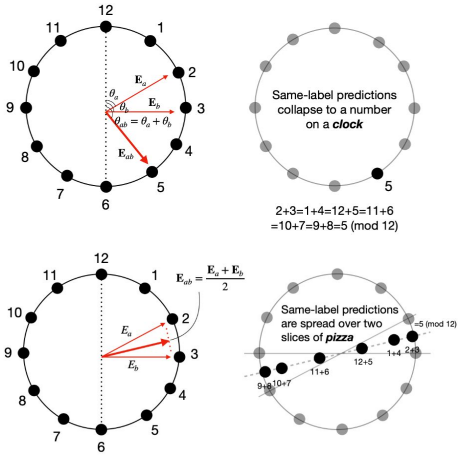
# Not All Language Model Features Are Linear

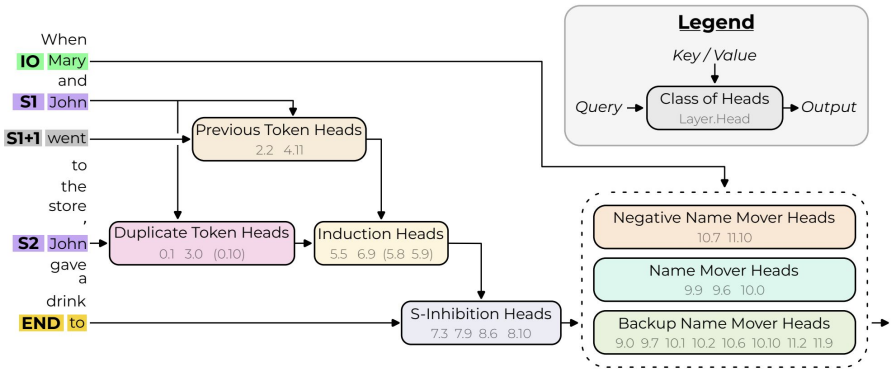



# Mechanistic Interpretability

- Goal: reverse engineer neural networks into human understandable algorithms
- In last few years, progress on toy models and LLMs



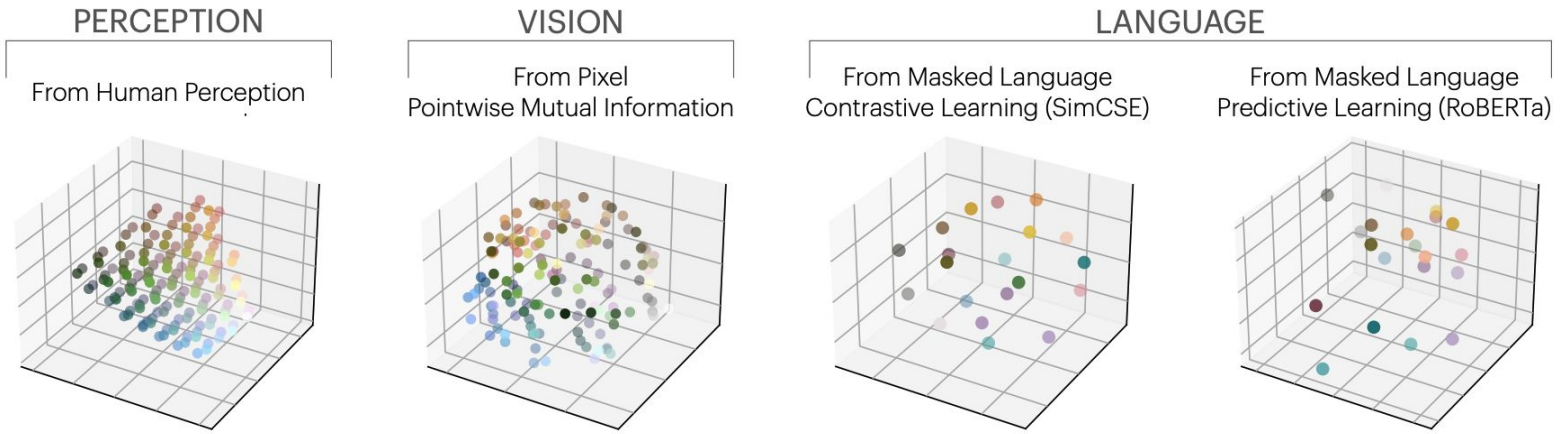

Zhong et al. 2023




Wang et. al. 2022

# Our Focus: Representations

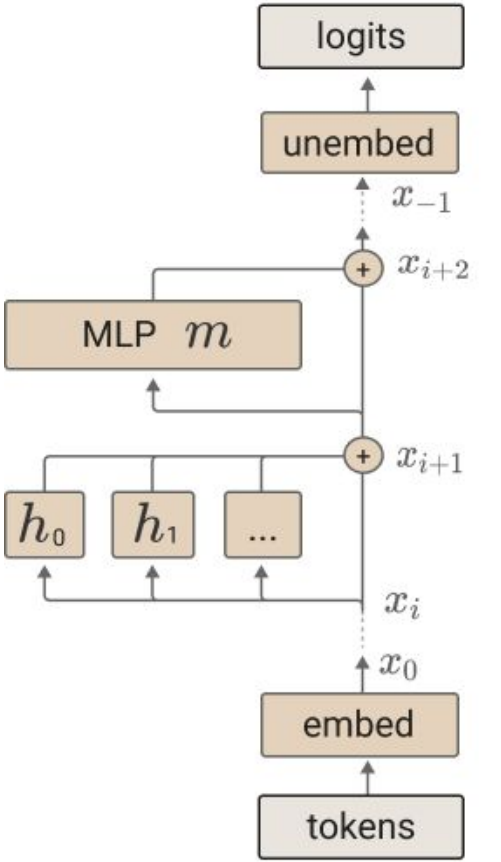
- What are the *variables* that models use for computation?
- Platonic representation hypothesis: representations are universal
  - So we can study just one or two specific models!




Huh et al. 2024

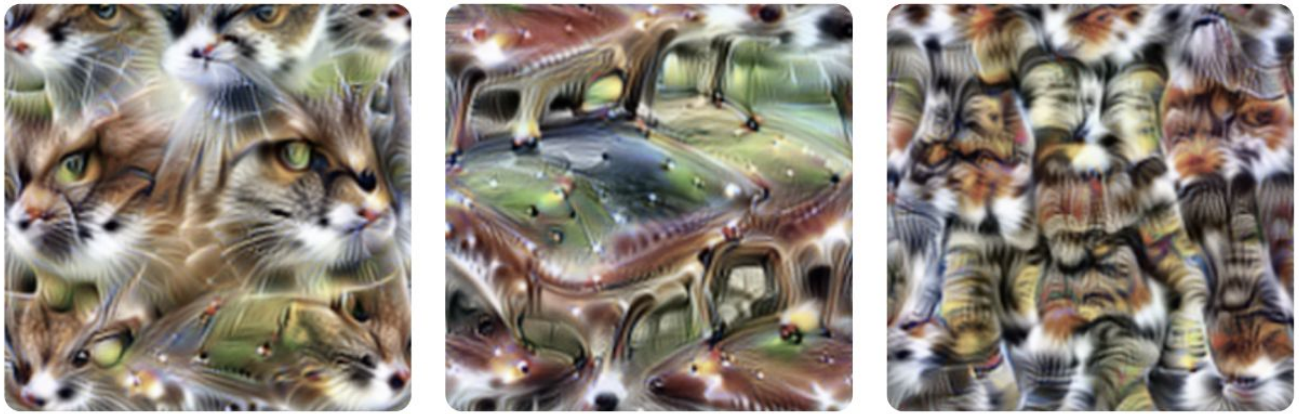
# How LLMs Work

- Transformer architecture:
  - Residual stream: sequence of high-d vectors, one per (token, layer) pair
  - Each layer: attention, MLP, update residual stream
- Residual stream vectors = snapshots of what the LLM is “thinking”
  - Only way to communicate across layers is residual stream, so most “variables” must be in the vector space somewhere!



# How Are Representations Stored in the Residual Stream?

- Simplest explanation: each dimension stores a single concept
- Unfortunately, usually not the case: neurons are polysemantic
    - In fact, this is necessary: an LLM knows far more than 10k concepts

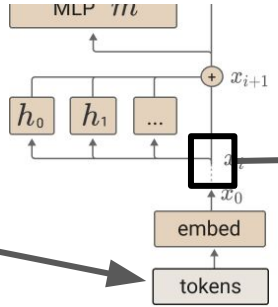


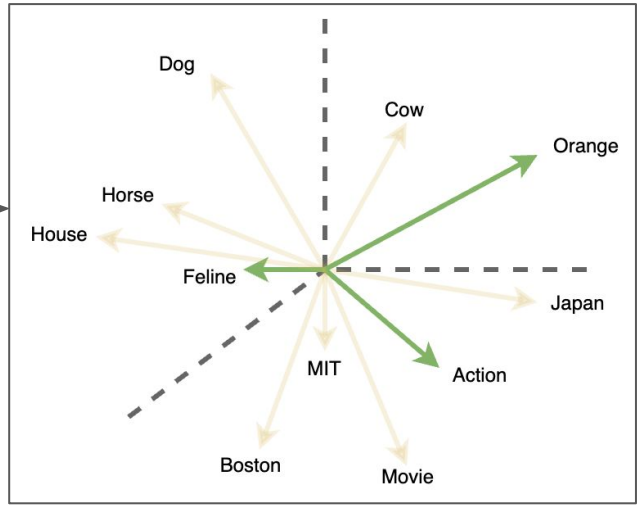
Olah et al. 2020

# Solution: Linear Representation Hypothesis + Superposition

- Activations are **sparse**, **linear** combinations of **meaningful feature vectors**
- Sparsity allows **>> d** feature vectors to be stored **almost orthogonally**
  - We say “in superposition”
- Models can decode superposed vectors or compute directly in superposition

The orange cat jumped



# Limitations Of The LRH

- What is the relationship between feature vectors? Can we say anything more than “almost orthogonal”?
- Are one dimensional vectors the correct model of an “atomic” feature? If not
  - What types of features might be inherently multi-dimensional?
  - What would they look like?
  - How could we find them?

# Multi-D Representation Hypothesis

- Activations are **sparse**, **linear** combinations of **meaningful feature vectors**

manifolds

aka ***multi-d features***

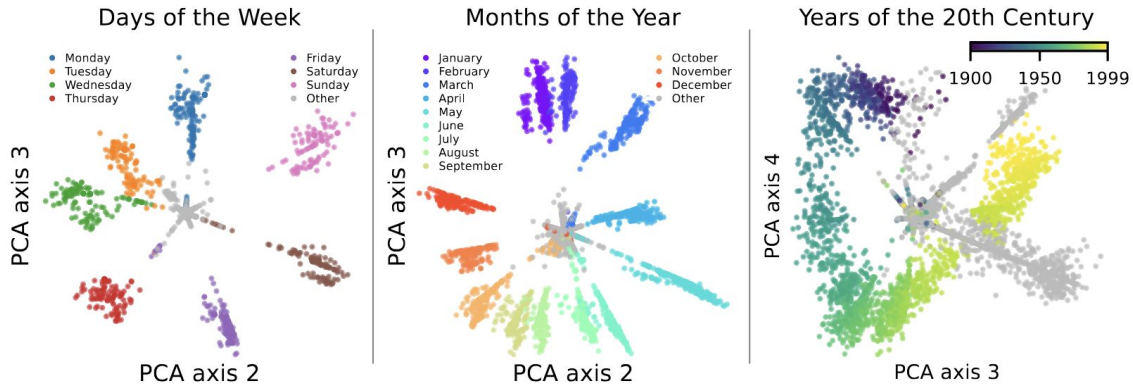
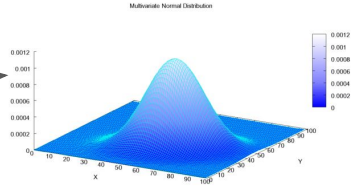
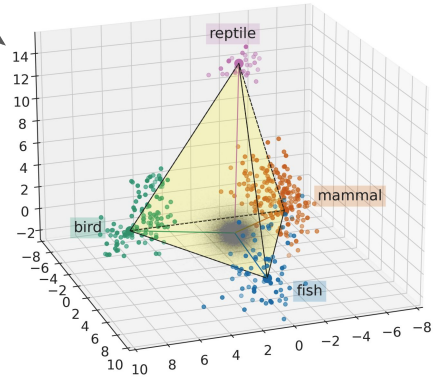



Figure 1: Circular representations of days of the week, months of the year, and years of the 20th century in layer 7 of GPT-2-small. These representations were discovered via clustering SAE dictionary elements, described in Section 4. Points are colored according to the token which created the representation. See Fig. 14 for other axes and Fig. 15 for similar plots for Mistral 7B.

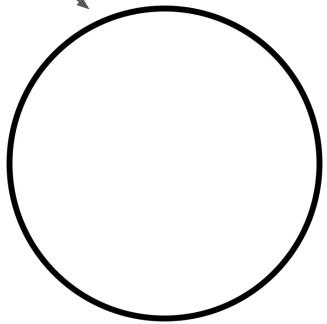
# Defining Multi-D Features

- k dimensional feature: function f that maps a subset of the input space into a k-dimensional point cloud
- Feature *reducible* if, across input distribution:
  - Composed of two independent lower dimensional features (e.g. gaussian)
  - Composed of two disjoint lower dimensional features (e.g. simplex)
- *Irreducible* if neither of these
  - E.g. a circle



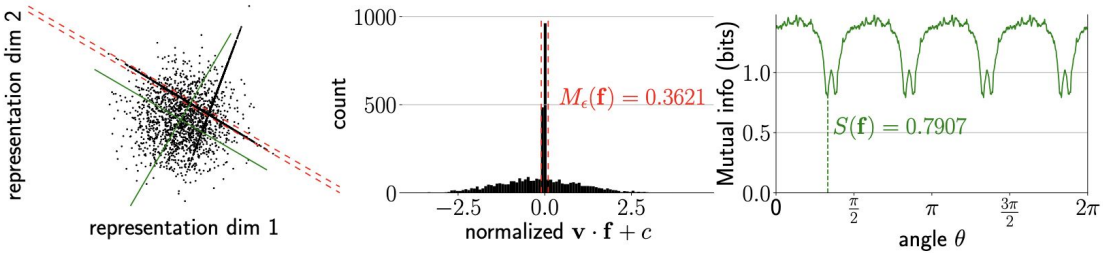



From Park et al. 2024

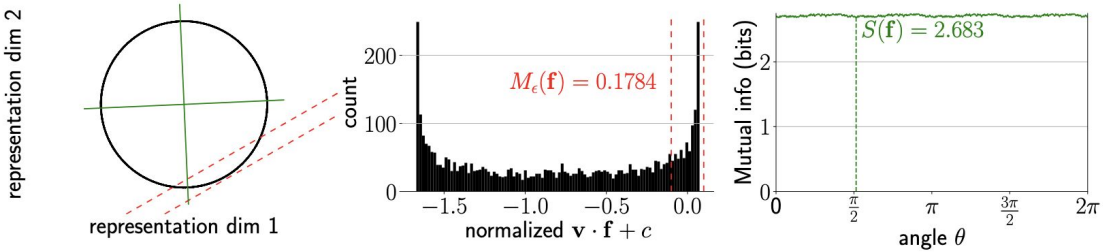


# Defining Multi-D Features In Practice

- We relax definitions
- Mixture index: how often can **f** be projected to zero
- Separability index: what is the minimum mutual information across all rotations of **f**



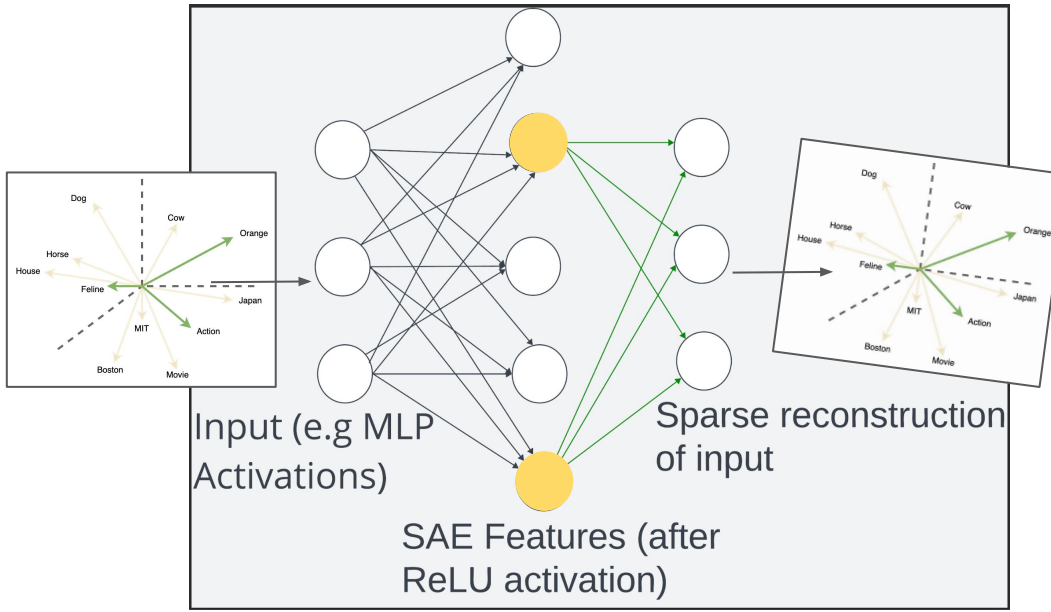

(a) Testing $S(\mathbf{a})$ and $M_\epsilon(\mathbf{a})$ on a reducible feature $\mathbf{a}$.




(b) Testing $S(\mathbf{b})$ and $M_\epsilon(\mathbf{b})$ on an irreducible feature $\mathbf{b}$

# Finding One-D Features: Sparse Autoencoders (SAEs)

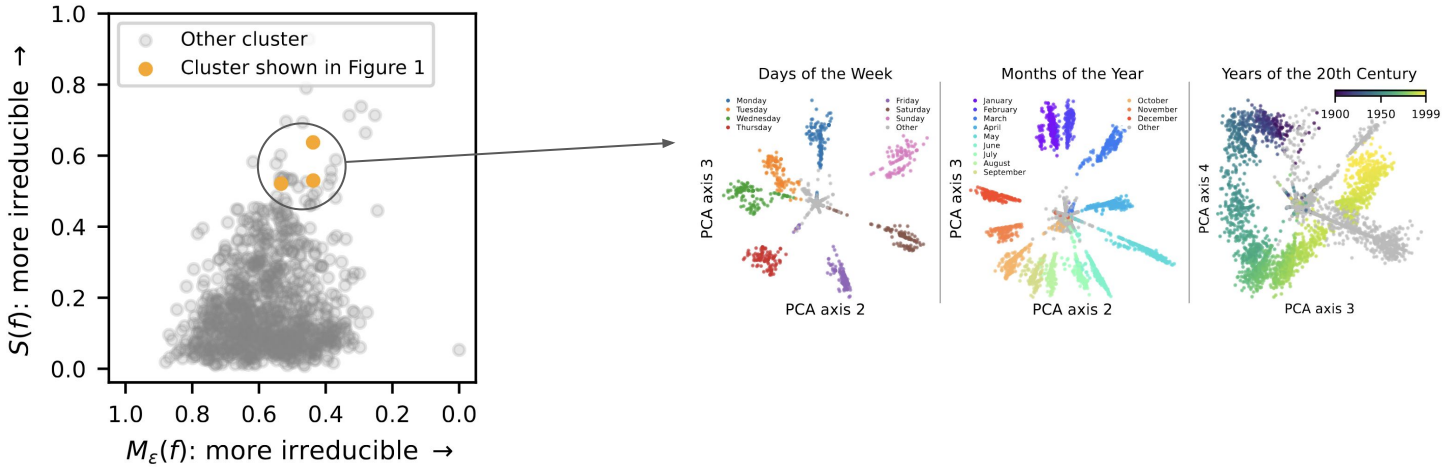
- **Key idea**: Train a (wide) autoencoder to reconstruct activations
- The hope: hidden state learns coefficients of features, and the decoder learns the meaningful feature vectors themselves
- L1 penalty on hidden activations



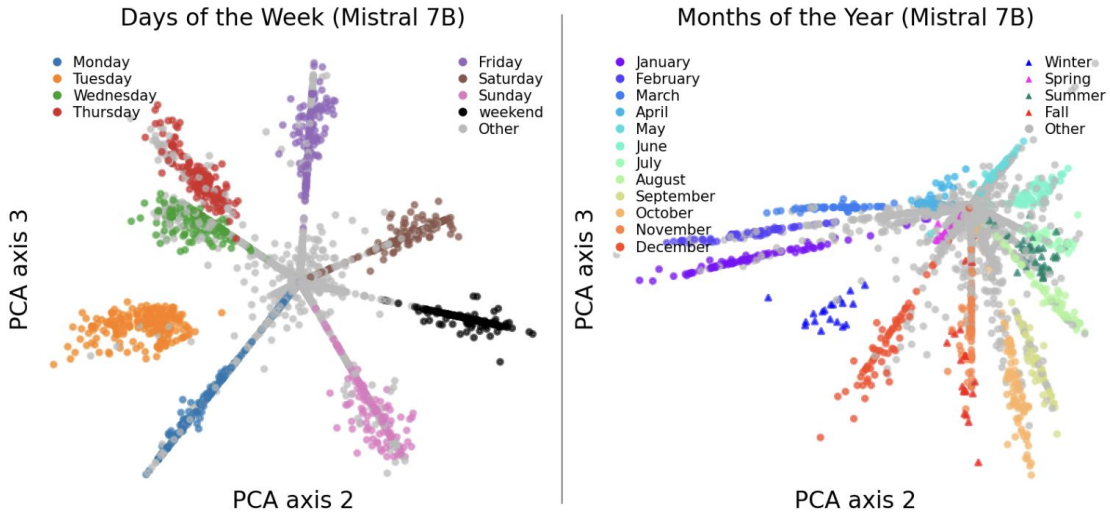

[slide adapted from Neel Nanda]

# Finding Multi-D Features: Also SAEs

- **Key idea:** many high similarity dictionary vectors will be learned for a given multi-d feature, so *cluster* SAE vectors
- To identify multi-d features, run mixture and separability tests on SAE reconstructions limited to each cluster



# We Find Some on Mistral 7B Too!



# When Do Models Use These Circles: Modular Addition?

- Define “natural” modular addition task, which models can do in practice:

`Weekdays` task: *“Let’s do some day of the week math. Two days from Monday is”*
`Months` task: *“Let’s do some calendar math. Four months from January is”*

| Model | `Weekdays` | `Months` |
|---|---|---|
| Llama 3 8B | 29 / 49 | 143 / 144 |
| Mistral 7B | 31 / 49 | 125 / 144 |
| GPT-2 | 8 / 49 | 10 / 144 |

# When Do Models Use These Circles: Modular Addition!

- Interventions show models do use the circle!

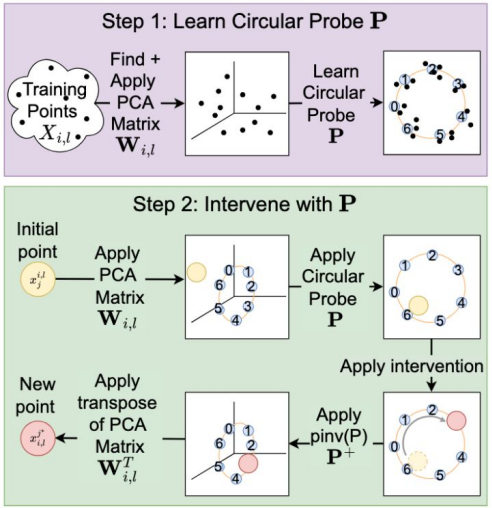



Figure 5: Visual representation of the intervention process.

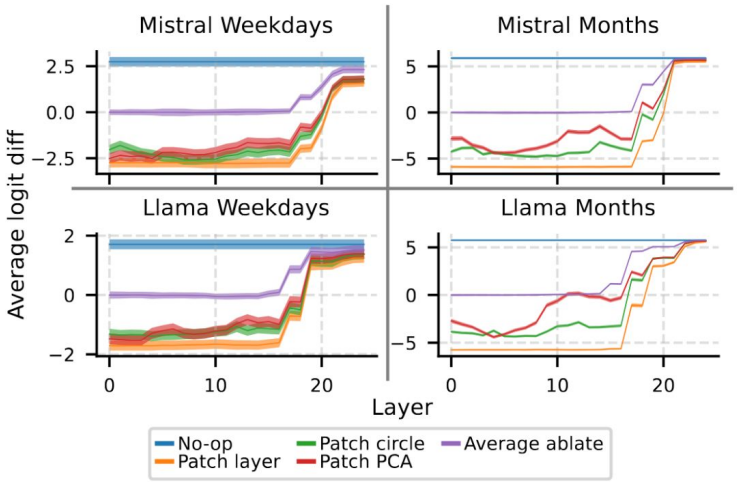
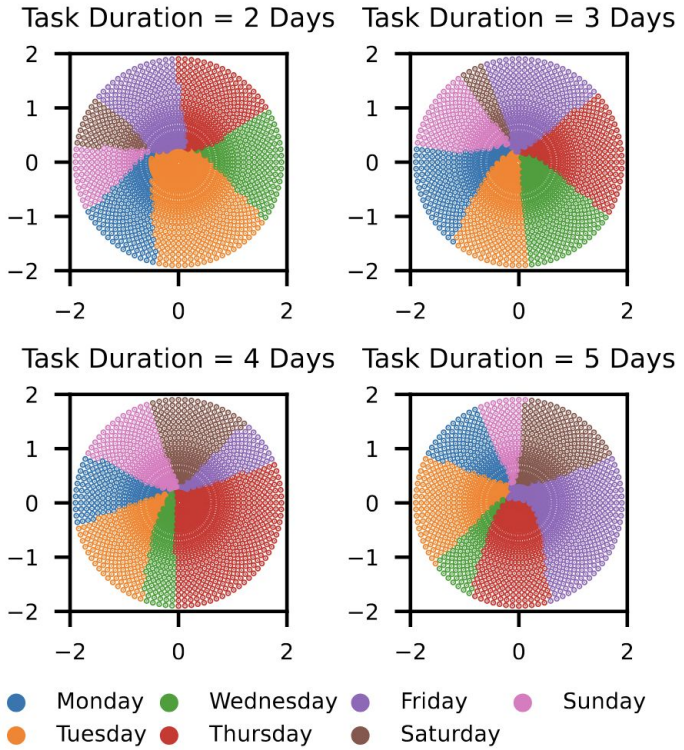


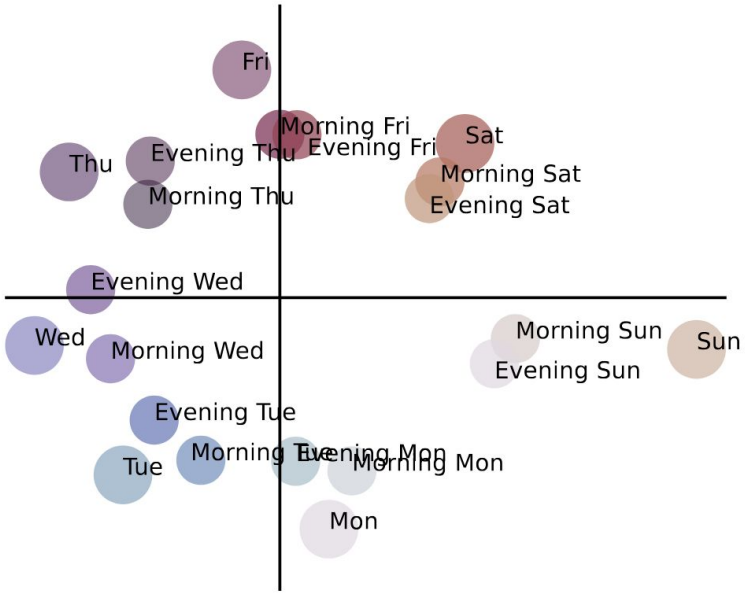

Figure 6: Mean and 96% error bars for intervention methods.

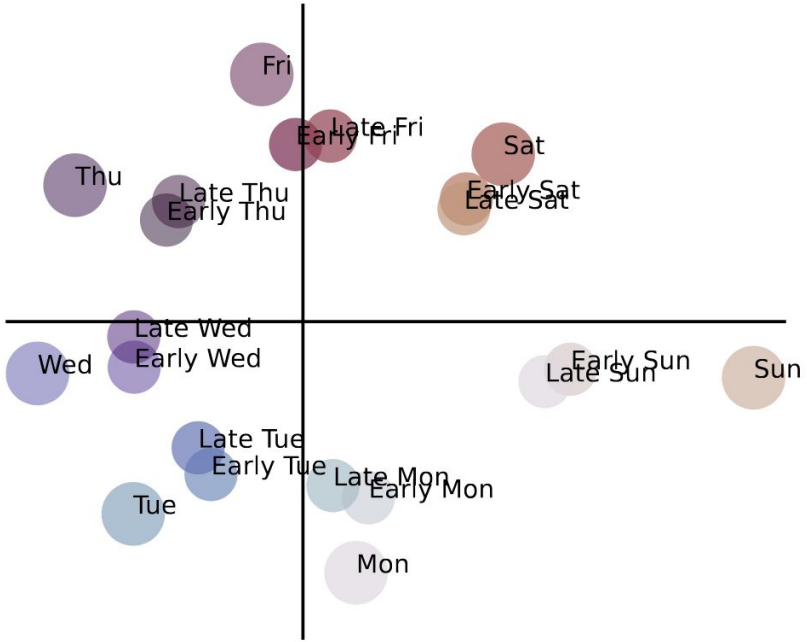
# Off Distribution Interventions

- Intervene everywhere in the circle (not just on 7 or 12 learned positions)
- Shows robustness of representation



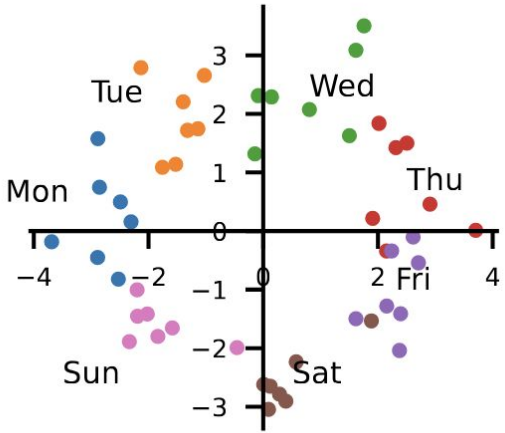
# Circular Representations Are Continuous



# Discovering Another Circle

- Method: use regression to remove components from activations that are linearly predictable from start day/month and duration
- In PCA, another circle in the *computed* day/month appears!



# Recent Followups

- Responding to our paper, Olah describes a two part LRH:
  - 1. Features are one-dimensional
  - 2. Feature magnitude correlates with intensity
- Recently, Csordás et. al. show evidence against part 2 by finding “onion-like” features in a toy RNN; our evidence is limited to be against part 1.
- Multiple papers on structure in SAEs

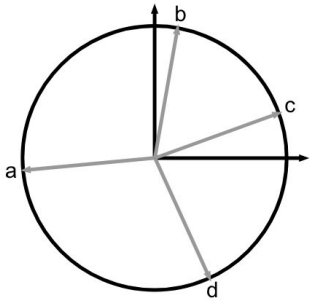
# Break for questions

# Open Questions on Multi-D Features

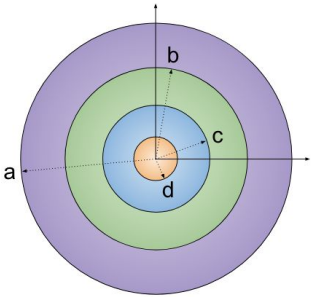
- Would be great to find stronger proof that model is really “rotating” circles
- How many features are best thought of as multi-d vs. one-d?
- Can we make SAEs learn multi-d features in the first place?
    - SAEs seem to characterize this continuous manifold of features, breaking it down into discrete pieces seems a bit naive

# Big Open Question: Are There True Nonlinear Features?

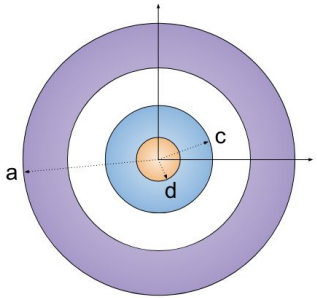
- Csordás et. al. find onion like features on a toy memorization task in RNNs
- Finding these in an LLM would be a death knell for the strong LRH
  - Still could be room for a weak LRH, where *most* features are linear + contained in a low-d subspace



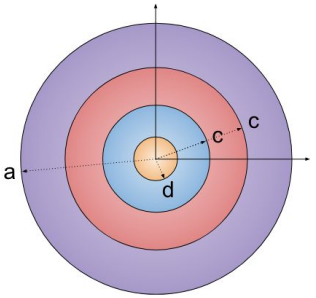
(a) Token Embeddings



(b) Onion for $(a, b, c, d)$.



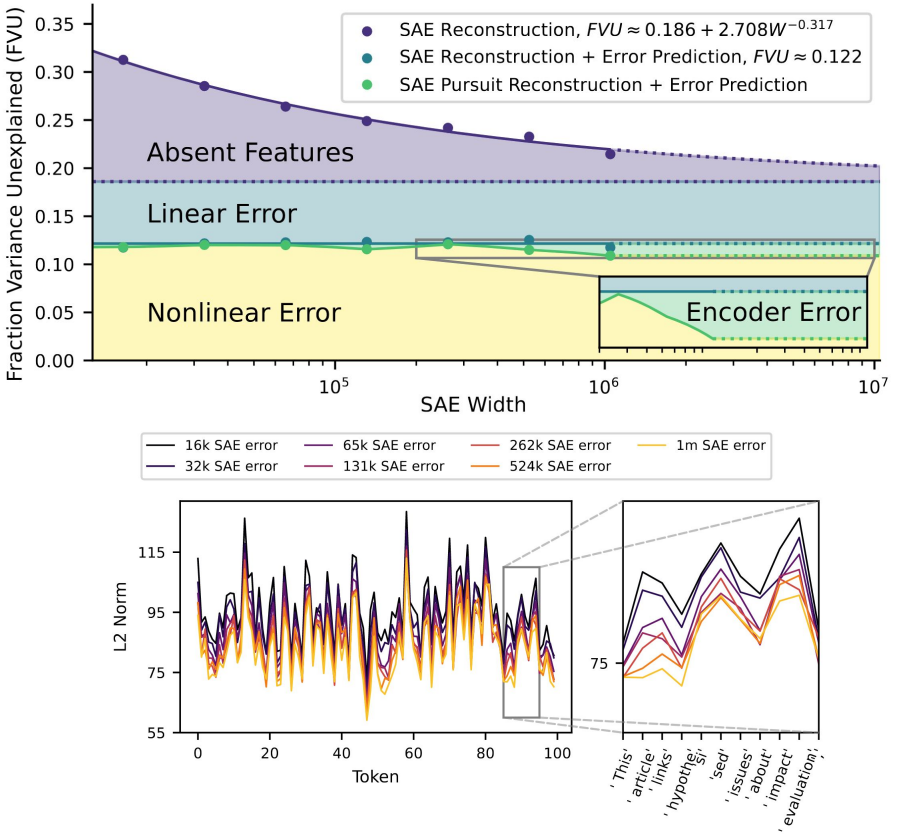
(c) Onion for $(a, \cdot, c, d)$.



(d) Onion for $(a, c, c, d)$.

# Are SAEs Missing Features?

- Examine how SAE errors scale
- Some specific contexts that SAEs of any size do badly on
- Presence of linearly predictable error as we scale = part of activation space we just aren't learning

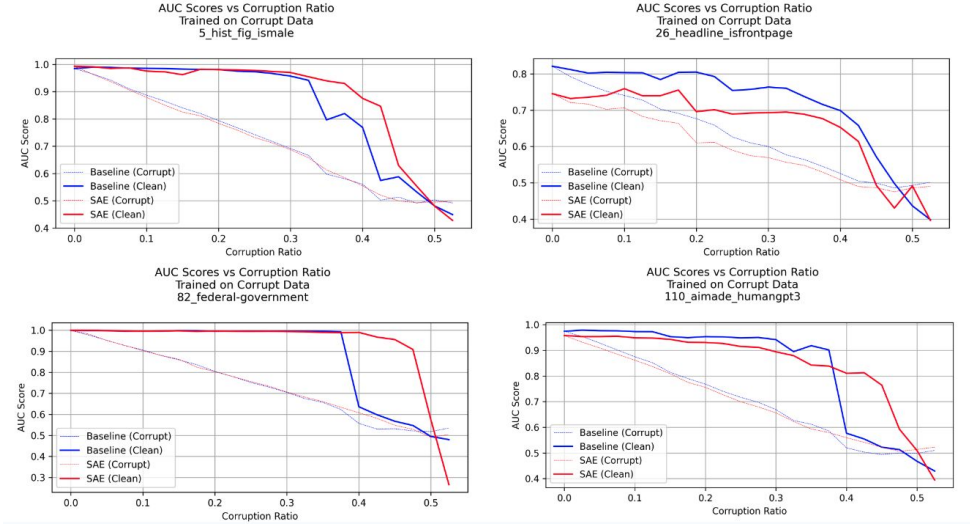


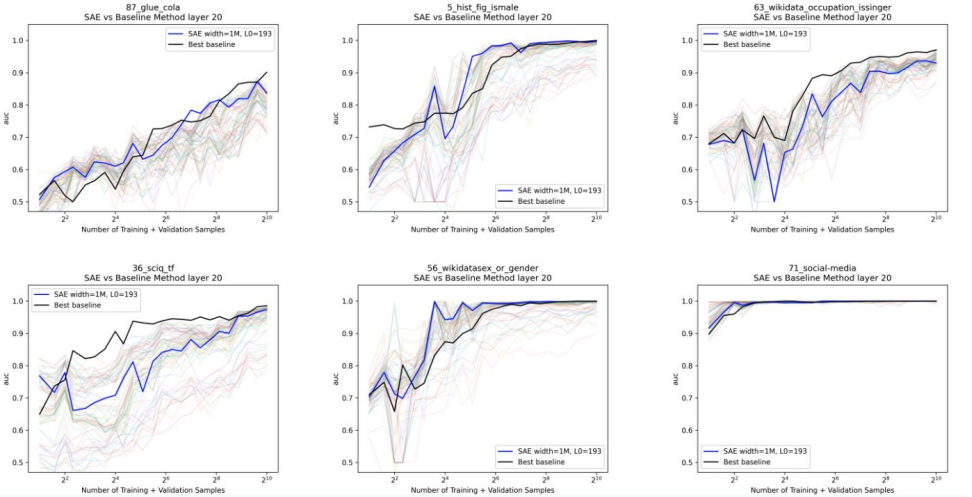
# Are SAEs Actually Useful For Anything?

- We know they struggle with:
  - Composition
  - Numerical features
  - Hiercharchy (feature splitting_
  - Interpretability!
- They do push the faithfulness vs. “interpretable” frontier
  - Maybe incremental progress will get us to ambitious mech interp goals?
- Some work showing maybe already useful:
  - Circuits
  - Probing

# Probing

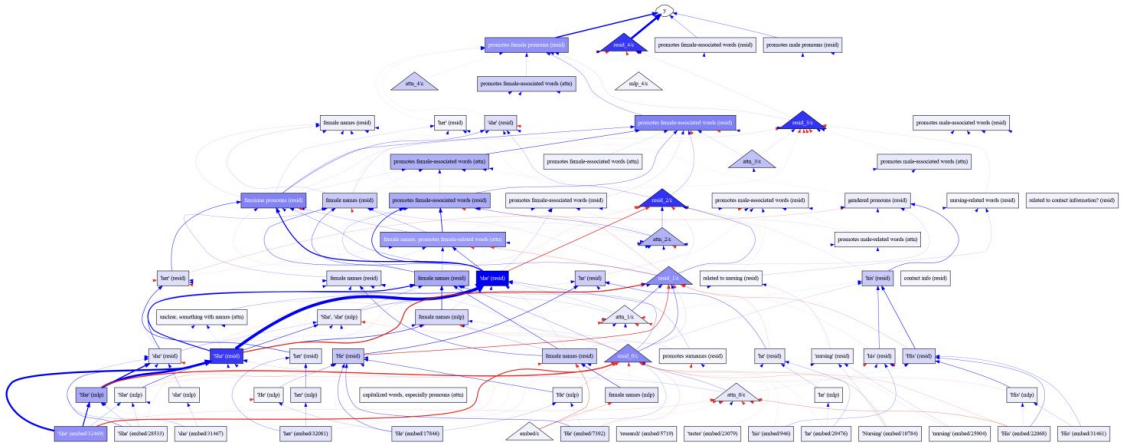
- Probing = logistic regression on hidden states
- Can predict:
  - Truthfulness
  - Gender bias
  - Buggy code
  - etc.
- SAEs are better in low data and corrupt data regimes
  - Evidence the SAE basis is a good inductive bias?



# Circuits

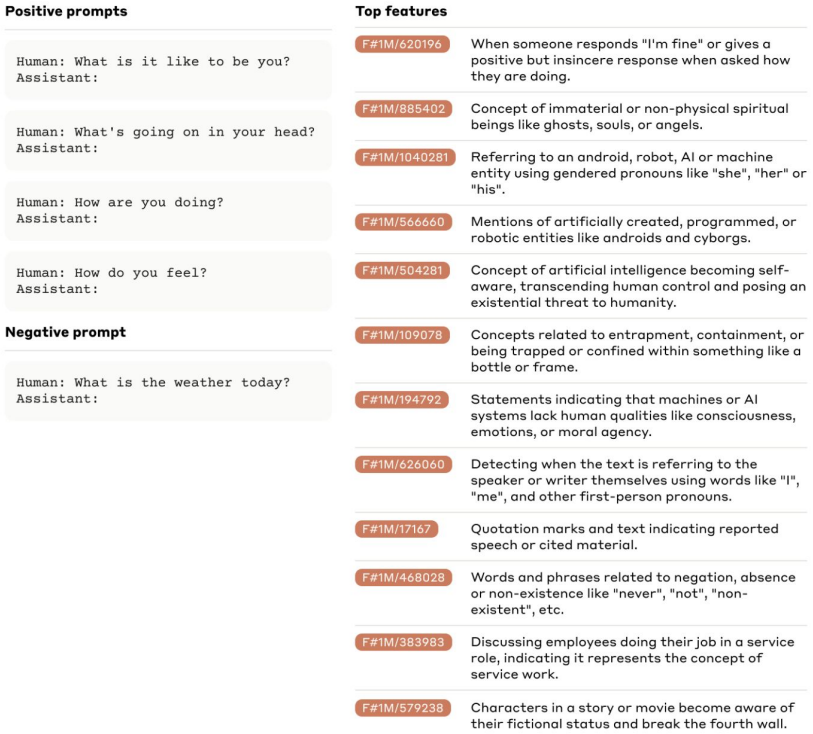
- SAE circuits win on faithfulness / correctness vs. # of components vs. neurons
- Case study: can reduce gender bias by ablating obviously spurious features



Marks 2024

# Can SAEs Tell Us Something About Consciousness?

- Anthropic find: when you ask the model how it is doing, SAE features fire for all sorts of things related to AI and consciousness and breaking the fourth wall and ghosts
  - They kind of brush this off, but it seems kind of important?
- How can we distinguish the model thinking about consciousness/itself vs. actually being conscious?
  - Can we apply consciousness definitions to SAE circuits?




Anthropic 2024

# How Realistic Is The Reverse Engineering Agenda?

- Recent work argues finding circuits might be computationally very difficult
- Are features even the right way of looking at the model?
  - Self repair
- Even if most features are linear + we can find the circuits, turning into a symbolic program might be very hard
  - Can we have case studies of LLM circuits where it is possible?
  - Where it is impossible?

Table 4: Classical and parameterized complexity results by problem variant.

| Classical & parameterized problems[2] $\mathcal{P}_{\mathcal{M}} = \{\hat{L}, \hat{U}_I, \hat{U}_O, \hat{W}, \hat{B}\}$ $\mathcal{P}_{\mathcal{C}} = \{\hat{l}, \hat{l}_w, \hat{u}, \hat{u}_I, \hat{u}_O, \hat{w}, \hat{b}\}$ | Problem variants | | | |
|---|---|---|---|---|
| | *Local* | | *Global* | |
| | *Decision/Search* | *Optimization* | *Decision/Search* | *Optimization* |
| SUFFICIENT CIRCUIT (SC) | NP-complete | {$c$, PTAS, 3PA}-inapproximable | $\Sigma_2^p$-complete | {$c$, PTAS, 3PA}-inapproximable |
| $\mathcal{P}$-SC ($\mathcal{P} = \mathcal{P}_{\mathcal{M}} \cup \mathcal{P}_{\mathcal{C}}$) | W[1]-hard | {$c$, PTAS, 3PA}-inapproximable | W[1]-hard | {$c$, PTAS, 3PA}-inapproximable |
| Minimal SC | NP-complete | ? | $\in \Sigma_2^p$ NP-hard | ? |
| $\mathcal{P}$-Minimal SC | W[1]-hard | ? | W[1]-hard | ? |
| Unbounded Minimal SC | ? | N/A | $\in \Sigma_2^p$ NP-hard | N/A |
| $\mathcal{P}$-Unbounded Minimal SC | ? | | ? | |
| Unbounded Quasi-Minimal SC | PTIME | | ? | |
| Count SC | #P-complete | N/A | #P-hard | N/A |
| $\mathcal{P}$-Count SC | #W[1]-hard | | #W[1]-hard | |
| Count Minimal SC | #P-complete | | #P-hard | |
| $\mathcal{P}$-Count Minimal SC | #W[1]-hard | | #W[1]-hard | |
| Count Unbounded Minimal SC | #P-complete | | #P-hard | |
| GNOSTIC NEURON (GN) | PTIME | N/A | ? | N/A |
| CIRCUIT ABLATION (CA) | NP-complete | {$c$, PTAS, 3PA}-inapproximable | $\in \Sigma_2^p$ NP-hard | {$c$, PTAS, 3PA}-inapproximable |
| $\{\hat{L}, \hat{U}_I, \hat{U}_O, \hat{W}, \hat{B}, \hat{u}\}$-CA | W[1]-hard | {$c$, PTAS, 3PA}-inapproximable | W[1]-hard | {$c$, PTAS, 3PA}-inapproximable |
| CIRCUIT CLAMPING (CC) | NP-complete | {$c$, PTAS, 3PA}-inapproximable | $\in \Sigma_2^p$ NP-hard | {$c$, PTAS, 3PA}-inapproximable |
| $\{\hat{L}, \hat{U}_O, \hat{W}, \hat{B}, \hat{u}\}$-CC | W[1]-hard | {$c$, PTAS, 3PA}-inapproximable | W[1]-hard | {$c$, PTAS, 3PA}-inapproximable |
| CIRCUIT PATCHING (CP) | NP-complete | {$c$, PTAS, 3PA}-inapproximable | $\in \Sigma_2^p$ NP-hard | {$c$, PTAS, 3PA}-inapproximable |
| $\{\hat{L}, \hat{U}_O, \hat{W}, \hat{B}, \hat{u}\}$-CP | W[2]-hard | {$c$, PTAS, 3PA}-inapproximable | W[2]-hard | {$c$, PTAS, 3PA}-inapproximable |
| Unbounded Quasi-Minimal CP | PTIME | N/A | ? | N/A |
| NECESSARY CIRCUIT (NC) | $\in \Sigma_2^p$ NP-hard | {$c$, PTAS, 3PA}-inapproximable | $\in \Sigma_2^p$ NP-hard | {$c$, PTAS, 3PA}-inapproximable |
| $\{\hat{L}, \hat{U}_I, \hat{U}_O, \hat{W}, \hat{u}\}$-NC | W[1]-hard | {$c$, PTAS, 3PA}-inapproximable | W[1]-hard | {$c$, PTAS, 3PA}-inapproximable |
| Count NC | ? | ? | ? | ? |
| CIRCUIT ROBUSTNESS (CR) | coNP-complete | ? | $\in \Pi_2^p$ coNP-hard | ? |
| $\{\hat{L}, \hat{U}_I, \hat{U}_O, \hat{W}, \hat{B}, \hat{u}\}$-CR | coW[1]-hard | ? | coW[1]-hard | ? |
| $\{|H|\}$-CR | FPT | FPT | ? | ? |
| $\{|H|, \hat{U}_I\}$-CR | FPT | FPT | FPT | FPT |
| SUFFICIENT REASONS (SR) | $\in \Sigma_2^p$ NP-hard | {3PA}-inapproximable | N/A | |
| $\{\hat{L}, \hat{U}_O, \hat{W}, \hat{B}, \hat{u}\}$-SR | W[1]-hard | {3PA}-inapproximable | | |

Adolfi et. al. 2024

# What’s Next?

- Everyone is just scaling SAEs
  - Can we come up with a better model for LLM “thoughts”?
- People have decided we’re stuck with the models we have now
  - Can we train more interpretable models w/ low tax?
- Much of the promise of mech interp seems far off
  - What can concretely do now to help AI safety?
- Curious what everyone else thinks!

# Questions