International Doctorate School in

Information and Communication Technologies

Department of Information Engineering and Computer Science

University of Trento

DEEP NEURAL NETWORK MODELS FOR IMAGE CLASSIFICATION AND REGRESSION

Salim MALEK

Advisor: Prof. Farid Melgani, University of Trento

THANKS

All my gratitude and thanks to God, who guided me to this way of science, and gave me the courage and the will to get to this level.

I would like to express my gratitude to my supervisor Dr. Farid Melgani for his academic and moral supports, and his valuable advices.

I thank also Dr. Yakoub Bazi and Dr. Mohamed Lamine Mekhalfi for their encouragement and support.

Finally, I want to thank anyone who helped the cause of this work in any possible way, in particular my dear family.

Abstract

Deep learning, a branch of machine learning, has been gaining ground in many research fields as well as practical applications. Such ongoing boom can be traced back mainly to the availability and the affordability of potential processing facilities, which were not widely accessible than just a decade ago for instance. Although it has demonstrated cutting-edge performance widely in computer vision, and particularly in object recognition and detection, deep learning is yet to find its way into other research areas. Furthermore, the performance of deep learning models has a strong dependency on the way in which these latter are designed/tailored to the problem at hand. This, thereby, raises not only precision concerns but also processing overheads. The success and applicability of a deep learning system relies jointly on both components. In this dissertation, we present innovative deep learning schemes, with application to interesting though less-addressed topics.

In this respect, the first covered topic is rough scene description for visually impaired individuals, whose idea is to list the objects that likely exist in an image that is grabbed by a visually impaired person, To this end, we proceed by extracting several features from the respective query image in order to capture the textural as well as the chromatic cues therein. Further, in order to improve the representativeness of the extracted features, we reinforce them with a feature learning stage by means of an autoencoder model. This latter is topped with a logistic regression layer in order to detect the presence of objects if any.

In a second topic, we suggest to exploit the same model, i.e., autoencoder in the context of cloud removal in remote sensing images. Briefly, the model is learned on a cloud-free image pertaining to a certain geographical area, and applied afterwards on another cloud-contaminated image, acquired at a different time instant, of the same area. Two reconstruction strategies are proposed, namely pixel-based and patch-based reconstructions.

From the earlier two topics, we quantitatively demonstrate that autoencoders can play a pivotal role in terms of both (i) feature learning and (ii) reconstruction and mapping of sequential data.

Convolutional Neural Network (CNN) is arguably the most utilized model by the computer vision community, which is reasonable thanks to its remarkable performance in object and scene recognition, with respect to traditional hand-crafted features. Nevertheless, it is evident that CNN naturally is availed in its two-dimensional version. This raises questions on its applicability to unidimensional data. Thus, a third contribution of this thesis is devoted to the design of a unidimensional architecture of the CNN, which is applied to spectroscopic data. In other terms, CNN is tailored for feature extraction from one-dimensional chemometric data, whilst the extracted features are fed into advanced regression methods to estimate underlying chemical component concentrations. Experimental findings suggest that, similarly to 2D CNNs, unidimensional CNNs are also prone to impose themselves with respect to traditional methods.

The last contribution of this dissertation is to develop new method to estimate the connection weights of the CNNs. It is based on training an SVM for each kernel of the CNN. Such method has the advantage of being fast and adequate for applications that characterized by small datasets.

Contents

List of Figures

List of Tables

Chapter 1

Introduction and Thesis Overview

1.1. Deep Neural Networks

Machine learning is a study field of artificial intelligence (AI) that enables systems to automatically learn and improve from experience without or with little explicit human interference. It focuses on the development of computer programs that can acquire data and build models in order to make better decisions according to prior observations or data records.

According to the adopted learning way, machine learning methods are usually categorized as being either supervised or unsupervised. In supervised learning, a model at hand is learned on a certain data along with its respective labels. Thus, once a model is learned on known data, it can be further fed with another set of data whose labels are unknown. In unsupervised learning, however, prior labels are inaccessible or accessible but unimportant for the application being addressed. This latter, thus, consists in studying how systems can infer functions to define hidden structures from unlabeled data. Semi-supervised learning is another direction whose aim is to exploit a small-sized label data and a large-sized unlabeled data.

A close look at the recent literature would tell that a big focus is being oriented towards deep learning. By contrast to traditional Neural Networks, various layers of neurons in deep learning perform a hierarchical learning of the data representation via non-linear transformations. In other words, the data is passed cumulatively across a long chain of layers (thus, the description deep), where each layer can be fully or partially connected to the preceding one.

Although deep architectures have long existed, the term "deep learning" was first introduced in 2006 by Hinton *et al*. [1], where they showed that a multilayer feedforward neural network can be more efficient by applying pretraining of one layer at a time and considering each layer as an unsupervised Restricted Boltzmann Machine (RBM), by using supervised back propagation for finetuning. One year later, Bengio *et al*. [2] developed the Stacked AutoEncoder (SAE), which is a deep architecture based on the concatenation of many AutoEncoders (AEs). Each AE has three layers, one visible layer (input), one hidden layer and one reconstruction layer with similar size as the input. Another famous deep architecture is the Convolutional Neural Network (CNN) [3]. CNNs are generally composed of many layers, where each layer has two parts, one for convolution (filtering) and one for pooling (subsampling). The chain of convolutional/pooling layers is normally concluded by a regression layer (e.g., logistic regression) in order to discern the class label of the image/object presented as input to the network. CNNs are shaped in a 2D structure, which offers the advantage of directly processing the raw images. This can be achieved with local connections and tied weights followed by subsampling. Yet, it is evident that deep models in general, and CNNs in particular, undergo a heavy processing, which demands highly powerful computation machines.

Figure 1-1, Figure 1-2 and Figure 1-3 show examples of architectures of a RBM, an AE and a CNN.

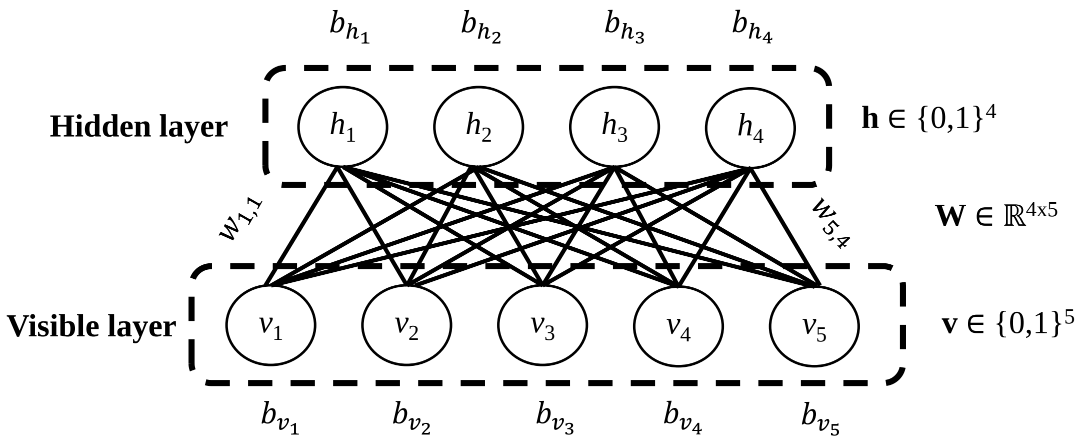


Figure 1-1: Example of a Restricted Boltzman Machine network.

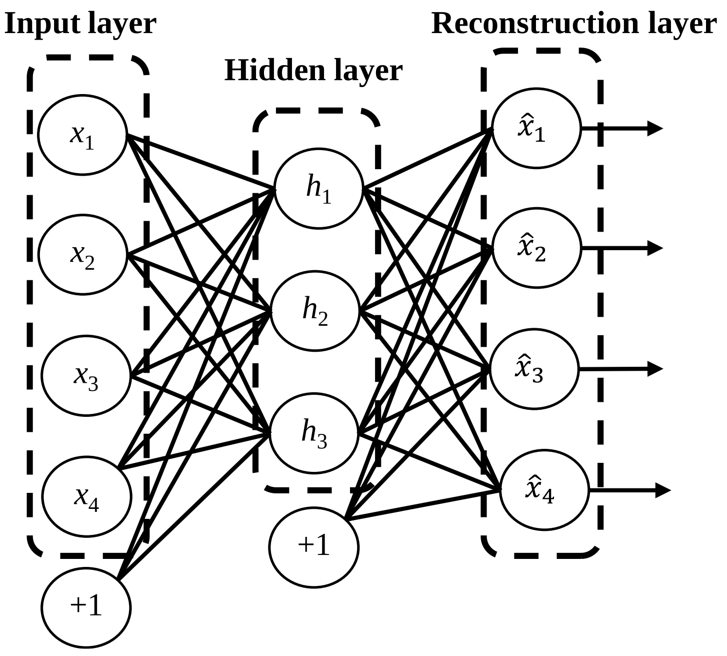


Figure 1-2: Example of an AutoEncoder network.

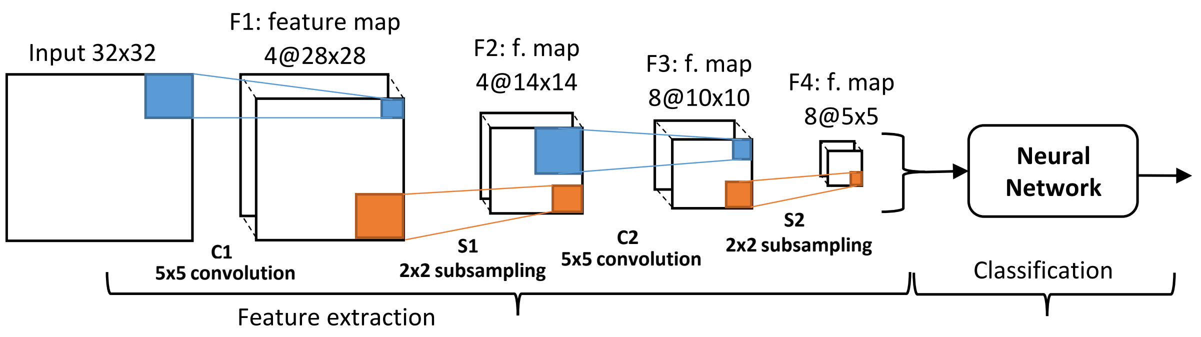


Figure 1-3: Example of a simple CNN architecture.

1.2. Applications and Open Issues

Deep learning techniques have been suggested to solve problems related to diverse research fields. For instance, in robotics, CNN was used in order to recognize the category and estimate the pose of garments hanging from a single point [4] and for real-time human detection with a feature-based layered pre-filter [5]. On the other hand, SAE was used for dimension reduction and combined with particle filter for real-time humanoid robot imitation [6]. In the remote sensing field, Huang *et al.* [7] proposed a new pan-sharpening method based on SAEs to address the remote sensing image fusion problem. Tang *et al.* [8] propose a method for ship detection based on a Stacked Denoising Autoencoder (SDA) for hierarchical ship feature extraction in the wavelet domain and extreme learning machine (ELM) for feature fusion and classification. Chen *et al.* [9] combine SAEs with principal component analysis (PCA) to learn deep features of hyperspectral images, they propose to extract spatial dominated information for the classification and use the PCA to reduce the large input dimension. A logistic regression is used as output layer for the classification. Fang *et al.* [10] use CNNs for scene classification, CaffeNet [11] is used as pretrained model in the classification architecture and finetuning is applied to the pretrained model in order to tailor it to scene classification. Regarding problems of detection and recognition, deep learning was widely used to solve problems of different nature such as speech recognition [12]-[14], face recognition [15]-[18], traffic signs [19]-[21], pedestrian detection and recognition [22]-[24] and detection of various objects [25]-[28]. In the biomedical field, RBM and SAEs were used to solve problems of abnormalities detection and classification of Electrocardiogram signals (ECG) [29]-[32], Electroencephalogram signals (EEG) [33]-[35], Electrooculogram signals (EOG) [36]-[37] and Electromyogram signals (EMG) [38].

Accordingly, from the state-of-the-art reported so far, it is possible to make out that deep learning has established a solid ground in many applications, but in fact still scarcely explored in others. This could be traced back to the fact that deep learning methods are all based on a neural network architecture, and the major objective is to extract high level features in order to apply them for classification problems. To the best of our knowledge so far, all the contributions focus on the single object (class) recognition. That is to say, all the approaches focus on the recognition of one specific category of objects. The problem of multilabeling classification was addressed before in machine learning and can be grouped in two categories. The first category, known as a binary relevance (BR) approach, is based on reducing the initial multilabel classification problem into multiple independent binary classification tasks and each classifier is trained on one class label [39]-[42]. However, such methods are characterized by the high computational costs and also the incapability to identify the correlation between the different classes. By contrast, in for the second category, methods revise and adapt the output formula of the classifier for a multilabel learning problem without the need to transform it into single-label sub-problems. For instance, AdaBoost.MH [43] is adapted from AdaBoost by minimizing Hamming loss and Support Vector Machine (SVM) is revised into Rank-SVM [44] by defining a new approach based on ranking method combined with the predictor.

A second argument constituting this dissertation relates to remote sensing, where only several recent works deal with the problem of clouds based on a deep learning approach [45]-[46]. For instance, all focus on the problem of cloud detection, whilst none, to the best of our awareness, tried to solve the problem of removing clouds and reconstructing the missing area (the area obscured by clouds). Such contribution can bring benefits for many applications, especially with those which deal with multitemporal images.

Another important field of research is chemometrics analyses from spectroscopic data. Usually researchers use reference methods of regression such as partial least squares regression (PLS regression), support vector machines for regression (SVR) and Gaussian process regression (GPR) in order to estimate the concentration of chemical components of interest in a given product. By introducing deep learning, chemometrics can benefit from the advantages that deep methods can provide, especially the capacity of extracting highly discriminative features.

The last concern of this dissertation relates to the manner that deep methods, especially CNNs, estimate the parameters of the network. As a matter of fact, they are all based on the back propagation of errors, which requires big training data and numerous iterations to converge to a satisfactory solution. Such situation involves the need for sophisticated hardware and long processing time. Thus, it would be of particular interest to find another solution to train the network in order to overcome such inconvenient and even handle small training datasets.

1.3. Thesis Objectives, Solutions and Organization

As mentioned earlier, deep learning was used in many research fields and applications and brought important improvements and contributions. However, in some application issues, deep learning methods cannot necessarily be directly applied as they may need some modification and improvement to be suited to the considered problems. To this end, we propose to use deep methods to solve problems related to (i) multilabel classification for scene description for the visually impaired (VI) people, (ii) reconstructing areas obscured by clouds in multispectral images, and (iii) chemometric analysis from spectroscopic data.

Regarding the first problem, the overwhelming majority of existing systems and prototypes pay attention to assisted navigation and obstacle avoidance whilst neglecting the evidently important need of object recognition. While only several solutions have been presented for assisted object recognition for VI individuals, they mainly remain focused on detecting a single object at once. To deal with this problem, we propose a new real-time method to describe the surrounding environment for a blind person based on a coarse image description strategy. i.e. provide the list of objects most likely existing in the scene regardless of their location. This method is based on extracting low-level features from the query image that is acquired by the VI user via an optical camera. The feature selected are: Local Binary Pattern (LBP), Histogram of Oriented Gradient (HOG) and Bag of Words (BoW). Those features are fed to an AutoEncoder Neural Network (AE) in order to extract more discriminative features (high-level features).

Once generated, the new features are fed into a logistic regression layer using a multilabeling strategy as to draw the final outcomes highlighting the objects present in the image of interest.

For the problem of cloud removal and consequent reconstruction of obscured areas in multispectral images, we propose to exploit the strength of the AE networks in the reconstruction phase to restore the missing data. Suppose we have two satellite images of the same area, taken at two different times. Let the first be the cloud-free image (reference image) and the second be the cloud-contaminated image (target image), the AE learning will be slightly modified in such a way that, rather than supposing that the output layer (the reconstruction layer) is equal to the input layer, we consider here that the output is constituted of pixels from the target image, and their corresponding pixels on the reference image are used as input. In other words, we try to find the essential mapping function between the reference and the target images using the AE.

Concerning the question of chemometrics analyses from spectroscopic data, we propose to profit from the advantages of CNNs in extracting high discriminative features from images to apply them on spectroscopic data. Since the concerning data is of one-dimensional nature, the architecture of the CNN is modified and adapted to fulfill spectroscopic data requirements. In particular, filtering and pooling operations as well as equations for training are revisited. Furthermore, we propose to use the particle swarm optimization (PSO) method to train the 1D-CNN.

As per the last concern of this thesis, we propose a new method to calculate the weights of the CNN kernels. The method consists in training an SVM for each kernel in the CNN. The advantage of this new way of training is the possibility to use small training dataset while retaining a satisfactory performance of the network. Furthermore, the training is applied in one pass i.e., just one iteration, which renders it so fast compared to conventional CNNs.

The remainder of this dissertation is outlined as follows. Chapter 2 describes the multilabeling method using the AEs to describe the indoor environment for VI persons. In chapter 3, we give details about the proposed method to reconstruct a missing area covered by clouds in multispectral images using AEs. Chapter 4 details the developed 1D-CNN for chemometric data analysis. In Chapter 5, we present the developed SVM_CNN for multilabel classification. Finally, Chapter 6 concludes the thesis and gives suggestions for possible future improvements.

Finally, we would like to mention that, although deep learning constitutes a denominator of all the addressed topics in this thesis, the applications remain conceptually distinct. Thus, the following chapters were conducted independently. That is, each chapter is self-contained, which eases access to the reader and removes the need of keeping track of the chapters in a sequential order. Nevertheless, we suppose that the reader is familiar with typical concepts related to computer vision and machine learning. Otherwise, the reader is recommended to consult the references provided in each chapter.

1.4. References

[1] G. E. Hinton, S. Osindero and Y. W. Teh, "A Fast Learning Algorithm for Deep Belief Nets," in *Neural Computation*, vol. 18, no. 7, pp. 1527-1554, July 2006.

[2] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, “Greedy layer-wise training of deep networks,” in *Proc. Neural Inf. Process. Syst*., Cambridge, MA, USA, , pp. 153–160, 2007.

[3] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278-2324, Nov 1998.

[4] I. Mariolis, G. Peleka, A. Kargakos and S. Malassiotis, "Pose and category recognition of highly deformable objects using deep learning," *2015 International Conference on Advanced Robotics (ICAR)*, Istanbul, 2015, pp. 655-662.

[5] E. Martinson and V. Yalla, "Real-time human detection for robots using CNN with a feature-based layered pre-filter," *2016 25th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, New York, NY, 2016, pp. 1120-1125.

[6] Y. Kondo and Y. Takahashi, "Real-time whole body imitation by humanoic robot based on particle filter and dimension reduction by autoencoder," *2017 Joint 17th World Congress of International Fuzzy Systems Association and 9th International Conference on Soft Computing and Intelligent Systems (IFSA-SCIS)*, Otsu, 2017, pp. 1-6.

[7] W. Huang, L. Xiao, Z. Wei, H. Liu and S. Tang, "A New Pan-Sharpening Method With Deep Neural Networks," in *IEEE Geoscience and Remote Sensing Letters*, vol. 12, no. 5, pp. 1037-1041, May 2015.

[8] J. Tang, C. Deng, G. B. Huang and B. Zhao, "Compressed-Domain Ship Detection on Spaceborne Optical Image Using Deep Neural Network and Extreme Learning Machine," in *IEEE Transactions on Geoscience and Remote Sensing*, vol. 53, no. 3, pp. 1174-1185, March 2015.

[9] Y. Chen, Z. Lin, X. Zhao, G. Wang and Y. Gu, "Deep Learning-Based Classification of Hyperspectral Data," in *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 7, no. 6, pp. 2094-2107, June 2014.

[10] Z. Fang, W. Li, J. Zou and Q. Du, "Using CNN-based high-level features for remote sensing scene classification," *2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, Beijing, 2016, pp. 2610-2613.

[11] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, and R. Girshick, "Caffe: Convolutional architecture for fast feature embedding," *arXiv preprint arXiv:1408.5093,*2014.

[12] M. Cai, Y. Shi and J. Liu, "Deep maxout neural networks for speech recognition," *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, Olomouc, 2013, pp. 291-296.

[13] S. Kundu, G. Mantena, Y. Qian, T. Tan, M. Delcroix and K. C. Sim, "Joint acoustic factor learning for robust deep neural network based automatic speech recognition," *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Shanghai, 2016, pp. 5025-5029.

[14] P. Harár, R. Burget and M. K. Dutta, "Speech emotion recognition with deep learning," *2017 4th International Conference on Signal Processing and Integrated Networks (SPIN)*, Noida, 2017, pp. 137-140.

[15] D. Menotti *et al*., "Deep Representations for Iris, Face, and Fingerprint Spoofing Detection," in *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 4, pp. 864-879, April 2015.

[16] S. Gao, Y. Zhang, K. Jia, J. Lu and Y. Zhang, "Single Sample Face Recognition via Learning Deep Supervised Autoencoders," in *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 10, pp. 2108-2118, Oct. 2015.

[17] T. Y. Fan, Z. C. Mu and R. Y. Yang, "Multi-modality recognition of human face and ear based on deep learning," *2017 International Conference on Wavelet Analysis and Pattern Recognition (ICWAPR)*, Ningbo, China, 2017, pp. 38-42.

[18] X. Peng, N. Ratha and S. Pankanti, "Learning face recognition from limited training data using deep neural networks," *2016 23rd International Conference on Pattern Recognition (ICPR)*, Cancun, 2016, pp. 1442-1447.

[19] C. Li and C. Yang, "The research on traffic sign recognition based on deep learning," *2016 16th International Symposium on Communications and Information Technologies (ISCIT)*, Qingdao, 2016, pp. 156-161.

[20] R. Q. Qian, Y. Yue, F. Coenen and B. L. Zhang, "Traffic sign recognition using visual attribute learning and convolutional neural network," *2016 International Conference on Machine Learning and Cybernetics (ICMLC)*, Jeju, 2016, pp. 386-391.

[21] F. Lin, Y. Lai, L. Lin and Y. Yuan, "A traffic sign recognition method based on deep visual feature," *2016 Progress in Electromagnetic Research Symposium (PIERS)*, Shanghai, 2016, pp. 2247-2250.

[22] B. Peralta, L. Parra and L. Caro, "Evaluation of stacked autoencoders for pedestrian detection," *2016 35th International Conference of the Chilean Computer Science Society (SCCC)*, Valparaiso, 2016, pp. 1-7.

[23] D. O. Pop, A. Rogozan, F. Nashashibi and A. Bensrhair, "Incremental Cross-Modality deep learning for pedestrian recognition," *2017 IEEE Intelligent Vehicles Symposium (IV)*, Los Angeles, CA, 2017, pp. 523-528.

[24] A. Dominguez-Sanchez, M. Cazorla and S. Orts-Escolano, "Pedestrian Movement Direction Recognition Using Convolutional Neural Networks," in *IEEE Transactions on Intelligent Transportation Systems*, vol. PP, no. 99, pp. 1-9, 2017.

[25] X. Wang, M. Yang, S. Zhu and Y. Lin, "Regionlets for Generic Object Detection," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 10, pp. 2071-2084, Oct. 1 2015.

[26] F. Deng, X. Zhu and J. Ren, "Object detection on panoramic images based on deep learning," *2017 3rd International Conference on Control, Automation and Robotics (ICCAR)*, Nagoya, 2017, pp. 375-380.

[27] B. Tian, L. Li, Y. Qu and L. Yan, "Video Object Detection for Tractability with Deep Learning Method," *2017 Fifth International Conference on Advanced Cloud and Big Data (CBD)*, Shanghai, 2017, pp. 397-401.

[28] F. A. Chang, C. C. Tsai, C. K. Tseng and J. I. Guo, "Embedded multiple object detection based on deep learning technique for advanced driver assistance system," *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*, Boston, MA, 2017, pp. 172-175.

[29] M. A. Rahhal, Y. Bazi, H. AlHichri, N. Alajlan, F. Melgani and R. R Yager, “Deep learning approach for active classification of electrocardiogram signals,” *Information Sciences 345*, 340–354, 2016.

[30] P. R. Muduli, R. R. Gunukula and A. Mukherjee, "A deep learning approach to fetal-ECG signal reconstruction," *2016 Twenty Second National Conference on Communication (NCC)*, Guwahati, 2016, pp. 1-6.

[31] Z. Wu *et al*., "A Novel Features Learning Method for ECG Arrhythmias Using Deep Belief Networks," *2016 6th International Conference on Digital Home (ICDH)*, Guangzhou, 2016, pp. 192-196.

[32] Lin Zhou, Yan Yan, Xingbin Qin, Chan Yuan, Dashun Que and Lei Wang, "Deep learning-based classification of massive electrocardiography data," *2016 IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, Xi'an, 2016, pp. 780-785.

[33] Z. Yin and J. Zhang, "Recognition of Cognitive Task Load levels using single channel EEG and Stacked Denoising Autoencoder," *2016 35th Chinese Control Conference (CCC)*, Chengdu, 2016, pp. 3907-3912.

[34] L. Fraiwan and K. Lweesy, "Neonatal sleep state identification using deep learning autoencoders," *2017 IEEE 13th International Colloquium on Signal Processing & its Applications (CSPA)*, Penang, Malaysia, 2017, pp. 228-231.

[35] H. Xu and K. N. Plataniotis, "Affective states classification using EEG and semi-supervised deep learning approaches," *2016 IEEE 18th International Workshop on Multimedia Signal Processing (MMSP)*, Montreal, QC, 2016, pp. 1-6.

[36] L. H. Du, W. Liu, W. L. Zheng and B. L. Lu, "Detecting driving fatigue with multimodal deep learning," *2017 8th International IEEE/EMBS Conference on Neural Engineering (NER)*, Shanghai, 2017, pp. 74-77.

[37] Bin Xia *et al*., "Electrooculogram based sleep stage classification using deep belief network," *2015 International Joint Conference on Neural Networks (IJCNN)*, Killarney, 2015, pp. 1-5.

[38] A. Ben Said, A. Mohamed, T. Elfouly, K. Harras and Z. J. Wang, "Multimodal Deep Learning Approach for Joint EEG-EMG Data Compression and Classification," *2017 IEEE Wireless Communications and Networking Conference (WCNC)*, San Francisco, CA, 2017, pp. 1-6.

[39] T. Grigorios, and K. Ioannis, "Multi-label classification: an overview," *International Journal of Data Warehousing & Mining*. Vol. 3, no. 3, 2007, pp. 1–13.

[40] Z. Min-Ling, Z. Zhi-Hua, "A review on multi-label learning algorithms", *IEEE Trans. Knowl. Data Eng*., vol. 26, no. 8, pp. 1819-1837, Aug. 2014

[41] J. Read, B. Pfahringer, G. Holmes, and E. Frank, "Classifier Chains for Multi-label Classification," *Machine Learning Journal. Springer*. Vol. 85, no. 3, 2011.

[42] W. W. Cheng, E. Hullermeier, "Combining instance-based learning and logistic regression for multilabel classification", *Mach. Learn*., vol. 76, no. 2/3, pp. 211-225, Sep. 2009.

[43] R. E. Schapire, Y. Singer, "Improved boosting algorithms using confidence-rated predictions", *Mach. Learn*., vol. 37, no. 3, pp. 297-336, Dec. 1999.

[44] A. Elisseeff, J. Weston, "A kernel method for multi-labelled classification", *Adv. Neural Inf. Process. Syst. 14*, vol. 14, pp. 681-687, 2002.

[45] F. Xie, M. Shi, Z. Shi, J. Yin and D. Zhao, "Multilevel Cloud Detection in Remote Sensing Images Based on Deep Learning," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 10, no. 8, pp. 3631-3640, Aug. 2017.

[46] Z. Shao, J. Deng, L. Wang, Y. Fan, N. S. Sumari, and Q. Cheng, "Fuzzy AutoEncode Based Cloud Detection for Remote Sensing Imagery," *remote sensing*, vol. 9, Mar. 2017.

Chapter 2

Real-Time Indoor Scene Description for the Visually Impaired Using AutoEncoder

2.1. Introduction

Strolling around, adjusting the walking pace and bodily balance, perceiving nearby or remote objects and estimating their depth, are all effortless acts for a well-sighted person. That is, however, hardly doable for other portions in society, such as individuals with certain cases of handicap, or visual impairment, which may require different forms of substantial training, and in many situations external physical and/or verbal intervention as to ease their mobility. In dealing with that, numerous attempts at different governmental, institutional, as well as societal spheres have been taking place.

One assistive line, ought to be undertaken by various research institutions, is the providence of either technological designs or end-user products that can help bridging the gap between the conditions being experienced by such disabled people and their expectations. As per the physically handicapped category, a well-established amount of rehabilitation (particularly robotic-based) layouts has been developed so far. However, when it comes to blindness rehabilitation technologies, relatively fewer attentions have been drawn in the relevant literature. As a side note, depending upon the severity of sight loss, vision disability is an umbrella term that encompasses a wide range of progressively inclusive cases, since it could be diagnosed as a: (i) mild impairment, (ii) middle-range impairment, (iii) severe impairment, and ends up to the unfortunate (iv) full blindness. Full sight loss is therefore a serious disability that entails far-reaching ramifications, as it blocks in many cases, the affected individual from conducting his/her daily routines smoothly.

In order to enable the visually disabled persons to move around more easily, several contributions have been proposed in the literature, which are commonly referred to as Electronic Travel Aids (ETAs). By and large, the current ETA methodologies can be identified according to two distinct but complementary aspects, namely: (i) mobility and navigation assistance, that undertakes as a goal assisting visually disabled people to autonomously walk around with the possibility to sense nearby obstacles, and avoid potential collisions thereby, and (ii) object recognition, whose underlying motive is to aid them recognize objects.

Regarding the mobility and obstacle avoidance part, a reasonable amount of works has been put forth thus far. Pundlik et al. [1], for instance, developed a collision detection approach based on a body-mounted camera for visually impaired (VI) people. They proceed by computing the sparse optical flow in the acquired videos and make use of a gyroscopic sensor to estimate the camera rotation. The collision risk is then estimated from the motion estimates. In another work, Balakrishnan et al. [2], presented a system to detect obstacles. The blind individual carries two small cameras mounted on sunglasses. From the captured pair of images, the disparity map is generated and the distances of the objects to the cameras are estimated, which allow for a further decision whether the objects lying ahead of the user make a potential threat. Another navigation aid, named the guide cane, was introduced in [3], which comprises a wheeled housing supplied with a set of ultrasonic sensors, a handle to guide the cane through, and a processing core that processes the ultrasonic signals emitted by the sensors as to infer whether an object is present along the walking path. The concept of the ultrasonic sensors is that they simultaneously emit beams of signals, which in case of obstacles if any, are reflected back. The distance to the obstacles is then deduced based on the time lapse between emission and reflection (commonly termed as time of flight − TOF). The same concept was adopted in [4], where the sensors are placed on a wearable belt instead. Another similar work was put forth in [5]. In this work, the sensors were placed on the shoulders of the user as well as on a guide cane. Another unique contribution proposes exploiting electromagnetic signals instead of ultrasonic ones by using a widespread antenna [6]. However, the capacity of the proposed prototype is limited to 3 m ahead of the user. Having a close look at the literature, it emerges clearly that TOF-based concepts have often been employed and exhibited promising outcomes. The apparent downsides of such methodologies, however, are mainly confined to the dimensions as well as weight of the developed prototypes on the one hand, which may compromise the user’s convenience, and the demanding power consumption (i.e., constant emission/reception of ultrasonic signals) on the other hand.

Regarding the object recognition aspect, introspectively far less contributions can be observed. This might be traced back to the reason that object recognition for the blind might be a harder task to fulfil as compared to navigation and object avoidance. In other words, mobility and object avoidance does not pay attention to the kind of potential objects but to their presence instead, whilst object recognition emphasises on the nature of the nearby objects (i.e., not only their existence). Furthermore, recognizing objects, in camera-shot images, might come at the cost of several challenges such as rotation, scale, and illumination variations, notwithstanding the necessity to carry out such task in a brief time lapse. Nevertheless, different computer-vision techniques have been tailored to tackle this issue. In [7], for instance, a food product recognition system in shopping spaces was proposed. It relies on detecting and recognizing the QR codes of food items by means of a portable camera. Another work considers detecting and recognizing bus line numbers for the VI [8]. Banknote recognition has also been addressed in [9]. Staircases, doors, and indoor signage detection/ recognition have been considered in [10–12]. In [13], the authors developed a prototype composed of ultrasonic sensors and a video camera, which is embedded in a smartphone for a real-time obstacle detection and classification. They first extract FAST feature points from the image and track them with a multiscale Lucas-Kanade algorithm. Then, in the classification phase, a Support Vector Machine was used to detect one of the four objects defined a priori. Consequently, it can be observed that the scarce amount of works that have been devoted to assisted object recognition for the VI so far, emphasize on detecting/recognizing single classes of objects. On this point, it is believed that extending the process into a multiobject recognition is prone to provide a richer description for the VI people.

Subsequently, posing the case of multiobject recognition in general, the mainstream research line suggests designing as many models as the number of objects of interest and then run those learned models on a given query image as to discern its potential object list. Such paradigm could be of notable efficiency, but it is achievable at the cost of prohibitively large processing overheads, which is not a wise choice if undertaken in the context of assistive recognition for the VI people. Departing from this limitation, Mekhalfi et al. [14] introduced a novel approach called coarse description, which operates on portable camera-grabbed images by listing the objects existing in a given nearby indoor spot, irrespective of their location in the indoor space. Precisely, they proposed Scale Invariant Feature Transform (SIFT), Bag of Words (BOW), and Principal Component Analysis (PCA) strategies as a means of image representation. For the sake of furthering the performance of their coarse image description, they suggested another scheme, which exploits Compressive Sensing (CS) theory for image representation and a semantic similarity metric for image likelihood estimation through a bunch of learned Gaussian Process Regression (GPR) models, and concluded that a trade-off between reasonable recognition rates and low processing times can be maintained [15].

In this Chapter, we propose a new method to describe the surrounding environment for a VI person in real-time. We use Local Binary Pattern (LBP) technique, Histogram of Oriented Gradient (HOG) and BOW to describe coarsely the content of the image acquired via an optical camera. In order to improve the state of the art results and deal properly with runtime, we propose to use a deep learning approach, in particular an Auto Encoder Neural Network (AE), to create a new high-level feature representation from the previous low-level features (HOG, BoW and LBP). Once generated, the new feature vectors are fed into a logistic regression layer using a multilabeling strategy as to draw the objects present in the image of concern. This work is a part of a project to guide a VI person in an indoor environment. As validated by the experimental setup, tangible recognition gains and significant speedups have been scored with respect to recent works.

In what follows, Section 2.2 recalls the coarse scene description in brief. Section 2.3 provides short but self-contained conceptual backgrounds of the different methodologies employed for image representation. Section 2.4 outlines the image multilabeling pipeline, which is meant for coarse description. In Section 2.5, we quantify the recognition rates and the processing time and discuss the different pros and cons of the proposed method in the context of indoor scene description. Finally, conclusions are given in Section 2.6.

2.2. Coarse Description

As mentioned earlier, the key idea is to sacrifice the objects' coordinates across the indoor space with the processing requirements, so as to render the recognition process faster yet more convenient for (at least near) real-time scenarios. In this respect, a coarse description of images captured in an indoor environment is adopted. The principle of this approach consists in checking the presence/absence of different objects, which were determined a priori, and turns out to convey the list of the objects that are most likely present in the scene. This approach is based on the multilabeling strategy by creating a binary vector (vector of labels). This vector, as shown in Figure 2-1, indicates which objects are present/absent in its corresponding image. In the training phase, a set of images are captured from the indoor environment and stored with their binary vector. In the classification phase, the proposed method gives in output a multilabel vector referring to the list of existing objects in the scene. This new representation aims to enhance the perception of the VI individual regarding the surrounding environment.

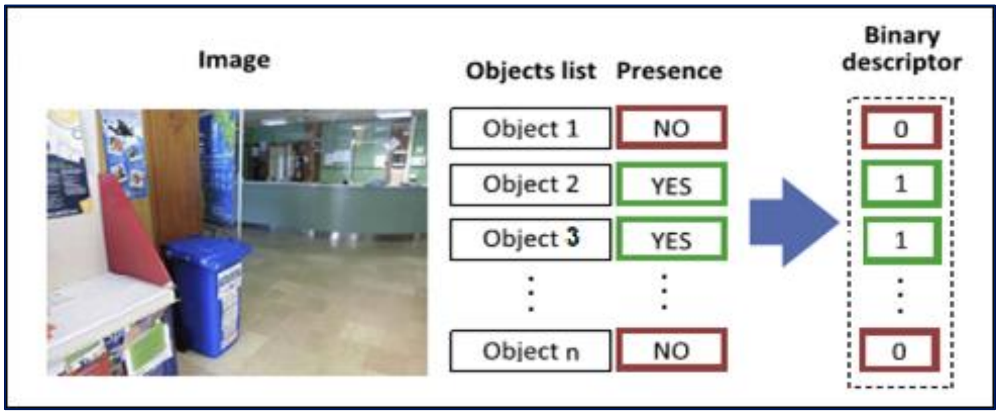


Figure 2-1: Binary descriptor construction for a training image.

2.3. Tools and Concepts

Let us consider a colour image X acquired by a portable digital camera in an indoor environment. Due to several inherent properties of the images, such as illumination, rotation and scale changes, the images cannot be used in their raw form but need to be transformed into an adequate feature space that is able to capture the spatial as well as the spectral variations. Such objective can normally be addressed from three perspectives, namely: (i) shape information, (ii) colour information, and (iii) textural changes. On this point, adopting one feature modality while omitting the others may drop the robustness of the classification algorithm being developed. We therefore resort to a more efficient representation, by making use of all three feature modalities. Precisely, we opt for reputed feature extractors. The first one is the HOG [16] to feature the different shapes distributed over the images. The second one is the BOW [17] based on colour information of the different chromatic channels (BOW_RGB). Finally, the LBP technique in order to express the textural behaviour of the images. As a matter of fact, all the mentioned features can yield interesting results, and this has been documented by previous works, mainly related to object, texture recognition, biometrics as well as remote sensing. In order to further boost their representativeness, we also put forth a feature learning scheme that maps the original feature vectors (derived by means of either feature type mentioned above) onto another lower/higher feature space that offers a better feature representation capability. A well-established feature learning model is the Stacked AutoEncoder (SAE) neural network, or simply AutoEncoder (AE), which constructs a model learned on features pertaining to training images, and then applies it on a given image in order to produce a final image representation.

The final step of the proposed image multilabeling method is the classification of the generated features. This step is performed by appending a logistic regression layer (LRL) to the top of the network. The general diagram of the multilabeling procedure is depicted in Figure 2-2. The following subsections are dedicated to provide basic elaborations of the feature extraction and learning methodologies.

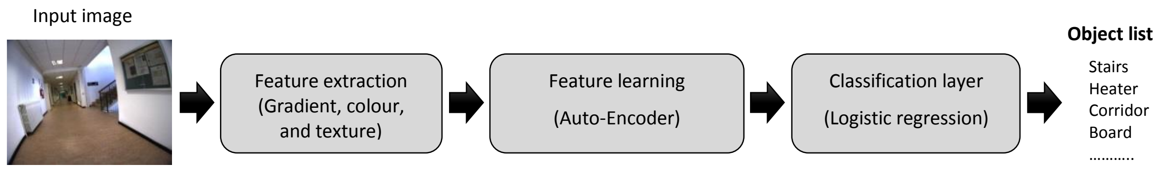

Input image
Feature extraction (Gradient, colour, and texture)
Feature learning (Auto-Encoder)
Classification layer (Logistic regression)
Object list
Stairs
Heater
Corridor
Board
……….

Figure 2-2: Pipeline of the feature learning-based image multilabeling scheme.

2.3.1 Histogram of Oriented Gradient

The HOG was initially aimed at pedestrian detection [16]. Soon later, it was utilized in other applications ranging from object recognition and tracking to remote sensing [18,19]. The basic idea of the HOG is to gather the gradient variations across a given image. Basically, this can be done by dividing the image into adjacent small-sized areas, called cells, and calculating the histograms of the magnitudes/directions of the gradient for the pixels within the cell. Each pixel of the cell is then assigned to one of the bins of the histogram, according to the orientation of the gradient at this point. This assignment is weighted by the gradient of the intensity at that point. Histograms are uniform from either 0 to 180° (unsigned case) or from 0 to 360° (signed case). Dalal and Triggs [16] point out that a fine quantization of the histogram is needed, and they get their best results with a 9-bin histogram. The combination of the computed histograms then forms the final HOG descriptor.

2.3.2 Bag of Visual Words

The BOW is a very popular model in the general computer vision literature. It is usually adopted for its notable property of promoting a concise but rich representation of a generic image. BOW signatures are generally reproduced from a certain feature space of the images, it can be the spectral intensities or alternatively keypoint-based descriptors derived from the images. The BOW is opted for in our work in order to produce a compact representation of the colour attributes of an image. We therefore depart from the chromatic (Red, Green, and Blue channels) values of the images. At first, a basis commonly referred to as codebook is established by gathering all the spectral features of the training images into a matrix. Afterwards, we apply a clustering technique i.e., the K-means clustering, on the built matrix to narrow down its size, which points out a small-sized basis (codebook). Next, the occurrences of the elements (words) of the codebook are observed in the chromatic space of a given image, which turns out to generate a compact histogram whose length equals to the number of the codebook's words. For a more detailed explanation, the reader is referred to [14,17].

2.3.3 Local Binary Pattern (LBP)

Texture is a very important information that can play a key-role in characterizing images and their objects. One of the most popular techniques in this regard is the Local Binary Pattern (LBP) which is a multiresolution, gray-scale, and rotation invariant texture representation. It was first proposed by Ojala et al. [20] and then improved by Guo et al. [21] who introduced a variant called Completed Local Binary Pattern (CLBP), followed by many other variants. The following part gives a brief review about the basic

LBP operator. Given a pixel in the image z(u, v) its LBP code is computed by comparing its intensity value to the values of its local neighbours:

$$LBP_{P,R}(u,v) = \sum_{p=0}^{P-1} H\big(\mathrm{z}(u,v) - \mathrm{z}(u_p,v_p)\big)2^p \tag{2.1}$$

where $\mathrm{z}(u_p, v_p)$ is the grey value of its *p*th neighbouring pixel, *P* is the total number of neighbours, *R* is the radius of the neighbourhood and *H*(·) is the Heaviside step function.

The coordinates of the neighbour $\mathrm{z}(u_p, v_p)$ are: $u_p = u + R\cos\left(\frac{2\pi p}{P}\right)$ and $v_p = v - R\sin\left(\frac{2\pi p}{P}\right)$. If the neighbors do not fall at integer coordinates, the pixel value is estimated by interpolation. Once the LBP label is constructed for every pixel $\mathrm{z}(u,v) \in \mathcal{R}_\mathrm{i}$, a histogram is generated to represent the texture region as follows:

$$Hist(k) = \sum_u \sum_v \delta\big(LBP_{P,R}(u,v), k\big), k \in [0, N_{bins}] \tag{2.2}$$

where N_{bins} is the number of bins and δ is the delta function.

In order to give more robustness for LBP and make it more discriminative, a similar strategy to the HOG method is applied. First, the image is divided into cells and the LBP is calculated for each cell. Then, the computed LBPs are combined to form the final LBP descriptor.

2.3.4 AutoEncoder Networks (AE)

The AE is at the basis a neural network architecture characterized by one hidden layer. It has then three layers, one visible layer of size n, one hidden layer of d nodes and one reconstruction layer with n nodes. Let $\mathbf{x} \in \mathcal{R}^n$ be the input vector, $\mathbf{h} \in \mathcal{R}^n$ the output of the hidden layer and $\hat{\mathbf{x}} \in \mathcal{R}^n$ the output of the AE (reconstruction of **x**). *d* can be inferior or superior to *n*. In the former case (i.e., $d < n$), the AE performs feature reduction. In the latter case, however, it performs an over-complete representation [22].

As can be shown in Figure 2-3, the output of the hidden and reconstruction layers can be calculated using the following equations:

$$\mathbf{h} = f(\mathbf{W}\mathbf{x} + \mathbf{b}) \tag{2.3}$$

$$\hat{\mathbf{x}} = f(\mathbf{W}'\mathbf{h} + \mathbf{b}') \tag{2.4}$$

Where *f*(.) is a non-linear activation function, **W** and **b** are the $d \times n$ weight matrix and the bias vector of dimension *d* of the encoding, and $\mathbf{W}'$ and $\mathbf{b}'$ are the $n \times d$ weight matrix and the bias vector of dimension n of the decoding part.

The parameters ($\mathbf{W}$, $\mathbf{W}'$, $\mathbf{b}$ and $\mathbf{b}'$) can be estimated by minimizing a cost function through a back-propagation algorithm [23]:

$$\underset{\mathbf{W},\mathbf{W}',\mathbf{b},\mathbf{b}'}{\operatorname{argmin}} [\, L(\mathbf{x}, \hat{\mathbf{x}})] \tag{2.5}$$

The loss function $L(\mathbf{x}, \hat{\mathbf{x}})$ adopted in this work is the squared error i.e., $\|\mathbf{x} - \hat{\mathbf{x}}\|^2$. After finding the optimal values of weights and biases, we proceed by removing the last layer (i.e., reconstruction) with its corresponding parameters ($\mathbf{W}'$ and $\mathbf{b}'$). The layer ‘h’ therefore contains a new feature representation, which can be directly used as inputs into a classifier, or alternatively fed into another higher layer to generate deeper features.

In our case, we add a multinomial logistic regression layer (LRL), known also as softmax classifier, at the end of the encoding part to classify the produced feature representations. The choice of using a LRL is justified by its simplicity and the fact that it does not require any parameter tuning. The LRL is trained by adopting the output of the encoding part (the new feature representation) as input, and the corresponding binary vector as target output.

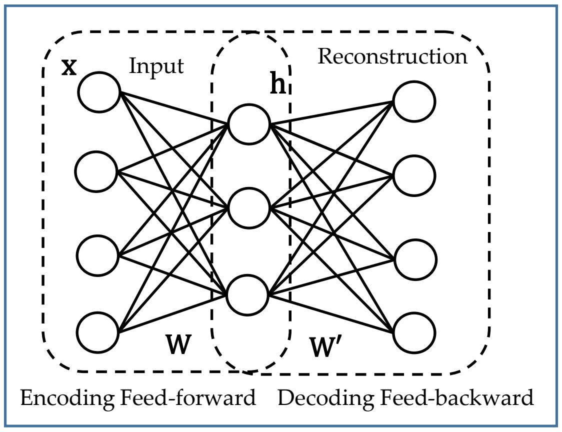


Figure 2-3: One layer architecture of an AE.

2.4. Feature Fusion

As described so far, three types of features are made use of in this work (HOG, BOW_RGB, and LBP). In order to further improve the classification efficiency, we propose three distinct feature fusion schemes. The first one is a stacked fusion, which consists of extracting the three feature vectors from a given image and then stack them up to form a global feature vector. This latter is injected as an input to an AE topped by a logistic regression layer (LRL). The general diagram of the stacked fusion is observed in Figure 2-4.

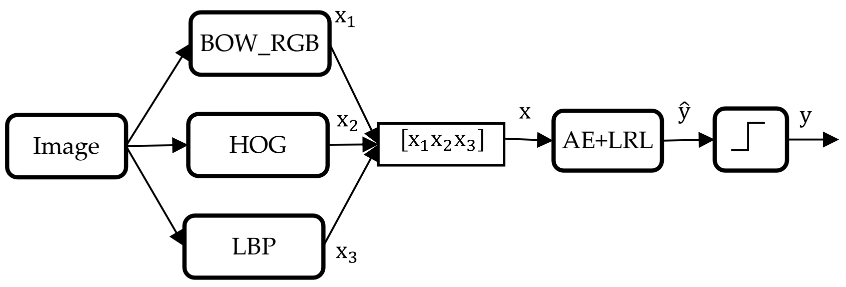


Figure 2-4: Diagram of the first fusion strategy (Fusion 1) based on a low-level feature aggregation.

The second technique is a parallel fusion, as shown in Figure 2-5, which proceeds by feeding each type of feature into an individual AE model, followed by concatenating the learned features to form a single vector. This latter is set as input to another AE model that is connected to a LRL that outputs the final classification results. It is worth to mention that the two blocks composing the autoencoders are trained separately.

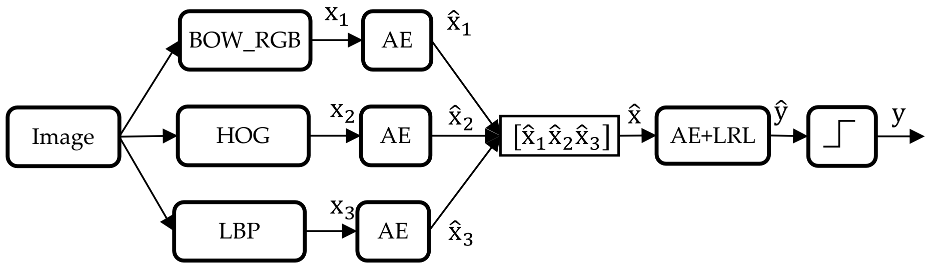


Figure 2-5: Diagram of the second fusion strategy (Fusion 2) based on a AE induced-level aggregation.

The third method is based on a linear sum of the individual decisions of the three types of features. In other words, each feature vector is fed into a separate AE model topped by a LRL. The outputs of each LRL are then averaged to come down to a single real-valued output, which is subsequently thresholded to force its values to either one or zero. Figure 2-6 gives an illustration of the fusion procedure.

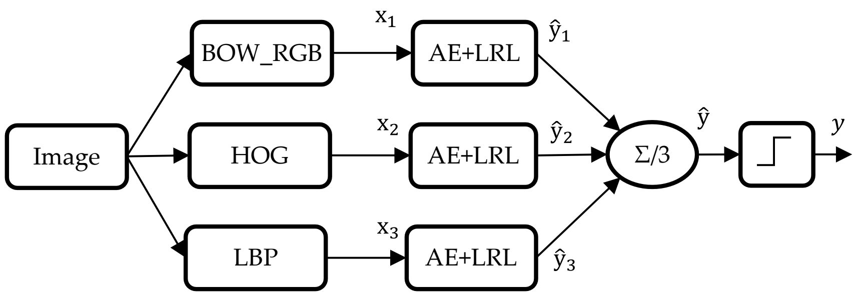


Figure 2-6: Diagram of the third fusion strategy (Fusion 3) based on a decision-level aggregation.

2.5. Experimental Results

2.5.1 Description of the Wearable System

The developed method is part of a complete prototype which is composed of two parts. The first part is the guidance system, which is responsible of guiding a visually impaired person across an indoor environment from an initial point to a desired destination taking into account the avoidance of the different static and/or dynamic obstacles. The second part is the recognition system, which is meant to describe the indoor site for the blind individual to give him better ability to sense the nearby surrounding environment by providing him with a list of existing objects. Regarding the hardware, the wearable system is composed of a laser range finder for detecting and determining objects distance to the user, a portable CMOS camera model UI-1240LE-C-HQ (IDS Imaging Development Systems, Germany) equipped with a LM4NCL lens (KOWA, Japan), a portable processing unit which can be a laptop, a tablet or a smartphone and a headset for voice input and audio output. The user controls the system by giving vocal instructions (i.e., specific keywords) via a microphone and receives information (e.g., list of objects) vocally synthesized through the earphone. All the hardware is mounted on a wearable jacket as can be seen on Figure 2-7. The design of the entire prototype was performed by taking into consideration the feedbacks we received from VI persons, in particular regarding interfacing and exploitation.

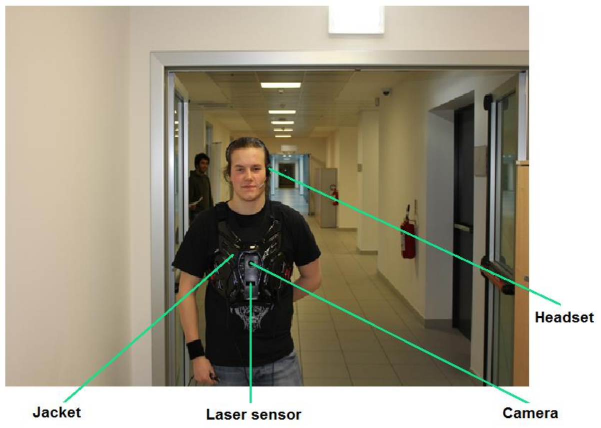


Figure 2-7: View of the wearable prototype with its main components.

2.5.2 Dataset Description

The set of images used in this work was acquired by means the CMOS UI-1240LE-C-HQ camera, with KOWA LM4NCL lens, which is carried by a wearable lightweight shield. The images were shot at two different indoor spaces within the faculty of science at the University of Trento (Italy). The size of each image is 640 × 480. The first ensemble amounts to a total of 130 images, which was divided into 58 training and 72 testing images. The second set accounts for 131 images, split up into 61 training images, and 70 for testing purposes. It is noteworthy that the training images for both datasets were selected in such a way to cover all the predefined objects in the considered indoor environments. To this end, we have selected the objects deemed to be the most important ones in the considered indoor spaces. Regarding the first dataset, 15 objects were considered as follows: ‘External Window’, ‘Board’, ‘Table’, ‘External Door’, ‘Stair Door’, ‘Access Control Reader’, ‘Office’, ‘Pillar’, ‘Display Screen’, ‘People’, ‘ATM’, ‘Chairs’, ‘Bins’, ‘Internal Door’, and ‘Elevator’. Whereas, for the second dataset, the list was the following: ‘Stairs’, ‘Heater’, ‘Corridor’, ‘Board’, ‘Laboratories’, ‘Bins’, ‘Office’, ‘People’, ‘Pillar’, ‘Elevator’, ‘Reception’, ‘Chairs’, ‘Self Service’, ‘External Door’, and ‘Display Screen’.

2.5.3 Evaluation Metrics and Parameter Setting

For evaluation purposes, we use the well-known sensitivity (SEN) and specificity (SPE) measures:

$$\mathrm{SEN} = \frac{True\ Positive}{True\ Positive + False\ Negative} \tag{2.6}$$

$$\mathrm{SPE} = \frac{True\ Negative}{True\ Negative + False\ Positive} \tag{2.7}$$

The sensitivity expresses the classification rate of real positive cases i.e., the efficiency of the algorithm towards detecting existing objects. The specificity, on the other hand, underlines the tendency of the algorithm to detect the true negatives i.e., the non-existing objects. In general, those two quantities reflect opposite measures. In other words, the more the method tends to detect existing objects (high True-Positive which entails higher sensitivity) the more is exposed to make wrong detections (high False-Positive which implies low specificity). In order to make a trade-off between the two measures and to make an adequate comparison of results with respect to the state-of-the-art methods, we propose to further include the average of both:

$$\mathrm{AVG} = \frac{\mathrm{SEN+SPE}}{2} \tag{2.8}$$

We set the parameters of the three feature extractors as follows.

- For HOG features, we set the number of bins to 9 and the size of the cells to 80, which gives a HOG feature vector of size 1260 (recall that the size of the image is 640 × 480).
- Regarding the BOW_RGB, the number of centroids i.e., ‘K’ of the K-means clustering is set to 200, which was observed as the best choice among other options.
- For the LBP, we set R = 1, P = 8, and size of the cells = 80, which produces a LBP feature vector of length 480 bins.

In input layer, all values are normalised between 0 and 1. Regarding output, In fact, the LRLs point out real values. In order to force them to ones or zeroes, a thresholding must take place. Therefore, to determine the most convenient threshold, we applied a 5-folds cross validation technique on the training dataset, with 4 folds chosen randomly as training and the last one as test. The threshold values range from 0.1 to 0.9 with a step of 0.1. We repeated the experiments 5 times, each time with different random permutation. The results presented in Figure 2-8, refer to the average of the sensitivity and the specificity (along with their average) by means of the first fusion method. By observing those figures, SEN and SPE exhibit opposite behaviours as the threshold value increases. On average between SEN and SPE, a threshold of 0.3 stands out as the best option, which will be adopted in the remaining experiments.

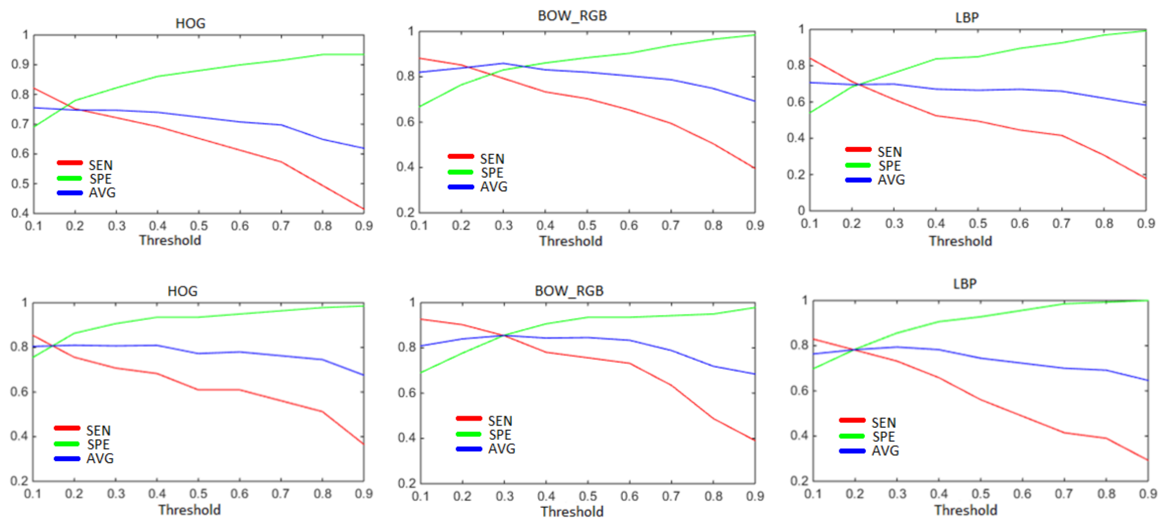


Figure 2-8: Impact of the threshold value on the classification rates using the three feature types. Upper row for Dataset 1, bottom row for Dataset 2.

2.5.4 Results

We first report the results pointed out by using the three types of features individually. We tried many configuration by changing the size of the hidden layer from 100 to 1000 nodes by a step of 100. Ultimately, 300 nodes turned out to be the best choice for all the features. It can be observed from Table 2-1 that on Dataset 1, the three features perform closely, with a slight improvement being noticed with BoW_RGB. On dataset 2, however, the BoW_RGB outperforms, by far, the remaining two, which was expected beforehand as this dataset particularly manifests richer colour information than the former one. Nevertheless, the yielded rates are quite reasonable taking into account the relatively large number of objects considered in this work, besides other challenges such as scale and orientation changes.

Table 2-1: Obtained recognition results using single features.

Dataset	**Dataset 1**			**Dataset 2**		
Method	**HOG**	**BoW_RGB**	**LBP**	**HOG**	**BoW_RGB**	**LBP**
SEN (%)	76.77	79.77	76.77	72.73	88.64	81.82
SPE (%)	82.16	82.9	80.07	88.67	90.24	86.27
AVG (%)	79.46	81.33	78.42	80.7	89.44	84.04

Coming to the fusion scenarios, an interesting point to initiate with is the determination of the optimal size of the hidden layer (i.e., number of hidden units). Different architectures have been explored. Precisely, we tried out values within the range of 100–1000 with a step of 100. Ultimately, the first fusion technique ended up having 900 neurons as an optimal choice, whilst the remaining two strategies pointed out their best at 500 and 300, respectively. It is to note that all the values within 100–1000 have pointed out nearby performances. The earlier optimal parameters will therefore be adopted in what follows.

The classification results of the fusion schemes are summarized in Table 2-2 and examples of results obtained for some query images are provided in Figure 2-9 for both datasets. As a first remark, it can be spotted that significant gains have been introduced with respect to using individual features (Table 2-1), which strengthens the assumption that fusing multiple features is likely to be advantageous over individual feature classification scenarios.

Table 2-2: Classification outcomes of all the fusion schemes.

Dataset	Dataset 1			Dataset 2		
Method	SEN (%)	SPE (%)	AVG (%)	SEN (%)	SPE (%)	AVG (%)
Fusion 1	79.40	87.45	83.42	85.00	91.80	88.40
Fusion 2	80.89	87.69	84.29	87.27	91.56	89.41
Fusion 3	89.51	81.30	85.40	90.00	90.12	90.06

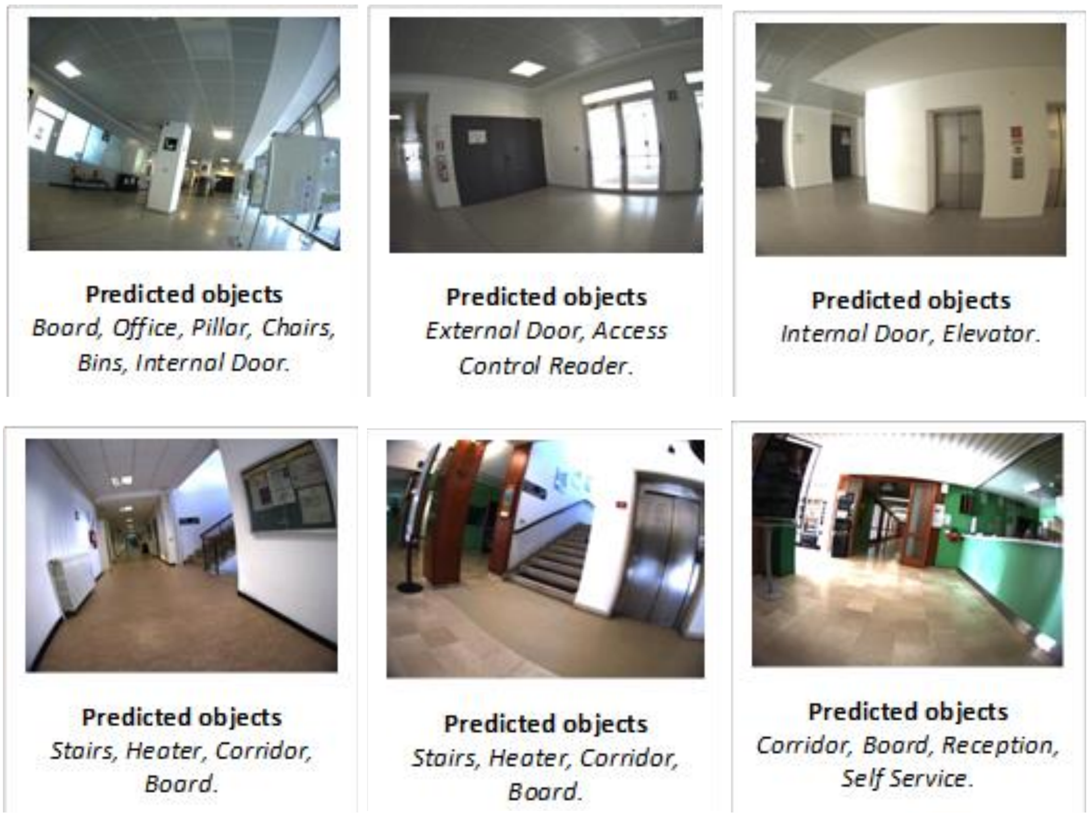

Predicted objects
Board, Office, Pillar, Chairs, Bins, Internal Door.
Predicted objects
External Door, Access Control Reader.
Predicted objects
Internal Door, Elevator.
Predicted objects
Stairs, Heater, Corridor, Board.
Predicted objects
Stairs, Heater, Corridor, Board.
Predicted objects
Corridor, Board, Reception, Self Service.

Figure 2-9: Example of results obtained by the proposed multilabeling fusion approach for both datasets. Upper row for Dataset 1, and lower one for Dataset 2.

Another observation is that the average classification rate gradually increases from Fusion 1 all the way to Fusion 3, with minor disparities. Moreover, the first two strategies seem to favour the SPE over the SEN on both datasets, while Fusion 3, which performs fusion at decision level, favours SEN on

Dataset 1 and exhibits a better SEN-SPE balance on Dataset 2. As a matter of fact, choosing between SEN or SPE depends upon the application being addressed. In our case, we will privilege SEN as we think it is more important to provide information on the presence of objects (even if it generates some false positives) rather than on the absence of objects. For such purpose, late fusion of individual decisions (i.e., Fusion 3) has proved to be a more efficient option than feature-level fusion (i.e., the first two schemes), with a tendency to score higher or equal SEN rates with respect to SPE.

For the sake of comparison of Fusion 3 strategy with state-of-the-art methods, we considered the contribution made in [15], namely the Semantic Similarity-based Compressed Sensing (SSCS) and the Euclidean Distance-based Compressed Sensing (EDCS) techniques, and also three different pretrained Convolutional Neural Networks (CNNs) which are ResNet [24], GoogLeNet [25] and VDCNs [26]. As shown in Table 2-3, for Dataset 1, our strategy outperforms largely the reference work in [15] with at least 10% of improvement and between 2% to 5% compared to the three pretrained CNNs. Moreover, our method offers the advantage of yielding far higher SEN. Both observations can be traced back to two considerations. On the one hand, the work put forth in [15] makes use of a small-sized dictionary of learning images to represent a given image by means of a compressive sensing-based approach, which might succeed in representing an image that has good matches in the dictionary but may fail when it comes to an (outlier) image that has no match in the dictionary. On the other hand, the proposed approach proceeds by extracting robust features capable to capture different variations across the images, followed by a customized feature learning step that furthers their discrimination capacity, which is ultimately reflected on higher classification rates as the results tell. The same observations apply for Dataset 2 as seen in Table 2-4.

Table 2-3: Comparison of classification rates on Dataset 1.

Method	**SEN (%)**	**SPE (%)**	**AVG (%)**
SSCS	79.77	66.54	73.15
EDCS	69.66	80.19	74.92
ResNet	66.29	**94.46**	80.38
GoogLeNet	67.04	94.22	80.63
VDCNs	71.91	**94.46**	83.19
Ours	**89.51**	81.3	**85.40**

Table 2-4: Comparison of classification rates on Dataset 2.

Method	**SEN (%)**	**SPE (%)**	**AVG (%)**
SSCS	75	74.09	74.54
EDCS	70	90.12	80.06
ResNet	68.18	96.75	82.46
GoogLeNet	72.27	**97.11**	84.69
VDCNs	81.82	96.39	89.10
Ours	**90.00**	90.12	**90.06**

An interesting fact to point out is that the size of the images has a direct influence on the processing time. Therefore, an option was to scale down the image resolution from full size (640 × 480), to its half (320 × 240), then (128 × 96), and finally (64 × 48), which respectively define a 100%, 50%, 20%, and

10% of the original size. The classification results in terms of AVG accuracy are shown in Table 2-5 and Table 2-6 for both datasets, respectively. It can be observed that the accuracy does not manifest drastic changes as the image size drops. In fact, there are instances where the smallest resolutions introduce slight improvements, which we believe can be interpreted by the fact that, in many images, there are large background surfaces (e.g., walls) that have usually uniform colours and textures, which may not be really useful as salient visual properties by which the images can be discriminated, reducing the image size thereby reduces the size occupied by those backgrounds, which may either maintain or even raise the classification performance.

Table 2-5: Comparison of classification rates on Dataset 1 under different resolutions.

Method	**100%**	**50%**	**20%**	**10%**
SSCS	73.15	73.34	74.52	74.51
EDCS	74.92	74.74	75.11	75.43
ResNet	80.38	79.88	79.32	78.76
GoogLeNet	80.63	81.63	82.52	79.02
VDCNs	83.19	83.37	84.50	84.57
Ours	**85.40**	**86.02**	**86.14**	**86.63**

Table 2-6: Comparison of classification rates on Dataset 2 under different resolutions.

Method	**100%**	**50%**	**20%**	**10%**
SSCS	74.54	74.54	73.91	74.48
EDCS	80.06	80.06	79.30	78.60
ResNet	82.46	82.40	84.69	87.13
GoogLeNet	84.69	84.69	84.74	84.12
VDCNs	89.10	88.71	87.80	88.13
Ours	**90.06**	**90.34**	**90.03**	**90.69**

Besides the classification rates, another important performance parameter is the runtime. For the proposed method, the runtime includes the feature extraction, the prediction and the fusion times. We provide the average processing time per image for both datasets in Table 2-7 and Table 2-8, from which it can be seen that, as expected, the runtime decreases with the image size, with our method being at least four times faster than the best runtime (GoogLeNet) provided by methods of reference. Particularly, 22 milliseconds per image is a very promising time span provided that fifteen objects are targeted in this work. Such processing time is based on a Matlab R2016b implementation, which is subject to be drastically reduced under for instance a C++ implementation. It is also worth mentioning that the number of objects in our work does not impact on the classification process.

Table 2-7: Comparison of average runtime per image on Dataset 1 under different resolutions.

Method	**100%**	**50%**	**20%**	**10%**
SSCS	2.16	1.42	1.22	1.17
EDCS	2.44	1.41	1.1	1.08
ResNet	0.136	0.132	0.131	0.131
GoogLeNet	**0.100**	**0.098**	0.096	0.093
VDCNs	0.300	0.295	0.291	0.288
Ours	1.230	0.200	**0.048**	**0.022**

Table 2-8: Comparison of average runtime per image on Dataset 2 under different resolutions.

Method	**100%**	**50%**	**20%**	**10%**
SSCS	2.66	1.53	1.21	1.17
EDCS	2.69	1.54	1.23	1.2
ResNet	0.136	0.132	0.131	0.131
GoogLeNet	**0.100**	**0.098**	0.096	0.093
VDCNs	0.300	0.295	0.291	0.288
Ours	1.230	0.200	**0.048**	**0.022**

2.6. Conclusions

This chapter presented a scene description (via image multilabeling) methodology meant to assist visually impaired people to conceive a more accurate perception about their surrounding objects in indoor spaces. The idea of the proposed method is promoted around detecting multiple objects at once within a possible short runtime. A key-determinant of our image multilabeling scheme is that the number objects is independent of the classification system, which entails the property of detecting as many objects as desired (depending on the offline setup to be customized by the user) within the same amount of time which amounts for much less than a second in our work.

The multilabeling algorithm exploits feature learning concept by means of an AutoEncoder neural network, which amply demonstrated a significant potential in generating discriminative image representations.

Pros: In the literature, there exist several multi-object recognition methods (but not in the context of visually impaired rehabilitation). Those methods, may show interesting recognition efficiency, but they are dependent on the number of objects considered. By contrast, our method as hinted earlier, does not, which renders it much faster yet more reliable if considered in real-time scenarios. The earlier two points are technically verified in [15], where it was concluded that coarse image description is more adequate in this sense.

Cons: While the aim of the conducted coarse description is to roughly list the present objects as to bridge the gap between the real indoor setup and the image conceived in the visually disabled person's imagination, inferring further information pertaining to the detected object's location in the indoor space remains a vivid endeavour in our future considerations. This, however, may come at the cost of a heavier processing but it is not out of question. To complement this missing component, we suggest to find a way to introduce the depth information (e.g., through Kinect sensors for instance) as a post-operation. Another issue is related to the scalability of the system since it will need to be completely retrained in case the set of predefined objects requires to be modified quantitatively or qualitatively.

2.7. References

[1] S. Pundlik, M. Tomasi, and G. Luo, “Collision Detection for Visually Impaired from a Body-Mounted Camera,” in *2013 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2013, pp. 41–47.

[2] G. Balakrishnan, G. Sainarayanan, R. Nagarajan, and S. Yaacob, “Stereopsis method for visually impaired to identify obstacles based on distance,” in *Third International Conference on Image and Graphics (ICIG’04)*, 2004, pp. 580–583.

[3] I. Ulrich and J. Borenstein, “The GuideCane-applying mobile robot technologies to assist the visually impaired,” *IEEE Trans. Syst. Man Cybern. - Part Syst. Hum.*, vol. 31, no. 2, pp. 131–136, Mar. 2001.

[4] S. Shoval, J. Borenstein, and Y. Koren, “Auditory guidance with the Navbelt-a computerized travel aid for the blind,” *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.*, vol. 28, no. 3, pp. 459–467, Aug. 1998.

[5] M. Bousbia-Salah, M. Bettayeb, and A. Larbi, “A Navigation Aid for Blind People,” *J. Intell. Robot. Syst.*, vol. 64, no. 3–4, pp. 387–400, Dec. 2011.

[6] L. Scalise *et al.*, “Experimental Investigation of Electromagnetic Obstacle Detection for Visually Impaired Users: A Comparison With Ultrasonic Sensing,” *IEEE Trans. Instrum. Meas.*, vol. 61, no. 11, pp. 3047–3057, Nov. 2012.

[7] D. López-de-Ipiña, T. Lorido, and U. López, “BlindShopping: Enabling Accessible Shopping for Visually Impaired People through Mobile Technologies,” in *Toward Useful Services for Elderly and People with Disabilities*, 2011, pp. 266–270.

[8] H. Pan, C. Yi, and Y. Tian, “A primary travelling assistant system of bus detection and recognition for visually impaired people,” in *2013 IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*, 2013, pp. 1–6.

[9] F. M. Hasanuzzaman, X. Yang, and Y. Tian, “Robust and Effective Component-Based Banknote Recognition for the Blind,” *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.*, vol. 42, no. 6, pp. 1021–1030, Nov. 2012.

[10] T. J. J. Tang, W. L. D. Lui, and W. H. Li, “Plane-based detection of staircases using inverse depth,” pp. 1–10, Jan. 2012.

[11] X. Yang and Y. Tian, “Robust door detection in unfamiliar environments by combining edge and corner features,” in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops*, 2010, pp. 57–64.

[12] S. Wang and Y. Tian, “Camera-Based Signage Detection and Recognition for Blind Persons,” in *Computers Helping People with Special Needs*, 2012, pp. 17–24.

[13] B. Mocanu, R. Tapu, and T. Zaharia, “When Ultrasonic Sensors and Computer Vision Join Forces for Efficient Obstacle Detection and Recognition,” *Sensors*, vol. 16, no. 11, Oct. 2016.

[14] M. L. Mekhalfi, F. Melgani, Y. Bazi, and N. Alajlan, “Toward an assisted indoor scene perception for blind people with image multilabeling strategies,” *Expert Syst. Appl.*, vol. 42, no. 6, pp. 2907–2918, Apr. 2015.

[15] M. L. Mekhalfi, F. Melgani, Y. Bazi, and N. Alajlan, “A Compressive Sensing Approach to Describe Indoor Scenes for Blind People,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 25, no. 7, pp. 1246–1257, Jul. 2015.

[16] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*, 2005, vol. 1, pp. 886–893 vol. 1.

[17] Y. Zhang, R. Jin, and Z.-H. Zhou, “Understanding bag-of-words model: a statistical framework,” *Int. J. Mach. Learn. Cybern.*, vol. 1, no. 1–4, pp. 43–52, Dec. 2010.

[18] S. Zhang, X. Yu, Y. Sui, S. Zhao, and L. Zhang, “Object Tracking With Multi-View Support Vector Machines,” *IEEE Trans. Multimed.*, vol. 17, no. 3, pp. 265–278, Mar. 2015.

[19] T. Moranduzzo and F. Melgani, “Detecting Cars in UAV Images With a Catalog-Based Approach,” *IEEE Trans. Geosci. Remote Sens.*, vol. 52, no. 10, pp. 6356–6367, Oct. 2014.

[20] T. Ojala, M. Pietikainen, and T. Maenpaa, “Multiresolution gray-scale and rotation invariant texture classification with local binary patterns,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 971–987, Jul. 2002.

[21] Z. Guo, L. Zhang, and D. Zhang, “A Completed Modeling of Local Binary Pattern Operator for Texture Classification,” *IEEE Trans. Image Process.*, vol. 19, no. 6, pp. 1657–1663, Jun. 2010.
[22] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, “Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion,” *J Mach Learn Res*, vol. 11, pp. 3371–3408, Dec. 2010.
[23] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986.
[24] 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)*, 2016, pp. 770–778.
[25] C. Szegedy *et al.*, “Going deeper with convolutions,” in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1–9.
[26] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” *ArXiv14091556 Cs*, Sep. 2014.

Chapter 3

Reconstructing Cloud-Contaminated Multispectral Images with Contextualized AutoEncoder

3.1. Introduction

Depending on the application, clouds in remotely sensed imagery can be seen as a source of information or noise. In the latter case, which represents the focus of this work, clouds are considered as a serious problem because they can cover partially or completely a region of interest, and thereby reduce the exploitability of the images. Detecting clouds in the images is often performed as a pre-processing step in order to remove them and recover the missing areas. In this work, we focus on reconstructing the missing areas supposing that the cloudy areas in the images are already identified.

In the literature, some contributions start from the hypothesis that clouds are thin and do not obscure completely the reflected signal from Earth. They generally use either monotemporal methods that exploit the different spectral bands of the single image to reconstruct the affected areas or multitemporal ones which deal with a temporal sequence of images acquired over the same location ([1]-[4]).

In order to address the case of opaque clouds, which is the scope of this chapter, Maalouf *et al.* [5] proposed an inpainting technique based on the Bandelet transform and the multiscale geometrical grouping. They presented interesting results but such techniques showed their limits compared to those based on multitemporal prediction [6]. Moreover, since satellite multitemporal imagery over a given area can be regularly acquired, most methods in the literature rely on the exploitation of the temporal dimension. Among the first related contributions, one can cite the work of Liew *et al.* [7]. They generate an ensemble of cloud-free image mosaics by composing several cloudy SPOT and IKONOS satellite images acquired from the same area. In [8], two unsupervised contextual reconstruction methods were proposed. The first method is a linear prediction based on the expectation-maximization (EM) algorithm and the second one is a nonlinear prediction based on a support vector machine (SVM). Tseng *et al.* [9] proposed a method to generate cloud-free mosaic images from multitemporal SPOT images based on a multiscale wavelet-based fusion method in order to ameliorate the transition between two mosaic parts. In [10], Lin *et al.* proposed to clone cloud-free information from a set of multitemporal images by adopting a batch-based reconstruction method formulated as a Poisson equation and solved using a global optimization process. Lorenzi *et al.* [11] proposed to rely on the compressive sensing (CS) theory to reconstruct the area covered by clouds. They developed two common CS solutions, namely, the basis pursuit (BP) and the orthogonal matching pursuit (OMP). A third CS solution based on exploiting the search capabilities of genetic algorithms (GAs) is also introduced.

In this chapter, we describe a new method to recover missing data in multispectral images due to presence of clouds. Specifically, we propose to exploit the strength of the AE networks in the reconstruction phase to restore the missing data. It is worth mentioning that AE networks have been used for general image restoration problems. For example, the authors in [12] combine them with sparse coding for image denoising and blind inpainting. In another work [13], the authors exploit them for the enhancement of natural low-light images. In [14], it is proposed a non-local AE with collaborative stabilization for natural image denoising and super-resolution. In the remote sensing literature, they were successfully applied to image pansharpening [15], object detection [16] and hyperspectral data classification [17]. In the present work, AEs are exploited to address the issue of missing data reconstruction in multispectral images. Given a cloud-free image (source or reference image) and a cloud-contaminated image (target image), the AE learning will be slightly modified in such a way that rather than supposing that the output layer (the reconstruction layer) is equal to the input layer, we consider here that the output is constituted of pixels from the target image and their corresponding pixels on the reference image are used as input. In other words, we try to find the essential mapping function between the source and the target image using the AE [18]. In the training phase, cloud-free pixels from both source and target images are exploited. After that, in the prediction phase, pixels from the reference image corresponding to contaminated pixels in the target image are used in order to reconstruct the missing areas in the second image. Moreover, in order to fix the problem of the empirical tuning of the hidden layer size (which is still an open issue in the definition of an AE architecture), the minimum descriptive length (MDL) criterion [19] in combination with a Pareto front selection method is applied to infer the best AE architecture. It is noteworthy that the trained neural network is contextualized, in the sense that it exploits contextual information of a given missing region to recover only that specific region. Accordingly, it is not intended to generalize to other missing regions for which other (fine-tuned) neural networks will be needed. Compared to [12], our method is different from various points of view: 1) [12] deals with the image denoising and inpainting problems while we focus on multitemporal image reconstruction; 2) [12] makes use of a fixed Stacked AutoEncoder (SAE) architecture (with 3 hidden layers, each of them composed of the same number of hidden nodes set to 5 times the number of input nodes) while our architecture is not stacked (more compact) and the number of hidden nodes is estimated automatically (thanks to MDL criterion); 3) our method is contextualized while [12] is not; and 4) our method is specifically developed for cloud-contaminated remote sensing image reconstruction.

The rest of this chapter is organized as follows. Section 3.2 formulates the problem of the reconstruction of a cloud-contaminated image and describes the two developed methods based on a modified learning of the AE network. The experimental results are presented in Section 3.3. Finally, conclusions are reported in Section 3.4.

3.2. Methodology

Let us consider two multitemporal multispectral images $I^{(1)}$ and $I^{(2)}$ with *nb* bands acquired by an optical sensor and registered over the same geographical area. The two images are acquired at two different dates, which are supposed to be sufficiently close to each other in order to keep similarity concerning the spatial structure. $I^{(1)}$ refers to the cloud-free image (source/reference image) and $I^{(2)}$ refers to the cloudy image (target image). The objective of the proposed method is to reconstruct any area of the target image which is contaminated by clouds. We note that the problem of the detection of clouds is out of scope of this work. We will call the cloudy area in the contaminated image $I^{(2)}$ as target region $Ts^{(2)}$ and its corresponding area in the cloud-free image $I^{(1)}$ as source region $Ts^{(1)}$. The areas surrounding the cloudy region (cloud-free area) in $I^{(2)}$ and $I^{(1)}$ are referred as target training region $Tr^{(2)}$ and source training region $Tr^{(1)}$, respectively.

The underlying idea of the proposed method is to find the transformation $s(\cdot)$ which captures the relationship between the source and the target images. For such purpose, the training regions are used to estimate $s(\cdot)$ and a model is learned such that:

$$Tr^{(2)}=s(Tr^{(1)}) \tag{3.1}$$

Once the mapping model is learned, one can reconstruct the cloudy region in the target image ($Ts^{(2)}$) by applying the estimated transformation $s(\cdot)$ on the reference test region $Ts^{(1)}$.

$$Ts^{(2)}=s(Ts^{(1)}) \tag{3.2}$$

The transformation function $s(\cdot)$ can be learnt using linear or nonlinear models. In multispectral images, the distributions of data are typically complex (multimodal) which makes the simple linear model not the most suitable choice to perform the transformation. As mentioned earlier, various ways have been proposed in the literature for solving this mapping problem. In this work, we propose an alternative solution based on AutoEncoder (AE) neural networks because of their promising capability to reconstruct data in a nonlinear way. More details are given in the next sections.

3.2.1 Pixel-based Reconstruction with AutoEncoder

The AE is basically an artificial neural network architecture that is characterized by one hidden layer. Stacking many AEs (many hidden layers) forms a so-called stacked AE (SAE) which is considered as a deep architecture.

A simple AE has three layers, one visible layer of size *n* (input layer), one hidden layer of *d* nodes and one reconstruction layer (output layer) with *n* nodes. Let $\boldsymbol{x} \in \mathcal{R}^n$ be the input vector, $\boldsymbol{h} \in \mathcal{R}^d$ the outcome of the hidden layer and $\hat{\boldsymbol{x}} \in \mathcal{R}^n$ the output of the AE (the reconstruction of $\boldsymbol{x}$).

In our case, since we are not interested in an auto-reconstruction operation of the AE (encoding/decoding) but in a mapping between two correlated spaces (source and target images), the training of the AE will be slightly modified in order to find the mapping function between the two images ($I^{(1)}$ and $I^{(2)}$).

Let us consider a input vector $\boldsymbol{x}^{(1)} = \left[x_1^{(1)}, x_2^{(1)}, \dots, x_n^{(1)}\right]^{\mathrm{T}}$ which represents a generic pixel from the region $Tr^{(1)}$ and $\boldsymbol{x}^{(2)} = \left[x_1^{(2)}, x_2^{(2)}, \dots, x_n^{(2)}\right]^{\mathrm{T}}$ is its corresponding pixel in $Tr^{(2)}$ (our target), $\hat{\boldsymbol{x}} = [\hat{x}_1, \hat{x}_2, \dots, \hat{x}_n]^{\mathrm{T}}$ is the reconstruction vector (the output of the AE), $\mathbf{W} = [\boldsymbol{w_1}, \dots, \boldsymbol{w_d}]^{\mathrm{T}}$ and $\boldsymbol{b}$ are the $d \times n$ weight matrix and the bias vector of dimension *d* of the encoding part, and $\mathbf{W'} = [\boldsymbol{w'_1}, \dots, \boldsymbol{w'_n}]^{\mathrm{T}}$ and $\boldsymbol{b'}$ are the $n \times d$ weight matrix and the bias vector of dimension *n* of the decoding part. The parameter *n* represents the number of nodes of the input and the reconstruction layers and it equals here to the number of bands of the images, namely *nb*.

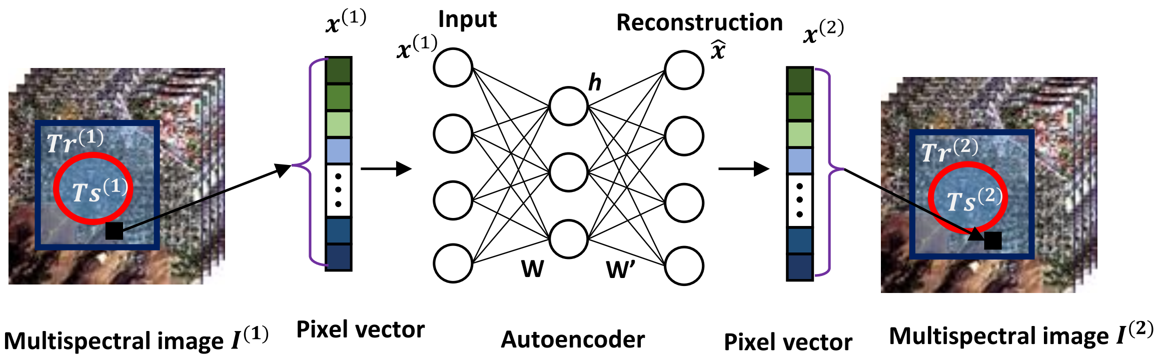


Figure 3-1: Proposed pixel-based AutoEncoding reconstruction method.

As can be shown in Figure 3-1, the output of the hidden and reconstruction layers can be calculated using the following equations.

$$h_{\mathrm{i}} = f\left(\boldsymbol{w}_{\mathbf{i}}\boldsymbol{x}^{(1)} + b_{\mathrm{i}}\right) \qquad (\mathrm{i}=1 \ldots d) \tag{3.3}$$

$$\hat{x}_{\mathrm{i}} = f(\boldsymbol{w'}_{\mathbf{i}}\boldsymbol{h} + b'_{\mathrm{i}}) \qquad (\mathrm{i}=1 \ldots n) \tag{3.4}$$

where $\boldsymbol{w}_{\mathbf{i}}$ and $\boldsymbol{w'}_{\mathbf{i}}$ are the *i*th rows of the weight matrices $\mathbf{W}$ and $\mathbf{W'}$ respectively, b_{i} and b'_{i} are the *i*th elements of the bias vectors $\boldsymbol{b}$ and $\boldsymbol{b'}$ respectively, and $f(\cdot)$ is a nonlinear activation function. Typically, a sigmoid activation function is used, which is expressed by the equation below:

$$f(z) = \frac{1}{1+e^{-z}} \tag{3.5}$$

The parameters ($\mathbf{W}, \mathbf{W'}$, $\boldsymbol{b}$ and $\boldsymbol{b'}$) can be estimated by minimizing a cost function $L(\cdot)$ which quantifies the error between the target $\boldsymbol{x}^{(2)}$ and the output $\hat{\boldsymbol{x}}$ of the AE.

$$\underset{\mathbf{W},\mathbf{W'},\boldsymbol{b},\boldsymbol{b'}}{\operatorname{argmin}}\,[L(\boldsymbol{x}^{(2)}, \hat{\boldsymbol{x}})] \tag{3.6}$$

In this work, we adopt the squared error function, i.e. $L(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \|\boldsymbol{x} - \hat{\boldsymbol{x}}\|^2$. Therefore, the minimization problem in (3.6) becomes:

$$\underset{\mathbf{W},\mathbf{W'},\boldsymbol{b},\boldsymbol{b'}}{\operatorname{argmin}}\,[\|\boldsymbol{x} - \boldsymbol{f}[\mathbf{W'}\boldsymbol{f}(\mathbf{W}\boldsymbol{x} + \boldsymbol{b}) + \boldsymbol{b'}]\|^2] \tag{3.7}$$

where $\boldsymbol{f}(\cdot)$ is the vector representation of the activation function $f(\cdot)$ defined in (3.5).

The computation of this objective function is performed on the available training samples and optimized according to a backpropagation method which relies on its gradient calculation. In particularly, we use the stochastic gradient descent method to optimize the cost function in (3.7) [20]. Since the weights are updated for each training sample, this method has proved faster, more reliable, and less prone to reach bad local minima than standard gradient descent.

3.2.2 Patch-based reconstruction strategy

In remote sensing images, it can be reasonably expected that neighbor pixels are highly correlated and present spectral similarities. To benefit from spatial correlation, we propose a second method which consists to reconstruct by patch rather than by single pixel, and then apply an opportune fusion to infer the estimation of each missed image pixel.

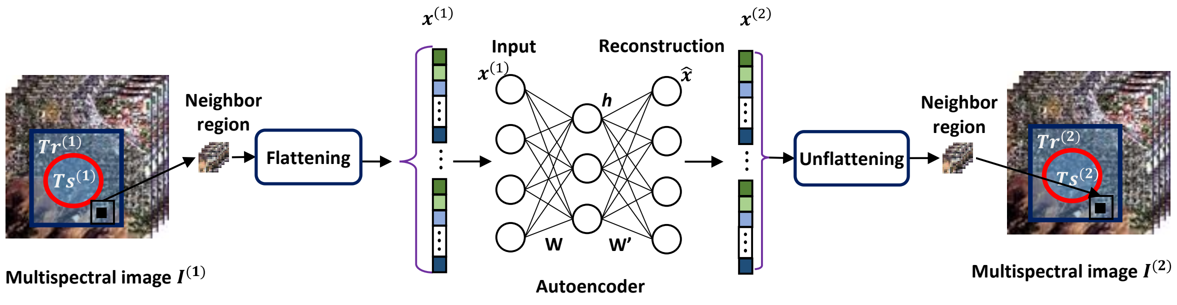


Figure 3-2: Proposed patch-based AutoEncoding reconstruction method.

In this method, we keep the same principle like the previous method, but the size of the input layer (same for the reconstruction layer) is increased. As illustrated in Figure 3-2, the input vector $\boldsymbol{x}^{(1)}$ (similar reasoning for the target vector $\boldsymbol{x}^{(2)}$) is formed by flattering the patch which is composed by the current pixel of interest (the central pixel) and its spatial neighboring pixels. If a window of size $sz \times sz$ is chosen, then the size of the input layer n is equal to $sz \times sz \times nb$. Therefore, rather than mapping just one pixel at a time, a region of size of $sz \times sz$ is mapped.

With such a reconstruction scheme, the problem which arises is that instead of getting an estimation for each target pixel, we obtain ($sz \times sz$) estimations (excluding the cases of pixels lying close to the image borders). This is due to the fact that we pass from a pixel-based to a patch-based reconstruction. In order to get an estimation for each pixel from the generated patches, we propose to apply a simple weighted average of all the ($sz \times sz$) cases as follows:

$$\boldsymbol{x}_{Ts^{(2)}} = \sum_{i=1}^{sz\times sz} \alpha_i \hat{\boldsymbol{x}}_i \tag{3.8}$$

where α_i are weights such that:

$$\sum_{i=1}^{sz\times sz} \alpha_i = 1 \quad \text{and} \quad 0 \le \alpha_i \le 1 \tag{3.9}$$

and $\hat{\boldsymbol{x}}_i$ is the mapping of the pixel $\boldsymbol{x}^{(1)}$ obtained from the ith patch containing the pixel of interest.

Regarding the selection of the weight values, we will opt for a simple strategy in which a weight of 0.5 is assigned to the central pixel while all remaining neighboring pixels will have a weight equal to $\frac{0.5}{(sz \times sz - 1)}$ (which means that 50% of the weighting is assigned to the central pixel and 50% is equally distributed over all remaining pixels of the patch). It is noteworthy that other strategies (i.e., equal weights and Gaussian function weights) were implemented as well but performed worse or did not provide any significant improvement compared to the above strategy.

Figure 3-3 shows an example of a 3 × 3 neighborhood. Case 1 is the case when the pixel of interest coincides with the center of the window. The other cases refer to situations when the pixel is still inside the window (but not at its center).

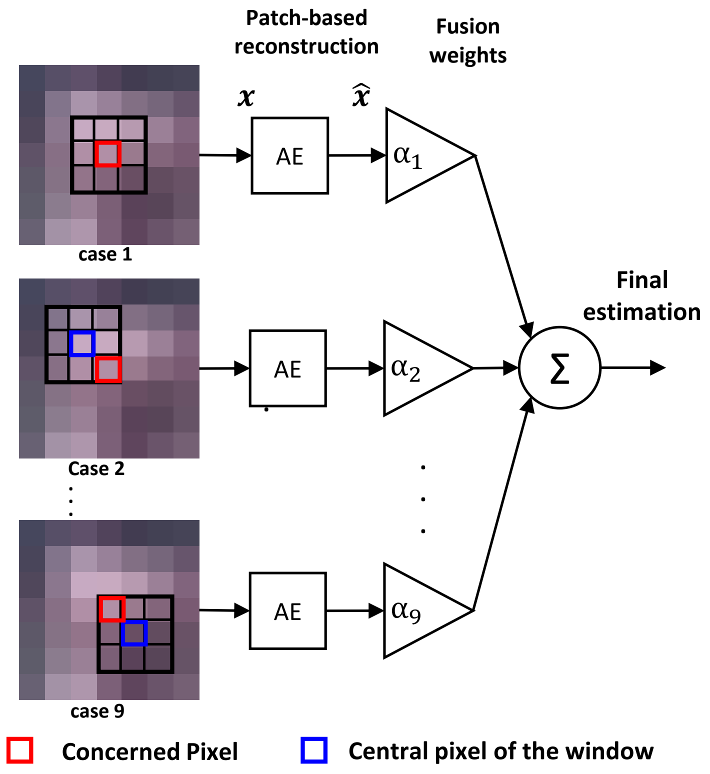


Figure 3-3: Illustration of the fusion of the patch-based results related to a given pixel of interest (in black) for a 3×3 neighborhood system. In this case, $\alpha_1 = \frac{1}{2}$ and all other weights $\alpha_2 = \ldots = \alpha_9 = \frac{1}{16}$.

3.2.3 *Estimation of the size of the hidden layer*

As the size of the hidden layer *d* is unknown a priori, it needs to be estimated. Typically, this is done empirically by trying various configurations and picking up the one with highest accuracy. In this proposed method, for solving this issue, we propose a new method which is inspired from the minimum description length (MDL) principle [21]. If we take for instance the problem of the selection of the number of components in a mixture, MDL permits to search for a tradeoff since, on the one hand the higher the number of components, the higher the risk of overfitting, while on the other, the smaller the number of components, the lower the model flexibility [8]. In such a case (mixture density problem), MDL is defined as [19]:

$$\mathrm{MDL}(d) = -\tilde{\ell}(\psi|\Theta) + \gamma . k . \log(L) \tag{3.10}$$

where $\tilde{\ell}(\psi|\Theta)$ represents the log-likelihood function value found with a maximum likelihood estimation algorithm, k is the number of free parameters to be estimated (in our case it will be equal to the number of weights and biases), γ is a constant and L is the number of data sample. Therefore, the MDL principle aims at achieving a balance between accuracy and complexity of the model so that to provide a good generalization capability. Applying a simple grid-search procedure guided by empirical risk minimization can instead lead to overfitting issues.

In order to adapt MDL to our problem, $\psi = \{\varepsilon_1, \varepsilon_2, \dots, \varepsilon_L\}$ will represent the set of multidimensional errors incurred by the AE (dimensionality = number of output nodes) for each of the L input data. For the sake of tractability, we assume the multidimensional errors are drawn from a multivariate normal distribution, namely $p(\varepsilon) = N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ where the mean vector $\boldsymbol{\mu}$ is supposed to be null and the covariance matrix $\boldsymbol{\Sigma}$ is supposed to be a diagonal matrix with values equal to σ^2 where σ is the standard deviation. Then, the distribution function can be written as follows:

$$p(\varepsilon) = \frac{1}{\sqrt{(2\pi)^n|\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}\varepsilon^{\mathrm{T}}\boldsymbol{\Sigma}^{-\mathbf{1}}\varepsilon\right) \tag{3.11}$$

The negative log-likelihood function will be represented as follows:

$$\begin{aligned} -\tilde{\ell}(\psi|\Theta) &= -log \prod_{i=1}^{L} p(\varepsilon_i) = -\sum_{i=1}^{L} \log(p(\varepsilon_i)) \\ &= \frac{1}{2}\sum_{i=1}^{L}((2\pi)^n|\boldsymbol{\Sigma}|) + \frac{1}{2}\sum_{i=1}^{L}\left(\varepsilon_i^{\mathrm{T}}\boldsymbol{\Sigma}^{-\mathbf{1}}\varepsilon_i\right) \end{aligned} \tag{3.12}$$

The optimal size of hidden layer $\hat{d}$ is estimated by minimizing the MDL criterion, i.e.,

$$\hat{d} = \underset{d\,=\,1..d_{max}}{\arg\min}\,\{\mathrm{MDL}(d)\} \tag{3.13}$$

where d_{max} is a predefined maximum size of the hidden layer.

During simulations, we noticed that the second term of the MDL criterion, which depends on the number of weights and biases involved in the considered AE architecture, increases rapidly compared to the first term (the log-likelihood function) which decreases very slowly. The observed best $\hat{d}$ value corresponds always to a hidden layer size of one or two units (i.e., $d = 1$ or $d = 2$). In order to get a better balance between the two terms of the MDL criterion, the parameter γ need to be as smaller as possible to keep under control the rapid increase of the second part. However, there is no guarantee that the best choice of γ for one dataset will give also good results for other datasets and in this case the developed method will be highly dependent on the value of the parameter γ. To overcome this issue, we propose to opt for a Pareto-like optimization, inspired from the multi-objective optimization literature (MO optimization) and the nondominated sorting concept.

3.2.4 Multi-objective Optimization

In the cases when there are multiple measures of competing objectives (criteria) to be simultaneously estimated like in ours, MO optimization can be solved by combining linearly the objectives into a single function (like previous MDL formulation) with opportune weights or by finding a set of optimal solutions rather than a single one. The selection of a solution from this set is not trivial and is usually user-dependent. From a mathematical viewpoint, a general MO optimization problem can be formulated as follows:

Find the vector $\boldsymbol{p}^*$ which minimizes the ensemble of Q objective functions:

$$\boldsymbol{f}(\boldsymbol{p}) = [f_i(\boldsymbol{p}), i = 1, \dots, Q] \tag{3.14}$$

subject to the J equality constraints

$$g_{\boldsymbol{j}}(\boldsymbol{p}) = 0 \qquad j=1,2,\dots,J \tag{3.15}$$

and the K inequality constraints

$$h_k(\boldsymbol{p}) \leq 0 \qquad k=1,2,\dots,K \tag{3.16}$$

where $\boldsymbol{p}$ is a candidate solution to the considered optimization problem. In our case, it consists of finding the solution that minimize the two criteria $f_1 = -\tilde{\ell}(\psi|\Theta)$ and $f_2 = k.\log(L)$ (Q = 2) without any constraints.

The solving of a MO optimization problem is based on the concept of dominance. A solution $\boldsymbol{p}_\mathrm{i}$ is said to dominate another solution $\boldsymbol{p}_\mathrm{j}$ if and only if $\boldsymbol{f}(\boldsymbol{p}_\mathrm{i})$ is partially less than $\boldsymbol{f}(\boldsymbol{p}_\mathrm{j})$, i.e.,

$$\forall\, k \in \{1,2, \dots, Q\}, f_k(\boldsymbol{p}_\mathrm{i}) \leq f_k(\boldsymbol{p}_\mathrm{j}) \;\; \wedge \; \exists k \in \{1,2, \dots, Q\}: f_k(\boldsymbol{p}_\mathrm{i}) < f_k(\boldsymbol{p}_\mathrm{j}) \tag{3.17}$$

This concept leads to the definition of *Pareto optimality*: a solution $\boldsymbol{p}_i^* \in \Omega$ (Ω is the solution space) is said to be *Pareto optimal* if and only if there exists no other solution $\boldsymbol{p}_j^* \in \Omega$ that dominates $\boldsymbol{p}_i^*$. The latter is said to be *nondominated* and the set of all nondominated solutions forms the so-called *Pareto front* of optimal solutions.

Once the Pareto front is identified, a solution has to be selected from the set of nondominated solutions. Although different strategies can be found in the literature, in our method we used the simple median solution to maintain a tradeoff between the two different criteria.

An example of nondominated sorting is shown in Figure 3-4, in which a joint optimization of the two criteria f_1 and f_2 is involved. The nondominated samples (in red) constitute the Pareto front, which represents the set of optimal solutions. From this set, the selected solution is given by the median one (in green). Dominated solutions are drawn with black crosses.

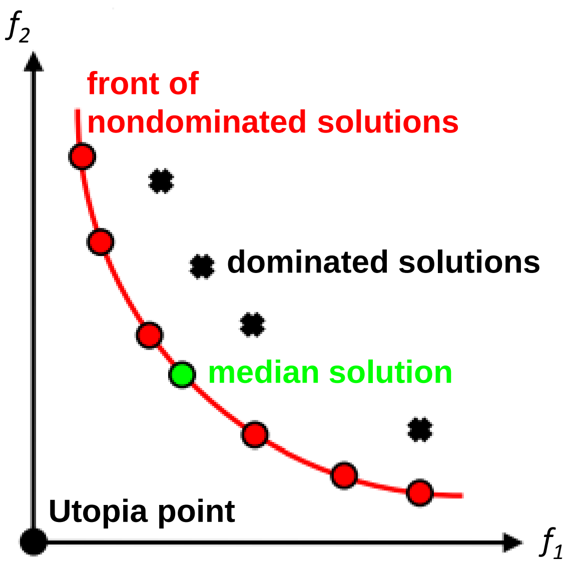


Figure 3-4: Illustration of a front of nondominated solutions.

3.3. Experimental Validation

3.3.1 Dataset description

In our simulations, we use two different datasets, each containing two images. We assume that one of the two images contains the cloudy regions. The first dataset was acquired by the Taiwanese optical high resolution FORMOSAT-2 satellite [22], which permits the acquisition of an area of interest every day, from the same viewpoint and under the same light conditions. These images represent part of the *Arcachon* basin in the south region of Aquitaine, in France. The images are composed of 400×400 pixels and four spectral bands (blue, green, red, and near infrared) with a pixel spacing of 8 m. They were acquired on the 24th of June and 16th of July, 2009, respectively (see Figure 3-5). The second dataset was acquired by the French satellite SPOT-5, whose images represent part of the Réunion island [23]. The images are characterized by a size of 450×450 pixels, four spectral bands (blue, green, red, and near infrared), and a pixel spacing of 10 m and were taken on May 2 and June 18, 2008, respectively (see Figure 3-6). The two datasets exhibit several disparities, which are (i) sensor-wise, and/or (ii) inter-pixel spacing, and importantly (iii) the land covers. In fact, the former dataset displays more vegetation than urban areas, whereas the latter shows otherwise.

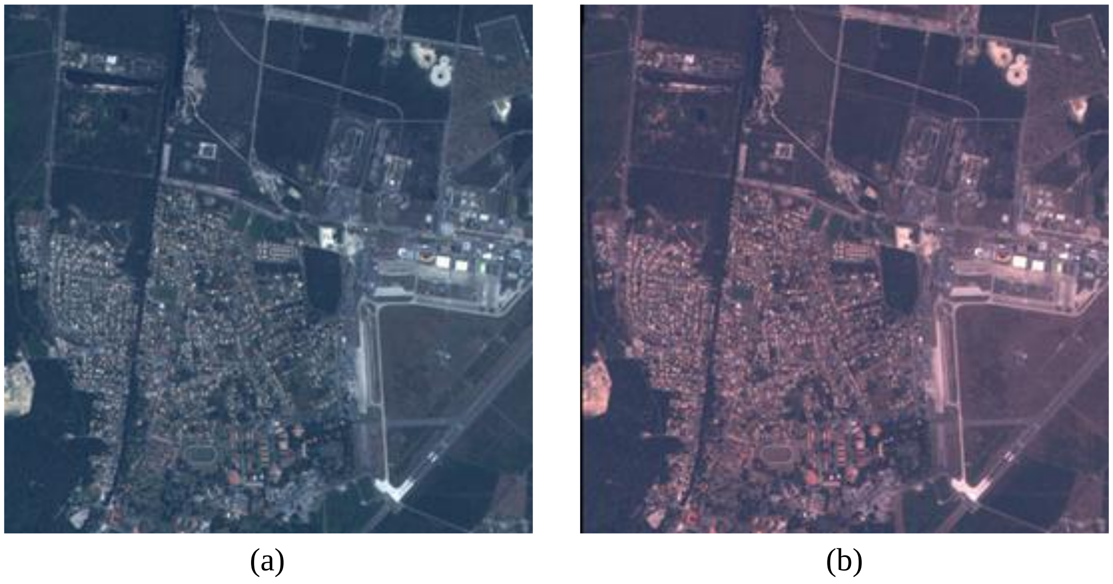


Figure 3-5: Color composite of first dataset acquired by FORMOSAT-2 over the *Arcachon* basin on (a) 24th June and (b) 16th June, 2009.

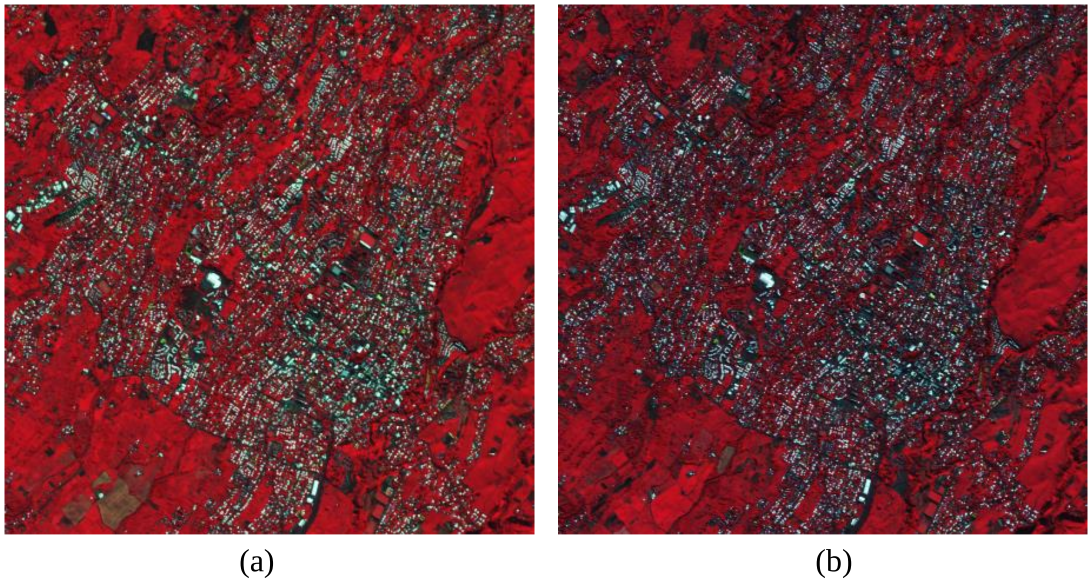


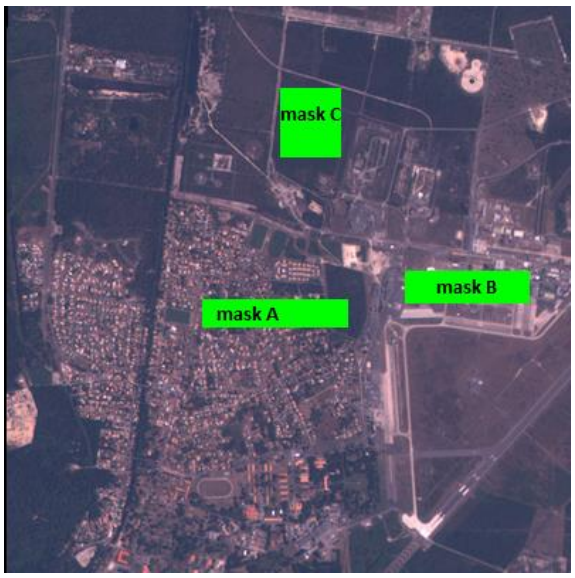
Figure 3-6: Color composite of second dataset acquired by SPOT-5 over the *Réunion* island on (a) May 2nd and (b) June 18th, 2008.

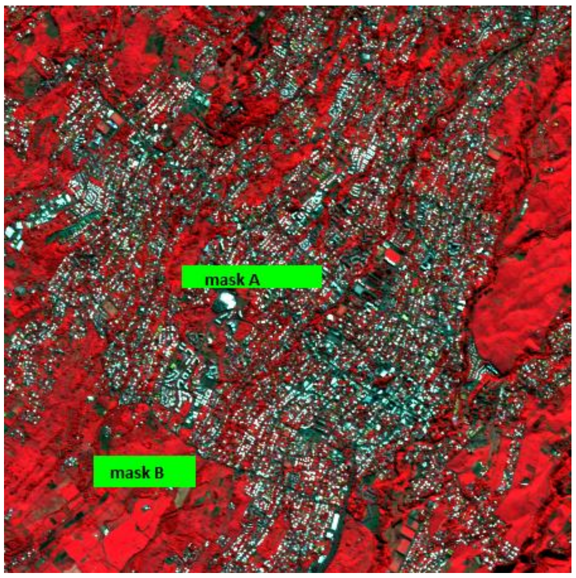
In order to evaluate the two developed methods, the rocedure adopted in our experiments consists: 1) to consider a cloud-free image, e.g., $I^{(1)}$; 2) to simulate the presence of clouds by partly obscuring the other image, e.g., $I^{(2)}$; and 3) to compare the reconstructed image with the original cloud-free image. This study aims at understanding the sensitivity of the two investigated methods regarding two aspects, which are: 1) the kind of ground covers obscured; and 2) the size of the contaminated area. In order to obtain a detailed assessment of the reconstruction quality, we adopt the popular peak signal-to-noise ratio (PSNR) measure [24], as well as the correlation coefficient.

3.3.2 Results

In the experiments, the size of the hidden layer d in the autoencoder is varied over a predefined range and the two objective functions f_1 and f_2 are calculated at convergence for each case. In the first method, the size n of the input feature vector is equal to four (number of image bands). For the second method, a neighborhood of 3×3 is chosen, the value of n in this case is equal to 4×3×3. The value of d for both cases is chosen to be within the range [1, 100]. The other parameters of the autoencoder are fixed as follows: momentum = 0.5, initial learning rate = 1, iterative decay factor for learning rate = 0.95, and number of epochs = 150. Regarding the multivariate normal distribution $N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, we used zero mean and $\sigma = 0.04$.

1) Contamination of Different Ground Covers: Figure 3-7 shows different masks whose positions were selected in such a way as to simulate the obscuration of different kinds of ground cover. In particular, for the first dataset, in Figure 3-7(a), mask A is over a completely urban area, mask B covers a region that includes mainly industrial zone, and mask C obscures a vegetation area. For the second dataset, Figure 3-7(b) shows mask A covering mainly a rural area, and mask B a vegetation region. The experiments were carried out by considering each mask at a time, where each mask is composed by around 2000 pixels, and the training set Tr is composed by around 4000 pixels from the surrounding region of each mask.



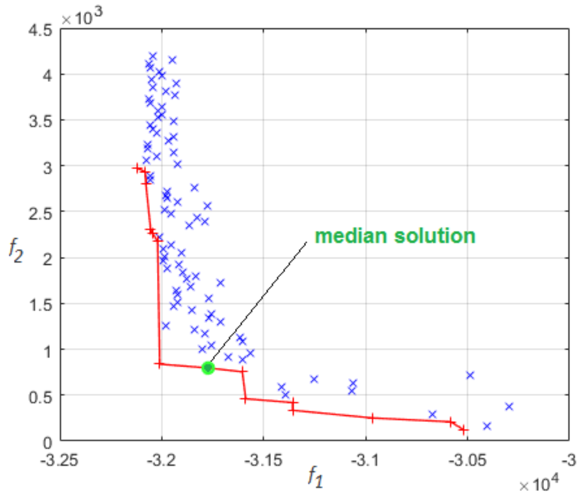


(a) (b)

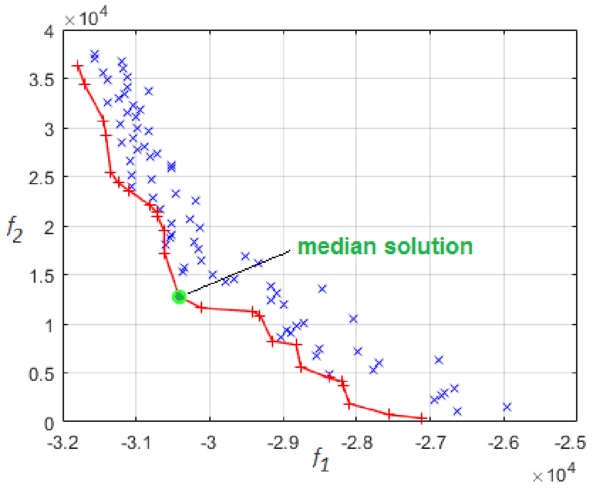
Figure 3-7: Masks adopted to simulate the different ground cover contaminations.

In the experiments, mask A is used for training an autoencoder AE_A, which on its turn is considered as pretrained autoencoder exploited to fine-tune the other models, i.e., AE_B for mask B and AE_C for mask C. In these cases (masks B and C), the number of epochs is reduced to half.

Figure 3-8 shows an example of the Pareto fronts obtained at convergence for the first dataset using mask A and for both developed methods of reconstruction where the nondominated solutions lie along a red curve and the selected solution is highlighted with a green circle. The different best solutions for both datasets are reported in Table 3.1.



(a)



(b)

Figure 3-8: Pareto fronts obtained at convergence for the first dataset and mask A simulation by (a) pixel-based autoencoding reconstruction, and (b) patch-based autoencoding reconstruction.

Table 3.1: Best size of hidden layer found for the different cases.

	Dataset 1		**Dataset 2**	
Method	**Pixel-AE**	**Patch-AE**	**Pixel-AE**	**Patch-AE**
Best *d* value	19	29	14	29

In order to evaluate our methods, we compare the obtained results in our experiments with results found by state-of-the-art methods based on compressive sensing theory, namely the Orthogonal Matching Pursuit (OMP), Basis Pursuit (BP) and Genetic Algorithm (GA) reconstruction techniques [11]. Compressed sensing in signal processing is considered as an efficient approach for reconstructing a signal by finding solutions of an underdetermined linear system under constraint of sparsity. BP convexifies the problem by solving it under L1 norm instead of L0 norm [25], [26]. OMP is a faster alternative of the MP method and is based on finding the atom that has the highest correlation with the signal and then subtracts off the correlated part from the signal and iterates the procedure on the resulting residual signal [27], [28]. Regarding GAs, they are considered as part of evolutionary computation methods which solves optimization problems by performing a search by regenerating a population of candidate solutions represented by chromosomes [29], [30]. The Non-Dominated Sorting Genetic Algorithm-II (NSGA-II) [31] is adopted in order to find the optimal solution.

The results are reported in Table 3.2, from which we can see that our methods perform better with exception of the case of mask B on dataset 2 where BP outperforms our method by 0.24 dB but provides a similar correlation coefficient (0.91). Regarding the other cases, the improvement is between 3.69 dB (mask B on dataset 1) and 11.49 dB for the case of mask C on dataset 1. Such improvements can be justified by the fact that methods based on compressive sensing approach are based on a linear reconstruction paradigm contrary to the AE which involves a nonlinear transformation. Moreover, in the method developed in [11], a dictionary is created by grid-sampling over all the image and with a limited number of atoms in order to reduce the processing complexity. By contrast in our methods, we select the training patterns from pixels of the surrounding region of the masks, which are potentially more correlated to those obscured. Finally, it can be seen that using patch-based reconstruction improves significantly the result compared to using pixel-based information for reconstruction, since it opportunely exploits the spatial correlation between neighboring pixels.

Table 3.2: (a) PSNR values and (b) correlation coefficients obtained by the different methods in the first simulation experiments.

(a)

Method	Dataset 1			Dataset 2	
	Mask A	Mask B	Mask C	Mask A	Mask B
OMP [11]	23.96	20.60	31.97	26.36	30.43
BP [11]	22.22	24.74	30.67	26.45	**31.63**
GA [11]	23.78	23.15	32.01	26.72	31.28
Pixel-AE	29.71	27.73	39.42	28.62	31.25
Patch-AE	**32.95**	**28.47**	**43.94**	**30.77**	**32.23**

(b)

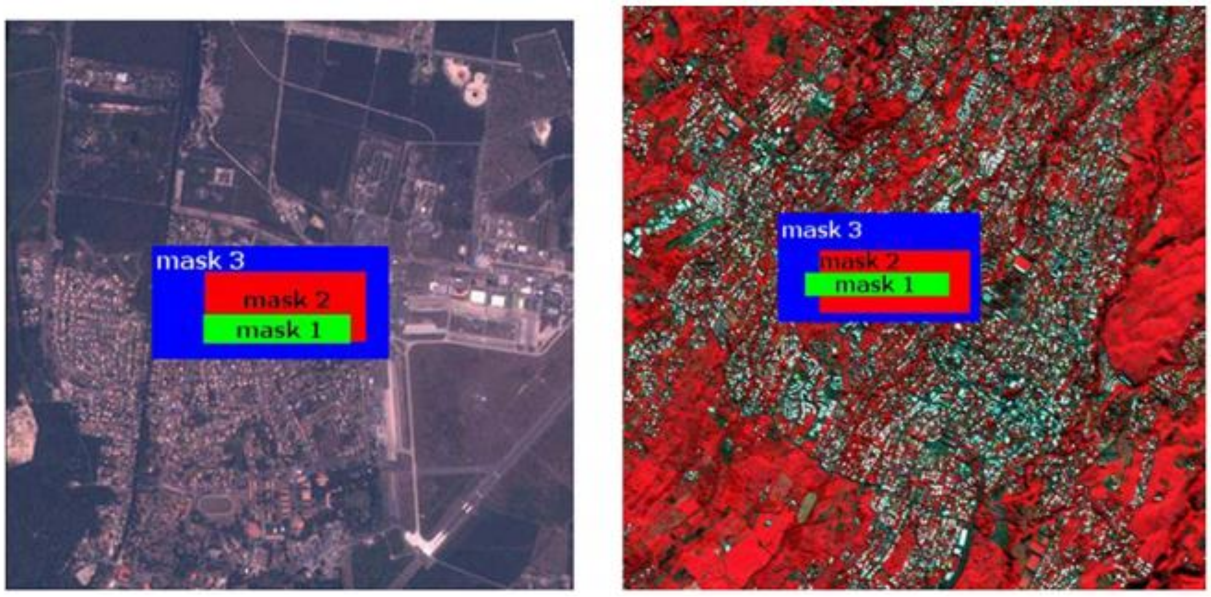
Method	Dataset 1			Dataset 2	
	Mask A	Mask B	Mask C	Mask A	Mask B
OMP [11]	0.81	0.94	0.89	0.77	0.88
BP [11]	0.86	0.96	0.90	0.76	0.91
GA [11]	0.84	0.95	0.90	0.78	0.90
Pixel-AE	0.93	**0.98**	0.98	0.90	0.91
Patch-AE	**0.97**	**0.98**	**0.99**	**0.91**	**0.93**

Finally, we analyzed the sensitivity of the patch-based strategy to the size of the patch, by increasing it from 3×3 to 7×7. The results which are provided in Table 3.3 suggest that the accuracy decreases as the size increases. This can be explained by the fact that an increasing size of the patch involves a quadratic increase in the dimensionality of both the input and output spaces and thus a potential risk of curse of dimensionality. Moreover, increasing the size leads to a decrease of correlation between the central pixel and the neighboring ones adding noise in the reconstruction process.

Table 3.3: Analysis of the sensitivity to the patch size in terms of PSNR for the first simulation experiments. 1×1 size refers to the pixel-based strategy.

Patch size	Dataset 1			Dataset 2	
	Mask A	Mask B	Mask C	Mask A	Mask B
1x1	29.71	27.23	39.71	28.62	31.25
3x3	**32.95**	**28.47**	**43.94**	**30.77**	**32.23**
5x5	30.74	25.57	43.53	29.86	31.1
7x7	23.61	16.83	32.12	27.93	31.25

2) Contamination with Different Sizes: the second simulation experiments consist of increasing the size of the obscured area. Figure 3-9 illustrates the three different masks adopted to simulate the different sizes of the clouds. Mask 1 is fixed with the same size as the masks A adopted in the previous experiments, i.e., it covers about 2000 pixels. Masks 2 and 3 are built by multiplying the previous size, by 3 and by 6, and the resulting masks cover around 6000 and 12000 pixels, respectively. Also in these experiments, we selected pixels for training from the surrounding regions of the obscured areas. The size of the training is chosen to be around double the size of the corresponding mask. Similarly to previous experiment, mask 1 is used to build the pretrained autoencoder AE_1. A fine tuning is applied by using AE_1 on the two other cases (mask 2 and mask 3) in order to build their corresponding models AE_2 and AE_3, respectively. Table 3.4 reports for the two datasets the results achieved by the two reconstruction techniques and by varying the amount of missing data.



(a) (b)

Figure 3-9: Masks adopted to simulate the different sizes of contamination.

From a quantitative viewpoint, in terms of PSNR and correlation coefficient, we have similar results as in the previous experiments. The two developed methods outperform by more than 3 dB for all cases compared to the other methods. Also, we can notice that by increasing the size of the contaminated region, the PSNR increases very slightly which shows that the developed strategy maintains the reconstruction quality of the cloudy regions almost independently from the size of the clouds. The correlation coefficients are much higher in most of the cases (around 0.95 on an average).

Table 3.4: (a) PSNR values and (b) correlation coefficients obtained by the different methods in the second simulation experiments.

(a)

Method	**Dataset 1**			**Dataset 2**		
	Mask 1	**Mask 2**	**Mask 3**	**Mask 1**	**Mask 2**	**Mask 3**
OMP [11]	23.96	23.21	25.01	26.36	26.42	27.39
BP [11]	22.22	22.89	21.47	26.45	26.82	28.25
GA [11]	23.78	23.85	23.03	26.72	27.10	28.15
Pixel-AE	29.71	30.05	29.68	28.92	28.53	29.18
Patch-AE	**32.95**	**33.15**	**32.37**	**30.77**	**30.85**	**30.91**

(b)

Method	**Dataset 1**			**Dataset 2**		
	Mask 1	**Mask 2**	**Mask 3**	**Mask 1**	**Mask 2**	**Mask 3**
OMP [11]	0.81	0.75	0.92	0.77	0.78	0.80
BP [11]	0.86	0.87	0.87	0.76	0.77	0.81
GA [11]	0.84	0.82	0.89	0.78	0.78	0.80
Pixel-AE	0.93	0.95	0.95	0.90	0.84	0.84
Patch-AE	**0.97**	**0.98**	**0.98**	**0.91**	**0.92**	**0.90**

From a qualitative viewpoint, Figure 3-10 and Figure 3-11 show the reconstruction results in color composites obtained for dataset 1 (with mask C) and for dataset 2 (with mask 2), respectively. In particular, OMP and Patch-AE reconstruction methods are considered for comparison since: 1) OMP represents the best compromise between accuracy and computation time compared to BP and GA strategies; and 2) Patch-AE outperforms Pixel-AE in all simulations with insignificant extra computation cost. From Figure 3-10.b and Figure 3-10.c, it can be observed that OMP generates a salt-and-pepper noise in the reconstruction while the Patch-AE result appears perfect visually (which confirms the 43.60 dB of reconstruction accuracy, see Table 3.2). In Figure 3-11.b and Figure 3-11.c, due to the complex structure of the urban area, the reconstruction task is a priori harder. The OMP result exhibits a noise under the form of dark spots spread along the urban structures, it loses some urban morphologies and introduces a bias which makes the reconstruction appear a bit brighter than it should. All these problems can be explained by the fact that the OMP is a very sparse strategy as mentioned in [11], which makes it less robust in particular when complex structures are contaminated. The Patch-AE result is definitely better as most of the structures appear correctly reconstructed but with a problem of blurring. This can be explained by the fact that the Patch-AE method reconstructs the single pixels by averaging the outcomes from all patches involving each single pixel (see Figure 3-3). Such a weighted averaging operation acts thus as a post-processing filter incurring in a blurring effect.

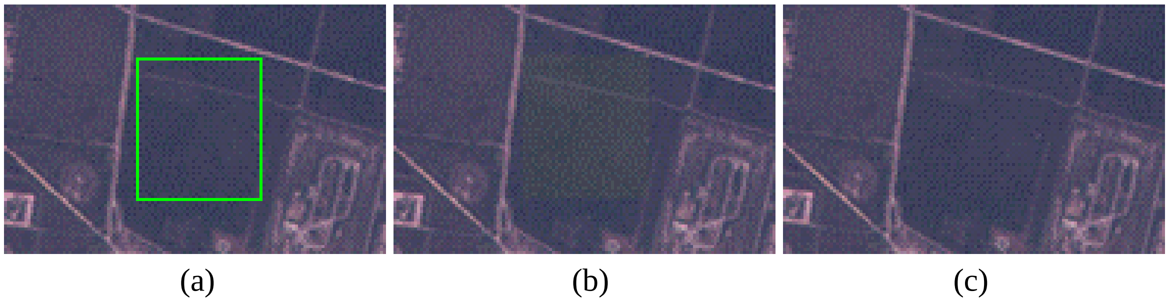


Figure 3-10: Examples of qualitative results for Dataset 1. (a) Original image. Image reconstructed (after contamination with mask C) by the (b) OMP and (c) patch-based reconstruction methods.

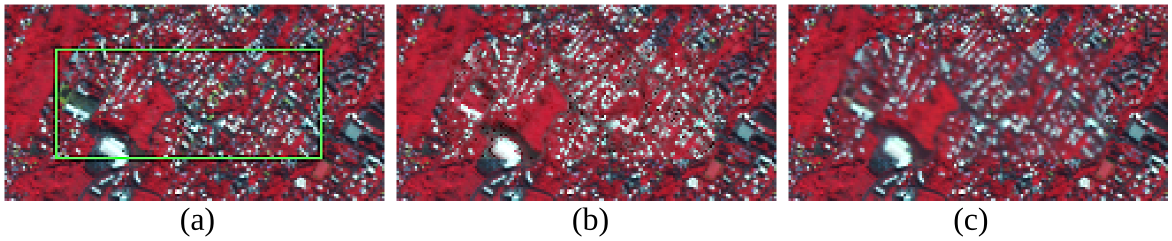


Figure 3-11: Examples of qualitative results for Dataset 2. (a) Original image. Image reconstructed (after contamination with mask 2) by the (b) OMP and (c) patch-based reconstruction methods.

As performed for the first set of experiments, we analyzed again the sensitivity of the patch-based strategy to the patch size (see Table 3.5). Similar observations can be drawn, leading to the conclusion that a 3×3 patch size is the best compromise in terms of accuracy and computation load.

Table 3.5: Analysis of the sensitivity to the patch size in terms of PSNR for the second simulation experiments. 1×1 size refers to the pixel-based strategy.

Patch size	**Dataset 1**			**Dataset 2**		
	Mask 1	**Mask 2**	**Mask 3**	**Mask 1**	**Mask 2**	**Mask 3**
1x1	29.71	30.05	29.68	28.62	28.53	29.18
3x3	**32.95**	**33.15**	**32.37**	**30.77**	**30.85**	**30.91**
5x5	30.74	31.26	30.99	29.86	29.9	30.22
7x7	23.61	23.78	23.16	27.93	27.91	28.68

3.3.3 Results on real clouds

In addition, we applied our method on a data set with real clouds (see Figure 3-12). In particular, this data set, useful for qualitative evaluation, is composed of three images. It was acquired by the European optical high resolution Sentinel-2 satellite. The images represent part of the *Washington* region in USA. They contain 800×700 pixels and four spectral bands (blue, green, red, and near infrared) with a pixel spacing of 10 m on the ground. They were acquired on September 14th 2015, August 5th 2015, and July 20th, 2016, respectively. The first image is used as reference image (cloud free image) and the two others as target images (cloudy images). Between the first and second images (Figure 3-12.a and

Figure 3-12.b), a first autoencoder AE_1 was trained and exploited as a pretrained model to fine-tune another autoencoder AE_2 for the third image (Figure 3-12.c). The clouds and their shadows on both cloudy images were masked manually (Figure 3-13). The obtained results on the two target images are provided in Figure 3-14. Figure 3-15 presents zooms of contaminated regions after reconstruction. The images show the good capability of the developed method in reconstructing areas contaminated by clouds (and their shadow). Indeed, the differences between the reconstructed areas and the surrounding regions are very small and can hardly be seen visually. On its side, the reference OMP method exhibits a slight spectral mismatch when compared to the surrounding uncontaminated area (see Figure 3-15.c and Figure 3-15.f) as well as some artifacts (see Figure 3-15.c).

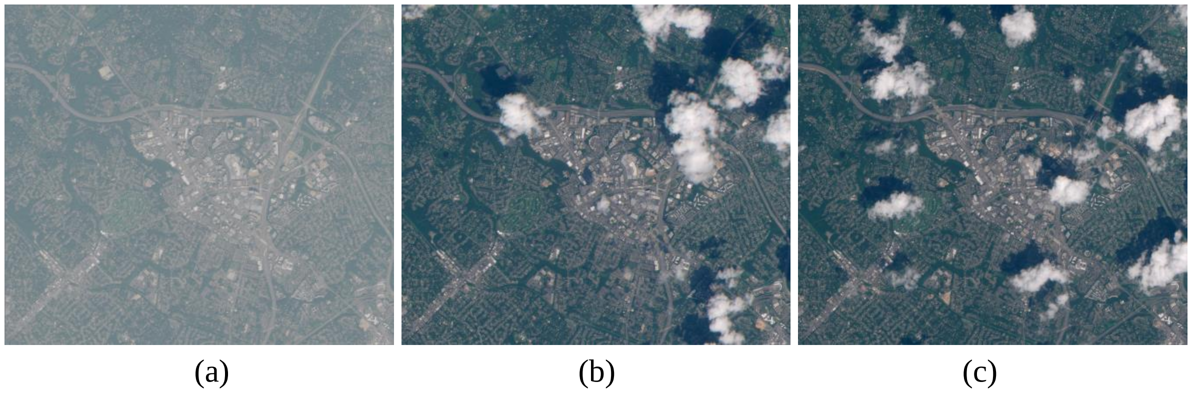


Figure 3-12: Color composite of the third dataset acquired by Sentinel-2 over Washington on (a) September 14th, 2015 (source image); (b) August 5th, 2015 (target image 1); and (c) July 20th, 2016 (target image 2).

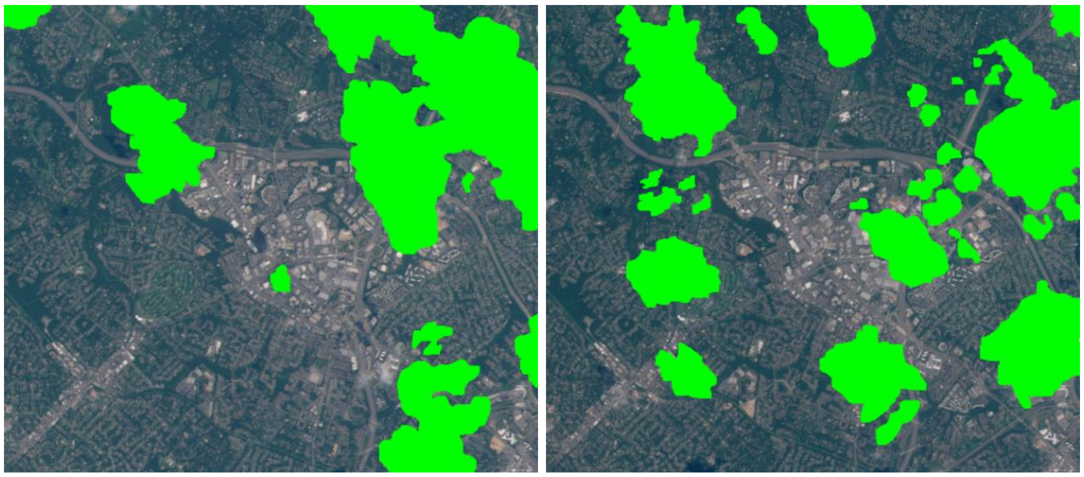
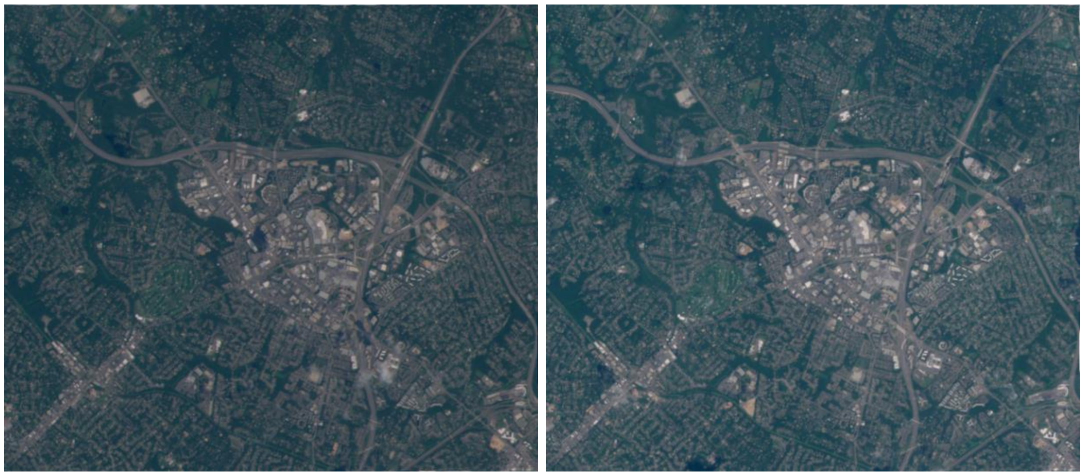
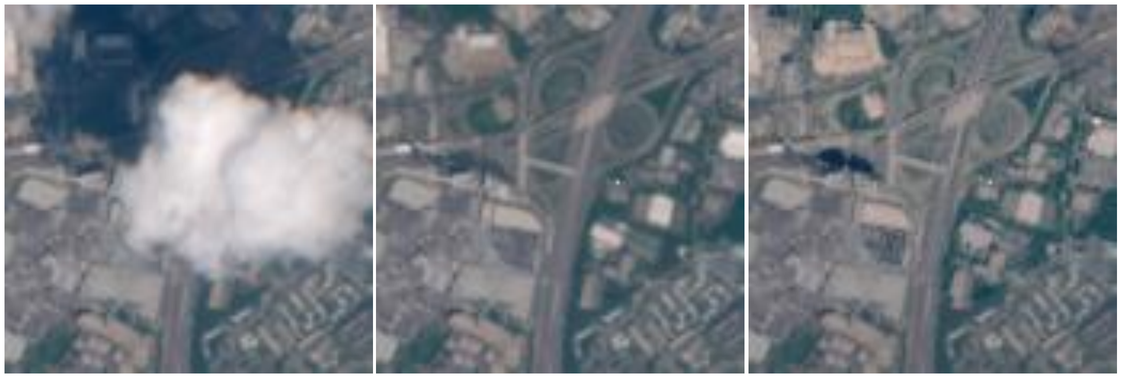


Figure 3-13: Masked clouds and shadows of the third dataset.



(a) (b)

Figure 3-14: Reconstructed images obtained for dataset 3. (a) second image (August 5th, 2015), (b) third image (July 20th, 2016).



(a) (b) (c)

Figure 3-15: Zooms of reconstruction results obtained for third image (July 20th, 2016) from source image (September 14th, 2015) of dataset 3, over (a)-(b) urban and (d)-(e) green areas. For comparison, results generated by the OMP method are provided in (c) and (f).

As a last experiment, we tested our method on an image characterized by the presence of phenological changes. This image was acquired in the region of Bejaia, Algeria, with the Landsat-8 satellite. The first (source) image was acquired on July 31st, 2016 (Figure 3-16.a), while the second (target) image was taken on March 28th, 2016 (Figure 3-16.b). The latter conveys healthy vegetation while the former reports dry vegetation due to high temperatures at that period of the year. Figure 3-16.c reveals that, despite the presence of significant spectral changes in the land covers, the proposed method is capable to capture them and provide a visually sound reconstruction result.

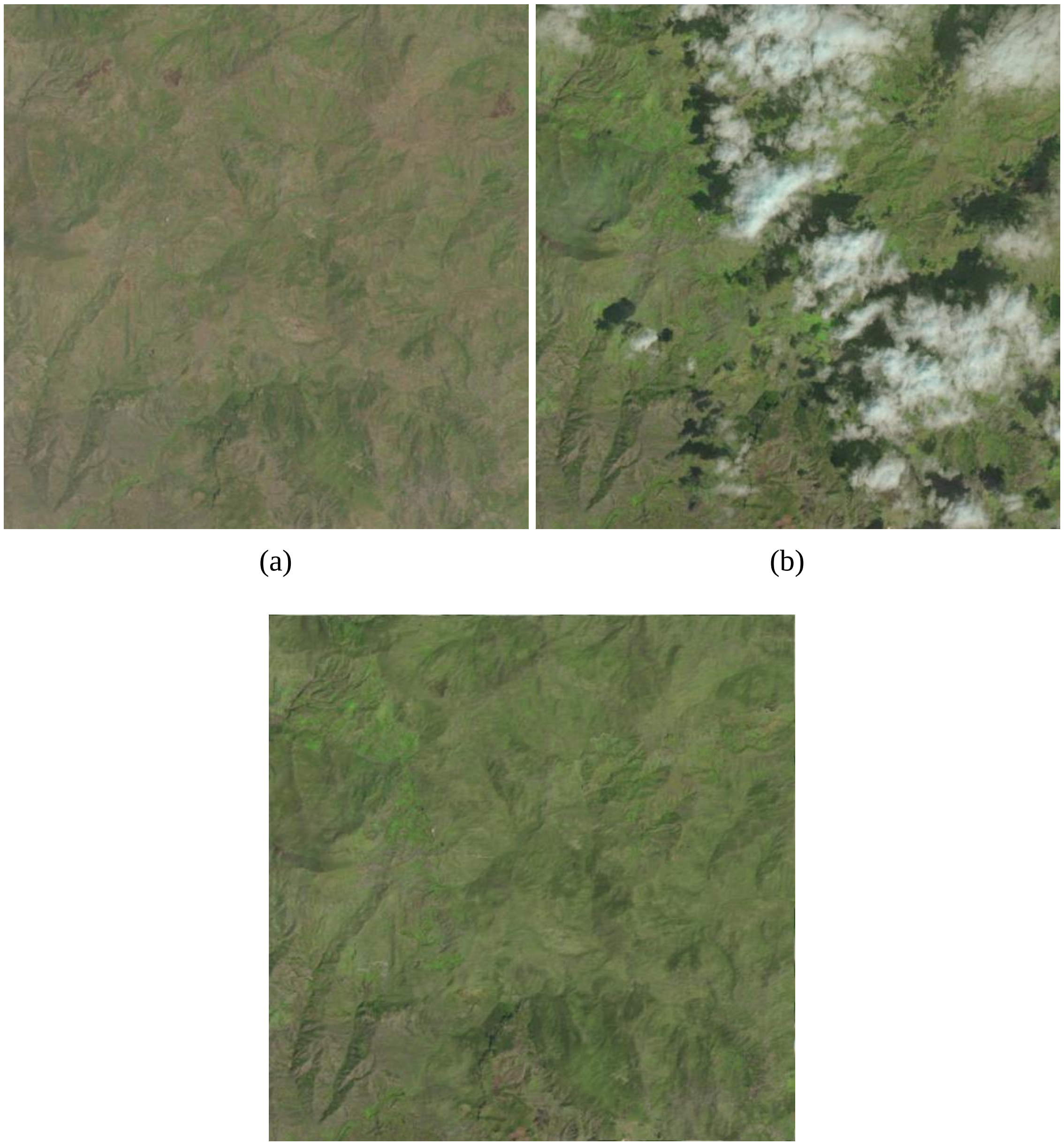


Figure 3-16: Results achieved for dataset 4. (a) first image (July 31st, 2016), (b) second image (March 28th, 2016), (c) reconstruction of second image.

3.4. Conclusion

We have proposed in this chapter a new approach to recover missing data in multispectral images due to the presence of clouds. In particular, the reconstruction problem is formulated under an autoencoding perspective, based on an AE neural network. Given a cloud-free image (source image) and a cloud-contaminated image (target image), the standard AE process is slightly modified so as to estimate the mapping function between the source and the target images. For this purpose, we have developed two strategies; the first relies on simple pixel-based information to calculate the transformation function whereas the second strategy performs a patch-to-patch mapping followed by a simple fusion step to reconstruct the single pixels of the missing areas in the target image. Moreover, in order to fix the problem of the hidden layer size, a new solution combining the minimum descriptive length (MDL) criterion and a Pareto-like selection method has been introduced.

The experimental results reveal that the two proposed methods (Pixel-AE and Patch-AE) show good results in reconstructing the missing areas and can significantly outperform state-of-the-arts methods. Compared to Pixel-AE, Patch-AE yields better accuracies thanks to the feeding of contextual, and thus richer information, in the reconstruction model. In general, it is noteworthy that the size of the contaminated region does not affect the performance of the methods. Nevertheless, since the training is performed over the neighborhood of the missing area it is important that the neighborhood be enough representative. Otherwise, the risk to get unsatisfactory results can sharply increase.

3.5. References

[1] C. Ji, "Haze reduction from the visible bands of LANDSAT TM and ETM+ images over a shallow water reef environment," *Remote Sens. Environ.*, vol. 112, no. 4, pp. 1773–1783, Apr. 2008.

[2] M. Xu, X. Jia, and M. Pickering, "Automatic cloud removal for Landsat 8 OLI images using cirrus band," in *Proc. IEEE IGARSS*, 2014, pp. 2511–2514.

[3] M. Xu, X. Jia M. Pickering, and A. J. Plaza, "Cloud Removal Based on Sparse Representation via Multitemporal Dictionary Learning," *IEEE Trans. Geosci. Remote Sens.*, Vol. 54, No. 5, pp. 2998–3006, Mars 2016.

[4] M. Xu, M. Pickering, A. J. Plaza, and X. Jia, "Thin cloud removal based on signal transmission principles and spectral mixture analysis," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 3, pp. 1659–1669, 2016.

[5] A. Maalouf, P. Carré, B. Augereau, and C.F. Maloigne, "A Bandelet-Based Inpainting Technique for Clouds Removal From Remotely Sensed Images," *IEEE Trans. Geosci. Remote Sens.*, vol. 47, no. 7, pp. 2363–2371, Jul. 2009.

[6] L. Lorenzi, F. Melgani, and G. Mercier, "A support vector regression with kernel combination for missing data reconstruction," *IEEE Geosci. Remote Sens. Lett.* S, vol. 10, no. 2, pp. 367-371, Mar. 2013.

[7] S. C. Liew, M. Li, and L. K. Kwoh, "Automated production of cloud-free and cloud-shadow image mosaics from cloudy satellite imagery," in *Proc. 20th ISPRS Congr.*, Istanbul, Turkey, Jul. 2004, pp. 523–530.

[8] F. Melgani, "Contextual reconstruction of cloud-contaminated multitemporal multispectral images," *IEEE Trans. Geosci. Remote Sens.*, vol. 44, no. 2, pp. 442–455, Feb. 2006.

[9] D. C. Tseng, H. T. Tseng, and C. L. Chien, "Automatic cloud removal from multi-temporal SPOT images," *Appl. Math. Comput.*, vol. 205, no. 2, pp. 584–600, Nov. 2008.

[10] C. H. Lin, P. H. Tsai, K. H. Lai, and J. Y. Chen, "Cloud removal from multitemporal satellite images using information cloning," *IEEE Trans. Geosci. Remote Sens.*, vol. 51, no. 1, pp. 232–241, Jan. 2013.

[11] L. Lorenzi, F. Melgani, and G. Mercier, "Missing-Area Reconstruction in Multispectral Images Under a Compressive Sensing Perspective," *IEEE Trans. Geosci. Remote Sens.*, vol. 51, no. 7, pp. 3998–4008, Jul. 2013.

[12] J. Xie, L. Xu, and E. Chen, "Image Denoising and Inpainting with Deep Neural Networks," *NIPS*, pp. 1–9, 2012.

[13] K. G. Lore, A. Akintayo, and S. Sarkar, "LLNet: A deep autoencoder approach to natural low-light image enhancement," *Pattern Recognit.*, vol. 61, pp. 650–662, 2017.

[14] R. Wang and D. Tao, “Non-Local Auto-Encoder with Collaborative Stabilization for Image Restoration,” *IEEE Trans. Image Process.*, vol. 25, no. 5, pp. 2117–2129, 2016.

[15] W. Huang, L. Xiao, Z. Wei, H. Liu, and S. Tang, “A New Pan-Sharpening Method With Deep Neural Networks,” *IEEE Geosci. Remote Sens. Lett.*, vol. 12, no. 5, May 2015.

[16] J. Tang, C. Deng, G. B. Huang, and B. Zhao, “Compressed-Domain Ship Detection on Spaceborne optical image using deep neural network and extreme learning machine,” *IEEE Trans. Geosci. Remote Sens.*, vol. 53, no. 3, Mar. 2015.

[17] Y. Chen, Z. Lin, X. Zhao, G. Wang, and Y. Gu, “Deep Learning-Based Classification of Hyperspectral Data,” *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 7, no. 6, Jun. 2014.

[18] Y. Liang, J. Wang, S. Zhang, and Y. Cong, "Learning Visual Co-Occurrence with Auto-Encoder for Image Super-Resolution," in *Proc. APSIPA*, 2014, pp. 1–4.

[19] J. Rissanen, *Stochastic Complexity in Statistical Enquiry*. Singapore: World Scientific, 1989.

[20] L. Bottou, “Stochastic gradient learning in neural networks”, in Proc. *Neuro-Nimes* 91, 1991.

[21] J. Rissanen, “Modeling by shortest data description, “ *Automatica*, vol. 14, no. 5, pp. 465–471, Sep. 1978.

[22] C. C. Liu, “Processing of FORMOSAT-2 daily revisit imagery for site surveillance,” *IEEE Trans. Geosci. Remote Sens.*, vol. 44, no. 11, pp. 3206–3215, Nov. 2006.

[23] A. Baudoin, “Mission Analysis for SPOT 5”, in *IEEE IGARSS*, 1993, vol. 3, pp. 1084.

[24] A. K. Jain, *Fundamentals of Digital Image Processing*. New York: Prentice Hall, 1988.

[25] S. S. Chen, D. L. Donoho, and M. A. Saunders, “Atomic decomposition by basis pursuit,” *SIAM J. Sci. Comput.*, vol. 20, no. 1, pp. 33–61, Aug. 1998.

[26] A. M. Bruckstein, D. L. Donoho, and M. Elad, “From sparse solutions of systems of equations to sparse modelling of signals and images,” *SIAM Rev.*, vol. 51, no. 1, pp. 34–81, Feb. 2009.

[27] N. Wang and Y. Wang, “An image reconstruction algorithm based on compressive sensing using conjugate gradient,” in *Proc. IUCS*, Oct. 2010, pp. 374–377.

[28] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad, “Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decompositions,” in *Proc. 27th Asilomar Conf. Signals., Syst. Comput.*, 1993, pp. 40–44.

[29] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA: Addison-Wesley, 1989.

[30] L. Chambers, *The Practical Handbook of Genetic Algorithms*. New York: Chapman & Hall, 2001.

[31] N. Srinivas, and K. Deb, “Multiobjective function optimization using nondominated sorting genetic algorithms,” *Evol. Comput.*, vol. 2, no. 3, pp. 221–248, 1995.

Chapter 4

1D-Convolutional Neural Networks for Spectroscopic Signal Regression

4.1. Introduction

Chemometrics is the application of mathematical and statistical tools to mostly retrieve chemical information, but may include also physical, biological, and other quantitative and/or qualitative data in order to address problems in chemistry, biology, medicine, and chemical engineering. Associated with chemometrics, spectroscopy allows simultaneous analysis of several parameters and gives possibility to replace many devices, thereby providing analysis solutions merged into a single platform. Spectroscopy has become nowadays a powerful tool for quality control and product analysis in different chemical fields [1]–[5]. However, estimating the concentration of chemical components of interest in a given product is difficult and challenging due to the collinearity between the spectral variables, and the large number of variables to be treated. To address this issue, machine learning can represent an attractive approach since it may exhibit various advantages compared to traditional methods such: 1) as nonlinear modeling capability; 2) good generalization capability thanks to an adequate handling of the overfitting risk; 3) little or no knowledge is required on the problem; 4) limited number of hyperparameters to be tuned; and 5) good results even with limited number of training samples. In this context, many solutions have been proposed in the literature, using predominantly machine learning estimation methods, such as partial least squares regression (PLS regression) [6]–[8], multiple linear regression (MLR) [9], artificial neural networks (ANNs) [10], support vector machines for regression (SVR) [11]–[13], Gaussian process regression (GPR) [14], extreme learning machines (ELM) [15] and fusion approach based on induced ordered weighted averaging operators (IOWA) applied on an ensemble generated by GPR and ELM estimators associated with different kernels [16].

Convolutional Neural Networks (CNNs) are considered as machine learning tools based on learning data models. They were developed by Y. LeCun *et al.* in 1998 [17] as a class of deep feed-forward artificial neural networks. They are now one of the most important deep learning architectures, and they have been applied for numerous tasks in different research fields which deal with images such as remote sensing [18], [19], biomedical imaging [20], [21] and biometrics [22], [23]. However, it can be interesting to explore the effectiveness of CNNs to other applications that do not deal with image data, such as biosignals and chemometric signals. To the best of our knowledge, the works developed by Acquarelli *et al.* [24], Chen *et al.* [25] and Kiranyaz *et al.* [26] are the only works where CNNs are used over 1D input signals. In particular, they were applied for spectroscopic data classification, hyperspectral images classification (pixel-based method) and ECG real time classification, respectively.

In this work, we propose a novel approach for chemometric data analysis by using 1-D convolutional neural networks (1D-CNNs). Conventional CNNs are hierarchical architectures based on an alternation of convolutional layers with subsampling layers and followed by a fully connected layer (or many layers similar to a multilayer perceptron MLP). Regarding 1D-CNNs, the basis of the architecture is similar to that of conventional CNNs. The difference is the use of 1D input data that requires the application of 1D filters on the convolution layers and consequently the modification of the equations of the forward propagation and back propagation during the training phase. Furthermore, in order to deal with the problem of limited data, we propose to optimize the 1D-CNN and estimate the weights of the filters by using an evolutionary method. We use for this purpose the particle swarm optimization (PSO) method. The 1D-CNN and the PSO-1DCNN methods are used here as feature extractor and the extracted features are fed to advanced regression methods such as GPR and SVR (see Figure 4-1).

To the best of our knowledge, the contribution of this work is twofold. First, it explores 1D-CNNs for regression issues, with application to spectroscopic signal regression. Secondly, it is also the first work proposing PSO to train a 1D-CNN for signal regression.

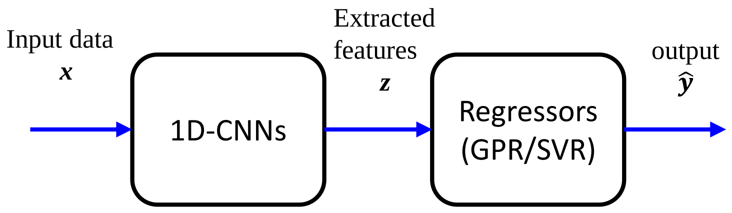


Figure 4-1: General scheme of the proposed method for chemometric data analysis.

The rest of this Chapter is organized as follows. Section II presents the architecture of the proposed 1D-CNN method. The developed PSO-1DCNN is described in Section III. In Section IV, we give brief details about the two methods of regression used in the prediction phase (GPR and SVR). The experimental results are presented in Section V. Finally, conclusions and future developments are reported in Section VI. Mathematical symbols adopted in this chapter are summarized in the Appendix.

4.2. 1D-CNNs

As stated before, 1D-CNNs are used in this work for feature extraction. The last fully connected layer (logistic regression layer) is added for the purpose of adjusting the different parameters just during the back propagation training.

Let us consider a matrix of training samples $\mathbf{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N]'$, where N is the number of training samples and each vector $\boldsymbol{x}_i$ is represented in the d-dimensional measurement space. Let us also denote as $\mathbf{y} = [y_1, y_2, \ldots, y_N]'$ the real output target vector associated with $\mathbf{X}$. A 1D-CNN is constituted of a number L of layers, each layer l ($l=1..L$) is composed of m^l feature signals and performs both convolution and subsampling operations. We assume that the factor of subsampling (ss) is always equal to 2 (ss = 2). Figure 4-2 illustrates a general 1D-CNN architecture.

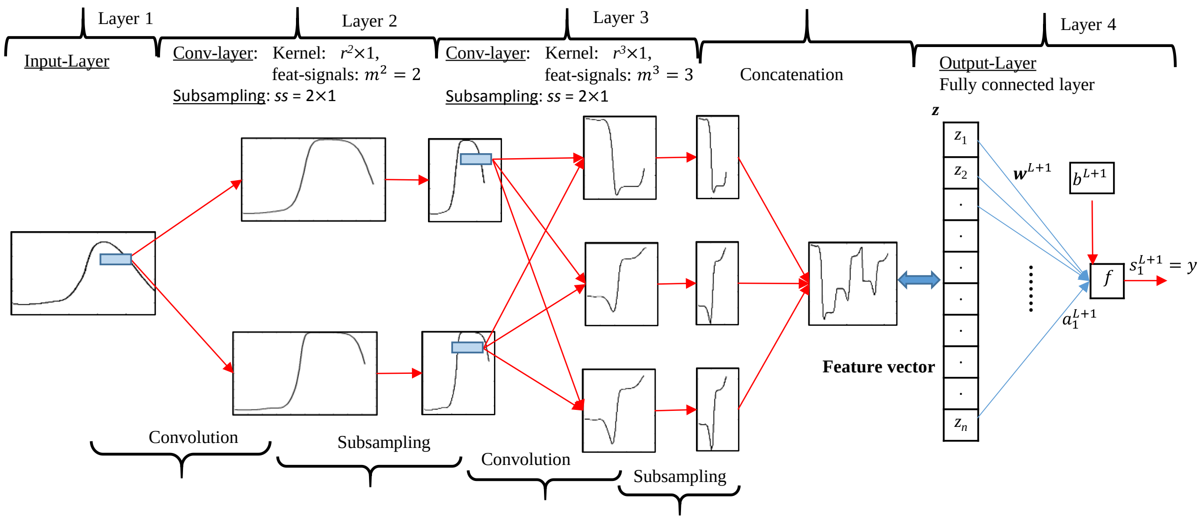


Figure 4-2: 1D-CNN general architecture.

4.2.1 *Forward propagation*

Assuming the current layer *l*, during the forward propagation, the input of each feature signal of the layer *l* is the result of the accumulation of the final output (after the subsampling) of the previous feature signal (*l* - 1) convolved with their proper filters and passed through an nonlinear activation function as follows:

$$\boldsymbol{a}_i^l = b_i^l + \sum_{j=1}^{m^{l-1}} \text{conv1D}(\boldsymbol{w}_{i,j}^l, \boldsymbol{s}_j^{l-1}) \quad (\text{i}=1, \ldots, m^l) \tag{4.1}$$

$$\boldsymbol{s}_i^l = f(\boldsymbol{a}_i^l) \tag{4.2}$$

where $\boldsymbol{a}_i^l$ is the input of the *i*-th feature signal of the layer *l*, b_i^l is the bias of this feature signal and $\boldsymbol{s}_i^l$ is its output, $\boldsymbol{s}_j^{l-1}$ is the output of the *j*-th feature signal on the previous layer (*l*-1), $\boldsymbol{w}_{i,j}^l$ is the filter (kernel) weights vector between the *j*-th feature signal on the *l*-1 layer and the *i*-th feature signal on the *l*-th layer, and $f(\cdot)$ is a nonlinear activation function. Typically, a sigmoid activation function is used, which is expressed by the equation below:

$$f(x) = \frac{1}{1+e^{-x}} \tag{4.3}$$

Regarding the dimension of the vectors on each part of the 1D-CNN, if we suppose that d^l is the dimension of the final output of each feature signal on the layer *l* and r^l is the length of its corresponding kernel (filter), the final output of the feature signal of the next layer *l*+1 (*l*+1 ≤ *L*) is:

$$d^{l+1} = \frac{d^{l} - r^{l} + 1}{2} \tag{4.4}$$

The outputs of feature signals of the last layer L are stacked in one vector $\boldsymbol{z}$ which is the feature vector of the 1D-CNN with a size *n* equals to $d^L \times m^L$. Neurons of this layer are fully connected to the output layer. In our case, since we deal with regression problems and each input $\boldsymbol{x}$ has one target value *y*, the output layer is formed by just one neuron and its output *y* ($\boldsymbol{s}^{L+1}$) is formulated as follows:

$$y = \boldsymbol{s}^{L+1} = f(b^{L+1} + \textstyle\sum_{i=1}^{n}(\boldsymbol{w}_{1,i}^{L+1} \times \boldsymbol{z})) \tag{4.5}$$

4.2.2 Back propagation

In order to train the 1D-CNN, we need to compute first the error at the output layer $E(y)$ and its gradient $\frac{\partial E}{\partial y}$. The objective of calculating this error is to be able to estimate the weights in order to minimize this error during the process of learning. To this end, we need to calculate the derivative of the error with respect to each weight $\frac{\partial E}{\partial \boldsymbol{w}_{i,j}^{l}} = \Delta \boldsymbol{w}_{i,j}^{l}$.

Using the chain rule, we get the following:

$$\frac{\partial E}{\partial \boldsymbol{w}_{i,j}^{l}} = \frac{\partial E}{\partial \boldsymbol{a}_{i}^{l}} \frac{\partial \boldsymbol{a}_{i}^{l}}{\partial \boldsymbol{w}_{i,j}^{l}} \tag{4.6}$$

From equation (4.1), we can deduce that:

$$\frac{\partial \boldsymbol{a}_{i}^{l}}{\partial \boldsymbol{w}_{i,j}^{l}} = \boldsymbol{s}_{j}^{l-1} \tag{4.7}$$

Using equation (4.7), equation (4.6) becomes:

$$\frac{\partial E}{\partial \boldsymbol{w}_{i,j}^{l}} = \frac{\partial E}{\partial \boldsymbol{a}_{i}^{l}} \boldsymbol{s}_{j}^{l-1} = \frac{\partial E}{\partial \boldsymbol{a}_{i}^{l}} f(\boldsymbol{a}_{j}^{l-1}) \tag{4.8}$$

We already know all the values of $\boldsymbol{s}$. In order to compute the gradient, we need to know the values $\frac{\partial E}{\partial \boldsymbol{a}_{i}^{l}}$. By using once more the chain rule, we can write:

$$\frac{\partial E}{\partial \boldsymbol{a}_{i}^{l}} = \frac{\partial E}{\partial \boldsymbol{s}_{i}^{l}} \frac{\partial \boldsymbol{s}_{i}^{l}}{\partial \boldsymbol{a}_{i}^{l}} = \frac{\partial E}{\partial \boldsymbol{s}_{i}^{l}} \frac{\partial}{\partial \boldsymbol{a}_{i}^{l}} f(\boldsymbol{a}_{i}^{l}) = \frac{\partial E}{\partial \boldsymbol{s}_{i}^{l}} f'(\boldsymbol{a}_{i}^{l}) \tag{4.9}$$

We can compute the derivative $\frac{\partial E}{\partial \boldsymbol{a}_{i}^{l}}$ at the current layer by just computing the derivative of the activation function $f'(\boldsymbol{a}_{i}^{l})$. Since we use a sigmoid function, the derivative of the activation function is written as follows:

$$f'(x) = f(x) \times (1 - f(x)) \tag{4.10}$$

Moreover, since we already know the error at the current layer $\frac{\partial E}{\partial \boldsymbol{s}_i^l}$, we can compute then the gradient with respect to the weights used by the considered convolutional layer.

Next task consists of propagating the errors back to the previous layer. By using again chain rule, we can find:

$$\frac{\partial E}{\partial \boldsymbol{s}_j^{l-1}} = \frac{\partial E}{\partial \boldsymbol{a}_i^l} \frac{\partial \boldsymbol{a}_i^l}{\partial \boldsymbol{s}_j^{l-1}} \tag{4.11}$$

From equation (4.1), we can deduce that:

$$\frac{\partial \boldsymbol{a}_i^l}{\partial \boldsymbol{s}_j^{l-1}} = \boldsymbol{w}_{i,j}^l \tag{4.12}$$

Now, we have everything we need to compute $\Delta \boldsymbol{w}_{i,j}^l$, we just need to update weights as follows:

$$\boldsymbol{w}_{i,j}^{l*} = \boldsymbol{w}_{i,j}^l + \eta \Delta \boldsymbol{w}_{i,j}^l \tag{4.13}$$

where $\boldsymbol{w}_{i,j}^{l*}$ corresponds to the weights of the next iteration and η is the learning rate.

4.2.3 Subsampling layers

The main objective of subsampling is to reduce the size of the final feature vector in order to allow the problem remaining tractable. The subsampling can be done by different ways. In our case, in forward propagation, each block of size ss×1 is reduced to a single value. This value equals to the average of its corresponding block. Therefore, it acquires an error computed from backward propagation from the previous layer. This error is then just expanded (upsampled) and forwarded to the next layer (in the backward propagation direction).

We refer the Reader to [27] where more details can be found regarding the general concepts behind CNNs.

4.3. PSO-1DCNN

Particle swarm optimization (PSO) [28] is a stochastic optimization technique which is inspired by social behavior of bird flocking and fish schooling. Similar to other evolutionary computation algorithms, PSO is a population-based search method that exploits the concept of social sharing of information. This means that each individual (called *particle*) of a given population (called *swarm*) can profit from the previous experiences of all other individuals from the same population. During the search process in the d-dimensional solution space, each particle (i.e., candidate solution) will adjust its flying velocity and position according to its own flying experience as well as the experiences of the other companion particles of the swarm.

We propose to introduce PSO to estimate the different parameters (weights) of the 1D-CNN as an alternative to the standard back propagation algorithm. The advantage of using PSO is to overcome the problem of overfitting due to limited number of training samples. Furthermore, the complexity of the network will be reduced by using a layer-wise optimization, where PSO will be applied on each layer independently as described in the following figure.

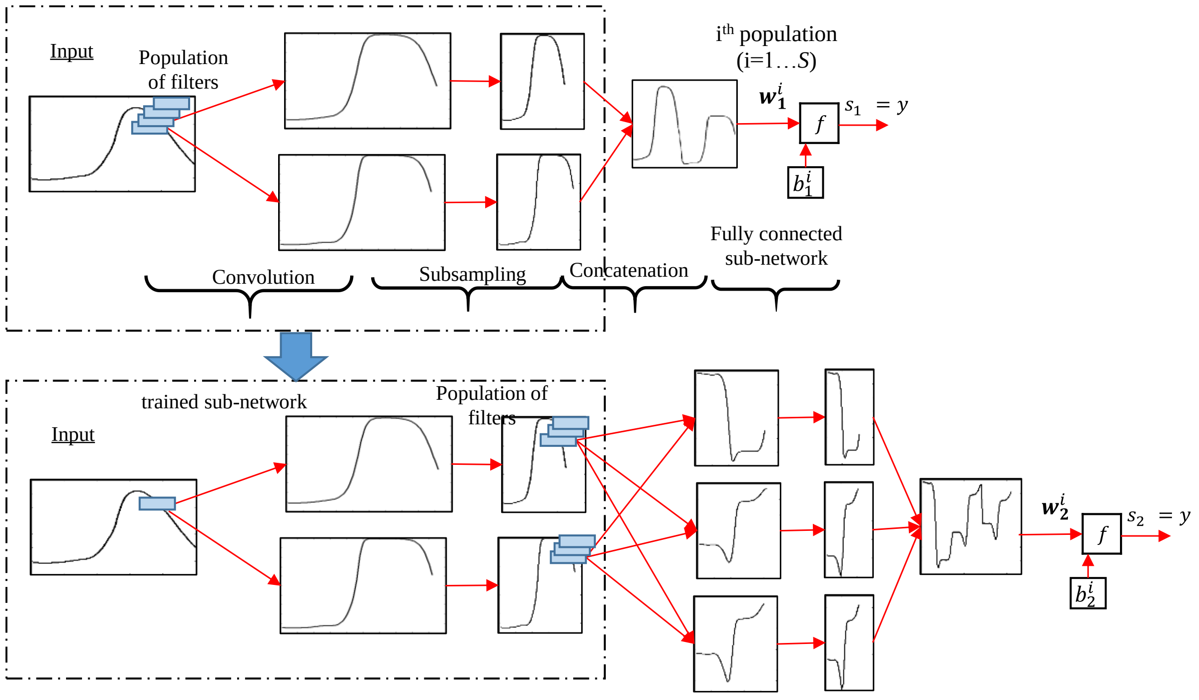


Figure 4-3: Architecture of the PSO-1DCNN.

In the following, we will describe briefly the main concepts of the basic PSO algorithm. Let us consider a swarm of size *S*. Each particle P_i (i = 1, 2,..., *S*) from the swarm is characterized by: 1) its current position $\mathbf{p}_i(t) \in \Re^n$, which refers to a candidate solution of the optimization problem at iteration *t*; 2) its velocity $\mathbf{v}_i(t) \in \Re^n$; and 3) the best position $\mathbf{p}_{bi}(t) \in \Re^n$ that is identified during its past trajectory. Let $\mathbf{p}_g(t) \in \Re^n$ be the best global position found over all trajectories that are traveled by the particles of the swarm. Since the PSO is applied in cascade on each layer of the 1DCNN, the coordinates of a particle will encode the values of all the weights characterizing that layer. The position optimality is measured by means of one or more fitness functions that are defined in relation to the considered optimization problem. During the search process, the particles move according to the following equations:

$$\begin{cases} \mathbf{v}_i(t+1) = \omega_0 \mathbf{v}_i(t) + c_1.r_1(t)\big(\mathbf{p}_{bi}(t) - \mathbf{p}_i(t)\big) \\ \qquad\qquad\qquad + c_2.r_2(t)\left(\mathbf{p}_g(t) - \mathbf{p}_i(t)\right) \\ \mathbf{p}_i(t+1) = \mathbf{p}_i(t) + \mathbf{v}_i(t) \end{cases} \tag{4.14}$$

where $r_1(t)$ and $r_2(t)$ are random variables that are drawn from a uniform distribution in the range [0, 1] to provide a stochastic weighting of the different components participating in the particle velocity definition. c_1 and c_2 are two acceleration constants regulating the relative velocities with respect to the best local and global positions, respectively. The inertia weight ω_0 is used as a tradeoff between global and local exploration capabilities of the swarm. Large values of this parameter permit better global exploration, whereas small values lead to a fine search in the solution space. First part of the equation (4.14) allows the computation of the velocity at iteration $t+1$ for each particle in the swarm by combining linearly its current velocity and the distances that separate the current particle position from its best previous position and the best global position, respectively. The updating of the particle position is performed with the second part of equation (4.14), which is iterated until convergence of the search process is reached.

4.4. Prediction

Similarly to conventional 2DCNNs, the feature signals of the last subsampling layer are gathered and concatenated to form the feature vector which will be considered as a new representation of the input sample. For the prediction, we will resort to two different methods of regression, namely GPR and SVR, which will be fed with features extracted by the 1D-CNN. The choice of these advanced machine learning methods is motivated by their successful applications in different research fields [29]–[34], as well as in the chemometric data analysis [16]. Such a success is mainly explained by their lower sensitivity to the risk of overfitting, and thus their higher generalization capability with respect to traditional regression methods. Moreover, they involve a very limited number of hyperparameters to be tuned (depending on the adopted covariance/kernel function). A brief description of these two methods is provided in the following.

4.4.1 GPR

Let us consider a set of N training samples $\mathbf{Z} = [\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_N]'$, where each vector $\mathbf{z}_i$ is the feature vector of dimension n extracted by the 1D-CNN from the input sample $\mathbf{x}_i$. Let us also denote as $\mathbf{y} = [y_1, y_2, \ldots, y_N]'$ the corresponding target vector associated with $\mathbf{Z}$. The objective of the GPR is to deduce a relationship between the set of training samples $\mathbf{Z}$ and the target vector $\mathbf{y}$ which is considered as the sum of a latent function $\mathbf{f}$ and a noise component $\boldsymbol{\varepsilon}_\mathbf{n}$, where:

$$\mathrm{f} \sim \mathrm{GP}(0, \mathrm{K}(\mathrm{Z}, \mathrm{Z})) \tag{4.15}$$

$$\boldsymbol{\varepsilon}_\mathbf{n} \sim \mathrm{N}(0, \sigma_n^2 \mathbf{I}) \tag{4.16}$$

The first equation implies that a Gaussian process GP(.,.) is assumed over the latent function **f** which is considered as a collection of random variables which follow a joint Gaussian distribution [35]. **K** is the covariance matrix built by means of a kernel function *K* computed on all the training sample pairs. The second equation states that the output target vector is corrupted by a noise that follows a Gaussian distribution with zero-mean and variance equals to σ_n^2. Since the latent function **f** and the noise **ε** are statistically independent, the noisy observations **y** are also modeled with GP, that is,

$$\mathbf{y} \sim \mathrm{GP}(0, \mathbf{K}(\mathbf{Z}, \mathbf{Z}) + \sigma_n^2 \mathbf{I}) \tag{4.17}$$

or equivalently

$$p(\mathbf{y}|\mathbf{Z}) = \mathrm{N}(0, \mathbf{K}(\mathbf{Z}, \mathbf{Z}) + \sigma_n^2 \mathbf{I}) \tag{4.18}$$

In the prediction phase, given the set of training samples, the best estimation of the output value y_t associated with an unknown sample $\mathbf{z}_\mathrm{t}$ is given by:

$$\hat{y}_\mathrm{t}|\mathbf{Z}, \mathbf{y}, \mathbf{z_t} \sim E\{y_\mathrm{t}|\mathbf{Z}, \mathbf{y}, \mathbf{z_t}\} = \int y_\mathrm{t}\, p(y_\mathrm{t}|\mathbf{Z}, \mathbf{y}, \mathbf{z_t}) \mathrm{d}y \tag{4.19}$$

It is clear from the last equation that in order to estimate the output value, the knowledge of the predictive distribution $p(y_\mathrm{t}|\mathbf{Z}, \mathbf{y}, \mathbf{z_t})$ is required. For this purpose, the joint distribution of the known observations **y** and the desired function value y_t should be first derived. Thanks to the assumption of a GP over **y** and to the marginalization property of GPs, this joint distribution is Gaussian. The desired predictive distribution can be derived simply by conditioning the joint one to the noisy observations **y** and takes the expression:

$$p(y_\mathrm{t}|\mathbf{Z}, \mathbf{y}, \mathbf{z_t}) = \mathrm{N}(\mu_\mathrm{t}, \sigma_\mathrm{t}^2) \tag{4.20}$$

where:

$$\mu_\mathrm{t} = \begin{bmatrix} k(\mathbf{z}_\mathrm{t}, \mathbf{z}_1) \\ \vdots \\ k(\mathbf{z}_\mathrm{t}, \mathbf{z}_N) \end{bmatrix}^\mathrm{T} . [\mathbf{K}(\mathbf{Z}, \mathbf{Z}) + \sigma_n^2 \mathbf{I}]^{-1} . \mathbf{y} \tag{4.21}$$

and

$$\sigma_\mathrm{t}^2 = k(\mathbf{z}_\mathrm{t}, \mathbf{z}_\mathrm{t}) - \begin{bmatrix} k(\mathbf{z}_\mathrm{t}, \mathbf{z}_1) \\ \vdots \\ k(\mathbf{z}_\mathrm{t}, \mathbf{z}_N) \end{bmatrix}^\mathrm{T} . [\mathbf{K}(\mathbf{Z}, \mathbf{Z}) + \sigma_n^2 \mathbf{I}]^{-1} . \begin{bmatrix} k(\mathbf{z}_\mathrm{t}, \mathbf{z}_1) \\ \vdots \\ k(\mathbf{z}_\mathrm{t}, \mathbf{z}_N) \end{bmatrix} \tag{4.22}$$

These are the key equation in the GPR approach. The mean μ_t expresses the best output value estimate for the considered sample and the variance σ_t^2 represents the confidence measure associated by the model to the output.

A central role in the GPR model is played by the kernel function $k(\mathbf{z}_\mathrm{i}, \mathbf{z}_\mathrm{j})$ (covariance) as it embeds the geometric structure of the training samples. Through it, it is possible to define the prior knowledge about the output function we wish to learn. The parameters of the covariance function can be determined empirically (for example by cross-validation). As an alternative, the intrinsic nature of GPs allows a Bayesian treatment for the estimation of parameter vectors $\boldsymbol{\Theta}$. To this end, one may resort to the ML-II estimation procedure. It consists in the maximization of the marginal likelihood with respect to $\boldsymbol{\Theta}$, that is, the integral of the likelihood times the prior.

$$p(\boldsymbol{y}|\mathbf{Z}) = p(\boldsymbol{y}|\mathbf{Z}, \boldsymbol{\Theta}) = \int p(\mathbf{y}|\mathbf{f}, \mathbf{Z}, \boldsymbol{\Theta}) p(\mathbf{f}|\mathbf{Z}, \boldsymbol{\Theta}) \mathrm{d}\mathbf{f} \tag{4.23}$$

with the marginalization over the latent function $\mathbf{f}$. Under GP modeling, both the prior and the likelihood follow Gaussian distributions. After some manipulation, it is possible to show that

the log marginal likelihood can be written as [35]:

$$\log p(\boldsymbol{y}|\mathbf{Z}, \boldsymbol{\Theta}) = -\frac{1}{2}\mathbf{y}^\mathrm{T}.[\mathbf{K}(\mathbf{Z}, \mathbf{Z}) + \sigma_n^2\mathbf{I}]^{-1}.\mathbf{y} - \frac{1}{2}\log|\mathbf{K}(\mathbf{Z}, \mathbf{Z}) + \sigma_n^2\mathbf{I}| - \frac{N}{2}\log(2\pi) \tag{4.24}$$

This equation is characterized by the sum of three terms. The first is the only one that involves the target observations. It represents the capability of the model to fit the data. The second one is the model complexity penalty, and the third term is a normalization constant. From an implementation viewpoint, this maximization problem can easily be solved by a gradient-based search routine [35].

4.4.2 SVR

Support Vector machine Regression (SVR) performs linear regression in a feature space using an epsilon-intensive loss (ε-SVM). This technique is based on the idea of deducing an estimate $\hat{g}(\mathbf{z}_i)$ of the true but unknown relationship $y_i = g(\mathbf{z}_i)$ $(i = 1, \dots, N)$ between the vector of observations $\mathbf{z}_i$ and the target value y_i such that: 1) $\hat{g}(\mathbf{z}_i)$ has, at most, ε deviation from the desired targets y_i and 2) it is as smooth as possible [36], [37]. This is performed by mapping the data from the original feature space of dimension n to a higher n'-dimensional transformed feature space (kernel space), i.e., $\Phi(\mathbf{z}_i) \in \Re^{n'}$ $(n' > n)$, to increase the flatness of the function and, by consequence, to approximate it in a linear way as follows:

$$\hat{g}(\mathbf{z}_i) = \omega^*.\Phi(\mathbf{z}_i) + b^* \tag{4.25}$$

Therefore, SVR is formulated as minimization of the following cost function:

$$\psi(\omega, \xi) = \frac{1}{2}\|\omega\|^2 + c\sum_{i=1}^{N}(\xi_i + \xi_i^*) \tag{4.26}$$

subject to:

$$\begin{cases} y_i - (\omega.\Phi(\mathbf{z}_i) + b) \leq \varepsilon + \xi_i \\ (\omega.\Phi(\mathbf{z}_i) + b) - y_i \leq \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \geq 0 \end{cases} \tag{4.27}$$

where, ξ_i and ξ_i^* are the slack variables that measure the deviation of the training sample $\mathbf{z}_i$ outside the ε-intensive zone. c is a parameter of regularization that allows tuning the tradeoff between the flatness of the function $\hat{g}(\mathbf{z})$ and the tolerance of deviations larger than ε.

The aforementioned optimization problem can be transformed through a Lagrange functional into a dual optimization problem expressed in the original dimensional feature space in order to lead to the following dual prediction model:

$$\hat{g}(\mathbf{z}) = \sum_{i \in U}(\alpha_i - \alpha_i^*)K(\mathbf{z}_i, \mathbf{z}) + b^* \tag{4.28}$$

where K is a kernel function, U is a subset of indices ($i = 1, \dots, N$) corresponding to the nonzero Lagrange multipliers α_i's or α_i^*'s. The training samples that are associated to nonzero weights are called *SVs*. The kernel $K(\cdot,\cdot)$ should be chosen such that it satisfies the condition imposed by the Mercer's theorem, such as the Gaussian kernel functions [36], [37].

4.5. Experimental results

4.5.1 Dataset description and performance evaluation

In the experiments, three different datasets are used, each has been decomposed in two sets. The first is a training set for model learning and selection, while the second is a test set for assessment and evaluation of the trained model.

The first dataset "Orange Juice" deals with the problem of determining the concentration of saccharose in orange juice samples by near-infrared reflectance spectroscopy [38], [39]. The training set contains 150 samples and the test set contains 68 samples, with 700 spectral variables (features). Those features are the absorbance (log 1/R) at 700 wavelengths between 1100 and 2500 nm (where R is the light reflectance on the sample surface). The saccharose concentration ranges from 0% to 95.2% by weight. Figure 4-4 shows the spectra of the training samples.

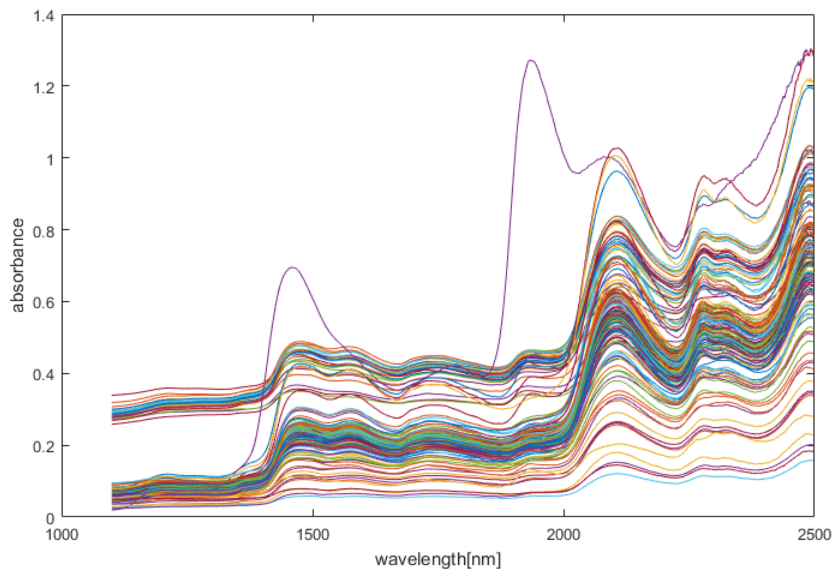


Figure 4-4: Near-infrared spectra of orange juice training samples.

The second dataset is related to the determination of alcohol content by mid-infrared spectroscopy in wine samples [38]–[40]. The dataset contains 124 samples: 94 as training samples (see Figure 4-5) and 30 as test samples, with 256 spectral variables that are the absorbance (log 1/T) at 256 wave numbers between 4000 and 400 cm^{-1} (where T is the light transmittance through the sample thickness).

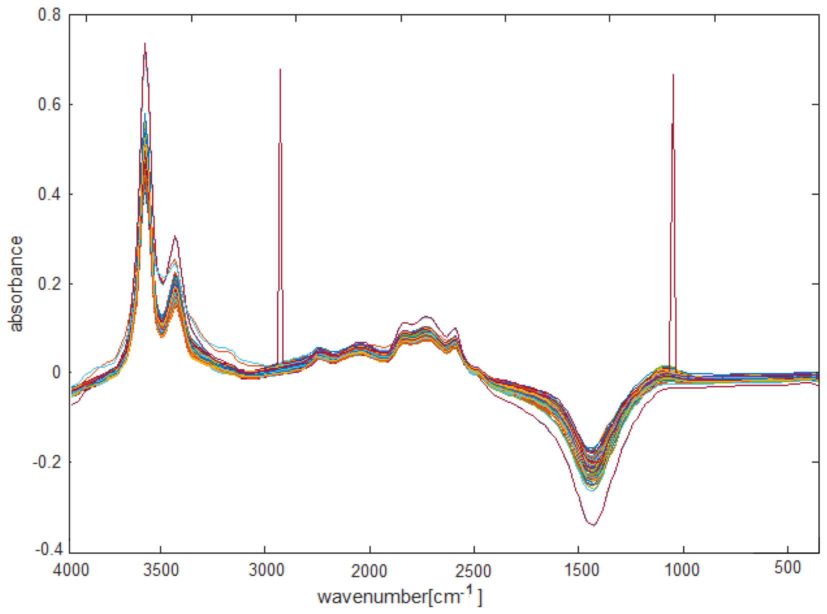


Figure 4-5: Mid-infrared spectra of wine training samples.

The third dataset deals with the prediction of the fat content of meat samples analyzed by near-infrared transmittance spectroscopy [40], [41]. The corresponding data were recorded on a Tecator Infratec Food and Feed Analyzer working in the wavelength range 850-1050 nm by the Near Infrared

Transmission (NIT) principle. The spectrometer records light transmittance through the meat samples at 100 channel spectrum of absorbance in the specified range. Each sample contains finely chopped pure meat with different moisture, fat and protein contents. Those contents, measured in percent by weight, are determined by analytic chemistry. There are 215 samples in this dataset, 172 samples are used as training (see Figure 4-6) and 43 samples as test.

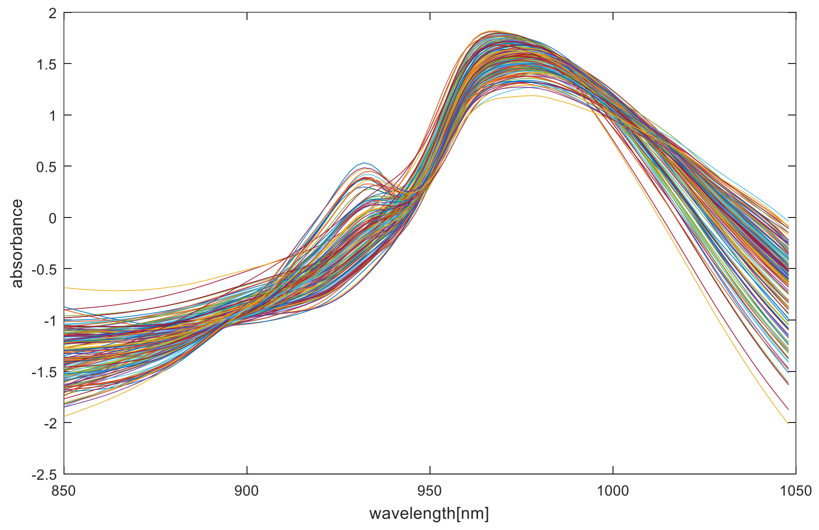


Figure 4-6: Near-infrared spectra of Tecator training samples.

In order to evaluate the regression methods and to perform a direct comparison with state-of-the-art results, we adopt the Normalized Mean Square Error (NMSE) metric. Since it normalizes the error with respect to the range of variation of the output, it is usually preferred over the popular MSE metric. It is given by:

$$NMSE = \frac{\sum_{i=1}^{N_t}(y_{it}-\hat{y}_{it})^2}{(\mathrm{var}\{y,y_t\})} \tag{4.29}$$

where N_t is the total number of test samples, y_{it} and $\hat{y}_{it}$ are the real and estimated output for the i-th test sample $\boldsymbol{x}_{it}$ and $\mathrm{var}\{y, y_t\}$ is the variance of all output samples.

We also use a gain in accuracy measure in order to give information about how much our methods improved those of the state-of-the-art methods [16]. This measure is given by:

$$gain = 100 \times \frac{(NMSE_2 - NMSE_1)}{NMSE_2} \tag{4.30}$$

where $NMSE_1$is the NMSE of our proposed method and $NMSE_2$ is the NMSE of the state-of-the-art method.

4.5.2 Parameter setting

The architecture of a 1D-CNN involves three main parameters: number of layers L, number of feature signals m^l and length of kernel r^l of each layer l (l=1,..,L). To compute the best parameter values, we use a cross-validation technique with a number of folds equal to 3. Due to the large number of combinations and also to the limited number of training samples, we decided to limit the maximum number of possible layers $L_{\max}$ to 3. Moreover, the maximum number of feature signals in each layer l ($m^l_{\max}$) is fixed to 5 with step of 1 and maximum kernel size on each layer l ($r^l_{\max}$) is fixed to 10% of the current layer features length d^l with step of 5 for the first dataset and 3 for the two other datasets. The obtained best values of the parameters by cross-validation are listed in Table 4.1. Regarding the PSO-1DCNN method, since the training is performed layer-by-layer, the number of possible cases in the cross-validation is reduced compared to the previous procedure. Thus, the number of feature signals in each layer l ($m^l_{\max}$) is fixed to 10 while keeping the other parameters similar to the previous method. The optimal parameters found by cross-validation are presented in Table 4.2. Figure 4-7 and Figure 4-8 show examples of the best NMSE found by cross-validation by changing the number of layers and changing the number of feature signals (in the case of one layer), respectively. We can observe from these figures that the best architecture needs to be neither large nor shallow in order to get the best results.

Regarding GPR and SVR, we use the Matérn covariance function for GPR and the Radial Basis Function (RBF) for SVR as kernel functions. During the cross validation, the parameter of regularization of SVR 'c' and the width of its kernel function 'γ' were varied in the range [1, 10^4] and [10^{-3}, 5] respectively. The ε value of the insensitive tube was fixed to 10^{-3}.

Table 4.1: Best parameter values of the 1D-CNN for each dataset.

Dataset	**Layer 1**		**Layer 2**	
	m^1	r^1	m^2	r^2
Orange Juice	5	9	1	19
Wine	1	17	5	9
Tecator	5	13	3	7

Table 4.2: Best parameter values of the PSO-1DCNN for each dataset.

Dataset	**Layer 1**		**Layer 2**		**Layer 3**	
	m^1	r^1	m^2	r^2	m^3	r^3
Orange Juice	1	31	1	21	1	13
Wine	4	15	4	7	0	0
Tecator	5	7	1	7	5	5

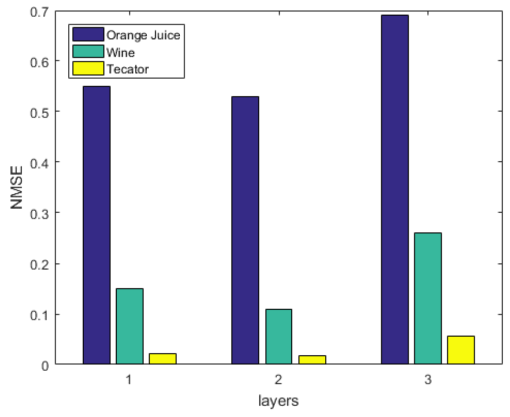


Figure 4-7: Effect of number of layers on estimation error.

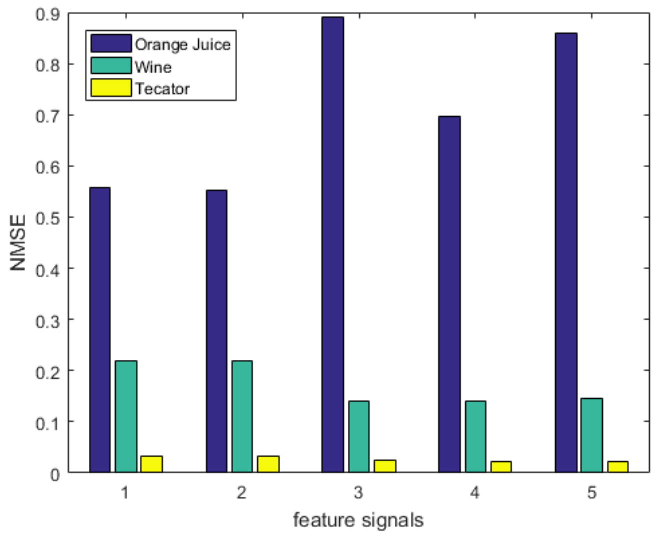


Figure 4-8: Effect of number of feature signals on estimation error.

4.5.3 Results

In order to evaluate our method, as already explained in the methodology part, we chose two known and effective regression methods, namely GPR and SVR, to apply them on the features extracted from the two proposed methods, i.e. 1D-CNN and PSO-1DCNN. It is worth recalling that, during the training of 1D-CNN and PSO-1DCNN, just a linear regression (LR) layer is put on top of the neural network. The accuracies achieved with LR are also analyzed. From [16], we took for comparison results achieved by the well-known partial least square regressor (PLSR), as well as the GPR and SVR fed with all available original features. All the results in terms of NMSE are reported in Table 4.3, Table 4.4 and Table 4.5 for Orange Juice, Wine and Tecator datasets, respectively. The obtained gains in accuracy for the three datasets are presented in Table 4.6.

Table 4.3: Results for the Orange Juice dataset.

Features	**Regression Method**	**No. features**	**NMSE**
[16]	PLSR	13	0.1626
[16]	GPR	700	1.1350
1D-CNN	LR	164	0.2869
PSO-1DCNN	LR	73	0.6007
1D-CNN	GPR	164	0.1661
PSO-1DCNN	GPR	73	**0.1569**
[16]	SVR	700	0.2488
1D-CNN	SVR	164	0.1462
PSO-1DCNN	SVR	73	**0.1416**

Table 4.4: Results for the Wine dataset.

Features	**Regression Method**	**No. features**	**NMSE**
[16]	PLSR	9	0.00610
[16]	GPR	256	0.00287
1D-CNN	LR	280	0.03150
PSO-1DCNN	LR	232	0.21010
1D-CNN	GPR	280	0.00261
PSO-1DCNN	GPR	232	**0.00226**
[16]	SVR	256	0.00590
1D-CNN	SVR	280	0.00320
PSO-1DCNN	SVR	232	**0.00282**

Table 4.5: Results for the Tecator dataset.

Features	**Regression Method**	**No. features**	**NMSE**
[16]	PLSR	12	0.02840
[16]	GPR	100	0.00124
1D-CNN	LR	57	0.00210
PSO-1DCNN	LR	45	0.03770
1D-CNN	GPR	57	0.00098
PSO-1DCNN	GPR	45	**0.00079**
[16]	SVR	100	0.00250
1D-CNN	SVR	57	**0.00076**
PSO-1DCNN	SVR	45	0.00088

Table 4.6: Gain in accuracy for the 3 datasets.

Dataset	**Gain (%)**	
	GPR	**SVR**
Orange Juice	86.2	43.1
Wine	21.3	52.2
Tecator	36.3	69.6

From these tables, it can be seen that using the features extracted from the two proposed methods (1D-CNN and PSO-1DCNN) coupled with GPR and SVR techniques provide better results in term of NMSE compared to the state-of-the-art methods [16]. Moreover, from Table 4.6, the calculated gains in accuracy by comparing the same regression method (GPR or SVR) fed with all original features and with best extracted features (among 1D-CNN and PSO-1DCNN) show clearly how important are the achievable improvements with the proposed method. As for the contribution of resorting to PSO to estimate the parameters of the CNN, it can be seen clearly on all cases (except the case of the Tecator dataset by using SVR) that there are noticeable improvements compared to the results yielded by using the classical back propagation algorithm. This encourages using evolutionary methods to train a CNN-based architecture especially with limited number of training samples. Also because the training time for both developed methods takes only few minutes and small memory space. As expected, the traditional gradient-based method is faster as, on an average, it takes about 1 minute with respect to 3 minutes for the PSO.

In more detail, considering the best obtained results among the two proposed methods, from Table 4.3 (orange juice dataset), the NMSE using the PSO-1DCNN features is equal to 0.1569 with GPR and 0.1416 with SVR. On the other hand, from [16], NMSE equals to 1.1350 and 0.2488 with GPR and SVR, respectively. For this dataset, the PLSR method with just 13 features achieved a better result compared to GPR and SVR with a NMSE equal to 0.1626. The gain in accuracy obtained by our method with respect to the state-of-the-art methods is equal to 86.2% and 43.1% for GPR and SVR, respectively. Regarding the Wine dataset (Table 4.4), the PSO-1DCNN features provide NMSE equal to 0.00226 and 0.00282 with GPR and SVR, respectively. In [16], it is equal to 0.00287 for GPR and 0.0059 for SVR. The corresponding gains in accuracy are equal to 21.3% and 52.2%, respectively. For the Tecator dataset (Table 4.5), GPR and SVR provide NMSE equal to 0.00079 and 0.00076 by using PSO-1DCNN and 1D-CNN features, respectively. In [16], they are equal to 0.00124 and 0.0025, respectively. In this case, the gain in accuracy is 36.3% for GPR and 69.6% for SVR. It is noteworthy that in most of the cases the PSO-1DCNN worsened the results for the LR regressor. The explanation is that, in all scenarios, the PSO-1DCNN is trained with LR, and thus the LR accuracy is used as fitness function. Accordingly, a high risk is incurred that the obtained PSO-1DCNN (with LR) overfits the data and provides poorer results with respect to traditional gradient descent optimization. Regarding the two other regressors (SVR and GPR), they are not affected by this issue as they were not part of the optimization process, but they just exploit the features provided by the 1DCNN optimized by the PSO on the LR.

Qualitatively, we can notice from Figure 4-10 and Figure 4-11 which correspond to datasets 2 and 3, respectively, that the estimated values by the proposed method are almost identical to the real ones. By contrast, in dataset 1 (Figure 4-9), more mismatches can be observed due to the higher complexity of this dataset. However, as the above quantitative assessments show, the estimation errors are on an average very low for this dataset.

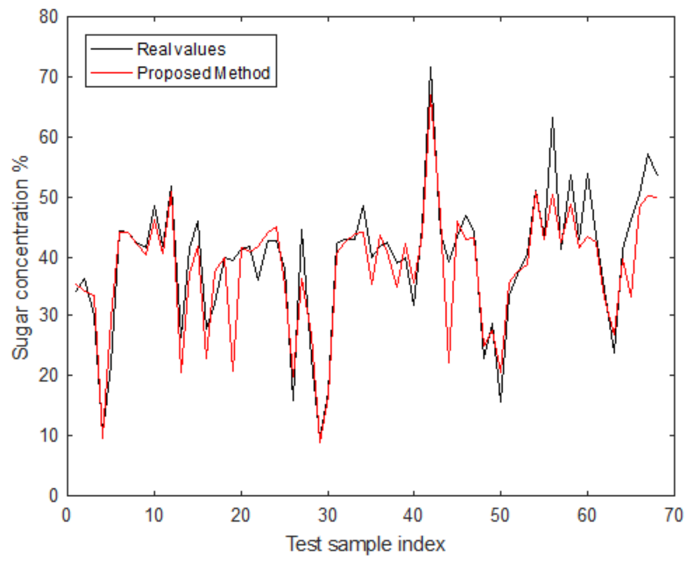


Figure 4-9: Sample-by-sample comparison between estimated and real output values for the test set of the Orange Juice dataset.

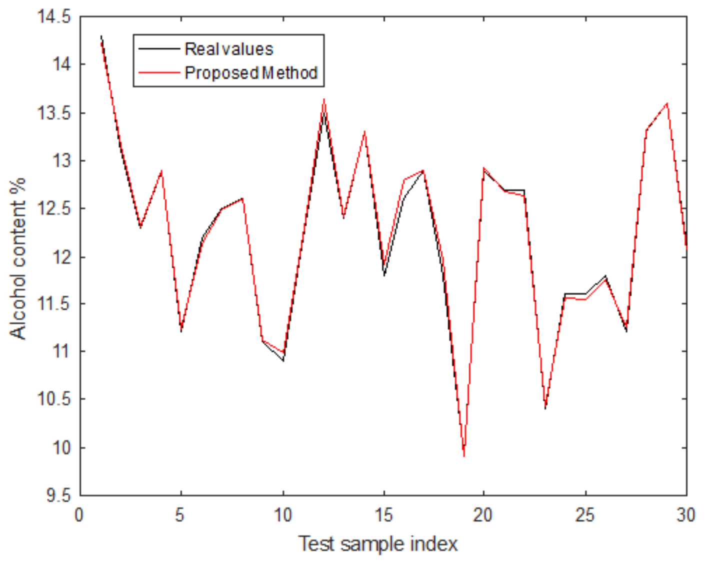


Figure 4-10: Sample-by-sample comparison between estimated and real output values for the test set of the Wine dataset.

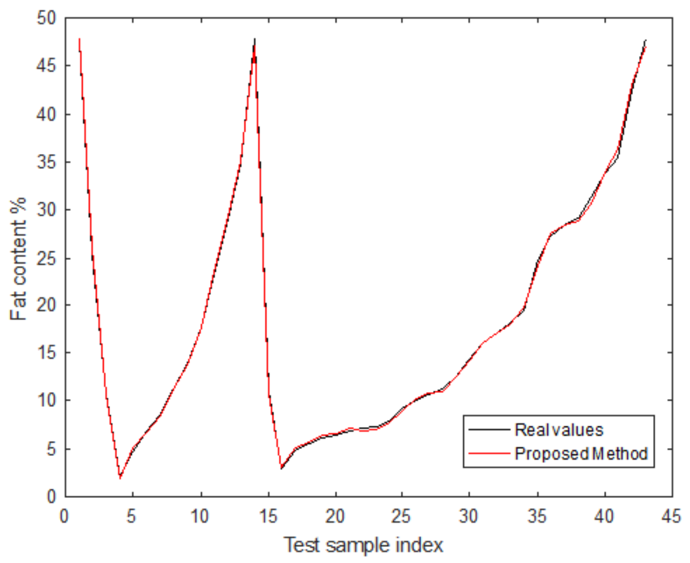


Figure 4-11: Sample-by-sample comparison between estimated and real output values for the test set of the Tecator dataset.

4.6. Conclusion

In this work, we propose new methods for chemometric data analysis based on convolutional neural networks. In particular, we modify the standard CNN architecture to adapt it to 1D input data. The proposed 1-D CNN architecture is thus based on an alternation of convolutional layers with subsampling layers and connected at the end to a linear regression layer. The convolution is applied using 1D filters in the convolution layers and pooling (subsampling) is applied by averaging the samples over a given sliding 1D window. The estimation of the architecture weights is performed using two methods. The first one consists of the standard back propagation algorithm. The second approach is based on a layerwise particle swarm optimization. Next step consists of applying a regression method (GPR and SVR, or any other method) using features provided by the proposed methods.

The experimental results show that results yielded by the proposed approach are very promising. Indeed, on the three considered datasets, and for both methods of regression, CNN-like extracted features can provide significant gains in accuracy (between 21.3% and 86.2%), suggesting that CNNs are able to extract powerful feature for regression on 1D signals. Moreover, the contribution of the PSO in the training of the neural network architecture appears valuable on two datasets over three, and with a very limited additional computational overload.

4.7. Appendix: List of Mathematical Symbols

- η: learning rate (1D-CNN)
- ω_0: inertia weight (PSO)
- ω: weight vector (SVR)
- $\boldsymbol{\varepsilon_n}$: noise component (GPR)
- ε : insensitivity parameter (SVR)
- α_i: Lagrange multipliers (SVR)
- ξ_i: slack variable (SVR)
- $\boldsymbol{a}_i^l$: i-th input feature signal of layer l (1D-CNN)
- b_i^l: bias of the i-th feature signal of layer l (1D-CNN)
- b: bias (SVR)
- c_1 c_2 : acceleration constants (PSO)
- c: regularization parameter (SVR)
- d^l : size of feature vector of layer l (1D-CNN)
- E: error criterion (1D-CNN)
- **f:** latent function (GPR)
- f: activation function (1D-CNN)
- $g(\cdot)$: input-output relationship (SVR)
- **I:** identity matrix (GPR)
- **K:** covariance matrix (GPR)
- K**:** kernel function (SVR)
- K**:** covariance function (GPR)
- L : number of layers (1D-CNN)
- l: index of current layer (1D-CNN)
- m^l : number of feature signals of layer l (1D-CNN)
- $N(\cdot,\cdot)$: Normal distribution (GPR)
- N: number of training samples
- N_t: number of test samples
- n: output size (1D-CNN)
- n': dimension of transformed feature space (SVR)
- P_i: i-th particle (PSO)
- $\mathbf{p}_i(t)$: position of the i-th particle P_i at iteration t (PSO)
- $\mathbf{p}_g(t)$: best global position (PSO)
- $\mathbf{p}_{bi}(t)$: best position during past trajectory (PSO)
- r^l : size of the filter of the lth layer (1D-CNN)
- $r_1(t)$ and $r_2(t)$: random numbers generated at iteration t (PSO)
- S: number of particles (PSO)
- ss: subsampling factor (1D-CNN)
- $\boldsymbol{s}_i^l$: output of i-th feature signal of layer l (1D-CNN)
- t: iteration index (PSO)
- U : subset of indices (SVR)

- $\mathbf{v}_i(t)$: velocity of the i-th particle P_i at iteration t (PSO)
- $\boldsymbol{w}_{i,j}^{l}$: filter weights between the j-th feature signal on the l-1 layer and the i-th feature signal on the l-th layer (1D-CNN)
- $\mathbf{X}=[\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N]'$: training samples in original space (1D-CNN)
- $\mathbf{y} = [y_1, y_2, \ldots, y_N]'$: targets of training samples
- $\mathbf{Z} = [\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_N]'$: training samples in 1D-CNN induced space (GPR and SVR).

4.8. References

[1] R. Gente *et al.*, "Quality Control of Sugar Beet Seeds With THz Time-Domain Spectroscopy," *IEEE Trans. Terahertz Sci. Technol.*, vol. 6, no. 5, pp. 754–756, Sep. 2016.

[2] P. Przybylek, "A new method for indirect measurement of water content in fibrous electro-insulating materials using near-infrared spectroscopy," *IEEE Trans. Dielectr. Electr. Insul.*, vol. 23, no. 3, pp. 1798–1804, Jun. 2016.

[3] I. Hiroaki, N. Toyonori, and T. Eiji, "Measurement of pesticide residues in food based on diffuse reflectance IR spectroscopy," *IEEE Trans. Instrum. Meas.*, vol. 51, no. 5, pp. 886–890, Oct. 2002.

[4] A. G. Mignani, L. Ciaccheri, A. A. Mencaglia, H. Ottevaere, and H. Thienpont, "Spectroscopy AS a #x201C;green #x201D; technique for food quality and safety applications," in *Technical Digest of the Eighteenth Microoptics Conference*, 2013, pp. 1–2.

[5] S. Nishizawa, H. Morita, T. Iwamoto, M. W. Takeda, and M. Tani, "Terahertz time-domain spectroscopy applied to nondestructive evaluation of pharmaceutical products," in *2011 International Conference on Infrared, Millimeter, and Terahertz Waves*, 2011, pp. 1–2.

[6] S. Wold, M. Sjöström, and L. Eriksson, "PLS-regression: a basic tool of chemometrics," *Chemom. Intell. Lab. Syst.*, vol. 58, no. 2, pp. 109–130, 2001.

[7] B. M. Nicolai, K. I. Theron, and J. Lammertyn, "Kernel PLS regression on wavelet transformed NIR spectra for prediction of sugar content of apple," *Chemom. Intell. Lab. Syst.*, vol. 85, no. 2, pp. 243–252, 2007.

[8] B. G. Arrobas *et al.*, "Raman spectroscopy for analyzing anthocyanins of lyophilized blueberries," in *SENSORS, 2015 IEEE*, 2015, pp. 1–4.

[9] R. K. H. Galvão *et al.*, "Multivariate analysis of the dielectric response of materials modeled using networks of resistors and capacitors," *IEEE Trans. Dielectr. Electr. Insul.*, vol. 20, no. 3, pp. 995–1008, 2013.

[10] A. Verikas and M. Bacauskiene, "Using artificial neural networks for process and system modelling," *Chemom. Intell. Lab. Syst.*, vol. 67, no. 2, pp. 187–191, 2003.

[11] H. Li, Y. Liang, and Q. Xu, "Support vector machines and its applications in chemistry," *Chemom. Intell. Lab. Syst.*, vol. 95, no. 2, pp. 188–198, 2009.

[12] O. Devos, C. Ruckebusch, A. Durand, L. Duponchel, and J.-P. Huvenne, "Support vector machines (SVM) in near infrared (NIR) spectroscopy: Focus on parameters optimization and model interpretation," *Chemom. Intell. Lab. Syst.*, vol. 96, no. 1, pp. 27–33, 2009.

[13] D. Porro, N. Hdez, I. Talavera, O. Núñez, Á. Dago, and R. J. Biscay, "Performance evaluation of relevance vector machines as a nonlinear regression method in real-world chemical spectroscopic data," in *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, 2008, pp. 1–4.

[14] T. Chen, J. Morris, and E. Martin, "Gaussian process regression for multivariate spectroscopic calibration," *Chemom. Intell. Lab. Syst.*, vol. 87, no. 1, pp. 59–71, 2007.

[15] J. Peng, L. Li, and Y. Y. Tang, "Combination of activation functions in extreme learning machines for multivariate calibration," *Chemom. Intell. Lab. Syst.*, vol. 120, pp. 53–58, 2013.

[16] H. AlHichri, Y. Bazi, N. Alajlan, F. Melgani, S. Malek, and R. R. Yager, "A novel fusion approach based on induced ordered weighted averaging operators for chemometric data analysis," *J. Chemom.*, vol. 27, no. 12, pp. 447–456, 2013.

[17] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[18] G. Cheng, P. Zhou, and J. Han, "Learning rotation-invariant convolutional neural networks for object detection in VHR optical remote sensing images," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 12, pp. 7405–7415, 2016.

[19] P. Ghamisi, Y. Chen, and X. X. Zhu, "A Self-Improving Convolution Neural Network for the Classification of Hyperspectral Data," *IEEE Geosci. Remote Sens. Lett.*, vol. 13, no. 10, pp. 1537–1541, 2016.

[20] M. Srinivas, D. Roy, and C. K. Mohan, "Discriminative feature extraction from X-ray images using deep convolutional neural networks," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 917–921.

[21] Z. Cui, J. Yang, and Y. Qiao, "Brain MRI segmentation with patch-based CNN approach," in *Control Conference (CCC), 2016 35th Chinese*, 2016, pp. 7026–7031.

[22] R. F. Nogueira, R. de Alencar Lotufo, and R. C. Machado, "Fingerprint Liveness Detection Using Convolutional Neural Networks," *IEEE Trans. Inf. Forensics Secur.*, vol. 11, no. 6, pp. 1206–1213, 2016.

[23] A. Rikhtegar, M. Pooyan, and M. T. Manzuri-Shalmani, "Genetic algorithm-optimised structure of convolutional neural network for face recognition applications," *IET Comput. Vis.*, 2016.

[24] J. Acquarelli, T. van Laarhoven, J. Gerretzen, T. N. Tran, L. M. C. Buydens, and E. Marchiori, "Convolutional neural networks for vibrational spectroscopic data analysis," *Anal. Chim. Acta*, vol. 954, pp. 22–31, Feb. 2017.

[25] Y. Chen, H. Jiang, C. Li, X. Jia, and P. Ghamisi, "Deep Feature Extraction and Classification of Hyperspectral Images Based on Convolutional Neural Networks," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 10, pp. 6232–6251, Oct. 2016.

[26] S. Kiranyaz, T. Ince, and M. Gabbouj, "Real-Time Patient-Specific ECG Classification by 1-D Convolutional Neural Networks," *IEEE Trans. Biomed. Eng.*, vol. 63, no. 3, pp. 664–675, 2016.

[27] I. Goodfellow, Y. Bengio and A. Courville, *Deep Learning*, The MIT Press: London, 2016.

[28] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *IEEE International Conference on Neural Networks, 1995. Proceedings*, 1995, vol. 4, pp. 1942–1948 vol.4.

[29] R. Dürichen, M. A. Pimentel, L. Clifton, A. Schweikard, and D. A. Clifton, "Multitask Gaussian processes for multivariate physiological time-series analysis," *IEEE Trans. Biomed. Eng.*, vol. 62, no. 1, pp. 314–322, 2015.

[30] D. Kwon, M. H. Azarian, and M. Pecht, "Remaining-Life Prediction of Solder Joints Using RF Impedance Analysis and Gaussian Process Regression," *IEEE Trans. Compon. Packag. Manuf. Technol.*, vol. 5, no. 11, pp. 1602–1609, 2015.

[31] H.-C. Yen and C.-C. Wang, "Cross-Device Wi-Fi Map Fusion with Gaussian Processes," *IEEE Trans. Mob. Comput.*, vol. 16, no. 1, pp. 44–57, 2017.

[32] H. Sun *et al.*, “Accurate Age Estimation of Bloodstains Based on Visible Reflectance Spectroscopy and Chemometrics Methods,” *IEEE Photonics J.*, vol. 9, no. 1, pp. 1–14, 2017.

[33] K. Y. Bae, H. S. Jang, and D. K. Sung, “Hourly Solar Irradiance Prediction Based on Support Vector Machine and Its Error Analysis,” *IEEE Trans. Power Syst.*, 2016.

[34] L. Garg and V. Sahula, “Macromodels for Static Virtual Ground Voltage Estimation in Power-Gated Circuits,” *IEEE Trans. Circuits Syst. II Express Briefs*, vol. 63, no. 5, pp. 468–472, 2016.

[35] C. E. Rasmussen, “Gaussian processes for machine learning,” 2006.

[36] V. N. Vapnik and V. Vapnik, *Statistical learning theory*, vol. 1. Wiley New York, 1998.

[37] A. J. Smola and B. Schölkopf, “A tutorial on support vector regression,” *Stat. Comput.*, vol. 14, no. 3, pp. 199–222, 2004.

[38] “Datasets provided by Prof. Marc Meurens, Université catholique de Louvain, BNUT, meurens@bnut.ucl.ac.be. Wine and orange juice datasets available from: http://www.ucl.ac.be/mlg/.” .

[39] F. Rossi, A. Lendasse, D. François, V. Wertz, and M. Verleysen, “Mutual information for the selection of relevant variables in spectrometric nonlinear modelling,” *Chemom. Intell. Lab. Syst.*, vol. 80, no. 2, pp. 215–226, 2006.

[40] C. Krier, F. Rossi, D. François, and M. Verleysen, “A data-driven functional projection approach for the selection of feature ranges in spectra with ICA or cluster analysis,” *Chemom. Intell. Lab. Syst.*, vol. 91, no. 1, pp. 43–53, 2008.

[41] “Tecator meat sample dataset. Available from: http://lib.stat.cmu.edu/datasets/tecator.” .

Chapter 5

Convolutional SVM

5.1. Introduction

Deep learning is a family of machine learning based on learning data representation. The learning can be supervised or unsupervised depending on the adopted architecture. The most famous deep architectures are the Stacked AutoEncoder (SAE) [1], which consists in a concatenation of many AutoEncoders (AEs), Deep Belief Network (DBN) [2], which is based on a series of Restricted Boltzmann Machine (RBM) and Convolutional neural networks (CNNs) [3].

Recently deep convolutional neural networks (CNNs) have achieved impressive results on a variety of applications including image classification [4]-[8], object detection [9]-[12], and image segmentation [13], [14]. Thanks to their sophisticated structure, they have the ability to learn powerful generic image representations in a hierarchical way compared to state-of-the-art shallow methods based on handcrafted features. Modern CNNs are made up of several alternating convolution and pooling layers followed by some fully connected layers. The feature maps produced by the convolution layers are usually fed to a nonlinear gating function such as the Rectified Linear Unit (ReLU). Then the output of this activation function can further be subjected to normalization (i.e., local response normalization). The whole CNN architecture is trained end-to-end using the backpropagation algorithm with dropout regularization [15] to reduce overfitting. It is worth recalling that recent deeper CNNs (winner of the ImageNet Large-Scale Visual Recognition ILSVRC14 and ILSVRC15 challenges) use inception modules [7] and residual learning [6].

Usually, CNNs perform well for analysing datasets with large labeled data. However they are prone to overfitting when dealing with datasets with limited labeled data. For these scenarios, it is has been shown that it is more interesting to transfer knowledge from CNNs (such as AlexNet [8], VGG-VD [16], GoogLeNet [7], and ResNet [6]) pretrained on an auxiliary recognition task with very large labeled data instead of training a CNN from scratch [17]-[20]. While the possible transfer learning solutions include fine-tuning the pretrained CNN on the labeled data of the target dataset or to exploit the CNN feature representations with an external classifier. We refer the reader to [18] where the authors introduce and investigate several factors affecting the transferability of these representations.

In this chapter, we propose an alternative strategy for training CNNs based on SVMs for handling these scenarios in a multilabeling context. SVMs are among the most popular supervised classifiers available in the literature. They rely on the margin maximization principle which makes them less sensitive to overfitting problems. They have been intensively used in conjunction with handcrafted features for solving various recognition problems. In addition, as discussed previously, they are also commonly placed on the top of a CNN feature extractor for carrying out the classification task [18]. Here, we use them to estimate the filters of the CNN convolutional layer. We call this network as convolutional SVM (CSVM).

Each convolution layer uses a set of linear SVMs as filter banks, which are convolved with the feature maps produced by the precedent layer to generate a new set of features maps. For the first convolution layer, the SVM filters are convolved with the original input images. The SVM weights of each convolution layer are computed directly in a supervised way by training on patches (extracted from the previous layer) representing the objects of interest. The high level representations obtained by the network are fed again to a linear SVM classifier for carrying out the classification task.

The rest of this chapter is organized as follows. In Section II, we give a description of the proposed CSVM architecture. First sub-section will be devoted for a case of single object detection, and the second sub-section for the general case (multi-object detection). The experimental results are presented in Section III. Finally, conclusions and future developments are reported in section IV.

5.2. Proposed Methodology

Let us consider a set of M training RGB images $\{\mathbf{X}_i, \boldsymbol{y}_i\}_{i=1}^{M}$ of size $r \times c$, where $\mathbf{X}_i \in \mathcal{R}^{r \times c \times 3}$ and r and c refer the number of rows and columns of the images. Let us assume also $\boldsymbol{y}_i = [y_1, y_2, \dots, y_K]'$ is its corresponding label vector, where K represents the total number of targeted classes. In a multilabel setting, the label $y_i = 1$ is set to 1 if the corresponding object is present, otherwise it is set to 0. Figure 5-1 shows a general view of the proposed multilabel classification system. For simplicity, we present the method for detecting the presence of a single object than we extend it to the case of the multilabel classification.

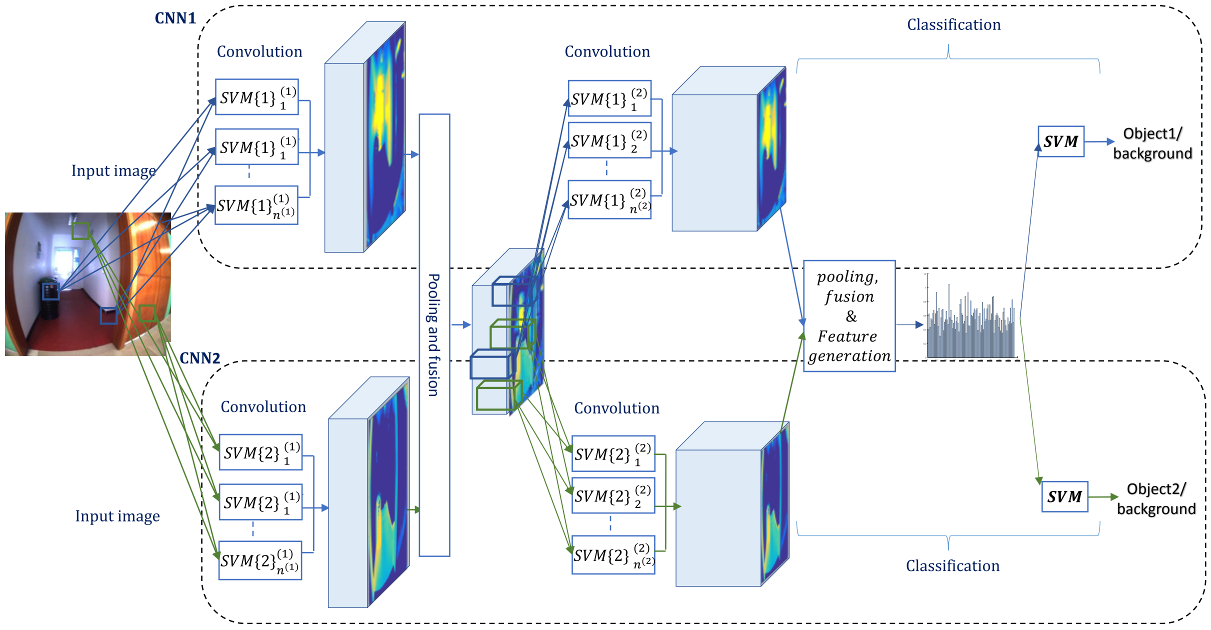


Figure 5-1: Estimating the weights of the convolution layer with SVM for detecting the presence of two objects in a given input image.

5.2.1 *Monolabel classification*

We recall that the application of our method to other advanced architectures such as those based on inception and residual modules is straightforward. As mentioned in the introduction, our main contribution is to use SVM for estimating the weights of the convolution filters in a supervised way. In the following, we detail this method for the first convolution layer.

In a binary classification setting, the training set $\{\mathbf{X}_i, y_i\}_{i=1}^{M}$ is supposed to be composed of M positive and negative RGB images and the corresponding class labels are set to $y_i \in \{+1, -1\}$. The positive images contain the object of interest, whereas the negatives ones just represent background. From each image $\mathbf{X}_i$, we extract a set of patches of size $h \times h \times 3$ and represent them as feature vectors $\mathbf{x}_i$ of dimension d, with $d = h \times h \times 3$. After processing the M training images, we obtain a large training set $Tr^{(1)} = \{\mathbf{x}_i, y_i\}_{i=1}^{m^{(1)}}$ of size $m^{(1)}$ as shown in Figure 5-2.

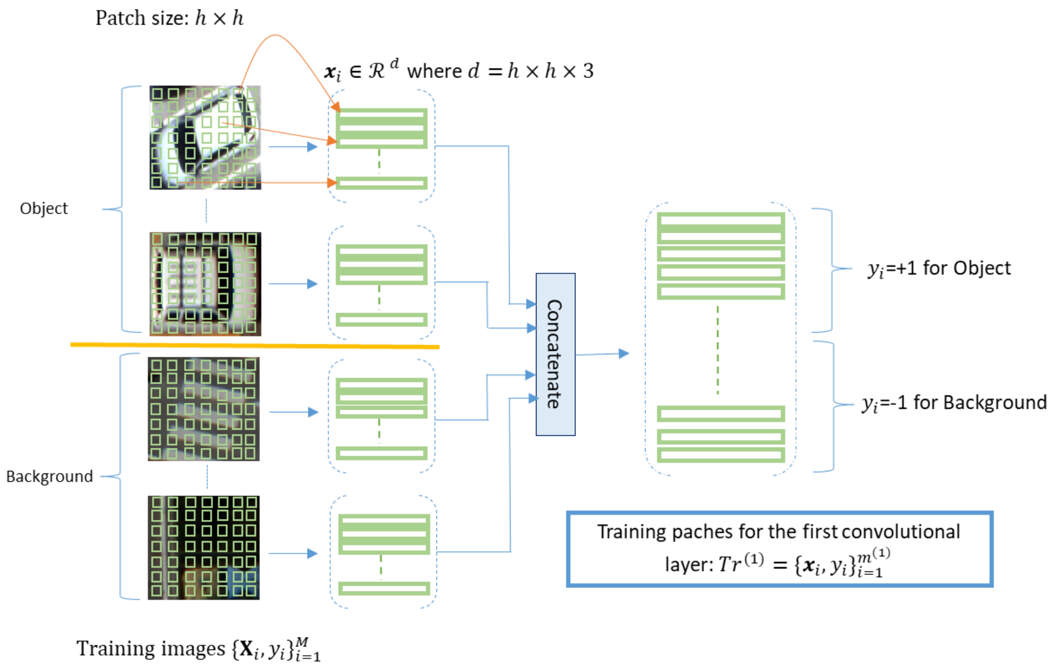

Patch size: $h \times h$
$\boldsymbol{x}_i \in \mathcal{R}^d$ where $d = h \times h \times 3$
Object
Background
Concatenate
y_i=+1 for Object
y_i=-1 for Background
Training paches for the first convolutional layer: $Tr^{(1)} = \{\boldsymbol{x}_i, y_i\}_{i=1}^{m^{(1)}}$
Training images $\{\mathbf{X}_i, y_i\}_{i=1}^{M}$

Figure 5-2: Training set generation for the first convolution layer.

Next, we learn a set of SVM filters on different sub-training sets $Tr_{sub}^{(1)} = \{\mathbf{x}_i, y_i\}_{i=1}^{l}$ of size l randomly sampled from the training set $Tr^{(1)}$. The weight vector $\boldsymbol{w} \in \boldsymbol{\mathcal{R}}^d$ and bias $b \in \boldsymbol{\mathcal{R}}$ of each SVM filter are determined by optimizing the following optimization problem [21], [22].

$$\min_{\boldsymbol{w},b} \boldsymbol{w}^T\boldsymbol{w} + C\sum_{i=1}^{l}\xi(\boldsymbol{w}, b; \boldsymbol{x}_i, y_i) \tag{5.1}$$

where C is a penalty parameter, which can be estimated through cross-validation. As loss function, we use $\xi(\boldsymbol{w}, b; \boldsymbol{x}_i, y_i) = \max(1 - y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b), 0)$ referred as the hinge loss. After training, we represent the weights of the SVM filters as $\left\{\mathbf{w}_k^{(1)}\right\}_{k=1}^{\boldsymbol{n}^{(1)}}$, where $\mathbf{w}_k^{(1)} \in \boldsymbol{\mathcal{R}}^{h\times h\times 3}$ refers to *k*th-SVM filter weight matrix, while $n^{(1)}$ is the number of filters. Then, the complete weights of the first CNN convolution layer are grouped into filter bank of four dimensions $\boldsymbol{W}^{(\mathbf{1})} \in \boldsymbol{\mathcal{R}}^{h\times h\times 3\times n^{(1)}}$.

In order to generate the feature maps, we simply convolve each training image $\{\mathbf{X}_i\}_{i=1}^{M}$, with the obtained filters as usually done in standard CNN to generate a set of 3D hyper-feature maps $\{\mathbf{H}_i^{(1)}\}_{i=1}^{M}$. Here $\mathbf{H}_i^{(1)} \in \boldsymbol{\mathcal{R}}^{r^{(1)}\times c^{(1)}\times n^{(1)}}$ is the new feature representation of image $\mathbf{X}_i$ composed of $n^{(1)}$ feature maps (Figure 5-2). To obtain the *k*th feature map $\mathbf{h}_{ki}^{(1)}$, we convolve the *k*th filter with a set of sliding windows of size $h \times h \times 3$ (with a predefined stride) over the training image $\mathbf{X}_i$ as shown in Figure 5-3:

$$\mathbf{h}_{ki}^{(1)} = f\left(\mathbf{X}_i * \mathbf{w}_k^{(1)}\right), k = 1, \dots, n^{(1)} \tag{5.2}$$

where $*$ is the convolution operator and f is the activation function.

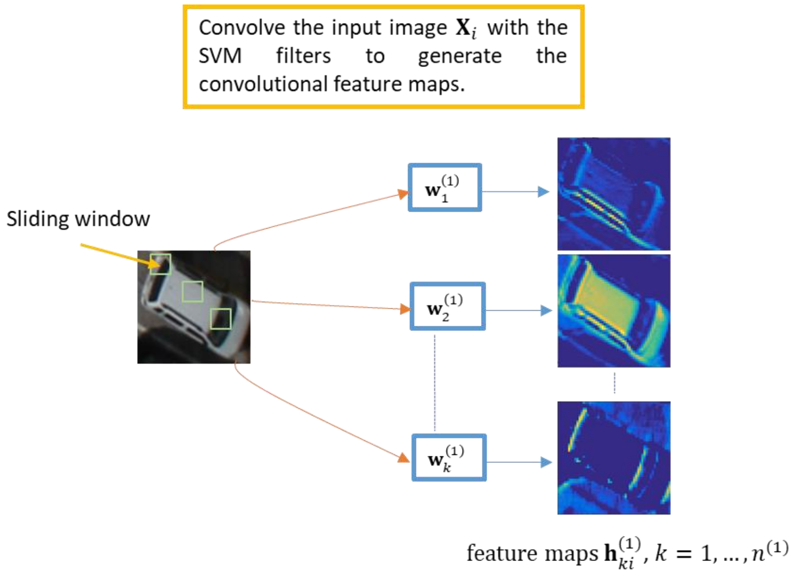

Convolve the input image $\mathbf{X}_i$ with the SVM filters to generate the convolutional feature maps.
Sliding window
$\mathbf{w}_1^{(1)}$
$\mathbf{w}_2^{(1)}$
$\mathbf{w}_k^{(1)}$
feature maps $\mathbf{h}_{ki}^{(1)}, k = 1, \dots, n^{(1)}$

Figure 5-3: Supervised feature map generation.

5.2.2 Multilabel classification

In the last sub-section, we showed how our proposed CSVM works with single class (object). When the problem posed is a multilabel classification, many CNNs are used depending on the number of existing classes. Each CNN apply the convolution on the current image separately using SVM as described in the previous sub-section. After the pooling (subsampling), a fusion strategy is applied in order to share the representation levels between the different objects. In our case we opt for the max strategy in order to highlight the different objects existing and detected by the CNNs. Figure 5-4 presents an example of the interest of this strategy with two classes (objects). The input image contains two objects, Laboratories (object1) and Bins (object2). The first CNN try to highlight the first object while the second CNN is devoted for the second object. The output maps after pooling of the two CNNs are fused in order to get a new map (image) where the two concerned objects are highlighted as it can be seen in Figure 5-4.

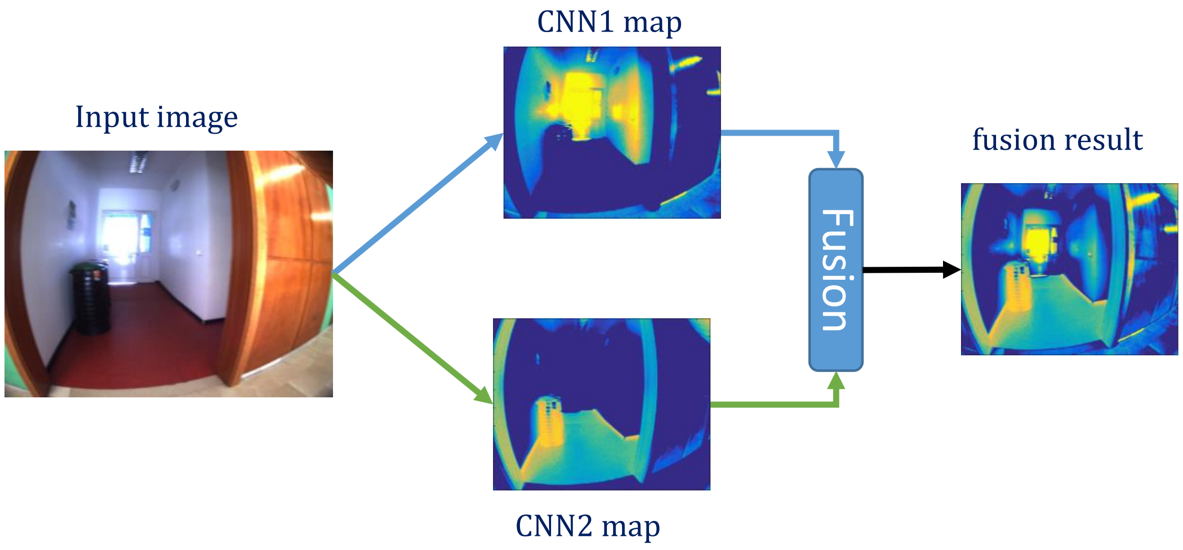


Figure 5-4: Example of fusion of output maps of two CNNs.

5.3. Experimental results

5.3.1 Dataset description and performance evaluation

The set of Images used in this work is divided on three groups. The first two groups of images have been taken at two different indoor spaces of the faculty of science of University of Trento (Italy). The size of each image is 320x240. The first ensemble amounts for a total of 130 images, which was divided into 58 training and 72 testing images. The second set accounts for 131 images, split up into 61 training images, and 70 for testing purposes. The third group of images represents an outdoor environment and was acquired at different locations across the city of Trento located in the Trentino-Alto Adige region. The locations were selected based on their importance as well as the density of people frequenting them. This third dataset comprises two hundred (200) images, which were split up into training and testing subsets (i.e., 100 each). The size of each image is 275x175. It is noteworthy that the training images for all datasets were selected in such a way to cover all the predefined objects in the considered indoor and outdoor environments. To this end, we have selected the objects deemed to be the most important ones in the considered spaces. Regarding the first dataset, 15 objects were considered as follows: ‘External Window’, ‘Board’, ‘Table’, ‘External Door’, ‘Stair Door’, ‘Access Control Reader’, ‘Office’, ‘Pillar’, ‘Display Screen’, ‘People’, ‘ATM’, ‘Chairs’, ‘Bins’, ‘Internal Door’, and ‘Elevator’. Whereas, for the second set, the list was the following: ‘Stairs’, ‘Heater’, ‘Corridor’, ‘Board’, ‘Laboratories’, ‘Bins’, ‘Office’, ‘People’, ‘Pillar’, ‘Elevator’, ‘Reception’, ‘Chairs’, ‘Self Service’, ‘External Door’, and ‘Display Screen’. Finally, for the last dataset, a total of 25 objects were definded as follows: ‘People’, ‘Building’, ‘Bar(s)’, ‘Monument(s)’, ‘Chairs/Benches’, ‘Green ground’, ‘Vehicle(s)’, ‘Stairs’, ‘Walk path / Sidewalk’, ‘Fence / Wall’, ‘Tree(s) / Plant(s)’, ‘Garbage can(s)’, ‘Bus stop’, ‘Crosswalk’, ‘River’, ‘Roundabout’, ‘Pole(s) / Pillar(s)’, ‘Shop(s)’, ‘Supermarket(s)’, ‘Pound/Birds’, ‘Underpass’, ‘Bridge’, ‘Railroad’, ‘Admiration building’, ‘Church’, and ‘Traffic signs’.

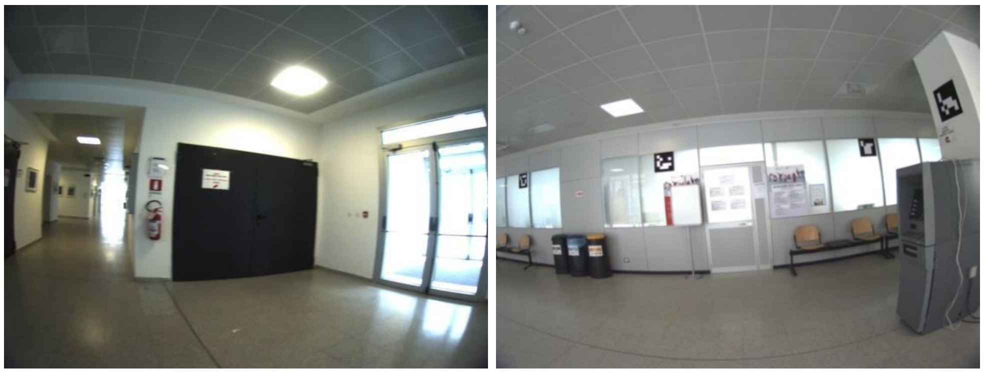


Figure 5-5: Example of images of the first dataset.

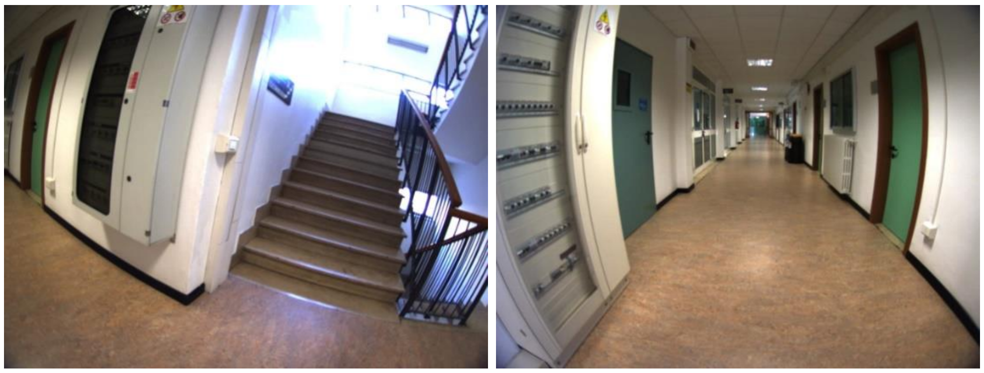


Figure 5-6: Example of images of the second dataset.

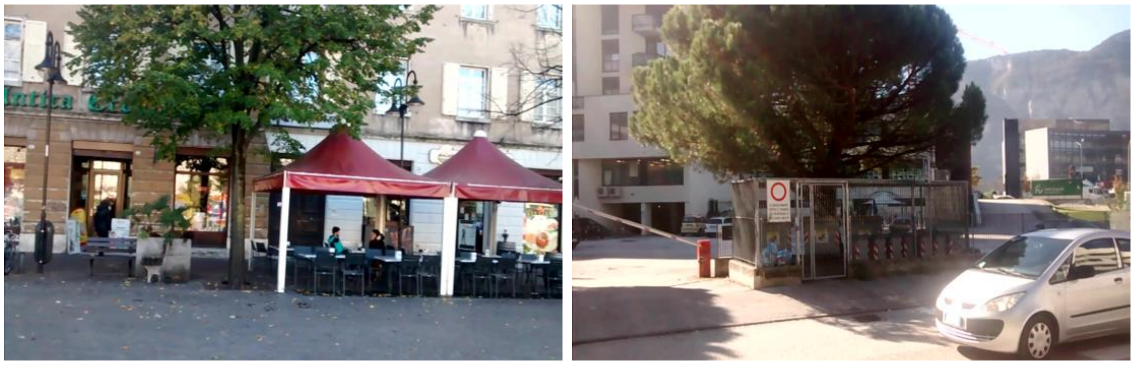


Figure 5-7: Example of images of the third dataset.

For evaluation purposes, we use the well-known sensitivity (SEN) and specificity (SPE) measures:

$$\text{SEN} = \frac{True\ Positive}{True\ Positive + False\ Negative} \tag{5.3}$$

$$\text{SPE} = \frac{True\ Negative}{True\ Negative + False\ Positive} \tag{5.4}$$

The sensitivity expresses the classification rate of real positive cases i.e., the efficiency of the algorithm towards detecting existing objects. The specificity, on the other hand, underlines the tendency of the algorithm to detect the true negatives i.e., the non-existing objects. We also propose to compute the average of the two earlier measures as follows:

$$\text{AVG} = \frac{\text{SEN}+\text{SPE}}{2} \tag{5.5}$$

5.3.2 Parameter setting

The architecture of the proposed CSVM involves several free parameters. In our work, we will focus on three main parameters which are: number of layers L, number of maps (kernels) m^l and length of kernel r^l of each layer l (l=1,..,L). To compute the best parameter values, we use a cross-validation technique with a number of folds equal to 3. Due to the limited number of training samples and the large number of possible combinations, we decided to limit the maximum number of possible layers $L_{\max}$ to 3. Moreover, the maximum number of kernels in each layer l ($m^l_{\max}$) is fixed to 512 with step of 2^i (i=0,…,9) and maximum kernel size on each layer l ($r^l_{\max}$) is fixed to 10% of the size of the current map (we consider the minimum between the high and the width as the considered size) with step of 2. The obtained best values of the parameters by cross-validation are listed in Table 5.1. We can deduce from this Table that only one layer is enough for the first dataset, whereas the two other datasets need a second layer (which is the last layer) to get the best performances. Concerning number and size of kernels (maps), dataset3 presents the simplest architecture with just one kernel on the first layer and two kernels on the second one with size of 3. Contrary to the other datasets where they require big number of kernels.

Regarding SVM, we use a linear SVM (with the linear kernel Function). During the cross validation, the parameter of regularization of SVM 'c' was fixed randomly in the range [1, 10^2]. The ε value of the insensitive tube was fixed to 10^{-3}.

Table 5.1: Best parameter values of the CSVM for each dataset.

Dataset	Layer 1		Layer 2	
	m^1	r^1	m^2	r^2
Dataset1	512	7	/	/
Dataset2	32	7	256	3
Dataset3	1	3	2	3

5.3.3 Results

In order to evaluate our method, we chose to compare it with results obtained using three different pretrained convolutional neural networks which are ResNet [6], GoogLeNet [7] and VDCNs [16]. All the results in terms of accuracies are reported in Table 5.2, Table 5.3 and Table 5.4 for Dataset 1, Dataset 2 and Dataset 3, respectively.

Table 5.2: Comparison of classification rates on Dataset 1.

Method	SEN (%)	SPE (%)	AVG (%)
ResNet	71.16	93.84	82.50
GoogLeNet	78.65	94.34	86.49
VDCNs	74.53	**94.58**	84.55
CSVM	**89.14**	84.26	**86.70**

Table 5.3: Comparison of classification rates on Dataset 2.

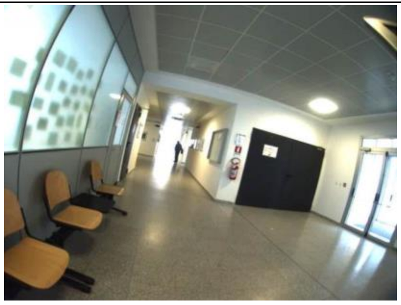
Methods	SEN (%)	SPE (%)	AVG (%)
ResNet	89.54	96.38	**92.96**
GoogLeNet	83.63	**96.86**	90.25
VDCNs	81.81	96.14	88.98
CSVM	**93.64**	92.17	92.90

Table 5.4: Comparison of classification rates on Dataset 3.

Methods	SEN (%)	SPE (%)	AVG (%)
ResNet	64.17	92.40	78.29
GoogLeNet	62.50	93.27	77.88
VDCNs	64.32	**93.98**	79.15
CSVM	**80.79**	82.27	**81.53**

From Table 5.2, Table 5.3 and Table 5.4, and in terms of average accuracy, it can be seen that in 8 cases among 9, our proposed method by far outperforms the different pretrained CNNs. In 7 cases the improvement is clearly important (more than 2%). Only in one case for Dataset2 where ResNet method gives slightly better results than our method (92.96% compared to 92.90%).

From qualitative point of view of the results, Figure 5-8, Figure 5-9 and Figure 5-10 present examples of classification results for Dataset1, Dataset2 and Dataset3, respectively.



Existing objects: ‘External Door’, ‘Stair Door’, ‘Office, ‘Chairs’, ‘Internal Door’

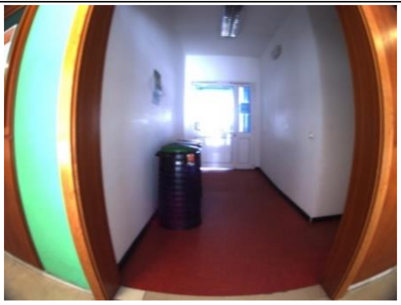
CSVM

Detected Objects: ‘External Door’, ‘Stair Door’, ‘Office, ‘Display Screen’, ‘Chairs’, ‘Internal Door’
Missed objects: Ø

GoogLeNet

Detected Objects: ‘External Door’, ‘Stair Door’, ‘Office, ‘Pillar’, ‘Chairs’,
Missed objects: ‘Internal Door’

Figure 5-8: Example of classification results with CSVM and GoogLeNet on Dataset1. In red are highlighted false positives.



Existing objects: 'bins', 'office'

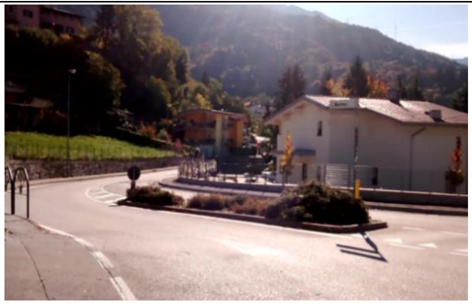
CSVM

Detected Objects: 'bins', 'office'
Missed objects: Ø

ResNet

Detected Objects: ‘Laboratories’, 'bins', 'office'
Missed objects: Ø

Figure 5-9: Example of classification results with CSVM and ResNet on Dataset2. In red are highlighted false positives.



Existing objects: 'building', 'green ground', 'stair(s)', 'walk path', 'Fence/Wall', 'Trees', 'pole/ Pillar', 'traffic sign'

CSVM

Detected Objects: 'building', 'green ground', 'vehicle/bus', 'stair(s)', 'walk path', 'Fence/Wall', 'Trees', 'pole/ Pillar', 'traffic sign'
Missed objects: Ø

VDCNs

Detected Objects: 'people', 'building', 'green ground', 'vehicle/bus', 'stair(s)', 'walk path', 'Fence/Wall', 'Trees', 'pole/ Pillar',
Missed objects: 'traffic sign'

Figure 5-10: Example of classification results with CSVM and VDCNs on Dataset3. In red are highlighted false positives.

Besides the classification accuracies, another important performance parameter is the runtime. Table 5.5 which shows the consumed time for training the proposed method on the 3 datasets. It can be seen clearly that the training of the CSVM is so fast and need just few seconds to few minutes in the worst case. In details, Dataset 1 presents the highest runtime (76 seconds) which is due to the high number of filters used for this dataset (512). While training of Dataset 3 is so fast with just 8 seconds which is due to the simple network architecture that this dataset requires to get the best performance (see Table 5.1).

Table 5.5: Training time of the proposed CSVM.

Dataset	Runtime (s)
Dataset1	76
Dataset2	42
Dataset3	8

Regarding runtime of the prediction step, which includes the feature extraction and the prediction, the CSVM method presents different runtime for the 3 datasets depending on the complexity of the adopted architecture. For instance, and as we can see on Table 5.6, Table 5.7 and Table 5.8, the highest runtime is with the first Dataset 1 with around 200 millisecond (ms) per image which is due to the high number of filter adopted for it (512). Contrary to the third dataset which requires only 2ms to extract features and estimate the classes for each image. This short time is due to the small number of kernels and their small size for the architecture adopted for this dataset. It is also important to mention that the runtime provided by our method outperforms the 3 pretrained CNNs on 2 datasets (Dataset 2 and 3), especially for the dataset 3 where the difference is so significant. Only for Dataset 1 GooLeNet is slightly faster. This is mainly due to the larger size of the first layer of CSVM (512), which typically consumes most of the processing time as applied on the original image. Moreover, in GoogLeNet, the image is smaller as it needs to be resized to 224×224.

Table 5.6: Comparison of average runtime per image on Dataset 1.

Method	Runtime (ms)
ResNet	207
GoogLeNet	**141**
VDCNs	291
CSVM	206

Table 5.7: Comparison of average runtime per image on Dataset 2.

Method	Runtime (ms)
ResNet	208
GoogLeNet	144
VDCNs	291
CSVM	**115**

Table 5.8: Comparison of average runtime per image on Dataset 3.

Method	Runtime (ms)
ResNet	207
GoogLeNet	145
VDCNs	300
CSVM	**2**

5.4. Conclusion

In this chapter, we have presented a novel method for multilabel classification, which has the following important properties: 1) it estimates the weights of CNNs based on SVMs learning formulation; 2) it uses a forward supervised learning strategy for computing the weights of the filters; i) the experimental results obtained on three datasets with limited training samples confirm the promising capability of the proposed method with respect to state-of-the-art methods based on pretrained CNNs.

5.5. References

[1] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, “Greedy layer-wise training of deep networks,” in *Proc. Neural Inf. Process. Syst.*, Cambridge, MA, USA, 2007, pp. 153–160.

[2] Hinton, G. E, Osindero, S., and Teh, Y. W. “A fast learning algorithm for deep belief nets.” *Neural Computation*, 18:1527-1554, 2006.

[3] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. “Gradient-based learning applied to document recognition.” *Proceedings of the IEEE*, 86(11), 1998, pp. 2278–2324.

[4] Mariolis, I. ; Peleka, G. ; Kargakos, A. ; Malassiotis, S. , Pose and category recognition of highly deformable objects using deep learning. *International Conference on Advanced Robotics (ICAR)*, pp. 655 - 662, 27-31 July 2015.

[5] K. He, X. Zhang, S. Ren, and J. Sun, “Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 37, no. 9, pp. 1904–1916, 2015.

[6] K. He, X. Zhang, S. Ren, et J. Sun, « Deep Residual Learning for Image Recognition », in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, p. 770-778.

[7] C. Szegedy *et al.*, « Going deeper with convolutions », in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, p. 1-9.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” in Advances in Neural Information Processing Systems, 2012, pp. 1097–1105.

[9] S. Ren, K. He, R. Girshick, X. Zhang, and J. Sun, “Object Detection Networks on Convolutional Feature Maps,” vol. 8828, no. c, pp. 1–9, 2016.

[10] R. Girshick, J. Donahue, S. Member, T. Darrell, and J. Malik, “Region-Based Convolutional Networks for Accurate Object Detection and Segmentation,” vol. 38, no. 1, pp. 142–158, 2016.

[11] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN : Towards Real-Time Object Detection with Region Proposal Networks,” vol. 8828, no. c, pp. 1–14, 2016.

[12] W. Ouyang et al., “DeepID-Net : Object Detection with Deformable Part Based Convolutional Neural Networks,” Pattern Anal. Mach. Intell. IEEE Trans., vol. 39, no. 7, pp. 1320–1334, 2017.

[13] C. Couprie, L. Najman, and Y. Lecun, “for Scene Labeling,” Pattern Anal. Mach. Intell. IEEE Trans., vol. 35, no. 8, pp. 1915–1929, 2013.

[14] E. Shelhamer, J. Long, and T. Darrell, “Fully Convolutional Networks for Semantic Segmentation,” Pattern Anal. Mach. Intell. IEEE Trans., vol. 39, no. 4, pp. 640–651, 2017.

[15] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout : A Simple Way to Prevent Neural Networks from Overfitting,” J. Mach. Learn. Res., vol. 15, pp. 1929–1958, 2014.

[16] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” Int. Conf. Learn. Represent., pp. 1–14, 2015.

[17] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, “Return of the Devil in the Details: Delving Deep into Convolutional Nets,” Proc. Conf. BMVC, May 2014.

[18] H. Azizpour, A. S. Razavian, J. Sullivan, A. Maki, and S. Carlsson, “Factors of Transferability for a Generic ConvNet Representation,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 38, no. 9, pp. 1790–1802, 2016.

[19] R. F. Nogueira, R. de Alencar Lotufo, and R. C. Machado, “Fingerprint Liveness Detection using Convolutional Networks,” Ieee Trans. Inf. Forensics Secur., vol. 11, no. 6, pp. 1206–1213, 2016.

[20] C. Gao, P. Li, Y. Zhang, J. Liu, and L. Wang, “People counting based on head detection combining Adaboost and CNN in crowded surveillance environment,” Neurocomputing, vol. 208, pp. 1–9, 2016.

[21] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, “LIBLINEAR: A Library for Large Linear Classification,” J. Mach. Learn. Res., vol. 9, no. 2008, pp. 1871–1874, 2008.

[22] K. Chang and C. Lin, “Coordinate Descent Method for Large-scale L2-loss Linear Support Vector Machines,” vol. 9, pp. 1369–1398, 2008.

Chapter 6

Conclusion

In this thesis, some deep learning methods and their applications have been covered. In particular, interest was given to Convolutional Neural Network (CNN) and Stacked AutoEncoder (SAE). CNN is composed of an ensemble of convolution and subsampling layers and concluded by a prediction in order to discern the class label of the input of the network (generally an image). Whereas a SAE is a series of many AutoEncoders (AEs), each AE is composed of three layers: one visible layer which is the input layer, one hidden layer and one reconstruction layer. Despite the big interest that deep learning methods are getting and their application on many research fields, they are still hardly explored in others. This could be justified by the fact that deep learning methods are all based on a neural network architecture which require huge training data in order to learn such networks. Furthermore, the major objective of deep methods is to extract high level features in order to apply them for classification problems. Departing from this fact, we have developed new deep approaches based on AEs and CNNs in order to deal with problems of i) multilabeling classification based on coarse description, ii) reconstructing a missing area covered by clouds in multispectral images, and iii) regression for chemometric data analysis. In the following, we give highlights emphasizing the four (04) proposed deep methods. For further details, we direct the reader to the respective chapters.

In Chapter 2, we provide a presentation about the scene description methodology using a multilabeling strategy. The objective is to assist visually impaired people to conceive a more accurate perception about their surrounding objects in indoor spaces. This method exploits feature learning concept by means of an AE neural network, which amply demonstrated a significant potential in generating discriminative image representations. The key-determinant of our image multilabeling scheme is that the number of objects is independent of the classification system, which entails the property of detecting as many objects as desired (depending on the offline setup to be customized by the user) within the same amount of time which amounts for much less than a second in our work and makes it possible to be applied in real time.

In Chapter 3, we present our proposed approach to reconstruct a missing area in multispectral images due to the presence of clouds based on an AE neural network. Given a cloud-free image (source image) and a cloud-contaminated image (target image), the standard architecture of the AE is slightly modified in order to be able to estimate the mapping function between the source and the target images. For this purpose, two strategies were developed. The first relies on simple pixel-based information to calculate the transformation function whereas the second strategy performs a patch-to-patch mapping followed by a simple fusion step to reconstruct the single pixels of the missing areas in the target image. Moreover, in order to fix the problem of the hidden layer size, a new solution combining the minimum descriptive length (MDL) criterion and a Pareto-like selection method has been introduced. The experimental results reveal that the two proposed methods show good results in reconstructing the missing areas and can significantly outperform state-of-the-arts methods. Compared to the pixel-based strategy, the patch-based one yields better accuracies thanks to the feeding of spatial contextual, and thus richer information, in the reconstruction model.

In Chapter 4, we describe the developed methods for chemometric data analysis based on CNN. In particular, we modify the standard CNN architecture to adapt it to 1D input data. The proposed 1D-CNN architecture is thus based on an alternation of convolutional layers with subsampling layers and connected at the end to a linear regression layer. The convolution is applied using 1D filters and pooling is performed by averaging the samples over a given sliding 1D window. The estimation of the architecture weights is performed using two methods. The first one consists of the standard back-propagation algorithm. The second approach is based on a layerwise particle swarm optimization. GPR and SVR are used in the regression step. The experimental results show that results yielded by the proposed approach are very promising. Indeed, on the three considered datasets, the extracted features can provide significant gains in accuracy, suggesting that CNNs are able to extract powerful features for regression on 1D signals. Moreover, the contribution of the PSO in the training of the neural network architecture appears valuable and with a very limited additional computational overload.

In Chapter 5, we introduce the proposed new method to train the CNNs. This method is based on using Support Vector Machine (SVM) in order to estimate the parameters of the network. The architecture of the developed network is similar to the standard CNN i.e., composed of a succession of convolution and subsampling layers and each layer is composed of a number of maps. The novelty of the proposed method is to estimate the weights of the kernels by means of SVM. The advantage of this method is that the training is applied in just one pass and does not require a big training dataset. The experimental results show that results yielded by the new developed method (CSVM) are very promising in term of accuracies and time consumed in both training and prediction phases.

Finally, in order to improve results and accuracies of the different proposed methods, we suggest the following hints for future developments:

- While the objective of the coarse description is to roughly list the present objects as to bridge the gap between the real indoor setup and the image conceived in the visually disabled person’s imagination, inferring further information pertaining to the detected object location in the indoor space remains a vivid endeavour in our future considerations. To complement this missing component, we suggest to find a way to introduce the depth information (e.g., through Kinect sensors for instance) as a post-operation. Another issue is related to the scalability of the system since it will need to be completely retrained in case the set of predefined objects requires to be modified quantitatively or qualitatively. As last suggestion, it is worth mentioning that the thresholding step for final decision could be made more sophisticated as proposed in [1].

- Regarding the reconstructing of missing area in multispectral images due to presence of clouds and in order to improve the accuracy of the reconstruction process, different aspects of the methods deserve to be investigated in future research studies. We limited our AE neural network to a shallow architecture (just one hidden layer). Stacked and thus deep AE could be a way to make the reconstruction even more accurate. However, this will raise the problem of the best architecture to be set (number of hidden layers and number of hidden nodes per layer) as well as of the computational complexity which will unavoidably increase. Another aspect regards the fact that our reconstruction is based on a one image-to-one image mapping. It would be also interesting to probe ways to integrate further the temporal dimension in the reconstruction by reasoning at the level of time series images (and not just on couples of images).

- Concerning the 1D-CNN, it may be interesting to explore other kinds of deep neural networks such as SAEs and train them with evolutionary methods, given the limitation of the number of training samples. Moreover, we can apply the 1D-CNNs for other applications where input data are under the form of 1D signals (e.g., prediction issues for ECG signals and solar energy plants). Another research direction regards the possibility to use the same model for different instruments. It is noteworthy that the model obtained in this work (and in most of those based on machine learning) is valid only for a given instrument, namely the one on which it was trained. However, moving from an instrument to the other would not necessarily require to change completely the model but to resort to the so-called domain adaption approach, in order to adapt the original model trained in a source domain (instrument 1) to a target domain (instrument 2) with hopefully little effort (by analyzing the data distributions from a domain to the other). Another potential issue is a higher risk of overfitting, requiring some modifications in the cost function in order to mitigate such risk. This could be the subject of another future research direction. Finally, we believe that it would be useful to tailor the 1D-CNN to other remote sensing applications, such us precision farming, traffic monitoring and prediction of natural disasters, among others.

- Regarding the CSVM, it can be interesting to test our approach to other datasets. In our case we focused in multilabeling real time problems. But it can be also used for classical classification of single objects. Also, it will be more interesting to use nonlinear SVM to sophisticate the CSVM network for applications where computational time is not a primordial factor.

[1] A. Zeggada, F. Melgani, and Y. Bazi, “A Deep Learning Approach to UAV Image Multilabeling,” *IEEE Geosci. Remote Sens. Lett.*, vol. 14, pp. 694–698, 2017.

<u>List of Publications achieved during the PhD activity</u>

Journal Papers:

[J1] **S. Malek**, F. Melgani, Y. Bazi and N. Alajlan, "Reconstructing Cloud-Contaminated Multispectral Images with Autoencoder Neural Networks", *IEEE Trans. Geosci. Remote Sens*. Vol. PP, no. 99, pp. 1-13, Dec. 2017.

[J2] **S. Malek**, F. Melgani, and Y. Bazi, "One-dimensional Convolutional Neural Networks for Spectroscopic Signal Regression", *Journal of Chemometrics*. Nov. 2017. e2977.

[J3] **S. Malek**, F. Melgani, M. L. Mekhalfi, Y. Bazi and N. Alajlan, "Indoor Scene Description for the Visually Impaired with Autoencoder Fusion Strategies", *Sensors*. vol. 17, no. 11, Nov. 2017.

Conference Proceedings:

[C1] **S. Malek**, and F. Melgani "Autoencoding approach to the cloud removal problem", *IEEE Int. Geosc. Remote Sens. Symp (IGARSS 2017)*, pp. 4848-4850, 2017.