0.1 Datasets

0.1.1 Gaussian Process

The 2D Gaussian Process is the natural extension of the 1D Gaussian Process described in subsection 0.1.1 where we use the squared exponential kernel. We continue to use the same range of lengthscale across both input dimensions as the 1D Gaussian Process.

0.1.2 Sawtooth

The 2D Sawtooth dataset is the natural extension of the 1D Sawtooth dataset described in subsection 0.1.2. We continue to use the same period T and noise n across both input dimensions as the 1D Sawtooth dataset.

0.1.3 Restricted Sawtooth

The restricted sawtooth limits the 'direction of travel' of the sawtooth function to the line of $x_1 = x_2$ or $x_1 = -x_2$. Under this dataset the model only learn a subset of the 'full sawtooth' function. We can use this probe how well the models can generalize to samples from the full sawtooth function.

0.2 Post or Pre MLP

In our original formulation of the TETNP (section 0.2) we pass the matrix of differences ($\boldsymbol{\Delta}$) between x values through an MLP to apply non-linearities then add it to the dot product attention Equation 0.2.3, whilst this performs well we can also consider applying this non-linearity after combining the dot product attention and the relative attention, this method is called the 'Post MLP'.

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

$$\text{Pre:} \quad \boldsymbol{E_{ij}} = \boldsymbol{D_{ij}} + \text{MLP}(\boldsymbol{\Delta_{ij}}) \tag{0.2.2}$$

$$\text{Post:} \quad \boldsymbol{E_{ij}} = \text{MLP}(\texttt{cat}[\boldsymbol{D_{ij}}, \boldsymbol{\Delta_{ij}}]) \tag{0.2.3}$$

Where

$$\boldsymbol{D_{ij}} = \boldsymbol{q_i} \cdot \boldsymbol{k_j} / \sqrt{d_k} \qquad \boldsymbol{\Delta_{ij}} = \boldsymbol{x_i} - \boldsymbol{x_j} \tag{0.2.4}$$

We will investigate the performance of the TETNP with the 'Post MLP' compared to the original 'Pre MLP'. We choose to use the Sawtooth dataset as it is more difficult to learn than the Gaussian Process dataset and will show the differences between the two models more clearly.

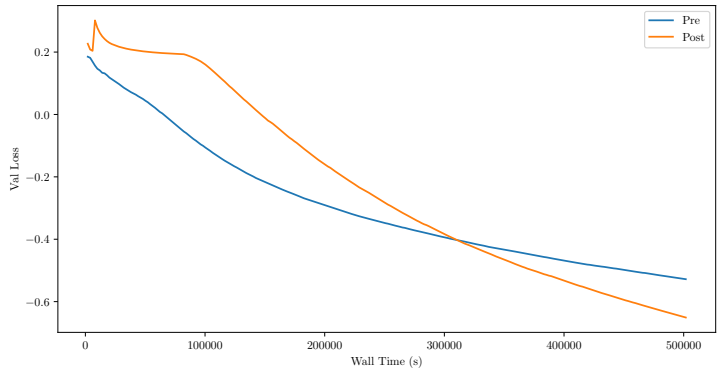


Figure 0.2.1: Validation Loss of TETNP with the 'Post MLP' and 'Pre MLP' on the 2D Sawtooth Dataset. Lower validation loss is better.

The results show that the TETNP with the 'Post MLP' outperforms the TETNP with the 'Pre MLP' by quite a large margin. Trivially the Post function can further refine the dot product attention through the MLP whilst in the 'Pre' function the MLP is *only* applied to the $\mathbf{\Delta}$ matrix. The computational complexity of these two functions are not too different as the MLP are small and applied to the same size matrices.

0.3 ConvNP vs TETNP

We have discovered in the 1D section that the TETNP outperforms the vanilla TNP in all cases, hence for the 2D experiments we will only compare the ConvNP to the TETNP. When performing our experiments we will use models which are both 1 million parameters in size, to ensure a fair comparison.

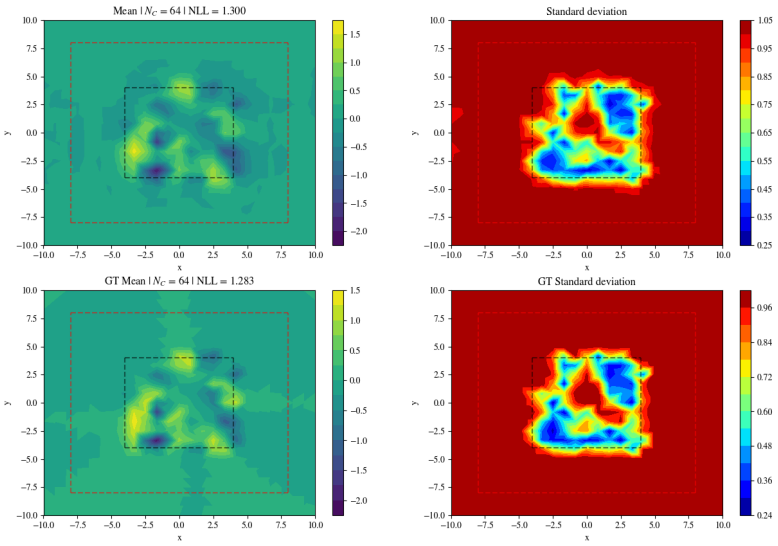
0.3.1 Gaussian Process

As mentioned previously, the Gaussian Process dataset is not very difficult to learn and the ConvNP and TETNP both perform very well on this dataset with the TETNP outperforming the ConvNP by a small margin as shown in Table 0.3.1.

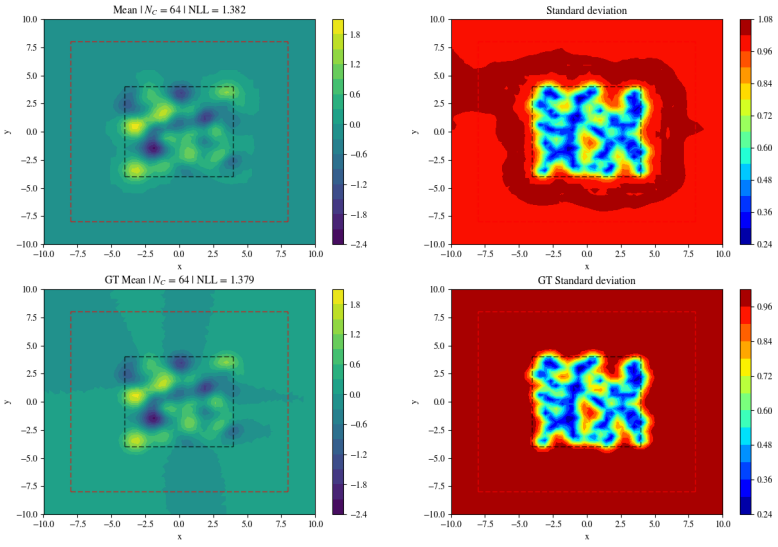
Model	Validation Loss
ConvNP	1.168
TETNP	1.134

Table 0.3.1: Validation Loss of ConvNP and TETNP on the 2D Gaussian Process dataset after training for 4 hours using 1 million parameters models and 64 context points. Lower is better.

Observing the samples from the ConvNP and TETNP for the 2D Gaussian Process dataset Figure 0.3.1, we can see that both models are able to learn the underlying process quite well, with the TETNP being able to perform slightly better than the ConvNP. Due to the smooth nature of the GP, it is fairly easy for both models to learn. To highlight the differences between the ConvNP and TETNP we will move on to the Sawtooth dataset.



(a) ConvNP (top plot is the model prediction and bottom is the ground truth GP)

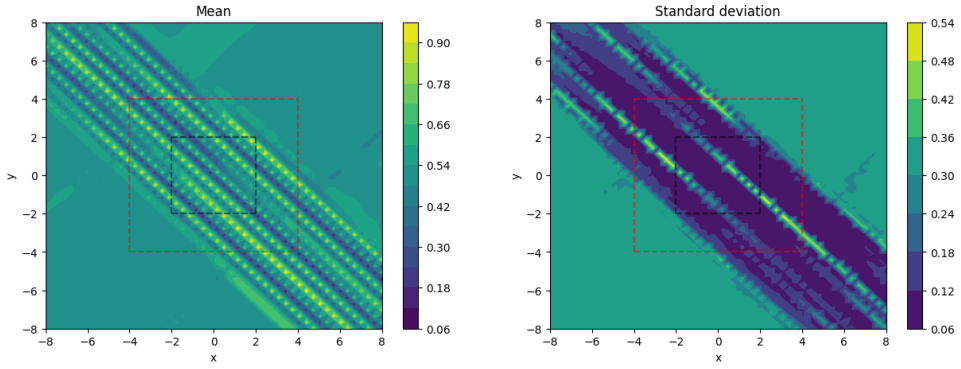


(b) TETNP (top plot is the model prediction and bottom is the ground truth GP)

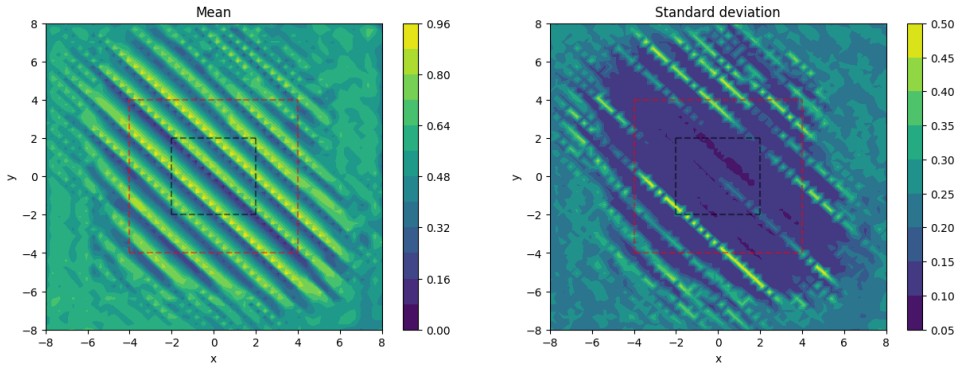
Figure 0.3.1: Samples from ConvNP and TETNP on a frequency 2D Gaussian Process using 64 context points.

0.3.2 Restricted Sawtooth and Rotational Equivariance

As previously stated the restricted sawtooth dataset is a subset of the full sawtooth dataset which restricts the 'direction of travel' of the sawtooth function to the line of $x_1 = x_2$ or $x_1 = -x_2$. Training the models on this dataset will highlight the generalization ability of the models.



(a) TETNP



(b) ConvNP

Figure 0.3.2: Samples from TETNP and ConvNP on the Restricted Sawtooth dataset. The region inside the black dotted box is the region the models were trained on and the region outside the black dotted box is the region the models were not trained on. 64 context points were used.

Figure 0.3.2 shows that the ConvNP performs excellently on this dataset, which is expected as CNNs have filters which learn features and patterns in the data explicitly, thus performs well on extrapolation tasks (outside the black dotted box region). The TETNP on the other hand struggles to generalize fully within the target region (red dotted box) and outside the target region. Instead, it learns to extrapolate the sawtooth along one axis, but not the other. This brings to light a limitation of the Transformer architecture - it has very little interpretability which can result in unexpected behaviour.

Is this TETNP able to generalize to the full sawtooth dataset? To answer this question we will simply run the TETNP on a rotated version of the restricted sawtooth dataset which is the full sawtooth dataset.

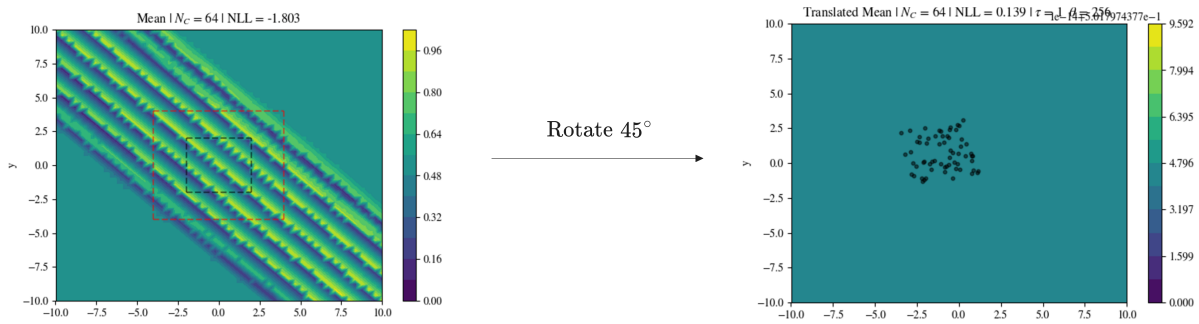


Figure 0.3.3: Generalization of the TETNP. The context points on the left are rotated by 45 degrees and the inference is performed on this rotated set giving us the right plot.

In Figure 0.3.3 we explicitly reduced the target and context region size to allow for the TETNP to cover the full target region. The bottom plot shows the predictions when we rotate the context points by 45 degrees, clearly the TETNP completely fails, it just predicts a constant mean and standard deviation. *Could introducing rotational equivariance to the TETNP help it generalize to the full sawtooth dataset?*

Rotational Equivariance

Introducing rotational equivariance is fairly simple, in our formulation of the Translation Equivariance Attention, we use a matrix $\boldsymbol{\Delta}$ which contains the differences between the $\boldsymbol{x}$ values of all the data points.

$$\text{Not RE}: \quad \boldsymbol{\Delta}_{ij} = \boldsymbol{x}_i - \boldsymbol{x}_j \tag{0.3.1}$$

$$\text{RE}: \quad \boldsymbol{\Delta}_{ij} = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2 \tag{0.3.2}$$

To introduce rotational equivariance we can simply take the L2 norm of the $\boldsymbol{\Delta}$ matrix which will give us the distance between all the data points. Distances are invariant to rotation, hence the TETNP should be rotationally equivariant. Using this new $\boldsymbol{\Delta}$ matrix we can train the TETNP on the restricted sawtooth dataset and see if it can generalize to the full sawtooth dataset.

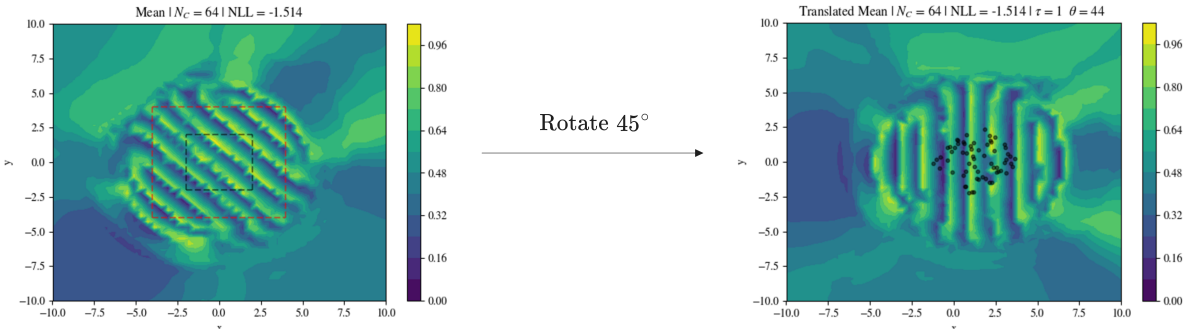


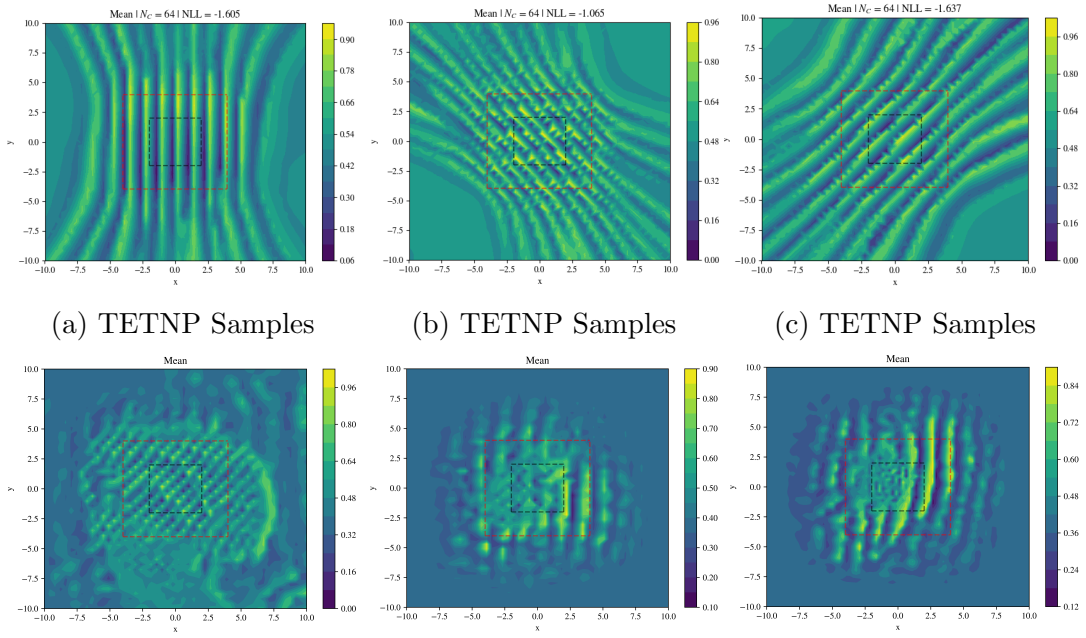
Figure 0.3.4: Generalization of the RE-TETNP. The context points on the left are rotated by 45 degrees and the inference is performed on this rotated set giving us the right plot.

Figure 0.3.4 demonstrates a massive improvement in the TETNP's ability to generalize to the full sawtooth dataset. It has learned to extrapolate the sawtooth in all directions, which is a significant improvement over the non-RE TETNP. Hence, we can conclude that RE is beneficial for the case when the inherent structure of the data is rotationally invariant. This illustrates a massive benefit of the Transformer architecture, as **it is very easy to introduce inductive biases to the Transformer model**, which is not the case for CNNs.

However, as we will see in the next section, if the data given to the model contains samples from many directions, the model will learn to be RE.

0.3.3 Full Sawtooth

The full sawtooth dataset is the sawtooth function which is not restricted to any direction of travel. We observe that the TETNP is able to generalize to the full sawtooth dataset without the need for rotational equivariance Figure 0.3.5.



(a) TETNP Samples (b) TETNP Samples (c) TETNP Samples

(d) ConvNP Samples (e) ConvNP Samples (f) ConvNP Samples

Figure 0.3.5: Samples from TETNP and ConvNP on the full Sawtooth dataset after training for 7 hours using 1 million parameters models and 64 context points. Context region is inside the black dotted box and target region inside the red dotted box. Out of the target region is the extrapolation region. Standard deviation is omitted for clarity.

The ConvNP performs terribly on the full sawtooth dataset, which may seem surprising at first, but it is not since CNNs are not rotationally equivariant. The CNN will need to learn filters for the sawtooth in all directions, which is a very difficult task, especially with limited parameters. The ConvNP was able to perform well on the restricted sawtooth dataset as it only needed to learn filters for the sawtooth in *two directions*. Perhaps with longer training times, the ConvNP could learn to generalize to the full sawtooth dataset, but this is not feasible for this project.

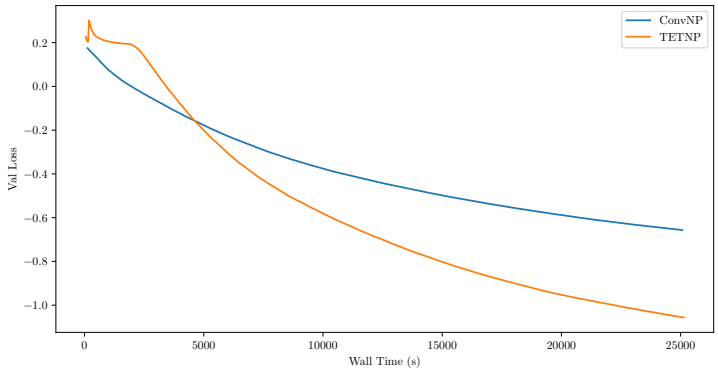


Figure 0.3.6: Validation Loss of ConvNP and TETNP on the 2D Sawtooth Dataset. Lower validation loss is better.

Figure 0.3.6 demonstrates that the TETNP outperforms the ConvNP by a significant margin on the full sawtooth dataset. Overall the TETNP is the clear winner and highlights the power of the Transformer architecture in learning complex patterns in the data.

0.4 Computational Complexity

N_c	N_t	ConvNP Memory (MB)	TETNP Memory (MB)
10	10	24	13
100	10	29	23
1000	10	257	984
5000	10	1273	24082
10	1000	224	26
100	1000	268	113
1000	1000	378	985
5000	1000	1273	24083

Table 0.4.1: Memory usage of ConvNP and TETNP on a 2D dataset with 1 million parameter models. N_c is the number of context points and N_t is the number of target points.

Table 0.4.1 clearly demonstrates that the ConvNPs memory requirement has increased by a factor of 2^3 compared to the 1D case Table 0.4.1, which agrees with the theoretical complexity of the ConvNP. The TETNP retains the same memory requirements as the 1D case, albeit with a significantly higher memory requirement compared to the ConvNP. When scaling systems to higher dimensions, the TETNP is the clear winner in terms of memory requirements as it is fixed with the data dimensionality.