Hindawi
Security and Communication Networks
Volume 2020, Article ID 2151570, 9 pages
https://doi.org/10.1155/2020/2151570

WILEY | Hindawi

Research Article

Jamming Prediction for Radar Signals Using Machine Learning Methods

Gyeong-Hoon Lee,[1] **Jeil Jo**,[2] **and Cheong Hee Park**[1]

[1]*Department of Computer Science and Engineering, Chungnam National University, Daejeon, Republic of Korea*
[2]*The 2nd Research and Development Institute, Agency for Defense Development, Daejeon, Republic of Korea*

Correspondence should be addressed to Cheong Hee Park; cheonghee@cnu.ac.kr

Received 7 May 2019; Revised 22 December 2019; Accepted 30 December 2019; Published 24 January 2020

Academic Editor: Roberto Di Pietro

Jamming is a form of electronic warfare where jammers radiate interfering signals toward an enemy radar, disrupting the receiver. The conventional method for determining an effective jamming technique corresponding to a threat signal is based on the library which stores the appropriate jamming method for signal types. However, there is a limit to the use of a library when a threat signal of a new type or a threat signal that has been altered differently from existing types is received. In this paper, we study two methods of predicting the appropriate jamming technique for a received threat signal using deep learning: using a deep neural network on feature values extracted manually from the PDW list and using long short-term memory (LSTM) which takes the PDW list as input. Using training data consisting of pairs of threat signals and corresponding jamming techniques, a deep learning model is trained which outputs jamming techniques for threat signal inputs. Training data are constructed based on the information in the library, but the trained deep learning model is used to predict jamming techniques for received threat signals without using the library. The prediction performance and time complexity of two proposed methods are compared. In particular, the ability to predict jamming techniques for unknown types of radar signals which are not used in the stage of training the model is analyzed.

1. Introduction

Electronic warfare is a military activity that uses the electromagnetic spectrum to attack an enemy or impede enemy assaults, and jamming is a form of electronic warfare where jammers radiate interference signals toward an enemy radar in order to disrupt the receiver. Jamming techniques can be categorized largely to noise jamming, range deception jamming, velocity deception jamming, and angle deception jamming [1]. Since the effect of jamming varies depending on the characteristics of the received radar signal, a jamming technique that is effective for the threat signal has to be applied.

When a threat signal is received, the conventional method to determine a jamming technique is based on information in the library which stores the appropriate jamming method for signal types. While a jamming method can be selected and applied easily when the type and value of a received threat signal exist in the library, there is a limit to the use of a library when a threat signal of a new type or a threat signal that has been altered differently from existing types is received. Hence, it is necessary to apply a machine learning model that can predict the appropriate jamming technique by learning the received radar signal characteristics.

An analog radar signal is converted to a digital signal and stored as a pulse description word (PDW) list where various feature values including the pulse width (PW), pulse repetition interval (PRI), and radio frequency (RF) are recorded according to the arrival time of pulses [2]. In particular, since PRI and RF are modulated with time and a radar signal has different modulation types, it is necessary to grasp sequential characteristics in PDW. In recent years, deep learning has been used effectively for sequential data analysis.

In this paper, we study two methods of predicting a jamming method corresponding to a threat signal using

deep learning. Firstly, a deep neural network can be modeled on feature values extracted manually from the PDW list. In this approach, the feature extraction process and the learning process of a neural network model are performed sequentially. Hence, an effective feature extraction method that will enhance the performance of the model has to be selected appropriately. Secondly, instead of extracting the feature values, we can learn long short-term memory (LSTM) [3] which uses the PDW list itself as input. Despite the outstanding performance of LSTM in dealing with the short- and long-term dependence of the sequence data, no studies using LSTM has been conducted to predict how to jam the threat signal as far as we know.

The prediction performance and time complexity of two approaches are compared. Various feature values including autocorrelation coefficients [4] and mel frequency cepstral coefficients (MFCCs) [5] are extracted from the radar signal, and a deep neural networks of different structures are tested. Also, the process of determining values of model parameters in LSTM is discussed. In particular, the ability to predict jamming methods for unknown types of radar signals which are not used to construct the model is analyzed.

The paper is organized as follows: in Section 2, the research related to radar signal analysis is discussed, and in Section 3, the radar signal data and testing setup are explained. In Section 4, the process of determining model parameters in LSTM is discussed. Section 5 explains the jamming prediction model based on feature value extraction and a deep neural network. In Section 6, we compare the prediction performance and training time in two approaches. Conclusions are made in Section 7.

2. Related Work

Generally, the methods of extracting feature values from sequence data can be divided into time-domain methods and frequency-domain methods [6]. Time-domain methods extract features such as various statistical values and autocorrelation coefficients. Studies in [7–9] have used statistical values including the root mean square, square root of the amplitude, kurtosis value, skewness value, peak-peak value, crest factor, impact factor, margin factor, shape factor, and kurtosis factor. In [4], for fuzzy clustering of time series, autocorrelation coefficients of different lags were used as feature values. In the frequency domain, fast Fourier transform, spectrum analysis, or wavelet transform are often used for pattern analysis over time. MFCC is a common and efficient technique for signal processing which is used in various domains such as speech recognition, speaker identification, and hand gesture recognition [5, 10].

Radar signal analysis uses various feature extraction methods in time and frequency domains in order to obtain the sequential characteristics with respect to time. A study in [11] analyzed deception jamming and target echo using the features of amplitude undulation, high-order cumulant, and bispectrum. Another study compared classification performance of frequency-modulated (FM) signals from additive white Gaussian noise in two cases: when using the autocorrelation coefficients in the time domain and when using the power spectrum in the frequency domain as the input of a neural network (NN) [12]. It showed that at a high signal-to-noise ratio (SNR), the NN classifier performs well for either case, but at very low SNR, it performs better when its input is the autocorrelation of the signal.

Machine learning techniques have been used for radar signal classification. In [13], a 64-dimensional vector for a signal was constructed based on the second difference of the pulse TOAs (time of arrival) and a 3-layer multilayer perceptron (MLP) was trained to classify radar signals to four PRI modulation types. There was also a study that classified radar signals according to the PRI modulation types by defining a set of five features based on the histogram of pulse intervals and the second difference of TOAs and training a MLP with one hidden layer [14]. The work in [15] investigated the recognition of generic radar data signal train pulse sources using three classification models, neural networks, support vector machines (SVM), and random forests (RF) which were combined with three missing feature imputation techniques such as multiple imputation, k-nearest neighbour imputation, and bagged tree imputation. It showed overall improved accuracy when the classifier is trained on the bagged tree imputed values. In [16], the restricted Boltzmann machine was used to extract the feature parameters and recognize the radar emitter. A bottom-up hierarchical unsupervised learning was used to obtain the initial parameters, and the traditional BP algorithm was conducted to fine tune the network parameters.

Research was also carried out on jamming signal classification using deep learning. A study in [17] used a 13-layer artificial neural network composed of a 3-layer convolutional neural network (CNN), 2 pooling layers, 4 fully connected layers, and 4 batch normalization layers for the classification of jamming signals of five types. CNN has the advantage of being able to automatically extract features and locally use the sequential information of a signal. However, it is difficult to determine the long-term time dependence of the signal because only local information can be obtained. In [18], the jammer classification problem in the Global Navigation Satellite System (GNSS) bands was treated as a black-and-white image classification problem based on a time-frequency analysis and image mapping of a jammed signal. GNSS signals interfered with one of the five jammer types were saved as a black-and-white image. SVM and CNN algorithms were applied for classification of six classes including one class where the jammer was absent.

Deep learning has also been used for various tasks in radar signal processing. In [19], recurrent neural network (RNN) model with gated recurrent unit (GRU) was applied to remove interference and reconstruct transmit signal. The authors of [20] argued that deep learning offers a unifying framework to integrate sensing, processing, and decision-making and applied their approach to the autofocus problem in the synthetic aperture radar (SAR) imaging.

3. Radar Signal Data

In this section, we explain the representation of radar signals and experimental setting including data construction.

3.1. Radar Signals. A radar signal can be represented as a sequence of pulses as shown in Figure 1. For each pulse, 5 characteristic values, p_k for PW, i_k for PRI, f_k for RF, a_k for AOA (angle of attack), and m_k for AMP (amplitude), are obtained, and a radar signal can be stored as a PDW list:

$$[X_1 = (p_1, i_1, f_1, a_1, m_1), \ldots, X_n = (p_n, i_n, f_n, a_n, m_n)]. \quad (1)$$

Generally, radar signals have modulations in the RF and PRI in order to evade to be detected. RF modulation refers to changes in a sequence $[f_1, \ldots, f_n]$ of pulse frequency values. There are 6 types of RF modulation including fix, agile, hopping, sine, −sawtooth, and +sawtooth as shown in Figure 2(a). PRI modulation refers to changes in a sequence $[i_1, \ldots, i_n]$ of pulse repetition interval values, and there are 7 types of PRI modulation including stable, jitter, stagger, dwell and switch, sine, −sawtooth, and +sawtooth as shown in Figure 2(b).

3.2. Experimental Setting. With reference to the specifications of the currently operating radar models, 2,258 signal types were configured with different types and magnitudes of RF and PRI. Each type had a minimum of 1,000 and maximum of 9,000 PDW lists, and a total of 6,870,000 PDW lists were composed. A total of eight jamming techniques corresponding to 2,258 signal types were established by assigning one jamming technique to each signal type based on the information of the library. The goal of a deep learning model is set to predict one jamming technique among eight jamming techniques for threat signal input. Hence, the output layer in a deep learning model consists of eight nodes, and each jamming technique is encoded as an 8-dimensional binary vector by the one-hot-encoding method, where only one component is 1 and the others are 0. In the training and testing stages of a deep learning model, the values in the output layer which are forwarded from a previous layer are translated as probability values for each jamming technique by the softmax function.

A deep learning model is trained with training data which is composed of the pairs of radar signals and corresponding jamming techniques. In the testing stage, one among eight jamming techniques is predicted as an output of the constructed model when a radar signal is given as an input. To evaluate the prediction performance of a deep learning model for an unknown radar signal type that is not used in the training stage as well as for a known signal type, we randomly divided 2,258 signal types into the set A of 2,035 types to be used for the training and testing and the set B of 223 types to be used for testing of the unknown types. The ratio of the two divisions was approximately 9 : 1. For signal types in A, dividing PDW lists belonging to each type into 3 groups with a ratio of 7 : 1 : 2, three data sets for training, validation, and testing of known types were composed. The training data were used to train the model, and the validation data were used to select the model parameters and determine the final model. Validation data are needed to prevent the trained model from overfitting with training data. Dropout [21] is also tested, which is a method for preventing overfitting by probably dropping nodes of hidden layers during learning. The test data are not used for training, and instead they are used to evaluate the performance of the final model. Lastly, the performance of the final model is assessed using the PDW lists of the unknown radar signal type in the set B. Figure 3 shows the configuration of the data sets.

Based on the configuration of the data sets shown in Figure 3, a flowchart to show the overall organization in the remaining sections is given in Figure 4 for easy readability. Based on train and validation data of signal types of set A, the construction of a LSTM model is explained in Section 4 and the construction of feature extraction and a deep neural network is described in Section 5. Test data of set A which represents the known signal types and test data of set B which indicates the unknown signal types are used for the performance comparison of two models in Section 6.

4. Construction of a Jamming Prediction Model Based on LSTM

In this section, the construction of a LSTM model which takes a PDW list of a threat signal as input is explained.

4.1. Model Parameters in LSTM. Preliminary experiment was carried out using the small scale of radar signal data to select the LSTM model structure and parameter values [22]. The data used in the preliminary test had 1,157,500 PDW lists for a total of 640 radar signal types and 4 types of jamming methods. The basic model was composed of an input layer with 5 nodes for the feature values of AOA, AMP, RF, PRI, and PW, a LSTM layer consisting of 200 nodes, and the output layer consisting of 4 nodes. A process of determining an optimization method, minibatch size, dropout ratio, the number of the LSTM layers and fully connected layers, and decay ratio of learning rate was performed. Table 1 shows the parameters and their values which were tested. Since there were so many combinations of the parameter values, we have adopted a strategy that determines the parameter values sequentially. When testing optimization methods in step 1, all other parameters were set to the underlined values in the third column of Table 1. After the best optimization method is selected, it is fixed in the next steps where the remaining parameter values are determined. This process is performed in sequence up to step 7 in Table 1. The fourth column of Table 1 shows the selected parameter values.

4.2. LSTM Model. With reference to the model structure configured in Table 1, the final LSTM model was composed of an input layer with 3 nodes for RF, PRI, and PW, 2 LSTM layers with 200 nodes, one fully connected layer with 400 nodes, and an output layer. Since one among the eight jamming techniques needs to be predicted for each input threat signal, the output layer has 8 nodes. Through additional tests, it was determined to apply the batch normalization in the fully connected layer and gradient clipping for

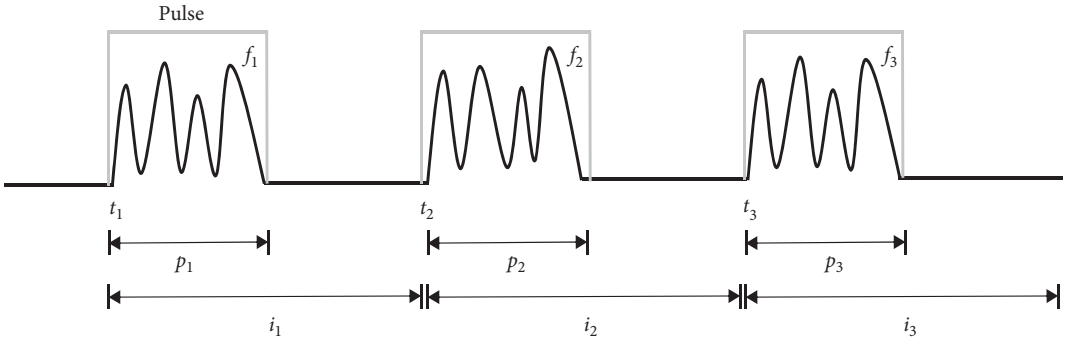

Figure 1: A radar signal represented as a sequence of pulses.

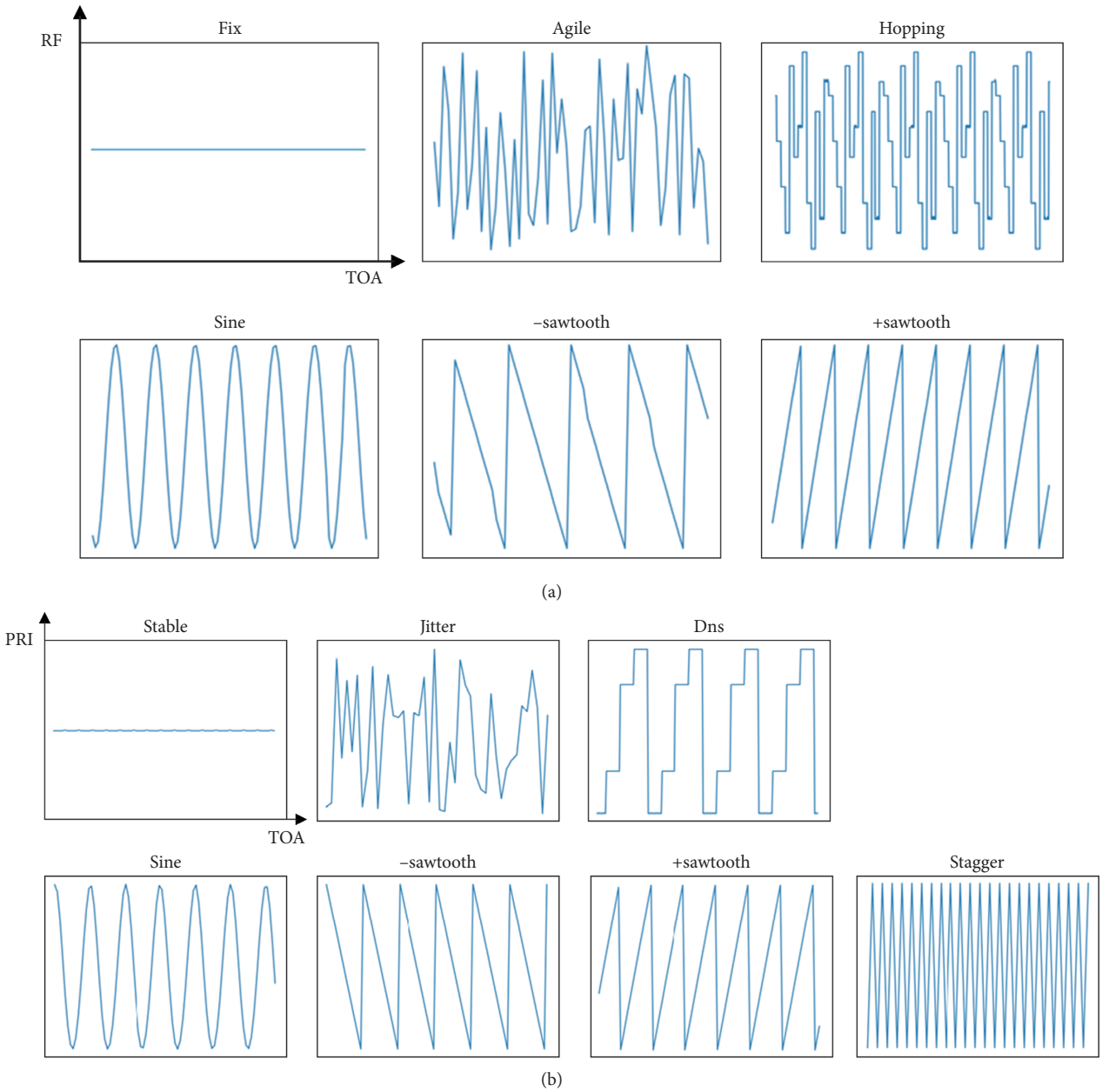

Figure 2: Various types of RF and PRI. (a) 6 types of RF. (b) 7 types of PRI.

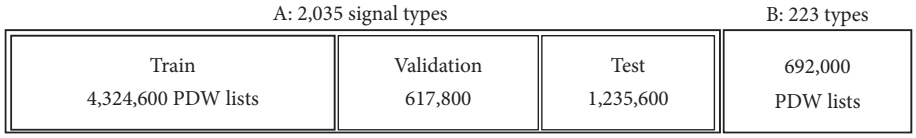

Figure 3: The configuration of the data sets.

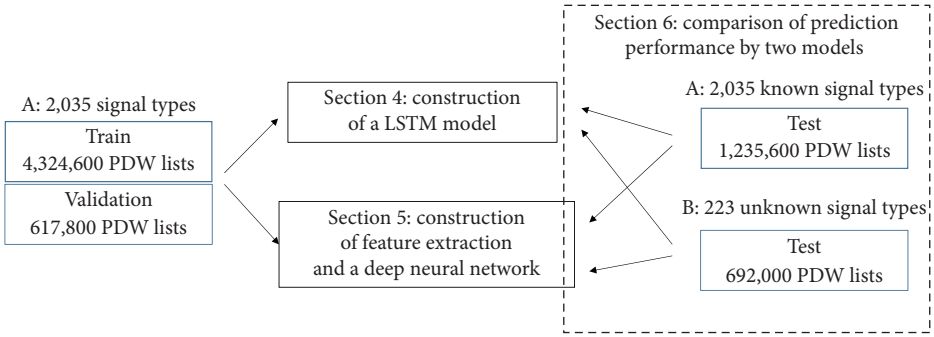

Figure 4: A flowchart to explain the overall organization.

Table 1: Parameters of a LSTM model.

Step	Parameters	Tested values	Selected value	Note
1	Optimization method	RMSProp, Adam, Adadelta	Adam	Initial learning rate was set as 0.003 for RMSProp [23], Adam [24], and 1.0 for Adadelta [25]
2	Minibatch size	50, 100, 200	200	
3	Dropout ratio (%)	0, 10, 30, 50	0	Dropout ratio means the rate at which the output gate units in a LSTM layer are randomly removed
4	LSTM layers	1, 2	2	The model with 2 LSTM layers showed higher accuracy than the model of 1 LSTM layer
5	Fully connected layer	0, 1	1	The model with the fully connected layer had a higher accuracy than the model with no fully connected layer
6	Input features	3, 5	3	Higher accuracy was obtained when using three features, RF, PRI, and PW, instead of using 5 features, AOA, AMP, RF, PRI, and PW
7	Decay ratio	Use, no use	Use	When gradually decreasing the learning rate by multiplying 0.9 to the previous learning rate per epoch after epoch 10, higher accuracy was obtained

stable learning. Figure 5 shows the structure of the final LSTM model.

5. Construction of Jamming Prediction Model Based on Feature Extraction and a Deep Neural Network

In this section, feature extraction from a PDW list of a threat signal and the construction of a deep neural network are explained in detail.

5.1. Feature Extraction. From sequence data expressed as $x_1, \ldots, x_n$, features such as statistical values, autocorrelation coefficients, and MFCCs were extracted.

Four statistic values of the mean $\overline{x}$, standard deviation s, skewness w, and kurtosis k were computed from the sequence data such as

$$\begin{aligned} \overline{x} &= \frac{1}{n}\sum_{1}^{n} x_t, \\ s &= \sqrt{\frac{1}{n}\sum_{1}^{n}\left(x_t - \overline{x}\right)^2}, \\ w &= \frac{\left(1/n\sum_{1}^{n}\left(x_t - \overline{x}\right)^3\right)}{\left(1/n\sum_{1}^{n}\left(x_t - \overline{x}\right)^2\right)^{3/2}}, \\ k &= \frac{\left(1/n\sum_{1}^{n}\left(x_t - \overline{x}\right)^4\right)}{\left(1/n\sum_{1}^{n}\left(x_t - \overline{x}\right)^2\right)^{2}}. \end{aligned} \tag{2}$$

The autocorrelation coefficient $R(\tau)$ represents the linear relevance between two subsequences with a lag τ, which is computed by

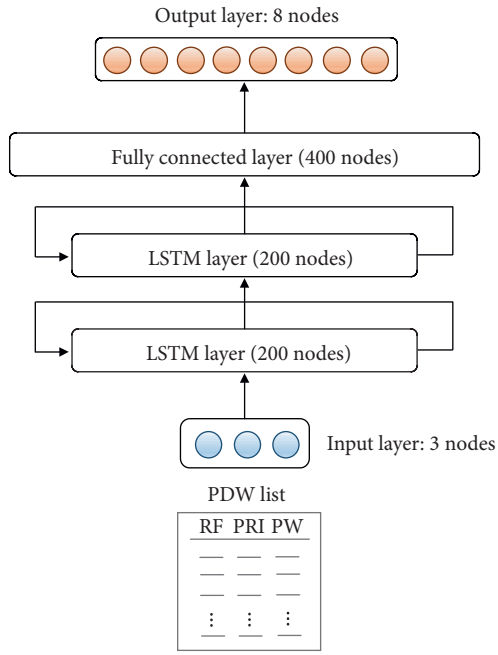

Figure 5: Structures of the final LSTM model.

$$R(\tau) = \frac{\sum_1^{n-\tau} (x_t - \overline{x})(x_{t+\tau} - \overline{x})}{\sum_1^n (x_t - \overline{x})^2}. \quad (3)$$

Various autocorrelation coefficients can be obtained by changing the time difference τ [4].

MFCC extracts feature values from the sequence data through the process of framing, Fast Fourier transform (FFT), mel filter bank, log function, and inverse FFT [5]. In the framing step, the sequence is divided into segments of equal size. FFT is applied to a subsequence of each segment to convert to the frequency domain, and then the power spectrum is obtained. The mel spectrum is obtained by applying mel filters to the power spectrum, and the coefficients are obtained by inverse FFT after applying the log function to the mel spectrum. These coefficients are called MFCCs. All or some of the coefficients obtained in each segment can be used as the feature values [5].

5.2. Composition of Feature Sets. We extracted feature values using RF, PRI, and PW sequences of a PDW list. By computing the mean, standard deviation, skewness, and kurtosis from each sequence, a total of 12 feature values were obtained from a PDW list. By increasing the time difference τ from 1 to 10 by incrementing 1, we extracted 10 autocorrelation coefficients from the RF and PRI sequences, and therefore, a total of 20 feature values were extracted from a PDW list. Also 10 MFCCs were computed for the RF and PRI sequences so that a total of 20 values were extracted. The autocorrelation coefficients and MFCCs were computed using Python libraries such as the function *acf* of statsmodels and the function *mfcc* of librosa [26].

Using statistical features as the basic features and combining the autocorrelation coefficients and MFCCs with basic features, we composed 4 feature sets as shown in Table 2 and tested prediction performance of a neural network model on each feature set. In the feature set F1, a radar signal is represented with only statistical features. The feature set F2 contains the statistical features and autocorrelation coefficients, while F3 is composed of the statistical features and MFCCs. The feature set F4 contains all of the statistical features, autocorrelation coefficients, and MFCCs, resulting in a total of 52 feature values.

5.3. Performance of Neural Network Models on Various Feature Sets. To select a neural network structure, comparative testing for 4 different models was performed varying the activation function and the number of layers, as shown in Table 3. Considering test results in Section 4.1, Adam was used as the optimization method and the minibatch size was set to 200.

The 4 feature sets in Table 2 and the 4 models in Table 3 were used to compare the performance. The total number of training epochs was set to 15, and the training data composed in Section 3.2 were used to train the models. The accuracy of the validation data was evaluated in each epoch. Table 4 shows the training and validation accuracies for 16 combinations of 4 feature sets and 4 NN models, which were evaluated in the epoch with the highest accuracy for the validation data. It revealed that model 4 on feature set F3 was found to have the highest performance. Hence, model 4 on feature set F3 was chosen as the final model. Figure 6 describes the structure of the final deep neural network model.

6. Performance Comparison of LSTM Model and Deep Neural Network Model Using Extracted Features

In this section, we compare the prediction performance and training time complexity of two deep learning models for test data of the known signal types in set A and test data of the unknown signal type in set B. As explained in Section 3.2, 2,258 signal types were randomly divided into the set A to be used for the training, validation, and testing and the set B to be used for testing of the unknown signal types. Now, we repeat the random division 10 times and measure the average accuracy for performance comparison of two approaches: using LSTM and using a deep neural network with extracted features (denoted as DNN/EF).

Table 5 shows test accuracy of known signal types in A and unknown types in B from 10 repeated experiments. In testing of known signal types, the average accuracy of 98.46% was obtained for the NN with extracted features, and the average accuracy of 99.36% was obtained for the LSTM model. The average accuracy for the unknown types when using the neural network with extracted features was 92.45%, while the average accuracy for the LSTM model was 93.53%. In both cases, the LSTM model showed a little higher accuracy.

The paired t-test was carried out in order to determine whether the higher average accuracy of the LSTM model is statistically significant. The paired t-test for the accuracy of known types gave the p value 0.0000612, which implies

Table 2: Composition of four feature sets.

Feature sets	Statistical features	Autocorrelation coef.	MFCC	Total
F1	12	—	—	12
F2	12	20	—	32
F3	12	—	20	32
F4	12	20	20	52

Table 3: Four deep neural network models.

Parameters	Model 1	Model 2	Model 3	Model 4
Activation function	tanh	relu	tanh	relu
Hidden layers	2		4	
Hidden units	400			
Learning rate	Initially set as 0.0005			
Learning rate decay	Multiplying the previous learning rate by 0.97 for every 1,000,000 samples			

Table 4: The performance in 16 combinations of feature sets and models.

Model	Feature set	Training accuracy (%)	Validation accuracy (%)
Model 1	F1	95.72	95.92
	F2	96.77	97.09
	F3	97.35	97.78
	F4	97.29	97.78
Model 2	F1	95.67	95.72
	F2	96.81	97.30
	F3	97.25	97.55
	F4	97.19	97.61
Model 3	F1	96.89	97.35
	F2	97.21	97.65
	F3	97.42	98.04
	F4	97.30	97.78
Model 4	F1	96.84	97.48
	F2	97.80	98.19
	F3	97.82	98.47
	F4	97.77	98.02

statistical significance in the difference between the accuracies of the two models in the significance level of 0.01. However, for the unknown type accuracy, the p value by the paired t-test was approximately 0.215, and so there was no significant difference in the significance level of 0.01. This result was thought to be due to the very large deviation in the accuracy of the LSTM model for the unknown type, which was a minimum of 89.69% and maximum of 97.20%.

In order to compare the learning time of the two methods, the average training time per epoch was measured. The specification of the computer used to measure the time consumed was CPU Intel i7-7700, 3.60 GHz and RAM 32.0 GB, and GPU NVIDIA GeForce GTX 1080 Ti. Table 6 compares the execution time for model training for 15 epochs. The process of extracting the features from the PDW lists was conducted once before the training of the deep neural network model. As shown in Table 6, the time consumed for training the LSTM model for 15 epochs was approximately 1.42 times longer than that of the deep neural network.

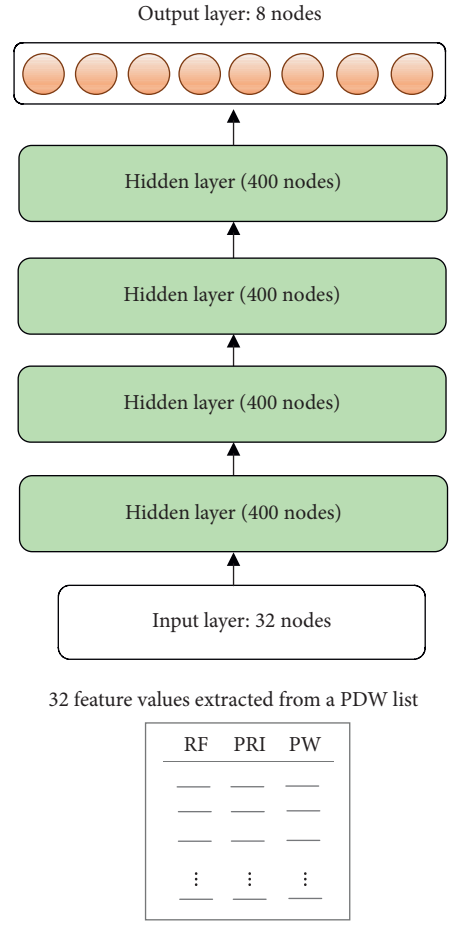

Figure 6: Structure of the final deep neural network.

Table 5: Test accuracy (%) of known types and unknown types.

	DNN/EF		LSTM model	
	Known types	Unknown types	Known types	Unknown types
1	98.49	93.33	99.72	97.20
2	98.35	92.47	99.70	95.09
3	98.49	90.57	98.78	90.76
4	98.60	92.81	99.06	90.94
5	98.61	92.72	99.32	93.42
6	98.55	94.31	99.60	95.06
7	98.39	92.62	99.71	96.73
8	98.47	92.11	99.01	89.69
9	98.31	92.19	99.00	90.64
10	98.33	91.32	99.65	95.76
Avg.	98.46	92.45	99.36	93.53

Table 6: Comparison of execution time in seconds.

Model	Feature extraction	Training for 1 epoch	Training for 15 epochs
DNN	73,742.33	19,329.43	363,683.78
LSTM	—	34,543.99	518,159.85

7. Conclusion

In this study, we applied deep learning for jamming prediction for a radar signal. Two methods were compared: using a deep neural network model based on the features extracted from radar signals and using LSTM model without the feature extraction process.

The deep neural network requires feature extraction in advance, and this is conducted once before the training of the deep neural network model. The training speed of the model per epoch was faster than that of the LSTM model. However, the feature extraction method has to be selected appropriately because the model prediction performance varies depending on the feature values extracted from the PDW list. The LSTM model has the advantages of being able to perform training and prediction by directly using the PDW list of the radar signal without the extraction of feature values. The prediction accuracy of the LSTM model was higher on average than that of the deep neural network model. However, there is the disadvantage that the training time takes longer than a deep neural network model trained on the extracted features.

Testing results demonstrate that the jamming method can be predicted for an unknown type of radar signal with an average accuracy of approximately 92% and higher. It shows that deep learning methods can be used effectively for predicting an appropriate jamming technique for new radar signals which are not used in model training.

Data Availability

The radar signal data used to support the findings of this study have not been made available because of the policy of Agency for Defense Development, Korea, that data distribution is limited.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

[1] M. S. R. Lothes, M. B. Szymanski, and R. G. Wiley, *Radar Vulnerability to Jamming*, Artech House, Boston, MA, USA, 1990.

[2] V. Iglesias, J. Grajal, O. Yeste-Ojeda, M. Garrido, M. Sánchez, and M. López-Vallejo, "Real-time radar pulse parameter extractor," in *Proceedings of the IEEE Radar Conference*, Cincinnati, OH, USA, May 2014.

[3] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[4] P. D'Urso and E. Maharaj, "Autocorrelation-based fuzzy clustering of time series," *Fuzzy Sets and Systems*, vol. 160, no. 24, pp. 3565–3589, 2009.

[5] S. Gupta, J. Jaafar, W. F. wan Ahmad, and A. Bansal, "Feature extraction using MFCC," *Signal & Image Processing: An International Journal*, vol. 4, no. 4, pp. 101–108, 2013.

[6] R. Shumway and D. Stoffer, *Time Series Analysis and its Application with R Examples*, Springer, Berlin, Germany, 2000.

[7] Z. Xia, S. Xia, L. Wan, and S. Cai, "Spectral regression based fault feature extraction for bearing accelerometer sensor signals," *Sensors*, vol. 12, no. 10, pp. 13694–13719, 2012.

[8] T. W. Rauber, F. de Assis Boldt, and F. M. Varejao, "Heterogeneous feature models and feature selection applied to bearing fault diagnosis," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 1, pp. 637–646, 2015.

[9] B. R. Nayana and P. Geethanjali, "Analysis of statistical time-domain features effectiveness in identification of bearing faults from vibration signal," *IEEE Sensors Journal*, vol. 17, no. 17, pp. 5618–5625, 2017.

[10] M. B. L. Muda and I. Elamvazuthi, "Voice recognition algorithm using mel frequency cepstral coefficient (MFCC) and dynamic time warping (DTW) techniques," *Journal of Computing*, vol. 2, no. 3, pp. 138–143, 2010.

[11] S. Q. L. Jian-xun and Y. Hai, "Signal feature analysis and experimental verification of radar deception jamming," in *Proceedings of the IEEE CIE International Conference on Radar*, Chengdu, China, October 2011.

[12] A. S. A. Mendoza and B. Flores, "Classification of radar jammer FM signals using a neural network," in *Proceedings of the SPIE Radar Sensor Technology XXI*, vol. 10188, May 2017.

[13] G. Noone, "A neural approach to automatic pulse repetition interval modulation recognition," in *Proceedings of the 1999 Information, Decision and Control. Data and Information Fusion Symposium, Signal Processing and Communications Symposium and Decision and Control Symposium. Proceedings (Cat. No.99EX251)*, Adelaide, Australia, February 1999.

[14] J. Kauppi and K. Martikainen, "An efficient set of features for pulse repetition interval modulation recognition," in *Proceedings of the IET International Conference on Radar Systems*, Edinburgh, UK, October 2007.

[15] I. Jordanov, N. Petrov, and A. Petrozziello, "Supervised radar signal classification," in *Proceedings of the International Joint Conference on Neural Networks*, Vancouver, Canada, July 2016.

[16] D. Zhou, X. Wang, Y. Tian, and R. Wang, "A novel radar signal recognition method based on a deep restricted Boltzmann machine," *Engineering Review*, vol. 37, pp. 165–171, 2017.

[17] Z. Y. Z. Wu, Y. Zhao, and H. Luo, "Jamming signals classification using convolutional neural network," in *Proceedings of the IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, Bilbao, Spain, December 2017.

[18] R. Ferre, A. Fuente, and E. Lohan, "Jammer classification in GNSS bands via machine learning algorithms," *Sensors*, vol. 19, no. 22, p. 4841, 2019.

[19] J. Mun, H. Kim, and J. Lee, "A deep learning approach for automotive radar interference mitigation," 2019, https://arxiv.org/abs/1903.06380.

[20] E. Masen, B. Yonei, and B. Yazici, "Deep learning for radar," in *Proceedings of the 2017 IEEE Radar Conference (RadarConf)*, Seattle, WA, USA, May 2017.

[21] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[22] G. Lee, "Radar jamming technique prediction using deep learning," Master thesis, Chungnam National University, Daejeon, Korea, 2019.

[23] M. Zeiler, "ADADELTA: an adaptive learning rate method," 2012, https://arxiv.org/abs/1212.5701.

[24] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: divide the gradient by a running average of its recent magnitude," *COURSERA: Neural Networks for Machine Learning*, vol. 4, no. 2, 2012.
[25] D. Kingma and J. Ba, "Adam: a method for stochastic optimization," 2014, https://arxiv.org/abs/1412.6980.
[26] https://www.python.org/downloads/release/python-352/.