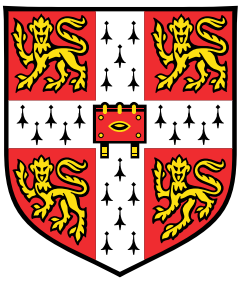
{On, Off}-The-Grid Data Modelling Using Transformer Neural Processes



Eric Robin Langezaal

Department of Engineering
University of Cambridge

This dissertation is submitted for the degree of
Master of Philosophy in Machine Learning and Machine Intelligence

Gonville & Caius College August 2024

Voor mijn familie, oude én nieuwe vrienden, door wie deze droom werkelijkheid werd.

Declaration

I, Eric Robin Langezaal of Gonville & Caius College, being a candidate for the MPhil in Machine Learning and Machine Intelligence, hereby declare that this report and the work described in it are my own work, unaided except as may be specified below, and that the report does not contain material that has already been used to any substantial extent for a comparable purpose.

Software declaration: This research builds upon a modified fork of the Python TNP/ICICL codebase created by my co-supervisor Matthew Ashman. Note that dual modality data, grid-encoding or structured data transformers were not originally part of this codebase, such that everything described in the methodology or experiments chapters stems from novel work. This codebase uses standard deep learning libraries such as PyTorch and Weights & Biases. The only non-synthetic dataset used is ECMWF Reanalysis v5 (ERA5), retrieved from the Copernicus Climate Data Store. The codebase used for this research is available at https://github.com/EricLangezaal/tnp-pp.

Word count: 14304.

Eric Robin Langezaal
August 2024

Acknowledgements

I would firstly like to thank my supervisor Professor Richard Turner, whose inspiring lectures motivated me to pursue his research proposal, and for whose invaluable guidance and continuous support I am incredibly grateful. His knowledge and enthusiasm continue to amaze me, and I am very thankful to have been supervised by him.

I also want to express my gratitude to my co-supervisors Matthew Ashman and Cristiana Diaconu, who have both taught me so much in the span of just a couple months. They helped create a fun and motivating environment where I felt encouraged to ask any question, and were always available to discuss research or look over programming questions. Their guidance steered me in the right direction, and without their help I would have never been able to complete this project. I also want to thank Matthew for creating and teaching me about the original TNP codebase that helped accelerate this research.

My gratitude goes towards Gonville & Caius college, the lively postgraduate community, and the amazing friends I made along the way. Thank you Ben, Ivan, James, Josh and Matias, without you life in Cambridge would not have been the same. I want to thank Rupert, Arielle, Pablo, Zsigmond and Leander who helped make the MPhil so much more fun.

Next I want to express my gratitude towards my friends from home, both from Leiden and from Amsterdam, who have continuously supported me for years. I deeply appreciate Fergus, Anna, Cas and Jeffrey for visiting me in Cambridge, which made me feel so much more at home. I want to thank Imme, for her support and belief in me, and for everything she taught me.

Lastly, I express my gratitude to my family, my grandparents and parents in particular, who supported my journey every step of the way. They empowered me to successfully apply to Cambridge, and helped me to move to and feel safe in another country.

Abstract

Neural processes (NPs) are a powerful class of meta-learning algorithms, where a model that is provided with a variable size context dataset and target points, is trained to do data efficient learning. In large-scale spatio-temporal modelling, such as when predicting environmental variables, mostly convolutional conditional neural processes (ConvCNPs) have been employed, achieving state-of-the-art results. Both from other research areas such as computer vision, and inspired by initial results, it is expected that transformer based neural processes (TNPs) can outperform convolutional approaches. A major bottleneck is the computational complexity of TNPs, which is quadratic in the number of datapoints, preventing the application of TNPs at scale. A second issue arises from the nature of environmental data, which is often comprised of gridded as well as off-the-grid data, where a model should ideally be able to learn from both.

This thesis aims to address both aforementioned challenges simultaneously. Firstly, to capture trends in both on- and off-the-grid data, both modalities are jointly encoded to a grid of structured pseudo-tokens. Two such grid-encoders are tested, the first inspired by ConvCNPs and the latter being a novel approach based on batched Multi-Head Cross-Attention (MHCA). Next, computationally efficient design principles, some inspired by the computer vision literature, are developed for structured data transformers. This entails different approaches to coarsen the grid of pseudo-tokens, shifted windows self-attention or efficient nearest-neighbour based cross-attention.

The models and pipelines developed in this research are evaluated on two different datasets. Initial experiments are conducted on synthetic multitask Gaussian process data, showing that the grid-encoders can effectively capture trends in both data modalities. To demonstrate efficiency and performance at a large scale, models are trained on multiple years of hourly ERA5 surface temperature data. The new models succeed in predicting unseen ERA5 measurements accurately, substantially outperforming comparable approaches, while achieving over 80 times more iterations per second than a naive TNP with quadratic computational complexity.

Table of contents

List of Illustrations

Figures

Tables

Nomenclature

Acronyms / Abbreviations

CNP	Conditional Neural Process
ConvCNP	Convolutional Conditional Neural Process
EQ	Exponentiated-Quadratic (kernel function)
GC	Grid Coarsening (inside grid-encoder)
LBANP	Latent Bottlenecked Attentive Neural Process
GP	Gaussian Process
MHCA	Multi-Head Cross-Attention (grid encoder)
MHSA	Multi-Head Self-Attention
(K)NN-CA	(K)-Nearest Neighbour (multi-head) Cross-Attention
NP	Neural Process
NWP	Numerical Weather Prediction
PC	Patch Coarsening (or patch embedding)
PPU	Points Per Unit
PT-TNP	Pseudo Token Transformer Neural Process
SetConv	Set Convolution (grid encoder)
Swin	Shifted Windows (attention)
TNP	Transformer Neural Process
ViT	Vision Transformer

Chapter 1

Introduction

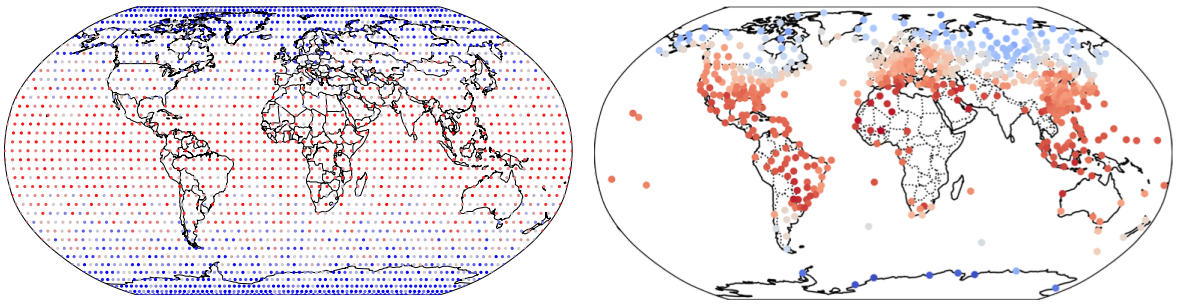
Recent years have seen immense research and development in the application of deep neural networks to increasingly complex problems [Fan et al., 2021; Sarker, 2021; Schmidhuber, 2015]. Most such networks focus on function approximation, learning a direct mapping between datapoints and their labelled targets. These approaches require an increasing amount of data to perform well [Bejani and Ghatee, 2021], and can illustrate poor transferability to related problems.

Meta-learning [Huisman et al., 2021] is a powerful didactic strategy, where a model is specifically trained to be able to infer general patterns from a dataset, without focussing on a specific downstream application, allowing it to train the learning process itself [Reed et al., 2017; Wang et al., 2016]. *Neural processes* (NPs) [Garnelo et al., 2018a,b] are a very effective relatively novel design paradigm for meta-learning, where the model is trained to do data efficient learning, based on a variable size context dataset and some targets datapoints. The model learns to capture the posterior predictive distribution over target outputs given the context set, under the data distribution that generated the training datasets.

The aggregation of the context set in neural processes can be achieved through different neural architectures. Convolutional Conditional Neural Processes (ConvCNPs) [Foong et al., 2020; Gordon et al., 2019] utilise convolutional neural networks and have achieved far better results compared to regular NPs when the inputs include spatial, temporal or spatio-temporal components. Transformer Neural Processes (TNPs) [Kim et al., 2019; Nguyen and Grover, 2022] utilise a transformer architecture instead, inspired by their state-of-the-art performance in various machine learning domains [Lin et al., 2022]. It has been shown that in computer vision, transformer models can outperform CNN based models, despite having significantly fewer parameters [Khan et al., 2022; Maurício et al., 2023].

However, a major problem for TNPs stems from their quadratic computational complexity, caused by the pairwise attention calculation between all datapoints. Pseudo Token Transformer Neural Processess (PT-TNPs) [Feng et al., 2022; Lee et al., 2019] sidestep this bottleneck by instead only calculating attention with latent pseudo-tokens extracted from the original data, but methods for constructing these pseudo-tokens are an area of research.

This thesis will focus specifically on the application of TNPs in the domain of environmental science, where datasets are often enormous, requiring architectures that are computationally efficient. Traditionally, problems in environmental modelling are approached by simulating a physics-based world model at a high fidelity. While this approach to Numerical Weather Prediction (NWP) achieves state-of-the-art results [Bauer et al., 2015], it is also increasingly computationally expensive when the fidelity of the predictions is scaled up. Recently, data-driven approaches have started to match the performance of operational NWP systems by replacing certain steps of the pipeline by machine learning models [Bi et al., 2022; Lam et al., 2023; Price et al., 2023]. More specifically, (convolutional) neural processes have also been used for multiple prediction tasks in the domain of climate or environmental modelling [Andersson et al., 2023; Vaughan et al., 2022; Vaughan et al., 2024].



(a) On-the-grid (structured) data **(b) Off**-the-grid (unstructured) data

Figure 1.1: Example of the distinction between the two modalities of environmental data modelled throughout this research. On-the-grid data (**left**) occurs at regular spatial intervals and is for example captured by satellites. Off-the-grid data (**right**) can be measured at any location, often being non-uniformly distributed.

Environmental data can be distributed across different variables, each having a potentially different composition, further complicating model design. An important distinction is made between two radically different types (or modalities) of environmental variables. The first modality is off-the-grid or unstructured data, which is collected at arbitrary locations on the globe. The second modality is on-the-grid or structured data, which occurs at regular equidistant spatial intervals. This combination of on- and off-the-grid data is ubiquitous in environmental research and can for example be found in meteorology, air quality modelling, ocean modelling, or weather prediction. In the weather prediction domain for example, there

is often a combination of on-the-grid satellite data and off-the-grid land or marine weather station measurements. The latter are typically concentrated near densely populated areas in high-income countries [Vaughan et al., 2024]. When modelling the oceans, numerical ocean models (on-the-grid) and off-the-grid measurements from robotic instruments [Wong et al., 2020] both contain crucial information. See figure 1.1 for a visual depiction of the difference between on- and off-the-grid data, considering for example global temperature measurements.

Some current environmental studies [Andersson et al., 2023; Bi et al., 2022] focus exclusively on either on- or off-the-grid data. However, both modalities of environmental data are often highly correlated since they stem from the same underlying physical variables. Due to these high correlations, modelling the union of both modalities would likely aid performance on downstream tasks, especially when the coverage of either separate modality might be limited [Gettelman et al., 2022; Vaughan et al., 2024].

The current best approaches for efficient PT-TNPs for off-the-grid data, such as the LBANP [Feng et al., 2022], do not maintain their predictive performance when scaling up the input data to the enormous size used for environmental science. Hence simply treating all structured data as a collection of off-the-grid datapoints will not suffice. Efficient transformers from the computer vision domain are known to perform at scale, but can natively handle on-the-grid data only.

The two main goals of this research are therefore to combine elements from both PT-TNPs and efficient vision transformers to create a model that can process on- and off-the-grid data on the one hand, while performing computationally efficient and accurate predictions on the other hand. In order to achieve both these goals, this research chooses to map the unstructured data to a grid, such that it can be processed in conjunction with the on-the-grid data by an efficient transformer for structured data. The novel models therefore aim to generate a grid of pseudo-tokens from the combination of both modalities of data. This study will explore multiple options to create this grid, as well as experiment with different setups for the structured data transformers, inspired by architectures like the Vision Transformer (ViT) [Dosovitskiy et al., 2020] or the Shifted Windows (Swin) Transformer [Liu et al., 2021].

1.1 Thesis contributions

The main contributions of this project are as follows:

1. **Approaches to encode on- and off-the-grid data to structured pseudo-tokens**: this thesis develops two grid-encoders to map both data modalities onto a grid of pseudo-tokens.
    - An approach based on the function-space embeddings that allow ConvCNPs to map off-the-grid data to an artificial grid [Gordon et al., 2019]. Our TNP model utilises this existing mechanism to transform the off-the-grid data, while leaving the on-the-grid data unaltered, after which both can be combined.
    - A second completely new approach based on efficient batched *multi-head cross-attention* (MHCA) between local neighbourhoods of datapoints and latent embeddings that exist on a grid. To calculate attention, this approach constructs a pseudo-batch for every on-the-grid datapoint, which also contains all its nearest off-the-grid neighbours.
2. **Efficient structured data transformers for TNPs:** After putting both modalities of data on a grid, this thesis adapts and constructs architectures to achieve efficient computation for structured pseudo-tokens.
    - Two approaches to coarsen high resolution grids are developed. The first is inspired by the patch embeddings inside a ViT, while the latter makes the grid-encoders natively map to a coarser grid.
    - A custom, efficient implementation of shifted windows self-attention for arbitrary dimensions is created, inspired by the Swin Transformer.
    - Lastly, a K-nearest neighbours cross-attention mechanism is constructed, allowing for more efficient MHCA between the pseudo-tokens and the target set.
3. **The development of datasets and pipelines to train dual modality TNPs**: Initial experiments are conducted on synthetic on- and off-the-grid data, created through correlated multitask Gaussian Processes [Bonilla et al., 2007; Rasmussen, 2003]. Later experiments will focus on predicting real-world environmental variables, showcasing the ability to learn from multiple years of hourly measurements.

1.2 Thesis outline

In the next chapter, the background of multiple topics related to this research are examined in greater detail. This entails different (transformer) neural process architectures as well as efficient structured data transformers from the computer vision domain.

Chapter 3 outlines the various methodologies developed in this research. Specific details regarding the datasets are also discussed, finally focussing on the complete modelling pipeline and possible permutations thereof.

The fourth chapter covers experimental results and assesses the performance of various models. This chapter is split between initial experiments on synthetic data, and larger scale real-world experiments.

Finally, the last chapter gives a macro level overview of the developed architectures and achieved results and places these in a wider context. This concludes with a discussion of potential avenues for future research.

Chapter 2

Background

This chapter will examine the theoretical foundations of this research in greater detail. First general meta-learning is briefly described, after which the focus is shifted towards the neural process family. Although this thesis focusses on TNPs, some of the developed approaches build on findings from convolutional CNPs, which are covered in section 2.1.2. Section 2.2 outlines the background regarding Transformers, both in the context of NPs as well as research regarding efficient Transformers from the field of computer vision.

2.1 Meta-learning

In the broader domain of machine learning, meta-learning [Huisman et al., 2021] is a particular strategy of learning, focusing on a higher-level abstraction as opposed to solving a singular problem [Reed et al., 2017]. Without aiming for a particular fixed downstream task, the goal is *"Learning to learn"* in general. As such, the model is not trained on a singular dataset, but should instead perform well on a *set* of datasets. This is in stark contrast to most supervised learning, where hyperparameters are often tuned towards a specific dataset, with poorer generalisability as a result. The following paragraphs will demonstrate this more formally, with notation tailored towards neural processes, but nonetheless explaining meta-learning in general.

Let $T_i \in \mathbf{T}$ be a specific machine learning task or problem, drawn from a collection of tasks $\mathbf{T}$. Each such task T_i comprises a tuple $(D_c, D_t, \mathscr{L})$ of three elements. The first two elements are the *context* dataset $D_c = \{x_n^{(c)}, y_n^{(c)}\}_{n=1}^{N_c}$ and *target* dataset $D_t = \{x_n^{(t)}, y_n^{(t)}\}_{n=1}^{N_t}$ with potentially different numbers of elements N_c and N_t respectively. Note that for both datasets, x_n entails a singular observed datapoint, with y_n being the associated label. The context dataset is fully

observed, but for the target dataset, only the datapoints $x_n^{(t)}$ are known, with the labels $y_n^{(t)}$ being held-out for evaluation purposes. The last element of the tuple defines the task-specific loss function $\mathscr{L}$, which defines the quality of a model's prediction given the true data label.

In classical supervised learning, one would define a model $f_{\theta^{(i)}}^{(i)}$ for each task T_i, with specific parameters $\theta^{(i)} \in \Theta$. This model is trained for a specific task on the context data, aiming to maximise performance on the loss function as defined in equation 2.1. This allows the model to find task optimal parameters $\theta_*^{(i)}$, such that it can predict $\hat{y_n}^{(t)} = f_{\theta^*}^{(i)}(x_n^{(t)})$ for target inputs, which can then be compared to the true $y_n^{(t)}$ to evaluate the trained model.

$$\theta_*^{(i)} = \underset{\theta^{(i)}}{\operatorname{argmax}} \Big\{ \sum_{n=1}^{N_c} \mathscr{L}\big(f_{\theta^{(i)}}^{(i)}(x_n^{(c)}), y_n^{(c)}\big) \Big\} \tag{2.1}$$

In meta-learning however, the procedure is crucially different. Instead of learning a model $f_{\theta^{(i)}}^{(i)}$ for a specific task T_i, a general model g_θ is trained, with the aim of achieving good performance on all tasks in $\mathbf{T}$. To formalise this, the notation $X_t = \{x_n^{(t)}\}_{n=1}^{N_t}$ and $Y_t = \{y_n^{(t)}\}_{n=1}^{N_t}$ is introduced, referring to the observations or labels of the whole target set respectively. For each task, this model is passed the entire context set D_c and all target datapoints X_t in a single forward pass and should then aim to predict the target labels Y_t. The model therefore maximises global parameters $\theta \in \Theta$, illustrated in equation 2.2

$$\theta_* = \underset{\theta}{\operatorname{argmax}} \Big\{ \mathbb{E}_{T_i \sim \mathbf{T}} \big[\mathscr{L}\big(g_\theta(D_c^{(i)}, X_t^{(i)}), Y_t^{(i)}\big) \big] \Big\} \tag{2.2}$$

The meta-learning paradigm allows a model to learn general patterns from a large set of datasets, while still maintaining the flexibility to tailor its predictions given a specific context set. It is therefore particularly useful in regimes with large collections of non-homogeneous data, where this model has to do data efficient learning. The next sections will detail neural processes in greater detail, which form the specific implementation of meta-learning which is the focus of this research.

2.1.1 (Conditional) Neural Processes

Neural Processes (NPs) [Garnelo et al., 2018a,b] were inspired by the combination of neural networks with Gaussian Processes (GPs) [Rasmussen, 2003], aiming to remedy the shortcomings of both. Neural networks require very little manual feature engineering and are computationally efficient at inference, but require a lot of data to be trained properly and require extra

considerations to model probability distributions. GPs instead natively handle distributions over functions including the associated uncertainty, and are effective in regimes with sparse data. Unfortunately, GPs require handcrafted kernel functions and their implementations can have cubic computation cost, unless approximations are used, but these approximations are often still comparatively costly [Quinonero-Candela and Rasmussen, 2005]. Neural processes aim to combine the benefits of GPs and neural networks.

NPs model the probability of the target outputs provided the target inputs and context set:

$$p(Y_t|X_t, D_c) \qquad Y_t = \{y_n^{(t)}\}_{n=1}^{N_t}, \quad X_t = \{x_n^{(t)}\}_{n=1}^{N_t} \tag{2.3}$$

Often NPs are defined through an encoder-decoder architecture, and the design of the original formulation by Garnelo et al. [2018b] will be elaborated below. Their NP encoder $h : X \times Y \mapsto R$ takes in a pair $(x_n^{(c)}, y_n^{(c)})$ and outputs an embedded representation $r_n^{(c)}$. This encoder is parametrised with a neural network, therefore avoiding the need to manually engineer features. The next step is to aggregate this collection of context representations in an order-invariant manner to form a fixed length vector $\mathbf{r}$, such that the NP can handle variable size context sets in single forward pass. Such an order-invariant set function is crucial for NPs. Both steps so far can be written as a valid deepset function approximator over the context set analogous to Zaheer et al. [2017] as per equation 2.4.

$$\mathbf{r} = \rho\Big(\sum_{n=1}^{N_c} h(x_n^{(c)}, y_n^{(c)})\Big) \tag{2.4}$$

In most NPs and notably in the first paper by Garnelo et al. [2018b], $\rho(\dots)$ simply amounts to dividing by the number of context datapoints, so that $\mathbf{r}$ effectively averages all r_n. This compressed representation of the context dataset $\mathbf{r}$ then parameterises a latent distribution, from which a global latent variable $\mathbf{z}$ is sampled. The final component of the NP consists of a decoder g, which takes as input the global latent variable $\mathbf{z}$ and each target datapoint $x^{(t)}$ and outputs (a probability distribution over) $y^{(t)}$. This decoder is again parameterised with a neural network. This entire process is summarised in the schematic in figure 2.1.

Although previously published, Conditional Neural Processes (CNPs) [Garnelo et al., 2018a] form a subclass of NPs. CNPs no longer rely on a latent variable to indirectly specify the output distribution and instead push $\mathbf{r}$ directly through the decoder. This can be seen as a specific form of a NP, where $\mathbf{z}$ is (deterministically) sampled from a Dirac delta distribution centered around $\mathbf{r}$. This allows for simpler training through maximum likelihood learning as opposed to a standard neural processes, which must be trained using an approximation such as variational

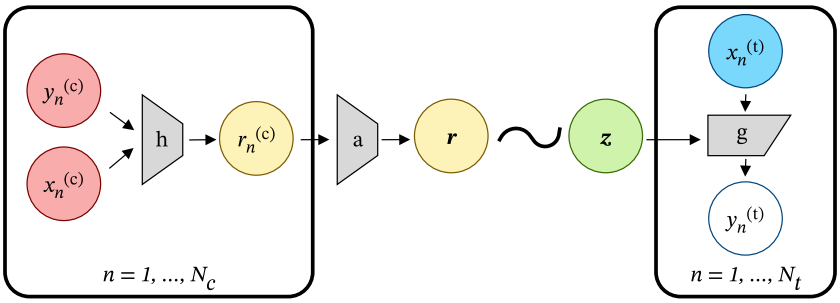


Figure 2.1: Graphical depiction of a generic neural process. Variables are denoted using circles, functions are indicated by trapezoidal shapes. The variables $x_n^{(c)}$, $y_n^{(c)}$ and $x_n^{(t)}$ are observed, with $y_n^{(t)}$ being predicted. Note that a Conditional NP omits the latent variable $\mathbf{z}$, instead deterministically passing the aggregated context set to the decoder. This decoder outputs deterministically, but it is also common to parameterise a distribution instead.

inference on an evidence lower bound (ELBO) [Blei et al., 2017]. A CNP can for example be summarised through equation 2.5, here parameterising a Gaussian likelihood.

$$p(y_n^{(t)}|x_n^{(t)}, D_c) = \mathcal{N}(y_n^{(t)}; \mu_n^{(t)}, \sigma^{2(t)}_n), \quad \left(\mu_n^{(t)}, \sigma^{2(t)}_n\right) = g(x_n^{(t)}, \frac{1}{N_c}\sum_{n=1}^{N_c} h(x_n^{(c)}, y_n^{(c)})) \tag{2.5}$$

2.1.2 Convolutional Conditional Neural Processes

Convolutional CNPs (ConvCNPs) [Gordon et al., 2019] are a substantially improved version of basic CNPs, introducing a more intelligent context set aggregation. ConvCNPs encode *translation equivariance*, which entails that a translation in input space yields a corresponding translation in output space [Cohen and Welling, 2016; Kondor and Trivedi, 2018]. Having translation equivariance built-in as an inductive bias would likely aid neural processes in many spatio-temporal problems, where NPs have shown promising results in the past [Kim et al., 2019]. Defining a translation by τ as T_τ and an analogous translation in function space as T'_τ allows the formalisation in equation 2.6 for a function $\phi \in \Phi$.

$$T_\tau D_c = \{x_n^{(c)} + \tau, y_n^{(c)}\}_{n=1}^{N_c}, \qquad T'_\tau \phi(x) = \phi(x - \tau) \tag{2.6}$$

A function ϕ can now be called translation equivariant if and only if $\phi(T_\tau D_c) = T'_\tau \phi(D_c)$, where T'_τ is the corresponding mapping in function space.

Gordon et al. [2019] have extended the work of Zaheer et al. [2017], by introducing Convolutional Deep Sets, which extend to sets of functional representations. To achieve translation equivariance, ConvCNPs use a custom Set-Convolution encoder (h) which maps into function

space, since translation equivariance is not well-defined for fixed vector space. They embed the sets into an infinite dimensional space by using a flexible positive definite kernel associated with a Reproducing Kernel Hilbert Space (RKHS) [Aronszajn, 1950]. In practice, an exponentiated-quadratic (EQ) kernel with a learnable length scale parameter is often used for this embedding.

After encoding the context set, it remains to specify the type of neural architecture used for the decoder g. From computer vision, it is known that the convolutional layers in CNNs [Li et al., 2022; O'shea and Nash, 2015] naturally exhibit translation equivariance [Cohen et al., 2021; Cohen and Welling, 2016]. CNNs were originally devised to work on structured images and are therefore readily applicable to on-the-grid data. For off-the-grid data, however, this structure cannot be directly exploited, meaning a prior transformation is required.

The notation in the upcoming paragraph is slightly different from the original formulation, instead using language from the transformer literature for consistency with the next sections. For off-the-grid data, ConvCNPs first discretise the data a uniform grid at pseudo-locations $\{t_i\}_{i=1}^T$. The context set is encoded with respect to its distance from each pseudo-location as $r_i = \sum_{n=1}^{N_c} \begin{bmatrix} 1 & y_n^{(c)} \end{bmatrix}^T \psi(t_i - x_n^{(c)})$, where $\psi(\cdot) : \mathbb{R} \mapsto \mathbb{R}$ are EQ-kernel functions. The pseudo-locations and context set embeddings are passed through the CNN, whose output in turn parameterises basis functions, which are weighted through the kernel functions ψ. This results in continuous functions that can be sampled at every possible target set input location. For off-the-grid data, the ConvCNP can be summarised as per equation 2.7, here modelling the predictive distribution as a Gaussian.

$$\{f(t_i)\}_{i=1}^T = \text{CNN}(\{t_i, r_i\}_{i=1}^T), \quad (\mu_n^{(t)}, {\sigma^2}_n^{(t)}) = \sum_{i=1}^T f(t_i)\psi(x_n^{(t)} - t_i) \tag{2.7}$$

Through this discretisation, ConvCNPs can be applied to both on-the-grid and off-the-grid data. They have been effective at introducing translation-equivariance to CNPs, increasing their performance at various downstream tasks [Foong et al., 2020; Gordon et al., 2019].

2.2 Transformers for Neural Processes

Several applications of neural processes to environmental modelling [Andersson et al., 2023; Foong et al., 2020; Vaughan et al., 2022] have previously consisted of ConvCNPs. In a lot of machine learning domains however, the start-of-the-art often involves *Transformers* rather than convolutional networks [Lin et al., 2022; Zhao et al., 2023]. Very recently, promising

research has also been conducted in the environmental domain with (non TNP) transformer based architectures [Bodnar et al., 2024; Lam et al., 2023]. Since their introduction in 2017 by Vaswani et al. [2017], Transformers have been dominant in natural language processing [Devlin et al., 2018; Gillioz et al., 2020]. Transformers are sequence models by default, naturally allowing for variable size input. Without a causal attention mask, transformers are set functions too, and can therefore parameterise the NP encoder without loss of generality. This section will hence focus on the application of Transformers in neural processes, first discussing the *attention* mechanism in the context of neural processes.

2.2.1 Multi-Head Attention (MHA)

Transformers naturally process sequences through the introduction of the attention mechanism, allowing the model to learn to what extent previous tokens contribute towards new tokens through a learned attention weight. To be able to abstract multiple representations simultaneously, separate attention calculations with different weights are often performed in parallel, commonly referred to as *multi-head attention*, denoted using heads $h \in H$.

To treat the input $x_i^{(c)}$ and target $y_i^{(c)}$ values in the context set as a pair while respecting context set permutation invariance, they are first jointly embedded, defined as $r_i^{(c)} = h(x_i^{(c)}, y_i^{(c)})$. Target inputs $x_m^{(t)}$ are also embedded, but with a zero vector in place of $y_m^{(t)}$ since their labels are not observed.

Attention can be calculated for a single sequence such as the context set, which is known as multi-head *self-attention* (MHSA). Provided two context set embeddings $r_i^{(c)}$ and $r_j^{(c)}$, and weight matrices $\mathbf{W}_{Q,h}$ and $\mathbf{W}_{K,h}$ the attention weight is defined as:

$$\alpha_h(r_i^{(c)}, r_j^{(c)}) = \frac{\exp\left([r_i^{(c)}]^T \mathbf{W}_{Q,h} \mathbf{W}_{K,h}^T r_j^{(c)}\right)}{\sum_{n=1}^{N^{(c)}} \exp\left([r_i^{(c)}]^T \mathbf{W}_{Q,h} \mathbf{W}_{K,h}^T r_n^{(c)}\right)} \tag{2.8}$$

The output of the MHSA calculation for the i'th element of the collection $r_{1:N^{(c)}}^{(c)}$, which is denoted through $\texttt{MHSA}_i(r_{1:N^{(c)}}^{(c)})$, can be calculated through equation 2.9, where $\mathbf{W}_{V,h}$ and $\mathbf{W}_O$ are weight matrices.

$$\texttt{MHSA}_i(r_{1:N^{(c)}}^{(c)}) = \texttt{cat}\left(\left\{\sum_{j=1}^{N^{(c)}} \alpha_h(r_i^{(c)}, r_j^{(c)})\, [r_j^{(c)}]^T \mathbf{W}_{V,h}\right\}_{h=1}^{H}\right) \mathbf{W}_O \tag{2.9}$$

Alternatively, attention can focus on two distinct sequences, which is called multi-head *cross-attention* (MHCA). Provided context set embeddings $r^{(c)}_{1:N^{(c)}}$ and target set embeddings $r^{(t)}_{1:N^{(t)}}$, the output for the i'th target set element can be calculated as per equation 2.10.

$$\texttt{MHCA}_i(r^{(t)}_{1:N^{(t)}}, r^{(c)}_{1:N^{(c)}}) = \texttt{cat}\Big(\Big\{\sum_{j=1}^{N^{(c)}} \alpha_h(r^{(t)}_i, r^{(c)}_j)\ [r^{(c)}_j]^T \mathbf{W}_{V,h}\Big\}_{h=1}^{H}\Big)\mathbf{W}_O \tag{2.10}$$

Having defined MHSA and MHCA in the context of transformer neural processes, the TNP itself will be discussed in the next section.

2.2.2 The standard Transformer Neural Process (TNP)

TNPs replace the NP encoder by a transformer [Kim et al., 2019; Nguyen and Grover, 2022]. Recall that for a neural process, we are aiming to predict target labels $y^{(t)}_n$ given a target input $x^{(t)}_n$ and a context dataset $D_c = \{x^{(c)}_n, y^{(c)}_n\}_{n=1}^{N_c}$. In the most general form, TNPs model the entire target set jointly as per equation 2.11.

$$p(\{y^{(t)}_n\}_{n=1}^{N_t} | \{x^{(t)}_n\}_{n=1}^{N_t}, D_c) = p(\{y^{(t)}_n\}_{n=1}^{N_t} | \texttt{TNP}(D_c, X_t)) \tag{2.11}$$

The original paper by Nguyen and Grover [2022] distinguishes between three slight variations of the architecture, of which only the variant with a diagonal covariance matrix (TNP-D) is computationally feasible at scale. Any further reference to a TNP will accordingly imply a TNP-D. The TNP-D assumes that the target datapoints are independent, and therefore models them according to a diagonal covariance matrix. As such it has similarities with the CNPs discussed in section 2.1.1.

The transformer inside the TNP is comprised of several alternating self- and cross-attention layers. The self-attention layers process the context set and cross-attention layers perform unmasked cross-attention between the context and target dataset. Important to note is that the TNP does not encode the relative position of the input in the context set (n), as that would induce an unwarranted order in the context set. This breaking with *context invariance* would imply the encoder is no longer a valid set function, [Nguyen and Grover, 2022], which could possibly make the model's outputs dependent on a fictitious ordering in the target set. The output of the attention layers is passed through an decoder MLP network $g(\dots)$, whose structure is the inverse of the MLP used for embedding originally. It maps from the embedding dimension to a mean and log-variance per datapoint that parameterise a heteroscedastic multivariate normal distribution with a diagonal covariance matrix over the target datapoints. For a schematic of

the TNP-architecture see figure 2.2. Equation 2.12 shows the probabilistic formulation of a TNP-D, with a single MHCA and MHSA layer.

$$p(\{y_n^{(t)}\}_{n=1}^{N_t}|\{x_n^{(t)}\}_{n=1}^{N_t}, D_c) = \prod_{n=1}^{N_t} p(y_n^{(t)}|\mu_n^{(t)}, \sigma^{2(t)}_n), \quad (\mu_n^{(t)}, \sigma^{2(t)}_n) = g(\texttt{MHCA}(r_n^{(t)}, \texttt{MHSA}(r_{1:N^{(c)}}^{(c)}))) \tag{2.12}$$

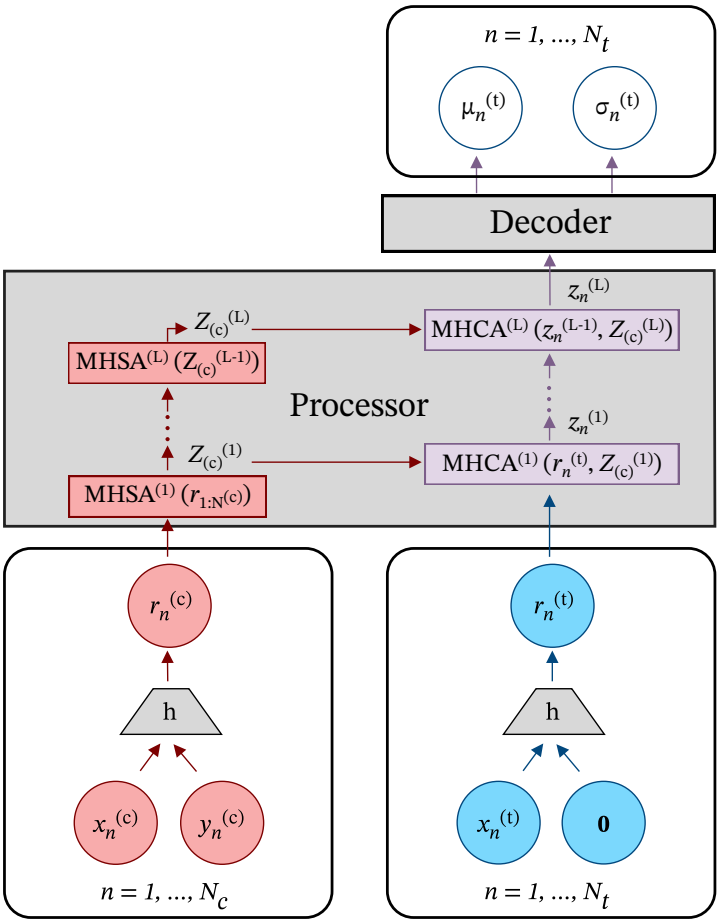


Figure 2.2: Graphical depiction of a Transformer Neural Process. Variables are denoted using circles, functions are indicated by trapezoidal shapes. Information related to context data is denoted in red, information related to the target data is coloured blue and operations derived from both are coloured purple.

TNPs have showcased promising initial results [Feng et al., 2022; Nguyen and Grover, 2022] and are therefore a interesting avenue of research. Unfortunately, regular TNPs have quadratic computational complexity in the number of context and target datapoints, due to their pairwise attention calculations. Therefore the next two sections will further examine known approaches

to address this computational bottleneck, making a distinction between approaches designed for off-the-grid and on-the-grid data.

2.2.3 Off-the-grid pseudo-token TNPs

The datasets prevalent in environmental modelling consist of both structured (on-the-grid) data, as well as off-the-grid data. By treating the on-the-grid data as a set of datapoints, both modalities could theoretically be combined without loss of generality. While this would be a perhaps inefficient representation of the originally structured data, it would allow for the application of approaches that reduce TNP complexity for off-the-grid data. One such approach are Pseudo-Token TNP's (PT-TNPs) [Feng et al., 2022; Lee et al., 2019]. Rather than having each (embedded) context point calculate pairwise attention with every other context point, PT-TNPs maintain a smaller number of latent *pseudo-tokens*, which summarise the influence of the full set of context tokens through cross-attention layers.

Feng et al. [2022] claim that their Latent-Bottlenecked Attentive NPs (LBANP) achieve state-of-the-art performance with a computational complexity only determined by the number of latent vectors, independent from the number of context datapoints. Provided initial latents $L^{(0)}$ and context and target embeddings, the LBANP calculates output $Z^{(1)}$ for a single layer as per equation 2.13.

$$L^{(1)} = \texttt{MHCA}(L^{(0)}, r^{(c)}_{1:N^{(c)}}), \quad L'^{(1)} = \texttt{MHSA}(L^{(1)}) \quad Z^{(1)} = \texttt{MHCA}(r^{(t)}_{1:N^{(t)}}, L'^{(1)}) \tag{2.13}$$

While the LBANP architecture is a lot more computationally efficient than regular TNPs, their results do illustrate that the claimed independence between performance and the number of context point does not fully hold at scale. For more complex problems, their efficient architecture did not rival TNP performance, unless the number of latent vectors was also scaled up, which in turn harms computational efficiency. As such PT-TNP architectures are a useful avenue of research, but cannot solely remedy the computational bottleneck of TNPs when processing large spatio-temporal datasets.

2.2.4 Efficient on-the-grid Transformers

While PT-TNPs are a relatively effective methodology for processing off-the-grid data, the inherent structure of on-the-grid data allows it to be more intelligently processed. Dedicated structured data transformer architectures have been shown to achieve state-of the-art performance in various domains [Lin et al., 2022]. In the computer vision literature [Han et al., 2023;

Khan et al., 2022; Maurício et al., 2023], transformer-based architectures have efficiently been applied to structured image data to outperform CNNs, despite having fewer parameters.

There are two major benefits to using dedicated structured data transformers. Firstly, datasets such as ImageNet [Deng et al., 2009] or Microsoft COCO [Lin et al., 2014], are commonly processed at image resolutions of 256 by 256 pixels or above. If the image were flattened into a sequence of pixel representations, this would imply a sequence of tens of thousands of tokens. Because naive attention is quadratic, this would yield attention matrices of hundreds of millions of elements, which cannot be computed nor stored in memory. Secondly, the inherent structure in the data is lost when an image is treated as a single sequence.

One of the first applications of transformers to computer vision is the ViT model as introduced by Dosovitskiy et al. [2020]. The main revelation was to first split the image into *patches*, which are then flattened and linearly embedded before being passed through the transformer encoder. More formally, an image $x \in \mathbb{R}^{H\times W\times C}$ is flattened into $x' \in \mathbb{R}^{N\times(P^2\times C)}$, with H and W the height and width of the image respectively, C the number of channels, P the patch-size (along one axis) and N the number of patches. The linear embedding can be constructed through $z = \texttt{MLP}(x') \in \mathbb{R}^{N\times D}$, where D is the dimensionality of the embedding. This way each patch summarises a region in the image into a high dimensional vector. The transformer encoder consists of several multi-head self attention layers, and through the patching the number of attention calculations is drastically reduced. For a full breakdown of the original ViT architecture see the left plot in figure 2.3, taken from Dosovitskiy et al. [2020].

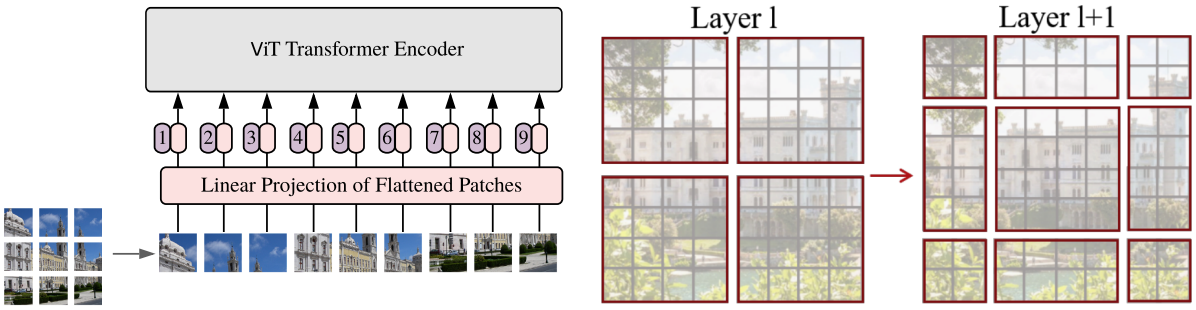


Figure 2.3: Left: ViT architecture, figure edited from Dosovitskiy et al. [2020]. An image (or grid) is split into patches, which are flattened and linearly projected to form the Transformer's input tokens. **Right:** Shifted windows inside a Swin Transformer [Liu et al., 2021], where a patch is outlined in grey and a window in red.

Dosovitskiy et al. [2020] found that when pretraining a ViT on large amounts of data, they were able to outperform the state-of-the-art with substantially fewer computational resources. However, they focus exclusively on classification, where an architecture needs to compress the input into a singular categorical class label at some stage in its pipeline. For their purpose,

effectively blurring a region into a singular patch was a sensible design choice. For other structured domains or downstream tasks which require high-fidelity, this might be undesirable. This is especially problematic, as the computational efficiency of the ViT is directly correlated to the patch size, and therefore inversely proportional to the fidelity level at which this architecture operates.

After the initial success of ViTs, Liu et al. [2021] created the Shifted Windows (Swin) Transformer. While this architecture still divides the image into patches, its main computational benefits stem from other design paradigms. This architecture features two specific new traits, namely the shifted windows attention calculation and hierarchical feature maps at increasingly coarse resolutions. Having multiple hierarchical feature maps requires significantly increased computation and is therefore left for future research, with only shifted windows attention being used in this research.

Instead of performing inefficient pairwise global attention, the Swin Transformer introduces the notion of local regularly spaced windows of patches. Each such window encompasses a few patches in each direction of the image, and attention is only calculated between patches in the same local window. The windowed self-attention has linear complexity with regard to the number of patches and only scales quadratically with the much smaller window size. However, having no connections across windows would strongly limit modelling power and lead to edge artefacts between neighbouring patches that happen to end up in different windows. To introduce attention across windows while maintaining efficiency, the breakthrough was to introduce layers with *shifted windows*. This shifted layer has identical window sizes, but the windows have been shifted by exactly half the window size, see the right plot in figure 2.3 for an illustration. More formally, given an output of the previous layer $z^{(l-1)}$, a window size M and number of windows (along one axis) K, the procedure can be defined as per equation 2.14, where $\texttt{W-MHSA}_{\{.\}}(\dots)$ indicates a windowed self-attention operation with the windows defined for each dimension in its subscript.

$$\hat{z}^{(l)} = \texttt{W-MHSA}_{\{0,\,M,\,\dots,\,KM\}}(z^{(l-1)}), \quad z^{(l)} = \texttt{W-MHSA}_{\{\lfloor \frac{M}{2} \rfloor,\, \lfloor \frac{3M}{2} \rfloor,\, \dots,\, \lfloor \frac{2KM-1}{2} \rfloor\}}(\hat{z}^{(l)}) \tag{2.14}$$

A final consideration to mention regarding shifted windows is the efficient batch-computation for windows at the edge of the input. After shifting, these windows will only have half the number of patches. A naive solution would be to pad these to match the number of patches across all shifted windows, but this would introduce unwanted additional computation. A better solution is to merge these half-windows together, using the attention mask to make sure they do not actually attend to each other.

The combination of windowed attention with the shifted windows allows for connections between neighbouring non-overlapping windows, which Liu et al. [2021] found to be very effective. Swin Transformers perform considerably better than regular ViT networks, and their design paradigms likely carry over to other domains. They have also been applied to dense prediction tasks such as object detection and segmentation, where fine-grained output is required.

2.3 Summary

In meta learning, models are not trained on a singular task, but instead learn to learn in general. Neural processes are a particular powerful meta-learning paradigm, which encode a variable length context dataset to enable prediction on a separate target dataset. NP encoders can utilise convolutional networks, which achieve efficient near state-of-the-art performance on spatio-temporal datasets, such as environmental data. A key aspect of such datasets is the abundance of both on-the-grid and off-the-grid data variables, both of which need to be considered for optimal results, which complicates model design.

From other domains such as language processing or computer vision, it is known that CNNs are outperformed by transformer-based architectures. Recent research has also demonstrated the utility of (non TNP) transformer models when applied to environmental data. TNPs attempt to introduce a transformer as the encoder of the NP, but are not feasible to use at scale, due to their quadratic computational complexity. PT-TNPs drastically reduce the number of attention calculations by introducing latent pseudo-tokens, but without exploiting structure in the data this leads to a loss of modelling power. For structured data, efficient transformer architectures exist, such as the ViT or Swin Transformer. The next chapter will focus on the contributions of this thesis, combining both data modalities onto a grid of pseudo-tokens, after which efficient structured-data transformers can be employed.

Chapter 3

Methodology

This chapter will outline the methodologies, procedures and techniques developed in this research, which jointly form a full modelling pipeline. To start, section 3.1 covers the construction or preparation of synthetic and real world datasets respectively. The next two sections elaborate on a TNP model that can intelligently combine both on- and off-the-grid data modalities to predict at arbitrary target locations. Finally section 3.4 illustrates how the architectures can be modified for greater computational efficiency, especially for structured data. Lastly, some ablations are discussed and the pipeline is summarised. A visual summary of the content covered in this chapter is provided in the two schematics in figure 3.1.

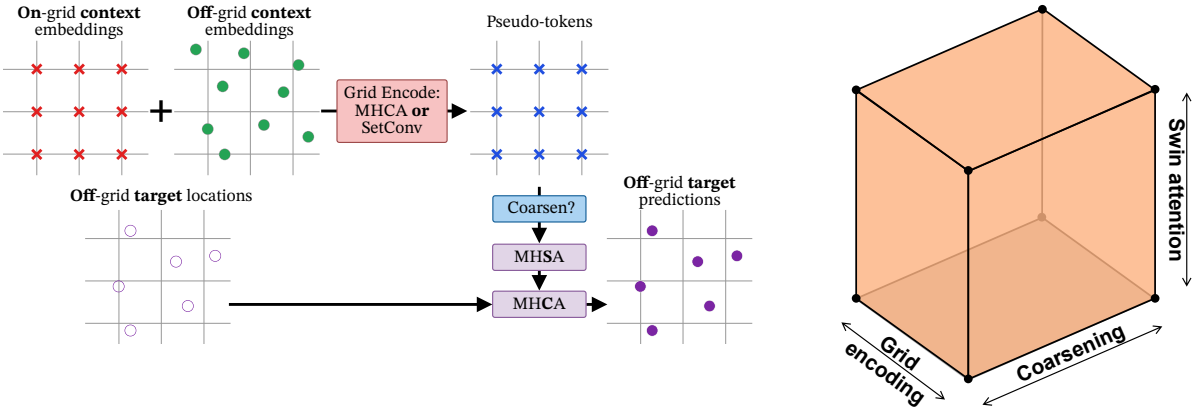


Figure 3.1: Left: Schematic of the overall modelling pipeline discussed in this chapter. Dual modality context data is embedded (section 3.2) and passed through a grid-encoder from section 3.3 to form pseudo-tokens. These pseudo-tokens can optionally be coarsened, after which they self-attend, possibly with shifted windows attention from section 3.4.1. Finally cross-attention with the target inputs is performed to obtain predictions at the target locations. **Right**: Design space of the models that will be explored in this section.

3.1 Data

This section will outline the datasets used in this research in more detail. A large part of this thesis focuses on designing and testing novel architectures, where elaborate testing and finetuning is especially important. It is therefore valuable to at least initially use small-scale datasets, with fine-grained control over the specifics of the dataset. For this reason, section 3.1.1 will focus on the creation of synthetic data, according to the specific characteristics required for this research. This artificial data is used for preliminary experiments, to test and optimise developed techniques at a smaller scale, such that they are ready to be used for larger scale real-world problems.

Next, section 3.1.2 will focus on large scale real-world data, specifically in the context of environmental modelling. The ECMWF Reanalysis v5 (ERA5) [Hersbach et al., 2020] dataset is introduced, which contains environmental measurement data of the real-world. This dataset is used to test the predictive performance of the architectures and methods developed.

Since a major focus area of this research is the combination of structured and unstructured data, datasets will be constructed or pre-processed so that their context sets also feature both modalities of data. While both on- and off-the-grid context data is used, target data will always be off-the-grid. The main reason for this decision is that during evaluation, it makes more sense to test a model at any location as opposed to constricting the evaluation to specific grid locations. Furthermore, if the two modalities come from different datasets, the off-the-grid variable is likely also not available at the grid locations. Each dataset will therefore consist of three collections of data, namely the on-the-grid (context) data, the off-the-grid context data and the (off-the-grid) target data. Each of these collections consists of both input locations x and output values y.

3.1.1 Synthetic GP data

The aim of generating synthetic data [Nikolenko, 2021] is to artificially mimic the challenging aspects of real world environmental data in a controlled setting at smaller scale. The practice of optimising models on artificial data prior to evaluating on real world data is quite common for NPs specifically [Foong et al., 2020; Gordon et al., 2019; Kim et al., 2019]. Since the on- and off-the-grid data should be sufficiently distinct, they are sampled differently, through two different *tasks*. Figure 3.2 illustrates an example of synthetic data sampled from two correlated tasks. Since the data is synthetic, it is also possible to calculate and show the ground-truth distribution, which is visualised through the mean and standard deviation ranges.

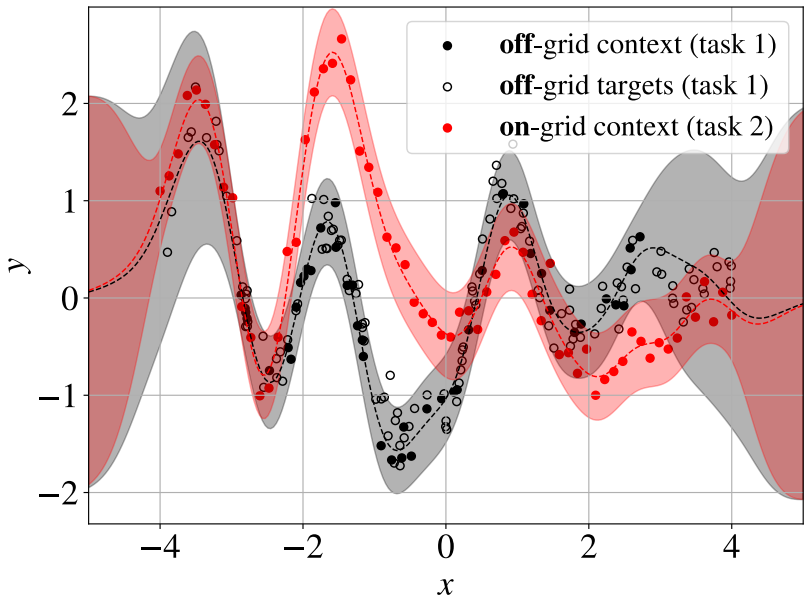


Figure 3.2: Illustration of the synthetic data, sampled from a Hadamard-style two-task Gaussian process. The target and off-the-grid context data originate from the same GP kernel, the on-the-grid data comes from a correlated GP kernel. The tasks have a correlation of 0.85. The ground truth means are also visualized for each task, with the shaded area indicating one standard deviation in both directions for each respective task.

The first step to generating artificial data, is to generate input data. For all off-the-grid data, this is achieved by sampling uniform random data within a configurable range, where the number of off-the-grid context and target points can be arbitrarily chosen. It is common to sample targets from a wider range than the off-the-grid context data, to evaluate a model's ability to extrapolate. For the on-the-grid input data, a grid is constructed between a configurable range, where the number of datapoints per unit of this grid range (*points per unit*) can also be specified. The dimensionality of the input data can be set arbitrarily, but is mostly set to one, adequately named synthetic 1D data.

Given the input data, the outputs are generated through *Gaussian Processes* (GPs) [Rasmussen, 2003; Williams and Rasmussen, 2006]. Given a mean function $\mu(x)$ and kernel function $k(x,x')$, a multivariate Gaussian over a function-space can be defined, which for specific inputs $\{x_i\}_{i=1}^n$ can be written as per equation 3.1.

$$\begin{bmatrix} f(x_1) \\ \vdots \\ f(x_n) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu(x_1) \\ \vdots \\ \mu(x_n) \end{bmatrix}, \begin{bmatrix} k(x_1,x_1) & \dots & k(x_1,x_n) \\ \vdots & \ddots & \vdots \\ k(x_n,x_1) & \dots & k(x_n,x_n) \end{bmatrix} \right) \tag{3.1}$$

Since the kernel function can be chosen arbitrarily, it is common to set the mean function $\mu(x) = 0$, and let the kernel function incorporate its effects without loss of generality [Williams and Rasmussen, 2006]. What remains is to specify the kernel function, with a common choice being the Radial Basis Function (RBF) kernel, sometimes also called the Exponentiated Quadratic (EQ) or Squared Exponential (SE) kernel. Duvenaud [2014] provides an overview of different kernel functions. Environmental data tends to show different trends, often at different length-scales of interest. By varying the length-scale in the kernel function for each batch of synthetic data, it is possible to generate synthetic data which also showcases patterns at different length-scales.

When the goal is to generate synthetic data of different but correlated modalities of data, using a singular GP kernel for both will not suffice. Multitask Gaussian Processes can model several related functions at once [Alvarez et al., 2012; Bonilla et al., 2007; Yu et al., 2005]. Most multitask GPs return an output for every single task, for every input. For our use-case however, this would generate redundant data, since only a single output for every input is necessary. Therefore, a a Hadamard-style kernel [Remes et al., 2017] is used, which is able to generate the desired type of output. By treating each data modality as a different task, this Hadamard multitask GP will generate a single output for every input, such that outputs from the same modality stem from the same GP kernel and outputs from different modalities from a correlated GP kernel. The Hadamard kernel achieves this through a kernel function $k(x,x')$, which expresses the covariance matrix between the tasks. In our use-case there are two modalities of data and therefore two GP tasks are used, where the covariance matrix can now be written as $C = \begin{bmatrix} 1 & c \\ c & 1 \end{bmatrix}$, with c being the configurable task-correlation parameter.
The covariance parameter is varied across experiments, where $c = 0$ indicates that both context modalities are completely independent, and $c = 1$ would mean that the outputs for both context modalities come from the same GP. Finally, after generating outputs from the Hadamard-style GP for all collections of datapoints, a small amount of random noise can be added to these outputs to make the learning problem slightly more realistic.

3.1.2 ECMWF Reanalysis v5 (ERA5) data

In environmental modelling [Eyre et al., 2022; Laloyaux et al., 2016], datasets often concern various sensor measurements, which are spread out across space (for example longitudinal and latitudinal) and are collected at different time-intervals. Common measurements include surface variables such as 2-metre temperature (T2M), eastward and northward components

of windspeed at 10-metres (U10 and V10 respectively) and sea level pressure (MLSP), often consolidated in datasets like the ECMWF Reanalysis v5 (ERA5) dataset [Hersbach et al., 2020]. These datasets can comprise millions of measurements, motivating approaches that are computationally efficient.

The ERA5 dataset contains hourly measurements from 1940 to the present day, for a large number of atmospheric and surface variables. It is created through a process known as data assimilation, where actual measurements are combined with predictions from numerical weather prediction (NWP) centres. These NWP centres simulate compute-heavy physics equations, such as the Navier-Stokes equation [Kimura, 2002], at high fidelity. The ERA5 dataset is globally complete with no missing values, and it has been gridded to contain measurements at exactly 0.25°. This means that the grid consists of 720 by 1440 measurements in latitudinal and longitudinal direction respectively, therefore containing just over a million measurements at each time instance.

Selecting ERA5 variables to model. For computational reasons we chose to model one single variable at a time for each modality. For the on-the-grid data, the *skin* temperature at the surface of the earth (SKT) is used, whereas the off-the-grid data is chosen to be the temperature two meters above the surface (T2M). The skin temperature is also a structured-data

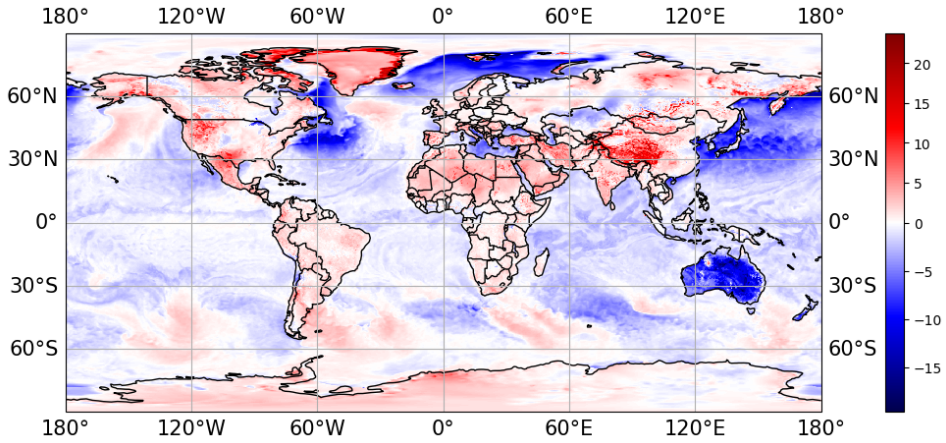


Figure 3.3: Global difference between measurements from ERA5 two meter temperature and skin temperature on 1st January 2019 at midnight.

measurement in the real-world, whereas ERA5 two meter temperature is (mostly) derived from operational NWP models, based on a multitude of other environmental variables. Our models perform a comparable task, where they attempt to derive two meter temperature provided correlated skin temperature. These two variables are highly correlated, for example having

an averaged Spearman correlation of 0.9931 over all of January 2019. Nonetheless individual measurements show enough differences such that both modalities are sufficiently distinct, as illustrated by figure 3.3.

To form the off-the-grid context and target data, distinct two meter measurements are sampled at arbitrary locations. Due to computational limitations, the skin temperature grid is downsampled to a lower resolution to form the on-the-grid data, by employing average pooling interpolation. Vaughan et al. [2024] similarly coarsen the grid to 128×256, which corresponds to grid regions spanning $1.41°$. Instead, this research opts for a 120×240 resolution corresponding to $1.5°$, since that makes the original resolution an exact multiple of the coarsened resolution.

Spherical embeddings. Finally, it remains to consider what happens at the edges of the grid imposed by the ERA5 dataset, which occur at the international dateline. The longitude spans the orthogonal map direction from America to Asia and has a value between -180 and 180, wrapping around so that both ± 180 are the same location. Therefore the first step our models take when processing ERA5 data, is to encode the coordinate values through a combination of Spherical Harmonics and Sinusoidal representation networks (SirenNet) [Rußwurm et al., 2024]. This is achieved through a single learnable MLP layer, which uses a fixed number of spherical harmonic functions to encode the latitude and longitude values. Each individual spherical harmonic function can be calculated with an associated *Legendre polynomial* [Lima, 2022], which in turn utilises a sinusoidal function to form an orthogonal basis function. For ERA5 data, 10 Legendre polynomials are used, as a SirenNet with that number of polynomials was shown by Rußwurm et al. [2024] to sufficiently model spherical data, but the exact number required will depend on the application.

3.2 Regular TNP

Before explaining various modifications to the TNP architecture, this section will outline the practical base implementation of the TNP for on- and off-the-grid data. To formally define the input to the TNP, some notation is required. Let C_u denote the *u*nstructured off-the-grid context data, comprising of a tuple $(X_u^{(c)}, Y_u^{(c)})$. In a similar fashion, $C_s = (X_s^{(c)}, Y_s^{(c)})$ is the *s*tructured on-the-grid data, such that the entire context dataset D_c is defined as $D_c = C_u \cup C_s$. Note that $|C_u|$ will indicate the length of the off-the-grid context set, defined as $|C_u| = |X_u^{(c)}| = |Y_u^{(c)}|$. The first element of the off-the-grid context data tuple can be defined as a collection $X_u^{(c)} = \{x_{ui}^{(c)}\}_{i=1}^{N_u^{(c)}}$, where the other elements can be defined analogously. The target data, which is always off-the-grid as per section 3.1, is defined using $D_t = (X^{(t)}, Y^{(t)}) = \{x_i^{(t)}, y_i^{(t)}\}_{i=1}^{N^{(t)}}$. This leads to three

collections of data, specifically the on-the-grid context data, off-the-grid context data and target data. Note that all X and Y are observed by the model, except for the target outputs $Y^{(t)}$.

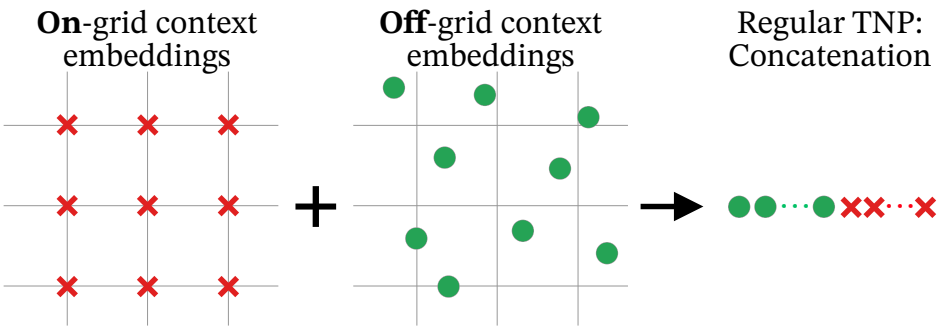


Figure 3.4: Schematic of the treatment of both context data modalities inside a regular TNP. Note that input locations $x^{(c)}$ are two dimensional in this example, with the circles and crosses representing embeddings at those locations.

The encoding for a TNP processing dual modality data is different from the foundation outlined in section 2.2, and will therefore be specified below.

1. Some additional information is first encoded into the data, which is achieved through adding two extra boolean flag dimensions to all output data y. The first of these dimensions encodes whether the data is context (0) or target data (1). The second dimension indicates if those data are on- or off-the-grid. Not having these extra dimensions would almost certainly hurt performance, as a model could for example no longer distinguish between actual context set outputs and placeholder target set outputs (consisting of a zero vector).

2. Secondly, the input x and output y (with flag dimensions) for each of the three collections of data are stacked on top of each other, so that a single multidimensional array (multi-array) is formed for each collection.

3. Next, each of the three multi-arrays are pointwise embedded separately, but using the same learnable multi-layer perceptron network. The embedding function is denoted as $h(\cdot,\cdot) : \mathbb{R}^{D_x+D_y+2} \to \mathbb{R}^{D_e}$. This produces embeddings $e_{uj}^{(c)} = h(x_{uj}^{(c)}, y_{uj}^{(c)}) \in E_u^{(c)}$ and $e_{sj}^{(c)} = h(x_{sj}^{(c)}, y_{sj}^{(c)}) \in E_s^{(c)}$ and finally $e_j^{(t)} = h(x_j^{(t)}, y_j^{(t)}) \in E^{(t)}$. The off-the-grid context, on-the-grid context and target embedding spaces are denoted with $E_u^{(c)}$, $E_s^{(c)}$ and $E^{(t)}$ respectively.

4. Finally, specifically for the regular TNP, both modalities of context data, C_u and C_s, are concatenated into a single multi-array with $N_u^{(c)} + N_s^{(c)}$ datapoints per batch, as

per figure 3.4. The next section will replace this concatenation with more intelligent approaches to combine both modalities of context data.

The attention layers and the decoder inside the regular TNP follow the definitions outlined in section 2.2 and illustrated in figure 2.2 exactly. Note that a standard attention layer expects the input to be a collection of datapoints, such that if any grid structure were originally present, it would have to be flattened first. The next sections will alter different aspects of the TNP described in this section. As the regular TNP does not modify the input data at all, nor does it reduce the number of attention calculations, it will not be computationally feasible at scale, but will likely achieve the best model performance for smaller experiments.

3.3 Combining both modalities as a grid of pseudo-tokens

The regular TNP from the previous section combines on- and off-the grid data naively, by treating both modalities as a collection of off-the-grid datapoints. It is be possible to decrease the computational resources this requires, but from section 2.2.3 it is evident that such an approach would have to ultimately trade computational efficiency, achieved by having a low number of pseudo-tokens, with quantitative performance.

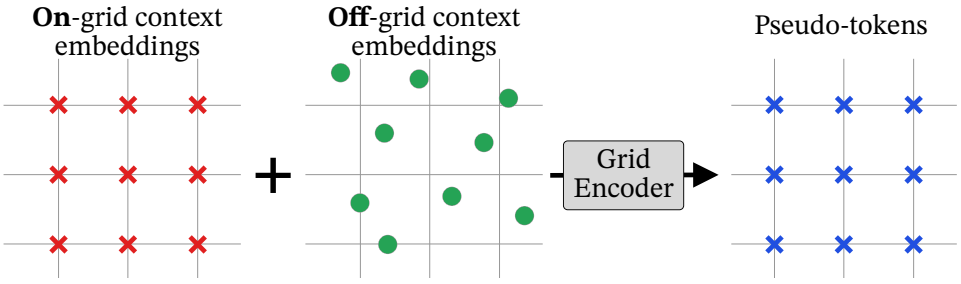


Figure 3.5: The treatment of both context data modalities inside a TNP with either or MHCA or SetConv grid-encoder. Note that input locations $x^{(c)}$ are two dimensional in this example, with the circles and crosses representing embeddings at those locations. The pseudo-grid depicted here has the same resolution as the on-the-grid data, but this is not a requirement.

Therefore the methodology developed in this research will instead aim to map both data modalities to a grid of pseudo-tokens, as per figure 3.5. Most experiments in this research introduce this new step after both modalities of input data have been embedded, replacing the simple concatenation. However, it is also possible to join both modalities prior to embedding. Once all data has been transformed to this pseudo-grid, structured data transformers can be applied to reduce the computational bottleneck compared to the quadratic attention present in regular TNPs. Both grid-encoders are identical in terms of output shape and dimensionality,

such that the downstream TNP is agnostic to which grid-encoder is used. Two distinct *grid-encoders* are explored. Section 3.3.1 illustrates an approach inspired by the treatment of off-the-grid data in Gordon et al. [2019]. The second approach detailed in section 3.3.2 utilises Multi-Head Cross-Attention instead.

3.3.1 SetConv Encoder

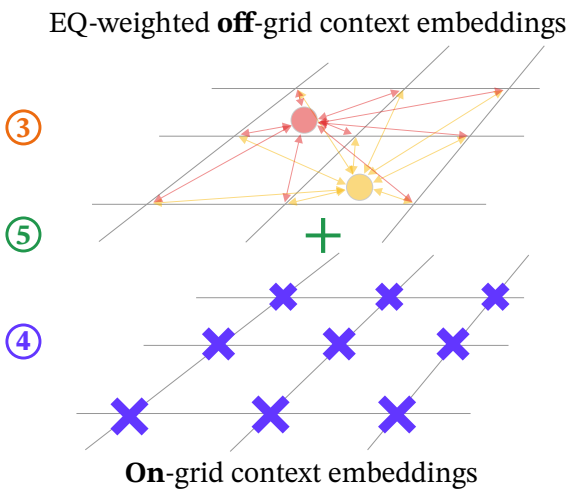
This approach is inspired by the function-space embedding in ConvCNPs [Gordon et al., 2019] and is therefore named the *SetConv Encoder*. ConvCNPs have a convolutional neural network as a backbone, meaning they necessarily need their input to be on-the-grid. Nonetheless, ConvCNPs can also model off-the-grid data, through a transformation of the datapoints, as outlined in section 2.1.2. By default ConvCNPs create a dummy grid of uniform tokens $\{t_i\}_{i=1}^{T}$ onto which the off-the-grid data is mapped. A key element of the SetConv Encoder is to instead use the locations of the on-the-grid data modality to form this uniform grid.

The following relies heavily on the specific notation for on- and off-the-grid data introduced in section 3.2. The on-the-grid data, more specifically the locations $X_s^{(c)}$, will serve as the tokens that form the grid locations of the final grid of pseudo-tokens. This decision has the advantage that now the embeddings for the structured data $E_s^{(c)}$ do not need to be transformed, since they are already present at the correct locations. The next step is to transform the off-the-grid points, specifically the embeddings $E_u^{(c)}$, to the grid locations $X_s^{(c)}$. This transformation is based on a weighting between every possible off-the-grid tuple $(x_{uj}^{(c)}, e_{uj}^{(c)})$ and every grid-location $x_{si}^{(c)}$. Provided a distance function $d(\cdot,\cdot) : X \times X \mapsto \mathbb{R}$, this weight is calculated through an EQ-basis function $\psi(\cdot,\cdot) : X \times X \mapsto \mathbb{R}$ as per equation 3.2, where l are learnable length-scales, one for each dimension of the grid locations.

$$\psi(x_{si}^{(c)},\, x_{uj}^{(c)}) = \exp\left(-\frac{d\left(x_{si}^{(c)}, x_{uj}^{(c)}\right)}{2l^2}\right) \qquad d\left(x_{si}^{(c)}, x_{uj}^{(c)}\right) = \left(x_{si}^{(c)} - x_{uj}^{(c)}\right)^2 \tag{3.2}$$

It is important to note that for spherical data such as ERA5, the more complex Haversine distance from equation 3.3 [Sinnott, 1984] is used instead.

$$d(x_{si}^{(c)}, x_{uj}^{(c)}) = 2\sin^{-1}\sqrt{\sin^2\left(\frac{[x_{si}^{(c)} - x_{uj}^{(c)}]_{lat}}{2}\right) + \cos\left([x_{si}^{(c)}]_{lat}\right)\cos\left([x_{uj}^{(c)}]_{lat}\right)\sin^2\left(\frac{[x_{si}^{(c)} - x_{uj}^{(c)}]_{lon}}{2}\right)} \tag{3.3}$$



Require: $\psi(\cdot,\cdot)$ and $h(\cdot,\cdot)$
Require: Off-grid context $(x_{uj}^{(c)}, y_{uj}^{(c)}) \in C_u$
Require: On-grid context $(x_{si}^{(c)}, y_{si}^{(c)}) \in C_s$

$\{p_i\}_{i=1}^{N_s^{(c)}} \leftarrow 0$
for $i \leftarrow 1$ to $N_s^{(c)}$ **do**
  $\hat{e}_{si}^{(c)} \leftarrow \sum_{j=1}^{N_u^{(c)}} \psi(x_{si}^{(c)}, x_{uj}^{(c)}) \cdot h(x_{uj}^{(c)}, y_{uj}^{(c)})$
  $e_{si}^{(c)} \leftarrow h(x_{si}^{(c)}, y_{si}^{(c)})$
  $p_i \leftarrow e_{si}^{(c)} + \hat{e}_{si}^{(c)}$
end for
return $\{p_i\}_{i=1}^{N_s^{(c)}}$

Figure 3.6: Schematic (**left**) and pseudo-code **(right)** for the SetConv encoder. Note that input locations $x^{(c)}$ are two dimensional in this example, with the circles and crosses representing embeddings at those locations. The distances between the locations of the off-the-grid points and the grid itself are passed through EQ-basis functions to weight the off-the-grid points, such that gridded pseudo-embeddings $\hat{e}_{si}^{(c)}$ can be obtained.

Through this weighting, the off-the-grid points C_u can be mapped to grid locations to form pseudo-embeddings $\hat{e}_{si}^{(c)}$ as detailed in equation 3.4.

$$\hat{e}_{si}^{(c)} = \sum_{j=1}^{N_u^{(c)}} \psi(x_{si}^{(c)}, x_{uj}^{(c)}) \cdot e_{uj}^{(c)}, \qquad x_{si}^{(c)} \in X_s^{(c)}, \quad x_{uj}^{(c)} \in X_u^{(c)}, \quad e_{uj}^{(c)} \in E_u^{(c)} \tag{3.4}$$

Because of the previous decision to use the on-the-grid locations to form the pseudo-grid, we now have both an on-the-grid embedding and a pseudo-embedding (from transformed off-the-grid points) for every on-the-grid location. The embeddings $e_{si}^{(c)}$ and pseudo-embeddings $\hat{e}_{si}^{(c)}$ are simply summed for every input location to obtain pseudo-tokens p_i.

The entire SetConv Encoder is outlined for two off-the-grid points in figure 3.6. This approach is computationally efficient yet powerful due to the flexibility of the basis functions with a learnable length-scale. The downside is that the weighted summation of off-the-grid points can lead to unwanted smoothing of local regions that originally had a lot of information in their off-the-grid points. This motivates an approach which is slightly more complex but does not suffer from this local blurring.

3.3.2 MHCA Encoder

The base MHCA encoder. The second approach consists of performing multi-head cross-attention (MHCA) between all datapoints and a learnable grid of latent embeddings, to output a grid of pseudo-tokens. A visual summary is given in figure 3.7. Adopting the same notation as for the SetConv approach, $(X_u^{(c)}, E_u^{(c)}) = \{x_{ui}^{(c)}, e_{ui}^{(c)}\}_{i=1}^{N_u^{(c)}}$ denotes the locations and embeddings of off-the-grid points respectively and $(X_s^{(c)}, E_s^{(c)})$ is analogously defined for the on-the-grid points.

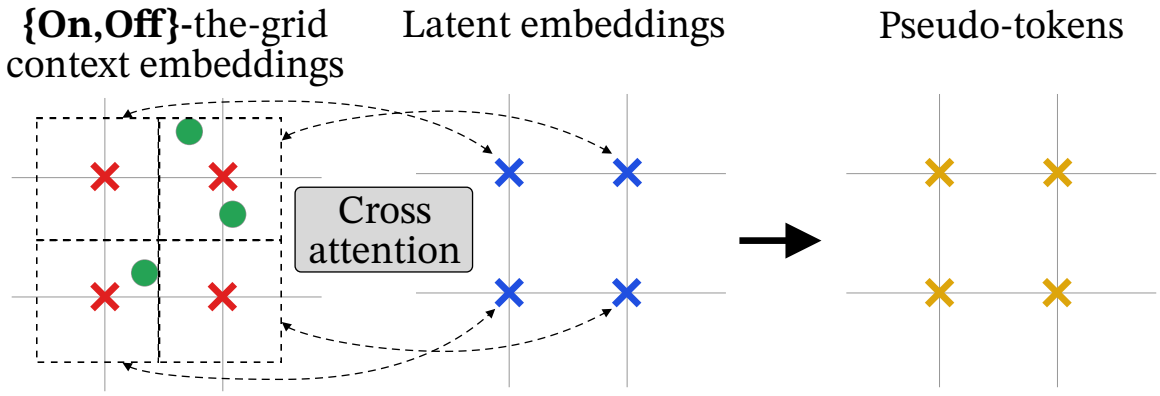


Figure 3.7: Schematic of the MHCA encoder, dotted lines are used to denote which datapoints and latent embeddings cross-attend. Context data is already embedded, but their locations and nearest neighbours are represented/calculated in two-dimensional x-space.

The grid-encoder features a learnable latent embedding of shape $(N_s^{(c)}, 1, E)$. Each latent embedding will therefore focus on a singular on-the-grid point, and all nearest off-the-grid datapoints. To find the nearest off-the-grid points, the distance is calculated with respect to the input locations $X_u^{(c)}$ and $X_s^{(c)}$ using the computationally efficient Manhattan distance. Note that since only the minimum distance is used for deciding assignment of nearest neighbours, any monotonic distance function would yield the same result and therefore a Haversine distance is never required. Each pseudo-token is calculated through attention between the corresponding latent embedding on the one hand, and the combination of the (embedded) on-the-grid point and nearest off-the-grid points on the other hand.

Efficient batching. The key strategy to implement this approach efficiently, is to transform each batch of dual modality data to $N_s^{(c)}$ pseudo-batches, such that there is a pseudo-batch for every on-the-grid point for every original batch. Each pseudo-batch b_i will store both the on-the-grid embedding e_{si} as well as the embeddings of any off-the-grid points that are closest to this particular on-the-grid point. Equation 3.5 denotes the selection of tokens for the pseudo-batch b_i more formally, where E denotes the embedding dimension.

$$b_i = \left\{ e_q : e_q = e_{si} \vee (\forall j) \left[e_q = e_{uj} \wedge x_{si} = \underset{x_s}{\operatorname{argmin}} \sum_{k=1}^{E} \left| [x_s]_k - [x_{uj}]_k \right| \right] \right\} \tag{3.5}$$

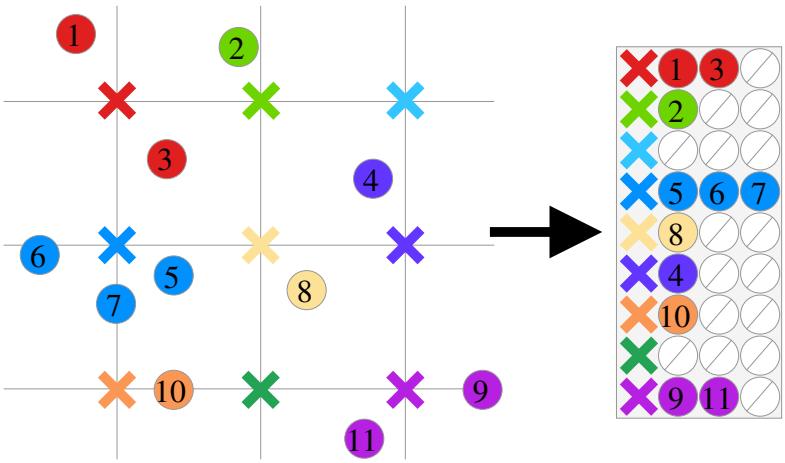


Figure 3.8: Illustration of nearest neighbour batching in the MHCA Encoder. Crosses (on-the-grid) and circles (off-the-grid) denote embedded datapoints, but their locations and distances are represented in two-dimensional x-space. Colour is used to denote the nearest neighbour, fake padding embeddings are coloured white.

The pseudo-batches will necessarily be of different length, with the maximum length dictated by the on-the-grid point with the most neighbours. Each pseudo-batch is therefore padded to be of this maximum length by appending (multiples of) the fake embedding $e_\emptyset$. This formation of pseudo-batches for a single original batch is outlined in figure 3.8.

Through the procedure outlined thus far, a stacked batch multi-array of shape $(B \cdot N_s^{(c)}, P, E)$ is created, where B is the original batch size and P is equal to the maximum number of nearest neighbours plus one. Although the reduction is not guaranteed, for most datasets $P << N_s^{(c)}$, provided the off-the-grid points are somewhat distributed across the grid.

The grid-encoder's latent embedding is repeated B times, so that the artificial batch shape matches that of the nearest-neighbour multi-array. The output of the MHCA-layer is reshaped to $(B, N_s^{(c)}, E)$, to have an embedded representation at every on-the-grid location containing information of both original modalities of data. An attention mask is used to ensure that no attention is calculated with fake embeddings $e_\emptyset$.

Scalability of the MHCA Encoder. This attention-based approach does not suffer from the quadratic computational bottleneck that is commonplace for TNPs. In the nearest-neighbour

multi-array, the dependency on the number of on-the-grid points is now placed in the batch dimension, as opposed to the dimension with which attention is performed. As such, the attention weight matrix is of shape P by $N_s^{(c)}$, which is notably not quadratic in the number of datapoints. The number of pseudo-batches will be a multiple of the number of on-the-grid datapoints, but batched calculations can be performed highly efficiently in parallel. As such, scaling the grid fidelity or the number of datapoints no longer quadratically increases the computation required in the attention layer.

The MHCA Encoder approach is computationally more intensive compared to the SetConv Encoder, but also has far greater flexibility, featuring of lot of extra learnable parameters, both in the layer itself as well as in the latent embeddings. If the off-the-grid data is distributed very unevenly, it is also possible to bottleneck its computation for efficiency by limiting the maximum allowed number of neighbours P.

3.4 Structured data transformers

The previous sections have outlined how the combination of on- and off-the-grid context data can be preprocessed, embedded, and finally jointly encoded to form a grid of pseudo-tokens, either through the SetConv or MHCA Encoder. Now the focus will be shifted towards realising more efficient transformer architectures, replacing the vanilla transformer inside the regular TNP from section 3.2. This is possible since after joining both modalities, the structure in on-the-grid pseudo-tokens can be exploited. See figure 3.9 for a summary of the data pipeline including grid-encoding, coarsening, shifted windows self-attention.

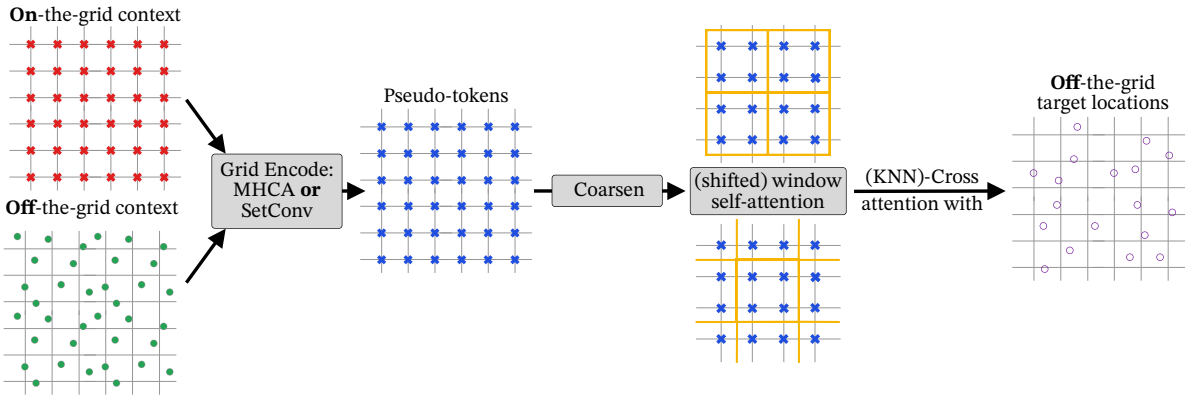


Figure 3.9: Schematic of modifications to the TNP for efficient structured data transformers. Datapoints are already embedded, but their locations are represented in two-dimensional x-space. Note that coarsening is optional, and also possible within the grid-encoder itself.

A simple approach to drastically reduce the number of pseudo-tokens is to employ patch embeddings as first introduced in the ViT [Dosovitskiy et al., 2020], before passing the pseudo-grid to the TNP. The formal workings of this mechanism have been outlined in section 2.2.4. To achieve patch embedding for grids of n dimensions, a simple n-dimensional convolutional neural layer is used. To avoid confusion the name ViT will not be employed, but instead any model using patch embeddings will be denoted as *Patch Coarsening* (PC) or *Patch Embedding*. Patch coarsening can then be followed by either a standard TNP or a more efficient further architecture. Note that the above explanation places the patch embedding just before the TNP, but it is also possible to coarsen the on-the-grid data before encoding both modalities of data to pseudo-tokens, which can be advantageous if the computation is bottlenecked by the grid-encoder.

3.4.1 Shifted windows self-attention

The next improvement stems from actually reforming the TNP architecture. A more involved architecture for processing large scale grid-structured data is the Shifted Windows Transformer (Swin) [Liu et al., 2021], as elaborated in section 2.2.4. We have implemented shifted windows self-attention from scratch so that it can be used for grids of an arbitrary number of dimensions. As per the original paper, the half-windows that originate from shifting are fused together for efficiency, where masking can be employed to prevent unrelated windows from attending. For ERA5, special care has been taken to only employ this masking along the latitudinal direction, such that along the longitudinal direction patches that are partially shifted across the date line will still fully attend with both sides of this date line. Note that Swin's hierarchical patch merging is left for future research as per section 2.2.4. As such, any reference of Swin after this section will refer to our custom implementation based on pure shifted windows attention.

3.4.2 Top-K nearest neighbours cross-attention

After replacing the self-attention layers in the regular TNP with shifted windows self-attention, a major bottleneck that remains are the cross-attention layers. Windowed cross-attention is not directly possible, since the target data is off-the-grid and therefore unstructured. To improve the efficiency of the cross-attention layer, a novel *top-k nearest neighbours cross-attention* concept is introduced, partially making use of the ideas from the MHCA Encoder from section 3.3.2. This form of cross-attention is not exclusive to shifted windows self-attention, but can also be used in conjunction with regular self-attention layers.

This new cross-attention selects the K nearest context output vectors for each embedded target vector, where distance is determined in terms of location on the grid. Similar to the MHCA encoder, all target datapoints are now stacked in the batch dimension to form pseudo-batches, creating a target multi-array of $(B*N^{(t)},1,E)$, where B is the original batch size and E the embedding dimension. A nearest-neighbours multi-array of shape $(B*N^{(t)},K,E)$ is again constructed for the context data, which has been passed through a self-attention layer previously. The nearest-neighbour multi-array and the target multi-array perform cross-attention, which now only has to calculate attention between each target datapoint and the K nearest context data outputs. In a similar fashion to the MHCA Encoder from section 3.3.2, this allows for efficient batched GPU computation of nearest neighbour based attention. The value of K controls how many context data output values influence the attention calculation for each target value, where for a 2D grid $K = 9$ implies that the target point attends to a 3×3 region in the context grid. For 2D grids in general, only squares of odd numbers are chosen for K, as that allows our implementation to exploit the grid structure to find neighbours, instead of performing distance calculations. Although this cross-attention could lead to decreased accuracy if all context data were important for each target datapoint, it can significantly increase computational efficiency and potentially induces a useful bias towards local information.

3.5 Ablations

Having introduced the grid-encoders used to transform the dual modality data into a pseudo-grid, and the efficient transformer architectures that can benefit from this grid, the main techniques developed in this thesis have been outlined. Some experiments however will deviate from the pipeline as outlined previously and this section will outline some more significant architecture modifications required for certain ablations.

3.5.1 Coarsening through the grid encoder

Both grid-encoders outlined previously join the dual modality data to a pseudo-grid which has identical dimensions to the on-the-grid input data. A possible ablation is to instead use the grid encoder to coarsen the grid, such that the pseudo-grid is of a smaller resolution than the on-the-grid data. This can be achieved by having the grid-encoder construct an internal grid of a smaller fixed dimensionality (and in the case of the MHCA encoder, also a latent grid of this dimensionality). Both incoming data modalities are then concatenated as a temporary "off-the-grid" joint collection. The grid-encoder's functionality itself remains unchanged, treating

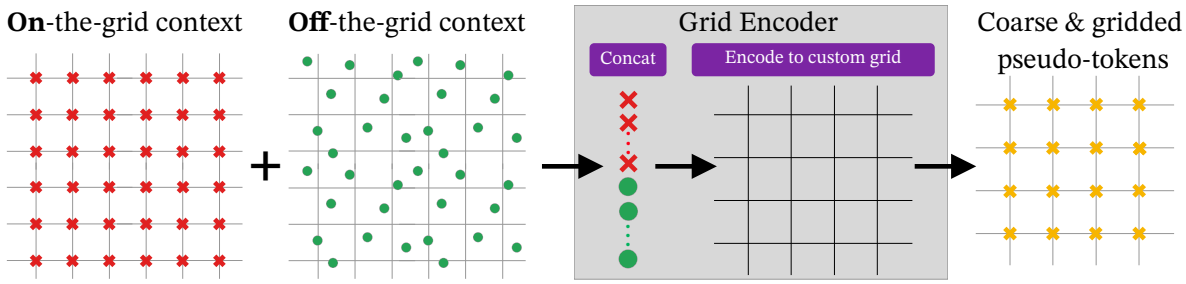


Figure 3.10: Schematic of coarsening through the grid-encoder. Datapoints are already embedded, but their locations are represented in two-dimensional x-space. Note that for the MHCA grid encoder, the encoder's internal grid also contains a latent embedding at every grid location.

the joint data collection as its off-the-grid input and the smaller internal grid (and latent grid if applicable) as its on-the-grid input. This ablation could therefore possibly replace the patch embedding by having the grid-encoder also perform the coarsening.

3.5.2 Using either data modality separately

It is important to validate that the models actually benefit from the combination of both modalities of data. To ensure this, an experiment can be repeated ceteris paribus, where the model only has access to a singular modality of data. If only on-the-grid data were used, the grid encoder could be omitted, provided the grid output resolution was left unchanged. This is possible because without grid coarsening, neither grid encoder would meaningfully modify the on-the-grid data. The MHCA Encoder would perform isolated cross-attention, but since this does not actually lead to any recombination of data, it is just another linear projection, that can already be captured by the embedding of the datapoints.

For modelling purely off-the-grid data, or in the scenario where the single modality on-the-grid data is encoded to a smaller grid and therefore treated as unstructured, a comparison to dual modality data is slightly more involved. For the SetConv Encoder from section 3.3.1, this can be achieved by not summing the pseudo-embeddings $\hat{e}_{si}^{(c)}$ of the unstructured data with the on-the-grid embeddings $e_{si}^{(c)}$. For section 3.3.2's MHCA encoder a single learnable fake embedding vector is provided, which is repeated into the nearest-neighbour tensor instead of the on-the-grid datapoints. These modifications allow a model to be trained that processes the off-the-grid data in a identical manner, but does not have access to the on-the-grid data.

3.6 Summary

The previous sections have gradually built a full procedure to model the combination of on- and off-the-grid data, after having introduced this dual modality data in section 3.1, where both synthetic and real environmental data are discussed. The base TNP architecture is described in section 3.2, whereas the next section elaborates how to join both modalities of data into a pseudo-grid. Section 3.4 explains how the regular TNP can be improved to adequately benefit from this structured pseudo-grid, with the penultimate section of this chapter detailing some more possible variations of this model pipeline. Some of the possible modelling decisions are summarised in figure 3.11.

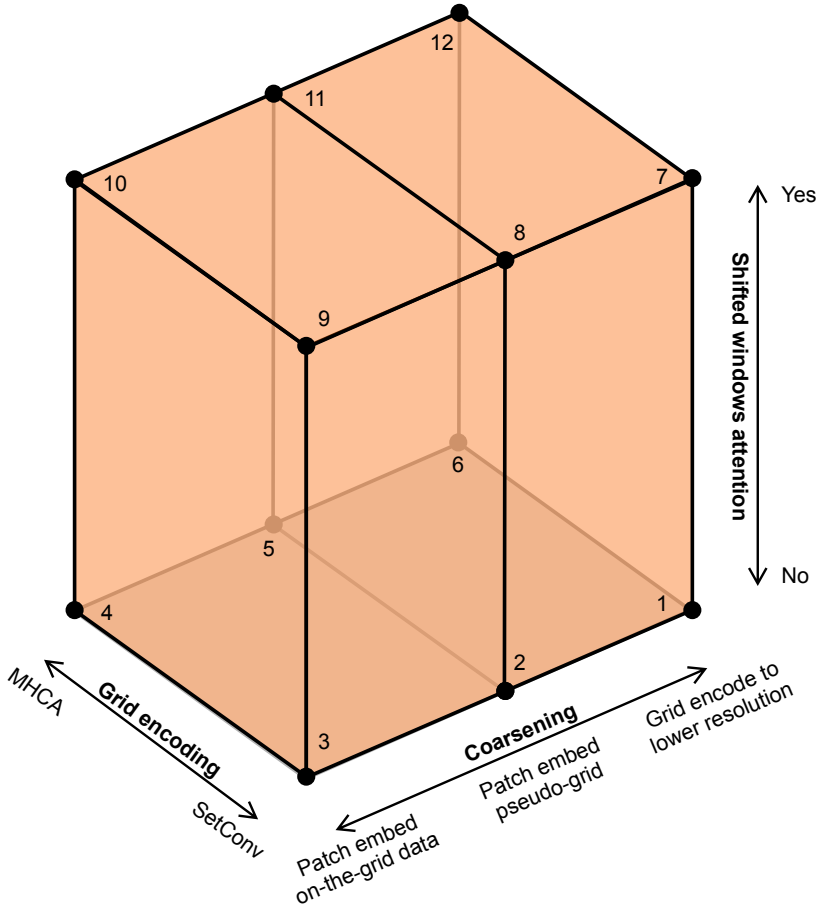


Figure 3.11: Illustration of some of the possible modifications to the regular TNP. All permutations of encoding dual modality data to a pseudo-grid of coarsened resolution are represented, with each number representing a possible experiment.

To distinguish the modifications of the TNP discussed in this chapter, some naming conventions need to be introduced for the next chapter. "PC" will be used to denote patch coarsening and the placement in a model's name indicates if this is applied before or after grid encoding.

"GC" indicates coarsening through the grid encoder. A model's name is suffixed by "Swin" or "TNP" depending on whether shifted windows self-attention or full self-attention is used. The next chapter will focus on showcasing experimental results, building from synthetic data to real-world experiments with various configurations.

Chapter 4

Experiments

This chapter will cover the experiments that examine and evaluate the various techniques and methodologies that were developed in the previous chapter. The content of this chapter is split with regard to the type of data that is being modelled.

The first half of the chapter (section 4.1) focusses on initial experiments on synthetic data, which allows us to explore the novel techniques in a controlled environment on a smaller scale, with experiments taking at most a few hours each. Firstly, the difficulty of the synthetic data task itself is assessed, through training regular TNPs on different combinations of context data. Next, the grid encoders and some structured data transformer architectures are evaluated to ensure their addition is not detrimental to performance. It is likely that performance differences between the new methods will likely be more or exclusively visible on the large scale real-world experiments, due to the simplicity of the synthetic data task.

The second half of this chapter (section 4.2) regards actual large scale real-world experiments conducted on ERA5 data, which can take up to one and a half days to train. With this highly complex data, we firstly aim to discover which grid-encoding and coarsening techniques work best at scale. Other experiments test the effect of full versus KNN cross-attention, as well the difference between full and shifted windows self-attention. Lastly, the computational speed of different architectures is evaluated. By combining results from these subsections, it should be possible to decide on a final model, which can then be compared to baselines such as the LBANP [Feng et al., 2022]. These experiments can therefore show the effectiveness and computational efficiency of the new models.

An important metric to evaluate is the validation log-likelihood, since the models output a distribution and the likelihood encompasses both the quality of its mean and uncertainty predictions. For the larger real-world dataset the training loss and validation Root Mean Squared

Error (RMSE) are also important. Since RMSE allows for a comparison with deterministic baselines, it is shown instead of log-likelihood for ERA5 experiments.

Shared experimental setting. All experiments are conducted on a singular Nvidia A100-SXM-80GB GPU, with 32 CPU cores. For consistency, some hyperparameters are shared across all experiments. Each model is trained for 200 epochs with a batch size of 16, with the number of batches per epoch depending on the dataset. The AdamW optimizer [Loshchilov and Hutter, 2019] is used with a learning rate of $5 * 10^{-4}$. The TNP encoder always outputs a 128 dimensional vector. The transformer inside the TNP consists of 10 attention layers in total, with self- and cross-attention layers alternating. Each transformer layer has 8 different heads. Each model is trained by optimising the negative log-likelihood of the true target values given the target distribution outputted by the model. The validation set is a quarter of the size of the training dataset, such that it is 20% of the total dataset size.

4.1 Synthetic GP data

This section will cover experiments regarding artificial datasets constructed using (correlated) Gaussian Processes, as outlined in section 3.1.1. Except for section 4.1.3, all experiments are conducted with one-dimensional data. The off-the-grid context data consists of anywhere between 1 and 64 datapoints sampled uniformly from $[-2.0, 2.0]$, whose corresponding output values come from an EQ-kernel with a length-scale that is randomly sampled per batch. The target data consists of 128 datapoints sampled uniformly from a larger range of $[-4.0, 4.0]$, with output coming from the same kernel as the off-the-grid context data. Unless stated otherwise, the on-the-grid data is sampled from the larger $[-4.0, 4.0]$ range, with 8 datapoints per unit, meaning there are 64 equidistant on-the-grid inputs. The output for these on-the-grid points comes from a different Hadamard-style task from the same EQ-kernel, where the task-correlation is 0.8. Each epoch consists of $16,000$ samples for training and $4,000$ samples for evaluation.

4.1.1 Examining the learning problem itself

Before exploring the new procedures, it is valuable to evaluate the quality of the synthetic data learning task itself. Therefore a control experiment is conducted with a regular TNP without any grid-encoding (i.e. both context modalities are concatenated), with varying correlations

between the two tasks generating the on- and off-the-grid data. Experiments are also conducted where either modality is not passed to the model at all.

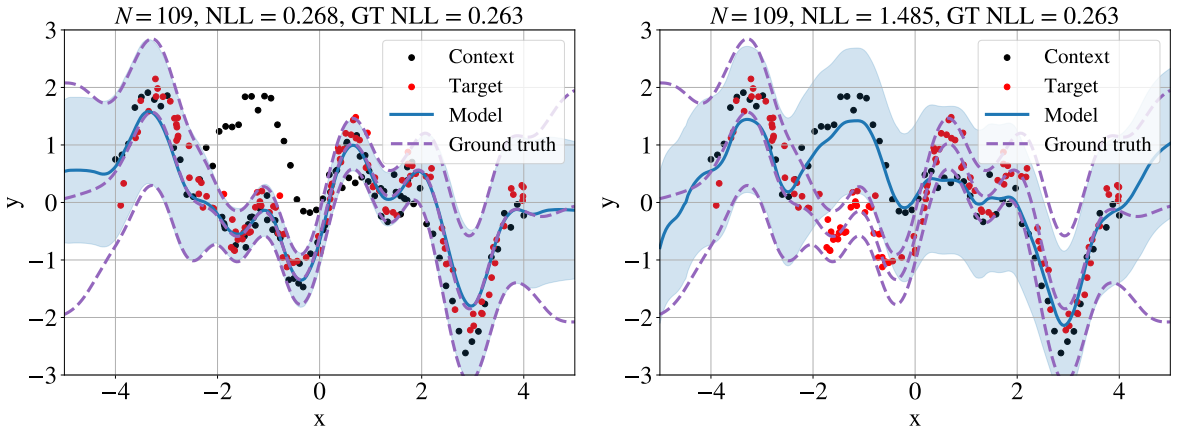


Figure 4.1: Qualitative illustrations of Regular TNPs predicted mean and one standard deviation for an off-the-grid target set, including ground truth calculated from synthetic data. The number of context points, negative log-likelihood (NLL) and ground truth NLL are also shown. **Left:** TNP trained on both context modalities with a correlation of 0.8. **Right:** TNP trained on only on-the-grid context data.

Figure 4.1 illustrates the 1D synthetic data, as well as the model's predicted distribution. Since the dataset is synthetic, it is also possible to calculate the ground-truth distribution of the target data. Both Gaussian distributions are visualised by showing their means and one standard deviation in either direction. Note that both modalities of context data are plotted as identical dots, with their number being denoted as N.

The left plot highlights how a model trained on both modalities of context data can adequately learn to model the target data distribution, as the model's predictive distribution closely follows the target ground-truth distribution in regions where data is observed. The right plot shows a prediction of a TNP trained solely on on-the-grid data, which stems from a correlated but different GP than the target data. Its predictive mean follows the context data as opposed to the target data. Since it does not have access to the variance in the true data generation process, it has to model the uncertainty induced by the largest difference between the context and target data at every location.

Figure 4.2 illustrates the validation log-likelihood for the first set of experiments. By comparing the three blue graphs on the left, one can see how increasing the correlation of the-on-grid context data with the off-the-grid-data leads to better validation performance. To ensure that the model learns from the on-the-grid data at all, an experiment is conducted where a model is only provided off-the-grid data. By comparing the light blue and green curves, it is shown how this experiment performs similar to having uncorrelated on-the-grid data. So the model still

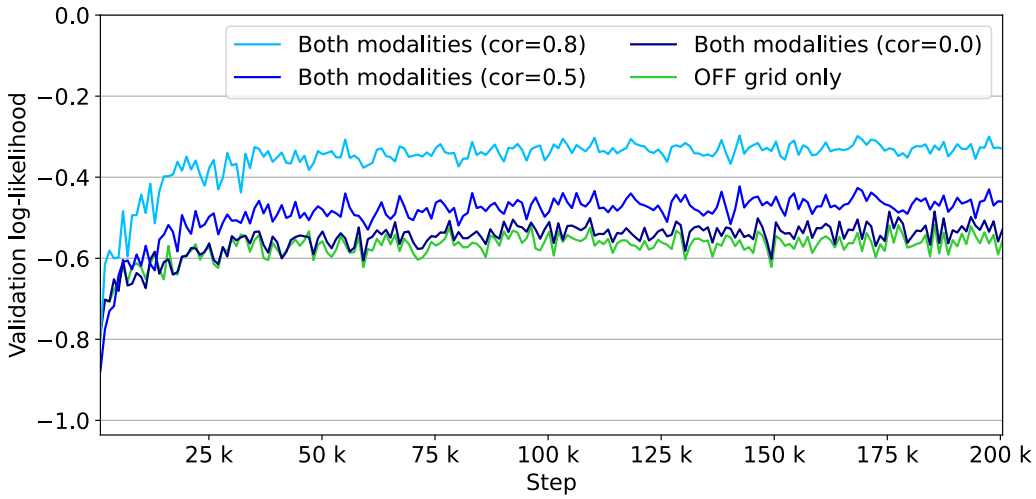


Figure 4.2: Validation log-likelihood curves for regular TNP models without a grid-encoder, trained on synthetic 1D Gaussian process data. The plot shows that having access to more correlated on-the-grid data tends to improve performance, with only being provided off-the-grid data performing the worst.

learns additional information from having access to the correlated on-the-grid data, despite it being sampled from a different process than the target data.

Secondly, two experiments are conducted where a model is passed purely on-the-grid data as per figure 4.3. If the on-grid data is highly correlated but sampled from a different process (i.e. having 0.8 correlation) the model fails to capture the data well, as can be seen from low the validation log-likelihood. In the experiment with only on-the-grid data from the target process (correlation of 1.0), the model is able to learn adequately, but still performs worse compared to having both modalities of data at high correlation. So even with perfect on-the-grid data, the model still benefits from also having access to off-the-grid data. Note that for fairness the on-the-grid data now has to be sampled from the smaller $[-2.0, 2.0]$ range, since none of the other models have seen context data from the target process outside this range either. Training with fully correlated on-the-grid data outperforms having access to just off-the-grid data, even with less datapoints, as the number of points per unit is lowered to 4. A likely explanation is that the perfect equidistant distribution of this structured data is more informative about the whole GP progress than datapoints at random and possible almost overlapping locations.

In conclusion, the models benefited from having access to both modalities of context data, despite the target data being exclusively sampled from the off-the-grid generating process. This shows that the 1D synthetic task is sufficiently realistic. A regular TNP without grid-encoding is able to model this synthetic data adequately, so the next step is to evaluate what happens when grid-encoders are introduced.

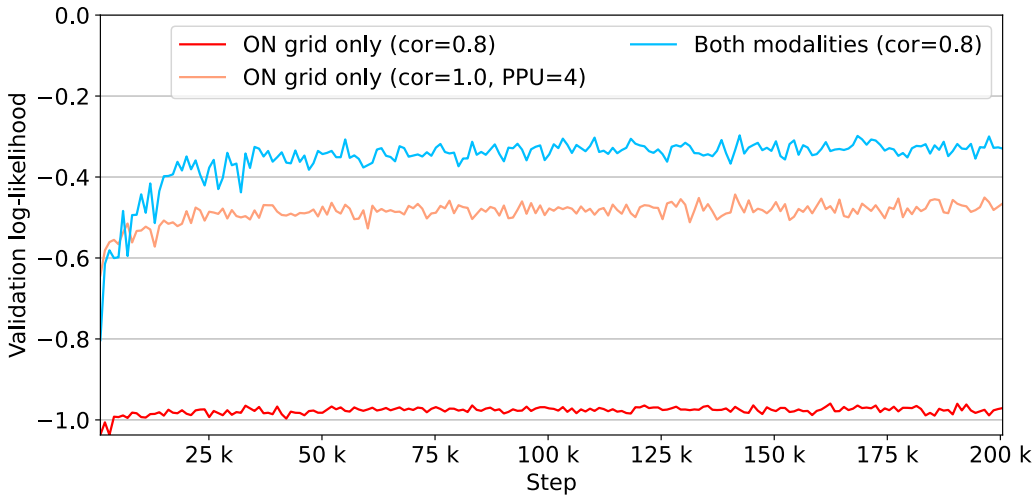


Figure 4.3: Validation log-likelihood curves for regular TNP models without a grid-encoder, trained on purely on-the-grid data. Even with perfectly correlated on-the-grid data, the model still benefits from being passed off-the-grid data too. If the model only observes correlated data, it fails to learn the data generating process.

4.1.2 Comparing both grid-encoders

This section serves to compare the performance of the grid-encoders outlined in section 3.3 with the regular TNP from the previous section. Ideally the performance lost when introducing the grid-encoders is negligible. Since the real-world task will also have correlated modalities, the learning task with quite correlated modalities is most important.

Figure 4.4 proves that both the SetConv and MHCA Grid encoder are able to obtain a performance which is almost identical to the standard TNP without a grid encoder in terms of validation log-likelihood on this synthetic task, which is expected to form an upper limit here as per section 3.2. Table 4.1 also lists the validation log-likelihood for a more extensive set of experiments.

Validation log-likelihood	Data modality		
Grid-encoding	Both (cor=0.8)	Both (cor=0.5)	Off-grid
None (regular TNP)	-0.3297	-0.4603	-0.5554
SetConv	-0.3335	-0.4669	-0.5566
MHCA	-0.3341	-0.4709	-0.5598

Table 4.1: Validation log-likelihood (less negative is better) for TNPs with different grid-encoders trained on 1D synthetic data of varying composition.

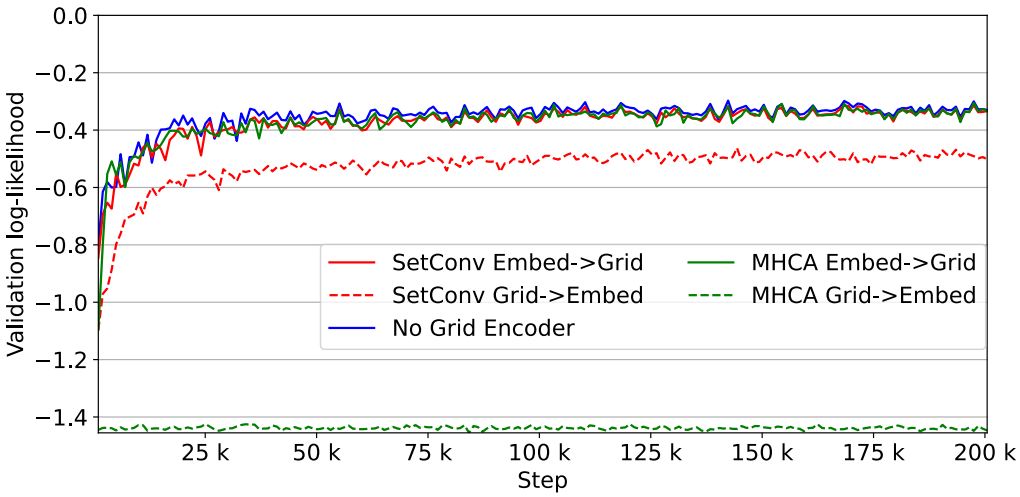


Figure 4.4: Validation log-likelihood curves for TNPs with different grid encoder combinations. Each model is evaluated with off-the-grid target data and is passed the same dual modality context data with a 0.8 task-correlation.

Figure 4.4 also contains an ablation where the order of the embedding and the grid encoder are swapped. Remember that the embedding encodes additional information into the data and projects it into a 128 dimensional vector as per section 3.2. Embedding the data prior to the grid encoder (Embed->Grid) performs far superior to joining both modalities before embedding (Grid->Embed) for both grid-encoders, but especially for the MHCA Encoder. There are two possible explanations that likely jointly explain this phenomenon. Firstly, when grid-encoding the raw data first, the data modalities do not have the additional flag dimensions with information about the modality of the data when they are grid-encoded. After the grid-encoder only on-the-grid data remains, meaning the embedding cannot learn to distinguish the modality of each datapoint prior to grid-encoding. Secondly, some of the data manipulations inside the grid-encoder likely do not perform well in low-dimensional spaces. Especially for the MHCA encoder, computing attention on raw observations will likely not allow any information of the original data to persist through this operation. Based on these results, all future experiments embed the data before applying the grid-encoders.

4.1.3 Validating structured data-transformers

After evaluating the grid-encoders in a standalone experiment, it remains to validate the implementation of some efficient structured-data transformers. Since these architectures will later be applied to two dimensional real-world grids, experiments are conducted on 2D synthetic

data. As the data generation process has cubic computational complexity, the grid resolution is lowered to one datapoint per unit. With a range of $[-4.0, 4.0]$ per dimension, this implies there are still 64 on-the-grid datapoints. The numbers of off-the-grid context and target datapoints are kept the same.

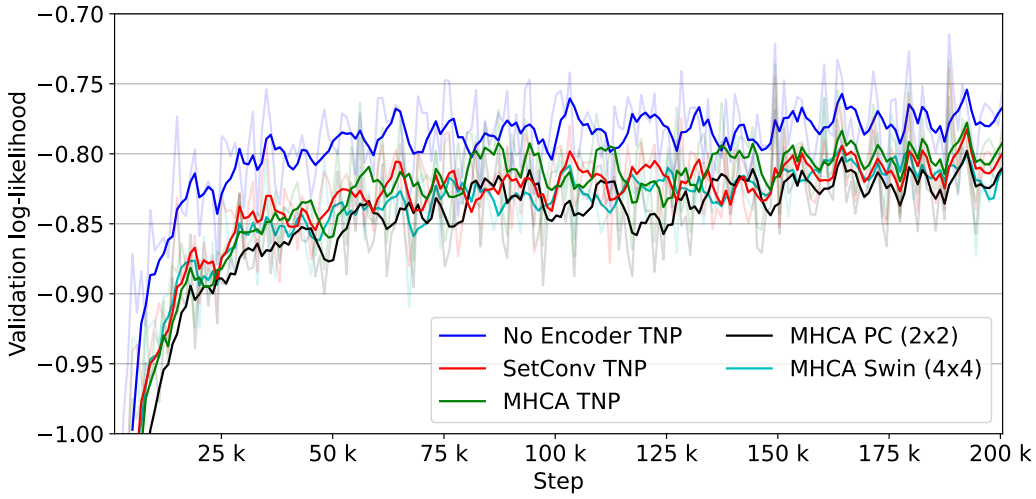


Figure 4.5: Validation log-likelihood for TNPs trained on 2D synthetic data. For shifted windows attention (Swin) and patch coarsening (PC), the number in brackets refers to the window sizes and patch sizes respectively. Note that this plot is smoothed with a moving average (window=5) for clarity.

Figure 4.5 shows the validation log-likelihood for models trained on two-dimensional synthetic data. The two grid-encoders are compared with a baseline no encoder TNP. The performance loss caused by using a grid-encoder is slightly more than before, which is likely caused by the grid being far coarser, implying more information is lost when moving unstructured data to this grid. Both grid-encoders still perform almost identical, and as they are interchangeable in terms of output, it suffices to perform the remaining experiments using just the MHCA Encoder. Applying either patch coarsening (PC) or using shifted windows attention instead of full self-attention does not seem to significantly alter performance, therefore validating the correctness of their implementation. The primary aim of these new architectures is to enable modelling data at a larger real-world scale, so achieving similar performance as the computationally expensive standard TNP is very promising. Speed comparisons will be conducted on large scale data in the next section, as the synthetic dataset is too small to show meaningful comparisons.

4.2 Predicting ERA5 temperature

The previous section has illustrated that the grid-encoders and efficient attention mechanism work on synthetic GP data. Some larger scale experiments can now be conducted on real-world data, by predicting surface temperature with the ERA5 dataset as per section 3.1.2. Based on the approaches outlined in the methodology, a lot of different models can be created, which are visualised in figure 4.6. Due to computational constrains not all permutations are tried, but instead the most sensible experiments are chosen based on intermediate results, exploiting the compartmental design of the model.

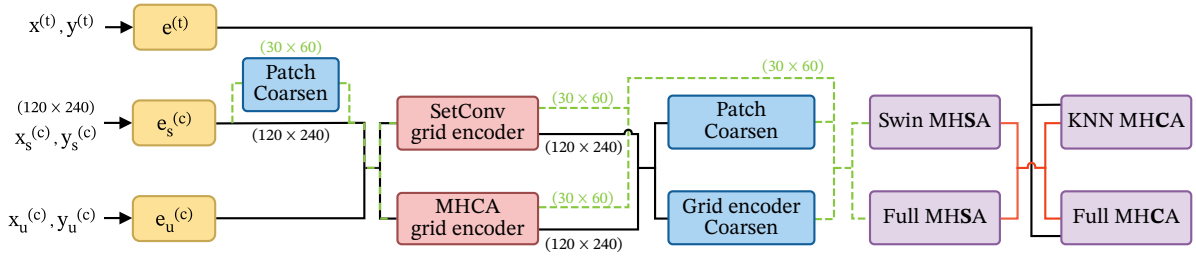
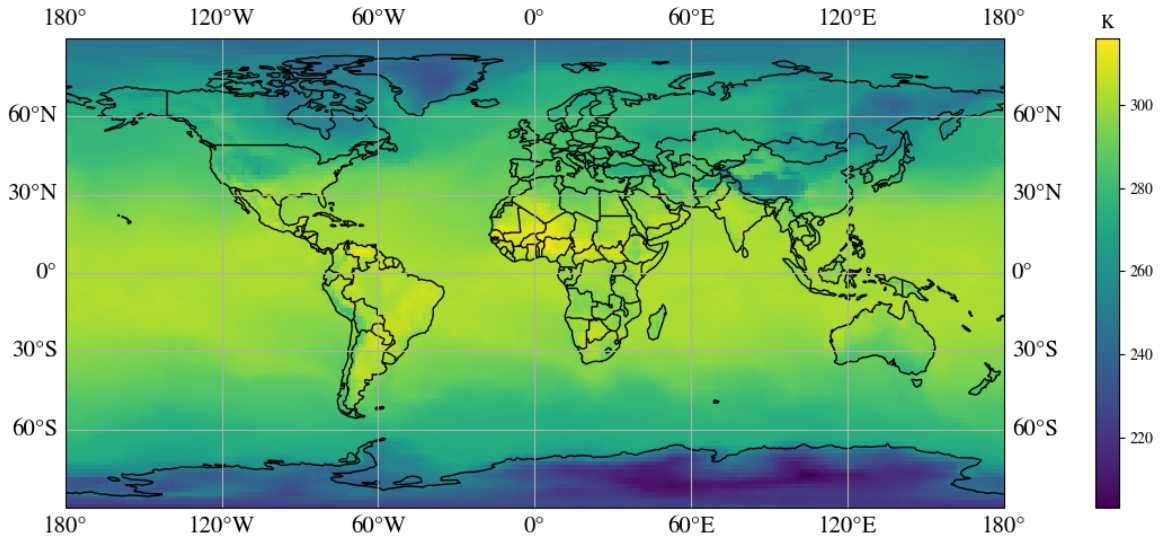


Figure 4.6: Schematic of possible modelling choices and subsequent pipeline for ERA5 data. The off-the-grid target and {on, off}-the-grid context data are embedded on the left. After optionally coarsening the on-the-grid context, both context modalities are grid-encoded. If no coarsening has happened yet, the output of the grid-encoder is then coarsened. Next the processed context data performs one of two forms of self-attention. Finally, cross-attention with the embedded target data is performed. Black lines indicate the data's original resolution, whereas green lines indicate a coarsened resolution.

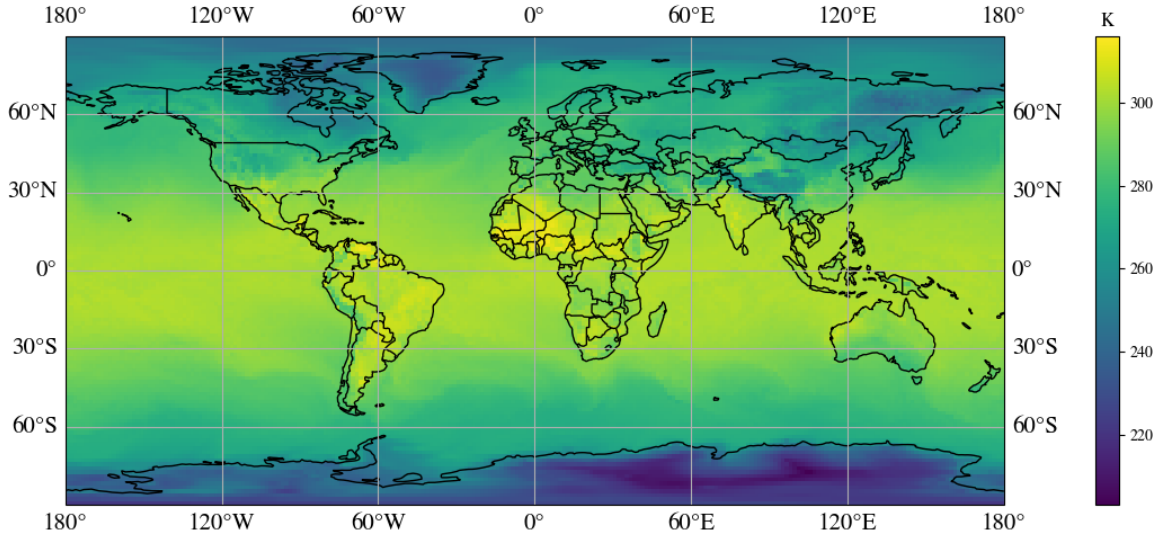
The ERA5 data is available at an hourly resolution, where care is taken to ensure the training and validation datasets focus on different years. Regardless of the experiment, each epoch consists of $8,000$ training and $2,000$ validation datasets, sampled at random from their respective date ranges. A single year of ERA5 temperature data already surpasses 18GB of storage, limiting the size of the date ranges used. Streaming data from cloud storage is also possible, but this increases the runtime of the experiments and is therefore not used yet. Note that the number of samples per epoch is not influenced by the date range provided, but a larger date range drastically reduces the probability of the same datapoint being sampled, even across epochs.

The next sections will first focus on the target cross-attention inside the TNP, corresponding to the last two purple boxes in figure 4.6. Secondly, the effect of coarsening and grid-encoding will be jointly evaluated, corresponding to the red and blue boxes in figure 4.6. This evaluation will take place on a TNP with complete self-attention, after which the third section will focus on introducing Swin mechanisms. The largest feasible resolution when doing full MHSA is around 30×60. When Swin self-attention is used, it is possible to remove the coarsening altogether and maintain the input resolution of 120×240, due to Swin's greater computational efficiency.

Section 4.2.5 investigates the potential improvements in computational speed, and the final section provides a more qualitative assessment of the model's predictions. See figure 4.7 for a qualitative illustration of the global temperature prediction task.



(a) Predictive mean



(b) Ground truth

Figure 4.7: Global two meter temperature (Kelvin) predictive mean and ground truth at a single time instance. The model is a MHCA grid encoding TNP with shifted windows self-attention and 9-NN cross-attention, operating at a pseudo-grid resolution of 120×240.

4.2.1 Cross-attention with the target data

This section investigates which form of cross-attention between processed context data and target data performs best. Either full cross-attention is used, where target points can attend to every processed context datapoint, or K-nearest neighbour cross-attention. $K = 9$ is used, since this is the lowest square of an odd number, as required by section 3.4.2. The dual modality ERA5 data with a 120×240 grid is first grid-encoded, before being coarsened to a 30×60 grid for the downstream TNP, since that is the largest resolution that every model can reasonably process as per section 4.2.5.

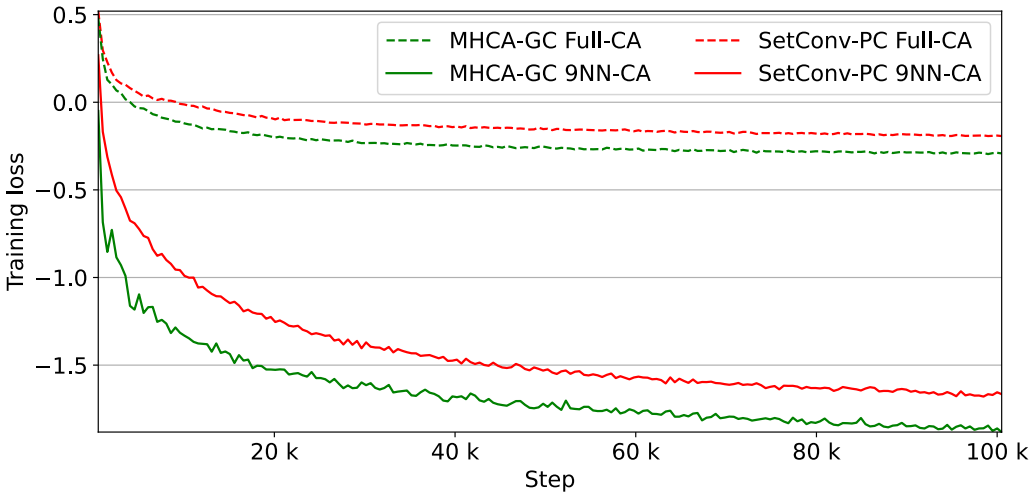


Figure 4.8: Comparison between TNPs (no shifted windows attention) with full target cross-attention and nearest neighbours (K=9) cross-attention, on two different grid-encoders, with either Patch Coarsening (PC) or Grid encoder Coarsening (GC). Models are trained on ERA5 temperature data from 2019.

For these initial experiments, models were trained on ERA5 temperature data from all of 2019 and validated on data from January 2020. The training loss in figure 4.8 clearly demonstrates a stark difference between the two types of target cross-attention. With full cross-attention both models fail to properly learn within the time limit set for this experiment, with 9-nearest neighbour cross-attention optimising far quicker and reaching superior results. We postulate this could be caused by the powerful inductive bias induced by KNN-cross attention; when predicting temperature relatively local information is perhaps most relevant, so it might be beneficial, especially at coarsened resolutions, to have this filter built-in. Because of this experimental result, all further architectures will feature 9-nearest neighbour target cross-attention.

4.2.2 The effect of coarsening the grid

This section aims to compare the MHCA and SetConv grid encoders and three different coarsening strategies. The coarsening strategies are to patch coarsen (PC) before grid-encoding, patch coarsen after grid-encoding or to let the grid-encoder map to a coarser grid (GC). For a fair comparison the remaining TNP architecture is identical across all experiments, and each model maps from the original 120×240 resolution to a 30×60 resolution. Since there are 1800 pseudo-tokens, 1-2k off-the-grid datapoints are sampled per epoch, so that the on-the-grid data remains beneficial for predictive inference.

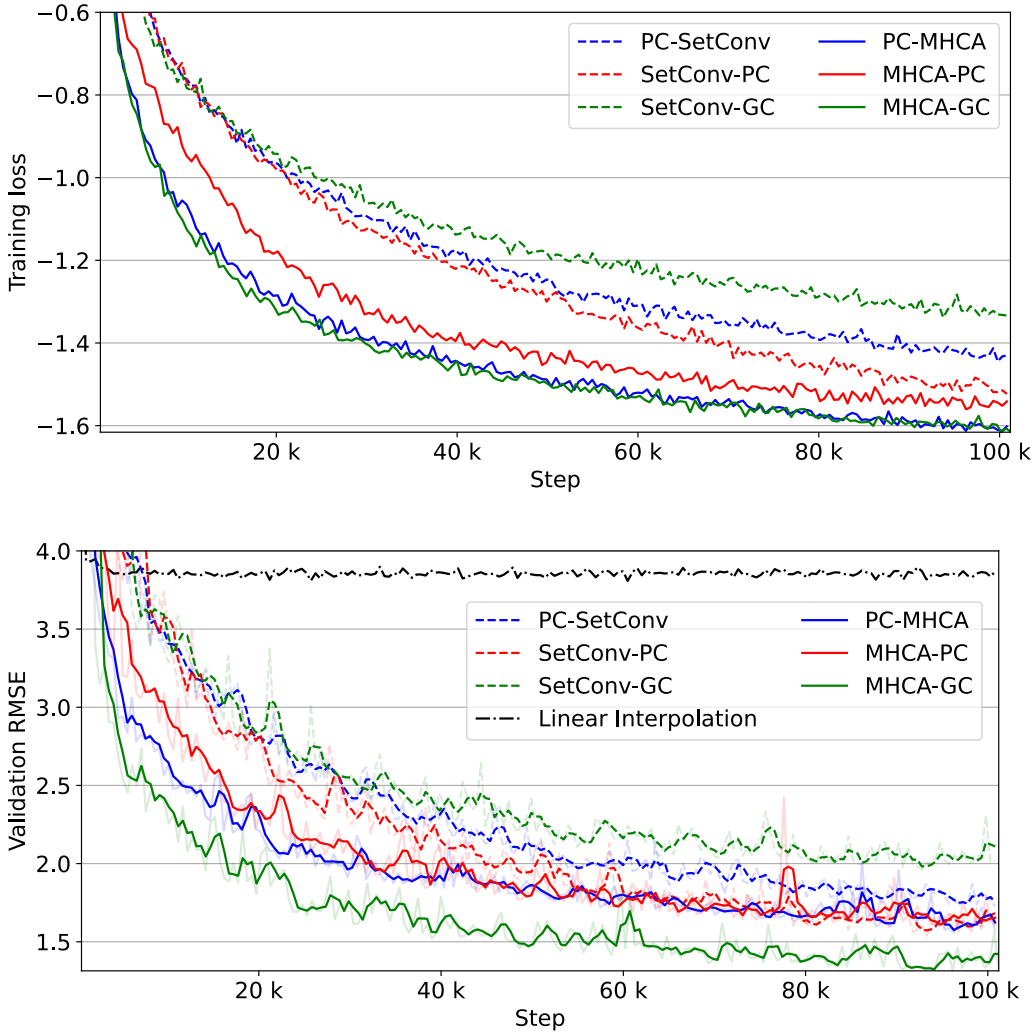


Figure 4.9: Training and validation curves for different grid encoders and coarsening strategies, each reducing to a 30×60 grid. Models are trained on dual modality temperature context data with 1-2k off-the-grid surface temperature measurements and 3000 target surface temperature points. **Top:** Training loss per step. **Bottom:** Moving average smoothed validation set RMSE (smoothing window=3)

Figure 4.9 illustrates the results of these experiments. A simple deterministic linear interpolation is added as a baseline for the validation RMSE. This baseline first linearly interpolates both modalities of data separately, and then outputs a linear combination of these interpolants using two trainable weights. From especially the training loss in figure 4.9, it is clear that the SetConv encoder is outperformed by the MHCA encoder regardless of coarsening strategy. While it could be argued that some SetConv experiments have not fully converged within the epoch limit, convergence speed is also very important due to the enormous scale of downstream applications in environmental modelling. For the MHCA encoder, grid-encoder coarsening and patch coarsening before grid-encoding perform best on the training loss, with grid-encoder coarsening achieving the best validation RMSE. This was expected, since the MHCA encoder is a lot more dynamic, which allows for a more intelligent coarsening than having to apply the same patch-coarsening convolution to each patch of data.

For the SetConv encoder the coarsening results are almost the opposite of the MHCA approach, with grid encoder coarsening performing by far the worst. A likely explanation is that the SetConv encoder, which only features a single learnable parameter, is not expressive enough to actually learn any coarsening apart from a homogeneous blurring. Patch embedding has a learnable convolution which can at least combine multiple local datapoints in its smoothing, making it more powerful. The MHCA encoder boasts a whole attention layer full of weights instead, which do allow it to learn a more involved coarsening.

4.2.3 Does the model learn from both modalities?

Having established how the grid-encoders and coarsening strategies perform, this section will conduct experiments to evaluate to what extent the model learns from both modalities of data. The MHCA-GC-TNP will be used, coarsening to a resolution of 30×60, as per the previous section. The composition and modalities used for the context data will be varied. The on-the-grid data (if applicable) will always originally be 120×240 and 3000 target points are used. Since the off-the-grid context data (if applicable) concerns the same variable as the target data, it is interesting to vary the amount available to the model.

Figure 4.10 shows the training loss and validation RMSE for these experiments. As expected, only having access to on-the-grid data performs far worse than learning from both modalities in any setting. When the number of off-the-grid points is sufficiently low, the model benefits substantially from having access to the on-the-grid data, especially when considering RMSE. For larger amounts of off-the-grid data the model starts to increasingly ignore the on-the-grid data. Considering that the model is compressing to an internal resolution of just 1800 pseudo-

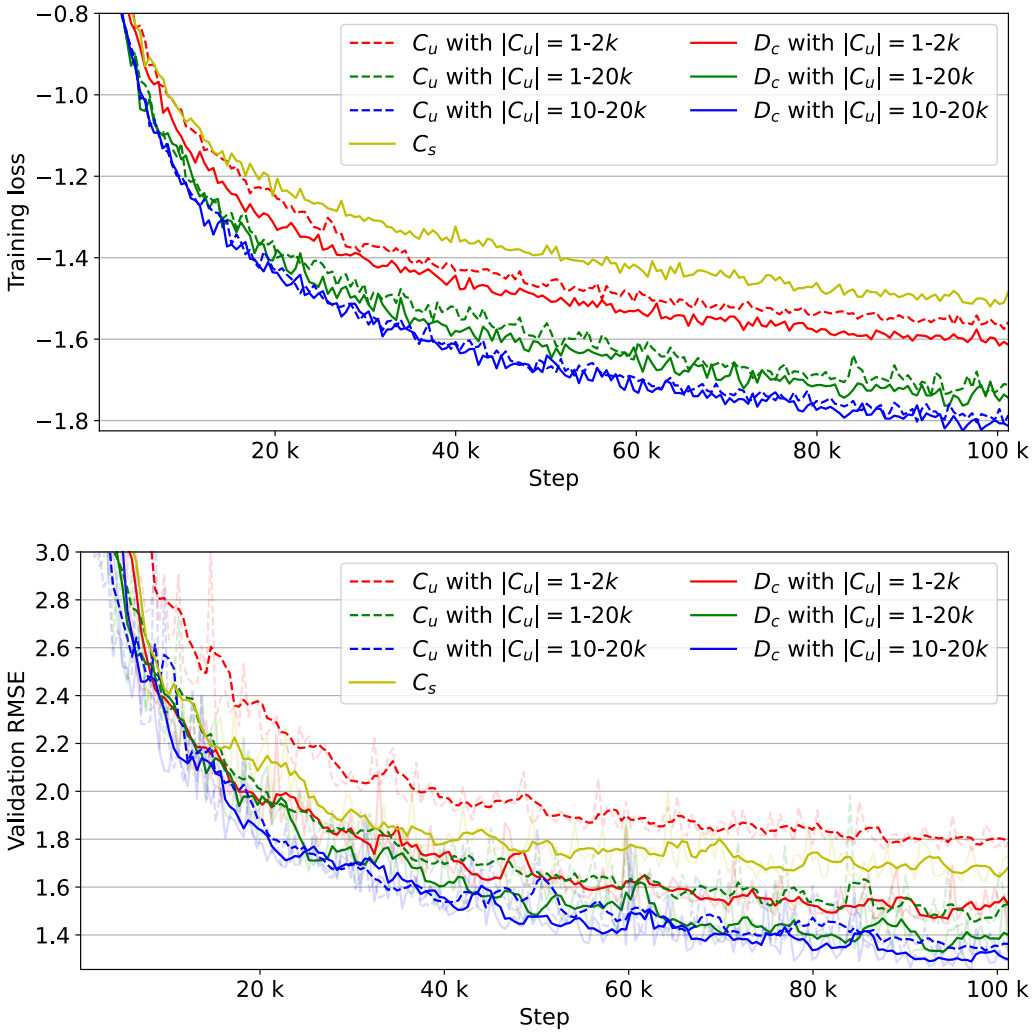


Figure 4.10: Training and validation curves for the same MHCA Grid Encoder TNP, where the grid encoder coarsens to a 30 by 60 resolution. The context data consists of either both modalities of data (D_c), or exclusively off-the-grid (C_u) or on-the-grid (C_s) data. The number of off-the-grid context points (if applicable) is also varied and denoted in the legend. **Top:** Training loss per step. **Bottom:** Validation set RMSE.

tokens however, this is not too surprising; with $20,000$ off-the-grid datapoints, this amounts to roughly 11 per pseudo-token, so the different (on-the-grid) temperature variable likely no longer contributes much information. This is especially true considering our off-the-grid data is uniformly sampled. In settings with far larger numbers of pseudo-tokens, we expect that it is possible to increase the number off off-the-grid context points while maintaining the model's ability to learn from both modalities. Increasing the number of pseudo-tokens would for example be feasible by using Swin self-attention.

4.2.4 Introducing Swin attention

This section focusses on testing the potential benefits from introducing Shifted windows self-attention (Swin). Firstly, the full self-attention inside the MHCA-GC-TNP from the previous section is replaced by Swin-style attention. Due to the greater computational efficiency (further illustrated in section 4.2.5), the input resolution of 120×240 can now also be used throughout the whole model, without coarsening.

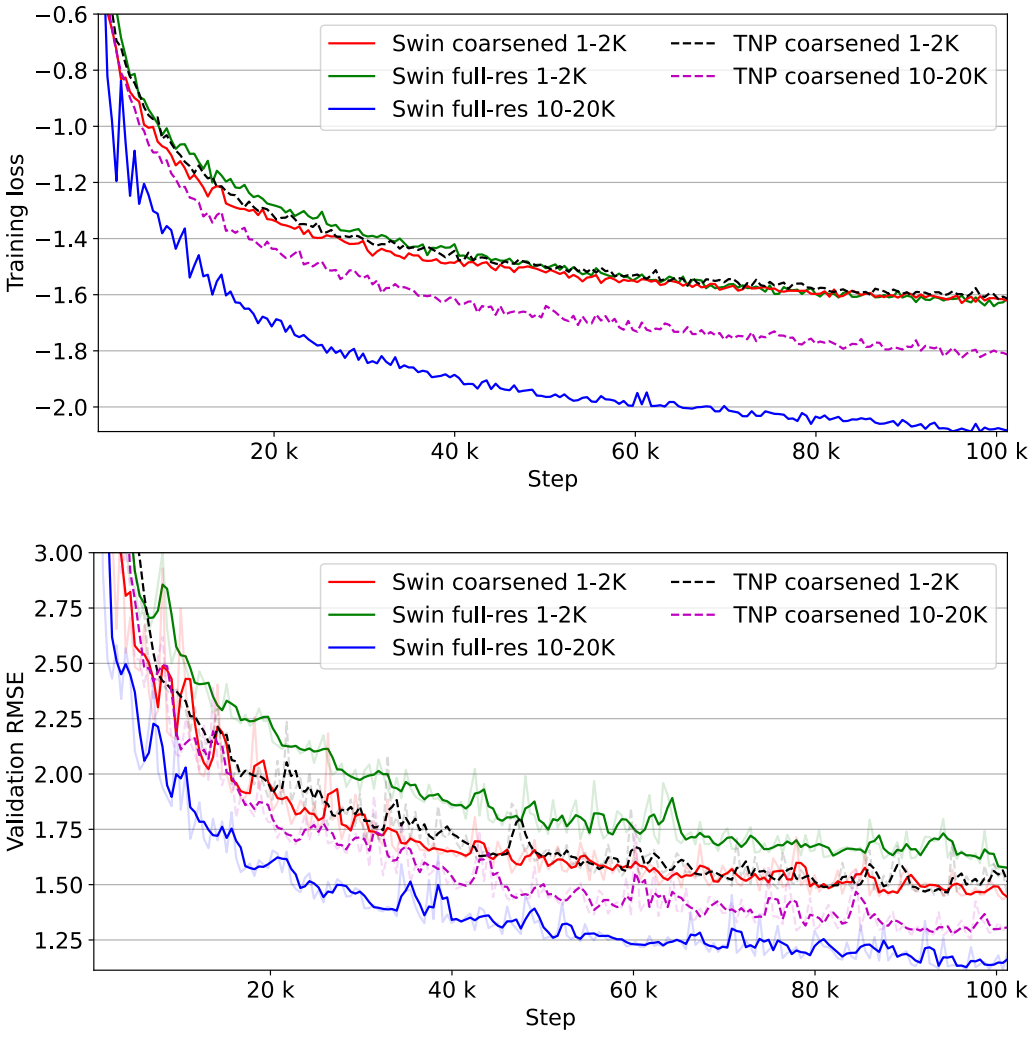


Figure 4.11: Training and validation curves for experiments comparing shifted windows self-attention (Swin) with regular self-attention (TNP). The MHCA Encoder is used for each experiment and the Swin window size is 5×5 (if applicable). The coarsened resolution entails that the encoder coarsens to 30×60, with the original full resolution being 120×240. **Top:** Training loss per step. **Bottom:** Moving average smoothed validation set RMSE (smoothing window=3).

Figure 4.11 compares Swin self-attention with full self-attention in terms of training loss and validation RMSE. When coarsening the Swin's pseudo-grid, it obtains a very similar training and validation performance to the full self-attention TNP. When a Swin processes the 1-2K off-the-grid context tokens, using the full resolution (no coarsening) actually achieves worse final performance compared to coarsening. While this was unexpected, it could be the case that in this scenario with very few off-the-grid datapoints, it is easier to learn a coarse resolution where each grid location has on average one associated off-the-grid point. It also seems as if the full resolution Swin has not yet fully converged within the epoch limit. At this full resolution with 28800 uncoarsened on-the-grid datapoints, we therefore expect the Swin's performance to be severely bottlenecked by its lack of off-the-grid context data, as investigated in section 4.2.3. This becomes evident when increasing the number of off-the-grid context points to 10-20k. In this last comparison, the full-resolution Swin significantly outperforms the coarsened TNP trained on the same data, illustrating the merit of modelling the grid at a higher resolution.

4.2.5 Computational speed

The previous two sections have outlined the modelling performance of the developed architectures, illustrating which are best at capturing and predicting trends in ERA5 surface temperature data. Apart from being able to correctly model the data, an equally important consideration is computational efficiency. Extrapolations from section 4.1 would for example suggest a TNP without coarsening would perform well, but this architecture would be completely infeasible to train on the large ERA5 dataset.

Iterations per second	Coarsening resolution			
Model	20×40	30×60	60×120	120×240 (original)
TNP with 9NN-CA	3.63	2.65	0.52	0.05[1]
Swin (windows of 10 x 10)	4.23	3.17	2.50	1.12
Swin (windows of 5 x 5)	4.30	3.90	2.64	1.24

Table 4.2: Iterations per second averaged over a full training epoch, measured on random noise data of identical composition to ERA5. Swin denotes a TNP with shifted windows self-attention and 9 nearest-neighbours cross attention. [1]Extrapolated after 30 minutes of runtime.

To compare computational efficiency, table 4.2 lists the iterations per second during training for various configurations. Since the data loader could be a bottleneck for some architectures, which prevents benchmarking the efficiency of the actual model, this experiment is instead conducted on random white noise data. However, care has been taken to ensure that the

dimensionality and structure of this random data matches that of ERA5 exactly, including the equidistant on-the-grid input locations $x_{si}^{(c)}$. Compared to the naive original resolution TNPs, some of the coarsened shifted windows based TNPs achieve over 80 times more iterations per second, which is an enormous improvement in computational efficiency.

4.2.6 Evaluating the final model

By combining the results from each previous section, an optimal model has been constructed. A TNP with shifted windows self-attention, nearest neighbours cross-attention, a MHCA grid encoder and no coarsening has achieved the best validation RMSE. This model will now first be compared with some baselines, after which it will be assessed qualitatively through exploring its predictions.

Model	Off-grid contexts	Val. log-likelihood	Val. RMSE	Iter. / s
Linear Interpolation	1-2K	-	3.859	-
LBANP (L=16)	1-2K	0.129	9.206	0.84
MHCA-GC-Swin (ours)	1-2K	1.586	1.448	**3.90**
Linear Interpolation	10-20K	-	1.862	-
MHCA-Fullres-Swin (ours)	10-20K	**2.032**	**1.193**	1.24

Table 4.3: Comparison of the newly developed architectures, the LBANP [Feng et al., 2022] and linear interpolation baselines. Iterations per second have been measured on white noise.

Two baselines are compared against the optimal model. The first baseline is the learned combination of two linear interpolations, as discussed in section 4.2.2. Secondly, we attempt to compare against the LBANP as introduced in Feng et al. [2022]. Since the LBANP is also a TNP, architecture hyperparameters, such as the number of attention layers, attention heads or embedding dimensionality are left unchanged. For a fairer comparison, the same SirenNet coordinate encoder is added prior to the LBANP. Although this is a learnable layer and therefore part of our model, it is sufficiently handcrafted to justify this addition. The LBANP cannot handle on-the-grid data, and therefore all ERA5 data is treated as an off-the-grid collection. The only hyperparameter that has to be specifically tuned for the LBANP is the number of latent vectors L. As the LBANP cannot efficiently handle large amounts of (on-the-grid) data, it is very slow on the ERA5 dataset. $L = 16$ is the highest number of latents that allows the model training to finish within the computing cluster's maximum of 36 hours. While it would be possible to interpolate or coarsen the on-the-grid data to achieve greater computation, these are contributions of this research and would go beyond comparing to the actual LBANP.

Table 4.3 illustrates that our models outperforms the baselines both in terms of computational efficiency, as well as validation performance. The full global prediction of the MHCA-Fullres-Swin have been shown in the beginning of this section in figure 4.7. Since the models output a probability distribution, it is also possible to visualise the uncertainty, by plotting the standard deviation. Figure 4.12 shows that the model seems to learn sensible uncertainties, as the highest standard deviations occurs in mountainous regions such as the Andes, Rocky mountains and the Himalayas. Due to the large and sudden altitude differences in these regions, the temperature can indeed change very suddenly.

Lastly, two derived plots are show in figure 4.13. The top plot shows the global error, the difference between the ground truth and predictive mean show in figure 4.7. The maximum errors are quite substantial, but the majority of errors are much smaller, especially over the oceans. Considering that at our 1.5° resolution a single token can still represent an area of over a 150 by 150km, having such high errors in for example mountainous regions is to be expected. The second plot in figure 4.13 shows the normalised error, obtained by dividing the error by the predicted standard deviation, which produces a Z-score. Over the oceans, where the predicted uncertainty is less than one, the Z-score is higher than the error, with the opposite happening across mountainous areas. The second plot shows a more uniform distribution of colour compared to the raw error, which suggests the model has learned to predict sensible uncertainties.

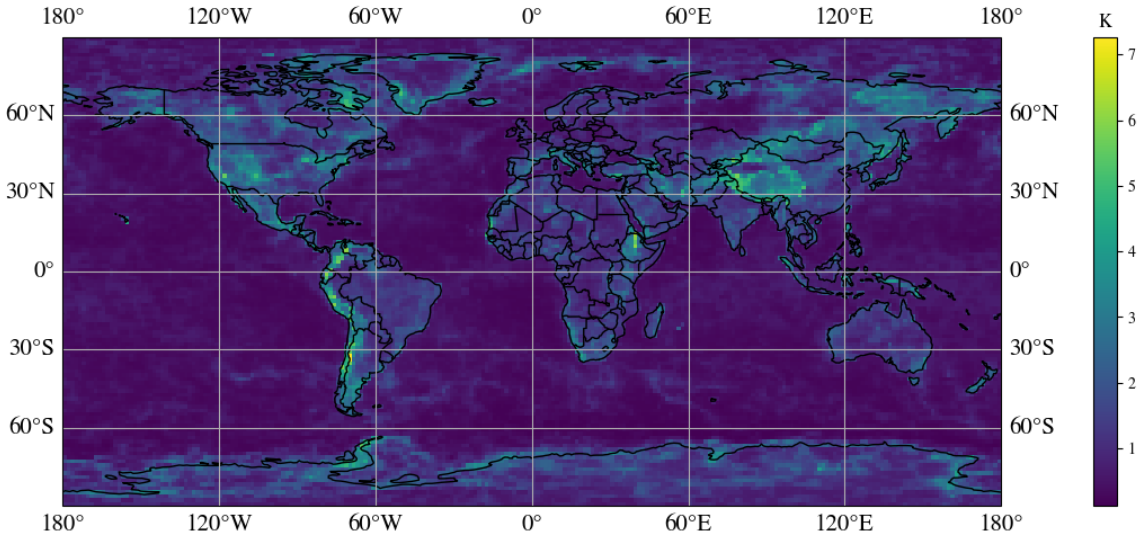
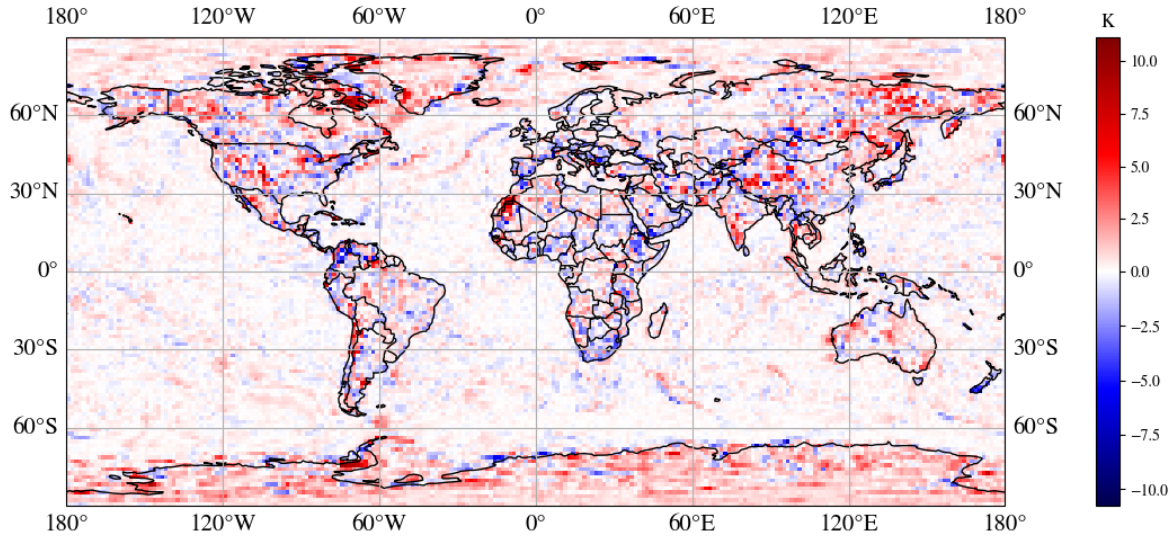
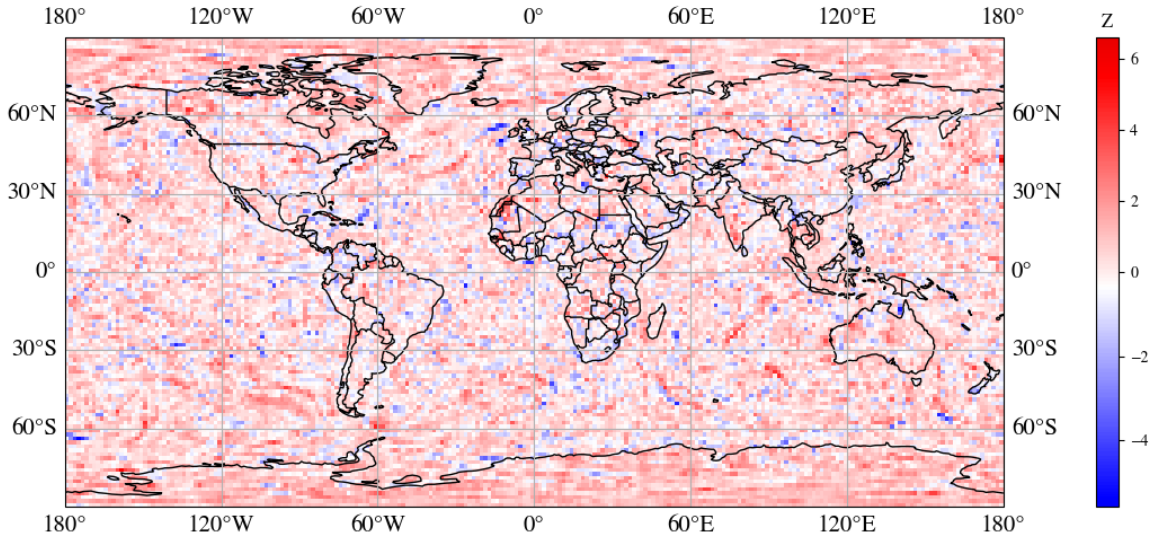


Figure 4.12: Global two meter temperature (Kelvin) predictive standard deviation. The model is a MHCA grid encoding TNP with shifted windows self-attention and 9-NN cross-attention, operating at a pseudo-grid resolution of 120×240.



(a) Error (ground truth - predictive mean)



(b) Normalised error

Figure 4.13: The error between the ground truth and the predictive mean is visualised. Normalisation entails diving the error by the predicted standard deviation, which produces a Z-score. The model is a MHCA grid encoding TNP with shifted windows self-attention and 9-NN cross-attention, operating at a pseudo-grid resolution of 120×240.

Chapter 5

Discussion

The purpose of this thesis was to improve the modelling of on- and off-the-grid data with Transformer Neural Processes, while simultaneously focussing on efficient computation. The particular use case of this research was spatio-temporal environmental modelling, but findings from this research are more generally applicable. The new models have been extensively evaluated, both on synthetic data and on real-world ERA5 data. Firstly, two different grid-encoders are explored, which can map dual modality data to a grid of pseudo-tokens. The SetConv encoder is inspired by Convolutional CNPs, while the MHCA Encoder is a new architecture based on an efficient batched nearest-neighbours MHCA layer. Secondly, different architecture improvements for structured data transformers are introduced. Both patch coarsening and shifted windows self-attention were inspired by breakthroughs from computer vision transformer research. The last improvement is the addition of nearest-neighbours cross-attention. The current models only have between 1 to 2 million weights, which is tiny compared to most Transformer architectures in use today [Han et al., 2023].

The main contributions and recommendations of this research are as follows:

- Both of the developed grid-encoders are able to adequately combine on- and off-the-grid data, such that a TNP can learn from both modalities. On real-world ERA5 data, the MHCA grid-encoder is shown to outperform the SetConv encoder.
- Two trainable strategies to coarsen the grid of pseudo-tokens have been created. They are effective at allowing computationally efficient modelling of high resolution data.
- The most effective coarsening can be achieved by letting the MHCA grid-encoder natively map to a coarser grid. If this is computationally unfeasible, coarsening the on-the-grid data using patch embeddings is the preferable option.

- Shifted windows self-attention and nearest-neighbours cross-attention are highly effective at reducing the computational complexity of the TNP's attention layers. By focussing on local information, they allow models to process higher fidelity grids efficiently, outperforming full-attention TNPs which require coarsening.

5.1 Future work and improvements

This research has introduced both new and modified design paradigms, and in doing so has accomplished much of the groundwork for the development and testing of these new models, on fairly large scale settings. Multiple interesting avenues of future research are possible, which could be inspired by the extensive array of environmental research that has been conducted with other (NP) models [Andersson et al., 2023; Bauer et al., 2015; Bi et al., 2022; Price et al., 2023; Vaughan et al., 2022].

Firstly, it would be interesting to evaluate a larger variety of permutations of the architectures developed in this research. Because of limited time and computational budget, not all options suggested in figures 3.11 and 4.6 have been attempted. Apart from increasing efficiency, the Swin self-attention and KNN cross-attention also force a model to pay more attention to local information. It is an open question to what extent this inductive bias also aids the model's predictions.

All real-world experiments in this research have predicted surface temperature. However, the ERA5 dataset [Hersbach et al., 2020] features 262 environmental variables, which are distributed over different height levels. Each variable is also available from 1940 to the present day, allowing for an even wider range of data. It would therefore be interesting to see how the models developed would fare on different ERA5 variables such as wind speed or air pressure.

A further avenue for future research could be to use off-the-grid data from a different dataset entirely, such as Dunn et al. [2016]. A major downside of the current approach is that the surface temperature is sampled at random, implying that the "off-the-grid" data is equally likely to be sampled in the middle of the ocean as it is to feature a densely populated area. This is not very realistic, as off-the-grid in-situ observations are often concentrated in Western regions with dense populations [Vaughan et al., 2024].

Fourthly, if more computing resources were available, it would be interesting to evaluate the model at even higher grid resolutions. While our resolution of 120×240 is comparable to Aardvark [Vaughan et al., 2024], deterministic NWP stations in production, such as IFS-HRES [Rasp et al., 2023], are able to predict at the original 720×1440 resolution.

Apart from modelling different ERA5 variables separately, it would be interesting to model multiple environmental variables simultaneously. This would not require any modification to the existing architecture, provided the variables are globally complete so that they can be stacked to have a multi-dimensional output vector at each data location. Larger scale research, such as Aardvark, models 24 variables simultaneously and it would be interesting to apply the architectures developed in this research to real-world weather forecasting or statistical downscaling for example.

When modelling multiple variables at the full 0.25° resolution, it might be beneficial to increase the complexity and number of weights of the model. A possibility would be to introduce hierarchical feature maps [Liu et al., 2021] into the shifted windows self-attention layers of the TNP.

When modelling multiple variables at the same time, the small MLP networks, used for encoding or decoding the data and model output respectively, could form a potential performance bottleneck. Increasing their number of weights and layers could be an option, but perhaps more interesting would be to instead replace them by a mixture of experts [Zhou et al., 2022], for example featuring an MLP for every modelled variable. From other models such as Aardvark, it is known that a model predicting multiple variables struggles to achieve the same performance as different models trained on each individual variable. Training individual models is more costly, and a mixture of experts architecture could potentially be a solution. This becomes especially important in regimes with limited data.

In general, this thesis has shown promising results when modelling dual modality ERA5 data across the entire earth. To be able to compare with state of the art research and NWP systems, multiple variables should be jointly modelled at a higher resolution, with different datasets for on- and off-the-grid data. If the results of this research hold at that large scale, the computational benefits could be huge. It would therefore be highly interesting to see how the new architectures perform at the scale described for operational numerical weather prediction systems.

References

Alvarez, M. A., Rosasco, L., Lawrence, N. D., et al. (2012). Kernels for vector-valued functions: A review. *Foundations and Trends® in Machine Learning*, 4(3):195–266.

Andersson, T. R., Bruinsma, W. P., Markou, S., Requeima, J., Coca-Castro, A., Vaughan, A., Ellis, A.-L., Lazzara, M. A., Jones, D., Hosking, S., et al. (2023). Environmental sensor placement with convolutional gaussian neural processes. *Environmental Data Science*, 2:e32.

Aronszajn, N. (1950). Theory of reproducing kernels. *Transactions of the American mathematical society*, 68(3):337–404.

Bauer, P., Thorpe, A., and Brunet, G. (2015). The quiet revolution of numerical weather prediction. *Nature*, 525(7567):47–55.

Bejani, M. M. and Ghatee, M. (2021). A systematic review on overfitting control in shallow and deep neural networks. *Artificial Intelligence Review*, 54(8):6391–6438.

Bi, K., Xie, L., Zhang, H., Chen, X., Gu, X., and Tian, Q. (2022). Pangu-weather: A 3d high-resolution model for fast and accurate global weather forecast. *arXiv preprint arXiv:2211.02556*.

Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. (2017). Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877.

Bodnar, C., Bruinsma, W. P., Lucic, A., Stanley, M., Brandstetter, J., Garvan, P., Riechert, M., Weyn, J., Dong, H., Vaughan, A., et al. (2024). Aurora: A foundation model of the atmosphere. *arXiv preprint arXiv:2405.13063*.

Bonilla, E. V., Chai, K., and Williams, C. (2007). Multi-task gaussian process prediction. *Advances in neural information processing systems*, 20.

Cohen, T. et al. (2021). *Equivariant convolutional networks*. PhD thesis, Taco Cohen.

Cohen, T. and Welling, M. (2016). Group equivariant convolutional networks. In *International conference on machine learning*, pages 2990–2999. PMLR.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

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

Dunn, R. J., Willett, K. M., Parker, D. E., and Mitchell, L. (2016). Expanding hadisd: Quality-controlled, sub-daily station data from 1931. *Geoscientific Instrumentation, Methods and Data Systems*, 5(2):473–491.

Duvenaud, D. (2014). *Automatic model construction with Gaussian processes*. PhD thesis, University of Cambridge.

Eyre, J., Bell, W., Cotton, J., English, S., Forsythe, M., Healy, S., and Pavelin, E. (2022). Assimilation of satellite data in numerical weather prediction. part ii: Recent years. *Quarterly Journal of the Royal Meteorological Society*, 148(743):521–556.

Fan, J., Ma, C., and Zhong, Y. (2021). A selective overview of deep learning. *Statistical science: a review journal of the Institute of Mathematical Statistics*, 36(2):264.

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

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

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

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

Gettelman, A., Geer, A. J., Forbes, R. M., Carmichael, G. R., Feingold, G., Posselt, D. J., Stephens, G. L., van den Heever, S. C., Varble, A. C., and Zuidema, P. (2022). The future of earth system prediction: Advances in model-data fusion. *Science Advances*, 8(14):eabn3488.

Gillioz, A., Casas, J., Mugellini, E., and Abou Khaled, O. (2020). Overview of the transformer-based models for nlp tasks. In *2020 15th Conference on Computer Science and Information Systems (FedCSIS)*, pages 179–183. IEEE.

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

Han, K., Wang, Y., Chen, H., Chen, X., Guo, J., Liu, Z., Tang, Y., Xiao, A., Xu, C., Xu, Y., Yang, Z., Zhang, Y., and Tao, D. (2023). A survey on vision transformer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(1):87–110.

Hersbach, H., Bell, B., Berrisford, P., Hirahara, S., Horányi, A., Muñoz-Sabater, J., Nicolas, J., Peubey, C., Radu, R., Schepers, D., et al. (2020). The era5 global reanalysis. *Quarterly Journal of the Royal Meteorological Society*, 146(730):1999–2049.

Huisman, M., Van Rijn, J. N., and Plaat, A. (2021). A survey of deep meta-learning. *Artificial Intelligence Review*, 54(6):4483–4541.

Khan, S., Naseer, M., Hayat, M., Zamir, S. W., Khan, F. S., and Shah, M. (2022). Transformers in vision: A survey. *ACM computing surveys (CSUR)*, 54(10s):1–41.

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

Kimura, R. (2002). Numerical weather prediction. *Journal of Wind Engineering and Industrial Aerodynamics*, 90(12-15):1403–1414.

Kondor, R. and Trivedi, S. (2018). On the generalization of equivariance and convolution in neural networks to the action of compact groups. In *International conference on machine learning*, pages 2747–2755. PMLR.

Laloyaux, P., Thépaut, J.-N., and Dee, D. (2016). Impact of scatterometer surface wind data in the ecmwf coupled assimilation system. *Monthly Weather Review*, 144(3):1203–1217.

Lam, R., Sanchez-Gonzalez, A., Willson, M., Wirnsberger, P., Fortunato, M., Alet, F., Ravuri, S., Ewalds, T., Eaton-Rosen, Z., Hu, W., et al. (2023). Learning skillful medium-range global weather forecasting. *Science*, 382(6677):1416–1421.

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

Li, Z., Liu, F., Yang, W., Peng, S., and Zhou, J. (2022). A survey of convolutional neural networks: Analysis, applications, and prospects. *IEEE Transactions on Neural Networks and Learning Systems*, 33(12):6999–7019.

Lima, F. M. S. (2022). Lecture notes on legendre polynomials: their origin and main properties.

Lin, T., Wang, Y., Liu, X., and Qiu, X. (2022). A survey of transformers. *AI open*, 3:111–132.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B. (2021). Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022.

Loshchilov, I. and Hutter, F. (2019). Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Maurício, J., Domingues, I., and Bernardino, J. (2023). Comparing vision transformers and convolutional neural networks for image classification: A literature review. *Applied Sciences*, 13(9):5521.

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

Nikolenko, S. I. (2021). *Synthetic data for deep learning*, volume 174. Springer.

O'shea, K. and Nash, R. (2015). An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*.

Price, I., Sanchez-Gonzalez, A., Alet, F., Ewalds, T., El-Kadi, A., Stott, J., Mohamed, S., Battaglia, P., Lam, R., and Willson, M. (2023). Gencast: Diffusion-based ensemble forecasting for medium-range weather. *arXiv preprint arXiv:2312.15796*.

Quinonero-Candela, J. and Rasmussen, C. E. (2005). A unifying view of sparse approximate gaussian process regression. *The Journal of Machine Learning Research*, 6:1939–1959.

Rasmussen, C. E. (2003). Gaussian processes in machine learning. In *Summer school on machine learning*, pages 63–71. Springer.

Rasp, S., Hoyer, S., Merose, A., Langmore, I., Battaglia, P., Russel, T., Sanchez-Gonzalez, A., Yang, V., Carver, R., Agrawal, S., et al. (2023). Weatherbench 2: A benchmark for the next generation of data-driven global weather models. *arXiv preprint arXiv:2308.15560*.

Reed, S., Chen, Y., Paine, T., Oord, A. v. d., Eslami, S., Rezende, D., Vinyals, O., and de Freitas, N. (2017). Few-shot autoregressive density estimation: Towards learning to learn distributions. *arXiv preprint arXiv:1710.10304*.

Remes, S., Heinonen, M., and Kaski, S. (2017). A mutually-dependent hadamard kernel for modelling latent variable couplings. In *Asian Conference on Machine Learning*, pages 455–470. PMLR.

Rußwurm, M., Klemmer, K., Rolf, E., Zbinden, R., and Tuia, D. (2024). Geographic location encoding with spherical harmonics and sinusoidal representation networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Sarker, I. H. (2021). Deep learning: a comprehensive overview on techniques, taxonomy, applications and research directions. *SN Computer Science*, 2(6):420.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural networks*, 61:85–117.

Sinnott, R. (1984). Computing under the open sky. *Sky and Telescope*, 68(2):158.

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

Vaughan, A., Markou, S., Tebbutt, W., Requeima, J., Bruinsma, W. P., Andersson, T. R., Herzog, M., Lane, N. D., Hosking, J. S., and Turner, R. E. (2024). Aardvark weather: end-to-end data-driven weather forecasting. *arXiv preprint arXiv:2404.00411*.

Vaughan, A., Tebbutt, W., Hosking, J. S., and Turner, R. E. (2022). Convolutional conditional neural processes for local climate downscaling. *Geoscientific Model Development*, 15(1):251–268.

Wang, J. X., Kurth-Nelson, Z., Tirumala, D., Soyer, H., Leibo, J. Z., Munos, R., Blundell, C., Kumaran, D., and Botvinick, M. (2016). Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*.

Williams, C. K. and Rasmussen, C. E. (2006). *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA.

Wong, A. P., Wijffels, S. E., Riser, S. C., Pouliquen, S., Hosoda, S., Roemmich, D., Gilson, J., Johnson, G. C., Martini, K., Murphy, D. J., et al. (2020). Argo data 1999–2019: Two million temperature-salinity profiles and subsurface velocity observations from a global array of profiling floats. *Frontiers in Marine Science*, 7:700.

Yu, K., Tresp, V., and Schwaighofer, A. (2005). Learning gaussian processes from multiple tasks. In *Proceedings of the 22nd international conference on Machine learning*, pages 1012–1019.

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

Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., et al. (2023). A survey of large language models. *arXiv preprint arXiv:2303.18223*.

Zhou, Y., Lei, T., Liu, H., Du, N., Huang, Y., Zhao, V., Dai, A. M., Le, Q. V., Laudon, J., et al. (2022). Mixture-of-experts with expert choice routing. *Advances in Neural Information Processing Systems*, 35:7103–7114.