remote sensing

MDPI

Article

Radar Intra–Pulse Signal Modulation Classification with Contrastive Learning

Jingjing Cai [1], Fengming Gan [1], Xianghai Cao [2,*], Wei Liu [3] and Peng Li [1]

[1] School of Electronic Engineering, Xidian University, Xi'an 710071, China
[2] School of Artificial Intelligence, Xidian University, Xi'an 710071, China
[3] Department of Electronic and Electrical Engineering, University of Sheffield, Sheffield S1 3JD, UK
* Correspondence: caoxh@xidian.edu.cn

Abstract: The existing research on deep learning for radar signal intra–pulse modulation classification is mainly based on supervised leaning techniques, which performance mainly relies on a large number of labeled samples. To overcome this limitation, a self–supervised leaning framework, contrastive learning (CL), combined with the convolutional neural network (CNN) and focal loss function is proposed, called CL—-CNN. A two–stage training strategy is adopted by CL–CNN. In the first stage, the model is pretrained using abundant unlabeled time–frequency images, and data augmentation is used to introduce positive–pair and negative–pair samples for self–supervised learning. In the second stage, the pretrained model is fine–tuned for classification, which only uses a small number of labeled time–frequency images. The simulation results demonstrate that CL–CNN outperforms the other deep models and traditional methods in scenarios with Gaussian noise and impulsive noise–affected signals, respectively. In addition, the proposed CL–CNN also shows good generalization ability, i.e., the model pretrained with Gaussian noise–affected samples also performs well on impulsive noise–affected samples.

Keywords: intra–pulse modulation classification; contrastive learning; convolutional neural network; focal loss function

check for updates

Citation: Cai, J.; Gan, F.; Cao, X.; Liu, W.; Li, P. Radar Intra–Pulse Signal Modulation Classification with Contrastive Learning. *Remote Sens.* **2022**, *14*, 5728. https://doi.org/10.3390/rs14225728

Academic Editor: Piotr Samczynski

Received: 15 September 2022
Accepted: 8 November 2022
Published: 12 November 2022

1. Introduction

Radar intra–pulse signal modulation classification is an important area in modern electronic warfare (EW) and plays a crucial role in electronic support measure (ESM) systems [1–3]. In the modern battlefield, the quality of the intercepted radar intra–pulse signals is usually poor, and its quantity is low, causing great difficulties to the following classification task [1].

There are mainly two categories of approaches for radar intra–pulse signal modulation classification: the traditional feature extraction–based approaches and the recent deep learning–based ones. For the first category, the algorithms usually extract some useful features from the signal before classification [4–9]. The feature extraction methods employed in these algorithms include time–frequency transform using short–time Fourier transform (STFT) in [4], Choi–Williams' time–frequency distribution in [9], power feature extraction using Rihaczek distribution (RD) and Hough transform (HT) in [5], integrated quadratic phase function (IQPF) and fractional Fourier transform (FrFT) in [6], time–frequency transform using Wigner Ville distribution (WVD) and FrFT in [7], and optimal classification atom and improved double–chains quantum genetic algorithm (IDCQGA) in [8]. Although these algorithms perform well in some high signal–to–noise ratio (SNR) situations, they have common shortcomings. Firstly, their computation complexity is usually high, which causes a slow response from the system; secondly, the classification success rate is usually low with lower SNRs; thirdly, the identification thresholds of these algorithms are usually too sensitive to the signal parameters. These shortcomings greatly limit the practical usage of this category of algorithms.

With the fast development of deep learning techniques [10–12], a new category of algorithms is applied to the radar intra–pulse signal modulation classification problem. Based on different domains of the signals worked on, they can be roughly divided into raw signal–based algorithms [13–15] and time–frequency transformation–based ones [16–20].

For the first class, the signal sequences are directly used as the input of the deep models, such as the network composed of convolutional neural network (CNN), long short–term memory (LSTM) network and fully connected network (FCN) in [13]; CNN with attention mechanism model in [14], etc. For the second class, time–frequency images are utilized as the input; for example, the model based on a convolutional autoencoder (CDAE) and CNN was proposed in [16], CNN architecture (CNN–Xia) composed of 11 layers was used in [18], and the deep residual network with attention mechanism was proposed in [19]. Generally, the classification performance of the time–frequency image–based algorithms is better than that of the raw signal–based ones. However, the classification speed of the raw signal–based one is usually higher. Compared with the traditional feature extraction–based approaches, the deep learning–based solutions can overcome almost all the shortcomings of the traditional ones; however, they require a large number of well–labeled training samples. If only a small number of labeled data samples are available, the performance of these algorithms will degrade or even fail to work completely.

Few–shot learning (FSL) is proposed to solve the samples lacking a problem, which greatly relieves the burden of collecting large–scale supervised data [21]. One of the most popular FSL methods is transfer learning [22], which has been widely used in many fields, such as image classification [23] and text classification [24]. Some radar intra–pulse signal modulation classification algorithms based on transfer learning have also been proposed [25–28]. It has been found that the time–frequency image–based algorithms provide higher classification accuracy than the raw signal–based ones. In fact, the limitation of transfer learning is that a large number of labeled samples are still needed for training, although the source signals can be different from the target signals.

On the other hand, contrastive learning (CL) has also been proposed to solve the small samples problem. CL requires a large number of unlabeled samples for pretraining and a small number of labeled ones for fine–tuning, and its performance has been demonstrated in image classification [29–31]. Although CL has been used in communication signal modulation classification [32], its performance has not been studied in the context of radar intra–pulse modulation classification yet.

In this paper, a CL–based CNN with focal loss function (CL–CNN) model is proposed for radar intra–pulse modulation classification, which retains the core idea of CL and improves the classification accuracy with self–supervised pretraining. One common scenario in radar intra–pulse modulation classification is that the labeled samples for training are affected by white Gaussian noise, while tested samples may be affected by other types of noise. The impulsive noise may be the one having the most adverse effect on radar signals. In order to show that the CL–based model has good generalization property for different noise scenarios, in the simulation parts, the training samples are all embedded with white Gaussian noise. The simulation results show that the proposed model has better performance on classification accuracy compared with the other four deep models, i.e., AAMC–DCNN, CNN–Xia, ResNet (residual network) and SCRNN (Sequential Convolutional Recurrent Neural Network) and two traditional methods, i.e., KNN (K–Nearest Neighbor) and SVM (Support Vector Machine).

The paper is organized as follows. In Section 2, some related works are introduced briefly. The signal preprocessing step is presented in Section 3. In Section 4, the CL–CNN model is provided, including an overview and detailed construction of the proposed model. The simulation results are presented in Section 5, where the proposed model is compared with the other deep models and traditional methods, the impact of various settings for the proposed model are studied, and the generalization ability is also evaluated. Conclusions are drawn in Section 6.

2. Deep Learning–Related Works

2.1. Self–Supervised Learning

Self–supervised learning (SSL) is a subset of unsupervised learning, which overcomes the limitation of manually labeling samples in deep learning [33,34]. In SSL, a pretext task is constructed at first, which is a predesigned task solved by the SSL–based network. The SSL–based network mines the features from a large number of unlabeled samples, and it is pretrained efficiently without involving manual labels, which alleviates the cost of collecting and annotating large–scale datasets [35,36]. After pretraining, the parameters of the SSL–based network are fine–tuned using a small number of labeled samples for downstream tasks [29].

2.2. Contrastive Learning

CL is a representative framework in the field of SSL, and it is used as the pretraining step in deep learning. In CL, positive and negative sample pairs are generated by some data augmentation methods, such as image rotation, image coloration, etc. [34,37]. The original sample is called the anchor sample, and the positive pairs are formed with anchor samples and their augmented versions. The negative pairs are formed with different anchor samples and the augmented versions. During training, the similarity between positive samples is strengthened, and the distance between negative samples is enlarged. The similarity should satisfy the following formulation:

$$\mathrm{score}\big(f(x), f\big(x^{+}\big)\big) \gg \mathrm{score}\big(f(x), f\big(x^{-}\big)\big) \tag{1}$$

where $\mathrm{score}(\cdot,\cdot)$ is a function that measures the similarity of two samples, $f(\cdot)$ is the encoding function,x is an anchor sample and x^{+} and x^{-} are the positive and negative samples of x, respectively.

Contrastive learning tries to assign a larger value to a positive sample pair and a smaller value to a negative sample pair. Then, the feature of x will be more similar to the feature of x^{+} and more dissimilar to the feature of x^{-}.

2.3. Focal Loss Function

Many existing deep models ignore the difficult imbalance of different sample categories, which may degrade their performance. Online hard example mining (OHEM) has been proposed to deal with the difficult imbalance issue [38]. However, it puts more weight on the misclassified samples and ignores the easy–to–classify samples. To solve this problem, the focal loss function is proposed, and it allows the model to focus more on difficult samples by reducing the weights of easy–to–classify samples during the training process [39]. Nowadays, the focal loss function has been widely used in many fields, such as text detection [40] and medical image processing [41].

3. Signal Model and Data Preprocessing

3.1. Signal Model

In a real electromagnetic environment, the signal $x(k)$ is usually disturbed by additive white Gaussian noise (AWGN). After time domain sampling, $x(k)$ can be written as follows:

$$x(k) = Ae^{j\phi(k)} + n(k) \tag{2}$$

where k is the sampling index, A is the amplitude of the signal, $\phi(k)$ is the instantaneous phase, which mainly represents the intra–pulse modulation information, and $n(k)$ is noise.

Some radar signal examples with different $\phi(k)$ are given in Table 1, including frequency–modulated signals (LFM); phase–modulated signals (BPSK); polytime codes (T1 and T2) and polyphase codes (Frank, P1, P2, P3 and P4). In the table, f_c represents the carrier frequency; μ the modulation frequency for LFM; θ the initial phase for BPSK; n, m and T are the phase state number, the number of segments and the coding duration of the

T1 and T2 codes, respectively, M and N_c represents the number of frequency steps of Frank, P1 and P2, and the pulse compression ratio of the P3 and P4 codes, respectively.

Table 1. The instantaneous phase representation of nine types of radar signals.

Modulation Type	Parameter	$\boldsymbol{\phi(k)}$
LFM	f_c μ	$2\pi\left(f_c k \pm \frac{\mu}{2}k^2\right)$
BPSK	f_c θ	$2\pi f_c k + \theta, \theta \in \{0, \pi\}$
T1	m T n	$2\pi f_c k + \bmod\left\{\frac{2\pi}{n}\mathrm{INT}\left[(mk - jT)\frac{jn}{T}\right], 2\pi\right\}, j = 1, 2, \ldots, k-1$
T2	m T n	$2\pi f_c k + \bmod\left\{\frac{2\pi}{n}\mathrm{INT}\left[(mk - jT)\left(\frac{2j-m+1}{T}\right)\frac{n}{2}\right], 2\pi\right\}, j = 1, 2, \ldots, k-1$
Frank	M	$2\pi f_c k + 2\pi(i-1)(j-1)/M, i = 1, 2, \ldots, M, j = 1, 2, \ldots, M$
P1	M	$2\pi f_c k - \pi[M - (2j-1)][(j-1)M + (i-1)]/M, i = 1, 2, \ldots, M, j = 1, 2, \ldots, M$
P2	M	$2\pi f_c k - \pi[2i - 1 - M][2j - 1 - M]/2M, i = 1, 2, \ldots, M, j = 1, 2, \ldots, M$
P3	N_c	$2\pi f_c k + \pi(i-1)^2/N_c, i = 1, 2, \ldots, N_c$
P4	N_c	$2\pi f_c k + \pi(i-1)^2/N_c - \pi(i-1), i = 1, 2, \ldots, N_c$

3.2. Impulsive Noise Model

In the real environment, there is impulse noise in addition to white Gaussian noise. Impulse noise consists of irregular pulses or noise spikes with a short duration and large amplitude, which is mainly caused by system defects [42,43]. The Bernoulli–Gaussian noise model is one of the most widely used in describing impulse noise, which is adopted here. Normally, when radar signals are affected by impulse noise, the additive white Gaussian noise also exists. As a result, the impulsive noise model can be written as:

$$u(k) = b(k)g_1(k) + g_2(k) \tag{3}$$

where $b(k)$ is a Bernoulli process with a probability of occurrence as p, $g_1(k)$ and $g_2(k)$ are the independent and zero mean with variances σ_1^2 and σ_2^2, separately, and they satisfy $\sigma_1^2 \gg \sigma_2^2$.

The probability distribution functions of $b(k)$ and $g_i(k), i = 1, 2$ are $f_b(y)$ and $f_{g_i}(y)$, respectively, which can be written as

$$\begin{aligned} f_b(y) &= p^y(1-p)^{1-y}, y \in \{0, 1\} \\ f_{g_i}(y) &= \frac{1}{\sqrt{2\pi}\sigma_i}\exp\left(-\frac{y^2}{2\sigma_i^2}\right), y \in (+\infty, -\infty) \end{aligned} \tag{4}$$

where y is the variable in functions $f_b(y)$ and $f_{g_i}(y)$.

3.3. Data Preprocessing

As contrastive learning usually requires image data as its input, a preprocessing of converting radar signal sequences into images is needed. There are several ways of transforming the data, and in this paper, we transform the signal into time–frequency images.

In the time–frequency analysis, the Choi–Williams distribution (CWD) has a relatively stable characterization for local characteristics of the signals, the continuous form of which can be expressed as [1]

$$\begin{aligned} C(w,t) &= \iiint_{\infty} f(\eta,\tau)e^{j2\pi\eta(s-t)}\overline{x}(s,\tau)e^{jw\tau}d\eta ds d\tau \\ \overline{x}(s,\tau) &= x(s+\tau/2)x^*(s-\tau/2) \end{aligned} \tag{5}$$

where $C(w,t)$ is the time–frequency distribution function; w and t refer to the coordinates of frequency and time; $(\cdot)^{*}$ is a conjugate operation; $f(\eta,\tau)$ is a kernel function, which satisfies $f(\eta,\tau) = e^{(\pi\eta\tau)^{2}/2\delta}$, and s, τ, η and δ are the time, time delay, frequency shift and controllable factor to suppress cross–terms, respectively.

In order to improve the classification performance, feature enhancement algorithms are usually used in radar intra–pulse signal modulation classification [16,44,45]. However, these algorithms incur additional computational complexity and a certain degree of information loss. Therefore, the CWD feature is directly used as the initial feature without any other feature enhancement processing in this paper.

4. CL–CNN Model

4.1. Overview of the CL–CNN Model

The CL–based model needs a deep network to extract the features of images, and as CNN is widely used in image classification for its excellent feature extraction performance, it is employed in our CL–based network. The newly constructed network is called CL–CNN, and its architecture is shown in Figure 1. There are mainly three parts in the pretraining stage, which are the stochastic data augmentation module, the encoder and the projection head. In the fine–tuning stage, the projection head is discarded, and the classification head is added after the encoder.

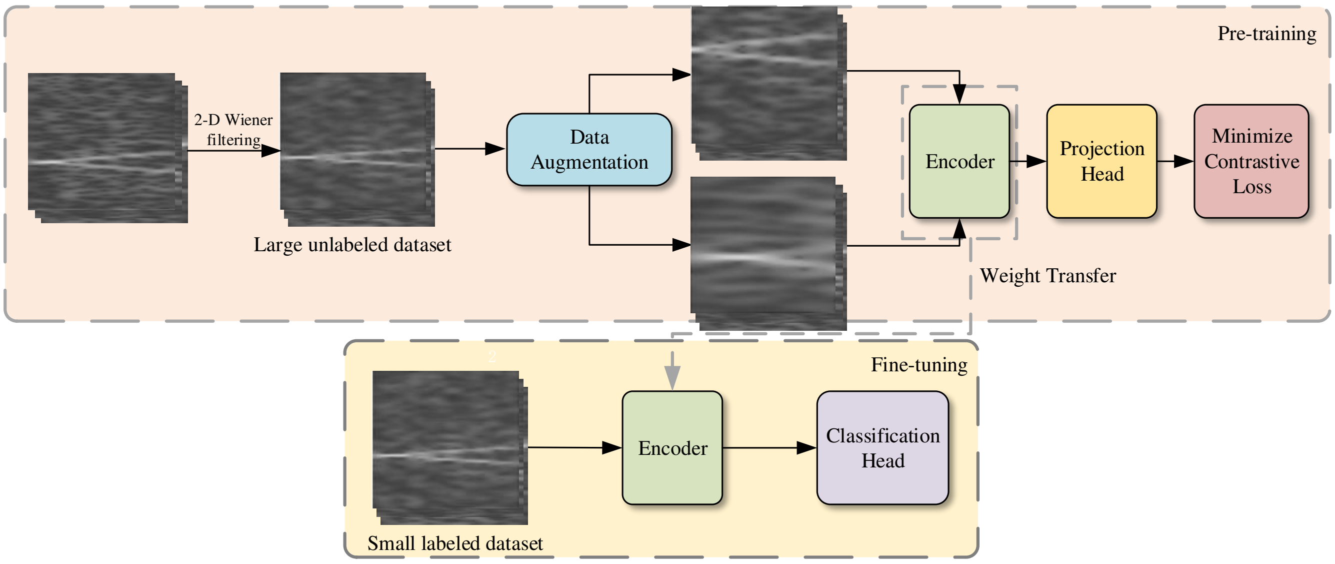

Figure 1. Architecture of the CL–CNN model.

These parts play different roles in radar signal modulation classification. In the pretraining stage, the stochastic data augmentation module is mainly used for producing positive sample pairs and negative sample pairs, the encoder extracts representative features from data examples, the projection head maps representations to the space where the contrastive loss is applied and the classification head makes the final decision for modulation classification in the fine–tuning stage.

The training process of CL–CNN includes two steps, pretraining and fine–tuning, as shown in Figure 1. In the pretraining step, the encoder and the projection head are pretrained with unlabeled augmented data to map the original time–frequency images into low–dimensional representations. The aim is to minimize the contrastive loss and make the features of the anchor data more similar to that of the positive data and more dissimilar to that of the negative ones.

In the fine–tuning step, the weight of the encoder is saved, and a classification head is built after it. The encoder and classification head are together trained with labeled data. The latent representation obtained from the last step has already learned the intrinsic information from the unlabeled data examples, so only a small number of labeled data is needed to fine–tune the model.

4.2. De–Noising Algorithm

A time–frequency analysis can reduce noise to some extent, but some noise still remains, especially at low SNRs. Denoising algorithms can be used to further reduce the noise in the time–frequency images. Two–dimensional Wiener filtering is one of the effective denoising algorithms that can adjust the effect of the filter according to the local variance of the image. A detailed introduction to 2D Wiener filtering is given below, and more details can be found in [46].

Consider a 1D gray–scaled image of $a \times b$ pixels with value matrix $H_{a\times b}$, and the value of each pixel is denoted as $H(i,j), i = 1,2,\ldots,a, j = 1,2,\ldots,b$. The size of a filter neighborhood is $\hat{a} \times \hat{b}$, and the mean $\hat{\mu}$ and variance $\hat{\sigma}^2$ of each pixel of the neighborhood can be calculated by the following equations:

$$\begin{aligned} \hat{\mu} &= \frac{1}{\hat{a}\hat{b}} \sum_{i,j\in\hat{\beta}} H(i,j) \\ \hat{\sigma}^2 &= \frac{1}{\hat{a}\hat{b}} \sum_{i,j\in\hat{\beta}} H^2(i,j) - \hat{\mu}^2 \\ \hat{H}(i,j) &= \hat{\mu} + \frac{\hat{\sigma}^2 + \sigma_N^2}{\hat{\sigma}^2}[H(i,j) - \hat{\mu}] \end{aligned} \tag{6}$$

where $\hat{H}(i,j)$ is the filtered pixel of $H(i,j)$ and σ_N^2 the variance of the noise.

4.3. Data Augmentation

In this paper, the data augmentation part is implemented by random resized crop, horizontal flip and Gaussian noise. These data augmentation methods increase the diversity of the positive and negative sample pairs. In detail, 'random resized crop' randomly crops an area on the original image and then resizes it into a given size, and 'horizontal flip' flips the image horizontally. These two data augmentation methods enhance the generalization ability of the model. 'Gaussian noise' adds zero mean Gaussian noise to the original image, which distorts the high–frequency features and enhances the learning ability of the neural network.

Suppose the batch size of the input data is N; after data augmentation, each input sample x_k in this batch forms a positive pair $(\hat{x}_{2k-1}, \hat{x}_{2k})$, and all remaining $2(N-1)$ samples in this batch are used to form negative pairs.

4.4. Encoder

The encoder is a deep network used for feature extraction. The network cannot be too deep, as the gradient disappearance problem will appear, and the parameters of the previous layer cannot be trained effectively. The residual network (ResNet) is composed of some residual blocks, and it can alleviate the training difficulty by shortcut connections. The output of ResNet can be obtained by summing the output and input of multiple cascaded convolutional layers [47]. There is a Rectified Linear Unit (ReLU) nonlinear activation function after each residual block. The structure of the residual block is shown in Figure 2. It can be seen that the residual block mainly contains two 3×3 convolution layers (Conv), two batch normalization layers (BN) and three nonlinear activation function layers. For the input $\hat{x}_i$, suppose h_i is the final output of the residual block, and it is the sum of the forward neural network $H(\hat{x}_i)$ and $\hat{x}_i$, i.e., $h_i = H(\hat{x}_i) + \hat{x}_i$.

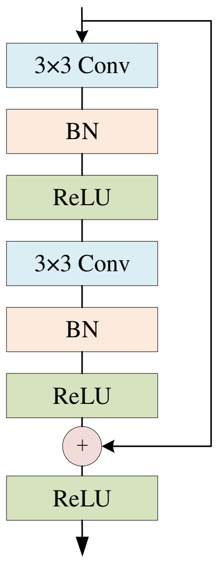

Figure 2. Structure of the residual block.

4.5. Projection Head

The projection head is composed of two fully connected layers, which maps the output of the encoder to a low–dimensional feature space and helps identify the invariant features of each input. The processing steps are listed as follows.

(1) Calculate the output of the projection head for each input h_i.

$$z_i = W_2(\sigma(W_1(h_i))) \tag{7}$$

where $i = 1, 2, \ldots, 2N$, N is the batch size, W_1 and W_2 are the weights of the two fully connected layers and $\sigma(\cdot)$ is the ReLU nonlinear activation function.

(2) Calculate the cosine similarity for each output of the projection head.

$$\mathrm{sim}(z_i, z_j) = \frac{z_i^T z_j}{\|z_i\|\|z_j\|} \tag{8}$$

where $\|\cdot\|$ is the $\updownarrow_2$ norm, and $(\cdot)^T$ is the conjugate transposition operation.

(3) Calculate the average normalized temperature–scaled cross–entropy loss.

$$\begin{aligned} \overline{L} &= \tfrac{1}{2N}\sum_{k=1}^{N}[L(2k-1, 2k) + L(2k, 2k-1)] \\ L(i,j) &= -\log \frac{\exp(\mathrm{sim}(z_i,z_j)/\xi)}{\sum_{k=1}^{2N} \mathbb{I}_{[k\neq i]} \exp(\mathrm{sim}(z_i,z_k)/\xi)} \end{aligned} \tag{9}$$

where ξ is the adjustable temperature parameter, and $\mathbb{I}_{[k\neq i]} \in \{0, 1\}$ is an indicator function. The value of the indicator is 0 if and only if $k = i$; otherwise, it is 1.

(4) Update the parameters of the encoder and projection head to minimize the loss $\overline{L}$.

4.6. Classification Head and Focal Loss Function

In the classification stage, as the parameters of the encoder have been pretrained and the projection head has been discarded, two fully connected layers are included to make the final decision on the modulation type.

In general, the cross–entropy loss function is widely used in the classification stage. However, the cross–entropy loss treats all the samples equally and does not differentiate the complicated samples and simple ones. Therefore, the classification performance may degrade due to the difficulty imbalance problem. The cross–entropy loss function can be written as

$$\mathrm{CE}(p) = -\log(p) \tag{10}$$

where $p \in [0,1]$ is the model's estimated probability for the ground–truth class.

Compared with the cross–entropy loss function $\mathrm{CE}(p)$, the focal loss function can be seen as the cross–entropy loss function with an extra adjustable parameter, which can be written as

$$\mathrm{FL}(p) = -(1-p)^{\gamma}\log(p) \tag{11}$$

where γ is an adjustable hyperparameter.

Normally, the probability p is small for difficult samples, which is $p \to 0$ and $(1-p)^{\gamma} \to 1$; as a result, the focal loss does not change significantly. On the contrary, the probability p of simple samples is large, so the contribution of this probability to the total loss is small. Employment of the focal loss in the classification head solves most of the difficulty imbalance problems and further improves the classification performance.

5. Simulations and Analyses

In this section, the classification performance of the CL–CNN model is compared with three deep models and two traditional methods, which are AAMC–DCNN, CNN–Xia, ResNet, SVM and KNN. The benchmark datasets are introduced, and the implementation details are introduced briefly. Then, the parameter settings of the CL–CNN model are studied, and the classification results are presented. All the simulation results are obtained by 50 trials.

5.1. Datasets

Simulations are conducted using two datasets, which contain nine types of radar intra–pulse signals shown in Table 1 and are contaminated by Gaussian white noise and impulsive noise, separately. The parameter settings of the signals are listed in Table 2, where $U(\cdot)$ denotes the uniform random distribution of data in a fixed interval; $\{\cdot\}$ represents a random parameter set and f_s, f_c, B, N_s, L_c, N_{cc}, N_g, M and ρ represent the sampling frequency, carrier frequency, bandwidth, number of samples, length of barker, number of carrier cycles in a single–phase symbol, number of segments, number of frequency steps and number of subcodes in a code, respectively.

Table 2. The parameter settings of nine types of radar intra–pulse signals.

Modulation Type	Parameter	Ranges
–	f_s	200 MHz
LFM	f_c	$U(3f_s/20, f_s/5)$
	B	$U(f_s/20, f_s/10)$
	N_s	$U(500, 1000)$
BPSK	L_c	$\{7, 11, 13\}$
	f_c	$U(3f_s/20, f_s/5)$
	N_{cc}	$\{20, 21, 22, 23\}$
T1, T2	f_c	$U(3f_s/20, f_s/5)$
	N_g	$\{4, 5, 6\}$
	N	$U(500, 1000)$
Frank, P1	f_c	$U(3f_s/20, f_s/5)$
	N_{cc}	$\{4, 5.6\}$
	M	$\{5, 6, 7, 8\}$

Table 2. *Cont.*

Modulation Type	Parameter	Ranges
P2	f_c	$U(3f_s/20, f_s/5)$
	N_{cc}	$\{4, 5, 6\}$
	M	$\{6, 8\}$
P3, P4	f_c	$U(3f_s/20, f_s/5)$
	N_{cc}	$\{4, 5.6\}$
	ρ	$\{36, 49, 64\}$

The time–frequency images of four types of signals, i.e., LFM, two PSK and Frank, under white Gaussian noise and impulsive noise are shown in Figure 3, respectively. There are clear differences between the signal images under different types of noise.

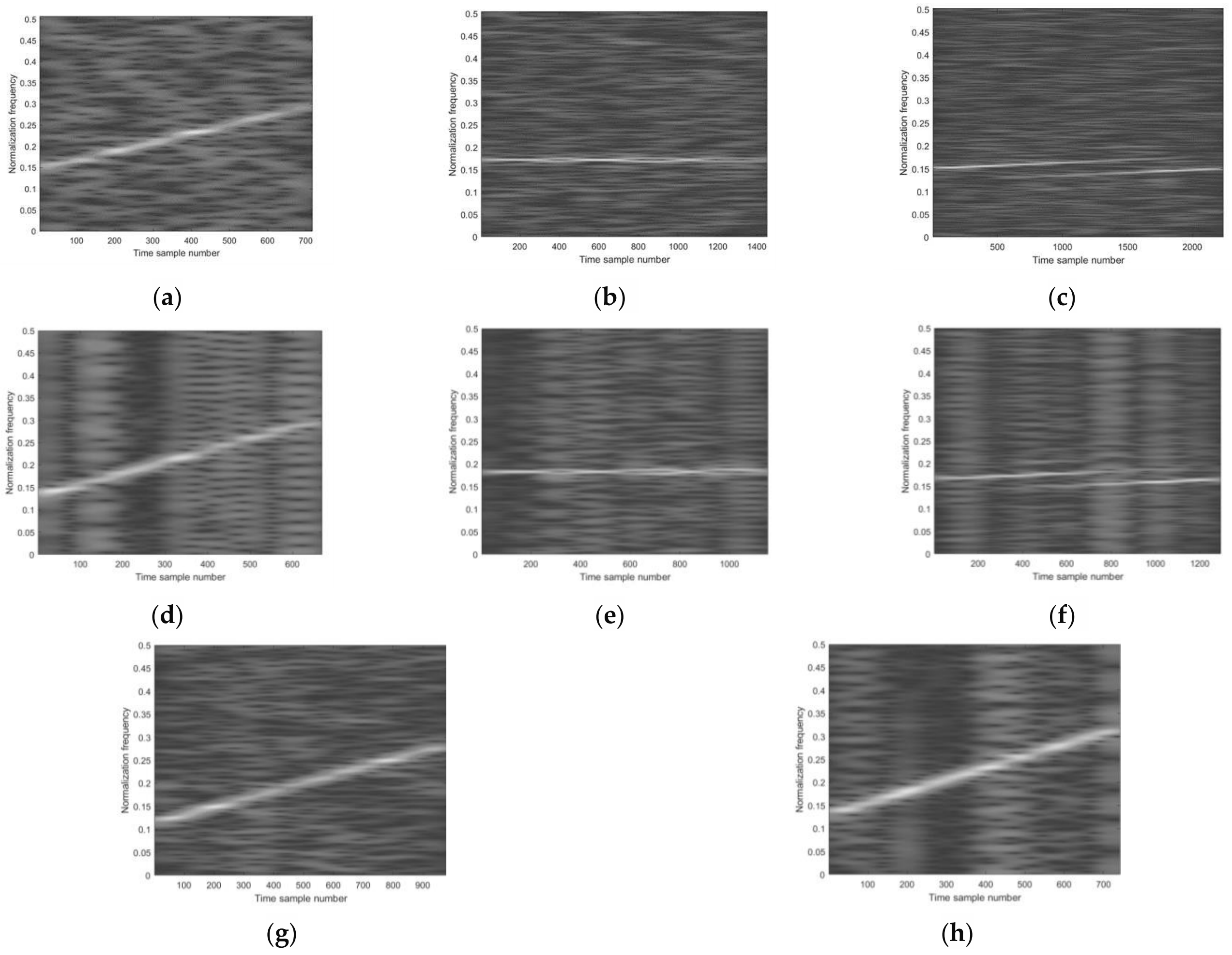

Figure 3. The time–frequency images: (**a**) LFM signal under Gaussian noise, (**b**) BPSK signal under Gaussian noise, (**c**) Frank signal under Gaussian noise, (**d**) LFM signal under impulsive noise, (**e**) BPSK signal under impulsive noise, (**f**) Frank signal under impulsive noise, (**g**) filtered image of LFM signal under Gaussian noise and (**h**) filtered image of LFM signal under impulsive noise.

5.2. Comparisons with Deep Models and Traditional Methods

Three types of deep models and two types of traditional methods are compared with the CL–CNN model for radar intra–pulse signal modulation classification. Furthermore, the CL–CNN–CE model using the cross–entropy loss function is also considered as one comparison model, which is different from the CL–CNN model using the focal loss function. For all the deep models, the softmax activation function is used in the last layer, and the

Adam optimizer and cross–entropy loss function are applied in the deep models. The details of the deep models and traditional methods are listed in Table 3.

Table 3. The details of the deep models and traditional methods.

Model	Input	Cross–Entropy Loss	Focal Loss	Transfer Learning	Contrastive Learning
CL–CNN	Time–frequency images	No	Yes	No	Yes
CL–CNN–CE	Time–frequency images	Yes	No	No	Yes
AAMC–DCNN	Time–frequency images	Yes	No	Yes	No
CNN–Xia	Time–frequency images	Yes	No	No	No
ResNet	Time–frequency images	Yes	No	No	No
SCRNN	Signal sequences	Yes	No	No	No
SVM	Signal sequences	-	-	-	-
KNN	Signal sequences	-	-	-	-

The AAMC–DCNN model is similar to DenseNet, but it has a dense–connection mechanism [28]. A two–stage training strategy is adopted for the model. In the first stage, the model is pretrained on the widely used ImageNet dataset. In the second stage, the pretrained model is fine–tuned for classification using the time–frequency images of our datasets.

The CNN–Xia model contains several convolution layers, maxpooling layers and fully connected layers [18]. There is a maxpooling layer after each convolution layer. Two fully connected layers and a dropout layer are added after the convolutional layers.

The ResNet model is a well–known deep convolutional neural network. The shortcut connections in the model can alleviate training difficulties in deeper networks without introducing either extra parameters or computational complexity. Many successful applications have been made using this model in modulation classification [48–50].

The SCRNN (Sequential Convolutional Recurrent Neural Network) model combines the speed and lightness of the CNN and temporal sensitivity of RNN [51]. It can be divided into three parts: the first part contains two convolutional layers with 128 filters and ReLU activation functions, the second part is two 128–unit LSTM layers with ReLU activation functions and the last part is a dense layer.

The SVM [52] maps the input data into a high–dimensional feature space, where a linear decision surface is constructed. Different types of classes are located on different sides of the decision surface.

The KNN (K–Nearest Neighbor) follows the nearest decision rule and assigns an unclassified sample point to the nearest classified set [53].

5.3. Implementation Details

For the first dataset, 5000 samples of each type of signals are produced for pretraining, while 85 and 1500 samples of each type of signals are produced for the fine–tuning and test, respectively. All of them are affected by Gaussian white noise. For the second dataset, 85 and 1500 samples of each type affected by impulse noise are produced to fine–tune and test the generalization ability of contrastive learning, respectively. The implementation details for the models and methods are shown in Table 4.

Table 4. The implementation details of the deep models and traditional methods.

Model	Pretraining	Fine–Tuning	Direct Training
CL–CNN	Dataset1: 5000 samples for each class	Dataset 1: 15–85 samples for each class Dataset 2: 15–85 samples for each class	No
AAMC–DCNN	ImageNet		
CNN–Xia	No	No	Dataset 1: 15–85 samples for each class Dataset 2: 15–85 samples for each class
ResNet			
SCRNN	No	No	Dataset 1: 15–85 samples for each class
SVM	-	-	
KNN			
All of the models and methods are tested with 1500 samples for each class.			

The Adam optimizer with a learning rate of 0.001 is utilized. All the trainings and predictions are implemented in Pytorch.

5.4. Parameter Analyses

As the classification accuracy varies with the parameter settings, it is necessary to find the optimal parameters for the CL–CNN model. The impact of the parameter settings of the CL–CNN model is investigated below, including the adjustable parameter γ, the batch size N and the number of CNN layers.

Figure 4 shows how the adjustable parameter γ affects the classification accuracy. The value of γ represents how much attention the model pays to difficult samples. A large γ leads to a poor classification performance, which means that the model focuses too much on those samples. On the other hand, a small γ also leads to a poor performance, which implies that the difficulty imbalance problem cannot be solved effectively, which degrades the performance of the model. It can be seen that the model with $\gamma = 0.1$ gives the best performance, and the difficulty imbalance problem can be solved to some extent by adjusting this parameter.

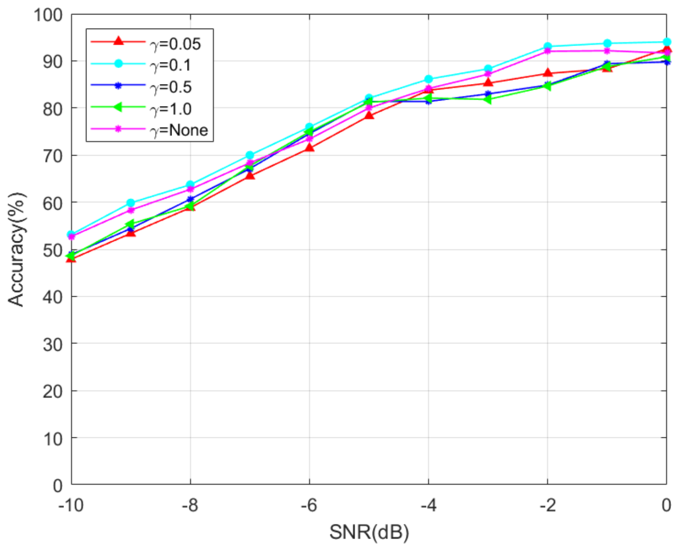

Figure 4. Classification performance of the CL−CNN model with different γ (the sample size per class is 55).

The results by different batch sizes N are shown in Table 5, which indicates that the classification performance with batch size 32 is the best. Normally, a larger batch size will bring a better classification performance, but the performance will degrade when the batch size exceeds the threshold value [54].

Table 5. Classification performance of the CL–CNN model at different batch sizes with −5 dB SNR and the sample size of each class being 55.

Batch Size	**8**	**16**	**32**	**64**	**128**
Accuracy(%)	78.48	79.70	82.09	78.71	76.19

The simulation results for different numbers of layers are shown in Table 6, which indicates that the classification performance of 34 layers is the best, with a running time of 7.72 s. Normally, a larger number of layers will increase the generalization ability and bring a better classification performance, but the performance will degrade due to the overfitting problem when the number of layers is too large.

Table 6. Results of the CL–CNN model with a different number of layers with −5 dB SNR and the sample size for each class being 55.

Layer Number	**18**	**34**	**50**	**101**
Accuracy(%)	79.32	82.09	75.96	72.54
Time cost * (s)	6.67	7.72	9.52	13.42

* The running time of the whole test samples.

5.5. Performance Comparisons

The classification results are presented in this subsection, including CL–CNN and CL–CNN–CE, four deep models and two traditional methods. In addition, CL–CNN–WP is also considered in the comparison, which has the same structure as CL–CNN– CE and is directly fine–tuned without pretraining. All the deep models and traditional methods are configured in their best performance. The first simulation tests the effect of denoising, the second to fifth simulations test the feature extraction ability of contrastive learning, the sixth and seventh simulations test the generalization ability and the last one is the statistical analysis. For all simulations, 1500 samples of each signal type are used for the test, and 15–85 samples of each signal type are used for fine–tuning.

In the first simulation, the performances of CL–CNN using filtered and unfiltered images as input are shown in Table 7. The sample size for each class is set as 55. It can be seen that the method using filtered images achieves 3% to 4% more classification accuracy improvement than the one using unfiltered images at low SNRs. The comparison results indicate that the denoising algorithm is effective in reducing the effect of noise.

Table 7. Classification performance of CL–CNN using filtered and unfiltered time–frequency images based on the first dataset.

Input	**SNR(dB)**										
	−10	**−9**	**−8**	**−7**	**−6**	**−5**	**−4**	**−3**	**−2**	**−1**	**0**
Filtered images	53.11	59.85	63.70	69.97	75.95	82.09	86.12	88.31	93.05	93.72	94.02
Unfiltered images	48.97	54.83	60.04	66.19	70.81	77.39	82.18	85.53	90.13	91.20	93.42

In the second simulation, the performances of CL–CNN, CL–CNN–CE and CL–CNN–WP are compared, and the results are shown in Table 8. It can be seen that CL–CNN achieves 1% to 2% classification rate improvement compared with CL–CNN–CE, which implies that the focal loss function effectively solves the difficult imbalance problem to further improve the classification performance. The performance of CL–CNN–WP is the worst, especially in the case of insufficient samples. It indicates that the pretraining greatly improves both the feature extraction ability and classification performance with a limited number of samples.

Table 8. Classification performance of the CL–CNN, CL–CNN–CE and CL–CNN–WP models at −5 dB SNR.

Model	**Sample Size for Each Class**							
	15	**25**	**35**	**45**	**55**	**65**	**75**	**85**
CL–CNN	63.61	72.57	76.64	79.09	82.09	83.69	84.74	87.47
CL–CNN–CE	62.24	69.49	75.11	77.62	80.27	81.22	82.54	86.14
CL–CNN–WP	52.41	62.74	63.39	64.50	68.66	73.73	78.69	83.73

In the third simulation, the performance of CL–CNN with different sample sizes for each class and SNR is presented, and the results are shown in Table 9. The sample size for each type of signal increases from 15 to 85 with an interval of 10. It can be concluded that the proposed model exhibits good performance in a different number of samples, and the classification accuracy improves as the number of samples increases or SNR increases. When the SNR and sample size for each class are equal to or greater than −1 dB and 55, separately, the classification accuracy is more than 90%.

Table 9. Classification performance of the CL–CNN model in varying sample sizes for each class and SNR.

SNR / Size	**−10 dB**	**−9 dB**	**−8 dB**	**−7 dB**	**−6 dB**	**−5 dB**	**−4 dB**	**−3 dB**	**−2 dB**	**−1 dB**	**0 dB**
15	37.73	42.57	47.63	52.21	56.58	63.61	67.43	70.82	76.13	83.63	85.36
25	37.94	45.67	50.77	54.38	60.41	72.57	74.19	75.91	78.69	85.71	86.92
35	42.54	46.15	54.81	57.80	68.60	76.64	76.63	80.72	83.55	87.35	87.47
45	46.78	48.91	57.46	62.36	71.92	79.09	81.49	82.37	85.92	90.63	91.19
55	53.11	59.85	63.70	69.97	75.95	82.09	86.12	88.31	93.05	93.72	94.02
65	53.62	60.46	63.30	70.44	76.43	83.69	86.68	88.56	93.22	93.30	93.44
75	53.23	60.74	63.69	71.39	76.87	84.74	86.84	88.48	93.19	93.66	93.94
85	57.03	64.86	68.92	74.78	82.51	87.47	90.95	90.42	93.57	94.56	94.79

In the fourth simulation, the classification results of the deep models and traditional methods with the sample size per class 15 are shown in Table 10. It can be seen that CL–CNN achieves much better classification accuracy than the other deep models and traditional methods, which indicates that CL improves the feature extraction ability of the proposed model. As the deep models, CNN–Xia, AAMC–DCNN and ResNet are based on time–frequency images, while SCRNN, SVM and KNN are based on signal sequences, the classification performances of the first three models are clearly better than those of the other three. As the architecture of AAMC–DCNN is complex and the number of samples is very small, the overfitting problem may cause degradation to its performance, and as a result, CNN–Xia performs better with a simple structure. The signal sequence–based solutions show poor performances due to low discriminating inputs; while, among them, the performance of SVM is better than that of KNN.

In the fifth simulation, the models with the sample sizes per class 55 are tested, and the results are shown in Table 11. Although the sample size per class increases compared with that in the fourth simulation, CL–CNN still performs the best. It outperforms CNN–Xia and ResNet by nearly 4% for the whole SNR range. The performance of AAMC–DCNN has some improvement, which is comparable to CL–CNN with the SNR equal to or greater than 0 dB. The performance of SCRNN does not improve obviously when the sample size per class increases, as it may still suffer from the overfitting problem. The performance of KNN is better than that of SVM with an increased sample size for each class.

Table 10. Classification performances of the deep models, and the traditional methods with the sample size for each class being 15 based on the first dataset.

Model	**SNR (dB)**										
	−10	**−9**	**−8**	**−7**	**−6**	**−5**	**−4**	**−3**	**−2**	**−1**	**0**
CL–CNN	37.73	42.57	47.63	52.21	56.58	63.61	67.43	70.82	76.13	83.63	85.36
AAMC–DCNN	30.29	32.03	34.74	37.90	41.54	43.59	47.30	50.89	58.64	69.34	75.06
CNN–Xia	34.35	37.21	43.12	47.43	51.46	57.25	61.19	62.52	70.59	75.67	81.61
ResNet	33.94	35.27	37.80	42.47	48.90	52.50	54.96	57.25	65.26	70.80	76.94
SCRNN	12.73	13.53	13.06	14.10	16.02	15.19	17.11	17.15	18.61	20.08	20.01
SVM	13.61	14.51	14.37	16.89	17.70	17.39	18.53	18.16	19.21	20.80	21.45
KNN	13.05	12.91	13.83	14.76	14.94	16.17	16.62	17.27	17.14	17.32	17.33

Table 11. Classification performances of the deep models and the traditional methods, with the sample size for each class being 55 based on the first dataset.

Model	**SNR(dB)**										
	−10	**−9**	**−8**	**−7**	**−6**	**−5**	**−4**	**−3**	**−2**	**−1**	**0**
CL–CNN	53.11	59.85	63.70	69.97	75.95	82.09	86.12	88.31	93.05	93.72	94.02
AAMC–DCNN	47.89	53.17	59.91	64.48	68.21	73.22	79.21	83.54	90.35	91.23	92.37
CNN–Xia	49.16	55.06	60.49	65.22	69.33	75.30	78.96	83.90	90.93	91.45	91.63
ResNet	50.07	56.26	61.11	65.91	70.25	77.36	81.59	84.63	91.59	93.42	93.96
SCRNN	14.81	13.73	16.51	17.23	18.04	19.33	19.03	22.30	23.94	26.86	29.36
SVM	16.52	17.05	19.10	21.25	21.83	23.45	24.95	25.98	26.84	29.62	29.13
KNN	14.71	15.54	17.21	20.52	22.63	24.4	26.32	27.62	29.65	33.37	33.05

The confusion matrix is an important visual tool to compare the predicted results, which is provided in the sixth simulation. Confusion matrices of CL–CNN at various SNRs are presented in Figure 5, where each column represents the predicted modulation class, and each row represents the real modulation class, and the numerical value on each grid denotes the classification probability of the corresponding modulation class. It can be seen that the diagonals become sharper with an increasing SNR, which illustrates that a higher SNR brings a better classification accuracy. However, confusion between Frank and P3 exists when the SNR is equal to or is lower than −5 dB, as it is difficult to distinguish their modulation features at low SNRs.

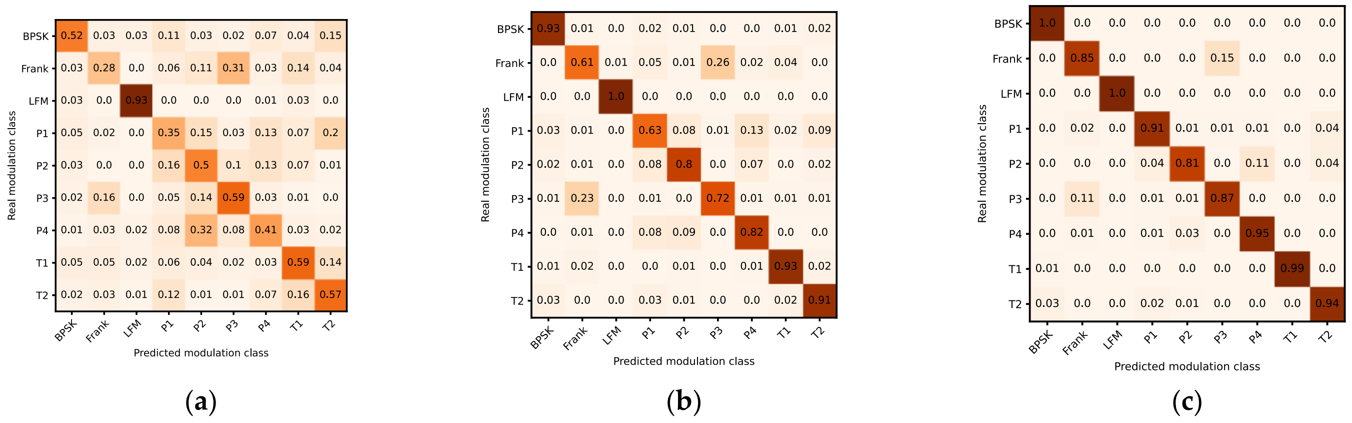

Figure 5. Confusion matrices of the CL–CNN model at different SNRs: (**a**) SNR -10dB, (**b**) SNR-5dB, (**c**) SNR 0dB.

5.6. Generalization Ability Test

In the following two simulations, the Gaussian noise–affected signals are used in the first pretraining stage, and the impulsive noise–affected signals are utilized in the fine–tuning stage. As SCRNN, SVM and KNN use the signal sequence as the input, their performance is very poor in the impulsive noise scenario, so they are not included in the comparison. The classification results are presented in Tables 12 and 13 with the sample size for each class set as 15 and 55, separately. It can be seen that the proposed CL–CNN has the best performance in Tables 12 and 13. Generally, the performance of CL–CNN using impulsive noise–affected signals in the fine–tuning stage does not degrade much compared with that using Gaussian noise, which indicates that the proposed CL–CNN has a good generalization ability.

Table 12. Generalization ability test with the sample size for each class being 15.

Model	**SNR (dB)**										
	−10	**−9**	**−8**	**−7**	**−6**	**−5**	**−4**	**−3**	**−2**	**−1**	**0**
CL–CNN	36.84	41.82	47.30	51.55	55.85	62.71	66.67	70.18	75.80	83.19	84.90
AAMC–DCNN	29.65	31.63	34.15	37.38	40.96	43.13	46.73	50.22	58.28	68.78	74.51
CNN–Xia	33.71	36.71	42.60	46.92	51.08	56.82	60.67	62.01	70.35	75.20	81.12
ResNet	33.47	34.73	37.22	41.83	48.29	52.03	54.33	56.73	64.69	70.31	76.52

Table 13. Generalization ability test with the sample size for each class being 55.

Model	**SNR(dB)**										
	−10	**−9**	**−8**	**−7**	**−6**	**−5**	**−4**	**−3**	**−2**	**−1**	**0**
CL–CNN	52.49	59.27	63.19	69.39	75.59	81.51	85.76	87.89	92.44	93.29	93.59
AAMC–DCNN	47.32	52.78	59.32	64.18	67.71	72.68	78.74	83.02	89.87	90.75	91.65
CNN–Xia	48.78	54.44	60.08	64.80	68.78	73.64	78.52	80.40	90.53	90.91	91.43
ResNet	49.53	55.72	60.79	65.49	69.72	76.80	81.09	84.29	91.06	92.89	93.52

5.7. Statistical Analysis with McNemar's Test

McNemar's test is employed to evaluate the statistical significance between the CL–CNN model and some other deep models, i.e., AAMC–DCNN, CNN–Xia and ResNet. McNemar's test can be written as

$$z = \frac{s_{12} - s_{21}}{\sqrt{s_{12} + s_{21}}} \tag{12}$$

where s_{12} is defined as the number of samples misclassified by model 1 but classified successfully by model 2, while the definition of s_{21} is in contrast. A greater value of $|z|$ means that two models have significant differences in performance.

Let CL–CNN be model 1 and AAMC–DCNN, CNN–Xia and ResNet be model 2, respectively. The values of $|z|$ for different models are listed in Table 14. It can be seen that the performance of the CL–CNN model is statistically different from those of comparison models.

Table 14. The values of $|z|$ with −5 dB SNR and the sample size for each class being 55.

	AAMC–DCNN	**CNN–Xia**	**Resnet**
$\lvert z \rvert$	24.36	21.09	15.80

6. Conclusions

In this paper, a new deep learning–based model called CL–CNN was developed and applied to the radar intra–pulse modulation classification problem successfully. The impact of different parameter settings on its performance was examined to find the best configuration. Simulations were conducted using two datasets. As demonstrated by the results, the proposed model outperformed the other four deep models (AAMC–DCNN, CNN–Xia, ResNet and SCRNN) and two traditional methods (KNN and SVM) in terms of classification accuracy. From the simulation results, it can also be found that CL alleviates the overfitting problem encountered in few–shot learning and improves the feature extraction ability of the proposed model. Furthermore, the proposed model has a good generalization ability facing different noise types. Possible future work can focus on investigating more effective CL–based frameworks for radar intra–pulse modulation classification, which may further improve its performance.

Author Contributions: Conceptualization, J.C. and P.L.; methodology, X.C.; software, F.G.; validation, W.L.; formal analysis, J.C.; investigation, X.C.; resources, W.L.; data curation, F.G.; writing—original draft preparation, F.G. and writing—review and editing, J.C., X.C. and W.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Natural Science Foundation of China (No. 61805189 and 62176199) and the Fundamental Research Funds for the Central Universities (No. JB210201).

Data Availability Statement: Not applicable.

Acknowledgments: The authors would like to express their gratitude to the editors and the reviewers for their insightful comments.

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

References

1. Gupta, M.; Hareesh, G.; Mahla, A.K. Electronic warfare: Issues and challenges for emitter classification. *Def. Sci. J.* **2011**, *61*, 228. [CrossRef]
2. Qu, Z.; Mao, X.; Deng, Z. Radar signal intra-pulse modulation recognition based on convolutional neural network. *IEEE Access* **2018**, *6*, 43874–43884. [CrossRef]
3. Yuan, S.; Li, P.; Wu, B.; Li, X.; Wang, J. Semi-supervised classification for intra-pulse modulation of radar emitter signals using convolutional neural network. *Remote Sens.* **2022**, *14*, 2059. [CrossRef]
4. López-Risuño, G.; Grajal, J.; Sanz-Osorio, A. Digital channelized receiver based on time-frequency analysis for signal interception. *IEEE Trans. Aerosp. Electron. Syst.* **2005**, *41*, 879–898. [CrossRef]
5. Zeng, D.; Zeng, X.; Cheng, H.; Tang, B. Automatic modulation classification of radar signals using the Rihaczek distribution and Hough transform. *IET Radar Sonar Navig.* **2012**, *6*, 322–331. [CrossRef]
6. Fan, X.; Li, T.; Su, S. Intra-pulse modulation type recognition for pulse compression radar signal. *J. Appl. Remote Sens.* **2017**, *11*, 035018. [CrossRef]
7. Kishore, T.R.; Rao, K.D. Automatic intra-pulse modulation classification of advanced LPI radar waveforms. *IEEE Trans. Aerosp. Electron. Syst.* **2017**, *53*, 901–914. [CrossRef]
8. Wan, J.; Ruan, G.; Guo, Q.; Gong, X. A new radar signal recognition method based on optimal classification atom and IDCQGA. *Symmetry* **2018**, *10*, 659. [CrossRef]
9. Lundén, J.; Koivunen, V. Automatic radar waveform recognition. *IEEE J. Sel. Top. Signal Process.* **2007**, *1*, 124–136. [CrossRef]
10. Hinton, G.E.; Osindero, S.; Teh, Y.-W. A fast learning algorithm for deep belief nets. *Neural Comput.* **2006**, *18*, 1527–1554. [CrossRef]
11. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. *Adv. Neural Inf. Process. Syst.* **2012**, *25*, 1097–1105. [CrossRef]
12. Chen, X.; Zhang, H.; Song, J.; Guan, J.; Li, J.; He, Z. Micro-motion classification of flying bird and rotor drones via data augmentation and modified multi-scale cnn. *Remote Sens.* **2022**, *14*, 1107. [CrossRef]
13. Wei, S.; Qu, Q.; Su, H.; Wang, M.; Shi, J.; Hao, X. Intra-pulse modulation radar signal recognition based on CLDN network. *IET Radar Sonar Navig.* **2020**, *14*, 803–810. [CrossRef]
14. Wu, B.; Yuan, S.; Li, P.; Jing, Z.; Huang, S.; Zhao, Y. Radar emitter signal recognition based on one-dimensional convolutional neural network with attention mechanism. *Sensors* **2020**, *20*, 6350. [CrossRef] [PubMed]
15. Wei, S.; Qu, Q.; Zeng, X.; Liang, J.; Shi, J.; Zhang, X. Self-attention bi-lstm networks for radar signal modulation recognition. *IEEE Trans. Microw. Theory Tech.* **2021**, *69*, 5160–5172. [CrossRef]
16. Qu, Z.; Wang, W.; Hou, C.; Hou, C. Radar signal intra-pulse modulation recognition based on convolutional denoising autoencoder and deep convolutional neural network. *IEEE Access* **2019**, *7*, 112339–112347. [CrossRef]
17. Gao, L.; Zhang, X.; Gao, J.; You, S. Fusion image based radar signal feature extraction and modulation recognition. *IEEE Access* **2019**, *7*, 13135–13148. [CrossRef]
18. Xia, Y.; Ma, Z.; Huang, Z. Over-the-Air Radar Emitter Signal Classification Based on SDR. In Proceedings of the 2021 6th International Conference on Intelligent Computing and Signal Processing (ICSP), Xi'an, China, 9–11 April 2021.

19. Jin, X.; Ma, J.; Ye, F. Radar Signal Recognition Based on Deep Residual Network with Attention Mechanism. In Proceedings of the 2021 IEEE 4th International Conference on Electronic Information and Communication Technology (ICEICT), Xi'an, China, 18–20 August 2021.
20. Zhang, X.; Zhang, J.; Luo, T.; Huang, T.; Tang, Z.; Chen, Y.; Li, J.; Luo, D. Radar signal intrapulse modulation recognition based on a denoising-guided disentangled network. *Remote Sens.* **2022**, *14*, 1252. [CrossRef]
21. Wang, Y.; Yao, Q.; Kwok, J.T.; Ni, L.M. Generalizing from a few examples: A survey on few-shot learning. *ACM Comput. Surv.* **2020**, *53*, 63. [CrossRef]
22. Weiss, K.; Khoshgoftaar, T.M.; Wang, D. A survey of transfer learning. *J. Big Data* **2016**, *3*, 9. [CrossRef]
23. Zhu, Y.; Chen, Y.; Lu, Z.; Pan, S.J.; Xue, G.-R.; Yu, Y.; Yang, Q. Heterogeneous Transfer Learning for Image Classification. In Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 7–11 August 2011.
24. Zhou, J.; Pan, S.; Tsang, I.; Yan, Y. Hybrid Heterogeneous Transfer Learning through Deep Learning. In Proceedings of the AAAI Conference on Artificial Intelligence, Québec, Canada, 27–31 July 2014.
25. Wang, Q.; Du, P.; Yang, J.; Wang, G.; Lei, J.; Hou, C. Transferred deep learning based waveform recognition for cognitive passive radar. *Signal Process.* **2019**, *155*, 259–267. [CrossRef]
26. Guo, Q.; Yu, X.; Ruan, G. LPI radar waveform recognition based on deep convolutional neural network transfer learning. *Symmetry* **2019**, *11*, 540. [CrossRef]
27. Lin, A.; Ma, Z.; Huang, Z.; Xia, Y.; Yu, W. Unknown radar waveform recognition based on transferred deep learning. *IEEE Access* **2020**, *8*, 184793–184807. [CrossRef]
28. Si, W.; Wan, C.; Zhang, C. Towards an accurate radar waveform recognition algorithm based on dense CNN. *Multimed. Tools Appl.* **2021**, *80*, 1779–1792. [CrossRef]
29. Chen, T.; Kornblith, S.; Norouzi, M.; Hinton, G. A Simple Framework for Contrastive Learning of Visual Representations. In Proceedings of the International Conference on Machine Learning, Online, 13–18 July 2020.
30. He, K.; Fan, H.; Wu, Y.; Xie, S.; Girshick, R. Momentum Contrast for Unsupervised Visual Representation Learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Washington, DC, USA, 14–19 June 2020.
31. Tian, Y.; Krishnan, D.; Isola, P. Contrastive Multi-View Coding. In Proceedings of the European Conference on Computer Vision, Glasgow, UK, 23–28 August 2020.
32. Liu, D.; Wang, P.; Wang, T.; Abdelzaher, T. Self-Contrastive Learning Based Semi-Supervised Radio Modulation Classification. In Proceedings of the MILCOM 2021–2021 IEEE Military Communications Conference (MILCOM), San Diego, CA, USA, 29 November–2 December 2021.
33. Devlin, J.; Chang, M.-W.; Lee, K.; Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv* **2018**, arXiv:1810.04805.
34. Gidaris, S.; Singh, P.; Komodakis, N. Unsupervised representation learning by predicting image rotations. *arXiv* **2018**, arXiv:1803.07728.
35. Hou, S.; Shi, H.; Cao, X.; Zhang, X.; Jiao, L. Hyperspectral imagery classification based on contrastive learning. *IEEE Trans. Geosci. Remote Sens.* **2022**, *60*, 1–13. [CrossRef]
36. Jing, L.; Tian, Y. Self-supervised visual feature learning with deep neural networks: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **2020**, *43*, 4037–4058. [CrossRef]
37. Zhang, R.; Isola, P.; Efros, A.A. Colorful Image Colorization. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 8–16 October 2016.
38. Shrivastava, A.; Gupta, A.; Girshick, R. Training Region-Based Object Detectors with Online Hard Example Mining. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 26 June–1 July 2016.
39. Lin, T.-Y.; Goyal, P.; Girshick, R.; He, K.; Dollár, P. Focal Loss for Dense Object Detection. In Proceedings of the IEEE International Conference on Computer Vision, Honolulu, Hawaii, 22–25 July 2017.
40. Tian, X.; Wu, D.; Wang, R.; Cao, X. Focal text: An Accurate Text Detection with Focal Loss. In Proceedings of the 2018 25th IEEE International Conference on Image Processing (ICIP), Athens, Greece, 7–10 October 2018.
41. Chen, M.; Fang, L.; Liu, H. Fr-net: Focal Loss Constrained Deep Residual Networks for Segmentation of Cardiac MRI. In Proceedings of the 2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019), Venezia, Italy, 8–11 April 2019.
42. Zimmermann, M.; Dostert, K. Analysis and modeling of impulsive noise in broad-band powerline communications. *IEEE Trans. Electromagn. Compat.* **2002**, *44*, 249–258. [CrossRef]
43. Clavier, L.; Peters, G.W.; Septier, F.; Nevat, I. Impulsive noise modeling and robust receiver design. *EURASIP J. Wirel. Commun. Netw.* **2021**, *2021*, 13. [CrossRef]
44. Zhou, Z.; Huang, G.; Chen, H.; Gao, J. Automatic radar waveform recognition based on deep convolutional denoising auto-encoders. *Circuits Syst. Signal Process.* **2018**, *37*, 4034–4048. [CrossRef]
45. Wan, J.; Yu, X.; Guo, Q. LPI radar waveform recognition based on CNN and TPOT. *Symmetry* **2019**, *11*, 725. [CrossRef]
46. Lim, J.S. *Two-Dimensional Signal and Image Processing*; Prentice-Hall: Englewood Cliffs, NJ, USA, 1990.
47. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 26 June–1 July 2016.
48. OShea, T.J.; Roy, T.; Clancy, T.C. Over-the-air deep learning based radio signal classification. *IEEE J. Sel. Top. Signal Process.* **2018**, *12*, 168–179. [CrossRef]

49. Qi, P.; Zhou, X.; Zheng, S.; Li, Z. Automatic modulation classification based on deep residual networks with multimodal information. *IEEE Trans. Cogn. Commun. Netw.* **2020**, *7*, 21–33. [CrossRef]
50. Lu, X.; Tao, M.; Fu, X.; Gui, G.; Ohtsuki, T.; Sari, H. Lightweight Network Design Based on ResNet Structure for Modulation Recognition. In Proceedings of the 2021 IEEE 94th Vehicular Technology Conference (VTC2021-Fall), Online, 27–30 September 2021.
51. Liao, K.; Zhao, Y.; Gu, J.; Zhang, Y.; Zhong, Y. Sequential convolutional recurrent neural networks for fast automatic modulation classification. *IEEE Access* **2021**, *9*, 27182–27188. [CrossRef]
52. Cortes, C.; Vapnik, V. Support-vector networks. *Mach. Learn.* **1995**, *20*, 273–297. [CrossRef]
53. Cover, T.; Hart, P. Nearest neighbor pattern classification. *IEEE Trans. Inf. Theory* **1967**, *13*, 21–27. [CrossRef]
54. Keskar, N.S.; Mudigere, D.; Nocedal, J.; Smelyanskiy, M.; Tang, P.T.P. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv* **2016**, arXiv:1609.04836.