# Iterative Random Training Sampling Convolutional Neural Network for Hyperspectral Image Classification

Chein-I Chang, *Life Fellow, IEEE*, Chia-Chen Liang, and Peter Fuming Hu

***Abstract*— Convolutional neural network (CNN) has received considerable interest in hyperspectral image classification (HSIC) lately due to its excellent spectral–spatial feature extraction capability. To improve CNN, many approaches have been directed at exploring the infrastructure of its network by introducing different paradigms. This article takes a rather different approach by developing an iterative CNN that extends a CNN by including a feedback system to repeatedly process the same CNN in an iterative manner. Its idea is to take advantage of a recently developed iterative training sampling spectral–spatial classification (IRTS-SSC) that allows CNN to update its spatial information of classification maps through a feedback spatial filtering system via IRTS. The resulting CNN is called iterative random training sampling CNN (IRTS-CNN) with several unique features. First, IRTS-CNN combines CNN and IRTS-SSC into one paradigm, an architecture that has never been investigated in the past. Second, it implements a series of spatial filters to capture spatial information of classified data samples and further feeds this information back via an iterative process to expand the current input data cube for the next iteration. Third, it utilizes the expanded data cube to randomly reselect training samples and then to reimplement CNN iteratively. Last but not least, IRTS-CNN provides a general framework that can implement any arbitrary CNN as an initial classifier to improve its performance through an iterative process. Extensive experiments are conducted to demonstrate that IRTS-CNN indeed significantly improves CNN, specifically when only a small size of limited training samples is used.**

***Index Terms*— Convolutional neural network (CNN), iterative random training sampling CNN (IRTS-CNN), random training sampling (RTS).**

Manuscript received 11 March 2023; revised 5 May 2023 and 12 May 2023; accepted 21 May 2023. Date of publication 26 May 2023; date of current version 12 June 2023. This work was supported in part by the National Science and Technology Council (NSTC) under Grant 111-2634-F-006-012. *(Corresponding author: Chia-Chen Liang.)*

Chein-I Chang is with the Center for Hyperspectral Imaging in Remote Sensing (CHIRS), Information and Technology College, Dalian Maritime University, Dalian 116026, China, also with the Remote Sensing Signal and Image Processing Laboratory, Department of Computer Science and Electrical Engineering, University of Maryland, Baltimore, MD 21250 USA, also with the Department of Electrical Engineering, National Cheng Kung University, Tainan 70101, Taiwan, and also with the Department of Computer Science and Information Management, Providence University, Taichung 02912, Taiwan (e-mail: cchang@umbc.edu).

Chia-Chen Liang is with the Remote Sensing Signal and Image Processing Laboratory, Department of Computer Science and Electrical Engineering, University of Maryland, Baltimore, MD 21250 USA (e-mail: lcjane212@gmail.com).

Peter Fuming Hu is with the Department of Anesthesia, University of Maryland School of Medicine, R. A. Cowley Shock Trauma Center, Shock Trauma Anesthesia Organized Research Center, Baltimore, MD 21201 USA (e-mail: phu@som.umaryland.edu)

Digital Object Identifier 10.1109/TGRS.2023.3280205

## NOMENCLATURE

| | |
|---|---|
| AA | Average accuracy. |
| AC | Average classification. |
| APR | Average precision rate. |
| $A^2S^2$K-ResNet | Attention-based adaptive spectral–spatial kernel improved residual network. |
| AUC | Area under an ROC curve. |
| BS | BKG suppressibility. |
| BSS | Band subset selection. |
| CNN | Convolutional neural network. |
| EPF | Edge-preserving filter. |
| FSKNet | Faster selective kernel mechanism network. |
| HSIC | Hyperspectral image classification. |
| HybridSN | Hybrid spectral convolutional neural network. |
| IEPF | Iterative edge-preserving filter. |
| IRTS | Iterative random training sampling. |
| MAP | Maximum a posteriori. |
| OA | Overall accuracy. |
| OC | Overall classification. |
| OPR | Overall precision rate. |
| ROC | Receiver operating characteristic. |
| RTS | Random training sampling. |
| SF | Spatial filter. |
| SFCMap | Spatial filtered classification map. |
| SSC | Spectral–spatial classification. |
| TI | Tanimoto index. |
| SQ-RDFBBS | Sequential RDF band subset selection. |
| VD | Virtual dimensionality. |

## I. INTRODUCTION

HSIC has been studied extensively in hyperspectral imaging. Many approaches have been reported in the past. Among them, SSC and CNN are of particular interest.

A hyperspectral imaging sensor is called "*hyperspectral*" because it utilizes hundreds of contiguous spectral bands to extract rich spectral information to uncover unknown and subtle material substances, such as subpixel targets and mixed data samples. Accordingly, it is critical to capture such spectral properties and characteristics in the first place before they are compromised by follow-up data processing. Consequently, spectral processing is generally used as an important initial process to preserve crucial spectral features for this purpose. This is particularly true for hyperspectral data compression where spectral information is very likely to be sacrificed and lost by spatial compression [1]. This same issue may be also encountered in HSIC. That is, the useful and crucial spectral information for classification may be lost if spatial processing is prior to spectral processing. To address this issue effectively, SSC has emerged as a promising technique in HSIC, which implements a spectral classifier to extract spectral information followed by SFs to capture spatial information [2], [3], [4] and is proved to be very effective in HSIC. One of the most widely used SSC methods is the EPF-based method developed in [5], which has been shown to outperform many existing HSIC methods with the spectral classifier and SFs specified by support vector machine (SVM) to perform spectral classification followed by EPF to capture spatial information from classified data samples. EPF-based methods were later improved by including iterative feedback loops in EPF methods to derive IEPF methods [6]. Furthermore, IEPF was extended to a general framework of the SSC network by making it a close feedback network, called the spectral–spatial feedback close network system (SSFCNS) [7] with IEPF included as its special case. Most recently, a new concept of RTS was developed as IRTS-SSC [8] to improve IEPF by allowing SSC to randomly reselect training samples in each new iteration during the entire iterative process.

As an alternative to SSC, CNN has been also shown to yield great performance in many applications in the visual information processing field, such as facial recognition [9], [10], human motion recognition [11], skin lesions classification [12], and image classification [13], [14]. In addition, significant progress has been also made by deep learning for HSIC in recent years [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31], [32], [33], [34], [35].

Generally speaking, a traditional CNN-based classification model mainly consists of three types of layers, taking a hyperspectral image (HSI) as the input layer, convolution sublayers and pooling sublayers implemented as hidden layers, and fully connected layers as the output layer, which produces the final classification maps. It can be implemented as 2D- or 3D-CNN depending upon its inputs. The CNN-based HSI classification model used in [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31], [32], [33], [34], and [35] is a 3D-CNN that has 3D image patches implemented as inputs with the labels are determined by the center pixel of the patch. In the convolutional sublayer, different filter kernels are applied to input samples and generate 3D volumes of feature maps. Since the number of feature maps becomes very large as the number of hidden layers increases, the computational burden also becomes excessively high layer by layer. To resolve this issue, pooling sublayers are applied to reduce the dimensions of the feature maps produced in the convolution sublayers. Once the images are reduced to a manageable level, a fully connected layer is implemented as the last layer for final classification. In general, the number of outputs in this layer is specified by the number of classes.

Despite that CNN can automatically learn high-level features using a more complex model, it usually takes up too much memory space in a computer and requires high-power computers for computing. Meanwhile, it also accompanies some disadvantages and complexity of the used models with high memory storage requirements for computers. In addition, due to the complicated network structure and availability of limited training samples, such deep learning models might be suboptimal or provide unstable performance [36].

To address the abovementioned issues, this article presents a new general architecture of a CNN shown in Fig. 1, to be called IRTS-CNN where CNN can be implemented as either 2D or 3D. Similar to the SSFCNS developed by Zhong et al. [7], it includes feedback loops to make CNN a close network system as opposed to CNN, which is an open feed-forward system. Most importantly, IRTS-CNN implements an RTS strategy in each iteration to randomly reselect a new set of training samples for CNN to be used for the next iteration. It extends the traditional CNN in many aspects. In particular, four key elements are used to design IRTS-CNN. One is its use of SFs to capture spatial information from the CNN-classified maps to yield soft-decision CNN classification maps, referred to as SFCmaps. A second element is to feed SFCMaps back to the current input data cube to create a new expanded data cube to be used for the reimplementation of CNN again. A third one is to use RTS to randomly reselect training samples from the newly augmented data cube to retrain CNN. Finally, an automatic stopping rule is particularly designed to terminate IRTS-CNN. All of these elements are new and cannot be found in the traditional CNN.

In order to facilitate understanding of IRTS-CNN, a simple and generic 3D-CNN model is used for experiments to illustrate its concept and utility. However, a more complex CNN with sophisticated structures can be also used to implement IRTS-CNN. Nevertheless, experimental results show that, even for such a simple 3D-CNN, it not only can have a great performance as a complex model did [17], [30], [33] but also can be computed in lower computational complexity. To substantiate the advantages of IRTS-CNN three image scenes, Purdue's Indian Pines, Salinas, and the University of Pavia were used for experiments. Specifically, the Purdue data have imbalanced class issues, and the University of Pavia has complicated background (BKG) issues. In addition, two new soft-decision detection measures developed from the 3D ROC curves in [37] are rederived as soft-decision classification measures, called AC and class-weighted classification (CWC). These two measures along with the traditional hard-decision classification measures, OA, and AA were used together to conduct an in-depth performance analysis.

Several contributions are summarized as follows.

1) Since IRTS-CNN is designed as a general framework, the CNN used in IRTS-CNN is generic. Accordingly, IRTS-CNN is applicable to any arbitrary CNN, including 2D- and 3D-CNNs along with more complicated CNN models, which will be illustrated in Section VII, and more complicated CNN models as demonstrated in Section VIII.

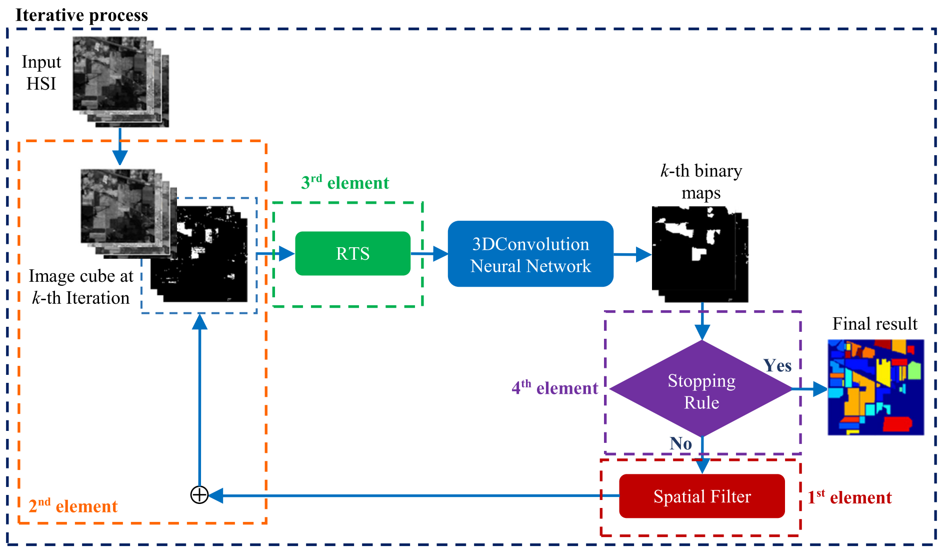


Fig. 1. Graphic diagram of implementing IRTS-3D-CNN.

2) Any CNN can be used to be implemented as an initial condition of IRTS-CNN. As a result, if a CNN performs very well and cannot be further improved by IRTS-CNN, then no feedback is necessary. In this case, IRTS-CNN will be terminated after the first iteration. Consequently, once IRTS-CNN is activated, its performance in classification is always as good as or better than its initial condition.
3) Finally, it culminates in an extensive experiment-based comprehensive study to conduct a comparative analysis among machine learning-based SSC and most recently developed deep learning-based classification techniques that provide evidence of the superior performance of IRTS-CNN resulting from using RTS. In particular, several new soft-decision-based classification measures derived from 3D ROC curves are also introduced to conduct quantitative analysis.

The remainder of this article is organized as follows. Section II briefly reviews related work on CNN. Section III derives IRTS-CNN and describes a method proposed by Song et al. [38] for training sample class size selection. Section IV includes three image datasets to be used for experiments. Section VI develops classification measures to be used for performance evaluation. Section VII conducts experiments on a comparative analysis among machine learning-based SSC methods and discusses the experimental results. Section VIII further conducts experiments on a comparative analysis among three recently developed deep learning-based methods. Finally, in Section IX, the experimental results summarize novelties derived from this article and discussions.

## II. Related Works

This section briefly reviews the current existing CNN reported in the recent literature. Many CNN-HSIC techniques have shown promising performance for HSI classification in the past decade. For instance, Fang et al. [15] proposed a network using dense convolutional networks with a spectral-wise attention mechanism to enhance the distinguishability of spectral features. Paoletti et al. [16] developed a deep pyramidal residual network to improve the spectral–spatial features uncovered by the convolutional filters where the residual-based approach in DPRN gradually increased the feature map dimensions in all convolutional layers and then grouped them in pyramidal bottleneck residual blocks. Roy et al. [17] proposed an HybridSN that is a spectral–spatial 3D-CNN followed by spatial 2D-CNN. Zhu et al. [18] used deformable convolutional sampling locations with size and shape adaptively adjusted according to HSI's complex spatial contexts to create a deformable HSI classification network. Yu et al. [19] derived a simplified 2D-3D-CNN that was based on the cooperation between 2D CNN and simplified 3D convolution layers. Sun et al. [20] further proposed a spectral–spatial attention network to capture discriminative spectral–spatial features form attention areas of an HSI image cube where an attention module was used to suppress the effects of interfering pixels.

Moreover, spectral–spatial feature-based CNNs have also been extended to extract multiscale features. Pooja et al. [21] proposed a CNN-based method along with multiscale and dilated convolutions with residual connection concepts. He et al. [22] introduced a handcrafted feature extraction method according to multiscale covariance maps for CNN. Wan et al. [23] developed a multiscale dynamic graph convolutional network, which used dynamic graphs to encode the intrinsic similarities among the image regions, and introduced multiple graphs with different neighborhood scales to fully exploit the multiscale information. Gong et al. [24] introduced a CNN with multiscale convolution and diversified metrics to obtain discriminative features for HSIC. Xu et al. [25] proposed a multiscale spectral–spatial CNN for HSIs to integrate multiple receptive fields fused features with multiscale spatial features at different levels. Furthermore, Wang et al. [26] proposed an alternately updated spectral–spatial convolution network with a recurrent feedback structure to learn refined spectral and spatial features, which is different from IRTS-3D-CNN that actually expands the input HSI cubes using the fed-back SFCMaps from the previous iterations.

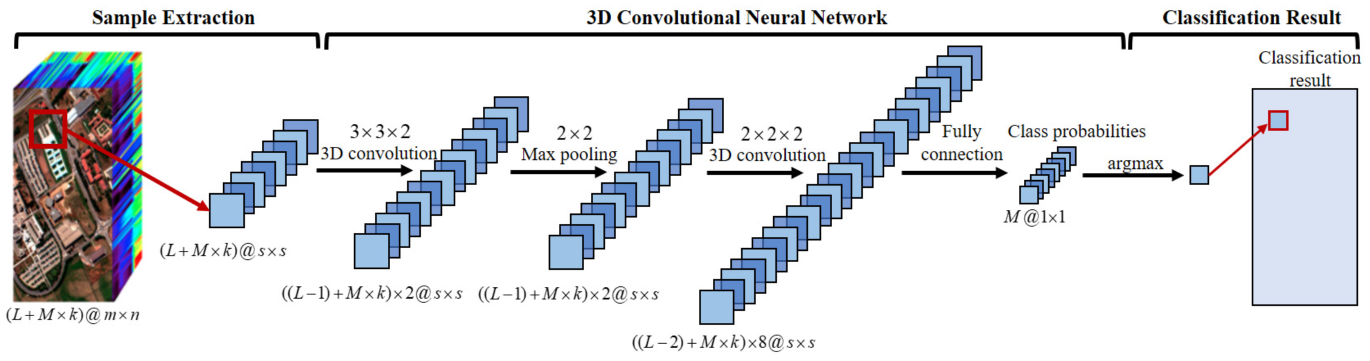


Fig. 2. 3D-CNN-based HSI classification framework. The first step is sample extraction, where $s \times s \times (L + M \times j)$ sized sample is extracted from a neighborhood window centered around the target pixel. Once samples are extracted from HSI, they are put through the 3D-CNN to extract deep spectral–spatial features and calculate classification probabilities.

In addition to feature extraction, two main issues are also obstacles to achieve high accuracy. One is the imbalanced class issue incurred by HSI classification, such as Zhu et al. [27] proposed a spectral–spatial dependent global learning (SSDGL) framework based on global convolutional long short-term memory and global joint attention mechanism. The other is the limited use of training samples. For example, Chen et al. [28] proposed a network based on a sparse representation with the quantum genetic algorithm and CNN to represent the image as a linear combination of base atoms in the dictionary so that the quantum genetic algorithm could sparsely decompose the image to generate the sparse representation of the image. Cao et al. [29] looked into a compressed CNN with the knowledge distillation method, which generated a large number of virtual samples to record the classification boundary information of a teacher network, and then used these samples to guide the training of a student network. Roy et al. [30] used selective 3D convolutional kernels with 3D residual building blocks and an efficient feature recalibration mechanism to design an $A^2S^2$K-ResNet. Furthermore, Roy et al. [31] also derived a model to automatically generate more samples from minority classes via their existing samples during training. Gao et al. [32] designed a deep model based on an induction network for small sample classification but required a metatraining strategy to improve the classification accuracy with a few labeled samples, which resulted in excessive execution time.

Due to the high computational complexity, Li and Zhang [33] combined 3D-CNN and 2D-CNN conversion modules with a selective kernel mechanism to reduce the computational complexity, called FSKNet. Morales et al. [34] later applied a novel band selection method to CNN to reduce redundant bands. Wang et al. [35] developed a supervised deep learning framework, called end-to-end cubic CNN, which applied principal component analysis (PCA) and 1-D convolution to remove the redundant information from HSI where the 3D convolution not only extracted but also fused the spatial and spatial–spectral features from different dimensions.

## III. IRTS-CNN

### A. 3D-CNN for HSIC

Fig. 2 shows the framework of a general 3D-CNN-based HSIC classifier considered to be generic. Technically speaking, any CNN described above can be used in this framework of structure. However, in order to simplify the presentation, a simple architecture of 3D-CNN shown in Fig. 2 is used for IRTS-3D-CNN in Fig. 1 to provide a better illustration.

At the beginning of the input layer, an HSI is divided into small overlapping 3D neighboring patches as a sample centered around a pixel $(x, y)$, which covers the $s \times s$ window, and the label of each sample is decided by the centered pixel of the patches. Let the original HSI cube be denoted by $\Omega \in \Re^{m \times n \times L}$, where $m$ is the width, $n$ is the height, and $L$ is the number of bands. Thus, in the $k$th iteration, the band number of the input HSI cube is presented as $(L + M \times k)$, where $M$ is the number of classes, and the size of the 3D patch $P_{x,y}$ is $s \times s \times (L + M \times k)$, where $s$ and $(L + M \times k)$ are the spatial size and the number of spectral bands, respectively. Once 3D samples are extracted from HSI, they are fed into the CNN model to obtain the probabilistic classification results.

In 3D-CNN, a 3D patch is convolved with a kernel of $3 \times 3 \times 2$ size with padding in the first convolution layer. Thus, the size of the output is $s \times s$ and its band number is $((L - 1) + M \times k) \times 2$. After applying batch normalization and the rectified linear unit (ReLU) function, its output is sent to the max pooling with a $2 \times 2$ size kernel with padding of stride $= 1$. The output of max pooling is a cube of size $s \times s \times [((L - 1) + M \times k) \times 2]$. Then, the second convolution layer is applied using a $2 \times 2 \times 2$ size kernel and padding. The output after batch normalization and ReLu function is a cube of size $s \times s \times [((L - 2) + M \times k) \times 8]$. After a series of convolutional and pooling layers, a fully connected layer combines the output values of the last convolution layer into a high-level feature vector with its number of outputs nodes equal to the number of classes $M$ where its output values are used to calculate the probabilistic classification results via softmax by

$$p_j = \frac{e^{x_j}}{\sum_{j=1}^{M} e^{x_j}}, \quad j = 1, 2, \ldots, M. \tag{1}$$

The final classification of pixel at $(x, y)$, $p_{(x,y)}$ can be predicted by

$$\text{class}_{p_{(x,y)}} = \arg_{j=1,2,\ldots,M} \max p_j. \tag{2}$$

### *B. IRTS*

As an alternative to deep learning model-based approaches, iterative SSC (ISSC) has also shown great promise in HSIC [7], [8], [9]. It is simple to implement and does not require a model to describe the architecture of a classifier. More recently, ISSC has been also shown to be very competitive with many existing HSIC methods in [8], [39], [40], and [41]. Its key idea consists of three elements. One is to include a set of SFs to capture spatial information of classified data samples. Then, a second one is to feed the spatial-filtered classification maps back to augment the current data cube to a new expanded data cube that can be used for the next round of HSIC. A third one is to introduce an RTS strategy to resample the expanded dataset iteratively. There are two crucial benefits resulting from ISSC, which cannot be accomplished by feed-forward deep learning model-based approaches. The most important one is the data cube augmented by each iteration, which contains more spatial information about classification maps than that obtained by its previous iterations. The second one is the use of RTS, which is able to randomly reselect data samples that were not selected by previous iterations. As a result, the worst scenario is that the training data samples reselected at all iterations are at the same locations, referred to as fixed training sampling, while the best scenario is that the training data samples reselected at all iterations are different. In the latter case, RTS may run through all the possible ground-truth data samples as long as the iterative process is continued indefinitely. Detailed implementations can be found in Section III-C.

### *C. IRTS-CNN*

Basically, the IRTS-CNN presented in this section combines CNN with IRTS into a single network that jointly implements CNN as a feed-forward network system and IRTS as a feed-backward system. Such a feed-forward/feed-backward system is first ever proposed and has never been investigated for HSIC in the past.

Since the proposed IRTS-CNN works for 2D- and 3D-CNNs, CNN is used to simplify notations throughout this article. Fig. 1 shows a graphic diagram of implementing the IRTS-CNN algorithm; each stage of IRTS-CNN is described as follows.

1) *Initial Condition:* The original image cube denoted by an HSI is considered the original input data $\Omega^{(0)}$. The training sample number for each class $n_j$ will be determined by the rule suggested in [38]. Then, at the $k$th iteration, the training samples will be produced by a randomly selected training sample set, denoted by $\mathrm{S}^{(k)}$.
2) *First Stage:* The initial randomly selected training sample set $\mathrm{S}^{(0)}$ is used to initialize CNN. The initial CNN-classified map C-MAP$^{(0)}_{\mathrm{CNN}}$ is produced by implementing CNN on $\Omega^{(0)}$.
3) *Second Stage:* CNN-classified binary maps are generated for each class from the CNN-classified map and will be used as the input of guided filters to generate the soft probability maps.
4) *Third Stage:* The soft probability maps produced in the second stage are fed back and combined $\Omega^{(0)}$ to create a new image cube $\Omega^{(1)}$ as new input data for CNN for the next round iteration.
5) *Final Stage:* The entire process from the first stage to the third stage will be repeated over and over again until a stopping rule is satisfied.

An algorithm to implement IRTS-CNN step by step in detail is provided in Algorithm 1.

IRTS-CNN

1) Initial condition: Let $\Omega^{(0)} = \{\mathrm{HSI}\}$ and a training sample number of the $m$-th class denoted be $n_j$ where $M$ is number of classes and $m = 1, w, \ldots, M$. An initial randomly selected training sample set, $\mathrm{S}^{(0)}$, generated according to $n_m$. Set $k = 1$.
2) 1st stage:
   Implement CNN on $\Omega^{(0)}$ and $\mathrm{S}^{(0)}$ to produce an initial classification map C-MAP$^{(0)}_{\mathrm{CNN}}$, i.e., $M$binary probability maps $\{\mathrm{B}^{(0)}_{\mathrm{CNN},m}\}^{M}_{m=1}$, Specifically, the value of pixel at $(x, y)$ in the $m$-th binary map is defined as follows:
   $$\mathrm{B}^{(0)}_{\mathrm{CNN},m}(x,y) = \begin{cases} 1, & \text{if pixel at } (x,y) \in \text{class } m \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$
3) 2nd stage:
   - At the $k$-th iteration, use (3) to generate $M$ binary probability maps $\{B^{(k)}_{\mathrm{CNN},m}\}^{M}_{m=1}$,i.e.,
   $$B^{(k)}_{\mathrm{CNN},m}(x,y) = \begin{cases} 1, & \text{if pixel at } (x,y) \in \text{class } m \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$
   - Apply an SF to generate soft probability maps $\{\mathrm{S}^{(k)}_{\mathrm{SF},m}\}^{M}_{m=1}$ found by (1) for $1 \le m \le M$.
4) 3rd stage:
   - Generate a new data cube
   $$\Omega^{(k)} = \Omega^{(k-1)} \cup \left\{\mathrm{S}^{(k)}_{\mathrm{SF},1}\right\} \cup \cdots \cup \left\{\mathrm{S}^{(k)}_{\mathrm{SF},M}\right\}. \tag{5}$$
   - Randomly select a new set of training samples, $\mathrm{S}^{(k)}$ and implement CNN on $\Omega^{(k)}$ and $\mathrm{S}^{(k)}$ to produce the CNN-classification map C-MAP$^{(k)}_{\mathrm{CNN}}$, i.e., $\{\mathrm{B}^{(k)}_{\mathrm{CNN},m}\}^{M}_{m=1}$.
   - Check if $\{\mathrm{B}^{(k)}_{\mathrm{CNN},m}\}^{M}_{m=1}$ satisfies a stopping rule, then go to step 5. Otherwise, let $k \leftarrow k + 1$, go to stage 2.
5) The algorithm is terminated and C-MAP$^{(k)}_{\mathrm{CNN}}$, i.e., $\{B^{(k)}_{\mathrm{CNN},m}\}^{M}_{m=1}$ is the final classification maps.

Figs. 3 and 4 depict flow charts of implementing the initial condition, zeroth iteration, i.e., $k = 0$ and the $k$th iteration of IRTS-CNN, respectively. As can be seen from Fig. 3, the traditional CNN is implemented in the zeroth iteration as an initial condition of IRTS-CNN.

### *D. Training Sample Number Decision for IRTS-CNN*

Most training sample selection methods determine the number of training samples based on the sample ratio (SR) for each class. However, these will lead to poor performance caused by the small number of training samples selection for small classes. In order to overcome this issue, Kang [38] suggested a method, referred to as Kang's method, to determine the

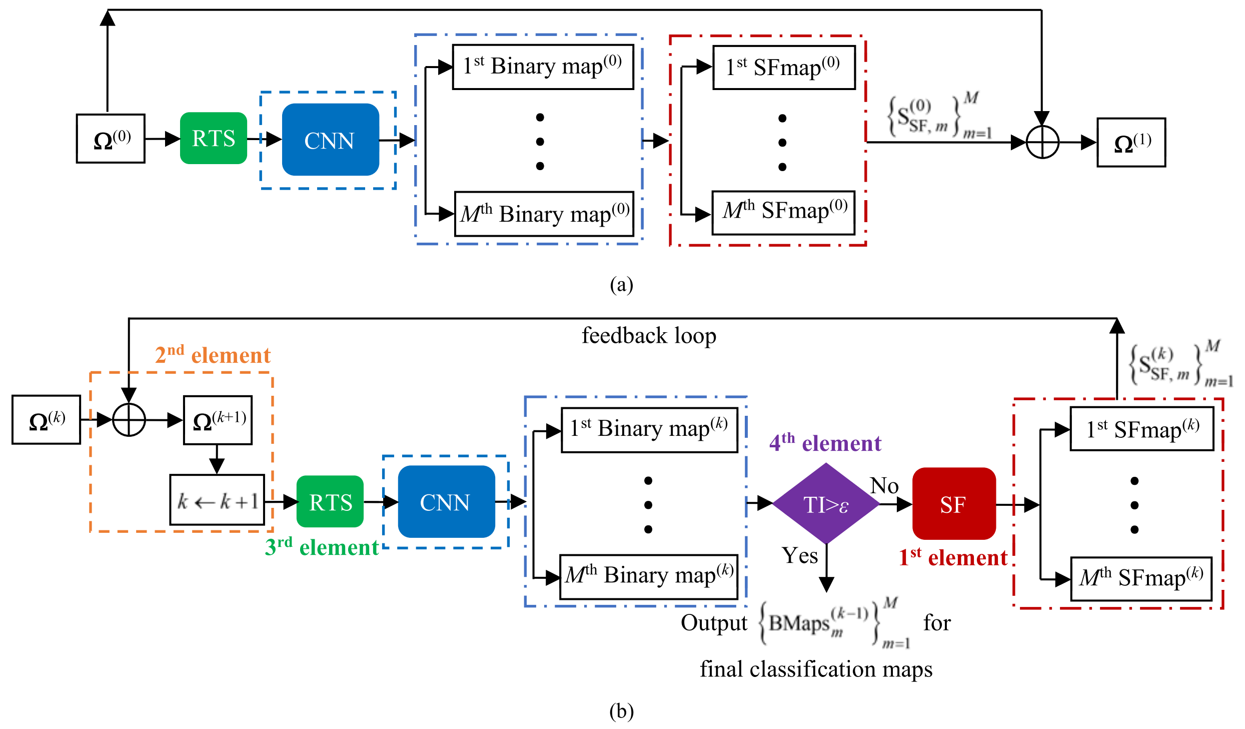


Fig. 3. Block diagram of implementing IRTS-CNN with initial condition and iterations. (a) Zeroth iteration ($k = 0$) implemented in IRTS-CNN. (b) $k$th iteration with $k \geq 1$ implemented in IRTS-CNN.

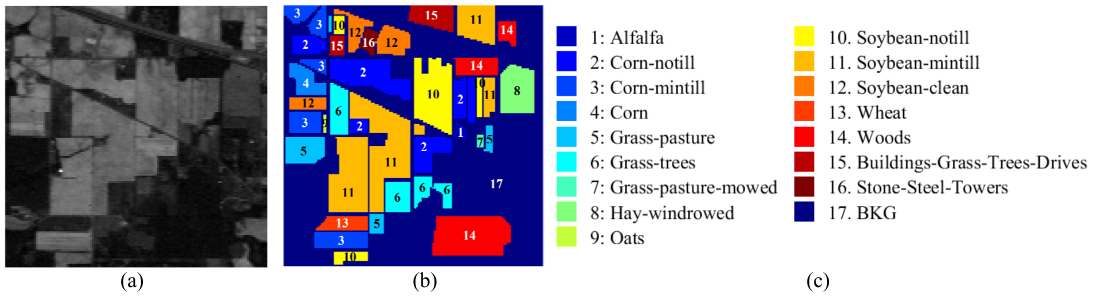


Fig. 4. Purdue's data and its ground truth with 16 classes. (a) Band 186 (2162.56 nm). (b) Ground-truth map. (c) Classes by colors.

training sample size for each of the classes. Most recently, Zhong et al. [42] and Song et al. [43] used class features to develop a new concept, called class significance that can be used to allocate training size for each class. Such training sample class size allocation guarantees that small classes will have enough training samples to be used for classification. Since Kang's method is simple to implement, for the purpose of illustration, it was used in this article to determine the number of training samples that need to be selected in each class. For those who are interested in this method, we refer details to Kang's website [38]. Nevertheless, the works in [42] and [43] can be used to replace Kang's method for a general purpose.

### E. Stopping Rule for IRTS-CNN

The stopping rule to be used for ICNN is based on a discrepancy between the two classification binary maps generated at the $k$th and $(k-1)$st iterations, which can be calculated by

$$\mathrm{TI}^{(k)} = \sum_{m=1}^{M} p(C_m)\mathrm{TI}_m^{(k)} \tag{6}$$

where $\mathrm{TI}_m^{(k)} = (|B_{\mathrm{CNN},m}^{(k)} \cap B_{\mathrm{CNN},m}^{(k-1)}|/|B_{\mathrm{CNN},m}^{(k)} \cup B_{\mathrm{CNN},m}^{(k-1)}|)$ with the TI defined in [44], and $B_{\mathrm{CNN},m}^{(k)}$ and $B_{\mathrm{CNN},m}^{(k-1)}$ are the $k$th and $(k-1)$st binary maps for class $m$ in two consecutive iterations. TI represents a ratio of the size of the intersection to the size of the union between two classification results. The range of TI is between 0 and 1. It is also known as intersection over union (IOU) in objection detection. A larger TI value indicates a higher overlap between $B_{\mathrm{CNN},m}^{(k)}$ and $B_{\mathrm{CNN},m}^{(k-1)}$. When the TI of the $k$th iteration is greater than a prescribed threshold $\varepsilon$, the iterative process will be terminated.

Finally, in order for readers who are interested in repeating IRTS-CNN, we have uploaded its source codes to the following our lab website provided in the following link https://wiki.umbc.edu/display/rssipl/10.+Download.

## IV. Classification Measures to Be Used for Performance Evaluation

There are two types of classification measures introduced in this section: hard-decision measures [45] and soft-decision measures.

## A. Hard-Decision Classification Measures

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

In traditional HSIC, the popular performance measurements are class accuracy (A) and OA, given by

$$p_A(C_m) = \text{accuracy of the } m\text{th class} = \frac{n_{mm}}{\sum_{j=1}^{M} n_{jm}} = \frac{n_{mm}}{n_m} \tag{7}$$

$$P_{\text{CWA}} = \sum_{m=1}^{M} p(C_m) p_A(C_m) \tag{8}$$

where $p(C_i)$ is the significant probability of data samples in the $m$th class, $C_m$, such as SR, which can be considered as weight of the $m$th class, $C_m$

$$P_{\text{OA}} = \frac{1}{N}\sum_{m=1}^{M} n_{mm} = \sum_{m=1}^{M} \frac{n_m}{N} p_A(C_m) \tag{9}$$

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

$$P_{\text{AA}} = \frac{1}{M}\sum_{m=1}^{M} \frac{n_{mm}}{n_m} = \frac{1}{M}\sum_{m=1}^{M} p_A(C_m). \tag{10}$$

Another measure is the precision rate (PR) given by

$$p_{\text{PR}}(C_m) = \text{precission rate of } C_m = \frac{\hat{n}_{mm}}{\hat{n}_m} \tag{11}$$

where $\hat{n}_m = \sum_{j=1}^{M} \hat{n}_{mj}$ is the total number of data samples that are classified into the $m$th class and $\hat{n}_{mj}$ is the number of data samples classified into the $m$th class but supposed to be in the $j$th class. Thus, the APR, $P_{\text{APR}}$, in correspondence to $P_{\text{AA}}$ in (10) is defined by

$$P_{\text{APR}} = \frac{1}{M}\sum_{m=1}^{M} p_{\text{PR}}(\hat{C}_m). \tag{12}$$

Since $n_{mm}$ is identical to $\hat{n}_{mm}$ and $\hat{N} = \sum_{i=1}^{M} \hat{n}_i$ is equal to $N = \sum_{i=1}^{M} n_i$. As a result, an OPR, $P_{\text{OPR}}$, can be also defined as AC, the counterpart of $P_{\text{OA}}$ in (9), by

$$P_{\text{CWPR}} = \sum_{m=1}^{M} p(\hat{C}_m) P_{\text{PR}}(\hat{C}_m) \tag{13}$$

where $p(\hat{C}_m)$ is the significant probability of classified data samples in the $m$th class, $\hat{C}_m$

$$P_{\text{OPR}} = \sum_{i=1}^{M} \left(\frac{\hat{n}_i}{\hat{N}}\right) P_{\text{PR}}(\hat{C}_i) = \sum_{i=1}^{M} \left(\frac{n_i}{N}\right) P_A(C_i) = P_{\text{OA}} \tag{14}$$

which implies that $P_{\text{OPR}}$ is essentially the same $P_{\text{OA}}$, provided that BKG class samples are not included for classification.

It should be noted that PR is also known as the user's accuracy as opposed to OA, which is also known as the producer's accuracy. It 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.

## B. Soft-Decision Classification Measures

In addition to the above classification measures, several 3D ROC curve-based soft-decision classification measures recently developed in [37] are also used for performance evaluation. ROC analysis is originally used as an evaluation tool for target detection but recently has been used for HSIC by interpreting target detection probability, $P_D$ as class accuracy, $p_A(C_m)$ in (7) with a specific class, $C_m$ to be classified as a target of interest and false alarm probability, and $P_F$ as a misclassification error resulting from a BKG data sample being classified into $C_m$. As a result, by interpreting a target to be detected as a specific class, $C_m$, to be classified, the concept of $P_{\text{PR}}$ in (11) is actually derived from $P_F$. The plot of a 2D ROC curve of $P_D$ versus $P_F$ can be translated to a plot of points $[P_A(C_m), P_F(C_m)]$ by using $P_F(C_m)$ as a parameter ranging from 0 to 1, and the AUC is generally used to evaluate the effectiveness of a classifier in classifying data samples into $C_m$.

According to the Neyman–Pearson (NP) detection theory [46], an NP detector is found by maximizing $P_D$ subject to the constraint $P_F \leq \beta$. In this case, $P_D$ cannot be calculated directly by $P_F$. It must first find the threshold $\tau$ that calculates $P_F$ with $P_F = \beta$. This found $\tau$ is then used to calculate $P_D$. Thus, a point of $(P_D, P_F)$ on the 2D ROC curve is a point indeed calculated by the same $\tau$. As a result, neither $P_D$ nor $P_F$ can stand alone as a detection measure. This may explain why the 2D ROC curve has not been used for classification. In order to resolve this issue, a 3D ROC was first proposed in [47] and later in [48] for hyperspectral subpixel target detection by considering the threshold $\tau$ as an independent parameter, and $P_D$ and $P_F$ are functions of $\tau$. Thus, a 3D ROC curve is actually a function of $(P_D, P_F, \tau)$. As a consequence, a 3D ROC curve can generate three 2D ROC curves of $(P_D, P_F)$, $(P_D, \tau)$, and $(P_F, \tau)$, where the 2D ROC curve of $(P_D, P_F)$ is exactly the same as the commonly used 2D ROC curve in detection theory. For each of such 2D ROC curves, their AUC values can be calculated as $\text{AUC}_{(D,F)}$, $\text{AUC}_{(D,\tau)}$, and $\text{AUC}_{(F,\tau)}$ corresponding to 2D ROC curves of $(P_D, P_F)$, $(P_D, \tau)$, and $(P_F, \tau)$, respectively. Specifically, $\text{AUC}_{(D,\tau)}$ can be interpreted to measure $p_A(C_m)$ in (7) where the threshold $\tau$ is used to binarize a soft-decision classifier to classify a data sample into $C_m$, such as MAP used by EPF-based methods.

By taking advantage of the $\text{AUC}_{(D,F)}$, $\text{AUC}_{(D,\tau)}$, and $\text{AUC}_{(F,\tau)}$ derived from a 3D ROC curve, several soft-decision classification measures can be further designed.

*1) Class Classifiability Measure:* Since the AUC of the 2D ROC curve of $(P_D, P_F)$, $\text{AUC}_{(D,F)}$, and AUC of 2D ROC curves of $(P_D, \tau)$, $\text{AUC}_{(D,\tau)}$ indicate how effective a detector is and how high the detection probability of a detector can achieve, respectively, $\text{AUC}_{(D,F)}$ and $\text{AUC}_{(D,\tau)}$ can be translated to measure the effectiveness of a classifier in class accuracy and the classifiability of a classifier, respectively. In this case, for each class $C_m$, we can define the class classifiability (CC) of a classifier as $\mathrm{AUC}_{\mathrm{CC}}(C_m)$ by

$$0 \le \mathrm{AUC}_{\mathrm{CC}}(C_m) = \mathrm{AUC}_{(D,F)}(C_m) + \mathrm{AUC}_{(D,\tau)}(C_m) \le 2 \quad (15)$$

to measure both the effectiveness and CC of a classifier to classify data samples into $C_m$.

*2) BKG Suppressibility:* On the other hand, since the AUC of the 2D ROC curve of $(P_F, \tau)$, $\mathrm{AUC}_{(F,\tau)}$, represents the probability of BKG samples classified by a classifier into a specific class, $C_m$, the smaller the value of $\mathrm{AUC}_{(F,\tau)}$, the better the classifier. In this case, to factor $\mathrm{AUC}_{(F,\tau)}$ in classification performance, we define BKG as data samples of no interest. For example, if the class of interest is $C_m$, then the data samples in all other classes, $\{C_j\}_{j \ne m}$ in BKG class will be considered as BKG. In this case, a BKG suppressibility (BS) measure for each class, $C_m$, $\mathrm{AUC}_{\mathrm{BS}}(C_m)$, can be defined by

$$-1 \le \mathrm{AUC}_{\mathrm{BS}}(C_m) = \mathrm{AUC}_{(D,F)}(C_m) - \mathrm{AUC}_{(F,\tau)}(C_m) \le 1. \quad (16)$$

*3) Joint Target Detectability With BKG:* In the definitions of CC and BS, $\mathrm{AUC}_{(D,F)}$ is used to measure the effectiveness of a classifier when $P_D$ and $P_F$ use the same threshold $\tau$ for their joint performance. In order to deal with CC and BS simultaneously, we need to consider CC and BS working together with the threshold $\tau$ as an independent parameter. Since $P_F$ is caused by the probability of misclassifying BKG samples into $C_m$, this error probability should be subtracted from $P_D$ as class accuracy $P_A$. Thus, to take care of this effect, a measure, called CC in BKG (CC-BS) by factoring $P_F$ into $P_D$, is further defined as $\mathrm{AUC}_{\mathrm{CC-BS}}(C_m)$ by

$$-1 \le \mathrm{AUC}_{\text{CC-BS}}(C_m) = \mathrm{AUC}_{(D,\tau)}(C_m) - \mathrm{AUC}_{(F,\tau)}(C_m) \le 1. \quad (17)$$

### C. Overall Classification Measures

It should be noted that the classification measures described above are defined for a specific class, $C_m$. To define the OC accuracy of a classifier over all classes, two ways can be done in a similar manner that $P_{\mathrm{AA}}$ is defined in (10) by averaging and $P_{\mathrm{OA}}$ in (9) by class weights. Accordingly, two new classification measures slightly different from those defined in [49] can be also defined.

One is AC to evaluate the average performance of a classifier by

$$\begin{aligned}\mathrm{AC} = {} & \mathrm{AAUC}_{(D,F)} + \mathrm{AAUC}_{(D,\tau)} - \mathrm{AAUC}_{(F,\tau)} \\ & + P_{\mathrm{AA}} + P_{\mathrm{APR}} \end{aligned} \quad (18)$$

with $\mathrm{AAUC}_{(D,F)}$, $\mathrm{AAUC}_{(D,\tau)}$, and $\mathrm{AAUC}_{(F,\tau)}$ defined by

$$\mathrm{AAUC}_{(D,F)} = \frac{1}{M}\sum_{m=1}^{M} \mathrm{AUC}_{(D,F)}(C_m) \quad (19)$$

$$\mathrm{AAUC}_{(D,\tau)} = \frac{1}{M}\sum_{m=1}^{M} \mathrm{AUC}_{(D,\tau)}(C_m) \quad (20)$$

$$\mathrm{AAUC}_{(\mathrm{F},\tau)} = \frac{1}{M}\sum_{m=1}^{M} \mathrm{AUC}_{(\mathrm{F},\tau)}(C_m). \quad (21)$$

The other is CWC by class weights, $\{p(C_m)\}_{m=1}^{M}$, to evaluate the weighted classification performance of a classifier by

$$\begin{aligned}\mathrm{CWC} = {} & \mathrm{CWAUC}_{(D,F)} + \mathrm{CWAUC}_{(D,\tau)} \\ & - \mathrm{CWAUC}_{(F,\tau)} + P_{\mathrm{CWA}} + P_{\mathrm{CWPR}} \end{aligned} \quad (22)$$

with $\mathrm{CWAUC}_{(D,F)}$, $\mathrm{CWAUC}_{(D,\tau)}$, and $\mathrm{CWAUC}_{(F,\tau)}$ defined by

$$\mathrm{CWAUC}_{(D,F)} = \sum_{m=1}^{M} p(C_m)\mathrm{AUC}_{(D,F)}(C_m) \quad (23)$$

$$\mathrm{CWAUC}_{(D,\tau)} = \sum_{m=1}^{M} p(C_m)\mathrm{AUC}_{(D,\tau)}(C_m) \quad (24)$$

$$\mathrm{CWAUC}_{(\mathrm{F},\tau)} = \sum_{m=1}^{M} p(C_m)\mathrm{AUC}_{(\mathrm{F},\tau)}(C_m). \quad (25)$$

Thus, when $p(C_m) = n_m/n$ is specified by SR, CWC is reduced to OC similar to OA in (9).

## V. Images to be Used for Experiments

In this section, three real HSIs were used for experiments to conduct comparative analyses among IEPF, IRTS-EPF, and IRTS-3D-CNN where 3D-CNN is implemented according to Fig. 2.

### A. Purdue Indiana Indian Pines

The first one is the Purdue Indiana Indian Pines test site with an aerial view shown in Fig. 4(a) along with its ground truth of 17 class maps in Fig. 4(b) and their class labels in Fig. 4(c). Table I tabulates the number of data samples in each class where there are four small classes with less than 100 samples, classes 9, 7, 1, and 16, and three classes with more than 1000 samples, classes, 2, 11, and 14. Thus, this scene clearly has an imbalanced class issue in classification. It is an airborne visible/infrared imaging spectrometer (AVIRIS) image scene and has a size of 145 × 145 × 200 pixel vectors with water absorption bands (bands: 104–108, 150–163, and 200). Thus, a total of 220 bands were used for experiments. It should be noted that, in many reports, 200 bands were used by excluding water absorption bands, such as in [5].

### B. Salinas

The second image dataset is Salinas shown in Fig. 5(a), which is also an AVIRIS scene. It was collected over Salinas Valley, California, USA, with a spatial resolution of 3.7 m per pixel with a spectral resolution of 10 nm. It has a size of 512 × 217 × 224 with 20 water absorption bands, 108–112, 154–167, and 224. Thus, a total of 224 bands were used for experiments. Fig. 5(b) and (c) shows the color composite of the Salinas image along with the corresponding ground-truth class labels. Unlike the Purdue data, the Salinas scene does not have an issue in imbalanced classes. According to the ground truth tabulated in Table II, the smallest class is class 13 with 916 data samples.

TABLE I
CLASS LABELS OF PURDUE'S DATA

| | | | | | |
|---|---|---|---|---|---|
| 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-mintill | Class 9 (20) | Oats | Class 15 (386) | Bldg-Grass-Trees-Drives |
| Class 4 (237) | Corn | Class 10 (972) | Soybean-notill | Class 16 (93) | Stone-Steel-Towers |
| Class 5 (483) | Grass-pasture | Class 11 (2455) | Soybean-mintill | Class 17 (10249) | BKG |
| Class 6 (730) | Grass-trees | Class 12 (593) | Soybean-clean | | |

TABLE II
CLASS LABELS OF CLASSES IN SALINAS

| | | | |
|---|---|---|---|
| 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 4 weeks |
| Class 3 (1976) | Fallow | Class 12 (1927) | Lettuce romaine 5 weeks |
| Class 4 (1394) | Fallow rough plow | Class 13 (916) | Lettuce romaine 6 weeks |
| Class 5 (2678) | Fallow smooth | Class 14 (1070) | Lettuce romaine 7 weeks |
| 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 vineyard develop | | |

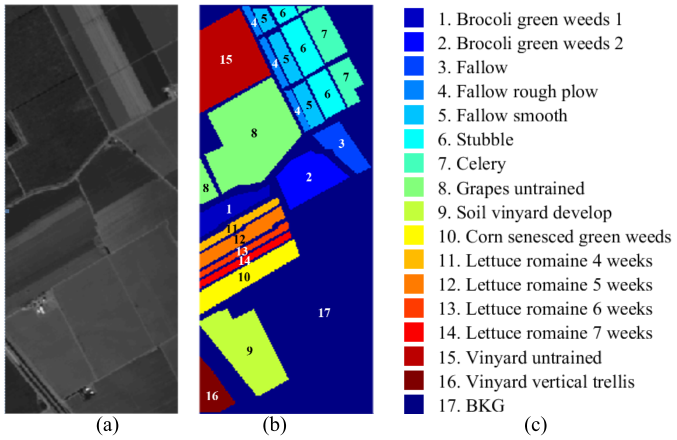


Fig. 5. Salinas and its ground truth with 16 classes. (a) Salinas scene. (b) Ground-truth image. (c) Classes by colors.

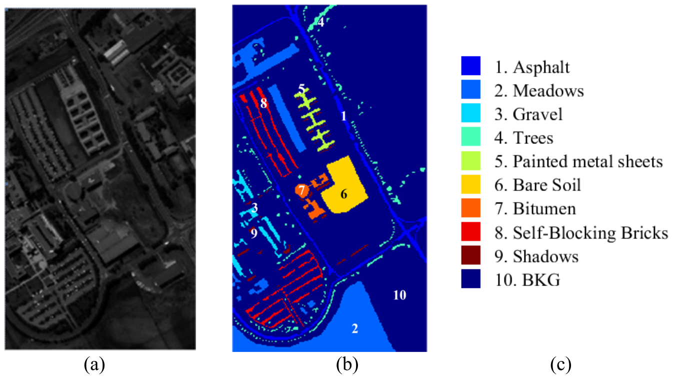


Fig. 6. University of Pavia and its ground truth with nine classes. (a) University of Pavia scene. (b) Ground-truth map. (c) Classes by colors.

### C. University of Pavia

The third HSI data is the University of Pavia image shown in Fig. 6, which is an urban area surrounding the University of Pavia, Italy. It 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 $\mu$m with a spectral resolution of 4 nm (12 most noisy channels were removed before experiments). Fig. 6(b) provides its ground-truth map of nine classes along with color class labels in Fig. 6(c). Table III also tabulates the number of data samples in parentheses collected for each class. Like the Salinas scene, this scene also has very large classes with only one small class with less than 1000 data samples, class 9 with 947 samples. However, this scene has a more complicated BKG than the other two studied scenes, as already shown in [49], where the PR of this scene was much lower than the other two scenes.

## VI. TEST CLASSIFIERS AND CLASSIFICATION MEASURES

In order to evaluate IRTS-3D-CNN, comprehensive experimental study and analysis are conducted for comparison.

### A. Classifications for Comparison

Two types of classification were compared.

*1) Machine Learning-Based SSC:* First, we need to select a set of test classifiers for comparison. Many classifiers have been developed and reported in the literature, among which SSC has emerged as one of the most promising classification techniques in HSIC. In particular, IEPF-based methods in [6] and [7] have been shown to outperform EPF, spectral–spatial composite kernel (CK) SVM-based methods developed in [50], and the SVM-MRF method proposed in [51]. Since IEPF used a fixed training sample set through all iterations, its performance was further improved by IRTS-EPF in [8], which implements RTS to reselect training samples randomly at each iteration. Accordingly, IEPF and IRTS-EPF were selected for comparison. Moreover, despite that there were four versions of SVM-EPF-based methods developed in [5], their performances were relatively the same where EPF-G-g seemed to perform a little bit better. In this case, EPF-G-g was particularly selected to implement IEPF and IRTS-EPF, denoted by IEPF-G-g and IRTS-EPF-G-g, respectively.

*2) Deep Learning-Based Classification:* Deep learning networks have been studied extensively in recent years. Numerous works have been reported in the literature. It is impossible

TABLE III
CLASS LABELS OF THE UNIVERSITY OF PAVIA

| | | | | | |
|---|---|---|---|---|---|
| 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 IV
TRAINING SAMPLE SIZES FOR PURDUE DATA, SALINAS, AND U. OF PAVIA

| Class | Purdue data | | | Salinas | | | U. of Pavia | | |
|---|---|---|---|---|---|---|---|---|---|
| | 1% | 5% | 10% | 0.1% | 0.5% | 1% | 0.1% | 0.5% | 1% |
| 1 | 6 | 23 | 25 | 3 | 16 | 35 | 4 | 25 | 48 |
| 2 | 6 | 37 | 83 | 3 | 18 | 33 | 6 | 24 | 51 |
| 3 | 6 | 39 | 78 | 3 | 16 | 34 | 5 | 25 | 47 |
| 4 | 6 | 33 | 68 | 3 | 16 | 33 | 7 | 23 | 47 |
| 5 | 7 | 34 | 79 | 4 | 17 | 34 | 4 | 23 | 47 |
| 6 | 7 | 35 | 78 | 4 | 19 | 33 | 4 | 24 | 47 |
| 7 | 6 | 14 | 14 | 3 | 17 | 33 | 4 | 23 | 47 |
| 8 | 6 | 34 | 66 | 4 | 21 | 35 | 5 | 23 | 47 |
| 9 | 6 | 10 | 10 | 4 | 17 | 34 | 4 | 24 | 47 |
| 10 | 6 | 37 | 81 | 3 | 18 | 36 | | | |
| 11 | 7 | 49 | 99 | 3 | 16 | 33 | | | |
| 12 | 7 | 35 | 73 | 3 | 16 | 33 | | | |
| 13 | 6 | 32 | 70 | 3 | 16 | 33 | | | |
| 14 | 7 | 34 | 90 | 3 | 16 | 33 | | | |
| 15 | 7 | 34 | 65 | 4 | 16 | 36 | | | |
| 16 | 6 | 32 | 46 | 4 | 16 | 33 | | | |
| Total | 102 | 512 | 1025 | 54 | 271 | 541 | 43 | 214 | 428 |

to compare them all in one article. In this article, we have selected three most recently developed state-of-the-art 3D-CNNs for a comparative analysis in the same manner that it was performed for SSC. The first one is the HybridSN developed in [17], which used 3D convolution layers to capture joint spatial–spectral features and then applied 2D convolution layers to capture spatial features to further reduce computational complexity. The second one is an A$^2$S$^2$K-ResNet derived in [30], which combined 3D residual building blocks and an efficient feature recalibration mechanism to improve the classification results. The third one is FSKNet in [33], which proposed a network with a selective kernel mechanism to adjust the receptive field size of each neuron. All of these networks have shown their superior performance over many exiting learning-based HSIC techniques and are available on their websites. Those people who are interested in repeating their experiments can find details in their websites.[1],[2],[3]

## B. Training Sample Class Size Allocation

One of the major issues in HSIC is how to deal with imbalanced issue, such as the Purdue dataset. When a class sample size is too small, such as class 9 with 20 data samples and class 7 with data 28 samples in the Purdue dataset according to Table I, using a small SR such as 0.5% of data samples to select training samples may end up with no or too few data samples that can be selected as training samples. To resolve this issue, Kang's method [38] was used to allocate the training size for each class tabulated in Table IV where the SR is selected as 10%, 5%, and 1% of the total number of data samples for the Purdue data.

[1]HybridSN: https://github.com/gokriznastic/HybridSN
[2]A$^2$S$^2$K-ResNet: https://github.com/suvojit-0 × 55aa/A2S2K-ResNet
[3]FSKNet: https://github.com/leeguandong/fsknet-for-hsi

Since Salinas and U. of Pavia are considered relatively large scenes, 1%, 0.5%, and 0.1% of the total number of data samples were randomly selected as training samples. The number of training samples for each class determined by Kang's method is also tabulated in Table IV.

## C. Classification Performance

The classification performance was evaluated by hard-decision classification measures, $P_A$ in (7), $P_{\text{OA}}$ in (9), $P_{\text{AA}}$ in (10), $P_{\text{APR}}$ in (12), and $P_{\text{OPR}}$ in (14) described in Section IV-A, and soft classification measures, AAUC$_{(D,F)}$ in (19), AAUC$_{(D,\tau)}$ in (20), AAUC$_{(F,\tau)}$ in (21), AC in (18), CWAUC$_{(D,F)}$ in (23), CWAUC$_{(D,\tau)}$ in (24), CWAUC$_{(F,\tau)}$ in (25), and CWC in (22) described in Section IV-C.

## VII. EXPERIMENTAL RESULTS AND DISCUSSIONS

### A. Purdue Indiana Indian Pines

By randomly selecting training samples for each class according to its class size allocated in Table IV, Tables V–VII tabulate $P_A$, $P_{\text{OA}}$, $P_{\text{AA}}$, $P_{\text{PR}}$, $P_{\text{OPR}}$, and $P_{\text{APR}}$ produced by IEPF-G-g, IRTS-EPF-G-g, 3D-CNN, and IRTS-3D-CNN using 10%, 5%, and 1% of data samples as training samples where the experiments were run ten times to obtain their means and standard deviations included in parentheses. When SR = 10% is large, IRTS-EPF-G-g performed slightly better than IRTS-3D-CNN, as shown in Table V, in terms of traditional hard-decision classification measures, $P_A$, $P_{\text{OA}}$, and $P_{\text{AA}}$. However, as SR is gradually reduced from 10% to 5% and 1%, IRTS began to show its competitivity in Table VIII and eventually took over IRTS-EPF, as shown in Table IX, when SR = 1% was used. Also, in all cases, 3D-CNN without IRTS produced far worse performance than the other three

TABLE V

$P_A$, $P_{\mathrm{OA}}$, $P_{\mathrm{AA}}$, $P_{\mathrm{PR}}$, $P_{\mathrm{OPR}}$, AND $P_{\mathrm{APR}}$ PRODUCED BY IEPF-G-G, IRTS-EPF-G-G, 3D-CNN, AND IRTS-3D-CNN BY USING 10% TRAINING SAMPLES FOR PURDUE DATA

| Method | IEPF-G-g | | IRTS-EPF-G-g | | 3D-CNN | | IRTS-3D-CNN | |
|---|---|---|---|---|---|---|---|---|
| Class | $P_A(C_i)$ | $P_{PR}(C_i)$ | $P_A(C_i)$ | $P_{PR}(C_i)$ | $P_A(C_i)$ | $P_{PR}(C_i)$ | $P_A(C_i)$ | $P_{PR}(C_i)$ |
| 1 | 99.13(1.12) | 53.56(7.23) | 99.13(1.12) | **57.3**(2.09) | 98.91(1.85) | 37.91(10.22) | **100**(0) | 48.28(6.49) |
| 2 | 96.83(1.36) | 66.62(4.93) | **98.19**(0.43) | 64.54(3.66) | 81.04(5.52) | 65.79(5.04) | 98.14(1.07) | **66.7**(2.23) |
| 3 | 99.06(0.73) | 70.79(3.06) | 98.9(0.81) | **74.24**(2.53) | 85.99(4.38) | 56.92(5.85) | **99.51**(0.46) | 70.41(4.52) |
| 4 | **100**(0) | 71(3.17) | **100**(0) | **73.11**(0.91) | 96.58(1.94) | 44.64(7.18) | **100**(0) | 64.01(2.32) |
| 5 | 98.96(1.47) | 27.93(5.68) | 99.71(0.42) | 25.23(2.61) | 96.29(1.06) | **32.42**(4.75) | **99.88**(0.2) | 29.73(3.59) |
| 6 | 99.44(0.75) | 52.79(6.89) | 99.7(0.16) | **53.43**(5.07) | 98.62(0.83) | 35.53(5.76) | **99.78**(0.42) | 30.2(3.99) |
| 7 | 96.07(1.13) | 66.84(11.88) | 96.43(0) | **73.7**(3.18) | **100**(0) | 19.52(6.74) | **100**(0) | 50(17.49) |
| 8 | **100**(0) | **74.22**(2.3) | **100**(0) | 73.42(0.89) | 99.1(1.23) | 70.59(5.35) | 99.98(0.07) | 66.08(2.33) |
| 9 | **100**(0) | **42.8**(2.71) | **100**(0) | 41.85(2.06) | 98.5(3.37) | 10.19(3.56) | **100**(0) | 29.73(10.07) |
| 10 | 95.91(2.62) | 75.86(4.72) | 97.81(1.46) | **77.79**(2.11) | 91.76(2.71) | 52.12(5.95) | **98.63**(0.64) | 61.91(4.45) |
| 11 | 97.16(2.32) | 79.78(2.44) | **99.23**(0.62) | **79.97**(2.58) | 81.89(3.01) | 74.42(4.11) | 98.26(0.74) | 78.64(2.34) |
| 12 | 99.09(0.46) | 59.2(9.21) | **99.46**(0.21) | **60.12**(5.63) | 90.2(5.13) | 47.68(3.82) | 99.31(0.67) | 57.39(5.61) |
| 13 | 99.22(0.34) | 80.95(2.43) | 99.17(0.24) | **82.67**(1.64) | 99.9(0.21) | 40.84(4.2) | **100**(0) | 61.88(9.08) |
| 14 | 99.2(1.64) | 29.47(2.32) | **99.9**(0.05) | **30.45**(1.22) | 94.85(3.14) | 28.54(2.62) | 99.83(0.18) | 28.02(1.42) |
| 15 | 99.95(0.16) | 11.32(1.33) | 99.97(0.08) | 11.33(1.16) | 93.26(4.25) | 16.12(1.58) | **100**(0) | **25.14**(4.91) |
| 16 | 97.53(1.52) | 45.57(10.08) | 96.34(1.36) | 46.92(7.18) | **99.89**(0.34) | **48.58**(7.36) | 99.78(0.45) | 44.74(10.93) |
| $P_{OA}$ | 98.12(0.55) | | **99.12**(0.23) | | 89.24(1.92) | | 99.07(0.26) | |
| $P_{AA}$ | 98.6(0.27) | | 99(0.22) | | 94.18(1.15) | | **99.57**(0.11) | |
| $P_{OPR}$(BKG) | 47.83(0.27) | | **48.32**(0.11) | | 43.5(0.94) | | 48.29(0.13) | |
| $P_{APR}$(BKG) | 56.8(1.29) | | **57.88**(0.82) | | 42.61(1.15) | | 50.8(2.06) | |

TABLE VI

$P_A$, $P_{\mathrm{OA}}$, $P_{\mathrm{AA}}$, $P_{\mathrm{PR}}$, $P_{\mathrm{OPR}}$, AND $P_{\mathrm{APR}}$ PRODUCED BY IEPF-G-G, IRTS-EPF-G-G, 3D-CNN, AND IRTS-3D-CNN BY USING 5% TRAINING SAMPLES FOR PURDUE DATA

| Method | IEPF-G-g | | IRTS-EPF-G-g | | 3D-CNN | | IRTS-3D-CNN | |
|---|---|---|---|---|---|---|---|---|
| Class | $P_A(C_i)$ | $P_{PR}(C_i)$ | $P_A(C_i)$ | $P_{PR}(C_i)$ | $P_A(C_i)$ | $P_{PR}(C_i)$ | $P_A(C_i)$ | $P_{PR}(C_i)$ |
| 1 | 99.57(0.92) | 48.74(12.68) | 99.57(0.92) | **51.42**(9.46) | 99.57(0.92) | 28.73(9.24) | **100**(0) | 43.3(10.23) |
| 2 | 89.59(4.03) | 64.94(4.9) | **95.81**(1.67) | **68.22**(1.84) | 64.04(6.82) | 61.28(5.94) | 95.2(2.88) | 66.4(3.05) |
| 3 | 94.83(1.77) | **72.89**(3.17) | 98.98(0.5) | 70.63(2.53) | 73.61(5.49) | 47.76(5.46) | **99.17**(0.75) | 65.08(10.22) |
| 4 | **100**(0) | 49.82(8.46) | 99.92(0.27) | **73.34**(1.95) | 95.11(3.25) | 35.04(3.3) | **100**(0) | 62.98(2.76) |
| 5 | 97.08(2.31) | 29.62(6.19) | 98.74(1.52) | 29.96(3.83) | 91.04(3.8) | 28.5(4.57) | **98.92**(1.62) | **32.47**(3.68) |
| 6 | 99.3(0.82) | 57.09(7.22) | 99.7(0.21) | **59.94**(4.06) | 95.88(1.59) | 29.91(3.75) | **99.81**(0.33) | 32.87(3.96) |
| 7 | 97.14(1.51) | 69.49(12.56) | 96.43(0) | **69.96**(9.74) | 98.57(2.5) | 14.82(6.49) | **100**(0) | 38.91(15.33) |
| 8 | **100**(0) | **76.64**(3.75) | **100**(0) | 75.66(2.92) | 96.59(1.51) | 72.08(5.87) | **100**(0) | 66.98(2.42) |
| 9 | **100**(0) | 39.73(2.32) | **100**(0) | **40.59**(2.8) | 98(3.5) | 7.04(2.47) | **100**(0) | 24.05(10.78) |
| 10 | 93.12(4.51) | 69.94(6.05) | 96.58(2.62) | **77.59**(3.34) | 78.42(8.62) | 47.09(4.06) | **97.2**(2.05) | 63.83(3.64) |
| 11 | 93.1(2.81) | 77.42(6.24) | **97.47**(2.28) | **77.55**(4.86) | 72.84(4.82) | 68.63(5.58) | 96.96(1.55) | 77.08(2.57) |
| 12 | 98.33(1.34) | **60.89**(7.47) | **99.38**(0.29) | 56.84(6.8) | 76.29(5.95) | 38.38(7.04) | 98.4(1.4) | 58.79(5.29) |
| 13 | 99.22(0.25) | 83.08(1.59) | 99.17(0.24) | **83.76**(1.64) | 98.49(1.92) | 41.1(9.56) | **99.95**(0.15) | 60(10.79) |
| 14 | 97.78(1.55) | **34.13**(2.91) | 99.34(0.65) | 32.94(1.9) | 89.82(3.27) | 31.11(2.28) | **99.62**(0.48) | 26.69(1.43) |
| 15 | 99.56(1.08) | 9.06(1.14) | **100**(0) | 9.13(0.82) | 90.26(3.39) | 14.88(2.35) | **100**(0) | **23.24**(3.29) |
| 16 | 97.96(1.86) | 37(11.46) | 96.02(3.69) | 47.5(8.3) | **99.78**(0.45) | 45.75(10.31) | **99.78**(0.45) | **47.8**(6.13) |
| $P_{OA}$ | 95.21(1.1) | | **98.14**(0.53) | | 80.28(2.69) | | 98.06(0.56) | |
| $P_{AA}$ | 97.28(0.52) | | 98.57(0.29) | | 88.64(1.52) | | **99.06**(0.32) | |
| $P_{OPR}$(BKG) | 46.41(0.54) | | **47.84**(0.26) | | 39.13(1.31) | | 47.8(0.27) | |
| $P_{APR}$(BKG) | 55.03(1.64) | | **57.82**(1.4) | | 38.26(1.36) | | 49.4(1.59) | |

classifiers. Moreover, when BKG is factored into classification, IEPF and IRTS-EPF performed much better than IRTS-3D-CNN in the sense of $P_{\mathrm{PR}}$, $P_{\mathrm{OPR}}$, and $P_{\mathrm{APR}}$. This is mainly because BKG is an unlabeled data sample and cannot be learned from training samples using 3D-CNN.

In addition to traditional hard-decision classification measures, we also used soft-decision classification measures to calculate AAUC$_{(D,F)}$, AAUC$_{(D,\tau)}$, AAUC$_{(F,\tau)}$, AC, CWAUC$_{(D,F)}$, CWAUC$_{(D,\tau)}$, CWAUC$_{(F,\tau)}$, and CWC, respectively. Their results are tabulated in Tables VIII and IX to further show that IRTS-3D-CNN and IRTS-EPF generally produced the best and second best performances with the worst performance produced by 3D-CNN, respectively.

However, as noted above, since BKG is unknown and cannot be learned from training samples using 3D-CNN, BKG suppression specified by AAUC$_{(F,\tau)}$ resulting from IRTS-3D-CNN was worse than IRTS-EPF and IEPF, as expected in Tables VIII and IX. This indicated that IEPF and IRTS-EPF had much better BS than 3D-CNN and IRTS-3D-CNN because the smaller the AAUC$_{(F,\tau)}$ value, the better the BKG to be suppressed. This fact was also reflected in PR. Nevertheless, when the number of selected training samples is small such as 1%, IRTS-3D-CNN began to show its superiority to IRTS-EPF. Overall speaking, the best performance was still IRTS-EPF with IRTS-3D-CNN as the second best. Both of these two classifiers used RTS to resample training data at each of the

TABLE VII

$P_A$, $P_{OA}$, $P_{AA}$, $P_{PR}$, $P_{OPR}$, AND $P_{APR}$ PRODUCED BY IEPF-G-G, IRTS-EPF-G-G, 3D-CNN, AND IRTS-3D-CNN BY USING 1% TRAINING SAMPLES FOR PURDUE DATA

| Method | IEPF-G-g | | IRTS-EPF-G-g | | 3D-CNN | | IRTS-3D-CNN | |
|---|---|---|---|---|---|---|---|---|
| Class | $P_A(C_i)$ | $P_{PR}(C_i)$ | $P_A(C_i)$ | $P_{PR}(C_i)$ | $P_A(C_i)$ | $P_{PR}(C_i)$ | $P_A(C_i)$ | $P_{PR}(C_i)$ |
| 1 | **99.35**(1.05) | 36.08(8.71) | 98.92(1.14) | **43.81**(7.71) | 84.78(9.5) | 12.74(5.86) | 99.13(1.52) | 34.64(5.96) |
| 2 | 48.47(14.47) | 44.93(15.64) | 66.42(15.67) | 49.84(18.64) | 29.44(10.14) | 27.18(4.78) | **78.6**(10.7) | **60.34**(8.07) |
| 3 | 52.58(15.51) | 38.19(15.47) | 82.29(6.57) | **68.25**(18.85) | 29.14(11.55) | 18.94(6.71) | **91.64**(4.72) | 59.85(11.01) |
| 4 | 78.35(20.8) | 31.32(11.09) | 84.56(13.2) | **62.95**(19.86) | 42.66(9.63) | 15.93(4.69) | **97.81**(2.96) | 56.55(10.54) |
| 5 | 85.38(10.76) | 41.2(19.14) | 83.93(9.76) | **55.34**(15.92) | 61.76(12.58) | 17.46(4.7) | **92.34**(3.83) | 35.26(7.5) |
| 6 | 86.31(8.27) | 53.61(15.89) | 96.04(3.38) | **62.87**(4.56) | 57.27(10.68) | 20.26(5.39) | **98.67**(1.86) | 34.5(9.23) |
| 7 | 97.14(1.51) | 70.11(17.98) | 96.79(1.13) | **77**(1.27) | 87.5(10.55) | 6.19(2.51) | **100**(0) | 26.13(11.62) |
| 8 | 94.14(9.84) | **83**(5.93) | 96.55(4.32) | 81.02(2.32) | 63.72(12.31) | 52.09(13.04) | **99.64**(0.85) | 71.88(4.41) |
| 9 | 99(3.16) | 27.22(9.81) | 99.5(1.58) | **37.41**(7.2) | 96(4.59) | 3.43(0.98) | **100**(0) | 21.43(10.63) |
| 10 | 68.91(12.03) | 56.07(12.6) | 86.83(6.23) | **63.5**(11.5) | 32.34(12.28) | 21.79(5.65) | **87.7**(5.08) | 56.4(9.05) |
| 11 | 57.26(13.12) | 58.12(14.11) | 77.26(8.69) | **78.2**(12.73) | 40.79(10.1) | 46.95(8.35) | **77.84**(6.78) | 76.95(5.34) |
| 12 | 65.52(15.53) | 21.15(8.8) | 89.66(13.24) | 37.33(20.36) | 28.62(12.24) | 14.1(5.3) | **90.35**(6.57) | **49.53**(8.62) |
| 13 | 98.34(0.66) | 86.09(1.88) | 98.63(0.5) | **86.63**(2.5) | 86.59(9.02) | 24.5(4.96) | **100**(0) | 65.13(10.33) |
| 14 | 86.19(11.61) | **39.48**(4.53) | 92.32(4.6) | 37.9(3.92) | 61.94(8.71) | 25.39(3.07) | **97.91**(3.27) | 26.76(2.41) |
| 15 | 85.8(8.56) | 7.41(1.72) | 97.12(5.23) | 7.89(1.78) | 48.08(14.98) | 7.49(1.76) | **98.08**(5.62) | **16.37**(3.39) |
| 16 | 84.19(9.95) | 68.69(24.77) | 75.16(13.42) | **79.76**(10.07) | 87.1(10.78) | 30.99(15.09) | **99.79**(0.68) | 42.23(9.01) |
| $P_{OA}$ | 68.93(5.73) | | 83.71(1.73) | | 44.69(4.99) | | **88.45**(2.86) | |
| $P_{AA}$ | 80.43(3.21) | | 88.87(1.67) | | 58.61(6.41) | | **94.34**(1.38) | |
| $P_{OPR}$(BKG) | 33.6(2.79) | | 40.81(0.85) | | 21.79(2.43) | | **43.12**(1.4) | |
| $P_{APR}$(BKG) | 47.67(2.47) | | **58.11**(2.64) | | 21.59(3.27) | | 45.87(1.42) | |

TABLE VIII

AC FOR IEPF, IRTS-EPF, 3D-CNN, AND IRTS-3D-CNN IN TERMS OF 3D ROC ANALYSIS, $P_{AA}$, $P_{OA}$, AND $P_{OPR}$ FOR PURDUE DATA

| | SR% | IEPF-G-g | IRTS-EPF-G-g | 3D-CNN | IRTS-3D-CNN |
|---|---|---|---|---|---|
| $AAUC_{(D,F)}$ | 10 | 0.9923(0.0005) | 0.9928(0.0004) | 0.9841(0.0025) | **0.9943**(0.0007) |
| | 5 | 0.9902(0.001) | 0.9924(0.0005) | 0.9747(0.0036) | **0.9934**(0.0009) |
| | 1 | 0.9461(0.0091) | 0.9753(0.0047) | 0.8926(0.0217) | **0.9832**(0.0032) |
| $AAUC_{(D,\tau)}$ | 10 | 0.8160(0.0059) | 0.8157(0.0082) | 0.7852(0.0101) | **0.8887**(0.0108) |
| | 5 | 0.8094(0.0073) | 0.8162(0.0067) | 0.7638(0.0091) | **0.8747**(0.0201) |
| | 1 | 0.7041(0.0246) | 0.7663(0.0075) | 0.6358(0.0368) | **0.8406**(0.0176) |
| $AAUC_{(F,\tau)}$ | 10 | 0.1596(0.0032) | **0.1577**(0.0027) | 0.3521(0.014) | 0.2017(0.0159) |
| | 5 | 0.1674(0.0043) | **0.1590**(0.0030) | 0.3537(0.0146) | 0.2177(0.0362) |
| | 1 | 0.1817(0.0074) | **0.1771**(0.0051) | 0.3639(0.0172) | 0.2108(0.0141) |
| $P_{AA}$ | 10 | 0.9860(0.0027) | 0.9900(0.0022) | 0.9417(0.0115) | **0.9957**(0.0011) |
| | 5 | 0.9729(0.0052) | 0.9857(0.0029) | 0.8864(0.0152) | **0.9906**(0.0032) |
| | 1 | 0.8043(0.0321) | 0.8887(0.0167) | 0.5861(0.0641) | **0.9434**(0.0138) |
| $P_{APR}$ | 10 | 0.5679(0.0129) | **0.5788**(0.0082) | 0.4261(0.0115) | 0.508(0.0206) |
| | 5 | 0.5503(0.0165) | **0.5781**(0.014) | 0.3826(0.0136) | 0.494(0.0159) |
| | 1 | 0.4767(0.0247) | **0.5811**(0.0264) | 0.2159(0.0327) | 0.4587(0.0142) |
| AC | 10 | 3.2026 | **3.2196** | 2.785 | 3.185 |
| | 5 | 3.1554 | **3.2135** | 2.6538 | 3.135 |
| | 1 | 2.7495 | **3.0343** | 1.9665 | 3.0151 |

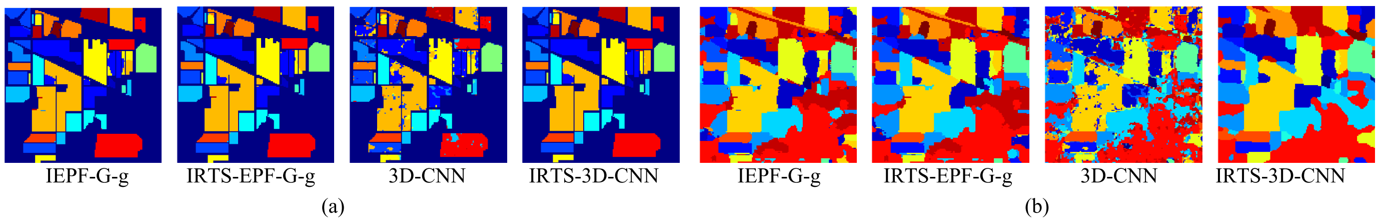


Fig. 7. Classification maps of IEPF-G-g, IRTS-EPF-G-g, 3D-CNN, and IRTS-3D-CNN for the Purdue data using 10% of training samples. (a) Without BKG. (b) With BKG.

new iterations. Unfortunately, 3D-CNN came with a much worst performance due to its inability in learning training samples without an iterative process.

In order to provide a visual inspection of test methods, Figs. 7–9 show their classification maps without/with BKG for the Purdue data using 10%–5% and 1% of training samples, respectively. By looking at classes 10 and 11, IRTS-EPF-G-g and IRTS-3D-CNN performed the best without BKG using 10% and 1% training samples, respectively. In particular, IRTS-3D-CNN began to show its superior performance when the training sample size decreased. However, when BKG is factored in, 3D-CNN without an iterative process performed

TABLE IX
CWC FOR IEPF, IRTS-EPF, 3D-CNN, AND IRTS-3D-CNN IN TERMS OF 3D ROC ANALYSIS, $P_{AA}$, $P_{OA}$, AND $P_{OPR}$ FOR PURDUE DATA

| | SR% | IEPF-G-g | IRTS-EPF-G-g | 3D-CNN | IRTS-3D-CNN |
|---|---|---|---|---|---|
| $CWAUC_{(D,F)}$ | 10 | 0.9887(0.0007) | 0.9896(0.0007) | 0.9712(0.004) | **0.9903**(0.0011) |
| | 5 | 0.9857(0.0026) | **0.9895**(0.0008) | 0.9567(0.0061) | 0.9883(0.0018) |
| | 1 | 0.9038(0.0264) | 0.9589(0.0063) | 0.8588(0.0195) | **0.9661**(0.0085) |
| $CWAUC_{(D,\tau)}$ | 10 | 0.8063(0.0205) | 0.8105(0.0222) | 0.7284(0.0099) | **0.8773**(0.0149) |
| | 5 | 0.7958(0.0177) | 0.8072(0.021) | 0.6986(0.0136) | **0.8515**(0.0244) |
| | 1 | 0.6419(0.0427) | 0.7397(0.019) | 0.5899(0.0312) | **0.7823**(0.0244) |
| $CWAUC_{(F,\tau)}$ | 10 | 0.1729(0.0049) | **0.1708**(0.0043) | 0.3553(0.0119) | 0.2222(0.0194) |
| | 5 | 0.1802(0.0078) | **0.1714**(0.0054) | 0.3564(0.0155) | 0.2294(0.0354) |
| | 1 | **0.2041**(0.0094) | **0.2041**(0.0156) | 0.3723(0.0227) | 0.2285(0.0165) |
| $P_{CWA}$ | 10 | 0.9812(0.0055) | **0.9912**(0.0023) | 0.8924(0.0192) | 0.9907(0.0026) |
| | 5 | 0.9521(0.011) | **0.9814**(0.0053) | 0.8027(0.0269) | 0.9806(0.0056) |
| | 1 | 0.6893(0.0573) | 0.8371(0.0173) | 0.4469(0.0499) | **0.8845**(0.0286) |
| $P_{CWPR}$ | 10 | 0.4783(0.0027) | **0.4832**(0.0011) | 0.435(0.0094) | 0.4829(0.0013) |
| | 5 | 0.4641(0.0054) | **0.4784**(0.0026) | 0.3913(0.0131) | 0.478(0.0027) |
| | 1 | 0.336(0.0279) | 0.4081(0.0084) | 0.2179(0.0243) | **0.4312**(0.0139) |
| CWC | 10 | 3.0816 | 3.1038 | 2.6717 | **3.1189** |
| | 5 | 3.0175 | **3.0850** | 2.4930 | 3.0690 |
| | 1 | 2.3669 | 2.7397 | 1.7412 | **2.8355** |

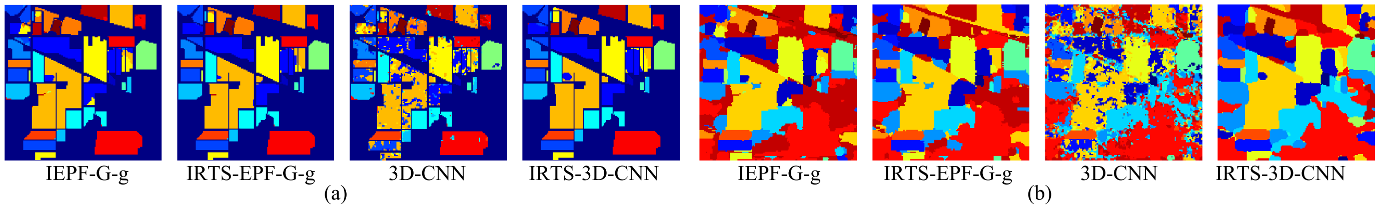


Fig. 8. Classification maps of IEPF-G-g, IRTS-EPF-G-g, 3D-CNN, and IRTS-3D-CNN for the Purdue data using 5% of training samples. (a) Without BKG. (b) With BKG.

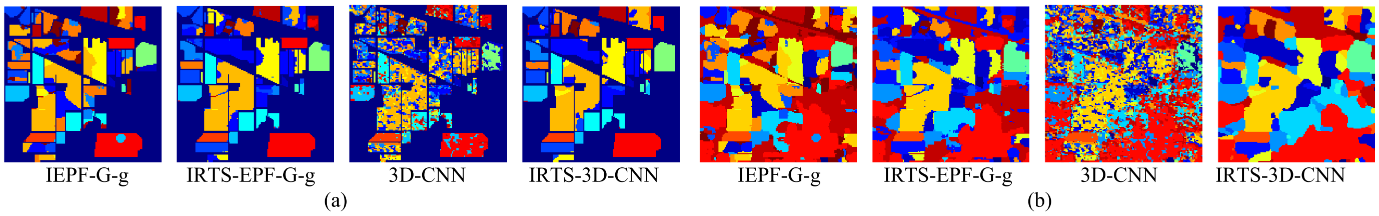


Fig. 9. Classification maps of IEPF-G-g, IRTS-EPF-G-g, 3D-CNN, and IRTS-3D-CNN for the Purdue data using 1% of training samples. (a) Without BKG. (b) With BKG.

the worst, but its IRTS-3D-CNN significantly improved and performed even better than IEPF-G-g and IRTS-EPF-G-g.

## B. Salinas

Unlike the Purdue data, Salinas has a large number of data samples. In this case, we selected small training sizes with 1%, 0.5%, and 0.1% of total data samples as training samples, respectively, according to the training class sizes allocated in Table IV by Kang's method. Tables X–XII tabulate $P_A$, $P_{OA}$, $P_{AA}$, $P_{PR}$, $P_{OPR}$, and $P_{APR}$ produced by IEPF-G-g, IRTS-EPF, 3D-CNN, and IRTS-3D-CNN where the results were calculated by tenfold experiments with standard deviations included in parentheses. Interestingly, IRTS-EPF was shown to be the best classifier, while 3D-CNN was the worst classifier because IRTS-3D-CNN did not perform well for class 8 and class 15. Since class 8 is the largest class, its accuracy has a strong impact on $P_{OA}$ of IRTS-3D-CNN. However, like the Purdue experiments, when SR was reduced to 0.1%, IRTS-3D-CNN began to show its superior performance over IEPF, as shown in Table XII.

Tables XIII and XIV also tabulate AC and CWC, respectively. As expected, using a large number of training samples, IRTS-EPF had the best performance followed by IEPF, while IRTS-3D-CNN had the worst performance. These results demonstrated that IRTS-3D-CNN lost its competitivity to IRTS-EPF-G-g when the number of training samples was large.

In order to provide a visual inspection of test methods, Figs. 10–12 show their classification maps without/with BKG for the Salinas data using 1%, 0.5%, and 0.1% of training samples, respectively. If we looked at classes 8 and 15, IRTS-EPF-G-g performed the best, and 3D-CNN was the worst with/without BKG in all cases. However, 3D-CNN was significantly improved after it was implemented as IRTS-3D-CNN and performed comparably to IRTS-EPF-G-g.

## C. University of Pavia

Following similar experiments conducted for Salinas, Table XV–XVII tabulate $P_A$, $P_{OA}$, $P_{AA}$, $P_{PR}$, $P_{OPR}$, and $P_{APR}$ produced by IEPF-G-g, IRTS-EPF, 3D-CNN, and

TABLE X

$P_A$, $P_{OA}$, $P_{AA}$, $P_{PR}$, $P_{OPR}$, AND $P_{APR}$ PRODUCED BY IEPF-G-G, 3D-CNN, AND IRTS-3D-CNN BY USING 1% TRAINING SAMPLES FOR SALINAS

| Method | IEPF-G-g | | IRTS-EPF-G-g | | 3D-CNN | | IRTS-3D-CNN | |
|---|---|---|---|---|---|---|---|---|
| Class | $P_A(C_i)$ | $P_{PR}(C_i)$ | $P_A(C_i)$ | $P_{PR}(C_i)$ | $P_A(C_i)$ | $P_{PR}(C_i)$ | $P_A(C_i)$ | $P_{PR}(C_i)$ |
| 1 | **100**(0) | **76.27**(0.51) | **100**(0) | 75.62(0.38) | 98.74(1.72) | 72.35(6.97) | 99.95(0.16) | 72.51(3.22) |
| 2 | 99.71(0.31) | 60.97(4.66) | 99.86(0.11) | 63.56(3.48) | 99.5(0.79) | **64.84**(8.51) | **100**(0) | 63.4(8.91) |
| 3 | 99.47(1.03) | 12.84(1.09) | 99.91(0.19) | 12.86(1.19) | 93.83(5.12) | **17.47**(3.84) | **99.92**(0.26) | 16.66(2.87) |
| 4 | 99.78(0.25) | **32.03**(3.12) | **99.86**(0.12) | 31.21(2.03) | 98.61(1.41) | 27.86(2.52) | 99.81(0.44) | 22.5(4.28) |
| 5 | 98.62(0.26) | 55.69(10.46) | 98.82(0.4) | 56.6(5.53) | 97.28(1.84) | 55.01(9.19) | **99.2**(0.87) | **63.52**(6.37) |
| 6 | 99.72(0.45) | **83.98**(0.88) | 99.72(0.64) | 83.7(1.21) | 99.52(0.47) | 80.52(3.75) | **99.83**(0.27) | 80.67(3.29) |
| 7 | 99.7(0.18) | **83.22**(0.89) | 99.75(0.04) | 83.2(0.55) | 99.37(0.56) | 81.48(5.54) | **99.84**(0.47) | 79.42(1.09) |
| 8 | 91.09(4.16) | **88.26**(4.83) | **97.09**(1.06) | 87.54(13.86) | 64.03(19.02) | 72.67(8.94) | 88.36(7.92) | 85.08(3.26) |
| 9 | 98.97(1.19) | **39.81**(6.37) | 99.21(0.67) | 39.29(3.24) | 99.18(0.53) | 26.73(1.83) | **99.9**(0.21) | 26.39(2.1) |
| 10 | 99.19(1.14) | 18.68(1.77) | **99.88**(0.2) | 20.57(7.31) | 91.91(3.24) | 54.86(10.38) | 97.99(1.42) | **59.44**(6.18) |
| 11 | 99.28(0.43) | **24.18**(6.56) | **99.63**(0.22) | 21.43(4.29) | 97.15(1.63) | 18.69(3.68) | 99.62(0.4) | 21.35(2.56) |
| 12 | 99.99(0.03) | 35.82(15.87) | **100**(0) | **40.76**(12.42) | 99.96(0.1) | 19.57(4.38) | 99.99(0.03) | 22.42(4.62) |
| 13 | 99.3(0.5) | 65.93(1.64) | 99.29(0.44) | **66.53**(0.94) | 98.76(1.15) | 50.77(6.46) | **99.37**(1) | 64.37(2.8) |
| 14 | 98.38(1.34) | 64.42(4.61) | 98.16(1.8) | **66.14**(4.96) | 95.52(1.92) | 48.81(5.23) | **99.54**(0.62) | 58.97(4.53) |
| 15 | 92.37(4.88) | 86.23(6.52) | **96.92**(1.89) | **94.18**(1.42) | 73.43(22.05) | 57.99(9.17) | 94.32(2.65) | 81.05(8.83) |
| 16 | 98.4(0.72) | **82.57**(7.32) | 98.9(0.51) | 74.89(8.14) | 97.89(0.53) | 46.09(8.53) | **99.65**(0.39) | 51.42(12.84) |
| $P_{OA}$ | 96.69(1.19) | | **98.68**(0.45) | | 87.57(2.72) | | 96.57(1.77) | |
| $P_{AA}$ | 98.38(0.45) | | **99.19**(0.25) | | 94.05(1.39) | | 98.58(0.57) | |
| $P_{OPR}$(BKG) | 47.11(0.58) | | **48.08**(0.22) | | 42.66(1.33) | | 47.05(0.86) | |
| $P_{APR}$(BKG) | 56.93(1.22) | | **57.38**(0.91) | | 49.73(1.15) | | 54.32(1.07) | |

TABLE XI

$P_A$, $P_{OA}$, $P_{AA}$, $P_{PR}$, $P_{OPR}$, AND $P_{APR}$ PRODUCED BY IEPF-G-G, 3D-CNN, AND IRTS-3D-CNN BY USING 0.5% TRAINING SAMPLES FOR SALINAS

| Method | IEPF-G-g | | IRTS-EPF-G-g | | 3D-CNN | | IRTS-3D-CNN | |
|---|---|---|---|---|---|---|---|---|
| Class | $P_A(C_i)$ | $P_{PR}(C_i)$ | $P_A(C_i)$ | $P_{PR}(C_i)$ | $P_A(C_i)$ | $P_{PR}(C_i)$ | $P_A(C_i)$ | $P_{PR}(C_i)$ |
| 1 | 99.93(0.15) | 75.77(1.05) | **100**(0.02) | **76.9**(1.13) | 95.17(7.29) | 70.73(14.84) | **100**(0) | 71.74(6.71) |
| 2 | 99.56(0.51) | 62.7(7.89) | 99.66(0.3) | **66.3**(5.07) | 98.68(2.38) | 60.01(11.42) | **100**(0) | 60.21(6.7) |
| 3 | 99.03(1.55) | 12.48(1.86) | 99.85(0.18) | 12.7(1.37) | 95.8(2.84) | 16.83(2.52) | **99.94**(0.19) | 15.52(1.93) |
| 4 | 99.71(0.35) | 33.18(2.84) | 99.75(0.23) | **34.53**(1.67) | 99.27(0.8) | 24.14(2.1) | **99.76**(0.36) | 22.28(3.9) |
| 5 | 98.38(0.77) | 51.79(7.58) | **98.64**(0.43) | 55.96(9.34) | 95(3.62) | 56.18(6.12) | 98.56(0.94) | **64.92**(8.88) |
| 6 | 99.67(0.76) | 83.35(1.17) | 99.59(0.78) | **84.11**(1.39) | 99.32(0.8) | 80.27(6.37) | **99.99**(0.01) | 80.2(4.72) |
| 7 | 99.66(0.11) | 83.26(0.68) | 99.42(0.25) | **84.22**(0.86) | 99.41(0.55) | 80.17(6.56) | **99.92**(0.23) | 80.7(3.29) |
| 8 | 85.92(6.07) | 64.92(22.55) | **95.49**(3.05) | 80.13(21.66) | 78.26(10.06) | 71.26(4.56) | 88.25(5.49) | **82.45**(5.21) |
| 9 | 99.4(0.53) | 38.75(2.01) | 99.24(0.58) | **39.49**(5) | 99.01(0.87) | 26.68(2.57) | **99.85**(0.2) | 26.5(2.22) |
| 10 | 95.51(3.29) | 46.88(26.52) | **99.26**(1.23) | 28.4(25.55) | 91.15(2.42) | 55.53(9.04) | 96.88(1.38) | **56.68**(9.06) |
| 11 | 98.83(0.93) | 24.66(8.54) | 98.91(0.41) | **30.02**(1.76) | 93.4(5.36) | 18.55(5.74) | **99.27**(1.02) | 21.97(3.38) |
| 12 | **100**(0) | 32.5(7.61) | 99.99(0.02) | **33.79**(5) | 99.86(0.14) | 21.19(3.32) | 99.96(0.11) | 23.41(3.51) |
| 13 | 99.05(0.6) | 65.05(3.15) | 98.74(0.43) | **67.53**(0.84) | 98.16(3.33) | 55.64(5.46) | **99.5**(0.79) | 63.64(6.14) |
| 14 | 95.36(4.9) | 61.2(14.07) | 97.71(2.67) | **65.12**(4.38) | 96.61(1.06) | 45.45(8.14) | **98.72**(1.4) | 63.38(3.99) |
| 15 | 83.58(12.14) | 78.6(6.77) | **93.89**(2.28) | **92.63**(4.97) | 61.15(14.41) | 64.95(8.38) | 86.04(10.7) | 78.98(6.16) |
| 16 | 97.11(4.33) | 65.54(26.76) | 98.08(0.54) | **87.16**(3.03) | 96.24(2.86) | 53.69(13.08) | **98.41**(1.74) | 52.99(10.22) |
| $P_{OA}$ | 94.09(1.78) | | **97.79**(0.63) | | 88.47(2.04) | | 95.28(2.24) | |
| $P_{AA}$ | 96.92(0.95) | | **98.64**(0.28) | | 93.53(1.42) | | 97.82(1) | |
| $P_{OPR}$(BKG) | 45.84(0.87) | | **47.64**(0.31) | | 43.1(0.99) | | 46.42(1.09) | |
| $P_{APR}$(BKG) | 55.04(1.78) | | **58.69**(1.15) | | 50.08(0.96) | | 54.1(0.99) | |

IRTS-3D-CNN with randomly selecting SR = 1%, 0.5%, and 0.1% of data samples as training samples according to Table IV, respectively. Since this dataset also contains a large number of data samples, the results were similar to those obtained for Salinas where the best performance was IRTS-EPF and IEPF followed by IRTS-3D-CNN with 3D-CNN being the worst when SR was over 1%. However, as SR was decreased below 1%, IRTS-3D-CNN began to outperform IRTS-EPF and IEPF, and became the best classifier, which showed significantly better improvement as SR = 0.1%. These results also demonstrated that IRTS-3D-CNN has tremendous advantages over IRTS-EPF and IEPF when SR became very small.

Tables XVIII and XIX also tabulate AC and CWC, respectively. Once again, IRTS-3D-CNN had the highest values when SR was smaller than 1%. In other words, when SR became small, the classifiers using RTS demonstrated their superiority to the classifiers without using RTS. Specifically, as SR was reduced from 0.5% to 0.1%, the performance of EPF dropped more rapidly than CNN methods.

In order to provide a visual inspection of test methods, Figs. 13–15 show their classification maps without/with BKG for the U. of Pavia data using 1%, 0.5%, and 0.1% of training samples, respectively. As pointed out in its data description, U. of Pavia has very complicated BKG as evidenced by comparing the classification maps with and without BKG in Figs. 13–15 where the classification maps with BKG showed that most of the data samples were mixed with BKG, which led to very poor PR of less than 25%, which was nearly twice worse than the PR results produced by two other datasets,

TABLE XII
$P_A$, $P_{\mathrm{OA}}$, $P_{\mathrm{AA}}$, $P_{\mathrm{PR}}$, $P_{\mathrm{OPR}}$, AND $P_{\mathrm{APR}}$ PRODUCED BY IEPF-G-G, 3D-CNN, AND IRTS-3D-CNN BY USING 0.1% TRAINING SAMPLES FOR SALINAS

| Method | IEPF-G-g | | IRTS-EPF-G-g | | 3D-CNN | | IRTS-3D-CNN | |
|---|---|---|---|---|---|---|---|---|
| Class | $P_A(C_i)$ | $P_{PR}(C_i)$ | $P_A(C_i)$ | $P_{PR}(C_i)$ | $P_A(C_i)$ | $P_{PR}(C_i)$ | $P_A(C_i)$ | $P_{PR}(C_i)$ |
| 1 | 99.52(1.17) | 75.62(2.95) | **99.8**(0.38) | **76.23**(1.55) | 92.13(18.7) | 57.81(17) | 99.35(1.73) | 68.62(7) |
| 2 | 97.85(2.74) | 66.53(9.68) | 98.65(2.64) | **70.31**(6.34) | 94.37(6.61) | 59.2(11.18) | **99.91**(0.28) | 63.77(7.63) |
| 3 | 75.07(22.02) | 13.17(5.55) | 81.76(22.38) | 14.2(5.36) | 80.05(13.73) | **14.46**(3.55) | **96.58**(5.22) | 14.45(4) |
| 4 | 94.68(9.62) | **39.07**(10.13) | 99.1(1.52) | 37.48(3.69) | 97.47(2.91) | 27.69(6) | **99.89**(0.11) | 26.02(5.18) |
| 5 | 98.54(0.48) | 38.1(16.22) | 97.93(1.11) | 41.44(9.03) | 92(6.13) | 51.3(7.89) | **98.82**(0.76) | **57.48**(10.1) |
| 6 | 96.74(3) | **86.46**(3.3) | 98.28(1.81) | 84.42(1.55) | 95.87(2.81) | 81.78(3.7) | **99.42**(0.8) | 82.43(0.83) |
| 7 | 99.09(0.86) | 81.34(7.7) | 99.46(0.28) | **83.67**(1.74) | 97.63(2.5) | 80.96(8.89) | **99.94**(0.06) | 78.33(2.38) |
| 8 | 55.73(17.35) | 61.15(29.75) | **74.13**(13.9) | 55.92(24.61) | 56.46(17.33) | 60.93(14.66) | 69.16(19.42) | **75.18**(12.12) |
| 9 | 97.01(3.41) | **43.01**(8.15) | 98.18(1.07) | 39.43(4.64) | 98.43(1.12) | 26.04(3.53) | **99.6**(0.5) | 28.6(3.59) |
| 10 | **89.64**(10.33) | 49.36(34.48) | 81.47(12.72) | **65.19**(31.69) | 72.78(19.29) | 52.1(13.92) | 84.49(10) | 64.65(13.72) |
| 11 | 96.1(4.48) | **30.78**(11.43) | 97.43(5.23) | 28.37(6.63) | 91.33(6.4) | 17.2(5.08) | **98.55**(1.55) | 19.41(8.27) |
| 12 | 97.13(7.34) | 31.02(17.74) | 98.53(3.62) | 29.3(7.05) | 97.57(5.47) | 26.17(15.09) | **99.98**(0.05) | **32.37**(22.41) |
| 13 | **99.08**(0.46) | 64.32(2.27) | 98.81(0.37) | **67.03**(0.9) | 94.77(5.55) | 48.65(8.91) | 98.47(1.39) | 60.09(11.1) |
| 14 | 91.61(5.94) | 65.2(9.59) | 92.36(5.66) | **68.39**(5.15) | 89.5(6.99) | 46.87(14.1) | **96.45**(3.27) | 60.49(6.11) |
| 15 | 75.91(19.29) | 53.97(8.71) | **82.74**(9.74) | **69.88**(11.86) | 59.48(15.35) | 44.84(8.92) | 81.05(15.62) | 57.44(13.72) |
| 16 | 86.58(10.91) | **73.47**(25.12) | 92.63(5.85) | 68.26(34.91) | 89.57(9.02) | 57.17(14.73) | **98.24**(1.17) | 55.08(16.7) |
| $P_{\mathrm{OA}}$ | 84.18(2.72) | | 89.39(3.24) | | 80.43(4.33) | | **89.6**(4.49) | |
| $P_{\mathrm{AA}}$ | 90.64(2.08) | | 93.2(2) | | 87.46(2.74) | | **94.99**(1.75) | |
| $P_{\mathrm{OPR}}$(BKG) | 41.02(1.33) | | 43.55(1.58) | | 39.19(2.11) | | **43.65**(2.19) | |
| $P_{\mathrm{APR}}$(BKG) | 54.53(2.67) | | **56.22**(2.91) | | 47.07(1.54) | | 52.77(2.18) | |

TABLE XIII
AC FOR IEPF, IRTS-EPF, 3D-CNN, AND IRTS-3D-CNN IN TERMS OF 3D ROC ANALYSIS, $P_{\mathrm{AA}}$, $P_{\mathrm{OA}}$, AND $P_{\mathrm{OPR}}$ FOR SALINAS

| | SR% | IEPF-G-g | IRTS-EPF-G-g | 3D-CNN | IRTS-3D-CNN |
|---|---|---|---|---|---|
| $\mathrm{AAUC}_{(\mathrm{D,F})}$ | 1 | 0.9871(0.0009) | 0.9888(0.0007) | 0.9801(0.0037) | **0.9892**(0.0019) |
| | 0.5 | 0.985(0.0019) | 0.9878(0.0005) | 0.9778(0.0048) | **0.9892**(0.0024) |
| | 0.1 | 0.967(0.0098) | 0.9796(0.0071) | 0.9631(0.0078) | **0.9827**(0.0067) |
| $\mathrm{AAUC}_{(\mathrm{D,T})}$ | 1 | 0.9231(0.0059) | **0.9313**(0.0027) | 0.7531(0.0358) | 0.9015(0.0253) |
| | 0.5 | 0.9066(0.0115) | **0.9213**(0.0055) | 0.7502(0.0054) | 0.889(0.0293) |
| | 0.1 | 0.848(0.019) | **0.87**(0.0168) | 0.6848(0.0619) | 0.8599(0.0438) |
| $\mathrm{AAUC}_{(\mathrm{F,T})}$ | 1 | 0.1185(0.0051) | **0.1124**(0.0047) | 0.411(0.0285) | 0.2311(0.0612) |
| | 0.5 | 0.1196(0.0103) | **0.1109**(0.0051) | 0.4103(0.0239) | 0.2508(0.0551) |
| | 0.1 | 0.117(0.0061) | **0.1154**(0.01) | 0.4315(0.0333) | 0.2275(0.0593) |
| $P_{\mathrm{AA}}$ | 1 | 0.9837(0.0046) | **0.9919**(0.0025) | 0.9404(0.0139) | 0.9858(0.0057) |
| | 0.5 | 0.9692(0.0095) | **0.9864**(0.0028) | 0.9353(0.0142) | 0.9781(0.01) |
| | 0.1 | 0.9064(0.0208) | 0.932(0.02) | 0.8746(0.0274) | **0.9499**(0.0175) |
| $P_{\mathrm{APR}}$ | 1 | 0.5693(0.0122) | **0.5738**(0.0091) | 0.4973(0.0115) | 0.5432(0.0107) |
| | 0.5 | 0.5504(0.0178) | **0.5869**(0.0115) | 0.5008(0.0096) | 0.541(0.0099) |
| | 0.1 | 0.5453(0.0267) | **0.5622**(0.0291) | 0.4707(0.0154) | 0.5277(0.0218) |
| AC | 1 | 3.3447 | **3.3715** | 2.7599 | 3.1886 |
| | 0.5 | 3.2916 | **3.3713** | 2.7548 | 3.1465 |
| | 0.1 | 3.1497 | **3.2284** | 2.5617 | 3.0927 |

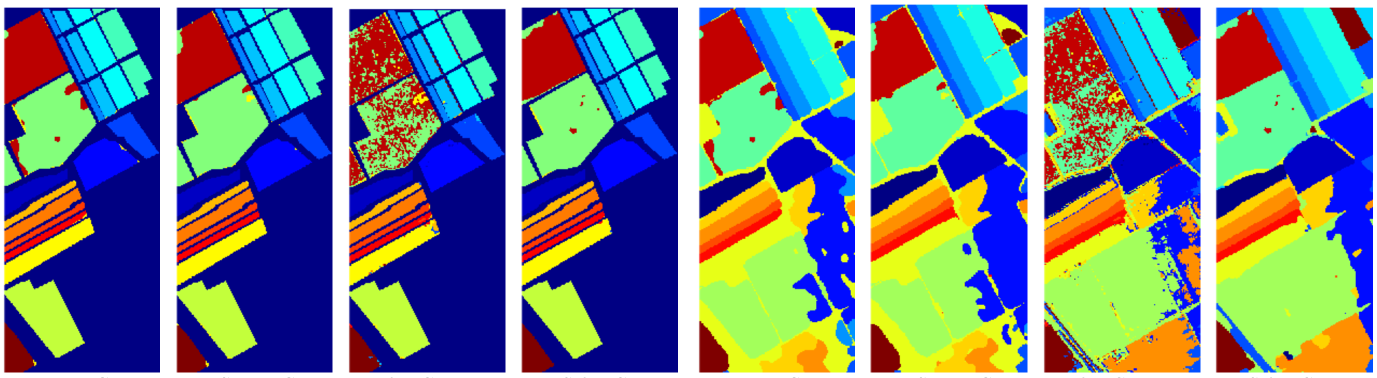


Fig. 10. Classification maps of IEPF-G-g, IRTS-EPF-G-g, 3D-CNN, and IRTS-3D-CNN for Salinas using 1% of training samples. (a) Without BKG. (b) With BKG.

TABLE XIV
CWC FOR IEPF, IRTS-EPF, 3D-CNN, AND IRTS-3D-CNN IN TERMS OF 3D ROC ANALYSIS, $P_{AA}$, $P_{OA}$, AND $P_{OPR}$ FOR SALINAS

| | SR% | IEPF-G-g | IRTS-EPF-G-g | 3D-CNN | IRTS-3D-CNN |
|---|---|---|---|---|---|
| $\mathrm{CWAUC}_{(D,F)}$ | 1 | 0.9866(0.0016) | **0.9892**(0.0006) | 0.9727(0.0043) | 0.9875(0.0025) |
| | 0.5 | 0.9785(0.0079) | **0.9870**(0.0017) | 0.9746(0.0056) | 0.9867(0.0037) |
| | 0.1 | 0.9426(0.0205) | 0.9701(0.0109) | 0.9551(0.0093) | **0.9705**(0.0205) |
| $\mathrm{CWAUC}_{(D,\tau)}$ | 1 | 0.9143(0.0113) | **0.9298**(0.0051) | 0.7398(0.0448) | 0.8773(0.0249) |
| | 0.5 | 0.8849(0.0183) | **0.9187**(0.0091) | 0.7597(0.0593) | 0.8682(0.0281) |
| | 0.1 | 0.8039(0.026) | **0.8431**(0.0191) | 0.6622(0.0643) | 0.8137(0.0429) |
| $\mathrm{CWAUC}_{(F,\tau)}$ | 1 | 0.1299(0.0086) | **0.1207**(0.0082) | 0.3957(0.0362) | 0.2461(0.0674) |
| | 0.5 | 0.1327(0.0089) | **0.127**(0.0078) | 0.4204(0.0249) | 0.2612(0.0542) |
| | 0.1 | 0.1335(0.0058) | **0.1319**(0.0118) | 0.4238(0.0339) | 0.2372(0.0666) |
| $P_{CWA}$ | 1 | 0.9669(0.0119) | **0.9868**(0.0045) | 0.8757(0.0272) | 0.9657(0.0177) |
| | 0.5 | 0.9409(0.0178) | **0.9779**(0.0063) | 0.8847(0.0204) | 0.9528(0.0224) |
| | 0.1 | 0.8418(0.0272) | 0.8939(0.0325) | 0.8043(0.0433) | **0.896**(0.0449) |
| $P_{CWPR}$ | 1 | 0.4711(0.0058) | **0.4808**(0.0022) | 0.4266(0.0133) | 0.4705(0.0086) |
| | 0.5 | 0.4584(0.0087) | **0.4764**(0.0031) | 0.431(0.0099) | 0.4642(0.0109) |
| | 0.1 | 0.4101(0.0133) | 0.4355(0.0158) | 0.3919(0.0211) | **0.4365**(0.0219) |
| CWC | 1 | 3.2090 | **3.2659** | 2.6191 | 3.0549 |
| | 0.5 | 3.1300 | **3.2330** | 2.6296 | 3.0107 |
| | 0.1 | 2.8649 | **3.0107** | 2.3897 | 2.8795 |

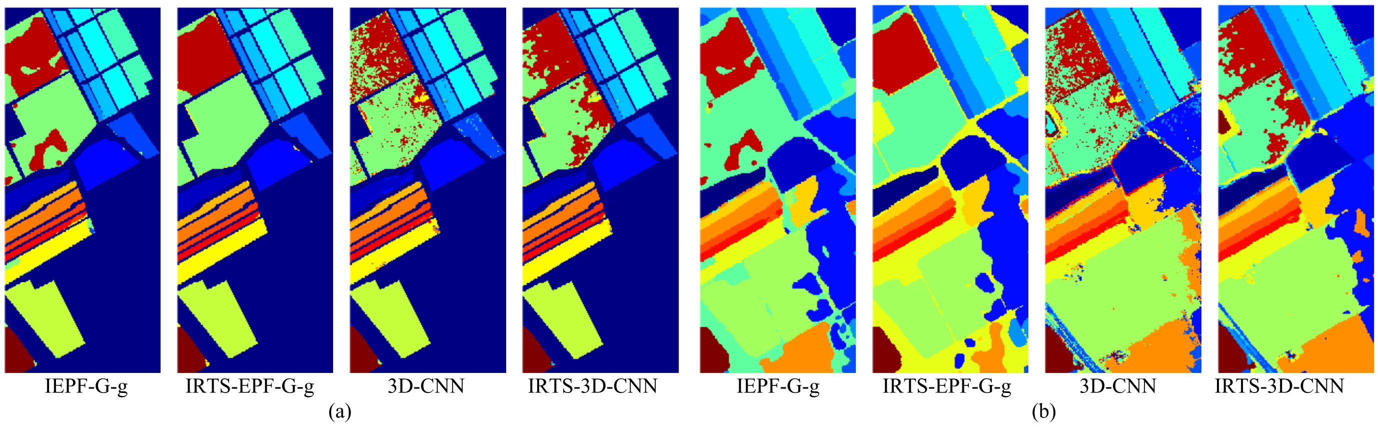


Fig. 11. Classification maps of IEPF-G-g, IRTS-EPF-G-g, 3D-CNN, and IRTS-3D-CNN for Salinas using 0.5% of training samples. (a) Without BKG. (b) With BKG.

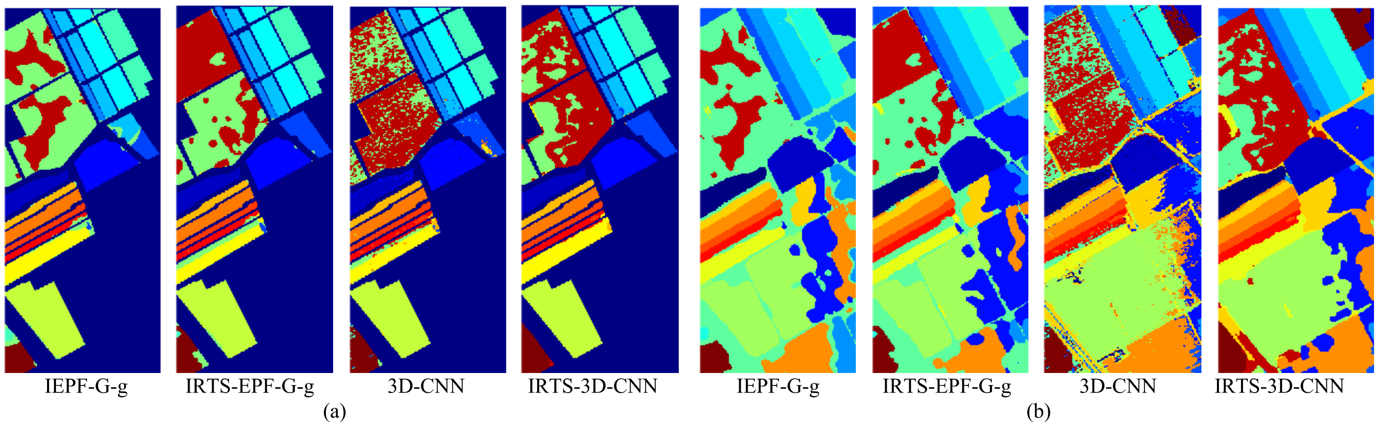


Fig. 12. Classification maps of IEPF-G-g, IRTS-EPF-G-g, 3D-CNN, and IRTS-3D-CNN for Salinas using 0.1% of training samples. (a) Without BKG. (b) With BKG.

even though IRTS was implemented. If we looked at particular classes 2 at the bottom of the scene and 6, IRTS-EPF-G-g and IRTS-3D-CNN performed the best and nearly the same when a 1% training sample size was used. However, when the sample size was reduced to 0.5% and 0.1%, IRTS-3D-CNN began to perform better than IRTS-EPF-G-g, specifically for the case of class 6 using a 0.1% training sample size.

Interestingly, the BKG issue of the U. of Pavia has been overlooked in HSIC in the past. Very little has been reported on how to deal with the BKG of U. of Pavia. As a matter

TABLE XV

$P_A$, $P_{\mathrm{OA}}$, $P_{\mathrm{AA}}$, $P_{\mathrm{PR}}$, $P_{\mathrm{OPR}}$, and $P_{\mathrm{APR}}$ Produced by IEPF-G-G, 3D-CNN, and IRTS-3D-CNN Using 1% Training Samples for U. Of Pavia

| Method | IEPF-G-g | | IRTS-EPF-G-g | | 3D-CNN | | IRTS-3D-CNN | |
|---|---|---|---|---|---|---|---|---|
| Class | $P_A(C_i)$ | $P_{PR}(C_i)$ | $P_A(C_i)$ | $P_{PR}(C_i)$ | $P_A(C_i)$ | $P_{PR}(C_i)$ | $P_A(C_i)$ | $P_{PR}(C_i)$ |
| 1 | 96.79(1.37) | 17.84(1.97) | **99.35**(0.48) | 18.23(1.58) | 87.86(6.52) | 20.09(2.47) | 98.26(0.66) | **21.98**(1.55) |
| 2 | 93.94(3.84) | 38.65(6.25) | **98.69**(0.77) | **39.74**(5.37) | 82.29(15.83) | 32.57(7.18) | 96.74(1.97) | 38.2(7.28) |
| 3 | 97.01(2.01) | **29.72**(6.11) | 99.66(0.53) | 27.27(4.24) | 90.7(4.32) | 17.7(3.96) | **99.92**(0.1) | 19.36(2.04) |
| 4 | 96.42(1.23) | 7.08(1.07) | 97.48(1.15) | 6.19(1.31) | 94.9(2.22) | **8.27**(0.84) | **98.41**(1.24) | 7.71(0.48) |
| 5 | **100**(0) | 38.44(12.19) | 99.97(0.05) | **45.53**(5) | 99.78(0.44) | 43.57(9.03) | **100**(0) | 43.68(6.78) |
| 6 | 98.22(1.58) | 13.12(2.9) | **99.27**(0.61) | **14.5**(2.05) | 78.44(19.95) | 9.17(1.37) | 98.1(1.47) | 13.15(2.81) |
| 7 | **100**(0) | 37.5(4.92) | **100**(0) | **41.94**(7.18) | 90.39(11.69) | 21.46(4.71) | 99.96(0.1) | 25.38(3.98) |
| 8 | 98.2(2.12) | 18.71(1.92) | 99.07(0.7) | **22.18**(2.11) | 82.68(17.66) | 18.58(5.94) | **99.15**(0.91) | 17.44(0.85) |
| 9 | **99.95**(0.07) | 11.03(0.77) | 99.74(0.29) | **11.94**(1.7) | 99.8(0.43) | 8.29(2.34) | 99.81(0.25) | 9.45(1.86) |
| $P_{OA}$ | 96.09(1.84) | | **98.96**(0.34) | | 85.24(6.55) | | 97.89(1.07) | |
| $P_{AA}$ | 97.84(0.72) | | **99.25**(0.17) | | 89.65(5.3) | | 98.92(0.49) | |
| $P_{OPR}$(BKG) | 19.82(0.38) | | **20.41**(0.07) | | 17.58(1.35) | | 20.19(0.22) | |
| $P_{APR}$(BKG) | 23.57(1.19) | | **25.28**(1.77) | | 19.97(1.93) | | 21.82(0.8) | |

TABLE XVI

$P_A$, $P_{\mathrm{OA}}$, $P_{\mathrm{AA}}$, $P_{\mathrm{PR}}$, $P_{\mathrm{OPR}}$, and $P_{\mathrm{APR}}$ Produced by IEPF-G-G, 3D-CNN, and IRTS-3D-CNN by Using 0.5% Training Samples for U. Of Pavia

| Method | IEPF-G-g | | IRTS-EPF-G-g | | 3D-CNN | | IRTS-3D-CNN | |
|---|---|---|---|---|---|---|---|---|
| Class | $P_A(C_i)$ | $P_{PR}(C_i)$ | $P_A(C_i)$ | $P_{PR}(C_i)$ | $P_A(C_i)$ | $P_{PR}(C_i)$ | $P_A(C_i)$ | $P_{PR}(C_i)$ |
| 1 | 94.76(4.65) | 18.88(2.88) | 97.34(1.36) | 18.2(0.67) | 81.94(10.84) | 20.11(2.5) | **98.66**(0.6) | **21.29**(2.56) |
| 2 | 86.59(5.14) | 32.93(5.85) | **95.86**(1.78) | 32.81(4.71) | 77.34(12.44) | 29.4(6.73) | 95.59(2.06) | **38.26**(8.13) |
| 3 | 94.28(3.67) | **28.63**(5.95) | 98.45(1.73) | 28.1(4.18) | 85.21(6.69) | 15.68(3.6) | **99.52**(0.57) | 19.52(3.03) |
| 4 | 96.33(2.44) | 6.29(1.83) | 95.65(2.43) | 7.36(1.92) | 93.3(3.02) | **8.75**(1.47) | **98.14**(1.21) | 7.84(0.66) |
| 5 | 99.91(0.1) | 38.06(12.74) | 99.93(0.08) | 42.22(8.73) | 99.7(0.54) | **46.39**(11.14) | **99.99**(0.02) | 41.77(8.1) |
| 6 | 92.85(4.32) | 14.73(4.73) | **98.6**(1.04) | **15.28**(3.99) | 83.08(6.98) | 9.39(1.78) | 98.11(2.11) | 13.11(2.64) |
| 7 | 99.66(1.07) | 37.37(9.26) | **99.99**(0.02) | **40.71**(7.18) | 90.7(9.38) | 18.72(4.49) | 98.71(2.78) | 27.96(2.82) |
| 8 | 94.69(3.07) | **21.78**(2.92) | **98.5**(1.24) | 21.65(2.5) | 81.98(21.62) | 17.68(5.41) | 98.06(1.35) | 18.82(4.36) |
| 9 | 98.55(1.5) | **14.23**(2.38) | 99.82(0.21) | 12(1.65) | 99.63(0.47) | 9.66(3.07) | **99.89**(0.09) | 8.66(1.21) |
| $P_{OA}$ | 91.45(2.62) | | 97.09(0.85) | | 82.27(6.45) | | **97.28**(1.14) | |
| $P_{AA}$ | 95.29(1.01) | | 98.24(0.33) | | 88.1(3.54) | | **98.52**(0.77) | |
| $P_{OPR}$(BKG) | 18.86(0.54) | | 20.03(0.18) | | 16.97(1.33) | | **20.06**(0.23) | |
| $P_{APR}$(BKG) | 23.66(1.67) | | **24.26**(1.9) | | 19.53(1.46) | | 21.91(0.96) | |

TABLE XVII

$P_A$, $P_{\mathrm{OA}}$, $P_{\mathrm{AA}}$, $P_{\mathrm{PR}}$, $P_{\mathrm{OPR}}$, and $P_{\mathrm{APR}}$ Produced by IEPF-G-G, 3D-CNN, and IRTS-3D-CNN by Using 0.1% Training Samples for U. Of Pavia

| Method | IEPF-G-g | | IRTS-EPF-G-g | | 3D-CNN | | IRTS-3D-CNN | |
|---|---|---|---|---|---|---|---|---|
| Class | $P_A(C_i)$ | $P_{PR}(C_i)$ | $P_A(C_i)$ | $P_{PR}(C_i)$ | $P_A(C_i)$ | $P_{PR}(C_i)$ | $P_A(C_i)$ | $P_{PR}(C_i)$ |
| 1 | 78.69(14.09) | 19.73(3.96) | 86.11(8.28) | 20.91(2.06) | 56.02(16.83) | 17.44(3.1) | **86.77**(12.78) | **23.44**(1.89) |
| 2 | 57.74(13.92) | 26.87(4.87) | 76.02(13.18) | 19.16(4.89) | 76.45(10.99) | 26.35(5.5) | **84.7**(7.64) | **32.75**(5.97) |
| 3 | 77.53(14.57) | 14.02(8.78) | 82.45(5.5) | **16.26**(6.09) | 64.12(9.92) | 9.15(3.33) | **93.67**(5.42) | 15.57(6.14) |
| 4 | 87.78(10.75) | 7.84(2.45) | 89.59(6.95) | **9.91**(2.89) | 86.24(8.76) | 8.6(1.96) | **96.5**(2.67) | 8.87(0.78) |
| 5 | 95.75(12.95) | 33.94(18.52) | 93.17(14.43) | **46.19**(16.85) | 87.52(12.53) | 30.81(12.98) | **99.03**(1.89) | 34.16(8.6) |
| 6 | 62.48(16.67) | 7.29(2.32) | 61.8(17.89) | **11.65**(3.91) | 42.66(17.05) | 6.31(1.11) | **71.17**(15.37) | 10.04(3.14) |
| 7 | 88.76(15.27) | 25(16.07) | 95.41(11.77) | **30.25**(7.14) | 81.5(9) | 10.76(3.84) | **97.43**(1.92) | 24.02(8.68) |
| 8 | 69.58(22.23) | 18.12(7.7) | 91.29(7.86) | **23.43**(5.07) | 67.19(15.37) | 15.31(5.07) | **95.38**(2.49) | 15.41(3.46) |
| 9 | 99.49(0.79) | **12.12**(1.7) | 99.44(1.34) | 11.58(3) | 97.04(2.91) | 7.88(2.32) | **99.84**(0.17) | 8.17(1.72) |
| $P_{OA}$ | 68.77(7.96) | | 80.18(5.12) | | 69.57(3.58) | | **86.82**(3.24) | |
| $P_{AA}$ | 79.76(6.48) | | 86.14(4.31) | | 73.19(3.91) | | **91.61**(2.57) | |
| $P_{OPR}$(BKG) | 14.18(1.64) | | 16.54(1.06) | | 14.35(0.74) | | **17.91**(0.67) | |
| $P_{APR}$(BKG) | 18.33(2.39) | | **21.04**(2.7) | | 14.73(1.5) | | 19.16(1.3) | |

of fact, when we removed its BKG, U. of Pavia could almost achieve as high as 97% of OA and AA of 98% when the training sample size was greater than 0.5%. Also, for this particular dataset, IRTS-3D-CNN nearly outperformed all other methods.

### *D. Discussions*

All the experiments conducted in this article demonstrated four important facts. One is that IRTS is always able to improve any classifier without using IRTS as shown by experiments where IRTS-IEPF-G-g and IRTS-3D-CNN always improved their corresponding counterparts without using IRTS. Second, when the training sample pool is sufficiently enough, such as 10%, 5% of the Purdue data, 1%, 0.5% of Salinas, and 1% of Pavia, IRTS-IEPF-G-g generally performed better than IRTS-3D-CNN. In this case, SSC can generally do better than deep learning-based classification. However, when the training sample pool is reduced to 1% of the Purdue data, 0.1% of Salinas, and 0.5%, 0.1% of Pavia, IRTS-3D-CNN begins to show its compatibility and superiority over IRTS-IEPF-G-g due to its deep learning capability. This is because IEPF-G-g is a statistical classifier, which requires

TABLE XVIII
AC for IEPF, IRTS-EPF, 3D-CNN, and IRTS-3D-CNN in Terms of 3D ROC Analysis, $P_{AA}$, $P_{OA}$, and $P_{OPR}$ for U. Of Pavia

| | SR% | IEPF-G-g | IRTS-EPF-G-g | 3D-CNN | IRTS-3D-CNN |
|---|---|---|---|---|---|
| $AAUC_{(D,F)}$ | 1 | 0.9716(0.0028) | 0.9751(0.002) | 0.9491(0.0064) | **0.9778**(0.0036) |
| | 0.5 | 0.9606(0.0059) | 0.9732(0.0019) | 0.9406(0.0069) | **0.9772**(0.0022) |
| | 0.1 | 0.8996(0.0266) | 0.9378(0.0137) | 0.9004(0.0125) | **0.9571**(0.0092) |
| $AAUC_{(D,T)}$ | 1 | 0.8014(0.008) | 0.8096(0.0084) | 0.7202(0.0238) | **0.8576**(0.0242) |
| | 0.5 | 0.788(0.0084) | 0.8016(0.0101) | 0.6838(0.0399) | **0.8458**(0.0309) |
| | 0.1 | 0.7073(0.0383) | 0.7402(0.0275) | 0.5922(0.0399) | **0.7667**(0.0479) |
| $AAUC_{(F,T)}$ | 1 | 0.2965(0.0132) | 0.2749(0.0237) | 0.4306(0.0268) | **0.2544**(0.0348) |
| | 0.5 | 0.2814(0.0296) | 0.2843(0.0236) | 0.4325(0.0181) | **0.2693**(0.0422) |
| | 0.1 | 0.3098(0.0162) | 0.3013(0.0208) | 0.418(0.0338) | **0.285**(0.0593) |
| $P_{AA}$ | 1 | 0.9784(0.0072) | **0.9925**(0.0017) | 0.8965(0.053) | 0.9893(0.0049) |
| | 0.5 | 0.9529(0.0101) | 0.9824(0.0033) | 0.881(0.0354) | **0.9852**(0.0077) |
| | 0.1 | 0.7976(0.0648) | 0.8614(0.0431) | 0.7319(0.0391) | **0.9161**(0.0257) |
| $P_{APR}$ | 1 | 0.2357(0.0119) | **0.2528**(0.0177) | 0.1997(0.0193) | 0.2182(0.008) |
| | 0.5 | 0.2366(0.0167) | **0.2426**(0.019) | 0.1953(0.0146) | 0.2191(0.0096) |
| | 0.1 | 0.1833(0.0239) | **0.2104**(0.027) | 0.1474(0.015) | 0.1916(0.013) |
| AC | 1 | 2.6906 | 2.7551 | 2.3349 | **2.7885** |
| | 0.5 | 2.6567 | 2.7155 | 2.2682 | **2.758** |
| | 0.1 | 2.2780 | 2.4485 | 1.9539 | **2.5465** |

TABLE XIX
CWC Calculated by (20) for IEPF, IRTS-EPF, 3D-CNN, and IRTS-3D-CNN in Terms of 3D ROC Analysis, $P_{AA}$, $P_{OA}$, and $P_{OPR}$ for U. Of Pavia

| | SR% | IEPF-G-g | IRTS-EPF-G-g | 3D-CNN | IRTS-3D-CNN |
|---|---|---|---|---|---|
| $CWAUC_{(D,F)}$ | 1 | 0.9591(0.0061) | **0.9666**(0.0035) | 0.9114(0.0179) | 0.9621(0.0095) |
| | 0.5 | 0.9297(0.0164) | 0.9583(0.0058) | 0.8931(0.0237) | **0.9607**(0.0051) |
| | 0.1 | 0.8345(0.0362) | 0.8755(0.0409) | 0.8596(0.0226) | **0.9271**(0.0158) |
| $CWAUC_{(D,T)}$ | 1 | 0.7965(0.0153) | 0.8099(0.0117) | 0.6776(0.0416) | **0.8265**(0.044) |
| | 0.5 | 0.773(0.0152) | 0.7914(0.0369) | 0.6255(0.0739) | **0.8255**(0.025) |
| | 0.1 | 0.6622(0.0577) | 0.6729(0.0496) | 0.5966(0.0721) | **0.7326**(0.0616) |
| $CWAUC_{(F,T)}$ | 1 | 0.3276(0.0263) | 0.3099(0.0245) | 0.4552(0.0365) | **0.2861**(0.0464) |
| | 0.5 | 0.3347(0.028) | 0.3273(0.0315) | 0.4363(0.0402) | **0.3029**(0.045) |
| | 0.1 | 0.3543(0.0341) | 0.3363(0.0271) | 0.4498(0.0683) | **0.3246**(0.0756) |
| $P_{CWA}$ | 1 | 0.9609(0.0184) | **0.9896**(0.0034) | 0.8524(0.0655) | 0.9789(0.0107) |
| | 0.5 | 0.9145(0.0262) | 0.9709(0.0085) | 0.8227(0.0645) | **0.9728**(0.0114) |
| | 0.1 | 0.6877(0.0796) | 0.8018(0.0512) | 0.6957(0.0358) | **0.8682**(0.0324) |
| $P_{CWPR}$ | 1 | 0.1982(0.0038) | **0.2041**(0.0007) | 0.1758(0.0135) | 0.2019(0.0022) |
| | 0.5 | 0.1886(0.0054) | 0.2003(0.0018) | 0.1697(0.0133) | **0.2006**(0.0024) |
| | 0.1 | 0.1418(0.0164) | 0.1654(0.0106) | 0.1435(0.0074) | **0.1791**(0.0067) |
| CWC | 1 | 2.5871 | 2.6603 | 2.1620 | **2.6833** |
| | 0.5 | 2.4711 | 2.5936 | 2.0747 | **2.6567** |
| | 0.1 | 1.9719 | 2.1793 | 1.8456 | **2.3824** |

a sufficient number of training samples to make up reliable statistics, while a deep learning-based classifier takes its learning capability in depth to learn unlabeled data samples from a very limited number of training samples. The third one is that, when the BKG is complicated such as U. of Pavia, IRTS-3D-CNN also proves its superior adaptability to dealing with BKG. Finally, it should be noted that Figs. 7–15 showed that there was salt and pepper noise present in only the classification maps produced by 3D-CNN with/without BKG. This was resulting from the use of small patch sizes and limited training samples used. As a result, 3D-CNN may not be able to capture all the relevant features of the images. Interestingly, as also shown in Figs. 7–15, IRTS-3D-CNN alleviated and removed such an effect. This phenomenon could be identified by the quantitative results of Tables IV–XIX. The visual inspection demonstrated that the salt and pepper issue can be effectively addressed by coupling the IRTS-feedback loop with the 3D-CNN model, even with small patch sizes and limited training samples. All these findings make sense; specifically, IRTS-3D-CNN has shown its great potential when a very limited number of data samples are only available and can be used for training.

## E. Time Complexity Analysis

Python and TensorFlow libraries were used to run experiments. All experimental results are generated on a desktop with an Intel Core i7-9700, 16-GB memory, and an NVIDIA GeForce RTX 2070. Table XX shows the averaged running time along with the averaged iteration and the threshold of TI for all tested algorithms tabulated in Table XXI.

According to Table XX, IRTS-3D-CNN required longer running time than other algorithms due to the fact that the training process used RTS was relatively complicated compared to EPF without using RTS. As also noted, the time required by IEPF and IRTS-EPF increased significantly as the training sample size was increased. However, one interesting finding was that the computing time of IRTS-3D-CNN was not necessarily proportional to the size of training size. It was

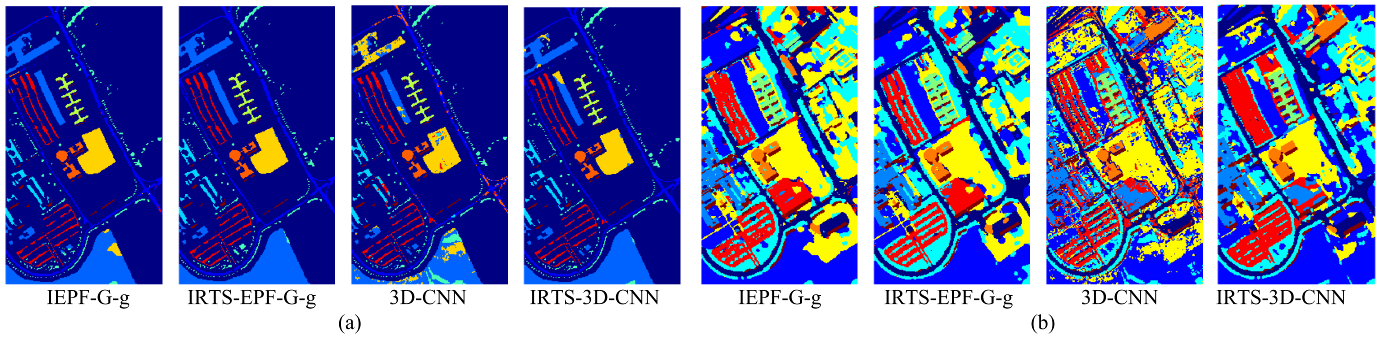


Fig. 13. Classification maps of IEPF-G-g, IRTS-EPF-G-g, 3D-CNN, and IRTS-3D-CNN for U. of Pavia using 1% of training samples. (a) Without BKG. (b) With BKG.

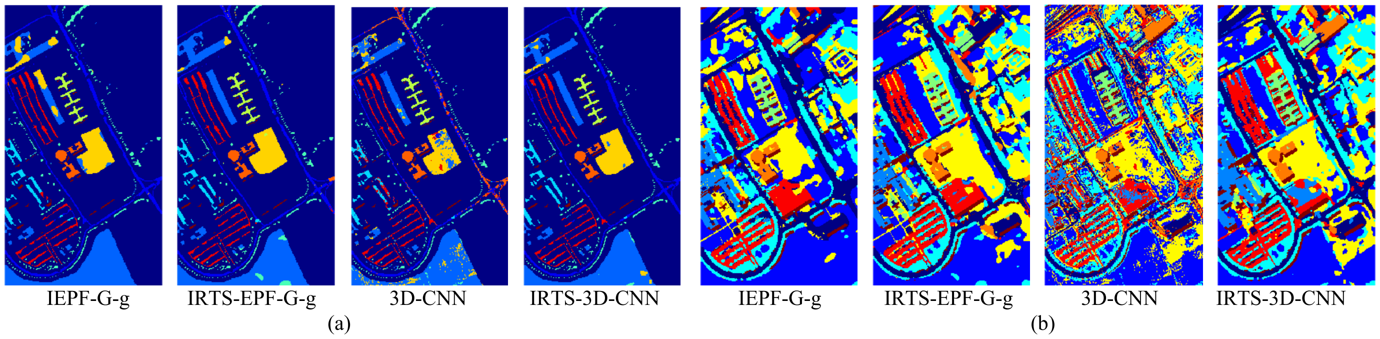


Fig. 14. Classification maps of IEPF-G-g, IRTS-EPF-G-g, 3D-CNN, and IRTS-3D-CNN for U. of Pavia using 0.5% of training samples. (a) Without BKG. (b) With BKG.

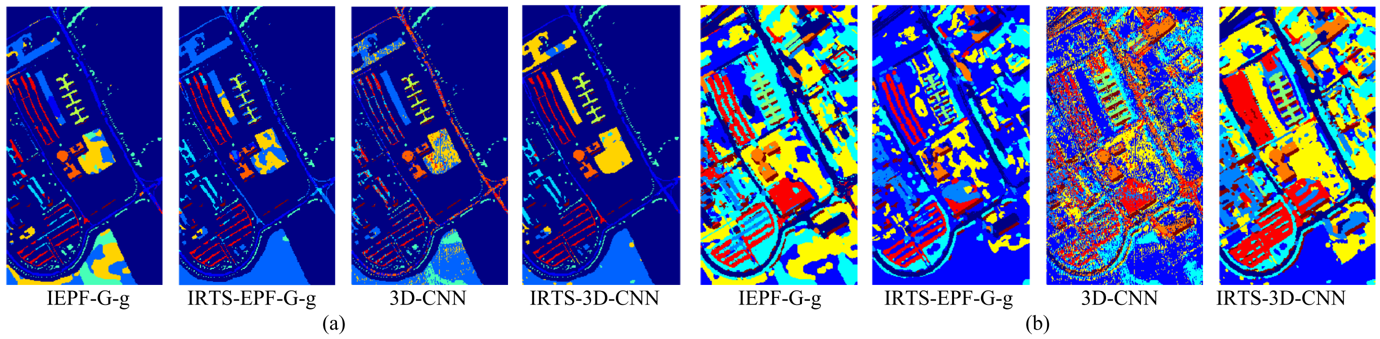


Fig. 15. Classification maps of IEPF-G-g, IRTS-EPF-G-g, 3D-CNN, and IRTS-3D-CNN for U. of Pavia using 0.1% of training samples. (a) Without BKG. (b) With BKG.

actually determined by the number of iterations. For example, the averaged running time required by IRTS-3D-CNN using SR = 10%, 5%, and 1% were 2484.7, 1591.7, and 2448.4 s, respectively. For different SRs, the number of iterations inside the CNN training process was always fixed. Therefore, the running time of IRTS-3D-CNN was mainly caused by RTS. That is, the averaged iteration numbers for IRTS-3D-CNN using SR = 10%, 5%, and 1% were 8.8, 6.5, and 8.8, respectively. As a result, the averaged running time for SR = 5% required less time than that required for SR = 10% and 1%.

## VIII. Comparison With State-of-the Art 3D-CNNs

In Section VII, IRTS-CNN was compared with the EPF-derived IEPF-G-g and showed its advantages and generalization ability to machine learning-based HSIC techniques. In this section, we further demonstrate that IRTS-CNN also shows its better performance and generalization ability to deep learning-based HSIC techniques over the same used CNN without using IRTS.

The three most recently developed state-of-the-art 3D-CNNs described in Section VI-A2 were selected to conduct a comparative analysis in the same manner that it was performed in Section VII since the experiments in Section VII already showed that CNN had advantages when small numbers of training samples were used. In this case, 1% of data samples were randomly selected from all three datasets, Purdue, Salinas, and the University of Pavia, and training samples were selected according to Kang's method for experiments. As will be demonstrated in this section, each of them did show its own merit in HSIC.

### *A. Purdue Data*

Table XXII tabulates the results of $P_A$, $P_{\text{OA}}$, $P_{\text{AA}}$, $P_{\text{PR}}$, $P_{\text{OPR}}$, and $P_{\text{APR}}$ produced by the four test methods, HybridSN,

TABLE XX

AVERAGED RUNNING TIMES IN SECONDS REQUIRED BY IEPF, IRTS-EPF, AND IRTS-3D-CNN FOR PURDUE DATA, SALINAS, AND THE UNIVERSITY OF PAVIA

| | SR | IEPF | IRTS-EPF | 3D-CNN | IRTS-3D-CNN |
|---|---|---|---|---|---|
| Purdue | 10 | 46.4 | 46.3 | 191.1 | 2484.7 |
| | 5 | 22.4 | 29.1 | 189.2 | 1591.7 |
| | 1 | 12.9 | 11.8 | 193.4 | 2448.4 |
| Salinas | 1 | 78.6 | 69.2 | 321.3 | 2853.8 |
| | 0.5 | 50.6 | 78.5 | 340.4 | 3107.1 |
| | 0.1 | 40.8 | 21.7 | 321.9 | 5531.7 |
| U. Pavia | 1 | 43.3 | 65.4 | 310.0 | 3363.9 |
| | 0.5 | 74.6 | 55.8 | 315.6 | 4775.8 |
| | 0.1 | 24.0 | 26.9 | 314.7 | 4234.9 |

TABLE XXI

AVERAGED ITERATION AND THE THRESHOLD OF TI REQUIRED BY IEPF, IRTS-EPF, AND IRTS-3D-CNN FOR PURDUE DATA, SALINAS, AND THE UNIVERSITY OF PAVIA

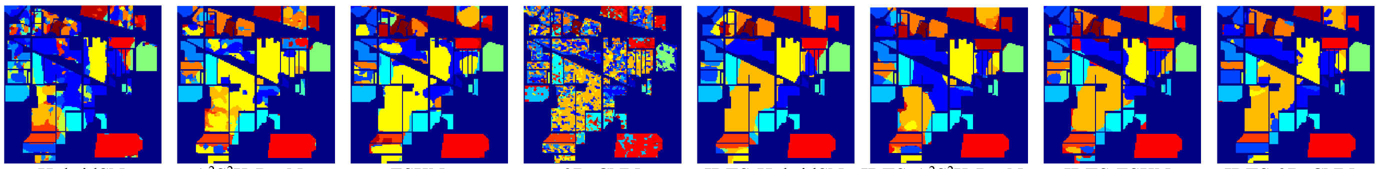
| | SR | IEPF | IRTS-EPF | IRTS-3D-CNN |
|---|---|---|---|---|
| | | Iteration/TI | Iteration/TI | Iteration/TI |
| Purdue | 10 | 12.5/0.99 | 11.6/0.92 | 8.8/0.99 |
| | 5 | 8.7/0.98 | 10.2/0.88 | 6.5/0.98 |
| | 1 | 12.4/0.97 | 9.8/0.69 | 8.8/0.81 |
| Salinas | 1 | 10.2/0.99 | 8.5/0.92 | 3.8/0.99 |
| | 0.5 | 7.2/0.98 | 11.3/0.92 | 4/0.98 |
| | 0.1 | 10/0.98 | 6.5/0.78 | 7/0.98 |
| U. Pavia | 1 | 8/0.97 | 10.9/0.88 | 6.6/0.99 |
| | 0.5 | 15/0.98 | 10.5/0.85 | 8.7/0.99 |
| | 0.1 | 7.7/0.95 | 8.1/0.66 | 7.7/0.83 |

TABLE XXII

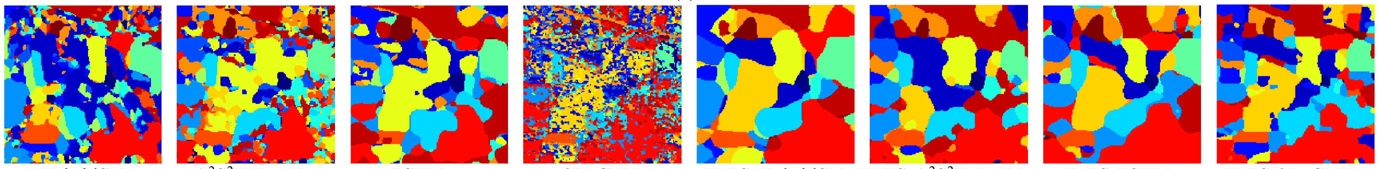
$P_A$, $P_{\mathrm{OA}}$, $P_{\mathrm{AA}}$, $P_{\mathrm{PR}}$, $P_{\mathrm{OPR}}$, AND $P_{\mathrm{APR}}$ PRODUCED BY HYBRIDSN, A$^2$S$^2$K-RESNET, FSKNET, 3D-CNN, IRTS-HYBRIDSN, IRTS-A$^2$S$^2$K-RESNET, IRTS-FSKNET, AND IRTS-3D-CNN BY USING 1% TRAINING SAMPLES FOR PURDUE DATA

| Method | **STATE-OF-THE ART 3D CNNS** | | | | **THEIR ITRS COUNTERPARTS** | | | |
|---|---|---|---|---|---|---|---|---|
| | HybridSN | A$^2$S$^2$K-ResNet | FSKNet | 3D-CNN | IRTS-HybridSN | IRTS-A$^2$S$^2$K-ResNet | IRTS-FSKNet | IRTS-3D-CNN |
| Class | $P_A(C_l)$ | $P_A(C_l)$ | $P_A(C_l)$ | $P_A(C_l)$ | $P_A(C_l)$ | $P_A(C_l)$ | $P_A(C_l)$ | $P_A(C_l)$ |
| 1 | 97.83(2.18) | **100**(0) | 97.83(3.76) | 84.78(9.5) | **100**(0) | **100**(0) | **100**(0) | 99.13(1.52) |
| 2 | 39.47(26.94) | **52.03**(15.65) | 34.32(17.62) | 29.44(10.14) | 65.63(5.16) | **84.74**(7.28) | 78.69(4.3) | 78.6(10.7) |
| 3 | 50.08(2.86) | 36.3(3.66) | **62.97**(9.69) | 29.14(11.55) | 69.25(9.66) | 47.35(4) | 71.28(16.22) | **91.64**(4.72) |
| 4 | 58.93(4.33) | **94.8**(2.58) | 89.03(11.76) | 42.66(9.63) | 95.36(5.6) | 97.47(0) | 97.75(3.54) | **97.81**(2.96) |
| 5 | 79.78(5.87) | 69.63(1.02) | **90.2**(3.44) | 61.76(12.58) | 87.73(4.55) | 69.15(0) | **93.51**(2.47) | 92.34(3.83) |
| 6 | 89.55(9.86) | **96.12**(2.42) | 90.46(6.65) | 57.27(10.68) | 96.03(2.83) | 97.72(0.75) | 94.52(2.73) | **98.67**(1.86) |
| 7 | **100**(0) | **100**(0) | **100**(0) | 87.5(10.55) | **100**(0) | **100**(0) | 75(26.96) | **100**(0) |
| 8 | 95.75(3.93) | 95.47(6.27) | **98.26**(2.49) | 63.72(12.31) | **100**(0) | 99.37(0.91) | 99.72(0.48) | 99.64(0.85) |
| 9 | **100**(0) | **100**(0) | **100**(0) | 96(4.59) | **100**(0) | 98.33(2.89) | **100**(0) | **100**(0) |
| 10 | 60.97(3.65) | **68.25**(26.75) | 65.74(11.52) | 32.34(12.28) | **88.01**(6.4) | 83.67(1.85) | 74.55(9.79) | 87.7(5.08) |
| 11 | 39.24(16.82) | **54.24**(33.65) | 39.93(24.08) | 40.79(10.1) | 79.12(4.87) | 52.62(15.2) | **83.83**(3.11) | 77.84(6.78) |
| 12 | 43.28(7.44) | **82.91**(7.11) | 59.25(13.61) | 28.62(12.24) | 70.83(20.96) | **90.95**(1.02) | 62.23(20.87) | 90.35(6.57) |
| 13 | 97.72(2.2) | **100**(0) | 86.99(18.02) | 86.59(9.02) | 94.76(4.75) | **100**(0) | 99.02(0.98) | **100**(0) |
| 14 | 87.56(2.1) | 76.5(2.61) | **91.46**(6.65) | 61.94(8.71) | 94.88(2.94) | 71.46(0) | 92.02(4.86) | **97.91**(3.27) |
| 15 | 82.81(13.37) | **91.19**(9.89) | 82.81(15.04) | 48.08(14.98) | 95.21(7.29) | **99.91**(0.15) | 97.67(2.99) | 98.08(5.62) |
| 16 | 88.17(6.71) | **100**(0) | 95.7(7.45) | 87.1(10.78) | 97.85(3.04) | 98.21(0.62) | 98.57(2.48) | **99.79**(0.68) |
| $P_{\mathrm{OA}}$ | 60.79(1.3) | **67.92**(6.18) | 64.37(4.56) | 44.69(4.99) | 82.99(1.64) | 74.94(4.42) | 84.3(2.04) | **88.45**(2.86) |
| $P_{\mathrm{AA}}$ | 75.7(1.15) | **82.34**(2.36) | 80.31(0.93) | 58.61(6.41) | 89.67(1.5) | 86.93(1.44) | 88.65(2.2) | **94.34**(1.38) |
| $P_{\mathrm{OPR}}$(BKG) | 29.64(0.63) | **33.11**(3.01) | 31.38(2.23) | 21.79(2.43) | 40.46(0.8) | 36.53(2.15) | 41.09(1) | **43.12**(1.4) |
| $P_{\mathrm{APR}}$(BKG) | 28.7(1.37) | **34.31**(2.1) | 34.12(1.51) | 21.59(3.27) | 37.05(1.22) | 38.69(2.11) | 41.31(1.56) | **45.87**(1.42) |
| Time | 341.07(16.06) | 263.17(27.78) | 518.76(5.33) | **193.4**(5.39) | 9615.3 (11278.36) | **2446.27** (1419.51) | 5794.88 (676.89) | 2448.4 (1938.35) |

A$^2$S$^2$K-ResNet, FSKNet, and 3D-CNN, and their corresponding counterparts, IRTS-HybridSN, IRTS-A$^2$S$^2$K-ResNet, IRTS-FSKNet, and IRTS-3D-CNN for comparison, respectively, where the best results are boldfaced. It is obvious that all the IRTS versions of the four test methods significantly improved their counterparts without using RTS ranging from the smallest improvement of A$^2$S$^2$K-ResNet with 6%–7% of OA and AA to the greatest improvement of 3D-CNN with 44% of OA and 40% of AA. Most interestingly, 3D-CNN was the worst but became the best after it was implemented as IRTS-3D-CNN with $P_{\mathrm{OA}} = 88.45$, $P_{\mathrm{AA}} = 94.34$, $P_{\mathrm{OPR}}$ (BKG) $=$ 43.12, and $P_{\mathrm{APR}}$ (BKG) $= 45.87$. When BKG was factored in the classification to calculate $P_{\mathrm{OPR}}$ (BKG) and $P_{\mathrm{APR}}$ (BKG), A$^2$S$^2$K-ResNet was the best among all the four test methods


HybridSN $A^2S^2$K-ResNet FSKNet 3D-CNN IRTS-HybridSN IRTS-$A^2S^2$K-ResNet IRTS-FSKNet IRTS-3D-CNN

(a)


HybridSN $A^2S^2$K-ResNet FSKNet 3D-CNN IRTS-HybridSN IRTS-$A^2S^2$K-ResNet IRTS-FSKNet IRTS-3D-CNN

(b)

Fig. 16. Classification maps of HybridSN, $A^2S^2$K-ResNet, FSKNet, 3D-CNN, and their IRTS counterparts for the Purdue data using 1% of training samples. (a) Without BKG. (b) With BKG.

but turned out to be the worst after IRTS is implemented. This is completely opposite to 3D-CNN. Nevertheless, $P_{\text{OPR}}$ (BKG) and $P_{\text{APR}}$ (BKG) for all the test methods were very low ranging from 21.8% of $P_{\text{OPR}}$ (BKG) and 21.6% of $P_{\text{APR}}$ (BKG) produced by 3D-CNN to 33.3% of $P_{\text{OPR}}$ (BKG) and 34.3% of $P_{\text{APR}}$ (BKG) produced by $A^2S^2$K-ResNet. Although IRTS did increase their PR results, the improvements were very limited due to unknown BKG. An intriguing finding was that IRTS did improve the $A^2S^2$K-ResNet performance in classification rates, but the improvement was small compared to the other three methods. These experiments demonstrated that IRTS did improve 3D-CNN greatly and significantly but did not help $A^2S^2$K-ResNet as much as expected.

In order to visually inspect the classification results of the Purdue data produced by the four test methods, HybridSN, $A^2S^2$K-ResNet, FSKNet, and 3D-CNN along with their corresponding counterparts, IRTS-HybridSN, IRTS-$A^2S^2$K-ResNet, IRTS-FSKNet, and IRTS-3D-CNN, Fig. 16 shows their classification maps for a comparative study where IRTS versions performed significantly better than their counterparts without using IRTS. In particular, the salt and pepper noise had a severe effect on the classification maps produced by 3D-CNN. IRTS-3D-CNN indeed resolved this issue and removed all the salt and pepper noise. In addition, classification results including BKG were much worse than that without BKG.

## B. Salinas

Table XXIII tabulates the results of $P_A$, $P_{\text{OA}}$, $P_{\text{AA}}$, $P_{\text{PR}}$, $P_{\text{OPR}}$, and $P_{\text{APR}}$ produced by the four test methods, HybridSN, $A^2S^2$K-ResNet, FSKNet, and 3D-CNN, and their corresponding counterparts, IRTS-HybridSN, IRTS-$A^2S^2$K-ResNet, IRTS-FSKNet, and IRTS-3D-CNN for comparison, respectively, where the best results are boldfaced and were produced by HybridSN among all the four test methods with $P_{\text{OA}} = 97.33$ and $P_{\text{AA}} = 98.65$, which were improved to $P_{\text{OA}} = 99.01$ and $P_{\text{AA}} = 99.47$ after IRTS was implemented. On the other hand, the worst results were produced by FSKNet $P_{\text{OA}} = 84.66$ and $P_{\text{AA}} = 93.71$, which were improved to $P_{\text{OA}} = 90.87$ and $P_{\text{AA}} = 95.89$ after IRTS was implemented. As for PR, the rates were greatly improved compared to that of the Purdue data but were still low. IRTS did improve a little bit but not much.

In analogy with Fig. 16, Fig. 17 also shows the classification results of Salinas produced by the four test methods, HybridSN, $A^2S^2$K-ResNet, FSKNet, and 3D-CNN along with their corresponding counterparts, IRTS-HybridSN, IRTS-$A^2S^2$K-ResNet, IRTS-FSKNet, and IRTS-3D-CNN for visual inspection where, once again, IRTS versions performed significantly better than their counterparts without using IRTS. Also, the effect of the salt and pepper noise present in the classification maps produced by 3D-CNN was removed by IRTS-3D-CNN. Similarly, the classification results including BKG were much worse than that without BKG. In particular, when BKG is included for classification, class 9 was completely lost.

## C. U. Of Pavia

Table XXIV tabulates the results of $P_A$, $P_{\text{OA}}$, $P_{\text{AA}}$, $P_{\text{PR}}$, $P_{\text{OPR}}$, and $P_{\text{APR}}$ produced by the four test methods, HybridSN, $A^2S^2$K-ResNet, FSKNet, and 3D-CNN, and their corresponding counterparts, IRTS-HybridSN, IRTS-$A^2S^2$K-ResNet, IRTS-FSKNet, and IRTS-3D-CNN for comparison, respectively, where the best results are boldfaced and were produced by 3D-CNNs and $A^2S^2$K-ResNet with $P_{\text{OA}} = 95.62$ and $P_{\text{AA}} = 95.84$ among the four test methods. Interestingly, after IRTS was implemented, the best one turned out to be IRTS-FSKNet with $P_{\text{OA}} = 99.43$ and $P_{\text{AA}} = 99.34$. Due to the very complicated BKG, the PR rates for all the test methods were extremely low ranging from 19.25% to 19.72%. Even though IRTS was implemented, their rate did not go up much from 20% to 22%.

Similar to Figs. 16 and 17, Fig. 18 shows the classification results of Salinas produced by the four test methods, HybridSN, $A^2S^2$K-ResNet, FSKNet, and 3D-CNN along with their corresponding counterparts, IRTS-HybridSN, IRTS-$A^2S^2$K-ResNet, IRTS-FSKNet, and IRTS-3D-CNN for visual inspection where IRTS versions seemed to perform similarly compared to their counterparts without using IRTS except the classification map produced by 3D-CNN whose class 6 and class 2 were corrected by IRTS-by 3D-CNN. Nevertheless,

TABLE XXIII

$P_A$, $P_{OA}$, $P_{AA}$, $P_{PR}$, $P_{OPR}$, AND $P_{APR}$ PRODUCED HYBRIDSN, $A^2S^2$K-RESNET, FSKNET, 3D-CNN, IRTS-HYBRIDSN, IRTS-$A^2S^2$K-RESNET, IRTS-FSKNET, AND IRTS-3D-CNN BY USING 1% TRAINING SAMPLES FOR SALINAS

| Method | STATE-OF-THE ART 3D CNNS | | | | THEIR ITRS COUNTERPARTS | | | |
|---|---|---|---|---|---|---|---|---|
| | HybridSN | $A^2S^2$K-ResNet | FSKNet | 3D-CNN | IRTS-HybridSN | IRTS-$A^2S^2$K-ResNet | IRTS-FSKNet | IRTS-3D-CNN |
| Class | $P_A(C_i)$ | $P_A(C_i)$ | $P_A(C_i)$ | $P_A(C_i)$ | $P_A(C_i)$ | $P_A(C_i)$ | $P_A(C_i)$ | $P_A(C_i)$ |
| 1 | 99.95(0.09) | **100**(0) | 97.26(3.18) | 98.74(1.72) | **100**(0) | **100**(0) | 99.38(1.05) | 99.95(0.16) |
| 2 | **100**(0) | 99.87(0.19) | **100**(0) | 99.5(0.79) | **100**(0) | 99.72(0.48) | 99.96(0.06) | **100**(0) |
| 3 | **100**(0) | **100**(0) | 99.97(0.06) | 93.83(5.12) | **100**(0) | **100**(0) | **100**(0) | 99.92(0.26) |
| 4 | 99.45(0.48) | **99.98**(0.04) | 99.19(1.22) | 98.61(1.41) | **100**(0) | 99.98(0.04) | 97.88(3.19) | 99.81(0.44) |
| 5 | **99.44**(0.51) | 98.71(0.48) | 98.01(1.1) | 97.28(1.84) | 99.08(0.7) | 98.29(1.86) | 96.95(2.25) | **99.2**(0.87) |
| 6 | 99.85(0.2) | **99.99**(0.02) | 99.6(0.35) | 99.52(0.47) | 99.99(0.02) | **100**(0) | 95.89(5.29) | 99.83(0.27) |
| 7 | **100**(0) | **100**(0) | 97.72(2.32) | 99.37(0.56) | 99.96(0.06) | **100**(0) | 99.66(0.47) | 99.84(0.47) |
| 8 | **92.67**(3.15) | 84.88(0.75) | 58.24(17.81) | 64.03(19.02) | **97.58**(1.97) | 91.79(1.23) | 71.26(27.03) | 88.36(7.92) |
| 9 | 99.89(0.2) | **99.95**(0.09) | 98.59(0.5) | 99.18(0.53) | **100**(0) | 99.99(0.02) | 99.56(0.62) | 99.9(0.21) |
| 10 | **98.88**(0.84) | 96.41(0.53) | 97.25(0.96) | 91.91(3.24) | **99.51**(0.54) | 96.65(0.3) | 96.36(4.27) | 97.99(1.42) |
| 11 | **99.94**(0.05) | 99.66(0.59) | 99.84(0.14) | 97.15(1.63) | 98.19(3.14) | **100**(0) | 94.34(10.23) | 99.62(0.4) |
| 12 | 99.97(0.06) | **99.98**(0.03) | 99.97(0.03) | 99.96(0.1) | **100**(0) | **100**(0) | 99.76(0.46) | 99.99(0.03) |
| 13 | 99.75(0.44) | **100**(0) | 99.89(0.11) | 98.76(1.15) | 99.93(0.06) | **100**(0) | 99.81(0.24) | 99.37(1) |
| 14 | 96.33(6.28) | **100**(0) | 99.07(1.31) | 95.52(1.92) | **100**(0) | **100**(0) | 99.88(0.23) | 99.54(0.62) |
| 15 | **93.39**(3.24) | 87.22(0.86) | 56.57(13.81) | 73.43(22.05) | **97.27**(2.19) | 91.3(2.22) | 83.71(10.08) | 94.32(2.65) |
| 16 | **98.87**(1.04) | 97.49(0.31) | 98.23(3.02) | 97.89(0.53) | **100**(0) | 97.99(0.61) | 99.86(0.14) | 99.65(0.39) |
| $P_{OA}$ | **97.33**(0.69) | 94.75(0.12) | 84.66(1.88) | 87.57(2.72) | **99.01**(0.4) | 96.75(0.46) | 90.87(6.71) | 96.57(1.77) |
| $P_{AA}$ | **98.65**(0.63) | 97.76(0.05) | 93.71(0.54) | 94.05(1.39) | **99.47**(0.15) | 98.48(0.27) | 95.89(2.23) | 98.58(0.57) |
| $P_{OPR}$(BKG) | **47.42**(0.33) | 46.16(0.06) | 41.24(0.92) | 42.66(1.33) | 48.24(0.2) | 47.13(0.23) | 44.27(3.27) | **47.05**(1.42) |
| $P_{APR}$(BKG) | **53.75**(0.54) | 52.88(0.64) | 51.36(2.19) | 49.73(1.15) | **54.45**(2.34) | 54.09(0.91) | 52.37(4.86) | 54.32(1.07) |
| Time | 700.38(7.28) | 353.22(28.51) | 2914.77 (195.66) | **321.3**(5.45) | 4104.75 (2075.63) | **1070.84** (60.29) | 10250.01 (4416.1) | 2853.8 (1534.45) |

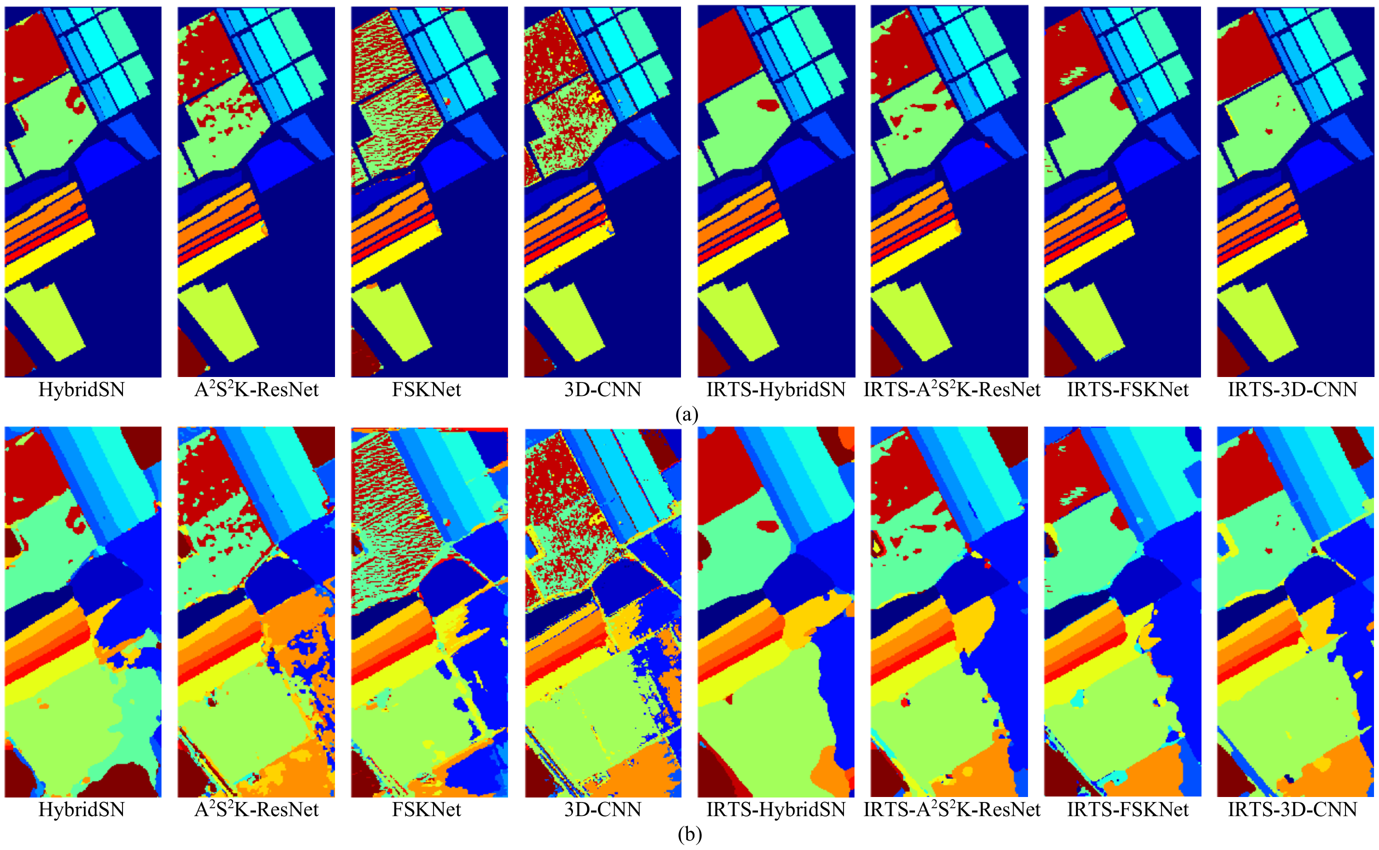


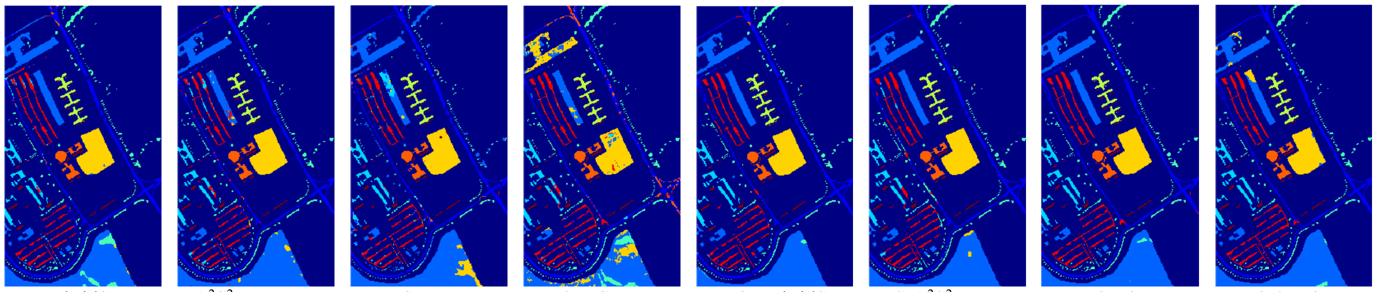
Fig. 17. Classification maps of HybridSN, $A^2S^2$K-ResNet, FSKNet, 3D-CNN, and their IRTS counterparts for Salinas using 1% of training samples. (a) Without BKG. (b) With BKG.

the quantitative results in Table XXIV did show that FSKNet and 3D-CNN could be tremendously improved using IRTS by 6% and 12%, respectively. Interestingly, the salt and pepper noise issue in the classification map produced by 3D-CNN was not as severe as that encountered in the Purdue data and Salinas.

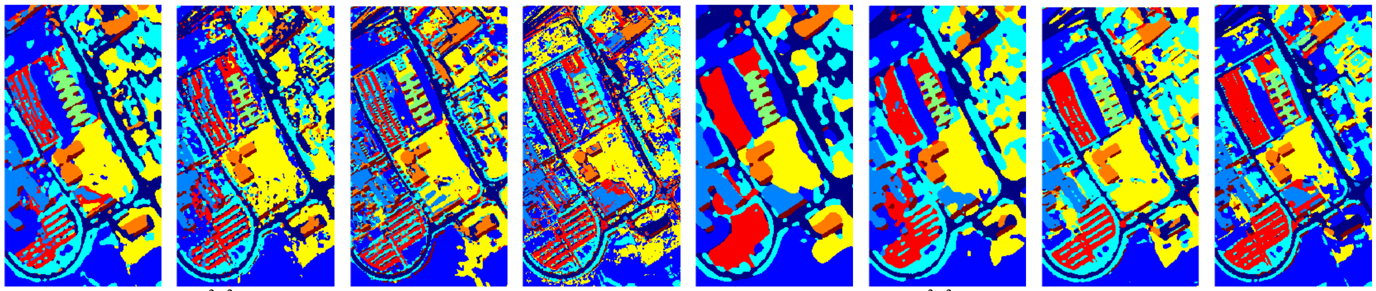
TABLE XXIV

$P_A$, $P_{\mathrm{OA}}$, $P_{\mathrm{AA}}$, $P_{\mathrm{PR}}$, $P_{\mathrm{OPR}}$, AND $P_{\mathrm{APR}}$ PRODUCED BY HYBRIDSN, A$^2$S$^2$K-RESNET, FSKNET, 3D-CNN, IRTS-HYBRIDSN, IRTS-A$^2$S$^2$K-RESNET, IRTS-FSKNET, AND IRTS-3D-CNN BY USING 1% TRAINING SAMPLES FOR U. OF PAVIA

| Method | **STATE-OF-THE ART 3D CNNS** | | | | **THEIR ITRS COUNTERPARTS** | | | |
|---|---|---|---|---|---|---|---|---|
| | HybridSN | A$^2$S$^2$K-ResNet | FSKNet | 3D-CNN | IRTS-HybridSN | IRTS-A$^2$S$^2$K-ResNet | IRTS-FSKNet | IRTS-3D-CNN |
| Class | $P_A(C_l)$ | $P_A(C_l)$ | $P_A(C_l)$ | $P_A(C_l)$ | $P_A(C_l)$ | $P_A(C_l)$ | $P_A(C_l)$ | $P_A(C_l)$ |
| 1 | 85.88(6) | 93.75(3.04) | **94.46**(2.39) | 87.86(6.52) | 96.24(2.51) | 94.29(1.06) | **99.16**(0.79) | 98.26(0.66) |
| 2 | 96.65(3.25) | **96.66**(1.62) | 94.55(6.14) | 82.29(15.83) | 98.2(0.59) | 99(0.56) | **99.51**(0.36) | 96.74(1.97) |
| 3 | 95.14(2.41) | 88.42(3.18) | **95.17**(3.86) | 90.7(4.32) | 99.68(0.31) | 90.76(1.23) | 99.46(0.94) | **99.92**(0.1) |
| 4 | 94.48(4.42) | **99.04**(0.26) | 78.46(29.85) | 94.9(2.22) | 92.13(1.78) | 97.46(0.46) | **98.69**(0.78) | 98.41(1.24) |
| 5 | 97.7(3.67) | **99.95**(0.09) | 99.88(0.21) | 99.78(0.44) | 99.5(0.57) | 99.95(0.09) | 99.54(0.74) | **100**(0) |
| 6 | 98.04(2.83) | **99.46**(0.47) | 93.32(9.9) | 78.44(19.95) | **100**(0) | **100**(0) | 99.99(0.01) | 98.1(1.47) |
| 7 | 99.87(0.22) | **100**(0) | 97.64(1.48) | 90.39(11.69) | 99.95(0.09) | 99.95(0.05) | 99.76(0.28) | **99.96**(0.1) |
| 8 | 88.05(6.2) | 85.6(5.29) | **91.26**(8.83) | 82.68(17.66) | 98.45(0.67) | 86(6) | **99.34**(0.31) | 99.15(0.91) |
| 9 | 95.78(2.46) | 99.65(0.42) | 99.12(1.52) | **99.8**(0.43) | 91.02(3.87) | 97.08(0.64) | 98.63(2.33) | **99.81**(0.25) |
| $P_{\mathrm{OA}}$ | 94.28(1.88) | **95.62**(0.95) | 93.35(4.59) | 85.24(6.55) | 97.71(0.51) | 96.77(0.07) | **99.43**(0.31) | 97.89(1.07) |
| $P_{\mathrm{AA}}$ | 94.62(1.07) | **95.84**(0.25) | 93.76(4.27) | 89.65(5.3) | 97.24(0.74) | 96.06(0.4) | **99.34**(0.35) | 98.92(0.49) |
| $P_{\mathrm{OPR}}$(BKG) | 19.45(0.38) | **19.72**(0.2) | 19.25(0.95) | 17.58(1.35) | 20.15(0.11) | 19.96(0.02) | **20.51**(0.06) | 20.19(0.22) |
| $P_{\mathrm{APR}}$(BKG) | 20.42(2.23) | **21.86**(1.2) | 21.85(2.52) | 19.97(1.93) | 20.63(0.52) | 21.81(0.58) | **22.12**(1.86) | 21.82(0.8) |
| Time | 1094.77 (11.27) | 1388.49 (22.38) | 1322.1 (11.69) | **310.0** (4.89) | 13306.52 (4781.79) | 5006.92 (114.45) | 9434.1 (2900.44) | **3363.9** (2357.81) |



(a)



(b)

Fig. 18. Classification maps of HybridSN, A$^2$S$^2$K-ResNet, FSKNet, 3D-CNN, and their IRTS counterparts for U. of Pavia using 1% of training samples. (a) Without BKG. (b) With BKG.

## *D. Discussions*

The above experiments provided evidence that the three datasets did have their own merits. Conclusions drawn from one dataset are not necessarily applied to another. Several interesting observations are worth being noted.

1) The best classifiers for the Purdue data, Salinas, and U. of Pavia were A$^2$S$^2$K-ResNet, HybridSN, and FSKNet, respectively. However, after IRTS was implemented, the best classifiers became IRTS-3D-CNN, HybridSN, and FSKNet, respectively. Nevertheless, IRTS-3D-CNN performed reasonably well compared to the best IRTS classifiers with only slight differences.
2) Despite the fact that IRTS-classifiers require very high computing times compared to those without using IRTS, the payoff is worthwhile. This is because the performance of including IRTS can be significantly improved at the expense of high computational complexity, specifically, IRTS-3D-CNN. In addition, if we take the network structure into account, then IRTS-3D-CNN is probably most preferable because it is easy to implement with simple architecture.
3) Among the three datasets, Salinas was the easiest one because it does not have an imbalance class issue as the Purdue data does and nor BKG issue as U. of Pavia does.
4) IRTS has shown to be very effective in improving 3D-CNN, specifically, the Purdue data. Despite the fact that 3D-CNN has a very simple network architecture,

TABLE XXV
ARCHITECTURE OF MODELS

| 3D-CNN | | | 3D-CNN-D | | |
|---|---|---|---|---|---|
| Layers | Type | Filter number | Layers | Type | Filter number |
| 1 | Convolutional | 2 | 1 | Convolutional | 2 |
| 2 | Pooling | - | 2 | Pooling | - |
| 3 | Convolutional | 4 | 3 | Convolutional | 4 |
| 4 | Fully connected | 112 | 4 | Convolutional | 8 |
| 5 | Fully connected | Class number | 5 | Fully connected | 112 |
| | | | 6 | Fully connected | Class number |

TABLE XXVI
$P_{OA}$, $P_{AA}$, $P_{OPR}$, AND $P_{APR}$ PRODUCED BY 3D-CNN, 3D-CNN-D, AND IRTS-3D-CNN BY USING 1% TRAINING SAMPLES FOR PURDUE DATA, SALINAS, AND U. OF PAVIA

| | Purdue **data** | | | Salinas | | | **U. of Pavia** | | |
|---|---|---|---|---|---|---|---|---|---|
| | 3D-CNN | 3D-CNN-D | IRTS-3D-CNN | 3D-CNN | 3D-CNN-D | IRTS-3D-CNN | 3D-CNN | 3D-CNN-D | IRTS-3D-CNN |
| Patch size | 5 | 15 | 5 | 5 | 21 | 5 | 5 | 19 | 5 |
| Batch size | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 |
| Training steps | 250 | 1000 | 250 | 250 | 1000 | 250 | 250 | 1000 | 250 |
| $P_{OA}$ | 56.26(2.71) | 62.52(0.83) | 82.56(5.17) | 91.71(1.01) | 96.3(0.57) | 97.04(0.52) | 85.62(4.66) | 96.41(1.77) | 98.84(0.75) |
| $P_{AA}$ | 70.65(0.88) | 75.1(1.25) | 90.8(2.84) | 95.64(0.44) | 98.6(0.27) | 98.48(0.21) | 86.03(3.22) | 97.24(0.9) | 99.2(0.19) |
| $P_{OPR}$(BKG) | 27.42(1.32) | 30.48(0.4) | 40.25(2.52) | 44.68(0.49) | 46.92(0.28) | 47.27(0.25) | 17.66(0.96) | 19.88(0.36) | 20.38(0.15) |
| $P_{APR}$(BKG) | 28.61(0.96) | 28.62(1.18) | 41.17(2.94) | 49.69(1.87) | 51.52(1.16) | 52.96(0.07) | 20.09(1.64) | 20.76(0.67) | 22.6(0.13) |
| Time | 20.9(0.15) | 445.99(5.4) | 373.35(158.8) | 81.77(0.49) | 883.86(90.37) | 770.11(18.78) | 126.11(5.24) | 1666.**5**(208.3) | 1563.9(397.**1**) |

its IRTS version is able to manage to achieve very high accuracy and performs very comparably to more complicated deep learning classifiers, such as A2S2K-ResNet, HybridSN, and FSKNet. This indicated that IRTS-3D-CNN not only proved to be effective but also showed its great potential in adaptability to any network.

5) Last but not least, according to Figs. 7–18, the classification maps produced by IRTS-3D-CNN with BKG included showed that IRTS-3D-CNN was able to smooth the irregularities between classes and their boundaries. Such a fact suggested that IRTS-3D-CNN can be considered a segmentation technique, which can partition an entire image into a finite number of homogeneous regions. This intriguing finding provides a further investigation into how to take advantage of IRTS-3D-CNN in image segmentation in addition to its capability in HSIC presented in this article.

## IX. COMPLEXITY COMPARISON BETWEEN CNN AND IRTS-CNN

It should be noted that there is no requirement that the step size of the spectral and spatial dimensions in 3D-CNN can be the same or different. In our experiments, the step size of 3D-CNN for both spectral and spatial dimensions was set to 1. There are two reasons for this choice. The first one is that using a small step size can capture more detailed information in the input data. As a tradeoff, a small step size may require more convolution operations to be performed and, thus, increase the complexity of the algorithm. The second reason is that our IRTS structure can not only reduce the complexity but also improve the performance by feeding back the new and additional spatial classification information to the model. To validate our assertion, this section shows that, even with the small setting of step size in the spectral and spatial domains, the complexity of the algorithm can be significantly reduced by coupling 3D-CNN with IRTS while also improving the classification accuracy.

In this section, we evaluated the effectiveness of our proposed IRTS structure in reducing the complexity of 3D-CNN models while maintaining high accuracy in HSIC. We conducted experiments with two 3D-CNN models: a simple 3D-CNN, as shown in Fig. 2, and a deeper 3D-CNN model, referred to as 3D-CNN-D, as shown in Table XXV. The input hyperspectral data were preprocessed by PCA to reduce the number of spectral bands to 30, 15, and 15 for the Purdue, Salinas, and U. of Pavia datasets, respectively.

The parameter settings and performance results are presented in Table XXVI. For both 3D-CNN and IRTS-3D-CNN, we used a patch size of 5 and trained the models for 250 steps for all three HSIs. In contrast, for 3D-CNN-D, we used a larger patch size of 15, 21, and 19 and trained the models for 1000 steps for Purdue, Salinas, and U. of Pavia, respectively. Our results show that the IRTS-3D-CNN model outperformed the 3D-CNN-D model in terms of accuracy and execution time.

We found that a larger patch size can provide more spatial information to improve classification accuracy, but it can also increase model complexity. On the other hand, smaller patch sizes can reduce complexity but may result in lower classification accuracy. Our IRTS structure allows us to use smaller patch sizes and fewer training steps while achieving high classification accuracy by feeding back both spectral and spatial information to the model.

## X. CONCLUSION

This article presents a general framework, which incorporates RTS into CNN for an iterative CNN network, called IRTS-CNN. It includes iterative feedback loops to make a

CNN a close network system while taking advantage of RTS to randomly reselect training samples in every new iteration, so as to improve CNN in classification performance. It is specifically effective when a limited number of training samples are used. There are several novelties.

1) Unlike CNN that is a feed-forward network system, IRTS-CNN is designed as a feedback network system. As a result, IRTS actually combines the advantages of both feed-forward and feed-backward information, so as to improve classification performance.
2) IRTS-CNN implements an iterative process to expand and augment the current data cube so that RTS can resample training samples from a new expanded data cube generated by each iteration. Such newly generated sets of resampled training samples contain more spatial classification information iteration by iteration that increases the classification ability of CNN.
3) IRTS-CNN includes an iterative process via feedback loops to make an open feed-forward CNN a close network system.
4) IRTS-CNN also implements an automatic stopping rule to terminate the iterative process.
5) As a result of using RTS, IRTS-CNN is capable of improving CNN, specifically, when only a limited number of training samples are available. This is because RTS allows CNN to resample training samples randomly via an iterative process, in which case some of the data samples that were not sampled at previous iterations may be sampled in a later iteration. Moreover, these new training samples are reselected from new expanded data cubes that include more crucial spatial classification that can increase classification accuracy for the next round iteration.
6) Finally, IRTS-CNN provides a new look at CNN. For example, a similar concept of using an iterative process to capture spatial information from anomaly detection maps has been recently investigated in [52]. Another example is to incorporate IRTS into semisupervised HSIC to improve classification performance, such as active learning in [41].

## REFERENCES

[1] C.-I. Chang, *Hyperspectral Data Processing: Algorithm Design and Analysis*. Hoboken, NJ, USA: Wiley, 2013.

[2] 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.

[3] P. Ghamisi, M. Dalla 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.

[4] 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.

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

[6] 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.

[7] S. Zhong, Y. Zhang, and C.-I. Chang, “A spectral–spatial feedback close network system for hyperspectral image classification,” *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 12, pp. 10056–10069, Dec. 2019.

[8] C. Chang, K. Y. Ma, C. Liang, Y. Kuo, S. Chen, and S. Zhong, “Iterative random training sampling spectral spatial classification for hyperspectral images,” *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 13, pp. 3986–4007, 2020.

[9] A. Verma, P. Singh, and J. S. Rani Alex, “Modified convolutional neural network architecture analysis for facial emotion recognition,” in *Proc. Int. Conf. Syst., Signals Image Process. (IWSSIP)*, Osijek, Croatia, Jun. 2019, pp. 169–173.

[10] G. Lou and H. Shi, “Face image recognition based on convolutional neural network,” *China Commun.*, vol. 17, no. 2, pp. 117–124, Feb. 2020.

[11] Z. Hu and E.-J. Lee, “Human motion recognition based on improved 3-dimensional convolutional neural network,” in *Proc. IEEE Int. Conf. Comput., Commun. Eng. (ICCCE)*, Fujian, China, Nov. 2019, pp. 154–156.

[12] K. Pai and A. Giridharan, “Convolutional neural networks for classifying skin lesions,” in *Proc. IEEE Region 10 Conf. (TENCON)*, Kochi, India, Oct. 2019, pp. 1794–1796.

[13] S. P. Kannojia and G. Jaiswal, “Ensemble of hybrid CNN-ELM model for image classification,” in *Proc. 5th Int. Conf. Signal Process. Integr. Netw. (SPIN)*, Noida, Feb. 2018, pp. 538–541.

[14] A. Kausar, M. Sharif, J. Park, and D. R. Shin, “Pure-CNN: A framework for fruit images classification,” in *Proc. Int. Conf. Comput. Sci. Comput. Intell. (CSCI)*, Las Vegas, NV, USA, Dec. 2018, pp. 404–408.

[15] B. Fang, Y. Li, H. Zhang, and J. Chan, “Hyperspectral images classification based on dense convolutional networks with spectral-wise attention mechanism,” *Remote Sens.*, vol. 11, no. 2, p. 159, Jan. 2019.

[16] M. E. Paoletti, J. M. Haut, R. Fernandez-Beltran, J. Plaza, A. J. Plaza, and F. Pla, “Deep pyramidal residual networks for spectral–spatial hyperspectral image classification,” *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 2, pp. 740–754, Feb. 2019.

[17] S. K. Roy, G. Krishna, S. R. Dubey, and B. B. Chaudhuri, “HybridSN: Exploring 3-D–2-D CNN feature hierarchy for hyperspectral image classification,” *IEEE Geosci. Remote Sens. Lett.*, vol. 17, no. 2, pp. 277–281, Feb. 2020.

[18] J. Zhu, L. Fang, and P. Ghamisi, “Deformable convolutional neural networks for hyperspectral image classification,” *IEEE Geosci. Remote Sens. Lett.*, vol. 15, no. 8, pp. 1254–1258, Aug. 2018.

[19] C. Yu, R. Han, M. Song, C. Liu, and C. Chang, “A simplified 2D-3D CNN architecture for hyperspectral image classification based on spatial–spectral fusion,” *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 13, pp. 2485–2501, 2020.

[20] X. Zhang et al., “Spectral–spatial self-attention networks for hyperspectral image classification,” *IEEE Trans. Geosci. Remote Sens.*, vol. 60, 2022, Art. no. 5512115.

[21] K. Pooja, R. R. Nidamanuri, and D. Mishra, “Multi-scale dilated residual convolutional neural network for hyperspectral image classification,” in *Proc. 10th Workshop Hyperspectral Imag. Signal Process., Evol. Remote Sens. (WHISPERS)*, Amsterdam, The Netherlands, Sep. 2019, pp. 1–5.

[22] N. He et al., “Feature extraction with multiscale covariance maps for hyperspectral image classification,” *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 2, pp. 755–769, Feb. 2019.

[23] S. Wan, C. Gong, P. Zhong, B. Du, L. Zhang, and J. Yang, “Multiscale dynamic graph convolutional network for hyperspectral image classification,” *IEEE Trans. Geosci. Remote Sens.*, vol. 58, no. 5, pp. 3162–3177, May 2020.

[24] Z. Gong, P. Zhong, Y. Yu, W. Hu, and S. Li, “A CNN with multiscale convolution and diversified metric for hyperspectral image classification,” *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 6, pp. 3599–3618, Jun. 2019.

[25] Z. Xu, H. Yu, K. Zheng, L. Gao, and M. Song, “A novel classification framework for hyperspectral image classification based on multiscale spectral–spatial convolutional network,” in *Proc. 11th Workshop Hyperspectral Imag. Signal Process., Evol. Remote Sens. (WHISPERS)*, Mar. 2021, pp. 1–5.

[26] W. Wang, S. Dou, and S. Wang, “Alternately updated spectral–spatial convolution network for the classification of hyperspectral images,” *Remote Sens.*, vol. 11, no. 15, p. 1794, Jul. 2019.

[27] Q. Zhu et al., “A spectral–spatial-dependent global learning framework for insufficient and imbalanced hyperspectral image classification,” *IEEE Trans. Cybern.*, vol. 52, no. 11, pp. 11709–11723, Nov. 2022.

[28] H. Chen, F. Miao, and X. Shen, "Hyperspectral remote sensing image classification with CNN based on quantum genetic-optimized sparse representation," *IEEE Access*, vol. 8, pp. 99900–99909, 2020.

[29] X. Cao, M. Ren, J. Zhao, H. Li, and L. Jiao, "Hyperspectral imagery classification based on compressed convolutional neural network," *IEEE Geosci. Remote Sens. Lett.*, vol. 17, no. 9, pp. 1583–1587, Sep. 2020.

[30] S. K. Roy, S. Manna, T. Song, and L. Bruzzone, "Attention-based adaptive spectral–spatial kernel ResNet for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 59, no. 9, pp. 7831–7843, Sep. 2021.

[31] S. K. Roy, J. M. Haut, M. E. Paoletti, S. R. Dubey, and A. Plaza, "Generative adversarial minority oversampling for spectral–spatial hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 60, 2022, Art. no. 5500615.

[32] K. Gao, W. Guo, X. Yu, B. Liu, A. Yu, and X. Wei, "Deep induction network for small samples classification of hyperspectral images," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 13, pp. 3462–3477, 2020.

[33] G. Li and C. Zhang, "Faster hyperspectral image classification based on selective kernel mechanism using deep convolutional networks," 2022, *arXiv:2202.06458*.

[34] G. Morales, J. Sheppard, R. Logan, and J. Shaw, "Hyperspectral band selection for multispectral image classification with convolutional networks," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2021, pp. 1–8.

[35] J. Wang, X. Song, L. Sun, W. Huang, and J. Wang, "A novel cubic convolutional neural network for hyperspectral image classification," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 13, pp. 4133–4148, 2020.

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

[37] M. Song, X. Shang, and C.-I. Chang, "3-D receiver operating characteristic analysis for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 58, no. 11, pp. 8093–8115, Nov. 2020.

[38] Accessed: May 2023. [Online]. Available: http://xudongkang.weebly.com/

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

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

[41] K. Y. Ma and C. Chang, "Iterative training sampling coupled with active learning for semisupervised spectral–spatial hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 59, no. 10, pp. 8672–8692, Oct. 2021.

[42] S. Zhong et al., "Class feature weighted hyperspectral image classification," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 12, no. 12, pp. 4728–4745, Dec. 2019.

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

[44] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*. New York, NY, USA: Academic, 1999, p. 366.

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

[46] H. V. Poor, *An Introduction to Signal Detection and Estimation*. New York, NY, USA: Springer, 1994.

[47] C. Chang, "Multiparameter receiver operating characteristic analysis for signal detection and classification," *IEEE Sensors J.*, vol. 10, no. 3, pp. 423–442, Mar. 2010.

[48] C. Chang, "An effective evaluation tool for hyperspectral target detection: 3D receiver operating characteristic curve analysis," *IEEE Trans. Geosci. Remote Sens.*, vol. 59, no. 6, pp. 5131–5153, Jun. 2021.

[49] C.-I. Chang and K. Y. Ma, "Band sampling of kernel constrained energy minimization using training samples for hyperspectral mixed pixel classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 60, 2022, Art. no. 5522721.

[50] G. Camps-Valls, L. Gomez-Chova, J. Munoz-Mari, J. Vila-Frances, and J. Calpe-Maravilla, "Composite kernels for hyperspectral image classification," *IEEE Geosci. Remote Sens. Lett.*, vol. 3, no. 1, pp. 93–97, Jan. 2006.

[51] Y. Tarabalka, M. Fauvel, J. Chanussot, and J. A. Benediktsson, "SVM- and MRF-based method for accurate classification of hyperspectral images," *IEEE Geosci. Remote Sens. Lett.*, vol. 7, no. 4, pp. 736–740, Oct. 2010.

[52] C. Chang, C. Lin, P. Chung, and P. F. Hu, "Iterative spectral–spatial hyperspectral anomaly detection," *IEEE Trans. Geosci. Remote Sens.*, vol. 61, 2023, Art. no. 5504330.

**Chein-I Chang** (Life Fellow, IEEE) received the Ph.D. degree in electrical engineering from the University of Maryland, College Park, MD, USA, in 1987.

He is currently a Professor with the University of Maryland, Baltimore County (UMBC), Baltimore, MD, USA. He has been a Chang Jiang Scholar Chair Professor and the Director of the Center for Hyperspectral Imaging in Remote Sensing (CHIRS), Dalian Maritime University, Dalian, China, since 2016. He is also a Yushan Scholar Chair Professor with National Cheng Kung University, Tainan, Taiwan. He has been a Chair Professor with National Yang Ming Chiao Tung University, Hsinchu, Taiwan, since 2019. He has authored 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 *Real-Time Recursive Hyperspectral Sample and Band Processing: Algorithm Architecture and Implementation* (Springer, 2017) and also edited three books: *Recent Advances in Hyperspectral Signal and Image Processing* (2006), *Hyperspectral Data Exploitation: Theory and Applications* (John Wiley & Sons, 2007), and *Advances in Hyperspectral Image Processing Techniques* (Wiley, 2023). He also coedited with A. Plaza the book *High Performance Computing in Remote Sensing* (CRC Press, 2007). His research interests include multispectral/hyperspectral image processing, automatic target recognition, and medical imaging.

Dr. Chang is a fellow of Society for Photo-Optical Instrumentation Engineers (SPIE). He is also serving as an Associate Editor for *Remote Sensing* and IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING.

**Chia-Chen Liang** received the B.S. degree in computer science from the National University of Kaohsiung, Kaohsiung, Taiwan, in 2014, the M.S. degree in electrical engineering from National Chung Hsing University, Taichung, Taiwan, in 2017, and the M.S. degree in electrical engineering from the University of Maryland, Baltimore County, Baltimore, MD, USA, in 2023, where she is currently pursuing the Ph.D. degree in electrical engineering with the Remote Sensing Signal and Image Processing Laboratory.

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

**Peter Fuming Hu** received the B.S. degree in electrical engineering from Shanghai University, Shanghai, China, in 1984, and the M.S. and Ph.D. degrees in computer science from the University of Maryland, Baltimore County (UMBC), Baltimore, MD, USA, in 1992 and 2013, respectively.

He has been with the School of Medicine, UMBC, since 1992, where he is currently a Professor with the Department of Anesthesiology, Surgery and Epidemiology. He is also an Adjunct Professor with UMBC. He is also the Chief Technologist with the University of Maryland Shock Trauma Center. He has more than 25 years of experience in clinical research at the Shock Trauma Center and the University of Maryland School of Medicine. During that time, he worked as a Principal Investigator (PI), a Co-PI, or a co-investigator on over 50 successfully completed peer-reviewed grants funded by various U.S. government agencies. His research interests include applying big data machine learning (AI) methods for the prediction of trauma patient outcomes.

Dr. Hu is also a member of the Editorial Board of *Critical Care Medicine* that is the leading clinical journal dedicated to Intensive Care Unit (ICU) patient care.