SAR Targets Classification Based on Deep Memory Convolution Neural Networks and Transfer Parameters

Ronghua Shang, *Member, IEEE*, Jiaming Wang, Licheng Jiao, *Fellow, IEEE*, Rustam Stolkin, *Member, IEEE*, Biao Hou, *Member, IEEE*, and Yangyang Li, *Member, IEEE*

***Abstract*—Deep learning has obtained state-of-the-art results in a variety of computer vision tasks and has also been used to solve SAR image classification problems. Deep learning algorithms typically require a large amount of training data to achieve high accuracy. In contrast, the size of SAR image datasets is often comparatively limited. Therefore, this paper proposes a novel method, deep memory convolution neural networks (M-Net), to alleviate the problem of overfitting caused by insufficient SAR image samples. Based on the convolutional neural networks (CNN), M-Net adds an information recorder to remember and store samples' spatial features, and then it uses spatial similarity information of the recorded features to predict unknown sample labels. M-Net's use of this information recorder may cause difficulties for convergence if conventional CNN training methods were directly used to train M-Net. To overcome this problem, we propose a transfer parameter technique to train M-Net in two steps. The first step is to train a CNN, which has the same structure as the part of CNN incorporated in M-Net, to obtain initial training parameters. The second step applies the initialized parameters to M-Net and then trains the entire M-Net. This two-step training approach helps us to overcome the nonconvergence issue, and also reduces training time. We evaluate M-Net using the public benchmark MSTAR dataset, and achieve higher accuracy than several other well-known SAR image classification algorithms.**

***Index Terms*—Deep learning, memory convolutional neural networks (M-Net), parameter transfer, synthetic aperture radar (SAR) targets classification.**

I. Introduction

SYNTHETIC aperture radar (SAR) is not restricted by light conditions, climate, and some other environmental factors. It can produce high-resolution images in day and night and all-weather operating conditions, and has some ability to penetrate obstacles in the earth's surface. SAR images have many applications, including resource exploration, military reconnaissance, surveillance, target acquisition, and other aspects. Nevertheless, SAR images are very sensitive to the pose and configuration parameters of targets. The SAR imaging process SAR tends to generate speckle noise, which causes difficulties in SAR image classification [1]–[3].

Over the past few decades, many algorithms have been proposed to solve the SAR image classification problem. Here, we discuss the SAR image classification literature in terms of conventional approaches versus more recent approaches based on deep learning.

Conventional approaches are many and diverse. Early work used template-based methods [4], [5]. These algorithms achieve satisfactory results when targets have consistent appearance, but they fail when target appearance is highly variable. With the emergence of machine learning approaches, many algorithms, such as support vector machine (SVM) with Gaussian kernel [6], AdaBoost [7], and many others, have also been applied to the SAR image classification problem. These methods are designed to construct a powerful classifier to distinguish between training samples from different classes. In contrast, other approaches, such as sparse representation [8], [9], scattering center models [10], [11], nonnegative matrix factorization [12], Riemannian manifold [13], discriminant embedding [14], have investigated the benefits of different feature representations of image data. These algorithms focus on extracting highly discriminatory features from the original data, to make the whole classification process easier, and improve the classification results. In addition, there are algorithms that are based on mathematical model, for instance, hidden Markov model [15], iterative graph thickening [16], conditional Gaussian method [17], and others. In summary, these conventional methods have many advantages and yield useful results, but also have shortcomings. They must establish complicated classification systems or elaborately designed "hand-crafted" feature extractors, requiring large effort from human experts. Hand-crafted classifiers and feature extractors which work well on one dataset may not generalize robustly to a different dataset. The entire architecture may need to be redesigned when processing different datasets.

An alternative approach to target classification is based on deep learning algorithms, which have made remarkable

Manuscript received December 6, 2017; revised April 7, 2018; accepted May 8, 2018. Date of publication June 6, 2018; date of current version August 21, 2018. This work was supported in part by the National Natural Science Foundation of China under Grant 61773304, Grant 61671350, Grant 61371201, and Grant 1772399. *(Corresponding author: Ronghua Shang.)*

R. Shang, J. Wang, L. Jiao, B. Hou, and Y. Li are with the Key Laboratory of Intelligent Perception and Image Understanding of Ministry of Education, International Research Center for Intelligent Perception and Computation, School of Artificial Intelligence, Xidian University, Xi'an 710071, China (e-mail: rhshang@mail.xidian.edu.cn; tjnuwjm@gmail.com; lchjiao@mail.xidian.edu.cn; avcodec@hotmail.com; yyli@xidian.edu.cn).

R. Stolkin is with the Extreme Robotics Lab, University of Birmingham, Birmingham B15 2TT, U.K. (e-mail: R.Stolkin@cs.bham.ac.uk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JSTARS.2018.2836909

achievements in solving many kinds of image classification problems in recent years. The convolutional neural network (CNN) is a very popular deep learning framework [18]. In 1990s, Lecun *et al*. proposed a multilayer neural network called LeNet establishing the modern structure of CNN [19]. Compared to multilayer fully connected neural networks, local connections and weights sharing in LeNet greatly reduce the number of parameters, making training easier and mitigating overfitting. In 2012, Krizhevsky *et al*. created a classic CNN model—AlexNet [20] that made a great breakthrough in the ImageNet picture classification task, reducing then the error rate from the state-of-the-art 26.2% down to 15.3%. In later work, researchers also invented other improved algorithms: ZFNet [21], VggNet [22], GoogleNet [23], and ResNet [24].

With the growing prominence of deep learning, it has also started being applied to solve SAR image classification problems. In contrast to the conventional algorithms mentioned above, CNN is an "end-to-end" learning approach, which does not need to separate the problem into feature extraction and classification. Instead, it automatically extracts hierarchical features of images from a large amount of data [25]. CNN can also directly use original raw images as input, which omits complex and cumbersome data preprocessing work to obtain useful features. However, CNN's can be prone to severe overfitting when training datasets are relatively small.

The moving and stationary target acquisition and recognition (MSTAR) is an open military target dataset [26]. Many researchers regard MSTAR as a benchmark for comparative performance evaluation. Unfortunately, because of the scarcity and concealment of military targets, as well as some unknown parameters of the SAR imaging mechanism [7], MSTAR dataset has only about 200 training samples per class in a ten-target classification challenge. This small amount of samples presents an obstacle for applying deep learning algorithms to MSTAR data. To overcome this problem, many variations of normal deep learning algorithms have been proposed. Deng *et al*. proposed an autoencoder with Euclidean distance (SAEED) [27] to extract linear separable features. Ma *et al*. [28] proposed a spatial updated deep autoencoder and a representation-based classification method. Zhong *et al*. [29] proposed the large patch convolutional neural network, replacing fully connected layers with global average pooling layers. Chen *et al*. proposed all-convolutional networks [30] which remove full-connected layers from a conventional CNN to reduce the number of parameters and strengthen general performance of the network.

Humans demonstrate remarkable capabilities to accurately classify an image they have never seen, after learning from only a few training images. In contrast, CNN needs to learn from thousands of ground-truthed training images, i.e., CNN is inefficient in the utilization of training data. Some researchers have experimented with introducing a "memory module" [31] in deep neural networks to reduce the networks' dependence on rich training data while still generating useful results. We, therefore, suggest that a memory module approach might be useful for the SAR target image classification task, in which only a small amount of training data is typically available. This paper proposes improvements on the basis of classical CNN and memory modules and designs a new deep memory convolutional neural network (M-Net). Additionally, a novel training method is proposed to train M-Net, inspired by transfer learning [32] and layer-wise greedy training methods [33].

Our objectives in designing M-Net are to extract more information from a small amount of training data, namely the spatial similarity information, and try to extract the underlying statistics of each class. There is an information recorder built into M-Net similar to the memory module aforementioned. The information recorder can discover distinctive and intrinsic connections between training samples, by analyzing the spatial similarity information of features. M-Net first converts raw image data to features using its front-end CNN component. Next, the extracted features are delivered into M-Net's information recorder to query their true label. The spatial cosine similarity, between the input feature and each record stored in the information recorder, is independently calculated in the query process. Larger spatial cosine similarity represents a greater likelihood that the input feature and the corresponding record belong to same class, and the record which generates the largest similarity with the incoming feature will be copied. Eventually, M-Net outputs the copied record's value as a query result.

The real-world distortions and variations in the appearance of, e.g., a target vehicle can significantly influence the accuracy of classification. Hence, robust generalization is a highly desirable capability for a SAR classification algorithm. M-Net's novel structure is conducive for efficiently extracting general features, enabling improved robustness. However, this special structure means that, if M-Net were trained in a conventional way, early stages of training might become unstable (divergent), and convergence would be slow. To achieve fast network training, and convergence on the globally optimal set of trainable parameters, we propose a novel two-step training method. The first step is to train a conventional CNN with the same convolution structure as M-Net. In the second step, the trained parameters from the first step are imported into M-Net which then undergoes further training, i.e., the parameters are pretrained in a conventional CNN, to create initial values of parameters for fine-tuning training inside M-Net. This two-step training approach, by transferring parameters, makes M-Net's training faster and more efficient.

The remaining part of this paper is arranged as follows. Section II introduces the structure and training methods of M-Net. Section III describes experiments on benchmark datasets. Section IV summarizes our contributions and provides concluding remarks.

II. Methodology

A. Structure of M-Net

The structure of M-Net is composed of three parts: basic CNN, mapping matrix, and information recorder, which is shown in Fig. 1.

The overall process, from input of SAR images to output of classifications, can be divided into three stages. In the first stage, the CNN part automatically extracts feature vectors from high-dimensional image data. The dimension of raw image data is greatly reduced, and the most useful information for

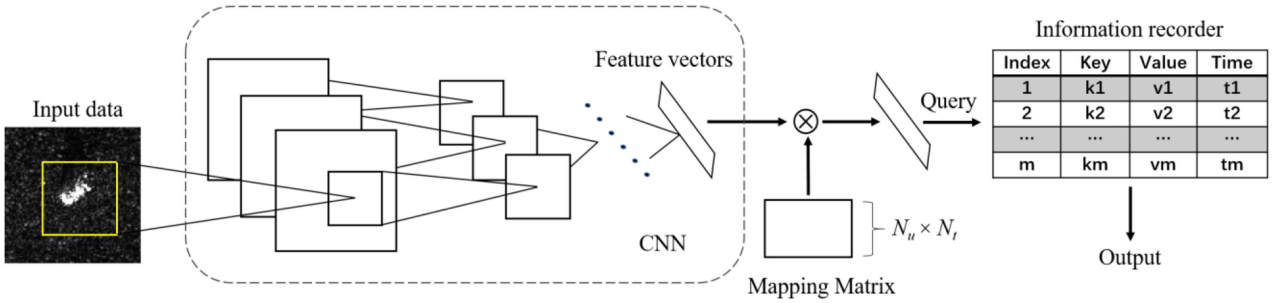


Fig. 1. Whole framework of M-Net. N_u is the length of feature vectors, and N_t is the length of the information recorder keys.

TABLE I
STRUCTURE AND PARAMETERS CONFIGURATION OF THE CNN COMPONENT

layer No.	structure	output feature size
1	Input layer	70 * 70 * 1
2	Conv. 16 @ 7*7 / ReLU	64 * 64 * 16
3	Po. 2 @ 2*2	32 * 32 * 16
4	Conv. 32 @ 5*5 / ReLU	28 * 28 * 32
5	Po. 2 @ 2*2	14 * 14 * 32
6	Conv. 64 @ 5*5 / ReLU	10 * 10 * 64
7	Po. 2 @ 2*2	5 * 5 * 64
8	Conv. 128 @ 3*3 / ReLU	3 * 3 * 128
9	Dropout layer	3 * 3 * 128
10	Conv. 10 @ 3*3	1 * 1 * 10
11	Output layer	1 * 10

Table I: Conv. (feature maps number) at (filter size)/(activation function)" represents a convolutional layer. "Po. (stride size) at (pooling size)" represents the max pooling layer. "$x \times y \times z$" represents the three dimensions of the feature.

classification is extracted. In the second stage, extracted feature vectors and a trainable mapping matrix are multiplied to obtain transformed feature vectors. In the final stage, transformed feature vectors enter the information recorder. Then, the system queries the records in the information recorder to find the records which have the smallest cosine distance with the input vectors. The following sections provide further details of each stage of M-Net.

B. M-Net's CNN Component

The CNN component comprises nine layers, as well as an input layer and an output layer. It includes five convolutional layers, three pooling layers, and one dropout layer. Its first five convolutional layers are all followed by a nonlinear activation function as a ReLU layer. The structure and configuration parameters of each layer in the CNN are shown in Table I.

In the input layer, a SAR image with random size is divided into 70×70 slices. On the one hand, the clipped slices with size of 70×70 are big enough to cover the identified target zones. On the other hand, the use of smaller input size is conducive to eliminate redundant parameters and decrease the depth of networks. In this way, the risk of overfitting is reduced and the speed of training and testing operations is increased. In cases where the target areas size is greater than 70×70, we can compress the target area to a 70×70 slice.

The main function of the convolutional layer is to filter the input data and acquire feature maps. Discrete convolution operations will be done in the convolutional layer and multilayer convolution operations are beneficial to get more deep features. If the lth ($l = 1, 2, \ldots, L$, where L is the number of layers) layer is a convolutional layer, the discrete convolution operation is defined as

$$Y_j^{(l)}(\mathrm{a},\mathrm{b}) = \sum_{i=1}^{\mathrm{I}} \sum_{s=0}^{kw} \sum_{t=0}^{kh} W_{ij}^{(l)}(s,t) X_i^{(l)}(a-s, b-t) + B_j^{(l)} \tag{1}$$

where $X_i^{(l)}$ is the input data of the ith node ($i = 1, 2, \ldots, I$, where I is the total number of input images) in the lth layer with size of $W_X^{(l)} \times H_X^{(l)}$ and $Y_j^{(l)}$ is the output data of the jth ($j = 1, 2, \ldots, J$, where J is the total number of output feature maps) node of the lth layer with size of $W_Y^{(l)} \times H_Y^{(l)}$. The output nodes in the previous layer are connected to the input nodes in the next layer. $W_{ij}^{(l)}$ is the convolutional kernel with size of $kw \times kh$. Kernels with size of 3×3 or 5×5 are recommended because they can play a more effective role in feature extraction [21]–[23]. $B_j^{(l)}$ is the bias of the lth layer and both $W_{ij}^{(l)}$ and $B_j^{(l)}$ are trainable. Every convolution operation in the convolutional layer is without padding operation, and has strides with size of 1. Therefore, the relationship between the input size and the output size of the lth convolutional layer is defined as

$$\begin{aligned} W_Y^{(l)} &= \left(W_X^{(l)} - kw\right) + 1 \\ H_Y^{(l)} &= \left(H_X^{(l)} - kh\right) + 1. \end{aligned} \tag{2}$$

Convolutional kernels with three different kinds of size are used in convolutional layers. The size of convolutional kernels in five convolutional layers is 7×7, 5×5, 5×5, 3×3, and 3×3 and the numbers of convolutional kernels are 16, 32, 64, 128, and 10, respectively, i.e., low-level features in image are relatively few but high-level features are more. In a sophisticated network, the convolutional kernel size decreases gradually layer by layer, while the number of kernels increases, which can help us to extract more complete features from images [34].

It is known that only linear operations can be performed in convolutional layers, so it is necessary to introduce activation functions for nonlinear transformation after convolutional layers. The commonly used activation functions include the tanh function and sigmoid function, which can output an activated value between 0 and 1. Although these functions are useful for nonlinear transformation, their gradients tend to zero when they

saturate. This can lead to the "vanishing gradient problem" and even lead to a training failure. Applying ReLU [20], [21] as the activation function can avoid this problem. The ReLU function controls its gradient in a defined range rather than arbitrary range. The gradient equals 0 (if $x < 0$) and gradient equals 1 ($x > 0$). Another benefit of ReLU is that it can accelerate the learning rate and the convergence process. If the *l*th layer is the active layer, the operation is shown as

$$Y_j^{(l)} = f(X_i^{(l)}) = \max(0, X_i^{(l)}) \tag{3}$$

where *f*(*x*) represents the ReLU activation function. The third, fifth, and seventh layers of the network are pooling layer also called as subsampling layer. The pooling layer can also be regarded as a nonlinear change. It outputs the max value point in the given local range. Two kinds of common pooling methods are average pooling and max pooling. The performance of max pooling is generally better than the average pooling [35]. The main function of max pooling is to rapidly reduce features' dimension, and overcome the influence of small target variations, such as shifting and scaling. If the *l*th layer is a max pooling layer, the operation is defined as

$$Y_i(x, y) = \max_{u,v \in \{0,\ldots,pz\}} X_i(x \bullet s + u, y \bullet s + v) \tag{4}$$

where *pz* is the max pooling size and *s* is the max pooling stride. Both *s* and *pz* are set to 2. With this configuration, the size of input data will be half after going through max pooling layer. Based on previous experience, max pooling size is generally 2×2 or 3×3 and stride is 2. Overly large max pooling size and stride can result in information loss.

The ninth layer is a dropout layer [36]. Dropout is a simple and cost-effective technology to prevent overfitting. Before training a network in each round, dropout randomly selects some units and inactivates them. These inactivated units are not trained and updated during this round but their weights are retained. Dropout technology can be seen as a kind of ensemble method, which is equivalent to the combination of many weak neural networks. It has been proven that dropout can improve neural networks' generalization ability and many researchers currently believe it is a necessary choice. In M-Net, the dropout layer is inserted before a convolutional layer. If the *l*th layer is a dropout layer, the calculation formula is as follows:

$$\begin{aligned} r_i &\sim \text{Bernoulli}(p) \\ \tilde{X}_i^{(l)} &= r_i \bullet X_i^{(l)} \\ Y_j^{(l)}(a, b) &= \sum_{i=1}^{\mathrm{I}} \sum_{s=0}^{kw} \sum_{t=0}^{kh} W_{ij}^{(l)}(s, t) \tilde{X}_i^{(l)}(a - s, b - t) + B_j^{(l)} \end{aligned} \tag{5}$$

where r_i is an independent Bernoulli variable, which equals 1 in probability of *p* and equals 0 in probability of 1-*p*. We set *p* to a usual 0.5. The input data of the *i*th node in the *l*th layer *X(l) i* multiply with r_i to get an amended value $\tilde{X}_i^{(l)}$, and $\tilde{X}_i^{(l)}$ replace $X_i^{(l)}$ as the input of next convolutional layer. After several convolution and pooling operations, an input data $X_i^{(l)}$ with size of $W_X^{(l)} \times H_X^{(l)}$ are finally converted to a feature map $Y_j^{(L)}$ which is composed by matrix with size of $N_u \times 1 \times 1$ ($Nu \in N^+$). As there is no fully connected layer in M-Net's structure, N_u is determined by the kernels number in the last convolutional layer. For convenient computing, the shape of $Y_j^{(L)}$ is transformed from $N_u \times 1 \times 1$ to N_u and the transformed $Y_j^{(L)}$ is assigned to a vector variable P_u which is the feature vector aforementioned.

TABLE II
STRUCTURE OF INFORMATION RECORDER

INDEX	KEY	VALUE	TIME
1	k_1	v_1	t_1
2	k_2	v_2	t_2
…	…	…	…
m	k_m	v_m	t_m

C. Mapping Matrix

In order to keep the size of the information recorder fixed at a suitable value, we use a mapping matrix to connect the CNN and the information recorder. When the dimension of the feature vector P_u changes, we can adjust the size of the mapping matrix to leave the structure of the information recorder unchanged. When the dimension of the feature vector is very large, this approach can also help us to reduce the dimension. The mapping matrix can be regarded as a fully connected layer. The only difference is that the fully connected layer is trained in both of our two learning steps, while the mapping matrix is only trained in the second step. In our network, the length of the feature vector is the same as that of the information recorder's values. Therefore, one can discard the mapping matrix with only a very small impact on the result.

The feature vector P_u is multiplied by the mapping matrix M_p and get $Q_u = M_p \cdot P_u$ where Q_u is a vector with size of N_t. M_p is a 2-D matrix with size of $N_u \times N_t$ and can also be updated by training.

D. Information Recorder

The information recorder is an improved version of the memory module described in [31], with several differences in structure. Our information recorder contains four parts: INDEX, KEY, VALUE, and TIME. Each part has its own special function and the architecture is shown in Table II.

In the information recorder, each line corresponds to a record. Each record is composed of four sections. The first is an index item represented by a positive integer z ($z = 1, 2, \ldots, m$). It can provide convenience for users in writing and searching messages, and generates a unique identification for every record. The second section is the key item represented by a float vector k_z ($||k_z|| = 1$) with size of N_t. The third section is the value item represented by an integer v_z. v_z is the true class label of k_z, thus its reasonable value range is determined by the number of classes in the data, e.g., if the dataset to be classified has 10 categories, $v_z \in (0, 1, \ldots, 9)$. The fourth section is a time parameter represented by an integer t_z. It stores the time at which each record was last updated. Each time a round of

updating parameters is completed, each corresponding time parameter will be updated accordingly.

The querying operation is very simple in information recorder. When the transformed feature Q_u enters the information recorder, Q_u is normalized first by setting its two-norm value to 1, that is, $||Q_u|| = 1$. Next, we calculate the spatial similarity between Q_u and each record in the information recorder. The spatial similarity is measured by cosine distance shown as

$$\cos(\theta)_z = \frac{Q_u \bullet k_z}{\|Q_u\| \|k_z\|}. \tag{6}$$

Because $||Q_u|| = 1$ and each record key $||k_z|| = 1$, (6) is transformed as

$$\cos(\theta)_z = Q_u \bullet k_z. \tag{7}$$

With (7), we can multiply m records' key $(k_1, k_2, \ldots, k_m)$ in the information recorder with the feature vector Q_u to get a similarity array $(\cos(\theta)_1, \cos(\theta)_2, \cdots, \cos(\theta)_m)$. The maximum value of this array will be chosen and its index is written as $z_{\max}$

$$z_{\max} = \text{agrmax}(\cos(\theta)_1, \cos(\theta)_2, \cdots, \cos(\theta)_m) \tag{8}$$

where argmax(x) function returns the index of the maximum value record in the input array. After obtaining $z_{\max}$, we query the corresponding record's value $v_{z_{\max}}$, and $v_{z_{\max}}$ is output as the final output label of the input feature Q_u.

When training the network and updating the parameters, for a given feature Q_u, it is assumed that Q_u corresponds to an original sample pair (x_q, y_q), where y_q is the true label of image data x_q. We classify all records in the information recorder to two collections. Records which have a different label as compared with y_q are incorporated into the set S_-. All other records are placed in a set S_+. According to whether S_- or S_+ is empty, there are three cases.

In the first case, neither S_- nor S_+ is empty. From (7), (8), we calculate the maximum positive similarity h_+ between the records in S_+ and Q_u, and the minimum negative similarity h_- between the records in S_- and Q_u. The loss function is calculated in accordance with h_+ and h_-. The loss function here is the hinge-loss function, shown as follows:

$$Ls(W) = \max(\alpha - |h_+ - h_-|, 0). \tag{9}$$

The usage of hinge loss is to maximize the distance of records in S_+ and S_-. When the distance in h_+ and h_- is greater than the threshold α, the loss becomes zero. If the distance is less than the threshold α, it means the distance is not big enough, and the loss will be negatively correlated with the distance. With the loss function, the backpropagation algorithm can be used to calculate the gradient.

Note that the dynamic parameter updating in the information recorder is not implemented by error backpropagation. When updating the information recorder parameters, the first thing is to compare the value of h_+ and h_-.

If $h_+ > h_-$, the current feature Q_u and the h_+ corresponding record will be integrated as a new key, and then the key will also be normalized. The key k_{z_+} will be updated

$$k_{z_+} = \frac{Q_u + k_{z_+}}{\|Q_u + k_{z_+}\|},\ v_{z_+} = y_q \tag{10}$$

where z_+ and z_- are the indices of the corresponding records h_+ and h_-. If $h_+ < h_-$, the oldest record in the information recorder should be found and be replaced with the current feature vector Q_u. The oldest record's value will be updated to the label of Q_u, namely y_q

$$k_{z_o} = Q_u,\ v_{z_o} = y_q \tag{11}$$

where z_o is the oldest record's index. For adding some randomness to the system, a time range will be given. z_o will be randomly set in this given range, that is,

$$z_o = \arg(\text{rand}(\max(t_z) - t_s, \max(t_z))) \tag{12}$$

where t_s is a small interval denoting for the width of the given range. The function rand(x) returns a random integer based on incoming parameters. The function arg(x) returns the subscript of a record. When a round of updating is finished, the nonupdated record's time value will automatically increment $t_z = t_z + 1$ and the updated record's time value will be cleared to zero $t_z = 0$.

In the second case, S_+ is empty but S_- is not empty or both of them are empty. We use (10) to update the parameters in the information recorder.

In the last case, S_- is empty and S_+ is not empty. We use (11) to update the parameters in the information recorder.

E. Training Methods of M-Net

In the conventional CNN, a softmax classifier will be directly connected after the last layer of the CNN as an output layer. The output layer can produce classification result which will be used to calculate network's loss during training. After that the loss will be backpropagated from the back layers to the front layers. At the same time, the partial derivatives of each trainable parameter will be computed and conserved. These partial derivatives are used to get gradients and update the trainable parameters. M-Net is quite different from CNN in structure and training form. If we directly train M-Net just as training CNN, it is likely that M-Net cannot converge or its convergence speed is very slow. This paper presents a new transfer parameter method to train M-Net. This method has two steps. The first step is to train the conventional neural network with the same structure as M-Net's CNN. After the first step being completed, the weights and biases of each layer are saved. In the second step, the saved parameters are transferred to M-Net where they become the initial parameter values for the M-Net training. Next, the M-Net is further trained with fine tuning to find the global optimum point of hinge loss. This two-step training method can reduce the training time and make the training more stable. The following section explains some details of the training.

F. First Step of Training

First, a normal CNN is constructed which has the same structure as M-Net, without the mapping matrix and information

recorder, but including a softmax classifier added after the CNN. This structure is convenient for enabling direct transfer of parameters. Softmax is a multiclass classifier, which can output the posterior probability of each class, and the sum of all the probabilities is 1. If the *l*th layer is softmax, then

$$p_i = \frac{\exp(X_i^{(l)})}{\sum_{j=1}^{r} \exp(X_j^{(l)})} \tag{13}$$

where *r* is the number of the classes. p_i represents the probability that a sample is assigned to class i ($i = 0, 1, 2, \ldots, r-1$), while $p_i = P(y^{(j)=}i|x^{(i)}; W)$.

With the output of softmax classifier, the next step is to calculate the loss function. A commonly used loss function is the cross entropy loss function, which is good at reflecting the difference between real outputs and predicted outputs. It is often better than mean square loss in practice. Supposing that there is a fixed sample set $\{(x^{(1)}, y^{(1)}), \ldots, (x^{(C)}, y^{(C)})\}$, which contains *C* examples, network's cross entropy loss function can be shown as follows:

$$L(W) = -\frac{1}{C}\sum_{j=1}^{C}\sum_{i=1}^{r} G_i^{(j)} \bullet \log(P(y^{(j)} = i|x^{(j)}; W)) \tag{14}$$

where $G_i^{(j)}$ denotes whether the true label $y^{(j)}$ of $x^{(j)}$ equal to *i* or not. If equal, $G_i^{(j)}$ is set to 1 otherwise it will be 0. Because $p_{i=}P(y^j = i|x^i; W)$, (14) can also be written as

$$L(W) = -\frac{1}{C}\sum_{j=1}^{C}\sum_{i=1}^{r} G_i^{(j)} \bullet \log\left(p_i^{(j)}\right). \tag{15}$$

It is also possible to add a regularization item to the loss [23] with the aim of reducing the magnitude of weights and preventing overfitting.

When the loss is obtained, the backpropagation algorithm [37], [38] can be used to transfer the cost information from the network's back to its front. With backpropagation, we can measure the partial derivative of each training parameter with respect to cross entropy loss.

Additionally, an optimization method is needed to update the parameters for finding the minimum of a loss function. Commonly used optimization algorithms include stochastic gradient descent [39], momentum, and others. However, these algorithms' learning rates need to be manually adjusted and the work often is quite cumbersome and depends on the experience of researchers. To overcome these limitations, optimization algorithms with adaptive learning rates have emerged, such as Adagrad [40], Adam [41], etc. The Adam algorithm is used in M-Net. Adam is less memory intensive and can adopt different adaptive learning rates for different data. Adam also is good at optimizing problems with large datasets or high-dimensional data spaces. It is applicable to most nonconvex optimization problems. The training process and the detailed parameter settings are shown in Table III.

The initial parameter points can determine whether the training algorithm converges or not. Some initial points are very unstable and make the algorithm encounter numerical problems and even fail completely. When the parameters of the network are learning and converging, the initialization values can determine the speed and the cost of convergence. In addition, similar parameter values can have significantly different generalization errors. The initialization values can also affect the generalization, and Glorot *et al.* [42] suggested to use a standard initialization in convolutional layers as

$$W \sim \mathrm{U}\left(-\sqrt{\frac{6}{n_i + n_o}}, \sqrt{\frac{6}{n_i + n_o}}\right) \tag{16}$$

where n_i is the input number and n_o is the output number in each unit. *W* is a trainable weight of this unit and U means the uniform distribution.

TABLE III
ADAM ALGORITHM

Algorithm 1: Adam algorithm

Input: Step length ε =0.001.

Input: The exponential decay rate of the moment estimation, $\rho 1$ and $\rho 2$ belong to [0,1), where $\rho 1 = 0.9$, $\rho 2 = 0.999$.

Input: A small constant for numerical stability $\varphi = 10^{-8}$.

Input: Initial parameters *W*.

Initialize the time *t*=0 and initialize the first-moment and second-moment variables $m_t = 0$, $n_t = 0$.

while $t < 1.5 \times 10^6$ do:

m random samples are collected from the training set $\{(x_1, y_1), (x_2, y_2), \cdots, (x_m, y_m)\}$ (*m*=100).

calculate the gradient: $g_t = \frac{1}{m}\nabla_{\mathrm{W}_{ij}} \sum_i L(f(x_i; \mathrm{W}), y_i)$.

update the iterative number: $t = t + 1$.

update the partial first moment: $m_t = \rho_1 m_{t-1} + (1 - \rho_1) g_t$.

update the partial second moment: $n_t = \rho_2 n_{t-1} + (1 - \rho_2) g_t \odot g_t$.

amend the deviation of the first moment: $\hat{m}_t = \frac{m_t}{1 - \rho_1^t}$.

amend the deviation of the second moment: $\hat{n}_t = \frac{n_t}{1 - \rho_2^t}$.

calculate the update value: $\Delta \mathrm{W}_t = -\varepsilon \frac{\hat{m}_t}{\sqrt{\hat{n}_t} + \varphi}$.

apply the update: $\mathrm{W}_t = \mathrm{W}_t - \Delta \mathrm{W}_t$.

end while

G. Second Training Step and Transfer Parameters

When the first step of training is completed, the trained parameters of all convolutional layers are maintained. In the second training step, the entire M-Net will be trained (see Fig. 2).

The CNN in M-Net is initialized using the parameters learned during the first step of the training (see Fig. 2) instead of a standard normal distribution as is commonly used in deep learning. The mapping matrix is initialized by a Gaussian distribution with 0 mean and a standard deviation of 0.01. The parameters in M-Net's CNN and mapping matrix are updated in the same manner as the first step. For the information recorder, we empty it as an initial state. The training process of the information recorder does not rely on Adam and backpropagation. Its parameters are trained in the forward operation stage using its own criterion as shown in (10)–(12). The second training step also uses the Adam algorithm and backpropagation, except that

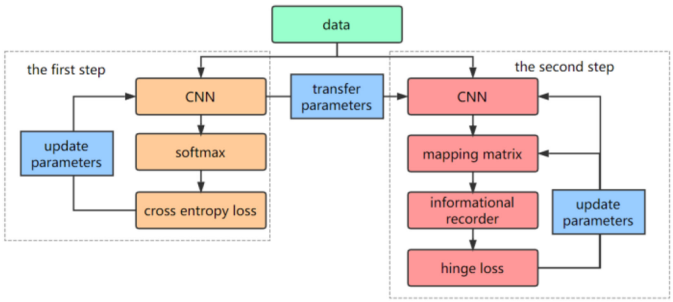


Fig. 2. Overall workflow of the two-step training process. Left and right sections denote the first and second training steps, respectively.

the loss function changes from the cross entropy loss (15) to the hinge loss (9). The initial learning rate becomes 0.0001 to accurately approximate the global optimum, but the other configurations of Adam are not changed. Although the two-step training method is more complicated than the direct training method, it consumes less training epoch, is more efficient, and also achieves good classification results.

III. Simulation Results and the Analysis

A. MSTAR Dataset

MSTAR is a project built by the US Defense Advanced Research Projects Agency and the US Air Force Research Laboratory. The image data were collected by the Sandia National Laboratory's X-band SAR target recognition system. All data is in 1-foot resolution under spotlight mode and covers full aspect (from 0° to 360°) [30]. The MSTAR dataset is often regarded as a benchmark for comparing performance of various SAR image classification algorithms. The current open data in MSTAR consist of 10 categories, including armored personnel carriers (four classes): BMP-2, BRDM-2, BTR-60, and BTR-70, tanks (two classes): T-62, T-72, rocket (one class): ZS-234, truck (one class): ZIL-131, and Bulldozer (one class): D7. Optical images and SAR images for each class are shown in Fig. 3.

B. Comparative Methods

To objectively evaluate the performance of M-Net, its performance on the MSTAR dataset is compared against several supervised machine learning and deep learning algorithms, including SVM, SAEED, and A-CovnNets.

C. Evaluation Metrics

We test the proposed algorithm on the ten-target classification problem under the standard test conditions (SOC) and extended operating condition (EOC) [16]. In the case of SOC, the training targets and test targets have the same class and serial type number, and the variation in depression angles is small. In the EOC experiment, the training data and test data have the same class, but the difference of depression angle between them is quite large. The variation of the depression angle has a huge impact on the performance of SAR image classification algorithm. A slight change of the depression angle may cause a serious decrease in accuracy. Therefore, the EOC experiment is useful for testing the robustness of algorithms.

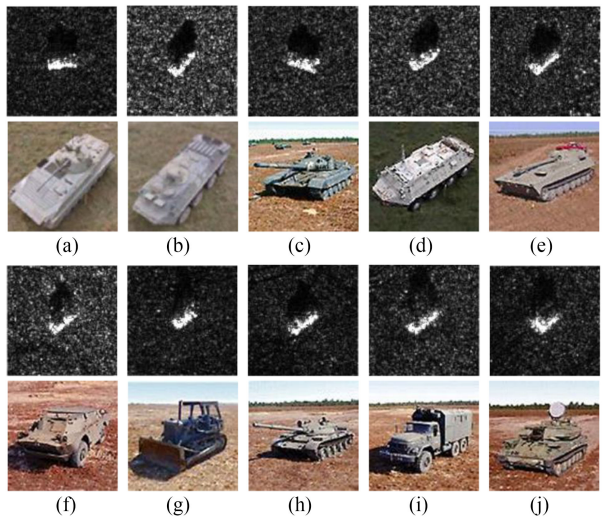


Fig. 3. Optical images and SAR images of 10 different types of military targets: (a) BMP-2; (b) BTR-70; (c) T-72; (d) BTR-60; (e) 2S1; (f) BRDM2; (g) D7; (h) T62; (i) ZIL131; and (j) ZSU234.

TABLE IV
Training Data and Test Data Used in the SOC Experiment

		Train		Test	
Class	Serial No.	Depression	Number	Depression	Number
BMP-2	9563	17°	233	15°	196
BTR-70	c71	17°	233	15°	196
T-72	132	17°	232	15°	196
BTR-60	k10yt7532	17°	256	15°	195
2S1	b01	17°	299	15°	274
BRDM-2	E-71	17°	298	15°	274
D7	92v13015	17°	299	15°	274
T62	A51	17°	299	15°	273
ZIL131	E12	17°	299	15°	274
ZSU234	d08	17°	299	15°	274

D. Parameter Settings

M-Net has two hyperparameters to be set, namely the threshold α of the hinge-loss function and the capacity size m of the information recorder, both of which may affect the performance of the network. Too small α will make the distance between different classes not big enough, while too large α will cause the network to become unstable. Too large m will increase the consumption of computing resources. Suitable values of α and m are very important. In order to analyze the exact impact of m and α on the network, an experiment is carried out on the training data in Table IV. Because training data are inadequate, data augmentation technology is used to generate more data, and then the augmented data are split into training samples (1/3 of total data) and validation samples (2/3 of total data). In order to obtain the optimal configuration, we use cross validation to tune the parameters.

The final results are shown in Fig. 4, where $m = \{500, 1000, 1500, 2000, 2500, 3000, 3500, 4000, 4500\}$ and $\alpha = \{0, 0.2, 0.4, 0.6, 0.8, 1.0, 1.2, 1.4, 1.6, 1.8, 2.0\}$. The changes in m and α affect the overall accuracy. When $m = 3000$

TABLE V
CONFUSION MATRIX OF SOC EXPERIMENT

Class	BMP-2	BTR-70	T-72	BTR-60	2S1	BRDM-2	D7	T62	ZIL131	ZSU234	P (%)
BMP-2	195	0	0	0	0	1	0	0	0	0	99.45
BTR-70	0	196	0	0	0	0	0	0	0	0	100
T-72	0	0	196	0	0	0	0	0	0	0	100
BTR-60	0	0	0	194	1	0	0	0	0	0	99.45
2S1	1	0	0	0	273	0	0	0	0	0	99.64
BRDM-2	0	0	1	0	0	273	0	0	0	0	99.64
D7	0	0	0	0	0	0	273	0	1	0	99.64
T62	0	0	0	0	0	0	0	273	0	0	100
ZIL131	0	0	0	0	1	0	0	0	272	1	99.27
ZSU234	0	0	0	0	0	0	0	0	0	274	100
Total											**99.71**

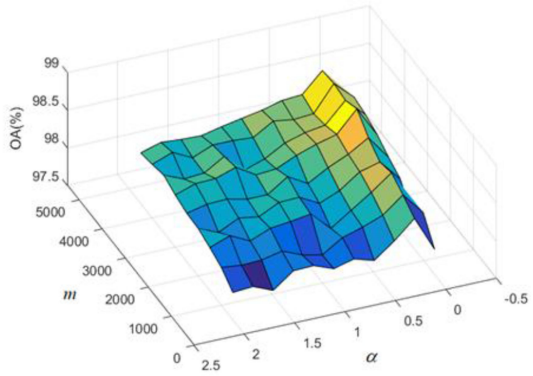


Fig. 4. OA (overall accuracy) under the change of the capacity size of the information recorder m and threshold α of hinge-loss function

and $\alpha = 0.2$, the overall accuracy reaches the maximum of 98.8%.

E. Results and Analysis of the SOC Experiment

In the SOC experiment, the training targets are captured under 17° depression angle and the test targets are captured under 15° depression angle. The training targets are more than the test targets in each class. All data used in this experiment are shown in Table IV.

There are ten categories of data, and each class only has approximately 200 images. This is a very small amount of training data compared to, e.g., the MNIST dataset which has more than 5000 images per class. Therefore, data augmentation is necessary. Data augmentation is a common method that produces more samples from a small amount of original data by shifting, rotating, and clipping images. This method is very effective in the case of insufficient data and can significantly improve the accuracy of classification. We standardize the raw SAR image and obtain the 70×70 clips that contain the targets by random clipping. After the data augmentation, the amount of training data is enough to train a M-Net with highly generalized performance. Furthermore, the original MSTAR data contain both amplitude and phase information of the SAR image, but we only use the amplitude information.

The number of training images is 2747 and the number of test images is 2426. The confusion matrix of SOC experiment is shown in Table V.

TABLE VI
TRAINING DATASET AND TEST DATASET OF EOC-1

	Train			Test		
Class	Serial No.	Depression	Number	Serial No.	Depression	Number
T-72	A64	17°	232	A64	30°	288
2S1	b01	17°	299	b01	30°	288
BRDM-2	E-71	17°	298	E-71	30°	287
ZSU234	d08	17°	299	d08	30°	288

TABLE VII
RESULTS OF EOC-1

Class	T-72	2S1	BRDM-2	ZSU234	P (%)
T-72	276	1	7	4	95.83
2S1	5	277	6	0	96.18
BRDM-2	0	2	285	0	99.30
ZSU234	4	0	0	284	98.61
Total					97.48

From Table V, we can see that the overall test accuracy under SOC reaches 99.71% and only seven test samples are misclassified. The accuracies of per class are all more than 99%. The four classes BTR-70, T-72, T62, and ZSU234 achieve accuracy of 100%. These results suggest that M-Net can achieve very good results on the SOC dataset.

F. Results and Analysis of EOC Experiment

The EOC dataset comprises a variety of data which is used for three different experiments, denoted EOC-1, EOC-2, and EOC-3.

In EOC-1, four kinds of data, 2S1, BRDM-2, T-72, and ZSU-234, are tested. Since the MSTAR dataset is limited, only these four classes of data could be used in this experiment. Therefore, EOC-1 is a four-target classification problem. EOC-1 has greater variability in depression angle between training data and test data. The training data are captured under depression angle of 17°, while the test data are captured under depression angle of 30°. Therefore, experimental results of EOC-1 reflect the robustness of the algorithm to change of depression angle. The training dataset and test dataset used in EOC-1 are shown in Table VI.

The confusion matrix and overall accuracies obtained in the EOC-1 experiment are shown in Table VII.

The accuracy of the EOC-1 experiment is somewhat worse than that of the SOC experiment. The change of the depression

TABLE VIII
TRAINING DATA OF EOC-2 AND EOC-3

	Train		
Class	Serial No.	Depression	No. Iamges
BMP-2	9563	17°	233
BRDM-2	E-71	17°	298
BTR-70	c71	17°	233
T-72	123	17°	232

TABLE IX
TEST DATA OF EOC-2

	Test		
Class	Serial No.	Depression	No. Iamges
T-72	S7	15°, 17°	419
	A32	15°, 17°	572
	A62	15°, 17°	573
	A63	15°, 17°	573
	A64	15°, 17°	573

TABLE X
TEST DATA OF EOC-3

	Test		
Class	Serial No.	Depression	No. Iamges
BMP-2	9566	15°, 17°	428
	c21	15°, 17°	429
T-72	812	15°, 17°	426
	A04	15°, 17°	573
	A05	15°, 17°	573
	A07	15°, 17°	573
	A10	15°, 17°	567

TABLE XI
RESULTS OF EOC-2

Class	Serial No.	BMP-2	BRDM-2	BTR-70	T-72	P (%)
T-72	S7	1	0	0	418	99.76
	A32	0	0	0	572	100
	A62	0	0	0	573	100
	A63	0	1	0	572	99.83
	A64	0	7	0	566	98.78
Total	99.67					

TABLE XII
RESULTS OF EOC-3

Class	Serial No.	BMP-2	BRDM-2	BTR-70	T-72	P (%)
BMP-2	9566	413	3	1	11	96.50
	c21	422	2	0	5	98.37
T-72	812	11	0	0	415	97.42
	A04	1	4	0	568	99.13
	A05	0	3	0	570	99.48
	A07	2	0	0	571	99.65
	A10	1	1	0	565	99.65
Total	98.46					

angle does have some influence on the result, as expected, however the accuracy is still high.

In the EOC-2 and EOC-3 experiments, we test the network's ability to distinguish objects which have similar appearance (e.g., BMP-2 and T-72). BMP-2 and T-72 are most similar in outline and shape and they are difficult to distinguish. Many algorithms confuse these two classes of target. Both EOC-2 and EOC-3 experiment use the same training dataset including four

TABLE XIII
COMPARISON OF DIFFERENT ALGORITHMS

Algorithm	SOC (%)	EOC-1 (%)	EOC-2 (%)	EOC-3 (%)
SVM [6]	91.1	80.65	78.86	58.67
SAEED [27]	93.6	91.51	90.48	85.49
A-CovNets [30]	99.09	96.09	98.86	98.35
Proposed method	**99.71**	**97.48**	**99.67**	**98.74**

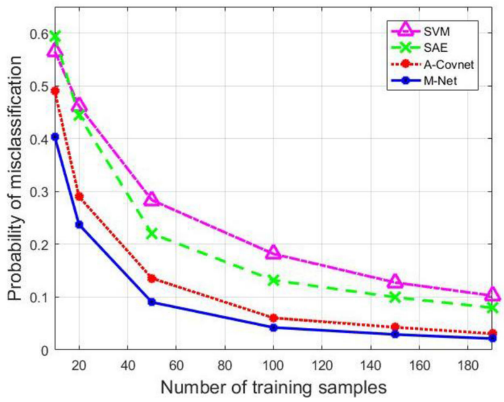


Fig. 5. Relationship between training sample number and SOC error rate.

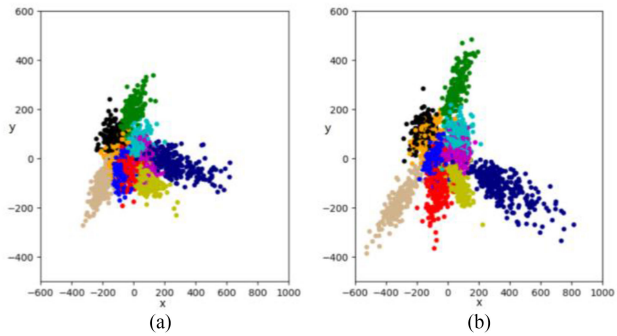


Fig. 6. Two-dimensional PCA projection of the extracted feature vectors: (a) Features extracted by CNN and (b) features extracted by M-Net. Different colors denote different classes of target.

TABLE XIV
COMPUTATIONAL COMPLEXITY AND TIME CONSUMPTION

Network	CNN	M-Net
Training computational complexity	$O(nl^2)$	$O(nl^2)$
Testing computational complexity	$O(nl^2)$	$O(nl^2)$
Training time per batch (in seconds)	0.6337	0.5963
Testing time per batch (in seconds)	0.1880	0.2029

classes: BMP-2, BRDM-2, BTR-70, and T-72. Their depression angles are all 17°, as shown in Table VIII.

The test dataset of the EOC-2 experiment is different from the EOC-3 experiment. The test dataset of EOC-2 has five different types of T-72, which are S7, A32, A62, A63, and A64 with depression angles of 15° or 17°, as shown in Table IX.

The test dataset of EOC-3 has five different types of T-72 (812, A04, A05, A07, A10), and two different types of BMP-2 (9566, c21). Their depression angles also are 15° or 17°, as shown in Table X.

There was no cross between the test dataset and the training dataset in both EOC-2 and EOC-3 experiments. The confusion

TABLE XV
VARIANT CNNS WITH DIFFERENT STRUCTURES

<table>
<tr><td colspan="7" rowspan="3">Conv.16@7*7</td><td>Conv.16@7*1</td><td>Conv.16@3*3</td></tr>
<tr><td>Conv.16@1*7</td><td>Conv.16@3*3</td></tr>
<tr><td></td><td>Conv.16@3*3</td></tr>
<tr><td colspan="9">ReLU + Po.2@2*2</td></tr>
<tr><td colspan="7" rowspan="2">Conv.32@ 5*5</td><td>Conv.32@5*1</td><td>Conv.32@3*3</td></tr>
<tr><td>Conv.32@1*5</td><td>Conv.32@3*3</td></tr>
<tr><td colspan="9">ReLU + Po.2@2*2</td></tr>
<tr><td colspan="7" rowspan="2">Conv.64@ 5*5</td><td>Conv.64@5*1</td><td>Conv.64@3*3</td></tr>
<tr><td>Conv.64@1*5</td><td>Conv.64@3*3</td></tr>
<tr><td colspan="9">ReLU + Po.2@2*2</td></tr>
<tr><td colspan="7">Conv.128@ 3*3</td><td>Conv.128@3*1
Conv.128@1*3</td><td>Conv.128@3*3</td></tr>
<tr><td colspan="9">ReLU + Dropout</td></tr>
<tr><td colspan="4">Conv.10@ 3*3</td><td>Conv.16@
3*3</td><td>Conv.32@
3*3</td><td>Conv.64@
3*3</td><td>Conv.128@
3*3</td><td>Conv.64@3*1
Conv.10@1*3</td><td>Conv.10@3*3</td></tr>
<tr><td rowspan="2">softmax</td><td rowspan="2">IR 3000*10</td><td colspan="7">MM N_u*10 (N_u= 10,16,32,64,128,10,10)</td></tr>
<tr><td colspan="7">IR 3000*10</td></tr>
</table>

The representation of convolutional and pooling layers is the same as those in Table I. "MM $N_u \times N_t$" represents a matrix with shape of $N u \times N_t$. "IR M × K" represents an information recorder of which capacity is M and key vector dimension is K.

matrix and the accuracy of EOC-2 on the test set are shown in Table XI.

The confusion matrix and the accuracy of EOC-3 on its test set are shown in Table XII.

The accuracies of EOC-2 and EOC-3, respectively, achieve 99.67% and 98.46%. The performance of M-Net is compared with that of several other well-known algorithms and the results of SOC, EOC-1, EOC-2, and EOC-3 experiments are shown in Table XIII.

As can be seen from Table XIII, the performance of M-Net in these experiments is better than that of the other algorithms'. In general, M-Net outperforms the other algorithms in accuracy, and also robustly handles variation in depression angle and different models of target vehicle. M-Net demonsrates strong ability to correctly distinguish between similar objects.

G. Variation in Performance With Different Numbers of Training Samples

Many algorithms can achieve good results in a particular dataset; however, when the amount of training data changes, the performance of the algorithm will be significantly affected. This can make satisfactory results difficult to obtain in practical applications where data may be limited. In order to analyze M-Net's sensitivity to variations in the quantity of training samples, we have evaluated M-Net's performance on the SOC experiment, when trained with different amounts of data. M-Net is compared against three other classifiers. The testing data used in this experiment are the same as the testing data in Table IV and the training data are randomly sampled from the training data in Table IV. To ensure realistic results, this experiment does not use data augmentation. The relationship between probability of error rate and the number of training samples is described in Fig. 5.

It can be seen from Fig. 5 that M-Net's error rate curve is always below then that of the three comparison algorithms, and achieves greater accuracies faster as learning progresses. The other curves perform significantly worse than M-Net when the number of training samples is reduced. M-Net generates the highest accuracy in every training sample number. When the number of training samples becomes greater than 100, M-Net's curve becomes relatively flat and smooth. This shows that, after the training data reach sufficient size, additional training data have a little effect on performance.

H. Feature Extraction

In the SOC experiment, the total number of samples in the test set is 2425. A feature vector of N_u dimensions is extracted from each image by M-Net. In order to visualize the feature vectors, principal component analysis (PCA) has been used to reduce the dimensionality of the feature vector from N_u dimensions to two dimensions [43] which can be depicted in a 2-D graph, as shown in Fig. 6.

Compared with a conventional CNN, M-Net extracts features for each class that are more widely separated from other classes. This effect is caused by adding the information recorder and using the hinge-loss function. The M-Net features are more discriminative and lead to greater robustness.

I. Computational Complexity and Time Consumption Analysis

In this section, we analyze the computational complexity and time consumption at different stages of normal CNN and M-Net. To facilitate the calculation, we assume that the structure of convolutional layers in the two networks does not change with the data. We assume that the amount of data is *n*, the height of every sample in data is *h,* and the width is *w* ($w = h = l$). Both the training and the testing computational complexity of the normal CNN are O(nl^2). While in forward operation of M-Net, computational complexities of convolutional layers, mapping matrix, and the information recorder are O(nl^2), O(nl^2), and O(n), respectively. Similarly, backward computational complexity operation of M-Net is O(nl^2). Therefore, both the training and the testing computational complexity of the M-Net are O(nl^2), which is the same as for a normal CNN.

Empirically, we have measured the training and testing time required for each batch (100 samples) using two different

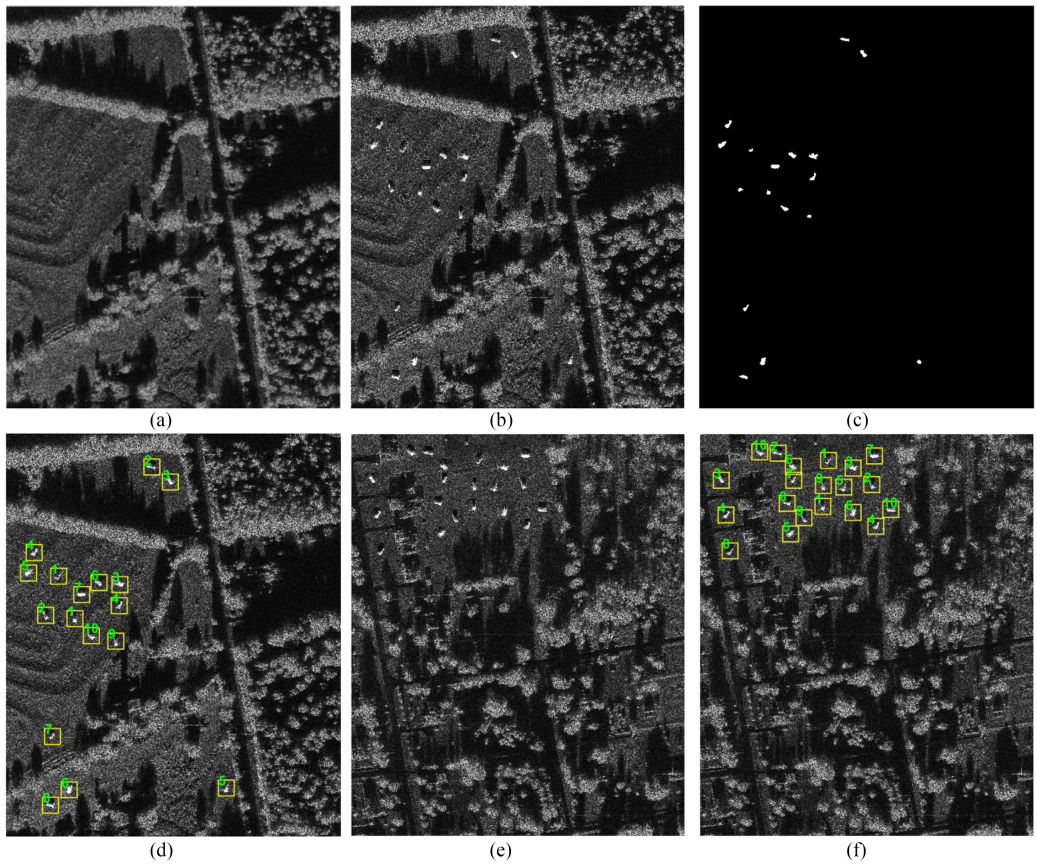


Fig. 7. (a) Clutter image. (b) Synthetic data with embedded targets. (c) Segmentation map after morphological operation. (d) Final results. The yellow boxes indicate the position of targets and the digits indicate the categories of the targets (green digit: correct; red digit: wrong; 1: BRDM-2; 2: 2S1; 3: D7; 4: T62; 5: ZIL131; 6: ZSU234; 7: BTR-60; 8: BMP2; 9: BTR-70; and 10: T-72). (e) and (f) Another clutter image and results.

networks. Results listed in Table XIV are obtained by tensorflow-cpu 1.0.1 and an intel i7-6700 2.60 GHz CPU.

J. Neural Networks With Different Structures

In this study, we have designed nine different networks to find the optimal network structure, as shown in Table XV.

In Table XV, C is an M-Net and A is an ordinary CNN. A and C have the same structure of CNN. B is a new network derived from configuration C without the mapping matrix. H and I change the convolutional kernel on the basis of C. H uses the structure of GoogleNet [23] to split one conventional kernel with size of $N \times N$ into two conventional kernels with size of $1 \times N$ and $N \times 1$ which can reduce the parameters of conventional kernels. I network imitates the structure of VggNet [21] to change the form of conventional kernels and split one conventional kernel with size of 7×7 to three conventional kernels with size of 3×3 and split one conventional kernel with size of 5×5 into two conventional kernels with size of 3×3. I can also reduce the parameters of convolutional layer. Both H and I reduce free parameters by changing the shape and number of convolutional kernels and the generalization performance is improved.

Table XVI gives the comparison results with different networks.

As can be seen in Table XVI, A's accuracy is lower than the C's. Without mapping matrix, B's accuracy has a little difference with C's. D, E, F, G, and C have similar structures, the only difference is the dimension of extracted features, where 16, 32, 64, and 128 dimensional features are extracted, respectively. The mapping matrix is used to map different dimensions of feature to a suitable fixed dimension. If the feature's dimension does not need to be changed, the mapping matrix can be removed (e.g., C and B). As can be seen from the table above, in both SOC and EOC, the accuracy of the D, E, and G networks is different as compared with M-Net. Although F acquires an accuracy as good as M-Net in SOC, its accuracy in EOC is lower than M-Net, which suggests that it is more sensitive to a reduction in training samples. Overall, M-Net performs best. The final results of H and I in SOC, EOC-1, EOC-2, and EOC-3 are not as good as M-Net.

TABLE XVI
COMPARISON OF DIFFERENT NETWORKS

Network	SOC (%)	EOC-1 (%)	EOC-2 (%)	EOC-3 (%)
A	98.93	95.13	99.19	97.56
B	99.67	97.39	**99.70**	98.68
C	**99.71**	**97.48**	**99.67**	**98.74**
D	99.42	96.18	99.23	98.54
E	99.55	95.57	99.15	97.59
F	**99.71**	97.21	99.59	98.51
G	99.55	94.70	99.63	97.73
H	98.97	91.66	98.63	96.19
I	99.18	94.17	99.15	98.04

K. Target Recognition in Cluttered Scenes

The MSTAR clutter dataset collects many images of different scenes, such as urban area, meadow, forest, etc., of which size are about 1784×1476, as shown in Fig. 7(a). These scene images do not contain any targets. Through the method of image synthesis, we have artificially embedded the target images (testing samples in Table IV) into the scene images to generate the scene images with targets surrounded by clutter [see Fig. 7(b)].

In this target recognition experiment, we designed two M-Nets to detect and classify targets in the artificially generated cluttered scene images. The first M-Net is used to detect the image, which is a binary classification network. In the training, it uses 2747 training samples (see Table IV) as its positive samples, and random slices of clutter images as its negative samples. When testing, the trained M-Net and a sliding window are applied to detect target images in the synthetic scene, and generate a segmentation map. Next, we use a morphological method to reduce the noise in the segmentation map [see Fig. 7(c)]. According to the segmentation map, it is easy to find the location of the targets in the test images and the target image segments can be obtained. Finally, the target image segments are classified by the second M-Net which has been trained on the SOC dataset. The final result is shown in Fig. 7(d). All targets are correctly classified.

IV. Conclusion

Convolution neural networks have demonstrated significant advantages for image processing and feature extraction. In order to alleviate the overfitting problems, encountered in conventional convolution neural networks, this paper proposes a new network named M-Net. M-Net is designed to extract more information from less training data, and to improve generalization performance when only a small amount of training data is available. As can be seen from the 2-D PCA projection map, M-Net extracts feature vectors that are better separated and more discriminatory than those extracted by a conventional CNN. We have tested M-Net on the SOC, EOC-1, EOC-2, EOC-3 experiments of the MSTAR dataset and compared its performance against several other classifiers including SVM, SAEED, and A-CovNets. M-Net demonstrates better performance than the other three algorithms in all four experiments and achieves an SOC accuracy of 99.71% in a ten-target classification problem. M-Net also demonstrates strong robustness to change in quantity and parameters of training data. M-Net maintains a high and stable accuracy when key parameters, such as loss function threshold and capacity size of the information recorder, are varied significantly. The two-step training and transfer parameter training methods are another innovation of this paper, which can make the training process of M-Net efficient and precise. Future work will investigate ways of modifying the information recorder to output a probabilistic rather than deterministic values, yielding functionality similar to that of a softmax classifier. In (12), we can investigate the benefits of selecting the top k most similar samples and combining their predictions as an ensemble or combination of experts.

Acknowledgment

The authors would like to thank the editors and the anonymous reviewers for their insightful comments, which have greatly helped in improving the quality of this paper.

References

[1] K. El-Darymli, E. W. Gill, P. Mcguire, D. Power, and C. Moloney, "Automatic target recognition in synthetic aperture radar imagery: A state-of-the-art review," *IEEE Access*, vol. 4, pp. 6014–6058, Sep. 2016.

[2] R. Shang, Y. Yuan, L. Jiao, B. Hou, A. M. Ghalamzan Esfahani, and R. Stolkin, "A fast algorithm for SAR image segmentation based on key pixels," *IEEE J. Sel. Topics Appl. Earth Obs. Remote Sens.*, vol. 10, no. 12, pp. 5657–5673, Oct. 2017.

[3] Y. Ding, Y. Li, and W. Yu, "SAR image classification based on CRFs with integration of local label context and pairwise label compatibility," *IEEE J. Sel. Topics Appl. Earth Obs. Remote Sens.*, vol. 7, no. 1, pp. 300–306, Jan. 2014.

[4] L. M. Novak, G. J. Owirka, W. S. Brower, and A. L. Weaver, "The automatic target-recognition system in SAIP," *Lincoln Lab. J.*, vol. 10, no. 2, pp. 187–202, 1997.

[5] L. M. Novak, G. J. Owirka, and W. S. Brower, "Performance of 10- and 20-target MSE classifiers," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 36, no. 4, pp. 1279–1289, Oct. 2000.

[6] Q. Zhao and J. C. Principe, "Support vector machines for SAR automatic target recognition," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 37, no. 2, pp. 643–654, Apr. 2001.

[7] Y. Sun, Z. Liu, S. Todorovic, and J. Li, "Adaptive boosting for SAR automatic target recognition," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 43, no. 1, pp. 112–125, Jan. 2007.

[8] G. Dong, G. Kuang, N. Wang, L. Zhao, and J. Lu, "SAR target recognition via joint sparse representation of monogenic signal," *IEEE J. Sel. Topics Appl. Earth Obs. Remote Sens.*, vol. 8, no. 7, pp. 3316–3328, Jul. 2015.

[9] G. Dong, N. Wang, G. Kuang, and H. Qiu, "Sparsity and low-rank dictionary learning for sparse representation of monogenic signal," *IEEE J. Sel. Topics Appl. Earth Obs. Remote Sens.*, vol. 11, no. 1, pp. 141–153, Jan. 2018.

[10] Z. Jianxiong, S. Zhiguang, C. Xiao, and F. Qiang, "Automatic target recognition of SAR images based on global scattering center model," *IEEE Trans. Geosci. Remote Sens.*, vol. 49, no. 10, pp. 3713–3729, Oct. 2011.

[11] B. Ding, G. Wen, X. Huang, C. Ma, and X. Yang, "Target recognition in synthetic aperture radar images via matching of attributed scattering centers," *IEEE J. Sel. Topics Appl. Earth Obs. Remote Sens.*, vol. 10, no. 7, pp. 3334–3347, Jul. 2017.

[12] Z. Cui, Z. Cao, J. Yang, J. Feng, and H. Ren, "Target recognition in synthetic aperture radar images via non-negative matrix factorisation," *IET Radar, Sonar Navigat.*, vol. 9, no. 9, pp. 1376–1385, 2015.

[13] G. Dong and G. Kuang, "Target recognition in SAR images via classification on riemannian manifolds," *IEEE Geosci. Remote Sens. Lett.*, vol. 12, no. 1, pp. 199–203, Jan. 2015.

[14] J. Pei, Y. Huang, W. Huo, J. Wu, J. Yang, and H. Yang, "SAR imagery feature extraction using 2DPCA-based two-dimensional neighborhood virtual points discriminant embedding," *IEEE J. Sel. Topics Appl. Earth Obs. Remote Sens.*, vol. 9, no. 6, pp. 2206–2214, Jun. 2016.

[15] C. Nilubol, Q. H. Pham, R. M. Mersereau, M. J. T. Smith, and M. A. Clements, "Hidden Markov modeling for SAR automatic target recognition," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, May 1998, vol. 1, pp. 1061–1064.

[16] U. Srinivas, V. Monga, and R. G. Raj, "SAR automatic target recognition using discriminative graphical models," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 50, no. 1, pp. 591–606, Jan. 2014.

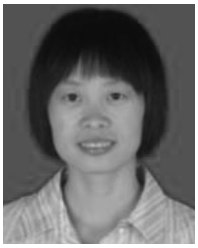
[17] J. A. O'Sullivan, M. D. DeVore, V. Kedia, and M. I. Miller, "SAR ATR performance using a conditionally Gaussian model," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 37, no. 1, pp. 91–108, Jan. 2001.

[18] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[19] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[20] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, Lake Tahoe, NV, USA, 2012, pp. 1106–1114.

[21] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Proc. Eur. Conf. Comp. Vis.*, 2014, pp. 818–833.
[22] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Represent.*, 2015.
[23] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Boston, MA, USA, 2015, pp. 1–9.
[24] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Las Vegas, NV, USA, 2016, pp. 770–778.
[25] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning hierarchical features for scene labeling," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1915–1929, Aug. 2013.
[26] T. Ross, W. Stephen, V. Vincent, M. John, and B. Michael, "Standard SAR ATR evaluation experiments using the MSTAR public release data set," *Proc. SPIE*, vol. 3370, pp. 566–573, 1998.
[27] S. Deng, L. Du, C. Li, J. Ding, and H. Liu, "SAR automatic target recognition based on Euclidean distance restricted autoencoder," *IEEE J. Sel. Topics Appl. Earth Obs. Remote Sens.*, vol. 10, no. 7, pp. 3323–3333, Jul. 2017.
[28] X. Ma, H. Wang, and J. Geng, "Spectral-spatial classification of hyperspectral image based on deep auto-encoder," *IEEE J. Sel. Topics Appl. Earth Obs. Remote Sens.*, vol. 9, no. 9, pp. 4073–4085, Sep. 2016.
[29] Y. Zhong, F. Fei, and L. Zhang, "Large patch convolutional neural networks for the scene classification of high spatial resolution imagery," *J. Appl. Remote Sens.*, vol. 10, no. 2, 2016, Art. no. 025006.
[30] S. Chen, H. Wang, F. Xu, and Y. Q. Jin, "Target classification using the deep convolutional networks for SAR images," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 8, pp. 4806–4817, Aug. 2016.
[31] L. Kaiser, O. Nachum, A. Roy, and S. Bengio, "Learning to remember rare events," in *Proc. Int. Conf. Learn. Represent.*, 2017.
[32] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.
[33] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, Cambridge, MA, USA, 2007, pp. 153–160.
[34] G. Hinton E, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *Comput. Sci.*, vol. 3, no. 4, pp. 212–223, 2012.
[35] Y. L. Boureau, F. Bach, Y. LeCun, and J. Ponce, "Learning mid-level features for recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, San Francisco, CA, USA, 2010, pp. 2559–2566.
[36] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.
[37] D. E. Rumelhart, G. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, p. 533, 1986.
[38] G. E. Hinton, S. Osindero, and Y. W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, Jul. 2006.
[39] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proc. Int. Conf. Comput. Statist.*, 2010, pp. 177–186.
[40] J. Duchi, H. Elad, and S. Yoram, "Adaptive subgradient methods for online learning and stochastic optimization," *J. Mach. Learn. Res.*, vol. 12, pp. 2121–2159. Jul. 2011.
[41] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Represent.*, 2015.
[42] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2011, pp. 315–323.
[43] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, Aug. 2013.



Ronghua Shang (M'09) received the B.S. degree in information and computation science and the Ph.D. degree in pattern recognition and intelligent systems from Xidian University, Xi'an, China, in 2003 and 2008, respectively.

She is currently a Professor with Xidian University. Her current research interests include evolutionary computation, image processing, and data mining.

Jiaming Wang received the B.E. degree in electronic information science and technology from the School of Electronic Science and Technology, Tianjin Normal University, Tianjin, China in 2016. He is currently working toward the M.S. degree in electronic circuit and system at the School of Electronic Engineering, Xidian University, Xi'an, China.

His current research interests include image processing and deep learning.

Licheng Jiao (SM'89—F'17) received the B.S. degree from Shanghai Jiaotong University, Shanghai, China, in 1982, the M.S. and Ph.D. degrees from Xi'an Jiaotong University, Xi'an, China, in 1984 and 1990, respectively, all in electronic engineering.

From 1990 to 1991, he was a Postdoctoral Fellow with the National Key Laboratory for Radar Signal Processing, Xidian University, Xi'an. Since 1992, he has been a Professor with the School of Electronic Engineering, Xidian University. He is currently the Director of the Key Lab of Intelligent Perception and Image Understanding of Ministry of Education of China, Xidian University. He is in charge of about 40 important scientific research projects, and published more than 20 monographs and 100 papers in international journals and conferences. His research interests include image processing, natural computation, machine learning, and intelligent information processing.

Dr. Jiao is a member of the IEEE Xi'an Section Execution Committee and the Chairman of awards and recognition committee, the Vice Board Chairperson of the Chinese Association of Artificial Intelligence, the Councilor of the Chinese Institute of Electronics, the Committee Member of the Chinese Committee of Neural Networks, and an expert of academic degrees committee of the state council.

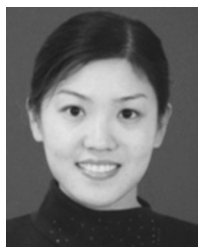
Rustam Stolkin (M'12) received the M.Eng. degree in engineering science from the University of Oxford, Oxford, U.K., in 1998, and the Ph.D. degree in computer vision from University College London, London, U.K., in 2004.

He is currently the Director of U.K.'s National Centre for Nuclear Robotics and a Professor of robotics with the University of Birmingham, Birmingham, U.K., where he is the Founder and Director of the Extreme Robotics Lab. He is also the Director of spinout company A.R.M Robotics Ltd. His research interests include computer vision and image processing, machine learning and AI, vision-guided robotic manipulation, and human–robot interaction.

Dr. Stolkin is a Fellow of the Royal Society Industry.

Biao Hou (M'07) received the B.S. and M.S. degrees in mathematics from Northwest University, Xi'an, China, in 1996 and 1999, respectively, and the Ph.D. degree in circuits and systems from Xidian University, Xi'an, in 2003.

Since 2003, he has been with the Key Laboratory of Intelligent Perception and Image Understanding of the Ministry of Education, Xidian University, where he is currently a Professor. His research interests include multiscale geometric analysis and synthetic aperture radar image processing.



Yangyang Li (M'07) received the B.S. and M.S. degrees in computer science and technology and the Ph.D. degree in pattern recognition and intelligent system from Xidian University, Xi'an, China, in 2001, 2004, and 2007, respectively.

She is currently a Professor with the Key Laboratory of Intelligent Perception and Image Understanding of Ministry of Education of China, Xidian University. Her current research interests include quantum-inspired evolutionary computation, artificial immune systems, and data mining.