Analysis of Interpretability of Features with Model Merging – A Visual CrossCoder Approach

July 21, 2025

Anthony Di Pietro (1960447), Tommaso Mattei (1884019), Federico Ziegler (1808702)

Abstract

In deep learning, the study of features is at the root of the subject in terms of importance. An interesting question is about how model merging affects the internal representation of the features in the new model. In the literature, we have seen through the crosscoder how features of a base model and a fine tuned version of it behave after merging. What we are interested in is to analyze the presence of shared, exclusive and new features, to give us insight on performance and interpretability. In particular, we diverged from prior studies by analyzing ResNet and interacting with images, using a visual crosscoder, to learn how features are transferred using different merging techniques.

1. Introduction

In our work we compare at the same time three different versions of the pretrained ResNet50 to do Model Diffing. In particular, we used images from two different datasets: Pokémon and tabletop dice. The datasets were built to solve classification tasks. The visual crosscoder study was done on two fine tuned ResNets, and on the merging of the two models. Beyond that, we also experimented with different merging techniques to analyze their performance and behavior.

2. Related work

The main inspiration for this study is the original paper of the crosscoder (Jack Lindsey, 2023), and a subsequent research on the interpretability (Siddharth Mishra-Sharma, 2025). In the former we see the introduction of crosscoders, a special version of sparse autoencoders geared towards the analysis of models superposition. In the latter, they studied the model diffing of two different models regarding LLMs, comparing a base version and a fine tuned version of the same one.

An important component of the research was the different types of model merging. Some of the observed studies are: (Hao Chen, 2023) which proposes the softMerge method, (Guodong Du, 2024) which uses a score based average and (Mitchell Wortsman, 2022) a more classic averaging of parameters.

3. Method

Our research focused on the study of the activations that were generated through three different versions of a pretrained ResNet50: the first was specialized on a Pokémon dataset, the second one was trained on tabletop dice and the third one which was a merged version of the first two models. We used different merging techniques, such as: interpolation, parameter averaging, and Parameter Competition Balancing (PCB). All the merged techniques were able to retain a lot of information, which resulted in a very high accuracy for both the datasets. In particular, the PCB merging technique performed approximately 5% better than the other techniques.

Our context differed from the previous researches because we mainly wanted to interpret how the features are carried through the merging; this is important because, in the literature, it emerged that model merging might cause the vanishing of features (Xingyu Qu, 2025), therefore a merged model might have dead features that could be removed without impacting negatively the performance, reducing the model's size and complexity. In order to do so, we first have to individuate the aforementioned dead features; this can be done through the study of how the network behaves when it sees meaningful data.

More in detail, we mainly aim to study three different kind of features across the three ResNets: the shared features, those that are shared across every model, the two baseline

Email:
Anthony Di Pietro <dipietro.1960447@studenti.uniroma1.it>,
Tommaso Mattei <mattei.1884019@studenti.uniroma1.it>,
Federico Ziegler <ziegler.1808702@studenti.uniroma1.it>,
Github Repository < Crosscoder-Based-Analysis-Interpretation-of-Model-Merging>.

Deep Learning and Applied AI 2025, Sapienza University of Rome, 2nd semester a.y. 2024/2025.

ResNets' exclusive features, which allowed the networks to respectively specialize their task, and the new features that were born in the merged model. The ResNets were fine-tuned without freezing the weights of any layer, to make the whole network to fully be able to adapt to the new set of data.

The dataset used to train the crosscoders consists of the activations gathered from the fourth block of the network – which is the most significant – where we can find higher-level, semantically rich features. The activations of the three models were collected at inference time by observing a common set of images from three datasets, respectively: Pokemon, dice, and a small portion of ImageNet, which contains random natural images. The latter was introduced to analyze the latent features that were learned through the original training, to allow the common features – if still present – to be activated. To enable the crosscoder to learn an unbiased set of features, we feed the three different activations of the three models at once: this triad constitutes a single datapoint for the crosscoder.

The model used in our research is a crosscoder variant for model diffing: a sparse autoencoder that learns a common set of features. The encoder aims to map the networks' activations to a latent space, and the decoder tries to reconstruct the original input through an ℓ_2 loss – the same behavior that we encounter in any autoencoder. The main distinctive characteristic is that sparsity is enforced to allow the model to distinguish exclusive features of the three networks; this is enacted using an ℓ_1 penalty for the sums of the decoder norms across models separately in the loss function.

In our studies, we trained three different crosscoders. All of them were fed with the activations generated from the two baselines ResNets finetuned with Pokémon and tabletop dice respectively, and the third model created from one of the merging techniques: interpolation, parameter averaging, and PCB.

4. Results

Our core focus was centered around model diffing of the three models in order to analyze the features that emerged from training the crosscoder. Our downstream task is to identify which features are shared and which are exclusives across the three models. We start by analyzing the decoder weights relative norms of the models two by two: the ones that have similar decoder magnitude influence the models equally. On the other hand, the model-exclusive features are stronger to one model rather than the others. We have also compared the results of different merging techniques, but to summarize our results, hereby we focus on the comparison between the interpolated and PCB merging techniques, as the parameter averaging results were very similar to the interpolation model.

On Figure 1 we can see the relative decoder norms of the crosscoder that focused on the interpolated merging technique. On Figure 2 we show the relative decoder norms of the crosscoder trained with PCB. The main difference is that in the PCB setup we are able to identify more exclusive features. Thereby, we can conclude that the merging technique is able to extract more meaningful features without carrying on redundant ones. This conjecture comes from the fact that the accuracy of the upstream ResNet is better, but the shared features are less, therefore those features were already wired in the network and not really meaningful.

We also noticed that there are more dice-related exclusive features across all the crosscoders. This is a really interesting behavior, and our hypothesis is that the upstream dice-focused ResNet trained on a bigger dataset with respect to the Pokémon one, so it had more time to adapt its parameters to the new task, resulting in a more specialized network.

On Figure 3, we show how the cosine similarity across the three unordered pairs of models behaves. This enforces our previous conjecture; the dice-focused network diverged more from the original one, as its cosine similarity is lower with respect to the other two.

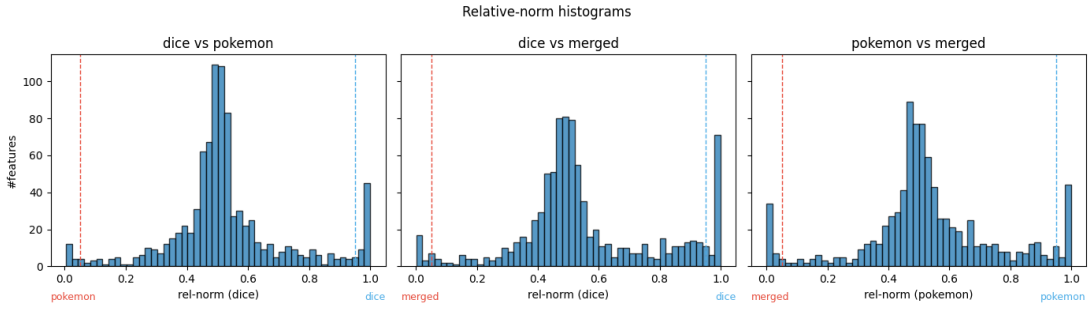

Figure 1. Relative norms of the decoder weights using the interpolation-merging technique.

In this sense, the interpolated model seems to favor more the Pokémon network in terms of representation in the latent space.

On the other hand, we don't experience the same behavior on Figure 4, the crosscoder trained with the PCB-generated network. The gaussian-like results explain that this merged model is more complex to explain and it differs more from the other two, therefore the decoder struggles to represent the three networks similarly in its latent space.

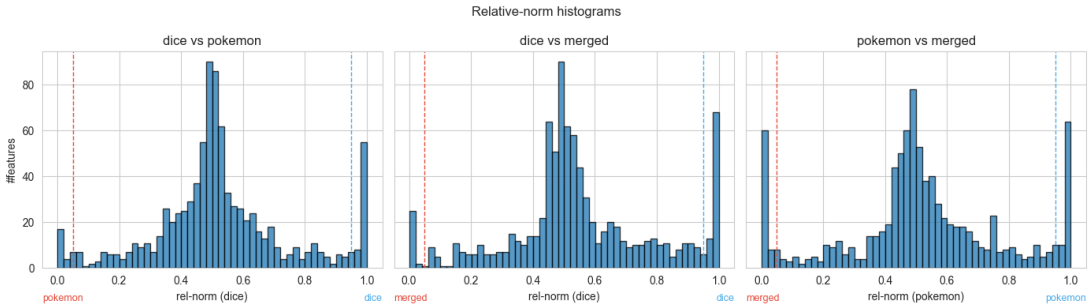

Figure 2. Relative norms of the decoder weights using the PCB-merging technique.

9

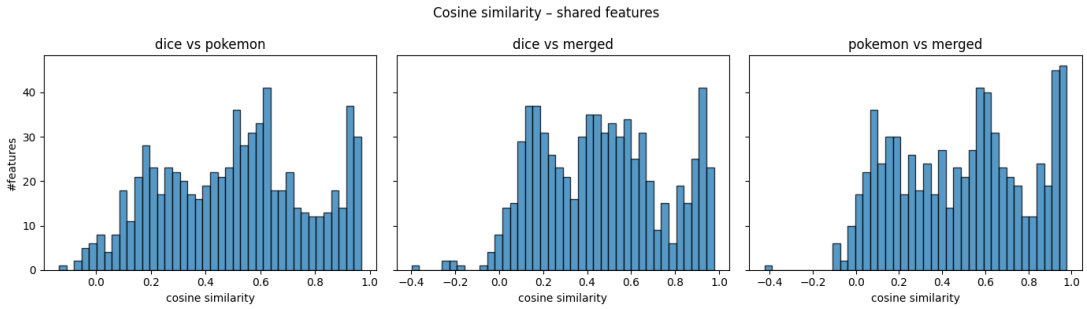

Figure 3. Cosine similarity of the decoder vectors across the three models with the interpolation-merging technique.

9

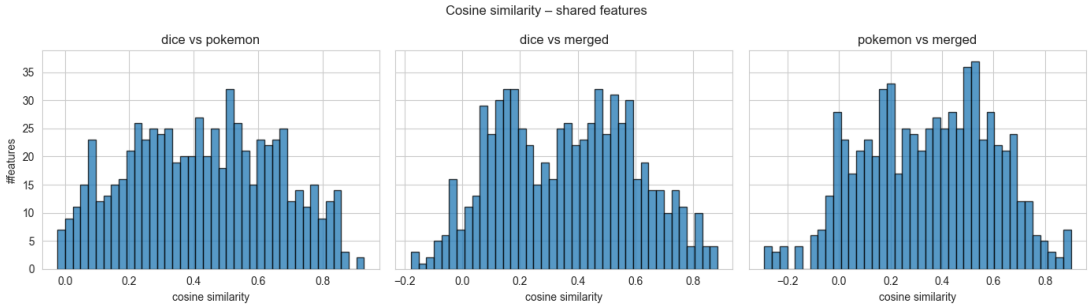

Figure 4. Cosine similarity of the decoder vectors across the three models with the PCB-merging technique.

5. Discussion and conclusions

It would be interesting to study the latent space of a crosscoder if it were a bidimensional vector, where each cell represented the latent space of a set of activations generated on a particular level of the network. Through this research, links between features at different levels could emerge, leading to a more-detailed analysis and expressive feature interpretation.

6. Appendix

6.1. Computational Complexity

The computational complexity to express the activations of the fourth block across the three different networks is really huge. We trained our crosscoders with a last-generation GPU with a total VRAM capacity of 16GBs (NVIDIA RTX 5080), yet we were forced to use a latent space of around 900 to avoid crashes, even though we wanted to make some experiments with a bigger latent space.

For the same reason, we were not able to feed to the crosscoders several blocks at the same time; the number of activations for the early blocks were even more, up to 300k, compared to the fourth block's 100k. However, we were happy to focus on the last block because it is the most significative.

6.2. Crosscoder Sensibility

We found an LLM-based crosscoder implementation on GitHub that we used as a general reference, however during the first trainings of the crosscoder, we tried different weights initializations to see how they impacted the downstream results.

We noticed pretty early that different initializations, such as `kaiming-normal` instead of the Github's `normal`, led to very different results. We played a bit with the initializations, but we empirically found that the Github's ones seemed more expressive.

On a similar note, we also tried to use a different activation function: PReLU, which is very similar to ReLU but does not completely inhibit the negative signals through our network. PReLU squishes the signals through a learned parameter; we have chosen this particular activation function to see how the network behaved if we let pass a tiny fraction of negative values. However, in this setting, the results were skewed: almost every feature was considered as shared.

We also tried different loss functions. Even though we gathered pretty good results with some of them, especially by moving the computation of the ℓ_2 loss before the activation function, in the end we stayed with the original idea to make the network learn through a non-linear function. We did so to avoid a linear map between the encoder's weights and the loss.

The hyperparameter search was very tedious: the network, beyond the latent space and the number of epochs, uses a λ coefficient to tune its sparseness. If we don't enforce too much sparseness, the crosscoder would not be able to distinguish between the three different models, resulting in everything being labeled as shared. On the other hand, if we assign to λ a value that is too strong, it degrades the reconstructions of the decoder, therefore the crosscoder is not able to mimic closely the original activations.

All the aforementioned characteristics of the training of the crosscoder made us realize how much sensitive it is to little variations of the architecture, initializations, and hyperparameters.

6.3. More Results

In this subsection, we show other results that we were not able to show in the previous sections.

Since our problem lies in a 3-dimensional space, we tried to plot the relative norms in 3D, where each axis represents the relative norm of a particular ResNet model. The results were pretty interesting and we were able to notice something really important. On Figure 6, we find a tridimensional representation of the relative decoder norms with the PCB-based crosscoder. In this plot we were able to show how the pair-wise exclusive features behaved, so features that are in common between two models but not with the third.

Our hypothesis, initially, was that very few features would have been exclusive to the Pokémon and dice pair. However, as we can see in the figure – and we experienced a similar behavior with PCB and average-based crosscoders – this is not the case, instead between the three pairs it presented the largest number of exclusive features. This left us puzzled, but after a careful analysis and research of the literature, we concluded that those features might actually be the vanishing features that come from the downstream merge (Xingyu Qu, 2025), which means we could remove them without impacting the merged-model performance.

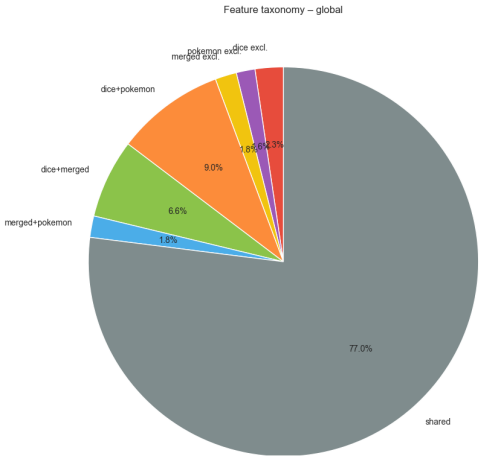

Figure 5. Summary of the exclusive, pair-exclusive and shared features of the PCB-based crosscoder.

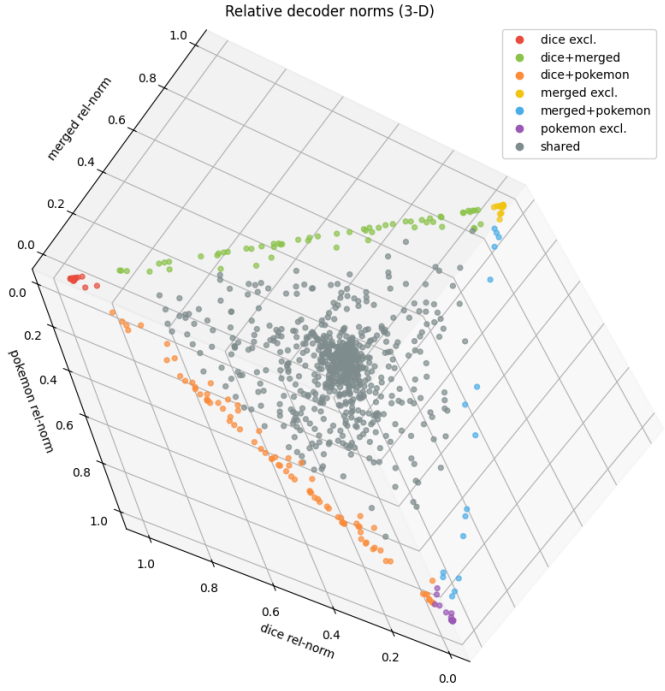

Figure 6. Relative norms of the decoder weights using the PCB technique in 3D.

Lastly, as we mentioned in the previous sections, we didn't focus on the outcome that came from the Parameter Averaging merging technique. We found it almost identical to the ones that were led by the interpolation technique, but those were results nevertheless, henceforth we present them now in Figure 7 and in Figure 8.

6.4. About Density

We wanted to replicate Anthropic's results by plotting the feature densities of the shared and model-exclusive features. Their plot showed that the exclusive features were poly-semantic, firing with more probability than the shared ones. However, in our context, the plot showed very different results that we were not really able to explain further. Our density can be viewed in Figure 9 for the PCB-based crosscoder.

In earlier iterations we actually got a good result on the density, which is shown in Figure 10. However, in that experiment, the ℓ_2 term of the loss function did not pass through the non-linear function ReLU. We preferred to have a more expressive encoder rather than a better density plot.

Bibliography. (Siddharth Mishra-Sharma, 2025). (Hao Chen, 2023). (Guodong Du, 2024) (Mitchell Wortsman, 2022) (Xingyu Qu, 2025)

References

Guodong Du, Junlin Lee, J. L. R. J. Y. G. S. Y. H. L. S. K. G. H.-K. D. H. M. Parameter competition balancing for model merging. *Advances in Neural Information Processing Systems 37 (NeurIPS 2024) Main Conference Track*, 37:84746–84776, 2024.

Hao Chen, Yusen Wu, P. N. C. L. Y. Y. Soft merging: A flexible and robust soft model merging approach for enhanced neural network performance. *arXiv:2309.12259*, 2023.

Jack Lindsey, Adly Templeton, J. M. T. C. J. B. C. O. Sparse crosscoders for cross-layer features and model diffing. *Transformer Circuits Thread*, 2023.

Mitchell Wortsman, Gabriel Ilharco, S. Y. G. R. R. R. G.-L. A. S. M. H. N. A. F. Y. C. S. K. L. S. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. *Proceedings of Machine Learning Research*, 39th edition(PMLR 162:23965-23998), 2022.

Siddharth Mishra-Sharma, Trenton Bricken, J. L. A. J. J. M.-K. R. C. O. T. H. Insights on crosscoder model diffing. *Transformer Circuits Thread*, 2025.

Xingyu Qu, S. H. Vanishing feature: Diagnosing model merging and beyond. *https://arxiv.org/abs/2402.05966*, 2025.

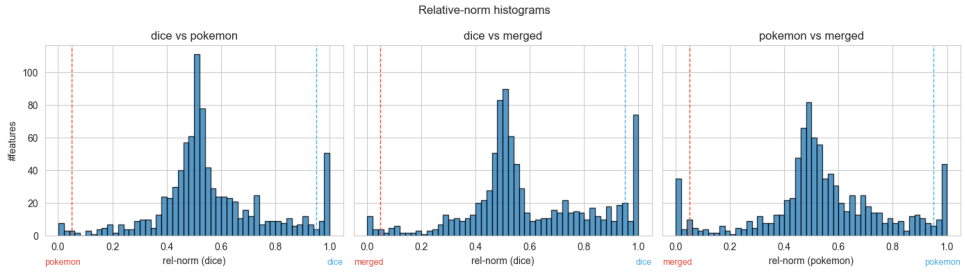

Figure 7. Relative norms of the decoder weights using the parameter averaging technique in 3D.

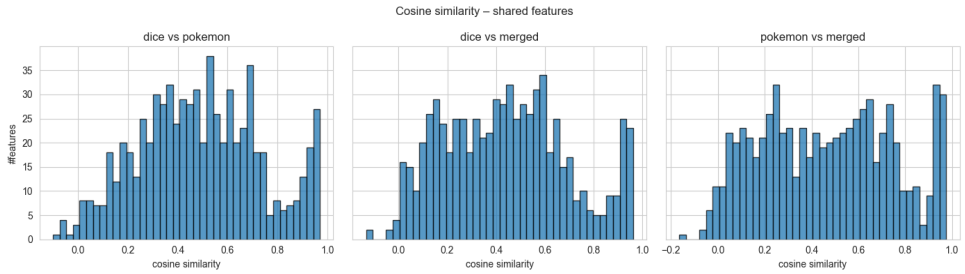

Figure 8. Cosine similarity of the decoder vectors across the three models with the parameter averaging merging technique.

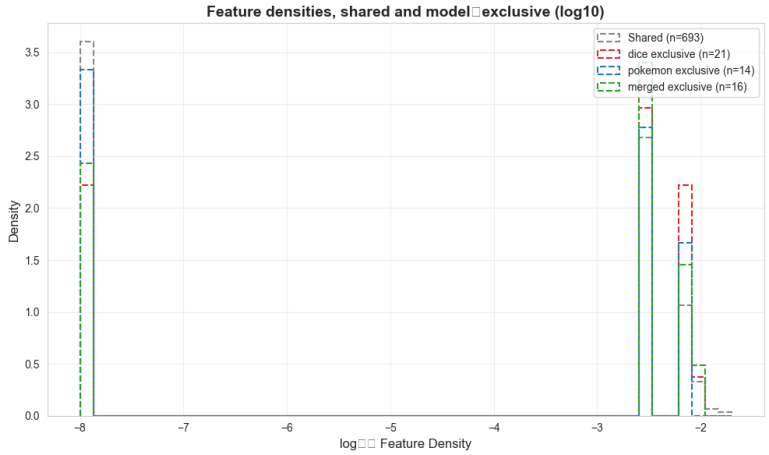

Figure 9. The PCB crosscoder-based density plot, shared and model-exclusive.

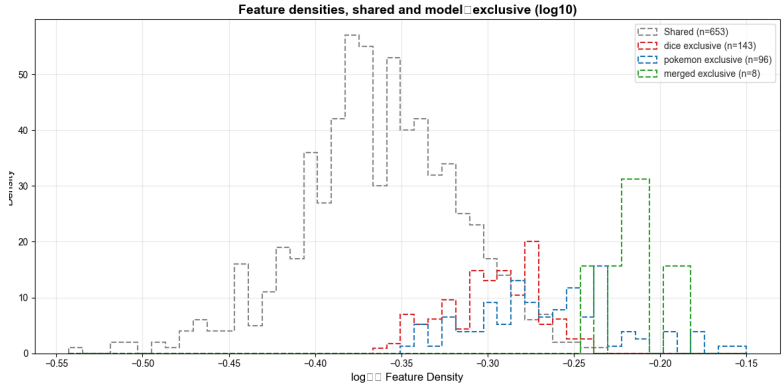

Figure 10. The interpolated-based density in one of the earlier experiments.