


Article

# Convolutional Neural Network (CNN)-Based Frame Synchronization Method

Eui-Rim Jeong <sup>1</sup>, Eui-Soo Lee <sup>1</sup>, Jingon Joung <sup>2</sup> and Hyukjun Oh <sup>3,\*</sup>

<sup>1</sup> Department of Info. and Commun. Engineering, Hanbat National University, Daejeon 34158, Korea; erjeong@hanbat.ac.kr (E.-R.J.); 30191055@edu.hanbat.ac.kr (E.-S.L.)

<sup>2</sup> School of Electrical and Electronics Engineering, Chung-Ang University, Seoul 06974, Korea; jgjoung@cau.ac.kr

<sup>3</sup> Department of Electronics and Communications Engineering, Kwangwoon University 26 Kwangwoon-ro, Nowon-gu, Seoul 01891, Korea

\* Correspondence: hj\_oh@kw.ac.kr

Received: 5 September 2020; Accepted: 14 October 2020; Published: 17 October 2020



**Abstract:** A new frame synchronization technique based on convolutional neural network (CNN) is proposed for synchronized networks. To estimate the exact packet arrival time, the receiver typically uses the correlator between the received signal and the preamble or pilot in front of the transmitted packet. The conventional frame synchronization technique searches the correlation peak within the time window. In contrast, the proposed method utilizes a CNN to find the packet arrival time. Specifically, in the proposed method, the 1D correlator output is converted into a 2D matrix by reshaping, and the resulting signal is inputted to the proposed 4-layer CNN classifier. Then, the CNN predicts the packet arrival time. To verify the frame synchronization performance, computer simulation is performed for two channel models: additive white Gaussian noise and fading channels. Simulation results show that the proposed CNN-based synchronization method outperforms the conventional correlation-based technique by 2 dB.

**Keywords:** CNN; 2D transformation; frame synchronization; deep learning; synchronized communication networks

## 1. Introduction

Finding the packet arrival time or frame synchronization at a receiver is an essential procedure that must be performed first for the data reception [1,2]. In synchronized communication networks, such as time division multiple access (TDMA) and synchronized carrier sense multiple access (CSMA) systems, a packet is transmitted at a predesignated time. Thus, the approximate arrival time of the transmission frame is known at the receiver because the transmission time is shared between the transmitter (Tx) and the receiver (Rx). Those systems are widely used in internet-of-things (IoT) communications [3]. However, due to the clock offset between the Tx and the Rx and the propagation delay caused by the distance between them, the arrival time of the packet varies and is unknown at the Rx. Depending on the amount of clock offset and the communication distance, the packet arrival time can be defined within a certain time interval, i.e., a time window. Finding the packet arrival time at the received signal is called frame synchronization. Frame synchronization for burst communication is a well-established research field [4–10].

To facilitate frame synchronization, the transmitter usually transmits a unique word in front of the packet (or preamble). The receiver finds the packet arrival time by detecting the received preamble. Many frame synchronization techniques by using preamble have been suggested for binary phase-shift keying (BPSK) systems [4,5], M-ary PSK systems [6], continuous phase modulation (CPM)

systems [7,8], and orthogonal frequency division multiple access (OFDM) systems [9,10]. Optimum frame synchronization is known as the maximum likelihood (ML) technique [4,8]. ML technique requires an exhaustive search among all possible packet arrival times. The huge computation is difficult to handle. Therefore, practical frame synchronization techniques have been researched so far [4–10]. Those practical frame synchronization techniques, regardless of modulation schemes, are based on the correlation peak search. In other words, the conventional frame synchronization methods find the packet arrival time through the correlation between the received signal and the preamble. In detail, when the output of the correlator exceeds a certain threshold, the instance is determined to be the packet arrival time. The best threshold, in general, is a function of signal-to-noise ratio (SNR). Thus, to find the optimal threshold, the SNR of the received signal should be estimated before the frame synchronization. As the frame synchronization performance is highly sensitive to the SNR estimation accuracy [11], as one of the most intuitive methods, the peak detection of the correlator output within the time window is widely used [4]. Frame synchronization based on peak search at correlator output has a long history, but it is still the most widely used technique in recent communication systems [6,8,10].

In this paper, to improve the synchronization performance of the synchronized networks, we propose a new frame synchronization method based on a convolution neural network (CNN) classifier. The CNN, one of the most famous deep learning methods, first appeared in the introduction of the LeNet-5 that recognizes handwritten numbers, and recently, it is widely used in the field of image processing [12–16] and wireless signal processing [17,18]. Herein, we propose a new frame synchronization method by transforming the frame synchronization problem into a CNN problem. To the best of our knowledge, there is no existing work that applies CNN-based techniques to the frame synchronization problem. Specifically, the one-dimensional (1D) correlator output for frame synchronization is transformed to a two-dimensional (2D) signal, and the 2D signals are used as the training samples with the ground truth labels, which are obtained in the training signal generation. As CNN is specialized in image processing, converting the original 1D signal into 2D signal is a widely used technique to apply CNN in other applications. The training samples are generated under additive white Gaussian noise (AWGN) channels with random arrival times and SNRs. Those 2D signals are inputted to the CNN, and the CNN classifier is trained to predict the packet arrival time. We design the CNN classifier with three convolutional layers and one fully connected layer. The proposed technique does not require any prior information except the correlator output. We examine the false detection probability (FDP) of the proposed CNN-based and the convolutional correlation-based methods through computer simulation. Without retraining the CNN, the FDP performances are evaluated under AWGN and fading channels. According to the results, it is verified that the proposed CNN-based technique outperforms the conventional method by 2 dB in both AWGN and fading channels. The main contribution of this study is summarized as follows.

- CNN-based techniques to the frame synchronization problem.
- 1D correlator output is transformed to a 2D signals for better training of the designed CNN.
- The designed CNN is evaluated under various channel environments, namely, AWGN and fading channels.
- The proposed CNN-based method improves approximately 2 dB SNR for the frame synchronization.

## 2. System Model

In this study, we consider a synchronized communication network, in which a Tx transmits signals to an Rx at predesignated time  $t_s$ . The Tx packet consists of the preamble and data in the front and end of the packet, respectively, as shown in Figure 1. In this scenario, the Tx transmits a packet at  $t_s$ , yet the packet arrival time has some deviation due to clock offset and propagation delay between the Tx and the Rx. As the time deviation is bounded within a certain time window with size  $W$  by designing the system depending on the amount of clock offset and the distance, we can assume that the packet arrival time falls within the time window  $[0, \dots, W - 1]$ . The preamble consists of the

BPSK modulated signals and its length is denoted by  $L$ . Here, we note that any modulation scheme is applicable to the proposed technique, which will be introduced shortly.

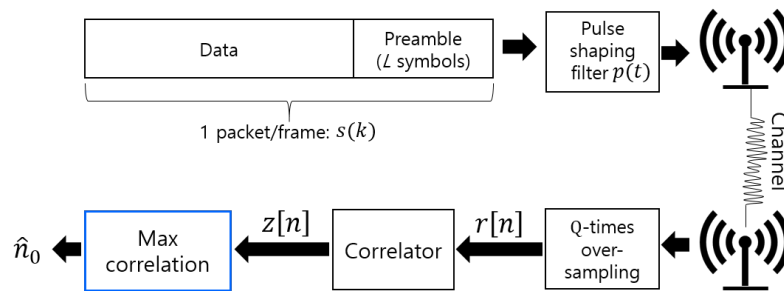


Figure 1. Overall block diagram of the conventional correlation-based frame synchronization.

Figure 1 shows the overall block diagram of frame synchronization. It is assumed that the received signal is  $Q$ -times oversampled compared to the transmitted symbol rate. The received analog signal and its sampled digital signal, respectively, can be represented as

$$r(t) = \sum_{k=-\infty}^{\infty} s(k)p(t - kT - t_0) + w(t), \tag{1}$$

$$r[n] = r(t)|_{t=nT/Q}, \tag{2}$$

where  $T$  is the symbol duration,  $s(k)$  is the Tx symbol (first  $L$  symbols are the preamble),  $p(t)$  is the impulse response of the pulse shaping filter,  $w(t)$  is white Gaussian noise, and  $t_0$  is the packet arrival time that should be found at the Rx. In conventional frame synchronizers,  $t_0$  is found via the correlation between the preamble  $s(k)$  and the received signal  $r[n]$ . As the sampling frequency of the received signals is  $Q$ -times higher than that of  $s(k)$ , the correlation in the correlator in Figure 1 can be obtained as

$$z[n] = \left| \sum_{l=0}^{L-1} r[n - lQ]s^*[L - l - 1] \right|^2, \tag{3}$$

and thus, the structure of the correlator is as shown in Figure 2.

As the sampling frequency is  $Q/T$ , the packet arrival time  $t_0$  corresponds to  $n_0 = \lceil t_0 \frac{Q}{T} \rceil$  in the correlator output, where  $\lceil \cdot \rceil$  denotes a ceiling operation. Therefore,  $n_0$  is the starting point of the packet and it should be found at the Rx.

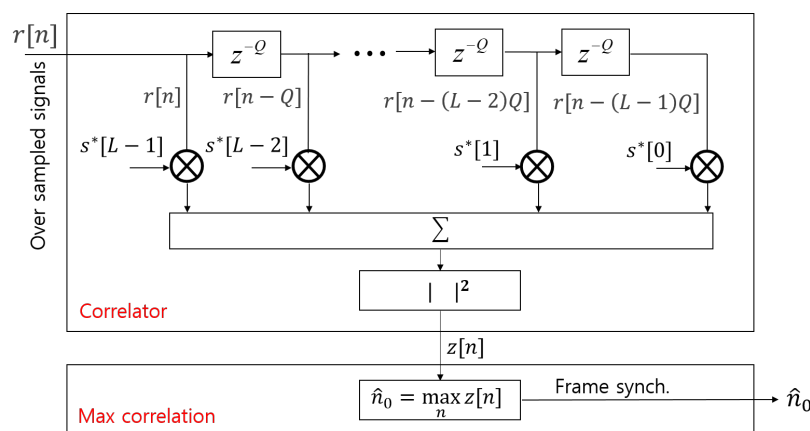


Figure 2. Structure of the correlator and frame synchronization based on the maximum correlation in Figure 1.

Figure 3 shows an example of the correlator output, where the window size is 10,000, i.e.,  $W = 10,000$ . The maximum of the correlator output occurs at  $n = 7,913$ , i.e.,  $n_0 = 7,913$ . If  $n_0$

exists within the time window, searching the maximum value of the correlator output  $z[n]$  within the window will be one of the best policies and it is described as follows,

$$\hat{n}_0 = \arg \max_{\{n\}} z[n], \forall n \in [0, W - 1]. \tag{4}$$

In this synchronized network case, it is advantageous that frame synchronization can be performed without SNR estimation. Throughout this paper, this correlation-based method is called a conventional method.

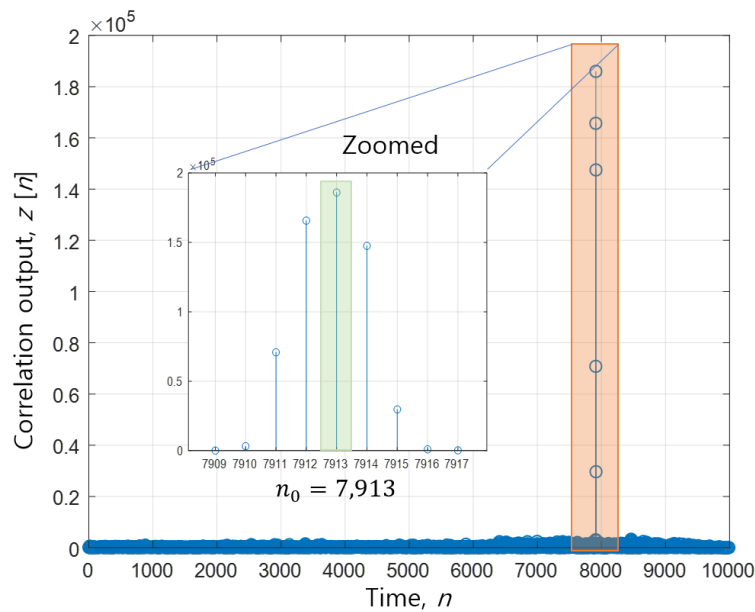


Figure 3. Example of the correlator output,  $z[n]$ , when  $W = 10,000$  and  $SNR = 0.9$  dB.

### 3. Proposed CNN-Based Frame Synchronizer

In this section, we propose a new synchronization method using a CNN classifier as shown in Figure 4. The CNN classifier generates the estimated arrival time, i.e.,  $\hat{n}_0$ , from the input signals that is the correlator output signals,  $z[n]$  in (3). The detailed procedure of the proposed CNN classifier is depicted in Figure 5, and the specific parameters for it are summarized in Table 1. For a simple description of the proposed method, we set the window size by 10,000, i.e.,  $W = 10,000$ . Note that the proposed method can be applied to an arbitrary size of time window by slightly modifying the CNN structure.

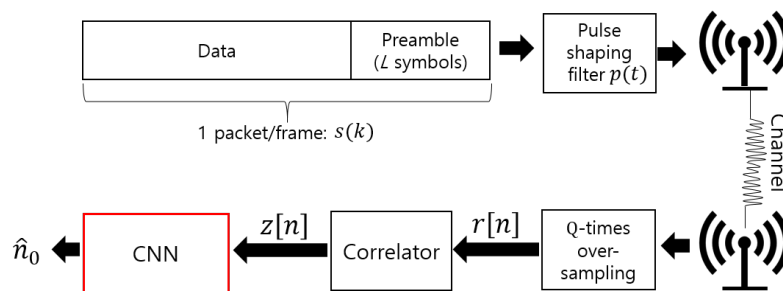
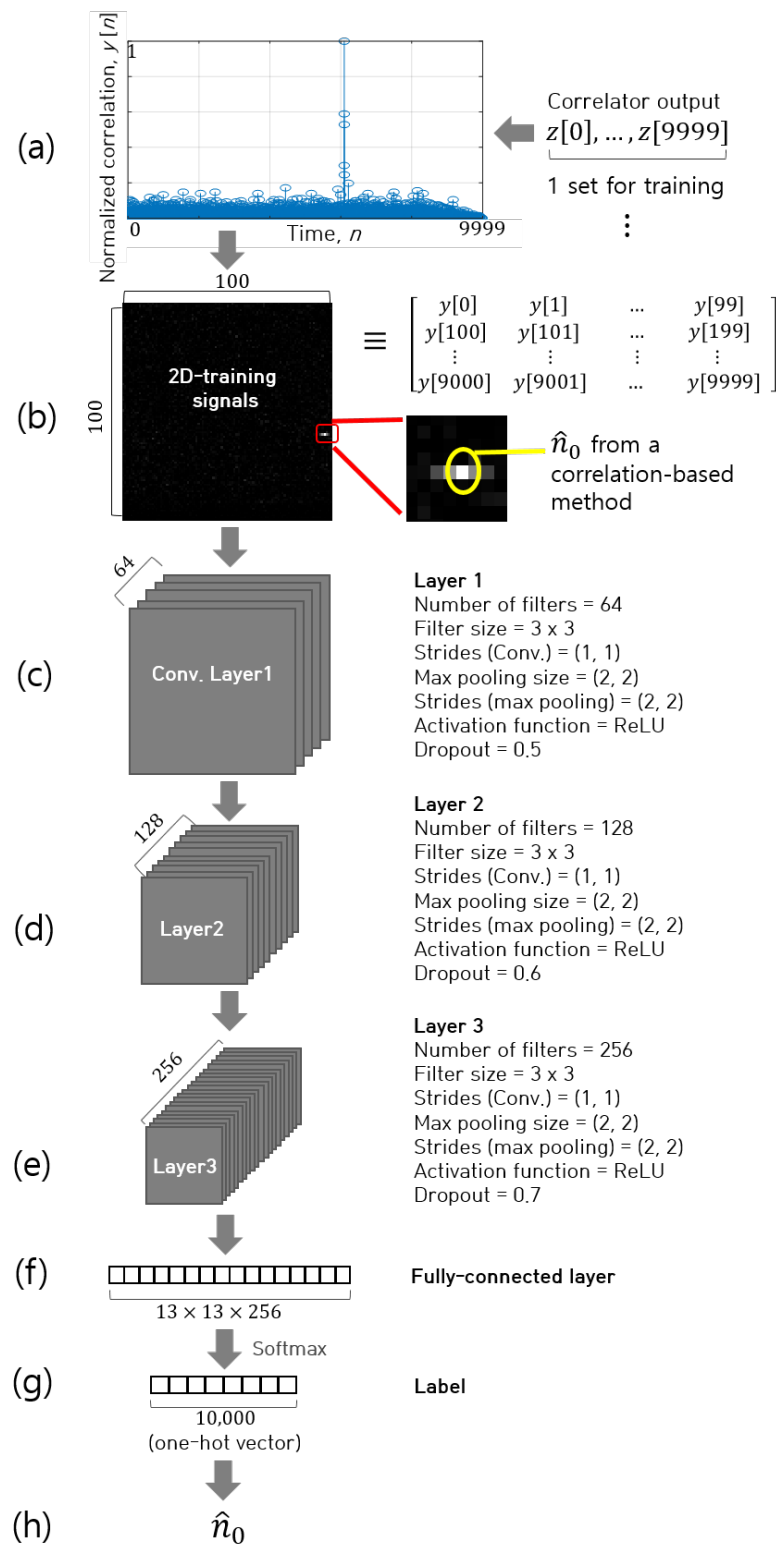


Figure 4. Overall block diagram of the proposed convolutional neural network (CNN)-based frame synchronization.



**Figure 5.** Diagram of the proposed CNN-based synchronization block in Figure 5, when  $W = 10,000$ . (a) Correlation output normalization. (b) 2D-training signal generation. (c) Convolutional layer 1. (d) Convolutional layer 2. (e) Convolutional layer 3. (f) Fully-connected layer. (g) Label vector. (h) Estimate of packet arrival time.

**Table 1.** Proposed CNN and its parameters.

Layers	Function & Parameters	Values
Input layer	Input size	$100 \times 100 \times 1$
Convolutional layer 1	Number of filters	64
	Filter size and stride	$(3 \times 3), (1,1)$
	Activation function	ReLU
	Pooling	Max pooling (size: $2 \times 2$ , stride: 2,2)
	Number of parameters	Weight: $3 \times 3 \times 1 \times 64$ Bias: $1 \times 1 \times 64$ Total: 640
Convolutional layer 2	Number of filters	128
	Filter size and stride	$(3 \times 3), (1,1)$
	Activation function	ReLU
	Pooling	Max pooling (size: $2 \times 2$ , stride: 2,2)
	Number of parameters	Weight: $3 \times 3 \times 64 \times 128$ Bias: $1 \times 1 \times 128$ Total: 73,856
Convolutional layer 3	Number of filters	256
	Filter size and stride	$(3 \times 3), (1,1)$
	Activation function	ReLU
	Pooling	Max pooling (size: $2 \times 2$ , stride: 2,2)
	Number of parameters	Weight: $3 \times 3 \times 128 \times 256$ Bias: $1 \times 1 \times 256$ Total: 295,168
Fully connected layer 1	Output size	512
	Activation function	ReLU
	Number of parameters	Weight: $512 \times 13 \times 13 \times 256$ Bias: $512 \times 1$ Total: 22,151,680
Fully connected layer 2	Output size	10,000
	Activation function	ReLU
	Number of parameters	Weight: $10,000 \times 512$ Bias: $10,000 \times 1$ Total: 5,130,000
Number of total parameters		27,651,344

The first step is a training sample normalization step as shown in Figure 5a, in which the correlator output signal,  $z[n]$ , is normalized such that its maximum value is one as follows,

$$y[n] = \frac{z[n]}{z_{\max}} = \frac{z[n]}{\max_{n' \in [0, W-1]} z[n']}, \quad \forall n \in [0, W-1]. \quad (5)$$

Next, the  $1 \times 10,000$  1D signals are converted to  $100 \times 100$  2D training signals by performing matricization operation with a row-major order and its dimension is  $\sqrt{W}$ . Precisely, as shown in Figure 5b, the first 100 samples become the first row, the next 100 samples become the second row, and so forth. The converted 2D-training signals can be represented by a black and white (monochrome)

image as depicted in Figure 5b, in which the bright and dark colors indicate large and small values of  $y[n]$ , respectively. Thus, we can interpret that the lightest part of the image implies a packet arrival instance of the conventional correlation-based synchronization method. Here, we have to note that the estimated arrival time  $\hat{n}_0$  from the designed CNN could be different from  $\hat{n}_0$  obtained from the maximum correlation-based method. This will be shown in the next section.

The 2D-training signals are then provided to the designed CNN classifier. The structure of the proposed CNN regressor is shown in Figure 5c–g. The input is  $100 \times 100$  2D-training signal and it passes through three convolutional layers (panels (c–e)) and one fully connected layer (panel (f)). The final output in panel (g) is  $1 \times 10,000$  one-hot vector and “1” indicates the packet arrival instance. The convolutional filter size is  $3 \times 3$ , and the number of filters (or channels) at each of the three convolutional layers is 64, 128, and 256, respectively. Using the sufficient number of training signals and their labels, the CNN parameters are updated to minimize the difference the CNN regressor output and the label. In a label vector, only one element has a value of one and the others have zeros, where the position of element 1 indicates the packet arrival time, i.e.,  $\hat{n}_0$  in (h), which is the output of the CNN block.

#### 4. Simulation Results

The performance of the proposed frame synchronizer is examined through computer simulation. Two preamble lengths ( $L = 500$  and  $L = 1000$ ) are considered. The preambles are pseudo-random sequences. Usually, the longer preamble results in better frame synchronization performance. For the training the proposed CNN, a total of 100,000 sets of the received signal are generated. The SNR of each training set is randomly selected between  $-30$  dB and  $30$  dB, and the packet arrival time is also randomly selected in the time window from 0 to 9999, i.e.,  $W = 10,000$ . The learning rate is 0.001 and an optimization algorithm is an adaptive moment estimation (ADAM). The proposed CNN is learned for 80 epochs, i.e., 500,000 training signals are reused 80 times. After successful training, the proposed CNN can find the any packet arrival time in the range of 0 to 9999; therefore, the additional training does not required for different transmission delays. For frame synchronization performance evaluation, test signals are generated under AWGN channel environments with  $SNR = -30$  dB,  $-28$  dB,  $\dots$ ,  $28$  dB,  $30$  dB, and at each SNR, 100,000 test signals are generated with random packet arrival times. As the performance evaluation signals are generated independently with the training signals, the two sets of data do not overlap.

The CNN training and performance evaluation are performed by using MATLAB 2020a. To use useful functions on deep learning, a Deep Learning Toolbox is also required. To accelerate training speed, a graphic process unit (GPU) GTX1080Ti with compute unified device architecture (CUDA) 10.0 is used. To training the CNN, a `trainNetwork` function is used. The input of `trainNetwork` is training input signals, designed neural network, and optimization parameters. The output of `trainNetwork` is the trained parameters of CNN. For performance evaluation of the trained CNN, a `predict` function is used. The input of the `predict` function is the trained CNN parameters and the input signals for the performance test. Table 2 summarizes the simulation software environments.

**Table 2.** Simulation software environments.

Tools and Software	Environments
Operation system	Windows 10
Simulation tool	MATLAB 2020a
MATLAB toolbos	Deep Learning Toolbox
GPU	GTX1080Ti with CUDA 10.0
Function for training	<code>trainNetwork</code>
Function for test	<code>predict</code>

The learning curves of the proposed CNN were shown across the number of epochs for the loss and training accuracy in Figure 6a,b, respectively. To train the CNN, the cross entropy is used for the loss function, defined as

$$V = - \sum_{\hat{t}_0=0}^{9999} p(\hat{t}_0) \log q(\hat{t}_0), \tag{6}$$

where  $\hat{t}_0$  is packet arrival time,  $q(\hat{t}_0)$  is the softmax output in Figure 5, and  $p(\hat{t}_0)$  is the ideal probability, i.e.,

$$p(\hat{t}_0) = \begin{cases} 1, & \hat{t}_0 = t_0, \\ 0, & \hat{t}_0 \neq t_0. \end{cases} \tag{7}$$

The loss in Figure 6a represents  $V$  in (6), and the accuracy in Figure 6b is the ratio of the correctly estimated cases among total 500,000 training signals. From the results, it was evidently shown that the proposed CNN accurately converges at approximately 50 epochs.

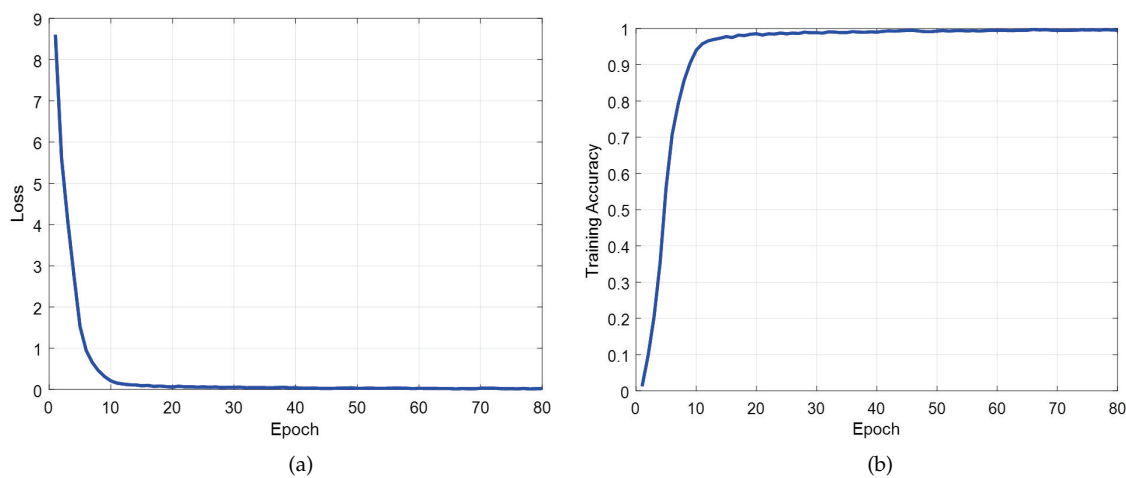


Figure 6. Learning curves for (a) loss and (b) training accuracy.

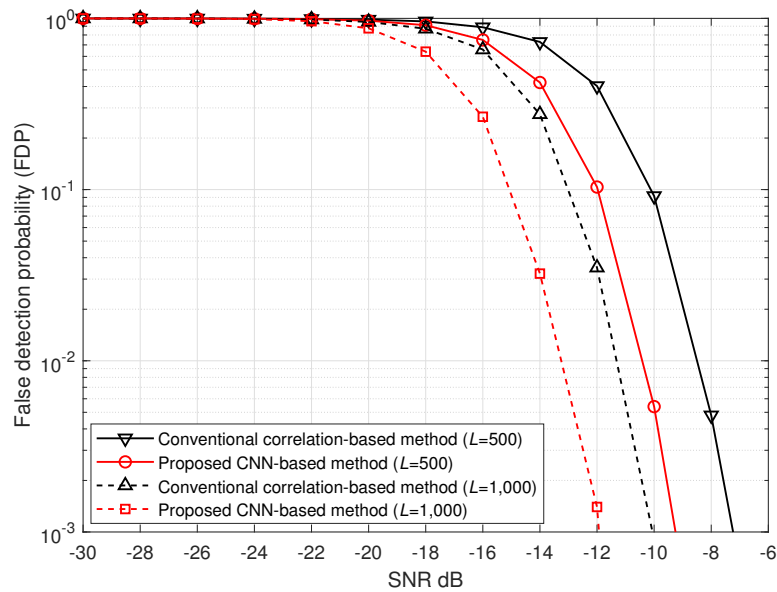
Figure 7 shows the FDPs of the proposed CNN-based and the conventional correlation-based synchronization methods [6,8,10] in AWGN channels. From the results, we observe that the proposed CNN-based technique outperforms the conventional correlation-based method regardless of the preamble length, and a longer preamble provides better FDP. Concretely, the proposed method shows 2 dB gain over the conventional method, regardless of the preamble length, and the preambles with length 1000 achieve 3 dB better than that with length 500.

Figure 8 shows the FDPs for flat fading channels with  $L = 500$ . During performance evaluation in fading channels, the CNN is not retrained. The same CNN trained with signals under AWGN environments is used and the performance evaluation signals are newly generated. To generate received signals in *fading channels*, the following model is used,

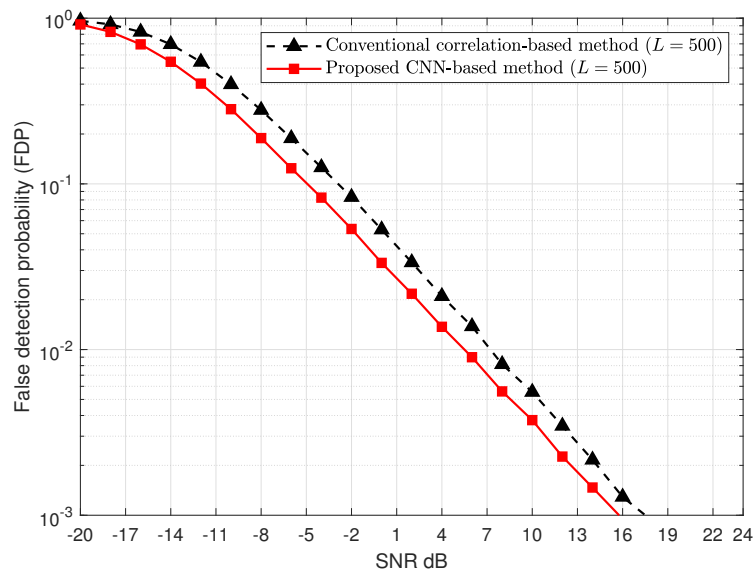
$$r(t) = c \sum_{k=-\infty}^{\infty} s(k)p(t - kT - t_0) + w(t), \tag{8}$$

where  $c$  is the fading coefficient which is Gaussian random variable with variance one. This channel model is suitable for non-line-of-sight terrestrial communication systems. This simulation is performed to confirm whether the proposed technique works well for the received signals different from the training situation. According to the results, the proposed method still 2 dB better than the conventional method. Those results indicate that the proposed CNN operates very flexibly in various channel environments. By improving the frame synchronization performance by 2 dB, the transmitter can reduce the transmit power to increase the battery lifetime, or extend a communication range farther.



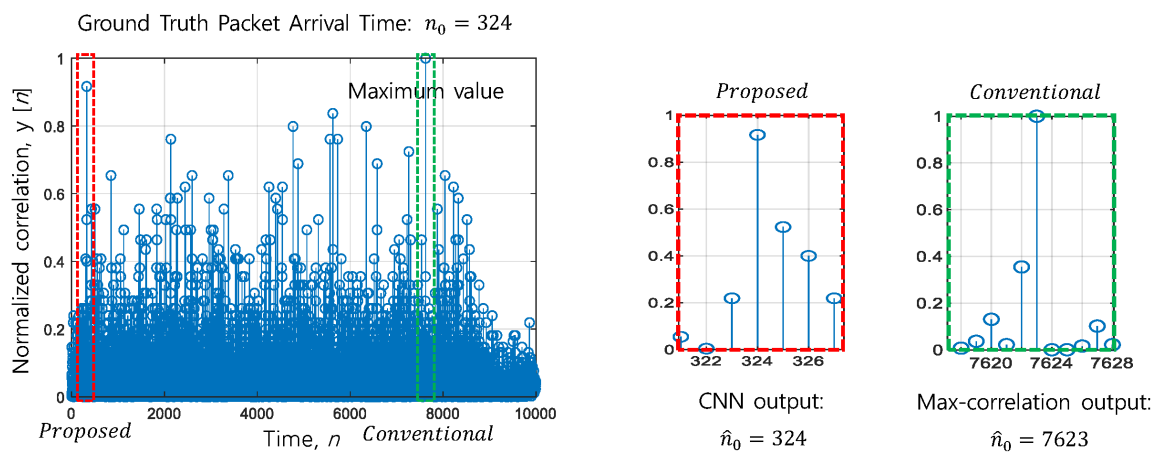


**Figure 7.** False detection probabilities (FDPs) of the proposed CNN-based and the conventional correlation-based synchronization methods in [6,8,10] for  $L = 500$  and  $L = 1000$ .



**Figure 8.** False detection probabilities (FDPs) of the proposed CNN-based and the conventional correlation-based synchronization methods in [6,8,10] for  $L = 500$  in fading channels.

To investigate the rationale of why the proposed CNN-based technique outperforms the conventional correlation-based method, we show one snapshot Figure 9 of the correlator output for the case when the correlation-based technique finds the wrong packet arrival time, while the CNN-based method finds it correctly. The ideal correlator output conforms to the normal distribution with a mean value around the packet arrival time due to oversampling effects. The conventional correlation-based frame synchronization method finds just the maximum position ignoring the shape of the distribution of  $y[n]$ . However, we can conjecture that the proposed CNN-based frame synchronization method finds the packet arrival time by considering not only the scale but also the shape of the correlation values.



**Figure 9.** A case when proposed CNN method is correct but the conventional method is wrong.

## 5. Conclusions

In this study, we proposed a CNN-based frame synchronization method for the synchronized networks. As a conventional correlation-based synchronization method, the proposed CNN-based synchronization method can find the packet arrival time only from the correlator output. Instead of finding the correlation peak, the proposed technique find the time offset directly by the five-layer CNN classifier. The designed CNN consists of three convolutional layers and two fully connected layers. The computer simulation verified that the proposed CNN-based method significantly outperforms the conventional correlation-based method regardless of the preamble size. Those results indicate that the proposed method enable power saving of the transmitter by reducing the transmitted power or longer range transmission due to the enhancement at low SNRs. The proposed technique for the synchronized networks can be applied to the carrier sense multiple access type networks if a valid packet arrival can be identified before the proposed frame synchronizer. In future studies, it is worth (i) verifying through a testbed that the proposed method performs under various channel conditions in practice, and (ii) developing a novel CNN structure to further improve the synchronization performance and/or reduce the training complexity.

**Author Contributions:** Conceptualization, E.-R.J.; writing—original draft preparation, E.-R.J. and H.O.; simulation, E.-R.J. and E.-S.L.; writing—review and editing, J.J.; supervision, E.-R.J., H.O., and J.J.; funding acquisition, H.O. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work has been supported by the Future Combat System Network Technology Research Center program of Defense Acquisition Program Administration and Agency for Defense Development (UD190033ED).

**Conflicts of Interest:** The authors declare no conflict of interest.

## Abbreviations

The following abbreviations are used in this manuscript.

ADAM	Adaptive Moment Estimation
AWGN	Additive White Gaussian Noise
BPSK	Binary Phase-Shift Keying
CNN	Convolutional Neural Network
CSMA	Carrier Sense Multiple Access
CUDA	Compute Unified Device Architecture
FDP	False Detection Probability
GPU	Graphic Process Unit
IoT	Internet-of-Things

ReLU	Rectified Linear Unit
Rx	Receiver
SNR	Signal-to-Noise Ratio
TDMA	Time Division Multiple Access
Tx	Transmitter
1D	One-Dimensional
2D	Two-Dimensional

## References

1. Kwon, T.-H.; Kim, J.-E.; Kim, K.-D. Time-Sharing-based Synchronization and Performance Evaluation of Color-Independent Visual-MIMO Communication. *Sensors* **2018**, *18*, 1553. [[CrossRef](#)] [[PubMed](#)]
2. Offiong, F.B.; Sinanović, S.; Popoola, W.O. Pilot-Aided Frame Synchronization in Optical OFDM Systems. *Appl. Sci.* **2020**, *10*, 4034. [[CrossRef](#)]
3. Yadav, P.; McCann, J.A.; Pereira, T. Self-Synchronization in Duty-Cycled Internet of Things (IoT) Applications. *IEEE Internet Things J.* **2017**, *4*, 2058–2069. [[CrossRef](#)]
4. Massey, J. Optimum Frame Synchronization. *IEEE Trans. Commun.* **1972**, *20*, 115–119. [[CrossRef](#)]
5. Chiani, M. Noncoherent Frame Synchronization. *IEEE Trans. Commun.* **2010**, *58*, 1536–1545. [[CrossRef](#)]
6. Elzanaty, A.; Koroleva, K.; Gritsutenko, S.; Chiani, M. Frame Synchronization for M-Ary Modulation with Phase Offsets. In Proceedings of the IEEE 17th International Conference Ubiquitous Wireless Broadband (ICUWB), Salamanca, Spain, 12–15 September 2017; pp. 1–6.
7. Hosseini, E.; Perrins, E. Timing, Carrier, and Frame Synchronization of Burst-Mode CPM. *IEEE Trans. Commun.* **2013**, *61*, 5125–5138. [[CrossRef](#)]
8. Rice, M.; Mcurdie, A. On Frame Synchronization in Aeronautical Telemetry. *IEEE Trans. Aerosp. Electron. Syst.* **2016**, *52*, 2263–2280. [[CrossRef](#)]
9. Gao, Z.; Zhang, C.; Wang, Z. Robust Preamble Design for Synchronization, Signaling Transmission, and Channel Estimation. *IEEE Trans. Broadcast.* **2015**, *61*, 98–104. [[CrossRef](#)]
10. Zhang, H.; Hou, Y.; Chen, Y.; Li, S. Analysis of Simplified Frame Synchronization Scheme for Burst-Mode Multi-Carrier System. *IEEE Commun. Lett.* **2019**, *23*, 1054–1056. [[CrossRef](#)]
11. Pauluzzi, D.; Beaulieu, N. A Comparison of SNR Estimation Techniques for the AWGN Channel. *IEEE Trans. Commun.* **2000**, *48*, 1681–1691. [[CrossRef](#)]
12. Lawrence, S.; Giles, C.; Tsoi, A.; Back, A. Face Recognition: A Convolutional Neural-Network Approach. *IEEE Trans. Neural Netw.* **1997**, *8*, 98–113. [[CrossRef](#)] [[PubMed](#)]
13. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based Learning Applied to Document Recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [[CrossRef](#)]
14. Krizhevsky, A.; Sutskever, I.; Hinton, G. Imagenet Classification with Deep Convolutional Neural Networks. In Proceedings of the Advances in Neural Information Processing Systems, Lake Tahoe, NV, USA, 3–6 December 2012; pp. 1097–1105.
15. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.
16. Karpathy, A.; Toderici, G.; Shetty, S.; Leung, T.; Sukthankar, R.; Fei-Fei, L. Large-Scale Video Classification with Convolutional Neural Networks. In Proceedings of the IEEE Conference of Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 1725–1732.
17. Joung, J.; Jung, S.; Chung, S.; Jeong, E.-R. CNN-based Tx-Rx Distance Estimation for UWB System Localization. *IET Elect. Lett.* **2019**, *55*, 938–940. [[CrossRef](#)]
18. Kim, M.; Zhang, Z.; Kim, D.; Choi, S. Deep-Learning-based Frame Format Detection for IEEE 802.11 Wireless Local Area Networks. *Electronics* **2020**, *9*, 1170. [[CrossRef](#)]

**Publisher’s Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).