Domain Adaptors for Hyperspectral Images

Gustavo Pérez
University of Massachusetts Amherst
Email: gperezsarabi@umass.edu

Subhransu Maji
University of Massachusetts Amherst
Email: smaji@cs.umass.edu

***Abstract*—We consider the problem of adapting a network trained on three-channel color images to hyperspectral images with a large number of channels. We propose domain adaptors that map the input to be compatible with a network trained on large-scale color image datasets such as ImageNet, to enable learning on small hyperspectral datasets where training a network from scratch may not be effective. We investigate architectures and strategies for training adaptors and evaluate them on a benchmark consisting of multiple datasets. We find that simple schemes such as linear projection or subset selection are often the most effective, but can lead to a loss in performance in some cases. We also propose a novel multi-view adaptor where of the inputs are combined in an intermediate layer of the network in an order-invariant manner that provides further improvements. We present experiments by varying the number of training examples in the benchmark to characterize the accuracy and computational trade-offs offered by these adaptors.**

I. Introduction

Transferring deep networks trained on large datasets of color images has been a key to their success on visual recognition [49], [51], [55]. However, the effectiveness of the transfer depends in part on how related the source and target domains are. For example, models trained on Internet images may not be as effective on recognizing medical or astronomy images. A further challenge arises when transferring to heterogeneous domains where some architectural modification to the network is necessary for it to process the input. This paper studies this problem by designing a *domain adaptor network* that can be plugged in before a color image network to process hyperspectral images consisting of different number of channels. These schemes are illustrated in Fig. 1.

Our problem is motivated by the fact that hyperspectral domains lack pretrained networks that can serve as general-purpose feature extractors. Thus one might benefit from architectural innovations and datasets in the domain of color images which are readily available in "model zoos" in modern libraries. While the literature contains several schemes for transferring pretrained networks from color to hyperspectral domains, a systematic evaluation is lacking. Our *first contribution* is therefore a benchmark of six hyperspectral datasets (§ III) divided into two groups as seen in Fig. 2. The first contains three remote sensing datasets: LEGUS [5], So2Sat LCZ42 [52], and EuroSAT [18]. The second modifies the images in Caltech-UCSD birds [45], FGVC aircraft [33], and Stanford cars [28] datasets by synthetically expanding the channels. The synthetic datasets allow us to measure the effectiveness of transfer by comparing it with the performance on the unmodified color images and control the amount of domain shift by varying the number of channels. Despite their simplicity these datasets reveal some difficulties in transfer learning. For example, permuting the color channels makes the transfer significantly less effective on the CUB dataset. We analyze this phenomenon and it's implications in § V-C.

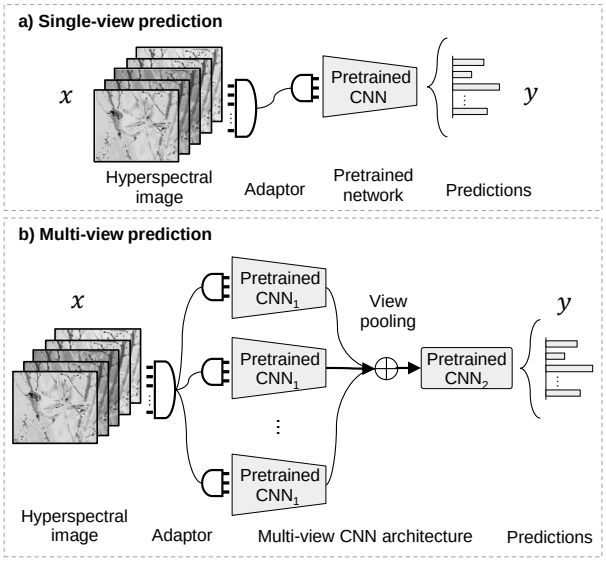

Fig. 1: **Single and multi-view adaptors. (a)** An adaptor maps multiple channels to three, making it compatible with a pretrained color network. **(b)** Multiple adaptors generate views of the input which are processed through a shared network.

Our *second contribution* is an evaluation of common schemes in the literature [3], [4], [8], [21], [29], [48]. These include: (a) a linear projection; (b) selecting a subset; (c) a multi-layer network; (d) "inflating" the first layer of the network. We also investigate techniques for pre-training adaptors in an unsupervised manner. Finally, we propose a novel multi-view scheme that generates a prediction by aggregating information across "views" of the input as illustrated in Fig. 1b.

Our experiments suggest that schemes such as linear projection or subset selection are often the most effective. Deeper adaptors and unsupervised training of adaptors offer little additional benefit highlighting the difficulty of adapting a network layer far from the output layer during transfer. Multi-view adaptors can be used with any baseline adaptor and provide consistent improvements when the domain has a large number of channels. The multi-view adaptor can be thought of as a light-weight ensemble, as these networks require few additional parameters compared to a separate network trained

on each view. Notably, all of these schemes are significantly better than training a custom network from scratch underlying the need to study techniques to adapt color image networks to heterogeneous domains.

To summarize, our contributions are:

1) We propose a benchmark for investigating the effectiveness of transfer learning on hyperspectral images (§ III). We analyze the effectiveness of adaptor networks that map multi-channels to a three-channel image compatible pretrained network (§ IV and § V).
2) We propose a novel multi-view scheme that is significantly more lightweight than ensembles but offer consistent gains. On the technical side, we propose a regularization scheme that encourages diversity across views which provides further benefits (§ IV-A).
3) We illustrate the difficulty in transfer learning even when domain shifts are simple (e.g., color channels are permuted), suggesting the need to investigate techniques beyond fine-tuning (§ V).

The training and evaluation source code is publicly available at https://github.com/gperezs/hyperspectral_domain_adaptors.

II. Related work

A. Transfer learning

Transfer learning is an effective strategy for learning from a few examples. Within the context of deep networks, this is typically done by training a network on a large, labeled dataset (e.g., ImageNet [10]) and fine-tuning its parameters on the downstream task after adding task-specific layers. Schemes vary from training a subset of layers [51], or parameters (e.g., batch norm statistics [22]), to adding a secondary network to parameterize the changes [20] and custom architectures where parts of the network are initialized (e.g., B-CNN [30], Faster R-CNN [39], etc.). For robustness to domain shifts techniques based on the aligning the statistics of deep network features across domains have been proposed [19], [32], [46]. When paired data across domains is available one might learn the mapping directly, e.g., using a GAN [12]) or apply cross-modal training techniques (e.g., CQD [43], data distillation [38]) to improve transfer. However, this requirement is rarely met in practice. Though less studied, somewhat surprisingly transfer learning across heterogeneous domains can be effective. For example, 3D shapes can be recognized by rendering them as images, or audio by rendering them as spectrograms, and applying image-based CNNs. Our work aims to design such adaptors for hyperspectral data.

B. Hyperspectral image datasets

Hyperspectral images are common in remote sensing (e.g., RADARs, Satellites, Telescopes) and medical imaging (e.g., [7], [11]) as they reveal properties not easily seen in color images. For instance, remote sensing data acquired from earth observation satellites are routinely used for land cover classification, infrastructure planning, and population assessment [17], [25], [37], [47], [50], [54]. The Hubble Space Telescope (HST) captures data through a variety of filters, each passing specific wavelengths of light, which are used in various scientific analysis. Tasks in these domains often lack large, labeled datasets and transfer learning from color images provides a compelling alternative. However, the differences in domains and their structure poses a barrier.

C. Hyperspectral image classification

Deep networks can be also be used in a straightforward manner when training a network from *scratch* and have been successfully applied to different tasks [2], [15], [24], [34], [36], [53]. However, the lack of large-scale labeled datasets makes training from scratch less effective than say transfer learning via color image networks. Several schemes are possible to account for the different number of channels. A common strategy is to ensemble networks, each trained on a subset of channels. While this has been applied successfully for some tasks (e.g., combining flow and appearance for video [41], or classifying astronomy images [48]), strategies for selecting subsets and combining them vary. Ensembles also increase the computational complexity and model size. Another approach is to *manually select* a subset of channels based on domain knowledge [3], [4], [9], [31]. The *dimensionality reduction* can also be learned using PCA [23]. While this is often used as a visualization tool, it is also been shown to be effective for transfer (e.g., classifying RADAR data [35]). Another approach is based on *inflating* the network by replicating the filters in the first layer. This strategy has been effectively applied to transfer color network to spatio-temporal [8], [21] and 3D domains [29]. Even when the number of channels are fewer, one can benefit from expanding it to a color image. For example, depth data rendered as color images (e.g., HHA encoding [14]) has been shown to lead to a better performance with pretrained color networks.

III. A hyperspectral classification benchmark

Our benchmark consists of six classification datasets with varying number of channels and classes as illustrated in Fig. 2. The datasets are divided in two groups called *synthetic* and *realistic*. The synthetic group consists of RGB images synthetically expanded to 5 or 15 channels, while the realistic group consists of hyperspectral datasets from remote sensing applications. They are described in detail next.

A. Synthetic datasets

We generate synthetic versions of the RGB images by mapping each color $\in \mathbb{R}^3$ to a vector in $\mathbb{R}^k$ with $k > 3$. We use k-means to cluster the color values of images in each dataset to find k centers. Then each channel is computed as:

$$f[x, y, i] = \exp\big(-||f_0[x, y] - c_i||_2^2\big), \quad (1)$$

where $f_0[x, y] \in \mathbb{R}^3$ is the RGB value of the pixel at location (x, y), and $c_i \in \mathbb{R}^3$ is the i^{th} center calculated with k-means. We now describe the synthetic datasets.

- **CUB-5 and CUB-15.** The CUB-200-2011 [45] dataset contains 11788 images divided in 5994 images for training

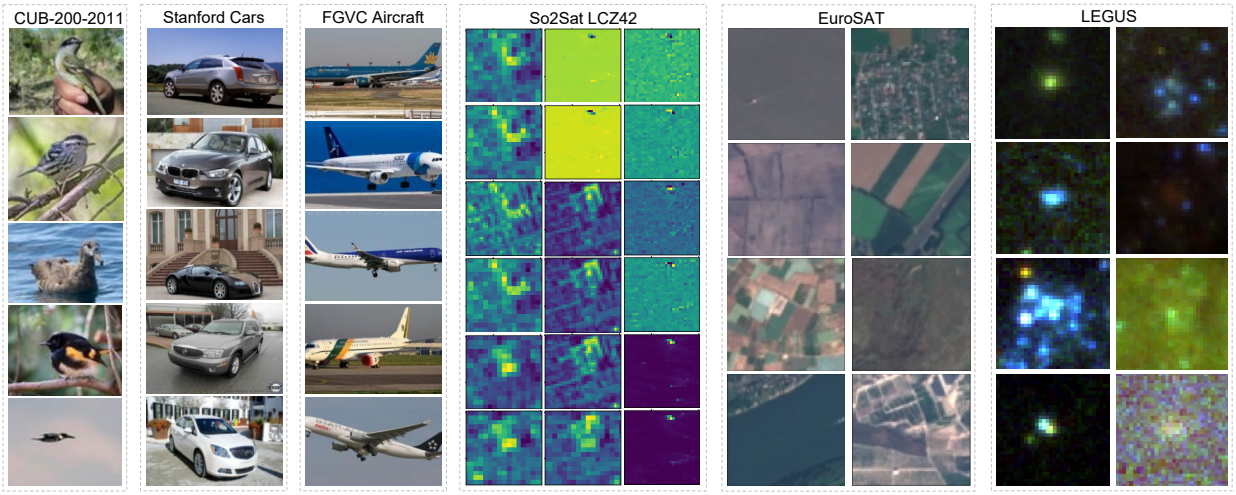

Fig. 2: **Datasets.** Sample images of the six datasets in the proposed benchmark. In the case of synthetic datasets (CUB-200-2011, Stanford Cars, and FGVC Aircraft) we show the original images. For So2Sat LCZ42, we show the 18 bands of a single sample. For EuroSAT, we show the RGB version of the images also publicly available. For LEGUS, we show RGB images using the conversion described at the end of § V-C.

and 5794 images for testing. The dataset contains images of 200 bird species. We generate the CUB-5 and CUB-15 dataset by expanding each color image to 5 and 15 channels.

- **Cars-5 and Cars-15.** The Stanford cars [28] dataset contains 16185 images of 196 car models. The data is split into 8144 training images and 8041 testing images, where each class has been divided following a 50-50 split. We generate the Cars-5 and Cars-15 dataset by expanding the channels.
- **Aircraft-5 and Aircraft-15.** The FGVC aircraft [33] dataset contains 10000 images of aircraft, with 100 images for each of 100 different aircraft model variants. The benchmark proposes a train-validation-test split of roughly the same number of images. In all our experiments, we use only the image-level variant annotations and the trainval-test splits of 6667 and 3333 images respectively. Aircraft-5 and Aircraft-15 were generated in a similar manner.

B. Realistic datasets

We choose the following datasets with different domain shifts from natural images and also consider a reduced version of these datasets with 1000 training images (So2Sat^S, EuroSAT^S, and LEGUS^S) to study the adaptor networks in the few-shot setting.

- **So2Sat LCZ42.** The Social Media to EO Satellites Local Climate Zones 42 (So2Sat LCZ42) [52] is a dataset consisting of co-registered synthetic aperture radar and multispectral optical image patches acquired by the Sentinel-1 and Sentinel-2 remote sensing satellites. The task is to classify the images into 1 of 17 local climate zones (LCZ). For the complete version of the dataset we use a train-test split of 10000 images with 18 spectral bands each.
- **EuroSAT.** The Dataset for Land Use and Land Cover Classification (EuroSAT) [18] is based on Sentinel-2 satellite images covering 13 spectral bands. The dataset consists of 27000 labeled and geo-referenced images with annotations of 10 classes of terrain.
- **LEGUS.** The Legacy ExtraGalactic UV Survey (LEGUS) [5] consists of 50 galaxies at distances between 3.5 Mpc and 16 Mpc, HST in five bands spanning the ultraviolet to infrared spectrum. The task is to classify the star-clusters within each galaxy into four classes. The first three classes correspond to different morphological properties and class 4 refers to non-cluster objects like foreground stars, background galaxies, or artifacts.

IV. Adaptors for hyperspectral images

The architecture of a pretrained color network has to be modified to process a multi-channel image. Bellow we describe the four choices of adaptor architectures.

- **Linear projection.** A linear adaptor maps a hyperspectral image $x \in \mathbb{R}^{n\times m\times k}$ to a view $v(x) \in \mathbb{R}^{n\times m\times 3}$ with a linear projection from k channels. The projection is implemented as a single convolutional layer with 3 filters of size $1\times 1\times k$. These are initialized randomly or using PCA, and jointly trained with the network during fine-tuning.
- **Subset selection.** The mapping is achieved by selecting three channels out of the k input channels to obtain a view $v(x) \in \mathbb{R}^{n\times m\times 3}$. We select the channels randomly.
- **Multi-layer adaptor.** A multi-layer adaptor is a multi-layer neural network that maps the hyperspectral image $x \in \mathbb{R}^{n\times m\times k}$ to a view $v(x) \in \mathbb{R}^{n\times m\times 3}$. The adaptor is attached to the pretrained network and trained jointly from scratch or initialized using unsupervised training.
- **Inflated network.** We inflate the network by replacing the filter weights of the first convolutional layer by first averaging the filter weights across the channel dimension and replicating it as many times as the number of channels in the input. Let $\theta \in \mathbb{R}^{h\times w\times 3\times n}$ be the n filters in

the first layer of network. This is inflated to parameters $\hat{\theta} \in \mathbb{R}^{h \times w \times k \times n}$ with each filter $\hat{\theta}^j$ given by:

$$\hat{\theta}^j = \underbrace{\begin{pmatrix} \text{mean}(\theta^j) & \text{mean}(\theta^j) & \cdots & \text{mean}(\theta^j) \end{pmatrix}}_{k \text{ times}}$$

A. Multi-view adaptors

All adaptors except the inflated network reduce the number of channels before feeding it to the pretrained network, which may lead to a loss of information. An alternate strategy is to train an ensemble, where different models are trained on a subset of channels and then combined to produce the final prediction. However this scheme leads to a increase in the parameters and also impacts the speed of classification.

To address this issue we propose a *novel* multi-view scheme that combines information from multiple views of the hyperspectral image using a shared network. Each view is obtained using a different adaptor (e.g., different subsets, or different linear projections). Instead of training a separate network, the multi-view scheme passes each view through the first section of the network (CNN_1 in Fig. 1b). Then the activations are aggregated using a *view pooling layer* using an orderless set aggregation scheme. Finally, the aggregated activations are passed through the remaining section of the network (CNN_2 in Fig. 1b) to produce the final output. The scheme allows the view pooling layer to be added at any layer in the network. Formally a set of views $v_i(x) \in \mathbb{R}^{n \times m \times 3}$, $i = 1, \ldots, N$ are obtained from the input x using any adaptor scheme described earlier. The output of the multi-view model is

$$y = \text{CNN}_2\Big(\Phi\big\{\text{CNN}_1(v_i(x))\big\}\Big), i = 1, \ldots, N, \qquad (2)$$

where $\Phi\{\cdot\}$ is an orderless set aggregation function such as the mean or max. We use max in our experiments.

Unlike ensembles parameters are shared across views — the only view specific parameters are in the adaptor layer which are significantly fewer than the number of network parameters ($< 0.001\%$). The memory and computational cost are increased depending on which layer is view pooled. For example early layer view-pooling leads to smallest increase in cost. While last-layer view pooling is as slow as an ensemble, but still leads to parameter savings. We present these details in § V-A. Multi-view networks exploit the ability of deep networks to combine multiple sources of information which has been exploited for tasks such as 3D shape classification from multiple views [42].

V. Experiments

We first describe our experimental setup and implementation details (§ V-A). Then we present our results across datasets in the benchmark comparing the performance of different single-view and multi-view adaptors, and ablation studies varying the number of views and initialization schemes (§ V-B). Finally we analyze the limits of adaptors by analyzing the effect of domain shifts on the synthetic datasets (§ V-C).

A. Implementation details

Fine-tuning details. We use an ImageNet pretrained VGG-D [40], ResNet18, and ResNet50 [16] networks in our experiments. For transfer learning we replace the last fully-connected layer to match the number of classes and train the network jointly with the input adaptor using Adam [26] optimizer to minimize a multi-class cross-entropy loss. We use the image-level annotations on the train-test splits as described in § III. We run our experiments with synthetic datasets for 30 epochs using a learning rate of 1E-04, a batch size of 64 images, and data augmentation with horizontal flips. We decrease the learning rate by a factor of 10 every 7, 15, and 10 epochs for birds, cars, and aircraft datasets respectively. For realistic datasets, we train the models for 10, 6, and 15 epochs for So2Sat, EuroSAT, and LEGUS respectively using a learning rate of 1E-04 (with a decrease by a factor of 10 every 5 epochs for LEGUS, and 4 epochs for EuroSAT and So2Sat). We use a batch size of 32 and data augmentation with random flips. In particular for LEGUS, which exhibit rotational invariance, we use random rotations.

Training from scratch. For each dataset we train the networks from scratch by modifying the first layer to match the number of input channels and the last layer to match the number of classes. We run training for 17$\times$ epochs than fine-tuning based on [27]. We use a learning rate of 1E-04, a batch size of 64, and data augmentation with horizontal flips. We decrease the learning rate by a factor of 10 every 14, 30, and 20 epochs for birds, cars, and aircraft datasets respectively, and every 10 epochs for LEGUS and 8 epochs for So2Sat and EuroSAT.

Multi-layer adaptors. The multi-layer adaptor has 4 convolutional layers, each one with 16 filters of size 3×3, batch normalization, and ReLU activations with no stride or pooling layers. The parameters are either initialized randomly or learned as an auto-encoder using the training samples. When pretraining the adaptor, we use Adam optimizer to minimize the mean squared error between the prediction and the hyperspectral input image. We train the adaptor for 10 epochs using a learning rate of 1E-3 and a batch size of 64.

Multi-view adaptors. We perform experiments using 2 and 5 views per image. Also, in our multi-view setup, the effective batch size is smaller since we are using shared parameters across all the views, so we apply a linear scaling rule (as proposed in [13]) to the learning rate, learning rate schedule, and the number of training epochs. We place the view pooling layer at conv_{5_3} layer in the VGG-D, conv_{5_2} in the ResNet18, and conv_{5_3} in the ResNet50 network. We report results of the multi-view experiments using linear projection and subset selection. For datasets with large number of channels we also experiment with larger number of views.

Adaptor training. During training we found it helpful to increase the learning rate of the linear and multi-layer adaptors by a factor of 10 compared to the rest of the network. Without this the weights do not change from their initial value. We obtain an average of 3.7% increase in performance by increasing the learning of the adaptors for the synthetic

Performance using VGG-D network

Dataset	From scratch	Single-view				Multi-view			
		Linear adaptor	Inflated network	Multi-layer adaptor	Subset selection	Subset selection 2	Subset selection 5	Linear adaptor 2	Linear adaptor 5
Synthetic									
CUB-5	0.5 ± 0.0	38.4 ± 3.1	30.5 ± 1.1	41.3 ± 2.8	47.1 ± 4.0	45.0 ± 1.7	49.8 ± 0.5	52.7 ± 0.6	**55.1 ± 0.3**
CUB-15	0.5 ± 0.0	41.2 ± 0.9	26.0 ± 2.8	45.4 ± 5.1	45.9 ± 5.0	49.9 ± 2.7	**56.5 ± 1.0**	**57.3 ± 0.5**	**56.7 ± 0.9**
Cars-5	2.2 ± 0.8	70.5 ± 2.1	70.2 ± 1.4	74.4 ± 1.7	72.6 ± 2.0	75.4 ± 0.9	**76.7 ± 0.5**	**76.7 ± 0.4**	**76.8 ± 0.5**
Cars-15	2.7 ± 0.4	70.7 ± 3.3	68.7 ± 3.4	70.1 ± 2.3	72.2 ± 1.7	75.8 ± 0.4	77.1 ± 0.3	77.1 ± 0.2	**78.1 ± 0.4**
Aircraft-5	1.0 ± 0.0	79.6 ± 0.8	76.6 ± 0.9	79.5 ± 0.5	79.2 ± 1.0	**80.8 ± 0.9**	**81.5 ± 0.3**	81.1 ± 0.4	**81.8 ± 0.2**
Aircraft-15	1.0 ± 0.0	78.4 ± 0.6	77.6 ± 0.9	79.1 ± 1.2	**79.8 ± 2.2**	**81.3 ± 1.0**	**81.6 ± 0.5**	**81.4 ± 0.5**	**81.3 ± 0.4**
Realistic									
So2Sat	50.3 ± 0.8	52.8 ± 0.3	51.8 ± 0.3	52.8 ± 0.7	51.6 ± 3.5	55.6 ± 1.0	**57.4 ± 0.4**	**57.0 ± 0.4**	**57.2 ± 0.5**
So2Sat^S	34.5 ± 0.7	45.1 ± 0.5	40.8 ± 0.8	40.8 ± 3.1	43.8 ± 4.9	49.8 ± 1.2	**51.4 ± 0.3**	45.2 ± 0.7	48.6 ± 0.8
EuroSAT	95.7 ± 0.2	97.1 ± 0.3	96.7 ± 0.3	97.7 ± 0.2	96.6 ± 0.7	**97.4 ± 0.4**	**97.7 ± 0.2**	**97.4 ± 0.2**	**97.5 ± 0.2**
EuroSAT^S	75.2 ± 2.6	88.4 ± 0.5	83.6 ± 1.8	92.8 ± 0.7	86.6 ± 3.6	91.6 ± 2.2	93.7 ± 0.5	**94.4 ± 0.2**	**94.4 ± 0.2**
LEGUS	51.9 ± 0.5	63.2 ± 0.5	61.4 ± 0.4	62.8 ± 0.3	61.1 ± 1.4	64.2 ± 0.4	**65.3 ± 0.5**	64.2 ± 0.4	63.9 ± 0.6
LEGUS^S	27.2 ± 5.5	54.9 ± 0.8	51.8 ± 1.3	53.1 ± 1.9	51.3 ± 2.9	55.0 ± 0.6	**59.3 ± 0.6**	55.7 ± 1.8	**60.0 ± 1.6**

^{S} Smaller version of the dataset using 1000 training samples.

TABLE I: **Results using domain adaptors.** Accuracy (%) for single and multi-view adaptors on our benchmark with an ImageNet pretrained VGG-D network. The top group indicates the *synthetic* datasets while the bottom group represents *realistic* datasets. The best results using single-view adaptors are shown in green and best overall results are shown in bold blue.

datasets, and an average of 5.8% by increasing the learning of the adaptors for the realistic datasets.

Diversity regularization. In the case of multi-view with learnable adaptors (e.g. linear projection), we propose a regularization to enforce the adaptors to learn to produce independent views from each other with the goal of increasing the amount of information per number of views. Formally, given the k views $v(x) \in \mathbb{R}^{n \times m \times 3k}$ produced by a multi-view adaptor, its gram matrix $G \in \mathbb{R}^{3k \times 3k}$ is produced by $G = \hat{v}\hat{v}^T$, where $\hat{v} \in \mathbb{R}^{3k \times nm}$ is a reshaped version of the views. Our regularization for a batch of N images is $R = \alpha \sum_i^N ||G_i||_p$, where p is the order of the norm and α is a scaling factor. We use the spectral norm ($p = 2$) and a scaling factor of $\alpha =$ 1E-2. The regularization leads to an average increase of 1.3% for synthetic datasets and 1.1% for realistic datasets when used with multi-view linear adaptors. All multi-view linear adaptor results use this regularization.

Baseline on synthetic datasets. For the synthetic datasets we fine-tune a pretrained network using the original RGB images. This provides a reference upper bound on the performance the model can achieve and is shown in Tab. IV in APPENDIX B.

B. Results and discussion

Tab. I shows the results on all datasets using VGG-D. Tab. III in APPENDIX A of the supplementary material shows the performance across all datasets using different adaptors and pretrained networks. Below we summarize these.

Training from scratch is usually not effective. Models trained from scratch with synthetic datasets and VGG-D network yield low accuracy. The ResNet18 and ResNet50 networks perform better than VGG-D, but is still quite low compared to using adaptors. This is not surprising as the tasks represent fine-grained classification problems which are quite challenging. On the realistic datasets the performance is better, though when small amounts of data is available the performance is much lower. The results of this approach are shown in the leftmost column of Tab. I and Tab. III in APPENDIX A of the supplementary material.

Using initializations for learnable adaptors. For learnable adaptors (linear and multi-layer), we consider random initialization and unsupervised initialization. This corresponds to PCA for the linear projection, or as an auto-encoder of the input with a bottleneck with three channels. After training the decoder is discarded and the encoder acts as an adaptor. As shown in APPENDIX C–Tab. V, these initializations do not lead to improvements over a random initialization. We suspect that since the adaptor is trained jointly with the rest of the network, the initialization does not impact the performance.

Simple adaptors are effective. When we use VGG-D, simple adaptors like random subset selection and linear adaptors yield the best results. With the exception of EuroSAT, random subset selection and linear projection outperforms all the other single-view adaptors. Best results using single-view adaptor networks are shown in green color in Tab. I.

Multi-view adaptors are effective. Multi-view adaptor networks provide improvements over single-view on all datasets. For CUB synthetic datasets, we get improvements of up to 11.4% from the best performing single-view adaptor networks. Using multi-view adaptors also yields an increase of accuracy of up to 4.9% in the cars dataset and 2.8% in aircraft. On realistic datasets, improvements of up to 2.3% from the best performing single-view adaptors in So2Sat, 0.2% in EuroSAT, and 2.1% in LEGUS are observed. The best overall results for each dataset are shown in bold blue color in Tab. I. Fig. 3 shows accuracies using multi-view adaptors with up to 10 views for So2Sat and CUB-15. The performance does not

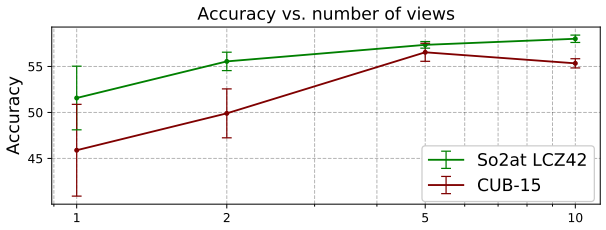

Fig. 3: **Accuracy vs. number of views.** Performance of subset multi-view adaptors varying the number of views.

increase significantly beyond 5 views.

Shared network for multi-view is effective. Combining the information of multiple views increases the performance from the best performing single-view approaches (∼11% in some cases). This result occurs despite using a shared architecture to process all the views. The shared network for multi-view adds a similar increase in computational cost as having separate networks for each view compared to single-view approaches. However, the shared network adds a negligible number of parameters to the model.

C. Domain shift and transferability

Since the synthetic datasets contain the same information as the original color images as the mapping is invertible, the performance difference between the oracle and synthetic dataset indicates the difficulty of transfer using adaptors (see § V-A). On Cars and Aircrafts, adaptors are within 1-2% of the oracle performance. However, on Birds the performance is about 15.3% lower (72% → 56.7%). This difference is likely as Cars and Aircrafts can be recognized using shape features which are largely preserved in the mapping (Fig. 2). This is also indicated by the good performance of the inflated network on these datasets which effectively collapse the color channels. However, color is important for bird identification and indeed, the inflated networks do poorly on this dataset.

However, one would expect that fine-tuning might be able to "invert" the mapping that was used to generate the synthetic images. But this appears not to be the case. As a further experiment we fine-tuned the network on multiple versions of the CUB dataset where the color channels are permuted (Tab. II). The performance decreases sharply. The reason for this is that tasks-specific information in color channels is not uniform in natural images (e.g., there are lot more "red blobs" than "blue blobs"), which encourages the ImageNet model to learn certain feature detectors better than others (e.g., a "red blob detector"). Thus, mapping the red channel to blue may make a "red blob" feature "invisible" to the network. While fine-tuning can help, there could be a limitation due to the phenomenon of critical periods [1].

These experiments suggest that there is a limit to transferability using adaptors and the final performance (in the limit of large training dataset) may be worse than training from scratch when large, labeled datasets are available. It also suggests that alternate optimization techniques (e.g., discrete search over permutations) might be more effective for learning adaptors than gradient-based optimization (See APPENDIX D).

Dataset	**RGB**	Channel permutation		Grayscale		Low resolution	
VGG-D							
Birds	**72.0**	65.4	(11.4%)	54.5	(24.3%)	58.4	(18.9%)
Cars	**78.2**	77.7	(1.0%)	76.5	(2.2%)	61.0	(22.0%)
Aircraft	**81.1**	82.0	(-1.3%)	80.0	(1.3%)	72.4	(10.7%)
ResNet18							
Birds	**70.5**	65.9	(7.7%)	50.9	(27.8%)	56.1	(20.4%)
Cars	**80.6**	79.8	(1.2%)	78.8	(2.3%)	68.0	(15.7%)
Aircraft	**76.9**	76.7	(0.3%)	75.2	(2.3%)	68.3	(11.2%)
ResNet50							
Birds	**77.0**	73.4	(5.4%)	61.5	(20.1%)	66.7	(13.4%)
Cars	**86.7**	85.8	(1.2%)	85.5	(1.4%)	76.1	(12.2%)
Aircraft	**84.0**	84.0	(1.4%)	82.2	(2.1%)	77.7	(7.5%)

TABLE II: **Accuracy with image degradations**. Similar to color permutation, grayscale conversion (which preserves shape rather than color) affects the performance of CUB dataset in a more significant way than Cars and Aircraft. On the other hand, low resolution (which preserves color rather than shape) reduces the performance of the three datasets similarly. We show the percentage decrease in parenthesis.

While we do not have a similar "upper bound" in accuracy for the realistic datasets, we include two experiments on EuroSAT and LEGUS to provide a sense of the task difficulty.

EuroSAT provides an RGB version of the images in addition to the 13 spectral bands. We use these to fine-tune the pretrained model and compare the performance using adaptors trained on the 13 channels. We obtain 70.3% accuracy with VGG-D network when training on RGB images compared to 97.7% accuracy on the hyperspectral images. This is no surprise since these hyperspectral channels were chosen for their informativeness on the tasks.

LEGUS. We fine-tune the pretrained network using LEGUS images converted to RGB using a conventional method used to visualize astronomical images. The three channels are constructed by merging the first two and the last two channels as: (1) R $= (\gamma_v V + \gamma_i I)/2$, (2) G $= (\gamma_b B)$, (3) B $= (\gamma_n NUV + \gamma_u U)/2$, where γ is the inverse gain of the filters used by the HST. We obtain 62.8% accuracy with a VGG-D network using these RGB images. In contrast, when using the spectral bands separately we achieve 65.3% accuracy.

VI. Conclusion

We considered the problem of transferring a three-channel (RGB) network to a hyperspectral domain using adaptors. Simple adaptors based on linear projection and subset selection are effective, but the performance can be low when the number of channels is large. Multi-view adaptors provide consistent gains and are a compelling alternative to ensembles due to the shared model structure. We also highlighted difficulties in transfer with gradient-based training. Future work could investigate techniques from the self- and semi-supervised learning, alternate parameterizations (e.g., using transformers [44]) or search-based optimization to improve transfer.

Acknowledgements. The work is supported in part by NSF grants 2017756, 1749833 and 1815267. The experiments were performed using equipment obtained under a grant from the Collaborative Fund managed by the Mass. Tech. Collaborative.

REFERENCES

[1] A. Achille, M. Rovere, and S. Soatto. *Critical learning periods in deep neural networks*. arXiv preprint arXiv:1711.08856, 2017.

[2] P.H. Barchi, R.R. de Carvalho, R.R. Rosa, R.A. Sautter, M. Soares-Santos, B.A.D. Marques, E. Clua, T.S. Gonc¸alves, C. de S´a-Freitas, and T.C. Moura. *Machine and deep learning applied to galaxy morphology - a comparative study*. Astronomy and Computing, 30:100334, Jan 2020.

[3] J. Bialopetravičius and D. Narbutis. *Deriving star cluster parameters with convolutional neural networks*. Astronomy Astrophysics, 633:A148, Jan 2020.

[4] J. Bialopetravičius, D. Narbutis, and V. Vansevičius. *Deriving star cluster parameters with convolutional neural networks*. Astronomy Astrophysics, 621:A103, Jan 2019.

[5] D. Calzetti, J. Lee, E. Sabbi, and A. Adamo. *Legacy extragalactic uv survey (legus) with the hubble space telescope. i. survey description*. The Astronomical Journal, 149(51):25, 2015.

[6] G. Camps-Valls, D. Tuia, L. Bruzzone, J. A. Benediktsson. *Advances in Hyperspectral Image Classification: Earth monitoring with statistical learning methods*. CoRR, 2013.

[7] O. Carrasco, R. B. Gomez, A. Chainani, and W. E. Roper. *Hyperspectral imaging applied to medical diagnoses and food safety*. In Geo-Spatial and Temporal Image and Data Exploitation III, volume 5097, pages 215 – 221. International Society for Optics and Photonics, SPIE, 2003.

[8] J. Carreira and A. Zisserman. *Quo vadis, action recognition? A new model and the kinetics dataset*. CoRR, abs/1705.07750, 2017.

[9] Z. Cheng, S. Gabriel, P. Bhambhani, D. Sheldon, S. Maji, A. Laughlin, and D. Winkler. *Detecting and Tracking Communal Bird Roosts in Weather Radar Data*. Association for the Advancement of Artificial Intelligence (AAAI), 2020.

[10] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. *ImageNet: A Large-Scale Hierarchical Image Database*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2009.

[11] D. Ferris, R. Lawhead, E. D. Dickman, N. Holtzapple, J. A. Miller, S. Grogan, S. Bambot, A. Agrawal, and M. Faupel. *Multimodal hyperspectral imaging for the noninvasive diagnosis of cervical neoplasia*. Journal of Lower Genital Tract Disease, 5:65–72, 2001.

[12] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. *Generative adversarial nets*. In Advances in neural information processing systems, pages 2672–2680, 2014.

[13] P. Goyal, P. Dollar, R. B. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He. *Accurate, large minibatch SGD: training imagenet in 1 hour*. CoRR, abs/1706.02677, 2017.

[14] S. Gupta, R. Girshick, P. Arbelaez, and J. Malik. *Learning rich features from rgb-d images for object detection and segmentation*. CoRR, 2014.

[15] R. Hausen and B. E. Robertson. *Morpheus: A deep learning framework for the pixel-level analysis of astronomical image data*. The Astrophysical Journal Supplement Series, 248(1):20, May 2020.

[16] K. He, X. Zhang, S. Ren, and J. Sun. *Deep residual learning for image recognition*. CoRR, abs/1512.03385, 2015.

[17] L. He, J. Li, A. Plaza, and Y. Li. *Discriminative low-rank gabor filtering for spectral-spatial hyperspectral image classification*. IEEE Transactions on Geoscience and Remote Sensing, PP:1–15, 12 2016.

[18] P. Helber, B. Bischke, A. Dengel, and D. Borth. *Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification*. CoRR, abs/1709.00029, 2017.

[19] S. Herath, M. Harandi, B. Fernando, and R. Nock. *Min-max statistical alignment for transfer learning*. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2019.

[20] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. de Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly. *Parameter-efficient transfer learning for nlp*. CoRR, 2019.

[21] Y. Huang, Y. Guo, and C. Gao. *Efficient parallel inflated 3d convolution architecture for action recognition*. IEEE Access, 8:45753–45765, 2020.

[22] S. Ioffe and C. Szegedy. *Batch normalization: Accelerating deep network training by reducing internal covariate shift*. CoRR, abs/1502.03167, 2015.

[23] I.T. Jolliffe. *Principal Component Analysis*. Springer Verlag, 1986.

[24] N. M. Khalifa, M. N. Taha, A. Hassanien, and I. M. Selim. *Deep galaxy: Classification of galaxies based on deep convolutional neural networks*. CoRR, 2017.

[25] M. Khodadadzadeh, J. Li, A. Plaza, H. Ghassemian, J. Bioucas-Dias, and X. Li. *Spectral–spatial classification of hyperspectral data using local and global probabilities for mixed pixel characterization*. Geoscience and Remote Sensing, IEEE Transactions on, 52:6298–6314, 10 2014.

[26] D. P. Kingma and J. Ba. *Adam: A method for stochastic optimization*. Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

[27] S. Kornblith, J. Shlens, and Q. V. Le. *Do better imagenet models transfer better?*. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2019.

[28] J. Krause, M. Stark, J. Deng, and L. Fei-Fei. *3d object representations for fine-grained categorization*. In 4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13), Sydney, Australia, 2013.

[29] R. LaLonde, I. Tanner, K. Nikiforaki, G. Z. Papadakis, P. Kandel, C. W. Bolan, M. B. Wallace, and U. Bagci. *Inn: Inflated neural networks for ipmn diagnosis*. Medical Image Computing and Computer Assisted Intervention (MICCAI), 2019.

[30] T. Lin, A. RoyChowdhury, and S. Maji. *Bilinear cnns for fine-grained visual recognition*. In Transactions of Pattern Analysis and Machine Intelligence (PAMI), 2017.

[31] T. Lin, K. Winner, G. Bernstein, A. Mittal, A. Dokter, K. Horton, C. Nilsson, B. Van Doren, A. Farnsworth, F. La Sorte, S. Maji, and D. Sheldon. *MistNet: Measuring historical bird migration in the US using archived weather radar data and convolutional neural networks*. Methods in Ecology and Evolution, 2019.

[32] M. Long, H. Zhu, J. Wang, and M. I. Jordan. *Deep transfer learning with joint adaptation networks*. CoRR, 2017.

[33] S. Maji, J. Kannala, E. Rahtu, M. Blaschko, and A. Vedaldi. *Fine-grained visual classification of aircraft*. Technical report, CoRR, 2013.

[34] G Martin, S Kaviraj, A Hocking, S C Read, and J E Geach. *Galaxy morphological classification in deep-wide surveys via unsupervised machine learning*. Monthly Notices of the Royal Astronomical Society, 491(1):1408–1426, Oct 2019.

[35] A. K. Mishra and B. Mulgrew. *Radar signal classification using pca-based features*. In 2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings, volume 3, pages III–III, 2006.

[36] A. Mittal, A. Soorya, P. Nagrath, and D. J. Hemanth. *Data augmentation based morphological classification of galaxies using deep convolutional neural network*. Earth Science Informatics, 13:601 – 617, 2019.

[37] C. Qiu, X. Tong, M. Schmitt, B. Bechtel, and X. X. Zhu. *Multilevel feature fusion-based cnn for local climate zone classification from sentinel-2 images: Benchmark results on the so2sat lcz42 dataset*. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 13:2793– 2806, 2020.

[38] I. Radosavovic, P. Dollar, R. Girshick, G. Gkioxari, and K. He. *Data distillation: Towards omnisupervised learning*. CoRR, 2017.

[39] S. Ren, K. He, R. Girshick, and J. Sun. *Faster r-cnn: Towards real-time object detection with region proposal networks*. In Advances in Neural Information Processing Systems, volume 28. Curran Associates, Inc., 2015.

[40] K. Simonyan and A. Zisserman. *Very deep convolutional networks for large-scale image recognition*. International Conference on Learning Representations (ICLR), 2015.

[41] K. Simonyan and A. Zisserman. *Two-Stream Convolutional Networks for Action Recognition in Videos*. Advances in Neural Information Processing Systems (NIPS), 2014.

[42] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller. *Multi-view convolutional neural networks for 3d shape recognition*. In Proceedings of the 2015 IEEE International Conference on Computer Vision, ICCV '15, page 945–953, USA, 2015. IEEE Computer Society.

[43] J. Su and S. Maji. Cross quality distillation. CoRR, abs/1604.00433, 2016.

[44] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. *Attention is all you need*. arXiv preprint arXiv:1706.03762, 2017.

[45] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. *The Caltech-UCSD Birds-200-2011 Dataset*. Technical report, 2011.

[46] J. Wang, Y. Chen, W. Feng, H. Yu, M. Huang, and Q. Yang. *Transfer learning with dynamic distribution adaptation*. ACM Transactions on Intelligent Systems and Technology, 11(1):1–25, Feb 2020.

[47] Q. Wang, Z. Meng, and X. Li. *Locality adaptive discriminant analysis for spectral-spatial classification of hyperspectral images*. IEEE Geoscience and Remote Sensing Letters, PP:1–5, 09 2017.

[48] W. Wei, E. A. Huerta, B. C. Whitmore, J. C. Lee, S. Hannon, R. Chandar, D. A. Dale, K. L. Larson, D. A. Thilker, L. Ubeda, M. Boquien, M. Chevance, J. M. D. Kruijssen, A. Schruba, G. A. Blanc, and E. Congiu. *Deep transfer learning for star cluster classification: I. application to the PHANGS-HST survey*. , 493(3):3178–3193, Feb. 2020.

[49] K. Weiss, T. M. Khoshgoftaar, and D. Wang. *A survey of transfer learning*. Journal of Big data, 3(1):9, 2016.

[50] X. Xu, J. Li, X. Huang, M. D. Mura, and A. Plaza. *Multiple morphological component analysis based decomposition for remote sensing image classification*. IEEE Transactions on Geoscience and Remote Sensing, 54:1–20, 01 2016.

[51] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. *How transferable are features in deep neural networks?*. In Advances in Neural Information Processing Systems 27, pages 3320–3328. Curran Associates, Inc., 2014.

[52] X. Zhu, J. Hu, C. Qiu, Y. Shi, H. Bagheri, J. Kang, H. Li, L. Mou, G. Zhang, M. Haberle, S. Han, Y. Hua, R. Huang, L. Hughes, Y. Sun, M. Schmitt, and Y. Wang. *So2sat lcz42*, 2018.

[53] X. Zhu, J. Dai, C. Bian, Y. Chen, S. Chen, and C. Hu. *Galaxy morphology classification with deep convolutional neural networks*. Astrophysics and Space Science, 364(4), Apr 2019.

[54] X. X. Zhu, D. Tuia, L. Mou, G. Xia, L. Zhang, F. Xu, and F. Fraundorfer. *Deep learning in remote sensing: A comprehensive review and list of resources*. IEEE Geoscience and Remote Sensing Magazine, 5(4):8–36, 2017.

[55] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He. *A comprehensive survey on transfer learning*. CoRR, 2020.

Supplementary Material

APPENDIX A

(a) Performance using VGG-D network

Dataset	From scratch	Single-view: Linear adaptor	Single-view: Inflated network	Single-view: Multi-layer adaptor	Single-view: Subset selection	Multi-view: Subset selection 2	Multi-view: Subset selection 5	Multi-view: Linear adaptor 2	Multi-view: Linear adaptor 5
Synthetic									
CUB-5	0.5 ± 0.0	38.4 ± 3.1	30.5 ± 1.1	41.3 ± 2.8	47.1 ± 4.0	45.0 ± 1.7	49.8 ± 0.5	52.7 ± 0.6	$\mathbf{55.1 \pm 0.3}$
CUB-15	0.5 ± 0.0	41.2 ± 0.9	26.0 ± 2.8	45.4 ± 5.1	45.9 ± 5.0	49.9 ± 2.7	$\mathbf{56.5 \pm 1.0}$	$\mathbf{57.3 \pm 0.5}$	$\mathbf{56.7 \pm 0.9}$
Cars-5	2.2 ± 0.8	70.5 ± 2.1	70.2 ± 1.4	74.4 ± 1.7	72.6 ± 2.0	75.4 ± 0.9	$\mathbf{76.7 \pm 0.5}$	$\mathbf{76.7 \pm 0.4}$	$\mathbf{76.8 \pm 0.5}$
Cars-15	2.7 ± 0.4	70.7 ± 3.3	68.7 ± 3.4	70.1 ± 2.3	72.2 ± 1.7	75.8 ± 0.4	77.1 ± 0.3	77.1 ± 0.2	$\mathbf{78.1 \pm 0.4}$
Aircraft-5	1.0 ± 0.0	79.6 ± 0.8	76.6 ± 0.9	79.5 ± 0.5	79.2 ± 1.0	$\mathbf{80.8 \pm 0.9}$	$\mathbf{81.5 \pm 0.3}$	81.1 ± 0.4	$\mathbf{81.8 \pm 0.2}$
Aircraft-15	1.0 ± 0.0	78.4 ± 0.6	77.6 ± 0.9	79.1 ± 1.2	$\mathbf{79.8 \pm 2.2}$	$\mathbf{81.3 \pm 1.0}$	$\mathbf{81.6 \pm 0.5}$	$\mathbf{81.4 \pm 0.5}$	$\mathbf{81.3 \pm 0.4}$
Realistic									
So2Sat	50.3 ± 0.8	52.8 ± 0.3	51.8 ± 0.3	52.8 ± 0.7	51.6 ± 3.5	55.6 ± 1.0	$\mathbf{57.4 \pm 0.4}$	$\mathbf{57.0 \pm 0.4}$	$\mathbf{57.2 \pm 0.5}$
So2Sat^S	34.5 ± 0.7	45.1 ± 0.5	40.8 ± 0.8	40.8 ± 3.1	43.8 ± 4.9	49.8 ± 1.2	$\mathbf{51.4 \pm 0.3}$	45.2 ± 0.7	48.6 ± 0.8
EuroSAT	95.7 ± 0.2	97.1 ± 0.3	96.7 ± 0.3	97.7 ± 0.2	96.6 ± 0.7	$\mathbf{97.4 \pm 0.4}$	$\mathbf{97.7 \pm 0.2}$	$\mathbf{97.4 \pm 0.2}$	$\mathbf{97.5 \pm 0.2}$
EuroSAT^S	75.2 ± 2.6	88.4 ± 0.5	83.6 ± 1.8	92.8 ± 0.7	86.6 ± 3.6	91.6 ± 2.2	93.7 ± 0.5	$\mathbf{94.4 \pm 0.2}$	$\mathbf{94.4 \pm 0.2}$
LEGUS	51.9 ± 0.5	63.2 ± 0.5	61.4 ± 0.4	62.8 ± 0.3	61.1 ± 1.4	64.2 ± 0.4	$\mathbf{65.3 \pm 0.5}$	64.2 ± 0.4	63.9 ± 0.6
LEGUS^S	27.2 ± 5.5	54.9 ± 0.8	51.8 ± 1.3	53.1 ± 1.9	51.3 ± 2.9	55.0 ± 0.6	$\mathbf{59.3 \pm 0.6}$	55.7 ± 1.8	$\mathbf{60.0 \pm 1.6}$

(b) Performance using ResNet18 network

Dataset	From scratch	Single-view: Linear adaptor	Single-view: Inflated network	Single-view: Multi-layer adaptor	Single-view: Subset selection	Multi-view: Subset selection 2	Multi-view: Subset selection 5	Multi-view: Linear adaptor 2	Multi-view: Linear adaptor 5
Synthetic									
CUB-5	12.0 ± 0.4	46.0 ± 0.9	41.1 ± 0.7	46.1 ± 2.3	48.1 ± 2.8	48.5 ± 1.3	47.6 ± 1.0	$\mathbf{58.1 \pm 0.5}$	55.8 ± 0.3
CUB-15	15.7 ± 0.7	50.9 ± 1.6	40.8 ± 2.4	47.6 ± 2.7	51.6 ± 4.6	50.3 ± 4.2	51.7 ± 3.1	$\mathbf{60.5 \pm 1.8}$	$\mathbf{57.3 \pm 2.6}$
Cars-5	9.1 ± 0.5	$\mathbf{75.4 \pm 0.4}$	72.6 ± 0.7	$\mathbf{76.7 \pm 1.4}$	73.6 ± 1.4	73.9 ± 1.4	74.2 ± 0.2	$\mathbf{75.1 \pm 0.5}$	$\mathbf{75.3 \pm 0.4}$
Cars-15	15.7 ± 0.9	73.8 ± 0.4	72.3 ± 0.4	$\mathbf{77.0 \pm 1.3}$	$\mathbf{74.5 \pm 1.4}$	74.6 ± 1.2	74.6 ± 0.4	$\mathbf{76.4 \pm 0.3}$	$\mathbf{76.4 \pm 0.6}$
Aircraft-5	38.1 ± 0.8	73.3 ± 0.4	73.4 ± 0.4	$\mathbf{75.6 \pm 0.9}$	$\mathbf{74.3 \pm 1.1}$	$\mathbf{74.0 \pm 1.1}$	73.8 ± 0.3	73.6 ± 0.6	$\mathbf{75.1 \pm 0.6}$
Aircraft-15	44.1 ± 0.6	73.3 ± 0.4	73.9 ± 0.4	$\mathbf{75.8 \pm 0.7}$	$\mathbf{74.9 \pm 0.8}$	73.9 ± 0.4	74.0 ± 0.7	$\mathbf{74.1 \pm 1.3}$	73.3 ± 1.4
Realistic									
So2Sat	47.9 ± 0.3	47.6 ± 0.8	46.7 ± 0.9	48.7 ± 0.7	48.1 ± 2.7	52.8 ± 1.0	$\mathbf{53.9 \pm 1.0}$	52.4 ± 1.1	$\mathbf{54.2 \pm 0.6}$
So2Sat^S	36.1 ± 0.5	41.7 ± 1.0	38.0 ± 0.8	40.4 ± 1.6	41.5 ± 3.9	45.3 ± 1.1	$\mathbf{48.3 \pm 0.1}$	43.3 ± 1.0	46.6 ± 0.5
EuroSAT	97.7 ± 0.0	98.4 ± 0.1	98.4 ± 0.1	98.2 ± 0.1	97.7 ± 0.5	$\mathbf{98.3 \pm 0.1}$	$\mathbf{98.5 \pm 0.2}$	98.1 ± 0.1	$\mathbf{98.3 \pm 0.1}$
EuroSAT^S	90.4 ± 0.3	94.0 ± 0.3	95.4 ± 0.2	93.9 ± 0.4	94.0 ± 0.9	95.3 ± 0.3	$\mathbf{96.2 \pm 0.2}$	95.1 ± 0.1	$\mathbf{96.0 \pm 0.1}$
LEGUS	57.8 ± 2.0	64.2 ± 0.6	62.5 ± 0.7	63.4 ± 0.6	61.0 ± 3.3	$\mathbf{65.7 \pm 0.6}$	$\mathbf{65.1 \pm 0.3}$	64.4 ± 0.0	$\mathbf{65.1 \pm 0.6}$
LEGUS^S	48.4 ± 0.4	52.6 ± 0.3	55.3 ± 0.8	52.0 ± 1.0	48.7 ± 1.6	$\mathbf{57.8 \pm 0.4}$	$\mathbf{58.1 \pm 0.8}$	56.4 ± 0.5	$\mathbf{57.9 \pm 0.3}$

(c) Performance using ResNet50 network

Dataset	From scratch	Single-view: Linear adaptor	Single-view: Inflated network	Single-view: Multi-layer adaptor	Single-view: Subset selection	Multi-view: Subset selection 2	Multi-view: Subset selection 5	Multi-view: Linear adaptor 2	Multi-view: Linear adaptor 5
Synthetic									
CUB-5	12.3 ± 0.2	61.7 ± 0.6	55.7 ± 0.8	61.3 ± 3.0	59.7 ± 1.8	59.6 ± 1.0	58.3 ± 1.2	$\mathbf{66.4 \pm 0.2}$	64.8 ± 0.2
CUB-15	13.8 ± 0.2	67.9 ± 0.8	57.4 ± 0.5	63.1 ± 2.6	62.2 ± 3.8	62.9 ± 3.3	63.9 ± 1.3	$\mathbf{68.2 \pm 0.2}$	66.8 ± 1.7
Cars-5	12.8 ± 0.6	$\mathbf{83.2 \pm 0.5}$	82.4 ± 0.3	$\mathbf{83.8 \pm 0.3}$	82.8 ± 0.6	82.5 ± 0.9	82.1 ± 0.5	82.2 ± 0.3	$\mathbf{83.3 \pm 0.4}$
Cars-15	14.6 ± 1.2	83.4 ± 0.2	82.0 ± 0.4	$\mathbf{84.5 \pm 0.8}$	82.8 ± 0.7	$\mathbf{83.2 \pm 0.6}$	82.3 ± 0.3	$\mathbf{83.9 \pm 0.4}$	$\mathbf{84.0 \pm 0.5}$
Aircraft-5	34.2 ± 0.9	81.6 ± 0.3	81.3 ± 0.5	$\mathbf{83.0 \pm 1.0}$	81.8 ± 0.1	81.2 ± 0.6	80.5 ± 0.5	80.9 ± 0.7	80.4 ± 0.6
Aircraft-15	38.6 ± 1.0	$\mathbf{81.8 \pm 0.4}$	$\mathbf{81.7 \pm 0.3}$	$\mathbf{82.4 \pm 0.5}$	$\mathbf{82.0 \pm 0.7}$	81.0 ± 0.3	80.7 ± 0.5	81.0 ± 0.5	$\mathbf{81.4 \pm 0.6}$
Realistic									
So2Sat	46.5 ± 0.5	47.7 ± 1.1	50.6 ± 1.0	50.7 ± 1.3	47.7 ± 3.0	52.6 ± 0.6	$\mathbf{54.9 \pm 0.6}$	52.2 ± 1.1	$\mathbf{54.6 \pm 0.7}$
So2Sat^S	32.0 ± 1.6	38.1 ± 0.5	42.9 ± 1.1	44.8 ± 0.9	41.3 ± 1.7	$\mathbf{48.9 \pm 1.4}$	$\mathbf{50.5 \pm 0.2}$	44.4 ± 0.7	49.1 ± 0.9
EuroSAT	97.1 ± 0.1	$\mathbf{98.6 \pm 0.1}$	$\mathbf{98.6 \pm 0.1}$	98.3 ± 0.2	98.0 ± 0.3	$\mathbf{98.5 \pm 0.2}$	$\mathbf{98.7 \pm 0.1}$	98.1 ± 0.1	98.4 ± 0.1
EuroSAT^S	80.3 ± 0.8	95.6 ± 0.2	95.5 ± 0.1	94.8 ± 0.7	94.6 ± 1.0	$\mathbf{96.2 \pm 0.4}$	$\mathbf{96.5 \pm 0.2}$	95.7 ± 0.3	$\mathbf{96.4 \pm 0.1}$
LEGUS	62.6 ± 0.4	64.1 ± 0.4	63.9 ± 0.4	$\mathbf{64.3 \pm 0.5}$	$\mathbf{63.2 \pm 1.8}$	$\mathbf{65.1 \pm 0.4}$	$\mathbf{65.5 \pm 0.7}$	$\mathbf{64.9 \pm 0.5}$	$\mathbf{65.0 \pm 0.6}$
LEGUS^S	49.4 ± 1.8	55.5 ± 0.7	56.5 ± 0.7	54.1 ± 0.6	53.7 ± 1.5	57.4 ± 1.4	$\mathbf{59.2 \pm 0.8}$	58.1 ± 0.6	$\mathbf{60.3 \pm 0.8}$

^{S} Smaller version of the dataset using 1000 training samples.

TABLE III: **Results using hyperspectral domain adaptation methods.** Accuracy (%) for single and multi-view adaptor networks on the six proposed datasets using VGG-D, ResNet18, and ResNet50. We present results of the six synthetic datasets (CUB-5, CUB-15, Cars-5, Cars-15, Aircraft-5, and Aircraft-15) in the first six rows. Also, we include results of the three realistic datasets (and its reduced variants) in the last six rows for each architecture. We show results of the baselines (From Scratch), of the four adaptors, and results using our multi-view scheme for random subset selection and linear adaptor. The best results using single-view adaptor networks are shown in green color. The best overall results are shown in bold blue color.

APPENDIX B

Baseline on synthetic datasets

	Accuracy (%)		
Dataset	VGG-D	ResNet18	ResNet50
CUB	72.0	70.5	77.0
Stanford Cars	78.2	80.6	86.7
FGVC Aircraft	81.1	76.9	84.0

TABLE IV: **Performance on the synthetic datasets.** Accuracy by fine-tuning an ImageNet pretrained network on the RGB images of each dataset. These provide an upper-bound on the performance of the synthetic datasets.

APPENDIX C

Using initializations for learnable adaptors

	Linear adaptor		Multi-layer adaptor	
Dataset	PCA	Random	Autoenc.	Random
CUB-5	38.0	**38.4**	**43.8**	41.3
Cars-5	67.0	**70.5**	72.5	**74.4**
Aircraft-5	78.9	**79.6**	79.3	**79.5**
So2Sat LCZ42	**53.9**	52.9	49.2	**52.8**
EuroSAT	94.7	**97.1**	97.5	**97.7**
LEGUS	61.0	**62.6**	61.9	**62.8**

TABLE V: **Effect of initialization.** Accuracy using PCA vs. random for linear adaptor, and auto-encoder vs. random initialization for the multi-layer adaptor using VGG-D.

APPENDIX D

Domain shift and transferability

To quantify the effect of color permutation in domain shift and transferability we permute the color channels and fine-tune a pretrained VGG-D on CUB. We find that permuting two channels leads to an accuracy reduction of $\sim$6% and permuting three channels a reduction of $\sim$8%. These results are shown in Table VI.

	Permutation					
Dataset	**RGB**	RBG	GRB	BGR	GBR	BRG
VGG-D						
Birds	**72.0**	66.8	66.5	66.2	63.6	64.0
Cars	**78.2**	78.1	78.0	77.6	77.9	76.9
Aircraft	**81.1**	81.7	81.7	82.3	81.9	82.3
ResNet18						
Birds	**70.5**	66.9	66.6	66.0	64.8	65.4
Cars	**80.6**	80.1	80.0	79.6	79.4	79.8
Aircraft	**76.9**	76.6	76.7	76.9	76.8	76.6
ResNet50						
Birds	**77.0**	74.0	73.8	73.5	73.1	72.6
Cars	**86.7**	86.2	85.8	85.9	85.4	85.9
Aircraft	**84.0**	83.1	83.3	83.6	82.2	83.5

TABLE VI: **Accuracy obtained by fine-tuning a VGG-D network on the Birds, Cars, and Aircraft datasets with permuted channels.** All permutations except the identity lead to large accuracy drop on Birds. Cars and Aircrafts are less effected, as shape more than color is informative of the class and is preserved by the permutation.