Training Neural Networks for Modularity Aids Interpretability

Satvik Golechha
Independent (MATS)
zsatvik@gmail.com

Dylan Cope
King's College London & Imperial College London
dylan.cope@kcl.ac.uk

Nandi Schoots
King's College London & Imperial College London
nandischoots@gmail.com

Abstract

An approach to improve network interpretability is via clusterability, i.e., splitting a model into disjoint clusters that can be studied independently. We find pretrained models to be highly unclusterable and thus train models to be more modular using an "enmeshment loss" function that encourages the formation of non-interacting clusters. Using automated interpretability measures, we show that our method finds clusters that learn different, disjoint, and smaller circuits for CIFAR-10 labels. Our approach provides a promising direction for making neural networks easier to interpret.

1 Introduction

Interpretability is an active area of research that aims to solve both high-stake deployment constraints for fairness and robustness [17] and AI safety and trustworthiness concerns [1]. Several interpretability breakthroughs over the last few years have helped us understand the inner workings of deep networks, both via circuits [22, 19, 5] and representation spaces [25, 2]. However, applying these methods to larger models [11] and complex behaviors has been a major hurdle for interpretability [14, 8], largely due to complicated computational subgraphs or circuits, and superposition [9], i.e., networks representing more features than they have neurons.

An alternative approach is to split models into modular clusters and interpreting them separately. However, this is feasible only if the interaction between clusters is minor. In this work, we attempt to make models modular and more interpretable during training.

The main contributions of this paper are as follows:

- We modify and test prior clusterability methods to split a trained neural network layer into bipartite clusters and show that the clusters thus found are highly enmeshed and not good for our interpretability goals.
- We introduce "enmeshment loss", a regularizing term that promotes modularity during neural network training to get non-interfering clusters during model training.
- We use automated interpretability measures to show that the clusters thus obtained make the model more interpretable by (a) reducing the search-space of circuit-style analyses by reducing circuit size (Section 3.2), and (b) by producing specialized clusters for each label (Section 3.1).

Preprint. Under review.

1.1 Bipartite Spectral Graph Clustering (BSGC)

First, we show that existing clustering methods are ineffective for interpreting neural networks. For our clustering algorithm, we modify the methodology of Filan et al. [6] and use normalized spectral clustering to split a neural network layer into k different bipartite clusters.

Our clustering method, called Bipartite Spectral Graph Clustering (BSGC) is shown in Algorithm 1. The similarity matrix for BSGC can be created by either the weights of the model or the accumulated gradients.

Weight-based BSGC. Here, we use the weight matrix of a layer as the similarity matrix between neurons of adjacent hidden layers, based on the idea that neurons with strong weights connecting them can be expected to cluster well.

Gradient-based BSGC. Analyzing the gradients of each parameter during training gives us another way to cluster models. The idea is that weights that update together are likely to be part of the same circuit and connect neurons that cluster well together. We set the similarity matrix in Algorithm 1 to the average cosine similarity of the gradients of each parameter.

Algorithm 1: Bipartite Spectral Graph Clustering (BSGC)

Input: A: the similarity matrix ($m \times n$), k: #clusters
Output: U, V: bipartite clusters of input/output neurons

1. Normalized similarity matrix
$D_U, D_V \leftarrow$
$\text{diag}\left(\sum_i A_{i,\cdot}\right), \text{diag}\left(\sum_j A_{\cdot,j}\right)$
$\tilde{A} \leftarrow D_U^{-1/2} A D_V^{-1/2}$

2. Singular Value Decomposition (SVD)
$U, \Sigma, V^T \leftarrow \text{SVD}(\tilde{A}, k)$

3. KMeans clustering
$\text{U}, \text{V} \leftarrow \text{KMeans}(k, U), \text{KMeans}(k, V^T)$

return U, V

2 Clusterability Evaluation and the Enmeshment Loss

We evaluate the efficacy of a clustering method by the amount of "clusterability" or "enmeshment" in the clusters obtained, i.e., the average fraction of weights that are inside a cluster as opposed to between clusters.

Let W be the weight matrix where W_{ij} represents the edge weight between nodes i and j in the layer's input and output neurons respectively. Define U and V as the clusters for the rows and columns of A, respectively. Let $C_U(u)$ and $C_V(v)$ denote the sets of nodes in clusters $u \in \{1, \ldots, k\}$ and $v \in \{1, \ldots, k\}$, respectively. The enmeshment measure E is defined as follows:

$$E = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} |W_{ij}|^2 \cdot \mathbb{I}_{(i \in C_U(u) \wedge j \in C_V(u))}}{\sum_{i=1}^{n}\sum_{j=1}^{n} |W_{ij}|^2}, \text{ where } \mathbb{I} = \begin{cases} 1 & \text{if } i \in C_U(u) \text{ and } j \in C_V(u) \\ 0 & \text{otherwise} \end{cases}$$

In Figure 1, we see that as the number of clusters increases, the amount of enmeshment decreases, and even at $k = 2$ clusters, we get $E = 0.6$, which is substantial interference between clusters and hinders our interpretability goals.

In Section 3, we train models to be more modular by optimizing for clusterability by adding the enmeshment loss to the usual cross-entropy loss:

$$\mathcal{L} = \mathcal{L}_{CE} + \lambda \mathcal{L}_E$$

with an appropriate clusterability coefficient λ.

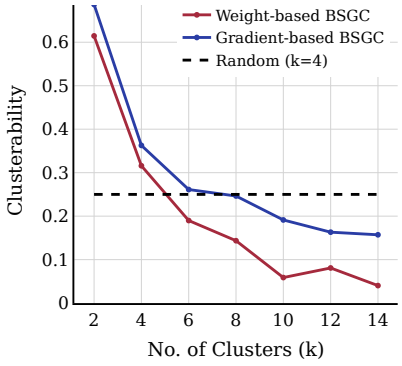

Figure 1: Clusterability (enmeshment) of the model with k bipartite clusters using Algorithm 1.

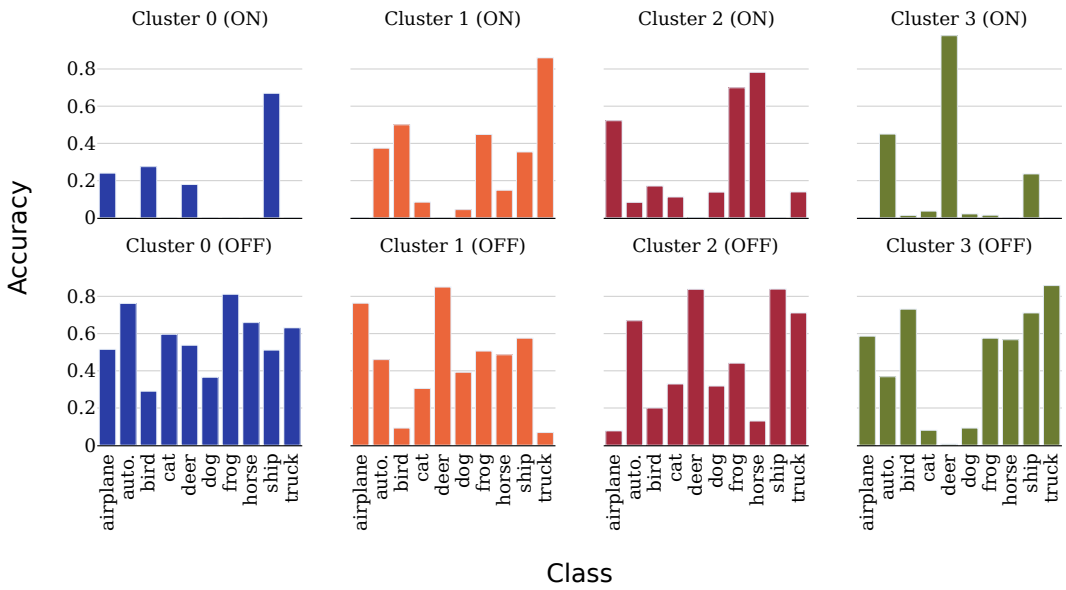

Figure 2: Class-wise accuracy for each label with clusters turned ON (top) and OFF (bottom). Note individual clusters learning near-complete circuits for various labels.

3 Training for Modularity and Evaluating Clusters for Interpretability

Our training pipeline comprises three steps:

1. Train the original model for a few steps. This helps important connections to form before we begin clustering. It can also allow "winning tickets" (as defined in the lottery ticket hypothesis [7]) to emerge.
2. Split a layer into clusters using weight-based bipartite spectral clustering (see Algorithm 1).
3. Complete the rest of the training with the "enmeshment loss" added to the cross-entropy loss to promote the clusters to be modular (see Section 2).

We experiment with vanilla MLPs on MNIST [4] and CNNs on the CIFAR-10 dataset [12]. We train for $n = 1$ epoch and then extract clusters and continue training for $n = 10$ more epochs to get a clustered model that perform similarly as the original model but with a clusterability score of 0.99. All our hyperparameters are given in Appendix A.

We evaluate our clusters and proxy their interpretability gains via two metrics: class-wise accuracy (with and without each individual cluster), and Effective Circuit Size (ECS) for each label.

3.1 Clusters Specialize in Class-level Features

Figure 2 compares class-wise accuracy of the model on the CIFAR-10 dataset with individual clusters turned OFF and ON respectively in a trained model with an accuracy $> 95\%$ for each label. Here, we define switching a cluster ON as zero-ablating the activations of all the other clusters, and switching it OFF as zero-ablating the activations of the given cluster while keeping the others the same. Thus, we get a sense of the extent to which various clusters contribute independently or with other clusters to predict a given label.

3.2 Effective Circuit Size (ECS)

Figure 3 compares the Effective Circuit Size (ECS) for each label for the clustered and un-clustered models on CIFAR-10. We use automated pruning (see ACDC [3]) to extract the circuit for each label and then compute the number of parameters in the circuit and define it to be the ECS for that

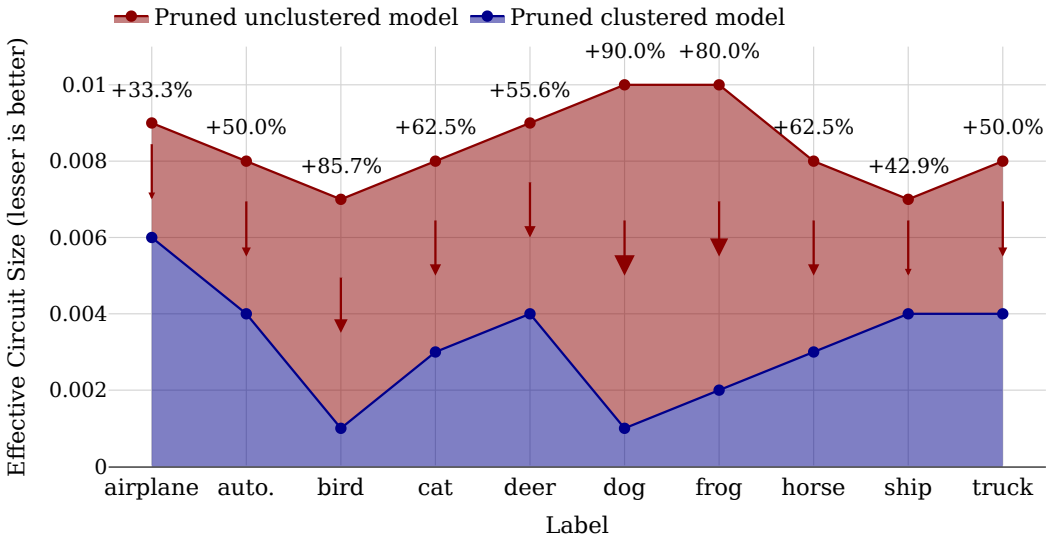

Figure 3: Percentage increase in Effective Circuit Size (ECS) for each label as a fraction of the whole model from the clustered to the un-clustered models. Larger arrows denote a larger change in ECS.

label and model. We show that an un-clustered model has on average 61.25% more parameters in its effective circuits, thus making the clustered model easier to interpret. In Appendix B, we share similar results for the MNIST dataset [4].

4 Related Work

Clustering of neural network weights is typically done either using graph properties of the weights (structural) [23, 6, 20] or using correlations between neuron activations (functional) [10, 13]. In this paper, we consider both weight-based and gradient-based clustering. MoEfication groups feedforward neurons in a pre-trained language model into clusters of 'experts' and at inference only activates the most clusters neurons [24].

Modularity metrics inspired by research in neuroscience incorporate transfer entropy [18, 21] or spatial metrics [15, 16]. Models are sometimes trained or pruned using modularity metrics [20].

Circuit Discovery is an active field of research [22, 19, 5, 3], which aims to uncover subnetworks that perform a specific functionality. In this work we use an enmeshment loss that calculates the structural connectedness or global functionality without considering specific functionalities. We evaluate the resulting modules by assessing the extent to which they have specialized on a class level, i.e. the extent to which they correspond to class-specific performance.

5 Conclusion

We show that a simple regularizer is effective at splitting a neural network layer into simpler, more interpretable clusters. We show that the average circuit size and the search space improve with more clusters without a decrease in overall performance (for the MNIST and CIFAR-10 models that we investigated).

We expect the Pareto frontier (clusterability versus performance) to be challenging as we scale to harder tasks and larger models. We are also interested in using our insights to train and align language models with modularity and see if it leads to better interpretability and control for unwanted behaviors. We hope that modularity can help with a number of mechanistic interpretability goals, and a concrete exploration into this (such as clustering attention heads or linear probes) is another future direction.

Acknowledgments and Disclosure of Funding

We would like to thank the MATS program, the organizers, funders and staff. We would especially like to thank Daniel Filan, Sandy Tanwisuth, and Matthew Wearden for the valuable discussions, feedback, and support. SG did this work when he was a MATS scholar funded by AI Safety Support.

References

[1] Bereska, L. and Gavves, E. (2024). Mechanistic interpretability for ai safety - a review. *ArXiv*, abs/2404.14082.

[2] Bricken, T., Templeton, A., Batson, J., Chen, B., Jermyn, A., Conerly, T., Turner, N., Anil, C., Denison, C., Askell, A., et al. (2023). Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, page 2.

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

[4] Deng, L. (2012). The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142.

[5] Elhage, N., Nanda, N., Olsson, C., Henighan, T., Joseph, N., Mann, B., Askell, A., Bai, Y., Chen, A., Conerly, T., et al. (2021). A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 1.

[6] Filan, D., Casper, S., Hod, S., Wild, C., Critch, A., and Russell, S. (2021). Clusterability in neural networks. *arXiv preprint arXiv:2103.03386*.

[7] Frankle, J. and Carbin, M. (2018). The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv: Learning*.

[8] Golechha, S. and Dao, J. (2024). Challenges in mechanistically interpreting model representations. In *ICML 2024 Workshop on Mechanistic Interpretability*.

[9] Henighan, T., Carter, S., Hume, T., Elhage, N., Lasenby, R., Fort, S., Schiefer, N., and Olah, C. (2023). Superposition, memorization, and double descent. *Transformer Circuits Thread*, 6:24.

[10] Hod, S., Casper, S., Filan, D., Wild, C., Critch, A., and Russell, S. (2022). Detecting modularity in deep neural networks.

[11] Kaplan, J., McCandlish, S., Brown, T., et al. (2020). Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

[12] Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images.

[13] Lange, R. D., Rolnick, D., and Kording, K. P. (2022). Clustering units in neural networks: upstream vs downstream information. *Trans. Mach. Learn. Res.*, 2022.

[14] Lieberum, T., Rahtz, M., Kramár, J., Nanda, N., Irving, G., Shah, R., and Mikulik, V. (2023). Does circuit analysis interpretability scale? evidence from multiple choice capabilities in chinchilla. *arXiv preprint arXiv:2307.09458*.

[15] Liu, Z., Gan, E., and Tegmark, M. (2023a). Seeing is believing: Brain-inspired modular training for mechanistic interpretability. *CoRR*, abs/2305.08746.

[16] Liu, Z., Khona, M., Fiete, I. R., and Tegmark, M. (2023b). Growing brains: Co-emergence of anatomical and functional modularity in recurrent neural networks. *CoRR*, abs/2310.07711.

[17] McClure, P., Moraczewski, D., Lam, K. C., Thomas, A. G., and Pereira, F. (2020). Evaluating adversarial robustness for deep neural network interpretability using fmri decoding. *ArXiv*, abs/2004.11114.

[18] Novelli, L., Atay, F. M., Jost, J., and Lizier, J. T. (2019). Deriving pairwise transfer entropy from network structure and motifs. *CoRR*, abs/1911.02931.

[19] Olah, C., Cammarata, N., Schubert, L., Goh, G., Petrov, M., and Carter, S. (2020). Zoom in: An introduction to circuits. *Distill*, 5(3):e00024–001.

[20] Patil, S. M., Michael, L., and Dovrolis, C. (2023). Neural sculpting: Uncovering hierarchically modular task structure in neural networks through pruning and network analysis. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S., 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*.

[21] Ursino, M., Ricci, G., and Magosso, E. (2020). Transfer entropy as a measure of brain connectivity: A critical analysis with the help of neural mass models. *Frontiers Comput. Neurosci.*, 14:45.

[22] Wang, K., Variengien, A., Conmy, A., Shlegeris, B., and Steinhardt, J. (2022). Interpretability in the wild: a circuit for llama object identification in gpt-2 small. *arXiv preprint arXiv:2211.00593*.

[23] Watanabe, C., Hiramatsu, K., and Kashino, K. (2018). Modular representation of layered neural networks. *Neural Networks*, 97:62–73.

[24] Zhang, Z., Lin, Y., Liu, Z., Li, P., Sun, M., and Zhou, J. (2022). Moefication: Transformer feed-forward layers are mixtures of experts. In Muresan, S., Nakov, P., and Villavicencio, A., editors, *Findings of the Association for Computational Linguistics: ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 877–890. Association for Computational Linguistics.

[25] Zou, A., Phan, L., Chen, S., Campbell, J., Guo, P., Ren, R., Pan, A., Yin, X., Mazeika, M., Dombrowski, A.-K., et al. (2023). Representation engineering: A top-down approach to ai transparency. *arXiv preprint arXiv:2310.01405*.

Appendix

A Hyperparameters

We list the hyperparameters and various design choices in our experiments in Tab. 1, and refer to our codebase for more details (to be added during de-anonymization).

Table 1: Hyperparameter choices and other experiment details.

Hyperparameter/Design Choice	**Value**
Dataset	CIFAR-10 (or MNIST)
Batch Size	64
Optimizer	Adam
Learning Rate (α)	1×10^{-3}
Criterion	Cross-Entropy
Clusterability Factor	20
Number of Clusters	4
MLP (for MNIST)	
Input Size	28×28
Hidden Layer Sizes	64, 64
Activation Function	ReLU
Output Size	10
Bias	False
CNN (for CIFAR-10)	
Input Channels	3
Conv Layer 1: Out Channels	16
Conv Layer 1: Kernel Size	3
Conv Layer 1: Stride	1
Conv Layer 1: Padding	1
FC Layer 1: Input Features	$16 \times 16 \times 16$
FC Layer 1: Output Features	64
FC Layer 2: Output Features	64
Output Size	10
Bias	False
Activation Function	ReLU
Pruning Method	Iterative weight pruning
Pruning Criteria	Performance-based (loss and accuracy)
Effective Circuit Size Calculation	Fraction of non-zero weights

B Results on MNIST

Here we present our results on the MNIST [4] dataset.

C Clustered Layer Visualization

Fig. 7 shows the visualization of the fully connected layer trained on CIFAR10 of the network with bipartite clusters.

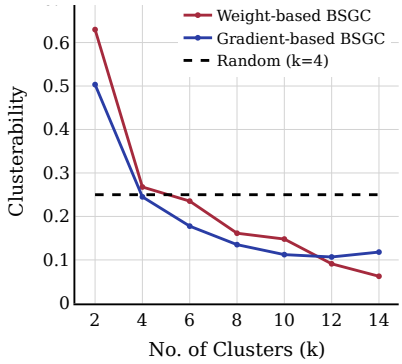

Figure 4: Clusterability (enmeshment) of the model with k bipartite clusters using Algorithm 1 on MNIST.

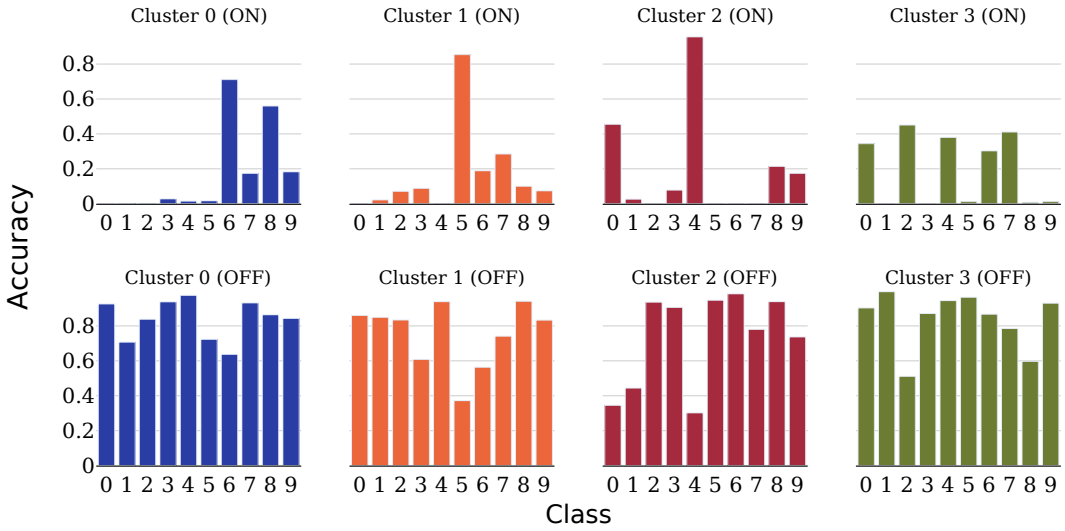

Figure 5: Class-wise accuracy for each label with clusters turned ON (top) and OFF (bottom) for MNIST. Note individual clusters learning near-complete circuits for various labels.

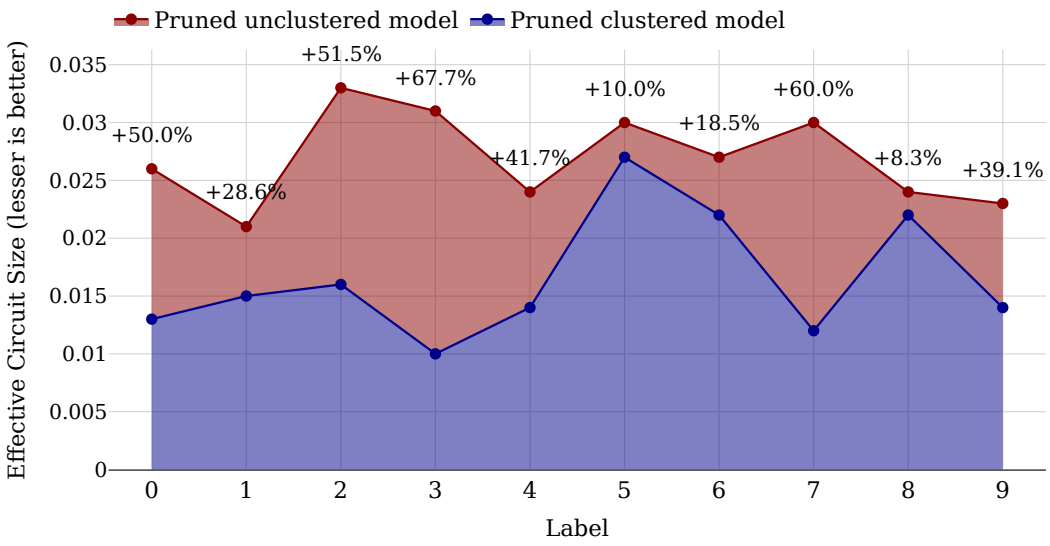

Figure 6: Effective Circuit Size (ECS) of circuits for each label as a fraction of the whole model for both clustered and un-clustered models trained on MNIST. Larger arrows denote a larger reduction in ECS.

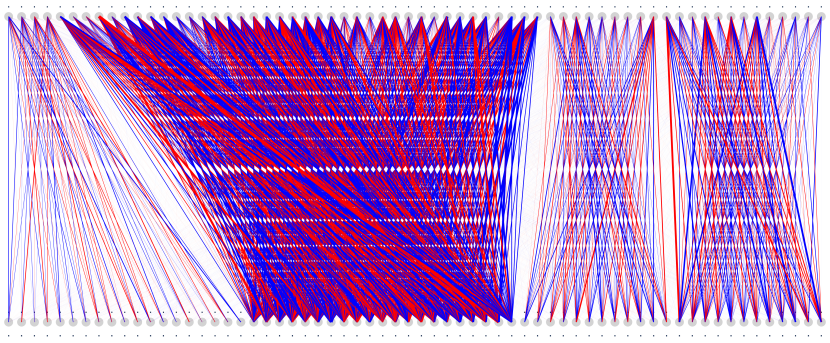

Figure 7: Visualizing the clustered layer $fc2$ of the model. Red and blue denote negative and positive weights respectively.