Article

Fast Target Localization in FMCW-MIMO Radar with Low SNR and Snapshot via Multi-DeepNet

Yunye Su 1, Xiang Lan 1*, Jinmei Shi 2, Lu Sun 3 and Xianpeng Wang 1,*

1 School of Information and Communication Engineering, Hainan University, Haikou 570228, China
2 College of Information Engineering, Hainan Vocational University of Science and Technology, Haikou 571158, China
3 Department of Communication Engineering, Institute of Information Science Technology, Dalian Maritime University, Dalian 116026, China
* Correspondence: wxpeng2016@hainu.edu.cn

Abstract: Frequency modulated continuous wave (FMCW) multiple-input multiple-output (MIMO) radars are widely applied in target localization. However, during the process, the estimation accuracy decreases sharply without considerable signal-to-noise ratio (SNR) and sufficient snapshot number. It is therefore necessary to consider estimation schemes that are valid under low signal-to-noise ratio (SNR) and snapshot. In this paper, a fast target localization framework based on multiple deep neural networks named Multi-DeepNet is proposed. In the scheme, multiple interoperating deep networks are employed to achieve accurate target localization in harsh environments. Firstly, we designed a coarse estimate using deep learning to determine the interval where the angle is located. Then, multiple neural networks are designed to realize accurate estimation. After that, the range estimation is determined. Finally, angles and ranges are matched by comparing the Frobenius norm. Simulations and experiments are conducted to verify the efficiency and accuracy of the proposed framework.

Keywords: deep learning; direction of arrival (DOA); range estimation; FMCW-MIMO radar; super-resolution estimation; localization method; low SNR and snapshot

1. Introduction

Frequency modulated continuous wave (FMCW) radar is widely used in civil fields, such as road vehicle monitoring and recording systems, car collision avoidance radars, traffic flow detectors and automatic driving systems, because of its easy implementation, simple structure, small size, light weight and low cost [1–3]. Researchers have also applied FMCW radar to micro-motion scenarios such as heartbeat detection and posture detection due to its excellent characteristics [4].

Recent studies have investigated multiple-input multiple-output (MIMO) FMCW radars, which pair the transmitter and receiver independently as virtual elements [5–7]. A large increase in array aperture is accompanied by an increase in data size. In addition, the received signal of the FMCW-MIMO radar contains DOA, speed and range information, which can be further used for target localization. In order to achieve this, the joint estimation of angle and range must be addressed. Since conventional algorithms, such as 2D-FFT, cannot obtain satisfactory performance due to the limitation of the Rayleigh criterion and the bandwidth of FMCW-MIMO radar, 2D-MUSIC is naturally applied to the joint angle and range estimation with the FMCW-MIMO radar [7,8]. In [9], J. Lee proposed a joint angle, speed and range estimation algorithm, where the 2D-MUSIC algorithm is applied to estimate both DOA and speed of the target. After that, a matched filter with the estimated angle and Doppler frequency as well as FFT operation is provided to estimate the range of the target. To further improve the performance, reference [10] proposed a gridless algorithm for joint DOA and range estimation. However, the estimation time required for 2D-MUSIC runs counter to the real-time requirements for radars today [11]. To enhance the speed of
the algorithm, the feasible sparse spectrum fitting (SpSF) algorithm for FDA-MIMO radar is proposed in [12]. In addition, in [13], a linear predictive-orthogonal propagator method (LP-OPM) based on forward–backward spatial smoothing (SS) is proposed for target localization. The DOA and range are estimated separately by the SS-LP-OPM algorithm, and then the parameters are matched by the least squares (LS) method. However when the SNR and snapshot are low, the above algorithm has large error, and the running time could be improved. Afterwards, two fast methods based on PARFAC decomposition were proposed in [14,15], but their performance is more general in the case of low snapshots and SNR.

Over the past few years, deep learning (DL) and machine learning (ML) have become increasingly popular research topics. Additionally, data-driven algorithms have been widely used in DOA estimation [16–22] with some excellent results. In [16,17], DOA estimation methods combining DL and MUSIC algorithm are proposed to improve estimation performance significantly over ordinary MUSIC algorithms. With low SNR, the DL-based DOA estimation algorithm can achieve better estimation performance than conventional algorithms [18]. In the non-ideal situation with some types of noise and defects, the DL-based DOA estimation algorithm [19] is more adaptable than other algorithms. As with the fast DOA estimation algorithm proposed in [20], DL-based DOA estimation algorithms often offer considerable runtime advantages as well. Furthermore, the DL-based algorithm proposed in [21] achieves fast and high-accuracy DOA estimation. Even though all of these DL-based algorithms are considered to be faster or better than traditional algorithms, they are not applicable to FMCW-MIMO radar. In 2021, Cong et al. proposed an algorithm in [22] that utilizes a very deep super-resolution (VDSR) neural network (NN) framework to speed up image shaping as a way to achieve faster joint angle and range estimation in FMCW-MIMO radar. However, the algorithm in [22] cannot handle the estimation in low SNR and snapshots scenarios, and the performance is not as good as that of the MUSIC grid with 0.1 degree.

In this paper, a fast target localization framework based on multiple deep neural networks is proposed. This framework achieves fast joint estimation in low SNR and snapshot through the cooperation of multiple NN with each other. The proposed framework is generally divided into two modules. In the first module, to address the high computational cost of traditional 2D-MUSIC algorithms for eigenvalue decomposition, we estimate DOA directly by using the DL framework, which avoids the large computational cost of traditional algorithms (e.g., MUSIC for eigenvalue decomposition and spectral peak search). In the second module, the ranges are estimated using the DL framework designed in this paper, which avoids the computational cost of range parameter estimation. In the above two modules, we improve the performance of parameter estimation beyond the traditional algorithms by using very deep neural networks and a large amount of training data. Finally, DOA and range are matched with each other by comparing the magnitude of the Frobenius norm to complete the joint DOA and range estimation. The simulation results demonstrate that the proposed algorithm has better performance than the traditional algorithm in the situation of low SNR and snapshot. Moreover, the simulation results show that the computational cost of the algorithm is much lower than that of the conventional algorithms in the abovementioned cases. The experimental results further verify the performance of the algorithm.

The main work and contributions of this paper are summarized as follows:

(1) A framework for fast target localization based on DNN is proposed. In spite of low SNR and snapshot, the framework is capable of quickly and accurately estimating the DOA and range of targets through FMCW-MIMO radars.

(2) Range and DOA within a data domain are matched using the proposed method in this paper. The separated DOA and range obtained by the proposed DL framework are matched by comparing the magnitude of the Frobenius norm.
To validate the framework, simulations and experiments were performed, and it has been demonstrated that the running time can be significantly reduced with high accuracy.

The remaining parts are organized in the following manner. An analysis of the data model and formulation of the problem is presented in Section 2. In Section 3, Multi-DeepNet, a framework based on multiple deep neural networks, is proposed. The matching method of DOA and range is described. The complete training strategy and detailed simulation results as well as experimental data are placed in Section 4. In Section 5, we conclude the paper.

The notations used in this paper are defined in Nomenclature.

2. Data Model

We performed follow-up work on Texas Instruments’ cascaded FMCW-MIMO radar systems (MMWAVCAS-RF-EVM and MMWAVCAS-DSP-EVM). As shown in Figure 1, this FMCW-MIMO radar has 16 receiving and 12 transmitting elements. With these array elements, the radar can form a large virtual array. However, in this paper, only a small fraction of the uniform linear array (ULA) of this FMCW-MIMO radar virtual array element (the first 10 arrays, as shown in Figure 2) are taken to achieve the case we envision.

![Figure 1. Texas Instruments’ cascaded FMCW-MIMO radar systems.](image)

From [7], the signal of the FMCW-MIMO radar transmitting by the transmitting array element to the receiving array element can be expressed as:

$$s(t) = e^{j(2\pi f_c t + \pi k_s t^2)}$$

(1)

where $f_c$ is the carrier frequency, and $k_s$ is the slope of the chirp. Assume that there are $K$ far-field narrow-band targets. Then, the signal received at the $i$-th receiving element can be expressed as:

$$r_i(t) = \sum_{j=1}^{K} \gamma_j s(t - \tau_{ij}) + n_i(t)$$

(2)

where $n_i(t)$ is the additive Gaussian white noise of the $i$-th receiving element, $\gamma_j$ is the complex reflection coefficient of the $j$-th target, and $\tau_{ij}$ is the time delay required for the
signal emitted from the transmitting elements to reach the $i$-th receiving element through the $j$-th target reflection. There is a direct relationship between $\tau_{ij}$ and the distance between the target and the FMCW-MIMO radar. The detail relationship is:

$$\tau_{ij} = \frac{2}{c} ( R_j + \frac{R_{ij}}{2} ) \approx \frac{2}{c} ( R_j + \frac{d_{0i}}{2} \sin \theta_j ) \quad (3)$$

where $c$, $R_j$ and $R_{ij}$ are the speed of light, the amount of distance between the $j$-th target and the transmitting element and amount of distance between the $j$-th target and the $i$-th receiving element; $d_{0i}$ is the distance between the $i$-th receiving element and the first receiving element (reference starting point); and $\theta_j$ is the DOA of the $j$-th target.

$$
\begin{align*}
\mathbf{R}_s &= \mathbf{X}\mathbf{X}^H / \mathbf{N} \\
\end{align*}
$$

where $\mathbf{N}$ is complex Gaussian noise with covariance $\sigma^2 \mathbb{I}_M$; $\mathbf{A} = [a_1, a_2, \ldots, a_K]$, $a_j = [1, e^{j2\pi f_c d \sin \theta_j}, \ldots, e^{j2\pi f_c (M-1)d \sin \theta_j}]^T, j = 1, 2, \ldots, K$; and $\mathbf{S} = [s_1, s_2, \ldots, s_K]$, $s_j = e^{j k \pi f_c R_j / c} e^{j k \pi d_{0i}}$.

3. Proposed Multi-DeepNet Architecture

The DL framework for estimating DOA, the DL framework for estimating range and the matching methods are presented separately in this section. The complete estimation process is shown in Figure 3.

The specific workflow of the proposed Multi-DeepNet is as follows. First, we obtain the sampling covariance matrix $\mathbf{R}_s$ from the received data $\mathbf{X}$.

$$
\mathbf{R}_s = \mathbf{X}\mathbf{X}^H / \mathbf{N} \\
$$

Figure 2. The selected elements and the remaining receiving elements.
Then, a three-channel tensor $T$ is formed by taking out the real part, imaginary part and phase of $R_s$, i.e.,

$$
T = [[[\text{Re}(R_s)]], [[[\text{Im}(R_s)]], [[[\text{Angle}(R_s)]]]]
$$

(7)

![Diagram](image)

**Figure 3.** The complete estimation process of Multi-DeepNet.

This tensor is then fed into the first-stage rough DOA estimation network. The first-stage rough DOA estimation network allows us to obtain the range ($10^\circ$ as an interval) of the angle grid. To enhance the performance at low SNRs, the network checks the results of the first-stage rough DOA estimation network. If the results do not satisfy the requirements, the low-SNR reinforcement network intervenes. Otherwise, the corresponding second-stage fine DOA estimation network is selected to be called based on the result of the first-order rough DOA estimation network. With the second-stage fine DOA estimation network, we can then obtain DOA estimates with a grid of $0.1^\circ$.

After obtaining the estimates of DOA, we obtain the transposed received signal covariance matrix as:

$$
RT_s = X^T(X^T)^H / N
$$

(8)

$$
T_T = [[[\text{Re}(RT_s)]], [[[\text{Im}(RT_s)]], [[[\text{Angle}(RT_s)]]]]
$$

(9)

Again, we stitch the real part, imaginary part and phase of $X^T$ to form a tensor $T_T$. The tensor is fed into the range estimation network, and with this distance estimation network we can directly obtain the range estimate with a grid of $0.1^\circ$.

Finally, we match the obtained DOA and range estimates to achieve joint angle and range estimation.

### 3.1. Introduction to the Multi-DeepNet Layers

According to [23], the 2D convolutional layer in the Multi-DeepNet can be described as follows:

$$
\text{Output} = \sum_{k=0}^{C_{out}-1} \text{Weight} \ast \text{Input} + \text{Bias}
$$

(10)

where $\text{Output} \in \mathbb{Q}^{bs \times C_{out}}$ is the output of the 2D convolutional layer, $bs$ is the batch-size, and $C_{out}$ is the number of out channels. $\text{Weight} \in \mathbb{Q}^{C_{out} \times k}$ is the weight of the kernel, and $\text{Bias} \in \mathbb{Q}^{C_{out}}$ is the bias of the 2D convolutional layer.
From [24], a dense layer, widely used in a variety of DNNs, can be derived as follows:

\[ O = W^T I + b \]  

(11)

where \( O \), \( W \), \( I \) and \( b \) are the output, weight, input and bias of the dense layer in the Multi-DeepNet.

In order to preserve the effect of negative numbers during 2D convolution, leakyReLU is used as the activation function behind the 2D convolution layer in the framework. According to [25,26], ReLU and leakyReLU can be described as:

\[
\text{ReLU}(x) = \begin{cases} 
  x, & x \geq 0, \\
  0, & x < 0 
\end{cases}
\]

(12)

\[
\text{leakyReLU}(x) = \begin{cases} 
  x, & x \geq 0, \\
  \alpha x, & x < 0 
\end{cases}
\]

(13)

where \( \alpha \) is preset leaky value, typically set to \( \alpha = 0.01 \).

Additionally, it is available from [27]. The batch normalization layer can be represented as follows:

\[
\text{BN}(x) = \frac{x - E[x]}{\sqrt{\text{Var}[x]} + \epsilon} \gamma + \beta
\]

(14)

where \( \gamma \) and \( \beta \) are learnable parameters. Depending on the output of the previous layer above the batch normalization layer, the \( x \) can be either a vector or a matrix.

3.2. The First-Stage DOA Rough Estimation Network Structure

In this part, as part of the DOA estimations for the interval \(-60^\circ\) to \(60^\circ\), 12 small intervals of \(10^\circ\) each are considered, the first-stage rough DOA estimation network structure is shown in Figure 4.

The input layer receives the \( \text{Tensor} \in \mathbb{R}^{10 \times 10 \times 3} \) assembled by the received data sampling covariance matrix, as in Equations (6) and (7), to extract the high-dimensional features of the sampled covariance matrix. This tensor will be passed through a 2D convolutional layer to accelerate the learning speed and strengthen the NN's ability to learn from the data. The above data is fed into the 2D batch normalization layer of the same scale after passing through the 2D convolutional layer. For the choice of activation function, we choose to use leaky RELU in order to avoid the disadvantage that the RELU function is too fragile on deep networks.
After passing through the three 2D convolutional layers, batch normalization layers and activation function layers, the data are unfolded by the flatten operation and fed into the dense layer. Then, after passing through the three dense layers, the probability of DOA occurrence in each interval is obtained. The detailed structural parameters of the first-stage rough DOA estimation network are shown in Table 1.

Table 1. Parameters of the first-stage rough DOA estimation network.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Activation</th>
<th>Learnables</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input Layer</td>
<td>Input</td>
<td>10 × 10 × 3</td>
<td>–</td>
</tr>
<tr>
<td>Cov.1</td>
<td>2D convolutional</td>
<td>10 × 10 × 256</td>
<td>Weights: 3 × 3 × 256 Bias: 1 × 1 × 256</td>
</tr>
<tr>
<td>BN.1</td>
<td>Batch Normalization</td>
<td>10 × 10 × 256</td>
<td>Offset: 1 × 1 × 256 Scale: 1 × 1 × 256</td>
</tr>
<tr>
<td>leakyReLU.1</td>
<td>LeakyReLU</td>
<td>10 × 10 × 256</td>
<td>–</td>
</tr>
<tr>
<td>Cov.2</td>
<td>2D convolutional</td>
<td>10 × 10 × 128</td>
<td>Weights: 3 × 3 × 128 Bias: 1 × 1 × 128</td>
</tr>
<tr>
<td>BN.2</td>
<td>Batch Normalization</td>
<td>10 × 10 × 128</td>
<td>Offset: 1 × 1 × 128 Scale: 1 × 1 × 128</td>
</tr>
<tr>
<td>leakyReLU.2</td>
<td>LeakyReLU</td>
<td>10 × 10 × 128</td>
<td>–</td>
</tr>
<tr>
<td>Cov.3</td>
<td>2D convolutional</td>
<td>10 × 10 × 64</td>
<td>Weights: 3 × 3 × 64 Bias: 1 × 1 × 64</td>
</tr>
<tr>
<td>BN.3</td>
<td>Batch Normalization</td>
<td>10 × 10 × 64</td>
<td>Offset: 1 × 1 × 64 Scale: 1 × 1 × 64</td>
</tr>
<tr>
<td>leakyReLU.3</td>
<td>LeakyReLU</td>
<td>10 × 10 × 64</td>
<td>–</td>
</tr>
<tr>
<td>Flatten</td>
<td>Flatten</td>
<td>1 × 1 × 6400</td>
<td>–</td>
</tr>
<tr>
<td>Dense.1</td>
<td>Dense</td>
<td>1 × 1 × 2048</td>
<td>Weights: 2048 × 6400 Bias: 2048 × 1</td>
</tr>
<tr>
<td>ReLU.1</td>
<td>ReLU</td>
<td>1 × 1 × 2048</td>
<td>–</td>
</tr>
<tr>
<td>Dense.2</td>
<td>Dense</td>
<td>1 × 1 × 512</td>
<td>Weights: 512 × 2048 Bias: 512 × 1</td>
</tr>
<tr>
<td>ReLU.2</td>
<td>ReLU</td>
<td>1 × 1 × 512</td>
<td>–</td>
</tr>
<tr>
<td>Dense.3</td>
<td>Dense</td>
<td>1 × 1 × 128</td>
<td>Weights: 128 × 512 Bias: 128 × 1</td>
</tr>
<tr>
<td>ReLU.3</td>
<td>ReLU</td>
<td>1 × 1 × 128</td>
<td>–</td>
</tr>
<tr>
<td>Dense.4</td>
<td>Dense</td>
<td>1 × 1 × 12</td>
<td>Weights: 12 × 128 Bias: 12 × 1</td>
</tr>
<tr>
<td>Output Layer</td>
<td>Classification</td>
<td>1 × 1 × 12</td>
<td>–</td>
</tr>
</tbody>
</table>

3.3. The Low-SNR Reinforcement Network Structure

The first-stage network is not always able to estimate the angle interval accurately when the SNR is low. It is for this reason that low-SNR reinforcement networks (LSR-net) are proposed. When the probability (maximum value) estimated by the network is less
than the threshold set initially, the LSR-net is invoked. Otherwise, the second-stage DOA estimation is performed directly.

The whole LSR-net is composed of six subnetworks, each of which has the same structure as that of the first-stage network, as shown in Figure 5. Finally, a combinatorial function is used to find the final probability distribution, which is denoted by

$$P_{LSR} = \sum_{k=1}^{6} P_k / 6 \quad (15)$$

where $P_{LSR} \in \mathbb{Q}_{1 \times 12}$ is the output of LSR-net, and $P_k \in \mathbb{Q}_{1 \times 12}$ is the output of $k$-th subnetwork. On the basis of $P_{LSR}$, we determine which second-stage networks should be invoked.

---

**Figure 5.** The low-SNR reinforcement network structure.

### 3.4. The Second-Order DOA Fine Estimation Network Structure

With the network of the two subsections mentioned above, we obtain the interval in which the angle is located. After obtaining the interval where the angle is located, the second-stage DOA fine estimation network is invoked for the corresponding interval. Thus, 12 structurally consistent subnetworks were designed, and the coarse estimation results were used to choose the subnetworks that call the intervals. The tensor is fed into the subnetwork as shown in Figure 6, and the final result is the peak value of the spectral peak in the corresponding interval, as shown in Figure 7. (The simulation in the figure is based on the $DOA = -14^\circ$, $Range = 5.0$ m, $SNR = 10$ dB.)
We optimized the corresponding structure for dense data based on the first-stage of the network. The parameters of the second-order DOA fine estimation network are specified in Table 2. The ... in Table 2 define leakyReLU and BN as shown in the tab leakyReLU.1 and BN.1.

The estimation of DOA is completed when the results of the second-order DOA fine estimation network are obtained.
Table 2. Parameters of second-stage fine DOA estimation network structure.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Activation</th>
<th>Learnables</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input Layer</td>
<td>Input</td>
<td>10 × 10 × 3</td>
<td>–</td>
</tr>
<tr>
<td>Input Size:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cov.1</td>
<td>2D convolutional</td>
<td>10 × 10 × 256</td>
<td>Weights: 5 × 5 × 256</td>
</tr>
<tr>
<td>Kernel Size:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 × 5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Filter number:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>256</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BN.1</td>
<td>Batch Normalization</td>
<td>10 × 10 × 256</td>
<td>Offset: 1 × 1 × 256</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>leakyReLU.1</td>
<td>LeakyReLU</td>
<td>10 × 10 × 256</td>
<td>–</td>
</tr>
<tr>
<td>leak: 0.3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cov.2</td>
<td>2D convolutional</td>
<td>10 × 10 × 128</td>
<td>Weights: 5 × 5 × 128</td>
</tr>
<tr>
<td>Kernel Size:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 × 5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Filter number:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>128</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cov.3</td>
<td>2D convolutional</td>
<td>10 × 10 × 64</td>
<td>Weights: 3 × 3 × 64</td>
</tr>
<tr>
<td>Kernel Size:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 × 3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Filter number:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>64</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cov.4</td>
<td>2D convolutional</td>
<td>10 × 10 × 32</td>
<td>Weights: 3 × 3 × 32</td>
</tr>
<tr>
<td>Kernel Size:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 × 3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Filter number:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flatten</td>
<td>Flatten</td>
<td>1 × 1 × 3200</td>
<td>–</td>
</tr>
<tr>
<td>Dense.1</td>
<td>Dense</td>
<td>1 × 1 × 512</td>
<td>Weight: 512 × 3200</td>
</tr>
<tr>
<td>Number of neurons:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>512</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ReLU.1</td>
<td>ReLU</td>
<td>1 × 1 × 512</td>
<td>–</td>
</tr>
<tr>
<td>Dense.2</td>
<td>Dense</td>
<td>1 × 1 × 121</td>
<td>Weight: 121 × 512</td>
</tr>
<tr>
<td>Number of neurons:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>121</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OutputLayer</td>
<td>Classification</td>
<td>1 × 1 × 121</td>
<td>–</td>
</tr>
<tr>
<td>loss: binary</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>crossentropy</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3.5. The Range Estimation Network Structure

For the range estimation, Equation (8) provides the covariance of the received data transposition from the FMCW-MIMO radar. From Equation (9), we can obtain Tensor\_T as the input for the range estimation network.

The interval of our considered range is from 1 m to 10 m, and the size of the grid is 0.1 m. The amount of data involved in range estimation is much smaller than that in the DOA estimation part. Therefore, in the range estimation part, we only design one deep neural network to implement fine range estimation.

Our proposed range estimation network is structurally similar to the first-stage rough DOA estimation network described above. Both are based on the aggregation of CNN and dense networks. The parameters of the detailed range estimation network are shown in Table 3.

With the range estimation network described above, a pseudo-spectrum of the range is obtained, and the spectral peak is the value where the maximum probability of the range occurrence is located.
### Table 3. Parameters of the range estimation network.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Activation</th>
<th>Learnables</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input Layer</td>
<td>Input</td>
<td>$10 \times 10 \times 3$</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>Input Size: $10 \times 10 \times 3$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cov.1</td>
<td>2D convolutional</td>
<td>$10 \times 10 \times 256$</td>
<td>Weights: $5 \times 5 \times 256$</td>
</tr>
<tr>
<td></td>
<td>Kernel Size: $5 \times 5$</td>
<td></td>
<td>Bias: $1 \times 1 \times 256$</td>
</tr>
<tr>
<td>Cov.2</td>
<td>2D convolutional</td>
<td>$10 \times 10 \times 128$</td>
<td>Weights: $4 \times 4 \times 128$</td>
</tr>
<tr>
<td></td>
<td>Kernel Size: $4 \times 4$</td>
<td></td>
<td>Bias: $1 \times 1 \times 128$</td>
</tr>
<tr>
<td>Cov.3</td>
<td>2D convolutional</td>
<td>$10 \times 10 \times 32$</td>
<td>Weights: $3 \times 3 \times 32$</td>
</tr>
<tr>
<td></td>
<td>Kernel Size: $3 \times 3$</td>
<td></td>
<td>Bias: $1 \times 1 \times 32$</td>
</tr>
<tr>
<td>Flatten</td>
<td>Flatten</td>
<td>$1 \times 1 \times 3200$</td>
<td>–</td>
</tr>
<tr>
<td>Dense.2</td>
<td>Dense</td>
<td>$1 \times 1 \times 91$</td>
<td>Weight: $91 \times 512$</td>
</tr>
<tr>
<td></td>
<td>Number of neurons: 91</td>
<td></td>
<td>Bias: $91 \times 1$</td>
</tr>
<tr>
<td>OutputLayer</td>
<td>Classification</td>
<td>$1 \times 1 \times 91$</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>loss: binary crossentropy</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 3.6. Matching Method

If there are multiple targets (multiple pairs of DOA and range), we need to consider the matching problem. From Equations (4) and (5), we know that DOA and range are coupled in the received signal. So in case of mismatch, the received signal $X$ will be dramatically different from the $X_c$ we constructed using the known DOA and range. However, in the case of correct pairing, the difference will be small.

Based on the above, we transform the matching problem into a problem of finding the minimum value. The pseudo-code is shown in Algorithm 1:

**Algorithm 1** Matching Method.

1. **Input:** The received data $X$; Estimated DOA and Range;
2. Obtain all combinations of DOA and Range;
3. **Initialization:** $\zeta = +\infty$, $\eta = 0$;
4. for $c$ in combinations do
5. Calculate $X_c$ according to (4-6), DOA$_c$ and Range$_c$;
6. Calculate $\eta = \|X - X_c\|_F$;
7. if $\eta < \zeta$ do
8. Update $\zeta = \eta$;
9. Update $\text{DOA}_M = \text{DOA}_c$, $\text{Range}_M = \text{Range}_c$;
10. end if
11. end for
12. **Output:** DOA$_M$ and Range$_M$;
4. Simulations and Experiments

In order to validate the proposed method, several simulations and experiments are conducted. According to Table 4, simulations and experiments have the same TI Cascade FMCW-MIMO radar parameters.

Table 4. Parameters of the FMCW radar.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c$</td>
<td>$3 \times 10^8$ m/s</td>
<td>$T_s$</td>
<td>125 ns</td>
</tr>
<tr>
<td>$f_c$</td>
<td>78.737692 GHz</td>
<td>$\lambda$</td>
<td>3.8 mm</td>
</tr>
<tr>
<td>$d$</td>
<td>1.9 mm</td>
<td>$M$</td>
<td>10</td>
</tr>
<tr>
<td>$k_s$</td>
<td>7.8986 MHz/μs</td>
<td>$N$</td>
<td>10</td>
</tr>
</tbody>
</table>


Cramer–Rao bounds, also known as the Cramer–Rao lower bounds (CRLB), are proposed for parameter estimation problems and establish lower bounds for the variance of any unbiased estimator. The variance of the unbiased estimator can only approach the Cramer–Rao bounds, also known as the Cramer–Rao lower bounds (CRLB), are obtained according to the Cramer criterion as

$$ CRB = \sqrt{\text{det}(I(\omega))} $$

where $\text{det}(I(\omega))$ is the determinant of the Fisher information matrix $I(\omega)$. Then, the covariance matrix $C_{\omega}$ is given by

$$ C_{\omega} = I^{-1}(\omega) $$

Then, the covariance matrix $C_{\omega}$ is given by

$$ C_{\omega} = I^{-1}(\omega) $$

where $\mu[m, n] = \text{Re}(x[m, n])$ and $v[m, n] = \text{Im}(x[m, n])$, $a, b = 1, 2$. Therefore, by setting $k[m, n] = 2\pi f_c (\frac{2R_j}{c} + \frac{m d \sin \theta_j}{c}) + 2\pi k_s (\frac{2R_j}{c} + \frac{m d \sin \theta_j}{c}) n T_s$, we can obtain:

$$ \frac{\partial \mu[m, n]}{\partial \omega_1} = -\frac{4\pi}{c} (f_c - k_s n T_s) \sin(k[m, n]) $$

$$ \frac{\partial \mu[m, n]}{\partial \omega_2} = -\frac{2\pi m}{c} (f_c - k_s n T_s) \sin(k[m, n]) $$

$$ \frac{\partial v[m, n]}{\partial \omega_1} = \frac{4\pi}{c} (f_c - k_s n T_s) \cos(k[m, n]) $$

$$ \frac{\partial v[m, n]}{\partial \omega_2} = \frac{2\pi m}{c} (f_c - k_s n T_s) \cos(k[m, n]) $$

The CRB of $\theta_j$ and $R_j$ can be obtained according to the Cramer criterion as

$$ CRB_{\theta_j} = \sqrt{|I^{-1}(\omega_j)|}_{1,1} $$

$$ CRB_{R_j} = \sqrt{|I^{-1}(\omega_j)|}_{2,2} \frac{d(d \sin \theta_j)}{d\theta_j} = \sqrt{|I^{-1}(\omega_j)|}_{2,2} / (d \cos \theta_j) $$
In the following subsections, we first describe the data generation, training environment and methods. Then, we compare several classical methods in the simulation of estimating DOA. After that, we compare several classical methods in the simulation of estimating range. Based on this, we compare the localization error of the algorithm. Next, we test the running speed of the algorithm. Finally, we apply the algorithm to experimental data.

4.2. Data Generation, Training Environment and Methods

The platform for the experiment is a PC with an Intel i7-12700H CPU 2.2 GHz, RTX 3070 laptop GPU and 16 GB of RAM. The experimental environment is Matlab 2022a and tensorflow 2.90. For the Multi-DeepNet optimizer we chose to use Adam with default parameters. The maximum value of epochs was set to 100, and the size of batch-size was set to 64. In the proposed Multi-DeepNet, training time was not taken into account since the training procedure was off-line.

To generate the training data of Multi-DeepNet, we assumed that the DOA of $n$ sources was located at $-60^\circ$ to $60^\circ$ and range was located at 1 m to 10 m. First, we randomly selected the number of incident signal sources $n$ between one to six, then we randomly selected n DOAs between intervals $-60^\circ$ to $60^\circ$, and n ranges between intervals 1 m to 10 m. Using the pair of DOAs and ranges generated above, we generated the corresponding radar received signal through Equations (4) and (5), and converted the received signal into the tensor received by the network through Equations (6)–(9). For the pair of DOA and range generated above, we first labeled the sector where the DOA was located, and then we labeled the pseudo-spectrum where the range was located. In this way, we can obtain a set of training data of the first-stage rough DOA estimation network $\{T, \text{Rough DOA}\}$ data and range estimation network $\{T, \text{Range}\}$ data. For the above pair of DOA and range, we will generate training data under different SNRs $-20$ dB to $20$ dB. At each SNR, we generated 500,000 sets of data.

Low-SNR reinforcement network training data are similar to the first-stage rough DOA estimation data. The difference is only in SNRs of training data. The data of low-SNR reinforcement network only considers $-20$ dB to 0 dB.

The training data for each sub-net of the second-order DOA fine estimation network were generated separately. For example, in sub-net 1, the DOA of our training data was only randomly generated between $-60^\circ$ and $-50^\circ$, and the generation method of range remained unchanged. Finally, 500, 000 sets of data were generated for each SNR between $-20$ dB and 20 dB.

When training the Multi-DeepNet, the corresponding training data was randomly divided into 90% training set and 10% validation set. The training history of the Multi-DeepNet is shown in Figure 8. In the figure, (a), (b), (c) and (d), respectively, describe the training history of the first-stage rough DOA estimation network, the low-SNR reinforcement network, the second-order DOA fine estimation network and the range estimation network. It can be seen from Figure 8 that validation losses and training losses were always close during training, and they coincided at the end, indicating no overfitting.

4.3. Simulations

Two far-field narrow-band fixed targets at ($-14.3^\circ$, 2.5 m) and ($17.8^\circ$, 7.5 m) were considered to verify the performance of the whole Multi-DeepNet. DOA and range were evaluated using the root mean square error (RMSE) metric. We define the RMSE equations for DOA, range and location separately as follows:

$$RMSE_{\text{DOA}} = \sqrt{\frac{1}{MK} \sum_{m=1}^{M} \sum_{k=1}^{K} (\theta_k - \hat{\theta}_{km})^2}$$  (25)
\[ \text{RMSE}_{\text{Range}} = \sqrt{\frac{1}{M \sum_m} \sum_{k=1}^{K} (R_k - \hat{R}_{k,m})^2} \]  

\[ \text{RMSE}_{\text{Location}} = \sqrt{\frac{1}{M \sum_m} \sum_{k=1}^{K} \left( \left\| (x_k, y_k) - (\hat{x}_{k,m}, \hat{y}_{k,m}) \right\|_2 \right)^2} \]  

where \( \theta_k \) and \( R_k \) are the \( k \)-th true DOA and range, \( \hat{\theta}_{k,m} \) and \( \hat{R}_{k,m} \) are the \( k \)-th estimated values of DOA and range in the \( m \)-th Monte Carlo trial, \( (x_k, y_k) \) is the \( k \)-th target actual location, and \( (\hat{x}_{k,m}, \hat{y}_{k,m}) \) is the \( k \)-th estimated values of location in the \( m \)-th Monte Carlo trial. We represent DOA and range in a right-angle coordinate system as shown in the following equation:

\[
\begin{align*}
    x_k &= R_k \sin \theta_k, \\
    y_k &= R_k \cos \theta_k, \\
    \hat{x}_{k,m} &= \hat{R}_{k,m} \sin \hat{\theta}_{k,m}, \\
    \hat{y}_{k,m} &= \hat{R}_{k,m} \cos \hat{\theta}_{k,m}
\end{align*}
\]  

\[ (28) \]

**Figure 8.** Training history of Multi-DeepNet.

We set the number of Monte Carlo experiments \( M \) to 200. In addition, we performed a runtime comparison of various algorithms as a way to demonstrate the advantages of real-time performance of the Multi-DeepNet.

We adopted several traditional DOA estimation methods to compare with the proposed Multi-DeepNet in the simulation setup of DOA estimation. Examples are MUSIC with Grid 0.1° [5,6], the OPM algorithm [29], the TLS-ESPRIT algorithm [10] and the ROOT-MUSIC algorithm [30]. As shown in Figure 9, we set \( \text{DOA} = [-13.77°, 13.32°] \) and \( \text{Range} = [2.72 \text{ m}, 8.72 \text{ m}] \), deliberately placing the DOA outside the grid, with some grid error. When the SNR increases, the Multi-DeepNet has better performance than other algorithms.
In the simulation of range estimation, we chose the same DOA and range of the above simulation to generate the simulation data. Since range estimation involves fewer algorithms, we only compared the performance differences among MUSIC with Grid 0.1°, OPM and Multi-DeepNet. As shown in Figure 10, it can be seen that the RMSE curve of Multi-DeepNet at −10 dB to 20 dB SNR was always located below the RMSE curve of the other two algorithms.

In the joint estimation, we generally converted DOA and range into a right-angle coordinate system for the comparison of RMSE as in Equation (28). We used MUSIC with Grid 1°, MUSIC with Grid 0.1°, OPM, and Multi-DeepNet for the comparison as shown in Figure 11. The simulation showed that Multi-DeepNet has a clear advantage in the SNR range of −20 dB to 20 dB. Multi DeepNet can achieve better performance than other algorithms under low SNR ratio due to the combination of reinforcement training under low SNR and multiple low-SNR reinforcement networks.
Other than the above simulations, the success rate of the algorithm localization was tested in this paper. The success rate can be defined as the ratio of the number of Monte Carlo simulations to the number of times the difference between the localized result and the true value is less than a threshold. In this simulation, the threshold was set to 0.5 and the number of Monte Carlo simulations was 200. In the simulation result, Figure 12, the success rate of the algorithm was acceptable at a low signal-to-noise ratio. The success rate at high signal-to-noise ratio also had a good level.

In addition, the simulation in Figure 13 shows the running time comparison, and we can clearly see that Multi-DeepNet has a much lower time requirement than other algorithms. Because the modules formed by Multi DeepNet are all simple matrix operations, they are faster than other algorithms.
4.4. Experiments

To fix the poor calibration in [22], a calibration matrix certified in a darkroom environment was used. The experimental data were obtained from the TI Cascade FMCW-MIMO Radar shown in Figure 1. The location of the experiment was a microwave darkroom, and the detection target of the experiment was the metal corner reflector. We measured two different scenarios throughout the experiment. The two scenarios are shown in Figure 14: target set at \((0.7°, 4.5\, \text{m})\) in Scenario 1 and target set at \((-6.4°, 5.2\, \text{m})\) and \((4.7°, 5.3\, \text{m})\) in Scenario 2. The locationing effect was selected as a performance metric in this part of the study. In the experiment, the SNR of Scenario 1 was 20 dB, and that of Scenario 2 was 15 dB.

The estimated results of the Multi-DeepNet on the measured data are shown. When the SNR increases, the Multi-DeepNet still has better performance than other algorithms shown in Figures 15 and 16, respectively. Figure 15 is the single target scenario: (a) is the estimated result of the first-stage rough DOA estimation network, (b) is final DOA estimation result, and (c) is the estimated range. The predicted DOA and range can be obtained directly by observing the output value of the Multi-DeepNet.

Figure 16 shows the results of the two target scenarios: (a) is the rough DOA estimation result, (b) and (c) are final DOA estimation results, and (d) is the estimated range. In the following section, we compare the results of Multi-DeepNet with that of other algorithms.

Figures 17 and 18 show the comparison of the localization results of the proposed algorithm with 2D-MUSIC, OPM and SS-LP-OPM on experimental data. It can be seen that several algorithms have good performance on the experimental data, and there is no
significant difference with the real position. As can be seen from the two experimental graphs, the proposed localization framework is practically feasible.

![Graph](image)

**Figure 15.** DOA and range estimation result of Scenario 1.

![Graph](image)

**Figure 16.** DOA and range estimation of Scenario 2.
Figure 17. Localization result of Scenario 1.

Figure 18. Localization result of Scenario 2.

5. Conclusions

In this paper, we have proposed a fast joint DOA and range estimation framework based on deep learning for target localization. With the proposed algorithm, better estimation performance with low signal-to-noise ratio and fewer snapshots can be achieved while reducing the required running time. The simulations show that the proposed algorithm has significant advantages in the case of low SNR and low snapshot, and satisfactory performance in the case of higher SNR. Finally, experimental data are also supplemented for verification. Although the proposed algorithm has good results, there are still some problems such as not having significant advantages at high SNR and the overall framework being too complex. Therefore, in subsequent work, the structure of the framework and network can be simplified to improve the performance under high SNR.

Author Contributions: Conceptualization, X.W. and L.S.; methodology, Y.S. and X.W.; writing—original draft preparation, Y.S.; writing—review and editing, X.W. and J.S.; supervision, X.W. and X.L.; funding acquisition, X.W. and X.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by Natural Science Foundation of Hainan Province (620RC555), the National Natural Science Foundation of China (No. 61861015, 61961013, 62101088), Radar Signal Processing National Defense Science and Technology Key Laboratory Fund (6142401200101).

Data Availability Statement: Not applicable.

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

The following nomenclature are used in this manuscript:

- $(\cdot)^H$: conjugate-transpose operator
- $(\cdot)^T$: transpose operator
- $||\cdot||_F$: the Frobenius norm operator
- $||\cdot||_2$: the 2-norm operator
- $Re(\cdot)$: the real part operator
- $Im(\cdot)$: the imag part operator
- $\text{Ang}l(\cdot)$: the Phases part operator
- $E[\cdot]$: the expectation operator
- $\text{Var}[\cdot]$: the variance operator
- $I_K$: $K \times K$ identity matrix
- $*$: 2D cross-correlation operator
- $C^{M \times N}$: $M \times N$ dimensional complex matrix set
- $Q^{M \times N}$: $M \times N$ dimensional rational numbers matrix set

References


**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.