Iterative Random Training Sampling Spectral Spatial Classification for Hyperspectral Images

Chein-I Chang, *Life Fellow, IEEE*, Kenneth Yeonkong Ma, Chia-Chen Liang, Yi-Mei Kuo, Shuhan Chen, and Shengwei Zhong

***Abstract*—Hyperspectral image classification (HSIC) has generated considerable interests over the past years. However, one of challenging issues arising in HSIC is inconsistent classification, which is mainly caused by random training sampling (RTS) of selecting training data. This is because a different set of training samples may produce a different classification result. A general approach to addressing this problem is the so-called *K*-fold method which implements RTS *K* times and takes the average of overall accuracy with respect to standard deviation to describe a confidence level of classification performance. To deal with this issue, this article develops an iterative RTS (IRTS) method as an alternative to the *K*-fold method to reduce the uncertainty caused by RTS. Its idea is to add the spatial filtered classification maps to the image cube that is currently being processed via feedback loops to augment image cubes iteratively. Then, the training samples will be reselected randomly from the new augmented image cubes iteration-by-iteration. As a result, the training samples selected from each iteration will be updated by new added spatial information captured by spatial filters implemented at the iteration. The experimental results clearly demonstrate that IRTS successfully improves classification accuracy as well as reduces inconsistency in results.**

***Index Terms*—Hyperspectral image classification (HSIC), iterative random training sampling (IRTS), *K*-fold method, random training sampling (RTS), spectral-spatial (SS).**

NOMENCLATURE

AA	Average accuracy.
CE	Classification entropy.
CSD	Class standard deviation.
CSI	Class self-information.
EPF	Edge preserving filter.
Gabor-EPF	Gabor-EPF fused filter.
GEPF	Gaussian-EPF fused filter.
HSIC	Hyperspectral image classification.
IEPF	Iterative EPF.
IRTS	Iterative random training sampling.
OA	Overall accuracy.
OCE	Overall class entropy.
OCSD	Overall class standard deviation.
PMF	Probability mass function.
RTS	Random training sampling.
SD	Standard deviation.
SE	Sample entropy.
SR	Sample ratio.
SS	Spectral-spatial.
SSD	Sample standard deviation.
SVM	Support vector machine.
C_m	*m*th class.
δ	Classifier.
M	Total number of classes.
N	Total number of data samples without including BKG samples.
$\delta_{S^{\text{training}}}$	Classifier using S^{training}.
N^{total}	Size of S^{total}.
n_m^{training}	Size of S_m^{training}.
$p_m(\mathbf{r})$	Probability of classifying the data sample $\mathbf{r}$ into the *m*th class C_m.
S_m^{training}	Set of training samples to be selected from class C_m.
S^{training}	Set of training samples.
$S^{\text{training}(l)}$	*l*th selected training sample set.
S^{total}	Collection of all possible training sample sets.
$\mathbf{\Omega}_{\text{data}}$	Data space $= \{\mathbf{r}_n\}_{n=1}^N$.
$\mathbf{\Omega}^{(l)}$	Data cube produced at the *l*th iteration.
$\zeta(\delta_{S^{\text{training}}}(\mathbf{r}))$	Random variable defined on data space $\mathbf{\Omega}_{\text{data}}$ via a classifier $\delta_{S^{\text{training}}}(\mathbf{r})$ using training sample set S^{training} with $\mathbf{r} \in \mathbf{\Omega}_{\text{data}} = \{\mathbf{r}_n\}_{n=1}^N$.
$\text{MAXMap}_m^{(l)}$	*m*th class (C_m)-classification map produced by a fusion process (13) using $S^{\text{training}(l)}$ as training sample set.
$\text{BMAXMap}_m^{(l)}$	*m*th class (C_m)-binary map produced from $\text{MAXMap}_m^{(l)}$ by MAP.

Manuscript received April 12, 2020; revised May 25, 2020; accepted July 1, 2020. Date of publication July 9, 2020; date of current version July 22, 2020. The work of Chein-I Chang was supported by the Fundamental Research Funds for Central Universities under Grant 3132019341. *(Corresponding author: Kenneth Yeonkong Ma.)*

Chein-I Chang is with the Center for Hyperspectral Imaging in Remote Sensing, Information and Technology College, Dalian Maritime University, Dalian 116026, China, with the Remote Sensing Signal and Image Processing Laboratory, Department of Computer Science and Electrical Engineering, the University of Maryland, Baltimore, MD 21250 USA, and also with the Department of Computer Science and Information Management, Providence University, Taichung 02912, Taiwan (e-mail: cchang@umbc.edu).

Kenneth Yeonkong Ma, Chia-Chen Liang, and Yi-Mei Kuo are with the Remote Sensing Signal and Image Processing Laboratory, Department of Computer Science and Electrical Engineering, the University of Maryland, Baltimore, MD 21250 USA (e-mail: kennethma1619@gmail.com; lcjane212@gmail.com; ccgsh10923@gmail.com).

Shuhan Chen is with the Zhejiang University, Hangzhou 310058, China (e-mail: 11410057@zju.edu.cn).

Shengwei Zhong is with the School of Computer Science and Engineering, the Nanjing University of Science and Technology, Nanjing 210094, China (e-mail: zhongsw_91@foxmail.com).

Digital Object Identifier 10.1109/JSTARS.2020.3008359

I. Introduction

HYPERSPECTRAL image classification (HSIC) has received considerable interest, for instance, [1]–[7], to name just a few. One of the challenges arising in HSIC is random selection of training samples. Associated with it are two issues—how to determine a training sample size for each class due to imbalanced classes with varying sizes and how to select training samples once each class training sample size is determined. These two issues are very similar to the two issues encountered in an endmember finding problem—how to determine the number of endmembers to be found such as virtual dimensionality (VD) [8] and how to find the desired endmembers such as N-finder algorithm (N-FINDR) [9].

Regarding the first issue, a recent approach using class features was developed [10], [11] to define the significance of each class, referred to as class significance, which can be used to determine the size of training samples allocated for each class, provided that a total number of training samples are given. As shown in [11], such class significance-determined class training sample size worked more effectively and was better in terms of average accuracy (AA) and overall accuracy (OA) than sample ratio (SR) widely used in classification. Because this issue has been addressed in [10] and [11], it will not be discussed in this article. Instead, the second issue will be the main focus of this article.

As for the second issue, a classic approach, called *K*-fold method is generally used in pattern classification community, specifically medical data analysis. Since a different set of randomly selected training samples generally produces a different classification result, the final classification results produced by different sets of randomly selected training samples cannot be reproducible and are also inconsistent. To mitigate this dilemma, the *K*-fold method is generally being used for cross validation. It requires running the same classification process repeatedly *K* times using *K* different sets of randomly and independently selected training samples, and then, calculating their mean as the desired results along with the standard deviation (SD) which can be used as a confidence level to determine the effectiveness of the classification. The larger the SD, the greater the inconsistency in classification, and thus, the higher the classification discrepancy. SD is particularly worsen when only a very limited number of training samples is available to be used for experiments.

This article looks into this random training sampling (RTS) issue quite differently from an information theory point of view [12], which has never been investigated in the past. It first considers the traditional classification as a random classification problem where a classifier using a set of RTS-selected training samples is treated as a random classifier. It then defines the concept of class self-information (CSI) to measure the uncertainty of individual class accuracy caused by such a random classifier. In this case, the class entropy (CE) is further defined by the expectation of CSI over all the classes to measure the overall uncertainty of AA and OA resulting from a random classifier using RTS-selected training samples. In order to reduce CE and SD, a new concept similar to but different from that used in the *K*-fold method is particularly developed. It is called iterative RTS (IRTS) which is implemented iteratively in a novel means.

Since IRTS is an iterative process, we further assume that there are a total of n_I iterations carried out by IRTS. At each iteration, IRTS randomly selects a set of training samples and these randomly selected training samples are performed independently, iteration-by-iteration. So, it appears that IRTS performs as if it is the *K*-fold method with $K = n_I$. However, there is a crucial difference between IRTS and the *K*-fold method in the sense that the former operates on data cubes augmented n_I times by iterations as opposed to the latter which operates the same original data cube over and over again *K* times. More specifically, IRTS randomly selects a new set of training samples from an augmented data cube expanded by including additional spatial classification information obtained from the preceding iteration, whereas the *K*-fold method randomly selects training samples from the same original dataset *K* times independently. Because of that, we refer to IRTS as one-fold n_I-iteration RTS in correspondence to the *K*-fold method without iterations to indicate that one-fold n_I-iteration RTS actually runs IRTS only once through n_I iterations. There are three major key factors that distinguish one-fold n_I-iteration RTS with $K = n_I$ from the *K*-fold method. One is the hyperspectral image cube on which they operate. The *K*-fold method executes the same process on the same dataset repeatedly and independently *K* times using different sets of randomly selected training samples. By contrast, one-fold n_I-iteration RTS with $K = n_I$ randomly selects new sets of training samples iteratively from n_I augmented data cubes which are generated iteration-by-iteration via feedback loops. Another is how to determine the parameter "*K*" in the *K*-fold method and "n_I". The parameter "*K*" used by the *K*-fold method is predetermined, whereas "n_I" used in one-fold n_I-iteration RTS is automatically determined by the stopping rule implemented in the iterative process. Because the "n_I" used in one-fold RTS is generally unknown but rather determined by the stopping rule, one-fold n_I-iteration RTS indeed performs IRTS. With this interpretation, IRTS can be considered as an iterative version of a one-fold method which implements RTS on n_I augmented data cubes by n_I iterations. On the other hand, when *K*-fold IRTS is referred, it actually implements IRTS *K* times independently using *K* different sets of training samples randomly selected from the same original data cube, but each fold implemented by IRTS is a single-fold IRTS, which is indeed one-fold n_I-iteration RTS. So, in such a *K*-fold IRTS, there are *K* different values of n_I, each of which is determined by the stopping rule of a single one-fold IRTS, i.e., one-fold n_I-iteration RTS. The third and also crucial one is how to augment datasets iteratively used by IRTS, i.e., one-fold n_I-iteration RTS. This issue does not exist in the *K*-fold method which uses the same dataset repeatedly *K* times. The idea of data augmentation by iterations is originated and derived from the iterative constrained energy minimization (ICEM) [13]. Because CEM is a subpixel target detector and does not take spatial information into account, ICEM includes a set of Gaussian filters to capture spatial correlation from CEM-detected maps and feeds the Gaussian filtered CEM-detected maps back to be added to the current image cube to create a new augmented image cube. Then, the same number of new training samples is randomly selected from this new augmented image cube again for the use of the next

iteration. Inspired by ICEM, a similar idea is developed for IRTS by replacing CEM-detected maps with spectral classification maps, such as edge preserving filter (EPF) in [2] or Gabor filter in [14]. Accordingly, IRTS implements a feedback system which augments the currently being processed image cube by adding a new set of SF-ed spectral classification maps provided by each feedback loop one iteration after another, and then randomly reselects training samples from the newly augmented image cube as a new set of training samples for the next round iteration. So, generally speaking, IRTS can be implemented coupled with any supervised classifier which requires training samples to perform classification. However, to manifest the utility of IRTS in classification, this article particularly selects spectral-spatial (SS) classification to perform spectral classification and EPFs as spatial filters (SFs) to capture spatial information from the spectral classification maps which will be used to augment the data cubes.

A general SS classifier implements a spectral classifier followed by an SF to perform classification. One of the most widely used supervised classifiers for HSIC is the support vector machine (SVM) to find an optimal hyperplane separating two classes. However, one main issue of SVM or any supervised classifier is the requirement of a training sample set which must be provided *a priori* to train the classifier. In other words, the result of classification is strongly influenced by a selected training sample set. Such RTS generally results in inconsistent and different classification results. That is, when two different SS classifiers are implemented for classification, it is often the case that one SS classifier may perform better than the other on a training sample set but worse on another different training sample set. This is why the *K*-fold method has been widely used to address this issue for cross validation.

Despite the fact that several sampling approaches for HSIC are proposed in the literature to improve the classification performance over traditional random sampling strategy, none of them is close to IRTS developed in this article. For example, [15] showed that the most informative training samples used to train an SVM classifier were those boundary samples which could create an effective hyperplane for binary classification. Another is recent work on training sample selection reported in [16], which introduced a controlled random sampling (CRS) strategy to alleviate the side effects resulting from traditional random sampling due to overlap between training and testing samples selected from the same image. IRTS is completely different from any existing sampling approach and has never been explored before. Specifically, the EPF-based classifiers in [2] are particularly selected as SS-classifiers implemented in conjunction with IRTS to derive IRTS-EPF. So, IRTS-EPF can be considered as an advanced version of IEPF [17] where IEPF initially selected a set of randomly generated training samples and used the same selected training sample set through the entire iterative processes. Unlike IEPF, IRTS-EPF randomly reselects a different set of training sample at each iteration.

Concluding this section, we summarize several contributions made by IRTS-SS.

1) A new concept of random classification via an RTS theory is introduced where a traditional classifier can be extended and treated as a random classifier which can make random decisions for classification. As a result, conventional SS classification can be extended to IRTS-SS classification.
2) To measure the effectiveness of IRTS-SS, several new information measures are derived from an information theory, such as CSI, CE, overall CE (OCE)/average CE (ACE), and sample entropy (SE).
3) By virtue of IRTS, many SS classifiers, such as ISVM [18], EPF [2], IEPF [17], can be readily extended to their corresponding IRTS versions, IRTS-SVM, IRTS-Gaussian, IRTS-Gabor, IRTS-EPF (including bilateral and guided filters), and IRTS-IEPF.
4) Since EPF is strongly affected by its use of SFs, two new fusion methods, referred to as IRTS-GEPF, which fuses Gaussian filters with EPF, and IRTS-Gabor-EPF, which fuses Gabor filters with EPF, are further proposed to improve classification performance.

II. Random Classification

In this section, we interpret the classification using a set of randomly selected training samples as a random classification problem. Specifically, let $\mathbf{r}(x,y)$ be a data sample at spatial location (x,y) and δ be a classifier which classifies $\mathbf{r}(x,y)$ into one of M classes $\{C_m\}_{m=1}^{M}$. Also, let the likelihood of a data sample to be classified into a class be expressed by M probabilities$\{p_m(\mathbf{r}(x,y))\}_{m=1}^{M}$ where $p_m(\mathbf{r}(x,y))$is the probability of $\mathbf{r}(x,y)$ to be classified by $\delta(\mathbf{r}(x,y))$ into mth class, C_m when S^{training} is used to train the classifier $\delta(\mathbf{r}(x,y))$ for classification. So, the uncertainty of the data sample $\mathbf{r}(x,y)$ to be classified by $\delta(\mathbf{r}(x,y))$ into $\{C_m\}_{m=1}^{M}$ is described by $\{p_m(\mathbf{r}(x,y))\}_{m=1}^{M}$ with the probability mass function (PMF) obtained by a given selected training sample set S^{training} denoted by $\text{PMF}(\mathbf{r}(x,y), S^{\text{training}})$. In this case, for each data sample $\mathbf{r}(x,y)$, we can define a random variable by $\zeta_{S^{\text{training}}}(\delta(\mathbf{r}(x,y)))$ on a sample space specified by M classes $\Omega_{\text{class}} = \{C_m\}_{m=1}^{M}$ via $\delta_{S^{\text{training}}}(\mathbf{r}(x,y))$with PMF determined by a given selected set of random training samples S^{training}.

To simplify and clarify notations, we let $\{\mathbf{r}(x,y)\}_{(x,y)} = \{\mathbf{r}_n = \mathbf{r}(x,y)\}_{n=1}^{N}$ denote a total number of N data samples in a hyperspectral image. In this case, there are N random variables $\{\zeta_{S^{\text{training}}}(\delta(\mathbf{r}_n))\}_{n=1}^{N}$, each of which has its own $\text{PMF}(\mathbf{r}(x,y), S^{\text{training}}) = \text{PMF}(\mathbf{r}_n, S^{\text{training}})$ to generate a probability distribution PMF defied on $\Omega_{\text{class}} = \{C_m\}_{m=1}^{M}$. By virtue of this $\text{PMF}(\mathbf{r}_n, S^{\text{training}})$, the uncertainty of a data sample $\mathbf{r}_n$ to be classified into $\Omega_{\text{class}} = \{C_m\}_{m=1}^{M}$ can then be described by classification inconsistency for $\mathbf{r}_n$ using S^{training}. On the other hand, since the training sample set S^{training} is selected randomly, it also creates a random variable to describe the randomness of selecting training samples set. That is, the probability of using S^{training} is denoted by PMF(S^{training}). Using the PMF of such random variables, we can further define SE to describe the uncertainty of a data sample $\mathbf{r}_n$ resulting from using S^{training} with the probability given by PMF(S^{training}).

Basically, there are two random variables associated with each of data sample $\mathbf{r}_n \in \{\mathbf{r}_n\}_{n=1}^{N}$. One is $\{\zeta_{S^{\text{training}}}(\delta(\mathbf{r}_n))\}_{n=1}^{N}$ used to describe uncertainty of the data sample $\mathbf{r}_n$ caused by

$\mathrm{PMF}(\mathbf{r}_n, S^{\mathrm{training}})$ using a specific S^{training}. The other is another random variable ξ to describe the SE caused by the uncertainty of choosing S^{training} according to PMF(S^{training}).

More specifically, assume that $S^{\mathrm{training}(l)} = \bigcup_{m=1}^{M} S_m^{\mathrm{training}(l)}$ where $S_m^{\mathrm{training}(l)}$ is the set of training samples in the *l*th selected training sample set used to train a classifier for the *m*th class. Then, all the total number of possible training sample sets are given by

$$S^{\mathrm{total}} = \bigcup_{l=1}^{N^{\mathrm{total}}} S^{\mathrm{training}(l)} = \bigcup_{l=1}^{N^{\mathrm{total}}} \left\{ \bigcup_{m=1}^{M} S_m^{\mathrm{training}(l)} \right\}. \quad (1)$$

Therefore, the random variable $\xi(S^{\mathrm{training}(l)})$ is defined on the sample space S^{total} defined in (1) with the probability of selecting the *l*th training sample set $S^{\mathrm{training}(l)}$ assumed to be equally likely and given by

$$\mathrm{PMF}(S^{\mathrm{training}(l)}) = \frac{1}{N^{\mathrm{total}}}. \quad (2)$$

Despite N^{total} being considerably large, it does not imply that we need all N^{total} of training sample sets for classification. The performance of a classifier is evaluated by AA and OA, and its class uncertainty measured by $\mathrm{PMF}(\mathbf{r}_n, S^{\mathrm{training}})$ and PMF(S^{training}). Finally, we use the three most widely used HSI scenes, Purdue's Indian Pines, Salinas, and University of Pavia to illustrate the effectiveness of IRTS in uncertainty reduction and improvement of classification accuracy.

III. RTS Theory

Following Section II, we can further develop a theory, called RTS theory to describe the uncertainty of classification inconsistency caused by random sampling.

Consider a data sample $\mathbf{r}_n$ to be classified into *M* classes $\{C_m\}_{m=1}^{M}$. The classification of $\mathbf{r}_n$ can be viewed as rolling a dice with *M* faces where each face represents a specific class of interest. We further assume that S^{training} is a set of training samples to be selected for classification and $\delta_{S^{\mathrm{training}}}(\mathbf{r}_n)$ is a classifier using S^{training} to classify $\mathbf{r}_n$ into one of *M* classes, say, *m*th class, i.e., $\delta_{S^{\mathrm{training}}}(\mathbf{r}_n) = C_m$. Then, we can define a random variable $\zeta(\delta_{S^{\mathrm{training}}}(\mathbf{r}_n))$ on a probability space with the sample space specified by *M* classes $\Omega_{\mathrm{class}} = \{C_m\}_{m=1}^{M}$ and its PMF given by $p(\zeta(\delta_{S^{\mathrm{training}}}(\mathbf{r}_n) = C_m) = p_m(\mathbf{r}_n)$, which represents the class membership probability of assigning $\mathbf{r}_n$ by the classifier $\delta_{S^{\mathrm{training}}}$ using the training sample set S^{training} to the *m*th class C_m. Now, we can further assume that the entire data sample space is given by $\Omega_{\mathrm{data}} = \{\mathbf{r}_n\}_{n=1}^{N}$ where $\mathbf{r}_n$ is the *n*th data sample located at its particular spatial coordinate (*x*,*y*). In this case, performing the classification of the *n*th data sample $\mathbf{r}_n$ is equivalent to rolling a dice with *M* faces using the random variable $\zeta(\delta_{S^{\mathrm{training}}}(\mathbf{r}_n))$ defined above for $\mathbf{r}_n$. So, classifying the entire data space $\Omega_{\mathrm{data}} = \{\mathbf{r}_n\}_{n=1}^{N}$ can be carried out by *N* independent random variables $\{\zeta_{S^{\mathrm{training}}}(\delta(\mathbf{r}_n))\}_{n=1}^{N}$ defined on the sample space $\Omega_{\mathrm{class}}^{N} = \prod_{n=1}^{N} \Omega_{\mathrm{class}}^{n}$ with $\Omega_{\mathrm{class}}^{n} = \Omega_{\mathrm{class}}$ for $1 \le n \le N$.

For each class C_m, let n_m be the number of data samples in C_m. Then, the total number of data samples to be classified without including BKG samples is denoted by *N* given by

$$N = \sum_{m=1}^{M} n_m. \quad (3)$$

Also, let S_m^{training} be the set of training samples selected from C_m to be used to classify data samples in C_m and S^{training} be the total number of training samples used for classification and given by

$$S^{\mathrm{training}} = \bigcup_{m=1}^{M} S_m^{\mathrm{training}} \quad (4)$$

where $n_m^{\mathrm{training}} = |S_m^{\mathrm{training}}|$ is the number of training samples in S_m^{training}.

According to RTS, we randomly select n_m^{training} samples from C_m according to ground truth. So, when the *K*-fold method is used to evaluate the performance of a classifier, it will have a total number of possible training sample sets that it can choose from, that is

$$\begin{aligned} N^{\mathrm{total}} &= \binom{n_1}{n_1^{\mathrm{training}}} \times \binom{n_2}{n_2^{\mathrm{training}}} \times \cdots \times \binom{n_M}{n_M^{\mathrm{training}}} \\ &= \prod_{m=1}^{M} \binom{n_m}{n_m^{\mathrm{training}}} \end{aligned} \quad (5)$$

where $\binom{n_m}{n_m^{\mathrm{training}}} = \frac{n_m!}{n_m^{\mathrm{training}}!(n_m - n_m^{\mathrm{training}})!}$.

So, the classification of $\Omega_{\mathrm{data}} = \{\mathbf{r}_n\}_{n=1}^{N}$ using the *l*th training sample set $S^{\mathrm{training}(l)}$ can be considered as rolling a dice with *M* faces described by *N* independent random variables $\{\zeta_{S^{\mathrm{training}(l)}}(\delta(\mathbf{r}_n))\}_{n=1}^{N}$. As $\{\zeta_{S^{\mathrm{training}(l)}}(\delta(\mathbf{r}_n))\}_{n=1}^{N}$ runs through all training sample sets $S^{\mathrm{training}(l)}$ over S^{total} in (1), we will have

$$\begin{aligned} &\left\{ \{\zeta_{S^{\mathrm{training}(l)}}(\delta(\mathbf{r}_n))\}_{n=1}^{N} \right\}_{l=1}^{N^{\mathrm{total}}} \\ &\quad = \left\{ \{\zeta_{S^{\mathrm{training}(l)}}(\delta(\mathbf{r}_n))\}_{n=1}^{N} \right\}_{S^{\mathrm{training}(l)} \in S^{\mathrm{total}}} \end{aligned} \quad (6)$$

which shows that the classifier δ runs through all the training sample sets given by (1) as well as the entire data sample space $\Omega_{\mathrm{data}} = \{\mathbf{r}_n\}_{n=1}^{N}$ to perform classification.

A. Probability of Classification

Suppose that δ is any arbitrary classifier used to classify data samples $\Omega_{\mathrm{data}} = \{\mathbf{r}_n\}_{n=1}^{N}$. In general, the class membership assignment maps produced by δ consist of *M* binary maps, each of which represents the classification result of an individual class. These *M* binary values produced by δ for each sample $\mathbf{r}_n$ can be represented by an *M*-dimensional column vector

$$\delta(\mathbf{r}_n) = \mathbf{y}(\mathbf{r}_n) = \begin{bmatrix} y_1(\mathbf{r}_n) \\ y_2(\mathbf{r}_n) \\ \vdots \\ y_M(\mathbf{r}_n) \end{bmatrix} \quad (7)$$

where $y_m(\mathbf{r}_n)$ is a binary-valued class membership assignment for classifying $\mathbf{r}_n$ into the *m*th class C_m given by

$$y_j(\mathbf{r}_n) = \begin{cases} 1 \text{ for } j = m \\ 0 \text{ for } j \neq m \end{cases} \quad \text{for } 1 \leq j \leq M \tag{8}$$

with $\mathbf{r}_n$ assumed to be in C_m, i.e., $\mathbf{r}_n \in C_m$. More precisely, for each $y_m(\mathbf{r}_n)$, we can formulate the prior probability of $y_m(\mathbf{r}_n)$ conditioning on $p_m(\mathbf{r}_n)$ according to $\{\zeta_{S^{\text{training}(l)}}(\delta(\mathbf{r}_n))\}_{S^{\text{training}(l)} \in S^{\text{total}}}$ by only focusing on the single data sample $\mathbf{r}_n$ as follows:

$$p(y_m(\mathbf{r}_n)\,|p_m(\mathbf{r}_n)\,) = \begin{cases} p_m(\mathbf{r}_n), & y_m(\mathbf{r}_n) = 1 \\ 1 - p_m(\mathbf{r}_n), & y_m(\mathbf{r}_n) = 0 \end{cases} \tag{9}$$

where $p_m(\mathbf{r}_n)$ indicates the probability of the data sample $\mathbf{r}_n$ assigned to C_m, i.e., a dice tossed for $\mathbf{r}_n$ with the face up turned out to be *m*. Specifically, $p_m(\mathbf{r}_n)$ is the probability that a classifier will classify $\mathbf{r}_n$ into C_m, $p_m(\mathbf{r}_n)$ and can be further defined as

$$p_m(\mathbf{r}_n) = \frac{N^{\text{total}}_{y_m(\mathbf{r}_n)=1}}{N^{\text{total}}}, 1 - p_m(\mathbf{r}_n) = \frac{N^{\text{total}}_{y_m(\mathbf{r}_n)=0}}{N^{\text{total}}} \tag{10}$$

where $N^{\text{total}}_{y_m(\mathbf{r}_n)=1}$ represents the number of training sample sets in N^{total} that classify $\mathbf{r}_n$ into the *i*th class, and $N^{\text{total}}_{y_m(\mathbf{r}_n)=0}$ otherwise. Therefore, the closer $N^{\text{total}}_{y_m(\mathbf{r}_n)=1}$ is to N^{total}, the easier it is for a classifier to classify $\mathbf{r}_n$. Finally,

$$\sum_{m=1}^{M} p_m(\mathbf{r}_n) = 1 \tag{11}$$

where the parameter $p_m(\mathbf{r}_n)$ indicates the probability or likelihood of $\mathbf{r}_n$ to be classified into C_m.

B. Uncertainty Reduction

The concept of data sample uncertainty is very simple, which is determined by $p_m(\mathbf{r}_n)$. That is, the smaller the value of $p_m(\mathbf{r}_n)$, the more uncertainty of $\mathbf{r}_n$ being assigned to C_m. Therefore, a good classifier tries to make data sample uncertainty as small as possible by letting

$$p_{j=m}(\mathbf{r}_n) \to 1, p_{j\neq m}(\mathbf{r}_n) \to 0 \text{ with } \mathbf{r}_n \in C_m. \tag{12}$$

This means that as long as there is a sufficient number of training sample sets $S^{\text{training}(l)}$ selected from S^{total} in (1) for the classifier δ, the degree of classification inconsistency will approach zero asymptotically. So, an interesting question is "is there an effective as well as an efficient way to minimize classification inconsistency without the need of going through a large number of training sample sets selected from S^{total}?" The IRTS proposed in Section IV is particularly developed to address this issue.

IV. SS Classification Using IRTS

The IRTS presented in this section is to implement the RTS discussed above iteratively where RTS is performed independently iteration-by-iteration.

Many approaches to HSIC have been reported in the literature, among which SS-based methods are probably one of most popular and widely used classification methods. An SS method first implements a spectral-based classifier to perform spectral classification and then follows up with SFs to obtain spatial information from spectral-classified data samples. Such spatial correlation captured by the SFs is indeed provided by the used training samples. If this information is fed back to be added to the current data cube to expand the data cube, then this new augmented data cube will contain much needed spatial classification information to help improve classification. Then, we can perform RTS again to select another new set of training samples from this new data cube. As a result of the newly added SF-ed classification information provided by the new RTS-selected training samples, the classification performance will be further improved. The same process is continued on iteratively and the issue of inconsistent classification results caused by random training sample selection can, therefore, be reduced. Due to its nice SS structure, an SS method is a perfect candidate to be used in conjunction with IRTS. So, when IRTS is implemented in conjunction with an SS method, it is referred to as IRTS-SS. For example, if SVM and Gaussian filter are used for spectral classifier and SF, respectively, it is referred to as IRTS-SVM-Gaussian.

Two key steps implemented in IRTS-SS are worth being mentioned. One is that the training sample set used in each iteration is randomly selected. This is quite different from using the same training sample set in all iterations as was done [13]–[14], [17]–[19]. The other is to fuse classification results produced by two randomly selected training sample sets. Since the captured SF-ed information provided by two different sets of training samples contains more classification information than that provided by the same fixed training sample set, the uncertainty caused by classification inconsistency is further reduced by fusion.

Suppose that at the *l*th iteration IRTS-SS implements an SF denoted by $\text{SF}^{(l)}$ which uses the *l*th training sample set $S^{\text{training}(l)}$ to produce the *l*th *M* spectral classification maps $\{\{\text{SF}^{(l)}_m(\mathbf{r}_n)\}^N_{n=1}\}^M_{m=1}$ where $\{\text{SF}^{(l)}_m(\mathbf{r}_n)\}^N_{n=1}$ is the C_m-classification map for $\Omega_{\text{data}} = \{\mathbf{r}_n\}^N_{n=1}$. Then, the fusion step of IRTS-SS fuses the *m*th spectral classification map at the $(l-1)$th iteration $\text{SF}^{(l-1)}_m(\mathbf{r}_n)$ and the *m*th spectral classification map at the *l*th iteration$\text{SF}^{(l)}_m(\mathbf{r}_n)$ for each data sample $\mathbf{r}_n$ by taking their maximum for $1 \leq m \leq M$ as follows:

$$\begin{aligned} \text{MAX}^{(l)}_m(\mathbf{r}_n) &= \max\left\{\text{SF}^{(l)}_m(\mathbf{r}_n), \text{SF}^{(l-1)}_m(\mathbf{r}_n)\right\} \\ \text{MAXMap}^{(l)}_m &= \left\{\text{MAX}^{(l)}_m(\mathbf{r}_n)\right\}_{\mathbf{r}_n \in \Omega_{\text{data}}}. \end{aligned} \tag{13}$$

It should be noted that the training sample sets $S^{\text{training}(l-1)}$ used in the $(l-1)$th iteration and $S^{\text{training}(l)}$ in the *l*th iteration are randomly selected and they are different and completely independent. The fusion process in (13) is to jointly capture the information provided by the two different training sample sets $S^{\text{training}(l-1)}$ and $S^{\text{training}(l)}$. Furthermore, the resulting fused *M* classification maps obtained from (13) are then fed back to the image data cube currently being processed to create a new data cube, which will be used as a new dataset to reprocess classification again in an iterative manner. This feedback process can reduce the uncertainty caused by RTS. More specifically, each iteration makes use of a different set of training data. If

IRTS-SS is terminated by K iterations, it means that IRTS-SS has used K different sets of training samples for classification. The niche is that IRTS-SS correlates classification results produced by (13) to improve classification accuracy while reducing classification uncertainty. This is very different from running the K-fold method which implements the same algorithm K times independently without taking advantage of the correlation among the K results. By including more and more spatial information provided by SFs using different sets of training samples from augmented data cubes iteration after iteration, the classification results become more consistent and gradually converge until the fused classification maps meet a stopping rule which is Tanimoto index (TI) [20], defined as

$$\mathrm{TI}_m^{(l)} = \frac{|\mathrm{BMAXMap}_m^{(l)} \cap \mathrm{BMAXMap}_m^{(l-1)}|}{|\mathrm{BMAXMap}_m^{(l)} \cup \mathrm{BMAXMap}_m^{(l-1)}|} > \varepsilon \qquad (14)$$

for $1 \le m \le M$ where the maximum *a posteriori* (MAP) defined by (15) is used to produce binary map $\mathrm{BMAXMap}_m^{(l)}$ as follows.

$$\mathrm{BMAX}_m^{(l)}(\mathbf{r}_n) = \begin{cases} 1; \mathrm{MAX}_m^{(l)}(\mathbf{r}_n) > \mathrm{MAX}_{i\in\{1,\cdots,M\}, i\neq m}^{(l)}(\mathbf{r}_n) \\ 0; \qquad\qquad \text{otherwise} \end{cases}$$

$$\mathrm{BMAXMap}_m^{(l)} = \left\{\mathrm{BMAX}_m^{(l)}(\mathbf{r}_n)\right\}_{\mathbf{r}_n\in\Omega_{\mathrm{data}}} \qquad (15)$$

at the lth iteration. The idea of TI is to measure the discrepancy between binary map $\mathrm{BMAXMap}_m^{(l-1)}$ and binary map $\mathrm{BMAXMap}_m^{(l)}$. If these two maps are identical, then TI = 1. On the other hand, if these two maps are disjoint, then TI = 0. So, TI is right between 0 and 1 and the greater TI is, the better the match between the two classification maps is. Once the algorithm is terminated, $\mathrm{BMAXMap}_m^{(l)}$ is the final classification map. The details of step-by-step implementation of IRTS-SS are provided in the following.

IRTS-SS

1) Initial condition: Let $\Omega^{0)}$ and $S^{\mathrm{training}(0)}$ be the original HSI image cube and an initial randomly selected training sample set. Let ε be a prescribed threshold. Set $l = 0$.
2) At the lth iteration, implement an SS classifier on $S^{\mathrm{training}(l)}$ to produce a probability map of each class C_m $\mathrm{SF}^{(l)} = \{\{\mathrm{SF}_m^{(l)}(\mathbf{r}_n)\}_{n=1}^N\}_{m=1}^M$ for $1 \le m \le M$.
3) If $l = 0$, let $l \leftarrow l + 1$and go to step 2. Otherwise, continue.
4) Fusion process: For each sample $\mathbf{r}_n$, find $\mathrm{MAXMap}_m^{(l)} = \{\mathrm{MAX}_m^{(l)}(\mathbf{r}_n)\}_{\mathbf{r}_n\in\Omega_{\mathrm{data}}}$ via (13).
5) Implement MAP (15) on$\{\mathrm{MAXMap}_m^{(l)}\}_{m=1}^M$to produce classification maps$\{\mathrm{BMAXMap}_m^{(l)}\}_{m=1}^M = \{\{\mathrm{BMAX}_m^{(l)}(\mathbf{r}_n)\}_{n=1}^N\}_{m=1}^M$.
6) If $\mathrm{TI}^{(l)} > \varepsilon$, go to step 8. Otherwise, continue.
7) Form a new data cube

$$\mathbf{\Omega}^{(l)} = \mathbf{\Omega}^{(l-1)} \cup \left\{\mathrm{MAXMap}_1^{(l)}\right\} \cup \cdots \cup \left\{\mathrm{MAXMap}_M^{(l)}\right\} \qquad (16)$$

and regenerate a new set of training samples $S^{(l)}$. Let $l \leftarrow l + 1$, go to step 2.
8) The algorithm is terminated and $\{\mathrm{BMAXMap}_m^{(l)}\}_{m=1}^M$ are the final classification maps.

The diagram of IRTS-SS depicted in Fig. 1 shows that at the lth iteration, IRTS-SS uses the fusion process described by (13) in step 4 in Fig. 2 to produce a new set of M SF-filtered classification maps that will be fed back via a feedback loop to be added to the current data cube $\mathbf{\Omega}^{(l)}$ to create a new data cube $\mathbf{\Omega}^{(l+1)}$ described by step 7 for the next $(l+1)$th iteration. Then, the new training sample set $S^{\mathrm{training}(l+1)}$ will be selected from $\mathbf{\Omega}^{(l+1)}$and used for spectral classification at the $(l+1)$th iteration. In other words, each iteration represents an SS classification using a newly selected training sample set. The key feature of IRTS-SS is to fuse two consecutive SF-ed spectral classification maps $\{\{\mathrm{SF}_m^{(l-1)}(\mathbf{r}_n)\}_{n=1}^N\}_{m=1}^M$ and $\{\{\mathrm{SF}_m^{(l)}(\mathbf{r}_n)\}_{n=1}^N\}_{m=1}^M$ described in Fig. 2.

When IRTS-SS is implemented by an SS classifier, it allows users to select a spectral classifier, such as SVM, and an SF, such as Gaussian filter, Gabor filter [14], or EPF, to perform classification. In this article, two SFs are selected to implement IRTS-SS. The first one is EPF proposed in [2]. Even though EPF shows significant improvement in classification performance of HSIC, its performance is strongly affected by the used guided image, which requires first or first three principal component (PC) images of the original image cube to capture spatial information for binary spectral-classification maps. In order to solve this problem, we propose a new method which fuses the results by a Gaussian filter and EPF, $\mathrm{G}^{(l)}$ and $\mathrm{EPF}^{(l)}$ at the lth iteration by taking their maximum for $1 \le m \le M$

$$\mathrm{GEPF}_m^{(l)}(\mathbf{r}_n) = \max\left\{\mathrm{G}_m^{(l)}(\mathbf{r}_n), \mathrm{EPF}_m^{(l)}(\mathbf{r}_n)\right\} \qquad (17)$$

over each data sample $\mathbf{r}_n$. The method resulting from using (17) is called Gaussian-EPF fused filter, referred to as GEPF.

Similar to (17), another filter, Gabor [14] can be also used to fuse with EPF to derive Gabor-EPF fused filter (Gabor-EPF) by fusing $\mathrm{Gabor}^{(l)}$ and $\mathrm{EPF}^{(l)}$ at the lth iteration by taking their maximum for $1 \le m \le M$

$$\mathrm{Gabor-EPF}_m^{(l)}(\mathbf{r}_n) = \max\left\{\mathrm{Gabor}_m^{(l)}(\mathbf{r}_n), \mathrm{EPF}_m^{(l)}(\mathbf{r}_n)\right\} \qquad (18)$$

over each data sample $\mathbf{r}_n$.

V. Information Measures Derived for RTS

As it is designed, IRTS-SS utilizes RTS to run an SS classifier iteratively. It works like the K-fold method to improve classification inconsistency caused by RTS. In the mean time, it also reduces uncertainty resulting from the randomness by RTS. But how to assess the effectiveness of RTS quantitatively in terms of classification inconsistency and uncertainty becomes a challenging issue. This section develops several quantitative measures from information theory.

For the purpose of illustration, we implement RTS K times and each time uses a different set of randomly selected training samples. We then calculate their average as a mean for final classification along with the SD to reflect the confidence levels of classification results. It should be noted that K must be sufficiently large to generate meaningful and reliable statistics.

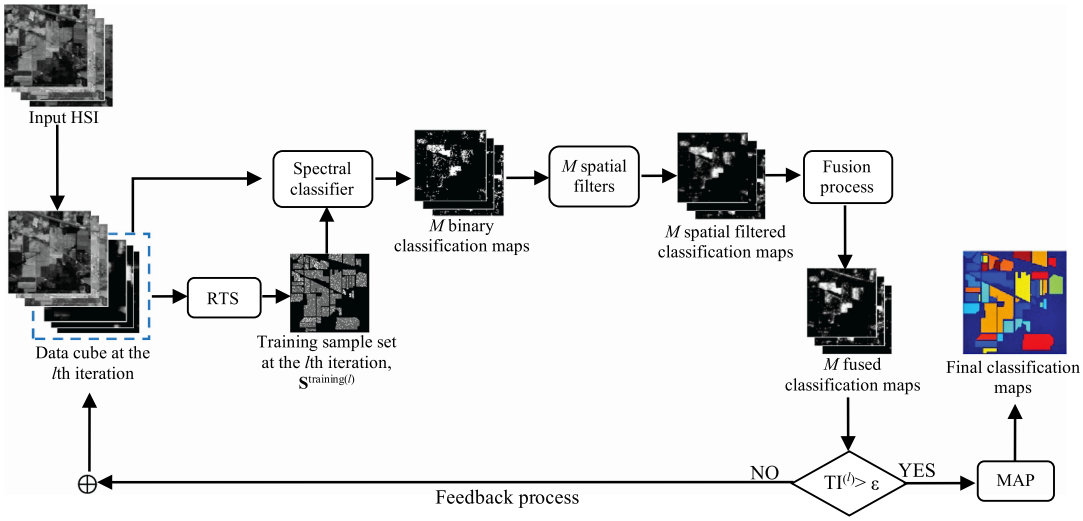


Fig. 1. Graphic diagram of iterative process in IRTS-SS.

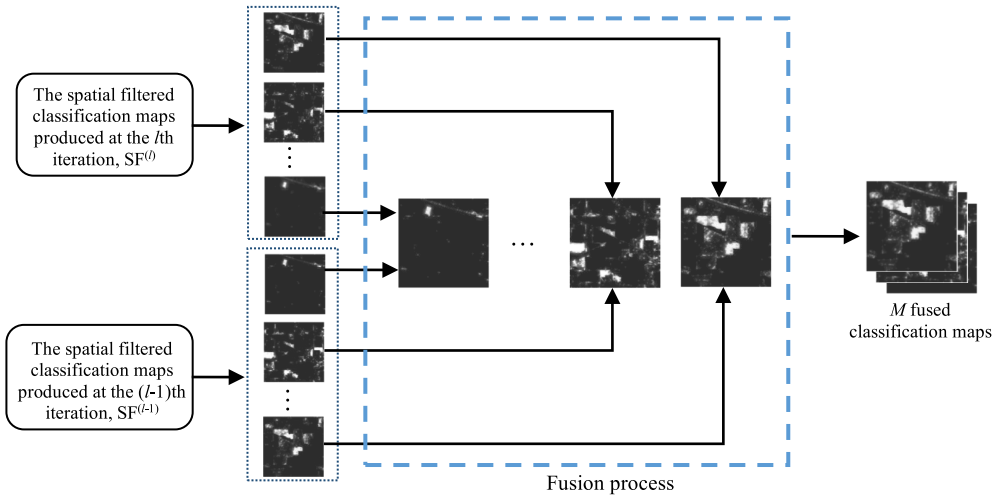


Fig. 2. Graphic diagram of fusion process (13) in IRTS-SS.

A. Classification Standard Deviation (CSD)

CSD calculates SD of classification of each data sample produced by running RTS K times. Specifically, for a given data sample $\mathbf{r}_n$, we define the single sample SD (SSD) of $\mathbf{r}_n$ in the C_m-classification map, denoted by $\mathrm{SSD}_m(\mathbf{r}_n) = \sigma_m(\mathbf{r}_n)$, as

$$\begin{aligned}\mathrm{SSD}_m(\mathbf{r}_n) &= \sigma_m(\mathbf{r}_n) \\ &= \sqrt{\frac{1}{K}\sum_{k=1}^{K}\left(\delta_m^{(k)}(\mathbf{r}_n) - \bar{\delta}_m^{K}(\mathbf{r}_n)\right)^2}\end{aligned} \tag{19}$$

where $\delta_m^{(k)}(\mathbf{r}_n)$ is defined by (7) with the subscript m indicating the value of $\delta(\mathbf{r}_n)$ obtained for the mth class C_m, K is the number of times RTS is carried out, and $\bar{\delta}_m^{(K)}(\mathbf{r}_n) = \frac{1}{K}\sum_{k=1}^{K}\delta_m^{(k)}(\mathbf{r}_n)$. We can then further define the single C_m-CSD (CSD_m) by averaging $\mathrm{SSD}_m(\mathbf{r}_k)$ over $\mathbf{r}_n$ classified into C_m by

$$\mathrm{CSD}_m = \frac{1}{n_m}\sum_{\mathbf{r}_n \in C_m}\mathrm{SSD}_m(\mathbf{r}_n). \tag{20}$$

Finally, the average of CSD_m in (20) over M classes $\{C_m\}_{m=1}^{M}$ can be calculated in two ways, either by the class sample ratio (SR), n_m/N, referred to as overall CSD (OCSD)

$$\mathrm{OCSD} = \sum_{m=1}^{M}\frac{n_m}{N}\mathrm{CSD}_m \tag{21}$$

which assumes that each class is weighted by SR or by letting $\frac{n_m}{N} = \frac{1}{M}$, referred to as average CSD (ACSD)

$$\mathrm{ACSD} = \frac{1}{M}\sum_{m=1}^{M}\mathrm{CSD}_m \tag{22}$$

which assumes that each class is weighted equally by 1/M.

B. Classification Entropy

For a given data sample $\mathbf{r}_n$ we consider the probability of K binary classification maps of $\mathbf{r}_n$ assigned to a particular class,

say, C_m by a classifier δ. From (7)–(9), we can define

$$\bar{p}_m^K(\mathbf{r}_n) = \frac{1}{K}\sum_{k=1}^{K} y_m^{(k)}(\mathbf{r}_n) \tag{23}$$

where $y_m^{(k)}(\mathbf{r}_n)$ indicates the binary classification map of C_m assigned to $\mathbf{r}_n$ by a classifier in the *k*th run, i.e., using the *k*th random training sample set $S^{\text{trianing}(k)}$. Then, the average probability vector can be defined by

$$\bar{\mathbf{p}}^K(\mathbf{r}_n) = \begin{bmatrix} \bar{p}_1^K(\mathbf{r}_n) \\ \bar{p}_2^K(\mathbf{r}_n) \\ \vdots \\ \bar{p}_M^K(\mathbf{r}_n) \end{bmatrix} = \frac{1}{K}\begin{bmatrix} \sum_{k=1}^{K} y_1^{(k)}(\mathbf{r}_n) \\ \sum_{k=1}^{K} y_2^{(k)}(\mathbf{r}_n) \\ \vdots \\ \sum_{k=1}^{K} y_M^{(k)}(\mathbf{r}_n) \end{bmatrix}. \tag{24}$$

Once $\bar{\mathbf{p}}^K(\mathbf{r}_n)$ is calculated by (24), we can define CSI of $\mathbf{r}_n$ being classified into C_m by running a classifier *K* times using *K* randomly selected training sample sets as $I_m(\mathbf{r}_n)$

$$I_m(\mathbf{r}_n) = -\log \bar{p}_m^K(\mathbf{r}_n) \tag{25}$$

which indicates that the higher the $\bar{p}_m^K(\mathbf{r}_n)$ value is, the smaller the CSI of $\mathbf{r}_n$ and thus, the less the uncertainty of $\mathbf{r}_n$ being classified into the *m*th class by *K* runs. Using (25), we can further define the uncertainty of the data sample $\mathbf{r}_n$ as Shannon's SE by

$$\begin{aligned} H(\mathbf{r}_n) &= -\sum_{m=1}^{M} \bar{p}_m^K(\mathbf{r}_n)\log \bar{p}_m^K(\mathbf{r}_n) \\ &= \sum_{m=1}^{M} \bar{p}_m^K(\mathbf{r}_n) I_m(\mathbf{r}_n). \end{aligned} \tag{26}$$

So, a higher $H(\mathbf{r}_n)$ indicates a higher uncertainty contained in $\mathbf{r}_n$ to be classified by a classifier running *K* times. Conversely, a smaller $H(\mathbf{r}_n)$ implies a smaller uncertainty contained in $\mathbf{r}_n$. As a result, the classification results of $\mathbf{r}_n$ gradually become deterministic. Furthermore, like (20), we can also define C_m-CE by averaging SE in (26) over the data samples classified into C_m by

$$H_{\text{CE}}(C_m) = \frac{1}{n_m}\sum_{\mathbf{r}_n \in C_m} H(\mathbf{r}_n). \tag{27}$$

In analogy with (21) and (22), we can also use (27) to define the OCE similar to (20) by SR as

$$H_{\text{OCE}} = \sum_{m=1}^{M} \frac{n_i}{N} H_{\text{CE}}(C_m) \tag{28}$$

and let $\frac{n_m}{N} = \frac{1}{M}$ to define ACE similar to (22)

$$H_{\text{ACE}} = \frac{1}{M}\sum_{m=1}^{M} H_{\text{CE}}(C_m). \tag{29}$$

V. Performance Evaluation

Several classification measures defined in [21] will be used for the experiments and are summarized as follows.

We first assume the following.

n_{mm} = Number of signal samples in the *m*th class correctly classified into the *m*th class $\hat{C}_m$.
n_{jm} = Number of data samples in the *m*th class C_m but actually classified into the *j*th class $\hat{C}_j$.
M = Number of classes.
C_m = Set of data samples in the *m*th class, by ground truth.
$n_m = \sum_{j=1}^{M} n_{jm}$ = Number of data samples in C_m.
N = Total number of data samples, $N = \sum_{m=1}^{M} n_m$.

In traditional HSIC, the popular performance measurements are CA and OA, given by

$$p_{\text{A}}(C_m) = \text{accuracy of the } m\text{th class} = \frac{n_{mm}}{\sum_{j=1}^{M} n_{jm}} = \frac{n_{mm}}{n_m} \tag{30}$$

$$\text{P}_{\text{OA}} = \frac{1}{N}\sum_{m=1}^{M} n_{mm} = \sum_{m=1}^{M} \frac{n_m}{N} p_{\text{A}}(C_m) \tag{31}$$

which shows that P_{OA} utilizes sample ratios as weights to average the accuracy of each class. In addition, we can consider all classes to be equally likely by letting $\frac{n_m}{N} = \frac{1}{M}$ to further define P_{AA} as AA by the number of classes *M*, expressed as

$$\text{P}_{\text{AA}} = \frac{1}{M}\sum_{m=1}^{M} \frac{n_{mm}}{n_m} = \frac{1}{M}\sum_{m=1}^{M} p_{\text{A}}(C_m). \tag{32}$$

Another measure is a new criterion developed in [13], called precision rate (PR) given by

$$p_{\text{PR}}(C_m) = \text{PR of } C_m = \frac{\hat{n}_{mm}}{\hat{n}_m} \tag{33}$$

where $\hat{n}_m = \sum_{j=1}^{M} \hat{n}_{mj}$ is the total number of data samples that be classified into the *m*th class and $\hat{n}_{mj}$ is the number of data samples classified into *m*th class but supposed to be in *j*th class. The overall PR (OPR) is defined by

$$P_{\text{OPR}} = \sum_{m=1}^{M} p(\hat{C}_m) p_{\text{PR}}(C_m) = \sum_{m=1}^{M} \frac{\hat{n}_m}{\hat{N}} p_{\text{PR}}(C_m) \tag{34}$$

where $\hat{N} = \sum_{m=1}^{M} \hat{n}_m$ is the total number of data samples being classified. Therefore, PR, also known as user's accuracy, is proposed to address the BKG issue. While many existing classification methods have removed all unlabeled data samples as BKG from classification, PR is the one that can include all data samples for evaluation.

VII. Experiments and Discussion of Results

Three popular and widely used real hyperspectral images, Purdue's Indiana Indian Pines, Salinas, and University of Pavia, available at the website http://www.ehu.eus/ccwintco/index.

Fig. 3. Purdue's Indiana Indian Pines scene. (a) Ground truth map. (b) Classes by colors.

php?title = Hyperspectral_Remote_Sensing_Scenes, were used for experiments.

Since IRTS can be implemented with any SS classifier, EPF-based methods developed in [2] were particularly selected for experiments based on two main reasons. According to [2], EPF-based methods generally performed well compared to other existing SS methods. The other is that the source codes of EPF-based methods are available on the authors' website [22] so that those who are interested in the work presented in this article can use these codes to compare their results. Although four versions of EPF were proposed in [2], only EPF-G-g was used in this article since EPF-G-g was shown to perform generally better than the other three EPF-based methods. It utilizes a guided filter and first PC as the reference image to perform SS classification. Recently, an iterative version of EPF, called iterative EPF (IEPF) was developed in [17] which has been shown to outperform EPF in classification. So, IEPF was also included for comparison. Interestingly, IRTS-EPF can be considered as a generalization of IEPF, which reselects training samples instead of fixing training samples throughout all iterations to improve the classification performance as well as reduce the inconsistency of classification results.

Three parts of experiments were performed.

1) IEPF developed in [17] has been shown to achieve very high classification accuracy ($P_{\rm AA} = P_{\rm OA} \approx 98.5\%$ for the Purdue data; $P_{\rm AA} \approx 0.99\%$, $P_{\rm OA} \approx 98.5\%$ for University of Pavia) and also perform significantly better than EPF. It was believed that it would be very difficult to further improve its classification performance. So, in the first part, we conducted experiments to show that IRTS-EPF using different sets of training samples at different iterations indeed performed better than IEPF which used the same set of training samples for all iterations.
2) With the second part, we investigated the issue of inconsistent classification caused by uncertainty resulting from the randomness of RTS. In doing so, IRTS was implemented 30 times as 30-fold IRTS to measure SD_$P_{\rm OA}$ to evaluate the inconsistency in classification results produced by six different methods, EPF, IEPF, IRTS-EPF, IRTS-GEPF, IRTS-SVM-Gabor, and IRTS-Gabor-EPF. It turned out that the SD_$P_{\rm OA}$ resulting from classification inconsistency was significantly reduced by IRTS.
3) As the final and third part, we explored the utility of SSD in (19) and SE in (25) in interpretation of misclassification errors. First of all, SSD and SE were calculated as quantitative measures. Second, their results were plotted as maps for visual inspection to show data samples with high SSD and SE, which were indeed the most difficult to be classified. This information can be obtained by K-fold IRTS, not classification maps produced by a single run (i.e., onefold) of any classifiers including EPF, IEPF, IRTS-EPF, IRTS-GEPF, IRTS-SVM-Gabor, and IRTS-Gabor-EPF.

A. *Purdue Indiana Indian Pines*

The first image used for the experiment was Purdue University's Indiana Indian Pines shown in Fig. 3(a). It is a well-known airborne visible/infrared imaging spectrometer (AVIRIS) image scene with a spatial resolution of 20-m per pixel and spectral resolution of 10 nm. Its ground truth of 16 class maps plus BKG is provided in Fig. 3(b). It was recorded in June 1992 with 200 bands without including 20 bands, which are the water absorption bands (bands 104–108 and 150–163, 220). Table I tabulates all the 16 classes of interest along with the number of data samples in each class where four small classes, classes 1, 7, 9, 16, contain less than 100 data samples and three large classes, classes, 2, 11, 14, contain more than 1000 data samples. These seven classes can be used to demonstrate the issue of imbalanced classes in classification performance as discussed in [10].

Following the same parameters used in [2] and [17], the total number of training samples $N^{\rm training} = 1025$ was chosen to be 10% of SR. Also, to compare the performance with the original EPF-based method, the same number of selected training samples was also approximately 1025 with the number of selected training sample numbers for each class tabulated in Table II.

According to the training sample size of each class determined in Table II, we randomly selected training samples for the experiments. Table III tabulates $P_{\rm A}$, $P_{\rm OA}$, and $P_{\rm AA}$ of EPF, IEPF,

TABLE I
CLASS LABELS OF PURDUE INDIANA INDIAN PINES WITH NUMBER OF DATA SAMPLES IN EACH CLASS

class 1 (46)	Alfalfa	class 7 (28)	grass/pasture-mowed	class 13 (205)	wheat
class 2 (1428)	corn-notill	class 8 (478)	hay-windrowed	class 14 (1265)	woods
class 3 (830)	corn-min	class 9 (20)	Oats	class 15 (386)	bldg-grass green-drives
class 4 (237)	corn	class 10 (972)	soybeans-notill	class 16 (93)	stone-steel towers
class 5 (483)	grass/pasture	class 11 (2455)	soybeans-min	class 17 (10249)	BKG
class 6 (730)	grass/trees	class 12 (593)	soybeans-clean		

TABLE II
NUMBER OF TRAINING SAMPLES ALLOCATED INTO EACH CLASS FOR PURDUE INDIANA PINES DATA

Class	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	Total
# of training sample	25	83	78	68	79	78	14	66	10	81	99	73	70	90	65	46	1025

TABLE III
P_{A}, P_{OA}, AND P_{AA} OF EPF, IEPF, IRTS-EPF, IRTS-GEPF, IRTS-SVM-GABOR, AND IRTS-GABOR-EPF FOR PURDUE INDIAN PINES WITH THE HIGHEST ACCURACY BOLDFACED

	EPF		IEPF		IRTS-EPF		IRTS-GEPF		IRTS-SVM-Gabor		IRTS-Gabor-EPF	
Class	$P_A(C_m)$	$P_{PR}(C_m)$	$P_A(C_m)$	$P_{PR}(C_m)$	$P_A(C_m)$	$P_{PR}(C_m)$	$P_A(C_m)$	$P_{PR}(C_m)$	$P_A(C_m)$	$P_{PR}(C_m)$	$P_A(C_m)$	$P_{PR}(C_m)$
1 (alfalfa)	**100**	61.33	97.83	63.38	97.83	62.5	**100**	62.16	**100**	65.71	97.83	**67.16**
2 (Corn_N)	84.24	**70.97**	97.41	69.07	98.18	65.58	99.02	70.63	99.44	37.40	**99.79**	66.68
3 (Corn_M)	89.64	68.57	98.43	74.68	98.43	75.93	**99.88**	73.17	98.55	73.63	99.52	**78.74**
4 (Corn)	98.73	66.29	**100**	72.48	**100**	74.06	**100**	68.10	**100**	70.12	**100**	68.5
5 (Grass_P)	97.10	**32.94**	**99.79**	22.60	**99.79**	28.07	99.59	27.44	99.17	26.96	98.96	24.99
6 (Grass_T)	99.73	43.03	99.59	55.75	99.86	57.22	**100**	**58.31**	**100**	48.03	99.86	50.66
7 (Grass_M)	96.43	57.45	96.43	72.97	96.43	57.45	**100**	54.90	96.43	**81.82**	**100**	60.87
8 (Hay_W)	**100**	71.77	**100**	70.29	**100**	73.77	**100**	73.31	99.37	**79.83**	**100**	72.42
9 (Oats)	**100**	52.63	**100**	43.48	**100**	43.48	**100**	**62.50**	**100**	**58.82**	**100**	41.67
10 (Soybean_N)	98.97	59.60	**99.49**	75.37	95.88	80.76	98.66	77.53	95.99	**86.55**	98.87	81.51
11 (Soybean_M)	90.71	**83.00**	94.99	79.35	**99.80**	74.13	99.27	79.20	98.45	79.80	98.78	80.62
12 (Soybean_C)	99.33	60.91	99.33	49.83	**99.66**	**61.43**	**99.66**	57.16	98.82	59.73	**99.66**	51.53
13 (Wheats)	99.51	74.45	**100**	75.65	99.02	82.86	**100**	85.77	98.05	**90.95**	99.51	86.81
14 (Woods)	98.58	29.70	99.37	34.71	**99.68**	**36.49**	99.53	33.41	98.81	31.43	99.45	32.03
15 (Building)	90.93	8.67	**100**	10.06	**100**	8.76	**100**	9.32	98.96	**16.25**	**100**	10.59
16 (Tower)	**100**	56.02	95.70	48.11	95.70	56.69	**100**	40.26	**100**	**82.30**	98.92	51.4
P_{OA}/P_{AA} (%)	93.88 /96.49		98.05 / 98.65		99.02 / 98.77		**99.45/ 99.73**		98.68/98.88		99.36/99.45	
P_{PR} (%)	45.76		47.80		48.27		**48.48**		48.11		48.43	

IRTS-EPF, IRTS-GEPF, IRTS-SVM-Gabor, and IRTS-Gabor-EPF for the Purdue data. As we can see, IRTS-EPF performed better than their counterparts, EPF and IEPF, with improvements of P_{OA} from 93.88% and 98.05% to 99.02%, and P_{AA} from 96.49% and 98.65% to 98.77%. The highest classification accuracy was produced by IRTS-GEPF with $P_{\mathrm{OA}} = 99.45\%$ and $P_{\mathrm{AA}} = 99.73\%$ followed by IRTS-Gabor-EPF with $P_{\mathrm{OA}} = 99.36\%$ and $P_{\mathrm{AA}} = 99.45\%$.

To further investigate the uncertainly caused by RTS, 30-fold IRTS was performed and their results were averaged along with their corresponding SDs and running times calculated and tabulated in Table IV where the computer environment used was 2.90 GHz Intel corei7 CPU and 8 GB of memory. In addition, their CSD with ACSD/OCSD and CE with ACE/OCE along with their respective running times were also calculated and tabulated in Table V. All the results clearly showed that IRTS considerably reduced the uncertainty caused by RTS. Compared to the results produced by IRTS in Table III, the results by 30-fold IRTS in Table IV were rather stable in terms of SD, specifically, from 1.17%/0.52% down to 0.14% by using EPF. Also, in Table IV, 30-fold IRTS not only reduced the SD of P_{OA} very effectively but also improved P_{OA} significantly, where four 30-fold IRTS methods produced P_{OA} up to 99% better than 30-fold EPF and 30-fold IEPF whose averaged P_{OA} values of 30-fold EPF and 30-fold IEPF were 93.02% and 97.95%. It is interesting to note that 30-fold IRTS-SVM-Gabor produced 98.54% of P_{OA} and 98.98% of P_{AA}, both of which were better than EPF and IEPF but worse than IRTS-EPF and IRTS-GEPF due to the fact that the used SVM was not as effective as EPF. However, the performance of IRTS-SVM-Gabor can be improved by fusing Gabor-filtered classification maps with EPF-filtered classification maps where the averaged P_{OA} and P_{AA} of IRTS-Gabor-EPF were increased to 99.17% and 99.40%, which performed better than IRTS-EPF and IRTS-SVM-Gabor. As for running time, EPF required the least time since it did not perform iterative processes. Among all the remaining classifiers which require iterative processes, the ones using Gabor filters required the most computing times with 2–4 times that required by IRTS-GEPF and IRTS-EPF. This is because Gabor filter uses 2-D Gaussian filters compared to EFP and GEPF which are 1-D

TABLE IV
AVERAGED P_{A}, P_{OA}, ACSD/OCSD, AND ACE/OCE CALCULATED BY 30-FOLD EXPERIMENTS OF PURDUE INDIAN PINES WITH THE HIGHEST ACCURACY BOLDFACED

	30-fold EPF	30-fold IEPF	30-fold IRTS-EPF	30-fold IRTS-GEPF	30-fold IRTS-SVM-Gabor	30-fold IRTS-Gabor-EPF
Class	$P_{\mathrm{A}}(C_m) \pm$ SD $P_{\mathrm{A}}(C_m)$	$P_{\mathrm{A}}(C_m) \pm$ SD $P_{\mathrm{A}}(C_m)$	$P_{\mathrm{A}}(C_m) \pm$ SD $P_{\mathrm{A}}(C_m)$	$P_{\mathrm{A}}(C_m) \pm$ SD $P_{\mathrm{A}}(C_m)$	$P_{\mathrm{A}}(C_m) \pm$ SD $P_{\mathrm{A}}(C_m)$	$P_{\mathrm{A}}(C_m) \pm$ SD $P_{\mathrm{A}}(C_m)$
1 (alfalfa)	99.31 ± 1.03	98.87 ± 1.11	**99.57 ± 0.89**	**99.57 ± 0.89**	98.96 ± 1.11	99.31 ± 1.03
2 (Corn_N)	85.36 ± 2.45	96.32 ± 1.90	98.13 ± **0.44**	**98.80** ± 0.51	98.12 ± 1.12	98.31 ± 0.79
3 (Corn_M)	85.38 ± 7.03	98.41 ± 0.80	99.24 ± 0.45	**99.64 ± 0.30**	98.19 ± 1.13	99.13 ± 0.51
4 (Corn)	99.81 ± 0.37	99.98 ± 0.08	**100 ± 0**	99.98 ± 0.08	99.04 ± 0.81	99.88 ± 0.19
5 (Grass_P)	96.88 ± 1.80	99.05 ± 1.38	99.88 ± **0.16**	**99.92** ± 0.17	99.29 ± 0.81	99.21 ± 0.93
6 (Grass_T)	99.62 ± 0.21	99.52 ± 0.50	99.76 ± **0.16**	**99.93** ± 0.10	99.77 ± 0.32	99.85 ± 0.12
7 (Grass_M)	96.43 ± **0**	96.57 ± 0.71	96.43 ± **0**	**99.86** ± 0.71	99.71 ± 0.99	98.86 ± 1.70
8 (Hay_W)	**100 ± 0**	**100 ± 0**	**100 ± 0**	99.98 ± 0.06	99.82 ± 0.24	**100 ± 0**
9 (Oats)	96.40 ± 6.21	**100 ± 0**	**100 ± 0**	**100 ± 0**	99.20 ± 2.77	**100 ± 0**
10 (Soybean_N)	93.22 ± 6.10	96.48 ± 2.76	98.00 ± 1.15	99.17 ± 0.68	98.35 ±1.03	**99.27 ± 0.56**
11 (Soybean_M)	91.26 ± 3.51	96.49 ± 2.04	98.97 ± 0.53	**99.18** ± 0.65	97.65 ±1.30	98.82 ± **0.46**
12 (Soybean_C)	97.23 ± 1.52	99.11 ± 0.45	99.41 ± 0.23	**99.61 ± 0.15**	99.06 ± 0.59	99.32 ± 0.28
13 (Wheats)	99.53 ± 0.10	99.31 ± 0.28	99.39 ± 0.26	**99.80 ± 0.25**	99.35 ± 0.66	99.61 ± 0.28
14 (Woods)	95.49 ± 2.92	99.59 ± 0.35	**99.70** ± 0.20	99.62 ± 0.37	99.04 ± 0.43	99.57 ± **0.19**
15 (Building)	97.49 ± 4.15	99.98 ± 0.10	**100 ± 0**	**100 ± 0**	98.45 ± 0.90	99.65 ± 0.55
16 (Tower)	99.74 ± 0.78	97.33 ± 1.43	97.46 ± 1.52	**100 ± 0**	99.66 ± 1.51	99.65 ± 0.51
$P_{\mathrm{OA}} \pm$ SD_ P_{OA}	93.02 ± 1.17	97.95 ± 0.52	99.10 ± 0.16	**99.44** ± 0.16	98.54 ± 0.38	99.17 ± **0.14**
$P_{\mathrm{AA}} \pm$ SD_ P_{AA}	95.82 ± 0.70	98.57 ± 0.22	99.12 ± 0.16	**99.69 ± 0.08**	98.98 ± 0.30	99.40 ± 0.15
Iterations	NA	5.52 ± 1.08	8.44 ± 2.32	10.60 ± 2.50	14.68 ± 1.14	11.84 ± 2.53
Time(s)	5.55 ± 0.62	23.25 ± 3.57	57.44 ± 10.56	67.18 ± 14.85	219.197 ± 45.03	123.15 ± 36.37

TABLE V
UNCERTAINTY CAUSED BY USING CSD AND CE FOR 30-FOLD EXPERIMENTS OF PURDUE INDIAN PINES WITH THE LEAST CSD AND CE BOLDFACED

	30-fold EPF		30-fold IEPF		30-fold IRTS-EPF		30-fold IRTS-GEPF		30-fold IRTS-SVM-Gabor		30-fold IRTS-Gabor-EPF	
Class	CSD(C_m)	CE(C_m)	CSD(C_m)	CE(C_m)	CSD(C_m)	CE(C_m)	CSD(C_m)	CE(C_m)	CSD(C_m)	CE(C_m)	CSD(C_m)	CE(C_m)
1 (alfalfa)	0.0103	0.0136	0.0111	0.0151	**0.0089**	**0.0138**	**0.0089**	**0.0138**	0.0111	0.0198	0.0103	0.0182
2 (Corn_N)	0.0942	0.1391	0.0479	0.0625	**0.0127**	**0.0157**	0.0170	0.0196	0.0440	0.0493	0.0212	0.0274
3 (Corn_M)	0.1448	0.1925	0.0234	0.0321	0.0095	0.0123	**0.0062**	**0.0085**	0.0379	0.0428	0.0159	0.0190
4 (Corn)	0.0043	0.0049	0.0008	0.0007	**0**	**0**	0.0008	0.0007	0.0237	0.0287	0.0027	0.0030
5 (Grass_P)	0.0420	0.0656	0.0274	0.0325	**0.0028**	0.0030	0.0030	**0.0029**	0.0163	0.0225	0.0206	0.0254
6 (Grass_T)	0.0048	0.0051	0.0071	0.0079	0.0020	0.0027	**0.0014**	**0.0016**	0.0068	0.0078	0.0019	0.0026
7 (Grass_M)	**0**	**0**	0.0071	0.006	**0**	**0**	0.0071	0.0060	0.0099	0.0100	0.0170	0.0224
8 (Hay_W)	**0**	**0**	**0**	**0**	**0**	**0**	0.0006	0.0006	0.0043	0.0062	**0**	**0**
9 (Oats)	0.0926	0.1055	**0**	**0**	**0**	**0**	**0**	**0**	0.0277	0.0334	**0**	**0**
10 (Soybean_N)	0.1034	0.1568	0.0593	0.0811	0.0253	0.0327	0.0181	0.0192	0.0388	0.0453	**0.0162**	**0.0182**
11 (Soybean_M)	0.1086	0.1624	0.0683	0.0827	0.0161	0.0190	**0.0160**	**0.0187**	0.0449	0.0542	0.0165	0.0219
12 (Soybean_C)	0.0307	0.0385	0.0108	0.0137	0.0044	0.0056	**0.0030**	**0.0038**	0.0135	0.0148	0.0079	0.0098
13 (Wheats)	0.0010	0.0008	0.0047	0.0054	0.0041	**0.0043**	**0.0024**	**0.0043**	0.0095	0.0161	0.0042	0.0045
14 (Woods)	0.0475	0.0557	0.0052	0.0061	**0.0027**	**0.0033**	0.0069	0.0077	0.0148	0.0236	0.0038	0.0055
15 (Building)	0.0735	0.0797	0.001	0.0009	**0**	**0**	**0**	**0**	0.0207	0.0345	0.0072	0.0116
16 (Tower)	0.0089	0.0090	0.0225	0.0281	0.0269	0.0319	**0**	**0**	0.0172	0.0144	0.0051	0.0067
OCSD/ACSD	0.0738/0.0479		0.0341/0.0186		0.0100/0.0072		**0.0098/0.0057**		0.0295/0.0213		0.0123/0.0094	
OCE/ACE	0.1048/0.0643		0.0430/0.0234		0.0123/0.0090		**0.0114/0.0067**		0.0359/0.0265		0.0157/0.0123	

filters. Also, IRTS-GEPF required more time than IRTS-EPF because IRTS-GEPF includes a fusion process that does not exist in IRTS-EPF.

By comparing four IRTS-based classifications, we can see that the smallest SD_P_{OA} was produced by 30-fold IRTS-Gabor-EPF $= 0.14\%$ and the smallest SD_$P_{\mathrm{AA}} = 0.08\%$ was generated by 30-fold IRTS-GEPF. On the other hand, the lowest OCSD/ACSD and OCE/ACE were both produced by 30-fold IRTS-GEPF with OCSD/ACSD $=$ 0.0098/0.0057 and OCE/ACE $=$ 0.0114/0.0067, which is lower than OCE/ACE $=$ 0.1048/0.0643 produced by EPF and OCE/ACE $=$ 0.0430/0.0234 by using IEPF. Interestingly, even though 30-fold IRTS-Gabor-EPF created the smallest SD_P_{OA}, it still produced higher OCSD/ACSD and OCE/ACE than 30-fold IRTS-EPF and 30-fold IRTS-GEPF. This demonstrated that the smaller value of SD_P_{OA} did not necessarily indicate more stable classification results. So, OCSD/ACSD and OCE/ACE are generally needed to evaluate the uncertainty of classification.

In the following, we will demonstrate the advantages of utilizing SSD in (19) and SE in (26) to interpret classification maps. Fig. 4 shows onefold classification results of all the tested methods for the Purdue data where it is difficult to make comparison by visual inspection. However, if we used the results in Tables IV and V to plot their results as SSD and SE maps in

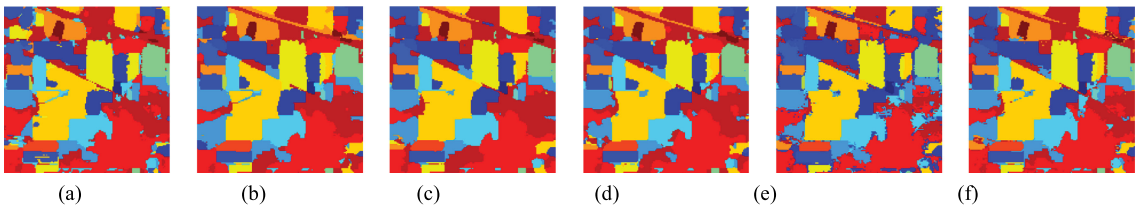


Fig. 4. Classification results of six SS methods for the Purdue data. (a) EPF. (b) IEPF. (c) IRTS-EPF. (d) IRTS-GEPF. (e) IRTS-SVM-Gabor. (f) IRTS-Gabor-EPF.

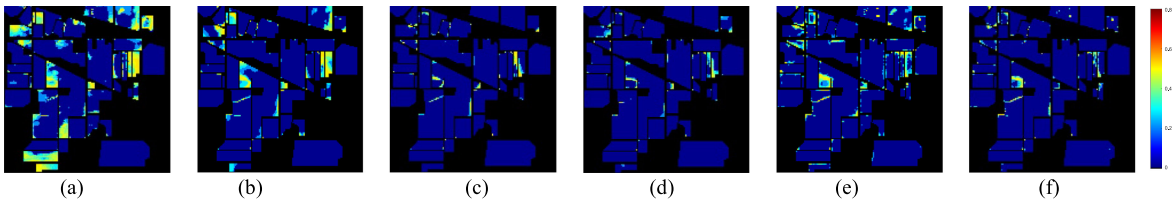


Fig. 5. SSD maps produced by 30-fold experiments along with color bar of SSD maps. (a) EPF. (b) IEPF. (c) IRTS-EPF. (d) IRTS-GEPF. (e) IRTS-SVM-Gabor. (f) IRTS-Gabor-EPF.

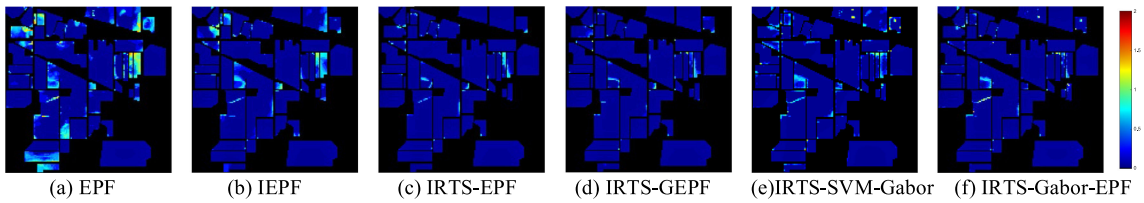


Fig. 6. SE maps produced by 30-fold experiments along with color bar of SE map. (a) EPF. (b) IEPF. (c) IRTS-EPF. (d) IRTS-GEPF. (e) IRTS-SVM-Gabor. (f) IRTS-Gabor-EPF.

Figs. 5 and 6, the benefit of interpreting classification results was immediately obvious. In 30-fold EPF results, most of the data samples in each class had high SSD and SE in Figs. 5 and 6. As for 30-fold IEPF, SSD of most data samples was reduced, but SSD of some regions was actually increased as shown in Figs. 5(b) and 6(b). In other words, 30-fold IEPF could significantly reduce the uncertainty caused by RTS, but it also increased the inconsistency in classification of data samples which were difficult to be classified, specifically class boundary data samples. Compared to Figs. 5(a) and (b) and 6(a) and (b) produced by 30-fold EPF and 30-fold IEPF, Figs. 5(c)–(f) and 6(c)–(f) showed significant reduction of uncertainties of the SSD and SE maps produced by 30-fold IRTS methods. We can also observe that 30-fold IRTS-SVM-Gabor created the most inconsistent classification result among the four IRTS methods but it still performed better than 30-fold EPF and 30-fold IEPF in terms of P_{OA} and P_{AA}. Nevertheless, it was also inevitable that some data samples would still show high uncertainty because they were edge points located along their class boundaries. As a result, these data samples were affected by BKG.

As a final experiment, we demonstrated the differences between various SF-ed classification maps and their corresponding MAP-threshold binary maps. To further illustrate the imbalance class issue, we specifically picked two particular classes—class 7 which has 28 data samples to represent a small class and class 11 which has 2455 data samples to represent a large class, both of which did not do as well as the other classes with their corresponding comparable sizes. Figs. 7 and 8 compare the differences between various SF-ed maps and their corresponding MAP-thesholded binary maps for class 7 and class 11, respectively, where (a) EPF ($P_{\mathrm{A}} = 96.43\%$), (b) IEPF ($P_{\mathrm{A}} = 96.43\%$), (c) IRTS-EPF ($P_{\mathrm{A}} = 96.43\%$), (d) IRTS-GEPF ($P_{\mathrm{A}} = 100\%$), (e) IRTS-SVM-Gabor ($P_{\mathrm{A}} = 100\%$), and (f) IRTS-Gabor-EPF ($P_{\mathrm{A}} = 100\%$)for class 7 and (a) EPF ($P_{\mathrm{A}} = 93.65\%$), (b) IEPF ($P_{\mathrm{A}} = 95.40\%$), (c) IRTS-EPF ($P_{\mathrm{A}} = 98.82\%$), (d) IRTS-GEPF ($P_{\mathrm{A}} = 99.80\%$), (e) IRTS-SVM-Gabor ($P_{\mathrm{A}} = 98.45\%$), and (f) IRTS-Gabor-EPF ($P_{\mathrm{A}} = 99.19\%$) for class 11. As shown in these figures, IRTS significantly improved the classification accuracy P_{A} of both classes regardless of their class sizes.

B. Salinas

A second real hyperspectral image scene is Salinas shown in Fig. 9(a) along with Fig. 9(b) and (c) which shows the color composite of the Salinas image and its corresponding ground truth map in Fig. 9(b) by color class labels in Fig. 9(c). It is also an AVIRIS image collected over Salinas Valley, California, and with a spatial resolution of 3.7-m per pixel with spectral resolution of 10 nm. The image cube has size of $512 \times 217 \times$

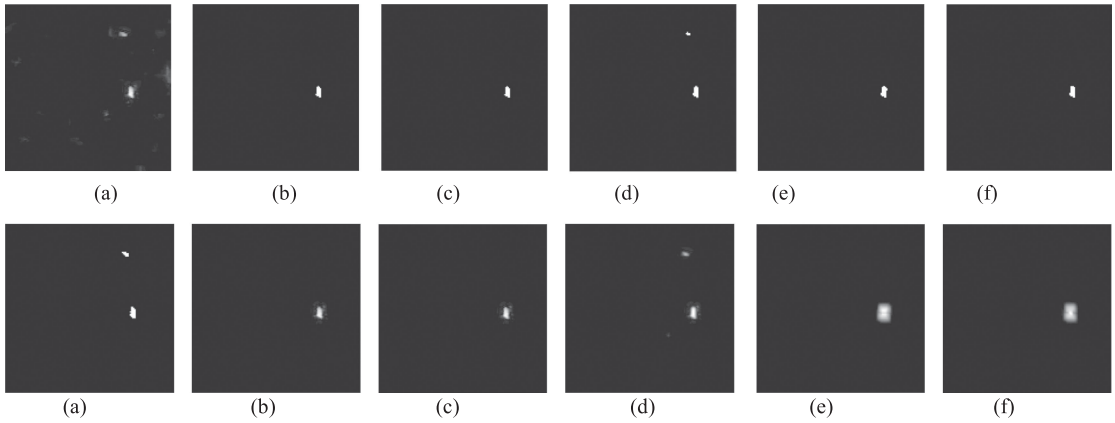


Fig. 7. Various SF-ed classification maps for class 7 in the first row and the corresponding binary classification maps in the second row. (a) EPF ($P_{\mathrm{A}} = 96.43\%$). (b) IEPF ($P_{\mathrm{A}} = 96.43\%$). (c) IRTS-EPF ($P_{\mathrm{A}} = 96.4\%3$). (d) IRTS-GEPF ($P_{\mathrm{A}} = 100\%$). (e) IRTS-SVM-Gabor ($P_{\mathrm{A}} = 100\%$). (f) IRTS-Gabor-EPF ($P_{\mathrm{A}} = 100\%$).

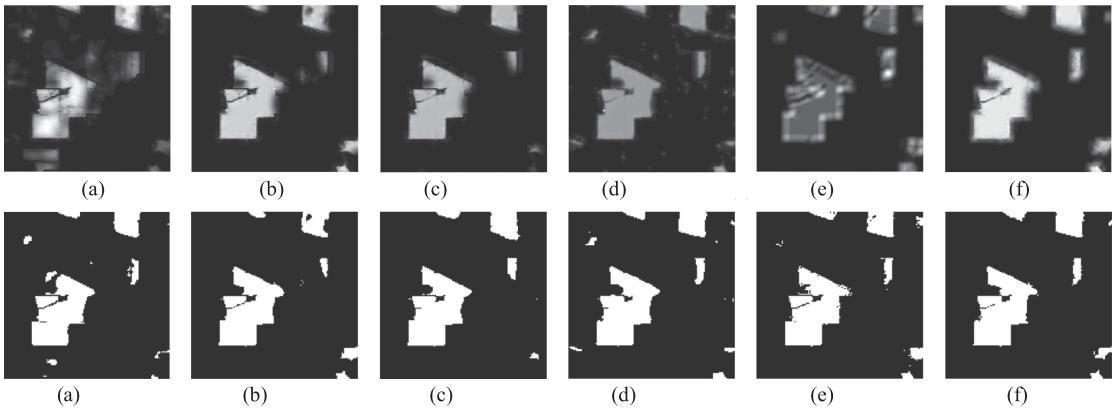


Fig. 8. Various SF-ed classification maps for class 11 in the first row and the corresponding binary classification maps in the second row. (a) EPF ($P_{\mathrm{A}} = 93.65\%$). (b) IEPF ($P_{\mathrm{A}} = 95.40\%$). (c) IRTS-EPF ($P_{\mathrm{A}} = 98.82\%$). (d) IRTS-GEPF ($P_{\mathrm{A}} = 99.80\%$). (e) IRTS-SVM-Gabor ($P_{\mathrm{A}} = 98.45\%$). (f) IRTS-Gabor-EPF ($P_{\mathrm{A}} = 99.19\%$).

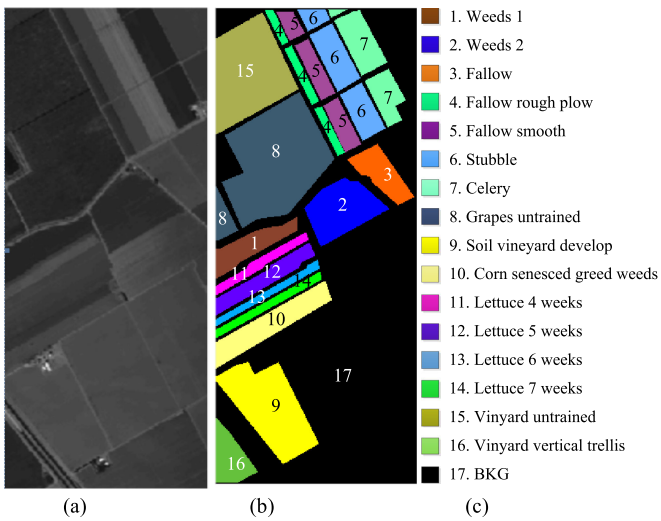


Fig. 9. Ground truth of Salinas scene with 16 classes. (a) Salinas scene. (b) Ground-truth image. (c) Classes by colors.

224. This scene is very similar to the Purdue Indiana Indian Pines scene which also excluded 20 water absorption bands which are 104–108, 150–163, and 220. Table VI tabulates the number of data samples in parentheses collected for each class.

Unlike the Purdue data, all the classes are relatively large and have over 900 data samples. So, there is no imbalanced class issue in this dataset. In Salinas experiments, the sample ratio to select training samples was set to 2% as was done in [2] where the training sample size for each class was nearly equal and around 67–70, as tabulated in Table VII.

According to the training sample size of each class determined in Table VII, we randomly selected training samples for each class and performed EPF, IEPF, IRTS-EPF, IRTS-GEPF, IRTS-SVM-Gabor, and IRTS-Gabor-EPF for Salinas data. Table VIII tabulates their P_{A}, P_{OA}, and P_{AA} values. As we can see from Table VIII, IRTS-EPF and IRTS-GEPF performed better than their counterparts EPF and IEPF, with improvements of P_{AA} by 0.2% (IEPF) and 2% (EPF) and P_{OA} by 1% (IEPF) and 5% (EPF). It should be noted that since P_{OA} and P_{AA} of IEPF were

TABLE VI
CLASS LABELS OF SALINAS WITH NUMBER OF DATA SAMPLES IN EACH CLASS

class 1 (2009)	Brocoli_green_weeds_1	class 10 (3278)	Corn_senesced_green_weeds
class 2 (3726)	Brocoli_green_weeds_2	class 11 (1068)	Lettuce_romaine_4wk
class 3 (1976)	Fallow	class 12 (1927)	Lettuce_romaine_5wk
class 4 (1394)	Fallow_rough_plow	class 13 (916)	Lettuce_romaine_6wk
class 5 (2678)	Fallow_smooth	class 14 (1070)	Lettuce_romaine_7wk
class 6 (3959)	Stubble	Class 15 (7268)	Vinyard_untrained
class 7 (3579)	Celery	class 16 (1807)	Vinyard_vertical_trellis
class 8 (11271)	Grapes_untrained	class 17 (56975)	BKG
class 9 (6203)	Soil_vinyard_develop		

TABLE VII
NUMBER OF TRAINING SAMPLES ALLOCATED INTO EACH CLASS FOR SALINAS

Class	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	Total
# of training Sample	67	67	67	69	67	67	68	69	68	68	68	67	67	67	70	67	1083

TABLE VIII
P_{A}, P_{OA}, AND P_{AA} OF EPF, IEPF, IRTS-EPF, IRTS-GEPF, IRTS-SVM-GABOR, AND IRTS-GABOR-EPF FOR SALINAS WITH THE HIGHEST ACCURACY BOLDFACED

	EPF		IEPF		IRTS-EPF		IRTS-GEPF		IRTS-SVM-Gabor		IRTS-Gabor-EPF	
Class	$P_{\mathrm{A}}(C_m)$	$P_{\mathrm{PR}}(C_m)$	$P_{\mathrm{A}}(C_m)$	$P_{\mathrm{PR}}(C_m)$	$P_{\mathrm{A}}(C_m)$	$P_{\mathrm{PR}}(C_m)$	$P_{\mathrm{A}}(C_m)$	$P_{\mathrm{PR}}(C_m)$	$P_{\mathrm{A}}(C_m)$	$P_{\mathrm{PR}}(C_m)$	$P_{\mathrm{A}}(C_m)$	$P_{\mathrm{PR}}(C_m)$
1(Weeds_1)	**100**	73.64	99.95	76.58	100	75.24	100	**79.79**	99.45	89.4	99.95	79.68
2(Weeds 2)	**100**	67.44	100	77.21	99.97	78.21	99.95	78.25	98.71	**81.57**	99.60	76.47
3(Fallow)	**100**	11.95	99.90	13.63	99.75	12.68	100	**17.45**	99.8	14.22	99.90	12.56
4(Fallow_P)	**99.86**	27.22	99.35	35.40	99.64	34.07	99.78	35.44	99.57	**36.57**	**99.86**	34.41
5(Fallow_S)	98.58	**67.69**	98.73	57.78	98.77	61.09	98.84	63.37	**99.37**	50.58	98.17	64.61
6(Stubble)	**100**	81.95	99.97	82.68	99.90	83.74	99.90	84.42	99.37	**87.19**	98.94	86.87
7(Celery)	**99.89**	81.51	99.83	82.04	99.53	83.22	99.39	84.63	98.99	87.74	99.08	**88.87**
8(Grapes)	93.27	82.05	95.19	93.09	98.57	94.03	98.00	61.21	95.13	92.55	98.53	**95.06**
9(Soil)	**99.89**	34.59	98.71	36.88	99.31	38.90	99.61	39.59	99.19	41.85	99.45	**40.31**
10(Corn)	98.63	**23.28**	98.72	16.91	**100**	14.53	99.97	15.98	99.85	13.02	**100**	14.83
11(Lettuce 4)	**99.72**	21.53	99.16	**27.93**	98.78	21.18	98.78	21.80	**99.72**	21.68	**99.72**	19.16
12(Lettuce 5)	**100**	31.21	**100**	29.61	**100**	**64.13**	**100**	49.92	**100**	45.16	**100**	44.93
13(Lettuce_6)	99.13	66.86	99.24	66.11	99.13	66.33	**99.67**	67.68	97.16	**69.48**	99.56	66.04
14(Lettuce 7)	**100**	62.83	**100**	62.1	97.57	69.14	99.91	66.60	98.50	66.16	98.32	**69.72**
15(Vinyard_U)	82.31	89.19	96.64	89.22	99.49	95.57	**99.79**	93.67	97.19	95.29	98.58	**97.10**
16(Vinyard_T)	99.11	69.26	98.89	59.33	98.95	86.42	97.95	**90.26**	98.67	88.84	**99.56**	82.00
$P_{\mathrm{OA}}/P_{\mathrm{AA}}$ (%)	96.00/98.15		98.16/99.02		**99.31/99.34**		99.29/99.47		98.11/98.79		99.11/99.33	
P_{PR} (%)	46.77		47.82		**48.39**		48.37		47.80		48.29	

98.16% and 99.02%, which were already very high, to improve IEPF would be very difficult, even by a small percentage. IRST-EPF managed to improve IEPF by 1% for P_{OA} and 0.3% for P_{AA}, which were considered to be significant.

To calculate SD, CSD, and CE along with their OCSD/ACSD and OCE/ACE we performed 30 folds for EPF, IEPF, IRTS-EPF,IRTS-GEPF, IRTS-SVM-Gabor, and IRTS-Gabor-EPF. Table IX tabulates the averaged P_{A} and P_{OA} produced by 30-fold EPF, 30-fold IEPF, 30-fold IRTS-EPF, 30-fold IRTS-GEPF, 30-fold IRTS-SVM-Gabor, and 30-fold Gabor-EPF along with their respective SD and running time at the last row where 30-fold IEPF improved 30-fold EPF by 2.6% from 96.02% to 98.67% and 30-fold IRTS-EPF improved P_{OA} of 30-fold IEPF from 98.67% to 99.28% and P_{AA} from 99.21% to 99.44%. So, once again, the results produced by using 30-fold IRTS were always better than their counterparts without IRTS in terms of P_{OA} and SD_P_{OA}. Apparently, 30-fold EPF was the worst one with SD_$P_{\mathrm{OA}} = 1.11\%$ and 30-fold IRTS-EPF was the best with SD_$P_{\mathrm{OA}} = 0.14\%$ which was eight times better than the worst one and 30-fold IEPF was somewhere in between. Once again, the ones using Gabor filters required the most computing times with about three times that required by IRTS-EPF and IRTS-GEPF.

Table X also tabulates CSD, CE, OCSD/ACSD, and OCE/ACE according to the results in Table IX. Once again, the most stable classifier is IRTS-EPF followed by IRTS-GEPF and IRTS-Gabor-EPF where the OCSD and OCE of 30-fold IRTS-EPF were 0.0056 and 0.0064 which is nearly ten times better than EPF with OCSD $= 0.0479$ and OCE $= 0.0567$.

Fig. 10 shows onefold classification results of EPF, IEPF, IRTS-EPF, IRTS-GEPF, IRTS-SVM-Gabor, and IRTS-Gabor-EPF for Salinas for visual inspection where the misclassified data samples could be identified by the ground truth. However, if we used the results in Tables IX and X to calculate SSD in (19) and SE in (26) and plotted their results as SSD and SE maps, the uncertainty of SSD and SE caused by these misclassified

TABLE IX
AVERAGED SD OF P_A AND P_{OA} CALCULATED BY 30-FOLD EXPERIMENTS OF SALINAS DATA WITH THE HIGHEST ACCURACY AND SMALLEST SD BOLDFACED

	30-fold EPF	30-fold IEPF	30-fold IRTS-EPF	30-fold IRTS-GEPF	30-fold IRTS-SVM-Gabor	30-fold IRTS-Gabor-EPF
Class	$P_A(C_m) \pm SD$ $P_A(C_m)$	$P_A(C_m) \pm SD$ $P_A(C_m)$	$P_A(C_m) \pm SD$ $P_A(C_m)$	$P_A(C_m) \pm SD$ $P_A(C_m)$	$P_A(C_m) \pm SD$ $P_A(C_m)$	$P_A(C_m) \pm SD$ $P_A(C_m)$
1(Weeds_1)	**100 ± 0**	99.99 ± 0.03	**100 ± 0**	99.84 ± 0.27	99.77 ± 0.22	99.92 ± 0.04
2(Weeds_2)	**100 ± 0**	99.99 ± 0.02	99.98 ± 0.03	99.50 ± 0.37	99.01 ± 1.12	99.43 ± 0.23
3(Fallow)	**99.98 ± 0.04**	99.83 ± 0.20	99.83 ± 0.08	99.83 ± 0.21	99.67 ± 0.31	99.81 ± 0.06
4(Fallow P)	**99.88 ± 0.05**	99.65 ± 0.25	99.77 ± 0.13	99.54 ± 0.32	99.53 ± 0.58	99.73 ± 0.30
5(Fallow_S)	98.73 ± 0.16	98.84 ± 0.20	98.77 ± **0.12**	98.83 ± 0.43	98.52 ± 0.65	**98.97** ± 0.42
6(Stubble)	**99.99 ± 0.02**	99.94 ± 0.10	99.97 ± 0.04	99.85 ± 0.09	99.19 ± 0.41	99.27 ± 0.72
7(Celery)	**99.90 ± 0.05**	99.72 ± 0.11	99.60 ± 0.08	99.49 ± 0.18	99.22 ± 0.44	99.35 ± 0.21
8(Grapes)	90.40 ± 4.26	96.20 ± 2.12	98.31 ± **0.52**	**98.37** ± 1.08	96.43 ± 1.26	97.91 ± 0.62
9(Soil)	**99.76 ± 0.14**	99.19 ± 0.57	99.48 ± 0.24	99.34 ± 0.37	98.97 ± 0.49	99.13 ± 0.42
10(Corn)	98.58 ± 0.81	99.73 ± 0.34	99.93 ± 0.11	**99.98 ± 0.06**	99.61 ± 0.53	99.55 ± 0.46
11(Lettuce 4)	**99.82 ± 0.13**	99.31± 0.36	99.09 ± 0.28	99.00 ± 0.42	98.99 ± 0.87	99.62 ± 0.31
12(Lettuce_5)	**100 ± 0**	**100 ± 0**	**100 ± 0**	99.94 ± 0.12	99.87 ± 0.21	**100 ± 0**
13(Lettuce 6)	99.16 ± **0.17**	99.23 ± 0.43	99.45 ± 0.41	**99.75** ± 0.27	98.94 ± 0.82	99.24 ± 0.74
14(Lettuce 7)	98.87 ± 1.08	98.87 ± 0.96	99.03 ± 0.68	**99.23 ± 0.42**	98.78 ± 0.69	98.60 ± 0.87
15(Vinyard U)	87.12 ± 5.61	98.18 ± 1.97	**99.15** ± 0.59	98.83 ± 0.81	97.77 ± 0.82	98.72 ± **0.54**
16(Vinyard T)	**99.24** ± 0.32	98.67 ± 0.65	98.66 ± **0.25**	98.34 ± 0.59	98.41 ± 1.09	98.88 ±0.45
$P_{OA} \pm SD\_P_{OA}$	96.02 ± 1.11	98.67 ± 0.61	**99.28 ± 0.14**	99.17 ± 0.20	98.41 ± 0.27	98.98 ± 0.15
$P_{AA} \pm SD$ P_{AA}	98.21 ± 0.42	99.21 ± 0.25	**99.44 ± 0.06**	99.36 ± 0.12	98.92 ± 0.22	99.26 ± 0.12
Iterations	Na	4.84 ± 0.98	6.56 ± 0.92	11.72 ± 2.57	10.56 ± 2.20	13.40 ± 1.66
Time(s)	17.18 ± 5.50	104.20 ± 36.44	167.42 ± 41.39	194.27 ± 51.80	460.75 ± 128.54	560.28 ± 203.54

TABLE X
UNCERTAINTY OF CSD AND CE FOR 30-FOLD EXPERIMENTS FOR SALINAS DATA WITH THE LEAST CSD AND CE BOLDFACED

	30-fold EPF		30-fold IEPF		30-fold IRTS-EPF		30-fold IRTS-GEPF		30-fold IRTS-SVM-Gabor		30-fold IRTS-Gabor-EPF	
Class	$CSD(C_m)$	$CE(C_m)$	$CSD(C_m)$	$CE(C_m)$	$CSD(C_m)$	$CE(C_m)$	$CSD(C_m)$	$CE(C_m)$	$CSD(C_m)$	$CE(C_m)$	$CSD(C_m)$	$CE(C_m)$
1 (alfalfa)	**0**	**0**	0.0003	0.0003	**0**	**0**	0.0036	0.0045	0.0038	0.0043	0.0005	0.0008
2 (Corn_N)	**0**	**0**	0.0002	0.0002	0.0003	0.0004	0.0056	0.0098	0.0175	0.0192	0.0035	0.0057
3 (Corn_M)	**0.0005**	**0.0005**	0.0027	0.0028	0.0009	0.0012	0.0031	0.0044	0.0066	0.0068	0.0009	0.0012
4 (Corn)	**0.0005**	**0.0005**	0.0037	0.0052	0.0015	0.0019	0.0046	0.0082	0.0091	0.0102	0.0060	0.0065
5 (Grass_P)	0.0035	0.0046	0.0040	0.0050	**0.0026**	**0.0030**	0.0088	0.0135	0.0126	0.0159	0.0072	0.0094
6 (Grass_T)	**0.0003**	**0.0003**	0.0013	0.0016	0.0005	0.0006	0.0016	0.0029	0.0095	0.011	0.0106	0.0121
7 (Grass_M)	**0.0010**	**0.0021**	0.0017	0.0038	0.0011	0.0014	0.0034	0.0070	0.0072	0.0106	0.0032	0.0080
8 (Hay_W)	0.1085	0.1198	0.0623	0.0706	**0.0116**	**0.0131**	0.0237	0.0277	0.0418	0.0503	0.0234	0.0277
9 (Oats)	**0.0025**	**0.0029**	0.0101	0.0126	0.0036	0.0043	0.0065	0.0100	0.0110	0.0124	0.0091	0.0108
10 (Soybean_N)	0.0197	0.0228	0.0073	0.0075	0.0019	0.0021	**0.0009**	**0.0008**	0.0122	0.0123	0.0114	0.0135
11 (Soybean M)	**0.0016**	**0.0027**	0.0053	0.0084	0.0040	0.0046	0.0061	0.0124	0.0164	0.0191	0.0046	0.0063
12 (Soybean_C)	**0**	**0**	**0**	**0**	**0**	**0**	0.0015	0.0020	0.0030	0.0034	**0**	**0**
13 (Wheats)	**0.0029**	**0.0034**	0.0060	0.0073	0.0054	0.0067	0.0038	0.0067	0.0133	0.0174	0.0095	0.0121
14 (Woods)	0.0189	0.0220	0.0156	0.0188	0.0100	0.0118	**0.0062**	**0.0099**	0.0161	0.0199	0.0196	0.0227
15 (Building)	0.1706	0.2147	0.0377	0.0421	**0.0134**	**0.0152**	0.0213	0.0276	0.0333	0.0424	0.0171	0.0233
16 (Tower)	0.0055	0.0077	0.0095	0.0166	**0.0040**	**0.0045**	0.0102	0.0204	0.0183	0.0226	0.0093	0.0136
OCSD/ACSD	0.0479/0.0210		0.0211/0.0105		**0.0056/0.0038**		0.0108/0.0069		0.0204/0.0145		0.0116/0.0085	
OCE/ACE	0.0567/0.0252		0.0243/0.0127		**0.0064/0.0044**		0.0145/0.0105		0.0246/0.0174		0.0147/0.0109	

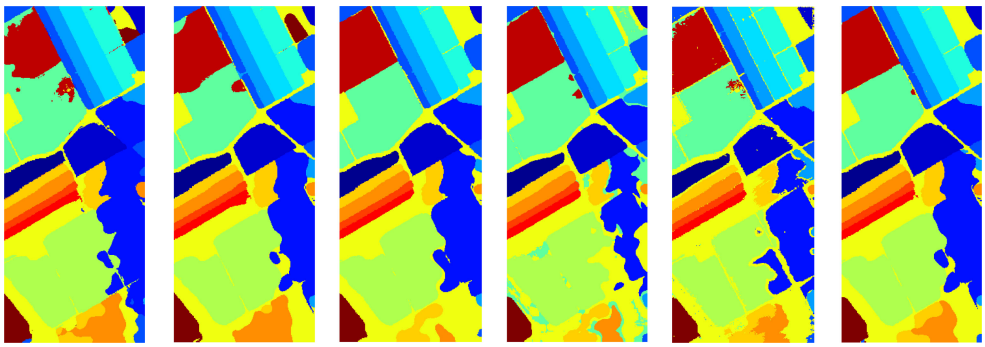


Fig. 10. Classification results of six SS methods for Salinas. (a) EPF. (b) IEPF. (c) IRTS-EPF. (d) IRTS-GEPF. (e) IRTS-SVM-Gabor. (f) IRTS-Gabor-EPF.

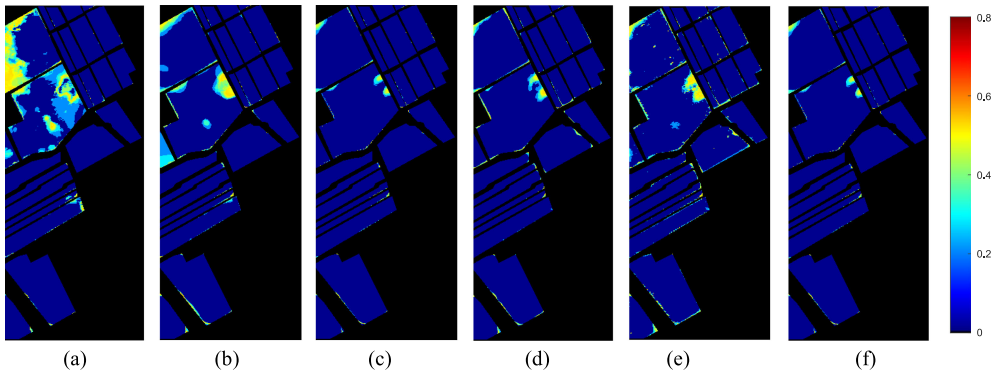


Fig. 11. SSD maps produced by 30-fold experiments along with color bar of SSD maps. (a) EPF. (b) IEPF. (c) IRTS-EPF. (d) IRTS-GEPF. (e) IRTS-SVM-Gabor. (f) IRTS-Gabor-EPF.

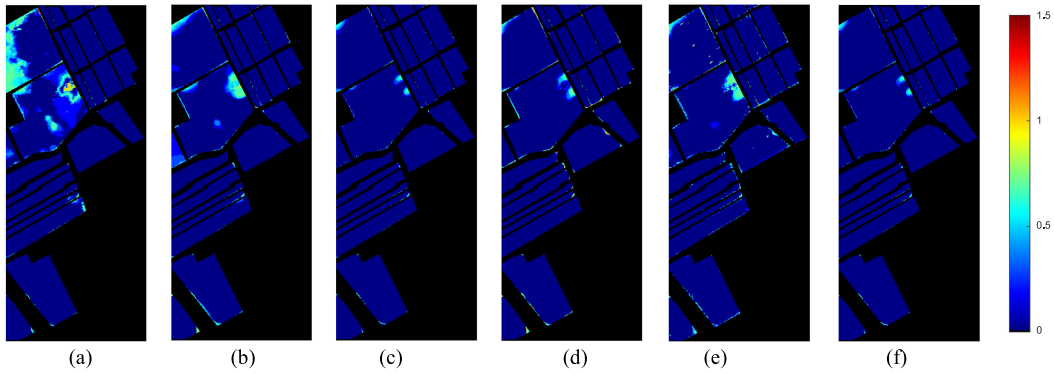


Fig. 12. SE maps produced by 30-fold experiments along with color bar of SE maps. (a) EPF. (b) IEPF. (c) IRTS-EPF. (d) IRTS-GEPF. (e) IRTS-SVM-Gabor. (f) IRTS-Gabor-EPF.

data samples was better characterized by Figs. 11 and 12. By visual inspection, it is clear to note that EPF was the worst one and IRTS-EPF, IRTS-GEPF, and IRTS-Gabor-EPF clearly outperformed EPF and IEPF with much smaller SSD and SE. Specifically, IRTS-EPF seemed to do better than the other three IRTS-based classifications.

Since Salinas has only large classes, the differences between the SF-ed classification maps and their corresponding MAP-threshold binary maps were very similar to that obtained from class 11 of the Purdue data. In this case, due to limited space and also to avoid duplication, their experiments are not included.

C. University of Pavia

A third hyperspectral image data shown in Fig. 13(a) was recorded by the ROSIS-03 satellite sensor over an urban area surrounding the University of Pavia, Italy. It is of size $610 \times 340 \times 115$ with a spatial resolution of 1.3-m per pixel and a spectral coverage ranging from 0.43 to 0.86 μm with spectral resolution of 4 nm (12 most noisy channels were removed before experiments). Nine classes of interest are considered for this image. Fig. 13(c) provides its ground-truth map along with color class labels in Fig. 13(b). Table X also tabulates the number of data samples in parentheses collected for each class where all

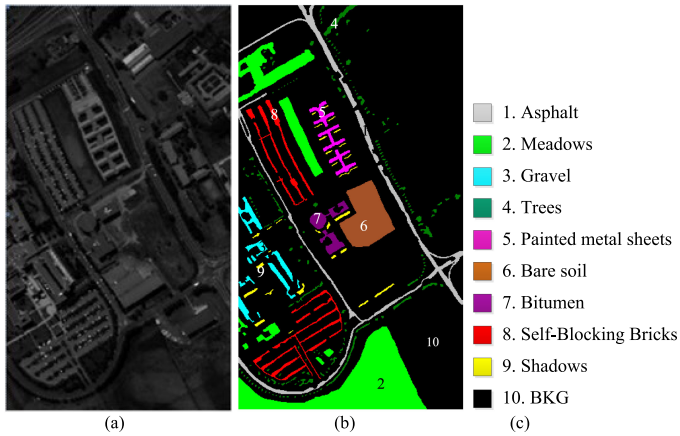


Fig. 13. Ground truth of the University of Pavia scene with nine classes. (a) University of Pavia scene. (b) Ground-truth map. (c) Classes by colors.

the classes have more than 1000 data samples except one class, class 9, which has 947 data samples. Like Salinas there is no imbalanced class issue for this scene. It should be noted that [2] used 6% of SR to select training samples which were too many according to our experiments. So, the training sample size for each class was set to 100 as was done in [14] and listed in Table XI with a total number of 900 training samples with

TABLE XI
CLASS LABELS OF THE UNIVERSITY OF PAVIA WITH NUMBER OF DATA SAMPLES IN EACH CLASS

class 1 (6631)	Asphalt	class 5 (1345)	Painted metal sheets	Class 9 (947)	Shadows
class 2 (18649)	Meadows	class 6 (5029)	Bare Soil	Class 10 (164624)	BKG
class 3 (2099)	Gravel	class 7 (1330)	Bitumen		
class 4 (3064)	Trees	class 8 (3682)	Self-Blocking Bricks		

TABLE XII
NUMBER OF TRAINING SAMPLES ALLOCATED INTO EACH CLASS FOR THE UNIVERSITY OF PAVIA

Class	1	2	3	4	5	6	7	8	9	Total
# of training Sample	100	100	100	100	100	100	100	100	100	900

TABLE XIII
P_{A}, P_{OA}, AND P_{AA} OF EPF, IEPF, IRTS-EPF, AND IRTS-GEPF FOR THE UNIVERSITY OF PAVIA WITH THE HIGHEST ACCURACY BOLDFACED

	EPF		IEPF		IRTS-EPF		IRTS-GEPF		IRTS-SVM-Gabor		IRTS- Gabor-EPF	
Class	$P_A(C_m)$	$P_{PR}(C_m)$	$P_A(C_m)$	$P_{PR}(C_m)$	$P_A(C_m)$	$P_{PR}(C_m)$	$P_A(C_m)$	$P_{PR}(C_m)$	$P_A(C_m)$	$P_{PR}(C_m)$	$P_A(C_m)$	$P_{PR}(C_m)$
1 (alfalfa)	94.54	18.84	99.08	17.1	**99.58**	17.4	99.55	16.45	98.72	**21.69**	99.03	18.04
2 (Corn_N)	98.37	32.24	97.69	**45.99**	99.68	38.84	99.47	39.67	99.08	40.39	**99.92**	36.58
3 (Corn_M)	91.62	32.95	97.38	34.98	99.05	33.29	99.33	35.45	**100**	20.53	99.90	**35.20**
4 (Corn)	90.96	**12.30**	97.45	5.9	**96.61**	7.59	98.76	8.37	99.38	4.89	98.76	7.60
5 (Grass_P)	**100**	44.68	99.85	43.18	**100**	**49.58**	**100**	45.45	99.93	26.91	99.93	33.21
6 (Grass_T)	98.29	9.78	**100**	12.56	**100**	12.51	99.98	11.26	**100**	**27.97**	99.98	13.71
7 (Grass_M)	99.10	**42.61**	**100**	38.57	**100**	38.83	**100**	41.54	**100**	29.60	99.92	30.86
8 (Hay_W)	97.66	18.48	98.97	18.91	99.46	19.24	99.54	20.06	**99.95**	18.23	99.59	**20.96**
9 (Oats)	**100**	7.49	99.26	**13.39**	**100**	8.61	**100**	10.07	**100**	8.40	**100**	8.04
P_{OA}/P_{AA} (%)	96.95/96.73		98.43/98.85		99.46/99.38		99.54/99.63		99.35/**99.67**		**99.68**/**99.67**	

TABLE XIV
AVERAGED SD OF P_{A} AND P_{OA} CALCULATED BY 30-FOLD EXPERIMENTS OF THE UNIVERSITY OF PAVIA WITH THE HIGHEST ACCURACY AND SMALLEST SD BOLDFACED

	30-fold EPF	30-fold IEPF	30-fold IRTS-EPF	30-fold IRTS-GEPF	30-fold IRTS-SVM-Gabor	30-fold IRTS-Gabor-EPF
Class	$P_A(C_m) \pm$ SD $P_A(C_m)$	$P_A(C_m) \pm$ SD $P_A(C_m)$	$P_A(C_m) \pm$ SD $P_A(C_m)$	$P_A(C_m) \pm$ SD $P_A(C_m)$	$P_A(C_m) \pm$ SD $P_A(C_m)$	$P_A(C_m) \pm$ SD $P_A(C_m)$
1 (alfalfa)	94.44 ± 2.82	98.31 ± 1.01	**99.63 ± 0.30**	99.51 ± 0.38	98.80 ± 0.72	99.37 ± 0.33
2 (Corn N)	96.77 ± 2.07	97.81 ± 1.54	99.46 ± 0.41	99.47 ± 0.33	99.09 ± 0.61	**99.66 ± 0.26**
3 (Corn M)	93.77 ± 2.45	98.37 ± 1.59	99.65 ± 0.54	99.35 ± 0.45	99.84 ± 0.11	**99.89 ± 0.07**
4 (Corn)	93.11 ± 1.51	96.40 ± 1.25	97.16 ± 0.70	99.02 ± **0.38**	**99.57** ± 0.54	99.10 ± 0.41
5 (Grass_P)	99.99 ± 0.02	99.96 ± 0.06	**100 ± 0**	99.99 ± 0.03	99.89 ± 0.12	99.93 ± 0.08
6 (Grass_T)	98.82 ± 1.86	99.53 ± 0.61	**100 ± 0**	99.91 ± 0.21	99.83 ± 0.31	99.85 ± 0.27
7 (Grass M)	99.56 ± 0.51	**100 ± 0**	**100 ± 0**	99.97 ± 0.06	99.94 ± 0.05	99.92 ± **0**
8 (Hay_W)	97.44 ± 1.71	98.87 ± 1.03	99.48 ± 0.36	99.23 ± 0.41	99.72 ± 0.19	**99.76 ± 0.11**
9 (Oats)	100 ± 0	99.96 ± 0.15	99.98 ± 0.04	99.96 ± 0.06	**100 ± 0**	99.99 ± 0.04
$P_{OA} \pm$ SD_ P_{OA}	96.56 ± 0.91	98.29 ± 0.64	99.44 ± 0.20	99.51 ± 0.18	99.33 ± 0.27	**99.64 ± 0.12**
$P_{AA} \pm$ SD_ P_{AA}	97.10 ± 0.53	98.80 ± 0.31	99.48 ± 0.13	99.60 ± 0.11	99.63 ± 0.11	**99.72 ± 0.08**
Iterations	Na	4.28 ± 0.61	7.20 ± 1.82	8.48 ± 4.55	13.72 ± 2.41	11.52 ± 2.67
Time(s)	16.65± 1.75	50.82 ± 0.2931	75.34 ±12.04	118.23 ± 33.96	412.65±119.54	270.60 ± 65.84

approximately 2% of SR, which was the same as that used for Salinas.

Now, we randomly selected training samples according to the training sample size of each class determined in Table XII and calculated P_{A}, P_{OA}, and P_{AA} of EPF, IEPF, IRTS-EPF, IRTS-GEPF, IRTS-SVM-Gabor, and IRTS-Gabor-EPF for the University of Pavia. The results are tabulated in Table XIII which showed that IRTS-EPF performed better than its counterparts, EPF and IEPF, with improvements from 0.5% (IEPF) and 2.6% (EPF) in P_{AA} and from 1% (IEPF) and 2.5% (EPF) in P_{OA}. Like Salinas, it should be noted that since the P_{OA} and P_{AA} of IEPF were 98.43% and 98.85%, which were already very high, to improve them would be very difficult. The fact that IRTS-EPF managed to improve IEPF by 1% for P_{OA} and 0.5% for P_{AA} was significant and the best IRTS result was produced by IRTS-Gabor-EPF with $P_{\mathrm{OA}} = 99.68\%$ and $P_{\mathrm{AA}} = 99.67\%$.

To calculate SD, we performed 30 folds for EPF, IEPF, IRTS-EPF, IRTS-GEPF, IRTS-SVM-Gabor, and IRTS-Gabor-EPF. Table XIV also tabulates the average and SD of P_{A} and P_{OA} for 30-fold EPF-, 30-fold IEPF-, and 30-fold IRTS-based classifiers along with their running times. Once again, the results produced by using IRTS were always better than their counterparts without suing IRTS, where 30-fold EPF was the worst one and 30-fold IRTS clearly outperformed 30-fold EPF and 30-fold IEPF with

TABLE XV
UNCERTAINTY MEASUREMENTS OF CSD AND CE FOR 30-FOLD EXPERIMENTS FOR THE UNIVERSITY OF PAVIA WITH THE SMALLEST CSD AND CE BOLDFACED

	30-fold EPF		30-fold IEPF		30-fold IRTS-EPF		30-fold IRTS-GEPF		30-fold IRTS-SVM-Gabor		30-fold IRTS-Gabor-EPF	
Class	CSD(C_m)	CE(C_m)	CSD(C_m)	CE(C_m)	CSD(C_m)	CE(C_m)	CSD(C_m)	CE(C_m)	CSD(C_m)	CE(C_m)	CSD(C_m)	CE(C_m)
1 (alfalfa)	0.0645	0.0807	0.0312	0.0362	**0.0070**	0.0083	0.0083	0.0070	0.0168	0.0212	0.0091	**0.0129**
2 (Corn_N)	0.0667	0.0727	0.0543	0.0574	0.0086	0.0104	0.0104	0.0086	0.0155	0.0182	**0.0080**	0.0086
3 (Corn_M)	0.0535	0.0634	0.0300	0.0354	0.0080	0.0088	0.0088	0.008	0.0028	0.0033	**0.0017**	0.0023
4 (Corn)	0.0461	0.0648	0.0287	0.0375	0.0179	0.0225	0.0225	0.0179	0.0100	0.0109	**0.0094**	0.0118
5 (Grass_P)	0.0002	0.0002	0.0007	0.0011	**0**	**0**	**0**	**0**	0.0021	0.0027	0.0015	0.0022
6 (Grass_T)	0.0362	0.0377	0.0132	0.0149	**0**	**0**	**0**	**0**	0.0046	0.0051	0.0037	0.0053
7 (Grass_M)	0.0075	0.0087	**0**	**0**	**0**	**0**	**0**	**0**	0.0006	0.0010	**0**	0.0009
8 (Hay_W)	0.0417	0.0511	0.0257	0.0262	0.0073	0.0072	0.0072	0.0073	0.0044	0.0054	0.0030	0.0040
9 (Oats)	**0**	**0**	0.0016	0.0015	0.0006	0.0006	0.0006	0.0006	**0**	**0**	0.0004	0.0005
OCSD/ACSD	0.0531/0.0352		0.0359/0.0206		0.0071/0.0055		0.0071/0.0055		0.0112/0.0063		**0.0064/0.0041**	
OCE/ACE	0.0611/0.0422		0.0391/0.0233		0.0085/0.0064		0.0085/0.0064		0.0133/0.0075		**0.0078/0.0054**	

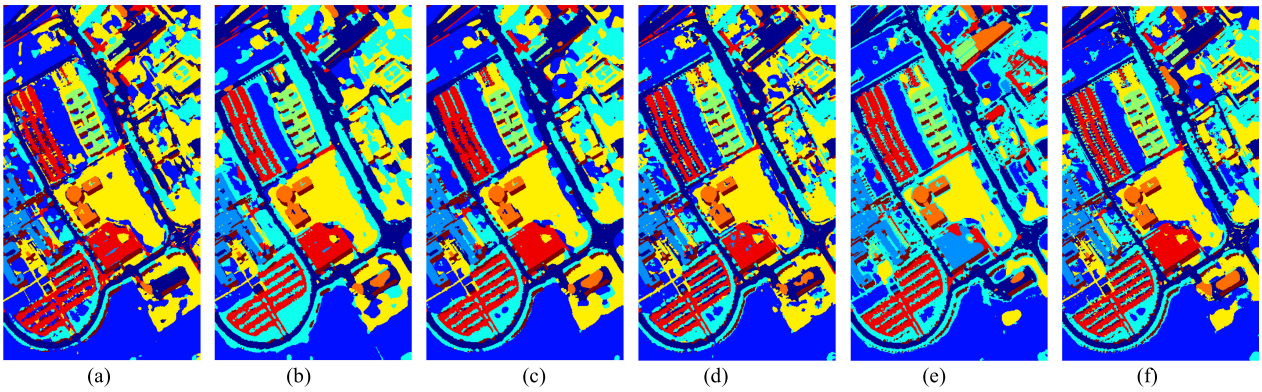


Fig. 14. Classification maps produced by six SS methods for the University of Pavia data. (a) EPF. (b) IEPF. (c) IRTS-EPF. (d) IRTS-GEPF. (e) IRTS-SVM-Gabor. (f) IRTS-Gabor-EPF.

higher P_{OA} and smaller SD_P_{OA}. The highest averaged $P_{\mathrm{OA}} = 99.64\%$ and $P_{\mathrm{AA}} = 99.72\%$ were both produced by 30-fold IRTS-Gabor-EPF along with the smallest SD_$P_{\mathrm{OA}} = 0.12\%$ and SD_$P_{\mathrm{AA}} = 0.08\%$. However, the ones using Gabor filters did require the most computing times with about 2–5 times that required by IRTS-GEPF and IRTS-EPF.

To further measure the uncertainty resulting from RTS, Table XV tabulates CSD and CE along with their OCSD/ACSD and OCE/ACE where IRTS-Gabor-EPF produced the most stable classification results with the smallest OCSD, ACSD and OCE, ACE.

Unlike Salinas, the classification maps produced by single-fold methods, EPF, IEPF, IRTS-EPF, IRTS-GEPF, IRTS-SVM-Gabor, and IRTS-Gabor-EPF shown in Fig. 14 were busy and very difficult to be used to assess the effectiveness of the test methods if we simply compare their results in Fig. 14 against the ground truth in Fig. 13(b) by visual inspection. In this case, we took advantage of plotting the SSD and SE maps of classification results in Figs. 15 and 16 produced by 30-fold EPF, 30-fold IEPF, 30-fold IRTS-EPF, 30-fold IRTS-GEPF, 30-fold IRTS-SVM-Gabor, and 30-fold IRTS-Gabor-EPF. By visual inspection of these SSD and SE maps, it turned out to be very obvious that EPF was the worst one and IRTS-EPF, IRTS-GEPF, and IRTS-Gabor-EPF clearly outperformed EPF and IEPF with much smaller SSD and SE. Also, according to the conducted experiments, IRTS-GEPF generally performed better than IRTS-EPF for the Purdue and the University of Pavia datasets but slightly worse than that for Salinas. This may be due to the reason that using Gaussian filters is not effective for this scene to correctly extract spatial correlation among classified data samples. On the other hand, IRTS-Gabor-EPF using Gabor filters, which did not work as well as IRTS-GEPF for the Purdue and Salinas datasets, worked effectively for the University of Pavia.

One again, like Salinas data, the University of Pavia also has large classes, the differences between the SF-ed classification maps and their corresponding MAP-threshold binary maps were very similar to that obtained from class 11 of the Purdue data. In this case, due to limited space and also to avoid duplication, their experiments are not included.

VIII. NOVELTIES AND DISCUSSIONS

Several novel ideas presented in this article are worth being mentioned in this section.

A. *Novelties of IRTS Theory*

The most important novelty of this article is to develop an IRTS theory for HSIC. As it is named, IRTS consists of two

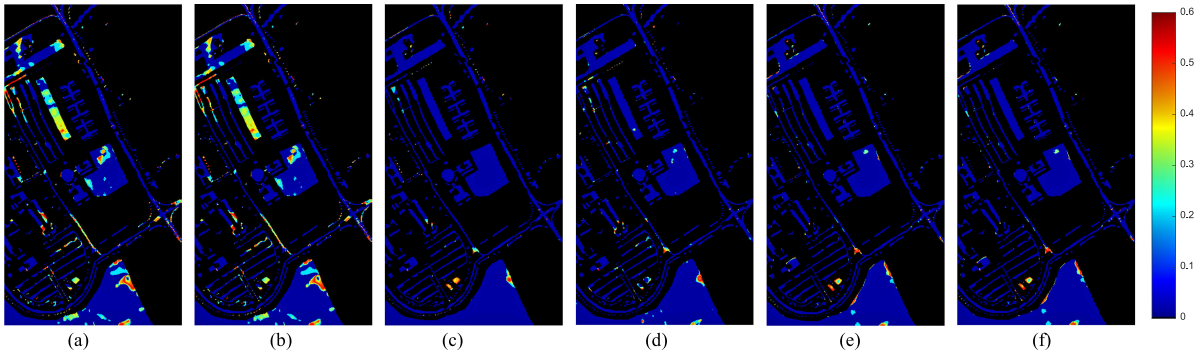


Fig. 15. SSD maps produced by 30-fold experiments for the University of Pavia data along with color bar of SSD maps. (a) EPF. (b) IEPF. (c) IRTS-EPF. (d) IRTS-GEPF. (e) IRTS-SVM-Gabor. (f) IRTS-Gabor-EPF.

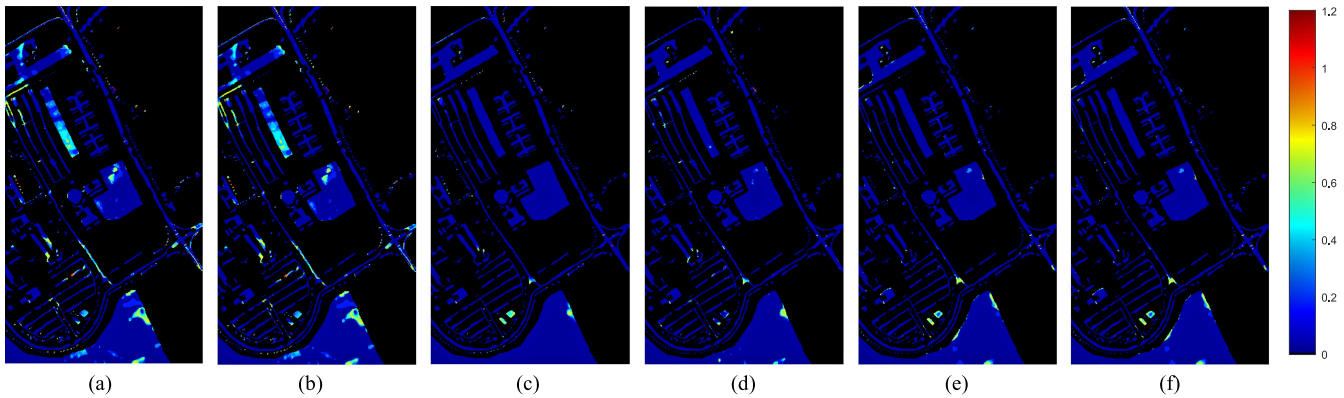


Fig. 16. SE maps produced by 30-fold experiments for the University of Pavia data along with color bar of SE maps. (a) EPF. (b) IEPF. (c) IRTS-EPF. (d) IRTS-GEPF. (e) IRTS-SVM-Gabor. (f) IRTS-Gabor-EPF.

processes. One is an iterative feedback process (IFP) which uses a set of selected training samples to perform SS classification and feeds back SF-ed spectral classification maps to be added to the current image cube to form a new image cube for the next round iteration. The other is RTS which implements random sampling strategy to select a different set of training samples in each iteration so as to reduce the uncertainty caused by RTS after each iteration. The reason for uncertainty reduction is the new image cube augmented by IFP after each iteration includes more and more SF-ed spatial information about the classification. Accordingly, data sample misclassification is decreased iteration after iteration. Thus, the inconsistency in classification caused by RTS is further reduced by such an iterative process. Implementing IFP in conjunction with RTS derives IRTS as an iterative process which carries out n_I iterations with each iteration using a randomly selected training sample set. With this interpretation, IRTS can be considered as an iterative version of a single-fold method n_I-iteration RTS. Since the parameter "n_I" is generally unknown and automatically determined by a stopping rule, this single-fold n_I-iteration RTS is referred to as onefold IRTS and abbreviated as IRTS. So, when IRTS is implemented *K* times, it is called *K*-fold IRTS which has *K* different values of n_I, that is, each fold IRTS runs its own n_I iterations, which varies with each different fold.

Another crucial novelty is to augment the data cube after each iteration. More specifically, the augmented data incorporates the SF-ed spectral classification maps produced by training samples in the preceding iteration to create a new data cube for the next round of iteration. Its idea is similar to semisupervised classification, which grows training samples by including new unlabeled data samples as new training samples in an iterative manner, such as active learning [23]–[26]. The key difference between IRTS and semisupervised classification is that the former grows data cube iteration-by-iteration, while the latter grows training samples iteratively. Their concepts are completely different.

A third important novelty is to take advantage of RTS to model the traditional classification as a random classification problem where a classifier can be implemented as a random classifier from a probabilistic point of view. As a result, an information theory approach can be used to measure randomness resulting from RTS.

With the introduction of a random classifier, a fourth and interesting novelty is to introduce new concepts, SSD, SE, and CE, from an information theory aspect as objective measures to calculate the uncertainty of data samples and classes. To the authors' best knowledge, no similar ideas were reported in the literature.

Finally, by virtue of SSD, SE, and CE, a fifth novelty is the development OCSD and ACSD, which can be considered as counterparts of OA and AA to be used to evaluate class inconsistency. Similarly, OCE and ACE can also be considered as counterparts of OA and AA to measure uncertainty.

B. Discussions

The central premise of this article is to address the RTS issue on the selection of training samples. The developed IRTS theory not only can work for classification but also can be applied to any supervised system which utilizes random sampling strategy to train the system. Specifically, IRTS is designed to iteratively resample training data randomly for each iteration. Interestingly, in previous approaches such as ICEM [13], IEPF [17], the training samples are randomly selected initially from labeled data samples but this same set of initially selected random training samples is then used throughout all the iterations. Comparing to these approaches, IRTS randomly selects a different set of labeled data samples as training samples for each new iteration. Consequently, the training samples selected for each iteration are new and completely random. Intuitively speaking, such random sampling seems to create more randomness. As a matter of fact, it is not. This is because the randomness is reduced by new augmented data cubes to be used for data reprocessing. Such new augmented data cubes are generated iteration-by-iteration by including the new SF-ed spectral classification maps into the current data cube via feedback. These added SF-ed classification maps provide additional spatial classification information about the classified data samples as to improve classification while reducing misclassification errors through an iterative process. This is completely different from the commonly used *K*-fold method for cross validation which uses the same data cube to run *K* times independently to calculate its mean and SD to reflect the confidential interval. IRTS offers a novel approach which reinterprets the *K*-fold method as an iterative method which grows data cubes iteration-by-iteration from which training samples are resampled randomly. Such a new resampled training dataset contains new spatial information fed back by previous iterations to help improve classification performance.

One common challenging issue in classification is classification inconsistency primarily caused by the randomness resulting from the use of training samples randomly selected from labeled data samples. It is not caused by classifiers.

Another is the use of limited labeled data samples which is also another major reason to create classification inconsistency. Such an inconsistency phenomenon occurs in many other fields of machine learning and computer vision. The developed IRTS theory addresses these two issues altogether by providing an effective means of reducing classification inconsistency and uncertainty.

Finally, since IRTS is independent of applications and classifiers, it can also be applied to other supervised applications which require selecting random training samples, such as target detection, spectral unmixing, etc. It can also be applied to other classifiers, such as deep learning network in [7] and convolutional neural networks (CNNs) [27]. As a matter of fact, a new approach of developing CNNs implemented using IRTS is currently undertaken.

IX. Conclusion

This article develops a new approach to HSIC, called IRTS-SS which can reduce the classification inconsistency and uncertainty caused by RTS so as to improve classification accuracy. In what follows, we summarize the key features of IRTS-SS.

1) IRTS is particularly designed to reduce randomness caused by RTS. Its key idea is to repeatedly use different sets of training samples selected by RTS in each iteration carried out by IRTS via feedback loops.
2) IRTS can be implemented in conjunction with any SS classifier to improve its classification performance. For example, it has been shown in [17] that IEPF performed much better than EPF in [2] via feedback. As demonstrated in our experiments, when they were implemented with IRTS, their classification rates were further increased up to more than 99% for all three image scenes. This is significant given the fact that it is generally very difficult to improve a classifier which already scores very high classification rates even by a very small percentage.
3) According to conducted experiments, IRTS can reduce SD_$P_{\rm OA}$ significantly in a single one-fold without using multiple folds as the *K*-fold method does. In other words, the classification uncertainty reduced by IRTS is nearly the same as that using *K*-fold IRTS. This demonstrates that a single one-fold IRTS is sufficiently enough to reduce the uncertainty of classification inconsistency caused by RTS without going through *K* folds. This finding is intriguing. When a single one-fold IRTS is implemented, it takes advantages of correlation among classification results produced by different sets of random training samples. As a consequence, it is expected that a single one-fold IRTS will only require a much smaller number of iterations n_I than *K* to achieve what a *K*-fold method for cross validation can do. This is a very important advantage because it demonstrated that there was no need for using a *K*-fold method to reduce the uncertainty caused by randomness resulting from RTS.
4) Furthermore, IRTS also extends IEPF in two ways. One is IRTS-EPF, which uses different sets of training samples at each iteration compared to IEPF, which uses the same fixed set of training samples in all iterations. The other is to fuse Gaussian filters or Gabor filters with an EPF to derive a new version of IRTS, called IRTS-GEPF and IRT-Gabor-EPF, which improve IEPF performance.
5) It must be noted that IRTS is different from utilizing a large amount of training samples to perform classification. That is, IRTS can use a small number of training samples to achieve the same performance by classification as when using a large number of training samples without IRTS.
6) Finally, IRTS offers a new way to look into an issue of using small sets of training samples. This is particularly important when a complete ground truth is not available or too costly to collect. An effort along this line is currently being investigated.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their insightful comments and suggestions which made this article presentation more readable.

REFERENCES

[1] M. Fauvel, Y. Tarabalka, J. A. Benediktsson, J. Chanussot, and J. Tilton, "Advances in spectral-spatial classification of hyperspectral images," *Proc. IEEE*, vol. 101, no. 3, pp. 652–675, Mar. 2013.

[2] X. Kang, S. Li, and J. A. Benediktsson, "Spectral-spatial hyperspectral image classification with edge-preserving filtrering," *IEEE Trans. Geosci. Remote Sens.*, vol. 52, no. 5, pp. 2666–2677, May 2014.

[3] X. Guo, X. Huang, L. Zhang, L. Zhang, A. Plaza, and J. A. Benedikisson, "Support tensor machines for classification of hyperspectral remote sensing imagery," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 6, pp. 3248–3264, Jun. 2016.

[4] W. Fu, S. Li, L. Fang, X. Kang, and J. A. Benediktsson, "Hyperspectral image classification via shape-adaptive joint sparse representation," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 9, no. 2, pp. 556–567, Feb. 2016.

[5] P. Ghamisi, M. D. Mura, and J. A. Benediktsson, "A survey on spectral–spatial classification techniques based on attribute profiles," *IEEE Trans. Geosci. Remote Sens.*, vol. 53, no. 5, pp. 2335–2353, May 2015.

[6] P. Ghamisi *et al.*, "New frontiers in spectral-spatial hyperspectral image classification: The latest advances based on mathematical morphology, Markov random fields, segmentation, sparse representation, and deep learning," *IEEE Geosci. Remote Sens. Mag.*, vol. 6, no. 3, pp. 10–43, Sep. 2018.

[7] S. Li, W. Song, L. Fang, Y. Chen, P. Ghamisi, and J. A. Benediktsson, "Deep learning for hyperspectral image classification: An overview," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 9, pp. 6690–6709, Sep. 2019.

[8] C.-I Chang and Q. Du, "Estimation of number of spectrally distinct signal sources in hyperspectral imagery," *IEEE Trans. Geosci. Remote Sens.*, vol. 42, no. 3, pp. 608–619, Mar. 2004.

[9] M. E. Winter, "N-FINDR: An algorithm for fast autonomous spectral end-member determination in hyperspectral data," *Proc. SPIE*, vol. 3753, pp. 266–275, 1999.

[10] S. Zhong, C.-I Chang, J. J. Li, X. Shang, M. Song, and Y. Zhang, "Class feature weighted hyperspectral image classification," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 12, no. 12, pp. 4728–4745, Dec. 2019.

[11] M. Song, X. Shang, Y. Wang, C. Yu, and C.-I Chang, "Class-information based band selection for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 11, pp. 8394–8416, Nov. 2019.

[12] T. Cover and J. Thomas, *Elements of Information Theory*. Hoboken, NJ, USA: Wiley, 1991.

[13] B. Xue *et al.*, "A subpixel target approach to hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 9, pp. 5093–5114, Sep. 2017.

[14] S. Zhong, C.-I Chang, and Y. Zhang, "Spectral-spatial feedback close network system for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 10, pp. 8131–8143, Oct. 2019.

[15] A. Plaza and J. Plaza, "Automatic selection of informative samples for SVM-based classification of hyperspectral data using limited training sets," in *Proc. 2nd Workshop Hyperspectral Image Signal Process., Evolution Remote Sens.*, Reykjavik, Iceland, 2010, pp. 1–4.

[16] J. Liang, J. Zhou, Y. Qian, L. Wen, X. Bai, and Y. Gao, "On the sampling strategy for evaluation of spectral-spatial methods in hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 2, pp. 862–880, Feb. 2017.

[17] S. Zhong, C. Chang, and Y. Zhang, "Iterative edge preserving filtering approach to hyperspectral image classification," *IEEE Geosci. Remote Sens. Lett.*, vol. 16, no. 1, pp. 90–94, Jan. 2019.

[18] S. Zhong, C.-I Chang, and Y. Zhang, "Iterative support vector machine for hyperspectral image classification," in *Proc. IEEE Int. Conf. Image Process.*, Athens, Greece, Oct. 2018, pp. 3309–3312.

[19] C. Yu, B. Xue, M. Song, Y. Wang, S. Li, and C. Chang, "Iterative target–constrained interference-minimized classifier for hyperspectral image classification," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 11, no. 4, pp. 1095–1117, Apr. 2018.

[20] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*. Burlington, MA, USA: Academic, 1999, p. 366.

[21] C.-I Chang, "Statistical detection theory approach to hyperspectral image classification: Performance analysis," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 4, pp. 2057–2074, Apr. 2019.

[22] [Online]. Available: http://xudongkang.weebly.com/, Accessed: May, 2014.

[23] J. Li, J. M. Bioucas-Dias, and A. Plaza, "Semisupervised hyperspectral image segmentation using multinominal logistic regression model with active learning," *IEEE Trans. Geosci. Remote Sens.*, vol. 48, no. 11, pp. 4085–4098, Nov. 2010.

[24] J. Li, J. M. Bioucas-Dias, and A. Plaza, "Hyperspectral image segmentation using a new Bayesian approach with active learning," *IEEE Trans. Geosci. Remote Sens.*, vol. 49, no. 10, pp. 3947–3960, Oct. 2011.

[25] M. Ahmad, S. Protasov, A. M. Khan, R. Hussain, A. M. K., and W. A. Khan, "Fuzziness-based active learning framework to enhance hyperspectral image classification performance for discriminative and generative classifiers," *PLOS ONE*, vol. 13, no. 1, Jan. 2018, Art. no. e0188996, doi: 10.1371/journal.pone.0188996.

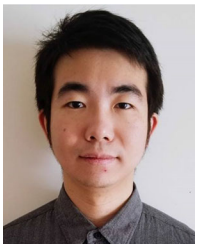
[26] M. Ahmad *et al.*, "Spatial prior fuzziness pool-based interactive classification of hyperspectral images," *Remote Sens.*, vol. 11, May 2019, Art. no. 1136, doi:10.3390/rs11091136.

[27] S. Yu, S. Jia, and C. Xu, "Convolutional neural networks for hyperspectral image classification," *Neurocomputing*, vol. 219, pp. 88–98, Jan. 2017.

Chein-I Chang (Life Fellow, IEEE) received the B.S. degree in mathematics from Soochow University, Taipei, Taiwan, in 1973, the M.S. degree in mathematics from the Institute of Mathematics, National Tsing Hua University, Hsinchu, Taiwan, in 1975, the M.A. degree in mathematics from the State University of New York at Stony Brook, Stony Brook, NY, USA, in 1977, the M.S. and M.S.E.E. degrees from the University of Illinois at Urbana–Champaign, Urbana, IL, USA, in 1982, and the Ph.D. degree in electrical engineering from the University of Maryland, College Park, MD, USA, in 1987.

Since 1987, he has been with the University of Maryland, Baltimore County (UMBC), Baltimore, MD, USA, and is currently a Professor with the Department of Computer Science and Electrical Engineering. He is currently the Chang Jiang Scholar Chair Professor and has been the Director of the Center for Hyperspectral Imaging in Remote Sensing (CHIRS), Dalian Maritime University, Dalian, China, since 2016. In addition, since 2019, he has also been the Chair Professor of National Chiao Tung University, Hsinchu, Taiwan. He was a Plenary Speaker for the Society for Photo-optical Instrumentation Engineers (SPIE) Optics+Applications, Remote Sensing Symposium, in 2009. He is the author of four books: *Hyperspectral Imaging: Techniques for Spectral Detection and Classification* (Kluwer Academic Publishers, 2003), *Hyperspectral Data Processing: Algorithm Design and Analysis* (John Wiley & Sons, 2013), *Real Time Progressive Hyperspectral Image Processing*: *Endmember Finding and Anomaly Detection* (Springer, 2016), and *Recursive Hyperspectral Sample and Band Processing: Algorithm Architecture and Implementation* (Springer, 2017). In addition, he is the Editor for two books, *Recent Advances in Hyperspectral Signal and Image Processing* (Transworld Research Network, 2006) and *Hyperspectral Data Exploitation: Theory and Applications*, (John Wiley & Sons, 2007) and Co-Editor with A. Plaza of a book on *High Performance Computing in Remote Sensing* (CRC Press, 2007). He also holds seven patents on hyperspectral image processing. His research interests include multispectral/hyperspectral image processing, automatic target recognition, and medical imaging.

Dr. Chang was the recipient of a National Research Council Senior Research Associateship Award from 2002 to 2003 sponsored by the U.S. Army Soldier and Biological Chemical Command, Edgewood Chemical and Biological Center, Aberdeen Proving Ground, MD, USA. He was the Guest Editor for a special issue of the *Journal of High Speed Networks* on *Telemedicine and Applications* (April 2000) and Co-Guest Editor of another special issue of the same journal on "*Broadband Multimedia Sensor Networks in Healthcare Applications*" (April 2007). He is also the Co-Guest Editor for special issues on "*High Performance Computing of Hyperspectral Imaging*" for the *International Journal of High Performance Computing Applications* (December 2007), "*Signal Processing and System Design in Health Care Applications*" for the *EURASIP Journal on Advances in Signal Processing* (2009), "*Multipsectral, Hyperspectral, and Polarimetric Imaging Technology*" for the *Journal of Sensors* (2016), and "*Hyperspectral Imaging and Applications*" for *Remote Sensing* (2018). He is a Fellow of SPIE.



Kenneth Yeonkong Ma received the B.S. degree in applied material and optoelectronic engineering from National Chi Nan University, Nantou County, Taiwan, in 2013, and the M.S. degree in optoelectronic engineering from the National Chung Hsing University, Taichung, Taiwan, in 2017. He is currently working toward the Ph.D. degree in electrical engineering with the Remote Sensing Signal and Image Processing Laboratory, University of Maryland, Baltimore County (UMBC), Baltimore, MD, USA.

He is currently a System Engineer with the Sustaining Engineering Team for Tracking and Data Relay Satellite (TDRS), Peraton Corp., Herndon, VA, USA. His research interests include hyperspectral image processing, pattern recognition, machine learning, and digital signal processing.

Chia-Chen Liang received the B.S. degree in computer science from the National University of Kaohsiung, Kaohsiung, Taiwan, in 2014, and the M.S. degree in electrical engineering from National Chung Hsing University, Taichung, Taiwan, in 2017. She is currently working toward the Ph.D. degree in electrical engineering with the University of Maryland, Baltimore County, Baltimore, MD, USA.

Her research interests include hyperspectral image classification, pattern recognition, and deep learning.

Yi-Mei Kuo received the B.S. degree in computer science and information engineering from the National University of Kaohsiung, Kaohsiung, Taiwan, in 2014, and the M.S. degree in electrical engineering from National Chung Hsing University, Taichung, Taiwan, in 2017. She is currently working toward the Ph.D. degree in electrical engineering with the University of Maryland, Baltimore County, Baltimore, MD, USA.

Her research interests include hyperspectral imaging, remote sensing, pattern recognition, and medical data analysis.

Shuhan Chen received the B.S. degree from Ludong University, Yantai, China, in 2011, and the M.S. degree from Liaoning Technical University, Huludao, China, in 2014. She is currently working toward the Ph.D. degree in control theory and control engineering from Zhejiang University, Hangzhou, China.

From 2018 to 2020, she was a Visiting Faculty Research Assistant with the Remote Sensing Signal and Image Processing Laboratory (RSSIPL), Department of Computer Science and Electrical Engineering, University of Maryland, Baltimore County (UMBC), Baltimore, MD, USA. Her research interests include image registration, hyperspectral image processing, and pattern recognition.

Shengwei Zhong received the B.E. degree in information countermeasure technology, the M.S. degree, and the Ph.D. degree in electronics and communication engineering from the Harbin Institute of Technology, Harbin, China, in 2013, 2015, and 2020, respectively.

She was an exchange Ph.D. student and a Faculty Research Assistant with the Remote Sensing Signal and Image Processing Laboratory (RSSIPL), University of Maryland, Baltimore County (UMBC), Baltimore, MD, USA. She is currently a Postdoctoral Researcher and a Teacher with the Nanjing University of Science and Technology, Nanjing, China. Her research interests include hyperspectral image processing, remote sensing image fusion, and applications.