Hidden Signatures: Using Crosscoders to Compare Transformer Features Between Model Pairs

Lisann Becker [1] **Morris de Haan** [1] **Krijn Dignum** [1] **Aaron Kuin** [1] **Gjalt Hoekstra** [1]

Abstract

Understanding how knowledge is preserved during model transformation remains challenging. We introduce novel metrics for quantifying feature sharing between transformer models using crosscoders—sparse autoencoders that learn shared dictionaries across multiple models. We extend crosscoders to handle different residual-stream dimensions and develop two metrics: "sharedness" based on weighted entropy of decoder norms, and feature alignment using canonical correlation analysis. Evaluating four model relationship types (identical, instruction-tuned, different scales, and distilled pairs), we find early layers consistently exhibit high feature sharing while middle layers diverge based on the transformation method. These findings reveal how different techniques preserve representations and establish crosscoders as a tool for mechanistic model analysis. Our code is available at github.com/lisannbecker/crosscoder-model-diff-extension.

1. Introduction

In recent years, foundation models achieved remarkable success across a wide range of scientific and commercial applications, largely due to their broad and generalizable world knowledge. However, directly integrating these models is often impractical, either because they are not well-suited to specific tasks or because of their substantial computational demands. Instead, relevant knowledge is often extracted from foundation models through compression techniques such as distillation or finetuning.

Despite the growing body of research on knowledge extraction techniques, the scientific literature offers limited insight into what is actually transferred this process. Yet, understanding how and what knowledge is conveyed is essential for evaluating the fidelity and transparency of derived models. To this end, our objective is to compare learned high level concepts between a foundation and derived model. To achieve this, it is not sufficient to naively compare neuron activation patterns, as models may encode identical concepts in different subspaces.

Crosscoders, introduced by (Lindsey et al., 2024) offer a promising solution to this challenge. Where sparse autoencoders learn dictionaries of interpretable features for individual layers, crosscoders extend this approach to learn shared dictionaries that can reconstruct activations from multiple models simultaneously. This enables the identification of both shared features—concepts preserved across models—and model-specific features that emerge or disappear during knowledge extraction processes. By training on concatenated activation data from model pairs, crosscoders create a unified latent space that facilitates direct comparison of internal representations.

Our contributions are threefold. First, we develop novel metrics for quantifying feature sharing: a "sharedness" metric based on weighted entropy of relative decoder norms, and a feature alignment metric using canonical correlation analysis that works across different dimensionalities. Second, we extend crosscoders to handle transformer models with varying residual-stream dimensions by introducing separate encoder and decoder heads for each model, enabling comparison between architectures of different scales and distilled model pairs—a capability not previously demonstrated. Third, we conduct the first systematic application of crosscoders to teacher-student distillation relationships, complementing existing work on instruction-tuned and differently-scaled models.

We investigate feature sharing across four distinct model relationship types: identical models (baseline), base and instruction-tuned models, separately trained models of different scales, and teacher-student distilled pairs. Our analysis reveals that early layers consistently exhibit high feature sharing across all model types, while middle and later layers show varying patterns of divergence depending on the knowledge extraction method employed. These findings pro-

[1]University of Amsterdam, Netherlands..

UvA Foundation Models Course 2025, Project Paper Submission.

vide new insights into how different compression techniques preserve or transform learned representations.

2. Background

2.1. Concept decoding

Concept decoding refers to techniques aimed at uncovering and interpreting the internal representations of neural networks. In large language models (LLMs), these representations are distributed across multiple layers and heads and can encode complex features. Concept decoding is crucial not only for interpretability but also for measuring representational alignment between different models. Which may in turn allow us to better manipulate these models to perform better on downstream tasks. In this section we detail two novel approaches to concept decoding, Sparse Auto-Encoder introduced in 2023 and Crosscoders introduced in 2025.

Sparse Auto-Encoders Sparse autoencoders are an increasingly interesting method for extracting interpretable features from the hidden layers of models (Bricken et al., 2023; Shi et al., 2025). By encouraging sparsity in the hidden layer activations, these models push the network to represent input information using a small number of high-activation features, which often correspond to meaningful features. In the context of transformer models, recent work Templeton et al. (2024) has demonstrated that Sparse autoencoders produce interpretable features for large models. The approach is particularly effective when applied to large foundation models, since the models themselves don't have to be updated in any way.

Crosscoders Crosscoders, introduced by Lindsey et al. (2024), are a novel interpretability tool designed to analyze and compare internal representations across different layers or models. In this paper we will focus on the comparison of the internal representations of different models. Thus we will omit the theory for comparison of different layers as this is not relevant to our research goals. Sparse crosscoders extend the concept of sparse autoencoders by learning a shared dictionary of features that can reconstruct activations from multiple models. This approach enables the identification of shared and distinct features between models, facilitating a deeper understanding of how specific concepts are represented and transformed through model architectures.

In practice, crosscoders are trained on concatenated activation data from two transformer-style models (e.g., a base model and its fine-tuned counterpart). The training objective encourages the discovery of latent features that can reconstruct the activations of both models. The resulting latent space allows for the classification of features into two main categories:

- **Shared features**: Features that are shared by both models, implying a measure of potential similarity between two models.
- **Model-specific features**: Features that are exclusive to either model. Implying a potential measure of dissimilarity between models.

These generated features allow us to compute quantitative metrics that measure model similarity. We elaborate on these metrics in section 3.3.

2.2. Knowledge extraction

Knowledge extraction methods aim to transfer meaningful representations learned by a larger teacher model, into a smaller and simpler student model. These approaches have been widely studied in the context of neural networks and language models, where interpretability and computational efficiency are both key concerns (Hinton et al., 2015; Romero et al., 2015; Zagoruyko & Komodakis, 2017; Sanh et al., 2020; Wang et al., 2020; Xu et al., 2024). In this section, we review three major techniques relevant to our work: softmax distillation, Jacobian distillation, and fine-tuning.

Softmax distillation First introduced by Hinton et al. (2015), softmax knowledge distillation is the de-facto standard for knowledge distillation. In this method, the student model is trained to match the softened output probabilities, controlled by a temperature, of the teacher model. This provides richer supervision than hard labels alone, as it encodes the teacher's confidence and relative ranking of predictions. In the context of large language models, this method has been used to create compact variants such as DistilBERT created by Google (Sanh et al., 2020) and DistilGPT2, created by Huggingface in a similar manner to DistilBERT. These distilled models retain much of the teacher's performance while reducing inference cost.

Jacobian distillation Jacobian-based distillation extends this idea by also aligning internal gradients. Originally proposed by Czarnecki et al. (2017) as sobolev training, this technique involves matching the Jacobian of the output with respect to the input between the teacher and student models. This encourages the student to replicate the sensitivity of the teacher model, not just its output distribution. Srinivas & Fleuret (2018); Wu et al. (2023); Kokane et al. (2024) have shown that Jacobian matching can improve transfer of generalization behavior and help align feature representations. While more computationally demanding, it is particularly relevant when comparing internal mechanisms of models, as we do in our work. An important note on the papers mentioned above. The Jacobian Matching method described

in their respective papers are applied to neural networks or convolutional networks. To the best of our knowledge there has been no research done on Jacobian Matching in the field of foundation models.

Fine tuning Finetuning is the process of adapting a pretrained model to a new task or domain by continuing training on a smaller, typically task-specific dataset (Han et al., 2024). Fine tuning can be applied in nearly all fields of machine learning. However, in this paper we will be referring to the fine tuning of foundation models such as Sonnet 3.0.

3. Methodology

In the following section, we detail our approach and choices for comparing a foundation and derived model step by step.

As established, crosscoders encode activations from corresponding layers in two transformer models into a shared latent feature space, allowing feature comparison between models.

Initially, crosscoders implementations utilised a unified encoder and decoder to project activations from both models into and out of a shared latent feature space. This approach requires both models to have identical residual-stream dimensionalities, and enables comparison between training snapshots, training runs, dataset changes, or finetuning. However, comparison between model scales, architectural changes (affecting dimensionality), or distilled models requires a setup that is flexible differently scaled models.

Our contribution includes the extension of crosscoders to handle transformer models with varying residual-stream dimensions. We introduced separate encoder and decoder heads, effectively creating two distinct encoders and decoders - one set for each model - to project model-specific activations into a common latent feature space. Concretely, for two models A and B with residual-stream dimensions d_A and d_B, the crosscoder consists of:

Model-specific encoder weight matrices:

$$W_{enc}^A \in \mathbb{R}^{d_{latent} \times d_A}, \quad W_{enc}^B \in \mathbb{R}^{d_{latent} \times d_B}$$

Model-specific decoder weight matrices:

$$W_{dec}^A \in \mathbb{R}^{d_A \times d_{latent}}, \quad W_{dec}^B \in \mathbb{R}^{d_B \times d_{latent}}$$

The encoder projects activations from model A and B to the latent space via:

$$z^A = W_{enc}^A x^A, \quad z^B = W_{enc}^B x^B$$

where x^A and x^B represent activations from the corresponding layer in each model, and $z^A, z^B \in \mathbb{R}^{d_{latent}}$ are the latent feature representations. The decoder reconstructs activations for each model from this shared latent representation:

$$\hat{x}^A = W_{dec}^A z^A, \quad \hat{x}^B = W_{dec}^B z^B$$

Training the crosscoder involves optimising a loss function composed of two components:

1. L2 Reconstruction loss, measuring fidelity of the decoded activations:

$$L_{rec} = \|x^A - \hat{x}^A\|_2^2 + \|x^B - \hat{x}^B\|_2^2$$

2. Sparsity regularisation, which encourages fewer active features:

$$L_{sparse} = \|z^A\|_1 + \|z^B\|_1$$

The total loss is defined as:

$$L = L_{rec} + \lambda L_{sparse}$$

where λ is a hyperparameter controlling the sparsity strength.

We use our extended crosscoder architecture to construct a shared latent space between models of different scales - an idea previously discussed ((Lindsey et al., 2024)) but not open-sourced - and, for the first time, between teacher–student (distilled) model pairs, a setting that has not been discussed or open-sourced before. Later, we analyse feature sharing by examining the relative decoder norms and latent space distributions.

3.1. Models

To investigate how concept conveyance differs between various compression and knowledge extraction schemes, we compare four distinct types of model pairs:

- **Pairs of identical models (Baseline)**: For all models we used in our experiments, we start by comparing it to itself to establish a baseline for maximum feature alignment when architecture is identical.
- **Base and instruction-tuned models (Replication)**: Replicating Lindsey et al. (2024) using Kissane et al. (2024); Kissane (2024)'s code base, we compare Gemma base (Gemma-2 2B) and its instruction-tuned counterpart Gemma-2 2B IT (Team, 2024). Both models have identical architectures with 2 billion parameters, featuring 26 transformer blocks and a residual-stream dimension of $d_{\text{model}} = 2048$.

- **Separately trained models of different scales**: We compare two independently trained Pythia models (Biderman et al., 2023) at different scales—160M (larger) and 70M (smaller). The larger model comprises 12 transformer blocks with a residual-stream dimension of $d_{\text{model}} = 768$, whereas the smaller model consists of 6 transformer blocks with a dimension of $d_{\text{model}} = 512$.
- **Teacher and distilled student models**: For distilled models, we use GPT-2 (137M parameters) (Radford et al., 2019) as the teacher model and DistilGPT-2 (88M parameters) (Sanh et al., 2019) as the student model, which is a KL-divergence distilled variant of GPT-2. GPT-2 has 12 transformer blocks with a residual-stream dimension of $d_{\text{model}} = 768$, while DistilGPT-2 consists of 6 transformer blocks, also with a residual-stream dimension of $d_{\text{model}} = 768$.

3.2. Training

We train a separate crosscoder for the first, middle, and last layers of each model pair described in Section 3.1.

For the scale (Pythia 160M–70M) and distillation (GPT-2–DistilGPT-2) experiments, the "middle" layer comparison refers to layers 6 (of 12) in the larger model and layer 3 (of 6) in the smaller model, while the "last" layer refers to layer 12 (final layer) in the larger model and layer 6 (final layer) in the smaller model. In contrast, for the instruction-tuning experiment (Gemma-2-2B and Gemma-2-2B-IT) and baseline scenarios involving identical models, we directly compare activations at matching layers (e.g., layers 13 and 26 for middle and last layers respectively for Gemma).

For the baseline, instruction-tuning, scale, and distillation scenarios, training setups are consistent across settings, differing mainly in terms of specific hyperparameters adapted for computational feasibility:

Optimisation details: We optimise the crosscoder parameters using Adam with hyperparameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The initial learning rate is set to 5×10^{-5}. As detailed above, our loss balanced reconstruction fidelity with feature sparsity.

Batching and data:

- For the scale (Pythia 160M–70M) and distillation (GPT-2–DistilGPT-2) experiments, we use a batch size of 1024 tokens per update, with gradient accumulation steps resulting in effective batch sizes of 4096 tokens.
- For the instruction-tuning experiment (Gemma-2-2B and Gemma-2-2B-IT), we directly use a batch size of 4096 tokens per update without gradient accumulation due to larger model dimensions and hardware constraints.

Training duration: Crosscoders in the distillation and scale experiments are trained until a total of 200 million token representations have been observed, while for the Gemma experiments, we end training when 400 million tokens have been seen.

Additionally, we use specialised buffers to efficiently retrieve pre-tokenised data sequences, which are cached locally for computational efficiency. For all the experiments, a custom token pipeline was created by retokenising data sequences specifically into tokens compatible with the models. The resulting data was then stored in memory-mapped files and efficiently retrieved during training.

During training, metrics such as latent feature sparsity, reconstruction accuracy (explained variance), and relative decoder showed convergence and stability of our training.

3.3. Metrics

After crosscoder training, we evaluate the resulting feature spaces along two dimensions: *feature sharedness* and *feature alignment*. Feature sharedness quantifies the extent to which different models rely on the same features during inference. A low sharedness score indicates that models base their decisions on distinct features, whereas a high score suggests substantial overlap in the features they use. Feature alignment, on the other hand, measures the similarity between learned feature representations across models. A higher alignment score implies a closer match between representations, indicating better alignment.

3.4. Feature sharing

To capture the extent to which high level features are shared between models, regardless of model-specific activation patterns, we introduce a metric called "sharedness". This metric is derived from a histogram of relative feature vector norms (Equation 1), taken from the crosscoder decoder. Each feature's contribution is weighted by its average activation strength, as determined by the crosscoder encoder over the entire dataset. This weighting downplays the influence of infrequently activated features and emphasizes more prominent features.

Empirically, the resulting distribution tends to be tri-modal: features with relative norms near 0 or 1 correspond to model-specific features, while those near $1/2$ suggest ambiguity, indicating that the feature is equally expressed in both models. Based on this interpretation, we use entropy to quantify the uncertainty or distribution spread of these sharedness values, forming the basis of our proposed metric (Equation 2).

$$n_{\text{rel}}(i; A, B) = \frac{|f_A(i)|}{|f_A(i)| + |f_B(i)|} \tag{1}$$

Figure 1. The relative norm of a feature i is computed as the feature norm in model A divided by the sum of feature norms in model A and B. A value of 1 denotes that the feature is solely present in model A, whereas a value of 0 that the feature is specific to model B. On the other hand, a value of $1/2$ means that the value is shared equally.

$$\text{sharedness}(A, B) = \sum_{i<N_f}^{i=0} [n_{\text{rel}}(i; A, B) \cdot \log n_{\text{rel}}(i; A, B)] \cdot \bar{a}(i) \tag{2}$$

Figure 2. The sharedness of features between model A and B given N_f features and the average feature activation strength $\bar{a}(i)$. The function represents the weighted entropy of relative norms.

3.5. Feature alignment

The second metric aims to assess the degree to which feature representations between two models are aligned. In the crosscoder paper (Lindsey et al., 2024), the authors rely on cosine distance to measure representation similarity. However, in our setting, we compare different model types that may have layers of varying dimensionalities. To address this, we use canonical component analysis (CCA) (Yang et al., 2019), which quantifies the correlation between two sets of random variables, even when their dimensionalities differ. More precisely, CCA identifies linear projections of both sets that maximize their mutual correlation. This makes CCA especially suitable for our purposes, as it can detect high similarity in features representations even when they occupy different subspaces.

4. Experimental setup

4.1. Models

Our experimental design investigates feature sharing across four distinct model relationships types, each representing different approaches to knowledge extraction and model development. All models are English-language transformer architectures accessed through the TransformerLens library.

Base and instruction-tuned models (Gemma): We compare Gemma-2-2B base and Gemma-2-2B-IT, both 2 billion parameter models with identical architectures. The base model was trained on 2 trillion tokens from diverse sources including web documents, code, and mathematical text. The instruction-tuned variant underwent additional fine-tuning on instruction-following data to improve conversational and task-specific capabilities while maintaining the same underlying architecture.

Separately trained models of different scales (Pythia): We examine two independently trained Pythia models—Pythia-160M and Pythia-70M, both developed by EleutherAI. Both models were trained on identical data in the exact same order: The Pile, an 825GB English dataset containing 299.9 billion tokens from 22 diverse sources spanning academic writing, internet content, prose, dialogue, and miscellaneous sources. This controlled training setup enables clean comparison of scale effects while eliminating data distribution confounds.

Teacher and distilled student models (GPT-2): For knowledge distillation analysis, we use GPT-2 (137M parameters) as the teacher model and DistilGPT-2 (82M parameters) as the student. GPT-2 was pre-trained on WebText, a 40GB corpus scraped from Reddit outbound links with at least 3 karma. DistilGPT-2 was created through knowledge distillation using OpenWebTextCorpus, an open-source reproduction of the original WebText dataset, following the distillation procedure described in (Sanh et al., 2020)

Baseline (identical models): As a control condition establishing maximum expected feature alignment, we compare each model against itself using GPT-2, providing an upper bound for feature sharing metrics when architecture and training are identical. This model selection enables systematic investigation of how different knowledge extraction methods—fine-tuning, scale variation, and distillation—affect the preservation and transformation of learned representations across model pairs.

4.2. Data

For the Gemma fine-tuning experiments, we replicated a custom-prepared dataset combining data from The Pile ((Gao et al., 2020)) and LMSYS-Chat ((Zheng et al., 2023)) datasets, previously tokenised for Gemma compatibility and released by (Kissane, 2024). Since this tokenised version was not readily available for streaming efficiently, we implemented a custom data pipeline. Our pipeline reprocessed this dataset into a compact, disk-backed memory map format, allowing efficient token streaming during crosscoder training without repeatedly accessing the HuggingFace API.

For the GPT-2 and Pythia experiments, we relied on OpenWebText ((Aaron Gokaslan*), a widely-used corpus in trans-

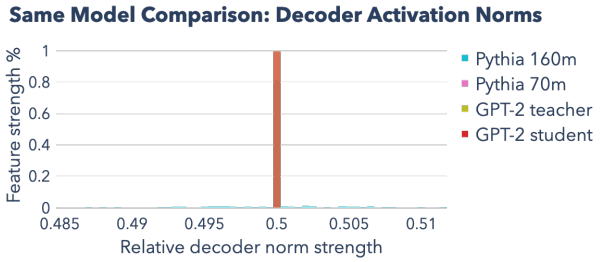

Figure 3. Decoder activation norm strength distributions for Pythia 160m, Pythia 70m, GPT-2 teacher, and GPT-2 student models. This comparison serves as a baseline for understanding differences in feature strengths across similar architectures.

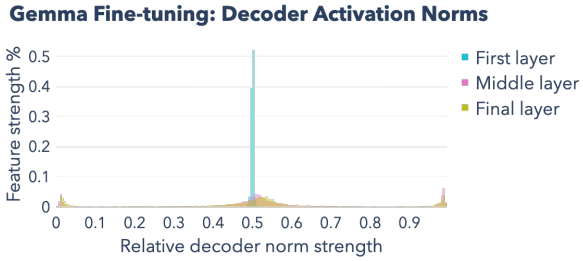

Figure 4. Distribution of decoder activation norm strengths across layers for fine-tuned Gemma models. This demonstrates how feature strengths are distributed in the first, middle, and final layers after fine-tuning.

former research that approximates the web data the used during pre-training of these models. For both GPT-2 and Pythia, we retokenised OpenWebText using their respective HuggingFace tokenisers to produce model-compatible token streams. Again, these token streams were converted into memory-mapped files for efficient and rapid access during training.

The custom tokenisation pipelines significantly streamlined the training process, ensuring computational feasibility and consistent experimental conditions across all model pairs.

5. Results

First of all, from Table 8 we observe that the initial layers exhibit high feature sharing, indicating a substantial overlap in used features across models. However, when a model is retrained from scratch, as is the case for Pythia, the resulting feature representations diverge significantly. Interestingly, applying distillation only at the final layer still leads to highly similar representations in the first layer, suggesting early features can be aligned with minimal supervision.

Another observed trend is that representation similarity decreases in the middle layers, suggesting models apply initial

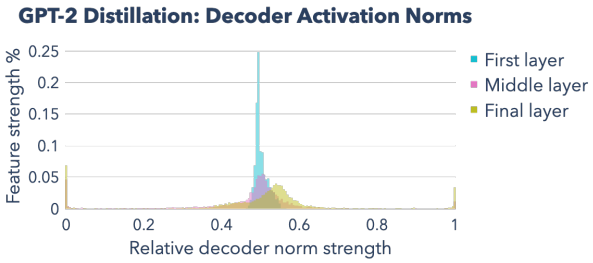

Figure 5. Distribution of decoder activation norm strengths across layers for distilled GPT-2 models, highlighting how feature strengths are distributed throughout the network.

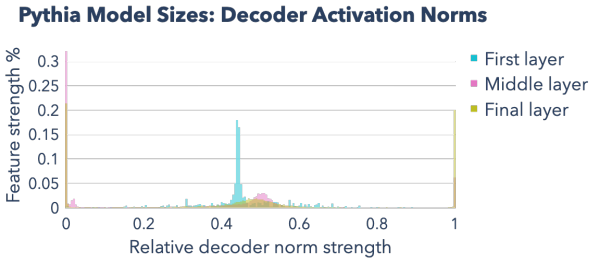

Figure 6. Comparison of decoder activation norm strengths across layers for different Pythia model sizes. This illustrates how feature strengths vary between the first, middle, and final layers.

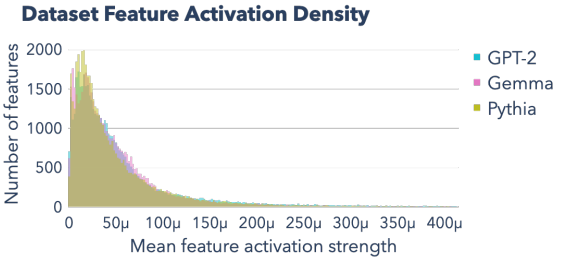

Figure 7. This shows how feature activations are distributed across different model architectures.

Transformation	First Layer	Middle Layer	Final layer
Gemma Fine-tuning	1.00 / 0.999	0.77 / 0.768	0.76 / 0.807
GPT-2 Distillation	1.00 / 0.937	0.91 / 0.604	0.85 / 0.715
Pythia Scale	0.96 / 0.286	0.54 / 0.343	0.54 / 0.353
Sanity Check	1.00 / 1.000	1.00 / 1.000	1.00 / 1.000

Figure 8. Sharedness and alignment metrics across model pairs and layers. Values show sharedness/alignment for first, middle, and final transformer layers. Green indicating high similarity, red indicating low similarity. Sanity check represents identical model comparisons establishing the upper bound.

representations in distinct ways. In the case of finetuning, however, middle and later representations remain relatively similar. This is expected, as the finetuned model begins with the exact parameterization of the base model, unlike the distilled model which acquires representations through supervision.

In the final layers, we observe equal or decreased feature sharing compared to the middle layers, yet an increase in feature alignment. This pattern suggests that while models may rely on distinct features, they converge towards similar task-specific representations. This likely reflects the common objective of optimizing performance for the target task.

Furthermore, from Table 8 and Figure 3 show that training a crosscoder on two instances of the same model yields maximal feature sharing and alignment. This is an expected outcome given the identical model parameterization.

The relative norm strength histograms in Figures 3, 4, 5, and 6 exhibit the same tri-modal pattern as previously identified by (Lindsey et al., 2024). Notably, the initial layers consistently show narrower distributional spread compared to the middle and final layers, suggesting that deeper layers introduce more variation in how representations are utilized across models. Lastly, Figure 7 illustrates the distribution of feature activations, revealing that activation strength varies significantly across features within the dataset.

6. Conclusion and Discussion

We introduced novel metrics for quantifying feature sharing between transformer models using crosscoders, extending their applicability to models with different residual-stream dimensions. Our systematic evaluation across four model relationship types reveals consistent patterns in how different transformation methods preserve or alter learned representations.

Main findings: Early layers consistently exhibit high feature sharing across all transformation types, indicating that fundamental representational structures are largely preserved. Middle layers show the greatest divergence, suggesting that models utilize these foundational features in distinct ways depending on the transformation method. Fine-tuning maintains higher similarity throughout the network compared to independent training or distillation, reflecting the preservation of the original parameterization. Notably, distillation achieves early-layer alignment despite supervision only at final layers, demonstrating that basic features can be preserved with minimal direct supervision.

Implications: These findings provide new insights into the mechanistic effects of different model transformation techniques. The consistent early-layer similarity across methods suggests that low-level linguistic features are universal and robust, while the middle-layer divergence indicates where task-specific adaptations occur. Our sharedness and alignment metrics offer interpretable measures for evaluating representation preservation during model development.

Limitations: Our analysis focuses on English-language transformer models and specific transformation types. The crosscoder approach, while providing valuable insights, abstracts away from the literal model mechanisms and may not capture all aspects of representational similarity.

Future work: We propose three directions for future work. Firstly, the crosscoder framework can be extended to new modalities, namely vision transformers, as current research is limited to large language models. Secondly, it should investigate the effects of different distillation techniques, specifically classic KL divergence with Jacobian matching, which might differently affect latent similarity. We prepared a nearly functional distillation pipeline for GPT-2 using Jacobian matching - an approach not available in open-source distilled models - to enable a direct comparison with KL-divergence-based distillation. Memory limitations prevented our full analysis before submission, however, our codebase lays the groundwork for future work. Lastly, layer feature analysis may be combined with circuit analysis methods like transcoders (Ameisen et al., 2025) for deeper mechanistic understanding. Pursuing these three directions will further demonstrate the value of our metrics and architectural extensions, solidifying crosscoders as

a robust framework for systematic model comparison and interpretability research.

References

Aaron Gokaslan*, Vanya Cohen*, E. P. S. T. Openwebtext corpus.

Ameisen, E., Lindsey, J., Pearce, A., Gurnee, W., Turner, N. L., Chen, B., Citro, C., Abrahams, D., Carter, S., Hosmer, B., Marcus, J., Sklar, M., Templeton, A., Bricken, T., McDougall, C., Cunningham, H., Henighan, T., Jermyn, A., Jones, A., Persic, A., Qi, Z., Thompson, T. B., Zimmerman, S., Rivoire, K., Conerly, T., Olah, C., and Batson, J. Circuit tracing: Revealing computational graphs in language models. https://transformer-circuits.pub/2025/attribution-graphs/methods.html, 2025. Accessed: 2025-06-01.

Biderman, S., Schoelkopf, H., Anthony, Q. G., Bradley, H., O'Brien, K., Hallahan, E., Khan, M. A., Purohit, S., Prashanth, U. S., Raff, E., et al. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pp. 2397–2430. PMLR, 2023.

Bricken, T., Nanda, N., Lieberum, E., Conerly, T., Schluntz, A., and Olah, C. Towards monosemanticity: Decomposing language models into understandable components. *Anthropic Interpretability Team*, 2023. Anthropic Blog Post.

Czarnecki, W. M., Osindero, S., Jaderberg, M., Świrszcz, G., and Pascanu, R. Sobolev training for neural networks, 2017. URL https://arxiv.org/abs/1706.04859.

Gao, L., Biderman, S., Black, S., Golding, L., Hoppe, T., Foster, C., Phang, J., He, H., Thite, A., Nabeshima, N., et al. The Pile: An 800GB dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

Han, Z., Gao, C., Liu, J., Zhang, J., and Zhang, S. Q. Parameter-efficient fine-tuning for large models: A comprehensive survey, 2024. URL https://arxiv.org/abs/2403.14608.

Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network, 2015. URL https://arxiv.org/abs/1503.02531.

Kissane, C. crosscoder-model-diff-replication. https://github.com/ckkissane/crosscoder-model-diff-replication, 2024. GitHub repository.

Kissane, C., Conmy, A., and Nanda, N. Open source replication of anthropic's crosscoder paper for model-diffing. LessWrong post, Oct 2024. LessWrong post, October 27, 2024.

Kokane, S., Uddin, M. R., and Xu, M. Improving knowledge distillation in transfer learning with layer-wise learning rates, 2024. URL https://arxiv.org/abs/2407.04871.

Lindsey, J., Templeton, A., Marcus, J., Conerly, T., Batson, J., and Olah, C. Sparse crosscoders for cross-layer features and model diffing. *Transformer Circuits*, 2024. https://transformer-circuits.pub/2024/crosscoders/index.html.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners. 2019.

Romero, A., Ballas, N., Kahou, S. E., Chassang, A., Gatta, C., and Bengio, Y. Fitnets: Hints for thin deep nets, 2015. URL https://arxiv.org/abs/1412.6550.

Sanh, V., Debut, L., Chaumond, J., and Wolf, T. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. In *NeurIPS EMC^{2}Workshop*, 2019.

Sanh, V., Debut, L., Chaumond, J., and Wolf, T. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2020. URL https://arxiv.org/abs/1910.01108.

Shi, W., Li, S., Liang, T., Wan, M., Ma, G., Wang, X., and He, X. Route sparse autoencoder to interpret large language models, 2025. URL https://arxiv.org/abs/2503.08200.

Srinivas, S. and Fleuret, F. Knowledge transfer with Jacobian matching. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4723–4731. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/srinivas18a.html.

Team, G. Gemma. 2024. doi: 10.34740/KAGGLE/M/3301. URL https://www.kaggle.com/m/3301.

Templeton, A., Conerly, T., Marcus, J., Lindsey, J., Batson, J., and Olah, C. Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet. *Transformer Circuits*, 2024. https://transformer-circuits.pub/2024/scaling-monosemanticity/.

Wang, W., Wei, F., Dong, L., Bao, H., Yang, N., and Zhou, M. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers, 2020. URL https://arxiv.org/abs/2002.10957.

Wu, C., Lubana, E. S., Mlodozeniec, B. K., Kirk, R., and Krueger, D. What mechanisms does knowledge distillation distill? In *UniReps: the First Workshop on Unifying Representations in Neural Models*, 2023. URL https://openreview.net/forum?id=uVZB637Dgx.

Xu, X., Li, M., Tao, C., Shen, T., Cheng, R., Li, J., Xu, C., Tao, D., and Zhou, T. A survey on knowledge distillation of large language models, 2024. URL https://arxiv.org/abs/2402.13116.

Yang, X., Liu, W., Liu, W., and Tao, D. A survey on canonical correlation analysis. *IEEE Transactions on Knowledge and Data Engineering*, 33(6):2349–2368, 2019.

Zagoruyko, S. and Komodakis, N. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer, 2017. URL https://arxiv.org/abs/1612.03928.

Zheng, L., Chiang, W.-L., Sheng, Y., Li, T., Zhuang, S., Wu, Z., Zhuang, Y., Li, Z., Lin, Z., Xing, E. P., Gonzalez, J. E., Stoica, I., and Zhang, H. Lmsys-chat-1m: A large-scale real-world llm conversation dataset, 2023.