0.1 Transformer Neural Process

An Attention based encoder for the Neural Process was investigated by Kim et al. 2019, although the results were impressive, the model fails to perform at larger scale and 'tends to make overconfident predictions and have poor performance on sequential decision-making problems' [Nguyen and Grover 2023]. Consequently, it is natural to progression to consider Transformer [Vaswani et al. 2017] as a potential candidate that can the performance of the Neural Process. Nguyen and Grover 2023 introduced the Transformer Neural Process (TNP) which uses an *encoder-only* Transformer to learn the relationships between the context and target points via self-attention with appropriate masking.

0.1.1 Model Architecture

Similar to the standard Neural Process architecture, we are required to encode data points within the context set into a vector representation, in language modelling literature we refer to this as tokenization. The tokenization is achieved using a simple Multi-Layer Perceptron (MLP) to encode the data points into tokens with a configurable token dimension, D_{em}.

The TNP tokenizes the target points with padded with zeros to represent the absence of values for the target data points. Subsequently, both context and target tokens are fed into the transformer at the same time, in contrast to the standard NP approach of computing a context representation and then inferencing the target points.

A flag bit is introduced into the tokens to indicate whether the token is a context or target token. For the context dataset $\mathcal{C} = \{(\boldsymbol{x_i}, \boldsymbol{y_i})\}_{i=1}^{N_c}$ and target set $\mathcal{T} = \{(\boldsymbol{x_i})\}_{i=N_c+1}^{N}$, the model will encode a data point into a token $\boldsymbol{t_i}$ as follows:

$$\boldsymbol{t_i} = \begin{cases} \text{MLP}(\texttt{cat}[\boldsymbol{x_i}, 0, \boldsymbol{y_i}]) & \text{if } i \leq N_c \\ \text{MLP}(\texttt{cat}[\boldsymbol{x_i}, 1, \boldsymbol{0}]) & \text{if } i > N_c \end{cases}$$

A flag bit of 0 represents indicate a context token and 1 represents a target token, `cat` is the concatenation operation and $\boldsymbol{0}$ is a vector of zeros of the same dimension as $\boldsymbol{y_i}$.

The tokens are passed into a standard Transformer encoder to learn the relationships between the context and target points. Importantly the Transformer encoder is masked such that the target tokens can only attend to the context tokens and previous target tokens. Alternatively one can perform self attention on the context tokens then cross attention between the context and target tokens giving a more efficient implementation [Feng et al. 2022] (we use this implementation in our experiments).

The output of the Transformer encoder is passed through a Multi-Layer Perceptron (MLP) to generate the mean and variance of the predictive distribution of the target points.

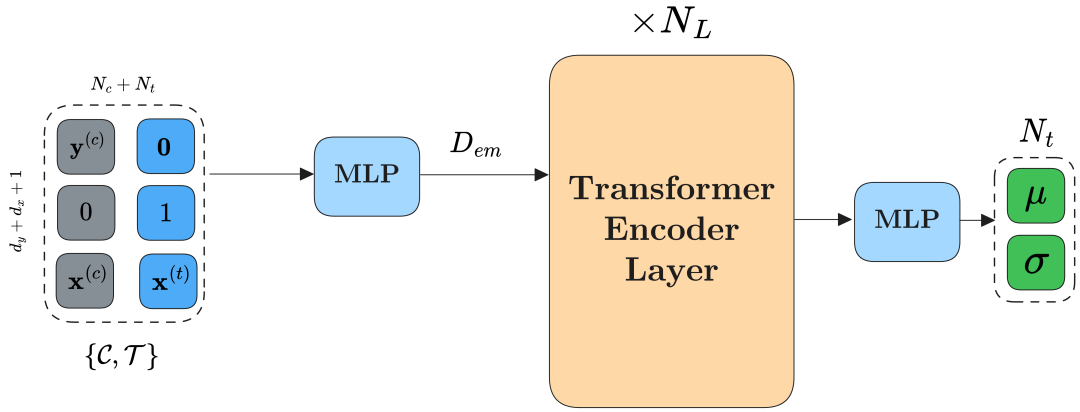

Figure 0.1.1: Vanilla Transformer Neural Process Architecture. All the context and target sets are tokenized and passed through the Transformer Layer. The output of the Transformer Layer is passed through an MLP to generate the mean and variance of the predictive distribution.

0.1.2 Performance

One of the key benefits of Transformers (and attention) is their ability to learn a global view of the data, analogous to an 'infinite receptive field' in Convolutional Neural Networks. This feature enables the model to learn complicated relationships across many length scales in the data. However, this can be a double-edged sword as the model can overfit to the data within its training region and fail to generalize to unseen data.

Translation Equivariance is a key property behind CNNs which allows them to generalize excellently to unseen data by learning features irrespective of their position in the data. Such feature learning is critical to modelling real world data, where the absolute positions of the data are often less significant the *relative positions* between the data points. For instance, in image classification the position of the object in the image does not matter as much as the features that characterize object itself. Transformers lack this property, resulting in unpredictable behaviors when the data is shifted out of context region, as shown in Figure 0.1.2.

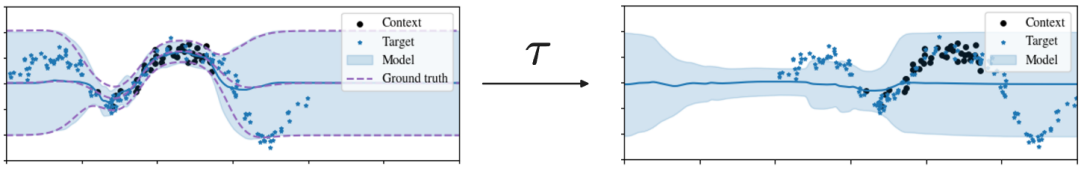

Figure 0.1.2: Vanilla Transformer Neural Process failing to generalize in out of context data. The TNP successfully models the data in the context region (left) but fails to make predictions when the data is shifted (right).

What if we could combine the best of both worlds? Could we create a Translation Equivariant Transformer Neural Process (TETNP) which possesses the global view of the Transformer and the generalization properties of the CNN? This is the question we will investigate in the next section and is the main contribution of this work and Ashman et al. 2024.

0.2 Translation Equivariant TNP

Translation Equivariance requires the model to yield the same output when the input is shifted. Such property must imply that the model only learns the relative distances between the data points ($\boldsymbol{x_i} - \boldsymbol{x_j}$) and *not* the absolute positions of the data points. How could one enforce such a property in the Transformer Neural Process? The solution used is very simple, we add a term to the attention mechanisms in the Transformer to enforce the Translation Equivariance property!

Consider the standard attention mechanism in the Transformer, the attention weights are computed as:

$$\text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}\left(\boldsymbol{E}\right)\boldsymbol{V} \tag{0.2.1}$$

$$\boldsymbol{E_{ij}} = \frac{\boldsymbol{q_i} \cdot \boldsymbol{k_j}}{\sqrt{d_k}} \tag{0.2.2}$$

To create a Translation Equivariant Attention mechanism we add a term to the attention weights which enforces the Translation Equivariance property. The new attention weights are computed with the following equation:

$$\text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}, \boldsymbol{X}) = \text{softmax}\left(\boldsymbol{E} + F(\boldsymbol{\Delta})\right)\boldsymbol{V} \tag{0.2.3}$$

$$\boldsymbol{\Delta_{ij}} = \boldsymbol{x_i} - \boldsymbol{x_j} \tag{0.2.4}$$

where F is some function that introduces non-linearity into the attention weights, which is applied to each entry of the matrix independently ($F(\boldsymbol{\Delta})_{ij} = F(\boldsymbol{\Delta_{ij}})$), we will investigate the effect of different functions in the experiments later on. If all the input locations are shifted by a constant $\boldsymbol{\tau}$, the relative term will remain the same $\boldsymbol{F(\Delta_{ij})} = F(\boldsymbol{x_i} + \boldsymbol{\tau} - \boldsymbol{x_j} - \boldsymbol{\tau}) = F(\boldsymbol{x_i} - \boldsymbol{x_j})$ and the attention weights will remain the same.

Importantly, as we are enforcing the Transformer to learn the relative distances between the data points, we must remove the $\boldsymbol{x}$ values from the tokenization of the data points. Therefore, the tokenization of the data points is given by:

$$\boldsymbol{t_i} = \begin{cases} \text{MLP}(\texttt{cat}[0, \boldsymbol{y_i}]) & \text{if } i \leq N_c \\ \text{MLP}(\texttt{cat}[1, \boldsymbol{0}]) & \text{if } i > N_c \end{cases}$$

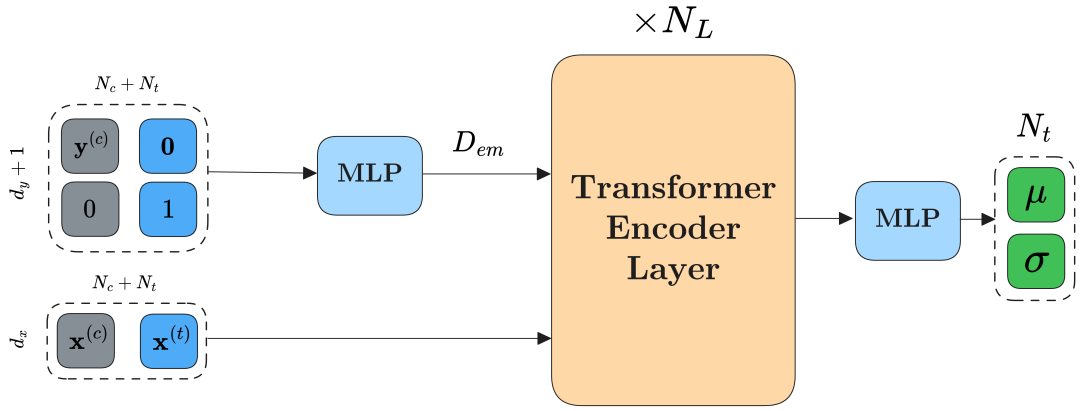

Figure 0.2.1: Translation Equivariant Transformer Neural Process Architecture. All the outputs of the context and target sets are tokenized and passed through the Transformer Layer, whilst the input locations are passed in raw to the attention mechanism to enforce the Translation Equivariance property. The output of the Transformer Layer is passed through an MLP to generate the mean and variance of the predictive distribution.

The removal of the $\boldsymbol{x}$ values from the tokenization allows the tokens to solely encode the $\boldsymbol{y}$ values, thereby reducing the dimensionality of the tokens. These separations the encoding for $\boldsymbol{x}$ and $\boldsymbol{y}$ values could be beneficial for the model, as in the vanilla TNP the model loses distinction between the $\boldsymbol{x}$ and $\boldsymbol{y}$ due to their joint tokenization. We hypothesize that separation of input and output tokens could aid in the learning of the relationships between input and output data points and improve the generalization of the model.

Bibliography

Ashman, Matthew, Cristiana Diaconu, Junhyuck Kim, Lakee Sivaraya, Stratis Markou, James Requeima, Wessel P. Bruinsma, and Richard E. Turner (2024). "Translation-Equivariant Transformer Neural Processes". In: *Forty-first International Conference on Machine Learning*. URL: https://openreview.net/forum?id=pftXzp6Yn3.

Feng, Leo, Hossein Hajimirsadeghi, Yoshua Bengio, and Mohamed Osama Ahmed (2022). "Efficient Queries Transformer Neural Processes". In: *Sixth Workshop on Meta-Learning at the Conference on Neural Information Processing Systems*. URL: https://openreview.net/forum?id=_3FyT_W1DW.

Kim, Hyunjik, Andriy Mnih, Jonathan Schwarz, Marta Garnelo, Ali Eslami, Dan Rosenbaum, Oriol Vinyals, and Yee Whye Teh (2019). *Attentive Neural Processes*. arXiv: 1901.05761 [cs.LG].

Nguyen, Tung and Aditya Grover (2023). *Transformer Neural Processes: Uncertainty-Aware Meta Learning Via Sequence Modeling*. arXiv: 2207.04179 [cs.LG].

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin (2017). *Attention Is All You Need*. arXiv: 1706.03762 [cs.CL].