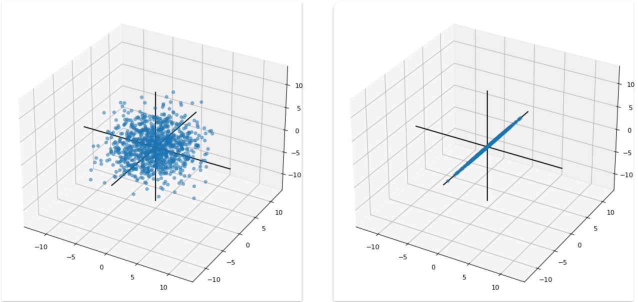
Discovering Functionally Sufficient Projections with Functional Component Analysis

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

Many neural interpretability methods attempt to decompose Neural Network (NN) activity into vector directions or features along which variability serves to represent some interpretable aspect of how the NN performs its computations. In correlative analyses, these features can be used to classify what inputs and outputs correlate with changes in the feature; in casual analyses, these features can be used to causally influence computation and behavior. In both cases, it is easy to view these features as satisfying as ways to interpret NN activity. What if each feature, however, is an incomplete part of the story? For any given feature, is it necessary for the NN's computations, or is it only sufficient? In this work, we present a method for isolating Functionally Sufficient Projections (FSPs) in NN latent vectors, and we use a synthetic case study on MultiLayer Perceptrons (MLPs) to find that multiple, mutually orthogonal FSPs can produce the same behavior. We use the results of this work as a cautionary tale about claims of neural necessity.

1 Introduction



Full Neural Activity

1D Functionally Sufficient Pathway

Figure 1: A hypothetical diagram showing the distinction between the complete neural activity for a latent vector size of 3 on the left and the activity projected onto a 1 dimensional FSP on the right. The axes represent dimensions in the latent vector, and each point is in instance of the NN's latent activations. If we assume in this contrived, pedagogical example that the NN exhibits the same behavior using only the projected activity on the right, then the projected axis serves as an example of an FSP. Note that FSPs do not, in general, lie along activation axes.

Many recent advances in Neural Network (NN) interpretability have proposed and used methods that rely on representational features, where features are defined as directions in vector space along which variation encodes or correlates with some interpretable attribute in the input or output of the NN

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

[4, 13, 3, 6, 2, 17, 16, 9]. Features are useful in that they can offer explanatory power over NNs, and in causal cases, they offer interpretable control over the NN's computations and behavior. NN control is useful both as a tool for AI safety, and for making causal claims about NN activity [15, 8, 7]. An easily overlooked aspect of NN features, however, is that they often are used in such a way so as to demonstrate sufficiency of a claim, without the ability to demonstrate necessity.

Is it possible to learn multiple features that explain/produce the same effect? Is any given feature both *necessary and sufficient* for its interpretable effect? In other words, is it possible to learn orthogonal features that produce the same causal effects? In this work, we explore these questions in a case study that uses Multi Layer Perceptrons (MLPs) trained on a synthetic hierarchical boolean task. We first offer a method for isolating functional subspaces in the native neural space whose span is sufficient to reproduce the MLP's complete behavior. We refer to the method for finding these subspaces as Functional Component Analysis (FCA), and we refer to these subspace bottlenecks as Functionally Sufficient Projections (FSPs). We then show that multiple, distinct, mutually orthogonal FSPs can produce the same NN behavior, demonstrating that individual FSPs are only sufficient for functionality without being necessary. We then explore how model dimensionality leads to a greater number of possible FSPs and assists in finding FSPs with fewer dimensions. Lastly, we examine the FSP created using the top Principal Components (PCs) that explain the most variance in descending order as a means of grounding our intuitions. Taken together, we use our findings as a cautionary tale about the dangers of assuming necessity when only sufficiency is demonstrated.

2 Methods

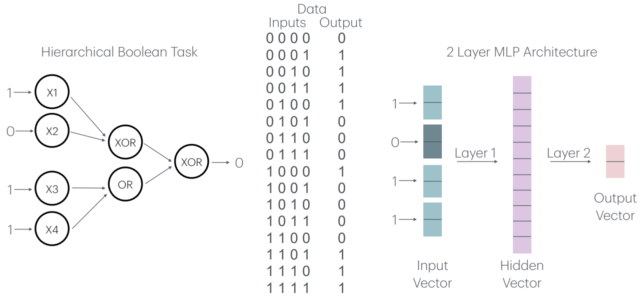
In this work we first train Multi-Layer Perceptrons (MLPs) to perform a hierarchical boolean task to 100% accuracy. We then freeze the MLP weights and attempt to isolate Functionally Sufficient Pathways (FSPs) which we define as vector subspaces in which all behaviorally relevant activity is encoded.

2.1 Hierarchical Boolean Task

The task consists of a series of boolean operations that build on the outputs of the previous layer shown in Figure 2. The inputs to the task are four binary values. The task consists of independently performing the XOR operation over inputs 1 and 2 and OR over inputs 3 and 4; then the task uses the outputs of both of these operations as the inputs to a final XOR operation for a binary output. These specific operations were chosen by randomly generating multiple candidate tasks and selecting one with an approximately equal number of 0 and 1 output values.

2.2 Multi-Layer Perceptrons (MLPs)

The models used in this work consist of an input embedding layer of 2 dimensions with 2 possible inputs followed by a layer normalization [1] with default PyTorch settings, a linear weight matrix $W_1 \in \mathbb{R}^{d \times 8}$ assuming the activations are column vectors and d is a hyperparameter defining the



Data

Inputs	Output
0 0 0 0	0
0 0 0 1	1
0 0 1 0	1
0 0 1 1	1
0 1 0 0	1
0 1 0 1	0
0 1 1 0	0
0 1 1 1	0
1 0 0 0	1
1 0 0 1	0
1 0 1 0	0
1 0 1 1	0
1 1 0 0	0
1 1 0 1	1
1 1 1 0	1
1 1 1 1	1

Figure 2: Visual depiction of the hierarchical boolean task and the model architecture. The inputs and corresponding outputs of the task are enumerated in the table. The model architecture consists of either 2 or 3 layers resulting in 1 or 2 hidden state layers. All FCA experiments are performed on the 1st hidden layer.

model's hidden dimensionality. The outputs of this layer are the focus of all analyses. After this layer, a ReLU nonlinearity is applied, followed by another layer norm and a linear weight matrix. For two layer models, this weight matrix is $W_2 \in \mathbb{R}^{d\times 2}$, and it produces the output predictions. The 3 layer models, however, use $W_2 \in \mathbb{R}^{d\times d}$ followed by a ReLU, layer norm, and a final output weight matrix, $W_3 \in \mathbb{R}^{d\times 2}$ to obtain the model outputs.

2.3 Functional Component Analysis and Functionally Sufficient Projections

In this section, we introduce Functional Component Analysis (FCA) which is an algorithm for finding Functionally Sufficient Projections (FSPs) which are defined as subspaces whose span is sufficient to produce the model's original behavior. Formally, for a prespecified NN layer, an FSP is defined as a matrix $U \in \mathbb{R}^{d\times n}$ with n orthogonal columns whose span is sufficient to produce some functionality of the NN. In this work, we only consider FSPs that reproduce the NN's complete behavior. Specifically, NN latent vectors, $h \in \mathbb{R}^d$, at the predefined layer are projected onto the span of the FSP using the following equation:

$$h_{fsp} = UU^\top h \tag{1}$$

After the projection, the NN continues its processing, using h_{fsp} in place of h.

U is defined and trained using the FCA algorithm as follows. At the start of training, U is constructed by first sampling an initial set of vectors from a normal distribution and storing these as trainable parameters. Then, before each forward pass of the NN (whose weights are frozen), the trainable vectors are orthogonalized and normalized using the Gram-Schmidt process detailed in Appendix A.2. Then Equation 1 is applied to the NN's latent vectors at the specified layer during its forward pass, and the model's processing continues from the interrupted layer now using h_{fsp}. U is then trained via gradient descent to minimize a cross entropy loss that uses the NN's predictions conditioned on h_{fsp} and the NN's original behavior as labels. Upon convergence of the training loss, if the model's FSP accuracy is below threshold (100% in our case), the algorithm adds a new vector to the trainable set and repeats the training process. The precise algorithm is detailed in Algorithm 1.

3 Results

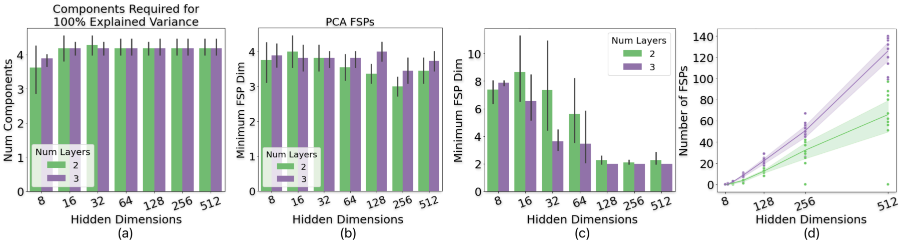


Figure 3: Hidden Dimensions on the x-axis refers to the models' latent dimensionality and the color indicates the number of layers in the analyzed MLP. **(a)** Bar graphs showing the average number of principal components (PCs) to achieve 100% explained neural variance. **(b)** the minimum number of PCs needed to reproduce the models' full behavior. We see that typically, fewer components are needed to reproduce behavior than components needed to explain 100% of the neural variance. **(c)** Bar plots show the minimum FSP dimenionality out of the set of FSPs for a given model. We see that larger models can have FSPs consisting of as few as 2 dimensions. We also see an inverse relationship between model dimensionality and min FSP dimensionality. However, this relationship is complicated by the imperfect FCA training technique as demonstrated by the PCA FSP results. **(d)** A scatter plot of the total number of orthogonal FSPs for each model seed. Each data point represents a single model seed. We see an approximately linear relationship between the number of FSPs and model dimensionality.

3.1 Larger model dimensionality results in more FSPs and smaller minimum FSP sizes

Turning to panels (c) and (d) of Figure 3, we see that we can successfully isolate multiple FSPs for individual models. We can see this in panel (d) which shows that the number of FSPs scales approximately linearly with the model dimensionality. Noteably, the 3 layer models appear to have an increased rate of FSPs relative to the 2 layer models. This section serves as a proof of principle, that multiple functionally equivalent subspaces can exist in a single MLP. Depending on how these results extrapolate to more complex experimental settings, this could raise potential issues with methods that only demonstrate functional sufficiency without attempting to address functional necessity. This is potentially very important in cases of AI safety.

We now turn to Figure 3(c) which shows that the minimum FSP size achieved for each model tends to diminish with increasing model size, where some models only need 2 dimensions to reproduce their behavior. This result demonstrates a potentially counter intuitive finding, that bigger models can use fewer dimensions in their computations for the same result. Although, this is potentially to be expected in light of the Lottery Ticket Hypothesis [5]. We also note that this could be a result of FCA being an imperfect training algorithm, where smaller models do possess 2D FSPs, but FCA fails to find them.

3.2 Functional sufficiency is achieved without explaining 100% of the neural variance

We show in Figure 3(a) that the number of principal components (PCs) to explain 100% of the variance in the neural activity remains relatively constant with different model dimensionalities. Interestingly, the average number of PCs required to achieve 100% accuracy, shown in panel (b), is slightly less than the number of PCs for 100% explained variance. Differences such as these highlight the difference between behaviorally relevant and behaviorally null subspaces in the representations [10].

4 Limitations/Future Directions

An important limitation of the claims that can be made using FCA is that the method lacks guarantees on isolating the smallest functionally sufficient set of dimensions due to the fact that the FSPs are learned through gradient descent and randomly sampled vectors in the Gram-Schmidt process. This limits the claims that can be made about differences in FSP minimum sizes as a function of NN scale as the resulting FSP size can contain functionally unnecessary dimensions as demonstrated by PCA on models with a hidden dimensionality of 8.

Another large limitation is that this work was performed on simplistic models trained to perform a simplistic task with only two behavioral outputs. And, due to the small data size, we performed both training and analysis on the same set of data. It is unclear how this work will translate to larger highly distributed NNs, and it is unclear how these analyses would generalize to unseen examples.

5 Discussion/Conclusion

In this work we explored how orthogonal subspaces in MLP processing layers can equivalently produce the same NN behavior. We showed that fewer PCs than are required to explain 100% of the neural variance can create an FSP, and we showed that the number of FSPs for a given layer tends to scale approximately linearly. What do these findings mean for the NN interpretability?

Although it is unclear from our experiments to what degree these results hold in more complex settings, if they do hold, the implications can be quite significant. For instance, in cases of AI safety, it can be important to isolate all activity that corresponds to a specific safety-related feature. If it is possible to learn multiple orthogonal subspaces for the same feature, it may be hopeless to excise such a feature from the model's computations. Furthermore, in cases of features learned by methods such as SAEs or transcoders [11, 12], it is possible that vector directions that are orthogonal to any given feature can activate the same attributes as that feature. This raises questions about the utility of such learned features for robustly addressing AI safety concerns.

We conclude by suggesting caution when making claims about how NNs solve particular tasks. With the highly redundant nature of NN latent vectors, it is generally possible that any given NN employs multiple orthogonal computations to solve any given problem. Thus, we caution interpretability researchers to be wary of any claims of necessity.

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016.

[2] Trenton Bricken, Neel Nanda, Nicholas Joseph, Arthur Conmy, Andy Jones, Anna Chen, Neal DasSarma, Nelson Elhage, Ben Mann, Catherine Olsson, Kamal Ndousse, Sam Ringer, Alex Tran-Johnson, Yuntao Bai, Liane Lovitt, Zac Hatfield-Dodds, Amanda Askell, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, and Sam McCandlish. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023.

[3] 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.

[4] 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.

[5] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks, 2019.

[6] Leo Gao, Tom Dupré la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. Scaling and evaluating sparse autoencoders. *arXiv preprint arXiv:2406.04093*, 2024.

[7] Atticus Geiger, Jacqueline Harding, and Thomas Icard. How causal abstraction underpins computational explanation. *arXiv preprint arXiv:2508.11214*, 2025.

[8] Atticus Geiger, Duligur Ibeling, Amir Zur, Maheep Chaudhary, Sonakshi Chauhan, Jing Huang, Aryaman Arora, Zhengxuan Wu, Noah Goodman, Christopher Potts, and Thomas Icard. Causal abstraction: A theoretical foundation for mechanistic interpretability, 2024.

[9] Atticus Geiger, Kyle Richardson, and Christopher Potts. Neural natural language inference models partially embed theories of lexical entailment and negation. *arXiv preprint arXiv:2004.14623*, 2020.

[10] Andrew Kyle Lampinen, Stephanie CY Chan, Yuxuan Li, and Katherine Hermann. Representation biases: will we achieve complete understanding by analyzing representations? *arXiv preprint arXiv:2507.22216*, 2025.

[11] Jack Lindsey, Wes Gurnee, Emmanuel Ameisen, Brian Chen, Adam Pearce, Nicholas L. Turner, Craig Citro, David Abrahams, Shan Carter, Basil Hosmer, Jonathan Marcus, Michael Sklar, Adly Templeton, Trenton Bricken, Callum McDougall, Hoagy Cunningham, Thomas Henighan, Adam Jermyn, Andy Jones, Andrew Persic, Zhenyi Qi, T. Ben Thompson, Sam Zimmerman, Kelley Rivoire, Thomas Conerly, Chris Olah, and Joshua Batson. On the biology of a large language model. *Transformer Circuits Thread*, 2025.

[12] Samuel Marks, Adly Templeton, Jacob Dunefsky, et al. Transcoders find interpretable llm feature circuits. *OpenReview*, 2024.

[13] Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability, 2023.

[14] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. *CoRR*, abs/1912.01703, 2019.

[15] Judea Pearl. An Introduction to Causal Inference. *The International Journal of Biostatistics*, 6(2):7, February 2010.

[16] Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Simas Sakenis, Jason Huang, Yaron Singer, and Stuart Shieber. Causal mediation analysis for interpreting neural nlp: The case of gender bias. *arXiv preprint arXiv:2004.12265*, 2020.

[17] Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: a circuit for indirect object identification in gpt-2 small, 2022.

Algorithm 1 Functional Component Analysis (FCA)

```
Require: Frozen neural network $\mathcal{M}$, target layer index $l$, projection accuracy threshold $\tau$ (e.g., 100%)
Initialize trainable matrix $U \in \mathbb{R}^{d \times n}$ with $n = 1$, entries from $\mathcal{N}(0, 1)$
repeat
    for all input batches $x$ do
        Orthogonalize and normalize the columns of $U$ via Gram-Schmidt (see Appendix A.2)
        Run forward pass on $\mathcal{M}$ to obtain original output $y = \mathcal{M}(x)$
        Compute hidden state $h = \mathcal{M}_l(x)$ at layer $l$
        Project onto the FSP subspace: $h_{\text{fsp}} \leftarrow UU^\top h$
        Replace $h$ with $h_{\text{fsp}}$ and resume forward pass to get prediction $\hat{y}$
        Compute loss $\mathcal{L}(U) = \text{CrossEntropy}(\hat{y}, y)$
        Update $U$ using gradient descent to minimize $\mathcal{L}(U)$
    end for
until convergence of training loss
if FSP accuracy $< \tau$ then
    Add new trainable column to $U$ (sampled from $\mathcal{N}(0, 1)$), increment $n$, and repeat
end if
```

A Appendix

A.1 Model Details

All artificial neural network models were implemented and trained using PyTorch [14] on Nvidia Titan X GPUs. We used a learning rate of 0.0001 and a batch size of 128 consisting of randomly sampled examples with replacement. Trainings continued for 140 epochs.

A.2 Gram-Schmidt Process

To progressively grow the size of the FSP through iterative training, we can introduce and orthogonalize new vectors using the Gram-Schmidt process as follows. Assume that the current FSP consists of n column vectors $\mathrm{v}_k \in R^d \, \forall k \in \{1, 2, ..., n\}$ that have been sampled from a normal distribution where d is the latent dimensionality of h. We can then make an orthogonal set of vectors $\mathrm{u}_k \in R^d$, and thus a partial orthogonal matrix $U \in R^{d \times n}$, by subtracting out the projections of all preceding components from each consecutive v_i where $\mathrm{u}_1 = \mathrm{v}_1$:

$$\text{proj}_u(\mathrm{v}) = \frac{\mathrm{u}^\top \mathrm{v}}{\mathrm{u}^\top \mathrm{u}} \mathrm{u} \tag{2}$$

$$\mathrm{u}_k = \mathrm{v}_k - \sum_{i=1}^{k-1} \text{proj}_{u_i}(\mathrm{v}_k) \tag{3}$$

$$U = \begin{bmatrix} \frac{u_1}{||u_1||} & \frac{u_2}{||u_2||} & \cdots & \frac{u_n}{||u_n||} \end{bmatrix} \tag{4}$$

We can then project onto the span of U with the following: $h_{fsp} = UU^\top h$.