Automated robust first peak detection in a time signal using computational intelligence

Information Technology (M.Eng.)
Computational Intelligence
Deblina Karmakar
Matriculation Number - 1427365
deblina.karmakar@stud.fra-uas.de

Information Technology (M.Eng.)
Computational Intelligence
Indranil Saha
Matriculation Number - 1427190
indranil.saha@stud.fra-uas.de

Information Technology (M.Eng.)
Computational Intelligence
Kumar Satyam
Matriculation Number - 1441354
kumar.satyam@stud.fra-uas.de

***Abstract*— Reliable peak detection in a time signal is a fundamental task in various fields including signal processing, chromatography, and biomedical applications. The main goal of peak detection techniques is to find peaks in the chosen signal and determine their locations, heights, widths, and regions. However, common peak detection approaches including threshold-based methods, derivative-based methods, local maxima/minima detection, and others are used to locate and approximatively quantify peaks in data. It usually does not make any explicit assumptions about the geometry of the peaks. In this project, our approach leverages the power of computational intelligence and proposes a method of using convolution neural network to address the challenges associated with noisy and complex signals. It also ensures robust performance across a wide range of applications which makes it versatile for real-life scenarios as well.**

***Index Terms*—Convolutional Neural Network, Fast Fourier Transform, Inverse Fast Fourier Transform, Hilbert Transform.**

I. INTRODUCTION

Detecting signal peaks constitutes a pivotal step in virtually all signal processing applications, particularly when the extraction of key features or comprehensive signal analysis is a requisite task. Peaks, within the context of signal processing, serve as vital indicators, highlighting significant events, anomalies, or specific data points of profound interest. The primary objective of our project is to establish a robust and dependable system tailored for the precise identification of peaks within mass spectral data. This system is designed to be user-friendly, requiring minimal user input, while surpassing existing standards for peak identification in terms of accuracy and efficiency.

Our project's expected outcome aims to align closely with the results achieved through manual peak detection in terms of comprehensiveness and accuracy. Achieving this goal requires a systematic approach, involving several critical steps in the peak detection process.

The first step entails selecting an appropriate peak detection algorithm, a decision often contingent upon various factors, including noise levels present in the data and the nature of the signal (continuous or discrete). Subsequently, signal preprocessing becomes imperative to mitigate noise and enhance peak visibility. Noise, acting as an interfering factor, can significantly degrade the quality of a signal, necessitating its reduction to attain a clear and unambiguous view of the peaks.

An important consideration in the peak detection process is the establishment of appropriate threshold levels. These thresholds serve as filters to eliminate insignificant signal fluctuations, ensuring that only relevant peaks are identified. The determination of the threshold can be executed in either a static or dynamic manner, depending on the specific objectives of the project.

In our project, we employ a sophisticated approach that integrates Convolutional Neural Networks (CNNs) with Fast Fourier Transform (FFT) for peak detection. Additionally, we leverage the Hilbert Transform to assist in pinpointing the peaks within the envelope signal. The term "envelope signal" denotes a smooth representation of the signal, capturing variations in amplitude over time. This approach effectively directs our attention to the primary peaks and valleys within the original signal, facilitating the identification and examination of critical signal features.

In summary, our project is driven by the objective of enhancing peak detection in mass spectral data through the utilization of advanced techniques and methodologies. By automating and streamlining this process, we aim to provide a powerful tool that outperforms current industry standards and enables researchers and analysts to extract meaningful insights from complex data with minimal user intervention.

In the past few decades, Convolutional Neural Network (CNN) has played an essential role in the field of computer vision [1]. CNN have significant impact on several audio and music processing tasks as well. 1D CNNs learn acoustic models directly from audio waveforms are becoming a popular method in audio processing. These networks have the ability to take advantage of the audio signal's fine time structure [2]. There are several applications of 1D CNN in today's world. A few of them include sound event detection, music analysis, and speech recognition.

II. METHODOLOGY

The experiment of automated robust first peak detection in a time signal using computational intelligence has been conducted by following the flowchart diagram (Figure 1).

It begins with calculating Fast Fourier Transform (FFT), converting the time domain signal to frequency domain signal. The results are obtained by appropriate modification of the Fourier transform coefficients and taking the complex conjugate of it to eliminate the imaginary values. Secondly, the process of filtering the noisy data takes place, where peaks of the signal above the noise floor are kept and everything below it is removed. After that, Inverse Fast Fourier Transform (IFFT) is performed to de-noise the signal. Further filtering is done by selecting all the values with a squared magnitude less than 1.5. A power spectrum is then calculated, which is subsequently multiplied by an index vector. This multiplication zeroes out indices with small Fourier coefficients, retaining only those values with a power greater than 1.5. The signal is then filtered by inverse Fourier transform. Then, Hilbert transform(HT) is used to extract complex components from a signal. HT generates the envelope of the time signal by calculating the magnitude of the time function. Consequently, the peaks are calculated and the real values of the same are saved to group them into the same window. Thereafter, an 1D CNN network model is created, and the input is fed into CNN model with supervised learning technique. The training process with a CNN model necessitates fine-tuning its weights based on the training and validation data. Finally, the trained CNN model is used for peak detection.

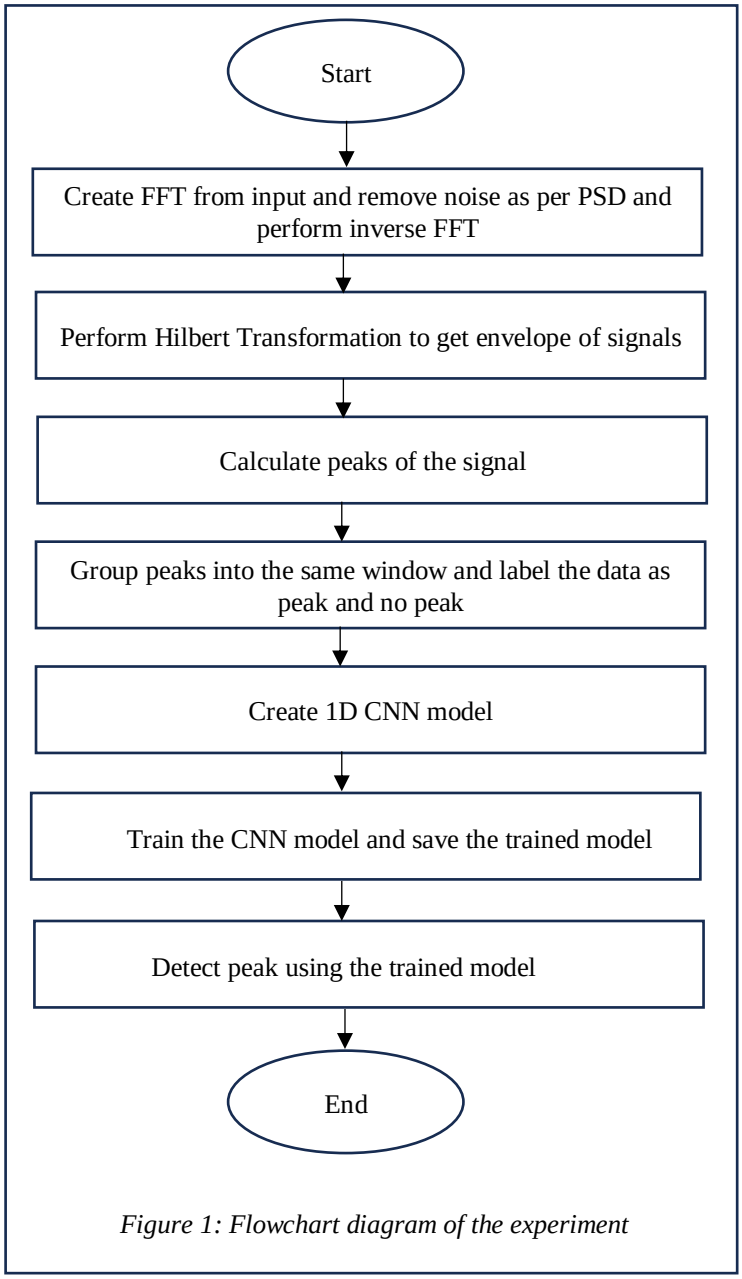

Figure 1: Flowchart diagram of the experiment

A. *Preprocessing*

Preprocessing is an important method to pre-process the input data. All the input variables from the dataset are fetched, normalized, de-correlated and then transformed to a neural network.

B. *First Fourier Transform:*

Preprocessing of the data includes performing FFT operation on time signal data. Fourier transform is one of the most widely used techniques in applied mathematics and engineering from image analysis to signal processing. In essence, the Fourier transform operates on situations where functions with specific characteristics can be expressed as a

sum of trigonometric functions. These trigonometric components serve as a means to extract frequency-related details from a time-domain signal, offering alternative and often more efficient approaches for analyzing or altering signals [3]. The Fast Fourier transform (FFT) algorithm was given by James Cooley and John Tukey in 1965 which has enabled a wide range of numerical applications since then. These applications include data compression, handling noisy signals, analysis of crystal structures, and solving partial differential equations [4]. The DFT equation $\boldsymbol{X(k) = \sum x(n) W_N^{nk}}$ is decomposed into several short transforms and recombined in the FFT formula. Within the FFT process, the initial input signal undergoes transformation into the frequency domain through the application of the Discrete Fourier Transform (DFT). It is subsequently convolved with the frequency response of the filter and, in the final step, translated back into the time domain using the inverse DFT operation. Fourier transform (FFT) and inverse fast Fourier transform (IFFT) ease the computational complexity of the DFT from O(N^2) to O (N log N). A sinusoidal signal in the time domain reveals its individual frequency components when subjected to a Fourier transform. Fourier transforms are versatile tools applicable to both periodic and non-periodic signals, enabling the seamless transition from the time domain to the frequency domain.

C. Hilbert Transform

Hilbert Transform is a mathematical operation which provides a way to analyse the phase and amplitude of a complex signal. It is named after the German mathematician David Hilbert and used widely all over the world in signal domain now. The Hilbert transform can be a filter which simply shifts phases of all frequency components of its input by -π/2 radians. The Hilbert transform of a function f(x) is defined by:

$$F(t) = \frac{1}{\pi}\int_{-\infty}^{\infty} \frac{f(x)}{t-x} dx$$

Computationally it can be one can write the Hilbert transform as the convolution:

$$F(t) = \frac{1}{\pi t} \cdot f(t)$$

Which, by the convolution theorem of Fourier transforms, may be evaluated as the product of the transform of f(x) with -i*sgn(x), where:

$$\operatorname{sgn}(x) = \begin{cases} -1 & x < 0 \\ 0 & x = 0 \\ 1 & x > 0 \end{cases}$$

[5]. The scientific computing software MATLAB has a Hilbert () function that "computes the so-called discrete-time analytic signal X = Xr + i*Xi such that Xi is the Hilbert transform of Xr" [6]. In MATLAB's implementation of the Hilbert () function takes advantage of the fast Fourier transform (FFT).

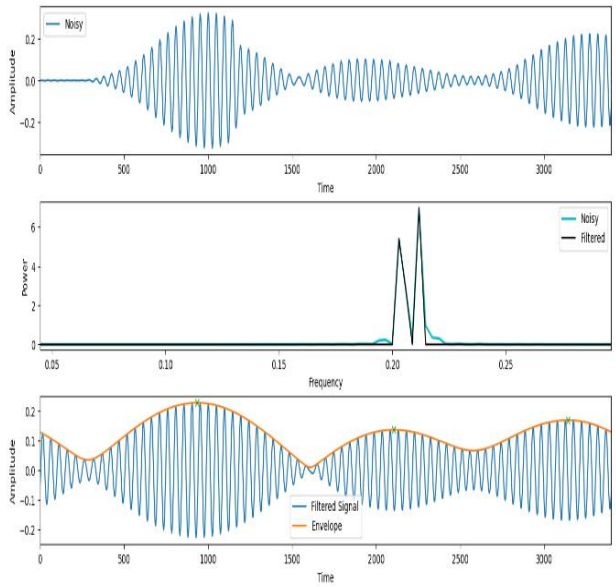

Figure 2: Upper enveloping by Hilbert Transform

The calculation is carried out in three phases by the Hilbert () function - Start by doing the FFT of Xr. Then, make all of the FFT elements that correspond to frequency $-\pi < \omega < 0$ to zero. Last step would be performing the Inverse FFT.

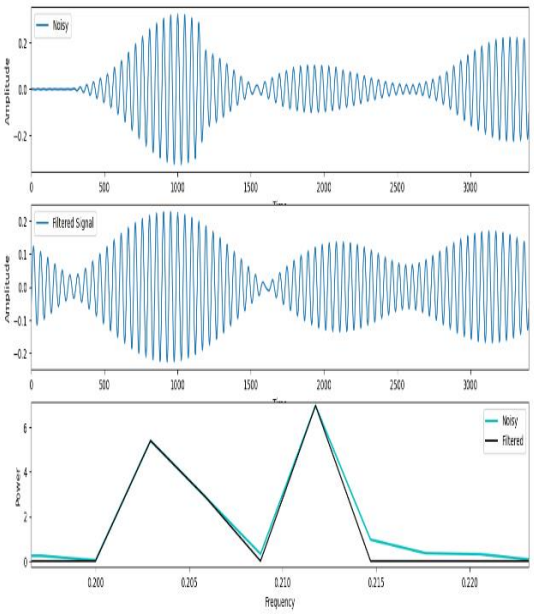

Figure 3: Filtering signal using FFT

Figure 3 illustrates how the Hilbert transformer is advantageous for extracting the envelope. This is explained by the formula that states the Hilbert transform yields a sin(t) for each cos(t) as HT provides a ±90° phase shift to the input signal. HT can produce the time signal's envelope by determining the time function's magnitude. The gradual overall change and sound intensity over time are analyzed using envelopes.

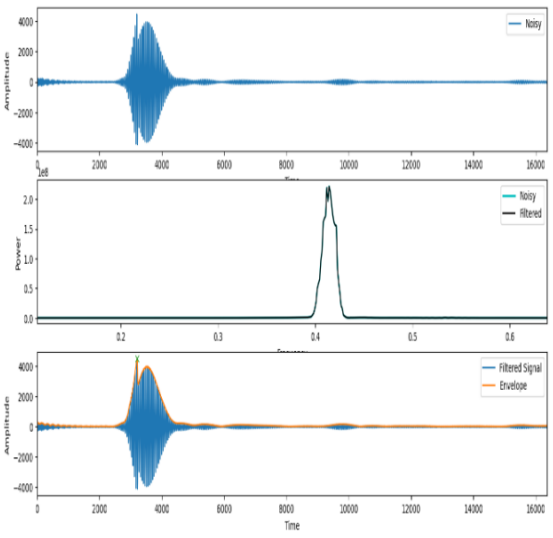

Figure 4: Performing FFT and Hilbert Transform

Since the real signal may be analytically extended from the real axis to the upper half of the complex plane, the Hilbert transform can be used to identify an imaginary function to a real valued signal. The outcome is as shown in figure 4.

D. Training and Testing

During the preprocessing stage, FFT is used for noise removal. Consequently, the signals are fed to the CNN method for peak detection. In the experiment, the supervised learning has been used to train the 1D CNN model. The neural network has been trained by us with the peak from preprocessing. The CNN model then detects the peak from trained model. Numerous conventional CNN models are readily accessible in internet. One of the models from those listed on the Keras website has been selected and made slight modifications to align it with our specific problem. In the context of 1D CNNs, these models typically take raw data as input, as opposed to relying on manually crafted features. The input data is passed through various trainable convolutional layers, where it learns to form a suitable representation of the input. In accordance with the local connectivity theorem, neurons within a layer establish connections exclusively with a limited region of the preceding layer, known as a receptive field. The input to a 1D CNN is an array representation of the audio waveform, which is denoted as X. Set of parameters Θ is used to map the input to the prediction by the equation,

$$T = F(X \mid \Theta) = fL(...f2(f1(X \mid \Theta 1) \mid \Theta 2) \mid \Theta L) \ldots\ldots (1)$$

Where L is the hidden number of layers in the network. For the convolutional layers, the operation of the layer 1 can be expressed as:

$$Tl = fl(Xl \mid \Theta l) = h(W \otimes Xl + b), \Theta l = [W, b] \ldots\ldots (2)$$

where ⊗ is the convolution operation, Xl is two-dimensional input matrix in N fields, W is a set of one-dimensional kernels used for extracting a new set of features from the input array, b defines bias vector, and h() is activation function. The shapes of Xl, W and Tl are denoted as (N, d), (N, m) and

(N, d − m + 1) respectively. For increasing the area covered by the next receptive fields, several pooling layers are applied between the convolutional layers. The output of the final convolutional layer is then used as input of several fully connected layers, which can be described as:

$$Tl = fl(Xl \mid \Theta l) = h(W\ Xl + b),\ \Theta l = [W, b] \ldots\ldots.(3)$$

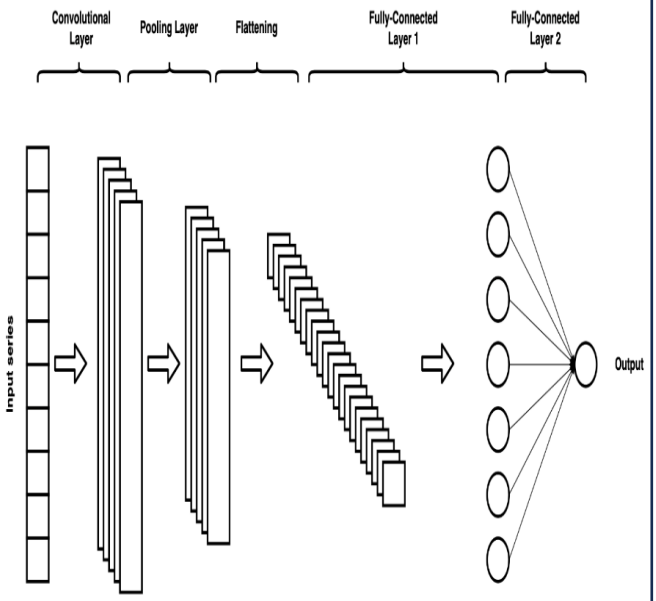

Figure 5: Simplified schema of 1D CNN model

1D Convolutional Neural Networks (CNNs) have the ability to combine nearby features and reduce the data's dimensionality by leveraging convolutional operations and spatial pooling techniques.

Here, the convolutional layers analyze a 1D time series by sliding a kernel across it. This kernel specifies the features we aim to identify within the time series and is designed to have the same width as the series itself, allowing it to move in only one direction. At each step of this movement, the input data is convolved with the kernel by element-wise multiplication. Subsequently, a non-linear activation function is applied to the outcome of this convolution operation. As a result, the initial input time series is transformed into a representation known as the filter map, which captures the relevant patterns and information from the original data. Following the convolutional layers, the next phase involves the application of a pooling layer. This layer serves the dual purpose of reducing the dimensionality of the series while retaining the essential features that have been detected. It's possible to stack multiple convolutional and pooling layers sequentially. The output of the pooling layer can then be fed into a fully connected layer for further processing.

First 1D CNN layer: To specify a filter in the top layer, which can also be referred to as a feature detector, you need to define its kernel size. When aiming for the neural network to learn a single feature in the initial layer, it's sufficient to define just one filter. In this case, the output matrix will consist of the weights associated with this single filter, with each column representing the weights for that specific filter.

Second 1D CNN layer: The second CNN layer will be fed the output of the first CNN, using the same reasoning as the top layer.

Max pooling layer: After the CNN layers, a pooling layer is frequently employed to simplify the output and avoid overfitting the data.

Third and fourth 1D CNN layer: To capture more complex and higher-level features, an additional layer of 1D CNNs are employed.

Average pooling layer: To mitigate the risk of overfitting, an additional pooling layer that utilizes average pooling is introduced. Unlike previous layers that used maximum pooling, this layer calculates the average value of two weights within the neural network. Consequently, in this specific layer, only one weight remains for each feature detector.

Dropout layer: In the dropout layer, neurons within the network are randomly assigned a weight of 0. When a dropout rate of 0.5 is chosen, it means that 50% of the neurons will have their weights set to 0 during each forward pass. This dropout mechanism serves to decrease the network's sensitivity to minor variations in the input data, which, in turn, is expected to enhance the overall accuracy of our data processing.

Fully connected layer: In the final layer, the height vector undergoes a reduction in dimensionality. This reduction is achieved through another matrix multiplication operation.

E. Evaluation

Following the CNN model's training, we evaluated its performance by examining its predictions in relation to actual data through the use of a confusion matrix. This matrix consists of four key values. True Positive (TP) represents the effectiveness of the model in correctly predicting positive outcomes compared to the actual positive cases. Conversely, False Positive (FP) quantifies the instances where the model incorrectly predicted positive outcomes. True Negative (TN) assesses the model's ability to accurately predict negative outcomes in accordance with the actual negative cases. False Negative (FN) is the value assigned to inaccurately foretelling negative outcomes.

Accuracy: Accuracy is a metric used to gauge the CNN model's prediction performance. It is calculated by dividing the number of correct predictions by the total number of predictions made. The formula for accuracy can be expressed as follows.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$

Precision: Precision is a measure of how well the CNN model can predict positive predictions overall, and how many of those positive predictions are right. A more accurate positive forecast results from greater precision.

$$Precision = \frac{TP}{TP+FP}$$

Recall: Similar to accuracy, recall counts the correctly optimistic predictions made from the right ones. Increasing recall value reduces the number of inaccurate predictions.

$$Recall = \frac{TP}{TP+FN}$$

F1-Score: F1-Score requires recall and precision. It strikes a balance between the two metrics. F1-Score is helpful when the model outputs high FP and FN.

$$F1\text{-}Score = \frac{2\times Precision\times Recall}{Precesion+Recall}$$

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d (Conv1D)             (None, 16365, 64)         256

 conv1d_1 (Conv1D)           (None, 16363, 64)         12352

 dropout (Dropout)           (None, 16363, 64)         0

 max_pooling1d (MaxPooling1D  (None, 3272, 64)         0
 )

 flatten (Flatten)           (None, 209408)            0

 dense (Dense)               (None, 1052)              220298268

 dense_1 (Dense)             (None, 263)               276939

=================================================================
Total params: 220,587,815
Trainable params: 220,587,815
Non-trainable params: 0
_________________________________________________________________
```

Figure 6: CNN layer working procedure

III. EXPERIMENT RESULT

We have randomly chosen a sample from the given datasets and found the result (figure 7). We found the accuracy is around 30-40%.

Sample 1: First input sample can be found into GitHub repository at the location: '\static\files\sample_input_001.xlsx'

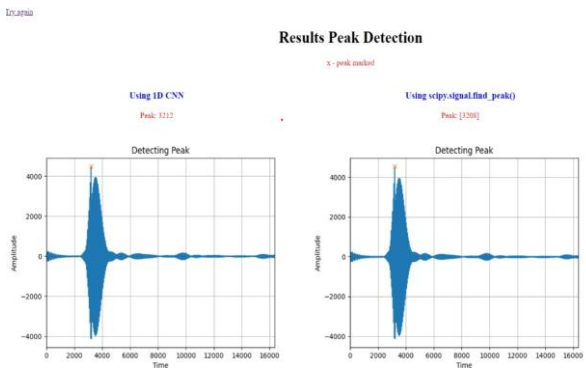

Figure 7: Result from the sample

IV. ENCOUNTERED PROBLEMS

While the assessment of our training experiments initially yielded promising results, after conducting the experiment a hundred times, we encountered unexpected drops in performance in just two instances out of a hundred. The exact cause of this decline remains elusive, but we have formulated a hypothesis suggesting that additional information regarding the signal may be necessary to achieve noise-free signal processing. Additionally, we encountered a separate issue where the dataset underwent changes in its dimensions during the final stages of the project.

V. DISCUSSION AND CONCLUTION

The proposed neural network based peak detector exploits peak shape information successfully with a superior performance in all evaluated signals. Furthermore, it is shown that after grouping, it needs approximation to match actual results that doesn't give high actual peak, but a window in which peak is present. To improve accuracy, it is required to make small window which matches kernel size of CNN. Current window size is 62 which cannot be commented as good or bad since the sampling rate is unknown.

REFERENCES

[1] H. Pan, Y. Chen, X. Niu, W. Zhou and D. Li, "Learning Convolutional Neural Networks in the Frequency Domain," Changsha, China, 2022.

[2] Hoshen Y., Weiss R. J. and Wilson K. W., "Speech acoustic modeling from raw mutichannel waveforms," IEEE International Conference on Acoustic, Speech and Signal Processing (ICASSP), 2015.

[3] Bruce A., Donoho D. and Gao H. Y., "Wavelet analysis [for signal processing]," *IEEE spectrum*, 1996.

[4] Osgood B. G., "Lectures on the Fourier Transform and it's application," American Mathematical Society, 2019.

[5] Bracewell R., The Fourier Transform and it`s application, McGraw-Hill, 1965.

[6] King F. W., Hilbert Transform, Vol. 2, Cambridge, UK: Cambridge University Press, 2009.