Translation Equivariant Transformer Neural Processes

Matthew Ashman [1] **Cristiana Diaconu** [1] **Junhyuck Kim** [1] **Lakee Sivaraya** [1] **Stratis Markou** [1] **James Requeima** [2] **Wessel P. Bruinsma** [3] **Richard E. Turner** [1 3]

Abstract

The effectiveness of neural processes (NPs) in modelling posterior prediction maps—the mapping from data to posterior predictive distributions—has significantly improved since their inception. This improvement can be attributed to two principal factors: (1) advancements in the architecture of permutation invariant set functions, which are intrinsic to all NPs; and (2) leveraging symmetries present in the true posterior predictive map, which are problem dependent. Transformers are a notable development in permutation invariant set functions, and their utility within NPs has been demonstrated through the family of models we refer to as transformer neural processes (TNPs). Despite significant interest in TNPs, little attention has been given to incorporating symmetries. Notably, the posterior prediction maps for data that are stationary—a common assumption in spatio-temporal modelling—exhibit translation equivariance. In this paper, we introduce of a new family of translation equivariant TNPs (TE-TNPs) that incorporate *translation equivariance*. Through an extensive range of experiments on synthetic and real-world spatio-temporal data, we demonstrate the effectiveness of TE-TNPs relative to their non-translation-equivariant counterparts and other NP baselines.

1. Introduction

Transformers have emerged as an immensely effective architecture for natural language processing and computer vision tasks (Vaswani et al., 2017; Dosovitskiy et al., 2020). They have become the backbone for many state-of-the-art models—such ChatGPT (Achiam et al., 2023) and DALL-E (Betker et al., 2023)—owing to their ability to learn complex dependencies amongst input data. More generally, transformers can be understood as permutation equivariant set functions. This abstraction has led to the deployment of transformers in domains beyond that of sequence modelling, including particle physics, molecular modelling, climate science, and Bayesian inference (Lee et al., 2019; Fuchs et al., 2020; Müller et al., 2021).

NPs (Garnelo et al., 2018a;b) are a broad family of meta-learning models which learn the mapping from sets of observed datapoints to predictive stochastic processes (Foong et al., 2020). They are straightforward to train, handle off-the-grid data and missing observations with ease, and can be easily adapted for different data modalities. This flexibility makes them an attractive choice for a wide variety of problem domains, including spatio-temporal modelling, healthcare, and few-shot learning (Jha et al., 2022). Exchangeability in the predictive distribution with respect to the context set is achieved through the use of permutation invariant set functions, which, in NPs, map from the sets of observations to some representation space. Given the utility of transformers as set functions, it is natural to consider their use within NPs. This gives rise to TNPs.

The family of TNPs include the attentive NP (ANP) (Kim et al., 2019), diagonal TNP (TNP-D), autoregressive TNP (TNP-AR), and non-diagonal TNP (TNP-ND) (Nguyen & Grover, 2022), and the latent-bottlenecked ANP (LBANP) (Kim et al., 2019). Despite a significant amount of interest in TNPs from the research community, there are certain properties that we may wish our model to possess that have not yet been addressed. In particular, for spatio-temporal problems the data is often roughly stationary, in which case it is desirable to equip our model with translation equivariance: if the data are translated in space or time, then the predictions of our model should be translated correspondingly. Although translation equivariance has been incorporated into other families of NP models, such as the convolutional conditional NP (ConvCNP) (Gordon et al., 2019) and relational CNP (RCNP) (Huang et al., 2023), it is yet to be incorporated into the TNP. The key ingredient to achieving this is to establish effective translation equivariant attention layers that can be used in place of the standard attention layers

[1]Department of Engineering, University of Cambridge, Cambridge, UK [2]Vector Institute, University of Toronto, Toronto, Canada [3]Microsoft Research AI for Science, Cambridge, UK. Correspondence to: Matthew Ashman <mca39@cam.ac.uk>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

within the transformer encoder. In this paper, we develop the TE-TNP. Our contributions are as follows:

1. We develop an effective method for incorporating translation equivariance into the attention mechanism of transformers, developing the translation equivariant multi-head self attention (TE-MHSA) and translation equivariant multi-head cross attention (TE-MHCA) operations. These operations replace standard MHSA and MHCA operations within transformer encoders to obtain a new family of translation equivariant TNPs.

2. We use pseudo-tokens to reduce the quadratic computational complexity of TE-TNPs, developing translation equivariant PT-TNPs (TE-PT-TNPs).

3. We demonstrate the efficacy of TE-TNPs relative to existing NPs—including the ConvCNP and the RCNP—on a number of synthetic and real-world spatio-temporal modelling problems.

2. Background

Throughout this section, we will use the following notation. Let $\mathcal{X} = \mathbb{R}^{D_x}$, $\mathcal{Y} = \mathbb{R}^{D_y}$ denote the input and output spaces, and let $(\mathbf{x}, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$ denote an input–output pair. Let $\mathcal{S} = \bigcup_{N=0}^{\infty} (\mathcal{X} \times \mathcal{Y})^N$ be a collection of all finite data sets, which includes the empty set $\varnothing$, the data set containing no data points. We denote a context and target set with $\mathcal{D}_c,\ \mathcal{D}_t \in \mathcal{S}$, where $|\mathcal{D}_c| = N_c$, $|\mathcal{D}_t| = N_t$. Let $\mathbf{X}_c \in \mathbb{R}^{N_c \times D_x}$, $\mathbf{Y}_c \in \mathbb{R}^{N_c \times D_y}$ be the inputs and corresponding outputs of $\mathcal{D}_c$, with $\mathbf{X}_t \in \mathbb{R}^{N_t \times D_x}$, $\mathbf{Y}_t \in \mathbb{R}^{N_t \times D_y}$ defined analogously. We denote a single task as $\xi = (\mathcal{D}_c, \mathcal{D}_t) = ((\mathbf{X}_c, \mathbf{Y}_c), (\mathbf{X}_t, \mathbf{Y}_t))$. Let $\mathcal{P}(\mathcal{X})$ denote the collection of stochastic processes on $\mathcal{X}$.

2.1. Neural Processes

NPs (Garnelo et al., 2018a;b) aim to learn the mapping from context sets $\mathcal{D}_c$ to ground truth posterior distributions over the target outputs, $\mathcal{D}_c \mapsto p(\mathbf{Y}_t | \mathbf{X}_t, \mathcal{D}_c)$, using meta-learning. This mapping is known as the *posterior prediction map* $\pi_P : \mathcal{S} \to \mathcal{P}(\mathcal{X})$, where P denotes the ground truth stochastic process over functions mapping from $\mathcal{X}$ to $\mathcal{Y}$. Common to all NP architectures is an encoder and decoder. The encoder maps from $\mathcal{D}_c$ and $\mathbf{X}_t$ to some representation, $e(\mathcal{D}_c, \mathbf{X}_t)$.[1] The decoder takes as input the representation and target inputs $\mathbf{X}_t$ and outputs $d(\mathbf{X}_t, e(\mathcal{D}_c, \mathbf{X}_t))$, which are the parameters of the predictive distribution over the target outputs $\mathbf{Y}_t$: $p(\mathbf{Y}_t | \mathbf{X}_t, \mathcal{D}_c) = p(\mathbf{Y}_t | d(\mathbf{X}_t, e(\mathcal{D}_c, \mathbf{X}_t)))$. An important requirement of the predictive distribution is permutation invariance with respect to the elements of $\mathcal{D}_c$.

[1] In many NP architectures, including the original conditional NP (CNP) and NP, the representation does not depend on the target inputs $\mathbf{X}_t$.

We shall focus on CNPs (Garnelo et al., 2018a), which factorise the predictive distribution as $p(\mathbf{Y}_t | \mathbf{X}_t, \mathcal{D}_c) = \prod_{n=1}^{N_t} p(\mathbf{y}_{t,n} | d(\mathbf{x}_{t,n}, e(\mathcal{D}_c, \mathbf{x}_{t,n})))$. CNPs are trained by maximising the posterior predictive likelihood:

$$\mathcal{L}_{\text{ML}} = \mathbb{E}_{p(\xi)} \left[\sum_{n=1}^{N_t} \log p(\mathbf{y}_{t,n} | d(\mathbf{x}_{t,n}, e(\mathcal{D}_c, \mathbf{x}_{t,n}))) \right]. \quad (1)$$

Here, the expectation is taken with respect to the distribution of tasks $p(\xi)$. As shown in Foong et al. (2020), the global maximum is achieved if and only if the model recovers the ground-truth posterior prediction map. When training a CNP, we often approximate the expectation with an average over the finite number of tasks available.

2.2. Transformers

A useful perspective is to understand transformers as a permutation equivariant set function f.[2] They take as input a set of N tokens, $\mathbf{Z}^0 \in \mathbb{R}^{N \times D_z}$, output a set of N tokens of the same cardinality: $f \colon (\mathbb{R}^{D_z})^N \to (\mathbb{R}^{D_z})^N$. If the input set is permuted, then the output set is permuted accordingly: $f(\mathbf{z}_1, \ldots, \mathbf{z}_N)_n = f(\mathbf{z}_{\sigma(1)}, \ldots, \mathbf{z}_{\sigma(N)})_{\sigma(n)}$ for all permutations $\sigma \in \mathbb{S}^N$ of N elements. At the core of each layer of the transformer architecture is the multi-head self attention (MHSA) operation (Vaswani et al., 2017). Let $\mathbf{Z}^\ell \in \mathbb{R}^{N \times D_z}$ denote the input set to the ℓ-th MHSA operation. The MHSA operation updates the nth token $\mathbf{z}_n^\ell$ as

$$\tilde{\mathbf{z}}_n^\ell = \mathrm{cat}\left(\left\{ \sum_{m=1}^{N} \alpha_h^\ell(\mathbf{z}_n^\ell, \mathbf{z}_m^\ell) \mathbf{z}_m^{\ell\,T} \mathbf{W}_{V,h}^\ell \right\}_{h=1}^{H^\ell} \right) \mathbf{W}_O^\ell \quad (2)$$

where cat denotes the concatenation operation across the last dimension. Here, $\mathbf{W}_{V,h}^\ell \in \mathbb{R}^{D_z \times D_V}$ and $\mathbf{W}_O^\ell \in \mathbb{R}^{H^\ell D_V \times D_z}$ are the value and projection weight matrices, where H^ℓ denotes the number of 'heads' in layer ℓ. Note that permutation equivariance is achieved through the permutation invariant summation operator. As this is the only mechanism through which the tokens interact with each other, permutation equivariance for the overall model is ensured. The attention mechanism, α_h^ℓ, is implemented as

$$\alpha_h^\ell(\mathbf{z}_n^\ell, \mathbf{z}_m^\ell) = \frac{e^{\mathbf{z}_n^{\ell\,T} \mathbf{W}_{Q,h}^\ell \left[\mathbf{W}_{K,h}^\ell\right]^T \mathbf{z}_m^\ell}}{\sum_{m=1}^{N} e^{\mathbf{z}_n^{\ell\,T} \mathbf{W}_{Q,h}^\ell \left[\mathbf{W}_{K,h}^\ell\right]^T \mathbf{z}_m^\ell}} \quad (3)$$

where $\mathbf{W}_{Q,h}^\ell \in \mathbb{R}^{D_z \times D_{QK}}$ and $\mathbf{W}_{K,h}^\ell \in \mathbb{R}^{D_z \times D_{QK}}$ are the query and key weight matrices. The softmax-normalisation ensures that $\sum_{m=1}^{N} \alpha_h^\ell(\mathbf{z}_n^\ell, \mathbf{z}_m^\ell) = 1\ \forall n, h, \ell$. Often, conditional independencies amongst the set of tokens—in

[2] Note that not all permutation equivariant set functions can be represented by transformers. For example, the family of informers (Garnelo & Czarnecki, 2023) cannot be represented by transformers, yet are permutation equivariant set functions. However, transformers are universal approximators of permutation equivariant set functions (Lee et al., 2019; Wagstaff et al., 2022).

the sense that the set $\{\mathbf{z}_n^\ell\}_{\ell=1}^{\ell=L}$ do not depend on the set $\{\mathbf{z}_m^\ell\}_{\ell=1}^{\ell=L}$ given some other set of tokens for some $n,\ m \in \{1, \dots, N\}$—are desirable. Whilst this is typically achieved through masking, if the same set of tokens are conditioned on for every n, then it is more computationally efficient to use multi-head cross attention (MHCA) operations together with MHSA operations than it is to directly compute Equation 2. The MHCA operation updates the nth token $\mathbf{z}_n^\ell$ using the set of tokens $\{\hat{\mathbf{z}}_m^\ell\}_{m=1}^M$ as

$$\tilde{\mathbf{z}}_n^\ell = \text{cat}\left(\left\{\sum_{m=1}^M \alpha_h^\ell(\mathbf{z}_n^\ell, \hat{\mathbf{z}}_m^\ell)\hat{\mathbf{z}}_m^{\ell\,T}\mathbf{W}_{V,h}^\ell\right\}_{h=1}^{H^\ell}\right)\mathbf{W}_O^\ell. \quad (4)$$

Note that all tokens updated in this manner are conditionally independent of each other given $\{\hat{\mathbf{z}}_m^\ell\}_{m=1}^M$. We discuss this in more detail in Appendix B. MHCA operations are at the core of the pseudo-token-based transformers such as the perceiver (Jaegle et al., 2021) and induced set transformer (IST) (Lee et al., 2019). We describe these differences in the following section.

MHSA and MHCA operations are used in combination with layer-normalisation operations and pointwise MLPs to obtain MHSA and MHCA blocks. Unless stated otherwise, we shall adopt the order used by Vaswani et al. (2017).

2.3. Pseudo-Token-Based Transformers

Pseudo-token based transformers reduce the quadratic computational complexity of the standard transformer through the use of pseudo-tokens. Concretely, let $\mathbf{U} \in \mathbb{R}^{M \times D_z}$ denote an initial set of $M \ll N$ tokens we call pseudo-tokens. There are two established methods for incorporating information about the set of observed tokens ($\mathbf{Z}$) into these pseudo-tokens in a computationally efficient manner: the perceiver-style approach of Jaegle et al. (2021) and the IST style approach of Lee et al. (2019). The perceiver-style approach iterates between applying $\text{MHCA}(\mathbf{U}^\ell, \mathbf{Z}^\ell)$ and $\text{MHSA}(\mathbf{U}^\ell)$, outputting a set of M pseudo-tokens, and has a computational complexity of $\mathcal{O}(MN)$ at each layer. The IST-style approach iterates between applying $\text{MHCA}(\mathbf{U}^\ell, \mathbf{Z}^\ell)$ and $\text{MHCA}(\mathbf{Z}^\ell, \mathbf{U}^\ell)$, outputting a set of N tokens and M pseudo-tokens, and also has a computational complexity of $\mathcal{O}(MN)$ at each layer. We provide illustrations these differences Appendix C.

2.4. Transformer Neural Processes

Given the utility of transformers as set functions, it is natural to consider their use in the encoder of a NP—we describe this family of NPs as TNPs. Let $\mathbf{Z}_c^0 \in \mathbb{R}^{N_c \times D}$ denote the initial set-of-token representation of each input-output pair $(\mathbf{x}_{c,n}, \mathbf{y}_{c,n}) \in \mathcal{D}_c$, and $\mathbf{Z}_{t,n}^0 \in \mathbb{R}^{N_t \times D}$ denote the initial set-of-token representation of each input $\mathbf{x}_{t,n} \in \mathbf{X}_t$. The encoding $e(\mathcal{D}_c, \mathbf{X}_t)$ of TNPs is is achieved by passing the union of initial context and target tokens, $\mathbf{Z}^0 = [\mathbf{Z}_c^0,\ \mathbf{Z}_t^0]$, through a transformer-style architecture, and keeping only the output tokens corresponding to the target inputs, $\mathbf{Z}_t^L$.

The specific transformer-style architecture is unique to each TNP variant. However, they generally consist of MHSA operations acting on the context tokens and MHCA operations acting to update the target tokens, given the context tokens.[3] The combination of MHSA and MHCA operations is a permutation invariant function with respect to the context tokens. We provide an illustration of this in Figure 1a. Enforcing these conditional independencies ensures that the final target token $\mathbf{z}_{t,n}^L$ depends only on $\mathcal{D}_c$ and $\mathbf{x}_{t,n}$, i.e. $[e(\mathcal{D}_c, \mathbf{X}_t)]_n = e(\mathcal{D}_c, \mathbf{x}_{t,n})$. This is required for the factorisation of the predictive distribution $p(\mathbf{Y}_t|\mathbf{X}_t, \mathcal{D}_c) = \prod_{n=1}^{N_t} p(\mathbf{y}_{t,n}|d(\mathbf{x}_{t,n}, e(\mathcal{D}_c, \mathbf{x}_{t,n})))$. We denote pseudo-token TNPs (PT-TNPs) as the family of TNPs which use pseudo-token based transformers. Currently, this family is restricted to the LBANP, which uses a perceiver-style architecture; however, it is straightforward to use an IST-style architecture instead.

2.5. Translation Equivariance

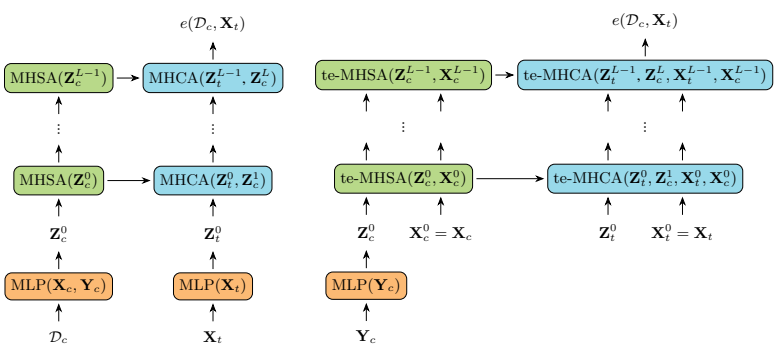
Here, we provide new theoretical results which show the importance of translation equivariance as an inductive bias in NPs. In particular, we first show that *if, and only if,* the ground-truth stochastic process is stationary, the corresponding predictive map is translation equivariant (Theorem 2.1). Second, we show the importance of translation equivariance in the ability of our models to generalise to settings outside of the training distribution (Theorem 2.2), for which Figure 3 provides some intuition.

Let $\mathsf{T}_{\boldsymbol{\tau}}$ denote a translation by $\boldsymbol{\tau} \in \mathbb{R}^{D_x}$. For a data set $\mathcal{D} \in \mathcal{S}$, $\mathsf{T}_{\boldsymbol{\tau}}\mathcal{D} \in \mathcal{S}$ translates the data set by adding $\boldsymbol{\tau}$ to all inputs. For a function $f\colon \mathcal{X} \to Z$, $\mathsf{T}_{\boldsymbol{\tau}} f$ translates f by producing a new function $\mathcal{X} \to Z$ such that $\mathsf{T}_{\boldsymbol{\tau}} f(\mathbf{x}) = f(\mathbf{x} - \boldsymbol{\tau})$ for all $\mathbf{x} \in \mathbb{R}^{D_x}$. For a stochastic process $\mu \in \mathcal{P}(\mathcal{X})$, $\mathsf{T}_{\boldsymbol{\tau}}(\mu)$ denotes the pushforward measure of pushing μ through $\mathsf{T}_{\boldsymbol{\tau}}$. A *prediction map* π is a mapping $\pi\colon \mathcal{S} \to \mathcal{P}(\mathcal{X})$ from data sets $\mathcal{S}$ to stochastic processes $\mathcal{P}(\mathcal{X})$. Prediction maps are mathematical models of neural processes. Say that *a prediction map π is translation equivariant* if $\mathsf{T}_{\boldsymbol{\tau}} \circ \pi = \pi \circ \mathsf{T}_{\boldsymbol{\tau}}$ for all translations $\boldsymbol{\tau} \in \mathbb{R}^{D_x}$.

The ground-truth stochastic process P is stationary if and only if the prediction map π_P is translation equivariant. Foong et al. (2020) provide a simple proof of the "only if"-direction. We provide a rigorous proof in both directions. Consider $\mathcal{D} \in \mathcal{S}$. Formally define $\pi_P(\mathcal{D})$ by integrating P against a density $\pi'_P(\mathcal{D})$ that depends on $\mathcal{D}$, so $\mathrm{d}\pi_P(\mathcal{D}) = \pi'_P(\mathcal{D})\,\mathrm{d}P$.[4] Assume that $\pi'_P(\varnothing) \propto 1$, so $\pi_P(\varnothing) = P$. Say

[3] As discussed in Section 2.2, this is often implemented as a single MHSA operation with masking operating.

[4] Intuitively, $\pi'_P(\mathcal{D})(f) = p(\mathcal{D}|f)/p(\mathcal{D})$, so $\pi'_P(\mathcal{D})(f)$ is the



(a) TNP. **(b)** TE-TNP.

Figure 1. Block diagrams illustrating the TNP and TE-TNP encoder architectures. For both models, we pass individual datapoints through pointwise MLPs to obtain the initial token representations, $\mathbf{Z}_c^0$ and $\mathbf{Z}_t^0$. These are then passed through multiple attention layers, with the context tokens interacting with the target tokens through cross-attention. The output of the encoder depends on $\mathcal{D}_c$ *and* $\mathbf{X}_t$. The TE-TNP encoder updates the input locations at each layer, in addition to the tokens.

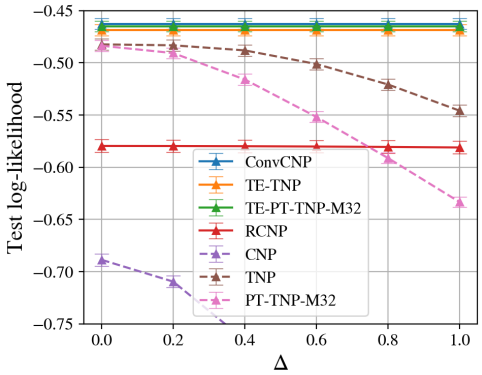


Figure 2. Average log-likelihood (↑) on the test datasets for the synthetic 1-D regression experiment. Δ denotes the amount by which the range from which the context and target inputs and sampled from is shifted at test time. Standard errors are shown.

that π'_P is *translation invariant* if, for all $\mathcal{D} \in \mathcal{S}$ and $\boldsymbol{\tau} \in \mathcal{X}$, $\pi'_P(\mathsf{T}_{\boldsymbol{\tau}}\mathcal{D}) \circ \mathsf{T}_{\boldsymbol{\tau}} = \pi'_P(\mathcal{D})$ P–almost surely.[5]

Theorem 2.1. *(1) The ground-truth stochastic process P is stationary and π'_P is translation invariant if and only if (2) π_P is translation equivariant.*

See Appendix E for the proof. If the ground-truth stochastic process is stationary, it is helpful to build translation equivariance into the neural process: this greatly reduces the model space to search over, which can significantly improve data efficiency (Foong et al., 2020). In addition, it is possible to show that translation equivariant NPs generalise spatially. We formalise this in the following theorem, which we present in the one-dimensional setting ($D_x = 1$) for notational simplicity, and we provide an illustration of these ideas in Figure 3.

Definitions for theorem. For a stochastic process $f \sim \mu$ with $\mu \in \mathcal{P}(\mathcal{X})$, for every $\mathbf{x} \in \mathbb{R}^N$, denote the distribution of $(f(x_1), \ldots, f(x_N))$ by $P_{\mathbf{x}}\mu$. We now define the notion of the *receptive field*. For two vectors of inputs $\mathbf{x}_1 \in \mathbb{R}^{N_1}$, $\mathbf{x}_2 \in \mathbb{R}^{N_2}$, and $R > 0$, let $\mathbf{x}_1|_{\mathbf{x}_2,R}$ be the subvector of $\mathbf{x}_1$ with inputs at most distance R away from any input in $\mathbf{x}_2$. Similarly, for a data set $\mathcal{D} = (\mathbf{x}, \mathbf{y}) \in \mathcal{S}$, let $\mathcal{D}|_{\mathbf{x}_2,R} \in \mathcal{S}$ be the subset of data points of $\mathcal{D}$ with inputs at most distance R away from any input in $\mathbf{x}_2$. With these definitions, say that a *stochastic process $f \sim \mu$ with $\mu \in \mathcal{P}(\mathcal{X})$ has receptive field* $R > 0$ if, for all $N_1, N_2 \in \mathbb{N}$, $\mathbf{x}_1 \in \mathbb{R}^{N_1}$, and $\mathbf{x}_2 \in \mathbb{R}^{N_2}$, $f(\mathbf{x}_2) \mid f(\mathbf{x}_1) \overset{\mathrm{d}}{=} f(\mathbf{x}_2) \mid f(\mathbf{x}_1|_{\mathbf{x}_2,\frac{1}{2}R})$. Intuitively, f only has local dependencies. Moreover, say that a *prediction map $\pi \colon \mathcal{S} \to \mathcal{P}(\mathcal{X})$ has receptive field $R > 0$* if, for all $\mathcal{D} \in \mathcal{S}$, $N \in \mathbb{N}$, and $\mathbf{x} \in \mathbb{R}^N$, $P_{\mathbf{x}}\pi(\mathcal{D}) = P_{\mathbf{x}}\pi(\mathcal{D}|_{\mathbf{x},\frac{1}{2}R})$. Intuitively, predictions by the neural process π are only influenced by context points at most distance $\frac{1}{2}R$ away.[6]

Theorem 2.2. *Let $\pi_1, \pi_2 \colon \mathcal{S} \to \mathcal{P}(\mathcal{X})$ be translation equivariant prediction maps with receptive field $R > 0$. Assume that, for all $\mathcal{D} \in \mathcal{S}$, $\pi_1(\mathcal{D})$ and $\pi_2(\mathcal{D})$ also have receptive field $R > 0$. Let $\epsilon > 0$ and fix $N \in \mathbb{N}$. Assume that, for all $\mathbf{x} \in \bigcup_{n=1}^{N} [0, 2R]^n$ and $\mathcal{D} \in \mathcal{S} \cap \bigcup_{n=0}^{\infty} ([0, 2R] \times \mathbb{R})^n$,*

$$\mathrm{KL}\left[P_{\mathbf{x}}\pi_1(\mathcal{D}) || P_{\mathbf{x}}\pi_2(\mathcal{D}))\right] \leq \epsilon. \tag{5}$$

Then, for all $M > 0$, $\mathbf{x} \in \bigcup_{n=1}^{N} [0, M]^n$, and $\mathcal{D} \in \mathcal{S} \cap \bigcup_{n=0}^{\infty} ([0, M] \times \mathbb{R})^n$,

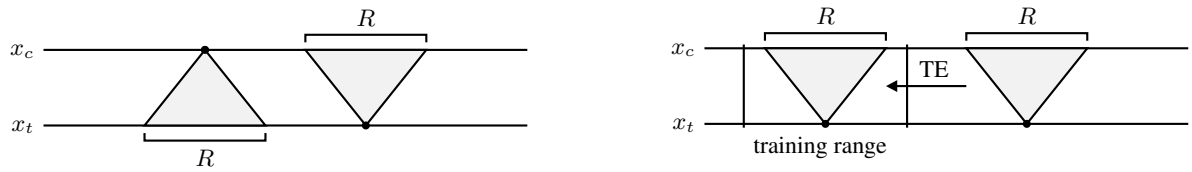
$$\mathrm{KL}\left[P_{\mathbf{x}}\pi_1(\mathcal{D}) || P_{\mathbf{x}}\pi_2(\mathcal{D}))\right] \leq \lceil 2M/R \rceil \epsilon. \tag{6}$$

See Appendix E for the proof. The notion of receptive field is natural to CNNs and corresponds to the usual notion of the receptive field. The notion is also inherent to transformers that adopt sliding window attention: the size of the window multiplied by the number of transformer layers gives the receptive field of the model. Intuitively, this theorem states that, if (a) the ground-truth stochastic process and our model are translation equivariant and (b) everything has receptive field size $R > 0$, then, whenever our model is accurate on $[0, 2R]$, it is also accurate on any bigger interval $[0, M]$. Note that this theorem accounts for dependencies between target points, so it also applies to latent-variable neural processes (Garnelo et al., 2018b) and Gaussian neural

modelling assumption that specifies how observations are generated from the ground-truth stochastic process. A simple example is $\pi'_P(\mathcal{D})(f) \propto \prod_{(\mathbf{x},\mathbf{y}) \in \mathcal{D}} \mathcal{N}(\mathbf{y} \mid f(\mathbf{x}), \sigma^2)$, which adds independent Gaussian noise with variance σ^2.

[5] For example, the usual Gaussian likelihood is translation invariant: $\mathcal{N}(\mathbf{y} \mid (\mathsf{T}_{\boldsymbol{\tau}} f)(\mathbf{x} + \boldsymbol{\tau}), \sigma^2) = \mathcal{N}(\mathbf{y} \mid f(\boldsymbol{\tau}), \sigma^2)$.

[6] For example, if f is a Gaussian process with a kernel compactly supported on $[-\frac{1}{2}R, \frac{1}{2}R]$, then the mapping $\mathcal{D} \mapsto p(f \mid \mathcal{D})$ is a prediction map which (a) has receptive field R and (b) maps to stochastic processes with receptive field R.



(a) For a model with receptive field $R > 0$, a context point at x_c influences predictions at target inputs only limitedly far away. Conversely, a prediction at a target input x_t is influenced by context points only limitedly far away.

(b) If a model is translation equivariant, then all context points and targets inputs can simultaneously be shifted left or right without changing the output of the model. Intuitively, this means that triangles in the figures can just be "shifted left or right".

Figure 3. Translation equivariance in combination with a limited receptive field (see (a)) can help generalisation performance. Consider a translation equivariant (TE) model which performs well within a *training range* (see (b)). Consider a prediction for a target input outside the training range (right triangle in (b)). If the model has receptive field $R > 0$ and the training range is bigger than R, then TE can be used to "shift that prediction back into the training range" (see (b)). Since the model performs well within the training range, the model also performs well for the target input outside the training range.

processes (Bruinsma et al., 2021; Markou et al., 2022). In practice, we do not explicitly equip our transformer neural processes with a fixed receptive field size by using sliding window attention, although this would certainly be possible. The main purpose of this theorem is to elucidate the underlying principles that enable translation equivariant neural processes to generalise.

3. Translation Equivariant Transformer Neural Processes

Let $\mathbf{Z}^\ell \in \mathbb{R}^{N \times D_z}$ and $\tilde{\mathbf{Z}}^\ell \in \mathbb{R}^{N \times D_z}$ denote the inputs and outputs at layer ℓ, respectively. To achieve translation equivariance, we must ensure that each token $\mathbf{z}_n^\ell$ is translation invariant with respect to the inputs, which can be achieved through dependency solely on the pairwise distances $\mathbf{x}_i - \mathbf{x}_j$ (see Appendix D). We choose to let our initial token embeddings $\mathbf{z}_n^0$ depend solely on the corresponding output $\mathbf{y}_n$, and introduce dependency on the pairwise distances through the attention mechanism. For permutation and translation equivariant operations, we choose updates of the form $\tilde{\mathbf{z}}_n^\ell = \phi\left(\oplus_{m=1}^N \psi\left(\mathbf{z}_n^\ell, \mathbf{z}_m^\ell, \mathbf{x}_n - \mathbf{x}_m\right)\right)$, where $\oplus$ is a permutation invariant operation. Adopting the MHSA approach, we instantiate this as

$$\tilde{\mathbf{z}}_n^\ell = \mathrm{cat}\left(\left\{ \textstyle\sum_{m=1}^N \alpha_{h,n,m}^\ell \mathbf{z}_m^{\ell\,T} \mathbf{W}_{V,h}^\ell \right\}_{h=1}^{H^\ell}\right) \mathbf{W}_O^\ell \tag{7}$$

where $\alpha_{h,n,m}^\ell = \alpha_h^\ell(\mathbf{z}_n^\ell, \mathbf{z}_m^\ell, \mathbf{x}_n - \mathbf{x}_m)$, the attention mechanism, depends on the difference $\mathbf{x}_n - \mathbf{x}_m$ as well as the input tokens $\mathbf{z}_n^\ell$ and $\mathbf{z}_m^\ell$. This differs to the standard attention mechanism in Equation 2, which depends solely on the input tokens. There exist a number of possible choices for the attention mechanism. Again, following the MHSA approach, we implement this as

$$\alpha_{h,n,m}^\ell = \frac{e^{\rho_h^\ell\left(\mathbf{z}_n^{\ell\,T} \mathbf{W}_{Q,h}^\ell \left[\mathbf{W}_{K,h}^\ell\right]^T \mathbf{z}_m^\ell, \mathbf{x}_n - \mathbf{x}_m\right)}}{\sum_{n=1}^N e^{\rho_h^\ell\left(\mathbf{z}_n^{\ell\,T} \mathbf{W}_{Q,h}^\ell \left[\mathbf{W}_{K,h}^\ell\right]^T \mathbf{z}_m^\ell, \mathbf{x}_n - \mathbf{x}_m\right)}} \tag{8}$$

where $\rho_h^\ell \colon \mathbb{R} \times \mathbb{R}^{D_x} \to \mathbb{R}$ is a learnable function, which we parameterise by an MLP.[7] We can also choose to update the input $\mathbf{x}_n$ used in Equation 8 with a function $\mathbf{f}_n^\ell(\{\mathbf{x}_m\}_{m=1}^N)$, which itself needs to satisfy both translation equivariance and permutation invariance with respect to the set of inputs. A general form for functions satisfying translation equivariance and permutation invariance is

$$\mathbf{f}_n^\ell(\{\mathbf{x}_m\}_{m=1}^N) = \textstyle\sum_{i=1}^N b_{ni} \left(\mathbf{x}_i + g_n\left(\sum_{j=1}^N h_n(\mathbf{x}_i - \mathbf{x}_j)\right)\right) \tag{9}$$

where $b_{n1}, \ldots, b_{nN} \in \mathbb{R}$ are *any* set of weights, possibly negative or even dependent on $\mathbf{x}_1, \ldots, \mathbf{x}_N$, that satisfy $\sum_{i=0}^M b_{ni} = 1$. See Appendix D for proof. A convenient choice is similar to that used by Satorras et al. (2021):

$$\mathbf{x}_n^{\ell+1} = \mathbf{x}_n^\ell + C \textstyle\sum_{h=1}^{H^\ell} \sum_{m=1}^N (\mathbf{x}_n^\ell - \mathbf{x}_m^\ell) \phi_h^\ell\left(\alpha_{h,n,m}^\ell\right) \tag{10}$$

where we have reused the computations of the attention-mechanism. Again, ϕ_h^ℓ is a learnable function, typically parameterised by an MLP, and C is a constant which we choose to be $1/N$. For the MHCA layers, we update the target inputs using the context points only.[8]

The attention mechanism defined in Equation 8 in combination with Equation 7 defines the TE-MHSA operation, which together with layer normalisation and pointwise MLPs described in Section 2.2 forms a TE-MHSA block. We can define a TE-MHCA operation in an identical manner to the MHCA operation in place of masking when conditional independencies are required. The TE-TNP shares an identical architecture to the regular TNP, with MHSA and MHCA blocks replaced by TE-MHSA and TE-MHCA blocks, which include both the translation equivariant operation and input location update steps. We illustrate the TE-TNP and TNP encoders in Figure 1.

[7] We implement $\left\{\rho_h^\ell\right\}_{h=1}^H$ as a single MLP with H output dimensions.

[8] i.e., $\mathbf{x}_{t,n}^{\ell+1} = \mathbf{x}_{t,n}^\ell + C \sum_h \sum_m (\mathbf{x}_{t,n}^\ell - \mathbf{x}_{c,n}^\ell) \phi_h^\ell(\alpha_{h,n,m}^\ell)$.

It is worthwhile highlighting the connections between the translation equivariant attention mechanism and the use of relative positional encodings (e.g. RoPE (Su et al., 2024)) in transformer-based large language models. Indeed, relative positional encodings are a specific implementation of our translation equivariant attention mechanism applied to a discrete, regular input domain that words exist on. Our work builds upon these methods, enabling relative positional encodings to be applied to more general, continuous input domains. In both cases, they serve as a useful inductive bias that can improve performance.

3.1. Translation Equivariance with Pseudo Tokens

As with PT-TNPs, we can introduce pseudo-tokens $\mathbf{U} \in \mathbb{R}^{M \times D_z}$ into the TE-TNP to reduce the computational complexity from $\mathcal{O}(N_c^2 + N_c N_T)$ to $\mathcal{O}(M N_c + M N_t)$ (assuming $M \ll N_c, N_t$). To do so, we must also introduce corresponding pseudo-locations $\mathbf{V} \in \mathbb{R}^{M \times D_z}$. At each layer, we perform cross-attention between the pseudo-tokens and context-set tokens as in Equation 7. For the overall architecture to be translation equivariant, we require the initial pseudo-locations to be translation equivariant with respect to the inputs. As before, this is satisfied by functions of the form shown in Equation 9. We choose to implement this as

$$\mathbf{v}_m = \mathbf{v}_m^0 + \sum_{n=1}^{N} \psi\left(\mathbf{u}_m, \mathbf{z}_n\right) \mathbf{x}_n \tag{11}$$

where $\sum_{n=1}^{N} \psi\left(\mathbf{u}_m, \mathbf{z}_n\right) = 1 \; \forall m$. We exclude dependency on the pairwise differences between the observed input locations as this introduces an $\mathcal{O}\left(N^2\right)$ complexity. Equation 11 can be thought of as constructing a weighted average of the inputs, where the weights depend on the initial token values, which in turn depend on observations. We implement ψ using the attention mechanism in Equation 3. It should be noted that there exists scenarios in which this initialisation results in undesirable behaviour. For example, suppose that $\mathbf{v}_m^0 = \mathbf{0}$, $\phi = 1/N$. If you observe 100 data points in $[-101, -100]$ and 100 data points in $[100, 101]$, then the pseudo-points will be near the origin, far from the data. In general, we suspect that for any ψ, there exists a pathological data set that breaks this initialisation. Nonetheless, we found this initialisation scheme to be effective in practice and did not observe pathological behaviour such as this.

In Section F.5, we perform an ablation study to determine the effectiveness of dynamically updating input-locations using Equation 11 and Equation 10. We find that for simple input distributions (e.g. uniformly distributed), dynamically updating input locations has little effect. However, for more complex input distributions (e.g. bimodal), dynamically updating input locations significantly improves performance.

4. Experiments

In this section, we evaluate the performance of both translation equivariant and non-translation-equivariant NP models on modelling both synthetic and real-world data. We provide more detailed descriptions of the architectures and datasets used in Section F. In preliminary experiments, we found that PT-TNPs using the IST-style architecture outperformed PT-TNPs using the perceiver-style architecture. We therefore only include results using the former.

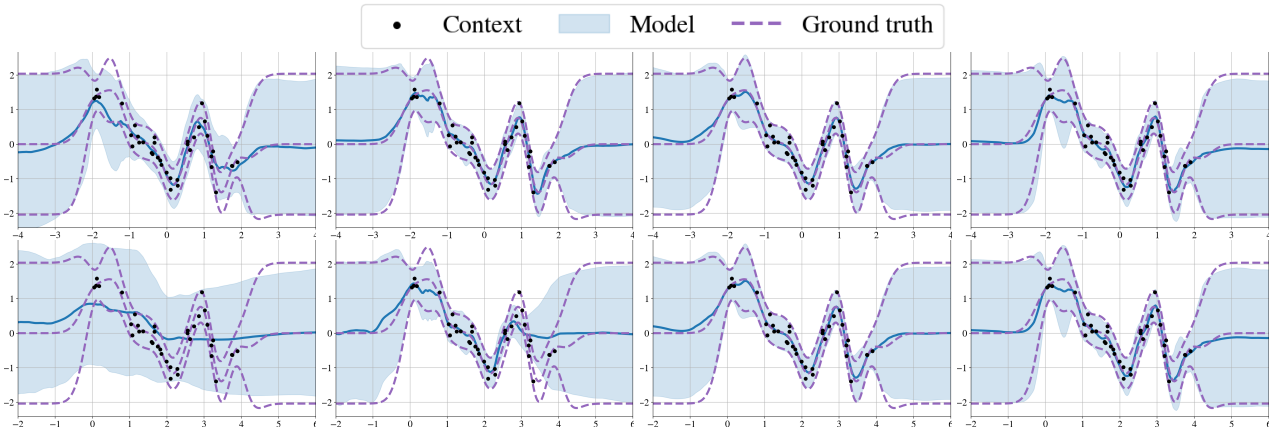
4.1. Synthetic 1-D Regression

We construct a meta-dataset by sampling from Gaussian processes (GPs) with periodic, squared-exponential and Matern 5/2 kernels, all of which define stationary GPs, with randomly sampled kernel hyperparameters. The number of context and target points are sampled according to $N_c \sim U(1, 64)$ and $N_t = 128$, and the context and target inputs are sampled according to $x_c \sim U(-2, 2)$ and $x_t \sim U(-3, 3)$. The range from which the context and target inputs are sampled from in the test set is shifted by amount Δ. As Δ increases, so too does the importance of translation equivariance. We compare the performance of TE-TNP and TE-PT-TNP with their non-translation-equivariant counterparts, a ConvCNP, and a simple RCNP. See Section F.1 for a detailed description.

Figure 2 plots the mean test log-likelihood across the test datasets as Δ increases from 0 to 1. For $\Delta = 0$, the input ranges for the test set and target set are equal and translation equivariance is not required to generalise. Nonetheless, we observe that both the TE-TNP and TE-PT-TNP-M32 outperform the TNP and PT-TNP-M32 models, recovering the performance of the ConvCNP model. As Δ increases, the performance of the non-translation-equivariant models deteriorate as they are unable to generalise to inputs sampled outside the training range. In Figure 4, we provide an example of this deterioration for the CNP and TNP, comparing their predictive posterior distributions to that of the ConvCNP and TE-TNP for a single test dataset.

4.2. Image Completion

We evaluate the TE-PT-TNP on image completion tasks, which can be interpreted as spatial regression of pixel values $\mathbf{y}_n \in \mathbb{R}^3$ given a 2-D pixel location $\mathbf{x}_n \in \mathbb{R}^2$. We consider experiments on MNIST (LeCun et al., 1998) and CIFAR10. For both datasets, we randomly translate the images by a maximum of $W/2$ horizontally and $H/2$ vertically, where W and H denote the width and height of the original image. The number of context points are sampled according to $N_c \sim U\left(\frac{N}{100}, \frac{N}{2}\right)$ and the number of target points are $N_t = N$, where N denotes the total number of pixels in the image. Due to the large number of context



(a) CNP. **(b)** TNP. **(c)** ConvCNP. **(d)** TE-TNP.

Figure 4. A comparison between the predictive distributions on a single synthetic-1D regression dataset of the CNP, TNP, TE-TNP and ConvCNP when the data is shifted by amount $\Delta = 0$ (top) and $\Delta = 2$ (bottom). Observe that the TE-TNP and ConvCNP models exhibit translation equivariance, whereas the CNP and TNP models do not. Context points are shown in black, and the ground-truth predictive mean and $\pm$ standard deviation are shown in dashed-purple.

Table 1. Average log-likelihood (↑) on the test datasets for the image completion experiments. Standard errors are shown.

Model	T-MNIST	T-CIFAR10
CNP	0.64 ± 0.00	0.72 ± 0.00
RCNP	1.04 ± 0.02	1.19 ± 0.02
ConvCNP	$\mathbf{1.20 \pm 0.02}$	1.44 ± 0.02
PT-TNP-M64	1.14 ± 0.02	1.43 ± 0.01
PT-TNP-M128	1.14 ± 0.02	1.46 ± 0.02
TE-PT-TNP-M64	1.18 ± 0.03	$\mathbf{1.50 \pm 0.02}$
TE-PT-TNP-M128	$\mathbf{1.25 \pm 0.03}$	$\mathbf{1.51 \pm 0.02}$

points, we do not evaluate the performance of the TE-TNP. See Section F.2 for full experimental details. We provide results for the average test log-likelihood for each model in Table 1. Unlike the previous task, here it is possible to learn the required translation equivariance from data. The results of this task demonstrate the utility of equipping our models with translation equivariance when appropriate, even when it is not explicitly required to perform well at test time. A visual comparison between the predictive mean of the ConvCNP and TE-PT-TNP is shown in Figure 8. Despite this task being well suited for the ConvCNP, as the pixels fall on a grid, the TE-PT-TNPs perform competitively.

4.3. Kolmogorov Flow

We consider a dataset generated by the 2-D Kolmogorov flow PDE used in (Lippe et al., 2023; Kochkov et al., 2021; Sun et al., 2023) (see Section F.3). The overall dataset consists of 921 simulations—we keep 819 for training and 102 for testing. Each simulation consists of a $64 \times 64 \times 64$ grid of 2-D observations. We sampled individual tasks by first sampling a $16 \times 16 \times 16$ region of a simulation. We then sample the number of context points $N_c \sim U(1, 500)$, with N_t set to all remaining points. As the inputs are 3-D, it is difficult to evaluate the performance of the ConvCNP due to the computational inefficiency of 3-D convolutions. Similarly, the large number of context points restricts our attention to PT-TNPs. We compare the performance of the PT-TNP and the TE-PT-TNP to the RCNP, CNP and a multi-task GP baseline with an SE kernel.[9] Table 2 compares the average test log-likelihoods. The TE-PT-TNP significantly outperforms all other models. In Figure 5, we visualise the vorticities computed using the predicted velocities of the TE-PT-TNP and multi-task GP models.

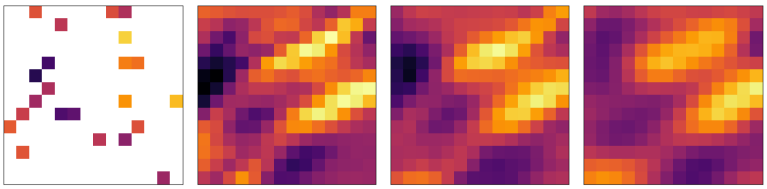
4.4. Environmental Data

We consider a real-world environmental dataset, derived from ERA5 (Copernicus Climate Change Service, 2020), consisting of surface air temperatures across space and time. Measurements are collected at a latitudinal and longitudinal resolution of $0.5°$, and temporal resolution of an hour ($\mathbf{x}_n \in \mathbb{R}^3$). We also provide the surface elevation at each coordinate as inputs. We consider measurements collected in 2019. Models are trained on measurements within the latitude / longitude range of $[42°,\ 53°]$ / $[8°,\ 28°]$ (roughly corresponding to central Europe), and evaluated on three non-overlapping regions: the training region, western Europe ($[42°,\ 53°]$ / $[-4°,\ 8°]$), and northern Europe ($[53°,\ 62°]$ / $[8°,\ 28°]$). During training, we sample datasets spanning 30 hours, sub-sampled to one every six hours, and $7.5°$ across each axis. Each dataset consists of a maximum of $N = 1125$ datapoints, from which the number of context

[9] The multi-task GP baseline is susceptible to overfitting when N_c is small. We therefore remove extreme values from the reported test log-likelihoods.

Table 2. Average test log-likelihood (↑) for Kolmogorov flow. Standard errors are shown.

Model	LL
Multi-task GP	0.48 ± 0.02
CNP	-0.97 ± 0.01
RCNP	0.16 ± 0.02
PT-TNP-M64	0.63 ± 0.003
PT-TNP-M128	0.74 ± 0.03
TE-PT-TNP-M64	$\mathbf{0.92 \pm 0.03}$
TE-PT-TNP-M128	$\mathbf{0.90 \pm 0.03}$



(e) Context. **(f)** Ground truth. **(g)** TE-PT-TNP. **(h)** Multi-task GP.

Figure 5. A comparison between the vorticity at a single point in time, computed using the predicted velocities for a single test Kolmogorov flow dataset. Here, the proportion of datapoints in the context set is 10%.

Table 3. Average log-likelihood (↑) for tasks sampled within the train range (C. Europe) and test ranges (W. Europe and N. Europe).

Model	C. Europe	W. Europe	N. Europe
GP	0.90 ± 0.01	1.22 ± 0.01	1.02 ± 0.01
CNP	0.36 ± 0.00	-5.70 ± 0.05	-1.64 ± 0.02
RCNP	0.96 ± 0.01	-0.09 ± 0.04	0.34 ± 0.01
PT-TNP	0.73 ± 0.01	-0.40 ± 0.01	-0.19 ± 0.00
TE-PT-TNP	$\mathbf{1.35 \pm 0.01}$	$\mathbf{1.27 \pm 0.01}$	$\mathbf{1.44 \pm 0.01}$

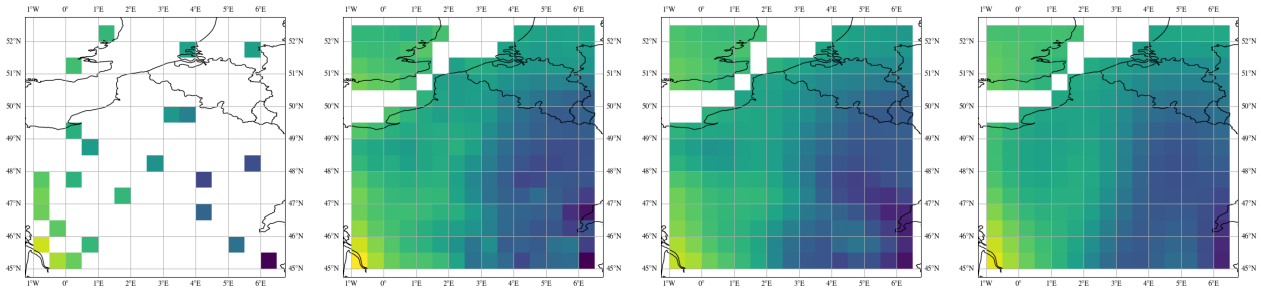
points are sampled according to $N_c \sim U(\frac{N}{100}, \frac{N}{3})$. As with the image completion experiments, the large number of datapoints present in each dataset render the TNP and TE-TNP too computationally intensive to train and evaluate. Similarly, because the inputs are 4-D, the ConvCNP cannot be implement due to insufficient support for 4-D convolutions. We therefore restrict our attention to the PT-TNP and TE-PT-TNP, both with $M = 128$ pseudo-tokens. We also evaluate the predictive performance of the CNP, RCNP and GPs with SE kernels. See Section F.4 for full experimental details. Table 3 compares the average test log-likelihood for each method on the different regions. The TE-PT-TNP model outperforms all baselines in all three regions. As noted by Foong et al. (2020), this is somewhat unsurprising given that: (1) the GP is prone to overfitting on small context sizes; and (2) the degree to which points influence each other is unlikely to be well explained by a SE kernel.

5. Related Work

Neural processes The combination of transformers and NPs was first considered by Kim et al. (2019) who developed the ANP. The ANP is characterised by multiple MHSA layers operating on the context tokens, followed by a single multi-head attention (MHA) layer in which the queries are the target locations, the keys are the context locations, and the values are the context tokens. The TNP-D architecture introduced by Nguyen & Grover (2022) built upon the ANP with multiple MHSA layers operating on the concatenation of context and target tokens. This repeated transfer of information from the context tokens to the target tokens at each layer improves performance, but is computationally inefficient in comparison to the efficient-query TNP (EQTNP) (Feng et al., 2022) which replaces masked MHSA layers with MHSA and MHCA. Finally, the LBANP (Feng et al., 2022) introduce the use of pseudo-tokens in a perceiver-style architecture to reduce the computational complexity further. To the best of our knowledge, there do not exist any TNP models which incorporate translation equivariance. However, there have been other NP variants which seek to achieve this. The RCNP (Huang et al., 2023) is similar to a single layer of the TE-TNP. However, as their choice of permutation aggregation function is linear (summation), they are not equivalent. ConvCNPs (Gordon et al., 2019), and more generally steerable CNPs (Holderrieth et al., 2021), incorporate translation equivariance by obtaining context representations in function space, discretising, and then performing translation equivariant operations using a CNN. A key computational advantage of the ConvCNP over the TE-TNP is the use of a CNN, which has computational complexity linear in the number of input points. However, the necessity of discretisation and convolutions restricts the ConvCNP to low-dimensional input domains. Further, the ConvCNP requires practioners to be much more judicious in their choice of model architecture, as careful consideration of discretisation the implied receptive field are required. We provide a summary of some important difference between the NP models discussed here in Table 4.

Table 4. A comparison between NP models. Complexity refers to computational complexity. FE denotes functional embedding: whether or not $e(\mathcal{D}_c, \mathbf{x}_t)$ depends on the input location $\mathbf{x}_t$. D_x-S denotes D_x scalability, where D_x is the input dimension. TE denotes translation equivariance.

Model	Complexity	D_x-S	FE	TE
CNP	$\mathcal{O}(N_c + N_t)$	✓	✗	✗
RCNP	$\mathcal{O}(N_c N_t)$	✓	✓	✓
ConvCNP	$\mathcal{O}(N_c D_x^3 + N_t D_x)$	✗	✗	✓
TNP	$\mathcal{O}(N_c^2 + N_c N_t)$	✓	✓	✗
PT-TNP	$\mathcal{O}(M N_c + M N_t)$	✓	✓	✗
TE-TNP	$\mathcal{O}(N_c^2 + N_c N_t)$	✓	✓	✓
TE-PT-TNP	$\mathcal{O}(M N_c + M N_t)$	✓	✓	✓



(a) Context dataset. **(b)** Ground truth data. **(c)** TE-PT-TNP. **(d)** GP.

Figure 6. A comparison between the predictive means of the TE-PT-TNP model and a GP for a single test dataset sampled from western Europe. The figures visualise a single slice through time of the (predictive) temperature, with the same colour scale being used throughout.

Equivariant GNNs and transformers When graph neural networks (GNNs) are applied to nodes in the euclidean domain, it is often beneficial to incorporate certain forms of euclidean equivariance. Doing so has been a topic of significant interest (Bronstein et al., 2021). Satorras et al. (2021) build E(n)-equivariant GNNs in the form of equivariant message passing, and bears close similarity to the approach taken in this paper. However, they tackle a different form of equivariance and their method does not closely resemble the attention-mechanism of transformers. Fuchs et al. (2020) build SE(3)-equivariant transformers, which our method shares similarities with. However, we avoid the complexities of irreducible representations by considering only translation equivariance. We choose a simpler, less memory intensive approach for incorporating translation equivariance that is effective in practice. The LieTransformer (Hutchinson et al., 2021) is a generalisation of translation equivariant transformers to Lie groups. Indeed, for certain choices of content-based and location-based attention mechanisms, our method can be recovered from the LieSelfAttention operation (see Appendix A). Nonetheless, Hutchinson et al. focus on SE-equivariance in their experiments, and do not consider integration into TNPs.

Data augmentation An alternative strategy to directly incorporating inductive biases, such as translation equivariance, into models is to use data augmentation during training. However, several works have shown that this approach has worse sample complexity and generalisation guarantees (Mei et al., 2021; Wang et al., 2022; Holderrieth et al., 2021). We found this to be empirically true in preliminary experiments, hence exclude comparisons in our experiments.

6. Conclusion

We have introduced the TE-TNP and TE-PT-TNP, expanding the family of TNPs to include their translation equivariant equivalents through the development of TE-MHSA and TE-MHCA operations. An extensive range of empirical results demonstrate that the TE-TNP and TE-PT-TNP perform on par or better than state-of-the-art NPs, such as the ConvCNP, whilst being versatile in their applicability. These models are not without their drawbacks, the two most significant being: (1) although not affecting the asymptotic behaviour with respect to the number of datapoints, in practice the need to pass pairwise computations through MLPs scales the computational complexity by a large factor; and (2) the need for the number of pseudo-tokens in PT-TNPs to scale with the number of datapoints. We seek to address both of these in future work.

Acknowledgements

CD is supported by the Cambridge Trust Scholarship. SM is supported by the Vice Chancellor's and Marie and George Vergottis Scholarship, and the Qualcomm Innovation Fellowship. JM acknowledges funding from the Data Sciences Institute at the University of Toronto and the Vector Institute. RET is supported by gifts from Google, Amazon, ARM, Improbable and EPSRC grant EP/T005386/1. We thank Emile Mathieu for their valuable feedback.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

References

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Betker, J., Goh, G., Jing, L., Brooks, T., Wang, J., Li, L.,

Ouyang, L., Zhuang, J., Lee, J., Guo, Y., et al. Improving image generation with better captions. *Computer Science. https://cdn. openai. com/papers/dall-e-3. pdf*, 2(3), 2023.

Bronstein, M. M., Bruna, J., Cohen, T., and Veličković, P. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.

Bruinsma, W. P., Requeima, J., Foong, A. Y. K., Gordon, J., and Turner, R. E. The Gaussian neural process. In *Proceedings of the 3rd Symposium on Advances in Approximate Bayesian Inference*, 2021.

Copernicus Climate Change Service. Near surface meteorological variables from 1979 to 2018 derived from bias-corrected reanalysis, 2020.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Feng, L., Hajimirsadeghi, H., Bengio, Y., and Ahmed, M. O. Latent bottlenecked attentive neural processes. *arXiv preprint arXiv:2211.08458*, 2022.

Foong, A., Bruinsma, W., Gordon, J., Dubois, Y., Requeima, J., and Turner, R. Meta-learning stationary stochastic process prediction with convolutional neural processes. *Advances in Neural Information Processing Systems*, 33: 8284–8295, 2020.

Fuchs, F., Worrall, D., Fischer, V., and Welling, M. Se (3)-transformers: 3d roto-translation equivariant attention networks. *Advances in neural information processing systems*, 33:1970–1981, 2020.

Gardner, J., Pleiss, G., Weinberger, K. Q., Bindel, D., and Wilson, A. G. Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. *Advances in neural information processing systems*, 31, 2018.

Garnelo, M. and Czarnecki, W. M. Exploring the space of key-value-query models with intention. *arXiv preprint arXiv:2305.10203*, 2023.

Garnelo, M., Rosenbaum, D., Maddison, C., Ramalho, T., Saxton, D., Shanahan, M., Teh, Y. W., Rezende, D., and Eslami, S. A. Conditional neural processes. In *International conference on machine learning*, pp. 1704–1713. PMLR, 2018a.

Garnelo, M., Schwarz, J., Rosenbaum, D., Viola, F., Rezende, D. J., Eslami, S., and Teh, Y. W. Neural processes. *arXiv preprint arXiv:1807.01622*, 2018b.

Gordon, J., Bruinsma, W. P., Foong, A. Y., Requeima, J., Dubois, Y., and Turner, R. E. Convolutional conditional neural processes. *arXiv preprint arXiv:1910.13556*, 2019.

Holderrieth, P., Hutchinson, M. J., and Teh, Y. W. Equivariant learning of stochastic fields: Gaussian processes and steerable conditional neural processes. In *International Conference on Machine Learning*, pp. 4297–4307. PMLR, 2021.

Huang, D., Haussmann, M., Remes, U., John, S., Clarté, G., Luck, K. S., Kaski, S., and Acerbi, L. Practical equivariances via relational conditional neural processes. *arXiv preprint arXiv:2306.10915*, 2023.

Hutchinson, M. J., Le Lan, C., Zaidi, S., Dupont, E., Teh, Y. W., and Kim, H. Lietransformer: Equivariant self-attention for lie groups. In *International Conference on Machine Learning*, pp. 4533–4543. PMLR, 2021.

Jaegle, A., Gimeno, F., Brock, A., Vinyals, O., Zisserman, A., and Carreira, J. Perceiver: General perception with iterative attention. In *International conference on machine learning*, pp. 4651–4664. PMLR, 2021.

Jha, S., Gong, D., Wang, X., Turner, R. E., and Yao, L. The neural process family: Survey, applications and perspectives. *arXiv preprint arXiv:2209.00517*, 2022.

Kim, H., Mnih, A., Schwarz, J., Garnelo, M., Eslami, A., Rosenbaum, D., Vinyals, O., and Teh, Y. W. Attentive neural processes. *arXiv preprint arXiv:1901.05761*, 2019.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kochkov, D., Smith, J. A., Alieva, A., Wang, Q., Brenner, M. P., and Hoyer, S. Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118(21):e2101784118, 2021.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Lee, J., Lee, Y., Kim, J., Kosiorek, A., Choi, S., and Teh, Y. W. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International conference on machine learning*, pp. 3744–3753. PMLR, 2019.

Lippe, P., Veeling, B. S., Perdikaris, P., Turner, R. E., and Brandstetter, J. Pde-refiner: Achieving accurate long rollouts with neural pde solvers. *arXiv preprint arXiv:2308.05732*, 2023.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Markou, S., Requeima, J., Bruinsma, W. P., Vaughan, A., and Turner, R. E. Practical conditional neural processes via tractable dependent predictions. In *Proceedings of the 10th International Conference on Learning Representations*, 2022.

Mei, S., Misiakiewicz, T., and Montanari, A. Learning with invariances in random features and kernel models. In Belkin, M. and Kpotufe, S. (eds.), *Proceedings of Thirty Fourth Conference on Learning Theory*, volume 134 of *Proceedings of Machine Learning Research*, pp. 3351–3418. PMLR, 15–19 Aug 2021. URL https://proceedings.mlr.press/v134/mei21a.html.

Müller, S., Hollmann, N., Arango, S. P., Grabocka, J., and Hutter, F. Transformers can do bayesian inference. *arXiv preprint arXiv:2112.10510*, 2021.

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

Ronneberger, O., Fischer, P., and Brox, T. 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*, pp. 234–241. Springer, 2015.

Rozet, F. and Louppe, G. Score-based data assimilation. *arXiv preprint arXiv:2306.10574*, 2023.

Satorras, V. G., Hoogeboom, E., and Welling, M. E (n) equivariant graph neural networks. In *International conference on machine learning*, pp. 9323–9332. PMLR, 2021.

Schilling, R. L. *Measures, Integrals and Martingales*. Cambridge University Press, 2005. doi: 10.1017/CBO9780511810886.

Su, J., Ahmed, M., Lu, Y., Pan, S., Bo, W., and Liu, Y. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Sun, Z., Yang, Y., and Yoo, S. A neural pde solver with temporal stencil modeling. *arXiv preprint arXiv:2302.08105*, 2023.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Wagstaff, E., Fuchs, F. B., Engelcke, M., Osborne, M. A., and Posner, I. Universal approximation of functions on sets. *The Journal of Machine Learning Research*, 23(1): 6762–6817, 2022.

Wang, R., Walters, R., and Yu, R. Data augmentation vs. equivariant networks: A theory of generalization on dynamics forecasting. *arXiv preprint arXiv:2206.09450*, 2022.

Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R. R., and Smola, A. J. Deep sets. *Advances in neural information processing systems*, 30, 2017.

A. Relationship to LieTransformer

At the core of the LieTransformer is the LieSelfAttention operation. Let group G denote the set of symmetries we wish to be equivariant to, $\mathbf{x}_n \in \mathbb{R}^{d_x}$ denote the spatial coordinate of a point and $\mathbf{z}_n \in \mathbb{R}^{d_z}$ the token corresponding to this spatial coordinate. Each $\mathbf{x}_n$ corresponds to the coset $s(\mathbf{x})H = \{s(\mathbf{x})h | h \in H\}$, where the subgroup $H = \{g \in G | g\mathbf{x}_0 = \mathbf{x}_0\}$ is called the stabiliser of origin $\mathbf{x}_0$ and $s(x) \in G$ is a group element that maps $\mathbf{x}_0$ to $\mathbf{x}$. In the case of the group of translations, $s(\mathbf{x})$ is just the translation that maps from the origin to $\mathbf{x}$. Thus, each spatial coordinate can be mapped to group elements in G. This can be thought of as lifting the feature map $f_{\mathcal{X}} : \mathbf{x}_n \to \mathbf{z}_n$ defined on $\mathcal{X}$ (i.e. $\mathbb{R}^{d_x}$) to a feature map $\mathcal{L}[f_{\mathcal{X}}] : g \to \mathbf{z}_n$ defined on G (also $\mathbb{R}^{d_x}$ in the case of translations).

Let $G_f = \cup_{n=1}^{N} s(\mathbf{x}_n)H$. The LieSelfAttention operations updates the feature values $\mathbf{z}_n$ at spatial coordinates $\mathbf{x}_n$ (corresponding to group element g_n) as

$$\tilde{\mathbf{z}}_n = \int \alpha_f(\mathbf{z}_n, \mathbf{z}_m, g_n, g_m)\mathbf{W}_V \mathbf{z}_m dg_m \tag{12}$$

with the attention-weights given by

$$\alpha_f(\mathbf{z}_n, \mathbf{z}_m, g_n, g_m) = \mathrm{softmax}\left(\phi\left(k_c\left(\mathbf{z}_n, \mathbf{z}_m\right), k_l\left(g_n^{-1} g_m\right)\right)\right). \tag{13}$$

In the case of the group of translations, this corresponds to

$$\tilde{\mathbf{z}}_n = \sum_{m=1}^{M} \alpha_f(\mathbf{z}_n, \mathbf{z}_m, g_n, g_m)\mathbf{W}_V \mathbf{z}_m. \tag{14}$$

Choosing $k_c(\mathbf{z}_n, \mathbf{z}_m) = \mathbf{z}_n^T \mathbf{W}_{Q,h} \mathbf{W}_{K,h}^T \mathbf{z}_m$ and $k_l(\mathbf{x}_n, \mathbf{x}_m) = \mathbf{x}_n - \mathbf{x}_m$ recovers the TE-MHSA layer with $H = 1$.

B. Efficient Masked Attention

Often, conditional independencies amongst the set of tokens—in the sense that the set $\{\mathbf{z}_n^\ell\}_{\ell=1}^{\ell=L}$ do not depend on the set $\{\mathbf{z}_m^\ell\}_{\ell=0}^{\ell=L}$ for some $n,\ m \in \{1, \dots, N\}$—are desirable. This is typically achieved through masking, such that the pre-softmax activations are replaced by $\tilde{\alpha}_h^\ell$, where

$$\tilde{\alpha}_h^\ell(\mathbf{z}_n^\ell, \mathbf{z}_m^\ell) = \begin{cases} -\infty, & m \in A(n). \\ {\mathbf{z}_n^\ell}^T \mathbf{W}_{Q,h}^\ell \left[\mathbf{W}_{K,h}^\ell\right]^T \mathbf{z}_m^\ell, & \text{otherwise.} \end{cases} \tag{15}$$

Here, $A(n) \subseteq \mathbb{N}_{\leq N}$ indexes the set of tokens we wish to make the update for token $\mathbf{z}_n^\ell$ independent of. If $A(n) = A$ (i.e. the same set of tokens are conditioned on for every n) then in practice it is more computationally efficient to use MHCA operations together with MHSA operations than it is to directly compute Equation 2. An MHCA operation uses the subset of tokens $\{\mathbf{z}_m^\ell | m \in A\}$ to update the complementary set of tokens $\{\mathbf{z}_n^\ell | n \in A^c\}$ in a computationally efficient manner. For N tokens that solely depend on a subset of N_1 tokens, the computational complexity is reduced from $\mathcal{O}\left(N^2\right)$ using masked-MHSA operations to $\mathcal{O}\left(NN_1\right)$ using MHSA and MHCA operations.

In the context of TNP, the tokens in both the context and target set are conditioned only on tokens in the context set. Thus, replacing masked-MHSA operations reduces the computational complexity from $\mathcal{O}\left((N_c + N_t)^2\right)$ to $\mathcal{O}\left(N_c^2 + N_c N_t\right)$. If the tokens in the target set are conditioned on context set tokens *and* themselves, then we can easily modify the standard MHCA operation to include the individual target tokens.

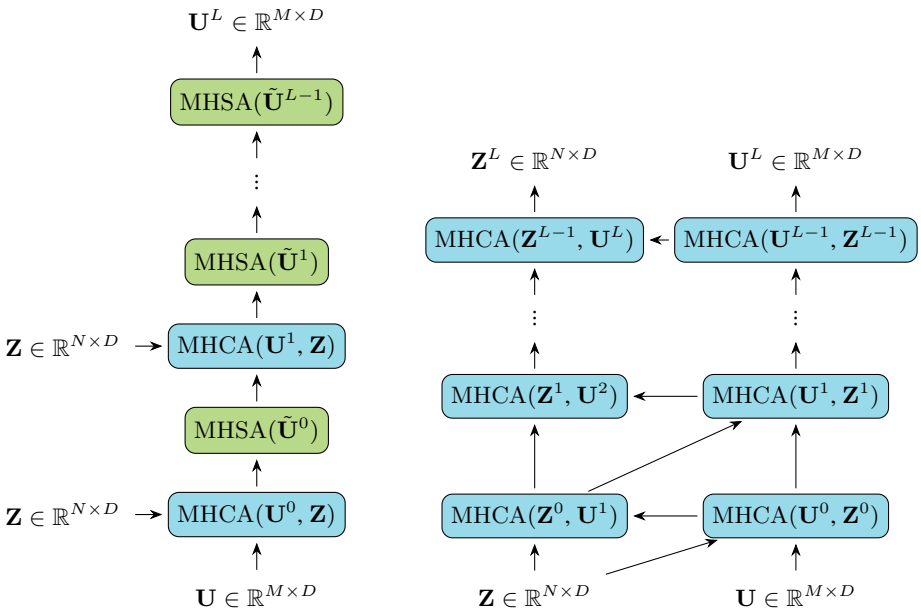
C. Pseudo-Token-Based Transformers

We illustrate the two types of pseudo-token-based transformers, the IST-style and perceiver-style, in Figure 7a and Figure 7b.

D. General Form for Translation Equivariant and Permutation Invariant Functions

Let $f\colon (\mathbb{R}^D)^N \to \mathbb{R}^D$ be a continuous function which is (1) permutation equivariant and (2) translation equivariant. That f is permutation equivariant means that, for all permutations $\sigma \in \mathbb{S}^N$ of N elements,

$$f(\mathbf{x}_1, \dots, \mathbf{x}_N) = f(\mathbf{x}_{\sigma(1)}, \dots, \mathbf{x}_{\sigma(N)}); \tag{16}$$



(a) Perceiver-style architecture. **(b)** IST-style architecture.

Figure 7. Block diagrams the two pseudo-token-based transformer architectures.

and that f is translation equivariant means that, for all translations $\boldsymbol{\tau} \in \mathbb{R}^D$,

$$f(\mathbf{x}_1 + \boldsymbol{\tau}, \ldots, \mathbf{x}_N + \boldsymbol{\tau}) = f(\mathbf{x}_1, \ldots, \mathbf{x}_N) + \boldsymbol{\tau}. \tag{17}$$

Let $\alpha_1, \ldots, \alpha_N \in \mathbb{R}$ be *any* set of weights, possibly negative, such that $\sum_{i=1}^N \alpha_i = 1$. Then, using translation equivariance of f,

$$f(\mathbf{x}_1, \ldots, \mathbf{x}_N) = \sum_{i=1}^N \alpha_i f(\mathbf{x}_1, \ldots, \mathbf{x}_N) = \sum_{i=1}^N \alpha_i f(\mathbf{x}_1 - \mathbf{x}_i, \ldots, \mathbf{x}_N - \mathbf{x}_i) + \sum_{i=1}^N \alpha_i \mathbf{x}_i. \tag{18}$$

Since f is also permutation equivariant, it can be written in the following way (Zaheer et al., 2017):

$$f(\mathbf{x}_1, \ldots, \mathbf{x}_N) = \rho\Big(\sum_{j=1}^N \phi(\mathbf{x}_j)\Big) \tag{19}$$

for some continuous functions ρ and ϕ. Therefore

$$f(\mathbf{x}_1, \ldots, \mathbf{x}_N) = \sum_{i=1}^N \alpha_i \rho\Big(\sum_{j=1}^M \phi(\mathbf{x}_j - \mathbf{x}_i)\Big) + \sum_{i=1}^N \alpha_i \mathbf{x}_i. \tag{20}$$

Note that this decomposes f into a *translation invariant* component and *translation equivariant* component. We emphasise that this holds for *any* set of weights $\alpha_1, \ldots, \alpha_N$. In particular, these weights may depend on $\mathbf{x}_1, \ldots, \mathbf{x}_N$. Finally, we can rewrite (20) into residual form. Let $K \in \mathbb{N}$, $1 \leq K \leq N$. Then

$$f(\mathbf{x}_1, \ldots, \mathbf{x}_N) = \mathbf{x}_K + \sum_{i=1}^N \alpha_i \rho\Big(\sum_{j=1}^M \phi(\mathbf{x}_j - \mathbf{x}_i)\Big) + \sum_{i=1}^N \alpha_i (\mathbf{x}_i - \mathbf{x}_K). \tag{21}$$

E. Proofs for Subsection 2.5

In the following, we use the notation and definitions from Section 2.5. Recall that the collection of all data sets $\mathcal{S}$ includes the empty set $\varnothing$, which is the data set containing no data points.

Proof of Theorem 2.1. Suppose that the ground-truth stochastic process $f \sim P$ is stationary: $f \stackrel{\mathrm{d}}{=} \mathsf{T}_{\boldsymbol{\tau}} f$ for all $\boldsymbol{\tau} \in \mathcal{X}$. Also suppose that π'_P is translation invariant. Let $\mathcal{D} \in \mathcal{S}$ and $\boldsymbol{\tau} \in \mathcal{X}$. Let B be a cylinder set. Then, by the changes-of-variables formula for pushforward measures (Theorem 14.1; Schilling, 2005),

$$\int_B \pi'_P(\mathsf{T}_{\boldsymbol{\tau}}\mathcal{D}) \,\mathrm{d}P = \int_B \pi'_P(\mathcal{D}) \circ \mathsf{T}_{\boldsymbol{\tau}}^{-1} \,\mathrm{d}P \qquad (\pi'_P \text{ is translation invariant}) \tag{22}$$

$$= \int_{\mathsf{T}_{\boldsymbol{\tau}}^{-1}(B)} \pi'_P(\mathcal{D}) \,\mathrm{d}\mathsf{T}_{\boldsymbol{\tau}}^{-1}(P) \qquad (\text{change of variables}) \tag{23}$$

$$= \int_{\mathsf{T}_{\boldsymbol{\tau}}^{-1}(B)} \pi'_P(\mathcal{D}) \,\mathrm{d}P. \qquad (f \text{ is stationary}) \tag{24}$$

We conclude that π_P is translation equivariant.

Conversely, if π_P is translation equivariant, then, considering the data set containing no data points $\varnothing$,

$$\mathsf{T}_{\boldsymbol{\tau}} \pi_P(\varnothing) = \pi_P(\mathsf{T}_{\boldsymbol{\tau}} \varnothing) = \pi_P(\varnothing) \quad \text{for all } \boldsymbol{\tau} \in \mathcal{X}. \tag{25}$$

Since $\pi_P(\varnothing) = P$, this means that f is stationary. Moreover, let B be a cylinder set. Then

$$\int_B \pi'_P(\mathsf{T}_{\boldsymbol{\tau}}\mathcal{D}) \,\mathrm{d}P = \int_{\mathsf{T}_{\boldsymbol{\tau}}^{-1}(B)} \pi'_P(\mathcal{D}) \,\mathrm{d}P \qquad (\pi_P \text{ is translation equivariant}) \tag{26}$$

$$= \int_{\mathsf{T}_{\boldsymbol{\tau}}^{-1}(B)} \pi'_P(\mathcal{D}) \,\mathrm{d}\mathsf{T}_{\boldsymbol{\tau}}^{-1}(P) \qquad (f \text{ is stationary}) \tag{27}$$

$$= \int_B \pi'_P(\mathcal{D}) \circ \mathsf{T}_{-\boldsymbol{\tau}} \,\mathrm{d}P. \qquad (\text{change of variables}) \tag{28}$$

Since this holds for all cylinder sets, $\pi'_P(\mathsf{T}_{\boldsymbol{\tau}}\mathcal{D}) = \pi'_P(\mathcal{D}) \circ \mathsf{T}_{\boldsymbol{\tau}}$ P–almost surely, so π'_P is translation invariant. □

The proof of Theorem 2.2 follows the idea illustrated in Figure 3. Let $\mathbf{x}_1 \oplus \mathbf{x}_2$ denote the concatenation of two vectors $\mathbf{x}_1$ and $\mathbf{x}_2$.

Proof of Theorem 2.2. Let $M > 0$, $n \in \{1, \ldots, N\}$, $\mathbf{x} \in [0, M]^n$, and $\mathcal{D} \in \mathcal{S} \cap \bigcup_{n=0}^{\infty}([0, M] \times \mathbb{R})^n$. Sort and put the n inputs $\mathbf{x}$ into $B = \lceil M/\frac{1}{2}R \rceil$ buckets $(B_i)_{i=1}^B$ such that $x_j \in [(i-1) \cdot \frac{1}{2}R, i \cdot \frac{1}{2}R]$ for all $j \in B_i$. More concisely written, $\mathbf{x}_{B_i} \in [(i-1) \cdot \frac{1}{2}R, i \cdot \frac{1}{2}R]^{|B_i|}$. Write $C_i = \bigcup_{j=1}^{i-1} B_i$. Let $\mathcal{D}_i$ be the sub–data set of $\mathcal{D}$ with inputs in $[\min(\mathbf{x}_{B_{i-2}}), \max(\mathbf{x}_{B_{i+1}})]$.

If $\mathbf{y}_1 \oplus \mathbf{y}_2 \sim P_{\mathbf{x}_1 \oplus \mathbf{x}_2} \pi(\mathcal{D})$, then denote the distribution of $\mathbf{y}_1 \mid \mathbf{y}_2$ by $P_{\mathbf{x}_1 | \mathbf{x}_2} \pi(\mathcal{D})$. Use the chain rule for the KL divergence to decompose

$$\mathrm{KL}[P_{\mathbf{x}} \pi_1(\mathcal{D}) \| P_{\mathbf{x}} \pi_2(\mathcal{D})] = \sum_{i=1}^{B} \mathbb{E}_{P_{\mathbf{x}_{C_i}} \pi_1(\mathcal{D})}[\mathrm{KL}[P_{\mathbf{x}_{B_i} | \mathbf{x}_{C_i}} \pi_1(\mathcal{D}) \| P_{\mathbf{x}_{B_i} | \mathbf{x}_{C_i}} \pi_2(\mathcal{D})]]. \tag{29}$$

We focus on the ith term in the sum. Using that $\pi_1(\mathcal{D})$ and $\pi_2(\mathcal{D})$ have receptive field R, we may drop the dependency on $B_1, \ldots, B_{i-2}$:

$$\mathrm{KL}[P_{\mathbf{x}_{B_i} | \mathbf{x}_{C_i}} \pi_1(\mathcal{D}) \| P_{\mathbf{x}_{B_i} | \mathbf{x}_{C_i}} \pi_2(\mathcal{D})] = \mathrm{KL}[P_{\mathbf{x}_{B_i} | \mathbf{x}_{B_{i-1}}} \pi_1(\mathcal{D}) \| P_{\mathbf{x}_{B_i} | \mathbf{x}_{B_{i-1}}} \pi_2(\mathcal{D})]. \tag{30}$$

Therefore,

$$\mathbb{E}_{P_{\mathbf{x}_{C_i}} \pi_1(\mathcal{D})}[\mathrm{KL}[P_{\mathbf{x}_{B_i} | \mathbf{x}_{C_i}} \pi_1(\mathcal{D}) \| P_{\mathbf{x}_{B_i} | \mathbf{x}_{C_i}} \pi_2(\mathcal{D})]] = \mathbb{E}_{P_{\mathbf{x}_{B_{i-1}}} \pi_1(\mathcal{D})}[\mathrm{KL}[P_{\mathbf{x}_{B_i} | \mathbf{x}_{B_{i-1}}} \pi_1(\mathcal{D}) \| P_{\mathbf{x}_{B_i} | \mathbf{x}_{B_{i-1}}} \pi_2(\mathcal{D})]] \tag{31}$$

$$\leq \mathrm{KL}[P_{\mathbf{x}_{B_i \cup B_{i-1}}} \pi_1(\mathcal{D}) \| P_{\mathbf{x}_{B_i \cup B_{i-1}}} \pi_2(\mathcal{D})]. \tag{32}$$

Next, we use that π_1 and π_2 also have receptive field R, allowing us to replace $\mathcal{D}$ with $\mathcal{D}_i$:

$$\mathrm{KL}[P_{\mathbf{x}_{B_i \cup B_{i-1}}} \pi_1(\mathcal{D}) \| P_{\mathbf{x}_{B_i \cup B_{i-1}}} \pi_2(\mathcal{D})] = \mathrm{KL}[P_{\mathbf{x}_{B_i \cup B_{i-1}}} \pi_1(\mathcal{D}_i) \| P_{\mathbf{x}_{B_i \cup B_{i-1}}} \pi_2(\mathcal{D}_i)]. \tag{33}$$

Finally, we use translation equivariance to shift everything back to the origin. Let τ_i be $\min(\mathbf{x}_{B_{i-2}})$. By translation equivariance of π_1,

$$P_{\mathbf{x}_{B_i \cup B_{i-1}}} \pi_1(\mathcal{D}_i) = P_{\mathbf{x}_{B_i \cup B_{i-1}}} \mathsf{T}_{\tau_i} \pi_1(\mathsf{T}_{-\tau_i} \mathcal{D}_i) = P_{\mathbf{x}_{B_i \cup B_{i-1}} - \tau_i} \pi_1(\mathsf{T}_{-\tau_i} \mathcal{D}_i) \tag{34}$$

where by $\mathbf{x}_{B_i \cup B_{i-1}} - \tau_i$ we mean subtraction elementwise. Crucially, note that all elements of $\mathbf{x}_{B_i \cup B_{i-1}} - \tau_i$ and all inputs of $\mathsf{T}_{-\tau_i} \mathcal{D}_i$ lie in $[0, 4 \cdot \frac{1}{2} R]$. We have a similar equality for π_2. Therefore, putting everything together,

$$\mathbb{E}_{P_{\mathbf{x}_{C_i}} \pi_1(\mathcal{D})} [\mathrm{KL}[P_{\mathbf{x}_{B_i} | \mathbf{x}_{C_i}} \pi_1(\mathcal{D}) \| P_{\mathbf{x}_{B_i} | \mathbf{x}_{C_i}} \pi_2(\mathcal{D})]] \leq \mathrm{KL}[P_{\mathbf{x}_{B_i \cup B_{i-1}} - \tau_i} \pi_1(\mathsf{T}_{-\tau_i} \mathcal{D}_i) \| P_{\mathbf{x}_{B_i \cup B_{i-1}} - \tau_i} \pi_2(\mathsf{T}_{-\tau_i} \mathcal{D}_i)], \tag{35}$$

which is less than ϵ by the assumption of the theorem. The conclusion now follows. □

F. Experimental Details and Additional Results

F.1. Synthetic 1-D Regression

For each dataset, we first sample a kernel $k \sim U([k_{\text{se}}, k_{\text{pe}}, k_{\text{ma}-2.5}])$, where

$$\begin{aligned} k_{\text{se}} &= \exp\left(-\frac{(x - x')^2}{2\ell^2}\right) \\ k_{\text{pe}} &= \exp\left(-2 \sin^2\left(\frac{\pi}{\ell}(x - x')\right)^2\right) \\ k_{\text{ma}-2.5} &= \frac{2^{-1.5}}{\Gamma(2.5)} \left(\frac{\sqrt{5}(x - x')}{\ell^2}\right)^{2.5} K_{2.5}\left(\frac{\sqrt{5}(x - x')}{\ell^2}\right). \end{aligned} \tag{36}$$

We sample $\ell \sim U(\log 0.25, \log 4)$, the number of context points $N_c \sim U(1, 64)$, the context inputs $x_{c,n} \sim U(-2, 2)$, and target inputs $x_{t,n} \sim U(-3, 3)$. All tasks use the same number of target points $N_t = 128$. The observations for each task are drawn from a GP with kernel

$$k_{\text{obs}} = k + \sigma_n^2 \delta(x - x') \tag{37}$$

where the observation noise $\sigma_n = 0.2$.

For all models, we use an embedding / token size of $D_z = 128$, and decoder consisting of an MLP with two hidden layers of dimension D_z. The decoder parameterises the mean and pre-softplus variance of a Gaussian likelihood with heterogeneous noise. Model specific architectures are as follows:

TNP The initial context tokens are obtained by passing the concatenation $[x, y, 1]$ through an MLP with two hidden layers of dimension D_z. The initial target tokens are obtained by passing the concatenation $[x, 0, 0]$ through the same MLP. The final dimension of the input acts as a 'density' channel to indicate whether or not an observation is present. The TNP encoder consists of five layers of self-attention and cross-attention blocks, each with $H = 8$ attention heads with dimensions $D_V = D_{QK} = 16$. In each of the attention blocks, we apply a residual connection consisting of layer-normalisation to the input tokens followed by the attention mechanism. Following this there is another residual connection consisting of a layer-normalisation followed by a pointwise MLP with two hidden layers of dimension D_z.

PT-TNP For the PT-TNP models we use the same architecture dimensions as the TNP. The initial pseudo-token values are sampled from a standard normal distribution.

RCNP We implement the simple RCNP from Huang et al. (2023), as the memory requirements of the full RCNP exceed the limits of our hardware for all but the simplest architectures. The simple RCNP encoder is implemented as

$$e(\mathbf{x}, \mathcal{D}_c) = \oplus_{n=1}^{N_c} \phi(\mathbf{x} - \mathbf{x}_{c,n}, \mathbf{y}_{c,n}) \tag{38}$$

where $\oplus$ denotes a permutation invariant aggregation, for which we use the mean operation.[10] We implement the relational encoder $\phi : \mathbb{R} \times \mathbb{R} \to \mathbb{R}^{D_z}$ as an MLP with five hidden layers of dimension D_z.

[10] We found that for large datasets, the summation operation resulted in inputs to the decoder that were very large, resulting in numerical instabilities.

CNP The CNP encoder is implemented as

$$e(\mathbf{x}, \mathcal{D}_c) = \oplus_{n=1}^{N_c} \phi\left(\mathbf{x}_{c,n}, \mathbf{y}_{c,n}\right) \tag{39}$$

where $\oplus$ denotes a permutation invariant aggregation, for which we use the mean operation. We implement $\phi : \mathbb{R} \times \mathbb{R} \to \mathbb{R}^{D_z}$ as an MLP with five hidden layers of dimension D_z.

ConvCNP For the ConvCNP model, we use a UNet (Ronneberger et al., 2015) architecture for the CNN with 11 layers. We use $C = 128$ input / output channels for the downwards layers, between which we apply pooling with size two. For the upwards layers, we use $2C$ input channels and C output channels, as the input channels are formed from the output of the previous layer concatenated with the output of the corresponding downwards layer. Between the upwards layers we apply linear up-sampling to match the dimensions of the downwards layer. We use a kernel size of nine with a stride of one. The input domain is discretised with 64 points per units.

TE-TNP The TE-TNP model share a similar architecture with the TNP model, with the attention blocks replaced with their translation equivariant counterparts. For the translation equivariant attention mechanisms, we implement $\rho^\ell : \mathbb{R}^H \times \mathbb{R} \to \mathbb{R}^H$ as an MLP with two hidden layers of dimension D_z. We implement $\phi^\ell : \mathbb{R}^H \to \mathbb{R}^H$ as an MLP with two hidden layers of dimension D_z. The initial context token embeddings are obtained by passing the context observations through an MLP with two hidden layers of dimension D_z. The initial target token embeddings are sampled from a standard normal.

TE-PT-TNP The TE-PT-TNP models adopt the same architecture choices as the TE-TNP. The initial pseudo-tokens and pseudo-input-locations are sampled from a standard normal.

Training Details For all models, we optimise the model parameters using AdamW (Loshchilov & Hutter, 2017) with a learning rate of 5×10^{-4} and batch size of 16. Gradient value magnitudes are clipped at 0.5. We train for a maximum of 500 epochs, with each epoch consisting of 16,000 datasets (10,000 iterations per epoch). We evaluate the performance of each model on test 80,000 datasets.

Table 5. Average log-likelihood (↑) on the test datasets for the synthetic 1-D regression experiment. Δ denotes the amount by which the range from which the context and target inputs and sampled from is shifted to the right at test time.

	Δ					
Model	0.0	0.2	0.4	0.6	0.8	1.0
TNP	-0.48 ± 0.00	-0.48 ± 0.01	-0.49 ± 0.01	-0.50 ± 0.01	-0.52 ± 0.01	-0.57 ± 0.01
PT-TNP-M8	-0.52 ± 0.00	-0.52 ± 0.01	-0.56 ± 0.01	-0.73 ± 0.01	-0.76 ± 0.01	-0.71 ± 0.01
PT-TNP-M16	-0.49 ± 0.00	-0.50 ± 0.01	-0.53 ± 0.01	-0.57 ± 0.01	-0.60 ± 0.01	-0.64 ± 0.01
PT-TNP-M32	-0.48 ± 0.00	-0.49 ± 0.01	-0.52 ± 0.01	-0.55 ± 0.01	-0.59 ± 0.01	-0.63 ± 0.01
CNP	-0.69 ± 0.01	-0.71 ± 0.01	-0.77 ± 0.005	-0.86 ± 0.01	-0.98 ± 0.00	-1.08 ± 0.00
RCNP	-0.58 ± 0.01	-0.58 ± 0.01	-0.58 ± 0.01	-0.58 ± 0.01	-0.58 ± 0.01	-0.58 ± 0.01
ConvCNP	$\mathbf{-0.46 \pm 0.01}$	$\mathbf{-0.46 \pm 0.01}$	$\mathbf{-0.46 \pm 0.01}$	$\mathbf{-0.46 \pm 0.01}$	$\mathbf{-0.46 \pm 0.01}$	$\mathbf{-0.46 \pm 0.01}$
TE-TNP	$\mathbf{-0.47 \pm 0.01}$	$\mathbf{-0.47 \pm 0.01}$	$\mathbf{-0.47 \pm 0.01}$	$\mathbf{-0.47 \pm 0.01}$	$\mathbf{-0.47 \pm 0.01}$	$\mathbf{-0.47 \pm 0.01}$
TE-PT-TNP-M8	-0.50 ± 0.00	-0.50 ± 0.00	-0.50 ± 0.00	-0.50 ± 0.00	-0.50 ± 0.00	-0.50 ± 0.00
TE-PT-TNP-M16	-0.50 ± 0.00	-0.50 ± 0.00	-0.50 ± 0.00	-0.50 ± 0.00	-0.50 ± 0.00	-0.50 ± 0.00
TE-PT-TNP-M32	$\mathbf{-0.46 \pm 0.00}$	$\mathbf{-0.46 \pm 0.00}$	$\mathbf{-0.46 \pm 0.00}$	$\mathbf{-0.46 \pm 0.00}$	$\mathbf{-0.46 \pm 0.00}$	$\mathbf{-0.46 \pm 0.00}$

F.2. Image Completion

The image completion dataset uses MNIST (LeCun et al., 1998) and CIFAR-10. Each MNIST dataset consists of a 28×28 black and white image, and each CIFAR-10 dataset consists of a 32×32 RGB image. To construct each randomly translated dataset from the original images, we sample a horizontal and vertical translation, $h, v \sim U([-14, 14])$ for MNIST and $h, v \sim U([-16, 16])$. The original image canvas is expanded to 56×56 for MNIST and 64×64 for CIFAR-10, with the new pixels given output values of zero, and the translated image is inserted. The training set consists of 60,000 images for T-MNIST and 50,000 images for T-CIFAR-10 (i.e. a single random translation applied to each image in the original training sets). The test set consists of 10,000 images for both T-MNIST and T-CIFAR-10. For each dataset, we sample the $N_c \sim U\left(\frac{N}{100}, \frac{N}{3}\right)$ and set N_t to the remaining pixels.

Table 6. Time taken to complete the first 200 training epochs.

Model	Training time (hours)
TNP	3.249
PT-TNP-M32	3.565
CNP	1.672
RCNP	2.326
ConvCNP	3.466
TE-TNP	7.767
TE-PT-TNP-M32	5.502

For all models, we use an embedding / token size of $D_z = 32$. This was chosen to be relatively small due to the limitations of the hardware available (both the TNP and RCNP models have a memory complexity that scales with $\mathcal{O}(N_c N_T)$). For both the T-MNIST and T-CIFAR-10, we use a Gaussian likelihood with homogeneous noise. Similar to Foong et al. (2020), which found that this significantly improve training stability. Model specific architectures are as follows:

PT-TNP Same as Section F.1.

RCNP Same as Section F.1.

CNP Same as Section F.1.

ConvCNP For the ConvCNP model, we use a standard CNN 5 layers. We use $C = 64$ input / output channels for each layer. As the input domain is already discretised, discretisation is not needed.

TE-PT-TNP Same as Section F.1.

Training Details For all models, we optimise the model parameters using AdamW (Loshchilov & Hutter, 2017) with a learning rate of 5×10^{-4} and batch size of 16 (8 for the TE-PT-TNP and RCNP). Gradient value magnitudes are clipped at 0.5. We train for a maximum of 500 epochs, with each epoch consisting of 1,000 iterations. We evaluate the performance of each model on the entire test set.

F.3. Kolmogorov Flow

The 2-D Kologorov flow PDE is defined as

$$\delta_t \mathbf{u} + \nabla \cdot (\mathbf{u} \otimes \mathbf{u}) = \nu \nabla^2 \mathbf{u} - \frac{1}{\rho} \nabla p + \mathbf{f} \tag{40}$$

where $\mathbf{u}$ is the velocity field, $\otimes$ the tensor product, ν the kinematic viscosity, ρ the fluid density, p the pressure field, and $\mathbf{f}$ the external forcing. We choose a 2-D domain with periodic boundary conditions. We use the data made available by Rozet & Louppe (2023), in which the PDE is solved on a 256×256 grid, coarsened to a 64×64 resolution and integration time between snapshots of $\Delta = 0.2$ time units, with 64 snapshots in total for each trajectory. The overall dataset consists of 1,024 independent trajectories of 64 states, of which 819 are used for training and 102 for testing. We sub-sample $16 \times 16 \times 16$ regions from these $64 \times 64 \times 64$ trajectories to construct individual tasks. We seek to model the velocity field, $\mathbf{u}(\mathbf{x}, t) \in \mathbb{R}^2$. For each task, we sample $N_c \sim U(1, 500)$ and set N_t to all remaining points. Input values are normalised to lie in the range $[-3, 3]$.

For all models, we use an embedding / token size of $D_z = 32$. We use a decoder consisting of an MLP with two hidden layers of dimension D_z. The decoder parameterises the mean and pre-softplus variance of a Gaussian likelihood with heterogeneous noise. Model specific architectures are as follows:

PT-TNP Same as Section F.1.

RCNP Same as Section F.1.

CNP Same as Section F.1.

TE-PT-TNP Same as Section F.1.

Multi-task GP For the multi-task GP baseline, we model the covariance between the i-th output at $\mathbf{x}$ and j-th output at $\mathbf{x}'$ using the multi-task kernel with diagonal observation noise:

$$k([\mathbf{x}, i], [\mathbf{x}', j]) = k_{\text{se}}(\mathbf{x}, \mathbf{x}') \times k_{\text{tasks}}(i, j) + \sigma_n^2 \delta(\mathbf{x} - \mathbf{x}', i - j). \tag{41}$$

where k_{tasks} is the inter-task covariance, which in this case is a 2×2 lookup table, and k_{se} is an SE kernel with independent lengthscales for each input dimension:

$$k_{\text{se}}(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp\left(-\sum_{i=1}^{D_x} \frac{(x_i - x_i')^2}{2\ell_i^2} \right) + \sigma_n^2 \delta\left(\mathbf{x} - \mathbf{x}'\right). \tag{42}$$

The GPs are implemented using GPytorch (Gardner et al., 2018), and optimisation of hyperparameters is performed using Adam (Kingma & Ba, 2014) for 1,000 iterations with a learning rate of 1×10^{-1}.

Training Details For all NP models, we optimise the model parameters using AdamW (Loshchilov & Hutter, 2017) with a learning rate of 5×10^{-4} and batch size of 16 (8 for the TE-PT-TNP and RCNP). Gradient value magnitudes are clipped at 0.5. We train for a maximum of 500 epochs, with each epoch consisting of 10,000 iterations. We evaluate the performance of each model on the entire test set.

F.4. Environmental Data

The environmental dataset consists of surface air temperatures derived from the fifth generation of the European Centre for Medium-Range Weather Forecasts (ECMWF) atmospheric reanalyses (ERA5) (Copernicus Climate Change Service, 2020). The data has a latitudinal and longitudinal resolution of $0.5°$, and temporal resolution of an hour. We consider data collected in 2019, sub-sampled at a temporal resolution of six hours. The training set consists of data within the latitude / longitude range of $[42°,\ 53°]$ / $[8°,\ 28°]$ (roughly corresponding to central Europe), and the test sets consists of two non-overlapping regions: western Europe ($[42°,\ 53°]$ / $[-4°,\ 8°]$), and northern Europe ($[53°,\ 62°]$ / $[8°,\ 28°]$). Individual datasets are obtained by sub-sampling the large regions, with each dataset consists of a $[15, 15, 5]$ grid spanning $7.5°$ across each axis and 30 hours. We also provide surface elevation as additional inputs, such that $D_x = 4$. The inputs and outputs are standardised using the mean and standard deviation values obtained from data within the training region. Each dataset consists of a maximum of $N = 1,125$ datapoints, from which the number of context points are sampled according to $N_c \sim U(\frac{N}{100}, \frac{N}{3})$, with the remaining set as target points.

For all models, we use an embedding / token size of $D_z = 32$. As with the image-completion experiment, we were limited by the hardware available. We use a decoder consisting of an MLP with two hidden layers of dimension D_z. The decoder parameterises the mean and pre-softplus variance of a Gaussian likelihood with heterogeneous noise. Model specific architectures are as follows:

PT-TNP Same as Section F.1.

RCNP Same as Section F.1.

CNP Same as Section F.1.

TE-PT-TNP Same as Section F.1.

GP For the GP baseline, we model the observations using an SE kernel with independent lengthscales for each input dimension plus observation noise:

$$k(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp\left(-\sum_{i=1}^{D_x} \frac{(x_i - x_i')^2}{2\ell_i^2} \right) + \sigma_n^2 \delta\left(\mathbf{x} - \mathbf{x}'\right). \tag{43}$$

The GPs are implemented using GPytorch (Gardner et al., 2018), and optimisation of hyperparameters is performed using Adam (Kingma & Ba, 2014) for 1,000 iterations with a learning rate of 1×10^{-1}.

Training Details For all NP models, we optimise the model parameters using AdamW (Loshchilov & Hutter, 2017) with a learning rate of 5×10^{-4} and batch size of 16 (8 for the TE-PT-TNP and RCNP). Gradient value magnitudes are clipped at 0.5. We train for a maximum of 500 epochs, with each epoch consisting of 10,000 iterations. We evaluate the performance of each model on the entire test set.

F.5. The Effectiveness of Dynamically Updating Input Locations

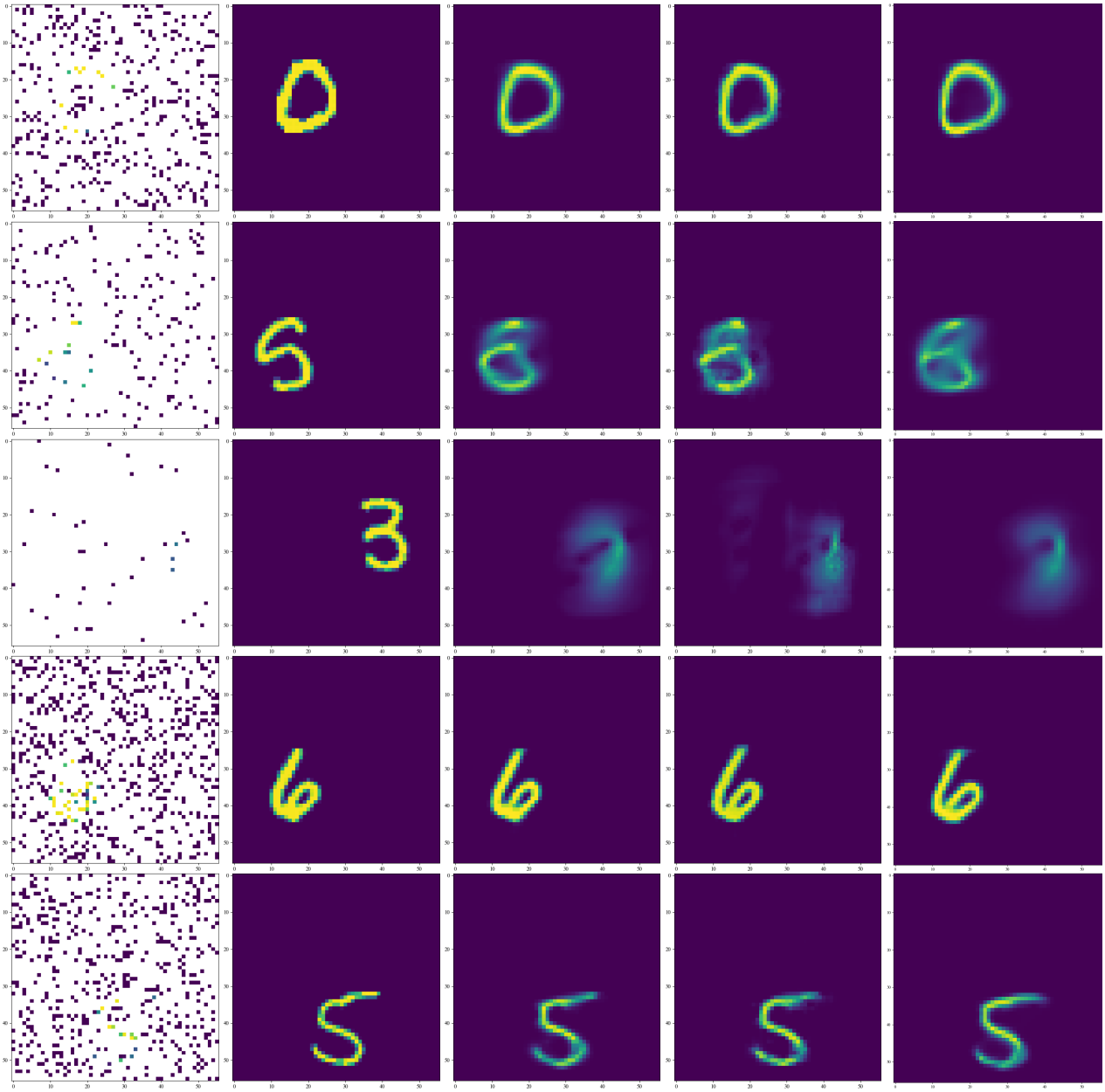
Here, we perform a simple ablation study to determine the effectiveness of dynamically updating input locations using Equation 11 and Equation 10 in the TE-PT-TNP. We first consider models trained on the synthetic-1D regression dataset in Section F.1 evaluated on two test sets: one drawn from the same distribution as the train tasks, and another for which the inputs are sampled according to the hierarchical model:

$$\begin{aligned} c &\sim \text{Bernoulli}(0.5) \\ x &\sim \begin{cases} U(-4, -1) & \text{if } c = 0, \\ U(1, 4) & \text{if } c = 1. \end{cases} \end{aligned} \tag{44}$$

In both cases, observations are sampled as described in Section F.1. The difficulty in this second task is that the bimodality of the input distribution wants the pseudo-locations to also be bimodal, however, the initial pseudo-tokens will have a roughly equal pull in both directions due to symmetry about 0. We therefore posit that it becomes increasingly necessary to dynamically update the input locations to account for this. We consider the TE-PT-TNP-M32 model described in Section F.1, and the same TE-PT-TNP-M32 model with $\psi(\mathbf{u}_m, \mathbf{z}_n)$ in Equation 11 set to $\psi(\mathbf{u}_m, \mathbf{z}_n) = 1/N$ and no Equation 10. Note that this model is trained from scratch on the same training dataset. The results are shown in Table 7. We observe that for the uniformly distributed inputs, there is no significant different in the performance of the two models. However, for the bimodal inputs the model without input adjustment performs significantly worse.

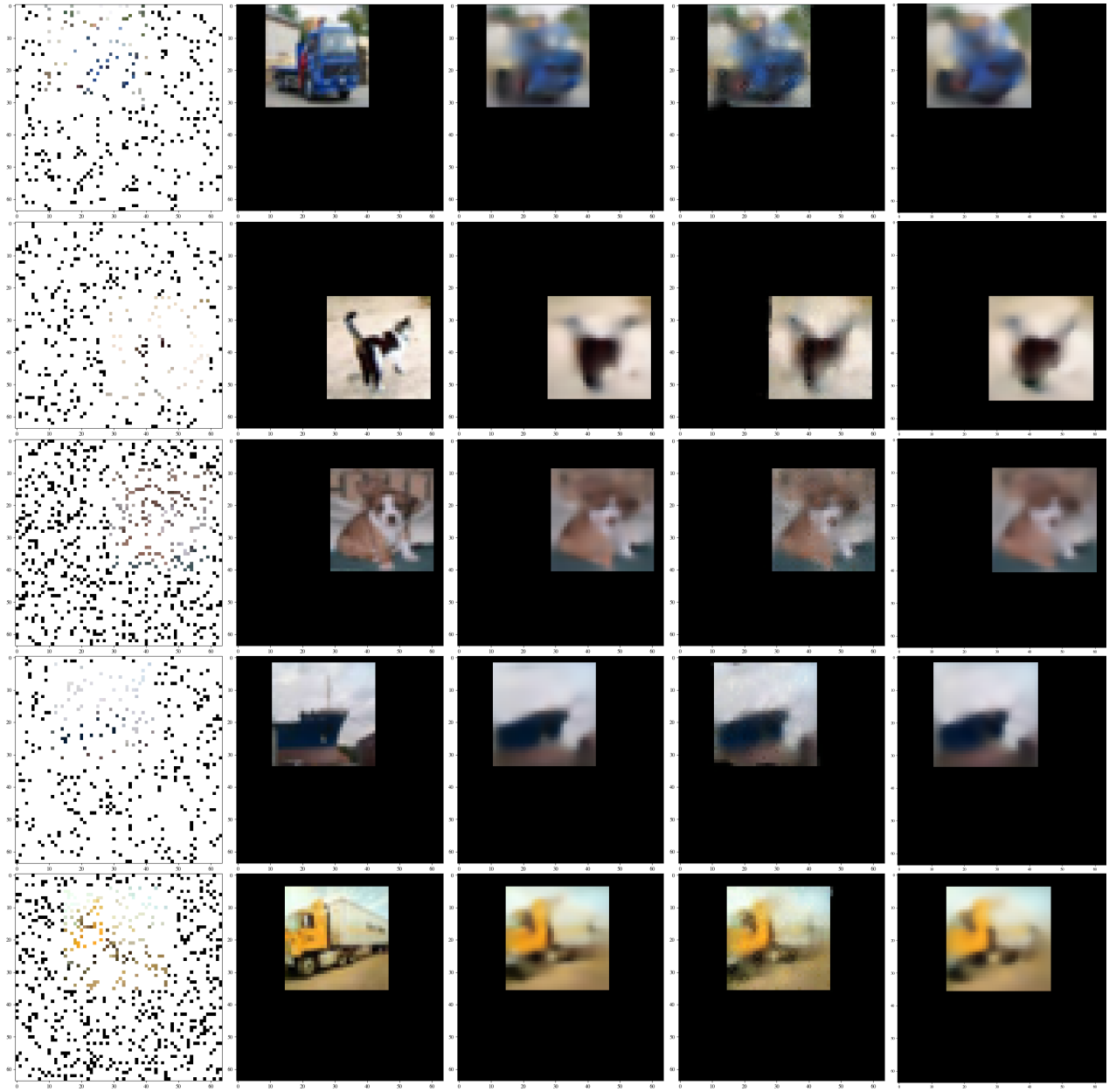
Table 7. Average log-likelihood (↑) on the two test datasets.

Model	Uniform Inputs	Bimodal Inputs
TE-PT-TNP-M32	$\mathbf{-0.44 \pm 0.01}$	$\mathbf{-0.67 \pm 0.01}$
TE-PT-TNP-M32 no input adjustment	$\mathbf{-0.46 \pm 0.01}$	-0.76 ± 0.01



(a) Context dataset. **(b)** Ground truth data. **(c)** TE-PT-TNP-M64. **(d)** ConvCNP. **(e)** PT-TNP-M64.

Figure 8. A comparison between the predictive mean of the TE-PT-TNP-M64 model, ConvCNP model and PT-TNP-M64 model, given the context datasets on the left.



(a) Context dataset. **(b)** Ground truth data. **(c)** TE-PT-TNP-M128 **(d)** ConvCNP. **(e)** PT-TNP-M128.

Figure 9. A comparison between the predictive mean of the TE-PT-TNP-M128 model, ConvCNP model and PT-TNP-M128 model, given the context datasets on the left.