Hyperspectral Image Classification Using CapsNet With Well-Initialized Shallow Layers

Jihao Yin, *Senior Member, IEEE*, Sen Li, Hongmei Zhu, and Xiaoyan Luo, *Member, IEEE*

***Abstract*— In this letter, an alternative data-driven HSI classification model based on CapsNet is proposed rather than recently predominant convolutional neural network (CNN)-based models. To adjust the CapsNet to HSI classification, we tune a new CapsNet architecture with three convolutional layers. The added shallow layer provides higher level features to the primary capsules, which indirectly speeds up the following routing procedure. To guarantee a good convergence of the whole CapsNet, the three shallow layers are initialized by transferring convolutional parameters from a pretrained CNN model. The improved CapsNet-based models with and without vote strategy both achieve significantly superior performance in HSI classification to the state-of-the-art CNN-based methods on real hyperspectral data sets.**

***Index Terms*— Capsule, convolutional neural network (CNN), hyperspectral image (HSI) classification, transfer learning.**

I. Introduction

HYPERSPECTRAL image (HSI) classification is an increasingly hot topic since HSI enables to differentiate land cover via recognizing the specific spectral information, which is applied to many important fields [1], such as agriculture management, military surveillance, environmental governance, and so on. However, the performance of HSI classifiers is restricted by three properties of hyperspectral data: the curse of high dimension, the limited number of labeled samples, and huge spatial variability of the spectrum [2]. To solve this problem, the existing methods are roughly divided into two categories: handcrafted feature-based methods and data-driven feature-based methods.

The handcrafted feature-based methods have occupied the hyperspectral classification field for ages, which designed the feature extractors and classifiers separately. For example, Bandos *et al.* [3] proposed an effective feature extractor called regularized linear discriminant analysis to transform the data subspace into a linear separable subspace. Bruce *et al.* [4] utilized the discrete wavelet transform to extract frequency-domain features for higher classification accuracy of HSI. Jia *et al.* [5] proposed a collaborative representation based on a Gabor feature to improve the discrimination power of material features. Instead of designing feature extractor, Melgani *et al.* [6] compared the performance of both linear and nonlinear support vector machines (SVMs) classifiers for HSI classification. Delalieux *et al.* [7] combined the hyperspectral mixture analysis with decision tree classifier for heathland conservation status mapping. To improve the generalization of the classifier, Ham *et al.* [8] proposed a random forest binary hierarchical classification method which incorporated bagging of training samples and adaptive random subspace feature selection within a binary hierarchical classifier. It can be observed that the interaction between feature extractors and classifiers is seldom considered during their construction phase in these methods.

Compared with the aforementioned methods, the data-driven feature-based methods extract features via learning, typically including classification methods based on sparse representation (SR) and convolutional neural networks (CNN). Chen *et al.* [9] initially proposed a joint sparse model (JSM) to express the hyperspectral pixel with both spatial constraint and sparse constraint. Gao *et al.* [10] introduced the discontinuity-preserving relaxation into JSM to weigh the contribution from each neighborhood pixel, which achieved better classification accuracy around class boundaries. It is noteworthy that SR-based classification methods have no explicit classifiers, which decide the class labels by solving a recovery optimization problem. In contrast, CNN-based methods usually train the feature extractor and the classifier end to end. Chen *et al.* [2] proposed a 3-D CNN-based feature extraction model with a regularization constraint to extract effective spectral-spatial features of HSI, which initially applied CNN to HSI classification. Lee *et al.* [11] presented a contextual deep CNN which integrated the multiscale kernels like GoogLeNet [12] and skip architectures like ResNet [13], so that their networks went deeper and wider. Li *et al.* [14] employed fully CNNs to enhance the deep features of HSI, but it is followed by an extreme learning machine (ELM) for classification rather than end-to-end training. In addition, data augmentation [15] and multisource data fusion [16] are also considered in recent work.

However, Sabour *et al.* [17] pointed out that the CNN model is less effective at exploring the spatial relationships among learned instantiation parameters, such as perspective, size, and orientation. They proposed a novel network architecture called "CapsNet," which is demonstrated to achieve better performance at recognizing highly overlapping digits. Thus, we assume that it has the potential to improve HSI

Manuscript received July 17, 2018; revised October 6, 2018 and November 19, 2018; accepted December 29, 2018. Date of publication January 29, 2019; date of current version June 24, 2019. This work was supported in part by the National Natural Science Foundation of China under Grant 41471278 and Grant 41871240 and in part by the Cooperative Agreement between the National Natural Science Foundation of China and China Civil Aviation Administration through Joint Project Funded in Civil under Grant U1833117. *(Corresponding author: Xiaoyan Luo.)*

The authors are with the School of Astronautics, Beihang University, Beijing 100191, China (e-mail: yjh@buaa.edu.cn; buaals@buaa.edu.cn; z_hongmei@buaa.edu.cn; luoxy@buaa.edu.cn).

Color versions of one or more of the figures in this letter are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/LGRS.2019.2891076

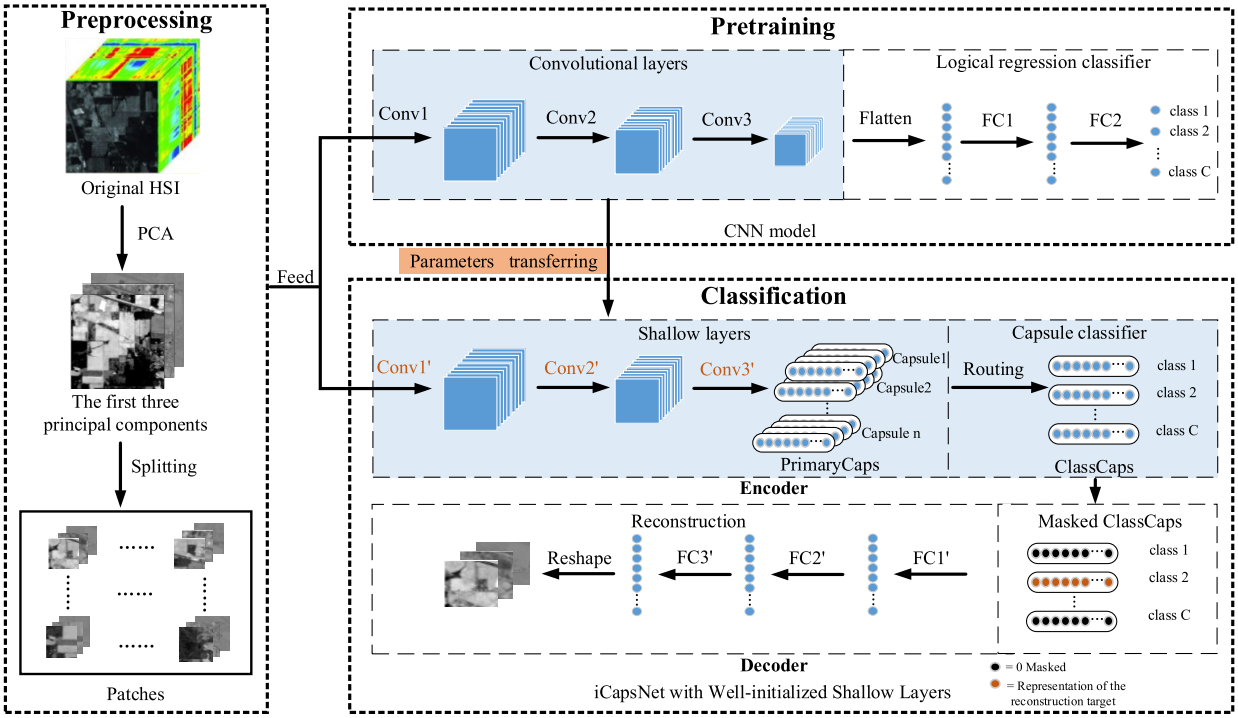


Fig. 1. Workflow of the proposed method including three parts, i.e., preprocessing, pretraining, and classification.

classification accuracy of mixed pixels around class boundaries as well and propose a novel HSI classification method based on CapsNet.

Similar to [18], we find that applying original CapsNet to HSI classification and training it from scratch does not work well. Thus, we improve the network architecture and present a reasonable initialization strategy to enhance its performance. Specifically, there are three shallow layers in the proposed network instead of two shallow layers in the original network. In addition, the shallow layers are initialized via transferring the convolutional parameters from a pretrained CNN model. The new architecture and the elaborated initialization strategy make the improved CapsNet (iCapsNet) much more suitable to HSI classification and outperform the state-of-the-art methods.

II. Methodology

This section describes the proposed method in detail. As shown in Fig. 1, the workflow mainly includes three parts as follows.

1) *Preprocessing Part:* To reduce the spectrum redundancy, the original HSI is compressed into three channels via the principal component analysis (PCA) strategy, and the three PCA components are further split into patches labeled according to their respective center pixels.
2) *Pretraining Part:* To facilitate the convergence of the proposed network, we pretrain a CNN model at first and transfer its convolutional parameters to the shallow layers of the CapsNet.
3) *iCapsNet Classification:* Encoder—iCapsNet with well-initialized shallow layers is trained for HSI classification; Decoder—to avoid overfitting, the reconstruction loss is added to regularize the encoder training process.

In the following, we review the original CapseNt initially and then present iCapsNet.

A. Original CapsNet

Recently, Sabour *et al.* [17] presented a new type of network computation unit called capsule, which is the core unit of CapsNet. By comparison, the similarity and the difference between CapsNet and CNNs can be briefly summarized. As with CNNs, CapsNet replicates learned knowledge across space and makes higher level layers cover larger regions of the image. The difference is that CapsNet replaces the scalar-output feature detectors of CNNs with vector-output capsules and max-pooling with routing-by-agreement. It enables CapsNet to capture the spatial relationship of instantiation parameters of a specific type of entity and enhances interact strength between higher level and lower level layers. In the following, we will review how the vector inputs and outputs of a capsule are computed [17].

Given a capsule j, its vector output is obtained via squashing as

$$\boldsymbol{v}_j = \frac{\|\boldsymbol{s}_j\|^2}{1+\|\boldsymbol{s}_j\|^2}\frac{\boldsymbol{s}_j}{\|\boldsymbol{s}_j\|} \tag{1}$$

where $\boldsymbol{s}_j$ is the total input defined as

$$\boldsymbol{s}_j \sum_i C_{ij}\hat{\boldsymbol{u}}_{j|i}, \quad \hat{\boldsymbol{u}}_{j|i} = \mathbf{W}_{ij}\boldsymbol{u}_i. \tag{2}$$

Here, $\hat{\boldsymbol{u}}_{j|1}$ are prediction vectors from the capsule i to capsule j, and $\mathbf{W}_{ij}$ are affine transformation matrixes which can model the spatial relationship of the instantiation parameters in $\boldsymbol{u}_i$. are the coupling coefficients calculated as

$$c_{ij} = \frac{\exp(b_{ij})}{\sum_k \exp(b_{ik})} \tag{3}$$

where b_{ij} are initialized to zeros so that capsule i is coupled to the capsule j with equal probabilities. That is to say,

the computation of a capsule includes four operations: softmax operation on prior b_{ij}, affine transformation on the output $\boldsymbol{u}_i$ of the capsule i, weighting sum of the prediction vectors $\hat{\boldsymbol{u}}_{j|i}$, and nonlinear squashing. With iterations, the training process called routing-by-agreement repetitively updates b_{ij} as

$$b_{ij} \longleftarrow b_{ij} + \hat{\boldsymbol{u}}_{j|i} \cdot \boldsymbol{v}_j \tag{4}$$

and then successively executes formula (3), (2), and (1).

The original CapsNet has an autoencoder structure. The encoder consists of one convolutional layer, a PrimaryCaps layer, and a DigitCaps layer. The length of the encoder's output vector indicates the existence of a specific digit. For regularization, Sabour *et al.* [17] employed the idea of autoencoder that minimizing the reconstruction loss to facilitate encoder ability. Three fully connected layers constitute the decoder so as to model the pixel intensities of the input image.

The total loss function of original CapsNet is a weighted sum of margin loss and reconstruction loss, which can be expressed as

$$L_{\text{total}} = \sum_k L_k + \lambda_0 L_{\text{res}}. \tag{5}$$

where λ_0 is a weighting factor given by the user. The margin loss of capsule k is defined as

$$L_k = T_k \max(0, m^+ - \|\boldsymbol{v_k}\|)^2 + \lambda_1 (1 - T_k) \max(0, \|\boldsymbol{v_k}\| - m^-)^2 \tag{6}$$

where $T_k = 1$ if a class k is present and $m^+ = 0.9$, $m^- = 0.9$. The down-weighting parameter λ_1 is set to 0.5 in this letter. The reconstruction loss of input image $\boldsymbol{x}$ is computed as

$$L_{\text{res}} = \|\boldsymbol{x} - \boldsymbol{x}'\|_2 \tag{7}$$

where $\boldsymbol{x}'$ is the reconstructed image by the decoder. For more details, refer to the literature [17].

B. CapsNet With Well-Initialized Shallow Layers

We train the original CapsNet from scratch on HSI. However, the performance is unsatisfactory in the following aspects: 1) the training loss easily falls into a local minimum staying at a high level; 2) the training accuracy is lower than that of some classic HSI classification methods, such as SVMs, and SR classification (SRC); and 3) the number of capsules in PrimaryCaps is too large to send their outputs to ClassCaps layer, which results in slightly time-consuming routing. Considering the above problems, we propose a new structure of CapsNet shown in Table I.

From the perspective of structure, iCapsNet is only added an additional convolutional layer before PrimaryCaps layer. Nonetheless, this simple idea can indirectly decrease the number of primary capsules and speed up the routing procedure. The main difference compared with the original CapsNet is the initializing strategy of the shallow layers, i.e., Conv1′, Conv2′, and PrimaryCaps (Conv3′) layers, which is proven to be dramatically enhanced the performance of the CapsNet for HSI classification in the experimental part.

Specifically, motivated by transfer learning, we pretrain a CNN model which consists of the three convolutional layers and a logistic regression classifier (LR), where the settings of convolutional parameters are identical to that of the shallow layers in iCapsNet. What we need to know is that the implementation of PrimaryCaps layer is a group of convolution operations in practice, namely, each capsule of PrimaryCaps layer is composed of eight convolutional units with 9×9 kernel. Thus, the convolutional parameters of the pretrained CNN model can be transferred to the corresponding shallow layers, which provide a good initialization to prevent the training from falling into local minimum. The rest of layers in iCapsNet are randomly initialized since they have different parameter size from the fully connected layers in pretrained CNN model. Given the iterations of routing and the maximum iterations of the whole model, all the parameters are updated via Adam optimizer with the hyperparameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 10^{-8}$, and the learning rate $lr = 0.001$. Only the encoder layers are utilized to carry out HSI classification. For fair comparison, the parameters (λ_0, m^+, m^-, λ_1) in loss function, Adam optimizer (β_1, β_2, ε, lr), and the final configuration (the kernel size, stride, channels of Conv1′ and PrimaryCaps, the dimension of the capsules in ClassCaps, and the size of FC1, FC2, and FC3) are with reference to the original CapsNet in [17]. Via trial-and-error analysis, the parameters of Conv2′ are selected at kernel size tuning from 3×3, 5×5, 7×7, and 9×9, and channels tuning from 128, 256, and 512. Finally, we select 7×7 and 128 for the kernel size and channels, respectively, of Conv2′ considering the best accuracy and the least time-consuming routing.

To further improve the classification accuracy, the vote strategy is selected to smoothen the resulted classification maps. The classification accuracies before and after voting are compared, which demonstrates its effectiveness.

TABLE I
CONFIGURATION OF THE iCAPSNET

	layer	parameter size/stride	output shape
input	-	-	(128, 27, 27, 3)
encoder	Conv1'	9×9 / 1	(128, 19, 19, 256)
	Conv2'	7×7 / 1	(128, 13, 13, 128)
	PrimaryCaps	9×9×8 / 2	(128, 288, 8)
	ClassCaps	128×288×8+ 8×16×C×128	(128, C, 16)
decoder	Fc1	73728+512/-	(128, 512)
	Fc2	524288+1024/-	(128, 1024)
	Fc3	2239488+2187/-	(128, 2187)
	Reshape	-	(128,27,27,3)

C: the total number of classes.

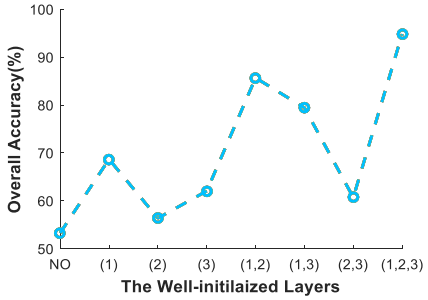
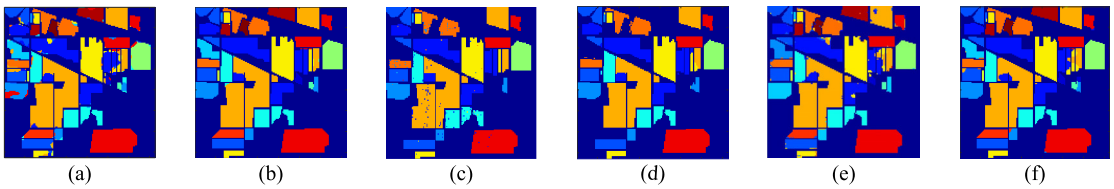


Fig. 2. OA curve of different transferring cases.



(a) (b) (c) (d) (e) (f)

Fig. 3. Classification maps obtained by using different classification methods for Indian Pines data set. (a) 2D-CNN-LR [2] (OA = 89.99%). (b) iCapsNet-vote[1] (OA = 99.23%). (c) CNN-PPF [15] (OA = 94.34%). (d) iCapsNet-vote[2] (OA = 98.60%). (e) FCN-ELM [14] (OA = 96.70%). (f) iCapsNet-vote[3] (OA = 96.88%).

TABLE II

CLASSIFICATION ACCURACY (%) BY USING DIFFERENT CLASSIFICATION METHODS ON INDIAN PINES DATA SET

Class	SVM	SRC	CNN	CapsNet	iCapsNet	iCapsNet -vote	2D-CNN-LR [2]	iCapsNet -vote[1]	CNN-PPF [15]	iCapsNet -vote[2]	FCN-ELM [14]	iCapsNet -vote[3]
1	77.78	75.61	95.65	0	98.08	100.00	99.65	96.36	-	-	92.50	100
2	81.00	67.06	84.86	40.34	95.43	96.95	90.64	99.72	92.99	98.42	97.03	95.52
3	80.69	60.04	96.24	59.25	91.86	96.22	99.11	98.46	96.66	99.16	97.67	94.31
4	74.81	47.24	94.91	87.69	95.90	100.00	100.00	100.00	-	-	87.32	98.73
5	93.64	89.54	97.92	74.32	99.77	99.11	98.48	99.79	98.58	96.47	98.20	93.9
6	95.19	84.31	84.61	66.87	94.44	95.62	97.95	98.29	100.00	99.07	98.51	98.41
7	80.65	76.00	94.12	88.89	100.00	100.00	100.00	92.86	-	-	100.00	100.00
8	98.34	97.09	97.03	89.75	98.79	100.00	100.00	100.00	100.00	99.39	99.31	100.00
9	70.37	23.26	48.78	31.25	47.62	74.07	100.00	100.00	-	-	50.03	90.00
10	81.11	65.01	86.20	45.15	92.21	96.48	95.33	98.26	96.24	97.35	96.04	98.42
11	79.61	77.33	79.40	54.18	95.20	96.30	78.21	99.59	87.8	99.92	96.23	96.56
12	85.99	80.38	89.13	20.00	96.75	98.56	99.39	99.03	98.98	97.61	95.54	97.76
13	93.78	84.00	86.92	68.94	100.00	100.00	100.00	100.00	-	-	99.57	100.00
14	92.13	91.85	96.54	78.20	99.15	99.31	97.71	99.16	99.81	100.00	97.94	96.23
15	78.07	56.16	90.56	75.80	98.93	100.00	99.31	100.00	-	-	94.60	100.00
16	100.00	98.88	100.00	0	94.74	100.00	99.22	98.92	-	-	97.61	100.00
OA	84.98	75.86	87.34	59.43	95.66	97.45	89.99	99.23	94.34	98.60	96.70	96.88

III. EXPERIMENTAL RESULTS

A. Experimental Data and Implementation Details

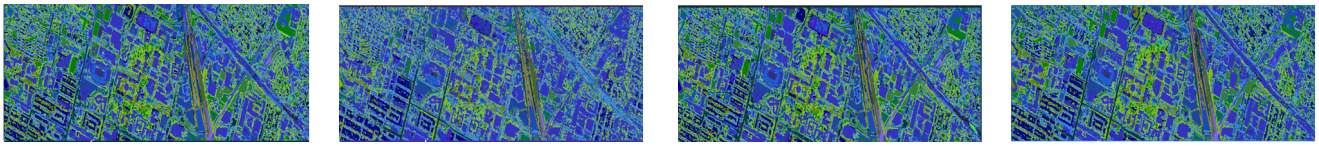
To demonstrate the performance of the iCapsNet, we evaluate it on a typical HSI data set, i.e., Indian Pines. Indian Pines data set with an image size 145×145 was obtained by AirborneVisible/Infrared Imaging Spetrometer sensor with a 20-m spatial resolution. It has 200 bands after removing the 24 water absorption bands covering 16 classes land features. It is true that future research studies should pay attention to more challenge data set such as the large data set IGARSS 2018 [20], which is acquired by the National Center for Airborne Laser Mapping, covering a 380–1050-nm spectral range with 48 bands at a 1-m ground sample distance (GSD) and 20 classes land cover. We train our model on the training data set with an image size of 601×2384 which is provided as raster at a 0.5-m GSD, superimposable to airborne images. Then, we test on the test data set with an image size of 1202×4172.

The first three components of original HSI are treated as pseudocolor image which is split into $27 \times 27 \times 3$ patches as the input of our network. The routing iterations are set to 3, and the maximum epoch of the whole network is set to 100. The initial transformation matrix $27 \times 27 \times 3$ is set randomly by Gaussian distribution $N(0, 0.001^2)$. Considering the effectiveness and efficiency, the size of the voting matrix is tuned from 3×3, 5×5, 7×7, and 9×9 for two data sets. Finally, 5×5 for Indian Pines and 7×7 for IGARSS 2018 data set achieve the best accuracy. All the implementations based on Keras with TensorFlow backend are performed on a desktop with one NVIDIA GTX1080Ti GPU.

To find the most appropriate network architecture, we actually tune for different transferring cases in small data set, i.e., Indian Pines data set, whose performance comparison is shown in Fig. 2. It can be seen that the overall accuracy (OA) indexes achieve the best performance in the case that all the shallow layers are well initialized. Furthermore, the OA is much greater than that of the case without any well-designed initialization layer, namely, the original CapsNet. Thus, the final iCapsNet is well initialized with all the three shallow layers.

B. Classification Results

For Indian pines data set, iCapsNet and iCapsNet with voting (iCapsNet-vote) are compared with three existing traditional HSI classification methods (SVM, SRC, and CNN) and three representative works (2D-CNN-LR [2], CNN-pixel-pair features (PPF) [15], and fully convolutional network (FCN)-ELM [14]) based on deep learning published in recent 2 years for comparison. As different experimental parameters setting in their works, the quantitative results (2D-CNN-LR [2], CNN-PPF [15], and FCN-ELM [14]) are directly obtained from their papers, respectively. For a fair comparison, we train and test original CapsNet and our iCapsNet/iCapsNet-vote with the same number of training samples and test samples to three traditional methods (SVM, SRC, and CNN) according to [1]. Similarly, we train and test iCapsNet-vote[1], iCapsNet-vote[2], and iCapsNet-vote[3] with the same number of training samples and test samples to 2D-CNN-LR [2], CNN-PPF [15], and FCN-ELM [14], respectively. The classification accuracy per class and OA coefficient are shown in Table II for the Indian Pines data set. Our proposed method achieves the best



(a) (b) (c) (d)

Fig. 4. Classification maps obtained by using different classification methods for IGARSS 2018 data set. (a) 2D-CNN-LR [2] (OA = 44.27%). (b) Two-branch [19] (OA = 26.81%). (c) CapsNet (OA = 43.97%). (d) iCapsNet-vote (OA = 47.53%).

performance either compared with three conventional methods or with three representative CNN-based methods. Furthermore, iCapsNet-vote and iCapsNet achieve the first and second place compared with three traditional methods, and the proposed method outperforms all the selected methods based on CNN in the classification performance. In particular, it can be observed that iCapsNet enhances by about 38% OA compared with the original CapsNet. Thus, it can be inferred that the good performance greatly benefits from the good initialization of the shallow layers via transferring the convolutional parameters from a pretrained CNN model. For qualitative evaluation, the classification maps obtained by corresponding methods illustrated in Table II are visually compared as shown in Fig. 3. For the limited space, we only present the map obtained by the proposed method and three recent CNN-based methods (2D-CNN-LR [2], CNN-PPF [15], and FCN-ELM [14]). Compared with CNN-based method, the ability to capture the spatial relationship of learned instantiation parameters gives the proposed method superiority in the classification of mixed pixels around class boundaries. The vote strategy further removes the outliers which promote the classification map to dramatically approximate the ground truth map.

For IGARSS 2018 data set, we select two traditional deep learning methods, 2D-CNN-LR [2] and Two-Branch CNN [19], but not compare with SVM or SRC, because these two methods are too slow on the large data sets without carrying on GPU and compute unified device architecture accelerator. For limited space, we only present the maps obtained by different methods in Fig. 4. It can be seen that our proposed methods are superior to all the comparative CNN-based methods. However, it is necessary to add some improvements to our proposed method to smooth the map in the future work.

IV. Conclusion

In this letter, a CapsNet-based HSI classification method is proposed instead of the predominant CNN model. An additional convolutional layer is embedded into the shallow layers of CapsNet, which reduces the number of primary capsules and accelerates the routing procedure. Next and more important, the shallow layers of iCapsNet are initialized by transferring the convolutional parameters of a correspondingly pretrained CNN model. This operation guarantees good convergence of the network and dramatically enhances the classification accuracy. In the future work, we will focus on the property of a deeper network with more capsule layers. In addition, the physic meanings of the learned instantiation parameters are also worthy to pay more attention. Furthermore, the spectrum characteristics may be extracted by capsules better. In conclusion, the application potential of capsules on the HSI field is at the beginning which deserves more exploration.

References

[1] J. Yin, H. Qv, X. Luo, and X. Jia, "Segment-oriented depiction and analysis for hyperspectral image data," *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 7, pp. 3982–3996, Jul. 2017.

[2] Y. Chen, H. Jiang, C. Li, X. Jia, and P. Ghamisi, "Deep feature extraction and classification of hyperspectral images based on convolutional neural networks," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 10, pp. 6232–6251, Oct. 2016.

[3] T. V. Bandos, L. Bruzzone, and G. Camps-Valls, "Classification of hyperspectral images with regularized linear discriminant analysis," *IEEE Trans. Geosci. Remote Sens.*, vol. 47, no. 3, pp. 862–873, Mar. 2009.

[4] L. M. Bruce, C. H. Koger, and J. Li, "Dimensionality reduction of hyperspectral data using discrete wavelet transform feature extraction," *IEEE Trans. Geosci. Remote Sens.*, vol. 40, no. 10, pp. 2331–2338, Oct. 2002.

[5] S. Jia, L. Shen, and Q. Li, "Gabor feature-based collaborative representation for hyperspectral imagery classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 53, no. 2, pp. 1118–1129, Feb. 2015.

[6] F. Melgani and L. Bruzzone, "Classification of hyperspectral remote sensing images with support vector machines," *IEEE Trans. Geosci. Remote Sens.*, vol. 42, no. 8, pp. 1778–1790, Aug. 2004.

[7] S. Delalieux, B. Somers, B. Haest, T. Spanhove, J. V. Borre, and C. A. Mücher, "Heathland conservation status mapping through integration of hyperspectral mixture analysis and decision tree classifiers," *Remote Sens. Environ.*, vol. 126, pp. 222–231, Nov. 2012.

[8] J. Ham, Y. Chen, M. M. Crawford, and J. Ghosh, "Investigation of the random forest framework for classification of hyperspectral data," *IEEE Trans. Geosci. Remote Sens.*, vol. 43, no. 3, pp. 492–501, Mar. 2005.

[9] Y. Chen, N. M. Nasrabadi, and T. D. Tran, "Hyperspectral image classification using dictionary-based sparse representation," *IEEE Trans. Geosci. Remote Sens.*, vol. 49, no. 10, pp. 3973–3985, Oct. 2011.

[10] Q. Gao, S. Lim, and X. Jia, "Hyperspectral image classification using joint sparse model and discontinuity preserving relaxation," *IEEE Geosci. Remote Sens. Lett.*, vol. 15, no. 1, pp. 78–82, Jan. 2018.

[11] H. Lee and H. Kwon, "Going deeper with contextual CNN for hyperspectral image classification," *IEEE Trans. Image Process.*, vol. 26, no. 10, pp. 4843–4855, Oct. 2017.

[12] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9.

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[14] J. Li, X. Zhao, Y. Li, Q. Du, B. Xi, and J. Hu, "Classification of hyperspectral imagery using a new fully convolutional neural network," *IEEE Geosci. Remote Sens. Lett.*, vol. 15, no. 2, pp. 292–296, Feb. 2018.

[15] W. Li, G. Wu, F. Zhang, and Q. Du, "Hyperspectral image classification using deep pixel-pair features," *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 2, pp. 844–853, Feb. 2017.

[16] Y. Chen, C. Li, P. Ghamisi, X. Jia, and Y. Gu, "Deep fusion of remote sensing data for accurate classification," *IEEE Geosci. Remote Sens. Lett.*, vol. 14, no. 8, pp. 1253–1257, Aug. 2017.

[17] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," in *Proc. Conf. Adv. Neural Inf. Process. Syst. (NIPS)*, Dec. 2017, pp. 3856–3866.

[18] E. Xi, S. Bing, and Y. Jin. (Dec. 2017). "Capsule network performance on complex data." [Online]. Available:https://arxiv.org/abs/1712.03480

[19] J. Yang, Y.-Q. Zhao, and J. C.-W. Chan, "Learning and transferring deep joint spectral–spatial features for hyperspectral classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 8, pp. 4729–4742, Aug. 2017.

[20] *DataOnline*. Accessed: Nov. 5, 2018. [Online]. Available: http://www.grss-ieee.org/community/technical-committees/data-fusion/data-fusion-contest/