Spectral Convolutional Conditional Neural Processes

Peiman Mohseni
Texas A&M University
peiman.mohseni@tamu.edu

Nick Duffield
Texas A&M University
duffieldng@tamu.edu

Abstract

Neural processes (NPs) are probabilistic meta-learning models that map sets of observations to the corresponding posterior predictive distributions, enabling inference at arbitrary domain points. Their ability to handle variable-sized collections of unstructured observations, combined with simple maximum-likelihood training and uncertainty-aware predictions, makes them well-suited for modeling data over continuous domains. Since their introduction, several variants have been proposed. Early approaches typically represented observed data using finite-dimensional summary embeddings obtained through aggregation schemes such as mean pooling. However, this strategy fundamentally mismatches the infinite-dimensional nature of the generative processes that NPs aim to capture. Convolutional conditional neural processes (ConvCNPs) address this limitation by constructing infinite-dimensional functional embeddings processed through convolutional neural networks (CNNs) to enforce translation equivariance. Yet CNNs with local spatial kernels struggle to capture long-range dependencies without resorting to large kernels, which impose significant computational costs. To overcome this limitation, we propose the Spectral ConvCNP (SConvCNP), which performs global convolution in the frequency domain. Inspired by Fourier neural operators (FNOs) for learning solution operators of partial differential equations (PDEs), our approach directly parameterizes convolution kernels in the frequency domain, leveraging the relatively compact yet global Fourier representation of many natural signals. We validate the effectiveness of SConvCNP on both synthetic and real-world datasets, demonstrating how ideas from operator learning can advance the capabilities of NPs.

1 Introduction

Stochastic processes offer a mathematical framework for modeling systems that evolve with inherent randomness over continuous domains such as time and space. They underpin a wide range of scientific applications—from spatio-temporal climate dynamics to biological and physical systems—thereby motivating the development of machine learning methods that can learn from data generated by such phenomena [Mathieu et al., 2021, Vaughan et al., 2021, Allen et al., 2025, Ashman et al., 2025, Dupont et al., 2021]. Among classical approaches, Gaussian processes (GPs; Rasmussen et al., 2006) provide a Bayesian framework with closed-form inference and uncertainty quantification. However, their cubic computational cost from matrix inversion and the difficulty of specifying suitable kernels—especially in high-dimensional settings—limit their scalability.

Motivated by the success of deep neural networks in large-scale function approximation, neural network-based alternatives have emerged. Neural processes (NPs; Garnelo et al., 2018a,b) exemplify this paradigm, combining ideas from GPs and deep learning within a meta-learning framework. By exposing the model to multiple realizations of an underlying stochastic process, each treated as a distinct task, NPs learn shared structure across tasks to parameterize a neural mapping that *directly* gives an approximation of the corresponding posterior predictive distribution [Bruinsma, 2024]. Once trained, the model enables efficient probabilistic predictions on new tasks without further training.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

Since the introduction of conditional neural processes (CNPs, Garnelo et al. [2018a]) as the first class within the NPs family [Jha et al., 2022], a wide range of extensions and variants have been proposed to enhance their effectiveness. One active line of research introduces explicit inductive biases into CNPs to encode the symmetries commonly present in scientific domains [Gordon et al., 2019, Kawano et al., 2021, Holderrieth et al., 2021, Huang et al., 2023, Ashman et al., 2024a,b]. Another major direction seeks to move beyond the mean-field factorized Gaussian predictive distributions to which CNPs are limited. A popular approach augments CNPs with stochastic latent variables, giving rise to the family of latent neural processes (LNPs; Garnelo et al. [2018b], Louizos et al. [2019], Wang and Van Hoof [2020], Foong et al. [2020], Lee et al. [2020], Volpp et al. [2021], Wang et al. [2022], Wang and van Hoof [2022], Kim et al. [2022], Jung and Park [2023], Lee et al. [2023], Xu et al. [2023]). Alternative approaches explore autoregressive prediction schemes [Bruinsma et al., 2023, Nguyen and Grover, 2022], Gaussian predictive distributions with non-diagonal covariances [Bruinsma et al., 2021, Markou et al., 2022], and quantile-based parameterizations of the predictive distribution [Mohseni et al., 2023].

This work focuses on CNPs, particularly convolutional CNPs (ConvCNPs; Gordon et al. [2019]), which were the first to endow NPs with translation equivariance. ConvCNPs introduce the convolutional deep set construction which characterizes a broad class of translation-equivariant mappings over finite, potentially unstructured sets of observations as a composition of a functional embedding with a translation-equivariant operator, typically realized through convolutional neural networks (CNNs; Fukushima [1980], LeCun et al. [1989, 1998]).

Despite their effectiveness, ConvCNPs can struggle to aggregate information from observations spread across large spatial domains—a challenge that becomes particularly pronounced in sparse data regimes. This limitation stems from their reliance on local convolutional kernels with small receptive fields, which hampers their ability to model long-range dependencies. A natural remedy is to enlarge the kernel size to extend the receptive field; however, this approach rapidly increases the number of model parameters and computational cost [Romero et al., 2021]. Alternatively, transformer-based architectures can capture long-range interactions but incur quadratic, rather than linear, computational complexity in the number of observations [Vaswani et al., 2017, Nguyen and Grover, 2022].

In this work, we propose an alternative approach that represents functions in the frequency domain, inspired by the well-established observation that many natural processes exhibit energy concentration in low-frequency bands [Field, 1987, Ruderman and Bialek, 1993, Wainwright and Simoncelli, 1999]. This characteristic allows for efficient approximation using only a subset of dominant spectral coefficients, enabling tractable computations while preserving the signal's global structure. By parameterizing convolution kernels directly in the Fourier domain over a finite set of frequencies and leveraging the convolution theorem, our method attains large effective receptive fields without incurring prohibitive computational costs.

While spectral methods have been extensively studied in neural operator learning for partial differential equations (PDEs) (e.g., [Li et al., 2020a, Gupta et al., 2021, Kovachki et al., 2021, 2023, Helwig et al., 2023]), their application within NPs framework remains relatively unexplored. To bridge this gap, we propose the Spectral Convolutional Conditional Neural Process (SConvCNP)—a model that adopts Fourier neural operators (FNOs; Li et al. [2020a]) to realize global convolution while maintaining computational efficiency. Across a suite of synthetic and real-world benchmarks, SConvCNP performs competitively with state-of-the-art baselines, illustrating how ideas from neural operators can enhance the flexibility and performance of NPs.

2 Preliminaries

2.1 Fourier Neural Operators

Neural operators [Chen and Chen, 1995, Li et al., 2020a, Kovachki et al., 2023] are neural network architectures designed to learn mappings between *function spaces* rather than finite-dimensional vectors. Much like conventional feed-forward networks, they consist of stacked layers that alternate between operator-based transformations and pointwise nonlinearities. The common transformation choice is an integral operator with kernel $\kappa : \mathcal{X} \times \mathcal{X} \to \mathcal{Y}$, acting on an input function $v : \mathcal{X} \to \mathcal{Y}$ as

$$\mathcal{K}[v](x) = \int \kappa(x, s)\, v(s)\, ds,$$

where $\mathcal{X} = \mathbb{R}^{d_x}$ and $\mathcal{Y} = \mathbb{R}^{d_y}$ are Euclidean spaces with dimensions d_x and d_y, respectively. While this work focuses on linear operators, nonlinear formulations have also been explored, including continuous formulations of softmax attention [Ashman et al., 2024a, Calvello et al., 2024]. When the kernel is *stationary* ($\kappa(x, s) = \kappa(x - s)$), the operator simplifies to a convolution, $\mathcal{K}[v] = \kappa * v$, connecting neural operators to CNNs, where κ is parameterized by learnable weights.

Typically, convolutional kernels are spatially local and possess a limited receptive field [Luo et al., 2016, Peng et al., 2017, Wang et al., 2018]. Consequently, modeling long-range dependencies requires increasingly large kernels, which substantially inflates the parameter count. The Fourier Neural Operator (FNO, Li et al. [2020a]) overcomes this limitation by exploiting the convolution theorem [Bracewell and Kahn, 1966, Oppenheim, 1999], which re-expresses convolution in the frequency domain as:

$$\mathcal{K}[v](x) = \mathcal{F}^{-1}\Big[\mathcal{F}[\kappa](\xi) \cdot \mathcal{F}[v](\xi)\Big](x), \tag{1}$$

where $\mathcal{F}$ and $\mathcal{F}^{-1}$ denote the Fourier and inverse Fourier transforms, respectively. Rather than parameterizing the kernel κ in the spatial domain, the FNO learns its representation directly in the Fourier domain. When the input function v is approximately band-limited—i.e., its spectrum $\mathcal{F}[v](\xi)$ carries negligible energy for frequencies $|\xi| > \xi_0$—the high-frequency components can be truncated with minimal loss of information. This property, observed in many natural signals [Field, 1987, Ruderman and Bialek, 1993, Wainwright and Simoncelli, 1999], allows setting $\mathcal{F}[\kappa](\xi) = 0$ outside the retained frequency band.

In practice, functions are accessible only through discrete samples, so transitions between spatial and frequency domains rely on the discrete Fourier transform (DFT). Given samples of v on a *uniform* grid $\mathcal{G} \subset \mathcal{X}$, the FNO applies the DFT, $\hat{\mathcal{F}}\{(v(x_g))_{x_g \in \mathcal{G}}\}$, computed efficiently via the fast Fourier transform (FFT; Cooley and Tukey [1965], Frigo and Johnson [2005]). The resulting spectrum is then truncated to a finite set of frequency modes $\hat{\Xi} \subset \mathbb{R}^{d_x}$, which are assumed to capture the dominant energy of the signal v. For each retained frequency $\hat{\xi} \in \hat{\Xi}$, the Fourier representation of the kernel, $\mathcal{F}[\kappa](\hat{\xi})$, is parameterized by learnable complex weights. In the simplest case, this takes the form $\mathcal{F}[\kappa](\hat{\xi}) = w_{\hat{\xi}} \in \mathbb{C}^{d_y}$, with $\mathbb{C}$ denoting the complex field[1]. Note that this formulation implicitly imposes periodicity in the physical domain, since the kernel is represented using a discrete set of harmonics (equivalently, a Dirac comb in frequency space). After pointwise multiplication in the frequency domain, an inverse FFT maps the result back to the physical domain, producing the operator output.

2.2 Neural Processes

Let $\mathcal{P}(\mathcal{Y}^{\mathcal{X}})$ denote the set of probability measures over functions $f : \mathcal{X} \to \mathcal{Y}$. We assume a data-generating process $\mu \in \mathcal{P}(\mathcal{Y}^{\mathcal{X}})$, from which functions are sampled. A task $\mathcal{D}$ consists of a finite collection of noisy input–output observations drawn from a realization $f \sim \mu$, partitioned into a context set and a query[2] set [Bruinsma, 2024, Ashman et al., 2024a,b]:

$$\mathcal{D} = (\mathcal{D}_c, \mathcal{D}_q) = \Big(\{(x_{c,k}, y_{c,k})\}_{k \in \mathcal{I}_c},\ \{(x_{q,k}, y_{q,k})\}_{k \in \mathcal{I}_q}\Big), \tag{2}$$

where

$$y_{c,k} = f(x_{c,k}) + \epsilon_{c,k}, \quad y_{q,k} = f(x_{q,k}) + \epsilon_{q,k}, \quad \epsilon_{c,k}, \epsilon_{q,k} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma_0^2).$$

Here, $\sigma_0 > 0$ denotes the observation noise scale, and $\mathcal{I}_c, \mathcal{I}_q \subset \mathbb{N}$ index the context and query sets, respectively. Neural processes (NPs; [Garnelo et al., 2018a,b]) constitute a class of models that employ neural networks to learn a mapping $\eta : \bigcup_{k=0}^{\infty} (\mathcal{X} \times \mathcal{Y})^k \to \mathcal{P}(\mathcal{Y}^{\mathcal{X}})$, which takes a finite (possibly empty) context set $\mathcal{D}_c$ and gives a stochastic process intended to approximate the Bayesian posterior induced by μ. This approximation is constructed implicitly through its finite-dimensional marginals[Garnelo et al., 2018b, Bruinsma et al., 2021, Bruinsma, 2024, Mathieu et al., 2023]. Denote $\boldsymbol{x}_q = (x_{q,k})_{k \in \mathcal{I}_q}$ and $\boldsymbol{y}_q = (y_{q,k})_{k \in \mathcal{I}_q}$. The NP approximation can then be expressed as

$$q(\boldsymbol{y}_q \mid \boldsymbol{x}_q, \mathcal{D}_c) \approx p(\boldsymbol{y}_q \mid \boldsymbol{x}_q, , \mathcal{D}_c)\,,$$

[1] More generally, a matrix-valued parameterization $\mathcal{F}[\kappa](\hat{\xi}) = w_{\hat{\xi}} \in \mathbb{C}^{c_{\text{out}} \times d_y}$ is used, where c_{out} is the number of output channels, allowing joint mixing of input channels and projection into a space of potentially different dimensionality.

[2] The NPs literature conventionally uses the term "target points." To avoid notational ambiguity with time indices, we instead use "query points."

where $q(\boldsymbol{y}_q \mid \boldsymbol{x}_q, \mathcal{D}_c)$ and $p(\boldsymbol{y}_q \mid \boldsymbol{x}_q, \mathcal{D}_c)$ are the finite-dimensional distributions—evaluated at the query inputs $\boldsymbol{x}_q$—of the learned process $\eta[\mathcal{D}_c]$ and of the posterior process induced by the true data-generating measure μ, respectively.

In this work, we focus on conditional NPs (CNPs, Garnelo et al. [2018a]), which restrict q to the family of mean-field Gaussians, i.e. $q\big(\boldsymbol{y}_q \mid \boldsymbol{x}_q, \mathcal{D}_c\big) = \prod_{k \in \mathcal{I}_q} q(y_{q,k} \mid x_{q,k}, \mathcal{D}_c)$. In general, CNPs parameterization of $q(y_{q,k} \mid x_{q,k}, \mathcal{D}_c)$ can be abstracted as a two-stage encoder-decoder pipeline [Bruinsma, 2024, Ashman et al., 2024a, 2025]. The encoder $\varphi_e : \mathcal{X} \times \bigcup_{k=0}^{\infty} (\mathcal{X} \times \mathcal{Y})^k \to \mathcal{H}$ maps the query location and context set into a latent representation, while the decoder $\varphi_d : \mathcal{X} \times \mathcal{H} \to \Theta$ transforms the query location and latent code into parameters $\theta \in \Theta$ of the predictive distribution $q(y_{q,k} \mid x_{q,k}, \mathcal{D}_c)$.

The vanilla CNP encoder summarizes the context set $\mathcal{D}_c$ using a *permutation-invariant* architecture [Qi et al., 2017, Zaheer et al., 2017]. Each context pair $(x_{c,k}, y_{c,k}) \in \mathcal{D}_c$ is independently mapped by a shared feedforward network to a *finite-dimensional* embedding $\varepsilon_{c,k}$, which are subsequently aggregated—typically via mean pooling—into a single representation ε_c. Notably, this encoding is independent of the query location (i.e., φ_e does not condition on the query). The decoder then fuses ε_c with the query input through another feedforward network to produce the parameters of the Gaussian predictive distribution.

Although sum-pooling aggregation provides universal approximation guarantees [Zaheer et al., 2017, Bloem-Reddy and Teh, 2020], NPs employing such mechanisms often exhibit underfitting in practice [Kim et al., 2019]. Prior works have partly attributed this phenomena to two primary factors[Xu et al., 2020]: (1) the limitation of summaries with *prespecified finite dimensionality* in representing context sets of arbitrary size [Wagstaff et al., 2019], and (2) the shortcomings of simple sum or mean pooling operations to effectively capture rich interactions between context and query points[Xu et al., 2020, Nguyen and Grover, 2022].

Given that the problems tackled by NPs are inherently *functional* in nature, it is natural to seek embeddings that also exhibit a functional character. In this spirit, Gordon et al. [2019] formalize a general framework for constructing translation-equivariant prediction maps over sets, satisfying

$$\eta\Big[\big\{(x_c + \tau, y_c) \mid (x_c, y_c) \in \mathcal{D}_c\big\}\Big]\big((x_{q,k})_{k \in \mathcal{I}_q}\big) \;=\; \eta\big[\mathcal{D}_c\big]\big((x_{q,k} - \tau)_{k \in \mathcal{I}_q}\big),$$

for every $\tau \in \mathcal{X}$. They show that a broad family of such maps is expressed as $\eta[\mathcal{D}_c] = \varphi_d\big[\varphi_e\left[\mathcal{D}_c\right]\big]$, where the functional embedding is defined by

$$\varphi_e[\mathcal{D}_c](\cdot) = \sum_{(x_c,\, y_c) \in \mathcal{D}_c} \phi(y_c)\, \psi_e(\cdot - x_c), \tag{3}$$

and $\varphi_d : \mathcal{H} \to C_b(\mathcal{X}, \mathcal{Y})$ is a translation-equivariant decoder operating on a function space $\mathcal{H}$. Here, $C_b(\mathcal{X}, \mathcal{Y})$ denotes the space of bounded continuous functions from $\mathcal{X}$ to $\mathcal{Y}$; $\phi(y) = (1, y)$[3] and $\psi_e : \mathcal{X} \to \mathbb{R}$ is a continuous, strictly positive-definite kernel typically chosen to be a Gaussian.

In implementing the ConvCNP, $\varphi_e[\mathcal{D}_c]$ is first evaluated on a *uniform* grid $\mathcal{G} \subset \mathcal{X}$ that spans the *joint* support of both the context and query locations. This yields a discretized representation $(\varphi_e[\mathcal{D}_c](x_g))_{x_g \in \mathcal{G}}$, which is then passed through φ_d to produce $\big(\varphi_d\big[\varphi_e[\mathcal{D}_c]\big](x_g)\big)_{x_g \in \mathcal{G}}$. Because the query inputs $(x_{q,k})_{k \in \mathcal{I}_q}$ may not lie exactly on the grid $\mathcal{G}$, ConvCNP employs an additional interpolation step to obtain predictions at query locations. Specifically

$$(\theta_{q,k})_{k \in \mathcal{I}_q} = \left(\sum_{x_g \in \mathcal{G}} \varphi_d\big[\varphi_e[\mathcal{D}_c]\big](x_g)\, \psi_d(x_{q,k} - x_g) \right)_{k \in \mathcal{I}_q}, \tag{4}$$

where ψ_d is another strictly positive kernel. Note that this step can be viewed as part of the decoder itself, thereby preserving the overall encoder–decoder abstraction discussed earlier.

3 Spectral Convolutional Conditional Neural Process

The decoder φ_d in ConvCNP is typically parameterized using standard CNNs such as U-Net [Ronneberger et al., 2015] or ResNet [He et al., 2016]. These architectures employ discrete convolutional

[3]For most applications, this representation suffices. Generally, $\phi(y) = (1, y, \ldots, y^M)$, where M is the multiplicity of repeated inputs. See Gordon et al. [2019] and Bruinsma [2024] for further discussion.

kernels—finite sets of learnable parameters that define localized filters operating over neighboring grid points. The kernel size, fixed *a priori*, determines the receptive field of each convolution [Ding et al., 2022] and is generally much smaller than the overall spatial extent of the input signals [Romero et al., 2021, Knigge et al., 2023].

This locality constraint limits the model's capacity to capture long-range dependencies and to integrate information from observations distributed across large domains [Peng et al., 2017, Wang et al., 2018, Ramachandran et al., 2019, Wang et al., 2020]. The issue becomes particularly pronounced when handling sparse or irregularly sampled data, where effective global reasoning cannot emerge solely from local convolution operations. While increasing the kernel size could enlarge the receptive field, it leads to a rapid growth in both parameter count and computational cost. Transformer-based architectures offer an alternative by enabling global interactions [Vaswani et al., 2017]; however, they typically incur quadratic rather than linear complexity in the number of inputs [Nguyen and Grover, 2022, Feng et al., 2022, Ashman et al., 2024a, 2025].

To overcome this limitation without relying on prohibitively large filters or expensive transformers, we exploit the Fourier representation of signals. This is motivated by the well-established observation that many natural signals are approximately band-limited (see Section 2.1), implying that their Fourier representation offers a more compact encoding *relative* to its physical-domain counterpart. Concretely, we instantiate φ_d via spectral convolution modules based on equation 1. This substitution effectively enlarges the receptive field, enabling the model to capture global structures from sparse or irregularly sampled data without incurring a parameter count explosion. We refer to the resulting model as the Spectral Convolutional Conditional Neural Process (SConvCNP).

Computational Complexity. The computational cost of the SConvCNP comprises three parts: (i) $\mathcal{O}(|\mathcal{D}_c||\mathcal{G}|)$ to compute the discretized functional embedding on the grid $\mathcal{G}$ (equation 3); (ii) $\mathcal{O}(|\mathcal{G}|\log|\mathcal{G}|)$ for the FFT-based spectral convolution (equation 1); and (iii) $\mathcal{O}(|\mathcal{D}_q||\mathcal{G}|)$ to interpolate grid embeddings at the query locations (equation 4). Overall, the total complexity is $\mathcal{O}\big(|\mathcal{G}|(|\mathcal{D}_c| + \log|\mathcal{G}| + |\mathcal{D}_q|)\big)$, matching the ConvCNP complexity of $\mathcal{O}\big(|\mathcal{G}|(|\mathcal{D}_c| + 1 + |\mathcal{D}_q|)\big)$ up to logarithmic factors. Both architectures therefore scale *linearly* with task size. A key limitation, however, is that $|\mathcal{G}|$ grows exponentially with input dimension, restricting these methods to low-dimensional domains. In contrast, transformer-based NPs (TNPs; Kim et al. [2019], Nguyen and Grover [2022], Feng et al. [2022], Ashman et al. [2024a]) avoid gridding entirely and thus scale more gracefully with input dimension. Instead, they scale quadraticaly in task size $\mathcal{O}(|\mathcal{D}_c|^2 + |\mathcal{D}_c||\mathcal{D}_q|)$.

Positional Encodings The convolution operator preserves translation equivariance under the Fourier transform (see equation 1). However, practical FNOs implementations often include explicit positional information to improve predictive accuracy [Li et al., 2020a, Tran et al., 2021, Gupta et al., 2021, Rahman et al., 2022b, Helwig et al., 2023, Tripura and Chakraborty, 2023, Liu et al., 2023a, Li et al., 2024]. Accordingly, we augment the functional embedding $\varphi_e[\mathcal{D}_c](x)$ with positional information:

$$\widetilde{\varphi}_e[\mathcal{D}_c](x) = \big(\varphi_e[\mathcal{D}_c](x),\, x\big).$$

Although this addition breaks translation equivariance, it helps improving the performance (see Section B.5 for ablation results). Future work may explore *relative* positional encodings [Shaw et al., 2018, Su et al., 2024], which preserve translation equivariance while providing spatial context.

Discretization Sensitivity of DFT. Unlike the continuous Fourier transform, the DFT—and therefore the FFT—depends inherently on the grid $\mathcal{G}$ on which $\varphi_e[\mathcal{D}_c]$ (or $\widetilde{\varphi}_e[\mathcal{D}_c]$) is represented. This dependence arises from both the grid resolution and its physical extent: changing either alters the resulting Fourier coefficients and can lead to inconsistent behavior in the outputs (see Section A). Sensitivity to resolution is not unique to DFT; CNNs exhibit analogous issues [Raonic et al., 2023, Bartolucci et al., 2023]. For example, ConvCNP mitigates this effect by fixing the grid resolution. While spatial CNNs become stable once the resolution is fixed, DFT-based methods remain sensitive unless *both* the resolution and the physical range are controlled. Accordingly, we fix both, choosing a domain sufficiently large to cover all context and query inputs across tasks. When this is impractical, the domain can instead be divided into (possibly overlapping) fixed-size patches, with spectral convolutions applied independently to each patch and the outputs aggregated—mirroring the mechanism of standard convolution layers. Related ideas in efficient transformer architectures suggest that this is a promising direction for future work [Beltagy et al., 2020, Zaheer et al., 2020, Liu et al., 2021, Ding et al., 2023].

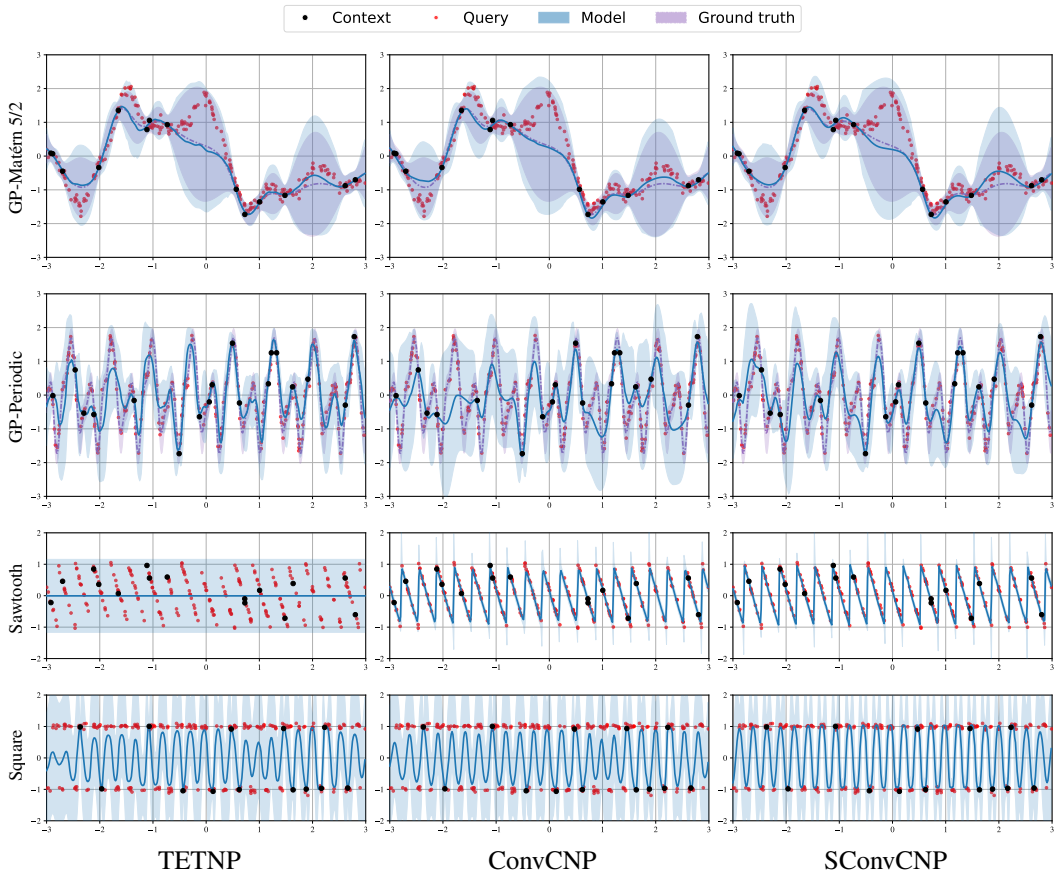

Figure 1: Example predictions for synthetic data. For each model, the blue curve shows its predictive mean and the shaded region indicates ± 2 standard deviations under the corresponding Gaussian predictive distribution. In the first two rows, where the data are generated from a Gaussian process, the true joint density is shown in purple: the dash-dotted line marks the true mean, and the shaded band spans ± 2 standard deviations around that mean. Red points denote query samples drawn from the ground-truth process, and black points indicate the context observations.

4 Experiments

We evaluate our framework on four regression benchmarks and compare its performance against several members of CNPs family. Specifically, we include the original CNP [Garnelo et al., 2018a], the Attentive CNP (AttCNP; Kim et al. [2019]), the Convolutional CNP (ConvCNP; Gordon et al. [2019]), the Transformer Neural Process (TNP; Nguyen and Grover [2022]), and the Translation-Equivariant Transformer Neural Process (TETNP; Ashman et al. [2024a]). To evaluate model performance, we report both the log-likelihood and root-mean-squared error (RMSE) metrics. Additional details on datasets, architectures, and training procedures are provided in Appendix B. The implementation and experimental code are available at https://github.com/peiman-m/SConvCNP.

4.1 Synthetic 1-D Regression

We begin by evaluating models on four synthetic benchmarks generated from distinct stochastic processes: a GP with a Matérn–5/2 kernel, a GP with a periodic kernel, a sawtooth-wave generator, and a square-wave generator. For each benchmark, the parameters of the generative process—kernel hyperparameters for the GPs, frequency and direction for the sawtooth generator, and frequency and duty cycle for the square-wave generator—are sampled randomly. Complete experimental details are provided in Appendix B.1. Table 1 reports the average evaluation metrics over 1,000 test batches, each containing 16 tasks (each task defined in equation 2). For every task—independent of batch—both x_c and x_q are sampled independently from $\mathcal{U}[-3, 3)$. For each test batch, the number of context

Table 1: Comparison of predictive performance across methods on synthetically generated tasks (4 seeds). Lower RMSE and higher log-likelihood indicate better performance. For each metric and experimental setting, boldface highlights the models ranking in the top two.

Metric	Data	Model					
		CNP	AttCNP	TNP	TETNP	ConvCNP	SConvCNP
Log-likelihood ↑	Matérn 5/2	$-0.54_{\pm 0.01}$	$-0.32_{\pm 0.00}$	$\mathbf{-0.29}_{\pm 0.00}$	$\mathbf{-0.28}_{\pm 0.00}$	$-0.30_{\pm 0.01}$	$\mathbf{-0.29}_{\pm 0.01}$
	Periodic	$-1.20_{\pm 0.00}$	$-0.87_{\pm 0.05}$	$-0.76_{\pm 0.03}$	$\mathbf{-0.57}_{\pm 0.02}$	$-0.73_{\pm 0.01}$	$\mathbf{-0.66}_{\pm 0.00}$
	Sawtooth	$-0.90_{\pm 0.00}$	$-0.90_{\pm 0.00}$	$-0.90_{\pm 0.00}$	$-0.90_{\pm 0.00}$	$\mathbf{0.10}_{\pm 0.30}$	$\mathbf{0.82}_{\pm 0.03}$
	Square	$-1.39_{\pm 0.00}$	$-1.41_{\pm 0.01}$	$-1.33_{\pm 0.02}$	$\mathbf{-1.17}_{\pm 0.06}$	$-1.35_{\pm 0.04}$	$\mathbf{-1.13}_{\pm 0.03}$
RMSE ↓	Matérn 5/2	$0.50_{\pm 0.00}$	$\mathbf{0.45}_{\pm 0.00}$	$\mathbf{0.44}_{\pm 0.00}$	$\mathbf{0.44}_{\pm 0.00}$	$\mathbf{0.45}_{\pm 0.00}$	$\mathbf{0.45}_{\pm 0.00}$
	Periodic	$0.81_{\pm 0.00}$	$0.65_{\pm 0.02}$	$0.60_{\pm 0.02}$	$\mathbf{0.50}_{\pm 0.01}$	$0.57_{\pm 0.00}$	$\mathbf{0.53}_{\pm 0.00}$
	Sawtooth	$0.58_{\pm 0.00}$	$0.58_{\pm 0.00}$	$0.58_{\pm 0.00}$	$0.58_{\pm 0.00}$	$\mathbf{0.40}_{\pm 0.06}$	$\mathbf{0.24}_{\pm 0.00}$
	Square	$0.98_{\pm 0.00}$	$1.00_{\pm 0.01}$	$0.93_{\pm 0.02}$	$\mathbf{0.81}_{\pm 0.01}$	$0.89_{\pm 0.02}$	$\mathbf{0.79}_{\pm 0.01}$

points, shared by all tasks within that batch, is drawn independently according to $|\mathcal{I}_c| \sim \mathcal{U}[5, 25)$. The number of query points, however, is fixed at $|\mathcal{I}_q| = 256$ for all the test tasks.

As shown, SConvCNP consistently outperforms or closely matches strong baselines, including TETNP and ConvCNP, which represent the current state of the art. Figure 1 provides qualitative comparisons of predictive maps produced by SConvCNP, ConvCNP, and TETNP (other models are omitted due to space constraints).

For the sawtooth-wave benchmark, we were unable to successfully train either TNP or TETNP. Across the configurations we attempted—including increased model capacity and multiple randomized initializations—their predictions consistently collapsed to zero. We hypothesize that this failure mode stems from *spectral bias*, the tendency of neural networks to favor low-frequency structure over high-frequency components [Rahaman et al., 2019, Ronen et al., 2019, Basri et al., 2020, Tancik et al., 2020, Fridovich-Keil et al., 2022]. Recent findings by Vasudeva et al. [2025] further suggest that this bias is *exacerbated* in transformer architectures compared with convolutional networks. This interpretation aligns with observations by Nguyen and Grover [2022], who originally introduced TNPs. They report degraded performance on GP samples drawn from a periodic kernel—reflected in poor log-likelihood scores—despite strong results on Matérn-kernel tasks. Interestingly, our experiments on periodic-kernel GP tasks (Table 1) do *not* replicate this limitation. A key distinction is that we employ the Efficient Query TNP (EQTNP) architecture of Feng et al. [2022], rather than the original TNP design [Nguyen and Grover, 2022] (see Appendix B.1.1 for architectural details). Finally, although sawtooth and square-wave functions share similar discontinuity and high-frequency characteristics, Table 1 shows that TNP and TETNP do *not* collapse when trained on square-wave signals. Investigating the source of this discrepancy falls beyond the scope of this work but represents an intriguing direction for future study.

4.2 Predator–Prey Dynamics

We next assess performance on simulated trajectories from a stochastic variant of the Lotka–Volterra predator–prey system [Lotka, 1910, Volterra, 1926], following the formulation in Bruinsma et al. [2023]. Let U_t and V_t denote the prey and predator populations at time t, respectively. Their dynamics evolve according to the stochastic Lotka–Volterra system

$$dU_t = \alpha U_t\, dt - \beta U_t V_t\, dt + \sigma U_t^{\nu}\, dB_t^{(1)}, \qquad dV_t = -\gamma V_t\, dt + \delta U_t V_t\, dt + \sigma V_t^{\nu}\, dB_t^{(2)},$$

where $B_t^{(1)}$ and $B_t^{(2)}$ are independent Brownian motions. In the deterministic part of the model, the prey population grows exponentially at rate α in the absence of predation, while the predator population declines at rate γ without access to prey. The interaction terms $\beta U_t V_t$ and $\delta U_t V_t$ describe predation and the resulting reproductive gain for predators. Stochasticity is introduced through the multiplicative noise terms σU_t^{ν} and σV_t^{ν}, where σ controls the overall noise intensity and ν determines how fluctuations scale with population size.

Table 2: Comparison of predictive performance across methods on test tasks constructed from the Pred–Prey population dynamics simulation (4 seeds). Lower RMSE and higher log-likelihood indicate better performance. Boldface highlights the models ranking in the top two for each metric.

	CNP	AttCNP	TNP	TETNP	ConvCNP	SConvCNP
Log-likelihood ↑	$-0.27_{\pm 0.01}$	$0.08_{\pm 0.00}$	$0.12_{\pm 0.00}$	$\mathbf{0.16}_{\pm 0.00}$	$\mathbf{0.14}_{\pm 0.00}$	$\mathbf{0.14}_{\pm 0.00}$
RMSE ↓	$0.37_{\pm 0.00}$	$\mathbf{0.31}_{\pm 0.00}$	$\mathbf{0.31}_{\pm 0.00}$	$\mathbf{0.30}_{\pm 0.0.00}$	$\mathbf{0.31}_{\pm 0.00}$	$\mathbf{0.30}_{\pm 0.00}$

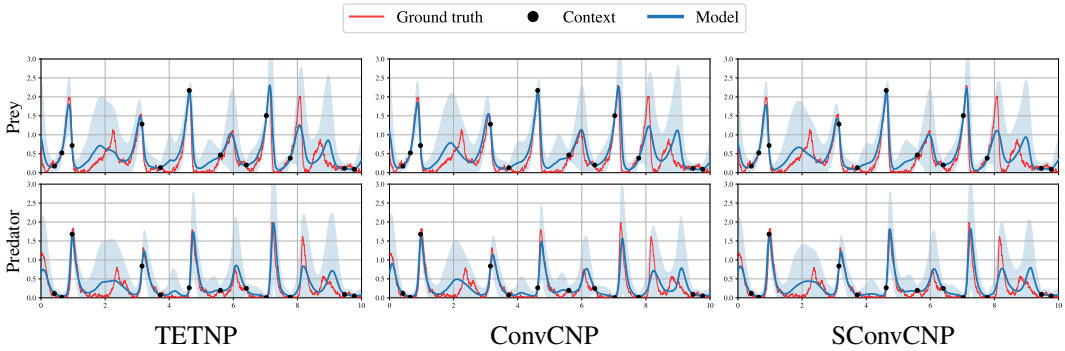

Figure 2: Illustrative model predictions based on simulated trajectories from Lotka–Volterra predator–prey dynamics. Black points represent the context observations. For each model, the blue curve depicts the predictive mean, and the shaded region marks the ± 2-standard-deviation band of the associated Gaussian predictive distribution. Ground-truth traffic measurements, shown in red, are included as a reference.

We simulate trajectories over a dense uniform grid on $t \in [-10, 100)$, discarding the initial 10 years as burn-in. For each task, context points x_c and query points x_q are drawn independently and uniformly from the retained interval $[0, 100)$. Population values (corresponding to $y = [U_t, V_t]$) and time inputs (corresponding to $x = t$) are rescaled by factors of 0.01 and 0.1, respectively, before being fed to the models. A complete description of the experimental protocol is provided in Appendix B.2.

Table 2 reports average evaluation metrics over 1,000 test batches, each containing 16 tasks. For each batch, the number of context points—shared across all tasks—is sampled as $|\mathcal{I}_c| \sim \mathcal{U}[5, 25)$, and the number of query points is fixed at $|\mathcal{I}_q| = 256$. SConvCNP achieves log-likelihood performance comparable to ConvCNP, and both are competitive with TETNP. In terms of RMSE, SConvCNP matches TETNP, indicating that its predictive uncertainty intervals are slightly wider. Qualitative comparisons of predictive maps for the top three models appear in Figure 2.

4.3 Traffic Flow

For our third experiment, we evaluate on the California traffic-flow dataset from LargeST [Liu et al., 2023b]. This dataset comprises five years (2017–2021) of traffic measurements recorded every 5 minutes by approximately 8,600 loop-detector sensors deployed across California's highway network. We focus on the year 2020, where traffic patterns are expected to exhibit heightened variability due to the abrupt onset of the COVID-19 pandemic. For each sensor, we segment its year-long time series into non-overlapping 14-day windows and downsample each window by a factor of 6 (from 5-minute to 30-minute resolution). Each window is treated as a dense trajectory from which individual tasks are constructed. Additional details of the experimental setup are provided in Appendix B.3.

Table 3: Comparison of predictive performance across methods on test tasks constructed from California traffic flow measurements (4 seeds). Lower RMSE and higher log-likelihood indicate better performance. Boldface highlights the models ranking in the top two for each metric.

	CNP	AttCNP	TNP	TETNP	ConvCNP	SConvCNP
Log-likelihood ↑	$1.73_{\pm 0.10}$	$1.80_{\pm 0.01}$	$1.93_{\pm 0.06}$	$1.72_{\pm 0.02}$	$\mathbf{1.98}_{\pm 0.02}$	$\mathbf{2.05}_{\pm 0.02}$
RMSE ↓	$\mathbf{0.05}_{\pm 0.00}$	$\mathbf{0.05}_{\pm 0.00}$	$\mathbf{0.04}_{\pm 0.00}$	$\mathbf{0.05}_{\pm 0.00}$	$\mathbf{0.04}_{\pm 0.00}$	$\mathbf{0.04}_{\pm 0.00}$

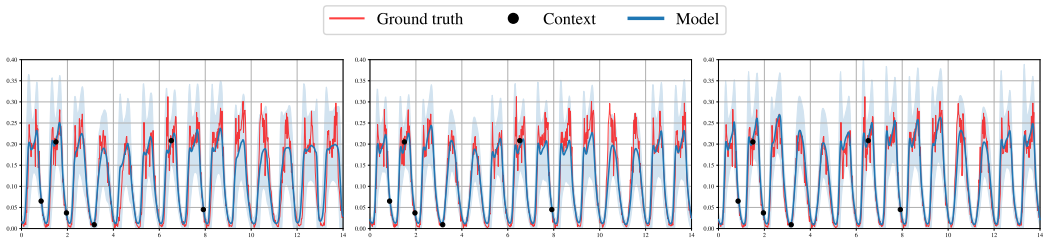

Figure 3: Illustrative model predictions on the California traffic-flow dataset. Black points represent the context observations. For each model, the blue curve depicts the predictive mean, and the shaded region marks the ± 2-standard-deviation band of the associated Gaussian predictive distribution. Ground-truth traffic measurements, shown in red, are included as a reference.

Table 3 reports evaluation metrics averaged over 66,586 test tasks, constructed from 2,561 held-out test sensors and partitioned into batches of size 32. For each batch, the number of context points—shared across all tasks—is drawn as $|\mathcal{I}_c| \sim \mathcal{U}[5, 25)$, and the number of query points is fixed at $|\mathcal{I}_q| = 50$. Across all metrics, SConvCNP achieves the best performance, attaining the highest log-likelihoods and lowest RMSE among the baselines. Figure 3 provides qualitative comparisons of the predictive maps produced by the three models.

4.4 Image Completion

For our final experiment, we evaluate model performance on an image-completion task formulated as a spatial regression problem, where the model maps 2D pixel coordinates to their corresponding intensity values. We use images from the Describable Textures Dataset (DTD; Cimpoi et al. [2014]), and construct each task from a processed 64×64 subsampled crop of an original image. Because each task contains a large number of context and query pixels, we were unable to fit TETNP within our computational budget—even after reducing its size—so we exclude it from this experiment. Additional experimental details are provided in Appendix B.4. Table 4 reports results averaged over 1,880 test tasks, evaluated in batches of 16. For each batch, the number of context points—shared across all tasks—is sampled as $|\mathcal{I}_c| \sim \mathcal{U}[5, 1024)$; all remaining pixels serve as query points, so that $|\mathcal{I}_q| = 64 \times 64 - |\mathcal{I}_c|$. Among the baselines, SConvCNP achieves the highest log-likelihoods and the second-lowest RMSE. Figure 4 presents qualitative comparisons of the predictive maps produced by the top three models.

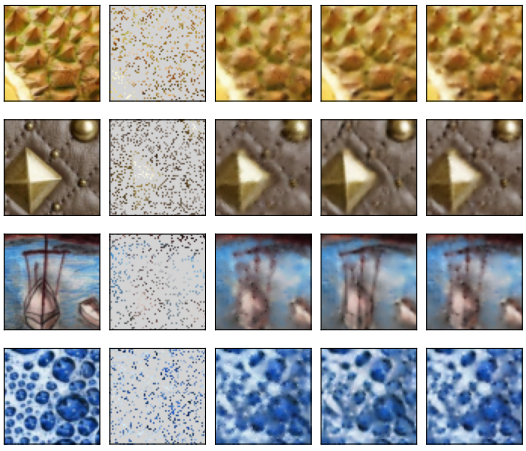

Figure 4: Illustrative model outputs for image-completion tasks using samples from the DTD dataset. Grey pixels indicate query regions, and the remaining pixels serve as context observations. For each model, query pixels are filled with the mean of the predictive distribution.

Table 4: Comparison of predictive performance across methods on image-completion test tasks constructed from images of the DTD dataset (4 seeds). Lower RMSE and higher log-likelihood indicate better performance. Boldface highlights the models ranking in the top two for each metric.

	CNP	AttCNP	TNP	ConvCNP	SConvCNP
Log-likelihood ↑	$0.67_{\pm 0.01}$	$1.39_{\pm .02}$	$1.39_{\pm 0.05}$	$\mathbf{1.48}_{\pm 0.00}$	$\mathbf{1.50}_{\pm 0.02}$
RMSE ↓	$0.14_{\pm 0.00}$	$\mathbf{0.08}_{\pm 0.00}$	$\mathbf{0.06}_{\pm 0.00}$	$\mathbf{0.08}_{\pm 0.00}$	$\mathbf{0.08}_{\pm 0.00}$

5 Related Works

Neural PDE Solvers The high computational cost of traditional numerical PDE solvers has motivated the development of more efficient alternatives based on deep learningGupta and Brandstetter [2022]. Among these, neural operators [Li et al., 2020b, Kovachki et al., 2023], and in particular Fourier neural operators (FNOs, Li et al., 2020a), have become popular. Numerous extensions have since been proposed: Helwig et al. [2023] introduced rotation- and reflection-equivariant FNOs; Gupta et al. [2021] developed multiwavelet neural operators by projecting kernels onto predefined polynomial bases; and Tran et al. [2021] reduced model complexity through separable Fourier representations. Xiao et al. [2024] proposed the Amortized FNO (AM-FNO), which substantially cuts parameter count by amortizing the Fourier kernel parameterization, while Qin et al. [2024] analyzed FNOs through the lens of spectral bias. Koshizuka et al. [2024] provided a mean-field theoretical perspective, and Zheng et al. [2024] introduced the Mamba Neural Operator, which couples global context via a state-space model to achieve linear complexity and representation equivalence. Bartolucci et al. [2023] examined continuous–discrete consistency, showing that it holds for convolutional neural operators [Raonic et al., 2023]. Additional works include orthogonal-attention eigenfunction methods for operator learning [Xiao et al., 2023] and hierarchical transformers with frequency-aware priors for resolution-invariant super-resolution [Luo et al., 2024].

Function Space Inference In non-parametric Bayesian modeling, GPs and deep GPs [Damianou and Lawrence, 2013] provide flexible function-space priors with well-calibrated uncertainty estimates, but their computational cost becomes prohibitive on large datasets. This challenge has motivated the development of Bayesian neural networks (BNNs; MacKay [1992], Hinton and Van Camp [1993], Neal [2012]), which combine neural network expressiveness with Bayesian uncertainty quantification. However, specifying meaningful priors over network weights remains notoriously difficult. Recent work therefore reframes Bayesian inference in neural networks as learning a posterior over the *functions* induced by stochastic weights [Wolpert, 1993, Qiu et al., 2023]. Variational implicit processes (VIPs; Ma et al. [2019], Santana et al. [2021], Ortega et al. [2022]) generalize GPs by defining implicit stochastic processes through latent variables, while functional variational BNNs (fBNNs; Sun et al. [2019]) enforce alignment between BNN-induced priors and target priors by minimizing a functional KL divergence—though this objective can be difficult to compute and, in some cases, ill-posed [Burt et al., 2020]. Subsequent work has aimed to address these limitations [Ma and Hernández-Lobato, 2021, Rudner et al., 2022, Wild et al., 2022, Rudner et al., 2023, Wu et al., 2024]. Orthogonally, energy-based models have been explored for representing stochastic processes in function space [Yang et al., 2020, Lim et al., 2022]. More recently, several approaches have extended diffusion and flow models to function spaces [Phillips et al., 2022, Dutordoir et al., 2023, Mathieu et al., 2023, Franzese et al., 2023, Lim et al., 2023, Kerrigan et al., 2024, Zhang and Scott, 2025].

6 Conclusion

In this work, we introduced the Spectral Convolutional Conditional Neural Process (SConvCNP), a new member of the CNP family that leverages advances in operator learning—particularly the Fourier Neural Operator—to model probabilistic prediction maps within the neural processes framework, with a special emphasis on the Convolutional Conditional Neural Process (ConvCNP). Through experiments on both synthetic and real datasets, we show that SConvCNP delivers highly competitive performance relative to recent state-of-the-art methods, highlighting the promise of bridging research on neural processes with developments in operator learning.

Acknowledgments

We thank Arman Hasanzadeh and Jonathan W. Siegel for their insightful feedback and constructive suggestions. We also acknowledge the Texas A&M High Performance Research Computing facility for providing the computational resources used in this study. Finally, we thank the anonymous reviewers for their valuable comments and suggestions, which helped improve the quality of this work.

References

Anna Allen, Stratis Markou, Will Tebbutt, James Requeima, Wessel P Bruinsma, Tom R Andersson, Michael Herzog, Nicholas D Lane, Matthew Chantry, J Scott Hosking, et al. End-to-end data-driven weather prediction. *Nature*, 641(8065):1172–1179, 2025.

Matthew Ashman, Cristiana Diaconu, Junhyuck Kim, Lakee Sivaraya, Stratis Markou, James Requeima, Wessel P Bruinsma, and Richard E Turner. Translation equivariant transformer neural processes. *arXiv preprint arXiv:2406.12409*, 2024a.

Matthew Ashman, Cristiana Diaconu, Adrian Weller, Wessel Bruinsma, and Richard Turner. Approximately equivariant neural processes. *Advances in Neural Information Processing Systems*, 37: 97088–97123, 2024b.

Matthew Ashman, Cristiana Diaconu, Eric Langezaal, Adrian Weller, and Richard E. Turner. Gridded transformer neural processes for spatio-temporal data. In *Forty-second International Conference on Machine Learning*, 2025. URL `https://openreview.net/forum?id=O0oe7hPtbl`.

Francesca Bartolucci, Emmanuel de Bezenac, Bogdan Raonic, Roberto Molinaro, Siddhartha Mishra, and Rima Alaifari. Representation equivalent neural operators: a framework for alias-free operator learning. *Advances in Neural Information Processing Systems*, 36:69661–69672, 2023.

Ronen Basri, Meirav Galun, Amnon Geifman, David Jacobs, Yoni Kasten, and Shira Kritchman. Frequency bias in neural networks for input of non-uniform density. In *International conference on machine learning*, pages 685–694. PMLR, 2020.

Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

Benjamin Bloem-Reddy and Yee Whye Teh. Probabilistic symmetries and invariant neural networks. *Journal of Machine Learning Research*, 21(90):1–61, 2020.

Ron Bracewell and Peter B Kahn. The fourier transform and its applications. *American Journal of Physics*, 34(8):712–712, 1966.

Wessel P Bruinsma. Convolutional conditional neural processes. *arXiv preprint arXiv:2408.09583*, 2024.

Wessel P Bruinsma, James Requeima, Andrew YK Foong, Jonathan Gordon, and Richard E Turner. The gaussian neural process. *arXiv preprint arXiv:2101.03606*, 2021.

Wessel P Bruinsma, Stratis Markou, James Requiema, Andrew YK Foong, Tom R Andersson, Anna Vaughan, Anthony Buonomo, J Scott Hosking, and Richard E Turner. Autoregressive conditional neural processes. *arXiv preprint arXiv:2303.14468*, 2023.

David R Burt, Sebastian W Ober, Adrià Garriga-Alonso, and Mark van der Wilk. Understanding variational inference in function-space. *arXiv preprint arXiv:2011.09421*, 2020.

Edoardo Calvello, Nikola B Kovachki, Matthew E Levine, and Andrew M Stuart. Continuum attention for neural operators. *arXiv preprint arXiv:2406.06486*, 2024.

Tianping Chen and Hong Chen. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE transactions on neural networks*, 6(4):911–917, 1995.

Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3606–3613, 2014.

James W Cooley and John W Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of computation*, 19(90):297–301, 1965.

Andreas Damianou and Neil D Lawrence. Deep gaussian processes. In *Artificial intelligence and statistics*, pages 207–215. PMLR, 2013.

Jiayu Ding, Shuming Ma, Li Dong, Xingxing Zhang, Shaohan Huang, Wenhui Wang, Nanning Zheng, and Furu Wei. Longnet: Scaling transformers to 1,000,000,000 tokens. *arXiv preprint arXiv:2307.02486*, 2023.

Xiaohan Ding, Xiangyu Zhang, Jungong Han, and Guiguang Ding. Scaling up your kernels to 31x31: Revisiting large kernel design in cnns. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11963–11975, 2022.

Charles Dugas, Yoshua Bengio, François Bélisle, Claude Nadeau, and René Garcia. Incorporating second-order functional knowledge for better option pricing. *Advances in neural information processing systems*, 13, 2000.

Emilien Dupont, Yee Whye Teh, and Arnaud Doucet. Generative models as distributions of functions. *arXiv preprint arXiv:2102.04776*, 2021.

Vincent Dutordoir, Alan Saul, Zoubin Ghahramani, and Fergus Simpson. Neural diffusion processes. In *International Conference on Machine Learning*, pages 8990–9012. PMLR, 2023.

Leo Feng, Hossein Hajimirsadeghi, Yoshua Bengio, and Mohamed Osama Ahmed. Latent bottlenecked attentive neural processes. *arXiv preprint arXiv:2211.08458*, 2022.

David J Field. Relations between the statistics of natural images and the response properties of cortical cells. *Journal of the Optical Society of America A*, 4(12):2379–2394, 1987.

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.

Giulio Franzese, Giulio Corallo, Simone Rossi, Markus Heinonen, Maurizio Filippone, and Pietro Michiardi. Continuous-time functional diffusion processes. *Advances in Neural Information Processing Systems*, 36:37370–37400, 2023.

Sara Fridovich-Keil, Raphael Gontijo Lopes, and Rebecca Roelofs. Spectral bias in practice: The role of function frequency in generalization. *Advances in Neural Information Processing Systems*, 35:7368–7382, 2022.

Matteo Frigo and Steven G Johnson. The design and implementation of fftw3. *Proceedings of the IEEE*, 93(2):216–231, 2005.

Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202, 1980.

Jacob Gardner, Geoff Pleiss, Kilian Q Weinberger, David Bindel, and Andrew G Wilson. Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. *Advances in neural information processing systems*, 31, 2018.

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, 2018a.

Marta Garnelo, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J Rezende, SM Eslami, and Yee Whye Teh. Neural processes. *arXiv preprint arXiv:1807.01622*, 2018b.

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.

Gaurav Gupta, Xiongye Xiao, and Paul Bogdan. Multiwavelet-based operator learning for differential equations. *Advances in neural information processing systems*, 34:24048–24062, 2021.

Jayesh K Gupta and Johannes Brandstetter. Towards multi-spatiotemporal-scale generalized pde modeling. *arXiv preprint arXiv:2209.15616*, 2022.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Jacob Helwig, Xuan Zhang, Cong Fu, Jerry Kurtin, Stephan Wojtowytsch, and Shuiwang Ji. Group equivariant fourier neural operators for partial differential equations. *arXiv preprint arXiv:2306.05697*, 2023.

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

Geoffrey E Hinton and Drew Van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the sixth annual conference on Computational learning theory*, pages 5–13, 1993.

Peter Holderrieth, Michael J Hutchinson, and Yee Whye Teh. Equivariant learning of stochastic fields: Gaussian processes and steerable conditional neural processes. In *International conference on machine learning*, pages 4297–4307. PMLR, 2021.

Daolang Huang, Manuel Haussmann, Ulpu Remes, ST John, Grégoire Clarté, Kevin Luck, Samuel Kaski, and Luigi Acerbi. Practical equivariances via relational conditional neural processes. *Advances in Neural Information Processing Systems*, 36:29201–29238, 2023.

Saurav Jha, Dong Gong, Xuesong Wang, Richard E Turner, and Lina Yao. The neural process family: Survey, applications and perspectives. *arXiv preprint arXiv:2209.00517*, 2022.

Yohan Jung and Jinkyoo Park. Bayesian convolutional deep sets with task-dependent stationary prior. In *International Conference on Artificial Intelligence and Statistics*, pages 3795–3824. PMLR, 2023.

Makoto Kawano, Wataru Kumagai, Akiyoshi Sannai, Yusuke Iwasawa, and Yutaka Matsuo. Group equivariant conditional neural processes. *arXiv preprint arXiv:2102.08759*, 2021.

Gavin Kerrigan, Giosue Migliorini, and Padhraic Smyth. Functional flow matching. In Sanjoy Dasgupta, Stephan Mandt, and Yingzhen Li, editors, *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*, volume 238 of *Proceedings of Machine Learning Research*, pages 3934–3942. PMLR, 02–04 May 2024. URL `https://proceedings.mlr.press/v238/kerrigan24a.html`.

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.

Mingyu Kim, Kyeongryeol Go, and Se-Young Yun. Neural processes with stochastic attention: Paying more attention to the context dataset. *arXiv preprint arXiv:2204.05449*, 2022.

David M Knigge, David W. Romero, Albert Gu, Efstratios Gavves, Erik J Bekkers, Jakub Mikolaj Tomczak, Mark Hoogendoorn, and Jan jakob Sonke. Modelling long range dependencies in nd: From task-specific to a general purpose CNN. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=ZW5aK4yCRqU`.

Takeshi Koshizuka, Masahiro Fujisawa, Yusuke Tanaka, and Issei Sato. Understanding the expressivity and trainability of fourier neural operator: A mean-field perspective. *Advances in Neural Information Processing Systems*, 37:11021–11060, 2024.

Nikola Kovachki, Samuel Lanthaler, and Siddhartha Mishra. On universal approximation and error bounds for fourier neural operators. *Journal of Machine Learning Research*, 22(290):1–76, 2021.

Nikola B Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew M Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces with applications to pdes. *J. Mach. Learn. Res.*, 24(89):1–97, 2023.

Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

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.

Hyungi Lee, Eunggu Yun, Giung Nam, Edwin Fong, and Juho Lee. Martingale posterior neural processes. *arXiv preprint arXiv:2304.09431*, 2023.

Juho Lee, Yoonho Lee, Jungtaek Kim, Eunho Yang, Sung Ju Hwang, and Yee Whye Teh. Bootstrapping neural processes. *Advances in neural information processing systems*, 33:6606–6615, 2020.

Shibo Li, Xin Yu, Wei Xing, Robert Kirby, Akil Narayan, and Shandian Zhe. Multi-resolution active learning of fourier neural operators. In *International Conference on Artificial Intelligence and Statistics*, pages 2440–2448. PMLR, 2024.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020a.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Graph kernel network for partial differential equations. *arXiv preprint arXiv:2003.03485*, 2020b.

Jen Ning Lim, Sebastian Vollmer, Lorenz Wolf, and Andrew Duncan. Energy-based models for functional data using path measure tilting. *arXiv preprint arXiv:2202.01929*, 2022.

Sungbin Lim, Eunbi Yoon, Taehyun Byun, Taewon Kang, Seungwoo Kim, Kyungjae Lee, and Sungjoon Choi. Score-based generative modeling through stochastic evolution equations in hilbert spaces. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL `https://openreview.net/forum?id=GrElRvXnEj`.

Ning Liu, Siavash Jafarzadeh, and Yue Yu. Domain agnostic fourier neural operators. *Advances in Neural Information Processing Systems*, 36:47438–47450, 2023a.

Xu Liu, Yutong Xia, Yuxuan Liang, Junfeng Hu, Yiwei Wang, Lei Bai, Chao Huang, Zhenguang Liu, Bryan Hooi, and Roger Zimmermann. Largest: A benchmark dataset for large-scale traffic forecasting. In *Advances in Neural Information Processing Systems*, 2023b.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Alfred J. Lotka. Contribution to the theory of periodic reactions. *The Journal of Physical Chemistry*, 14(3):271–274, 1910. ISSN 0092-7325. doi: 10.1021/j150111a004. URL `https://doi.org/10.1021/j150111a004`.

Christos Louizos, Xiahan Shi, Klamer Schutte, and Max Welling. The functional neural process. *Advances in Neural Information Processing Systems*, 32, 2019.

Wenjie Luo, Yujia Li, Raquel Urtasun, and Richard Zemel. Understanding the effective receptive field in deep convolutional neural networks. *Advances in neural information processing systems*, 29, 2016.

Xihaier Luo, Xiaoning Qian, and Byung-Jun Yoon. Hierarchical neural operator transformer with learnable frequency-aware loss prior for arbitrary-scale super-resolution. *arXiv preprint arXiv:2405.12202*, 2024.

Chao Ma and José Miguel Hernández-Lobato. Functional variational inference based on stochastic process generators. *Advances in Neural Information Processing Systems*, 34:21795–21807, 2021.

Chao Ma, Yingzhen Li, and José Miguel Hernández-Lobato. Variational implicit processes. In *International Conference on Machine Learning*, pages 4222–4233. PMLR, 2019.

David JC MacKay. A practical bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472, 1992.

Stratis Markou, James Requeima, Wessel P Bruinsma, Anna Vaughan, and Richard E Turner. Practical conditional neural processes via tractable dependent predictions. *arXiv preprint arXiv:2203.08775*, 2022.

Emile Mathieu, Adam Foster, and Yee Teh. On contrastive representations of stochastic processes. *Advances in Neural Information Processing Systems*, 34:28823–28835, 2021.

Emile Mathieu, Vincent Dutordoir, Michael J Hutchinson, Valentin De Bortoli, Yee Whye Teh, and Richard E Turner. Geometric neural diffusion processes. *arXiv preprint arXiv:2307.05431*, 2023.

Peiman Mohseni, Nick Duffield, Bani Mallick, and Arman Hasanzadeh. Adaptive conditional quantile neural processes. In *Uncertainty in Artificial Intelligence*, pages 1445–1455. PMLR, 2023.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

Radford M Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

Tung Nguyen and Aditya Grover. Transformer neural processes: Uncertainty-aware meta learning via sequence modeling. *arXiv preprint arXiv:2207.04179*, 2022.

Alan V Oppenheim. *Discrete-time signal processing*. Pearson Education India, 1999.

Luis A Ortega, Simón Rodríguez Santana, and Daniel Hernández-Lobato. Deep variational implicit processes. *arXiv preprint arXiv:2206.06720*, 2022.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318. Pmlr, 2013.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Chao Peng, Xiangyu Zhang, Gang Yu, Guiming Luo, and Jian Sun. Large kernel matters – improve semantic segmentation by global convolutional network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

Angus Phillips, Thomas Seror, Michael Hutchinson, Valentin De Bortoli, Arnaud Doucet, and Emile Mathieu. Spectral diffusion processes. *arXiv preprint arXiv:2209.14125*, 2022.

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.

Shaoxiang Qin, Fuyuan Lyu, Wenhui Peng, Dingyang Geng, Ju Wang, Xing Tang, Sylvie Leroyer, Naiping Gao, Xue Liu, and Liangzhu Leon Wang. Toward a better understanding of fourier neural operators from a spectral perspective. *arXiv preprint arXiv:2404.07200*, 2024.

Shikai Qiu, Tim GJ Rudner, Sanyam Kapoor, and Andrew G Wilson. Should we learn most likely functions or parameters? *Advances in Neural Information Processing Systems*, 36:35814–35835, 2023.

Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In *International conference on machine learning*, pages 5301–5310. PMLR, 2019.

Md Ashiqur Rahman, Zachary E Ross, and Kamyar Azizzadenesheli. U-no: U-shaped neural operators. *arXiv preprint arXiv:2204.11127*, 2022a.

Md Ashiqur Rahman, Zachary E Ross, and Kamyar Azizzadenesheli. U-no: U-shaped neural operators. *arXiv preprint arXiv:2204.11127*, 2022b.

Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jon Shlens. Stand-alone self-attention in vision models. *Advances in neural information processing systems*, 32, 2019.

Bogdan Raonic, Roberto Molinaro, Tim De Ryck, Tobias Rohner, Francesca Bartolucci, Rima Alaifari, Siddhartha Mishra, and Emmanuel de Bézenac. Convolutional neural operators for robust and accurate learning of pdes. *Advances in Neural Information Processing Systems*, 36: 77187–77200, 2023.

Carl Edward Rasmussen, Christopher KI Williams, et al. *Gaussian processes for machine learning*, volume 1. Springer, 2006.

David W Romero, Anna Kuzina, Erik J Bekkers, Jakub M Tomczak, and Mark Hoogendoorn. Ckconv: Continuous kernel convolution for sequential data. *arXiv preprint arXiv:2102.02611*, 2021.

Basri Ronen, David Jacobs, Yoni Kasten, and Shira Kritchman. The convergence rate of neural networks for learned functions of different frequencies. *Advances in Neural Information Processing Systems*, 32, 2019.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241. Springer, 2015.

Daniel Ruderman and William Bialek. Statistics of natural images: Scaling in the woods. *Advances in neural information processing systems*, 6, 1993.

Tim GJ Rudner, Zonghao Chen, Yee Whye Teh, and Yarin Gal. Tractable function-space variational inference in bayesian neural networks. *Advances in Neural Information Processing Systems*, 35: 22686–22698, 2022.

Tim GJ Rudner, Sanyam Kapoor, Shikai Qiu, and Andrew Gordon Wilson. Function-space regularization in neural networks: A probabilistic perspective. In *International Conference on Machine Learning*, pages 29275–29290. PMLR, 2023.

Simón Rodríguez Santana, Bryan Zaldivar, and Daniel Hernández-Lobato. Function-space inference with sparse implicit processes. *arXiv preprint arXiv:2110.07618*, 2021.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155*, 2018.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Shengyang Sun, Guodong Zhang, Jiaxin Shi, and Roger Grosse. Functional variational bayesian neural networks. *arXiv preprint arXiv:1903.05779*, 2019.

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.

Alasdair Tran, Alexander Mathews, Lexing Xie, and Cheng Soon Ong. Factorized fourier neural operators. *arXiv preprint arXiv:2111.13802*, 2021.

Tapas Tripura and Souvik Chakraborty. Wavelet neural operator for solving parametric partial differential equations in computational mechanics problems. *Computer Methods in Applied Mechanics and Engineering*, 404:115783, 2023.

Bhavya Vasudeva, Deqing Fu, Tianyi Zhou, Elliott Kau, Youqi Huang, and Vatsal Sharan. Transformers learn low sensitivity functions: Investigations and implications. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=4ikjWBs3tE.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Anna Vaughan, Will Tebbutt, J Scott Hosking, and Richard E Turner. Convolutional conditional neural processes for local climate downscaling. *Geoscientific Model Development Discussions*, 2021:1–25, 2021.

Michael Volpp, Fabian Flürenbrock, Lukas Grossberger, Christian Daniel, and Gerhard Neumann. Bayesian context aggregation for neural processes. In *ICLR*, 2021.

V. Volterra. Variazioni e fluttuazioni del bumero d'ondividui in specie animali conviventi. *Memoria della Reale Accademia Nazionale dei Lincei*, 2:31–113, 1926.

Edward Wagstaff, Fabian Fuchs, Martin Engelcke, Ingmar Posner, and Michael A Osborne. On the limitations of representing functions on sets. In *International conference on machine learning*, pages 6487–6494. PMLR, 2019.

Martin J Wainwright and Eero Simoncelli. Scale mixtures of gaussians and the statistics of natural images. *Advances in neural information processing systems*, 12, 1999.

Huiyu Wang, Yukun Zhu, Bradley Green, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. Axial-deeplab: Stand-alone axial-attention for panoptic segmentation. In *European conference on computer vision*, pages 108–126. Springer, 2020.

Qi Wang and Herke Van Hoof. Doubly stochastic variational inference for neural processes with hierarchical latent variables. In *International Conference on Machine Learning*, pages 10018–10028. PMLR, 2020.

Qi Wang and Herke van Hoof. Learning expressive meta-representations with mixture of expert neural processes. *Advances in neural information processing systems*, 35:26242–26255, 2022.

Qi Wang, Marco Federici, and Herke van Hoof. Bridge the inference gaps of neural processes via expectation maximization. In *The Eleventh International Conference on Learning Representations*, 2022.

Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7794–7803, 2018.

Veit David Wild, Robert Hu, and Dino Sejdinovic. Generalized variational inference in function spaces: Gaussian measures meet bayesian deep learning. *Advances in Neural Information Processing Systems*, 35:3716–3730, 2022.

David H Wolpert. Bayesian backpropagation over io functions rather than weights. *Advances in neural information processing systems*, 6, 1993.

Mengjing Wu, Junyu Xuan, and Jie Lu. Functional wasserstein bridge inference for bayesian deep learning. In *The 40th Conference on Uncertainty in Artificial Intelligence*, 2024. URL https://openreview.net/forum?id=Wnht2IqzlN.

Zipeng Xiao, Zhongkai Hao, Bokai Lin, Zhijie Deng, and Hang Su. Improved operator learning by orthogonal attention. *arXiv preprint arXiv:2310.12487*, 2023.

Zipeng Xiao, Siqi Kou, Hao Zhongkai, Bokai Lin, and Zhijie Deng. Amortized fourier neural operators. *Advances in Neural Information Processing Systems*, 37:115001–115020, 2024.

Jin Xu, Jean-Francois Ton, Hyunjik Kim, Adam Kosiorek, and Yee Whye Teh. Metafun: Meta-learning with iterative functional updates. In *International Conference on Machine Learning*, pages 10617–10627. PMLR, 2020.

Jin Xu, Emilien Dupont, Kaspar Märtens, Thomas Rainforth, and Yee Whye Teh. Deep stochastic processes via functional markov transition operators. *Advances in Neural Information Processing Systems*, 36:37975–37994, 2023.

Mengjiao Yang, Bo Dai, Hanjun Dai, and Dale Schuurmans. Energy-based processes for exchangeable data. In *International Conference on Machine Learning*, pages 10681–10692. PMLR, 2020.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. *Advances in neural information processing systems*, 30, 2017.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, 33:17283–17297, 2020.

Jianxin Zhang and Clayton Scott. Flow straight and fast in hilbert space: Functional rectified flow. *arXiv preprint arXiv:2509.10384*, 2025.

Jianwei Zheng, Wei Li, Ni Xu, Junwei Zhu, and Xiaoqin Zhang. Alias-free mamba neural operator. *Advances in Neural Information Processing Systems*, 37:52962–52995, 2024.

A Discretization Sensitivity of the DFT

To illustrate the sensitivity of the discrete Fourier transform (DFT) to grid shape—and how changes in discretization can disturb predictive performance—we consider a simple one-dimensional example (although the analysis extends to higher dimensions). Let $0 < \Delta \ll 1$ denote a fixed discretization resolution, interpreted as the spacing between successive points. Define the grid $\mathcal{G}_1 = (x_0, \ldots, x_{m_1-1})$ as a uniform discretization of the interval $[0, 1]$ with $m_1 = \lfloor \frac{1}{\Delta} \rfloor$ and ordered such that $x_{k-1} < x_k$ for all $k \in \{1, \ldots, m_1 - 1\}$. Let the sampled values of $\varphi_e[\mathcal{D}_c]$ at $\mathcal{G}_1$ be denoted by $\left(\varphi_e\left[\mathcal{D}_c\right](x)\right)_{x\in\mathcal{G}_1} = (\varphi_e[\mathcal{D}_c](x_0),\ \ldots\ , \varphi_e[\mathcal{D}_c](x_{m_1-1}))$. The DFT of $\varphi_e[\mathcal{D}_c]$ over this grid is given by:

$$\hat{\mathcal{F}}\left\{\left(\varphi_e\left[\mathcal{D}_c\right](x)\right)_{x\in\mathcal{G}_1}\right\}(k) = \sum_{n=0}^{m_1-1} \varphi_e[\mathcal{D}_c](x_n)\, e^{-i2\pi\frac{k}{m_1}n}, \quad \forall\, k \in \{0, \ldots, m_1 - 1\}.$$

Thus, the DFT yields Fourier coefficients at the normalized frequencies $\hat{\Xi}_1 = \left(\frac{k}{m_1}\right)_{k=0}^{m_1-1}$. As described in Section 2.1, the FNO implicitly ties its kernel spectrum parameterization to the set $\hat{\Xi}_1$ encountered during training. Now suppose we wish to evaluate the model on a larger domain, say $[0, 2]$, using the same resolution Δ. Then the corresponding grid $\mathcal{G}_2$ has size $m_2 = \lfloor \frac{2}{\Delta} \rfloor$, and the DFT computes coefficients at normalized frequencies $\hat{\Xi}_2 = \left(\frac{k}{m_2}\right)_{k=0}^{m_2-1}$. Since $m_2 \geq 2m_1$, we have $|\hat{\Xi}_2| \geq 2|\hat{\Xi}_1|$. Let $1 \leq k_{\text{max}} \leq m_1$ denote the number of frequency modes retained during training. Clearly, the first k_{max} frequencies of $\hat{\Xi}_2$ do not align with those of $\hat{\Xi}_1$, resulting in a mismatch.

B Experimental Details

All implementations are written in PyTorch [Paszke et al., 2019] and publicly available at https://github.com/peiman-m/SConvCNP. We used a single NVIDIA A100 GPU with 40 GB of memory for all the computations. Our code is based on the implementations of Ashman et al. [2024a] and Bruinsma et al. [2023].

B.1 Synthetic 1-D Regression

B.1.1 Model Architectures

This section details the architectures of all CNP variants used in our experiments. Each model outputs a Gaussian predictive distribution parameterized by a mean and a pre-softplus [Dugas et al., 2000] scale. The pre-softplus value is passed through a softplus transformation, and a minimum noise term of 10^{-6} is then added to the resulting scale. The Gaussian factorizes across query points—as is standard for CNPs (see Section 2.2)—and, for multidimensional outputs, across output dimensions as well. Thus, predictions assume independence both across query locations and across components of each output vector. Unless otherwise noted, all nonlinearities use ReLU activations [Nair and Hinton, 2010].

Conditional Neural Process (CNP): The CNP encodes each context pair $(x_{c,k}, y_{c,k}) \in \mathcal{D}_c$ using separate input and output pathways. The input $x_{c,k}$ and output $y_{c,k}$ are first passed through distinct MLPs with two hidden layers each of dimension 256, producing 256-dimensional representations $\varepsilon_{c,k}^{(x)}$ and $\varepsilon_{c,k}^{(y)}$. Their concatenation $\varepsilon_{c,k} = [\varepsilon_{c,k}^{(x)}, \varepsilon_{c,k}^{(y)}]$ is then processed by a deeper MLP with six hidden layers of width 256 and a final 256-dimensional output layer, yielding an embedding for each context pair. Context embeddings $\varepsilon_{c,k}$ are averaged to form a single aggregated representation ε_c of dimension 256. For prediction, each query input $x_{q,k}$ is concatenated with the context representation ε_c and passed through a decoder MLP consisting of six hidden layers of dimension 256. The decoder's final output layer has dimensionality $2d_y$, parameterizing the mean and the log-scale parameters of the Gaussian predictive distribution.

Attentive Conditional Neural Process (AttCNP). Our AttCNP implementation follows the deterministic architecture introduced by Kim et al. [2019]. The initial encoding stage mirrors that of the CNP: each context pair $(x_{c,k}, y_{c,k}) \in \mathcal{D}_c$ is processed by two separate MLPs (each with

two hidden layers of width 256), producing embeddings $\varepsilon_{c,k}^{(x)}$ and $\varepsilon_{c,k}^{(y)}$. These are concatenated to form $\varepsilon_{c,k} = [\varepsilon_{c,k}^{(x)}, \varepsilon_{c,k}^{(y)}]$, which is subsequently passed through an additional two-layer MLP with hidden width 256. Departing from the CNP, the AttCNP applies self-attention to the set of context embeddings. We use two layers of multi-head self-attention, each with 8 heads (head dimension 32), a feedforward subnetwork with hidden width 256, residual connections, and pre-layer normalization for both the attention and feedforward blocks. The input and output dimensionality of each attention layer is fixed at 256. Query representations $\varepsilon_{q,k}$ are computed using a single layer of multi-head cross-attention between the (attended) context embeddings and the query embeddings, with the same configuration as above (8 heads of dimension 32, feedforward width 256, residual connections, and pre-normalization). The decoder is an MLP with six hidden layers of width 256. Its final layer outputs $2d_y$ units, corresponding to the mean and log-scale parameters of a Gaussian predictive distribution.

Transformer Neural Process (TNP). To apply the TNP, we first construct token representations for both context and query points. For each context pair $(x_{c,k}, y_{c,k}) \in \mathcal{D}_c$, we form the token

$$[x_{c,k},\, y_{c,k},\, 1],$$

where the final singleton "1" serves as a density flag indicating that the observation $y_{c,k}$ at $x_{c,k}$ is available. For each query location $x_{q,k}$, we instead construct

$$[x_{q,k},\, \mathbf{0},\, 0],$$

where the zero-vector matches the shape of $y_{c,k}$ and the final "0" indicates the absence of an observation [Nguyen and Grover, 2022, Ashman et al., 2024a]. All tokens are then passed through a shared two-layer MLP (width 256), producing initial embeddings $\varepsilon_{c,k}$ and $\varepsilon_{q,k}$.

In the original TNP architecture [Nguyen and Grover, 2022], context and query embeddings are processed jointly by a transformer encoder. An attention mask prevents (i) interactions among queries and (ii) context→query attention, ensuring that context representations remain query-independent, while queries may attend to contexts. This design incurs quadratic complexity in the total sequence length,

$$\mathcal{O}((|\mathcal{D}_c| + |\mathcal{D}_q|)^2).$$

Because queries never attend to one another, Feng et al. [2022] introduced the Efficient Query TNP, which we adopt here. This variant first applies self-attention to context tokens to produce updated context embeddings, then processes each query using cross-attention over these updated contexts. The resulting two-branch structure (contexts processed twice, queries once) reduces the overall complexity to

$$\mathcal{O}\big(|\mathcal{D}_c|^2 + |\mathcal{D}_q|\,|\mathcal{D}_c|\big).$$

Our model uses six transformer layers with eight attention heads (head dimension 32), each followed by a feedforward subnetwork of width 256. We use the standard pre-norm architecture, applying layer normalization before both the attention and feedforward modules. All attention operations share a 256-dimensional embedding dimension. At each layer, we first apply context self-attention and then query–context cross-attention. Notably, the same multi-head attention parameters are used for both operations; the distinction arises only from which embeddings serve as queries, keys, and values (yielding self-attention when they coincide, and cross-attention otherwise). This is not a modeling requirement—separate parameter sets could be used, as demonstrated in Appendix B.4.1. The final-layer query embeddings are passed to a decoder MLP with two hidden layers (width 256), whose output parameterizes a Gaussian predictive distribution via $2d_y$ units corresponding to the mean and log-scale.

Translation Equivariant Transformer Neural Process (TETNP). The token construction for TETNP closely mirrors that of the TNP, with the key difference that token representations omit the input locations in order to preserve translation equivariance. For each context pair $(x_{c,k}, y_{c,k}) \in \mathcal{D}_c$, we therefore construct the token

$$[y_{c,k},\, 1],$$

and for each query location $x_{q,k}$ we use

$$[\mathbf{0},\, 0].$$

All tokens are processed by a shared two-layer MLP (width 256), yielding initial embeddings $\varepsilon_{c,k}$ and $\varepsilon_{q,k}$. TETNP replaces the standard multi-head attention mechanism with the translation-equivariant

variant introduced by Ashman et al. [2024a]. To compute attention scores, the pairwise scaled dot-product between token embeddings for each head is concatenated with the corresponding pairwise differences in token locations. This concatenated feature is passed through an MLP with two hidden layers (width 256) and an output dimension equal to the number of attention heads, producing translation-equivariant attention logits. Except for this modified attention module, the remaining architecture and computation flow follow the implementation of the TNP.

Convolutional Conditional Neural Process (ConvCNP): For ConvCNP, we begin by determining, for each input dimension, the minimum and maximum coordinates observed across both the context and query sets. These extrema are expanded by a small margin of 0.1 in each dimension. The interval between the expanded minima and maxima is then uniformly discretized at a resolution of 64 points per unit. When necessary—for example, to satisfy the CNN's minimum grid-size requirements—the discretization range is further enlarged while maintaining the same resolution. The resulting one-dimensional grids are combined via a Cartesian product to yield a uniform d_x-dimensional grid $\mathcal{G}$.

The functional embedding in equation 3 is evaluated on this grid. The Gaussian kernels used in this embedding are initialized with per-dimension length scales set to twice the grid resolution, i.e., $2/64$. For multi-dimensional inputs and outputs, the model uses separate length scales for each dimension. The embedding values are further normalized using a *density channel*, defined as

$$\mathrm{Density}(x) = \sum_{(x_c,\, y_c) \in \mathcal{D}_c} \psi_e(x - x_c).$$

The functional and density channels are concatenated to form the representation

$$\left(\mathrm{Density}(x_g),\ \frac{\sum_{(x_c,\, y_c) \in \mathcal{D}_c} \phi(y_c)\, \psi_e(x_g - x_c)}{\sum_{(x_c,\, y_c) \in \mathcal{D}_c} \psi_e(x_g - x_c)} \right)_{x_g \in \mathcal{G}}. \tag{5}$$

Each grid location is processed by an MLP with two hidden layers of width 128. The resulting features are passed to a CNN based on a U-Net architecture [Ronneberger et al., 2015], consisting of six residual convolutional blocks (kernel size 11, stride 2, 128 channels) in the encoder, followed by a symmetric sequence of six transposed convolutional blocks in the decoder, with skip connections following the design of Bruinsma et al. [2023]. Because the U-Net downsamples the spatial resolution by a factor of 64, we ensure that the constructed grid size is divisible by 64. For the one-dimensional benchmarks considered here, we therefore enlarge the discretization interval when necessary so that the CNN's output aligns exactly with the input grid size.

To obtain predictive distribution parameters at query locations—which may not lie on the grid $\mathcal{G}$—an interpolation scheme analogous to equation 5 is applied, except that the weighted sum over the feature-map values is not normalized. For this interpolation, we employ a separate Gaussian kernel with learnable length-scale parameters, distinct from the kernel used in the functional embedding. The resulting query-specific embeddings are then processed by a decoder MLP with two hidden layers (width 128), whose output consists of $2d_y$ units parameterizing the mean and log-scale of a Gaussian predictive distribution.

Spectral Convolutional Conditional Neural Process (SConvCNP) The SConvCNP replaces the ConvCNP's U-Net backbone with a U-shaped Fourier Neural Operator (FNO) architecture [Li et al., 2020a, Rahman et al., 2022a]. A residual Fourier block is denoted

$$\mathrm{F}(c_{\mathrm{in}},\, c_{\mathrm{out}},\, s_{\mathrm{in}},\, s_{\mathrm{out}},\, m_{\mathrm{f}}),$$

where c_{in} and c_{out} are channel dimensions, s_{in} and s_{out} are spatial sizes, and m_{f} is the number of retained Fourier modes.

The representation from equation 5, augmented with positional encodings (grid coordinates), is first processed by an MLP with one hidden layer of width 64 and GELU activations [Hendrycks and Gimpel, 2016]. Its output is then passed through the following sequence of residual Fourier blocks:

- $\mathrm{L}_1 = \mathrm{F}(64,\, 128,\, |\mathcal{G}|,\, \lfloor |\mathcal{G}|/2 \rfloor,\, 32)$
- $\mathrm{L}_2 = \mathrm{F}(128,\, 128,\, \lfloor |\mathcal{G}|/2 \rfloor,\, \lfloor |\mathcal{G}|/4 \rfloor,\, 32)$

- $\mathrm{L}_3 = \mathrm{F}(128,\ 256,\ \lfloor |\mathcal{G}|/4 \rfloor,\ 32)$
- $\mathrm{L}_4 = \mathrm{F}(256,\ 128,\ \lfloor |\mathcal{G}|/2 \rfloor,\ 32)$
- $\mathrm{L}_5 = \mathrm{F}(256,\ 128,\ |\mathcal{G}|,\ 32)$

Layers $\mathrm{L}_1 \to \mathrm{L}_2 \to \mathrm{L}_3 \to \mathrm{L}_4$ form the contractive–expansive path. The final block L_5 receives the channel-wise concatenation of the outputs of L_4 and L_1, yielding the U-shaped skip connection. Its output is concatenated with the initial MLP features and processed by a final MLP (one hidden layer, width 128, GELU) to produce the SConvCNP representation for the decoder.

Parameter Count. Table 5 summarizes the number of learnable parameters for all models evaluated in our 1D synthetic regression experiments.

Table 5: Learnable parameter counts for all models evaluated in the 1D synthetic regression experiments.

	CNP	AttCNP	TNP	TETNP	ConvCNP	SConvCNP
Number of parameters (million)	1.3	2.1	2.6	3.1	3.8	3.7

Forward run time. Table 6 reports the forward-pass runtime (in seconds) for a batch of 16 tasks during both training and validation across all models evaluated in our 1D synthetic regression experiments. As described in Appendix B.1.2, the number of query points is fixed to 256 during validation, whereas during training it is sampled uniformly from $\mathcal{U}[5, 25)$.

Table 6: Forward-pass runtime (in seconds) for a batch of 16 tasks during training and validation across all models in the 1D synthetic regression setting.

	CNP	AttCNP	TNP	TETNP	ConvCNP	SConvCNP
Train	0.003	0.004	0.007	0.014	0.009	0.009
Validation	0.006	0.009	0.013	0.023	0.033	0.030

B.1.2 Data and Experimental Setup

We evaluate our methods on four families of synthetic 1D regression tasks, each defined by a distinct stochastic generative process: a Gaussian process (GP) with a Matérn–5/2 kernel, a GP with a periodic kernel, a sawtooth-wave generator, and a square-wave generator. GP tasks are sampled using `GPyTorch` [Gardner et al., 2018]. All processes include independent Gaussian observation noise.

Process Parameters. For each task family, hyperparameters are sampled independently at the task level:

- **GP with Matérn–5/2 kernel.** The scaled distance is
$$d = \sqrt{(x - x')^\top \Lambda^{-2} (x - x')}, \qquad \Lambda = \lambda I, \quad \lambda \sim \mathcal{U}[0.25, 1).$$
The kernel is
$$k_{m2.5}(x, x') = \frac{2^{-1.5}}{\Gamma(2.5)} \, (\sqrt{5}\, d)^{2.5} K_{2.5}(\sqrt{5}\, d),$$
where $K_{2.5}$ is a modified Bessel function of the second kind. Observation noise has variance $\sigma_0^2 = 0.01$.
- **GP with periodic kernel.** The kernel is parameterized by period $\rho \sim \mathcal{U}[0.5, 2)$ and lengthscale $\lambda \sim \mathcal{U}[0.25, 1)$:
$$k_p(x, x') = \exp\left(-2 \sum_{i=1}^{d} \frac{\sin^2\left(\frac{\pi}{\rho}(x_i - x_i')\right)}{\lambda} \right),$$
with observation noise variance $\sigma_0^2 = 0.01$.

- **Sawtooth wave.** Piecewise-linear periodic functions with frequency $\xi \sim \mathcal{U}(0.5, 5)$, direction $u \in \{+1, -1\}$, and offset $c \sim \mathcal{U}[0, 1)$. Noise follows $\mathcal{N}(0, 0.0025)$.
- **Square wave.** Discontinuous periodic functions with frequency $\xi \sim \mathcal{U}(0.5, 5)$, duty cycle $D \sim \mathcal{U}[0.25, 0.75)$, and offset $c \sim \mathcal{U}[0, 1)$. Noise follows $\mathcal{N}(0, 0.0025)$.

Table 7: Synthetic generative processes used in 1D regression experiments.

Process	$f(x)$
GP (Matérn 5/2)	$f \sim \mathcal{GP}(0, k_{m2.5} + \sigma_0^2 I)$
GP (Periodic)	$f \sim \mathcal{GP}(0, k_p + \sigma_0^2 I)$
Sawtooth	$f(x) = 2\Big(\big(\xi(u^\top x) - c\big) \bmod 1\Big) + \sigma_0(x),\ \sigma_0(x) \sim \mathcal{N}(0, \sigma_0^2)$
Square Wave	$f(x) = 2\,\mathbb{1}_{\left\{\left((\xi x - c) \bmod 1\right) < D\right\}} - 1 + \sigma_0(x),\ \sigma_0(x) \sim \mathcal{N}(0, \sigma_0^2)$

Each model is trained for 500 epochs using AdamW [Loshchilov and Hutter, 2017] with a learning rate of 5×10^{-4}. We apply gradient clipping with a maximum norm of 0.5 [Pascanu et al., 2013]. Each epoch consists of 1000 iterations, and every iteration processes a batch of 16 tasks, yielding a total of 8 million on-the-fly sampled training tasks. For each batch, the numbers of context and query points are independently drawn as $|\mathcal{I}_c| \sim \mathcal{U}[5, 25)$ and $|\mathcal{I}_q| \sim \mathcal{U}[5, 25)$. These values are shared across all tasks in the batch. Input locations are sampled uniformly and independently from $[-3, 3)$ for each task.

For validation, we use a fixed set of 4,096 tasks, organized into 256 batches of 16 tasks. In these tasks, the number of query points is fixed at 256, while the number of context points is sampled using the same procedure as during training. Testing follows the same configuration as validation, except that we evaluate on 16,000 test tasks. Unlike the dynamically generated training tasks, the validation and test sets remain fixed across all experiments.

B.2 Predator-Prey Model

B.2.1 Model Architectures

The model architectures and parameter counts closely follow those described in B.1.1. The only modification concerns the discretization of the functional embedding in ConvCNP and SConvCNP. Specifically, we first expand the range defined by the minima and maxima of all context and query coordinates by a margin of 0.5 in each dimension. The resulting interval is then uniformly discretized at a resolution of 48 points per unit length.

B.2.2 Data and Experimental Setup

We generate data from a stochastic variant of the Lotka–Volterra predator–prey system [Lotka, 1910, Volterra, 1926], following the formulation of Bruinsma et al. [2023]. Let U_t and V_t denote the prey and predator populations at time t. Their dynamics evolve according to

$$dU_t = \alpha U_t\, dt - \beta U_t V_t\, dt + \sigma U_t^\nu\, dB_t^{(1)},$$
$$dV_t = -\gamma V_t\, dt + \delta U_t V_t\, dt + \sigma V_t^\nu\, dB_t^{(2)},$$

where $B_t^{(1)}$ and $B_t^{(2)}$ are independent Brownian motions. The deterministic drift recovers classical predator–prey behavior: prey grow exponentially at rate α in the absence of predators, while predators decline at rate γ without prey. The interaction terms $\beta U_t V_t$ and $\delta U_t V_t$ model predation and predator reproduction. Stochasticity enters through the multiplicative noise terms σU_t^ν and σV_t^ν, where σ controls noise intensity and ν determines how fluctuations scale with population size. Setting $\nu = 1$ yields noise proportional to population level, while $\nu < 1$ and $\nu > 1$ induce sublinear and superlinear scaling, respectively.

Table 8: Parameter distributions for the stochastic Lotka–Volterra equations.

Parameter	Distribution
Initial condition U_{-10}	$\mathcal{U}([5, 100])$
Initial condition V_{-10}	$\mathcal{U}([5, 100])$
α	$\mathcal{U}([1.0, 5.0])$
β	$\mathcal{U}([0.04, 0.08])$
γ	$\mathcal{U}([1.0, 2.0])$
δ	$\mathcal{U}([0.04, 0.08])$
σ	$\mathcal{U}([0.5, 10])$
η	$\mathcal{U}([1, 5])$
ν	Fixed at $1/6$

We simulate trajectories using the parameter distributions in Table 8 over a total of 110 years, discarding the first 10 years as burn-in to reduce sensitivity to initial conditions. Integration is performed using the Euler–Maruyama method with time step $\Delta t = 0.022$, producing 5000 solver steps over the interval $t \in [-10, 100)$. Although the solver runs at this finer resolution, we record states only on a uniform grid with spacing 0.05, yielding approximately 2200 selected time points spanning $[-10, 100)$.

Models are trained for 500 epochs using AdamW with a learning rate of 10^{-4}. We apply gradient clipping with a maximum norm of 0.5. Each epoch consists of 1,000 iterations, each processing a batch of 16 tasks, yielding a total of 8 million on-the-fly sampled training tasks. For every batch, the numbers of context and query points are drawn independently as $|\mathcal{I}_c| \sim \mathcal{U}[5, 25)$ and $|\mathcal{I}_q| \sim \mathcal{U}[5, 25)$; the sampled values are shared across all tasks within the batch.

For validation, we use a fixed set of 4,096 tasks, arranged into 256 batches of 16 tasks. In these tasks, the number of query points is fixed at 256, while the number of context points is sampled using the same procedure as in training. Testing follows the same protocol as validation, except that we evaluate on 16,000 test tasks.

To construct a task from a simulated trajectory, input locations t are sampled uniformly and independently from the recorded time points in $[0, 100)$ and paired with the corresponding values (U_t, V_t). We rescale time by a factor of 0.1, mapping the 100-year interval to $[0, 10)$, and rescale population sizes by multiplying the values by 0.01. Unlike the dynamically generated training tasks, the randomization used to construct the validation and test tasks is fixed across all experiments.

B.3 Traffic Flow

B.3.1 Model Architectures

Consistent with the Predator–Prey experiments described in Section B.2.1, the model architectures follow those introduced in Section B.1.1. For the discretization of the functional embedding in both ConvCNP and SConvCNP, we use a margin of 0.5 and a resolution of 48 grid points per unit length.

B.3.2 Data and Experimental Setup

We use the California traffic-flow dataset from LARGEST [Liu et al., 2023b], a large-scale benchmark for traffic forecasting available at `https://www.kaggle.com/datasets/liuxu77/largest`. The dataset provides 5-minute traffic-flow measurements from 8,600 loop-detector sensors across California's highway system, collected from 2017–2021. Our experiments focus on the year 2020.

We discard sensors with more than 50% missing values. For the remaining sensors, missing entries are linearly interpolated between observed measurements, with leading or trailing gaps filled using forward/backward propagation. Sensors are randomly partitioned into training, validation, and test splits using a 6:1:3 ratio.

Each sensor's time series is segmented into non-overlapping 14-day windows (4,032 time steps), yielding 26 windows per sensor. Within each window, timestamps are reset to start at 0 and increment

in 5-minute steps. We then downsample each window by a factor of 6, producing a 30-minute resolution. Time indices are rescaled to days by dividing by (60×24). Traffic-flow values are normalized to the $[0, 1]$ range using min–max statistics computed from the training set.

The resulting dataset contains:

- **Training:** 5,119 sensors $\rightarrow$ 133,094 windows
- **Validation:** 853 sensors $\rightarrow$ 22,178 windows
- **Test:** 2,561 sensors $\rightarrow$ 66,586 windows

Models are trained for 100 epochs using AdamW with a learning rate of 10^{-4}. We apply gradient clipping with a maximum norm of 0.5. Each epoch consists of approximately 4,160 iterations, each processing a batch of 32 tasks. For every batch, the numbers of context and query points are drawn independently as $|\mathcal{I}_c| \sim \mathcal{U}[5, 25)$ and $|\mathcal{I}_q| \sim \mathcal{U}[5, 25)$; the sampled values are shared across all tasks within the batch. A task is formed by independently sampling $|\mathcal{I}_c| + |\mathcal{I}_q|$ time steps from a window without replacement. Training tasks are sampled on-the-fly from training windows. For validation and testing, we pre-generate fixed sets of tasks:

- **Validation:** 22,178 tasks (batched in 32), with $|\mathcal{I}_q| = 256$ and $|\mathcal{I}_c|$ sampled as in training.
- **Test:** 66,586 tasks (batched in 32), with $|\mathcal{I}_q| = 50$ and $|\mathcal{I}_c|$ sampled as in training.

B.4 Image Completion

B.4.1 Model Architectures

All models output Gaussian predictive distributions parameterized by a mean and a pre-softplus standard deviation. Because pixel values are normalized to $[0, 1]$, the predicted mean is passed through a sigmoid. The predicted standard deviation is obtained by applying a softplus to the network output, scaling the result by 0.99, and adding a minimum noise of 0.01.

Conditional Neural Process (CNP). The CNP architecture follows the design used in previous experiments (see Section B.1.1), with the only modification being an increase in the MLP hidden size from 256 to 512.

Attentive Conditional Neural Process (AttCNP). The AttCNP architecture is likewise based on the design used in earlier experiments (see Section B.1.1). The MLP hidden size is increased from 256 to 512. For the self-attention applied to the set of context embeddings, we increase the per-head dimension from 32 to 64, set the input/output dimension of each attention layer to 512, and increase the feedforward subnetwork width from 256 to 512. The same modifications apply to the multi-head cross-attention used to update query embeddings (eight heads of dimension 64 and feedforward width 512).

Transformer Neural Process (TNP). As with the other models, all MLP hidden sizes are increased from 256 to 512. We adopt the Efficient Query TNP design described in Section B.1.1, which employs a two-branch structure in which context embeddings are processed twice and query embeddings once. In contrast to previous sections, we allocate *separate* parameters for the two branches: at each layer, the multi-head self-attention over contexts and the multi-head cross-attention between queries and contexts no longer share parameters. For each multi-head attention module, the feedforward width is increased from 256 to 512, and all input/output embedding dimensions are set to 512. All other configurations follow Section B.1.1.

Convolutional Conditional Neural Process (ConvCNP). Because the data lie on a regular grid, we use the on-the-grid implementation of ConvCNP [Gordon et al., 2019], eliminating the discretization step and improving computational efficiency. Let I denote an incomplete image, with unobserved pixels filled with dummy values, and let M_c be the binary mask indicating observed (context) pixels. For multi-channel images, the mask is broadcast along the channel dimension.

As in Section B.1.1, the output of the convolutional deep set module has two components: (i) a density channel capturing the spatial distribution of context pixels, and (ii) a kernel-smoothed representation.

The kernel is implemented via a 2D convolutional layer (kernel size 11, d_y input channels, 128 output channels, no bias) with a nonnegativity constraint enforced by taking absolute values of the learned weights during the forward pass [Gordon et al., 2019].

The density channel is obtained by convolving this modified filter with the mask M_c. The kernel-smoothed component is computed by first multiplying I elementwise with M_c (setting non-context pixels to zero) and then applying the same modified convolution. Following Gordon et al. [2019], we omit normalization by the density channel, as it did not provide empirical benefits and occasionally introduced instability; however, we retain the positivity constraint on the kernel.

The resulting representation is processed pointwise by an MLP with two hidden layers of width 128. These features are then fed into a ResNet-style CNN [He et al., 2016] consisting of six residual convolutional blocks (kernel size 11, 128 channels), following the implementation of Bruinsma et al. [2023]. The embeddings corresponding to query pixels are finally passed to a decoder MLP with two hidden layers of width 128.

Spectral Convolutional Conditional Neural Process (SConvCNP). The SConvCNP mirrors the on-grid ConvCNP architecture described above, with the distinction that the ResNet backbone is replaced by a U-shaped FNO. To construct positional information, we uniformly discretize the interval $[-1, 1]$ along each spatial axis into grids whose sizes match the corresponding image dimensions. Their Cartesian product yields the 2D positional encoding, which is concatenated with the output of the convolutional deep set. This concatenated tensor is then passed pointwise through an MLP with one hidden layer of width 128 and GELU activations.

Let s_h and s_w denote the image height and width, respectively. Using the notation introduced in Section B.1.1, the operator consists of the following sequence of residual Fourier blocks:

- $\mathrm{L}_1 = \mathrm{F}(512,\ 512,\ (s_h, s_w),\ (s_h, s_w),\ 32)$
- $\mathrm{L}_2 = \mathrm{F}(512,\ 512,\ (s_h, s_w),\ (\lfloor s_h/4 \rfloor,\ \lfloor s_w/4 \rfloor),\ 8)$
- $\mathrm{L}_3 = \mathrm{F}(512,\ 512,\ (\lfloor s_h/4 \rfloor,\ \lfloor s_w/4 \rfloor),\ (\lfloor s_h/16 \rfloor,\ \lfloor s_w/16 \rfloor),\ 2)$
- $\mathrm{L}_4 = \mathrm{F}(512,\ 512,\ (\lfloor s_h/16 \rfloor,\ \lfloor s_w/16 \rfloor),\ (\lfloor s_h/16 \rfloor,\ \lfloor s_w/16 \rfloor),\ 2)$
- $\mathrm{L}_5 = \mathrm{F}(512,\ 512,\ (\lfloor s_h/16 \rfloor,\ \lfloor s_w/16 \rfloor),\ (\lfloor s_h/4 \rfloor,\ \lfloor s_w/4 \rfloor),\ 8)$
- $\mathrm{L}_6 = \mathrm{F}(512,\ 512,\ (\lfloor s_h/4 \rfloor,\ \lfloor s_w/4 \rfloor),\ (s_h, s_w),\ 16)$
- $\mathrm{L}_7 = \mathrm{F}(512,\ 512,\ (s_h, s_w),\ (s_h, s_w),\ 32)$

The forward pass follows a standard U-shaped pattern. Layers $\mathrm{L}_1 \to \mathrm{L}_2 \to \mathrm{L}_3 \to \mathrm{L}_4$ are applied sequentially. On the upward path, L_5 receives the channel-wise concatenation of the outputs of L_4 and L_3. Similarly, L_6 receives the concatenation of the outputs of L_5 and L_2, and L_7 receives the concatenation of the outputs of L_6 and L_1. The output of L_7 is then concatenated with the features from the initial MLP and fed into a final MLP (one hidden layer, width 128, GELU) to yield the SConvCNP representation consumed by the decoder.

Parameter Count. Table 9 summarizes the number of learnable parameters for all models evaluated in our image completion experiments.

Table 9: Learnable parameter counts for all models evaluated in the image completion experiments.

	CNP	AttCNP	TNP	ConvCNP	SConvCNP
Number of parameters (million)	5	8.4	13.7	12.2	14.2

B.4.2 Data and Experimental Setup

For our image-completion experiments, we use the Describable Textures Dataset (DTD; Cimpoi et al. [2014]). We adopt the standard DTD split, which contains 1,880 images each for training, validation, and testing. All images are RGB, with heights ranging from 231 to 778 pixels and widths from 271 to 900 pixels. During training, validation, and testing, we sample from each image a random 192×192 crop, which we then resize to 64×64. Pixel coordinates of the resulting 64×64 grid along each

axis are linearly mapped to $[-1, 1]$, and pixel intensities are normalized to $[0, 1]$ independently across channels.

Models are trained for 500 epochs using AdamW with a learning rate of 10^{-4}. We apply gradient clipping with a maximum norm of 0.5. At each epoch, the model processes batches of 16 tasks. For each batch, we draw the number of context pixels as $|\mathcal{I}_c| \sim \mathcal{U}[5, 1024)$, and treat all remaining pixels as queries, i.e. $|\mathcal{I}_q| = 64 \times 64 - |\mathcal{I}_c|$. A single sampled value of $|\mathcal{I}_c|$ is shared across all tasks within the batch.

B.5 Ablation Studies

We performed a series of ablation studies to quantify the contribution of three key design choices in SCONVCNP: (i) the number of retained Fourier modes, (ii) the discretization resolution of the functional embedding, and (iii) the inclusion of positional encodings. Experiments were conducted on two 1D function families—samples from a Gaussian process with a Matérn–5/2 kernel and a sawtooth wave. To isolate the effect of each component, we used a slightly reduced-capacity SCONVCNP model (3.1M parameters, 32 Fourier modes). Training followed the same protocol as in Section B.1, and evaluation was performed on 8,096 test tasks. A summary of our findings is provided below.

Number of Fourier modes. Increasing the number of Fourier modes substantially improves performance on the sawtooth signal but yields negligible gains for Matérn–5/2 functions (Table 10). This difference is spectral: the sawtooth's Fourier amplitudes decay only as $1/\xi$, leaving substantial energy at high frequencies and requiring many modes for accurate representation. In contrast, Matérn–5/2 samples exhibit spectra decaying as $1/\xi^6$, with extremely weak high-frequency content; thus, a small number of low-frequency modes already captures nearly all signal energy.

Table 10: Predictive log-likelihood of SConvCNP for different numbers of Fourier modes m (4 seeds).

	$m = 8$	$m = 16$	$m = 32$
Matérn 5/2	$-0.29_{\pm 0.00}$	$-0.29_{\pm 0.00}$	$-0.29_{\pm 0.00}$
Sawtooth wave	$-0.14_{\pm 0.03}$	$0.20_{\pm 0.02}$	$0.80_{\pm 0.03}$

Discretization resolution. Finer discretization improves performance in the sawtooth case but has minimal effect for Matérn–5/2 samples (Table 11). The sawtooth's slowly decaying spectrum implies a high Nyquist frequency, so denser sampling is required to resolve its high-frequency components. In contrast, the Matérn–5/2 spectrum concentrates almost all energy at low frequencies; coarser discretizations already capture the relevant structure.

Table 11: Predictive log-likelihood of SConvCNP for different discretization resolutions (number of points per unit; 4 seeds).

	16	32	64
Matérn 5/2	$-0.30_{\pm 0.00}$	$-0.29_{\pm 0.00}$	$-0.29_{\pm 0.00}$
Sawtooth wave	$0.03_{\pm 0.02}$	$0.26_{\pm 0.08}$	$0.20_{\pm 0.02}$

Positional encoding. Adding positional encodings yields consistent improvements, particularly for the sawtooth wave (Table 12).

Table 12: Predictive log-likelihood of SConvCNP with and without positional encoding (4 seeds).

	with positional encoding	without positional encoding
Matérn 5/2	$-0.29_{\pm 0.00}$	$-0.31_{\pm 0.00}$
Sawtooth wave	$0.80_{\pm 0.03}$	$0.67_{\pm 0.02}$

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly state that we introduce Spectral Convolutional Conditional Neural Processes (SConvCNPs), explain the motivation for using spectral methods to address limitations in ConvCNPs, and accurately describe the scope of our experimental validation. The claims match the actual content and results presented in the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 3.1 discusses the limitations of our approach, including the trade-off between positional encodings and translation equivariance, and the discretization sensitivity of the FFT.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: This paper primarily presents a methodological contribution with empirical validation rather than theoretical results requiring formal proofs.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper provides detailed descriptions of the model architecture, data generation process, and experimental setup, ensuring that the main results can be reproduced. In addition, all implementations are publicly available in our code repository.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: All implementations are publicly available in our code repository.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper provides detailed descriptions of the model architecture, data generation process, and experimental setup

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: All experimental results in Tables 1-3 report standard deviations across 4 seeds, providing a clear measure of the statistical variability in our results.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification:The paper explicitly states that all computations were performed on a single NVIDIA A100 GPU with 40 GB of memory. Additionally, forward-pass runtimes for all models on one of the benchmarks are reported, providing further clarity on the execution time requirements.

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research centers on methodological advancements in neural network architectures for probabilistic meta-learning, relying solely on synthetic data and publicly available datasets. The work does not involve human subjects, sensitive data, or applications with foreseeable harmful impact, and it faithfully represents all results. Accordingly, it aligns with the NeurIPS Code of Ethics.

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No]

    Justification: The paper does not currently include a dedicated discussion of the broader societal impacts of the work, and therefore does not address potential positive or negative implications.

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: Our work focuses on meta-learning for regression tasks. The models and data used do not pose significant risks for misuse or dual-use concerns.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We have been meticulous in citing all prior work and assets used in our experiments. For each external codebase, dataset, or model, we provide proper attribution and ensure that the associated licenses and terms of use are respected.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The paper provides a detailed description of the experiments, and the complete code implementation is publicly released with accompanying documentation, ensuring that all newly introduced assets are well described and accessible.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: This research does not involve crowdsourcing or human subjects.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: This research does not involve human subjects, so IRB approval was not required.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: The research methodology does not involve the use of large language models as components of the proposed method. Any LLM assistance used was solely for writing and editing purposes and does not impact the scientific rigor or originality of the research.