The Empirical Impact of Neural Parameter Symmetries, or Lack Thereof

Derek Lim* DEREKLIM@MIT.EDU
MIT CSAIL

Theo (Moe) Putterman* MOEPUTTERMAN@BERKELEY.EDU
UC Berkeley

Robin Walters
Northeastern University

Haggai Maron
Technion, NVIDIA

Stefanie Jegelka
TU Munich, MIT

Abstract

Many algorithms and observed phenomena in deep learning appear to be affected by parameter symmetries — transformations of neural network parameters that do not change the underlying neural network function. These include linear mode connectivity, model merging, Bayesian neural network inference, metanetworks, and several other characteristics of optimization or loss-landscapes. In this work, we empirically investigate the impact of neural parameter symmetries by introducing new neural network architectures that have reduced parameter space symmetries. We develop two methods, with some provable guarantees, of modifying standard neural networks to reduce parameter space symmetries. With these new methods, we conduct a comprehensive experimental study consisting of multiple tasks aimed at assessing the effect of removing parameter symmetries. Our experiments reveal several interesting observations on the empirical impact of parameter symmetries; for instance, we observe linear mode connectivity and monotonic linear interpolation in our networks, without any alignment of weight spaces.

1. Introduction

Neural networks have found profound empirical success, but have many associated behaviors and phenomena that are difficult to understand. One important property of neural networks is that they generally have many *parameter space symmetries* — for any set of parameters, there are typically many other choices of parameters that correspond to the same exact neural network function [23]. These parameter symmetries are a type of (not-necessarily detrimental) redundancy in the parameterization of neural networks, that adds much non-Euclidean structure to parameter space.

Parameter space symmetries appear to influence several phenomena observed in neural networks. For instance, parameter symmetries appear to play a role in interpretability of neurons [19], optimization [46, 76, 80], model merging [58], learned equivariance [6], Bayesian deep learning [33], loss

. *Denotes equal contribution.

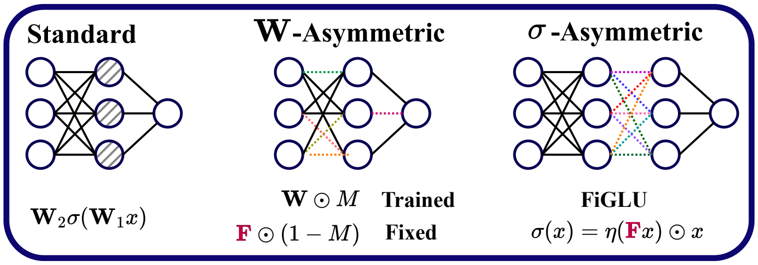
Standard
$\mathbf{W}_2\sigma(\mathbf{W}_1x)$
$\mathbf{W}$-Asymmetric
$\mathbf{W}\odot M$ Trained
$\mathbf{F}\odot(1-M)$ Fixed
σ-Asymmetric
FiGLU
$\sigma(x)=\eta(\mathbf{F}x)\odot x$

Figure 1: (Left) Standard MLP. The hidden nodes can be freely permuted, which induces permutation parameter symmetries. Black edges denote trainable parameters. (Middle) Our $\mathbf{W}$-Asymmetric MLP, which fixes certain weights to be constant (colored dashed lines) to break parameter symmetries. (Right) Our σ-Asymmetric MLP, which uses our FiGLU nonlinearity with a fixed matrix $\mathbf{F}$ (colored dashed lines) to break parameter symmetries.

landscape geometry [52], processing neural network weights as input data using metanetworks [38], and generalization measures [10, 47].

To rigorously study the effect of parameter symmetries, we study the effect of removing them. In particular, we introduce two ways of modifying neural network architectures to remove parameter space symmetries (see Figure 1):

(1) $\mathbf{W}$-Asymmetric networks fix certain elements of each linear map to break symmetries in the computation graph.
(2) σ-Asymmetric networks use a new nonlinearity (FiGLU) that does not act elementwise, and hence does not induce symmetries such as permutations.

We theoretically prove that both of our approaches remove parameter symmetries under certain conditions. Our Asymmetric networks are similar structurally to standard networks and can be trained with standard backpropagation and first-order optimization algorithms like Adam. Thus, they are a reasonable "counterfactual" system for studying neural networks without parameter symmetries.

With our Asymmetric networks, we run a suite of experiments to study the effects of removing parameter symmetries on several base architectures, including MLPs, ResNets, and graph neural networks. For example, through the lenses of linear mode connectivity [17] and monotonic linear interpolation [40], we see that the loss landscapes of our Asymmetric networks are remarkably more well-behaved and closer to convex than the loss landscapes of standard neural networks. Overall, our Asymmetric networks provide valuable insights for empirical study and hold promise for advancing our understanding of the impact of neural parameter symmetries.

2. Background and Definitions

Let Θ be the space of parameters of a fixed neural network architecture. For any choice of parameters $\theta \in \Theta$, we have a neural network function $f_\theta : \mathcal{X} \to \mathcal{Y}$ from an input space $\mathcal{X}$ to an output space $\mathcal{Y}$. We call a function $\phi : \Theta \to \Theta$ a *parameter space symmetry* if $f_\theta(x) = f_{\phi(\theta)}(x)$ for all inputs x and parameters $\theta \in \Theta$ (i.e. if f_θ and $f_{\phi(\theta)}$ are always the same function).

For instance, consider a two-layer MLP with no biases, parameterized by matrices $\theta = (\mathbf{W}_2, \mathbf{W}_1)$ with an elementwise nonlinearity σ. Then $f_\theta(x) = \mathbf{W}_2\sigma(\mathbf{W}_1x)$. Let P be a permutation matrix,

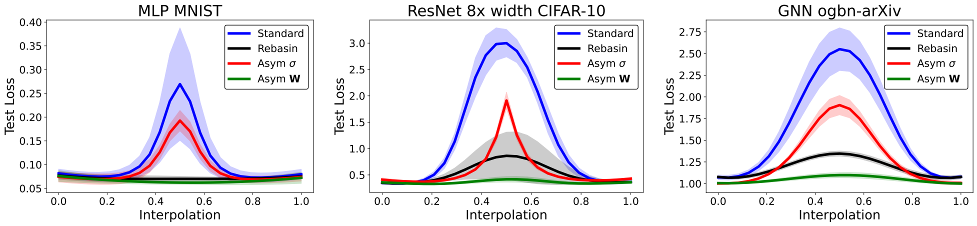

Figure 2: Linear mode connectivity: test loss curves along linear interpolations between trained networks. (Left) MLP on MNIST. (Middle) ResNet with $8\times$ width on CIFAR-10. (Right) GNN on ogbn-arXiv. $\mathbf{W}$-Asymmetric networks interpolate the best, followed by networks aligned with Git-ReBasin, then σ-Asymmetric networks, and finally standard networks.

and let $\phi(\theta) = (\mathbf{W}_2 P^\top, P\mathbf{W}_1)$. Then for any input x,

$$f_{\phi(\theta)}(x) = \mathbf{W}_2 P^\top \sigma(P\mathbf{W}_1 x) = \mathbf{W}_2 P^\top P \sigma(\mathbf{W}_1 x) = \mathbf{W}_2 \sigma(\mathbf{W}_1 x) = f_\theta(x), \tag{1}$$

so ϕ is a parameter space symmetry. A key step is the second equality, which holds because $P\sigma(x) = \sigma(Px)$: any elementwise nonlinearity σ is permutation equivariant. Any other equivariance of σ also induces a parameter symmetry; for instance, if $\sigma(x) = \max(0, x)$ is the ReLU function, then $\alpha\sigma(x) = \sigma(\alpha x)$ for any $\alpha > 0$, so there is a positive-scaling-based parameter symmetry [10, 19, 47].

3. Asymmetric Networks

Here, we introduce our two types of Asymmetric Networks. We explain how to make Asymmetric MLPs with no biases and cover extensions to other architectures in Appendix B.

3.1. Computation Graph Approach (W-Asymmetric Networks)

Our first approach to developing neural networks with reduced parameter space symmetries relies on their computation graph. In particular, we can write a feedforward neural network architecture as a DAG $G = (V, E)$ with neurons as nodes V and connections between them as edges E. For a choice of parameters $\theta \in \mathbb{R}^{|E|}$, we get a function f_θ from input neuron space to output neuron space [18, 38, 47]. Lim et al. [38] showed that neural DAG automorphisms ϕ, which are graph automorphisms of the DAG G that preserve types of nodes and weight-sharing constraints, induce permutation parameter symmetries ϕ that leave the function unchanged: $f_\theta = f_{\phi(\theta)}$. Thus, any feedforward neural network architecture that has no permutation parameter symmetries must necessarily have a computation graph with no nontrivial neural DAG automorphisms.

To remove nontrivial neural DAG automorphisms from MLPs, we mask edges in the computation graph, by setting certain edge weights to constant values that are not updated during training. For an MLP, we can do this by enforcing that every linear layer $T : \mathbb{R}^{d_1} \to \mathbb{R}^{d_2}$ takes the form of a matrix $\mathbf{W} \in \mathbb{R}^{d_2 \times d_1}$, where each row has a unique pattern of untrained weights. To achieve this, define a mask $M \in \{0,1\}^{d_2 \times d_1}$ such that $\mathbf{W}_{ij}$ is a trainable parameter if and only if $M_{ij} = 1$, and the rows of M are pairwise distinct binary vectors in $\{0,1\}^{d_1}$. We call any neural network with linear maps masked as such a $\mathbf{W}$-Asymmetric neural network. In Appendix D.1, we show that masking these entries so that they are not trained is sufficient to remove all nontrivial neural DAG automorphisms.

Theorem 1 *If each mask matrix M has unique nonzero rows, then $\mathbf{W}$-Asymmetric MLPs with fixed entries set to zero have no nontrivial neural DAG automorphisms.*

In practice, we generate a binary mask M by randomly selecting a subset of n_{fix} fixed elements for each row. For the fixed entries, we sample them from a normal distribution $\mathcal{N}(0, \kappa I)$ with standard deviation $\kappa > 0$ that we tune. Our asymmetric linear layer can be written as

$$\mathbf{W}' = M \odot \mathbf{W} + (1 - M) \odot \mathbf{F}, \tag{2}$$

where $\mathbf{W} \in \mathbb{R}^{d_2 \times d_1}$ is a matrix of trainable parameters, and $\mathbf{F} \in \mathbb{R}^{d_2 \times d_1}$ is a matrix of fixed elements, sampled from $\mathcal{N}(0, \kappa I)$.

3.2. Nonlinearity Approach (σ-Asymmetric Networks)

Another approach for removing parameter symmetries is to change the nonlinearity. As studied by Godfrey et al. [19], equivariances of the nonlinearity induce parameter symmetries in MLPs. That is, if $\sigma \circ A = B \circ \sigma$ for $A, B \in GL(d)$, then for a two-layer network:

$$\mathbf{W}_2 \circ \sigma \circ \mathbf{W}_1 = \mathbf{W}_2 B^{-1} B \circ \sigma \circ \mathbf{W}_1 = \mathbf{W}_2 B^{-1} \circ \sigma \circ A \mathbf{W}_1. \tag{3}$$

So $(\mathbf{W}_2, \mathbf{W}_1)$ and $(\mathbf{W}_2 B^{-1}, A\mathbf{W}_1)$ give the same neural network function. Thus, in order to define a model class without parameter symmetries, it is necessary for σ to have *no linear equivariances*, i.e. we desire that if $\sigma \circ A = B \circ \sigma$ for $A, B \in GL(d)$, then $A = B = I$. Under certain conditions, this is in fact sufficient to remove all parameter symmetries: we prove this in Appendix D.2.

3.2.1. FIGLU: THE FIXED GATED LINEAR UNIT NONLINEARITY

Thus, we define a non-elementwise nonlinearity that does not have the equivariances of standard nonlinearities. Letting η be the sigmoid function $\eta(x) = \frac{1}{1+e^{-x}}$, we define our nonlinearity as

$$\sigma(x) = \eta(\mathbf{F}x) \odot x, \tag{4}$$

for a randomly sampled, untrained matrix $\mathbf{F}$. In this work, we sample $\mathbf{F}$ as an i.i.d Gaussian matrix with variance that we tune. This nonlinearity is similar to Swish / SiLU [25, 55] with an additional matrix $\mathbf{F}$ to mix feature dimensions, and it is also similar to a gated linear unit (GLU) with no trainable parameters [8]. Thus, we call our nonlinearity FiGLU: the Fixed Gated Linear Unit.

In Appendix D.2.1, we prove that FiGLU does not have permutation equivariances or diagonal equivariances, which are the only equivariances for most elementwise nonlinearities [19].

Proposition 2 *With probability* 1 *over the sampling of* $\mathbf{F}$*, FiGLU has no permutation equivariances or diagonal equivariances.*

We call any network with our symmetry-breaking FiGLU nonlinearity a σ-Asymmetric Network.

4. Experiments

4.1. Linear Mode Connectivity without Permutation Alignment

Background. Many works have studied linear mode connectivity, which is when all networks on the line segment in parameter space between two well-performing trained networks are also well-performing [12, 17, 42]. When the two networks are randomly initialized and trained independently, linear mode connectivity generally does not hold [1, 12]. However, if one of the two networks is

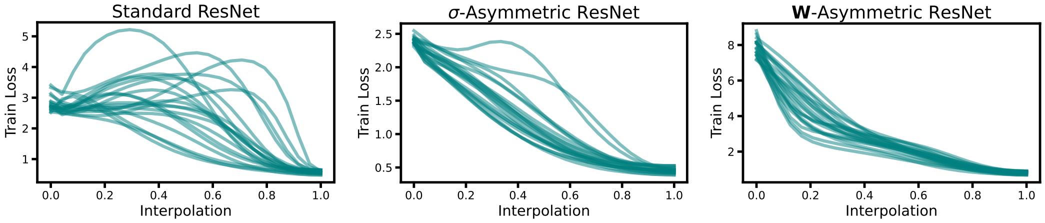

Figure 3: Train loss vs. α for the interpolation $(1-\alpha)\theta_0 + \alpha\theta_T$ between initial parameters θ_0 and trained parameters θ_T. Trajectories for the 20 (θ_0, θ_T) pairs of lowest train loss for each architecture are plotted. The trajectories for Asymmetric ResNets appear significantly more monotonic and convex.

permuted with a parameter symmetry that does not change its function, but that aligns its parameters with the other network, then linear mode connectivity empirically and theoretically holds for many more model / task combinations [1, 12, 13, 75]. Since our Asymmetric networks remove parameter space symmetries, we may expect linear mode connectivity to hold, without any post-processing.

Hypothesis. Asymmetric networks are more linearly mode connected than standard networks, and do not require post-processing or alignment of pairs of networks before merging.

Experimental Setup. We consider several networks and tasks: MLPs on MNIST, ResNets [22] on CIFAR-10, and GNNs [71] on ogbn-arXiv [26]. For each architecture and task, we compute the midpoint test loss barrier: $L(\frac{1}{2}\theta_1 + \frac{1}{2}\theta_2) - \frac{1}{2}(L(\theta_1) + L(\theta_2))$. We train standard networks, pairs of networks aligned with Git-Rebasin [1], σ-Asym networks, and $\mathbf{W}$-Asym networks.

Results. Figure 2 plots interpolation curves and Table 1 in Appendix C.1 displays midpoint test loss barriers of various methods. Our σ-Asymmetric approach lowers the test loss barrier compared to standard networks, but falls short of the alignment approach of Git-Rebasin. On the other hand, our $\mathbf{W}$-Asymmetric approach achieves strong (and sometimes perfect) interpolation, and interpolates better than standard networks aligned via Git-ReBasin. This may be caused by failure of the Git-ReBasin approaches to find the optimal permutations, importance of other parameter symmetries besides layer-wise permutations, or other properties of $\mathbf{W}$-Asymmetric networks.

4.2. Monotonic Linear Interpolation

Background. Several works have studied the line segment between parameters at initialization and parameters after training. For some models and tasks, different works have observed *monotonic linear interpolation* (MLI) — where the training loss monotonically decreases along this line segment [16, 20, 40, 66, 68]. Loss landscapes of convex problems have this property as well, so presence of MLI has been used as a rough measure of how well-behaved the loss landscape is. Since removing parameter symmetries substantially improves linear interpolation between trained networks (Section 4.1), one may expect removing parameter symmetries to improve MLI.

Hypothesis. The training loss along the line segment between initialization and trained parameters is more monotonic and convex for Asymmetric networks.

Experimental setup. For the learning task, we train standard ResNets and $\mathbf{W}$-Asymmetric ResNets with varying hyperparameters sampled from the distributions in Appendix Table 8. For each of these networks, we linearly interpolate between its initial parameters θ_0 and its final trained parameters θ_T: $(1-\alpha)\theta_0 + \alpha\theta_T$ for $\alpha \in [0, 1]$.

Results. Qualitatively, we can see in Figure 3 that $\mathbf{W}$-Asymmetric ResNets do not have any clear loss barriers from initialization, nor any loss plateaus that indicate nonconvexity. In contrast, the majority of standard ResNets have non-monotonic trajectories, and the monotonic trajectories seem to be more nonconvex. σ-Asymmetric network trajectories are signficantly more convex and monotonic than standard network trajectories, but there are some non-monotonic or nonconvex trajectories still. In Appendix C.2, we see the same trends in quantitative measurements of monotonicity and convexity.

4.3. Other Optimization and Loss Landscape Properties

In the Appendix, we show that Asymmetric networks make for better base models in Bayesian deep learning (C.3), are easier to predict properties of using metanetworks (C.4), and differ from standard networks in other properties of optimization and loss landscape (C.5).

Acknowledgements

We would like to thank Kwangjun Ahn, Benjamin Banks, Nima Dehmamy, Nikos Karalias, Jinwoo Kim, Marc Law, Hannah Lawrence, Thien Le, Jonathan Lorraine, James Lucas, Behrooz Tahmasebi, and Logan Weber for discussions at various points of this project. DL is supported by an NSF Graduate Fellowship. RW is supported in part by NSF award 2134178. HM is the Robert J. Shillman Fellow, and is supported by the Israel Science Foundation through a personal grant (ISF 264/23) and an equipment grant (ISF 532/23). This research was supported in part by Office of Naval Research grant N00014-20-1-2023 (MURI ML-SCOPE), NSF AI Institute TILOS (NSF CCF-2112665), NSF award 2134108, and the Alexander von Humboldt Foundation.

References

[1] Samuel Ainsworth, Jonathan Hayase, and Siddhartha Srinivasa. Git re-basin: Merging models modulo permutation symmetries. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=CQsmMYmlP5T.

[2] Laurence Aitchison, Adam Yang, and Sebastian W Ober. Deep kernel processes. In *International Conference on Machine Learning*, pages 130–140. PMLR, 2021.

[3] Stephen Ashmore and Michael Gashler. A method for finding similarity between multi-layer perceptrons by forward bipartite alignment. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE, 2015.

[4] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[5] Vijay Badrinarayanan, Bamdev Mishra, and Roberto Cipolla. Understanding symmetries in deep networks. *arXiv preprint arXiv:1511.01029*, 2015.

[6] Georg Bökman and Fredrik Kahl. Investigating how reLU-networks encode symmetries. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=8lbFwpebeu.

[7] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[8] Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks. In *International conference on machine learning*, pages 933–941. PMLR, 2017.

[9] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

[10] Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. In *International Conference on Machine Learning*, pages 1019–1028. PMLR, 2017.

[11] Gabriel Eilertsen, Daniel Jönsson, Timo Ropinski, Jonas Unger, and Anders Ynnerman. Classifying the classifier: dissecting the weight space of neural networks. *arXiv preprint arXiv:2002.05688*, 2020.

[12] Rahim Entezari, Hanie Sedghi, Olga Saukh, and Behnam Neyshabur. The role of permutation invariance in linear mode connectivity of neural networks. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=dNigytemkL.

[13] Damien Ferbach, Baptiste Goujaud, Gauthier Gidel, and Aymeric Dieuleveut. Proving linear mode connectivity of neural networks via optimal transport. In *International Conference on Artificial Intelligence and Statistics*, pages 3853–3861. PMLR, 2024.

[14] Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019.

[15] Marc Finzi, Max Welling, and Andrew Gordon Wilson. A practical method for constructing equivariant multilayer perceptrons for arbitrary matrix groups. In *International conference on machine learning*, pages 3318–3328. PMLR, 2021.

[16] Jonathan Frankle. Revisiting "qualitatively characterizing neural network optimization problems", 2020.

[17] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis. In *International Conference on Machine Learning*, pages 3259–3269. PMLR, 2020.

[18] Cedric Gegout, Bernard Girau, and Fabrice Rossi. *A mathematical model for feed-forward neural networks: theoretical description and parallel applications.* PhD thesis, Laboratoire de l'informatique du parallélisme, 1995.

[19] Charles Godfrey, Davis Brown, Tegan Emerson, and Henry Kvinge. On the symmetries of deep learning models and their internal representations. *Advances in Neural Information Processing Systems*, 35:11893–11905, 2022.

[20] Ian J Goodfellow, Oriol Vinyals, and Andrew M Saxe. Qualitatively characterizing neural network optimization problems. *ICLR*, 2015.

[21] William L Hamilton. *Graph representation learning*. Morgan & Claypool Publishers, 2020.

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[23] Robert Hecht-Nielsen. On the algebraic structure of feedforward network weight spaces. In *Advanced Neural Computers*, pages 129–135. Elsevier, 1990.

[24] Robert Hecht-Nielsen. Theory of the backpropagation neural network. In *Neural networks for perception*, pages 65–93. Elsevier, 1992.

[25] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

[26] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133, 2020.

[27] Moritz Imfeld, Jacopo Graldi, Marco Giordano, Thomas Hofmann, Sotiris Anagnostidis, and Sidak Pal Singh. Transformer fusion with optimal transport. *arXiv preprint arXiv:2310.05719*, 2023.

[28] Keller Jordan, Hanie Sedghi, Olga Saukh, Rahim Entezari, and Behnam Neyshabur. REPAIR: REnormalizing permuted activations for interpolation repair. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=gU5sJ6ZggcX.

[29] Laurent Valentin Jospin, Hamid Laga, Farid Boussaid, Wray Buntine, and Mohammed Bennamoun. Hands-on bayesian neural networks—a tutorial for deep learning users. *IEEE Computational Intelligence Magazine*, 17(2):29–48, 2022.

[30] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[31] Miltiadis Kofinas, Boris Knyazev, Yan Zhang, Yunlu Chen, Gertjan J Burghouts, Efstratios Gavves, Cees GM Snoek, and David W Zhang. Graph neural networks for learning equivariant representations of neural networks. *arXiv preprint arXiv:2403.12143*, 2024.

[32] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images, 2009.

[33] Richard Kurle, Ralf Herbrich, Tim Januschowski, Yuyang Bernie Wang, and Jan Gasthaus. On the detrimental effect of invariances in the likelihood for variational inference. *Advances in Neural Information Processing Systems*, 35:4531–4542, 2022.

[34] Lauro Langosco, Neel Alex, William Baker, David John Quarel, Herbie Bradley, and David Krueger. Towards meta-models for automated interpretability, 2024. URL https://openreview.net/forum?id=fM1ETm3ssl.

[35] Olivier Laurent, Emanuel Aldea, and Gianni Franchi. A symmetry-aware exploration of bayesian neural network posteriors. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=FOSBQuXgAq.

[36] Guillaume Leclerc, Andrew Ilyas, Logan Engstrom, Sung Min Park, Hadi Salman, and Aleksander Mądry. Ffcv: Accelerating training by removing data bottlenecks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12011–12020, 2023.

[37] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[38] Derek Lim, Haggai Maron, Marc T. Law, Jonathan Lorraine, and James Lucas. Graph metanetworks for processing diverse neural architectures. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=ijK5hyxs0n.

[39] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2018.

[40] James Lucas, Juhan Bae, Michael R Zhang, Stanislav Fort, Richard Zemel, and Roger Grosse. Analyzing monotonic linear interpolation in neural network loss landscapes. *arXiv preprint arXiv:2104.11044*, 2021.

[41] Haggai Maron, Ethan Fetaya, Nimrod Segol, and Yaron Lipman. On the universality of invariant networks. In *International conference on machine learning*, pages 4363–4371. PMLR, 2019.

[42] Seyed Iman Mirzadeh, Mehrdad Farajtabar, Dilan Gorur, Razvan Pascanu, and Hassan Ghasemzadeh. Linear mode connectivity in multitask and continual learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=Fmg_fQYUejf.

[43] Mahdi Pakdaman Naeini, Gregory Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 29, 2015.

[44] Aviv Navon, Aviv Shamsian, Idan Achituve, Ethan Fetaya, Gal Chechik, and Haggai Maron. Equivariant architectures for learning in deep weight spaces. In *International Conference on Machine Learning*, pages 25790–25816. PMLR, 2023.

[45] Aviv Navon, Aviv Shamsian, Ethan Fetaya, Gal Chechik, Nadav Dym, and Haggai Maron. Equivariant deep weight space alignment. *arXiv preprint arXiv:2310.13397*, 2023.

[46] Behnam Neyshabur, Russ R Salakhutdinov, and Nati Srebro. Path-sgd: Path-normalized optimization in deep neural networks. *Advances in neural information processing systems*, 28, 2015.

[47] Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. Norm-based capacity control in neural networks. In *Conference on learning theory*, pages 1376–1401. PMLR, 2015.

[48] Theodore Papamarkou, Jacob Hinkle, M Todd Young, and David Womble. Challenges in markov chain monte carlo for bayesian neural networks. *Statistical Science*, 37(3):425–442, 2022.

[49] Theodore Papamarkou, Maria Skoularidou, Konstantina Palla, Laurence Aitchison, Julyan Arbel, David Dunson, Maurizio Filippone, Vincent Fortuin, Philipp Hennig, Aliaksandr Hubin, et al. Position paper: Bayesian deep learning in the age of large-scale ai. *arXiv preprint arXiv:2402.00809*, 2024.

[50] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[51] Fidel A Guerrero Peña, Heitor Rapela Medeiros, Thomas Dubail, Masih Aminbeidokhti, Eric Granger, and Marco Pedersoli. Re-basin via implicit sinkhorn differentiation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20237–20246, 2023.

[52] Fabrizio Pittorino, Antonio Ferraro, Gabriele Perugini, Christoph Feinauer, Carlo Baldassi, and Riccardo Zecchina. Deep networks on toroids: removing symmetries reveals the structure of flat regions in the landscape geometry. In *International Conference on Machine Learning*, pages 17759–17781. PMLR, 2022.

[53] Arya A Pourzanjani, Richard M Jiang, and Linda R Petzold. Improving the identifiability of neural networks for bayesian inference. In *NIPS workshop on bayesian deep learning*, volume 4, page 31, 2017.

[54] Xingyu Qu and Samuel Horvath. Rethink model re-basin and the linear mode connectivity. *arXiv preprint arXiv:2402.05966*, 2024.

[55] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for activation functions, 2017.

[56] Aviv Shamsian, Aviv Navon, David W Zhang, Yan Zhang, Ethan Fetaya, Gal Chechik, and Haggai Maron. Improved generalization of weight space networks via augmentations. *arXiv preprint arXiv:2402.04081*, 2024.

[57] Sidak Pal Singh and Martin Jaggi. Model fusion via optimal transport. *Advances in Neural Information Processing Systems*, 33:22045–22055, 2020.

[58] George Stoica, Daniel Bolya, Jakob Brandt Bjorner, Pratik Ramesh, Taylor Hearn, and Judy Hoffman. Zipit! merging models from different tasks without training. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=LEYUkvdUhq.

[59] Héctor J Sussmann. Uniqueness of the weights for minimal feedforward nets with a given input-output map. *Neural networks*, 5(4):589–593, 1992.

[60] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[61] Norman Tatro, Pin-Yu Chen, Payel Das, Igor Melnyk, Prasanna Sattigeri, and Rongjie Lai. Optimizing mode connectivity via neuron alignment. *Advances in Neural Information Processing Systems*, 33:15300–15311, 2020.

[62] Marcin Tomczak, Siddharth Swaroop, and Richard Turner. Efficient low rank gaussian variational inference for neural networks. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 4610–4622. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/310cc7ca5a76a446f85c1a0d641ba96d-Paper.pdf.

[63] Thomas Unterthiner, Daniel Keysers, Sylvain Gelly, Olivier Bousquet, and Ilya Tolstikhin. Predicting neural network accuracy from weights. *arXiv preprint arXiv:2002.11448*, 2020.

[64] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[65] Neha Verma and Maha Elbayad. Merging text transformer models from different initializations. *arXiv preprint arXiv:2403.00986*, 2024.

[66] Tiffany J. Vlaar and Jonathan Frankle. What can linear interpolation of neural network loss landscapes tell us? *ICML*, 2022.

[67] Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris Papailiopoulos, and Yasaman Khazaeni. Federated learning with matched averaging. *arXiv preprint arXiv:2002.06440*, 2020.

[68] Xiang Wang, Annie N. Wang, Mo Zhou, and Rong Ge. Plateau in monotonic linear interpolation — a ”biased” view of loss landscape for deep networks. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=z289SIQOQna.

[69] Jonas Gregor Wiese, Lisa Wimmer, Theodore Papamarkou, Bernd Bischl, Stephan Günnemann, and David Rügamer. Towards efficient mcmc sampling in bayesian neural networks by exploiting symmetry. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 459–474. Springer, 2023.

[70] Tim Z Xiao, Weiyang Liu, and Robert Bamler. A compact representation for bayesian neural networks by removing permutation symmetry. *arXiv preprint arXiv:2401.00611*, 2023.

[71] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ryGs6iA5Km.

[72] Chulhee Yun, Srinadh Bhojanapalli, Ankit Singh Rawat, Sashank Reddi, and Sanjiv Kumar. Are transformers universal approximators of sequence-to-sequence functions? In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=ByxRM0Ntvr.

[73] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. *Advances in neural information processing systems*, 30, 2017.

[74] Bo Zhao, Iordan Ganev, Robin Walters, Rose Yu, and Nima Dehmamy. Symmetries, flat minima, and the conserved quantities of gradient flow. *arXiv preprint arXiv:2210.17216*, 2022.

[75] Bo Zhao, Nima Dehmamy, Robin Walters, and Rose Yu. Understanding mode connectivity via parameter space symmetry. In *UniReps: the First Workshop on Unifying Representations in Neural Models*, 2023.

[76] Bo Zhao, Robert M. Gower, Robin Walters, and Rose Yu. Improving convergence and generalization using parameter symmetries. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=L0r0GphlIL.

[77] Allan Zhou, Kaien Yang, Kaylee Burns, Adriano Cardace, Yiding Jiang, Samuel Sokota, J Zico Kolter, and Chelsea Finn. Permutation equivariant neural functionals. *Advances in Neural Information Processing Systems*, 36, 2023.

[78] Allan Zhou, Kaien Yang, Yiding Jiang, Kaylee Burns, Winnie Xu, Samuel Sokota, J Zico Kolter, and Chelsea Finn. Neural functional transformers. *Advances in Neural Information Processing Systems*, 36, 2023.

[79] Allan Zhou, Chelsea Finn, and James Harrison. Universal neural functionals. *arXiv preprint arXiv:2402.05232*, 2024.

[80] Liu Ziyin. Symmetry leads to structured constraint of learning. *arXiv preprint arXiv:2309.16932*, 2023.

Appendix A. Related Work

Characterizing parameter space symmetries. While many works spanning several decades have noted specific parameter space symmetries in neural networks [23, 59], some works take a more systematic approach to deriving parameter space symmetries. Godfrey et al. [19] characterize all global linear symmetries induced by the nonlinearity for two-layer multi-layer perceptrons with pointwise nonlinearities. Zhao et al. [74] study several types of symmetries, and derive nonlinear, data-dependent parameter space symmetries. Lim et al. [38] show that graph automorphisms of the computation graph of a neural network induce permutation parameter symmetries, which captures hidden neuron permutations in MLPs and hidden channel permutations in CNNs.

Constraints and post-processing to break parameter space symmetries. Several works develop methods for constraining or post-processing the weights of a single neural network to remove ambiguities from parameter space symmetries. This includes methods to remove scaling symmetries induced by normalization layers or positively-homogeneous nonlinearities such as ReLU [5, 35, 52, 53], methods to remove permutation symmetries [35, 52, 53, 69], and methods to remove sign symmetries induced by odd activation functions [69]. Unlike these previous works, we develop neural network architectures that have reduced parameter space symmetries. Our models are optimized using standard unconstrained gradient-descent based methods like Adam. Hence, our networks do not require any non-standard optimization algorithms such as manifold optimization or projected gradient descent [5, 53], nor do they require post-training-processing to remove symmetries or special care during analysis of parameters (such as geodesic interpolation in a Riemannian weight space [52]).

Aligning multiple networks for relative invariance to parameter symmetries. One way to reduce the impact of parameter symmetries in certain settings, especially for model merging, is to align the parameters of one network to another. Several methods have been proposed for choosing permutations that align the parameters of two neural networks of the same architecture, via efficient heuristics or learned methods [1, 3, 12, 45, 51, 61, 65, 67]. Other approaches relax the exact permutation-parameter-symmetry constraint or do additional postprocessing to achieve effective merging of models in parameter space [27, 28, 54, 57, 58]. As our Asymmetric networks have removed parameter symmetries, they can often be successfully merged and linearly interpolated between without any alignment.

Table 1: Test loss interpolation barriers at midpoint: $L(\frac{1}{2}\theta_1 + \frac{1}{2}\theta_2) - \frac{1}{2}(L(\theta_1) + L(\theta_2))$. We use different methods of breaking symmetries in each column; from left to right: no symmetry breaking, Git-Rebasin [1], our σ-Asym approach, and our $\mathbf{W}$-Asym approach. We report mean and standard deviation of the barrier across at least 5 pairs of networks, and bold lowest barriers.

	Standard	Git-ReBasin	σ-Asym (ours)	$\mathbf{W}$-Asym (ours)
MLP (MNIST)	$0.188 \pm .12$	$-.006 \pm .00$	$0.117 \pm .01$	$-\mathbf{0.012} \pm .00$
ResNet (CIFAR-10)	$3.287 \pm .32$	$2.041 \pm .21$	$2.521 \pm .46$	$\mathbf{0.934} \pm .72$
ResNet 8x width (CIFAR-10)	$2.640 \pm .24$	$0.509 \pm .45$	$1.492 \pm .15$	$\mathbf{0.031} \pm .05$
GNN (ogbn-arXiv)	$1.475 \pm .24$	$0.269 \pm .02$	$0.901 \pm .11$	$\mathbf{0.095} \pm .03$

Appendix B. Extension of Asymmetric Networks to Other Architectures

The graph-based approach ($\mathbf{W}$-Asymmetric Networks) works naturally for neural network architectures with "channel" dimensions, such as convolutional neural networks (CNNs), graph neural networks (GNNs) [21], Transformers [64], and equivariant neural networks based on equivariant linear maps [15]. In these types of networks, permutations of entire channels induce permutation parameter symmetries [38]. For such networks, we thus mask entire connections between channels, e.g. entire filters in CNNs. For CNNs, we also experiment with randomly masking some number of entries in each filter (instead of masking entire filters), and find that this also works well in removing parameter symmetries.

The nonlinearity-based approach (σ-Asymmetric Networks) can be straightforwardly applied to many general architectures as well. Though, the fixed matrix $\mathbf{F}$ may have to be changed to a structured linear map; for instance, in CNNs we take $\mathbf{F}$ to be a 1D convolution.

Appendix C. Additional Experiments

C.1. Linear Mode Connectivity

Table 1 shows average test loss barrier for different types of parameter-symmetry-breaking on several architectures and tasks. We see that $\mathbf{W}$-Asymmetric networks interpolate very well, while σ-Asymmetric networks do not interpolate as well (although they do improve over standard networks).

C.2. Monotonic Linear Interpolation

Table 2: Monotonic linear interpolation: properties of linear interpolations between 300 pairs of initialization and trained parameters. Arrows denote behavior that is more similar to convex optimization, e.g. there is a downarrow (↓) next to Δ because convex objectives have nonpositive Δ, while nonconvex can have positive Δ. For both types of Asymmetric networks, all differences from Standard ResNets are statistically significant ($p < .001$) under a two-sided T-test: Asymmetric networks have significantly more monotonic and convex linear interpolations from initialization.

	$\Delta \downarrow$	Percent Monotonic ↑	Local Convexity ↑	Global Convexity ↑
Standard ResNet	$.079 \pm .109$	26.3%	$.548 \pm .139$	$.823 \pm .229$
σ-Asym ResNet	$.004 \pm .047$	87.3%	$.675 \pm .143$	$.976 \pm .098$
$\mathbf{W}$-Asym ResNet	$-\mathbf{.027} \pm .026$	**100%**	$\mathbf{.769} \pm .165$	$\mathbf{1.00} \pm .000$

When interpolating between the weights of a trained network and its initialization, we use some quantitative metrics to measure monotonicity and convexity of the trajectory. To measure monotonicity, we record the maximum increase between adjacent networks $\Delta = \max(L(\alpha_{i+1}) - L(\alpha_i))$, and the percentage of networks that have $\Delta \leq 0$ i.e. the percentage of networks that satisfy monotonic linear interpolation. To measure convexity, we consider a local convexity measure (the proportion of α_i where the centered difference second derivative approximation is nonnegative) and a global convexity measure (the proportion of α_i such that $L(\alpha_i)$ lies below the line segment between the endpoints, i.e. $L(\alpha_i) \leq (1 - \alpha_i)L(0) + \alpha_i L(1)$).

Table 2 shows the measures of monotonicity and convexity for standard, σ-Asymmetric, and $\mathbf{W}$-Asymmetric ResNets. Remarkably, every single one of the 300 $\mathbf{W}$-Asymmetric ResNets satisfies monotonic linear interpolation and has a trajectory that lies underneath the line segment between the endpoints.

C.3. Bayesian Neural Networks

Background. Bayesian deep learning is a promising approach to improve several deficits of mainstream deep learning methods, such as uncertainty quantification and integration of priors [29, 49]. However, parameter symmetries are problematic in Bayesian neural networks, as they are a major source of statistical nonidentifiability [29]. Parameter symmetries introduce modes in the posterior $p(\theta|\mathcal{D})$ that make the posterior harder to approximate [2, 33, 70], sample from [48, 69], and otherwise analyze [35]. For instance, one common technique for training Bayesian neural networks is variational inference via fitting a Gaussian distribution to the true posterior $p(\theta|\mathcal{D})$. This approach suffers because the Gaussian distribution has only one mode, whereas the true posterior has at least one mode for every parameter symmetry.

Table 3: Bayesian neural network results. All results (except for last column) are after 50 epochs of training. $\mathbf{W}$-Asymmetric networks tend to improve over their standard counterparts, especially early in training. 16-layer MLPs fail to train, but 16-layer $\mathbf{W}$-Asymmetric MLPs successfully train. Standard or Asymmetric networks better than their counterpart by a standard deviation are bolded.

	Model	Train Loss ↓	Test Loss ↓	ECE ↓	Test Acc ↑	Test Acc (25 Epochs) ↑
CIFAR-10	MLP-8	$1.34 \pm .00$	$1.24 \pm .01$	$.039 \pm .009$	$56.37 \pm .31$	52.87 ± 0.2
	$\mathbf{W}$-Asym MLP-8	$\mathbf{1.31} \pm .01$	$\mathbf{1.22} \pm .01$	$.042 \pm .009$	$\mathbf{57.08} \pm .50$	$\mathbf{54.15} \pm 0.2$
	MLP-16	$2.29 \pm .02$	$2.28 \pm .03$	$.026 \pm .017$	13.54 ± 2.0	13.34 ± 2.7
	$\mathbf{W}$-Asym MLP-16	$\mathbf{1.39} \pm .01$	$\mathbf{1.27} \pm .01$	$.045 \pm .009$	$\mathbf{55.16} \pm .44$	$\mathbf{51.42} \pm 0.3$
CIFAR-10	ResNet20	$\mathbf{.596} \pm .01$	$.535 \pm .03$	$.045 \pm .007$	81.98 ± 1.2	72.37 ± 1.0
	$\mathbf{W}$-Asym ResNet20	$.600 \pm .02$	$.535 \pm .01$	$.044 \pm .004$	81.94 ± 0.6	73.64 ± 1.5
	ResNet110	$.803 \pm .08$	$.706 \pm .08$	$.052 \pm .007$	75.71 ± 2.8	59.85 ± 3.9
	$\mathbf{W}$-Asym ResNet110	$\mathbf{.745} \pm .07$	$\mathbf{.658} \pm .06$	$.049 \pm .004$	77.40 ± 2.4	$\mathbf{63.20} \pm 3.0$
CIFAR-100	ResNet20 (BN)	$1.68 \pm .03$	$1.57 \pm .02$	$.078 \pm .004$	$56.83 \pm .62$	46.80 ± 0.9
	$\mathbf{W}$-Asym ResNet20 (BN)	$\mathbf{1.62} \pm .02$	$\mathbf{1.50} \pm .03$	$.076 \pm .006$	$\mathbf{58.40} \pm .62$	$\mathbf{49.29} \pm 0.4$
	ResNet20 (LN)	$1.97 \pm .02$	$1.88 \pm .02$	$.090 \pm .007$	$50.02 \pm .54$	37.24 ± 1.1
	$\mathbf{W}$-Asym ResNet20 (LN)	$\mathbf{1.91} \pm .03$	$\mathbf{1.82} \pm .02$	$.086 \pm .006$	$\mathbf{51.20} \pm .47$	$\mathbf{39.03} \pm 1.0$

Hypothesis. Using Asymmetric networks as the base model improves Bayesian neural networks, as the posterior will have less modes.

Experimental setup. We train Standard Bayesian and Asymmetric Bayesian Networks for image classification using variational inference. We use the method of [62] for variational inference, which fits a Gaussian approximate posterior with a diagonal plus low-rank covariance. We train 10 instances of each model and then report train loss, test loss, test accuracy, and Expected Calibration Error (ECE) [43], which is a measure of calibration.

Results. See training curves in Figure 4, and quantiative results in Table 3. Using $\mathbf{W}$-Asymmetric networks as a base for Bayesian deep learning improves training speed and convergence. Most strikingly, Bayesian MLPs of depth 16 cannot train at all, while $\mathbf{W}$-Asymmetric Bayesian MLPs train well. In general, the $\mathbf{W}$-Asymmetric approach improves training and test accuracy across the several models (MLPs, ResNets of varying sizes, and ResNets with either batch norm or layer norm).

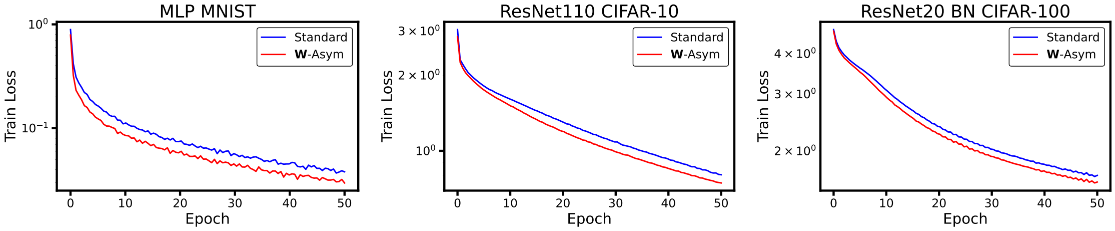

Figure 4: Bayesian neural network training loss over time for depth 8 MLPs on MNIST (left), ResNet110 on CIFAR-10 (middle), and ResNet20 with BatchNorm on CIFAR-100 (right). $\mathbf{W}$-Asymmetric networks train more quickly, and achieve lower training loss.

C.4. Metanetworks

Table 4: Metanetwork performance for predicting the test accuracy of small ResNets and our $\mathbf{W}$-Asym ResNets. Each row is a different metanetwork. Reported are R^2 and Kendall τ on the test set — higher is better.

	ResNet		$\mathbf{W}$-Asym ResNet	
	R^2	τ	R^2	τ
MLP	$-.171 \pm .11$	$.311 \pm .02$	$\mathbf{.594} \pm .12$	$\mathbf{.864} \pm .01$
DMC [11]	$.950 \pm .01$	$.787 \pm .02$	$\mathbf{.967} \pm .01$	$\mathbf{.911} \pm .01$
DeepSets [73]	$.855 \pm .01$	$.617 \pm .03$	$\mathbf{.936} \pm .00$	$\mathbf{.858} \pm .00$
StatNN [63]	$.976 \pm .00$	$.866 \pm .00$	$\mathbf{.978} \pm .00$	$\mathbf{.935} \pm .01$

Background. Metanetworks [38] — also referred to as deep weight-space networks [44, 56], metamodels [34], or neural functionals [77–79] — are neural networks that take as inputs the parameters of other neural networks. Recent work has found that making metanetworks invariant or equivariant to parameter-space symmetries of the input neural networks can substantially improve metanetwork performance [31, 38, 44, 77].

Hypothesis. Asymmetric networks are easier to train metanetworks on because they do not have to explicitly account for symmetries.

Experimental setup. We experiment with metanetworks on the task of predicting the CIFAR-10 test accuracy of an input image classifier, which many metanetworks have been tested on [11, 38, 63, 77]. We use metanetworks based on simple MLPs, 1D-CNN metanetworks [11], and metanetworks that are exactly invariant to permutation parameter symmetries: DeepSets [73] and StatNN [63]. We train two separate datasets of 10,000 image classifiers: one dataset of small ResNet models, and one dataset of $\mathbf{W}$-Asymmetric ResNet models. More information on the data, metanetworks, and training details are in Appendix G.3.

Results. In Table 4, we see that metanetworks are signficantly better at predicting the performance of our $\mathbf{W}$-Asymmetric ResNets than standard ResNets. Interestingly, simple MLP metanetworks,

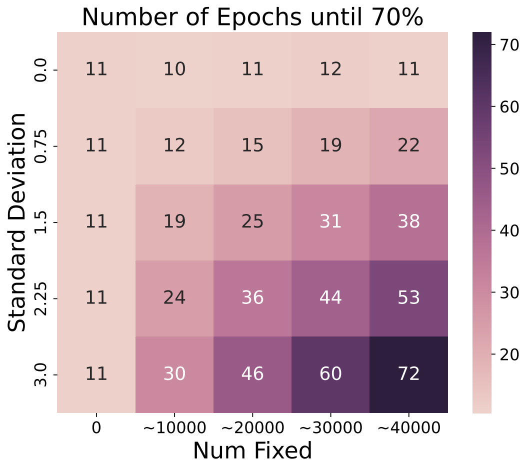

Figure 5: Epochs until reaching 70% training accuracy on CIFAR-10 when varying the hyperparameters of $\mathbf{W}$-Asymmetric ResNets; we vary number of fixed entries n_{fix} and standard deviation κ of the fixed entries $\mathbf{F}$. Entries further to the bottom and right are more asymmetric, while the entries further to the top and left are more like standard networks (the leftmost column are all standard networks). We see that more-asymmetric networks need more time to train.

which view the input parameters as a flattened vector, can predict the test accuracy of Asymmetric Networks quite well, but fail on standard networks. Also, the permutation equivariant metanetworks (DeepSets and StatNN) both improve on $\mathbf{W}$-Asym ResNets compared to on ResNets, even though the permutation symmetries of standard ResNets do not affect these metanetworks; thus, it may be possible that other symmetries in standard ResNets (but not Asym-ResNets) harm metanetwork performance, or they may be other factors besides symmetries that improve metanetwork performance for Asym-ResNets.

C.5. Additional Observations on Asymmetric Networks

There are several other interesting differences in the optimization and loss landscape properties of Asymmetric and standard neural networks. For one, even though Asymmetric networks generally interpolate significantly better than standard networks, this cannot be seen by measuring distances in parameter space. For instance, in GNN experiments following the setup of Section 4.1, pairs of standard GNNs have a distance per parameter of .000174 on average, whereas $\mathbf{W}$-Asymmetric GNNs have .000159, which is only slightly lower. However, the average test loss barrier is 1.448 for standard GNNs while it is only 0.069 for $\mathbf{W}$-Asymmetric GNNs. Likewise, in our datasets of 10,000 standard and $\mathbf{W}$-Asymmetric ResNets, the average distance per parameter between the weights of trained standard classifiers is .0034, which is actually lower than the distance per parameter of .0051 for $\mathbf{W}$-Asymmetric ResNets (estimated on 20,000 pairs of networks). Thus, although we sometimes imagine well-interpolating pairs of networks to lie in the same local basin of parameter space, $\mathbf{W}$-Asymmetric networks are actually rather far apart in parameter space, but nonetheless have linear segments of low loss between them.

We also find that Asymmetric networks often do not overfit as much as standard networks. For instance, in the GNN setup of Section 4.1, standard GNNs have a max training accuracy of 84.6% on average, with a validation accuracy of 71.6%. On the other hand, σ-Asym GNNs have $70.8\%/70.1\%$ train/validation accuracy, while $\mathbf{W}$-Asym GNNs have $70.7\%/70.06\%$ train/validation accuracy. This difference does not show as much in our datasets of 10,000 standard ResNets and $\mathbf{W}$-Asym ResNets, possibly because of the substantial regularization (data augmentation, weight decay, and label smoothing) used for training (standard gets $74.8\%/73.8\%$ train/test accuracy while $\mathbf{W}$-Asymmetric gets $64.0\%/64.0\%$).

Further, in Figure 5, we see that training speed is slower for $\mathbf{W}$-Asymmetric ResNets when we increase the amount of asymmetry (by increasing the number of fixed entries and the standard deviation of the fixed entries). While standard ResNets take on average 11 epochs to reach 70% training accuracy on CIFAR-10, $\mathbf{W}$-Asymmetric ResNets with the most extreme hyperparameters take up to 72 epochs.

Appendix D. Proofs of Theoretical Results

D.1. Graph-based approach

Here, we prove that as long as each mask matrix M in our $\mathbf{W}$-Asymmetric MLPs with fixed entries set to zero has unique nonzero rows, then our architecture has no nontrivial neural DAG automorphisms. In practice, we find that setting the standard deviation κ of the fixed entries $\mathbf{F}$ to be positive (and in fact orders of magnitude larger than the standard deviation that we typically initialize trainable weights with) is important to achieve properties such as linear mode connectivity that Asymmetric networks have but standard networks do not have. When $\kappa = 0$ (i.e. when fixed entries are set to zero), we can directly work in the framework of Lim et al. [38] that connects parameter symmetries to computation graph automorphisms. To work towards generalizing our result to $\kappa > 0$, we would have to modify the definitions and results of Lim et al. [38]; for instance, we would need to add edges associated to untrainable parameters in the computation graph, and redefine the concept of neural DAG automorphisms. We leave such exploration to future work.

Theorem 3 *If each mask matrix M has unique nonzero rows, then $\mathbf{W}$-Asymmetric MLPs with $\kappa = 0$ have no nontrivial neural DAG automorphisms.*

Proof Consider an L-layer $\mathbf{W}$-Asymmetric MLP with fixed entries set to zero. Denote its weights as $\mathbf{W}_L, \ldots, \mathbf{W}_1$ and the corresponding binary masks as $M_L, \ldots, M_1$. The forward pass of such a network on an input x is then

$$[\mathbf{W}_L \odot M_L]\sigma(\cdots\sigma([\mathbf{W}_1 \odot M_1]x)\cdots), \tag{5}$$

for some elementwise nonlinearity σ. The dimension of $\mathbf{W}_i$ is $d_i \times d_{i-1}$. In the framework of Lim et al. [38], this is a feedforward neural network with a computation graph defined as follows.

The node set is $V_0 \times V_1 \times \ldots \times V_L$, where V_i has d_i nodes, and no nodes are shared between different V_i. If a node v is in V_i, then we say that $\mathrm{layer}(v) = i$. V_0 contains the input nodes and V_L

contains the output nodes. The adjacency matrix can be written as

$$A = \begin{bmatrix} 0 & & & & \\ M_1 & 0 & & & \\ & M_2 & & & \\ & & \ddots & 0 & \\ & & & M_L & 0 \end{bmatrix}. \tag{6}$$

Every block besides the ones containing masks is zero. There are $L + 1 \times L + 1$ blocks, and the (i, j) block is of size $d_i \times d_j$.

Recall that a neural DAG automorphism is a relabelling of nodes $\tau : V \to V$ such that τ is bijective, $(i, j) \in E$ if and only if $(\tau(i), \tau(j)) \in E$, and every input node and output node is a fixed point of τ.

Now, let $\tau : V \to V$ be a neural DAG automorphism. Further, let P be the corresponding permutation matrix. We will show that τ is the identity, i.e. that $P = I$. By Lemma 4, we know that τ preserves layer number of nodes, meaning $\text{layer}(\tau(i)) = \text{layer}(i)$. Thus, P is a block diagonal permutation matrix:

$$P = \begin{bmatrix} P_0 & & & \\ & P_1 & & \\ & & \ddots & \\ & & & P_L \end{bmatrix}, \tag{7}$$

where P_i is $d_i \times d_i$. Morever, $P_0 = I$ and $P_L = I$ because input nodes and output nodes are fixed points. Applying this to the adjacency matrix, we see that

$$\tau(A) = PAP^\top = \begin{bmatrix} 0 & & & \\ P_1 M_1 P_0^\top & 0 & & \\ & & \ddots & \\ & & P_L M_L P_{L-1}^\top & 0 \end{bmatrix}. \tag{8}$$

Since τ is a neural DAG automorphism, we have that $\tau(A) = A$. Equating blocks, this means that $P_1 M_1 P_0^\top = M_1$. As $P_0 = I$, we have $P_1 M_1 = M_1$. But M_1 has unique rows, so $P_1 = I$ as well.

For the inductive step, assume $P_i = I$ for some i. Then $P_{i+1} M_{i+1} P_i^\top = P_{i+1} M_{i+1} = M_{i+1}$, so since M_{i+1} has unique rows, we have that $P_{i+1} = I$. As this holds for any i by induction, this means that $P = I$, so τ is a trivial neural DAG automorphism and we are done. ■

Lemma 4 *Neural DAG automorphisms preserve layer number in* $\mathbf{W}$*-Asymmetric MLPs that have masks with nonzero rows.*

Proof Let τ be a neural DAG automorphism. This means that $PAP^\top = A$, where P is the permutation matrix associated to τ. Then, using $PAP^\top = A$ for the first equality and the definition of P in the second, we have that

$$A_{\tau(i),\tau(j)} = (PAP^\top)_{\tau(i),\tau(j)} = A_{i,j}. \tag{9}$$

We proceed by induction on layer number l. For any input node i we know that $\tau(i) = i$, so of course $\text{layer}(\tau(i)) = \text{layer}(i)$.

Now, suppose that $\text{layer}(\tau(i)) = \text{layer}(i)$ for any i in layer $l \geq 1$. If node j is in layer $l+1$, then there is some i in layer l such that $(i, j) \in E$ because M_{l+1} has no nonzero rows. We have that $A_{\tau(i),\tau(j)} = A_{i,j}$, so $\tau(i)$ is connected to $\tau(j)$. As we know that $\tau(i)$ is in layer l, we have that $\tau(j)$ is in layer $l+1$. ■

D.2. Symmetry Breaking via Nonlinearities

Proposition 5 *Let the parameter space Θ be all pairs of square invertible matrices $\theta = (\mathbf{W}_2, \mathbf{W}_1)$ for $\mathbf{W}_2, \mathbf{W}_1 \in GL(d)$, and let $f_\theta(x) = \mathbf{W}_2\sigma(\mathbf{W}_1 x)$. If σ has no linear equivariances, then $f_{\theta_1} = f_{\theta_2}$ if and only if $\theta_1 = \theta_2$. In other words, there are no nontrivial parameter space symmetries.*

Proof If $\theta_1 = \theta_2$, then clearly $f_{\theta_1} = f_{\theta_2}$. For the other direction, suppose $f_{\theta_1} = f_{\theta_2}$, and denote $\theta_1 = (\mathbf{W}_2, \mathbf{W}_1)$ and $\theta_2 = (\widetilde{\mathbf{W}}_2, \widetilde{\mathbf{W}}_1)$. Then for any input $z \in \mathbb{R}^n$, we have

$$\mathbf{W}_2\sigma(\mathbf{W}_1 z) = \widetilde{\mathbf{W}}_2\sigma(\widetilde{\mathbf{W}}_1 z) \tag{10}$$

$$\widetilde{\mathbf{W}}_2^{-1}\mathbf{W}_2\sigma(\mathbf{W}_1 z) = \sigma(\widetilde{\mathbf{W}}_1 z). \tag{11}$$

Now, choose an arbitrary $x \in \mathbb{R}^n$. We let z in the above equation (11) be $\mathbf{W}_1^{-1}x$, so we have

$$\widetilde{\mathbf{W}}_2^{-1}\mathbf{W}_2\sigma(x) = \sigma(\widetilde{\mathbf{W}}_1\mathbf{W}_1^{-1}x). \tag{12}$$

This holds for any x, so $\widetilde{\mathbf{W}}_2^{-1}\mathbf{W}_2 \circ \sigma = \sigma \circ \widetilde{\mathbf{W}}_1\mathbf{W}_1^{-1}$, i.e. we have found a linear equivariance of σ. Since σ has no linear equivariances,

$$\widetilde{\mathbf{W}}_2^{-1}\mathbf{W}_2 = I = \widetilde{\mathbf{W}}_1\mathbf{W}_1^{-1}, \tag{13}$$

meaning that $\widetilde{\mathbf{W}}_2 = \mathbf{W}_2$ and $\widetilde{\mathbf{W}}_1 = \mathbf{W}_1$, i.e. $\theta_1 = \theta_2$, so we are done. ■

D.2.1. FiGLU NONLINEARITY PROOFS (PROPOSITION 2)

Now, we study the properties of our FiGLU nonlinearity $\sigma(x) = \eta(\mathbf{F}x) \odot x$, where η is the sigmoid function $\eta(x) = \frac{1}{1+e^{-x}}$. For proving Proposition 2, we want to prove that with probability 1 over samples of $\mathbf{F}$, σ has no permutation or diagonal equivariances.

We say that σ has *no permutation equivariances* if whenever $P_2 \circ \sigma = \sigma \circ P_1$ for permutation matrices P_1 and P_2, then $P_1 = P_2 = I$. Likewise, we say that σ has *no diagonal equivariances* if whenever $B \circ \sigma = \sigma \circ A$ for invertible diagonal matrices A and B, then $A = B = I$.

We will show that these two properties hold for any $\mathbf{F}$ that has no permutation symmetries and no zero entries. We say that $\mathbf{F}$ has *no permutation symmetries* if $P_2\mathbf{F}P_1 = \mathbf{F}$ for permutation matrices P_1 and P_2 implies that $P_1 = P_2 = I$. Note that if $\mathbf{F}$ has distinct entries, then it has no permutation symmetries. Thus, $\mathbf{F}$ satisfies both of these conditions with probability 1, since the set of matrices with nondistinct entries or with at least one zero entry are of Lebesgue measure zero, so they have zero probability under the Gaussian distribution. We now proceed to show that σ has no permutation or diagonal equivariances under these conditions on $\mathbf{F}$.

Proposition 6 *If $\mathbf{F}$ is a square matrix with no permutation symmetries, then $\sigma(x) = \eta(\mathbf{F}x) \odot x$ has no permutation equivariances.*

Proof Suppose $\sigma \circ P_1 = P_2 \circ \sigma$ for permutation matrices P_1, P_2. We will show that $P_1 = P_2 = I$. For any input x, we have

$$\eta(\mathbf{F}P_1 x) \odot P_1 x = P_2 \left[\eta(\mathbf{F}x) \odot x\right] \tag{14}$$

$$P_2^\top \left[\eta(\mathbf{F}P_1 x) \odot P_1 x\right] = \eta(\mathbf{F}x) \odot x \tag{15}$$

$$\eta(P_2^\top \mathbf{F} P_1 x) \odot P_2^\top P_1 x = \eta(\mathbf{F}x) \odot x, \tag{16}$$

where we used permutation equivariance of η, which acts elementwise. Let $x = e_i$, the standard basis vector that is 1 in the ith coordinate and 0 elsewhere. If i is not a fixed point of the permutation $P_2^\top P_1$, then let j be the index that it is mapped to. Then equation (16) gives that

$$\eta(P_2^\top \mathbf{F} P_1 e_i) \odot P_2^\top P_1 e_i = \eta(\mathbf{F}e_i) \odot e_i \tag{17}$$

$$\eta(P_2^\top \mathbf{F} P_1 e_i) \odot e_j = \eta(\mathbf{F}e_i) \odot e_i. \tag{18}$$

In the ith coordinate of this equality of vectors, we see that $\eta(\mathbf{F}e_i) = 0$, which is impossible, since η is the sigmoid function. Thus, i cannot be a fixed point of $P_2^\top P_1$, so $P_2^\top P_1 = I$ is the identity permutation. Now, let x be an arbitrary vector with no zero entries. Equation (16) gives that

$$\eta(P_2^\top \mathbf{F} P_1 x) \odot x = \eta(\mathbf{F}x) \odot x. \tag{19}$$

Since x is nonzero, we can divide by x_i in the ith coordinate of this vector equality for each i to get that

$$\eta(P_2^\top \mathbf{F} P_1 x) = \eta(\mathbf{F}x). \tag{20}$$

As η is bijective,

$$P_2^\top \mathbf{F} P_1 x = \mathbf{F}x. \tag{21}$$

Because this holds for all x with no zero entries (and in particular for a basis of the input space), we know that

$$P_2^\top \mathbf{F} P_1 = \mathbf{F} \tag{22}$$

as matrices. But since $\mathbf{F}$ has no permutation symmetries, we have that $P_1 = P_2 = I$, so we are done. ■

Proposition 7 *If $\mathbf{F}$ is a square matrix with no zero entries, then $\sigma(x) = \eta(\mathbf{F}x) \odot x$ has no diagonal equivariances.*

Proof Let $A = \mathrm{Diag}(\alpha)$ and $B = \mathrm{Diag}(\beta)$ be invertible diagonal matrices, and suppose that $\sigma \circ A = B \circ \sigma$. We will show that $A = B = I$. For any input x, we have

$$\eta(\mathbf{F}[\alpha \odot x]) \odot (\alpha \odot x) = \beta \odot \left[\eta(\mathbf{F}x) \odot x\right]. \tag{23}$$

Let $x = ce_i$, where e_i is the ith standard basis vector and $c \neq 0$ is any nonzero number. Then

$$\eta(\mathbf{F}c\alpha_i e_i) \odot c\alpha_i e_i = \beta \odot \left[\eta(c\mathbf{F}e_i) \odot ce_i\right]. \tag{24}$$

At the ith coordinate of this equality, we have

$$\eta(\mathbf{F}c\alpha_i e_i)_i c\alpha_i = \beta_i \eta(c\mathbf{F}e_i)_i c \tag{25}$$

$$\frac{\alpha_i}{\beta_i} = \frac{\eta(c\mathbf{F}e_i)_i}{\eta(\alpha_i c\mathbf{F}e_i)_i} \tag{26}$$

Thus, the right hand side is constant in c. We must have that $\alpha_i > 0$, because if not, then increasing c would increase either the numerator or denominator and decrease the other, hence contradicting the equality (here we use that $\mathbf{F}$ has no zero entries, so $c\mathbf{F}e_i$ is nonzero in every entry). Thus, letting $c \to \infty$, we see that $\frac{\alpha_i}{\beta_i} = 1$, so $\alpha_i = \beta_i$. Plugging this back into Equation (26), we have

$$1 = \frac{\eta(c\mathbf{F}e_i)_i}{\eta(\alpha_i c\mathbf{F}e_i)_i} \tag{27}$$

$$\eta(\alpha_i c\mathbf{F}e_i)_i = \eta(c\mathbf{F}e_i)_i \tag{28}$$

$$\alpha_i c(\mathbf{F}e_i)_i = c(\mathbf{F}e_i)_i \tag{29}$$

$$\alpha_i = 1, \tag{30}$$

where in the third line we used the fact that η is invertible. We have shown that $\alpha_i = \beta_i = 1$ for each i, so $A = B = I$ and we are done. ■

We note that the proofs of these two results about FiGLU are reminiscent of some proof techniques from Godfrey et al. [19], such as those used in their analysis of GELU nonlinearities.

D.3. Universal Approximation

Our two approaches remove parameter symmetries from standard neural networks, but still intuitively retain much of the structure of standard networks. One important property of widely-used neural network architectures is universal approximation — for any target function of a certain type, there exists a neural network of the given architecture that approximates the target to an arbitrary accuracy [7, 24, 41, 72]. Here, we show that $\mathbf{W}$-Asymmetric MLPs retain this property:

Theorem 8 *Let η be any nonpolynomial elementwise nonlinearity with $\eta(x) - \eta(-x) = x$ (e.g. $\mathrm{ReLU}, \mathrm{GELU}, \mathrm{swish}$), let $\Omega \subseteq \mathbb{R}^D$ be a compact domain, and let $f_{\text{target}} : \Omega \to \mathbb{R}$ be a continuous target function. Fix $\varepsilon > 0$ and $\delta > 0$.*

There exists a width n' such that for all $n > n'$, with probability $1 - \delta$, for a randomly sampled 4-layer $\mathbf{W}$-Asymmetric MLP f with η nonlinearity, hidden dimensions $24n \to n \to 24n$, and $n_{\text{fix}} \in o(n^{1/4})$ hardwired entries per neuron, there will exist $\theta \in \Theta$ such that the $\mathbf{W}$-Asymmetric MLP $f : \mathbb{R}^n \to \mathbb{R}$ approximates f_{target} to ε:

$$\left\| f_\theta([x; 0]) - f_{\text{target}}(x) \right\| < \varepsilon \text{ for all } x \in \Omega. \tag{31}$$

Importantly, we require that the input to f_θ be padded with $n - D$ zeroes, so $[x; 0] \in \mathbb{R}^n$.

D.3.1. PROOF SKETCH

To approximate f_{target} to $\varepsilon > 0$, we will leverage the universal approximation for standard MLPs with nonlinearity η to first obtain a standard 2-layer MLP that approximates f_{target} to within ε, meaning $\left\| f_{\text{target}}(x) - f_{\text{MLP}}(x) \right\| < \varepsilon$ for all $x \in \Omega$. Then we will exactly represent f_{MLP} using an Asymmetric Network f.

This will be done by approximating each linear map W of f_{MLP} by two layers of an Asymmetric network: $W_2' \circ \eta \circ W_1' = W$ for Asymmetric linear maps W_2' and W_1'. For the sake of exposition, we will show how to do this first when both W_2' and W_1' have no Asymmetric mask (i.e. fitting a linear map W using a standard two-layer η-MLP), then when only W_2' has an Asymmetric mask, and finally when both W_2' and W_1' have an Asymmetric mask.

D.3.2. FITTING A LINEAR MAP WITH A TWO-LAYER STANDARD MLP

Let $W \in \mathbb{R}^{n \times n}$ be the target linear map, and let $B \in \mathbb{R}^{2n \times n}$ and $A \in \mathbb{R}^{n \times 2n}$ be parameters of a two-layer MLP, defined by $f_{A,B}(x) = A\eta(Bx)$. We will choose A and B such that $f_{A,B}(x) = Wx$ for all $x \in \mathbb{R}^n$.

Denote the ith row of W by W_i, so that

$$W = \begin{bmatrix} W_0 \\ \vdots \\ W_{n-1} \end{bmatrix}, \qquad Wx = \begin{bmatrix} W_0 \cdot x \\ \vdots \\ W_{n-1} \cdot x \end{bmatrix} \tag{32}$$

We set A and B as follows, where I_n is the $n \times n$ identity matrix:

$$A = I_n \otimes \begin{bmatrix} 1 & -1 \end{bmatrix} = \begin{bmatrix} 1 & -1 & & & & \\ & & 1 & -1 & & \\ & & & \ddots & \ddots & \\ & & & & 1 & -1 \end{bmatrix} \qquad B = \begin{bmatrix} W_0 \\ -W_0 \\ W_1 \\ -W_1 \\ \vdots \\ W_{n-1} \\ -W_{n-1} \end{bmatrix}. \tag{33}$$

Then we can see that $f_{A,B}$ exactly computes the linear transformation Wx.

$$A\eta(Bx) = \begin{bmatrix} \eta(W_0 \cdot x) - \eta(-W_0 \cdot x) \\ \vdots \\ \eta(W_{n-1} \cdot x) - \eta(-W_{n-1} \cdot x) \end{bmatrix} = \begin{bmatrix} W_0 \cdot x \\ \vdots \\ W_{n-1} \cdot x \end{bmatrix} = Wx. \tag{34}$$

D.3.3. FITTING A LINEAR MAP WITH ONE ASYMMETRIC AND ONE STANDARD LINEAR MAP

Let $n_{\text{fix}} > 0$ and let each row of $\mathbf{N} \in \{0,1\}^{n \times 6n}$ have n_{fix} entries equal to 0, selected at random. Let $B \in \mathbb{R}^{6n \times n}$ and $A \in \mathbb{R}^{n \times 6n}$. Define A' to be an Asymmetric linear map: $A' = A \odot \mathbf{N} + (1 - \mathbf{N}) \odot \mathbf{P}$, where $\mathbf{N}$ is a randomly sampled binary mask, and $\mathbf{P}$ a randomly sampled Gaussian matrix. We consider a two-layer network with one Asymmetric and one standard linear map: $f_{A,B}(x) = A'\eta(Bx)$. We want $f_{A,B}(x) = Wx$ for all x. For the remainder of this proof, we will assume that $\mathbf{N}$ never has three consecutive entries in a row set to zero; we will later show that this holds with high probability over the sampling of $\mathbf{N}$.

First, we define B in a similar way to the purely linear setting, but with additional copies of entries to allow for error correction of the random noisy entries fixed in A'.

$$B = \begin{bmatrix} W_0 \\ W_0 \\ W_0 \\ -W_0 \\ -W_0 \\ -W_0 \\ \vdots \\ W_{n-1} \\ W_{n-1} \\ W_{n-1} \\ -W_{n-1} \\ -W_{n-1} \\ -W_{n-1} \end{bmatrix}. \tag{35}$$

Recall that each row A'_i of A' has n_{fix} entries that are randomly hardwired to constants. Ideally, we would want A'_i to pick out $\eta(W_i \cdot x) - \eta(-W_i \cdot x) = W_i \cdot x$, but because of the hardwired constants, A'_i might randomly add $c * \eta(W_j \cdot x)$. However, since there are three copies of $\eta(W_j \cdot x)$ in $\eta(Bx)$, as long as not all three corresponding entries in A'_i are fixed, one of the un-fixed copies can be changed such that the coefficients sum to 0. Since by our assumption $\mathbf{N}$ never has three consecutive entries all set to 0, the coefficients of A can be picked such that $A'\eta(Bx) = Wx$. For example, a possible A' matrix would be

$$A' = \left[\begin{array}{cccccc|cccccc|c|cccccc} 1 & 0 & 0 & -1 & 0 & 0 & \mathbf{1.1} & \mathbf{.37} & -1.47 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 & 0 & 0 \\ .89 & \mathbf{-.89} & 0 & 0 & 0 & 0 & \mathbf{.37} & .63 & 0 & -1 & 0 & 0 & \dots & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 1 & 0 & 0 & -1 & 0 & 0 \end{array}\right]$$

Thus we have shown that under the assumption that $\mathbf{N}$ never has three consecutive entries equal to 0, A can be picked such that $A'\eta(Bx) = Wx$. We will now show that

$$\mathbb{P}(\mathbf{N} \text{ never has three consecutive entries equal to } 0) \tag{36}$$

can be made arbitrarily small by increasing the width n while keeping n_{fix} to be $o(n^{1/3})$.

The probability there are three consecutive entries in a given row of $\mathbf{N}$ that are zero is $O(\frac{n_{\text{fix}}^3}{n^2})$. By the union bound, the probability that any row has 3 consecutive hardwired entries is $O(\frac{n_{\text{fix}}^3}{n})$. For any $n_{\text{fix}} \in o(n^{1/3})$, this tends towards 0. Thus, with probability $\geq 1 - O(\frac{n_{\text{fix}}^3}{n})$, A can be picked such that $A'\eta(Bx) = Wx$.

D.3.4. FITTING A LINEAR MAP WITH TWO ASYMMETRIC LINEAR MAPS

Once again let $n_{\text{fix}} > 0$, and let each row of $\mathbf{M}$ have n_{fix} entries equal to 0, selected at random. Let $B \in \mathbb{R}^{24n \times n}$ and $A \in \mathbb{R}^{n \times 24n}$. Further, define Asymmetric maps

$$A' = A \odot \mathbf{N} + (1 - \mathbf{N}) \odot \mathbf{P}, \qquad B' = B \odot \mathbf{M} + (1 - \mathbf{M}) \odot \mathbf{Q}, \tag{37}$$

where $\mathbf{N}, \mathbf{M}$ are randomly sampled masks, and $\mathbf{P}, \mathbf{Q}$ are normal matrices. Then we let $f_{A,B}(x) = A'\eta(B'(x))$, and we once again desire choices of parameters A and B such that $f_{A,B}(x) = Wx$.

Constructing B

Consider the randomly drawn mask $\mathbf{M} \in \{0,1\}^{24n \times n}$, and denote the ith row by M_i.

$$\mathbf{M} = \begin{bmatrix} M_0 \\ M_1 \\ \vdots \\ M_{24n-1} \end{bmatrix} \tag{38}$$

where $M_i \in \{0,1\}^n$. We partition $\mathbf{M}$'s rows into n blocks of 24 rows. $\beta_1 = \{M_0 \dots M_{23}\}, \dots \beta_i = \{M_{24i} \dots M_{24i+23}\}$. Now, consider β_1, the first 24 rows of $\mathbf{M}$.

Definition 9 *We say two rows M_j, M_k are **intersecting** if there is some column index α such that $M_{j,\alpha} = M_{k,\alpha} = 0$. That is, two rows of are intersecting if they share a 0 at the same index.*

Note that for any two given rows of M, the probability that they share a 0 in the same location is $\leq \frac{n_{\text{fix}}^2}{n}$.

We assume that β_i contains no more than one pair of intersecting rows; later, we show this to hold with high probability. Then, every β_i can be broken into two disjoint sets of 12 rows, $\beta_{i,0}$ and $\beta_{i,1}$, such that neither set of 12 contains a single pair of intersecting rows. Intuitively, this means that each row in B corresponding to $\beta_{i,0}$ will have unique fixed indices.

Our goal will be for the rows in β_i to mimic the row W_i. We will show how to do this for each i. Fix an arbitrary index $i \in \{0, \dots, n-1\}$.

Without loss of generality, assume $\beta_{i,0}$ and $\beta_{i,1}$ are continguous, so $\beta_{i,0} = M_{24i:24i+11}$ and $\beta_{i,1} = M_{24i+11:24i+23}$. By our assumption, for $j,k \in \beta_{i,0}$ (i.e. $j,k \in \{0,\dots,11\}$), the mask rows M_{24i+j} and M_{24i+k} are never 0 in the same two column indices. Similarly, for $j,k \in \beta_{i,1}$ (i.e. $j,k \in \{12,\dots,23\}$), the mask rows M_{24i+j} and M_{24i+k} are never 0 in the same two column indices.

Next, we define $c_{i,j}$ as the difference between B' and B in the $(24i+j)$th row:

$$c_{i,j} = B'_{24i+j} - B_{24i+j}. \tag{39}$$

In particular, we have that

$$c_{i,j} = -B_{24i+j} \odot (1 - M_{24i+j}) + (1 - M_{24i+j}) \odot Q_{24i+j}. \tag{40}$$

Lemma 10 *For any indices $j \neq k$ such that $j,k \in \beta_{i,0}$ or $j,k \in \beta_{i,1}$, we have that*

$$c_{i,j} \odot M_{24i+k} = c_{i,j}. \tag{41}$$

Proof By the definition of $c_{i,j}$, we know that $c_{i,j}$ is only nonzero at indices where M_{24i+j} is equal to zero. Since j,k are either both in $\beta_{i,0}$ or $\beta_{i,1}$, we know that M_{24i+k} cannot also be zero at indices where M_{24i+j} is zero. Thus, M_{24i+k} is equal to 1 at every index where $c_{i,j}$ is nonzero, so $c_{i,j} \odot M_{24i+k} = c_{i,j}$ as desired. ■

Next, we construct B, by constructing this block of 24 rows. Let $[\mathbf{c}_{i,0}, \mathbf{c}_{i,1}, \mathbf{c}_{i,2}, \mathbf{c}_{i,3}]$ be contiguous sums of length-3 segments of $c_{i,:}$:

$$\mathbf{c}_{i,0} = c_{i,0} + c_{i,1} + c_{i,2} \tag{42}$$
$$\mathbf{c}_{i,1} = c_{i,3} + c_{i,4} + c_{i,5} \tag{43}$$
$$\mathbf{c}_{i,2} = c_{i,6} + c_{i,7} + c_{i,8} \tag{44}$$
$$\mathbf{c}_{i,3} = c_{i,9} + c_{i,10} + c_{i,11} \tag{45}$$

We assign the first 12 rows of B as follows.

$$(0 \leq j < 3) \rightarrow B_{24i+j} = W_i + \mathbf{c}_{i,0} - \mathbf{c}_{i,1} + \mathbf{c}_{i,2} - \mathbf{c}_{i,3} - c_{ij} \tag{46}$$
$$(3 \leq j < 6) \rightarrow B_{24i+j} = -W_i - \mathbf{c}_{i,0} + \mathbf{c}_{i,1} - \mathbf{c}_{i,2} + \mathbf{c}_{i,3} - c_{ij} \tag{47}$$
$$(6 \leq j < 9) \rightarrow B_{24i+j} = +\mathbf{c}_{i,0} - \mathbf{c}_{i,1} + \mathbf{c}_{i,2} - \mathbf{c}_{i,3} - c_{ij} \tag{48}$$
$$(9 \leq j < 12) \rightarrow B_{24i+j} = -\mathbf{c}_{i,0} + \mathbf{c}_{i,1} - \mathbf{c}_{i,2} + \mathbf{c}_{i,3} - c_{ij} \tag{49}$$

Defining $\mathbf{c} = \mathbf{c}_{i,0} - \mathbf{c}_{i,1} + \mathbf{c}_{i,2} - \mathbf{c}_{i,3}$, we have the nice property:

$$(0 \leq j < 3) \rightarrow B'_{24i+j} = W_i + \mathbf{c} \tag{50}$$
$$(3 \leq j < 6) \rightarrow B'_{24i+j} = -W_i - \mathbf{c} \tag{51}$$
$$(6 \leq j < 9) \rightarrow B'_{24i+j} = +\mathbf{c} \tag{52}$$
$$(9 \leq j < 12) \rightarrow B'_{24i+j} = -\mathbf{c} \tag{53}$$

By the construction above, $B'_{24i} = -B'_{24i+3}$, and $B'_{24i+6} = -B'_{24i+9}$. This means that

$$\eta(B'_{24i} \cdot x) - \eta(B'_{24i+3} \cdot x) = B'_{24i} \cdot x = (W_i + \mathbf{c}) \cdot x \tag{54}$$

and likewise that

$$\eta(B'_{24i+6} \cdot x) - \eta(B'_{24i+9} \cdot x) = \mathbf{c} \cdot x \tag{55}$$

So that a simple linear map gives our desired output:

$$\begin{aligned} \eta(B'_{24i} \cdot x) - \eta(B'_{24i+3} \cdot x) - [\eta(B'_{24i+6} \cdot x) - \eta(B'_{24i+9} \cdot x)] &= (W_i + \mathbf{c} - \mathbf{c}) \cdot x & (56) \\ &= W_i \cdot x. & (57) \end{aligned}$$

In the next part, we will construct A' to compute this linear map, which will follow the method of Appendix D.3.3 (because A' has certain fixed entries).

What remains is to define the rows of B corresponding to $\beta_{i,1}$ in an error correctible manner. This can be done easily by defining

$$\mathbf{d} = \sum_{j=12}^{23} c_{ij} \tag{58}$$

and then defining

$$(12 \leq j < 24) \rightarrow B_{24i+j} = \mathbf{d} - c_{ij} \tag{59}$$

By similar reasoning to before, this means that

$$(12 \leq j < 24) \to B'_{24i+j} = \mathbf{d}. \tag{60}$$

Recall that we constructed B' under the assumption that no β_i has at most one pair of intersecting rows. We now show that the β_i each have at most one pair of intersecting rows with high probability. Within the 24 rows of any given β_i, the probability that more than one pair of rows are intersecting is $\leq C\frac{n_{\text{fix}}^4}{n^2}$ for some constant C. So, by the union bound, the probability over M that any of the β_i have more than one pair of intersecting rows is $\leq C\frac{n_{\text{fix}}^4}{n}$. Thus, we can construct B in this manner with probability $\geq 1 - C\frac{n_{\text{fix}}^4}{n}$. For sufficiently large n and $n_{\text{fix}} \in o(n^{1/4})$, this probability approaches 1.

CONSTRUCTION OF A

With our above construction, each block of the 24 rows in β_i of $\eta(B'x)$ is of the form

$$\begin{bmatrix} \eta((W_i + \mathbf{c}) \cdot x) \\ \eta((W_i + \mathbf{c}) \cdot x) \\ \eta((W_i + \mathbf{c}) \cdot x) \\ \eta((-W_i - \mathbf{c}) \cdot x) \\ \eta((-W_i - \mathbf{c}) \cdot x) \\ \eta((-W_i - \mathbf{c}) \cdot x) \\ \eta(\mathbf{c} \cdot x) \\ \eta(\mathbf{c} \cdot x) \\ \eta(\mathbf{c} \cdot x) \\ \eta(-\mathbf{c} \cdot x) \\ \eta(-\mathbf{c} \cdot x) \\ \eta(-\mathbf{c} \cdot x) \\ \eta(\mathbf{d} \cdot x) \\ \eta(\mathbf{d} \cdot x) \\ \eta(\mathbf{d} \cdot x) \\ \eta(\mathbf{d} \cdot x) \\ \eta(\mathbf{d} \cdot x) \\ \eta(\mathbf{d} \cdot x) \\ \eta(\mathbf{d} \cdot x) \\ \eta(\mathbf{d} \cdot x) \\ \eta(\mathbf{d} \cdot x) \\ \eta(\mathbf{d} \cdot x) \\ \eta(\mathbf{d} \cdot x) \\ \eta(\mathbf{d} \cdot x) \end{bmatrix} \tag{61}$$

Importantly, each row here is n wide. Recall that $A' \in \mathbb{R}^{n \times 24n}$. Denote the ith row of A' by $A'_i \in \mathbb{R}^{24n}$ with $A'_i \in \mathbb{R}^{24n}$. If A' had 0 hardwired entries, then setting $A_i = \mathbb{1}_{24i} - \mathbb{1}_{24i+3} - (\mathbb{1}_{24i+6} - \mathbb{1}_{24i+9})$ would give $A_i \eta(B' \cdot x) = W_i \cdot x$, by the same argument as in Appendix D.3.2.

Unfortunately this is not the case, so we have to use the construction in Appendix D.3.3. Recall, that A' has n_{fix} fixed entries in each row. This means that N_i has n_{fix} entries equal to 0. Since every entry of $B'x$ has three copies, as long as N_i does not have three elements set to 0 in a row, A'_i can be made equivalent to $A_i = \mathbb{1}_{24i} - \mathbb{1}_{24i+3} - (\mathbb{1}_{24i+6} - \mathbb{1}_{24i+9})$. This is because, as in Appendix D.3.3, if at most 2 elements out of any 3 three copies are hardwired, then the third can be changed arbitrarily to offset the hardwiring.

Further, just as in Appendix D.3.3, the probability that a given row of A' has three items hardwired in a row is $O(\frac{n_{\text{fix}}^3}{n^2})$. Thus, by the union bound, the probability that some row of A' has three items hardwired in a row is $O(\frac{n_{\text{fix}}^3}{n})$. So, with large enough width n, A can be chosen such that $A'\eta(B'x) = Wx$.

Similarly, any linear map in $\mathbb{R}^{\tilde{n} \times n}$ for $\tilde{n} < n$ can also be fit using this method.

Conclusion

We have shown that a $\mathbf{W}$-Asymmetric MLP with hidden dimension $24n$ can exactly fit an $n \times n$ linear map with high probability over the choice of Asymmetric masks M. It is known by [7] that for any continuous function $f_{\text{target}} : \Omega \subseteq \mathbb{R}^D \to \mathbb{R}$ and any $\varepsilon > 0$, there exists a width k' such that 2-layer MLPs of width k' can approximate f to within ε.

Let k be sufficiently big so that the probability that the masks do not satisfy the conditions of Appendix D.3.4 is less than δ. Such a k exists as long as $n_{\text{fix}} \in o(k^{\frac{1}{4}})$. Let $m \geq \max(k, k', D)$.

Importantly, if a 2-layer MLP of width k' can approximate f_{target} to within ε, a 2-layer MLP of width m with $m \geq k'$ can also approximate f to within ε. Let f_{MLP} be a width m MLP that approximates f_{target} to within ε.

We now pad the input x to f_{target}, with $m - D$ zeros. This allows us to define a new function $f^0_{\text{target}} : \mathbb{R}^m \to \mathbb{R}$ by $f^0_{\text{target}}([x; 0]) = f(x)$. Clearly f^0_{target} can also be approximated by a width m MLP.

Let f^0_{MLP} denote the width m MLP that approximates f_0 to within ε. Now, f^0_{MLP} has dimensions $m \to m \to 1$, with corresponding linear maps $W_1 \in \mathbb{R}^{m \times m}$, $W_2 \in \mathbb{R}^{1 \times m}$. Each of these maps can be exactly fit using a 2-layer $\mathbf{W}$-Asymmetric MLP, since their corresponding matrices have at least as many columns as rows. Concatenating these two exact fits yields an asymmetric MLP whose output exactly matches f^0_{MLP} and thus approximates f_0 to within ε.

Thus, setting $n' = m$, there exists a width n' such that for all $n > n'$, with probability $1 - \delta$, for a randomly sampled 4-layer $\mathbf{W}$-Asymmetric MLP f with η nonlinearity, hidden dimensions $24n \to n \to 24n$, and $n_{\text{fix}} \in o(n^{1/4})$ hardwired entries per neuron, there will exist $\theta \in \Theta$ such that the $\mathbf{W}$-Asymmetric MLP $f : \mathbb{R}^n \to \mathbb{R}$ approximates f_{target} to ε.

Appendix E. Limitations

Although our $\mathbf{W}$-Asymmetric and σ-Asymmetric networks are motivated by removing parameter space symmetries, their distinct empirical behavior may be caused by other factors besides just parameter space symmetries. For instance, the fixed entries $\mathbf{F}$ for the $\mathbf{W}$-Asymmetric approach are taken to be much larger than the standard initialization of linear maps, which could cause several changes to optimization and loss landscapes besides just parameter symmetry breaking.

Also, our theoretical results could be strengthened by future work in several ways. For instance, for the σ-Asymmetric approach, Proposition 5 only gives a guarantee of no parameter symmetries in the two-layer network case with square invertible weights. Future work could also give tighter analysis of the required width and depth for universal approximation using our $\mathbf{W}$-Asymmetric architecture.

Appendix F. Broader Impacts

This work does not focus on any particular application area. Instead, we study fundamental phenomena and theory of deep learning in general. Our work has potential to improve known deficits of neural networks: by making neural network loss landscapes more similar to convex landscapes, we can improve our understanding of them, and by improving Bayesian neural networks we advance one paradigm for bettering uncertainty quantification in neural networks. However, unlike standard neural networks, which have millions of papers studying them, we have only scratched the surface of Asymmetric networks. Important properties such as generalization, robustness to distribution shifts, and adversarial robustness have not been extensively studied for Asymmetric networks, and the interaction of parameter symmetries with these properties is not clear. Future research should further explore these important properties.

Appendix G. Experimental Details

G.1. Linear Mode Connectivity Experimental Details

G.1.1. Image Classifier Interpolation

For the image classification experiments, we use two types of models.

1. **ResNet** We train ResNet20s with LayerNorm of width 64 and $8 \cdot 64$. We use a batch size of 128 and a learning rate that warms up from .0001 to .01 over 20 epochs. In the width $8\times$ multiplier case we train for 50 epochs, and in the width $1\times$ multiplier case we train for 100. For σ-Asymmetric ResNets, we warm up to a learning rate of .001 instead of .01 due to training instability.

2. **MLP** We train MLPs with 4 layers, LayerNorm, and width 512. For MNIST we tuned the hyperparameters (epochs, learning rate, weight decay) of both the Asymmetric and Standard models to minimize loss barrier. We use a batch size of 64.

For MNIST we use no data augmentation, and for CIFAR-10 we use random cropping and horizontal flipping. For the Git-ReBasin tests, we use the weight matching algorithm from [1]. For MLPs on MNIST, we used the Asymmetry hyperparameters in Table 5. Table 6 gives the Asymmetric hyperparameters for ResNet20 on CIFAR-10, and Table 7 lists the same for ResNet20 with 8x larger width.

G.1.2. Graph Neural Network Interpolation

For the GNN experiments, we use a GNN architecture similar to GIN [71] with mean aggregation. The base GNN has three message passing layers and a hidden dimension of 256, which gives 176,424 trainable parameters. The dataset is ogbn-arXiv [26], which is a citation network of computer science arXiv papers with 169,343 nodes and 1,166,243 edges. The task is transductive node classification, where the label of each paper node is the primary subject area of the paper.

As is common in transductive node classification on modestly sized graphs, we train each network with full-batch gradient on the whole graph. Thus, the randomness in training is purely from the

Table 5: $\mathbf{W}$-Asymmetric network hyperparameters for depth 4 MLPs. n_{fix} refers to the number of weights we randomly fix per neuron. κ refers to the standard deviation of the normal distribution that the fixed entries $\mathbf{F}$ are drawn from.

Layer	n_{fix}	κ
Linear-1	64	1
Linear-2	64	1
Linear-3	64	$\frac{1}{2}$
Linear-4	256	$\frac{1}{4}$

Table 6: $\mathbf{W}$-Asymmetric network hyperparameters for ResNet20s with width multiplier 1. n_{fix} refers to the number of weights we randomly fix per output channel (for convolutional layers) or neuron (for linear layers). κ refers to the standard deviation of the normal distribution that the fixed entries $\mathbf{F}$ are drawn from.

Block	n_{fix}	κ
First Conv	12	2
Block 1 - Conv	36	2
Block 1 - Skip	4	2
Block 2 - Conv	54	2
Block 2 - Skip	6	2
Block 3 - Conv	72	2
Block 3 - Skip	8	2
Linear	8	2

Table 7: $\mathbf{W}$-Asymmetric network hyperparameters for ResNet20s with width multiplier 8 on CIFAR-10. We use 3 times more fixed entries per output channel or neuron than for Table 6.

Block	n_{fix}	κ
First Conv	27	2
Block 1 - Conv	108	2
Block 1 - Skip	12	2
Block 2 - Conv	162	2
Block 2 - Skip	18	2
Block 3 - Conv	216	2
Block 3 - Skip	24	2
Linear	24	2

initialization — there is no noise from minibatch selection in SGD. We use the Adam optimizer [30] with a peak learning rate of .001. The learning rate is linearly warmed up for 25 epochs to the peak, and then is held constant. Each network is trained for 500 epochs.

For the Git-ReBasin alignment, we implement the activation matching approach. For the σ-Asymmetric GNN, we take σ to be FiGLU, in which we randomly initialize each fixed matrix $\mathbf{F}$ as a standard normal matrix with standard deviation $.01/\sqrt{d}$ where d is the number of hidden channels; we found that having small standard deviation helped with training and interpolation. For the $\mathbf{W}$-Asymmetric GNN, we fix 6 constants in each row of each linear map, and randomly initialize these constants from a normal distribution with standard deviation $.5$.

G.2. Bayesian Neural Network Experimental Details

For training Bayesian neural networks, we use the variational inference approach of Tomczak et al. [62], which fits an approximate posterior that is Gaussian with a diagonal plus rank-4 covariance matrix structure. For the $\mathbf{W}$-Asym ResNet tests, we train ResNet20s with the same Asymmetric hyperparameters as in Table 6, though with $\kappa = .5$. For the CIFAR-100 experiments, we use a standard linear layer instead of hardwiring weights for the last fully-connected linear layer. On CIFAR-100 we also use a width multiplier of 2 for our ResNets. For the ResNet experiments, we use a learning rate of $.001$. We train with a batch size of 250 for 50 epochs.

For the MLP experiments, we use $\kappa = .5$, 8 hardwired entries per neuron, and a learning rate of $.0005$. A batch size of 250 is used for 50 epochs again.

We use standard data augmentation (horizontal flips and random crops) on CIFAR-10 and CIFAR-100, and no data augmentations for MNIST. All training is done with the Adam optimizer [30].

G.3. Metanetwork Experimental Details

G.3.1. Dataset Details

We trained two datasets of image classifiers on CIFAR-10: one consisting of 10,000 small ResNet-like convolutional neural networks, and one consisting of 10,000 networks with a similar architecture, that use our graph-based approach to removing parameter symmetries. For fast training of many image classifiers, we use the FFCV package [36]. In particular, we use their CIFAR-10 sample script `https://github.com/libffcv/ffcv/tree/main/examples/cifar`, which includes data augmentation (random horizontal flips, random translations, and Cutout [9]), label smoothing [60], and a linear learning rate warmup and decay. In total, training all 20,000 classifiers takes just under 400 GPU hours (about 2 GPU-weeks) on NVIDIA RTX 2080 Ti GPUs.

See Table 8 for the hyperparameters and ranges that we varied across the networks in our datasets. In each dataset, the trained networks all have the same architecture.

Each ResNet has 78,042 trainable parameters, and each $\mathbf{W}$-Asym ResNet has 60,634 trainable parameters. Both have the same architecture, except the $\mathbf{W}$-Asym ResNet has certain filters that are fixed to constants to break the parameter symmetries. The ResNets each have 8 convolution layers, LayerNorm [4], and a final fully-connected linear classification layer after average pooling across spatial dimensions.

G.3.2. Metanetwork Details

We trained several types of metanetworks for our experiments. All of these metanetworks are trained for 50 epochs using the AdamW optimizer [39]. For each metanetwork, on each dataset, we choose the learning rate in $\{10^{-5}, 10^{-4}, 5 \cdot 10^{-4}, 10^{-3}, 5 \cdot 10^{-3}, 10^{-2}\}$ that gives the best validation R^2 performance on one training run. Then we run train each type of metanetwork 5 times on each dataset, and report the mean and standard deviation for each metric in Table 4.

Table 8: Hyperparameters and distributions we sampled from for the datasets of image classifiers that we trained on CIFAR-10. $\mathrm{Unif}(a, b)$ is the uniform distribution over $[a, b]$, and $\mathrm{RandInt}(a, b)$ is the uniform distribution over integers in $[a, b]$ (inclusive of endpoints).

Hyperparameter	Distribution
Learning rate	$.5 \cdot 10^{-\mathrm{Unif}(0,2)}$
Weight decay	$10^{-\mathrm{Unif}(1,5)}$
Label smoothing	$\mathrm{Unif}(0, .2)$
Epochs	$\mathrm{RandInt}(10, 40)$

Table 9: Learning rate and number of parameters for each type of metanetwork trained in Table 4.

	ResNet		$\mathbf{W}$-Asym ResNet	
	LR	# Params	LR	# Params
MLP	10^{-4}	4,994,945	10^{-4}	3,880,833
DMC [11]	10^{-3}	105,357	$5 \cdot 10^{-3}$	105,357
DeepSets [73]	10^{-2}	8,897	$5 \cdot 10^{-3}$	8,897
StatNN [63]	10^{-3}	119,297	10^{-2}	119,297

G.4. Monotonic Linear Interpolation Experimental Details

For the monotonic linear interpolation experiments, we used the same setup as in the training of the datasets of CIFAR-10 image classifiers in Section C.4. For each architecture, we sample 300 sets of hyperparameters from the distributions in Table 8, and train one network for each set of these sampled hyperparameters. When evaluating training loss, we include the labeling smoothing term.

For the σ-Asymmetric networks, we initialize the FiGLU $\mathbf{F}$ with a standard deviation of $1/\sqrt{d}$, where d is the number of channels in the layer. Note that this is considerably larger than the standard deviation of $.01/\sqrt{d}$ used in the GNN experiments of Section 4.1; we found this setting to train better (note that this initialization is in line with standard initializations of trainable parameters). Further, for the σ-Asymmetric networks, 24 out of the 300 networks diverged during training (giving `NaNs`), so we exclude them from the computation of statistics in Table 2. From manual inspection, this divergence seems to happen when the learning rate is high (greater than .1). In contrast, none of the standard or $\mathbf{W}$-Asymmetric networks diverged.

G.5. Miscellaneous Experimental Details

The datasets we use are MNIST [37], CIFAR-10 [32], CIFAR-100 [32], and ogbn-arXiv [26], which are all widely used in machine learning research. The first three appear to not have licenses and are open to use, while the last dataset is from the Open Graph Benchmark, which has an MIT License in the Github repository.

We use software packages including PyTorch [50] (for all neural network experiments), FFCV [36] (for building our dataset in Section C.4), and PyTorch Geometric [14] (for GNN experiments).

We ran our experiments on several types of NVIDIA GPUs and compute systems, including 2080 Ti, 3090 Ti, 4090 Ti, and V100 GPUs. Every training run was conducted on at most one GPU.