Successor Heads: Recurring, Interpretable Attention Heads In The Wild

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

In this work we present successor heads: attention heads that increment tokens with a natural ordering, such as numbers, months, and days. For example, successor heads increment 'Monday' into 'Tuesday'. We explain the successor head behavior with an approach rooted in mechanistic interpretability, the field that aims to explain how models complete tasks in human-understandable terms. Existing research in this area has found interpretable language model components in small toy models. However, results in toy models have not yet led to insights that explain the internals of frontier models and little is currently understood about the internal operations of large language models. In this paper, we analyze the behavior of successor heads in large language models (LLMs) and find that they implement abstract representations that are common to different architectures. They form in LLMs with as few as 31 million parameters, and at least as many as 12 billion parameters, such as GPT-2, Pythia, and Llama-2. We find a set of 'mod 10' features[1] that underlie how successor heads increment in LLMs across different architectures and sizes. We perform vector arithmetic with these features to edit head behavior and provide insights into numeric representations within LLMs. Additionally, we study the behavior of successor heads on natural language data, identifying interpretable polysemanticity in a Pythia successor head.

1 Introduction

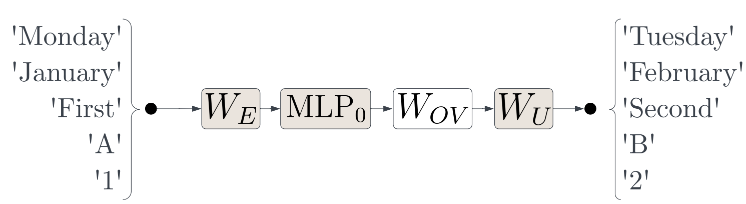


Figure 1: A ***successor head*** with OV matrix W_{OV} maps numbered tokens in embedding space (e.g. 'Monday') to their successor values in unembedding space (e.g. 'Tuesday'). The circuit consists of the embedding matrix, the first MLP block, one attention head, and the unembedding matrix.

Mechanistic interpretability [2] is the process of reverse-engineering the algorithms that trained neural networks have learned. Recently, much attention has been paid to interpreting transformer-based large language models (LLMs), as while these models have demonstrated impressive capabilities [3], there is little understanding of how these models produce their outputs. Existing interpretability

[1]Here, following Elhage et al. [1], a 'feature' refers to an interpretable (linear) direction in activation space.

Submitted to 37th Conference on Neural Information Processing Systems (NeurIPS 2023). Do not distribute.

research includes comprehensive reverse-engineering efforts into toy models [4] and small language models [5, 6], though few insights have been gained about how frontier LLMs function.

In mechanistic interpretability, universality [7, 8] is a hypothesis that there are *common representations* in neural networks – specifically, that neural networks with different architectures and scales form common internal representations. In this work, we use ***abstraction*** to refer to how common representations are used for different *tasks* models perform. Strong evidence for (or against) the universality hypothesis and task abstraction could significantly affect research priorities in interpretability. If common representations form across different language models and tasks, then research on small or toy language models [1, 9] and narrow tasks [6, 10, 11] may be the best way to gain insights into LLM capabilities. Conversely, if the representations used by language models do not generalize to different model scales and/or tasks, then developing methods that can be applied to larger language models and don't rely on lessons from small models generalizing (such as Wu et al. [12], Bills et al. [13], Conmy et al. [14]) may be the most important direction for interpretability.

In this work, we find an interpretable set of attention heads we call ***successor heads*** in models of many different scales and architectures. Successor heads are attention heads that perform incrementation in language models. The input to a successor head is the representation of a token in an ordinal sequence such as 'Monday', 'first', 'January', or 'one'. The output of a successor head assigns a higher likelihood to the incremented token, such as 'Tuesday', 'second, 'February', or 'two'. We find successor heads in the smaller and larger Pythia language models [15] with between 30 million and 12 billion parameters. We can understand the role of successor heads through a simple, end-to-end path through language models (Figure 1) and we identify transferrable features that these attention heads use across different tasks (Section 3). In our work, we find evidence for a weak form of universality ([16]; points 1. and 2.) as well as abstraction (point 3.) in language models, as

1. Successor heads use a *common representation*, such as a linear mod-10 features.
2. Successor heads form in LLMs with as few as 31M and as many as 12B parameters.
3. Successor heads perform incrementation across several different tasks.

Related work is discussed in Appendix P. Our contributions can be summarised as follows:

1. Introducing and interpreting successor heads (Section 2-3)

(a) We introduce and explain successor heads, which to the best of our knowledge are the most closely studied components in LLMs that occur in both small and large models.
(b) Using findings from **3.**, we edit successor heads' numeric inputs with vector arithmetic.

2. Showing that the succession mechanism is important in the wild (Appendix J)

(c) We show that successor heads play an important role in incrementation-based tasks in natural language datasets – for instance, predicting the next number in a numbered list of items.

3. Finding abstract numeric representations in language models (Section 3)

(d) We isolate a common *numeric subspace* within embedding space, that for any given token (e.g. 'February') encodes the index of that token within its ordinal sequence (e.g. months).
(e) Moreover, we find that this numeric subspace has *interpretable features*, as an unsupervised decomposition of token representations yields a crucial set of features we call the ***mod-10 features*** $\{f_0, ..., f_9\}$. f_n is present in all tokens whose numerical index $\equiv n \pmod{10}$, e.g. f_2 is present in the model's representations of '2', '32', '172', 'February', 'second' and 'twelve'.

2 Successor Heads

LLMs are able to increment elements in an ordinal sequence. For instance, Pythia-1.4B will complete the prompt "If this is 1, the next is" with " 2", and the prompt "If this is January, the next is" with " February". Given this observation, we find evidence for attention heads within LLMs (which we refer to as ***successor heads***) responsible for performing this type of incrementation. To get evidence for successor heads we require three definitions: i) the **succession dataset** of tasks involving abstract numeric representations, ii) an **effective OV circuit** to measure how attention heads affect model outputs, and finally iii) **successor score**.

The ***succession dataset*** consists of tokens from eight different ordinal sequences such as numbers, days and months (see Appendix Q for more details). We also include different forms of the tokens, as language model tokenizers often have several tokens corresponding to the same word, such as words with/without a space at the start being different tokens.

We determine whether attention heads perform succession by studying their ***effective OV circuit***, which measures how the direct effect of input tokens when multiplied by an OV matrix W_{OV}. The (non-effective) OV circuit $W_U W_{OV} W_E$ (1) from Elhage et al. [9] is the inspiration for our ***effective OV circuit*** $W_U W_{OV} \text{MLP}_0(W_E)$ (2). Intuitively, (2)'s columns represent input tokens to the head and the rows represent the logits on each possible output token. We then operationalize ***successor heads*** by considering an input token T from our succession dataset (e.g. 'Monday'). If the effective OV circuit column for input T has a larger output on the successor to T ('Tuesday') than on any other of the tokens in that task ('Monday' or 'Wednesday' or 'Thursday' ...) then we consider the head to have performed succession in this case. ***Successor Heads*** are then defined as the attention heads that pass this test for more than half of the tokens in the succession dataset. We call the proportion of succession dataset tokens on which an attention head performs succession the ***succession score***. We plot successor scores across models of varying size in Figure 2.

To better understand the role successor heads fulfill in real datasets, we perform a case study of the successor head L12H0 in Pythia-1.4B (see Appendix J): we find the head displays interpretable polysemanticity, performing incrementation but also copying, acronym, and greater-than behaviour.

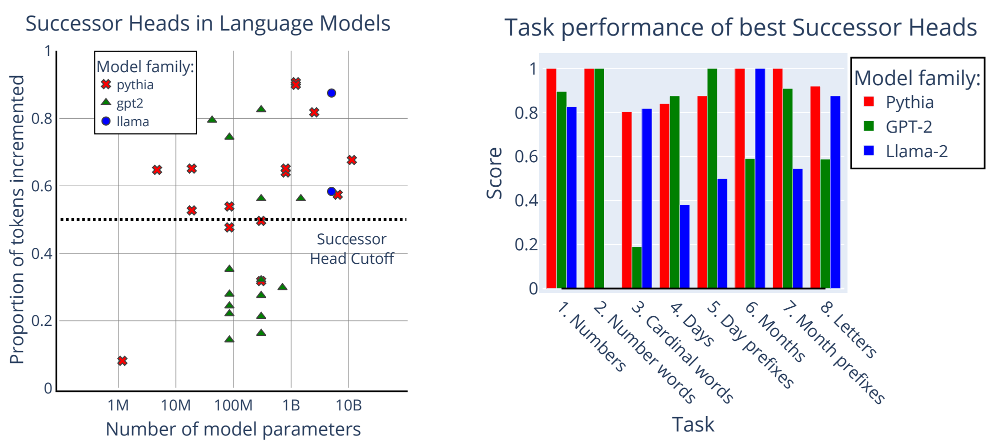


Figure 2: Plots of successor scores (proportion of tokens where succession occurs) for each model tested. A plot of the highest successor score observed across all attention heads for each model tested (left) and successor scores of the best successor heads in models (Pythia-1.4B, GPT-2 XL, Llama-2 7B) across different tasks (right).

3 Decomposing Numeric Representations

In the rest of this work, we perform a case study on the attention head (L12H0) with the maximal successor score in Pythia-1.4B. (Note that we also observe similar results across other models too – see Appendix C.3.) In Section 3.1, we find evidence for the existence of a *shared numeric subspace* within MLP0 representation-space. In Section 3.2, we find 'mod-10 features' in a decomposition of these representations, and we use these features to steer model behavior across different tasks.

3.1 Ordinal sequences are represented compositionally

We find evidence that ordinal sequences share a numeric subspace encoding the index of a token within its ordinal sequence. These findings are described in Appendix A.

3.2 Exploring mod-10 features

Finding mod-10 features. Now, given this evidence for a shared numeric subspace, a natural question to ask is whether these numeric embeddings themselves have any structure. To answer this question, we train a Sparse Autoencoder (SAE) (Ng [17], Elhage et al. [1], and Cunningham et al.

[18]) to recover the significant features across all tasks using reconstruction loss on the MLP0 outputs (see Appendix C for more details).Given a number token T and trained SAE, we define T's ***most important feature*** as the SAE feature that, when ablated from the MLP0 output reconstruction, causes the biggest decrease in the probability of the successor of T (by calculating probabilities from the logits obtained by multiplying by $W_U W_{OV}$).

We find that the most important feature for a number is usually a common feature across other numbers in the same mod-10 class (on average, the most common feature in a mod class is shared by 5.85 numbers out of the 10). For example, most numbers amongst 3, 13, ..., 93 share the same important feature. This property gives rise to a mod-10 pattern in feature activations across the numbers, when we visualize the components of each most important feature across all the input prompts (Figure 3). We also verify that these mod-10 features have causal importance for the successor head's incrementation: multiplying these features by $W_U W_{OV}$ (in Figure 4) shows strong modulo 10 bands that are shifted by one to show that the successor head increments these features.

To define the ***mod-10 features*** we take the modal most important SAE feature for tokens within a particular mod class, and average this feature over 100 training runs, denoting the resulting features $f_0, \ldots, f_9$. We can then multiply these features by $W_U W_{OV}$ to study their effect on logits, as shown in Figure 5. We find unsurprisingly that f_i increases the logits on $f_{i+1 \pmod{10}}$. Further, the increase on logits for single-digit numbers is larger than the increase in logits for double-digit numbers.

Transferability of mod-10 features. Are our mod-10 features simply an artifact of the SAE technique? We provide evidence that the mod-10 features are natural, causally important features by using two independent methods to recover them; (1) linear probing and (2) identifying MLP0 neurons. We also show that the features transfer to other tasks in the succession dataset (Section 2). These findings are described in Appendix F.

Arithmetic with mod-10 features. The generalization of the linear probe to unseen numeric tasks provides us with evidence that there is a shared mod 10 structure across tasks. In this section, we stress test our understanding of this shared structure by trying to alter the index of ordinal sequence

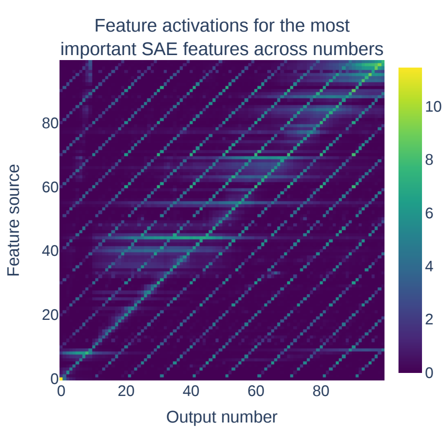


Figure 3: Feature activations, where for each number token (y-axis), we observe how its most important feature activates across all number tokens (x-axis). Values averaged over 100 SAE training runs.

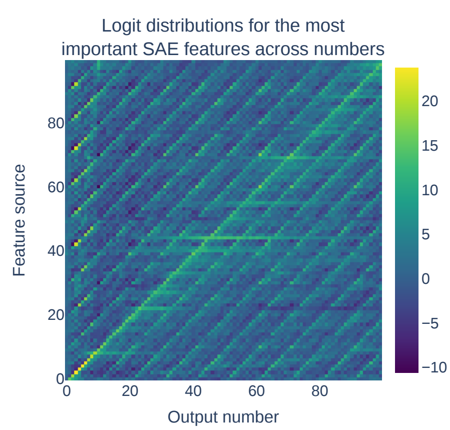


Figure 4: Logit distribution of important features, where for each number token (y-axis), we multiply its important feature by $W_U W_{OV}$ and observe how logits are attributed across all number tokens (x-axis).

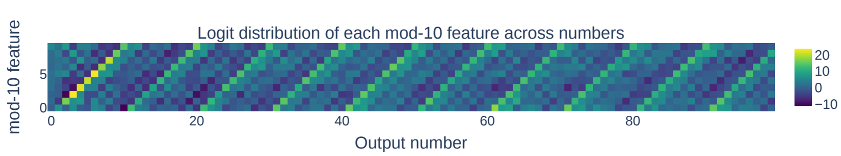


Figure 5: Logit distribution $W_U W_{OV} f_i$ for each mod-10 feature f_i.

tokens via vector arithmetic with the mod 10 features. For example, we expect $\text{MLP}_0(W_E(\text{‘fifth’}))-f_5+f_7$ (3) to be causally used by the model in a similar way to how the model would use a representation of the token ‘seventh’. We use the successor head to test this hypothesis. If (3) behaves like $\text{MLP}_0(W_E(\text{‘seventh’}))$, we would expect that multiplying (3) by $W_U W_{OV}$ attributes more logits on ‘eighth’ than any of the other tokens from task 2 in the succession dataset (Section 2). Indeed, this is correct as indicated by the circled checkmark ✓ in Figure 6. We can perform a similar experiment with all tokens in the succession dataset and with features other than f_7 added. The cases where the max logits are on the successor token are check-marked in Figure 6. We describe the experiment in more detail in Appendix G. We find that when the mod 10 addition feature is larger than the source value (modulo 10), vector arithmetic works on 53% (for months) and 89% (for digits 20-29) of cases. Below the diagonal, we see mostly failures, and we find that this is due to successor heads possessing a ***greater-than bias***, described in Appendix H.

	0	1	2	3	4	5	6	7	8	9
20	✓	✓		✓	✓	✓	✓	✓	✓	✓
21		✓	✓	✓	✓	✓			✓	✓
22			✓	✓	✓	✓	✓	✓	✓	✓
23				✓	✓	✓	✓		✓	✓
24					✓	✓	✓	✓	✓	✓
25						✓	✓	✓	✓	✓
26							✓	✓		✓
27								✓		
28									✓	✓
29										✓
ten	✓	✓	✓	✓	✓	✓	✓	✓	✓	
eleven		✓	✓	✓	✓	✓	✓	✓	✓	✓
twelve			✓	✓	✓	✓	✓		✓	
thirteen				✓	✓	✓	✓	✓	✓	✓
fourteen					✓	✓	✓	✓	✓	
fifteen					✓	✓	✓		✓	✓
sixteen						✓	✓	✓	✓	✓
seventeen							✓	✓	✓	✓
eighteen								✓	✓	✓
nineteen								✓	✓	✓
twenty								✓		✓

	1	2	3	4	5	6	7	8	9	0
January	✓	✓	✓							
February		✓	✓	✓						
March			✓	✓	✓					
April			✓	✓	✓					
May				✓	✓	✓				
June			✓		✓	✓	✓	✓	✓	
July							✓	✓	✓	
August								✓	✓	
September			✓						✓	✓
first	✓	✓	✓	✓						
second		✓	✓	✓	✓					
third			✓	✓	✓	✓	✓			
fourth				✓	✓	✓	✓			
fifth					✓	✓	✓	✓		
sixth					✓	✓	✓	✓		
seventh						✓	✓	✓		
eighth							✓	✓	✓	
ninth									✓	

Figure 6: Table displaying in which cases where vector arithmetic such as (3) are successful. Rows: source tokens. Columns: target residues modulo 10.

4 Conclusion

In this work, we discovered and interpreted a class of attention heads we call *successor heads*. We showed these heads increment tokens like numbers, months, and days partly by mapping them to an abstract mod-10 numeric representation. We provided evidence that successor heads exhibit a weak form of universality, arising in models across different architectures and scales, and using similar underlying mod-10 features in all cases. Additionally, we validated our understanding by demonstrating that a successor head reduced the loss on training data by predicting successor tokens.

Additional findings that stemmed from our work include:

1. Finding a greater-than bias, where a language model was much more likely to predict numeric answers larger than the values in the prompt, compared to smaller values than tokens present in the prompt, that was observable by a weights-level analysis.
2. Surprisingly interpretable individual MLP0 neurons on this narrow task.
3. A novel example of attention head polysemanticity (successor heads predicting acronyms).

Findings 1-3 could prompt further future work into how language models represent numeric concepts, particularly as 2 and 3 were surprising given existing evidence from existing work. Our work in finding a language model component that arises in models of many different scales (and uses abstract underlying numeric representations) may be a valuable contribution toward understanding the inner workings of frontier LLMs.

References

[1] 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". In: *arXiv preprint arXiv:2209.10652* (2022).

[2] Chris Olah. *Mechanistic Interpretability, Variables, and the Importance of Interpretable Bases*. https://www.transformer-circuits.pub/2022/mech-interp-essay. 2022.

[3] OpenAI. *GPT-4 Technical Report*. 2023. arXiv: 2303.08774 [cs.CL].

[4] Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. "Progress measures for grokking via mechanistic interpretability". In: *The Eleventh International Conference on Learning Representations*. 2023. URL: https://openreview.net/forum?id=9XFSbDPmdW.

[5] Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, et al. *In-context learning and induction heads*. 2022. URL: https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html.

[6] Kevin Ro Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. "Interpretability in the Wild: a Circuit for Indirect Object Identification in GPT-2 Small". In: *The Eleventh International Conference on Learning Representations*. 2023. URL: https://openreview.net/forum?id=NpsVSN6o4ul.

[7] Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. "Zoom In: An Introduction to Circuits". In: *Distill* (2020). DOI: 10.23915/distill.00024.001.

[8] Yixuan Li, Jason Yosinski, Jeff Clune, Hod Lipson, and John Hopcroft. *Convergent Learning: Do different neural networks learn the same representations?* 2016. arXiv: 1511.07543 [cs.LG].

[9] Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. "A Mathematical Framework for Transformer Circuits". In: *Transformer Circuits Thread* (2021). URL: https://transformer-circuits.pub/2021/framework/index.html.

[10] Stefan Heimersheim and Jett Janiak. *A circuit for Python docstrings in a 4-layer attention-only transformer*. 2023. URL: https://www.alignmentforum.org/posts/u6KXXmKFbXfWzoAXn/a-circuit-for-python-docstrings-in-a-4-layer-attention-only.

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

[12] Zhengxuan Wu, Atticus Geiger, Christopher Potts, and Noah D. Goodman. *Interpretability at Scale: Identifying Causal Mechanisms in Alpaca*. 2023. arXiv: 2305.08809 [cs.CL].

[13] 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.

[14] Arthur Conmy, Augustine N. Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. *Towards Automated Circuit Discovery for Mechanistic Interpretability*. 2023. arXiv: 2304.14997 [cs.LG].

[15] 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. arXiv: 2304.01373 [cs.CL].

[16] Bilal Chughtai, Lawrence Chan, and Neel Nanda. *A Toy Model of Universality: Reverse Engineering How Networks Learn Group Operations*. 2023. URL: https://arxiv.org/abs/2302.03025.

[17] Andrew Ng. *Sparse Autoencoder*. Online Course Notes, Stanford University. 2011. URL: https://web.stanford.edu/class/cs294a/sparseAutoencoder.pdf.

[18] Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. *Sparse Autoencoders Find Highly Interpretable Features in Language Models*. 2023. arXiv: 2309.08600 [cs.LG].

[19] Lee Sharkey, Dan Braun, and Beren Millidge. *[Interim research report] Taking features out of superposition with sparse autoencoders*. 2022. URL: https://www.lesswrong.com/posts/z6QQJbtpkEAX3Aojj/interim-research-report-taking-features-out-of-superposition.

[20] 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*. 2023. arXiv: 2303.08112 [cs.LG].

[21] Stanford CRFM. *Mistral: A framework for transparent and accessible large-scale language model training*. https://github.com/stanford-crfm/mistral. 2021.

[22] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. *Network Dissection: Quantifying Interpretability of Deep Visual Representations*. 2017. arXiv: 1704.05796 [cs.CV].

[23] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. "Featture Visualization". In: *Distill* (2017). https://distill.pub/2017/feature-visualization. DOI: 10.23915/distill.00007.

[24] Nicholas Goldowsky-Dill, Chris MacLeod, Lucas Sato, and Aryaman Arora. *Localizing Model Behavior with Path Patching*. 2023. arXiv: 2304.05969 [cs.LG].

[25] Kevin Du, Lucas Torroba Hennigen, Niklas Stoehr, Alexander Warstadt, and Ryan Cotterell. *Generalizing Backpropagation for Gradient-Based Interpretability*. 2023. arXiv: 2307.03056 [cs.LG].

[26] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. *Distributed Representations of Words and Phrases and their Compositionality*. 2013. arXiv: 1310.4546 [cs.CL].

[27] Jack Merullo, Carsten Eickhoff, and Ellie Pavlick. *Language Models Implement Simple Word2Vec-style Vector Arithmetic*. 2023. arXiv: 2305.16130 [cs.CL].

[28] Nishant Subramani, Nivedita Suresh, and Matthew E. Peters. *Extracting Latent Steering Vectors from Pretrained Language Models*. 2022. arXiv: 2205.05124 [cs.CL].

[29] Alexander Matt Turner, Lisa Thiergart, David Udell, Gavin Leech, Ulisse Mini, and Monte MacDiarmid. *Activation Addition: Steering Language Models Without Optimization*. 2023. arXiv: 2308.10248 [cs.CL].

[30] Neel Nanda. *A Comprehensive Mechanistic Interpretability Explainer & Glossary*. https://www.neelnanda.io/glossary. Dec. 2022.

A Factoring ordinal sequences

Let i_s denote the ith token in ordinal sequence s (such that e.g. 2_{Month} corresponds to the token 'February'), and let $[\![i_s]\!] = \text{MLP}_0(W_E(i_s))$ denote the model's *internal representation* of token i_s (the output of MLP_0 in Figure 1). Given successor heads $S = W_{OV}$ can increment tokens s_i from a range of ordinal sequences s (e.g. numbers, months, days of the week), one might hypothesise that numeric representations have *compositional structure* – i.e. that information about a token's *position* i in its ordinal sequence is encoded independently from information about *which ordinal sequence* s it comes from. More precisely, we claim that we can decompose representations $[\![i_s]\!]$ into features $\mathbf{v}_i$ living in some 'index space' and $\mathbf{v}_s$ living in some 'domain space', such that $[\![i_s]\!] = \mathbf{v}_i + \mathbf{v}_s$.

Method. To test this compositionality hypothesis, we wish to learn two linear maps – an *index-space projection* $\pi_{\mathbb{N}} : \mathbb{R}^{d_{\text{model}}} \to \mathbb{R}^{d_{\text{model}}}$ and a *domain-space projection* $\pi_D : \mathbb{R}^{d_{\text{model}}} \to \mathbb{R}^{d_{\text{model}}}$ – such that, for all pairs of tokens i_s and j_t (with i_t a valid token), $\widehat{[\![i_t]\!]} := \pi_{\mathbb{N}}([\![i_s]\!]) + \pi_D([\![j_t]\!]) \approx [\![i_t]\!]$. To do so, we enforce that $\pi_{\mathbb{N}} + \pi_D = I$, and ensure predicted representations $\widehat{[\![i_t]\!]}$ 'behave like' ground truth representations $[\![i_t]\!]$ for randomly sampled pairs of tokens i_s and j_t – in other words, that there is low L2-distance between $\widehat{[\![i_t]\!]}$ and $[\![i_t]\!]$, that $\widehat{[\![i_t]\!]}$ decodes to i_t (***output-space decoding***), and that $S([\![i_t]\!])$ decodes to $(i+1)_t$ (***successor decoding***). For full experimental details see Appendix B.

Source token (sequence)	No succ	I	II	III	IV	V	VI	VII	VIII	IX	X	XI	XII
	1	9	2	3	4	22	6	7	8	9	24	11	12
	one	nine	two	three	four	twenty	six	seven	eight	nine	twenty	eleven	twelve
	first	ninth	second	third	fourth	second	sixth	seventh	eighth	ninth	fourth	-	-
	Monday	Sunday	Tuesday	Wednesd	Thursday	Tuesday	Saturday	Sunday	-	-	-	-	-
	Mon	Sep	Tue	Wed	Thu	Tue	Sat	Sun	-	-	-	-	-
	January	Septemb	February	March	April	February	June	July	August	Septemb	Decembe	Novembe	Decembe
	Jan	Sep	Feb	Mar	Apr	Feb	Jun	Jul	Aug	Sep	Dec	Nov	Dec
	A	I	B	C	D	V	F	G	H	I	X	W	L

Source token (sequence)	With succ	I	II	III	IV	V	VI	VII	VIII	IX	X	XI	XII
	1	10	Third	Fourth	5	23	VII	8	9	10	25	12	13
	one	10	Third	fourth	fifth	23	seventh	eighth	ninth	10	25	12	13
	first	tenth	Third	fourth	fifth	VI	seventh	ninth	ninth	tenth	-	-	-
	Monday	RS	Third	MC	Friday	MV	VII	-	-	-	-	-	-
	Mon	RS	HM	HM	HM	MV	MTP	-	-	-	-	-	-
	January	RS	cs	cs	May	23	cs	cs	Septembe	October	25	cs	
	Jan	CS	cs	cs	May	23	cs	cs	Sept	rs	35	cs	
	A	III	III	MC	Fifth	WV	VII	VIII	9	cx	Y	XII	iii

Table 1: A table presenting top-1 predicted tokens (under output-space projection and successor decoding) from representations $\pi_D(1_s) + \pi_{\mathbb{N}}(i_{Rom})$. Green cells denote predictions which match their target exactly; yellow cells denote predictions which match their target modulo formatting (e.g. 'six' rather than '6'); red cells denote incorrect predictions.

Results. On our held-out dataset of token pairs, we obtained a top-1 output-space decoding accuracy of 1.00, and a top-1 successor decoding accuracy of 0.90.[2] To explore out-of-distribution performance, we also test whether $\pi_{\mathbb{N}}$ can project out the numeric component of Roman numerals (which weren't in the successor dataset), by taking Roman numerals $i_{\text{Rom}} \in \{\text{'I'}, ..., \text{'XII'}\}$ and tokens 1_s from sequences s in the successor dataset, and testing whether $\pi_D(1_s) + \pi_{\mathbb{N}}(i_{\text{Rom}})$ decodes to i_s. We present the top-1 predicted tokens (under both output-space and successor decoding) in Table 1. Observe that, while output-space decoding yields perfect top-1 accuracy (apart from $i \in \{1, 5, 10\}$, which we can attribute to the Roman numerals I, V and X being single-letter and impossible to disambiguate from 9_{Letter}, 22_{Letter} and 24_{Letter}), successor decoding achieves an accuracy of 0.125 (or 0.29 if we allow for incorrectly formatted predictions).

These results – in particular, our ability to project the numeric component out of tokens from unseen sequences and transfer indices across domains – suggest that there is a shared numeric subspace storing the index of a token within its ordinal sequence. Indeed, informal testing suggests that this numeric subspace may be interpretable even for tokens not part of an ordinal sequence: for instance, $d(\pi_{\mathbb{N}}(\llbracket\text{' triangle'}\rrbracket) + \pi_D(1_{\text{Num}}))$ yields 3_{Num}, and $d(\pi_{\mathbb{N}}(\llbracket\text{' week'}\rrbracket) + \pi_D(1_{\text{Num}}))$ yields 7_{Num}.

Despite this, the drop in performance when applying the successor head to our constructed representations (and in particular, the leakage of Roman-numeral information into $\pi_D(1_s) + \pi_{\mathbb{N}}(i_{Rom})$ – notice the VII and VIII in Table 1) suggests our numeric projection $\pi_{\mathbb{N}}$ might be capturing slightly more than just the numeric subspace. Specifically, there may be some components of domain-space which are ignored by output-space decoding, but which our successor head lifts into output-space.

B Training details for compositionality experiments

Remark: obtaining a decoding function. Recall that we wish to learn two linear maps – an *index-space projection* $\pi_{\mathbb{N}} : \mathbb{R}^{d_{\text{model}}} \to \mathbb{R}^{d_{\text{model}}}$ and a *domain-space projection* $\pi_D : \mathbb{R}^{d_{\text{model}}} \to \mathbb{R}^{d_{\text{model}}}$ – such that, for all pairs of tokens i_s and j_t (with i_t a valid token), $\hat{\llbracket i_t \rrbracket} := \pi_{\mathbb{N}}(\llbracket i_s \rrbracket) + \pi_D(\llbracket j_t \rrbracket) \approx \llbracket i_t \rrbracket$. To evaluate the above identity, we want a decoding function $d : \mathbb{R}^{d_{\text{model}}} \to \texttt{Logits}$, such that $d(\llbracket i_s \rrbracket) = i_s$. Given the informal observation that directly unembedding $\llbracket i_s \rrbracket$ yields next-token predictions for i_s, whereas unembedding $S(i_s)$ yields $(i+1)_s$, we hypothesise that the unembedding matrix W_U reads from some 'output space' $\mathcal{O}$ and the embedding transform $\llbracket\cdot\rrbracket$ writes to some 'input space' $\mathcal{I}$ – and that the successor head reads from $\mathcal{I}$ and writes to $\mathcal{O}$. Indeed, when training an *output-space projection* $\pi_O : \mathcal{I} \to \mathcal{O}$ over tokens in the vocabulary such that $W_U(\pi_O(\llbracket i_s \rrbracket)) = i_s$,

[2] Note that, as our successor dataset contains 1041 tokens, a random classifier (even when restricted to tokens in the successor dataset) would achieve an accuracy of 0.001.

we obtain 97.4% top-1 accuracy on a set of 1000 held-out tokens – which both confirms the output-space hypothesis, and gives us a decoding function $d(\mathbf{x}) = W_U(\pi_O(\mathbf{x}))$.

Method. With our decoding function in hand, we can train $\pi_{\mathbb{N}}$ and π_D to satisfy our identity. Specifically, we define $\pi_{\mathbb{N}}$ and π_D to be matrices such that $\pi_{\mathbb{N}} + \pi_D = I$. For valid token pairs i_s and j_t, we obtain predicted representations $[\![\hat{i_t}]\!] = \pi_{\mathbb{N}}([\![i_s]\!]) + \pi_D([\![j_t]\!])$, and minimise a combination of 'closeness metrics':

$$||[\![\hat{i_t}]\!] - [\![i_t]\!]||^2 + \mathcal{L}(W_U(\pi_O([\![\hat{i_t}]\!])), i_t) + \mathcal{L}(W_U(S([\![\hat{i_t}]\!])), W_U(S([\![i_t]\!])))$$

for $\mathcal{L}$ the cross-entropy loss. Specifically, we ensure that predicted and ground truth representations 'behave in the same way' – in other words, that they are close together, that predicted representations $[\![\hat{i_t}]\!]$ decode to tokens i_t (***output-space decoding***), and that the logit distribution when decoding incremented predicted representations $S([\![i_t]\!])$ matches that when decoding incremented ground truth representations $S([\![i_t]\!])$ (***successor decoding***).

More succinctly, we can frame the training procedure as learning $\pi_{\mathbb{N}}, \pi_D$ such that the following diagram commutes:

$$\begin{tikzcd}
\mathcal{R}_{\mathbb{N}} \times \mathcal{R}_D \arrow[rrr, "(+)"] & & & \mathcal{I} \arrow[dl, "W_U \circ S"'] \arrow[d, "W_U \circ \pi_O"] \arrow[dr, "\mathtt{id}_{\mathcal{I}}"] & \\
\mathcal{I} \times \mathcal{I} \arrow[u, "\pi_{\mathbb{N}} \times \pi_D"] & & Logits[\mathbb{N} \times \mathcal{D}] & \mathbb{N} \times \mathcal{D} & \mathcal{I} \\
(\mathbb{N} \times D) \times (\mathbb{N} \times D) \arrow[u, "{[\![\cdot]\!] \times [\![\cdot]\!]}"] \arrow[rrr, "\langle \pi_1 \circ \pi_1, \pi_2 \circ \pi_2 \rangle"'] & & & \mathbb{N} \times \mathcal{D} \arrow[ul, "W_U \circ S \circ [\![\cdot]\!]"] \arrow[u, "\mathtt{id}_{\mathbb{N} \times \mathcal{D}}"] \arrow[ur, "{[\![\cdot]\!]}"'] &
\end{tikzcd}$$

We trained for 10 epochs over valid token pairs sampled from the succession dataset, and evaluated on a held-out dataset of 500 randomly-sampled token pairs.

C Sparse auto-encoders

C.1 Definition

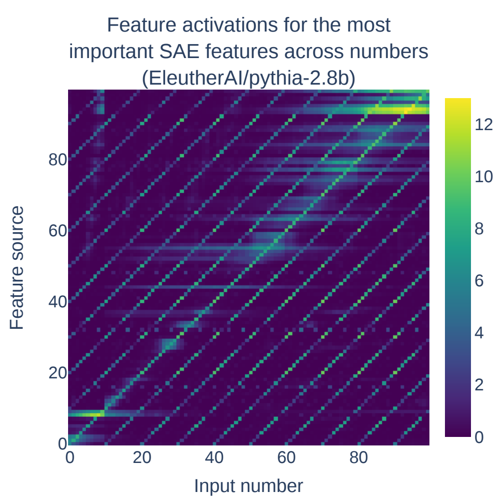
We refer to a single-layer autoencoder with a sparsity regularization term in its loss as a **sparse auto-encoder**.

For a dataset generated from a set of underlying vectors (each dataset example is a sparse linear combination of such vectors), it has been empirically observed [19, 18] that sparse auto-encoders are capable of retrieving the underlying set of vectors. We hope to obtain a set of sparse, interpretable features from the SAEs that decompose some of the structure of MLP0 space that we can use to analyze the way numeric operations are performed.

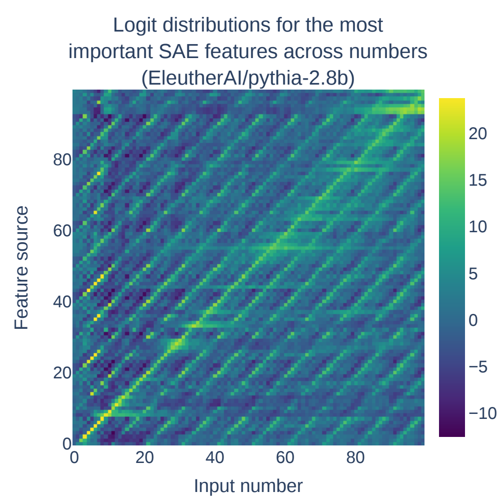
C.2 Training process for mod 10 features

Training a sparse auto-encoder with D features and regularization coefficient λ on a dataset of tokens in MLP0 space results in a map $F : \text{str} \to (\mathbb{R}^d \times \mathbb{R}_+)^D$, with $F(x) = \{(f_1, a_1), \ldots, (f_D, a_D)\}$, mapping a token to a set of feature and feature-activation pairs, with reconstruction $R(x) = \sum_{i=1}^{D} a_i f_i$.

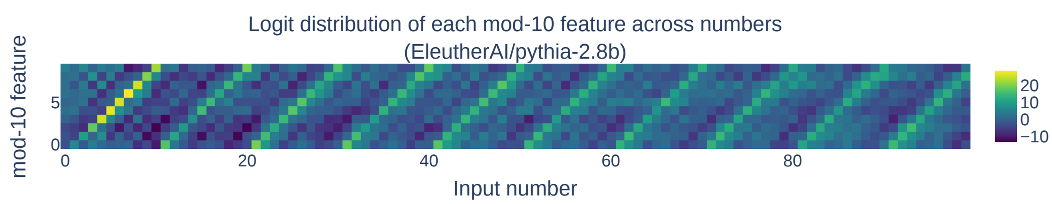
We train the SAE using number tokens from 0 to 500, both with and without a space ('123' and ' 123'), alongside other tasks, such as number words, cardinal words, days, months, etc. 90% of these tokens go into the train set, and the remaining 10% to the validation set. Even with the other tasks, the dataset is dominated by numbers, but creating a more balanced dataset would give us less data to work with, and without enough data, the SAE fails to generalize to the validation set. Hence, we only concern ourselves with the features that the SAE learns for number tokens, and we then



(a) SAE feature activations



(b) Feature logit distribution



(c) Logit distribution for each mod-10 feature

Figure 7: SAE plots for Pythia-2.8b analogous to Figure 3, Figure 4, and Figure 5

separately check whether these features generalize to the other tasks on the basis of logits, rather than SAE activations.

We used the hyperparameters $D = 512$ and $\lambda = 0.3$, with a batch size of 64, and trained for 100 epochs. To find these hyperparameters, we used the metric of mean max cosine similarity between two trained SAEs, as described in Sharkey, Braun, and Millidge [19] and Cunningham et al. [18].

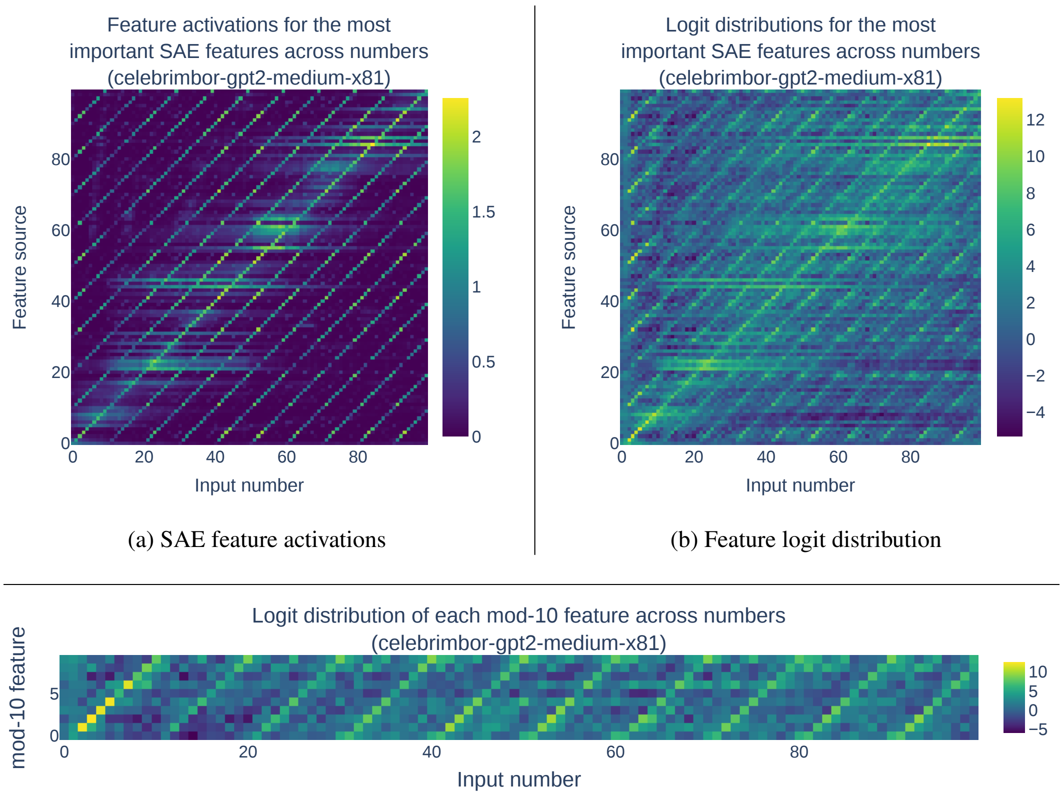
C.3 Universality of mod-10 results

We also observe the mod 10 structure via SAEs across models other than Pythia-1.4B, without any finetuning of SAE parameters to these models. We reproduce the SAE figures seen in Section 3.2 for other models, with Appendix C.3 for Pythia-2.8B, and Appendix C.3 for celebrimbor-gpt2-medium-x81.

D Linear probing

We train a linear probe to predict the mod 10 value of tokens. Specifically, we train on number tokens from ‘0’ to ‘500’, both with and without a space, assigning 90% of tokens to a train set, and the remaining 10% to a valid set. We use a learning rate of 0.001, and a batch size of 32, for 100 epochs.

We then evaluate on a dataset of unseen tasks, including number words (from ’one’ to ’nineteen’), placements, numerals, months, days, and any valid spaced and capitalized variants. Out of the total 102 such examples, 94/102 are correct, and the 8 failures are: [‘January’, ‘December’, ‘Friday’, ‘Saturday’, ‘Sunday’, ‘ V’, ‘ X’, ‘ XV’].



(a) SAE feature activations

(b) Feature logit distribution

(c) Logit distribution for each mod-10 feature

Figure 8: SAE plots for celebrimbor-gpt2-medium-x81 analogous to Figure 3, Figure 4, and Figure 5

The failures of 3 out of 7 days are consistent with our inability to interpret the day task well with our mod 10 features. Additionally, we see 'January' and 'December' as failure cases, which is also consistent with our finding that there does not seem to be a mod 10 feature that corresponds to any of them: f_1 behaves as 'November' rather than 'January', and f_2 as 'February'.

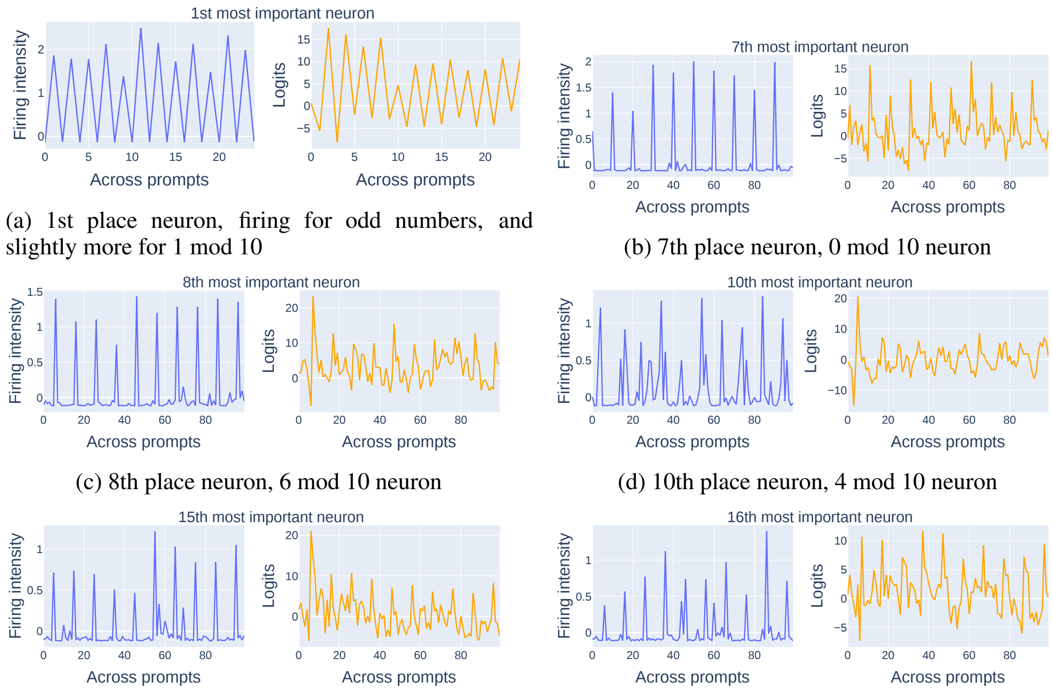
E MLP0 neurons

In our MLP0 neuron experiments, we do the following: for each $T \in \{`0', `1', \ldots, `99'\}$, we ablate each neuron from the final activation in MLP0 (the final activation is just before the final linear layer of MLP0), and store the probability attributed to the successor of T after passing the modified (due to ablation) MLP0 output through the successor.

Averaging the correct probabilities across all 100 prompts then gives an averaged correct probability for each neuron after ablation. We then look at the intensities and logits (across number tokens) for neurons with the lowest correct probability after ablation, meaning they have the most impact on successorship when ablated. This gives us the plots seen in Figure 9.

F Transferability of mod-10 features

(1) Linear probing: we train a linear probe to predict the mod-10 value across numbers from their MLP0 representations. We find that the ith row of the linear probe matrix has a high cosine similarity (on average 0.70764) to the corresponding mod-10 feature f_i obtained with the SAE. This suggests that the features f_i are likely to be directly finding the mod-10 value of input tokens rather than picking up on a property that is correlated with mod classes. Further, the probe generalizes,



(a) 1st place neuron, firing for odd numbers, and slightly more for 1 mod 10

(b) 7th place neuron, 0 mod 10 neuron

(c) 8th place neuron, 6 mod 10 neuron

(d) 10th place neuron, 4 mod 10 neuron

(e) 15th place neuron, 5 mod 10 neuron

(f) 16th place neuron, another 6 mod 10 neuron

Figure 9: Some examples of neurons firing strongly in modulo 10 patterns out of the top 16 most important MLP0 neurons for successorship.

correctly predicting the mod-10 value for 94/102 examples from the succession dataset tasks 2-8 (Section 2). Appendix D describes our full experimental setup.

(2) MLP0 neurons: we perform ablative experiments on individual MLP0 neurons to find the most important neurons for successorship across numbers, as measured by probability decrease as in the SAE experiments. This reveals neurons that strongly fire in a mod-10 pattern, which is the most common frequency amongst the top 16 most important neurons. Some examples of such neurons in the top 16 most important neurons can be seen in Figure 9. We also verify that the neurons indeed increase probability on successor tokens by multiplying their output values by $W_U W_{OV}$ in the same figures. Further technical details can be found in Appendix E.

Our results on the interpretability of individual neurons are surprising given recent research suggesting that the individual neurons of language models may be inappropriate as the units of understanding [1]. However, our results do not contradict previous finding that understanding MLPs requires understanding distributed behaviors, since for example the 6 mod 10 feature appears to be in superposition across at least two neurons (Figure 9c, 9f).

G Arithmetic experiments

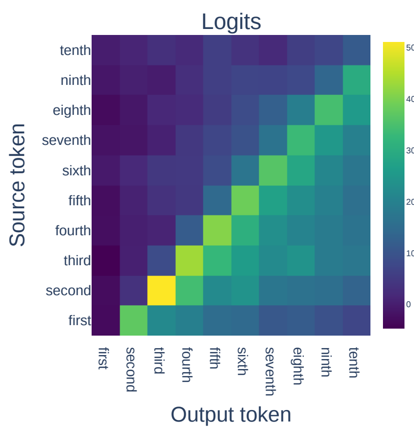
For a token $T \in V$ (row of arithmetic table) in numeric class V and feature f_i (column of arithmetic table), we consider how $x := \text{MLP}_0(W_E(T)) + k(-f_{\text{ord}(T)} + f_i)$ attributes logits to tokens in V, with $k \in \mathbb{R}_+$ a scaling, and $\text{ord}(T)$ the numeric order of T with respect to V. We denote whether x correctly attributes maximal logits to the token $U \in V$ with $\text{ord}(U) = i + 1$ by a checkmark, giving Figure 6. Since our mod-10 features $\{f_i\}_i$ obtained from the SAE are normalized to unit norm, hence some scaling is necessary in order to have an effect on the order of the tokens. We pick $k = \text{MLP}_0(W_E(T)) \cdot f_{\text{ord}(T)}$ everywhere other than months, where we observe that we must use a scaling of 2 times this to affect the order while keeping the task identity intact.

H Greater-than bias

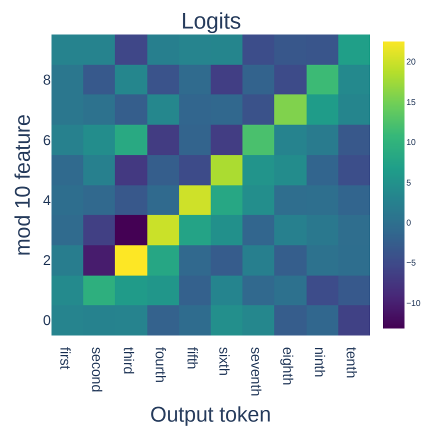
The vector arithmetic experiments (Figure 6) mostly fail below the diagonal, when the mod-10 addition is smaller than the source tokens's ordinal sequence position mod 10. This is because successor heads are biased towards values greater than the successor, compared to values less than the successor.

This effect can be seen in Figure 10a on the tokens ‘first’, ‘second’, ..., ‘tenth’. However, our mod-10 features do not exhibit a greater-than bias, as seen in Figure 10b. We survey these effects across all tasks in Appendix M. As a result, using the mod-10 features to shift logits towards tokens of a lower order than the input token fails, as the changes in logits are not significant compared to the large logits on higher-order tokens. In the case of numbers, this leads to the effect that, for example, $W_U W_{OV}\,(\mathrm{MLP}_0(W_E(\text{‘35’})) - f_5 + f_3)$ has high logits on ‘44’, rather than ‘34’ (this ’+10’ effect occurs for 2/3 of entries below the diagonal in the 20-29 numbers table of Figure 6).

Limitations: The absence of a strong greater-than bias in our mod 10 features suggests this feature-level description is missing some details – specifically that successor heads must use other numeric information to produce the greater-than bias we observe. Additionally, while we see a good generalization of the mod 10 features across various tasks in the table in Figure 6, the mod 10 features are not able to steer the Days and Letters tasks from Section 2. We describe this in Appendix I.



(a) Evaluating the effective OV circuit on the input and output tokens ‘first’, ‘second’, ..., ‘tenth’.



(b) Multiplying all mod 10 features f_i by $W_U W_{OV}$.

Figure 10: The Successor Head OV circuit displays a clear bias against decrementation (Figure 10a), i.e. the logits on or above the main diagonal are less than the logits below the main diagonal. This bias isn’t captured in the mod 10 feature (Figure 10b).

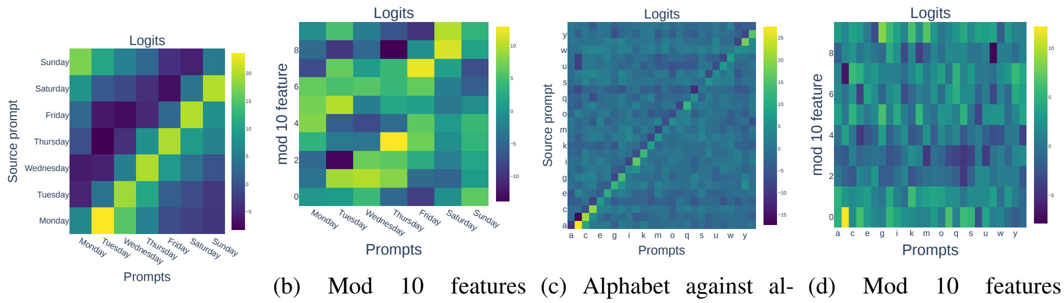
I Failure cases of mod 10 features

For the day and alphabet task, analogously to Figure 10, we look at the logits across the task and the mod 10 features. These are displayed in Figure 11, and demonstrate that our mod 10 features are not very interpretable in the context of days and the alphabet in terms of logits, with no clear diagonal of high logits.

J Successor Heads in the Wild

In this section, we analyze the behaviour of successor heads within natural-language datasets, and observe that they aren’t simply responsible for incrementation: indeed, we identify four distinct, interpretable categories of successor head behavior, highlighting successor heads as an example of an ***interpretably polysemantic*** attention head ‘in the wild’.

In order to characterize the behavior of Pythia-1.4B’s successor head on natural-language data, we sample 128 length-512 contexts from *The Pile*, and for each prefix of each context, we assess



(a) Days against days

(b) Mod 10 features against day tokens

(c) Alphabet against alphabet

(d) Mod 10 features against alphabet tokens

Figure 11: Logit plots across day and alphabet tasks, where attempting to steer the model with mod 10 features fails.

whether the successor head is important for the model's ability to predict the correct next token. We evaluate prefixes using two different metrics for per-prompt successor head importance:

Winning cases. We identify prefixes where the head that most decreases the logit for the correct next token under direct effect mean ablation is the successor head, denoting them as ***winning cases***.

Loss-reducing cases. We identify prefixes $\mathtt{p}$ where direct effect mean ablation of the successor head increases next-token prediction loss (by $\Delta\mathcal{L}(\mathtt{p})$), denoting them as ***loss-reducing cases***.

J.1 Interpretable Polysemanticity in Successor Heads

On analyzing prefixes where the successor head is particularly important for next-token prediction – i.e. loss-reducing and winning cases – we observe four main categories of behavior, which we operationalize as follows (denoting a ***top-n-attended token*** as a token at one of the top n positions to which the successor head attends most strongly):

Successorship behavior: *the successor head pushes for the successor of a token in the context.* We say this behavior occurs when one of the top-5-attended tokens is in the successorship dataset, and the correct next token is the successor of t.

Acronym behavior: *the successor head pushes for an acronym of words in the context.* We say this behavior occurs when the correct next token is an acronym whose last letter corresponds to the first letter of the top-1-attended token. (For example, if the successor head attends most strongly to the token 'Defense', and the correct next token is 'OSD'.)

Copying behavior: *the successor head pushes for a previous token in the context.* We say this behavior occurs when the correct next token t has already occurred in the prompt, and token t is one of the top-5-attended tokens.

Greater-than behavior: *the successor head pushes for a token greater than a previous token in the context.* We say this behavior occurs when we do not observe successorship behavior, but when the correct next token is still part of an ordinal sequence and has greater order than some top-5-attended token (e.g. if the successor head attends to the token 'first' and the model predicts the token 'third'.)

We plot the proportions of each behavior observed across winning cases in Figure 12, and the fraction of total reduced loss over all contexts ($\Delta\mathcal{L}$) attributable to contexts of each behavior in Figure 14. We also illustrate a random sample of 5 winning cases in Figure 13, and of 5 loss-reducing cases in Figure 15. We observe that, while successorship is the predominant behavior across both winning and loss-reducing cases, acronym and greater-than behaviors also form a non-negligible fraction of successor head behavior. In other words, the successor head is an example of an attention head with ***interpretable polysemanticity***[3]. While polysemanticity has been observed in both vision models [7] and toy models trained to perform simple tasks [1], to the best of our knowledge the presence of both successorship and acronym behavior in head L12H0 is the cleanest example of polysemantic behavior identified so far in an LLM.

[3]A component of a network is said to be *interpretably polysemantic* if it performs multiple distinct, interpretable functions.

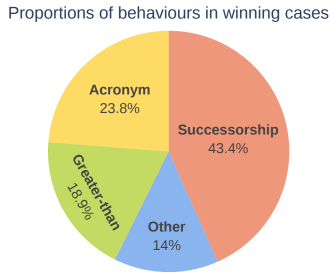


Figure 12: Proportions of three dominant behaviors across winning cases (copying behavior is negligible here).

Prompt	Completion
(...) [@B2] Hence, bonding to ceramic requires strict attention to detail for optimal clinical outcomes.[@B	3
(...) designated as boxazomycin A and	B
(...) called Generalized Single Step Single Solve (	GS
(...) More than two-	thirds
(...) where one or more access points (	AP

Figure 13: Random sample of 5 winning cases.

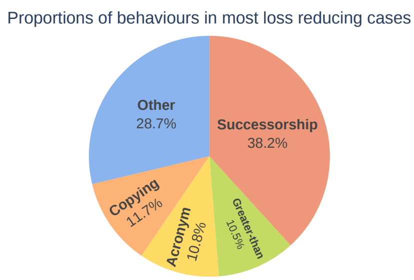


Figure 14: Proportions of reduced loss ($\Delta\mathcal{L}$) attributable to prompts of each behaviour.

Prompt	Completion
(...) low-dose administration of 14- and	15
(...) in the second round, let's get it to the	third
(...) for 1) investigators to ask the missionaries,	2
(...) You are using Microsoft Test Manager (	MT
(...) known as the Medical Research Council Technology (	MR

Figure 15: The top 5 most loss reducing examples.

Note that in this section, while we identified that successor heads are often used in tasks involving incrementation, we did not explicitly demonstrate that successor heads are *necessary* for incrementation. In Appendix O we describe an experiment that reveals that successor heads are necessary for a specific incrementation task (*numbered listing*).

K Residual Connections Are Not Important For Succession

To show that there is no relevant information in the residual stream, i.e. the path $W_U \text{MLP}_0(W_E)$ is not sufficient to predict successors, we perform an experiment using the Tuned Lens [20], which approximates the optimal predictions after a given layer inside a transformer.

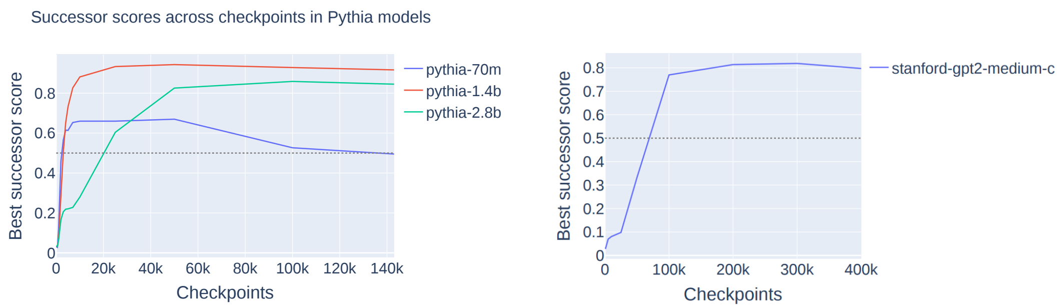
For all tasks in the succession dataset (Section 2), we used prompt formats (where | denotes a gap between tokens)

1. |Here| is| a| list|:| alpha| beta| gamma| and| here| is| another|:|<token1>|
2. |The|Monday|Tuesday|Wednesday| and| The|<token1>|

in order to measure how well models were able to predict the successor <token2> (e.g 'February') given the final token of the prompt was <token1> (e.g 'January'), as LLMs, predict successors given these prompts.

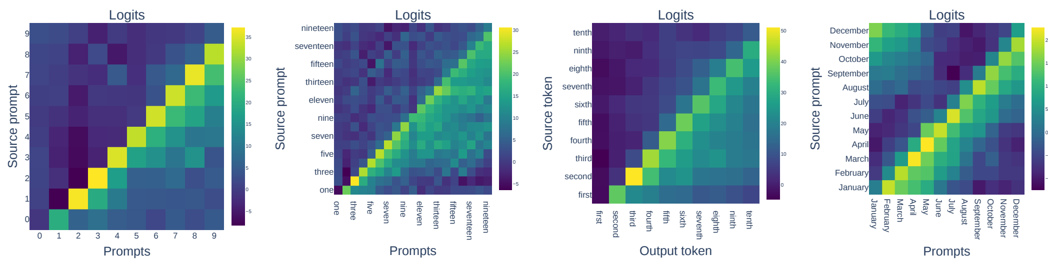
We then took GPT-2 Small and Pythia-1.4B's output after MLP0 and used the Tuned Lens to get logits on output tokens.[4] The resulting successor score was $<1\%$ and commonly predicted bigrams, such as <token1>=" first" giving " time" as a completion and <token1>=' Sunday' giving ' morning' as a prediction. This suggests that MLP0 information is insufficient for incrementation and the successor head is critical for succession.

[4]Note we did run with attention layer 0 to maximise the model's chances are being able to perform succession.



(a) Pythia models (b) Stanford GPT-2 Medium-C

Figure 16: Best successor scores across successor heads throughout training checkpoints for Pythia and stanford-gpt2 models.



(a) Numbers (b) Number words (c) Placements (d) Months

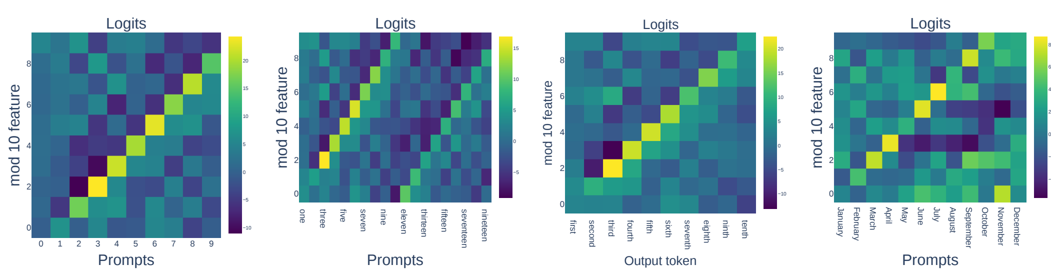
Figure 17: Plots of logits across various numeric classes, analogous to Figure 10a

L Testing successor score over training steps

Another line of evidence that Successor Heads are an important model component for low training loss can be found by studying successor scores across training points. We study a Pythia model [15] as well as a Stanford GPT model [21], as these models have training checkpoints. The emergence of Successor Heads throughout training is displayed in Figure 16.

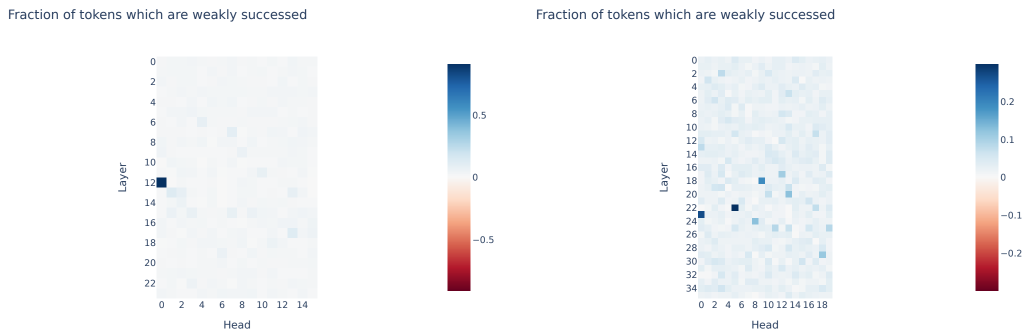
M Decrementation bias across different tasks

We show the strength of the decrementation bias in figures Figure 17 and 18.



(a) Numbers (b) Number words (c) Placements (d) Months

Figure 18: Plots of mod 10 feature logits across various numeric classes, analogous to Figure 10b



(a) Pythia-1.4B

(b) GPT-2 Large

Figure 19: Successor scores for Pythia-1.4B and GPT-2 Large.

Prompt	Answer
(...) (A) Colony formation and (<	B
(...) (i) $f_g^*(y)$ equals the factual density $f(y)$ for all $g \in \mathbb{G}$; (	ii
(...) [ˆ2]: Conceived and designed the experiments (...) [ˆ	3
(...) 6. Kirovsky Zavod Station – St. Petersburg, Russia (...) you can see a statue of Lenin here.	7
(...) [9] Minutes, Criminal Law Revision Commission, January 28, 1972, 16.[	10

Figure 20: Some examples of numbered listing prompts from the Pile dataset.

N All successor scores in a model

In Figure 19 we find that for both Pythia-1.4B (the mainline model in the paper) and GPT-2 Large (a randomly selected model without a successor head from Figure 2 on the left), the heads with highest successor score are sparse: in Pythia-1.4B L12H0 has eight times as great a successor score to the next higher successor score and in GPT-2 Large only three heads have a successor score that's above 1/10.

O Case study: numbered listing

In Appendix J we demonstrate that when the successor head is contributing usefully, the prompts often required some kind of incrementation. However, we want to investigate whether the converse holds: are prompts requiring incrementation mostly solved by successor heads?

Numbered listing is widespread across real datasets and requires incrementation. Additionally, even small LLMs are capable of this task in the case of incrementing citations.[5] Examples of prompts involving numbered listing can be seen in Figure 20.

We collect 64 such prompts and check for whether the successor head in Pythia-1.4b is the most important head for this prompt under direct effect mean ablation, and find that the successor head is indeed the most important head across **all** 64 prompts. Hence this provides some evidence prompts requiring incrementation in real datasets are indeed mostly solved by successor heads.

P Related Work

Mechanistic Interpretability research aims to reverse engineer trained neural networks analogously to how software binaries can be reverse-engineered [2]. This research was largely developed in vision models [22, 23] though most recent research has studied language models [9, 5] and trans-

[5] `https://www.lesswrong.com/posts/LkBmAGJgZX2tbwGKg`

formers [4]. Olah et al. [7] introduces the universality hypothesis and we use Chughtai, Chan, and Nanda [16]'s 'weak universality' notion in this work (Section 1).

Transformer circuits. More specifically, our work builds from the insights of Elhage et al. [9]'s framework for understanding circuits in transformers, including how autoregressive transformers have a residual stream. Due to the residual stream, different paths from input to output can bypass as many attention heads and MLPs as necessary. This has further been explored in specific case studies [6, 24] and generalizes to backwards passes [25]. One related case study to our work is Hanna, Liu, and Variengien [11] which studies a Greater-Than circuit in GPT-2 Small, similar to how we indirectly found the Greater-Than operation in Section 3. Hanna, Liu, and Variengien [11] focus mainly on numbers, not other tasks.

LLMs and arithmetic. Mikolov et al. [26]'s seminal work on word embedding arithmetic showed that latent language model representations had compositionality, e.g. 'King' $-$ 'Man' $+$ 'Woman' approximated the embedding of 'Queen'. Recently Merullo, Eickhoff, and Pavlick [27] showed some extension of these arithmetic results to LLMs. Additionally Subramani, Suresh, and Peters [28] and Turner et al. [29] use residual stream additions to steer models. Our work differs in that it considers shallow targeted paths through networks, rather than deep hidden states in networks.

Q Succession dataset

We present the full succession dataset in Table 2. Note that the days and months tasks are special as the final tokens in these classes ('Sunday' and 'December') have cyclical successors ('Monday' and 'January'); we don't consider the end tokens of the other tasks to have cyclical successors. Full details of our dataset can be found in our open-sourced experiments.[6]

Task	**Tokens**
Numbers	'1', '2', ..., '199', '200'
Number words	'one', 'two', ..., 'nineteen', 'twenty'
Cardinal words	'first', 'second', ..., 'tenth'
Days	'Monday', 'Tuesday', ..., 'Sunday'
Day prefixes	'Mon', 'Tue', ..., 'Sun'
Months	'January', 'February', ..., 'December'
Month prefixes	'Jan', 'Feb', ..., 'Dec'
Letters	'A', 'B', ..., 'Z'

Table 2: Tokens in the succession dataset

Glossary

W_{OV} The $\mathbb{R}^{d_{\text{model}}} \times \mathbb{R}^{d_{\text{model}}}$ matrix for an attention head that is the product of its W_O and W_V matrices. See Elhage et al. [9] for motivation.

Direct effect involves identifying the output attributed to the head irrespective of the behavior of other heads. This is different from the indirect effect, where the effects of ablating a head are hidden by a backup head..

Feature is a term we use to refer to an interpretable (linear) direction in activation space, inspired by definition 2 from Elhage et al. [1]. Since SAEs can be viewed as learning a set of directions which their W_{in} matrix reads in (i.e. their neurons pre-ReLU activations are dot products between these directions and the SAE input), we similarly refer to these directions as SAE features. SAEs have been found to be interpretable in prior and our paper (Section 3).

Linear probe is a learnable $\mathbb{R}^{10} \times \mathbb{R}^{d_{\text{model}}}$ matrix that acts on MLP0's output space to predict the mod-10 value of the underlying numeric token.

Mean ablation involves patching the output of a head with its mean output over a chosen distribution.

[6] We will release our code upon successful publication.

Numbered listing is a special case of incrementation where the token being incremented designates items in a list, such as ‘A’ and ‘B’ in ‘A) ... B)’ or ‘i’ and ‘ii’ in ‘[i] ... [ii]’. See Figure 20 for in-the-wild examples.

Ordinal sequence refers to a series of words arranged in a specific, meaningful order, where the position of each item is important.

Patching replaces part of a model’s forward pass with activations from a different input [6, 30] .

SAE stands for Sparse Autoencoder [17], see Appendix C.