Effective transfer learning for hyperspectral image classification with deep convolutional neural networks

Wojciech Masarczyk, Przemysław Głomb, Bartosz Grabowski, Mateusz Ostaszewski

Institute of Theoretical and Applied Informatics, Polish Academy of Sciences
Bałtycka 5, 44-100 Gliwice, Poland
Email: {wmasarczyk,przemg,bgrabowski,mostaszewski}@iitis.pl
Telephone: +48 32 2317319

Abstract

Hyperspectral imaging is a rich source of data, allowing for multitude of effective applications. On the other hand such imaging remains challenging because of large data dimension and, typically, small pool of available training examples. While deep learning approaches have been shown to be successful in providing effective classification solutions, especially for high dimensional problems, unfortunately they work best with a lot of labelled examples available. To alleviate the second requirement for a particular dataset the transfer learning approach can be used: first the network is pre-trained on some dataset with large amount of training labels available, then the actual dataset is used to fine-tune the network. This strategy is not straightforward to apply with hyperspectral images, as it is often the case that only one particular image of some type or characteristic is available. In this paper, we propose and investigate a simple and effective strategy of transfer learning that uses unsupervised pre-training step without label information. This approach can be applied to many of the hyperspectral classification problems. Performed experiments show that it is very effective in improving the classification accuracy without being restricted to a particular image type or neural network architecture. An additional advantage of the proposed approach is the unsupervised nature of the pre-training step, which can be done immediately after image acquisition, without the need of the potentially costly expert’s time.

Keywords: hyperspectral image classification; deep learning; convolutional neural networks; transfer learning; unsupervised training sample selection

1. Introduction

Hyperspectral imaging (HSI), or imaging spectroscopy, is a form of image acquisition which collects information across a range of electromagnetic spectrum. Each hyperspectral pixel contains a result of a narrowband sampling of a selected frequency range, usually corresponding to Visual and Near-Infrared (VNIR) or Short-Wave Infrared (SWIR) part of the spectrum. This process provides a detailed object reflectance information, which allows to observe interaction of light of certain frequencies with the surface particles. Such approach allows for efficient detection of foreign compounds present in the scene, diagnosing the state of process being monitored or identification of materials present in the image. Numerous applications benefit from this, e.g. medical diagnosis [1], crop health monitoring [2] or mineral mapping [3].

Classification of hyperspectral images has many potential applications, e.g. land cover segmentation [4], mineral identification [5], or anomaly detection [6]. The classification algorithms applied include both general models, e.g. the SVM [7] and dedicated approaches, taking into account spectral properties or spatial class distribution [8]. Recently there have been attempts to use Deep Learning Neural Networks...
(DLNN) for the HSI classification. The reason is that such methods have gained attention after achieving state of the art in natural image processing tasks \cite{9}. Their unique ability to process an image using a hierarchical composition of simple features learned during training makes them a powerful tool in areas where manipulation of high-dimensional data is needed. There have been a number of attempts to classify HSI with DLNN; approaches range from adapting a general vision network architecture with data preprocessing e.g. \cite{10, 11} or investigate algorithms designed with HSI in mind e.g. \cite{12, 13, 14}.

A significant problem in practical hyperspectral classification is the small number of training samples. It is related to the difficulty of obtaining verified labels \cite{4}, as often each pixel must be individually evaluated before labelling. Therefore, a reference hyperspectral classification experiment may assume number as low as 1% available samples per class \cite{5}. A number of approaches has been exploited to deal with this difficulty, e.g. including combining spatial and spectral features \cite{15}, additional training sample generation \cite{16}, extending the classification algorithm with segmentation \cite{17}, or employing Active Learning \cite{18}.

For the DLNN classification, the lack of high volume of training data is a serious complication, as they typically require a lot of data to achieve high efficiency. Optimal use of DLNN in HSI classification would require learning them with just a few labelled samples. This may be obtained by searching for well-tailored architecture for specific task \cite{10}, however such approach requires relatively big validation set to obtain meaningful results. The other approach is to expand the available training set. It may be achieved either by artificially augmenting training set or using different dataset as a source for pre-training \cite{19}. Another approach is to add a regularization step to improve the generalization ability with limited number of training samples \cite{20}. A simplification of the network architecture for classification with few training samples is employed in the MugNet network \cite{21}. Finally, where possible, the transfer learning approach is used, e.g. \cite{22}.

The transfer learning \cite{23} uses training samples from two domains, which share common characteristics. A network is first pre-trained on the first domain, which has plentiful supply of training samples but does not solve the problem at hand. Then, the training is updated with the second domain, which adapts the weights to the actual problem. Unfortunately, due to the specifics of HSI application, the cases where we can borrow on a related dataset with a supply of training samples are rare. Depending on the application, the additional information for transfer learning can be either from another HSI image \cite{24}, natural (non-HSI) image \cite{25} or through algorithmic computation \cite{26}. However, we note that many of remote sensing images share common properties, most notably the ‘cluster assumption’ \cite{27} – pixels that are close to one another or form a distinct cluster or group frequently share the class label. This leads us to propose a simple unsupervised labelling scheme, which uses the original problem image. Given a HSI $X$ and a set of limited training labels $Y$, we: (1) generate a set of artificial training labels $Y'$, with count much larger than the original set $|Y'| \gg |Y|$ (2) pretrain the network on $(X, Y')$, then (3) finalize the training on the original labels $(X, Y)$. The proposed labelling heuristic, which divides the image into rectangular blocks, can be universally applied; yet as the experiments show improves significantly the classification accuracy of DLNN networks. We investigate the performance of this approach across a number of selected network architectures and datasets. Our results show that this technique consequently improves accuracy of DLNN in realistic classification cases, while at the same time is cheap to use and of possible applications in wide range of scenes.

An additional advantage is the unsupervised nature of the method. As it does not require training samples, it can be applied right after image acquisition, when human expert’s labellings are not yet available. This can significantly shorten the time to train on the final set of labels.

1.1. Related work

This Section presents a short survey of related literature. First the classification of HSI with DLNN is discussed, then the accuracy improvement with transfer learning, especially considering the unsupervised approach with artificial labelling applied to the HSI images.
1.1.1. HSI classification with DLNN

A number of DLNN architectures have been proposed, inspired by mathematical derivations and/or neuroscience studies. The Convolutional Neural Networks (CNN) [28] are the special case of deep neural networks which were originally developed to process images but are also used for other types of data like audio. They combine traditional neural networks with biologically inspired structure into a very effective learning algorithm. They scan multidimensional input piece by piece with the convolutional window which is the set of neurons with common weights. Convolution window processes local dependencies (features) in the input data. The output corresponding to one convolutional window is called a feature map and it can be interpreted as a map of activity of the given feature on the whole input. The CNN remain one of the most popular architectures for DLNN classification in use today.

Other approaches include the generative architectures, e.g. the Restricted Boltzmann Machine (RBM) [29, 30], Autoencoders (AE) [31] or Deep Belief Network (DBN) [32, 33]. Yet another popular architecture is the Recurrent Neural Network (RNN) which, through directed cycles between units, has the potential of representing the state of processed sequence. They are applicable e.g. for time series prediction or outlier detection. The most popular types of RNN are Long Short-Term Memory (LSTM) networks [34] and Gated Recurrent Units (GRUs) [35]. They improve the original RNN architecture by dealing with exploding and vanishing gradient problem.

For classification of HSI data, the CNN is the most popular architecture chosen. In [10] the simple CNN architecture is adapted to HSI classification; the lack of training labels is mitigated by adding geometric transformations to available training data points. In [36] authors use three kinds of convolutional windows: two of them are 3D convolutions which analyse spatial and spectral dependencies in the input picture, while the third is the 1D kernel. Next the feature maps from these three types of convolutions are stacked one after the other and create joint output of this first part of the network. The following layers consist only of the one dimensional convolutional kernels and residual connections. The authors of [11] introduce a parallel stream of processing with an original approach for spatial enhancement of hyperspectral data. The authors of [14] design a deep network that reduces the effect of Hughes phenomenon (curse of dimensionality) and use additional unlabelled sample pool to improve performance. In [12] authors propose an alternative architecture called RPNet based on prefixed convolutional kernels. It combines shallow and deep features for classification. Another architecture (MugNet) is proposed in [21] with a focus on simplicity of processing for classification of hyperspectral data with few training samples and reduced number of hyperparameters. A yet another architecture approach is used in [20] where a multi-branch fusion network is introduced, which uses merging multiple branches on an ordinary CNN. An additional L2 regularization step is introduced to improve the generalization ability with limited number of training samples. The work [13] proposes a strategy based on multiple convolutional layers fusion. Two distinct networks, composed of similar modules but different organization, are examined.

Other architectures are also used. For example in [37] authors utilize the sequential nature of hyperspectral pixels and use some variations of recurrent neural networks – Gated Recurrent Unit (GRU) and Long-Short Term Memory (LSTM) networks. Moreover, in [38] one dimensional convolutional layers followed by LSTM units were used. Chen et.al. [39] use artificial neural networks for feature extraction. They utilize stacked autoencoders (SAE) for feature extraction from pixels, and PCA for reduction of the spectral dimensionality of the training segments taken from the picture. Next, the logistic regression is performed on this spectral (SAE) and spatial (PCA) extracted information. Another approach uses stacked SAE for an application study – detection of a rice eating insect. RNN architectures are also employed, as they are suitable for processing the spectral vector data. The work applies sequential spectral processing of hyperspectral data, using a RNN supported by a guided filter. In [42] authors use the multi-scale hierarchical recurrent neural networks (MHRNNs) to learn the spatial dependency of non-adjacent image patches in the two-dimension (2D) spatial domain. Another idea to analysing HSI is spatial–spectral method in which network takes information not only from spectrum bands but also from spatial dependencies of image [56].
1.1.2. Improving DLNN classification with transfer learning

Transfer learning is one of the approaches to mitigate a situation when one either lacks sufficient training samples or computing power to train a network. It adapts a model (algorithm with trained parameters) from one problem domain to another, when the domains are similar. Formally, we can define it as in \([23]\). Let \(D = (X, P(X))\) be a domain consisting of feature space \(X\) and its probability distribution \(P(X)\), and \(T = (Y, f(\cdot))\) be a task consisting of label space \(Y\) and an objective predictive function \(f(\cdot)\). This function can be approximated using samples of the form \((x_i, y_i)\), where \(x_i \in X\) and \(y_i \in Y\). Transfer learning consists of using knowledge concerning source domain \(D_s\) and its task \(T_s\) to better approximate objective function \(f(\cdot)\) belonging to the different target task \(T_t\). We assume that at least one of the following inequalities is true: \(D_s \neq D_t, T_s \neq T_t\). Transfer learning is especially useful whenever \(|D_s| \ll |D_t|\).

Transfer learning is simple to apply in the case of convolutional neural networks (CNNs). In \([33]\), authors compared different versions of transfer learning for CNNs in the case of natural images classification. They studied its effects depending on the number of transferred layers and whether they were fine tuned or not as well as depending on the differences between the considered datasets. In \([44]\), authors used transfer learning on CNNs to recognize emotions from the pictures of faces. Other uses include evaluating the level of poverty in a region given its remote sensing images \([45]\) and computer-aided detection using CT scans \([46]\).

There have been applications of transfer learning in the general remote sensing (not-hyperspectral) images. In \([47]\) deep learned features are transferred for effective target detection; negative bootstrapping is used for improving the convergence of the detector. A similar approach is applied in \([48]\) where RNN network trained on multispectral city images is used to derive features for studying urban dynamics across seasonal, spatial and annual variance. The authors of \([49]\) study the performance of transfer learning in two remote sensing scene classifications. The results show that features generalize well to high resolution remote sensing images. As the work \([50]\) shows, transfer learning can be applied in remote sensing using RNN architectures also.

Recently, transfer learning has been also applied to the HSI data. In \([22]\), authors applied it for CNNs originally used for classifying well known remote sensing hyperspectral images to classify images acquired from field-based platforms and regarding a different domain. The authors of \([24]\) use an intermediate step of supervised similarity learning for anomaly detection in unlabelled hyperspectral image. A different approach to transfer learning is proposed in \([51]\) which explores the high level feature correlation of two HSI. A new training principle simultaneously processes both images, to estimate a common feature space for both images. A yet another approach is proposed in \([25]\) where HSI superresolution is achieved using supported high resolution natural image. This natural image is used as a training reference, which is later adapted to HSI domain. In \([52]\), iterative process combines training and updating the currently used training label set. Two specialized architectures (for spatial and spectral processing) are used. The training iteratively extends the current label set, starting from the initial expert’s labels.

The above approaches do not apply to the arguably most popular practical scenario, where only a single HSI with a handful of labels is available. Moreover, getting the training labels often requires additional resources (e.g. expert consultation and/or site visit). It is thus desirable to have unsupervised methods for realization of the pretraining step. Authors of \([26]\) use outlier detection and segmentation to provide candidates for training of target detector in HSI. This information is used to construct a subspace for target detection by transfer learning theory. This shows the potential of using an unsupervised approach, however limited to separation of target/anomaly points from the background. In the work of \([19]\), a separate clustering step is used for generation of pseudo-labels, using Dirichlet process mixture model. The network is trained on the pseudo-labels, then the all but last layers are extracted, and the final network is trained on the originally provided training labels. While this scheme is shown to be effective in the presented results, it relies on a complex non-neural preprocessing and tailoring the DLNN configuration to each dataset separately. Also, the effect of size of label areas and effects on different architectures are not investigated. We show that similar gains can be made with a simpler preprocessing, independent of
the DLNN architecture chosen. The authors of [53] propose to use a sparse coding to estimate high level features from unlabelled data from different sources. This approach does not require training data, but is tailored to the case where multiple inputs are available, preferably with diverse contents.

Our approach, which divides the image into rectangular blocks with assigned artificial classes, is motivated by two observations. First, the complicated spatial and spectral statistical properties present in the input classes are not necessarily discovered by the automatic clustering algorithms; it seems that local image patches have a similar balance between homogeneity and diversity as the results of more complex segmentation methods. The potential drawback of inadequate artificial class assignments can be offset by having much larger pool of training samples. The second observation, which provides additional motivation for unsupervised approach, is that when training sample number is low, it may not sample all of the class space (see discussion in [16]) and may lead to biased class models through a negative feedback loop. The simplicity of our scheme allows us to investigate its performance independent of DLNN architectures or parameter settings.

Our method is a modification of the regular transfer learning approach. In our case, the source domain $D_s$ consists of every point in the hyperspectral image, while the target domain $D_t$ is composed of only the labelled samples. Therefore, in our approach $X_s = X_t$, but $P(X_s) \neq P(X_t)$. Moreover, because we assume that the true source domain labels are unknown to us, we create the artificial labels for our source domain $D_s$ using only the information about the spatial placement of the samples that is available. This has an additional advantage of making possible to shift the potentially time consuming pre-training from the expert labelling moment to the acquisition moment. In other words, network training does not need to be held back until the expert’s labels are available, but can commence right after the image is recorded.

2. Method

Our approach is based on the following two step algorithm. In step one, artificial labels are created for the input image. In step two, the network is trained; first pre-trained on the artificial labels, then fine-tuned on the true class labels. Those steps, along with architectures we use for evaluation of the approach, are detailed below.

2.1. Creating artificial labels for the source domain

Our proposition is based on two observations. First, it has been noted that starting from the initial neurons, subsequent layers of a deep neural network form a representation of a local input data structure [54]. Second, in many hyperspectral classification problems (e.g. land cover labelling in remote sensing, paint identification in heritage science, scene analysis in forensics) we observe classes with a number of local spectral pattern variations. These variations are cumulative result of: variations in classes, non-uniformity in class structure (e.g. the same class can contain differing crop types), spectral variations (e.g. same crop in two areas can have differing properties due to sunlight exposure, soil type) or acquisition conditions (e.g. level of lighting, shadows). This effect often leads to a blob-like structure of a hyperspectral dataset, where the blobs can have different properties. By introducing locally continuous artificial classes, even when they don’t correspond to the true ones, we can force the network to represent those blobs patterns.

Our method for creating artificial labels is a simple segmentation algorithm which assumes the local homogeneity of samples’ spectral characteristics. It works by dividing the considered image into $k$ rectangles, where each of these rectangles has its own label. For an image of height $h$ and width $w$, we divide its height into $m$ roughly equal parts and its width into $n$ roughly equal parts, so that $k = m \cdot n$. We then get $k$ rectangles, where each one’s height equals approximately $h/m$, while its width equals approximately $w/n$. Each of these rectangles defines a different artificial class with a different label. A schematic is presented in Figure 1.

An open question is whether a clustering algorithm, like [19] or outlier segmentation [26] could be adapted here leading to greater efficiency. Our initial experiments with several variants of clustering did
Figure 1: The overview of the unsupervised pretraining algorithm proposed in this work. First, the network is pretrained on grid-based scheme of artificially assigned labels. The network weights are then fine-tuned on a limited set of training samples selected from true labels, consistent with typical hyperspectral classification scenario.

not provide significant gains in performance in comparison to the proposed method. This may suggest that the regular segmentation provides a regularization effect that prevents overtraining on artificial classes, where the network learns the clustering scheme used in addition to the data properties. Our approach has common motivation with self-taught learning [53], where we want the classifier to derive high-level input representation from the unlabelled data; however we use the same data for both training stages and instead change the label set. It also avoids combining neural and non-neural approaches, and prevents introducing additional assumptions through the manual selection of the latter.

2.2. Selected network architectures

In our experiments three architectures were tested, based on [36, 10, 55]. All three share a common approach to exploit local homogeneity of hyperspectral images, however each one has its unique strengths and weaknesses making them an interesting testbed for the universality of the proposed method. The first architecture features relatively high number of convolutional layers which might be helpful in transfer learning application. The second architecture, to the best of authors knowledge, is one of the best networks that are trained on limited number of samples per class. However due to its constrained capacity, it may not benefit as much from the pretraining phase. The last of the considered convolutional neural networks is conceptually the simplest of the three, which allows us to test our approach using more conventional convolutional architecture.
Multi-scale filter bank Residual Units Classification layers

(a) Illustration of the ‘deep’ architecture (denoted A9, based on [36]). Each consecutive cuboid represents data blobs obtained after applying convolutional filters which are depicted as horizontal pyramids. In the last layer softmax and argmax functions are applied in training and testing, respectively.

(b) Illustration of the ‘shallow’ architecture (denoted A3, based on [10]). Each consecutive cuboid but the last represents data blobs obtained after applying convolutional filters which are depicted as horizontal pyramids. To obtain the last layer, the global average pooling is used.

(c) Illustration of the ‘simple’ architecture (A5, based on [55]). The convolutional layer is followed by flattening of the features and four dense layers. In the last layer softmax and argmax functions are applied in training and testing, respectively.

(d) Details of the ‘deep’ network implementation. To transform $n$-dimensional vector to $c$-dimensional vector with fully connected layer one needs $n \times c$ weights (right). To process $(1 \times 1 \times n)$ data cube and achieve $c$ different feature maps with $(1 \times 1)$ convolution it requires $n \times c$ different weights (left). In contrast to fully connected layer, filters used in convolutional layers for processing one single vector might be reused across whole processed patch/image, hence reduce number of total weights in network.

Figure 2: Convolutional networks’ architectures used in this work.
In the pretraining phase, each deep learning neural network is trained on the whole image with artificial labels. It is followed by the fine tuning phase, where whole network except last layer is initialized by weights obtained in pretraining phase. Last layer is then swapped by randomly initialized one with the number of neurons which allows classification on real classes. Such a network is then only fine tuned on the part of data that were labeled by an expert with real labels.

2.2.1. The ‘deep’ architecture (A9)

First tested model is based on [36] which is well established network in community (see Figure 2a). Additionally, it is one of the deepest convolutional neural networks successfully used for classifying hyperspectral data. It is shown in [56] that deep architectures enable network to learn a hierarchically composed abstract data representations that are generally invariant to local changes in the data. Additionally, number of abstract representations that network is able to generate by re-using lower-level features is growing exponentially with the depth the network, which makes them a good fit for transfer learning. As shown in empirical study [43], even in case of frozen pre-trained weights, the network achieves far better accuracy than randomly initialized one. This allows one to assume that previously created filters were re-used with positive impact on classification accuracy.

Throughout this description all the parameters and architectural settings were taken from [36] unless it is stated differently. Proposed network consists of three main parts: multi-scale filter bank, residual units and classification layers.

Multi-scale filter bank extracts both spectral and spatial information from provided data. It is achieved by applying three convolutional layers with different kernel sizes. Sizes of kernels are $(5 \times 5)$, $(3 \times 3)$, $(1 \times 1)$ and number of filters in each layer is 128 as visualized in Figure 2a with red, blue and green, respectively. Different sizes of convolutional kernels result in outputs of different shapes and to mitigate this problem max pooling is applied, for details see [36]. At the end, output of each layer is stacked along the channel dimension and serves as an input to residual units.

Residual unit [57] is a stack of a few convolutional layers (usually 2 or 3) with additional identity mapping between input and the output of the unit. Let $H(x)$ be the desired mapping one wants to fit, then according to authors, it is easier to fit a mapping $F(x) := H(x) − x$ than $H(x)$. Once $F(x)$ is fitted, the desired mapping is obtained by addition $H(x) = F(x) + x$. Possible explanation of the effectiveness of this approach is linked with propagation of the gradient, which does not vanish or explode thanks to the identity mapping [58]. Analysed network employs two consecutive residual units, each consisting of two convolutional layers with $(1 \times 1)$ kernel size and 128 filters. Output of the second residual unit is then propagated to classification layers.

Classification layers’ objective is to take data transformed by previous layers and output desired class scores. It is usually achieved by fully connected layers, however, in this architecture the convolutional layers with $(1 \times 1)$ kernel size were applied. Applying $(1 \times 1)$ convolutional layer with $n$ filters to a single pixel input with $c$ channels is equivalent to applying dense layer with $c$ input neurons and $n$ output neurons as presented in Figure 2a. It is important to note that convolutional layers apply filters with the same weights across whole input image and hence, in contrast to fully connected layers, reduce number of required parameters. First and second classification layers in discussed architecture have 128 filters and the last layer has the number of filters equal to the number of classes $c$ in processed data.

At the end of the network, softmax function is applied elementwise and result of this operation is used as argument to cross entropy loss function along with labels. During test time softmax function is swapped with argmax operation that is applied once at the whole output patch.

After every layer ReLU (Rectified Linear Unit) activation function is applied. In Residual Units ReLU is applied after the addition of identity mapping. As the authors suggest we used Local Response Normalization (LRN) only at the output of „multi-scale filter bank” and after the first convolutional layer in first Residual Unit. The parameters of LRN suggested by the authors were used (depth radius $n = 5$, bias $k = 1$, scale factor $\alpha = 0.0001$ and exponent $\beta = 0.75$). Another authors’ suggestion that was applied
are dropout layers which are placed after first and second classification layers with dropout rate $p = 0.5$.

As the important feature of this network is the 9 processing layers, it will be denoted as A9 or ‘deep’ network in the further parts of this paper.

Training schedule. As in original work [36], the optimizer is SGD (Stochastic Gradient Descent) with momentum $m = 0.9$. Batch size is set to 10 and whole training takes 100,000 iterations, where one iteration equals processing one batch. Authors also proposed learning rate schedule that reduces learning rate by an order of magnitude at 33,333 iteration and 66,666. In contrast to original paper data augmentation was not used in any part of training. Because of authors’ uncertainty regarding the parameter setting in the original work, weight decay technique was also not utilized.

Fine tuning of the pretrained model. In pre-training phase, where artificial labels serve as supervisory signal to network, weight are initialized according to schema proposed in the reference work [36]. Weights are initialized randomly from Gaussian distribution with mean zero, but with different standard deviation. First residual model’s and last classification layer’s weights are initialized with standard deviation set to 0.005 and the rest of weights are initialized with standard deviation set to 0.01. All biases except last layer are initialized with constant 1 and the last is initialized with 0.

In the fine tuning phase, identical network is used except for two details. First, at architecture level, last classification layer is modified to match the number of classes $c$ that are in fine-tuning set. Second, random initialization is used only in last classification layer according to schema in pre-training phase, all the other weights are copied from the model that was trained in pre-training phase.

2.2.2. The ‘shallow’ architecture (A3)

The second of the considered architectures is a three layer convolutional neural network which was proposed by Yu et al [10] (see Figure 2b). It is the network particularly tailored for classifying high-dimensional hyperspectral data with highly limited number of training samples. As well as Deep model, Shallow network also processes inputs in the form of $(5 \times 5)$ patches. It is done by applying three consecutive convolutional layers with kernels of size $(1 \times 1)$ and the number of filters equal to 128, 64 and $c$, respectively, $c$ being the number of classes. These convolutional kernels cannot extract features in spatial domain, but their use reduces overfitting. To extract features in spatial domain, two local response normalization layers with depth radius of 3, the scale factor $\alpha = 0.0001$ and the exponent $\beta = 0.75$ are used after the first two convolutional layers. Additionally, like in [59], the global average pooling (GAP) layer is used after the last convolutional layer, replacing the usually implemented dense layers and reducing the number of trainable parameters. To further ensure that the model will not overfit, dropout with drop rate set to 0.6 is used after the first two convolutional layers and tested by the authors, confirming better results both on the training and test sets.

As this network has 3 processing layers, it will be denoted as A3 or ‘shallow’ network in the further parts of this paper.

Training schedule. To train the shallow model, standard Adam optimizer is used with fixed learning rate set to 0.00001, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. Batch size equals 8 and the number of iterations is set to 300,000. Loss function is a standard cross-entropy. Again, the data augmentation strategy is not applied. Both for pre-training and fine-tuning the same training procedure is applied.

Fine tuning of the pretrained model. In pretraining, the weights are initialized randomly from the normal distribution with zero mean and standard deviation equal to 0.05. Such initialization setup was used in the configuration files from the shared repository [59] from article [10]. In fine tuning, the first two layers are initialized by weights obtained from pretraining, while the last layer is initialized as in the pretraining.

[1] https://github.com/ShiqiYu/caffe
2.2.3. The ‘simple’ architecture (A5)

The last architecture was inspired by the work of Liu et al. [55] (see Figure 2c). In their work, the convolutional network was developed and used as part of the siamese network model, which is a model composed of two identical neural networks used for comparing pairs of data. In our case, only the convolutional network is required, so the part of the model specifically related to siamese networks was discarded. In consequence, we obtained conceptually simplest of the three considered convolutional architectures, which allows us to further validate our method.

The considered architecture processes inputs in the form of \((9 \times 9)\) patches in order to utilize both spectral and spatial information and improve the performance of classification. First, the convolutional layer with ReLU activation on top of it and with the kernel of size \((3 \times 3)\) and 100 or 60 filters, depending on the considered dataset, is applied. These filter sizes are set to be smaller than the number of features present in the input to account for the strong correlation between different bands and minimize the number of parameters. It is then followed by max pooling layer of pool size equal to \((2, 2)\) and with the stride of 2. The output is then flattened and processed by three consecutive dense layers with 1000, 500 and 300 units, respectively. The ReLU is applied after each of these layers. The last layer is a dense layer with the number of units equal to the number of classes and with the softmax function on top of it. This layer is a classification layer of the considered network.

As this network has effectively 5 processing layers, with last three being simple dense layers, it will be denoted as A5 or ‘simple’ network in the further parts of this paper.

**Training schedule.** In the case of this architecture, SGD without momentum is used. The authors tested different values of optimizer’s learning rate and found that its value can be neither too big nor too small. They applied the learning rate of 0.01 at the start of the training and 0.001 after some iterations, which provides effective learning at the start of the training process and ensures the consistent decrease of the loss function at the later stage. In our setting, learning rate is equal to 0.01 for the first 50000 iterations and 0.001 for the rest of the training.

Batch size is equal to 50 and the number of iterations equal 100000. Cross-entropy loss function is used. Again, no data augmentation is used and both for pre-training and fine-tuning the same training procedure is applied.

**Fine tuning of the pretrained model.** In pretraining, all weights are initialized using Glorot uniform initializer [60]. In fine tuning phase, every layer except the last is initialized using weights obtained from pretraining, while the last layer is again initialized using Glorot uniform initializer.

3. Experiments

This section presents the experiments performed for evaluation of the proposed pre-training approach with unsupervised labelling. Our main focus is the quantitative and qualitative evaluation of the proposed scheme, which is realized as the first experiment. Due to the potential variability introduced by the size and shape of the patches used to define artificial classes, we have performed the second experiments to observe this relationship. Finally, noticing that big patches (larger area, smaller total number of classes) perform worse then little ones (smaller area, larger total number of classes), we investigate this in the third experiment.

3.1. Data sets

For the experiments we have used three hyperspectral datasets (see Figure 3). The first two are well-known remote sensing images Indian Pines and Pavia University, which are the basis of the first and second experiment. The third dataset is the image of paints from museum’s collection, from [61], which has suitable properties for the third experiment.
The Indian Pines dataset was collected by the AVIRIS sensor over the Northwest Indiana area. The image consists of $145 \times 145$ pixels. Each pixel has 220 spectral bands in the frequency range $0.4–2.5 \times 10^{-6}$ m. Channels affected by noise and/or water absorption were removed (i.e. [104–108], [150–163], 220), bringing the total image dimension to 200 bands. The reference ground truth contains 16 classes representing mostly different types of crops. To be consistent with experiments performed in [36], we choose only 8 classes (see table in Figure 4a).

The Pavia University dataset was collected by the ROSIS sensor over the urban area of the University of Pavia in Italy. This image consists of $610 \times 340$ pixels. It has 115 spectral bands in the frequency range from 0.43 to 0.86 $\times 10^{-6}$ m. The noisiest 12 bands were removed, and remained 103 were utilized in the experiments. Ground truth includes 9 classes, corresponding mostly to different building materials.

The Pigments dataset [61] was collected by the SPECIM hyperspectral system in the Laboratory of Analysis and Nondestructive Investigation of Heritage Objects (LANBOZ) in National Museum in Kraków. This image consists of $455 \times 310$ pixels. Each pixel has 256 spectral bands in the frequency range from 1000 to 2500 nm. Ground truth consists of manual annotations of different green pigments used in the mixture of paints for various painting regions. The image of oil paints on paper was used, selected from four available, as it was considered one of the more challenging of the set.

To assure the stability of considered models during training and testing phases, all three datasets were subjected to a feature transformation. For a given dataset, mean $m_b$ and standard deviation $\sigma_b$ of each hyperspectral band $b$ were calculated. In the case of all three datasets, and for each given pixel $x$ and band $b$, the corresponding mean $m_x$ was subtracted, $x(b) := x(b) - m_x$. In the case of Pigments dataset, all pixels were additionally divided by the corresponding standard deviation value $\sigma_x$, $x(b) := \frac{x(b)}{\sigma_x}$.

### 3.2. Experiment design

The first experiment’s objective is the evaluation of the proposed approach. For this task the Indian Pines and Pavia University datasets were used. As discussed previously, the training was divided into pretraining and fine-tuning stages. In pretraining, the data was labelled through assigning an artificial class to each block of dimensions $5 \times 5$ No ground truth data was used at this stage. In the fine-tuning stage, a selected number of ground truth labels was used. The number of training samples from each class was set at $n = 5, 15, 50$. This allowed to observe the performance both in typical hyperspectral scenarios (small number of classes used) and deep network scenarios (larger number of samples per class available). Because the classification accuracy depends on the training set used in fine tuning each experiment was repeated $n_1 = 15$ times for error reporting and $n_2 = 50$ times for sensitivity analysis and stability verification. The performance is reported in Overall Accuracy (OA) after fine tuning. Additionally, Average Accuracy (AA) and $\kappa$ coefficient were inspected and improvements verified with statistical tests.

The second experiment investigates the variability introduced by the size and shape of the patches used in artificial labelling. For this experiment only the Indian Pines image is used, as it is the more challenging dataset. The network investigated is the A9 ‘deep’ architecture, chosen because it has the most potential to be affected by the transfer learning process. The investigated square patches were created by dividing horizontal and vertical side of an image into $w = 2, 3, 5, 7, 9, 15, 19, 25, 29, 36, 39, 48, 72$ equal parts. The vertical stripes were created by dividing horizontal side of an image into $s = 2, 5, 9, 16, 25, 36, 49, 81$ equal parts. The vertical stripes were included to observe whether the pixel distance affects the performance – for square patches, all the pixels share similar neighbourhood; for stripes, the top and bottom pixels have a notable spatial separation and, arguably, the distant pixels should not be marked with the same class label without prior knowledge of spatial class distribution. Note that in case of square patches made by dividing each side of an image into $w = 29, 36, 39, 48, 72$ equal parts the size of a square patch is smaller than the size of a processing window $5 \times 5$ in tested architecture. That means no sample fed to a network during pre-training phase has a coherent class representation (i.e. a single class present in the window).
During the realization of the experiments it has been observed that more numerous patches division produces a better pre-training set than less numerous. To investigate this further, we perform the third experiment, using the Pigment dataset, which has ground truth structure corresponding to the proposed artificial class division scheme. There, we investigate the following cases:

1. The performance of DLNN without pre-training (GT).
2. The performance of DLNN with pre-training with artificial patches of size 5 × 5, 20 × 20, 30 × 30.
3. The performance of DLNN with pre-training performed with 2 classes prepared from joining the ground truth classes (GT-2).
4. The performance of DLNN with pre-training performed with 10 classes prepared by splitting the ground truth classes (GT-10).

The joining and splitting (GT-2 and GT-10) scenarios were the main investigation in this experiment, to observe how joining and splitting affects the ‘true’ ground truth classes, and whether this tendency correlates with artificial classes studied in experiment two. In all cases we use 9-layer DLNN. The joining and splitting scheme is presented in Figure 2B.

### 3.3. Experiment results

The first experiment’s results are presented in Table 1. Each column presents the result for one type of network, each row for a set dataset and the number of training examples. Each table cell presents the results with and without pre-training, in percent of Overall Accuracy, including the standard deviation of the result. The results from Table 1 were computed from a batch of 15 independent runs for each case. Additionally, a batch of 50 of experiments were performed for sensitivity analysis of small variations of hyperparameter setting and computational library versions; the results were very similar to those presented. A Mann–Whitney U test was performed on the results to confirm statistical significance of the improvement gained with the proposed method. As Overall Accuracy can be sensitive to class imbalances, Average Accuracy and \( \kappa \) coefficient were computed for additional verification, and were inspected for negative performance.

The presented results show that application of the proposed method leads to definite and consistent improvement in accuracy across different images, number of ground truth labels used and network architectures. In all but one case, the improvement is statistically significant, and in some cases approaches 20 percentage points. The most improvement is seen in the ‘deep’ (A9) network; this is to be expected as this

|                  | A9 (‘deep’) architecture | A3 (‘shallow’) architecture | A5 (‘simple’) architecture |
|------------------|--------------------------|-----------------------------|---------------------------|
|                  | pretraining/no pretraining | pretraining/no pretraining | pretraining/no pretraining |
| IP 5/class       | 74.04 ± 4.1\(^1\)/52.62 ± 4.4 | 72.80 ± 3.2\(^1\)/66.15 ± 4.5 | 63.52 ± 4.2\(^1\)/50.05 ± 5.1 |
| IP 15/class      | 87.04 ± 2.4\(^1\)/67.58 ± 3.2 | 87.04 ± 2.1\(^1\)/82.61 ± 2.8 | 75.30 ± 1.7\(^1\)/64.18 ± 2.8 |
| IP 50/class      | 93.66 ± 1.3\(^1\)/80.51 ± 4.8 | 94.65 ± 1.0\(^1\)/93.75 ± 1.2 | 87.06 ± 0.9\(^1\)/81.39 ± 1.1 |
| PU 5/class       | 79.50 ± 5.7\(^1\)/64.77 ± 8.0 | 80.33 ± 5.2\(^1\)/73.31 ± 4.1 | 74.34 ± 7.0\(^1\)/65.55 ± 3.8 |
| PU 15/class      | 91.46 ± 2.7\(^1\)/81.95 ± 3.6 | 91.96 ± 2.6\(^1\)/88.21 ± 2.9 | 89.33 ± 3.4\(^1\)/75.50 ± 2.4 |
| PU 50/class      | 97.41 ± 1.0\(^1\)/93.53 ± 1.0 | 96.84 ± 1.2\(^1\)/96.08 ± 0.9 | 96.55 ± 0.5\(^1\)/87.79 ± 1.7 |

\(^1\) Statistically significant improvement, evaluated with Mann–Whitney U test, with \( P < 0.01 (\dagger) \) or \( P < 0.05 (\ddagger) \).
Table 2: The second experiment results. Grid density describes number of rectangular patches which represent artificial labels for pre-training phase. Num of stripes denotes number of vertical stripes which represent artificial labels for pre-training phase. Accuracies are given as Overall Accuracy for learning of the ‘deep’ network (A9) with transfer learning on the Indian Pines dataset.

| Grid density / model | OA     | Num of stripes / model | OA     |
|----------------------|--------|------------------------|--------|
| (2x2)                | 61.88% | 2                      | 58.53% |
| (3x3)                | 64.45% | 3                      | 67.68% |
| (5x5)                | 75.05% | 5                      | 68.58% |
| (7x7)                | 72.33% | 7                      | 69.25% |
| (9x9)                | 74.06% | 9                      | 69.09% |
| (14x14)              | 74.13% | 12                     | 69.23% |
| (19x19)              | 73.24% | 16                     | 70.08% |
| (24x24)              | 73.43% | 25                     | 70.41% |
| (29x29)              | 70.19% | 36                     | 70.41% |
| (36x36)              | 69.88% | 49                     | 70.41% |
| (39x39)              | 68.69% | 81                     | 68.41% |
| (48x48)              | 69.25% | 12                     | 68.41% |
| (72x72)              | 66.70% | 16                     | 68.41% |

Table 3: The third experiment results. Evaluation of pretraining on Pigments dataset using the proposed approach and classes created from ground truth. The objective was to collate the performance of artificial labels of different sizes with those created through splitting or joining the ground truth.

| Experiment setting | GTa | (5x5)b | (20x20)b | (30x30)b | GT-2c | GT-10c |
|--------------------|-----|--------|----------|----------|-------|--------|
| OA                 | 61.15% | 68.35% | 75.70%   | 73.99%   |       | 85.40% |

a No pretraining.
b Pretraining with artificial classes (proposed method).
c Pretraining with modified ground truth classes (verification).

architecture has the most potential to benefit from additional training samples. The qualitative evaluation of selected realizations (corresponding to the median score) is presented in Figures 5 and 6. The results confirm the validity of the proposed approach.

In the second experiment (Table 2), first the grid size was investigated. The range varies from 2 × 2, which equals 4 artificial classes, up to 72 × 72, 5184 artificial classes. It can be seen that the score rises sharply until the number of artificial classes reaches approximately the number of original classes (at 5 × 5, note that the original IP ground truth leaves a sizeable portion of background unmarked, which most probably would contribute some additional classes if marked). After that value, there’s a declining trend. It can be noted that the scores are higher with smaller patches. It seems viable to form a conclusion, that when the original class number is unknown, it is better to overestimate than underestimate their number. In the latter case, it is possible that even a chance guess would provide a satisfactory performance. The stripes do not form as good a training set as rectangular grid, which confirms the initial supposition that artificial classes should be confined to local areas. Some improvement however is still seen, which supports our overall proposition, that general artificial labelling can be used for improving the DLNN performance without precise estimation of the artificial class patch size. This experiment was performed with A9 ‘deep’ network, with 5 training samples per class and 50 experiment runs.

Table 3 presents result of the third experiment. Here, the original performance (GT) can be significantly improved by the grid-based artificial labelling (see results for 5 × 5, 20 × 20, 30 × 30). However,
in this case the performance gain can be confronted with a label dataset created from ground truth data (GT-2, GT-10). As can be expected, the ground truth data provides a higher performance; however, the artificial labelling provides half of that gain with no prior information needed. The ground truth experiments GT-2 and GT-10 also confirm the observation that classes split is a better option than joining. This experiment was performed with A9 ‘deep’ network with 5 training samples per class and 50 experiment runs.

3.4. Discussion

Our results confirm the validity of the original idea: a simple artificial labelling through grouping of the samples based on a local neighbourhood provides an efficient transfer learning scheme. It brings significant improvements of accuracy across datasets and DLNN configurations. The results for different datasets, which have differing ground truth layouts suggest that it is not the random alignment with the regularity of some ground truth pattern. It is also seen that the local structure is important, as seen in the advantage of grid division over stripe division. The generally better performance of higher over lower number of artificial classes suggests an explanation in that for transfer learning, it is not as important to locate the exact number of classes, but to isolate and learn their components, perhaps for better internal feature representation. While in case of hundreds of artificial classes network was unable to achieve satisfying accuracy during pre-training phase, it was still a good base model to apply transfer learning and achieve accuracy boost. Besides improving the accuracy in some network training scenario, we note two more advantages of the proposed approach. First it makes easier choosing the architecture and/or tuning the hyperparameters of the network configuration. Second, it can be used to shift some of the computational burden of network training to the time before an expert is called in to perform labelling, and make more effective use of his or hers time.

A qualitative examination of the pretraining results shows that some class structure is visible after pretraining (see examples in Figure 7). No identifiable features have been noticed when investigating pretraining images when associated with better or worse final (after fine-tuning) results. However, the general level of structure visible after pretraining relates to the final performance. The A5 ‘simple’ architecture is best in learning the artificial classes grid and also the worst at the final classification. The other two networks (A3 ‘shallow’ and A9 ‘deep’) have more complex pretraining results and correspondingly better final results. This suggests that the training scheme and/or network architecture functions as a form of regularization that prevents overtraining, and that the pretraining classification result can be possibly used to control pretraining and avoid overtraining too. The ability to isolate class structure in an unsupervised fashion also suggests that this approach can be adapted for image segmentation and clustering.

To provide additional verification, we’ve analysed per-class classification scores for both datasets, using the data from experiment one, and the same Mann–Whitney U with $P < 0.05$. As could be expected, performance gains are unequal, as classes differ with their overlap and general difficulty of classification. However, the individual classes showed improvement in most of the cases. Across 198 tests in 104 cases the improvement was statistically significant; for the remaining cases, in 39 cases the accuracy of 100% was achieved irrespective of pretraining, in 32 cases pretraining improved the mean of the class score. In the remaining cases where pretraining score mean was lower that the reference, the average difference was below two percentage points. The proposed method thus can be viewed as ‘not damaging’ to individual class scores.

Our results confirm the general conclusion that the more layers a network has, the more it gains with pre-training scheme. Recently, a number of works affirm that it is possible to effectively learn low
and high level features within the network in an unsupervised fashion. Our approach could be easily transferred to traditional (RGB) computer vision, improving e.g. the automatic learning scheme [62]. Semi-automatic systems like [63], which use only a part of the annotation, could be made fully unsupervised. Furthermore, we believe this is one approach for self-taught learning [53], that can be helpful in diverse application of deep learning models. We note, however, that optimization would require further studies to address the issue of which layers benefit most of this scheme, i.e. similar to [43]. Our experiments show, that the proposed scheme is largely resistant to the incorrect estimation of the number of classes, hence its parametrization can be considered low-cost. It can be also viewed as a confirmation of traditional software development principle of ‘divide and conquer’, as of even older proverb, ‘divide et impera’.

4. Conclusions

We have presented and verified a simple, yet effective scheme for unsupervised pre-training of DLNN for hyperspectral classification. Our proposition is shown to be effective across a number of architectures and datasets, and thus has wide potential of application in hyperspectral classification.

The absence of training labels requirement provides an important advantage: it shifts the need of expert’s participation and data labelling from the start of the data analysis process to its late stages. This allows for the use of the potentially long time from the acquisition to the start of data interpretation stage for pretraining the network, and decreases the delay between expert’s labelling to getting the classification result. Considering the length of time required to train deep neural networks, this is a significant advantage for their applications. An additional benefit is that multiple unannotated images can be used in the pretraining stage, potentially increasing the robustness of the result.

While our experiments have been confined to the hyperspectral images area, we believe that it can be extended to general image segmentation problem. It also can possibly be generalized for unsupervised pretraining in many vision classification scenarios.

Acknowledgements

This work has been partially supported by the projects: ‘Representation of dynamic 3D scenes using the Atomic Shapes Network model’ financed by the National Science Centre, decision DEC-2011/03/D/ST6/03753 and ‘Application of transfer learning methods in the problem of hyperspectral images classification using convolutional neural networks’ funded from the Polish budget funds for science in the years 2018-2022, as a scientific project under the „Diamond Grant” program, no. DI2017 013847. M.O. acknowledges support from Polish National Science Center scholarship 2018/28/T/ST6/00429. This research was supported in part by PLGrid Infrastructure.

The authors would like to thank Laboratory of Analysis and Nondestructive Investigation of Heritage Objects (LANBOZ) in National Museum in Kraków for providing the pigments dataset, in particular to Janna Simone Mostert for her help in the preparation of paintings and Agata Mendys for acquisition of the dataset. The authors also thank Zbigniew Puchała for help in carrying out statistical analysis of the results. Additionally authors thank to Yu et al [10] for sharing the code.

References

[1] Guolan Lu and Baowei Fei. Medical hyperspectral imaging: a review. Journal of Biomedical Optics, 19:19 – 19 – 24, 2014.

[2] M. Teke, H. S. Deveci, O. Haliloğlu, S. Z. Gürbüz, and U. Sakarya. A short survey of hyperspectral remote sensing applications in agriculture. In 2013 6th International Conference on Recent Advances in Space Technologies (RAST), pages 171–176, 2013.
[3] N. Fox, A. Parbhakar-Fox, J. Moltzen, S. Feig, K. Goemann, and J. Huntington. Applications of hyperspectral mineralogy for geoenvironmental characterisation. Minerals Engineering, 107:63 – 77, 2017.

[4] J.M. Bioucas-Dias, A. Plaza, G. Camps-Valls, P. Scheunders, N.M. Nasrabadi, and J. Chanussot. Hyperspectral remote sensing data analysis and future challenges. IEEE Geoscience and Remote Sensing Magazine, 1(2):6–36, 2013.

[5] P. Ghamisi, J. Plaza, Y. Chen, J. Li, and A. J. Plaza. Advanced spectral classifiers for hyperspectral images: A review. IEEE Geoscience and Remote Sensing Magazine, 5(1):8–32, 2017.

[6] Chein-I Chang. Hyperspectral Imaging: Techniques for Spectral Detection and Classification. Springer, Boston, MA, 2003.

[7] F. Melgani and L. Bruzzone. Classification of hyperspectral remote sensing images with Support Vector Machines. IEEE Transactions on Geoscience and Remote Sensing, 42(8):1778–1790, 2004.

[8] Michal Romaszewski, Przemysław Glomb, and Michał Cholewa. Semi-supervised hyperspectral classification from a small number of training samples using a co-training approach. ISPRS Journal of Photogrammetry and Remote Sensing, 121:60 – 76, 2016.

[9] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, Advances in Neural Information Processing Systems 25, pages 1097–1105. Curran Associates, Inc., 2012.

[10] Shiqi Yu, Sen Jia, and Chunyan Xu. Convolutional neural networks for hyperspectral image classification. Neurocomputing, 219:88–98, 2017.

[11] Mengxin Han, Runmin Cong, Xinyu Li, Huazhu Fu, and Jianjun Lei. Joint spatial-spectral hyperspectral image classification based on convolutional neural network. Pattern Recognition Letters, 2018.

[12] Yonghao Xu, Bo Du, Fan Zhang, and Liangpei Zhang. Hyperspectral image classification via a random patches network. ISPRS Journal of Photogrammetry and Remote Sensing, 142:344 – 357, 2018.

[13] Guangzhe Zhao, Guangyun Liu, Leyuan Fang, Bing Tu, and Pedram Ghamisi. Multiple convolutional layers fusion framework for hyperspectral image classification. Neurocomputing, 2019.

[14] Xichuan Zhou, Nian Liu, Fang Tang, Yingjun Zhao, Kai Qin, Lei Zhang, and Dong Li. A deep manifold learning approach for spatial-spectral classification with limited labeled training samples. Neurocomputing, 331:138 – 149, 2019.

[15] Antonio Plaza, Jon Atli Benediktsson, Joseph W Boardman, Jason Brazile, Lorenzo Bruzzone, Gustavo Camps-Valls, Jocelyn Chanussot, Mathieu Fauvel, Paolo Gamba, Anthony Gualtieri, et al. Recent advances in techniques for hyperspectral image processing. Remote sensing of environment, 113:S110–S122, 2009.

[16] M. Cholewa, P. Glomb, and M. Romaszewski. A spatial-spectral disagreement-based sample selection with an application to hyperspectral data classification. IEEE Geoscience and Remote Sensing Letters, 16(3):467–471, 2019.
[17] Yuliya Tarabalka, Jocelyn Chanussot, and Jón Atli Benediktsson. Segmentation and classification of hyperspectral images using minimum spanning forest grown from automatically selected markers. Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, 40(5):1267–1279, 2010.

[18] Inmaculada Dópido, Jun Li, Antonio Plaza, and Paolo Gamba. Semi-supervised classification of urban hyperspectral data using spectral unmixing concepts. In Urban Remote Sensing Event (JURSE), 2013 Joint, pages 186–189. IEEE, 2013.

[19] H. Wu and S. Prasad. Semi-supervised deep learning using pseudo labels for hyperspectral image classification. IEEE Transactions on Image Processing, 27(3):1259–1270, 2018.

[20] Hongmin Gao, Yao Yang, Sheng Lei, Chenming Li, Hui Zhou, and Xiaoyu Qu. Multi-branch fusion network for hyperspectral image classification. Knowledge-Based Systems, 167:11 – 25, 2019.

[21] Bin Pan, Zhenwei Shi, and Xia Xu. Mugnet: Deep learning for hyperspectral image classification using limited samples. ISPRS Journal of Photogrammetry and Remote Sensing, 145:108 – 119, 2018. Deep Learning RS Data.

[22] L. Windrim, A. Melkumyan, R. J. Murphy, A. Chlingaryan, and R. Ramakrishnan. Pretraining for hyperspectral convolutional neural network classification. IEEE Transactions on Geoscience and Remote Sensing, 56(5):2798–2810, 2018.

[23] S. J. Pan and Q. Yang. A survey on transfer learning. IEEE Transactions on Knowledge and Data Engineering, 22(10):1345–1359, 2010.

[24] W. Li, G. Wu, and Q. Du. Transferred deep learning for anomaly detection in hyperspectral imagery. IEEE Geoscience and Remote Sensing Letters, 14(5):597–601, 2017.

[25] Y. Yuan, X. Zheng, and X. Lu. Hyperspectral image superresolution by transfer learning. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 10(5):1963–1974, 2017.

[26] Bo Du, Liangpei Zhang, Dacheng Tao, and Dengyi Zhang. Unsupervised transfer learning for target detection from hyperspectral images. Neurocomputing, 120:72 – 82, 2013. Image Feature Detection and Description.

[27] Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. Semi-Supervised Learning. The MIT Press, 2006.

[28] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. The handbook of brain theory and neural networks, 3361(10):1995, 1995.

[29] Paul Smolensky. Information processing in dynamical systems: Foundations of harmony theory. Technical report, Colorado Univ at Boulder Dept of Computer Science, 1986.

[30] David H Ackley, Geoffrey E Hinton, and Terrence J Sejnowski. A learning algorithm for boltzmann machines. Cognitive science, 9(1):147–169, 1985.

[31] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. science, 313(5786):504–507, 2006.

[32] Geoffrey E Hinton. Deep belief networks. Scholarpedia, 4(5):5947, 2009.
[33] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. Neural computation, 18(7):1527–1554, 2006.

[34] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. Neural computation, 9(8):1735–1780, 1997.

[35] Rahul Dey and Fathi M Salemt. Gate-variants of gated recurrent unit (gru) neural networks. In 2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS), pages 1597–1600. IEEE, 2017.

[36] Hyungtae Lee and Heesung Kwon. Going deeper with contextual cnn for hyperspectral image classification. IEEE Transactions on Image Processing, 26(10):4843–4855, 2017.

[37] Lichao Mou, Pedram Ghamisi, and Xiao Xiang Zhu. Deep recurrent neural networks for hyperspectral image classification. IEEE Transactions on Geoscience and Remote Sensing, 2017.

[38] Hao Wu and Saurabh Prasad. Convolutional recurrent neural networks for hyperspectral data classification. Remote Sensing, 9(3):298, 2017.

[39] Yushi Chen, Zhouhan Lin, Xing Zhao, Gang Wang, and Yanfeng Gu. Deep learning-based classification of hyperspectral data. IEEE Journal of Selected topics in applied earth observations and remote sensing, 7(6):2094–2107, 2014.

[40] Yangyang Fan, Chu Zhang, Ziyi Liu, Zhengjun Qiu, and Yong He. Cost-sensitive stacked sparse auto-encoder models to detect striped stem borer infestation on rice based on hyperspectral imaging. Knowledge-Based Systems, 168:49 – 58, 2019.

[41] Yanhui Guo, Siming Han, Han Cao, Yu Zhang, and Qian Wang. Guided filter based deep recurrent neural networks for hyperspectral image classification. Procedia Computer Science, 129:219 – 223, 2018. 2017 INTERNATIONAL CONFERENCE ON IDENTIFICATION, INFORMATION AND KNOWLEDGE IN THE INTERNET OF THINGS.

[42] Cheng Shi and Chi-Man Pun. Multi-scale hierarchical recurrent neural networks for hyperspectral image classification. Neurocomputing, 294:82 – 93, 2018.

[43] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2, NIPS’14, pages 3320–3328, Cambridge, MA, USA, 2014. MIT Press.

[44] Hong-Wei Ng, Viet Dung Nguyen, Vassilios Vonikakis, and Stefan Winkler. Deep learning for emotion recognition on small datasets using transfer learning. In Proceedings of the 2015 ACM on International Conference on Multimodal Interaction, ICMI ’15, pages 443–449. ACM, 2015.

[45] M. Xie, N. Jean, M. Burke, D. Lobell, and S. Ermon. Transfer Learning from Deep Features for Remote Sensing and Poverty Mapping. ArXiv e-prints, 2015.

[46] H.-C. Shin, H. R. Roth, M. Gao, L. Lu, Z. Xu, I. Nogues, J. Yao, D. Mollura, and R. M. Summers. Deep Convolutional Neural Networks for Computer-Aided Detection: CNN Architectures, Dataset Characteristics and Transfer Learning. ArXiv e-prints, 2016.

[47] Peicheng Zhou, Gong Cheng, Zhenbao Liu, Shuhui Bu, and Xintao Hu. Weakly supervised target detection in remote sensing images based on transferred deep features and negative bootstrapping. Multidimensional Systems and Signal Processing, 27(4):925–944, 2016.
[48] H. Lyu and H. Lu. A deep information based transfer learning method to detect annual urban dynamics of beijing and newyork from 1984–2016. In 2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS), pages 1958–1961, 2017.

[49] Fan Hu, Gui-Song Xia, Jingwen Hu, and Liangpei Zhang. Transferring deep convolutional neural networks for the scene classification of high-resolution remote sensing imagery. Remote Sensing, 7(11):14680–14707, 2015.

[50] Haobo Lyu, Hui Lu, and Lichao Mou. Learning a transferable change rule from a recurrent neural network for land cover change detection. Remote Sensing, 8(6), 2016.

[51] J. Lin, R. Ward, and Z. J. Wang. Deep transfer learning for hyperspectral image classification. In 2018 IEEE 20th International Workshop on Multimedia Signal Processing (MMSP), pages 1–5, 2018.

[52] Bei Fang, Ying Li, Haokui Zhang, and Jonathan Cheung-Wai Chan. Semi-supervised deep learning classification for hyperspectral image based on dual-strategy sample selection. Remote Sensing, 10(4), 2018.

[53] Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, and Andrew Y. Ng. Self-taught learning: Transfer learning from unlabeled data. In Proceedings of the 24th International Conference on Machine Learning, ICML ’07, pages 759–766. ACM, 2007.

[54] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep Learning. MIT Press, 2016. [http://www.deeplearningbook.org]

[55] B. Liu, X. Yu, P. Zhang, A. Yu, Q. Fu, and X. Wei. Supervised deep feature extraction for hyperspectral image classification. IEEE Transactions on Geoscience and Remote Sensing, 56(4):1909–1921, April 2018.

[56] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. IEEE Trans. Pattern Anal. Mach. Intell., pages 1798–1828, 2013.

[57] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 770–778, June 2016.

[58] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In ECCV, 2016.

[59] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. In International Conference on Learning Representations (ICLR), 2013.

[60] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington, editors, Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, volume 9 of Proceedings of Machine Learning Research, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

[61] Bartosz Grabowski, Wojciech Masarczyk, Przemysław Glomb, and Agata Mendys. Automatic pigment identification from hyperspectral data. Journal of Cultural Heritage, 31:1–12, 2018.

[62] Quoc V. Le, Marc’Aurelio Ranzato, Rajat Monga, Matthieu Devin, Kai Chen, Greg S. Corrado, Jeff Dean, and Andrew Y. Ng. Building high-level features using large scale unsupervised learning. In Proceedings of the 29th International Conference on International Conference on Machine Learning, ICML’12, pages 507–514. Omnipress, 2012.
[63] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR ’14, pages 580–587. IEEE Computer Society, 2014.
(a) Indian Pines dataset in false-colour RGB (bands 50, 27, 17).

(b) Indian Pines ground truth. Note that some classes have been removed to satisfy the sample requirement in experiment (see text).

(c) Pavia University dataset in false-colour RGB (bands 64, 101, 1).

(d) Pavia University ground truth.

(e) Pigment dataset in false-colour RGB (bands 50, 27, 17).

(f) Pigment dataset ground truth.

Figure 3: Datasets used in the experiments. The first two are the classic remote sensing datasets. The third one is used for explanatory analysis of the networks’ behaviours. See text.
(a) Selected classes (with sample counts) used with the Indian Pines dataset. This subset was selected for consistency with [36].

(b) Scheme of creating artificial classes on Pigment dataset. From left: original class labels, classes artificially joined into 2 sets, classes artificially split into 10 sets. Dark rectangles denote background, excluded from the experiment.

Figure 4: Details of the individual dataset preparations (see text).
Figure 5: Sample results from experiment one, Indian Pines dataset. Rows present the three examined architectures (A9, A3 and A5). Columns present the three cases of number of true training samples per class in fine-tuning (5s, 15s and 50s). For each result, the Overall Accuracy (OA), Average Accuracy (AA) and κ coefficient are reported. Isolated grey points mark locations of the training samples, and are excluded from the evaluation.
Figure 6: Sample results from experiment one, Pavia University dataset. The scheme is identical to the Figure 5.
Figure 7: Sample pretraining results, corresponding to the cases presented in Figures 5 and 6. Top row Indian Pines, bottom row Pavia University datasets. Columns present the three architectures studied. Some class structure is visible depending on the dataset and network selected, which can possibly be used to control the pretraining process (see text).