How to Train Neural Field Representations: A Comprehensive Study and Benchmark

Supplementary Material

1. NeFs Architectures

SIREN [9]. SIRENs, short for sinusoidal representation networks, are a variant of multilayer perceptron characterized by the utilization of sinusoidal activations. By employing periodic activation functions, SIRENs enhance the network's capability to grasp nuances in the signal, particularly when handling wave-based signals and images. The configuration of a SIREN network is the following:

$$\begin{aligned} h^{(1)} &= x \\ h^{(i)} &= \sin\left(\Omega_0 W^{(i-1)} h^{(i-1)} + b^{(i-1)}\right), \; i = 2, \ldots, k-1 \\ f_\theta(x) &= W^{(k)} h^{(k)} + b^{(k)}, \end{aligned} \tag{1}$$

where $W^{(i-1)} \in \mathbb{R}^{d_i \times d_{i-1}}$ and $b^{(i-1)} \in \mathbb{R}^{d_i}$ denote the weight matrix and bias vector for the i-th layer, respectively, and Ω_o is a scalar. Here, the $\sin$ function is utilized as the activation function at each layer.

MFNs [5]. *Multiplicative Filters Networks* are NeFs architectures that don't rely on compositional depth for expressivity, but rather on nonlinear filters that are applied to the input and iteratively passed through linear functions and multiplied together. In particular, an MFN is defined by the recursion

$$\begin{aligned} z^{(1)} &= g\left(x; \psi^{(1)}\right) \\ z^{(i+1)} &= \left(W^{(i)} z^{(i)} + b^{(i)}\right) \odot g\left(x; \psi^{(i+1)}\right), \; i = 1, \ldots, k-2 \\ f_\theta(x) &= W^{(k-1)} z^{(k-1)} + b^{(k-1)}, \end{aligned} \tag{2}$$

where $\odot$ is the element-wise multiplication, $W^{(i)} \in \mathbb{R}^{d_{i+1} \times d_i}$, $b^{(i)} \in \mathbb{R}^{d_{i+1}}$ and $g : \mathbb{R}^d \to \mathbb{R}^{d_i}$ are the nonlinear filters parameterized by ψ_i that are applied to the input.

In this paper we use a linear layer composed with a sine function as filters: $g\left(x; \psi^{(i+1)}\right) = \sin \omega x + \phi$ where $\omega_i \in \mathbb{R}^{d \times d_1}$ and $\phi_i \in \mathbb{R}^{d_1}$. Interestingly, one can show that such a multiplicative filter network is equivalent to a linear function of an exponential (in k) number of Fourier basis functions.

RFFNets [10]. *Random Fourier Features Networks* leverage the Random Fourier Features technique, initially introduced in machine learning, to approximate kernel methods efficiently. In the context of neural fields, RFFNet uses random Fourier features to approximate the feature maps resulting from the kernel functions. Instead of explicitly computing the kernel function, which can be computationally intensive for high-dimensional data, RFFNet employs random projections to map input data into a higher-dimensional space. These random projections mimic the feature space resulting from a kernel function, such as the Gaussian or radial basis function (RBF) kernel. The expression of a RFFNet is

$$\begin{aligned} h^{(1)} &= \sqrt{\sigma} W^{(0)} x \\ h^{(2)} &= [\sin(z^{(1)}),\, cos(z^{(1)})] \\ h^{(i)} &= \text{ReLU}\left(W^{(i-1)} h^{(i-1)} + b^{(i-1)}\right), \; i = 3, \ldots, k-1 \\ f_\theta(x) &= W^{(k)} h^{(k)} + b^{(k)}, \end{aligned} \tag{3}$$

where $W^{(0)} \in \mathbb{R}^{d_1 \times 2}$ and $\sqrt{\sigma}$ is also called the `std` and is a hyperparameter that controls the frequency range of the embedding layer.

Metrics for reconstruction quality. *Peak Signal-to-Noise Ratio (PSNR)* is a widely used metric in signal processing that measures the quality of a reconstructed or processed signal concerning the original signal. It evaluates the fidelity of the reconstructed signal by comparing the maximum possible power of the original signal to the power of the difference between the original and reconstructed signals, expressed in decibels (dB). A higher PSNR value indicates a smaller difference between the signals, suggesting better reconstruction quality and less distortion.

The *Intersection over Union, (IOU)* is a metric often used in shape reconstruction or object detection tasks within computer vision. In the context of shape reconstruction, the IOU measures the overlap between the predicted shape (such as a bounding box, mask, or region) and the ground truth shape in an image. It calculates the ratio between the area of overlap and the area of union between the predicted and ground truth shapes. Higher IOU values, closer to 1, indicate a better match between the predicted and actual shapes, signifying more accurate reconstruction or detection.

2. NeFs Initialization

SIRENs. As pointed out by the authors, if not initialized properly, a SIREN results in poor reconstruction. We follow the principled initialization scheme proposed by the authors aimed at maintaining the activation distribution across the network, ensuring that the initial output remains independent of the number of layers. This can be done by sampling the rows w_i of the weight matrices from a uniform distribu-

Hyperparameter	Range	Range type
Initialization	[True, False]	Categorical
Weight decay	[1e-8, 1e-2]	Logarithmic
Learning rate	[1e-5, 1e-1]	Logarithmic
Number of steps	[20, 200, 1000, 5000, 10000, 20000]	Categorical

Table 1. Range of the *optimization* hyperparameters used for the initial study.

tion:

$$w_i \sim \mathcal{U}\left(-\frac{c}{\sqrt{\text{fan\_in}}}, \frac{c}{\sqrt{\text{fan\_in}}}\right). \tag{4}$$

We refer to [9] for the derivations.

MFNs. The FourierNet has similar behavior to the SIREN network, as $\sin$ activation functions are being used as filters. The linear layers – i.e. $W^{(i)}, b^{(i)}, i = 1, \dots, k-1$ – are initialized with the same scheme as a SIREN. However, the number of filters will now directly affect the final result, as these are not acting sequentially upon each other, instead, they are a multiplicative factor that affects each linear layer element-wise. Therefore, the filters are initialized according to the following:

$$w_i \sim \mathcal{U}\left(-\sqrt{\frac{\text{s}}{k-1}}, \sqrt{\frac{s}{k-1}}\right), \tag{5}$$

where $k-1$ is the number of filters and s is an input scaling constant. The latter is a hyperparameter that has a similar effect to the `std` of the RFFNet, which we fix to 16 in our experiments.

RFFNets. For the RFFNet, we choose to initialize the coefficients of the embedding using a standard Gaussian, such that $W^{(0)} \sim \mathcal{N}(0, I)$. Then, the layers are initialized using the uniform version of Kaiming He's initialization:

$$w_i \sim \mathcal{U}\left(-\sqrt{\frac{6}{\text{fan\_in}}}, \sqrt{\frac{6}{\text{fan\_in}}}\right), \tag{6}$$

which is a uniform distribution with variance of $2/\text{fan\_in}$.

3. The Classifier Architectures

Our classifier network is inspired by the work of [7], where the authors propose to use the computational graph of neural networks and encode NeF representations with graph networks or transformers that are invariant to the permutation symmetries present in the parameter space. We construct a message-passing GNN where the biases of each NeF layer correspond to node features, while the weights correspond to edge features. The authors also extend transformer architectures, in particular PNA [3], and Relational Transformer [4]. In the benchmarks, the Transformer mentioned is a Relational Transformer. The DWSNet is again a permutation invariant architecture that has been proposed in [8]. It is a composition of several different linear equivariant layers, such as DeepSets [11], and pointwise nonlinearities. To obtain an invariant classifier we simply compose an equivariant DWSNet with an invariant linear layer, eventually follwd by a fully connected MLP for expressivity.

The GNN uses 4 steps of message passing, with a hidden dimension of 64. The MLP used for the update function on the nodes has 3 layers and 256 hidden dimension, while the one used for the edges has 2 layers and 256 hidden dimension.

NeF	Hyperparameter	Range	Range type
SIREN	Hidden dim.	[8, 128]	Logarithmic
	Num. layers	[3, 6]	Linear
	Ω_0	[1, 1e2]	Logarithmic
RFFNet	Hidden dim.	[4,128]	Logarithmic
	Num. layers	[3, 6]	Linear
	$\sqrt{\sigma}$	[1e-2, 1e2]	Logarithmic
MFN	Hidden dim.	[8, 128]	Logarithmic
	Num. filters	[1, 6]	Linear

Table 2. Range of the *architecture* hyperparameters used for the initial study.

NeF	Hyperparameter	Range
SIREN	Hidden dim.	[8, 32, 64]
	Num. steps	[5, 15, 25, 50, 75, 1000, 5000, 10000, 20000, 50000]
	Learning rate	1e-3
	Weight decay	0
	Num. layers	3
	Ω_0	9
RFFNet	Hidden dim.	[16, 32, 64]
	Num. layers	[5, 15, 25, 50, 75, 1000, 5000, 10000, 20000, 50000]
	Learning rate	1e-4
	Weight decay	0
	Num. layers	5
	$\sqrt{\sigma}$	1e-1
MFN	Hidden dim.	[16, 32, 64]
	Num. filters	[5, 15, 25, 50, 75, 1000, 5000, 10000, 20000, 50000]
	Learning rate	5e-3
	Weight decay	0
	Num. filters	4

Table 3. Parameters used during the second phase of the study. These are chosen based on the results obtained during the first phase.

3.1. Ablation on the Architectures

To evaluate the ability of the models to overfit to the neural representations, we carried out an ablation study using both an MLP and a GNN. The MLP uses linear layers with ReLU activations and batch normalization [6]. The GNN is the one previously described. For both, we decide to train a small/*base*/large versions. For the MLP these correspond

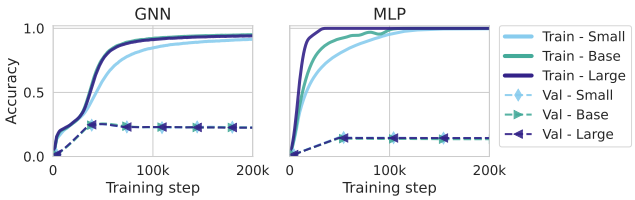


Figure 1. Training and test accuracy for three different-size GNN and MLP models on a CIFAR10 neural dataset. Overfitting ability is clear also for small capacities.

to 74k/116k/363k parameters, while for the GNN they correspond to 500k/1.2M/1.5M parameters. In both cases, we use a CIFAR10 dataset from our study and simply use the neural representations as inputs. For the MLP the input is the concatenated and flattened version of the weights and biases, while for the GNN it is the actual graph of the neural field, where the connectivity is determined by the weights, and node features by the biases. We show the training and validation curves in Fig. 1. These show clear signs of overfitting, the training curve of the GNN slowly approaches the maximum accuracy, while the one of the MLP does so quickly. For both, the validation curves are almost identical at all scales, likely due to capacity saturation.

4. Extent of the Study

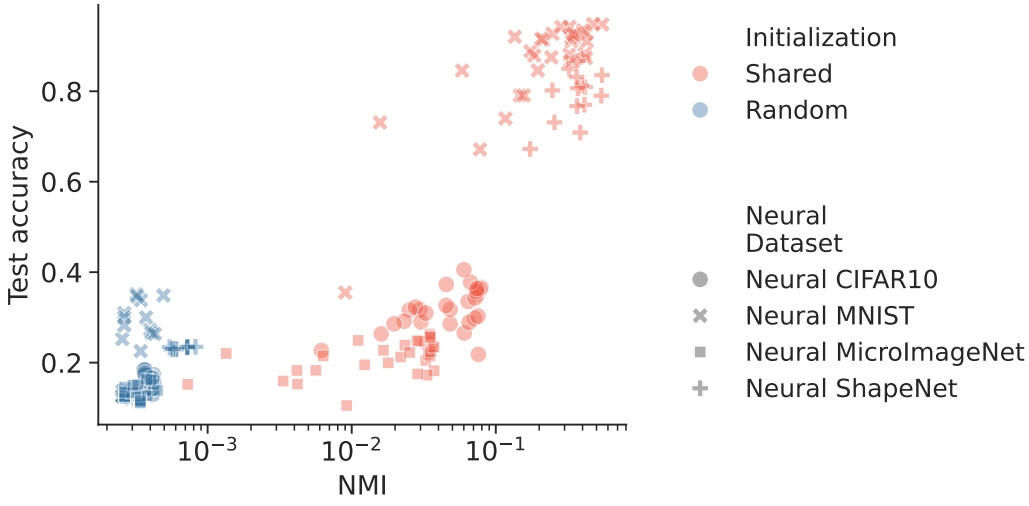
In total, the study we carried out required performing around 30k experiments, each corresponding to a new neural dataset being fit, and a classifier being trained on it. The study was split into two phases.

In the initial phase, we searched a large space of hyperparameters using a smaller subset of the whole dataset, and trained the classifier only for 10 epochs. In the initial hyperparameter exploration, we looked at *optimization* – see Tab. 1 – and *architecture* hyperparameters – see Tab. 2. For this exploration, we used the TPE sampler implemented in Optuna [1, 2]. In short, this sampler uses the previous results to inform the choice of the best hyperparameters to use next. Each experiment took between a few minutes to a few hours to run, depending on the number of steps and the size of the architecture. This was done for all 4 datasets and 3 models.

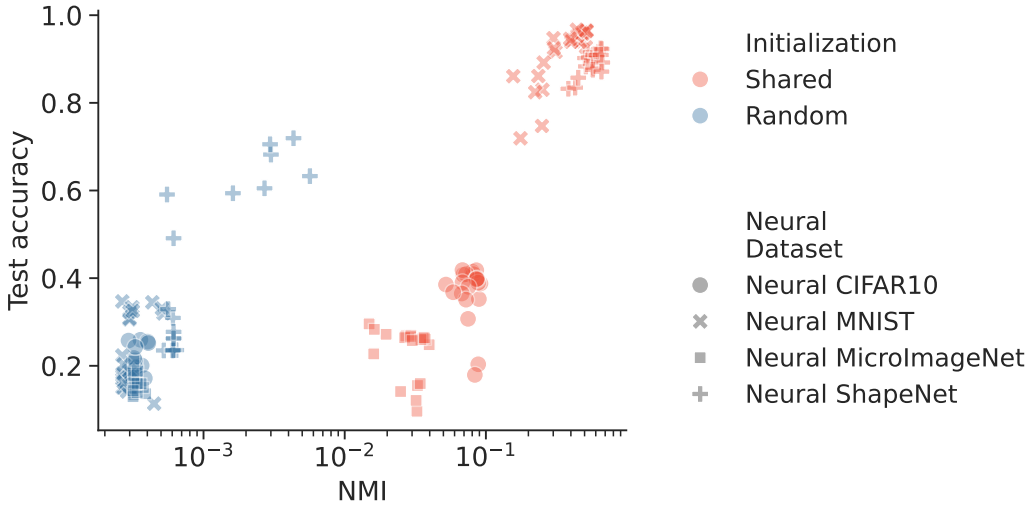
In the second phase, we ran a full grid search over the initialization scheme, the number of steps, and the hidden dimension of the NeFs. These were the most impactful parameters according to the search performed in the first phase. This grid search was performed for all datasets and models. Refer to Tab. 3.

5. Additional Experiments

Follows additional experiments carried out on FourierNet (MFN) and RFFNet. The results align with what was found when using SIREN, which suggests that these findings hold true across architectures. It is important to remember that, although the architectures tested are different and employ different activation functions, they are still very similar to each other, as the fundamental building block is fully connected feed-forward neural networks, with a small number of layers.



(a) RFFNet

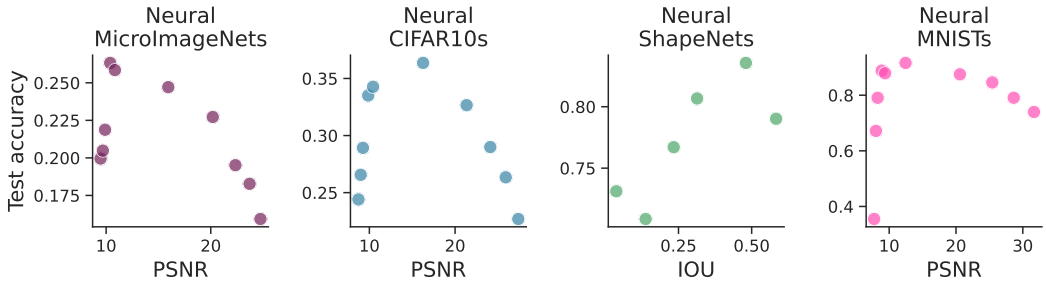


(b) MFN

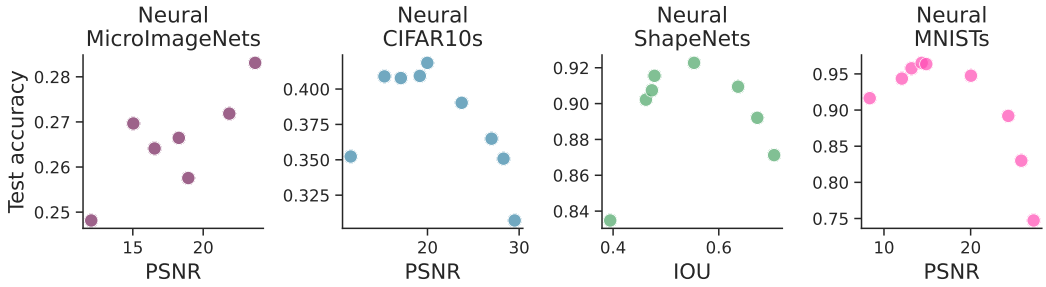
Figure 2. Results of the *test accuracy* (↑) vs *NMI* (↑) using different initialization on 220 Neural Datasets created using different hidden dimensions and the number of steps with RFFNets and MFNs. Different datasets are stylized using different markers. Shared initialization leads to semantically structured NeF representation and, generally to better performance. Both RFFNet and MFN achieve the same separation when using the shared initialization, as SIREN did.

NeF	Neural dataset	Initialization	Test accuracy	Gain (%)
RFFNet	*Neural MNISTs*	Random	$0.20_{\pm 0.09}$	
		Shared	$\mathbf{0.86}_{\pm \mathbf{0.12}}$	327.10
RFFNet	*Neural CIFAR10s*	Random	$0.16_{\pm 0.02}$	
		Shared	$\mathbf{0.31}_{\pm \mathbf{0.05}}$	102.33
RFFNet	*Neural MicroImageNets*	Random	$0.13_{\pm 0.02}$	
		Shared	$\mathbf{0.21}_{\pm \mathbf{0.04}}$	60.65
RFFNet	*Neural ShapeNets*	Random	$0.23_{\pm 0.003}$	
		Shared	$\mathbf{0.79}_{\pm \mathbf{0.06}}$	239.21
MFN	*Neural MNISTs*	Random	$0.23_{\pm 0.08}$	
		Shared	$\mathbf{0.91}_{\pm \mathbf{0.07}}$	304.58
MFN	*Neural CIFAR10s*	Random	$0.20_{\pm 0.03}$	
		Shared	$\mathbf{0.37}_{\pm \mathbf{0.06}}$	82.87
MFN	*Neural MicroImageNets*	Random	$0.16_{\pm 0.02}$	
		Shared	$\mathbf{0.23}_{\pm \mathbf{0.06}}$	46.39
MFN	*Neural ShapeNets*	Random	$0.42_{\pm 0.19}$	
		Shared	$\mathbf{0.89}_{\pm \mathbf{0.03}}$	109.83

Table 4. Average *test accuracy* (mean $\pm$ standard deviation) and percentage gain across 30 neural datasets trained using either shared initialization or random initialization with different number of steps and hidden dimensions. The gains achieved with shared initialization are consistent across all tested settings.

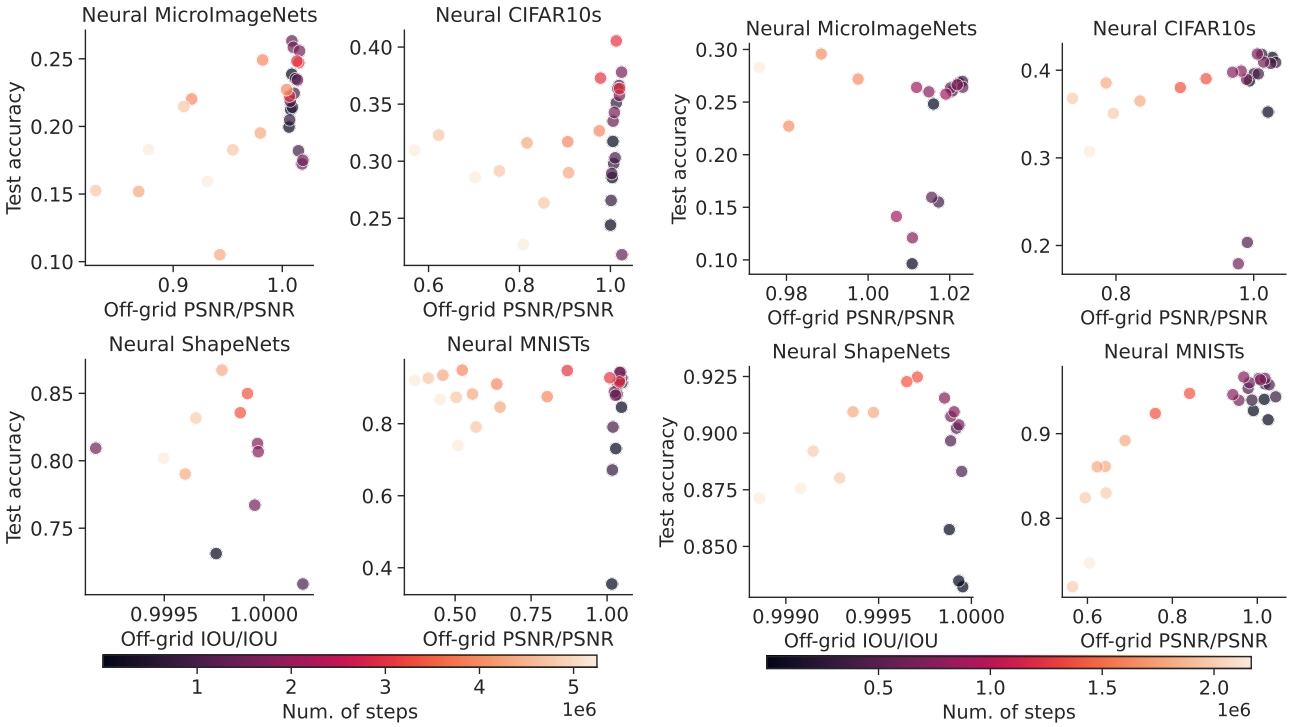


(a) RFFNet



(b) MFN

Figure 3. Results for the *reconstruction quality* (↑) vs *test accuracy* (↑) experiment. Similarly to SIRENs, for RFFNets and MFNs, by fixing the NeF's architecture we can more clearly see that there is a trade-off between visual quality and classification accuracy.



(a) RFFNet

(b) MFN

Figure 4. We fit 220 neural datasets using different hidden dimensions and number of steps while keeping the same shared initialization. Similarly to SIRENs, for MFNs and RFFNets the ratio of off-grid reconstruction quality and in-grid reconstruction quality can be used to form a heuristic that correlates with high test accuracy. The results across different architectures may be more subtle because of the tuning of different parameters that were not explored here, such as the `std` in the RFFNet or the input scaling s for the MFN.

References

[1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2623–2631, 2019. 3

[2] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. *Advances in neural information processing systems*, 24, 2011. 3

[3] Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. Principal neighbourhood aggregation for graph nets. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 2

[4] Cameron Diao and Ricky Loynd. Relational attention: Generalizing transformers for graph-structured tasks. In *International Conference on Learning Representations (ICLR)*, 2023. 2

[5] Rizal Fathony, Anit Kumar Sahu, Devin Willmott, and J Zico Kolter. Multiplicative filter networks. In *International Conference on Learning Representations*, 2020. 1

[6] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015. 2

[7] Miltiadis Kofinas, Boris Knyazev, Yan Zhang, Yunlu Chen, Gertjan J Burghouts, Efstratios Gavves, Cees G. M. Snoek, and David W. Zhang. Graph Neural Networks for Learning Equivariant Representations of Neural Networks. In *12th International Conference on Learning Representations (ICLR)*, 2024. 2

[8] 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. 2

[9] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in neural information processing systems*, 33:7462–7473, 2020. 1, 2

[10] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in Neural Information Processing Systems*, 33:7537–7547, 2020. 1

[11] 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. 2