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.

The following plots show some samples from the GP dataset.

TODO

Add plots of GP dataset

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.

The following plots show some samples from the Sawtooth dataset.

TODO

Add plots of Sawtooth dataset

0.1.3 Restricted Sawtooth

By accident when generating the 2D Sawtooth dataset, we ended up restricted the 'direction of travel' of the sawtooth function to the line of $x_1 = x_2$ or $x_1 = -x_2$. This was not intentional but when training both models on this dataset, we found very interesting results. As the models only learn a subset of the 'full sawtooth' function, we can see how well the models can generalize to samples from the full sawtooth function.

The following plots show some samples from the Restricted Sawtooth dataset.

0.2 Post or Pre Relative Attention Function

TODO

Highlight MLP

In our original formulation of the TETNP (section 0.2) we pass the matrix of differences ($\boldsymbol{\Delta}$) between x values through a function F 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 Relative Attention Function'.

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

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

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

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 Relative Attention Function' compared to the original 'Pre Relative Attention Function'. 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.

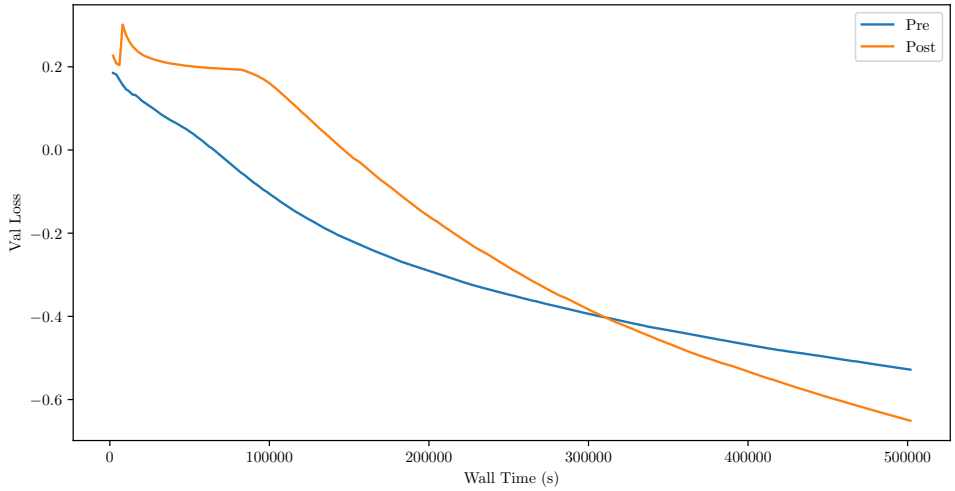


Figure 0.2.1: Validation Loss of TETNP with the 'Post Relative Attention Function' and 'Pre Relative Attention Function' on the 2D Sawtooth Dataset. Lower validation loss is better.

The results show that the TETNP with the 'Post Relative Attention Function' outperforms the TETNP with the 'Pre Relative Attention Function' by quite a large margin. Trivially this makes a lot of sense as 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 $\boldsymbol{\Delta}$ 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.

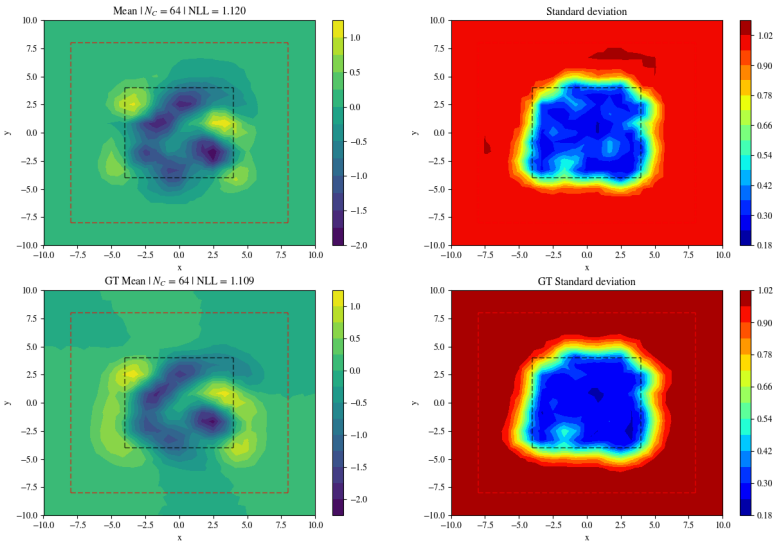
0.3.1 Gaussian Process

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.

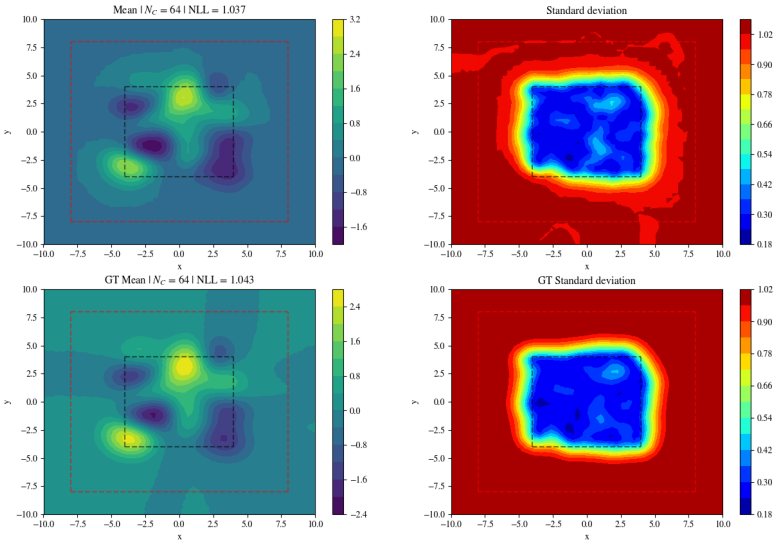
Model	Validation Loss
ConvNP	1.168
TETNP	1.134

Table 0.3.1: Validation Loss of ConvNP and TETNP on the 2D Gaussian Process dataset after training for 3 hours using 1 million parameters models and 64 context points. Lower is better.

Observing the samples from the ConvNP and TETNP for low frequency Figure 0.3.1 and high frequency functions Figure 0.3.2, we can see both models are able to generate very similar predictions to the ground truth GP. The TETNP performs slightly better than the ConvNP in the low frequency case, but the ConvNP performs better in the high frequency case for these samples. However overall the TETNP performs better than the ConvNP on the Gaussian Process dataset.

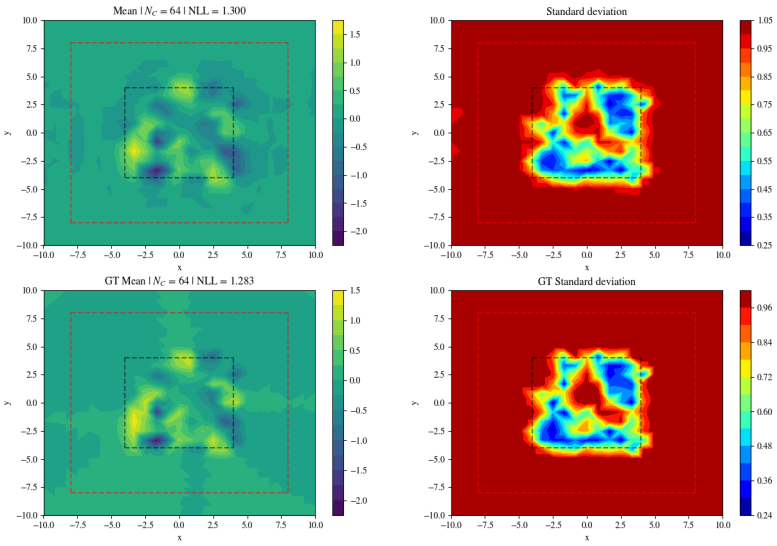


(a) ConvNP (top plot is the model prediction and bottom is the ground truth GP)

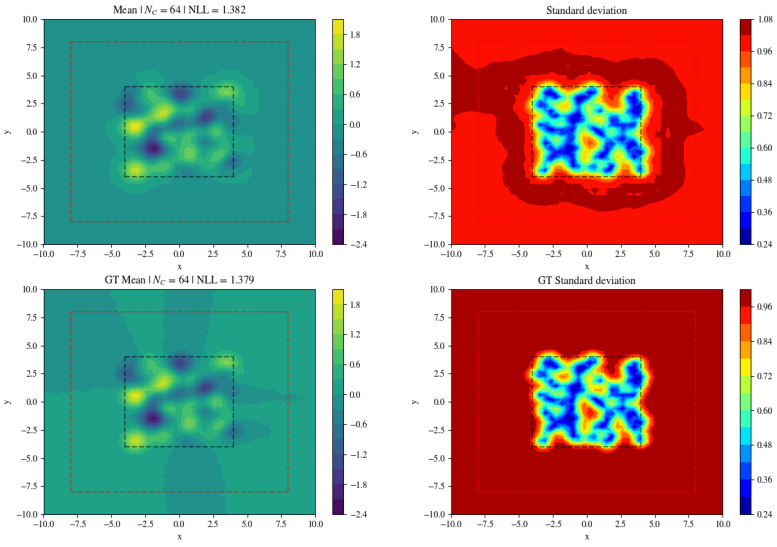


(b) TETNP (top plot is the model prediction and bottom is the ground truth GP)

Figure 0.3.1: Samples from ConvNP and TETNP on a low frequency 2D Gaussian Process.



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

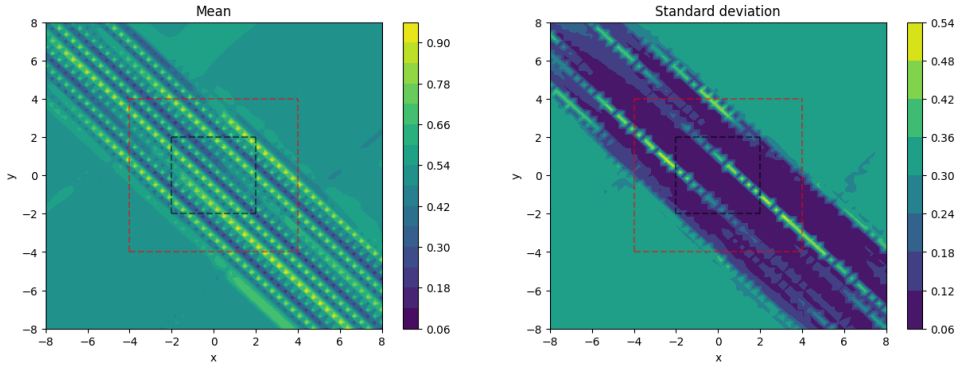


(b) TETNP (top plot is the model prediction and bottom is the ground truth GP)

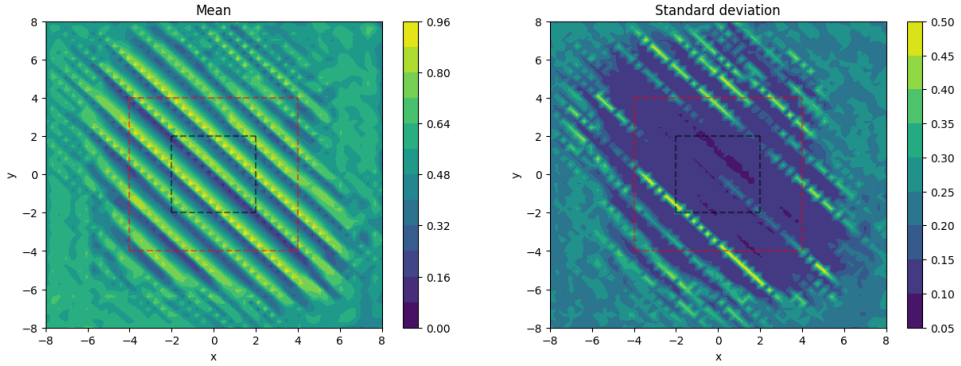
Figure 0.3.2: Samples from ConvNP and TETNP on a high frequency 2D Gaussian Process.

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 $x_1 = -x_2$. By training both models on this dataset we can see how well the models can generalize.



(a) TETNP



(b) ConvNP

Figure 0.3.3: 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.

Figure 0.3.3 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 clearly highlights a limitation of the Transformer architecture which has very little interpretability hence producing weird results.

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.

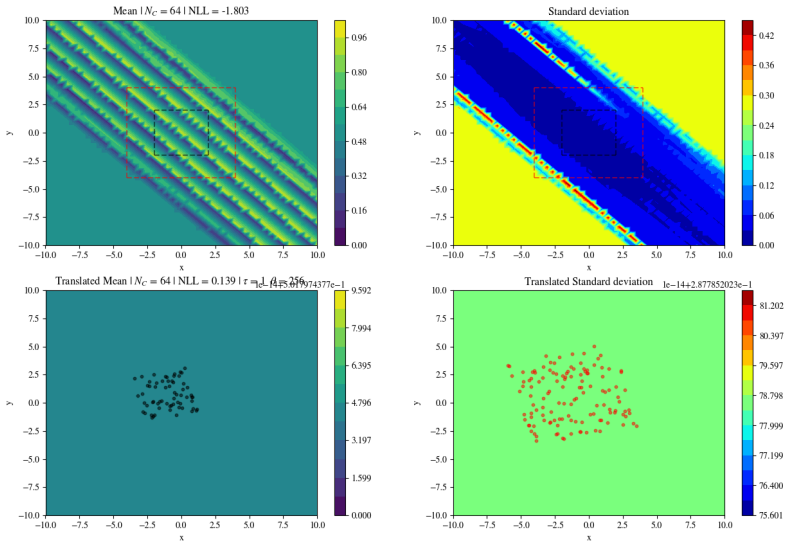


Figure 0.3.4: Samples from TETNP on the full Sawtooth dataset. The top plot is the model prediction on the restricted sawtooth dataset and the bottom plot is the model prediction on the full sawtooth dataset which is a rotated version of the restricted sawtooth dataset. The region inside the black dotted box is the region the model was trained on and the region outside the black dotted box is the region the model was not trained on.

In Figure 0.3.4 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 256 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

To introduce rotational equivariance is fairly simple, in our formulation of the Translation Equivariance Attention, we use a matrix $\boldsymbol{\Delta}$ which is the difference 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.

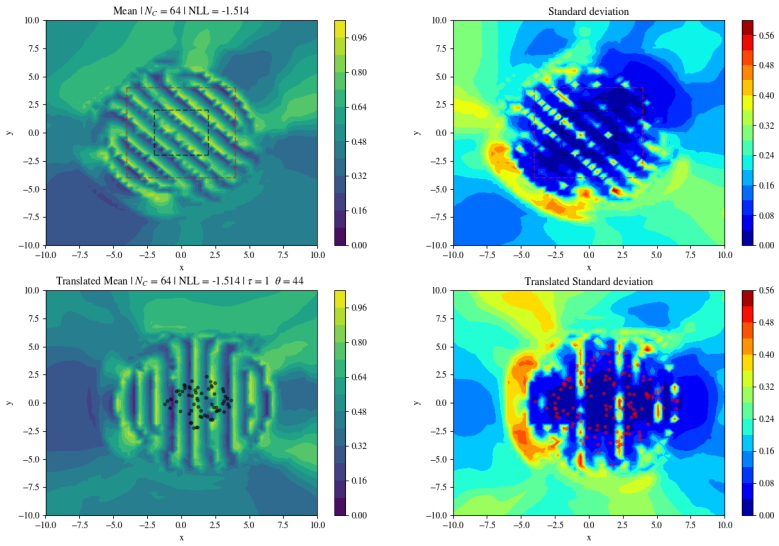


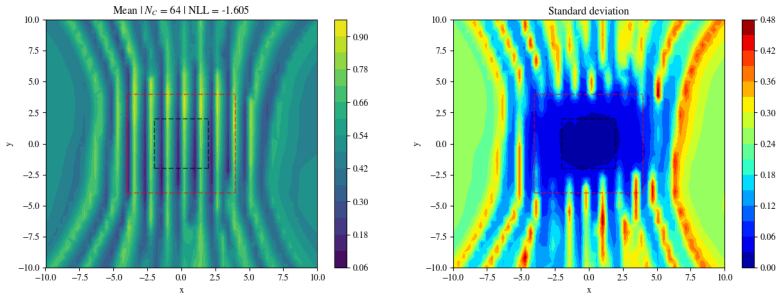
Figure 0.3.5: Samples from TETNP on the full Sawtooth dataset with Rotational The region inside the black dotted box is the region the model was trained on and the region outside the black dotted box is the region the model was not trained on.

Figure 0.3.5 demonstrates a massive improvement in the TETNP's ability to generalize to the full sawtooth dataset. 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.

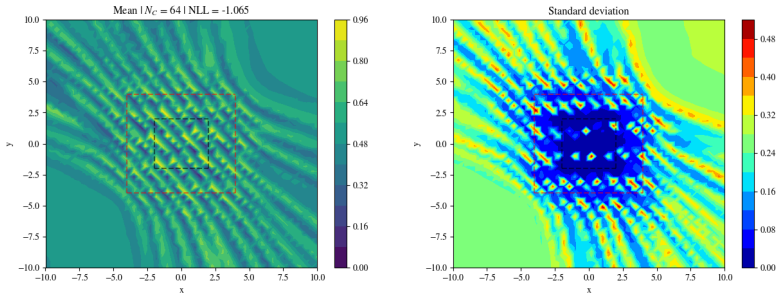
However, as we will see in the next section, if the data given to the model contains samples from many directions, the model will learn to be RE.

0.3.3 Full Sawtooth

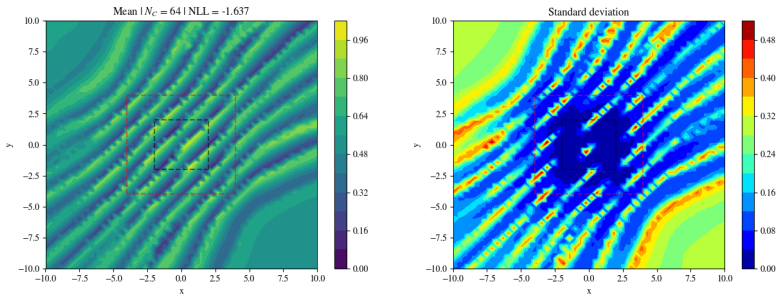
The full sawtooth dataset is the full sawtooth function which is not restricted to any direction of travel. We observe that the TETNP is able to generalize to the full sawtooth dataset without the need for rotational equivariance Figure 0.3.6.



(a)



(b)



(c)

Figure 0.3.6: Samples from TETNP on the full Sawtooth dataset. The region inside the black dotted box is the region the model was trained on and the region outside the black dotted box is the region the model was not trained on.

TODO

GET PLOTS OF CONVNP ON FULL SAWTOOTH AND COMPARE RESULTS!!!!!!!!!!!!
!
!
!
!

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	268	26
100	1000	268	113
1000	1000	268	985
5000	1000	1273	24083

Table 0.3.2: Memory usage of ConvNP and TETNP on the 2D Sawtooth dataset.

0.4 Computational Complexity

TODO

Run the code for this
!
!
!
!