DISSECTING LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Understanding and shaping the behaviour of Large Language Models (LLMs) is increasingly important as applications become more powerful and more frequently adopted. This paper introduces a machine unlearning method specifically designed for LLMs. We introduce a *selective pruning* method for LLMs that removes neurons based on their relative importance on a targeted capability compared to overall network performance. This approach is a compute- and data-efficient method for identifying and removing neurons that enable specific behaviours. Our findings reveal that both feed-forward and attention neurons in LLMs are specialized; that is, for specific tasks, certain neurons are more crucial than others.

1 INTRODUCTION

In the last two years, Large Language Models (LLMs) have been shown to achieve impressive performance in a wide array of skills. These skills have huge potential to create significant benefits for humanity. However, certain abilities may carry inherent risks, both through wide access to powerful models enabling misuse by bad actors, and from misalignment of the model's goals to its user's. Elimination of high-risk skills from an LLMs repertoire could be a valuable precaution against both misuse and misalignment. Additionally, datasets may contain sensitive user information or copyrighted material (where consent was not obtained or withdrawn) which should be removed.

Machine unlearning is a field that focuses on forgetting ability on one dataset while maintaining ability on a retain dataset. In recent years a wide variety of approaches has sprung up (Nguyen et al., 2022b), (Golatkar et al., 2020). However, there are challenges in applying these methods to LLMs, since a forward or backward pass in an LLM is costly (Bender et al., 2021).

In this paper, we introduce a method we call *selective pruning*. We evaluate our method by selectively removing coding ability in LLMs. Coding was chosen due to it being a common and powerful skill with excellent datasets, but with limited risk in research settings. Our proposed method is task-agnostic, requiring only a small dataset representative of the target task, and thus, we anticipate their applicability in the removal of other potentially harmful skills, such as manipulation. Selective pruning demands only a very small amount of additional data and computational resources.

A secondary aim of this research is to gain a deeper understanding of how various abilities are interconnected within LLMs. Our aim is separability rather than sparsity per se, which is why, contrary to most pruning methods (Blalock et al., 2020), we investigated pruning neurons (structured pruning) rather than pruning weights. If capabilities can be separated on the level of neurons, then this can lead to modularity inside models. We find that certain neurons are task-specialized, and removing them dramatically decreases performance on the forget dataset while hardly affecting performance on the retain dataset.

2 RELATED WORK

Machine unlearning aims to selectively remove information corresponding to specific data points without retraining the entire model from scratch. It has applications in for example privacy protection; complying with regulations such as GDPR; and in removing outdated or incorrect information from a trained model (Bourtoule et al., 2021). For example, Tarun et al. (2021) introduce a fast yet effective machine unlearning technique, to unlearn entire classes from a classification model.

Certain machine unlearning methods geared towards neural networks are impractical to apply to LLMs. For example DeltaGrad requires storing updates based on single data items during training (Nguyen et al., 2022b), which is costly for LLMs. We instead propose a post-hoc model surgery approach, in which we calculate influence functions after training. Ma et al. (2022) introduce a technique also performing neuron masking in neural networks, but focus on unlearning specific data points, use gradient-based update techniques and do not focus on LLMs.

Behavioural control. Reinforcement Learning from Human Feedback (RLHF) Christiano et al. (2017) can suppress behaviour, but does not eradicate knowledge, as observed through adversarial prompting and jailbreaking (Deng et al., 2023). Gandikota et al. (2023) erase concepts from text-to-image diffusion models by editing model weights. A very recent approach to LLM behaviour control is activation engineering. Turner et al. (2023) introduce a method that adds a vector to activations to control the model outputs.

Pruning. ACDC is a pruning-based approach to find a sub-circuit responsible for performance on a specific dataset (Conmy et al., 2023). This method aims to automate a step in the mechanistic interpretability pipeline. ACDC is related to selective pruning in that it uses pruning on LLMs, but the method prunes weights rather than neurons and has a very different application.

Neural network pruning typically focuses on retaining capabilities with a smaller compute budget. Networks are made sparser to e.g. reduce the storage footprint of the network, the computational cost of inference, or the energy requirements of inference (Blalock et al., 2020). For example, Michel et al. (2019) prune unused attention heads without significantly impacting performance. In contrast, we prune with the *aim* of *selectively* reducing performance.

3 Selective Pruning

We introduce a machine unlearning method for LLMs. Our method performs structured pruning to a trained LLM to selectively remove capabilities from the model. We either iteratively prune nodes in the feed-forward layers or attention head layers.

The task or dataset that we aim to reduce performance on is referred to as the *forget* dataset (D_{forget}) and the task that we are optimizing for as the *retain* dataset (D_{retain}). Our method is a heuristic pruning technique and we selectively prune nodes based on their relative importance to the D_{forget} and D_{retain} datasets. A scoring function is used to determine which nodes to prune.

3.1 Importance Functions and Scoring

We notice that zero is a default value for most activations, and we base our importance functions on how much the activations deviate from this value for a specific dataset. In particular, we make the following observations: 1) The probability distributions for feedforward neuron activations has a large spike around zero (Zhang et al., 2022b) 2) Information theoretically, more information can be transferred by a node that frequently takes on many different values (high entropy) compared to one that always takes on the same value (low entropy) such as zero.

Based on these observations, we assume that for a given neuron the activations are zero for most inputs (providing the default "null" information), and occasionally non-zero to provide information when relevant. When we prune neurons, we choose which nodes to prune based on their relative importance to datasets, and we set all their activations to zero. Below we describe the statistics we use to assess importance.

Definition 1 (Importance Functions)**.** *Let n be a neuron and denote its corresponding activations by z. We define the following importance metrics relative to a dataset D*

$$I_{freq}(D,n) := \tfrac{1}{\#D} \cdot \#\{z(d) > 0 : d \in D\} \quad I_{abs}(D,n) := \tfrac{1}{\#D} \textstyle\sum_{d\in D} |z(d)|$$

$$I_{rms}(D,n) := \sqrt{\tfrac{1}{\#D} \textstyle\sum_{d\in D} z(d)^2} \qquad I_{std}(D,n) := \sqrt{\tfrac{1}{\#D} \textstyle\sum_{d\in D} \left(z(d) - \bar{z(d)}\right)^2}$$

The rationale behind these importance metrics is as follows. First, I_{freq} captures the intuition that non-zero activations are important to the output. Second, the root-mean-square and the mean of

absolute activation of the values are another way of capturing how much the activations deviate from zero. Lastly, information theoretically, the more a node's activations vary, the more information can be obtained from its activation value. The standard deviation is a way of capturing this activation value variance.

Neurons are pruned based on their importance to the retain dataset versus forget dataset.

Definition 2 (Scoring Function)**.** *Given a forget dataset D_{forget} and retain dataset D_{retain} we define the scoring function of a neuron n as*

$$\mathit{Score}(n, D_{retain}, D_{forget}) := \frac{\mathit{Importance}(D_{retain}, n)}{\mathit{Importance}(D_{forget}, n) + \epsilon}.$$

3.2 Pruning Procedure

We consider two pruning approaches: 1) pruning some set fraction of the model in one step based on the activations of the base model; and 2) iteratively pruning smaller fractions based on the activations of the partially pruned model. One rationale behind pruning iteratively is that often when pruning one part of the model, a 'backup' part of the model pops up to complete the task (McGrath et al., 2023). In this paper we focus on using iterative pruning.

Whatever layer type we prune, we use a variation of the following algorithm:

1. Calculate the importance of each neuron (in the layer that is to be pruned) for both the retain and forget datasets;
2. Calculate the scoring function for each neuron, i.e, the ratio of importances between the two datasets;
3. Choose some top fraction of neurons (ranked by their scoring function value);
4. Delete these neurons.

As a baseline we also randomly pruned layers. We do so by first deciding on what fraction of neurons to remove and then randomly selecting neurons to be pruned. In Appendix B we discuss which feed-forward and attention neurons we prune.

4 Results

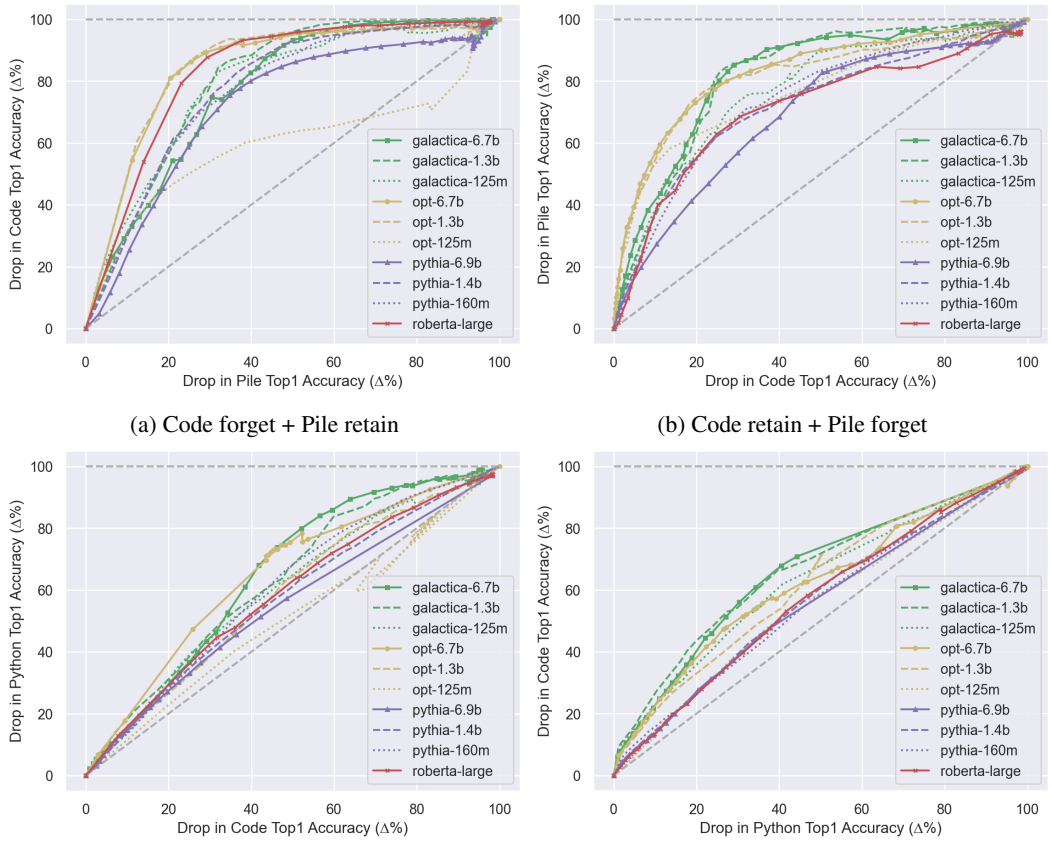
In this section, we show that we can forget a specific skill while retaining a general skill or vice versa, using our selective pruning method. In Appendix A we discuss the models and datasets we use. We prune OPT, Galactica, Pythia and Roberta models of various sizes in 50 pruning steps. In each pruning step 2% of feed-forward nodes are pruned. We investigate the forget and retain effects of pruning. In Figure 1 we show the relative performance drop and in Figure 2 we show the relative perplexity (for the same pruned models).

In Figure 1a we see that selectively pruning away neurons useful for coding or selectively pruning away neurons useful for pile performance leads to very different performances on these tasks. A trend in Figure 1 is that the larger a model the more selectively we are able to prune it. Additionally, OPT, Galactica and RoBERTa are more separable than Pythia. This is surprising as we had expected dropout would lead to more redundancy and therefore less separability. In Appendix C we find that pruning neurons from feed-forward layers is more effective than pruning attention neurons.

When forgetting code performance, in Figure 1a we see for example that for the largest OPT model (6.7B) the first reduction in code performance of around 80% requires a reduction in pile performance of 20%. Alternatively, for a retain accuracy reduction of 5% we achieve a forget reduction of around 35%. The unlearning is fairly continuous in the number of nodes pruned. For comparison, Nguyen et al. (2022a) plot a retain drop of 1.5% and a forget drop of 3% for their best method (MCU) applied to forgetting medical data from a classification model.

In Figure 2 we show how the perplexity increases as we prune away more nodes. For example, we find that in the biggest OPT model, when we forget code and retain pile, a code perplexity increase of 64x 'costs' a 2x increase in pile perplexity. The activation vector steering method ActAdd shows no increase in perplexity after steering activations more in the direction of the concept 'wedding'

(Turner et al., 2023). However, it is difficult to compare the two methods as we remove ability on a very broad task (coding) and they deal with a single word (wedding).



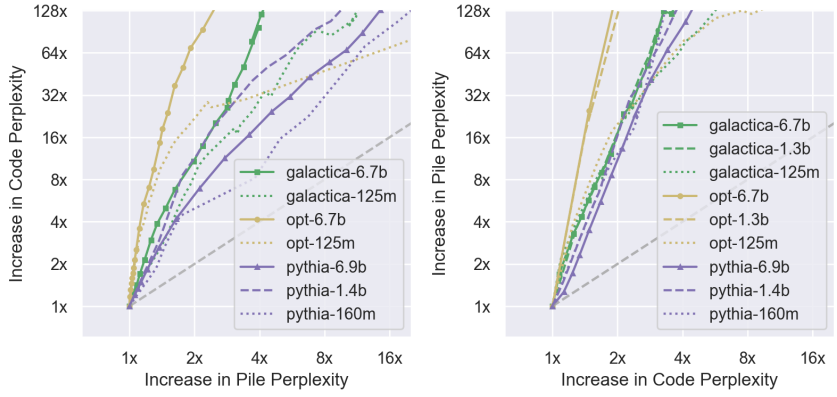
(a) Code forget + Pile retain

(b) Code retain + Pile forget

(c) Python forget + Code retain

(d) Python retain + Code forget

Figure 1: We either selectively forget or retain Code ability (top graphs) or selectively forget or retain Python ability (bottom graphs). For each graph we show the drop in forget accuracy on the y-axis, and drop in retain accuracy on the x-axis both measured in terms of accuracy. We plot a smoothed graph between the 50 pruning steps. For the biggest models, we also plot a dot for every datapoint.



(a) Code forget + Pile retain

(b) Code retain + Pile forget

Figure 2: Pile vs Code perplexity on various models. We show a smoothed curve over the course of pruning steps and for the biggest models we plot a dot at every pruning step.

5 DISCUSSION

In this paper, we introduced a method to selectively prune neurons based on those neurons' relative importance to two datasets. This machine unlearning method is effective as measured in differential drop in accuracy and as measured in perplexity, and provides a low-cost baseline for future work to compare against. We hypothesize that our method is more likely to actually remove the undesired behaviour from the model (as opposed to covering it up) compared to fine-tuning.

We find that pruning feed-forward neurons is more selective than pruning attention neurons. A potential explanation for feed-forward neurons being the best-found place to intervene in OPT and Galatica models, is that these models are trained with dropout in their feed-forward layers. We hypothesise that adding dropout to individual attention neurons during training could have the same effect on separability. Relatedly, we think our work has applications for the architecting and training of deep neural networks, specifically to constructing and training more modular networks.

Another advantage of our pruning method is that it is very quick. Pruning methods that prune weights based on computing a Hessian require computing second derivatives of n^2 parameters (Hassibi et al., 1993), where n is the number of neurons in the model. Recently, advances were made that reduced the computation time of pruning a model with weight matrices of size $d_{row} \times d_{col}$ down to $\mathcal{O}(d_{row} \cdot d_{col}^3)$ time and $\mathcal{O}(d_{col}^2)$ memory (Frantar & Alistarh, 2022), which works well for medium-size models, such as ResNet50 (25 million parameters), but quickly becomes too expensive for large language models. In contrast, we ran our experiments on a single Nvidia RTX 4090.

5.1 LIMITATIONS

Our method can only be applied to remove a capability when that capability is neatly captured by a dataset. For example, we removed coding based on a coding dataset and toxic comments based on news data labelled with toxicity. However, often we will want to remove capabilities for which we do not have a specific dataset.

The selectiveness of our pruning method relies on the separability of the capabilities of an LLM. It performs less well on, for example, Pythia (trained without dropout) and on smaller LLMs. Further work may unravel why these models seem to be less separable.

5.2 FUTURE WORK

A popular machine unlearning evaluation metric is the anamnesis index (Chundawat et al., 2023) which assesses the fine-tuning or retraining steps needed to regain the original model performance on the forget dataset. Unfortunately, retraining LLMs is costly, and so we have not evaluated our method on this retrainability metric. We think this metric would be very interesting for testing how 'fundamentally' a behaviour is removed from the model.

Furthermore, we could investigate the relationship between retained skills. For example, when we prune away coding ability, are we removing the ability to correctly handle prompts in the format that code prompts are generally given in, or are we removing internal knowledge about coding principles. This is an empirical question about how sub-skills of the model are represented and related.

Moving forward, we are excited to enhance the effectiveness of selective pruning. A notable area of exploration is the potential benefit of adding dropout to attention neurons during the training or fine-tuning phases. This could also offer advancements in our understanding of modularity.

5.3 BROADER IMPACTS

Our pruning method isolates a capability, but does not enhance or improve it. Methods that instead rely on fine-tuning to remove specific skills, can typically also be applied to increase ability on that skill, which means they may be misused by bad actors. Contrary to other approaches towards capability removal, our method is unlikely to generate systems that are more harmful than the base model.

REFERENCES

Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. On the dangers of stochastic parrots: Can language models be too big? In Madeleine Clare Elish, William Isaac, and Richard S. Zemel (eds.), *FAccT '21: 2021 ACM Conference on Fairness, Accountability, and Transparency, Virtual Event / Toronto, Canada, March 3-10, 2021*, pp. 610–623. ACM, 2021. doi: 10.1145/3442188.3445922. URL https://doi.org/10.1145/3442188.3445922.

Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. Pythia: A suite for analyzing large language models across training and scaling. *arXiv preprint arXiv:2304.01373*, 2023.

Davis W. Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John V. Guttag. What is the state of neural network pruning? In Inderjit S. Dhillon, Dimitris S. Papailiopoulos, and Vivienne Sze (eds.), *Proceedings of Machine Learning and Systems 2020, MLSys 2020, Austin, TX, USA, March 2-4, 2020*. mlsys.org, 2020. URL https://proceedings.mlsys.org/book/296.pdf.

Lucas Bourtoule, Varun Chandrasekaran, Christopher A. Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. In *42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021*, pp. 141–159. IEEE, 2021. doi: 10.1109/SP40001.2021.00019. URL https://doi.org/10.1109/SP40001.2021.00019.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

Vikram S. Chundawat, Ayush K. Tarun, Murari Mandal, and Mohan S. Kankanhalli. Zero-shot machine unlearning. *IEEE Trans. Inf. Forensics Secur.*, 18:2345–2354, 2023. doi: 10.1109/TIFS.2023.3265506. URL https://doi.org/10.1109/TIFS.2023.3265506.

Arthur Conmy, Augustine N. Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. Towards automated circuit discovery for mechanistic interpretability. *CoRR*, abs/2304.14997, 2023. doi: 10.48550/arXiv.2304.14997. URL https://doi.org/10.48550/arXiv.2304.14997.

Gelei Deng, Yi Liu, Yuekang Li, Kailong Wang, Ying Zhang, Zefeng Li, Haoyu Wang, Tianwei Zhang, and Yang Liu. Jailbreaker: Automated jailbreak across multiple large language model chatbots. *CoRR*, abs/2307.08715, 2023. doi: 10.48550/arXiv.2307.08715. URL https://doi.org/10.48550/arXiv.2307.08715.

Elias Frantar and Dan Alistarh. Optimal brain compression: A framework for accurate post-training quantization and pruning. *arXiv preprint arXiv:2208.11580*, 2022.

Rohit Gandikota, Hadas Orgad, Yonatan Belinkov, Joanna Materzynska, and David Bau. Unified concept editing in diffusion models. *CoRR*, abs/2308.14761, 2023. doi: 10.48550/arXiv.2308.14761. URL https://doi.org/10.48550/arXiv.2308.14761.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The Pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pp. 5484–5495. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.emnlp-main.446. URL https://doi.org/10.18653/v1/2021.emnlp-main.446.

Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Eternal sunshine of the spotless net: Selective forgetting in deep networks. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pp. 9301–9309. Computer Vision Foundation / IEEE, 2020. doi: 10.1109/CVPR42600.2020.00932.

B. Hassibi, D.G. Stork, and G.J. Wolff. Optimal brain surgeon and general network pruning. In *IEEE International Conference on Neural Networks*, pp. 293–299 vol.1, 1993. doi: 10.1109/ICNN.1993.298572.

Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, Joe Davison, Mario Šaško, Gunjan Chhablani, Bhavitvya Malik, Simon Brandeis, Teven Le Scao, Victor Sanh, Canwen Xu, Nicolas Patry, Angelina McMillan-Major, Philipp Schmid, Sylvain Gugger, Clément Delangue, Théo Matussière, Lysandre Debut, Stas Bekman, Pierric Cistac, Thibault Goehringer, Victor Mustar, François Lagunas, Alexander Rush, and Thomas Wolf. Datasets: A community library for natural language processing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 175–184, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. URL https://aclanthology.org/2021.emnlp-demo.21.

Zhuo Ma, Yang Liu, Ximeng Liu, Jian Liu, Jianfeng Ma, and Kui Ren. Learn to forget: Machine unlearning via neuron masking. *IEEE Transactions on Dependable and Secure Computing*, 2022.

Thomas McGrath, Matthew Rahtz, János Kramár, Vladimir Mikulik, and Shane Legg. The hydra effect: Emergent self-repair in language model computations. *CoRR*, abs/2307.15771, 2023. doi: 10.48550/arXiv.2307.15771. URL https://doi.org/10.48550/arXiv.2307.15771.

Paul Michel, Omer Levy, and Graham Neubig. Are sixteen heads really better than one? In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 14014–14024, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/2c601ad9d2ff9bc8b282670cdd54f69f-Abstract.html.

Quoc Phong Nguyen, Ryutaro Oikawa, Dinil Mon Divakaran, Mun Choon Chan, and Bryan Kian Hsiang Low. Markov chain monte carlo-based machine unlearning: Unlearning what needs to be forgotten. In Yuji Suga, Kouichi Sakurai, Xuhua Ding, and Kazue Sako (eds.), *ASIA CCS '22: ACM Asia Conference on Computer and Communications Security, Nagasaki, Japan, 30 May 2022 - 3 June 2022*, pp. 351–363. ACM, 2022a. doi: 10.1145/3488932.3517406. URL https://doi.org/10.1145/3488932.3517406.

Thanh Tam Nguyen, Thanh Trung Huynh, Phi Le Nguyen, Alan Wee-Chung Liew, Hongzhi Yin, and Quoc Viet Hung Nguyen. A survey of machine unlearning. *CoRR*, abs/2209.02299, 2022b. doi: 10.48550/arXiv.2209.02299. URL https://doi.org/10.48550/arXiv.2209.02299.

Edoardo Pona. Superposition and dropout. 2023. URL https://www.lesswrong.com/posts/znShPqe9RdtB6AeFr/superposition-and-dropout.

Ayush K. Tarun, Vikram S. Chundawat, Murari Mandal, and Mohan S. Kankanhalli. Fast yet effective machine unlearning. *CoRR*, abs/2111.08947, 2021. URL https://arxiv.org/abs/2111.08947.

Ross Taylor, Marcin Kardas, Guillem Cucurull, Thomas Scialom, Anthony Hartshorn, Elvis Saravia, Andrew Poulton, Viktor Kerkez, and Robert Stojnic. Galactica: A large language model for science. *CoRR*, abs/2211.09085, 2022. doi: 10.48550/arXiv.2211.09085. URL https://doi.org/10.48550/arXiv.2211.09085.

Lewis Tunstall, Leandro Von Werra, and Thomas Wolf. *Natural language processing with transformers*. " O'Reilly Media, Inc.", 2022.

Alex 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. URL https://doi.org/10.48550/arXiv.2308.10248.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In Anna Korhonen, David R. Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Conference of the Association*

for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers, pp. 5797–5808. Association for Computational Linguistics, 2019. doi: 10.18653/v1/p19-1580. URL https://doi.org/10.18653/v1/p19-1580.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona T. Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. OPT: open pre-trained transformer language models. *CoRR*, abs/2205.01068, 2022a. doi: 10.48550/arXiv.2205.01068. URL https://doi.org/10.48550/arXiv.2205.01068.

Zhengyan Zhang, Yankai Lin, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. Moefication: Transformer feed-forward layers are mixtures of experts. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Findings of the Association for Computational Linguistics: ACL 2022, Dublin, Ireland, May 22-27, 2022*, pp. 877–890. Association for Computational Linguistics, 2022b. doi: 10.18653/v1/2022.findings-acl.71. URL https://doi.org/10.18653/v1/2022.findings-acl.71.

A MODELS AND TASKS

In this section, we provide technical details on the pre-trained models, datasets, and task composition we use to explore selective pruning for capability-removal.

A.1 PRE-TRAINED MODELS

We work with Meta's OPT (Zhang et al., 2022a), Meta's Galactica (Taylor et al., 2022), as well as EleutherAI's Pythia models (Biderman et al., 2023). For each model type we consider three different model sizes 125M, 1.3B and 6.7B parameters[1]. The models are accessed via the Hugging Face transformer library (Taylor et al., 2022).

All three models are decoder-only large language models. Below we list some of the key differences between the architectures.

Table 1: Key differences between OPT, Galactica and Pythia.

	OPT	Galactica	Pythia	Roberta
Dropout	0.1	0.1	None	0.1
MLP activation function	ReLU	GeLU	GeLU	GeLU
MLP & Attention Biases	Yes	No	Yes	Yes

A.2 TASK DATASETS — PILE, CODE AND PYTHON

We evaluated the above models on the following datasets accessed via the Hugging Face datasets library (Lhoest et al., 2021). **Pile**, short for EleutherAI's 'The Pile' (Gao et al., 2020), is a general text dataset. In addition, we use a coding dataset referred to as **Code**, short for 'CodeParrot GitHub Code (all-all)' (Tunstall et al., 2022), as a dataset consisting of various programming languages from GitHub; and, second, **Python**, short for 'CodeParrot GitHub Code (python-all)', is the subset of the Code dataset that only contains Python code.

The Pile dataset contains around 10% code, when comparing a model's performance on Pile against code, we additionally filter out most code examples from Pile by skipping text labelled as coming from GitHub. Note that if we more comprehensively removed code from the Pile dataset then this would likely slightly improve the separability and thus our results.

In our experiments, we selectively prune away ability on one dataset (the 'forget' dataset) while maintaining high accuracy on another (the 'retain' dataset). We use the following pairs: Pile vs Code; and Code vs Python.

Our choice for a coding dataset is based on the idea that writing code is a powerful skill (with which in theory algorithms could be written that for example act on the stock market). By forgetting python ability while retaining coding ability, we aim to show that our method can also selectively prune away a dataset that 'looks similar' to the retain dataset. Note however that our method is dataset agnostic.

A.3 SAMPLES

In Section 3.1 we explain how we calculate a pruning score for neurons based on their activations. To calculate a score for each neuron, we collect a sample of 100,000 next-token predictions. The main performance metric that we use is Top1 next token prediction accuracy, which we refer to as accuracy. Additionally we use perplexity. In Appendix **??** we discuss additional metrics that are less widely used, but may be more suitable.

We considered different sample sizes (namely 10^3, 10^4, 10^5, and 10^6 samples) and found that larger samples for both activations and evaluations lead to more targeted pruning, but at the cost of the pruning process being proportionally slower: we choose 100k samples as a reasonable trade-off.

[1]The models are labelled as OPT-125M, OPT-1.3B, OPT-6.7B, Galactica-125M, Galactica-1.3B, Galactica-6.7B, Pythia-160M, Pythia-1.4B, Pythia-6.9B. Excluding biases, the true number of parameters is equivalent.

B OBJECTS OF PRUNING: FEED-FORWARD VS ATTENTION NEURONS

We prune within either the feed-forward or the attention blocks. We keep the Embedding, Positional Embedding and Output Unembedding unmodified. This is because we do not want to remove the generality of the model, or to blacklist any words in particular in other domains. We also do not want to directly modify specific dimensions of the embedding space. Because we think this is likely the wrong level of granularity for pruning most tasks, since the task might not be precisely aligned with the latent space dimensions.

Feed-Forward. We describe the feed-forward sub-layer in a decoder layer l, given a (layer-normed) input z to be: $f_l(z) = W_{OUT} \cdot \sigma(W_{IN} \cdot z + B_{IN}) + B_{OUT}$. We label the $\sigma(W_{IN} \cdot z) + B_{IN}$ as *middle sub-layer neuron* activations. We choose to prune the middle sub-layer neurons of the feed-forward network based on the idea that the key-value memories reside in those layers (Geva et al., 2021). Additionally, in Zhang et al. (2022b) they find in BERT that 'more than 80% of inputs activate less than 10% of [Feed-Forward] neurons.' This makes it likely that these neurons are highly specialized.

Feed-forward layers are pruned using the $\text{Importance}_{\text{abs}}$ metric (unless specified otherwise), i.e. a neuron was pruned based on the ratio between the importance function (in this case the average absolute activation) value on the retain dataset and on the forget dataset.

We delete a neuron in a feed-forward mid-layer by setting the input and output weights and biases, W_{IN}, W_{OUT}, B_{IN} to 0.0 for that neuron.

Attention. The main units of neurons we consider in an attention head, are the 'value' neurons and 'pre-out' neurons. The activations of the value neurons $V_i = \sum_i W_{vij} x_j$ are the directions from the output of the value matrix W_v. The 'pre-out' neuron activations $Z_i = \sum_j A_{ij} V_j$ are the directions after the values are multiplied by the attention weights $A_{ij} = \text{softmax}(\frac{Q_i \cdot K_j}{\sqrt{d}})$, but before they are returned to the residual stream through W_O.

Intervening on the 'value' and on the 'pre-out' neurons gives similar results on our metrics. In the main body of this paper, we focus on 'value' neurons to simplify the analysis. Additional details on 'pre-out' pruning performance can be found in the Appendix **??**.

There is no activation function on the value layer that maps negative pre-activations to zero. Hence the frequency importance metric is not useful in this case. We used all other three importance metrics, i.e. $\text{Importance}_{\text{abs}}$, $\text{Importance}_{\text{rms}}$ and $\text{Importance}_{\text{std}}$.

Based on a hypothesis that Singular Value Decomposition (SVD) might improve feature separation, we considered altering the weights W_v and W_o using SVD on $W_v W_o$ making their weights orthogonal to each other. We did not find this to substantially impact our results, see Appendix **??**.

To delete a 'neuron' in a value layer we remove the parameters: W_v row weights, B_v bias entry, and W_o column weights relating to that neuron. Deleting neurons in this 'value' layer avoids interfering with any of the dimensions of the residual stream. We only interfere with the adjustments made by the computational components (the attention and feed-forward layers).

C PRUNING FEED-FORWARD NEURONS MORE EFFECTIVE THAN ATTENTION NEURONS

To evaluate how effectively a method prunes, we consider the maximum difference in accuracy drop between the forget and retain datasets. The more selective a pruning method is, the larger this difference. In Figure 3 we plot the maximum difference in accuracy drop for a variety of importance functions and objects of pruning.

Previous work shows that specific abilities can be removed from LLMs by pruning away entire attention heads (Voita et al., 2019), but does not consider other objects of pruning. In Appendix **??** we investigate the effect of pruning away entire attention heads. We find that the object of pruning is crucial and that feed-forward layers are the best object to intervene on. Pruning attention neurons is effective as well, whereas pruning entire attention heads leads to poor results by comparison. In this

appendix we also show the maximum difference in accuracy drop for the reverse task of retaining Code and forgetting Pile.

In Figure 3 we find that for the tasks and models we investigate, feed-forward neurons are more specialized than attention neurons.

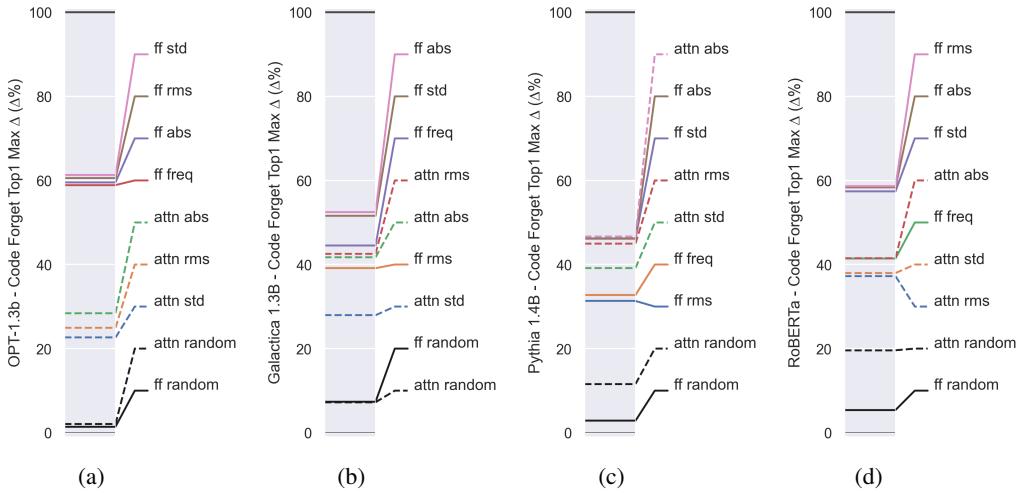


Figure 3: We evaluate methods for pruning OPT-1.3B (a), Galactica-1.3B (b), Pythia-1.4B (c), and Roberta-355M (d). We use various different importance functions (freq, abs, rms, std), on different regions of the model (feed-forward neurons, attention "pre-out neurons"). The graphs show the maximal difference between accuracy in Code and accuracy in Pile performance over 50 pruning steps, which each prune away 2% of neurons.

Pruning strategy	OPT-1.3B	Galactica-1.3B	Pythia-1.4B	Roberta-355M
Feed-forward neurons	59.6	52.4	46.2	58.3
Attention neurons	28.4	41.7	46.6	41.5

Table 2: Largest difference in Top1 accuracy drop on Code versus Pile after 50 pruning steps.

We find that the choice of importance metric (freq, abs, std or rms), is sometimes rather marginal (such as in OPT FF pruning), but can be substantial (such as in Galactica FF pruning). This suggests there could be better importance metrics we have not tried. From our research, the metric that seemed to most reliably perform well in both feed-forward (FF) and attention layers was $\text{Importance}_{\text{abs}}$, which is why we have used it in our figures and tables (unless otherwise specified).

In models trained with FF dropout (OPT and Galactica), we see in Table 2 that pruning FF neurons has a huge differential effect on performance compared to pruning of attention value neurons. In contrast, pruning Pythia FF neurons is only marginally more effective than pruning attention value neurons. This suggests that dropout during training makes a neuron more task-specific, and that adding dropout to attention value layers during training could potentially yield the same task specialisation benefits. Our finding is consistent with the finding that dropout suppresses superposition (Pona, 2023).

We focus on pruning either only feed forward layers or only attention layers. However, we expect the optimal pruning ratio and percentage of different model regions will differ per application.