End-to-End Implicit Neural Representations for Classification

Alexander Gielisse, Jan van Gemert
Delft University of Technology

Abstract

Implicit neural representations (INRs) such as NeRF and SIREN encode a signal in neural network parameters and show excellent results for signal reconstruction. Using INRs for downstream tasks, such as classification, is however not straightforward. Inherent symmetries in the parameters pose challenges and current works primarily focus on designing architectures that are equivariant to these symmetries. However, INR-based classification still significantly under-performs compared to pixel-based methods like CNNs. This work presents an end-to-end strategy for initializing SIRENs together with a learned learning-rate scheme, to yield representations that improve classification accuracy. We show that a simple, straightforward, Transformer model applied to a meta-learned SIREN, without incorporating explicit symmetry equivariances, outperforms the current state-of-the-art. On the CIFAR-10 SIREN classification task, we improve the state-of-the-art without augmentations from 38.8% to 59.6%, and from 63.4% to 64.7% with augmentations. We demonstrate scalability on the high-resolution Imagenette dataset achieving reasonable reconstruction quality with a classification accuracy of 60.8% and are the first to do INR classification on the full ImageNet-1K dataset where we achieve a SIREN classification performance of 23.6%. To the best of our knowledge, no other SIREN classification approach has managed to set a classification baseline for any high-resolution image dataset. Our code is available at https://github.com/SanderGielisse/MWT.

1. Introduction

Implicit neural representations (INRs) [3, 5–7, 10, 24–26, 32–34] encode complex continuous signals compactly in a neural network parameterized by θ. In the case of images, an INR continuously maps spatial image coordinates $x, y \in \mathbb{R}$ to RGB values. Specifically, we define the function $f_\theta : \mathbb{R}^2 \rightarrow \mathbb{R}^3$ such that:

$$f_\theta(x, y) = (r, g, b),$$

where (x, y) denotes continuous spatial coordinates, $r, g, b \in \mathbb{R}$ are the RGB values at that position. By doing so, θ encodes the image implicitly, instead of explicitly in a pixel value grid.

A common model choice for f_θ is a multilayer perceptrons (MLP) [25, 32, 34, 36], which enable high-quality reconstructions. We identify two main advantages of using MLP-based implicit neural representations (INRs). First, the capacity of the model $f_\theta(x, y)$ is not necessarily uniformly distributed across the image space, unlike representations based on fixed-resolution pixel grids. Second, the signal used as input is not required to be an equidistant pixel grid; any subset of observations from a signal can be used to train $f_\theta(x, y)$. Unfortunately, while INRs are highly effective for high-resolution reconstructions, using these implicit representations directly for downstream tasks, such as classification, remains challenging, as it requires reasoning on the parameters θ.

To perform a downstream task such as classification on the parameters θ, an additional model g is required, which takes θ as input. This involves constructing a model architecture that can process the weights of another architecture as its input. However, θ can contain many symmetries. For example, in the case of MLPs, reordering the nodes and their associated weights introduces permutation symmetries; that is, a different arrangement of weights that corresponds to the exact same function. Similarly, scale-symmetries allows scaling all parameters in one layer by a factor s and the following layer by $1/s$ results in an identical function, even though θ has changed.

One approach to address these symmetries is to realign the weights so that all symmetries map to the same network. Unfortunately, this alignment problem is intractable [1, 28]. An alternative solution is to design the downstream architecture g to be equivariant to the symmetries in θ, effectively bypassing the alignment issue. Consequently, many recent works have adopted this equivariant design approach for the design of the downstream architecture [8, 15, 17, 21, 27, 45]. However, the performance of these approaches remains behind that of pixel-based classification methods. One possible reason for this could be that RGB pixel-based representations are inherently easier

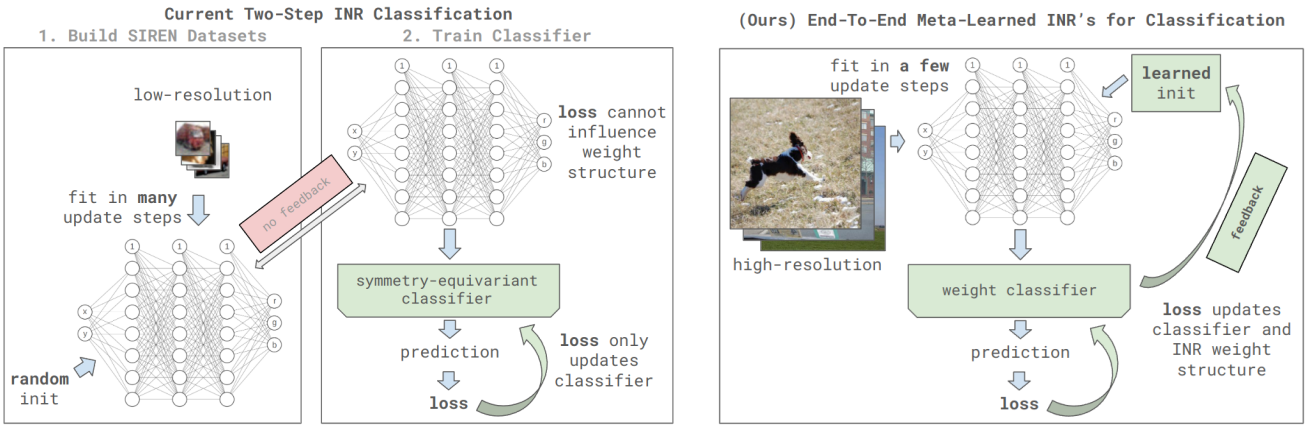


Figure 1. **Left:** the existing two-step INR classification approach, where the process involves building INR datasets by fitting images using many update steps without classifier feedback for its weight structure. The classifier is trained separately, and the classification loss cannot influence the INR. **Right:** the proposed end-to-end meta-learned INR approach for classification. This method fits high-resolution images using a few update steps with learned initialization and feedback from the classifier, allowing the classification loss to update both the classifier and the INR weight structure, enhancing downstream performance while ensuring quick convergence.

for a downstream model to interpret than the weights of another neural network. It may be that the weights θ of the INR lack sufficient "structure", making it difficult for downstream models to identify useful image features. The cause of sub-par performance being the absence of structure is supported by the findings of [29], who find that using the same, shared, INR initialization for all images and then make image-specific INRs by updating the shared initialization for each particular image INR yields improved classification results. This shared initialization may avoid symmetries by picking a fixed reference point. This is corroborated by the observations of [29] that more INR update steps to minimize the reconstruction loss improves the reconstruction quality, it actually reduces classification performance. Essentially, as the weights diverge from the initial shared INR, the structure becomes increasingly distinct, making classification of the INR more challenging. For this reason, we balance the INR fitting process between reconstruction quality and classification performance, in an end-to-end meta-learning setting using only a few update steps. What is more, using only a few update steps allows for a highly efficient implementation, which allows INR classification on high-resolution images and fitting INRs on data-augmentations in the interpretable image space.

Rather than merely sharing a common INR initialization across all images, we enforce structural consistency beyond shared initialization. Namely, we propose a meta-learning approach to jointly optimize the INR initialization and the learning rate schedule used in the image-specific updates. The objective is to find an initialization and a learning rate schedule such that, when an INR is fitted to an image signal, the resulting parameters θ have a structure that is directly interpretable for the classification model $g(\theta)$. Rather than the current two-step approach of first converting a set of images into their INR representations and as an independent next step training a classifier, we make the INR fitting process part of the classifier training loop. We do this by back-propagating through the INR optimization steps themselves. This way, an end-to-end trainable setting is achieved where the classifier can influence the structure of the INR, see Figure 1. We focus specifically on classification as the downstream task and use the popular SIREN as our MLP-based INR as commonly done [8, 15, 27, 45], although our approach is likely applicable to other tasks and other INRs as well. Our contributions are as follows.

- **End-to-end learning of INR classification**: Development of a meta-learning initialization strategy for SIRENs, combined with a meta-learned learning-rate scheme, aimed at enhancing classification performance on SIRENs.
- **Computational efficiency**: The high convergence speed of our approach allows high-resolution image classification and enables the use of interpretable spatial image augmentations during training. We explore a computationally efficient variant where a SIREN is learned on a subset of the pixels in each step. This further reduces computational cost, without deterioration of reconstruction quality or classification accuracy.
- **Simple classifier design**: We apply a straightforward, standard, Transformer model to the meta-learned SIREN representations, achieving improvements in classification accuracy without requiring complex classifiers explicitly designed to be equivariant to weight symmetries.
- **State of the art classification results**: We improve the SOTA on MNIST [19] accuracy from 96.6% to 98.8%, Fashion-MNIST [43] from 80.8% to 90.4% and CIFAR-

10 [18] from 38.8% to 59.6%. For CIFAR-10, we also use augmentations and improve the SOTA from 63.4% (inr2array [46] to 64.7%. Moreover, to our knowledge, we are the first to set SIREN classification baselines on high-resolution datasets. On Imagenette [14] we achieve 60.8% classification accuracy, while on the full ImageNet-1K we achieve 23.63% accuracy.

- **Comprehensive ablation study**: Detailed ablation studies on key components of the proposed meta-learning and Transformer-based approach, analyzing the impact of meta-initialization, learning-rate schemes, and Transformer architecture choices on reconstruction and classification performance.

2. Related Work

Implicit Neural Representations. INRs can be defined as a single differentiable and continuous function over the entire signal space, globally, often parameterized by an MLP [10, 32, 34, 36]. This allows flexible INRs capacity allocation, using less parameters on simpler areas. A key drawback, however, is that these models can take a long time to converge and are challenging to train to a low reconstruction error. An alternative is a hybrid explicit-implicit variant, where multiple local implicit representations are explicitly arranged spatially, either uniformly across the image or based on specific heuristics [3, 5–7, 23, 24, 26, 33]. In our work, we do not consider hybrid local models and focus only on fully implicit, global, models.

For global MLP-based INRs, ReLU activations [32] struggle to represent high-frequency components, often requiring positional embeddings [37]. The seminal SIREN [36] approach, on the other hand, demonstrates that a specific initialization with $\sin$ activation functions achieves smooth convergence, high reconstruction quality, and well-defined higher-order derivatives without requiring positional embeddings leading to several follow-up works [32, 34]. In our work, we adopt the SIREN approach due to its widespread adoption [3, 4, 10, 15], straightforward implementation, and proven effectiveness in preserving high-frequency details across diverse images.

MLP-based INR Classification. Classifying INRs is challenging as it requires reasoning on the MLP weights of the INR. The work by INSP-Net [44] addresses this by differential operators, directly applied to INRs, enabling both low-level image-to-image translation and more complex tasks like classification. Alternatively, INR2VEC [8] uses a feature encoder for each node in the MLP, followed by max-pooling to obtain a global feature vector. However, this approach does not account for potential symmetries present in the neuron weights.

Equivariant INR Classifiers. DWS-Net [27] proposes a network that can take as input an INR parameterized by an MLP, by processing it through a series of layers that are equivariant to the permutation symmetries of an MLP. NFN [45] improves upon this by making stronger symmetry assumptions to improve parameter efficiency and practical scalability. Graph Metanetworks [17, 21] extend this work to not only accept MLPs as input, but show that any set of operations that can be described by a graph. Neural Functional Transformers (NFTs) [46] define an attention-based architecture to process the weights of MLPs and CNNs, and propose their INR2ARRAY method to convert an INR into its permutation invariant latent representation. Moreover, while most work has been focused on equivariance to the inherent permutation symmetries, recent work has also explored the use of graph meta-networks that are scale-equivariant [15] to the scale symmetries in the weights. However, in our work, we will take a different approach. Namely, we demonstrate that enforcing structure on the MLP parameters offers an alternative to the downstream symmetry equivariance approach, showing that strong performance can be achieved without explicitly modeling equivariances in the classifier, allowing the use of straightforward, standard, classifiers.

Figure 2. Our method illustrated for a high-resolution Imagenette [14] image. A meta-learned SIREN initialization is updated for a small amount of gradient steps, in this case $k = 4$. The resulting weights are then passed to a classifier. Meta-learning allows us to back-propagate through the update steps, end-to-end optimizing the SIREN both for reconstruction as for classification.

Differentiable Meta-Learning. Model-Agnostic Meta-Learning (MAML) [11] is a meta-learning technique, with the objective of finding a set of model initialization parameters θ that can be quickly adapted, in k 'inner loop' steps, using only a few samples. Meta-SGD [20] extends MAML by also learning the learning rates for the k steps. This approach uses the property that the gradient descent operator itself is differentiable, *i.e.*,

$$\theta_{k+1} = \theta_k - \alpha \nabla_\theta \mathcal{L}_{\text{inner}}(\cdot; \theta_k), \tag{1}$$

is differentiable with respect to the learning rate α and to θ_k if the loss function $\mathcal{L}_{\text{inner}}$ is also differentiable with respect to θ_k. Then, for this θ_k, we can compute a different task loss $\mathcal{L}_{\text{task}}$. This task loss can be minimized by back-propagating

through the gradient steps of the inner-loop to update the initial θ_0, such that after k steps of minimizing $\mathcal{L}_{\text{inner}}$ we obtain θ_k that minimizes $\mathcal{L}_{\text{task}}$. We use both MAML and meta-SGD for meta-learning the INR initialization and learning rate schedule, to balance reconstruction as well as downstream classification.

Meta-Learning for INRs. Approaches similar to MAML are used in [38] to fit an implicit 3D NeRF [25] in just a few update steps. The data-to-functa [10] approach uses a meta-learning to speed up the INR fitting process while also doing dimensionality reduction with reconstruction quality as the only optimization goal, although follow-up work shows that this does not scale to high-resolution images [3]. In fit-a-nef [29] they manually tune and investigate meta properties and confirm that sharing an initialization and using only a few update steps helps classification. Our work differs in that we do not want to manually tune such parameters, but use meta-learning to automatically learn a balance between good reconstruction and classification accuracy, using the benefits of a shared initialization with the computational advantages of having only a few update steps, for high-resolution images. To our knowledge, no previous works use classification performance as an optimization goal in the INR meta-learning.

3. Method: Meta Weight Transformer (MWT)

3.1. Meta-Learned INR

We use SIREN [36] as the INR, and follow their initialization scheme to generate the very first random initial θ. We meta-learn this SIREN initialization θ so it can be used as a shared starting point for all images for fitting an image-specific SIREN, which is updated k steps for each individual image. We also learn a learning rate schedule $\alpha \in \mathbb{R}^{k \times |\theta|}$ which contains a learning rate for each parameter at each of the k update steps, based on [20]. The objective when learning this θ and α is not only to ensure rapid convergence in only k steps in terms of good reconstruction quality, but also to obtain SIREN parameters that are directly interpretable by a simple downstream INR classifier.

For clarity, we explain our method using a batch size of 1 with a single data point of one image $\mathbf{x}$ with class label y. In Figure 2 we show a visual overview. We explain the method below, and pseudo-code is given in Algorithm 1.

The meta-learning makes use of two loops: an outer loop over all training images, and an inner loop to update the shared learned initialization θ which is updated k times for each specific image $\mathbf{x}$ in an inner loop. We denote with ϕ the SIREN parameters θ after updating them k times. So, we define a downstream model $h_\psi(\phi)$ that takes as input the k times updated sample-specific SIREN parameters ϕ. To update the initial θ to become ϕ, we set $\mathcal{L}_{\text{inner}}$ to be the MSE reconstruction loss of the SIREN reconstruction $f_\theta(\cdot)$

Algorithm 1 Task-Specific SIREN Meta-Learning.
For the inner-loop, we minimize a reconstruction loss $\mathcal{L}_{\text{rec}}$. We then optimize the initial SIREN parameters θ and learning rate schedule α such that ϕ does not only encode the image with high quality, but is also in a format that can be correctly classified by our classifier $h_\psi(\phi)$. Here, f is our SIREN, w_{cls} is a scalar that we can use to change the classifier influence on the meta-learning process, and Adam refers to the use of the Adam [16] optimizer.

Init random SIREN with parameters θ
Init learning rates α for all k update steps $\alpha \in \mathbb{R}^{k \times |\theta|}$
while not converged **do** `<outer loop>`
  Sample training image $\mathbf{x}$ with classification label y
  Set starting INR parameters to shared base $\phi = \theta$
  for $i = (1, 2, \ldots, k)$ **do** `<inner loop>`
    $\phi \leftarrow \phi - \alpha_i \nabla_\phi \mathcal{L}_{\text{rec}}(f_\phi, \mathbf{x})$
  end for
  Predict classification label $\hat{y} \leftarrow h_\psi(\phi)$
  Get θ, α gradient $g^{\text{cls}}_{\theta,\alpha} \leftarrow \nabla_{\theta,\alpha} \mathcal{L}_{\text{cls}}(\hat{y}, y)$
  Get θ, α gradient $g^{\text{rec}}_{\theta,\alpha} \leftarrow \nabla_{\theta,\alpha} \mathcal{L}_{\text{rec}}(f_\phi, \mathbf{x})$
  Combine θ, α gradients $g_{\theta,\alpha} \leftarrow g^{\text{rec}}_{\theta,\alpha} + g^{\text{cls}}_{\theta,\alpha} * w_{\text{cls}}$
  Update SIREN initialization $\theta \leftarrow \text{Adam}(\theta, g_\theta)$
  Update learning rates $\alpha \leftarrow \text{Adam}(\alpha, g_\alpha)$
  Get ψ gradient $g_\psi \leftarrow \nabla_\psi \mathcal{L}_{\text{cls}}(\hat{y}, y)$
  Update classifier $\psi \leftarrow \text{Adam}(\psi, g_\psi)$
end while

for each of the k inner-loop steps, for all n pixels in image $\mathbf{x}$ as

$$\mathcal{L}_{\text{rec}}(f_\phi, \mathbf{x}) = \frac{1}{n} \sum_{j=1}^{n} (\mathbf{x}_j - f_\phi(\mathbf{x}_j))^2 \tag{2}$$

We cannot use the classification loss in the inner-loop, as it has to be performed at test time for new samples, where we do not have the ground truth y of that sample. After the inner-loop, we classify ϕ to the predicted class label $\hat{y}$ using our INR classifier $\hat{y} \leftarrow h_\psi(\phi)$. Then, we compute both the reconstruction loss $\mathcal{L}_{\text{rec}}(f_\phi, \mathbf{x})$ and the classifier loss $\mathcal{L}_{\text{cls}}(\hat{y}, y)$. First, we update the parameters of the meta-learning process. For both these losses, we compute the gradients with respect to both the initialization values θ, and learning rates α. We denote these gradients as $g^{\text{rec}}_{\theta,\alpha}$ and $g^{\text{cls}}_{\theta,\alpha}$ which gives

$$g^{\text{rec}}_{\theta,\alpha} \leftarrow \nabla_{\theta,\alpha} \mathcal{L}_{\text{rec}}(f_\phi, \mathbf{x}) \tag{3}$$

$$g^{\text{cls}}_{\theta,\alpha} \leftarrow \nabla_{\theta,\alpha} \mathcal{L}_{\text{cls}}(\hat{y}, y) \tag{4}$$

Then, we combine the gradients using a scalar w_{cls}, with which we can control the influence of the classifier on the meta-learning process, to obtain $g_{\theta,\alpha}$

$$g_{\theta,\alpha} \leftarrow g^{\text{rec}}_{\theta,\alpha} + g^{\text{cls}}_{\theta,\alpha} * w_{\text{cls}} \tag{5}$$

which we use to obtain the gradients that we will use to update θ and α. Lastly, we compute the gradients to update the classifier weights ψ. We denote this gradient as g_ψ which we compute as

$$g_\psi \leftarrow \nabla_\psi \mathcal{L}_{\text{cls}}(\hat{y}, y) \tag{6}$$

As the weights of the classifier ψ are solely optimized for classification performance, so we only use g_ψ to update them.

3.2. SIREN Classification using Transformers

To classify our image-specific SIREN parameters ϕ, we require an architecture $h_\psi(\phi)$ that is capable of accepting linear layers with bias terms as input. We use a simple, straight-forward, Transformer model [41] that operates directly on the weight space of a SIREN model without considering any explicit INR symmetries.

To input the SIREN network to the classifier we use the property that a linear transformation followed by the addition of biases can be combined into a single matrix operation, if we pad each layer with a 1-valued input. Specifically, the weights of a linear layer $W_L \in \mathbb{R}^{c_{\text{in}} \times c_{\text{out}}}$ and the bias term $W_b \in \mathbb{R}^{c_{\text{out}}}$ can be represented together as $W \in \mathbb{R}^{(c_{\text{in}}+1)\times c_{\text{out}}}$. This combined matrix can be applied to the input vector padded with a 1, resulting in an operation that is identical to applying the weight matrix followed by the bias addition. We show a SIREN network this way in Figure 1.

We train the classifier on the hidden layers of the SIREN. SIREN networks often maintain the same layer dimensionality throughout their depth. This means that, if we only consider the hidden layers, each neuron in a SIREN network has $c_{in} + 1$ input weights. We use each output neuron of the SIREN's hidden layers to be a token for our Transformer model, where each neuron has a $c_{\text{in}} + 1$ dimensional feature vector; those being the input weights to that neuron.

So, for a SIREN network with l hidden layers and dimensionality d, there are $d \times l$ input tokens for our Transformer model. The feature vectors for these tokens do not inherently have any positional bias, making it challenging for the Transformer to identify which neuron a specific feature vector corresponds to. To address this, we introduce a learned positional bias $\beta \in \mathbb{R}^{|\theta|}$ for each weight. Additionally, we observe that the SIREN base weights θ and the image-specific weights after the inner loop ϕ are often similar. Namely, element-wise $|\theta - \phi|$ results in mostly small numbers. Consequently, our classifier struggles to interpret these weights accurately, likely due to the low-frequency bias of our model [37]. To mitigate this, we provide the network with the difference between the base SIREN weights θ and the image-specific weights ϕ instead, rather than the absolute values. Because this difference often results in small values, and the commonly used Transformer initializations expect normalized inputs, we scale the feature vectors by a scalar λ, which we typically set to 500. So, before providing the weights to the Transformer, we first convert our SIREN as

$$\phi_{\text{scaled}} \leftarrow \lambda(\phi - \theta + \beta). \tag{7}$$

Reducing Computational Cost Computationally, the reconstruction loss in the inner loop is typically calculated for each pixel at each of the k inner-loop steps, requiring $H \times W \times k$ forward passes through the SIREN network, where H and W represent the image height and width in pixels, respectively. The computation graph for all k steps and for all $H \times W$ pixels must be stored to facilitate meta-learning of the base initialization θ, which can become resource-intensive for high-resolution images. To address this, we also explore an alternative approach that fits the SIREN to the image using a random subset S of pixels at each inner-loop step, rather than processing all pixels. We denote the fraction of pixels used with $s \in [0, 1]$ such that $|S| = (s \times H \times W)$. Note that when s is set to $(1/k)$, on average the model still sees every pixel once. By decreasing s, we effectively make the k inner-loop steps more stochastic, while simultaneously saving on computational cost.

	Classification Accuracy (%)		
Method	**MNIST**	**Fashion-MNIST**	**CIFAR-10**
MLP	17.55 ± 0.01*	19.91 ± 0.47*	11.38 ± 0.34*
Inr2Vec [8]	23.69 ± 0.10*	22.33 ± 0.41*	-
DWS [27]	85.71 ± 0.57	67.06 ± 0.29	34.45 ± 0.42*
NFN_{NP} [45]	78.50 ± 0.23*	68.19 ± 0.28*	33.41 ± 0.01*
NFN_{HNP} [45]	79.11 ± 0.84*	68.94 ± 0.64*	28.64 ± 0.07*
NG-GNN [17]	91.40 ± 0.60	68.00 ± 0.20	36.04 ± 0.44*
ScaleGMN [15]	96.57 ± 0.10	80.46 ± 0.32	36.43 ± 0.41
ScaleGMN-B [15]	96.59 ± 0.24	80.78 ± 0.16	38.82 ± 0.10
WT (Ours)	91.38 ± 1.67	83.97 ± 1.38	43.78 ± 0.64
MWT (Ours)	98.33 ± 0.11	89.41 ± 0.25	56.90 ± 0.29
MWT-L (Ours)	**98.80 ± 0.06**	**90.43 ± 0.23**	**59.57 ± 0.52**

Table 1. Accuracy of various methods on MNIST, Fashion-MNIST, and CIFAR-10 datasets where models operate directly on the SIREN weights. The values represent the mean ± standard deviation ($n = 3$), in the no data-augmentation setting. The star * denotes that the numbers were taken from [15], the (Ours) rows are computed by us, and other values are taken from the original papers. We show that end-to-end meta-learned classifier gradients (MWT) strongly improves over not having such classifier gradients (WT). Moreover, MWT, as well as our large model MWT-L set a strong new baseline for single SIREN classification.

4. Experiments

Implementation details. The classifier is a 10 block Transformer [41], each block with a dimensionality matching that of the SIREN network. LayerNorm [2] is applied prior to the self-attention operation and the fully connected GELU [13] layer. We implement multi-head attention with a head

dimension of 64 and use LayerScale [39] initialized at 0.1. For optimization, we employ the AdamW [16, 22] optimizer with a batch size of 16, a learning rate of 1×10^{-4}, and a weight decay of 1×10^{-4}.

In the inner-loop optimization described in Algorithm 1, we use plain SGD without momentum. We observed that a higher AdamW learning rate of 1×10^{-2} for the inner-loop learning rates α improves performance, so we adopt this value there. The inner-loop learning rates for meta-SGD are initialized as $\alpha \sim \text{Uniform}(0.1, 1.0)$. We set the rescaling factor for SIREN weights to $\lambda = 500$. For the SIREN network itself, we use a $\omega = 10.0$ on the first layer, a hidden dimensionality of 128, and a depth of 4, unless specified otherwise. All experiments were conducted on NVIDIA A40 GPUs with mixed-precision enabled.

4.1. Classification of INRs

We evaluate INR classification on MNIST [19], Fashion-MNIST [43], and CIFAR-10 [18]. We train each model on each of the datasets for 10 epochs.

Impact of End-to-End Classifier Gradients We investigate our end-to-end Meta Weight Transformer (MWT) to verify if gradients from the classifier effectively influence the meta-learned SIREN initialization and learning rate schedule. Thus, we compare MWT to a variant of Algorithm 1 that does not back-propagate the $\mathcal{L}_{\text{cls}}$ loss through the inner loop and instead uses it solely for optimizing the downstream Transformer model. We label this variant as the Weight Transformer (WT), for which we set $w_{\text{cls}} = 0$. In this modified version, the downstream Transformer classifier does not influence the meta-learned initialization nor the learning rate schedule, leading to a learning procedure that is learned purely for fast and high-quality reconstruction. This WT approach resembles the traditional two-step method outlined in Figure 1. Here, the INR is trained separately with only a reconstruction objective, after which a classifier is trained on the INRs separately.

The results in Table 1 show a large improvement of MWT over WT. This indicates that incorporating end-to-end classifier gradients to guide the structuring of the INR through meta-learning is advantageous.

Comparisons to State-of-the-art The Table 1 also includes a comparison with current state-of-the-art results in SIREN classification where no augmentations are used. Both WT and MWT mostly perform similar or better than previous state-of-the-art models, with MWT clearly outperforming all earlier methods.

A possible explanation for why WT does well, is that our SIREN is learned through the meta-learning process designed to achieve convergence within just a few update steps. Namely, while non-meta learning approaches typically require several dozen update steps to achieve reasonable reconstruction accuracy, our method uses only $k = 6$ update steps. This reduction in the number of steps may contribute to the observed improvement in classification performance, consistent with findings from [29], which demonstrated that reducing the number of update steps can enhance downstream classification performance.

Comparing to other INR baselines that do use augmentations on CIFAR-10, these report 41.27% [27], 46.60% [45], 56.95% [15], and 63.4% [46]. Our MWT-L models scores $59.57\% \pm 0.52$, which is quite reasonable with respect to computation effort, because the augmented models typically need to fit 10x-50x more SIRENS. Furthermore, we scale our MWT approach further, use a 20-layer Transformer, use spatial augmentations, increase epochs to 40, and achieve a new CIFAR-10 [18] SIREN classification SOTA of 64.7%.

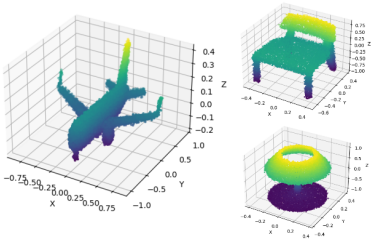
4.2. Upscaling to High-Resolution

So far we focused on low-resolution images, we now explore the scalability of our approach to high-resolution images. To begin, we train our MWT classifier on Imagenette [14], a 10-class subset of the ImageNet dataset [9], with augmentations enabled, as we clarify below.

Spatial Augmentations When constructing datasets composed of SIREN networks, image-space augmentations such as scaling, rotation, flipping, and translation can be computationally expensive, because it requires re-training the SIREN to each new augmented signal. To address this, weight-space augmentations have been proposed [27, 35]. However, augmenting the weight space in a way such that it actually still represents the type of images that can naturally occur is challenging. Fortunately, since we can fit a SIREN to a new signal in just a few gradient steps, we can simply apply image-space augmentations. Therefore, in the following experiment, we also evaluate model variants where spatial image augmentations of the training data are used. We decrease the number of inner-loop steps to $k = 4$, as this is computationally less costly, and the results in Table 4 did not show a large drop in performance for this change. Furthermore, we investigate if we can push the performance by using our larger model variant MWT-L with SIREN dimensionality of 256. By increasing the SIREN dimensionality, we also automatically increase the dimensionality of the Transformer tokens to 256. To facilitate the higher image resolution, we use $\omega = 30.0$ for the first layer of the SIREN for all of the high-resolution models. We vary the amount of sub-sampling by setting different values for s. We present the results in Table 2. Note that for $s = 0.25$ with $k = 4$, the model on average sees every pixel once, similar to traditional image processing models.

First, we compare the WT and MWT models. Surprisingly, we find that the best performance for classification using the small models is achieved for MWT with augmentations while only using 5% of all the pixels in each of the reconstruction inner-loop steps, resulting in a test accuracy of 57.27% on the high-resolution Imagenette [14] dataset. Interestingly, using smaller values for s does not appear to strongly impact classification performance nor reconstruction quality. A similar pattern is found for our large MWT-L model, where the classification accuracy, as well as the reconstruction quality, is relatively consistent between varying s. This suggests that the meta-learned SIREN network may have learned an implicit image bias that enables it to interpolate the missing pixels with sufficient accuracy, perhaps similar to [40], or simply that just a partial signal observation is sufficient to achieve the current performance. This might allow the model to maintain performance even when provided with incomplete data during training, suggesting potential further applications in scenarios involving sparse input data [12] or point cloud processing.

ImageNet-1K We train MWT-L on the full large-scale ImageNet-1K [9] dataset. We use a dimensionality of 256, use spatial image augmentations, use a subsampling rate of 0.01 for each of the $k = 4$ inner-loop steps. In Table 3 we show the performance for varying w_{task}. To our knowledge, our work is the first to train SIREN networks for a high-resolution collection of images such as ImageNet [9].



MWT-L Model Subsampling (parameter s)	Accuracy (%)	Validation (sec / epoch)	Training Time (min / epoch)	Memory (train, GiB)
0.1	85.82	116.2	7.8	15.2
0.01	85.41	49.5	3.7	11.0
0.001	84.04	46.7	3.5	10.8
0.00005	30.35	41.7	3.1	10.7

Figure 3. ModelNet40 [42] results on unsigned distance functions. Trained for 150 epochs using dimensionality of 256, number of inner-loop steps $k = 4$, with augmentations enabled. Timings include the inner-loop fitting of the INR. The inr2vec work [8] scores an accuracy of 87.0% on this dataset, with non-INR approaches like PointNet [30] and PointNet++ [31] outperforming these with accuracies of 88.8% and 89.7% respectively.

4.3. Ablations

To analyze the impact of our model's hyperparameters, we conduct an ablation study using the CIFAR-10 dataset, with results presented in Table 4. Notably, when examining the task influence on the meta-learning of the SIREN initialization, denoted as w_{cls}, we observe a clear trade-off between reconstruction quality and classification performance. Specifically, if w_{cls} in Algorithm 1 is set too high, both reconstruction quality and classification accuracy decline. On the other hand, if w_{cls} is set too low, only classification performance is negatively affected. We find that setting $w_{\text{cls}} = 0.01$ strikes a good balance, yielding good classification performance while maintaining acceptable reconstruction quality. For the number of inner-loop steps k, we observe that taking more gradient steps generally enhances both classification and reconstruction performance. However, higher values of k can be computationally expensive, which is a known restriction of MAML [11], so we use $k = 6$ as a practical compromise. Similarly, increasing the depth of the Transformer and the width of the SIREN network also improves performance but leads to a substantial increase in the number of model parameters. We set the SIREN dimensionality to 128 for our MWT model, and to 256 for our MWT-L model.

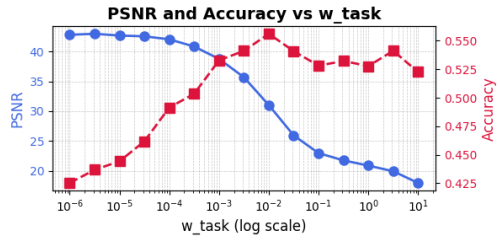


Figure 4. We provide a visualization of the trade-off made in the ablation of the MWT model. Increasing the influence of the classifier on the meta-learning of the INR (w_{task}) decreases reconstruction quality, but increases classification performance up to about 0.01, after which both decrease.

Classifier	Classifier Influence	Accuracy (%)	PSNR (dB)	#Params CLS	#Params SIREN	Validation (test set, sec)	Training (min / epoch)	Memory (GiB)
WT	($w_{\text{task}} = 0$)	38.98	**43.09**	1.1M	465K	30.3	10.8	4.2
	($w_{\text{task}} = 0.01$)	**56.02**	32.64	1.1M	465K	29.0	11.6	4.2
NFN_{NP}	($w_{\text{task}} = 0$)	29.54	**44.21**	1.8M	465K	35.4	16.0	0.7
	($w_{\text{task}} = 0.01$)	**49.89**	32.55	1.8M	465K	28.1	11.7	4.5
NFN_{HNP}	($w_{\text{task}} = 0$)	25.77	**44.41**	3.6M	465K	35.6	14.2	5.2
	($w_{\text{task}} = 0.01$)	**48.37**	32.83	3.6M	465K	28.0	11.7	5.8

Table 5. We show our method on the NFN classifier [45] on CIFAR-10 [18]. For NFN, we use three-layered network with a layer dimensionality of 64, followed by two fully connected layers. We train for 10 epochs, with spatial augmentations enabled. To compare, NFN [45] reports 44.1% for NFN_{HNP} , and an accuracy of 46.6% for NFN_{NP} , but do this through augmentation by fitting 20 SIRENs from different initializations for each image.

Model	Accuracy (%)		PSNR (dB)		#Params	#Params	Validation	Training	Memory
	No Aug.	With Aug.	No Aug.	With Aug.	CLS	SIREN	(sec / epoch)	(min / epoch)	(GiB)
$\text{WT}_{s=0.25}$	47.96 ± 0.29	46.83 ± 0.25	$\mathbf{23.23 \pm 0.08}$	$\mathbf{23.27 \pm 0.05}$	1.1M	332K	40.7	5.0	19.6
$\text{WT}_{s=0.1}$	48.11 ± 0.42	47.02 ± 0.37	22.50 ± 0.02	22.47 ± 0.04	1.1M	332K	20.5	2.3	9.0
$\text{WT}_{s=0.05}$	49.01 ± 0.17	47.97 ± 1.27	21.64 ± 0.03	21.79 ± 0.04	1.1M	332K	12.7	1.4	5.6
$\text{MWT}_{s=0.25}$	48.79 ± 0.36	56.13 ± 0.28	21.14 ± 0.02	21.14 ± 0.08	1.1M	332K	41.4	8.0	21.1
$\text{MWT}_{s=0.1}$	50.14 ± 0.87	56.78 ± 0.27	20.82 ± 0.04	21.14 ± 0.02	1.1M	332K	20.5	3.5	9.6
$\text{MWT}_{s=0.05}$	50.23 ± 0.31	57.27 ± 0.38	20.68 ± 0.03	20.98 ± 0.06	1.1M	332K	13.5	2.1	5.9
$\text{MWT-L}_{s=0.1}$	51.80 ± 0.23	60.62 ± 0.37	21.94 ± 0.02	22.31 ± 0.03	4.3M	1.3M	42.4	7.6	24.1
$\text{MWT-L}_{s=0.05}$	52.04 ± 0.45	60.75 ± 0.65	21.56 ± 0.02	21.86 ± 0.04	4.3M	1.3M	28.7	4.7	17.4
$\text{MWT-L}_{s=0.02}$	$\mathbf{52.93 \pm 0.90}$	$\mathbf{60.75 \pm 0.37}$	20.95 ± 0.04	21.14 ± 0.02	4.3M	1.3M	21.2	2.9	13.5

Table 2. Accuracy and PSNR results for WT, MWT and MWT-L models with varying subsampling settings on the Imagenette [14] dataset. The columns (with aug.) indicate the use of spatial augmentations to the training data; scaling, translating, rotating and flipping. Note that due to random uniform subsampling, also the PSNR is computed on a random, though *different*, uniform subset of the pixels. So, the shown PSNR is just an approximation of the real PSNR.

Model	Accuracy	PSNR
$w_{\text{task}} = 0.01$	**24.11%**	21.78 dB
$w_{\text{task}} = 0.001$	21.87%	22.09 dB
$w_{\text{task}} = 0.0001$	19.21%	**22.14** dB

Table 3. Results for the full ImageNet-1K [9] dataset trained for 40 epochs. Accuracy indicates the percentage of correctly classified samples, while PSNR represents the average reconstruction quality of the image. This model uses 12.1 GiB of memory for training, takes 306.0 ± 4.2 minutes for a single epoch, takes 216.8 ± 9.4 seconds for making predictions on the full validation set. The SIREN model has 1.3M parameters, and the WT classifier has 4.5M parameters.

5. Discussion

We introduce a straightforward Transformer architecture for classifying SIREN networks, without being equivariant to any parameter symmetries of the SIREN. We integrate the SIREN fitting process into the classifier's training loop in an end-to-end manner, allowing the classifier to adjust the initial parameterization and learning rate schedule of the SIREN for new images. This approach enables the classifier to shape the parameter space in a way that enhances classification performance, while also optimizing reconstruction quality. Our method establishes a new state-of-the-art baseline across four SIREN image classification benchmarks.

However, we are uncertain whether the initial parameterization and learning rate schedule alone can impose sufficient structure. Future work could explore additional learnable constraints on the SIREN parameters to further enhance classification performance. While we consider the current reconstruction quality sufficient for our work, improvements may be possible, especially for high-resolution datasets, which could be an area for further investigation.

We view this work as a step away from the conventional focus on achieving optimal reconstruction quality in implicit neural representations (INRs), shifting toward developing INRs that also optimize for classification accuracy. Future research on balancing the INR optimization between classification accuracy and reconstruction quality could further advance this direction. In principle, we do believe that continuous signals are best represented by a continuous function, instead of the accidental, discrete, sensor grid and as such we, explicitly, believe that INRs have a bright future.

Ablation	Parameter	Accuracy (%)	PSNR (dB)	Validation (sec/epoch)	Training (min/epoch)	Mem (GiB)	#Params CLS	#Params SIREN
Task Influence w_{task}	0	42.24	43.11	14.0	3.1	3.0	1.1M	465K
	0.001	53.53	38.92	14.2	4.0	3.0	1.1M	465K
	<u>0.01</u>	55.79	30.93	15.2	4.3	3.0	1.1M	465K
	0.1	54.00	23.11	15.5	4.2	3.0	1.1M	465K
	1.0	52.35	20.89	14.2	4.2	3.0	1.1M	465K
	10.0	52.34	18.14	14.0	4.1	3.0	1.1M	465K
Inner-Loop Steps k	1	49.70	23.63	6.7	1.7	2.2	1.1M	133K
	2	52.46	25.24	9.1	2.5	2.3	1.1M	199K
	4	54.79	29.10	11.0	3.1	2.6	1.1M	332K
	<u>6</u>	55.79	30.93	15.2	4.3	3.0	1.1M	465K
	8	55.52	30.96	16.9	5.0	3.3	1.1M	598K
	10	55.58	30.79	19.8	6.0	3.6	1.1M	731K
Scaling Factor λ	1	48.60	31.38	15.0	4.2	3.0	1.1M	465K
	10	54.80	31.71	13.9	4.0	3.0	1.1M	465K
	100	55.16	31.43	14.0	4.1	3.0	1.1M	465K
	<u>500</u>	55.79	30.93	15.2	4.3	3.0	1.1M	465K
	1000	55.54	30.59	14.0	4.1	3.0	1.1M	465K
	LayerNorm	54.89	31.18	14.0	4.1	3.0	1.1M	465K
SIREN Depth	2	55.39	28.50	10.5	2.9	1.5	1.0M	234K
	<u>4</u>	55.79	30.93	15.2	4.3	3.0	1.1M	465K
	6	55.52	31.96	21.9	6.2	4.9	1.1M	696K
	8	55.79	32.73	31.6	8.5	7.4	1.1M	927K
SIREN Width	64	51.54	27.65	9.0	3.3	1.1	273K	118K
	<u>128</u>	55.79	30.93	15.2	4.3	3.0	1.1M	465K
	256	57.19	33.15	43.6	11.2	12.7	4.3M	1.8M
Transformer Depth	5	54.41	31.03	12.2	3.6	2.2	581K	465K
	<u>10</u>	55.79	30.93	15.2	4.3	3.0	1.1M	465K
	15	56.77	30.99	15.8	4.6	3.7	1.6M	465K
	20	56.82	30.82	18.0	5.2	4.5	2.1M	465K
Meta-SGD Shared	<u>False</u>	55.79	30.93	15.2	4.3	3.0	1.1M	465K
	True	54.38	29.86	15.2	4.2	3.0	1.1M	133K
Meta-SGD LR	<u>0.01</u>	55.79	30.93	15.2	4.3	3.0	1.1M	465K
	0.001	53.34	27.82	13.9	4.1	3.0	1.1M	465K
	0.0001	50.58	25.95	15.2	4.2	3.0	1.1M	465K
Subsampling Rate s	<u>1.0</u>	55.79	30.93	15.2	4.3	3.0	1.1M	465K
	0.1	54.21	24.06	11.5	3.4	2.1	1.1M	465K
	0.01	46.45	18.84	12.0	3.5	2.0	1.1M	465K
	0.001	18.03	16.53	12.0	3.4	2.0	1.1M	465K

Table 4. Full ablation results with timings and memory usage for the proposed MWT, trained on 80% of the CIFAR-10 training dataset. The reported numbers are evaluated on the remaining 20% of the training set that was held out for validation. Accuracy indicates the percentage of correctly classified samples, while PSNR represents the average reconstruction quality of the image.

References

[1] Samuel K Ainsworth, Jonathan Hayase, and Siddhartha Srinivasa. Git re-basin: Merging models modulo permutation symmetries. *arXiv preprint arXiv:2209.04836*, 2022. 1

[2] Jimmy Lei Ba. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 5

[3] Matthias Bauer, Emilien Dupont, Andy Brock, Dan Rosenbaum, Jonathan Richard Schwarz, and Hyunjik Kim. Spatial functa: Scaling functa to imagenet classification and generation. *arXiv preprint arXiv:2302.03130*, 2023. 1, 3, 4

[4] Eric R Chan, Marco Monteiro, Petr Kellnhofer, Jiajun Wu, and Gordon Wetzstein. pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5799–5809, 2021. 3

[5] Yinbo Chen, Sifei Liu, and Xiaolong Wang. Learning continuous image representation with local implicit image function. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8628–8638, 2021. 1, 3

[6] Zhang Chen, Yinda Zhang, Kyle Genova, Sean Fanello, Sofien Bouaziz, Christian Häne, Ruofei Du, Cem Keskin, Thomas Funkhouser, and Danhang Tang. Multiresolution deep implicit functions for 3d shape representation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13087–13096, 2021.

[7] Zhang Chen, Zhong Li, Liangchen Song, Lele Chen, Jingyi Yu, Junsong Yuan, and Yi Xu. Neurbf: A neural fields representation with adaptive radial basis functions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4182–4194, 2023. 1, 3

[8] Luca De Luigi, Adriano Cardace, Riccardo Spezialetti, Pierluigi Zama Ramirez, Samuele Salti, and Luigi Di Stefano. Deep learning on implicit neural representations of shapes. *arXiv preprint arXiv:2302.05438*, 2023. 1, 2, 3, 5, 7

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 6, 7, 8

[10] Emilien Dupont, Hyunjik Kim, SM Eslami, Danilo Rezende, and Dan Rosenbaum. From data to functa: Your data point is a function and you can treat it like one. *arXiv preprint arXiv:2201.12204*, 2022. 1, 3, 4

[11] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017. 3, 7

[12] Benjamin Graham and Laurens Van der Maaten. Submanifold sparse convolutional networks. *arXiv preprint arXiv:1706.01307*, 2017. 7

[13] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016. 5

[14] Jeremy Howard and Sylvain Gugger. Fastai: a layered api for deep learning. *Information*, 11(2):108, 2020. 3, 6, 7, 8

[15] Ioannis Kalogeropoulos, Giorgos Bouritsas, and Yannis Panagakis. Scale equivariant graph metanetworks. *arXiv preprint arXiv:2406.10685*, 2024. 1, 2, 3, 5, 6

[16] Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 4, 6

[17] 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. 1, 3, 5

[18] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 3, 6, 7

[19] 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. 2, 6

[20] Zhenguo Li, Fengwei Zhou, Fei Chen, and Hang Li. Meta-sgd: Learning to learn quickly for few-shot learning. *arXiv preprint arXiv:1707.09835*, 2017. 3, 4

[21] Derek Lim, Haggai Maron, Marc T Law, Jonathan Lorraine, and James Lucas. Graph metanetworks for processing diverse neural architectures. *arXiv preprint arXiv:2312.04501*, 2023. 1, 3

[22] I Loshchilov. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 6

[23] Joost Luijmes, Alexander Gielisse, Roman Knyazhitskiy, and Jan van Gemert. ARC: Anchored representation clouds for high-resolution INR classification, 2025. 3

[24] Julien NP Martel, David B Lindell, Connor Z Lin, Eric R Chan, Marco Monteiro, and Gordon Wetzstein. Acorn: Adaptive coordinate networks for neural scene representation. *arXiv preprint arXiv:2105.02788*, 2021. 1, 3

[25] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021. 1, 4

[26] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM transactions on graphics (TOG)*, 41(4):1–15, 2022. 1, 3

[27] 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. 1, 2, 3, 5, 6

[28] Aviv Navon, Aviv Shamsian, Ethan Fetaya, Gal Chechik, Nadav Dym, and Haggai Maron. Equivariant deep weight space alignment. *arXiv preprint arXiv:2310.13397*, 2023. 1

[29] Samuele Papa, Riccardo Valperga, David Knigge, Miltiadis Kofinas, Phillip Lippe, Jan-Jakob Sonke, and Efstratios Gavves. How to train neural field representations: A comprehensive study and benchmark. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22616–22625, 2024. 2, 4, 6

[30] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017. 7

[31] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017. 7

[32] Sameera Ramasinghe and Simon Lucey. Beyond periodicity: Towards a unifying framework for activations in coordinate-mlps. In *European Conference on Computer Vision*, pages 142–158. Springer, 2022. 1, 3

[33] Vishwanath Saragadam, Jasper Tan, Guha Balakrishnan, Richard G Baraniuk, and Ashok Veeraraghavan. Miner: Multiscale implicit neural representation. In *European Conference on Computer Vision*, pages 318–333. Springer, 2022. 3

[34] Vishwanath Saragadam, Daniel LeJeune, Jasper Tan, Guha Balakrishnan, Ashok Veeraraghavan, and Richard G Baraniuk. Wire: Wavelet implicit neural representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18507–18516, 2023. 1, 3

[35] 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. 6

[36] 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, 3, 4

[37] 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. 3, 5

[38] Matthew Tancik, Ben Mildenhall, Terrance Wang, Divi Schmidt, Pratul P Srinivasan, Jonathan T Barron, and Ren Ng. Learned initializations for optimizing coordinate-based neural representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2846–2855, 2021. 4

[39] Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 32–42, 2021. 6

[40] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep image prior. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9446–9454, 2018. 7

[41] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017. 5

[42] 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. 7

[43] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017. 2, 6

[44] Dejia Xu, Peihao Wang, Yifan Jiang, Zhiwen Fan, and Zhangyang Wang. Signal processing for implicit neural representations. *Advances in Neural Information Processing Systems*, 35:13404–13418, 2022. 3

[45] 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, 2024. 1, 2, 3, 5, 6, 7

[46] 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, 2024. 3, 6