information

Article

Instantaneous Frequency Estimation of FM Signals under Gaussian and Symmetric α-Stable Noise: Deep Learning versus Time–Frequency Analysis

Huda Saleem Razzaq [1] and Zahir M. Hussain [1,2,*]

[1] Computer Science and Mathematics, University of Kufa, Najaf 54001, Iraq
[2] School of Engineering, Edith Cowan University, Joondalup, WA 6027, Australia
* Correspondence: z.hussain@ecu.edu.au or zmhussain@ieee.org

Abstract: Deep learning (DL) and machine learning (ML) are widely used in many fields but rarely used in the frequency estimation (FE) and slope estimation (SE) of signals. Frequency and slope estimation for frequency-modulated (FM) and single-tone sinusoidal signals are essential in various applications, such as wireless communications, sound navigation and ranging (SONAR), and radio detection and ranging (RADAR) measurements. This work proposed a novel frequency estimation technique for instantaneous linear FM (LFM) sinusoidal wave using deep learning. Deep neural networks (DNN) and convolutional neural networks (CNN) are classes of artificial neural networks (ANNs) used for the frequency and slope estimation for LFM signals under additive white Gaussian noise (AWGN) and additive symmetric alpha stable noise (SαSN). DNN is composed of input, output, and two hidden layers, where several nodes in the first and second hidden layers are 25 and 8, respectively. CNN is the content input layer; many hidden layers include convolution, batch normalization, ReLU, max pooling, fully connected, and dropout. The output layer consists of a fully connected softmax and classification layers. SαS distributions are impulsive noise disturbances found in many communication environments such as marine systems, their distribution lacks a closed-form probability density function (PDF), except for specific cases, and infinite second-order statistics, hence geometric SNR (GSNR) is used in this work to determine the effect of noise in a mixture of Gaussian and SαS noise processes. DNN is a machine learning classifier with few layers for reducing FE and SE complexity. CNN is a deep learning classifier, designed with many layers, and proved to be more accurate than DNN when dealing with big data and finding optimal features. Simulation results show that SαS noise can be much more harmful to the FE and SE of FM signals than Gaussian noise. DL and ML can significantly reduce FE complexity, memory cost, and power consumption as compared to the classical FE based on time–frequency analysis, which are important requirements for many systems, such as some Internet of Things (IoT) sensor applications. After training CNN for frequency and slope estimation of LFM signals, the performance of CNN (in terms of accuracy) can give good results at very low signal-to-noise ratios where time–frequency distribution (TFD) fails, giving more than 20 dB difference in the GSNR working range as compared to the classical spectrogram-based estimation, and over 15 dB difference with Viterbi-based estimate.

Keywords: frequency estimation; LFM; sensors; IoT; software-defined radio (SDR); alpha-stable noise; TFD; deep learning

check for updates

Citation: Razzaq, H.S.; Hussain, Z.M. Instantaneous Frequency Estimation of FM Signals under Gaussian and Symmetric α-Stable Noise: Deep Learning versus Time–Frequency Analysis. *Information* **2023**, *14*, 18. https://doi.org/10.3390/info14010018

Academic Editor: Riccardo Bernardini

Received: 28 November 2022
Revised: 22 December 2022
Accepted: 23 December 2022
Published: 28 December 2022

1. Introduction

Frequency estimation is utilized in various engineering applications, including communications, RADAR, frequency identification of sinusoidal signals, and resonance sensing systems. Many signals in practice are nonstationary, such as FM, which is a signal found in communication and other application. Those signals can be classified as either monocomponent or multicomponent signals. The estimation of the instantaneous frequency (IF)

is a natural evolution from the measure of the steady-state sinusoidal frequency, which has been intensively studied for many years [1,2].

Ref. [3] proposes an algorithm for frequency estimation of sinusoidal FM signals using interpolation of the fast Fourier transform (FFT) and discrete-time Fourier transform (DTFT), relying on N-point FFT to find the position of the maximum FFT. Three spectral lines located within the main lobe are used to estimate the frequency. A non-zero-padded sinusoidal signal frequency estimation technique was developed using FFT and DTFT. The model of a single-frequency signal with additive white Gaussian noise is used to estimate the frequency based on the maximum spectrum line of the frequency spectrum indexed by the FFT. The root mean square error (RMSE) of a suggested algorithm is much lower than that of Candan, a rational combination of three spectrum lines (RCTSL), and Aboutanios and Mulgrew (A&M) algorithms and close to the Hybrid A&M and q-shift estimator (HAQSE) algorithm. When the SNR is between 5 dB and 100 dB, the suggested approach outperforms the competitor algorithms [3].

Ref. [4] studies the instantaneous frequency estimation of multicomponent signals within the time–frequency domain, where a combination of Eigen decomposition of time–frequency distributions and time–frequency filtering is used to extract signal components and estimate their instantaneous frequencies using the ridge detection and tracking procedure, where time frequency (TF) signal analysis, signal decomposition, and IF estimation methods are based on quadratic forms to estimate the IF. The Wigner-Ville distribution is a quadratic class prototype distribution. Eigen decomposition of TFDs is used to extract components from a quadratic TFD. The ridge tracking approach is then used to estimate instantaneous frequency. The estimation approach may occasionally fail to evaluate the IFs of elements that overlap in the TF domain. This method extracts IFs based on amplitudes, neglecting directional variations in ridge curves. In other words, this approach only considers the patterns of the remaining components after detecting the target ridge component. As a result, the estimating process may take the incorrect course [4].

1.1. State-of-the-Art Methods

Ref. [5] offers a method that relies on neural networks for F0 estimate. It includes two sub-tasks as a classification to determine whether or not the frame has the voice and regression for estimating the (F0) value. A single model is used for both, and the output is (F0) values for voice frames and zeroes for unvoiced frames. F0 estimation consists of two sub-tasks: a classification task determining whether the structure contains a voice or not and a regression task that estimates the F0 value. A primary solution is accomplished by utilizing a single model for both, with reference F0 values for voice frames and 0 for "unvoice" frames (frames not presenting a valid voice F0). This strategy, however, may be performance-limited because varying target values rather than probabilities are not appropriate for classification and may result in an unstable boundary between voice and unvoice. The zeros for unvoice denote an infinitely large F0 period, which should not be included in the training data for numerical regression. A numerical regression is used to formalize the F0 estimation. By changing the output layer to rectified linear unit (ReLU) activation, the networks for voice detection should be suitable as an F0 estimator [5].

Ref. [6] proposes estimating the Doppler frequency using an artificial neural network (ANN). The results explain that this method has a better performance and lower computational cost compared to traditional methods such as Robust Chinese Remainder Theorem (RCRT). They used an ANN with three neurons in the input layer (remainders by RCRT), ten neurons in a hidden layer, and 17 neurons in the output layer. It is randomly divided into three parts: 60% is used for training, 10% is used for validation, and 30% is used for the test. A feed-forward network comprises layers of neurons, and an input layer is used to introduce data into the system. Following that, processing occurs in one or more intermediate (hidden) layers.

The network's final layer produces output data. The ANN learning algorithms aim to modify the weights on all the edges. The weighted inputs from the previous layer

are then pooled within each neuron and pass via an activation function, which generally constrains its output to [0,1]. Although different functions are feasible, sigmoid functions are commonly utilized. A feed-forward network is made up of layers of neurons. An input layer is used to introduce data into the system. Following that, processing occurs in one or more intermediate (hidden) layers. The network's final layer produces output data. The ANN learning algorithms aim to modify the weights on all the edges. The weighted inputs from the previous layer are then pooled within each neuron and pass via an activation function, which generally constrains its output to [0,1]. Although different functions are feasible, sigmoid functions are commonly utilized [6].

Ref. [7] introduces a model that relies on a convolutional neural network to signal frequency signal and LFM signal detection and estimation. The pre-trained model is based on signals with a 2-dimensional domain containing multiple convolutional layers, pooled layers, and fully connected layers. Finally, softmax classification is used as the output layer. The RADAR echo is first demodulated, and the pulse is compressed. The preprocessed LFM signal is then sampled, yielding a series of one-dimensional sequences. The one-dimensional line is split into equal-length segments at regular intervals, splicing from top to bottom in chronological order to generate a one-dimensional matrix that serves as samples for the training and test sets. At the same time, fractional Fourier transform (FRFT) calculates all of the LFM signal sequences to acquire genuine starting frequency and frequency modulation (chirp rate) information. The training set is then categorized using CNN based on the obtained frequency and chirp rate. Finally, the trained model is put through its paces with the test. The dataset is used for calculating the parameters of the LFM signal. There are three types of data utilized in model training and testing. The first type is a single-frequency signal, with uniform motion as the matching target motion state. The second type is the chirp rate signal, and the target motion is acceleration with an initial velocity of 0 m/s (acceleration 1). The third type is the LFM signal, and the associated target motion is acceleration with a non-zero beginning velocity (acceleration 2). AlexNet is used for training and testing. The simulation findings show that when the SNR is large, the recognition rate of all three types of signals is greater than 90% [7].

In various applications, such as wireless communications and image processing, SαS noise is widely encountered. Ref. [8] analyzes the characteristics of α-stable noise, and the chirp signal in α-stable noise is converted into Gaussian-like distribution. Then, fractional Fourier transform was used to estimate the initial frequency and chirp rate of signal in α-stable noise. The FRFT approach produces good parameter estimation results for chirp signals in Gaussian noise. However, when the signal is polluted with α-stable noise, the performance of FRFT suffers. As a result, based on the pulse characteristics of α-stable noise, FRFT can remove the sharp spikes of the echo signal and convert the non-Gaussian noise into a Gaussian-like distribution, and then uses energy concentration of FRFT to gain accurate initial frequency and chirp rate estimates of the chirp signal. The simulation results show that the approach has high anti-noise performance when predicting chirp signal parameters in α-stable noise, and the estimated effects are consistent with noise-free signals [8].

In [9], Aboutanios and Mulgrew (A&M) suggested two similar numerical methods that outperform all existing DFT-interpolation-based methods, with asymptotic variances that are only 1.0147 times the Cramer-Rao Lower Bound (CRLB). The A&M approach employs the signal's DFT coefficients shifted by ∓ 0.5 from the peak DFT coefficient, which can be thought of as interpolating the signal's DFT twice. Moreover, they demonstrated using the fixed-point theorem that the iterative A&M algorithm converges in only two iterations and that adding some other iteration does not enhance estimation performance [9].

Ref. [10] presents two approaches for estimating the frequency of a complex sinusoidal in noisy conditions. The first approach interpolates on the signal's Q-Shifted Estimator (QSE) DFT coefficients, and the optimal iterative process number is found to be a logarithmic role of the signal size. Hybrid half-shifted and q-shifted (HAQSE) DFT interpolators are used in the second approach, which converges in only two iterations. Both estimators are

shown to be asymptotically unbiased, with their mean squared errors performing near the Cramer-Rao lower bound. The algorithm's effectiveness is uniform over [0.5,0.5], and the proposed estimator is unbiased. The number of optimized iterations is chosen, and performing additional iterations does not enhance the total asymptotic performance; then, the value of the optimum DFT shift is determined. The frequency estimate has been asymptotically standard, with mean frequency and variance near the Cramer-Rao lower bound (CRLB). Estimators QSE and HAQSE are efficient methods because they are minimum-variance, unbiased estimators. HAQSE needs two iterations to converge, whereas QSE may need more than two. It was also demonstrated that HAQSE performs better with shorter signal lengths. QSE requires one more iterative process for most practical signal lengths [10].

Ref. [11] suggests that the parameters of the LFM signal under an α-stable noise environment can be estimated using the fractional Fourier transform and the Sigmoid transform. Two new functions are defined: the sigmoid fractional correlation function and the sigmoid fractional spectral density (Sigmoid-FPSD). A novel method for estimating LFM signal parameters based on Sigmoid-FPSD under alpha-characterized noise is proposed based on these two definitions. Furthermore, the boundedness of the Sigmoid-FPSD under SαS noise and the feasibility analysis of the Sigmoid-FPSD are described to evaluate the proposed method's performance. Both theoretical studies and simulations show that the proposed approach outperforms other existing methods [11].

1.2. Related Works

In a parallel direction of IF estimation for single-tones, Almayyali and Hussain reached promising results for using deep neural networks [12], where complexity has been reduced compared with classical techniques, making the DL approach suitable for SDR networks, sensors, and IoT applications. Generally, it is difficult to estimate the parameters of FM signal under a mixture of α-stable noise (non-Gaussian noise) and Gaussian noise. Moreover, the extensive dataset for noisy LFM signals is used for frequency and slope estimation. Deep learning significantly contributes to estimating parameters under the influence of these conditions. The convolutional network extracted features from these signals and then classified them for frequency and slope prediction. It is then compared with the classical method TFD.

The rest of this paper is outlined as follows: Section 2 introduces the problem. Section 3 presents objectives and contributions; Section 4, FM and noise. Section 5 introduces the proposed method. Section 6 introduces IF estimation based on TFD. Section 7 discusses the results; Section 8, further remarks; and Section 9 presents the paper's conclusion.

2. Problem Definition

Two fundamental issues in signal processing are a signal estimation and the separation of nonstationary signals. The parameter estimation includes the measure of the IF and linear chirp rate (LCR) or slopes for LFM signals under noise. Usually, Gaussian noise is considered. However, in underwater communications and many other environments, impulsive noise is the real problem. An essential kind of impulse noise is the α-stable noise, which is impulsive in the time domain and highly affects the accuracy of estimating the parameters of noisy LFM signals. Impulse noise is typically associated with Gaussian noise, making the estimation problem more difficult. FM signals are used in various engineering applications, including RADAR, SONAR, and communications. The frequency content of such signals contains the intended information. This work considers the recovery of information carried by linear FM signals contaminated by a mixture of α -stable noise and Gaussian noise.

3. Objectives and Contributions

The frequency estimation problem has been processed by classical techniques such as Fourier and correlative techniques. Moreover, the same problem is currently processed by deep neural networks and CNN. This work aims to:

- Provide an accurate and fast estimation of IF and instantaneous slope using deep learning, as deep learning for frequency classification is promising. The proposed method can be used for RADAR and medical SONAR applications, where RADAR functions include range (localization), angle, and velocity, while medical SONAR functions include diagnosis, classification, and tracking. The use of the proposed approach can lead to improved RADAR localization and improved medical SONAR diagnosis.
- Create a dataset of noisy LFM signals with varying LCR and frequency.
- Two types of noise are combined by a linear equation, as explained in Section 5.1.
- Convolutional deep learning, rather than recursive networks, estimate parameters. Researchers commonly use convolutional networks to classify signal types.
- A comparison between the classical and convolutional deep learning methods.
- They obtain high accuracy in the presence of impulsive noise without the use of de-noise methods.

The contributions of this work involve the following:

- The performance of the DL approach is compared with the version of the still-active classical techniques based on Fourier analysis. It is shown that the classical time–frequency-based methods are ineffective under the damaging alpha-stable noise, especially under low signal-to-noise ratios, where a difference of 20 dB in performance has been noticed compared to the DL approach. This result is vital for underwater RADAR systems, where impulsive noise is dominant.
- The DL approach is SNR-dependent, so an investigation of the system performance under various SNRs is presented. Based on our previous work [12], a change in SNR will have little effect on the performance of the DL-based approach.
- The reduced complexity introduced by DL-based FE and avoiding complex valued arithmetic will make FE easier and cheaper for IoT communications, sensors, sensor networks, and SDR. This work presents discussions on such possibilities.

4. FM Signals and Noise

This section illustrates FM signals, AWGN, and SαS noise as follows:

4.1. Instantaneous Frequency and FM

A key feature of FM transmissions is the instantaneous frequency, which describes the fluctuations in frequency content across time. The IF of a signal is a time derivative of its instantaneous phase $\theta(t)$ [13,14]:

$$f(t) = \frac{1}{2\pi}\frac{d\theta(t)}{dt} \tag{1}$$

$$\theta(t) = 2\pi(f_o t + \sigma\frac{t^2}{2} + \rho\frac{t^3}{3}) \tag{2}$$

Note that the initial phase θ_o has been omitted as it has no effect on the frequency estimation.

The signal model with LFM law used in this work is [15]:

$$s(t) = A\, e^{j2\pi(f_o t + \frac{\sigma}{2}t^2)} \tag{3}$$

where σ is the linear modulation index, f_o is the initial frequency (in Hertz), and A is the amplitude. Using Equation (1), the LFM signal IF will be [16]:

$$f(t) = f_o + \sigma t \tag{4}$$

The quadratic IF law has also been used to consider the quadratic frequency modulation (QFM) signal in this work:

$$s(t) = A\, e^{j2\pi(f_o t + \frac{\sigma}{2}t^2 + \frac{\rho}{3}t^3)} \tag{5}$$

where ρ is the quadratic modulation index of the QFM signal, with the quadratic IF law:

$$f(t) = f_o + \sigma t + \rho t^2 \tag{6}$$

4.2. Additive White Gaussian Noise

AWGN has the following probability density function with zero mean and variance (power) v^2 [13]:

$$\mathrm{F}_Z = \frac{1}{v\sqrt{2\pi}} e^{-Z^2/2v^2} \tag{7}$$

where Z is a random variable and v is the standard deviation of the noise.

The procedure for generating AWGN is as follows:

1. Calculating the power p_x contained in the input signal $x(t)$, were

$$p_x = \frac{1}{L}\sum_{i=0}^{L-1} |x[i]|^2,\ L = |x| \tag{8}$$

2. Converting the supplied SNRdB (SNR in dB) to a linear scale and finding the noise power in terms of SNR and signal power p_x, were

$$\mathrm{SNR} = 10^{\mathrm{SNRdB}/10},\ N_0 = p_x/\mathrm{SNR} \tag{9}$$

3. Using the following equations to determine the AWG noise:

$$G = v{\cdot}Z. \text{ if } x \text{ is real} \tag{10a}$$

$$G = v(Z + i\,M),\ \text{if } x \text{ is complex} \tag{10b}$$

where $Z, M \in \mathcal{N}(0, v^2)$. For a real signal $v = \sqrt{N_0}$, for a complex signal $v = \sqrt{N_0/2}$.

4.3. Symmetric α-Stable Noise

The α-stable distribution noise necessitates four parameters (α, γ, β, and μ), with the stable distribution characteristic function specified as [17,18]:

$$\psi(\omega) = \exp(-\gamma|\omega|^\alpha\ [1 + \beta \mathrm{sign}(\omega) W(\omega, \alpha)) \tag{11a}$$

$$W(\omega, \alpha) = \left\{ \begin{array}{l} \tan\left(\frac{\alpha\pi}{2}\right) \text{ for } \alpha \neq 1 \\ \frac{2}{\pi}\log|\omega| \ \text{ for } \alpha = 1 \end{array} \right\} \tag{11b}$$

And sign(ω) is the signum function.

The characteristic function for the SαS distribution when β = 0 is specified as follows:

$$\psi(\omega) = \exp\left(-\gamma|\omega|^\alpha\right) \tag{11c}$$

where ($0 < \alpha \leq 2$) is also known as the tail index or characteristic exponent. When $\alpha < 2$, the distribution is algebraic-tailed with a constant tail α, meaning infinite variance. The density of tails becomes heavier as it gets smaller. When $\alpha = 2$, the SαS distribution is reduced to the Gaussian distribution. When $\alpha = 1$ and $\beta = 0$, the SαS distribution is reduced to the Cauchy distribution. When $\alpha = 0.5$ and $\beta = 1$, the SαS distribution is reduced to the Lévy distribution. The parameter $\gamma > 0$, usually called the dispersion, is a positive constant related to the distribution scale. The parameter γ plays a role that is analogous to that of the variance for a second-order process. The skewness parameter is $\beta \in [-1,1]$. The location parameter is $\mu \in \mathbb{R}$. The procedure of SαS simulation is explained in Appendix A.

5. The Proposed Method

This section explains the proposed method for estimating the frequency and slope. Deep learning and machine learning were used to predict the frequency and slope of noisy signals and then calculate the instantaneous frequency. Then it is compared with the

time–frequency distribution as shown in the results. ANN includes feedforward neural network (FNN), the same multilayer perceptron (MLP). The input–output layer is called a single-layer network, and one hidden layer is called a shallow neural network. Two or more hidden layers are called DNN. The nodes in neighboring layers are fully connected. DNN with a complex structure is time-consuming for training [19]. The activation functions are chosen depending on the type of problem to be solved by the network. The most common activation functions are sigmoid or logistic and hyperbolic tangent or tanh [20]. SCG and adaptive moment estimation (Adam) are used as optimization algorithms in ANN. SCG uses second-order neural network information while requiring just $\mathcal{O}(N)$ memory use, where N is the number of weights in the network [21]. The procedures of Adam are explained in reference [22]. DL is a subset of machine learning. CNN is used in deep learning. A CNN type of deep ANN [23], consists of input, output, and hidden layers. CNN works with multiple hidden layers and 2D data, so the input data must be transformed into 2D matrices before it can detect frequency or slope. Each input image is passed through a series of convolution layers with filters (kernels), pooling, fully connected layers, and the Softmax function to train and test deep learning CNN models. CNN transforms manual feature extraction methods into automated processes [24,25]. Many metrics are used to evaluate ML and DL methods. The perfect models chosen using these metrics [26] include accuracy, precision, recall, F-measure, and receiver operating characteristic (ROC).

This paper demonstrates for the first time the estimation of FM parameters by deep learning, which is one of the main contributions, as most researchers use deep learning to classify the signals, where the proposed CNN model has less complexity than found models. We have not used pre-trained CNN models such as AlexNet, VGG, GoogLeNet, or ResNet, as they have more layers and complexity. The signal was used in the time domain because conversions to the frequency domain are complex and affect the efficiency of deep learning. We also propose the b ratio to combine the Gaussian noise and the symmetric α-stable noise to form a noisy signal.

5.1. Hybrid Noise and Noisy Signal Generation

A nonstationary signal is one with a changing frequency content over time. This work is based on LFM signals influenced by noise (AWGN and SαSN). SαSN requires four parameters $(\alpha,\ \gamma,\ \beta,\ \mu)$. The most critical parameters are the tail index (α) and scale of the distribution $(\gamma > 0)$, while the less essential parameters are β and μ. Gaussian noise is fixed power, and SαS noise is geometric power.

Geometric SNR (GSNR) is used to determine noise impulsiveness, characterized by zero-order statistics. Since all 2nd order moments are infinite, the standard SNR does not apply. The geometric power of $S\alpha SN$ is defined as follows:

$$p_S = \gamma^2.\, C^{\left(\frac{2}{\alpha}-1\right)} \tag{12}$$

where C is the exponential of Euler's constant, $C = e^{E_c} \approx 1.7811$, E_c Is Euler's constant $(E_c = 0.5772156649)$. When $\alpha = 2$, SαS noise is Gaussian with finite variance $\sigma^2 = \gamma^2$.

$$\text{GSNR} = p_x/\gamma^2 \tag{13}$$

$$\text{GSNRdB} = 10 \cdot \log_{10}(\text{GSNR}) \tag{14}$$

$$p_x = A^2/2 \tag{15}$$

In wireless networks, received signals are corrupted by the noise mixture of Gaussian (G) and SαS (Y) noises. Total noise (N_T) is represented in the equation as follows:

$$N_T = G + Y \tag{16}$$

The overall GSNR is defined as follows,

$$GSNR = p_x / p_T \tag{17}$$

Let $p_T = p_G + p_S$, where p_T total noise power, and p_G be the Gaussian power. We proposed b ratio, if $p_G = b \cdot p_S$, then $p_T = (1+b)p_S$, $p_S = \frac{p_T}{1+b}$, and

$$b = p_G / p_S \tag{18}$$

If b is less than one, then p_G is less than p_S, else p_G is greater than p_S. The scale parameter is:

$$\gamma = \sqrt{p_S / C^{(\frac{2}{\alpha}-1)}} \tag{19}$$

Consider AWGN and SαSN affected by single-tone sinusoidal and FM signals as follows:

$$x(t) = A\cos(\theta(t) + \varnothing_0) + N_T \tag{20}$$

where A is the signal amplitude, and $\varnothing_0$ is an initial phase. The single tone and LFM signals are generated, where the amplitude of signals is $A = 1$, signal power is $p_x = \frac{A^2}{2}$, initial phase $\varphi_o = 0$. To find the instantaneous phase as shown in Equation (2), the frequency and LCR are computed as follows:

- The frequency (f_o) range:

  Initial frequency is $f_1 = 10$
  Final frequency is $f_2 = 19$
  The number of frequencies is $n_f = 10$
  The differential frequency step is $df = (f_2 - f_1)/n_f$
  The range of frequency is $f_o = [\, f_1 \ : \ \text{increasing by } df \ : \ f_2]$

- The LFM slope (σ) range:

  Initial slope is $e_1 = 0.1$
  Final slope is $e_2 = 0.9$
  The number of slopes is $n_e = 10$
  The differential frequency step is $d_e = (e_2 - e_1)/n_e$
  The range of slope is $\sigma = [e_1 \ : \ \text{increasing by } d_e \ : \ e_2]$

- The time vector (t) range:

  Initial time is $t_1 = 0$
  Final time is $t_2 = 640$
  Sampling period is $T_s = 0.1$
  The range of time in seconds is $t = [t_1 \ : \ \text{increasing by } T_s \ : t_2 - T_s]$

The following procedure shows a summary of hybrid noise (N_T) Generation:

- GSNR range is chosen as $[-50\ 50]$dB.
- To generate SαS as shown in Equations (27b), (33) and (35) with four parameters chosen as follows: $\alpha = 1.8$, $\beta = 0$, $\mu = 0$, while the choice of γ is (scale parameter) relies on the ratio $b = 20$ as shown in Equation (19).

  Total power is $p_T = p_x/\text{GSNR}$ as shown in Equation (17).
  $S\alpha S$ geometric power is $p_S = p_T/(1+b)$ and $p_G = b \times p_S$ as explained in the line above Equation (18).
  The AWGN power is $p_{ndB} = 10 \cdot \log 10(p_G)$.
- AWGN (G) is generated as shown in Equation (10).
- Total noise (N_T) is a hybrid: AWGN with SαSN, as shown in Equation (16).

5.2. Converting 1D Signal into 2D Signal and Dataset Creation

After the noisy LFM signals are generated in the time domain with 1-dimensional (1D) signals and length 6400, it is converted into 2-dimensional (2D) with size [80 80] and

in the time domain because CNN deals with 2D signals. An example illustrating how to convert a 1D signal in the time domain into a 2D signal, let input 1D-signal with length 9 is as follows:

$$Input_{sigID} = [1\,2\,3\,4\,5\,6\,7\,8\,9]$$

It converts into 2D signals with a length of 3×3 as follows:

$$Output_{sig2D} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

Data are converted to the graphics format of an image. The images are represented as two-dimensional arrays (matrices) and stored in one of the graphics file formats, such as tagged image file format (TIFF).

After reshaping the input signal into 2D, it saves it as a TIFF type in a folder. The proposed CNN is then trained using 2D signals or images as input. The 1D signals are converted into 2D signals as explained in Algorithm 1.

Algorithm 1: Converting 1D Signals into 2D Signals

Input: The noisy signals 1D in the time domain $(NSignal)$.
Output: The noisy signals with 2D $(Outsig_{2D})$.
Begin:
1. For $h \leftarrow 1$ *to* NS // $NS = 12120$
2. initial counter $k \leftarrow 1$, where $k \in [1\ 6400]$.
3. tack the values from sig_{1D} and place in sig_{2D} column by column and row by row as follows:
4. $sig_{1D} = Nsignal(h)$
5. For $i \leftarrow 1$ *to* R // $R = 80$
6. For $j \leftarrow 1$ *to* C // $C = 80$
7. $sig_{2D}(i,j) \leftarrow sig_{1D}(k)$
8. $k \leftarrow k + 1$
9. End for j
10. End for i
11. Save the sig_{2D} with TIFF type in folder, where $Outsig_{2D} = sig_{2D}$ .TIFF
12. End for h
13. Return $Outsig_{2D}$ in folders. // name folder is label.
End algorithm

The Dataset Generates

The dataset was created after converting signals into 2D, containing ten folders and the name folder is the label, each representing one class. Each class has an IF and an LCR. The number of samples in each class equals the number of GSNR samples 101 multiplied by the number of realizations 12, which equals 1212. The total number of samples in the dataset equals the number of samples in each class multiplied by the number of classes 12120. The target or true label of samples denotes $([0\ 9])$, which represents the number of frequencies and slopes. The dataset is randomly divided into 80% for the training set, 10% for the validation set, and 10% for the testing set, where Algorithm 2 illustrates the dataset divided, where noisy LFM signals dataset denoted $Data\_X$ and labels denoted by $Data_L$.

Algorithm 2: The Dataset is Divided into Training, Validation, and Testing Sets

Input: Noisy LFM signals dataset $(Data\_X)$ and labels $(Data_L)$, where each label is presented as frequency and slope.
Output: Noisy LFM Signals divided into a training set $(XTr,\ LTr)$, validation set $(XVa,\ LVa)$, and testing set $(XTe,\ LTe)$.
Begin:

1. Select a random index as the following:
   $\text{Rind} = \text{Random}(N)$, where $N = |Data\_X|$
2. Sorted dataset according to *Rind* as follows:
   $X = Data\_X(:, \text{Rind}), \quad L = Data\_L(:, \text{Rind})$
3. Select the length of training, validation, and testing percent:
   $Tr = N \times 0.8$
   $Va = N \times 0.1$
   $Te = N \times 0.1$
4. Divided dataset into a training set (XTr), and labels (LTr) as follows:
   $XTr = X(:,\ 1 : Tr)$
   $LTr = L(:,\ 1 : Tr)$
5. Divided dataset into validation set (XVa), and labels (LVa) as follows:
   $XVa = X(:, Tr + 1 : Tr + Va)$
   $LVa = L(:, Tr + 1 : Tr + Va)$
6. Divided dataset into the testing set (XTe), and labels (LTe) as follows:
   $XTe = X(:, Tr + Va + 1 : Tr + Va + Te)$
   $LTe = L(:, Tr + Va + 1 : Tr + Va + Te)$
7. **End of algorithm**

5.3. Estimation of IF and LCR by DNN

A deep neural network (DNN) is used for IF and LCR estimates. It contains the network's components and represents the network's structure. It also needs two phases, forward and backward, for training and predicting the output. Each of them is explained as follows:

DNN structure includes an input layer, two hidden layers, and an output layer. The input layer contains twenty nodes. The number of nodes in the first and second hidden layers is 25 and 8, respectively. The output layer has four nodes (the number of classes).

DNN forward stage, in which the training set is used, finds the sum of the product of input nodes with corresponding weights, then adds bias. The initial weights and biases are generated at random. Then an activation function is applied, where the first and second hidden layers use ReLU for activation. The output layer's activation function is sigmoid. The most important term in the loss expression is an error. There are numerous cost functions. In this work, cross-entropy was used to compute the error for each node in the output layer and the loss value. When the error (the difference between the desired and expected output) is greater than a certain threshold, the neural network stops training, or the neural network stops its access to the last epoch.

DNN backward stage applies an optimization algorithm to update the parameters (weights and biases) and compute the error ratio for each layer in the backward path based on gradient descent, as shown in the optimization algorithm. The optimization algorithm governs how the network's parameters are adjusted. The scale conjugate gradient (SCG) optimization algorithm is used in this work to update weights and biases. The maximum number of epochs for stope training is 1000, and the learning rate is 0.0001. The update is given by:

$$\text{New\_parameters} = \text{Old\_parameters} + R_{\text{SCG}} \times \text{learning\_rate} \tag{21}$$

where New_parameters are new weights and biases, Old_parameters are older weights and biases, and $R_{\text{SCG}} = \alpha_k p_k$, α_k is the k^{th} (adaptive) step-size and p_k is the search direction as explained in reference [12,21], noting that SCG updates the weights and biases relying on the derivative of the loss (entropy). The derivative of the entropy is applied in the

backward stage for DNN as gradient descent, while the loss is applied in the forward stage for DNN to measure the performance of a network.

Evaluation of the training set for each epoch is performed using a validation set. The validation loss is similar to the training loss and is calculated as the sum of the errors for each sample in the validation set. Forward and backward steps are performed until reaching the last epoch (end of training), where the trained DNN model is returned.

The final step is to test the DNN model with a testing set, where a testing data set is applied to the trained DNN model to predict the label. Then the DNN model's evaluation is performed using some metrics such as accuracy.

After the label's prediction, it is possible to estimate the frequency and slope to which it belongs. Then the amount of error and the accuracy between the predicted parameters and the true parameters are calculated as follows:

- True frequency (T_f) is the target frequency of a signal
- Estimate frequency (P_f) is the predicted frequency of the DNN network
- The relative absolute error of the frequency $E_f = \left|T_f - P_f\right| / T_f$
- True slope (T_S) is the target slope of a signal
- Estimate slope (P_S) is the predicted slope of the DNN network
- The relative absolute error of the slope is $E_s = |T_s - P_s| / T_s$
- Estimate IF is $IF = P_f + P_s\, t$

Figures 1–6 explain the several essential cases to show the effect of feature extraction and deep neural network design on the error rate, frequency, and slope estimation accuracy. Figure 7 shows GSNR vs. RMSE of FE for noisy LFM by DNN. Figure 8 shows the ROC of FE for noisy LFM by DNN. Figure 9 shows a confusion matrix for noisy LFM by DNN.

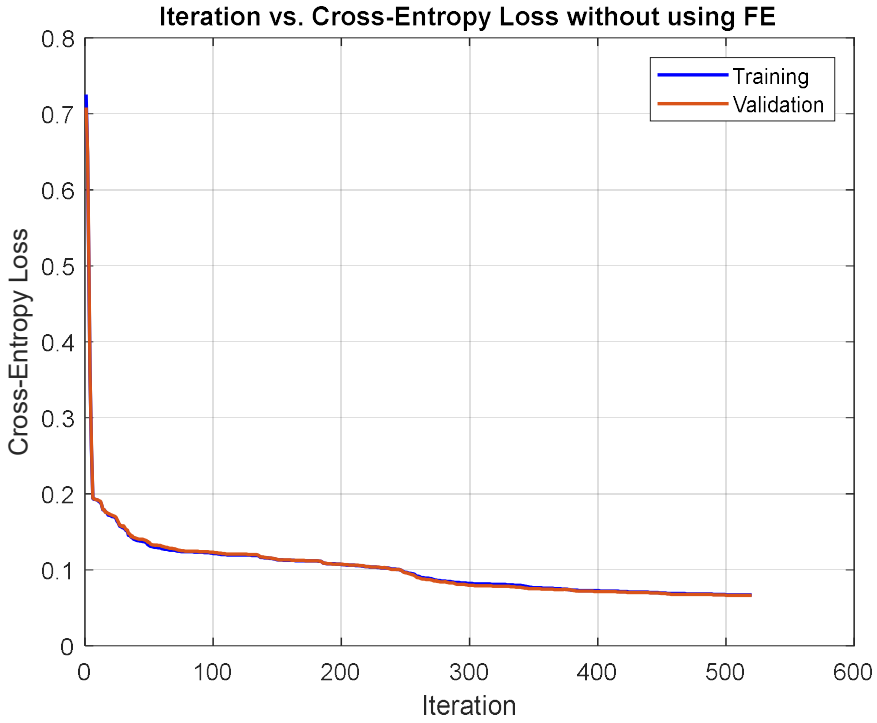

Figure 1. Iteration vs. cross-entropy loss for noisy signals with length 6400 and one hidden layer with ten nodes for DNN. Note that the error is higher than in Figure 32 and has less accuracy at 74 with more than 500 iterations.

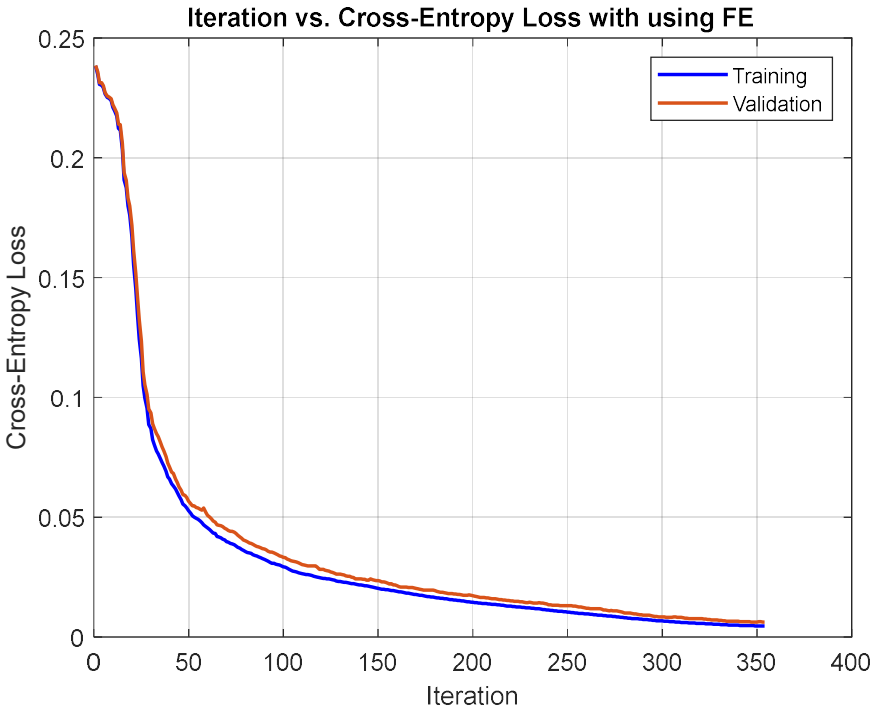

Figure 2. Iteration vs. cross-entropy loss for features extraction of noisy signals with length 6400, one hidden layer with ten nodes for DNN.

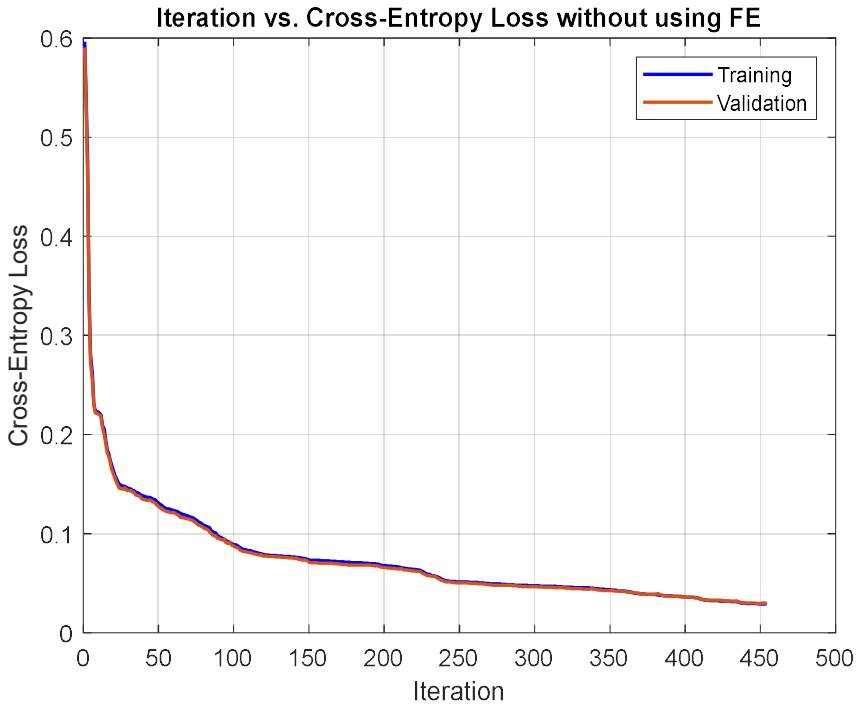

Figure 3. Iteration vs. cross-entropy loss for noisy signals with length 1600, and one hidden layer with ten nodes for DNN.

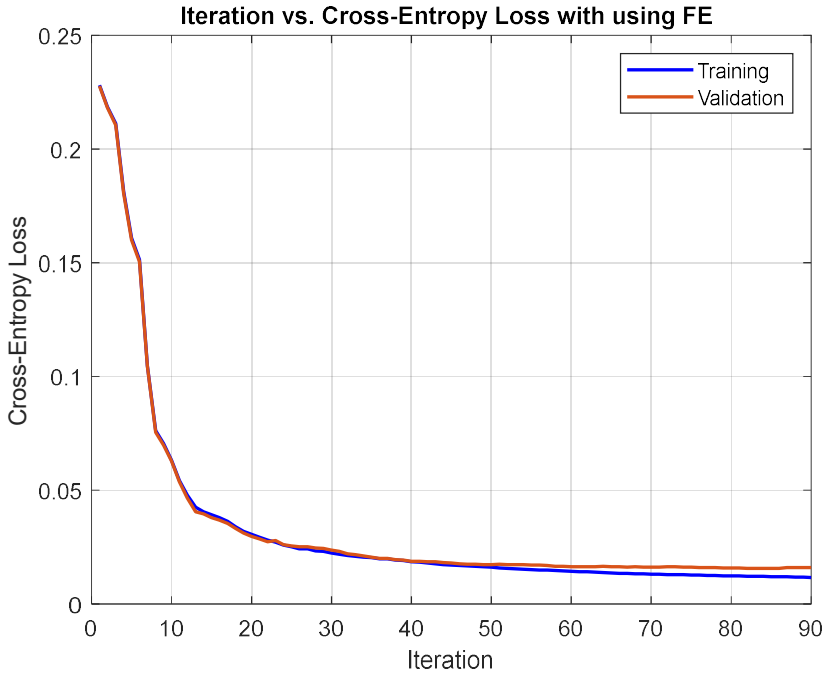

Figure 4. Iteration vs. cross-entropy loss for features extraction of noisy signals with length 1600 and one hidden layer.

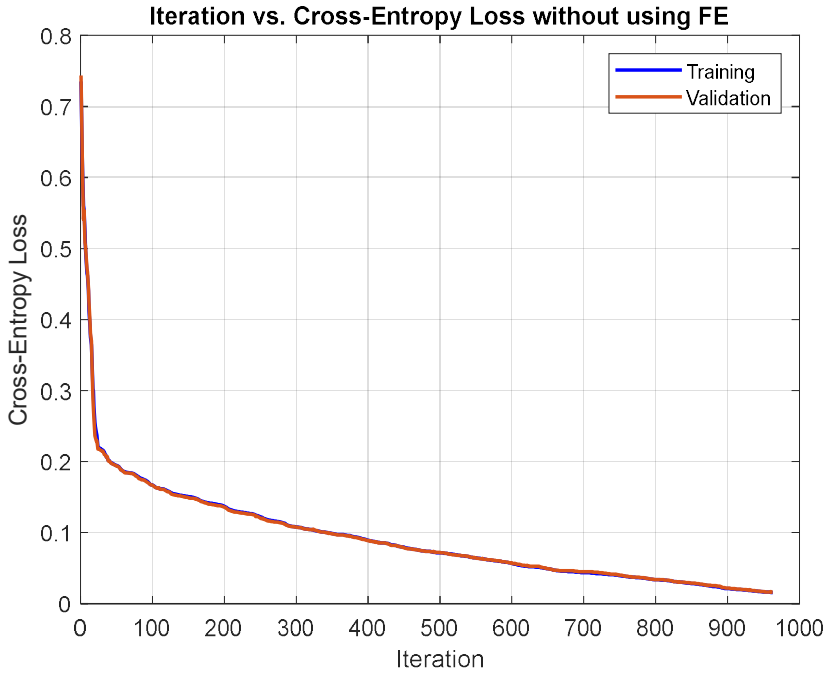

Figure 5. Iteration vs. cross-entropy loss for noisy signals with length 6400 and three hidden layers with (30, 25, and 20) nodes. Note that increased layers and nodes lead to increased accuracy of 96 and more iterations of 963.

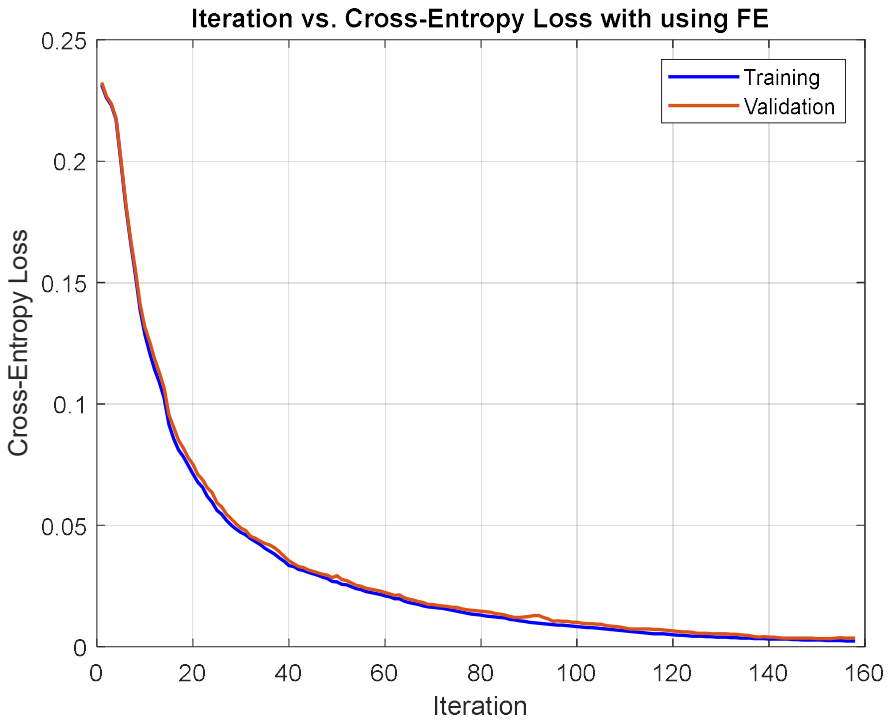

Figure 6. Iteration vs. cross-entropy loss for features extraction of noisy signals with length 6400 and three hidden layers with (30, 25, 20) nodes. Note that increased layers and nodes lead to an accuracy of 98 with fewer iterations of 158.

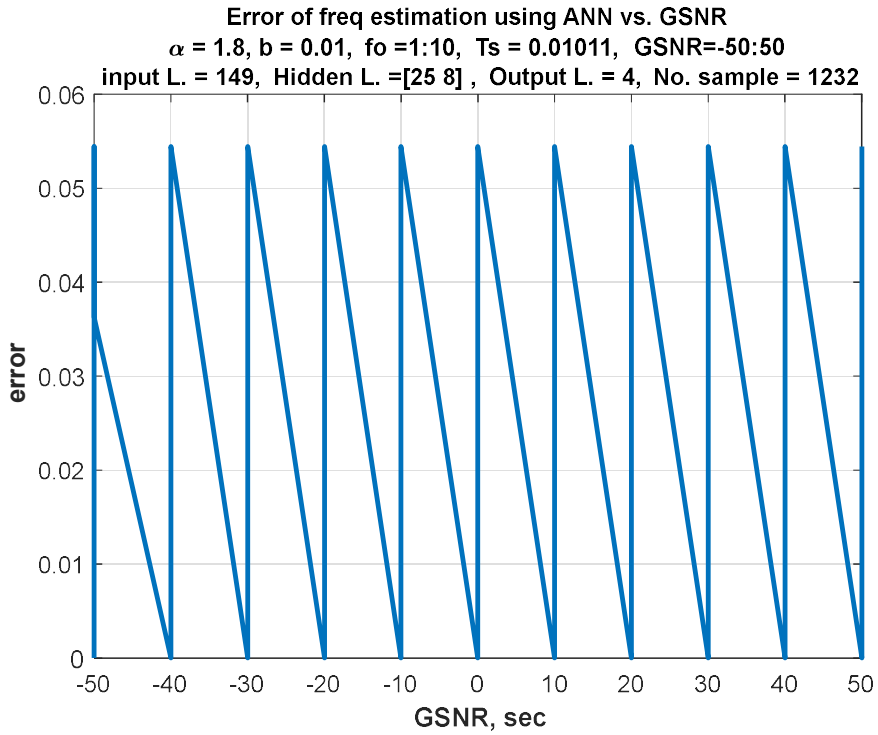

Figure 7. GSNR vs. RMSE of FE for noisy LFM by DNN.

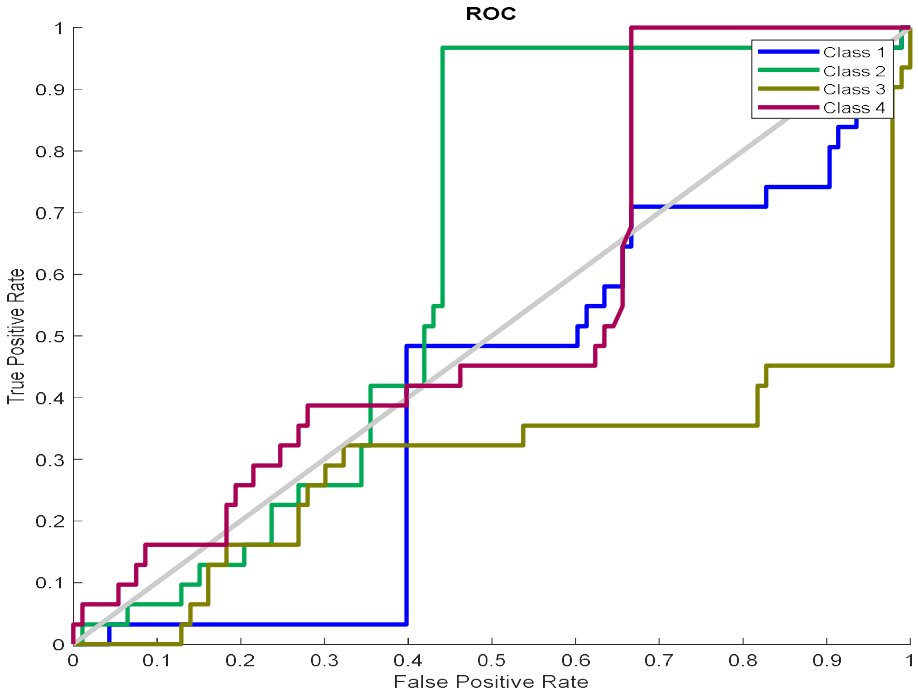

Figure 8. ROC of FE for noisy LFM by DNN.

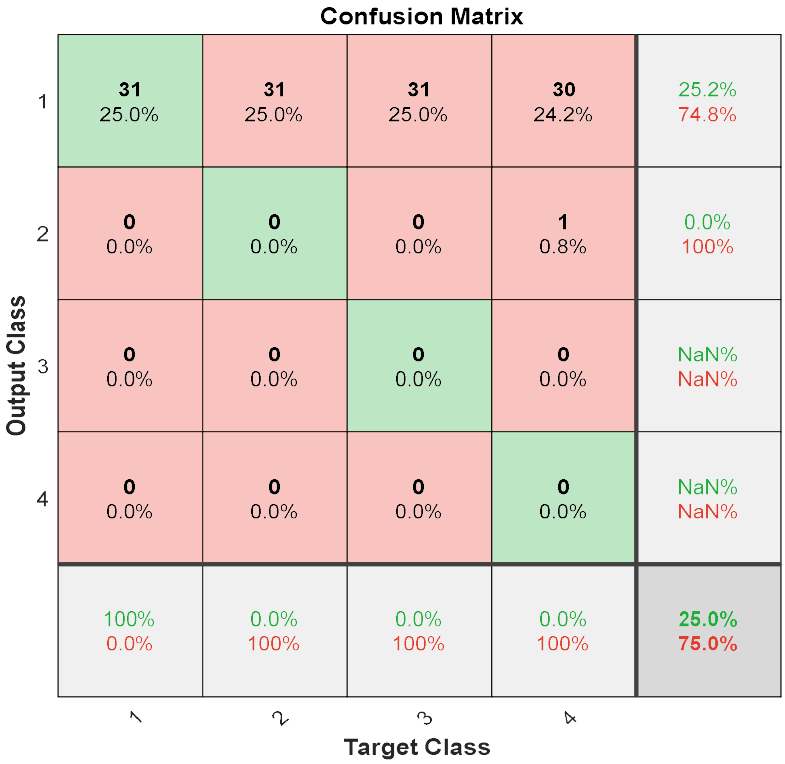

Figure 9. A confusion matrix for noisy LFM by DNN.

5.4. Estimation of IF and LCR by CNN

The CNN model's input layer includes a training set with 2D. The hidden layer (feature extraction stage) includes convolution, batch normalization, ReLU, max pooling, fully connected, and dropout. The output layer (classification stage) consists of a fully connected, softmax classification layer. The number of nodes in the input layer is 80×80.

The number of nodes in the hidden layer is based on the parameters for each layer. The number of nodes in the output layer equals ten; it represents the number of labels. The convolution layer includes 30 filters, where the filter size is 3×3, and the stride equals 1×1. Other convolution layers have different numbers of filters: 60, 90, and 128; max pooling with size 2×2, stride 2×2, zero padding; fully connected layer with 100 output nodes in feather extraction stage. The dropout layer ratio is 50%. A fully connected layer has ten output nodes in the classification stage. The classification layer uses cross-entropy. These layers are explained in Appendix B as shown in Table A1. Figure 10 shows the proposed CNN model layers. Algorithm 3 illustrates the CNN model procedure. CNN contains the network's components and represents the network's layers and options. It also needs two phases, forward and backward, for training and predicting the output. Each of them is explained as follows:

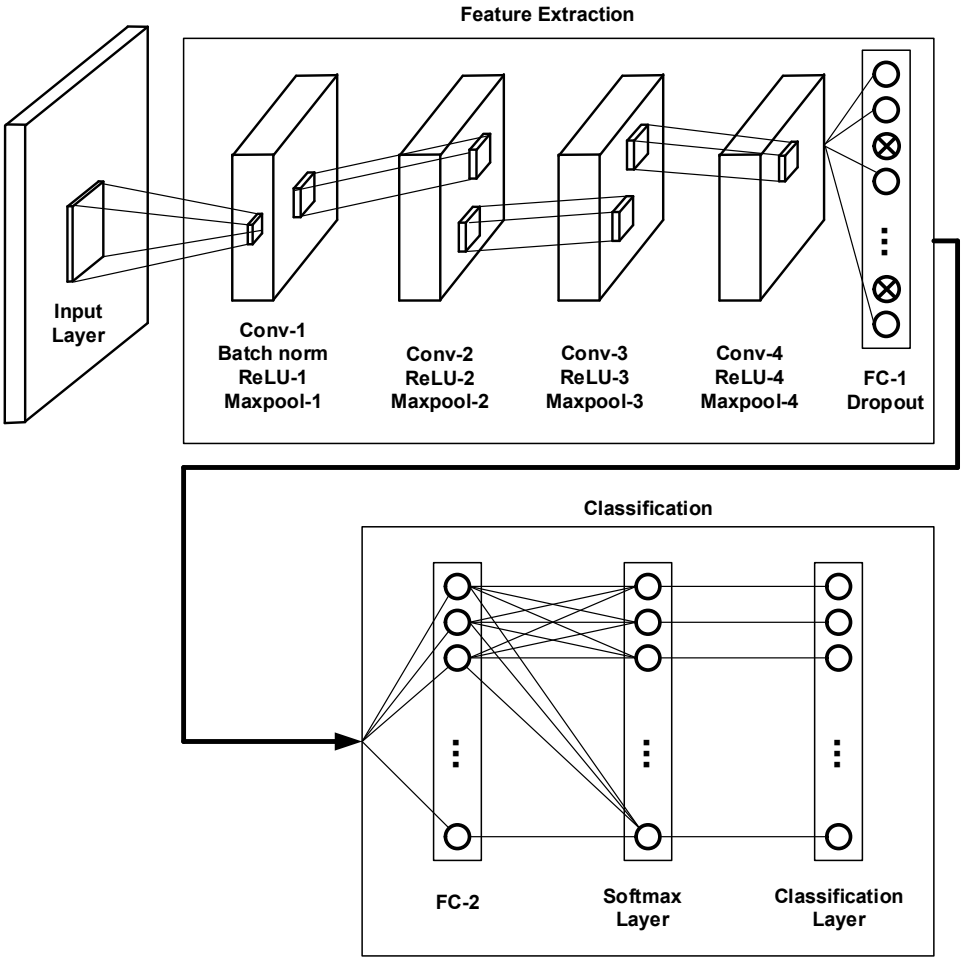

Figure 10. Proposed CNN model layers.

Forward Stage of CNN Model:

1. Batch normalization is applied on input signals; it speeds up training by halving the epochs (or faster).
2. Feature map is found by computing the sum of product for input node with weights (filters), then summation with bias is performed, where bias is used in the forward stage.
3. An activation function is applied to the sum of the product, where the activation function is ReLU.
4. Max pooling is applied; it is used to decrease the size of the feature maps.

5. The convolution layer is applied three times with the number of filters (60, 90, and 128), and each of them is followed by a ReLU layer and max pooling layer.
6. A fully connected network is applied: it is a feedforward neural network where all of the inputs from the previous layer are linked to each node in the next layer.
7. Dropout is applied: it is a method of stochastic regularization. It aids in the prevention of overfitting, and the accuracy and loss will gradually improve.
8. The classification stage includes three layers that are fully connected, followed by softmax, then a classification operation. The fully connected layer has ten output nodes: a feedforward neural network. The softmax is utilized as the activation function in the output layer for multi-class classification tasks. The loss function is finding the error for a single sample in data training that relies on actual and predicted labels. The loss function is cross entropy loss used to measure how well a classification model in deep learning performs. The loss (or error) is calculated as a number between zero and one, where the zero value referred the perfect model. The cost function is the summation of the loss function for all data training. The objective function is referred to as a cost function (cross-entropy).
9. The parameters of the CNN model are a learning rate of 0.0001, max epochs are 5, and mini-batch size is 8. At the same time, Adam uses an initial learning rate is 0.001, Epsilon is 0.00000001, squared gradient decay factor is 0.999, and Gradient decay factor is 0.9. Hyperparameters in convolution are: the number of filters (30, 60, 90, and 128), the filter size is 3×3, and padding and stride are equal to one. In batch normalization, the parameters are ε, B, and λ, max pooling uses the size 2×2, and the stride is 2×2. In the dropout layer, the probability of dropout is 50%, and in the fully connected layer, the number of classes is 10.

Backward Stage of CNN Model:

1. In this stage, an optimization algorithm is needed to update the parameters, and each layer's error should be calculated. The optimization algorithm controls how the parameters of the neural network are adjusted. In this work, Adam is used as the optimization algorithm
2. The weights and biases update is given by:

$$\text{New\_parameters} = \text{Old\_parameters} + \text{R}_{\text{Adam}} \times \text{learning\_rate} \tag{22}$$

3. where New_parameters are new weights and biases, Old_parameters are old weights and biases, and $\text{R}_{\text{Adam}} = \text{m}_{\text{t}}^{\sim} / \sqrt{\text{v}_{\text{t}}^{\sim} + \epsilon}$ as explained in reference [22], Adam is updating the weights and biases relying on the derivative of the loss function (entropy). The derivative of the loss function is applied in the backward stage for CNN, while the loss function is applied in the forward stage for CNN. It measures the performance of the network. The training set is evaluated for each epoch using the validation set. The validation loss is similar to the training loss and is calculated as the sum of the errors for each sample in the validation set. Forward and backward stages are performed until reaching the last epoch (end of training).
4. The final step is to evaluate the CNN model by applying a testing set to the trained CNN model to predict the label (class) for the test data. The CNN model is evaluated using relevant metrics such as accuracy.
5. After the class number (label) prediction, it is possible to estimate the frequency and slope to which it belongs. Then it is possible to calculate the amount of error and the accuracy of the predicted parameters versus the true parameters as follows:
   - True frequency (T_f) is the target frequency of a signal
   - Estimate frequency (P_f) is the predicted frequency of the CNN network
   - The relative absolute error of the frequency $E_f = \left| T_f - P_f \right| / T_f$
   - True slope (T_S) is the target slope of a signal
   - Estimate slope (P_S) is the predicted slope of the CNN network

- The relative absolute error of the slope is $E_s = |T_s - P_s| / T_s$
- Estimate IF is $IF = P_f + P_s\ t$

Algorithm 3: CNN Model Procedure

Input: Training set $(XTr,\ LTr)$, validation set $(XVa,\ LVa)$, and testing set $(XTe,\ LTe)$, where each label is represented by the target frequency (T_f) and target slope (T_S).
Output: Frequency and slope estimation.
Begin:
Step 1: A training stage

1. The input layer is the training set $(XTr,\ LTr)$ with length N, and validation set $(XVa,\ LVa)$.

2. Divided training set into min-batches, where mini-batch size (MB) equals 8. Iteration is one-time processing for forward and backward for a batch of samples. Iterations per epoch = number of training samples ÷ mini-batch size. Iterations = iterations per epoch × number of epochs.

3. Find the batches, where $batches = \frac{N}{MBS}$.

4. Find batch list, where $blist = 1 \rightarrow MB \rightarrow (N - MB + 1)$

5. **For epoch** =1 → No of epoch

6. **For iteration** =1 → batches

7. Select data by $b = blist(\text{iteration})$

8. **For** $k = b \rightarrow b + MB - 1$

9. The convolutional layer applies convolutional operation between input signals and filters, where the number of filters equals 30.

10. A batch normalization layer is applied as a result of step 2.

11. ReLU is used as an activation function.

12. Max pooling layer is applied to reduce the features map.

13. The convolution layer is applied with several filters 60 to find feature maps, and then ReLU and max pooling layer are used.

14. The convolution layer is applied with several filters 90 to find feature maps, and then ReLU and max pooling layer are used.

15. The convolution layer is applied with the number of filters 128 to find feature maps and then used, ReLU and max pooling layer.

16. A fully connected layer and dropout layer are applied to avoid overfitting.

17. A fully connected layer is used with an equal output number of classes.

18. Softmax is applied to find the output of the network that presents the predicate label $(\hat{LTr})$, where each label is predicate frequency and slope.

19. In the classification layer, calculate the performance of the network by entropy as follows:

$$ETr_j = -\sum_{i=1}^{n} LTr_i \cdot \log\left(\widehat{LTr_i}\right)$$

where n is the number of classes, $\hat{LTr}_i$ is predicate labels, LTr_i is actual labels in one-hot encoding, $j \in [1, \ldots,\ N]$, N is the number of samples training.

20. In the backward stage, calculate the error by the derivative of the loss function and compute the change of the weights $(\Delta \mathrm{w})$ as follows:

$$E = LTr - \widehat{LTr}$$

$$ML_k = \left(LTr = \widehat{LTr}\right)$$

$$\Delta \mathrm{w} = \frac{\widehat{m}_t}{\sqrt{\widehat{v}_t + \epsilon}}$$

21. **End of k loop**

22. Cross entropy is computing the cost function of the network. The sum of loss for each batch in the training set calculates the cost function of the training loss function:

$$Loss_{Tr} = \frac{1}{N}\sum_{j=1}^{N} ETr_j$$

Then compute the accuracy of the training data set:

$$Acc_{Tr} = \frac{\sum ML}{\text{size of minibatch}}$$

23. The weights (w_t) are updated as follows:

$$w_t = w_{t-1} - \eta.\ \Delta \text{w}$$

24. Evaluate the network during training by using validation data for each iteration, where applied same network layers on validation data and new weights are used to predict the label of validation data, and the cost of loss for validation is similar to the cost of loss for training as follows.
Moreover, find loss as follows:

$$Loss_{Va} = -\frac{1}{N_v}\sum_{i=1}^{Nv}\sum_{j=1}^{n} LVa_{ij} \cdot \log\left(\widehat{LVa}_{ij}\right)$$

where n is the number of classes, $\widehat{LVa}_{ij}$ is predicate labels, LVa_{ij} is actual labels in one-hot encoding, $j \in [1, \ldots, N_v]$, N_v is number of validation samples. The accuracy of validation is:

$$Acc_{Va} = \frac{\sum_{j=1}^{Nv} LVa_j = \widehat{LVa}_j}{N_v}$$

25. **End of iteration**

26. **End of epoch**

27. CNN model trained (CNN_{Train}) is returned.

Step 3: Testing stage

1. The testing set $(XTe,\ LTe)$ It used the CNN model trained, which has the same layers but uses optimal weights.

2. The CNN trained is applied on the testing set to predict the label, where each label is predicate frequency and slope.

• Estimate frequency (P_f) is predicted frequency and estimated slope (P_S) is the predicted slope from the predicated label (P_{label}) of CNN network as follows:

$$P_{label} = CNN_{Train}(XTe)$$

• Compute the error of frequency and slope as follows:
• The relative absolute error of the frequency $E_f = \left|T_f - P_f\right| / T_f$
• The relative absolute error of the slope is $E_s = |T_s - P_s| / T_s$
• Estimate IF is $IF = P_f + P_s\ t$

3. Evaluation of the CNN model trained using metrics is accuracy, precision, recall, F-Measure, and ROC.

End of algorithm

6. IF Estimation Based on TFD

The Fourier transform cannot detect the time-varying properties of nonstationary signals with time-varying frequency content (such as FM and bio-logical signals). This is due to the FT's use of a time-averaging process (time integration). TFDs are two dimensional double transforms from the time domain to the time–frequency domain representing the Fourier transform of an analytical signal's instantaneous autocorrelation. The most

straightforward formula for a time–frequency distribution is the short-time Fourier transform (STFT), which is a windowed frequency distribution [27,28]. STFT is a sequence of Fourier transforms of a windowed signal; it is used to determine the sinusoidal frequency of local sections of a signal as it changes over time. STFTs divide a longer time- signal into segments of equal length and then compute the Fourier transform separately on each segment. A nonstationary signal is one with a changing frequency content over time. A nonstationary signal's spectrogram estimates its frequency content's time evolution. TFD IF estimation in the presence of noise AWGN and SαSN in FM signals. TFD and STFT are used to estimate the IF for analytical signals. Before frequency calculation, the analytic noisy signal is computed by using the Hilbert transformation. A real-valued signal's Fourier Transform (FT) is complex and symmetric. This implies that the content at negative frequencies is superfluous. Negative frequencies may cause aliasing while analyzing FM. The analytic signal helps avoid aliasing. Although the analytic signal is complex valued, its spectrum is unipolar (only positive frequencies). The real part of the analytic signal is the original signal, and the imaginary part is the signal's Hilbert Transform (HT). To pass complex-valued data to a neural network, you can use the input layer to split the complex values into their real and imaginary parts before it passes the data to the subsequent layers in the network. See reference [29].

We calculate the spectrogram of STFT (spec$(t.f)$), then estimate the IF from the peak (max) of the spec as follows:

$$\hat{f} = \arg(\max\{\mathrm{spec}(t,f)\}), 0 \le f \le \frac{f_s}{2} \tag{23}$$

Then, we calculate the relative squared error for each GSNR as follows:

$$e = \left| \left(\hat{f} \times df - \mathrm{IF}_t \right) / f_o \right|^2 \tag{24}$$

where f_o fundamental frequency, $\hat{f}$ estimated frequency, IF_t theoretical IF with spectrogram timing, $df = \frac{f_s}{N}$ and $N = 1024$. TFD estimated IF using spectrogram and pspectrum MATLAB functions, pspectrum differs from spectrogram in segment lengths, overlapping segments, and window. Spectrogram length $= 1 \times \left[\frac{N}{2} + 1 \right]$, while pspectrum length $= 1 \times N$. Pspectrum controls the length of the segments and the overlap between adjacent segments using the time resolution and overlap percent pair arguments. It divides the signal into overlapping segments and applies a Kaiser window to each segment.
The algorithms (4, 5, and 6) illustration frequency estimation by TFD, Hilbert transform, and spectrogram by short time Fourier transform respectively as shown in Appendix C.

7. Discussion of Results

This section simulates the instantaneous frequency and slope estimation of FM signals using additive white Gaussian noise and symmetric stable noise. Simulation was performed with MATLAB under Academic License 40635944. $[-50\ 50]$ dB is the GSNR range. The network learns from the input data and predicts the frequency and slope. The DNN and CNN models were used to simulate frequency and slope estimation for LFM signals. The results show high accuracy for parameter estimation by confusion matrix and some measures such as accuracy, precision, F1-score, FNR, FPR, and ROC, as well as few errors rate, and SαS is an impulsive model, where alpha is more harmful even if it is of small value, where it affects the slope and frequency guess. A variable b determines the ratio of AWGN and SαSN.

Figure 11 shows α-stable probability density functions with different parameters. Figure 12 shows alpha-stable noise in the time domain; it is impulsive. Figure 13 shows single-tone and noise signals. Figures 14 and 15 show frequency estimation of single-tone (ST) and LFM signals by DNN. Figure 16 shows the slope estimation of single-tone and LFM signals by DNN. Figure 17 show the accuracy and loss rate of FE and SE for

noisy LFM. Figure 18 show confusion matrices for FE and SE for noisy LFM. Figure 19 show the accuracy and loss rate of FE and SE for noisy LFM. Figure 20 show confusion matrices for FE and SE for noisy LFM. Tables 1 and 2 show the performance evaluation criteria of noisy LFM signals. Figure 21 shows the accuracy of the frequency estimation of LFM. Figure 22 shows the test error of frequency estimation for LFM. Figure 23 shows the accuracy of the slope estimation of LFM. Figure 24 shows the test error of slope estimation for LFM. Figures 25 and 26 show MSE versus GSNR for TFD of a noisy single-tone signal by spectrogram and pspectrum, where α = 1 and b = 20. Figures 27 and 28 show MSE versus GSNR for TFD of noisy LFM signal by spectrogram and pspectrum, where α = 1 and b = 20. Figures 29 and 30 show the accuracy of FE for noisy single tone and LFM signals by TFD (pspectrum). Figure 31 shows the accuracy of FE for noise LFM by DNN and TFD (spectrogram and pspectrum). Figure 32 shows the test error of FE for noisy LFM by DNN and TFD (spectrogram and pspectrum). Figure 33 shows the test error rate for SE of noisy LFM by CNN, where f_o = 19.0005. Figure 34 shows ROC for noisy LFM by 2D-CNN. Figure 35 shows ROC for noisy LFM by 2D-CNN, where the epoch number equals 30. Table 3 shows measures of the 1D-CNN model for noisy LFM signals. Figure 36 shows ROC for noisy LFM by 1D-CNN. Figure 37 signals in the time domain with different frequencies, LCR, and GSNR. Figure 38 shows 2D-signals in the time domain with different initial frequencies. Figure 39 shows signals in time frequency distribution.

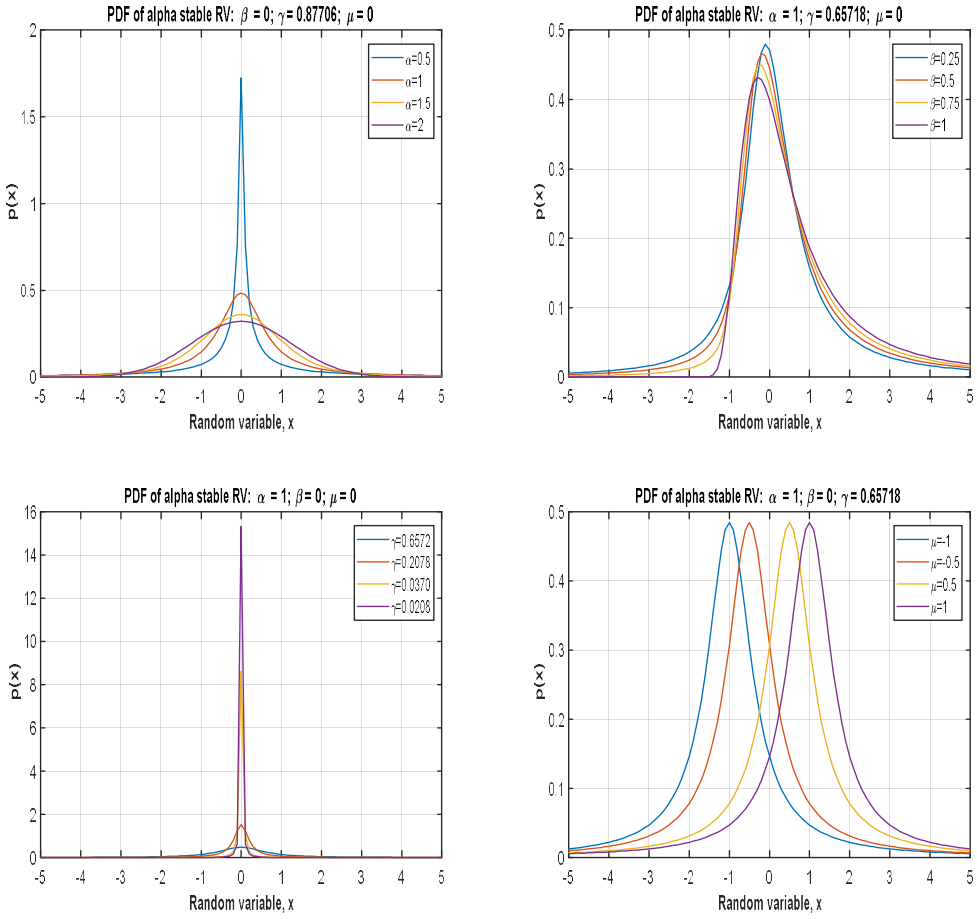

Figure 11. α-Stable probability density functions with different parameters.

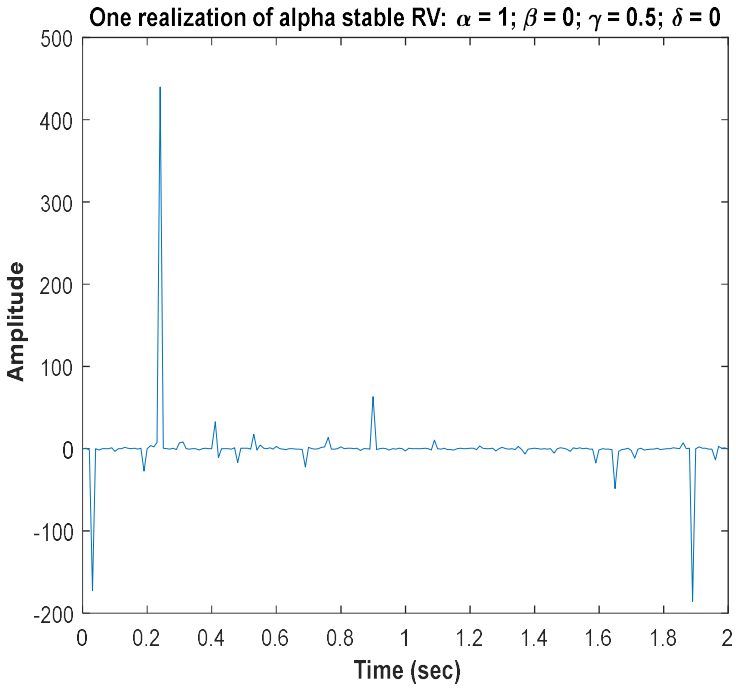

Figure 12. α-Stable noise in time domain. It is impulsive.

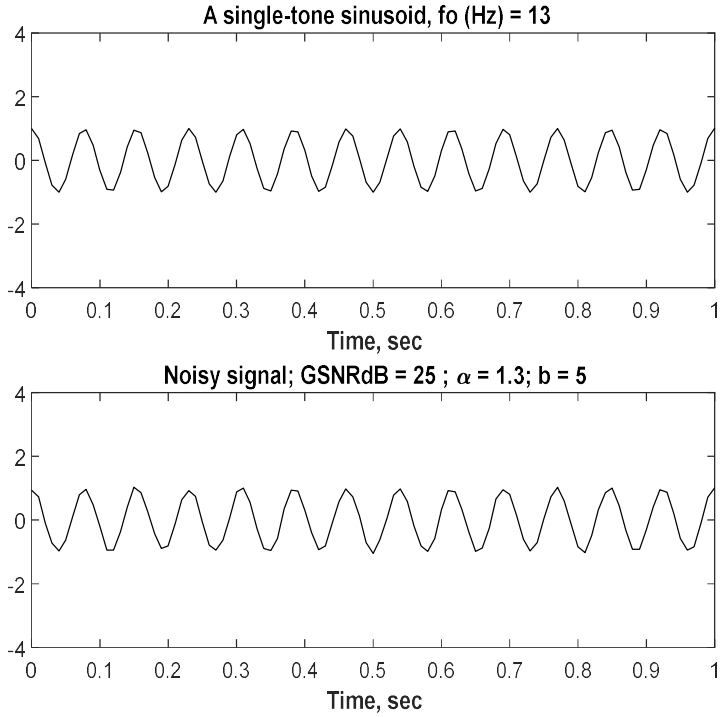

Figure 13. A single tone and noise signals.

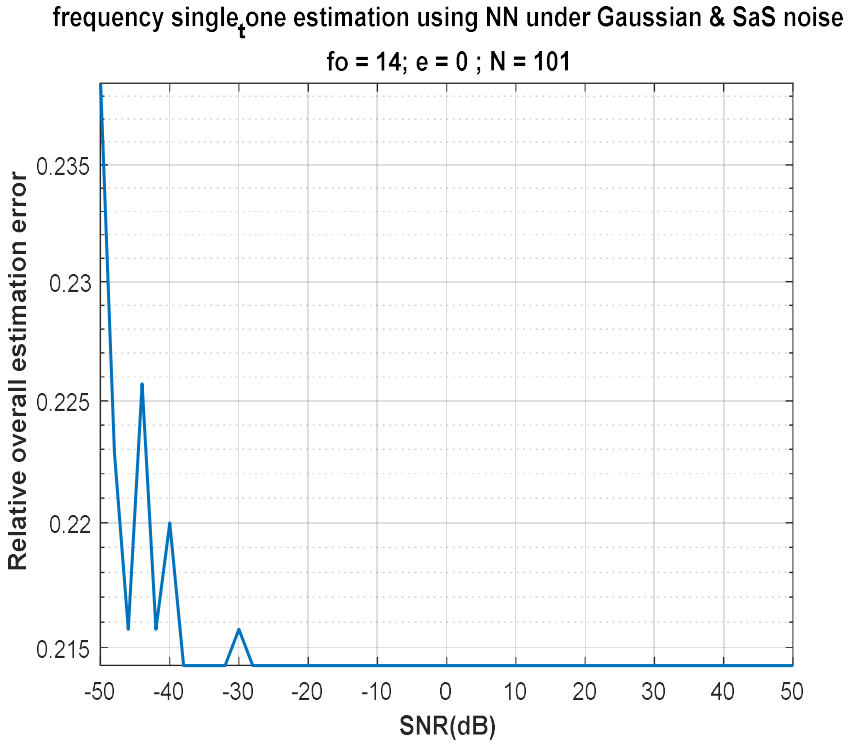

Figure 14. FE of noisy single tone signals by DNN.

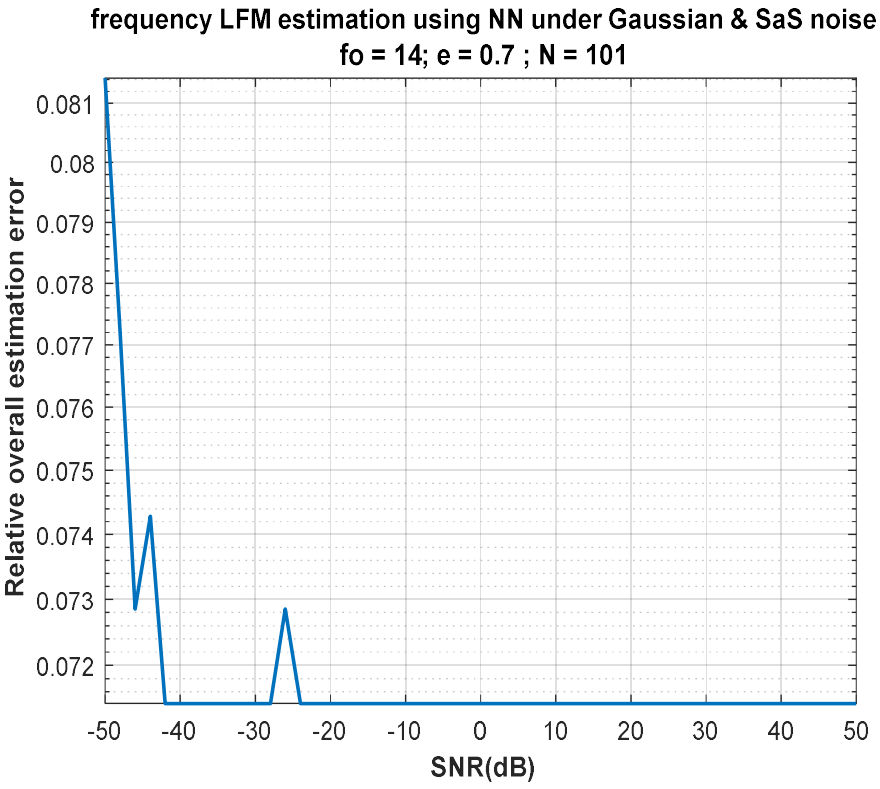

Figure 15. Error rate for FE of noisy LFM signals by DNN.

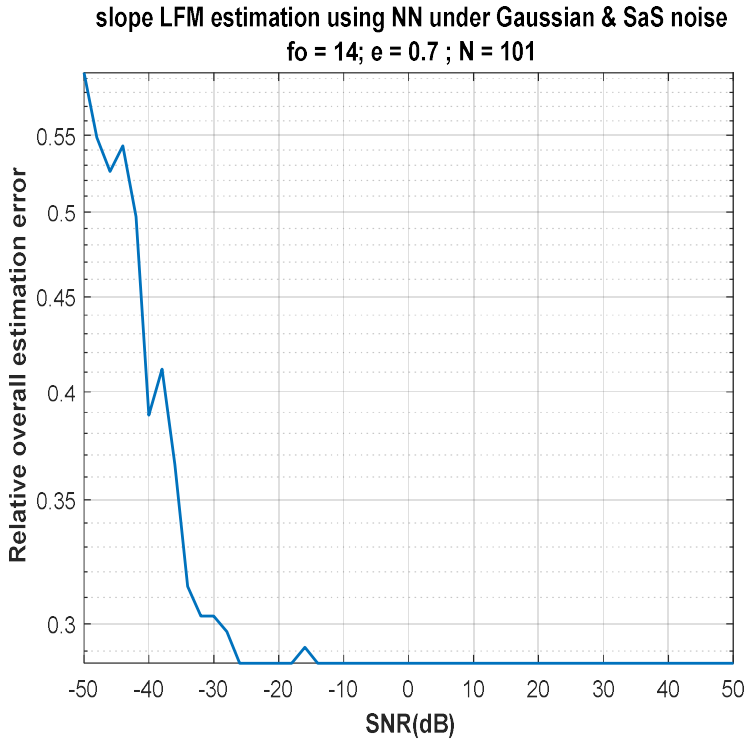

Figure 16. Error rate for SE of noisy LFM signals by DNN.

Figure 17. Accuracy and loss rate of FE for noisy LFM by CNN.

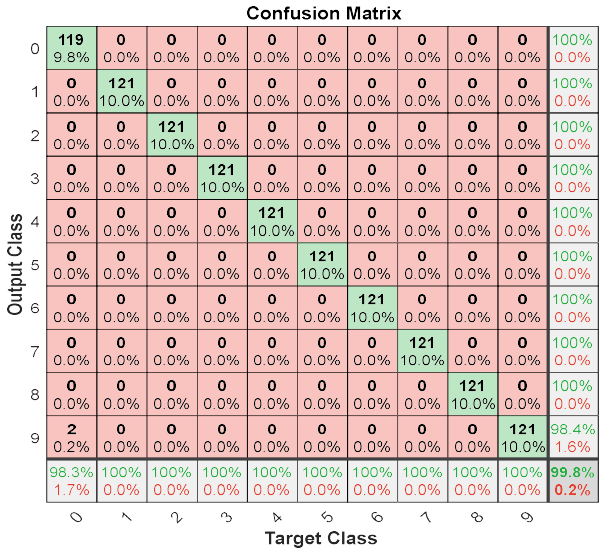

Figure 18. A confusion matrix of FE for LFM signals by CNN.

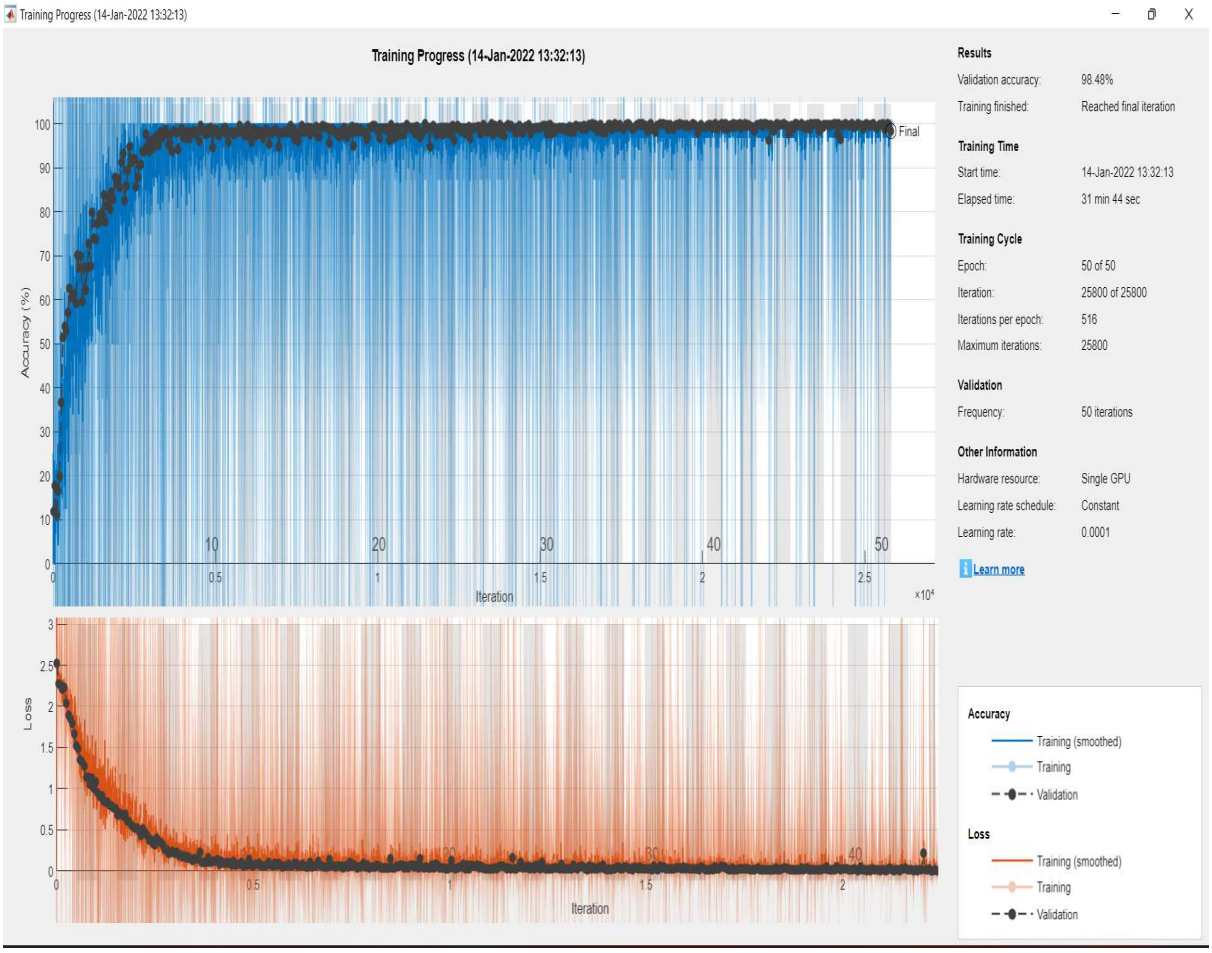

Figure 19. Accuracy and loss rate of SE for noisy LFM by CNN.

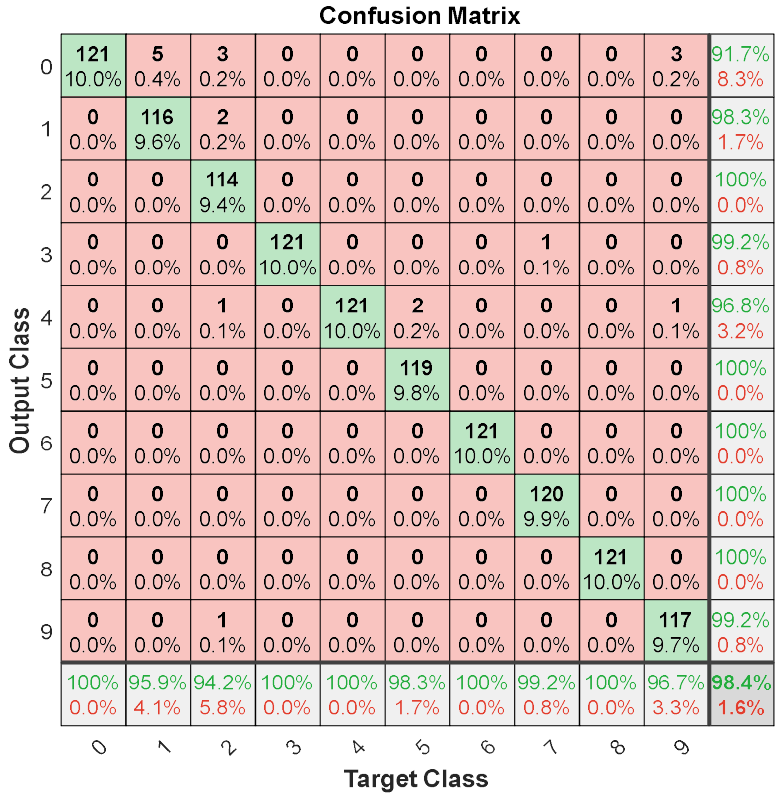

Figure 20. A confusion matrix of SE for LFM.

Table 1. Measures of FE and SE for noisy LFM signals.

Measures	LFM	
	Frequency	**Slope**
Accuracy	99.8118	98.4431
Precision	99.8303	99.0445
Recall	99.8118	98.4432
F1_Score	99.8210	98.5477
FNR	0.0039	0.0216
FPR	0.0037	0.0195

Table 2. Measures of 1D-CNN model for noisy LFM signals.

Measures	Values
Accuracy	56.6575
Precision	57.5019
Recall	56.6575
F1_Score	57.0766
Epoch	10

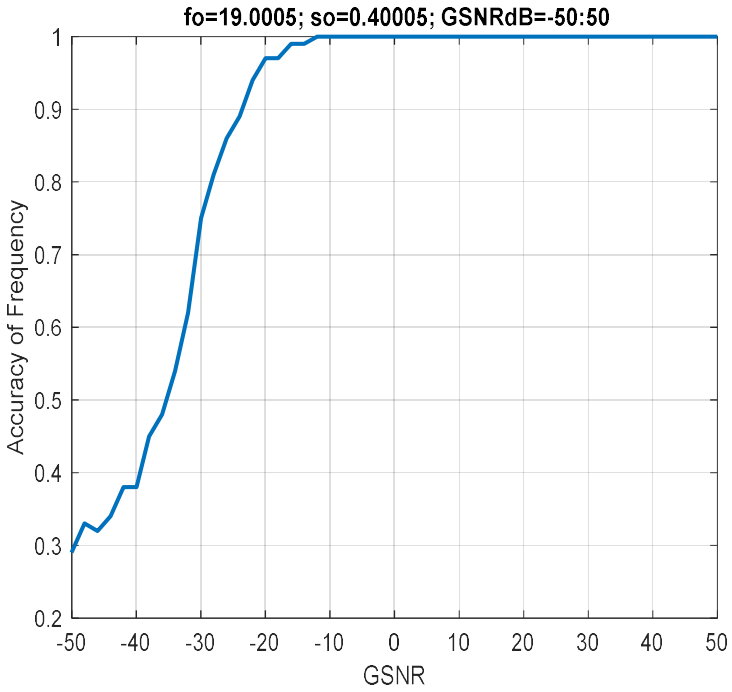

Figure 21. Accuracy of FE for noisy LFM by CNN, where f_o = 19.0005.

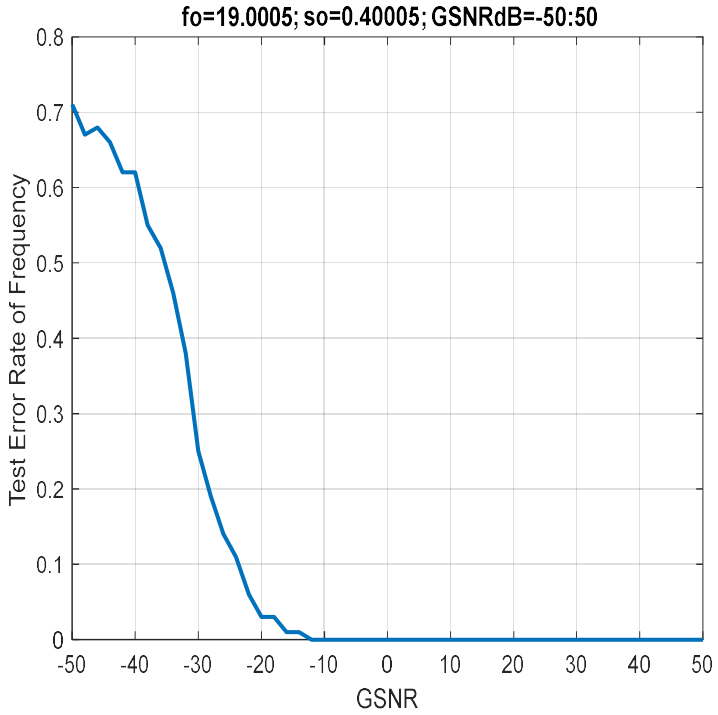

Figure 22. Test error rate of FE for noisy LFM by CNN, where f_o = 19.0005.

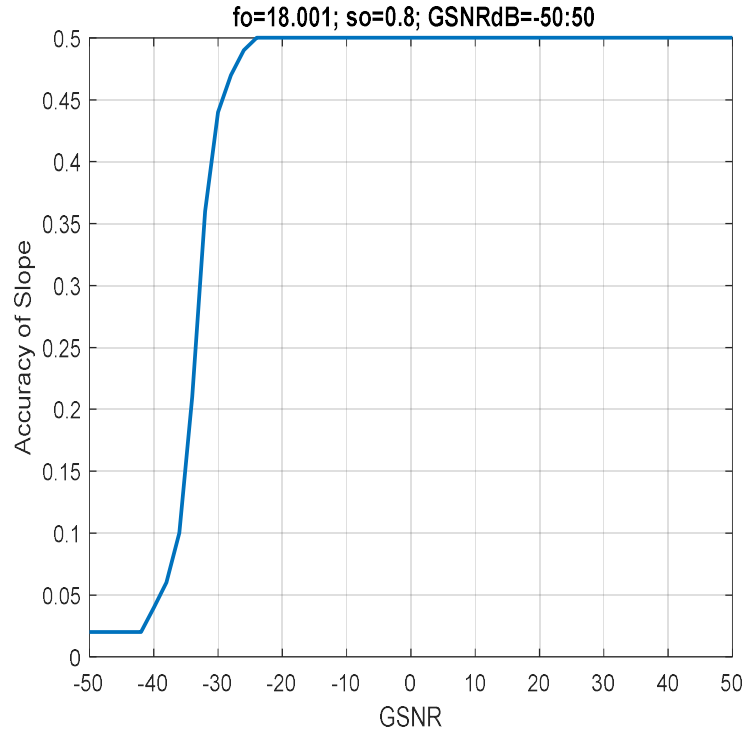

Figure 23. Accuracy of SE for noisy LFM by CNN, where f_o = 18.001.

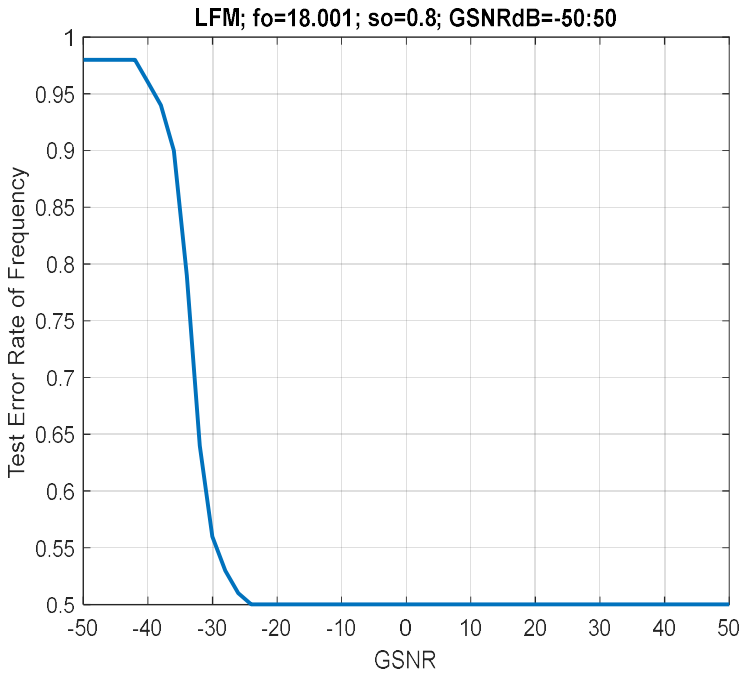

Figure 24. Test error rate for SE of noisy LFM by CNN, where f_o = 18.001.

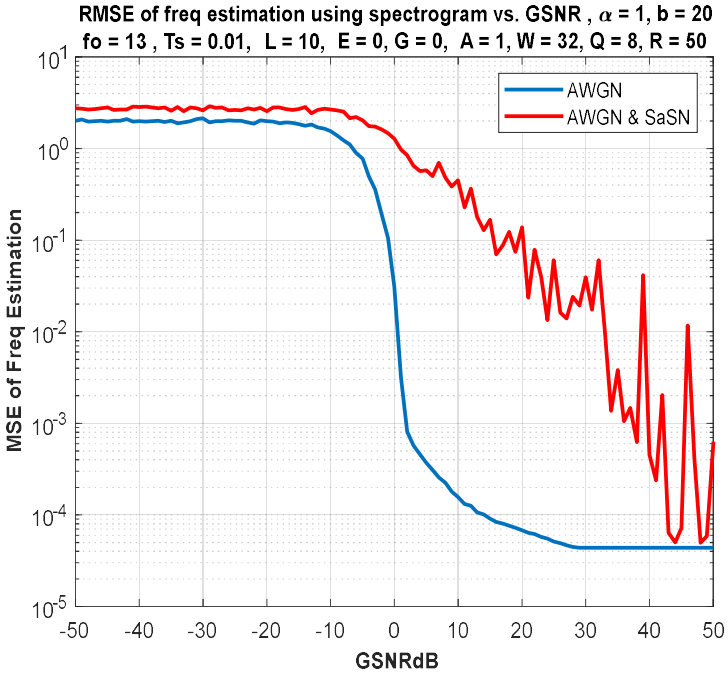

Figure 25. MSE versus GSNR for TFD of noisy single tone signal by spectrogram, where $\alpha = 1$ and b = 20.

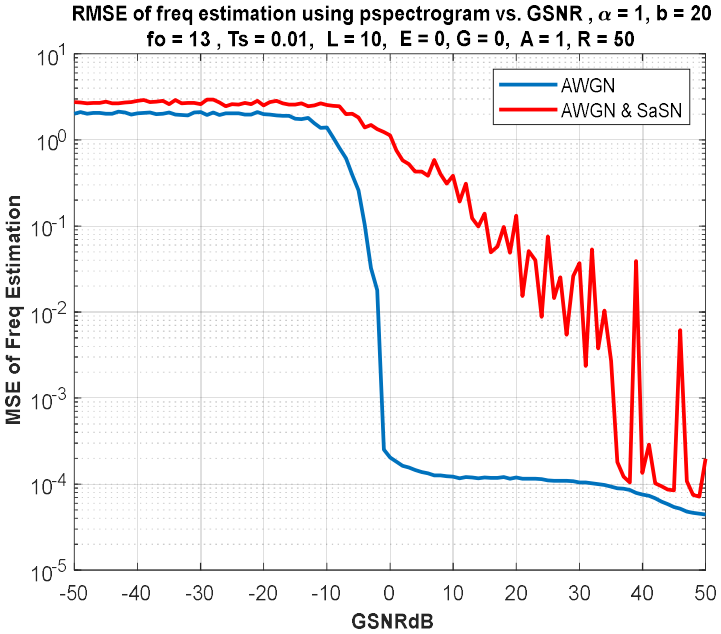

Figure 26. MSE versus GSNR for TFD of noisy single tone signal by pspectrum, where $a = 1$ and $b = 20$.

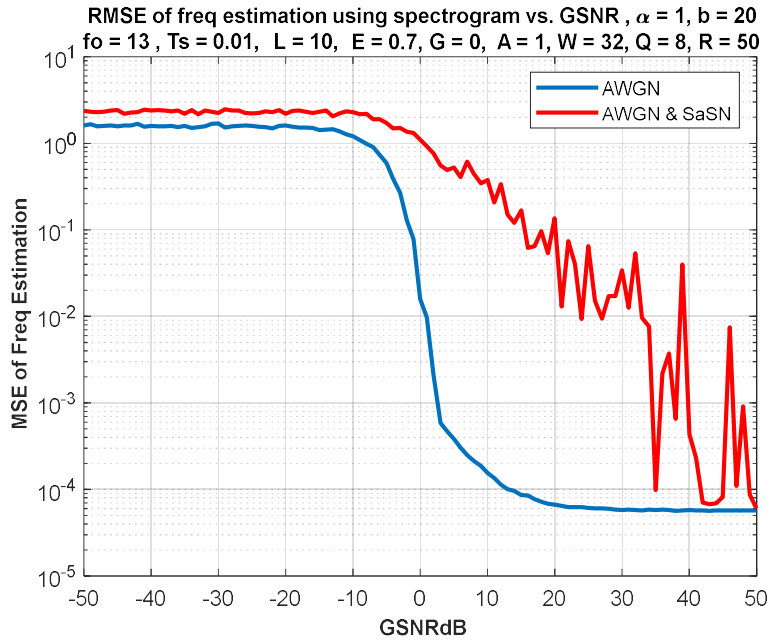

Figure 27. MSE versus GSNR for TFD of noisy LFM signal by spectrogram, where $\alpha = 1$ and b = 2.

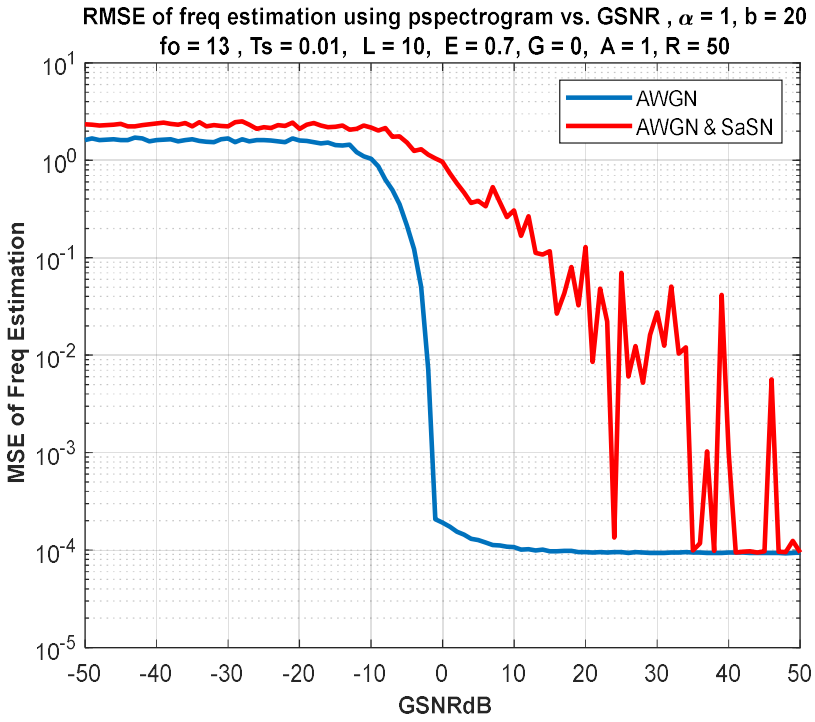

Figure 28. MSE versus GSNR for TFD of noisy LFM signal by pspectrum, where $\alpha = 1$ and $b = 20$.

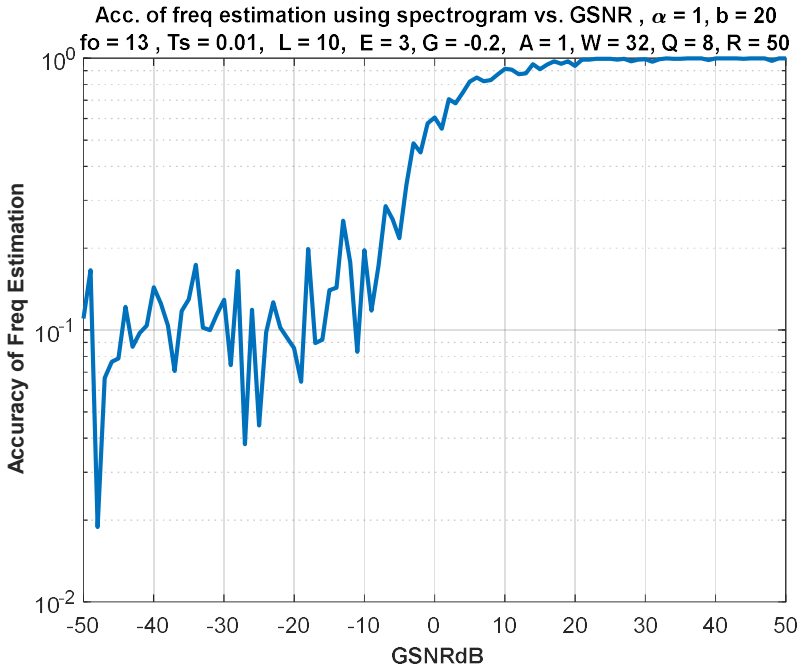

Figure 29. Accuracy of FE for noisy single tone signal by TFD (pspectrum).

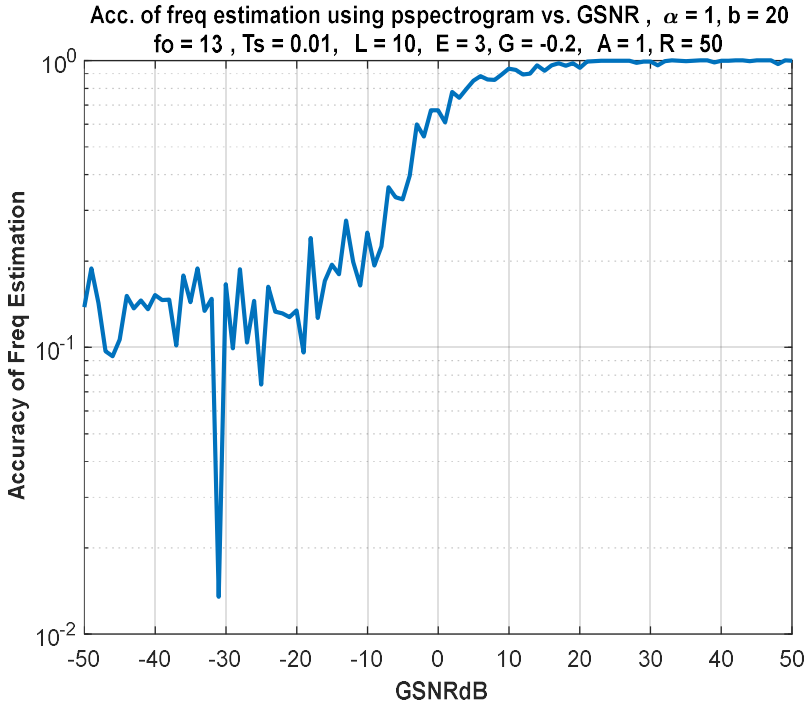

Figure 30. Accuracy of FE for noisy LFM by TFD (pspectrum).

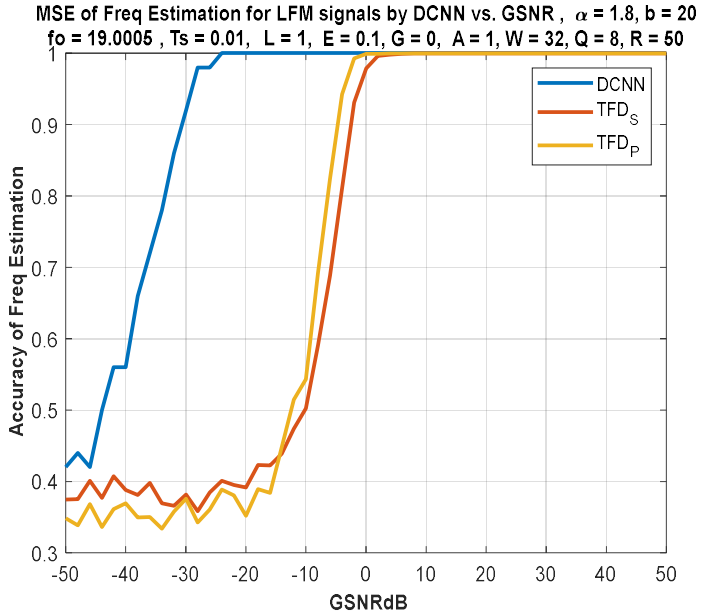

Figure 31. Accuracy of FE for noisy LFM (AWGN and SαSN) by DNN and TFD for spectrogram (TFD_s) and pspectrum (TFD_p).

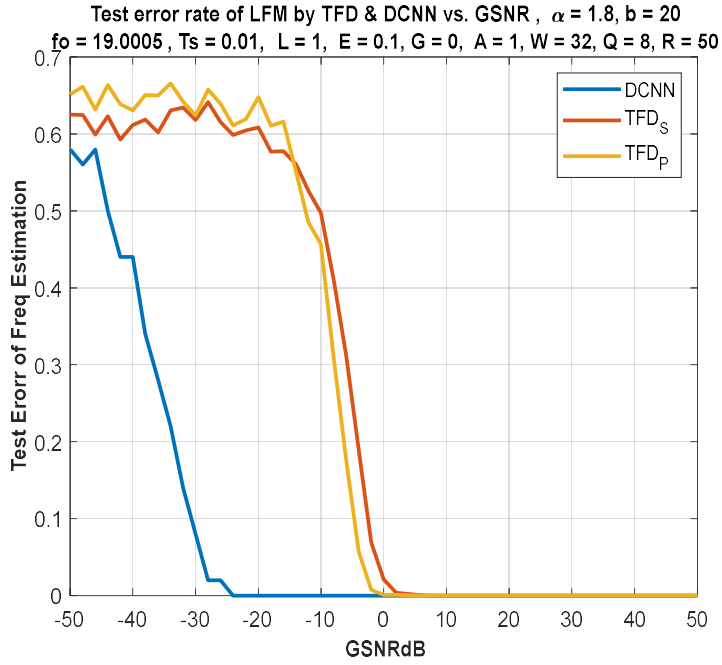

Figure 32. The error rate of FE for noisy LFM (AWGN and SαSN) by DNN and TFD for spectrogram (TFD_s) and pspectrum (TFD_p).

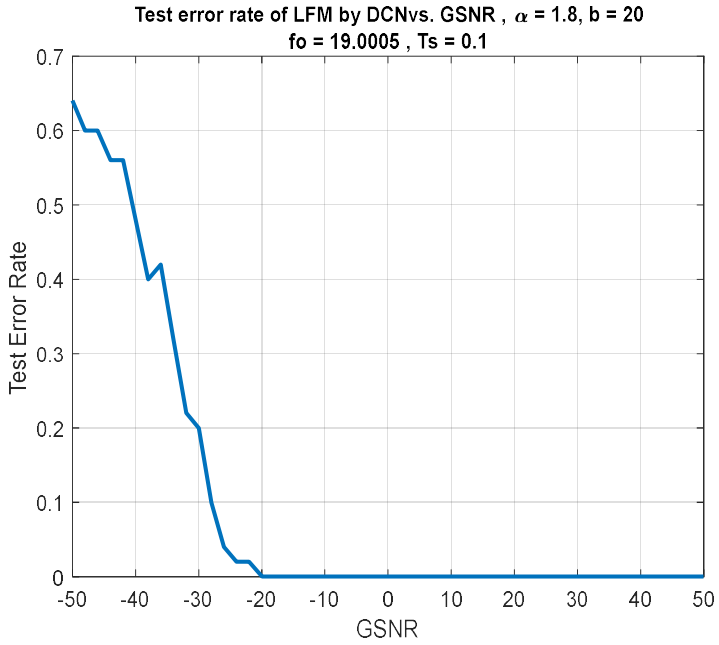

Figure 33. Test error rate for SE of noisy LFM by CNN, where $f_o = 19.0005$.

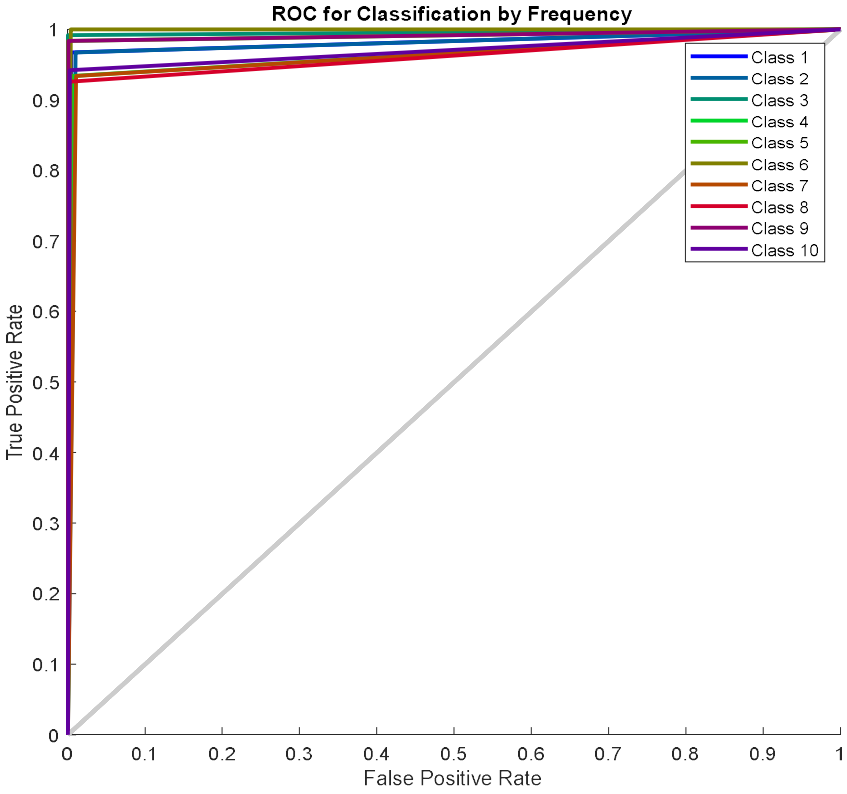

Figure 34. ROC for noisy LFM by 2D-CNN.

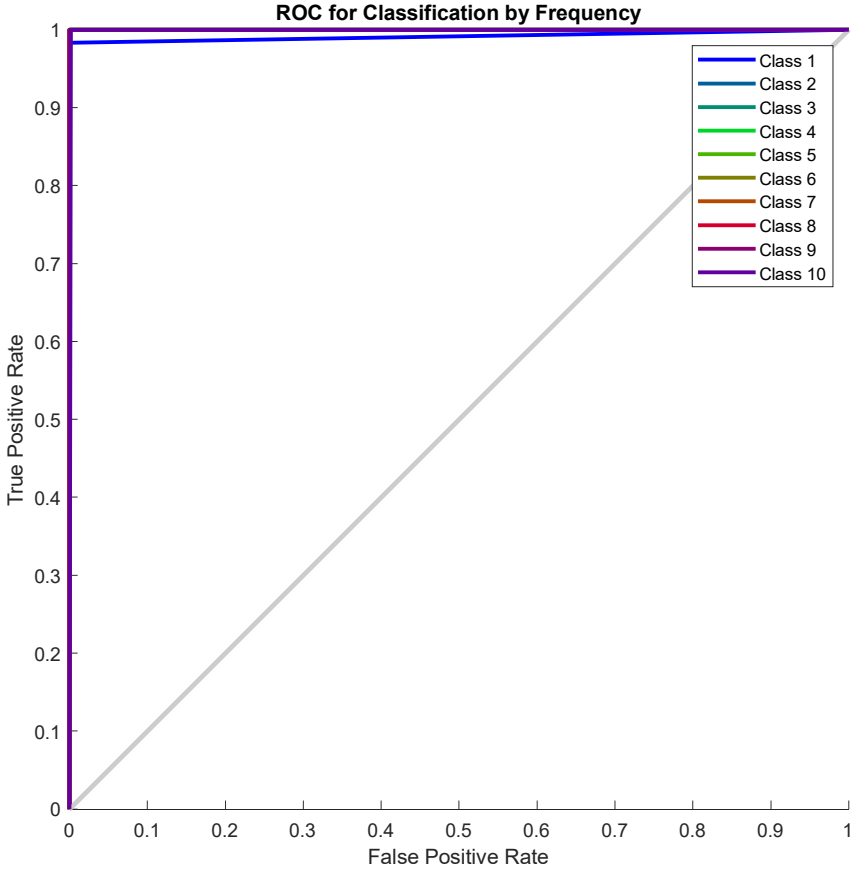

Figure 35. ROC for noisy LFM by 2D-CNN, when epoch equal 30.

Table 3. Training parameters of noisy LFM dataset. Best performance is shown in bold.

Parameters		Accuracy	Recall	Precision	F1-Score
Learning rate	10^{-3}	90.4959	90.4959	91.5224	91.0062
	10^{-4}	**96.2810**	**96.2810**	**96.3523**	**96.3166**
	10^{-5}	70.4959	70.4959	77.6223	73.8876
Epoch	5	96.2810	96.2810	96.3523	96.3166
	10	97.6033	97.6033	97.7144	97.6588
	30	**99.8347**	**99.8347**	**99.8374**	**99.8361**
Min-patch	8	96.2810	96.2810	96.3523	96.3166
	32	97.6033	97.6033	97.7144	97.6588
	64	**99.8347**	**99.8347**	**99.8374**	**99.8361**
Training Method	SGDA	74.2149	74.2149	82.8702	78.3041
	RMSPROP	93.7190	93.7190	94.2774	93.9974
	Adam	**96.2810**	**96.2810**	**96.3523**	**96.3166**

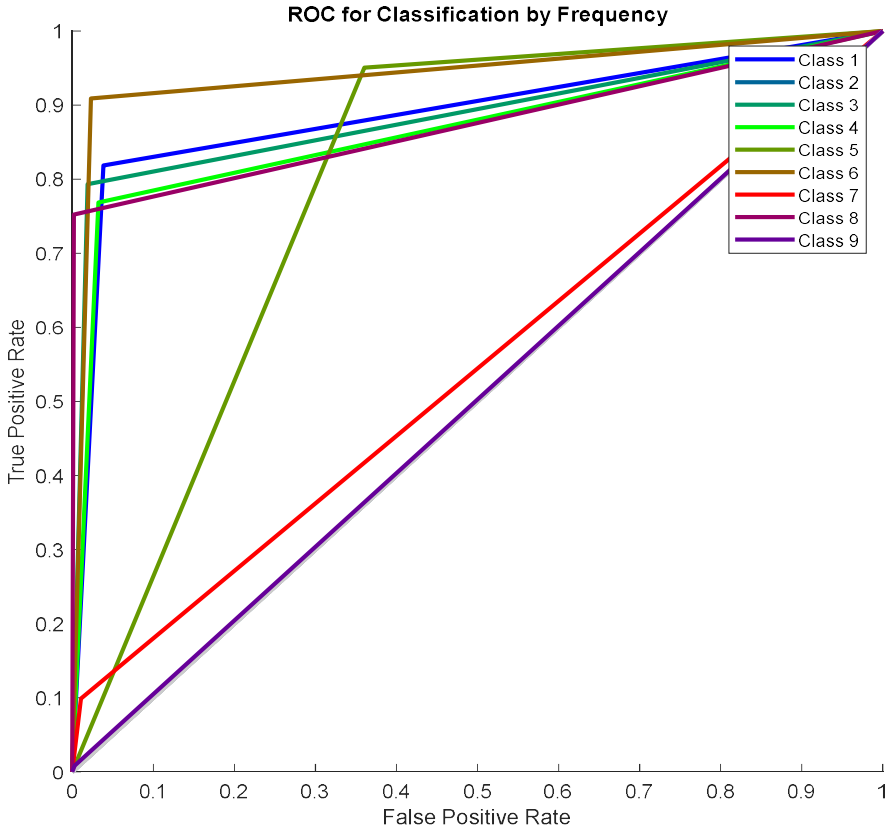

Figure 36. ROC for noisy LFM by 1D-CNN.

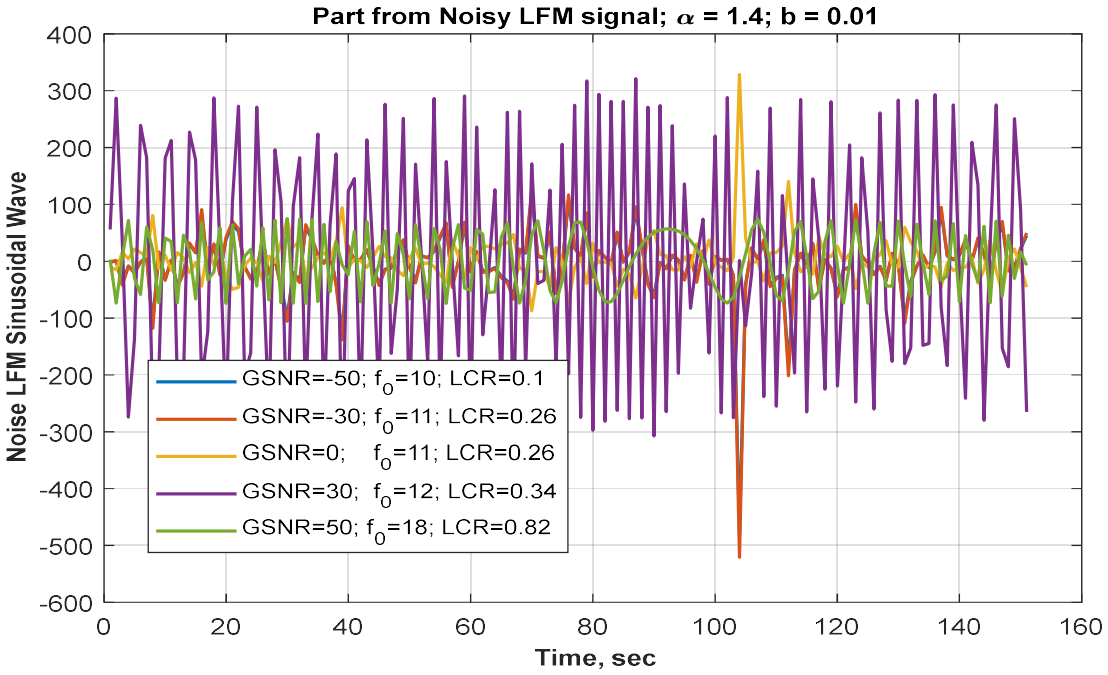

Figure 37. 1D-signals in time domain with different frequency, LCR, and GSNR.

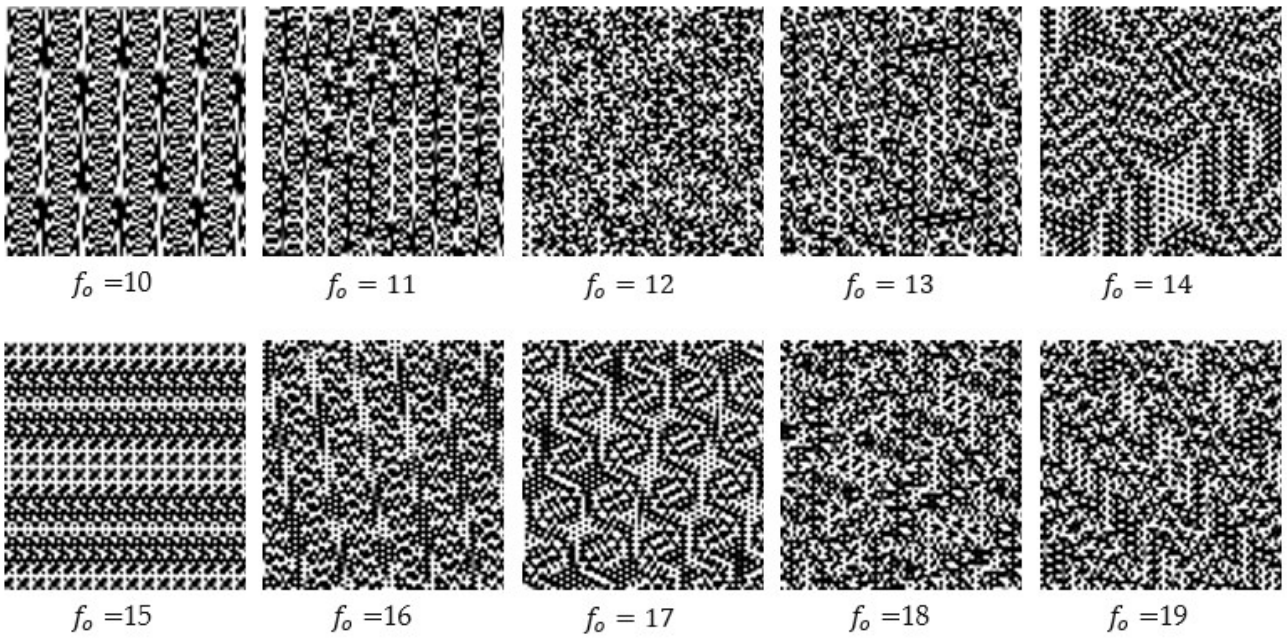

Figure 38. 2D-signals in the time domain.

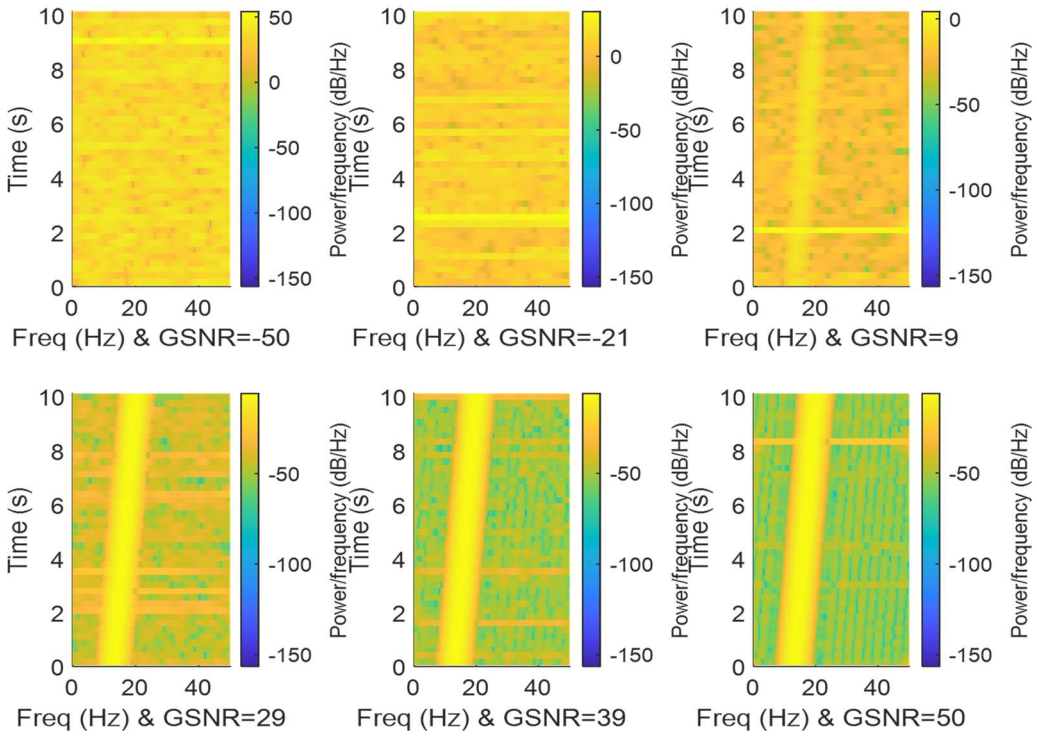

Figure 39. Signals in the time–frequency distribution.

Tables 1 and 2 show the differences between implementing a 1D-CNN and 2D-CNN. Table 3 shows the training parameters of the noisy LFM dataset. Because the classification accuracy in a one-dimensional convolutional network is very low, the guess of parameters is influenced. The lower error of the slope and frequency estimation refers to the higher accuracy of the classification and the rest of the measures. The measures are used to evaluate the efficiency of the CNN model. If these measurements have high values, we can conclude that the slope and frequency estimations are correct. ROC scale is used to test the effectiveness of the CNN model (as shown in Figures 34–36).

The results show that deep neural networks are better than time–frequency distribution for estimating the instantaneous frequency, and CNN is better than deep neural networks in estimating the instantaneous frequency of nonstationary signals. For time–frequency distribution, spectrogram and pspectrum functions have been used, where the results show that pspectrum is better than spectrogram for the IF estimate.

Further inspection of Figures 31 and 32 reveals that the performance of CNN (in terms of accuracy) can give acceptable results at very low signal-to-noise ratios where TFD fails, giving more than 20 dB difference in the GSNR working range as compared to the classical spectrogram-based estimation. Considering more advanced (though more complex) methods of LFM instantaneous frequency estimation such as Viterbi-based approach [30] and fractional Fourier transform-based approach [31], the performance of CNN gives more than 15 dB difference (please compare with Figure 3 in [30], knowing that reference [30] considers only Gaussian noise. In contrast, this work finds the much more damaging α-stable noise.

Future Directions: It should be noted that this research considers only mono-component linear FM signals. Future directions may consider multicomponent signals and non-linear FM, where more advanced time–frequency distributions would be used instead of the spectrogram of Equation (23) [32–34]. As we only considered mono-component LFM signals in this work, the spectrogram and Wigner–Ville distribution (WVD) are the widely used FE methods, where the spectrogram is chosen as per Equation (23) as it behaves better under noise than WVD due to the fact that it does not suffer from the cross-term effect [32,34]. In addition, as compressive sensing is a promising technology for sensors, wireless sensor networks (WSNs), and IoT applications [35], this research can be further enhanced by considering frequency estimation from compressed measurements [36].

8. Further Remarks

We will highlight some points in the proposed network prediction.

8.1. Comparing Network Training with and without Extracted Features

From Figures 31 and 32, we conclude that extracting the features of the noisy input signals significantly reduces the error rate and increases the system's accuracy. Thus, this affects the estimation of slope and frequency. A previously trained CNN model extracted the features. The CNN model was used to extract the features and classification as described in Section 5.4. The CNN model has two functions, which are feature extraction and classification. The parameters were estimated by extracting the features from the pre-trained CNN model and predicting parameters using simple DNN, which consists of one hidden layer containing ten nodes, ReLU activation function for the hidden layer, and softmax activation function for the output layer. Note the increasing number of iterations when DNN is trained without feature extraction.

8.2. Different Lengths for the Input Signal and Feature Vector

From Figures 33 and 34, we conclude that the input length also affects the neural network. If the input is a noisy signal with a large number of samples (size), it would be difficult for the network to work with high accuracy, especially if the network structure is simple. Then when reducing the length of such signals, the accuracy improves. Reducing the features has the opposite effect. That is the fewer features, the lower the accuracy. DNN consists of one hidden layer containing ten nodes, a ReLU activation function for the hidden layer, and a softmax activation function for the output layer. Note that decreasing the length of input signals leads to better accuracy and less time, but reducing the size of input features leads to less accuracy and less time.

8.3. Effect of Network Training by the Number of Layers and Number of Nodes

The number of hidden layers and the number of nodes per layer play significant roles in deciding the speed and accuracy of neural networks. Figures 35 and 36 illustrates the

effect of the network structure on the prediction results, where the network performance improves when a specific number increases the number of hidden layers. In contrast, an excessive increase may lead to negative results. DNN consists of three hidden layers containing (30, 25, and 20) nodes, ReLU activation function for hidden layers, and softmax activation function for the output layer.

9. Conclusions

This paper provided an overview of the performance of machine learning and deep learning approaches for estimating the frequency and slope of a noisy LFM signal. Under additive white Gaussian noise and symmetric α-stable noise, the simulation is a relevant signal (impulsive model). This work solves problems using traditional, machine, and deep learning approaches. It examines the frequency and slope estimation error under various GSNRs. In DNN, only two hidden layers are used. The convolution layer, ReLU activation function, max-pooling layers, dropout layer, fully connected layer, softmax layer, and classification layer are the 19 layers used in the CNN model. The simple structure designed for the DNN or CNN model works to reduce the communication system's complexity, power consumption, and cost. These characteristics are advantageous for systems with limited memory and computational processes, such as WSNs, which connect to the internet of things applications. The simulation results show that alpha is more harmful than beta, even if it has a small incapacity, and it significantly affects guess frequency and slope. CNN was used to estimate the parameters of LFM signals in this paper. Future research will focus on compressed and multicomponent non-linear FM signals.

Author Contributions: H.S.R. and Z.M.H. contributed equally to this work. H.S.R. contributed to the initial tasks of this work during her postgraduate preparatory year of coursework in 2019, where she obtained the top rank of high distinction before she started her research project on deep learning for frequency estimation under the supervision of Z.M.H. All authors have read and agreed to the published version of the manuscript.

Funding: This project was partially funded by Edith Cowan University via the ASPIRE Program.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: All types of data were generated using mathematical equations.

Acknowledgments: The authors would like to thank Edith Cowan University for partially supporting this project. Thanks are extended to the (anonymous) reviewers for their constructive comments that improved this paper.

Conflicts of Interest: The authors declare that no conflict of interest is associated with this work.

Appendix A

The procedures of SαS are as follows:

1. For β = 0: symmetric α-stable noise (Y) can be generated as follows. A uniformly distributed random variable V and an independent exponential random variable W are generated as follows:

$$V = \frac{\pi}{2} \cdot (2U - 1) \tag{A1}$$

$$W = -\log(H) \tag{A2}$$

where $U,\ H \in \mathcal{U}$, the standard uniform distribution. Then SαS for $\alpha \neq 1$ is obtained:

$$X = \frac{\sin(\alpha \cdot V)}{\{\cos(V)\}^{1/\alpha}} \cdot \left[\frac{\cos(V \cdot (1-\alpha))}{W}\right]^{(1-\alpha)/\alpha} \tag{A3a}$$

$$Y = \gamma X + \mu \tag{A3b}$$

Moreover, SαS for $\alpha = 1$ is obtained:

$$X = \tan V \tag{A3c}$$

$$Y = \gamma X + \mu. \tag{A3d}$$

2. For $\alpha \neq 1$, a uniformly distributed random variable V and an independent exponential random variable W are generated as follows to get X:

$$V = \pi \cdot (U - 0.5) \tag{A4}$$

$$W = -\log(H) \tag{A5}$$

$$X = S_{\alpha,\beta} \cdot \frac{\sin\{\alpha(V + B_{\alpha,\beta})\}}{\{\cos(V)\}^{1/\alpha}} \cdot \left[\frac{\cos\{V - \alpha(V + B_{\alpha,\beta})\}}{W}\right]^{(1-\alpha)/\alpha} \tag{A6}$$

$$S_{\alpha,\beta} = \left\{1 + \beta^2 \tan^2\left(\frac{\pi\alpha}{2}\right)\right\}^{1/(2\alpha)} \tag{A7}$$

$$B_{\alpha,\beta} = \frac{\arctan\left(\beta \tan \frac{\pi\alpha}{2}\right)}{\alpha}. \tag{A8}$$

When scale and shift are applied, we have:

$$Y = \gamma X + \mu. \tag{A9}$$

3. For $\alpha = 1$, random variables V and W are generated as above, then:

$$X = \frac{2}{\pi}\left\{\left(\frac{\pi}{2} + \beta V\right)\tan V - \beta \log\left(\frac{\frac{\pi}{2} W \cos V}{\frac{\pi}{2} + \beta V}\right)\right\} \tag{A10}$$

When scale and shift are applied as an equation, we have:

$$Y = \gamma X + \frac{2}{\pi}\beta\gamma \log\gamma + \mu. \tag{A11}$$

Appendix B

Table A1 shows the parameters description of CNN layers, where FS is filter size, NF is the number of filters (number of output feature maps), S is stride, P is padding, C is the number of channels (number of input feature maps), a total of parameters is number of weights plus number of biases; in batch normalization ε is epsilon, B is sifted, λ is scale, $T\rho$ is the total number of parameters in a convolutional layer, and bias b = 1:

$$T\rho = (FS \cdot FS \cdot C + b) \cdot NF \tag{A12}$$

Table A1. Topology of the proposed CNN model.

Indexes	Layers Name	Input Size	OutputSize	Hyperparameters	Total of Parameters
1.	Image Input	$80 \times 80 \times 1$	$80 \times 80 \times 1$	Normalization zero-center	0
2.	Convolution	$80 \times 80 \times 1$	$80 \times 80 \times 30$	$FS = 3 \times 3$, $NF = 30$, $S = 1$, $P = 1$, $C = 1$	300
3.	Batch Normalization	$80 \times 80 \times 30$	$80 \times 80 \times 30$	ε =0, B =0, λ =1	0
4.	ReLU	$80 \times 80 \times 30$	$80 \times 80 \times 30$	-	0
5.	Max Pooling	$80 \times 80 \times 30$	$40 \times 40 \times 30$	$FS = 2 \times 2$, $S = 2$	0

Table A1. *Cont.*

Indexes	Layers Name	Input Size	OutputSize	Hyperparameters	Total of Parameters
6.	Convolution	$40 \times 40 \times 30$	$40 \times 40 \times 60$	$FS = 3 \times 3$, $NF = 60, S = 1, P = 1$, C=30	16,260
7.	ReLU	$40 \times 40 \times 60$	$40 \times 40 \times 60$	-	0
8.	Max Pooling	$40 \times 40 \times 60$	$20 \times 20 \times 60$	$FS = 2 \times 2, S = 2$	0
9.	Convolution	$20 \times 20 \times 60$	$20 \times 20 \times 90$	$FS = 3 \times 3$, $NF = 90, S = 1, P = 1$	48,690
10.	ReLU	$20 \times 20 \times 90$	$20 \times 20 \times 90$	-	0
11.	Max Pooling	$20 \times 20 \times 90$	$10 \times 10 \times 90$	$FS = 2 \times 2, S = 1$	0
12.	Convolution	$10 \times 10 \times 90$	$10 \times 10 \times 128$	$FS = 3 \times 3$, $NF = 128, S = 1, P = 1$	103,808
13.	ReLU	$10 \times 10 \times 128$	$10 \times 10 \times 128$	-	0
14.	Max Pooling	$10 \times 10 \times 128$	$5 \times 5 \times 128$	$FS = 2 \times 2, S = 1$	0
15.	Fully Connected	$5 \times 5 \times 128 = 3200$	100	Nodes = 100	320,100
16.	Dropout	-	-	Probability = 0.5	0
17.	Fully Connected	100	10	No. class = 10	1010
18.	Softmax	10	10	-	0
19.	Classification	10	1	Loss Function = cross-entropy	0
	Number of weights for convolution layers = (270 + 16200 + 48600 + 103680) = 168750 Number of biases for convolution layers = (30 + 60 + 90 + 128) = 308 Total parameters for convolution layers = 169058				
	Total parameters for all network = 169058 + 320100+1010 = 490168				

Appendix C

Algorithm A1: Frequency Estimation by TFD

Input: The $\sin gle - \text{tone}$ and LFM signals $x(t)$.
Output: Instantaneous frequency estimate $\hat{f}$.
Begin:
Step 1: Initial parameters are initial frequency $f_o = 23$, $Ts = 0.01$,$fs = \frac{1}{Ts}$, $fs2 = \frac{fs}{2}$, $df = \frac{fs}{N}$, $f_1 = 0 \rightarrow df \rightarrow fs2$, and $N = 1024$.
Step 2: Geometric SNR vector (sr), $sr \in [-50\ 50]$, number of realizations $R = 50$, the duplicate input signal in R rows, may repeat run R times to get a good result.
Step 3: For $\boldsymbol{m = 1 \rightarrow |sr|}$

- The SαS noise (Y) and Gaussian noise (G) are generated as above explained.
- The input signals are corrupted by hybrid noise, where total noise (N_T) is a mixture of both Gaussian noise and SαS noise:

$$N_T = G + Y$$

$$y(t) = x(t) + N_T$$

- IF error temporary matrix, where $E =$ zeros (initially).

Step 4: For $\boldsymbol{h = 1 : R}$

■ Take h^{th} realization, where $z = y_h$.

■ Eliminate the negative portion of the signal spectrum by called Alg. (5) with an input parameter (z) and an output parameter (S_A).

■ IF estimate with spectrogram and pspectrum by called Alg. (6) with input parameter (S_A) and output parameter $(\text{spec}(t, f))$:

■ Estimate the IF from the peak (max) of the $spec(t,\ f)$ as follows:

$$\hat{f} = \max\{\text{spec}(t, f)\}$$

■ Calculate the relative squared error for each GSNR:

$$e = \left| \frac{\left(\hat{f} \times df - IF_t\right)}{f_o} \right|^2$$

$$E = E + e$$

where IF_t theoretical instantaneous frequency.

■ **End of *h* loop**

Step 6 : IF estimation error at GSNR dB, where $FE(m) = \frac{E}{R}$

End of *m* loop

End of algorithm

Algorithm A2: Hilbert Transform

Input : The signal $(z(t))$.

Output : Analytic signal $(S_A(t))$.

Begin:

Step 1 : Compute the FFT of the signal $z(t)$ to give spectrum $(Z(f))$.

Step 2 : Creates a vector h whose elements h_i have three values as follows:

$$h_i = \begin{cases} 1 & \text{if } i = 1 \text{ or } \left(\frac{n}{2}\right) + 1 \\ 2 & \text{if } i = 2, 3, \ldots, \left(\frac{n}{2}\right) \\ 0 & \text{if } i = \left(\frac{n}{2}\right) + 2, \ldots, n \end{cases}$$

where $n = |Z|$.

Step 3 : Find signal (ms), where $ms = Z \times h$.

Step 4 : Compute (S_A) by inverse FFT of vector ms, where $S_A = \text{IFFT}(ms)$

End of algorithm

Algorithm A3: Spectrogram of Short-Time Fourier Transform

Input: Analytic signal $S_A(t)$ and $N = |S_A(t)|$.

Output: Spectrogram $\text{spec}(t,\ f)$, time and frequency vectors $t,\ f$.

Begin:

Step 1: Define the parameters of STFT are window length (m), hop size (h), window overlap (q), sampling frequency (f_s), and the number of FFT points (nfft).

Step 2: Create Hamming window (win) of length $m = 32$, $q = \frac{m}{4}$, and $h = m - q$.

Step 3: STFT matrix (S) with size $[\text{NUP},\ L]$, where L is the number of signal frames.

$$\text{NUP} = \text{nfft}$$

$$\text{L} = \left\lceil \frac{N - q}{h} \right\rceil$$

Step 4: Divides the input signal $S_A(t)$ into overlapping segments and multiply each segment by the window (Sw). Then fast Fourier transform is applied to each segment (Sw) :

For $i = 0 \rightarrow L-1$

$$Sw = S_A(i*h+1 \rightarrow i*hop+m)*win$$

$$X = FT(Sw)$$

$$S(:, i+1) = X(1 \rightarrow NUP)$$

End of *i* loop

Step 5: Find the spectrogram $(\mathrm{spec}(t, f))$, were

$$\mathrm{spec}(t, f) = \frac{2}{\sum \mathrm{win}/\mathrm{N}} \times \mathrm{S}.$$

Step 6: Calculation of the time (t) vector in second and frequency (f) vector in Hz, were

$$t = \left(\frac{m}{2}, \frac{m}{2}+h, \ldots, \frac{m}{2}+(L-1)\times h\right)/f_s$$

$$f = (0, \ldots, \mathrm{NUP}-1) \times f_s/\mathrm{nfft}$$

End of algorithm

References

1. Boashash, B. Estimating and interpreting the instantaneous frequency of a signal—I: Fundamentals. *Proc. IEEE* **1992**, *80*, 520–538. [CrossRef]
2. Boashash, B. Estimating and interpreting the instantaneous frequency of a signal—II: Algorithms and applications. *Proc. IEEE* **1992**, *80*, 540–568. [CrossRef]
3. Liu, J.; Fan, L.; Jin, J.; Wang, X.; Xing, J.; He, W. An Accurate and Efficient Frequency Estimation Algorithm by Using FFT and DTFT. In Proceedings of the 39th Chinese Control Conference (CCC), Shenyang, China, 27–29 July 2020.
4. Akram, J.; Khan, N.A.; Ali, S.; Akram, A. Multi-component instantaneous frequency estimation using signal decomposition and time-frequency filtering. *Signal Image Video Process.* **2020**, *14*, 1663–1670. [CrossRef]
5. Xu, S.; Shimodaira, H. Direct F0 Estimation with Neural-Network-Based Regression. In Proceedings of the INTER-SPEECH, Graz, Austria, 15–19 September 2019; pp. 1995–1999.
6. Silva, B.; Habermann, D.; Medella, A.; Fraidenraich, G. Artificial Neural Networks to Solve Doppler Ambiguities in Pulsed Radars. In Proceedings of the International Conference on Radar (RADAR), Brisbane, Australia, 27–31 August 2018; pp. 1–5.
7. Chen, X.; Jiang, Q.; Su, N.; Chen, B.; Guan, J. LFM Signal Detection and Estimation Based on Deep Convolutional Neural Network. In Proceedings of the Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC), Lanzhou, China, 18–21 November 2019; pp. 753–758.
8. Xuelian, L.; Wang, C. A novel parameter estimation of chirp signal in α-stable noise. *IEICE Electronics Express* **2017**, *14*, 20161053.
9. Aboutanios, E.; Mulgrew, B. Iterative frequency estimation by interpolation on Fourier coefficients. *IEEE Trans. Signal Process.* **2005**, *53*, 1237–1241. [CrossRef]
10. Ahmet, S. Fast and efficient sinusoidal frequency estimation by using the DFT coefficients. *IEEE Trans. Commun.* **2018**, *67*, 2333–2342.
11. Li, L.; Qiu, T. A Robust Parameter Estimation of LFM Signal Based on Sigmoid Transform Under the Alpha Stable Distribution Noise. *Circuits Syst. Signal Process.* **2019**, *38*, 3170–3186. [CrossRef]
12. Almayyali, H.; Hussain, Z. Deep Learning versus Spectral Techniques for Frequency Estimation of Single Tones: Reduced Complexity for Software-Defined Radio and IoT Sensor Communications. *Sensors* **2021**, *21*, 2729. [CrossRef]
13. Boashash, B. (Ed.) *Time-Frequency Signal Analysis and Processing: A Comprehensive Reference*, 2nd ed.; Elsevier: Oxford, UK, 2016.
14. Milczarek, H.; Leśnik, C.; Djurović, I.; Kawalec, A. Estimating the Instantaneous Frequency of Linear and Nonlinear Frequency Modulated Radar Signals—A Comparative Study. *Sensors* **2021**, *21*, 2840. [CrossRef]
15. Boashash, B.; O'Shea, P.; Arnold, M.J. *Algorithms for Instantaneous Frequency Estimation: A Comparative Study*; SPIE: Bellingham, WA, USA, 1990.
16. Zhang, J.; Li, Y.; Yin, J. Modulation classification method for frequency modulation signals based on the time–frequency distribution and CNN. *IET Radar Sonar Navig.* **2018**, *12*, 244–249. [CrossRef]
17. Liu, M.; Han, Y.; Chen, Y.; Song, H.; Yang, Z.; Gong, F. Modulation Parameter Estimation of LFM Interference for Direct Sequence Spread Spectrum Communication System in Alpha-Stable Noise. *IEEE Syst. J.* **2020**, *15*, 881–892. [CrossRef]
18. Kristoffer, H. Symmetric Alpha-Stable Adapted Demodulation and Parameter Estimation. Master's Dissertation, Lulea University of Technology, Lulea, Sweden, 2018.
19. Braspenning, P.J.; Thuijsman, F.; Weijters, A.J.M.M. Artificial neural networks: An introduction to ANN theory and practice. In *Lecture Notes in Computer Science*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 1995; Volume 931.

20. Nwankpa, C.; Ijomah, W.; Gachagan, A.; Marshall, S. Activation functions: Comparison of trends in practice and research for deep learning. *arXiv* **2018**, arXiv:1811.03378.
21. Møller, M.F. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Netw.* **1993**, *6*, 525–533. [CrossRef]
22. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. *arXiv* **2014**, arXiv:1412.6980.
23. Yamashita, R.; Nishio, M.; Do, R.K.G.; Togashi, K. Convolutional neural networks: An overview and application in radiology. *Insights Imaging* **2018**, *9*, 611–629. [CrossRef] [PubMed]
24. Moons, B.; Bankman, D.; Verhelst, M. *Embedded Deep Learning: Algorithms, Architectures and Circuits for Always-on Neural Network Processing*; Springer: Berlin/Heidelberg, Germany, 2018.
25. Phil, K. *Matlab Deep Learning with Machine Learning, Neural Networks and Artificial Intelligence*; Apress: New York, NY, USA, 2017.
26. Powers, D.M. Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation. *arXiv* **2011**, arXiv:2010.16061.
27. Hussain, Z.M.; Sadik, A.Z.; O'Shea, P. *Digital Signal Processing: An Introduction with MATLAB and Applications*; Springer: Berlin, Germany, 2011.
28. Razzaq, H.S.; Hussain, Z.M. Instantaneous Frequency Estimation for Frequency-Modulated Signals under Gaussian and Symmetric α-Stable Noise. In Proceedings of the 31st International Telecommunication Networks and Applications Conference (ITNAC), Sydney, Australia, 24–26 November 2021.
29. Gao, J.; Deng, B.; Qin, Y.; Wang, H.; Li, X. Enhanced radar imaging using a complex-valued convolutional neural network. *IEEE Geosci. Remote Sens. Lett.* **2018**, *16*, 35–39. [CrossRef]
30. Djurović, I. Viterbi algorithm for chirp-rate and instantaneous frequency estimation. *Signal Process.* **2011**, *91*, 1308–1314. [CrossRef]
31. Yin, Z.; Chen, W. A New LFM-Signal Detector Based on Fractional Fourier Transform. *EURASIP J. Adv. Signal Process.* **2010**, *2010*, 1–7. [CrossRef]
32. Hussain, Z.M.; Boashash, B. Design of time-frequency distributions for amplitude and IF estimation of multicomponent signals. In Proceedings of the Sixth International Symposium on Signal Processing and its Applications (ISSPA2001), Kuala Lumpur, Malaysia, 13–16 August 2001.
33. Nelson, D.J. Instantaneous Higher Order Phase Derivatives. *Digit. Signal Process.* **2002**, *12*, 416–428. [CrossRef]
34. Yin, Q.; Shen, L.; Lu, M.; Wang, X.; Liu, Z. Selection of optimal window length using STFT for quantitative SNR analysis of LFM signal. *J. Syst. Eng. Electron.* **2013**, *24*, 26–35. [CrossRef]
35. Hussain, Z.M. Energy-Efficient Systems for Smart Sensor Communications. In Proceedings of the IEEE 30th International Telecommunication Networks and Applications Conference (ITNAC), Melbourne, Australia, 25–27 November 2020.
36. Alwan, N.; Hussain, Z. Frequency Estimation from Compressed Measurements of a Sinusoid in Moving-Average Colored Noise. *Electronics* **2021**, *10*, 1852. [CrossRef]