to recover the terms which maximize the quality of the full reconstruction, rather than the average dimension-wise reconstruction. Another advantage is instead of having to run parameter tuning on different values of k for each system variable, we can specify and cross-validate one global value of the desired sparsity, and the model will automatically determine the correct level of sparsity per dimension.

While not pursued here, there are many extensions that could be appropriate for more specific circumstances. For instance, it is possible to add lower and upper bounds on the magnitudes of coefficients, add more sophisticated conditional logic on the relationship between nonzero coefficients, put controls on the level of multicollinearity [31], require coefficients be statistically significant [31], automatically prune outliers [67, 262], and much more. In short, as such a general framework, MIO-SINDy fully empowers the researcher to express a vast range of model desiderata and impose additional structure to aid the learning process.

5.3 Results

We benchmark our approach on nine canonical dynamical systems across a wide variety of statistical regimes. Our analysis focuses on attributes that most differentiate SINDy from alternative techniques, and then illustrates how using optimal methods furthers these advantages. In particular, we study sample efficiency, robustness, constraint enforcement, and computational efficiency where our evaluation focuses on identifying correct sparse models.

All of our experiments are built off of the open source PySINDy library [83, 147]. We use a shared university cluster with heterogeneous hardware where individual trials are confined to one CPU core with sufficient RAM. Our implementation of MIO sparse regression utilizes Gurobi 9.5.0 [120] to optimize and prove optimality. We make all of our code, data, and results publicly available at our Github repository https://github.com/wesg52/sindy_mio_paper.

5.3.1 Experimental Overview

Most of our experiments follow the same high level structure. We vary a quantity of interest (e.g., data length, data noise) for 50 random initial conditions, each with additive Gaussian

noise scaled to be a certain percentage of the l_2 norm of the training data. Then for each sampled trajectory, we split the data into a training and a validation segment, using the validation segment to select the hyperparameters (see Section 5.3.1) of each of the baseline algorithms (Section 5.3.1). We follow the standard practice of unbiasing the final model by refitting an unregularized least squares regression on the selected coefficients. This final model is then evaluated on a suite of metrics (Section 5.3.1), with a focus towards identifying the correct coefficient support.

While the specifics of our results are somewhat sensitive to the details of our experimental procedure, we take steps to ensure our conclusions are robust to such design choices. For instance, we use random initial conditions whereas many papers in the SINDy literature report results using a fixed initial condition. While this aids reproducibility, we found performance to be sensitive to initial conditions for many systems, especially in the low data limit. Additionally, for each experiment we test our approach on multiple systems, each of which raises different qualitative behavior, to better understand the factors which differentiate performance.

Model Selection

A critical component of learning parsimonious models is in choosing hyperparameters that appropriately tradeoff model fit with sparsity, since adding more degrees of freedom monotonically decreases insample error. We strike this balance by selecting parameters which minimize the Akaike information criterion (AIC) metric [4, 181].

In our setting of sparse regression, for a learned model $\hat{\boldsymbol{\Xi}}$, the corrected AIC is given by

$$AIC_c = m \ln(RSS/m) + 2k + \frac{2(k+1)(k+2)}{m-k-2} \tag{5.19}$$

where RSS is the residual sum of squared errors
$\sum_{i=1}^{n}\sum_{j=1}^{d}(\dot{\boldsymbol{X}} - \boldsymbol{\Theta}(\boldsymbol{X})\hat{\boldsymbol{\Xi}})_{ij}^2$, $m = n \times d$ is the total number of measurements, k is the sparsity of the solution, and the last term is the correction for finite samples.

Unless otherwise noted, for every trial of every experiment discussed below, we run the following model selection procedure. Split the sampled trajectory into a train and a validation interval, typically the first 2/3 and the last 1/3 respectively. For each algorithm and choice of hyperparameters, train on the training split and compute the AIC_c with respect to the validation data. The final model is the one which minimizes the AIC_c metric on the validation data. Note, for algorithms which fit each dimension separately, we compute the AIC_c dimensionwise and combine the best coefficients per dimension to create the final model.

Algorithms

Given our focus on accurate support recovery, and for sake of consistent comparison, we restrict our baselines to other l_0 regularized or constrained sparse regression algorithms. This notably excludes l_1 regularized sparse regression methods like LASSO and its variants [61, 263], probabilistic methods, and deep learning based approaches which are less suitable for accurate variable selection [31, 67]. Specifically, we compare our MIO based approach (MIOSR) to four common optimizers in the SINDy literature: sequential threshold least squares (STLSQ) [54], sparse relaxed regularized regression (SR3) [67], stepwise sparse regression (SSR) [47], and ensembling using STLSQ (E-STLSQ) [101].

For every experiment trial, we perform a grid search over the relevant hyperparameters of each algorithm to find the set which minimize the AIC_c as described above. Each algorithm has a hyperparameter corresponding to regularization strength and sparsity promotion. MIOSR, STLSQ, SSR, and E-STLSQ all have an explicit ridge regression penalty while SR3 has a relaxation parameter ν. For MIOSR and SSR, we tune the sparsity of each dimension, where we control the can control k exactly. For (E-)STLSQ and SSR, we tune the threshold parameter which acts as a proxy for the true sparsity. Parameter ranges for each algorithm are included in the appendix.

For SSR, we use the greedy criterion of removing the smallest magnitude coefficient at each iteration. For SR3, we run until convergence up to a max 10000 iterations. For MIOSR, we set a timeout of 30 seconds per dimension to prove convergence. For E-STLSQ we use robust bagging (bragging) where we randomly sample time slices with replacement to train 50 different models, and then use the median of the coefficients as the final model.

Evaluation Metrics

To evaluate each algorithm, we focus on three common metrics: true positivity rate (TPR), normalized coefficient error (NCE), and root mean squared error (RMSE) of the estimated dynamics. Throughout, let $\boldsymbol{\Xi}$ be the true dynamics of the system and $\hat{\boldsymbol{\Xi}}$ be the estimated dynamics.

The true positivity rate measures the ability to identify the correct nonzero terms of the dynamics, and importantly, to not select superfluous terms. Specifically, the true positivity rate is the ratio of intersection over union of nonzero coefficients

$$\text{TPR} = \frac{|\{i : \Xi_i \neq 0\} \cap \{i : \hat{\Xi}_i \neq 0\}|}{|\{i : \Xi_i \neq 0\} \cup \{i : \hat{\Xi}_i \neq 0\}|}, \tag{5.20}$$

or equivalently, the ratio of true coefficients identified to the combined number of true coef-

ficients, false zero coefficients, and false nonzero coefficients identified.

The normalized coefficient error simply measures the normalized euclidean distance between the true coefficients and the learned coefficients

$$\text{NCE} = \frac{\|\boldsymbol{\Xi} - \hat{\boldsymbol{\Xi}}\|_2}{\|\boldsymbol{\Xi}\|_2}. \tag{5.21}$$

This metric is less punitive that the true positivity rate because it mainly captures the difference in large coefficients, and minimally penalizes small nonzero terms.

Finally, to contextualize how the coefficient error actually impacts the quality of the recovered model, we also report the root mean squared error of the estimated derivatives on a clean test set. That is, for a testing trajectory of length n we calculate

$$\text{RMSE} = \sqrt{\frac{1}{nd}\sum_{i=1}^{n}\sum_{j=1}^{d}(\Theta(\boldsymbol{X}_{test})\boldsymbol{\Xi} - \Theta(\boldsymbol{X}_{test})\hat{\boldsymbol{\Xi}})_{ij}^2}. \tag{5.22}$$

To test that the model truly generalizes, we independently sample 10 initial conditions, each run for 10 seconds, and take the average RMSE across each of these trajectories. Due to the chaotic nature of the systems we study, we only compare the derivatives and not the trajectories of the forward models, since initial errors will rapidly compound regardless of the accuracy of the underlying model.

In the results that follow, we will typically report the log RMSE and log l_2 coefficient errors, averaged over 50 trials. When calculating the mean and standard errors, we do so on the log of these statistics. This avoids one outlier from dragging up the mean by many orders of magnitude, and makes the standard errors symmetric in log space.

5.3.2 Sample Efficiency

A key advantage of SINDy over deep learning techniques is the reduced data requirements to learn high quality sparse models. To test how MIOSR extends this advantage, we study the effect of varying the length of the training trajectory on model recovery for three canonical dynamical systems studied in the SINDy literature: the Lorenz model [174], the Hopf system [183], and a triadic magnetohydrodynamical (MHD) model [60].

The Lorenz model is the classic example of a chaotic system and is given by

$$\dot{x} = \sigma(y - x) \tag{5.23a}$$
$$\dot{y} = x(\rho - z) - y \tag{5.23b}$$
$$\dot{z} = xy - \beta z \tag{5.23c}$$

where we use the standard parameters $\sigma = 10,\ \beta = 8/3,\ \rho = 28$. Another canonical system in the study of nonlinear dynamics is the Hopf system given by

$$\dot{x} = \mu x + \omega y - Ax\left(x^2 + y^2\right) \tag{5.24a}$$
$$\dot{y} = -\omega x + \mu y - Ay\left(x^2 + y^2\right) \tag{5.24b}$$

where we set $\mu = -0.05,\ \omega = A = 1$. Finally, we consider a simplified plasma model of a joint velocity and magnetic field with 0 fluid viscosity or resistivity

$$\dot{V}_1 = 4\left(V_2V_3 - B_2B_3\right) \tag{5.25a}$$
$$\dot{V}_2 = -7\left(V_1V_3 - B_1B_3\right) \tag{5.25b}$$
$$\dot{V}_3 = 3\left(V_1V_2 - B_1B_2\right) \tag{5.25c}$$
$$\dot{B}_1 = 2\left(B_3V_2 - V_3B_2\right) \tag{5.25d}$$
$$\dot{B}_2 = 5\left(V_3B_1 - B_3V_1\right) \tag{5.25e}$$
$$\dot{B}_3 = 9\left(V_1B_2 - B_1V_2\right). \tag{5.25f}$$

Figure 5.1 depicts our results for each combination of algorithm and system for varying lengths of training data. For each system we sample trajectories with 0.002 second time granularity with 0.2% added Gaussian noise. In anticipating the assumption that exact optimization methods are not scalable, we use oversized libraries: 5th order polynomials for the Lorenz and Hopf systems and 3rd order polynomials for MHD. This yields libraries with dimension 56, 21, and 84 respectively.

The high level conclusion is that MIOSR consistently out performs all other methods by finding more accurate models with less data, and with less variance in the quality of the fit. While the gap is more muted in the Lorenz case, the stark differences in the Hopf and MHD systems surface two distinct scenarios where heuristics can fail: the presence of small coefficients and large libraries. For Hopf, with bifurcation parameter $\mu = -0.05$, thresholding techniques cannot select such a small coefficient in the presence of even modest noise. Indeed, the original SINDy paper used multiple independent trajectories to learn the

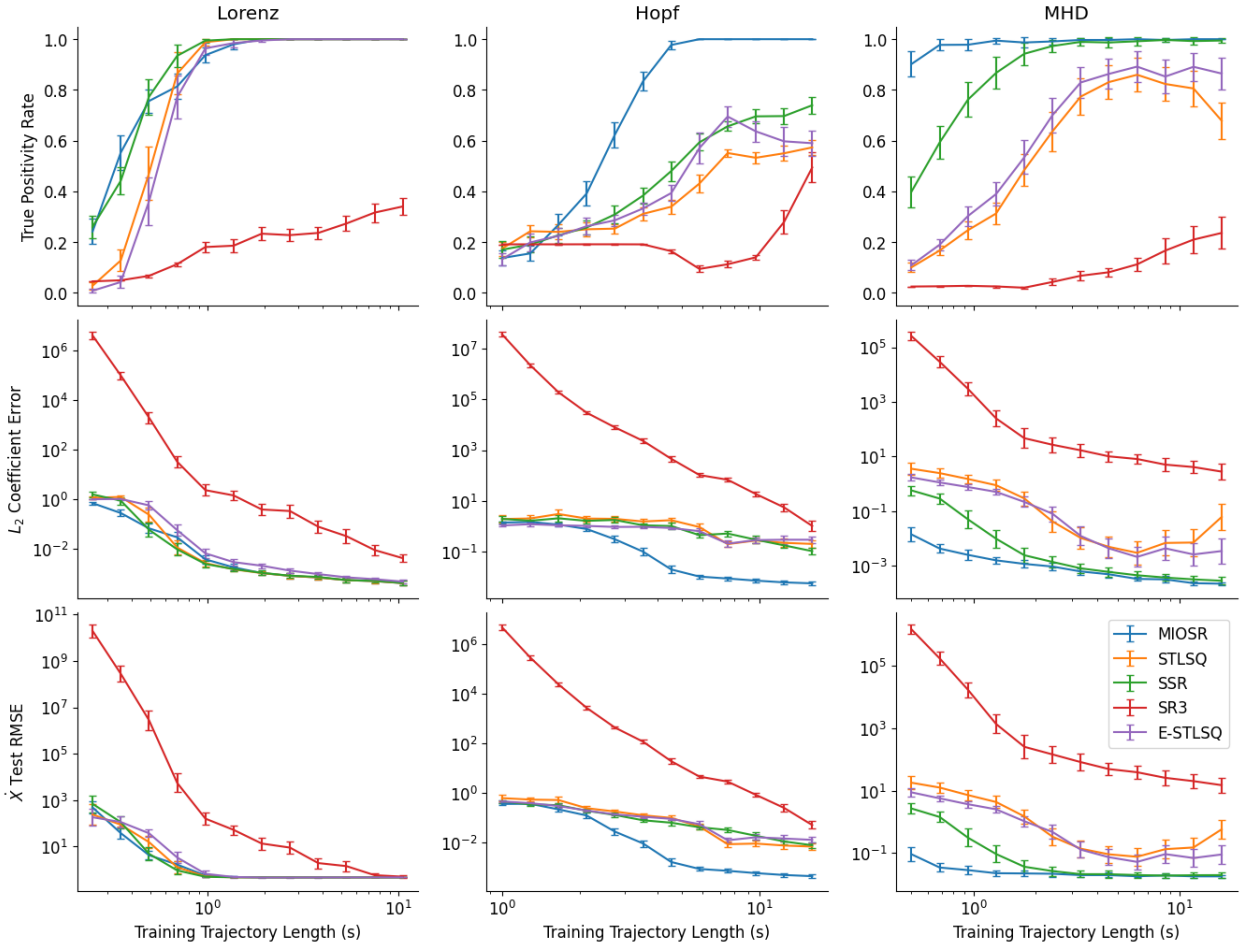


Figure 5.1: Performance comparison of sparse regression algorithms for the differential form of SINDy under varying amounts of training data for three different canonical systems: Lorenz, Hopf, and MHD. Results are averaged over 50 trials with added Gaussian noise of 0.2%.

Hopf dynamics, because the system quickly converges to a fixed point. In the low data limit for MHD, a larger 6 dimensional system, there are many combinations of the large library of terms which fit the small training set well. With so many degrees of freedom, iterative methods break down by taking incorrect intermediate steps, but by taking a global view, MIOSR can identify the true model.

In this low data regime, some baselines completely fail, in particular SR3. This is perhaps unsurprising because SR3 fits all coordinates jointly and therefore is solving a more difficult, higher dimensional optimization problem. While sometimes the baselines methods achieve achieve comparable test RMSE, they often do so by overfitting as evidenced by the low true positivity rate. This largely defeats the purpose of SINDy in discovering robust, interpretable, and scientifically illuminating models.

Many of the aforementioned difficulties are partially ameliorated by using a smaller library. In the appendix, we perform the same experiment with "tight" libraries, those which don't include higher order polynomials than are necessary to express the system (Figure D.1). While MIOSR maintains a clear edge, the difference is less striking. Of course for novel systems, this information is not available *a priori*, and therefore this represents the most idealized scenario.

5.3.3 Computational Efficiency

Nearly every SINDy paper contains a sentence that justifies the need for sparse regression heuristics by claiming that the feature selection problem is computationally intractable due to the combinatorial nature. To directly dispel this claim, we compare the wallclock computation time of each of the different algorithms for varying library sizes and amounts of training data (see Figure 5.2). Similar to the previous experiment, for each system, library size, and amount of data, we train each algorithm on 50 random trajectories with 0.2% additive Gaussian noise with 0.002 sample frequency. We precompute the derivatives and library, so the reported times corresponds to just the regression time, not the whole SINDy pipeline. Unlike the previous experiment, we are not performing hyperparameter tuning, and instead use appropriate defaults learned above.

Several points deserve elaboration. Perhaps most surprising to those unfamiliar with MIO based machine learning, is that MIOSR is often faster as the amount of data increases, sometimes significantly [34]. This is partially due to the fact that the final optimization problem has no dependence on n, as we only require a one time $n \times D$ matrix multiplication to initially construct the objective value coefficients. On the other hand, for small n, there are more ways to fit the data, so the bounds in the branch-and-bound tree are weaker,

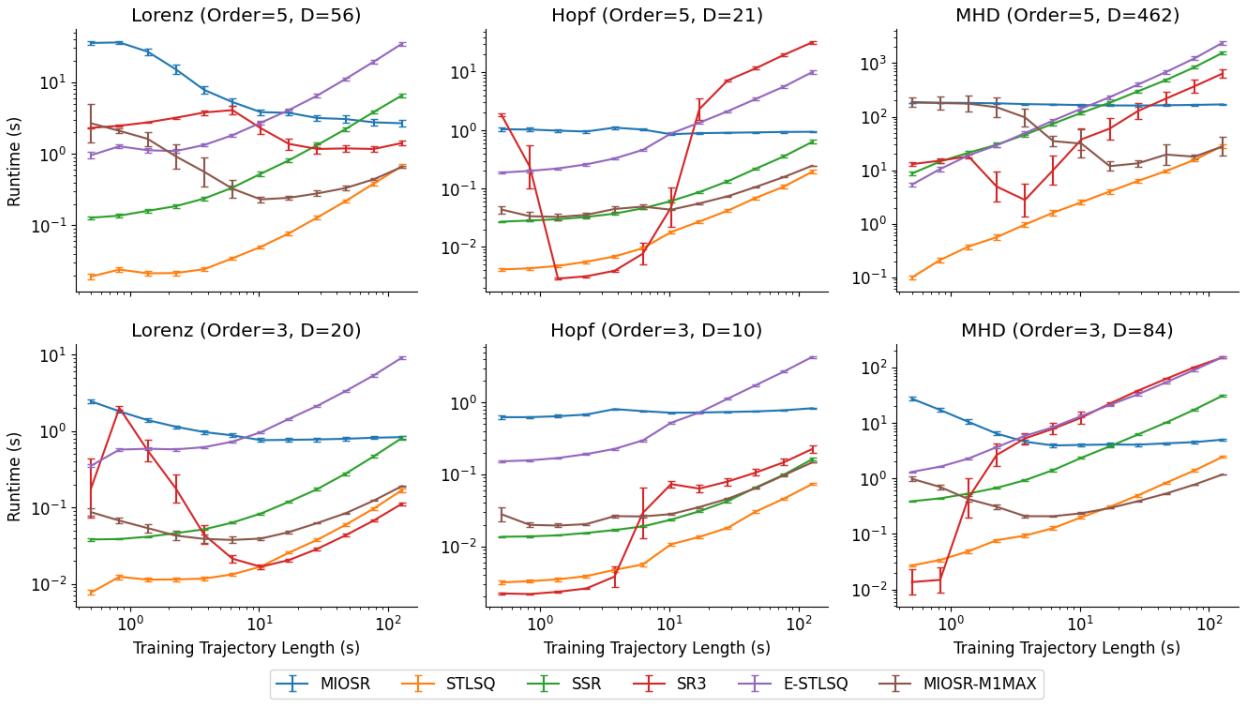


Figure 5.2: Comparison of sparse regression algorithm computational efficiency for the differential form of SINDy under varying amounts of training data for three different canonical systems: Lorenz, Hopf, and MHD. The top row uses a fifth order polynomial library for each of the three systems while the bottom row uses a third order polynomial library. Results are averaged over 50 trials with added Gaussian noise of 0.2%

necessitating more node exploration. Combining these results with those from Section 5.3.2, we conclude that either MIOSR takes a comparable amount of time while achieving the same accuracy, or it takes longer but the extra computational cost buys extra statistical performance. That is, regardless of data size, the cost of MIOSR is justified.

Another critical point is the dramatic speed up associated with utilizing more powerful hardware and parallelization. When run on a high end laptop (2021 Macbook Pro with M1 Max chip), as opposed to a single core of a cluster node, MIOSR can be up to 100x faster, while all other methods only improve by a few percent (see Figure D.2). This is because MIO is a highly parallelizable technology, as multiple threads can explore, expand, and prune different branches of the branch-and-bound tree in parallel. Therefore, one can simply allocate additional compute to achieve increasing levels of performance, and the time gap between optimal methods and heuristics will continue to close as more powerful MIO software and parallel hardware emerge in the future.

In addition to hardware, there are several other factors which understate how efficient MIOSR can be relative to the results reported in Figure 5.2. The first is that MIO requires less hyperparameter tuning than other methods because we are directly tuning the sparsity and not a proxy threshold. Second, for especially difficult or large problems, one can use warm starts from heuristic methods to get good initial solutions, or reuse solutions and models from previous steps in the hyperparameter search (which also avoids reconstructing the full optimization model). Finally, much of the computational effort is dedicated toward proving optimality by improving the dual lower bound [34]. Therefore, one could set a short timeout on the solver to get what is likely an optimal solution, but give up on the optimality guarantee if so desired.

While not studied in detail here, it is important to note that runtime also has a dependence on the sparsity parameter k because the number of possible models scales exponentially as $\binom{D}{k}$. However, given the dynamics are assumed to be sparse (e.g., $k < 10$), this scaling limitation is less relevant. We refer the reader to other works [31, 33] on general MIOSR methods for more extensive runtime benchmarking.

5.3.4 Physical Constraints

A central goal within scientific machine learning is to incorporate existing physical knowledge into the models, both to aid the learning process and to ensure that physically plausible models are learned. To illustrate the improved capability of MIOSR in service of this goal, we replicate the experiment performed by [67] on the two-dimensional Duffing system.

To briefly describe their setup, the 2D Duffing system is both a Hamiltonian system

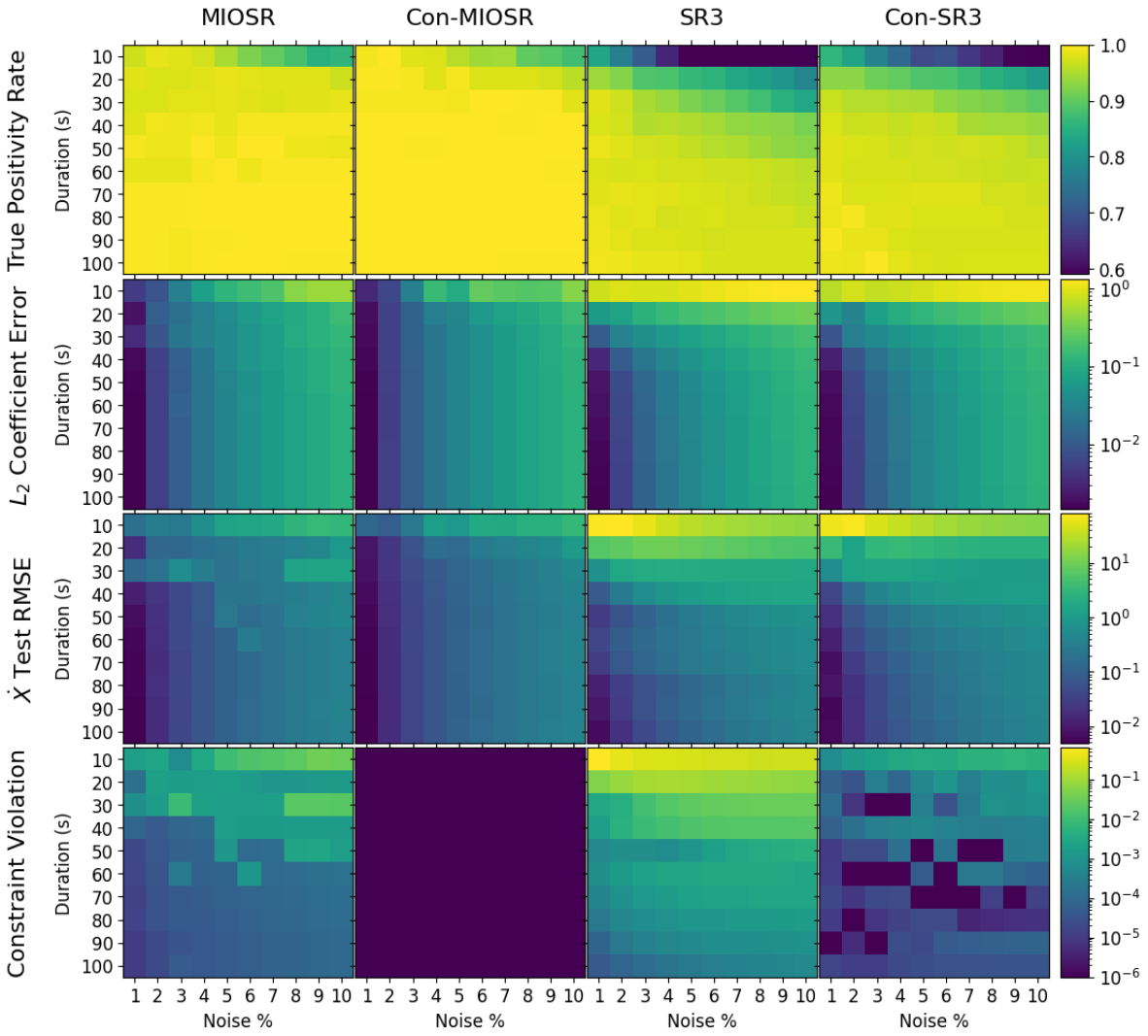


Figure 5.3: Performance comparison of constrained versus unconstrained sparse regression for the differential form of SINDy under varying amounts of training data and noise for the 2D Duffing system. Results are averaged over 50 trials with random initial conditions.

and a gradient system. These properties induce constraints on the coefficients because each individual governing equation must be a partial derivative of a Hamiltonian or potential function. The full system is described by

$$\dot{x} = X \tag{5.26a}$$

$$\dot{y} = Y \tag{5.26b}$$

$$\dot{X} = -\frac{\partial}{\partial x} V(x, y) \tag{5.26c}$$

$$\dot{Y} = -\frac{\partial}{\partial y} V(x, y) \tag{5.26d}$$

where $x,\ y$ give the spatial position, $X,\ Y$ give the momentum, and the potential function is

$$V(x, y) = -\frac{\omega}{2}(x^2 + y^2) + \frac{\alpha}{4}(x^2 + y^2)^2. \tag{5.27}$$

We use SINDy to just fit the spatial coordinates and set $\omega = -2$ and $\alpha = 0.1$. We refer the interested reader to [67] for the detailed derivation of the constraints, but for our purposes, the relevant fact is simply that the potential function imposes a set of equality constraints on $\boldsymbol{\Xi}$. We can then use a vectorized representation of the coefficients $\bar{\xi}$, and add $A\bar{\xi} = b$ to the MIO model where $A,\ b$ are based on the partial derivatives of $V(x, y)$. Because these constraints extend between dimensions, we use the joint formulation (5.17) to fit x and y simultaneously.

As in previous experiments, we sample 50 independent trajectories with random initial conditions, for every combination of training duration and noise depicted in Figure 5.3. For every trajectory, we run MIOSR with and without constraints, as well as SR3 with and without constraints, as it is the only baseline which naturally accommodates constraints. To stay consistent with [67], we use a third order polynomial library with a sample rate of 0.01 seconds. Additionally, we do not unbias the coefficients after feature selection and report the average constraint violation $\frac{1}{c}\|A\hat{\xi} - b\|_1$ where c is the number of constraints.

The immediate takeaways from Figure 5.3 are that adding constraints help both algorithms, especially with limited data, but that even without constraints, MIOSR is more accurate. This is in slight contrast to the findings of [67] where "the constrained and unconstrained models have nearly identical $[R^2]$ scores at all noise levels." However, this is because we study a more difficult statistical regime, one with less data and with coefficients of different magnitudes. In particular, they used training sets that included 20 independent trajectories, hence the constraints did not add additional information.

Another drawback of SR3 is that it enforces the constraints on a set of relaxed coeffi-

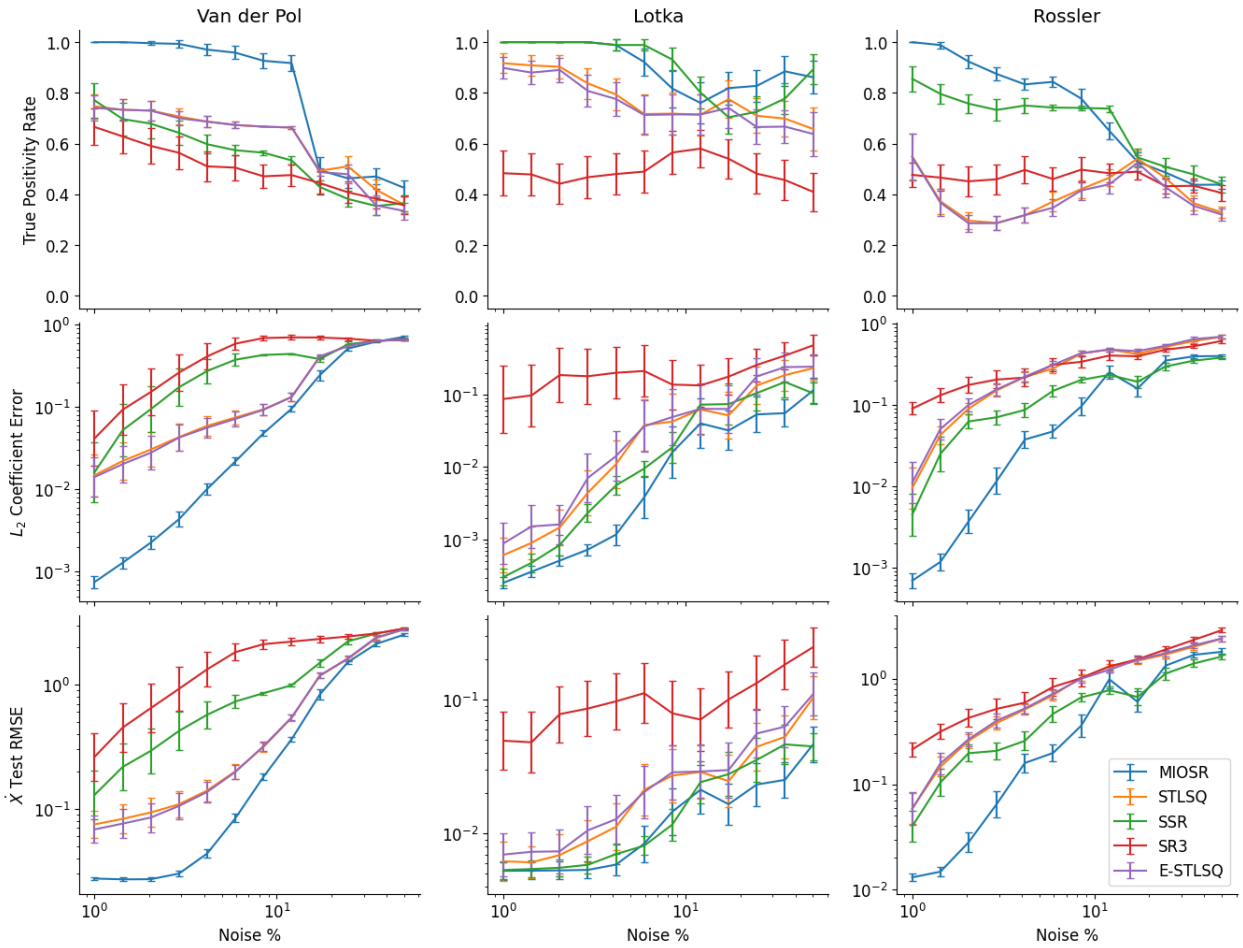


Figure 5.4: Performance comparison of sparse regression algorithms for the integral form of SINDy under varying amounts of added Gaussian noise for three different canonical systems: Van der Pol, Lotka, and Rossler. Results are averaged over 50 trials, each with 50 seconds of training data.

cients, hence the constraints are not strict and there exist nontrivial violations. This occurs most frequently in training regimes with less data and more noise. Unfortunately, these are precisely the regimes where constraints are most useful. MIOSR, in contrast, always satisfies constraints up to solver numerical precision, which can be adjusted or relaxed as desired. Finally, we observe that adding constraints does not add to the solution time of MIOSR, so the runtimes are comparable to those reported in Section 5.3.3, while constraining SR3 increases the runtime by about 30%.

5.3.5 Robustness

For the remainder of our experiments, we study system recovery under substantial noise, and therefore, we use the weak form of SINDy, where the data matrix is given by Eq. 5.4.

We first study robust recovery of three canonical ODEs: the Van der Pol oscillator [269], Lotka-Volterra equations [175], and the Rössler system [241].

The Van der Pol system is given by

$$\dot{x} = y \tag{5.28a}$$
$$\dot{y} = \mu(1 - x^2)y - x \tag{5.28b}$$

where we use $\mu = 3$. The Lotka-Volterra equations, sometimes also known as the predator-prey equations given their origin in modeling wildlife populations, are given by

$$\dot{x} = p_1 x - p_2 xy \tag{5.29a}$$
$$\dot{y} = p_2 xy - 2p_1 y \tag{5.29b}$$

where we set $p_1 = 1$ and $p_2 = 10$. Finally the Rössler system is

$$\dot{x} = -y - z \tag{5.30a}$$
$$\dot{y} = x + ay \tag{5.30b}$$
$$\dot{z} = b - cz + xz \tag{5.30c}$$

where we have $a = b = 0.2$ and $c = 5.7$.

For all three systems, we use a third order polynomial library with 50 seconds of training data with time intervals of 0.002 seconds (i.e., 25000 total time steps). Our weak libraries are composed of 2400 spatial domains each with 400 points per domain. To validate weak models on noisy data, one can either use the weak form of the validation set, or try to differentiate the validation data with more aggressive smoothing. We observe that the weak form of the validation data yields a less reliable tuning signal, partially because the weak form uses randomized domains, and leads to less sparse models. However, the numerical derivative also becomes increasingly unreliable with more noise. Therefore, under 15% noise we use the a smoothed derivative (with window size 21) for validation, and use the weak form of the validation data above 15% additive noise.

Figure 5.4 depicts our results. The general trend is that for low to medium amounts of noise, MIOSR is significantly more accurate, often perfectly recovering the underlying model. Even with just 1% noise, the baselines struggle to identify the true model coefficients. However, unlike in Section 5.3.2, the baselines are mostly identifying all of the correct coefficients, but find small false positive coefficients that are fitting noise.

At very high levels of noise, MIOSR starts to break down, converging to the approximate

performance of heuristic methods. While more elaborate ensembling could help, we believe better data preparation is likely a more effective way to learn accurate models, especially with MIOSR. Examples include applying more aggressive smoothing, expanding the number of domains or points per domain in the weak form, or utilizing tailored differentiation techniques [268]. Beyond data preparation, there is also more recent work on more sophisticated iterative schemes to prune the library by regressing on Fourier transforms [84], which could benefit from utilizing optimal methods.

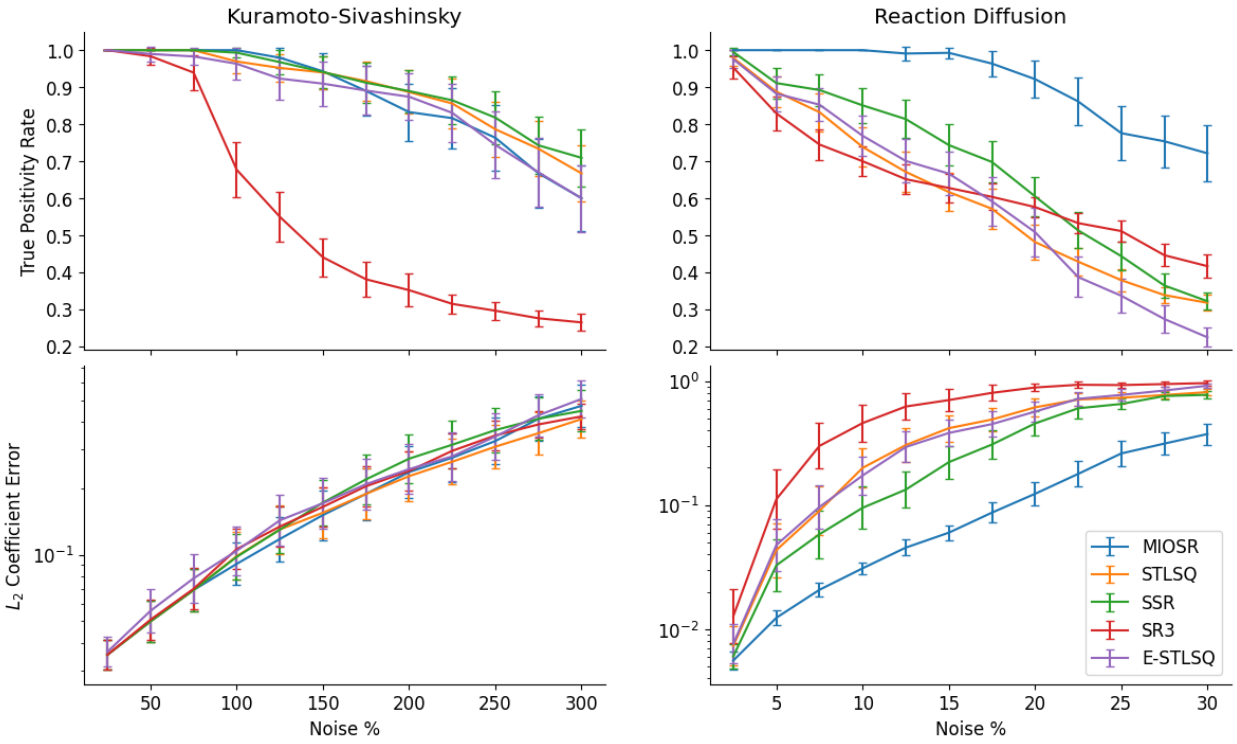


Figure 5.5: Best PDE model found by sparse regression algorithms for the integral form of SINDy under varying amounts of Gaussian noise for two different canonical PDE systems: Kuramoto-Sivashinsky and reaction diffusion. Results are averaged over 50 trials with random initial conditions.

5.3.6 PDEs

For our last experiment, we study the recovery of scalar PDEs under substantial measurement noise. This is likely the regime most relevant to advancing modern science and engineering practice. In particular, we study the one-dimensional Kuramoto-Sivashinsky equation [256], an early model of laminar flame fronts, and the two-dimensional reaction diffusion system, a ubiquitous model in chemistry. The Kuramoto-Sivashinsky in spatial dimension x and time

t is given by

$$u_t = -uu_x - u_{xx} - u_{xxxx} \tag{5.31}$$

where the notation u_{xx} denotes the second partial derivative with respect to x. We study the 2D reaction diffusion system given by

$$u_t = \frac{1}{10}u_{xx} + \frac{1}{10}u_{yy} + u - uv^2 - u^3 + v^3 + u^2v \tag{5.32a}$$

$$v_t = \frac{1}{10}v_{xx} + \frac{1}{10}v_{yy} + v - uv^2 - u^3 - v^3 - u^2v. \tag{5.32b}$$

For both systems we use weak third order polynomial libraries, with up to fourth order derivatives for Kuramoto-Sivashinsky and second order derivatives for reaction diffusion, yielding library dimensions of 19 and 109 respectively. For Kuramoto-Sivashinsky, our weak library is composed of 200 spatiotemporal domains, each with 50 points, sampled 10 times a second for 25 seconds on a periodic domain with 1024 spatial points. For reaction diffusion, our weak library is composed of 400 spatiotemporal domains, each with 36 points, sampled 50 times a second for 5 seconds on a 256×256 periodic grid.

Unlike the previous experiments, we do not perform model tuning and selection using AIC, due to the computational cost of fitting a large number of PDEs. Instead, we perform an achievability analysis, loosely inspired by [180], where we choose the algorithm parameters knowing the dynamics, to determine if it is even possible to learn a correct model given an appropriate model selection method. In particular, for MIOSR, we set the sparsity constraint to be the actual dimensionwise sparsity; for STLSQ, we try several thresholds slightly below the actual smallest coefficient; for E-STLSQ, we use library ensembling [101] and take the k terms with the highest inclusion probability, where k is the true sparsity.

Figure 5.5 depicts our results for PDE learning based on 50 trials with random initial conditions and varying amounts of additive Gaussian noise. These results further underscore how MIOSR has a substantial edge over heuristic methods when using larger libraries or when the system coefficients are of different orders of magnitude; it also further illustrates the converse, that problems with smaller libraries and similar magnitude coefficients do not benefit from exact methods because the heuristics converge to the correct solution. Regardless of optimizer, we see how effective the weak SINDy framework can be in identifying noisy systems, with Kuramoto-Sivashinsky often being recovered even with 300% measurement noise.

5.4 Application to Language Models

5.4.1 Motivation

In this section, we analyze to what extent the Llama3-8B language model [267] is capable of modeling and in-context learning how to complete dynamical systems trajectories. This section is inspired by recent work [172] showing that LLMs do in fact learn governing principles of dynamical systems.

As we discussed in the previous chapter, it is an open question the degree to which state of the art language models possess internal world models. While we previously showed that LLMs possess basic spatiotemporal representations, these representations were *static*. However a world model required more than just a literal map of the world—it requires dynamic simulation. In fact, an ability to learn efficiently from examples and perform mental simulations a key ingredients of intelligence itself.

To the extent that models can in-context learn dynamical systems, this observation also raises interesting interpretability questions. In particular, it suggests that features cannot be fixed directions in activation space, since the model needs to somehow create novel features on the fly, and allocate activation space for them.

5.4.2 Case Study: Logistic Map

Our first case study concerns the logistic map

$$x_{n+1} = rx_n(1 - x_n) \tag{5.33}$$

a one dimensional discrete time chaotic dynamical system parameterized by r.

Setup We begin by sampling a trajectory with $r = 3.9$, rounding each element in the sequence to the nearest thousands place, and concatenating the sequence values into a single string separated by commas. This string is then tokenized, usually as `|0|.|124|,|0|.|424|,|...`, and fed through the model.

In-context learning

The first obvious question is whether an LLM can even model a simple dynamical system at all. To test this, we collect the predictions of the LLM before every non-trivial numeric "answer" token. That is, we check the next-token prediction on every decimal point, where the correct prediction is the three digits following the decimal point. Conveniently, the

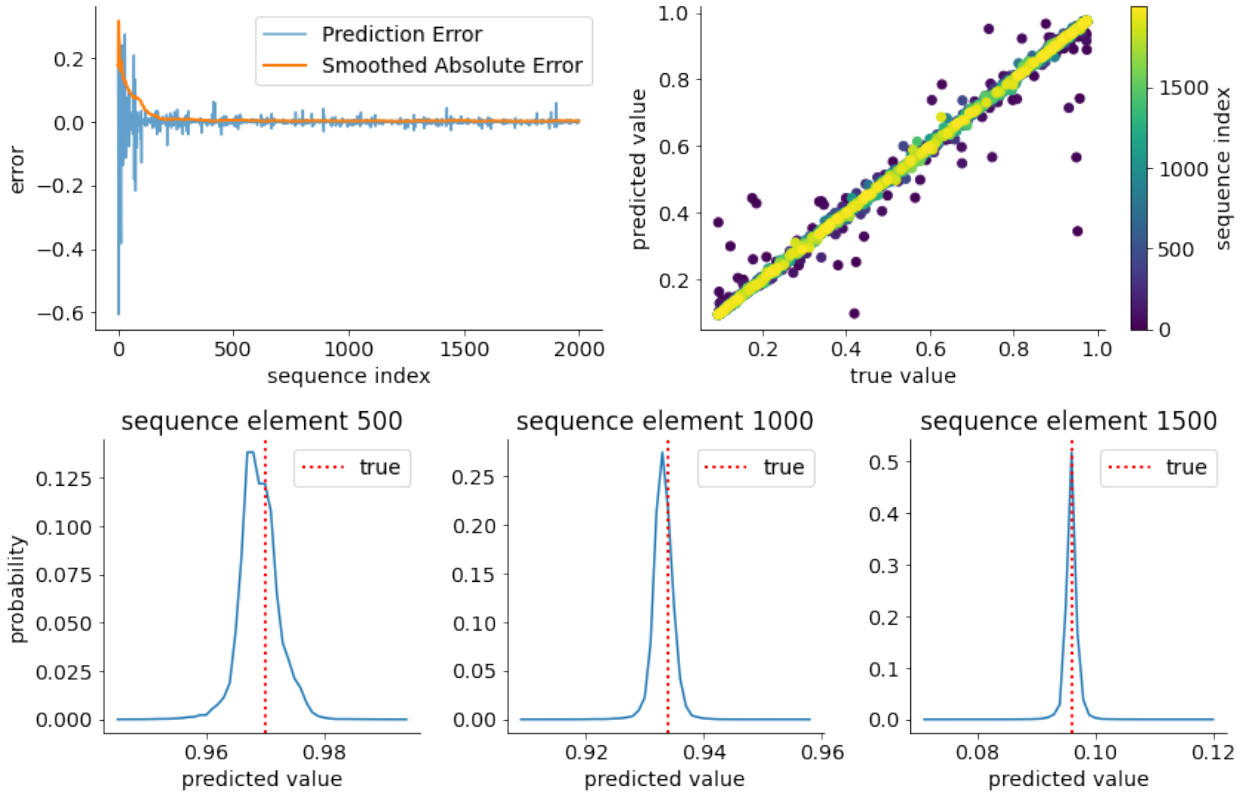


Figure 5.6: Prediction error of Llama3-8b on predicting the continuation of a logistic map sequence. (Top) error as a function sequence position. (Bottom) probability distribution for next element in the sequence for steps 500, 1000, and 1500 respectively.

LLama tokenizer has a separate token for every 3 digit number (with zero padding), so we can compute the model's scalar prediction by taking the weighted (by probability) average over the three digit tokens in the model's vocabulary.

As can be seen in Figure 5.6, the model can predict the next element of the sequence with impressive accuracy, and clearly benefits from in-context learning. That is, as the model sees more elements of the sequence, it continues to improve its predictions, not only in its mean estimate, but also in the variance of the prediction.

SINDy on System State Probes

Probing Having established that the LLM is capable of in-context learning the continuation of a dynamical system, we perform several preliminary experiments to better understand what sort of representations are enabling the model to perform these predictions. First, we train linear regression probes on the residual stream at every layer of the comma tokens separating sequence elements to predict the previous element is as well as the next element of the sequence. In Figure 5.7, we report the out-of-sample R^2, mean squared error (MSE),

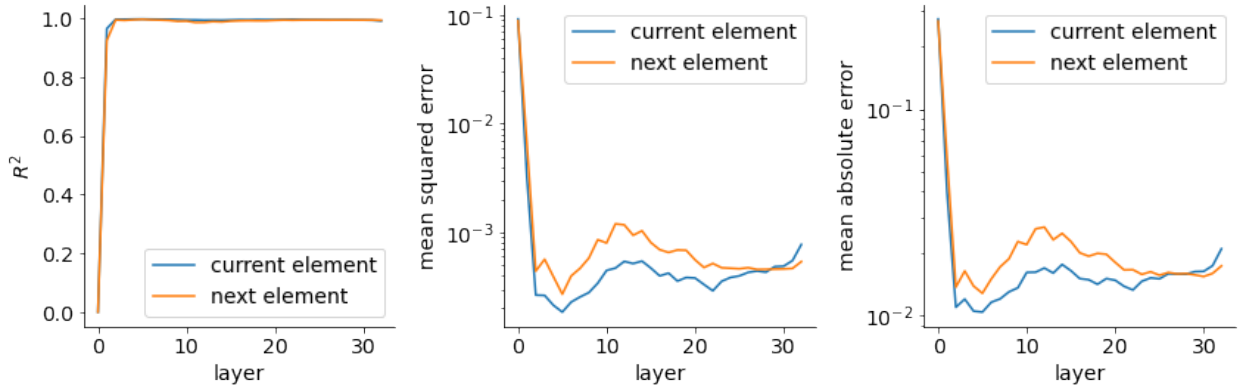


Figure 5.7: Out-of-sample linear probe performance on current sequence element and next sequence element by layer.

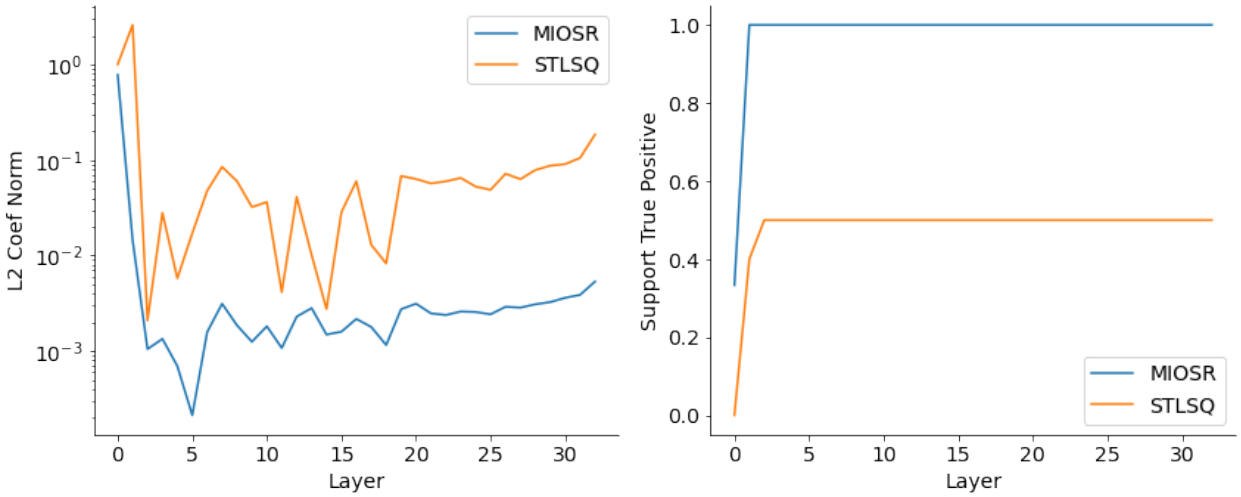


Figure 5.8: System recovery metrics of SINDy model fit to probe predictions of logistic system using the STLSQ and MIOSR optimizers.

and mean absolute error (MAE) for both types of probes for each layer. As expected, the current token estimate is very accurate from early on in the network since the previous token value explicitly gives this number. More surprisingly, the *next* sequence value is also linearly decodable from essentially the first layer onwards.

SINDy Recovery To further validate the accuracy of the probes, we next train a SINDy model on the recovered system (i.e., the sequence elements as predicted by the probes) with both MIOSR and STLSQ optimizers. This is a kind of bi-direction evaluation, where (1) we use SINDy to show that the probes (and hence the underlying network) faithfully represent the dynamical system, and (2) validate that SINDy methods can recover system dynamics that are "noised" by an LLM, i.e., can still perform in a less contrived setting where

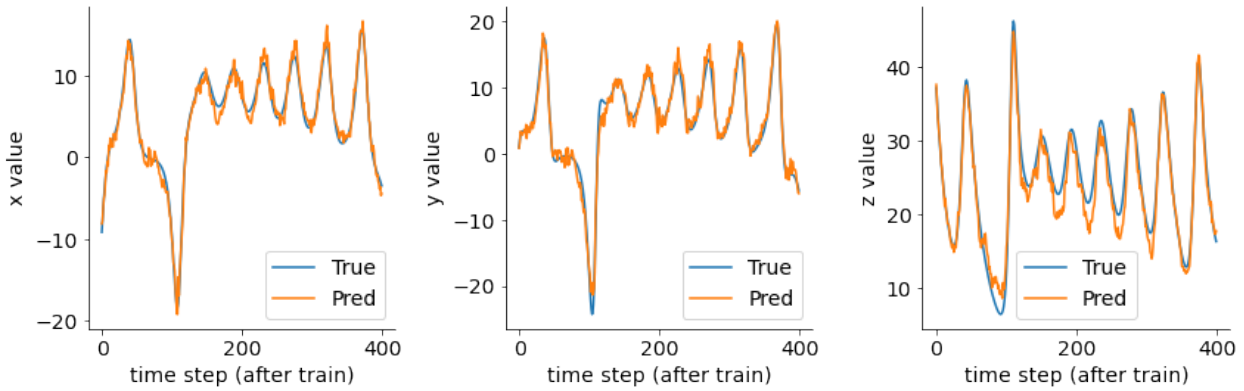


Figure 5.9: Out-of-sample (time separated) probe predictions for the x, y, and z coordinate of the Lorenz system compared to ground truth for layer 16.

measurement accuracy is not perfect (as opposed to Gaussian noise).

We report the normalized coefficient error and the support true positivity rate discussed in Section 5.3.1 in Figure 5.7. As can be seen, from layer 1 onwards, the MIOSR optimized SINDy model is able to perfectly recover the underlying system from the probe states. In contrast, the STLSQ baseline includes spurious higher order terms.

5.4.3 Case Study: Lorenz System

Setup In the previous case study, it was perhaps not so surprising that it was possible to probe out the current state given the state was fed at input, i.e., the current state was fully observable. To make a more challenging example, we examine the three dimensional Lorenz system with a twist: we only give the LLM the x-coordinate as input. That is, just as before, we sample a trajectory, round the entries, and convert it into a string and then a sequence of tokens, but here we only include the first of three dimensions. Despite only having access to one dimension as input, we will still attempt to probe out the y- and $z-$ coordinates to see the extent to which the model is able infer the other variables from limited information.

Probing For our first experiment, we sample a trajectory of the Lorenz system with length 1600. We then take the first 1200 observations as the train set and the remaining 400 as the test set. We then train linear probes on the true x, y, and z coordinates for each of the layers. In Figure 5.9, we depict the probe predictions versus the true coordinates for probes trained at layer 16. Remarkably, even though there was no state data for y and z coordinates, these are still *linearly* recoverable, suggesting the model is able to dynamically infer the coordinates in-context!

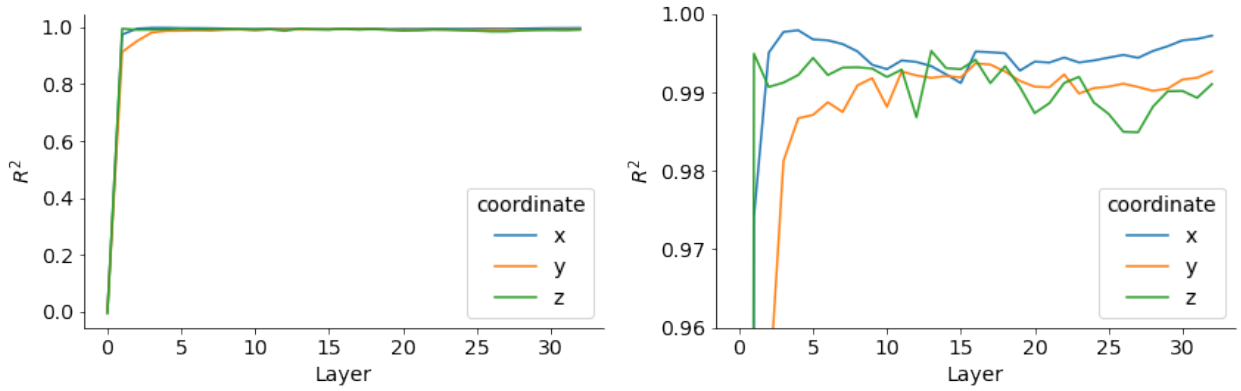


Figure 5.10: Out-of-sample R^2 for probes trained on every layer of the x, y, and z coordinate of the Lorenz system. Right subplot is zoomed in version of left subplot.

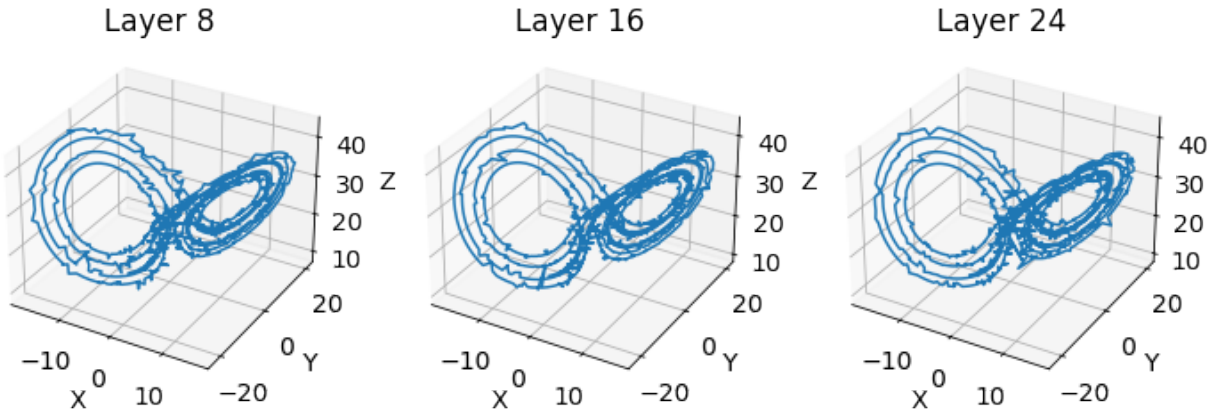


Figure 5.11: Recovered Lorenz trajectories from linear residual stream probes at different layers.

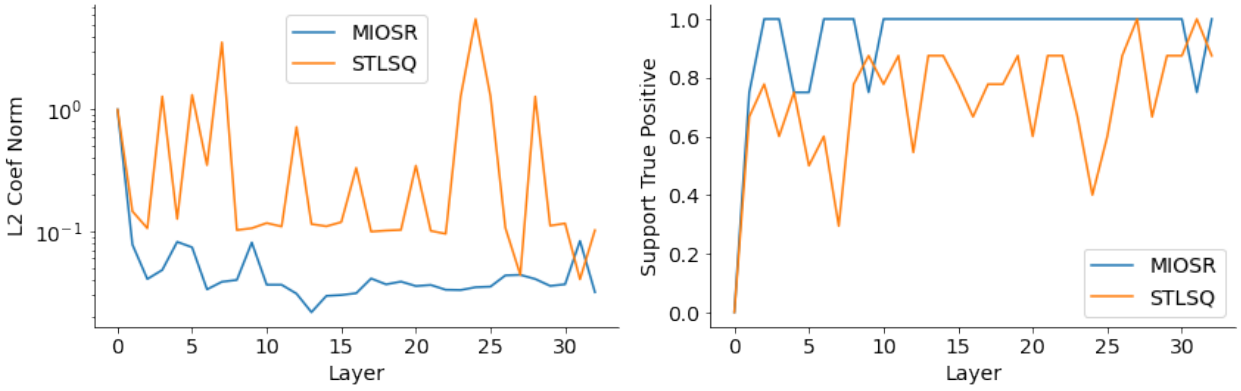


Figure 5.12: System recovery metrics of SINDy model fit to probe predictions of Lorenz system using the STLSQ and MIOSR optimizers.

SINDy Similar to our experiments on the logistic system, we perform a bi-directional evaluation, where we train a SINDy model on the coordinates recovered by the linear probes. Figure 5.11, depicts examples of the recovered trajectories of the Lorenz system from different network depths. However, the noise in these estimates is substantially greater than those in the logistic system, which is further compounded in the differentiation step, resulting in a poor fit. Therefore, to recover the system, we instead use the weak or integral formulation of SINDy (as introduced in Section 5.3.5) where both sides of the SINDy equality are integrated, which results in a regression target which is denoised.

Using the integral form, we fit the SINDy model to the recovered coordinates, again with both MIOSR and STLSQ optimizers. We report the results of the recovery for all layers in Figure 5.12. Similar to the results on the logistic system, MIOSR is consistently more accurate in recovering the true coefficients of the system, often with perfect accuracy of the sparsity structure. It also achieves this impressive accuracy by the 3rd layer of the model, implying there must be a relatively simple (or at least shallow depth) circuit which is able to infer (or at least produce linear representations of) the other coordinates.

5.5 Conclusion

In this work, we demonstrate the superior performance of mixed-integer optimization in learning sparse nonlinear dynamics (MIO-SINDy) as compared with popular heuristic approaches. The biggest advantage is in finding models which are as simple as possible, but not simpler—a model which learns the truth and only the truth. In addition to more accurate support recovery, MIO sparse regression is capable of incorporating a huge range of additional model structure as auxiliary objective terms or constraints, while solving the underlying optimization problem to provable optimality.

Contrary to the predictions of complexity theory, MIOSR is highly tractable, and can actually be faster than heuristics for large amounts of data. Indeed, MIOSR runs slower when the regression is harder, that is, when the sample size is small, signal-to-noise ratio is low, or the coefficients span multiple orders of magnitude. However, this is exactly where heuristic methods perform poorly, so MIOSR requires more time when it improves upon heuristic methods while being comparable in running time when the dynamics are more easily recoverable. Given the practicality of the approach, and the theoretical guarantees, we see no reason why MIOSR should not be the default choice of optimizer for real applications.

Due to the modularity of the SINDy framework, MIO-SINDy is compatible with other methodological advancements concerning data preprocessing, library construction, numerical differentiation, and outer loop algorithms. We restricted our study of these extensions to

the weak form and PDE learning, but we expect MIOSR to offer similar benefits to other variants like control [53] or identifying implicit equations [144]. We hope domain experts find use for the additional modeling and statistical power afforded by MIO. We believe this is an exciting development that advances the state-of-the-art in system discovery.

Data Availability

All data, code, and results are available at our Github repository https://github.com/wesg52/sindy_mio_paper.

Acknowledgements

This chapter is based off of joint work Dimitris Bertsimas [30].

Appendix A

Sparse Probing Appendix

A.1 Frequently Asked Questions

A.1.1 Why expect ambitious interpretability to be possible or worthwhile?

We divide our answer into two parts: why it should be possible even in theory, and how we interpret the progress so far.

Although gradient descent has no reason or incentive to learn representations scrutable to humans, the same could be said of all the biological structures "learned" by natural selection. Yet biological structures are constrained by the laws of nature—an organism must make efficient use of limited energy and space, its genetic material is encoded in RNA and DNA, its functionality is encoded in proteins, and these proteins fit together into biological circuits. Although these biological circuits are complex, often idiosyncratic to organisms, and at times seem inscrutable, these constraints give us insight into their functioning and create common motifs and principles that can be reverse-engineered [6]. In a similar way, neural networks are trying to achieve a complex tasks and have significant constraints, namely, the number of parameters, non-linearities and weight norm. And more broadly, the algorithms their architecture allows them to represent. By studying analogous structures of neural nets—both at a mathematical level [96] and in the wild [274]—it is possible to uncover some of these principles and motifs, and to start gaining a foothold in reverse-engineering these artificial circuits.

A potential source of pessimism for ambitious interpretability is the progress made thus far. Though we have gained some insights about model internals, most works in mechanistic interpretability have focused on small or toy models [58, 72, 96, 202, 274], while frontier

models become larger at a far faster pace [51, 71, 218]. While more scalable approaches to interpretability have often been shown to have results that are easy to misinterpret, or sometimes actively misleading [3, 39, 126, 149]. It is natural to look at this confusion and seeming incomprehensibility and to feel discouraged. Yet, natural structures were no doubt similarly incomprehensible to early pioneers in molecular biology, genetics, and neuroscience, exhibiting an emergent complexity that seemed irreducible. However, rather than claiming cells, genes, or brains were "uninterpretable," entire scientific disciplines emerged which have made great strides in understanding the core principles in sufficient detail to enable intervening, engineering, and controlling biological systems. We believe artificial neural networks can be fully interpreted—even reverse-engineered—but doing so requires a comparable amount of effort as interpreting biological ones. In many ways, this emerging *artificial neuroscience* is unusually amenable to the scientific method [210]: it is possible to run arbitrary counterfactual experiments, iteration times are rapid, resource requirements are minimal, all with full observability and measurement precision equal to floating point machine precision. While ambitious, we think such an effort is paramount for addressing issues in the alignment and control of increasingly capable AI systems.

A.1.2 What do you mean by a neuron?

The term **neuron** is sometimes used to refer to elements of any activation tensor, in the standard basis. Here, we instead reserve neuron to refer to the elements of the model activations immediately after an elementwise non-linearity: the post GELU activation in the hidden state of a transformer's MLP layer. We do *not* use neuron to refer to elements of a transformer's residual stream, a layer's output, or the key, query or value within an attention head. These are the output of a linear map, and so do not have a privileged basis [96]: we expect the model to function the same under an arbitrary rotation (modulo optimiser quirks [99])

A.1.3 What is the difference between polysemanticity, superposition, and distributed representations?

These are similar concepts with crucial differences, that are commonly confused. We propose the following schema:

Superposition is the phenomena when an activation represents more features than it has dimensions. Each feature is a direction in space, and as a consequence they *cannot* all be orthogonal.

Polysemanticity is when a neuron seems to represent multiple, unrelated concepts. This is in contrast to a **monosemantic** neuron that activates if and only if a certain feature is present.

Distributed representations are where a feature is represented as a linear combination of neurons, i.e., a direction that does *not* correspond to a single basis element. Also known as non-basis aligned features. Notably, this could be a fairly sparse distributed representation (e.g., using 2-5 neurons), fairly dense (e.g., using 10-20% of all neurons in the layer), to completely dense/not at all basis aligned.

Notably, polysemanticity and distributed representations are **local** notions; they can be demonstrated by studying individual neurons or features respectively. Superposition is a **global** phenomena, and requires identifying more features than neurons to conclusively show.

A.1.4 What is the relationship between polysemanticity, superposition, and distributed representations?

A linear representation may be distributed because of rotation/skew (e.g., when lacking a privledged basis), composition, or superposition [211].

Superposition necessarily implies polysemantic neurons (and likely distributed representations), because there are more features than neurons! However, it is possible to have distributed representations and polysemanticity without superposition. There may be as many features as dimensions, but in some rotated basis not aligned with the neuron basis.

Indeed, without a privileged basis, this is what we would expect to observe even without superposition. The model has no incentive to align features with basis elements, and as we would expect, prior work [44] has found polysemanticity of residual stream basis elements. However, it is surprising that this should occur in the presence of an elementwise non-linearity. If a model is acting as a feature extractor, it is useful to represent features such that they can independently vary. Individual neurons have a separate GELU, but if multiple features share a neuron then they may significantly interfere with each other.

Moreover, we can have polysemantic neurons and superposition without a distributed representation if a neuron represents multiple features, but each of those features is *only* represented by that neuron. This is a form of superposition. The model will then need to have circuitry to disambiguate which feature activated the neuron, but we can imagine ways this could be efficiently implemented. A toy example: a model could have 100 neurons, each of which represents both some feature of Python code and some feature of romance novels. Python code and romance novels are mutually exclusive contextual features that

could each already be computed and represented in the residual stream, which can then be used to disambiguate which feature each neuron activated for. Though this could be considered a certain kind of distributed representation where a feature is represented as a linear combination of its neuron and the romance/Python feature.

Our interpretation of our results is that this is further evidence that, at least in early layers, models engage in superposition involving both polysemanticity and distributed representations.

A.1.5 What is the difference between faceted and polysemantic neurons?

A basic approach to detecting polysemanticity is to apply clustering to the activations of texts that strongly activate a neuron (e.g., the residual stream immediately before the MLP layer [42]): if each cluster has some shared semantic meaning, then if there are multiple clusters the neuron is seemingly polysemantic! And indeed, if there is a single clear cluster, that is evidence of monosemanticity. However, there are two distinct phenomena here: related and unrelated clusters. A neuron is **faceted**[205] if it activates for multiple different things with shared meaning, e.g., a cutlery neuron activating for knives, forks and spoons. While a **polysemantic** neuron activates for clusters without a shared meaning, e.g., dice and poetry. Unfortunately, this is currently a subjective definition: what does it *mean* to have shared meaning? We see formalizing these intuitive concepts as a promising area of future work.

A.1.6 What are the two different kinds of interference, and how do they affect superposition?

The phenomena of superposition arises because models can decrease loss by representing more features, yet having non-orthogonal features introduces interference which increases loss. This creates a pareto-frontier trading off the two for each feature and each model, and this determines the representations a model learns. Crucially, interference between two features A and B decomposes into two conceptually different forms of interference [200]: **alternating interference** where A is present, B is not present (or vice versa), and the model needs to tell that A is present and B is not, and **simultaneous interference** where A and B are present, but the model needs to tell that A and B are both present, but that, for example A is not present at twice strength. To understand the difference, let's consider the case where A and B are binary features corresponding to different directions of unit norm.

Alternating interference is fundamentally about distinguishing high activations (feature

A is on) from low activations (feature B is on, feature A is off, but its direction is not the same as feature A's direction), which is fairly straightforwards with GELUs and softmaxes. But simultaneous interference is fundamentally about treating medium activations (feature A is on, feature B is off) as the same as high activations (feature A is on and feature B is on) but different from low activations (feature A is off and feature B is on, or both are off). This is much harder to do with GELUs or softmaxes.

In toy models, models seem to avoid superposition with high simultaneous interference (e.g., correlated features), but tolerate high alternating interference (e.g., anti-correlated features). Notably, [98] observed that independent features occurring with probability p engage in more superposition when p is lower - the probability of alternating interference is linear in p while simultaneous is quadratic in p. The expected loss reduction from representing a feature is also linear in p, suggesting that whether or not a model engages in superposition is predominantly driven by the costs of simultaneous interference, not alternating. For sufficiently rare and uncorrelated features simultaneous interference is a non-issue.

Mutually exclusive features like n-grams are an extreme case where simultaneous interference can never occur, and so are well suited to superposition.

A.1.7 How does the type of feature affect how it might be expressed in superposition?

Three notable categories of features are **continuous** (e.g., the height of an object), **binary** (e.g., whether 'social security' is present) and **categorical** (e.g., whether an object is blue, red or yellow). Categorical features can be represented as one-hot encoded binary features, so it is most instructive to compare binary features and continuous features. We argue that interference is significantly greater for continuous features than binary features—intuitively alternating interference will prevent us from distinguishing small values of the true feature and large values of an interfering feature.

This effect has been observed in the literature when studying toy models. [98] studied continuous variables and found superposition tending not to compress more than 5 features into two dimensions (and with significant interference), while [132] studied binary variables (memorization) and found that models could compress significantly more features into two dimensions with minimal performance lost (though this is confounded, as memorization is also mutually exclusive).

A.1.8 Why might we expect n-gram detection in particular to use superposition?

n-gram detection is unusually well suited to superposition for several reasons. n-grams are mutually exclusive ('social security' cannot co-occur with 'prime factors', for example), and are binary variables.

Moreover, n-grams are an example of a simple feature: to detect 'social security', a neuron need only compose with a previous token head to detect ' social' and with the current token embedding to detect ' security', which will create a sharp decision boundary: a significant gap between the smallest neuron activation when the feature is present and the largest neuron activation when the feature is not present. This means that for downstream computation the ' social security' feature can be used with minimal noise from false positives. In contrast, complex or subtle features like whether a text is in French may require compiling many tiny pieces of evidence, creating an unclear decision boundary between true and false. In this way, despite the ground truth being a binary feature, models may think of complex and nuanced features as continuous variables associated with the probability of being present, making these features harder to store in superposition.

A.1.9 What is the difference between residual stream and neuron superposition?

Residual stream superposition is an example of **representational superposition** (sometimes called bottleneck superposition) and is fundamentally about compression. That is, mapping a high dimensional space to a low dimension space such that the model can recover the compressed features at a later point in the network. This also implies that the model has already computed the feature, and the superposition is just for the purposes of storage. In addition to occurring in the residual stream, we also expect the keys, queries and value vectors of attention heads to employ representational superposition to represent more features than there are dimensions.

In contrast, neuron superposition is an instantiation of **computational superposition**, where a model can extract or compute with more features than it has dimensions (in this case neurons). If there exist more features than neurons that must be potentially computed, then each individual neuron must (on average) be involved in the computation of many features. Prior work on superposition[98] predominantly (but not entirely) focused on linear superposition, and we understand neuron superposition less well. As another example of computational superposition, preliminary findings suggest attention layers also employ su-

perposition between heads to implement a larger number of skip-trigram circuits than would naively be possible [141].

Conceptually, computational superposition is a more complex phenomena. To perform superposition, the model must trade-off the benefit of representing more features against the costs of interference. With representational superposition the interference between features is just given by their dot product, and by embedding features as almost orthogonal vectors we can achieve low interference while representing many more features. But computational superposition involves nonlinearities (elementwise GELUs in the neuron case), which introduce significantly more interference in ways we do not fully understand.

A.1.10 How can you conclude you found superposition as opposed to simply non-basis aligned features?

We think that our findings of superposition rest on two sub-claims: (1) that there exist more features than neurons to represent them and (2) that the neuron basis is meaningful, such that models are incentivized to align features with neurons *if* there are fewer features than neurons, and that superposition is implemented as a sparse combination of neurons.

We think claim (1) is evidently true, given the extremely long tail of features in internet text in addition to all the possible meaningful n-grams and other repeated patterns that model would want to memorize. In practice, models can detect and respond appropriately to a wide array of compound words and facts, seemingly far more so than the size of their vocabulary or number of neurons. As an example, just take the case of names of people. There are likely more people that GPT-3 knows about than it has neurons (5 million). A first and last name gets split into at least two tokens, but with most surnames being tokenized further. We think it would be impossible to fit all of the possible features of these people as just linear combinations of the token embeddings, therefore requiring some nonlinearity to detect the presence of a sequence of tokens corresponding to a name, and adding back the appropriate information about this person.

To further support claim (2), we run an experiment on the compound words dataset comparing the classification performance of the top 50 individual neurons in a layer to the top 50 dimensions in a random basis. In particular, for both the standard neuron basis, and the activation dataset when multiplied by a random $d \times d$ Gaussian matrix, we train 1-sparse probes for the 100 neurons with the largest class mean difference, and report the out of sample F1-score for the top 50 (averaged over 5 such random bases). Our results in Figure A.8 demonstrate that individual neurons are more predictive than random linear combinations of neurons, which implies that the neuron basis is meaningful.

Although there are usually 1-5 neurons with notably higher F1 score than the rest, the full top 50 neurons still maintain high F1 scores. This seems consistent with results from Section 8 of [98], where there exists a main neuron per feature, with a longer tail of less important neurons. Intuitively, just achieving linear separability is unlikely to be sufficient for overcoming interference. To minimize interference models likely want to maximize the separation between potentially interfering features, implying that the potential number of neurons in superposition may be much higher than the minimal number required to achieve near perfect classification performance. If the largest activation on negative examples is close to the smallest activation on positive examples, then using the represented feature for downstream computation will introduce significant noise. For a rough approximation, in Figure A.9, we measure the logistic test loss, rather than F1 score, of k-sparse probes for a range of values of k. Again, we see that probes trained in the neuron basis generally have lower loss than those trained in a random basis. Perhaps more interestingly, there appears to be two regimes governing the loss with respect to sparsity, one with power law scaling, and then a kink with no returns to using additional neurons. It is plausible this kink gives an indication of the "true" sparsity used to implement features in superposition.

Finally, we note that these results likely understate the basis alignment, because they are with respect to an "easy" classification task: just distinguishing bigrams `XY` from bigrams `XZ` or `WY` rather than all possible n-grams, and that there likely exist many confounding contextual correlates that improve the performance of the random basis dimensions. Designing more careful experiments to untangle these different effects is an important area for future work.

A.1.11 Should we ever expect to find monosemantic neurons?

Monosemantic neurons have been convincingly demonstrated in image classification models [58], but we are not aware of work rigorously showing monosemanticity of transformer language model neurons. So on an empirical front, this is an open question. It has been shown in toy models [98] that in the presence of significant differences in feature importance (i.e., expected reduction in loss from representing that feature) and for continuous features that the most important features will have dedicated monosemantic neurons. Textual features vary significantly in importance (e.g., detecting Python code is far more important than knowing some niche fact) suggesting that crucial features may get their own neuron (e.g. our context neurons). However, for important binary features this is less clear. Further, even important and highly prevalent features likely have rare features that they are anti-correlated with, and may be represented in superposition with these. Rare features are a case that may

be particularly hard to detect.

A.1.12 Which sparse probing method should I use?

It depends. If your experiments must be extremely fast or scalable, then simply doing maximum mean difference is likely best. If you are less runtime sensitive and want to sweep over a large range of k, then adaptive thresholding is most appropriate. Finally, if you require any sort of formal guarantees then you should use optimal sparse probing, with the caveat that this requires potentially substantial compute time.

A.1.13 Why think of features as directions?

This is known as a linear, decomposable representation [98]—the model's activations may be decomposed into independently varying features, and these features correspond to directions in activation space. The intuition behind this is that the primary capability of models is doing linear algebra—addition and matrix multiplication, which further breaks down into addition, scalar multiplication, and projecting onto specific directions. Given these capabilities, it is especially natural for a model to represent features as directions: if a later layer wants to access a feature it can project onto that feature's direction, a neuron can easily access and combine multiple features, features can vary independently, and the component in that feature direction represents the strength of that feature. If a model used some more complex and non-linear representation then it would need to dedicate downstream parameters to decoding these, a costly endeavour. Previous findings of monosemantic neurons in image models[58] and reverse-engineering toy models on mathematical tasks[72, 202] have found linear features, however the empirical evidence base here is slim. Mechanistic interpretability is still a young and pre-paradigmatic field, and we see gaining empirical evidence for foundational assumptions like this as a key area of future work. If the features-as-directions hypothesis were false, the task of reverse-engineering models would seem far more daunting: models are extremely high dimensional objects, and we must be able to decompose them *somehow* into independently meaningful units in order to evade the curse of dimensionality and understand them.

A naive way to test this hypothesis is by using linear and non-linear probes (e.g., a one hidden layer MLP) to extract features: if non-linear probes work and linear probes do not, then this is evidence that those features are represented non-linearly. However, this can be illusory, as the model may instead linearly represent simpler features and the non-linear probe itself does the computation to find the more complex feature. In the extreme, we could imagine having our 'probe' be GPT-3 trained on the token embeddings—this can

surely probe for any feature, represented or not! This is a concern even with linear probes, eg a model may represent whether a shape is red and whether it is a triangle as directions in space, and a linear probe on the sum of these directions may seem to indicate an 'is a red triangle' feature.

There was a recent natural experiment in the literature supporting the features as direction hypothesis. [164] trained a model to predict the next move in the board game Othello, and were able to probe for an emergent representation of the board state. They could only find this representation with non-linear probes, not linear probes, yet could validate this representation with causal interventions. However, follow-up work [201] showed that there was in fact a linear representation of the board state, but that as the model predicted both black and white moves, it was in terms of whether a cell had color equal to the current player or to the current opponent. We take results like this as promising additional evidence that the features-as-directions hypothesis has real predictive power.

A.1.14 What are possible conceptual models for what MLP layers are actually doing?

Perhaps the main model, as hypothesized and studied in [107], is that of a key-value store (note that the key and value terms here are unrelated to those used in the attention layers). A key-value store can be used in tasks such as memorization and factual recall, where the model wants to look up a fact, such as the location of the Eiffel Tower. If the residual stream contains both the features "Eiffel" and "Tower," then a neuron's key could be the sum of both of those features, and the value could be the feature indicating that the current token is in Paris.

The non-linearity of the MLP might serve several possible roles, such as thresholding, where only if the "Eiffel" and "Tower" features are present, do we conclude that it is in Paris. It may also perform **saturation**: for features where the model must accumulate many small pieces of evidence but the underlying ground truth is bimodal, such as "is Python code", the model may want to have the same thresholding . Additionally, some form of translation or memory management could be necessary. If features like "is in Paris" are only used in middle layers, it would not be in the model's interests to produce the "is in Paris" feature earlier on for fear of causing interference. Another possible role is that of activation range management. More broadly, neurons seem well-suited to representing Boolean functions such as AND, which is inherent in studying simple structures such as bigrams and trigrams. An AND can be implemented by stating that if the features A and B are both present with size 1, then only if their sum is present with size 2 does the neuron fire, and otherwise it is set

to zero (e.g., $x, y \to \text{ReLU}(x + y - 1)$).

Another role that neurons might play is that of disambiguation. There are certain tokens which occur in very different contexts, but where the same token arises, such as "Die" in English corresponding to "death," "dice," or "Die" as a common token in languages such as German or Dutch or Afrikaans. Neurons that disambiguate "Die" have been observed in the literature [73, 97]. In some sense, this can be thought of as having a more general "this is in Dutch" neuron producing an "is in Dutch" feature, and having the neuron having the "Die in Dutch" feature be the linear combination of the "is in Dutch" feature and the "Die" token is present feature. Depending on perspective, this could be considered a superposed, or distributed representation of the "Die in Dutch" feature.

One useful model for early layer MLPs is **detokenization** [97]: taking the raw input of tokens and converting it to more useful semantic concepts, a la the n-gram detectors we study. These are analogous to sensory neurons in biological systems, or gabor filters and high-low frequency detectors of early vision layers [187, 213]: neurons that take the raw sensory input and convert it to a more useful form. A non-obvious consequence of this is that detokenization neurons may *simultaneously* serve as key-value stores. An "Eiffel Tower" n-gram detector may have an "is in Paris" feature that it outputs, in addition to the "is Eiffel Tower" feature. From another perspective, assuming the representations are linear, it could be the case that the "is in Paris" feature is nothing more than the average of features like "is Eiffel Tower", "is the Louvre", etc.

The converse model for late layer MLPs is **retokenization** [97]: converting the features represented in some semantic space into the concrete output tokens, analogous to motor neurons. This is likely to be particularly important with multi-token words: if the model completes a prompt like "A famous landmark in Paris is" with " Eiffel Tower", it must in fact output 4 separate tokens (" E","iff","el"," Tower"), and this is natural to be performed by late layer MLPs on the current and subsequent tokens. Though, a notable difficulty in detecting these neurons is distinguishing between neurons converting an "output Eiffel Tower" feature on the " is" token to concrete outputs, from neurons that detect and continue the n-gram of " Eiffel Tower", but only figure it out after the " E" and "iff". This is a similar to difficulties in evaluating model performance when predicting multi-token outputs.

One interesting application of neurons in transformers that does not occur with simpler models such as convolutional networks is neurons engaging with and enhancing the function of attention heads. Models with a single attention-only layer can exhibit functions such as skip trigrams, but where this has bugs [96], a neuron could be used to fix this. They could also compose with important attention head circuits, such as induction heads, to help clarify edge cases. Induction heads detect and continue repeated text, but if a token arises in

multiple different contexts in the prior text with different subsequent tokens, we may want the induction heads to not activate at all. This may be easier with clarifying neurons than with an attention-focused circuit.

Many of our intuitions here involving thinking of GELU as essentially the same activation as a ReLU, while we know that GELUs perform better in practic e[130]. Though this may be down to ease of optimisation, we speculate that the "bowl" of the GELU, in particular the fact that it is close to a quadratic when close to the original, may allow the model to express more complex non-linear functions. We would be particularly excited to see future work exhibiting such case studies.

A.1.15 Did you train your probes on pre-GELU or post-GELU activations?

We train our probes on the post-GELU neuron activations (i.e., $\text{GELU}(W_{in}x)$) because there's a linear map from these to residual stream, so linear combinations of them are meaningful to the network, in a way that pre-GELU linear combinations are not. We often choose to plot histograms of pre-GELU activations for individual neurons for clarity, to better show the negative range, and to avoid an enormous spike around zero from all highly negative activations.

A.1.16 Did you have any negative results?

Yes. We had two types of feature datasets which did not appear to be sparsely represented: falsehoods in the counterfact dataset or the occurrence of particular suffixes and prefixes, though both attained fairly high accuracy with a dense probe. For prefixes and suffixes, since these are purely a property of single tokens, we believe these aren't likely to get dedicated neuron representation since the feature already exists within the token embedding.

For counterfact, a dataset of true and false factual completions, we hypothesized there might exist some dedicated "falsehood" neuron. However, sparse probes performed quite poorly, while increasing k continued to yield performance gains until about $k = 1024$ where we achieved 90% accuracy on the best layer. Our interpretation of this is that the probe is picking up some pattern of confusion or dissonance, associated with an unexpected completion, rather than a true falsehood or truth feature.

Our last negative result was in attempting to causally implicate the factual neurons. That is, run ablations that degrade the performance of few shot prompting based classification of people's gender, occupation, and whether they are alive or not. However, ablating individual neurons seemed to have very limited effect, indicating that either the "fact" was computed

previously, and these neurons are simply responding to this fact, or that there is some amount of ensembling or redundancy [80] that is robust to the deletion of a single neuron.

A.1.17 What do you mean by the model "wants"?

Throughout this FAQ, we sometimes use anthropomorphic language, such as that the model "wants" to minimise interference. We do not, of course, think that these small models are capable of having wants or intentionality in the sense of a human! In fact, it would be more accurate to say that stochastic gradient descent selects for models with low expected loss, and that model weights that minimise interference seem likely to have lower expected loss. But we believe that notions like "want" can be valuable intuition pumps to reason about what algorithms may be expressed in a model's weights, so long as the reader is careful to keep in mind the limits and potential illusions of anthropomorphism.

A.2 Experimental Details

A.2.1 Models

We study the Pythia model suited [37] which was trained on The Pile [104], with each model trained for the same number of steps with an identical data ordering. These models follow a fairly standard architecture, but use parallel attention and rotary positional embeddings. See Table A.1 for the architectural parameters of each model studied.

Model	n_{layers}	d_{model}	n_{heads}	d_{head}	Total Number of Neurons
Pythia 70M	6	512	8	64	12288
Pythia 160M	12	768	12	64	36864
Pythia 410M	24	1024	16	64	98304
Pythia 1B	16	2048	8	256	131072
Pythia 1.4B	24	2048	16	128	196608
Pythia 2.8B	32	2560	32	80	327680
Pythia 6.9B	32	4096	32	128	524288

Table A.1: Hyperparameters of Pythia models studied.

A.2.2 Datasets

Table A.2 contains summary statistics of all of our probing datasets in addition to the full list of features in each. We briefly describe the design, preprocessing, and motivation of each

one.

Dataset	Sequences	n_{ctx}	Non-pad tokens	Source	Pos ratio	Total Features
part-of-speech	1438	512	281044	EWT	0.20	16
dependencies	1438	512	281044	EWT	0.20	29
morphology	1438	512	281044	EWT	0.20	22
code language	5397	512	2757867	pile-github	0.11	9
natural language	28084	512	14350924	pile-europarl	0.11	9
text features	10000	256	2248714	pile-test-all	0.26	11
data subset	8413	512	4299043	pile-test-all	0.11	9
compound words	167959	24	4031016	pile-test-all	0.20	21
latex	4486	1024	4589178	pile-arxiv	0.26	14
wiki sex or gender	6000	128	688085	pile-test-all	0.50	2
wiki is alive	6000	128	688179	pile-test-all	0.50	2
wiki occupation	6000	128	677512	pile-test-all	0.17	6
wiki athlete	5000	128	575350	pile-test-all	0.20	5
wiki political party	3000	128	337755	pile-test-all	0.50	2

Table A.2: Summary statistics of probing datasets studied.

Natural Language We used both the raw text and the labels from the EuroParl subset of the pile, which contain a large number of parliamentary proceedings in many different languages. Many documents were quite a bit longer than our context length, so for each, we took a random contiguous sequence from the document. This choice was also made to minimize any context clues from the beginning of the document about the country speaking. For probing, we performed mean-aggregation. That is, computing the activation for all non-padding tokens in the sequence and using the average neuron activations as the representation for the sequence to be used for probing. One problem with this dataset is that all of the sequences are similarly styled, and about similar topics, hence our probes could be picking up on a particular style or topic feature rather than a true language feature.

Data Subset Similar to the above, we randomly selected several thousand sequences from each data subset of the Pile test set, where the labels were implied. We also used the same random sub-sequencing and mean aggregation.

Programming Languages For our programming language dataset, we took all of the github subset from the Pile test set. We then used a code recognition package to classify the type of code, and only include the source files with a prediction of over 90% confidence. A challenge with code is that it includes many tokens which aren't code (licenses, copywrite,

Dataset	Features
part of speech	AUX, ADP, VERB, ADJ, X, CCONJ, PROPN, NOUN, INTJ, SYM, PRON, DET, SCONJ, ADV, PUNC, NUM
dependencies	aux:pass, acl:relcl, nsubj, xcomp, flat, cc, mark, acl, ccomp, appos, root, nmod:poss, aux, amod, nsubj:pass, obj, obl, det, advmod, punct, parataxis, conj, case, list, advcl, cop, compound, nummod, nmod
morphology	eos_True, Person_2, Gender_Fem, VerbForm_Inf, PronType_Dem, Gender_Masc, first_eos_True, Gender_Neut, VerbForm_Part, NumType_Card, PronType_Int, PronType_Prs, Person_3, Tense_Past, Number_Plur, PronType_Art, Voice_Pass, PronType_Rel, VerbForm_Ger, Mood_Imp, Person_1, VerbForm_Fin
code language	Python, XML, Java, C++, HTML, C, Go, PHP, JavaScript
natural language	Swedish, Portuguese, German, English, French, Spanish, Greek, Italian, Dutch
text features	leading_capital, no_leading_space_and_loweralpha, all_digits, is_not_ascii, has_leading_space, contains_all_whitespace, all_capitals, is_not_alphanumeric, contains_whitespace, contains_capital, contains_digit
data subset	github, pubmed_abstracts, stack_exchange, wikipedia, freelaw, hackernews, arxiv, enron, uspto
compound words	mental-health, magnetic-field, trial-court, control-group, human-rights, north-america, clinical-trials, high-school, third-party, public-health, cell-lines, living-room, second-derivative, credit-card, social-media, prime-factors, federal-government, social-security, blood-pressure, gene-expression, side-effects
latex	is_superscript, is_inline_math, is_title, is_subscript, is_reference, is_denominator, is_author, is_numerator, is_display_math, is_math, is_abstract, is_frac
wikidata sex or gender	is_female, is_male
wikidata is alive	true, false
wikidata occupation	is_actor, is_athlete, is_journalist, is_politician, is_researcher, is_singer
wikidata athlete	is_american_football_player, is_association_football_player, is_baseball_player, is_basketball_player, is_ice_hockey_player
wikidata political party	is_democratic_party, is_republican_party

Table A.3: All feature within the feature collections.

comments, etc.). As a very coarse cleaning attempt, we ignore the first 50 tokens in the sequence when applying mean aggregation, to avoid beginning of file boilerplate. However, for some languages like HTML or XML that likely contain substantial plain text, this does little to help (and hence we see the lowest accuracy on HTML and XML).

Compound Words For our compound words dataset, we computed the top thousand alphabetical bigrams `XY` over the pile test set. We then filtered for bigrams where $P(X = x|Y = y)$ and $P(Y = y|X = x)$ were below 0.3 to make the prediction task less trivial. We then manually filtered to bigrams with the property that the first and second word together means something quite distinct from either word separately. For the probing dataset, we then included 2000 short (24 tokens) sequences from The pile ending with the bigram `XY`, 4000 examples ending with `XW`, `W`$\neq$`Y`, and 4000 examples ending with `ZY`, `Z`$\neq$`X`. We then probe on the activations of the last token of every sequence.

Latex Features For our latex feature dataset, we included the first 1024 tokens of all documents from the ArXiV subset of the Pile test set which contained at least 1024 tokens. We mostly relied on regular expressions to extract features, though used a simple finite state machine to parse the math text. For each of the features involving any sort of "containment," (which is most of them), we used the policy that all tokens within the containment counted as being part of the feature, but any tokens defining the containment for not. For instance for `is_subscript`, the "`_{`" token is not included, but anything within the braces is. For our positive class, we randomly sample tokens from the set of all positive tokens. For features with a natural complement (e.g., `is_numerator` and `is_denominator`) we make the negative class be occupied by half of tokens within the complement, and half random tokens.

Text Features For our plain-text feature dataset, we similarly extracted features using regular expressions, but just applied to the raw token strings. Therefore, for our text features, a feature could be correctly predicted simply by partitioning the tokens into two classes, with no additional context. We sampled tokens randomly from a subset of the Pile test set.

Linguistic Features For our linguistic features, we use the text and labels from the well known Penn Treebank Corpus. The dataset is provided at a sentence level, so we first merge sentences from the same documents, and then perform a token alignment and character mapping to adapt the dataset labels to the specific tokenization of the Pythia models. For features that apply to all tokens (like part-of-speech or dependency relations), we have no restrictions on the tokens sampled for the negative class. For features restricted to tokens of

specific types (e.g. verb tense), we restrict the negative examples to be from the same token class (e.g. only verbs).

Wikidata Features To create a Wikidata feature dataset for a specific property (e.g. gender, occupation), we search through a JSON dump of all Wikidata entities and compile a table containing the names and relevant property value for each person with data on the chosen property. When there are multiple Wikidata entities with the same name, we choose the entity with more total Wikidata properties defined as a heuristic for relevance. We then search the text of the test set of The Pile for instances of these names and filter out all examples with names that occur only once in the dataset. We tokenize each matching string and truncate to 128 tokens such that the tokenized name is at the end (if there are sufficient preceding tokens, otherwise we pad the tokenized string to length 128). We further filter the tokenized examples to ensure that each name occurs no more than three times in each feature dataset. This helps us find general knowledge neurons that aren't overly impacted by individual people. Then when probing, we consider only the activations for the tokens of the surname for each example to reduce the effect of correlations between first names and certain properties (such as gender).

A.2.3 Feature Selection Methods

Throughout, assume we have a dataset of token activations for n tokens and d neurons $X \in \mathbb{R}^{n \times d}$, and a vector of labels $y = \{-1, 1\}^n$. We refer to the set of tokens in the positive class P and the set of tokens in the negative class N. Below we describe the different feature selection methods in more detail. With the exception of optimal sparse probing and adaptive thresholding, we can think of these approaches as scoring algorithms that can be used to rank the different neurons by importance. Then, to get a k sparse classifier, in all cases one just selects the k neurons with highest score s.

Mean Difference The score is given by the average mean difference between classes for each neuron. That is, for neuron j we have $s_j = \frac{1}{|P|} \sum_{i \in P} X_{ij} - \frac{1}{|N|} \sum_{i \in N} X_{ij}$.

Mutual Information We use the algorithm presented by [240], to compute the mutual information between continuous data (in this case each neuron independently) and a discrete target relying on a nearest-neighbor approximation. The score for each neuron is then just the estimated mutual information.

L_1 regularized We train a dense logistic regression probe on the activations with L_1 regularization. The score for each neuron is the absolute value of the corresponding coefficient of the dense probe.

F-statistic We utilize a one-way ANOVA test to evaluate the relationship between the continuous data of each neuron independently and a discrete target. The F-statistic is calculated by partitioning the total variability into between-group and within-group components, represented by sum of squares between groups (SSB) and sum of squares within groups (SSW), respectively. The ratio of the mean square between groups (MSB) to the mean square within groups (MSW) is computed as the F-statistic. A higher F-statistic indicates a larger discrepancy between group means and is used as the score for each neuron.

Optimal Sparse Probing We train a cardinality-constrained support vector machine (SVM) with hinge-loss trained to provable optimality using the cutting planes technique as described in [36]. The selected coefficients are simply those selected by the classifier.

Adaptive Thresholding Starting from $k = d_{mlp}$ (or some large k on a filtered set of neurons from a different heuristic), we train a series of logistic regression probes with elasticnet regularization (that is combined l_1 and l_2 regularization). Given a schedule for reducing k, at each step t, we take the $k_t < k_{t-1}$ neurons with highest absolute coefficient magnitude and retrain a logistic regression probe on these neurons. This process enables us to sweep through many values of k while leveraging past computation to achieve better feature selection.

A.2.4 Experimental Procedure

Here we provide additional details on the experimental procedure regarding both the feature selection experiments and the sparsity with scale experiments within Section 4.

Perhaps most significantly, for all experiments we preceded probe training with a heuristic filtering step, where we only trained probes on the top 1024 neurons, as judged by mean difference, for each combination of layer, feature, and model. This was done for computational reasons, as computing the mutual information, training dense probes, and optimal sparse probing methods are very expensive to run on hundreds of features for many layers containing in excess of 10,000 neurons. For optimal sparse probing, we only used the top 50, and set a timeout of 60 seconds.

All classifiers were trained with balanced class weights to account for class imbalances in the dataset. Finally, we selected hyperparameters using a small subset of feature and model

Method	Sparsity								Runtime (s)
	1	2	3	4	5	6	7	8	
Random	0.573	0.629	0.663	0.690	0.712	0.729	0.743	0.755	0.257
MMD	**0.832**	0.857	0.867	0.874	0.880	0.884	0.887	0.890	0.292
FS	**0.832**	0.856	0.867	0.874	0.880	0.884	0.887	0.890	0.303
LR	0.818	0.853	0.871	0.882	0.890	**0.897**	**0.902**	**0.906**	7.796
MI	0.831	0.851	0.863	0.869	0.876	0.880	0.884	0.886	32.241
OSP	0.831	**0.861**	**0.876**	**0.884**	**0.891**	0.896	0.899	0.902	53.028

Table A.4: Comparison of sparse feature selection methods. Results are for the out-of-sample F1 score averaged for the best scoring layer for each combination of model and feature.

combinations, and then used the best performing hyperparameters for all other experiments, as doing seperate tuning for every trial would be too computationally expensive.

A.2.5 Feature Selection Results

For each combination of model, feature, and layer described above, we train a sparse probe for values of $k = 1, \ldots, 8$ using the features selected by the aforementioned subset selection methods. In Table A.4, we report the out-of-sample F1 score of each method averaged across models and features, and for the maximum across layers since features tend to be only represented in specific layers.

While our results are averaging over many trials, several points stand out. The top methods are always within 1% of each other, with no method strictly dominating the others. Although OSP comes with optimality guarantees, we set a one minute timeout, which is frequently reached especially for larger values of k. Moreover, this provable optimality is with respect to the specific level of l_2 regularization which often needs to be turned fairly high to speed up convergence [36]. Nevertheless, it is impressive that such fast and simple heuristics as the mean difference are comparable to far more sophisticated approaches, with even random neurons containing substantial information of the target task. This suggests that perhaps the best design is a two-stage approach where very fast heuristics are used to find relevant neurons and layers, and optimal methods are used to improve and verify the solution.

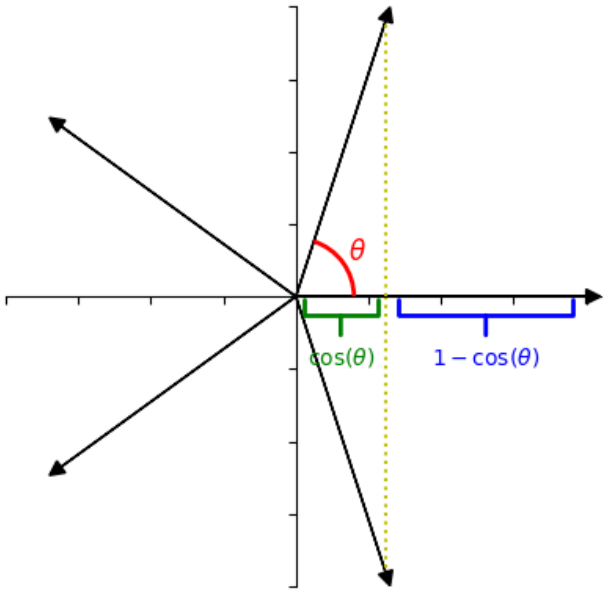


Figure A.1: Intuition for representing 5 features in 2 dimensions in superposition.

A.3 Superposition Construction

Consider a set of n binary and mutually exclusive features (i.e., one-hot vectors like n-grams), that we want to embed in two dimensions, and later losslessly recover. Specifically, given a one hot vector $x \in \mathbb{R}^n$, we seek a $W \in \mathbb{R}^{n \times 2}$ such that

$$x = \text{ReLU}(WW^T x + b)$$

where $\text{ReLU}(x) = \max(x, 0)$. By embedding each point to be equally spaced around a circle of radius α (i.e., $w_i = \alpha \cos(2\pi i/n), \alpha \sin(2\pi i/n)]$, we get that

$$(WW^T e_i)_j = \alpha^2 \cos\left(2\pi \frac{|i-j|}{n}\right)$$

where $\alpha = \|w_i\|_2$. To recover the initial one-hot representation, we require

$$\alpha^2 \left(1 - \cos\left(\frac{2\pi}{n}\right)\right) = 1 \quad \text{and} \quad b = -\alpha^2 \cos\left(\frac{2\pi}{n}\right)$$

Analyzing our proxy metric, $b\|w\|_2 = \frac{\cos(2\pi/n)}{(cos(2\pi/n)-1)}$ monotonically decreases for n with $n > 2$.

See [98] for a far more extensive set of experiments and results on related topics.

Note that this construction refers to residual stream superposition and *not* neuron super-

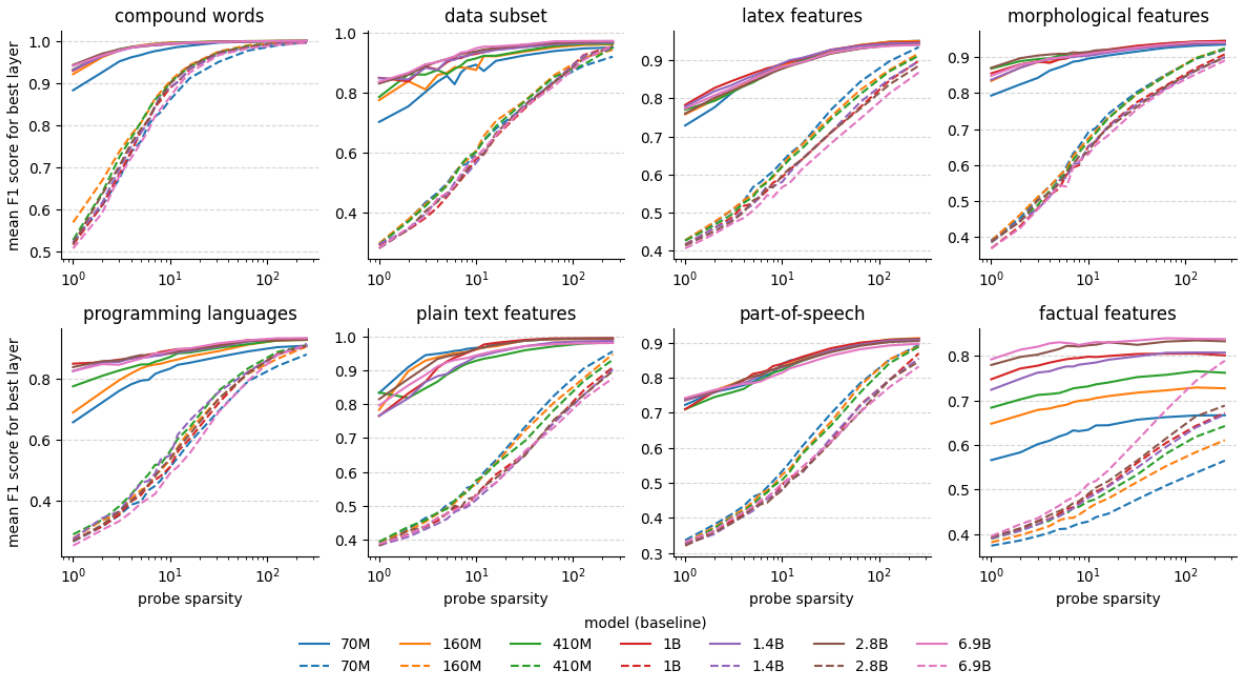


Figure A.2: Difference between a k-sparse classifier using sparse features selection versus a random baseline. Specifically, the F1 score of training a classifier with k coefficients on the full $n \times d$ activation dataset when multiplied by a random $d \times k$ orthogonal matrix.

position since we are assuming we have exactly as many neurons as features. However, we expect the same basic motif to clean up interference to hold for the case with fewer neurons than features. Moreover, it is plausible increased levels of residual stream superposition is associated with increased levels of neuron superposition. Hence we believe this construction, and therefore the presence of large weight norms and negative biases, to be suggestive evidence of neuron superposition.

Of course, it is possible factors other than superposition explain this observed weight pattern. Additionally, there might be multiple mechanisms for computing features in superposition (likely those that are not as binary or mutually exclusive as n-grams) that do not strongly affect the distributions of weights and biases. We would be very excited about future work that more carefully explained the source of these weight statistics, or that studied other concrete mechanisms or motifs enabling effective computation in superposition.

A.4 Additional Results

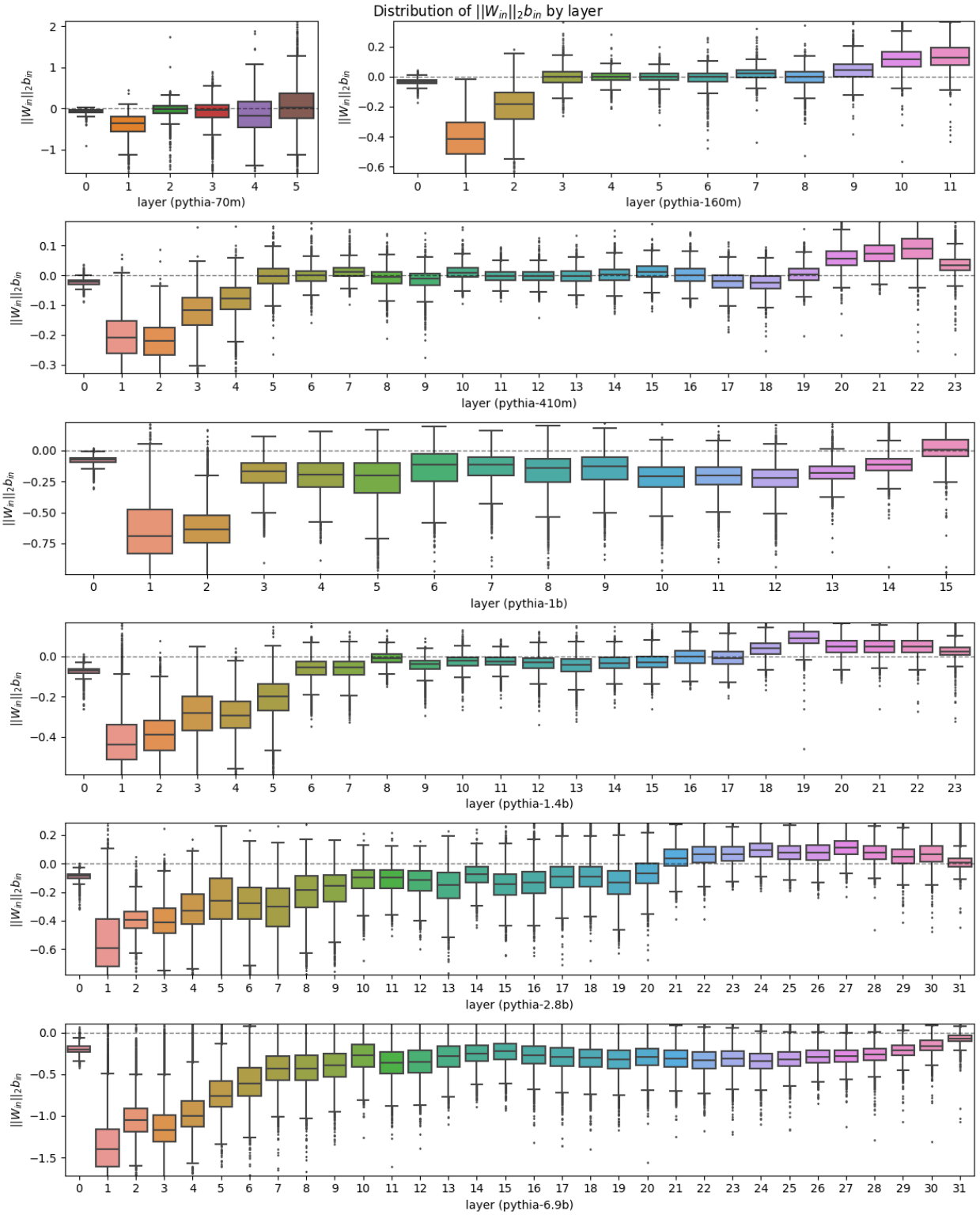


Figure A.3: Distribution of input weight norms and biases for all Pythia models studied.

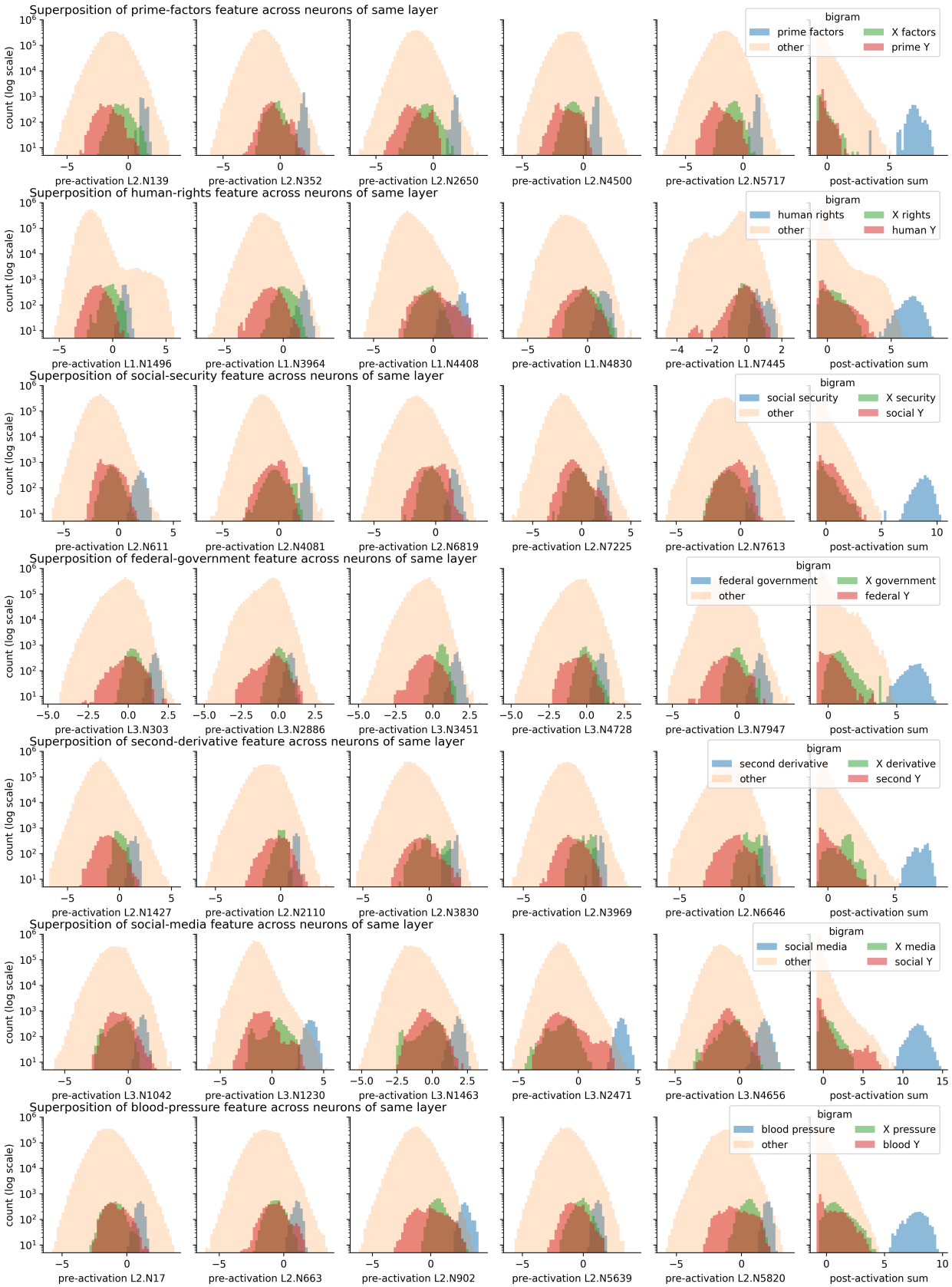


Figure A.4: More examples of superposition of compound words in Pythia-1B

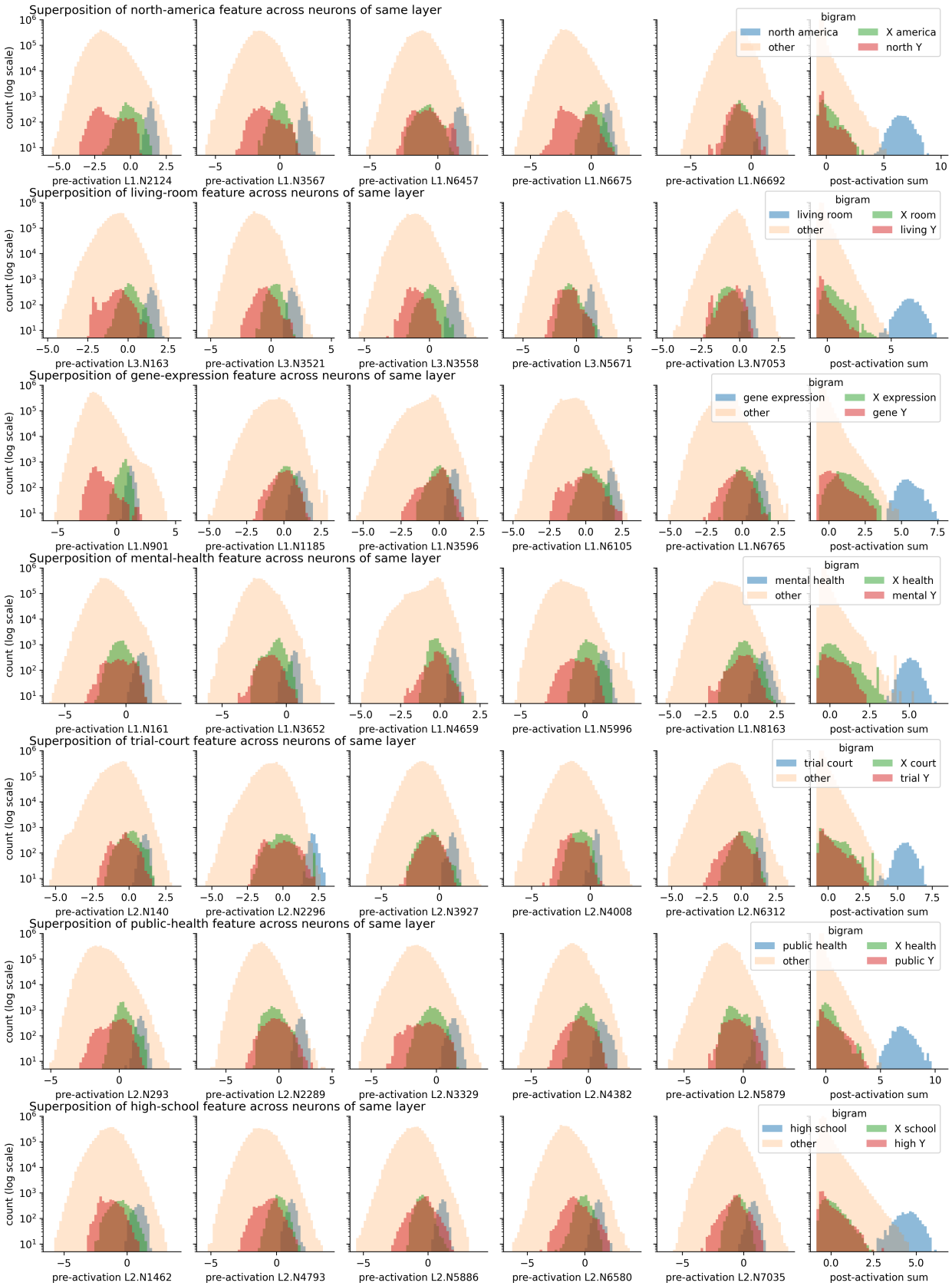


Figure A.5: More examples of superposition of compound words in Pythia-1B

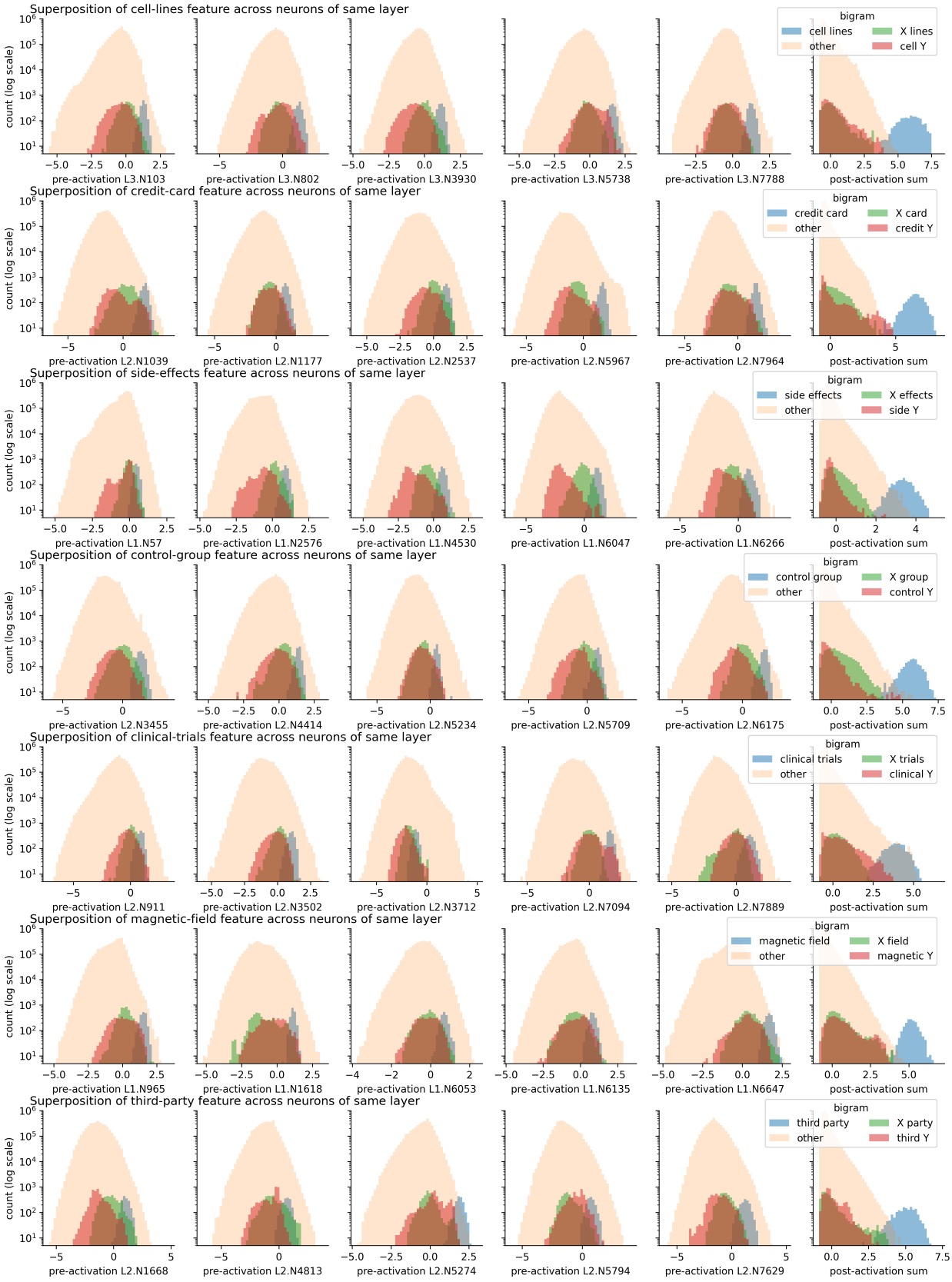


Figure A.6: More examples of superposition of compound words in Pythia-1B

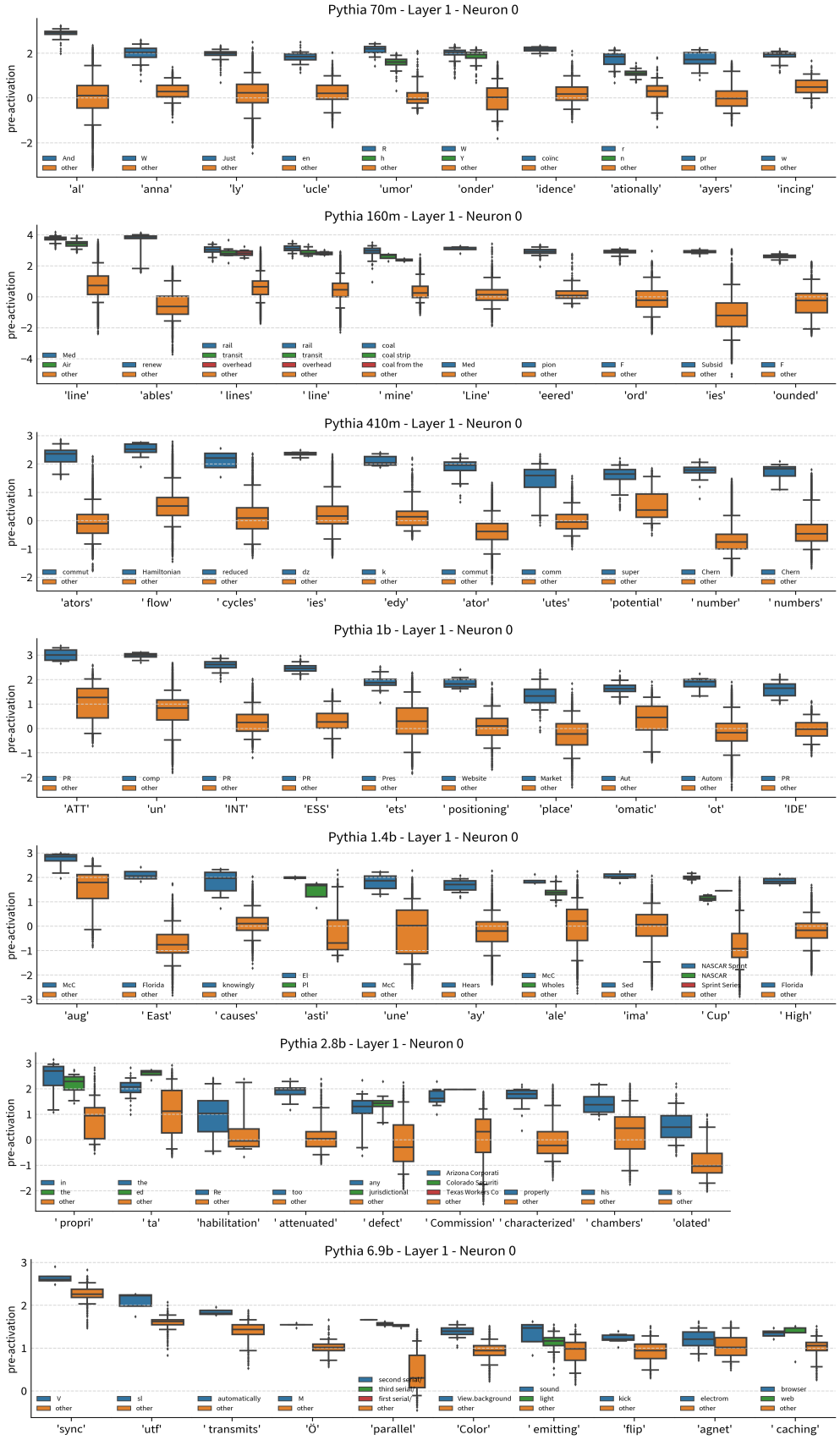


Figure A.7: More examples of n-gram polysemanticity in layer 1 neuron 0 of all Pythia models studied.

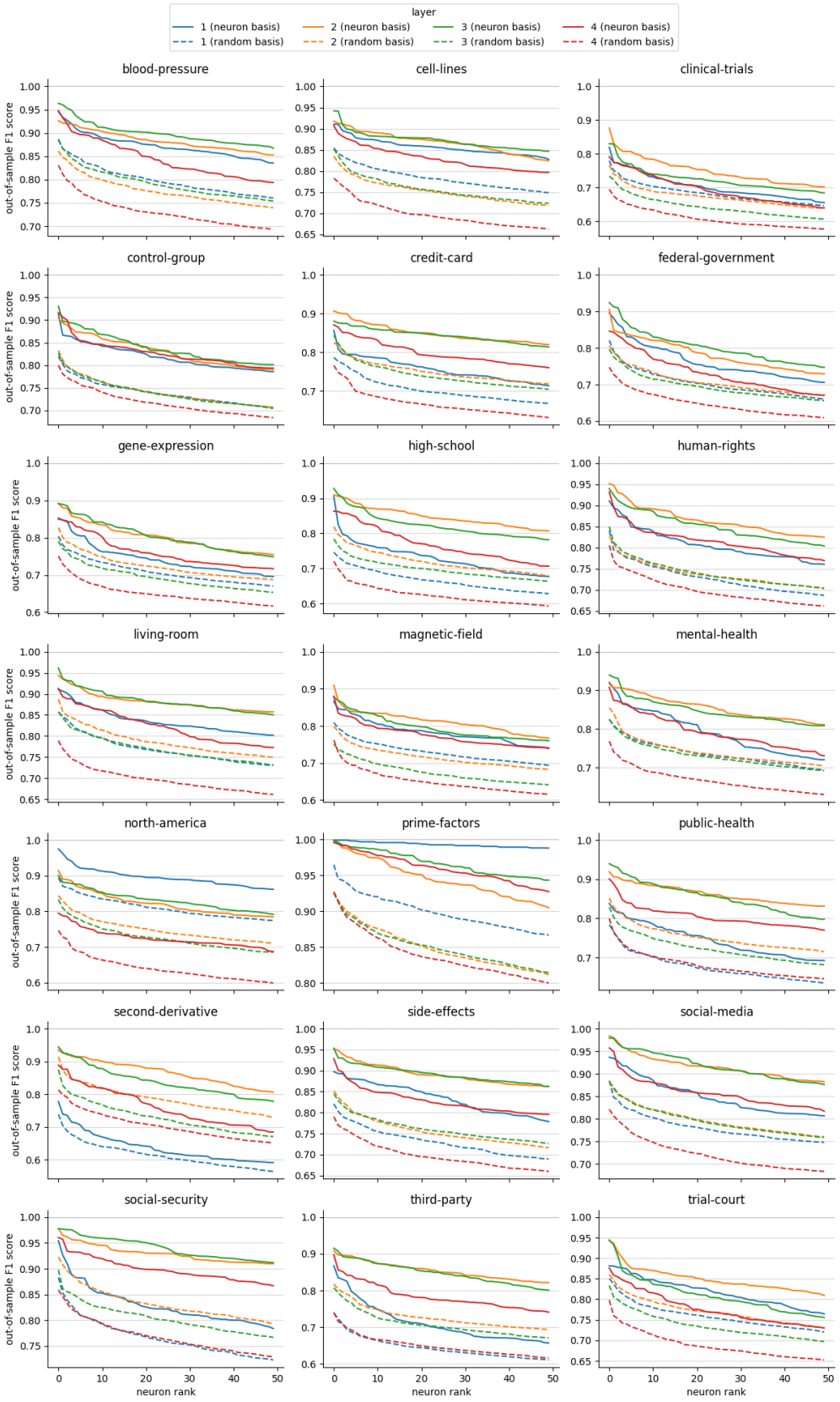
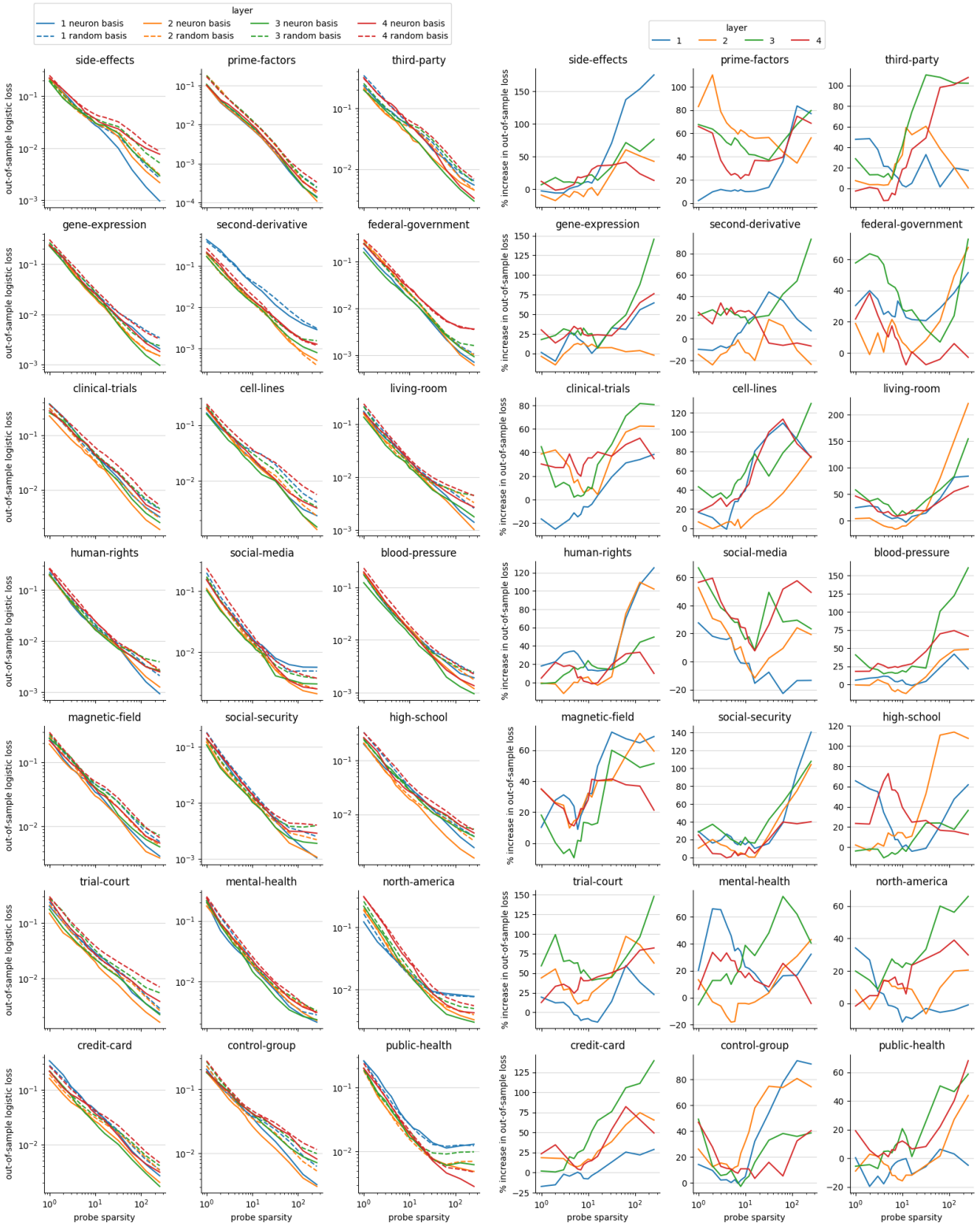


Figure A.8: Demonstration of neuron basis alignment in early layers of Pythia-1B. Compares the compound-word classification performance of the top 50 neurons for layers 1-4 in the standard neuron basis, as opposed to a random basis (the post-activation dataset multiplied by a random Gaussian $d \times d$ matrix).



(a) Logistic loss versus probe sparsity (b) Logistic loss increase versus probe sparsity

Figure A.9: Logistic loss for classifying the occurrence of a particular compound word in layers 1-4 of Pythia-1B (in both the neuron basis and the average of random bases) undergo power law scaling until hitting a break point (left). In general, probes in the neuron basis achieve lower overall loss, further suggesting the neuron basis is meaningful (right).

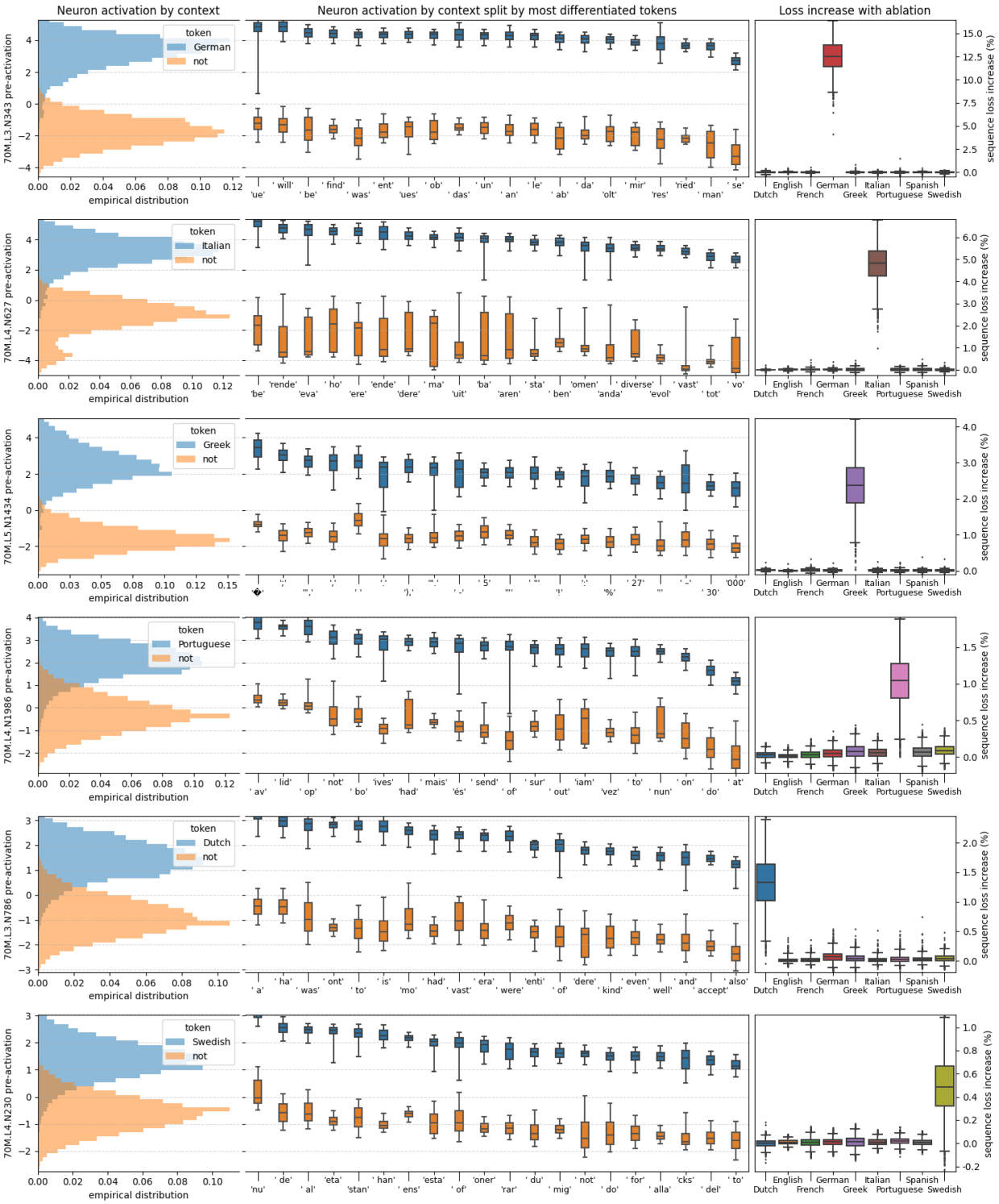


Figure A.10: Monosemantic language context neurons in Pythia-70M.

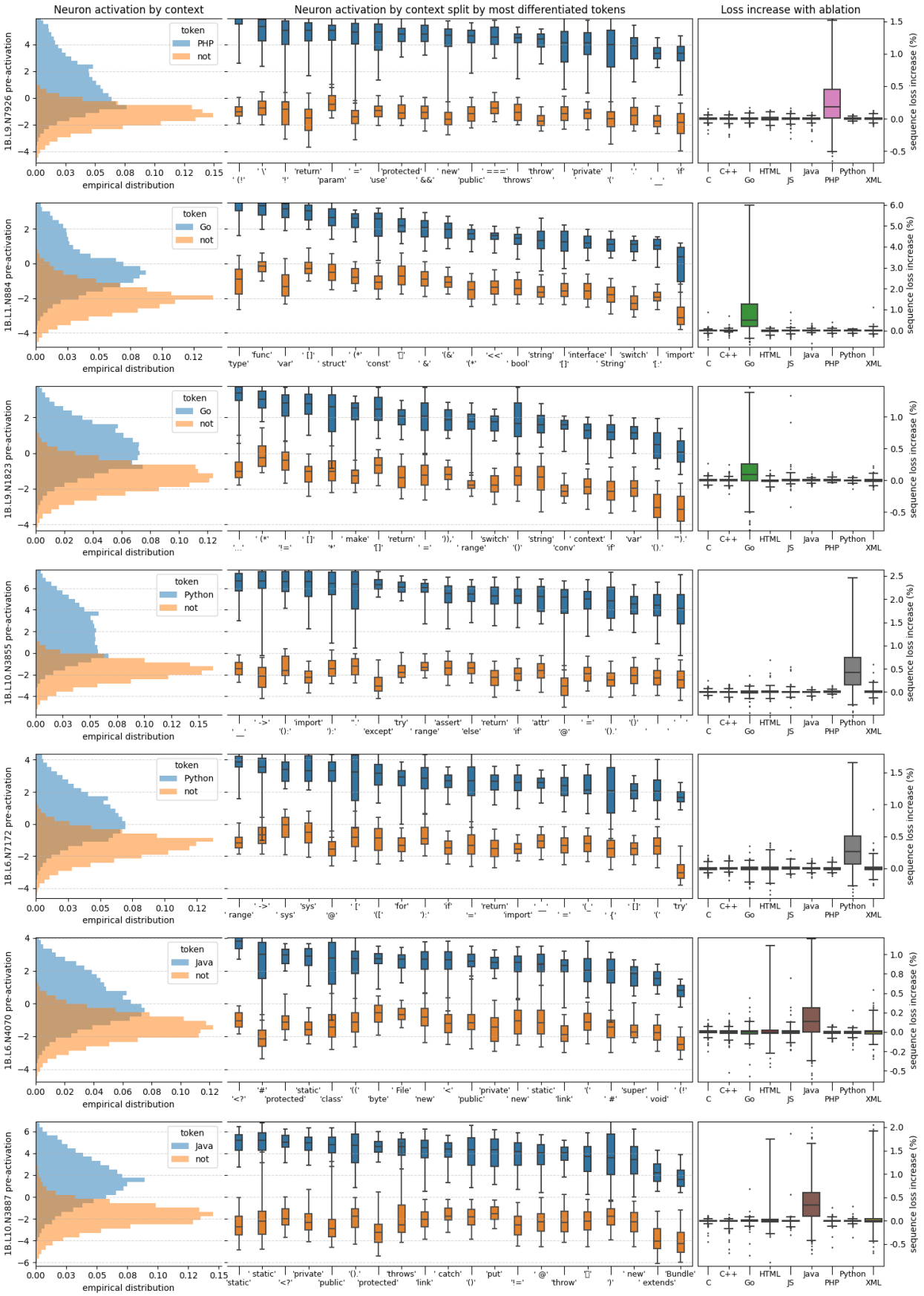


Figure A.11: Monosemantic code context neurons in Pythia-1B.

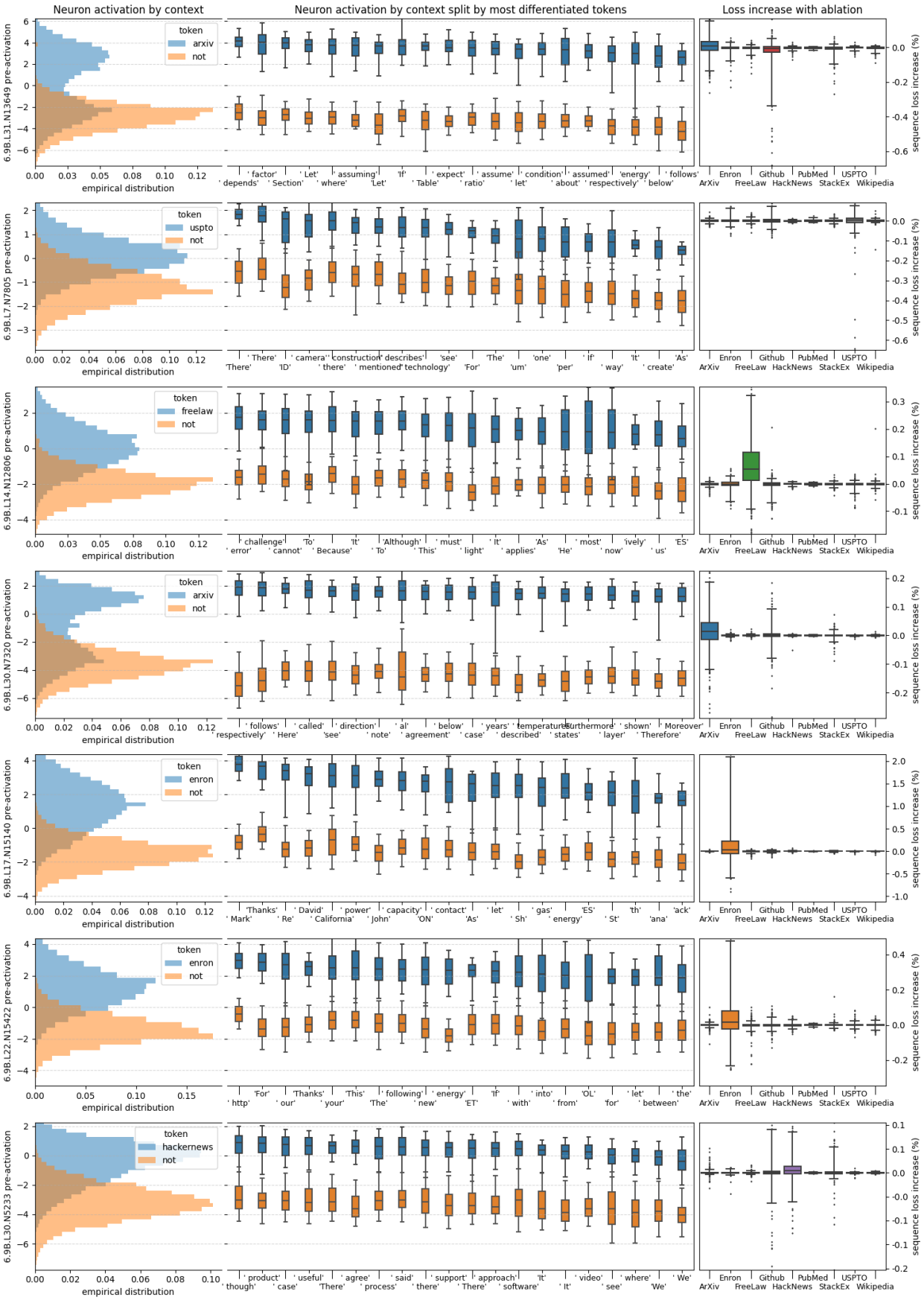


Figure A.12: Monosemantic distribution identification context neurons in Pythia-6.9B.

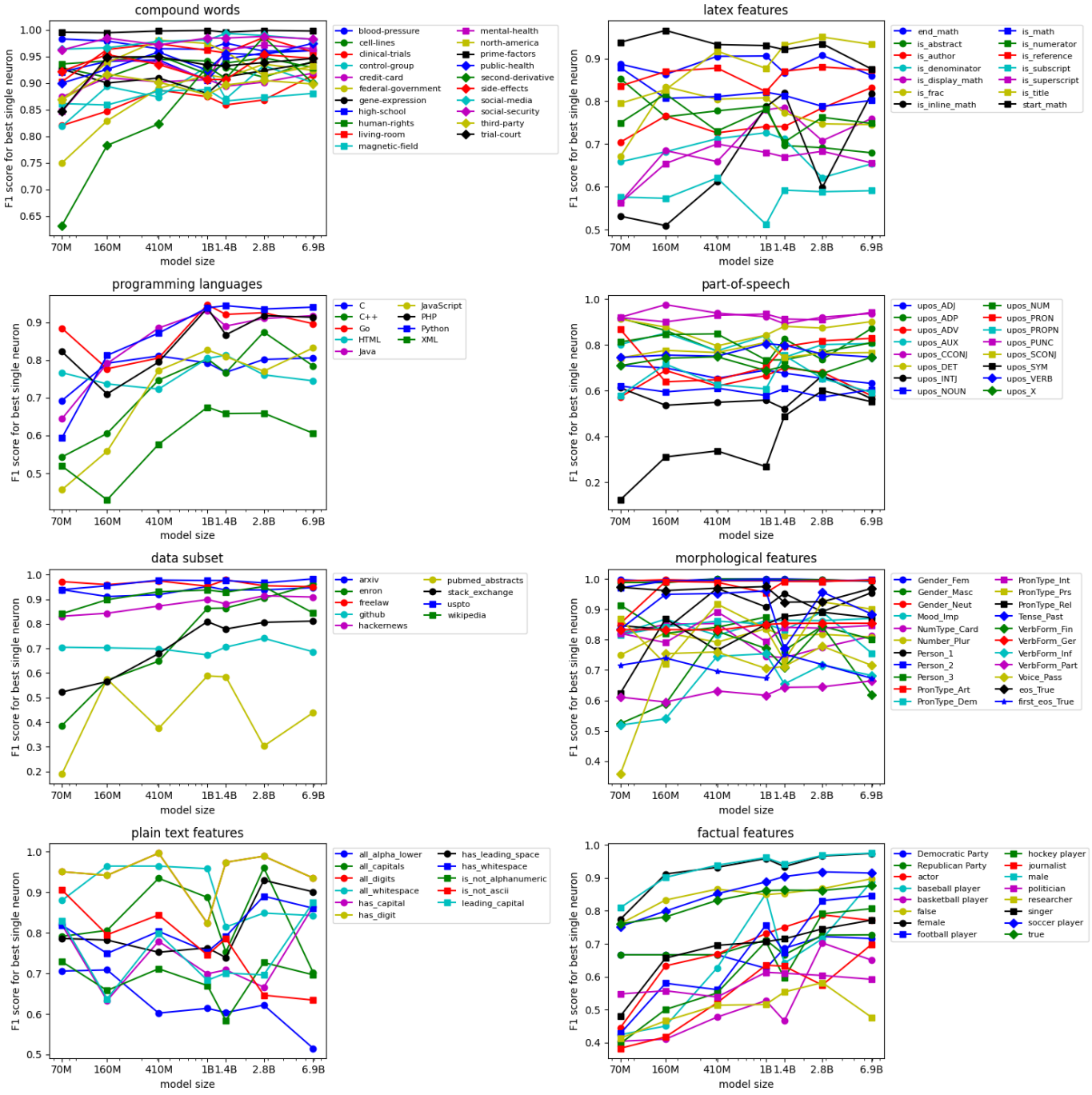


Figure A.13: The single neuron with highest F1 classification performance for each feature by model size.

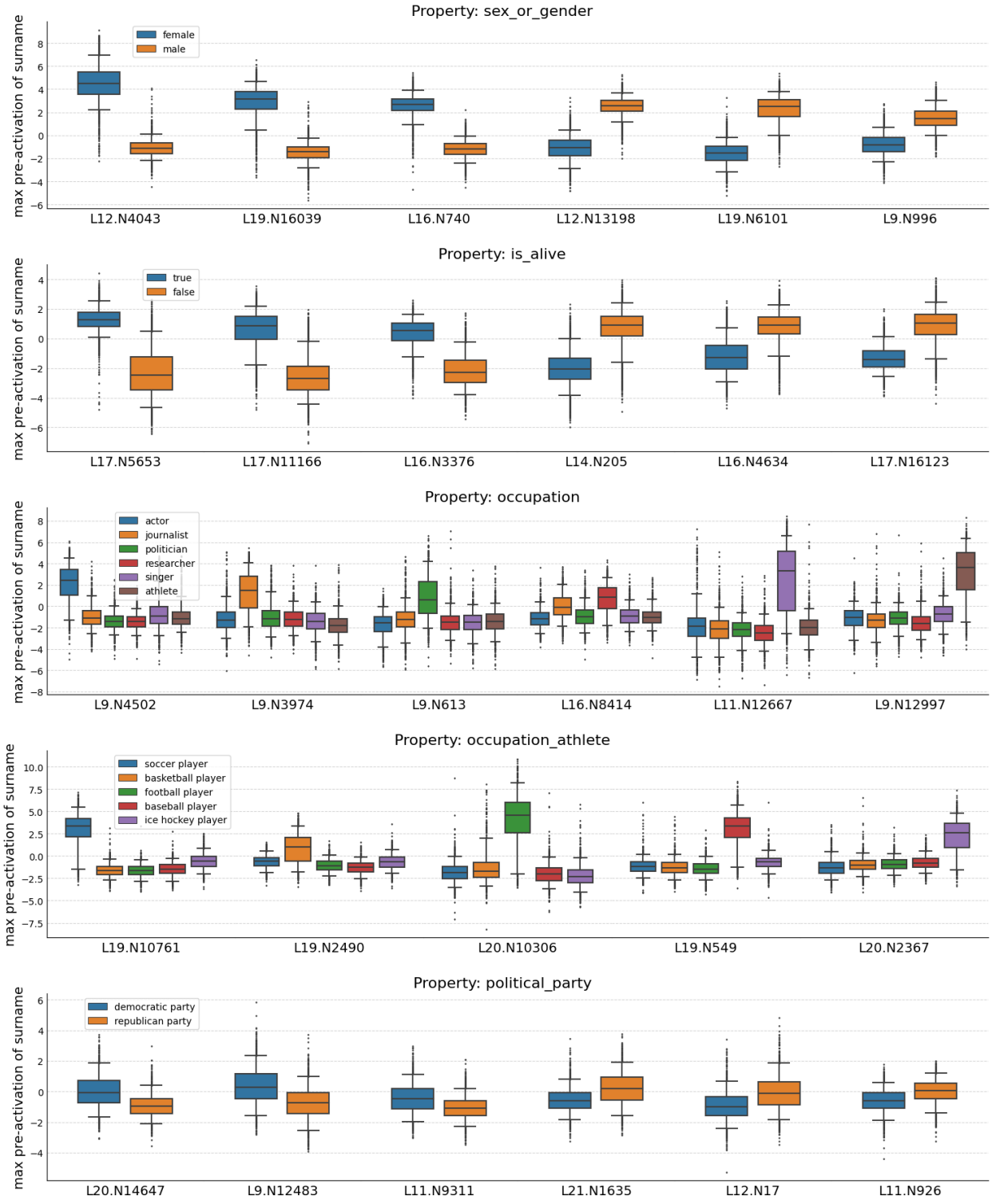


Figure A.14: Top neurons in Pythia-6.9B for each category of factual neuron.

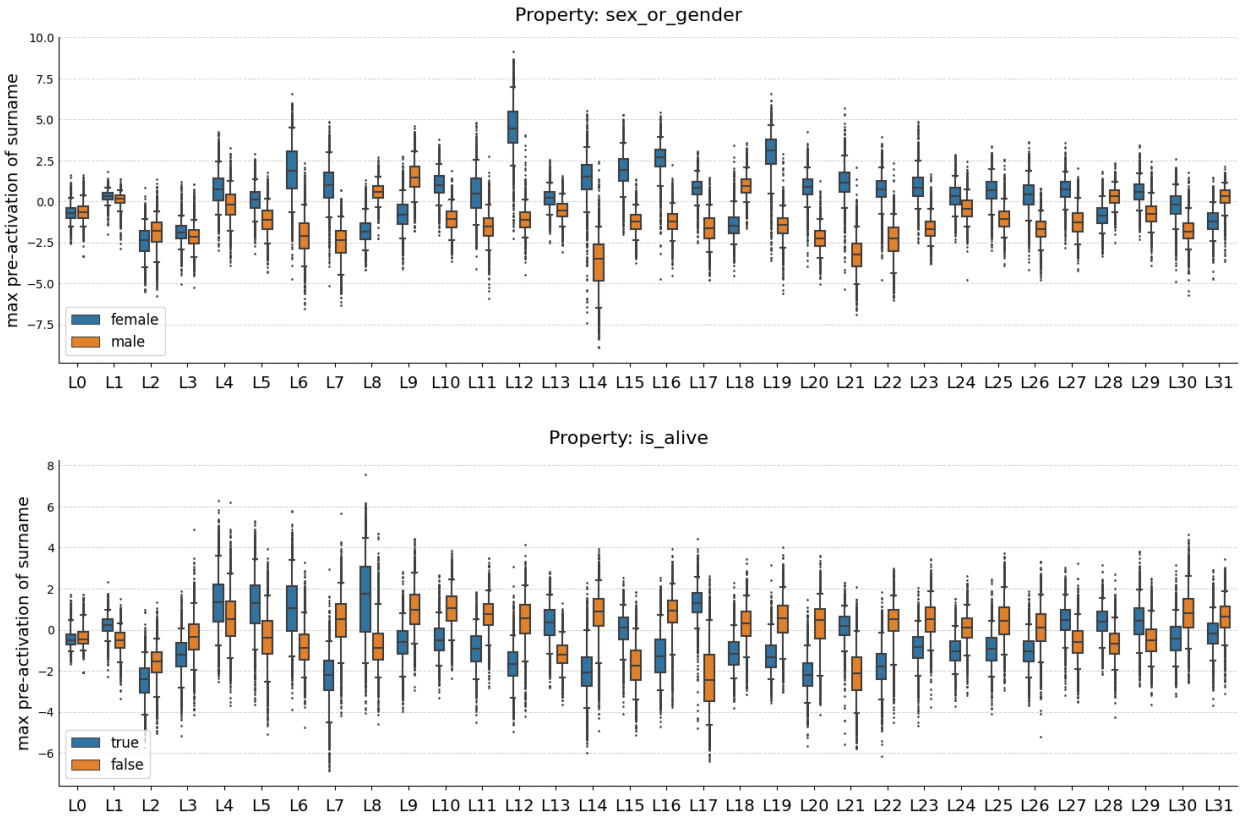


Figure A.15: Top neuron for each layer in Pythia-6.9B for the `sex_or_gender` and `is_alive` feature categories.

Appendix B

Universal Neurons Appendix

B.1 Additional Discussion

B.1.1 Why expect universal neurons to be more monosemantic?

The intuition for the connection comes from [98] which showed that important features (those that are constructed to have maximum effect on the loss) are the features which get their own dedicated neuron (making the neuron monosemantic). Since these GPT2 models are training on the same data, the important features should be similar across the networks. So if monosemantic neurons primarily represent important features, and the important features are shared (i.e., universal) across the networks, then we would expect the neurons which are universal across the networks to be representing the same set of important features.

Another line of argument is to consider the probability of a polysemantic neuron being universal. It is extremely unlikely that there would exist a pair of neurons which activates for the same $k \gg 0$ unrelated features, assuming unrelated features have roughly similar probabilities of being assigned to neurons. Moreover, this simple model suggests that as $k \to 1$ it is much more likely for there to exist a neuron which represents the same set of features.

B.1.2 What are these observations useful for?

Our motivation here is primarily to improve our understanding of models for interpretability sake, rather than make models more performant. Specifically, we think interpretability is largely an immature field, without much grounding theory. Therefore, we think it's inherently valuable to gain a lot of "empirical surface area" on real networks to constrain the hypothesis space and develop the theory and practice of interpretability.

Example insights from our paper that might help future interpretability researchers

- Single neuron ablations can be misleading if there is an ensemble of similar neurons or there are neurons which consistently cancel each other out.
- The depth specialization results and the prediction followed by suppression neuron transition emphasize the sequential and residual nature or model processing.
- Our results on entropy neurons exemplify a concrete case where it would not be valid to linearize layer norm.
- Our results on attention deactivation neurons show how certain features might be difficult to interpret, because they are features which take internal actions rather than represent external inputs.

B.2 Additional Empirical Details

All of our code and data is available at https://github.com/wesg52/universal-neurons.

Most of our plots in the main text (and therefore neuron indices) correspond to the HuggingFace model `stanford-crfm/arwen-gpt2-medium-x21` with our additional correlation experiments being conducted on `stanford-crfm/alias-gpt2-small-x21` and `EleutherAI/pythia-160m`.

B.2.1 Weight Preprocessing

We employ several standard weight preprocessing techniques to simplify calculations [199].

Folding in Layer Norm Most layer norm implementations also include trainable parameters $\boldsymbol{\gamma} \in \mathbb{R}^n$ and $\mathbf{b} \in \mathbb{R}^n$

$$\text{LayerNorm}(\mathbf{x}) = \frac{\mathbf{x} - \mathbb{E}(\mathbf{x})}{\sqrt{\text{Var}(\mathbf{x})}} * \boldsymbol{\gamma} + \mathbf{b}. \tag{B.1}$$

To account for these, we can "fold" the layer norm parameters in to W_{in} by observing that the layer norm parameters are equivalent to a linear layer, and then combine the adjacent linear layers. In particular, we can create effective weights

$$\mathbf{W}_{\text{eff}} = \mathbf{W}_{\text{in}}\ \mathbf{diag}(\boldsymbol{\gamma}) \qquad \mathbf{b}_{\text{eff}} = \mathbf{b}_{\text{in}} + \mathbf{W}_{\text{in}}\mathbf{b} \tag{B.2}$$

Finally, we can center the reading weights because the preceding layer norm projects out the all ones vector. Thus we can center the weights $\mathbf{W}_{\text{eff}}$ becomes

$$\mathbf{W}_{\text{eff}}^{'}(i,:) = \mathbf{W}_{\text{eff}}(i,:) - \bar{\mathbf{W}}_{\text{eff}}(i,:)$$

Writing Weight Centering Every time the model interacts with the residual stream it applies a LayerNorm first. Thus the components of $\mathbf{W}_{\text{out}}$ and $\mathbf{b}_{\text{out}}$ that lie along the all-ones direction of the residual stream have no effect on the model's calculation. So, we again mean-center $\mathbf{W}_{\text{out}}$ and $\mathbf{b}_{\text{out}}$ by subtracting the means of the columns of $\mathbf{W}_{\text{out}}$

$$\mathbf{W}_{\text{out}}^{'}(:,i) = \mathbf{W}_{\text{out}}(:,i) - \bar{\mathbf{W}}_{\text{out}}(:,i)$$

Unembed Centering Additionally, since softmax is translation invariant, we modify $\mathbf{W}_U$ into

$$\mathbf{W}_{\text{U}}^{'}(:,i) = \mathbf{W}_{\text{U}}(:,i) - \mathbf{w}_i$$

For both of theses, see the transformer lens documentation for more details.

The purpose of all of these translations is to remove irrelevant components and other parameterization degrees of freedom so that cosine similarities and other weight computations have mean 0.

B.2.2 Correlation Computations

We compute our correlations over a 100 million token subset of the Pile test set [104], tokenized to a context length of 512 tokens. We compute correlations over all tokens that are not padding, beginning-of-sequence, or new-line tokens.

Efficient Computation Because storing neuron activations for two models over 100M tokens would be 36 petabytes of data, we require a streaming algorithm. To do so, observe that given a pair of neuron activations $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ consisting of n pairs, the correlation can be computed as

$$\rho_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} = \frac{\sum_i x_i y_i - n\bar{x}\bar{y}}{\sqrt{\sum_i x_i^2 - n\bar{x}^2}\sqrt{\sum_i y_i^2 - n\bar{y}^2}}$$

where $\bar{x}, \bar{y}$ are the sample mean. Therefore, instead of saving all neuron activations, we can maintain four `n_neuron` dimensional vectors and one `n_neuron` $\times$ `n_neuron` matrix corresponding to the running neuron activation means in model A and model B, a running sum

of each neurons squared activation, and a running sum of pairwise products. At the end of the dataset, we perform the appropriate arithmetic to combine the results into pairwise correlations for all models.

B.2.3 Model Hyperparameters

Property	GPT-2 Small	GPT-2 Medium	Pythia 160M
layers	12	24	12
heads	12	16	12
d_{model}	768	1024	768
d_{vocab}	50257	50257	50304
d_{MLP}	3072	4096	3072
parameters	160M	410M	160M
context	1024	1024	2048
activation function	gelu_new	gelu_new	gelu
pos embeddings	absolute	absolute	RoPE
rotary percentage	N/A	N/A	25
precision	Float-32	Float-32	Float-16
dataset	OpenwebText	OpenwebText	Pile
p_{dropout}	0.1	0.1	0

Table B.1: Hyperparameters of models

B.3 Additional Mysteries

We conclude our investigation by commenting on several miscellaneous results that we think are worth reporting but that we do not fully understand.

B.3.1 Cosine and Activation Frequency

An unexpectedly strong relationship we observed is the correlation between activation frequency of a neuron and the cosine similarity between its input and output weight vectors $\cos(\mathbf{w}_{\text{in}}, \mathbf{w}_{\text{out}})$ as shown in Figure B.1. Almost all neurons with a very high activation frequency have input and output weights in almost opposite directions. These neurons are predominantly in the first quarter of network depth and have small excess correlation, i.e., they are not universal as measured by activation. We also find it noteworthy that there appears to be an approximate ceiling and floor on the cosine similarity of approximately ± 0.8.

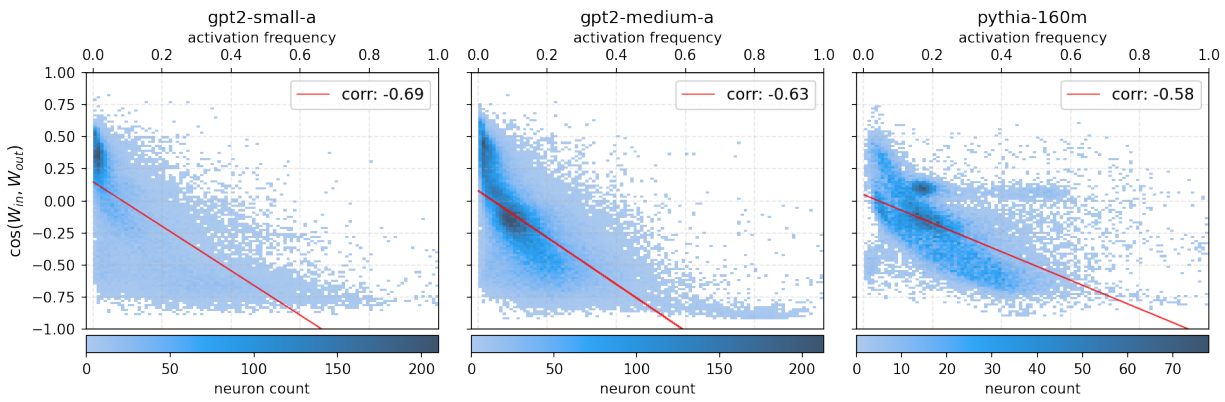


Figure B.1: Activation frequency of neuron (fraction of activation values greater than zero) versus cosine similarity of neuron input and output weights for GPT2-small-a (left), GPT2-medium-a (center), and Pythia-160M (right).

B.3.2 Duplication and Universality

While neuron redundancy has been observed in models before [65, 80] and large models can be effectively pruned [282], we were surprised by the number of seemingly duplicate universal neurons we observed (e.g., Figure B.8 or the 105 BOS neurons we observed). Naively, this is surprising, as it seems wasteful to dedicate multiple neurons to the same feature. Larger models have more capacity and are empirically much more effective so why have redundant neurons when you could instead have one neuron with twice the output weight norm?

A few potential explanations are (1) these models were trained with weight decay, creating an incentive to spread out the computation. (2) Dropout—however, in these models dropout is applied to the output of the MLP layer, rather than the MLP activations themselves. (3) These neurons are vestigial remnants that were useful earlier in training [225], but are potentially stuck in a local minima and are no longer useful. (4) The duplicated neurons are only activating the same on common features, but are polysemantic with different sets of rarer features. (5) Ensembling, where each neuron computes the same feature but with some independent error, and together form an ensemble with lower average error.

By measuring redundancy in terms of similarity in weights (Figure B.2), we find very few neurons which are literal duplicates, providing more evidence for (4) and (5). Based on the much higher level of similarity for universal neurons, it is possible this effect is relatively small in general.

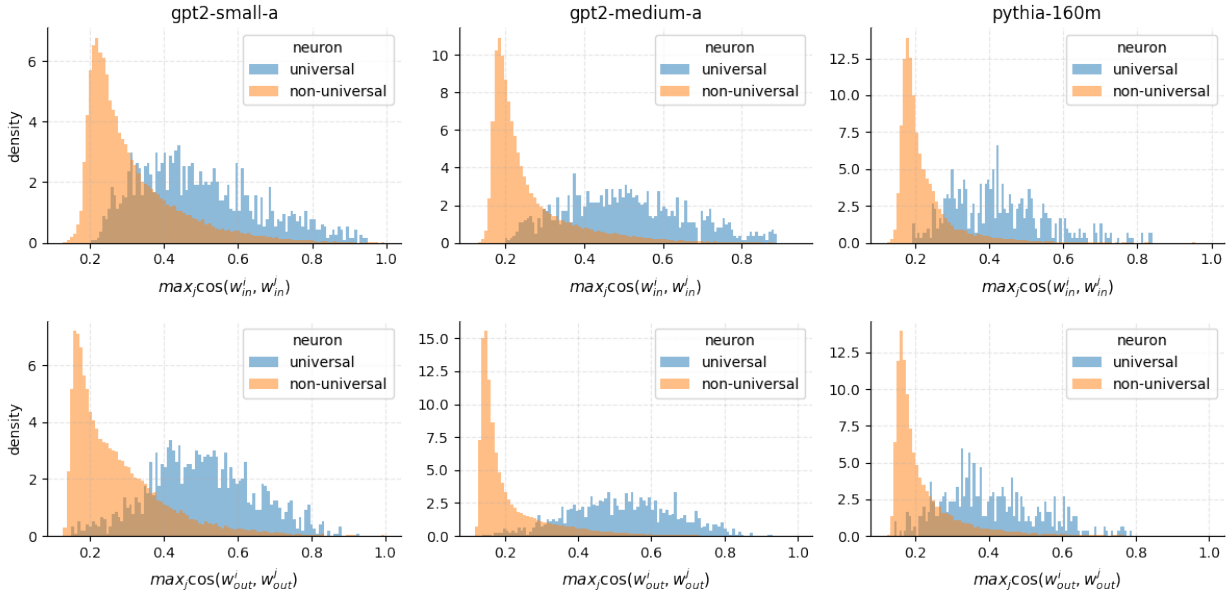


Figure B.2: Distribution of cosine similarities of most similar neurons measured by input weights (top) and output weights (bottom) for GPT2-small-a (left), GPT2-medium-a (middle), and Pythia-160M (right) colored by universality ($\varrho > 0.5$).

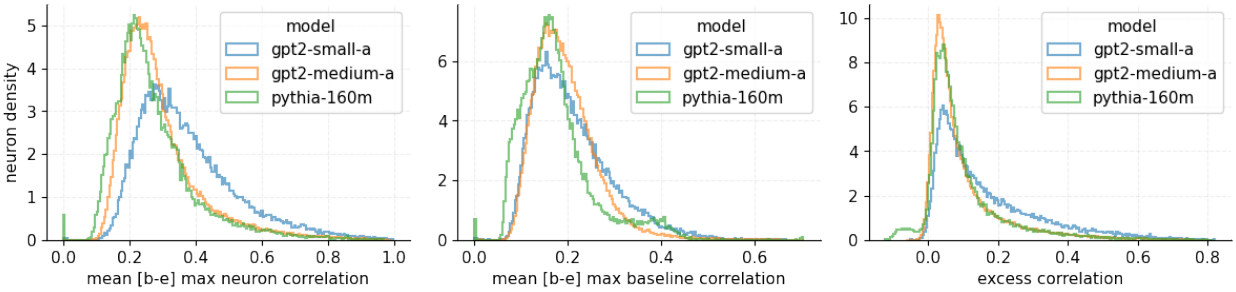


Figure B.3: Empirical distribution of max neuron correlation averaged across models (left), max baseline correlation averaged across models (middle), and the difference denoted as the excess correlation (right).

B.3.3 Scale and Universality

As mentioned in § 3.4, GPT2-medium and Pythia-160M have a consistent number of universal neurons (1.23% and 1.26% respectively), while GPT2-small-a has many more 4.16%. In Figure B.3 we show the distribution of max, baseline, and excess correlations for all models, where we see that GPT2-medium and Pythia-160M have almost identical distributions while GPT2-small is an outlier. GPT2-small also has correspondingly greater weight redundancy as shown in Figure B.2. One explanation for this is the number of universal neurons decreases in larger models. This is potentially implied by results from [38] who observe larger models have fewer neurons which admit high quality natural language interpretations. However, without additional experiments on larger models trained from random seeds, this remains an open question.

B.4 Additional Results

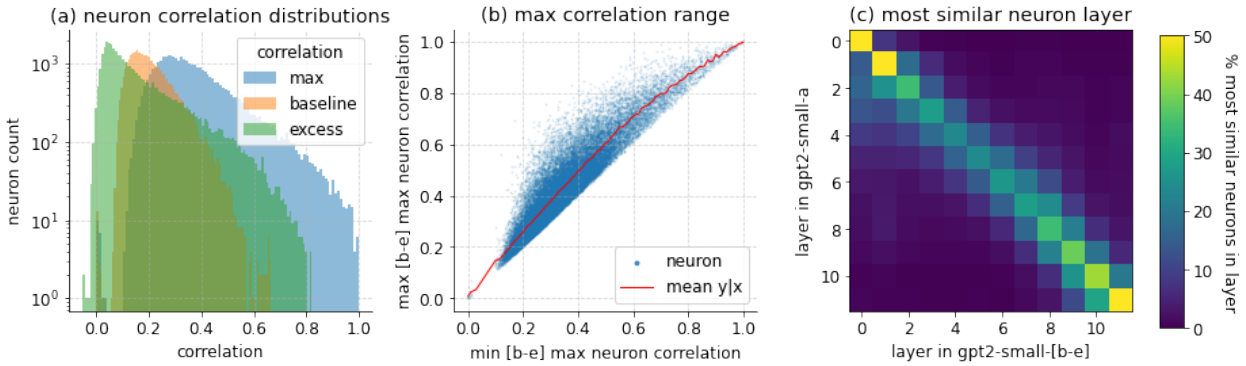


Figure B.4: Summary of neuron correlation experiments in GPT2-small-a. (a) Distribution of the mean (over models b-e) max (over neurons) correlation, the mean baseline correlation, and the difference (excess). (b) The max (over models) max (over neurons) correlation compared to the min (over models) max (over neuron) correlation for each neuron. (c) Percentage of layer pairs with most similar neuron pairs.

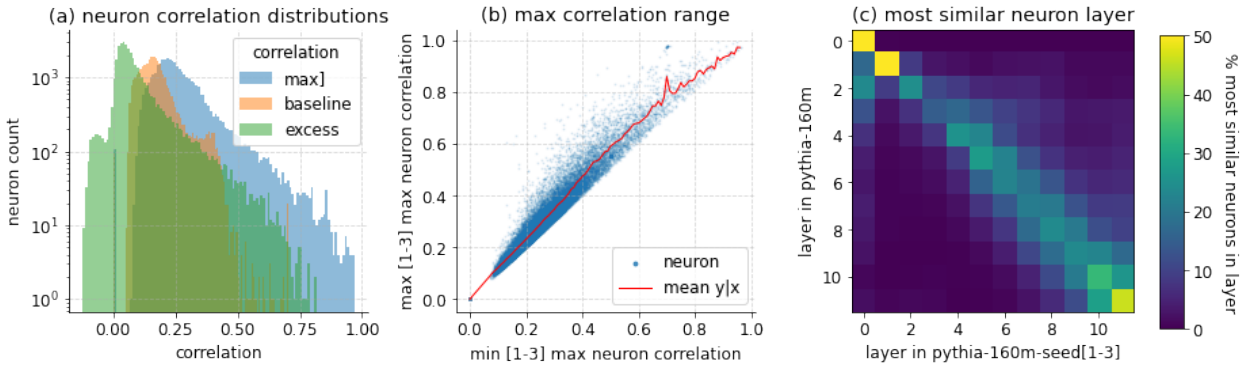


Figure B.5: Summary of neuron correlation experiments in Pythia-160m. (a) Distribution of the mean (over models b-e) max (over neurons) correlation, the mean baseline correlation, and the difference (excess). (b) The max (over models) max (over neurons) correlation compared to the min (over models) max (over neuron) correlation for each neuron. (c) Percentage of layer pairs with most similar neuron pairs.

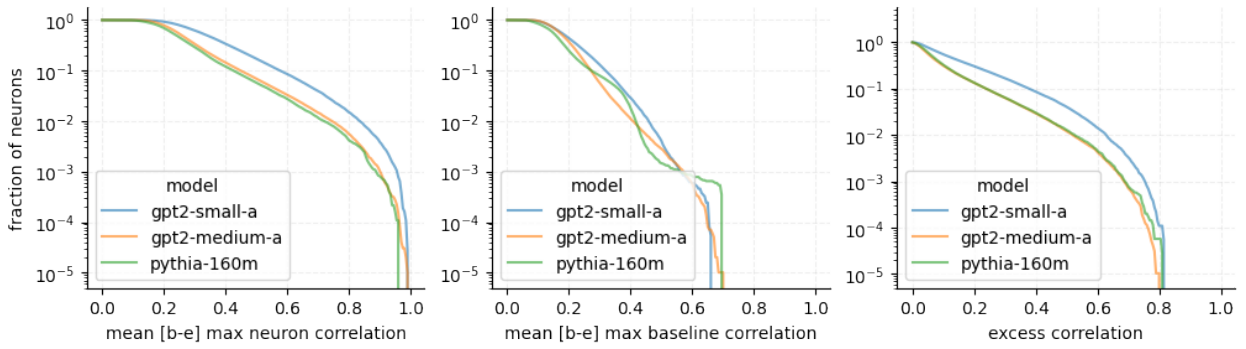


Figure B.6: Complement cumulative distribution function for correlation metrics across all model families.

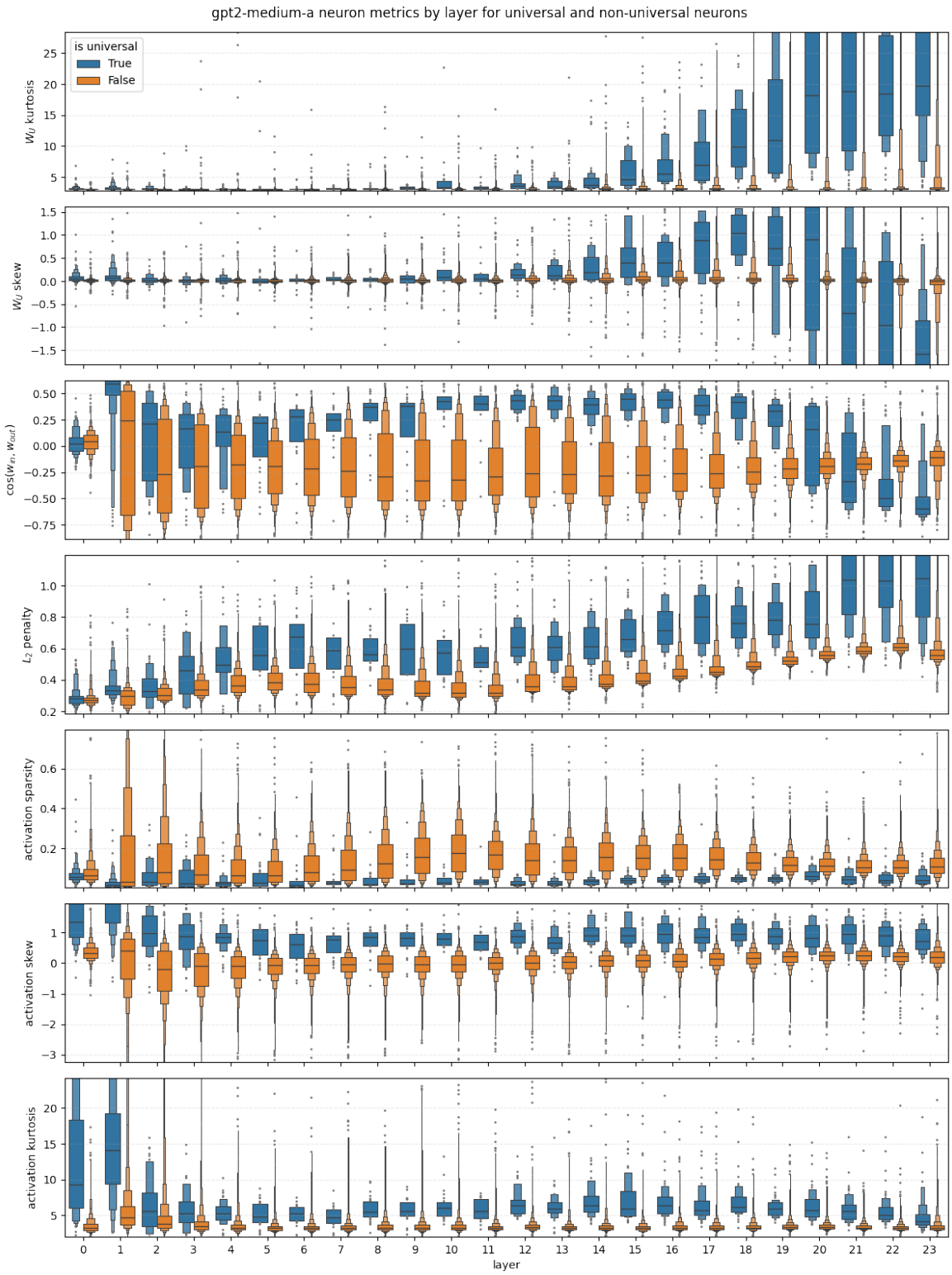


Figure B.7: Distribution of neuron metrics for universal and non-universal neurons in GPT2-medium-a by layer. From top to bottom: the kurtosis of $\cos(\mathbf{W}_U, \mathbf{w}_{out})$, the skew of $\cos(\mathbf{W}_U, \mathbf{w}_{out})$, cosine similarity between input and output weight $\cos(\mathbf{w}_{in}, \mathbf{w}_{out})$, weight decay penalty $\|\mathbf{w}_{in}\|_2^2 + \|\mathbf{w}_{out}\|_2^2$, activation frequency (percentage of activations greater than 0), the pre-activation skew, and the pre-activation kurtosis.

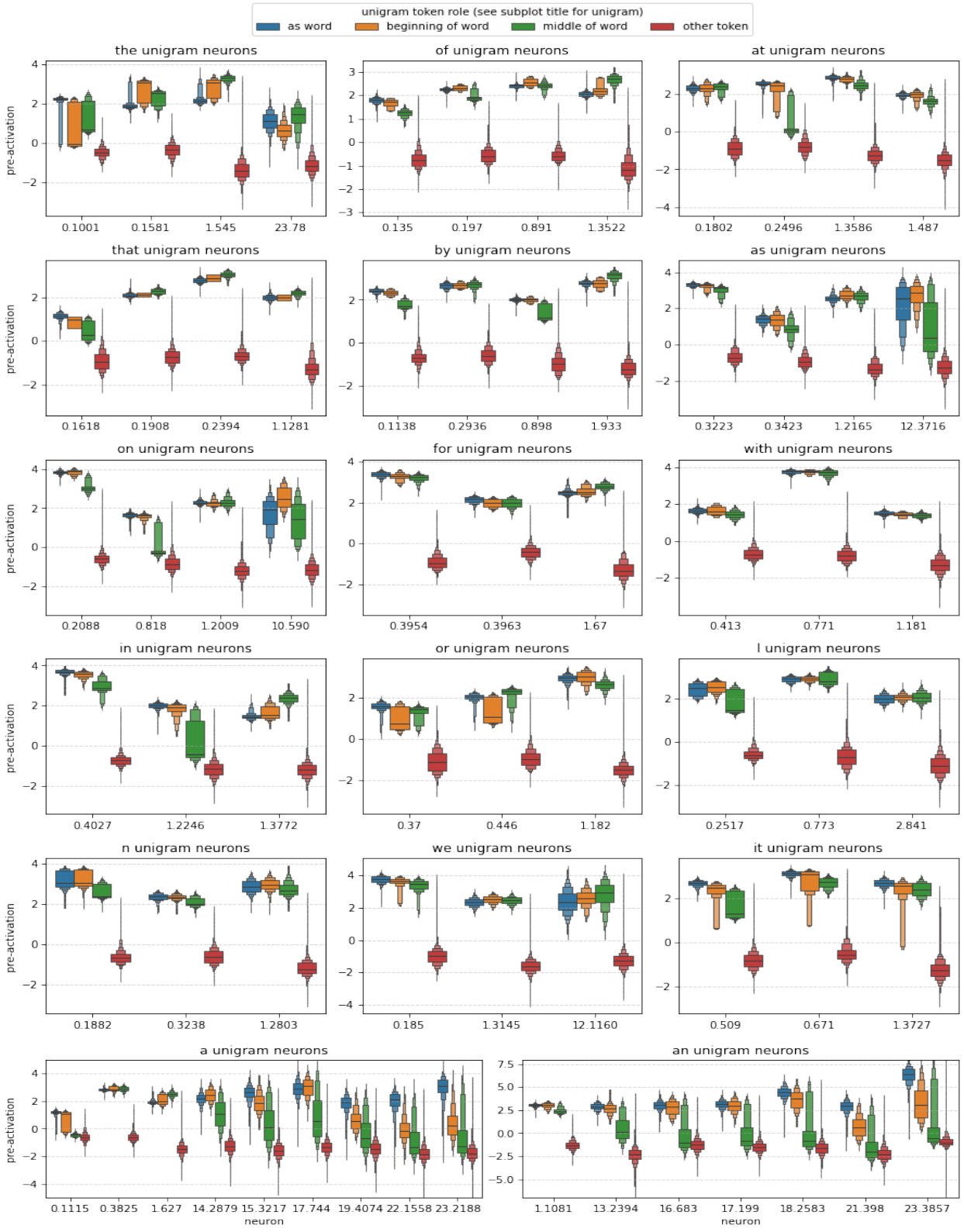


Figure B.8: Duplicate unigram neurons in GPT2-medium-a. Each subplot depicts several neurons which activate on a particular token, broken down by whether this token exists as a standalone word, is the first token in a multi-token word, or is a non-first token in a multi-token word, versus all other tokens (e.g., "an," "an|agram," "Gig|an|tism").

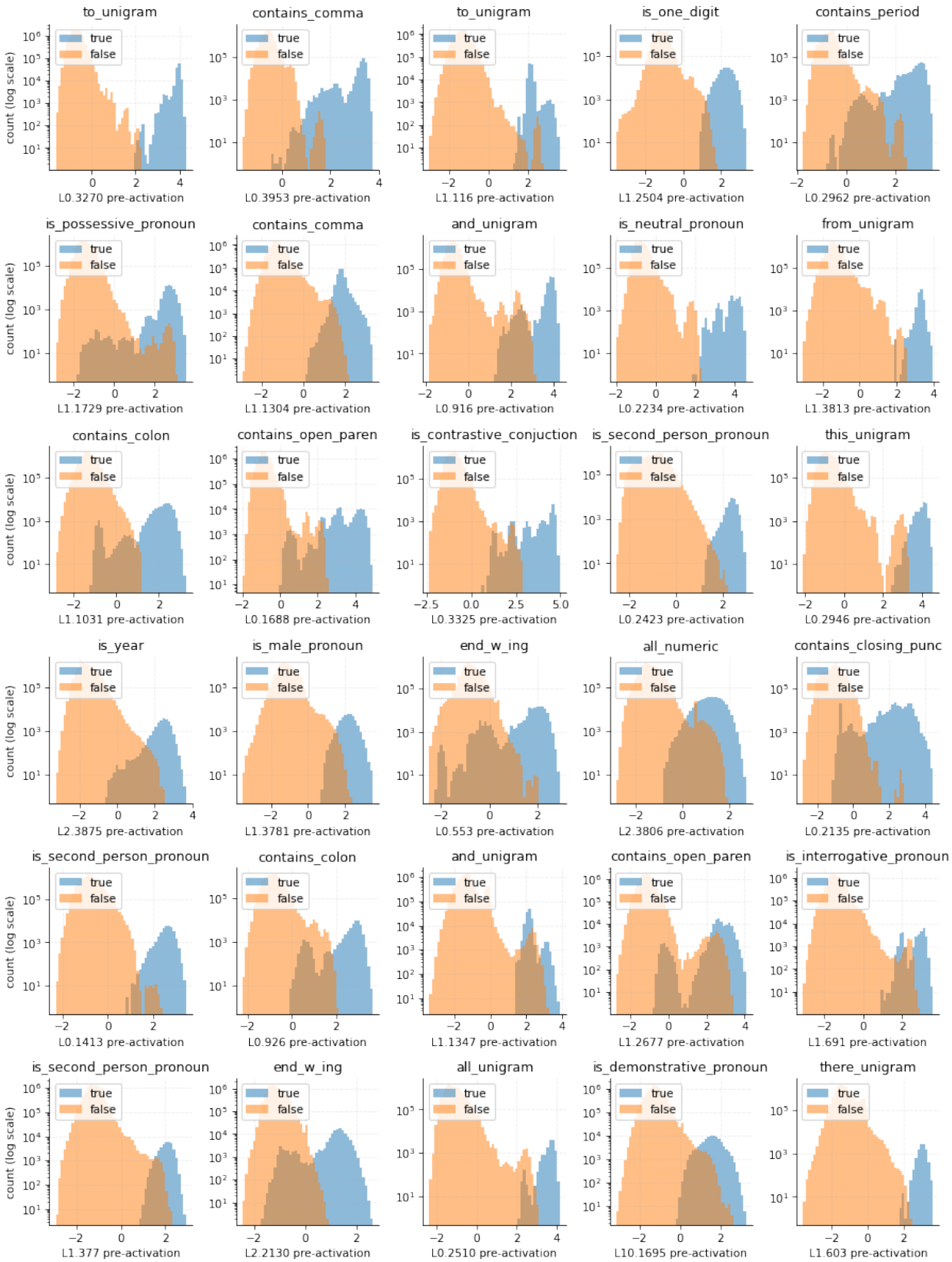


Figure B.9: Universal unigram neurons in GPT2-medium-a.

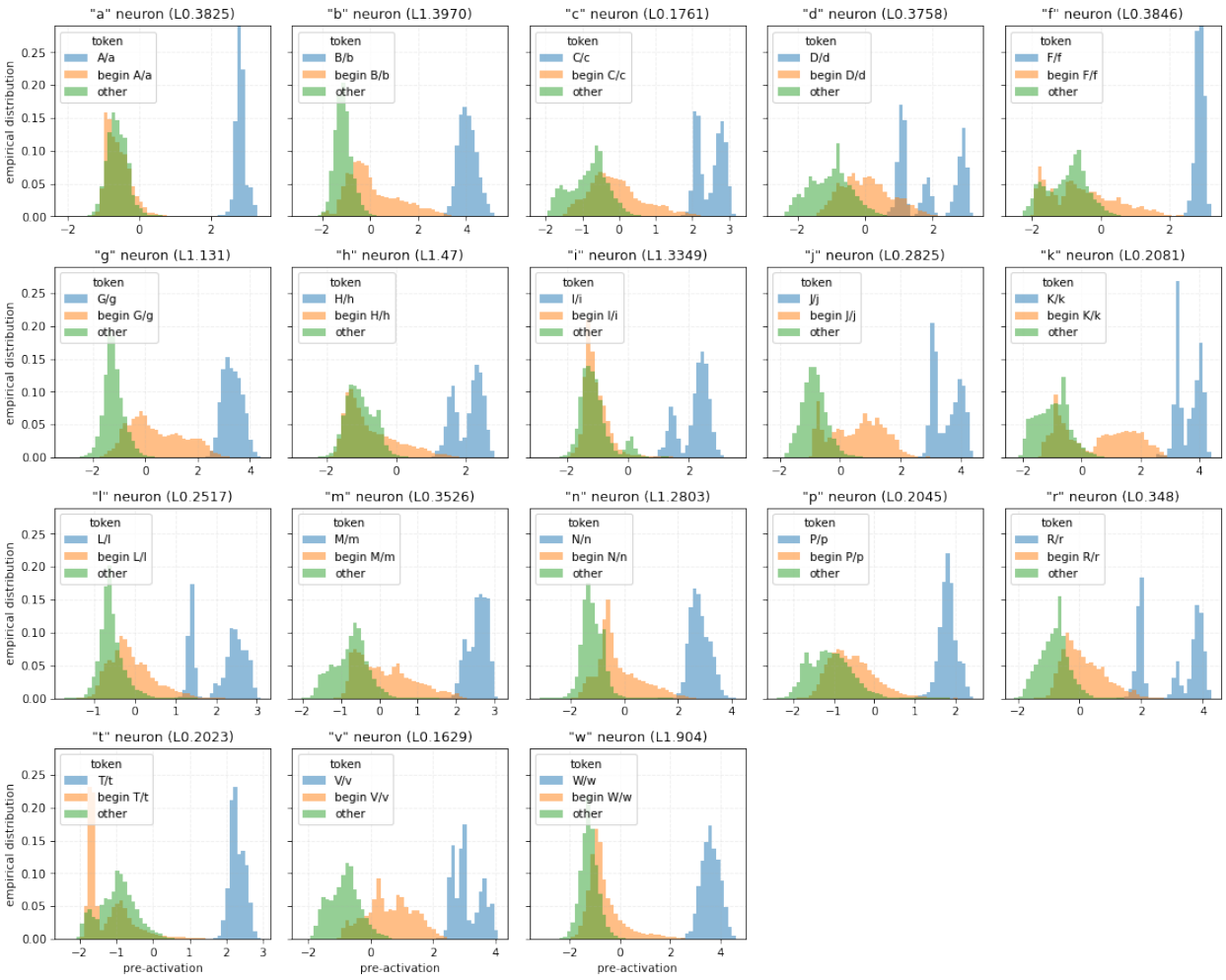


Figure B.10: Universal alphabet neurons in GPT2-medium-a.

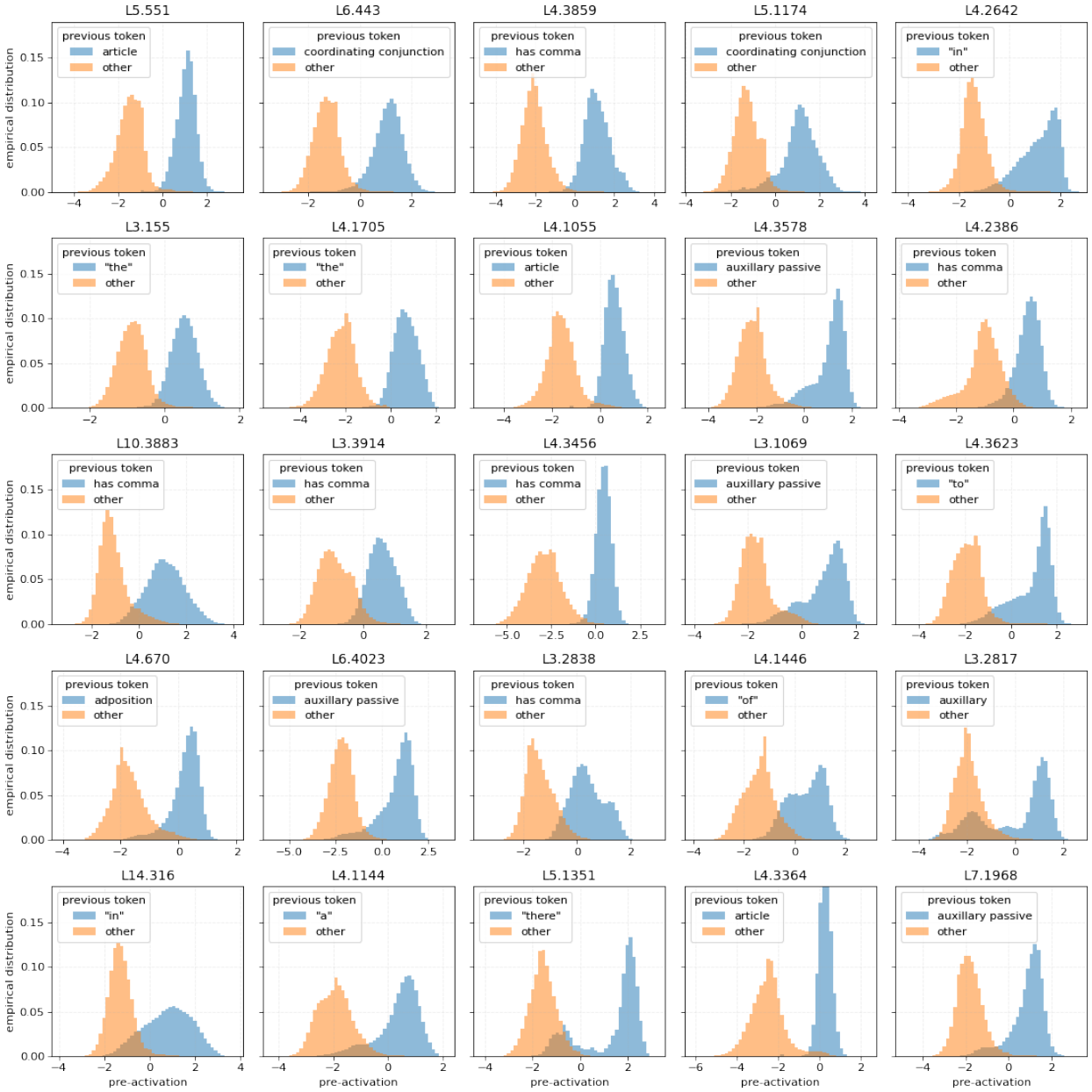


Figure B.11: Universal previous token neurons in GPT2-medium-a.

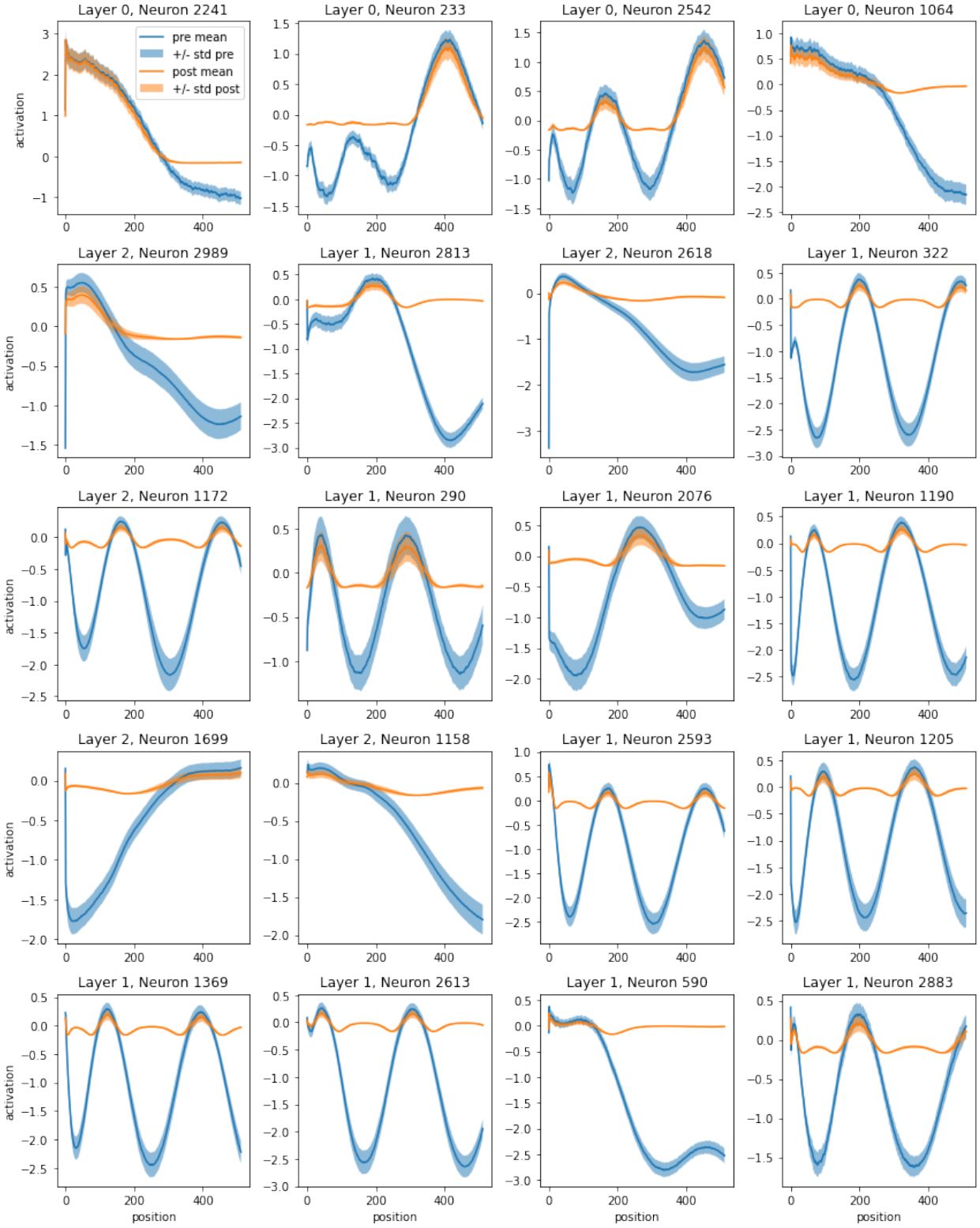


Figure B.12: Universal position neurons in GPT2-small-a.

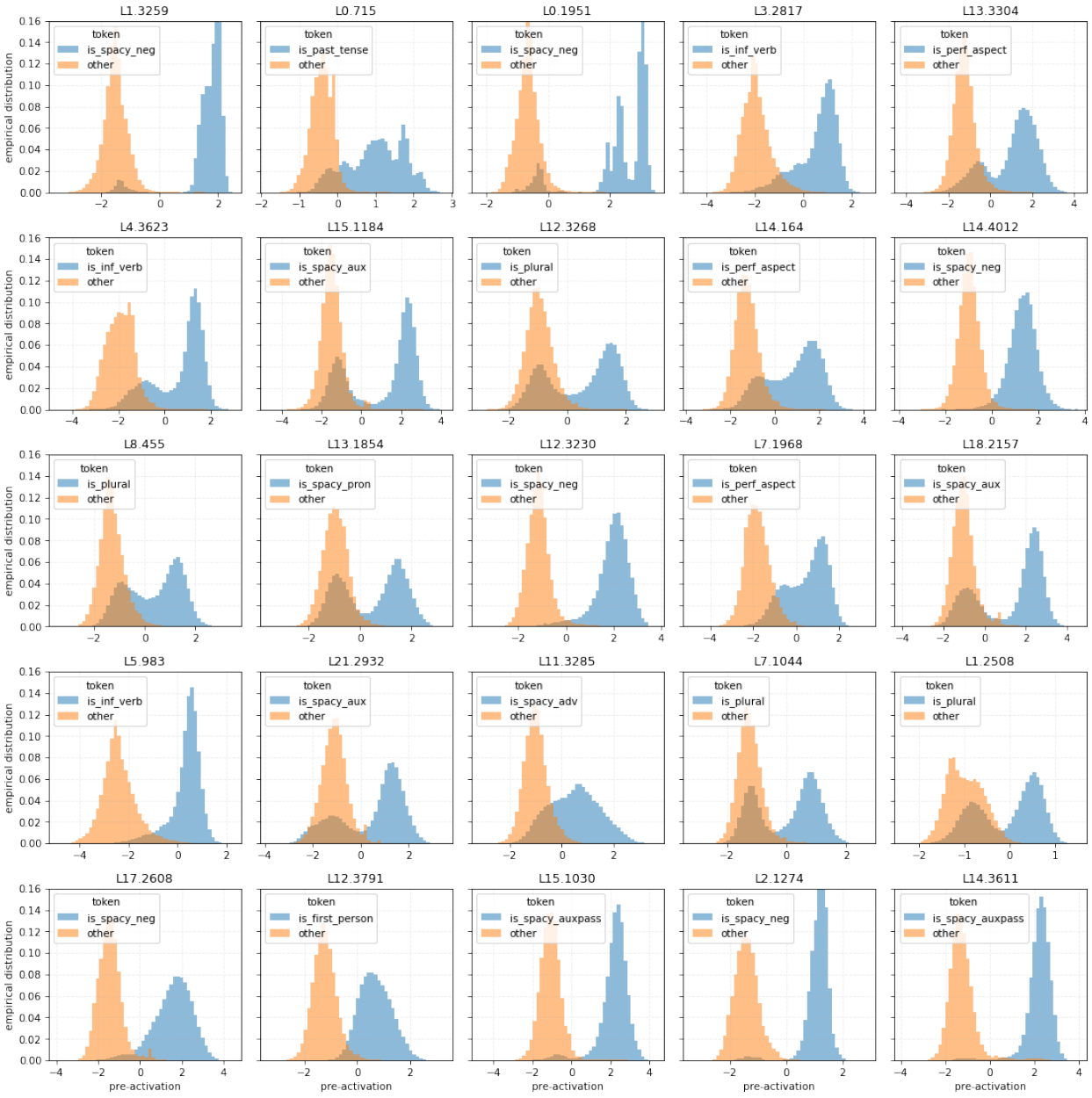


Figure B.13: Universal syntax neurons in GPT2-medium-a.

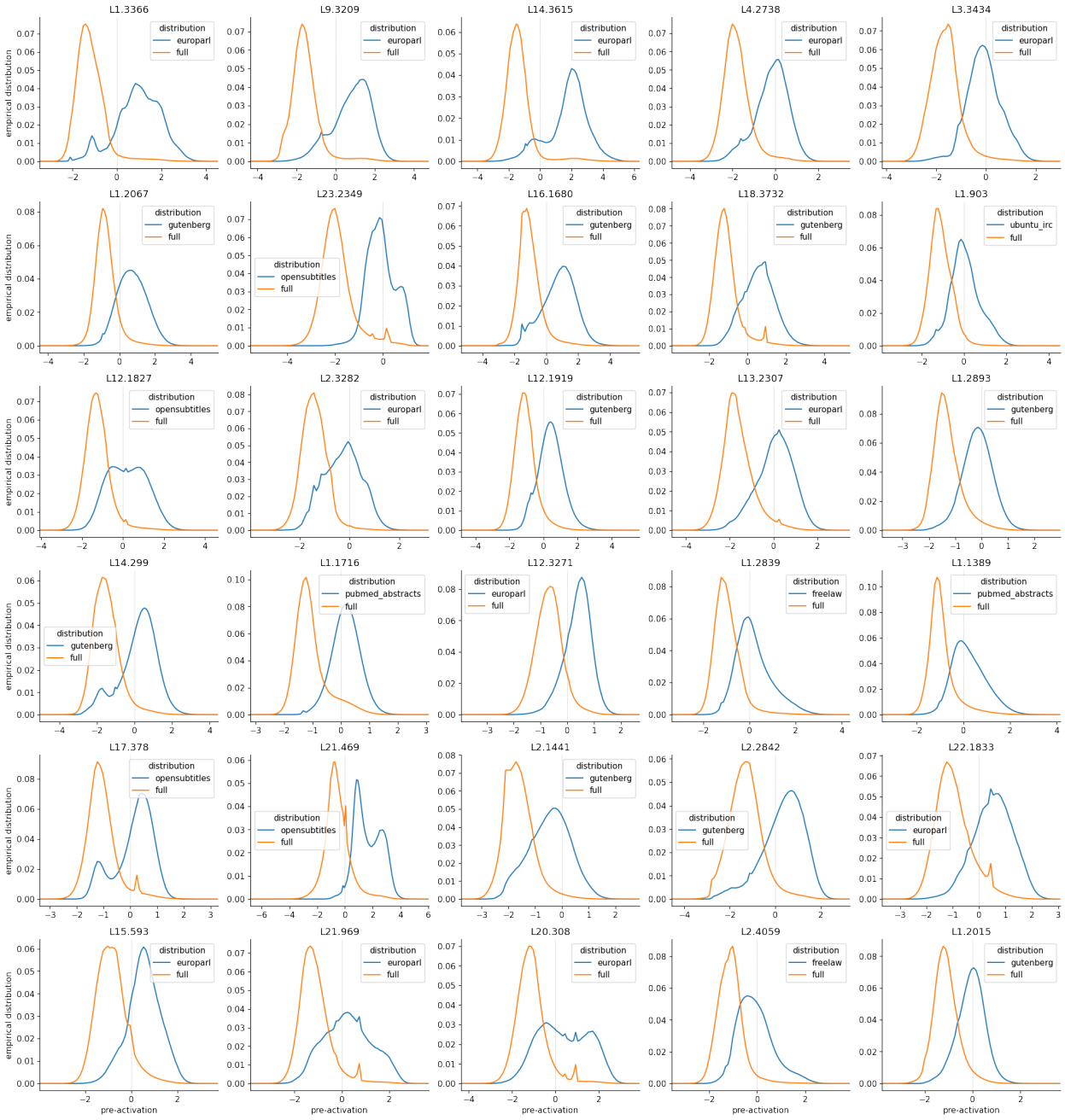


Figure B.14: Universal context neurons in GPT2-medium-a.

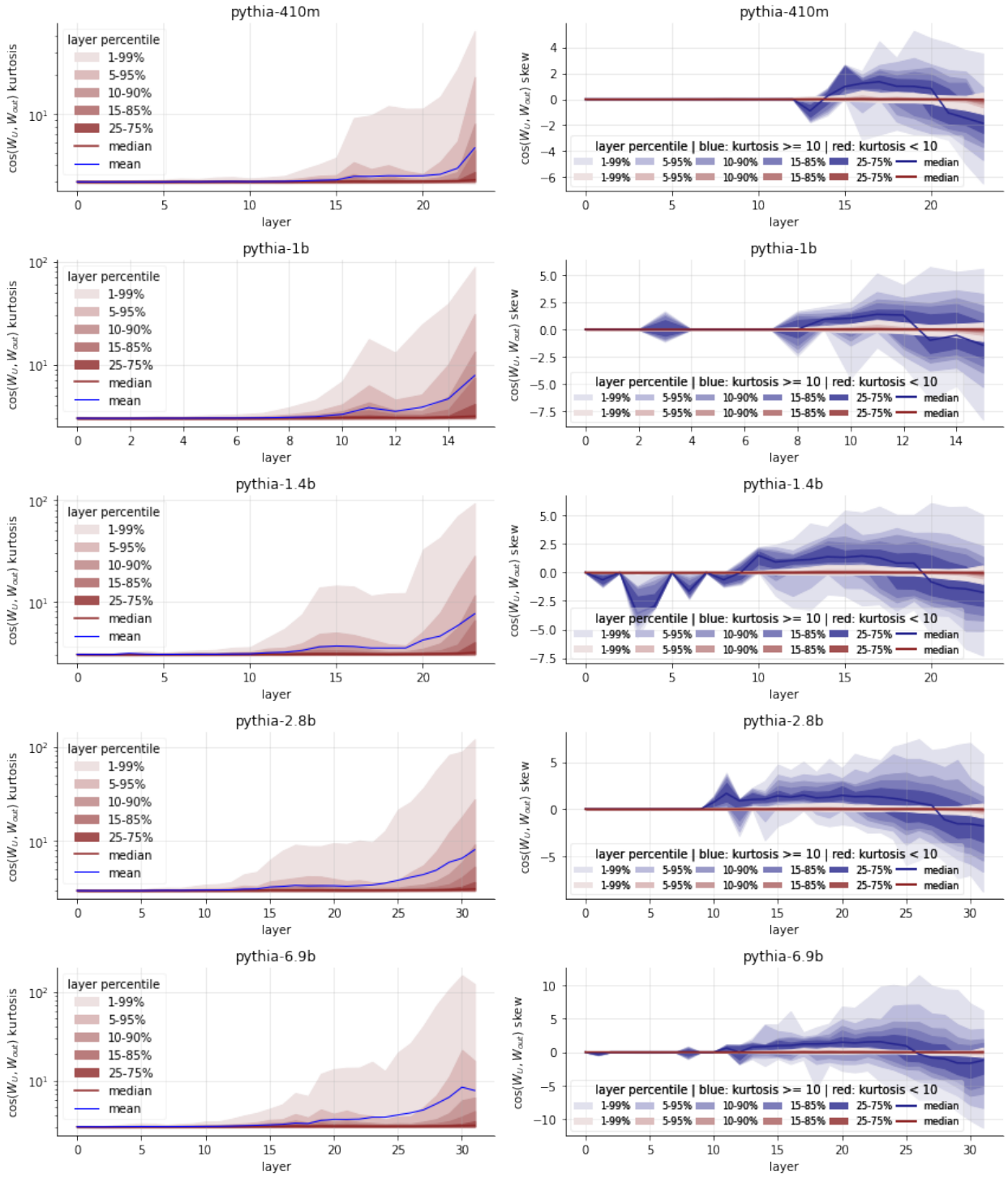


Figure B.15: Distribution of vocabulary composition statistics for five different Pythia models measured over layers. Left shows percentiles of $\cos(\mathbf{W}_U, \mathbf{W}_{out})$ kurtosis. Right shows percentiles of $\cos(\mathbf{W}_U, \mathbf{W}_{out})$ skew broken down by whether neuron has $\cos(\mathbf{W}_U, \mathbf{W}_{out})$ kurtosis greater than or less than 10.

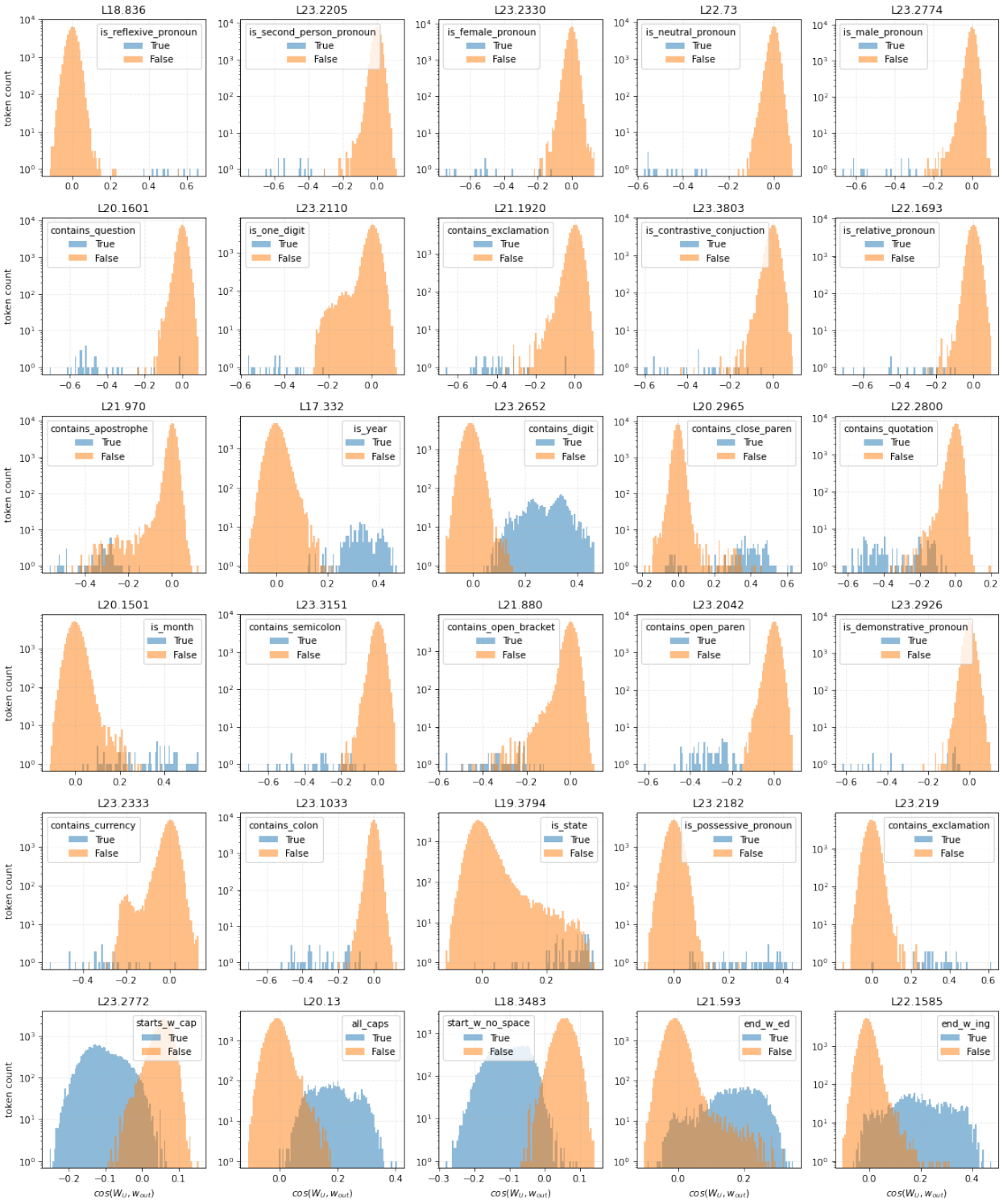


Figure B.16: Universal prediction neurons in GPT2-medium-a.

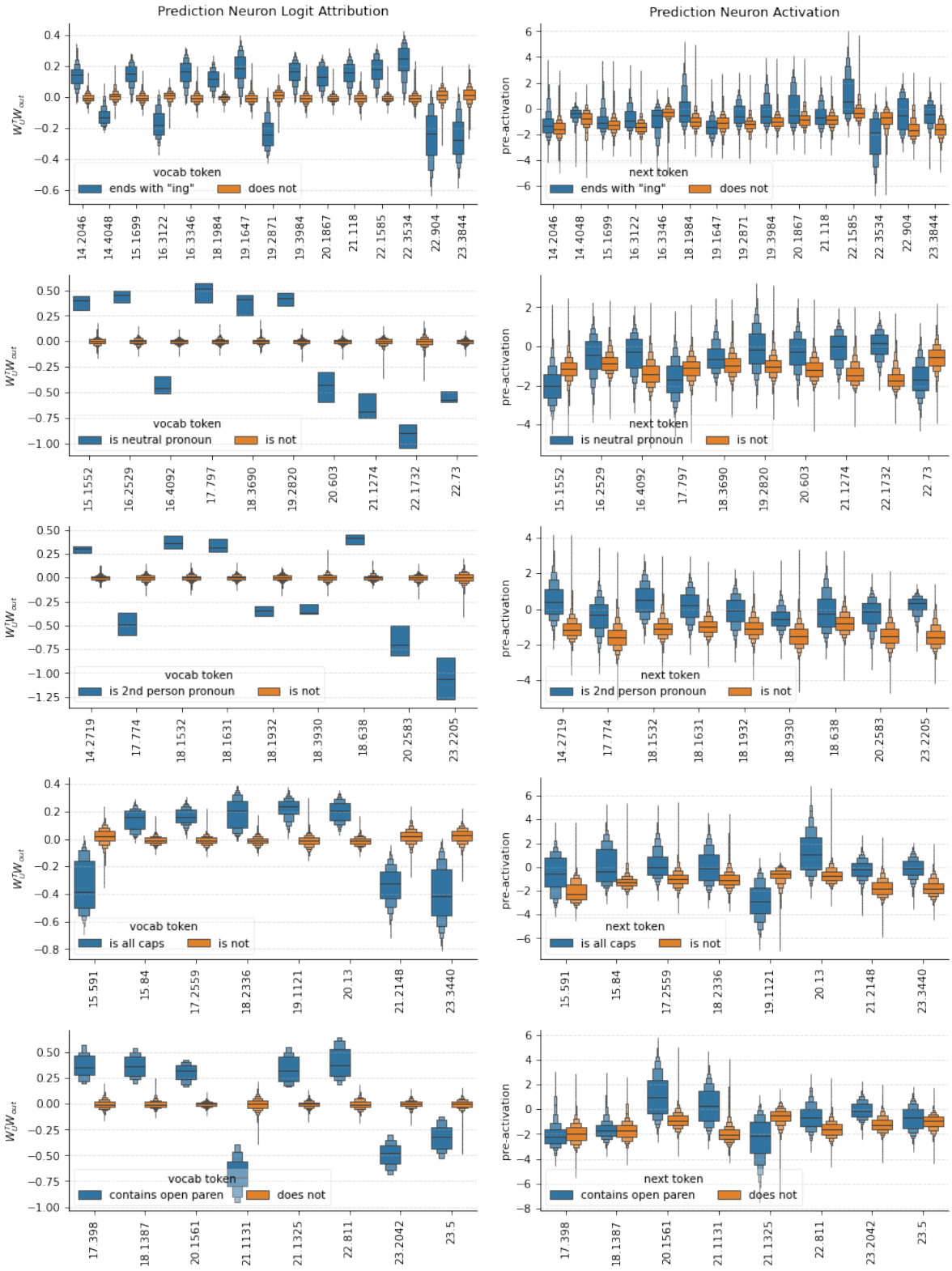


Figure B.17: Prediction neurons for the same feature in GPT2-medium-a. Left column depicts logit effect broken down by vocabulary item per neuron and right column shows activation value broken down by true next token per neuron.

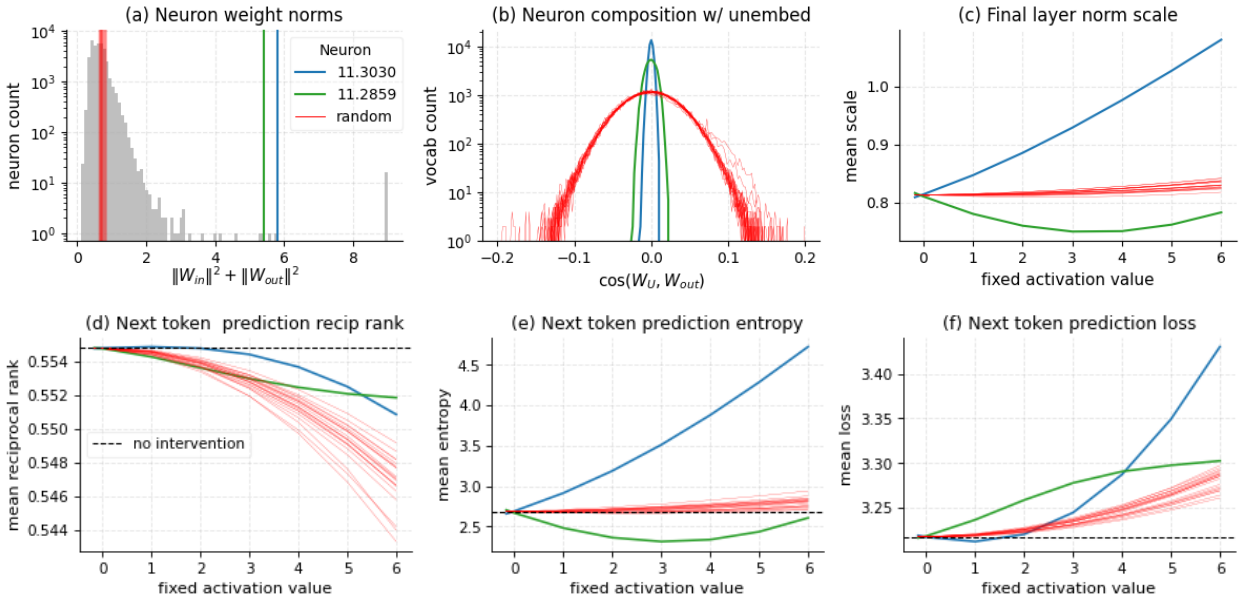


Figure B.18: Summary of (anti-)entropy neurons in GPT2-small-a compared to 20 random neurons from final two layers. Entropy neurons have high weight norm (a) with output weights mostly orthogonal to the unembedding matrix (b). When activated, this causes the final layer norm scale to increase dramatically (c) while leaving the relative ordering over the next token prediction mostly unchanged (d). Increased layer norm scale squeezes the logit distribution, causing a large increase in the prediction entropy (e; or decrease for anti-entropy neuron) and an increase or decrease in the loss depending on the model's baseline level of under- or over-confidence (f). Legend applies to all subplots.

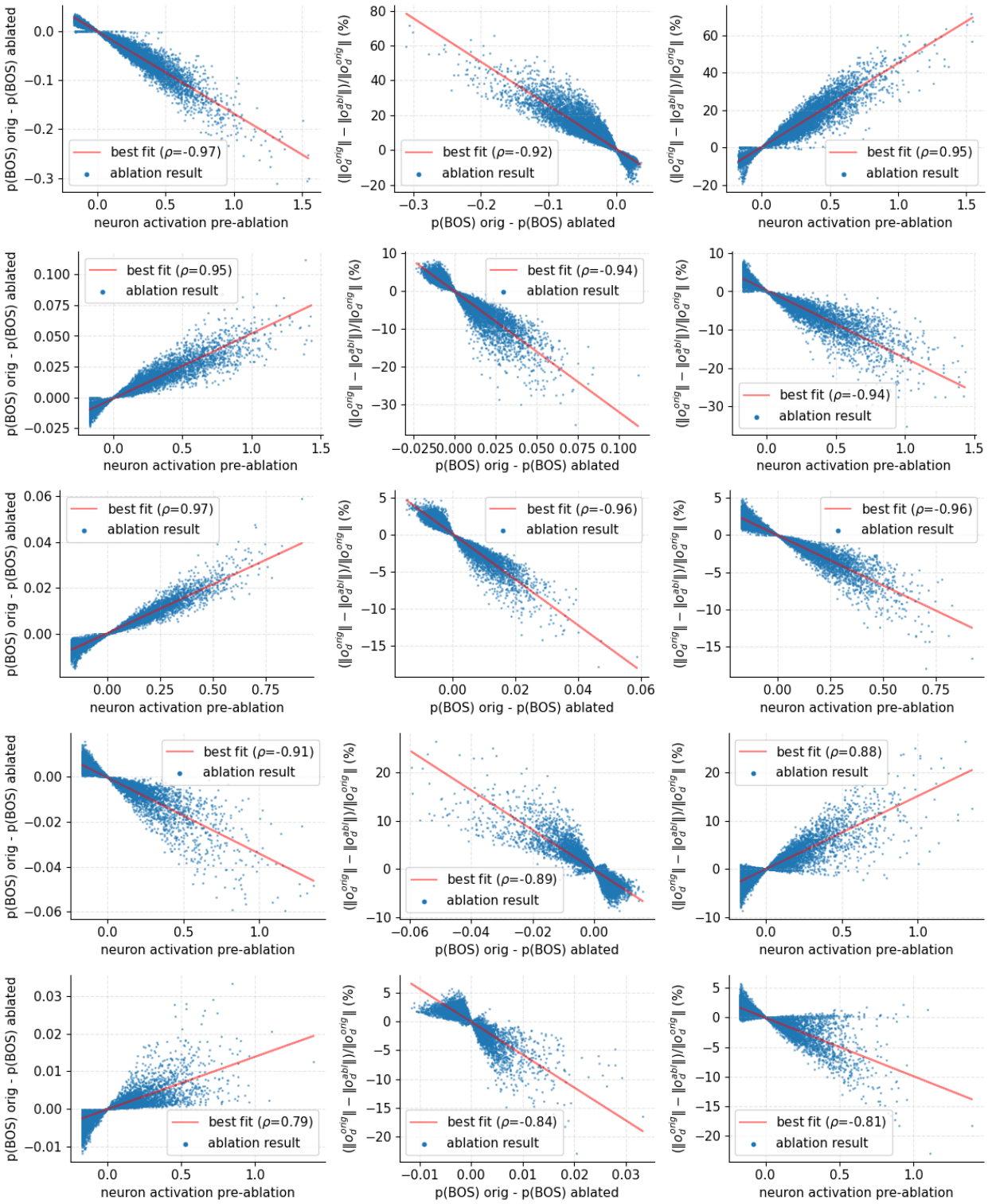


Figure B.19: Further examples of attention activation and deactivation neurons. Row 1: A15H8 with L14N411, Row 2: A15H8 with L14N2335, Row 3: A15H8 with L14N1625, Row 4: A20H4 with L19N2509, Row 5: A22H7 with L20N2114

Appendix C

Space and Time Representations Appendix

C.1 Datasets

We describe the construction and post-processing of our data in more detail in addition to known limitations. All datasets and code are available at https://github.com/wesg52/world-models.

World Places We ran three separate queries to obtain the names, location, country, and associated Wikipedia article of all physical places, natural places, and structures within the DBPedia database [161]. Using the Wikipedia article link, we joined this information with data from the Wikipedia pageview statistics database [1] to query how many times this page was accessed over 2018-2020. We use this as a proxy for whether we should expect an LLM to know of this place or not, and filter those with less than 5000 views over this time period.

Several limitation are worth highlighting. First, our data only comes from English Wikipedia, and hence is skewed towards the Anglosphere. Additionally, the distribution of entity types is not uniform, e.g. we noticed the United Kingdom has many more railway stations than any other country, which could introduce unwanted correlations in the data that may affect the probes. Finally, about 25% of the samples had some sort of state or province modifier at the end like"Dallas County, Iowa". Because many of these locations were more obscure or would be ambiguous without the modifier, so we chose to rearrange the string to be of the from "Iowa's Dallas County" such the entity is disambiguated but that we are not probing on a token that is a common country or state name.

[1] https://en.wikipedia.org/wiki/Wikipedia:Pageview_statistics

USA Places The United States places dataset uses structures and natural places within the US from the world places dataset as a starting point, in addition to another DBPedia for US colleges. We then collect the name, population total, and state for every county[2], zipcode [3], and city [4] from a census data aggregator. We then remove all duplicate county or city names (there are 31 Washington counties in the US!), though we keep any duplicates that have 2x the population has the next largest place of the same name. We also filter out cities with fewer than 500 people, zipcodes with fewer than 10000 (or with population density greater than 50 and population greater than 2000), and any place not in the lower 48 contiguous states (or Washington D.C.).

NYC Places Our New York City dataset is adapted from the NYC Open Date points of interest dataset [207] containing the names of locations tracked by the city government. This includes the names of schools, places of worship, transit locations, important roads or bridges, government buildings, public housing, and more. Each of these places comes with a complex ID for locations comprised of multiple such buildings (e.g. New York University or LaGuardia airport). We construct our test train splits to make sure that all locations within the same complex are put in the same split to avoid test-train leakage. We filtered out a large number of locations describing the position of bouys in the multiple waterways surrounding NYC.

Historical Figures Our historical figures dataset contains the names and occupation of historical figures who died between 1000BC-2000AD adapted from [8]. We filtered the dataset to only contain the 350 most famous people who died from each decade, imperfectly measured by the index of their Wikidata entity identifier.

Art and Entertainment Our art and entertainment dataset consists of the names of songs, movies, and books with their corresponding artist, director, and author release date. We constructed this dataset from DBpedia and similarly filtered out entities which had received less than 5000 page views over 2018-2020. Because many songs or books have fairly generic titles, we include the creator's name in the prompt to disambiguate (e.g. "Stephen Kings' It" for the `empty` prompt). However, because some artists or authors release many songs or books, we sample our test-train split by creator to avoid leakage.

[2] https://simplemaps.com/data/us-counties
[3] https://simplemaps.com/data/us-zips
[4] https://simplemaps.com/data/us-cities

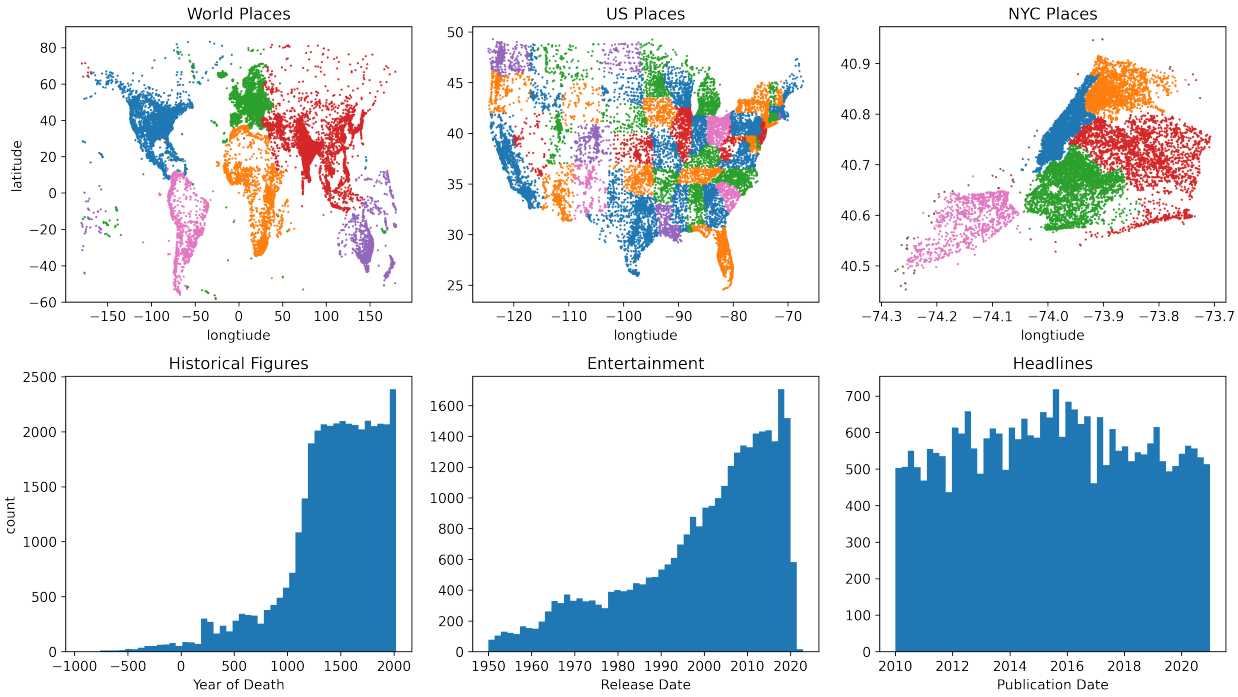


Figure C.1: Distribution of samples in space or time for all datasets.

Headlines Our headlines dataset is adapted from a scrape of all New York Times headlines of the past 30 years [13]. In an attempt to filter out headlines which do not describe an event that could be localized in time, we employ a number of strategies. First we filter anything which is not within the first 10 pages of the print edition. Second we filter out articles that don't come from the Foreign, National, Politics, Washington, or Obits news desks. Third we removed any titles that contained a question mark.

C.2 Neuron Ablations and Interventions

To better understand the role of space and time neurons in LLMs, we conduct several neuron ablation and intervention experiments.

Time Intervention We study the effect of intervening on a single time neuron (L19.3610; correlation with art and entertainment release date of 0.77) within Llama-2-7b. Given a prompt of the form `<media> by <creator> was written in 19`, we pin the activation of the time neuron on all tokens and sweep over a range of pinned values, and track the predicted probability of the top five tokens. Results are depicted in Figure C.2 and show that just adjusting the time neuron activation can change the next token prediction in all cases.

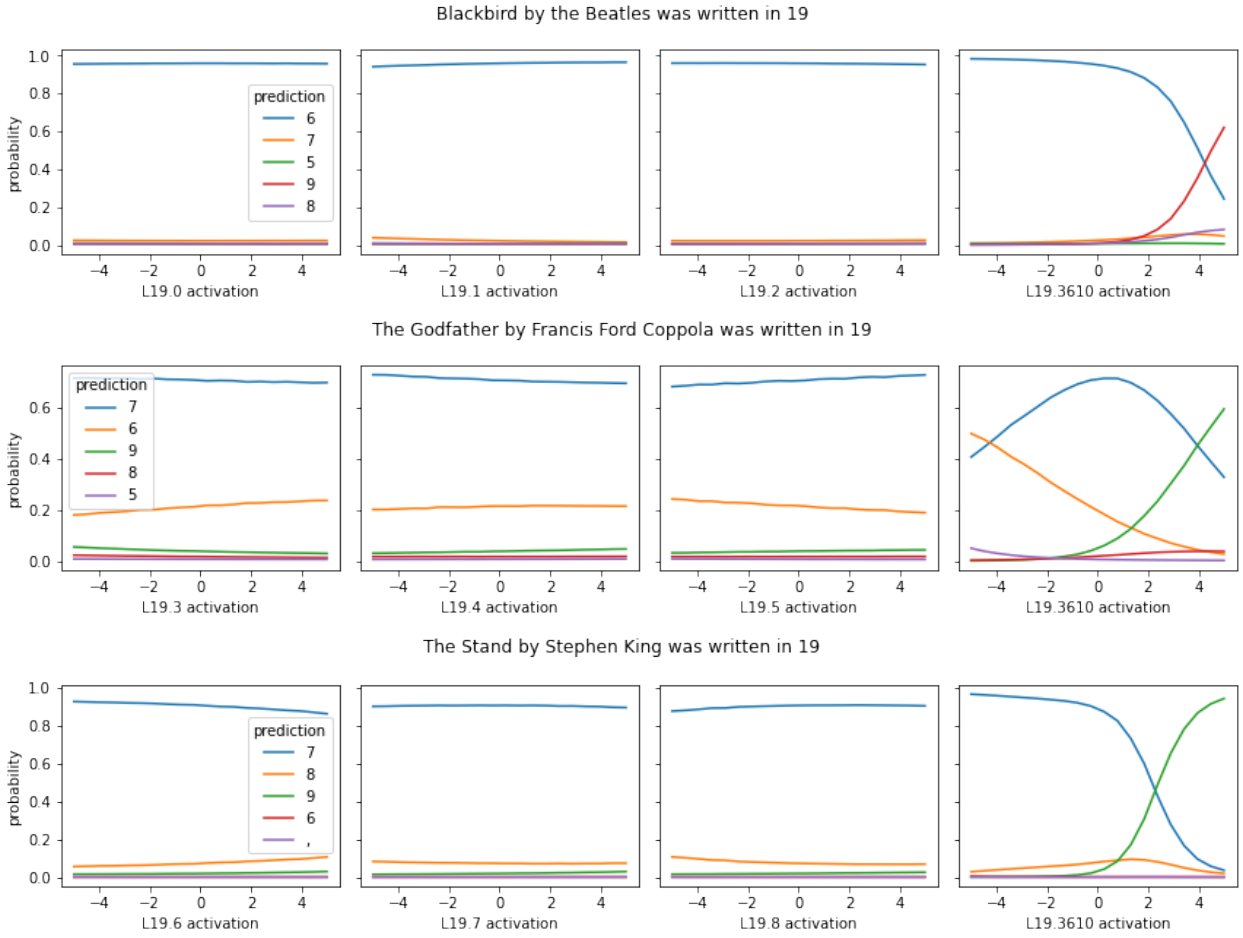


Figure C.2: Prediction of decade of publication for a famous song, movie, and book when a time neuron (L19.3610) is pinned to a particular value, compared to 9 random neurons within the same layer (L19.[0-8]) of Llama-2-7b.

context	true token	loss increase
Bom Jesus has a rather dry tropical savanna climate (Köppen	Aw	2.107
line tropical monsoon/humid subtropical climate (Köppen	Am	2.035
of . The highest temperatures are reached at the end of the dry season in	March	1.960
8.9 °C. In January, the average temperature is 1	8	1.930
a Tropical wet-and-dry climate (Köppen climate classification	Aw	1.876
Goroka has a relatively cool tropical monsoon climate (Köppen	Am	1.854
ie range from 26.4 °F in January to 7	0	1.835
wet summers and warm, very wet winters (Köppen climate classification	Am	1.807
tropical wet and dry climate/semi-arid climate (Köppen	Aw	1.783
ably mild, tropical-maritime climate, (Köppen climate classification	Aw	1.762

Table C.1: Contexts with the highest loss when ablating space neuron L20.7573 from Llama-2-7b.

context	true token	loss increase
mark is Mount Etna, one of the tallest active volcanoes in	Europe	1.971
2,800 years, making it one of the oldest cities in	Europe	1.676
keeper with 63 caps for Portugal including participation in the 198	4	1.631
15. Tenerife also has the largest number of endemic species in	Europe	1.512
Georgia to the south-west. Mount Elbrus, the highest mountain in	Europe	1.246
At one point, the village boasted the longest aluminium rolling mill in	Western	1.219
centre and the leading economic hub of the Iberian Peninsula and of	Southern	1.181
atican City, a sovereign state—and possibly the second largest in	Europe	1.103
name. This is because the British Isles were likely repopulated from the	I	1.082
Category 4 stadium by UEFA, hosted matches at the 199	8	1.072

Table C.2: Contexts with the highest loss when ablating space neuron L20.7423 from Llama-2-7b.

Neuron Ablations We also study the effect of zero ablating neurons, and the contexts for which the loss increases the most. For a subset of Wikipedia which includes articles corresponding to world places and contemporary art and entertainment, we first run Llama-2-7b as normal and record the loss. Then, for two space neurons and two time neurons, we run the model with the neuron activation pinned to 0 (we always pin exactly one neuron to 0). For each neuron, we report the top 10 contexts in which the loss most increased for the next token prediction in Tables C.1-C.4.

context	true token	loss increase
was released in June 1993 as the fourth single from the album	He	2.254
was released in November 1992 as the second single from her album	He	1.973
93. Yearwood's version was the third single from her album	He	1.749
Hot Country Singles & Tracks chart, behind Shania Twain's ¨	Any	1.574
was released in February 1992 as the third single from the album	What	1.559
and provided additional production on her singles "Like A Prayer" and "	Express	1.481
was released in May 1992 as the fourth single from the album	What	1.367
filmmaker Ramesh Aravind in Telugu cinema. P.	L	1.328
label Columbia as the second single from their second studio album	Gold	1.272
Gessle for the duo's 1991 album,	Joy	1.253

Table C.3: Contexts with the highest loss when ablating time neuron L18.9387 from Llama-2-7b.

context	true token	loss increase
016 as the third single from Reyes debut studio album, Louder	!	1.082
as the second radio single in support of the band's third studio album,	Life	0.996
Song, but ultimately lost the award to Barbra Streisand's "	The	0.965
Released as the first single from the group's seventh studio album,	Super	0.961
original 30 Carry On films (1958–19	7	0.912
. After listening to No Doubt's 2002 single "	Under	0.867
ix9ine for his debut mixtape Day69 (201	8	0.866
BA. It was originally featured on the group's fifth studio album, The	Album	0.864
ck that appeared in the Porky Pig cartoons It's	an	0.805
was written by Andrew Lloyd Webber and Tim Rice and produced by	Fel	0.805

Table C.4: Contexts with the highest loss when ablating time neuron L19.3610 from Llama-2-7b.

C.3 Additional Results

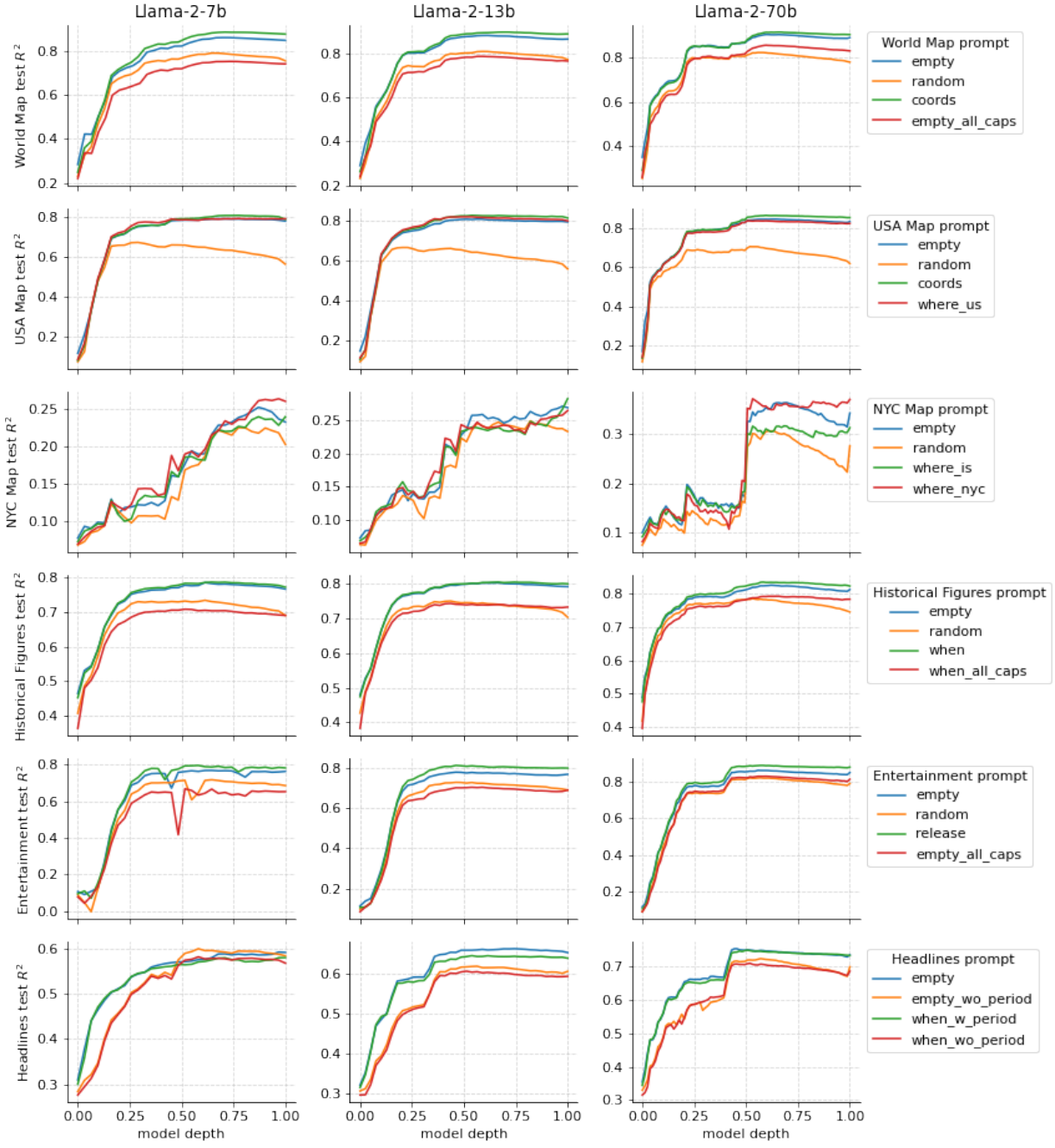


Figure C.3: Out-of-sample R^2 when entity names are included in different prompts for all models.

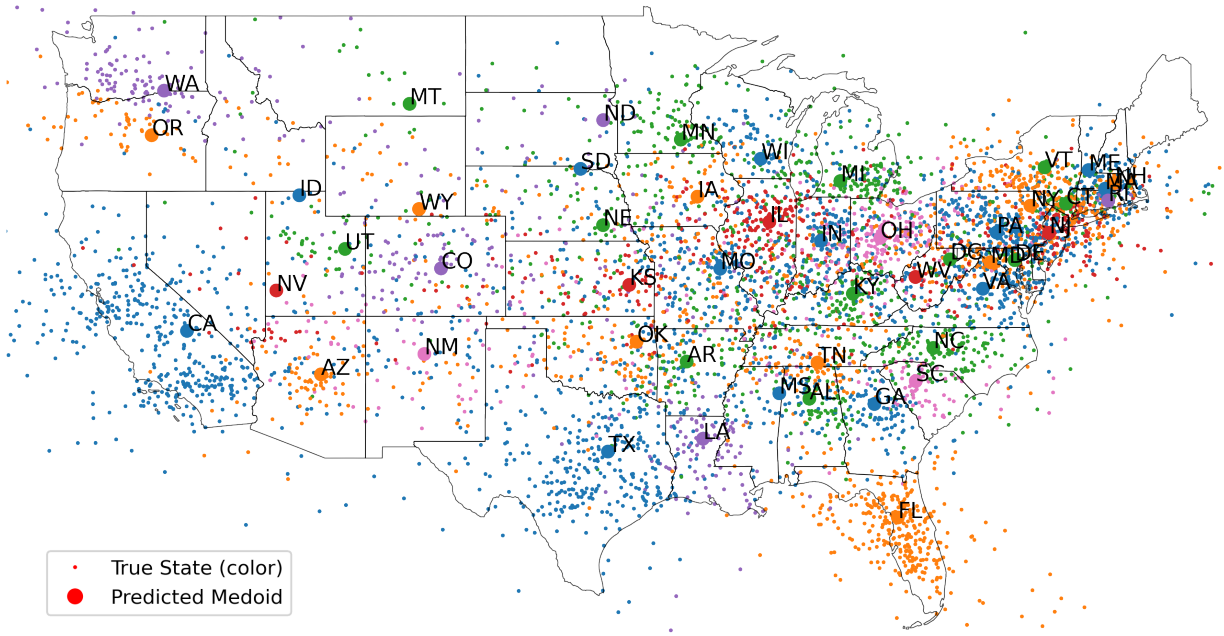


Figure C.4: Llama-2-70b layer 50 model of the United states. Points are projections of activations of heldout US places onto learned latitude and longitude directions colored by true state, with median state prediction enlarged. All points depicted are from the test set.

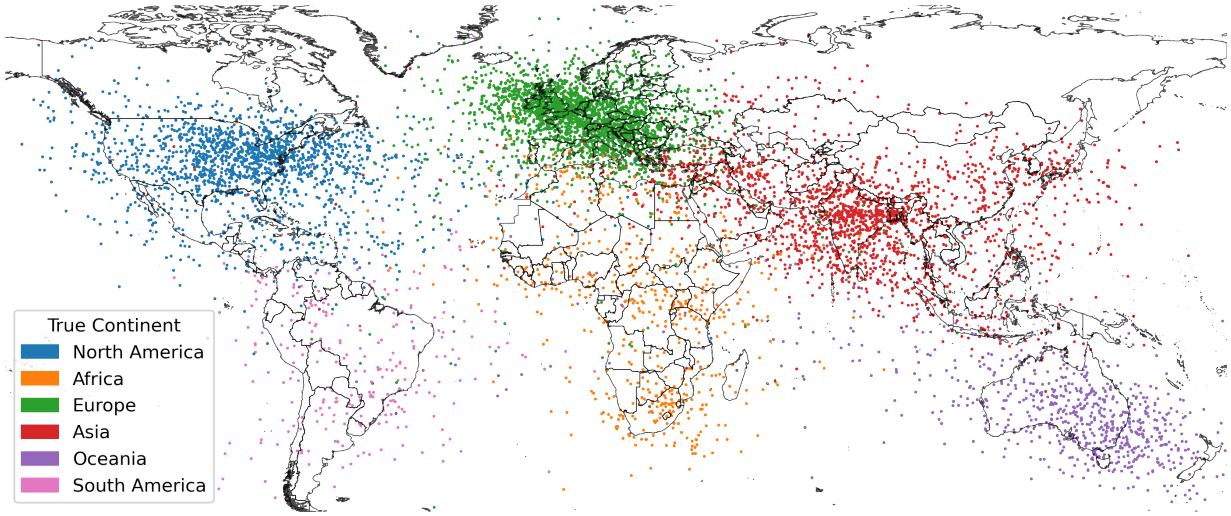


Figure C.5: Llama-2-70b layer 50 model of the world. Points are projections of activations of heldout world places onto learned latitude and longitude directions colored by true continent. All points depicted are from the test set.

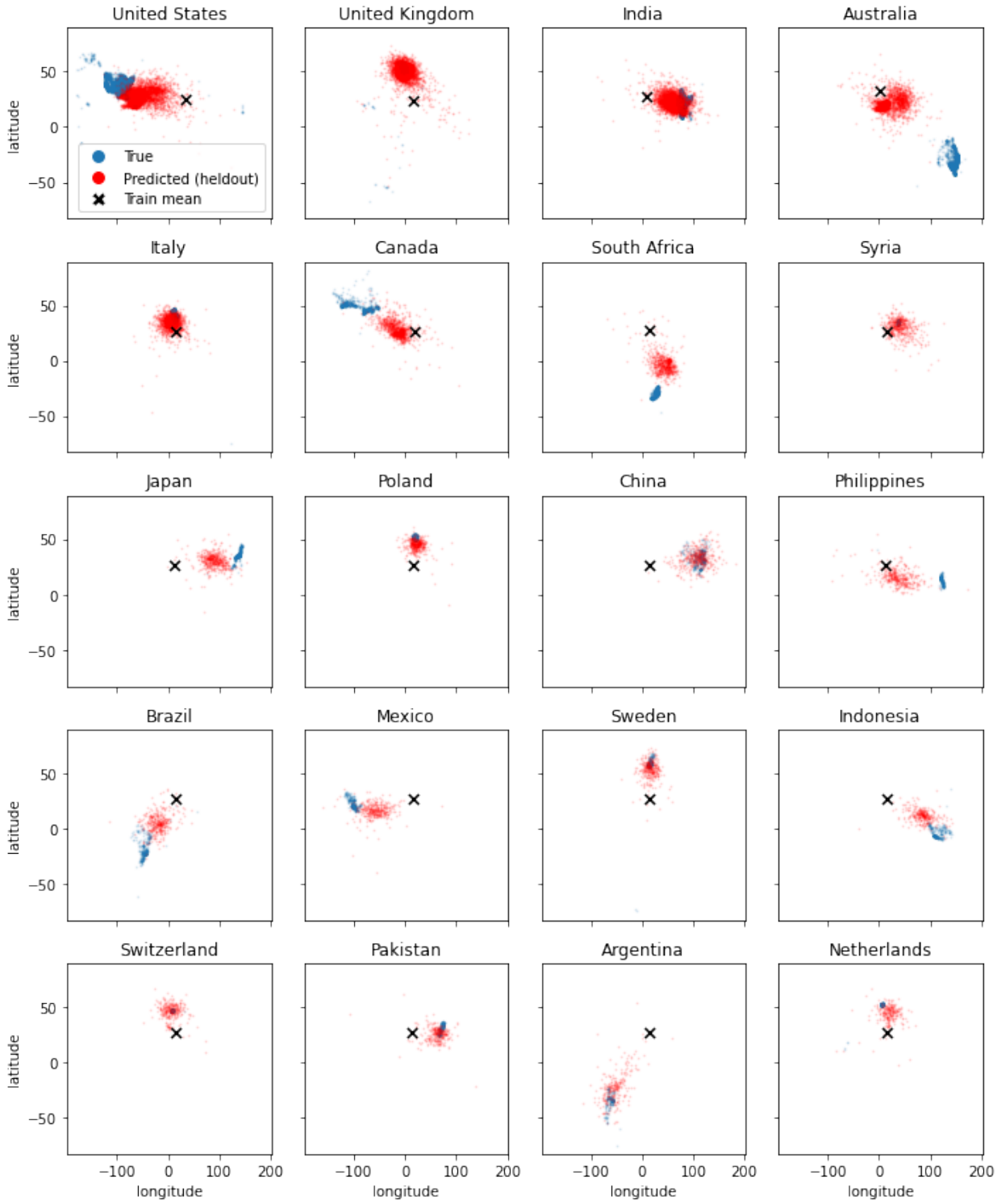


Figure C.6: Out-of-sample predictions for each country when the probe training data contains no samples from the country as compared to true locations and the mean of the training data. The results imply that the learned feature direction correctly generalizes to the relative position of a country but that the probes memorizes the absolute positions.

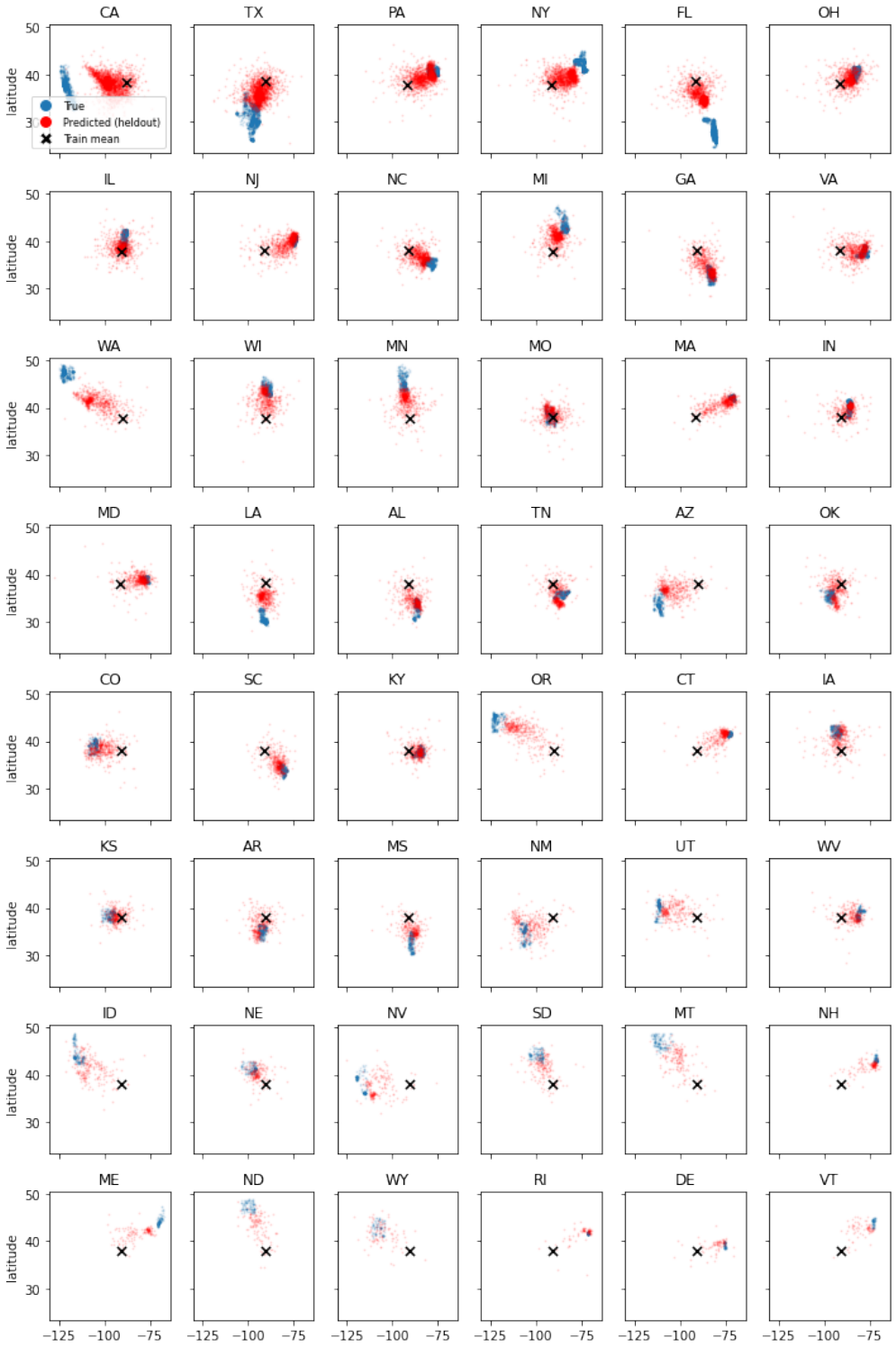


Figure C.7: Out-of-sample predictions for each state when the probe training data contains no samples from the state as compared to true locations and the mean of the training data. The results imply that the learned feature direction correctly generalizes to the relative position of a country but that the probes memorizes the absolute positions.

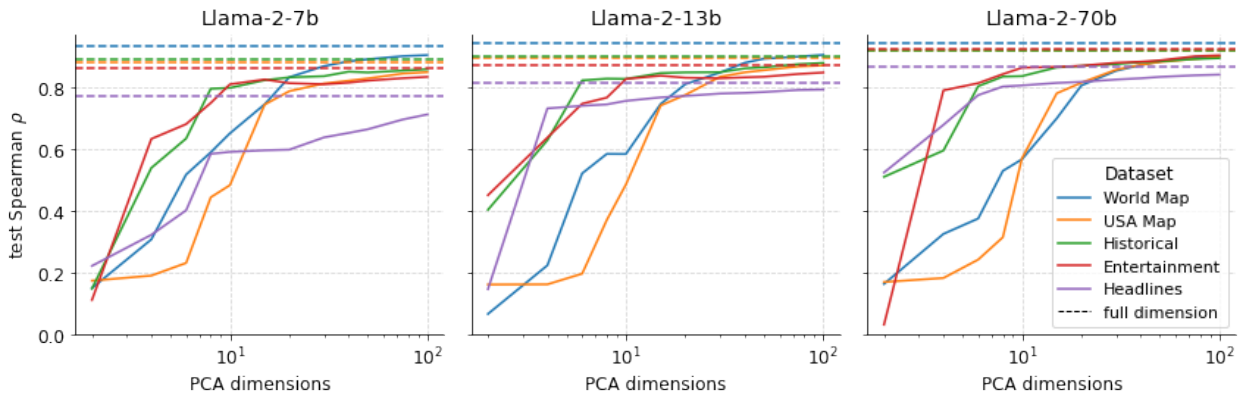


Figure C.8: Test Spearman rank correlation for probes trained on activations projected onto k largest principal components.

Appendix D

Sparse Nonlinear Dynamics Appendix

D.1 Additional Experiment Details

Here we record in greater detail the set of initial conditions we use for each system, the parameter ranges in tuning the various algorithms, and other relevant implementation details. Additionally, in our raw results files made available on Github, we include the initial condition, random seed, and chosen hyperparameters for every trial in every experiment.

D.1.1 Sample Efficiency

For all systems and algorithms we use smoothed finite difference differentiation with a smoothing window length of 9. We use the same regularization grid for all systems. For MIOSR, STLSQ, and SSR we tune over the regularization strength $\alpha \in \{0, 10^{-5}, 10^{-3}, 10^{-2}, 0.05, 0.2\}$. For E-STLSQ we use the best α for STLSQ. For SR3 we tune relaxation parameter $\nu \in \{\frac{1}{30}, \frac{1}{10}, \frac{1}{3}, 1, \frac{10}{3}\}$. For thresholds, we try to tailor the range based on the system to give the best shot at finding a sparse model (since the heuristics are quite sensitive to the threshold). We choose 50 values uniformly in log space. That is, 10^a for $a \in [b : c : d]$ where $d = 50$ values equally spaced on the interval $[b, c]$. In particular we use $[-2 : 1 : 50]$, $[-2 : 0 : 50]$, and $[-1.5 : 1.5 : 50]$ for Lorenz, Hopf, and MHD, respectively (where we increase the range by 0.5 for SR3 since it does not use a hard threshold). For MIOSR, we tune the sparsity k for each dimension over integers $k \in [1, 5]$, and SSR by nature fits a model at every level of sparsity between one and the full library size.

Regarding initial conditions, we sample uniformly from a specified volume. For Hopf, we sample in polar coordinates: a radius uniformly at random between 0.75 and 1.25 and an angle at random from 0 to 2π radians. For Lorenz, we sample $x, y \in [-5, 5]$ and $z \in [10, 40.$ For MHD we sample every coordinate independently from $[-1.5, 1.5]$. We use the same

sampling strategy for the runtime experiment.

D.1.2 Physical Constraints

For both algorithms we use smoothed finite difference differentiation with a smoothing window length of 21 to accommodate the noisier data. As before, for SR3 we tune over $\nu \in \{\frac{1}{30}, \frac{1}{10}, \frac{1}{3}, 1, \frac{10}{3}\}$ and 50 thresholds $\lambda = 10^a,\ a \in [-3:0:20]$. For MIOSR, we tune over a global sparsity constraint $k \in [2, 10]$, and regularizer in $\{0.0001, 0.001, 0.01\}$. As in [67], we sample initial conditions uniformly for each dimension in $[-\pi, \pi]$.

D.1.3 Robustness

For all weak form experiments we normalize the data matrix to have unit column norm. We use the same values of α and ν as before. Again, in an effort to get the best baseline model, we tailor the thresholds to the system and check that all chosen thresholds fall in the range we tune over. For Van der Pol, Lotka, and Rossler respectively we tune the threshold λ over 2^a $a \in [-1:5:50]$, $[-3:4:50]$ and $[2:6:50]$ for STLSQ. For E-STLSQ we use the same regularization that was optimal for STLSQ.

For Van der Pol, we sample initial conditions from the box $x \in [-1, 1],\ y \in [-\mu, \mu]$ where we use $\mu = 3$. For Lotka, we sample initial conditions from the box $x, y \in [0, 1]$. For Rossler, we sample uniformly from the a canonical trajectory with initial condition $(5, 3, 0)$ and add 10% Gaussian noise. For Rossler, we additionally take the absolute value of the z coordinate to preclude unstable trajectories.

D.1.4 PDEs

For achievability analysis, we don't need to tune the sparsity of MIOSR since we can simply use the true sparsity and check if the correct model is learned. However, for (E-)STLSQ the optimal threshold does change because we add substantial noise to the data. Therefore we still train for thresholds in $\lambda \in [0.4, 2.0, 20]$ and $\lambda \in [0.04, 0.16, 20]$ for Kuramoto Sivashinsky and reaction diffusion respectively.

The initial conditions for Kuramoto Sivashinsky are sampled as $\frac{1}{Z}(\cos(x+r_0)+\sin(4r_1x))$ where $r_0, r_1 \in [0, 1]$ are sampled uniformly at random, x is the 1D mesh on $[0, 2\pi]$ with 1024 grid points, and Z is the normalization factor $\|\cos(x + r_0) + \sin(4r_1x))\|_\infty$. For reaction diffusion we use a spiral initial condition that is randomly rotated and slightly expanded or

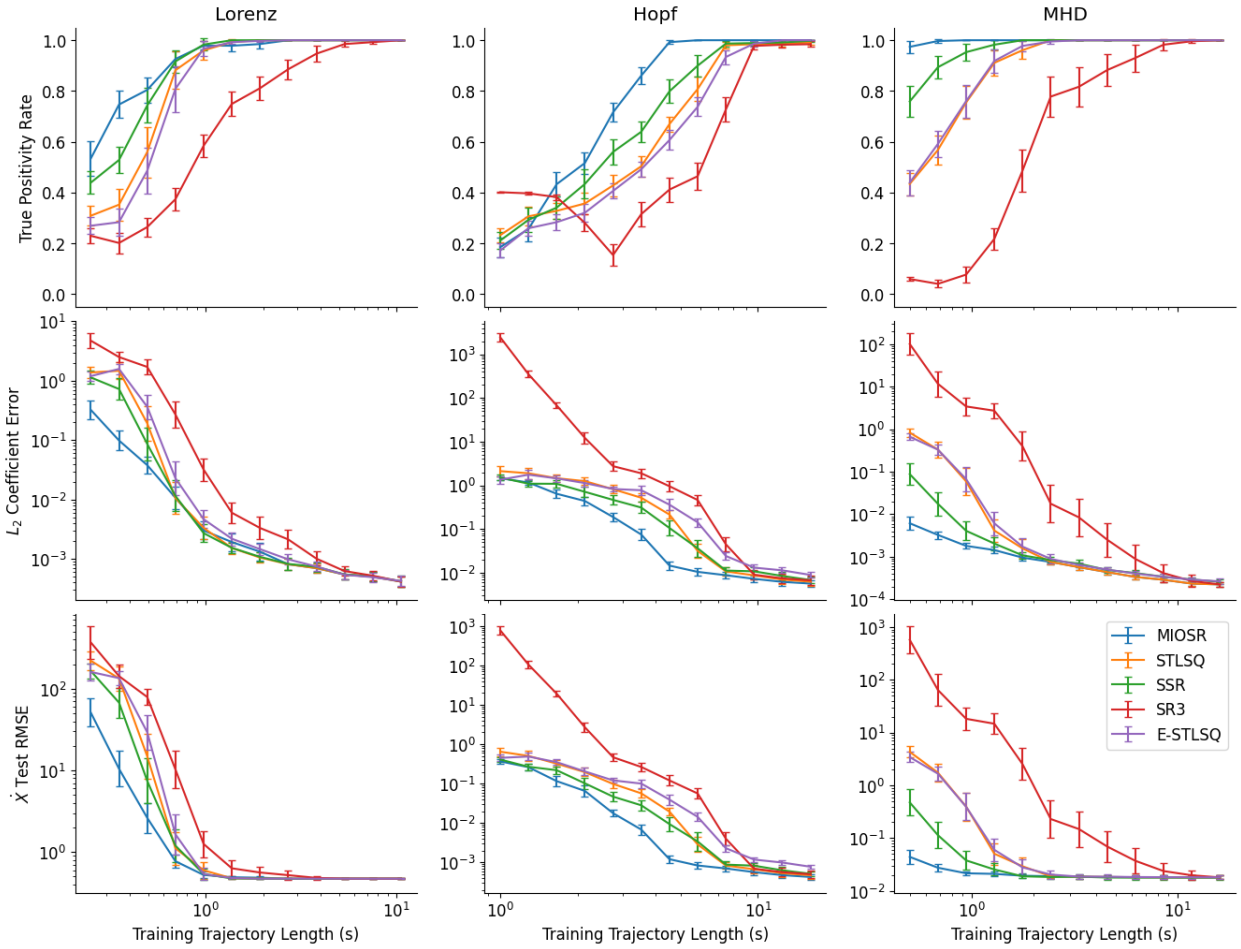


Figure D.1: Performance comparison of sparse regression algorithms for the differential form of SINDy using minimal polynomial libraries under varying amounts of training data for three different canonical systems: Lorenz, Hopf, and MHD. Results are averaged over 50 trials with added Gaussian noise of 0.2%.

contracted. In particular, for X, Y representing the 256×256 spatial mesh

$$u_0 = \tanh((X^2 + Y^2)^{1/2}) \cos((\text{angle}(X + iY) + o) - (s(X^2 + Y^2)^{1/2}))$$

$s \in [0.95, 1.05]$ and $o \in [0, 2\pi]$ both sampled uniformly at random. v_0 is the same but with sin instead of cos.

D.2 Additional Results

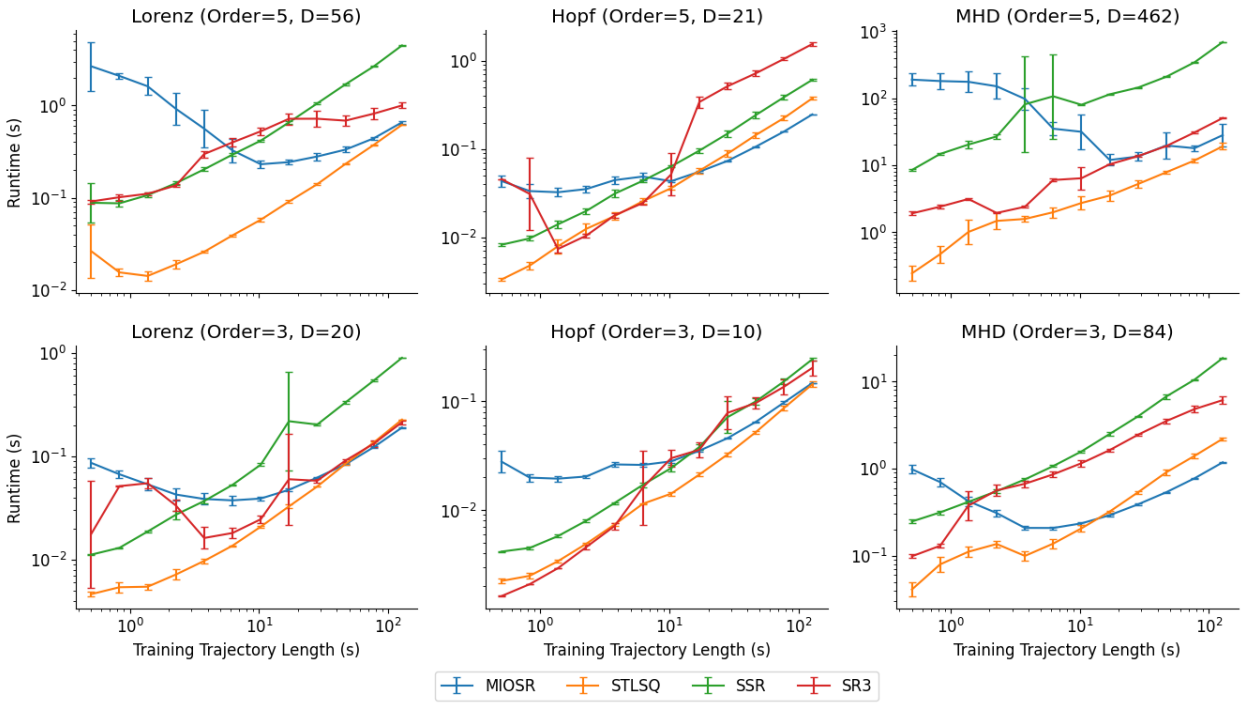


Figure D.2: Comparison of sparse regression algorithm computational efficiency, when executed on a 2021 Macbook Pro, for the differential form of SINDy under varying amounts of training data for three different canonical systems: Lorenz, Hopf, and MHD. The top row uses a fifth order polynomial library for each of the three systems while the bottom row uses a third order polynomial library. Results are averaged over 5 trials with added Gaussian noise of 0.2%

References

[1] Mostafa Abdou, Artur Kulmizev, Daniel Hershcovich, Stella Frank, Ellie Pavlick, and Anders Søgaard. Can language models encode perceptual structure without grounding? a case study in color. *arXiv preprint arXiv:2109.06129*, 2021.

[2] Tobias Achterberg and Roland Wunderling. Mixed integer programming: Analyzing 12 years of progress. In *Facets of Combinatorial Optimization*, pages 449–481. Springer, 2013.

[3] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. *Advances in neural information processing systems*, 31, 2018.

[4] Hirotugu Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, 1974.

[5] Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. *arXiv preprint arXiv:1610.01644*, 2016.

[6] Uri Alon. Design principles of biological circuits. *Biophysical Journal*, 96(3):15a–15a, 2009.

[7] Rana Ali Amjad, Kairen Liu, and Bernhard C Geiger. Understanding neural networks and individual neuron importance via information-ordered cumulative ablation. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[8] Issa Annamoradnejad and Rahimberdi Annamoradnejad. Age dataset: A structured general-purpose dataset on life, work, and death of 1.22 million distinguished people. *International AAAI Conference on Web and Social Media (ICWSM)*, 16, 2022.

[9] Anthropic. Circuits updates - july 2023, 2023. https://transformer-circuits.pub/2023/july-update/index.html.

[10] Omer Antverg and Yonatan Belinkov. On the pitfalls of analyzing individual neurons in language models. *arXiv preprint arXiv:2110.07483*, 2021.

[11] Andy Arditi, Oscar Obeso, Aaquib Syed, Daniel Paleka, Nina Rimsky, Wes Gurnee, and Neel Nanda. Refusal in language models is mediated by a single direction. *arXiv preprint arXiv:2406.11717*, 2024.

[12] Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. Linear algebraic structure of word senses, with applications to polysemy. *Transactions of the Association for Computational Linguistics*, 6:483–495, 2018.

[13] Jack Bandy. Three decades of new york times headlines, 2021. URL https://www.kaggle.com/datasets/johnbandy/new-york-times-headlines. Kaggle dataset.

[14] Yamini Bansal, Preetum Nakkiran, and Boaz Barak. Revisiting Model Stitching to Compare Neural Representations, 2021. URL http://arxiv.org/abs/2106.07682.

[15] Yohai Bar-Sinai, Stephan Hoyer, Jason Hickey, and Michael P Brenner. Learning data-driven discretizations for partial differential equations. *Proceedings of the National Academy of Sciences*, 116(31):15344–15349, 2019.

[16] Serguei Barannikov, Ilya Trofimov, Nikita Balabin, and Evgeny Burnaev. Representation Topology Divergence: A Method for Comparing Neural Network Representations, 2022. URL http://arxiv.org/abs/2201.00058.

[17] Horace Barlow. Redundancy reduction revisited. *Network: computation in neural systems*, 12(3):241, 2001.

[18] Anthony Bau, Yonatan Belinkov, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James Glass. Identifying and controlling important neurons in neural machine translation. *arXiv preprint arXiv:1811.01157*, 2018.

[19] David Bau, Jun-Yan Zhu, Hendrik Strobelt, Agata Lapedriza, Bolei Zhou, and Antonio Torralba. Understanding the role of individual units in a deep neural network. *Proceedings of the National Academy of Sciences*, 2020. ISSN 0027-8424. doi:10.1073/pnas.1907375117. URL https://www.pnas.org/content/early/2020/08/31/1907375117.

[20] Yonatan Belinkov. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1):207–219, 2022.

[21] Nora Belrose, Zach Furman, Logan Smith, Danny Halawi, Igor Ostrovsky, Lev McKinney, Stella Biderman, and Jacob Steinhardt. Eliciting latent predictions from transformers with the tuned lens. *arXiv preprint arXiv:2303.08112*, 2023.

[22] Emily M Bender and Alexander Koller. Climbing towards nlu: On meaning, form, and understanding in the age of data. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 5185–5198, 2020.

[23] Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, pages 610–623, 2021.

[24] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[25] Yoshua Bengio, Geoffrey Hinton, Andrew Yao, Dawn Song, Pieter Abbeel, Yuval Noah Harari, Ya-Qin Zhang, Lan Xue, Shai Shalev-Shwartz, Gillian Hadfield, et al. Managing ai risks in an era of rapid progress. *arXiv preprint arXiv:2310.17688*, 2023.

[26] Gal Berkooz, Philip Holmes, and John L Lumley. The proper orthogonal decomposition in the analysis of turbulent flows. *Annual review of fluid mechanics*, 25(1):539–575, 1993.

[27] Dimitris Bertsimas and Martin S Copenhaver. Characterization of the equivalence of robustification and regularization in linear and matrix regression. *European Journal of Operational Research*, 270(3):931–942, 2018.

[28] Dimitris Bertsimas and Vassilis Digalakis. The backbone method for ultra-high dimensional sparse machine learning. *Machine Learning*, pages 1–52, 2022.

[29] Dimitris Bertsimas and Jack Dunn. *Machine learning under a modern optimization lens*. Dynamic Ideas LLC, 2019.

[30] Dimitris Bertsimas and Wes Gurnee. Learning sparse nonlinear dynamics via mixed-integer optimization. *Nonlinear Dynamics*, 111(7):6585–6604, 2023.

[31] Dimitris Bertsimas and Michael Lingzhi Li. Scalable holistic linear regression. *Operations Research Letters*, 48(3):203–208, May 2020. ISSN 0167-6377.

doi:10.1016/j.orl.2020.02.008. URL https://www.sciencedirect.com/science/article/pii/S0167637720300262.

[32] Dimitris Bertsimas and Robert Weismantel. *Optimization over integers*, volume 13. Dynamic Ideas, 2005.

[33] Dimitris Bertsimas, Angela King, and Rahul Mazumder. Best subset selection via a modern optimization lens. *The Annals of Statistics*, 44(2):813–852, 2016.

[34] Dimitris Bertsimas, Jean Pauphilet, and Bart Van Parys. Sparse regression: Scalable algorithms and empirical performance. 2020.

[35] Dimitris Bertsimas, Jean Pauphilet, and Bart Van Parys. Sparse regression: Scalable algorithms and empirical performance. *Statistical Science*, 35(4):555–578, 2020.

[36] Dimitris Bertsimas, Jean Pauphilet, and Bart Van Parys. Sparse classification: a scalable discrete optimization perspective. *Machine Learning*, 110:3177–3209, 2021.

[37] Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar van der Wal. Pythia: A suite for analyzing large language models across training and scaling, 2023.

[38] Steven Bills, Nick Cammarata, Dan Mossing, Henk Tillman, Leo Gao, Gabriel Goh, Ilya Sutskever, Jan Leike, Jeff Wu, and William Saunders. Language models can explain neurons in language models. https://openaipublic.blob.core.windows.net/neuron-explainer/paper/index.html, 2023.

[39] Blair Bilodeau, Natasha Jaques, Pang Wei Koh, and Been Kim. Impossibility theorems for feature attribution. *arXiv preprint arXiv:2212.11870*, 2022.

[40] Yonatan Bisk, Ari Holtzman, Jesse Thomason, Jacob Andreas, Yoshua Bengio, Joyce Chai, Mirella Lapata, Angeliki Lazaridou, Jonathan May, Aleksandr Nisnevich, et al. Experience grounds language. *arXiv preprint arXiv:2004.10151*, 2020.

[41] Robert E Bixby. Solving real-world linear programs: A decade and more of progress. *Operations research*, 50(1):3–15, 2002.

[42] Sid Black, Lee Sharkey, Leo Grinsztajn, Eric Winsor, Dan Braun, Jacob Merizian, Kip Parker, Carlos Ramón Guevara, Beren Millidge, Gabriel Alfour, and Connor Leahy. Interpreting neural networks through the polytope lens, 2022.

[43] Enric Boix-Adsera, Hannah Lawrence, George Stepaniants, and Philippe Rigollet. GULP: a prediction-based metric between representations, 2022. URL http://arxiv.org/abs/2210.06545.

[44] Tolga Bolukbasi, Adam Pearce, Ann Yuan, Andy Coenen, Emily Reif, Fernanda Viégas, and Martin Wattenberg. An interpretability illusion for bert. *arXiv preprint arXiv:2104.07143*, 2021.

[45] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[46] Josh Bongard and Hod Lipson. Automated reverse engineering of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 104(24):9943–9948, 2007.

[47] Lorenzo Boninsegna, Feliks Nüske, and Cecilia Clementi. Sparse learning of stochastic dynamical equations. *The Journal of Chemical Physics*, 148(24):241723, June 2018. ISSN 0021-9606. doi:10.1063/1.5018409. URL https://aip.scitation.org/doi/10.1063/1.5018409.

[48] Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Zac Hatfield-Dodds, Alex Tamkin, Karina Nguyen, Brayden McLean, Josiah E Burke, Tristan Hume, Shan Carter, Tom Henighan, and Christopher Olah. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023. https://transformer-circuits.pub/2023/monosemantic-features/index.html.

[49] Shaked Brody, Uri Alon, and Eran Yahav. On the expressivity role of layernorm in transformers' attention. *arXiv preprint arXiv:2305.02582*, 2023.

[50] Davis Brown, Nikhil Vyas, and Yamini Bansal. On privileged and convergent bases in neural network representations. *arXiv preprint arXiv:2307.12941*, 2023.

[51] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[52] Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the national academy of sciences*, 113(15):3932–3937, 2016.

[53] Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Sparse identification of nonlinear dynamics with control (sindyc). *IFAC-PapersOnLine*, 49(18):710–715, 2016.

[54] Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937, April 2016. ISSN 0027-8424, 1091-6490. doi:10.1073/pnas.1517384113. URL http://www.pnas.org/lookup/doi/10.1073/pnas.1517384113.

[55] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023.

[56] Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. Discovering latent knowledge in language models without supervision. *arXiv preprint arXiv:2212.03827*, 2022.

[57] György Buzsáki and Rodolfo Llinás. Space and time in the brain. *Science*, 358(6362): 482–485, 2017.

[58] Nick Cammarata, Gabriel Goh, Shan Carter, Chelsea Voss, Ludwig Schubert, and Chris Olah. Curve circuits. *Distill*, 2021. doi:10.23915/distill.00024.006. https://distill.pub/2020/circuits/curve-circuits.

[59] Steven Cao, Victor Sanh, and Alexander M Rush. Low-complexity probing via finding subnetworks. *arXiv preprint arXiv:2104.03514*, 2021.

[60] V Carbone and P Veltri. Relaxation processes in magnetohydrodynamics-a triad-interaction model. *Astronomy and Astrophysics*, 259:359–372, 1992.

[61] Alejandro Carderera, Sebastian Pokutta, Christof Schütte, and Martin Weiser. CINDy: Conditional gradient-based Identification of Non-linear Dynamics – Noise-robust recovery. *arXiv:2101.02630*, 2021. URL http://arxiv.org/abs/2101.02630.

[62] Joe Carlsmith. Scheming ais: Will ais fake alignment during training in order to get power? *arXiv preprint arXiv:2311.08379*, 2023.

[63] Joseph Carlsmith. Is power-seeking ai an existential risk? *arXiv preprint arXiv:2206.13353*, 2022.

[64] Stephen Casper, Tilman Rauker, Anson Ho, and Dylan Hadfield-Menell. Sok: Toward transparent ai: A survey on interpreting the inner structures of deep neural networks. In *First IEEE Conference on Secure and Trustworthy Machine Learning*.

[65] Stephen Casper, Xavier Boix, Vanessa D'Amario, Ling Guo, Martin Schrimpf, Kasper Vinken, and Gabriel Kreiman. Frivolous units: Wider networks are not really that wide. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 6921–6929, 2021.

[66] Kathleen Champion, Bethany Lusch, J Nathan Kutz, and Steven L Brunton. Data-driven discovery of coordinates and governing equations. *Proceedings of the National Academy of Sciences*, 116(45):22445–22451, 2019.

[67] Kathleen Champion, Peng Zheng, Aleksandr Y Aravkin, Steven L Brunton, and J Nathan Kutz. A unified sparse optimization framework to learn parsimonious physics-informed models from data. *IEEE Access*, 8:169259–169271, 2020.

[68] Lawrence Chan, Leon Lang, and Erik Jenner. Natural abstractions: Key claims, theorems, and critiques, 2022. https://www.alignmentforum.org/posts/z6QQJbtpkEAX3Aojj.

[69] Anindya Chatterjee. An introduction to the proper orthogonal decomposition. *Current Science*, pages 808–817, 2000.

[70] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. *Advances in neural information processing systems*, 29, 2016.

[71] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

[72] Bilal Chughtai, Lawrence Chan, and Neel Nanda. A toy model of universality: Reverse engineering how networks learn group operations. *arXiv preprint arXiv:2302.03025*, 2023.

[73] Andy Coenen, Emily Reif, Ann Yuan, Been Kim, Adam Pearce, Fernanda Viégas, and Martin Wattenberg. Visualizing and measuring the geometry of bert, 2019.

[74] Arthur Conmy, Augustine N Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. Towards automated circuit discovery for mechanistic interpretability. *arXiv preprint arXiv:2304.14997*, 2023.

[75] Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. What you can cram into a single vector: Probing sentence embeddings for linguistic properties, 2018.

[76] IBM ILOG Cplex. V12. 8: User's manual for cplex. *International Business Machines Corporation*, 46(53):157, 2017.

[77] Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models. *arXiv preprint arXiv:2309.08600*, 2023.

[78] Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. Knowledge neurons in pretrained transformers. *arXiv preprint arXiv:2104.08696*, 2021.

[79] Fahim Dalvi, Nadir Durrani, Hassan Sajjad, Yonatan Belinkov, Anthony Bau, and James Glass. What is one grain of sand in the desert? analyzing individual neurons in deep nlp models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6309–6317, 2019.

[80] Fahim Dalvi, Hassan Sajjad, Nadir Durrani, and Yonatan Belinkov. Analyzing redundancy in pretrained transformer models. *arXiv preprint arXiv:2004.04010*, 2020.

[81] Guy Dar, Mor Geva, Ankit Gupta, and Jonathan Berant. Analyzing transformers in embedding space. *arXiv preprint arXiv:2209.02535*, 2022.

[82] Nicola De Cao, Leon Schmid, Dieuwke Hupkes, and Ivan Titov. Sparse interventions in language models with differentiable masking. *arXiv preprint arXiv:2112.06837*, 2021.

[83] Brian M. de Silva, Kathleen Champion, Markus Quade, Jean-Christophe Loiseau, J. Nathan Kutz, and Steven L. Brunton. PySINDy: A Python package for the Sparse Identification of Nonlinear Dynamics from Data. *arXiv:2004.08424 [physics]*, April 2020. URL http://arxiv.org/abs/2004.08424. arXiv: 2004.08424.

[84] Charles B Delahunt and J Nathan Kutz. A toolkit for data-driven discovery of governing equations in high-noise regimes. *arXiv preprint arXiv:2111.04870*, 2021.

[85] Kedar Dhamdhere, Mukund Sundararajan, and Qiqi Yan. How important is a neuron? *arXiv preprint arXiv:1805.12233*, 2018.

[86] Frances Ding, Jean-Stanislas Denain, and Jacob Steinhardt. Grounding Representation Similarity with Statistical Testing, 2021. URL http://arxiv.org/abs/2108.01661.

[87] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*, pages 647–655. PMLR, 2014.

[88] Jonathan Donnelly and Adam Roegiest. On interpretability and feature representations: an analysis of the sentiment neuron. In *Advances in Information Retrieval: 41st European Conference on IR Research, ECIR 2019, Cologne, Germany, April 14–18, 2019, Proceedings, Part I 41*, pages 795–802. Springer, 2019.

[89] David L Donoho. Compressed sensing. *IEEE Transactions on information theory*, 52 (4):1289–1306, 2006.

[90] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.

[91] Amil Dravid, Yossi Gandelsman, Alexei A Efros, and Assaf Shocher. Rosetta neurons: Mining the common units in a model zoo. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1934–1943, 2023.

[92] Lyndon R. Duong, Jingyang Zhou, Josue Nassar, Jules Berman, Jeroen Olieslagers, and Alex H. Williams. Representational dissimilarity metric spaces for stochastic neural networks, 2023. URL http://arxiv.org/abs/2211.11665.

[93] Nadir Durrani, Hassan Sajjad, Fahim Dalvi, and Yonatan Belinkov. Analyzing individual neurons in pre-trained language models. *arXiv preprint arXiv:2010.02695*, 2020.

[94] Nadir Durrani, Fahim Dalvi, and Hassan Sajjad. Linguistic correlation analysis: Discovering salient neurons in deepnlp models. *arXiv preprint arXiv:2206.13288*, 2022.

[95] Yanai Elazar, Shauli Ravfogel, Alon Jacovi, and Yoav Goldberg. Amnesic probing: Behavioral explanation with amnesic counterfactuals. *Transactions of the Association for Computational Linguistics*, 9:160–175, 2021.

[96] N Elhage, N Nanda, C Olsson, T Henighan, N Joseph, B Mann, A Askell, Y Bai, A Chen, T Conerly, et al. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021.

[97] Nelson Elhage, Tristan Hume, Catherine Olsson, Neel Nanda, Tom Henighan, Scott Johnston, Sheer ElShowk, Nicholas Joseph, Nova DasSarma, Ben Mann, Danny Hernandez, Amanda Askell, Kamal Ndousse, Dawn Drain, Anna Chen, Yuntao Bai, Deep Ganguli, Liane Lovitt, Zac Hatfield-Dodds, Jackson Kernion, Tom Conerly, Shauna Kravec, Stanislav Fort, Saurav Kadavath, Josh Jacobson, Eli Tran-Johnson, Jared Kaplan, Jack Clark, Tom Brown, Sam McCandlish, Dario Amodei, and Christopher Olah. Softmax linear units. *Transformer Circuits Thread*, 2022. https://transformer-circuits.pub/2022/solu/index.html.

[98] Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, et al. Toy models of superposition. *arXiv preprint arXiv:2209.10652*, 2022.

[99] Nelson Elhage, Robert Lasenby, and Christopher Olah. Privileged bases in the transformer residual stream. *Transformer Circuits Thread*, 2023. https://transformer-circuits.pub/2023/privileged-basis/index.html.

[100] Joshua Engels, Isaac Liao, Eric J Michaud, Wes Gurnee, and Max Tegmark. Not all language model features are linear. *arXiv preprint arXiv:2405.14860*, 2024.

[101] Urban Fasel, J Nathan Kutz, Bingni W Brunton, and Steven L Brunton. Ensemble-sindy: Robust sparse model discovery in the low-data, high-noise limit, with active learning and control. *Proceedings of the Royal Society A*, 478(2260):20210904, 2022.

[102] Jiahai Feng and Jacob Steinhardt. How do language models bind entities in context? *arXiv preprint arXiv:2310.17191*, 2023.

[103] R Fuentes, N Dervilis, K Worden, and Elizabeth J Cross. Efficient parameter identification and model selection in nonlinear dynamical systems via sparse bayesian learning. In *Journal of Physics: Conference Series*, volume 1264, page 012050. IOP Publishing, 2019.

[104] Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

[105] Leo Gao, Tom Dupré la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. Scaling and evaluating sparse autoencoders. *arXiv preprint arXiv:2406.04093*, 2024.

[106] Nikhil Garg, Wes Gurnee, David Rothschild, and David Shmoys. Combatting gerrymandering with social choice: The design of multi-member districts. In *Proceedings of the 23rd ACM Conference on Economics and Computation*, pages 560–561, 2022.

[107] Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. *arXiv preprint arXiv:2012.14913*, 2020.

[108] Mor Geva, Avi Caciularu, Kevin Ro Wang, and Yoav Goldberg. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. *arXiv preprint arXiv:2203.14680*, 2022.

[109] Mor Geva, Jasmijn Bastings, Katja Filippova, and Amir Globerson. Dissecting recall of factual associations in auto-regressive language models. *arXiv preprint arXiv:2304.14767*, 2023.

[110] Charles Godfrey, Davis Brown, Tegan Emerson, and Henry Kvinge. On the Symmetries of Deep Learning Models and their Internal Representations, 2023. URL http://arxiv.org/abs/2205.14258.

[111] Gabriel Goh, Nick Cammarata, Chelsea Voss, Shan Carter, Michael Petrov, Ludwig Schubert, Alec Radford, and Chris Olah. Multimodal neurons in artificial neural networks. *Distill*, 6(3):e30, 2021.

[112] Nicholas Goldowsky-Dill, Chris MacLeod, Lucas Sato, and Aryaman Arora. Localizing model behavior with path patching. *arXiv preprint arXiv:2304.05969*, 2023.

[113] Rhys Gould, Euan Ong, George Ogden, and Arthur Conmy. Successor heads: Recurring, interpretable attention heads in the wild. *arXiv preprint arXiv:2312.09230*, 2023.

[114] Abhijeet Gupta, Gemma Boleda, Marco Baroni, and Sebastian Padó. Distributional vectors encode referential attributes. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 12–21, 2015.

[115] Daniel R Gurevich, Patrick AK Reinbold, and Roman O Grigoriev. Robust and optimal sparse regression for nonlinear pde models. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 29(10):103113, 2019.

[116] Wes Gurnee and David B Shmoys. Fairmandering: A column generation heuristic for fairness-optimized political districting. In *SIAM Conference on Applied and Computational Discrete Algorithms (ACDA21)*, pages 88–99. SIAM, 2021.

[117] Wes Gurnee and Max Tegmark. Language models represent space and time. *arXiv preprint arXiv:2310.02207*, 2023.

[118] Wes Gurnee, Neel Nanda, Matthew Pauly, Katherine Harvey, Dmitrii Troitskii, and Dimitris Bertsimas. Finding neurons in a haystack: Case studies with sparse probing. *arXiv preprint arXiv:2305.01610*, 2023.

[119] Wes Gurnee, Theo Horsley, Zifan Carl Guo, Tara Rezaei Kheirkhah, Qinyi Sun, Will Hathaway, Neel Nanda, and Dimitris Bertsimas. Universal neurons in gpt2 language models. *arXiv preprint arXiv:2401.12181*, 2024.

[120] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2021. URL https://www.gurobi.com.

[121] Matthew Gwilliam and Abhinav Shrivastava. Beyond Supervised vs. Unsupervised: Representative Benchmarking and Analysis of Image Representation Learning, 2022. URL http://arxiv.org/abs/2206.08347.

[122] Torkel Hafting, Marianne Fyhn, Sturla Molden, May-Britt Moser, and Edvard I Moser. Microstructure of a spatial map in the entorhinal cortex. *Nature*, 436(7052):801–806, 2005.

[123] William L. Hamilton, Jure Leskovec, and Dan Jurafsky. Diachronic Word Embeddings Reveal Statistical Laws of Semantic Change, 2018. URL http://arxiv.org/abs/1605.09096.

[124] Jessica Hamrick and Shakir Mohamed. Levels of analysis for machine learning. *arXiv preprint arXiv:2004.05107*, 2020.

[125] Michael Hanna, Ollie Liu, and Alexandre Variengien. How does gpt-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model. *arXiv preprint arXiv:2305.00586*, 2023.

[126] Peter Hase, Mohit Bansal, Been Kim, and Asma Ghandeharioun. Does localization inform editing? surprising differences in causality-based localization vs. knowledge editing in language models. *arXiv preprint arXiv:2301.04213*, 2023.

[127] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

[128] Hussein Hazimeh, Rahul Mazumder, and Ali Saab. Sparse regression at scale: branch-and-bound rooted in first-order optimization. *Mathematical Programming*, October 2021. ISSN 0025-5610, 1436-4646. doi:10.1007/s10107-021-01712-4. URL https://link.springer.com/10.1007/s10107-021-01712-4.

[129] Roee Hendel, Mor Geva, and Amir Globerson. In-context learning creates task vectors. *arXiv preprint arXiv:2310.15916*, 2023.

[130] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

[131] Dan Hendrycks, Mantas Mazeika, and Thomas Woodside. An overview of catastrophic ai risks. *arXiv preprint arXiv:2306.12001*, 2023.

[132] Tom Henighan, Shan Carter, Tristan Hume, Nelson Elhage, Robert Lasenby, Stanislav Fort, Nicholas Schiefer, and Christopher Olah. Superposition, memorization, and double descent. *Transformer Circuits Thread*, 2023. https://transformer-circuits.pub/2023/toy-double-descent/index.html.

[133] Lucas Torroba Hennigen, Adina Williams, and Ryan Cotterell. Intrinsic probing through dimension selection. *arXiv preprint arXiv:2010.02812*, 2020.

[134] Evan Hernandez, Arnab Sen Sharma, Tal Haklay, Kevin Meng, Martin Wattenberg, Jacob Andreas, Yonatan Belinkov, and David Bau. Linearity of relation decoding in transformer language models. *arXiv preprint arXiv:2308.09124*, 2023.

[135] J Hewitt and P Liang. Designing and interpreting probes with control tasks. *Proceedings of the 2019 Con*, 2019.

[136] Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. spacy: Industrial-strength natural language processing in python. 2020. doi:10.5281/zenodo.1212303.

[137] Tomáš Hromádka, Michael R DeWeese, and Anthony M Zador. Sparse representation of sounds in the unanesthetized auditory cortex. *PLoS biology*, 6(1):e16, 2008.

[138] Andrew Hryniowski and Alexander Wong. Inter-layer Information Similarity Assessment of Deep Neural Networks Via Topological Similarity and Persistence Analysis of Data Neighbour Dynamics, 2020. URL http://arxiv.org/abs/2012.03793.

[139] Jing Huang, Atticus Geiger, Karel D'Oosterlinck, Zhengxuan Wu, and Christopher Potts. Rigorously assessing natural language explanations of neurons. *arXiv preprint arXiv:2309.10312*, 2023.

[140] Stanisław Jastrzębski, Devansh Arpit, Nicolas Ballas, Vikas Verma, Tong Che, and Yoshua Bengio. Residual connections encourage iterative inference. *arXiv preprint arXiv:1710.04773*, 2017.

[141] Adam Jermyn, Chris Olah, and Tom Henighan. Circuits updates — may 2023: Attention head superposition, 2023. https://transformer-circuits.pub/2023/may-update/index.html#attention-superposition.

[142] Adam S Jermyn, Nicholas Schiefer, and Evan Hubinger. Engineering monosemanticity in toy models. *arXiv preprint arXiv:2211.09169*, 2022.

[143] Brett A Johnson and Michael Leon. Modular representations of odorants in the glomerular layer of the rat olfactory bulb and the effects of stimulus concentration. *Journal of Comparative Neurology*, 422(4):496–509, 2000.

[144] Kadierdan Kaheman, J. Nathan Kutz, and Steven L. Brunton. SINDy-PI: A Robust Algorithm for Parallel Implicit Sparse Identification of Nonlinear Dynamics. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 476(2242): 20200279, October 2020. ISSN 1364-5021, 1471-2946. doi:10.1098/rspa.2020.0279. URL http://arxiv.org/abs/2004.02322. arXiv: 2004.02322.

[145] E. Kaiser, J. N. Kutz, and S. L. Brunton. Sparse identification of nonlinear dynamics for model predictive control in the low-data limit. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 474(2219):20180335, November 2018. ISSN 1364-5021, 1471-2946. doi:10.1098/rspa.2018.0335. URL https://royalsocietypublishing.org/doi/10.1098/rspa.2018.0335.

[146] Alan A. Kaptanoglu, Kyle D. Morgan, Chris J. Hansen, and Steven L. Brunton. Physics-constrained, low-dimensional models for MHD: First-principles and data-driven approaches. *Physical Review E*, 104(1):015206, July 2021. ISSN 2470-0045, 2470-0053. doi:10.1103/PhysRevE.104.015206. URL http://arxiv.org/abs/2004.10389. arXiv: 2004.10389.

[147] Alan A. Kaptanoglu, Brian M. de Silva, Urban Fasel, Kadierdan Kaheman, Andy J. Goldschmidt, Jared Callaham, Charles B. Delahunt, Zachary G. Nicolaou, Kathleen Champion, Jean-Christophe Loiseau, J. Nathan Kutz, and Steven L. Brunton. Pysindy: A comprehensive python package for robust sparse system identification. *Journal of Open Source Software*, 7(69):3994, 2022. doi:10.21105/joss.03994. URL https://doi.org/10.21105/joss.03994.

[148] Siddharth* Karamcheti, Laurel* Orr, Jason Bolton, Tianyi Zhang, Karan Goel, Avanika Narayan, Rishi Bommasani, Deepak Narayanan, Tatsunori Hashimoto, Dan Jurafsky, Christopher D. Manning, Christopher Potts, Christopher Ré, and Percy Liang. Mistral - a journey towards reproducible language model training, 2021. URL https://github.com/stanford-crfm/mistral.

[149] Amir-Hossein Karimi, Krikamol Muandet, Simon Kornblith, Bernhard Schölkopf, and Been Kim. On the relationship between explanation and prediction: A causal view. *arXiv preprint arXiv:2212.06925*, 2022.

[150] Valentin Khrulkov and Ivan Oseledets. Geometry Score: A Method For Comparing Generative Adversarial Networks, 2018. URL http://arxiv.org/abs/1802.02664.

[151] Hyunjik Kim and Andriy Mnih. Disentangling by factorising. In *International Conference on Machine Learning*, pages 2649–2658. PMLR, 2018.

[152] Max Klabunde, Tobias Schumacher, Markus Strohmaier, and Florian Lemmerich. Similarity of Neural Network Models: A Survey of Functional and Representational Measures, 2023. URL http://arxiv.org/abs/2305.06329.

[153] Philipp Koehn. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of machine translation summit x: papers*, pages 79–86, 2005.

[154] Michal Konkol, Tomáš Brychcín, Michal Nykl, and Tomáš Hercig. Geographical evaluation of word embeddings. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 224–232, 2017.

[155] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International conference on machine learning*, pages 3519–3529. PMLR, 2019.

[156] Dennis Kreber. Cardinality-constrained discrete optimization for regression. 2019.

[157] Jan Kronqvist, David E Bernal, Andreas Lundell, and Ignacio E Grossmann. A review and comparison of solvers for convex minlp. *Optimization and Engineering*, 20(2): 397–455, 2019.

[158] J Nathan Kutz and Steven L Brunton. Parsimony as the ultimate regularizer for physics-informed machine learning. *Nonlinear Dynamics*, pages 1–17, 2022.

[159] Vedang Lad, Wes Gurnee, and Max Tegmark. The remarkable robustness of llms: Stages of inference? *arXiv preprint arXiv:2406.19384*, 2024.

[160] Richard D. Lange, David S. Rolnick, and Konrad P. Kording. Clustering units in neural networks: upstream vs downstream information, 2022. URL http://arxiv.org/abs/2203.11815.

[161] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. Dbpedia - a large-scale, multilingual knowledge base extracted from wikipedia, 2015. URL http://dbpedia.org. Version 2023.

[162] Stefan Leutgeb, Jill K Leutgeb, Carol A Barnes, Edvard I Moser, Bruce L McNaughton, and May-Britt Moser. Independent codes for spatial and episodic memory in hippocampal neuronal ensembles. *Science*, 309(5734):619–623, 2005.

[163] Belinda Z Li, Maxwell Nye, and Jacob Andreas. Implicit representations of meaning in neural language models. *arXiv preprint arXiv:2106.00737*, 2021.

[164] Kenneth Li, Aspen K Hopkins, David Bau, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Emergent world representations: Exploring a sequence model trained on a synthetic task. *arXiv preprint arXiv:2210.13382*, 2022.

[165] Yixuan Li, Jason Yosinski, Jeff Clune, Hod Lipson, and John Hopcroft. Convergent learning: Do different neural networks learn the same representations? *arXiv preprint arXiv:1511.07543*, 2015.

[166] Yixuan Li, Jason Yosinski, Jeff Clune, Hod Lipson, and John Hopcroft. Convergent Learning: Do different neural networks learn the same representations?, February 2016. URL http://arxiv.org/abs/1511.07543.

[167] Isaac Liao, Ziming Liu, and Max Tegmark. Generating interpretable networks using hypernetworks. *arXiv preprint arXiv:2312.03051*, 2023.

[168] Bastien Liétard, Mostafa Abdou, and Anders Søgaard. Do language models know the way to rome? *arXiv preprint arXiv:2109.07971*, 2021.

[169] Jia Lim and Hady Lauw. Disentangling transformer language models as superposed topic models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8646–8666, 2023.

[170] Baihan Lin. Geometric and Topological Inference for Deep Representations of Complex Networks. In *Companion Proceedings of the Web Conference 2022*, pages 334–338, 2022. doi:10.1145/3487553.3524194. URL http://arxiv.org/abs/2203.05488.

[171] Leo Z Liu, Yizhong Wang, Jungo Kasai, Hannaneh Hajishirzi, and Noah A Smith. Probing across time: What does roberta know and when? *arXiv preprint arXiv:2104.07885*, 2021.

[172] Toni JB Liu, Nicolas Boullé, Raphaël Sarfati, and Christopher J Earls. Llms learn governing principles of dynamical systems, revealing an in-context neural scaling law. *arXiv preprint arXiv:2402.00795*, 2024.

[173] Jean-Christophe Loiseau and Steven L Brunton. Constrained sparse galerkin regression. *Journal of Fluid Mechanics*, 838:42–67, 2018.

[174] Edward N Lorenz. Deterministic nonperiodic flow. *Journal of Atmospheric Sciences*, 20(2):130–141, 1963.

[175] Alfred James Lotka. *Elements of physical biology*. Williams & Wilkins, 1925.

[176] Max M Louwerse and Nick Benesh. Representing spatial structure through maps and language: Lord of the rings encodes the spatial structure of middle earth. *Cognitive science*, 36(8):1556–1569, 2012.

[177] Max M Louwerse and Rolf A Zwaan. Language encodes geographical information. *Cognitive Science*, 33(1):51–73, 2009.

[178] Yao Lu, Wen Yang, Yunzhe Zhang, Zuohui Chen, Jinyin Chen, Qi Xuan, Zhen Wang, and Xiaoniu Yang. Understanding the Dynamics of DNNs Using Graph Modularity, 2022. URL http://arxiv.org/abs/2111.12485.

[179] Bethany Lusch, J Nathan Kutz, and Steven L Brunton. Deep learning for universal linear embeddings of nonlinear dynamics. *Nature communications*, 9(1):1–10, 2018.

[180] Suryanarayana Maddu, Bevan L. Cheeseman, Ivo F. Sbalzarini, and Christian L. Müller. Stability selection enables robust learning of partial differential equations from limited noisy data. *arXiv:1907.07810*, July 2019. URL http://arxiv.org/abs/1907.07810.

[181] N. M. Mangan, J. N. Kutz, S. L. Brunton, and J. L. Proctor. Model selection for dynamical systems via sparse regression and information criteria. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 473(2204): 20170009, August 2017. ISSN 1364-5021, 1471-2946. doi:10.1098/rspa.2017.0009. URL https://royalsocietypublishing.org/doi/10.1098/rspa.2017.0009.

[182] David Marr. *Vision: A computational investigation into the human representation and processing of visual information.* MIT press, 2010.

[183] Jerrold E Marsden and Marjorie McCracken. *The Hopf bifurcation and its applications*, volume 19. Springer Science & Business Media, 2012.

[184] Rowan Hall Maudslay and Ryan Cotterell. Do syntactic probes probe syntax? experiments with jabberwocky probing. *arXiv preprint arXiv:2106.02559*, 2021.

[185] Callum McDougall, Arthur Conmy, Cody Rushing, Thomas McGrath, and Neel Nanda. Copy suppression: Comprehensively understanding an attention head. *arXiv preprint arXiv:2310.04625*, 2023.

[186] Thomas McGrath, Matthew Rahtz, Janos Kramar, Vladimir Mikulik, and Shane Legg. The hydra effect: Emergent self-repair in language model computations. *arXiv preprint arXiv:2307.15771*, 2023.

[187] Rajiv Mehrotra, Kameswara Rao Namuduri, and Nagarajan Ranganathan. Gabor filter-based edge detection. *Pattern recognition*, 25(12):1479–1494, 1992.

[188] Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *Advances in Neural Information Processing Systems*, 35: 17359–17372, 2022.

[189] Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. Mass-editing memory in a transformer. *arXiv preprint arXiv:2210.07229*, 2022.

[190] Jack Merullo, Carsten Eickhoff, and Ellie Pavlick. Circuit component reuse across tasks in transformer language models. *arXiv preprint arXiv:2310.08744*, 2023.

[191] Daniel A Messenger and David M Bortz. Weak sindy: Galerkin-based data-driven model selection. *Multiscale Modeling & Simulation*, 19(3):1474–1497, 2021.

[192] Daniel A. Messenger and David M. Bortz. Weak SINDy For Partial Differential Equations. *Journal of Computational Physics*, 443:110525, October 2021. ISSN 00219991. doi:10.1016/j.jcp.2021.110525. URL http://arxiv.org/abs/2007.02848.

[193] Eric J Michaud, Ziming Liu, Uzay Girit, and Max Tegmark. The quantization model of neural scaling. *arXiv preprint arXiv:2303.13506*, 2023.

[194] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26, 2013.

[195] Tomáš Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies*, pages 746–751, 2013.

[196] Ari S. Morcos, Maithra Raghu, and Samy Bengio. Insights on representational similarity in neural networks with canonical correlation, 2018. URL http://arxiv.org/abs/1806.05759.

[197] Jesse Mu and Jacob Andreas. Compositional explanations of neurons. *Advances in Neural Information Processing Systems*, 33:17153–17163, 2020.

[198] Neel Nanda. Neuroscope: A website for mechanistic interpretability of language models, 2022. URL https://neuroscope.io/.

[199] Neel Nanda. Transformerlens, 2022. URL https://github.com/neelnanda-io/TransformerLens.

[200] Neel Nanda. A comprehensive mechanistic interpretability explainer & glossary, 2023. URL https://neelnanda.io/glossary.

[201] Neel Nanda. Actually, othello-gpt has a linear emergent world model, Mar 2023. URL <https://neelnanda.io/mechanistic-interpretability/othello>.

[202] Neel Nanda, Lawrence Chan, Tom Liberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability. *arXiv preprint arXiv:2301.05217*, 2023.

[203] Neel Nanda, Andrew Lee, and Martin Wattenberg. Emergent linear representations in world models of self-supervised sequence models. *arXiv preprint arXiv:2309.00941*, 2023.

[204] Richard Ngo, Lawrence Chan, and Sören Mindermann. The alignment problem from a deep learning perspective, 2023.

[205] Anh Nguyen, Jason Yosinski, and Jeff Clune. Multifaceted feature visualization: Uncovering the different types of features learned by each neuron in deep neural networks, 2016.

[206] Nostalgebraist. Interpreting gpt: The logit lens. https://www.alignmentforum.org/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens, 2020.

[207] NYC OpenData. Points of interest, 2023. URL https://data.cityofnewyork.us/City-Government/Points-Of-Interest/rxuy-2muj. Accessed: 2023-07-01.

[208] John O'Keefe and Jonathan Dostrovsky. The hippocampus as a spatial map: preliminary evidence from unit activity in the freely-moving rat. *Brain research*, 1971.

[209] Chris Olah. Interpretability vs neuroscience. https://colah.github.io/notes/interp-v-neuro/, 2021.

[210] Chris Olah. Interpretability vs neuroscience, 2021. URL https://colah.github.io/notes/interp-v-neuro/.

[211] Chris Olah. Distributed representations: Composition & superposition, 2023. https://transformer-circuits.pub/2023/superposition-composition/index.html.

[212] Chris Olah and Adam Jermyn. Reflections on qualitative research. https://transformer-circuits.pub/2024/qualitative-essay/index.html, 2024.

[213] Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. An overview of early vision in inceptionv1. *Distill*, 2020. doi:10.23915/distill.00024.002. https://distill.pub/2020/circuits/early-vision.

[214] Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. *Distill*, 5(3):e00024–001, 2020.

[215] Bruno A Olshausen and David J Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, 1996.

[216] Bruno A Olshausen and David J Field. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision research*, 37(23):3311–3325, 1997.

[217] Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. In-context learning and induction heads. *Transformer Circuits Thread*, 2022. https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html.

[218] OpenAI. Gpt-4 technical report, 2023.

[219] Wei Pan, Ye Yuan, Jorge Gonçalves, and Guy-Bart Stan. A sparse bayesian approach to the identification of nonlinear state-space systems. *IEEE Transactions on Automatic Control*, 61(1):182–187, 2015.

[220] Stefano Panzeri, Jakob H Macke, Joachim Gross, and Christoph Kayser. Neural population coding: combining insights from microscopic and mass signals. *Trends in cognitive sciences*, 19(3):162–172, 2015.

[221] Roma Patel and Ellie Pavlick. Mapping language models to grounded conceptual spaces. In *International Conference on Learning Representations*, 2021.

[222] Yagyensh Chandra Pati, Ramin Rezaiifar, and Perinkulam Sambamurthy Krishnaprasad. Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In *Proceedings of 27th Asilomar Conference on Signals, Systems and Computers*, pages 40–44. IEEE, 1993.

[223] Tiago Pimentel, Josef Valvoda, Rowan Hall Maudslay, Ran Zmigrod, Adina Williams, and Ryan Cotterell. Information-theoretic probing for linguistic structure. *arXiv preprint arXiv:2004.03061*, 2020.

[224] Alexandre Pouget, Peter Dayan, and Richard Zemel. Information processing with population codes. *Nature Reviews Neuroscience*, 1(2):125–132, 2000.

[225] Lucia Quirke, Lovis Heindrich, Wes Gurnee, and Neel Nanda. Training dynamics of contextual n-grams in language models. *arXiv preprint arXiv:2311.00863*, 2023.

[226] R Quian Quiroga, Leila Reddy, Gabriel Kreiman, Christof Koch, and Itzhak Fried. Invariant visual representation by single neurons in the human brain. *Nature*, 435 (7045):1102–1107, 2005.

[227] Alec Radford, Rafal Jozefowicz, and Ilya Sutskever. Learning to generate reviews and discovering sentiment. *arXiv preprint arXiv:1704.01444*, 2017.

[228] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[229] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[230] Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. SVCCA: Singular Vector Canonical Correlation Analysis for Deep Learning Dynamics and Interpretability, 2017. URL http://arxiv.org/abs/1706.05806.

[231] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10561*, 2017.

[232] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Multistep neural networks for data-driven discovery of nonlinear dynamical systems. *arXiv preprint arXiv:1801.01236*, 2018.

[233] Tilman Räuker, Anson Ho, Stephen Casper, and Dylan Hadfield-Menell. Toward transparent ai: A survey on interpreting the inner structures of deep neural networks. In *2023 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, pages 464–483. IEEE, 2023.

[234] Abhilasha Ravichander, Yonatan Belinkov, and Eduard Hovy. Probing the probing paradigm: Does probing accuracy entail task relevance? *arXiv preprint arXiv:2005.00719*, 2020.

[235] Patrick A. K. Reinbold, Daniel R. Gurevich, and Roman O. Grigoriev. Using noisy or incomplete data to discover models of spatiotemporal dynamics. *Physical Review E*, 101(1):010203, January 2020. ISSN 2470-0045, 2470-0053. doi:10.1103/PhysRevE.101.010203. URL https://link.aps.org/doi/10.1103/PhysRevE.101.010203.

[236] Albert Reuther, Jeremy Kepner, Chansup Byun, Siddharth Samsi, William Arcand, David Bestor, Bill Bergeron, Vijay Gadepally, Michael Houle, Matthew Hubbell, Michael Jones, Anna Klein, Lauren Milechin, Julia Mullen, Andrew Prout, Antonio Rosa, Charles Yee, and Peter Michaleas. Interactive supercomputing on 40,000 cores for machine learning and data analysis. In *2018 IEEE High Performance extreme Computing Conference (HPEC)*, pages 1–6. IEEE, 2018.

[237] Fred Rieke, David Warland, Rob de Ruyter Van Steveninck, and William Bialek. *Spikes: exploring the neural code*. MIT press, 1999.

[238] Anna Rogers, Olga Kovaleva, and Anna Rumshisky. A primer in bertology: What we know about how bert works, 2020.

[239] Anna Rogers, Olga Kovaleva, and Anna Rumshisky. A primer in bertology: What we know about how bert works. *Transactions of the Association for Computational Linguistics*, 8:842–866, 2021.

[240] Brian C Ross. Mutual information between discrete and continuous data sets. *PloS one*, 9(2):e87357, 2014.

[241] Otto E Rössler. An equation for continuous chaos. *Physics Letters A*, 57(5):397–398, 1976.

[242] Samuel H Rudy and Themistoklis P Sapsis. Sparse methods for automatic relevance determination. *Physica D: Nonlinear Phenomena*, 418:132843, 2021.

[243] Samuel H Rudy, Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Data-driven discovery of partial differential equations. *Science advances*, 3(4):e1602614, 2017.

[244] Hassan Sajjad, Nadir Durrani, and Fahim Dalvi. Neuron-level interpretation of deep nlp models: A survey. *Transactions of the Association for Computational Linguistics*, 10:1285–1303, 2022.

[245] Hassan Sajjad, Nadir Durrani, Fahim Dalvi, Firoj Alam, Abdul Rafae Khan, and Jia Xu. Analyzing encoded concepts in transformer language models. *arXiv preprint arXiv:2206.13289*, 2022.

[246] Hayden Schaeffer. Learning partial differential equations via data discovery and sparse optimization. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 473(2197):20160446, 2017.

[247] Hayden Schaeffer and Scott G. McCalla. Sparse model selection via integral terms. *Physical Review E*, 96(2):023302, August 2017. ISSN 2470-0045, 2470-0053. doi:10.1103/PhysRevE.96.023302. URL http://link.aps.org/doi/10.1103/PhysRevE.96.023302.

[248] Adam Scherlis, Kshitij Sachan, Adam S Jermyn, Joe Benton, and Buck Shlegeris. Polysemanticity and capacity in neural networks. *arXiv preprint arXiv:2210.01892*, 2022.

[249] Michael Schmidt and Hod Lipson. Distilling free-form natural laws from experimental data. *Science*, 324(5923):81–85, 2009.

[250] Daniel R Schonhaut, Zahra M Aghajan, Michael J Kahana, and Itzhak Fried. A neural code for time and space in the human brain. *Cell Reports*, 42(11), 2023.

[251] Ludwig Schubert, Chelsea Voss, Nick Cammarata, Gabriel Goh, and Chris Olah. High-low frequency detectors. *Distill*, 6(1):e00024–005, 2021.

[252] Ludwig Schubert, Chelsea Voss, Nick Cammarata, Gabriel Goh, and Chris Olah. High-low frequency detectors. *Distill*, 2021. doi:10.23915/distill.00024.005. https://distill.pub/2020/circuits/frequency-edges.

[253] Mahdiyar Shahbazi, Ali Shirali, Hamid Aghajan, and Hamed Nili. Using distance on the Riemannian manifold to compare representations in brain and in models. *NeuroImage*, 239:118271, 2021. ISSN 1053-8119. doi:10.1016/j.neuroimage.2021.118271. URL https://www.sciencedirect.com/science/article/pii/S1053811921005474.

[254] Lee Sharkey, Dan Braun, and Beren Millidge. Taking features out of superposition with sparse autoencoders, 2022. https://www.alignmentforum.org/posts/z6QQJbtpkEAX3Aojj.

[255] Xiaotong Shen, Wei Pan, Yunzhang Zhu, and Hui Zhou. On constrained and regularized high-dimensional regression. *Annals of the Institute of Statistical Mathematics*, 65(5):807–832, 2013.

[256] Gregory I Sivashinsky. Nonlinear analysis of hydrodynamic instability in laminar flames—i. derivation of basic equations. *Acta Astronautica*, 4(11):1177–1206, 1977.

[257] Alessandro Stolfo, Ben Wu, Wes Gurnee, Yonatan Belinkov, Xingyi Song, Mrinmaya Sachan, and Neel Nanda. Confidence regulation neurons in language models. *arXiv preprint arXiv:2406.16254*, 2024.

[258] Xavier Suau, Luca Zappella, and Nicholas Apostoloff. Finding experts in transformer models. *arXiv preprint arXiv:2005.07647*, 2020.

[259] Shuai Tang, Wesley J. Maddox, Charlie Dickens, Tom Diethe, and Andreas Damianou. Similarity of Neural Networks with Gradients, 2020. URL http://arxiv.org/abs/2003.11498.

[260] Ian Tenney, Dipanjan Das, and Ellie Pavlick. Bert rediscovers the classical nlp pipeline. *arXiv preprint arXiv:1905.05950*, 2019.

[261] Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R Bowman, Dipanjan Das, et al. What do you learn from context? probing for sentence structure in contextualized word representations. *arXiv preprint arXiv:1905.06316*, 2019.

[262] Ryan Thompson. Robust subset selection. *Computational Statistics & Data Analysis*, page 107415, 2022.

[263] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[264] Andreas M Tillmann, Daniel Bienstock, Andrea Lodi, and Alexandra Schwartz. Cardinality minimization, constraints, and regularization: a survey. *arXiv preprint arXiv:2106.09606*, 2021.

[265] Eric Todd, Millicent L Li, Arnab Sen Sharma, Aaron Mueller, Byron C Wallace, and David Bau. Function vectors in large language models. *arXiv preprint arXiv:2310.15213*, 2023.

[266] Shubham Toshniwal, Sam Wiseman, Karen Livescu, and Kevin Gimpel. Chess as a testbed for language model state tracking. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 11385–11393, 2022.

[267] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[268] Floris Van Breugel, J Nathan Kutz, and Bingni W Brunton. Numerical differentiation of noisy data: A unifying multi-objective optimization framework. *IEEE Access*, 8: 196865–196877, 2020.

[269] Balth Van der Pol. Lxxxviii. on "relaxation-oscillations". *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):978–992, 1926.

[270] Alexandre Variengien and Eric Winsor. Look before you leap: A universal emergent decomposition of retrieval tasks in language models, 2023.

[271] William E Vinje and Jack L Gallant. Sparse coding and decorrelation in primary visual cortex during natural vision. *Science*, 287(5456):1273–1276, 2000.

[272] Elena Voita and Ivan Titov. Information-theoretic probing with minimum description length. *arXiv preprint arXiv:2003.12298*, 2020.

[273] Elena Voita, Javier Ferrando, and Christoforos Nalmpantis. Neurons in large language models: Dead, n-gram, positional. *arXiv preprint arXiv:2309.04827*, 2023.

[274] Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: a circuit for indirect object identification in gpt-2 small. *arXiv preprint arXiv:2211.00593*, 2022.

[275] Liwei Wang, Lunjia Hu, Jiayuan Gu, Yue Wu, Zhiqiang Hu, Kun He, and John Hopcroft. Towards Understanding Learning Representations: To What Extent Do Different Neural Networks Learn the Same Representation, 2018. URL http://arxiv.org/abs/1810.11750.

[276] Tongzhou Wang and Phillip Isola. Understanding Contrastive Representation Learning through Alignment and Uniformity on the Hypersphere, 2022. URL http://arxiv.org/abs/2005.10242.

[277] Xiaozhi Wang, Kaiyue Wen, Zhengyan Zhang, Lei Hou, Zhiyuan Liu, and Juanzi Li. Finding skill neurons in pre-trained transformer-based language models. *arXiv preprint arXiv:2211.07349*, 2022.

[278] Christoph Wehmeyer and Frank Noé. Time-lagged autoencoders: Deep learning of slow collective variables for molecular kinetics. *The Journal of Chemical Physics*, 148 (24):241703, 2018.

[279] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022.

[280] Laura Weidinger, Jonathan Uesato, Maribeth Rauh, Conor Griffin, Po-Sen Huang, John Mellor, Amelia Glaese, Myra Cheng, Borja Balle, Atoosa Kasirzadeh, et al. Taxonomy of risks posed by language models. In *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*, pages 214–229, 2022.

[281] Alex H. Williams, Erin Kunz, Simon Kornblith, and Scott W. Linderman. Generalized Shape Metrics on Neural Representations, 2022. URL http://arxiv.org/abs/2110.14739.

[282] Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. Sheared llama: Accelerating language model pre-training via structured pruning. *arXiv preprint arXiv:2310.06694*, 2023.

[283] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*, 2023.

[284] Ji Xin, Jimmy Lin, and Yaoliang Yu. What part of the neural network does this? understanding lstms by measuring and dissecting neurons. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5823–5830, 2019.

[285] Enoch Yeung, Soumya Kundu, and Nathan Hodas. Learning deep neural network representations for koopman operators of nonlinear dynamical systems. In *2019 American Control Conference (ACC)*, pages 4832–4839. IEEE, 2019.

[286] Sheng Zhang and Guang Lin. Robust data-driven discovery of governing physical laws with error bars. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 474(2217):20180305, 2018.

[287] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

[288] Peng Zheng, Travis Askham, Steven L Brunton, J Nathan Kutz, and Aleksandr Y Aravkin. A unified framework for sparse relaxed regularized regression: Sr3. *IEEE Access*, 7:1404–1423, 2018.

[289] Ziqian Zhong, Ziming Liu, Max Tegmark, and Jacob Andreas. The clock and the pizza: Two stories in mechanistic explanation of neural networks. *arXiv preprint arXiv:2306.17844*, 2023.

[290] Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, et al. Representation engineering: A top-down approach to ai transparency. *arXiv preprint arXiv:2310.01405*, 2023.