A Self-Training Hierarchical Prototype-based Ensemble Framework for Remote Sensing Scene Classification

Xiaowei Gu[1, *], Ce Zhang[2, 3, *], Qiang Shen[4], Jungong Han[4], Plamen P. Angelov[5] and Peter M. Atkinson[2]

[1] School of Computing, University of Kent, Canterbury, CT2 7NZ, UK
[2] Lancaster Environment Centre, Lancaster University, Lancaster, LA1 4YQ, UK
[3] UK Centre for Ecology & Hydrology, Lancaster, LA1 4AP, UK
[4] Department of Computer Science, Aberystwyth University, Aberystwyth, SY23 3DB, UK
[5] School of Computing and Communications, Lancaster University, Lancaster, LA1 4WA, UK
E-mails: x.gu@kent.ac.uk;{qqs, juh22}@aber.ac.uk; {c.zhang9, p.angelov, pma}@lancaster.ac.uk

Abstract: Remote sensing scene classification plays a critical role in a wide range of real-world applications. Technically, however, scene classification is an extremely challenging task due to the huge complexity in remotely sensed scenes, and the difficulty in acquiring labelled data for model training such as supervised deep learning. To tackle these issues, a novel semi-supervised ensemble framework is proposed here using the self-training hierarchical prototype-based classifier as the base learner for chunk-by-chunk prediction. The framework has the ability to build a powerful ensemble model from both labelled and unlabelled images with minimum supervision. Different feature descriptors are employed in the proposed ensemble framework to offer multiple independent views of images. Thus, the diversity of base learners is guaranteed for ensemble classification. To further increase the overall accuracy, a novel cross-checking strategy was introduced to enable the base learners to exchange pseudo-labelling information during the self-training process, and maximize the correctness of pseudo-labels assigned to unlabelled images. Extensive numerical experiments on popular benchmark remote sensing scenes demonstrated the effectiveness of the proposed ensemble framework, especially where the number of labelled images available is limited. For example, the classification accuracy achieved on the OPTIMAL-31, PatternNet and RSI-CB256 datasets was up to 99.91%, 98. 67% and 99.07% with only 40% of the image sets used as labelled training images, surpassing or at least on par with mainstream benchmark approaches trained with double the number of labelled images.

Keywords: self-training; pseudo-labelling; prototypes; remote sensing; scene classification.

1. Introduction

Remote sensing scene classification aims to allocate remote sensing images into different land-use categories automatically, and is an important research goal due to its utility in a wide range of Earth observation applications [1]–[3]. However, remote sensing scenes often present huge complexity and heterogeneity, with high intra-class similarity (and low inter-class variation) and complex geometrical structures, often compounded by limited availability of labelled images. Remote sensing scene classification is, thus, considered a challenging task for the machine learning community and has received wide attention [4], [5].

Currently, deep neural networks (DNNs) are mainstream methods for remote sensing scene classification, outperforming handcrafted feature engineering methods on a variety of benchmark datasets [5]–[7]. However, DNNs are "black box" models with a significant number of hyper-parameters. Their training processes are computationally expensive and commonly restricted to offline learning. Furthermore, a DNN model requires a large number of labelled images to be trained properly. DNNs unfortunately are not able to utilize the huge number of unlabelled images available for training purposes. In real-world application scenarios, high-quality labelled images are hard to obtain due to the expensive costs of manual annotation [8], [9]. Annotating remote sensing images manually usually requires a high-level of knowledge and expertise, and can only be done by well-trained researchers. This makes labelled remote sensing images a scarce resource. Although it is possible to increase the number of labelled images through data augmentation techniques, the generalisation of DNN models is scarified, and the models become fragile to uncertainties and lack robustness. On the other hand, semi-supervised learning methods [10] can build a powerful predictive model from both labelled and unlabelled images simultaneously with less human input. Therefore, semi-supervised methods have been increasingly explored in the remote sensing domain [11], [12]. However, the vast majority of existing works using semi-supervised learning techniques

* Corresponding author

focused on land-cover classification of remotely sensed images [13]–[17]. The potential of semi-supervised learning for land-use classification has not been sufficiently explored.

The self-training hierarchical prototype-based (STHP) classifier was introduced recently as a semi-supervised learning method exploiting "pseudo-labelling" [18]. Initially primed with a small number of labelled data, STHP is able to self-learn a multi-layered prototype-based structure from unlabelled data. These prototypes represent local models of data distributions identified at multiple granularity levels, and are aggregated into pyramidal hierarchies. STHP model offers high transparency and interpretability compared with mainstream classifiers thanks to its prototype-based nature, while maintaining high classification accuracy with minimum human supervision. As STHP can capture the most distinctive characteristics between different classes at multiple specificity levels with its multi-layered structure, it is highly suitable for solving classification problems involving complex data structures, such as remote sensing scene classification. In addition, unlike traditional semi-supervised methods, STHP is able to learn from unlabelled data on a "chunk-by-chunk" basis and self-evolve its system structure continuously to adapt to unfamiliar data patterns.

In this paper, using STHP+ (a modified version of the original STHP [18]) as the base learner, a novel self-training hierarchical prototype-based ensemble framework (STHPEF) is proposed for remote sensing scene classification. STHPEF boosts the diversity of its base classifiers using different types of feature descriptors from multiple views of remote sensing images. As the base classifiers self-expand their knowledge bases from unlabelled images by pseudo-labelling, the pseudo-labelling mechanism of the original STHP model is modified to increase precision by using valuable information mined from unlabelled data at multiple specificity levels, leading to the enhanced version STHP+. In addition, a cross-checking mechanism is introduced to STHPEF, allowing the base classifiers to self-train using unlabelled images, with information exchange to maximize pseudo-labelling accuracy. Integrating the new pseudo-labelling and cross-checking mechanism has great potential to increase the overall accuracy of the proposed ensemble framework.

Major contributions of this paper are:

1) A novel pseudo-labelling mechanism is proposed by considering the multi-granular information mined from data in decision-making.

2) A novel semi-supervised ensemble framework with a cross-checking mechanism is used to learn from unlabelled images autonomously with accuracy.

The remainder of this paper is organized as follows. Section 2 presents a review of related works. Details of the proposed STHP+ classifier and STHPEF system are described in Sections 3 and 4. Numerical experiments are provided in Section 5, and a conclusion is drawn in Section 6.

2. Related Works

2.1. Remote Sensing Scene Classification

The main aim of remote sensing scene classification is to assign automatically distinct land-use class labels to remotely sensed images based on their semantic content. Existing scene classification methods generally fall into three categories [2]: 1) methods based on low-level visual features, 2) methods based on mid-level visual features, and 3) methods based on high-level visual features. Low-level methods are based mostly on hand-crafted features such as Gist [19], scale-invariant feature transform (SIFT) [20], local binary pattern (LBP) [21] and colour histogram (CH) [22]. These handcrafted features require domain expertise for their production, and have limited descriptive capabilities. Mid-level methods attempt to construct holistic representations by encoding low-level visual features [1]. Typical mid-level methods include bag of visual words (BoVW) [23] and unsupervised feature learning methods, such as sparse encoding [24], [25] and the autoencoder [26]. However, BoVW-based methods lack flexibility and are hard to adapt to different problems, while unsupervised feature-based methods fail to capture class-specific information [27]. Hence, the performance of mid-level methods may not be satisfactory.

In contrast to the previous two types, high-level methods utilize high-level semantic features learned by DNNs (mostly, deep convolutional neural networks, DCNNs) for remote sensing scene classification [8], [28]. DCNNs have achieved impressive results on a variety of complex real-world problems such as image classification, object detection and natural language processing [5], [29]. Thanks to their powerful feature representation, DCNN-based methods have achieved high accuracies in remote sensing scene classification. For example, Tong et al. [1]

employed the lightweight DenseNet-121 as the backbone to extract spatial features at multiple scales and introduced a channel attention mechanism to strengthen the weights of important feature channels. Zhang et al. [30] proposed a multi-scale dense network for hyperspectral remote sensing image classification. Lu et al. [31] introduced an end-to-end feature aggregation convolutional neural network (CNN) that aggregates different convolutional features for scene classification. Liu et al. [32] constructed Siamese CNNs to learn discriminative feature representations for accurate remotely sensed scene classification. Zhang et al. [33] introduced a lightweight CNN with high classification accuracy. Recent review papers [4], [5] summarised some of the latest development in DNNs for remote sensing scene classification.

Despite the above advantages, DCNNs are data- and computational resource-hungry, and their training may require a huge number of labelled training images. Transfer learning techniques [34] can prevent DNNs from overfitting when trained with small-size datasets, but fine-tuned networks are still unstable without sufficient labelled images. An effective approach is to employ off-the-shelf DNNs pre-trained using large-scale natural images (i.e., ImageNet [35]) as feature descriptors to extract high-level representations for remotely sensed scene classification [2], [9]. Researchers [36], [37] have reported high accuracy by training mainstream classifiers such as support vector machines (SVM) and random forests (RF) using high-level features from pre-trained DNNs (e.g. AlexNet, VGGNet, GoogLeNet). However, these pre-trained DNNs often fail to capture the most distinctive characteristics of different land-use categories, due to the intricate and highly complicated geometric structures and spatial patterns involved.

To date, the majority of remote sensing scene classification involves training in a fully supervised fashion to learn a predictive model [38]–[40]. However, as introduced above, labelled images are scarce and expensive to obtain, while unlabelled images are plentiful. Supervised methods are unable to utilize unlabelled images during the training process. In contrast, semi-supervised methods can build strong predictive models using both labelled and unlabelled images [10] with minimum human labelling effort. Despite this, most existing semi-supervised methods were developed for land cover classification using hyperspectral/multispectral satellite sensor images [13]–[16], [41], [42] and polarimetric synthetic aperture radar images [17], [43], [44], with the main purpose of assigning a pixel (or a group of pixel) of a remotely sensed image to a particular land-cover class. Very few semi-supervised methods have been proposed for remote sensing scene classification [9], which deserves further exploration.

2.2. Semi-supervised Learning

Semi-supervised learning is a hybrid machine learning technique combining features of both supervised and unsupervised learning [18], [45]. Semi-supervised learning methods can be categorized broadly into two major categories: 1) inductive methods and 2) transductive methods [10]. Inductive methods aim to construct a predictive model from both labelled and unlabelled data, which can be used to predict class labels of newly available data samples after the training phase. Typical inductive methods include semi-supervised support vector machine (S3VM) [46], safe S3VM (S4VM) [47], self-training [48], co-training [49] and SemiBoost [50]. Transductive methods aim to predict class labels of unlabelled samples that are given during the training phase. Since transductive methods do not build predictive models, their predictive power is limited to the unlabelled samples seen during training. Mainstream transductive method include the transductive support vector machine (TSVM) [46], Laplacian SVM [51], local and global consistency (LGC) [52], and anchor graph regularization (AGR) [53].

Self-training [48] and co-training [49], [54] are the most popular inductive semi-supervised learning methods. Typically, the training processes of the two methods are composed of two alternating steps; training and pseudo-labelling [10]. A standard self-training method first trains a predictive model with the labelled data and then uses the trained model to classify the unlabelled samples. Unlabelled samples with the highest classification confidence are chosen to augment the labelled training set with their predicted labels, which are the so-called “pseudo-labels”. The predictive model is then retrained with the augmented labelled training set and the same process is repeated.

A standard co-training method [49], [54] firstly trains two or more classifiers with labelled data. Then, the classifiers are used for predicting the class labels of unlabelled samples from their respective views. The most confident predictions of each classifier are used to augment the training set of the other(s). The classifiers are then retrained with the augmented training sets, and the same process is repeated, which is similar to the standard procedure of self-training methods. However, the key to the success of co-training methods is that the trained classifiers show a certain degree of disagreement between each other. To achieve this, co-training methods either split the feature space into multiple views (namely, sub-feature sets), split data into multiple views (namely, sub-

datasets), or promote the diversity of trained classifiers [10]. In general, self-training is easy to implement compared with other semi-supervised learning methods and, importantly, it does not impose any assumptions on the data generation models with parameters, providing a simple but effective approach to utilize valuable hidden information from unlabelled data [18].

3. STHP+ Classifier

In this section, technical details of the STHP+ classifier are presented. Compared with the original STHP classifier introduced in [18], the self-training mechanism of STHP+ is modified to increase the pseudo-labelling precision by considering multi-granular data distribution information.

Let $\mathbf{X}_K = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_K\}$ be a particular dataset in a N dimensional real data space, $\mathbf{R}^N$; $\boldsymbol{x}_k = [x_{k,1}, x_{k,2}, \dots, x_{k,N}]^T$ is the sample observed at the kth time instance; K is the total number of data samples. $\mathbf{X}_K$ is composed of data samples of C different classes, and only the first L samples are labelled ($L \ll K$) with their corresponding class labels denoted as $\mathbf{Y}_L = \{y_1, y_2, \dots, y_L\}$; $y_i \in \{1,2, \dots, C\}$ for $\forall y_i \in \{y\}_L$. Hence, $\mathbf{X}_K$ can be divided into a labelled subset, denoted as $\mathbf{X}_L = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_L\}$ and an unlabelled set, denoted as $\mathbf{X}_U = \{\boldsymbol{x}_{L+1}, \boldsymbol{x}_{L+2}, \dots, \boldsymbol{x}_K\}$. The labelled set can be further divided into C subsets, namely, $\mathbf{X}_{i,L_i} = \{\boldsymbol{x}_{i,1}, \boldsymbol{x}_{i,2}, \dots, \boldsymbol{x}_{i,L_i}\}$ according to the class labels ($i = 1,2, \dots, L_i$) and there is $L = \sum_{i=1}^{C} L_i$. For clarity, a list of key notations is given by Table 1.

Table 1. Key notations and definitions

Notations	***Definitions***
$\mathbf{X}_K$	A dataset
$\boldsymbol{x}_k$	The data sample observed at the kth time instance
$\mathbf{I}_k$	The image observed at the kth time instance
y_k	The true label of $\boldsymbol{x}_k$ or $\mathbf{I}_k$
$\hat{y}_k$	The estimated label of $\boldsymbol{x}_k$ or $\mathbf{I}_k$
C	Number of classes
$\mathbf{X}_L$	The set of labelled data
$\mathbf{Y}_L$	Labels of $\mathbf{X}_L$
$\mathbf{X}_U$	The set of unlabelled data
L	Cardinality of $\mathbf{X}_L$
$\mathbf{X}_{i,L_i}$	The set of data belonging to the ith class
$\acute{\mathbf{X}}_j$	The jth unlabelled data chunk
Q	The cardinality of $\acute{\mathbf{X}}_j$
$\boldsymbol{x}_{j,k}$	The kth unlabelled sample in $\acute{\mathbf{X}}_j$
L^i	Cardinality of $\mathbf{X}_{L^i}^i$
$\widehat{\mathbf{X}}_j$	The set of pseudo-labelled samples from $\acute{\mathbf{X}}_j$
$\widehat{\mathbf{Y}}_j$	Pseudo-labels of $\widehat{\mathbf{X}}_j$
N	Dimensionality of the data space
$\mathbf{R}^N$	The data space
λ_i	The confidence score produced by the ith hierarchy
$\mathbf{P}_i^h$	The set of prototypes at the hth layer of the ith hierarchy
M_i^h	Cardinality of $\mathbf{P}_i^h$
$\boldsymbol{p}_{i,j}^h$	The jth prototype in $\mathbf{P}_i^h$
$\boldsymbol{\ell}_{i,j}^h$	The set of immediate subordinates of $\boldsymbol{p}_{i,j}^h$
$S_{i,j}^h$	The number of data samples associated with $\boldsymbol{p}_{i,j}^h$
DR^i	The ith feature descriptor
E	Number of feature descriptors
$\boldsymbol{x}_k^i$	The feature vector of $\mathbf{I}_k$ extracted by the ith feature descriptor
$\mathbb{I}_j$	The jth image chunk
$\acute{\mathbf{X}}_j^i$	The collection of feature vectors extracted from $\mathbb{I}_j$ by the ith feature descriptor
$\widehat{\mathbf{X}}_j^i$	The set of pseudo-labelled feature vectors from $\acute{\mathbf{X}}_j^i$
$\widehat{\mathbf{Y}}_j^i$	Pseudo-labels of $\widehat{\mathbf{X}}_j^i$

$\widetilde{\mathbf{X}}_j^i$	The cross-checked subset of $\widehat{\mathbf{X}}_j^i$
$\widetilde{\mathbf{Y}}_j^i$	Pseudo-labels of $\widetilde{\mathbf{X}}_j^i$

3.1. General Architecture

The general architecture of STHP+ (Fig. 1 [18]) is composed of C prototype-based hierarchies that are self-organized from data based on their ensemble properties and mutual distances. Each hierarchy corresponds to a particular class and is composed of prototypes arranged by multiple layers, representing local peaks of a multimodal data distribution at different levels of granularity/specificity. Prototypes at the higher layers of the hierarchies represent global patterns of the data distribution and have strong generalization capabilities. Prototypes at the lower layers disclose local patterns and provide fine details. Without loss of generality, all the pyramidal hierarchies within the system have H layers.

The learning process of STHP+ is composed of two stages [18]. First, the STHP+ classifier learns from a small number of labelled training data to prime its knowledge base, system structure and meta-parameters in a supervised manner. In the second stage, it self-learns from unlabelled data continuously to improve its knowledge base and update its system structure and meta-parameters by exploiting pseudo-labelling.

During the pseudo-labelling/decision-making process, the STHP+ classifier assigns class labels to unlabelled samples in terms of confidence scores produced by the pyramidal hierarchies, $\hat{y} = label(\boldsymbol{x})$:

$$\hat{y} \leftarrow i^*; \quad i^* = \underset{i=1,2,\ldots,C}{\operatorname{argmax}}\left(\lambda_i(\boldsymbol{x})\right) \tag{1}$$

These confidence scores are based on the similarities between these samples and the prototypes identified from data of different classes, following the "multi-nearest prototypes" principle, given by Eq. (2) ($i = 1,2,\ldots,C$).

$$\lambda_i(\boldsymbol{x}) = e^{-\sum_{l=1}^{W_0}\left\|\boldsymbol{x}-\boldsymbol{p}_{i,n_l^*}\right\|^2} \tag{2}$$

where W_0 is a user-controlled parameter, as a positive integer indicating the number of nearest prototypes per class used for decision-making and pseudo-labelling; $\boldsymbol{p}_{i,n_l^*}$ is the lth nearest prototype at the ith hierarchy, $\boldsymbol{p}_{i,n_l^*} \in \mathbf{P}_i^1 \cup \mathbf{P}_i^2 \cup \ldots \cup \mathbf{P}_i^H$. The recommended value of W_0 is 4 to extract sufficient granular information from multiple levels. Compared with the original STHP classifier, which calculates the confidence score based on the nearest bottom-layer prototype only [18], multiple nearest prototypes from all layers are considered when computing the confidence scores to increase pseudo-labelling accuracy and the robustness of STHP+ classifier to noise.

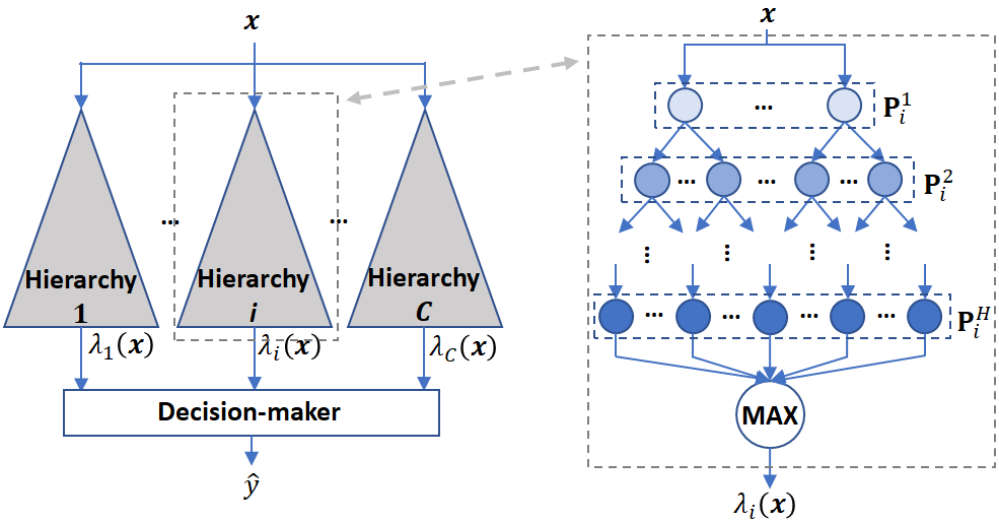


Fig. 1. General architecture of STHP+ classifier

3.2. Self-Training Process

The supervised and self-training procedures of the STHP+ classifier are described as follows. By default, each training/testing sample is normalized by its Euclidean norm, namely,

$$\boldsymbol{x} \leftarrow \frac{\boldsymbol{x}}{\|\boldsymbol{x}\|} \tag{3}$$

where $\|\boldsymbol{x}\| = \sqrt{\boldsymbol{x}^T\boldsymbol{x}}$.

A. The supervised learning procedure

The STHP+ classifier follows the same supervised learning procedure as the original version [18].

The classifier starts a new learning cycle when the kth labelled sample $(\boldsymbol{x}_k, y_k)$ is observed. Assuming that $y_k = i$, the ith prototype-based hierarchy is initialized if $\boldsymbol{x}_k$ is the very first data sample of this class. In this case, $\boldsymbol{x}_k$ is used as the first prototype at each layer of this new hierarchy ($h = 1,2,\dots,H$):

$$M_i^h \leftarrow 1; \quad \boldsymbol{p}_{i,M_i^h}^h \leftarrow \boldsymbol{x}_k; \quad S_{i,M_i^h}^h \leftarrow 1; \quad \mathbf{P}_i^h \leftarrow \left\{\boldsymbol{p}_{i,M_i^h}^h\right\} \tag{4}$$

where $\boldsymbol{p}_{i,j}^h$ denotes the jth prototype at the hth layer of the ith hierarchy; $\mathbf{P}_i^h$ is the collection of all prototypes at this layer; M_i^h is the cardinality of $\mathbf{P}_i^h$, and; $S_{i,j}^h$ is the support/number of samples associated with $\boldsymbol{p}_{i,j}^h$.

Then, the ith hierarchy is established by linking prototypes of successive layers ($h = 1,2,\dots,H-1$):

$$\boldsymbol{\ell}_{i,M_i^h}^h \leftarrow \left\{\boldsymbol{p}_{i,M_i^{h+1}}^{h+1}\right\} \tag{5}$$

where $\boldsymbol{\ell}_{i,j}^h$ is the collection of immediate subordinate prototypes at the h+1th layer associated with $\boldsymbol{p}_{i,j}^h$. After the links between the H layers are built, the initialization of the ith hierarchy is completed. Otherwise (namely, $\boldsymbol{x}_k$ is not the very first data sample of the ith class), $\boldsymbol{x}_k$ is used for updating the hierarchy in a top-down manner starting from the top layer, $h = 1$. The nearest prototype, $\boldsymbol{p}_{i,n^*}^h$ at the hth layer is identified by Eq. (6) [18]:

$$\boldsymbol{p}_{i,n^*}^h = \begin{cases} \underset{\boldsymbol{p}\in\mathbf{P}_i^h}{\operatorname{argmin}}(\|\boldsymbol{p}-\boldsymbol{x}_k\|), & h = 1 \\ \underset{\boldsymbol{p}\in\boldsymbol{\ell}_{i,n^*}^{h-1}}{\operatorname{argmin}}(\|\boldsymbol{p}-\boldsymbol{x}_k\|), & h > 1 \end{cases} \tag{6}$$

where $\boldsymbol{\ell}_{i,n^*}^{h-1}$ is the collection of immediate subordinates of the nearest prototype at the h-1th layer.

Condition 1 is examined to determine whether $\boldsymbol{x}_k$ is sufficiently distinctive to become a new prototype at the hth layer [18]:

$$Condition\ 1: \quad \begin{array}{c} If\ \left(\left\|\boldsymbol{p}_{i,n^*}^h - \boldsymbol{x}_k\right\|^2 > r^h\right) \\ Then\ (\boldsymbol{x}_k\ becomes\ a\ new\ prototype\ at\ the\ hth\ layer) \end{array} \tag{7}$$

where r^h is the radius of the area of influence around the prototypes at the hth layer of hierarchies derived using Eq. (8):

$$r^h = 2\left(1 - cos\left(\frac{\theta_0}{2^{h-1}}\right)\right) \tag{8}$$

Here θ_0 specifies the maximum angle between any two similar samples at the first level of granularity. The recommended value of θ_0 is $\frac{\pi}{3}$.

The influence of θ_0 on the system performance will be investigated by numerical experiments later. However, it is worth noting that the radii, r^h $(h = 1,2,\dots,H)$ can be determined based on user preference without prior knowledge of the problem. The only constraint one needs to consider when setting manually their values is that $r^1 > r^2 > \cdots > r^H$.

If Condition 1 is satisfied, $\boldsymbol{x}_k$ becomes a new prototype at the hth layer and also the successive lower layers of the ith hierarchy, adding a new branch to this hierarchy ($j = h, h+1, \dots, H$).

$$M_i^j \leftarrow M_i^j + 1; \quad \boldsymbol{p}_{i,M_i^j}^j \leftarrow \boldsymbol{x}_k; \quad S_{i,M_i^j}^j \leftarrow 1; \quad \mathbf{P}_i^j \leftarrow \mathbf{P}_i^j \cup \left\{\boldsymbol{p}_{i,M_i^j}^j\right\} \tag{9}$$

If $\boldsymbol{p}_{i,M_i^h}^h$ is not an apex prototype, the nearest prototype at the h-1th layer, $\boldsymbol{p}_{i,n^*}^{h-1}$ is recognized as the starting node of this new branch, and $\boldsymbol{\ell}_{i,n^*}^{h-1}$ is updated as:

$$\boldsymbol{\ell}_{i,n^*}^{h-1} \leftarrow \boldsymbol{\ell}_{i,n^*}^{h-1} \cup \left\{\boldsymbol{p}_{i,M_i^h}^h\right\} \tag{10}$$

and the links between $\boldsymbol{p}_{i,M_i^h}^h$, $\boldsymbol{p}_{i,M_i^{h+1}}^{h+1}$, …, $\boldsymbol{p}_{i,M_i^H}^H$ are built using Eq. (6). However, if $\boldsymbol{x}_k$ fails to satisfy Condition 1, it is used for updating the meta-parameters of $\boldsymbol{p}_{i,n*}^h$ as follows:

$$\boldsymbol{p}_{i,n^*}^h \leftarrow \frac{S_{i,n^*}^j \boldsymbol{p}_{i,n^*}^h + \boldsymbol{x}_k}{S_{i,n^*}^j + 1}; \quad \boldsymbol{p}_{i,n^*}^h \leftarrow \frac{\boldsymbol{p}_{i,n^*}^h}{\left\|\boldsymbol{p}_{i,n^*}^h\right\|}; \quad S_{i,n^*}^j \leftarrow S_{i,n^*}^j + 1 \tag{11}$$

Then, $\boldsymbol{x}_k$ is passed to the next layer ($h \leftarrow h + 1$), and the same process starting from Eq. (6) is repeated until Condition 1 is satisfied at a certain layer or $\boldsymbol{x}_k$ reaches the bottom layer.

The STHP+ classifier starts a new learning cycle if the next labelled sample is available ($k \leftarrow k + 1$), otherwise, it starts the self-training process to learn from unlabelled samples.

The supervised learning process of the STHP+ classifier is summarized by the following pseudo-code [18].

Algorithm 1. STHP+ supervised learning procedure

inputs: $\mathbf{X}_L$; $\mathbf{Y}_L$; H; θ_o;
algorithm begins

for $k = 1$ **to** L **do:**
  a. read $\boldsymbol{x}_k$ and y_k ($y_k = i$);
  b. **if** ($\mathbf{P}_i^1 = \emptyset$) **then:**
    i. **for** $h = 1$ **to** H **do:**
      1. initialize M_i^h, $\boldsymbol{p}_{i,M_i^h}^h$, $S_{i,M_i^h}^h$, $\mathbf{P}_i^h$ by (4);
    ii. **end for**
    iii. **for** $h = 1$ **to** $H - 1$ **do:**
      1. initialize $\boldsymbol{\ell}_{i,M_i^h}^h$ by (5);
    iv. **end for**
  c. else:
    i. **for** $h = 1$ **to** H **do:**
      1. identify $\boldsymbol{p}_{i,n^*}^h$ by (6);
      2. **if** (Condition 1 is satisfied) **then:**
        * **for** $j = h$ **to** H **do:**
          - update M_i^h, $\boldsymbol{p}_{i,M_i^h}^h$, $S_{i,M_i^h}^h$, $\mathbf{P}_i^h$ by (9);
        * **end for**
        * **if** ($h \neq 1$) **do:**
          - update $\boldsymbol{\ell}_{i,n^*}^{h-1}$ by (10);
        * **end if**
        * **for** $j = h$ **to** $H - 1$ **do:**
          - initialize $\boldsymbol{\ell}_{i,M_i^h}^h$ by (5)
        * **end for**
        * break **for** loop
      3. **else:**
        * update $\boldsymbol{p}_{i,n^*}^h$ and S_{i,n^*}^j by (11);
      4. **end if**
    ii. **end for**
  d. **end if**
end for

algorithm ends
output: *trained hierarchies*

B. The self-training procedure

The self-training process selects a subset of samples from $\acute{\mathbf{X}}_j = \{\boldsymbol{x}_{j,1}, \boldsymbol{x}_{j,2}, \ldots, \boldsymbol{x}_{j,Q}\}$ as the *j*th data chunk, together with corresponding pseudo-labels $\hat{\mathbf{Y}}_j$ ($j = 1,2, \ldots, J$; J is the number of chunks and $\acute{\mathbf{X}}_1 \cup \acute{\mathbf{X}}_2 \cup \ldots \cup \acute{\mathbf{X}}_J = \mathbf{X}_U$) that STHP+ is highly confident about, and uses them to update the classifier. Thus, each self-training cycle is composed of the following two steps. Note that it is assumed in this paper that the data chunks are of the same

size, Q for simplicity. However, the sizes of different data chunks do not necessarily have to be the same in real-world applications.

Step 1. Pseudo-labelling data

The set of pseudo-labelled samples and their corresponding pseudo-labels are initialized as: $\widehat{\mathbf{X}}_j \leftarrow \emptyset$ and $\widehat{\mathbf{Y}}_j \leftarrow \emptyset$. Condition 2 is checked to identify the samples that locate in the areas of influence of prototypes of a single class only, starting from the top layer ($h = 1$) until the bottom layer ($h = H$) [18].

$$Condition\ 2\text{: } \begin{array}{c} \text{If } \left(\left\|\boldsymbol{p}_{i,n^*}^h - \boldsymbol{x}_{j,k}\right\|^2 \le r^h\right) \text{ and } \left(\left\|\boldsymbol{p}_{l,n^*}^h - \boldsymbol{x}_{j,k}\right\|^2 > r^h, \forall l \ne i\right) \\ \text{Then } \left(\boldsymbol{x}_{j,k} \text{ is assigned with pseudo label } \hat{y}_{j,k} = i\right) \end{array} \tag{12}$$

where $\boldsymbol{p}_{i,n^*}^h$ is the nearest prototype to $\boldsymbol{x}_{j,k}$ at the *h*th layer of the *i*th hierarchy identified by Eq. (13) ($i = 1,2,\dots,C$):

$$\boldsymbol{p}_{i,n^*}^h = \underset{\boldsymbol{p} \in \mathbf{P}_i^h}{\operatorname{argmin}}\left(\left\|\boldsymbol{p} - \boldsymbol{x}_{j,k}\right\|\right) \tag{13}$$

Instead of using Eq. (6) to identify the nearest prototype as the original STHP classifier [18], which wastes multi-granular information and may lead to errors due to the layer-by-layer searching process, Eq. (13) can identify the prototypes closest to $\boldsymbol{x}_{j,k}$ in data space, thus, increasing the pseudo-labelling accuracy by Condition 2.

If Condition 2 is satisfied at the *h*th layer, $\boldsymbol{x}_{j,k}$ is spatially close to prototypes of the *i*th class ($i = 1,2,\dots,C$) and is distant to prototypes of other classes. In such cases, it is highly likely that $\boldsymbol{x}_{j,k}$ belongs to the *i*th class, and $\boldsymbol{x}_{j,k}$ is pseudo-labelled as $\hat{y}_{j,k} = i$. However, if $\boldsymbol{x}_{j,k}$ fails to satisfy Condition 2 at the bottom layer, every hierarchy within the system will first produce a confidence score calculated by Eq. (2). Then, Condition 3 is examined to check whether a pseudo-label can be assigned to $\boldsymbol{x}_{j,k}$ [18]:

$$Condition\ 3\text{: } \begin{array}{c} \text{If } \left(\lambda_i\left(\boldsymbol{x}_{j,k}\right) > \gamma_0 \cdot \lambda_l\left(\boldsymbol{x}_{j,k}\right), \forall i \ne l\right) \\ \text{Then } \left(\boldsymbol{x}_{j,k} \text{ is assigned with pseudo label } \hat{y}_{j,k} = i\right) \end{array} \tag{14}$$

where $\gamma_0 > 1$ is a user-controlled parameter.

If $\boldsymbol{x}_{j,k}$ satisfies Conditions 2 or 3, $\boldsymbol{x}_{j,k}$ is put into $\widehat{\mathbf{X}}_j$ after being removed from $\acute{\mathbf{X}}_j$, and $\hat{y}_{j,k}$ is put into $\widehat{\mathbf{Y}}_j$ such that $\boldsymbol{x}_{j,k}$ and $\hat{y}_{j,k}$ are used to update the classifier in *Step 2*:

$$\widehat{\mathbf{X}}_j \leftarrow \widehat{\mathbf{X}}_j \cup \left\{\boldsymbol{x}_{j,k}\right\}; \quad \widehat{\mathbf{Y}}_j \leftarrow \widehat{\mathbf{Y}}_j \cup \left\{\hat{y}_{j,k} = i\right\}; \quad \acute{\mathbf{X}}_j \leftarrow \acute{\mathbf{X}}_j \backslash \left\{\boldsymbol{x}_{j,k}\right\} \tag{15}$$

However, if both conditions are not met, $\boldsymbol{x}_{j,k}$ will put back to $\acute{\mathbf{X}}_j$. After all samples in the current chunk have been examined by Conditions 2 and/or 3, the second step of the current cycle begins.

Step 2. Self-updating with pseudo-labelled data

With the obtained $\widehat{\mathbf{X}}_j$ and $\widehat{\mathbf{Y}}_j$, the STHP+ classifier self-evolves to expand its knowledge base by following *the same supervised learning* procedure as described before. It then goes back to *Step 1* and looks for more eligible samples from the remaining $\acute{\mathbf{X}}_j$ to update the system. The current self-training cycle ends when there is no sample in $\acute{\mathbf{X}}_j$ that can meet Conditions 2 or 3 anymore. The STHP+ classifier enters a new self-training cycle if a new data chunk $\acute{\mathbf{X}}_{j+1}$ is available.

The self-training process of the STHP+ classifier is summarized by the following pseudo-code.

Algorithm 2. STHP+ self-training procedure

inputs: $\acute{\mathbf{X}}_1, \acute{\mathbf{X}}_2, \dots, \acute{\mathbf{X}}_J$; γ_0; ***algorithm begins***
for $j = 1$ **to** J **do:** *a.* read $\acute{\mathbf{X}}_j$; *b.* $Q \leftarrow \left\lvert\acute{\mathbf{X}}_j\right\rvert$; *c.* **while** ($\acute{\mathbf{X}}_j \ne \emptyset$): // *Step 1. Pseudo-labelling* //

i. $\widehat{\mathbf{X}}_j \leftarrow \emptyset$; $\widehat{\mathbf{Y}}_j \leftarrow \emptyset$;
ii. **for** $k = 1$ **to** Q **do:**
1. **for** $h = 1$ **to** H **do:**
* **if** (Condition 2 is satisfied) **then:**
- $\widehat{\mathbf{X}}_j \leftarrow \widehat{\mathbf{X}}_j \cup \{\boldsymbol{x}_{j,k}\}$; $\widehat{\mathbf{Y}}_j \leftarrow \widehat{\mathbf{Y}}_j \cup \{\hat{y}_{j,k} = i\}$; $\acute{\mathbf{X}}_j \leftarrow \acute{\mathbf{X}}_j \backslash \{\boldsymbol{x}_{j,k}\}$;
- break **for** loop;
* **end if**
* **if** (Condition 3 is satisfied) **then:**
- $\widehat{\mathbf{X}}_j \leftarrow \widehat{\mathbf{X}}_j \cup \{\boldsymbol{x}_{j,k}\}$; $\widehat{\mathbf{Y}}_j \leftarrow \widehat{\mathbf{Y}}_j \cup \{\hat{y}_{j,k} = i\}$; $\acute{\mathbf{X}}_j \leftarrow \acute{\mathbf{X}}_j \backslash \{\boldsymbol{x}_{j,k}\}$;
- break **for** loop;
* **end if**
2. **end for**
iii. **end for**
// *Step 2. Self-updating* //
d. **if** ($\widehat{\mathbf{X}}_j = \emptyset$) **then:**
1. break **while** loop;
e. **else:**
1. update the hierarchies with $\widehat{\mathbf{X}}_j$ and $\widehat{\mathbf{Y}}_j$ using *Algorithm 1*;
f. **end if**
g. $Q \leftarrow |\acute{\mathbf{X}}_j|$;
h. **end while**
end for

algorithm ends
output: *self-trained hierarchies*

4. Proposed Ensemble System

The main aim of STHPEF is to learn an ensemble classification model from both labelled and unlabelled training images to perform more accurate classification. As for an ensemble component STHP+, the learning process, as shown in Fig. 2, is composed of two stages, namely, *i)* supervised learning and *ii)* self-training. In the first stage, STHPEF is primed with labelled training images. In the second stage, STHPEF further self-improves with unlabelled training images on a chunk-by-chunk basis by exploiting the "pseudo-labelling" technique. Unlike traditional ensemble systems where each ensemble component is trained separately, the STHPEF system involves a novel cross-checking mechanism such that each ensemble learner exchanges information with other ensemble components during the self-training process to maximize pseudo-labelling precision.

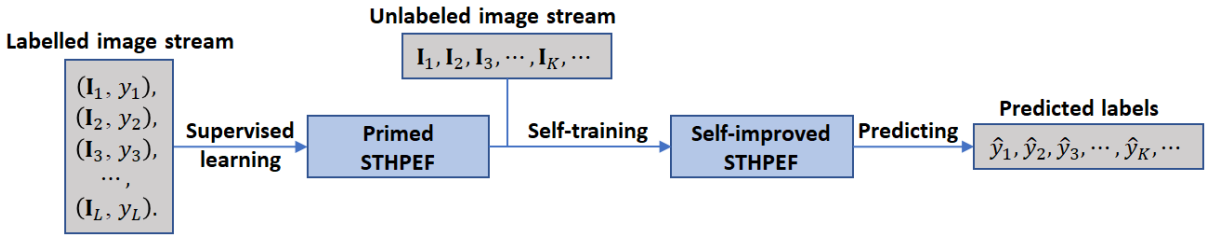


Fig. 2. Learning process of the STHPEF system

The zoomed-in architecture of the proposed ensemble framework is shown in Fig. 3. STHPEF consists of five components: 1) an image pool; 2) an image pre-processing module; 3) E feature descriptors of different types; 4) E STHP+ classifiers as the base learners, and 5) a joint decision-maker.

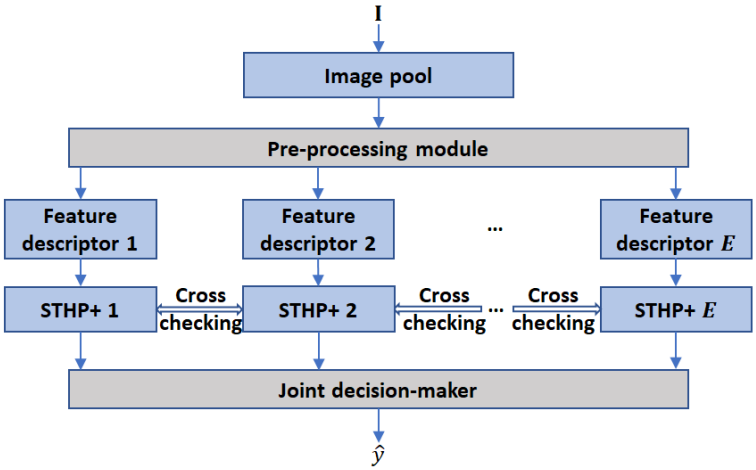


Fig. 3. Detailed architecture of the STHPEF system.

The image pool collects images and can store a maximum of Q images. During the supervised learning stage, the image pool will pass the newly received images directly to the pre-processing module. During the self-training stage, all the images in the pool will be packaged as an input image chunk once the pool is full, and then passed to the pre-processing module. The image pool becomes empty again and starts to collect new images for constructing the next chunk.

The pre-processing module serves mainly the following two purposes: to prepare the received images for feature extraction; and to increase the generalization ability of STHPEF and reduce over-fitting by image augmentation. Therefore, it involves various image pre-processing techniques, including scaling, segmentation, flipping, normalization and rotation.

After the input images have been processed by the pre-processing module, they are passed to the E feature descriptors for feature extraction. Each (assuming the ith one) of the E feature descriptors extracts a unique feature vector, denoted by $\boldsymbol{x}_k^i \leftarrow \mathrm{DR}^i(\mathbf{I}_k)$ ($\mathrm{DR}^i(\cdot)$ denotes the ith feature descriptor; $i = 1,2,\dots,E$) from every input image, $\mathbf{I}_k$ and passes it to the connected data pool. The feature descriptors used by STHPEF could be of any types that are used in computer vision (e.g., high-level ones such as pre-trained and/or fine-tuned DCNNs [55], or low-level ones such as Gist [56] and HOG [57]). The different types of feature descriptors in the ensemble framework can provide different views of input images and enhance the diversity of the base learners.

With new input images available, the E STHP+ classifiers self-learn from their input feature vectors to expand their knowledge bases following the same learning procedure as described in Section 3. Since each STHP+ classifier within the ensemble framework is trained with a different set of feature vectors extracted from input images by a different feature descriptor, these base learners will inevitably produce some disagreements on the pseudo-labelling outcomes. The disagreements occur when the base learners assign contradictory pseudo-labels to the feature vectors, $\boldsymbol{x}_k^i$ ($i = 1,2,\dots,E$) of the same image, $\mathbf{I}_k$ or some of them refuse to assign pseudo-labels due to low confidence levels. The disagreements are typically caused by the different descriptive abilities and foci of the employed feature descriptors. Such disagreements should be avoided because involving these images in self-training may significantly undermine the overall performance of the ensemble system as the base classifiers may contradict each other.

As one of its key features, the proposed ensemble framework has a cross-checking mechanism to avoid disagreements amongst the base learners, leading to increased precision. During the self-training process, assuming $\acute{\mathbf{X}}_j^i$ is the collection of the feature vectors of the jth input image chunk, $\mathbb{I}_j$ passed to the ith STHP+ classifier ($i = 1,2,\dots,E$), the base learner chooses a set of pseudo-labelled feature vectors from $\acute{\mathbf{X}}_j^i$ with the corresponding pseudo-labels denoted as $\hat{\mathbf{X}}_j^i$ and $\hat{\mathbf{Y}}_j^i$. Before using $\hat{\mathbf{X}}_j^i$ and $\hat{\mathbf{Y}}_j^i$ to update the base learner, the cross-checking mechanism of STHPEF examines the pseudo-labels produced by all base classifiers together to avoid discrepancies.

$$\textit{Condition } 4\text{:} \quad \begin{array}{c} \text{If } \left(\hat{y}_k^i = \hat{y}_k^l, \forall i \neq l\right) \\ \text{Then (no pseudo labelling discrepancy is observed on } \mathbf{I}_k) \end{array} \tag{16}$$

Here $\hat{y}_k^i$ is the pseudo-label of $\mathbf{I}_k$ produced by the ith base classifier added to $\widehat{\mathbf{Y}}_j^i$ during the pseudo-labelling process. If Condition 4 is satisfied, $\boldsymbol{x}_k^i$ and $\hat{y}_k^i$ are put into $\widetilde{\mathbf{X}}_j^i$ and $\hat{y}_k^i$, respectively ($i = 1,2,\dots,E$):

$$\widetilde{\mathbf{X}}_j^i \leftarrow \widetilde{\mathbf{X}}_j^i \cup \{\boldsymbol{x}_k^i\}; \quad \widetilde{\mathbf{Y}}_j^i \leftarrow \widetilde{\mathbf{Y}}_j^i \cup \{\hat{y}_k^i\} \tag{17}$$

where $\boldsymbol{x}_k^i \in \widehat{\mathbf{X}}_j^i$ is the feature vector of the unlabelled image $\mathbf{I}_k$ extracted by the ith feature descriptor; $\widetilde{\mathbf{X}}_j^i$ is a subset of $\widehat{\mathbf{X}}_j^i$ consisting of the feature vectors of these images such that all the base learners give the same pseudo-labels; and $\widetilde{\mathbf{Y}}_j^i$ represents the corresponding pseudo-labels.

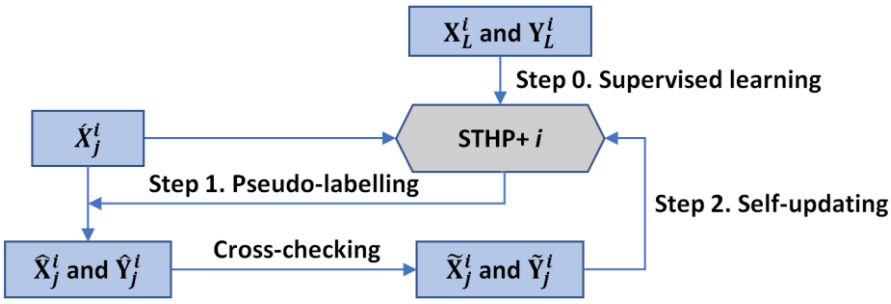


Fig. 4. Flowchart of the self-training process with cross-checking

Instead of $\widehat{\mathbf{X}}_j^i$ and $\widehat{\mathbf{Y}}_j^i$, $\widetilde{\mathbf{X}}_j^i$ and $\widetilde{\mathbf{Y}}_j^i$ are used for updating the ith base learner, and the feature vectors within $\widehat{\mathbf{X}}_j^i$ that fail to satisfy Condition 4 are put back to $\acute{\mathbf{X}}_j^i$ for the next round of pseudo-labelling. Fig. 4 provides a flowchart is given to depict the self-training process of the ith base classifier in STHPEF.

A joint decision-maker is used to determine the class labels of the unlabelled images after the self-training process has been completed. For each unlabelled image, $\mathbf{I}_k$, each STHP+ classifier produces a set of confidence scores using Eq. (2), denoted as $\boldsymbol{\Lambda}^i(\mathbf{I}_k) = \left[\lambda_1^i(\boldsymbol{x}_k^i), \lambda_2^i(\mathbf{I}_k), \dots, \lambda_C^i(\mathbf{I}_k)\right]^T$. The joint decision-maker then combines the E sets of confidence scores together using multiplication and determines the class label of $\mathbf{I}_k$ following the "winner takes all" principle:

$$\hat{y}_k \leftarrow j^*; \quad j^* = \underset{j=1,2,..,C}{\operatorname{argmax}}\left(\lambda_j(\mathbf{I}_k)\right) \tag{18}$$

where $\lambda_j(\mathbf{I}_k)$ is the aggregated confidence score of the jth class, $\lambda_j(\mathbf{I}_k) = \prod_{i=1}^{E} \lambda_j^i(\boldsymbol{x}_k^i)$.
The supervised learning, self-training and decision-making procedures of STHPEF are summarized by the following pseudo-codes.

Algorithm 3. STHPEF supervised learning procedure

inputs: $\mathbf{I}_1, \mathbf{I}_2, \dots, \mathbf{I}_L$; $y_1, y_2, \dots, y_L$; H; θ_o; ***algorithm begins***
for $k = 1$ **to** L **do:** *a.* read $\mathbf{I}_k$ and y_k, and store them in the image pool; *b.* pass $\mathbf{I}_k$ and y_k to the pre-processing module; *c.* empty the image pool; *d.* pre-process $\mathbf{I}_k$; *e.* **for** $i = 1$ **to** E **do:** *i.* extract feature vector: $\boldsymbol{x}_k^i \leftarrow \mathrm{DR}^i(\mathbf{I}_k)$; *ii.* update the ith base learner with $\boldsymbol{x}_k^i$ and y_k (*Algorithm 1*); *f.* **end for** **end for**
algorithm ends ***output:*** *trained ensemble system*

Algorithm 4. STHPEF self-training procedure

inputs: $\mathbf{I}_1, \mathbf{I}_2, \dots, \mathbf{I}_K$; γ_0; ***algorithm begins***
for $k = 1$ **to** K **do:**

a. read $\mathbf{I}_k$ and store it in the image pool;
b. **if** (image pool is full) **then:**
i. pass $\mathbb{I}_J$ to the pre-processing module;
ii. empty the image pool;
iii. pre-process $\mathbb{I}_J$;
iv. **for** $i = 1$ **to** E **do:**
1. $\acute{\mathbf{X}}_J^i \leftarrow \mathrm{DR}^i(\mathbb{I}_j)$;
2. pass $\acute{\mathbf{X}}_J^i$ to the *i*th base learner;
v. **end for**
vi. **while** ($\acute{\mathbf{X}}_J^i \neq \emptyset;\ \forall i = 1,2,..,E$):
1. **for** $i = 1$ **to** E **do:**
* obtain $\widehat{\mathbf{X}}_J^i$ and $\widehat{\mathbf{Y}}_J^i$ from $\acute{\mathbf{X}}_J^i$ by Conditions 2 and 3 (*Step 1, Algorithm 2*);
2. **end for**
3. **for** $i = 1$ **to** E **do:**
* obtain $\widetilde{\mathbf{X}}_J^i$ and $\widetilde{\mathbf{Y}}_J^i$ from $\widehat{\mathbf{X}}_J^i$ and $\widehat{\mathbf{Y}}_J^i$ by Condition 4 (*Cross-checking*);
4. **end for**
5. **if** ($\widetilde{\mathbf{X}}_J^i \neq \emptyset;\ \forall i = 1,2,..,E$) **then:**
* **for** $i = 1$ **to** E **do:**
- update the *i*th base learner with $\widetilde{\mathbf{X}}_J^i$ and $\widetilde{\mathbf{Y}}_J^i$ (*Step 2, Algorithm 2*);
* **end for**
6. **else:**
***** break **while** loop;
7. **end if**
vii. **end while**
viii. $J \leftarrow J + 1$;
c. **end if**
end for

algorithm ends
output: *self-trained ensemble system*

Algorithm 5. STHPEF decision-making procedure

inputs: $\mathbf{I}_1, \mathbf{I}_2, \ldots, \mathbf{I}_K$;
algorithm begins

for $k = 1$ **to** K **do:**
a. read $\mathbf{I}_k$ and store it in the image pool;
b. pass $\mathbf{I}_k$ to the pre-processing module;
c. pre-process $\mathbf{I}_k$;
d. **for** $i = 1$ **to** E **do:**
i. $\boldsymbol{x}_k^i \leftarrow \mathrm{DR}^i(\mathbf{I}_k)$;
ii. pass $\boldsymbol{x}_k^i$ to the *i*th base learner;
iii. produce $\boldsymbol{\Lambda}^i(\boldsymbol{x}_k^i)$ by (2);
e. **end for**
f. estimate $\hat{y}_k$ by (19);
end for

algorithm ends
output: $\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_K$;

5. Numerical Experiments and Results

Numerical experiments were conducted to evaluate the performance of the proposed STHPEF system for remote sensing scene classification. The STHP+ and STHPEF algorithms were developed on a MATLAB R2018a platform using a laptop with a dual core i7 CPU and 16 GB RAM. The DCNNs employed for feature extraction by the proposed approach were implemented using Tensorflow and fined-tuned on a Linux server with two NVIDIA GP100GL GPUs.

5.1. STHPEF System Implementation

The pre-processing module is composed of three sub-layers: *i)* scaling layer; *ii)* segmentation layer; *iii)* flipping layer. The image augmentation process of "centre, four corners and horizontal flipping" [58], [59] was adopted. The scaling layer resizes the remote sensing scenes into the uniform size of 248×248 pixels. The segmentation layer crops the centre and four corners from each image $\mathbf{I}$, and creates five new segments, $\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_5$ with the same size of 224×224 pixels. The flipping layer creates five additional new segments, $\mathbf{s}_6, \mathbf{s}_7, \ldots, \mathbf{s}_{10}$ from $\mathbf{I}$ by flipping horizontally $\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_5$. A total of 10 new sub-images were created from each labelled/unlabelled image by the pre-processing module.

The proposed STHPEF system employs three DCNNs for feature extraction ($E = 3$), namely, *i)* ResNet50 [60], *ii)* DenseNet121 [61] and *iii)* InceptionV3 [62]. The three DCNNs demonstrated strong performances in remote sensing scene classification [63]–[65]. To enhance their descriptive abilities, transfer learning [66] was leveraged to fine-tune the three DCNNs for extracting discriminative representations from remote sensing scenes. The NWPU45 (NWP) dataset [4] was employed for fine-tuning. The final prediction layers of the three employed DCNNs were replaced with two fully connected layers (each one consists of 1024 ReLUs with a dropout rate of 0.3) and a 45 way soft-max layer, given NWP has 45 different land-use categories. The size of input images for the three DCNNs was adjusted to 224×224 pixels. The remaining parameters were frozen prior to fine-tuning. The adaptive moment estimation (Adam) [67] algorithm was used to optimize the hyper-parameters of the two fully connected layers and the soft-max layer to minimize the categorical cross-entropy loss function.

During the fine-tuning process, 80% of the images of the full dataset were used as the training set and the remaining images were retained for validation. Both the training and validation sets were augmented five times larger by replacing each original image with five new images cropped from its central area and four corners. Following [68] with additional vertical flipping, the augmentation techniques (Table 2) were applied randomly to the training set to further increase the generalization capability of the fine-tuned DCNNs. The three DCNNs were fine-tuned for 25 epochs to avoid overfitting, where the batch size was set as 10 and the learning rate as 10^{-5}.

Table 2. Settings for augmentation [68]

Augmentation	Value
Rotation range	22
Width shifting range	0.19
Height shifting range	0.18
Horizontal flipping	True
Vertical flipping	True
Fill mode	Nearest

The final soft-max layers of the modified DCNNs were removed and the 1024×1 dimensional activations from the second fully connected layers extracted as the feature vectors of input images. The general architecture of the fine-tuned DCNNs used as feature descriptors of STHPEF is shown in Fig. 5.

Each feature descriptor extracts a 1024×1 dimensional unique discriminative representation, denoted as $\boldsymbol{x}^i$ from every labelled/unlabelled image as the arithmetic mean of feature vectors of its S_0 sub-images (produced by the pre-processing module), namely:

$$\boldsymbol{x}^i \leftarrow \mathrm{DR}^i(\mathbf{I}) = \frac{1}{S_0} \sum_{j=1}^{S_0} \mathrm{DR}^i(\mathbf{s}_j) \tag{19}$$

where $S_0 = 10$; $\boldsymbol{x}^i$ is the discriminative representation of $\mathbf{I}$ extracted by the *i*th feature descriptor, DR^i; $\mathrm{DR}^i(\mathbf{s}_j)$ is the feature vector extracted from the *j*th segment of $\mathbf{I}$.

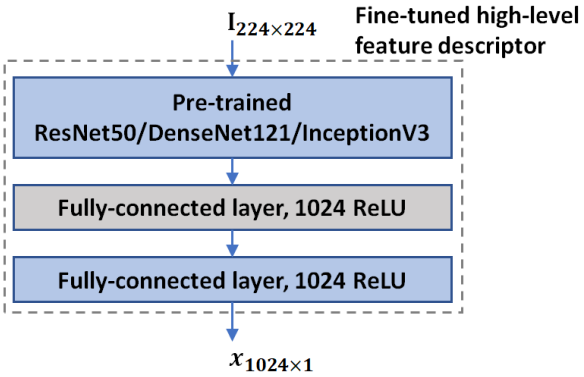


Fig. 5. Architecture of fine-tuned DCNNs used as feature descriptors.

Unless specifically declared otherwise, the externally controlled parameters for STHPEF were set as: $\theta_0 = \frac{\pi}{3}$; $H = 3$; $\gamma_0 = 1.1$; $W_0 = 4$ and $Q = 500$ for all numerical examples presented in this research. A sensitivity analysis was performed to investigate the impact of these parameters on system performance. In addition, an ablation analysis was also performed to test the effectiveness of STHPEF. Both analysis results were shown in the Appendix.

5.2. Dataset Description

Eight benchmark datasets for land-use scene classification were used: 1) NWP [4]; 2) WHU-RS19 (WHU) [69]; 3) UCMerced (UCM) [23]; 4) RSSCN7 (RSS) [70]; 5) AID [2]; 6) OPTIMAL-31 (OPT) [39]; 7) PatternNet (PTN) [71], and 8) RSI-CB256 (RSI) [72] (see Table 3). Web links to these datasets are provided in Table 4. Except for the NWP dataset used for fine-tuning the three employed high-level feature descriptors, all the other seven datasets were used for benchmark comparison.

Table 3. Key information on the benchmark remote sensing scenes

Dataset	Images	Categories	Images per category	Image size	Spatial resolution (m)
NWP	31500	45	700	256 × 256	0.2~30
WHU	950	19	50	600 × 600	up to 0.5
UCM	2100	21	100	256 × 256	0.3
RSS	2800	7	400	400 × 400	-
AID	10000	30	220~420	600 × 600	0.5~8
OPT	1860	31	60	256 × 256	-
PTN	30400	38	800	256 × 256	0.062~4.693
RSI	24747	35	198~1331	256 × 256	0.22~3

Table 4. Web links to the benchmark datasets

Dataset	Web link
NWP	https://www.tensorflow.org/datasets/catalog/resisc45
WHU	https://captain-whu.github.io/BED4RS/#
UCM	http://weegee.vision.ucmerced.edu/datasets/landuse.html
RSS	https://github.com/palewithout/RSSCN7
AID	https://captain-whu.github.io/AID/
OPT	https://drive.google.com/file/d/1Fk9a0DW8UyyQsR8dP2Qdakmr69NVBhq9/
PTN	https://sites.google.com/view/zhouwx/dataset
RSI	https://github.com/lehaifeng/RSI-CB

5.3. Performance Demonstration

The influence of different number of labelled images on the performance of STHPEF was investigated first using WHU, UCM and RSS datasets. The amount of labelled training images varied from 5% to 60% during the experiment, and the accuracy curves on the unlabelled images are shown in Fig. 6. It can be seen that STHPEF is

able to achieve 98.0%, 94.0% and 90.5% classification accuracy on the three datasets with 10% labelled images only, and the accuracy reaches up to 98.5%, 97.0% and 93.5% using 50% labelled training images.

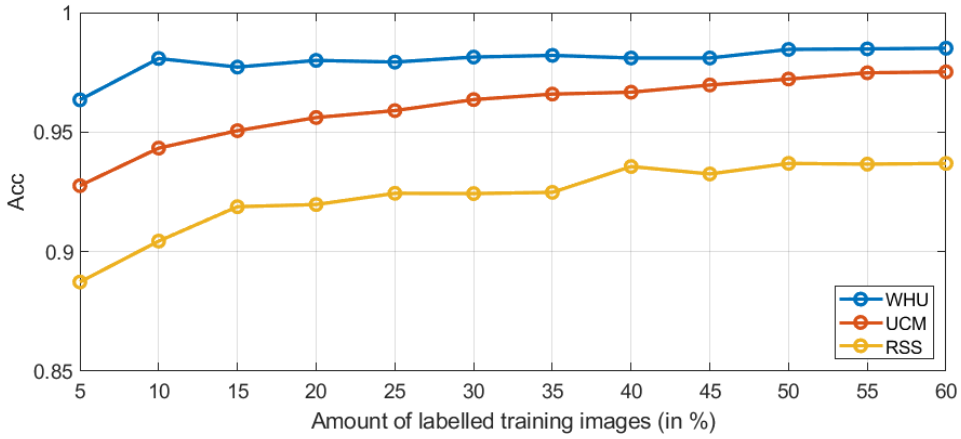


Fig. 6. Evolution of classification accuracy of STHPEF using different number of labelled training images

Next, the performance of STHPEF was compared with representative semi-supervised learning approaches on the WHU, UCM, RSS and AID datasets under the same experimental protocol as a benchmark comparison, including 1) SeRBIA [9]; 2) Local and global consistency (LGC) [52]; 3) Laplacian SVM (LapSVM) [51]; 4) Anchor graph regularization with kernel weights (AGRK) [53]; 5) Anchor graph regularization with local anchor embedding weights (AGRL) [53], and; 6) Efficient anchor graph regularization (EAGR) [73]. In the numerical examples, four different split ratios between labelled and unlabelled images were considered, namely 1:19, 1:9, 1:4 and 3:7. The obtained accuracy results on unlabelled images of the four datasets are reported in Table 5 using $mean \pm standard\ deviation$.

In this paper, the user-controlled parameters of SeRBIA are set as $\varphi = 1.1$ (the soft threshold for pseudo-labelling), $\gamma = 0$ (the threshold for identifying new classes) and $W = 500$ (the chunk size). Note that $\gamma = 0$ means SeRBIA will not learn new classes from unlabelled training images. LGC uses the k-nearest neighbour graph with $k = 5$, and the other parameter, α is set as $\alpha = 0.99$ [52]. LapSVM employs the "one versus all" strategy and uses the radial basis function kernel. Since the performance of LapSVM is highly sensitive to externally controlled parameters, three different parameter settings are considered here, namely, *i)* $\sigma = 10$, $\mu_I = 1$, $\mu_A = 10^{-6}$, $k = 15$ (as suggested by [51]); *ii)* $\sigma = 10$, $\mu_I = 0.5$, $\mu_A = 10^{-6}$, $k = 15$, and; *iii)* $\sigma = 1$, $\mu_I = 1$, $\mu_A = 10^{-5}$, $k = 10$. During the experiments, AGRK, AGRL and EAGR identify a total of $0.1K$ anchors from both labelled and unlabelled data. The number of closest anchors s for AnchorK, AnchorL and EAnchor is set as 3, and the iteration number of local anchor embedding for AnchorL is 10 [53].

Since the six comparative algorithms are all single-model approaches, ensemble models are created for them as the respective base classifiers for a fair comparison. In particular, since SeRBIA has a similar semi-supervised learning mechanism as STHP and its new version STHP+, the same framework as in Fig. 3 was used to create an ensemble system with SeRBIA, which was denoted as SeRBIAEF. For the other five approaches, the ensemble systems were created following the ensemble framework illustrated in Fig. 7. The key differences between the ensemble frameworks in Figs. 3 and 7 are 1) all the labelled and unlabelled images are given at once in the ensemble framework depicted in Fig. 7, and 2) there is no cross-checking between different base classifiers. The constructed ensemble systems with the five semi-supervised classifiers are named as AGRKEF, AGRLEF, LGCEF, LapSVMEF and EAGREF, respectively. The five ensemble classifiers determine the class labels of the unlabelled images based on the soft labels produced by the base learners jointly. As LapSVM used three different parameter settings, the created ensemble systems were re-denoted as LapSVMEF_1, LapSVMEF_2 and LapSVMEF_3, respectively to differentiate between them.

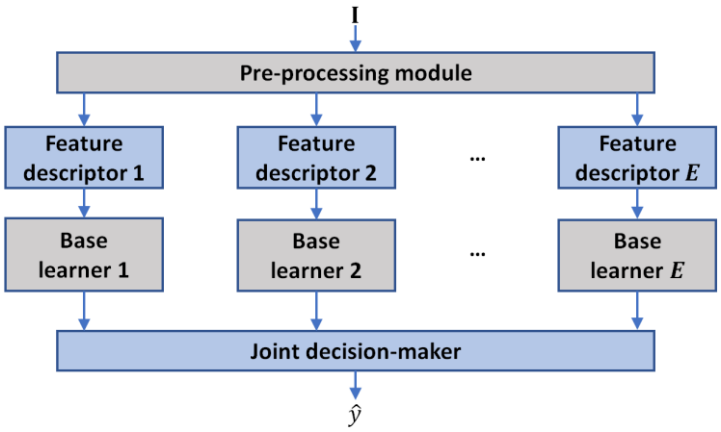


Fig. 7. Architecture of ensemble systems for comparison

One of the most popular semi-supervised ensemble frameworks, tri-training [54] was used in the experimental comparison. The tri-training framework employed a decision tree classifier as its base classifier. Thus, this tri-training ensemble framework is denoted as TriDTEF. In this paper, each of the three base classifiers within the tri-training framework learns from the feature vectors of both labelled and unlabelled training images extracted by its corresponding high-level feature descriptor. In addition, the *k*-nearest neighbour (KNN) and multi-layer perceptron (MLP) classifiers were also applied in the numerical comparison as the base learners of the tri-training framework. Here KNN uses $k = 5$ and the MLP has one hidden layer with 128 neurons. The hyper-parameters of the MLP were trained using gradient descent through backpropagation. The obtained ensemble frameworks with two types of base learners are denoted as TriKNNEF and TriMLPEF, respectively. The performances of TriDTEF, TriKNN and TriMLPEF are reported in Table 5.

Table 5. Classification accuracy comparison between different semi-supervised ensemble models

Algorithm	Split Ratio	WHU	UCM	RSS	AID
STHPEF	1:19	0.9635±0.0181	0.9276±0.0104	0.8873±0.0135	0.8765±0.0044
SeRBIAEF		0.9502±0.0198	0.9277±0.0107	0.8844±0.0153	0.8728±0.0050
AGRKEF		0.9163±0.0591	0.9165±0.0134	0.8749±0.0112	0.8700±0.0045
AGRLEF		0.8988±0.1088	0.9134±0.0094	0.8760±0.0118	0.8672±0.0029
LGCEF		0.8267±0.1554	0.9180±0.0089	**0.8923±0.0058**	0.8666±0.0043
LapSVMEF_1		0.2719±0.2388	0.9041±0.0147	0.8835±0.0094	0.6139±0.1505
LapSVMEF_2		0.4698±0.4134	0.9076±0.0161	0.8830±0.0101	0.7694±0.0821
LapSVMEF_3		0.2900±0.2593	0.9057±0.0178	0.8817±0.0105	0.6665±0.1308
EAGREF		**0.9684±0.0032**	**0.9326±0.0102**	0.8922±0.0069	**0.8850±0.0033**
TriDTEF		0.3621±0.0562	0.7102±0.0257	0.7882±0.0141	0.7040±0.0155
TriKNNEF		0.6768±0.0858	0.8630±0.0229	0.8548±0.0189	0.8415±0.0191
TriMLPEF		0.6184±0.1064	0.8099±0.0217	0.8717±0.0202	0.6431±0.0298
STHPEF	1:9	**0.9808±0.0042**	**0.9433±0.0067**	**0.9044±0.0049**	**0.8988±0.0029**
SeRBIAEF		0.9657±0.0106	0.9405±0.0066	0.9027±0.0060	0.8987±0.0033
AGRKEF		0.9509±0.0090	0.9408±0.0069	0.8864±0.0098	0.8850±0.0023
AGRLEF		0.9577±0.0112	0.9349±0.0070	0.8865±0.0116	0.8836±0.0056
LGCEF		0.8236±0.0841	0.9333±0.0072	0.8950±0.0077	0.8740±0.0039
LapSVMEF_1		0.3797±0.2139	0.9341±0.0056	0.9031±0.0077	0.8244±0.0253
LapSVMEF_2		0.6712±0.1438	0.9359±0.0052	0.9017±0.0074	0.8624±0.0114
LapSVMEF_3		0.4125±0.2201	0.9361±0.0064	0.9024±0.0086	0.8422±0.0177
EAGREF		0.9667±0.0039	0.9423±0.0034	0.8975±0.0067	0.8970±0.0023
TriDTEF		0.6646±0.0398	0.7921±0.0159	0.8183±0.0104	0.7375±0.0264
TriKNNEF		0.9185±0.0299	0.9194±0.0105	0.8802±0.0113	0.8680±0.0120
TriMLPEF		0.8375±0.0359	0.8684±0.0111	0.8881±0.0071	0.6460±0.0239
STHPEF	1:4	**0.9800±0.0034**	**0.9561±0.0034**	**0.9197±0.0067**	0.9147±0.0021
SeRBIAEF		0.9787±0.0034	0.9557±0.0046	0.9196±0.0058	**0.9152±0.0032**
AGRKEF		0.9635±0.0053	0.9455±0.0057	0.9055±0.0076	0.9024±0.0029
AGRLEF		0.9693±0.0063	0.9476±0.0074	0.9039±0.0082	0.9035±0.0038

LGCEF		0.9274±0.0406	0.9385±0.0056	0.9015±0.0045	0.8798±0.0032
$LapSVMEF_1$		0.8974±0.0397	0.9442±0.0044	0.9164±0.0051	0.8835±0.0039
$LapSVMEF_2$		0.9402±0.0169	0.9448±0.0055	0.9165±0.0051	0.8957±0.0044
$LapSVMEF_3$		0.9052±0.0398	0.9436±0.0045	**0.9197±0.0043**	0.8829±0.0046
EAGREF		0.9730±0.0038	0.9482±0.0035	0.9056±0.0055	0.9076±0.0025
TriDTEF		0.8080±0.0184	0.8289±0.0109	0.8417±0.0121	0.7710±0.0191
TriKNNEF		0.9605±0.0090	0.9320±0.0097	0.8999±0.0057	0.8854±0.0081
TriMLPEF		0.9298±0.0125	0.8846±0.0097	0.9003±0.0075	0.6309±0.0270
STHPEF	3:7	**0.9814±0.0034**	0.9636±0.0050	**0.9243±0.0053**	**0.9229±0.0013**
SeRBIAEF		0.9798±0.0039	**0.9653±0.0039**	0.9221±0.0063	0.9226±0.0014
AGRKEF		0.9637±0.0047	0.9549±0.0037	0.9107±0.0079	0.9095±0.0019
AGRLEF		0.9710±0.0063	0.9554±0.0062	0.9116±0.0078	0.9117±0.0019
LGCEF		0.9124±0.0395	0.9387±0.0044	0.9014±0.0086	0.8809±0.0033
$LapSVMEF_1$		0.9387±0.0158	0.9467±0.0038	0.9160±0.0073	0.8927±0.0062
$LapSVMEF_2$		0.9566±0.0061	0.9471±0.0040	0.9136±0.0078	0.9051±0.0052
$LapSVMEF_3$		0.9443±0.0139	0.9469±0.0036	0.9181±0.0064	0.8947±0.0034
EAGREF		0.9724±0.0042	0.9519±0.0060	0.9097±0.0070	0.9102±0.0014
TriDTEF		0.8492±0.0213	0.8544±0.0071	0.8472±0.0111	0.7787±0.0183
TriKNNEF		0.9698±0.0084	0.9401±0.0082	0.9081±0.0079	0.8952±0.0052
TriMLPEF		0.9360±0.0120	0.8952±0.0087	0.9041±0.0083	0.6318±0.0310

For clear demonstration, the average classification accuracy rates of the 12 semi-supervised ensemble models across the four datasets are presented in Fig. 8. In addition, the average accuracies of the 12 models on each dataset are ranked in descending order in Table 6. The overall ranks are also provided in the same table. The STHPEF achieved the highest classification accuracy on the unlabelled images of the four benchmarks with 10%, 20% and 30% of labelled training images. Clearly, the superior performance of STHPEF is shown by ranking at the top in Table 6.

Table 6. Classification accuracy ranks of different models from the highest to the lowest.

Algorithm	Split Ratio				Overall
	1:19	1:9	1:4	3:7	
STHPEF	2.5000	**1.0000**	**1.3750**	**1.2500**	**1.5313**
SeRBIAEF	3.0000	3.0000	2.0000	1.7500	2.4375
AGRKEF	5.5000	5.5000	5.5000	5.5000	5.5000
AGRLEF	6.0000	6.2500	5.0000	4.0000	5.3125
LGCEF	4.2500	7.5000	9.2500	10.5000	7.8750
$LapSVMEF_1$	9.5000	8.0000	7.7500	7.5000	8.1875
$LapSVMEF_2$	7.5000	7.0000	5.7500	6.0000	6.5625
$LapSVMEF_3$	9.0000	7.2500	7.1250	6.5000	7.4688
EAGREF	**1.2500**	3.2500	3.7500	5.0000	3.3125
TriDTEF	10.7500	11.2500	11.7500	11.7500	11.3750
TriKNNEF	8.7500	8.5000	8.5000	7.5000	8.3125
TriMLPEF	10.0000	9.5000	10.2500	10.7500	10.1250

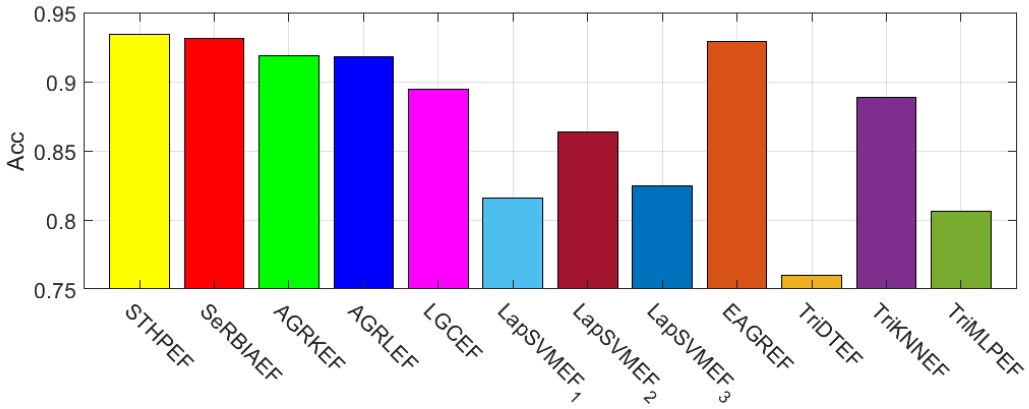


Fig. 8. Average classification accuracy rates of different semi-supervised ensemble models across four datasets

It is crucial to show whether the accuracy increase of STHPEF over the rest is of statistical significance. Wilcoxon signed rank tests were applied and the test outcomes for each individual model are reported in Table 7 in terms of *p*-value and *z*-score obtained during the experiments with 10% of data used as labelled images. The majority of *p*-values returned by the tests are zero or approximately zero, far below the level of significance specified by α = 0.05, suggesting that the accuracy of STHPEF is significantly greater than for the other methods.

Table 7. Statistical Wilcoxon signed-rank test analysis results.

STHPEF vs	WHU		UCM		RSS		AID	
	p-value	*z*-value	*p*-value	*z*-value	*p*-value	*z*-value	*p*-value	*z*-value
SeRBIAEF	0.1040	1.6258	0.0097	-2.5880	0.0002	3.7215	0.0000	-10.1000
AGRKEF	0.0000	-6.1027	0.0137	2.4650	0.0000	15.3228	0.0000	-4.5657
AGRLEF	0.0059	-2.7516	0.0106	2.5540	0.0000	13.2246	0.0000	-5.0600
LGCEF	0.0000	8.4408	0.0012	-3.2479	0.0000	23.9263	0.0000	20.0189
$LapSVMEF_1$	0.6518	0.4512	0.0000	-4.2797	0.0000	17.9199	0.0013	3.2085
$LapSVMEF_2$	0.0000	5.4629	0.0003	-3.6052	0.0000	19.6135	0.0000	4.4528
$LapSVMEF_3$	0.5941	0.5329	0.0000	-5.8269	0.0000	18.4499	0.0000	6.2499
EAGREF	0.0001	-3.9570	0.0055	-2.7784	0.0000	19.6585	0.0011	3.2554
TriDTEF	0.0000	38.6686	0.0000	37.9060	0.0000	33.6535	0.0000	99.7485
TriKNNEF	0.0000	11.5691	0.0000	13.9545	0.0000	26.5347	0.0000	62.9152
TriMLPEF	0.0000	16.3617	0.0000	26.3243	0.0000	26.4937	0.0000	100.0786

Computational efficiency is one of key factors for evaluating the proposed ensemble system in real-world applications, particularly in the context of green computing [74]. With 10% of data used as labelled images, we report the computational time used for different models in Table 8. For visual clarity, the average computational times of these ensemble models across the four datasets are depicted in Fig. 9. The STHPEF achieves comparable computational efficiency with other methods, and significantly faster than LGCEF, LapSVMEF, and TriMLPEF. Unlike other semi-supervised learning models, STHPEF is able to learn from data chunk-by-chunk similar to SeRBIA [9], which provides flexibility to handle large-scale datasets. As such, the computational efficiency of STHPEF does not decrease significantly with large remote sensing dataset (e.g. AID).

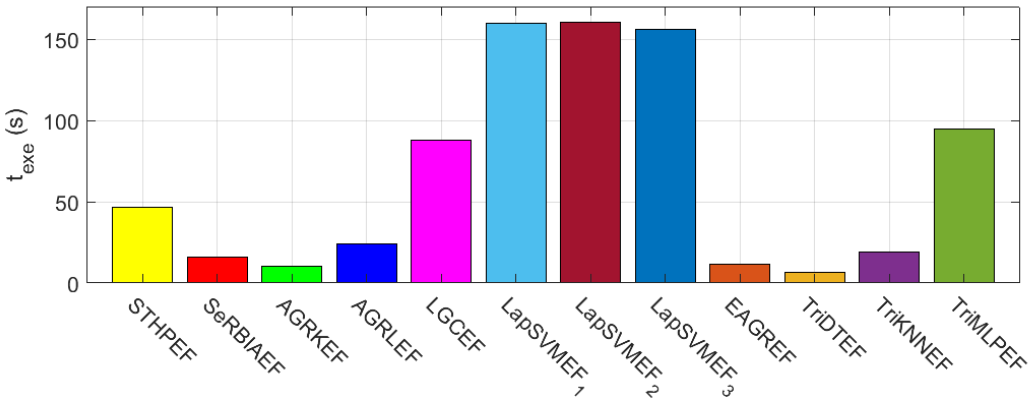


Fig. 9. Average computational times of the different semi-supervised ensemble models across the four datasets

Table 8. Average execution time (in second) amongst different semi-supervised ensemble models

Algorithm	WHU	UCM	RSS	AID
STHPEF	3.4607	8.4301	22.2339	151.8168
SeRBIAEF	0.8375	3.4113	8.0086	51.7548
AGRKEF	0.4142	1.7168	3.1758	37.1372
AGRLEF	3.6571	8.5552	13.0320	71.0997
LGCEF	0.9080	5.0580	11.3423	335.1835
LapSVMEF$_1$	4.6270	20.7529	12.0283	601.5974
LapSVMEF$_2$	4.6185	20.5077	12.1135	603.5129
LapSVMEF$_3$	4.5565	20.5365	11.9407	586.3214
EAGREF	0.6292	2.1736	3.8455	40.4778
TriDTEF	0.6257	1.6889	1.2469	21.4427
TriKNNEF	1.7904	4.2290	5.1440	65.4547
TriMLPEF	64.6624	53.8152	68.5902	190.4936

5.4. Benchmark Comparison

The proposed ensemble framework was compared with mainstream methods on seven benchmark datasets for remote sensing scene classification. Similar to [2], [75], [76], the train/test split ratios of WHU dataset were set as 4:6 and 6:4. For the UCM and RSI datasets, the split ratios were set as 5:5 and 8:2. The RSS and AID datasets were split in the ratios of 2:8 and 5:5. The train/test split ratios of OPT and PTN datasets were both set as 8:2. The training set was used for priming STHPEF, and the testing set left as the unlabelled set to self-train and build an accurate predictive ensemble model. The accuracy of STHPEF was measured by comparing the predicted class labels of unlabelled images with 'ground' reference data. The accuracies of STHPEF on the seven benchmark datasets were compared with a selected group of benchmark algorithms (Table 9). The reported results for the alternative approaches were obtained from the literature under the standard experimental protocols.

Since STHPEF is a semi-supervised learning framework, it can learn from both labelled and unlabelled images. To compare with the fully supervised counterparts, the numerical experiments were repeated with modified experimental protocols, such that only half of the training set was used for priming STHPEF and the other half was joined to the testing unlabelled set for STHPEF to self-learn. Therefore, the number of labelled training images was reduced to half for priming STHPEF. The prediction accuracy of STHPEF under the modified experimental protocols is reported using an asterisk "*" (STHPEF*) in Table 9.

The proposed ensemble framework can achieve comparable results or even be more accurate than the popular models, but with only half of the labelled training samples (Table 9). For example, STHPEF is able to achieve accuracy of 96.67%, 99.91%, 98.67% and 99.07% on UCM, OPT, PTN and RSI using only 40% of the images as the labelled training sets, compared with the accuracy of 97.60%, 99.73%, 98.87% and 99.34% using 80% of labelled training images. The accuracy maintains as 95.75%, 92.34%, 91.99% and 98.85% on UCM, RSS, AID and PTN with 25% labelled training images only, in comparison with 97.22%, 93.69%, 93.18%, 99.15% using 50% labelled training images. The maximum difference is extremely small in the two cases (0.93% and 1.47%).

This clearly shows that STHPEF can utilise the unlabelled images effectively to construct a highly precise ensemble model. Thus, STHEPF has a much reduced requirement for labelled images, but can achieve the same level of accuracy as the popular benchmark approaches.

Table 9. Performance comparison between STHEPF and benchmark approaches

Algorithm	WHU		UCM	
	4:6	6:4	1:1	4:1
STHPEF*	0.9800±0.0034	0.9814±0.0034	0.9575±0.0027	0.9667±0.0040
STHPEF	0.9810±0.0041	0.9840±0.0053	0.9722±0.0050	0.9760±0.0064
CaffeNet [2]	0.9511±0.0120	0.9624±0.0056	0.9398±0.0067	0.9502±0.0081
VGG-VD-16 [2]	0.9544±0.0060	0.9605±0.0091	0.9414±0.0069	0.9521±0.0120
GoogLeNet [2]	0.9312±0.0082	0.9471±0.0133	0.9270±0.0060	0.9431±0.0089
Two-stream deep fusion [55]	0.9823±0.0056	0.9892±0.0052	-	-
TEX-Net-LF [40]	0.9848±0.0037	0.9888±0.0049	-	-
SalM^{3}LBP-CLM [77]	0.9535±0.0076	0.9638±0.0082	0.9421±0.0075	0.9575±0.0080
GBNet [75]	0.9732±0.0032	**0.9925±0.0050**	0.9705±0.0019	0.9857±0.0048
ARCNet-VGG16 [39]	-	-	0.9681±0.0014	0.9912±0.0040
CAD [1]	-	-	**0.9857±0.0033**	**0.9916±0.0027**
Fine-tune MobileNet V2 [33]	0.9682±0.0035	0.9815±0.0033	-	-
SE-MDPMNet [33]	**0.9864±0.0021**	0.9897±0.0024	-	-
EfficientNetB3-Basic [38]	0.9728±0.0024	0.9768±0.0010	0.9763±0.0006	0.9873±0.0020
EfficientNetB3-Attn-2 [38]	0.9860±0.0040	0.9868±0.0093	0.9790±0.0036	0.9912±0.0022
Algorithm	RSS		AID	
	1:4	1:1	1:4	1:1
STHPEF*	0.9044±0.0049	0.9234±0.0051	0.8988±0.0029	0.9199±0.0022
STHPEF	0.9197±0.0067	0.9369±0.0042	0.9147±0.0021	0.9318±0.0022
CaffeNet [2]	0.8557±0.0095	0.8825±0.0062	0.8686±0.0047	0.8953±0.0031
VGG-VD-16 [2]	0.8398±0.0091	0.8718±0.0094	0.8659±0.0029	0.8964±0.0036
GoogLeNet [2]	0.8255±0.0111	0.8584±0.0092	0.8344±0.0040	0.8639±0.0055
TEX-Net-LF [40]	0.9245±0.0045	0.9400±0.0057	-	-
MARTA GAN [8]	-	-	0.7539±0.0049	0.8157±0.0033
Attention GAN [28]	-	-	0.7895±0.0023	0.8452±0.0018
SalM^{3}LBP-CLM [77]	-	-	0.8692±0.0035	0.8976±0.0045
GBNet [75]	-	-	0.9220±0.0023	0.9548±0.0012
MSNet [76]	-	-	0.9559±0.0015	0.9697±0.0027
ARCNet-VGG16 [39]	-	-	0.8875±0.0040	0.9310±0.0055
CAD [1]	-	-	**0.9573±0.0022**	0.9716±0.0026
GANet [78]	-	-	0.8796±0.0023	0.9136±0.0018
LANet [78]	-	-	0.8941±0.0024	0.9235±0.0024
GLANet [78]	-	-	0.9502±0.0028	0.9666±0.0019
MAA-CNN [27]	-	-	0.9554±0.0008	**0.9748±0.0007**
Fine-tune MobileNet V2 [33]	0.8904±0.0017	0.9246±0.0066	0.9413±0.0028	0.9596±0.0027
SE-MDPMNet [33]	0.9265±0.0013	0.9471±0.0015	0.9468±0.0017	0.9714±0.0015
EfficientNetB3-Basic [38]	0.9206±0.0039	0.9439±0.0010	0.9343±0.0033	0.9537±0.0041
EfficientNetB3-Attn-2 [38]	**0.9330±0.0019**	**0.9617±0.0023**	0.9445±0.0073	0.9656±0.0012
Algorithm	OPT	PTN	RSI	
	4:1	4:1	1:1	4:1
STHPEF*	**0.9991±0.0000**	0.9867±0.0007	0.9885±0.0006	0.9907±0.0008
STHPEF	0.9973±0.0000	0.9887±0.0016	**0.9915±0.0011**	0.9934±0.0009
GBNet [75]	0.9328±0.0027	-	-	-
MSNet [76]	0.9392±0.0041	-	-	-
ARCNet-VGG16 [39]	0.9270±0.0035	-	-	-
ARCNet-ResNet34 [39]	0.9128±0.0045	-	-	-
ARCNet-AlexNet [39]	0.8575±0.0035	-	-	-
Fine-tune VGGNet16 [39]	0.8745±0.0045	-	-	-
Fine-tune GoogLeNet [39]	0.8257±0.0012	-	-	-
Fine-tune AlexNet [39]	0.8122±0.0019	-	-	-

MAA-CNN [27]	0.9570±0.0054	-	-	-
GANet [78]	-	**0.9970±0.0026**	-	-
LANet [78]	-	0.9961±0.0023	-	-
GLANet [78]	-	**0.9970±0.0015**	-	-
EfficientNetB3-Basic [38]	0.9476±0.0026	-	-	-
EfficientNetB3-Attn-2 [38]	0.9586±0.0022	-	-	-
SIFT [72]	-	-	0.3796±0.0027	0.4012±0.0034
LBP [72]	-	-	0.6910±0.0020	0.7198±0.0036
CH [72]	-	-	0.8408±0.0026	0.8472±0.0033
Gist [72]	-	-	0.6174±0.0035	0.6359±0.0045
ResNet [72]	-	-	-	0.9502
GoogLeNet [72]	-	-	-	0.9407
VGG-VD-16 [72]	-	-	-	0.9513
Enhanced Fusion of DCNNs [65]	-	**0.9970**	-	**0.9950**

* Results obtained under the modified experimental protocols.

Some existing research has reported highly accurate results. For example, CAD [1] provides the highest accuracy on the UCM and AID datasets, SE-MDPMNet [33] and GBNet [75] performs the best on the WHU datasets, EfficientNetB3-Attn-2 [38] achieves the highest accuracy on the RSS dataset, etc.. However, these approaches [1], [27], [33], [38], [75] trained or fine-tuned their DNN models directly on the targeted datasets. Although these approaches claimed to report the classification accuracy rates on the testing sets, the testing sets were already involved in DCNN training as validation sets. As a consequence, the ground reference data of the testing set was utilized in the training process to some extent. In addition, these approaches [27], [65] also used a carefully selected set of image augmentation techniques to augment the sizes of the training sets significantly. As aforementioned, the classification accuracy can be increased, but they are likely to be overfitting.

In contrast, STHPEF employs DCNNs that are fine-tuned on a separate NWP dataset as the feature descriptors, instead of on the targeted datasets. In this way, STHPEF performs knowledge transfer *twice* during the experiments. The first knowledge transfer occurs during the fine-tuning process from natural images to remote sensing images and the second knowledge transfer occurs during feature extraction from one remote sensing scene to another. Although STHPEF is given both the labelled training and unlabelled testing sets together to perform semi-supervised learning, the labelling information of the testing sets is not utilized at all during the self-training process. In addition, only basic image augmentation techniques are employed by the proposed framework, and this can be easily expanded. Hence, the reported results by STHPEF are more realistic for real-world application scenarios, where commonly only limited prior knowledge and labelled training images are available. On the other hand, it is worth noting that the architecture of the STHPEN system is highly flexible and can be adjusted for different classification purposes. One may further apply other image augmentation techniques, such as rotation, shifting and zooming [65], or employ more advanced feature descriptors with other base learners for joint decision-making, these are all interesting subjects of research deserving future investigation.

5.5. Application to Satellite Sensor Images

In this subsection, examples based on large-scale satellite sensor images were presented to demonstrate the applicability of STHPEF to fully automated image analysis in real-world scenarios.

The UCM dataset was used to initialise the proposed ensemble framework. This dataset contains 21 commonly seen land-use categories (Fig. 10). Thanks to the relatively smaller image size, simpler geometrical structures and lesser variety in terms of semantic content within individual images, this dataset is suitable for priming the knowledge base of STHPEF.

Six satellite sensor images, given by Figs. 11-16, captured from different areas of the UK were captured from Google Earth (Google Inc.) with the same size of 800×1200 pixels. The spatial resolutions of these images vary from 0.3 m to 30 m. Great variation in terms of geometric structure, spatial patterns and semantic content are observable in these images.

Fig. 10. Example images of UCM dataset

During each experiment, the satellite sensor image was segmented by a 4×6 grid net into 24 sub-images, denoted as $\mathbf{I}_k$ ($k = 1,2,\ldots,24$). Each image segment contains a non-overlapping local region within the large-size image, and the size of these segments is 200×200 pixels. 24 image segments were used as unlabelled images for STHPEF to self-improve its knowledge base using *Algorithm 4*, and STHPEF assigned different land-use class labels to these segments based on the semantic content presented in these local regions.

Since a local region of a satellite sensor image may exhibit high-level semantic features of different land-use categories, assigning only one land-use class label to each subregion may result in a significant loss of information. Instead, a feasible solution is to assign one or multiple land-use class labels to the image segments based on the semantic content presented in these local regions [9]. Hence, Condition 5 was used to identify the relevant land-use categories that share similar semantic features with each image segment, $\mathbf{I}_k$ [9]:

$$\begin{aligned} Condition\ 5\text{:}\quad & \text{If } \left(\gamma_1 \cdot \lambda_j(\mathbf{I}_k) > \max_{l=1,2,\ldots,C}\left(\lambda_l(\mathbf{I}_k)\right)\right) \\ & \text{Then } (\mathbf{I}_k \text{ possesses the semantic feature of the jth category}) \end{aligned} \tag{20}$$

where $\gamma_1 = \gamma_0^E$. Assuming that there are T_k land-use categories satisfising Condition 5, the likelihoods (ratios of importance of distinctive semantic features) of the T_k categories in $\mathbf{I}_k$ are calculated using Eq. (21) [9]:

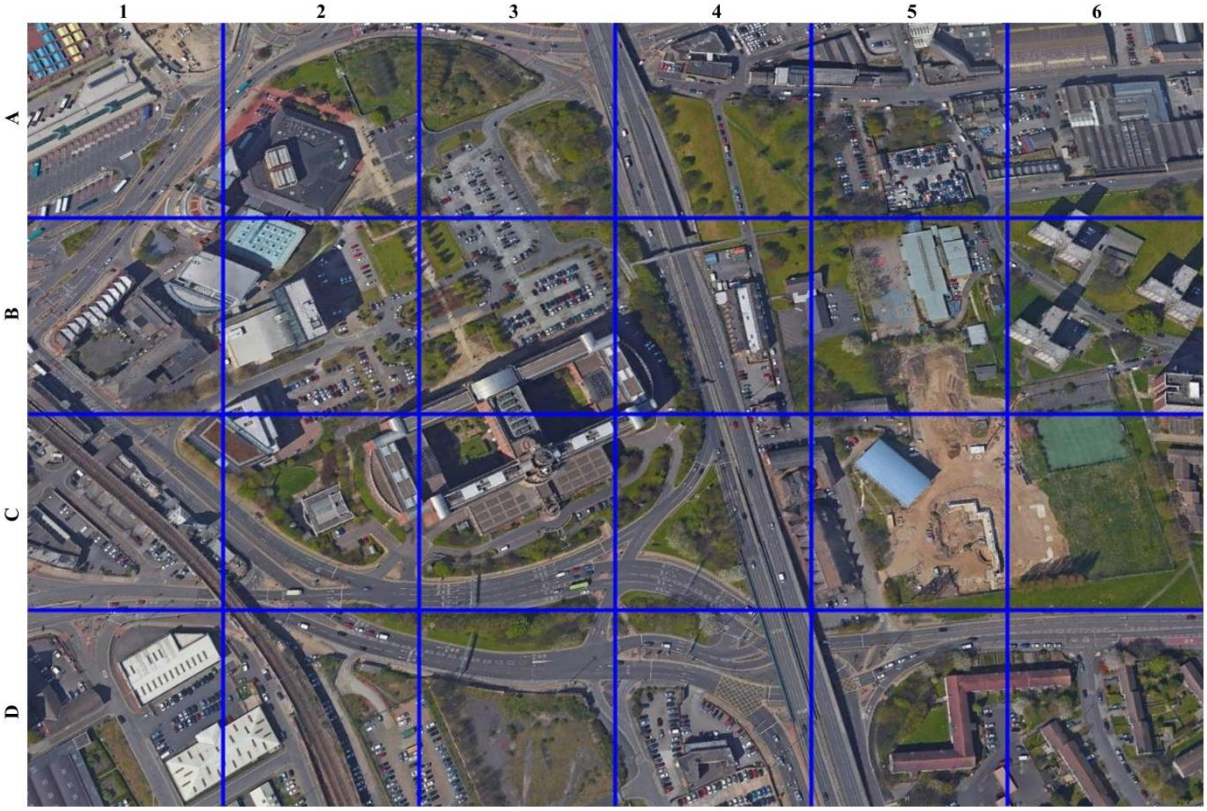
$$\mathcal{L}_j^*(\mathbf{I}_k) = \frac{\bar{\lambda}_j^*(\mathbf{I}_k)}{\sum_{l=1}^{T_k} \bar{\lambda}_l^*(\mathbf{I}_k)} \tag{21}$$

where $\mathcal{L}_j^*(\mathbf{I}_k)$ is the likelihood of the jth land-use category satisfying Condition 5; $\bar{\lambda}_j^*(\mathbf{I}_k)$ is the standardized confidence score calculated using Eq. (22):

$$\bar{\lambda}_j^*(\mathbf{I}_k) = \frac{\lambda_j^*(\mathbf{I}_k) - \mu_{j,k}}{\sigma_{j,k}} \tag{22}$$

Here $\mu_{j,k}$ and $\sigma_{j,k}$ are the mean and standard deviation of the C aggregated confidence scores that $\mathbf{I}_k$ receives.

Results for the six satellite sensor images produced by STHPEF are presented in Figs. 11-16 in the form of a 4×6 table with the background in white and blue colours. In this research, the maximum value of T_k was set to five for visual clarity.



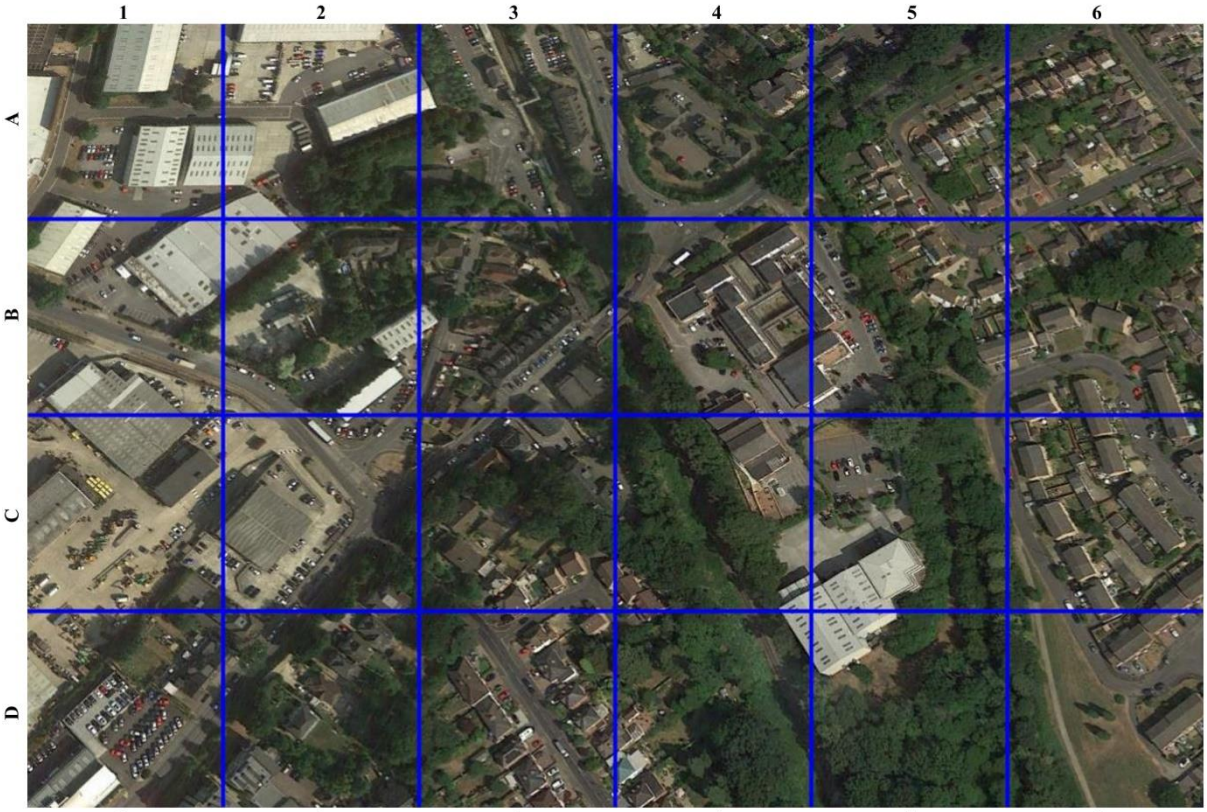
	1	2	3	4	5	6
A	Intersection: 23.4% Storage tanks: 22.7% Overpass: 22% Parking lot: 16.5% Freeway: 15.4%	Buildings: 59.5% Storage tanks: 40.5%	Overpass: 100%	Freeway: 58.5% Overpass: 41.5%	Parking lot: 100%	Buildings: 100%
B	Buildings: 50.8% Storage tanks: 49.2%	Buildings: 100%	Parking lot: 100%	Freeway: 100%	Buildings: 30.4% Medium residential: 28% Dense residential: 21.6% Mobile home park: 20%	Medium residential: 57.4% Dense residential: 42.6%
C	Storage tanks: 100%	Overpass: 39.7% Tennis courts: 30.8% Buildings: 29.4%	Tennis courts: 53.8% Buildings: 46.2%	Overpass: 55.9% Freeway: 44.1%	Medium residential: 100%	Baseball diamond: 100%
D	Intersection: 100%	Freeway: 100%	Intersection: 100%	Overpass: 100%	Overpass: 100%	Dense residential: 100%

Fig. 11. Classification result for satellite sensor image 1

From Figs. 11-16, it is clear that STHPEF (primed by UCM dataset) is capable of self-expanding its knowledge base with these unlabelled image segments, capturing the most distinctive semantic features and assigning one or multiple land-use class labels to each local region of the satellite sensor images with great precision. In most cases, the respective likelihoods of the relevant land-use categories associated with the local regions of these images are also described accurately. These illustrative examples demonstrate the great promise of the proposed STHPEF in real-world applications.

On the other hand, one may also notice that in some rare cases, STHPEF failed to make correct categorizations. For example, STHPEF assigned the land-use class label "river" to the local regions "D1" and "D2" of Fig. 13 instead of the correct label "forest". In addition, it wrongly allocated the local regions "B2" of Fig. 14, "B2" of Fig. 15 and "D4" of Fig. 16 to the land-use categories "overpass", "beach" and "mobile home park", which should

be “buildings”, “river” and “residential”/“building”, respectively. The main reason for such mis-categorisations is the very high similarity amongst high-level semantic features shared by different land-use categories. Nevertheless, this issue can be addressed by making a pre-selection on benchmark datasets and removing the less representative images [9].



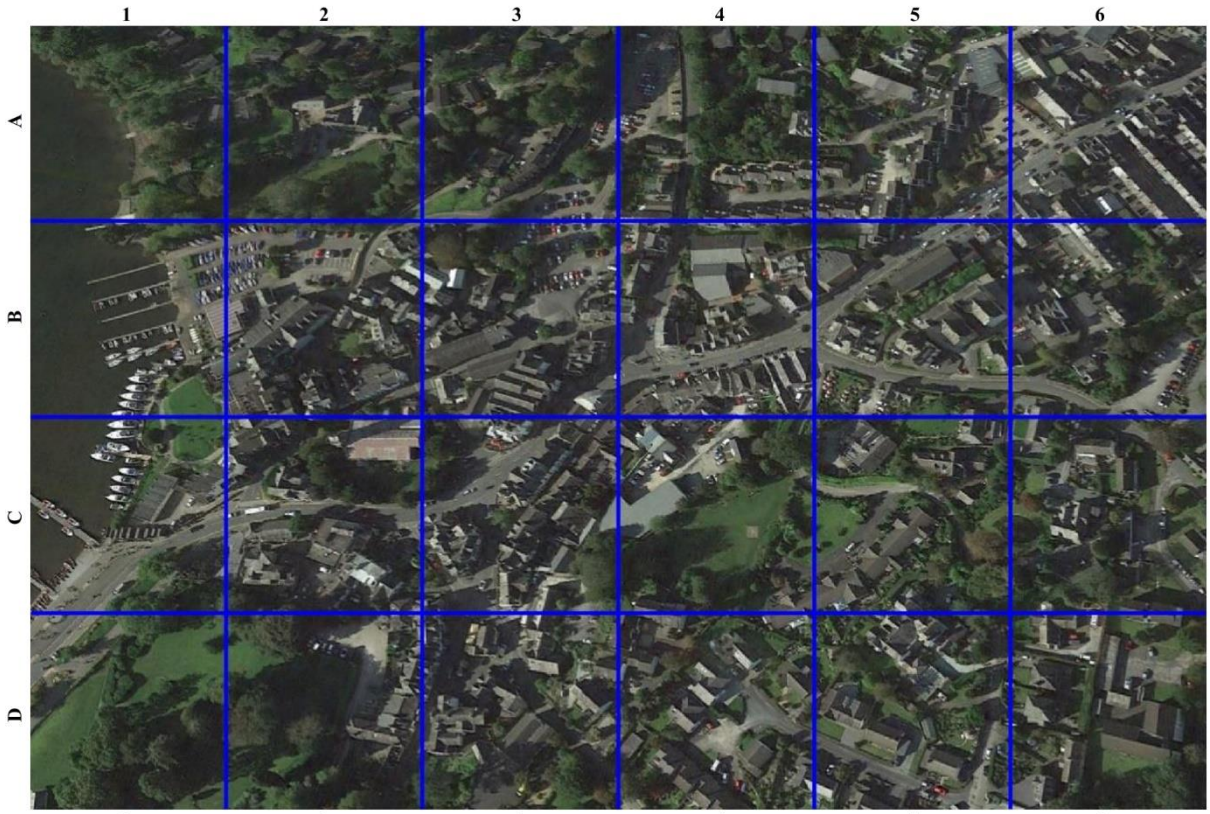
	1	2	3	4	5	6
A	Buildings: 100%	Buildings: 100%	Storage tanks: 40.7% Mobile home park: 34.8% Intersection: 24.6%	Medium residential: 53.1% Sparse residential: 46.9%	Medium residential: 100%	Dense residential: 51.1% Medium residential: 48.9%
B	Buildings: 100%	Buildings: 51.2% Dense residential: 48.8%	Dense residential: 55.6% Mobile home park: 44.4%	Medium residential: 53.1% Dense residential: 46.9%	Medium residential: 41.1% Dense residential: 31.6% Sparse residential: 27.3%	Medium residential: 100%
C	Medium residential: 58.8% Mobile home park: 41.2%	Storage tanks: 56.2% Intersection: 43.8%	Medium residential: 100%	Sparse residential: 100%	Sparse residential: 58.8% Buildings: 41.2%	Dense residential: 100%
D	Parking lot: 100%	Medium residential: 56.3% Dense residential: 43.7%	Medium residential: 54.2% Dense residential: 45.8%	Forest: 57.8% River: 42.2%	Forest: 100%	Sparse residential: 100%

Fig. 12. Classification result for satellite sensor image 2

6. Conclusion

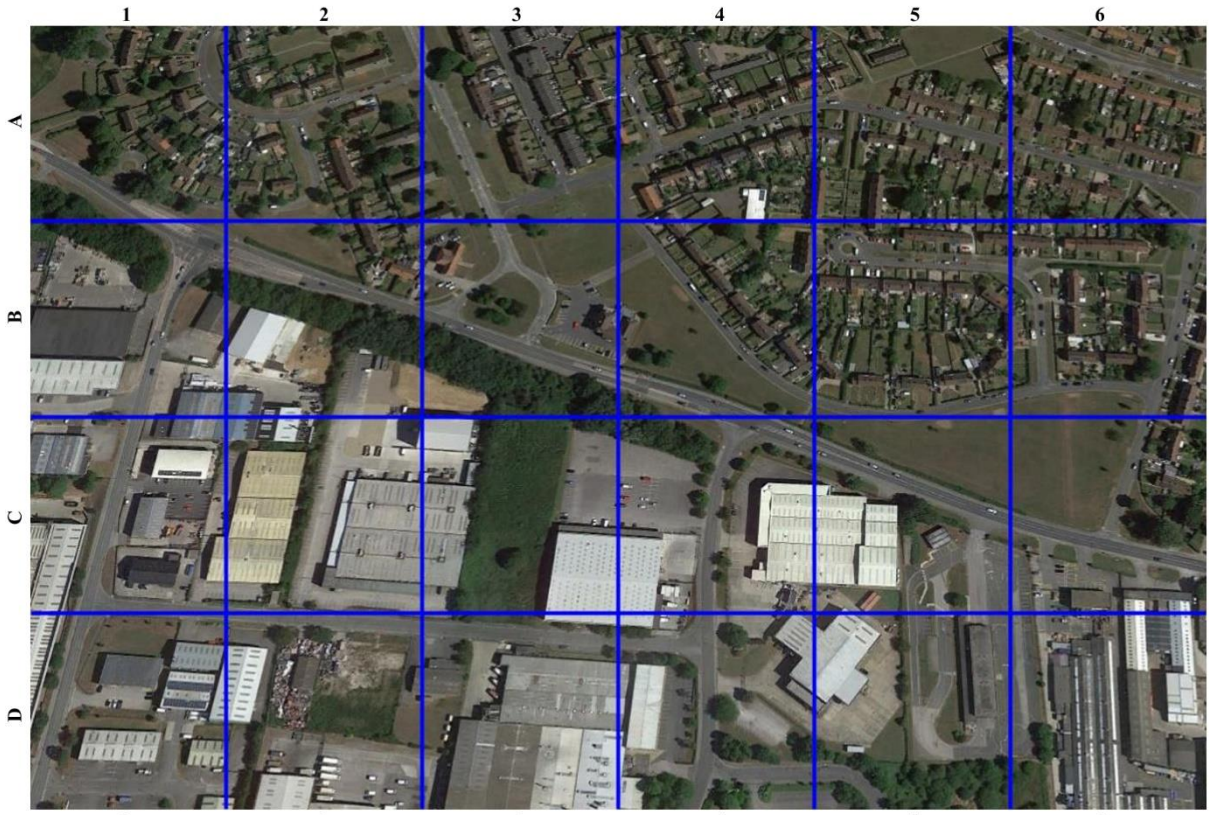
In this paper, a novel semi-supervised ensemble framework (STHPEF) was proposed for remote sensing scene classification. Using STHP+ as its base classifiers and multiple different feature descriptors, STHPEF is capable of self-learning a multi-layered prototype-based ensemble system structure from both labelled and unlabelled images at multiple levels of granularity. Thanks to its unique pseudo-labelling and cross-checking mechanisms, STHPEF constructs a highly accurate ensemble predictive model with a much reduced requirement for human effort. Importantly, STHPEF performs self-learning on a chunk-by-chunk basis, enabling the proposed ensemble model to self-adapt to unfamiliar patterns of new images. Extensive numerical experiments demonstrated the effectiveness of the proposed ensemble framework.

There are several directions for future research. First, in the numerical examples, three DCNN models (ResNet50, DenseNet121 and InceptionV3) were employed for feature extraction. The main reason for employing the three DCNNs is because they are the most popular models for remote sensing scene classification. However, it is worth investigating the optimal combination of DCNNs through comprehensive comparison. Second, the employed DCNNs were fine-tuned on the NWP dataset during the experiments and only basic image augmentation techniques were used during the experiments. This can be further expanded using different levels of scale, illumination and resolution, or through generative adversarial networks for model fine-turning. Third, the chunk-by-chunk semi-supervised learning mechanism gives STHP+ and its ensemble STHPEF the opportunity to handle data streams and self-adapt to new unfamiliar data patterns, which opens up vast potential for real-world application scenarios, such as using time-series streaming data. Finally, a better semi-supervised learning mechanism with self-adaptive parameters can also be adopted for STHPEF. Currently, it requires four parameters predefined by users. Although these externally controlled parameters can be determined without prior knowledge, they may impose some challenges to users with no or little knowledge or relevant expertise.



	1	2	3	4	5	6
A	River: 100%	Sparse residential: 100%	Medium residential: 52.5% River: 47.5%	Medium residential: 100%	Buildings: 54.8% Dense residential: 45.2%	Mobile home park: 100%
B	Harbour: 100%	Buildings: 100%	Dense residential: 50.9% Mobile home park: 49.1%	Buildings: 100%	Buildings: 100%	Dense residential: 57.8% Buildings: 42.2%
C	Harbour: 100%	Medium residential: 35.4% Dense residential: 34.6% Buildings: 30%	Mobile home park: 52.4% Dense residential: 47.6%	Golf course: 100%	Medium residential: 100%	Medium residential: 100%
D	River: 100%	River: 100%	Dense residential: 100%	Medium residential: 100%	Medium residential: 100%	Dense residential: 56.1% Medium residential: 43.9%

Fig. 13. Classification result for satellite sensor image 3



	1	2	3	4	5	6
A	Medium residential: 100%	Mobile home park: 53.4% Dense residential: 46.6%	Freeway: 100%	Dense residential: 56.9% Mobile home park: 43.1%	Dense residential: 54.4% Mobile home park: 45.6%	Dense residential: 56.3% Mobile home park: 43.7%
B	Overpass: 100%	Freeway: 100%	Freeway: 100%	Freeway: 100%	Dense residential: 100%	Medium residential: 56.5% Dense residential: 43.5%
C	Buildings: 100%	Buildings: 100%	Buildings: 100%	Buildings: 100%	Freeway: 100%	Freeway: 100%
D	Medium residential: 50.1% Buildings: 49.9%	Buildings: 100%	Buildings: 100%	Intersection: 39.1% Medium residential: 37.8% Dense residential: 23.1%	Storage tanks: 100%	Buildings: 100%

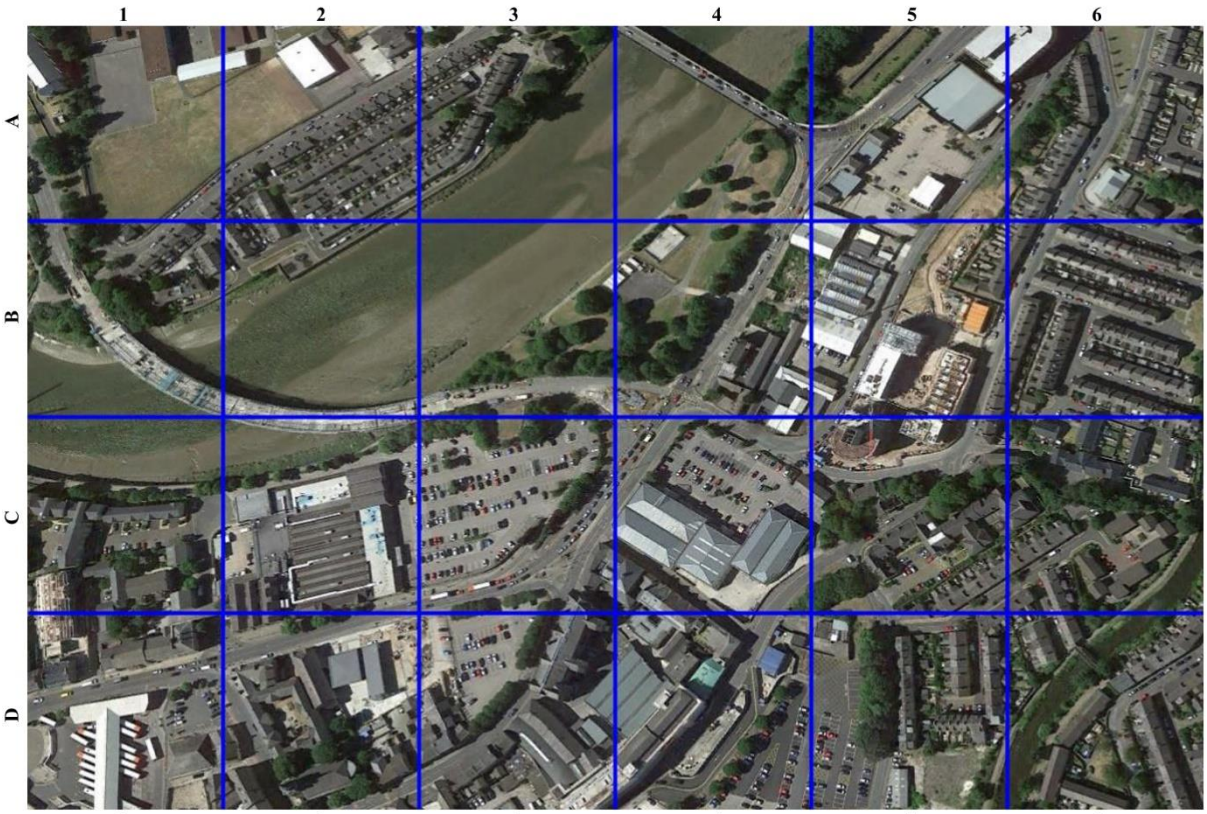
Fig. 14. Classification result for satellite sensor image 4

Appendix

A. Sensitivity Analysis

The proposed STHPEF system has five externally controlled parameters, namely, *i)* θ_0, the maximum angle between any two similar samples at the first level of granularity; *ii)* H, the number of layers of the hierarchies; *iii)* γ_0, the threshold used by Condition 3 for pseudo-labelling; *iv)* W_0, the number of nearest prototypes considered for pseudo-labelling and decision-making; *v)* Q, the size of data chunks. These parameters are investigated via a sensitivity analysis of the proposed ensemble system.

First, the influences of θ_0 and H are examined. The WHU and UCM datasets are used here, with 10% used as the labelled training set and the remaining data as the unlabelled training set. The values of θ_0 and H vary from $\frac{\pi}{6}$ to $\frac{\pi}{2}$ and 1 to 4, respectively. Other parameters are set as: $\gamma_0 = 1.1$; $W_0 = 4$; $Q = 500$. The classification accuracy rates of STHPEF on the unlabelled image sets after self-training are reported in Table 10. To understand the influences of the two parameters on system complexity, the average number of prototypes per layer per base classifier is reported in the same table.

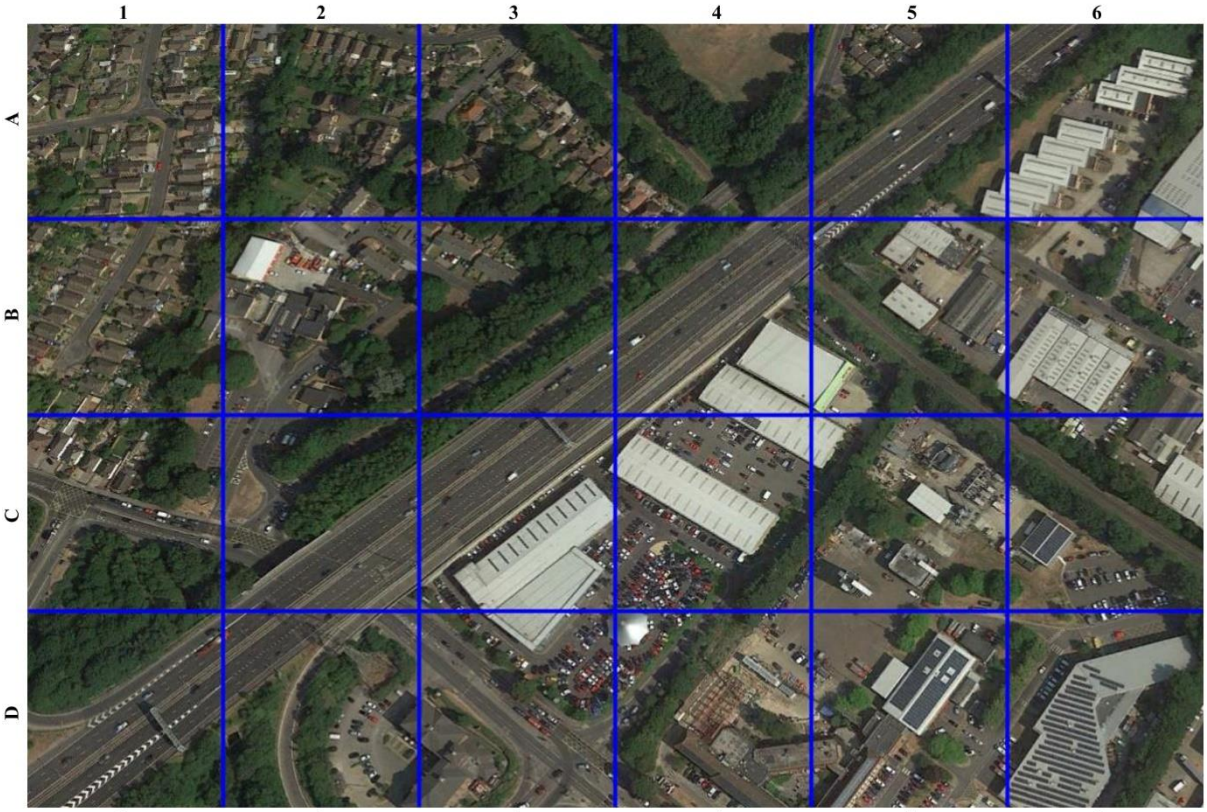


	1	2	3	4	5	6
A	Baseball diamond: 100%	Buildings: 58.4% Freeway: 41.6%	River: 100%	Overpass: 100%	Buildings: 41.8% Medium residential: 31.5% Chaparral: 26.7%	Buildings: 100%
B	River: 100%	Beach: 100%	River: 100%	Buildings: 50.2% Dense residential: 49.8%	Buildings: 100%	Buildings: 100%
C	Buildings: 100%	Buildings: 100%	Parking lot: 100%	Buildings: 100%	Dense residential: 54.6% Medium residential: 45.4%	Medium residential: 50.7% Dense residential: 49.3%
D	Intersection: 31% Parking lot: 24.1% Mobile home park: 23.7% Medium residential: 21.2%	Dense residential: 51.9% Buildings: 48.1%	Buildings: 100%	Buildings: 100%	Buildings: 58.8% Dense residential: 41.2%	Medium residential: 30.6% Dense residential: 26.6% Freeway: 21.7% Buildings: 21.1%

Fig. 15. Classification result for satellite sensor image 5

From Table 10, both θ_0 and H influence the performance of STHPEF by determining the minimum and maximum levels of granularity it can achieve for data partitioning and prototype identification. With finer data partitioning, STHPEF can identify more prototypes from training data and learn more details. In particular, a smaller θ_0 enables STHPEF to identify more apex prototypes from training data and achieve finer data partitioning in the successive layers. A greater H enables its base classifier STHP+ to partition the data with a high level of granularity at its bottom layer. In this paper, $\theta_0 = \frac{\pi}{3}$ and $H = 3$ are used for the rest of the numerical experiments.

Second, the influence of γ_0 on system performance was investigated. The value of γ_0 varies from 1.01 and 1.35, and other parameters were set as $\theta_0 = \frac{\pi}{3}$; $H = 3$; $W_0 = 4$; $Q = 500$. The same experimental protocol was applied here with the results reported in Table 11. A greater γ_0 makes STHP+ more cautious and selective during the pseudo-labelling process, leading to greater accuracy thanks to high pseudo-labelling precision. However, the performance of STHPEF decreases if γ_0 is set to be too large since fewer unlabelled samples will be used to expand the knowledge base of the system, leading to a waste of important information from unlabelled images. From Table 11, the recommended value range of γ_0 is $[1.05,1.15]$. Therefore, $\gamma_0 = 1.1$ was used as the average for the numerical experiments.



	1	2	3	4	5	6
A	Mobile home park: 28.9% Dense residential: 28.1% Forest: 21.8% Medium residential: 21.1%	Medium residential: 100%	Forest: 53.9% Medium residential: 46.1%	River: 58.4% Forest: 41.6%	Freeway: 100%	Buildings: 100%
B	Dense residential: 51.6% Medium residential: 48.4%	Medium residential: 57.3% Dense residential: 42.7%	Freeway: 100%	Freeway: 100%	Buildings: 100%	Buildings: 100%
C	Intersection: 100%	Freeway: 100%	Freeway: 100%	Parking lot: 100%	Mobile home park: 56.5% Medium residential: 43.5%	Freeway: 100%
D	Freeway: 100%	Freeway: 100%	Freeway: 100%	Mobile home park: 100%	Buildings: 33.6% Overpass: 28.4% Freeway: 19.5% Chaparral: 18.5%	Buildings: 100%

Fig. 16. Classification result for satellite sensor image 6

Third, the influence of W_0 on the unlabelled images and system complexity was investigated in terms of classification precision. Here, the value of W_0 varies from 1 to 8, and the rest of the externally controlled parameters were set to be: $\theta_0 = \frac{\pi}{3}$; $H = 3$; $\gamma_0 = 1.1$; $Q = 500$. Other experimental settings remain the same. From Table 12, a greater W_0 helps STHPEF to identify more prototypes from the data. This is because the pseudo-labelling process is mostly based on the confidence scores, which are calculated based on the similarities between unlabelled samples and the prototypes. A greater W_0 means that the system will consider the fitness of the unlabelled samples in multiple local patterns of data distribution represented by prototypes, which increases the robustness of the pseudo-labelling processes effectively. Nevertheless, if W_0 is too large, the precision of pseudo-labelling will decrease since too many irrelevant local patterns spatially distant from the input samples are taken into account. The recommended values of W_0 are 3, 4 and 5. $W_0 = 4$ is used hereafter.

Table 10. Impact of θ_0 and H on system performance ($\gamma_0 = 1.1$; $W_0 = 4$; $Q = 500$)

Dataset	H	Meas.	θ_0			
			$\frac{\pi}{6}$	$\frac{\pi}{4}$	$\frac{\pi}{3}$	$\frac{\pi}{2}$
WHU	1	Acc	0.8340	0.7462	0.9573	0.9599
		M_1	4.6947	1.4175	1.0053	1.0000
	2	Acc	0.9766	0.9771	0.9599	0.8818
		M_1	10.1544	1.4509	1.0105	1.0000
		M_2	40.6123	24.6825	8.5719	1.4281
	3	Acc	0.9800	0.9808	0.9808	0.9774
		M_1	10.0737	1.4649	1.0053	1.0000
		M_2	40.3544	24.7000	10.3860	1.4561
		M_3	44.2088	43.8123	40.9912	24.8649
	4	Acc	0.9786	0.9800	0.9803	0.9811
		M_1	9.9754	1.4772	1.0053	1.0000
		M_2	40.1842	24.6544	10.2789	1.4649
		M_3	44.0456	43.7842	40.7825	24.8579
		M_4	44.2544	44.6386	44.6509	43.9842
UCM	1	Acc	0.9204	0.5341	0.9024	0.9182
		M_1	10.0508	2.0095	1.0175	1.0000
	2	Acc	0.9415	0.9432	0.9403	0.7704
		M_1	17.0032	2.0905	1.0175	1.0000
		M_2	66.2444	38.6143	15.7540	2.0397
	3	Acc	0.9425	0.9433	0.9433	0.9454
		M_1	16.8460	2.0984	1.0175	1.0000
		M_2	66.0317	38.3254	17.1159	2.0968
		M_3	82.4222	76.5730	66.4540	38.5905
	4	Acc	0.9421	0.9434	0.9423	0.9449
		M_1	16.7238	2.1175	1.0175	1.0000
		M_2	65.5984	38.3587	17.0032	2.1111
		M_3	82.0032	76.5651	66.3159	38.6000
		M_4	82.6238	82.6651	82.7302	76.8238

Table 11. Impact of γ_0 on system performance ($\theta_0 = \frac{\pi}{3}$; $H = 3$; $W_0 = 4$; $Q = 500$)

Dataset	Meas.	γ_0							
		1.01	1.05	1.10	1.15	1.20	1.25	1.30	1.35
WHU	Acc	0.9808	0.9810	0.9808	0.9810	0.9804	0.9791	0.9786	0.9756
	M_1	1.0053	1.0053	1.0053	1.0053	1.0053	1.0053	1.0053	1.0053
	M_2	10.9772	10.7211	10.3860	10.0289	9.6895	9.3667	9.0807	8.7965
	M_3	42.0035	41.5368	40.9912	40.3579	39.5982	39.0158	38.4070	37.8333
UCM	Acc	0.9453	0.9429	0.9433	0.9440	0.9413	0.9408	0.9411	0.9411
	M_1	1.0175	1.0175	1.0175	1.0175	1.0175	1.0175	1.0175	1.0175
	M_2	17.9968	17.5683	17.1159	16.6667	16.2317	15.7317	15.2111	14.6857
	M_3	68.6397	67.5048	66.4540	65.4127	64.3111	63.1651	61.8968	60.6095

Finally, the influence of Q on system performance was investigated. The value of Q varies from 50 and 700, and other parameters were set as $\theta_0 = \frac{\pi}{3}$; $H = 3$; $\gamma_0 = 1.1$; $W_0 = 4$. Table 13 shows that a larger chunk size allows STHPEF to identify more prototypes from data because pseudo-labelling becomes more robust and accurate with more unlabelled samples available. Nevertheless, the ensemble system becomes less computational efficient if Q is too large since it consumes more time and system memory to process a huge chunk of images. The recommended value range of Q is $[400, 600]$. $Q = 500$ is used here for the rest of the experiments.

Table 12. Impact of W_0 on system performance ($\theta_0 = \frac{\pi}{3}; H = 3; \gamma_0 = 1.1; Q = 500$)

Dataset	Meas.	W_0							
		1	2	3	4	5	6	7	8
WHU	Acc	0.9677	0.9797	0.9812	0.9808	0.9797	0.9791	0.9788	0.9787
	M_1	1.0053	1.0053	1.0053	1.0053	1.0053	1.0053	1.0053	1.0053
	M_2	7.4053	9.4298	10.1246	10.3860	10.4789	10.5965	10.6333	10.5737
	M_3	34.6105	39.0158	40.4509	40.9912	41.1789	41.2860	41.3000	41.2825
UCM	Acc	0.9406	0.9440	0.9441	0.9433	0.9430	0.9421	0.9411	0.9405
	M_1	1.0175	1.0175	1.0175	1.0175	1.0175	1.0175	1.0175	1.0175
	M_2	12.1286	15.4429	16.6619	17.1159	17.3159	17.4587	17.6095	17.6508
	M_3	53.3857	62.0651	65.5540	66.4540	66.9032	67.2556	67.5206	67.5968

Table 13. Impact of Q on system performance ($\theta_0 = \frac{\pi}{3}; H = 3; \gamma_0 = 1.1; W_0 = 4$)

Dataset	Meas.	Q							
		50	100	200	300	400	500	600	700
WHU	Acc	0.9692	0.9711	0.9736	0.9793	0.9807	0.9808	0.9806	0.9803
	M_1	1.0053	1.0053	1.0053	1.0053	1.0053	1.0053	1.0053	1.0053
	M_2	7.9491	9.1018	9.8175	10.1947	10.1965	10.3860	10.4667	10.5632
	M_3	27.2298	34.0175	37.9193	39.7368	40.0105	40.9912	41.0456	41.3053
UCM	Acc	0.9373	0.9419	0.9422	0.9426	0.9423	0.9433	0.9443	0.9440
	M_1	1.0175	1.0175	1.0175	1.0175	1.0175	1.0175	1.0175	1.0175
	M_2	13.4349	15.2683	16.2429	16.5667	16.9667	17.1159	17.1889	17.3762
	M_3	44.4683	54.9810	61.5238	63.8159	65.4762	66.4540	66.6698	67.6238

B. Ablation Analysis

An ablation analysis was performed to test the effectiveness of the proposed ensemble framework. Here, the WHU, UCM and RSS datasets were used as numerical examples. For the three datasets, two different split ratios between labelled and unlabelled images were considered, namely 1:19 and 1:9.

The performance of STHPEF was compared with its single-model base learner (STHP+) in terms of classification accuracy on unlabelled images. In this experiment, STHP+ employs the same pre-processing module as STHPEF and uses one of the three high-level feature descriptors. The prediction accuracies of STHPEF and STHP+ on the three datasets with different split ratios are reported in Table 14, and the respective average accuracies across the experiment are presented in Fig. 17 for visual clarity. From Table 14 and Fig. 17, the proposed ensemble framework enhances the prediction accuracy of the base classifiers greatly. For example, using a single feature descriptor, and with 5% images only as labelled ones, the accuracies of the most accurate STHP+ on three datasets are 89.66%, 88.31% and 85.19%. With the proposed ensemble framework, the classification accuracy increased to 96.35%, 92.76% and 88.73%, respectively, which is 7.46%, 5.04% and 4.27% higher than the best-performing single models.

Table 14. Performance comparison between STHPEF and STHP+

Algorithm	Feature Descriptor	Split Ratio	WHU	UCM	RSS
STHP+	ResNet50	1:19	0.8790	0.8825	0.8519
	DenseNet121		0.8966	0.8806	0.8403
	InceptionV3		0.8960	0.8831	0.8351
STHPEF			0.9635	0.9276	0.8873
STHP+	ResNet50	1:9	0.9133	0.9125	0.8800
	DenseNet121		0.9235	0.9088	0.8600
	InceptionV3		0.9170	0.9043	0.8573
STHPEF			0.9808	0.9433	0.9044

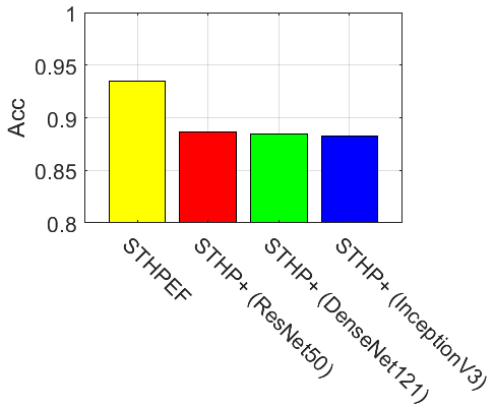


Fig. 17. Average prediction accuracies of STHPEF and STHP+

Four variations of STHPEF are considered: 1) $STHPEF_1$: the same ensemble framework without semi-supervised learning; 2) $STHPEF_2$: the same ensemble framework without cross-checking; 3) $STHPEF_3$: the same ensemble framework with the original STHP as base classifier; 4) $STHPEF_4$: the same ensemble framework with the original STHP as base classifier and no cross-checking. The accuracies of the four versions as well as the original STHPEF obtained on the three datasets under the two different split ratios are reported in Table 15. Correspondingly, the average accuracies of the original STHPEF and its four versions across the experiment are presented in Fig. 18. Comparing between STHPEF and $STHPEF_1$, the proposed semi-supervised learning mechanism can effectively enhance the overall classification precision of the proposed ensemble framework because the unlabelled images are also involved in producing the predictive ensemble model. In addition, by comparing between STHPEF, $STHPEF_1$ and $STHPEF_2$, without cross-checking, $STHPEF_2$ performs less accurately if the available labelled training images are insufficient for precise pseudo-labelling. However, with crossing-checking, STHPEF is able to classify more accurately. Finally, by comparing between STHPEF and $STHPEF_3$, $STHPEF_2$ and $STHPEF_4$, one can conclude that STHP+ plays a better role in the proposed ensemble framework than the original STHP system thanks to the more robust "multi-nearest prototypes" principle for pseudo-labelling and decision-making.

Table 15. Performance comparison between STHPEF and its variations as ablation analysis

Algorithm	Split Ratio	WHU	UCM	RSS
STHPEF	1:19	0.9635	0.9276	0.8873
$STHPEF_1$		0.9590	0.9270	0.8848
$STHPEF_2$		0.9425	0.9173	0.8809
$STHPEF_3$		0.9456	0.9070	0.8843
$STHPEF_4$		0.9502	0.9222	0.8862
STHPEF	1:9	0.9808	0.9433	0.9044
$STHPEF_1$		0.9600	0.9360	0.8987
$STHPEF_2$		0.9736	0.9428	0.9000
$STHPEF_3$		0.9526	0.9358	0.8961
$STHPEF_4$		0.9612	0.9359	0.8974

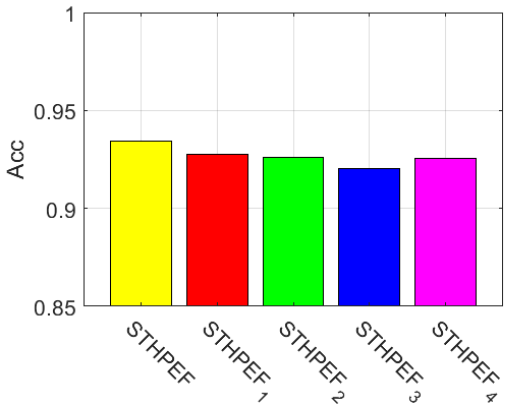


Fig. 18. Average prediction accuracies of STHPEF and its four versions

References

[1] W. Tong, W. Chen, W. Han, X. Li, and L. Wang, "Channel-attention-based DenseNet network for remote sensing image scene classification," *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, vol. 13, pp. 4121–4132, 2020.

[2] G. Xia *et al.*, "AID: a benchmark dataset for performance evaluation of aerial scene classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 7, pp. 3965–3981, 2017.

[3] S. Salcedo-Sanz *et al.*, "Machine learning information fusion in Earth observation: A comprehensive review of methods, applications and data sources," *Inf. Fusion*, vol. 63, pp. 256–272, 2020.

[4] G. Cheng, J. Han, and X. Lu, "Remote sensing image scene classification: benchmark and state of the art," *Proc. IEEE*, vol. 105, no. 10, pp. 1865–1883, 2017.

[5] Y. Li, H. Zhang, X. Xue, Y. Jiang, and Q. Shen, "Deep learning for remote sensing image classification: a survey," *WIREs Data Min. Knowl. Discov.*, vol. 8, no. 6, p. e1264, 2018.

[6] H. Zhang, Y. Li, Y. Zhang, and Q. Shen, "Spectral-spatial classification of hyperspectral imagery using a dual-channel convolutional neural network," *Remote Sens. Lett.*, vol. 8, no. 5, pp. 438–447, 2017.

[7] H. Zhu, W. Ma, L. Li, L. Jiao, S. Yang, and B. Hou, "A dual–branch attention fusion deep network for multiresolution remote sensing image classification," *Inf. Fusion*, vol. 58, pp. 116–131, 2020.

[8] D. Lin, K. Fu, Y. Wang, G. Xu, and X. Sun, "MARTA GANs: unsupervised representation learning for remote sensing image classification," *IEEE Geosci. Remote Sens. Lett.*, vol. 14, no. 11, pp. 2092–2096, 2017.

[9] X. Gu, P. Angelov, C. Zhang, and P. M. Atkinson, "A semi-supervised deep rule-based approach for complex satellite sensor image analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, DOI: 10.1109/TPAMI.2020.3048268, 2020.

[10] J. E. van Engelen and H. H. Hoos, "A survey on semi-supervised learning," *Mach. Learn.*, vol. 109, no. 2, pp. 373–440, 2020.

[11] L. Gómez-Chova, G. Camps-Valls, J. Munoz-Mari, and J. Calpe, "Semisupervised image classification with Laplacian support vector machines," *IEEE Geosci. Remote Sens. Lett.*, vol. 5, no. 3, pp. 336–340, 2008.

[12] L. Bruzzone, M. Chi, and M. Marconcini, "A novel transductive SVM for semisupervised classification of remote-sensing images," *IEEE Trans. Geosci. Remote Sens.*, vol. 44, no. 11, pp. 3363–3373, 2006.

[13] N. S. Kothari, S. K. Meher, and G. Panda, "Improved spatial information based semisupervised classification of remote sensing images," *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, vol. 13, pp. 329–340, 2020.

[14] F. Ratle, G. Camps-Valls, and J. Weston, "Semisupervised neural networks for efficient hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 48, no. 5, pp. 2271–2282, 2010.

[15] L. Bruzzone, R. Cossu, and G. Vernazza, "Combining parametric and non-parametric algorithms for a partially unsupervised classification of multitemporal remote-sensing images," *Inf. Fusion*, vol. 3, no. 4, pp. 289–297, 2002.

[16] J. Liu *et al.*, "Urban green plastic cover mapping based on VHR remote sensing images and a deep semi-supervised learning framework," *ISPRS Int. J. Geo-Information*, vol. 9, no. 9, p. 527, 2020.

[17] S. Wang *et al.*, "Semi-supervised PolSAR image classification based on improved tri-training with a minimum spanning tree," *IEEE Trans. Geosci. Remote Sens.*, vol. 58, no. 12, pp. 8583–8597, 2020.

[18] X. Gu, "A self-training hierarchical prototype-based approach for semi-supervised classification," *Inf. Sci. (Ny).*, vol. 535, pp. 204–224, 2020.

[19] A. Oliva and A. Torralba, "Modeling the shape of the scene: a holistic representation of the spatial envelope," *Int. J. Comput. Vis.*, vol. 42, no. 3, pp. 145–175, 2001.

[20] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, 2004.

[21] T. Ojala, M. Pietikäinen, and T. Mäenpää, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 971–987, 2002.

[22] M. J. Swain and D. H. Ballard, "Color indexing," *Int. J. Comput. Vis.*, vol. 7, no. 1, pp. 11–32, 1991.

[23] Y. Yang and S. Newsam, "Bag-of-visual-words and spatial extensions for land-use classification," in *International Conference on Advances in Geographic Information Systems*, 2010, pp. 270–279.

[24] A. M. Cheriyadat, "Unsupervised feature learning for aerial scene classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 52, no. 1, pp. 439–451, 2014.

[25] J. Fan, T. Chen, and S. Lu, "Unsupervised feature learning for land-use scene recognition," *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 4, pp. 2250–2261, 2017.

[26] S. Gao, Y. Zhang, K. Jia, J. Lu, and Y. Zhang, "Single sample face recognition via learning deep supervised auto-encoders," *IEEE Trans. Inf. Forensics Secur.*, vol. 6013, no. c, pp. 1–1, 2015.

[27] F. Li, R. Feng, W. Han, and L. Wang, "An augmentation attention mechanism for high-spatial-resolution remote sensing image scene classification," *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, vol. 13, pp. 3862–3878, 2020.

[28] Y. Yu, X. Li, and F. Liu, "Attention GANs: unsupervised deep feature learning for aerial scene classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 58, no. 1, pp. 519–531, 2020.

[29] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nat. Methods*, vol. 13, no. 1, pp. 35–35, 2015.

[30] C. Zhang, G. Li, and S. Du, "Multi-scale dense networks for hyperspectral remote sensing image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 11, pp. 9201–9222, 2019.

[31] X. Lu, H. Sun, and X. Zheng, "A feature aggregation convolutional neural network for remote sensing scene classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 10, pp. 7894–7906, 2019.

[32] X. Liu, Y. Zhou, J. Zhao, R. Yao, B. Liu, and Y. Zheng, "Siamese convolutional neural networks for remote sensing scene classification," *IEEE Geosci. Remote Sens. Lett.*, vol. 16, no. 8, pp. 1200–1204, 2019.

[33] B. Zhang, Y. Zhang, and S. Wang, "A lightweight and discriminative model for remote sensing scene classification with multidilation pooling module," *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, vol. 12, no. 8, pp. 2636–2653, 2019.

[34] B. Cui, X. Chen, and Y. Lu, "Semantic segmentation of remote sensing images using transfer learning and deep convolutional neural network with dense connection," *IEEE Access*, vol. 8, pp. 116744–116755, 2020.

[35] O. Russakovsky *et al.*, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, 2015.

[36] A. Fotso Kamga Guy, T. Akram, B. Laurent, S. Rameez, M. Mbom, and N. Muhammad, "A deep heterogeneous feature fusion approach for automatic land-use classification," *Inf. Sci. (Ny).*, vol. 467, pp. 199–218, 2018.

[37] B. Zhao, B. Huang, and Y. Zhong, "Transfer learning with fully pretrained deep convolution networks for land-use classification," *IEEE Geosci. Remote Sens. Lett.*, vol. 14, no. 9, pp. 1436–1440, 2017.

[38] H. Alhichri, A. S. Alswayed, Y. Bazi, N. Ammour, and N. A. Alajlan, "Classification of remote sensing images using EfficientNet-B3 CNN model with attention," *IEEE Access*, vol. 9, pp. 14078–14094, 2021.

[39] Q. Wang, S. Liu, and J. Chanussot, "Scene classification with recurrent attention of VHR remote sensing images," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 2, pp. 1155–1167, 2019.

[40] R. M. Anwer, F. S. Khan, J. van de Weijer, M. Molinier, and J. Laaksonen, "Binary patterns encoded convolutional neural networks for texture recognition and remote sensing scene classification," *ISPRS J. Photogramm. Remote Sens.*, vol. 138, pp. 74–85, 2018.

[41] F. Sun *et al.*, "An impartial semi-supervised learning strategy for imbalanced classification on VHR images," *Sensors (Switzerland)*, vol. 20, no. 22, pp. 1–20, 2020.

[42] D. Hong, N. Yokoya, N. Ge, J. Chanussot, and X. X. Zhu, "Learnable manifold alignment (LeMA): a semi-supervised cross-modality learning framework for land cover and land use classification," *ISPRS J. Photogramm. Remote Sens.*, vol. 147, pp. 193–205, 2019.

[43] F. Zhao, M. Tian, W. Xie, and H. Liu, "A new parallel dual-channel fully convolutional network via semi-supervised FCM for polsar image classification," *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, vol. 13, pp. 4493–4505, 2020.

[44] S. Ren and F. Zhou, "Semi-supervised classification for PolSAR data with multi-scale evolving weighted graph convolutional network," *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, vol. 14, pp. 2911–2927, 2021.

[45] X. Zhu, "Semi-supervised learning literature survey," 2008.

[46] J. Thorsten, "Transductive inference for text classification using support vector machines," in *International conference on Machine learning*, 1999, pp. 200–209.

[47] Y. F. Li and Z. H. Zhou, "Towards making unlabeled data never hurt," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 1, pp. 175–188, 2015.

[48] D.-H. Lee, "Pseudo-label: the simple and efficient semi-supervised learning method for deep neural networks," in *Workshop on Challenges in Representation Learning, ICML*, 2013, p. 2.

[49] Z.-H. Zhou and M. Li, "Semi-supervised regression with co-training," in *International Joint Conference on Artificial Intelligence*, 2005, pp. 908–913.

[50] P. K. Mallapragada, R. Jin, A. K. Jain, and Y. Liu, "SemiBoost: boosting for semi-supervised learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 11, pp. 2000–2014, 2008.

[51] M. Belkin, P. Niyogi, and V. Sindhwani, "Manifold regularization: a geometric framework for learning from labeled and unlabeled examples," *J. Mach. Learn. Res.*, vol. 7, no. 2006, pp. 2399–2434, 2006.

[52] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," in *Adv. Neural. Inform. Process Syst*, 2004, pp. 321–328.

[53] W. Liu, J. He, and S.-F. Chang, "Large graph construction for scalable semi-supervised learning," in *International Conference on Machine Learning*, 2010, pp. 679–689.

[54] Z. H. Zhou and M. Li, "Tri-training: exploiting unlabeled data using three classifiers," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 11, pp. 1529–1541, 2005.

[55] Y. Yu and F. Liu, "A two-stream deep fusion framework for high-resolution aerial scene classification," *Comput. Intell. Neurosci.*, p. 8639367, 2018.

[56] A. Okabe, B. Boots, K. Sugihara, and S. N. Chiu, *Spatial tessellations: concepts and applications of Voronoi diagrams*, 2nd ed. Chichester, England: John Wiley & Sons., 1999.

[57] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2005, pp. 886–893.

[58] M. I. Lakhal, H. Çevikalp, S. Escalera, and F. Ofli, "Recurrent neural networks for remote sensing image classification," *IET Comput. Vis.*, vol. 12, no. 7, pp. 1040–1045, 2018.

[59] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances In Neural Information Processing Systems*, 2012, pp. 1097–1105.

[60] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[61] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4700–4708.

[62] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2818–2826.

[63] R. P. de Lima and K. Marfurt, "Convolutional neural network for remote-sensing scene classification: transfer learning analysis," *Remote Sens.*, vol. 12, no. 1, p. 86, 2020.

[64] W. Li *et al.*, "Classification of high-spatial-resolution remote sensing scenes method using transfer learning and deep convolutional neural network," *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, vol. 13, pp. 1986–1995, 2020.

[65] G. J. Scott, K. C. Hagan, R. A. Marcum, J. A. Hurt, D. T. Anderson, and C. H. Davis, "Enhanced fusion of deep neural networks for classification of benchmark high-resolution image data sets," *IEEE Geosci. Remote Sens. Lett.*, vol. 15, no. 9, pp. 1451–1455, 2018.

[66] F. Hu, G. S. Xia, J. Hu, and L. Zhang, "Transferring deep convolutional neural networks for the scene classification of high-resolution remote sensing imagery," *Remote Sens.*, vol. 7, no. 11, pp. 14680–14707, 2015.

[67] D. P. Kingma and J. L. Ba, "Adam: a method for stochastic optimization," in *International Conference on Learning Representations*, 2015, pp. 1–15.

[68] A. Bahri, S. Ghofrani Majelan, S. Mohammadi, M. Noori, and K. Mohammadi, "Remote sensing image classification via improved cross-entropy loss and transfer learning strategy based on deep convolutional neural networks," *IEEE Geosci. Remote Sens. Lett.*, vol. 17, no. 6, pp. 1087–1091, 2020.

[69] G. Xia, W. Yang, J. Delon, Y. Gousseau, H. Sun, and H. Maitre, "Structural High-resolution Satellite image indexing," in *ISPRS, TC VII Symposium Part A: 100 Years ISPRS—Advancing Remote Sensing Science*, 2010, pp. 298–303.

[70] Q. Zou, L. Ni, T. Zhang, and Q. Wang, "Deep learning based feature selection for remote sensing scene classification," *IEEE Geosci. Remote Sens. Lett.*, vol. 12, no. 11, pp. 2321–2325, 2015.

[71] W. Zhou, S. Newsam, C. Li, and Z. Shao, "PatternNet: a benchmark dataset for performance evaluation of remote sensing image retrieval," *ISPRS J. Photogramm. Remote Sens.*, vol. 145, pp. 197–209, 2018.

[72] H. Li *et al.*, "RSI-CB: a large-scale remote sensing image classification benchmark using crowdsourced data," *Sensors*, vol. 20, no. 6, pp. 28–32, 2020.

[73] M. Wang, W. Fu, S. Hao, D. Tao, and X. Wu, "Scalable semi-supervised learning by efficient anchor graph regularization," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 7, pp. 1864–1877, 2016.

[74] R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni, "Green ai," *arXiv Prepr. arXiv1907.10597*, 2019.

[75] H. Sun, S. Li, X. Zheng, and X. Lu, "Remote sensing scene classification by gated bidirectional network," *IEEE Trans. Geosci. Remote Sens.*, vol. 58, no. 1, pp. 82–96, 2020.

[76] N. Liu, T. Celik, and H. Li, "MSNet: a multiple supervision network for remote sensing scene classification," *IEEE Geosci. Remote Sens. Lett.*, DOI: 10.1109/LGRS.2020.3043020, 2020.

[77] X. Bian, C. Chen, L. Tian, and Q. Du, "Fusing local and global features for high-resolution scene

classification," *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, vol. 10, no. 6, pp. 2889–2901, 2017.

[78] Y. Guo, J. Ji, X. Lu, H. Huo, T. Fang, and D. Li, "Global-local attention network for aerial scene classification," *IEEE Access*, vol. 7, pp. 67200–67212, 2019.