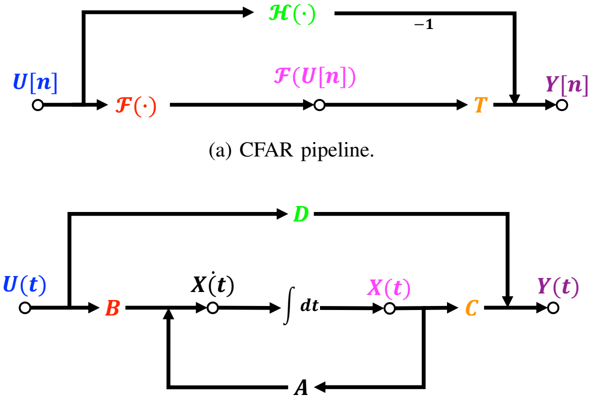
Unfolding Target Detection with State Space Model

Luca Jiang-Tao Yu
Department of Computer Science
The University of Hong Kong
lucayu@connect.hku.hk

Chenshu Wu
Department of Computer Science
The University of Hong Kong
chenshu@cs.hku.hk

***Abstract*—Target detection is a fundamental task in radar sensing, serving as the precursor to any further processing for various applications. Numerous detection algorithms have been proposed. Classical methods based on signal processing, *e.g.*, the most widely used CFAR, are challenging to tune and sensitive to environmental conditions. Deep learning-based methods can be more accurate and robust, yet usually lack interpretability and physical relevance. In this paper, we introduce a novel method that combines signal processing and deep learning by unfolding the CFAR detector with a state space model architecture. By reserving the CFAR pipeline yet turning its sophisticated configurations into trainable parameters, our method achieves high detection performance without manual parameter tuning, while preserving model interpretability. We implement a lightweight model of only 260K parameters and conduct real-world experiments for human target detection using FMCW radars. The results highlight the remarkable performance of the proposed method, outperforming CFAR and its variants by 10× in detection rate and false alarm rate. Our code is open-sourced here: `https://github.com/aiot-lab/NeuroDet`.**

***Index Terms*—Target Detection, Machine Learning in Signal Processing**



(a) CFAR pipeline.

(b) Continuous model pipeline of the state space model. The $\mathbf{dt}$ means the derivation of continuous step.

Figure 1: Comparison of CFAR and continuous state space model. Colors show the correspondence relationship between them.

I. Introduction

Detection is commonly recognized as the holy grail and a prelude task for most RF sensing applications, such as localization and tracking. So far, various detection algorithms have been proposed, *e.g.*, improved matched filter [1], Bayesian detector [2], *etc.*, but most of them suffer from various limitations. For example, the matched filter requires knowing a reference signal a priori, which is frequently not available, *e.g.*, in the presence of non-cooperative targets. The Bayesian detector also requires priority distributions and is sensitive to the environment.

The CFAR detector is widely used for its adaptive thresholding and clutter suppression, maintaining a constant false alarm rate without requiring special signal distribution or prior knowledge. It evaluates the noise level of neighboring cells around the cell under test (CUT) to determine the threshold for detection. Among its many variants, Cell Averaging CFAR (CA-CFAR) is simple and computationally efficient but can be sensitive to non-uniform noise distributions. To address this, Ordered Statistics CFAR (OS-CFAR) [3] is introduced to improve performance in heterogeneous environments, yet it increases computation due to the ordering operation. Other variants, like the Greatest Of CFAR (GO-CFAR) [4] and the Smallest Of CFAR (SO-CFAR) [5], aim to improve the detection in different scenarios. More advanced methods, such as adaptive [6], [7] and variation index CFAR [8], combine multiple measurement techniques at the cost of higher computational complexity. Additionally, these approaches all require careful parameter selection with extensive domain knowledge, and an improper configuration may lead to degraded performance.

In response to these drawbacks, machine learning (ML) techniques have been introduced to improve CFAR detection. Some researchers replace the measurement of CFAR with a Support Vector Machine (SVM) to select the best scheme [9]. However, the function of the SVM selector is limited because it can only choose the variants between CA-CFAR and GO-CFAR. Supervised deep learning is exploited in [10] and [11] to enable better detection. A recent work CFARnet [12] proves that CFAR can be maintained within a deep learning framework while providing computational efficiency and flexibility. Importantly, although these deep learning-based methods can improve the target detection performance, by incorporating prior knowledge in the trained models, they mostly leverage black-box neural networks and lack interpretability, raising concerns in RF sensing applications that are tightly coupled with the physical environment.

In this paper, inspired by recent advances in neural architectures based on state space models [13], we present a novel signal processing-guided deep learning design that features the advantages of both approaches while overcoming their respective drawbacks. Specifically, our method unfolds the CFAR algorithm by devising a trainable network architecture that strictly follows the CFAR processing pipeline. We conduct experiments on CNNs, RNNs, and CFAR variants. The results show that our method achieves approximately 10× higher detection rate at the same false alarm rate and 10× lower false alarm rate at the same detection rate than traditional CFAR variants while demonstrating significantly lower complexity compared to previous neural detectors based on CNNs and RNNs. Our approach also shows strong generalization ability to unseen datasets, thanks to the interpretable architecture.

II. Methodology

A. CFAR Pipeline

Denote the input sequence as $U[n]$, the CUT as $\bar{u}$, and the adjacent guard cells u_g. The sequences will pass by two different functions $\mathcal{F}(\cdot)$ and $\mathcal{H}(\cdot)$. The former works as *selecting operator*, varying depending on the categories of CFAR, like average for CA-CFAR and selection after sorting for OS-CFAR. The later performs as *testing operator*, normally a linear function, $\mathcal{H}(U[n]) = \bar{u}$. Then, the selected value $\mathcal{F}(U[n])$ will multiply the pre-defined threshold $\mathbf{T}$ (according to the constant false alarm rate), and subtract the tested value $\mathcal{H}(U[n])$ to get the output:

$$Y[n] = \mathbf{T} \cdot \mathcal{F}(U[n]) - \mathcal{H}(U[n]). \tag{1}$$

The output $Y[n]$ can be regarded as a sequence with the same length as the input after the sliding window, with the binary values representing the detection results for each cell. Albeit being efficient and scalable, the performance of CFAR detectors is sensitive to pre-configurations, *e.g.*, selecting and testing operators, false alarm rate, and fixed sliding window size. Optimizing these parameter settings requires comprehensive domain knowledge of radar signals, and improper configurations may significantly degrade the performance.

B. Network Model

Our design is inspired by the linear state space model from control systems and recent advances in trainable linear state space models [13]. By incorporating activation functions into the trainable linear state space model, we can introduce nonlinearity into the model. Consequently, this allows for the design of an interpretable unfolding detector based on the state space model.

The linear state space model maps the input sequence to the output sequence $U(t) \mapsto Y(t)$ via implicit state $X(t)$ by simulating a linear continuous-step state space representation in discrete step [13]. The state space representation can be viewed as the inclusion of unobservable variables, providing more information about the system's internal properties [14]. Considering the continuous model first:

$$\dot{X}(t) = \mathbf{A}X(t) + \mathbf{B}U(t), \tag{2}$$

$$Y(t) = \mathbf{C}X(t) + \mathbf{D}U(t), \tag{3}$$

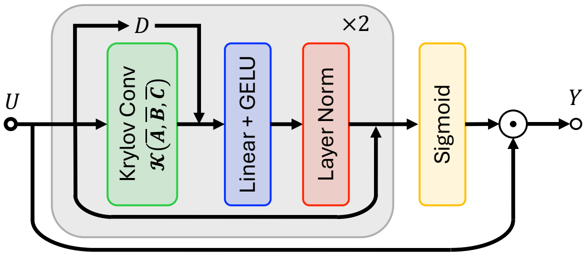


Figure 2: Model architecture.

where $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, and $\mathbf{D}$ are state, control, output, and command matrix. We define $\mathcal{F}(U(t))$ as the states $X(t)$, meaning that the output of *selecting operator* contains implicit information and $\mathbf{T}$ as the output matrix $\mathbf{C}$. Additionally, $\mathcal{H}(U(t))$ can be expressed as the matrix multiplication. Obviously, in Fig. 1, each part of CFAR has a correspondence relationship in the linear state space model. After adapting CFAR to the linear state space model, we need to discretize the model so that it can perform iterative training on the computer. We rewrite Eqn. (2) into an ordinary differential equation (ODE) $\dot{X}(t) = f(t, X(t))$ has an equivalent integral equation $X(t) = X(t_0) + \int_{t_0}^{t} f(s, X(s))ds$.

Function approximation. According to *Picard–Lindelöf theorem* [15], given an initial value problem, we can solve the problem by storing approximation for $X(t)$, and keeping the integral format fixed when iterating given initial function $X_0(t)$ and converge to $X(t)$, defined by:

$$X^{(0)}(t) = X_0(t), \tag{4}$$

$$X^{(\ell)}(t) = X_0(t) + \int_{t_0}^{t} f(s, X^{(\ell-1)}(s))ds, \tag{5}$$

which means the approximations of the ODE are a sequence of Picard iterates $\{X^{(0)}(t), X^{(1)}(t) \dots\}$. When Eqn. (5) in the ℓ-th iteration, the integral will hold the previous estimate of $X^{(\ell)}(t)$ fixed.

Step approximation and discretization. Meanwhile, we need to calculate numerical integration on the right-hand side of Eqn. (5) for each function approximation iteration, which can be converted to calculate the series of discrete-step values $\{X^{(\ell)}(t_0), X^{(\ell)}(t_1), \dots\}$:

$$X^{(\ell)}(t_j) = X^{(\ell)}(t_{j-1}) + \int_{t_{j-1}}^{t_j} f(s, X^{(\ell)}(s))ds, \tag{6}$$

then we apply the *Trapezoidal rule* [16] to approximate the integral. Given step size $\Delta t_j = t_j - t_{j-1}$:

$$\begin{aligned} X^{(\ell)}(t_j) - X^{(\ell)}(t_{j-1}) &= \int_{t_{j-1}}^{t_j} f(s, X^{(\ell)}(s))ds \\ &\approx \frac{\Delta t_j}{2}[f(t_j, X^{(\ell)}(t_j)) + f(t_{j-1}, X^{(\ell)}(t_{j-1}))]. \end{aligned} \tag{7}$$

Considering $f(t, X(t)) = \mathbf{A}X(t) + \mathbf{B}U(t)$, the integral treats $U(t)$ as a fixed value, therefore, we define $U(t_j) =$

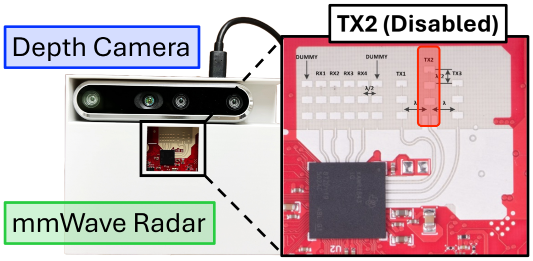


Figure 3: Layout of the sensing node contains a TI FMCW mmWave radar and an Intel RealSense depth camera.

$\frac{1}{\Delta t_j}\int_{t_{j-1}}^{t_j} U(s)ds$. Substituting Eqn. (2) into Eqn. (7) yields:

$$\begin{aligned} X^{(\ell)}(t_j) - X^{(\ell)}(t_{j-1}) &= \int_{t_{j-1}}^{t_j} (\mathbf{A}X(s) + \mathbf{B}U(s))ds \\ &\approx \frac{\Delta t_j}{2}\mathbf{A}(X^{(\ell)}(t_j) + X^{(\ell)}(t_{j-1})) + \Delta t_j \mathbf{B}U(t_j), \end{aligned} \tag{8}$$

which means the discrete $X^{(\ell)}(t_j)$ updates as:

$$\begin{aligned} X^{(\ell)}(t_j) =& (\mathbf{I} - \frac{\Delta t_j}{2}\mathbf{A})^{-1}(\mathbf{I} + \frac{\Delta t_j}{2}\mathbf{A})X^{(\ell)}(t_{j-1}) \\ &+ (\mathbf{I} - \frac{\Delta t_j}{2}\mathbf{A})^{-1}\Delta t_j \mathbf{B}U(t_j). \end{aligned} \tag{9}$$

We redefine $X^{(\ell)}(t_j) = \overline{\mathbf{A}}X^{(\ell)}(t_{j-1}) + \overline{\mathbf{B}}U(t_j)$ and $X^{(\ell)}(t_j)$ and $U(t_j)$ as the discrete sequence notation $X^{(\ell)}[n]$ and $U[n]$, so the Picard iterative discrete linear state model turns:

$$X^{(\ell)}[n] = \overline{\mathbf{A}}X^{(\ell)}[n-1] + \overline{\mathbf{B}}U[n], \tag{10}$$

$$Y[n] = \mathbf{C}X^{(\ell)}[n] + \mathbf{D}U[n], \tag{11}$$

where $X^{(\ell)}[n]$ converges to $X[n]$ after Picard iteration. The proposed approach can be viewed as a convolution and step size Δt_j can control the width of the convolutional kernel, which would be automatically learned when training [13], meaning that the size of convolution window input can be trainable.

C. Training Details

The input $U \in \mathbb{R}^L$, where L is the length of the flattened input sample and the output of each module is $Y \in \mathbb{R}^L$. The state matrix is $X \in \mathbb{R}^{N \times L}$, where N is the feature dimension of the state matrix. Therefore, the other matrices in the model are $\mathbf{A} \in \mathbb{R}^{N \times N}$, $\mathbf{B} \in \mathbb{R}^N$, $\mathbf{C} \in \mathbb{R}^N$, and $\mathbf{D} \in \mathbb{R}$. Simply, we define the initial state $X[-1] = 0$ and recursively substitute Eqn. (10) into Eqn. (11) yields (omit the Picard iteration number):

$$Y[n] = \sum_{k=0}^{n} \mathbf{C}(\overline{\mathbf{A}})^k \overline{\mathbf{B}}U[n-k] + \mathbf{D}U[n]. \tag{12}$$

By introducing the *Krylov function* [13]:

$$\mathcal{K}(\mathbf{A}, \mathbf{B}, \mathbf{C}) = (\mathbf{C}\mathbf{A}^k\mathbf{B})_{k \in [L]}, \tag{13}$$

the output Y can be expressed as the non-circular convolution:

$$Y = \mathcal{K}(\overline{\mathbf{A}}, \overline{\mathbf{B}}, \mathbf{C}) * U + \mathbf{D}U, \tag{14}$$

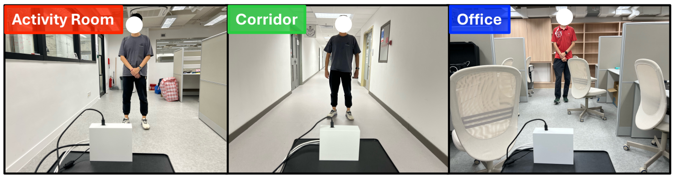


Figure 4: Data collection scenarios include activity room, corridor, and office.

where the convolution operation can be efficiently implemented using the Fast Fourier Transform to accelerate computation.

We construct our model by stacking two modules, incorporating linear transformation, layer normalization, and GELU [17] as the activation function to introduce non-linearity into the network, resulting in approximately 260K parameters. Before training, the input is cloned into H independent copies, enabling the multiple non-interacting training flows simultaneously. By applying $N = 256$ and $H = 256$, the outputs Y are averaged along the corresponding axis. To enhance training stability during iteration, we apply the translated Legendre (LegT) method to model the state matrices from the HiPPO framework [18]. Additionally, we introduce a residual connection from the input layer to the final layer, followed after a Sigmoid function, to improve the representative capacity of the stacked state space neural network. The model is trained using BCE loss. The model can be seen in Fig. 2.

III. Experiments

A. Experiments Setup

We focus on 2D range-azimuth data in our experiment. We take the range-azimuth spectrum of TI IWR1843BOOST FMCW mmWave radar as the input and use the point cloud of Intel RealSense D455 depth camera as the ground truth. We encapsulate them in a 3D-printed case to a sensing node for data collection, as illustrated in Fig. 3. The radar has a default azimuth angular resolution of 15°, and the frame length of the radar is set to 50 ms. The radar's maximum range is around 8.6 m and azimuth FoV is 120°. For the depth camera, the detectable depth is from 0.6 m to 6 m and the equivalent azimuth FoV is 87°.

As shown in Fig. 4, we consider both single-user and multi-user cases in different scenarios, including an activity room, corridor, and office. In total, we collect 2,222 samples, each including both the radar data and RGB & depth images of the depth camera. We perform range-FFT with 256 points, beamforming, and calibration on the radar data to calculate the range-azimuth spectrum. To label the ground truth of the spectrum, we utilize YOLO v8 [19] to detect the bounding box of the target, and then employ Segment Anything [20] to extract the target pixels within the bounding box on the RGB image. We then convert the corresponding pixels of the depth image into the range-azimuth point cloud image, which will be downsampled into a binary mask image (*i.e.*, 1 for

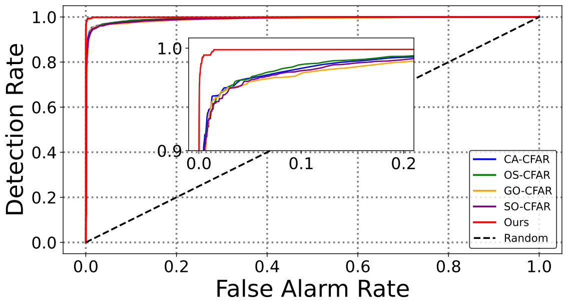


Figure 5: ROC curves of CFAR methods compared to our proposed method.

occupied cells and 0 otherwise) with the same size as the input spectrum. Due to the range and azimuth differences between the two sensors, we crop them to the overlapped field, resulting in a range of 0.6m to 6m and an azimuth from -43.5° to 43.5°.

B. Results

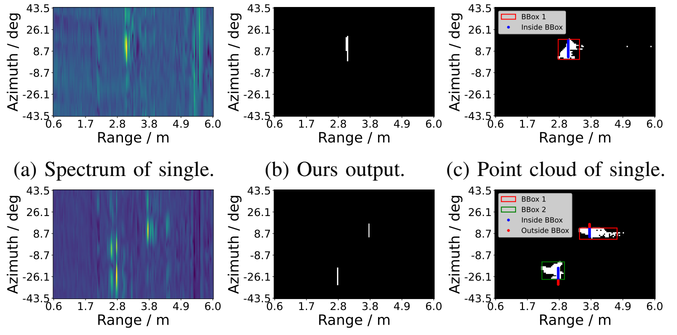
We use two metrics to evaluate our system: detection rate and false alarm rate. The detection rate ($\mathbf{p}_d$) is defined as the ratio of detected cells in the radar output that fall within the target's bounding box determined by YOLOv8 on the RGB image. The false alarm rate ($\mathbf{p}_f$) denotes how many cells are detected outside the bounding box, meaning there is no presence.

We implement our proposed method and compare it with various CFAR methods and CNN-based [21] and RNN-based [22] networks for comparison. For the trainable methods (*i.e.*, ours, CNNs, and RNNs), we train all models on the same dataset and evaluate both the detection and false alarm rate. For CFAR methods, we evaluate the detection rate and false alarm rate separately and tune the adjustable parameters of CFAR, ensuring that the other metric is maintained at the same level as in the proposed method.

Table I: Experimental results.

Methods	$\mathbf{p}_d$	$\mathbf{p}_f$
Ours	**92.60%**	**0.06%**
CNNs [21]	88.76%	0.17%
RNNs [22]	83.94%	0.46%
False Alarm Rate Fixed at the Same Level		
CA-CFAR	7.80%	0.06%
OS-CFAR	11.10%	0.06%
GO-CFAR	6.02%	0.06%
SO-CFAR	12.62%	0.06%
Detection Rate Fixed at the Same Level		
CA-CFAR	92.09%	1.08%
OS-CFAR	93.14%	1.40%
GO-CFAR	92.84%	1.51%
SO-CFAR	92.55%	1.53%

The results are presented in Tab. I. As seen, the CNN-based and RNN-based methods, benefiting from prior knowledge, outperform CFAR methods in both detection rate and false alarm rate. However, our approach achieves a higher detection rate and lower false alarm rate than both categories of existing approaches. Compared to CFAR methods, when the false alarm rate is fixed at 0.06%, CFAR cannot achieve a higher detection rate. Similarly, when the detection rate is fixed, CFAR's false alarm is, on average, 10 times higher than ours. Fig. 5 demonstrates the ROC curves comparing ours with CFAR methods, clearly showing the superior performance of our method. Additionally, as illustrated in the Tab. II, we conduct various evaluation experiments to confirm further the robustness and generalization of the proposed method, including the single and multiple targets scenarios and the unseen dataset. Moreover, our method is significantly more efficient, with only 260K parameters compared to 740K for the CNN-based model and 6M for the RNN-based model. The proposed method also offers better interpretability than these black-box neural networks.



(a) Spectrum of single. (b) Ours output. (c) Point cloud of single.

(d) Spectrum of multiples. (e) Ours output. (f) Point cloud of multiples.

Figure 6: Samples of Results. Along each row, the figures demonstrate the spectrum of the mmWave radar, the output of our method, and the point cloud of the depth camera.

Table II: Experimental results on different scenarios, including single and multiple targets, and unseen dataset.

Methods	Single		Multiple		Unseen	
	$\mathbf{p}_d$	$\mathbf{p}_f$	$\mathbf{p}_d$	$\mathbf{p}_f$	$\mathbf{p}_d$	$\mathbf{p}_f$
Ours	**94.72%**	**0.04%**	**86.89%**	**0.09%**	**91.92%**	**0.06%**
CNNs [21]	90.68%	0.17%	73.12%	0.18%	87.09%	0.17%
RNNs [22]	82.99%	0.47%	70.73%	0.41%	76.21%	0.46%

IV. Conclusion

This paper introduces an innovative target detection method that integrates CFAR's classical pipeline with the unfolding state space model, preserving the interpretability of traditional signal processing techniques. Our proposed approach utilizes two stacked modules with only 260K parameters. Our approach outperforms both CNN-based and RNN-based detectors. Compared to CFAR variants, the proposed method shows approximately a 10× improvement in detection rate at the same false alarm rate and a 10× reduction in false alarms at the same detection rate. These results highlight our remarkable performance in trainable and interpretable target detection.

References

[1] Pierre V Villeneuve, Herbert A Fry, James P Theiler, William B Clodius, Barham W Smith, and Alan D Stocker, "Improved matched-filter detection techniques," in *Imaging Spectrometry V*. SPIE, 1999, vol. 3753, pp. 278–285.

[2] Oktay Ureten, Nur Serinken, et al., "Bayesian detection of radio transmitter turn-on transients.," in *NSIp*, 1999, pp. 830–834.

[3] Gandhi and Kassam, "An adaptive order statistic constant false alarm rate detector," in *IEEE 1989 International Conference on Systems Engineering*. IEEE, 1989, pp. 85–88.

[4] V Gregers Hansen, "Constant false alarm rate processing in search radars. in radar—present and future," in *IEE Conf Publ*, 1973, vol. 105, p. 325.

[5] GV Trunk, BH Cantrell, and FD Queen, "Modified generalized sign test processor for 2-d radar," *IEEE Transactions on Aerospace and Electronic Systems*, , no. 5, pp. 574–582, 1974.

[6] Francesco Bandiera, Antonio De Maio, and Giuseppe Ricci, "Adaptive cfar radar detection with conic rejection," *IEEE Transactions on Signal Processing*, vol. 55, no. 6, pp. 2533–2541, 2007.

[7] Mohammad Ali Khalighi and Mohammad Hasan Bastani, "Adaptive cfar processor for nonhomogeneous environments," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 36, no. 3, pp. 889–897, 2000.

[8] Xinchao Zhu, Lingying Tu, Shun Zhou, and Zhengwen Zhang, "Robust variability index cfar detector based on bayesian interference control," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–9, 2021.

[9] Leiou Wang, Donghui Wang, and Chengpeng Hao, "Intelligent cfar detector based on support vector machine," *IEEE Access*, vol. 5, pp. 26965–26972, 2017.

[10] Jabran Akhtar and Karl Erik Olsen, "A neural network target detector with partial ca-cfar supervised training," in *2018 International Conference on Radar (RADAR)*. IEEE, 2018, pp. 1–6.

[11] Chia-Hung Lin, Yu-Chien Lin, Yue Bai, Wei-Ho Chung, Ta-Sung Lee, and Heikki Huttunen, "Dl-cfar: A novel cfar target detection method based on deep learning," in *2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*. IEEE, 2019, pp. 1–6.

[12] Tzvi Diskin, Yiftach Beer, Uri Okun, and Ami Wiesel, "Cfarnet: deep learning for target detection with constant false alarm rate," *Signal Processing*, vol. 223, pp. 109543, 2024.

[13] Albert Gu, Isys Johnson, Karan Goel, Khaled Saab, Tri Dao, Atri Rudra, and Christopher Ré, "Combining recurrent, convolutional, and continuous-time models with linear state space layers," *Advances in neural information processing systems*, vol. 34, pp. 572–585, 2021.

[14] Wing-Kuen Ling, *Nonlinear digital filters: analysis and applications*, Academic Press, 2010.

[15] Edward L Ince, *Ordinary differential equations*, Courier Corporation, 1956.

[16] Guofeng Zhang, Tongwen Chen, and Xiang Chen, "Performance recovery in digital implementation of analogue systems," *SIAM journal on control and optimization*, vol. 45, no. 6, pp. 2207–2223, 2007.

[17] Dan Hendrycks and Kevin Gimpel, "Gaussian error linear units (gelus)," *arXiv preprint arXiv:1606.08415*, 2016.

[18] Albert Gu, Tri Dao, Stefano Ermon, Atri Rudra, and Christopher Ré, "Hippo: Recurrent memory with optimal polynomial projections," *Advances in neural information processing systems*, vol. 33, pp. 1474–1487, 2020.

[19] Glenn Jocher, Ayush Chaurasia, and Jing Qiu, "Ultralytics YOLO," Jan. 2023.

[20] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al., "Segment anything," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 4015–4026.

[21] Faruk Yavuz, "Radar target detection with cnn," in *2021 29th European Signal Processing Conference (EUSIPCO)*. IEEE, 2021, pp. 1581–1585.

[22] Zachary Baird, Michael K Mcdonald, Sreeraman Rajan, and Simon J Lee, "A cnn-lstm network for augmenting target detection in real maritime wide area surveillance radar data," *IEEE Access*, vol. 8, pp. 179281–179294, 2020.