

# Deep Learning for Radar Pulse Detection

Ha Q. Nguyen, Dat T. Ngo and Van Long Do

*Viettel Research and Development Institute, Hoa Lac High-tech Park, Hanoi, Vietnam*

**Keywords:** Deep Neural Network, Radar Pulse Detection, Radar Pulse Parameter Estimation, Pulse Description Word, Change Point Detection, Pruned Exact Linear Time.

**Abstract:** In this paper, we introduce a deep learning based framework for sequential detection of rectangular radar pulses with varying waveforms and pulse widths under a wide range of noise levels. The method is divided into two stages. In the first stage, a convolutional neural network is trained to determine whether a pulse or part of a pulse appears in a segment of the signal envelop. In the second stage, the change points in the segment are found by solving an optimization problem and then combined with previously detected edges to estimate the pulse locations. The proposed scheme is noise-blind as it does not require a noise floor estimation, unlike the threshold-based edge detection (TED) method. Simulations also show that our method significantly outperforms TED in highly noisy cases.

## 1 INTRODUCTION

The detection of radar pulses—or the estimation of the times of arrival (TOAs) and the times of departure (TODs)—plays a central role in passive location systems as it provides input for other algorithms to locate the emitter (Torrieri, 1984; Poisel, 2005). This is a challenging task since radar pulses are modulated and coded with a variety of waveforms and most of the time buried in noise. Existing methods for radar pulse detection are usually threshold-based (Torrieri, 1974; Iglesias et al., 2014) in which the thresholds are determined via an estimation of the noise statistics. These methods work well with high or moderate Signal-to-Noise-Ratios (SNRs) but perform poorly with low SNRs. Furthermore, the noise floor estimation—a prerequisite of these algorithms—is itself a hard problem, especially in highly varying environments.

In the past few years, Deep Learning (LeCun et al., 2015; Goodfellow et al., 2016) has proved a powerful tool for many tasks in computer vision and signal processing, notably, image classification (Krizhevsky et al., 2012; Szegedy et al., 2015; He et al., 2016) and object detection (Ren et al., 2017; Redmon et al., 2016; Liu et al., 2016). Motivated by these successes, we propose a novel method for radar pulse detection in which edges are sequentially estimated from *segments* of the received signal *envelop* via a deep-learning-based segment classification followed by a find-change-points algorithm

(Killick et al., 2012). The segment classification essentially determines if a pulse is present, partially present, or absent in a segment through a Convolutional Neural Network (CNN). For a segment of small-enough length, it can only fall into one of the 5 categories: ‘2 edges’, ‘TOA only’, ‘TOD only’, ‘All pulse’, and ‘All noise’. Based on the output of the CNN, the find-change-points routine seeks edges in the segment by minimizing a cost function associated to the number of edges, instead of a thresholding procedure. This approach is therefore able to get rid of the unreliable noise floor estimation. The contributions of our paper are summarized as follows.

- A novel CNN architecture for segment classification.
- An algorithm for adaptive segment classification in which the CNN predicts the class of the current segment based on the confidence of its previous prediction.
- An algorithm for sequential pulse localization that combines the segment classification with a find-change-points algorithm. This method significantly surpasses the performance of the Threshold-based Edge Detection (Iglesias et al., 2014), especially in the low-SNR regimes.

The rest of the paper is outlined as follows: Sec. 2 formulates the problem. Sec. 3 constructs the CNN and integrates it into an adaptive algorithm for segment classification. Sec. 4 presents the main algo-

rithm for sequential pulse localization. Sec. 5 reports some numerical results. Finally, Sec. 6 provides concluding remarks.

## 2 PROBLEM FORMULATION

In this paper, we are interested in the detection of radar pulses in a narrow bandwidth of, say, 78.125 MHz, centered at frequency 0.<sup>1</sup> Assuming Nyquist sampling, we acquire a discrete-time corrupted signal

$$\hat{x}[n] = x[n] + w[n], \quad n \in \mathbb{Z}, \quad (1)$$

where  $x$  is a *complex-valued* signal that contains a train of rectangular pulses and  $w$  is an additive white Gaussian noise (AWGN). Note that, each sample of the discrete-time signal then corresponds to a period of 12.8 ns. The pulses might have various waveforms and pulse widths. We consider in this paper 7 types of pulse modulations that are commonly used in modern radar systems (Levanon and Mozeson, 2004; Richards, 2014; Pace, 2009), namely, Continuous Waveform (CW), Step Frequency Modulation (SFM), Linear Frequency Modulation (LFM), Non-Linear Frequency Modulation (NLFM), Costas code (COSTAS), Barker code (BARKER), and Frank code (FRANK). See Fig. 1 for an illustration of these modulations. The pulse widths vary in an extremely large range from about 0.1  $\mu$ s to about 400  $\mu$ s, which corresponds to a range from 8 samples to 31250 samples.

The location of each pulse is characterized by the time of arrival (TOA)—the middle of the rising edge—and the time of departure (TOD)—the middle of the falling edge. Our task is to estimate the series of TOAs and TODs in a *sequential* manner by looking at one *segment* of the noisy signal  $\hat{x}$  at a time. To reduce the effect of noise, we perform a preprocessing step in which a lowpass filter  $h$  of bandwidth 20 MHz is applied to  $\hat{x}$  to get

$$\hat{x}_f = h * \hat{x} = h * x + h * w =: x_f + w_f, \quad (2)$$

where  $w_f$  is now a *color* Gaussian noise. Our detection algorithm takes as input sequential segments of the *envelop* (or magnitude)  $|\hat{x}_f|$  and outputs a list of pulse description words (PDWs), each of which consists of a TOA and a TOD of the corresponding pulse. This is done via a CNN-based *adaptive* classification of the segments followed by a find-change-points algorithm that will be described in the next sections.

<sup>1</sup>In practice, the detection of radar pulses would involve a frequency tuning from wide band to narrow band. This crucial step is out of the scope of this paper and will be discussed elsewhere.

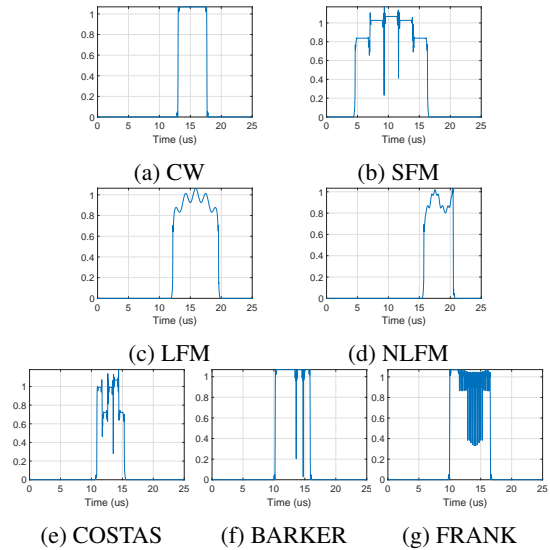


Figure 1: Envelops of 7 modulation types of rectangular pulses without noise. What shown here are the results achieved after the 20-MHz-lowpass filtering.

## 3 CNN-BASED SEGMENT CLASSIFICATION

From a practical point of view, we can reasonably assume that the distance between consecutive pulses is at least 30  $\mu$ s, which is equivalent to about 2344 samples. Therefore, we choose to divide the envelop  $|\hat{x}_f|$  into segments of  $k = 2000$  samples with 50% overlapping.<sup>2</sup> This is to make sure that there will be at most one pulse in a segment. Then, each segment can only fall into one of the following 5 classes.

- Class 1 (‘2 edges’): a TOA and a TOD of a pulse both appear in the segment.
- Class 2 (‘TOA only’): a TOA appears in the segment without TOD.
- Class 3 (‘TOD only’): a TOD appears in the segment without TOA.
- Class 4 (‘All pulse’): the whole segment is part of a long pulse.
- Class 5 (‘All noise’): the segment contains only background noise.

Fig. 2 depicts these classes under a relatively low SNR. To perform the segment classification, we propose a Convolutional Neural Network (CNN) whose architecture is shown in Fig. 3. This network consists of 13 hidden layer, including 3 convolution layers and 3 dense layers. Each convolution layer is followed

<sup>2</sup>In general, the segment length should be chosen to be equal to the minimum distance between consecutive pulses.

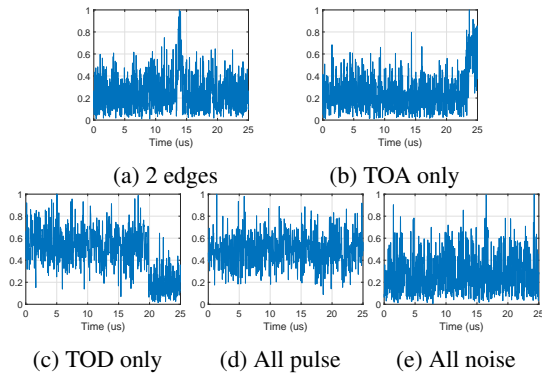


Figure 2: The 5 classes of segments of the signal envelop for SNR = 7 dB at bandwidth 20 MHz.

by a batch normalization, which helps eliminate the internal covariate shift problem (Ioffe and Szegedy, 2015). Two max pooling layers are also inserted to reduce the number of features by a factor of 4 after the first two convolutions. That results in a vector of the same length as the input (2000) right after the flatten layer. A dropout layer with dropping ratio 0.2 is then added to avoid over-fitting. We use the Rectified Linear Unit (ReLU) as the activation function in all layers except for the last dense layer where the softmax function is applied to produce a score vector of the 5 classes. Note that, before feeding a segment  $s$  to the CNN, we normalize it according to

$$s_{\text{norm}}[n] = \frac{s[n]}{\max_i s[i]}, \forall n. \quad (3)$$

It is remarkable that the classes of the segments are not independent of each other. Specifically, the following rules must hold:

- ‘2 edges’ cannot be followed by ‘All pulse’
- ‘TOA only’ cannot be followed by ‘All noise’
- ‘TOD only’ cannot be followed by ‘All pulse’
- ‘All pulse’ must be followed by ‘TOD only’ or ‘All pulse’
- ‘All noise’ cannot be followed by either ‘TOD only’ or ‘All pulse’.

Putting these observations together, we devise an algorithm for adaptive segment classification. In this method, the current prediction and confidence are computed from the CNN and the previous prediction if the previous confidence is above some threshold  $T$ ; otherwise, the CNN makes a prediction as normal. The confidence of a prediction is nothing but the score of the predicted class. In experiments, we always choose the confidence threshold to be  $T = 0.7$ . The pseudocode for this algorithm is provided in Algorithm 1.

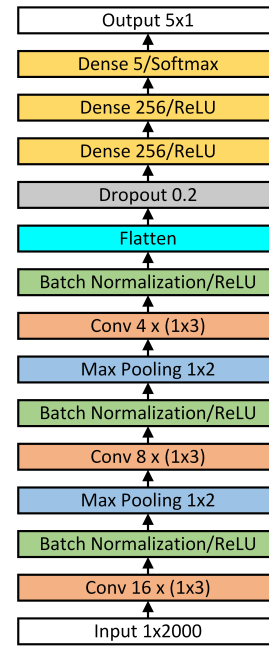


Figure 3: Architecture of the classification network.

Algorithm 1: Adaptive Classification.

```

function AdaptClass ( $s, net, clPr, cfPr, T$ )
    input : A segment  $s$ , classification model  $net$ ,
            previous class  $clPr$ , previous
            confidence  $cfPr$ , confidence threshold
             $T$ 
    output: A class  $cl$  and a confidence  $cf$ 
    // predict using pretrained net
     $score \leftarrow net.predict(s)$ ;
    // check previous confidence
    if  $cfPr < T$  then  $I \leftarrow \{1, 2, 3, 4, 5\}$ ;
    else // restrict the score vector
        switch  $clPr$  do
            case 1  $I \leftarrow \{1, 2, 3, 5\}$ ;
            case 2  $I \leftarrow \{1, 2, 3, 4\}$ ;
            case 3  $I \leftarrow \{1, 2, 3, 5\}$ ;
            case 4  $I \leftarrow \{3, 4\}$ ;
            case 5  $I \leftarrow \{1, 2, 5\}$ ;
        endsw
    end
     $cl \leftarrow \arg \max_{i \in I} (score[i])$ ;
     $cf \leftarrow score[cl]$ ;
    return  $cl, cf$ ;
end function
    
```

## 4 PULSE LOCALIZATION

Based on the outputs of the adaptive segment classification, we sequentially perform the estima-

tion of TOAs and TODs via two find-change-point functions denoted by  $\text{FindChangePts}(s, 1)$  and  $\text{FindChangePts}(s, 2)$ . Specifically, for a segment  $s$  of length  $k$ , the function  $\text{FindChangePts}(s, 1)$  returns a single point  $p^* \in \{1, 2, \dots, k-1\}$  that minimizes the cost function

$$C(p) = p \text{Var}(s_{1:p}) + (k-p) \text{Var}(s_{p+1:k}), \quad (4)$$

where  $\text{Var}(s_{m:n})$  denotes the variance of the sequence  $\{s[i]\}_{m \leq i \leq n}$ . Similarly, the function  $\text{FindChangePts}(s, 2)$  returns two points  $p_1^*$  and  $p_2^*$  that minimize the cost function

$$C(p_1, p_2) = p_1 \text{Var}(s_{1:p_1}) + (p_2 - p_1) \text{Var}(s_{p_1+1:p_2}) + (k - p_2) \text{Var}(s_{p_2+1:k}). \quad (5)$$

These optimization problems can be efficiently solved by using the Pruned Exact Linear Time (PELT) (Killick et al., 2012), which is a speed-up version of the optimal partition method (Jackson et al., 2005).

Once the TOAs and TODs are estimated, overlapping or nearby pulses must be merged into a single one. In particular, if the difference between the current TOA and the previous TOD is less than some minimum distance  $d$ , we update the previous TOD to the current TOD and discard the current pulse. The whole procedure for sequential pulse localization is described in Algorithm 2.

## 5 SIMULATIONS

In this section, we provide some numerical results for the detection of simulated radar pulses. We generated all data and implemented Algorithms 1 and 2 in Matlab 2018a running on PC with an Intel Core i7-7700 CPU @ 3.6 GHz. To realize the find-change-points algorithm, we invoked the Matlab built-in function `findchangepts`. The training of the classification network was implemented in Python with Keras library and Tensorflow backend running on an Nvidia Tesla P100 GPU. The trained Keras model was then imported and run in Matlab via the Neural Network Toolbox.

### 5.1 Training the Classification Net

We trained the classification net using 500,000 training examples, each of which is a segment of 2000 samples randomly truncated from a longer signal with a single rectangular pulse. Each pulse is randomly generated with one of the 7 modulation types in Fig. 1, with a pulse width in the range from  $0.1 \mu\text{s}$  to  $400 \mu\text{s}$ , and with an SNR (at bandwidth 20 MHz) in the range from 0 dB to 30 dB. To guarantee that the

Algorithm 2: Sequential Pulse Localization.

**input** : An envelop  $env$ , segment length  $k$ , overlap length  $m$ , classification model  $net$ , confidence threshold  $T$ , minimum pulse distance  $d$

**output**: A list of PDWs with TOA and TOD attributes

```

Initialize an empty list PDW;
idx ← 0; start ← 1; stop ← k;
cf = 0; cl = 1;
while stop ≤ Length(env) do
    // extract a segment
    s = env[start : stop];
    // classify the segment
    [cl, cf] ← AdaptClass(s, net, cl, cf, T);
    // Find change points
    switch class do
        case 1
            [toa, tod] ← FindChangePts(s, 2);
        case 2
            toa ← FindChangePts(s, 1);
            tod ← k;
        case 3
            toa ← 1;
            tod ← FindChangePts(s, 1);
        case 4
            toa ← 1; tod ← k;
        end
    endsw
    // convert to global coordinates
    toa ← toa + start - 1;
    tod ← tod + start - 1;
    if idx = 0 then
        idx ← idx + 1;
        PDW(idx).TOA ← toa;
        PDW(idx).TOD ← tod;
    end
    else
        if toa - PDW(idx).TOD < d then
            // Update TOD
            PDW(idx).TOD ← tod;
        end
        else
            idx ← idx + 1;
            PDW(idx).TOA ← toa;
            PDW(idx).TOD ← tod;
        end
    end
    start ← stop - m + 1;
    stop ← start + k - 1;
end

```

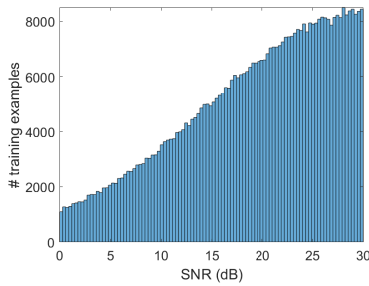


Figure 4: Distribution of the number of training examples with respect to SNR level.

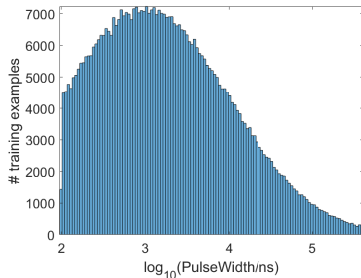


Figure 5: Distribution of the number of training examples with respect to pulse width in logarithmic scale.

classification accuracy would increase with SNR, we generated each noisy signal with an SNR drawn from a truncated normal distribution rather than a uniform distribution, as shown in Fig. 4. As the range of the pulse widths is too large compared to the length of a segment, we chose to draw each pulse width from a truncated normal distribution in the logarithmic scale, as shown in Fig. 5. Furthermore, to reduce the number of false alarms in later detection, too-short pulses with too-low SNRs were excluded from the training set. More precisely, we restricted the range of the pulse widths to  $[PW_{\min}, PW_{\max}]$  where  $PW_{\max} = 400 \mu s$  and  $PW_{\min}$  is dependent on SNR as

$$PW_{\min}(\text{SNR}) = \max\{2 \times 10^{-\text{SNR}/10}, 0.1\} (\mu s).$$

For instance,  $PW_{\min} = 2 \mu s$  for  $\text{SNR} = 0 \text{ dB}$ ,  $PW_{\min} \approx 0.4 \mu s$  for  $\text{SNR} = 7 \text{ dB}$ ,  $PW_{\min} = 0.1 \mu s$  for all  $\text{SNR} \geq 13 \text{ dB}$ , and so on.

For the purpose of testing the classification net, we also simulated a testing set of 125,000 examples using the same procedure as that of the training set. The classification net was trained via Adam optimizer with a learning rate of  $10^{-4}$  for 100 epochs with a batch size of 256. After being trained for about half an hour, the classification net yields an accuracy of 99.47% on the training set and of 99.18% on the testing set. The confusion matrices are plotted in Fig. 6.

## 5.2 Pulse Detection Results

To test the whole detection procedure, we run Algorithm 2 on a sequence of  $N_{\text{true}} = 10,000$  randomly generated pulses under different SNR levels. The intervals between consecutive pulses are fixed to be 6,000 samples. It is noteworthy that, in contrast to training the classification net, we generated the pulse widths in the fixed range  $[0.1 \mu s, 400 \mu s]$  regardless of the SNR level.

Let us denote the list of ground-truth TOAs and TODs by  $\{(a_i, d_i)\}_{i=1}^{N_{\text{true}}}$  and the list of estimated TOAs and TODs by  $\{(\hat{a}_i, \hat{d}_i)\}_{i=1}^{N_{\text{est}}}$ . A pulse  $(a_i, d_i)$  is called *detected* if there exists  $j \in \{1, 2, \dots, N_{\text{est}}\}$  such that

$$|a_i - \hat{a}_j| < 200 \text{ ns}. \quad (6)$$

By a renumbering, suppose the set of detected pulses is  $\{(a_i, d_i)\}_{i=1}^{N_{\text{det}}}$ , which is matched by the subset  $\{(\hat{a}_i, \hat{d}_i)\}_{i=1}^{N_{\text{det}}}$  of estimated pulses. The remaining pulses,  $\{(\hat{a}_i, \hat{d}_i)\}_{i=N_{\text{det}}+1}^{N_{\text{est}}}$ , are considered false alarms. The detection performance of the proposed algorithm is then measured by 4 numbers: the detection rate, the F1 score, the TOA mean absolute error (MAE), and the TOD MAE. In particular, these parameters are computed as

$$\text{Detection\_Rate} = \frac{N_{\text{det}}}{N_{\text{true}}}, \quad (7)$$

$$\text{F1\_Score} = \frac{2N_{\text{det}}}{N_{\text{true}} + N_{\text{est}}}, \quad (8)$$

$$\text{TOA\_MAE} = \frac{1}{N_{\text{det}}} \sum_{i=1}^{N_{\text{det}}} |a_i - \hat{a}_i|, \quad (9)$$

$$\text{TOD\_MAE} = \frac{1}{N_{\text{det}}} \sum_{i=1}^{N_{\text{det}}} |d_i - \hat{d}_i|. \quad (10)$$

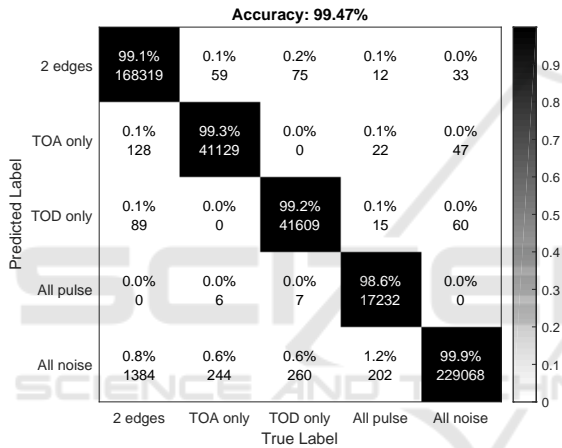
Note that the detection rate measures the sensitivity of the algorithm while the F1 score balances the true detection rate and the false alarm rate.

As a baseline, we also implemented the Threshold-based Edge Detection (TED) algorithm (Iglesias et al., 2014) in Matlab with some modifications to optimize the performance for the data of interest. The variance of the noise was estimated using the first 2000 samples of the signal, which were known to be all of noise. The detection performances of the two algorithms on the same set of testing signals under various SNR levels are reported in Table 1. It can be seen that the proposed method significantly outperforms TED in all measures for low SNR levels. For example, in the fairly noisy case when  $\text{SNR} = 6 \text{ dB}$ , the proposed scheme boosts the F1 score from 46.87% to 85.03%. In high-SNR regimes, we are on par with TED in terms of the detection rate and the F1 score while surpass TED in

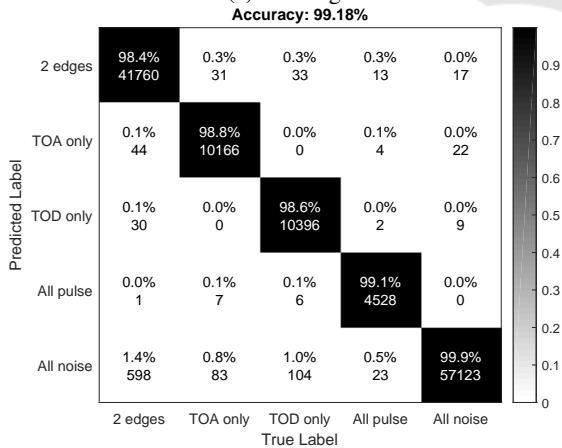


Table 1: Performance comparison between the proposed method and the Threshold-based Edge Detection (TED) for different SNR levels at the bandwidth of 20 MHz.

		Detection Rate	F1 Score	TOA MAE (ns)	TOD MAE (ns)
SNR = 15 dB	TED	99.14%	<b>99.51%</b>	19	27
	Ours	<b>99.24%</b>	99.43%	<b>10</b>	<b>16</b>
SNR = 12 dB	TED	<b>98.37%</b>	<b>99.03%</b>	22	35
	Ours	97.99%	98.72%	<b>16</b>	<b>19</b>
SNR = 9 dB	TED	89.62%	92.52%	42	135
	Ours	<b>92.19%</b>	<b>95.14%</b>	<b>28</b>	<b>45</b>
SNR = 6 dB	TED	52.81%	46.87%	75	2538
	Ours	<b>78.39%</b>	<b>85.03%</b>	<b>44</b>	<b>180</b>
SNR = 3 dB	TED	9.99%	5.13%	92	13791
	Ours	<b>53.29%</b>	<b>62.66%</b>	<b>59</b>	<b>1331</b>
SNR = 0 dB	TED	2.23%	1.84%	100	17404
	Ours	<b>24.79%</b>	<b>32.72%</b>	<b>75</b>	<b>3539</b>



(a) Training



(b) Testing

Figure 6: Confusion matrices of the classification net.

the other 2 measures. Fig. 7 visualizes the estimation of TOAs and TODs in some parts of the testing signal for SNR = 6 dB.

## 6 CONCLUSION

We have presented a deep-learning-based approach to the detection of radar pulses with various waveforms over a wide range of SNRs. The proposed scheme combines a classical find-change-points algorithm with a convolutional neural network for segment classification. Informally speaking, the classification net plays a guiding role to facilitate the find-change-points routine. Experiments on simulated data suggest that our method strikingly outperforms the Threshold-based Edge Detection (TED) algorithm, especially in low-SNR regimes. Another advantage of the proposed method is its noise-blindness, as opposed to TED which heavily relies on a noise floor estimation. The shortcoming of our method, however, is its costly computations, while TED can be implemented in FPGA for a real-time system. For the moment, the running time of the algorithm is about 1  $\mu$ s/sample, which is still far from the real-time target, 12.8 ns/sample. Future work would, therefore, focus on reducing the computational cost of the algorithm via a compression of the classification net like what have been done in (Han et al., 2016), as well as a more efficient implementation of the find-change-points procedure. Another potential direction would be replacing the find-change-points algorithm with a deep neural network.

## REFERENCES

- Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.
- Han, S., Mao, H., and Dally, W. J. (May 02-04, 2016). Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman

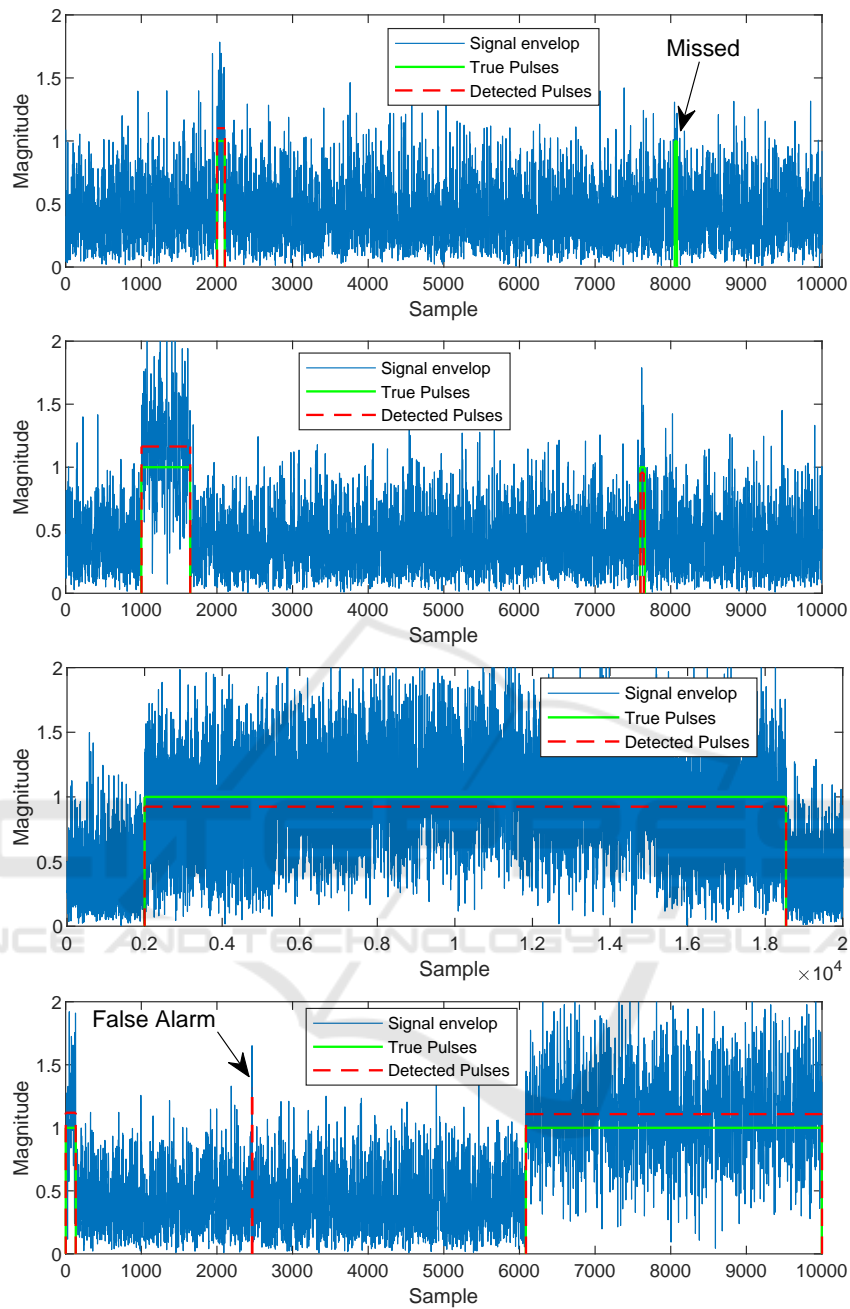


Figure 7: Detection results in different parts of the noisy signal with SNR = 6 dB at bandwidth 20 MHz. For each detected pulse, the pulse amplitude is estimated by averaging the signal magnitudes in between the estimated TOA and TOD. Misses and false alarms are marked with arrows.

coding. In *Proc. Int. Conf. Learning Representations (ICLR)*, pages 1–14, Puerto Rico.

He, K., Zhang, X., Ren, S., and Sun, J. (Jun. 27-30, 2016). Deep residual learning for image recognition. In *Proc. Conf. Comput. Vis. Pattern Recogn. (CVPR)*, pages 770–778, Las Vegas, NV, USA.

Iglesias, V., Grajal, J., Yeste-Ojeda, O., Garrido, M., Sánchez, M. A., and López-Vallejo, M. (May 19-23, 2014). Real-time radar pulse parameter extractor. In

*Proc. IEEE Radar Conf.*, pages 1–5.

Ioffe, S. and Szegedy, C. (Jul. 06-11, 2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proc. Int. Conf. Machine Learning (ICML)*, pages 1097–1105, Lille, France.

Jackson, B., Scargle, J. D., Barnes, D., Arabhi, S., Alt, A., Gioumoussis, P., Gwin, E., Sangtrakulcharoen, P., Tan, L., and Tsai, T. T. (2005). An algorithm for optimal

- partitioning of data on an interval. *IEEE Signal Process. Lett.*, 12(2):105–108.
- Killick, R., Fearnhead, P., and Eckley, I. A. (2012). Optimal detection of changepoints with linear computational cost. *J. Am. Stat. Assoc.*, 107(500):1590–1598.
- Krizhevsky, A., Sutskever, I., and Hinton, G. E. (Dec. 03-08, 2012). Imagenet classification with deep convolutional neural networks. In *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, pages 1097–1105, Lake Tahoe, NV, USA.
- LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521:436–444.
- Levanon, N. and Mozeson, E. (2004). *Radar Signals*. Wiley-Interscience.
- Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (Oct. 08-16, 2016). SSD: Single shot multibox detector. In *Proc. Eur. Conf. Comput. Vis. (ECCV)*, pages 21–37, Amsterdam, The Netherlands.
- Pace, P. E. (2009). *Detecting and Classifying Low Probability of Intercept Radar*. Artech House, 2 edition.
- Poisel, R. (2005). *EW Target Location Methods*. Artech House.
- Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (Jun. 27-30, 2016). You Only Look Once: Unified, real-time object detection. In *Proc. Conf. Comput. Vis. Pattern Recogn. (CVPR)*, pages 779–788, Las Vegas, NV, USA.
- Ren, S., He, K., Girshick, R., and Sun, J. (2017). Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Trans. Pattern Anal. Machine Intell.*, 39(6):1137–1149.
- Richards, M. A. (2014). *Fundamental of Radar Signal Processing*. McGraw-Hill Education, 2 edition.
- Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (Jun. 07-12, 2015). Going deeper with convolutions. In *Proc. Conf. Comput. Vis. Pattern Recogn. (CVPR)*, pages 1–9, Boston, MA, USA.
- Torrieri, D. J. (1974). Arrival time estimation by adaptive thresholding. *IEEE Trans. Aerospace Electro. Systems*, AES-10(2):178–184.
- Torrieri, D. J. (1984). Statistical theory of passive location systems. *IEEE Trans. Aerospace Electro. Systems*, AES-20(2):183–198.