Yale

Mechanistic Interpretability for Large Language Models

CPSC 4710/5710: Trustworthy Deep Learning

Rex Ying

Can we understand *why* LLMs behave the way they do?

- **Introduction to Mechanistic Interpretability**
- **Early Interpretability Efforts and Semantics**
- **Sparse Autoencoders**
- **Circuit Theory and ACDC**

- **Introduction to Mechanistic Interpretability**
- **Early Interpretability Efforts and Semantics**
- **Sparse Autoencoders**
- **Circuit Theory and ACDC**

Mechanistic Interpretability

Mechanistic interpretability seeks to reverse engineer neural networks, similar to how one might reverse engineer a compiled binary computer program.

~ Chris Olah, Anthropic

A term to distinguish early interpretability efforts (eg: saliency maps) from latter-day efforts to **uncover *algorithms* learned by the weights of LLMs**

Mechanistic Interpretability

Mechanistic interpretability seeks to reverse engineer neural networks, similar to how one might reverse engineer a compiled binary computer program.

~ Chris Olah, Anthropic

Mechanistic Interpretability intervenes on different components of LLMs to understand how they contribute to the output for a given input query.

Existing Mechanistic Interpretability Landscape

- Mechanistic Interpretability is used to both **understand** and **manipulate** algorithms learned by LLMs
- There are several tools widely today:
  - **Sparse Autoencoders**
  - Transcoders
  - Activation Patching
  - Logit Lens
  - **Circuit Theory**
  - Probing
  - Token Geometry analysis
- Mechanistic Interpretability is rapidly evolving → new developments come out very quickly!
- The underlying premise:

To control an LLM, we must know what it knows about the world.

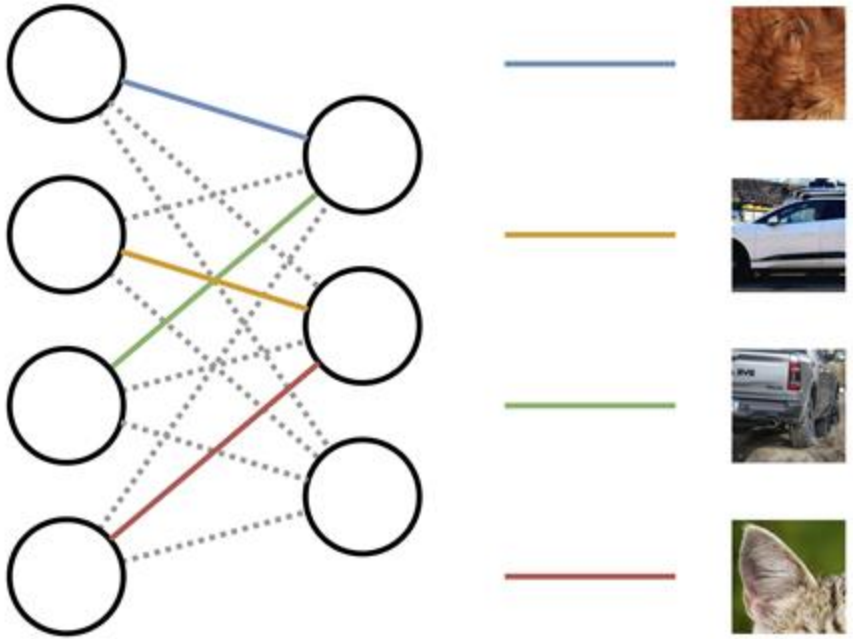
What does each dimension along the embedding correspond to?

Can we associate human-understandable concepts to values seen inside the neural network?

- Introduction to Mechanistic Interpretability
- **Early Interpretability Efforts and Semantics**
- Sparse Autoencoders
- Circuit Theory and ACDC

Early Interpretability Efforts

- Originates in computer vision
- Single neurons (or nearby neurons) are studied for their ability to understand concepts from images
- We see when they do and do not activate when we pass an image through the NN

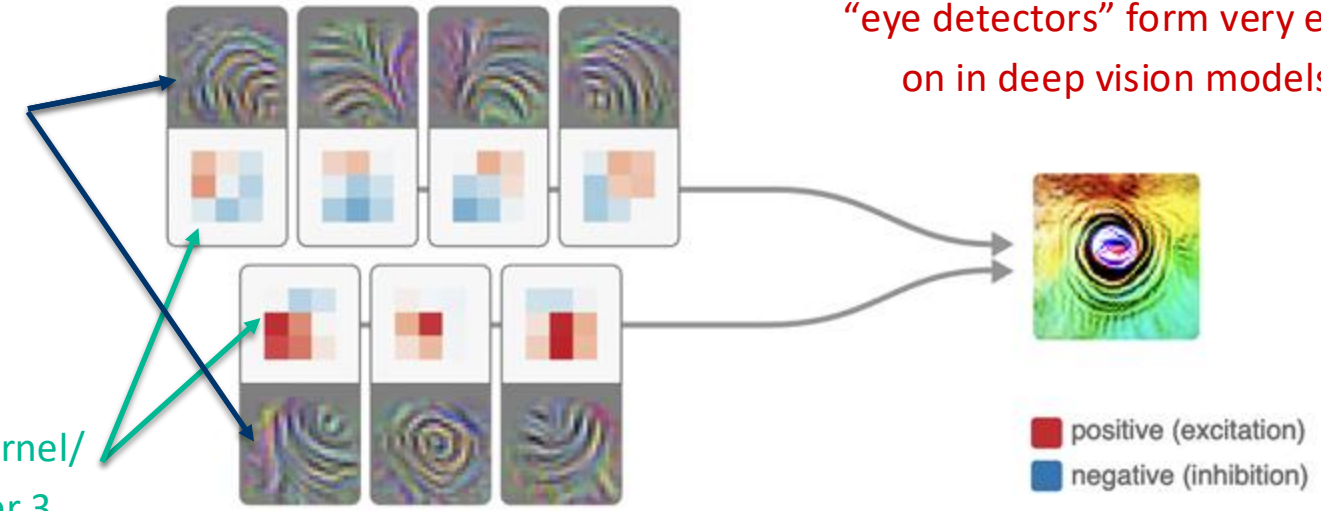


A VM learns “concepts” from images and encodes them within its neurons

Early Interpretability Efforts

Activation map from Layer 3

Convolutional kernel/ filter from Layer 3



"eye detectors" form very early on in deep vision models

Features are spatially resolved despite VMs being under-parameterized.

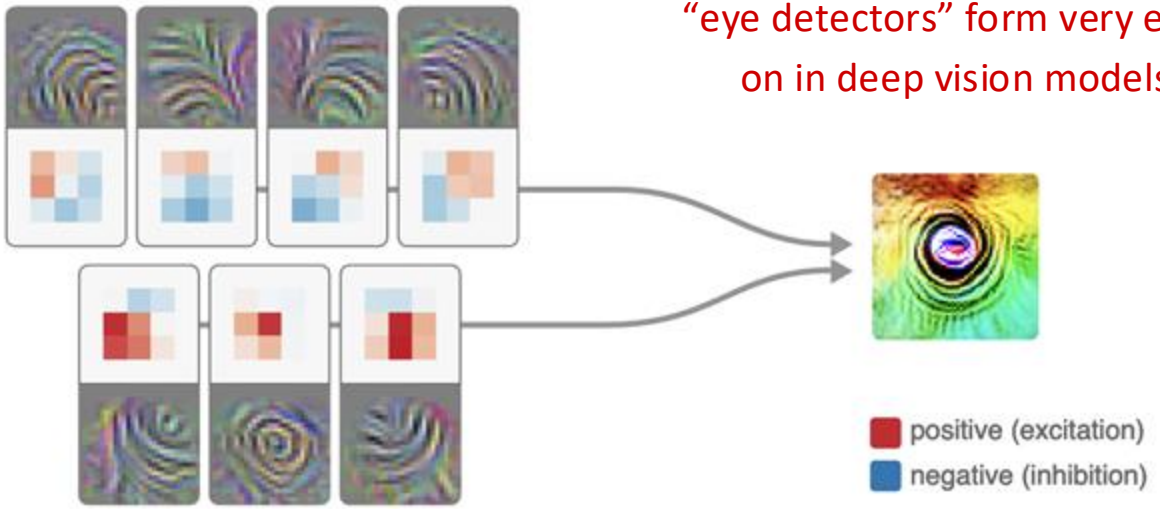
Olah et al. (2020). An Overview of Early Vision in InceptionV1

Early Interpretability Efforts

Remember GradCAM?

Gradients of these filters w.r.t. the input image are computed to obtain a heatmap.

This heatmap is overlaid on the input image, which is a good measure of *saliency*.



"eye detectors" form very early on in deep vision models

Features are spatially resolved despite VMs being under-parameterized.

Olah et al. (2020). An Overview of Early Vision in InceptionV1

Dealing with Polysemanticity

- *Concepts* refer to human-understandable ideas (concrete or abstract)
  - Eg: texture, color, real-world entities, shape, complex emotions like "anguish"
- In VMs, single/nearby neurons usually encode just one concept (eg: fur texture, eyes)
- In LMs, the same neuron activates for many seemingly-unrelated concepts [Elhage et al., 2022]

Realization: LLMs “overload” the same neuron with different concepts

- We call this phenomenon *polysemanticity* → we ideally want *monosemanticity*
- We need to **disentangle** these representations within LMs

- Introduction to Mechanistic Interpretability
- Early Interpretability Efforts and Semantics
- **Sparse Autoencoders**
- Circuit Theory and ACDC

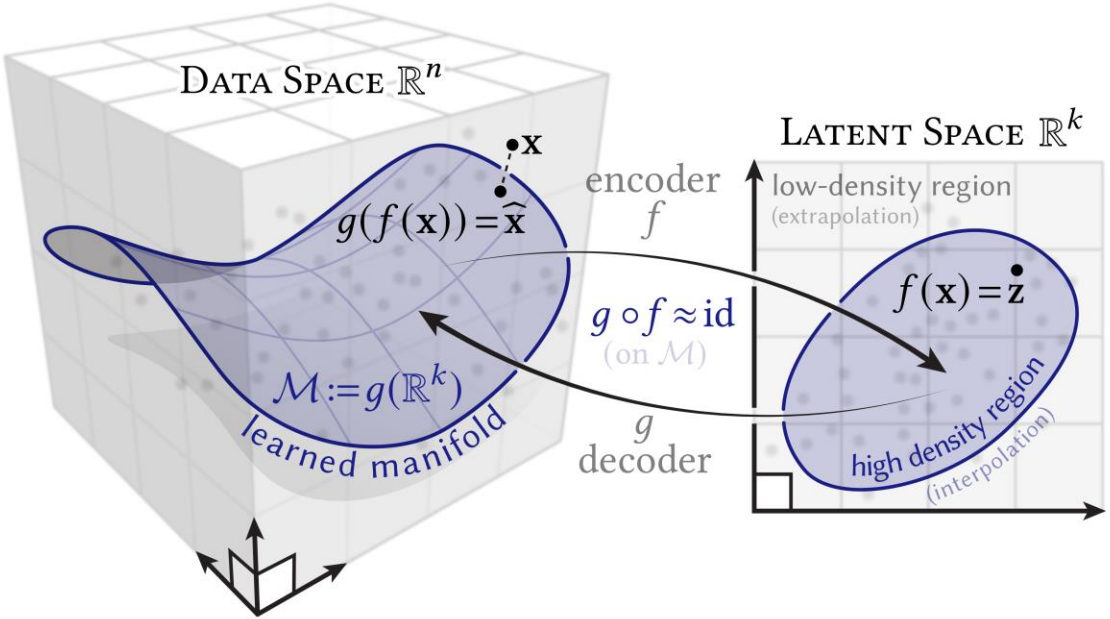
Disentangled Representations

- LMs are **under-parameterized** compared to dataset size
  - Eg: ChatGPT was trained on ~13T tokens but only has ~175B parameters
- Language represents semantic patterns that are not spatially resolved
  - Naturally, LM neurons will encode multiple, unrelated concepts
  - These concepts interfere with each other, making them difficult to identify one by one
- Disentangled representations are complex patterns inside a neural network **decomposed into simpler parts**
- Simpler parts allow us to understand the bigger story on what the neuron is extracting from the input data / prompt

Can we map finite-dimensional embeddings to a different vector space so that different concepts can reside in different far-apart subspaces?

Refresher: Autoencoders

- Unsupervised neural network that compress data into a latent space
- Data lies on a manifold that we wish to learn via gradient descent



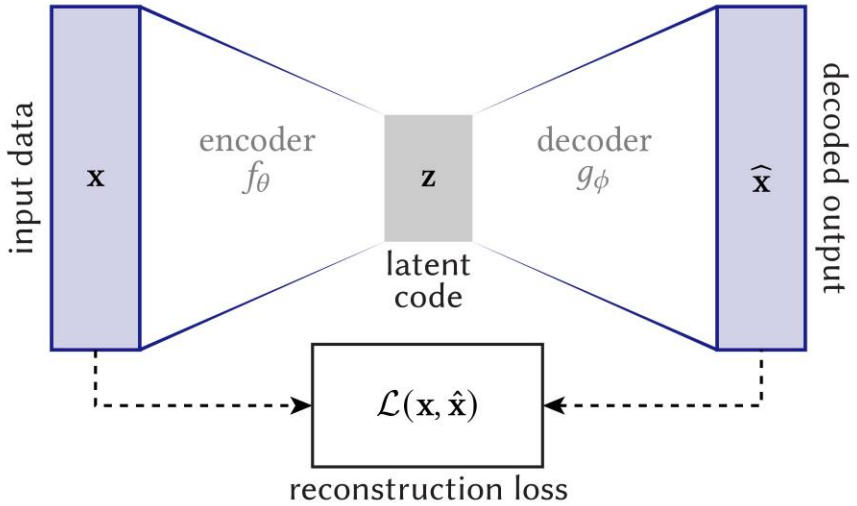
Source: https://x.com/keenanisalive/status/1964434335911858552

Refresher: Autoencoders

Two-part architecture with a reconstruction loss.

Encoder *f* : Deep neural network that that compresses data into a lower dimension

Decoder *g* : Deep neural network that reconstructs latent point to data dimension



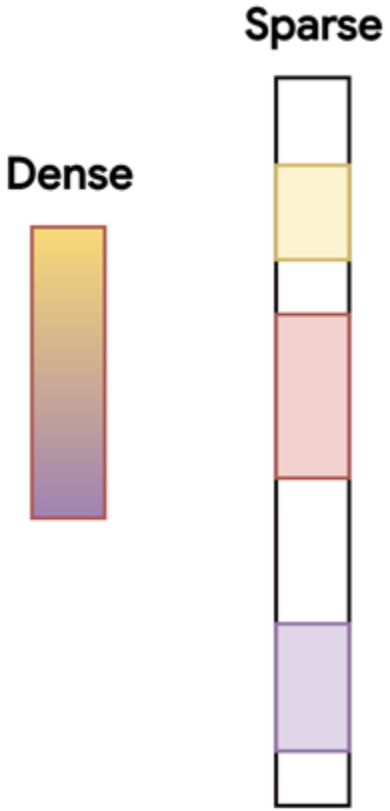
Source: https://x.com/keenanisalive/status/1964434335911858552

Sparse Autoencoders

- The goal is to learn highly ***monosemantic*** features representing simple ideas
- Token embeddings are dense vectors in $\boldsymbol{R^d}$
    - We must decompose dense vectors into features that correspond to human-understandable concepts
- **Dictionary Learning**: each learned feature typically corresponds to a specific concept, entity, or linguistic pattern

Sparsity and Disentanglement

- **Sparsity prevents polysemantic neurons**
- Forces model to allocate separable features for distinct concepts rather than superimposing them together
- **Linear separability** allows each concept to occupy different dimensions in the sparse vector
- Features corresponding to a concept don’t accidentally interfere with unrelated features



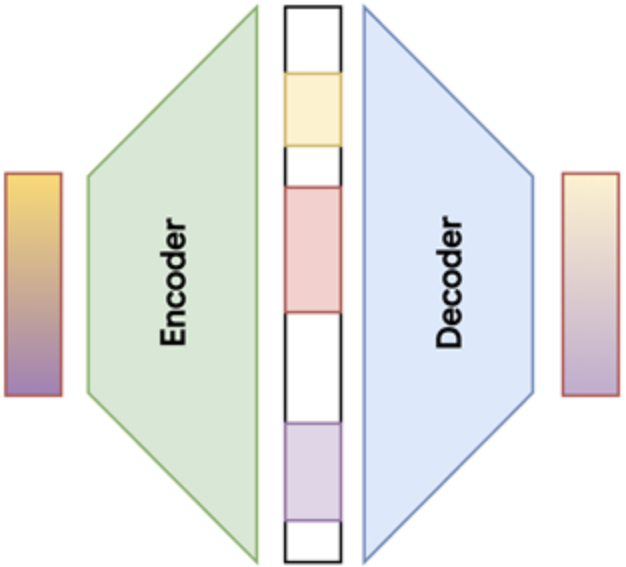
SAE Architecture

- Simple two-layer architecture with ReLU activation
- Encoder E_ϕ maps dense activations x to larger, sparse representations
  - Hidden representation is 4x-8x larger instead of shrinking it down like regular autoencoders
- Decoder D_θ reconstructs original activations, minimizing the following loss:

$$\min_{\theta,\phi} \|\mathbf{x} - \mathrm{D}_\theta(\mathrm{E}_\phi(\mathbf{x}))\|_2^2 - \lambda\|\mathrm{E}_\phi(\mathbf{x})\|_{L1}$$

- To save memory, you can share weights between the encoder and decoder

Sparsity penalty $\lambda||\bullet||_{L1}$ ensures hidden representation is sparse



$$\hat{\mathbf{x}} = \mathrm{Linear}(\mathrm{ReLU}(\mathrm{Linear}(\mathbf{x})))$$

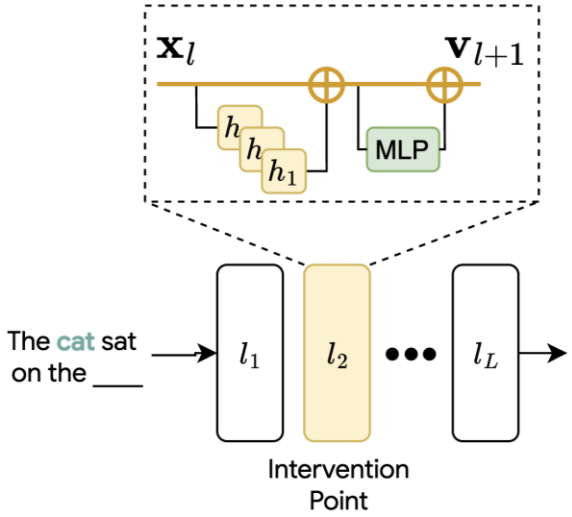
Training SAEs for LLMs

1. Collect activations from probing dataset

- Prepare diverse dataset of prompts containing the concept of interest (eg: cars)
- Run inference on LLM of interest (eg: GPT-2)
- Store pre-layer and post-layer activations at a chosen layer ***l*** (eg: HDF5/npy)

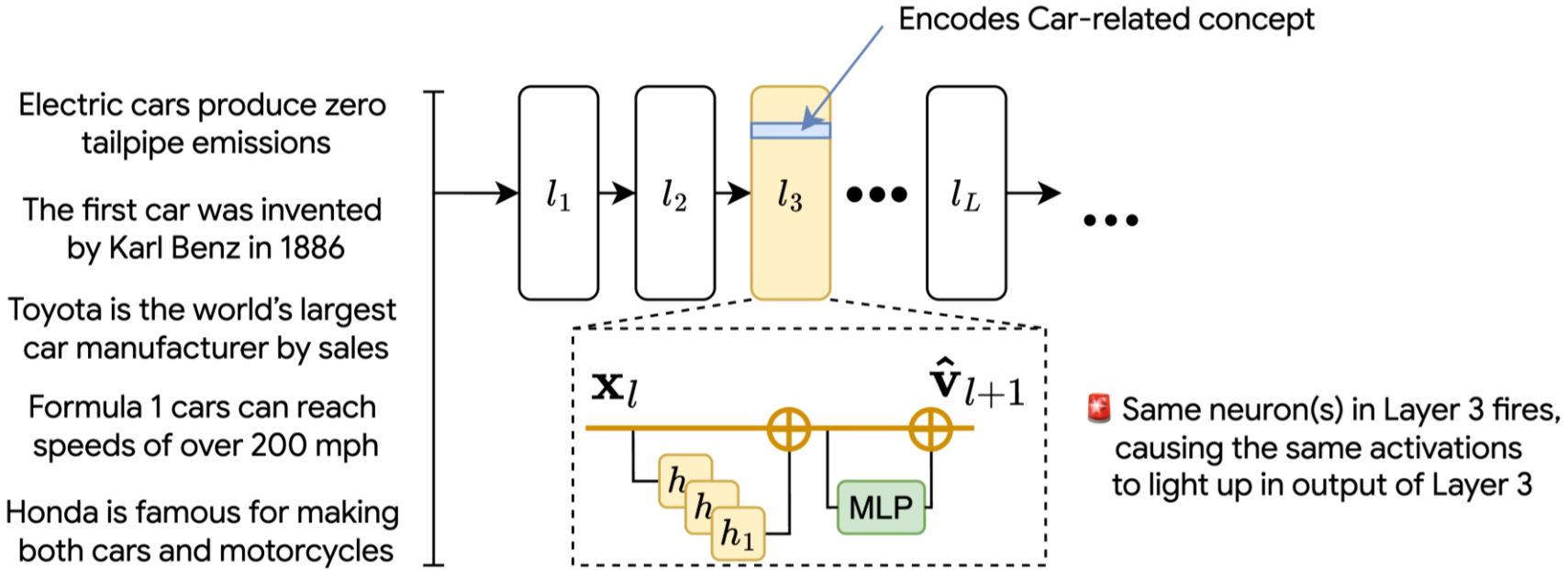
2. Train SAE via stochastic gradient descent methods

- Wide *expansion factor* allows for greater sparsity and feature separation
- Tune sparsity coefficient λ to balance reconstruction quality and interpretability



Training SAEs for LLMs

Assumption: prompts that contain similar concepts will fire the same neuron(s). This causes the same activations to light up (ie, have high values) in the layer outputs.



Interpreting SAE Features

Once you have a trained SAE for a specific layer,

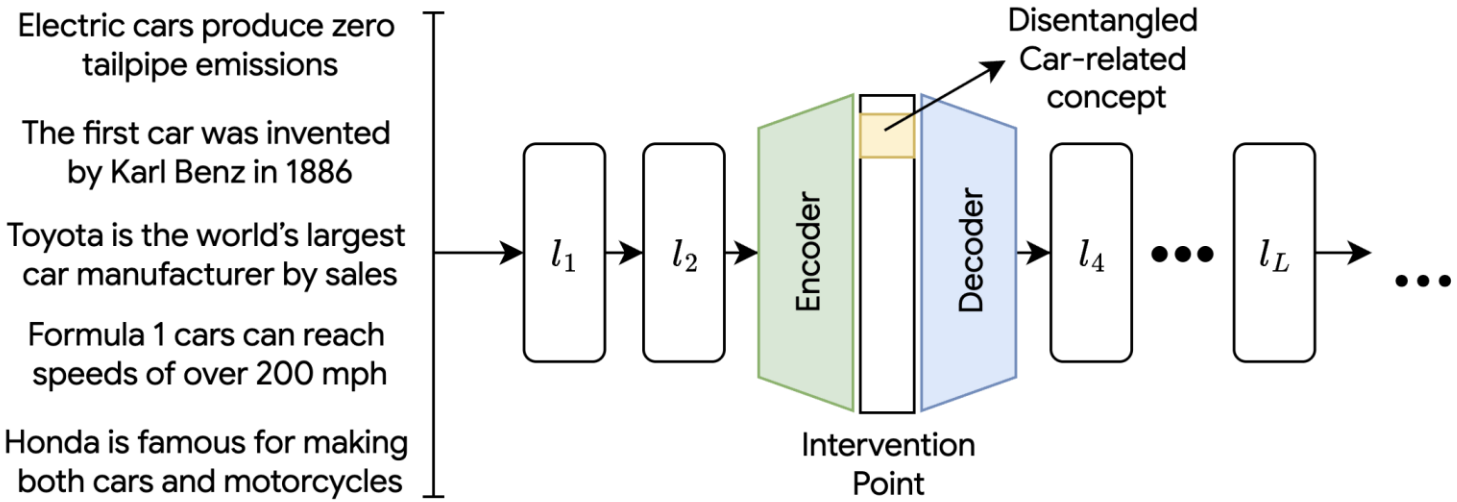
- For a new carefully chosen prompt, pass it through the LLM until the $l-1^{\text{th}}$ layer
- Pass these activations through the SAE instead of the l^{th} layer and finally, to the remaining layers as-is.

Based on the prompt's *content*, observe and identify which concepts are active by looking at **non-empty dimensions** along SAE's hidden representation

This gives us a rough indication of what concepts are encoded within an activation.

Interpreting SAE Features

The representations of such neurons will map to the same region of the disentangled, sparse vector in the SAE hidden representations regardless of prompts.



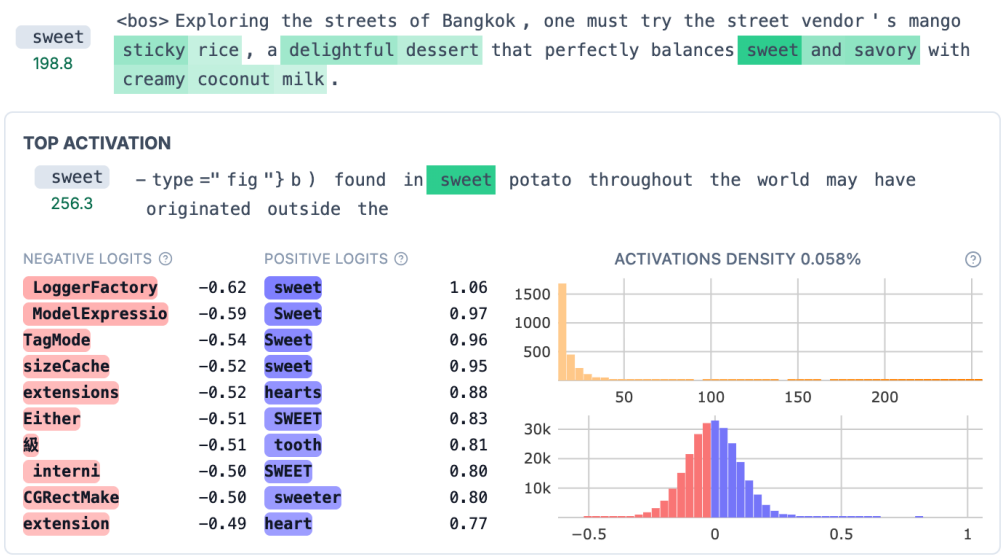
Gemma Scope

Consider an SAE trained on Layer 24 from Gemma-2.

"Exploring the streets of Bangkok, one must try the street vendor's mango sticky rice, a delightful dessert that perfectly balances sweet and savory with creamy coconut milk."

Feature #12895 activates on terms related to sweetness or sweet food items.

This neuron fires only for tokens corresponding to the "sweet" concept



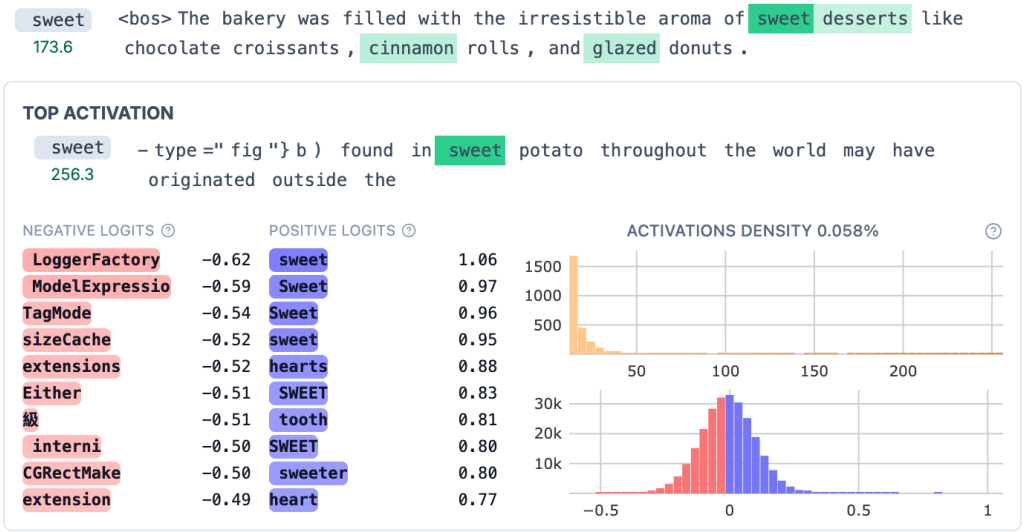
Gemma Scope

Consider an SAE trained on Layer 24 from Gemma-2.

"The bakery was filled with the irresistible aroma of sweet desserts like chocolate croissants, cinnamon rolls, and glazed donuts."

Feature #12895 activates on terms related to sweetness or sweet food items.

This neuron fires only for tokens corresponding to the "sweet" concept

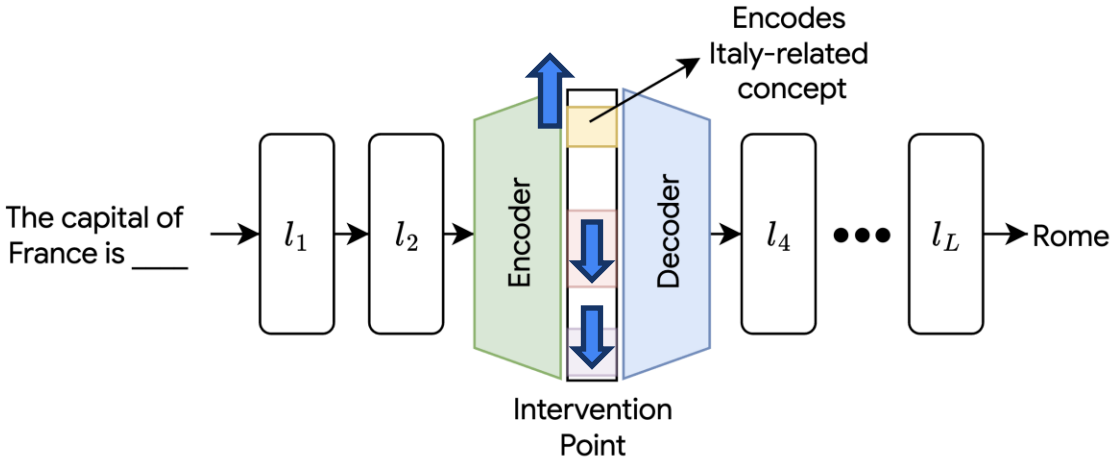


Intervention with SAEs

Causal intervention can be performed by clipping / blocking / scaling certain dimensions and reconstructing them using the decoder → also called **Steering**

You can control how much "essence" of the concept is in the reconstructed activations.

These dimensions can correspond to concepts we want to guard against (eg: profanities)



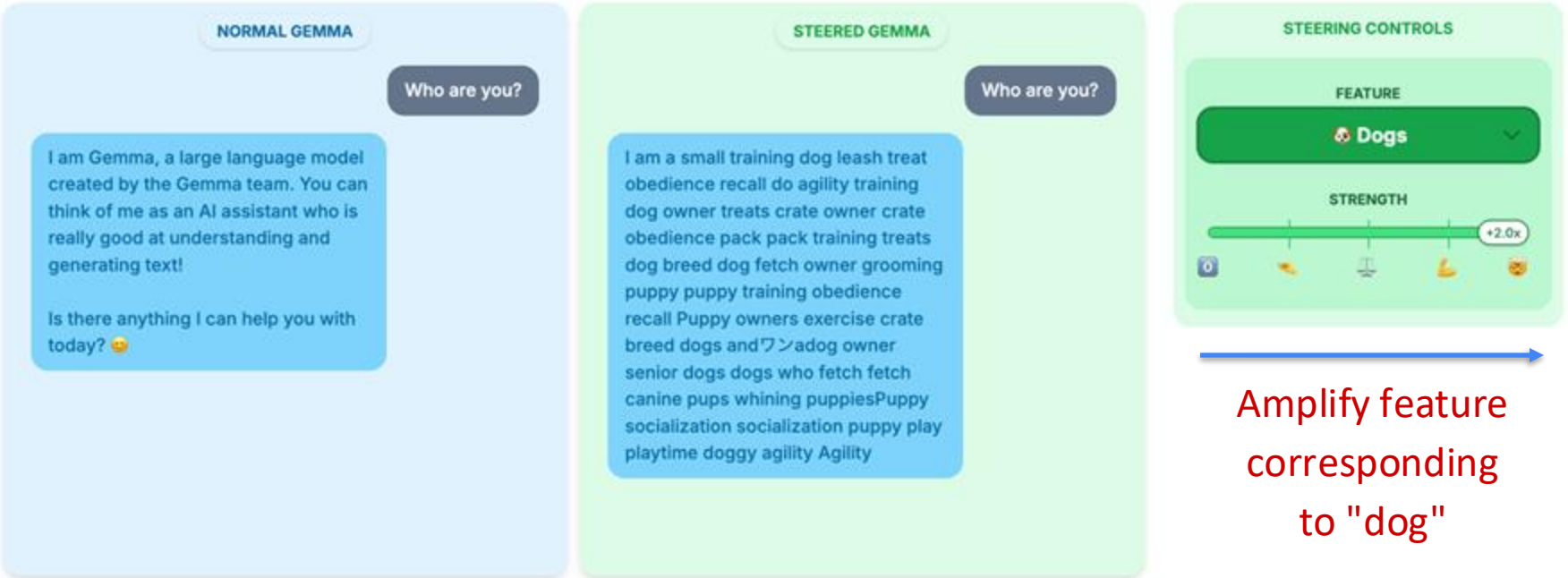
Replace the target layer with the trained SAE.

Identify the dimensions that correspond to the concept of interest.

Intervention with SAEs

Try it yourself:

neuronpedia.org/
gemma-scope#steer

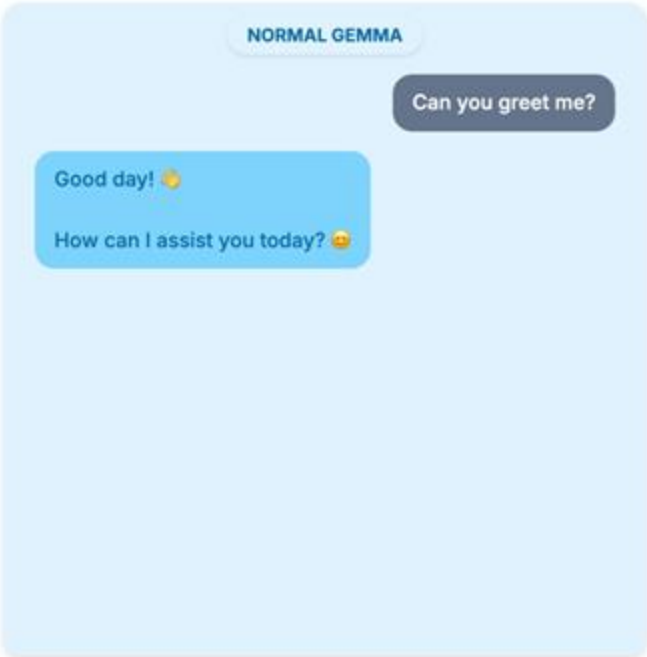


Steering Gemma 2 by amplifying the Dog concept

Intervention with SAEs

Try it yourself:

neuronpedia.org/
gemma-scope#steer



Amplify feature corresponding to "Refusal"

Steering Gemma 2 by amplifying the **Refusal** concept

Pros of SAEs

- **Interpretable**: Each dimension can capture human-interpretable concepts
- **Steering**: Allows us to intervene on LLMs directly, mitigating undesirable behaviors
- **Scalable**: automated way to discover thousands of features corresponding to unique concepts
- **Decomposable**: concepts can be combined via linear separability

Cons of SAEs

- **Erratic**: different SAE initializations can lead to different dictionaries being learned
- **Incompleteness**: no guarantee that features learned represent all possible concepts. The concepts also need to human-understandable, otherwise they risk being vague or not very disentangled.
- **Overhead**: an SAE needs to be trained for each layer separately

Negative Results for Sparse Autoencoders On Downstream Tasks and Deprioritising SAE Research (Mechanistic Interpretability Team Progress Update)

DeepMind Safety Research · Follow · 9 min read · Mar 26, 2025

*"... **we are deprioritising fundamental SAE research** for the moment and exploring other directions, though SAEs will remain a tool in our toolkit"* ~ DeepMind

- Introduction to Mechanistic Interpretability
- Early Interpretability Efforts and Semantics
- Sparse Autoencoders
- **Circuit Theory and ACDC**

Can we identify which parts of the LLM contribute most to their predictions for a specific task?

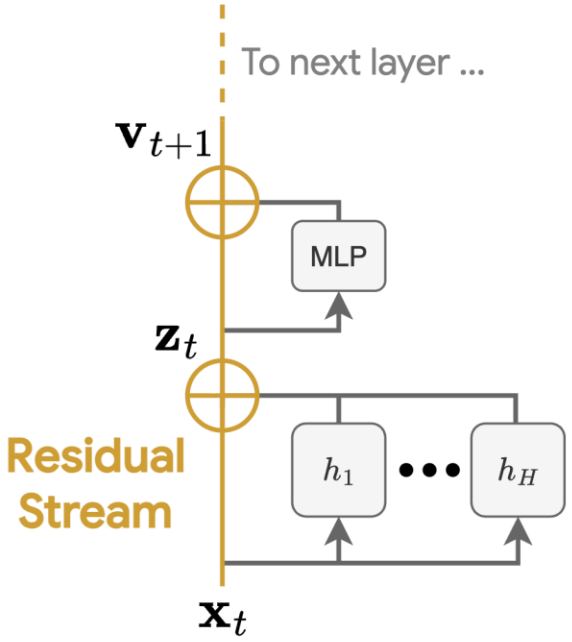
If yes, it'd tell us where we can focus our interpretability and intervention efforts.

Circuit Theory for LLMs

- In LLMs, components can communicate across layers, relaying concepts, context, and semantics encoded within embeddings
- For a given prompt, certain parts of the LLM are more important than others. These critical parts are called **Circuits**
- You can corrupt or remove the components *outside* the circuit and the LLM will still give you the same answer.
- Circuit discovery lies between NP-Complete and NP-Hard
  [Adolfi et al. (2024); ICLR'25 Spotlight]

LLMs as Computational Graphs

- *Computational Graphs* are directed graphs through which data moves from one end to the other
- Nodes represent functions/operations and edges represent flow of information between them
- In LLMs, components like the MLP and Attention Heads are the main operations. Make them nodes
- Connect components within and between layers using edges for token embeddings to flow



A Transformer layer is a computational graph!

Finding LLM Circuits using ACDC

Automated **C**ircuit **D**is**C**overy (ACDC) [Conmy et al., 2023] is a well-known circuit discovery algorithm that works in a recursive manner.

Rough Intuition:

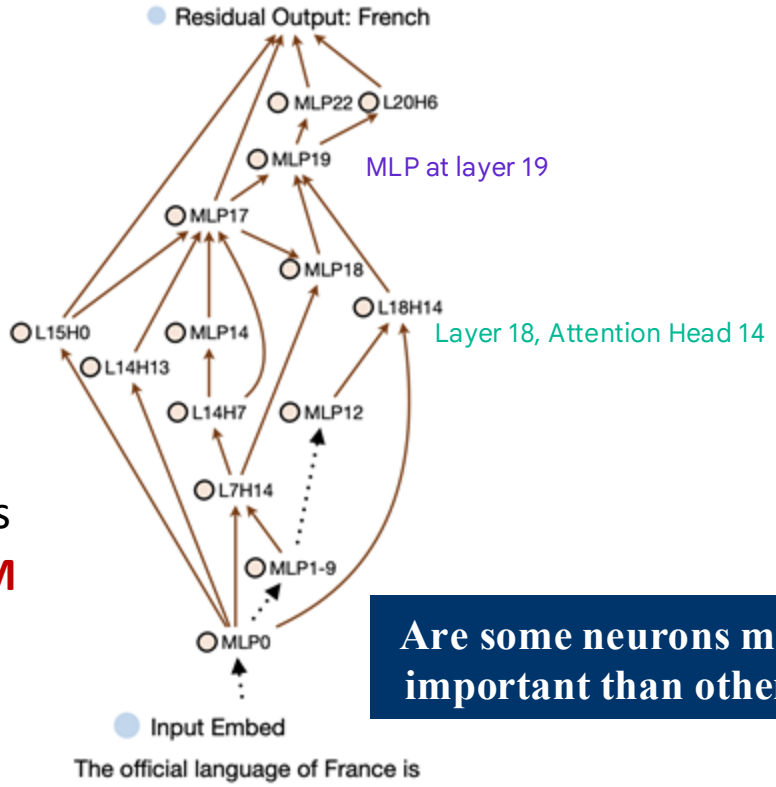
1. Treat the LLM as a giant *computational graph* with the prompt as the input.
2. Suppose we consider every component (MLP/Attention Head) as a "node".
3. Prune the computation graph to find subset of nodes that allow the LLM to still answer the prompt correctly. Remove all other unimportant nodes.
4. Remaining nodes form the Circuit subgraph for this prompt.

Circuit Theory for LLMs

In LLMs, circuits are more intricate because of **polysemantic neurons** across the architecture.

Consider the GPT-2 circuit on the left.
Each node represents either an MLP or Attention Head in a specific layer.

If we remove these specific components and pass embeddings through the rest of the LLM, **the LLM forgets what the language spoken in France is!**

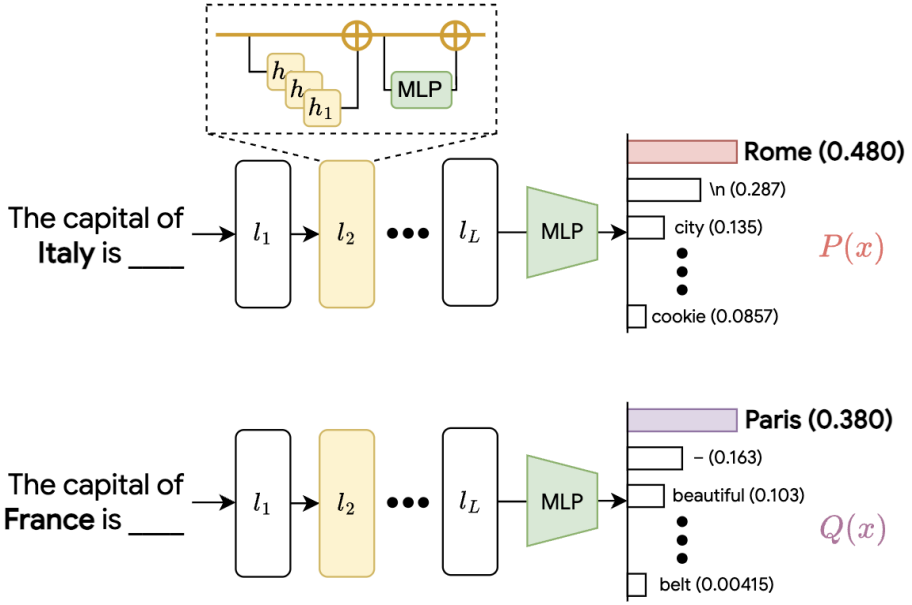


Are some neurons more important than others?

Recap: Logit Distributions and KL Divergence

- Autoregressive LLMs output **logits**, which is a **probability distribution across the entire vocabulary**
- Changes in logits directly correspond to changes in next token prediction/sampling
- KL Divergence measures ***by how much*** the logits change when we modify the model

How can we study how much each component in the LLM contributes to the final token prediction?

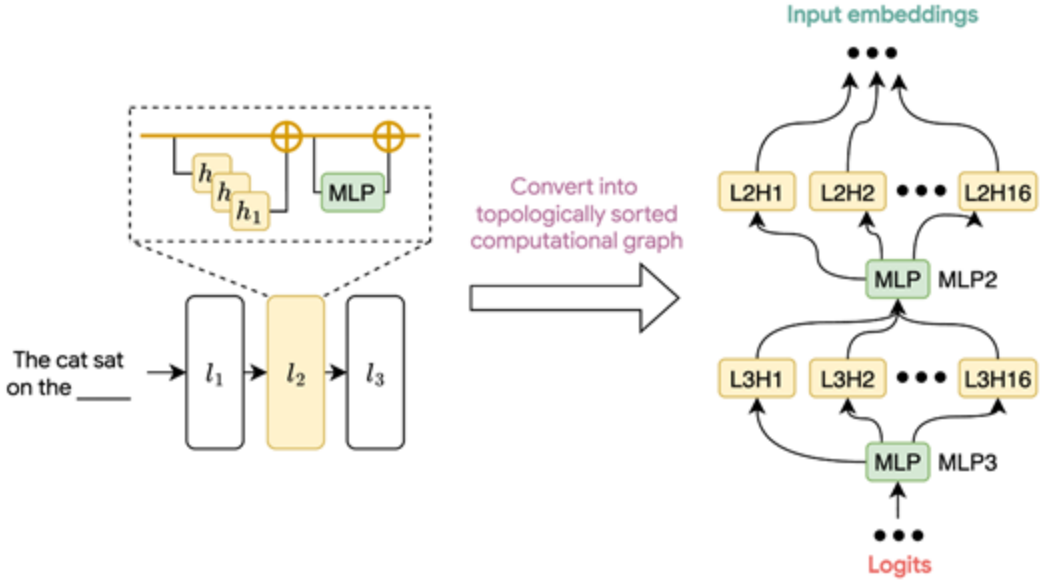


$$D_{KL}(P|Q) = \sum_x P(x) \log \frac{P(x)}{Q(x)}$$

Automated Circuit DisCovery (ACDC)

1. Start with a *complete* computational graph *G*

Every attention head node (**LXHY** for Layer X Head Y) and MLP node (**MLPX** for layer X MLP) is connected to every other succeeding node to form a *Directed Acyclic Graph*. ACDC prunes edges from this DAG in a recursive manner.



This computational graph is ***G***. Create a copy ***H*** and topologically sort it.

This allows us to study how logit changes happen due to changes in intermediate activations by comparing incrementally edited versions of ***H***.

Automated Circuit DisCovery (ACDC)

2. Create a probing dataset

- A collection of diverse prompts that study a specific behavior.
  Eg: studying location-specific factual recall like capitals/geography

3. Set a threshold τ

- This will decide how important an edge between two component nodes is to the circuit
- If an edge's importance is below this threshold, it is removed from the circuit

Datasets for Circuit Discovery

- We care about specific LLM behaviors where prompts contain certain concepts of interest
- Datasets used for this are called **Indirect Object Identification** (IOI) datasets

Friends Juana and Kristi found a mango at the bar. Kristi gave it to _____.
Then, Yvette and Angie were working at the mountain. Yvette decided to give a banana to_____
After Doris and Marsha went to the mountain, Marsha gave a strawberry to _____
While Bernadette and Harriet were commuting to the desert, Bernadette gave a watermelon to _____
Afterwards, Ginger and Bernadette went to the library. Bernadette gave a mango to _____

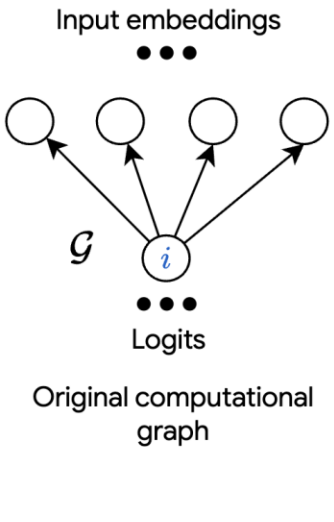
- For each _____, we expect the LLM to predict the correct next token autoregressively

Automated Circuit DisCovery (ACDC)

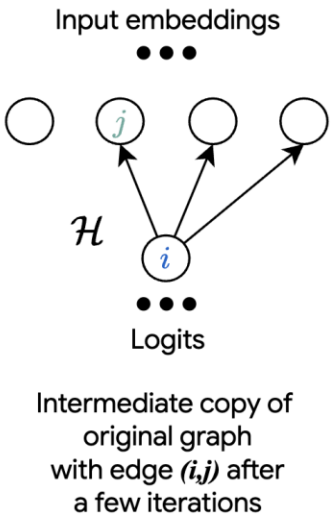
4. Iteratively prune edges by comparing copies of the computation graph to the original graph

For each node i, for each of its parent nodes j, create a new copy of H called H_{new} but without edge (i,j).

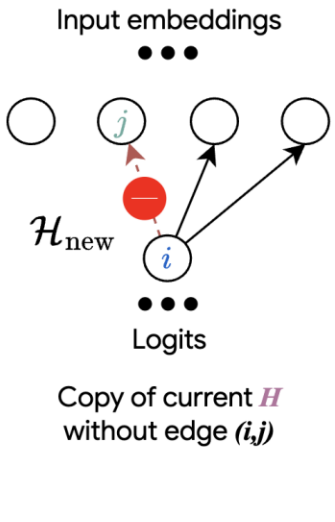
Perform a forward pass for H and H_{new} to get logits for modified architecture.



Original computational graph



Intermediate copy of original graph with edge *(i,j)* after a few iterations



Copy of current H without edge *(i,j)*

Greedily check:

$$D_{KL}(G\|H_{\text{new}}) - D_{KL}(G\|H) < \tau$$

Intuition Behind Edge Pruning Logic

For a given edge (*i*,*j*), we check if the difference between KL divergences is above a threshold:

1. Compare the original graph G with the current iteration's copy H and modified version H_{new}
2. Check if the current edge is important enough to significantly change the logits
3. Retain edge if logits change significantly

$$D_{KL}(G||H_{\text{new}}) - D_{KL}(G||H) < \tau$$

Edge (i, j) is NOT important.
Logits do not change significantly.
Remove edge.

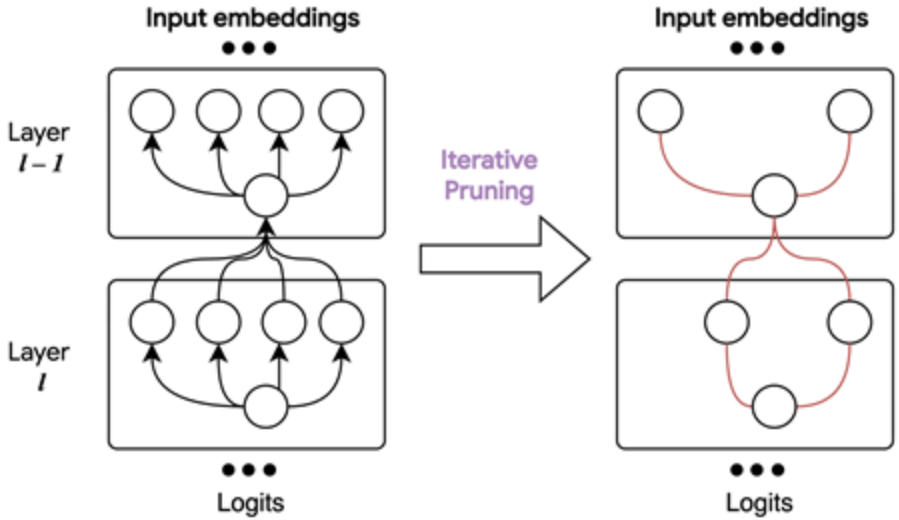
$$D_{KL}(G||H_{\text{new}}) - D_{KL}(G||H) \geq \tau$$

Edge (i, j) is important.
Logits change significantly after modification.
Retain edge.

Automated Circuit DisCovery (ACDC)

5. Return iteratively pruned graph H

Repeatedly checking if an edge changes the logits reveals its importance. It means the adjacent components share important information relevant for the task.



The remaining set of nodes and edges correspond to the key components used by the LLM to perform the task or emulate the behavior in question.

Automated Circuit DisCovery (ACDC)

Algorithm 1: The ACDC algorithm.

Data: Computational graph G, dataset $(x_i)_{i=1}^n$, corrupted datapoints $(x'_i)_{i=1}^n$ and threshold $\tau > 0$.

Result: Subgraph $H \subseteq G$.

```
H ← G                                   // Initialize H to the full computational graph
H ← H.reverse_topological_sort()        // Sort H so output first
for v ∈ H do
    for w parent of v do
        H_new ← H \ {w → v}             // Temporarily remove candidate edge
        if D_KL(G||H_new) − D_KL(G||H) < τ then
            H ← H_new                   // Edge is unimportant, remove permanently
        end
    end
end
return H
```

A. Iterate through edges

B. Modify model

C. Pruning condition

Conmy et al. (2023)

Finding LLM Circuits using ACDC

Eg: A circuit from GPT-2 Small corresponding to the prompt,

When John and Mary went to the store, Mary gave a bottle of milk to _____

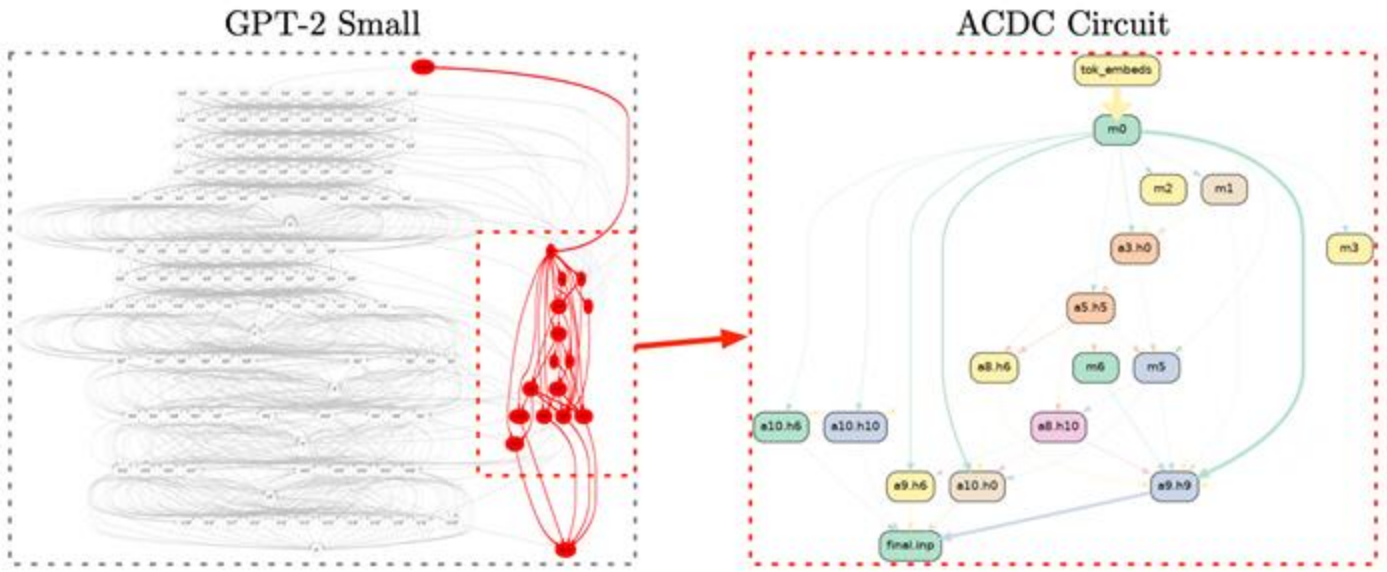


Figure and prompt from Wang et al. (2023) and Conmy et al. (2023)

Pros of Circuit Theory

- **Extracts causal relationships**: provides precise component-wise breakdown of how complex behaviors emerge.
- **Enables targeted intervention**: circuits reveal *exact* locations to make edits to stop undesirable tendencies or amplify specific behaviors (eg: for safety/alignment purposes).

Cons of Circuit Theory

- **Very slow and tedious**: finding circuits is a very arduous endeavour that takes a lot of trial and error, with careful curation of probing datasets.
- **Quality depends on granularity**: fine-grained nodes (at the neuron level) are slower to excavate while coarse-grained nodes (layer/component level) are faster to discover.
- **Not scalable**: circuits discovered may not be transferable between different LLMs. Intervention techniques from one LLM may apply to other LLM families.

Future of Mechanistic Interpretability

- LLMs are complex high-dimensional objects that must be studied very carefully
- There are any correlated behaviors that may be spurious in nature
- The jury is still out on the reliability of Mechanistic Interpretability methods
- Currently, Mechanistic Interpretability efforts are prone to reliability issues
  - Eg: changing the random seed changes the features learned by SAEs [Paulo & Belrose (2025)]
- Concurrent works study the evolution of token geometry through the layers
  - Provides several insights into the mechanisms learned by RoPE, LayerNorm, Softmax, etc.
  - This offers a different (geometric) perspective on why LLMs behave the way they do