Computer Science Department

TECHNION Israel Institute of Technology

Project Report: Spectro-Temporal Attention

Final project in CS236781: Deep Learning

October 2020

Written by Hadas Orgad and Ariel Iny

The project was advised by Yonatan Elul and Aviv A. Rosenberg

TABLE OF CONTENTS

Abstract 3

Introduction 4

Methods 5

  The Spectro-Temporal attention mechanism 5

  Structure of the STA mechanism 5

    Input to the STA 5

    The STA scheme 5

    The STA output 8

  Data description 8

Implementation 9

  Models Architecture 9

  STA implementation 9

  Data Generation and Processing 10

  Training methods 10

  Experiments 10

Results and Discussion 11

  Constant function 11

  Periodic functions - Different strengths of noise 11

  Periodic functions - Different training noise than testing noise 14

ABSTRACT

This project is a continuation of a previous effort to develop a new attention mechanism which is based on the information stored in both the temporal and spectral domain of a signal. It was shown to perform well on ECG signals, and this justifies the motivation to apply this mechanism to other types of signals such as voice, photos and more.

This report describes the efforts to show the mechanism's success on synthetic data by performing the task of denoising on signals of periodic functions. We show that STA succeeds in getting superior results in a significant portion of the experiments we ran.

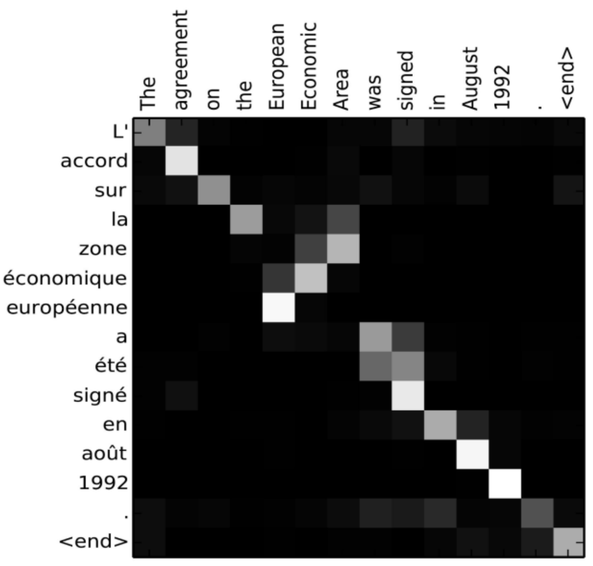
The project code can be found in: https://github.com/aeiny/Deep_project

INTRODUCTION

In the last few years, attention has been one of the most effective concepts in the emerging deep learning field, which has revolutionized the field of Natural Language Processing: it has shown to produce state-of-the-art results in machine translation and other tasks. It didn't stop at the field of Natural Language Processing, showing breakthroughs in the fields of recommendation, image processing, speech recognition etc. Therefore, we have strong reasons to believe that when applied to other fields as well, we will see new and superior results.

Attention allows a Neural Network to learn which parts of the input "to attend" when performing a task. It is interpreted as a set of Keys, Values and Queries which describe how each element in one sequence correlates with each element in another sequence, or even in the same sequence (a.k.a Self-Attention).

One of the most famous examples of use of attention is in the task of translation - using attention enables better learning of the dependencies of words of the sentence in the original language with the words in the target language [2]:



Example attention weights for attentional encoder-decode

In this work, we continue the effort of Aviv A. Rosenberg and Yonatan Elul in developing a new self-attention mechanism, called Spectro-Temporal Attention (STA). This new mechanism was previously applied to medical signals (ECG), and we now strive to expand it to other types of signals like voice, picture and more.

METHODS

In this project, we implemented the STA mechanism as presented in the research proposal [1].

The Spectro-Temporal attention mechanism

The Spectro-Temporal Attention Mechanism (STA) is an Attention-based mechanism which aims to assist in process signals. Nowadays, in signal processing, we look at the signals in 2 separate domains: the temporal domain and the spectral domain. As we know from Signal Processing theory, we can find a connection between these two domains. When wishing to use this combined knowledge of both domains, engineers are mapping the areas and connection in each one of the domains by hand.

The STA mechanism tries to find the connection between the domains with a Self-Attention mechanism, by mapping each area in the temporal domain to an area in the spectral domain which affects it the most (therefore we give more "attention" to it), and vice versa.

Structure of the STA mechanism

Input to the STA

The input to the STA is the temporal domain and the spectral domain of the given signal. Originally, the STA was developed for ECG signals, this is why the two arguments are of the shape of $X, S \in R^{L \times KB}$ when:

- L - The number of ECG leads that are used.
- K - The length of the beats' resampling grid plus.
- B - The total number of ECG beats in each resampling window.

The STA scheme

1. Find the L-dimensional embedding self-attention's "keys", "queries" and "values" X_e,S_e for the temporal and spectral points using a linear transformation $W_{x,1}, W_{s,1}$ on the input:

$$X_e = W_{1,x} S \in R^{L \times KB}$$

$$S_e = W_{1,s} X \in R^{L \times KB}$$

   a. Where $W_{1,x}, W_{1,s} \in R^{L \times L}$ are learned parameter matrices that represent the embed matrices.
   b. On the self-attention notation, X_e,S_e are both the "keys", "queries" and "values" products for our attention.

- Note that we used just one linear transformation for those values, unlike the self-attention we learned in the course which used 3 linear transformations (one for each).

2. Compute the self-attention weights A_x, A_S using Softmax on the resulted “keys” and “queries” on every temporal and spectral point within the window:

$$A_x = softmax({X_e}^T X_e) \in R^{KB\times KB}$$

$$A_S = softmax({S_e}^T S_e) \in R^{KB\times KB}$$

   a. On the self-attention notation:
      I. The $softmax()$ function is the energy function between keys and queries.
      II. The A_x, A_S are self-attention “weights”.
   b. Thus the A_x, A_S columns contain pointwise scores corresponding to the agreement between each pair temporal/spectral locations in the input.

3. Apply the attention weights A_x, A_S to the embedding “values” X_e,S_e:

$$X_a = X_e \quad A_X \in R^{L\times KB}$$

$$S_a = S_e \quad A_S \in R^{L\times KB}$$

   a. X_a,$S_a$contain an L-dimensional embedding of each point after weighting it based on its similarity with all other points.

Note: Until this point, the STA was very similar to the regular self-attention scheme. but from here we diverged this scheme because its equivariance with respect to the input sequence ordering.

This property makes it difficult to process the signals and therefore we don’t apply the attention masks now and wait to step 6, after a second, non-permutation-equivariant, linear transformation.

4. Reshape X_a,$S_a \in R^{L\times KB}$ to shape of ${X_a}'$,${S_a}' \in R^{K\times LB}$ reducing the number of learned parameters in the next step from $KB^2$to K^2:

$${X_a}' = reshape(X_a) \in R^{K\times LB}$$

$$S_a' = reshape(S_a) \in R^{K \times LB}$$

5. Create the attention mask M_x, M_s by applying another linear transformation and reshape the answer matrices to $R^{L \times KB}$:

$$M_x' = W_{2,x} X_a' \in R^{K \times LB} \rightarrow M_x = reshape(M_x') \in R^{L \times KB}$$

$$M_s' = W_{2,s} S_a' \in R^{K \times LB} \rightarrow M_s = reshape(M_s') \in R^{L \times KB}$$

   a. Where $W_{2,x}, W_{2,s} \in R^{K \times K}$ are learned parameter matrices that represent the embedding matrices.
   b. The reshaping is done to return to the original shape of the inputs.
   c. Note that the temporal attention mask M_x is computed only by using the spectral input S, and the spectral attention mask M_s id computed only by using the temporal input X.

6. Apply the attention masks to the inputs by perform element-wise multiplication between the masks to the inputs:

$$X_m = M_x \odot X \quad \in R^{L \times KB}$$

$$S_m = M_S \odot S \quad \in R^{L \times KB}$$

7. Concatenate and reshape the attended and raw inputs to a 3D tensor in shape of $R^{2L \times K \times B}$, and for the spectral domain representation add a "positional encoding" term P:

$$\hat{X} = reshape([X_m; X]) \in R^{2L \times K \times B}$$

$$\hat{S} = reshape([S_m; S]) + P \in R^{2L \times K \times B}$$

   Where P is positional encoding of the spectral domain. The reason for this addition is to force the convolutional filter in step 8 to rely on the physical frequency band locations and not just on the shape of the spectrum.

8. Apply a convolutional layer to compute the final temporal and spectral embeddings,

$$\hat{X}_e = Conv(\hat{X}, W_{3,x}) \in R^{L \times KB}$$

$$\hat{S}_e = Conv(\hat{S}, W_{3,s}) \in R^{L \times KB}$$

   The convolutional parameters:

   I. Filter shape = 3×3
   II. Channel-preserving
   III. Padding = 1

IV. Stride = 1

The STA output

The output of STA is two metrics $\hat{X}_e, \hat{S}_e \in R^{L \times KB}$ that represent the attention of the spectral and temporal domains and will be the input to our denoising model.

Data description

As was mentioned above, the STA mechanism works on signals and therefore the data we will need for testing is signals. We created 3 types of synthetic signals:

1. Constant signal -A
2. Periodic, simple signal -$Acos(2\pi f * x)$
3. Periodic, more complex signal -$Ae^{cos(2\pi f * x)}$

Where - A represents the signal amplitude and f represents the signal frequency.

The motivation for choosing these functions is their simplicity. It will be easy for the model to train on them in our experiments and to observe improvements for the STA model.

In addition to these signals, we used noise signals:

1. Uniform noise - $Uni[a, b]$
2. Normal noise - $N(\mu, \sigma)$

By adding the noise signals to the synthetic signals, we created data with which we can perform our experiments - cleaning the signals.

IMPLEMENTATION

Models Architecture

For the evaluation process, we ran the experiments on 4 types of models. Each model is based on the previous model and has additional components / input.

1. Simple FC network with 2 hidden layers with layers of sizes [input_size, input size] and no non-linearities, which takes the temporal signal as input.
2. Additional RELU layer between the 1st and 2nd hidden layers.
3. Gets additional input - spectral signal concatenated with the temporal signal.
4. Full STA model, which takes temporal and spectral signals, applies STA on it and then inputs it to the FC network.

We implemented these architectures in one modular Pytorch model, which in initialization takes as input three flags: nonlinearities, gets_spectral_input and sta_enabled. Using these flags, the model operates differently on the input given to it.

The 4 models are summarized in the table below:

	nonlinearities	gets_spectral_input	sta_enabled
1			
2	✓		
3	✓	✓	
4	✓	✓	✓

STA implementation

We implemented the STA as a class in Pytorch according to the description in the previous section. Most of the operations are straight-forward. For the matrix multiplications with learned matrix weights, we created a torch.Tensor object and declared it as a torch.nn.parameter with requires_grad=true. This makes the matrices part of the backpropagation graph and therefore the matrices are also learned.

Data Generation and Processing

Data is generated using the CreateData.py script. To use it, you need to edit the call to function MakeSignalAndNoise. We support the creation of the following signals: CosX, Cos(X^2), E^Cos(X), E^Cos(X^2) and the following noises: Uniform and Normal. For each generated signal, we generated its temporal and spectral representations with 450 points of sampling and a space of 50 points between every two signals to avoid overlap.

The training data for each signal contains 50K sample-arrays of 450 points, the test data contains 10K arrays and the validation contains 5K arrays. These numbers are configurable for future use.

Each generated signal's training, testing and validation data was saved to different files.

Training methods

We trained the models to minimize the MSE loss between the output signal and the clean signal. We optimized using SGD/Adam for the simple models and Adam for the STA model. We did it because we found that SGD usually worked better for the simpler models and Adam worked better on the STA, however when we got better results for the STA model, we tried optimizing the simple models with Adam, to make sure that the improvement was not due to the optimization method.

We trained the models for 5000 epochs each and LR=0.00001 for ADAM and LR=0.001 for SGD.

Experiments

We had 3 sets of experiments:

1. Constant function - as a first sanity check to the ability of STA to clean signals, we took a constant signal with value A. This signal is not complicated and therefore we had expected that STA would not be have significant advantage.
2. Periodic functions with different noises - we checked the ability of the STA model in cleaning noise of different types. In all of these experiments, the function and the noise we used to generate the training set was the same as the ones used in the validation and test sets. We experimented with various noise levels, increasing the expectation and variance to make the problem harder.
3. Periodic functions with different training noise than testing noise - in this last set of experiments we demonstrated the strength of STA: the ability to generalize when trained on a specific kind of noise and tested on another.

RESULTS AND DISCUSSION

Constant function

Signal	Noise Train	Noise Test	Linear model	Non-linear model	Non-linear model with spectral	STA model
A=0(Train) A=2(Test)	Uni(-1,1)	Uni(-2,2)	3.998	4.028	3.999	4.00
A=2	Uni(-1,1)	Uni(-2,2)	0.002	0.003	0.098	4.730e-05
A=2	Uni(-2,2)	Uni(-2,2)	0.0005	0.0007	0.0014	7.785e-09
A=2	Uni(-1,1)	Uni(-4,4)	0.011	0.040	2.719	3.743e-05

The STA achieves significantly lower losses which are numerically equivalent to 0, however the other models achieve very good results as well – as we expected, since the function learned is very easy (DC). We notice that all of the models fail to generalizing form one constant function to another.

Periodic functions - Different strengths of noise

Signal	Noise	Linear model	Non-linear model	Non-linear model with spectral input	STA model
$5\,cos\,(\frac{x}{5})$	N(0,1)	0.0048	0.0074	12.501	0.0031
$5\,cos\,(\frac{x}{5})$	N(4,1)	0.0045	0.0044	12.501	0.0047

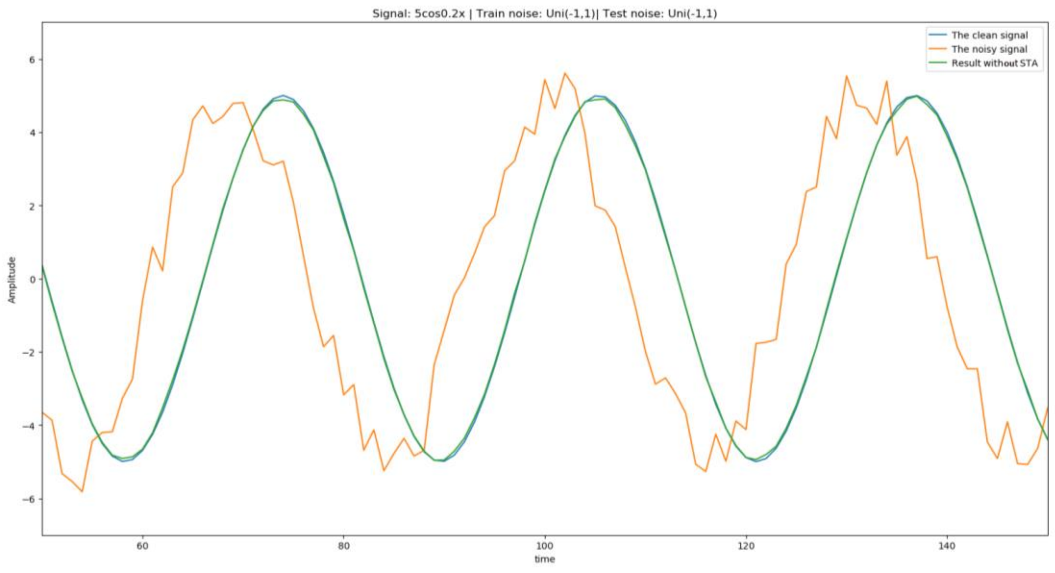
$5\,cos\,(\frac{x}{5})$	N(0,3)	0.040	0.0460	12.501	0.0375
$5\,cos\,(\frac{x}{5})$	U(-1,1)	0.0026	0.0050	12.501	0.0009
$5\,cos\,(\frac{x}{5})$	U(0,8)	0.0238	0.0189	12.501	0.032
$5\,exp\,(cos\,(\frac{x}{5}))$	N(0,1)	0.015	0.007	16.919	0.01031
$5\,exp\,(cos\,(\frac{x}{5}))$	N(4,1)	0.01438	0.00599	16.9188	0.00353
$5\,exp\,(cos\,(\frac{x}{5}))$	N(0,3)	0.11410	0.04307	16.958	0.03458
$5\,exp\,(cos\,(\frac{x}{5}))$	U(-1,1)	0.0060	0.004	16.9187	0.00103
$5\,exp\,(cos\,(\frac{x}{5}))$	U(0,8)	0.0652	0.01982	16.9187	0.01884

As we can see in the above table, STA achieves superior results for tests with low noise expectancy and variance, and for some of the harder tests with higher noise and variance it performs significantly better, especially in tests with normal-distributed noise.

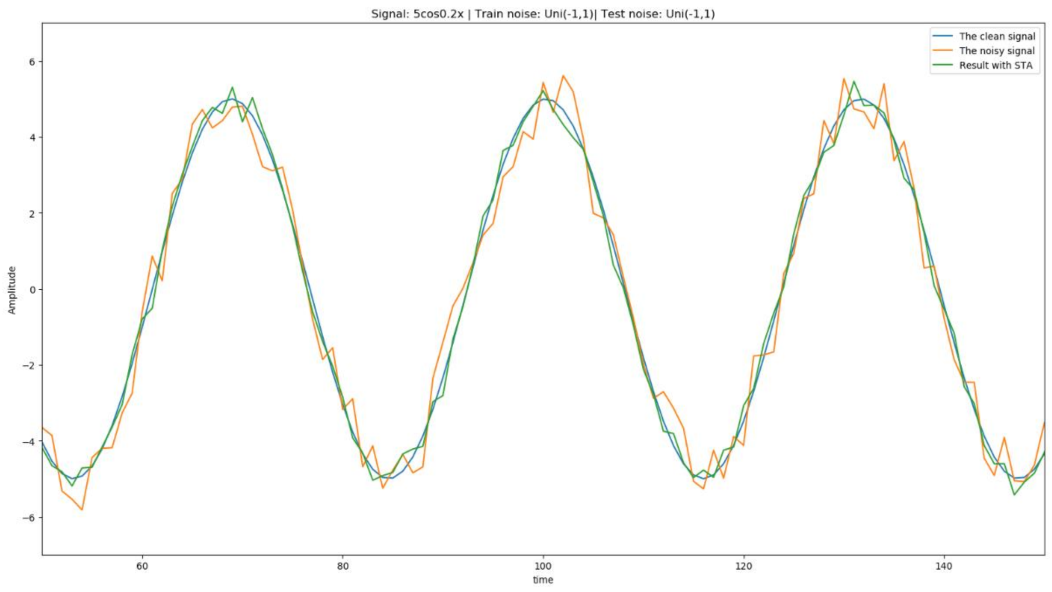
We also note that when giving the model the spectral input as well, without the STA mechanism, it fails to utilize it to improve results and generally fails in learning any signal cleansing. We will not continue in checking this model in the next set of tests.

A closer look into the experiment of signal $5\,cos\,(\frac{x}{5})$ and noise $Uni(-1,1)$:

- The following graph shows the noisy signal, the original signal and the cleaned by the linear network signal:



- Now we'll look at the graph for the STA model: we can conclude that for the simple case when the noise in the training set is the same as the noise in the testing set, the STA gets better results. Nonetheless, the linear model also gets good results with low loss result is intuitive since this task is simple and therefore doesn't require the sophistication of the STA model.



Periodic functions - Different training noise than testing noise

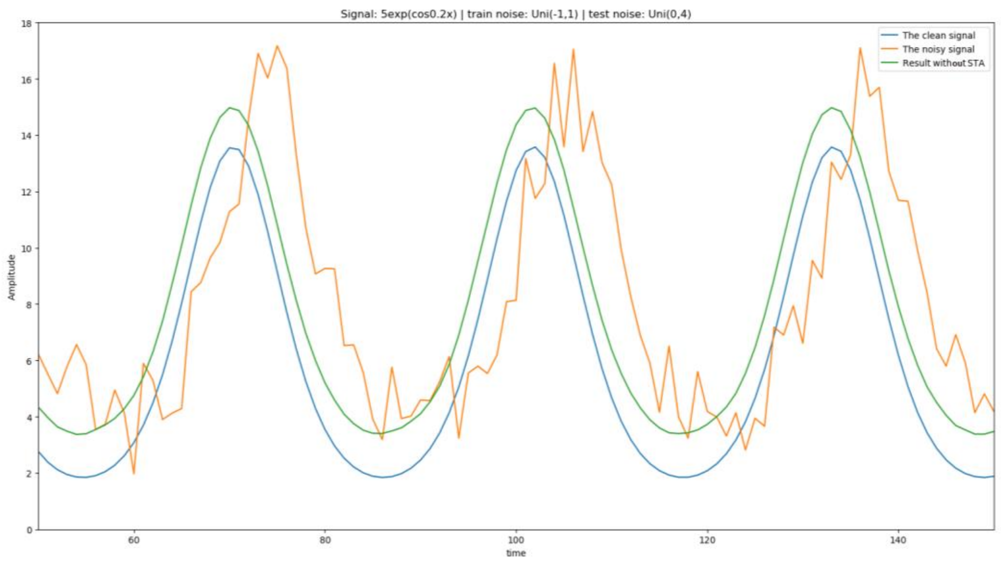
Training Signal	Testing Signal	Training noise	Testing Noise	Linear model	Non-linear model	STA model
$5\,cos\,(\frac{x}{5})$	$5\,cos\,(\frac{x}{5})$	U(-1,1)	U(0,4)	0.030	0.068	0.056
$5\,cos\,(\frac{x}{5})$	$5\,cos\,(\frac{x}{5})$	U(-1,1)	U(0,8)	0.121	0.266	2.731
$5\,cos\,(\frac{x}{5})$	$5\,cos\,(\frac{x}{5})$	N(0,1)	N(2,1)	0.0074	0.022	0.095
$5\,cos\,(\frac{x}{5})$	$5\,cos\,(\frac{x}{5})$	N(0,1)	N(4,1)	0.014	0.068	0.697
$5\,exp\,(cos\,(\frac{x}{5}))$	$5\,exp\,(cos\,(\frac{x}{5}))$	U(-1,1)	U(0,4)	0.466	3.186	0.029
$5\,exp\,(cos\,(\frac{x}{5}))$	$5\,exp\,(cos\,(\frac{x}{5}))$	U(-1,1)	U(0,8)	1.863	13.051	0.502
$5\,exp\,(cos\,(\frac{x}{5}))$	$5\,exp\,(cos\,(\frac{x}{5}))$	N(0,1)	N(2,1)	4.002	2.418	0.613
$5\,exp\,(cos\,(\frac{x}{5}))$	$5\,exp\,(cos\,(\frac{x}{5}))$	N(0,1)	N(4,1)	15.956	10.019	2.37

While for cos signals, we get that the STA does not show any improvement, we see a significant improvement for signals of Exp(cosX). While cosX is a simple functions which consists of one frequency, E(cosX) contains an infinite amount of frequencies which provide important information in the attention process, and therefore the STA model succeeds in generalizing the signal cleaning task more than the linear and non-linear networks.

We'll take a closer look into the case where the function is $5\ exp\ (cos\ (\frac{x}{5}))$, the training noise was $Uni(-1,1)$ and the testing noise is $Uni(0,4)$. This test is considered harder for the following reasons:

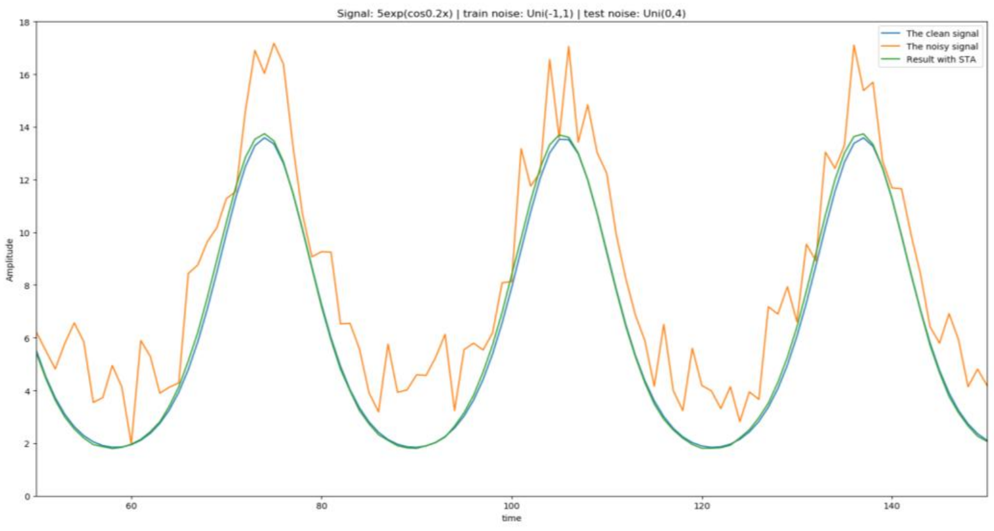
- The testing noise is different from the training noise.
- The noise expectation for the testing data is larger.

The resulted graph shows the results for the linear model:



As can be seen, there is clear difference between the cleaned signal and the original signal.

The resulted graph for the STA model:



We can see from the graph that the STA restored the original signal in much more success than the linear model. We can conclude that even though the model was trained on a different noise, the STA model could generalize for the test data.

To conclude, we see that the STA model brings new qualities, especially in generalizing for different noises in some of the functions.

Seeing these results, we are optimistic about using the STA model on real, non-synthetic signals like photos and voice signals. Real signals are much more complicated and not periodic; therefore, we do not expect such clean results when applied to real signals but still expect to see an improvement. A possible intermediate step is to explore the STA performance on synthetic functions with a non-discrete spectral representation, which is more like the real-life signals we encounter.

REFERENCES

[1] Research Proposal - Spectro-Temporal Attention (STA) by Aviv A. Rosenberg, Yonatan Elul.

[2] Memory-Augmented Neural Networks for Machine Translation by Mark Collier and Joeran Beel.