DEEP SELF-TAUGHT LEARNING FOR REMOTE SENSING IMAGE CLASSIFICATION

Anika Bettge, Ribana Roscher, Susanne Wenzel

Remote Sensing, Institute of Geodesy and Geoinformation, University of Bonn, Germany, 53115 Bonn
s7anbett@uni-bonn.de, ribana.roscher@uni-bonn.de, wenzel@igg.uni-bonn.de

ABSTRACT

This paper addresses the land cover classification task for remote sensing images by deep self-taught learning. Our self-taught learning approach learns suitable feature representations of the input data using sparse representation and undercomplete dictionary learning. We propose a deep learning framework which extracts representations in multiple layers and use the output of the deepest layer as input to a classification algorithm. We evaluate our approach using a multispectral Landsat 5 TM image of a study area in the North of Novo Progresso (South America) and the Zurich Summer Data Set provided by the University of Zurich. Experiments indicate that features learned by a deep self-taught learning framework can be used for classification and improve the results compared to classification results using the original feature representation.

Index Terms— self-taught learning, deep learning, archetypal analysis, landcover classification, remote sensing

1. INTRODUCTION

Classification of remote sensing images is an important task for land cover mapping. Recently, deep learning has become a valuable approach particularly for such classification tasks. As already pointed out by [1], the most successful approaches are supervised deep learning frameworks using a huge amount of labeled data for training. However, labeled data are scarce for remote sensing applications. In contrast, huge amounts of unlabeled data are available and easy to acquire.

In our approach we use self-taught learning (STL, [2]) which has turned out as a valuable procedure to the combined exploitation of unlabeled and labeled data, without the constraint that both datasets need to follow the same distribution. Therefore, we can utilize datasets from further scenes and acquisition times for feature/representation learning.

The most common approach to STL is sparse representation (SR), which learns features in an unsupervised way in order to use them for supervised classification. Some approaches use deep sparse representations DSR, which shows improved results over shallow representations. E.g., [3] propose a deep unsupervised feature learning approach, but include only labeled data. The authors of [4] use stacked convolutional autoencoders as well as independent component analysis with non-linearity for learning deep representations. He et al. [5] also use multi-layers of SR to obtain higher-level features. For this they combine a fully unsupervised feature learning procedure with hand-crafted feature extraction and pooling. The deep belief network of [6] benefits from the geometric data structure achieved by the local coordinate coding. They represent all data samples in two stacked layers as sparse linear combination of anchor points. However, so far all deep feature learning approaches do not produce fully interpretable representations.

In this paper, the overall goal is to learn deep features with the help of big amounts of unlabeled data, leading to good classification results and interpretable features, the latter being important for many classification or unmixing tasks [7]. We achieve this by designing a deep framework, called deep STL (DSTL), which combines STL and deep learning concepts. We extend the shallow approach of [8], which shows that high classification accuracies can be achieved by combining STL with archetypal dictionaries. Furthermore, we use the approach of [9] to find archetypes (extreme points of the data distribution), and adapt the dictionary learning to be suitable for our deep learning framework.

2. DEEP SELF-TAUGHT LEARNING

STL uses unlabeled data ${}^u\boldsymbol{X} = [{}^u\boldsymbol{x}_q], q = 1, ..., Q$, training data ${}^{tr}\boldsymbol{X} = [{}^{tr}\boldsymbol{x}_n], n = 1, ..., N$ with labels ${}^{tr}\boldsymbol{y} = [{}^{tr}y_n]$ given, and test data ${}^t\boldsymbol{X} = [{}^t\boldsymbol{x}_p], p = 1, ..., P.$, also with labels ${}^t\boldsymbol{y}$. All data samples consist of M-dimensional feature vectors $\boldsymbol{x} \in \mathbb{R}^M$, and the labels $y \in \{1, ..., c, ..., C\}$, where C is the number of classes. The labels are also represented by target vectors $\boldsymbol{t} = [t_c]$ of length C coding the label with $t_c = 1$ for $y = c$ and $t_c = 0$ otherwise.

We use SR approximating each sample by a linear combination of only a few elements of a dictionary. We initialitize the dictionary following archetypal analysis [10], achieving a sparse data approximation $\boldsymbol{x}_n \approx \boldsymbol{D}\,\boldsymbol{\alpha}_n$, with an $(M \times K)$-dimensional dictionary $\boldsymbol{D}$. To learn the dictionary $\boldsymbol{D} = [\boldsymbol{d}_k]$ we apply simplex volume maximization (SiVM) [9]. This approach finds archetypes as extreme points lying on the convex hull of the unlabeled data set $\{\boldsymbol{d}_k\} \in \{{}^u\boldsymbol{x}_q\}$, where $K \leq Q$. The coefficient vectors $\boldsymbol{\alpha}_n$ are the new SR of the data sam-

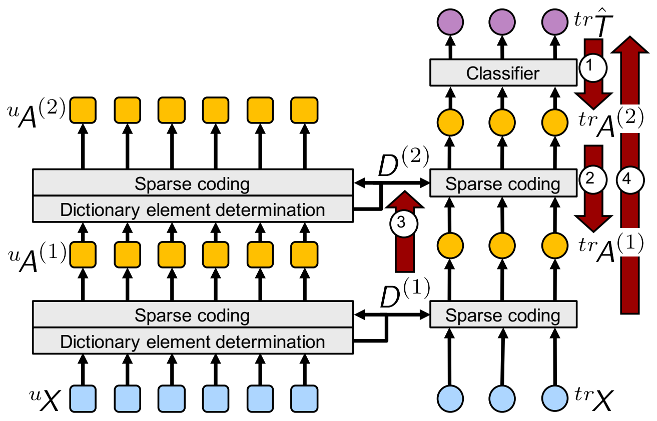

Fig. 1. Structure and update procedure of the DSTL approach: (left block) unlabeled and (right block) labeled data with classifier. The numbers represent the update procedure: ① gradient descent update of the L^{th} training representation; ② backpropagation of the gradient; ③ dictionary update with training representations; ④ learning new training representations and classifier.

ples. We derive these SRs by minimizing $\|D\boldsymbol{\alpha}_n - \boldsymbol{x}_n\|$ subject to non-negativity constraint $\alpha_{kn} \geq 0$ for all k and sum-to-one constraint $\sum_{k=1}^{K} \alpha_{kn} = 1$.

We extend the STL approach to DSTL to learn deep representations. Our network structure is shown in Fig. 1. The left side illustrates the exploitation of unlabeled data, and the right side the used of the labeled data; the data and their representations are symbolized by filled rectangle for the unlabeled and by circles for the labeled data. In general the network consists of L layer, where Fig. 1 shows the structure for $L = 2$. Pre-training is performed layer-wise, so that in each layer $l = 1, ..., L$ the dictionary elements $D^{(l)}$ are determined from ${}^{u}X$ for $l = 1$ and from ${}^{u}A^{(l-1)} = \left[{}^{u}\boldsymbol{\alpha}_q^{(l-1)}\right]$ for $l \neq 1$. Given the dictionaries, we learn ${}^{u}A^{(l)}$ by least square estimation with non-negativity and sum-to-one constraint (left side of Fig. 1). Likewise, we learn ${}^{tr}A^{(l)}$ of the labeled data of the network from $D^{(l)}$. We train the logistic regression model ([11] p. 205-210) with the new features ${}^{tr}A^{(L)}$ predicting conditional probabilities $P(c|{}^{tr}\boldsymbol{x}_n)$, which we can interpret as ${}^{tr}\hat{T} = [{}^{tr}\hat{t}_n]$.

To improve the network the reconstruction error can be minimized by updating the networks parameter. The number of parameters which need to be learned in a 2-layers network are $(M + K^{(2)}) \cdot K^{(1)} + (K^{(2)} + 1) \cdot C$ with $K^{(\cdot)}$ being the number of dictionary elements in the layer $(\cdot)$. It contains the number of dictionary entries in each layer and parameters of the classifier models. In order to learn these parameters we perform the following steps illustrated by ① - ④ in Fig. 1: In the first update step the dictionaries are fixed and only the training representations are updated. Given ${}^{tr}\boldsymbol{t}_n$ and ${}^{tr}\hat{\boldsymbol{t}}_n$, the backpropagation loss function is given by the following equation:

$$J\left({}^{tr}\boldsymbol{\alpha}_n^{(L)}\right) = \frac{1}{2}\|{}^{tr}\boldsymbol{t}_n - {}^{tr}\hat{\boldsymbol{t}}_n\|^2. \tag{1}$$

Here the target vectors ${}^{tr}\boldsymbol{t}_n$ expresses the true membership of the training samples to the classes in the from of the 1-of-C coding scheme, and the ${}^{tr}\hat{\boldsymbol{t}}_n$ is the likewise encoded conditional probability for class membership estimated by our network. With the help of the gradient of this loss function with respect to the training representations ${}^{tr}\boldsymbol{\alpha}_n^{(L)}$ we update the training representations of the last layer:

$${}^{tr}\alpha_{kn}^{*(L)} = {}^{tr}\alpha_{kn}^{(L)} + a\frac{\partial J({}^{tr}\boldsymbol{\alpha}_n^{(L)})}{\partial\, {}^{tr}\alpha_{kn}^{(L)}}. \tag{2}$$

Here the gradient is clipped element-wise to a threshold t_1 to avoid too excessive modifications of the representations [12]. We then backpropagate the gradient through the net in order to update all labeled representations using

$${}^{tr}A^{*(l)} = D^{(l+1)}\, {}^{tr}A^{*(l+1)}, \tag{3}$$

for $l = L - 1, ..., 1$ (① and ②).

In step ③, given the updated training representations, we update the dictionaries $D^{(l)}$ using the gradient descent method as proposed by [3]. To compute the dictionary update, we define a loss function

$$J_D\left(D^{(l)}\right) = \frac{1}{2}\|D^{(l)}\, {}^{tr}A^{*(l)} - {}^{tr}A^{*(l-1)}\|^2, \tag{4}$$

which will be minimized. In the first layer, ${}^{tr}A^{*(0)}$ is given by the original data ${}^{tr}X$. The gradient descent updating rule for the k^{th} dictionary element of the l^{th} layer is given by

$$\boldsymbol{d}_k^{(l)} = \boldsymbol{d}_k^{(l)} - \gamma\left(D^{(l)}\, {}^{tr}A^{(l)} - {}^{tr}A^{*(l-1)}\right)\, {}^{tr}\boldsymbol{\alpha}_k^{(l)}, \tag{5}$$

where γ is the learning rate. Again the gradient is clipped to a threshold t_2. Due to the dictionary updates their entries do not represent raw data samples anymore, which makes them not interpretable. We want to keep the dictionary elements interpretable by restricting them to true data samples. In case a dictionary element has changed sufficiently, we shift it to the nearest neighbor in feature space which contains the set of unlabeled data samples. Step ④ finally readjusts the labeled representations with the updated $D^{(l)}$ by minimizing the reconstruction error of ${}^{tr}X$ and updates the classifier. We iterate steps ① - ④ until convergence of the dictionaries.

3. EXPERIMENTAL SETUP AND RESULTS

In this section we test our DSTL approach for two multispectral image data sets. For this we apply our two layered DSTL to the data sets and compare the accuracy with the results of a simple logistic regression.In Section 3.1 the two data sets are briefly introduced, followed by the data (Sec. 3.2) and experimental setup (Sec. 3.3). Finally, the results are presented in Sec. 3.4.

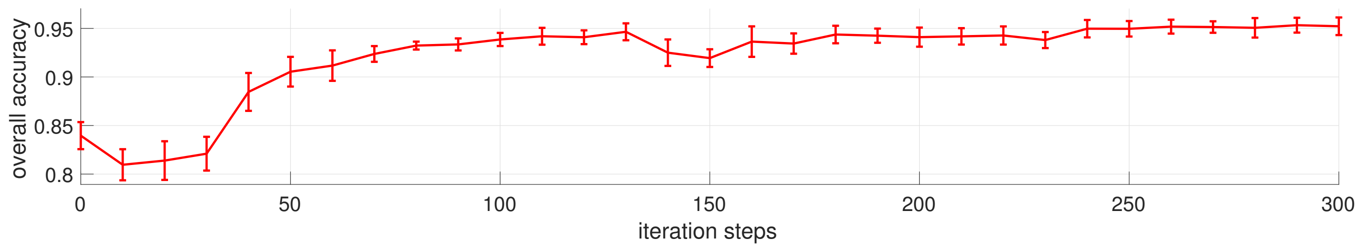

Fig. 2. Average and standard deviation over 10 runs of overall accuracy [%] of the DSTL approach for the Landsat 5 Data Set (right) over up to 300 iterations..

3.1. Data Sets

We use the following two multi-spectral data sets for the testing of our approach:

Landsat 5 Data Set: Our first data set is a multi-spectral Landsat 5 TM image from a study area located in the North of Novo Progresso (South America). It contains data for 6 bands (red, green, blue, Mid-Infrared and two NIR-bands) for $6,962 \times 7,921$ pixels. Parts of the image are labeled (approx. 57,000 pixels) with 6 classes (see Tab. 1). Additionally, we collect about $600,000$ image patches from diverse areas worldwide as unlabeled data samples.

Zurich Summer Data Set: The Zurich Summer Data Set, provided by the University of Zurich [13], contains 20 VHR images of Zurich recorded 2002 by the QuickBird satellite. The images comprise four bands: red, green, blue and NIR with spatial resolution of $61.5\ cm$. They are labeled by 8 urban and periurban classes (roads, buildings, trees, grass, bare soil, water, railways, and swimming pools).

3.2. Data Setup

For both data sets we choose 5×5-pixel image patches, leading to 100-dimensional input feature vectors for the Zurich Summer Data Set and 150-dimensional input feature vectors for the Landsat 5 Data Set. These input vectors are global contrast normalized and then shifted to positive values as input vectors for the DSTL network.

Landsat 5 Data Set: We randomly extract ten sub-data sets with $1,000$ training samples (${}^{tr}X$), around $56,000$ test samples (${}^{t}X$), and $1,000$ validation samples each from the Landsat 5 image. We use the patches (around $600,000$ pixels) as unlabeled samples (${}^{u}X$).

Zurich Summer Data Set: We randomly extract $9,500$ test samples (${}^{t}X$) and 500 validation samples from one of the 20 images, and $1,000$ training samples (${}^{tr}X$) from the remaining 19 images. The $10,000$ unlabeled (${}^{u}X$) samples are randomly selected from all images. The data selection is done 20 times, so that the test and validation data are selected from each image.

3.3. Experimental Setup

In our experiments we create a DSTL with two layers to test if the test accuracy benefits from the DSTL approach over a simple logistic regression.We perform the following experiments on the two data sets:

Landsat 5 Data Set: The DSTL approach is carried through with 20 archetypes in the first layer and 30 in the second.

Zurich Summer Data Set: The DSTL approach is performed with 30 and 40 archetypes for the two layers.

In all experiments the threshold of the gradient clipping is set to $t_1 = t_2 = 0.001$ and the learning rates to $a = \gamma = 1$. The DSTL update is iterated $1,000$ times to find the best dictionary, judged by application to the validation data. We achieve the best results, in terms of classification accuracy, by stacking the representations of all layers for classification, similar to the idea used in denseNet [14].

3.4. Results

In all our experiments we achieve an improvement over the original representations:

Landsat 5 Data Set: Table 1 shows the class-wise, overall, and average test accuracy as well as the Kappa coefficient for the experiment of the Landsat 5 Data Set. Our DSTL approach with stacked representations yields better overall and average accuracies than the original data, but the Kappa coefficient is decreased. The average and standard deviation of the overall accuracy is illustrated in Fig. 2. It becomes obvious, that the average increases over the most iterations and the standard deviation is with maximal $0.03\ \%$ very small.

Zurich Summer Data Set: Table 2 shows that the DSTL with stacked features leads to an improvement in the mean over-all and mean average accuracy. Here also the kappa coefficient increases. Running this experiment with $1,000$ iterations with Matlab version 16b on an Intel Core i5-2400 processor takes ca. 14 hours.

We expect to further improve the results by using larger dictionaries, but this will significantly increase run time.

Table 1. Class-wise accuracies [%], overall accuracy [%], average accuracy [%] and Kappa coefficient (Kappa) obtained by logistic regression using the original features of the Landsat 5 Data Set, and logistic regression on the stacked deep representations of the DSTL approach. The average results over ten runs is given with the standard deviation. The best results are highlighted in bold-print.

	original features	DSTL features
water	90.03 ± 23.06	$\mathbf{97.21 \pm 2.23}$
urban	88.90 ± 5.08	$\mathbf{91.44 \pm 4.94}$
secondary forest	55.43 ± 7.94	$\mathbf{74.29 \pm 3.33}$
pasture	$\mathbf{99.11 \pm 0.83}$	97.81 ± 1.54
burned pasture	$\mathbf{100.00 \pm 0.02}$	99.96 ± 0.08
primary forest	$\mathbf{99.88 \pm 0.07}$	99.34 ± 0.13
overall	94.23 ± 0.90	$\mathbf{96.17 \pm 0.37}$
average	88.89 ± 4.17	$\mathbf{93.34 \pm 1.00}$
Kappa	$\mathbf{0.86 \pm 0.02}$	0.79 ± 0.07

Table 2. Mean results of the logistic regression of the original Zurich Summer Data Set and of the stacked deep representations of the DSTL approach.

	original features	DSTL features
overall accuracy	68.0 %	74.1 %
average accuracy	60.1 %	65.9 %
Kappa	0.54	0.55

4. CONCLUSION

In this work we present a deep self-taught learning framework to combine the advantages of the STL with interpretable dictionaries and the deepness of neural networks. The accuracy is tested with two different multi-spectral image data sets. Further research will deal with a deeper DSTL network and interpretable non-linearity to raise the the accuracy. Altogether, the deep self-taught learning framework profits by the huge amount of unlabeled data, so that the learned deep features improve the results compared to classification results using the original feature representation and are achieved with still interpretable dictionaries.

Acknowledgments

This work has partly been supported by the EC under contract number H2020-ICT-644227-FLOURISH.

5. REFERENCES

[1] Y. Bengio, A. Courville, and P. Vincent, “Representation learning: A review and new perspectives,” *IEEE*, 2013.

[2] R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng, “Self-taught learning: transfer learning from unlabeled data,” in *ICML*, 2007.

[3] Y. Gwon, M. Cha, and HT Kung, “Deep sparse-coded network (dsn),” in *ICPR*, 2016.

[4] R. Kemker and C. Kanan, “Self-taught feature learning for hyperspectral image classification,” *IEEE*, 2017.

[5] K. He, Y.and Kavukcuoglu, Y. Wang, A. Szlam, and Y. Qi, “Unsupervised feature learning by deep sparse coding,” in *SIAM*, 2014.

[6] Y. Lin, T. Zhang, S Zhu, and K. Yu, “Deep coding network,” in *NIPS*, 2010.

[7] C. Römer et al., “Early drought stress detection in cereals: simplex volume maximisation for hyperspectral image analysis,” *Functional Plant Biology*, 2012.

[8] R. Roscher, C. Römer, B. Waske, and L. Plümer, “Landcover classification with self-taught learning on archetypal dictionaries,” in *IGARSS*, 2015.

[9] C. Thurau, K. Kersting, and C. Bauckhage, “Yes we can: simplex volume maximization for descriptive web-scale matrix factorization,” in *CIKM*, 2010.

[10] A. Cutler and L. Breiman, “Archetypal analysis,” *Technometrics*, 1994.

[11] C. M. Bishop, *Pattern recognition and machine learning*, springer, 2006.

[12] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016, `http://www.deeplearningbook.org`.

[13] M. Volpi and V. Ferrari, “Semantic segmentation of urban scenes by learning local class interactions,” in *CVPR*, 2015.

[14] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten, “Densely connected convolutional networks,” *arXiv preprint arXiv:1608.06993*, 2016.