SS-MLA: a semisupervised method for multi-label annotation of remotely sensed images

Tolga Üstünkök[a,*] **and Murat Karakaya**[b]
[a]Atılım University, Faculty of Engineering, Department of Software Engineering, Ankara, Turkey
[b]Atılım University, Faculty of Engineering, Department of Computer Engineering, Ankara, Turkey

Abstract. Recent technological advancements in satellite imagery have increased the production of remotely sensed images. Therefore, developing efficient methods for annotating these images has gained popularity. Most of the current state-of-the-art methods are based on supervised machine learning techniques. We propose a method called semisupervised multi-label annotizer (SS-MLA) that adapts vector-quantized temporal associative memory to annotate remotely sensed images. One of the advantages of SS-MLA over the supervised methods is that it extracts features not only from the given sample but also from similar samples that are previously seen without using an explicit attention mechanism. Thus SS-MLA enhances the learning efficiency of the training process. We conduct extensive performance comparisons with five different methods in the literature over four datasets. The comparison results indicate the success of the proposed method over the existing ones: SS-MLA generates the best results in 7 out of 11 comparisons. © *2021 Society of Photo-Optical Instrumentation Engineers (SPIE)* [DOI: 10.1117/1.JRS.15.036509]

Keywords: semisupervised; vector-quantized temporal associative memory; remote sensing; multi-label; image classification.

Paper 210289 received May 4, 2021; accepted for publication Aug. 12, 2021; published online Aug. 27, 2021.

1 Introduction

From the technological advances in satellite and airborne imagery techniques, lately, the acquisition rate of remotely sensed images has been increasing significantly. As a result, the image datasets are available for researchers to serve various application areas, such as military surveillance and tracking the growth of urban locations.[1–4] One way to utilize these datasets in applications is to categorize the data either by labeling/tagging or clustering them. Annotation (assigning labels to images) or clustering can be made manually by a group of people inspecting each image. However, this is a very tedious and cumbersome task. Therefore, various automated methods have been proposed to classify these images. Some of them annotate an image with only one label (multi-class classification),[5,6] whereas others assign multiple labels (multi-label classification) to each image.[7–11] In multi-class and multi-label classification, there have been significant challenges and important differences.[12]

Most of the multi-label classification algorithms rely on supervised methods. A supervised learning method requires both training samples and their true labels. Convolutional neural networks, recursive neural networks, and other neural networks supported with various attention mechanisms[13] are all supervised methods and have gained popularity in this research field recently.[14–17]

The performance of the supervised classification methods might be improved with the help of unsupervised methods. Although we have not found solid examples of unsupervised methods used in the field of multi-label annotation of remotely sensed images, some studies successfully perform multi-label annotation of regular images.[18,19] The results of these studies[18],[19] are promising. In addition, in one of our early studies,[20] we employed the unsupervised self-organizing

*Address all correspondence to Tolga Üstünkök, tolga.ustunkok@atilim.edu.tr maps (SOM) method to refine the existing labels of given images. In that work, we observed an improvement over the common supervised approaches. That observation encourages us to develop a semisupervised method that integrates supervised and unsupervised methods as presented below.

In this paper, we propose a semisupervised method for multi-label classification of remotely sensed images. The proposed method is based on an unsupervised neural modeling technique called vector-quantized temporal associative memory (VQTAM).[21] Essentially, VQTAM approximates non-linear discrete-time difference systems more effectively than Kohonen's SOM.[22] The effectiveness of VQTAM comes from the temporal memory that employs the previous time-step values of the system. VQTAM utilizes both previous time-step values and the current time-step values to approximate the future values of the given function. We exploit this property of VQTAM to store the image labels instead of previous time-steps. We also modify VQTAM such that it produces the occurrence probabilities of labels for a given image. To the best of our knowledge, this is the first time that VQTAM is utilized for multi-label classification.

Briefly, we modify VQTAM so that it can learn the label and image relationships as follows. Each node in the VQTAM model is influenced by the surrounding nodes during training. The neighbor nodes are also approximations of several similar images and their respected labels. Therefore, the current node can construct features that otherwise cannot be learned. Because of this automated interaction between nodes to create features, we consider the proposed method a semisupervised one. We believe that the proposed semisupervised method would be a successful alternative to the presented supervised methods thanks to the increased learning efficiency.

The reminder of this paper is organized as follows. Section 2 introduces the semisupervised method for the multi-label classification of remotely sensed images. Section 3 presents the experiment design, datasets, evaluation metrics, and results of the proposed method. Section 4 emphasizes the significance of the results. Section 5 provides a quick review of the work done and concludes this paper with suggestions for future improvements.

2 Proposed Method

In this section, we present the details of the proposed method for the multi-label annotation of remotely sensed images. In brief, the proposed method consists of two modules as depicted in Fig. 2. The first module is a deep neural network model (a pretrained ResNet50 model[23]), and the second one is a modified version of the SOM model (VQTAM[21]).

In training, the first module extracts image features from the given image using a supervised learning approach. Then we concatenate these extracted image features with the multi-hot encoded bag of words (BoW) of the corresponding image's annotations. After that, we pass the concatenated feature vectors to the VQTAM model that clusters these concatenated feature vectors applying the SOM's unsupervised learning approach. Thus, we have clusters of similar concatenated features vectors produced from training data. Since these two modules apply two different learning approaches (supervised and unsupervised), we call the proposed method the semisupervised multi-label annotizer (SS-MLA).

In inference, the SS-MLA method first generates an image feature vector for the given image using the first module. Since we aim at predicting multi-labels, we cannot create a BoW vector for the given image. Therefore, we build a BoW vector that has the same shape as the BoW vector used in training but contains all zero values. Then this BoW vector is concatenated to the image feature vector and fed into the modified VQTAM model as shown in Fig. 2. The modified VQTAM model generates and returns a new BoW vector that we can use to predict the multi-label annotation of the given image. In brief, during training, the VQTAM model learns the relationship among image features and annotations in an unsupervised way. On the other hand, during inference, the VQTAM model delivers a multi-hot encoded BoW vector as the reply for the queried image feature. We use this vector as the prediction of the labels for the given image.

Below, we first introduce the SOM algorithm as well as the VQTAM technique and explain the adaptation of VQTAM into the proposed solution. Then we define the SS-MLA method, the hyper-parameter tuning procedure, and the training process.

2.1 *Background*

Since SOM is the foundation of VQTAM, we first present the basics of SOM. Then we explain VQTAM by elaborating the differences between SOM and VQTAM.

2.1.1 *Self-organizing map*

Kohonen proposed the original SOM algorithm in 1982. The SOM algorithm compresses the non-linear high-dimensional relationships between the data samples to a much smaller dimensional vector space, i.e., two-dimensional. This property enables SOM to be used as a visualization or an abstraction tool in complex tasks such as signal processing, control, machine learning, and communication.[22]

The foundation of SOM's learning procedure is vector quantization (VQ). In essence, VQ is a signal approximation or a lossy data compression technique depending on the context. VQ generally constructs an approximation of the distribution of the data samples $x \in \mathbb{R}^n$ in terms of finite codebook vectors, $m_i \in \mathbb{R}^n$, $i = 1,2,\ldots,k$. Finding an approximation for a data point x means finding the closest codebook vector m_c to that point. In that case, the index c is found as

$$c = \arg\min_i\{\|x - m_i\|\}. \tag{1}$$

An optimal selection of the codebook vector should minimize the average expected square of the quantization error, which is defined as

$$E = \int \|x - m_c\|^2 p(x)\mathrm{d}x, \tag{2}$$

where the integral is calculated for the n-dimensional space and $p(x)$ is the probability density function of x. The detailed derivation of the VQ algorithm can be found in Ref. 22.

One downside of VQ is that it cannot map the data in an ordered manner. In contrast, the original SOM algorithm is a non-linear, ordered, smooth mapping of high-dimensional input data onto the elements of a low-dimensional array. Despite the lack of an ordered mapping feature, VQ helps to create this type of mapping between input–output pairs. SOM contains parametric real vectors $m_i = [\mu_{i1}, \mu_{i2} \ldots, \mu_{\mathrm{in}}]^{\mathrm{T}} \in \mathbb{R}^n$; each one is called a model and is associated with each of the input vectors $x = [\xi_1, \xi_2, \ldots, \xi_n]^{\mathrm{T}} \in \mathbb{R}^n$. There are multiple ways to initialize the model vectors, e.g., random. Then finding the best model m_c for a given input vector x is very similar to the VQ and is defined as

$$c = \arg\min_i\{d(x, m_i)\}, \tag{3}$$

where $d(x, m_i)$ is a general distance metric between x and m_i.

SOM first assigns each input sample to a model m_i. In other words, each model has a list of assigned input samples. The assignment is done by comparing each input vector x with all m_i and assigning the input vector to the most similar m_i considering the general distance metric according to Eq. (3). The nodes (models) that are topographically close to a certain geometric distance will activate each other to create a smoothing effect during the learning. Thus the next value of m_i with respect to time is calculated as

$$m_i(t+1) = m_i(t) + h_{ci}(t)[x(t) - m_i(t)], \tag{4}$$

where $t = 0,1,2,\ldots$ is an integer. $h_{ci}(t)$ is called the neighborhood function. The function $h_{ci}(t) \to 0$ when $t \to \infty$ is necessary for the algorithm to converge. In the literature, a smooth neighborhood function, a kind of Gaussian function, is widely utilized. In that case, the smooth neighborhood kernel is written as

$$h_{ci}(t) = \alpha(t) \cdot \exp\left(\frac{\|r_c - r\|_i^2}{2\sigma^2(t)}\right), \tag{5}$$

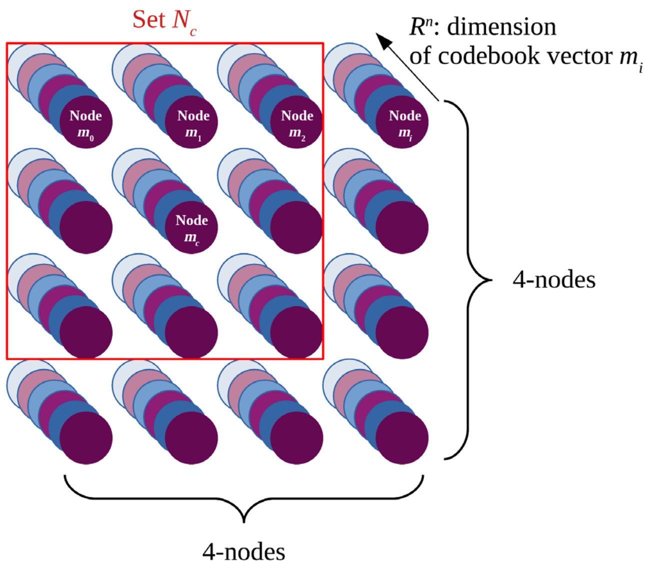

Fig. 1 A 4×4 grid for an example rectangular SOM lattice.

where $\alpha(t)$ is the learning rate; $r_c \in \mathbb{R}^2$ and $r_i \in \mathbb{R}^2$ are the location vectors of node c and i in the lattice, respectively; and $\sigma(t)$ is the width of the kernel. In other words, the radius of the set $N_c(t)$ is the neighborhood set around the node c as shown in Fig. 1.

2.1.2 *Vector-quantized temporal associative memory*

VQTAM[21] is an unsupervised neural modeling technique that uses SOM to approximate nonlinear dynamic mappings. SOM alone can only learn static input–output mappings. By adding a short-term memory, SOM can remember the past information about the system input and output vectors. Although this is a very useful feature for time series analysis, it can also be twisted to convert VQTAM to a semisupervised regressor. Instead of using the short-term memory to store the information from the previous time step, one can use it to store dependent variables as in any of the supervised learning algorithms.

To enable the approximation of dynamic mappings for SOM, the input vector $x(t)$ is divided into two parts. The first part is denoted by $x^{\text{in}}(t)$, and the second part is denoted by $x^{\text{out}}(t)$. Following this change, the weight vectors of each node i also have been divided as $m_i^{\text{in}}(t)$ and $m_i^{\text{out}}(t)$. The inputs and weight vectors are summarized as follows:

$$x(t) = \begin{pmatrix} x^{\text{in}}(t) \\ x^{\text{out}}(t) \end{pmatrix} \quad \text{and} \quad m_i(t) = \begin{pmatrix} m_i^{\text{in}}(t) \\ m_i^{\text{out}}(t) \end{pmatrix}. \tag{6}$$

VQTAM only uses the part corresponding to $x^{\text{in}}(t)$ during training to find the winning neurons. Thus updating c in Eq. (3) as $c*$ is necessary:

$$c*(t) = \arg\min_i \{d(x^{\text{in}}(t), m_i^{\text{in}}(t))\}. \tag{7}$$

In contrast with the training, it uses both $x^{\text{in}}(t)$ and $x^{\text{out}}(t)$ while updating the weights. Hence, we update Eq. (4) as follows:

$$m_i^{\text{in}}(t+1) = m_i^{\text{in}}(t) + h_{c*i}(t)[x^{\text{in}}(t) - m_i^{\text{in}}(t)], \tag{8}$$

$$m_i^{\text{out}}(t+1) = m_i^{\text{out}}(t) + h_{c*i}(t)[x^{\text{out}}(t) - m_i^{\text{out}}(t)]. \tag{9}$$

With this, we conclude the brief introduction of the fundamental algorithms. In our method, we use VQTAM to find the corresponding clusters of the query image from their image features vectors. VQTAM finds the cluster of the incoming query and returns it. After that, we isolate the labels part from the returned cluster and report it as the multi-labels of the query image. In the inference, we form our query vectors by omitting the BoW vector part as in real-world applications. In the next section, we define the multi-label annotation problem of remotely sensed images. Then we introduce the proposed SS-MLA method.

2.2 *Multi-Label Image Annotation*

In multi-label image annotation, multiple labels are assigned to the corresponding images by detecting the important features and relating them to the respected labels. Assume that a dataset of images are denoted as $\mathbf{X} = \{\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_n\}$, where n is the number of images and a dictionary of labels is denoted as $\mathcal{L} = \{l_1, l_2, \dots, l_k\}$, where k is the number of unique labels $|\mathcal{L}| = k$. If we call the subset $L \in 2^{\mathcal{L}}$ as the set of relevant labels of $\mathbf{X}_i$, then $\mathcal{L} \setminus L$ indicates the set of irrelevant labels. L can be denoted with a binary vector $Y = \{y_1, y_2, \dots, y_k\}$, where $y_i = 1$ if $l_i \in L$. We denote all such possible combinations of Y with $\mathcal{Y} = \{0,1\}^k$. The aim of the SS-MLA method is to learn a relation $\mathfrak{F}: \mathbf{X} \to \mathcal{Y}$ from the dataset $\mathcal{D} = \{(\mathbf{X}_i, Y_i) | 1 \le i \le n, \mathbf{X}_i \in \mathbf{X}, Y_i \in \mathcal{Y}\}$. Hence, we can call $Z_i = \mathfrak{F}(\mathbf{X_i}) \in \{0,1\}^k$ the predicted label set.

To create BoW vectors (shown as the pink box in Fig. 3), we opt out using the multi-hot encoding method. In this method, we first create a dictionary from all labels. Then according to the size of the dictionary, we decide the size of the BoW vector. The index of the labels in the dictionary determines the location (index) of the labels in the BoW vector. For the labels of an image, we mark their existence in the corresponding BoW vector by setting the label index values as 1. All other indices of the corresponding BoW vector have 0 values. Thus using the BoW vector, we can encode the labels in the image as a vector consisting of 0 and 1 values.

2.3 *Semisupervised Multi-Label Annotizer*

As demonstrated in Fig. 2, the SS-MLA method consists of three structurally different modules: ResNet50, BoW, and VQTAM. Training the ResNet50 and VQTAM modules and using them in inference are very different from each other. Therefore, we design a three-phase approach for integrating these modules. First, we explain the training process and leave the inference process for later.

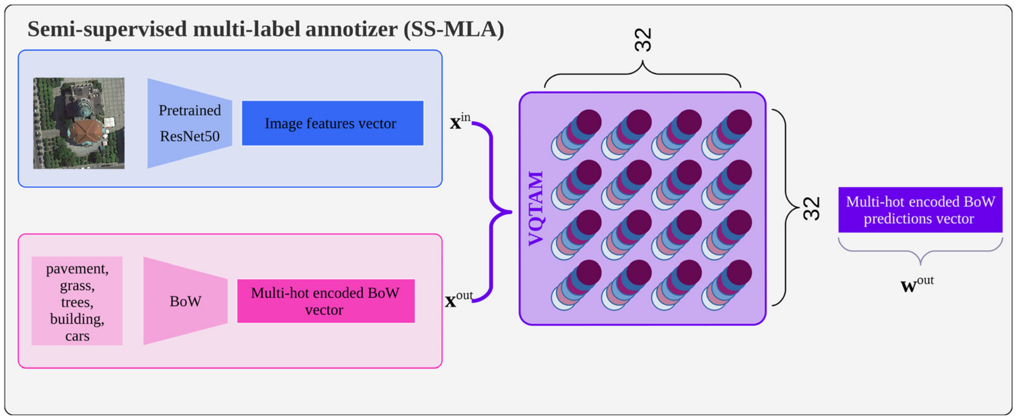
Semi-supervised multi-label annotizer (SS-MLA)
Pretrained ResNet50
Image features vector
$\mathbf{x}^{in}$
pavement, grass, trees, building, cars
BoW
Multi-hot encoded BoW vector
$\mathbf{x}^{out}$
VQTAM
32
32
Multi-hot encoded BoW predictions vector
$\mathbf{w}^{out}$

Fig. 2 A high-level view of the SS-MLA method.

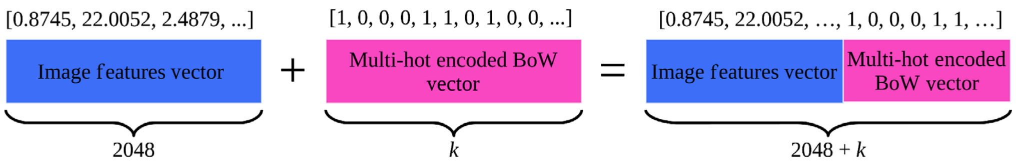

Fig. 3 The vector concatenation operation that occurs in the SS-MLA method.

2.3.1 *Training process*

Execution of training is done in three phases. In the first phase, we apply fine-tuning to the ResNet50 model, pretrained with the ImageNet dataset.[24] Therefore, we drop the top dense layers of the ResNet50 model. Then we attach three layers instead of the dropped top dense layers. The first layer is a global average pooling layer.[25] This layer reduces the output vector size of the ResNet50 from $7 \times 7 \times 2048$ to 2048 scalars only. After that, a dropout layer[26] follows the previous global average pooling layer with a network weight dropping probability of 0.5. Finally, a dense sigmoid layer is placed to perform the multi-label classification. The number of units in this layer is determined by the number of unique labels in the dataset. We train this network with the previously defined training images for 10 epochs with a learning rate of 0.0005.

In the second phase, we transform each training image with the ResNet50 network to construct the $x^{\text{in}} \in \mathbb{R}^{n\times 2048}$. Next, using the training data, we generate multi-hot encoded BoW vectors for corresponding multi-labels of images and concatenate them $x^{\text{out}} \in \{0,1\}^{n\times k}$ to the x^{in} to create the $x \in \mathbb{R}^{n\times(2048+k)}$ in the training time.

The above two phases apply supervised learning methods to generate image and label features. However, in the last phase, we design an unsupervised learning approach by adapting the VQTAM model as a clustering method. We feed the concatenated feature vectors to the modified VQTAM model with x for e number of epochs. We choose the learning rate as α, the neighborhood function $h_{ci}(t)$ as the Gaussian kernel presented in Eq. (5), the topology as a 32×32 rectangular grid, the width of N_c as 1.0, and the activation distance as the L2 distance metric. Thus the modified VQTAM model builds an SOM model considering the image and label feature vectors.

2.3.2 *Impact of the grid size*

The grid size of an SOM (VQTAM) model is one of its most important hyper-parameters since it indicates how many clusters can be formed inside the map. To decide which grid size best suits our purpose, we try different grid dimensions. We start from an 8×8 grid size and increment it to 16×16, 32×32, and 64×64 in each iteration. Each time, we measure the $F1$-score as well as the training time. We report the results measured by utilizing the UCM multi-label dataset in Fig. 4. As can be seen from this figure, the 32×32 grid size gives us the best $F1$-score in a reasonable amount of time. Therefore, we build all of our models with the 32×32 grid size.

To achieve an efficient training process, we set up a grid search to determine the optimum hyper-parameter values of VQTAM for the rest of the hyper-parameters. These parameters are the number of epochs (e), α and σ, which is the width of N_c. For the number of epochs, we search the range from 1000 to 25,000 epochs. For α, we search the range from 0.1 to 1.0. Finally, for σ, we search the range from 1.0 to 5.0. The optimum hyper-parameter values are found to be different for each dataset. To validate the training, we randomly select a node from the lattice after training with the UCM multi-label dataset and check for the clustered images. The clustered images are all similar images and they are shown in Fig. 5.

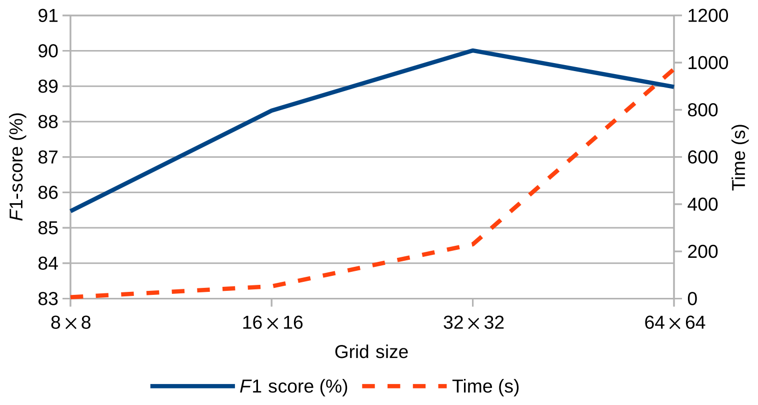

Fig. 4 The effect of the grid size measured on the UCM multi-label dataset.

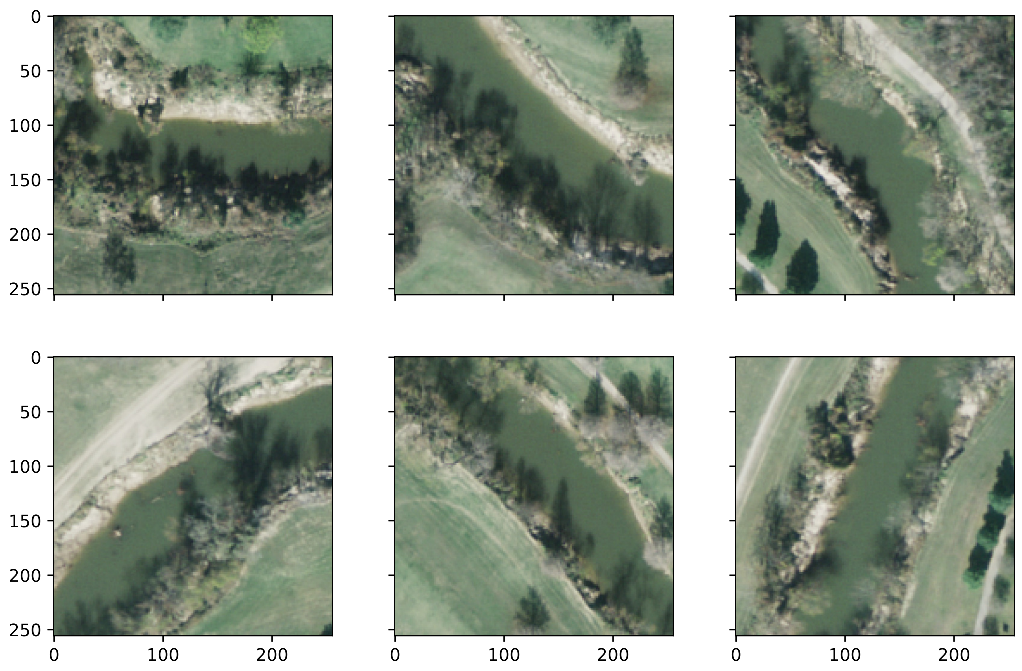

Fig. 5 Clustered images on the VQTAM node with the coordinates (13, 18) after training. All have the same labels of grass, sand, trees, and water.

2.3.3 *Inference process*

In the inference process, we take the image $\mathbf{X}_i \in \{k|k \in \mathbb{Z}^{224\times224\times3}, 0 \leq k \leq 255\}$ and transform it into a vector $x_i^{\text{in}} \in \mathbb{R}^{2048}$ using the pretrained, fine-tuned ResNet50 architecture given in Fig. 2. Then we create a multi-hot encoded BoW vector $x_i^{\text{out}} \in \{0,1\}^k$ consisting of only zeros and append it to the previously constructed vector to prepare the query input vector $x_i \in \mathbb{R}^{2048+k}$ for VQTAM. As seen at the bottom of Fig. 2, VQTAM consumes the x_i, finds the winning node inside the lattice by looking for the closest m_j to the x_i according to Eq. (7), and returns its m_j^{out}, where j is limited with the number of nodes in the lattice.

m_j^{out} is the probability value of the target labels. To convert it to a multi-hot encoded vector, we adapt a brute force search to find the optimum threshold value $\tau \in (0,1)$. While in the training process (we discuss the details later), we search the space with steps of 0.01 and pick the one that gives us the highest $F1$-score. After getting the optimum τ, we threshold the whole vector to get the multi-hot encoded labels of the given image $\mathbf{X}_i$.

In the following section, we present the evaluation results of the proposed method on four Łdifferent datasets along with the comparisons made with five methods.

3 Experiment and Result

In this section, we first present a summary of the datasets that we use in this work. Then we introduce the evaluation metrics to compare the methods. Finally, we compare the classification performances of five other methods, given in Sec. 1, with the proposed SS-MLA method. We present the reported results of the related work considering four different datasets, yielding 11 comparisons in total. To achieve a fair comparison, the example-based metrics ($F1$-score, $F2$-score, precision, and recall) are used. While interpreting the performance of the methods, we only consider the $F1$-score, which is the most popular metric throughout the related literature. The highest $F1$-score is printed with a bold font in all of the tables.

3.1 *Datasets*

There are various datasets to be used in the multi-label classification of remotely sensed images. We gather a collection of accessible datasets that are employed by the related work of this study.

We analyze them by defining metrics such as distinct label set (DLS), proportion of distinct label set (PDLS), label cardinality (LC), and label density (LD).[27] The variability of these statistical properties may uncover the distinctive characteristics of models applied on different datasets. The details of the metrics are given below.

DLS is the number of all unique subsets of $2^{\mathcal{L}}$, which is assigned to at least one sample. It is defined as follows:

$$\mathrm{DLS}(\mathcal{D}) = |\{Y_i | \exists\ \mathbf{X}_i : (\mathbf{X}_i, Y_i) \in \mathcal{D}\}|. \tag{10}$$

PDLS is the normalized version of DL by the number of samples in the dataset. It is defined as follows:

$$\mathrm{PDLS}(\mathcal{D}) = \frac{\mathrm{DLS}(\mathcal{D})}{|\mathcal{D}|}. \tag{11}$$

LC is the average number of labels per sample. It is defined as follows:

$$\mathrm{LC}(\mathcal{D}) = \frac{1}{n}\sum_{i=1}^{n} |Y|_i. \tag{12}$$

LD is the normalized version of LC with the number of labels. It is defined as follows:

$$\mathrm{LD}(\mathcal{D}) = \frac{\mathrm{LC}(\mathcal{D})}{k}. \tag{13}$$

All the defined statistic metrics are calculated for each dataset and presented in Table 1. The details of each dataset are given below.

3.1.1 *AID multi-label dataset*

The creators of the AID multi-label dataset[15] describe the dataset as the multi-class version of the widely used AID dataset. The original AID dataset was constructed from 10,000 high-resolution images taken from Google Earth from various locations worldwide. It contains 30 scene labels. In the AID multi-label dataset, there are 3000 images with a size of 600×600 each. The number of scene labels is reduced to 17 by manually labeling each image. The LC is 5.1523. Hence, it yields an LD of 0.3031, which is almost the same as the DFC15 multi-label and Ankara datasets.

3.1.2 *UCM multi-label dataset*

Chaudhuri et al.[28] created a multi-label version of the UCMERCED archive[29] called the UCM multi-label dataset. The UCM multi-label dataset contains 2100 images, each of size 256×256. Each image is manually relabeled from a dictionary of 17 labels. Statistical properties of this dataset are somewhat similar to the AID multi-label dataset, although DLS and LC properties are a little low when compared.

Table 1 Remotely sensed multi-label image classification datasets.

#	Dataset name	Image size	#Images	#Distinct labels	DLS	PDLS	LC	LD
1	AID multi-label	600×600	3000	17	297	0.0990	5.1523	0.3031
2	UCM multi-label	256×256	2100	17	202	0.0962	3.3348	0.1962
3	DFC15 multi-label	600×600	3342	8	65	0.0194	2.7953	0.3494
4	Ankara dataset	63×63	216	29	127	0.5880	9.1204	0.3145

3.1.3 *DFC15 multi-label dataset*

The DFC15 multi-label dataset has 3342 images, each of size 600×600. The dataset was built from another dataset called DFC15.[30] The DFC15 multi-label dataset has the least DLS and number of labels among the datasets, which also affects the PDLS, LC, and LD values. These properties may simplify the classification tasks on this dataset, especially the DLS statistic.

3.1.4 *Ankara dataset*

The Ankara dataset is the smallest dataset that we run in our experiments. It contains 216 hyperspectral images, each of size 63×63. There are 29 labels and 127 DLS in this dataset, which increase the PDLS and LC values of this dataset quite a bit. A high PDLS value means a low number of samples for each image, and each image has ∼9 labels. These statistics show that the Ankara dataset is very different from the other datasets.

3.2 *Evaluation Metrics*

In multi-label classification, the predictions can be fully correct, partially correct, or fully incorrect. While evaluating the results, this partialness should be handled differently from the binary classification and multi-class classification. Otherwise, the alternative might be assuming the partially correct ones as incorrect. This way, also known as exact match ratio, obviously cannot differentiate between partially correct ones and incorrect ones.[27]

One way to handle this is to calculate the average difference between actual labels and predicted labels for each sample and average the results for all samples. This averaging technique is known as example-based averaging. The metrics used for binary and multi-class classification (precision, recall, $F1$-score, and $F2$-score) can also be used in multi-label classification by calculating the average as example-based. The example-based versions of these metrics are explained below.

The precision (P) is the ratio of correctly predicted labels to the predicted labels, averaged over all samples. It is defined as follows:

$$P = \frac{1}{n}\sum_{i=1}^{n}\frac{|Y_i \cap Z_i|}{|Z_i|}, \tag{14}$$

where n is the number of samples in the test set.

The recall (R) is the ratio of correctly predicted labels to the actual labels, averaged over all samples. It is defined as follows:

$$R = \frac{1}{n}\sum_{i=1}^{n}\frac{|Y_i \cap Z_i|}{|Y_i|}, \tag{15}$$

where n is the number of samples in the test set.

The harmonic mean of precision and recall is called $F1$-score. If recall is more important than precision depending on the problem, a generalized version of $F1$-score can be used. This version of $F1$-score (F_1) is called F-beta (F_β) score. In the following equations, one can see the definitions of $F1$-score and F-beta score, respectively,

$$F_1 = \frac{1}{n}\sum_{i=1}^{n}\frac{2|Y_i \cap Z_i|}{|Y_i| + |Z_i|}, \tag{16}$$

$$F_\beta = \frac{1}{n}\sum_{i=1}^{n}\left[(1+\beta^2)\frac{|Y_i \cap Z_i|}{\beta^2|Y_i| + |Z_i|}\right]. \tag{17}$$

3.3 *Experiment Design*

We design an experiment setup to measure the performance of SS-MLA. In this setup, each dataset has been resampled 10 times randomly. Every resampling is repeated twice, except for the Ankara dataset, for two possible training, validation, and test splits (either 70% to 10% to 20% or 80% to 10% to 10%, respectively). For a fair comparison, data split ratios are decided according to the compared study. Training, validation, and test splits are created using the same random seed within each resampling. After splitting the training and test datasets, we train newly created instances of SS-MLA on the training data for each training set split as explained in Sec. 2.3.1. Each model is evaluated on the test set of the respected dataset.

We report the mean of the example-based precision, recall, $F1$-score, and $F2$-score along with the 95% confidence interval for every 10 experiments for different splits of the same dataset yielding 7 different results; 2 for AID multi-label, 2 for UCM multi-label, 2 for DFC15 multi-label, and 1 for Ankara. The results are presented in the following section.

3.4 *Result and Discussion*

Table 2 shows the results obtained from the models using Ankara dataset. We compare the observed performance of SS-MLA with the reported results of the method developed by Zhu et al.[14] One can observe that SS-MLA ($F1$-score of 82.39%) surpasses the technique developed by Zhu et al. ($F1$-score of 76.60%) by 5.79%.

Table 3 shows the evaluation metric values of the second and third comparisons. In this comparison set, we used the DFC15 multi-label dataset. In this table, we create two comparisons; one with GRN-SNDL-BCE[17] and the other with CA-ResNet-BiLSTM.[8] We conduct each comparison by considering the train–test split ratio of the compared study. In the first split group, GRN-SNDL-BCE[17] ($F1$-score of 95.80%) succeeded above our method ($F1$-score of 94.92%) by only 0.88%. In the second split group, SS-MLA achieved better results than CA-ResNet-BiLSTM[8] by 11.55%. In the end, we succeed in one out of two comparisons in this dataset.

Table 4 presents the evaluation metric values of the fourth, fifth, and sixth comparisons. We utilize the AID multi-label dataset to be able to measure the quality of the produced results. We compare SS-MLA with GRN-SNDL-BCE,[17] Zhu et al.[14] and AL-RN-ResNet.[15] As seen at the table, SS-MLA is surpassed by GRN-SNDL-BCE with only 1.25% of $F1$-score in the first split group. For the remaining comparisons, we outperform both of them with $F1$-score differences of 2.39% and 1.16% for Zhu et al.[14] and AL-RN-ResNet,[15] respectively. Conclusively, the SS-MLA framework produces better $F1$-scores than two out of three methods in comparison.

Table 2 Comparison of SS-MLA with the literature on the Ankara dataset.

Method name	Results of published methods				Results of SS-MLA				Testing method
	P	R	F_1	F_2	P	R	F_1	F_2	
Zhu et al.[14]	81.22	82.12	76.60	—	85.30 ± 1.81	82.92 ± 1.75	82.39 ± 1.09	82.30 ± 1.31	Random split[a]

[a]80% train, 10% test, and 10% validation.

Table 3 Comparison of SS-MLA with the literature on the DFC15 multi-label dataset.

Method name	Results of published methods				Results of SS-MLA				Testing method
	P	R	F_1	F_2	P	R	F_1	F_2	
GRN-SNDL-BCE[17]	96.53	95.95	**95.80**	95.78	95.70 ± 0.39	95.31 ± 0.33	94.92 ± 0.25	95.01 ± 0.27	Random split[a]
CA-ResNet-BiLSTM[8]	91.93	79.12	83.65	80.61	96.34 ± 0.29	95.18 ± 0.56	95.20 ± 0.29	95.06 ± 0.45	Random split[b]

[a]70% train, 20% test, and 10% validation.
[b]80% train, 10% test, and 10% validation.

Table 4 Comparison of SS-MLA with the literature on the AID multi-label dataset.

Method name	Results of published methods				Results of SS-MLA				Testing method
	P	R	F_1	F_2	P	R	F_1	F_2	
GRN-SNDL-BCE[17]	92.79	91.08	**90.95**	90.82	91.36 ± 0.48	90.48 ± 0.63	89.70 ± 0.16	89.87 ± 0.42	Random split[a]
Zhu et al.[14]	89.72	88.41	87.49	—	91.21 ± 0.61	91.02 ± 0.70	89.88 ± 0.38	90.26 ± 0.53	Random split[b]
AL-RN-ResNet[15]	91.00	88.95	88.72	88.54					

[a]70% train, 20% test, and 10% validation.
[b]80% train, 10% test, and 10% validation.

Table 5 Comparison of SS-MLA with the literature on the UCM multi-label dataset.

Method name	Results of published methods				Results of SS-MLA				Testing method
	P	R	F_1	F_2	P	R	F_1	F_2	
GRN-SNDL-BCE[17]	91.98	92.83	**91.31**	91.92	89.23 ± 0.78	92.15 ± 0.87	89.55 ± 0.56	90.78 ± 0.68	Random split[a]
Zhu et al.[14]	91.75	91.65	90.62	—	89.37 ± 0.86	92.23 ± 0.87	89.60 ± 0.51	90.83 ± 0.64	Random split[b]
AL-RN-ResNet[15]	88.81	87.07	86.76	86.67					
CA-ResNet-BiLSTM[8]	77.94	89.02	81.47	85.27					
LR-ResNet[7]	87.10	85.80	85.30	—					

[a]70% train, 20% test, and 10% validation.
[b]80% train, 10% test, and 10% validation.

Table 5 shows the evaluation results of the related work for the UCM multi-label dataset. We compare SS-MLA with GRN-SNDL-BCE,[17] Zhu et al.,[14] AL-RN-ResNet,[15] CA-ResNet-BiLSTM,[8] and LR-ResNet.[7] SS-MLA is barely outperformed by two of the previous studies by 1.76% and 1.02%. On the other hand, SS-MLA outperforms the rest by 2.84%, 8.13%, and 4.3% $F1$-scores, respectively. When we query all of the test images on VQTAM and randomly select one of the nodes (15, 0), we observe that the clustered images are very similar to each other as shown in Fig. 6. In addition to that, all of their labels are the same with the correct labels. Of course, this is not the case for all of the nodes. However, this example is a positive indicator that VQTAM successfully learns and clusters unseen images as well. As a summary, SS-MLA achieves better results in three comparisons out of five for this dataset.

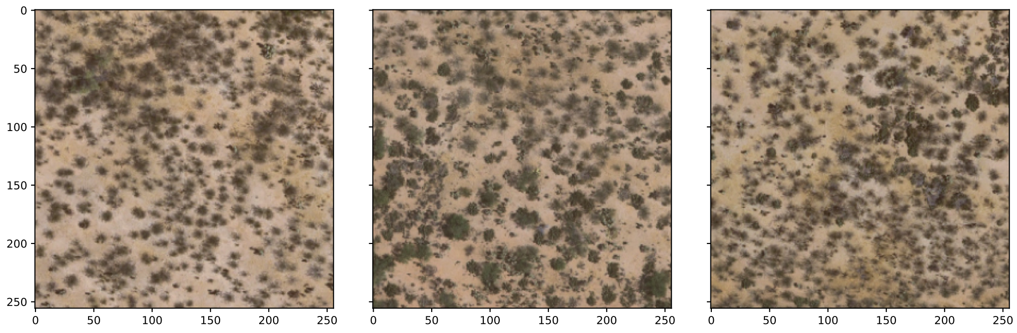

Fig. 6 All of the unseen test images are clustered into the node (15, 0). They all have the labels bare-soil and chaparral. This example is taken from the experiments run on the UCM multi-label dataset.

Table 6 Summary of the results.

Dataset	#Methods	Rank of SS-MLA
Ankara dataset	2	1
DFC15 multi-label	3	2
AID multi-label	4	2
UCM multi-label	6	3

Considering the experiments overall, SS-MLA successfully generates better results than seven out of eleven comparisons. In the next section, we elaborate on the results. While refining the ideas, we explain the relations between different approaches and the datasets used in the experiments.

4 Discussions

SS-MLA can attain competitive performances compared to the considered supervised methods in the literature. In Table 6, we summarize the overall results. In addition to the dataset names, we include the PDLS and LC columns to visualize the effect of those values on the other methods and SS-MLA. The #methods column is the total number of methods (including SS-MLA) that performed tests on the respected dataset. The final column is the rank of SS-MLA among the other methods. Here it can be inferred that SS-MLA is more successful than the majority of the supervised methods that we compare.

The results of the experiments support that an influential factor in the results is the characteristics of a dataset. In our experiments, we use four different datasets. Two of them—AID and UCM multi-label—have similar characteristics. Conversely, the Ankara dataset and DFC15 multi-label have different characteristics both from each other and from the AID and UCM multi-label datasets. The most varying characteristics of the datasets are given in Table 6.

The first observation is when the dataset has a low number of images—i.e., the Ankara dataset with a high LC and a high DLS; SS-MLA reliably achieves acceptable results while the other methods cannot. This finding may be the result of the clustering operation performed in the training phase of VQTAM. Unfortunately, we cannot compare SS-MLA with other methods since those studies did not test their methods on the Ankara dataset except for the method proposed by Zhu et al.[14] On the other hand, the method of Zhu et al.[14] was tested on two other datasets and obtained better scores. Therefore, it might support our claim. At this point, further examination is needed in future studies to dispel doubts.

Another observation is that the DFC15 multi-label dataset has a very low number of DLS with 65 sets. When this number is proportional to the number of images, one can get a PDLS of 0.0194, which is lower than all of the other datasets. SS-MLA is the second-best method in this dataset by a difference of 0.88%. At the same time, it has an 11.55% higher $F1$-score than the third-best method.

These results indicate that a semisupervised method such as SS-MLA can be applied successfully to the multi-label classification of remotely sensed images instead of a supervised method. According to the results, small datasets have a less negative effect on the classification accuracy of semisupervised methods than the supervised and sophisticated ones.

5 Conclusion

It is obvious that the production speed of remotely sensed images has increased and continues to increase. These images must be labeled and classified to be used properly in digital environments. This job is very tedious for a person to do. For this reason, many multi-label classifiers have been developed. Many of these classifiers work with supervised learning. The accuracy of the methods that perform supervised learning can be improved by methods that perform unsupervised learning. For this reason, we have proposed a method called SS-MLA that performs both supervised and unsupervised learning in this study. Since this method uses both learning systems, we put it in the category of methods that do semisupervised learning. As a semisupervised method, SS-MLA can achieve more successful results than seven out of eleven comparisons in total, which shows that SS-MLA can be as successful as other compared supervised methods.

As a future study, we plan to utilize SS-MLA to work on regular images other than remotely sensed images. In addition, we will also consider adding an attention layer to discover the relation between labels explicitly and feed this information to the VQTAM alongside the implicit label relation information that is discovered by the VQTAM itself.

Acknowledgments

The authors have no relevant financial interests in the manuscript and no other potential conflicts of interest to disclose the manuscript.

References

1. J. R. Townshend, C. O. Justice, and V. Kalb, "Characterization and classification of South American land cover types using satellite data," *Int. J. Remote Sens.* **8**(8), 1189–1207 (1987).
2. C. J. Tucker, J. R. Townshend, and T. E. Goff, "African land-cover classification using satellite data," *Science* **227**, 369–375 (1985).
3. W. G. Bastiaanssen, M. U. D. Ahmad, and Y. Chemin, "Satellite surveillance of evaporative depletion across the Indus Basin," *Water Resour. Res.* **38**, 9-1–9-9 (2002).
4. R. D. Hudson and J. W. Hudson, "The military applications of remote sensing by infrared," *Proc. IEEE* **63**(1), 104–128 (1975).
5. T. Chen, Y. Zhao, and Y. Guo, "Sparsity-regularized feature selection for multi-class remote sensing image classification," *Neural Comput. Appl.* **32**, 6513–6521 (2020).
6. G. Cheng et al., "Multi-class geospatial object detection and geographic image classification based on collection of part detectors," *ISPRS J. Photogramm. Remote Sens.* **98**, 119–132 (2014).
7. Y. Hua, L. Mou, and X. X. Zhu, "Label relation inference for multi-label aerial image classification," in *Int. Geosci. and Remote Sens. Symp. (IGARSS)*, Institute of Electrical and Electronics Engineers Inc., pp. 5244–5247 (2019).
8. Y. Hua, L. Mou, and X. X. Zhu, "Multi-label aerial image classification using a bidirectional class-wise attention network," in *Joint Urban Remote Sens. Event, JURSE 2019*, Institute of Electrical and Electronics Engineers Inc. (2019).
9. A. Alshehri et al., "Deep attention neural network for multi-label classification in unmanned aerial vehicle imagery," *IEEE Access* **7**, 119873–119880 (2019).
10. Y. Hua, L. Mou, and X. X. Zhu, "Recurrently exploring class-wise attention in a hybrid convolutional and bidirectional LSTM network for multi-label aerial image classification," *ISPRS J. Photogramm. Remote Sens.* **149**, 188–199 (2019).
11. R. Stivaktakis, G. Tsagkatakis, and P. Tsakalides, "Deep learning for multilabel land cover scene categorization using data augmentation," *IEEE Geosci. Remote Sens. Lett.* **16**(7), 1031–1035 (2019).
12. Y. Li et al., "Deep learning for remote sensing image classification: a survey," *WIREs Data Mining Knowl Discov.* **8**(6), e1264 (2018).
13. S. Chaudhari et al., "An attentive survey of attention models," (2019).
14. P. Zhu et al., "Deep learning for multilabel remote sensing image annotation with dual-level semantic concepts," *IEEE Tran. Geosci. Remote Sens.* **58**, 4047–4060 (2020).
15. Y. Hua, L. Mou, and X. X. Zhu, "Relation network for multilabel aerial image classification," *IEEE Trans. Geosci. Remote Sens.* **58**, 4558–4572 (2020).
16. S. Koda et al., "Spatial and structured SVM for multilabel image classification," *IEEE Trans. Geosci. Remote Sens.* **56**, 5948–5960 (2018).

17. J. Kang et al., "Graph relation network: modeling relations between scenes for multilabel remote-sensing image classification and retrieval," *IEEE Trans. Geosci. Remote Sens.* **59**, 1–15 (2020).
18. M. Guillaumin, J. Verbeek, and C. Schmid, "Multimodal semi-supervised learning for image classification," in *Proc. IEEE Comput. Soc. Conf. Comput. Vision and Pattern Recognit.*, pp. 902–909 (2010).
19. Z. Wei, H. Wang, and R. Zhao, "Semi-supervised multi-label image classification based on nearest neighbor editing," *Neurocomputing* **119**, 462–468 (2013).
20. T. Ustunkok, O. C. Acar, and M. Karakaya, "Image tag refinement with self-organizing maps," in *1st Int. Inf. and Software Eng. Conf.: Innov. Technol. for Digital Transform., IISEC 2019, Proc.*, Institute of Electrical and Electronics Engineers Inc. (2019).
21. G. De A Barreto and A. F. Araújo, "Temporal associative memory and function approximation with the self-organizing map," in *Neural Networks for Signal Process. Proc. IEEE Workshop, 2002-January*, Institute of Electrical and Electronics Engineers Inc., pp. 109–118 (2002).
22. T. Kohonen, *Self-Organizing Maps*, Springer Series in Information Sciences, Vol. 30, Springer Berlin Heidelberg, Berlin, Heidelberg (1997).
23. K. He et al., "Deep residual learning for image recognition," in *Proc. IEEE Comput. Soc. Conf. Comput. Vision and Pattern Recognit., 2016-December*, IEEE Computer Society, pp. 770–778 (2016).
24. J. Deng et al., "ImageNet: A Large-Scale Hierarchical Image Database," in *EEE Conf. Comput. Vision and Pattern Recognit.*, pp. 248–255, Institute of Electrical and Electronics Engineers (IEEE) (2010).
25. I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, Cambridge, Massachusetts (2016).
26. N. Srivastava et al., "Dropout: a simple way to prevent neural networks from overfitting," *J. Mach. Learning Res.* **15**(1), 1929–1958 (2014).
27. M. Sorower, *A Literature Survey on Algorithms for Multi-Label Learning*, pp. 1–25, Oregon State University, Corvallis (2010).
28. B. Chaudhuri et al., "Multilabel remote sensing image retrieval using a semisupervised graph-theoretic method," *IEEE Trans. Geosci. Remote Sens.* **56**, 1144–1158 (2018).
29. L. Yang, S. Hanneke, and J. Carbonell, "A theory of transfer learning with applications to active learning," *Mach. Learn.* **90**(2), 161–189 (2013).
30. G. Moser et al., "2015 IEEE GRSS data fusion contest: Extremely high resolution LidAR and optical data [technical committees]," *IEEE Geosci. Remote Sens. Mag.* **3**(1), 40–41 (2015).

Tolga Üstünkök received his bachelor's degree in computer engineering in 2017 and his MSc degree in software engineering in 2021 from Atılım University, Ankara, Turkey. He is currently a research assistant in the Department of Software Engineering at Atılım University. His research interests are machine learning, deep learning, natural computing, and embedded systems design for flying machines.

Murat Karakaya received his BSEE degree in 1991 from the Turkish Military Academy, Ankara, Turkey and his MS degree and PhD in computer engineering from Bilkent University, Ankara, Turkey, in 2000 and 2008, respectively. From 2000 to 2005, he worked as an instructor and software engineer at the Turkish Military Academy, Ankara, Turkey. From 2008 to 2012, he worked as an instructor and software engineer at the Turkish Military School of Electronics, Communications, and Information Systems and the Turkish Military Academy, Ankara, Turkey. He joined the faculty of Atılım University in 2012 and is currently an associate professor in the Department of Computer Engineering. His research interests are machine learning, deep learning, natural computing, sensor networks, and peer-to-peer networks.