TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Mihhail Daniljuk 211579IAPM

# APPLICATION OF DEEP LEARNING TRANSFORMER FOR BALTIC SEA WAVE SPECTRA ESTIMATION

Master's Thesis

Supervisor: Sven Nõmm
Tenured Associate Professor

Co-supervisor: Sander Rikka
Senior Researcher

Tallinn 2024

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Mihhail Daniljuk 211579IAPM

# SÜVAÕPPE TRANSFORMERI RAKENDAMINE LÄÄNEMERE LAINESPEKTRI HINDAMISEKS

Magistritöö

Juhendaja: Sven Nõmm
Tenured Associate Professor

Kaasjuhendaja: Sander Rikka
Senior Researcher

Tallinn 2024

# Author’s Declaration of Originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Mihhail Daniljuk

05.01.2024

# Abstract

Application of the deep learning transformer to estimate wave spectra in the Baltic Sea on the basis of satellite-based synthetic aperture radar (SAR) imagery constitutes the scope of the present thesis.

The present research is motivated by the following factors. Wave spectre contains essential information for the areas of maritime navigation, coastal engendering, and environmental studies. For oceanic waves, precise analytic methods exist to compute wave spectre based on SAR data. For coastal seas, oceanic models are unapplicable because of a short wavelength. While in-situ devices like buoys provide the most accurate spectre measurement, covering the sea with large numbers of such devices is not feasible economically, complicated technically, and may become an obstacle for the navigation.

The choice of the methods is based on the following points. Previous studies have demonstrated that regression models could not provide the desired model goodness, and long-short-term memory networks have shown promising results. Deep learning transformers have demonstrated highly accurate results in many different areas because of their ability to capture complex relationships between different elements of the data. The latest leads to the idea to adapt transformer models known from the area of natural language processing to the case of numeric sequences describing wave spectre.

Purely synthetic data sets were used to train and validate models. For these data sets, the following goodness metrics were achieved - mean correlation 0.7, mean MSE of 0.02, and the average difference between the predicted sequence peak value index and the true sequence peak value index is 3.68

The thesis is written in English and is 51 pages long, including 10 chapters, 28 figures, and 2 tables.

# Annotatsioon
## Süvaõppe transformeri rakendamine Läänemere lainespektri hindamiseks

Käesoleva töö raames käsitletakse süvaõpe transformeri rakendamist lainespektri hindamiseks Balti mere piirkonnas satelliidil baseeruva sünteetilise ava radari (SAR) andmete põhjal.

Käesoleva uurimistöö motivatsioon põhineb järgmistel teguritel. Laine spekter sisaldab olulist teavet merenavigatsiooni, rannikuarenduse ja keskkonnauuringute valdkondade jaoks. Ookeanilainete jaoks on olemas täpsed analüütilised meetodid, mis põhinevad SAR-andmetel lainespektri arvutamiseks. Rannikumerede puhul pole ookeanimudelid rakendatavad lühikese lainepikkuse tõttu. Kuigi kohapealsed seadmed nagu poijad pakuvad kõige täpsemaid spektri mõõtmisi, ei ole mere katmine nende seadmetega majanduslikult teostatav, tehniliselt keeruline ja võib muutuda navigatsiooni takistuseks.

Meetodite valik põhineb järgmistel punktidel. Varasemad uuringud on näidanud, et regressioonimudelid ei suuda pakkuda soovitud mudeli headust ning LSTM tüübi närvivõrgud on näidanud paljulubavaid tulemusi. Sügava õppimise transformerid on mitmetes erinevates valdkondades näidanud äärmiselt täpseid tulemusi tänu nende võimele püüda keerulisi seoseid erinevate andmeelementide vahel. See viib mõtteni kohandada transformeeri mudeleid, mida tuntakse loomuliku keele töötlemise valdkonnast, numbriliste jadade vormis kirjeldatud laine spektritele.

Selles uurimuses kasutati kasutati puhtalt sünteetilisi andmehulki mudelite treenimiseks ja valideerimiseks. Nende andmehulkade puhul saavutati järgmised headuse mõõdikud. Keskmine korrelatsioon 0,7, keskmine MSE 0,02 ning ennustatud järjestuse tipuväärtuse indeksi ja tegeliku järjestuse tipuväärtuse indeksi keskmine erinevus on 3,68.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 51 leheküljel, 10 peatükki, 28 joonist, 2 tabelit.

# List of Abbreviations and Terms

| | |
|---|---|
| RS | Remote Sensing |
| SAR | Synthetic Aperture Radar |
| NLP | Natural Language Processing |
| VV | Vertical-Vertical |
| VH | Vertical-Horizontal |

# Table of Contents

# List of Figures

# List of Tables

# 1. Introduction

The present thesis proposes applying the deep learning transformer model to estimate the wave spectrum from a satellite-based synthetic aperture radar (SAR) image in the Baltic Sea. The wave spectrum contains essential information about waves for areas such as coastal engendering, environmental protection, wind park planning and building, and of course, commercial navigation. Although buoys positioned in different parts of the sea may provide the same information, their wider use contains many drawbacks. The larger number of buoys may affect navigation, are expensive to maintain and operate, and constant data exchange is difficult, finally, the buoy may be lost during stormy weather. All this motivates the usage of SAR to estimate the wave spectrum.

For the oceanic areas with longer swell waves, compared to wind waves predominant in closed seas, there exist analytic inversion models, to estimate the wave spectrum from the SAR image spectrum. These models are hardly applicable for waves with a shorter wavelength that are dominant in the Baltic and other closed seas. This leads to the idea of using artificial intelligence techniques for the task. One can enquire about statistical machine learning techniques that are less computationally demanding. To the best knowledge of the author, no results have been published in this area. The experience of our workgroup has demonstrated that, while it is possible to fit polynomial models and regression trees, their quality is rather lower.

The task of predicting wave spectra based on SAR images may be formulated as the transformation of the SAR image spectrum to the wave spectrum. Therefore, the problem is to fit the model, which allows us to convert one spectrum into another. One of the possible reasons why simpler models fail is due to the complexity required to capture the relations between different elements of the specter. SAR image spectrum retrieved by Fast Fourier Transform (FFT) is challenging to interpret because it incorporates various elements related to the SAR platform and waves. For example, SAR backscatter is dependent on the surface roughness, incidence angle, sending and retrieving polarization, and satellite flight direction relation to wave propagation direction. Moreover, there are many other oceanographic processes that have effects seen on the surface but with no clear explanation . All the above requires a powerful AI system, that is able to find relations between different processes and can optimally parametrize those relations.

The Transformer architecture, originally designed for natural language processing tasks

[1], has exhibited exceptional capabilities in capturing complex relationships in sequential data. By treating wave spectra as sequential information as in [2], the Transformer model offers a novel perspective that may overcome the limitations of traditional techniques.

# 2. Literature review

The estimation of wave spectra is a critical aspect of sea science, providing valuable insights into sea state conditions. The idea of retrieving surface gravity wave parameters from spaceborne or airborne images is not new. One of the earliest Earth-observing satellites, SEASAT, was launched in 1978, and designed to test various oceanographic sensors and gain a better understanding of Earth's seas. The satellite operated in Earth orbit for 105 days, measuring sea-surface winds and temperatures, wave heights, atmospheric liquid water content, sea ice features, and ocean topography [3]. In the same year [4] proposed the two-frequency microwave technique for measuring ocean-wave spectra from an airplane or satellite. In this article, authors have shown that by carrying out a specific signal processing it is possible to measure the ocean-wave spectra from an airplane or satellite.

Despite synthetic aperture radars being used in the 1980-ies to image ocean surface waves, it was still not fully understood how they relate to the actual wave field. The [5] reviews and extends models on the imaging mechanism. It is noted that a description of the imaging process by the functions suggested can only be adequate for low to moderate sea states.

In 1982, a direct method to estimate wave height from digitally processed SAR imagery was introduced in [6]. The method is simple and is based on measuring the contrast of the image and the wavelength of the dominant wave. Comparison with sea truth measurements shows agreement to within about only 20 percent.

In 1983 the relationship between the SAR image stripes and the real surface wave condition was first investigated by using Monte-Carlo simulation [7], as a result - a theoretical model previously developed for describing the imaging of monochromatic ocean waves by synthetic aperture radar (SAR) is extended to relate ocean wave spectra to SAR image spectra. In the same year substantial limitations of spaceborne SAR for global ocean wave monitoring were discussed in [8] and additional correction strategies were suggested.

Later in 1985 the basic synthetic aperture radar (SAR) theory of ocean wave imaging mechanisms was reviewed, using both known work and recent experimental and theoretical results from the Marine Remote Sensing (MARSEN) Experiment. The analysis has revealed a more comprehensive understanding of the SAR imaging phenomenon than what was previously accessible. A novel and broadly applicable imaging model is proposed [9].

In 1986 the relative importance of the various motion-related contributions to the SAR imaging mechanism of ocean surface waves was studied in [10]. It is shown that in cases where imaging is nonlinear (wind waves, which are predominant in closed areas such as the Baltic Sea), the wave spectrum cannot be retrieved from the SAR image spectrum by applying linear inversion techniques. The study also presents computer simulations of SAR imaging of fully developed wind seas with different peak wavelengths and propagation directions.

In 1990, [11] proposed that while remote sensing of ocean surface waves can be difficult using conventional synthetic aperture radar (SAR) techniques, they can be observed clearly by SAR in the interferometric configuration (INSAR). This enhancement is attributed to INSAR's capability to provide images of the local surface velocity field, in contrast to conventional SAR. It is demonstrated that INSAR can be used to obtain wavenumber spectra that are in agreement with power spectra measured in situ.

Later in 1991, [12, 13] suggested a solution to a non-linearity problem discussed in [10]. Authors have derived the nonlinear mapping relation between the SAR image spectrum and the ocean wave spectrum. This inversion algorithm is called the Max Planck Institute (MPI) Algorithm. In 1994 [14] proposed generalizations of the non-linear ocean-SAR transform and a simplified SAR inversion algorithm. It is shown that the non-linear transformation from the ocean wave spectrum to the synthetic aperture radar (SAR) image spectrum can be extended to include point target spreading effects. The drawback of such algorithms is that they require some first-guess wave spectrum, which is iteratively modified until a satisfactory agreement is obtained.

In 1995 image cross-spectra obtained by combining pairs of single-look SAR images were utilized in an inversion scheme for extracting the underlying ocean wave spectrum [15]. It is stated that the image cross-spectra is shown to significantly reduce the speckle noise level while preserving the spectral shape and providing information about the wave propagation direction.

In 1996 [16] introduced an improved MPI algorithm, which uses a modified cost function and additional iteration loop. The new algorithm retrieves smooth spectra avoiding the discontinuities that tended to arise in the previous algorithm. It is stated that the overall correlation of a large set of simulated SAR spectra with the measured SAR spectra is found to be significantly higher than with the previous algorithm, indicating that the algorithm not only overcomes isolated shortcomings of the earlier algorithm but also yields retrieved wave spectra which are generally more consistent with the input SAR data.

All mentioned studies allowed to estimate swell parameters over the ocean, but for the coastal areas, such algorithms provide less accuracy. This motivated the development of empirical methods. 10 years later a new empirical approach to retrieve integral ocean wave parameters from SAR data is presented [17]. The main idea of this approach is to estimate bulk wave parameters (significant wave height, peak period, etc.) used in practical applications without retrieving the two-dimensional wave spectrum.

Later in 2012, a semi-empirical algorithm for SAR wave height retrieval was proposed in [18]. The objective of the algorithm is to estimate the wave height from SAR imagery without any prior knowledge. Multiple empirical algorithms have been developed in 2016, which are aimed at estimating the same significant wave height from SAR data [19, 20].

In 2017 neural networks were used to relate the nonlinear relationships between the input SAR image parameters and output geophysical wave parameters [21]. The developed neural networks extend the SAR ability to retrieve useful wave information under a large range of environmental conditions including extratropical and tropical cyclones in which significant wave height estimation is traditionally challenging. The study confirms that using machine learning for this task is possible.

In 2018 deep deep-learning algorithm Inception v3 was applied to the SAR images to classify geophysical phenomena [22]. Preliminary results of the study show that deep-learning methodology is effective for SAR imagery, with overall accuracy exceeding 0.9 for the described task.

Since 2018 many studies applied machine- or deep learning techniques to the SAR image data for classification/regression [23, 24, 25, 26]. A different approach was proposed in [2], where spectra were considered to be a sequence with specific relations between the values within itself. The authors applied the LSTM algorithm to model these relationships and achieved promising results.

The idea of using the transformer model presented in [1] for the described problem, comes directly from the sequential nature of the spectra. Transformer models showed exceptional performance in many different areas and were adapted to many different tasks [27] such as language modeling, computer vision, audio and speech recognition, healthcare, etc. As the transformer model was initially developed for NLP problems, it is assumed the datatype for the model is a text (sequence of words), where words or some combinations of words, letters, etc represent different classes.

This thesis aims to investigate and evaluate the application of deep learning transformers for the estimation of Baltic Sea wave spectra from the spectra of SAR image data. Both spectra are numeric sequences, which makes this a regression problem. In 2022, a transformer-based regression scheme for estimating significant wave height in oceans was proposed in [28]. The proposed method outperforms existing state-of-the-art machine learning and numerical approaches for significant wave height prediction. However this method included multiple input data sources, not the SAR image alone, and as in the majority of related studies, the method is applied to ocean waves. Another drawback is that significant wave height alone doesn't provide as much information as full wave spectra. These limitations motivated the use of a transformer for the estimation of Baltic Sea wave spectra from the spectra of SAR image data.

# 3. Background

## 3.1 SAR and remote sensing for oceanography

Covering more than 70% of the Earth's surface, oceans play a crucial role in providing various services to both humans and the environment. Recognizing the significance of ocean environments, it becomes imperative to employ advanced technologies for monitoring. In this regard, datasets obtained through in situ, shipborne, airborne, and spaceborne systems are actively employed [29, 30, 31].

While in situ measurements offer the most accurate datasets for ocean studies, they come with certain limitations. These include being point-based observations covering small areas, and the deployment and maintenance of in situ platforms, such as buoys, being costly and labor-intensive [32]. On the contrary, the use of airborne and spaceborne Remote Sensing (RS) systems for ocean mapping and monitoring is of considerable interest due to their extensive coverage, diverse temporal and spatial resolutions, and cost-effectiveness of the corresponding datasets [32, 33, 34].

Synthetic Aperture Radar (SAR) systems, as side-looking radar instruments, capture surface information in two-dimensional directions (azimuth and range). These systems transmit pulses toward different Earth targets, recording the resulting scattering echoes. SAR systems typically possess day-and-night imaging capabilities and can operate in all weather conditions. Through platform movement and signal processing techniques, high-resolution SAR data can be generated [35].

Various methods have been developed to derive oceanographic parameters from SAR datasets. These methods generally fall into three groups: statistical, physical, and Machine Learning (ML) models. Statistical algorithms rely on correlation relationships between in situ measurements of oceanographic parameters and information collected by RS systems. These models need optimization for different study areas. Physical models are based on the physical laws of RS systems, providing better results but requiring numerous inputs that are often unavailable. ML algorithms, including traditional ones like Random Forest (RF) and Support Vector Machine (SVM), or more advanced models such as Convolutional Neural Network (CNN), are increasingly used for various oceanographic applications. Deep Learning (DL) methods, like in many other RS applications, generally offer higher accuracies compared to statistical, physical, and traditional ML algorithms [29, 2, 36].

# 4. Formal Problem Statement

**The main goal of the thesis**

This thesis aims to investigate and evaluate the application of deep learning transformers for the estimation of Baltic Sea wave spectra from the spectra of SAR image data. Formally, this is a regression problem where the spectre computed on the SAR imagery constitutes an independent variable and the in situ spectre measured by the physical buoy constitutes the dependent variable.

**Research hypothesis**

While early attempts to use linear and non-linear regression models did not result in a model quality comparable to those achieved by analytic models for oceanic waves, the application of the LSTM models led to promising results. The most probable reason for this is the complexity of the relation between the SAR image spectre and the in situ measured spectre. The relative success of LSTM models for this task leads to the working hypothesis of this work. Adaptation of the NLP transformer may lead to more accurate predictions compared to LSTM. The fact that Transformers have been used successfully for language translation and image analysis tasks supports this hypothesis.

**Research questions**

While transformers have shown sophisticated results in other areas, there are a few drawbacks that need to be addressed. The first is that the majority of scholars agree that input sequence length may be an issue and that transformers handle shorter sequences better. Another issue is the required computational resources and time. Combining this with the main goal of the thesis, the following research goals may be formulated.

1. Adapt the transformer model for the case of numeric data.
2. Develop the framework for spectra data preprocessing answering limitations of transformer models.

For quality assessment, previous work has used mean squared error (MSE) and linear correlation coefficient. This contribution extends the quality assessment technique by adding a few more metrics. The difference between the predicted sequence maximum value index and the true sequence maximum value index. The difference between the predicted sequence maximum value and the true sequence maximum value. These metrics are precisely described in Section 7 and constitute another novel point of the present research.

# 5. Datasets

## 5.1 Datasets' description

The NORA3 wave hindcast, which is based on the WAM wave model [37], [38], is used as the ground truth. The hindcast covers the pan-Arctic domain, including also the Baltic Sea with a 3 km resolution. However, the spectra were only outputted with a 30 km resolution. The model spectra are discretized using 30 frequencies logarithmically spaced in the range of 0.0345-0.5476 Hz and 24 equally spaced directions covering a full circle.

Sentinel-1 Interferometric Wide Swath (IW) Single Look Complex (SLC) sub-images were calibrated and speckle filtered with a Frost (5x5) filter. During the multi-look operator, pixels were not averaged to represent squares, and the images were left in radar projection. The image spectra ($ISP$) are calculated from both polarisations ($VV$ and $VH$) with the fast Fourier transform (FFT). Subsequently, the values of the $ISP$ components $ISP_x$ and $ISP_y$ were interpolated to fixed wavelength values between 215 and 30 m (23 values with variable intervals). SAR data was collected around the location of the model spectra from the beginning of 2015 to the end of 2021.

As the thesis goal is to find a mapping from the SAR signal (wavelength domain) to the buoy signal (frequency domain), then from now on the SAR signal dataset will be called as the input dataset and the buoy dataset will be related as the output dataset. To include information from both, co-polarization (VV) and cross-polarization (VH) of SAR signal, co polarization was divided over the cross polarization. The total number of observations before the pre-processing was 71215 and after pre-processing it is 41380. Sequence lengths of both datasets are fixed - the SAR sequence length is 23 units, and the buoy sequence length is 43 units.

## 5.2 Data pre-processing

In order to feed the data into the transformer model and proceed with the training process the data is pre-processed in multiple steps. These steps should be performed in the same order as they are described further in this thesis as some of them require the whole dataset's information to perform the associated transformation, f.e. normalization. To be more precise, the main objective of preprocessing is to prepare the data for the tokenization layer by reducing the amplitude and variety, of both datasets, so it is possible not to lose much information when tokenizing data (see Section 6.1.2). It is required because of the evenly spaced samples between the maximum and minimum value of the dataset used in the tokenization-detokenization process. Both, input and output datasets should be pre-processed.

### 5.2.1 Filtering

At first, the data were filtered according to the distribution of the maximum values of the sequence. There are two datasets, and therefore two filtering steps were applied to the data, one is considering the input sequence (See Figure 1), and the second - the output sequence (See Figure 2). Filtering was required to drop out outlying sequences with high or low maximum values because these negatively affect the tokenization and thereafter the training process.

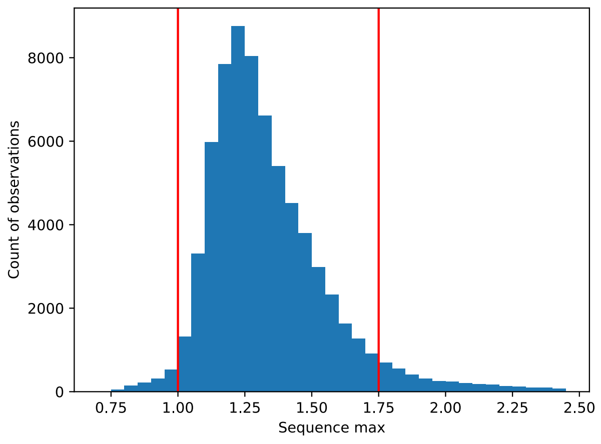


Figure 1. *Input sequences' max values histogram*

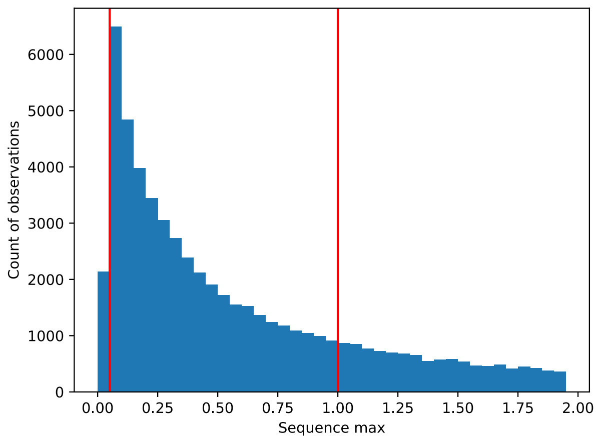


Figure 2. *Output sequences' max values histogram*

Filtering thresholds were found experimentally so that smaller tokenizer powers could be used in combination with a sufficient number of observations to train the model. Note that figures (1) and (2) are limited at values 2.5 and 2.0 max respectively. There were observations with max values above 20, and even though their count was not significant compared to all other observations, even one such outlier equally influences further data processing and model training.

The effect of such outliers on the tokenization process can be seen in figures 3 and 4. Observing the range of every graph, there is a clear pattern - the lower the values of the signal, the less precise the tokenization. That is because when creating a tokenizer for a dataset, its *max* and *min* values are used as parameters (see section 6.1.2)

- The leftmost graph has the greatest values and thus tokenization works well
- The middle observation contains values ∼10 times smaller and there are already problems in capturing the initial signal with the same tokenizer. Instead of ca 15 different values of the initial sequence, there are only two tokens describing it - 0 and 1.
- The rightmost observation peak value is ∼200 times smaller than the peak value of the leftmost observation and tokenization completely fails on it - all sequence values are tokenized as the same token and such signal detokenization results in 0 information signal

The necessity of applying such harsh filtering comes from the fact that ∼65% of output dataset sequences have a peak value below 1.0. These sequences are similar to the observation in the middle section of figure 3, and therefore tokenizer would fail to accurately tokenize more than a half of the dataset.

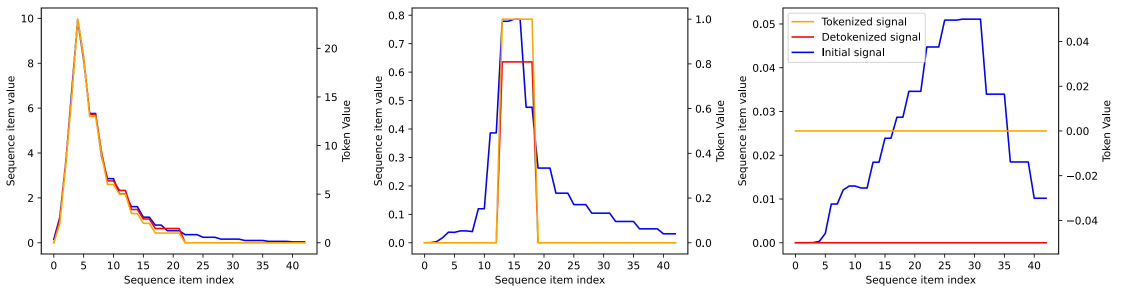


Figure 3. *Tokenization on raw data*

### 5.2.2 Logarithm transformation

After filtering the dataset the tokenizer can represent most of the observations without significant loss of information, but there are still some observations like the middle

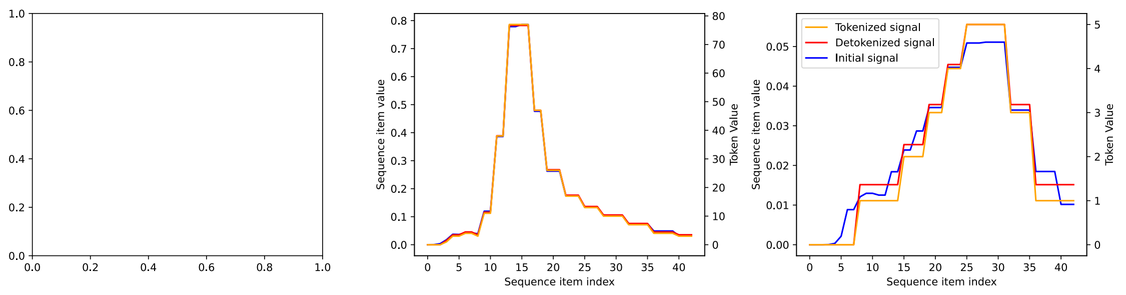


Figure 4. *Tokenization on filtered data*

section of figure 5, which can't be represented properly. To fix this problem logarithm transformation (5.1) was used to "flatten" values in sequences, moving low and high values closer to each other. Results of tokenization on filtered and logarithm-transformed data can be seen in figure 6, regardless of the signal values tokenizer represents them well. There is one drawback though, which is related to the scope of newly created tokens, comparing the two first sections of figures 5 and 6 one can note that token peak values before logarithm-transformation in these two sections were significantly different - ∼80 versus ∼5, whereas after the transformation they are much more alike ∼95 versus ∼75. This can confuse the transformer model because two different sequences can have similar token values.

$$x_{new} = \ln(x) \tag{5.1}$$

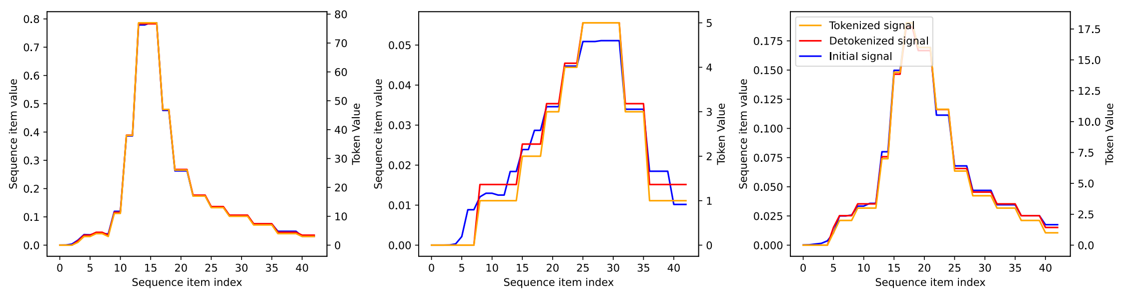


Figure 5. *Tokenization of full filtered data*

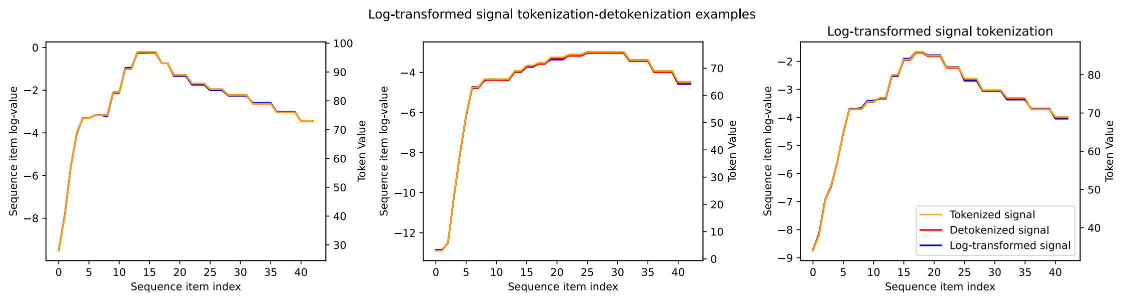


Figure 6. *Tokenization of logarithm-transformed data*

### 5.2.3 Normalization

During the development stage, two ways of normalizing the data were tested:

1. Z-score normalization
$$x_{new} = \frac{x - \mu}{\sigma}, \tag{5.2}$$
   - μ- Mean of the dataset
   - σ- Standard deviation of the dataset

   Z-score normalization is one of the common ways to normalize the data. This way of normalization results in a new dataset such that the mean of all values is 0 and the standard deviation is 1.
2. Min-max normalization
$$x_{new} = \frac{x - \min(D)}{\max(D) - \min(D)} \tag{5.3}$$
   - $D$ - Dataset

   Min-max normalization is another well-known way to normalize the data. For every feature, the minimum value of that feature gets transformed into a 0, the maximum value gets transformed into a 1, and every other value gets transformed into a decimal between 0 and 1. In the case of sequence datasets, the minimum value of the whole dataset is mapped to a 0, the maximum value is mapped to a 1, and all other values are calculated according to formula 5.3

Both normalization methods can be applied to the data, but due to the higher stability and immunity to outliers, the z-score normalization method was chosen to process the data. This is the latest step of data pre-processing after which data is ready to be wrapped into appropriate data loaders for PyTorch to run the model training.

# 6. Transformer

## 6.1 Model architecture

The transformer model implemented in this thesis is slightly different from the one described in the original paper [1], but shares the same architecture (see Fig. 7). Blocks, which concept applies well to the problem specified in this thesis are left without changes. It is an encoder-decoder model that consists of different building blocks, such as tokenization, embedding, multi-head attention blocks, and feed forward neural networks.

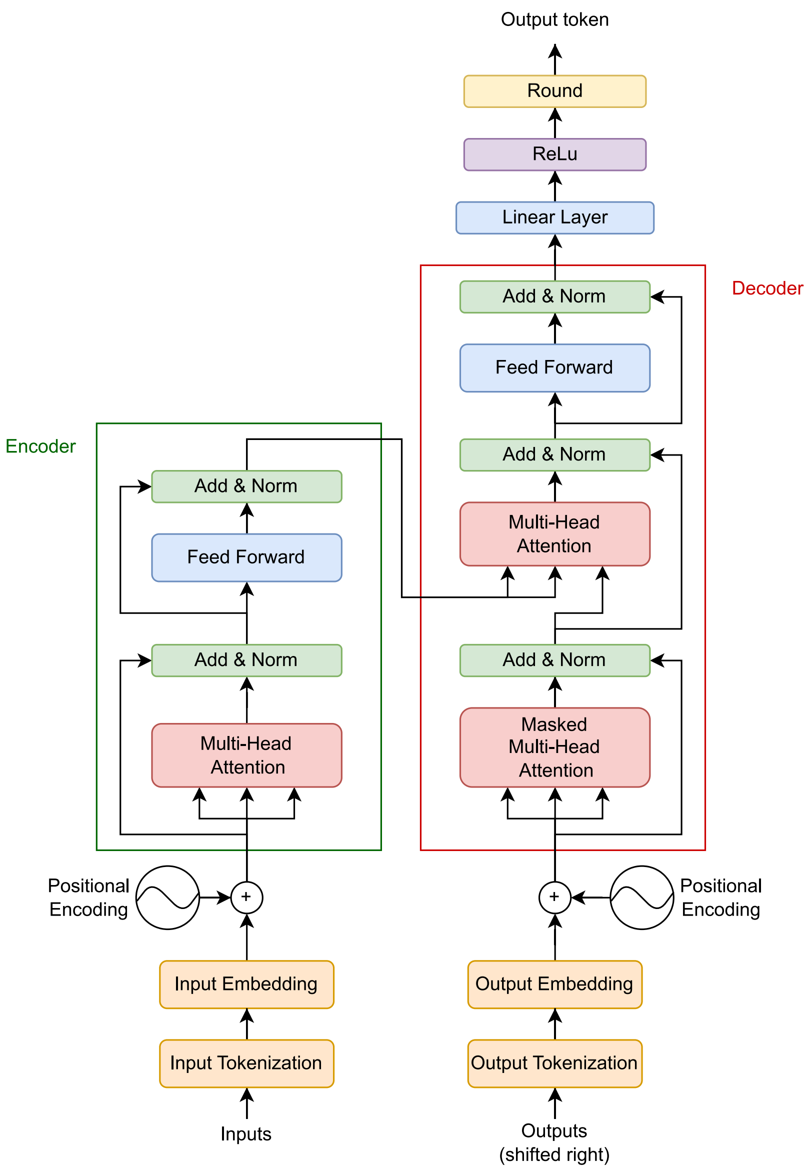


Figure 7. *Transformer architecture*

In the original paper, [1] both the encoder and the decoder are composed of 6 identical layers, which are stacked one after another. This stack allows the model to learn more complex interactions within a sequence which is required for language modelling. The transformer model implemented in this thesis is aimed at solving a potentially easier problem that doesn't require such a high level of complexity and therefore there is only 1 encoder and 1 decoder implemented. The transformer is based on the [39].

### 6.1.1 Inputs

On the transformer diagram, there are two separate inputs, one for the encoder and one for the decoder (See Figure 8). The encoder takes the so-called input sequence as the input, that is the sequence from which transformation is learned during the training of the model. The decoder input is the whole desired sequence excluding the final ***END*** token. Thus a full sequence starts from the ***START*** token and ends on the penultimate token.

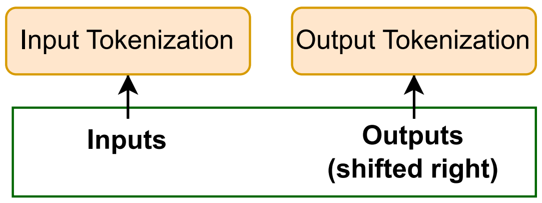


Figure 8. *Transformer inputs*

### 6.1.2 Tokenization layer

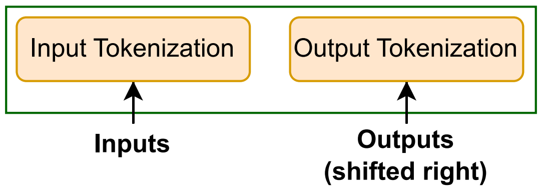


Figure 9. *Tokenization layer*

The first layer is the tokenization layer (See Figure 9). Regular transformer, which is used in NLP tasks requires this layer to transform words into numbers to proceed with model training. The objective of the tokenization is to transform string input into numeric input. Considering the type of the data, which is a floating point number with 16 decimal places - both datasets are already numeric and most of the values are unique. It may seem that tokenization is not needed at all, but due to the fact that each token is tied to a unique value in the dataset this will result in an enormous amount of unique input and output values $\sim (41380 * 23)$ and $\sim (41380 * 43)$ respectively. Therefore model would be too complicated to learn all the parameters.

As implemented tokenizer is used for the numeric input, from now on, vocabulary size (amount of tokens required to describe the dataset) will be related as a tokenizer power.

### Tokenization process

Description of the tokenization process used in this transformer:

- Create evenly spaced samples between the dataset's maximum and minimum point, the number of samples is defined using the tokenizer power parameter.
- Using samples (bins) created in the previous step map each value in the dataset to index of range this value belongs to
- Adding ***START*** token and ***END*** to the beginning of the sequence and the end respectively

Example (See Figure 10):
The dataset maximum value is 20, the dataset minimimum value is 0, and the tokenizer power specified is 5.

1. by creating evenly spaced samples next bins are received - [0, 5, 10, 15, 20] and thus ranges are
   - 0 <= x < 5 - index 1
   - 5 <= x < 10 - index 2
   - 10 <= x < 15 - index 3
   - 15 <= x < 20 - index 4
   - >= 20 - index 5
2. having dataset like [1, 5, 6, 12, 0, 15, 20] and applying indexing, next tokens as a result - [1, 2, 2, 3, 1, 4, 5]
3. Adding ***START*** token and ***END*** tokens - [***START***, 1, 2, 2, 3, 1, 4, 5, ***END***]

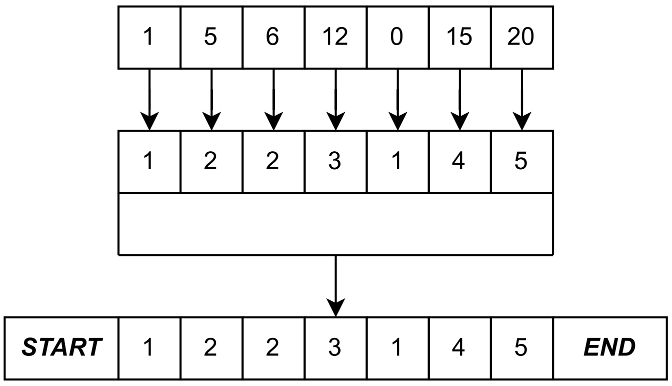


Figure 10. *Tokenization example*

This way of tokenizing data is similar to discretization, and therefore, there is a loss of information. The magnitude of the loss is strongly dependent on the tokenizer power. With such tokenizer the same token will represent multiple values close to each other, but at the same time, drastically smaller tokenizer power can be used $\rightarrow$ thus model has reasonable complexity and bigger predictive power (see section 7).

**Detokenization process**

Description of the detokenization process (See Figure 11) used in this transformer:

- Take away ***START*** and ***END*** tokens from the tokenized sequence.
- Using samples (bins) created in the tokenization process map each token back to the value using token as an index for bins list.

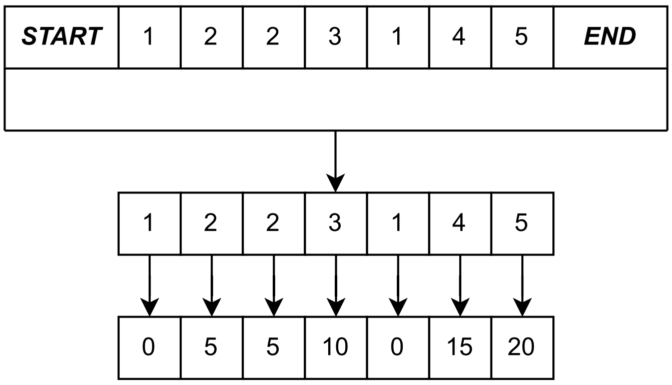


Figure 11. *Detokenization example*

Figure 11 shows that applying such detokenization doesn't always result in an initial signal before tokenization.

By specifying the tokenizer power and by controlling the difference between dataset maximum and minimum points it is possible to control the precision of the tokenization and therefore detokenization. The bigger tokenizer power results in more ranges and these ranges are more narrow, which, as a result, helps to describe the dataset values more accurately.

Compared to NLP tokenizers, the aforementioned tokenizer results in tokens that contain semantic meaning, allowing for them to be compared to each other. This feature is later used to modify the transformer model for a regression problem instead of classification (see section 6.1.11 and 6.2.4).

***START*, *END* and *PAD* tokens**

Due to the technical limitations, there is a need to specify the power of the tokenizer more by 2, these two exceeding tokens are reserved for ***START*** and ***END*** tokens

- ***START*** = *N*-2
- ***END*** = *N*-1

, *N* - Power of the tokenizer

Considering the ***PAD*** token - both datasets used in this thesis have fixed size sequences, therefore there is no need to use ***PAD*** token at all, but one possible solution could be one more reserved token as for the ***START*** and ***END*** tokens.

### 6.1.3 Embedding layer

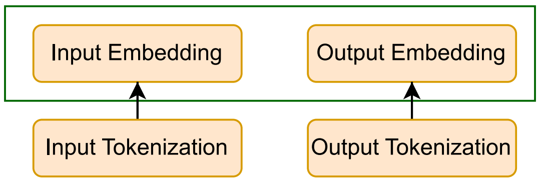
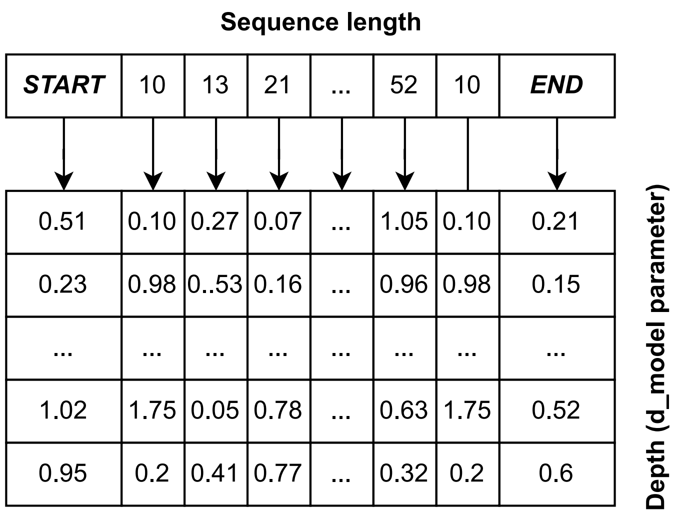


Figure 12. *Embedding layer*

The tokenization layer is followed by the embedding layer (See Figure 12), which task is to represent each token by the unique vector of a specified length (See Figure 13). This mapping is learned during the training and similar tokens are mapped to similar vectors. This layer is required to add more information about each token into the model and reveal more connections. As in [1], experiments showed that specifying a bigger depth of the model results in better performance of the model (see section 6.2.5 or chapter 7). This layer is left without changes because the idea of enriching tokens with more information applies well to the specified task.



| 0.51 | 0.10 | 0.27 | 0.07 | ... | 1.05 | 0.10 | 0.21 |
|---|---|---|---|---|---|---|---|
| 0.23 | 0.98 | 0..53 | 0.16 | ... | 0.96 | 0.98 | 0.15 |
| ... | ... | ... | ... | ... | ... | ... | ... |
| 1.02 | 1.75 | 0.05 | 0.78 | ... | 0.63 | 1.75 | 0.52 |
| 0.95 | 0.2 | 0.41 | 0.77 | ... | 0.32 | 0.2 | 0.6 |

Figure 13. *Embedding example*

### 6.1.4 Positional encoding

After two layers of transforming there is still one problem, which is neglected - that is the position of each transformed value. As both datasets contain sequences - the order of the values also contains information, and therefore it can't be ignored. Another reason to provide positional information into the model is that two different sequences, coming from the same set of tokens will be interpreted by the transformer the same way, which leads to confusion and lower performance of the model.

To add information about the absolute position of the value in the sequence there is a positional encoding layer (see figure 14). Note that positional encoding is done before data is fed into the encoder/decoder, such implementation makes it possible not to integrate the encoding into the model itself, but rather allow the input data to inject this information.

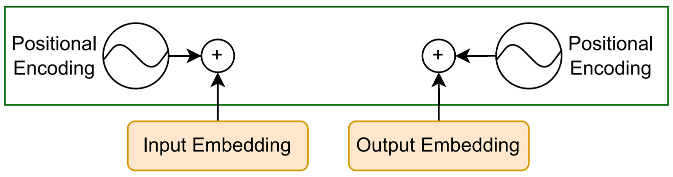


Figure 14. *Positional encoding layer*

The way of adding positional information implemented in this thesis is very similar to the one proposed in [1], the only difference is the formula used. The main idea is to use sine and cosine functions with different periods for each depth level, sine for even levels, and cosine for odd. Combine these functions with the position of the value in the sequence (see formulas 6.1 and 6.2) and the result is a unique vector of position encoding values.

Formulas to calculate positional encoding vector values:

$$PE(pos, 2i) = \sin\left(pos * \exp\left(2i * \frac{-\ln\left(10000.0\right)}{d\_model}\right)\right), \tag{6.1}$$

$$PE(pos, 2i+1) = \cos\left(pos * \exp\left(2i * \frac{-\ln\left(10000.0\right)}{d\_model}\right)\right), \tag{6.2}$$

The resulting vector is the same size as the embedding, so one can be added to another. After adding the positional encoding values to the embedding, the input is ready for the encoder/decoder.

### 6.1.5 Feed forward network

Two more building blocks require description before going to the encoder/decoder. One such block is a feed forward network (see figure 15) which follows multi-head attention blocks. The task of the feed forward network is to apply an element-wise non-linear

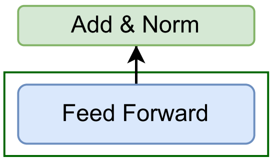


Figure 15. *Feed forward layer structure*

transformation to each token separately and identically [1]. This transformation is done using two dense-layer neural network with ReLu activation function in between (see formula 6.3)

$$FFN(x) = \max(0, xW_1 + b_1)W_2 + b_2, \tag{6.3}$$

To align this block with multi-head attention blocks, feed forward network input and output layers have the same size as the embeddings. Between these two layers, there is one hidden layer with a specified number of nodes (see figure 16). This layer adds nonlinear transformation to each token separately. Such a structure allows the output from the multi-head attention block to be fed into the neural network without any loss of information. At the same time, the output of this feed forward network can be fed further into a multi-head attention block, therefore there is no loss of information. The structure of this layer is left without changes.

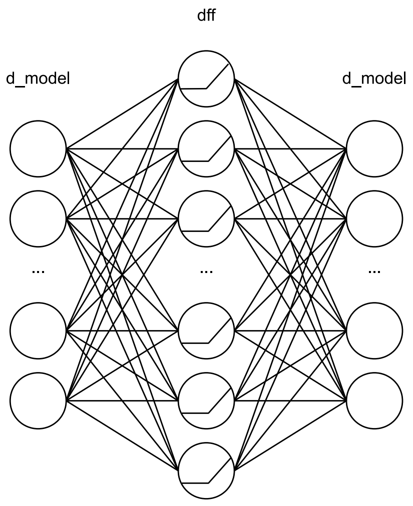


Figure 16. *Feed forward layer structure*

### 6.1.6 Add and normalization layer

The second block which is shared by both the encoder and the decoder is the add and normalization block (see figure 17). This block consists of two consequent operations

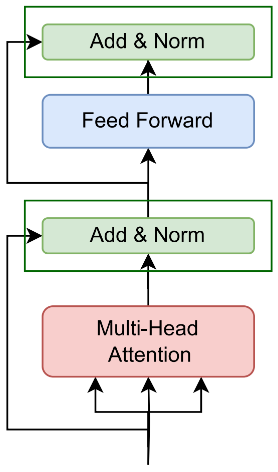


Figure 17. *Add & Norm layer*

1. Residual connection [40]
2. Normalization [41]

Residual connection is used to preserve original input, which in its turn helps to learn more complex functions. At the same time, the residual connection also helps to avoid the vanishing gradient problem. To align all residual connections, all the sub-layers as well as the embedding produce output of the same dimension $d_{model}$

The normalization layer in its turn helps to keep the training stable. The output of the add and normalization layer is then as follow (see formula 6.4

$$LayerAdd\&Norm(x) = LayerNorm(x + SubLayer(x)), \tag{6.4}$$

where $SubLayer(x)$ is the function implemented by the sub-layer itself (it may be multi-head attention or feed forward network)

### 6.1.7 Encoder

The task of the encoder is to map the continuous input sequence $(x_1, x_2, x_3, ..., x_n)$ to another continuous sequence $z = (z_1, z_2, z_3, ..., z_n)$, which later will be used in the decoder together with decoder input sequence to make a prediction. The encoder does such transformation using the attention mechanism, feed forward neural network, and residual connections (see figure 18).

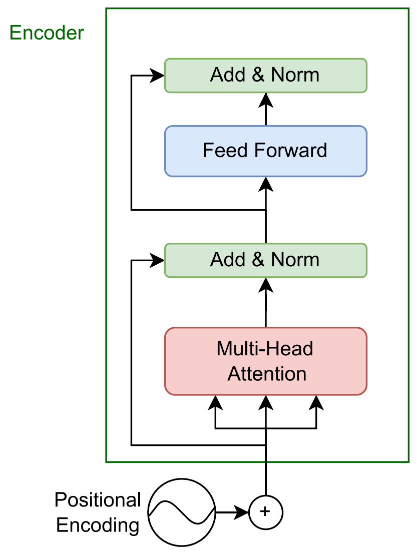


Figure 18. *Encoder structure*

### 6.1.8 Decoder

Given the $z = (z_1, z_2, z_3, ..., z_n)$ from the encoder, decoder generates an output sequence $y = (y_1, y_2, y_3, ..., y_m)$, one element at a time. At each step model is autoregressive [42], in other words, each prediction is based on previously predicted values. The first prediction is based on one value, which is the ***START*** token, and after that model predicts values one after another until it predicts ***END*** token or reaches a maximum length of the predicted sequence.

### 6.1.9 The attention mechanism

The main idea behind the transformer model is the attention mechanism. All the multi-head attention blocks are left without changes. The concept of attention applies the same way to the datasets described in this thesis because the values are tokenized, and embedded the same way as it is done in [1]. Analogously to NLP, where the self-attention mechanism is used to capture dependencies and relationships between words within the sentence, it captures the relations between the signal values within the sequence.

To capture the attention within the sequence three matrices are learned during the training process:

- Q - queries
- K - keys
- V - values

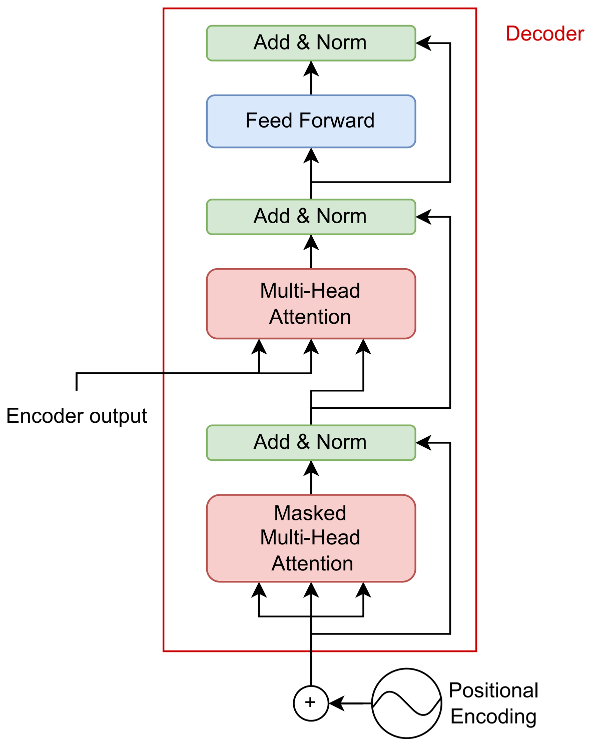


Figure 19. *Decoder structure*

The idea of queries, keys, and values comes from the database work. First, the query is matched against the keys, this is done by calculating the dot product of ***Q*** an ***K***. To normalize the compatibility of each query-key pair $softmax$ is used. To receive the attention scores resulting vector is then multiplied by the appropriate vector of values. In other words, values ***(V)*** are weighted by the query-key compatibility. All these calculations are packed into matrices for faster and more convenient computation (see formula 6.5).

The described method of calculating self-attention assumes it will capture only one type of relationship within the sequence. To make it capture different types of interactions aforementioned matrices are split into multiple heads, or even parts, which do the same thing. All these heads calculate their values in parallel, which allows them to reveal different dependencies within the sequence. That is why it is called multi-head attention.

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{6.5}$$

The transformer model has three different multi-head attention blocks:

- Encoder multi-head self-attention
- Decoder masked multi-head self-attention
- Encoder-decoder global attention

These three blocks are implemented all the same way. The difference comes from the task each block is aimed to solve.

### 6.1.10 Encoder multi-head self-attention

The encoder multi-head self-attention mechanism is aimed at revealing the dependencies and relations within the input sequence. This is done by calculating attention scores between each pair of tokens. Analogously to NLP transformer, where such self-attention

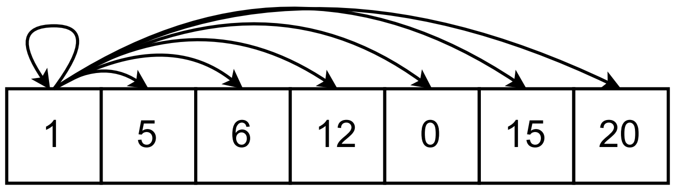


Figure 20. *Encoder Self-Attention example*

mechanism allows to capture the context of the whole sequence, similar attention can reveal specific patterns in signal depending on the whole sequence values (see figure 21)

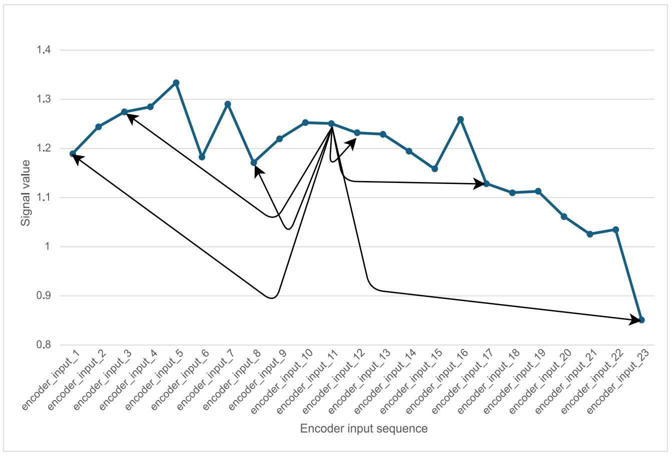


Figure 21. *Encoder Self-Attention signal example*

#### Decoder masked multi-head self-attention

The decoder self-attention mechanism. Due to the auto-regressive property of the model, the decoder self-attention is masked. That is at each point in time attention scores are calculated only for the preceding tokens (in reality all the attention scores are calculated but subsequent tokens are masked)

Such attention allows the model to reveal dependencies within the decoder input sequence, for the signal values, similar to the encoder, it helps to capture additional contextual information to predict the next token, which depends on the whole preceding sequence.

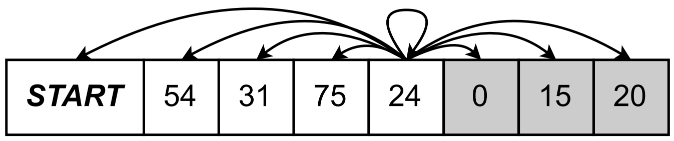


Figure 22. *Decoder masked self-attention example*

Masking subsequent tokens is required to prevent the model from cheating by looking at future tokens when generating new token (see figure 23).

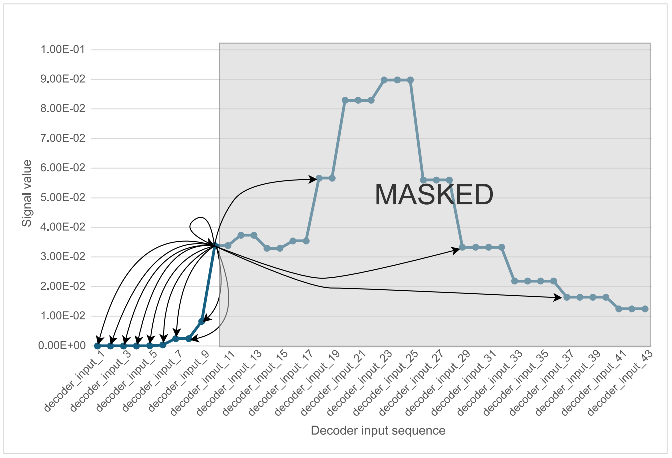


Figure 23. *Decoder masked self-attention signal example*

### Encoder-decoder global multi-head attention

The encoder-decoder multi-head attention in its turn allows the model to learn the relations between the input and output sequence. That is done by calculating queries using the decoder input and for keys and values the encoder output is used. This allows every position in the decoder to attend all the positions in the input sequence (see figure 24)

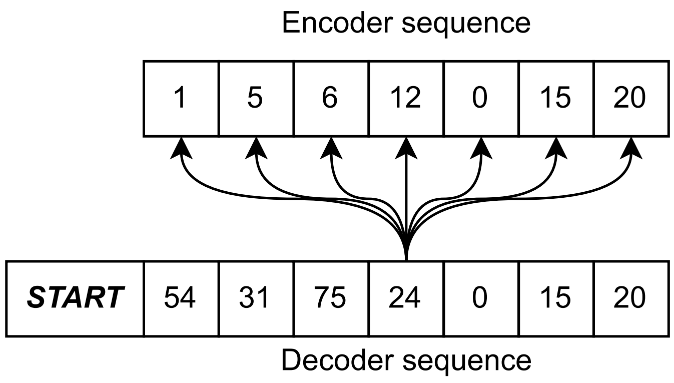


Figure 24. *Encoder-decoder global attention*

Analogously to previous attention layers, this attention layer is aimed at capturing the contextual information from the whole encoder output (input sequence which is encoded by the encoder) to predict the next token (see figure 25).

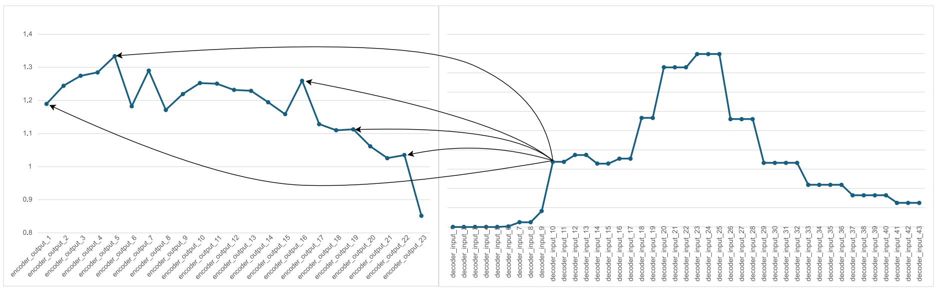


Figure 25. *Encoder-decoder global attention signal example*

During the training process, all of the described attention layers are learned using the loss function and target label. To be more specific linear transformations of the attention layer inputs are learned, these are the transformations that transform inputs into $Q$, $K$, and $V$ matrices.

### 6.1.11 Final linear layer

This is the final layer of the model, which finalizes all the calculations done before the next predicted output token. The main difference between the regular transformer used in NLP tasks and the one described in this thesis is this final projection layer (See Figure 26). The difference comes from the idea of using a transformer model like a regression model,

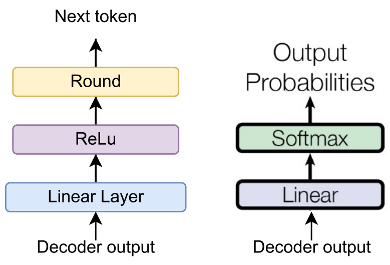


Figure 26. *Final neural network structure*

because of the way sequence values are mapped to tokens. To make the transformer work with the MSE loss (see section 6.2.4) final linear network architecture was changed to have 1 output neuron with a ReLu activation function followed (see figure 27). Instead of predicting probabilities for each token to be the next such a structure results in one floating point number output. As tokens are integer numbers and the prediction is a floating point number the rounding operation is applied to all predictions.

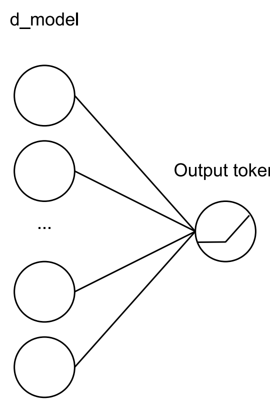


Figure 27. *Final neural network structure*

## 6.2 Model training

### 6.2.1 Train data and batching

Training data contains $41380 * 0.8 = 33104$ records, Train set size is a specified parameter, that can be changed in the configuration file. All the models described in chapter 7 were trained on 33104 records, which makes 80% of the total records, therefore 20% (8276 records) is left for the validation set.

Train batch size is also a hyperparameter, that can be specified in the model configuration file. Different batch sizes were tested, from 32 up to 256. Most of the models were trained on a batch size equal to 64.

### 6.2.2 Hardware and schedule

Multiple models were trained in a cloud on TalTech AI-LAB resources. Particularly model was trained on the next specification:

- Processor: AMD Threadripper 3970X 32-Core/64-thread Processor / AMD Threadripper 3960X 24-Core/48-thread Processor
- RAM: 128 GB of memory
- Video card: NVidia 3090 GPU with 24 GB of graphics memory
- OS: Ubuntu 22.04
- Nvidia driver version: 545
- Python version: 3.10.12

On average models with train batch size being equal to 64 take $\sim$16 hours to train for 5000 epochs. At the same time training time may drastically vary because of other parameters specified, such as $d_{model}$ and $d_{ff}$

An early-stop criterion was added, because of the specific behavior of the model - In some cases, validation loss measured during the training after stable reduction starts to grow. For such cases, there is a specific stopping criterion - if the current validation loss is higher than the previous minimum validation loss by more than $5\% \rightarrow$ stop training. Such a check is carried out every 500 epochs, which is a specified period for saving the model state. $5\%$ is required because the loss may slightly jump during training but have an overall descending trend.

### 6.2.3 Optimizer

The optimizer used for this transformer model is the same as in [1]. It is an Adam optimizer [43] with the following parameters: $\beta_1 = 0.9, \beta_2 = 0.98$ and $\epsilon = 10^{-9}$

The learning rate is varying over the training according to the next formula 6.6:

$$lrate = d_{model}^{-0.5} * \min(step\_num^{-0.5}, step\_num * warmup\_steps^{-1.5}) \tag{6.6}$$

Such a formula results in linearly increasing learning rate for the first $warmup\_steps$ training steps, and decreasing thereafter. $warmup\_steps$ used is 4000. After some experiments with the optimizer parameters, they were left without changes, due to providing a stable training process.

### 6.2.4 Loss function

To include the semantic meaning of the tokens into the model loss function was changed. As the original transformer produces output probabilities for each sequence unit to be a specific token, it uses a cross-entropy loss as a loss function. Even though cross-entropy loss increases as the predicted probability diverges from the actual label, it doesn't take into account the actual distance or similarity between the predicted and actual token.

Imagine two predictions made on the same observation in a case when using cross-entropy loss, the actual token is 10:

- Prediction: $10\%$ - token 5; $40\%$ - token 10; $50\%$ -token 20 $\rightarrow$ predicted token is 20, cross-entropy loss takes $40\%$ probability into account and results in some loss $x$. Note that the distance between the predicted token 20 and the actual token 10 is equal to 10.

- Prediction: 50% - token 5; 40% - token 10; 10% - token 20 $\rightarrow$, therefore, predicted token is 5, cross-entropy loss again takes 40% probability into account when calculating loss from the actual token - 10, and results in the same loss $x$, whereas the actual distance between the predicted and actual token is lower than it is in the previous case.

MSE or mean squared error loss function is one of the most common loss functions used for a regression problem. Note that during the training MSE loss is computed on token values, whereas in the validation stage, MSE loss is computed on detokenized signal values.

### 6.2.5 Hyperparameters

The model has many hyperparameters that can be tuned and affect the model performance:

- $d_{model}$ - depth of the embedding, meaning the length of the vector representing each single token.
- $d_{ff}$ - feed forward network hidden layer size.
- $h$ - Each attention layer is split into $h$ parallel blocks, which can learn different dependencies between the tokens.
- $ivs/ovs$ - power of the input/output sequence tokenizer. (Can be specified separately)
- $batch\_size$ - count of observations that the model is trained on in one iteration during the training stage.
- $epochs$ - number of passes of the entire training dataset through the model.

Fine-tuning each parameter is a time-consuming process because on average, a model with the train batch size equal to 64 is trained for 5000 epochs around 16 hours. Initial hyperparameter optimization can be observed in the Chapter 7.

# 7. Results

### Models

As the transformer model has many hyperparameters (see Section 6.2.5), models with different combinations of such parameters were trained to obtain multiple results and select the best model. In Table 1 there is a base model **1** and other models, all different from each other.

This is pretty simple hyperparameter optimization because each model differs by one hyperparameter alone, no combinations of two or more parameters at a time were tested.

| **Nr** | $ivs/ovs$ | $batch\_size$ | $d_{model}$ | $d_{ff}$ | $h$ | $epochs$ |
|---|---|---|---|---|---|---|
| **Base (1)** | 102 | 64 | 128 | 2048 | 1 | 5000 |
| **2** | 102 | 64 | 1024 | 2048 | 1 | 5000 |
| **3** | 102 | 64 | 2048 | 2048 | 1 | 1500 |
| **4** | 102 | 64 | 128 | 1024 | 1 | 2000 |
| **5** | 102 | 64 | 128 | 2048 | 8 | 1000 |
| **6** | 102 | 256 | 128 | 2048 | 1 | 500 |
| **7** | 202 | 64 | 128 | 2048 | 1 | 3000 |
| **8** | 402 | 64 | 128 | 2048 | 1 | 5000 |

Table 1. Transformer models trained

Notes:

- Values for the parameters are taken by intuition and logic from [1]. Most of the changes are multiplication/division by 2 or 4 for simpler comparison.
- Model **3** training process was suspended on 1500 epochs due to too long training
- Models **4,5,6** and **7** were trained for fewer epochs because of the early-stop criterion (see section 6.2.1).
- Model **8** didn't converge in 5000 epochs, possibly due to greater tokenizer powers.

### Models evaluation

To compare the results of different trained models, the mean value and three quartile values were calculated for the next evaluation metrics (see Table 2):

- $MSELoss_{validation}$ calculated between the predicted sequence and true sequence. Note that this is different from the $MSELoss_{train}$ used during model training because $MSELoss_{train}$ is calculated based on the token values of the sequences, but $MSELoss_{validation}$ is calculated using processed sequence values.

- Correlation coefficient *r* (RHO) - Pearson correlation coefficient calculated between predicted and true sequence.
- $\Delta_{max\_loc}$ - The difference between the predicted sequence peak value index and the true sequence peak value index. Example: true sequence peak value index = 11, predicted sequence peak value index = 16 $\rightarrow 16 - 11 = 5$
- $\Delta_{max\_value}$ - The difference between predicted sequence peak value and true sequence peak value. Analogous to the previous metric, but instead of index, the value itself is used.

As the last two metrics represent differences between two values, which can be both positive and negative, when measuring performance, the modulo of these values is taken into consideration. Note that for the metrics $MSELoss_{validation}$, $\Delta_{max\_loc}$, $\Delta_{max\_value}$ the lower the value - the better, therefore Q3 value for them is actually Q1. That is because the top 25% of observations have the lowes value for this metrics.

| **Nr** | **Metrics** | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **MSE** | | | **RHO** | | | $\Delta_{max\_value}$ | | | $\Delta_{max\_loc}$ | | |
| | AVG | Q2 | Q3 | AVG | Q2 | Q3 | AVG | Q2 | Q3 | AVG | Q2 | Q3 |
| **1** | 0.03 | 0.02 | 0.01 | 0.65 | 0.76 | 0.92 | 0.28 | 0.25 | 0.12 | 4.12 | 4 | 2 |
| **2** | 0.02 | 0.02 | 0.01 | 0.66 | 0.78 | 0.93 | 0.26 | 0.25 | 0.12 | 3.98 | 3 | 2 |
| **3** | 0.02 | 0.01 | 0.01 | 0.53 | 0.52 | 0.75 | 0.22 | 0.16 | 0.07 | 4.56 | 4 | 2 |
| **4** | 0.02 | 0.02 | 0.01 | 0.66 | 0.78 | 0.92 | 0.24 | 0.23 | 0.11 | 4.09 | 4 | 2 |
| **5** | 0.02 | 0.01 | 0.003 | 0.64 | 0.75 | 0.88 | 0.23 | 0.18 | 0.07 | 4.51 | 4 | 2 |
| **6** | **0.02** | **0.01** | **0.004** | **0.7** | **0.81** | **0.92** | **0.2** | **0.17** | **0.08** | **3.68** | **3** | **2** |
| **7** | 0.02 | 0.01 | 0.01 | 0.67 | 0.78 | 0.91 | 0.23 | 0.2 | 0.09 | 3.73 | 3 | 1 |
| **8** | 0.02 | 0.01 | 0.01 | 0.59 | 0.71 | 0.88 | 0.24 | 0.19 | 0.09 | 4.8 | 4 | 1 |

Table 2. Transformer models evaluation

A comparison of the base model and model nr. 2 shows that there is improvement in all the metrics, meaning a deeper model (each token represented by a deeper vector of values) is overall better. Model nr. 3 doesn't prove the aforementioned logic, but most likely, it is due to the suspended training.

Comparing the base model and model nr. 4 there is a slight improvement in RHO and $\Delta_{max\_value}$ metrics, which concludes that there is no need for such a deep hidden layer in feed-forward networks.

Increasing the parameter ***h*** to 8 is ambiguous, there is an improvement in some of the metrics, and in the same way there is a worsening in other metrics (model nr. **5**). Recall, that this parameter is the number of parallel attention mechanisms required to capture different types of contextual information from the sequence, therefore there is no clear linear relation between this parameter and model performance. In [1] authors have shown that too many heads can lead to worsening of the performance.

Model nr. **7** shows that an increase in the tokenizer power positively affects the model performance, there is an improvement in all the metrics. There is at least one drawback though, which is revealed in model nr. **8**, with the tokenizer power parameter increase - increases the set of possibly predicted tokens, therefore transformer needs more time to learn such mapping.

Based on the metrics provided in Table 2 model nr. **6** shows the most optimal performance. That is the model with a higher training batch size, and which was learning the least time. This model has exceptionally good performance in **RHO** metric, which is the correlation coefficient. To gain a more intuitive understanding of RHO metric and transformer performance, there are three examples of predicted and true sequences plotted with different RHO values (see Figure 28). The leftmost figure shows one of the best predictions, with the correlation between predicted and true sequence being equal to almost 1. The middle section shows average prediction, even though the correlation coefficient is still relatively high, the sequences are not so aligned. The rightmost section is dedicated to one of the worst predictions, where the RHO value is negative.

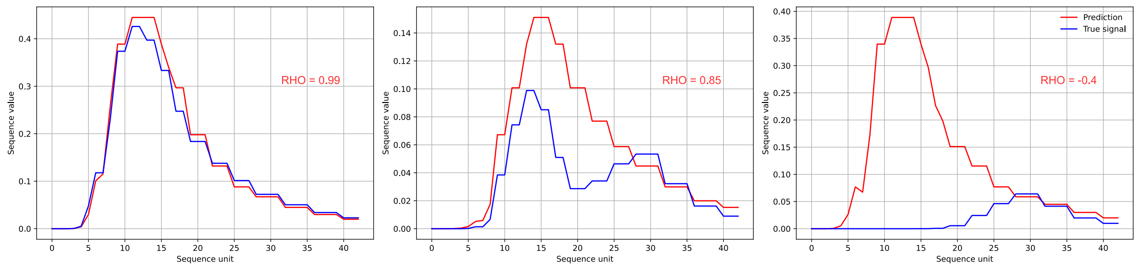


Figure 28. *Good, mid and bad prediction*

That said, looking at average correlation and all the quartile values results are impressive. Given only SAR image spectra as the input sequence simple one-pair encoder-decoder transformer is capable of estimating wave spectra sequences with an average correlation coefficient with true wave spectra of ∼0.7. At the same time median value and third quartile values of the correlation and not only are even more surprising. Half of the test observations were estimated with a 2-unit difference in peak value index along the sequence, and have more than 0.8 correlation with true sequence. A more detailed discussion of the implemented model can be found in the next chapter 8

# 8. Discussion

The implemented model has shown impressive and promising results, achieving a 0.7 average correlation coefficient metric. In [2] it is stated that LSTM achieved the MSE in the test set ∼0.12. The transformer model implemented for this task achieved an MSE of 0.02 on the test set, which is 6 times lower than the LSTM. This improvement is achieved by multiple attention mechanisms, which allow transformers to better capture long-term relationships within a sequence. However, the correlation coefficient of the transformer model is on an average level, and in this direction, the model may still be improved.

**Current implementation limitations**

Despite the implemented transformer model showed good results for the described task, it has its own limitations.

The first main limitation is the pre-processing stage in combination with the tokenizer. The Implemented tokenizer is strongly dependent on the dataset parameters, therefore strict pre-processing is required. The filtering stage ideally should be removed from the pre-processing flow. Current thresholds allowed to train the reasonably complex model with a relatively big training sample, but at the same time such a method of filtering might have filtered out all the observations of storms, heavy seas, or vice-versa complete calms, which makes the model less applicable for the real data.

Secondly, this was already mentioned, is the possible problem of logarithm-transformation, which makes different observations look much more alike. It can add confusion to the implemented transformer model. Given different input sequences, the output is nearly the same, whereas without logarithm-transformation it has ∼20 times lower values.

On the other hand, previous problems can be resolved with a more optimized tokenizer, which is not dependent so much on the data values. At the same time, the tokenizer **should** depend on the data, because the main idea is to use a transformer for regression, and not classification. Regression assumes that there is some meaning behind the values and they can be compared to each other. It also may be possible to completely avoid the tokenization layer and proceed with the initial numeric input, but that is up to research.

Another problem of the tokenizer, which was slightly described in Section 6.1.2 is the loss of information when performing tokenization-detokenization (multiple different values are mapped to one token, and one token maps back to one exact value). Even though this loss might be small, depending on the tokenizer power, it is still there. For very specific tasks this approach might not work at all because of the required precision. Seems that such loss is inevitable if a tokenizer is required for regression problem, therefore one should account for it when modeling specific problems.

One more limitation in the current implementation is more technical, which is related to the stack of encoders and decoders. Assuming that the problem described is less complex than language modeling, a structure with multiple encoders-decoders stacked is not implemented, and thus only one encoder-decoder pair is used. The problem might be much more complex than it is assumed and therefore more complex models are required.

**Future improvement ideas**

Because the current model is built mainly to prove the concept of using the transformer for numeric input and sequence-to-sequence regression problems, the model is simpler than in the original paper [1], where it is used to model the language. Therefore the straightforward improvement idea is to make the model more complex and observe how this affects the predictive power.

Coming back to the tokenizer - one interesting way to tokenize the data might be in a learnable tokenizer. The idea is to learn how to tokenize the data, it might be a neural network or some other machine learning algorithm. Thinking deeper, it may be possible to learn how to group some $n$-length parts of the sequence into specific tokens, where $n$ is not fixed over the sequence. At the same time, this may add more information loss to the whole process of tokenization-detokenization.

Thinking about more complex further possible improvements or research topics, one can use such sequence-to-sequence numerical transformer with other additional data to make more precise predictions. Another way is to fuse the additional data somehow into the transformer input data and apply special attention mechanisms to such variables. As an example with the described data, there is the possibility to add an additional token representing the wind direction and apply an attention mechanism for this token separately revealing the relations of other sequence values to the wind direction.

# 9. Conclusion

This thesis proposes the application of a deep learning transformer model for the estimation of Baltic Sea wave spectra from the spectra of SAR image data. The proposed method showed promising results. Applying a simple transformer with one encoder-decoder pair trained for 500 epochs achieved a 0.7 average correlation between the predicted and true wave spectra on the test set, given only SAR sequence data as an input.

Such results are obtained due to the attention mechanisms used in the transformer model. Their application allows the model to capture complex dependencies and relations within the sequences and between them. This helps better understand the context of the sequences and reveals long-term relations between the different wavelengths in the case of SAR spectra sequence and wave energy levels across different frequencies in the case of wave spectra.

The original transformer was modified in order to fit the numerical data into it and proceed with the training process. The main difference comes from the loss function required for regression problem as the estimated sequence is numeric. The final linear layer of the transformer was also changed to result in one token prediction instead of output probabilities. A transformer model with such a structure can be successfully applied to sequence-to-sequence task when the data is numeric.

The main outcome of the thesis is working deep learning transformer model, which can be used to estimate Baltic Sea wave spectra from the spectra of SAR image data. Application of such a model can be used in coastal engendering, maritime safety, environmental monitoring, and of course, commercial navigation.

However, it is important to acknowledge model limitations and the precision of the estimations. Multiple possible solutions for current model limitations and improve points/directions were suggested and discussed.

# 10. Acknowledgement

This work was partially supported by ETAG through the proof of concept grand EAG 249. I would like to thank my supervisor Sven Nõmm and co-supervisor Sander Rikka for their encouragement, inspiration, and for trusting me with such interesting and state-of-the-art research. Last but not least, I would like to express my deepest gratitude to my friends, family, and beloved girlfriend for their support and belief in me.

## References

[1] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in neural information processing systems* 30 (2017).

[2] Martin Simon et al. “Application of the LSTM Models for Baltic Sea Wave Spectra Estimation”. In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 16 (2023), pp. 83–88. DOI: `10.1109/JSTARS.2022.3220882`.

[3] NASA Jet Propulsion Laboratory. *Seasat Mission*. URL: `https://www.jpl.nasa.gov/missions/seasat`.

[4] W. Alpers and K. Hasselmann. *The two-frequency microwave technique for measuring ocean-wave spectra from an airplane or satellite*. 1978.

[5] Werner R. Alpers, Duncan B. Ross, and Clifford L. Rufenach. “On the detectability of ocean surface waves by real and synthetic aperture radar”. In: *Journal of Geophysical Research: Oceans* 86.C7 (1981), pp. 6481–6498. DOI: `https://doi.org/10.1029/JC086iC07p06481`. eprint: `https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/JC086iC07p06481`. URL: `https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/JC086iC07p06481`.

[6] MHB Thomas. “The estimation of wave height from digitally processed SAR imagery”. In: *International Journal of Remote Sensing* 3.1 (1982), pp. 63–68.

[7] Werner Alpers. “Monte Carlo simulations for studying the relationship between ocean wave and synthetic aperture radar image spectra”. In: *Journal of Geophysical Research: Oceans* 88.C3 (1983), pp. 1745–1759. DOI: `https://doi.org/10.1029/JC088iC03p01745`. eprint: `https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/JC088iC03p01745`. URL: `https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/JC088iC03p01745`.

[8] R. C. Beal, D. G. Tilley, and F. M. Monaldo. “Large-and small-scale spatial evolution of digitally processed ocean wave spectra from SEASAT synthetic aperture radar”. In: *Journal of Geophysical Research: Oceans* 88.C3 (1983), pp. 1761–1778. DOI: `https://doi.org/10.1029/JC088iC03p01761`. eprint: `https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/JC088iC03p01761`. URL: `https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/JC088iC03p01761`.

[9] K. Hasselmann et al. “Theory of synthetic aperture radar ocean imaging: A MARSEN view”. In: *Journal of Geophysical Research: Oceans* 90.C3 (1985), pp. 4659–4686. DOI: `https://doi.org/10.1029/JC090iC03p04659`. eprint: `https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/JC090iC03p04659`. URL: `https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/JC090iC03p04659`.

[10] Werber R. Alpers and Claus Bruening. “On the Relative Importance of Motion-Related Contributions to the SAR Imaging Mechanism of Ocean Surface Waves”. In: *IEEE Transactions on Geoscience and Remote Sensing* GE-24.6 (1986), pp. 873–885. DOI: `10.1109/TGRS.1986.289702`.

[11] M Marom et al. “Remote sensing of ocean wave spectra by interferometric synthetic aperture radar”. In: *Nature* 345.6278 (1990), pp. 793–795.

[12] Klaus Hasselmann and Susanne Hasselmann. “On the nonlinear mapping of an ocean wave spectrum into a synthetic aperture radar image spectrum and its inversion”. In: *Journal of Geophysical Research: Oceans* 96.C6 (1991), pp. 10713–10729. DOI: `https://doi.org/10.1029/91JC00302`. eprint: `https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/91JC00302`. URL: `https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/91JC00302`.

[13] Harald E. Krogstad. “A Simple Derivation of Hasselmann’s Nonlinear Ocean-Synthetic Aperture Radar Transform”. In: *Journal of geophysical research* 97 (1992).

[14] Harald E Krogstad, Oddgeir Samset, and Paris W Vachon. “Generalizations of the non-linear ocean-SAR transform and a simplified SAR inversion algorithm.” In: *Atmosphere–Ocean (Canadian Meteorological & Oceanographic Society)* 32.1 (1994).

[15] G. Engen and H. Johnsen. “SAR-ocean wave inversion using image cross spectra”. In: *IEEE Transactions on Geoscience and Remote Sensing* 33.4 (1995), pp. 1047–1056. DOI: `10.1109/36.406690`.

[16] S. Hasselmann et al. “An improved algorithm for the retrieval of ocean wave spectra from synthetic aperture radar image spectra”. In: *Journal of Geophysical Research: Oceans* 101.C7 (1996), pp. 16615–16629. DOI: `https://doi.org/10.1029/96JC00798`. eprint: `https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/96JC00798`. URL: `https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/96JC00798`.

[17] J. Schulz-Stellenfleth, T. König, and S. Lehner. "An empirical approach for the retrieval of integral ocean wave parameters from synthetic aperture radar data". In: *Journal of Geophysical Research: Oceans* 112.C3 (2007). DOI: https://doi.org/10.1029/2006JC003970. eprint: https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/2006JC003970. URL: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2006JC003970.

[18] He Wang et al. "A semiempirical algorithm for SAR wave height retrieval and its validation using Envisat ASAR wave mode data". In: *Acta Oceanologica Sinica* 31 (2012), pp. 59–66.

[19] AL Pleskachevsky, Wolfgang Rosenthal, and Susanne Lehner. "Meteo-marine parameters for highly variable environment in coastal regions from satellite radar images". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 119 (2016), pp. 464–484.

[20] Weizeng Shao et al. "Ocean wave parameters retrieval from Sentinel-1 SAR imagery". In: *Remote Sensing* 8.9 (2016), p. 707.

[21] Justin E Stopa and Alexis Mouche. "Significant wave heights from S entinel-1 SAR: Validation and applications". In: *Journal of Geophysical Research: Oceans* 122.3 (2017), pp. 1827–1848.

[22] Chen Wang et al. "Automated Geophysical Classification of Sentinel-1 Wave Mode SAR Images Through Deep-Learning". In: *IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium*. 2018, pp. 1776–1779. DOI: 10.1109/IGARSS.2018.8518354.

[23] Sude Bao et al. "Detection of ocean internal waves based on Faster R-CNN in SAR images". In: *Journal of Oceanology and Limnology* 38.1 (2020), pp. 55–63.

[24] David Bazzett, Behzad Golparvar, and Ruo-Qian Wang. "Using SAR satellite imagery, HF radar, and machine learning to estimate significant wave height along the new jersey coast". In: *AGU Fall Meeting Abstracts*. Vol. 2020. 2020, OS037–0014.

[25] Brandon Quach et al. "Deep learning for predicting significant wave height from synthetic aperture radar". In: *IEEE Transactions on Geoscience and Remote Sensing* 59.3 (2020), pp. 1859–1867.

[26] Weizeng Shao et al. "Wave Retrieval Under Typhoon Conditions Using a Machine Learning Method Applied to Gaofen-3 SAR Imagery". In: *Canadian Journal of Remote Sensing* 45.6 (2019), pp. 723–732. DOI: 10.1080/07038992.2019.1683444.

[27] Saidul Islam et al. *A Comprehensive Survey on Applications of Transformers for Deep Learning Tasks*. 2023. arXiv: 2306.07303 [cs.LG].

[28] Pujan Pokhrel et al. "A Transformer-Based Regression Scheme for Forecasting Significant Wave Heights in Oceans". In: *IEEE Journal of Oceanic Engineering* 47.4 (2022), pp. 1010–1023. DOI: 10.1109/JOE.2022.3173454.

[29] Meisam Amani et al. "Ocean Remote Sensing Techniques and Applications: A Review (Part I)". In: *Water* 14.21 (2022). ISSN: 2073-4441. DOI: 10.3390/w14213400. URL: https://www.mdpi.com/2073-4441/14/21/3400.

[30] Gayathri K Devi, BP Ganasri, and GS Dwarakish. "Applications of remote sensing in satellite oceanography: A review". In: *Aquatic Procedia* 4 (2015), pp. 579–584.

[31] Mohammad Haji Gholizadeh, Assefa M Melesse, and Lakshmi Reddi. "A comprehensive review on water quality parameters estimation using remote sensing techniques". In: *Sensors* 16.8 (2016), p. 1298.

[32] PJ Minnett et al. "Half a century of satellite remote sensing of sea-surface temperature". In: *Remote Sensing of Environment* 233 (2019), p. 111366.

[33] Anne G O'carroll et al. "Observational needs of sea surface temperature". In: *Frontiers in Marine Science* 6 (2019), p. 420.

[34] Sahel Mahdavi et al. "A probability-based daytime algorithm for sea fog detection using GOES-16 imagery". In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 14 (2020), pp. 1363–1373.

[35] Nicolas Baghdadi and Mehrez Zribi. *Microwave remote sensing of land surfaces: techniques and methods*. Elsevier, 2016.

[36] Chen Wang et al. "Automated geophysical classification of sentinel-1 wave mode sar images through deep-learning". In: *IGARSS 2018-2018 IEEE International Geoscience and Remote Sensing Symposium*. IEEE. 2018, pp. 1776–1779.

[37] Øyvind Breivik et al. "The Impact of a Reduced High-Wind Charnock Parameter on Wave Growth With Application to the North Sea, the Norwegian Sea, and the Arctic Ocean". In: *Journal of Geophysical Research: Oceans* 127.3 (Mar. 2022). DOI: 10.1029/2021jc018196. URL: https://doi.org/10.1029%5C%2F2021jc018196.

[38] *Norwegian Meteorological Institute, spectra database*. https://thredds.met.no/thredds/catalog/windsurfer/mywavewam3km_spectra/catalog.html. Accessed: 2022.

[39] *The Annotated Transformer*. URL: https://nlp.seas.harvard.edu/annotated-transformer/.

[40] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: `1512.03385 [cs.CV]`.

[41] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. *Layer Normalization*. 2016. arXiv: `1607.06450 [stat.ML]`.

[42] Alex Graves. *Generating Sequences With Recurrent Neural Networks*. 2014. arXiv: `1308.0850 [cs.NE]`.

[43] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: `1412.6980 [cs.LG]`.

# Appendix 1 – Non-Exclusive License for Reproduction and Publication of a Graduation Thesis[1]

[1]The non-exclusive licence is not valid during the validity of access restriction indicated in the student's application for restriction on access to the graduation thesis that has been signed by the school's dean, except in case of the university's right to reproduce the thesis for preservation purposes only. If a graduation thesis is based on the joint creative activity of two or more persons and the co-author(s) has/have not granted, by the set deadline, the student defending his/her graduation thesis consent to reproduce and publish the graduation thesis in compliance with clauses 1.1 and 1.2 of the non-exclusive licence, the non-exclusive license shall not be valid for the period.