0.1 Introduction

Neural Process (NP) is a meta-learning framework introduced in [Garnelo, Rosenbaum, et al. 2018; Garnelo, Schwarz, et al. 2018] that is used for few-shot uncertainty aware meta learning. There exists two variants of the Neural Process, the Conditional Neural Process (CNP) and the Latent Neural Process (LNP), we will focus entirely on the CNP for this project and hence we will implicitly refer to CNP as NP.

Neural Processes learn a distribution over the input locations *conditioned on the training data*. In the NP literature we refer the training data as the *context set* and the input locations we want to predict the output for as the *target set*. The model is trained on a meta datasets of context-target pairs by maximizing the likelihood of the target set *conditioned* the context set.

0.2 Architecture

0.2.1 Conditional Neural Processes

The general framework for a CNP requires us to take a context set $\mathcal{C} = \{\mathcal{C}_i\}_{i=1}^{N_c}$ containing input-output pair points $\mathcal{C}_i = (\boldsymbol{x}_i^{(c)}, \boldsymbol{y}_i^{(c)})$ and a target set $\mathcal{T} = \{\boldsymbol{x}_i^{(t)}\}_{i=1}^{N_t}$ containing inputs $\boldsymbol{x}_i^{(t)}$ we want to predict the outputs for.

The data points in the context set $\mathcal{C}_i$ are encoded into an embedding using network as follows:

$$\boldsymbol{r}(\mathcal{C}_i) = \text{Enc}_\theta(\mathcal{C}_i) = \text{Enc}_\theta([\boldsymbol{x}_i^{(c)}, \boldsymbol{y}_i^{(c)}]) \tag{0.2.1}$$

where $\boldsymbol{r}$ is the embedding of the context point $\mathcal{C}_i$ and θ are the parameters of the encoder. The embeddings of the context sets under processing to obtain a global representation of the dataset $\mathcal{C}$ as follows:

$$\boldsymbol{R}(\mathcal{C}) = \text{Process}(\{\boldsymbol{r}(\mathcal{C}_i)\}_{i=1}^{D}) \tag{0.2.2}$$

The 'processing' must be **permutation invariant**, so typically it is a simple summation of the embeddings. The global representation $\boldsymbol{R}$ is then used to condition the decoder to predict the outputs of the target set to giving us a posterior distribution over the outputs $\boldsymbol{y}_i^{(t)}$.

$$p(\boldsymbol{y}_i^{(t)}|\boldsymbol{x}_i^{(t)}, \mathcal{C}) = \text{Dec}_\theta(\boldsymbol{x}_i^{(t)}, \boldsymbol{R}(\mathcal{C})) \tag{0.2.3}$$

The overall architecture for the CNP is shown in Figure 0.2.1.

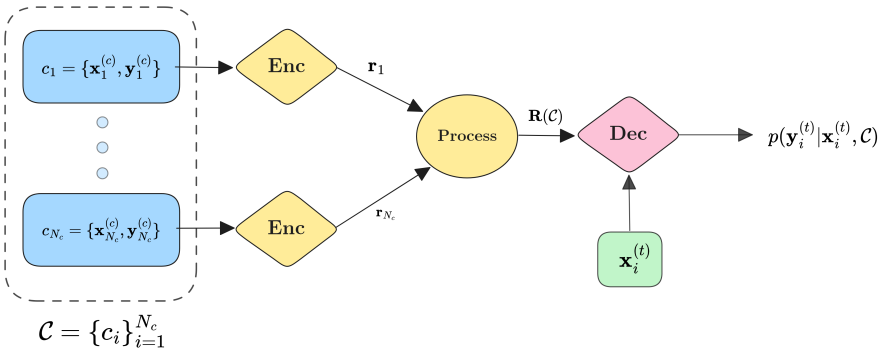

Figure 0.2.1: CNP Architecture: The model vectorizes each individual data point $\mathcal{C}_i$ in the context set $\mathcal{C}$ and then processes them to obtain a global representation $\boldsymbol{R}(\mathcal{C})$ which is then used to condition the decoder to predict a distribution over the target points $\boldsymbol{y}^{(t)}$.

In the original CNP paper, the encoder and decoder are implemented as simple Multi-Layer Perceptrons (MLPs) and the processing is implemented as a mean operation, which is an implementation of the 'DeepSet' architecture [Zaheer et al. 2018].

Importantly, CNPs make the strong assumption that the posterior distribution *factorizes* over the target points:

$$p(\boldsymbol{Y}^{(t)}|\boldsymbol{X}^{(t)}, \mathcal{C}) \overset{(a)}{=} \prod_{i=1}^{N_t} p(\boldsymbol{y_i^{(t)}}|\boldsymbol{x_i^{(t)}}, \boldsymbol{R}(\mathcal{C})) \tag{0.2.4}$$

$$\overset{(b)}{=} \prod_{i=1}^{N_t} \mathcal{N}(\boldsymbol{y_i^{(t)}}|\boldsymbol{\mu_i}, \boldsymbol{\sigma_i^2}) \tag{0.2.5}$$

The factorization assumption (a) allows the model can scale linearly with the number of target points with a tractable likelihood. However, this assumption means **CNPs are unable to generate coherent sample paths, they are only able to produce distributions over the target points.** Furthermore, we need to select a marginal likelihood for the distribution (b) which is usually a Heteroscedastic Gaussian Likelihood (Gaussian with a variance that varies with the input) [Garnelo, Rosenbaum, et al. 2018]. This adds an assumption since likelihood chosen may not be appropriate for the data we are modeling.

The model can be trained using simple maximum likelihood estimation (MLE) by minimizing the negative log-likelihood, giving us the following loss function:

$$\mathcal{L} = \mathbb{E}_{(\mathcal{C},\mathcal{T})}\left[-\sum_{i=1}^{|\mathcal{T}|} \log p(\boldsymbol{y_i^{(t)}}|\boldsymbol{x_i^{(t)}}, \mathcal{C})\right] \tag{0.2.6}$$

0.3 Performance of Vanilla NP

Whilst the CNP using DeepSets is flexible and scalable, in reality it is unable to perform well on more complicated and higher dimensional data as the model is unable to learn a

good representation of the data using a simple MLP and summation operations.

Could we replace the encoder and decoder with more powerful networks? And if so, what would be the best architecture to use?

We aim to answer this by exploring the use of a Convolutional Neural Network (CNN) and a Transformer as encoders of our NP. CNNs and Transformers have been shown to perform well on a variety of tasks and at scale, thus we hypothesize that they will be able to learn a better representation of the context set and improve the performance of the NP. Both are bound to have their unique advantages and disadvantages which we will explore in the following chapters.