A New Convolutional Kernel Classifier for Hyperspectral Image Classification

Mohsen Ansari, Saeid Homayouni, *Senior Member, IEEE*, Abdolreza Safari, and Saeid Niazmardi

***Abstract*—Multiple kernel learning (MKL) algorithms are among the most successful classification methods for hyperspectral data. Nevertheless, these algorithms suffer from two main drawbacks of computational complexity and debility to admit to the end-to-end learning paradigm. This article proposed a convolutional kernel classifier (CKC) for hyperspectral remote sensing images to address these issues. The CKC uses the Nyström approximation method to estimate a low-rank approximation of the basis kernels, thus solves the issues associated with the high dimensionality of the basis kernels. The CKC uses deep architecture to learn the optimal combination of the basis kernels and the classification task to enable end-to-end learning. The proposed CKC's architecture is based on a one-dimensional-convolutional neural network (CNN-1-D), and it uses kernel dropout to prevent overfitting. It is the first instance of deep-kernel algorithms in the field of remote sensing. The proposed method was compared with several well-known hyperspectral image analysis MKL algorithms, including a multi-kernel variant of the deep kernel machine optimization, MKL-average, Simple-MKL, and generalize MKL, and state-of-the-art deep learning models, including Vanilla recurrent neural network (VanillaRNN) and CNN-1-D in classifying four benchmark hyperspectral datasets. The experimental results show that the CKC consistently outperforms all the competitor methods, and its runtime is lower than its MKL algorithm counterparts on four benchmark hyperspectral datasets. Moreover, the Nyström approximation solves the high dimensionality of the basis kernels and boosts classification accuracy. The source codes of CKC are available from: https://github.com/MohsenAnsari1373/A-New-Convolutional-Kernel-Classifier-for-Hyperspectral-Image-Classification.**

***Index Terms*—Convolutional neural network (CNN), deep kernel, hyperspectral classification, multiple kernel learning (MKL).**

I. Introduction

RECENT advances in electro-optical technology have led to hyperspectral sensors that can sample the object's visible and near-infrared spectral reflectance into hundreds of spectral bands. With the rapid development of hyperspectral sensors, considerable attention has been paid to spectral-based processing and analysis, including classification [1], [2], spectral unmixing [3], [4], dimensionality reduction [5], [6], and object detection [7]. Hyperspectral images (HSIs) allow more accurate classification than multispectral ones due to their high spectral resolution. However, the high spectral resolution of this data makes their classification more challenging and necessitates more advanced algorithms [1, 8, 9]. Consequently, HSI classification has become one of the most active research areas in remote sensing to overcome the challenges mentioned above and find proper solutions and efficient algorithms.

Some recent papers about deep learning classifiers for HSI, such as [10]–[12], emphasize that building highly nonlinear models are crucial to modern machine learning techniques to classify HSIs [13]. The success of deep neural networks (DNNs) in a wide variety of classification of HSI tasks has proven the superiority of highly nonlinear models [14]. In these cases, connecting the-state-of-the-art deep topologies with large datasets [12], [15], adopting effective optimization strategies [16], [17], graphical processing units utilization, and using a combination of multiple nonlinear transformations, with novel loss function, DNNs can easily approximate a considerable number of classes of function for classifying HSIs. As respects, by increasing complication of the DNNs, exhaustive tuning of several hyper-parameters is required, usually leading to model overfitting or suboptimal solutions [14]. Although advances in data augmentation and regularization techniques have partially addressed these issues [18], [19], in the classification of HSIs, it is challenging to propose DNNs that provide considerable performance improvements over conventional machine learning solutions [1, 14]. In this case, a more popular alternative solution to obtaining more effective and nonlinear models is to employ kernel methods [14].

Kernel-based methods have recently gained much attention among different proposed algorithms for HSI data classification thanks to their robust theoretical background [20]. These methods map input data from its original space into a higher dimensional reproducing kernel Hilbert space (RKHS), in which the data has a more straightforward representation. The inner product between all pairs of the mapped data points in the RKHS can be computed using their values in the input space using a kernel function.

The performance of kernel-based methods highly depends on the choice of kernel function and tuning the values of its

Manuscript received August 10, 2021; revised September 18, 2021 and October 10, 2021; accepted October 21, 2021. Date of publication October 27, 2021; date of current version November 15, 2021. *(Corresponding author: Mohsen Ansari.)*

Mohsen Ansari and Abdolreza Safari are with the School of Surveying and Geospatial Engineering, College of Engineering, University of Tehran, Tehran 14395-515, Iran (e-mail: moh.ansari@ut.ac.ir; asafari@ut.ac.ir).

Saeid Homayouni is with the Centre Eau Terre Environnement, Institut National de la Recherche Scientifique, Quebec G1K 9A9, Canada (e-mail: saeid.homayouni@ete.inrs.ca).

Saeid Niazmardi is with the Department of Surveying Engineering, Faculty of Civil and Surveying Engineering, Graduate University of Advanced Technology, Kerman 7631885356, Iran (e-mail: s.niazmardi@kgut.ac.ir).

Digital Object Identifier 10.1109/JSTARS.2021.3123087

hyperparameters [21]. In this regard, several model selection algorithms have been proposed to assist this choice, but arguably the most used methods are MKL algorithms [20]. MKL combines a set of predefined basis kernels into a composite kernel. The basis kernels are usually constructed by using different kernel functions or setting different values for their hyperparameters. To fuse the information contents of different kernel functions, MKLs construct a combination function using a wide variety of linear or non-linear combination functions.

Although MKL algorithms can be successfully used to address the problem associated with kernel parameter selections, some problems are yet to be solved. The first and foremost of these problems is the computational complexity of the kernels method, which arises from the dimensionality of kernel matrices [14]. The dimensionality of kernel matrices grows quadratically with the sample size, which impedes using these methods for big datasets. Besides, using several kernels for MKL algorithms escalates this problem [14]. In [20], [22]–[24], some accurate MKL algorithms are introduced to address this problem, but they are simply compatible with the small classification tasks. In a recent effort [25], Alioscha-Perez *et al.* connected the traditional batch solutions and stochastic gradient descent MKL approaches to introduce new methods for handling big data.

The second problem arises because the data representation provided by kernel functions is independent of the learning task [14], unlike the deep learning methods that learn a task-specific representation of the data through an end-to-end learning paradigm. It is well-established that one of the main reasons for the outstanding performance of the deep learning models is the end-to-end learning paradigm to which the kernel methods are inherently incapable of admitting [15].

To overcome the problems mentioned above, it has been tried to incorporate the advantages of both learning paradigms by proposing new methods over the last years. These methods, known as deep kernels, usually adopt different kernel functions as inputs to various DNNs [21], [26]. The neural network learns higher levels and task-specific data representations from the input kernel functions [27]. This idea was implemented by Cho and Saul [28] for the first time by introducing an arc-cosine kernel. Cho and Saul have shown that an arc-cosine kernel behaves similarly to an infinitive single-layer neural network, so they proposed to mimic the behavior of DNNs by the composition of arc-cosine kernels with the nested method [28]. Optimization of kernel learning by applying DNNs architecture ideas can also be done by Gaussian processes' marginal likelihood as published in [29]. In [30], a scalable deep kernel machine is introduced, which extracts features by multiple layers. Each feature extraction layer mimics an unsupervised MKL producer. This novel framework uses the arc-cosine kernel [28], a multiple kernel form of the proposed algorithm [31].

In [32], the authors proposed a new strategy by merging deep multilayer features (MDFs) with an extended region-aware multiple kernel learning (ER-MKL). The ER-MKL uses pre-learned classifiers to fuse the MDF-generated representations. Agethen and Hsu [33] proposed a novel convolutional long short-term memory (LSTM) that supports convolutional kernels' composition. In that study, a set of convolutional kernels used the convolutional LSTM network instead of a single convolutional kernel to form an MKL method. Lauriola *et al.* [27] claimed that, in contrast to the top fully-connected (FC) layer of a convolutional neural network (CNN), which extract finer global features and high-level representation, the intermediate convolutional layers encode contextual information and maintain the locality in the feature map. Thus, the authors merged these two layers by MKL methods and introduced the KerNET algorithm [27].

Some authors assumed that the traditional MKL methods are not robust enough to cope with complex problems because they find the composite kernel from one layer of kernels. Accordingly, in [34]–[36], the authors proposed novel learning paradigms by extending the single layer MKL to multilayer MKL as deep MKL (DMKL). In this way, in [34], a self-adaptive DMKL (SA-DMKL) method is proposed. In the SA-DMKL method, each layer's number and kernel's number is not fixed in the whole learning stage. SA-DMKL method first finds the optimal parameters of each kernel by grid-search method and then finds the optimal number and type of kernels in each layer using the generalization error bound. At the end of the learning process, if the appropriate parameters are not reached, the SA-DMKL method adds a new layer. In connection with the previous study, in [35], depth-width-scaling MKL (DWS-MKL) is introduced. The DWS-MKL architecture highly depends on input data. The input of each layer is the combined outputs of the previous layers along with the network's depth. The network's width is extended by parallelizing separate channels of these deep structures.

More recently, in [14], Song *et al.* proposed the M-DKMO method to optimize MKL. To summarize, they combine MKL with multilayer FC networks to develop end-to-end classifiers, in which approximated kernels (ensemble embedding) are learned as a feature extractor, and FC layers combine the features extracted by the kernels to infer task-specific MKL and use the kernel dropout to prevent the overfitting [14]. In the continuation, in [37], Instead of performing representation learning for each ensemble embedding using a multilayer FC network like M-DKMO, the authors proposed an ensemble DKMO (eDKMO) method to perform representation learning for one ensemble embedding. Compared with the DKMO, the eDKMO reduces computation and time-space complexity because only one FC network for representation learning.

This article proposes a novel convolutional kernel classifier (CKC) algorithm that can learn a task-specific representation using either big kernels or multiple kernels. In contrast to previous research works that cannot handle big kernels because of the high computational complexity of the kernels, CKC can support much more kernels than others. Additionally, the previous papers mostly use the FC architecture through which the more straightforward representations are produced than convolutional architecture. In this regard, the proposed CKC algorithm creates a dense embedding of the HSI in the RKHS by the Nyström method and then learns a task-specific representation using CNN-1-D. The CKC algorithm calculates a low-rank approximation of

kernel matrices using the Nyström approximation method [38] for the first time in remote sensing literature. Since these

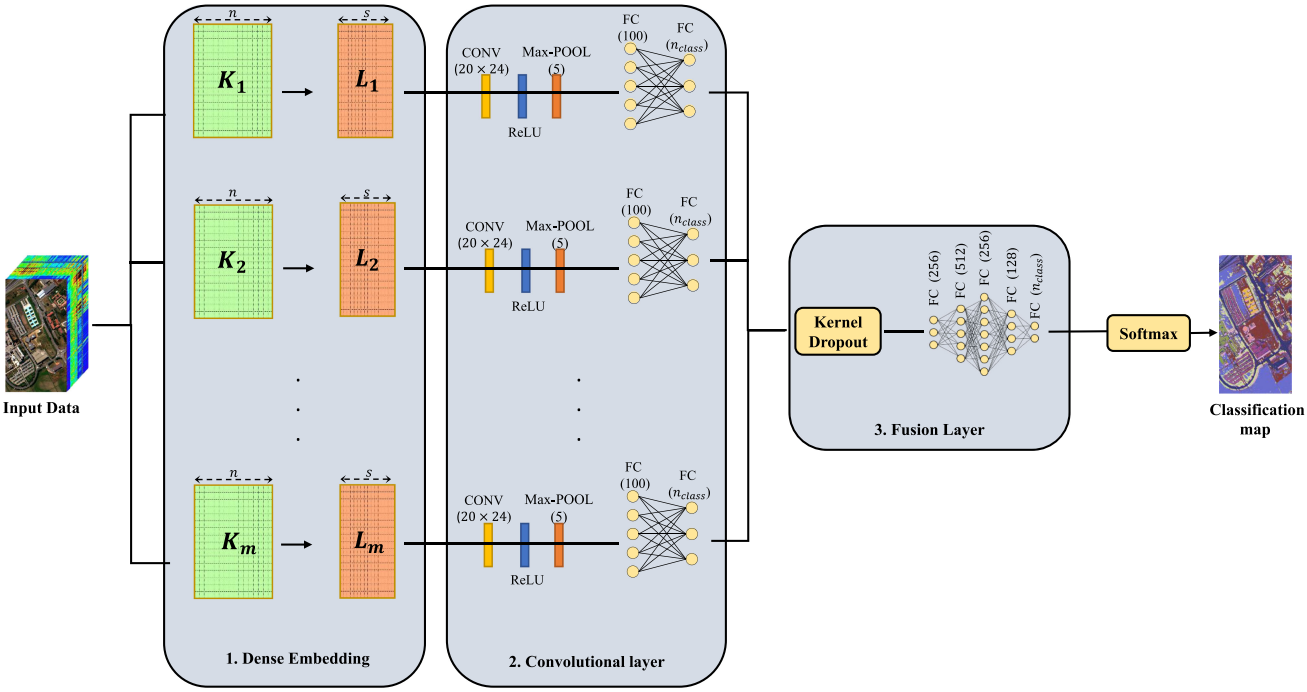

Fig. 1. Diagram of the proposed method.

approximated kernels have much less dimensionality than the original ones, the CKC algorithm can handle big or numerous kernels and cope with the first-mentioned problem (i.e., the computational complexity of the kernels method problems) and reduce the runtime of the algorithm.

Most importantly, the CKC algorithm takes advantage of a new architecture based on the CNN-1-D and the kernel dropout method [14] for the first time in deep kernel literature. Thanks to CNN-1-D architecture, the CKC can learn a task-specific representation, cope with the second problem above, and extract more features shared across the HSI via localized filters than the FC architecture. To the best of our knowledge, the CKC algorithm is the first instance of the deep-kernel algorithms in the field of remote sensing. Similar methods are proposed in [14] and [37] in machine learning, whose authors have used a multilayer FC network to represent learning. However, we used CNN-1-D in this article because its filter can process and extract the spectral features more robustly than multilayer FC networks [1], [15]. Besides, the CKC algorithm can be considered a novel optimization strategy in the MKL framework, which its application is not limited to a fixed combination function. The main contributions of this article are summarized as follows.

1) We developed the CKC as the first instance of deep kernel methods in remote sensing literature, which creates dense embedding of the HSI in the RKHS and learns a task-specific representation of the data through an end-to-end learning paradigm.
2) We created a dense embedding of HSI obtained from the Nyström approximation method for the first time in remote sensing literature to improve the effectiveness and reduce the computational complexity of the classification.
3) We introduced the CNN-1-D for representation learning for the first time in deep kernel methods literature because its filters can process the spectral signatures and spectral features in a more robust way than FC networks and traditional machine learning methods.
4) To prevent overfitting, we presented kernel dropout for the first time in HSI classification.

II. Methodology

The CKC algorithm consists of three main stages: dense embedding; convolutional layer; and fusion Layer. First, the predefined basis kernels were used in the dense embedding stage to map the input data into different RKHS. Additionally, we utilized the Nyström approximation to approximate kernels to reduce the computational complexity of the learning procedure. In the convolutional layer stage, a CNN-1-D architecture was used to learn a task-specific representation for each kernel. The last section used kernel dropout and FC architecture to combine features extracted by the previous stage. These stages are shown in Fig. 1.

A. Dense Embedding

The dense embedding stage is the critical component enabling end-to-end learning because the fusion and convolutional layers stages are separate from input, and the dense embedding stage links them to input data. A kernel matrix can be viewed as

$$K_{\mathrm{i,j}} = k\left(x_i, x_j\right) \tag{1}$$

where $\mathbf{K} \in \mathbb{R}^{n\times n}$ is the kernel matrix, k is a kernel function, x_i is ith pixel of HSI, and n is the number of training samples.

$K_{i,j}$ has large value if training samples belong to the same class as x_j and small value otherwise.

Although there are many kernel functions in computer vision and remote sensing literature, not all of them are useable and popular in HSIs classification. Radial basis function (RBF) and polynomial kernels are two widely used kernel functions in the field of HSI classification

$$k(x_i, x_j)_{\mathrm{RBF}} = \exp\left(-\|x_i - x_j\|^2 / \gamma^2\right) \tag{2}$$

$$k(x_i, x_j)_{\mathrm{polynomial}} = (\langle x_i, x_j \rangle + \alpha)^{\beta} \tag{3}$$

where (2) is related to RBF kernel, γ is a smoothing parameter, (3) is related to polynomial kernel, β is the degree of the polynomial kernel, and α is a number that usually considered as 0. The CKC can be used for any number and any type of kernels from one kernel to *m* (see Fig. 1).

Kernels are usually high-dimensional matrices. Accordingly, their direct use in the first layer of DNN is (or can be) computationally challenging. A possible approach to alleviate this challenge is transforming the original kernel matrix into a lower-dimensional and more tractable representation using kernel approximation methods [14]. Nyström approximation [38] and the random Fourier feature-based methods [39] are two of the most popular kernel approximation approaches. The random Fourier features-based methods are dedicated for shift-variant kernels [39].

The Nyström approximation method directly constructs dense embedding using the kernel matrix, where the source features are not accessible [14], [38]. In this method, an approximate kernel matrix $\mathbf{L} \in \mathbb{R}^{n \times r}$ can be found by a subset of s columns, selected from $\mathbf{K}$, where $\mathbf{K} \simeq \mathbf{L}\mathbf{L}^T$, $s \ll n$, and $r \leq s$. In this article, we extract approximate kernels through random sampling of the kernel matrix. Thus, s columns are randomly selected from $\mathbf{K}$ without replacement as a landmark. A single set containing the s selected columns are considered as $\mathbf{E} \in \mathbb{R}^{n \times s}$, the intersection of s and corresponding rows of $\mathbf{K}$ is denoted as $\mathbf{W} \in \mathbb{R}^{s \times s}$, the optimal rank-$r$ approximation of $\mathbf{W}$ obtained using truncated SVD is denoted as $\bar{\mathbf{W}}_r$, and the rank-r approximation of $\mathbf{K}$ is considered as $\bar{\mathbf{K}}_r$ which is obtained by

$$\bar{\mathbf{K}}_r = \mathbf{E}\bar{\mathbf{W}}_r\mathbf{E}^{\mathrm{T}}. \tag{4}$$

The time complexity of the kernel approximation corresponds to performing SVD on $\mathbf{W}$, which reduces to $\mathcal{O}(s^3)$ and can be more reduced by randomizing the SVD algorithm, as shown in [40]. The final form of approximated kernel mapping function $\mathbf{L}$ can be gathered by [40]

$$\mathbf{L} = \mathbf{E}\left(\mathbf{U}_{\bar{\mathbf{W}}_r}\Lambda_{\bar{\mathbf{W}}_r}^{-1/2}\right) \tag{5}$$

where $\Lambda_{\bar{\mathbf{W}}_r}$ and $\mathbf{U}_{\bar{\mathbf{W}}_r}$ are top r Eigenvectors and Eigenvalues of $\mathbf{W}$. So, proposed methods can work with $\mathbf{L}$ instead of $\overline{\mathbf{K}}_r$. An ensemble of embedding of HSIs obtained from Nyström approximation methods reduces the computational complexity of the classification task [41].

B. Convolutional Layer

In this stage, convolutional representation learning is performed for each dense embedding using a CNN-1-D to facilitate and improve the effectiveness of the task-specific classification. CNN-1-Ds are composed of one-dimensional convolutional layers (CONVs), activation function, and pooling layers (POOLs), which are explained in detail in [15]. As shown in Fig. 2, every row of the dense embedded data is considered a dense embedded vector, considered input data for the convolutional layer. In this regard, CONV can be considered a traditional sliding window [42], [43], where Q fixed-size one-dimensional filters overlap the q size over the dense embedded vector. The convolutional representation is resulted after applying the activation function (ReLU) and POOL. At the end of the convolutional layer stage, the fully connected architecture receives the convolutional representation and produces the final representation. The CNN-1-D models can extract the features which are shared across the HSI via localized filters.

C. Fusion Layer

The fusion layer, which can choose from various fusion strategies to access the final multiple kernel representation for the classification task, receives each kernel's learned representation. Concatenation, multiplication, averaging, and summation are the most common fusion strategies. Although the backpropagation algorithm can jointly optimize both the parameters of the fusion layer and those of the representation learning, the combination of different kernels and the significant number of parameters may overfit the learning process [14]. In [14], Song *et al.* introduced *kernel dropout* to alleviate this problem.

Kernel dropout has been inspired by typical dropout, introduced in [17]. For training DNNs, a typical dropout randomly chooses neurons to remove from the network along with all their outgoing and incoming connections [17]. Kernel dropout drops the kernel representation learned from some randomly chosen kernel dense embedding to optimize the fusion layer [14]. The total number of kernels is denoted by P, the hidden layer before the fusion layer is denoted as $\mathbf{\Omega} = \{\omega_p\}_{p=1}^{P}$, a vector associated with P independent Bernoulli trials is considered as t, the representation which is dropped from fusion layer if t_p is 0, is denoted as ω_p. Therefore, the feed-forward is expressed as [14], [17]

$$t_p \sim \mathrm{Bernoulli}\,(P) \tag{6}$$

$$\bar{\mathbf{\Omega}} = \{\boldsymbol{\omega} | \omega \in \Omega \text{ and } t_p > 0\} \tag{7}$$

$$\bar{\boldsymbol{\omega}} = (\omega_j)\,,\ \omega_j \in \bar{\mathbf{\Omega}} \tag{8}$$

$$\bar{y}_j = f\left(w_j\bar{\boldsymbol{\omega}} + b_j\right) \tag{9}$$

where the weight vector for hidden neuron j is denoted as $\mathrm{w_j}$, the vector concatenation is considered as $(\cdot)$, the softmax activation function is considered as f, and the output of the softmax activation function is considered as $\bar{y}_j$. In [14], Song *et al.* reported that the kernel dropout leads to speed-up convergence of the network and improves learning performance. After concatenating the latent representation of kernels by concatenation layer and kernel dropout, in order to fusion those,

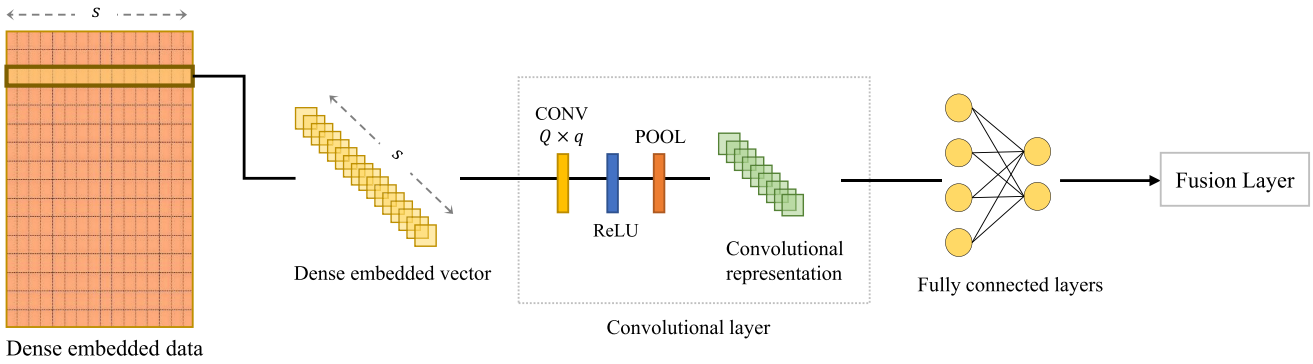

Fig. 2. Convolutional layer stage.

TABLE I
USED HSI DATSETS AND THE NUMBER OF THEIR LABELED SAMPLES

Salinas Valley		**University of Houston**		
Class name	No. of samples	Class name	No. of samples	
			Train	Test
Brocoli_green_weeds_1	2,009	Grass healthy	198	1,053
Brocoli_green_weeds_2	3,726	Grass stressed	190	1,064
Fallow	1,976	Grass synthetic	192	505
Fallow_rough_plow	1,394	Tree	188	1,056
Fallow_smooth	2,678	Soil	186	1,056
Stubble	3,959	Water	182	143
Celery	3,579	Residential	196	1,072
Grapes_untrained	11,271	Commercial	191	1,053
Soil_vinyard_develop	6,203	Road	193	1,059
Corn_senesced_green_weeds	3,278	Highway	191	1,036
Lettuce_romaine_4wk	1,068	Railway	181	1,054
Lettuce_romaine_5wk	1,927	Parking lot1	192	1,041
Lettuce_romaine_6wk	916	Parking lot2	184	285
Lettuce_romaine_7wk	1,070	Tennis court	181	247
Vinyard_untrained	7,268	Running track	187	473
Vinyard_vertical_trellis	1,807			

University of Pavia		**Berlin**	
Class name	No. of samples	Class name	No. of samples
Asphalt	6,631	Vegetation	20,148
Meadows	18,649	Built-up	28,752
Gravel	2,099	Impervious	30,589
Trees	3,064	Soil	5,834
Painted metal sheets	1,345	Water	4,677
Bare Soil	5,029		
Bitumen	1,330		
Self-Blocking Bricks	3,682		
Shadows	947		

the FC architecture is used to access the final multiple kernel representation for the HSIs classification task.

III. HYPERSPECTRAL DATASETS AND EXPERIMENTAL SETUP

This article selected four widely-used HSI datasets in HSI classification to evaluate the proposed method. It should be noted that all HIS datasets have been normalized in advance. Tables I and II give the details of four HSI datasets.

In this article, we have evaluated the proposed method using three experiments. In the first experiment, first of all, we evaluate the performance of the CKC with different values of the landmark parameter (s) and the kernel dropout to find the optimal value of the s and the kernel dropout, and then we try to illustrate the performance of the CKC during the training and validation stages on all datasets. In this experiment, we find out how the loss function of CKC converges to the low value during the training and validation stages on all datasets.

TABLE II
HSI DATASETS DETAILS

Dataset	U. of Pavia	Salinas Valley	U. of Houston	Berlin
Sensor	ROSIS	AVIRIS	CASI	HyMap
Spatial resolution(m)	1.3	3.7	2.5	3.5
Spectral range (μm)	0.43 – 0.86	0.4 – 2.5	0.38 – 1.05	0.4 – 2.5
Type of scene	Urban	Agriculture	Urban	Urban
row × column × band	610 × 340 ×103	512 × 217 ×224	349 × 1905 ×144	300 × 300 ×114
Num. of classes	9	16	15	5
Num. of samples	207,400	111,104	2,832 (Training) 12,197 (Test)	90,000

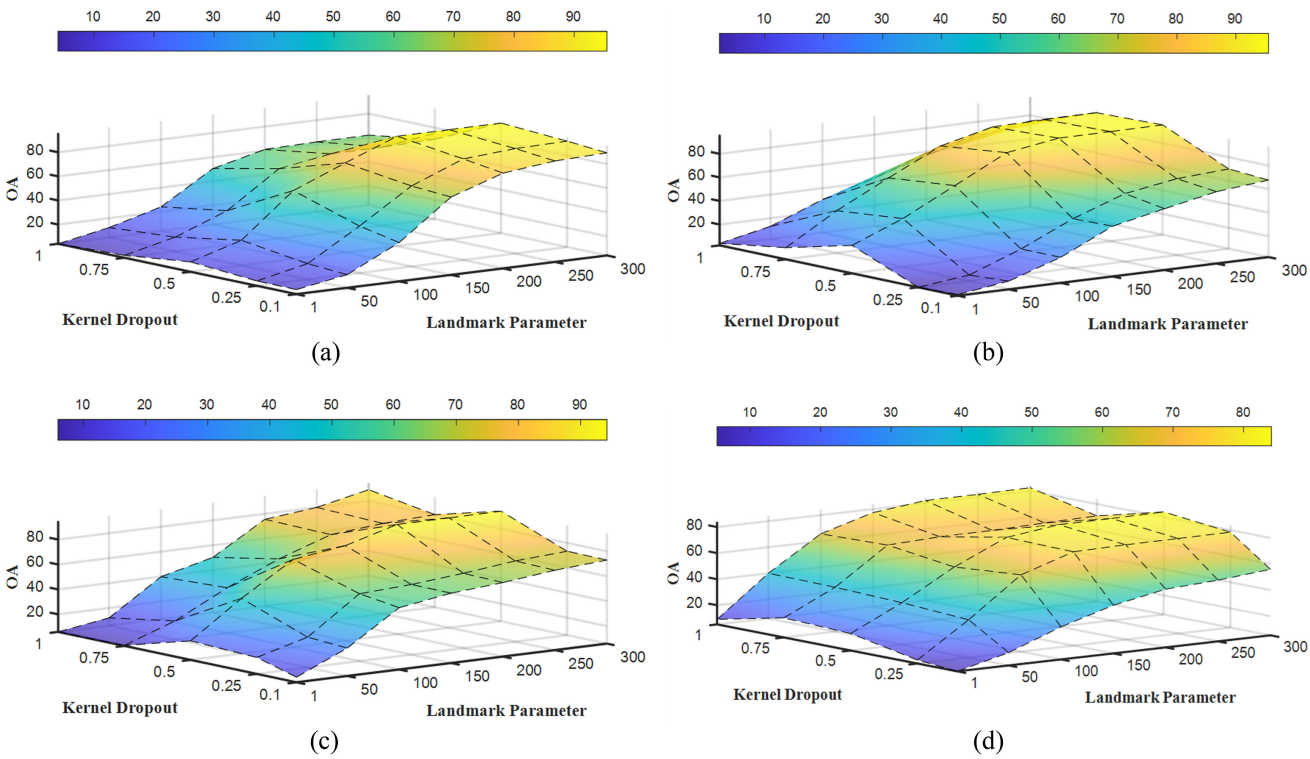

Fig. 3. Sensitivity of CKC to the change of the landmark parameter (s) and the kernel dropout on (a) the University of Pavia, (b) Salinas Valley, (c) Berlin, and (d) the University of Houston.

Additionally, we try to determine whether the kernel dropout can reduce the overfitting or not. Therefore we compare the CKC's using kernel dropout and concatenation method (i.e., simple mode). To this end, 20% of the labeled data of the University of Pavia, Salinas Valley, and Berlin datasets and training dataset of the University of Houston [44] are used to construct RBF kernel, third and second polynomial degree kernel, and the linear kernel is used as basis kernel. The optimal γ parameters for The RBF kernel for the University of Pavia, Salinas Valley, Berlin, and the University of Houston datasets were set to 2, 0.25, 0.125, and 0.03125, respectively [15]. The Adam optimizer was carried out to optimize the network, with the learning rate of 0.001 for the University of Houston and Berlin datasets and 0.0008 for the University of Pavia and Salinas Valley datasets [15]. The categorical cross-entropy loss function was used to evaluate the performance of the CKC during the training and validation process because of its relevant results in HIS classification in the previous studies [1], [15]. The CKC has been trained using 300 epochs for all of the datasets.

In the second experiment, we evaluated every three stages of the CKC on its classification accuracy and runtime. To this end, 20% of the labeled data of the University of Pavia, Salinas Valley, and Berlin datasets and training dataset of the University of Houston [44] are used. We then evaluated the performances of the CKC algorithms in the following four scenarios.

(1) In this scenario, RBF kernel, third and second polynomial degree kernel, and linear kernel are considered input data without Nyström approximation. The FC architecture (see Table IV) is used instead of the convolutional layer stage. In the fusion layer of this scenario, the kernel dropout method is omitted.

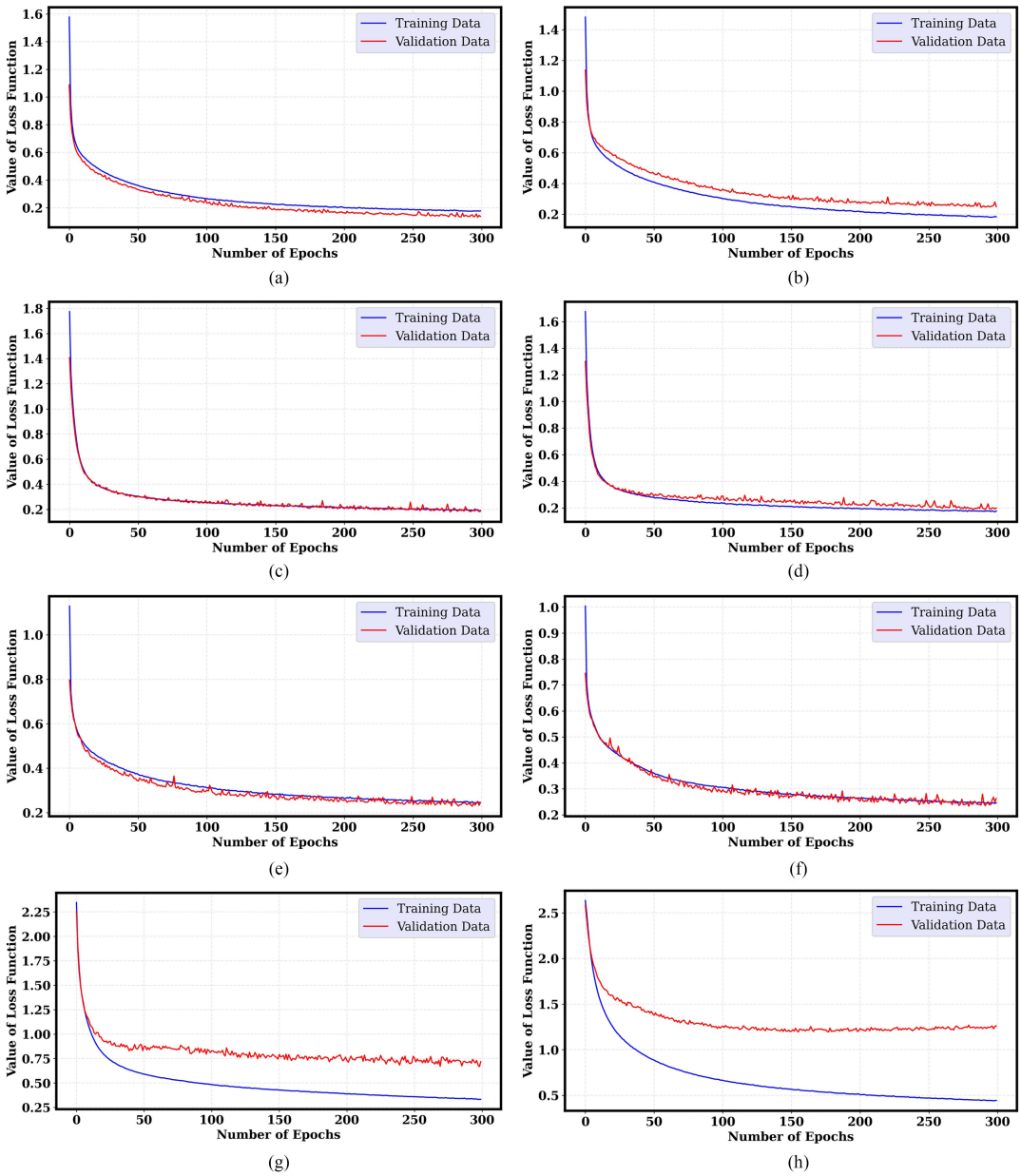

Fig. 4. CKC convergence during both training and validation process over (a) the University of Pavia, (c) Salinas Valley, (e) Berlin, and (g) the University of Houston datasets using the kernel dropout and CKC convergence during both training and validation process over: (b) the University of Pavia; (d) Salinas Valley; (f) Berlin; and (h) the University of Houston datasets using the concatenation method.

TABLE III
ADOPTED PARAMETERS FOR IMPLEMENTING CNN-1-D. THE NUMBER IN THE PARENTHESES SHOWS THE DIMENSIONS OF KERNELS IN CONV AND MAX-POOL LAYERS AND THE NUMBER OF NEURONS IN FC LAYERS

Layers	Activation Function
CONV (20×24)	ReLU
Max-POOL(5)	-
FC(100)	ReLU
FC (n_{class})	ReLU

TABLE IV
ADOPTED PARAMETERS FOR IMPLEMENTING FC ARCHITECTURE. THE NUMBER IN THE PARENTHESES SHOWS THE NUMBER OF NEURONS

Layers	Activation Function
FC(256)	ReLU
FC(512)	ReLU
FC(256)	ReLU
FC (128)	ReLU
FC (n_{class})	softmax

(2) The difference between this scenario and the 1) scenario is the dense embedding, which means that the four kernels, as mentioned earlier, are approximated by the Nyström approximation.

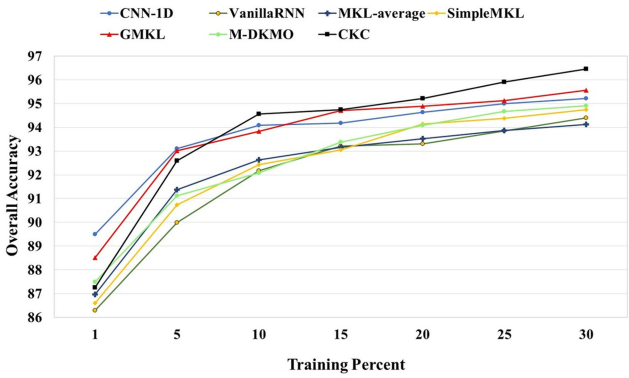

Fig. 5. OA evolution of each mentioned classifier with different training percentages over (a) the University of Pavia, (b) Salinas Valley, (c) Berlin, and (d) the University of Houston datasets.

(3) The difference between this scenario and the 2) scenario is kernel dropout, used in the fusion layer section. It is noteworthy that this architecture is introduced in [14] as M-DKMO.
(4) This scenario is our proposed method. The difference between this scenario and the 3) scenario is the CNN-1-D architecture of the representation learning stage.

Table III gives the topology of the CNN-1-D, which Is used in the convolutional layer stage.

Table IV gives the topology of the FC architecture in the fusion layer stage. Additionally, dropout with a fixed rate of 0.5 and batch normalization are used after every hidden layer in the training process to avoid exploding/vanishing gradient.

In the third experiment, we have compared CKC with M-DKMO [14], SimpleMKL [45], GMKL [46], and MKL-average [47], which are the most successful MKL algorithms in the literature, and two state-of-the-art deep learning methods, vanilla recurrent neural network (VanillaRNN) [15], [48] and CNN-1-D [15]. Besides comparing general performances of the methods mentioned above, this experiment aims to analyze how the different number of training samples affect the performances of these algorithms. This experiment was conducted using the same kernels and parameter setting as the previous one. We evaluated the methods using 1%, 5%, 10%, 15%, 20%, 25%, and 30% of the labeled data of the University of Pavia, Salinas Valley, and the Berlin datasets and training dataset of the University of Houston. In this article, a support vector machine (SVM) classifier is used for MKL-average algorithms. Using the five-fold cross-validation method, optimal C parameter for SVM, SimpleMKL, and GMKL for the University of Pavia, Salinas Valley, Berlin, and the University of Houston datasets was respectively set to 1×10^3 , 1×10^3, 200, and 1×10^5.

In order to assess the classification results for all three experiments, five quantitative metrics were employed, the overall accuracy (OA); the average accuracy (AA); the Kappa coefficient; the recall accuracy (per class); and McNemar [49]. Our experiments have been conducted on a hardware environment composed of an 8th-generation Intel R Core TM i7- 8550 K processor, with 8 MB of Cache and a processing speed of 4.20 GHz with four cores/8 way multitask processing. It includes 16 GB of DDR4 RAM with 2400 MHz serial speed. The software environment consists of Microsoft Windows 10 and Python 3.9 as the programming language.

IV. Experimental Result

A. Results of the First Experiment

Fig. 3 shows the performance of the CKC with different values of the landmark parameter (s) and the kernel dropout. Although the OA has increased with the increase of landmark parameters, the difference in OA in case 200 onward has not been noticeable. Therefore, the optimal combination of landmark parameter and kernel dropout for all datasets set 200 and 0.5, respectively.

Fig. 4 illustrates the convergence of the CKC during the training and validation stages on all datasets, where the horizontal and vertical axes represent the number of epochs and the value of loss function, respectively. From the Fig. 4 (a), (c), (e), and (g),

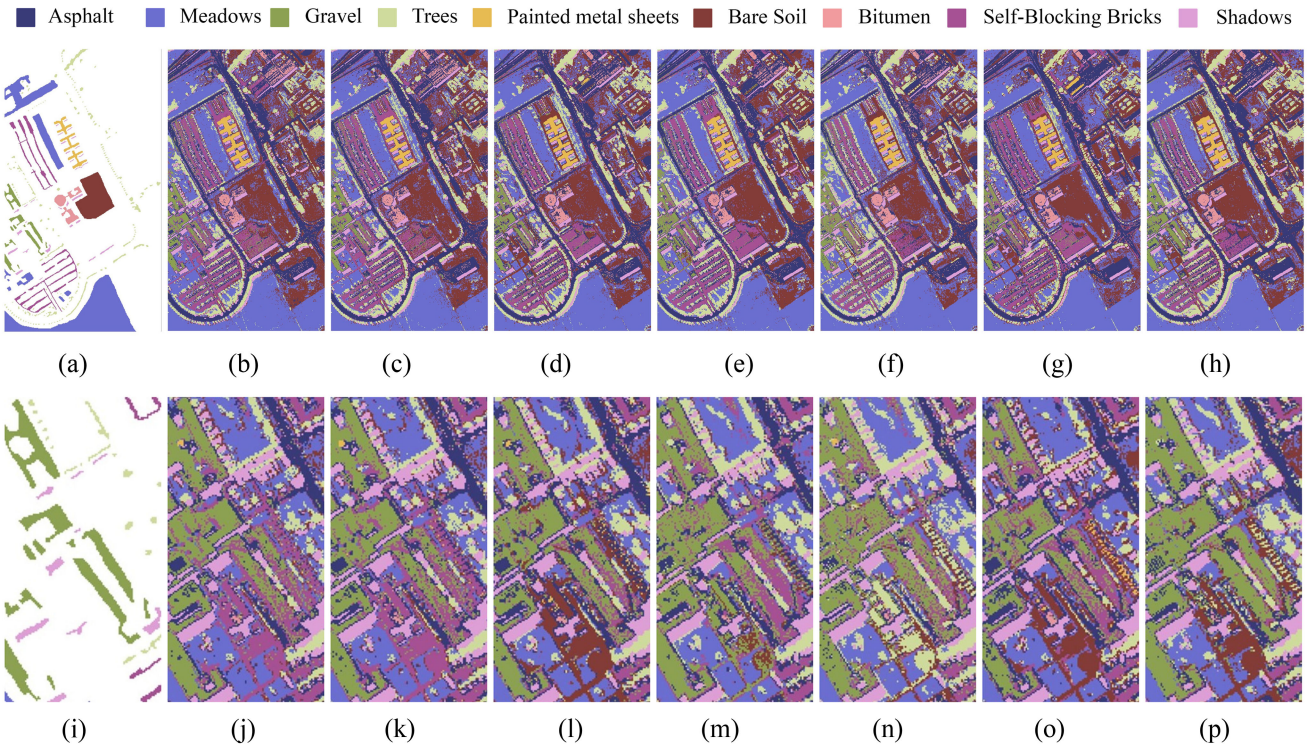

Fig. 6. Ground truth and the classification maps obtained from classifying the University of Pavia dataset using different methods. (a) Ground truth. (b) MKL-average. (c) SimpleMKL. (d) Generalize MKL. (e) Multikernel variant of the deep kernel machine optimization. (f) Vanilla recurrent neural network. (g) One-dimensional-convolutional neural network. (h) Convolutional kernel classifier. The close-up of ground truth and the classification maps of Gravel class using different methods. (i) Ground truth. (j) MKL-average. (k) SimpleMKL. (l) Generalize MKL. (m) Multikernel variant of the deep kernel machine optimization. (n) Vanilla recurrent neural network. (o) One-dimensional-convolutional neural network . (p) Convolutional kernel classifier.

TABLE V
EFFECTS OF EACH STAGE OF CKC ON HSIS CLASSIFICATION ACCURACY

Scenario	Overall Accuracy			
	U. of Pavia	Salinas Valley	Berlin	U. of Houston
(a) : FC	89.18	90.04	87.85	75.97
(b) : FC + Nyström	93.04	93.36	90.61	77.12
(c) : M-DKMO	94.09	94.27	92.20	79.33
(d) : CKC	95.22	95.23	93.94	83.41

TABLE VI
EFFECTS OF EACH STAGE OF CKC ON HSIS CLASSIFICATION RUNTIME

Scenario	Runtime(.s)			
	U. of Pavia	Salinas Valley	Berlin	U. of Houston
(e) : FC	490.98	570.69	521.98	489.61
(f) : FC + Nyström	270.61	321.81	342.13	202.48
(g) : M-DKMO	267.32	318.98	337.59	198.63
(h) : CKC	202.80	298.46	203.31	131.49

it can be seen that the value of loss function of CKC decreases with the number of training epochs, and the training value of loss function is approximately equal to the validation value of loss function at the 300 th epoch in the case of the University of Pavia, Salinas Valley, and the Berlin datasets. In Salinas Valley, the CKC converges faster than other cases and reaches the low loss function value after a few epochs. In the case of the University of Houston, the differences between the training value of loss function and the validation value of loss function are higher than those obtained in other cases. Additionally, by comparing the convergence of the CKC during the training and validation stages in the case of using the kernel dropout (i.e., Fig. 4 (a), (c), (e), and (g)) with using the concatenation method (i.e., Fig. 4 (b), (d), (f), and (h)), it can be concluded that the the kernel dropout can reduce the overfitting, especially in the University of Houston dataset. In better words, the differences between the

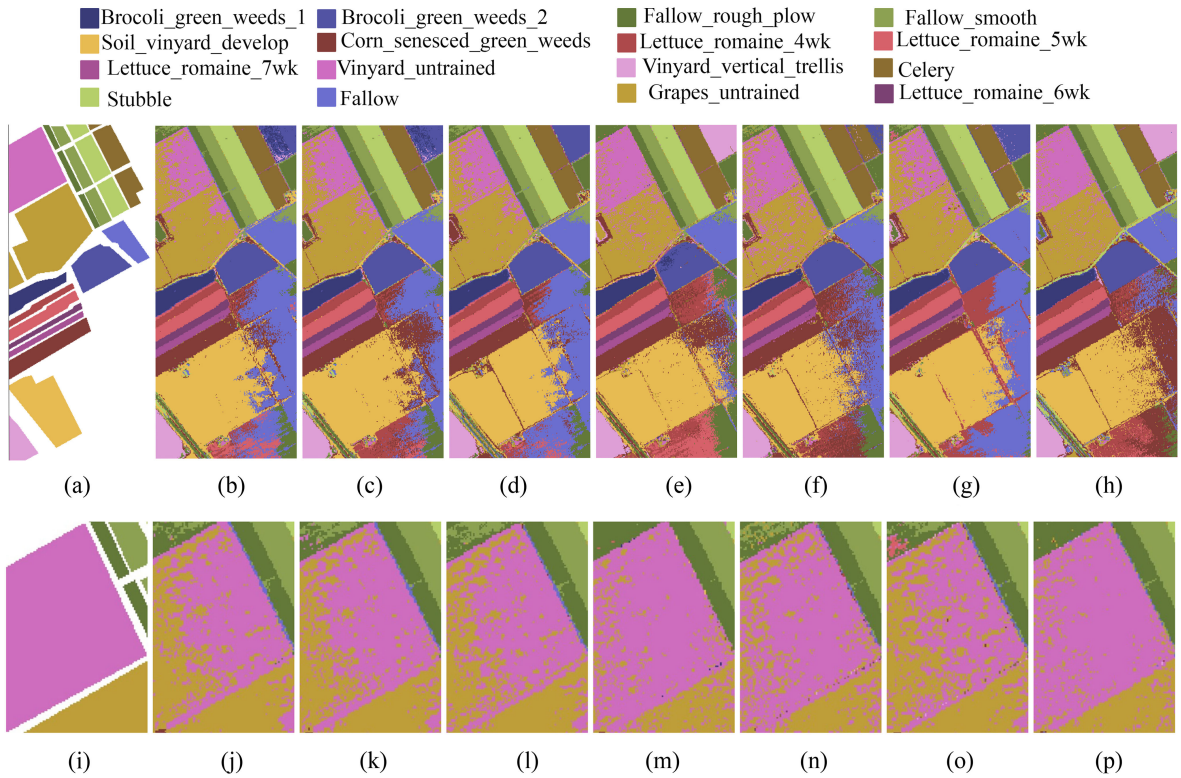

Fig. 7. Ground truth and the classification maps obtained from classifying the Salinas Valley dataset using different methods: (a) Ground truth. (b) MKL-average. (c) SimpleMKL. (d) Generalize MKL. (e) Multikernel variant of the deep kernel machine optimization. (f) Vanilla recurrent neural network. (g) One-dimensional-convolutional neural network. (h) Convolutional kernel classifier. The close-up of ground truth and the classification maps of Vinyard-untrained class using different methods. (i) Ground truth. (j) MKL-average. (k) SimpleMKL. (l) Generalize MKL. (m) Multikernel variant of the deep kernel machine optimization. (n) Vanilla recurrent neural network. (o) One-dimensional-convolutional neural network. (p) Convolutional kernel classifier.

training value of loss function and the validation value of loss function are closer to each other using the kernel dropout rather than using simple concatenation method.

B. Results of the Second Experiment

Tables V and VI give the classification performance as OA and runtime on four used datasets and scenarios, respectively. The importance of dense embedding (i.e., Nyström approximation) is evident from the comparison between scenarios (a) and (b) in Tables V and VI. Nyström approximation not only increased the classification accuracies by about 3.86% for the University of Pavia dataset, 2.96% for the Salinas Valley dataset, 2.76% for the Berlin dataset, and 1.15% for the University of Houston dataset, but also decreased the runtimes by about 220.37 s for the University of Pavia dataset, 248.88 s for the Salinas valley dataset, 179.85 s for the Berlin dataset, and 287.13 s for the University of Houston dataset. This fact substantiates the claim that using the Nyström approximation method reduces the computational complexity and improves classification performance. The results show that kernel dropout has also increased the classification accuracy and decreased the runtime for all datasets. For example, the kernel dropout has increased network accuracy for Berlin and the University of Houston datasets by 1.41% and 2.21%, respectively. The increased performance of classification accuracy, the decreased runtime, and (d) compared to scenario (c) is due to the convolutional layer. As an example, it can be seen that the obtained classification accuracy of Berlin and the University of Houston datasets has increased 1.74% and 4.08% using the convolutional layer. Most importantly, the runtime decreased by about 64.52 s for the University of Pavia dataset, 20.52 s for the Salinas Valley dataset, 134.28 s for the Berlin dataset, and 67.14 s for the University of Houston dataset using the convolutional layer.

According to these results, the three mentioned stages could be arranged in the dense embedding stage, convolutional layer stage, and kernel dropout stage, based on increasing classification accuracy and decreasing runtime. Additionally, each stage of the proposed algorithm has been effective in improving accuracy and runtime.

C. Results of the Third Experiment

Fig. 5 shows the performance of CKC and all other methods trained with different percentages of labeled data. From analyzing Fig. 5, we can observe that the behavior of SimpleMKL and MKL-average is very similar, especially in the University of Pavia and Salinas Valley datasets. Although their performance is acceptable, they are not comparable to the CKC. Additionally, the performance of the MKL-average method in the University of Houston dataset has been abysmal. The performance of the GMKL is better than the SimpleMKL and

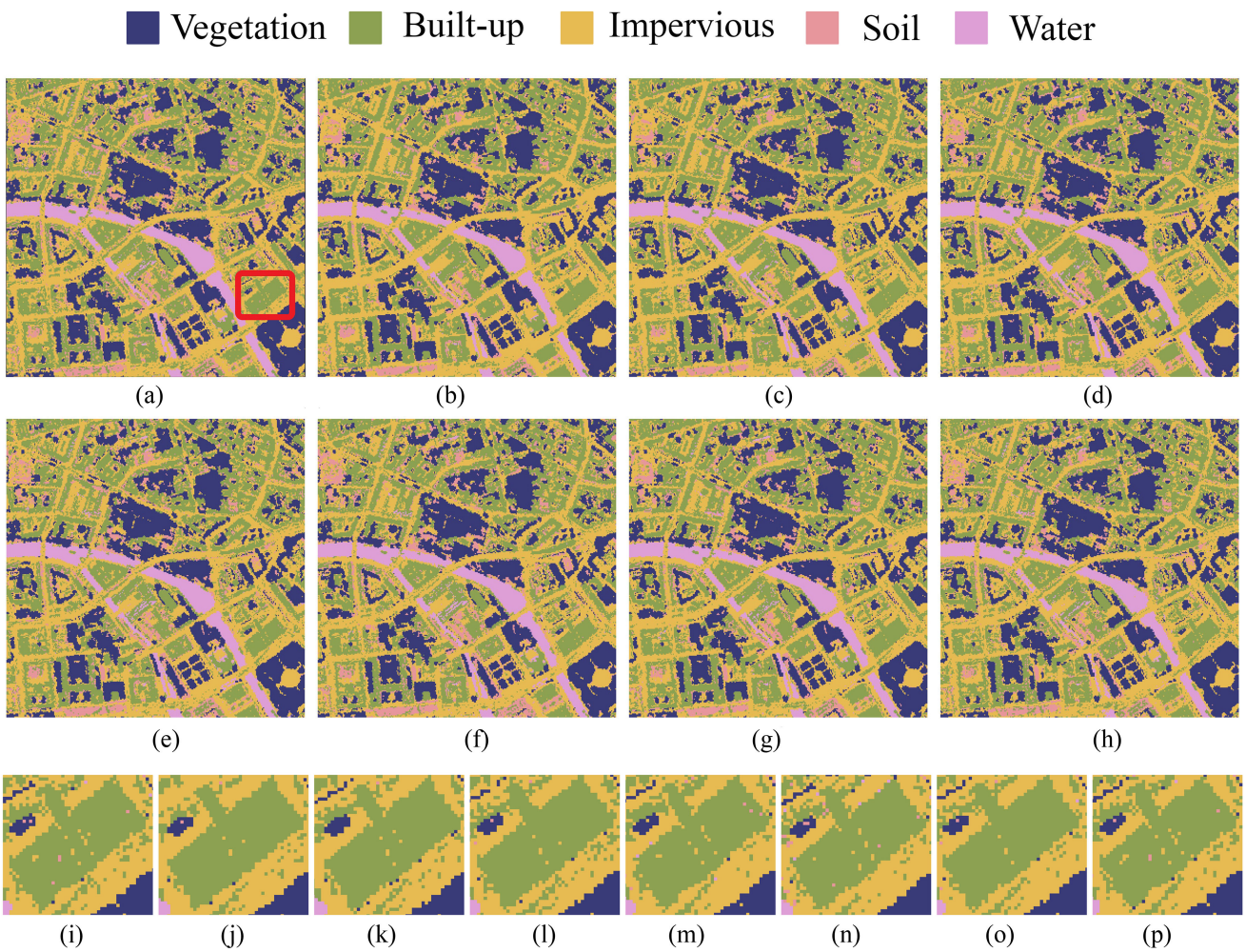

Fig. 8. Ground truth and the classification maps obtained from classifying the University of Houston dataset using different methods. (a) Ground truth (train). (b) Ground truth (test). (c) MKL-average. (d) SimpleMKL. (e) Generalize MKL. (f) Multikernel variant of the deep kernel machine optimization. (g) VanillaRNN. (h) One-dimensional-convolutional neural network. (i) Convolutional kernel classifier.

MKL-average, especially in the University of Pavia dataset. It can be said that the performance of the GMKL in the University of Pavia dataset is comparable to that of CKC because it can have higher accuracy than CKC in the case of 1% and 5% training data mode, and this superiority can be justified due to the high trainable parameters of the CKC compared to the GMKL method. As can be seen from Fig. 5, although the M-DKMO and CNN-1-D performed well, in most cases, it had lower accuracy than the proposed method, especially in Berlin and Salinas Valley datasets. The performance of the VanillaRNN method is in no way comparable to the proposed method, but in some cases, such as the 5% of training data of the Berlin dataset, it has performed better than GMKL, SimpleMKL, and MKL-average. In general, In the case of using more than 10% of training data, the performance of the CKC has been better and more accurate than all the methods mentioned earlier. As shown in Fig. 5, all methods reached low OA in the case of the low training dataset. The crux of this matter lies in the fact that one of the main reasons for low classification accuracy in the case of the small dataset is overfitting. Although the kernel dropout can prevent overfitting (see Fig. 4), it is not powerful enough to solve this problem in a small dataset case.

For further analysis, the OA, AA, Kappa coefficient, recall accuracy, and runtime of all the mentioned methods on four datasets are given in Tables VII–X, and their classification maps are presented in Figs. 6–10. It can be observed that the proposed method achieved the highest classification accuracy as compared with other mentioned methods in all HSI datasets, and its classification maps have the sharpest object boundary and the least noise than others.

The MKL-average methods are the fastest runtime, although the consumed time during their parameter search has not been reflected, and the GMKL method is the slowest due to its computational complexity. Due to the Nyström approximation and convolutional architecture of the proposed method, the computational time of CKC is lower than the SimpleMKL, GMKL, and M-DKMO in all datasets.

The CKC achieved the highest OA of 95.22% for the University of Pavia dataset (see Table VII), which exceeds M-DKMO, MKL-average, SimpleMKL, GMKL, VanillaRNN, and CNN-1-D. Our proposed method achieved better recall accuracy than others in most cases, especially in the Gravel class, as shown in Fig. 6. Fig. 6(i) shows the close-up map of the ground truth of the University of Pavia dataset. By

TABLE VII
CLASSIFICATION OF THE UNIVERSITY OF PAVIA DATASET USING 20% OF LABELED DATA

Class	MKL average	Simple MKL	GMKL	M-DKMO	VanillaRNN	CNN-1D	CKC
Alfalfa	95.85	94.81	95.65	95.62	93.29	94.86	**97.73**
Meadows	97.49	97.88	97.98	98.26	96.87	**98.77**	96.66
Gravel	78.32	78.84	80.70	52.35	76.72	59.33	**91.13**
Trees	92.90	94.47	95.76	94.12	87.93	95.13	**98.58**
Painted metal sheets	100.00	99.80	99.44	99.58	99.72	99.91	**100.00**
Bare Soil	88.42	89.32	92.37	93.35	93.56	91.55	**94.98**
Bitumen	79.89	86.56	86.37	89.34	90.99	**91.28**	85.41
Self-Blocking Bricks	86.28	88.23	88.73	93.23	87.32	**95.25**	83.86
Shadows	100.00	99.86	100.00	99.76	100.00	99.87	**100.00**
OA	93.52	94.14	94.89	94.09	93.31	94.64	**95.22**
AA	91.02	92.20	93.00	90.62	91.82	91.77	**94.26**
Kappa	0.9139	0.9221	0.9321	0.9213	0.9131	0.9287	**0.9369**
Time (s.)	**59.02**	287.12	301.59	267.32	194.21	165.42	202.80

The best result in each row is highlighted in bold

TABLE VIII
CLASSIFICATION OF SALINAS VALLEY DATASET USING 20% OF LABELED DATA

Class	MKL average	Simple MKL	GMKL	M-DKMO	VanillaRNN	CNN-1D	CKC
Broccoli-green-weeds_1	99.87	99.93	99.87	99.75	99.72	**100.00**	99.80
Brocoli-green-weeds_2	100.00	99.96	**100.00**	99.97	99.85	99.92	99.95
Fallow	99.66	99.66	99.66	99.23	**99.71**	93.95	99.69
Fallow-rough-plow	99.52	99.52	99.43	99.45	99.67	98.54	**99.85**
Fallow-smooth	98.51	98.71	98.66	99.05	99.11	**99.78**	99.62
Stubble	99.87	99.87	99.76	99.81	99.94	99.93	**100.00**
Celery	100.00	**100.00**	99.93	99.75	99.87	99.96	99.94
Grapes-untrained	90.96	90.61	91.80	95.18	86.31	**97.58**	88.95
Soil-vinyard-develop	99.94	99.96	99.96	98.97	100.00	100.00	**100.00**
Corn-senesced-green-weeds	98.45	**98.74**	98.37	97.90	95.61	98.58	97.76
Lettuce-romaine-4wk	98.75	99.38	99.00	98.33	98.51	**99.86**	98.66
Lettuce-romaine-5wk	100.00	99.93	**100.00**	99.80	**100.00**	**100.00**	99.68
Lettuce-romaine-6wk	99.56	99.71	99.13	**99.72**	98.14	95.86	98.44
Lettuce-romaine-7wk	97.63	98.63	97.14	98.57	97.56	**99.32**	97.71
Vinyard-untrained	62.81	66.57	66.67	68.37	76.96	68.49	**84.25**
Vinyard-vertical-trellis	99.04	98.97	99.04	99.15	98.93	**99.68**	98.31
OA	92.80	93.29	93.47	94.27	93.55	94.80	**95.23**
AA	96.54	96.88	96.78	97.06	96.87	96.96	**97.66**
Kappa	0.9196	0.9251	0.9271	0.9360	0.9281	0.9419	**0.9469**
Time (s.)	**68.32**	335.89	341.64	318.98	228.18	214.64	298.46

The best result in each row is highlighted in bold

comparing Fig. 6(i) and (p), it can be seen that the proposed method can classify the Gravel class better than its counterparts. The processing time of CKC is lower than the SimpleMKL, GMKL, and M-DKMO by 84.32, 98.79, and 64.52 seconds, respectively, and the lower processing time of CKC indicates that the computational complexity of CKC is lower than other MKL methods.

Based on the Salinas Valley dataset (see Table III), the performance of CKC was better than all the other methods and achieved the highest performance. The Vinyard-untrained class was classified with an accuracy of 84.25% using our proposed method, which was 7.29% percent more accurate than the obtained result of the second-best algorithms (i.e., VanillaRNN). It can be observed from Fig. 7 that the CKC correctly labeled almost all classes with similar spectral behavior, such as Vinyard-untrained and Grapes-untrained. Additionally, the computational time of CKC is lower than the other MKL methods based on their processing time given in Table VIII.

The CKC for Berlin dataset (see TableIX) also had the best accuracy, and its OA of 93.94% exceeds MKL-average, SimpleMKL, and GMKL by 5.2%, 3.61%, and 3.51%, respectively. The higher performance of the CKC method can be seen from

TABLE IX
CLASSIFICATION OF BERLIN DATASET USING 20% OF LABELED DATA

Class	MKL average	Simple MKL	GMKL	M-DKMO	VanillaRNN	CNN-1D	CKC
Vegetation	97.97	98.65	98.27	97.74	97.72	98.12	**99.26**
Built-up	86.11	91.13	91.31	93.32	89.37	92.11	**94.32**
Impervious	90.99	89.48	90.30	91.89	88.38	92.58	**94.20**
Soil	57.37	58.00	56.37	64.53	69.26	**81.09**	75.57
Water	89.52	95.40	94.62	97.92	96.74	**97.96**	90.04
OA	88.74	90.33	90.43	92.20	89.98	93.20	**93.94**
AA	84.39	86.53	86.17	89.08	88.29	90.37	**90.68**
Kappa	0.8435	0.8660	0.8673	0.8919	0.8619	0.9065	**0.9162**
Time (s.)	**107.82**	289.91	313.91	337.59	192.01	179.38	203.31

The best result in each row is highlighted in bold

TABLE X
CLASSIFICATION OF THE UNIVERSITY OF HOUSTON DATASET USING AVAILABLE TRAINING DATA

Class	MKL average	Simple MKL	GMKL	M-DKMO	VanillaRNN	CNN-1D	CKC
Healthy Grass	**82.66**	82.07	82.07	81.83	81.95	82.07	78.69
Stressed Grass	84.25	82.73	82.84	83.20	**84.61**	83.43	82.92
Synthetic Grass	99.75	99.75	99.75	99.75	99.75	99.50	**100.00**
Tree	98.93	92.90	98.82	92.78	**99.29**	96.57	93.11
Soil	96.57	98.58	99.05	**99.17**	98.46	97.75	95.84
Water	**98.25**	96.49	97.37	97.37	96.49	**98.25**	97.37
Residential	74.94	75.99	78.44	75.99	84.62	75.52	**85.85**
Commercial	27.79	53.92	59.50	52.26	41.09	**63.30**	50.00
Road	71.90	78.51	75.91	78.87	77.92	75.91	**79.76**
Highway	49.94	61.88	60.92	58.75	57.54	51.51	**88.88**
Railway	66.55	78.77	80.07	77.58	78.88	**88.73**	73.25
Parking lot1	52.70	78.63	80.07	70.83	73.23	89.32	**90.72**
Parking lot2	52.63	68.42	71.05	66.23	66.23	**73.68**	66.08
Tennis court	99.49	**100.00**	100.00	100.00	**100.00**	98.48	98.48
Running track	97.35	96.83	96.56	97.09	98.68	**98.94**	97.88
OA	73.43	80.44	81.70	79.33	79.94	82.37	**83.41**
AA	76.91	83.03	84.16	82.11	82.58	84.86	**85.26**
Kappa	0.7137	0.7892	0.8024	0.7775	0.7833	0.8097	**0.8200**
Time (s.)	**33.21**	209.06	223.43	198.63	112.68	100.01	131.49

The best result in each row is highlighted in bold

TABLE XI
STATISTICAL SIGNIFICANCE FROM THE MCNEMAR'S TEST

Classifiers	CKC			
	U. of Pavia	Salinas Valley	Berlin	U. of Houston
MKL-average	17.42	13.21	32.46	23.79
SimpleMKL	11.65	10.28	27.15	18.54
GMKL	10.87	9.15	25.73	15.39
M-DKMO	8.44	7.63	13.89	9.50
VanillaRNN	9.19	11.03	33.36	16.82
CNN-1D	10.51	8.49	29.82	19.13

the comparison between the obtained classification accuracy of the Built-up and Impervious classes. By comparing the close-up maps in Fig. 8, the CKC can classify the built-up and Impervious classes better than the other methods. As shown in Fig. 8(i)–(p), the classification map of the CKC [see Fig. 8(p)] is more similar to the ground truth map [see Fig. 8(i)] than those obtained using other methods. Thanks to the Nyström approximation method, the processing time of CKC is lower than the

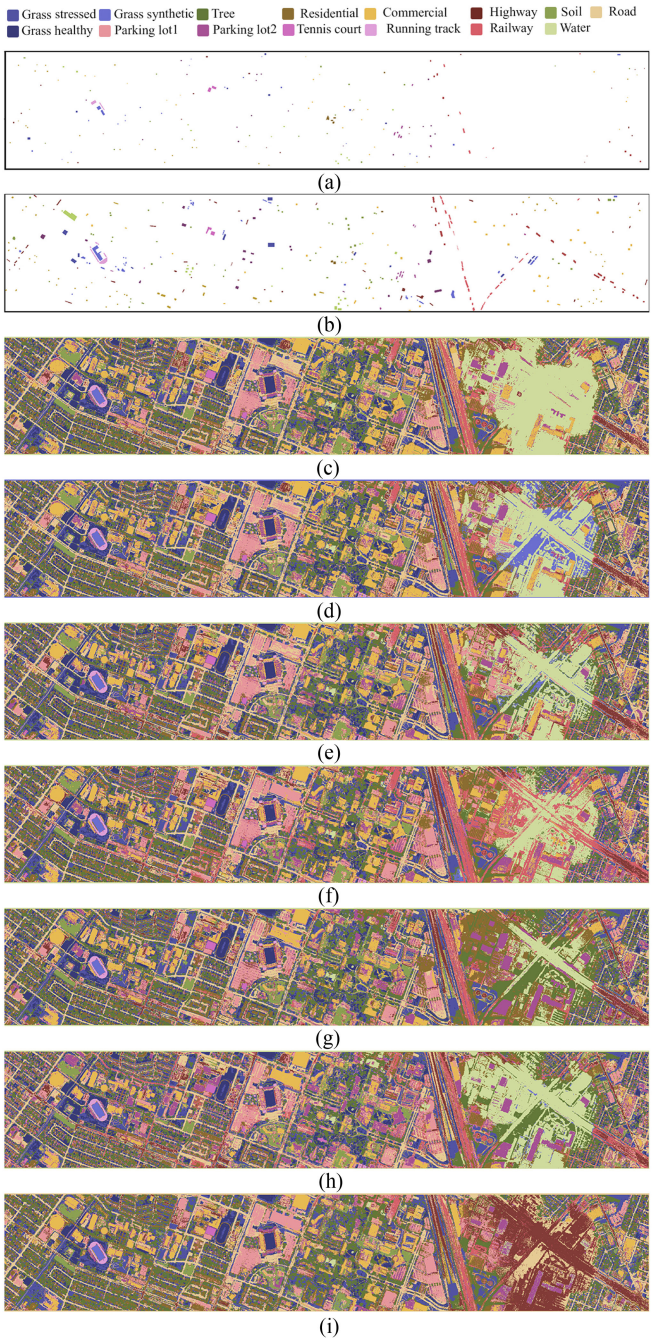

Fig. 9. Ground truth and the classification maps obtained from classifying the University of Houston dataset using different methods. (a) Ground truth (train). (b) Ground truth (test). (c) MKL-average. (d) SimpleMKL. (e) Generalize MKL. (f) Multikernel variant of the deep kernel machine optimization. (g) Vanilla recurrent neural network. (h) One-dimensional-convolutional neural network. (i) Convolutional kernel classifier.

SimpleMKL, GMKL, and M-DKMO by 86.6, 110.6, and 134.28 s, respectively.

The most challenging class of the University of Houston dataset is the highway class since the cloud partially covers it. As we can observe in Table X and Fig. 4, our proposed method in the University of Houston dataset reached better accuracy than other methods and classifies Highway class 27% more accurately than the obtained result of the second-best algorithms

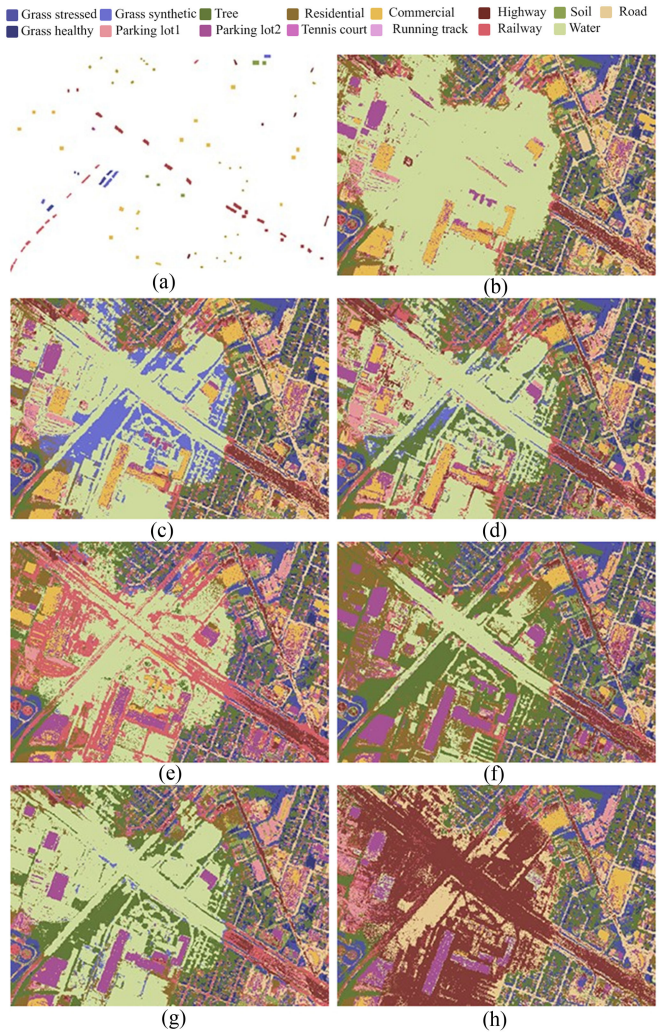

Fig. 10. Close-up ground truth and the classification maps of Highway class of the University of Houston dataset using different methods. (a) Ground truth, (b) MKL-average. (c) SimpleMKL. (d) Generalize MKL. (e) Multikernel variant of the deep kernel machine optimization. (f) Vanilla recurrent neural network. (g) One-dimensional-convolutional neural network. (h) Convolutional kernel classifier.

(i.e., SimpleMKL). As shown in Fig. 10, MKL-average and SimpleMKL could not classify the cloud-covered area in the University of Houston dataset, while M-DKMO, GMKL, and CKC reached better results and showed some spatial structures of the hidden area, for example, Parking lot1 and Parking lot2.

To further evaluate, the nonparametric McNemar's test [49] has been employed. McNemar's test evaluates whether the difference between the obtained accuracy of two classifications (CKC and other MKL or deep learning methods) is statistically different. The absolute values of McNemar's test statistics larger than 2.58 indicate that the two results are statistically different at 99% confidence levels. Table XI gives the values of McNemar's test statistics when the CKC is compared against the other methods. Based on Table XI, the differences between accuracies obtained by the CKC and those obtained by all the other methods were statistically different at a 99% confidence level. The CKC and the MKL-average are statistically very different because

their absolute values of McNemar's test are greater than 13 in all four cases. The absolute values of McNemar's test of CKC compared against SimpleMKL and GMKL are acceptable and

In contrast, the better classification results of CKC compared to the M-DKMO in previous experiments can result from the fact that the features extracted by CNN-1-D are more robust than those extracted by FC architecture. The differences between CKC and CNN-1-D indicates that the kernel representation can extract different feature than CNN-1-D architecture. Therefore, it can be concluded that the combination of multiple kernels and convolutional representation can complement each other to classify the HSI dataset accurately.

To summarize, the CKC outperforms all its counterparts in terms of OA. Moreover, its computational complexity is lower than the other standard MKL methods. The kernel representation and convolutional representation complement each other in the CKC for classifying mentioned benchmark datasets, and they were successful in most cases. However, in the case of using few training samples, some other methods perform better. Although the processing time of VanillaRNN and CNN-1-D is lower than our proposed method, the CKC outperformed these algorithms considering the classification accuracy. As discussed in [4], Although spectral variability in the HSIs reduces the classification accuracy of all HSIs classifiers, the CKC can reduce its effects intrinsically using its statistical and physical normalization. Nevertheless, the remaining of spectral variability can reduce the classification accuracy. In this regard, the CKC can accurately classify most classes of the four mentioned HSI datasets, but some classes cannot reach the highest accuracy.

V. Conclusion

This article introduced a new multiple kernels classifier, namely CKC, using deep kernel methods for land cover mapping from HSI data. The proposed method utilizes the Nyström approximation to alleviate the problems associated with the dimensionality of kernels and employs CNN-1-D to carry out end-to-end learning. As the first instance of the deep kernel method in remote sensing, the CKC inherits the informative representation provided by a multiple kernel learning method and the end-to-end learning capability of deep learning models. We designed three experiments to study the effectiveness of the CKC for HSI data classification using four benchmark datasets.

The first experiment showed how well the loss function value of CKC converges to the low value during both training and validation stages and the ability of kernel dropout to prevent overfitting. The second experiment was designed to evaluate the effects of the different components of the CKC algorithm on its performance. The results from the second experiment showed that using the Nyström approximation reduces the computational cost of the algorithm and can improve classification accuracy and processing time. Furthermore, the results of this experiment confirm the superiority of CNN-1-D over the FC method in the CKC algorithm. The third experiment showed that the CKC algorithm, as an MKL framework, outperformed the most common MKL algorithms in the literature in both OA and processing time. In addition, the CKC performance was better than its state-of-the-art deep learning model counterparts. The CKC's success can substantiate the claim that the deep kernel method can be considered a proper method in remote sensing and an alternative to building multiple kernel learning approaches. In future studies, we plan to extend CKC algorithms' capability by substituting its 1-D convolutional layer with 2-D and 3-D layers, which enable the algorithm to learn spatial-spectral features.

Acknowledgment

The authors would like to acknowledge the IEEE Geoscience and Remote Sensing Society (GRSS) Image Analysis and Data Fusion Technical Committee during the 2013 Data Fusion Contest (DFC) for preparing the University of Houston dataset [44].

References

[1] S. Li, W. Song, L. Fang, Y. Chen, P. Ghamisi, and J. A. Benediktsson, "Deep learning for hyperspectral image classification: An overview," *IEEE Trans. Geosci. Remote Sens.,* vol. 57, no. 9, pp. 6690–6709, Sep. 2019, doi: 10.1109/TGRS.2019.2907932.

[2] D. Hong, N. Yokoya, J. Xu, and X. Zhu, "Joint & progressive learning from high-dimensional data for multi-label classification," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 469–484.

[3] D. Hong and X. X. Zhu, "SULoRA: Subspace unmixing with low-rank attribute embedding for hyperspectral data analysis," *IEEE J. Sel. Topics Signal Process.,* vol. 12, no. 6, pp. 1351–1363, Dec. 2018, doi: 10.1109/JSTSP.2018.2877497.

[4] D. Hong, N. Yokoya, J. Chanussot, and X. X. Zhu, "An augmented linear mixing model to address spectral variability for hyperspectral unmixing," *IEEE Trans. Image Process.,* vol. 28, no. 4, pp. 1923–1938, Apr. 2018, doi: 10.1109/TIP.2018.2878958.

[5] D. Hong, N. Yokoya, and X. X. Zhu, "Local manifold learning with robust neighbors selection for hyperspectral dimensionality reduction," in *Proc. IEEE Int. Geosci. Remote Sens. Symp.*, 2016, pp. 40–43, doi: 10.1109/IGARSS.2016.7729001.

[6] D. Hong, N. Yokoya, and X. X. Zhu, "Learning a robust local manifold representation for hyperspectral dimensionality reduction," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.,* vol. 10, no. 6, pp. 2960–2975, Jun. 2017, doi: 10.1109/JSTARS.2017.2682189.

[7] D. Hong, W. Liu, J. Su, Z. Pan, and G. Wang, "A novel hierarchical approach for multispectral palmprint recognition," *Neurocomputing,* vol. 151, no. 1, pp. 511–521, 2015, doi: 10.1016/j.neucom.2014.09.013.

[8] W. Sun and Q. Du, "Hyperspectral band selection: A review," *IEEE Geosci. Remote Sens. Mag.*, vol. 7, no. 2, pp. 118–139, Jun. 2019.

[9] J. Peng *et al.*, "Low-Rank and sparse representation for hyperspectral image processing: A review," *IEEE Geosci. Remote Sens. Mag.,* to be published, doi: 10.1109/MGRS.2021.3075491.

[10] D. Hong *et al.*, "More diverse means better: Multimodal deep learning meets remote-sensing imagery classification," *IEEE Trans. Geosci. Remote Sens.,* vol. 59, no. 5, pp. 4340–4354, May 2021, doi: 10.1109/TGRS.2020.3016820.

[11] D. Hong, L. Gao, J. Yao, B. Zhang, A. Plaza, and J. Chanussot, "Graph convolutional networks for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.,* vol. 59, no. 7, pp. 5966–5978, Jul. 2021, doi: 10.1109/TGRS.2020.3015157.

[12] D. Hong, J. Hu, J. Yao, J. Chanussot, and X. X. Zhu, "Multimodal remote sensing benchmark datasets for land cover classification with a shared and specific feature learning model," *ISPRS J. Photogramm. Remote Sens.,* vol. 178, no. Sep./Oct., pp. 68–80, 2021, doi: 10.1016/j.isprsjprs.2021.05.011.

[13] D. Hong *et al.*, "Interpretable hyperspectral AI: When non-convex modeling meets hyperspectral remote sensing," *IEEE Geosci. Remote Sens. Mag.,* vol. 9, no. 2, pp. 52–87, Apr. 2021, doi: 10.1109/MGRS.2021.3064051.

[14] H. Song, J. J. Thiagarajan, P. Sattigeri, and A. Spanias, "Optimizing kernel machines using deep learning," *IEEE Trans. Neural Netw. Learn. Syst.,* vol. 29, no. 11, pp. 5528–5540, Mar. 2018, doi: 10.1109/TNNLS.2018.2804895.

[15] M. Paoletti, J. Haut, J. Plaza, and A. Plaza, "Deep learning classifiers for hyperspectral imaging: A review," *ISPRS J. Photogramm. Remote Sens.,* vol. 158, pp. 279–317, Dec. 2019, doi: 10.1016/j.isprsjprs.2019.09.006.

[16] X.-L. Li, "Preconditioned stochastic gradient descent," *IEEE Trans. Neural Netw. Learn. Syst.,* vol. 29, no. 5, pp. 1454–1466, May 2017, doi: 10.1109/TNNLS.2017.2672978.
[17] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.,* vol. 15, no. 1, pp. 1929–1958, Jun. 2014.
[18] D. Hong *et al.*, "Endmember-Guided unmixing network (EGU-Net): A general deep learning framework for self-supervised hyperspectral unmixing," *IEEE Trans. Neural Netw. Learn. Syst.,* to be published, doi: 10.1109/TNNLS.2021.3082289.
[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Commun. ACM,* vol. 60, no. 6, pp. 84–90, 2017, doi: org/10.1145/3065386.
[20] S. Niazmardi, B. Demir, L. Bruzzone, A. Safari, and S. Homayouni, "Multiple kernel learning for remote sensing image classification," *IEEE Trans. Geosci. Remote Sens.,* vol. 56, no. 3, pp. 1425–1443, Mar. 2018, doi: 10.1109/TGRS.2017.2762597.
[21] T. Wang, L. Zhang, and W. Hu, "Bridging deep and multiple kernel learning: A review," *Inf. Fusion,* vol. 67, pp. 3–13, Mar. 2020.
[22] A. Jain, S. V. Vishwanathan, and M. Varma, "SPF-GMKL: Generalized multiple kernel learning with a million kernels," in *Proc. 18th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2012, pp. 750–758.
[23] S. Sonnenburg, G. Rätsch, C. Schäfer, and B. Schölkopf, "Large scale multiple kernel learning," *J. Mach. Learn. Res.,* vol. 7, pp. 1531–1565, Jul. 2006.
[24] F. R. Bach, G. R. Lanckriet, and M. I. Jordan, "Multiple kernel learning, conic duality, and the SMO algorithm," in *Proc. 21st Int. Conf. Mach. Learn.*, 2004.
[25] M. Alioscha-Perez, M. C. Oveneke, and H. Sahli, "Svrg-mkl: A fast and scalable multiple kernel learning solution for features combination in multi-class classification problems," *IEEE Trans. neural Netw. Learn. Syst.,* vol. 31, no. 5, pp. 1710–1723, May 2020, doi: 10.1109/TNNLS.2019.2922123.
[26] L. A. Belanche Muñoz and M. R. Costa-Jussà, "Bridging deep and kernel methods," in *Proc. 25th Eur. Symp. Artif. Neural Netw.*, 2017, pp. 1–10.
[27] I. Lauriola, C. Gallicchio, and F. Aiolli, "Enhancing deep neural networks via multiple kernel learning," *Pattern Recognit.,* vol. 101, May 2020, Art. no. 107194.
[28] Y. Cho and L. Saul, "Kernel methods for deep learning," *Adv. Neural Inf. Process. Syst.,* vol. 22, pp. 342–350, 2009.
[29] A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing, "Deep kernel learning," in *Proc. 19th Int. Conf. Artif. Intell. Statist.*, 2016, pp. 370–378.
[30] A. Afzal and S. Asharaf, "Deep multiple multilayer kernel learning in core vector machines," *Expert Syst. Appl.,* vol. 96, pp. 149–156, Apr. 2018, doi: 10.1016/j.eswa.2017.11.058Get.
[31] A. Afzal and S. Asharaf, "Deep kernel learning in core vector machines," *Pattern Anal. Appl.,* vol. 21, no. 3, pp. 721–729, Feb. 2018.
[32] B. Sheng, J. Li, F. Xiao, and W. Yang, "Multilayer deep features with multiple kernel learning for action recognition," *Neurocomputing,* vol. 399, no. 3, pp. 65–74, Jul. 2020.
[33] S. Agethen and W. H. Hsu, "Deep multi-kernel convolutional LSTM networks and an attention-based mechanism for videos," *IEEE Trans. Multimedia,* vol. 22, no. 3, pp. 819–829, Mar. 2020.
[34] S. Ren, W. Shen, C. N. Siddique, and Y. Li, "Self-Adaptive deep multiple kernel learning based on rademacher complexity," *Symmetry,* vol. 11, no. 3, 2019.
[35] T. Wang, H. Su, and J. Li, "DWS-MKL: Depth-width-scaling multiple kernel learning for data classification," *Neurocomputing,* vol. 411, no. 18, pp. 455–467, 2020, doi: 10.1016/j.neucom.2020.06.039.
[36] M. Jiu and H. Sahbi, "Deep representation design from deep kernel networks," *Pattern Recognit.,* vol. 88, pp. 447–457, 2019.
[37] X. Zhang *et al.*, "Emotion recognition from multimodal physiological signals using a regularized deep fusion of kernel machine," *IEEE Trans. Cybern.*, vol. 51, no. 9, pp. 4386–4399, Sep. 2021, doi: 10.1109/TCYB.2020.2987575.
[38] P. Drineas and M. W. Mahoney, "On the Nyström method for approximating a gram matrix for improved kernel-based learning," *J. Mach. Learn. Res.,* vol. 6, pp. 2153–2175, Dec. 2005.
[39] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," *Adv. Neural Inf. Process. Syst.,* vol. 3, no. 4, pp. 1177–1184, 2008.
[40] M. Li, W. Bi, J. T. Kwok, and B.-L. Lu, "Large-scale nyström kernel matrix approximation using randomized SVD," *IEEE Trans. Neural Netw. Learn. Syst.,* vol. 26, no. 1, pp. 152–164, Jan. 2015, doi: 10.1109/TNNLS.2014.2359798.
[41] S. Sun, J. Zhao, and J. Zhu, "A review of nyström methods for large-scale machine learning," *Inf. Fusion,* vol. 26, pp. 36–48, Nov. 2015.
[42] Y. Guo, Y. Liu, A. Oerlemans, S. Lao, S. Wu, and M. S. Lew, "Deep learning for visual understanding: A review," *Neurocomputing,* vol. 187, pp. 27–48, 2016.
[43] Y. Li, W. Xie, and H. Li, "Hyperspectral image reconstruction by deep convolutional neural network for classification," *Pattern Recognit.,* vol. 63, pp. 371–383, 2017.
[44] C. Debes *et al.*, "Hyperspectral and LiDAR data fusion: Outcome of the 2013 GRSS data fusion contest," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.,* vol. 7, no. 6, pp. 2405–2418, Jun. 2014, doi: 10.1109/JSTARS.2014.2305441.
[45] A. Rakotomamonjy, F. R. Bach, S. Canu, and Y. Grandvalet, "SimpleMKL," *J. Mach. Learn. Res.,* vol. 9, pp. 2491–2521, 2008.
[46] Z. Sun, N. Ampornpunt, M. Varma, and S. Vishwanathan, "Multiple kernel learning and the SMO algorithm," in *Proc. Adv. Neural Inf. Process. Syst.*, 2010, pp. 2361–2369.
[47] C. Cortes, M. Mohri, and A. Rostamizadeh, "Learning non-linear combinations of kernels," in *Proc. Adv. Neural Inf. Process. Syst.*, 2009, pp. 396–404.
[48] K. Doya, "Universality of fully connected recurrent neural networks," Dept. Biol., Univ. California San Diego, San Diego, CA, USA, 1993.
[49] A. Villa, J. A. Benediktsson, J. Chanussot, and C. Jutten, "Hyperspectral image classification with independent component discriminant analysis," *IEEE Trans. Geosci. remote Sens.,* vol. 49, no. 12, pp. 4865–4876, Dec. 2011, doi: 10.1109/TGRS.2011.2153861.

Mohsen Ansari received the B.Sc. degree in surveying and geomatics engineering and the M.Sc. degree in remote sensing from the School of Surveying and Geospatial Engineering, College of Engineering, University of Tehran, Tehran, Iran, in 2018 and 2021, respectively.

He has worked on various remote sensing applications, including but not limited to land cover/land use classification, water resource management, water salinity, and change detection. His research interests include analyzing different remote sensing datasets through AI and ML approaches for urban and agro-environmental applications.

Saeid Homayouni (Senior Member, IEEE) received the B.Sc. degree in surveying and geomatics engineering from the University of Isfahan, Isfahan, Iran, in 1996, the M.Sc. degree in remote sensing and geographic information systems from Tarbiat Modares University, Tehran, Iran, in 1999, and the Ph.D. degree in signal and image from Télécom Paris Tech, Paris, France, in 2005.

From 2006 to 2007, he was a Postdoctoral Fellow with the Signal and Image Laboratory, University of Bordeaux Agro-Science, Bordeaux, France. From 2008 to 2011, he was an Assistant Professor with the Department of Surveying and Geomatics, College of Engineering, University of Tehran, Tehran, Iran. From 2011 to 2013, through the Natural Sciences and Engineering Research Council of Canada Visitor Fellowship Program, he worked with the Earth Observation Group of the Agriculture and Agri-Food Canada, Ottawa Center of Research and Development, Ottawa, ON, Canada. In 2013, he was a Replacing Assistant Professor of remote sensing and geographic information systems with the Department of Geography, Environment, and Geomatics, University of Ottawa, Ottawa, ON, Canada. Since April 2019, he has been an Associate Professor of environmental remote sensing and geomatics with the Centre Eau Terre Environnement, Institut National de la Recherche Scientifique, Quebec, QC, Canada. He is currently leading a research group on Earth Observation Analytics by Artificial Intelligence with interests in optical and radar earth observations analytics for urban and agroenvironmental applications.

Abdolreza Safari received the B.S. degree in surveying and geomatics engineering, and the M.Sc. and Ph.D. degrees in geodesy from the School of Surveying and Geospatial Engineering, College of Engineering, University of Tehran, Tehran, Iran, in 1993, 1998, and 2004, respectively.

He is currently a Professor with the School of Surveying and Geospatial Engineering, College of Engineering, University of Tehran. His research interests include the mathematical modeling of remote sensing and geodetic data.

Dr. Safari is a Member of the Center of Excellence for Surveying Engineering in Natural Disaster Management, College of Engineering, University of Tehran.

Saeid Niazmardi (Student Member, IEEE) received the B.S. degree in surveying and geomatics engineering from the University of Isfahan, Isfahan, Iran, in 2009, and the M.Sc. and Ph.D. degrees in remote sensing from the University of Tehran, Tehran, Iran, in 2009 and 2016, respectively.

He is currently an Assistant Professor with the Department of Surveying and Geomatics Engineering, Faculty of Civil and Surveying Engineering, Graduate University of Advanced Technology, Kerman, Iran. His research interests include machine learning using remotely sensed data.