Abstract

Coordinate-based neural representations have shown significant promise as an alternative to discrete, array-based representations for complex low dimensional signals. However, optimizing a coordinate-based network from randomly initialized weights for each new signal is inefficient. We propose applying standard meta-learning algorithms to learn the initial weight parameters for these fully-connected networks based on the underlying class of signals being represented (e.g., images of faces or 3D models of chairs). Despite requiring only a minor change in implementation, using these learned initial weights enables faster convergence during optimization and can serve as a strong prior over the signal class being modeled, resulting in better generalization when only partial observations of a given signal are available. We explore these benefits across a variety of tasks, including representing 2D images, reconstructing CT scans, and recovering 3D shapes and scenes from 2D image observations.

1. Introduction

Recent work has demonstrated the potential of representing complex low-dimensional signals using deep fully-connected neural networks (typically referred to as multilayer perceptrons, or MLPs). A coordinate-based neural representation \( f_\theta \) for a given signal is an MLP (with weights \( \theta \)) that is optimized to map from an input coordinate \( x \) to the signal’s value at that coordinate. For example, \( f_\theta \) could map from 2D pixel coordinates to RGB color values to encode an image. Unlike a signal stored as a discretely sampled array of values, a coordinate-based neural representation is continuous and is not constrained to have a fixed spatial resolution. This fact has recently been exploited to design representations for 3D shapes (which typically occupy a small 2D subset of 3D space) that do not require cubic storage complexity, in contrast to 3D voxel grids [22, 24, 27, 34]. However, one limitation of these neural representations is that computing network weights \( \theta \) that reproduce a given signal typically requires solving an optimization problem by running many steps of gradient descent. This can take between seconds (when encoding a small image) and hours (when solving an inverse problem to recover a high resolution radiance field, as in NeRF [24]). Common approaches to address this issue include concatenating a latent vector to the input coordinate and supervising a single neural network to represent an entire class of signals [22, 27], or training a hypernetwork to map from signal observations (or a latent code) to MLP weights [33, 34]. However, each of these strategies is restricted to representing only signals within its learned latent space, potentially limiting its ability to express previously unseen target signals.

Recent work [32] has shown that optimization-based meta-learning can dramatically reduce the number of gra-
dient descent steps required to optimize a neural representation to encode a new signal in the case of signed distance fields of 2D and 3D shapes. In this work, we propose learning the weight initialization for neural representations across a wide variety of underlying signal types, such as images, volumetric data, and 3D scenes. We show that compared to a standard random initialization, using fixed, learned values for the initial network weights acts as a strong prior that enables both faster convergence during optimization and better generalization when only partial observations of the target signal are available. In the context of using neural representations for 3D reconstruction from images, a learned initialization specialized to a particular ShapeNet [2] class allows the network to recover 3D shape from a single image over the course of optimization, whereas a standard randomly initialized network fails unless provided with multiple input views. Given a meta-training set consisting of observations of different signals sampled from a fixed underlying class, our setup applies an optimization-based meta-learning algorithm (MAML [6] or Reptile [25]) in order to produce initial weights better suited for representing that specific signal class (e.g., face images from CelebA [20] or 3D chairs from ShapeNet [2]).

The biggest advantage of our approach is its simplicity. Given an existing framework for test-time optimization of a neural representation, implementing an outer loop with MAML or Reptile update steps only requires a few extra lines of code and a dataset of training examples. Once the meta-learning phase is complete, the learned initial weights can be stored and later reloaded in place of a standard network initialization whenever a new signal needs to be encoded. This minor implementation change can significantly alter the behavior of the network during optimization.

2. Related Work

Neural Representations Neural representations have recently risen to prominence as compact representations for 3D shapes. These methods represent shapes as implicit surfaces defined as a level set of an MLP network and enable full object reconstruction from incomplete 3D point cloud data or depth scans [3, 4, 9, 10, 14, 22, 23, 27]. Later work combined this idea with various formulations of differentiable rendering to recover neural representations of 3D shape using only 2D image observations [18, 19, 24, 26, 34, 37].

Coordinate-based neural networks have also been used to represent other low-dimensional signals, such as 2D images, where such networks (when trained via genetic algorithms) have been referred to as compositional pattern—producing networks [35]. Recent works have shown that standard ReLU MLPs fail to adequately represent fine details in these complex low-dimensional signals due to a spectral bias [28] and address this issue by either replacing the ReLU activations with sine functions [33] or by lifting the input coordinates into a Fourier feature space [36]. Our work makes use of these observations and presents a technique that enables a coordinate-based MLP to learn from the process of fitting many signals within a category so that it can quickly optimize to fit any new signal using fewer steps and fewer observations.

Meta-learning Meta-learning typically addresses the problem of few-shot learning, where some examples of a given task (including training and test data) are used to learn an algorithm that achieves better performance on new, previously unseen instances of the same task. A prototypical example from computer vision is few-shot image classification, where a network must learn to differentiate between new classes at test time based on only a small number of labeled instances of each class.

Most relevant to this work are optimization-based meta-learning algorithms such as Model-Agnostic Meta Learning (MAML) [6] and Reptile [25], as well as various extensions [1, 5, 7, 17, 29]. Given a network architecture for performing a task, these methods use an outer loop of gradient-based learning to find a weight initialization that allows the network to more efficiently optimize for new instances of the underlying task at test time. These methods assume the use of a standard gradient-based optimization method such as stochastic gradient descent or Adam [16] at test time, making them easy to layer on top of existing implementations, as opposed to more complex methods such as Ravi et al. [30], which trains a “meta-learner” LSTM network to perform gradient updates for the underlying task. An exhaustive review of meta-learning algorithms is provided in the survey paper by Hospedales et al. [12].

MetaSDF [32] specifically applies this idea of learning a weight initialization to the task of fitting neural representations to represent signed distance fields, and shows that this strategy achieves much more rapid convergence than standard approaches such as DeepSDF [27]. Our work applies meta-learning to neural representations for a wider variety of underlying signal types and further explores the power of using initial weight settings as a prior.

3. Overview

We define a finite signal $T$ as a function mapping from a bounded set $C \subset \mathbb{R}^d$ to $\mathbb{R}^n$, where we refer to elements $x \in C$ as $d$-dimensional coordinates. Examples include images (mapping from 2D pixel coordinates to 3D color values) or volumetric representations for 3D shapes (mapping from 3D locations to 4D tuples of color and density). A coordinate-based neural representation $f_\theta$ for $T$ is a fully connected neural network with $d$ input and $n$ output channels whose weights $\theta$ are optimized such that $f_\theta$ matches $T$ as closely as possible for all coordinates in $x \in C$. 
If direct pointwise observations \( \{(x_i, T(x_i))\} \) of the signal \( T \) are available, \( f_0 \) can be supervised by gradient descent using a simple L2 loss:

\[
L(\theta) = \sum_i \| f_0(x_i) - T(x_i) \|^2_2. \tag{1}
\]

Let \( \theta_0 \) denote the initial network weights before any gradient steps are taken, and let \( \theta_i \) denote the weights after \( i \) steps of optimization. Basic gradient descent applies the rule:

\[
\theta_{i+1} = \theta_i - \alpha \nabla_\theta L(\theta)|_{\theta=\theta_i}, \tag{2}
\]

with a learning rate parameter \( \alpha \), whereas more sophisticated optimizers such as Adam [16] keep track of gradient moments over time to redirect the optimization trajectory. Given a fixed budget of \( m \) optimization steps, different initial weight values \( \theta_0 \) will result in different final weights \( \theta_m \) and signal approximation error \( L(\theta_m) \). When emphasizing the functional dependence of \( \theta_m \) on the initial weights and a particular signal, we will write \( \theta_m(\theta_0, T) \).

It is often the case that only indirect observations of \( T \) are available, taken through some forward measurement model \( M(T, p) \). For example, if \( T \) is a 3D object, \( M(T, p) \) could be a 2D image captured of the object from camera pose \( p \). In this case, recovering a neural representation for \( T \) from observations \( \{p_i, M(T, p_i)\} \), requires solving an inverse problem by taking gradient steps on a loss that incorporates the forward model \( M \):

\[
L_M(\theta) = \sum_i \| M(f_\theta, p_i) - M(T, p_i) \|^2_2. \tag{3}
\]

If \( M \) discards too much information about \( T \) or the set of provided observations is too small, the resulting network \( f_\theta \) may not match \( T \) closely. For example, accurately recovering a 3D object from a single 2D view may not be possible without strong a priori knowledge of the object’s shape.

### 3.1. Optimizing initial weights

We assume that we are given a dataset of observations of signals \( T \) from a particular distribution \( \mathcal{T} \) (e.g., 2D face images or 3D chairs) and our goal is to find initial weights \( \theta_0^* \) that will result in the lowest possible final loss \( L(\theta_m) \) when optimizing a network \( f_0 \) to represent a new, previously unseen signal from the same distribution:

\[
\theta_0^* = \arg \min_{\theta_0} \mathbb{E}_{T \sim \mathcal{T}} [L(\theta_m(\theta_0, T))] \tag{4}
\]

This problem of trying to learn the initial weights of a network to serve as a good starting point for gradient descent across a distribution of tasks is addressed by a variety of optimization-based meta-learning algorithms, such as MAML [6] and Reptile [25].

### MAML [6]

Given a task \( T \), calculating the weight values \( \theta_m(\theta_0, T) \) requires taking \( m \) optimization steps, which are collectively referred to as the inner loop. MAML wraps an outer loop of meta-learning around this inner loop in order to learn the initial weights \( \theta_0 \). Each outer loop samples a signal \( T_j \) from \( \mathcal{T} \) and applies the update rule:

\[
\theta_0^{j+1} = \theta_0^j - \beta \nabla_\theta L(\theta_m(\theta_0, T_j))|_{\theta=\theta_0^j} \tag{5}
\]

with meta-learning step size \( \beta \). This update rule applies gradient descent to the loss on the weights \( \theta_m(\theta_0^j, T_j) \) resulting from the inner loop optimization.

### Reptile [25]

Reptile uses the same meta-learning setup as MAML but applies a simpler update rule that does not require calculating second-order gradients:

\[
\theta_0^{j+1} = \theta_0^j - \beta (\theta_m(\theta_0^j, T_j) - \theta_0^j) . \tag{6}
\]

This rule moves the previous weight initialization \( \theta_0^j \) in the direction of the task-optimized weights \( \theta_m(\theta_0^j, T_j) \).

### 3.2. Experimental setup

The meta-learning algorithms described previously are conceptually simple, requiring no changes to the architecture or optimization procedure of a coordinate-based neural representation when given a new signal to encode at “test time” (after meta-learning is complete). These algorithms produce only a set of initial network weights \( \theta_0^j \) that are then used as a starting point for gradient descent. Test-time optimization on new signals is not limited to the same number of steps \( m \) as were used in the inner loop during meta-learning; indeed, at test time we often observe benefits from optimizing for significantly more iterations than were used during the inner loop of the meta-learning algorithm.

MAML is typically able to produce a better initialization than Reptile given a fixed number of inner loop steps \( m \), but Reptile can be unrolled for more inner loop steps because it is less memory-intensive than MAML. For some tasks, MAML’s limited number of inner loop steps means that it can only observe a small percentage of the observations of a target signal. In these cases, we use Reptile to maximize the number of different observations seen over the course of the inner loop. Experimentally we find it beneficial to unroll more steps for more complex tasks.

Each of our experiments involves two phases:

1. **Meta-learning**, where we use MAML or Reptile in combination with a training dataset of example tasks (observations of different signal instances) to optimize initial network weights for that class of signals, and

2. **Test-time optimization**, where we use standard gradient-based optimization to fit the weights of a network to observations of a previously unseen signal from the same class.
We aim to answer the following question: how do different initial network weight settings influence the ability of a neural representation to fit to a new signal during test-time optimization?

4. Results

We present results on 2D image regression, 2D computed tomography (CT) reconstruction, 3D object reconstruction, and 3D scene reconstruction. For each task, we demonstrate the benefits of using meta-learned initial weights optimized to reconstruct a specific class of signals.

For 2D image regression, a meta-learned weight initialization leads to faster convergence and better performance during test-time optimization. For CT reconstruction, it allows for better reconstruction quality from fewer supervision views during test-time optimization. For 3D shape reconstruction from images, it allows for faster convergence at test time and makes single view reconstruction possible. For Phototourism landmark reconstruction, it can be optimized at test time to transfer the appearance of a single input image onto the whole landmark, which can then be rendered from novel camera views.

4.1. Tasks

Here we provide the basic setup for each task. Please see the supplement for full implementation details.

**Image regression** A prototypical example of a coordinate-based neural representation is an MLP optimized to represent a 2D image [33, 36] by taking in 2D pixel coordinates and outputting RGB color values. We consider four different distributions $T$: images of faces (CelebA [20]), natural images (Imagenette [13]), images of text (Text), and 2D signed distance fields of simple curves (SDF). Each category contains around ten thousand examples. Given a sampled image $T \sim T$, we provide all $178 \times 178$ pixels as observations for optimizing the network weights $\theta$ in the inner loop. Since this task is not memory constrained, we use MAML to meta-learn the weights over 2 unrolled gradient steps (separately for each category $T$). In each of these inner loop steps, the entire image is reconstructed and used to calculate the loss. For the MLP $f_{\theta}$, we use 5 layers with 256 channels each and sine function nonlinearities, as in SIREN [34].

**CT reconstruction** Computed tomography (CT) is a widely used medical imaging technique that captures projective measurements of the volumetric density of a target object. Tancik et al. [36] use a coordinate-based neural representation to reconstruct a 2D signal from 1D integral projections; the underlying MLP $f_{\theta}$ takes in a 2D coordinate and outputs a scalar volume density at that location. Here $T$ is a dataset of 2048 randomly generated $256 \times 256$ pixel Shepp-Logan phantoms [31], where we provide 2D integral projections of a bundle of 256 parallel rays from a random angle as the measurement for each sampled signal $T$ during meta-learning. We use Reptile to meta-learn the initial weights over 12 unrolled gradient steps. We found this to outperform MAML, which was limited to 3 unrolled steps due to memory constraints. For the MLP $f_{\theta}$, we use 5 layers with 256 channels each and ReLU nonlinearities, and we apply random Fourier features to the input coordinates [36].

**View synthesis for ShapeNet [2] objects** The goal of view synthesis is to generate a novel view of a scene from a set of reference images. Recently, neural radiance fields (NeRF) [24] proposed a method to accomplish this task by using a neural representation that predicts a color and density for any input 3D location and 2D viewing direction within the scene, along with a differentiable volumetric rendering model to generate new views from that representation. This network is optimized to minimize the resid-


<table>
<thead>
<tr>
<th>Init. Method</th>
<th>2 Step PSNR $\uparrow$</th>
<th># of iters to match $\downarrow$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard</td>
<td>10.88</td>
<td>37.92 ± 6.31</td>
</tr>
<tr>
<td>Mean</td>
<td>14.48</td>
<td>25.59 ± 4.57</td>
</tr>
<tr>
<td>Matched</td>
<td>13.73</td>
<td>26.32 ± 4.17</td>
</tr>
<tr>
<td>Shuffled</td>
<td>16.29</td>
<td>25.80 ± 4.02</td>
</tr>
<tr>
<td>Meta</td>
<td>30.37</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 1. Comparison of different initialization methods on an image regression task using the CelebA dataset. We report reconstruction PSNR after two steps of test-time optimization. The meta-learned initialization (Meta) significantly outperforms all other initializations. We also report the average number of iterations necessary to match the accuracy of Meta after two steps.

Table 2. PSNR comparison of four different learned initializations for image regression. Each row corresponds to an initialization meta-learned over a different underlying image dataset. The columns indicate which dataset images are sampled from during testing. The best initialization for each task (bolded) is the one specifically optimized on training images drawn from the same dataset. We observe that initializations transfer better between more similar datasets (CelebA and Imagenette, both natural images) and poorly between less similar datasets (the frequency spectrum of Text images is unlike that of the other categories).

4.2. Baselines

As well as a Standard randomly initialized network (Glorot et al. [11]), we compare to various other initialization schemes in several of our experimental settings:

- **Mean**: we optimize a network from scratch such that its output matches the mean signal $E_{T \sim T}[T]$ from the current class $T$.
- **Matched**: we optimize a network from scratch such that its output matches the output of a network using the meta-learned initialization for the current class $T$.
- **Shuffled**: we randomly permute the weights (within each network layer) of the meta-learned initialization $\theta^*_0$ for the current class $T$.

Both the Mean and Matched baselines demonstrate the difference between having a good initialization in signal space versus weight space—despite Mean and Matched being initialized so that the loss against a randomly sampled signal will be low, they are a worse starting point for gradient descent than the actual meta-learned initial weights. The Shuf-
baseline demonstrates that matching the statistical distribution of the meta-learned initial weights is not sufficient for better convergence or generalization. We find that using the Adam [16] optimizer performs best for all of the baseline initializations, but that standard stochastic gradient descent works best for the meta-learned initializations (we choose the best optimizer and hyperparameters for each task and initialization using a held-out validation set, see supplement for details).

### 4.3. Faster convergence

#### Image regression

In Figure 2, we visualize the network output for a variety of initial weight settings, showing the output images after 0, 1, and 2 gradient steps of test-time optimization. The meta-learned initial weights are optimized to represent face images (CelebA [20]). When using the learned initial weights $\theta_0^*$ (Meta), the target image is already clearly visible after the very first step. In contrast, the baseline initialization methods take an order of magnitude more iterations to represent the target image to the same accuracy (see Table 1). The Mean, Matched, and Shuffled baselines perform better than the completely random Standard initialization, but still take over ten times as many iterations to reach the same quality as the meta-initialized network can after 2 steps. In particular, this demonstrates that neither matching the image space output nor the statistical distribution of the meta-learned weights is sufficient for achieving a similar speedup.

#### View synthesis for ShapeNet [2] objects

In Figure 5, we plot the image reconstruction accuracy for a held-out test set of objects from the Chair category. During test-time optimization, 25 views are observed. We find that starting from the optimized weights $\theta_0^*$ allows the network to recover the chair more quickly compared to the Standard weight initialization. We note that after many steps, both methods end up at a similar quality.

### 4.4. Generalizing from partial observations

#### Image regression within a category

We perform meta-learning experiments across multiple datasets to determine the extent that the optimized weight initialization acts as a class-specific prior. We compare initializations trained on four different image datasets (CelebA, Imagenette, Text, and SDF). Table 2 presents a confusion matrix demonstrating that optimizing the network initialization does in fact induce a dataset-dependent prior, with each learned initialization generalizing best to the same dataset distribution it was trained on.

#### CT reconstruction from sparse views

We report the reconstruction quality over a test set of phantoms given varying numbers of views at test time in Table 3 and visualize one test example in Figure 3. We observe poor reconstructions from the Standard initialization when few views are provided. The meta-learned initializations are consistently able to match the PSNR of Standard with half as many views. The Mean initialization is generated by training a network to reconstruct the mean of the training phantoms. It is better able to preserve the structure of the phantom compared to Standard but still performs worse than the meta-learned initializations.

<table>
<thead>
<tr>
<th>Init. Method</th>
<th>1 Views PSNR</th>
<th>2 Views PSNR</th>
<th>4 Views PSNR</th>
<th>8 Views PSNR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>14.72</td>
<td>15.39</td>
<td>17.43</td>
<td>25.19</td>
</tr>
<tr>
<td>Matched</td>
<td>14.07</td>
<td>15.51</td>
<td>20.25</td>
<td>24.77</td>
</tr>
<tr>
<td>Shuffled</td>
<td>13.64</td>
<td>14.17</td>
<td>16.69</td>
<td>22.09</td>
</tr>
<tr>
<td>Meta</td>
<td><strong>15.09</strong></td>
<td><strong>18.70</strong></td>
<td><strong>22.00</strong></td>
<td><strong>27.34</strong></td>
</tr>
</tbody>
</table>

Table 3. Comparison of initialization methods on a CT reconstruction task. Each “view” consists of 256 parallel rays. The data-dependent prior acquired during meta-learning improves reconstruction quality when fewer views are observed.

---

![Figure 3. Sparse Recovery: Examples of CT reconstructions of a Shepp-Logan phantom from a sparse set of views. The meta-learned initial weights encode a data-dependent prior that improves reconstruction in the limited data regime.](image-url)
Single image view synthesis for ShapeNet [2] A simple-NeRF model with a Standard random initialization relies on multi-view consistency to reconstruct the appearance of a 3D object. With only a single view, this naïve model is unable to recover any meaningful shape. We find that a learned initialization “bakes in” a class-specific shape prior that enables the recovery of 3D geometry (Figure 4, Table 4). We can meta-learn an effective weight initialization for single-view reconstruction by optimizing over a dataset with 25 training views of each object (MV Meta). We find that this prior persists even if the meta-training dataset only contains a single reference image per scene (SV Meta), meaning that the meta-learning phase has no access to multiview information for any particular object.

View synthesis with appearance transfer for Phototourism [15] As described in §4.1, these images have different camera poses and visual appearance (lighting, sky, etc.) as they are taken by tourists at different times. Our goal at test time is to explore the landmark from varying camera viewpoints but rendered with the same appearance as in a target photograph. In every step of the meta-learning outer loop, we supervise the simple-NeRF model to match the appearance of a random photo of the landmark (with varying pose and appearance). We find that performing test-
we optimize on one half of an image and report image metrics on the other half. We compare our Reptile setup, we train a simple-NeRF model with a standard random initialization from scratch on each landmark, then test-time optimize it to match the left half of each new view before evaluating it on the right half. This is algorithmically equivalent to Reptile with one inner optimization step. We find that unrolling Reptile for 64 inner steps performs better, producing significantly clearer renderings of the landmark.

5. Conclusion

Our results show that simply modifying a coordinate-based neural representation’s initial weight values can guide the network along a significantly better optimization trajectory, without changing the underlying architecture or test-time optimization procedure. These meta-learned initial weights can result in faster convergence or act as a strong prior for representing signals from a given distribution. This partially ameliorates a major shortcoming of neural representations (separately optimizing a network for each new signal) without limiting their representational power.

There are many additional directions to explore, such as applying more sophisticated meta-learning algorithms or more precisely characterizing the geometry of weight space for these networks. One limitation of our current approach is that it requires a sizable dataset of example signals from a target distribution in order to derive beneficial initial weights. Another shortcoming is that our method still requires some amount of test-time optimization.

As the number of use cases for neural representations continues to rapidly expand, we believe this work takes an important step toward understanding the importance of their initial weights and optimization behavior.
References


