Decomposition of Small Transformer Models

Casper L. Christensen*
Independent

Logan Riggs Smith
Independent

Abstract

Recent work in mechanistic interpretability has shown that decomposing models in parameter space may yield clean handles for analysis and intervention. Previous methods have demonstrated successful applications on a wide range of toy models, but the gap to "real models" has not yet been bridged. In this work, we extend Stochastic Parameter Decomposition (SPD) to Transformer models, proposing an updated causal importance function suited for sequential data and a new loss function. We demonstrate that SPD can successfully decompose a toy induction-head model and recover the expected 2-step circuit. We also show that applying SPD to GPT-2-small can successfully locate subcomponents corresponding to interpretable concepts like "golf" and "basketball". These results take the first step in the direction of extending SPD to modern models, and show that we can use the method to surface interpretable parameter-space mechanisms.

1 Introduction

Much of mechanistic interpretability work can be characterised as belonging to one of two waves [16]. In the first, researchers attempted to understand models by dissecting and studying neurons individually, but this proved infeasible due to polysemanticity [12] and superposition [9]. The second wave shifted into *activation space*, where sparse dictionary learning (SDL) [15] uncovered thousands of highly-interpretable concepts [7]. However, SDL methods struggle with feature geometry and anomalies such as feature absorption and splitting [4], while offering no clear definition of what features actually are. Newer methods operate directly in parameter-space and provide a complementary line of work: Braun et al. [2] decompose weights into “mechanism-space“ vectors, and Bushnaq et al. [3] extend this with Stochastic Parameter Decomposition (SPD), which represents weights as sparsely-activating rank-1 matrices. In this work, we extend this method to Transformer models. Our contributions are: **a)** a causal importance function for sequential data, **b)** decomposing a toy model of induction heads, **c)** a loss function that enables faithful decomposition over few examples, and **d)** decomposing GPT2-small and locating interpretable subcomponents.

2 Motivation

Mechanistic interpretability has mostly progressed through methods that operate on activations. Sparse dictionary learning, causal tracing, and circuit discovery let us ask, for a given input, “what fired and what did it do?” These approaches are powerful, but they do not yet give us a canonical way to factor the model itself into a small set of reusable mechanisms: features can split, reconstruction can be lossy, and the geometry of the learned features is often left unexplained. Recent work on parameter-space decomposition argues that if what gradient descent is really doing is writing mechanisms directly into the weights, then it is natural to look for mechanisms in that same space.

*Casper L. Christensen was funded by the Pivotal research fellowship throughout this project.

3 Parameter Decomposition

Attribution-based Parameter Decomposition (APD) makes this idea concrete: it flattens a trained network into parameter "components," trains them to sum back to the original network (faithfulness), and then uses an attribution-and-top-k procedure to make only a few components active on each example (minimality + simplicity). Stochastic Parameter Decomposition (SPD) keeps the same goal but makes it more practical by working with rank-1 subcomponents and by learning a causal-importance function instead of hard-coding top-k. Concurrently, Chrisman et al. [5] propose Local Loss Landscape Decomposition (L3D), which also learns sparsely active parameter-space directions, but does so by reconstructing per-sample gradients of a divergence, rather than the weights themselves. Their sparsity is induced by a top-k reconstruction in gradient space, and their subnetworks are allowed to be higher-rank via Tucker decompositions. We directly extend the SPD line of work.

4 Method

SPD aims to provide interpretability by learning an "auxiliary model" that matches the original model's computations. Strictly speaking, this means we want a decomposition that is **faithful** to the original model, and for a given input x, **reconstructs** the original model's output, and is **minimal**.

SPD decomposes a model's weights (components) $W^l \in \mathbb{R}^{n\times m}$ into subcomponents $(W_1^l, ..., W_C^l)$ where C is a hyperparameter and each $W_c^l = \vec{U}_c^l \otimes \vec{V}_c^l,\ U_c^l, V_c^l \in \mathbb{R}^d$, such that they are rank 1. For any given input x, we can assemble a weight matrix $W'^l = \sum_{c=1}^{C} \alpha \cdot W_c^l \quad \alpha \in [0,1]$, which we can use instead of W^l. α then determines how "active" a subcomponent is on a given input. In practice, $\alpha \sim \mathcal{U}(g_c^l(x), 1)$, where $g_c^l(x)$ is the causal importance of subcomponent c in layer l on x. We define r^s as the mask of all α-samples across all layers and $r^{l,(s)}$ as the mask of α-samples for layer l.

Faithfulness is enforced by requiring that all subcomponents sum to their original component. This means that the original model is the case where $\alpha = 1$ everywhere. This ensures that we can always recover the real model from its decomposition, and restricts the decomposition's ability to learn some alternative factorisation of the original model's parameters. Let the model and its decomposition be respectively f and f', we then obtain a minimal decomposition by ensuring that $f(x) \approx f'(x)$ while assigning $g_c^l(x) > 0$ to as few subcomponents as possible.

This naturally leads to the losses from Bushnaq et al [3], and we train on a coefficient-weighted sum of them. See A.4 for a description of a training step.

$$\mathcal{L}_{faithfulness} = \tfrac{1}{N} \sum_{l=1}^{L} \sum_{i,j} \left(W_{i,j}^l - \sum_{c=1}^{C} U_{i,c}^l V_{c,j}^l \right)^2, \quad \mathcal{L}_{minimality} = \sum_{l=1}^{L} \sum_{c=1}^{C} |g_c^l(x)|^p \tag{1}$$

$$\mathcal{L}_{stochastic_recon} = \frac{1}{S} \sum_{s=1}^{S} D_{KL}(f(x|W'(x, r^{(s)})), f(x|W)) \tag{2}$$

$$\mathcal{L}_{stochastic_recon_layerwise} = \frac{1}{LS} \sum_{l=1}^{L} \sum_{s=1}^{S} D_{KL}\Big(f(x \mid W^1, \ldots, W^l(x, r^{l,(s)}), \ldots, W^L),\ f(x \mid W)\Big) \tag{3}$$

The stochastic sampling serves 2 purposes: it creates a mechanism by which subcomponents can still be turned on (and thus provide gradient signal) even if their causal importance became 0, and - because reconstruction should happen with as few subcomponents as possible - it sets the causal importance as the lower bound for how active a subcomponent must be in order to not significantly change the output. The layer-wise loss ensures that each decomposed weight must be substitutable in the original model without changing the output, further restricting the decomposition's ability to derive an equivalent model. In addition, we train on $\mathcal{L}_{recon}$, which is equivalent to $\mathcal{L}_{stochastic_recon}$ with all mask-values set to exactly their causal importance (no sampling). The stochastic loss gives noisier signals, where the deterministic version tightens the lower bound and helps convergence.

4.1 Causal Importances

In Bushnaq et al [3], causal importance for a subcomponent is defined as how *ablatable* it is. We follow this same operationalisation, noting that ablations are not truly causal, because they are subject to interactions within the model [6]. Within this definition, a causally important subcomponent should have high impact on the output and is therefore not ablatable. The choice of stochastic losses ensures that this is enforced, as non-important subcomponents may be arbitrarily turned on, whereas important ones are guaranteed to stay on. Causal importances are learned with an independent γ-function (an MLP) per subcomponent, which takes as input either the inner-activation $x \times \vec{U}$ (a scalar) or x. However, this is suboptimal for models operating on sequences. Consider the string "Sat by the river bank, the bank manager" and "manager" as the query-token. Independent γ-MLPs would be forced to assign the same causal importance score to the 2 instances of bank, even though the latter is likely to be more important for the output [2]. This is especially true for the OV-circuit, where 2 value vectors with the same representation may be unequally attended to. Motivated by this, we apply attention with a minimal attention network (1 head, 1 layer, QKV-only) across the tokens prior to computing g. We use learned, relative positional encodings in the attention-network for expressivity and learned absolute positional encodings on the value-vectors, such that the downstream γ-MLP *also* has positional awareness. This means that for subcomponents $(W_1^l, ..., W_C^l)$ and positions $(1, ..., n)$, we decide causal importance $g_{c,n}^l$ via:

$$g_{c,n}^l = \sigma_H\big(\gamma_c^l(\bar{x}_n)\big), \quad \text{where } \bar{x_n} = \left(\text{softmax}\left(\frac{q_n K^\top + r_n}{\sqrt{d_k}}\right) V\right) \oplus x_n \tag{4}$$

where V and x_n have absolute, learned positional encodings, r_n is the row of learned, relative positional biases, γ_c^l are independent MLPs per subcomponent as in the previous literature, and σ_H is the leaky-hard sigmoid function. While this scales worse than the original implementation, we implement this efficiently with flex-attention [8]. See the appendix for a speed comparison.

5 Experiments

5.1 Induction Heads

Figure 1: The type of sequence we train on for the induction-head task.

Figure 1 shows an example of the data generated for the task. Formally, we sample $n-2$ tokens from a vocabulary V, and we then insert 2 s-tokens; one randomly and one at the end of the sequence. The task is to predict the token following the first s-token (we call this token the m-token). The s-tokens are unique such that they serve as an indicator for induction. Crucially, this task is only solvable by first moving information from s to m, and then m to the final s, which forces the model to form an induction-head [13]. We train a 2 layer, 1-head-per-layer attention-only Transformer [17] to on sequences of this form, such that we can later decompose it. See Table 3 for our hyperparameters. We consider loss only on the final token both when training the model and when decomposing it. We analyse the recovered circuit and its consistency with the original model.

5.2 GPT2-Small

One hope of methods like SPD is the ability to elicit latent knowledge. Decomposing over data points naturally creates a correspondence between the amount of data we decompose over and the proportion of the model we can "explain". However, in some cases, we may want to concretely investigate the model's behaviour on only a small amount of data points. To investigate this, we create a dataset of 2 sentences that GPT-2 correctly predicts the last token for: "Kobe Bryant plays the sport of basketball"

[2]Note that this does not necessarily apply to the QK-circuit in Transformers with absolute positional encodings, as the representations will differ slightly. For RoPE encodings that are not applied until *after* the projections, this will matter as the γ-MLPs will be positionally-unaware.

and "Tiger Woods plays the sport of golf", and use just these examples to decompose *GPT2-small* [14]. We analyse the subcomponents and suppress specific subcomponents via orthogonal projection $W \leftarrow W - \sum_{k=1}^{r} \left(\hat{u}_k^\top W \, \hat{v}_k\right) \hat{u}_k \, \hat{v}_k^\top, \quad \hat{u}_k = \frac{u_k}{\|u_k\|}, \quad \hat{v}_k = \frac{v_k}{\|v_k\|}$. We also check transfer to a 100-example synthetic fact set (See: A.3.4) to show the edit is not purely single-example overfitting. When a decomposition covers only a small proportion of the original model, there is higher risk of the decomposition instead developing a new circuit through optimising for the minimality loss. To prevent this, we introduce randomized partial versions of the reconstruction losses that evaluate only a subset of layers at a time. This discourages the decomposition from having computation in locations that do not match the original model.

$$\mathcal{L}_{recon_partial} = D_{KL}\big(f(x \mid W^1, \ldots, W^{l \in \mathcal{M}}(x, g^l(x)), \ldots, W^L),\ f(x \mid W)\big), \tag{5}$$

$$\mathcal{L}_{stochastic_recon_partial} = \frac{1}{S} \sum_{s=1}^{S} D_{KL}\Big(f(x \mid W^1, \ldots, W^{l \in \mathcal{M}^{(s)}}(x, r^{l,(s)}), \ldots, W^L),\ f(x \mid W)\Big), \tag{6}$$

where $\mathcal{M} \subseteq \{1, ..., L\}$. See A.3.3 for further discussion on this approach.

6 Results

6.1 Induction Heads

	s_1	m	s_2	*random*
Q_0	0.000	1.000	0.000	0.001
K_0	1.000	0.050	0.000	0.183
V_0	1.000	0.000	0.000	0.000
Q_1	0.000	0.000	1.000	0.000
K_1	0.000	1.000	0.000	0.000
V_1	0.000	5.053	0.000	0.000

(a) Average active subcomponents.

	Total unique
Q_0	1
K_0	1
V_0	1
Q_1	1
K_1	1
V_1	11

(b) Total unique subcomponents.

Metric	Value
$\mathcal{L}_{faithful}$	$3e-9$
$\mathcal{L}_{recon}$	$1e-4$
$\mathcal{L}_{recon_layerwise}$	$1e-4$
$\mathcal{D}_{KL}Attn$ (Layer 1)	0.385
$\mathcal{D}_{KL}Attn$ (Layer 2)	0.002
$\mathcal{D}_{KL}Attn$ (Mean)	0.194

(c) Evaluation metrics.

Table 1: Results for the induction-head toy model decomposition

For the induction head toy model, only two positions matter per layer; m must first attend to s_1 and 'understand' that it follows s_1, and then s_2 must attend to m to obtain m's identity. We see in Table 1a that subcomponents are assigned largely to the relevant positions, and Table 1b shows a very small amount of unique subcomponents. Table 1c shows almost-zero faithfulness, which indicates that if we consider all subcomponents causally important, we recover exactly the original model. $\mathcal{L}_{recon}$ and $\mathcal{L}_{recon_layerwise}$ are equivalent to substituting all the weight matrices in the original; either all at once or layer-by-layer. This can be compared to ablating all the presumed-unimportant components of a model and seeing if it still works.

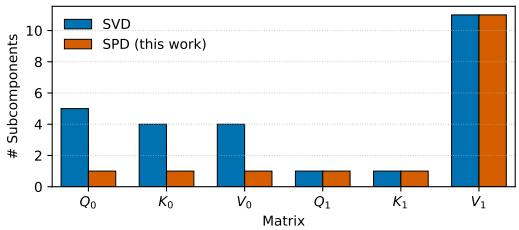

Figure 2: SPD vs. greedy SVD low-rank approximation (Algorithm 2). Even on a very simple model, SPD finds a more minimal decomposition that matches the full model's output distribution.

Investigating the concrete mechanisms, we learn that K_0's objective is to align the representation of token n with the positional encoding of $n+1$. When we do not add positional encodings prior to projecting m through the Q-matrix, the attention-strength from m to s_1 drops from 1.0 to 0.0941, even if s_1's key gets to retain its positional information. Q_1 and K_1 implement the circuit in which s_2 attends to m, and we see considerably more active subcomponents in V_1 for m's position. We believe this is because m's identity among 128 tokens must be unambiguously separable in the residual stream at the final position, and therefore higher rank than 1 is required to carry that information.

6.2 GPT2-Small

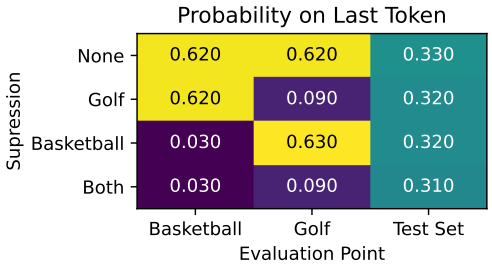

Figure 3: Ablating the slices associated with recovered facts has significantly higher effect on the specific data points.

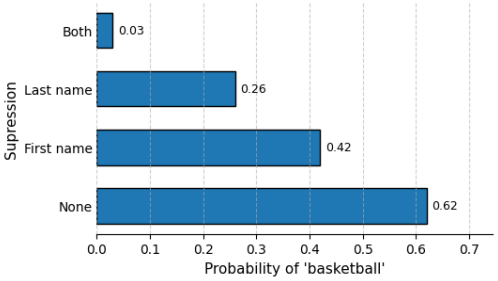

Figure 4: Both the first and last name for Kobe Bryant appear to carry basketball information and removing a rank 1 slice from both is most effective.

For the 2 examples, we find 96 active subcomponents in total, which is a 99% size reduction compared to the whole model [3]. We find 2 subcomponents that uniquely activate in the layer 0 MLP for tokens "obe" and "Bryant" and 1 uniquely for "Woods", that significantly reduce the probability of the sport being correctly predicted, if we suppress their directions in the original model. We see in figure 3 that these effects are isolated, which suggests we may have discovered where the network "stored" that fact. This extends previous work [11] which initially suggested facts may be stored closer towards the middle of the network, but where follow-up work [10] came to the same conclusions we do, namely that you *can* edit a fact somewhere the network does not store it. Our results suggest some facts originate as early as the very first MLP layer, and propagate through the network. Interestingly, we find that for the "Kobe Bryant" example, we must suppress a rank-1 direction on both the first and last name, to maximally reduce the output probability. We believe this is because "Kobe" has a strong enough association with basketball by itself, whereas "Tiger" generally does not refer to a name. Importantly we suppress only the athlete $\rightarrow$ sport direction. If we reverse the direction and prompt with "The most famous athlete in golf is..", the model accurately answers "Tiger Woods". This is consistent with previous research [11] [1], which finds that knowledge is stored asymmetrically in language models.

7 Discussion

While our results suggest that SPD can be extended to Transformer models, several limitations remain. First, our experiments are restricted to small-scale models, leaving open questions about scalability to larger architectures. Second, the causal importance parameterization introduces additional computational overhead. Third, our evaluation is qualitative and task-specific, so further benchmarks are needed to establish generality across architectures and tasks. We also note the difficulty in establishing ground truth even for small, toy Transformers. While we obtain evidence that suggests our decompositions may be faithful, this is largely obtained through ablations.

Our results show that parameter decomposition can surface semantically specific mechanisms in Transformers. For induction heads, SPD recovers the expected two-step copy circuit. For GPT-2-small, ablating up to 2 rank-1 slices selectively reduces the probability of the targeted fact (e.g., “golf” for Tiger Woods) while leaving unrelated examples largely unaffected. This indicates that SPD yields narrow, causally relevant directions rather than broad capacity that would degrade performance indiscriminately.

These findings suggest that parameter-space objects can act as clean handles for mechanistic interventions. In future work we plan to extend this work to larger models, and to improve our definition of causal importance for computations with feature interactions.

[3] One caveat: Causal importance of subcomponents in the attention-mechanism. For instance, if a QK-circuit ends up with 0 active subcomponents, it is still "in-use", but attention is uniform (softmax over 0-vector). For this reason, we also do not compare sparsity or size with e.g pruning methods.

References

[1] Lukas Berglund, Meg Tong, Max Kaufmann, Mikita Balesni, Asa Cooper Stickland, Tomasz Korbak, and Owain Evans. The reversal curse: Llms trained on" a is b" fail to learn" b is a". *arXiv preprint arXiv:2309.12288*, 2023.

[2] Dan Braun, Lucius Bushnaq, Stefan Heimersheim, Jake Mendel, and Lee Sharkey. Interpretability in parameter space: Minimizing mechanistic description length with attribution-based parameter decomposition. *arXiv preprint arXiv:2501.14926*, 2025.

[3] Lucius Bushnaq, Dan Braun, and Lee Sharkey. Stochastic parameter decomposition. *arXiv preprint arXiv:2506.20790*, 2025.

[4] David Chanin, James Wilken-Smith, Tomáš Dulka, Hardik Bhatnagar, Satvik Golechha, and Joseph Bloom. A is for absorption: Studying feature splitting and absorption in sparse autoencoders. *arXiv preprint arXiv:2409.14507*, 2024.

[5] Brianna Chrisman, Lucius Bushnaq, and Lee Sharkey. Identifying sparsely active circuits through local loss landscape decomposition. *arXiv preprint arXiv:2504.00194*, 2025.

[6] Ian Covert, Scott Lundberg, and Su-In Lee. Explaining by removing: A unified framework for model explanation. *Journal of Machine Learning Research*, 22(209):1–90, 2021.

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

[8] Juechu Dong, Boyuan Feng, Driss Guessous, Yanbo Liang, and Horace He. Flex attention: A programming model for generating optimized attention kernels. *arXiv preprint arXiv:2412.05496*, 2024.

[9] Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. Toy models of superposition. *Transformer Circuits Thread*, 2022. `https://transformer-circuits.pub/2022/toy_model/index.html`.

[10] Peter Hase, Mohit Bansal, Been Kim, and Asma Ghandeharioun. Does localization inform editing? surprising differences in causality-based localization vs. knowledge editing in language models. *Advances in Neural Information Processing Systems*, 36:17643–17668, 2023.

[11] Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *Advances in neural information processing systems*, 35:17359–17372, 2022.

[12] Anh Nguyen, Jason Yosinski, and Jeff Clune. Multifaceted feature visualization: Uncovering the different types of features learned by each neuron in deep neural networks. *arXiv preprint arXiv:1602.03616*, 2016.

[13] Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. In-context learning and induction heads. *Transformer Circuits Thread*, 2022. https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html.

[14] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[15] Lee Sharkey. Taking features out of superposition with sparse autoencoders. `https://www.lesswrong.com/posts/z6QQJbtpkEAX3Aojj/interim-research-report-taking-features-out-of-superposition`, 2022. LessWrong post.

[16] Lee Sharkey. Mech interp is not pre-paradigmatic. https://www.lesswrong.com/posts/beREnXhBnzxbJtr8k/mech-interp-is-not-pre-paradigmatic, 2025. LessWrong post.

[17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

A Appendix

A.1 Causal Importance Algorithm

Algorithm 1 Minimal Attention Causal Importance

Require: Input $x \in \mathbb{R}^{B \times s \times d}$
Require: $\Gamma^L = (\gamma_1^l, ..., \gamma_C^l)$ // A function that assigns C causal importances per layer
Require: *Learnable* RelPosEnc $\in \mathbb{R}^{2 \cdot s \times d}$
Require: *Learnable* AbsPosEnc $\in \mathbb{R}^{s \times d}$
Require: *Learnable* $Q, K, V \in \mathbb{R}^{d \times d}$
Ensure: Output $G^l \in \mathbb{R}^{B \times s \times C}$

$x_{pos} \leftarrow x + \text{AbsPosEnc}(x)$
$q \leftarrow Q(x); \quad k \leftarrow K(x); \quad v \leftarrow V(x_{pos})$
$r \leftarrow \text{RelPosEnc}(s)$
$x_{combined} \leftarrow \text{softmax}(\frac{qk^\top + r}{\sqrt{d}})v$ // Implemented with Flex Attention
$x_{combined} \leftarrow \text{concat}(x_{combined}, x_{pos})$
$G^l \leftarrow \Gamma^l(x_{combined})$
return G^l

	Sequence Length			
Method	16	64	1024	10240*
Scalar	12	12	9	11
Vector	13	13	10	11
Attention (no flex)	10	10	2	1
Attention (flex)	10	10	6	6

Table 2: Iterations per second for different γ-MLP inputs on the induction-head decomposition task. *Batch size modified from 64 to 4 for sequence length 10240 due to memory limits.

A.2 Induction-Head Transformer

A.2.1 Hyperparameters

Model hyperparameters	
Batch Size	1024
Optimizer	AdamW
Learning Rate	0.001
Weight Decay	0.01
Learning Rate Schedule	Constant
Learning Rate Warmup	1000
Steps	100000
D_{model}	16
Pos_{enc}	Shortformer
Use Layer Norm	False
Use FF	False
Vocab Size	128
Sequence Length	64

Table 3: Hyperparameters for the Induction Head toy-model task

Decomposition hyperparameters	
Batch Size	1024
Optimizer	Adam
Learning Rate	0.001
Learning Rate Schedule	Cosine
Learning Rate Warmup	0
Steps	100000
C	100
D_{gate}	16
CI-function	Attention
Stochastic Recon β	1
Stochastic Layerwise Recon β	1
Recon β	0.5
Faithfulness β	1000
Importance Minimality β	0.02
P_{start}	0.9
P_{final}	0.1
Anneal P?	True
Last Token Only?	True

Table 4: Hyperparameters for the Induction Head decomposition task

A.2.2 Loss Curve

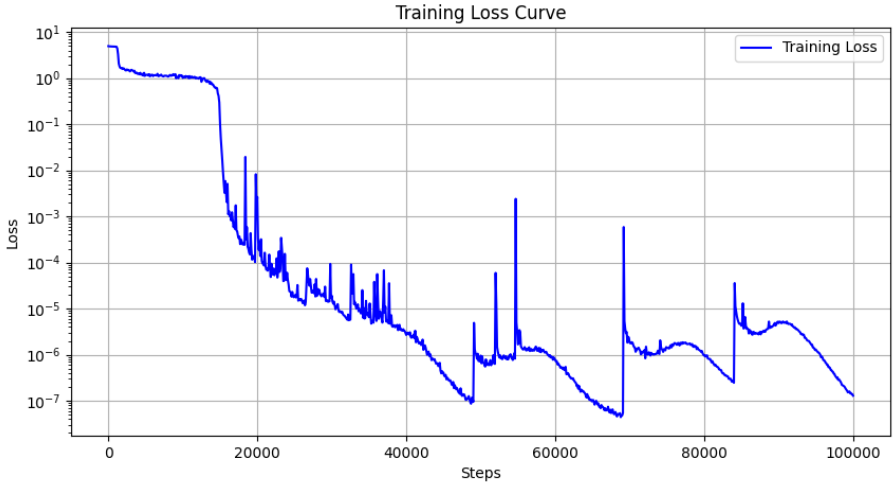

Figure 5: Loss curve for the Induction Head Transformer showing that expected 'phase changes' are present.

A.2.3 SVD Baseline

Algorithm 2 Greedy Rank-1 SVD Pruning

Require: Weight matrices $\{W_i\}$ from target model $\mathcal{M}$
Require: Dataset $\mathcal{D} = \{x_b\}_{b=1}^{N}$
Require: Tolerance τ
Require: Loss function $\mathcal{L}$ // D_{KL} in our case.
Ensure: Reduced-rank weight matrices $\{\tilde{W}_i\}$

```
Decompose each W_i with SVD: W_i = U_i S_i V_i^⊤
current_loss ← 0
P ← M(D)
while current_loss ≤ τ do
    for W_i that still has rank > 0 do:
        Form W_i' by removing its smallest remaining singular value/vector
        Define M̃ as M with W_i replaced by W_i'
        Q ← M̃(D)
        Evaluate L_i = D_KL(P||Q)
        Pick index j with lowest L_j
        if L_j ≤ τ then
            Update W_j ← W_j' in M
            current_loss ← L_j
        else
            break
        end if
    end for
end while
return pruned weight matrices {W̃_i}
```

A.3 GPT-2

A.3.1 Hyperparameters

Decomposition hyperparameters	
Batch Size	2
Optimizer	AdamW
Learning Rate	0.001
Weight Decay	0.05
Learning Rate Schedule	Constant
Learning Rate Warmup	0.01
Steps	20000
C	768
D_{gate}	64
CI-function	vector
Stochastic Recon β	0.1
Partial Stochastic Recon β	10
Stochastic Layerwise Recon β	0.1
Recon β	0.1
Partial Recon β	10
Faithfulness β	1000000
Importance Minimality β	0.05
P_{start}	0.9
P_{final}	0.1
Anneal P?	True
Last Token Only?	False

Figure 6: Hyperparameters for the GPT2 decomposition task

A.3.2 Fact Location

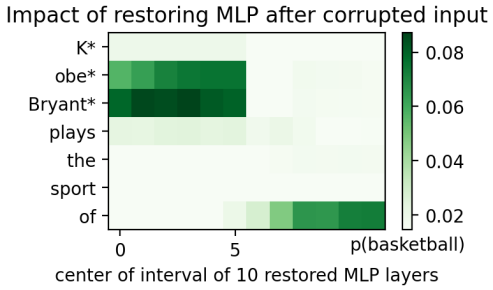

Figure 7: Causal tracing on the Kobe Bryant example

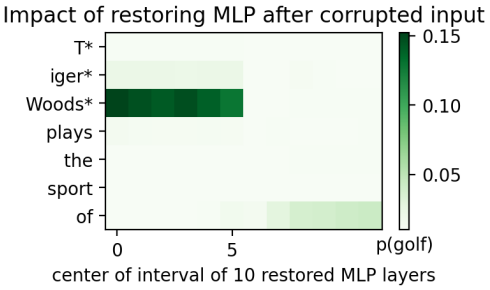

Figure 8: Causal tracing on the Tiger Woods example

Figure 7 and 8 show the result of running causal tracing from Meng et al.[11] on our 2 fact-examples. Causal tracing suggests that the fact can be edited all the way from layer 0 MLPs to the layer 5 MLPs. In our findings, we see evidence that the athlete $\rightarrow$ sport association is present as early as after the first MLP layer. The fact is editable up until the fifth MLP layer, after which the information is likely moved across the sequence with attention.

Editing a fact comes down to correlating a specific read direction $\vec{U}$ ($\vec{K}$ in [11]) with a desired write direction $\vec{V}$. Under that framing, it is easy to see how *any* MLP that encounters a certain fact before the unembedding. In Meng et al.[11] they edit facts by solving a small optimization problem to yield $\vec{V}$ that maximises the probability of the fact they want to predict. The key-vector $\vec{K}$ – the input to the

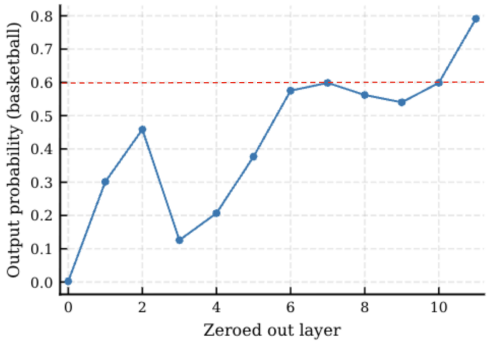

Figure 9: The effect of zeroing out entire MLP-layers in GPT2-small on the "Kobe Bryant plays the sport of" example. Dashed line is the full model's output probability.

MLP – could in theory be anything. In later layers the *basketball*-concept may have already been written to the residual stream and we are introducing a mapping from basketball to e.g. football.

In figure 9 we take this to the extreme, and show that you can actually entirely zero-out the layer in which it is most beneficial to edit the fact, showing conclusively that the fact does not live there. Note that the drop in probability to 0 when zeroing out layer 0 does *not* imply the fact lives there – zeroing out this layer simply destroys the model.

A.3.3 Partial losses

When decomposing large models over very few samples, the model has room to "cheat" the task and simply learn a function that produces the same output. Specifically, we often see this in the form of the model maintaining one layer (the last) with a suboptimal amount of subcomponents active, such that it can keep the other layers very sparse. If you consider a model decomposed over *just* the example "Tiger Woods plays golf", it is easy to see how the model could just learn a handful of rank-1 subcomponents in the final layer, that writes the relevant token directions onto the residual stream, instead of learning the actual underlying mechanism. When decomposing over larger datasets, this effect is minimised as the "cheating" subcomponents would incur heavy reconstruction losses for all other examples.

We circumvent this with the 2 losses from section 6.2:

$$\mathcal{L}_{recon_partial} = D_{KL}\big(f\big(x \mid W^1, \ldots, W^{l\in\mathcal{M}}(x, g^l(x)), \ldots, W^L\big),\ f(x \mid W)\big)\,, \tag{7}$$

$$\mathcal{L}_{stochastic_recon_partial} = \frac{1}{S}\sum_{s=1}^{S} D_{KL}\Big(f\big(x \mid W^1, \ldots, W^{l\in\mathcal{M}^{(s)}}(x, r^{l,(s)}), \ldots, W^L\big),\ f(x \mid W)\Big)\,, \tag{8}$$

where $\mathcal{M} \subseteq \{1, ..., L\}$.

These are equivalent to $\mathcal{L}_{recon}$ and $\mathcal{L}_{stochastic_recon}$, but they sample only some subset of layers in the model to replace and leave everything else in the target model intact. This severely reduces the ability for the model to implement "cheating" mechanisms in any layer, as it **1)** may not be able to rely on that layer and **2**) these mechanisms might clash with the weight matrices of the full model.

A.3.4 Fact dataset

```
Serena Williams is a professional tennis player
The Great Wall of China is in China
The Mona Lisa was painted by Leonardo da Vinci
Mount Everest is the tallest mountain on Earth
The Eiffel Tower is located in Paris France
Usain Bolt is the fastest man in the world
The Pacific Ocean is the largest ocean on Earth
Albert Einstein developed the theory of relativity
J.K. Rowling wrote the Harry Potter series
The Amazon River flows through South America
William Shakespeare wrote Romeo and Juliet
Beethoven composed the Fifth Symphony
The Sahara Desert is in Africa
Apple Inc. was founded by Steve Jobs Steve Wozniak and Ronald Wayne
The sun is a star
The Beatles were a British rock band
Cristiano Ronaldo is a professional soccer player
The Statue of Liberty is in New City
Pablo Picasso was a famous painter
The Nile is the longest river in the world
Charles Darwin developed the theory of evolution
Google was founded in 1998
The Taj Mahal is in India
The Titanic sank in 1912
Jane Austen wrote Pride and Prejudice
The Moon orbits the Earth
Mount Kilimanjaro is in Tanzania
The Leaning Tower of Pisa is in Italy
Walt Disney created Mickey Mouse
Vincent van Gogh painted Starry Night
```

Figure 10: Excerpt from GPT-5 generated "fact" dataset.

A.4 Training routine

Algorithm 3 SPD Training Step (Single Minibatch)

Require: Original model $\mathcal{M}_\theta$
Require: Decomposed model $\tilde{\mathcal{M}}_\phi$
Require: Causal-importance routine $\mathcal{C}_\phi$
Require: Dataset $\mathcal{D}$
Require: Loss functions $\{\mathcal{L}_k\}_{k=1}^n$ with coefficients $\{\lambda_k\}_{k=1}^n$

```
Sample minibatch X ⊂ D
Y ← M_θ(X)      // run original model
C ← C_φ(X)      // compute causal importances on this batch
R ← SampleMasks(C)      // sample stochastic subcomponent masks
Ỹ ← M̃_φ(X; R)      // run decomposed model with masks
L ← 0
for k = 1 ... n do
    L ← L + λ_k L_k(X, Y, Ỹ, C, R, M_θ, M̃_φ)
end for
Update parameters of M̃_φ and C_φ
```