Extended Abstract Track

Data Augmentations in Deep Weight Spaces

Aviv Shamsian[*1] **David W. Zhang**[*3] **Aviv Navon**[1] **Yan Zhang**[4] **Miltiadis Kofinas**[3]
Idan Achituve[1] **Riccardo Valperga**[3] **Gertjan J. Burghouts**[5] **Efstratios Gavves**[3]
Cees G. M. Snoek[3] **Ethan Fetaya**[1] **Gal Chechik**[1,6] **Haggai Maron**[2,6]
Bar Ilan University[1] *Technion*[2] *University of Amsterdam*[3] *Samsung - SAIT AI Lab, Montreal*[4] *TNO*[5] *NVIDIA*[6]

Abstract

Learning in weight spaces, where neural networks process the weights of other deep neural networks, has emerged as a promising research direction with applications in various fields, from analyzing and editing neural fields and implicit neural representations, to network pruning and quantization. Recent works designed architectures for effective learning in that space, which takes into account its unique, permutation-equivariant, structure. Unfortunately, so far these architectures suffer from severe overfitting and were shown to benefit from large datasets. This poses a significant challenge because generating data for this learning setup is laborious and time-consuming since each data sample is a full set of network weights that has to be trained. In this paper, we address this difficulty by investigating data augmentations for weight spaces, a set of techniques that enable generating new data examples *on the fly* without having to train additional input weight space elements. We first review several recently proposed data augmentation schemes and divide them into categories. We then introduce a novel augmentation scheme based on the Mixup method. We evaluate the performance of these techniques on existing benchmarks as well as new benchmarks we generate, which can be valuable for future studies.

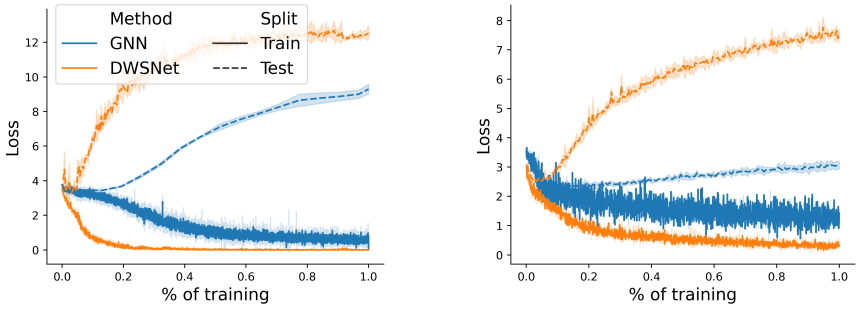
1. Introduction

Learning in deep weight spaces is the problem of training neural networks for processing the weights of other deep neural networks – a problem that has recently gathered significant interest (Eilertsen et al., 2020; Unterthiner et al., 2020; Andreis et al., 2023). The focus in recent work has largely been on the development of novel architectures that take into consideration the permutation symmetry of neurons (Navon et al., 2023; Zhou et al., 2023a,b; Zhang et al., 2023). These architectures have demonstrated promising results on multiple benchmarks, significantly outperforming earlier naïve approaches.

However, there is still a significant gap between the results obtained by such equivariant networks operating on Implicit Neural Representations (INRs) and those obtained by applying standard deep models such as CNNs and MLPs, which take as input the original image (or other raw signal) representation. For example, current state-of-the-art models for processing INRs achieve only 16% accuracy on the ModelNet40 3D shape classification benchmark, while neural networks operating on the same data, but represented using a point cloud, achieve over 90% accuracy (Atzmon et al., 2018; Wang et al., 2019).

To understand this problem, we first quantify the generalization error when learning with INRs: Figure 1 demonstrates that existing architectures suffer from severe overfitting

[*] Equal contribution



(1) Training with 1 view per INR (2) Training with 10 views per INR

Figure 1: *Overfitting of weight space architectures on the ModelNet40 dataset*: We visualize train and test losses for DWSNets (Navon et al., 2023) and graph-based architectures (Zhang et al., 2023) on the ModelNet40 datasets with 1 or 10 trained input networks per point cloud (views). Notably, both methods tend to overfit early during training, even when using more data.

(see details in Section 2). A possible explanation for this finding is that previous equivariant architectures account for the permutation symmetries in the weight space, but remain sensitive to other types of variability present in weight spaces. These include, for example, scaling transformations, weight perturbations, and more. While it is possible to bridge the generalization gap by collecting more data, this is challenging when learning in weight spaces, where generating any data sample requires training a deep neural network.

To alleviate this problem and effectively increase the number of training examples without generating more data, we provide the first study of data augmentation schemes for weight spaces: we study simple transformations that can be applied to input samples (weight space elements) to achieve more diversity while preserving the functions represented by those weights. Data augmentation schemes are widely used and heavily studied for common data types like images (Shorten and Khoshgoftaar, 2019). For weight spaces, augmentations are challenging and unexplored, in part, due to their symmetry structure.

We first propose a taxonomy of known weight space augmentation schemes. (i) *input-space augmentation*, transformations of weight space elements that reflect simple transformations in the input space, like rotating a 3D object by a linear transformation of its 3D INR; (ii) *generic augmentations*, like adding noise, feature masking or dense feature Mixup (applying Mixup to the representation in the penultimate layer). See also (Zhang et al., 2017; Zhou et al., 2023b). In addition, we propose (iii) *augmentations inspired by activation functions*, a novel family of augmentations that exploit activation symmetries.

We then develop *Weight-Space Mixup*, a novel data augmentation scheme based on generalizing the Mixup method (Zhang et al., 2017) to weight spaces. Unlike mixup for dense vectors and images, applying mixup directly to weight space elements is challenging. A crucial reason for that is that due to the permutation symmetries of weight spaces, the weights of two independently trained models are rarely aligned, and directly averaging them may not yield an appropriate model (Ainsworth et al., 2022). We address this difficulty and develop several variants of weight space mixup, building on recent works in weight-space alignment algorithms (Ainsworth et al., 2022; Peña et al., 2023).

Our results indicate that data augmentation schemes, and specifically our proposed Weight-Space Mixup method, can enhance the accuracy of weight space architectures by up

to 18%. This improvement is equivalent to reducing the required amount of generated data by almost 10 , which saves countless hours of computation and electricity consumption. We also contribute two new benchmarks on both image and 3D modalities.

2. Overfitting in deep weight spaces

In this section, we show that Weight space networks, which operate on INRs, significantly underperform compared to their counterparts that operate on the original data space (e.g., point clouds or images). We attribute this performance gap primarily to an overfitting problem in deep weight spaces. In order to validate this, we train two different weight space models DWSNet (Navon et al., 2023) and GNN (Zhang et al., 2023) on ModelNet40 INR datasets with 1 or 10 views per object, where by views we mean differently initialized INRs that fit the same object in the original dataset. In Figure 1 (left panel), we observe that when training with a single view all runs start overfitting in the early stages of the training process at around 5 10% of the total update steps. Training with 10 views per object (Figure 1 right panel) somewhat alleviates the overfitting problem and can be seen as a type of data augmentation. We note that although the gap between the train and test error becomes smaller, the overfitting problem still remains. Furthermore, generating 10 views per object requires substantial computing time. Next, we consider alternative augmentation methods that can be applied directly to weight space elements.

3. A taxonomy of augmentations for weight-space elements

We present three families of augmentation schemes for weight spaces, and use these schemes to categorize previously proposed augmentation schemes.

Input-space augmentations. Data augmentations like random rotations, translations, and scalings are frequently used when learning image and 3D data. As shown in Navon et al. (2023), in many cases, these augmentations can be applied to INRs by applying the relevant geometric transformations to the input coordinates of the INR. As an example, rotating the object represented by an INR by a random rotation R can be accomplished by replacing W_1, the first weight matrix of the INR, with $W_1 R$.

Generic augmentations. General data augmentation techniques are augmentation techniques that can be applied to any type of data. This category includes several methods such as dropout (Srivastava et al., 2014), which randomly deactivates a fraction of weights during training, quantile dropout, which removes weights with magnitudes below a defined threshold, and the addition of random Gaussian noise to the input weights.

Augmentations inspired by activation functions. In many cases, activation functions induce symmetries that are not easy to incorporate into the weight space architecture. We propose three activation space augmentations that exploit this symmetry. For ReLU activation, we can arbitrarily scale the weights $\frac{1}{c}W_{i+1}\mathrm{ReLU}(cW_i x+cb_i)+b_{i+1} = W_{i+1}\mathrm{ReLU}(W_i x+b_i)+b_{i+1}$ with some $c2\mathrm{R}^+$. In SIREN (Sitzmann et al., 2020), the sinusoidal activation function induces two additional symmetries. First, since the function is odd we can negate the weight and biases of layer i and the weight of the following layer $i+1$ as $W_{i+1}\mathrm{Sine}(W_i x+b) = W_{i+1}\mathrm{Sine}(W_i x b)$. The second symmetry results from the

shift of the phase in an even or odd multiple of , more formally: $W_{i+1}\,\text{Sine}(W_i x + b) = (-1)^k W_{i+1}\,\text{Sine}(W_i x + b + k\,)$. for $k \in \mathbb{Z}$. We incorporate these symmetries through random data augmentations and refer to them as*SIREN negation* and *SIREN bias* respectively.

4. Mixup in weight space

Mixup (Zhang et al., 2017) is a popular data augmentation technique where the basic idea is to randomly interpolate a pair of input images x_1, x_2 and their ground truth labels y_1, y_2 to create a new training example $(\lambda x_1 + (1-\lambda)x_2,\ \lambda y_1 + (1-\lambda)y_2)$. In the last few years, Mixup was successfully generalized to several data types such as point clouds and graphs (Chen et al., 2020; Achituve et al., 2021; Han et al., 2022).

Alignment and interpolation in weight spaces. To design a mixup method for wight spaces, we first need to understand the weight space alignment problem: given two weight space elements $x_1 = [W_1^{(l)}, b_1^{(l)}]$ and $x_2 = [W_2^{(l)}, b_2^{(l)}]$, $l = 1, \ldots, M$, this problem seeks a sequence of permutations $p = (P_1, \ldots, P_{M-1})$ that minimizes $\|x_1 - p(x_2)\|$, where $p(x_2)$ applies the permutations to the weight vectors without changing the underlying function, as in Equation 5 in Navon et al. (2023). Intuitively, this problem seeks permutations such that the weights of these networks are as close as possible when compared directly. Several recent works (Entezari et al., 2022; Ainsworth et al., 2022; Peña et al., 2023; Navon et al., 2023) have shown that the interpolation between a weight vector x, to the optimally permuted version of the other vector $p(x')$ has a property called*linear mode connectivity*, which states that the loss value on this path is only marginally worse compared to its endpoints. This is in contrast to weights obtained from the direct interpolation between x, x' which produces a signi cant increase in this loss.

Weight-space mixup. The naive (standard) weight-space mixup is formally de ned as an interpolation between two weight space samples $[W_1^{(l)}, b_1^{(l)}]$ and $[W_2^{(l)}, b_2^{(l)}]$ with $\lambda \sim U(0,1)$: $W^{(l)} = \lambda W_1^{(l)} + (1-\lambda) W_2^{(l)}$; $b^{(l)} = \lambda b_1^{(l)} + (1-\lambda) b_2^{(l)}$, where the weight parameter is randomly drawn from a uniform distribution.

Next, we de ne the randomized weight space mixup in which random permutations are applied to one of the input weights before mixing two samples. While the weights, in this case, are still not aligned (with high probability), we do get a much greater degree of diversity than we would obtain with the standard approach.

Lastly, we de ne matching based weight space mixup where we use a sequence of permutation matrices p to first align the weights and then perform the interpolation. As the weight space alignment problem is NP-hard, we obtain an approximate alignment using the Weight Matching algorithm suggested by (Ainsworth et al., 2022).

5. Experiments

We evaluate various weight-space augmentations for classifying INRs. Speci cally, we create INR datasets for ModelNet40 (3D point clouds) and FMNIST (2D greyscale images). We generate 10 di erent INRs { referred to as 10 di erent views { for each example in the original dataset. We compare each augmentation individually for 1 and 10 views and use DWS (Navon et al., 2023) and GNN (Zhang et al., 2023) as weight-space architectures. We report the average accuracy and standard deviations for 3 random seeds. More details on the experimental setup and data generation processes are in Appendix C, B.

Table 1: ModelNet40 and FMNIST results: test accuracy results for 1 and 10 views.

Augmentation type	Model	ModelNet40				FMNIST			
		1 View		10 View		1 View		10 View	
No augmentation	DWS	16 :17	0:25	30:25	0:95	68:30	0:62	76:01	1:20
No augmentation	GNN	8 :82	1:08	34:51	1:24	68:84	0:41	79:58	3:01
Translate	DWS	18 :18	0:97	31:17	0:02	67:90	0:24	77:61	0:36
Rotation	DWS	|		|		68 :55	0:28	77:04	0:47
Scale	DWS	16:41	0:57	30:54	0:72	67:99	0:14	75:77	1:09
Gaussian noise	DWS	14:10	0:71	25:31	1:78	68:53	0:09	77:60	0:13
SIREN bias	DWS	4 :69	0:10	4:90	0:01	58:20	0:01	62:21	0:55
SIREN negation	DWS	20 :14	0:98	32:31	0:70	71:40	0:29	77:71	1:38
Dropout	DWS	11 :43	2:44	14:71	1:14	68:48	0:14	75:57	1:91
Quantile dropout	DWS	15 :13	2:45	29:88	0:62	68:72	0:27	76:22	0:72
Translate	GNN	8 :17	0:81	34:93	1:31	70:17	1:26	83 :83	0:25
Rotation	GNN	|		|		69 :35	2:18	83:72	1:14
Scale	GNN	8:58	0:65	34:70	5:19	68:96	1:46	83:67	0:19
Gaussian noise	GNN	9:06	0:27	32:82	1:14	77:55	0:33	81:28	0:50
SIREN bias	GNN	11 :63	2:48	34:32	1:57	68:09	0:49	77:20	1:03
SIREN negation	GNN	11 :41	3:22	37:93	2:26	72:74	4:29	82:36	3:66
Dropout	GNN	8 :10	0:43	18:04	1:24	68:55	1:21	79:72	1:35
Quantile dropout	GNN	8 :12	0:85	34:36	1:14	69:96	2:08	83:78	0:76
MixUp	DWS	26 :96	0:91	31:92	0:37	74:36	1:17	78:58	0:20
MixUp + random perm.	DWS	26 :62	0:18	33:55	1:40	73:89	0:89	78:04	1:02
Alignment + MixUp	DWS	27:40	0:97	33:33	0:43	75:67	0:36	79 :41	0:56
MixUp	GNN	20 :45	3:82	42:25	3:83	80:18	0:59	82:20	0:52
MixUp + random perm.	GNN	24 :46	2:92	41:67	4:55	78:45	2:29	82:24	0:68
Alignment + MixUp	GNN	26:88	1:75	42 :83	4:18	78:80	2:12	82:94	0:31

Table 1 showcases the e ectiveness of on-the- y weight space data augmentation schemes. Notably, Mixup augmentations with a single view are comparable to training with 10 more data on both datasets: ModelNet40 and FMNIST. Furthermore, data augmentation is still e ective with 10 views. Augmentations applied in the input space, which are limited to the rst layer of the INR, are less e ective compared with other types of augmentations that modify the weights in all the layers. Overall, the e ectiveness of input-space and generic augmentations varies between the models and also between the datasets. In contrast, Weight Space Mixup provides consistent improvements, with the alignment-based version frequently outperforming other variants.

6. Conclusion

This paper examines the over tting issue associated with weight space architectures and proposes novel weight space augmentation techniques that mitigate this issue and enhance model performance. Notably, our experiments demonstrate that training with these augmentations has comparable results to training with substantially larger datasets.

Limitations. It is important to note that weight space augmentations may vary in e ectiveness across di erent datasets and tasks, which requires further investigation. In addition, some augmentations, such as Mixup with alignment, may require additional computational overhead that may be prohibitive in some resource-constrained environments.

Acknowledgements. 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).

References

Idan Achituve, Haggai Maron, and Gal Chechik. Self-supervised learning for domain adaptation on point clouds. In Proceedings of the IEEE/CVF winter conference on applications of computer vision, pages 123–133, 2021.

Samuel K Ainsworth, Jonathan Hayase, and Siddhartha Srinivasa. Git re-basin: Merging models modulo permutation symmetries. arXiv preprint arXiv:2209.04836, 2022.

Bruno Andreis, Soro Bedionita, and Sung Ju Hwang. Set-based neural network encoding. arXiv preprint arXiv:2305.16625, 2023.

Matan Atzmon, Haggai Maron, and Yaron Lipman. Point convolutional neural networks by extension operators. arXiv preprint arXiv:1803.10091, 2018.

Chengtai Cao, Fan Zhou, Yurou Dai, and Jianping Wang. A survey of mix-based data augmentation: Taxonomy, methods, applications, and explainability. arXiv preprint arXiv:2212.10888, 2022.

Yunlu Chen, Vincent Tao Hu, Efstratios Gavves, Thomas Mensink, Pascal Mettes, Pengwan Yang, and Cees GM Snoek. Pointmixup: Augmentation for point clouds. In Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16, pages 330–345. Springer, 2020.

Emilien Dupont, Hyunjik Kim, S. M. Ali Eslami, Danilo Jimenez Rezende, and Dan Rosenbaum. From data to functa: Your data point is a function and you can treat it like one. In International Conference on Machine Learning, 2022. URL https://api.semanticscholar.org/CorpusID:249395684 .

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.

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 .

Xiaotian Han, Zhimeng Jiang, Ninghao Liu, and Xia Hu. G-mixup: Graph data augmentation for graph classification. In International Conference on Machine Learning, pages 8230–8248. PMLR, 2022.

Dan Hendrycks, Norman Mu, Ekin D Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. Augmix: A simple data processing method to improve robustness and uncertainty. arXiv preprint arXiv:1912.02781, 2019.

Hongyi Ling, Zhimeng Jiang, Meng Liu, Shuiwang Ji, and Na Zou. Graph mixup with soft alignments. arXiv preprint arXiv:2306.06788, 2023.

Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam. ArXiv, abs/1711.05101, 2017. URL https://api.semanticscholar.org/CorpusID:3312944.

Aviv Navon, Aviv Shamsian, Idan Achituve, Ethan Fetaya, Gal Chechik, and Haggai Maron. Equivariant architectures for learning in deep weight spaces. arXiv preprint arXiv:2301.12780, 2023.

Fidel A Guerrero Pena, Heitor Rapela Medeiros, Thomas Dubail, Masih Aminbeidokhti, Eric Granger, and Marco Pedersoli. Re-basin via implicit sinkhorn di erentiation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 20237{20246, 2023.

Konstantin Schurholt, Diyar Taskiran, Boris Knyazev, Xavier Gir'o i Nieto, and Damian Borth. Model zoos: A dataset of diverse populations of neural network models. ArXiv, abs/2209.14764, 2022. URL https://api.semanticscholar.org/CorpusID:252595733.

Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. Journal of big data, 6(1):1{48, 2019.

Vincent Sitzmann, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. ArXiv, abs/2006.09661, 2020. URL https://api.semanticscholar.org/CorpusID:219720931.

Nitish Srivastava, Geo rey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from over tting. The journal of machine learning research, 15(1):1929{1958, 2014.

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

Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. ACM Transactions on Graphics (tog), 38(5):1{12, 2019.

Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 1912{1920, 2015.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. ArXiv, abs/1708.07747, 2017. URL https://api.semanticscholar.org/CorpusID:702279.

Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classi ers with localizable features. In Proceedings of the IEEE/CVF international conference on computer vision, pages 6023{6032, 2019.

David W Zhang, Miltiadis Ko nas, Yan Zhang, Yunlu Chen, Gertjan J Burghouts, and Cees GM Snoek. Neural networks are graphs! graph neural networks for equivariant processing of neural networks. 2nd Annual Topology, Algebra, and Geometry in Machine Learning Workshop at ICML, 2023.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. arXiv preprint arXiv:1710.09412, 2017.

Allan Zhou, Kaien Yang, Kaylee Burns, Yiding Jiang, Samuel Sokota, J Zico Kolter, and Chelsea Finn. Permutation equivariant neural functionals. arXiv preprint arXiv:2302.14040, 2023a.

Allan Zhou, Kaien Yang, Yiding Jiang, Kaylee Burns, Winnie Xu, Samuel Sokota, J Zico Kolter, and Chelsea Finn. Neural functional transformers. arXiv preprint arXiv:2305.13546, 2023b.

Appendix A. Previous work

Learning in deep weight spaces Recently there has been a growing interest in applying deep learning architectures directly to neural network weights. The attention to this domain was brought up by studies that presented weight-space related datasets (Dupont et al., 2022; Schürholt et al., 2022). Others (Sitzmann et al., 2020) showed the possibility of representing a data point as an implicit neural representation on several tasks from classification to generation. These datasets raised the motivation to study how to learn in deep weight spaces. Early methods proposed using simple architectures such as MLPs and transformers to predict test errors or the hyperparameters that were used for training input networks (Eilertsen et al., 2020; Unterthiner et al., 2020). Recently, Navon et al. (2023) presented the first neural architecture that accounts for natural permutation symmetries of weight spaces and demonstrated significant performance improvements over prior methods. Zhou et al. (2023a) proposed a similar approach, which was later enhanced by the addition of attention mechanisms (Zhou et al., 2023b). Finally, Zhang et al. (2023) proposed a GNN architecture to process neural networks modeled as computational graphs.

Data augmentation in deep learning Data augmentation is an essential technique in deep learning that plays a crucial role in mitigating overfitting while enhancing the generalization capabilities of NNs. Data augmentation helps the model learn more robust and invariant features. Various techniques are employed to achieve this, such as geometric transformations like rotation, scaling, and translation which make the model robust to changes in object orientation and position. Additionally, color-based augmentations like brightness adjustments, contrast changes, and color jittering contribute to improved generalization by increasing the model's tolerance to variations in lighting conditions. Dropout (Srivastava et al., 2014) is another prominent data augmentation approach that randomly deactivates weights during training. These methods collectively enhance the model's ability to generalize from limited training data and reduce the risk of overfitting, resulting in more robust and accurate deep learning models.

Mixup Mixup is a data augmentation method that blends two or more training samples to create new synthetic instances (Cao et al., 2022). Mixup (Zhang et al., 2017) operates by taking a weighted linear combination of two input samples, where both the input data and their corresponding labels are mixed. The resulting mixed data point contains characteristics of both original samples, effectively generating a smooth interpolation between them. Various mixup variants have been proposed, including CutMix (Yun et al., 2019), which combines two images by cutting and pasting rectangular regions; and AugMix (Hendrycks et al., 2019), which applies multiple augmentation operations before mixing to further diversify the dataset. Recently, several works (Ling et al., 2023) proposed performing Mixup after first aligning the feature or input space, resulting in smoother interpolation between the mixed objects.

Appendix B. Datasets

The increasing usage of INRs in many machine-learning domains, specifically in images and 3D objects, raises the need for INR benchmarks. Implicit representations, such as neural radiance fields and neural implicit surfaces, offer a more flexible and expressive way to

model complex 3D scenes and objects. However, as these techniques gain traction, it becomes crucial to establish standardized benchmarks to assess and compare the performance of architectures designed for weight space data. To address this issue, we present new INR classification benchmarks based on ModelNet40 (Wu et al., 2015) and Fashion-MNIST (Xiao et al., 2017) datasets. We use the SIREN (Sitzmann et al., 2020) architecture, i.e. MLP with sine activation, and fit each data point in the original dataset. To negate the possibility of canonical representation that may lead to globally aligned data representation, we randomly initialize the weights for every generated INR. In the case of ModelNet40, INRs are generated through training an MLP to accurately predict the signed distance function values of a 3D object given a set of 3D point clouds. For Fashion-MNIST an MLP is trained to map from the 2D xy-grid to the corresponding gray level value in the original image. We fit 10 unique INRs, namely views, per sample in the original dataset resulting in a total of $123K$ and $700K$ INRs for ModelNet40 and Fashion-MNIST respectively.

Appendix C. Experimental Details

DWS. In all experiments, we use DWS (Navon et al., 2023) network with 4 hidden layers and hidden dimension of 128. We optimized the network using a $5e$ 3 learning rate with AdamW (Loshchilov and Hutter, 2017) optimizer.

GNN. For the GNN, we use the version of Relation Transformer presented in Zhang et al. (2023) with 4 hidden layers, node dimension of 64, and edge dimension of 32. We optimized the network using a $1e$ 3 learning rate with AdamW (Loshchilov and Hutter, 2017) optimizer and a 1000 steps warmup schedule.

General. We optimized the model for 250 epochs for the ModelNet40 experiments and 300/100 epochs for the FMNIST 1/10 views respectively. Additionally, we utilize the validation set for early stopping, i.e. selecting the best model w.r.t validation accuracy. We repeat all experiments using 3 random seeds and report the average classification accuracy along with the standard deviation.

Appendix D. Weight space augmentation details

input space-based augmentations. Similar to rotation, scaling the coordinates by a factor s is equivalent to scaling the weights $W_1(sx) = (W_1 s)x$. Furthermore, a translation by an offset t can be absorbed into the bias $W_1(x + t) + b_1 = W_1 x + (W_1 t + b_1)$. These augmentations are natural, but they only change the parameters of the first layer, so their effectiveness may be limited.

General data augmentations. Dropout augmentation sets a parameter to 0 with probability p_{drop}. Quantile-based dropout first computes a threshold based on which it zeroes out the q-th quantile that is closest to 0. Gaussian noise augmentation adds Gaussian noise to all parameters with the standard deviation set in relation to the layer's standard deviation between the parameters.