Identifying Subnetworks

Sondre Wold

June 11, 2024

Abstract

Studies that dissect the hidden representations of deep neural networks are commonplace in machine learning research. A long-standing problem within the field has been to determine to what extent these representations are modular and specialized. For example, if a model is trained on an arithmetic dataset, is there a part of the model that computes addition, and is this part distinguishable from the part that computes subtraction? Recently, the field of mechanistic interpretability has become a popular approach for answering such questions. This is typically done by identifying so-called circuits, which are subnetworks that correspond to the individual operations involved in solving a task. This functional definition of modularity, however, is also found under other names and by other methods than those typically used in mechanistic interpretability. This work attempts to synthesize existing methods for identifying such subnetworks under a shared vocabulary, with an emphasis on applications within NLP.

1 Introduction

The work surveyed in this article are all, in some way or another, concerned with the following question: do deep learning models learn solutions to problems that are modular and specialized? And if so, can we find evidence of this if we inspect the latent space? This is especially relevant for tasks where there exists a compositional solution that follows from the application and combination of a set of individual functions or smaller units of computation, such as in different types of reasoning, multi-hop question answering and code verification. This question can also be formulated more formally: Given a model M parameterized by $\theta \in \mathbb{R}^d$ and a target task T that is composed of a set of i distinguishable subtasks ST_i, is there a set of parameters $\hat{\theta} \subset \theta$, that can solve ST_i with a comparable performance to the overall model, so that $P(x|M_\theta) \approx P(x|M_{\hat{\theta}})$ for $x \sim ST_i$?

This question is closely related to the Lottery Ticket Hypothesis (Frankle and Carbin, 2019), which states that for a randomly initialized network there exists a subnetwork that is initialized such that it matches the performance of the overall network when trained for the same number of iterations. In this work, however, we are not interested in the cases where $\hat{\theta}$ is a winning ticket with respect to the whole task, but the cases where it is clear that $\hat{\theta}$ is responsible for computing one functional aspect of a task that involves multiple functions.

Because this question is central to different research directions within machine learning there exists many descriptions of $\hat{\theta}$ that are similar. Each of these directions also have their own methods for identifying and separating $\hat{\theta}$ from θ. In this work, we try to systematize and synthesize these descriptions and methods under a common notation and framework.

2 Descriptions

This section presents common ways of describing $\hat{\theta}$.

Subnetwork Csordás et al. (2020) uses the terms *module* and *subnetwork* interchangeably to refer to $\hat{\theta}$. Here, a subnetwork is defined as a subset of θ that is responsible for performing a specific function within an overall task. The same functional definition is used in Lepori et al. (2023) but there exclusively under the name subnetwork. This description of a subnetwork is related to the one used in works on pruning of deep neural networks. For example, Savarese et al. (2020) defines a subnetwork as given by a binary *mask* $m \in \{0,1\}^d$, where θ_i is kept if $m_i = 1$ and removed otherwise. This means that while M_θ has d parameters, $M_{\hat{\theta}}$ has effectively $||m||_1$ parameters. This method for separating $\hat{\theta}$ from θ is used in multiple works that try to isolate functional subnetworks. Section 3.1 describes existing methods for finding m under this description.

Cluster Watanabe (2019) uses the same functional definition of modularity, but uses the term *cluster* to refer to a set of feature vectors that are the most influential on the output of a model for a set of specific inputs. This term is also used by Casper et al. (2022), who defines a cluster to be a subset of the network when viewed as a computational graph (with neurons being the node abstraction). These clusters are analyzed with respect to their *local specialization*, where the goal of the analysis is to determine to what extent certain clusters translate to functional abstractions from the target task.

Subset In contrast to the functional definition, Ansell et al. (2022) uses the term *subset* to refer to the parameters of M that are the most influential on a general finetuning task–without focusing on the individual functions that this task might be composed of. This definition is closely related to works on efficient finetuning, such as adapters (Houlsby et al., 2019), where additional parameters are inserted into M. As these parameters are not part of the original model, these methods fall out of scope for this survey.

Circuit The term *circuit* is the standard in the field of mechanistic interpretability and is commonly described as a subset of of a network when viewed as a computational graph (Conmy et al., 2023; Nanda et al., 2023; Wang et al., 2023). Most works in mechanistic interpretability focuses on the Transformer, where circuit is a path along the edges of the graph that corresponds to the residual stream of the model. The nodes of the graph typically represent attention heads or nodes from MLPs, i.e model components, but could also represent more fine-grained elements.

3 Identification methods

In this section we discuss existing methods for identifying $\hat{\theta}$ from θ.

3.1 Masks

3.1.1 Differentiable weight masks

Csordás et al. (2020) proposes a method for training binary weight masks over θ. Their method requires a set of subtasks ST_i that correspond to the functions required to solve T. The first step is to train M_θ on samples from T. Next, they train a mask m on samples from ST_i while keeping θ frozen. The resulting mask reveals the parameters $\hat{\theta}_i$ responsible for solving the functionality for the samples in ST_i

The mask m is initialized as a set of learnable logits $l_i \in \mathbb{R}$, where $i \in [i, N]$ for N weights in θ. l_i is initially set to 0.9 for each i in order to have a high probability of keeping weights. During training, l_i is regularized such that the probability for weight θ_i not being masked out during inference is high if θ_i is necessary for solving ST_i. The regularization term r is set as $r = \alpha \sum_i l_i$, where α is a hyperparameter that controls the strength of the regularization. The mask training procedure is based on sampling. For each l_i, a sample $s_i \in [0, 1]$ is drawn from the mask as follows:

$$s_i = \sigma((l_i - \log(\log U_1 / \log U_2)/\tau), \tag{1}$$

with $U_1, U_2 \sim U(0,1)$, and where τ is a hyperparameter and σ is the sigmoid function. s_i is then gated to become the final binary mask, b_i. This is done with a straight-through estimator, which allows for estimating the gradient of threshold functions—like the one needed here to turn the continuous s_i into the discrete b_i.[1] The authors sample 4-8 binary masks per batch and apply it to different parts of the batch. After training, the mask is applied to M_θ through elementwise multiplication of the mask with the original weights: $\theta_i \odot b_i$, revealing $\hat{\theta}$ as those parameters that are not set to zero from this multiplication.

Lepori et al. (2023) uses almost the exact same approach as Csordás et al. (2020) but with a different and simpler masking technique. Their approach relies on a pruning technique called *continuous sparsification* (Savarese et al., 2020), which the authors claim is both deterministic and better at finding sparser subnetworks than the one used in Csordás et al. (2020). This method uses l_0 regularization (Louizos et al., 2018) to find sparse weight masks by maximising the number zero-elements in the masks. Given a model M_θ that is trained to solve T, the first step is to initialize the mask m as a set of parameters with the same dimensionality as θ. The next step is to train mask m_i on samples from ST_i while keeping θ frozen, optimizing for the following function:

$$\min_{\theta \in \mathbb{R}^d, m \in \mathbb{R}^d} L(M_{\theta \odot m}(x)) + \lambda * ||\sigma(\beta * m)||_1, \tag{2}$$

where L is the cross entropy, $x \sim ST_i$, σ is the sigmoid function applied elementwise, and λ is hyperparameter that effectively controls the balance between the loss and the number of zero-elements in m. β is a parameter that makes it possible to approximate a threshold function, like the straight-through gradient estimator used in Csordás et al. (2020), deterministically. When $\lim_{\beta \to \infty}$, $\sigma(\beta * m)$ approximates the heaviside function:

$$H(S) = \left\{ \begin{array}{l} 0, s < 0 \\ 1, s > 0 \end{array} \right\}, \tag{3}$$

while for $\beta = 1$ we have $\sigma(\beta * m) = \sigma(m)$. β is increased linearly during training. During inference, the mask is made binary and applied elementwise with the original network by substituting $\sigma(\beta * m_i)$ with $H(m)$. This can

[1] There was a lot of details here that I did not quite understand, but I think this should explain the gist of it at least

also be used to locate the subnetwork responsible for computing ST_i: $\hat{\theta}_i = M_{\theta \odot H(m_i)}$

3.1.2 Lottery Ticket Sparse Fine-Tuning

Another way of identifying masks is the Lottery Ticket Sparse Fine-Tuning approach proposed by Ansell et al. (2022). After finetuning a pretrained network on a target task, they identify the subset of parameters that changed the most during this training phase. Given a pretrained model M parameterized by θ^0, finetuning M on a target task yields the parameters θ^1. Parameters are then ranked according to their greatest absolute difference: $|\theta_i^1 - \theta_i^0|$. A binary mask is then constructed by selecting the top K parameters and setting all elements in $\theta_{i \in K}$ to 1 and $\theta_{i \notin K}$ to 0, which gives $\hat{\theta}$. A similar approach is also used in Frankle and Carbin (2019).

3.2 Clustering

Watanabe (2019); Casper et al. (2022)

3.3 Circuits

(Conmy et al., 2023; Nanda et al., 2023; Wang et al., 2023)

4 Applications

References

A. Ansell, E. Ponti, A. Korhonen, and I. Vulić. Composable sparse fine-tuning for cross-lingual transfer. In S. Muresan, P. Nakov, and A. Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1778–1796, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.125. URL https://aclanthology.org/2022.acl-long.125.

S. Casper, S. Hod, D. Filan, C. Wild, A. Critch, and S. Russell. Graphical clusterability and local specialization in deep neural networks. In *ICLR 2022 Workshop on PAIR2Struct: Privacy, Accountability, Interpretability, Robustness, Reasoning on Structured Data*, 2022.

A. Conmy, A. Mavor-Parker, A. Lynch, S. Heimersheim, and A. Garriga-Alonso. Towards automated circuit discovery for mechanistic interpretability. *Advances in Neural Information Processing Systems*, 36:16318–16352, 2023.

R. Csordás, S. van Steenkiste, and J. Schmidhuber. Are neural nets modular? inspecting functional modularity through differentiable weight masks. In *International Conference on Learning Representations*, 2020.

J. Frankle and M. Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJl-b3RcF7.

N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pages 2790–2799. PMLR, 2019.

M. Lepori, T. Serre, and E. Pavlick. Break it down: Evidence for structural compositionality in neural networks. *Advances in Neural Information Processing Systems*, 36:42623–42660, 2023.

C. Louizos, M. Welling, and D. P. Kingma. Learning sparse neural networks through l_0 regularization. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=H1Y8hhg0b.

N. Nanda, L. Chan, T. Lieberum, J. Smith, and J. Steinhardt. Progress measures for grokking via mechanistic interpretability. In *The Eleventh*

International Conference on Learning Representations, 2023. URL https://openreview.net/forum?id=9XFSbDPmdW.

P. Savarese, H. Silva, and M. Maire. Winning the lottery with continuous sparsification. *Advances in neural information processing systems*, 33: 11380–11390, 2020.

K. R. Wang, A. Variengien, A. Conmy, B. Shlegeris, and J. Steinhardt. Interpretability in the wild: a circuit for indirect object identification in GPT-2 small. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=NpsVSN6o4ul.

C. Watanabe. Interpreting layered neural networks via hierarchical modular representation. In *Neural Information Processing: 26th International Conference, ICONIP 2019, Sydney, NSW, Australia, December 12–15, 2019, Proceedings, Part V 26*, pages 376–388. Springer, 2019.