

Exploring Cross-Domain Pretrained Model for Hyperspectral Image Classification

Hyungtae Lee, *Member, IEEE*, Sungmin Eum, *Member, IEEE*, and Heesung Kwon, *Senior Member, IEEE*

Abstract—A *pretrain-finetune* strategy is widely used to reduce the overfitting that can occur when data is insufficient for CNN training. First few layers of a CNN pretrained on a large-scale RGB dataset are capable of acquiring general image characteristics which are remarkably effective in tasks targeted for different RGB datasets. However, when it comes down to hyperspectral domain where each domain has its unique spectral properties, the pretrain-finetune strategy no longer can be deployed in a conventional way while presenting three major issues: 1) inconsistent spectral characteristics among the domains (e.g., frequency range), 2) inconsistent number of data channels among the domains, and 3) absence of large-scale hyperspectral dataset.

We seek to train a universal *cross-domain* model which can later be deployed for various spectral domains. To achieve, we physically furnish multiple inlets to the model while having a universal portion which is designed to handle the inconsistent spectral characteristics among different domains. Note that only the universal portion is used in the finetune process. This approach naturally enables the learning of our model on multiple domains simultaneously which acts as an effective workaround for the issue of the absence of large-scale dataset.

We have carried out a study to extensively compare models that were trained using *cross-domain* approach with ones trained from scratch. Our approach was found to be superior both in accuracy and in training efficiency. In addition, we have verified that our approach effectively reduces the overfitting issue, enabling us to deepen the model up to 13 layers (from 9) without compromising the accuracy.

Index Terms—Hyperspectral image classification, Pretrain-finetune strategy, Cross-domain

I. INTRODUCTION

IN many classification tasks, convolutional neural network (CNN) has been showing a series of innovative performances. However, when only given a small-sized target dataset, it is difficult to avoid the overfitting issue due to an enormous number of parameters that need to be optimized in a deep CNN. One widely known approach to go around this issue is to finetune the model from first few layers of a

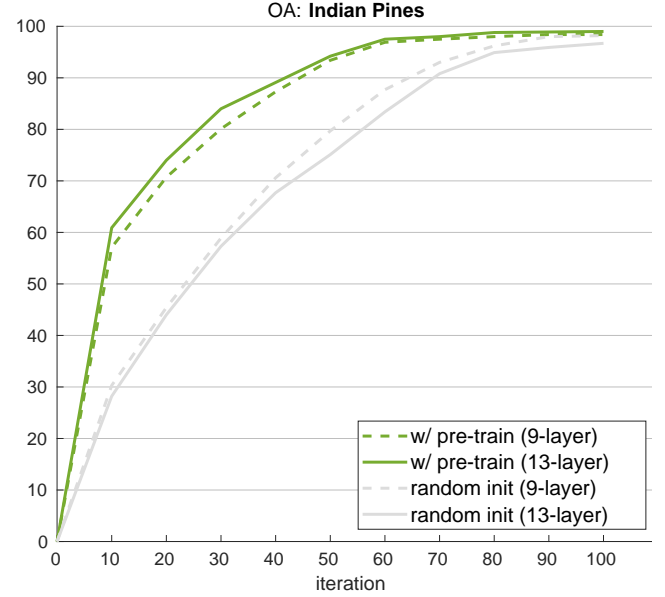


Fig. 1: Learning curve on OA. 9- and 13-layer models initialized from scratch or a pretrained model are evaluated with Indian Pines domain. The pretrained model is trained on six source domains by using our cross-domain approach. Evaluation metric is overall accuracy (OA).

pretrained model which was previously trained on a large-scale dataset [1]. This approach can be applied effectively when the source and the target datasets share equivalent spectral characteristics (e.g., RGB to RGB) and the size of the source dataset is much larger than that of the target dataset.

However, when considering classification tasks in hyperspectral domain¹ where domains contain their own spectral properties, there are challenges when using this conventional *pretrain-finetune* strategy. First of all, hyperspectral datasets acquired by different sensors have different spectral characteristics. For example, Indian Pines domain has 200 bands representing $0.4\sim 2.5\mu\text{m}$ frequency range while Pavia Centre domain has 102 bands covering $0.43\sim 0.86\mu\text{m}$. The spectral feature learned from the source may be unfit for the target usage. Moreover, as number of data channels among different hyperspectral domains are inconsistent, it is infeasible to build an initial layer which can universally handle different domains. Lastly, due to difficulties in annotating hyperspectral images, most of the domains only contain an extremely small

¹In this paper, “hyperspectral domain” is represented by a dataset consisting of hyperspectral data (typically one single image) and its associated classification task.

Manuscript received April 19, 2005; revised August 26, 2015.

H. Lee and H. Kwon are with the Intelligent perception branch, the Computational & Information Sciences Directorate (CISD), Army Research Laboratory, Adelphi, MD, 20783 USA (e-mail: {hyungtae.lee, heesung.kwon}.civ@mail.mil).

S. Eum is with Booz Allen Hamilton Inc., McLean, VA, 22102 USA and with the Intelligent perception branch, the Computational & Information Sciences Directorate (CISD), Army Research Laboratory, Adelphi, MD, 20783 USA (e-mail: eum_sungmin@bah.com).

© 2022 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

number of labeled examples ($\sim 10K$), which can easily cause overfitting issues when training high-capacity CNN models.

To address the aforementioned issues, we devise a *cross-domain* approach which enables the training of a CNN model which can be universally deployed for various spectral domains. We have equipped this model with multiple inlets (i.e., data analysis layers) to physically handle different domains. Meanwhile, in order to capture the universal aspect of different domains, we prepare a shared portion in the model (i.e., mid-level feature analysis layer). Task-specific layers can be appended on the other side of the shared portion of the model dependent upon the task properties such as category size. As the overall design of the architecture allows simultaneous training of all the available hyperspectral domains, the issue regarding the absence of large-scale dataset can be remedied effectively.

In the overall finetune process for a given target domain, only the shared portion of the model pretrained on multiple source domains is to be inherited while all the other layers (data analysis and task-specific layers) are initialized and trained from scratch. It is noteworthy to mention that using a $10\times$ learning rate for the data analysis layer (while keeping $1\times$ for the shared portion) was crucial to obtain a significant accuracy gain. This specific procedure can quickly adapt the data analysis layer to the newly introduced target domain while slowly optimizing the shared portion of the model which already contained the general spectral characteristics acquired from the source domain.

In this paper, we validate the effectiveness of using a cross-domain pretrain-finetune approach for the task of hyperspectral image classification. We have first designed a CNN model which outperforms all the available CNN-based models by employing recently introduced performance-increasing modules. Across all the experiments, this model has been set as the backbone model to evaluate the performance with/without the cross-domain pretrain-finetune approach. Based on our experiments, we observed several benefits of using a cross-domain pretrained model, as shown in Figure 1. Most importantly, the pretraining provides better accuracy than its counterpart randomly initialized from scratch. In addition, the proposed approach moderates the overfitting issue which can occur when the network depth increases, allowing deeper layers (up to 13 layers from 9) without compromising accuracy. We also observed that our approach results in faster training convergence which can reduce the training time.

To provide a practical set of guides in training an effective cross-domain pretrained CNN model, we carried out an additional comprehensive study to answer the following questions²:

- 1) *Is pretraining necessary for hyperspectral image classification?*
- 2) *Does having a larger source domain improve overall accuracy?*
- 3) *Is pretraining effective when target and source domains are obtained from different sensors?*
- 4) *Does introducing more variety in the source domains for pretraining increase the accuracy?*

II. RELATED WORKS

A. Pretrain and Finetune

Girshick et al. [1] firstly use a pretrained model trained on very large-scale dataset to overcome data scarcity. At that time, existing object detection dataset (e.g., PASCAL VOC [4]) did not include enough images annotated with object information (bounding boxes, categories, etc.). Therefore, the state-of-the-art CNN architecture for image classification becomes the backbone of object detection model, and the model weights are finetuned from the backbone trained on a very large-scale ImageNet dataset [5]. In [1], this strategy increases the accuracy by 8.0% on PASCAL VOC. Due to such a large margin of accuracy, pretrained models have been used undoubtedly in many tasks [6]–[12] over the past few years.

Recently several researchers have begun comprehensive analyses on the effectiveness of pretraining. Mahajan et al. [13] analyze the effect of pretraining when increasing the pre-training dataset size. To significantly increase the dataset size, [13] collect social media images and adopt weakly supervised strategy due to a lack of labels of these images. When dataset scale was extremely enlarged, the classification accuracy was proportionally increased. He et al. [14] questioned whether pretraining actually increases the classification accuracy. According to [14], a randomly initialized model provides compatible accuracy to the model finetuned from a pretrained model as long as it is trained with extremely large amount of training time. Based on this observation, they conclude that using pretrained models trained on large datasets is not requirement in achieving high accuracy. In this paper we also carry out comprehensive studies on the pretraining for hyperspectral image classification according to recent trends.

B. Hyperspectral Image Classification

Recently, many CNN-based approaches have been introduced to tackle hyperspectral image classification. Most of CNN-based approaches have rebuilt the architecture with existing layers or modules used for other typical recognition problems such as image classification (e.g., LeNet [17], residual module [15], multi-scale filter bank [15], [18], [19], deconvNet exploiting deconvolutional layers [20], Long Short-Term Memory (LSTM) [21], [22], Recurrent Neural Network (RNN) [23], capsule module [24] etc.). Chen et al. [25] introduce a system that automatically designs a structure with the existing layers to provide the highest classification accuracy.

²Although we elaborated such practical research questions within our preliminary work [2], we managed to analyze our pretrain-finetune approach with a limited number of experimental evaluations and thus could not address the resolutions for the questions in a thorough manner. In this journal manuscript, the entire set of experiments has been redesigned to reach at the conclusions through more comprehensive analyses.

While the initial form of the cross-domain architecture used for the pretraining process was introduced in another preliminary work [3], a restructuring procedure was carried out in order to suit the need for our final pretrain-finetune process. In addition, we constructed a completely different backbone yielding a higher accuracy. We have also conducted an extensive set of ablation studies, and newly added experimental comparisons with additional set of CNN-based hyperspectral image classification models. Instead of appending additional experiments and analyses on the aforementioned manuscripts, substantial portion of this manuscript has been rewritten.

TABLE I: The proposed backbone model architecture. The table shows the architectural differences between the backbone CNN and the contextual CNN from which the backbone CNN originated. Floating-point operations (FLOPs, multiply & addition) are calculated assuming that the Indian Pines domain is used. That is, image height (H), image width (W), spectrum dimension, and the number of categories (C) are set as 145, 145, 200, and 9, respectively. In order to have 9 layers in both models, k_1 and k_2 are set to 2 and 3, respectively.

layer name	output size	contextual CNN [15]	output size	our backbone
Conv 1	$H \times W \times 128$	$1 \times 1, 128$	$H \times W \times 128$	$5 \times 5, 128, \text{pad } 2$
	$H \times W \times 128$	$3 \times 3, 128, \text{pad } 2^*$		
	$H \times W \times 128$	$5 \times 5, 128, \text{pad } 4^\dagger$		
	$H \times W \times 384$	channel-wise concat.		
Conv 2	$H \times W \times 128$	$1 \times 1, 128$	$H \times W \times 128$	$1 \times 1, 128$
Res x	$H \times W \times 128$	$\begin{bmatrix} 1 \times 1, 128 \\ 1 \times 1, 128 \end{bmatrix} \times k_1$	$H \times W \times 128$	$\begin{bmatrix} 1 \times 1, 128 \\ 1 \times 1, 128 \end{bmatrix} \times k_2$
Classif 1	$H \times W \times 128$	$1 \times 1, 128$	$H \times W \times C$	$1 \times 1, C$
Classif 2	$H \times W \times 128$	$1 \times 1, 128$		
Classif 3	$H \times W \times C$	$1 \times 1, C$		
FLOPs ‡		43.9×10^9 ($k_1=2$)		31.8×10^9 ($k_2=3$)

* 3×3 max pooling is applied to the output.

† 5×5 max pooling is applied to the output.

‡ FLOPs is calculated as the equation given in [16].

Some previous approaches improve accuracy by integrating multiple methods [26] or multiple different features [27], [28].

There are also several literatures that introduce new layers or modules to deal with the inherent limitations of hyperspectral images, especially the overfitting problem that occurs because there are very few hyperspectral images labeled due to the difficulty of annotations (e.g., pyramidal bottleneck module [29], lightweight unit [30], multi-bias layer module [31], the Generative Adversarial Network (GAN) [32]). Instead of new layers or modules, new data augmentation strategies based on either GAN [33], [34] or active learning [35] are also used to cope with the overfitting problem. It is also widely used to simply reduce the dimensions of the image spectrum via PCA or spectrum selection to reduce the size of the model, thus alleviating the overfitting problem to some extent [36]–[38].

Similar to our approach, some works [30], [39]–[41] also use pretraining, but do not properly tackle the issues addressed in this paper (no large-scale source domains, different spectral characteristics between different domains). Windrim et al. [40] collect multiple source domains and interpolate spectral characteristics of different domains into the shared spectrum. However, due to the large fluctuations in the spectrum, the interpolation cannot restore the missing spectrum information accurately, and this affected accuracy degradation. Other approaches [30], [39], [41] use only one for the source domain, so large-scale source domains are not built. Our cross-domain approach can address such issues which other works miss and comprehensive studies are conducted to verify the effectiveness of our approach to improve the accuracy of hyperspectral image classification.

III. METHODOLOGY

Our goal is to explore and figure out a way to resolve the issue of not being able to apply a conventional way of pretraining a CNN model for the task of hyperspectral image classification. We first design a state-of-the-art CNN model for this task by employing recently introduced

performance-increasing modules (subsection III-A). Across all the experiments, this model has been set as the backbone model to evaluate the performance with/without the proposed pretrain-finetune approach. Then, we will also describe the newly devised cross-domain pretrain-finetune approach (subsection III-B). In subsection III-C, details used for optimizing the cross-domain pretrain-finetune approach are given.

A. Backbone

The backbone is built based on a contextual CNN architecture which is the deepest model among all existing CNN-based methods [15], [42]. The contextual CNN is a fully convolutional network and contains the sequentially connected multi-scale filter bank, one convolutional layer, two 2-layer residual modules, and three convolutional layers, as shown in Table I. Multi-scale filter bank [43] consists of 1×1 , 3×3 , and 5×5 filters for better analysis of the spatial characteristics. Local response normalization (LRN) [44] for better generalization during training is applied after the first and second convolutional layers. Dropout layers [45] were adopted after 7th and 8th layers to enable faster training by preventing complex co-adaptation of each neuron. Each layer consists of 128 convolutional filters.

The backbone was constructed by adopting the contextual CNN with a recently introduced new layer known to improve accuracy while removing ineffective modules, as follows:

- (i) **No multi-scale filter bank:** Instead of a complex multi-scale filter bank, a convolutional layer with only 5×5 filters is used.
- (ii) **More residual modules:** All layers are made up of residual modules, except the first, second, and last layers. Dropout is not necessary because it has the similar functionality as the residual module for easing network training.
- (iii) **Batch normalization (BN):** BN [46] is adopted right after each convolutional layer, so there is no need for another normalization method such as LRN. BN helps

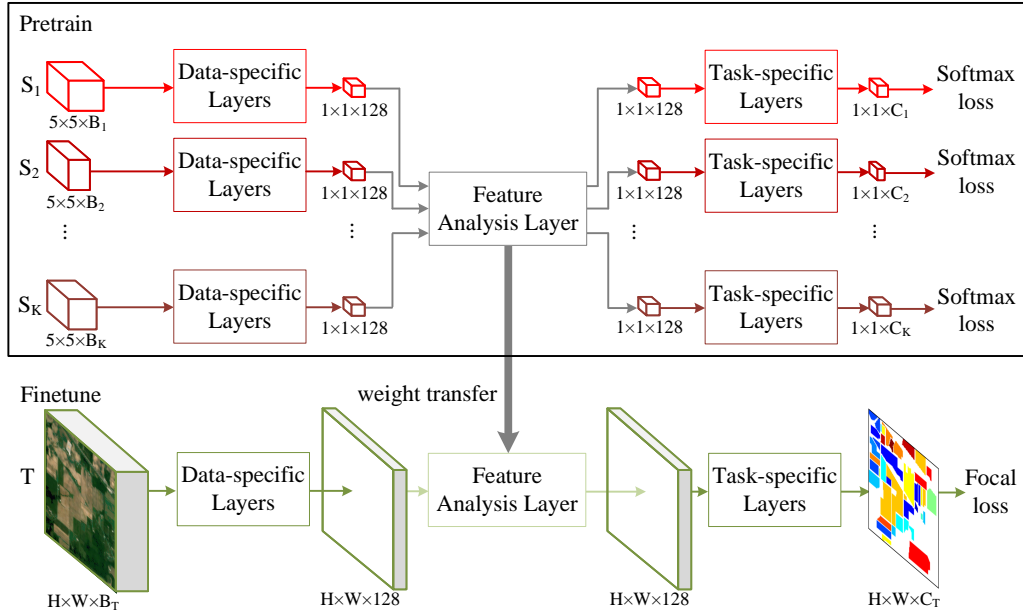


Fig. 2: Pretrain-finetune strategy. A pretrained model is trained with K source domains (S_1, \dots, S_K) and the target model is finetuned from this pretrained model on a target domain T . Feature dimensions are denoted below each blob. For each domain, the number of data channels is denoted by B_i , and the number of categories classified by the associated task is denoted by C_i .

the network learn the representation of the entire hyper-spectral image when using a small number of training example. By adding BN to the network, bias terms are unnecessary unlike the original CNN architecture.

- (iv) **Focal loss:** We use the focal loss [47] to optimize the model for a target task. This loss enables all examples to be effectively considered for each iteration by giving higher weights to harder examples. Since hyperspectral domains usually contain many easy examples, the focal loss was found to be suitable. Softmax loss is used when pretraining on multiple source domains as used in contextual CNN.

Such changes reduce the network size and FLOPs while increasing the accuracy according to our experiments (Table VI). Table I compares the modified backbone with the contextual CNN in terms of architectural differences and FLOPs.

B. Building Cross-Domain Pretrained Model

Our pretrained model is built by using a cross-domain approach. This enables the training of a CNN model which can be universally deployed for various hyperspectral domains. The architecture sequentially consists of multiple inlets (i.e., data analysis layers), shared portion across multiple domains (i.e., mid-level feature analysis layers), and multiple task-specific layers. The data analysis layers that physically handle different datasets cannot be shared universally due to the inconsistent number of channels in each dataset. Task-specific layers that depend on the task properties such as category size cannot be shared as well.

After careful consideration, all residual modules are selected as “mid-level feature analysis layers” that are shared by the various spectral domains. The 1st and 2nd layers are treated as “data-specific layers” that analyze data-specific spectral

characteristics, and the last set of layers are considered as “task-specific layers”.

In making use of a pretrained model towards a “target” dataset, we finetune the middle portion (“mid-level feature analysis layers”) from the pretrained model while other layers are learned from scratch. In the finetune process, we use $10\times$ learning rate for the data analysis layers (while keeping $1\times$ for the shared portion). This specific procedure allows the data analysis layers to quickly converge on the target dataset, while slowing down the optimization of the shared portion of the model in order to maintain the ability to extract the general spectral characteristics acquired from the source domain. This was crucial to obtain a significant accuracy gain when compared to not adjusting the learning rate. The overall pretrain-finetune strategy is illustrated in Figure 2.

Reasoning about the role of each layer. Even though it is challenging to semantically find the role of each layer of CNN model, we can infer these roles. In general, all CNN layers other than the last layer are considered *feature encoder* which encodes input examples in the learned common feature space, and the last layer uses these features to carry out a classification task. If there are multiple domains to be encoded in a common feature space and the characteristics of domains are significantly different, different encoders are required for each domain. In our cross-domain model, we technically create different encoders for different domains, and this was found to be possible by simply sharing the layers across different encoders but uniquely setting the first layers for different domains.

Relation to previous works. The effectiveness of our design strategies (i.e., multiple task-specific layers, multiple encoders, and transferring ability when properties between source and

TABLE II: Specifications for different hyperspectral domains. We use the domain abbreviations written in parentheses throughout the paper. Reduced bands are acquired by removing the bands which correspond to the water absorption. Note that, datasets acquired using the same type of sensor can have different reduced bands because the deleted bands were chosen according to the task unique to each dataset.

domain	sensor	range	bands	reduced bands	# data	# class
Indian Pines (I)	AVIRIS	0.4~2.5 μ m	224	200	8,504	9
Salinas (S)				204	54,129	17
Kennedy Space Center (KSC)				176	5,211	14
Pavia Centre (PC)	ROSIS	0.43~0.86 μ m	115	102	7,456	10
Pavia University (PU)				103	42,776	10
Botswana (B)	Hyperion	0.4~2.5 μ m	242	145	3,248	15
Houston University (H)	Unknown	0.38~1.05 μ m	144	144	15,029	15

target domains are different) have been validated in several previous works shown below:

- **Multiple task-specific layers.** *Multi-task learning*, performing multiple tasks with different layers in a common learned space, is widely used and has presented better accuracy than using multiple single task networks [9], [48]–[51].
- **Multiple encoders.** The key component of *Contrastive Language-Image Pretraining (CLIP)* [52], which has recently been attracting attention due to its remarkable representative ability in the feature space, is to use two encoders to encode the language input and the image input in a common feature space, like our cross-model architecture.
- **Transferring ability when properties between source and target domains are different.** *Self-supervised learning* [7], [53]–[57], which has recently been explosively used in representation learning, shows remarkable performance when transferring to target tasks using a pretrained networks on a simple pretext task without relying on class labels. These works show that the pretrain-finetune strategy is effective even when the source task and the target task are completely different.

C. Optimization

We prepare different training strategies depending on the training scenario:

- **Batch containing the entire set of examples (Target domain):** When training a model with a single target domain, we use a batch containing the entire set of examples for each training iteration to expedite the convergence. Focal loss has a capability to effectively train on such a batch.
- **Mini-batch built with randomly selected examples (Source domain):** Unlike the target domain scenario, a mini-batch optimization is used. This is due to the fact that the source domain training involves multiple domains concurrently, inevitably providing GPU memory issues when the entire set of examples are used for training.
- **Two-step cascade training strategy (Source domain):** When using multiple datasets for training, we should consider a potential issue caused by the imbalanced datasets. Since the Salinas dataset (S) or the Pavia University (PU) has much more data, it requires more iterations than the

others. To cope with this issue, we adopt a two-step optimization strategy introduced in [49], [50], [58]. Under this scheme, the model is initially trained on only the largest dataset (Step I (S or PU)) and then is updated using the whole dataset (Step II). S and PU are used together as source domains, cross-domain optimization is also used in step I. Both steps use a mini-batch optimization.

Initialization. The layers that are not inherited from the pretrained model are initialized according to the Gaussian with zero-mean and standard deviation of 0.001. This setting was very important in providing high accuracy. When we had adopted recently introduced initialization methods (e.g., Xavier [59], kaiming_init [60]) or higher standard deviation (e.g., 0.01), the accuracy was significantly degraded by $\sim 30\%$.

Data augmentation. Hyperspectral image classification typically uses a small number of examples for training (e.g., several thousands examples to optimize 1.1M parameters), which can cause overfitting problems. To provide richer set of examples to cope with this issue, the training examples are augmented eight-fold by mirroring each example across the vertical, horizontal, and two diagonal axes [15].

Learning rate tuning for shared layers. When learning the shared layers, we multiply $1/N$ (where N is the number of domains involved in the training process) to the learning rate because updating the weights in these layers are affected by all N losses when back-propagation takes place at each iteration. This strategy has proven effective for multi-task learning in [8].

IV. EXPERIMENTS

A. Settings

Domains. Seven hyperspectral domains shown in Table II are used for the experiments. The Indian Pines domain has 17 classes but only 9 classes (including background class) with relatively large numbers of examples are used. For other domains, we use all available classes. Indian Pines domain is used as the target domain while various combinations of the remaining six domains are used as the source domains. Once a pretrained model is built using source domains, finetuning process is carried out on a target domain.

Our main goal is to explore how pretrain-finetune strategy affects the overall accuracy for the target domain. Therefore,

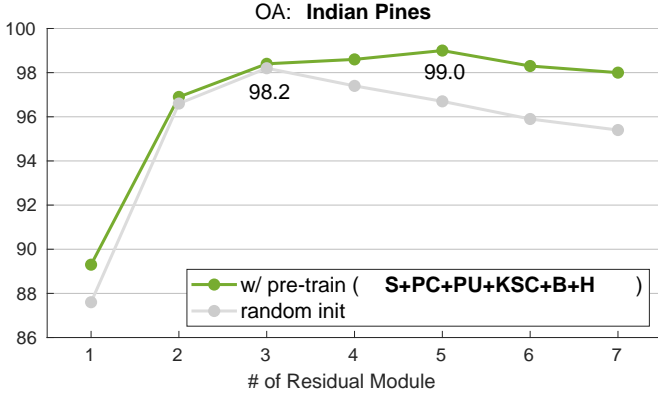


Fig. 3: OA on Indian Pines domain with architectures having different depths. The proposed approach has different architectures depending on the number of the residual modules. The pretrained model is built with six source domains (S, PC, PU, KSC, B, and H).

when training on the source domains (i.e., pretraining), all the available examples are used. For the target domain, 200 pixels randomly selected from each category are used for training and all the remaining pixels are used for testing. All the accuracy values reported hereafter are acquired solely on the target domain.

Learning specifications. We use the stochastic gradient descend (SGD) approach for training. We set the momentum, gamma, and weight decay as 0.9, 0.1, and 0.0005, respectively.

When **S** or **PU** is not included in the source datasets (i.e., one-step optimization), a pretrained model is trained with a learning rate of 0.01 for 2K iterations. When a two-step cascade optimization is used, we use a learning rate of 0.01 for 1K iterations for the first step, while the second step uses the same settings used in the one-step optimization. For the target domain, the model is trained with a learning rate of 0.001 for 100 iterations.

Evaluation metrics. We consider three metrics for evaluation: overall accuracy (OA), average accuracy (AA), and kappa static. All three metrics are used when comparing with other previous methods, while only OA is used for the ablation experiments.

B. Analysis of Pretrained Model

In this subsection, we answer each question claimed in the introduction section and provide relevant experiments to support our answer.

Is pretraining necessary? Yes. There are three benefits to using a pretrained model as below:

- (i) Models with a pretrained model consistently provide higher accuracy than their random initialization counterparts, regardless of model architecture.
- (ii) Model using a pretrained model can be built deeper with more residual modules up to 13 layers without compromising the accuracy.

(iii) Learning converges faster, which reduces the training time.

Figure 3 compares the accuracy of models using pretraining and models trained from scratch. Using pretraining consistently provides 0.2%~2.6% higher accuracy than all of the randomly initialized models.

It is also observed that the 9-layer model provides the highest accuracy when trained from scratch, which is also shown in [15]. However, with a pretrained model, the 13-layer model provides the best accuracy. This “going deeper” (i.e., network deepening) was made possible since the overfitting issue could be addressed by finetuning on a pretrained model that has the same effect as increasing the size of the training dataset.

Figure 4 shows learning curves for three models (5-, 9-, 13-layer). Models that use pretraining are generally shown to converge faster than their counterparts. Furthermore, a model trained from scratch takes more time to converge as the model goes deeper. This may be due to the expansion of overfitting issues as the model size increases.

Interestingly, our observation is very different from that of [14]. For the object detection described in [14], when using a pretrained model, fast convergence was observed, but this model does not outperform the random initialization counterpart regarding to the accuracy. The major claim made in [14] is that given sufficient iterations, the model trained from scratch can achieve comparable accuracy as the model using pretraining. It is noteworthy to mention that the same trend was said to have happened when only small number of training examples (e.g., subset of COCO dataset) were available.

However, unlike the claim made in [14], the performance comparison shows that a pretrained model indeed outperforms the one trained from scratch leading to an opposite conclusion. We consider the hyperspectral image classification as an extreme case where very few examples are available for training, thus showing a great demand to expand the training dataset.

Does a larger source dataset improve accuracy? Yes. We question how much influence the dataset size of a source domain has over the classification accuracy. To answer this question, we generate 63 source domains by combining up to 6 source domains and use them to train separate models. Figure 5 shows the sizes of 63 source domains (along with one constructed with random initialization) and OAs obtained by finetuning each pretrained model on the target domain. In the figure, we observed that using a larger source domain generally improves the classification accuracy. In all the source cases, accuracy was better than the case where the model was randomly initialized.

Does source domain need to come from same sensor as target domain? No. Pretraining was effective regardless of the sensor type by which the source domain was obtained.

We verify the effect of pretraining a model on a source domain acquired by a sensor different from the target data. In general, using a pretrained model to finetune is effective only if the source data which is used for pretraining and the target data share the same sensor.

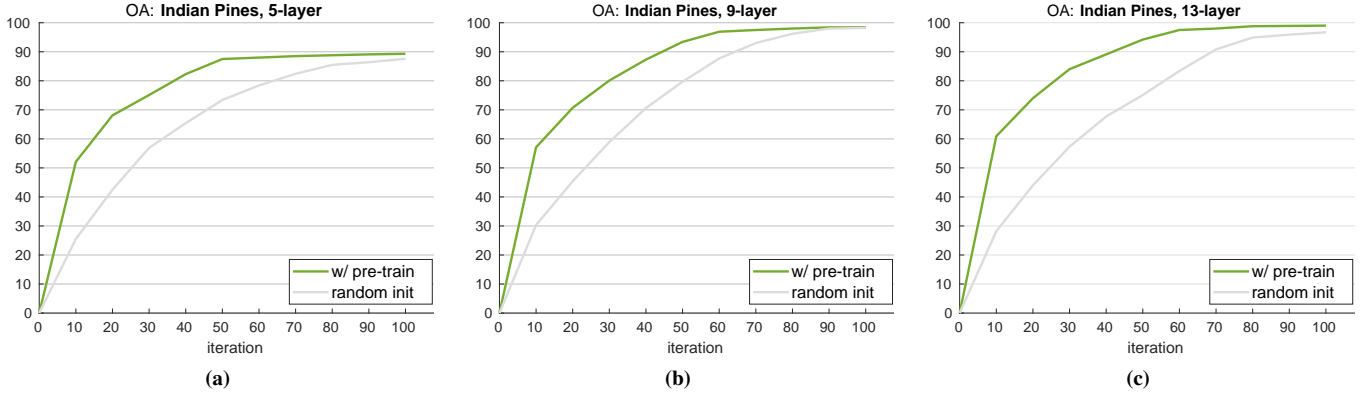


Fig. 4: Learning curves on OA with three models (5-layer, 9-layer, 13-layer). Pretrained models are trained on six source domains and target model is evaluated on the Indian Pines domain.

TABLE III: Comparison w.r.t. source data sensors. The accuracy of the models using single source domain is shown on the left side. Several combinations of source domains which includes or excludes **S** or **KSC** are constructed and tested on the right side. Note that **S** and **KSC** are taken from the same sensor as **I**.

	S	KSC	PC	PU	B	H	S+	KSC+	S+ ∪ KSC+	/(S+ ∪ KSC+)
OA	96.9	97.5	96.9	97.0	96.6	96.7	98.2	97.9	97.9	97.1
data size	52.8k	5.1k	7.3k	41.8k	3.2k	14.7k

+: all sets including the corresponding source dataset (shown before the + mark)

∪: union of two sets

/(): a complement set of ()

TABLE IV: Performance comparison of using single source and multiple sources. There are five cases where a single domain and a combination of multiple domains have similar data size in figure 5. Domains are sorted according to the data size so the larger size is placed on the right. In each case, the best accuracy is written in bold.

	PC	KSC+B	KSC+PC	H	KSC+PC+B	KSC+PC+B+H	PU	PU+B	PC+PU+B	S	KSC+PC+PU
OA	96.9	97.3	96.9	96.7	97.9	97.2	97.0	97.0	97.3	96.9	97.0
data size	7.3k	8.3k	12.4k	14.7k	15.5k	30.2k	41.8k	44.9k	52.2k	52.9k	54.1k

As shown in Table III, using source domains that share the same sensor as the target domain (i.e., **S** and **KSC**) did not provide the benefit of improving the accuracy compared to the cases where the source and the target sensors were different (i.e., **PC**, **PU**, **B**, and **H**). Instead, accuracy seems to be more affected by the dataset size of the domain. In addition, we have tested to validate how the inclusion/exclusion of source data acquired by specific sensors (i.e., matching the target sensor with the source) affect the accuracy when considering combinations of domains. When comparing groups consisting of source domains containing either **S** or **KSC** with a group excluding both domains, the latter group ($/(S+ \cup KSC+)$) offers the worst accuracy among all groups, but the difference is marginal (0.8%).

Does introducing more variety in the source domains for pretraining always increase the accuracy? No. The source training domain can be either a single domain or a combination of multiple domains. In Figure 5, there are four groups where a single domain and combinations of multiple domains have similar data size. The selected groups are analyzed in detail in Table IV. In all four groups, introducing multiple sources did not bring forth any drastic performance increase. This demonstrates that the accuracy is not affected by the number

of different source domains involved in the pretraining.

C. Ablation Study

We carried out several ablation experiments to validate each component (e.g., cross-domain approach, new network components, etc.) used to analyze the need for a pre-trained model.

Train on source and test on source: effectiveness of cross-domain pretraining. The cross-domain approach used in the pretraining process has the ability to train a model simultaneously in multiple hyperspectral domains with different spectral characteristics. In this section, we evaluate the effectiveness of the proposed cross-domain pretraining approach (leaving out the finetuning process) when trained and tested on the source domain (source-to-source). Note that this setting is different from the case shown in the main article where pretraining is carried out on the source domain and finetuned on the target domain (source-to-target).

We evaluate whether building a single cross-domain pre-trained model which handles multiple domains altogether is better than the models separately trained for specific domains. We use seven domains (i.e., **I**, **S**, **KSC**, **PC**, **PU**, **B**, and **H**). Since **PC** is considered, we use two-stage cascaded

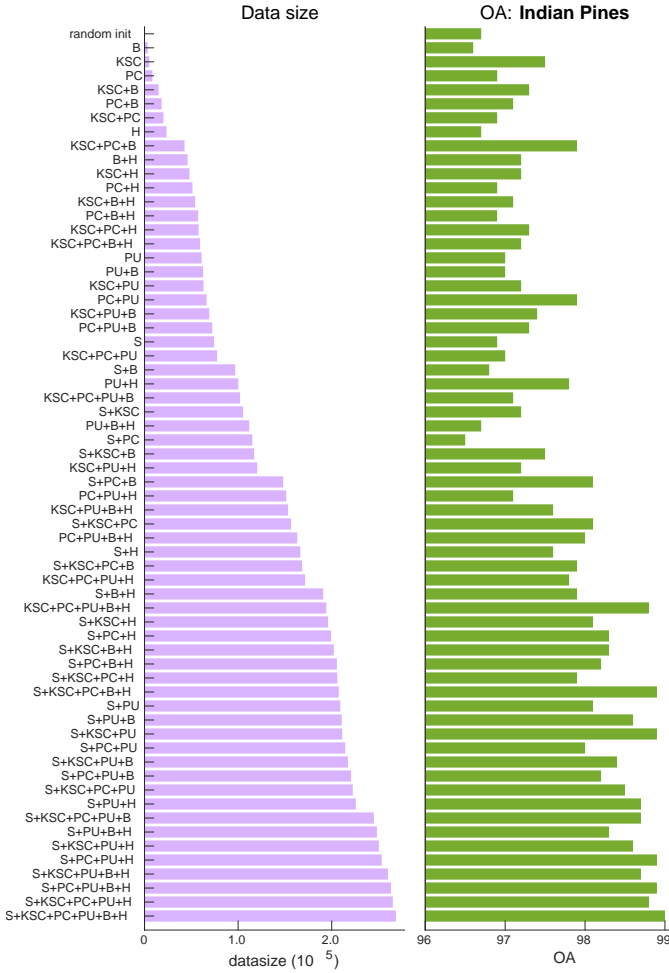


Fig. 5: Data size and OA for all combinations of 6 source datasets. Datasets are sorted according to their sizes and listed on the x axis. Two graphs share the x axis..

optimization approach according to the criteria provided in the main article. For each domain, we use 200 examples randomly chosen from each positive category for training and the remaining for testing. OAs reported in this experiment are the average value over 5 runs. As shown in Table V, our cross-domain approach consistently outperforms the individual models on the Indian Pines, Salinas, KSC, Pavia Center, Pavia University, Botswana, and Houston by 1.1%, 1.1%, 1.3%, 1.7%, 2.3%, 1.6%, and 1.4%, respectively.

We also compare two cases (source-to-source vs. source-to-target) which are both tested on Indian Pines dataset. For both cases, the dataset is divided into train and test splits using the same regime. We observe that the source-to-source accuracy was higher than that of the source-to-target (99.3 vs. 98.9). This strategy of co-training multiple domains together within source-to-source setting, can be another effective solution to address the overfitting issue caused by insufficient amount of data. However, it suffers from the fact that large memory and multi-domain dataset should be available for training.

Effectiveness of backbone architecture. The backbone used in our experiments incorporates various new components that

TABLE V: OA on seven domains. For each domain, model is trained separately from the other domains (individual) or simultaneously with other domains (cross-domain).

dataset	individual	cross-domain	gain
I	98.2	99.3	+1.1
S	97.6	98.7	+1.1
KSC	96.6	97.9	+1.3
PC	96.9	98.6	+1.7
PU	95.1	97.4	+2.3
B	97.5	99.1	+1.6
H	96.9	98.3	+1.4

has been proved to be effective in increasing the performance. Table VI shows the list of components along with the resulting accuracy when each of those were added to the model.

There are three architectural modifications: no multi-scale filters, more residual modules, and BN. Using a single 5×5 convolutional layer instead of a complicated multi-scale filters can improve performance slightly (0.3%) and it simplifies the architecture with fewer parameters. The other two modifications serve to increase the accuracy with a large gain of 3.1% and 3.3%, respectively. Easing model optimization by incorporating more residual modules seems to be very effective in improving performance. From the observation that BN also provides improved performance, better regularization by normalizing output of each layer over the examples in each batch was effective.

Focal loss was not very effective in improving accuracy. When using focal loss for training [15] and our backbone, the accuracy change was 0.1% and 0.8%, respectively. The benefit of using focal loss is faster convergence than using other loss types. This will be analyzed further in the next subsection.

The pretrained model was effective in improving performance only when using a large learning rate for the initial layers. Without such adjustment, it is observed that accuracy was rather decreased. Adjusting the learning rate for the shared layers (i.e., shared layer lr/N) provides a further improvement of 0.1%.

Analysis of the use of a focal loss. As described in [47], two hyper-parameters of focal loss, γ and α control the strength of the modulation term. α is used for addressing the positive/negative imbalance problem, while γ gives more weight to difficult training examples. OA on the target domain (i.e., Indian pines) for various γ and α are shown in Table VIIa. For this comparison, we use 9-layer model initialized from scratch. The highest accuracy was achieved when taking a high γ (5.0) and a low α (0.25). This co-aligns with the fact that Indian Pines domain has many easy examples and the low ratio between foreground and background examples. The selected numbers for the two parameters are used throughout all the experiments carried out in the main article.

In table VIIb, we compare the case where we train the entire set of examples at once (using focal loss) to the case where mini-batch-based optimization approaches are used (e.g., random selection, online hard example mining (OHEM) [61]) in terms of accuracy and training time. While the focal loss is used to process the batch consisting of the entire

TABLE VI: Newly adopted components of the proposed model. The proposed model was implemented by adding the recently introduced components to the contextual CNN [15].

	contextual [15]									the proposed
no multi-scale filters?			✓		✓	✓	✓	✓	✓	✓
more residual modules?				✓	✓	✓	✓	✓	✓	✓
BN?						✓	✓	✓	✓	✓
focal loss?	✓						✓	✓	✓	✓
w/ pre-train?								✓	✓	✓
initial layer $10 \times lr$									✓	✓
shared layer lr/N ?										✓
OA	93.6	93.7	93.9	96.7	96.9	97.3	98.0	98.3	98.4	99.0

TABLE VII: Ablation experiments for focal loss (FL). 9-layer model initialized from scratch is used. (a) for each γ value, the optimal value of α is used. (b) FL uses the entire set of training sample as a single batch, while random and OHem use mini-batches with specific batch sizes. For random/OHEM, softmax loss is used. Time indicates the relative wall-clock training time w.r.t. the case of FL, and was measured until the highest accuracy was achieved. Since each iteration time is different for the three methods (random, OHem, and FL), the wall clock training time is used instead of the total iteration.

(a) Varying γ and α			(b) Batch Method			
γ	α	OA	method	batch size	OA	time
0	.75	98.0	Random	128	98.0	$\times 2.4$
0.1	.75	98.0		256	97.6	$\times 2.1$
0.2	.75	98.0		512	97.6	$\times 1.6$
0.5	.50	98.2	OHem	128	98.2	$\times 3.8$
1.0	.25	98.3		256	98.2	$\times 3.7$
2.0	.25	98.9		512	97.9	$\times 3.4$
5.0	.25	99.0	FL	1800	99.0	

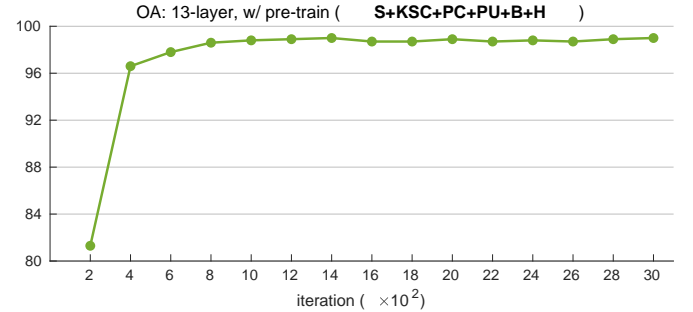
examples, mini-batch-based optimization uses a softmax loss. Using a focal loss provides higher accuracy by at least 0.8 and was effective in reducing training time by at least $\times 1.6$ compared to other mini-batch-based methods. Among mini-batch-based optimization methods, OHem generally provides higher accuracy but requires longer training time than random selection. Both random selection and OHem do not greatly depend on batch size when considering OA.

Appropriate training schedule for a pretrained model.

To find the proper training schedule, we compare various pretrained models with different training iterations in terms of OA. Figure 6 shows the OA with various pretraining iterations. The highest accuracy was achieved at 2k iterations and did not show any additional improvement thereafter. Accordingly, we have selected 2k as the number of iterations to acquire our pretrained model.

D. Comparison with Baselines

We have compared our model with the CNN-based baselines that can be trained in an end-to-end fashion. To make a fair comparison, bells and whistles that cannot be used to carry out the end-to-end training have been omitted in all the models. Table VIII compares our model with the selected baselines using the evaluation metrics of OA, AA, and k . Our backbone

**Fig. 6: Evolution w.r.t. pretraining schedule.** We have tested various pretrained models using different schedule.**TABLE VIII: Comparisons with CNN-based baselines.** For all the baseline models except SSRN and HybridSN, we use DeepHyperX implementations [62]. For SSRN and HybridSN, we have used the original code provided by the authors.

method	layer	# params.	OA	AA	k
Semi-super 1D [63]	10	6.7K	89.1	87.6	84.9
1D [64]	2	69.9K	90.1	89.6	86.9
3D [65]	4	1.0M	90.3	90.6	87.4
2D [66]	5	2.5M	92.8	90.3	85.9
Contextual [15]	9	1.0M	93.6	95.8	94.7
3D-2D [67]	4	99.0M	93.8	92.8	89.8
3D-1D [68]	10	38.8K	94.3	93.8	89.7
Semi-super 2D [69]	6	4.6M	94.6	96.2	92.4
SSRN [70]	11	717.8K	95.8	97.1	93.2
2-stream 3D [71]	3	78.7K	96.5	94.9	89.7
Multi-scale 3D [72]	10	169.7K	96.6	93.8	90.4
HybridSN [73]	7	4.8M	96.8	97.7	95.3
Our backbone	13	1.1M	98.0	98.9	97.7
w/ pretraining	13	1.1M	98.9	99.3	98.1

model even without finetuning is better than the baselines by at least 1.2% for OA, 1.2% for AA, and 2.4% for k . When adopting the pretrain-finetune strategy to utilize representative capability of a pre-trained model on the six source domains, the accuracy was further enhanced by 0.9% for OA, 0.4% for AA, and 0.4% for k . We also confirmed that our model has the deepest architecture without compromising the accuracy.

E. Proof of Concept

Representation capability. We carried out an additional experiment to evaluate the representation capability of pretrained network without finetuning. For this evaluation, the layers

TABLE IX: Effectiveness of pretrain/finetune. “random init.” is the model in which the layers except data-specific layer and task-specific layer are randomly initialized and not updated during training.

method	random init.	w/o pretrain	w/o finetune	w/ finetune
OA	15.0	98.0	87.3	98.9

TABLE X: Accuracy of the pretrain-finetune strategy on Salinas (S) and Botswana (B). When a pretrained model is used, all domains except the target domain are used as source domains.

domain	w/o pretrain	w/ pretrain	gain
S	97.6	98.1	+0.5
B	97.3	98.2	+0.9

transferred from a pretrained network is fixed with the weights trained on source domains (i.e., **S+KSC+PC+PU+B+H**), but the remaining layers (data-specific layer and task-specific layer) are trained on the novel classes defined in the target task. As shown in the Table IX, accuracy achieved without finetuning was much better than “random init.” by a significant margin and comparable to other models without/with the pretrain-finetune strategy in which all layers are updated during training. This comparison supports the representation capability of the pretrained model.

Other source domain? We carried out further experiments considering other datasets (Salinas and the Botswana dataset) as target domains. Two selected datasets have distinct inherent properties compared to other datasets as the Salinas dataset contains the largest number of labeled examples and the Botswana dataset was taken from the satellite unlike other datasets taken from airborne. When acquiring the pretrained model (source domain), we use all the datasets except the one used as the target domain. As shown in Table X, using the pretrain-finetune strategy consistently provides better accuracy than its counterpart of training-from-scratch, regardless of the source domain. Margins achieved by the pretrain-finetune strategy were 0.5% and 0.9% for Salinas and Botswana dataset, respectively. When the source domain is a large-scale (e.g., **S**), the impact of using the pretrained model seems to be reduced. On the other hand, the fact that the source and target domains are from different sources between satellites or airborne (e.g., **B**) does not seem to significantly affect the effectiveness of the pretrain-finetune strategy.

V. CONCLUSION

The conventional pretrain-finetune strategy cannot be directly deployed in hyperspectral domains due to three major issues: 1) inconsistent spectral characteristics among the domains, 2) inconsistent number of data channels among the domains, and 3) absence of large-scale domains. We have devised a cross-domain model that can be universally deployed for various hyperspectral domains. The proposed architecture has multiple inlets and outlets that handle different domains and classification tasks, with the middle portion shared by all domains. The middle portion is expected to extract the universal spectral properties from all the domains involved in

training and to transfer them to the target task in the finetune process.

We carried out a comprehensive study using a variety of ablation experiments to confirm the effectiveness of this pretrain-finetune strategy. Our experiments demonstrate the benefits of using pretraining in three aspects: i) increasing accuracy, ii) building deeper models without sacrificing the performance, and iii) providing faster training convergence which results in reducing training time. From the experiments, we have also observed that the accuracy of the target task does not depend on sensor types or the number of source domains, but rather on the data size of the combined source domain.

ACKNOWLEDGMENT

We would like to thank Prof. Wonkook Kim at Pusan National University for his help with the experiments.

REFERENCES

- [1] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Region-based convolutional networks for accurate object detection and segmentation,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 1, pp. 142–158, Jan. 2016.
- [2] H. Lee, S. Eum, and H. Kwon, “Is pretraining necessary for hyperspectral image classification?” in *Proc. IGARSS*, Jul. 2019, pp. 3321–3324.
- [3] H. Lee, S. Eum, and H. Kwon, “Cross-domain CNN for hyperspectral image classification,” in *Proc. IGARSS*, Jul. 2018, pp. 3627–3630.
- [4] M. Everingham, L. Gool, C. Williams, J. Winn, and A. Zisserman, “The PASCAL Visual Object Classes (VOC) challenge,” *Int. J. Comput. Vis.*, vol. 88, no. 2, pp. 303–338, Jun. 2010.
- [5] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, “ImageNet large scale visual recognition challenge,” *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.
- [6] A. J. Bency, H. Kwon, H. Lee, S. Karthikeyan, and B. Manjunath, “Weakly supervised localization using deep feature maps,” in *Proc. ECCV*, Sep. 2016, pp. 714–731.
- [7] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, “Momentum contrast for unsupervised visual representation learning,” in *Proc. CVPR*, Jun. 2020, pp. 9729–9738.
- [8] H. Lee, S. Eum, and H. Kwon, “ME R-CNN: Multi-expert R-CNN for object detection,” *IEEE Trans. Image Process.*, vol. 29, pp. 1030–1044, 2020.
- [9] H. Lee*, S. Eum*, J. Levis*, H. Kwon, J. Michaelis, and M. Kolodny, “Exploitation of semantic keywords for malicious event classification,” in *Proc. ICASSP*, Apr. 2018, (* indicates authors who contributed equally.).
- [10] H. Lee, H. Kwon, A. J. Bency, and W. D. Nothwang, “Fast object localization using a CNN feature map based multi-scale search,” arXiv:1604.03517, 2016. [Online]. Available: <https://arxiv.org/1604.03517>
- [11] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, “Learning and transferring mid-level image representations using convolutional neural networks,” in *Proc. CVPR*, Jun. 2014, pp. 1717–1724.
- [12] C. Reale, H. Lee, and H. Kwon, “Deep heterogeneous face recognition networks based on cross-modal distillation and an equitable distance metric,” in *Proc. CVPRW*, Jul. 2017, pp. 32–38.
- [13] D. Mahajan, R. Girshick, V. Ramanathan, K. He, M. Paluri, Y. Li, A. Bharambe, and L. van der Maaten, “Exploring the limits of weakly supervised pretraining,” in *Proc. ECCV*, Sep. 2018, pp. 185–201.
- [14] K. He, R. Girshick, and P. Dollár, “Rethinking ImageNet pre-training,” in *Proc. ICCV*, Oct. 2019.
- [15] 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.
- [16] H. Lee and H. Kwon, “How to practically deploy deep neural networks to distributed network environments for scene perception,” in *Proc. SPIE Defense + Commercial Sensing (DCS)*, May 2019, pp. 629–635.
- [17] S. Mei, J. Ji, J. Hou, X. Li, and Q. Du, “Learning sensor-specific spatial-spectral features of hyperspectral images via convolutional neural networks,” *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 8, pp. 4520–4533, Aug. 2017.

- [18] Z. Gong, P. Zhong, Y. Yu, W. Hu, and S. Li, "A CNN with multiscale convolution and diversified metric for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 6, pp. 3599–3618, Jun. 2019.
- [19] M. Zhang, W. Li, and Q. Du, "Diverse region-based CNN for hyperspectral image classification," *IEEE Trans. Image Process.*, vol. 27, no. 6, pp. 2623–2634, Jun. 2018.
- [20] X. Ma, A. Fu, J. Wang, H. Wang, and B. Yin, "Hyperspectral image classification based on deep deconvolution network with skip architecture," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 8, pp. 4781–4791, Aug. 2018.
- [21] L. Mou, P. Ghamisi, and X. X. Zhu, "Deep recurrent neural networks for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 7, pp. 3639–3655, Jul. 2017.
- [22] Y. Xu, L. Zhang, B. Du, and F. Zhang, "Spectral–spatial unified networks for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 10, pp. 5893–5909, Oct. 2018.
- [23] X. Yang, Y. Ye, X. Li, R. Y. K. Lau, X. Zhang, and X. Huang, "Hyperspectral image classification with deep learning models," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 9, pp. 5408–5423, Sep. 2018.
- [24] M. E. Paoletti, J. M. Haut, R. Fernandez-Beltran, J. Plaza, A. Plaza, J. Li, and F. Pla, "Capsule networks for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 4, pp. 2145–2160, Apr. 2019.
- [25] Y. Chen, K. Zhu, L. Zhu, X. He, P. Ghamisi, and J. A. Benediktsson, "Automatic design of convolutional neural network for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 9, pp. 7048–7066, Sep. 2019.
- [26] X. Cao, F. Zhou, L. Xu, D. Meng, Z. Xu, and J. Paisley, "Hyperspectral image classification with markov random fields and a convolutional neural network," *IEEE Trans. Image Process.*, vol. 27, no. 5, pp. 2354–2667, May 2018.
- [27] G. Cheng, Z. Li, J. Han, X. Yao, and L. Guo, "Exploring hierarchical convolutional features for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 11, pp. 6712–6722, Nov. 2018.
- [28] A. J. X. Guo and F. Zhu, "A CNN-based spatial feature fusion algorithm for hyperspectral imagery classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 9, pp. 7170–7181, Sep. 2019.
- [29] M. E. Paoletti, J. M. Haut, R. Fernandez-Beltran, J. Plaza, A. J. Plaza, and F. Pla, "Deep pyramidal residual networks for spectral–spatial hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 2, pp. 740–754, Feb. 2019.
- [30] H. Zhang, Y. Li, Y. Jiang, P. Wang, Q. Shen, and C. Shen, "Hyperspectral classification based on lightweight 3-D-CNN with transfer learning," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 8, pp. 5813–5828, Aug. 2019.
- [31] L. Fang, G. Liu, S. Li, P. Ghamisi, and J. A. Benediktsson, "Hyperspectral image classification with squeeze multibias network," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 3, pp. 1291–1301, Mar. 2019.
- [32] L. Zhu, Y. Chen, P. Ghamisi, and  n Atli Benediktsson, "Generative adversarial networks for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 9, pp. 5046–5063, Sep. 2018.
- [33] J. Feng, H. Yu, L. Wang, X. Cao, X. Zhang, and L. Jiao, "Classification of hyperspectral images based on multiclass spatial–spectral generative adversarial networks," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 8, pp. 5329–5343, Aug. 2019.
- [34] H. Lee, H. Kwon, and W. Kim, "Generating hard examples for pixel-wise classification," *IEEE J. Sel. Top. Appl. Earth Observ. Remote Sens.*, vol. 14, pp. 9504–9517, 2021.
- [35] J. M. Haut, M. E. Paoletti, J. Plaza, J. Li, and A. Plaza, "Active learning with convolutional neural networks for hyperspectral image classification using a new bayesian approach," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 11, pp. 6440–6461, Nov. 2018.
- [36] W. Zhao and S. Du, "Spectral–spatial feature extraction for hyperspectral image classification: A dimension reduction and deep learning approach," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 8, pp. 4544–4554, Aug. 2016.
- [37] N. He, M. E. Paoletti, J. M. Haut, L. Fang, S. Li, A. Plaza, and J. Plaza, "Feature extraction with multiscale covariance maps for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 2, pp. 755–769, Feb. 2019.
- [38] L. Shu, K. McIsaac, and G. R. Osinski, "Hyperspectral image classification with stacking spectral patches and convolutional neural networks," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 10, Oct. 2018.
- [39] 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.
- [40] L. Windrim, A. Melkumyan, R. J. Murphy, A. Chlingaryan, and R. Ramakrishnan, "Pretraining for hyperspectral convolutional neural network classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 5, pp. 2798–2810, May 2018.
- [41] C. Deng, Y. Xue, X. Liu, C. Li, and D. Tao, "Active transfer learning network: A unified deep joint spectral–spatial feature learning model for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 3, pp. 1741–1754, Mar. 2019.
- [42] H. Lee and H. Kwon, "Contextual deep CNN based hyperspectral classification," in *Proc. IGARSS*, Jul. 2016, pp. 3322–3325.
- [43] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. CVPR*, Jun. 2015, pp. 1–9.
- [44] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. NeurIPS*, Dec. 2012, pp. 1097–1105.
- [45] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," arXiv:1207.0580, 2012. [Online]. Available: <https://arxiv.org/1207.0580>
- [46] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. ICML*, Jul. 2015, pp. 448–456.
- [47] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Doll r, "Focal loss for dense object detection," in *Proc. ICCV*, Oct. 2017, pp. 2999–3007.
- [48] I. Kokkinos, "UberNet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory," in *Proc. CVPR*, Jun. 2017, pp. 6129–6138.
- [49] S. Eum*, H. Lee*, H. Kwon, and D. Doermann, "IOD-CNN: Integrating object detection networks for event recognition," in *Proc. ICIP*, Sep. 2017, pp. 875–879, (* indicates authors who contributed equally.).
- [50] H. Lee, S. Eum, and H. Kwon, "DOD-CNN: Doubly-injecting object information for event recognition," in *Proc. ICASSP*, May 2019.
- [51] S. Eum, C. Reale, H. Kwon, C. Bonial, and C. Voss, "Object and text-guided semantics for CNN-based activity recognition," in *Proc. ICASSP*, May 2019, pp. 1458–1462.
- [52] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," in *Proc. ICML*, Jul. 2021.
- [53] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *Proc. ICML*, Jun. 2020, pp. 1597–1607.
- [54] J.-B. Grill, F. Strub, F. Altch , C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar, B. Piot, k. kavukcuoglu, R. Munos, and M. Valko, "Bootstrap your own latent - a new approach to self-supervised learning," in *Proc. NeurIPS*, Dec. 2020, pp. 21 271–21 284.
- [55] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin, "Unsupervised learning of visual features by contrasting cluster assignments," in *Proc. NeurIPS*, Dec. 2020, pp. 9912–9924.
- [56] X. Chen and K. He, "Exploring simple siamese representation learning," in *Proc. CVPR*, Jun. 2021, pp. 15 750–15 758.
- [57] X. Chen*, S. Xie*, and K. He, "An empirical study of training self-supervised vision transformers," in *Proc. ICCV*, Oct. 2021, pp. 9640–9649, (* indicates authors who contributed equally.).
- [58] H. Lee, S. Eum, and H. Kwon, "S-DOD-CNN: Doubly injecting spatially-preserved object information for event recognition," in *Proc. ICASSP*, May 2020.
- [59] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. AISTATS*, May. 2010, pp. 249–256.
- [60] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proc. ICCV*, Dec. 2015, pp. 1026–1034.
- [61] A. Shrivastava, A. Gupta, and R. Girshick, "Training region-based object detectors with online hard example mining," in *Proc. CVPR*, Jul. 2016, pp. 761–769.
- [62] N. Audebert, "DeepHyperX," <https://github.com/nshaud/DeepHyperX>. git, 2019.
- [63] A. Boulch, N. Audebert, and D. Dubucq, "Autoencodeurs pour la visualisation d'images hyperspectrales," in *GRETSI*, 2017, pp. 1–4.
- [64] W. Hu, Y. Huang, L. Wei, F. Zhang, and H. Li, "Deep convolutional neural networks for hyperspectral image classification," *J. of Sens.*, pp. 1–12, 2015.

- [65] 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.
- [66] V. Sharma, A. Diba, T. Tuytelaars, and L. V. Gool, "Hyperspectral CNN for image classification & band selection, with application to face recognition," in *Technical Report*, 2016, pp. 1–14.
- [67] Y. Luo, J. Zou, C. Yao, T. Li, and G. Bai, "HSI-CNN: A novel convolution neural network for hyperspectral image," in *Proc. ICALIP*, Aug. 2018, pp. 464–469.
- [68] A. B. Hamida, A. Benoit, P. Lambert, and C. B. Amar, "3-D deep learning approach for remote sensing image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 8, pp. 4420–4434, Aug. 2018.
- [69] B. Liu, X. Yu, P. Zhang, X. Tan, A. Yu, and X. Xue, "A semi-supervised convolutional neural network for hyperspectral image classification," *Remote Sens. Letters*, vol. 8, no. 9, pp. 839–848, May 2017.
- [70] Z. Zhong, J. Li, Z. Luo, and M. Chapman, "Spectral–spatial residual network for hyperspectral image classification: A 3-D deep learning framework," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 2, pp. 847–858, Feb. 2018.
- [71] Y. Li, H. Zhang, and Q. Shen, "Spectral—spatial classification of hyperspectral imagery with 3D convolutional neural network," *Remote Sens.*, vol. 9, no. 1, Jan. 2017.
- [72] M. He, B. Li, and H. Chen, "Multi-scale 3D deep convolutional neural network for hyperspectral image classification," in *Proc. ICIP*, Sep. 2017, pp. 3904–3908.
- [73] S. K. Roy, G. Krishna, S. R. Dubey, and B. B. Chaudhuri, "HybridSN: Exploring 3-D–2-D CNN feature hierarchy for hyperspectral image classification," *IEEE Geosci. Remote Sens. Lett.*, vol. 17, no. 2, pp. 277–271, Feb. 2020.