Constructing Neural Network Parameters with Downstream Trainability

Jingyi Chen[1]

July 31, 2024

ABSTRACT

The pretrain-finetune paradigm is shown to be beneficial and is widely adopted. However, the reason why it works has not been fully understood, especially when using the lenses of mechanistic interpretability and constructions. From another perspective, pre-training is a common solution to the problem of providing a good neural network initialization or powerful inductive bias. Nevertheless, it would be great if the inductive bias could be directly and accurately controlled, instead of indirectly controlled via losses or datasets. In this work, we propose to construct the neural network parameters manually and train (fine-tune) them on downstream tasks afterward. After that, the network can be mechanistically interpreted or utilized. The source of the manual construction is the combination of reverse engineering and human thinking. We conduct very preliminary experiments to demonstrate the workflow. Our approach remains highly interpretable with an improvement of 6.1% top-1 and 9.1% top-10 accuracy in the IsarStep-small dataset compared with the random initialization baseline. Compared with the pre-train baseline, our demonstration has higher interpretability and only uses 18% modules while achieving the majority of accuracy. The limitations of the experiments and the approach are also discussed. This is not a real paper, but a short informal report.

1 Introduction

Pre-training with fine-tuning is known to achieve high performance and is widely used [1, 2]. Researches have been conducted to understand why it works since its emergence [3, 4]. One stream of work utilizes the lenses of mechanistic interpretability [5], which reverse-engineers the neural network to understand its implementation mechanisms. In addition, as Richard Feynman famously said, "what I cannot create, I do not understand". If we can construct network parameters manually, deeper insights may be obtained. Some previous studies constructed network weights, but no fine-tuning was conducted on downstream tasks. Though there have been extensive prior arts, it still needs further research to fully understand the reason why the pretrain-finetune paradigm works, especially with the aid of constructions.

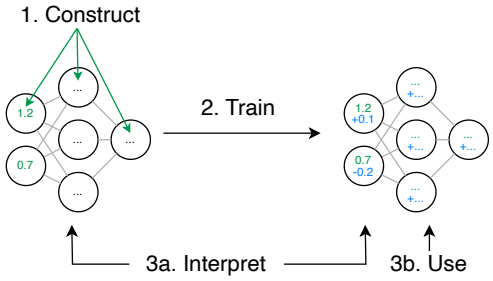

Figure 1: Overview. The parameters of the neural network are constructed manually by both reverse engineering and human thinking, and is trained (fine-tuned) afterward. Then, the network can be either interpreted or utilized.

In another perspective, consider the problem of providing good network initialization or inductive bias, which is beneficial for downstream tasks. One approach is to initialize the network using pre-trained parameter weights [1, 2], which is also believed to enhance the inductive bias [6, 7]. It is widely used and achieves great performance, especially in the large language model era [8]. However, there remains a small potential shortcoming: it can only be controlled indirectly, such as by changing the loss terms and datasets. We cannot directly and accurately specify the inductive bias.

As an attempt to alleviate the two limitations mentioned above, we propose the approach in figure 1. The network parameters are constructed manually, which is then used for training (fine-tuning, if we treat construction as an

[1]ch271828n@gmail.com; code will be released at `https://github.com/fzyzcjy/constructions`.

analogy of pre-training), followed by mechanistic interpretation or direct usage. The source of the manual construction, informally speaking, is the combination of the intelligence of nature and human beings. On one hand, since it may be too hard to manually construct complex network functionalities at the current stage, mechanistic interpretability tools can be utilized to reverse engineer an existing neural network to inspire manual construction. On the other hand, the ideas coming from humans can also be integrated into the constructed parameters.

This approach has the advantage of accurate direct control and full explainability (white box). Firstly, we can control every detail of the neural network directly in an accurate manner, because all parameter weights are constructed manually without any potential indirections. Secondly, the neural network becomes a white box, since all parameters are created by ourselves instead of learned and can be explained in full detail. We conducted some very preliminary experiments which demonstrate the process. Our approach keeps the interpretability high while improving 6.1% top-1 and 9.1% top-10 accuracy on the IsarStep-small task when compared with the random initialization baseline. Compared with pre-training, our method only uses 18% of the modules, has higher interpretability, while achieving the majority of the accuracy metrics. However, we admit the limitations of both the experiments and the approach, primarily being too small and too weak, and discuss in full detail. Nevertheless, we hope this direction may be somehow useful for understanding the pretrain-finetune paradigm as well as providing better inductive bias and initialization.

2 Related Work

Understanding the pretrain-finetune paradigm It has been widely adopted to pre-train a neural network using upstream datasets and then fine-tune it on downstream tasks. Since the emergence of this paradigm, researchers have been attempting to explain its effectiveness. Early works [3, 4] attributed the improvements to better regularization. The understanding was deepened by considering the geometry of per-sample latent spaces [9], examining the embedding space [10], constructing simple synthetic datasets [11], exploring the weight space containing all possible trained networks [12], measuring intrinsic dimensions after pre-training [13], and localizing the parameters corresponding to task-specific skills [14]. Utilizing the lenses of mechanistic interpretability [5], neural networks were trained on the task same as construction with modified model components [15] or data distribution [16] and then were reverse-engineered. The change of entity tracking circuits during fine-tuning of large language models (LLM) was also examined [17] via mechanistic interpretability. Although there exist extensive previous studies, we still need more research to fully understand the reason of the effectiveness.

Constructing network parameters Neural network weights were constructed for specific algorithms, such as simulating graph algorithms [18], recognizing a subset of counter languages [19], understanding automata [20], and solving dynamic programming problems with chain-of-thought [21]. For general-purpose tasks, parameters were constructed to make neural networks simulate Turing machines, but requires special network architectural properties [22, 23, 24] or external "punchcard" and memory [25]. For more typical scenarios of Transformers, a domain-specific language was introduced to construct weights [26], and the translation process was further automated in [15]. Though the constructed weights were beneficial for mechanistic interpretation, they cannot be used for fine-tuning on downstream tasks.

Inductive bias and initialization via pre-training Initializing neural networks with pre-trained parameter weights has been widely adopted. From the aspect of inductive bias, pre-training was believed to enhance the inductive bias [6, 27, 7], and was easier than constructing new neural network architectures when injecting inductive bias of abstract concepts [6]. Pre-training was shown to be beneficial on encoder-decoder Transformers [28, 29], encoder-only ones [1, 30], and decoder-only architectures [2, 31]. Other than real large-scale corpus, synthetic datasets were utilized during pre-training as well, such as formal languages [32], logic primitives [6], its simplified versions [11], and permuted trees [33]. On summarization tasks, synthetic datasets were reported to nearly match real datasets in the aspect of pre-train performance [34]. Though these approaches achieve great performance, there remains a small improvable point: it cannot be directly and accurately controlled to add the inductive bias we want. Instead, it needs to be controlled indirectly, such as by changing the dataset constructions and loss terms.

Mechanistic interpretability Similar to how programmers reverse-engineer a binary executable, the goal of mechanistic interpretability [5] is to understand the underlying mechanism of how neural networks implement functionalities. Various algorithms have been invented to aid the process, which can be utilized as tools for the approach proposed in this paper. Activation patching [35, 36] considers a corrupted run and a clean run. The significant component can be detected when replacing the activation of the corrupted run with the clean run corrects the answer. Path patching [37, 38] is a variant of activation patching, and patches the path between components instead of component activations. Knockout (ablation) [39, 37, 40] removes a part of the neural network and examines the change in outputs, where a large degradation in performance indicates the importance of the module. Besides providing toolkits for manual reverse-engineering, prior works also attempted to automate the process. The computational graph of a functionality can be automatically discovered [41], and network weights can be converted to Python programs for a class of restricted Transformers

[42]. Though they cannot handle thorough interpretation automatically yet, this line of research is also promising for enhancing the reverse-engineering part of this paper.

3 Methods

As is shown in figure 1, the neural network weights are constructed manually. After construction, it is then trained on downstream tasks. If the construction is considered as an analogy of pre-training, the aforementioned training can be regarded as fine-tuning. Given the trained (fine-tuned) neural network, it can be interpreted or directly used. If the network is reverse-engineered, it may be possible to gain insights into the pretrain-finetune paradigm, with the aid of direct construction of "pre-trained" model weights. If the network is directly used, the method provides a direct and accurately controllable way to inject inductive bias.

There are two primary sources for the construction, informally categorized as "nature" and "humans". Firstly, the construction can be inspired by the "nature". It may be too challenging to construct weights for complicated scenarios fully manually at this stage, because we may not yet have enough experience to know how to construct network weights in a way that can be further trained (fine-tuned). Instead, the "nature" (network weights by pre-training) can be utilized as a teacher and source of inspiration. Nevertheless, after gaining enough understanding of how networks should be constructed in a finetune-friendly way, we may be able to outperform the teacher and not need to refer to the reverse-engineered results in the future.

Secondly, the weight can also be constructed with human thinking. For example, suppose the goal is to initialize the network such that it achieves high performance on specific tasks. Then, we can inject our understanding of the downstream task into the neural network by constructing a series of cooperating MLP layers, attention layers, and embeddings that implement our thoughts. As another instance, if the target is to make the network simpler or more mechanistically interpretable, we can sacrifice performance to gain more interpretability or a smaller number of constructed modules when choosing between multiple possible construction details.

4 Experiments

In this section, we discuss the details of the experiments, the construction details of the weights, and the results of experiments.

4.1 Experiment Details

We admit the limitations and potential future directions of the experiments as follows. Firstly, due to limited computational power, the scale of the experiment is very small, so the results may not transfer to other scenarios. Nevertheless, the scale is unrelated to the fact that the proposed method allows exact whitebox control of the network, and it may serve as a toy model or proof of concept. Secondly, for the part of constructing with human thinking, we only demonstrate with the goal of enhancing interpretability, and do not conduct experiments on the target of improving performance. This is because the latter shares some similarity with constructing new neural architectures, and thus may require many trial-and-error experiments, i.e. a nontrivial amount of computational power. Thirdly, it should be possible to let the constructed model achieve performance similar to the pre-trained one, since we can mimic the reverse-engineered results carefully; even if we cannot, the failure itself may also reveal insights. Last but not least, no experiments are conducted utilizing the construction and fine-tuning pipeline to do mechanistic interpretability research about pre-training and fine-tuning. For the last two points, we skip them because the current experiment is too small to reveal something serious for the real world. If there were enough computational power, experiments could be conducted on more interesting datasets.

The experiment setup primarily follows LIME [6]. The fine-tuning task is chosen to be IsarStep [43] used by LIME [6]. It is a benchmark aiming to test high-level mathematical reasoning abilities, and contains theorems from undergraduate to research-level extracted from proofs written in the Isabelle proof assistant [44]. The task of IsarStep is to output an intermediate proposition given the surrounding proofs as inputs. To reduce the computational power required, we filter the dataset with a sequence length of less than 128 for inputs and 32 for outputs, and this is referred to as IsarStep-small in this paper. The task used as a reference of pre-training is the set deduplication problem introduced in [11], since it is reported to achieve similar performance as the task in LIME while being simpler. The inputs of this task are random token sequences which can contain duplicated tokens, and the outputs are the sequence with deduplicated tokens. We follow the original paper and choose one million as the number of training samples. We choose the Transformer architecture with 6 encoder and decoder layers, respectively, 8 attention heads, 512 hidden

size, and 2048 FFN dimension. We use the Adam optimizer [45] with learning rate $1 \cdot 10^{-4}$, dropout rate 0.1, label smoothing 0.1, and 8000 linear warmup steps.

To evaluate a method, the neural network is firstly initialized using the approach being evaluated, such as random initialization, using pre-trained weights, or manually constructed. Then, it is fine-tuned on the aforementioned IsarStep-small dataset. We use top-1 and top-10 accuracy metrics as is defined in LIME [6]. In other words, a beam search of width 10 is conducted, and the top-k accuracy is defined to be the percentage of the correct sequences appeared in the best k output sequences. We also report token-level accuracy during teacher forcing, because the RASP/Tracr baseline does not have nontrivial accuracy values on the aforementioned metrics. Since we choose mechanistic interpretability and simplicity as the goal of constructing parameters via human thinking, we also evaluate the interpretability (higher is better) as well as the number of constructed layers and modules (lower is better). For simplicity, vocabulary embedding, positional embedding, attention sublayer and MLP sublayer are considered as modules, but a more complicated counting approach has similar results.

Random initialization (no pre-training), pre-training, and RASP/Tracr are chosen to be baselines. The no pre-training baseline initializes network parameters using the default mean and variance. The pre-training is conducted using the dataset aforementioned. The RASP/Tracr baseline constructs the network parameters by converting the network specified in the RASP language into weights. LayerNorm [46] is known to be of importance for network optimization, but Tracr does not support it. In order to make the RASP/Tracr baseline fairer, both the original version and the version with extra LayerNorms are tested.

4.2 Weight Construction

In this experiment, we choose the network pre-trained on the set deduplication task as the reverse engineering target, and define the human thinking goal to be maximizing the simplicity and mechanistic interpretability of the weights. Following the aforementioned method, we firstly reverse-engineer the pre-trained neural network for inspiration. Many of its attention heads as well as MLP neurons are nontrivial to understand, and do not have a clear or easily interpretable meaning. Therefore, we do not blindly copy and paste its details to avoid violating the human thinking goal, but instead construct our weight patterns and use them only as a rough reference.

Our construction consists of three attention sublayers and one MLP sublayer. The first attention sublayer in the encoder computes whether a token needs to be copied or skipped. The query and key matrices are constructed such that the attention is higher when the query token and key token are the same, when the key position is on the left of the query position, and when the key is begin-of-sentence (BOS). The last one is implemented to allow attention to be on BOS when it does not have a valid token to focus on. The value and output matrices simply mark whether the token should be copied. The second attention sublayer in the encoder, if described at a high level, looks at the nearest token on the left that needs to be copied, and copies the key's vocabulary to the query's residual stream for future usage. This is implemented by making the key and query matrices read the outputs of the previous attention sublayer to detect whether to copy, and observing the differences between query and key token positions. After that, an MLP sublayer on the decoder part is constructed, which clears up and transforms the residual stream filled by the input embeddings. Without this, when the unembed matrix is multiplied by the residual stream, it will output a large value on the same token (for example) regardless of other factors. Lastly, a cross-attention sublayer in the decoder is created. The key matrix copies the vocabulary information from the residual stream to activation space, and the query matrix reads the output of the previous attention sublayer. The output of the first attention sublayer is also used as a gate to avoid looking at tokens that do not need to be copied. The value and output matrix simply copies the vocabulary of the key token.

4.3 Experiment Results

The results of the experiment are shown in table 1. When comparing our method with random initialization, ours outperform theirs by 6.1% top-1 and 9.1% top-10 accuracy, while still maintaining full interpretability. As for the pre-train baseline, a shortcoming to our approach is that we have not made it achieve performance similar to the baseline yet, though theoretically it may be, which is explained in the limitation paragraph above. Nevertheless, it has much higher interpretability (fully interpretable) and the construction is more condensed (6 vs 33), which demonstrates our goal of adding human thoughts to make it more interpretable.

The RASP/Tracr baseline is experimented with both the original version and one with LayerNorm enabled. probably because it does not have LayerNorm [46]. However, the approach still performs worse than random initialization, showing that it is probably designed for direct usage instead of downstream task fine-tuning.

Table 1: Experiment results. The interpretability column represents how easily the network weights can be interpreted; #Con. layers and modules mean the number of layers and modules constructed, respectively (lower is better); the top-1 and top-10 are full-sequence accuracy; top-1, top-10, and token accuracy are all in percentage units. Each row represents initializing the neural network with an approach, and then fine-tune it using the IsarStep-small dataset and measure the metrics. As can be seen, our method outperforms random initialization and RASP/Tracr baseline by at least 6.1% top-1 and 9.1% top-10 accuracy. When compared with the pre-train baseline, our approach has not achieved similar performance, but has higher interpretability and more condensed construction (only 18% of the modules), demonstrating the fulfillment of the human thinking goal.

Method	Interpretability	#Con. Layers	#Con. Modules	Top-1	Top-10	Token Acc.
Random initialization	High	-	-	16.1	27.6	87.4
Pre-train [11]	Low	12	33	25.7	39.5	89.9
RASP/Tracr [26, 15]	High	4	6	0.0	0.0	75.9
Ours	High	4	6	22.2	36.7	89.5

5 Discussion

In this report, we proposed that neural network parameters can be constructed manually, and the network is trained (fine-tuned) on downstream tasks afterward, followed by interpretation or usage of the network. The construction can be inspired by reverse-engineering existing networks as well as direct human thinking. We conducted proof-of-concept experiments demonstrating the approach, which outperform baselines by performance or interpretability.

The limitations of this report are evident. On one hand, in section 4.1, we discussed the limitations of the experiments. On the other hand, the proposed approach may not work for large-scale pre-training like the ones commonly used in the LLM era, because the current mechanistic interpretability tools have not been powerful enough to fully reverse-engineer models of such scale. However, the approach currently may serve to understand small-scale models which may also provide insights, and the reverse-engineering may become a reality with the development of the mechanistic interpretability field in the future. Looking into the far future, it may even not be impossible to create automatic decompilers and compilers, such that existing pre-trained models can be automatically decompiled, and human insights can be automatically compiled and merged into it, creating more powerful neural networks combining the intelligence of both nature and humans.

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. URL http://arxiv.org/abs/1810.04805.

[2] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. .

[3] Dumitru Erhan, Pierre-Antoine Manzagol, Yoshua Bengio, Samy Bengio, and Pascal Vincent. The Difficulty of Training Deep Architectures and the Effect of Unsupervised Pre-Training. In *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, pages 153–160. PMLR, . URL https://proceedings.mlr.press/v5/erhan09a.html.

[4] Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why Does Unsupervised Pre-training Help Deep Learning? 11(19):625–660, . ISSN 1533-7928. URL http://jmlr.org/papers/v11/erhan10a.html.

[5] Elhage Nelson. A Mathematical Framework for Transformer Circuits. URL https://transformer-circuits.pub/2021/framework/index.html.

[6] Yuhuai Wu, Markus N. Rabe, Wenda Li, Jimmy Ba, Roger B. Grosse, and Christian Szegedy. Lime: Learning inductive bias for primitives of mathematical reasoning. In *International Conference on Machine Learning*, pages 11251–11262. PMLR, .

[7] Tianyang Lin, Yuxin Wang, Xiangyang Liu, and Xipeng Qiu. A Survey of Transformers. URL http://arxiv.org/abs/2106.04554.

[8] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. LLaMA: Open and Efficient Foundation Language Models. URL https://arxiv.org/abs/2302.13971v1.

[9] Matthew B. A. McDermott, Brendan Yap, Peter Szolovits, and Marinka Zitnik. Structure-inducing pre-training. 5 (6):612–621. ISSN 2522-5839. doi:10.1038/s42256-023-00647-z. URL https://www.nature.com/articles/s42256-023-00647-z.

[10] Yichu Zhou and Vivek Srikumar. A Closer Look at How Fine-tuning Changes BERT. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1046–1061. Association for Computational Linguistics. doi:10.18653/v1/2022.acl-long.75. URL https://aclanthology.org/2022.acl-long.75.

[11] Yuhuai Wu, Felix Li, and Percy Liang. Insights into Pre-training via Simpler Synthetic Tasks, . URL http://arxiv.org/abs/2206.10139.

[12] Almog Gueta, Elad Venezian, Colin Raffel, Noam Slonim, Yoav Katz, and Leshem Choshen. Knowledge is a Region in Weight Space for Fine-tuned Language Models. URL https://openreview.net/forum?id=vq4BnrPyPb.

[13] Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. Intrinsic Dimensionality Explains the Effectiveness of Language Model Fine-Tuning. 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 7319–7328. Association for Computational Linguistics. doi:10.18653/v1/2021.acl-long.568. URL https://aclanthology.org/2021.acl-long.568.

[14] Abhishek Panigrahi, Nikunj Saunshi, Haoyu Zhao, and Sanjeev Arora. Task-specific skill localization in fine-tuned language models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *ICML'23*, pages 27011–27033. JMLR.org.

[15] David Lindner, János Kramár, Sebastian Farquhar, Matthew Rahtz, Thomas McGrath, and Vladimir Mikulik. Tracr: Compiled Transformers as a Laboratory for Interpretability. URL https://arxiv.org/abs/2301.05062v5.

[16] Samyak Jain, Robert Kirk, Ekdeep Singh Lubana, Robert P. Dick, Hidenori Tanaka, Edward Grefenstette, Tim Rocktäschel, and David Scott Krueger. Mechanistically analyzing the effects of fine-tuning on procedurally defined tasks. URL http://arxiv.org/abs/2311.12786.

[17] Nikhil Prakash, Tamar Rott Shaham, Tal Haklay, Yonatan Belinkov, and David Bau. Fine-Tuning Enhances Existing Mechanisms: A Case Study on Entity Tracking. URL http://arxiv.org/abs/2402.14811.

[18] Kimon Fountoulakis and Artur Back deLuca. Simulation of Graph Algorithms with Looped Transformers. URL https://arxiv.org/abs/2402.01107v2.

[19] Satwik Bhattamishra, Kabir Ahuja, and Navin Goyal. On the Ability and Limitations of Transformers to Recognize Formal Languages. URL http://arxiv.org/abs/2009.11264.

[20] Bingbin Liu, Jordan T. Ash, Surbhi Goel, Akshay Krishnamurthy, and Cyril Zhang. Transformers Learn Shortcuts to Automata, . URL http://arxiv.org/abs/2210.10749.

[21] Guhao Feng, Bohang Zhang, Yuntian Gu, Haotian Ye, Di He, and Liwei Wang. Towards Revealing the Mystery behind Chain of Thought: A Theoretical Perspective. URL https://openreview.net/forum?id=qHrADgAdYu.

[22] Jorge Pérez, Javier Marinković, and Pablo Barceló. On the Turing Completeness of Modern Neural Network Architectures, . URL http://arxiv.org/abs/1901.03429.

[23] Jorge Pérez, Pablo Barceló, and Javier Marinkovic. Attention is Turing-Complete. 22(75):1–35, . ISSN 1533-7928. URL http://jmlr.org/papers/v22/20-302.html.

[24] Colin Wei, Yining Chen, and Tengyu Ma. Statistically Meaningful Approximation: A Case Study on Approximating Turing Machines with Transformers. URL http://arxiv.org/abs/2107.13163.

[25] Angeliki Giannou, Shashank Rajput, Jy-yong Sohn, Kangwook Lee, Jason D. Lee, and Dimitris Papailiopoulos. Looped Transformers as Programmable Computers. URL http://arxiv.org/abs/2301.13196.

[26] Gail Weiss, Yoav Goldberg, and Eran Yahav. Thinking Like Transformers. URL http://arxiv.org/abs/2106.06981.

[27] Alex Warstadt and Samuel R. Bowman. Can neural networks acquire a structural bias from raw linguistic data? URL http://arxiv.org/abs/2007.06761.

[28] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. URL http://arxiv.org/abs/1910.13461.

[29] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. 21(140): 1–67. ISSN 1533-7928. URL http://jmlr.org/papers/v21/20-074.html.

[30] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A Robustly Optimized BERT Pretraining Approach, . URL http://arxiv.org/abs/1907.11692.

[31] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 1(8):9, .

[32] Isabel Papadimitriou and Dan Jurafsky. Injecting structural hints: Using language models to study inductive biases in language learning. URL http://arxiv.org/abs/2304.13060.

[33] Zexue He, Graeme Blackwood, Rameswar Panda, Julian McAuley, and Rogerio Feris. Synthetic Pre-Training Tasks for Neural Machine Translation. URL http://arxiv.org/abs/2212.09864.

[34] Kundan Krishna, Jeffrey Bigham, and Zachary C. Lipton. Does Pretraining for Summarization Require Knowledge Transfer? In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3178–3189. Association for Computational Linguistics. doi:10.18653/v1/2021.findings-emnlp.273. URL https://aclanthology.org/2021.findings-emnlp.273.

[35] Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and Editing Factual Associations in GPT. URL http://arxiv.org/abs/2202.05262.

[36] Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart Shieber. Investigating Gender Bias in Language Models Using Causal Mediation Analysis. In *Advances in Neural Information Processing Systems*, volume 33, pages 12388–12401. Curran Associates, Inc. URL https://proceedings.neurips.cc/paper/2020/hash/92650b2e92217715fe312e6fa7b90d82-Abstract.html.

[37] 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. URL https://openreview.net/forum?id=NpsVSN6o4ul.

[38] Nicholas Goldowsky-Dill, Chris MacLeod, Lucas Sato, and Aryaman Arora. Localizing Model Behavior with Path Patching. URL http://arxiv.org/abs/2304.05969.

[39] Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. In-context Learning and Induction Heads. URL http://arxiv.org/abs/2209.11895.

[40] LawrenceC, Adrià Garriga-alonso, Nicholas Goldowsky-Dill, ryan_greenblatt, jenny, Ansh Radhakrishnan, Buck, and Nate Thomas. Causal Scrubbing: A method for rigorously testing interpretability hypotheses [Redwood Research]. URL https://www.alignmentforum.org/posts/JvZhhzycHu2Yd57RN/causal-scrubbing-a-method-for-rigorously-testing.

[41] Arthur Conmy, Augustine N. Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. Towards Automated Circuit Discovery for Mechanistic Interpretability. URL https://openreview.net/forum?id=89ia77nZ8u.

[42] Dan Friedman, Alexander Wettig, and Danqi Chen. Learning Transformer Programs. URL http://arxiv.org/abs/2306.01128.

[43] Wenda Li, Lei Yu, Yuhuai Wu, and Lawrence C Paulson. Isarstep: A benchmark for high-level mathematical reasoning.

[44] Lawrence C. Paulson, editor. *Isabelle: A Generic Theorem Prover*, volume 828 of *Lecture Notes in Computer Science*. Springer-Verlag. ISBN 978-3-540-58244-1. doi:10.1007/BFb0030541. URL http://link.springer.com/10.1007/BFb0030541.

[45] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. URL http://arxiv.org/abs/1412.6980.

[46] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer Normalization. URL http://arxiv.org/abs/1607.06450.