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

Information Technology (M.Eng.)
Computational Intelligence
Md Rizwanul Islam
Matriculation 1396438
md.islam9@stud.fra-uas.de

Information Technology (M.Eng.)
Computational Intelligence
Mandar Anil Patkar
Matriculation 1390942
mandar.patkar@stud.fra-uas.de

Information Technology (M.Eng.)
Computational Intelligence
Nishat Shama
Matriculation 1386147
nishat.shama@stud.fra-uas.de

Information Technology (M.Eng.)
Computational Intelligence
SM Mehedi Hasan
Matriculation 1344008
s.hasan@stud.fra-uas.de

Information Technology (M.Eng.)
Computational Intelligence
Chandan Dev Singh
Matriculation 1390269
chandan.singh@stud.fra-uas.de

***Abstract*— a common requirement in scientific data processing is detecting peaks in a signal and measuring their positions, heights, widths, and/or areas. Standard peak detection techniques do not intend to provide details about the anticipated peak shape. Because of this, ordinary detectors ignore this crucial information and don't function at their best in relation to the given possibilities. A detector based on has been developed and assessed. A pattern recognition tool called convolution neural network recognition in order to make use of the peak shape data. This paper proposes a method for peak detection in a time signal using (1D) (CNN) and the Fast Fourier Transformation (FFT).**

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

I. INTRODUCTION

In many signals processing applications, finding signal peaks is a crucial step. Up to now, a variety of techniques have been created, including those based on conventional the window-threshold approach, wavelet transform, Hilbert transform, combining Hilbert and wavelet transform, artificial neural networks. The goal of our research was to create a system for robustly and reliably recognizing peaks in mass spectral data that would be easy to use, need little user input, outperform existing standard methods for peak identification, and produce findings that were more similar in scope to manual peak detection. Additionally, we sought to create a technique that would be quick enough to fit into already in place workflows. The parts of a spectrum known as the signal are those from which it is possible to extract relevant information. Contrarily, noise is defined as those components that do not carry meaningful information. The signal must have sufficient intensity to stand out from the background noise in order to be noticed. Understanding a threshold level that will be used to indicate whether a point is thought to be more likely to result from signal or more likely to result from noise is essential for being able to extract information from a mass spectrum in two ways: First, since the areas of the peak will be detected using this threshold (since they will be above it), and second, because the relative height of the peak will be described in terms of the signal to noise ratio (SNR). In this paper, we propose a CNN-based peak determination approach. For peak identification, transform-based methods are also suggested. These approaches make use of Hilbert transform and other signal processing techniques. Most de noising approaches use wavelet-based methods. These methods divide signals into several sub bands using the same resolution as the original signals. Afterward, the informative sections are created again and baseline drifts and high-frequency (HF) sounds are removed by constructing the desired sub-bands. For peak detection tasks, the Hilbert transform is also used. The immediate analysis of a signal's amplitude and frequency using the Hilbert transform is very effective. Convolutional Neural Networks (CNN) have significant impact on several audio and music processing tasks. 1D CNNs learn acoustic models directly from audio waveforms are becoming a popular method in audio processing because these networks have the ability to take advantage of the signal’s fine time structure [1]. End-to-end applications using 1D CNNs are becoming common in signal processing because of their ability to benefit from its structure that can learn from waveforms directly [2]. Zhu et al. (2016) proposed an end-to-end learning approach for speech recognition, where multiscale convolutions learns the representation directly from audio waveforms. Three 1D convolutional layers are used with different kernel sizes for feature extraction. Where the features are links together in a series by a pooling layer which ensures a consistent sampling frequency for the rest of the network. They reported with 23.28% of word error rate from a collection of datasets including read, conversational, accented, and noisy speech [3]..

II. METHODOLOGY

The experiment of automated robust first peak detection in a time signal by CNN has been conducted following the flowchart diagram (Figure 1). For the sake of simplicity, the experiment is divided into parts. The process of experiment begins with calculating FFT which converts time domain signal to frequency domain signal, the results are obtained by suitable modification of the Fourier transform coefficients and take complex conjugate to eliminate imaginary value. Secondly, we use this case to filter noisy data by finding which peaks are above the noise floor and remove everything below that floor, and then IFFT to de noise the signal. Here, we filter and take all the values that have F magnitude squared less than 1.5, than take the power spectrum multiplied by indices vector which zero out all the indices that have small Fourier coefficient and keep only those that have power greater than 1.5. The signal is then filtered by inverse Fourier transform. Then, Hilbert transform is used to extract complex components from a signal. HT generate the envelope of the time signal by calculating the magnitude of the time function. Calculate the peaks and Save the real values of peaks to group the peaks into the same window. Then create an 1D CNN network model and feed the input into CNN model with supervised learning technique. The training process using CNN model requires the CNN model to adjust weigh according to training and validated data. Lastly, the trained CNN model is used for peak detection.

A. *Preprocessing*

Preprocessing is a method to pre-process on data. All input variables are normalized, de correlated and then transformed to a neural network. Preprocessing composes of

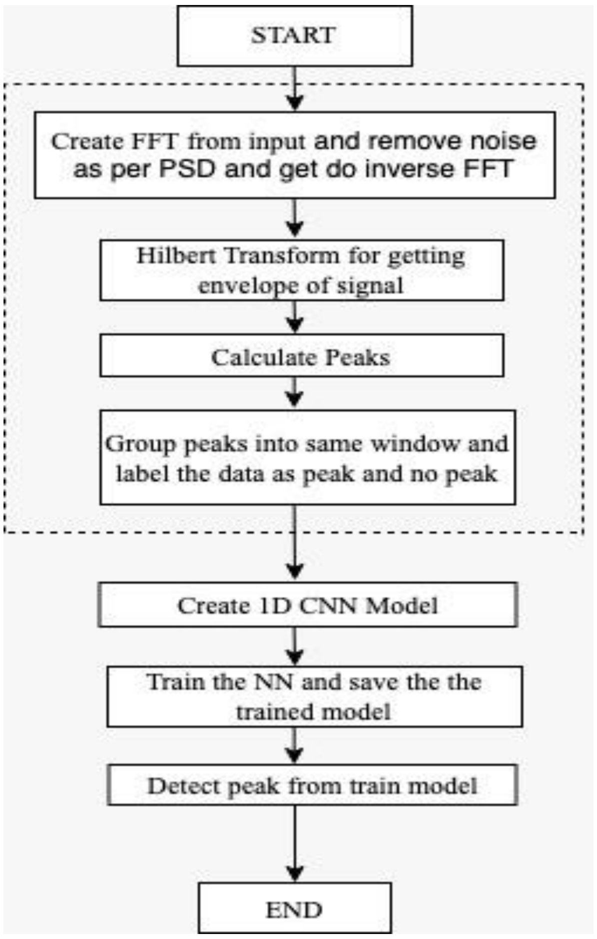

START
Create FFT from input and remove noise as per PSD and get do inverse FFT
Hilbert Transform for getting envelope of signal
Calculate Peaks
Group peaks into same window and label the data as peak and no peak
Create 1D CNN Model
Train the NN and save the the trained model
Detect peak from train model
END

Figure 1: Flowchart Diagram of the Experiment

B. *First Fourier Transform:*

Preprocessing performs FFT on time signal data. Fourier transform is one of the most widely used techniques in applied mathematics and engineering from filter design to signal processing. In brief, Fourier transform is based on where functions of suitable properties can be represented with a linear combinations of trigonometric functions. This Trigonometric Functions can be seen as extracting frequency-domain information from a time-domain signal, used for alternative or more efficient ways of investigating or modifying a signal [4]. 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. These applications include data compression, handling noisy signals, analysis of crystal

Structures, and solving partial differential equations [5]. The DFT equation $\boldsymbol{X(k)} = \sum \boldsymbol{x(n) W_N^{nk}}$ is decomposed into several short transforms and recombined in the FFT formula. In FFT the input signal is converted into frequency domain using DFT and multiplied by the frequency response of the filter, and then converted back into time domain using inverse DFT. Fourier transform (FFT) and inverse fast Fourier transform (IFFT) eases the computational complexity of the DFT from O(N2) to O (N log N). Here, given a sinusoidal signal which is in time domain, when applied Fourier transform it provides the constituent signal frequencies. Fourier transforms are used for both periodic and non-periodic signals where they can be transformed from time domain to frequency domain.

C. *Hilbert Transform*

Hilbert Transform has played a vital role in signal domain, and it has many applications in different fields. HT provides a ±90° Phase shift to the input signal, therefore if the input signal chooses to be a sine function the output will give cosine function. In Fourier transformation one can change the time domain to frequency domain, but in Hilbert Transform the operation remains the same in its time or frequency domain [6]. HT is an alternative method to extract complex components from a signal. The Hilbert Transform characterized by,

$$h(t)= \frac{1}{\pi t}$$

$$\xLeftrightarrow{\text{Fourier transform}} H(\omega) = -jsin(\omega) = \begin{cases} -j, \ 0 < \omega < \pi \\ j, \ -\pi < \omega < 0 \end{cases}$$

HT rotates the Fourier components in complex space by the formula dt. 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” [7]. In MATLAB’s implementation of the Hilbert () function it takes advantage of the fast Fourier transform (FFT).

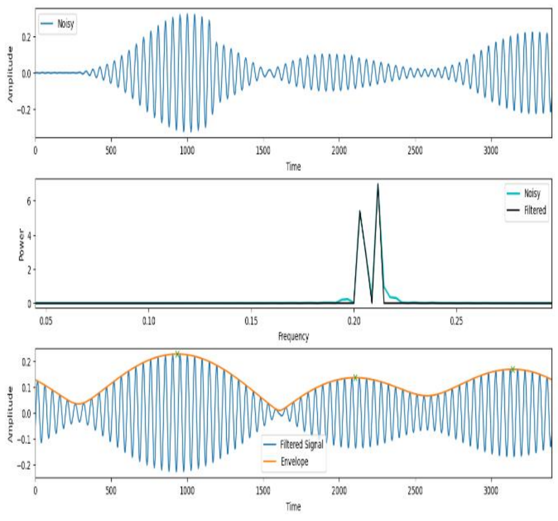


Figure 3: Upper enveloping by Hilbert Transform

The Hilbert () function does the calculation in three steps: First do the FFT of Xr. Then set the elements in FFT which correspond to frequency −π < ω <0 to zero. Finally do the inverse FFT. A

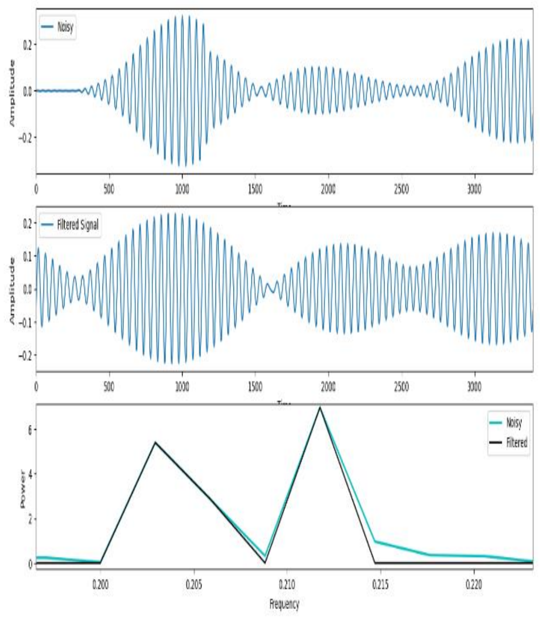


Figure 2: Filtering signal using FFT

Hilbert transformer is useful for envelope extraction shows in figure 3. This can be understood by the formula that Hilbert transform produces a sin(ωt) for every cos(ωt). HT can generate the envelope of the time signal by calculating the magnitude of the time function. Envelope analysis is the slow overall change and sound intensity over time. Hilbert transform

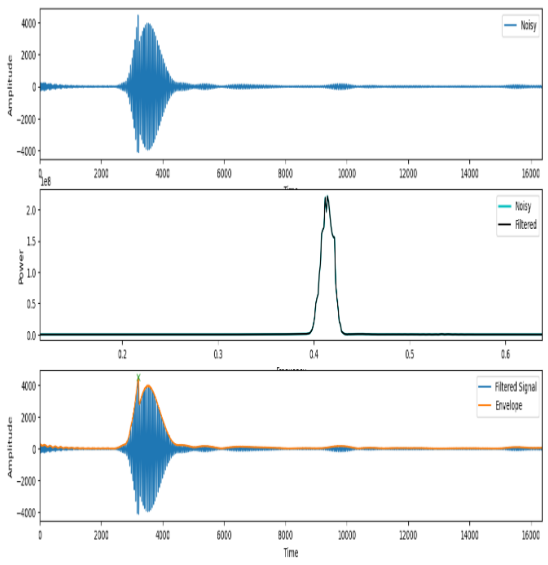


Figure 4: Performing FFT and Hilbert Transform

can be used to find an imaginary function to a real valued signal, as the real signal can be analytically extended from the real axis to the upper half of the complex plane. (Figure 4) shows the final result.

D. *Training and Testing*

In the preprocessing stage, FFT is used for noise removal. Then, 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. We train the neural network with the peak from preprocessing. The CNN model then detects the peak from trained model. There are numerous available conventional CNN models. We chose one of the models listed on the Keras website and made a small adjustment to it to meet the issue. A 1D CNN has generally raw data as input instead of handcrafted features. Input data is processed through different trainable convolutional layers for learning an appropriate representation of the input data. Accordingly, local connectivity theorem, the neurons in a layer are connected only to a small region of the previous layer which is called a receptive field. The input of 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,

$$T = F(X \mid \Theta) = fL(\ldots f2(f1(X \mid \Theta1) \mid \Theta2) \mid \Theta L) \ldots.(1)$$

Where L is the hidden layers number 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 $\otimes$ 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 | Θl) = h(W Xl + b), Θl = [W, b]……….(3)

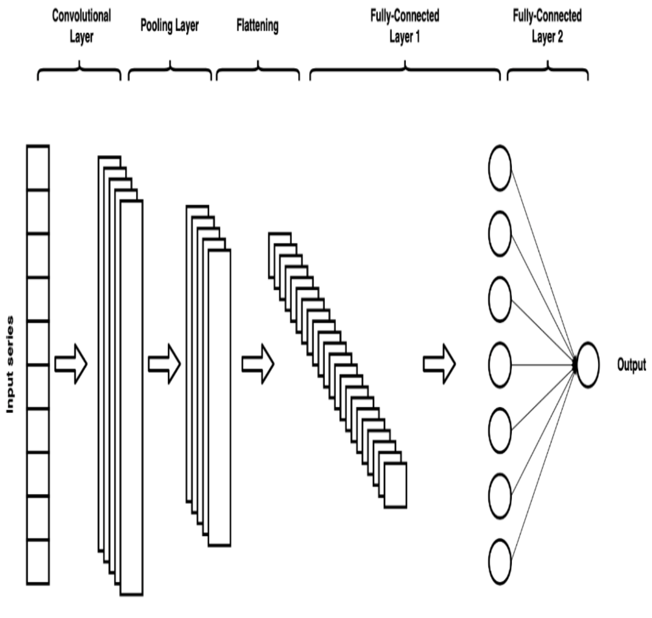


Figure 5: Simplified schema of 1D CNN

The 1D CNN can aggregate the local features and cut down the dimensions of data through convolutional learning and spatial pooling operations.

Here, the convolutional layers read the input of 1D time series and drag a kernel over the time series. The kernel defines the features we want to locate in the series and it has the same width as the time series, so it can only move in one direction. For each step, the input is multiplied by the values of the kernel. Then a non-linear activation function is applied to the result. This way, the original input series is transformed into an interpretation of the input called the filter map. The next step is to apply a pooling layer, which reduces the size of the series, but-at the same time-preserves the identified characteristics. Convolutional and pooling layers can be stacked on top of each other to provide multiple layers. The results of the pooling layer can be passed to a fully connected layer.

First 1D CNN layer: Specifying a filter in the top layer (or also called feature detector also called kernel size). In order for the neural network to learn a single feature in the first layer, only one filter would need to be defined. The output matrix contains the weights for a single filter in each column.

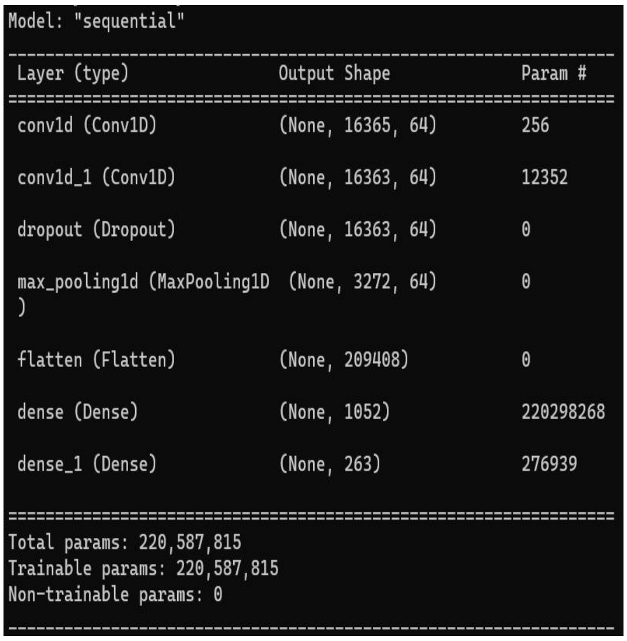
Second 1D CNN layer: The second CNN layer will be fed the output of the first CNN. Using the same reasoning as the top layer

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

Third and fourth 1D CNN layer: In order to learn higher level features, a further set of 1D CNN layers is used.

Average pooling layer: A further pooling layer averaged out will help prevent overfitting. This time, the average value of two weights within the neural network is used rather than the maximum value. In the neural network on this layer, there is only one weight left for each feature detector.

Dropout layer: The neurons in the network will get 0 weights at random in the dropout layer. Given that we selected a rate of 0.5, 50% of the neurons will have their weight set to 0. By



```
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

performing this function, the network's sensitivity to slight fluctuations in the data is reduced. As a result, it ought to further improve our data accuracy.

Fully connected layer: The height vector will be decreased in the last layer. Another matrix multiplication is used to perform this reduction.

E. *Evaluation*

After training the CNN model, we assessed its performance by comparing predictions to real data using the confusion matrix. There are four values in the confusion matrix. True Positive (TP) is a measure of how well projections of positive outcomes compare to actual positive cases. The value of inaccurately making positive forecasts is known as a false positive (FP). True Negative (TN) measures the value of accurate negative predictions made from actual negative cases. False Negative (FN) is the value assigned to inaccurately foretelling negative outcomes.

Accuracy: Accuracy measures how well the CNN model performs when making predictions. It is determined as the ratio of the accurate forecast to the total prediction. The formula from measures the accuracy.

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

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.

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

III. Experiment Result

We have randomly chosen 2 samples from the given datasets and found the results into (Figure 7&8). We found the accuracy is around 30-40%.

Sample 1: First input sample can be found into GitHub repository at

`\static\files\sample_input_001.xlsx`

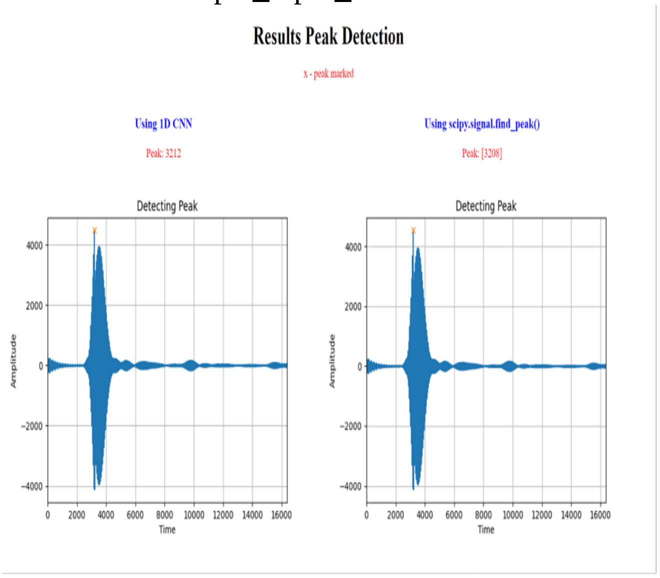


Figure 7: Result from the sample

Sample 2: Second input sample can be found into GitHub repository at

`\static\files\sample_input_002.xlsx`

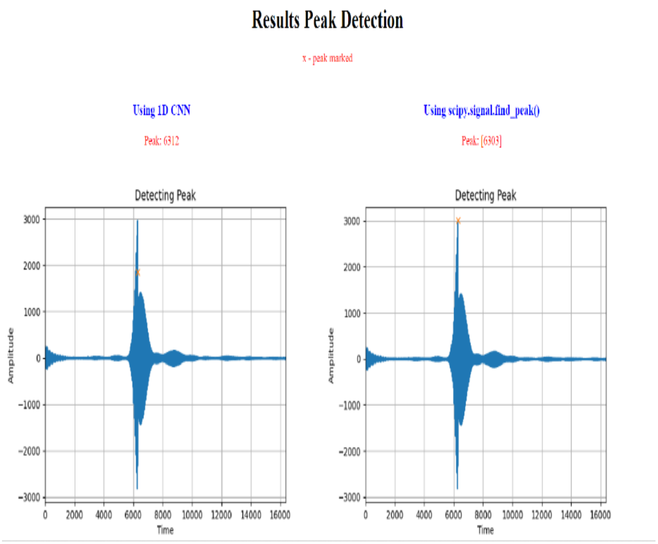


Figure 8: Result from the sample

IV. Encountered Problems

Although the evaluation of training experiment shows the promising result, as we conduct the experiment for 100 times, we found the low evaluation result in 1 of 100 times. The cause is still unknown, but we make an assumption that more information on signal is required to process noise free signal. We have faced another issue that dataset was changed in dimensions at last stage of 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 need approximation to match actual results that doesn't give high actual peak. But a window in which peak is present. To improve accuracy need to make small window which matches kernel size of CNN. Current window size is 62 which cannot be comment as good or bad since the sampling rate is unknown.

References

[1] Hoshen, Y., Weiss, R. J., & Wilson, K. W. (2015). Speech acoustic modeling from raw multichannel waveforms. In IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 4624–4628)

[2] LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. Nature 2015, 521, 436–444

[3] Zhu, Z., Enge, J. H., & Hannun, A. (2016). Learning multiscale features directly from waveforms. Proceedings of the Annual Conference of the International Speech Communication Association (INTERSPEECH), (pp. 1305–1309).

[4] Bruce, A., Donoho, D., & Gao, H. Y. (1996). Wavelet analysis [for signal processing]. *IEEE spectrum*, *33*(10), 26-35.

[5] Osgood, B. G. (2019). *Lectures on the Fourier transform and its applications* (Vol. 33). American Mathematical Soc.

[6] Aditi Singh, Survey Paper on Hilbert Transform with Its Applications in Signal Processing, (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5 (3), 2014, 3880-3882.

[7] King, F. W. (2009b). Hilbert Transforms, Vol. 2, Cambridge University Press, Cambridge, UK.