Emerging Structures in Computational Graphs of Neural Networks

Grégoire DHIMOÏLA
ENS Paris-Saclay
gregoire.dhimoila@ens-paris-saclay.fr

March - August 2024

Abstract

While neural networks form the basis of modern machine learning, their inner workings are still far from well understood. To comprehend neural networks' (NN) behaviors and ensure their safe and fair application, a key direction is to measure and interpret their internal mechanisms. The field of mechanistic interpretability aims to explain these internal mechanisms. One of its main framework is *circuit discovery*, which attempts to find and interpret subsets of computational nodes responsible for certain tasks or functionalities.

Mechanistic analyses of NNs commonly involve neurons, which are often polysemantic and hard to interpret, or *features* that disentangle neurons. Neurons and features are fine-grained units of analysis in that they provide a detailed level of information. In this work, we introduce regions of interest (ROIs) as sets of functionally similar features. These can be used as coarse units for low granularity studies of computational graphs. We show the functional relevance of these ROIs.

We then propose a novel algorithm that finds computational graphs through causal relationships between nodes. This algorithm generalises several previous work on circuit discovery. It is general in that it can accommodate nodes of arbitrary granularity, from features to attention heads or ROIs. We systematically compare desired functional and structural properties of these graphs.

Contents

1 Introduction

Developing tools to study the internal mechanisms of neural networks (NN) is key in the understanding of their internal representations, world models and what algorithms they can implement. These insights help identify and prevent potential biases [Marks et al., 2024] and lead to the design of interventions to predictably steer the model's behavior [Turner et al., 2023, Rimsky et al., 2023].

The field of *mechanistic interpretability* (MI) [Bereska and Gavves, 2024] aspires to explain these internal mechanisms. This usually results either in the attribution of some function or behavior to a specific computational units [Alain and Bengio, 2017, Mallen and Belrose, 2023, Marks and Tegmark, 2023, Lu and Ester, 2019] or to a subgraph of the network formed by these units [Conmy et al., 2023, Merullo et al., 2024]. We see in Section 3.6 that both of these approaches have historical equivalents in the field of neurosciences.

The current MI framework relies heavily on sparse coding to learn disentangled and more interpretable *features*, encoding concepts and generating representations. Features in later layers can be viewed as functions of earlier features, and the model can then be thought of as a large composition of these simple functions. When conducting mechanistic analyses in language models, it is common practice to use these features as units of analysis for detailed - fine-grained - information [Marks et al., 2024], or to use attention heads for coarser insights or level of detail [Conmy et al., 2023, Syed et al., 2023].

Circuit discovery in MI aims to find computational subgraphs responsible for some selected task or behavior [Conmy et al., 2023, Syed et al., 2023, O'Neill and Bui, 2024, He et al., 2024]. It mostly focuses on finding nodes that best explain a behavior, and on providing manual interpretations of them [Marks et al., 2024, Conmy et al., 2023]. Additionally considering the importance of edges can lead to sparser graphs especially if the number of relevant nodes is large (see Section 3.5.3). This can be fine for task specific studies where the number of nodes involved in the task is significantly smaller than the total number of nodes.

However, if we want to do more general studies of the computations of the model, these approaches would basically leave the computational graph unaltered as most nodes are likely to be relevant in at least one context. For this reason, we believe that circuit discovery studies should evaluate actual subgraphs through edge ablation rather than focusing on the subsets of nodes fully connected through node ablation (Section 3.5.3).

The purpose of this study is to construct sparse computational graphs faithful to the original model's behavior and investigate their structures. This work is organised as follows. In Section 2, we give general comments on the internship. In Section 3, we provide some background on notions from neurosciences[1] (Section 3.2), network sciences (Section 3.3) and MI (Sections 3.4 and 3.5). These notions are the basis upon which we build our discussions and contributions. In Section 3.6, we draw parallels between the way neurosciences and MI explain complex systems through the analysis of underlying networks or structures. Following sections contain our contributions.

We introduce in Section 4 a novel algorithm to partition features by functional equivalence. We call the resulting classes *regions of interest* (ROIs). ROIs give coarse units of analysis for any further mechanistic study of the model. Typically, while circuit discovery often uses attention heads in transformers as low granularity[2] nodes, this approach may be suboptimal as these units are often polysemantic and hard to interpret outside of specific contexts. Moreover, there is no equivalent in MLP layers. We provide a set of experiments to validate the relevance of these ROIs. We did not yet compare the use of ROIs and attention heads in mechanistic studies, which is left for future work.

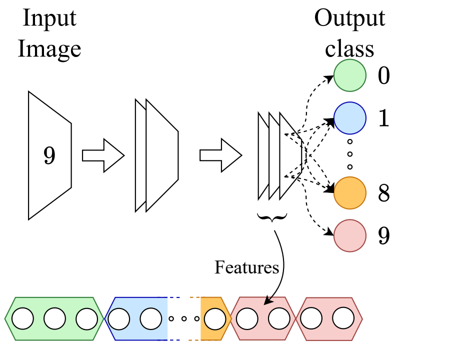


Figure 1: Illustration of ROIs in a the feature extractor of a simple CNN. Features are sorted by ROI. See Section 4.2.

[1]Due to the large overlap between the means of study of brain networks (Section 3.2) and artificial neural networks (Section 3.5), we try to use terminology from the field of neurosciences wherever relevant.

[2]Granularity refers to the level of detail and information captured by nodes

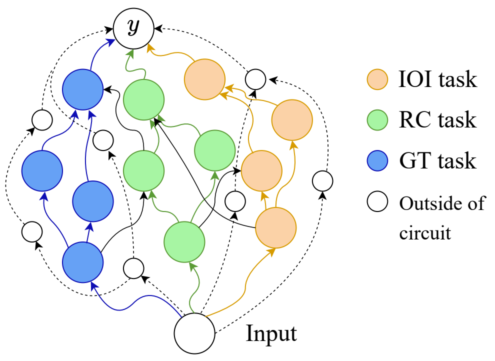


Figure 2: Illustration of a computational graph for a mixture of three different language tasks from Section 5.2.

Section 5 describes a general class of algorithms using attribution to construct a computational graph. It can accommodate several levels of granularity. For example, when working with fine-grained features as nodes, the resulting algorithm is similar to the one introduced by Marks et al. [2024], though some details for edge attribution are different. When using coarser nodes like attention heads, it simplifies to the algorithm from Syed et al. [2023] with a different attribution method - integrated gradients instead of raw gradients (see Section 3.5.1). In this section, we also provide some preliminary results on the properties of these graphs.

Contribution. We propose to study emerging structures in computational graphs to improve and automate MI, using both *functional* and *causal*[3] relationships between nodes.

- We define new low granularity nodes for the study of neural networks as groups of *functionally* similar fine-grained nodes[4] (Section 4) as defined by *Stochastic Blockmodels* (SBM) (Section 3.3). We call these *regions of interest* (ROIs).
- We establish the relevance of these ROIs through interventions and predictable behavioral changes in the model (Section 4.2) and show that ROIs given by SBMs are functionally more relevant than those given by community detection algorithms commonly used in the literature (see Lu and Ester [2019], Filan et al. [2021], Bushnaq et al. [2024]).
- We define a novel algorithm to construct computational graphs in neural networks that can incorporate several levels of abstraction (Section 5.1) along with methods to study these graphs.
- We show the relevance of these methods by demonstrating the functional equivalence between the original model and the one induced by our computational graph (Section 5.2), as well as intervening on this graph with predictable effects on the resulting model's behavior. Our results outperform those of previous research.

In Section 6 we discuss shortcomings of the current study and directions for future work.

2 Meta Information

This section provides an overview of the meta aspects of my internship, including supervision, team environment, key events and a rough timeline.

A publication relative to the work done during this internship is currently planned.

2.1 Supervision and Environment

During my internship, I received substantial support from my supervisor - Prof. Holger Dell - who was consistently available and engaged. We had discussions basically on a daily basis, allowing for continuous feedback and guidance. He organized meetings with various teams, enabling me to present my work, receive constructive feedback, and explore potential collaborations.

Overall, the lab was welcoming and open to discussion throughout my internship. I was able to gain understanding from the inside of how a lab operates on both teaching and research aspects.

[3]This would be referred to as *effective* connectivity in neuroscience

[4]Fine-grained nodes can be anything from canonical dimensions - also known as neurons - to sparse features through any change of basis relevant to the model internal geomerty.

2.2 Events

Throughout the internship, I participated in a range of events. These included seminars and meetings with both my team and other teams on a regular basis. These gatherings provided opportunities to present our work, brainstorm new ideas, and explore potential collaborations.

Noteworthy events included the HAAISS Summer School and the ICML Conference both in July. At the summer school, I was able to meet a substantial amount of the MI community. At ICML, I presented some of my work as part of a workshop, which was my first experience in this kind of events.

2.3 Timeline

- In depth exploration of the field of MI, preliminary work on several potentially interesting directions 1 month
- Focus on one promising direction: use attribution methods on feature compositions to construct a graph of dependencies.......1 month
- First experiments and results .... 1 month
- Discovery and exploration of the SBM literature.......2 weeks
- HAAISS Summer School .... 1 week
- ICML conference.......1 week
- Report writing and further experiments.......1 month

3 Background

This section gives some background on notions from neurosciences (Section 3.2), network sciences (Section 3.3) and mechanistic interpretability (Sections 3.4 and 3.5). These notions are the fundations upon which we build our work.

3.1 Notations

We regroup here some notations that may come handy later on. In particular, those relative to sparse autoencoders will be introduced again and explained in Section 3.4.

We will denote by $\mathcal{M}$ a neural network model, and by m its *modules* - MLP layers for multi-layer perceptrons (MLP), transformer blocks, attention layers or attention heads for transformers, convolutional layers for convolutional neural networks (CNNs).

For transformers, the dimension of the residual stream will be denoted by d_{model}. When $\mathcal{M}$ denotes a model with an arbitrary architecture, d_{model} will be the dimension of the input and we call d_m the dimension of the output of module m in general. The output of module m on input x will be denoted by $x^m \in \mathbb{R}^{d_m}$.

For all modules m, let

$$E_m : \mathbb{R}^{d_m} \to \mathbb{R}^{d_{\text{dict}}}$$
$$D_m : \mathbb{R}^{d_{\text{dict}}} \to \mathbb{R}^{d_m}$$

be an encoder and decoder pair, with $f^m = E_m(x^m)$. These can be arbitrary autoencoders, from a linear change of basis to a gated SAE - see Section 3.4. Each canonical dimension $i < d_{\text{dict}}$ of the dictionary will be called a feature, its associated feature vector b_i in the weights of the dictionary D_m is supposed to be unitary, and the magnitude $\alpha_i = f_i^m$ of the activation of this feature on any input will be called the feature's activation. We will often identify b_i and α_i under the notation f_i or simply f, for feature.

Scalar functions depending on the evaluation of x by $\mathcal{M}$ will be denoted by λ. It can be a metric function on the output of the model, or any scalar function depending on its internal activations.

3.2 Brain Networks

The field of neurosciences has been trying to understand the brain by explaining the function of each of its parts. There are two main paradigms when trying to explain the function of the brain: *functional segregation* and *functional integration* of brain regions [Tononi et al., 1994, Sporns, 2013, Pavlovic, 2015]. The former, which historically appeared first, assumes that the brain is divided into many parts, each specialising in various functions by engaging in local processes, while the latter assumes that the brain is a complex network of neuronal elements whose connections facilitate different functional processes. *Functional integration* has been shown to be more successful in explaining higher cognitive processes like visual recognition, social cognition or emotions [van den Heuvel and Sporns, 2013].

When it comes to building a brain network, three major classes of connectivity can be distinguished: *structural*, *functional* and *effective* connectivity [Hagmann et al., 2007, Friston, 2011]. Structural connectivity refers to the anatomical connections between brain regions. Functional connectivity measures temporal correlations between the activation of brain regions, the idea being that regions which share the same function, or work together in the emergence of more complex ones should be active at the same time. Effective connectivity measures causal relationships between brain regions through actual information flow.

3.3 Block Models

Block Models [Daudin et al., 2008, Peixoto, 2017] are a class of generative models for graphs. For a graph with $n \in \mathbb{N}$ nodes, they assume that the nodes are partitioned into $B \leq n$ blocks and that the edge probabilities are summarised in a matrix $P \in \mathbb{R}^{B \times B}$ such that the probability of an edge (i, j) with nodes assigned respectively to block b_i and b_j depends only on these blocks and is given by P_{b_i,b_j}. The adjacency matrix A is then generated by sampling each entry A_{ij} from a Bernoulli distribution with parameter P_{b_i,b_j}.

This is a generalisation in blocks of the Erdos-Renyi model, which corresponds to the case where $B = 1$. It can model a large variety of graph structures, some of which are presented in Figure 3.

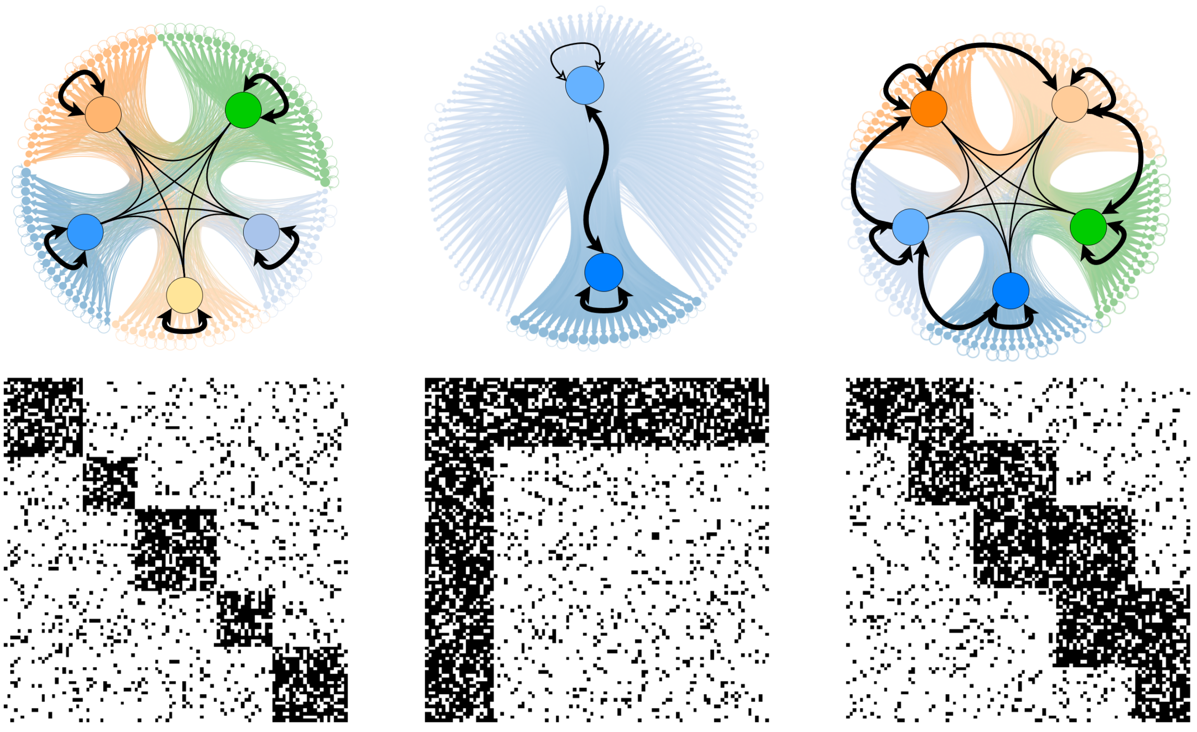

(a) modular (b) core-periphery (c) ordered

Figure 3: Examples of graphs with block structure recovered by a SBM. *Top row*: contains visualisations of the graphs with nodes ordered and colored by block. The size of the arrow represent the number of edges between blocks. *Bottom row*: shows the corresponding adjacency matrix.

One can either use this framework to generate random graphs with a given structure, or to infer the underlying structure of a given graph. In the latter case, the goal is to find the block structure that maximises the likelihood of the observed graph. We will focus on the latter and fit Stochastic Block Models (SBMs) [Peixoto, 2017, Lee and Wilkinson, 2019] to a given graph - in our case, it will be the computational graph of a neural network.

Community detection algorithms are commonly used in the fields of neurosciences [Chen et al., 2013] and neural networks [Filan et al., 2021, Bushnaq et al., 2024] to try to find the underlying structure of complex networks and hopefully explain some mechanisms of the overall function of the model. However, these algorithms assume that nodes are arranged in communities - where nodes are densely connected within communities, but sparsely between them - which restricts the possibilities on the discovered structures. Moreover, they are often prone to overfitting, where community structure is discovered even though it is merely an artefact of random fluctuations - see Peixoto [2017].

SBMs allows us to relax these assumptions and find more general structures. Through the choice of clever prior distributions in Bayesian models [Peixoto, 2017], we can avoid the pitfall of underfitting as well as overfitting statistical fluctuations in the data by finding blocks that do not account for statistically significant structures in the data that can't be explained by random fluctuations. Underfitting is mainly avoided through recursive block models to get the likelihood of some choice for the probabilities between blocks, which also enables the discovery of nested structures in the data. Efficient inference algorithms arise from the choice of these priors and bayesian frameworks, using Markov Chain Monte Carlo methods to find the block structure that maximises the likelihood of the observed graph.

SBMs were succesfully used in neurosciences [Pavlovic, 2015, Peixoto, 2018] to identify established brain structures and advance the field as a whole. They were able to identify both functional segregation in modular parts of the network and functional integration in densely connected cores.

3.4 Sparse Autoencoders

Linear Representation Hypothesis. The linear representation hypothesis assumes that a model's latent space behaves mostly linearly, with high level concepts represented as disantangled directions in this space often called features. It also states that linear directions can be found using linear probes and that they can be used as steering vectors to predictably influence the model's behavior. Evidence of all these aspects can be found in Park et al. [2023], Mikolov et al. [2013], Pennington et al. [2014], Nanda et al. [2023], Gurnee and Tegmark [2024], Wang et al. [2023b], Turner et al. [2023].

Sparse coding. This motivates the use of dictionary learning and sparse coding methods to find overcomplete bases able to sparsely represent the data at any point in the model. The hope here is that the learnt dictionary's feature vectors correspond to actual features used by the model. Formally, for some module m we want a family $\{b_i \in \mathbb{R}^{d_m}\}_{i=0}^{d_{\text{dict}}}$ such that, for all activations x^m, there is a set $\{\alpha_i \in \mathbb{R}\}_{i=0}^{d_{\text{dict}}}$ mostly 0 such that $x = \sum_{i=0}^{d_{\text{dict}}} \alpha_i b_i$. There are two dual problems here, one is to find such a family, the other one is, given a family and a data point, to find the α_i.

Notation. In the rest of this work, these b_i will be unit vectors called features, and both b_i and its magnitude α_i will be identified under the notation f_i, as announced in Section 3.1.

Autoencoders. People usually train sparse autoencoders (SAE) (E_m, D_m) to solve both of these problems. The decoder D_m serves as the dictionary, with its weight matrix containing the b_i in column, while the encoder E_m is the sparse coding function.

The majority of current research in transformer interpretability uses one layer MLPs [Huben et al., 2024, Gao et al., 2024] trained to learn the identity function with an added sparsity loss. Rajamanoharan et al. [2024] proposed an improvement to this method by adding a gated mechanism to the MLPs.

Bushnaq et al. [2024] introduced a change of basis called the *Local Interaction Basis* (LIB) that can be viewed as a linear autoencoder capturing functionally important features, though the decomposition is not sparse. Any relevant change of basis found by future research can be similarly used as a linear autoencoder.

All of this work finds highly interpretable features, and is one direction in trying to find better geometric representations of model's latent space.

3.5 Interpretability

We provide here some background on previous interpretability work (Sections 3.5.1 and 3.5.2), on some key implementation details (Section 3.5.3) and on current limitations of these methods.

3.5.1 Attribution and explanations

In this section, we discuss classical attribution methods in neural networks. Attribution is the process of explaining a model's prediction by attributing it to some part of the input or intermediate feature. This is a very active field of research, and many methods have been proposed.

We begin with the most basic and popular methods. Springenberg et al. [2015] used gradient in vision models to highlight which pixels of an image best explain some output. Let $\lambda : x \in \mathbb{R}^{d_{\text{model}}} \mapsto \lambda(x) \in \mathbb{R}$ be a scalar function on the model's computations - say the output value for a given class in a classifier - and $x \in \mathbb{R}^{d_{\text{model}}}$ an input image of size d_{model}.

$$\text{Attr}(x_i) = \frac{\partial \lambda(x)}{\partial x_i}$$

They show that even though results are satisfactory, the explanation is really noisy and imprecise. They introduce guided backprop, a variant that removes negative gradients in *ReLU* - Rectified Linear Unit - layers in the backward gradient computation, which seems to improve the quality of the explanation.

Thus, the attribution when passing through a *ReLU* layer is not

$$\text{Attr}^l = (x^l > 0) \cdot \text{Attr}^{l+1}$$

which would be the gradient found in a typical backward pass, where x^l and Attr^l are respectively the activation and attribution of the layer l, but

$$\text{Attr}^l = (x^l > 0) \cdot (\text{Attr}^{l+1} > 0) \cdot \text{Attr}^{l+1}$$

which is the masked gradient, kept only when positive: we intervene in the backward pass so that it does not compute the actual gradient but this quantity instead. For non *ReLU* layers, the attribution is simply the gradient.

Later on, Zhou et al. [2016] and Selvaraju et al. [2020] introduced CAM, GradCAM and guided GradCAM, methods building upon these results to refine the attribution of the model's prediction to the input specifically for convolutional layers. Another noteworthy method for attribution is Layer-Wise Relevance Propagation [Montavon et al., 2019]. Methods purely based on gradients most notably suffer from the fact that the model is not a linear function and that we don't know which directions pointed by the gradient are relevant - unless there is some sense of counterfactual input, as in [Syed et al., 2023]. Moreover, if the relevant bits' effects are saturated, their gradient is null and we miss them completely.

Building upon this work, Sundararajan et al. [2017] introduced Integrated Gradients (IG) along with a set of axioms desirable for an attribution method. This method can be formalised in a general manner as follows. Let

$$\lambda : x \in \mathbb{R}^{d_{\text{model}}} \mapsto \lambda(x) \in \mathbb{R}$$

be a scalar function, $x_{\text{clean}} \in \mathbb{R}^{d_{\text{model}}}$ an input, $x_{\text{patch}} \in \mathbb{R}^{d_{\text{model}}}$ a baseline input. Let also $\gamma : [0, 1] \to \mathbb{R}^{d_{\text{model}}}$ be a differentiable path function between x_{patch} and x_{clean}. They define the attribution function as:

$$\text{Attr}_\gamma(x_i) = \int_0^1 \frac{\partial \lambda(\gamma(t))}{\partial \gamma_i(t)} \cdot \frac{\partial \gamma_i(t)}{\partial t} dt$$

The main idea is that since the model is not linear, the gradient alone is not enough to explain the model's prediction - adressing the first issue of the methods mentioned earlier. E.g. if some very important feature happens to be on a plateau, it will not be picked up by the gradient.

Smilkov et al. [2017], Miglani et al. [2020], Kapishnikov et al. [2021] improved this method of integrated gradient mainly through the choice of path and baseline which constitute the main limitations of integrated gradients, and can lead to noise in the attribution.

Explanation methods, initially focusing on attributing the observed output to a subset of the input, paved the way for *circuit discovery*.

3.5.2 Circuit Discovery

Early work involving vision models find related features across layers seemingly responsible for some common behavior [Olah et al., 2020]. This sparked interest in the community, and recent work on both vision and language transformers find subsets of model's components - nodes in a computational graph - working together to implement some behavior. These subsets of nodes are called *circuits*.

In the circuit discovery literature, people used methods developed in computer vision [Syed et al., 2023, Marks et al., 2024] as they got interested in attributing the observed output to intermediate nodes and edges in the computational graph, not only input nodes, in order to get a more refined explanation of the whole computations leading to some output. They also introduced novel methods [Wang et al., 2023a, Conmy et al., 2023, Ferrando and Voita, 2024, He et al., 2024] mostly involving patching single nodes or edges to get an estimation of their contribution, or linearizing the internal computations to simplify attribution.

Conmy et al. [2023] patches each edge with a baseline value. Edges are those of the complete computational graph, as defined by the original transformer architecture - between modules. They then compare the induced model to the unmodified one. Edges with an effect smaller than some threshold are removed. This is done for a specific task with its associated dataset. However, this approach is very computationally expensive and doesn't scale well with model size.

Syed et al. [2023] reused some gradient based attribution method by comparing the gradient to some patching direction in order to have a much more efficient method.

Such methods based on whole modules are very coarse and aim at finding the role played by polysemantic units under a specific context. Thus, people started to try to disentangle the model's computation using more refined nodes to get a more interpretable representation.

To this end, O'Neill and Bui [2024] used SAE's trained specifically on the task under study, with a final graph still based on heads but using information flow through these disantangled features. He et al. [2024] also used SAE's, building dependencies between features by linearising the model's computations, removing the need for patches.

Marks et al. [2024] similarly used sparse autoencoders to disantangle the model's computation. They used integrated gradients to get the contribution of each feature to the model's output, keeping only features with an attribution higher that some threshold. They also provide a way to construct relevant edges between these features, however they don't use these edges and are only interested in features : they evaluate their circuits with node ablation (see Section 3.5.3). We compare in Section 5.2 the evaluation of their graphs using node and edge ablation.

Some works additionally tried to use clever changes of basis, or linear autoencoders, to disantangle the model's computation. Merullo et al. [2024] worked purely based on the weights of the model to draw edges between singular vectors of weight matrices. Bushnaq et al. [2024] introduced a change of basis called Local Interaction Basis (LIB) that is supposed to capture functionally important features and dependencies.

All these methods seem to be able to give interesting cherry picked explanations on toy models or tasks.

3.5.3 Node vs edge ablation

During the evaluation of a circuit, one can either run it through node ablation or, if the circuit discovery methodology also constructed edges, these can be taken into account to run edge ablation. We illustrate the difference between the two in Figures 4 and 5: Figure 4 shows a three layer MLP densely connected and an example of circuit yielded by an arbitrary circuit discovery method, while Figure 5 shows the actual model induced by node or edge ablation.

Node ablation. Given a set of nodes in a circuit $V_C \subset V$ - where V is the set of all nodes in the network - node ablation consists in the intervention on the value of nodes in $V \setminus V_C$, replacing them by some patch value just after their computation. The effect will propagate to all downstream nodes.

Edge ablation. Given sets of nodes and edges $(V_C \subset V, E_C \subset E)$ in a circuit - where V is the set of all nodes and E all edges given by the architecture of the model, each node is seen as a function of its predecessors according to E_C. Let d be a downstream node, U_C be the set of its ancestors with respect to E_C, and U its ancestors with respect to E. We compute the value of d by using the unmodified values of nodes in U_C and some patch value for nodes in $U \setminus U_C$.

These two procedures lead to extremely different effective circuits during evaluation, as illustrated in Figure 5, and the practice of using node ablation to evaluate a circuit defined by edges should be abandoned.

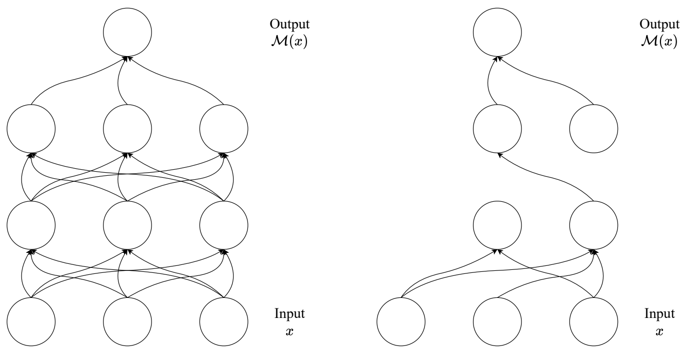


Figure 4: *Left*: Example of a 3 layer MLP. *Right*: Example of circuit discovery result.

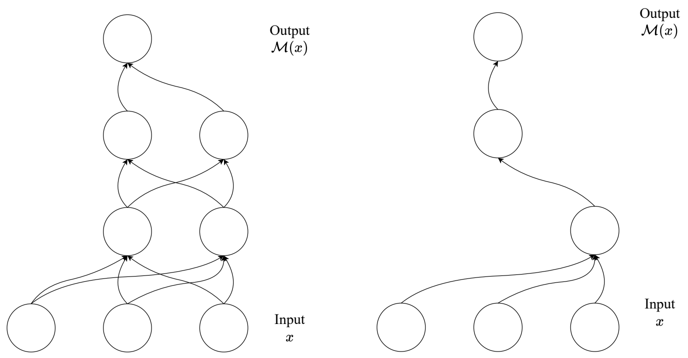


Figure 5: Model induced by node (*left*) and edge (*right*) ablation on the example circuit from Figure 4

3.5.4 Weaknesses of explanations and circuit discovery

Explanations produced by attribution methods fail to align with the ground truth, particularly in controlled scenarios where the correct explanatory factors are known and should be straightforward. Casper et al. [2023] fine-tuned vision classifiers and inserted a Trojan, providing a known and simple ground truth explanation for the model's prediction on patched inputs, yet attribution methods mostly discover uncorrelated explanations. Sreenivasan et al. [2022] showed that in randomly initialised networks, subsets of weights can approximate a target behavior.

These two facts show that multiple explanations for a single behavior can be found and current attribution methods can't distinguish between a correct explanation and an artefact resulting of random fluctuations in parameter space.

3.6 Parallels between neurosciences and MI

Recent work in mechanistic interpretability (MI) seems to align well with neuroscience terminology and ideas. Probing methods [Alain and Bengio, 2017, Marks and Tegmark, 2023] for example would lie in the paradigm of *functional segregation* as they try to find specific features without puting them in relation with others, while circuit discovery would correspond to *functional integration* - with all *structural*, *functional* and *effective* connectivity having their equivalent in MI studies. For a more detailed discussion, refer to Appendix B.

Another well established notion in neurosciences is sparse coding, which shows that the brain efficiently represents important information. It became a very popular method in MI to find interpretable features in the model's latent space and disentangle its computation since the work of Bricken et al. [2023].

4 Regions of Interest

In this section, we define regions of interest through *functional connectivity* and empirically validate their relevance in the study of neural networks.

4.1 Method

Let $f : x \mapsto f(x) \in \mathbb{R}^{d_{\text{dict}}}$ be the activations of some features for a given input x in a model $\mathcal{M}$. These can be features at some specific module output or a concatenation of several module outputs. We define a *functional* graph on these features as follows.

We compute the correlation matrix of these features across a dataset $\mathcal{D}$:

$$C = \text{Corr}(f(\mathcal{D}))$$

We then fit a SBM to the complete graph with features as nodes and weighted by the matrix W defined by: $\forall i, j < d_{\text{dict}}, W_{i,j} = \frac{1}{2} \log \left(\frac{1+C_{i,j}}{1-C_{i,j}} \right)$. This is a common practice in the field of neurosciences to get a functional graph from fMRI data [Peixoto, 2018]. Using Fisher transformation on the correlations ensures that the weights approximately follow a normal distribution and are not bounded in $[-1, 1]$, which is necessary for the SBM to work.

These weights encode functional relationships between features, as two features that activate in similar patterns will have a high correlation. By no means can a functional graph be used to infer causality or actual information flow, it can only be used to study the functional relationships between nodes.

If f contains all activations from all modules of a model or if f's canonical dimensions correspond to coarser nodes, this would be a typical *functional connectivity* graph as used in neurosciences. However studying the whole model with high granularity using this method is unrealistic as the total number of features is typically too high to get their correlation matrix, let alone fit a SBM to it.

We will restrict this method to single module's output features and define a region of interest (ROI) as a block - as defined by a SBM - of functionally similar features in this space. Our ROI's are thus module-wise partitions of feature dictionaries. Figure 1 illustrates this process for the last layer of a feature extractor in a CNN - setting that we also use in the following experiments.

We can take either the activation or the attribution for each feature. In practice, integrated gradient (see Section 3.5.1) appears to yield the best results, so we will adopt it.

4.2 Experiments

In this section we identify regions of interest in vision models. We then validate the quality of these regions by intervening on them and checking the model's behavior. We compare results with communities found by clustering algorithms.

We first test our method on CNN classifiers trained on the MNIST dataset. Full details of this experiment can be found in Appendix A.1. We focus on the last layer of the feature extractor where we train a sparse autoencoder and get ROI's on these features as described in Section 4.1. We find that in this simple setting, SAEs don't improve the results, and we report results for the identity dictionary - each feature is a canonical direction. Similar results are obtained for all other layers of the model as well as more complex models trained on CIFAR-10.

We automatically label regions based on the class with highest average attribution. Figure 6 shows a visualisation of ROIs and their attribution on each class.

To validate the quality of our ROIs, we intervene on them and check if the model's behavior changes predictably in two additional experiments.

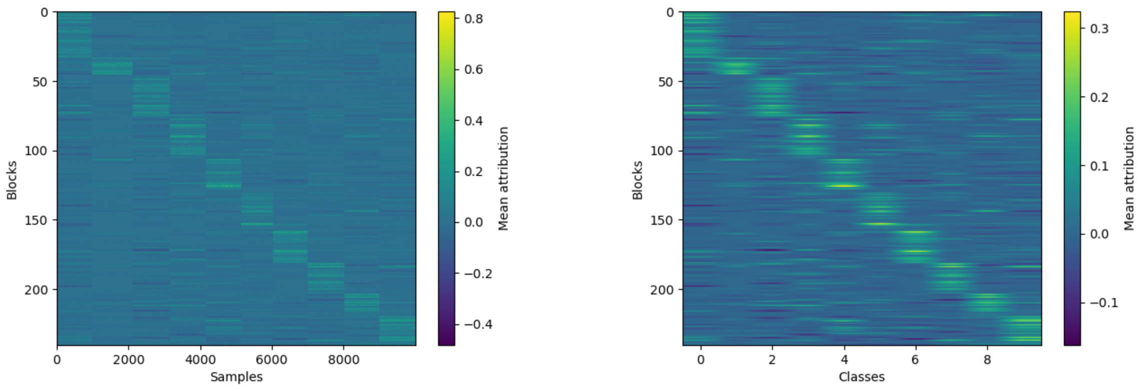


Figure 6: *Left*: This figure shows the mean attribution of each ROI (block) for all inputs. ROIs are sorted by the most activating class. Inputs are sorted by class. *Right*: Same visualisation but inputs are averaged by class.

Our first experiment is inspired from Lu and Ester [2019] and checks that ROIs are relevant for every input separately. For each input we sort the ROIs by highest attribution, ablate the top τ of them in the forward pass and observe whether this modified model was able to classify the input. We ablate ROIs by setting all their features' activations to 0 during the forward pass. We plot the mean accuracy for several τ in Figure 7.

We also measure the complement of this, where we instead keep only the top τ ROIs. We expect to recover good accuracy for very small τ since now we keep the most important ROIs and discard the unimportant ones.

We see a significant improvement over Lu and Ester [2019]'s results, with a sharp accuracy drop for even a small number of ROI ablation. We find that 10% of blocks are enough to completely break the model's behavior - respectively to recover it -, which is coherent with our finding that ROIs tend to have sparse attribution.

As we do not have access to original code from Lu and Ester [2019], we give in Appendix A.2 a comparison between the partition given by a SBM and that given by a community detection algorithm - here, spectral clustering - under the exact same settings and show that SBMs yield better results.

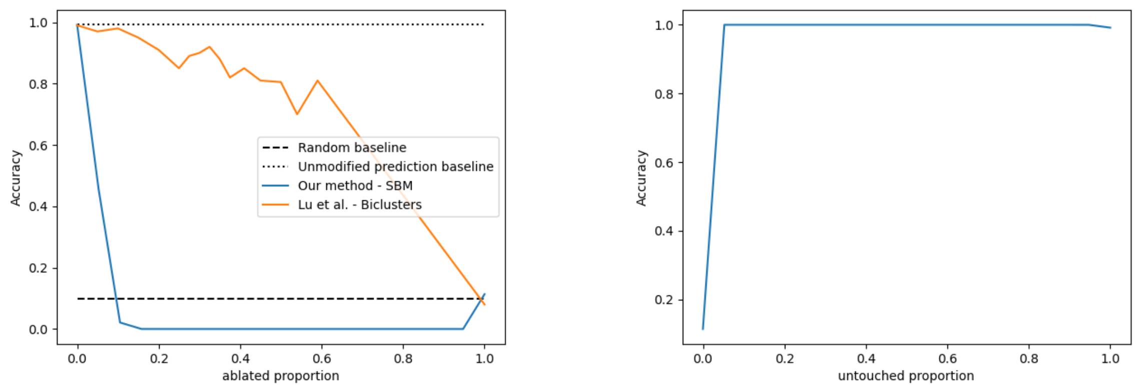


Figure 7: *Left*: Accuracy vs proportion of ablated ROIs - sorted by attribution. *Right*: Accuracy vs proportion of untouched ROIs.

Next, we want to verify that ROIs capture general behaviors, independently of the input. To this end, we label each region by its most attributed class. For each class c, we ablate all ROIs labeled by c. We then measure the average accuracy of this truncated model class-wise to check that the model becomes unable to classify correctly on c while keeping its performance on all $c' \neq c$.

Reciprocally, we ablate all ROIs labeled by $c' \neq c$ and check that the model preserves its performance on c while being unable to classify other classes.

Results are reported in Figure 8. In the first scenario, we observe significant accuracy drops on the target class for all classes. In the latter scenario we find that the truncated model becomes a constant classifier for the target class : we completely remove the model's ability to do anything but the target behavior.

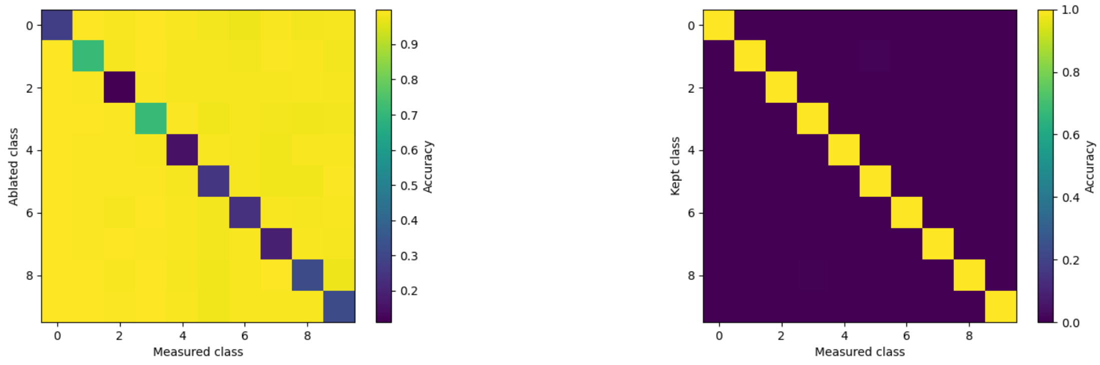


Figure 8: Accuracy on all classes for every ablation setting. *Left*: Ablation of ROIs labeled by the target class. *Right*: Ablation of all ROIs except those labeled by the target class.

5 Computational Graphs

In this section, we propose a novel algorithm to build computational graphs that can incorporate several levels of abstraction in neural networks through *effective connectivity*. We then study these graphs by comparing their behavior to that of the original model and intervening on their identified structures to predictably change the model's behavior.

5.1 Method

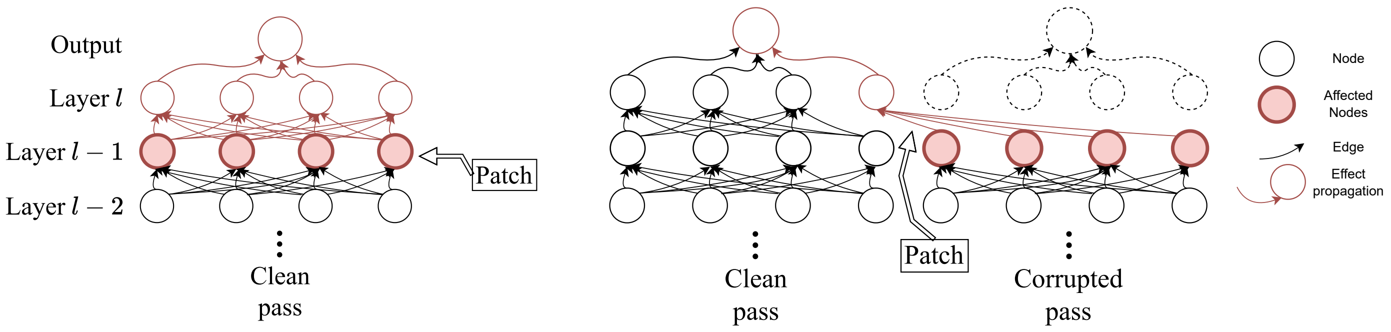


Figure 9: Illustration of our attribution method. All nodes of a given module are patched simultaneously to compute IG. For node attribution (*left*), their value is overwritten in the forward pass. For edge attribution (*right*), the patched values are only used as input for the target downstream node.

We want to construct a computational graph $G = (V_C, E_C)$, where $V_C \subset V$ is a subset of V the network's nodes, and $E_C \subset E$ is a subset of E all edges between nodes of V allowed by the model's architecture. We construct two functions: $w_V : V \to \mathbb{R}$ used for *node ablation* settings, and $w_E : E \to \mathbb{R}$ used for *edge ablation* settings, respectively assigning weights to nodes and edges.

For the nodes, we partition feature dictionaries per module and select the corresponding classes as nodes. For instance, this can correspond to individual features in the case of the discrete partition or ROIs as defined in Section 4. While features should correspond to atomic functions in the disentangled computations of the model, ROIs provide a layer of abstraction and reduce the size of the graph by regrouping functionally similar features. We only describe the case of ROIs as individual features are particular cases of partitions.

We add a sink node y for the model's final output. To construct w_V and w_E we proceed as follows.

First, we select a model $\mathcal{M}$ to analyse along with some partition of feature dictionaries in this model to define the nodes.

Then, we chose a dataset $\mathcal{D}$ corresponding to the task under study. We include the choice of patches in this dataset - typically involving noise, zero ablation, mean ablation, resampling or counterfactual inputs. Let $x \in \mathcal{D}$ be some input.

We begin by constructing G_x, w_V^x and w_E^x, the graph of computations of $\mathcal{M}$ on x.

In the case where $\mathcal{M}$ is a transformer, we duplicate each node for every token position in the input. Once this is done, we contract this expanded graph over token position typically summing node or edge weights in w_V and w_E.

For some module m with output $x^m \in \mathbb{R}^{d_m}$, let $f^m := E_m(x^m) \in \mathbb{R}^{d_{\text{dict}}}$ be the vector of activations of this module's features and $r(f^m)$ be the canonical projection of f^m to a subset of its dimensions corresponding to the ROI r.

Let $\lambda : x^m_{\text{patch}} \mapsto \lambda(x^m_{\text{patch}})$ be a metric function on the model's evaluation of x where the forward pass was modified using ablation with x^m_{patch} as patch. We use the integrated gradient attribution method to get the contribution of $r(f^m)$ to $\lambda(x^m)$ - the metric on the unmodified model :

$$\text{Attr}(r(f^m)) = \int_0^1 \nabla_r \lambda(\gamma(t)) \cdot \left(\frac{dr(\gamma(t))}{dt}\right) dt$$

where $\gamma : [0,1] \to \mathbb{R}^{d_{\text{dict}}}$ is defined as the linear interpolation path between the baseline f^m_{patch} and f^m the clean activation: $\gamma(t) = t \cdot f^m + (1-t) \cdot f^m_{\text{patch}}$. Based on the choice of $\mathcal{D}$, the baseline can either be 0, noise or computed from a corrupted input. We define

- $\forall r \in V,\ w_V^x(r) := \text{Attr}(r(f^m))$ where λ is a metric on the model's output computed with *node* ablation, typically λ_{logit} defined below.
- $\forall (r, y) \in E,\ w_E^x((r, y)) := \text{Attr}(r(f^m))$ where λ is similar to the previous case.
- $\forall (r, r') \in E$ s.t. $r' \neq y,\ w_E^x((r, r')) := \text{Attr}(r(f^m))$ where λ is either a metric on the model's output or the activation value of r' [5] computed with *edge* ablation.

Now that we have w_V^x and w_E^x for all $x \in \mathcal{D}$, let α be an aggregation function, typically we use *max* or *mean*. In practice, we chose the *mean* function. See Appendix C for a discussion on this choice. We aggregate over input as follows :

$$\forall v \in V,\ w_V^\alpha(v) := \alpha_{x \in \mathcal{D}}(w_V^x(v))$$
$$\forall (u, v) \in E,\ w_E^\alpha((u, v) := \alpha_{x \in \mathcal{D}}(w_E^x((u, v))$$

We then use thresholding to get binary membership values : let $\tau \in \mathbb{R}$ be some threshold value:

$$\forall v \in V,\ w_V(v) := \begin{cases} 1 & \text{if } w_V^\alpha(v) > \tau \\ 0 & \text{otherwise} \end{cases}$$
$$\forall (u, v) \in E,\ w_E((u, v) := \begin{cases} 1 & \text{if } w_E^\alpha((u, v)) > \tau \\ 0 & \text{otherwise} \end{cases}$$

These membership values finally define our circuit $G = (V_C, E_C)$. For node ablation settings, we have $V_C = \{v \in V | w_V(v) \neq 0\}$ and $E_C = E|_{V_C}$. For edge ablation, we have $E_C = \{(u, v) \in E | w_E((u, v)) \neq 0\}$, $V_C = V$, and we prune the graph to keep only nodes and edges that can both reach the sink node y and be reached from at least one input node. This procedure gives a functionally equivalent circuit, we do it to study the properties of the resulting computational graph.

The threshold gives us some control on the sparsity of the resulting graph. In practice, we typically threshold on the fly, during the construction of the graph for each input, for computational reasons.

[5] We typically chose the latter when using discrete partitions - individual features. The former is equivalent to choosing as metric the attribution of r', so the former and the latter differ only in that one uses the activation while the other uses the attribution of the downstream node.

Faithfulness. Now that we have a computational graph G, we want to check whether it is functionally equivalent to $\mathcal{M}$. To do so, we run the graph using edge ablation[6]. We identify G a computational graph and its induced function. We denote the empty graph, with $E_V = \emptyset$, by $\emptyset$ and we identify $\mathcal{M}$ and (V, E). We compare $G(x)$ to $\mathcal{M}(x)$ for all $x \in \mathcal{D}$ using a variety of metrics :

- Logit difference: $\lambda_{\text{logit}}(G(x)) = G(x)[\text{trg}^{\text{good}}] - G(x)[\text{trg}^{\text{bad}}]$ where trg^{good} are the indices of desirable target answers and trg^{bad} are the indices of unwanted answers. Large positive values mean that G can succesfully complete the task[7].
- Kullback-Leibler divergence: $\lambda_{\text{KL}}(G(x)) = KL[\sigma(G(x))||\sigma(\mathcal{M}(x))]$, where σ is the softmax function. The closer this is to 0 the better, but it is not clear what a good value for this metric is, especially when faced with some degeneracies - e.g. when $\sigma(\mathcal{M}(x))$ contains some 0s.
- Statistical distance: $\lambda_{\text{stat}}(G(x)) = \frac{1}{2}\sum_i |\sigma(G(x))[i] - \sigma(\mathcal{M}(x))[i]|$, where σ is the softmax function. This metric is bounded in $[0, 1]$ and its interpretation is straightforward.

For all metrics λ, we measure the *faithfulness* ϕ of G as $\phi(\lambda, G) = \frac{\lambda \circ G - \lambda \circ \emptyset}{\lambda \circ \mathcal{M} - \lambda \circ \emptyset}$. This normalisation metric is supposed to capture how much of the model's performance is explained by our circuit relative to a baseline (here the empty circuit): values close to 0 indicate that G is not better than the empty graph, while values close to 1 indicate that G align with $\mathcal{M}$

Structure. Having a faithful representation of $\mathcal{M}$ as a computational graph G is not enough, as the $G = (V, E)$ trivially maximises faithfulness. We also want it to be a useful representation of the model's computation.

We first hypothesise that for some choice of V, high faithfulness is achieved already by sparse computational graphs. This would mean that the graph successfully captures and disentangles the essential functionalities of the model and relationships between its computational units. Sparsity can be measured by the ratio between the number of edges and the maximum possible number of edges. In neural networks, this is often less than the square of the number of nodes, as the architecture of the model restricts the number of possible connections.

Another hypothesis is that important behaviors or functionalities inside of the model would correspond to some structures in G. To discover potential structures, we use the SBM to infer a block structure in G - unweighted, as opposed to Section 4 where the graph structure was weighted and reflected functional correlation, not effective connectivity. Interventions on these structures, e.g. through ablation or patching and measuring the impact on task-wise faithfulness, can validate their relevance and any potential interpretation of their functionalities.

5.2 Experiments

We test our technique using pythia 70m deduped [Biderman et al., 2023] and pretrained SAEs from Marks et al. [2024] to which we compare our results as our method to obtain a graph is quite similar. We use only residual stream and embedding SAEs, with individual features as nodes.

We chose a dataset $\mathcal{D} = \mathcal{D}_{\text{RC}} \cup \mathcal{D}_{\text{IOI}} \cup \mathcal{D}_{\text{GT}}$ made of a mixture of three equal-size task datasets. The first one is a subject-verb agreement in relative clauses (RC) task from Finlayson et al. [2021], used for comparison with Marks et al. [2024]. Details on the RC setting can be found in Appendix D.1. We also use the now established indirect object identification (IOI) task [Wang et al., 2023a], described in detail in Appendix D.2. Finally, we use a numerical task consisting of finding $n > m$ (GT) described in Appendix D.3. Figure 2 shows an abstract illustration of a computational graph obtained on a mixture of these three tasks.

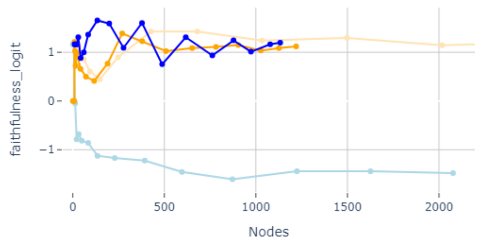
We compare both *node* and *edge* ablation settings described in Section 5.1 on a test split of $\mathcal{D}$. We also include results using the implementation of the circuit discovery from Marks et al. [2024] for comparison, using both their implementation of *node* ablation and our implementation of *edge* ablation.

We find that experiments are very sensitive to the choice of parameters, and report results using zero ablation for the construction of G and mean ablation with corrupted inputs for the evaluation of G.

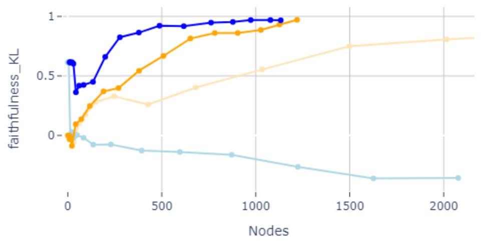
[6]given the definition of G's edges, this is equivalent to node ablation in the corresponding context.
[7]it doesn't mean that G captures the behavior of the model !

Our first experiments explore the choice of metric for evaluation. We sweep over τ to get graphs of increasingly large sizes and measure all metrics and corresponding faithfulness. Figure 10 show the results for all tasks. We can see that either in the node ablation or edge ablation setting, our implementation surpasses that of Marks et al. [2024]. Our circuits evaluated with edge ablation seem to produce the best results both in terms of faithfulness and sparsity, whether you consider the number of features required for a desired faithfulness, or the number of edges.

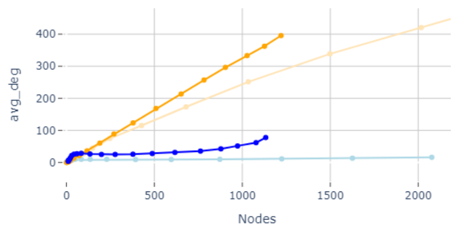
We find that extremely sparse graphs can have near perfect faithfulness to the original model. Additionally, as mentioned in Section 5.1, λ_{logit} is quite noisy and hard to interpret. For the next experiments, we will only report λ_{KL}



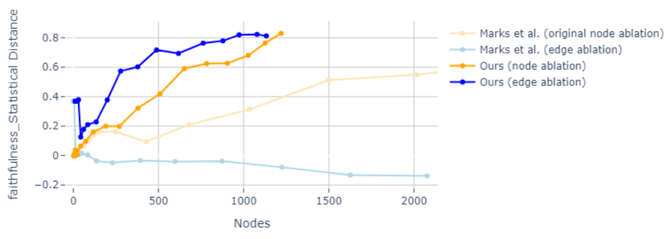
(a) Faithfulness of the logit difference: $\phi(\lambda_{\text{logit}})$



(b) Faithfulness of the KL divergence: $\phi(\lambda_{\text{KL}})$



(c) average degree



(d) Faithfulness of the statistical distance: $\phi(\lambda_{\text{stat}})$

Figure 10: Metric's faithfulness against number of nodes left after thresholding and pruning. Figure 10c additionally shows the average degree against the number of nodes, which summarises the number of edges and the density.

Next, we show the trade-off between faithfulness and average degree, which is closely related to the number of edges and the density, in Figure 11. Ideal computational graphs are sparse (low average degree) and have high faithfulness.

We find that our method with edge ablation surpasses by far all other settings, and report results only with this setting for the last experiment.

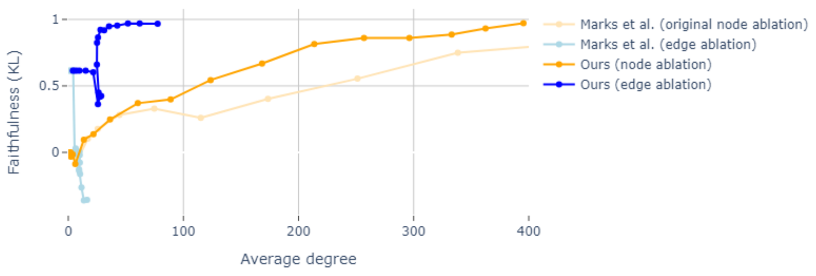


Figure 11: Faithfulness of the KL divergence $\phi(\lambda_{\text{KL}})$ against the average degree in G.

For our last experiment, we use the fact that $\mathcal{D}$ is a mixture of tasks and run the same experiment as in Section 4.2: we measure our ability to automatically disable some behaviors of the model. We fit a SBM to G, automatically assign each block to its most attributed task, prune the graph to remove all nodes assigned to the task under study and measure the resulting faithfulness - as opposed to the accuracy in Section 4.2. We show the results in Figure 12.

Results are less sharp than for vision classifiers but we are still able to significantly alter the graph's faithfulness in the expected direction. These results could lead to novel methods for the removal of unwanted behavior.

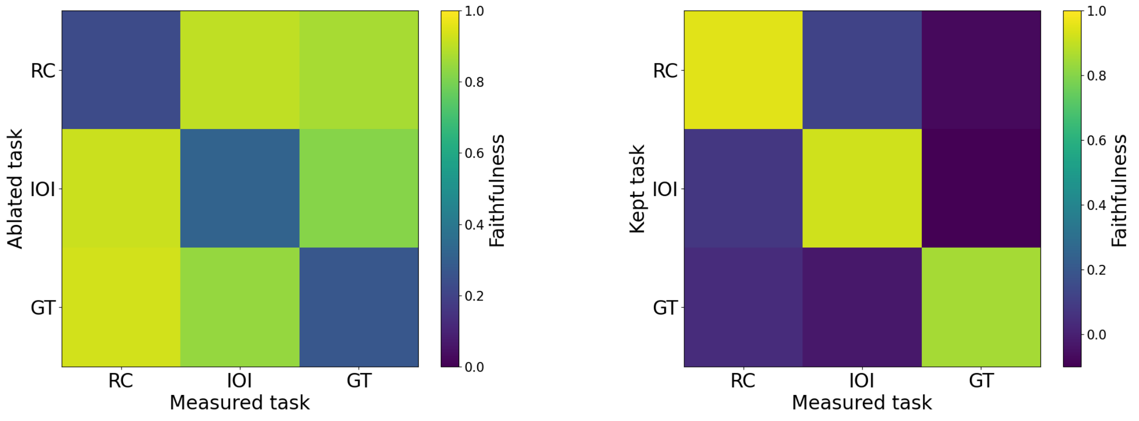


Figure 12: Faithfulness on all three tasks for every ablation setting. *Left*: Ablation of blocks labeled by the target class. *Right*: Ablation of all blocks except those labeled by the target class.

6 Limitations and future work

In this work we established the superiority of edge-ablation for mechanistic studies of the computational graph of language models. This is limited by the fact that edge ablation is significantly more expensive than node ablation, both at inference and while constructing the graph.

Another limitation related to time complexity is that the python library we use for fitting SBMs is quite slow and may not scale well with model size.

In this work we introduced coarse computational nodes as ROIs and compared them to similar methods. Checking whether these ROIs are more relevant than attention heads, in the particular case of transformer models, is left to future work. In the case of coarse computational graphs, comparing heads and ROIs as choice of node is also left to future work.

Additionally, all of our methods are independent of the choice of neurons, and any future work investigating the geometry of representations and introducing new changes of basis or dictionaries is compatible with our work.

This work relies on attribution methods which have been shown to be uncorrelated to simple ground truth explanations (Section 3.5.4). More work on this subject is required to ensure that explanation methods reflect the actual model's computations.

Beyond the aforementioned limitations and improvements, another exciting direction for future research is to explore the emergence of structures throughout the training process and compare structures in models of different sizes. Investigating how these structures evolve across training could corroborate findings of phase transition during training with sharp transition in the loss or Local Learning Coefficient, while finding consistent structures in different model sizes for specific sub algorithm could provide insight on the optimal architecture for these sub algorithms and the scaling properties of the model.

References

Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=HJ4-rAVtl.

Leonard Bereska and Efstratios Gavves. Mechanistic interpretability for ai safety – a review, 2024. URL https://arxiv.org/abs/2404.14082.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar van der Wal. Pythia: A suite for analyzing large language models across training and scaling. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 2397–2430. PMLR, 2023. URL https://proceedings.mlr.press/v202/biderman23a.html.

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, et al. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2, 2023.

Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. Discovering latent knowledge in language models without supervision. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=ETKGuby0hcs.

Lucius Bushnaq, Jake Mendel, Stefan Heimersheim, Dan Braun, Nicholas Goldowsky-Dill, Kaarel Hänni, Cindy Wu, and Marius Hobbhahn. Using degeneracy in the loss landscape for mechanistic interpretability. *CoRR*, abs/2405.10927, 2024. doi: 10.48550/ARXIV.2405.10927.

Stephen Casper, Tong Bu, Yuxiao Li, Jiawei Li, Kevin Zhang, Kaivalya Hariharan, and Dylan Hadfield-Menell. Red teaming deep neural networks with feature synthesis tools. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/febe5c5c6973f713cc43bf0f7c90edbe-Abstract-Conference.html.

Hanbo Chen, Kaiming Li, Dajiang Zhu, Xi Jiang, Yixuan Yuan, Peili Lv, Tuo Zhang, Lei Guo, Dinggang Shen, and Tianming Liu. Inferring group-wise consistent multimodal brain networks via multi-view spectral clustering. *IEEE Trans. Medical Imaging*, 32(9):1576–1586, 2013. doi: 10.1109/TMI.2013.2259248.

Arthur Conmy, Augustine N. Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. Towards automated circuit discovery for mechanistic interpretability. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/34e1dbe95d34d7ebaf99b9bcaeb5b2be-Abstract-Conference.html.

Jean-Jacques Daudin, Franck Picard, and Stéphane Robin. A mixture model for random graphs. *Stat. Comput.*, 18(2):173–183, 2008. doi: 10.1007/S11222-007-9046-7.

Javier Ferrando and Elena Voita. Information flow routes: Automatically interpreting language models at scale. *CoRR*, abs/2403.00824, 2024. doi: 10.48550/ARXIV.2403.00824.

Daniel Filan, Stephen Casper, Shlomi Hod, Cody Wild, Andrew Critch, and Stuart Russell. Clusterability in neural networks. *CoRR*, abs/2103.03386, 2021. URL https://arxiv.org/abs/2103.03386.

Matthew Finlayson, Aaron Mueller, Sebastian Gehrmann, Stuart Shieber, Tal Linzen, and Yonatan Belinkov. Causal analysis of syntactic agreement mechanisms in neural language models. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1828–1843, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.144. URL `https://aclanthology.org/2021.acl-long.144`.

Karl J. Friston. Functional and effective connectivity: A review. *Brain Connect.*, 1(1):13–36, 2011. doi: 10.1089/BRAIN.2011.0008.

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. *CoRR*, abs/2406.04093, 2024. doi: 10.48550/ARXIV.2406.04093.

Wes Gurnee and Max Tegmark. Language models represent space and time. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL `https://openreview.net/forum?id=jE8xbmvFin`.

Patric Hagmann, Maciej Kurant, Xavier Gigandet, Patrick Thiran, Van J. Wedeen, Reto Meuli, and Jean-Philippe Thiran. Mapping human whole-brain structural networks with diffusion mri. *PLOS ONE*, 2(7):1–9, 07 2007. doi: 10.1371/journal.pone.0000597.

Zhengfu He, Xuyang Ge, Qiong Tang, Tianxiang Sun, Qinyuan Cheng, and Xipeng Qiu. Dictionary learning improves patch-free circuit discovery in mechanistic interpretability: A case study on othello-gpt. *CoRR*, abs/2402.12201, 2024. doi: 10.48550/ARXIV.2402.12201.

Robert Huben, Hoagy Cunningham, Logan Riggs, Aidan Ewart, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL `https://openreview.net/forum?id=F76bwRSLeK`.

Andrei Kapishnikov, Subhashini Venugopalan, Besim Avci, Ben Wedin, Michael Terry, and Tolga Bolukbasi. Guided integrated gradients: An adaptive path method for removing noise. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 5050–5058. Computer Vision Foundation / IEEE, 2021. doi: 10.1109/CVPR46437.2021.00501. URL `https://openaccess.thecvf.com/content/CVPR2021/html/Kapishnikov_Guided_Integrated_Gradients_An_Adaptive_Path_Method_for_Removing_Noise_CVPR_2021_paper.html`.

Clement Lee and Darren J. Wilkinson. A review of stochastic block models and extensions for graph clustering. *Appl. Netw. Sci.*, 4(1):122, 2019. doi: 10.1007/S41109-019-0232-2.

Jialin Lu and Martin Ester. Checking functional modularity in dnn by biclustering task-specific hidden neurons. In *Real Neurons {\&} Hidden Units: Future directions at the intersection of neuroscience and artificial intelligence@ NeurIPS 2019*, 2019.

Alex Mallen and Nora Belrose. Eliciting latent knowledge from quirky language models. *CoRR*, abs/2312.01037, 2023. doi: 10.48550/ARXIV.2312.01037.

Samuel Marks and Max Tegmark. The geometry of truth: Emergent linear structure in large language model representations of true/false datasets. *CoRR*, abs/2310.06824, 2023. doi: 10.48550/ARXIV.2310.06824.

Samuel Marks, Can Rager, Eric J. Michaud, Yonatan Belinkov, David Bau, and Aaron Mueller. Sparse feature circuits: Discovering and editing interpretable causal graphs in language models. *CoRR*, abs/2403.19647, 2024. doi: 10.48550/ARXIV.2403.19647.

Jack Merullo, Carsten Eickhoff, and Ellie Pavlick. Talking heads: Understanding inter-layer communication in transformer language models. *CoRR*, abs/2406.09519, 2024. doi: 10.48550/ARXIV.2406.09519.

Vivek Miglani, Narine Kokhlikyan, Bilal Alsallakh, Miguel Martin, and Orion Reblitz-Richardson. Investigating saturation effects in integrated gradients. *CoRR*, abs/2010.12697, 2020. URL `https://arxiv.org/abs/2010.12697`.

Tomás Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 3111–3119, 2013. URL `https://proceedings.neurips.cc/paper/2013/hash/9aa42b31882ec039965f3c4923ce901b-Abstract.html`.

Grégoire Montavon, Alexander Binder, Sebastian Lapuschkin, Wojciech Samek, and Klaus-Robert Müller. Layer-wise relevance propagation: An overview. In Wojciech Samek, Grégoire Montavon, Andrea Vedaldi, Lars Kai Hansen, and Klaus-Robert Müller, editors, *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, volume 11700 of *Lecture Notes in Computer Science*, pages 193–209. Springer, 2019. doi: 10.1007/978-3-030-28954-6_10.

Neel Nanda, Andrew Lee, and Martin Wattenberg. Emergent linear representations in world models of self-supervised sequence models. In Yonatan Belinkov, Sophie Hao, Jaap Jumelet, Najoung Kim, Arya McCarthy, and Hosein Mohebbi, editors, *Proceedings of the 6th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP, BlackboxNLP@EMNLP 2023, Singapore, December 7, 2023*, pages 16–30. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.BLACKBOXNLP-1.2.

Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. *Distill*, 5(3):e00024–001, 2020.

Charles O'Neill and Thang D. Bui. Sparse autoencoders enable scalable and reliable circuit identification in language models. *CoRR*, abs/2405.12522, 2024. doi: 10.48550/ARXIV.2405.12522.

Kiho Park, Yo Joong Choe, and Victor Veitch. The linear representation hypothesis and the geometry of large language models. *CoRR*, abs/2311.03658, 2023. doi: 10.48550/ARXIV.2311.03658.

Dragana M. Pavlovic. *Generalised stochastic blockmodels and their applications in the analysis of brain networks*. PhD thesis, University of Warwick, September 2015. URL `http://webcat.warwick.ac.uk/record=b2863162~S1`. Unpublished.

Tiago P. Peixoto. Bayesian stochastic blockmodeling. *arXiv e-prints*, art. arXiv:1705.10225, May 2017. doi: 10.48550/arXiv.1705.10225.

Tiago P. Peixoto. Nonparametric weighted stochastic block models. *Phys. Rev. E*, 97:012306, Jan 2018. doi: 10.1103/PhysRevE.97.012306.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543. ACL, 2014. doi: 10.3115/V1/D14-1162.

Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Tom Lieberum, Vikrant Varma, János Kramár, Rohin Shah, and Neel Nanda. Improving dictionary learning with gated sparse autoencoders. *CoRR*, abs/2404.16014, 2024. doi: 10.48550/ARXIV.2404.16014.

Nina Rimsky, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Matt Turner. Steering llama 2 via contrastive activation addition. *CoRR*, abs/2312.06681, 2023. doi: 10.48550/ARXIV.2312.06681.

Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. *Int. J. Comput. Vis.*, 128(2):336–359, 2020. doi: 10.1007/S11263-019-01228-7.

Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda B. Viégas, and Martin Wattenberg. Smoothgrad: removing noise by adding noise. *CoRR*, abs/1706.03825, 2017. URL `http://arxiv.org/abs/1706.03825`.

Olaf Sporns. Network attributes for segregation and integration in the human brain. *Current Opinion in Neurobiology*, 23(2):162–171, 2013. ISSN 0959-4388. doi: https://doi.org/10.1016/j.conb.2012.11.015. URL https://www.sciencedirect.com/science/article/pii/S0959438812001894. Macrocircuits.

Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin A. Riedmiller. Striving for simplicity: The all convolutional net. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6806.

Kartik Sreenivasan, Jy-yong Sohn, Liu Yang, Matthew Grinde, Alliot Nagle, Hongyi Wang, Eric P. Xing, Kangwook Lee, and Dimitris S. Papailiopoulos. Rare gems: Finding lottery tickets at initialization. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/5d52b102ebd672023628cac20e9da5ff-Abstract-Conference.html.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 3319–3328. PMLR, 2017. URL http://proceedings.mlr.press/v70/sundararajan17a.html.

Aaquib Syed, Can Rager, and Arthur Conmy. Attribution patching outperforms automated circuit discovery. *CoRR*, abs/2310.10348, 2023. doi: 10.48550/ARXIV.2310.10348.

Giulio Tononi, Olaf Sporns, and Gerald M Edelman. A measure for brain complexity: relating functional segregation and integration in the nervous system. *Proceedings of the National Academy of Sciences*, 91(11):5033–5037, 1994.

Alexander Matt Turner, Lisa Thiergart, David Udell, Gavin Leech, Ulisse Mini, and Monte MacDiarmid. Activation addition: Steering language models without optimization. *CoRR*, abs/2308.10248, 2023. doi: 10.48550/ARXIV.2308.10248.

Martijn P. van den Heuvel and Olaf Sporns. Network hubs in the human brain. *Trends in Cognitive Sciences*, 17(12):683–696, 2013. ISSN 1364-6613. doi: https://doi.org/10.1016/j.tics.2013.09.012. URL https://www.sciencedirect.com/science/article/pii/S1364661313002167. Special Issue: The Connectome.

Alexandre Variengien and Eric Winsor. Look before you leap: A universal emergent decomposition of retrieval tasks in language models. *CoRR*, abs/2312.10091, 2023. doi: 10.48550/ARXIV.2312.10091.

Kevin Ro Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: a circuit for indirect object identification in GPT-2 small. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023a. URL https://openreview.net/forum?id=NpsVSN6o4ul.

Zihao Wang, Lin Gui, Jeffrey Negrea, and Victor Veitch. Concept algebra for (score-based) text-controlled generative models. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023b. URL http://papers.nips.cc/paper_files/paper/2023/hash/6f125214c86439d107ccb58e549e828f-Abstract-Conference.html.

Bolei Zhou, Aditya Khosla, Àgata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 2921–2929. IEEE Computer Society, 2016. doi: 10.1109/CVPR.2016.319.

A More on ROI experiments

A.1 Experiment details

We train a CNN classifier with the following architecture: a feature extractor consisting of two convolutional layers, and a classifier of a single dense layer. The first convolutional layer has 32 kernels of size 5 by 5, the second one has 64 3 by 3 kernels. Both are followed by a ReLU activation function and a max-pool layer with a pool size of 2. The output of this last pooling layer is then flattened and fed to a dense layer of shape 1600 by 10.

The dataset is the standard MNIST dataset, split into 60 000 training images and 10 000 test images.

After the model training, we compute the attribution score of each of the 1600 hidden neurons to the target class' logit, and use these to get a correlation matrix over the whole *train* data. We then partition these neurons as described in Section 4.1.

These blocks are used to run the validating experiments on the *test* set as described in Section 4.2.

A.2 Additional experiment

We present here a comparison between the partition of neurons as given by an SBM and as given by spectral clustering in the weight matrix as given by Section 4.1 applied to the correlation matrix obtained in Appendix A.1. Figures 13 and 14 shows that SBMs present a significant improvement over spectral clustering, and that in our setting, spectral clustering is also vastly more effective than biclusters found by Lu and Ester [2019].

Figure 13 shows the result for the first experiment from Section 4.2, with input specific ablation, while Figure 14 corresponds to the second experiment, with class specific ablation.

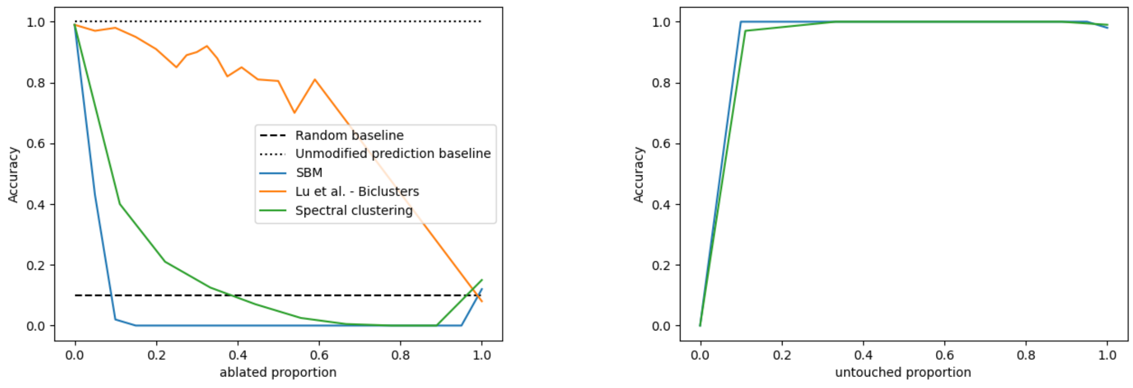


Figure 13: *Left*: Accuracy vs proportion of ablated ROIs - sorted by attribution. *Right*: Accuracy vs proportion of untouched ROIs.

B Discussions on parallels between neurosciences and MI

Recent work in MI seems to fit quite well under neuroscience denominations. In particular, *functional segregation* seems to be very present in MI, with many studies focusing on which isolated part of a model explains or modify some behaviors [Variengien and Winsor, 2023, Burns et al., 2023, Rimsky et al., 2023]. It could also correspond to finding some communities of neurons in a single module that are together responsible for some behavior. Lu and Ester [2019] showed the existence of such communities in vision classifiers, although they did not try intervening on them. We run similar experiments using our method and improve upon their results in Section 4.

Functional integration also has its dual in MI in *circuit discovery*, as it tries to interpret behaviors through the connection between computational nodes. All *structural*, *functional* and *effective* connectivity have their equivalent.

- *structural* connectivity: Filan et al. [2021] used raw weight matrices in MLPs and CNNs as *structural* connections between canonical directions, also knwon as *neurons*. Merullo et al. [2024] worked

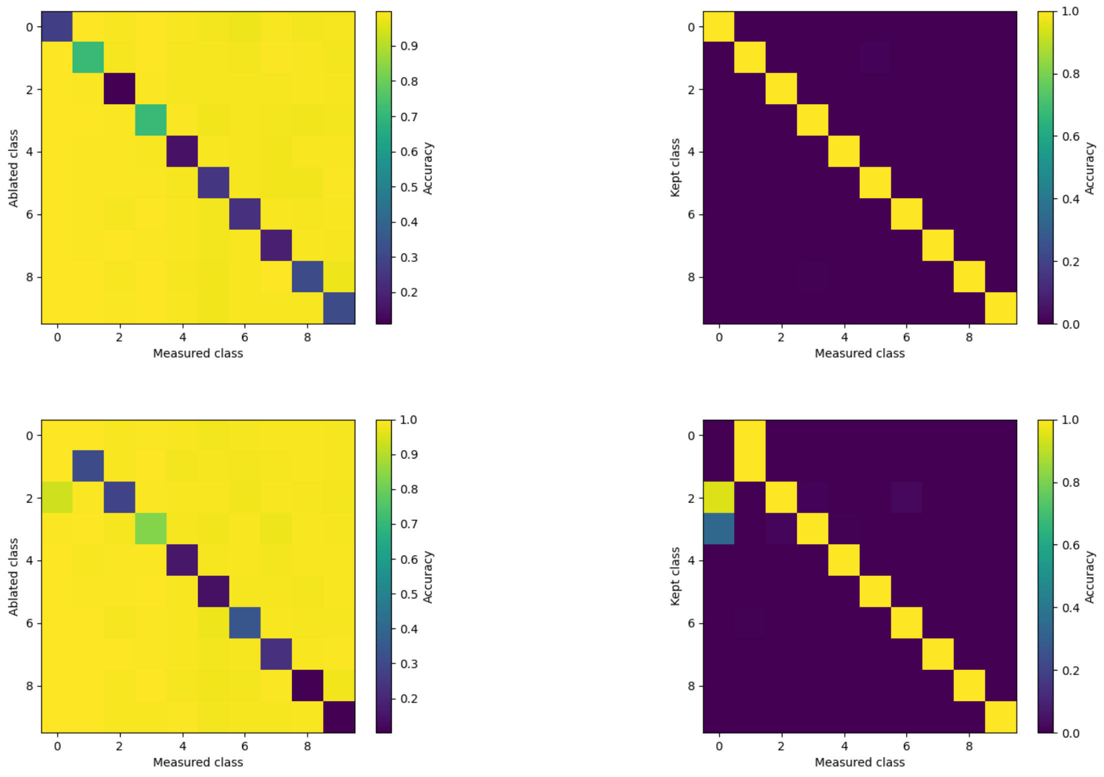


Figure 14: Accuracy on all classes for every ablation setting. *Left*: Ablation of ROIs labeled by the target class. *Right*: Ablation of all ROIs except those labeled by the target class. *Top row*: ROIs are defined by SBM blocks. *Bottom row*: ROIs are defined by communities from spectral clustering.

based on singular decompositions of weight matrices to find structural connections between their principal components in transformer models.

- *functional* connectivity: Lu and Ester [2019] used biclustering on activations across examples to find *functionally* similar neurons in vision models.
- *effective* connectivity: Most of the *circuit discovery* literature builds computational graphs, or graphs of dependencies based on actual information flow across model components. This seems to be the most popular framework in circuit discovery, although most of the time, evaluations of these *circuits* focus only on their nodes.

In both the brain and neural networks, the choice of the granularity for the nodes under study is crucial. In the brain, regions of interests (ROIs) can be defined by using physical proximity as a proxy for functional similarity. In neural networks, there is no such thing as physical proximity. When studying transformer models with low granularity, it is common practice to use attention heads and MLP modules. This is unlikely to be the best coarse unit of computation. We introduce *regions of interest* by grouping sets of functionally similar neurons or features in Section 4.

C Aggregation

Both *mean* and *max* have their drawbacks. *mean* presents the risk to put edges corresponding to rare but important behaviors under the threshold, while keeping edges unimportant but consistent across examples. *max* on the other hand keeps any rare behaviors, but as attribution methods are noisy, the amount of noise might be too high and the graph of dependencies becomes mostly noise and not exploitable.

D Language tasks details

D.1 Subject-Verb agreement

The subject-verb agreement task is adapted from Finlayson et al. [2021]. It consists of prompts of the form :

```
The athlete confuses/*confuse
The farmer (that) the parents love confuses/*confuse
```

In these two examples, the positive target token would be "confuses", while "confuse" would be the negative one for the logit difference metric. The corresponding corrupted inputs would be :

```
The athletes confuses/*confuse
The farmers (that) the parents love confuses/*confuse
```

D.2 Indirect Object Identification

The IOI task contains templates adapted from Wang et al. [2023a] of the form:

```
When {A} and {B} went to the store, {B} gave a book to
```

where A and B are replaced by single-token names. Here, the positive target token is A while the negative is B. Corrupted inputs are of the form:

```
When {A} and {B} went to the store, {C} gave a book to
```

D.3 Greater Than

The greater than (GT) task consists of finding two-digits tokens n greater than some previously mentioned one m. It contains templates of the form:

```
The {noun} took place between {start} and {end_century}
```

where *noun* is replaced by some event, *start* is replaced by a year, and *end_century* is replaced by the century digit of *start*. Postive target tokens are any two digit number greater than the last two digits of *start*, while negative target tokens are smaller numbers. Corrupted inputs modify the century and decade digits.