ROBUSTNESS OF EDITED NEURAL NETWORKS

Davis Brown,[1,*] **Charles Godfrey,**[1,*] **Cody Nizinski,**[1] **Jonathan Tu,**[1] **Henry Kvinge**[1,2]
[1]Pacific Northwest National Laboratory, [2] Department of Mathematics, University of Washington
[*] equal contribution
{first}.{last}@pnnl.gov

ABSTRACT

Successful deployment in uncertain, real-world environments requires that deep learning models can be efficiently and reliably modified in order to adapt to unexpected issues. However, the current trend toward ever-larger models makes standard retraining procedures an ever-more expensive burden. For this reason, there is growing interest in *model editing*, which enables computationally inexpensive, interpretable, post-hoc model modifications. While many model editing techniques are promising, research on the properties of edited models is largely limited to evaluation of validation accuracy. The robustness of edited models is an important and yet mostly unexplored topic. In this paper, we employ recently developed techniques from the field of deep learning robustness to investigate both how model editing affects the general robustness of a model, as well as the robustness of the specific behavior targeted by the edit. We find that edits tend to reduce general robustness, but that the degree of degradation depends on the editing algorithm chosen. In particular, robustness is best preserved by more constrained updates that modify less of the model. Motivated by these observations we introduce a new model editing algorithm, *1-layer interpolation (1-LI)*, which uses weight-space interpolation to navigate the trade-off between editing task accuracy and general robustness.

1 INTRODUCTION

Many applications require a model to be repeatedly modified after initial training to address model deficiencies and other undesirable behavior. One such family of such methods is called *model editing*. These methods aim to modify a model without the need for extensive computations or a large collection of new training examples. Most of these methods aim to change the way a model behaves in very specific instances, for example modifying an image model so that it correctly classifies cars even on streets covered with snow (Santurkar et al., 2021) or updating a language model to correctly identify the new prime minister of the United Kingdom (Mitchell et al., 2022a;b). Moreover, editing methods often apply a tightly constrained update to network weights, for example adding a low-rank matrix to the weights of a single layer. Section 2 (and Appendix B) contain summaries of the editing tasks (datasets and objectives) and algorithms that we consider.

The robustness of deep learning models has become an active area of research, motivated by observations that models may be more brittle than previously recognized (Recht et al., 2019; Hendrycks & Dietterich, 2019; Barbu et al., 2019; Hendrycks et al., 2021b;a). Given model editing's potential use as a tool to easily update deployed models, it is important to understand how editing affects robustness. For example, we know that strong performance on a test set does not always imply strong performance upon deployment where out-of-distribution (OOD) data is likely to be encountered. Should we expect edited models to generalize? To what extent? Given that many editing methods are quite different from traditional network optimization and fine-tuning, the answers to these questions are not obvious and have not yet been addressed in the literature. The purpose of this paper is to stress test model editing by evaluating how robust edited models really are. In summary, our **contributions:**

- We evaluate model editing techniques to understand their effect on robustness, finding that while editing tends to reduce robustness, the magnitude of this reduction depends on the type of editing used and layer chosen.

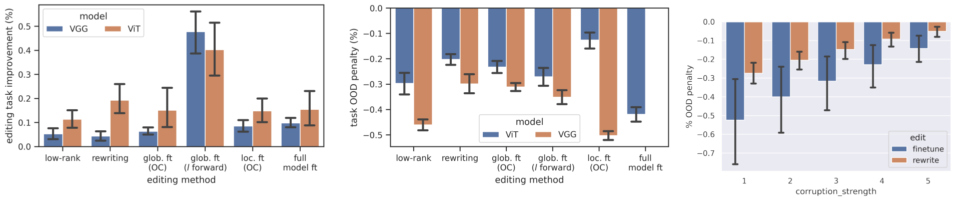


Figure 1: **OOD performance of edited models**. **Left**: Editing validation accuracies of VGG16 and ViT models on the synthetic concept-style task. **Middle:** Editing task OOD penalties Equation (7) of the edited models in on the left. **Right**: original task OOD penalties on ImageNet-C Equation (6) of VGG models edited using the synthetic concept-style task.[1]

- We introduce new approaches to editing, most notably *1-layer interpolation* which results in less robustness degradation than previous weight-space interpolation methods, in certain layers exhibiting *effective robustness* (see Figure 3).

2 Model Editing

Our model editing problem and data set-up are described in full detail in Appendix B. In summary, we consider two different kinds of editing set-ups, largely borrowing from (Santurkar et al., 2021) and (Ilharco et al., 2022). The first set-up uses a training dataset $\mathcal{D}_{\text{edit}}^{\text{train}}$ comprised of triplets (x, x', y). The model is either "edited" such that $f(x) \approx f(x')$ (we call this *output collision*, and it includes the low-rank, rewriting, and fine-tuning for output collision methods) or can be edited such that $f(x) \approx y$ (called fine-tuning). Note that in fine-tuning, the editing task introduces new labels y for the model to target, whereas for output collision the editing tasks only introduces new inputs. All editing methods studied in this paper are defined in full in Appendix B, and summarized in Table 1.

The second editing set-up examine the related problem of adapting open-vocabulary models such as CLIP (Radford et al., 2021) to the MNIST (Lecun et al., 1998), SVHN (Netzer et al., 2011) and KITTI distance estimation (Geiger et al., 2013), via interpolations (Ilharco et al., 2022). We measure the the robustness of the resulting edited models (see Section 3.1). The interest in MNIST, SVHN and KITTI stems from the fact that for all its impressive zero-shot performance on a wide variety of datasets, the original CLIP model struggled on these three (famously underperforming logistic regression on raw pixels in the case of MNIST). Next, we introduce two new editing methods motivated by our experiments on editing robustness.

Single-layer interpolation (1-LI): We apply local fine-tuning at layer l along with weight-space interpolation. The inspiration for this approach was to combine a common feature of several editing algorithms, namely weight updates restricted to a single layer, with the empirical benefits of weight-space interpolation (Ilharco et al., 2022; Wortsman et al., 2022). In Section 3.1 we show that this method often reduces degradation of model robustness.

Direct low-rank editing: The motivation for our direct low-rank editing algorithm comes from asking how much of the effectiveness of other model editing methods stem *simply from employing low-rank weight updates*, and our experimental results in Section 3.1 suggest that at least when comparing with rewriting, the answer is that much of the effectiveness is due to rank restriction alone. Using the framework and notation in Appendix B, we update a weight W_l with $W_l + UV^T$, where U, V are low-rank matrices. We provide additional details, contrast the method with those of (Bau et al., 2020; Santurkar et al., 2021) and provide a code sample in Appendix C.

3 OOD performance of edited models

We consider two different performance measures for an edited model on OOD data: the accuracy of an edited model on a distribution-shifted version of the original task validation set $\mathcal{D}_{\text{orig}}^{\text{val}}$, as well as

[1] Unfortunately results for editing (resp. original task) OOD penalties of full fine-tuning for VGG (resp. ViT) models did not return by the time of publication, so they are missing in the middle (resp. right) plots.

the accuracy of the edited model on a distribution-shifted version of the *editing* task validation set $\mathcal{D}^{\text{val}}_{\text{edit}}$. These two evaluations are quite different: in the example of the vehicles-on-snow task with a distribution shift generated by blurring images, the first performance measure requires evaluating the edited model on blurry images from all ImageNet classes, while the second requires evaluating blurry images of vehicles on snow. We model corrupted ImageNet images using the ImageNet-C dataset released by (Hendrycks & Dietterich, 2019), and use the ImageNet-C code to corrupt non-ImageNet data (see Appendix D.2). In addition, we evaluate models on ImageNet-R (Hendrycks et al., 2021a) and ImageNet-A (Hendrycks et al., 2021b). Further dataset details can be found in Appendix D.1.

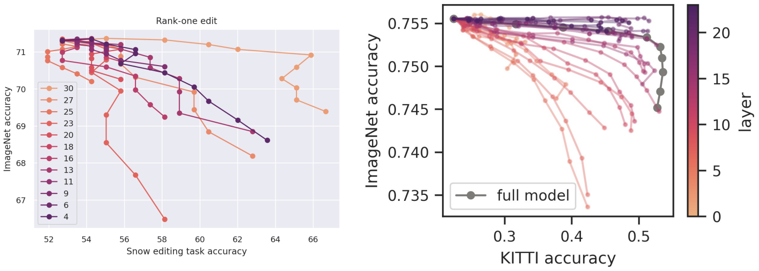


Figure 2: Linear interpolation between original and edited weights W and W' for edited models. Each curve corresponds to a an editing layer, and each point on that curve corresponds to evaluation of a model with weights $(1 - \alpha)W + \alpha W'$ for some $t \in [0, 1]$. **Left**: direct rank one editing for VGG16 models on the vehicles-on-snow dataset. **Right**: Full and single layer linear interpolation between original and edited weights of CLIP models for the KITTI task.

3.1 Results

Due to constraints of space we relegate a complete discussion of models, tasks and editing algorithms to Appendix B. Our first basic finding is that all editing methods result in *negative* OOD penalties, meaning the edited models tend to be less robust. Figure 1 (a) displays editing validation accuracies of VGG and ViT models on the synthetic concept-style task, taking the maximum validation accuracy over edited layers and averaged over all concept-style pairs considered. Figure 1 (b) displays editing task OOD penalties for the same models and editing tasks as Figure 1 (a), where distribution shift is modelled by applying corruptions to the synthetic concept-style validation sets. We find direct low-rank editing underperforms in terms of editing task OOD penalty (for both models) and that full model fine-tuning seems to break the relative trend for the two models. Figure 1 (c) displays original task OOD penalties on corrupted ImageNet (i.e., corruptions of $\mathcal{D}^{\text{val}}_{\text{orig}}$) for VGG and ViT models edited using the synthetic concept-style task.

Figure 2 displays both ImageNet accuracy and editing task accuracy along line segments $(1-\alpha)W + \alpha W'$ in weight space, where W are the original model weights and W' are the edited weights. For all editing methods ImageNet accuracy generally decreases as editing accuracy increases, and moreover the resulting curves appear to be roughly concave. This was found to be the case for global fine-tuning in (Ilharco et al., 2022), but to our knowledge has not been established for local fine-tuning or low-rank editing. Note that performance is layer-dependent, indicating that the choice of editing layer is task dependent. These results are a new observation of a monotone linear interpolation (MLI) phenomenon. The first observation of MLI found empirically that training loss decreased monotonically along line segments between randomly initialized weights and weights at the endpoint of SGD training (Goodfellow et al., 2014). The basis for weight-space interpolation as a fine-tuning mechanism (as in (Ilharco et al., 2022; Wortsman et al., 2022)) is another form of MLI: empirically, as the interpolation parameter α increases, accuracy on the fine-tuning task increases monotonically while accuracy on the original task decreases monotonically. Figure 2 shows the same behaviour occurs when weights are updated using single layer fine-tuning or low-rank editing methods. We believe this is new, with the following exception: one can prove that if only the final (pre-logit) weights are updated and the pairwise loss function is convex (for example, cross entropy loss) then MLI holds.

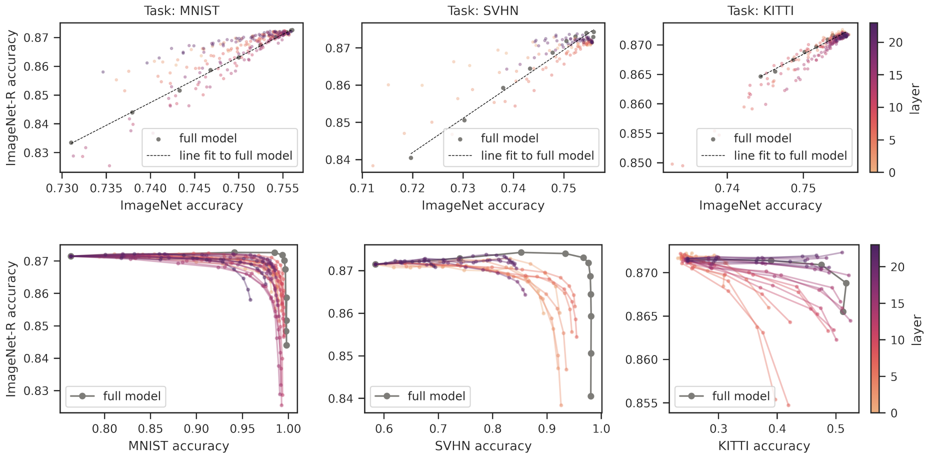


Figure 3: ImageNet-R vs ImageNet (top) and task accuracy (bottom) along line segments $(1-\alpha)W + \alpha W'$ between original and edited weights W and W' of CLIP models edited on various tasks, where different points correspond to different interpolation parameters α.

Takeaway: *Weight interpolation continues to be an effective way of balancing original task accuracy and editing task accuracy, even when editing updates weights at a single layer (as opposed to global fine-tuning). Surprisingly, 1-LI methods outperform full fine-tuning and weight space interpolation on some tasks and robustness evaluations.*

In addition to measuring accuracies on original and editing tasks, we can measure *OOD accuracy* along line segments $(1-\alpha)W + \alpha W'$. Figure 3 (bottom) displays ImageNet-R accuracies of the CLIP models appearing in Figure 5, plotted with respect to editing task accuracy. Additional results for ImageNet-C and ImageNet-A are provided in Appendix A.2. These curves are generally monotonically *decreasing*, showing that the edited models are less robust to corrupted variants of the original validation set than the original model. For all three tasks there are layers where single-layer fine-tuning provides higher ImageNet-R (and ImageNet-C/A in Figures 6 and 7) accuracy than global fine-tuning along some portion of the interpolation path. Considering Figures 2, 3 and 5 to 7 as a whole, we see that while accuracy on OOD variants of ImageNet accuracy does appear to be linearly correlated with ImageNet accuracy, as is often observed to be the case (Recht et al., 2019; Taori et al., 2020; Miller et al., 2021; Wortsman et al., 2022), there are notable exceptions: a simple linear correlation would predict global fine-tuning to exhibit higher ImageNet-C accuacy for all values of α in Figure 6 (a, b), which is not the case. This deviation from linear correlation suggests that single-layer edited models may exhibit a form of *effective robustness* as formalized in (Taori et al., 2020). In Figure 3 (top), which directly plots ImageNet-R vs ImageNet accuracy, we see that this does appear to occur in some cases. Another question is whether the finding that effective robustness decreases over the course of global or last-layer fine-tuning (Andreassen et al., 2021) holds in the case of local fine-tuning of early layers. Investigation of this last point would be an exciting direction for future work.

Takeaway: *Edits tend to degrade model robustness on the original task, but the magnitude of this decrease depends strongly on the method used. In particular, methods constrained to certain layers can retain significantly more robustness on the original task.*

4 Conclusion

In this work we evaluate the changes to model robustness that result from editing techniques. We find that in general, all editing techniques tend to degrade model robustness but that edits that are constrained to update only a fraction of model parameters tend to result in a less pronounced drop in robustness. We hope that these results can be used to better inform the choice of editing method in real-world situations where a model is expected to encounter out-of-distribution data upon deployment.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

James A. Anderson. A simple neural network generating an interactive memory. *Mathematical Biosciences*, 14(3):197–220, August 1972. ISSN 0025-5564. doi: 10.1016/0025-5564(72)90075-2.

Anders Andreassen, Yasaman Bahri, Behnam Neyshabur, and Rebecca Roelofs. The Evolution of Out-of-Distribution Robustness Throughout Fine-Tuning, June 2021.

Andrei Barbu, David Mayo, Julian Alverio, William Luo, Christopher Wang, Dan Gutfreund, Josh Tenenbaum, and Boris Katz. ObjectNet: A large-scale bias-controlled dataset for pushing the limits of object recognition models. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

David Bau, Steven Liu, Tongzhou Wang, Jun-Yan Zhu, and Antonio Torralba. Rewriting a Deep Generative Model, July 2020.

Holger Caesar, Jasper Uijlings, and Vittorio Ferrari. COCO-Stuff: Thing and Stuff Classes in Context, March 2018.

Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs, May 2017.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. Knowledge Neurons in Pretrained Transformers, March 2022.

Nicola De Cao, Wilker Aziz, and Ivan Titov. Editing Factual Knowledge in Language Models, September 2021.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A large-scale hierarchical image database. In *CVPR09*, 2009.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale, June 2021.

Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Brandon Tran, and Aleksander Madry. Adversarial Robustness as a Prior for Learned Representations. *arXiv:1906.00945 [cs, stat]*, September 2019.

Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. *arXiv preprint arXiv:1811.12231*, 2018.

Golnaz Ghiasi, Honglak Lee, Manjunath Kudlur, Vincent Dumoulin, and Jonathon Shlens. Exploring the structure of a real-time, arbitrary neural artistic stylization network, August 2017.

Ian J Goodfellow, Oriol Vinyals, and Andrew M Saxe. Qualitatively characterizing neural network optimization problems. *arXiv preprint arXiv:1412.6544*, 2014.

Peter Hase, Mona Diab, Asli Celikyilmaz, Xian Li, Zornitsa Kozareva, Veselin Stoyanov, Mohit Bansal, and Srinivasan Iyer. Do language models have beliefs? methods for detecting, updating, and visualizing model beliefs. *arXiv preprint arXiv:2111.13654*, 2021.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition, December 2015.

Dan Hendrycks and Thomas Dietterich. Benchmarking Neural Network Robustness to Common Corruptions and Perturbations, March 2019.

Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, Dawn Song, Jacob Steinhardt, and Justin Gilmer. The many faces of robustness: A critical analysis of out-of-distribution generalization. *ICCV*, 2021a.

Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural Adversarial Examples, March 2021b.

Gabriel Ilharco, Mitchell Wortsman, Samir Yitzhak Gadre, Shuran Song, Hannaneh Hajishirzi, Simon Kornblith, Ali Farhadi, and Ludwig Schmidt. Patching open-vocabulary models by interpolating weights, August 2022.

Bobak Kiani, Randall Balestriero, Yann LeCun, and Seth Lloyd. projUNN: Efficient method for training deep networks with unitary matrices, October 2022.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization, January 2017.

Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, Tony Lee, Etienne David, Ian Stavness, Wei Guo, Berton A. Earnshaw, Imran S. Haque, Sara Beery, Jure Leskovec, Anshul Kundaje, Emma Pierson, Sergey Levine, Chelsea Finn, and Percy Liang. WILDS: A benchmark of in-the-wild distribution shifts. In *International Conference on Machine Learning (ICML)*, 2021.

Teuvo Kohonen. Correlation Matrix Memories. *IEEE Transactions on Computers*, C-21(4):353–359, April 1972. ISSN 1557-9956. doi: 10.1109/TC.1972.5008975.

Teuvo Kohonen. *Associative Memory*, volume 17 of *Communication and Cybernetics*. Springer, Berlin, Heidelberg, 1977. ISBN 978-3-642-96386-5 978-3-642-96384-1. doi: 10.1007/978-3-642-96384-1.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

Yoonho Lee, Annie S Chen, Fahim Tajwar, Ananya Kumar, Huaxiu Yao, Percy Liang, and Chelsea Finn. Surgical fine-tuning improves adaptation to distribution shifts. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=APuPRxjHvZ`.

James Lucas, Juhan Bae, Michael R. Zhang, Stanislav Fort, R. Zemel, and R. Grosse. Analyzing Monotonic Linear Interpolation in Neural Network Loss Landscapes. *ArXiv*, April 2021.

Sébastien Marcel and Yann Rodriguez. Torchvision the machine-vision package of torch. In *Proceedings of the 18th ACM International Conference on Multimedia*, MM '10, pp. 1485–1488, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781605589336. doi: 10.1145/1873951.1874254. URL `https://doi.org/10.1145/1873951.1874254`.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and Editing Factual Associations in GPT, June 2022a.

Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. Mass-Editing Memory in a Transformer, October 2022b.

John Miller, Rohan Taori, Aditi Raghunathan, Shiori Sagawa, Pang Wei Koh, Vaishaal Shankar, Percy Liang, Yair Carmon, and Ludwig Schmidt. Accuracy on the Line: On the Strong Correlation Between Out-of-Distribution and In-Distribution Generalization, October 2021.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D. Manning. Fast Model Editing at Scale, June 2022a.

Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D. Manning, and Chelsea Finn. Memory-Based Model Editing at Scale, June 2022b.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011. URL `http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf`.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. Curran Associates Inc., Red Hook, NY, USA, 2019.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, 2021.

Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do ImageNet Classifiers Generalize to ImageNet?, June 2019.

Shiori Sagawa, Pang Wei Koh, Tony Lee, Irena Gao, Sang Michael Xie, Kendrick Shen, Ananya Kumar, Weihua Hu, Michihiro Yasunaga, Henrik Marklund, Sara Beery, Etienne David, Ian Stavness, Wei Guo, Jure Leskovec, Kate Saenko, Tatsunori Hashimoto, Sergey Levine, Chelsea Finn, and Percy Liang. Extending the wilds benchmark for unsupervised adaptation. In *International Conference on Learning Representations (ICLR*, 2022.

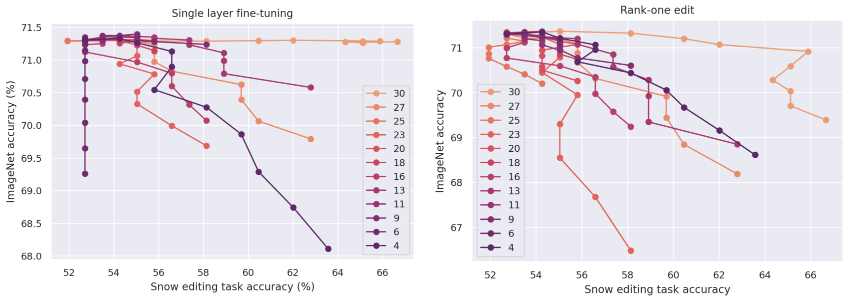
Shibani Santurkar, Dimitris Tsipras, Mahalaxmi Elango, David Bau, Antonio Torralba, and Aleksander Madry. Editing a classifier by rewriting its prediction rules. In *Advances in Neural Information Processing Systems*, volume 34, pp. 23359–23373. Curran Associates, Inc., 2021.

Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition, April 2015.

Rohan Taori, Achal Dave, Vaishaal Shankar, Nicholas Carlini, Benjamin Recht, and Ludwig Schmidt. Measuring Robustness to Natural Distribution Shifts in Image Classification. In *Advances in Neural Information Processing Systems*, volume 33, pp. 18583–18599. Curran Associates, Inc., 2020.

Ross Wightman. Pytorch image models. `https://github.com/rwightman/pytorch-image-models`, 2019.

Mitchell Wortsman, Gabriel Ilharco, Jong Wook Kim, Mike Li, Simon Kornblith, Rebecca Roelofs, Raphael Gontijo-Lopes, Hannaneh Hajishirzi, Ali Farhadi, Hongseok Namkoong, and Ludwig Schmidt. Robust fine-tuning of zero-shot models, June 2022.



(a)

(b)

Figure 4: Linear interpolation between original and edited weights W and W' for edited VGG16 models the vehicles-on-snow dataset. Each curve corresponds to a an editing layer, and each point on that curve corresponds to evaluation of a model with weights $(1-\alpha)W + \alpha W'$ for some $t \in [0,1]$. **(a)**: local fine-tuning for output collision. **(b)**: direct rank one editing.

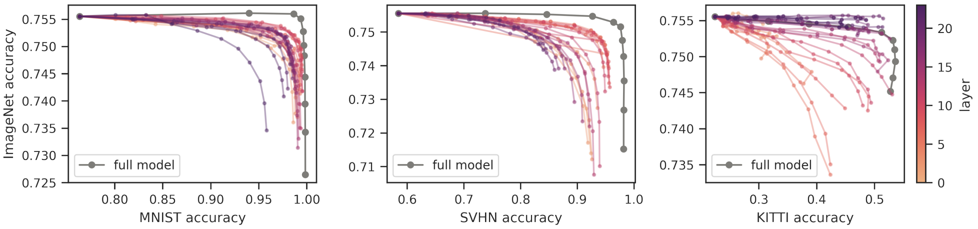


Figure 5: Linear interpolation between original and edited weights W and W' of CLIP models edited on various tasks. Each curve corresponds to a an editing layer (or to global fine-tuning), and each point on that curve corresponds to evaluation of a model with weights $(1-\alpha)W + \alpha W'$ for some $\alpha \in [0,1]$.

A Additional editing results and analysis

A.1 Editing task performance

Figure 4 displays both ImageNet accuracy and vehicles-on-snow task accuracy along line segments $(1-\alpha)W + \alpha W'$ in weight space, where W are the original model weights and W' are the edited weights. In this experiment we use VGG16 models, and two different editing methods: local fine-tuning for output collision (Item i)) and direct low-rank editing (with rank $= 1$). For both editing methods we see that ImageNet accuracy generally decreases as editing accuracy increases, and moreover that the resulting curves appear to be roughly concave. This shows that weight space interpolation may serve as a viable alternative to early stopping for the purpose of balancing accuracy on the original and editing task. As noted in Section 3.1, this was found to be the case for global fine-tuning in (Ilharco et al., 2022), but to our knowledge has not been established for local fine-tuning or low-rank editing.

Figure 5 shows full results for a similar phenomenon occurs for a much larger CLIP open vocabulary model (Radford et al., 2021) on a variety of editing tasks: MNIST, KITTI and SVHN. When fine-tuning a single layer, ImageNet accuracy generally decreases as editing accuracy increases, and the resulting curves appear to be roughly concave. On the MNIST and SVHN tasks, the curve for global fine-tuning dominates those of single layer fine-tuning, but interestingly we see that fine-tuning a single (later) layer can result in superior KITTI accuracy at a given ImageNet accuracy. Note also that for SVHN, tuning later layers results in worse performance than tuning earlier layers, indicating that the choice of editing layer is task dependent.

It is worth mentioning (Lucas et al., 2021) exhibits realistic examples of neural network training where MLI fails, even though models train successfully — a common theme of these examples is that they involved long-distance travel of model weights (for example, using the Adam optimizer (Kingma & Ba, 2017) reliably broke MLI). We suspect that a reason failures of MLI are not more commonly observed in the context of fine-tuning and editing is the common practice of limiting the distance travelled by weights in these situations (see for example the discussion in §4.2, 8 of (Ilharco et al., 2022)).

A.2 OOD PERFORMANCE

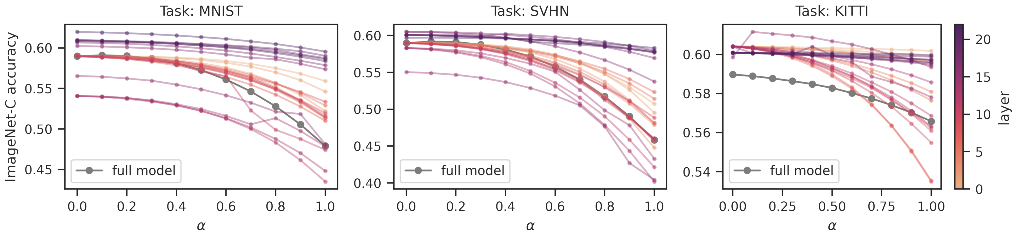


Figure 6: ImageNet-C accuracy along line segments $(1-\alpha)W + \alpha W'$ between original and edited weights W and W' of CLIP models edited on various tasks, plotted with respect to the interpolation parameter α.

In Figures 1, 3, 6 and 7 there appears to be at least one common thread: editing weight updates with restricted capacity (as measured by the effective number of model parameters the update modifies) can preserve more of the robustness of the original model. Indeed, in Figure 1 we see that rewriting, a method that perturbs a single layer's weight matrix with a low-rank matrix, incurrs a smaller (i.e. less negative) original task OOD penalty than full model fine-tuning, which is free to modify all network parameters. In Figure 3 (bottom) and Figure 6 we see that 1-LI, a method that by definition only modifies weights at a single network layer, retains higher accuracy on ImageNet-C and ImageNet-R than full model fine-tuning for many editing layers. Of course, in the case of 1-LI there are also layers that fare worse on ImageNet-C and ImageNet-R than full model fine-tuning, illustrating that editing weight update capacity cannot be the only factor in play.

B DESCRIPTION OF MODEL EDITING

The model editing problem can be thought of in terms of two distinct training tasks, executed in sequence. The first is the *original task*, in which a neural network f is trained on a dataset $\mathcal{D}^{\text{train}}_{\text{orig}}$ comprising input-output pairs (x, y); the goal is to achieve a good approximation $f(x) \approx y$. This is simply the standard neural network training procedure, usually achieved by minimizing a cost function using a gradient-based optimizer, like stochastic gradient descent (SGD).

Second is the *editing task*, defined by a second dataset $\mathcal{D}^{\text{train}}_{\text{edit}}$. In this work we focus on two types of editing tasks, *output collision* and *fine-tuning*. In the case of output collision, the goal is for the model to generate similar outputs for distinct but semantically similar inputs. More specifically, we let $\mathcal{D}^{\text{train}}_{\text{edit}}$ comprise input pairs (x, x') and train the model such that $f(x) \approx f(x')$. We focus on output collision editing tasks because they are interpretable, effective, and extend naturally to a wide variety of models.[2] For fine-tuning tasks $\mathcal{D}^{\text{train}}_{\text{edit}}$ is a second dataset of input-output pairs (x, y) and the model is trained to achieve a good fit $f(x) \approx y$ on this new data.[3] Note that in fine-tuning, the editing task introduces new labels y for the model to target, whereas for output collision the editing tasks only introduces new inputs.

For both types of editing tasks, there is typically a validation dataset $\mathcal{D}^{\text{val}}_{\text{edit}}$ consisting of input-output pairs (x, y) (note that this is true even in the output collision setting).

[2] Output collision tasks encompass those used by the rewriting algorithm, which extends to GANs, image classifiers and language models as discussed in Appendix E.

[3] We emphasize that whereas in some other applications of fine-tuning model architecture is modified (for example replacing a final layer to account for different labels on the new dataset) this does not occur in our editing experiments.

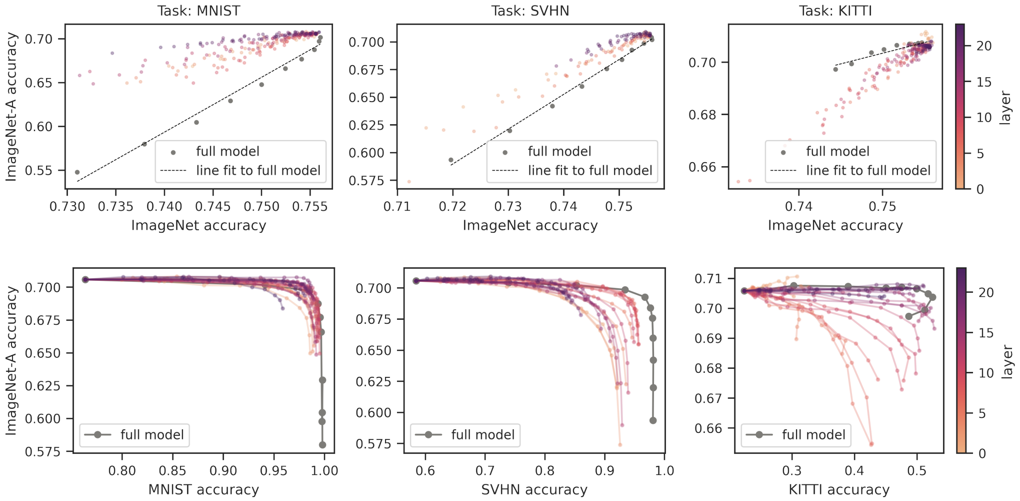


Figure 7: ImageNet-A vs ImageNet (top) and task accuracy (bottom) along line segments $(1-\alpha)W+\alpha W'$ between original and edited weights W and W' of CLIP models edited on various tasks, where different points correspond to different interpolation parameters α.

B.1 Edit tasks and data

In our first experiments we study image classification models trained on ImageNet (Deng et al., 2009) object recognition as the original task. We consider two different editing tasks, both closely related to the experiments performed in (Santurkar et al., 2021).

The first task is motivated by the observation that a particular image classifier could accurately identify vehicles on roadways, but was noticeably less accurate when classifying vehicles on snow. To address this shortcoming, (Santurkar et al., 2021) took datapoints (x, y) from ImageNet and used style transfer to replace road segments in the image x with snow texture. Each modified image x' was paired with the original version; the result is a set of triples (x, x', y) which can be used for an output collision task (by dropping y) or a fine-tuning task (by dropping the original image x). Validation is performed using a curated collection of images of vehicles on show downloaded from Flickr, the original source of ImageNet images. We refer to this editing task as “vehicles on snow.”

Our second editing task synthesizes data using a pre-trained segmentation model and a style transfer model. Starting with a given ImageNet datapoint (x, y), we generate a number of modified versions $x'_1, x'_2, \ldots, x'_k$ to pair with it as follows: First, we select a concept like “mountain.” Then we use the segmentation model to identify regions in x matching that concept. Next we select a style, in practice an image of a texture like “fall colors,” and use style-transfer to replace those regions in x labelled as “mountain” with “fall colors.” Choosing k different images of “fall colors” to run through style transfer results in the k different modified versions $x'_1, x'_2, \ldots, x'_k$. The pairing of a mountains-to-fall colors modification with the original datapoint yields a triple (x, x'_i, y) for each $i = 1, \ldots, k$, analogously to the vehicles-on-snow task. As in that case, we can obtain pairs for output collision (resp. fine-tuning) tasks by dropping y (resp. x). Note that as in (Santurkar et al., 2021), we select images for modification from the ImageNet validation set, then split the resulting image pairs into training set $\mathcal{D}^{\text{train}}_{\text{edit}}$ and validation set $\mathcal{D}^{\text{val}}_{\text{edit}}$ for the editing task. Further details as well as example datapoints can be found in Appendix D.2. We refer to this as the “synthetic concept-style” editing task.

We also examine the problem of adapting open-vocabulary models such as CLIP (Radford et al., 2021) to the MNIST (Lecun et al., 1998), SVHN (Netzer et al., 2011) and KITTI (distance estimation) (Geiger et al., 2013). While traditionally viewed as a sort of fine-tuning or continual learning problem, we show that it can be approached using model editing methods such as single-layer fine-tuning, and that measuring the the robustness of the resulting edited models yields interesting results (see Section 3.1). The interest in MNIST, SVHN and KITTI stems from the fact that for all its impressive zero-shot performance on a wide variety of datasets, the original CLIP model struggled on these three (famously underperforming logistic regression on raw pixels in the case of MNIST).

B.2 Methods

We start by introducing some notation. For ease of exposition, consider a simplified neural network whose architecture can be described below (for discussion on the adaptations to more complicated architectures, e.g., with residual connections, see Appendix C). Let f be a neural network defined as the composition

$$\mathbb{R}^{n_0} \xrightarrow{f_1} \mathbb{R}^{n_1} \xrightarrow{f_2} \cdots \xrightarrow{f_{L-1}} \mathbb{R}^{n_{L-1}} \xrightarrow{f_L} \mathbb{R}^{n_L}, \tag{1}$$

where $f_l(x) = \sigma(W_l x)$, with W_l an $n_l \times n_{l-1}$ matrix for each $l = 1, \ldots, L$ and σ a coordinate-wise non-linearity (e.g., ReLU). For any such l, let $f_{\leq l}$ denote the composition of the first l layers: $f_l \circ f_{l-1} \circ \cdots \circ f_2 \circ f_1$. Similarly, let $f_{>l}$ denote the composition of the last $L - l$ layers $f_L \circ f_{L-1} \circ \cdots \circ f_{l+1}$.

In what follows we discuss two tools in model editing: methods for updating model weights and methods for balancing the trade off in accuracy between the original and editing task. Most of the references in Appendix E apply some weight update procedure together with cross validation, however (Ilharco et al., 2022) recently obtained impressive results using global fine-tuning and weight space linear interpolation. Further details, including notes on implementation, are deferred to Appendix C.

Methods of updating weights: Consider an input pair (x, x'), as in an output collision editing task. If $f_{\leq l}(x) = f_{\leq l}(x')$ for any l, it trivially follows that $f(x) = f(x')$, as desired. This is the basic motivation for the following weight update procedures, all of which attempt to minimize the mean squared error between $f_{\leq l}(x)$ and $f_{\leq l}(x')$ creating the situation where $f_{\leq l}(x) \approx f_{\leq l}(x')$:

i) *Local fine-tuning for output collision*: Optimize W_l using SGD.
ii) *Global fine-tuning for output collision*: Optimize $W_1, \ldots, W_l$ using SGD.
iii) *Rewriting* (Bau et al., 2020; Santurkar et al., 2021; Meng et al., 2022a;b): Perform update $W_l \leftarrow W_l + UV^T$, where U, V are low-rank matrices. V is derived by viewing W_l as a linear associative memory (Kohonen, 1972; Anderson, 1972; Kohonen, 1977) and U is optimized using SGD.

For fine-tuning tasks where updates use pairs (x', y) rather than (x, x'), additional methods are available:

iv) *Local fine-tuning at layer l*: Optimize W_l with SGD such that $f(x) \approx y$ for new datapoints (x, y).
v) *Global fine-tuning from layer l forward*: Optimize $W_l, \ldots, W_L$ with SGD such that $f(x) \approx y$ for new datapoints (x, y)

Approaches to balancing original and editing accuracy: The two dominant approaches that we will consider are:

i) *Cross validation*: In this simple approach, one monitors the validation accuracy on the original and editing validation datasets while performing optimization with respect to the editing task. One chooses a desirable "best iteration" (i.e. early stopping) as suits the application.
ii) *Weight space linear interpolation*: Surprisingly, linearly interpolating weights often interpolates between performance on the original and editing tasks. Specifically, given original weights $W := (W_l)_{l=1}^L$ and edited weights $W' := (W'_l)_{l=1}^L$ one chooses $\alpha \in [0, 1]$ such that $(1 - \alpha)W + \alpha W'$ balances performance between the two tasks.

We note that weight space interpolations have only been explored for full-model fine-tuning. To our knowledge we are the first to observe that linear interpolation works when weights have been modified using an editing method that only updates weights at a single layer (see Section 3.1).

C Detailed descriptions and references

method	figures	reference
1-layer interpolation	3, 4, 5	**ours**
direct low-rank editing	2(b)	**ours**
local fine-tuning for output collision	2(a)	**ours**, but closely related to (Bau et al., 2020; Santurkar et al., 2021)
global fine-tuning for output collision	1(a-b)	**ours**, but see also (Engstrom et al., 2019)
local fine-tuning at layer l		general knowledge (in the context of robustness see (Lee et al., 2023))
full model fine-tuning	1(a-b), 3^*, 4^*, 5^*	general knowledge (* combined with weight space interpolation (Ilharco et al., 2022))
rewriting	1(a-c)	(Bau et al., 2020; Santurkar et al., 2021)
global fine-tuning from layer l forward	1(a-b)	general knowledge

Table 1: Editing methods studied in this paper, relevant figures and references where applicable.

FOR WEIGHT UPDATE METHODS

In this section we only provide a complete description of direct low-rank editing. The rewriting algorithm is described in detail in (Bau et al., 2020) (see also (Santurkar et al., 2021), which adapts rewriting to image classifiers). We take fine-tuning to be well known (or easy to locate in existing literature), and emphasize 1-LI differs from the PAINT method of (Ilharco et al., 2022) only in the restriction of weight updates during fine-tuning to a single layer.

Direct low-rank editing is related to two successful editing methods, rewriting and the MEND method (Mitchell et al., 2022a), that employ low-rank perturbations to weight matrices. Beyond this common feature of low-rank perturbations, the two methods differ in many other details: MEND trains hypernetworks to optimize its low-rank perturbations UV^T and initializes UV^T by applying a singular value decomposition to model gradients on batches of editing samples[4] — this stands in contrast to the description of rewriting appearing in Item iii) above.[5]

In the case of direct low-rank editing at layer l of a network f as in Equation (1), we introduce new parameters U, V where U (resp. V) is an $n_l \times r$ (resp. $r \times n_{l-1}$) matrix; here r is the rank of the edit, a hyper parameter decided in advance. Then, with all weights $W_1, \dots, W_L$ frozen, we optimize the network weights

$$(W_1, \dots, W_{l-1}, W_l + UV^T, W_{l+1}, \dots, W_L) \tag{2}$$

to minimize the mean squared error $\frac{1}{n_l}|f_{\leq l}(x) - f_{\leq l}(x')|^2$ on the feature collision style editing training set using SGD on the parameters U, V.[6] A few notes:

- This method shares the low-rank feature of rewriting.
- It lacks a principled derivation through the lens of associative memories.
- It is potentially computationally less expensive: for example, rewriting requires precomputation of the covariance matrix Σ of the features $f_{\leq l-1}(x)$ over datapoints in the original training set $\mathcal{D}^{\text{train}}_{\text{orig}}$ and computing its inverse square root (to later apply a ZCA transformation). In the case of today's large neural networks, this feature covariance matrix

[4] Which are naturally low-rank, in fact with rank bounded by the batch size (see e.g. Appendix E of (Kiani et al., 2022))

[5] We do not compare with MEND in this work since it has somewhat different data requirements than those described in Appendix B.1 — in particular, an auxiliary dataset of editing examples is needed to train the hypernetwork to edit models. Evaluating direct low-rank editing on the tasks considered in (Mitchell et al., 2022a) would be an interesting experiment for future work.

[6] Note that since no earlier layers are trained, this is equivalent to minimizing the mean squared error between $\sigma((W_l + UV^T)f_{\leq l-1}(x))$ and $\sigma((W_l + UV^T)f_{\leq l-1}(x'))$, where the $l-1$st features $f_{\leq l-1}(x)$ and $f_{\leq l-1}(x')$ are fixed throughout editing optimization.

can be quite large (e.g. 4096×4096). Direct low-rank editing requires no preliminary numerical linear algebra. However, it is also true that a careful implementation of rewriting can avoid a large matrix inversion by the classic trick of replacing a “matrix-inverse-vector” operation $A^{-1}y$ with a call to a least squares solver for $|Ax - y|^2$, and we have not formally benchmarked the computational cost of direct low-rank and rewriting.

Most of the image classifiers we experiment with are CNNs, and in this case with the exception of the last few layers we are editing 2D *convolution* weights, rather than the matrices appearing in our toy model Equation (1). All editing methods other than direct low-rank and rewriting extend essentially verbatim to the case of CNNs. For direct low-rank and rewriting, we implement the low-rank perturbations as 1×1 convolutions (equivalent to multiplying channel vectors with the same low-rank matrix at every spatial position). More precisely, let W_l be the original, unedited convolution weight, say of shape $C_l \times C_{l-1} \times K \times K$ where K is the kernel size. We then let U and V be convolutions of shapes $C_l \times r \times 1 \times 1$ and $C_{l-1} \times r \times 1 \times 1$ respectively and replace the expression $W_l + UV^T$ used for fully connected linear layers with

$$W_l + U * V^T \tag{3}$$

where $*$ denotes convolution and transpose flips the first two indices. Note that as 1×1 convolutions U and V^T simply apply a fixed matrix over the channels of an input hidden feature at each spatial position. In (Santurkar et al., 2021) it was noted that this improves performance of rewriting, when compared to applying the perturbation $W_l + U * V^T$ only at spatial positions contained in the segmentation mask of the editing task (“road” in the case of vehicles on snow, or the concept being replaced via style transfer in the case of the large scale synthetic task).

A convolution using the weights of Equation (3) can be implemented in a few lines of idomatic PyTorch:

```
import torch
from torch.nn import functional as F


# u, v are matrices of shape (C_l, r), (C_{l-1}, r) respectively
# multiply to get a rank <= r matrix:
uv = torch.einsum('i,j->ij', u,  v)
# reshape to get a 1x1 convolution weight:
uv = uv.reshape(uv.shape + (1, 1))
# w is the original convolution weight of shape (C_l, C_{l-1}, K, K)
# x is an input of shape (C_{l-1}, H, W)
output = F.conv2d(x, w) + F.conv2d(x, uv)
```

Equation (3) only strictly-speaking makes sense when the kernel size K is *odd*: in that case, we can pad the 1×1 convolutions U and V with zeros in the spatial tensor dimensions to obtain $K \times K$ convolutions, say $\tilde{U}$ and $\tilde{V}$ such that if (k, l) denotes the index at the center of the $K \times K$ kernel (which depends on indexing conventions)

$$\tilde{U}_{cc'ij} = \begin{cases} U_{cc'ij} & \text{if } i = k, j = l \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

and similarly for $\tilde{V}$. Then we can sensibly evaluate $W_l + \tilde{U} * \tilde{V}^T$. In the case where K is even, if one wants an update that truly only modifies the weights of a network (rather than appending an auxiliary convolutional weight), there exist reasonable workarounds, for example if $\{k, k+1\} \times \{l, l+1\}$ is the center of the 2×2 grid at the center of the $K \times K$ kernel (which again depends on indexing conventions) one could use the weight sharing scheme

$$\tilde{U}_{cc'ij} = \begin{cases} U_{cc'ij} & \text{if } i - k \leq 1, j - l \leq 1 \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

Thankfully, since the VGG and ResNet CNNs in our experiments only use odd kernel sizes we never have to resort to such measures.

Lastly, following the discussion in §A.4 of (Santurkar et al., 2021) for ResNets we only edit convolutional layers appearing at the end of residual blocks.

D Experimental details

D.1 Experimental setup

One common form of distribution shift for natural image data is *corruption*. We model corrupted ImageNet images using the ImageNet-C dataset released by (Hendrycks & Dietterich, 2019), and use the ImageNet-C code to corrupt non-ImageNet data (see Appendix D.2 for further details). ImageNet-C contains many forms of corruption at varying severity levels; a table as well as example images can be found in (Hendrycks & Dietterich, 2019). In addition, we evaluate models on ImageNet-R (Hendrycks et al., 2021a) (which contains paintings and drawings of ImageNet classes) and ImageNet-A (Hendrycks et al., 2021b) (curated by downloading images related to ImageNet labels and retaining those incorrectly classified by a fixed ensemble of ResNet-50 models).

We use three ImageNet trained models: a VGG16 (Simonyan & Zisserman, 2015), a ResNet50 (He et al., 2015), and a Vision Transformer (ViT) (Dosovitskiy et al., 2021). For the VGG and ResNet we use pretrained weights available in `torchvision` (Marcel & Rodriguez, 2010), and for the ViT we use weights available in `timm` (Wightman, 2019).

In addition, we experiment with an open-vocabulary CLIP model, using the ViT-L/14 weights available in the code release for (Radford et al., 2021). This model was trained on a large dataset of 400 million image-text pairs scraped from the internet that is not publicly available.

Using the notation of Section 2 let $\mathcal{D}_{\text{orig}}^{\text{val}}$ be a validation set for the original task, (for example, the ImageNet validation set) and let $\tilde{\mathcal{D}}_{\text{orig}}^{\text{val}}$ be a distribution shifted variant of $\mathcal{D}_{\text{orig}}^{\text{val}}$ (for example obtained by blurring images from $\mathcal{D}_{\text{orig}}^{\text{val}}$). Letting f_{orig} and f_{edited} denote the model before and after editing, we compute the *original task OOD penalty* of the model edit

$$\text{Acc}(f_{\text{edited}}, \tilde{\mathcal{D}}_{\text{orig}}^{\text{val}}) - \text{Acc}(f_{\text{orig}}, \tilde{\mathcal{D}}_{\text{orig}}^{\text{val}}), \tag{6}$$

which is in words the raw change in accuracy on $\tilde{\mathcal{D}}_{\text{orig}}^{\text{val}}$ caused by model editing. On the other hand if $\tilde{\mathcal{D}}_{\text{edit}}^{\text{val}}$ is a distribution shifted variant of the editing validation set $\mathcal{D}_{\text{edit}}^{\text{val}}$, we compute the *editing task OOD penalty*

$$\text{Acc}(f_{\text{edited}}, \tilde{\mathcal{D}}_{\text{edit}}^{\text{val}}) - \text{Acc}(f_{\text{edit}}, \mathcal{D}_{\text{orig}}^{\text{val}}). \tag{7}$$

In words, this is the decrease in editing validation performance caused by the distribution shift in question.

D.2 Synthetic editing datasets and corruptions thereof

We start with the segmented ImageNet validation datapoints[7] and style images made available in the code release of (Santurkar et al., 2021) — these segmented ImageNet validation datapoints are organized according to "concepts", i.e. segmentation labels assigned to a large region of the image (for example "mountain"), and the style images are organized into groups (for exampe "fall colors") each of which contains several images. Then, for each concept-style pair $(c, s$ we generate a dataset of triples (x, x', y) where (x, y) is an ImageNet validation datapoint for which the concept occupies a large region in the segmentation map and x' is obtained by replacing the region of the image x segmented as the concept c with the style s, using Magenta's "arbitrary image stylization" model (Ghiasi et al., 2017). Images (a) and (b) of Figure 8 show an example datapoint.

The result of this pipeline is a large number of synthetic editing datasets $\mathcal{D}_{\text{edit}}(c, s)$: from 17 concepts and 8 styles we obtain 136 editing datasets, one per concept-style pair. The concepts and styles used are listed below.

Concepts "waterother", "clouds", "wallconcrete", "boat", "table", "mountain", "skyother", "diningtable", "light", "plantother", "tree", "sea", "person", "snow", "tv", "road", "grass".

Styles "falltrees", "snowyground", "woodentexture", "furrytexture", "blackandwhite", "colorfulflowers", "graffiti", "gravel".

[7] These segmentations were obtained with a DeepLabV2 segmentation model (Chen et al., 2017) trained on the COCO-Stuff dataset (Caesar et al., 2018).

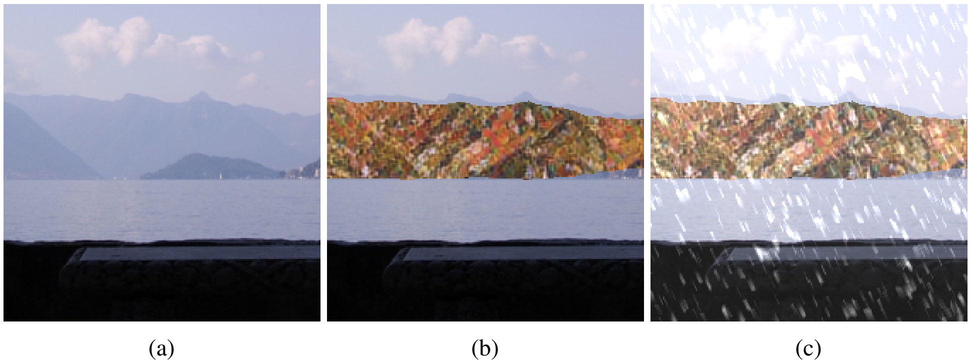


Figure 8: An example of an editing datapoint created using segmentation and style transfer. Image (a) is an ImageNet validation datapoint, with label 976 (promontory, headland, head, foreland). In image (b), the concept "mountain" has been replaced with the style "fall trees." In image (c), the corruption "snow" has been applied at a low severity level. Best viewed in color.

Our methodology for creating train-validation splits of such a synthetic dataset with concept c and style s is as follows. First, a number N_{train} is specified (in all of our editing experiments on ImageNet models it is 10), along with a "minimum training ratio" ρ (in all our experiments this is $1/2$). In addition, a number S_{train} is specified and the different images associated with the style s are broken into train and validation splits (in all our experiments, $S_{\text{train}} = 1$, i.e. we include only one variant of the style s in the train split, and the rest in the validation split). Then, we restrict attention to ImageNet class labels y with at least $N_{texttrain}/\rho$ associated images in the full synthetic dataset with concept c and style s. For each label y, we randomly sample[8] N_{train} corresponding datapoints (x, y), and for each of those datapoints we add the S_{train} triples (x, x', y) where x' is obtained by replacing concept c with a training split variant of the style s to $\mathcal{D}_{\text{edit}}^{\text{train}}(c, s)$. The remaining triples (x, x', y) are added to $\mathcal{D}_{\text{edit}}^{\text{val}}(c, s)$. A few comments on this splitting strategy:

- With our choices of the $N_{\text{train}} = 10$, $\rho = 1/2$ and $S_{\text{train}} = 1$ the resulting training set size is $|\mathcal{D}_{\text{edit}}^{\text{train}}(c, s)| = 10\cdot$ the number of ImageNet labels with ≥ 20 images in the dataset of segmentations provided by (Santurkar et al., 2021). This means that the training data size varies with the concept c (but not with the style s). It also means that for every datapoint appearing in $\mathcal{D}_{\text{edit}}^{\text{val}}(c, s)$ there is a datapoint of $\mathcal{D}_{\text{edit}}^{\text{train}}(c, s)$ with the same label. Hence this editing task is in some sense easier than that of (Santurkar et al., 2021), which required $\mathcal{D}_{\text{edit}}^{\text{train}}(c, s)$ to contain images from a single ImageNet class.
- (Santurkar et al., 2021) correctly points out that for some class labels y, replacing the concept c with the style s makes classification (even by a human) difficult if not impossible. They manually filter such label-concept-style combinations from their analysis. We did not perform such a filtration. On the other hand, as mentioned above we do not restrict our editing training sets to come from a single ImageNet class, and as such it is unlikely that problematic label-concept-style combinations dominate our editing training sets.
- In light of the above two points, note that we do not perform any analyses of how editing performance varies with editing training set size, nor any analyses of edited model performance on specific ImageNet classes.

We add corruptions to the validation sets $\mathcal{D}_{\text{edit}}^{\text{val}}(c, s)$ with the scripts used to generate ImageNet-C (Hendrycks & Dietterich, 2019). We use all 19 corruption types (including those denoted "extra" in ibid.) and all 5 severity levels. This results in 95 corrupted variants of each validation set $\mathcal{D}_{\text{edit}}^{\text{val}}(c, s)$. Image (c) in Figure 8 is an example of a corrupted concept-style-swapped image.

It must be acknowledged that in our evaluations of models edited using these synthetic datasets on ImageNet-C, there is a form of train-test leakage: the editing training sets $\mathcal{D}_{\text{edit}}^{\text{train}}(c, s)$ contain images from the (uncorrupted) ImageNet validation set, and our test set ImageNet-C consists of corrupted versions of the ImageNet validation set. This leakage is undesirable to say the least. In follow-on

[8] Random sampling (including the random sampling of S_{train} training style variants) is controlled by a fixed random seed.

work, we would like to re-run these experiments and instead produce $\mathcal{D}_{\text{edit}}^{\text{train}}(c, s)$ from images in the ImageNet training set, or alternatively (if it is important to model the deployment scenario where new datapoints outside the original training set are collected) from images in ImageNetV2. Note in the later case we would face leakage evaluating edited models on ImageNetV2.

We note, however, that the train-test leakage in our experiments was limited: in the worst case (COCO-Stuff label "person"), an edited model was exposed to 1.04% of the ImageNet validation set, and in the average case only 0.215%. Moreover, there are no literally identical images in the editing training set and ImageNet-C (although there certainly are corrupted versions of training images in ImageNet-C).

D.3 Editing method hyperparameters

All experiments in this paper are implemented in PyTorch (Paszke et al., 2019) (the one exception being the Magenta neural style transfer for synthetic editing dataset creation, where we use the original TensorFlow (Abadi et al., 2015) implementation) and run on NVIDIA GPUs.

In the editing methods of Appendix B.2, wherever applicable we use PyTorch's `torch.optim.SGD` with momentum 0.9 and decay 10^{-4} (only applied to parameters whose names contain "weight"). We do not update batch norm running means and variances during editing. Our reasoning is that over the course of editing optimization those statistics will move from those of the original training dataset towards those of the editing dataset, which could have a destructive on model performance on the original task or corrupted variants thereof. We allow editing optimization to proceed for a maximum of 10000 epochs, with an early stopping criterion that terminates optimization if the editing validation accuracy drops below half of its best-so-far. The large number of maximum epochs is chosen to account for the huge variation in optimal learning rates observed at different layers (or sets of layers) of different models; the idea here is to gaurd against the possibility of never observing editing learning do to using a too-small learning rate (which did occur in many preliminary test runs). The early stopping criterion is in place to offset the high worst-case computational cost of using 10000 epochs in every single edit — it seems to detect editing overfitting (which occurs quite regularly on the small datasets involved) in many cases, terminating optimization well before the maximum possible number of epochs is attained.

As mentioned above, effective[9] learning rates for the editing methods considered appear to span many orders of magnitude (based on preliminary tests, as low as 10^{-5} in some cases of supervised fine-tuning of later layers and as high as 100 in some cases of direct low-rank and rewriting). Effective learning rates depend on model, editing dataset, editing method and layer (for the editing methods where a model layer is specified). We first use preliminary tests on a smaller set of editing datasets including vehicles on snow and synthetic datasets for a subset of concept-style pairs, and where relevant with a subset of possible editing layers, to determine for each model and editing method a reasonable interval in which to search for learning rates. Then in our experiments we choose learning rates using cross validation over a grid search in the relevant reasonable learning rate interval.

In the case direct low-rank and rewriting, editing validation accuracy can exhibit substantial noise with respect to random initialization (of U and V in the case of direct low-rank, and of U in the case of rewriting). For these methods we run editing optimization several (≈ 10) times with different random initializations (within the learning rate hyperparameter search).

We record the best observed editing validation accuracy and save the corresponding best set of edited model weights for downstream robustness evaluations.

Our implementation of rewriting is a "from scratch" rewrite: here we use quotes since we of course heavily reference (Bau et al., 2020; Santurkar et al., 2021) and the `EditingClassifiers` codebase available at https://github.com/MadryLab/EditingClassifiers. The choice to use a custom implementation was based on multiple factors, including the goal of porting rewriting to image classifiers not appearing in `EditingClassifiers` (namely Vision Transformers), and a desire to have a relatively uniform functional interface for the various editing methods in our experiments. We plan to open source our research code.

[9] Here we say "effective" since optimal would be too strong (we do not do any fine-grained cross validation). By effective learning rate we just mean a learning rate that works, as in it decreases training loss (and hopefully increases validation accuracy in the process).

Unfortunately, there was a discrepancy between our version of rewriting and that in (Bau et al., 2020; Santurkar et al., 2021): specifically, the code used in our experiments applied mean centering the the features denoted K in (Bau et al., 2020, §3.3, eq. 14-15), thus essentially forcing KK^T to be a covariance matrix, rather than the second moment matrix as stated in ibid.[10] The code at https://github.com/davidbau/rewriting and https://github.com/MadryLab/EditingClassifiers clearly does not apply mean-centering. We are in the process of assessing the sensitivity of the results for the rewriting method presented above to this mean centering issue. One reason to suspect an application of mean centering does not make a large difference is that all layers of all models edited in our experiments are either immediately preceded by a batch normalization (or in the case of ViTs, a layer normalization) layer, or have a normalization layer within the preceding 3 or so layers. The nearby presence of normalization layers would potentially limit the magnitude of feature means.

For our PAINT (Ilharco et al., 2022) and our 1-LI interpolation and runs, we adapt the code and training hyperparameter defaults from https://github.com/mlfoundations/patching unless stated otherwise.

E Further related work

Model robustness and generalization: Robustness to distribution shift is an important topic in deep learning, in large part because it addresses a key challenge in real-world model deployment. Indeed, pioneering studies have revealed that even when models achieve superhuman performance on a held-out test set, their performance often degrades significantly on alternative test sets that are either mildly corrupted (Hendrycks & Dietterich, 2019) or out of distribution (Recht et al., 2019). Many of these works introduced datasets or tasks designed to evaluate various aspects of model robustness (Koh et al., 2021; Sagawa et al., 2022; Hendrycks & Dietterich, 2019; Hendrycks et al., 2021a; Geirhos et al., 2018). Other work that studies robustness aspects of weight updates constrained to modify a subset of model parameters is (Lee et al., 2023), which compares local fine-tuning of subsets of weights occurring in small contiguous blocks of layers. Our work differs from (Lee et al., 2023) in that we include a broader variety of parameter-constrained updates or edits, including low rank perturbations of weights as well as weight space interpolation (where using an interpolation parameter can be viewed as an implicit regularization keeping the updated weights closer to the initial pretrained weights).

Editing methods: The concept of model editing was first introduced in the context of generative adversarial networks (Bau et al., 2020). The authors developed a novel user interface in which users could specify desired image modifications ("remove text", "add people"); given these, specific weights of the model were then edited to achieve the desired result. In a striking example, users were able to generate the imaginative sight of trees growing out of the tops of cathedrals. The model edits were implemented by idealizing a neural network layer as a linear associative memory and then, based on that idealization, deriving a low-rank update to the layer weights. That approach has also been employed to alter the behavior of image classifiers (Santurkar et al., 2021) and autoregressive language models (Meng et al., 2022a;b).

In addition, a variety of alternative editing algorithms have been proposed. De Cao et al. (2021) and Mitchell et al. (2022a) trained hyper-networks to update model weights. (Dai et al., 2022) identified "knowledge neurons" and then surgically modified them. Mitchell et al. (2022b) augmented a target model with an auxiliary detector trained to identify inputs relevant to the editing task, and then diverted them to an edited model. See also (Hase et al., 2021) for analysis of beliefs stored in language models and methods to update those beliefs. Recent work also extended model editing to open vocabulary image-language models (e.g., CLIP), where it was demonstrated that interpolating between the original and edited weights allows users to navigate a trade-off between performance on the original and edited tasks (Ilharco et al., 2022).

Of particular relevance to our work are recent findings of Lee et al. (2023) that updating only a subset of model weights can result in greater robustness to distribution shift. We consider a different set of models and editing/fine-tuning tasks than that work, and update different subsets of weights (Lee et al. (2023) updates "blocks" of weights, whereas in many cases we update weights only in a single layer).

[10] One source of confusion was the later derivation of editing feature rank reduction in (Bau et al., 2020, §D) which idealizes the hidden features of the original (non-editing) dataset as a mean-zero Gaussian distribution, although it is never suggested to force them to be mean-zero in an implementation.

F Limitations and open questions

There are a range of important ways that the present study could have been made more comprehensive. Large language models are an important application area of editing techniques which we were not able to cover in this work. There is also a wide range of methods for robustness to different types of distribution shift. Of particular interest would be those approaches that either capture or simulate real-world distribution shift (such as (Koh et al., 2021)). Prior work on model editing has studied aspects of the generalization capabilites of edited models on inputs related to the editing task not considered in this work, where we chose to focus on robustness to shifts of natural image distribtions. In particular, it would be interesting to know whether or not direct low-rank editing and 1-LI enjoy the same benefits as rewriting in the cases where the editing training and validation set are derived from ImageNet images with disjoint underlying sets of labels (see Figs. 3, 5 of (Santurkar et al., 2021)) Finally, in performing our experiments, a large number of hyperparameters needed to be chosen. While we strove to optimize all of these, it remains possible that further refinement may have improved the performance of some of the editing methods.

G Acknowledgments

This research was supported by the Mathematics for Artificial Reasoning in Science (MARS) initiative at Pacific Northwest National Laboratory. It was conducted under the Laboratory Directed Research and Development (LDRD) Program at at Pacific Northwest National Laboratory (PNNL), a multiprogram National Laboratory operated by Battelle Memorial Institute for the U.S. Department of Energy under Contract DE-AC05-76RL01830.