Generalizable Neural Fields as Partially Observed Neural Processes

Jeffrey Gu
ICME
Stanford, CA
jeffgu@stanford.edu

Kuan-Chieh Wang
Department of Computer Science
Stanford, CA
wangkua1@stanford.edu

Serena Yeung
Department of Computer Science
Stanford, CA
syyeung@stanford.edu

Abstract

Neural fields, which represent signals as a function parameterized by a neural network, are a promising alternative to traditional discrete vector or grid-based representations. Compared to discrete representations, neural representations both scale well with increasing resolution, are continuous, and can be many-times differentiable. However, given a dataset of signals that we would like to represent, having to optimize a separate neural field for each signal is inefficient, and cannot capitalize on shared information or structures among signals. Existing generalization methods view this as a meta-learning problem and employ gradient-based meta-learning to learn an initialization which is then fine-tuned with test-time optimization, or learn hypernetworks to produce the weights of a neural field. We instead propose a new paradigm that views the large-scale training of neural representations as a part of a partially-observed neural process framework, and leverage neural process algorithms to solve this task. We demonstrate that this approach outperforms both state-of-the-art gradient-based meta-learning approaches and hypernetwork approaches.

1. Introduction

Neural fields, also known as neural implicit representations or coordinate-based neural networks, have shown great promise as an alternative to traditional discrete representations. Neural fields consider signals and other objects as functions (*fields*) mapping coordinates to values, and approximate these functions with neural networks. They have been used to represent a myriad of signals including 3D objects/scenes [25, 34], CT scans [35], and 3D human motion [36]. This approach has several advantages over discrete representations, including being continuous and having non-zero derivatives [32] (depending on the choice of neural network architecture), and scale much more efficiently than traditional grid-based representations with increasing resolution [1, 4, 15, 23, 28, 29, 34].

Typically, the field quantities are not themselves directly observable, but partial sensor observations are available. The function relating the field quantities to sensor observations is called the forward map. The most common algorithm for training neural fields is thus follows the following paradigm [37]: first, coordinates are sampled and passed through a neural network to predict the corresponding field quantities. Then the forward map maps the field quantities to the sensor domain, where we can calculate a reconstruction loss between the neural network's predictions and our ground-truth partial observations in the sensor domain. This reconstruction loss is then used to supervise the training of the neural field (see Figure 1).

However, one of the major drawbacks of this neural field training paradigm is that it is computationally expensive for a large dataset of signals, since a separate neural network must be trained from scratch for each signal, requiring a large amount of memory and computation [20], which we will henceforth refer to as the neural field generalization problem. Previous approaches to this problem include conditioning, hypernetworks [3], and gradient-based meta-learning [20, 35]. The first two approaches [11, 31, 32] seek to directly learn some or all of the parameters of a neural field. The third approach views the problem as a meta-learning problem, where each individual signal is considered a different task [20, 31, 35], and apply standard gradient-based meta-learning algorithms such as MAML [9] or Reptile [26] to find a parameter initialization that can be quickly fine-tuned for any specific signal. Previous results are mixed: [31] finds that gradient-based meta-learning can outperform concatenation and MLP hypernetworks, whereas [3] finds that hypernetworks with a vision transformer [6] encoder can outperform gradient-based meta-learning [35] on tasks where the available partial observations are 2D images.

However, gradient based meta-learning suffers from many problems, such as underfitting on large datasets and hyperparameter sensitivity. Recent research [12] finds that neural processes, another meta-learning framework, are more flexible and efficient for complex visual regression tasks, which we hypothesize will carry over to the neural

field domain. Additionally, one of our key observations is that the encoder-decoder structure of neural processes is an extension of the hypernetwork approach, making a neural process-style framework a natural choice for the neural field generalization problem. However, the usual neural process framework can not be applied directly to most neural field tasks, since in most neural field tasks the training of the neural field is supervised by partial sensor observations, whereas neural processes generally assume that we have direct observations of the field available. Therefore, we propose a simple partially-observed framework to adapt the neural process framework to the neural field domain that incorporates the forward map relating field observations to sensor observations.

In this work, we propose neural processes as an alternative to the traditional gradient-based meta-learning and hypernetwork approaches for neural field generalization. Our contributions are as follows:

1. We propose to use neural processes to tackle the neural field generalization problem. In particular, we adapt the traditional neural process framework to the common neural field setting where only partial observations of the field are available, and our framework is agnostic to the neural process architecture, allowing us to incorporate new advances in neural processes.

2. We show that neural processes can outperform both state-of-the-art gradient-based meta learning and state-of-the-art hypernetwork approaches on typical neural field generalization benchmarks, including 2D image regression and completion [3, 35], CT reconstruction from sparse views [35], and recovering 3D shapes from 2D image observations [3, 35].

2. Related Works

Fitting neural fields for multiple signals There are three main approaches to efficiently train neural networks for a large dataset of signals. The first two approaches are latent variable approaches where a latent representation is leveraged to produce the weights of the neural field. The latent code can be learned through an encoder or through auto-decoding [37]. The first approach is to concatenate a latent representation of the signal to the input coordinates. This has been shown to be equivalent to defining an affine function Ψ that maps the latent code to the biases of the first layer of the neural representation [8, 22, 31, 37]. Hypernetworks [5, 27, 32–34] generalize the concatenation approach by seeking to predict the complete set of weights of a neural field. The third major approach is to use gradient-based meta-learning in order to learn a prior over the signals, which is specialized to the final neural field for any particular signal by test-time optimization, and previous work has shown this approach to be superior to the hypernetwork

approach [31]. Recent work [3] has returned to the hypernetwork idea by proposing to use vision transformers [6] as the encoder of the hypernetwork, and show that it can outperform gradient-based meta-learning methods. However, this approach is limited to settings where the partial observations or sensor data are 2D images. Neural processes were included as a baseline to the proposed meta-learning method in [31], but otherwise has not been explored in the literature. We show that advances in neural process architectures [16, 19] allow their performance to exceed that of gradient-based meta-learning, and that this persists when moving to more complex forward maps.

Gradient-based Meta-learning Gradient-based meta-learning algorithms such as MAML [9] or Reptile [26] have been the dominant approach [20, 31, 35] used to efficiently train neural fields over many signals. This approach takes the view that for a dataset of neural fields, each neural field is a specialization of a meta-network [37]. [31] finds that using the gradient-based meta-learning of neural fields for signed-distance functions outperform concatenation, hypernetworks, and conditional neural processes. [35] applies standard gradient-based meta-learning algorithms (MAML and Reptile) to image regression, CT reconstruction, and view synthesis tasks. Our work instead proposes to use a new partially-observed NP framework to perform these tasks. [20] proposes using MAML in conjunction with pruning techniques to learn sparse neural representations. This method tackles a problem that is orthogonal to the generalization problem and could possibly be adapted to work in conjunction with our method.

Neural Processes Neural processes is a meta-learning framework that aims to learn a distribution over models by modelling a stochastic process. Architecturally, Neural processes are encoder-decoder models that are trained with probabilistic inference. There are two major classes of neural processes: conditional neural processes [7, 13, 16] and latent neural processes [10, 14, 19], which are latent variable models. Latent variable neural processes can offer better modelling at the cost of having intractable training objectives [7]. There are three major architectures for neural processes: MLP [13, 14], where both the encoder and decoder are MLPs and the an aggregate representation is produced via taking the mean, attention-based [19], where the aggregator is an attention module, and convolutional [10, 16], which uses a discrete version of the convolution operator in order to give the network translation equivariance. MLP-based models tend to underfit compared to the other two classes of models, and convolutional models tend to create smoother samples compared to attentive models [7, 19]. Neural processes have been used for low-dimensional regression [10, 13, 14, 16, 19], image comple-

tion [10, 13, 14, 16, 19], Bayesian optimization [10, 14], and visual regression tasks [12]. There is concurrent work [17] that proposes a Versatile Neural Process (VNP) architecture, based on attentive neural processes [19], for neural field generalization. In contrast, our work proposes a neural process framework that is both encoder and neural-field agnostic.

3. Method

In this section, we will first introduce necessary notation and give an overview of neural processes (Section 3.1). We then relate neural processes to neural fields and present our proposed neural process-based framework for neural field generalization (Section 3.2). Finally, we will describe the training of our neural process framework in Section 3.3.

Notation In the meta-learning setting, we have a set of tasks $\mathcal{M} = \{\mathcal{D}_i\}_{i=1}^{N_{\text{tasks}}}$. Each task $\mathcal{D}_i$ is a dataset with a training set, which we will call the **context set** and denote by $\mathcal{C}$, and a test set, which consists of **target inputs** (denoted $\mathbf{x}_{\mathcal{T}}$) and **target outputs** (denoted $\mathbf{y}_{\mathcal{T}}$). Our goal is to train a function $f(\cdot; \mathcal{C})$ such that given the context $\mathcal{C}$ of some new task, we get a predictor $f_{\mathcal{C}}(\cdot)$ that maps the target inputs to the target outputs with low error.

When discussing neural fields, we follow the terminology of [37]: data points or signals are viewed as a **field** $\Phi : \mathcal{X} \rightarrow \mathcal{Y}$. that assigns to all coordinates $\mathbf{x} \in \mathcal{X}$ some quantity $\mathbf{y} \in \mathcal{Y}$. A **neural field** parameterizes Φ with a neural network with parameters Θ, which we will denote Φ_{Θ}. The goal is to find Θ such that $\Phi_{\Theta}(\mathbf{x}) \approx \Phi(\mathbf{x}), \forall \mathbf{x} \in \mathcal{X}$." As discussed above, typically we only have partial observations from a sensor domain. The sensor data is related to the field quantities via a **forward map**, which we formally view as an operator $F(\Phi; \theta)$ that takes in a neural field Φ and sensor parameters θ and produces an output in the sensor domain, which is typically much lower-dimensional. Examples of sensor parameters θ are the projections angles in CT reconstruction or the camera parameters for NeRFs. We will assume that F is differentiable, allowing the use of gradient descent optimization.

3.1. Neural Processes

Neural processes [14] are a class of latent-variable neural network models that aim to learn distributions over functions. This makes them a natural meta-learning algorithm, as they can be viewed as a probabilistic model over task predictors. Neural processes seeks to parameterize the distribution $p(\mathbf{y}_{\mathcal{T}}|\mathbf{x}_{\mathcal{T}}, \mathcal{C})$ with a neural network. The basic structure of a neural process consists of three main components [14]: 1.) a permutation-invariant encoder that encodes the context set $\mathcal{C}$, 2.) an aggregator that produces a representation $\mathbf{z}$ of $\mathcal{C}$, and 3.) a decoder that takes as input both the representation $\mathbf{z}$ as well as the target inputs $\mathbf{x}_i$

and outputs $p(\mathbf{y}_{\mathcal{T}}|\mathbf{x}_{\mathcal{T}}, \mathcal{C})$. Training a NP proceeds as follows: first a dataset $\mathcal{D}_i \in \mathcal{M}$ is chosen, and then a context (training) set $\mathcal{C}$ is chosen from $\mathcal{D}_i$. $\mathcal{C}$ is then encoded into a global representation $\mathbf{z} = A(E(\mathbf{x}_{\mathcal{C}}, \mathbf{y}_{\mathcal{C}}))$. The decoder takes as input the test inputs $\mathbf{x}_{\mathcal{T}}$, is conditioned with $\mathbf{z}$, and outputs $p(\mathbf{y}_{\mathcal{T}}|\mathbf{x}_{\mathcal{T}}, \mathcal{C}) = D(\mathbf{x}_{\mathcal{T}}; \mathbf{z})$. The true log likelihood $\mathcal{L} = \log p_{\theta}(\mathbf{y}_{\mathcal{T}}|\mathbf{x}_{\mathcal{T}}; \mathcal{C})$ is then computed analytically or estimated. The gradient $\nabla \mathcal{L}$ is used to optimize the parameters of the NP via stochastic gradient optimization.

Conditional NPs vs Latent NPs There are two main ways to model the distribution $p(\mathbf{y}_{\mathcal{T}}|\mathbf{x}_{\mathcal{T}}, \mathcal{C})$: the first assumes conditional independence with respect to $\mathcal{C}$ and are known as **conditional neural processes (CNPs)**, and the second, which models the distribution with a probabilitic latent vector $\mathbf{z}$ in the same manner as a variational autoencoder (VAE), known as **latent neural processes (LNPs)**. Implementation-wise, CNPs resemble autoencoder whereas LNPs resemble VAEs. CNPs can be trained with maximum likelihood estimation, but for LNPs, as with VAEs, the true likelihood is intractable. To get around this, the true likelihood can be estimated either using Monte-Carlo integration, called NPML [10], or with variational inference (NPVI) [14]. LNPs trained with NPML often outperform those trained with NPVI but require more computation in the form of additional Monte-Carlo samples [10].

3.2. Our Partially-Observed Neural Process Framework

Connection between neural processes and neural fields To illustrate the connection between neural processes and neural fields, we examine one of the tasks frequently benchmarked in the neural process literature: image completion [10, 13, 14, 16, 19], where a subset of the pixels of an image and their color values are provided as the context inputs and outputs, and the goal is to predict the color values of all the pixels of the image. The target inputs are the coordinates of pixels, and the desired target outputs are the color values of those pixels. Our key observation is that this is a neural field task, with the forward map being the projection map onto the seen context pixels. Recall that in the neural process framework (Section 3.1), the decoder of the neural process takes as input coordinates as well as a representations of the context set, and returns their color values, so the neural process decoder is equivalent to a conditional neural field (Figure 1). In this example, the neural process encoder also encodes the partial sensor observations (output of the forward map).

Our neural process framework Based on the observations of the previous section, we propose our partially-observed neural process framework (PONP) for training neural fields (see Figure 1): 1.) a task-specific encoder,
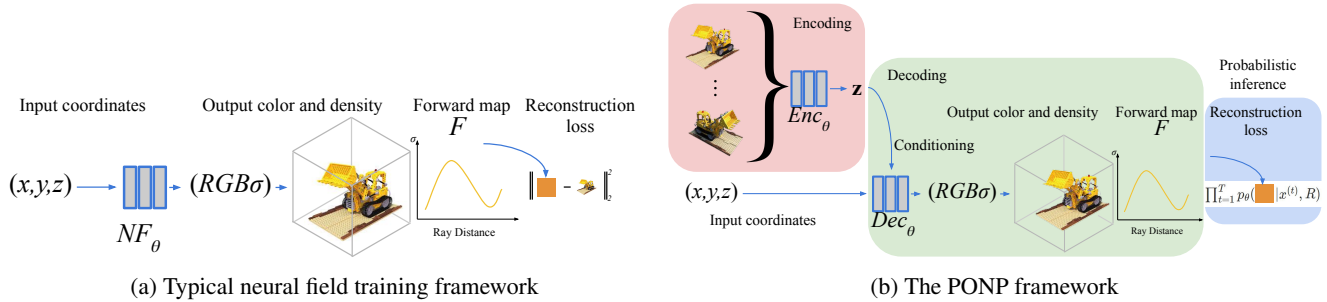
Figure 1: An example of PONP using NeRF [24]. (a) The typical neural field training framework with forward map F, which is essentially the bottom branch of (b). (b) Our proposed PONP framwork. First, ground truth partial sensor observations are encoded into a representation $\mathbf{z}$, possibly with sensor parameters (not shown). Second, $\mathbf{z}$ is used to condition a neural field decoder, and we generate a prediction for the field quantities and apply F. Third, we supervise the network with probabilistic inference (e.g. maximum likelihood). Unlike (a), the networks are shared across all signals (not shown).

which takes as input partial sensor observations and aggregates them into a representation $\mathbf{z}$, (red box) 2.) a decoder, which consists of a conditional neural field decoder, which is conditioned on $\mathbf{z}$ and designed to take as input field coordinates, and whose output is fed through the forward map F, producing a distribution (green box), and 3.) is trained with probabilistic inference (blue box). Note that with our framework we are able to leverage both existing neural process architectures (MLP, attention-based, and convolutional) and latent variables approaches (i.e., LNPs). Details on how we define the distribution are deferred to Sec. 3.3.

Conditioning the neural field   In order to produce different neural fields for different input signals, we need to condition our neural field decoder using the learned representation extracted from our context set. The simplest approach for doing this is to concatenate the representation with the input to the neural field [31]. However, since concatenation is equivalent to predicting the biases of the first layer of a neural network, concatenation is only suitable for datasets with very similar signals, such as the 2D CT reconstruction dataset [35]. To learn more diverse signals, more powerful approaches are required. Hypernetworks [18] generalize concatenation by having the encoder learn to produce the complete set of weights for the neural field, at the cost more parameters. This approach has been used by previous methods [3, 31, 33] as a way to learn more complex structures. Our choice of conditioning varies based on the task structure.

3.3. Probabilistic inference

Recall that neural processes are trained with a maximum likelihood objective. For our neural process-inspired framework, there are two choices of how to define the distribution: either over the field space or over the sensor space. As discussed previously, these spaces are typically not the same, as the sensor space is generally lower dimensional. Since we only have groundtruth observations in the sensor domain, defining our distribution over the field space has the disadvantage that we need to use the change-of-variables formula for probability distributions on the forward map F to compute the likelihood of groundtruth observations as required by maximum likelihood training. Since the forward map F may be very complex and thus not bijective or even injective, we cannot use the typical (bijective) change-of-variables formula but instead need to use a more general change-of-variables formula, which are much harder to compute or estimate. Defining the distribution on the sensor space avoids these problems.

Our framework   In our framework, we define our distribution as an independent Gaussian distribution at each coordinate in the sensor domain. Practically, this is implemented as follows: given a conditional neural field as the first stage of the decoder, we modify the last layer to have two heads, producing outputs M, Σ. The output of both heads are then fed separately through the forward map F to produce μ, σ, which become the mean and standard deviation of our distribution, respectively.

4. Experiments

4.1. Experimental Setup

We adapt the following experimental setup, which consists of two steps: 1.) meta-learning and 2.) test-time optimization, if necessary. Meta-learning consists of training a model on all signals in order to find a good initialization for the neural fields, and test-time optimization specializes the weights of the neural field to a particular signal. In order to make a fair comparison, we allow the same number of test-time optimization steps, if necessary, for all methods, and use comparable neural field architectures. For all tasks,

reconstruction quality is measured by peak signal-to-noise ratio (PSNR).

4.2. Baselines

We use the following baselines: standard neural field initialization, with test-time optimization, the method of [35], which applies a standard gradient-based meta-learning algorithm (either MAML [9] or Reptile [26]) and then fine-tunes with test-time optimization, and where applicable the Transformer INR method [3], which uses a vision transformer-based hypernetwork approach that uses a vision transformer to learn the weights of a neural field from 2D image sensor data. The use of a vision transformer encoder means that the Transformer INR method is not suitable for all tasks. Unlike the previous two baselines, this method does not require test-time optimization.

4.3. Tasks

In order to directly compare our neural process approach against previous methods, we test our method using four benchmarks, three of which were first proposed by [35]. More details about the tasks can be found in Appendix A.

Table 1: Forward maps F for our three tasks.

Task	F	Input & Output
Image	Masking	Pixel Location $\rightarrow$ RGB value
CT	Integral projection	2D image $\rightarrow$ 1D sinogram
NVS	Volume rendering	Radiance field $\rightarrow$ 2D image

2D image regression and completion In the image regression task, a neural field is trained to represent a 2D image by taking as input 2D pixel coordinates and outputting the corresponding RGB color values [35]. We use the CelebA [21] dataset, which consists of over 200K images of human faces. We downsample each image to 32×32. For the 2D image completion task, the groundtruth RGB values are known at some percentage of the pixels and the rest of the pixels are masked. As discussed earlier, this is a neural field task where the forward map is the indicator function on the pixels for which we have groundtruth (see Table 1).

2D CT reconstruction The CT reconstruction task is to reconstruct a 2D CT scan given a sparse set of 1D projections of the CT scan. The CT reconstruction dataset consists of 2048 randomly generated 256×256 Shepp-Logan phantoms [30]. The projections are generated from the CT scans by the forward map F given by 2D integral projections of a bundle of 256 parallel rays from a given angle (see Table 1). At test time, we are given 1, 2, 4, and 8 projections (views) with which to reconstruct the phantom. We are al-

Table 2: Comparison of neural field generalization methods on the 2D image regression task. The best PSNRs in the whole table are shown in bold and the second-best PSNRs are underlined. Parameters refers to the total number of parameters for the model, including the neural field itself. The TTO column is checked if test-time optimization was used.

MODEL	TTO	PSNR	PARAMS
RANDOM INIT.	✓	10.87	50K
REPTILE [35]	✓	25.27	50K
MAML [35]	✓	32.36	50K
TRANSFORMER INR [3]	✗	<u>48.36</u>	44.3M
PONP (OURS)	✗	**78.87**	340K

Table 3: Comparison of neural field generalization methods on the 2D image completion task. The best PSNRs in the whole table are shown in bold and the second-best PSNRs are underlined. Parameters refers to the total number of parameters for the model, including the neural field itself. The TTO column is checked if test-time optimization was used.

MODEL	TTO	PSNR	PARAMS
RANDOM INIT.	✓	10.23	50K
REPTILE [35]	✓	14.65	50K
MAML [35]	✓	13.35	50K
TRANSFORMER INR [3]	✗	<u>18.09</u>	44.3M
PONP (OURS)	✗	**23.24**	340K

lowed 50, 100, 1000, and 1000 test-time optimization steps, respectfully, for each of the four settings.

View synthesis for ShapeNet objects View synthesis is the task of generating a novel view of a 3D object or scene given some number of input views, and is typically tackled with neural radiance fields [3, 24]. Neural radiance fields learn the radiance of a 3D scene with a neural field and use the volume rendering forward map to generate 2D views from a given viewing angle (see Table 1). For the ShapeNet [2] view synthesis task, the data is created from objects taken from one of three ShapeNet classes: *chairs*, *cars*, and *lamps*. For each 3D object, 25 128×128 views are generated from viewpoints chosen randomly on a sphere. The goal is to generate given unseen views. The two different settings proposed in [35] are the multi-view setting, where we are allowed to train with the full 25 training views of each object, and the single-view setting, where we are only allowed to train with a single view per object. Since the Transformer INR baseline only reports results on the single-view and 2-shot (2 view) versions of the task, we only consider these tasks as well. We are allowed 1000-2000 test-time optimization steps, depending on the class of the object and the setting (multi-view or single-view).
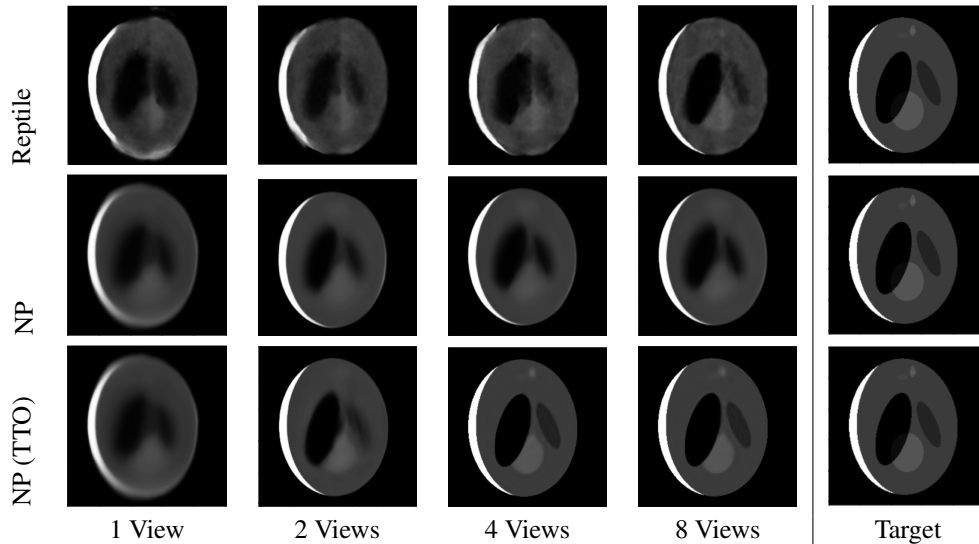
Figure 2: Examples of CT scan reconstruction from a sparse set of projections. The reconstructed scans learned by the neural process models are much sharper than their counterparts generated by gradient-based meta-learning.

Table 4: Comparison of initialization methods on the CT Reconstruction task. The best PSNRs in the whole table are shown in bold and the second-best PSNRs are underlined. The TTO column is checked if test-time optimization was used.

MODEL	TTO	1 VIEWS	2 VIEWS	4 VIEWS	8 VIEWS
RANDOM INIT. [35]	✓	13.63	14.15	16.31	21.49
REPTILE [35]	✓	15.09	18.70	22.00	<u>27.34</u>
PONP (OURS)	✗	<u>16.39</u>	<u>20.39</u>	<u>22.13</u>	22.60
PONP (OURS)	✓	**24.67**	**29.11**	**37.54**	**37.66**

4.4. Results

2D image regression and completion For both the 2D image regression and 2D image completion tasks, the field space and sensor space are the same, so our PONP framework coincides with the normal neural process framework. We use a convolutional CNP architecture [7], as the convolutional encoder is well-suited to handling 2D image data. For both the image regression and image completion tasks, we set the context and target inputs to be the 2D pixel coordinates, and the context and target outputs as RGB color values. For the image completion task, we randomly mask between 10-30% of the pixels. For the Transformer INR baseline, which requires 2D images as input, the masked pixels are assigned a value of $\mathbf{0}$.

Quantitative results for the image regression and completion tasks can be found in Tables 2 and 3, respectively. We find that even without the benefit of test-time optimization, our PONP framework greatly outperform gradient-based meta-learning methods and hypernetwork methods. In particular, PONP greatly outperforms the state-of-the-art Transformer INR hypernetwork method while using just a fraction of the parameters. We also observe that, as in [3], the Transformer INR method outperforms gradient-based meta-learning methods on both tasks. We also find that within gradient-based optimization methods, Reptile outperforms MAML in partially-observed settings, whereas the reverse is true in the fully-observed setting, as in [35]. Qualitative results can be found in the Appendix (Figure 5).

2D CT reconstruction For this task, we use an attention-based encoder in our PONP framework (see Section 4.5). We let the target inputs $\mathbf{x}_\mathcal{T}$ be the coordinates of the 2D image, and the target outputs $\mathbf{y}_\mathcal{T}$ be the corresponding volume densities. We also set the context output be the volume density, and we choose the context inputs to be the angles used to produce the projections, since the image coordinates do not add any extra information. We also drop the random Fourier features used by the neural field in [35], as we find that the performance of our method is much lower with Fourier features. For this task, we do not compare against the Transformer INR baseline [3], which is unsuitable to handling the 1D partial information.
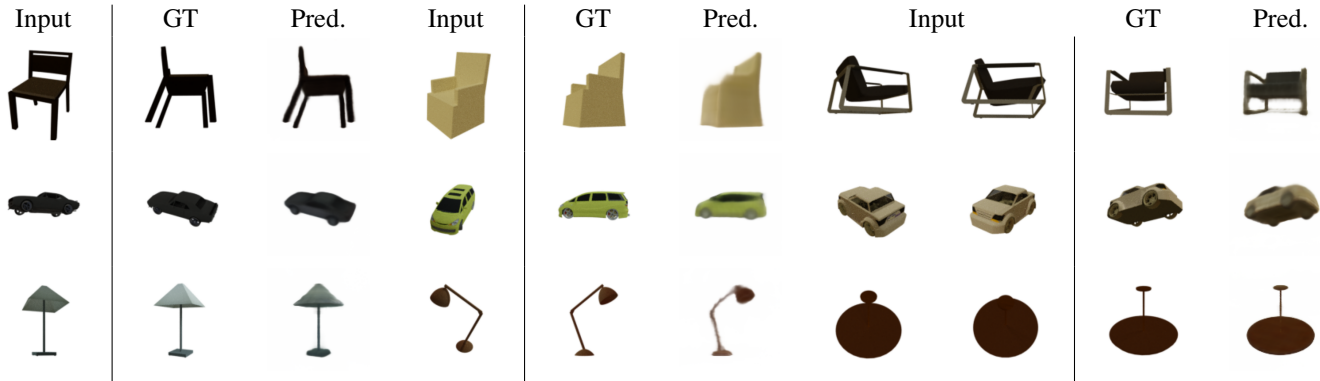
Figure 3: Qualitative examples of our PONP framework for the ShapeNet view synthesis experiment.

Table 5: Comparison of neural field generalization and initialization methods on the ShapeNet view synthesis task. The best PSNRs without test-time optimization and with test-time optimization are bolded.

MODEL	VIEWS	TTO	CHAIRS	CARS	LAMPS
RANDOM INIT. [35]	0	✓	12.49	11.45	15.47
MATCHED [35]	25	✓	16.40	22.39	20.79
SHUFFLED [35]	25	✓	10.76	11.30	13.88
REPTILE [35]	1	✓	16.54	22.10	20.95
REPTILE [35]	25	✓	18.85	22.80	22.35
TRANSFORMER INR [3]	1	✗	**19.66**	23.78	22.76
PONP (OURS)	1	✗	19.48	**24.17**	**22.78**
TRANSFORMER INR [3]	2	✗	21.10	25.45	23.11
PONP (OURS)	2	✗	**21.13**	**25.98**	**23.28**
TRANSFORMER INR [3]	1	✓	**20.56**	24.73	24.71
PONP (OURS)	1	✓	20.55	**24.99**	**24.86**
TRANSFORMER INR [3]	2	✓	23.59	27.13	27.01
PONP (OURS)	2	✓	**23.73**	**27.49**	**27.04**

Quantitative results can be found in Table 4 and qualitative results can be found in Figure 2. We observe that our PONP method significantly outperforms Reptile. Visually, we also observe that the predicted CT reconstructions are much sharper with our PONP method than with Reptile. We also observe that even without test-time optimization, our neural process method is able to outperform random initialization (i.e., training a neural field from scratch) and is able to outperform Reptile in the 1, 2, and 4 view settings. This shows that with similar amounts of test-time optimization, neural processes greatly outperform gradient-based meta-learning, and in the test-time optimization free setting neural processes are still competitive, especially in setting with low amounts of partial information, while requiring zero test-time computation.

Due to probabilistic nature of our method, we can use the standard deviations learned post-forward map as a mea-

sure of uncertainty. The mean and standard deviation of reconstructions from 100 latent **z** samples from our AttnLNP CT model with 1 input view. Our model shows uncertainty about the boundaries of the different regions of the CT scan. See Figure 4.

View synthesis for ShapeNet objects For this task, we also report the Matched and Shuffled baselines of [35]. Matched is an initialization that is trained from scratch to match the output of the meta-learned initialization, and Shuffled is an initialization that permutes the weights in the meta-learned initialization [3, 35]. Since a key part of instantiating our PONP framework for a given task is finding a suitable encoder, we adapt the vision transformer [6] encoder of the Transformer INR method. This makes sense since the vision transformer encoder can be thought of as an attention-based neural process encoder that takes is de-

Table 6: Comparison of different neural process architectures for our PONP method on the CT reconstruction task. In each of the second and third sections of the table (corresponding to methods not using test-time optimization and methods using test-time optimization, respectively), the best performing PSNRs are bolded and the second-best PSNRs are underlined. The TTO column is checked if test-time optimization was used.

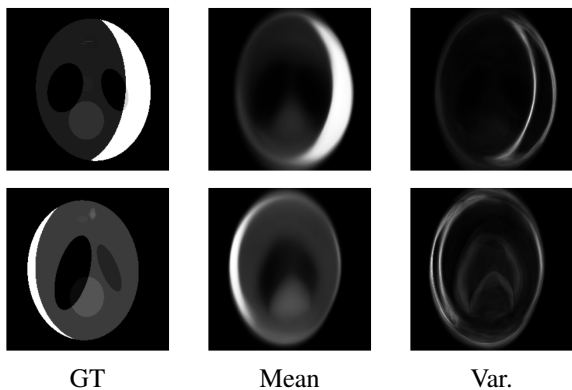| ARCHITECTURE	TTO	1 VIEWS	2 VIEWS	4 VIEWS	8 VIEWS
RANDOM INIT. [35]	✓	13.63	14.15	16.31	21.49
CNP [13]	✗	**16.39**	**20.39**	22.14	22.60
LNP [14]	✗	<u>16.28</u>	19.43	21.02	21.34
ATTNCNP [7]	✗	16.24	20.12	<u>22.60</u>	<u>23.60</u>
ATTNLNP [19]	✗	16.26	<u>20.27</u>	**23.07**	**23.88**
REPTILE [35]	✓	15.09	18.70	22.00	27.34
CNP [13]	✓	**24.67**	**29.11**	<u>37.54</u>	<u>37.66</u>
LNP [14]	✓	<u>24.66</u>	<u>26.92</u>	29.22	29.34
ATTNCNP [7]	✓	18.32	25.14	**38.18**	**38.24**
ATTNLNP [19]	✓	18.10	23.42	32.47	32.94

GT   Mean   Var.

Figure 4: Visualizing uncertainty prediction by PONP.

signed for the 2D image sensor observations and conditions a neural field via hypernetwork. To fully incorporate the vision transformer encoder into our framework, we modify the neural field to produce the mean and variance of a Gaussian distribution and apply the CNP loss (see Sec. 3.2).

Quantitative results can be found in Table 5 and qualitative results can be found in Figure 3. Noting that PSNR is a on a log scale, we find that our method is comparable to or slightly outperforms the previous state-of-the-art Transformer INR baseline, while being more widely applicable. Our method also slightly improves over the Transformer INR method when the number of views increases to 2. We also find that even with only 1 view, our method clearly outperforms gradient-based meta-learning methods, even without test-time optimization.

4.5. Ablations

Architecture In the absence of a specially-designed encoder, our PONP framework can leverage existing neural process architectures. We examine the performance of different neural process architectures on the CT reconstruction task. We compare among the CNP [13], LNP [14], AttnCNP [7], and AttnLNP [19] architectures. Quantitative results on the CT reconstruction experiment can be found in Table 6. We find that the more recent attention-based architectures tend to outperform older MLP-based architectures when no test-time optimization is allowed. With test-time optimization, all neural process architectures outperform the Reptile and random initialization baselines. Unexpectedly, CNP architectures outperforms LNP architectures by a wide margin, which is unexpected as both the pre-test-time optimization results are similar and and LNP models do not achieve significantly better likelihoods during training. We hypothesize that this is due to sampling from the latent distribution required by LNP architectures. Without test-time optimization, we find that all neural process architectures outperform the Reptile baseline when there are a low amount of projections and test-time optimization steps, with the best model (AttnLNP) outperforming all baselines in the 1, 2, and 4 view settings.

5. Conclusion

In summary, we introduce a new framework for neural field generalization that is inspired by neural processes. We show that our framework outperforms both gradient-based meta-learning and hypernetwork approaches for a variety of different neural field problems. With this work, we demonstrate the promise of neural process-based algorithms for efficiently learning neural fields. In this direction, avenues for future investigation include quantifying the predictive uncertainty learned by neural process algorithms and applications of our method to applications such as biomedical imaging and reconstruction.

References

[1] Matan Atzmon and Yaron Lipman. Sal: Sign agnostic learning of shapes from raw data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2565–2574, 2020. 1

[2] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 5

[3] Yinbo Chen and Xiaolong Wang. Transformers as meta-learners for implicit neural representations. In *European Conference on Computer Vision*, pages 170–187. Springer, 2022. 1, 2, 4, 5, 6, 7, 11, 12

[4] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5939–5948, 2019. 1

[5] Pei-Ze Chiang, Meng-Shiun Tsai, Hung-Yu Tseng, Wei-Sheng Lai, and Wei-Chen Chiu. Stylizing 3d scene via implicit representation and hypernetwork. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1475–1484, 2022. 2

[6] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 1, 2, 7

[7] Yann Dubois, Jonathan Gordon, and Andrew YK Foong. Neural process family. http://yanndubs.github.io/Neural-Process-Family/, September 2020. 2, 6, 8, 11

[8] Vincent Dumoulin, Ethan Perez, Nathan Schucher, Florian Strub, Harm de Vries, Aaron Courville, and Yoshua Bengio. Feature-wise transformations. *Distill*, 3(7):e11, 2018. 2

[9] 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. 1, 2, 5

[10] Andrew Foong, Wessel Bruinsma, Jonathan Gordon, Yann Dubois, James Requeima, and Richard Turner. Meta-learning stationary stochastic process prediction with convolutional neural processes. *Advances in Neural Information Processing Systems*, 33:8284–8295, 2020. 2, 3, 11

[11] Tomer Galanti and Lior Wolf. On the modularity of hypernetworks. *Advances in Neural Information Processing Systems*, 33:10409–10419, 2020. 1

[12] Ning Gao, Hanna Ziesche, Ngo Anh Vien, Michael Volpp, and Gerhard Neumann. What matters for meta-learning vision regression tasks? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14776–14786, 2022. 1, 3

[13] Marta Garnelo, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Rezende, and SM Ali Eslami. Conditional neural processes. In *International Conference on Machine Learning*, pages 1704–1713. PMLR, 2018. 2, 3, 8, 11

[14] Marta Garnelo, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J Rezende, SM Eslami, and Yee Whye Teh. Neural processes. *arXiv preprint arXiv:1807.01622*, 2018. 2, 3, 8, 11

[15] Kyle Genova, Forrester Cole, Daniel Vlasic, Aaron Sarna, William T Freeman, and Thomas Funkhouser. Learning shape templates with structured implicit functions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7154–7164, 2019. 1

[16] Jonathan Gordon, Wessel P Bruinsma, Andrew YK Foong, James Requeima, Yann Dubois, and Richard E Turner. Convolutional conditional neural processes. *arXiv preprint arXiv:1910.13556*, 2019. 2, 3, 11

[17] Zongyu Guo, Cuiling Lan, Zhizheng Zhang, Zhibo Chen, and Yan Lu. Versatile neural processes for learning implicit neural representations. *arXiv preprint arXiv:2301.08883*, 2023. 3

[18] David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016. 4

[19] Hyunjik Kim, Andriy Mnih, Jonathan Schwarz, Marta Garnelo, Ali Eslami, Dan Rosenbaum, Oriol Vinyals, and Yee Whye Teh. Attentive neural processes. *arXiv preprint arXiv:1901.05761*, 2019. 2, 3, 8, 11

[20] Jaeho Lee, Jihoon Tack, Namhoon Lee, and Jinwoo Shin. Meta-learning sparse implicit neural representations. *Advances in Neural Information Processing Systems*, 34:11769–11780, 2021. 1, 2

[21] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision*, pages 3730–3738, 2015. 5

[22] Ishit Mehta, Michaël Gharbi, Connelly Barnes, Eli Shechtman, Ravi Ramamoorthi, and Manmohan Chandraker. Modulated periodic activations for generalizable local functional representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14214–14223, 2021. 2

[23] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4460–4470, 2019. 1

[24] 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. In *European conference on computer vision*, pages 405–421. Springer, 2020. 4, 5

[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

[26] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018. 1, 2, 5

[27] Yuval Nirkin, Lior Wolf, and Tal Hassner. Hyperseg: Patch-wise hypernetwork for real-time semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4061–4070, 2021. 2

[28] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 165–174, 2019. 1

[29] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *European Conference on Computer Vision*, pages 523–540. Springer, 2020. 1

[30] Lawrence A Shepp and Benjamin F Logan. The fourier reconstruction of a head section. *IEEE Transactions on nuclear science*, 21(3):21–43, 1974. 5

[31] Vincent Sitzmann, Eric Chan, Richard Tucker, Noah Snavely, and Gordon Wetzstein. Metasdf: Meta-learning signed distance functions. *Advances in Neural Information Processing Systems*, 33:10136–10147, 2020. 1, 2, 4, 11

[32] 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

[33] Vincent Sitzmann, Semon Rezchikov, Bill Freeman, Josh Tenenbaum, and Fredo Durand. Light field networks: Neural scene representations with single-evaluation rendering. *Advances in Neural Information Processing Systems*, 34:19313–19325, 2021. 4

[34] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. *Advances in Neural Information Processing Systems*, 32, 2019. 1, 2, 11

[35] 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. 1, 2, 4, 5, 6, 7, 8, 11, 12

[36] Kuan-Chieh Wang, Zhenzhen Weng, Maria Xenochristou, João Pedro Araújo, Jeffrey Gu, Karen Liu, and Serena Yeung. Nemo: Learning 3d neural motion fields from multiple video instances of the same action. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22129–22138, 2023. 1

[37] Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. Neural fields in visual computing and beyond. In *Computer Graphics Forum*, volume 41, pages 641–676. Wiley Online Library, 2022. 1, 2, 3