0.1 Motivation

Machine learning models have been immensely successful in variety of applications to generate predictions in data-driven domains such as computer vision, robotics, weather forecasting. While the success of these models is undeniable, they tend to lack the ability to understand the uncertainty in the predictions. This is a major drawback in the deployment of these models in real-world applications, for example, in weather forecasting, the uncertainty of the prediction of the weather is arguably as valuable as the prediction itself. In this work, we aim to implement a model that is **uncertainty aware** whilst also possessing further desirable properties.

0.2 Desirable Properties

On top of being uncertainty aware, we would like to insert some desirable inductive biases that help the model to generalize better and be more interpretable. These properties are:

Flexible: *The model should be able to work on a variety of data types.* As long as a data point can be represented as a vector, the model should be able to operate on it. This allows the model to be used in a variety of applications and domains.

Scalable: *The model should be able to learn large datasets and scale to as many inputs.* Which is not the case with many traditional models such as Large Language Models (LLMs) which are usually limited to a max number of tokens. Another aspect of scalability is the ability learn high-dimensional data with good computational efficiency.

Permutation Invariant: *The prediction of the model should not change if the order of the input data is changed.* When each data point contains the information about input and output pairs, the model should not care about the order in which they are fed into the model. For example, in the case of a weather forecasting model, which uses data from multiple weather stations, the model should not care about the order in which the data from the weather stations is fed into the model, thus making the model permutation invariant.

Translation Equivariant: *Shifting the input data by a constant amount should result in a constant shift in the predictions.*

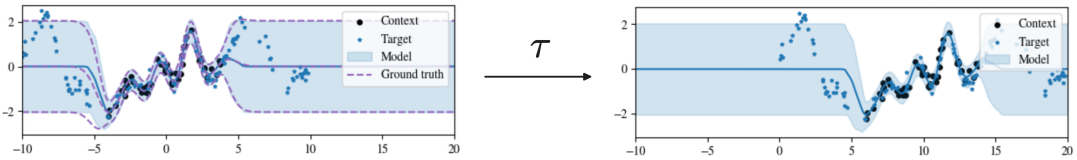

Figure 0.2.1: The Translation Equivariant property on a 1D dataset.

Figure 0.2.1 illustrates this property, when the input data on the left plot is shifted by a constant amount, the prediction should also shift by the same amount (right). Mathematical, a model f is translation equivariant if it satisfies the following property:

$$f : \boldsymbol{x} \rightarrow (\boldsymbol{x}, \hat{\boldsymbol{y}}) \tag{0.2.1}$$
$$f : \boldsymbol{x} + \boldsymbol{\tau} \rightarrow (\boldsymbol{x} + \boldsymbol{\tau}, \hat{\boldsymbol{y}}) \tag{0.2.2}$$

where $\boldsymbol{x}$ is the input and $\hat{\boldsymbol{y}}$ is the output and $\boldsymbol{\tau}$ is a constant shift in the input. Such property allows the model to be more robust and generalize better to unseen data, particularly in the case of stationary data.

Off the Grid Generalization: *The model should be capable of operating on off-the-grid data points.* Off the grid data points are the data points that are not in a regular gridded structure, such as images that have missing pixel values. Traditional models like Convolutional Neural Networks (CNNs) are not able to operate on off-the-grid data points since they require a regular structure to apply the convolution operation. By making the model off-the-grid generalizable, we can create models that can work on many types of datasets and easily handle missing data points. Furthermore, aiding in the performance of the model outside the context data. Applications such as image inpainting can particularly benefit from off-the-grid generalization.

Neural Processes (NPs) [Garnelo et al. 2018] are a class of models that satisfy the above properties. The framework undermining NPs is general purpose, and thus can be modified with a variety of neural network architectures.

0.3 Aims and Objectives

In this work, we aim to implement and compare two different neural network architectures for Neural Processes, the first being based on a Convolutional Neural Network (CNN) called Convolutional Neural Processes (ConvNP) and the second being based on a Transformer architecture called Transformer Neural Processes (TNP). Our objective is to compare the performance of two models on a variety of datasets, focusing on their generalization capabilities and scalability.

We introduce extra inductive biases into the TNP to enhance its ability to generalize. Furthermore, we explore new Transformer architectures that have better computational efficiency compared to the original Transformer architecture.

Our objective is to develop a comprehensive understanding of the properties of these models. We aim to identify the best practices for using these models in different scenarios, highlighting contexts where they perform well and where they do not. This investigation will provide us valuable insights into the capabilities of these models and how they can be used in real-world applications.

Bibliography

Garnelo, Marta, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J. Rezende, S. M. Ali Eslami, and Yee Whye Teh (2018). *Neural Processes*. arXiv: 1807.01622 [cs.LG].