Spectro-Temporal Attention (STA)

Idan Shtark, Yarin Bar

Supervisors: Aviv A. Rosenberg, Yonantan Elul

Abstract

The "Attention" mechanism revolutionized the field of sequential data analysis with neural networks (NN). Inspired by the Attention mechanism, the Spectro-Temporal Attention (STA) aspires to explore the connection between the temporal and the spectral data. Although the STA excels in pattern recognition in periodic signals, in its current form it is susceptible to be misled by noise and pseudo-periodic signals. In this project, we will compare the original STA implementation to our own variation of the STA where we visualize the signal as a continuous 1D vector as opposed to a 2D matrix in the original work. Our modification proved itself effective in denoising both periodic and pseudo-periodic synthetic signals while outperforming the original implementation in most tasks.

Introduction

In recent years, the attention mechanism became a vital component in sequence modeling tasks such as translation, text generation and more. The "Transformer" which was discussed in the paper "Attention is All You Need" [1], offered a novel approach that abstains of using recurrence and relying mainly on the attention mechanism, thus significantly alleviating long term memory usage and allows parallelization.

Using the temporal data alone, although meaningful, might overlook key relations between the temporal data and the underlying spectral information. The "Spectro-Temporal Attention" is a new architecture suggested by Aviv A. Rosenberg and Yonantan Elul in their research proposal [2]. The STA harnesses the existing attention mechanism to use on the spectral information of the time series to create a powerful and robust sequence to sequence models.

The proposed mechanism will preprocess each input and extract the spectral sequence using the Fourier transformation (Welch transformation specifically) on the temporal sequence. Then, the mechanism builds an attention mask to use on the temporal input using the spectral data and vice versa.

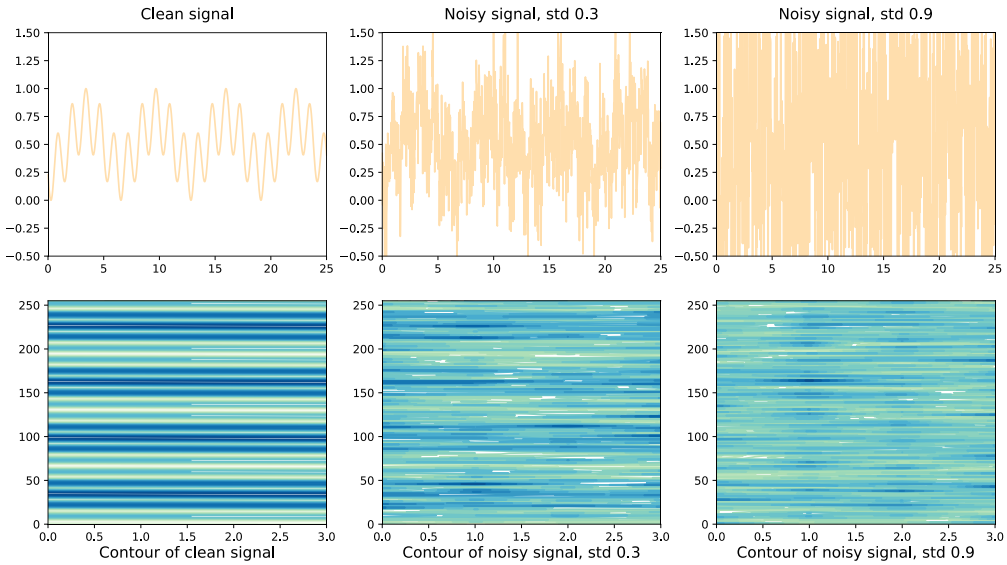

Figure 1: A comparison between a clean synthetic cyclic signal and the same signal with noise (gaussian noise with 0 mean and variable std). The top row it a visualization of the signal and the bottom row is the contour map of the tensor containing the signal. The tensor is generated by slicing the signal every 2π to align the clean signal perfectly. It is apparent that the increased noise undoubtedly impairs the ability to analyze the tensor effectively.

The original STA implementation uses the input as a 2D matrix where each column is a different interval of the same signal. This approach relies on clever segmentation of the data into equally long sections. It can be extremely useful when a convenient segmentation is known and the data is relatively clean. For example, segmenting the $\sin(x)$ function into 2π long sequences aligns each segment perfectly with all other segments. However, real-world data is rarely perfectly periodic and with noise and no prior knowledge it is almost impossible to segment the signal in a meaningful way.

As seen in figure 1, with higher noise the clear pattern that is present in the clean signal is progressively less apparent. This phenomenon gets worse when the data is not perfectly periodic but only pseudo-periodic, as seen in figure 2.

For the task of denoising, we must take into consideration the possible loss of the clean patterns that are created using an entirely periodic function. To remedy this, we suggest a simple way to handle the data; each signal would be processed as a 1D signal. This approach avoids looking into future intervals and focuses on the current state of the input. It can prove useful when the data is noisy, and the future intervals are not perfectly aligned.

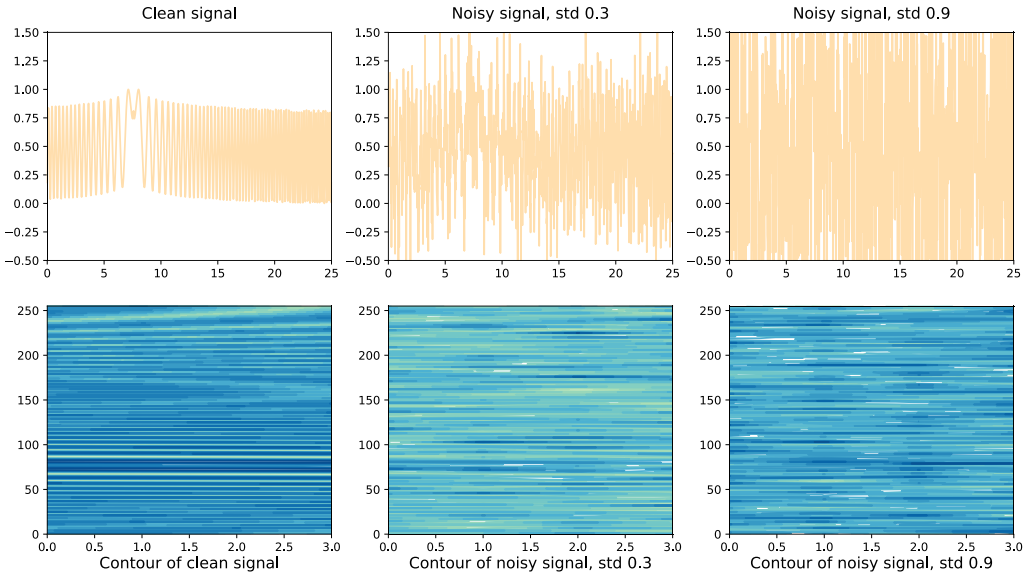

Figure 2: When the signal is pseudo-periodic the input matrix loses its clean, horizontal lines and becomes much harder to clean as a result. At std $= 0.9$ the contour looks much like the pure periodic signal which might fool the denoiser.

Methods

In this work, we are going to compare the denoising capabilities of two different implementations of the STA; the original implementation created by Aviv A. Rosenberg and Yonantan Elul and our own implementation that its changes we are going to review after we explain the STA architecture in detail.

The first step when the input signal is in hand, is to create the input embeddings. Assuming the inputs of the STA are X, S representing the temporal and spectral signals respectively, the attention masks of the signals are derived from:

$$X_e = W_{1,x}\boldsymbol{S}$$
$$S_e = W_{1,s}\boldsymbol{X}$$

Where X_e is the base for the temporal attention mask in the temporal path, using the spectral data and S_e is the base for the spectral attention mask in the spectral path using the temporal data. We can now define our attention wights to be:

$$A_x = \text{softmax}(X_e^T X_e) = \text{softmax}\left(S^T W_{1,x}^T W_{1,x} S\right)$$
$$A_s = \text{softmax}(S_e^T S_e) = \text{softmax}\left(X^T W_{1,s}^T W_{1,s} X\right)$$

By Vaswani et al. [1] terms, by computing A_x, A_s we have computed the *keys* and *queries* product and to complete the attention process we multiply the matrices X_e, S_e which represent the *value* with A_x, A_s respectively:

$$X_a = X_e A_x$$
$$S_a = S_e A_s$$

At this point, we arrive at the feed-forward layer which simply applies weights to the input. This step will yield the attention masks:

$$M'_x = W_{2,x} X'_a$$
$$M'_s = W_{2,s} S'_a$$

Note that X'_a, S'_a are reshaped to reduce the number of learnable parameters and therefore reduce the learning time needed. The final masks M_x, M_s are computed by reshaping M'_x, M'_s back to the original dimensions.

It is important to note that up to this point, the usage of X and S is merely a label and we yet to mix the spectral input with the temporal input. In the following step we do exactly that using pointwise multiplication:

$$X_m = M_x \odot X$$
$$S_m = M_s \odot S$$

After obtaining the two masked sequences we want the STA to learn a new embedding for both domains. That is achieved by concatenating the masked and the original input. Then, positional encoding is added to the spectral information. Lastly, we compute the final embeddings using a channel-preserving convolution layer.

Based on the proposed architecture, we are going to test two versions of the STA. The original implementation and our implementation. We are going to focus on the initial embeddings

The original implementation transforms the input into a matrix. The number of rows is the number of samples in each interval – denoted by K and the number of columns is the number of intervals in each example – denoted by B. Thus, if we consider the numbers of channels L, each example is a $L \times K \times B$ tensor. Since the model handles 3D tensors, the original STA uses 2D convolution layers to extract the embeddings of the sequence.

Due to the concerns of poor representation of noisy and pseudo-periodic data discussed in the introduction, we decided to modify the first encoder of the STA. In this variation, the STA receives each channel as a flat 1D sequence. Stacked on top of each other, all the channels create a 2D tensor with dimensions $L \times KB$, instead of a 3D tensor in the previous approach. Since the data is now flat, we used two 1D convolution layers to encode the input. One that enriches the information by increasing the number of channels, then a convolution layer that preserves the enriched channels and finally a convolution layer to reduce the number of channels back to the original number.

In this approach we hope our encoding system can pick up some patterns without "looking into the future", as we cannot rely on a constant segmentation system nor the existence of visible recurrence in the received signals.

The data that is being used in this project is all synthetic and was generated using a group of periodic functions. Each example is a linear combination of these cyclic functions. Given enough function and an adequate range for the coefficients, we can generate millions of unique examples to train on. After the examples were generated, gaussian noise is added to the clean signal to create the final training example.

In this project we explore the performance of each approach using both pure-periodic signals and pseudo-periodic signals to evaluate the potential of the denoisers. After training, we use three main evaluation methods:

1. **Stronger Noise** – the data was generated with the original set of function but with higher intensities of noise. Succeeding to clean noisier input requires the models to generalize the function set used and effectively find the underlaying pattern while ignoring the irrelevant information.

2. **Enriched Set of Functions** – the data was generated by the original group of function used for training with additional functions that were not present before. This method checks if the models can infer new patterns if the signal contains a familiar pattern.

3. **Different Frequencies** – in this method the function set used for training changes completely by changing the frequencies in each function. For instance, if $\sin(x)$ was used in training then in this part we test the model using $\sin(1.5x)$. Although this method is essentially testing the models of a brand-new set of functions, the changes made to the functions are relatively small, therefore making inference plausible.

Implementation

Architecture

Considering the STA as a sophisticated embedder, we used the embedded output as an input for a simple autoencoder. Each model created and tested was built according to the architecture shown in Figure 3, where the difference between each model lays inside the components themselves.

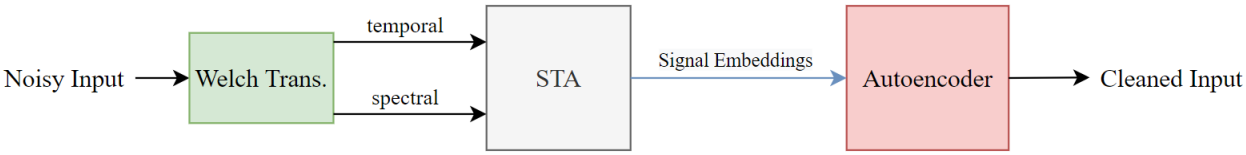

Figure 3: The spectral information is extracted from the noisy temporal sequence using the Welch method. Then both the temporal and the spectral are sent into the STA where the self-attention mechanism helps create the signal embeddings. The embeddings in turn, go to the autoencoder and out as cleaned signals.

The autoencoder we used in this project is comprised of fully connected layers and utilizes a well-known technique that "inflates" the input to extract more information before encoding it into the latent space. The general architecture used in the autoencoder is demonstrated in figure 4.

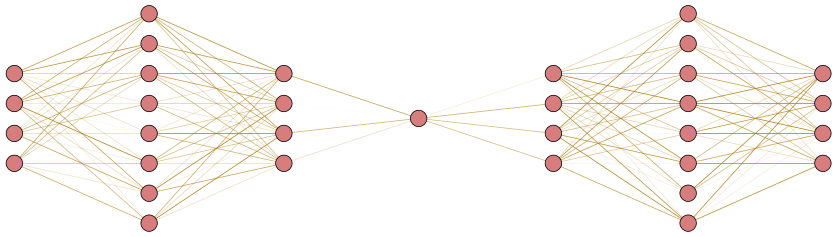

Figure 4: The first part is the encoder which takes the input sequence, inflates it and promptly deflates it into the latent space. Then the decoder inflates it back to the original size.

To examine the performance of the denoisers we used 4 models:

1. **Our STA Linear** – Uses our implementation of the STA, where it uses the input as a 2D matrix where each row is a different channel. The autoencoder does not have activation functions making it a linear autoencoder.
2. **Our STA Non-Linear** - Uses our implementation of the STA and after each linear layer in the autoencoder there is a $\tanh(x)$ activation function, where x is the output of the linear layer.
3. **Ref STA Linear** – In this model the STA is the reference implementation that uses the input as a 3D tensor. Then the output is sent into a linear autoencoder as described above.

4. **Ref STA Non-Linear** – Uses the reference STA implementation and a non-linear autoencoder as described.

Data

To generate the data, we use a function set F. Each example $x \in \mathbb{R}^{L\times K\cdot B}$ is a linear combination of these $|F|$ functions as follows:

$$x = \sum_{i=0}^{|F|} \alpha_i \cdot f_i(x+\beta), \quad \alpha_i \in (-1,1), \quad \beta \in (-10,10)$$

Where α_i is the coefficient of the function f_i and β is an arbitrary sample offset. After synthesizing the examples, we normalize each example:

$$x_{\text{normed}} = \frac{x - \min(x)}{\max(x) + \varepsilon} \quad \varepsilon > 0$$

Where $\min(x), \max(x)$ denote the minimal and maximal values respectively in the example x. After the normalization is completed, the clean data is ready:

$$X_{\text{clean}} = \begin{bmatrix} x^1_{\text{clean}} \\ \vdots \\ x^n_{\text{clean}} \end{bmatrix}$$

To obtain the noisy data, we add gaussian noise to X_{clean}:

$$X_{\text{noisy}} = X_{\text{clean}} + \mathcal{N}(\mu, \sigma^2)$$

In our project, we used two training sets:

- Pure Periodic – All of the functions are purely periodic
- Pseudo Periodic – The functions used are close to being periodic but not perfectly, some have a slightly increasing period time and some have a decreasing period time (i.e. $\sin(|x|^{1.1}), \sin(|x|^{0.9})$)

Evaluation and Hyper-Parameters

For both training sets we used $L = 1,\ K = 256,\ B = 4$ as an input dimensions and $\mu = 0,\ \sigma^2 = 0.1$ to generate the noise. The only variation between the training sets is the function set used.

In the pure-periodic test set we used:

$$F = \{\sin(x), \sin(5x), \sin(10x), \sin(15x), \sin(20x), \sin(30x)\}$$

And for the pseudo-periodic training we used:

$$F = \{\sin(|x|^{1.3}), \sin(5|x|^{0.8}), \sin(10|x|^{1.2}), \sin(15|x|^{1.1}), \sin(20|x|^{0.7}), \sin(30|x|^{0.85})\}$$

To isolate the implementation differences between our architecture and the reference architecture, we used the same AE architecture in the denoising models. As a result, all AE have latent space dimensionality of 16.

Since the success of the models lays in their ability to restore the noisy data as close as possible to the clean original signal, we chose to evaluate the performance of the models with the "Mean Square Error" (MSE) loss function in pair with the SGD optimizer. We trained all models with $\mathrm{lr} = 0.05$ and $\mathrm{momentum} = 0.9$ for 300 epochs using batch size of 32.

Experiments

The two different training sets produced two groups of models, where in each group there are 4 different architectures as reviewed above. Both groups were tested on all three methods of assessment described in the "Method" section (stronger noise, enriched function set and different frequencies).

In each experiment we make a specific change to the dataset to check the capability of the denoisers:

1. **Stronger Noise** – while in training we use standard deviation $\sigma^2 = 0.1$, during testing we are attempting to clean a signal with $\sigma^2 = 0.3$ and $\sigma^2 = 0.9$ which is significantly harder.

2. **Enriched Function Set** – we added the following functions (with no change in noise - $\mu = 0, \sigma^2 = 0.1$):

$$G = \left\{e^{\sin(x)}, \sin(x) \cdot \cos(x), \sin(cos(x))\right\}$$
$$F_{\text{enriched}} = F \cup G$$

   The enriched dataset creates a bigger function variation and introduces new patterns which were never learned by the models.

3. **Different Frequencies** – in this experiment we modified the frequency of each function. For example, if trained on $\sin(x)$, in this part we test the model on $\sin(1.25x)$. In this test we changed the frequency only by a small amount to test if the model can pick up slightly different frequencies and adapt.

Since the distance between the denoised signal and the clean signal is an effective measurement of the models' performance, we can conveniently use the MSE loss function to evaluate the models.

Results and Discussion

First, we will look at the result of the models that were trained and tested on the pure-periodic dataset. The values in each column is the L2 distance from the clean signal.

Trained on Pure-Periodic

Experiment	Architecture	Noisy Signal	Our STA	Ref STA
Stronger Noise	std 0.3 Linear	0.0899	0.00116	0.00290
	std 0.3 Non-Linear		0.00253	0.00519
	std 0.9 Linear	0.8093	0.01044	0.01441
	std 0.9 Non-Linear		0.01336	0.03949
Enriched Function Set	Linear	0.010	0.00326	0.00444
	Non-Linear		0.00449	0.00656
Different Frequencies	Linear	0.010	0.03395	0.03648
	Non-Linear		0.03490	0.03946

It is apparent that our architecture surpassed the reference architecture in all tests. Most notably, the "Stronger Noise" experiment generated impressive results with up to 775% cleaner output. In the "Different Frequencies" test however, the noisy signal was closer to the original than any of the models – in this experiment the models corrupted the data.

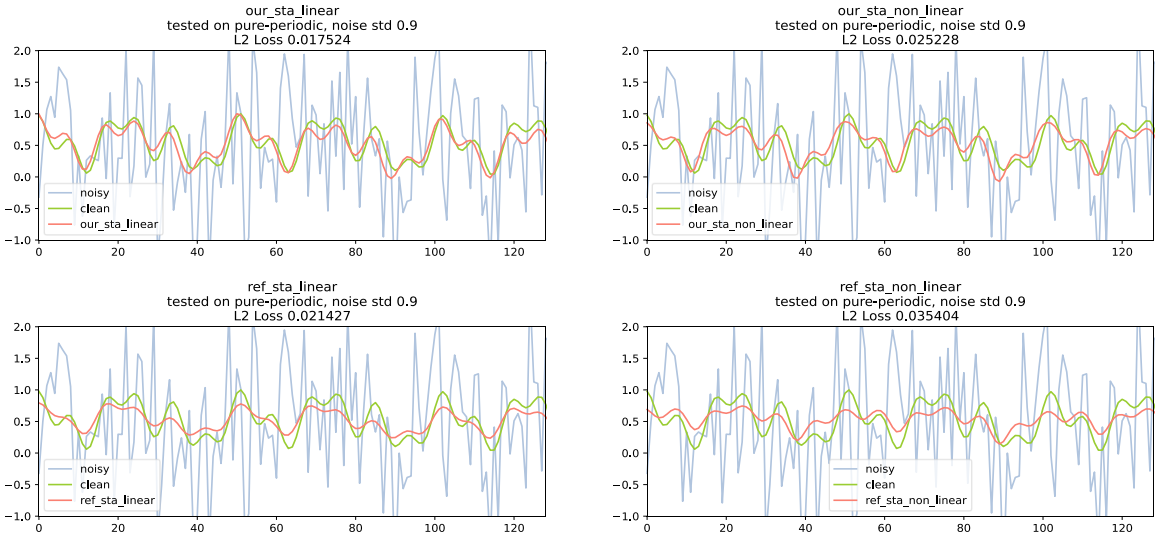

Figure 5: Performance of the 4 models (trained on pure-periodic functions) on a highly noisy example. L2 in the title indicates the distance between the specific example and clean signal, whereas in the table the loss is averaged between all test examples.

It was not anticipated that our architecture would outperform the original architecture on pure-periodic signals, considering figure 1. One explanation might be that high noise degrades the information to a degree where looking into future intervals is no longer a reliable approach. On the other hand, a 1D convolution only traverses locally through the data averting from analyzing contaminated information of future intervals.

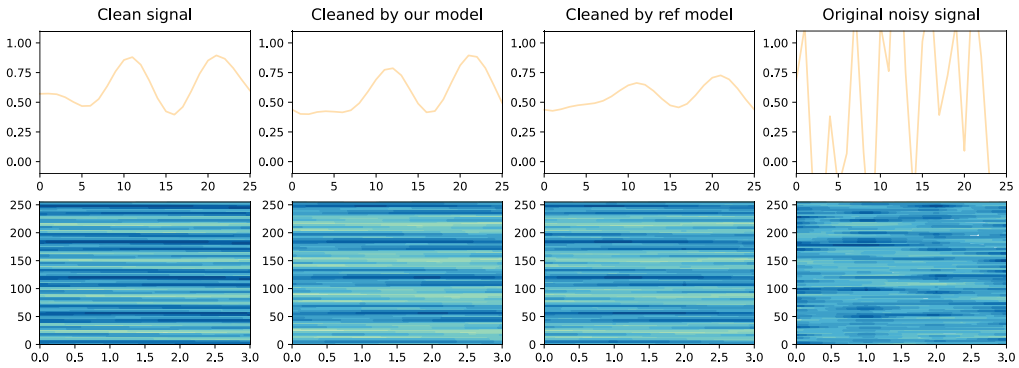

Figure 6: Denoising comparison between our & reference linear models on stronger noise dataset with $\sigma^2 = 0.9$.

In the "Enriched Function Set" experiment the performance of the models was considerably lower than the "Stronger Noise" experiment. Yet interestingly, the denoisers were still extremely effective with up to 300% cleaner output. We believe this accomplishment was possible because the new functions only added minute changes to the waveforms allowing the models to adapt well.

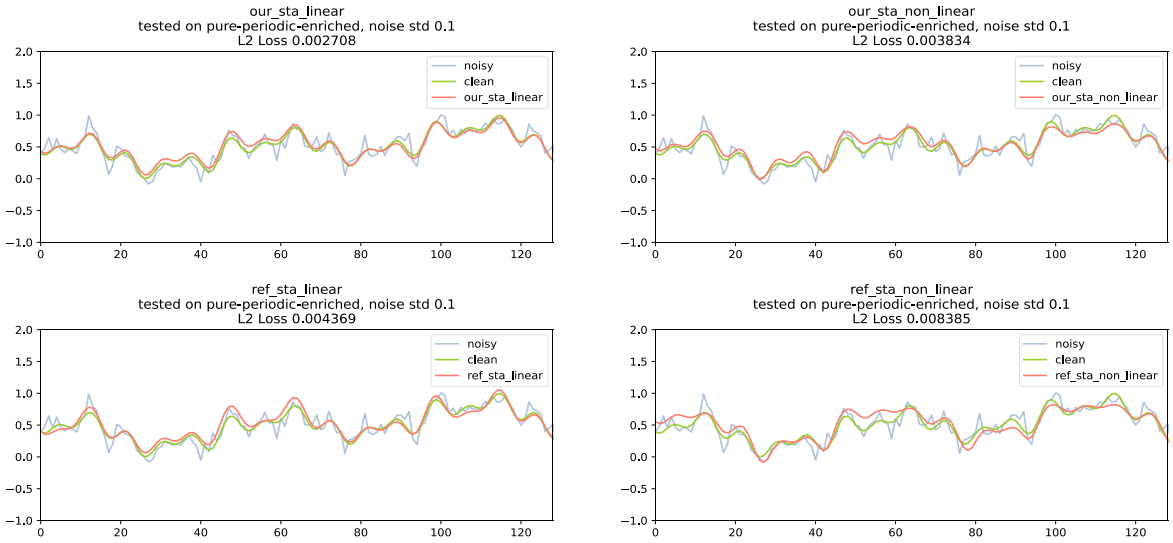

Figure 7: Denoising comparison between our & reference linear models (trained on pure-periodic functions) on enriched function dataset with $\sigma^2 = 0.1$.

Next, we will review the models that were trained on pseudo-periodic dataset. Naturally, in this experiment we expect to see lower success in denoising as the patterns are much harder to recognize and they vary according to the sample offset β.

Trained on Pseudo-Periodic

Experiment	Architecture	Noisy Signal	Our STA	Ref STA
Stronger Noise	std 0.3 Linear	0.0898	0.02682	0.02886
	std 0.3 Non-Linear		0.02714	0.02854
	std 0.9 Linear	0.8089	0.03952	0.03404
	std 0.9 Non-Linear		0.03686	0.03382
Enriched Function Set	Linear	0.010	0.02387	0.02624
	Non-Linear		0.02404	0.02606
Different Frequencies	Linear	0.010	0.02550	0.02750
	Non-Linear		0.02563	0.02753

As anticipated, the models that were trained on pseudo-periodic signals were significantly less successful in denoising. In fact, in 2 of the 3 experiments conducted the models have worsened the accuracy of the output signal.

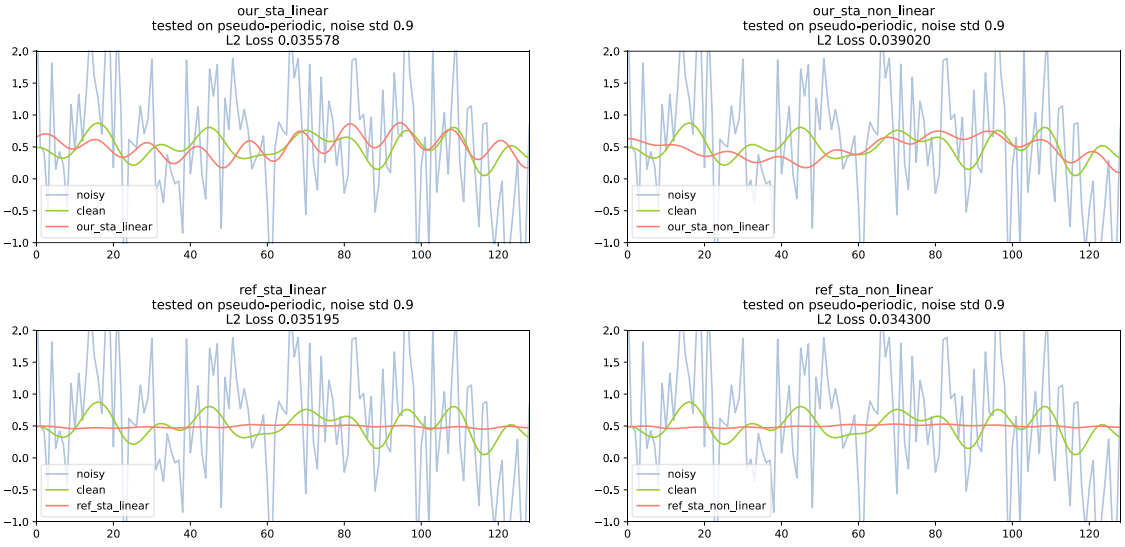

Figure 8: Denoising comparison between our & reference linear models (trained on pseudo-periodic functions) on enriched function dataset with $\sigma^2 = 0.9$.

Although the L2 distance is shorter in the reference output, we can see a lot of information has been lost during the denoising and the model has essentially learned to transform the input into an almost constant function around the mean of the waveform. In this regard, we believe our model has yet again proven to be an effective tool to extract the patterns from a noisy signal.

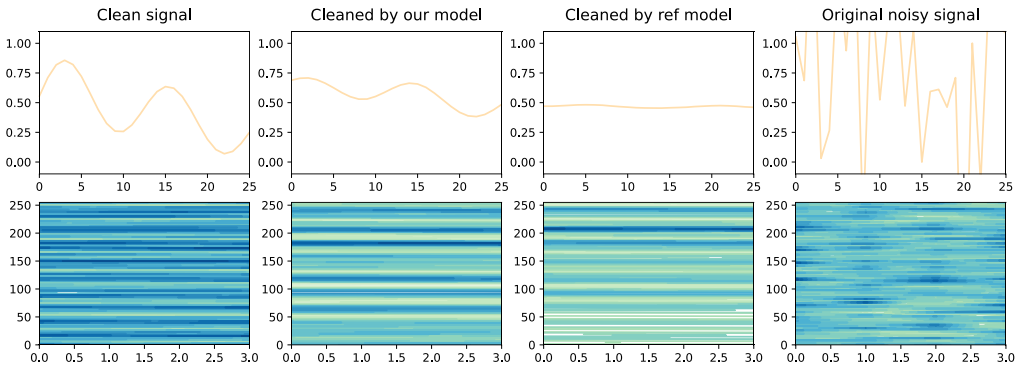

Figure 9: Denoising comparison between our & reference linear models on stronger noise dataset with $\sigma^2 = 0.9$.

When studying at the contour in figure 9 however, it is evident that both models have picked up similar patterns. When plotting the contour, the input is being normalized thus the slight differences are being amplified. That suggests that if post-processed, the output from the reference model can be a good representation of the original waveform too.

In conclusion, after examining the shape of noisy periodic signals, we argued that a real-world signal is harder to segment and much less likely to be perfectly periodic. We conducted experiments to test our hypothesis and found that a 1D representation of the data is a viable approach to get a better denoising performance using both pure-periodic and pseudo-periodic signals. Additionally, both models were able to show adaptability to new patterns when introduced to an existing signal base.