MESA – Model Editing with Sparse AutoEncoders

Noam Diamant
Bar Ilan University

Introduction

- **Goal:** Optimize *model editing* — improve the precision of existing model editing techniques.
- Approach: Apply an *interpretability* method called **Sparse Autoencoders (SAEs)** to better understand and leverage LLM activations.
- Briefly explain:
  - Interpretability of LLM internal activations.
  - How SAEs can disentangle meaningful features.
- Present our **workflow**, including **interfaces** and the **tech stack**.
- Review **improvements** made during the process.
- Conclude with **key learnings** from the project.

Motivation

Model Editing is important and challenging:

- Large autoregressive language models can recall many common facts.
- they may recall obsolete information if not updated periodically.
- Maintaining fresh, customizable information is valuable across applications.
- **Edits should be precise and strictly limited to the relevant subject matter.**

Model Editing Methods

- **Focus**: explore methods that modify the model's parameters directly.
- **Approach**: Locate key parameters, then edit them by adding a perturbation Δ.
- **Goal**: Minimize error on the new (to-be-updated) knowledge.
- **Constraint**: Limit error to preserve existing knowledge.
- **Methods**: ROME, MEMIT, ALPHAEDIT (SOTA).

Challenge

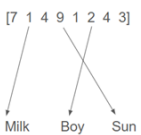
Goal: Improve model-editing performance across *efficacy*, *generalization*, and *specificity* simultaneously.

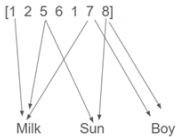
- **Efficacy**
- **Generalization**
- **Specificity**
- For each:
  - **Score (S)** – Fraction of successful cases.
  - **Magnitude (M)** – Average probability difference between target and original fact.
- **Overall tradeoff measure:** Harmonic mean of S across all three dimensions.

The Challenge: Achieve simultaneous improvement in those metrics for existing model editing techniques through precise editing.

polysemanticity and monosemanticity

- **Monosemantic Feature**: A single feature corresponds clearly to one interpretable concept.
- **Polysemantic Feature**: A feature activates for multiple, unrelated concepts.
- This entanglement makes intervention harder — changing one feature may affect multiple meanings.
- Disentangling features is crucial for precise and targeted model editing.





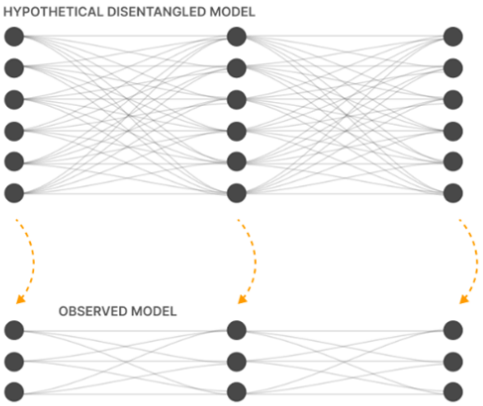
Superposition Hypothesis

Hypothesis – Superposition

- Neural activations may combine multiple abstract features.
- Each feature represents multiple overlapping concepts.
- A larger, disentangled model could separate these meanings.

We hope to successfully...

- Decompose activations into disentangled, interpretable features.
- Enable clearer understanding and manipulation of internal states.



Sparse Autoencoder (SAE)

- We use a sparse autoencoder to learn disentangled dictionary features.
- Each input activation vector x is encoded into a sparse representation c.
- The reconstruction $\hat{x}$ is obtained via a sparse linear combination of dictionary features.

$$\hat{x} = M^\top \operatorname{ReLU}(Mx + b) = M^\top \mathrm{c} = \sum_{i=0}^{d_{\text{hid}}-1} c_i f_i$$

- The model is trained to minimize reconstruction error while encouraging sparsity.

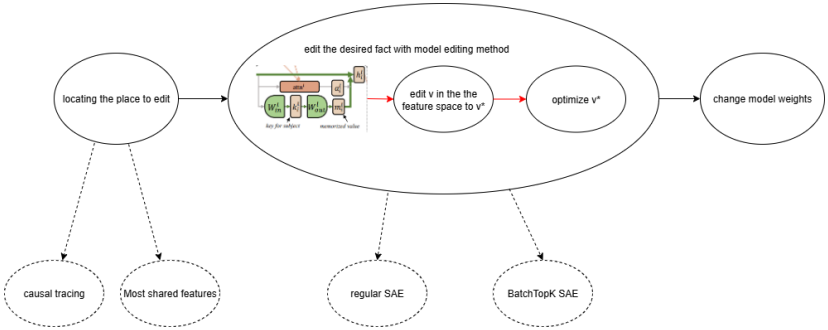
$$L(x) = \|x - \hat{x}\|_2^2 + \alpha \|c\|_1$$

SAE Successfully Disentangles Features

- We evaluated the SAE by collecting sentences related to each concept.
- For each concept, we recorded the most activated features across the sentences.
- Many top features were unique to each concept, indicating successful disentanglement.

Basketball	Football	Baseball	General
4703 (9.69)	4703 (8.04)	4703 (7.89)	19 (0.0007)
6102 (5.48)	8796 (3.37)	6102 (2.55)	0 (0.0000)
0 (0.00)	0 (0.00)	8796 (2.54)	1 (0.0000)
1 (0.00)	1 (0.00)	0 (0.0000)	2 (0.0000)
2 (0.00)	2 (0.00)	1 (0.0000)	3 (0.0000)
3 (0.00)	3 (0.00)	2 (0.0000)	4 (0.0000)
4 (0.00)	4 (0.00)	3 (0.0000)	5 (0.0000)
5 (0.00)	5 (0.00)	4 (0.0000)	6 (0.0000)
6 (0.00)	6 (0.00)	5 (0.0000)	7 (0.0000)
7 (0.00)	7 (0.00)	6 (0.0000)	8 (0.0000)

MESA schema and interfaces



- The main interface occurs between the activation vector v obtained from the MLP layer and the process of returning a more precise version of it.
- This refined vector is integrated directly into the weights optimization process, improving editing precision.

Quick demo: v^* Modification Process

1. Obtain both δ and v in the activation space (e.g., for Qwen2.5B-0.5B: dimension 896)
2. Encode them into the feature space (dictionary size: 65,536)
3. Sum the encoded vectors in the feature space
4. Decode the result back to the activation space
5. Repeat the process while optimizing v
6. Output the final optimized vector v^*

ROME vs. ROME + MESA - Initial Results

Results on the CF dataset (Qwen2.5B-0.5B)

	ES	EM	PS	PM	NS	NM	S
ROME	99.9	97.94	98.8	59.76	68.04	1.82	86.15
ROME + MESA	84.0	62.24	62.5	18.33	78.4	6.89	73.78

- E - **Efficacy**
- N - **Specificity**
- P - **Paraphrase (Generalization)**
- S - **Score**

Important improvements along the way - part 1

The importance of selecting the right SAE

- Initially, I used the **standard SAE** architecture for encoding activations.
- After reviewing early results, I realized that SAE choice could have a large impact on model editing precision.
- I conducted a **literature review** to explore alternative SAE architectures.
- Chose **BatchTopK SAE** due to:
  - Simplicity in implementation.
  - Enhanced performance reported in **SAEBench** benchmarks.
- **This change has shown improvements in our evaluation metrics.**

SAEBench: A Comprehensive Benchmark for Sparse Autoencoders in Language Model Interpretability

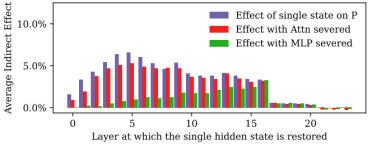
---

Adam Karvonen [*1] Can Rager [*1] Johnny Lin [*2] Curt Tigges [*2] Joseph Bloom [*2] David Chanin [3]
Yeu-Tong Lau [1] Eoin Farrell [1] Callum McDougall Kola Ayonrinde Demian Till [4] Matthew Wearden [5]
Arthur Conmy Samuel Marks [6] Neel Nanda

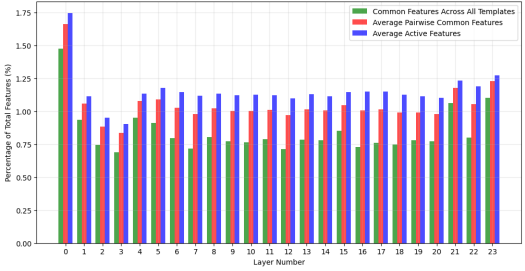
Important improvements along the way - part 2

Choosing the right place to edit the v vector

- Initially used **Causal Tracing** to locate the optimal layer/submodule for v edits.
- Later switched to finding the layer with the **most common features** across editing templates.
- This led to a different layer choice and **improved evaluation metrics**.



Causal Tracing



Most Common Features

ROME/MEMIT vs. + MESA - Improved Results

Results on the CF dataset (Qwen2.5B-0.5B)

ROME vs. ROME + MESA

	ES	EM	PS	PM	NS	NM	S
ROME	99.9	97.94	98.8	59.76	68.04	1.82	86.15
ROME + MESA	99.85	97.39	98.9	58.96	74.38	3.29	89.36

MEMIT vs. MEMIT + MESA

	S	ES	PS	NS
MEMIT	85.8	98.9	88.6	73.7
MEMIT + MESA	87.96	99.01	89.11	78.07

- E - **Efficacy**, N - **Specificity**, P - **Paraphrase (Generalization)**, S- **Score**
- Please note that each of the tables represents a different experiment.
- To be continued!

Lessons Learnt

- **Choosing the right place to work on is crucial** The location where you intervene in the model can significantly influence the outcome.
- **Methods for finding the right place matter** Techniques such as causal tracing, feature analysis, or other localization strategies should be selected carefully.
- **Adapt the location to the method** Different editing or fine-tuning approaches may require different intervention points; the choice should be method-specific.
- **Selecting the right tools is equally important** Standard tools may work, but they are not always optimal for your task.
- **Review and experiment** Explore existing tools, compare them, and run experiments to determine which best suits your needs.

Tech Stack Used

- **PyTorch** and official GitHub repositories of existing editing methods Used implementations from research papers to run experiments and evaluate results.
- **TransformerLens** and **PyTorch Hooks** Manipulated LLM activations; TransformerLens provides an open-source, convenient interface for this purpose.
- **Dictionary Learning** and **SAE Lens** (open source) Trained and analyzed Sparse Autoencoders (SAEs) for interpretability studies.
- **Hugging Face Transformers** Used for exploring, loading, and interacting with pre-trained language models.

Q&A – Thank You!