

EXPLORANDO O APRENDIZADO DE  
CARACTERÍSTICAS ESPACIAIS PARA  
SENSORIAMENTO REMOTO



KEILLER NOGUEIRA

EXPLORANDO O APRENDIZADO DE  
CARACTERÍSTICAS ESPACIAIS PARA  
SENSORIAMENTO REMOTO

Tese apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Doutor em Ciência da Computação.

ORIENTADOR: JEFERSSON ALEX DOS SANTOS  
COORIENTADOR: WILLIAM ROBSON SCHWARTZ

Belo Horizonte

Maio de 2019



KEILLER NOGUEIRA

GOING DEEP INTO REMOTE SENSING SPATIAL  
FEATURE LEARNING

Thesis presented to the Graduate Program  
in Ciência da Computação of the Univer-  
sidade Federal de Minas Gerais in partial  
fulfillment of the requirements for the de-  
gree of Doctor in Ciência da Computação.

ADVISOR: JEFERSSON ALEX DOS SANTOS  
CO-ADVISOR: WILLIAM ROBSON SCHWARTZ

Belo Horizonte

May 2019

© 2019, Keiller Nogueira.  
Todos os direitos reservados.

Nogueira, Keiller

N778g      Going Deep into Remote Sensing Spatial Feature  
Learning / Keiller Nogueira — Belo Horizonte, 2019  
xxix, 155 p.: il.; 29cm.

Tese (doutorado) — Universidade Federal de Minas  
Gerais – Departamento de Ciência da Computação.

Orientador: Jefersson Alex dos Santos

Coorientador: William Robson Schwartz

1. Computação - Teses. 2. Aprendizado do  
computador. 3. Classificação de imagens.  
4. Segmentação de Imagens. I. Orientador.  
II. Coorientador. III. Título.

CDU 519.6\*82.10(043)



UNIVERSIDADE FEDERAL DE MINAS GERAIS  
INSTITUTO DE CIÊNCIAS EXATAS  
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

## FOLHA DE APROVAÇÃO

Going Deep into Remote Sensing Spatial Feature Learning

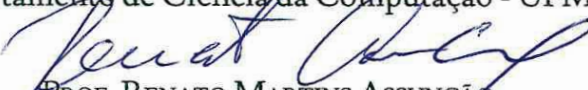
**KEILLER NOGUEIRA**

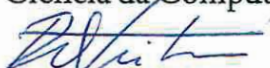
Tese defendida e aprovada pela banca examinadora constituída pelos Senhores:


  
PROF. JEFERSSON ALEX DOS SANTOS - Orientador  
Departamento de Ciência da Computação - UFMG

  
PROF. WILLIAM ROBSON SCHWARTZ - Coorientador  
Departamento de Ciência da Computação - UFMG

  
PROF. PEDRO OLMO STANCIOLI VAZ DE MELO  
Departamento de Ciência da Computação - UFMG

  
PROF. RENATO MARTINS ASSUNÇÃO  
Departamento de Ciência da Computação - UFMG

  
PROF. RAUL QUEIROZ FEITOSA  
Departamento de Engenharia Elétrica - PUC-RJ

  
PROF. HEMERSON PISTORI  
Engenharia de Computação - UCDB

Belo Horizonte, 31 de Maio de 2019.





*Aos meus pais,  
Vander (in memoriam)  
e Maria do Carmo.*



# Acknowledgments

Agradeço...

À toda minha família, principalmente minha mãe, que me apoiou durante o caminho.

À Cristiane, minha namorada, por estar ao meu lado em todos os momentos deste aprendizado. Mesmo à distância, você me auxiliou bastante, às vezes, sem perceber.

Aos meus orientadores, professores Jefersson dos Santos e William Schwartz, pela confiança e ensinamentos. Aos pesquisadores franceses que me receberam durante meu período de sanduíche, professores Jocelyn Chanussot e Mauro Dalla Mura.

Aos muitos companheiros que contribuíram para realização deste trabalho e que fizeram os momentos fora do laboratório mais agradáveis além de ajudar com novas ideias.

À todos aqueles que direta ou indiretamente contribuíram para o desenvolvimento deste trabalho.

Às agências de fomento que financiaram este projeto.



# Resumo

Informações podem ser extraídas da superfície terrestre através de imagens adquiridas por sensores aéreos. Essa informação pode auxiliar tarefas de diferentes aplicações, como urbanização, manejo de culturas e florestas, etc. Embora muito usadas, é complexo minerar as informações sobre essas imagens, já que o primeiro e mais importante passo em direção a tal objetivo é baseado em codificar as características das imagens, uma tarefa desafiadora dadas as propriedades distintas de cada imagem. De fato, independentemente da tarefa, codificar as características espaciais de forma eficiente e robusta é fundamental para gerar bons modelos discriminativos. Mesmo que muitos descritores tenham sido propostos e usados com sucesso para codificar características espaciais de imagens de sensoriamento remoto, a maioria das aplicações exigem técnicas mais específicas. Aprendizado profundo é capaz de aprender, ao mesmo tempo, características e classificadores específicos e ajustando-os em tempo de processamento atendendo às necessidades de cada problema. Neste projeto, abordamos duas principais tarefas, a classificação de cenas e de pixels, usando aprendizado profundo para codificar características espaciais em imagens de sensoriamento remoto.

Inicialmente, enfrentamos o problema de classificação de cenas, em que o desempenho final do modelo é dependente da qualidade das características extraídas. Com base em obras recentemente publicadas e após um conjunto de experimentos, a primeira contribuição foi uma arquitetura de rede convolucional, treinada especificamente para sensoriamento remoto. Experimentos mostraram que a arquitetura proposta superou algoritmos de última geração, demonstrando a eficácia de métodos de aprendizado profundo para codificar características, mesmo para o domínio de sensoriamento remoto. Em seguida, numa tentativa de amenizar as adversidades encontradas ao explorar redes convolucionais, foram avaliadas diferentes estratégias para melhor aproveitar os benefícios dessas técnicas no domínio de sensoriamento remoto. Três estratégias específicas foram avaliadas em três conjuntos distintos de dados. Experimentos mostraram que treinar a rede a partir de um ponto pré-treinado e ajustar seus filtros para o domínio específico é a melhor estratégia para explorar a rede convolucional.

A segunda tarefa abordada é a classificação de pixels. Considerando tal tarefa, a primeira contribuição é uma nova arquitetura da rede convolucional para realizar a classificação de pixels. Essa técnica é baseada em janelas de contexto, ou seja, pequenas imagens que carregam o contexto dos pixels centralizados e ajudam a entender os padrões espaciais em torno deles. Experimentos mostraram que a abordagem proposta proporciona melhorias significativas quando comparada aos outros métodos. Com base nos problemas da técnica anterior e dos métodos existentes, propusemos uma nova abordagem que emprega uma versão evoluída da rede proposta anteriormente para executar a classificação de todos os pixels da imagem de entrada. Esta nova técnica é capaz de agregar informação multicontexto sem aumentar o número de parâmetros. Os resultados mostraram que o algoritmo proposto fornece melhorias na classificação de pixel quando comparado aos métodos de estado da arte.

Finalmente, a última contribuição é um novo método que, quando comparado a outras abordagens existentes, é capaz de melhor aprender algumas características visuais, como bordas e cantos, úteis para algumas aplicações e domínios, incluindo o sensoriamento remoto. Este método, chamado Rede Morfológica Profunda (DeepMorphNet), é capaz de realizar e otimizar operações morfológicas, que podem ser capazes de lidar melhor com algumas propriedades visuais dos objetos. Essa DeepMorphNet pode ser treinada e otimizada de ponta a ponta, utilizando técnicas tradicionais existentes, comumente empregadas no treinamento de abordagens de aprendizagem profunda. Uma avaliação sistemática do algoritmo proposto foi conduzida. Resultados mostraram que a DeepMorphNet é uma técnica promissora capaz de aprender características distintas quando comparadas àquelas aprendidas pelos atuais métodos de aprendizagem profunda.

**Palavras-chave:** Aprendizado Profundo, Redes Neurais, Morfologia Matemática, Classificação de Imagens, Classificação de pixels, Segmentação Semântica, Sensoriamento Remoto, Aprendizado de Máquina.

# Abstract

A lot of information may be extracted from the Earth's surface through images acquired by airborne sensors. Even more, nowadays, enhanced information may be extracted from high spatial resolution images obtained from new sensor technologies. This information may be exploited in several tasks (including, classification and segmentation) assisting in different applications, such as urban planning, disaster relief, phenological studies, etc. Although widely used, the process of distilling this information is strongly based on efficiently encoding features, a challenging task given the distinct properties of the images. In fact, independently of the task, encoding the spatial features in an efficient and robust fashion is the key to generating good discriminative models. Even though many descriptors have been proposed and successfully used to encode spatial features of remote sensing images, most applications demand specific description techniques. Deep Learning, an emergent machine learning approach based on neural networks, is capable of learning specific features and classifiers at the same time and adjust at each step, in processing time, to better fit the need of each problem. For several tasks, it has achieved very good results, mainly boosted by the feature learning performed which allows the method to extract specific and adaptable features depending on the data. In this project, we tackled two main tasks, image and pixel classification, using Deep Learning to encode spatial features over high-resolution remote sensing images.

Initially, we faced the scene classification problem, in which the final model performance is highly dependent on the quality of extracted features. Based on recently published works and after a systematic set of experiments, our first contribution was a novel Convolutional Network architecture fully trained specifically for the remote sensing domain. Experiments showed that the proposed architecture outperformed state-of-the-art algorithms, demonstrating the effectiveness of deep learning methods to encode features even for remote sensing domain. Then, based on adversities to explore deep learning-based techniques (such as small data), we proposed to evaluate different strategies to better exploit Convolutional Networks in the remote sensing do-

main. Specifically, three strategies were evaluated in three very distinctive datasets. Experiments demonstrate that fine-tuning a network into the specific domain is the best strategy to exploit Convolutional Networks.

A second task tackled by this project is the pixel classification (also known as semantic segmentation). This task is of great interest to the remote sensing community since it involves the automatic creation of thematic maps from labeled samples. Such maps may help to understand events over a specific region, such as new urban areas, deforestation, etc. Considering such task, our first contribution is a new Convolutional Network architecture to perform pixel classification of remote sensing images. This technique is based on context windows, i.e., overlapping patches that carry the context of the centered pixels and help to understand the spatial patterns around them. Experiments showed that the proposed approach provides significant improvements when compared to traditional and state-of-the-art methods. Then, based on the problems of the previous technique and of other existing methods, we proposed a novel approach that employs an evolved version of the previously proposed network (as well as other architectures) to perform pixel labeling of the entire input image (instead of classifying only the centered pixel). Moreover, this new technique is able to aggregate multi-context information without increasing the number of parameters, i.e., without requiring the combination of several networks or layers. Results showed that the proposed algorithm provides improvements in pixel classification accuracy when compared to state-of-the-art methods.

Finally, our last contribution is a novel method that, when compared to other existing approaches, is capable of better learning some visual characteristics (such as edges and corners), which are very useful for some applications and domains, including the remote sensing one. Precisely, this method, called Deep Morphological Network (DeepMorphNet), is able to perform and optimize non-linear morphological operations, that may be able to better cope with some interesting visual properties of the objects. These DeepMorphNets can be trained and optimized end-to-end using traditional existing techniques commonly employed in the training of deep learning approaches and can be used in several applications, such as image and pixel classification, detection, and so on. A systematic evaluation of the proposed algorithm was conducted. Experimental results showed that the DeepMorphNets is a promising technique that can learn distinct features when compared to the ones learned by current deep learning methods.

**Keywords:** Deep Learning, Neural Networks, Mathematical Morphology, Image Classification, Pixel Classification, Semantic Segmentation, Remote Sensing, Machine Learning.



# List of Figures

3.1	Some steps of a $3 \times 3$ window of a convolutional layer extracting the features from an image. Figure adapted from [Ng et al., 2011a]. . . . .	23
3.2	Example of dilated convolutions. Dilation supports expansion of the receptive field without loss of resolution or coverage of the input. The blue dot represents the pixel that is being considered as reference during the process.	25
3.3	Comparison between dilated and standard convolutions. Top (red) row presents the feature extraction process using a standard convolution over a downsampled image and then an upsample in order to recover the input resolution (a common procedure performed in ConvNets). Bottom (blue) row presents the feature extraction process using dilated convolution with rate $r = 2$ applied directly to the input (without downsample). The outcomes clearly show the benefits of dilated convolutions over standard ones. . . . .	26
3.4	A pooling layer selecting the max value between the features inside a window of size $2 \times 2$ . Figure adapted from [Ng et al., 2011b]. . . . .	27
3.5	Examples of common structuring elements employed in the literature. These SEs can be seen as sets that define the activated pixels. . . . .	30
3.6	Examples of morphological images generated for the UCMerced Land-use Dataset. All these images were processed using a $5 \times 5$ square as structuring element. . . . .	31
4.1	Examples of the UCMerced Land Use Dataset. . . . .	44
4.2	Examples of the WHU-RS19 Dataset. . . . .	45
4.3	Examples of coffee and non-coffee samples in the Brazilian Coffee Scenes dataset. The similarity among samples of opposite classes is notorious. The intraclass variance is also perceptible. . . . .	46
4.4	The Coffee Dataset. Multispectral images and ground-truths. Legend – White: Coffee areas. Black: Non Coffee areas. . . . .	46

4.5	The GRSS Data Fusion Dataset. Training and test data and their respective ground-truths. Legend – Black: unclassified. Light purple: road. Light green: trees. Red: red roof. Cyan: gray roof. Dark purple: concrete roof. Dark green: vegetation. Yellow: bare soil. . . . .	46
4.6	Examples of the Vaihingen Dataset. Legend – White: impervious surfaces. Blue: buildings. Cyan: low vegetation. Green: trees. Yellow: cars. Red: clutter, background. . . . .	47
4.7	Examples of the Potsdam Dataset. Legend – White: impervious surfaces. Blue: buildings. Cyan: low vegetation. Green: trees. Yellow: cars. Red: clutter, background. . . . .	47
5.1	Illustrative example of a <a href="#">ConvNet</a> being fully trained. Weights from the whole network are randomly initialized and then trained for the target dataset.	52
5.2	Illustrative example of two options for the fine-tuning process. In one of them (highlighted in red), all layers are fine-tuned according to the target dataset, but final layers have increased learning rates. In the other option (highlighted in green), weights of initial layers can be frozen and only final layers are tuned. . . . .	54
5.3	Illustrative example of the use of a <a href="#">ConvNet</a> as feature extractor. The final classification layer is ignored and one should only choose from which layer to consider the features. The figure shows the use of the features from the last layer before the classification layer, which is commonly used in the literature.	54
5.4	Architectures of different the <a href="#">ConvNets</a> evaluated in this work. Purple boxes indicate the layers from where features were extracted in the case of using the <a href="#">ConvNets</a> as feature extractors. . . . .	59
5.5	Average accuracy of pre-trained <a href="#">ConvNets</a> used as feature extractors and low- and mid-level descriptors for the UCMerced Land-use Dataset. . . . .	64
5.6	Average accuracy of pre-trained <a href="#">ConvNets</a> used as feature extractors and low- and mid-level descriptors for the RS19 Dataset. . . . .	65
5.7	Average accuracy of pre-trained <a href="#">ConvNets</a> used as feature extractor and low- and mid-level descriptors for the Brazilian Coffee Scenes Dataset. . . . .	65
5.8	Average accuracy considering all possible strategies to exploit <a href="#">ConvNets</a> for the UCMerced Land-use Dataset. . . . .	66
5.9	Average accuracy considering all possible strategies to exploit <a href="#">ConvNets</a> for the RS19 Dataset. . . . .	67
5.10	Average accuracy considering all possible strategies to exploit <a href="#">ConvNets</a> for the Brazilian Coffee Scenes Dataset. . . . .	67

5.11	Examples of convergence of AlexNet for all datasets, considering Fold 1. . .	68
5.12	Three examples of wrong predictions of each aerial dataset, UCMerced and RS19, for a fine-tuned AlexNet. (a)-(f) The first image is the misclassified one, while the second is a sample of the predicted class. Notice the similarity between the classes. . . . .	70
5.13	Comparison between state-of-the-art baselines and the best results of each strategy to exploit ConvNets for the UCMerced Land-use Dataset. The Fine-Tuned Descriptors extracted by GoogLeNet achieved the highest accuracy rates. . . . .	71
5.14	Comparison between state-of-the-art baselines and the best results of each strategy to exploit ConvNets for the RS19 Dataset. The Fine-Tuned Descriptors extracted by GoogLeNet achieved the highest accuracy rates. . .	71
5.15	Comparison between state-of-the-art baselines and the best results of each strategy to exploit ConvNets for the Brazilian Coffee Scenes Dataset. The Fine-Tuned Descriptors extracted by CaffeNet achieved the highest accuracy rates. . . . .	72
6.1	(a) Example of a context window. The pattern is represented by a large window that is centered on the pixel of interest in order to include the context of its neighborhood. (b) The process performed by a pixelwise network. A set of context windows are generated for each pixel and then classified by the network. The predicted class for the context window is actually the label of the centered pixel. . . . .	76
6.2	Example showing the importance of multi-context information. In the top case, while smaller contexts may not provide enough information for the understanding of the scene, a large context brings more information that may help the model to identify that it is a road with a car on it. In the bottom scenario, smaller contexts bring enough information for the identification of cars, while a large context may confuse the network and lead it to misclassify a different object as a car. . . . .	78
6.3	Dilated Convolutional Network architectures. . . . .	82
6.4	Convergence of Dilated6 network for all datasets. For the Coffee Dataset, only the fold 1 is reported. For Vaihingen and Potsdam Datasets, the validation set (created according [Volpi and Tuia, 2017]) is reported. . . . .	90
6.5	Images of the Coffee Dataset, their respective ground-truths, and the relevance maps generated by the proposed algorithm. Legend – White: True Positive. Black: True Negative. Red: False Positive. Green: False Negative. . . . .	94

6.6	The GRSS Data Fusion test image, the respective ground-truth, and the prediction maps generated by the proposed algorithm. Legend – Black: unclassified. Light purple: road. Light green: trees. Red: red roof. Cyan: gray roof. Dark purple: concrete roof. Dark green: vegetation. Yellow: bare soil. . . . .	95
6.7	Example predictions for the validation set of the Vaihingen Dataset. Legend – White: impervious surfaces. Blue: buildings. Cyan: low vegetation. Green: trees. Yellow: cars. Red: clutter, background. . . . .	98
6.8	Example predictions for the test set of the Vaihingen Dataset. Legend – White: impervious surfaces. Blue: buildings. Cyan: low vegetation. Green: trees. Yellow: cars. Red: clutter, background. . . . .	99
6.9	Comparison, in terms of overall accuracy and number of trainable parameters, between proposed and existing networks for Vaihingen and Potsdam Datasets. Ideal architectures should be in the top left corner, with fewer parameters but higher accuracy. Since the x axis is logarithmic, a change of only 0.3 in this axis is equivalent to more than 1 million new parameters in the model. . . . .	100
6.10	Example predictions for the validation set of the Potsdam Dataset. Legend – White: impervious surfaces. Blue: buildings. Cyan: low vegetation. Green: trees. Yellow: cars. Red: clutter, background. . . . .	101
6.11	Example predictions for the test set of the Potsdam Dataset. Legend – White: impervious surfaces. Blue: buildings. Cyan: low vegetation. Green: trees. Yellow: cars. Red: clutter, background. . . . .	102
7.1	Illustration showing an input image (representing what can be seen as an x-shaped border), some ConvNet filters and the produced outputs generated by these filters. Note that none of the network outcomes is similar to the desired output, which was generated using a morphological erosion with a $3 \times 3$ filter/structuring element, equal to the first one employed by the ConvNet. This shows that regardless of the filter, the Convolutional Network is not able to produce an output as desired due to its fully linear operations. . . . .	104

7.2	<p>Example of a morphological erosion based on the proposed framework. The 4 filters <math>W</math> (with size <math>4 \times 4</math>) actually represent a unique <math>4 \times 4</math> SE. Each filter <math>W</math> is first converted to binary <math>W^b</math>, and then used to process each input channel (step 1, blue dashed rectangle). The output is then processed via a pixel and depthwise min-pooling to produce the final eroded output (step 2, green dotted rectangle). Note that the binary filters <math>W^b</math>, when superimposed, retrieve the final SE <math>B</math>. The dotted line shows that the processing of the input with the superimposed SE <math>B</math> using the standard morphological erosion results in the same eroded output image produced by the proposed morphological erosion. . . . .</p>	111
7.3	<p>Definition of morphological neurons based on the proposed framework. In this case, just one type of each neuron is represented. Some neurons have tied weights to force the network to use the same filters (i.e. SE) in both operations. Other neurons have skip connections in order to allow an new operation using the original input and the processed data. . . . .</p>	112
7.4	<p>Concept of a morphological layer. Note that a single morphological layer can have neurons performing different operations. This process is able to aggregate heterogeneous and complementary information. . . . .</p>	115
7.5	<p>The three Deep Morphological Network (<b>DeepMorphNet</b>) architectures proposed and exploited in this work. Note that the number of morphological neurons of each type in each layer is codified using the Equations of Section 7.1.2. Also, observe that the two depthwise convolutions of a same layer share the kernel size, sometimes differing only in the stride and padding. Moreover, the padding and stride of each layer are presented as follows: the value related to the first depthwise convolution is reported separated by a comma of the value related to the second depthwise convolution. Though not visually represented, the pointwise convolutions explored in the morphological layers always use the same configuration: kernel <math>1 \times 1</math>, stride 1, and no padding. . . . .</p>	120
7.6	<p>Learned structuring elements for the synthetic datasets. In both cases, the learned SEs are exactly as expected, i.e., a square SE larger than <math>5 \times 5</math> but smaller than <math>9 \times 9</math> for the square synthetic dataset, and a rectangle larger than the rectangles of one class and in the same orientation for the rectangle synthetic dataset. . . . .</p>	124

7.7	Examples of the output of the opening neuron for the square synthetic dataset. The first column represents the input image, the second one is the output of the erosion, and the last one is the output of the dilation. Since erosion and dilation have tied weights (i.e., the same structuring element), they implement an opening. . . . .	124
7.8	Examples of the output of the opening neuron for the synthetic rectangle dataset. The first column represents the input image, the second one is the output of the erosion, and the last one is the output of the dilation. Since erosion and dilation have tied weights (i.e., the same structuring element), they implement an opening. . . . .	125
7.9	Convergence of the proposed morphological networks and the baselines for both datasets. Note that only fold 1 is reported. . . . .	127
7.10	Input images and some produced (upsampled) feature maps extracted from all layers of the AlexNet-based networks for the UCMerced Land-use Dataset. For each input image: the first row presents features from the AlexNet-based network, the second row presents the feature maps learned by the Depth-AlexNet-based architecture, and the last row presents the features of the proposed morphological network. . . . .	128
7.11	Input images and some produced (upsampled) feature maps extracted from all layers of the AlexNet-based networks for the WHU-RS19 Dataset. For each input image: the first row presents features from the AlexNet-based network, the second row presents the feature maps learned by the Depth-AlexNet-based architecture, and the last row presents the features of the proposed morphological network. . . . .	130

# List of Tables

4.1	Overview statistics for the three classification datasets employed in this work.	36
4.2	Overview statistics for the two pixel classification datasets employed in this work. . . . .	37
4.3	Number of pixels per class for the GRSS Data Fusion Dataset. . . . .	38
4.4	Number of pixels per class for ISPRS datasets, i.e., Vaihingen and Potsdam.	39
4.5	Confusion matrix in which each $x_{ij}$ represents the number of scenes or pixels (depending on the task) in the classified (observed) image category $i$ and the ground truth (reference) cover category $j$ . Adapted from [Liu et al., 2007].	41
5.1	Some statistics about the deep networks evaluated in this work. . . . .	58
5.2	Parameters utilized in fine-tuning and full-training strategies. . . . .	63
6.1	Hyperparameters employed in each dataset. . . . .	85
6.2	Results over different distributions. . . . .	86
6.3	Results over different score functions. . . . .	87
6.4	Results of the proposed approach when varying the input range of patch sizes. For Vaihingen, a validation set (created according [Volpi and Tuia, 2017]) is employed. Bold patch size ranges were selected for all further experiments. . . . .	89
6.5	Results of the Dilated6 network trained using distinct convolution types. .	90
6.6	Comparison between the dilated network trained using the proposed and the traditional method. . . . .	92
6.7	Results for the Coffee dataset. . . . .	93
6.8	Results for the GRSS Data Fusion Dataset. . . . .	95
6.9	Official results for the Vaihingen Dataset. . . . .	97
6.10	Official results for the Potsdam Dataset. . . . .	99
7.1	Results, in terms of average accuracy, of the proposed method and the baselines for the square synthetic dataset (Section 7.2.1.1). . . . .	123

7.2	Results, in terms of average accuracy, of the proposed method and the baselines for the synthetic rectangle dataset (Section 7.2.1.1). . . . .	124
7.3	Results, in terms of accuracy, of the proposed method and the baselines for the UCMerced Land-use Dataset. . . . .	126
7.4	Results, in terms of average accuracy, of the proposed <a href="#">DeepMorphNets</a> and baselines for the WHU-RS19 dataset. . . . .	129



# Abbreviations

**ACC** *Auto-Correlogram Color*

**BIC** *Border/Interior Pixel Classification*

**BoVW** *Bag of Visual Words*

**ConvNet** *Convolutional Network*

**CCNN** *Cascade Convolutional Neural Network*

**CNN** *Convolutional Network*

**DeepMorphNet** *Deep Morphological Network*

**FCN** *Fully Convolutional Network*

**HOG** *Histogram of Oriented Gradients*

**LAS** *Local Activity Spectrum*

**LCH** *Local Color Histogram*

**RSI** *Remote Sensing Image*

**SASI** *Statistical Analysis of Structural Information*

**SE** *Structuring Element*

**SIFT** *Scale-Invariant Feature Transform*



# Contents

Acknowledgments	xi
Resumo	xiii
Abstract	xv
List of Figures	xvii
List of Tables	xxiii
Abbreviations	xxv
<b>1 Introduction</b>	<b>1</b>
1.1 Research challenges . . . . .	4
1.2 Hypothesis, objectives, and contributions . . . . .	8
1.3 Organization of the text . . . . .	10
<b>2 Related Work</b>	<b>11</b>
2.1 Scene Classification . . . . .	11
2.2 Pixel Classification . . . . .	14
2.3 Morphological-Based Feature Extraction . . . . .	18
<b>3 Background Concepts</b>	<b>21</b>
3.1 Convolutional Networks . . . . .	21
3.1.1 Processing Units . . . . .	22
3.1.2 Layers . . . . .	23
3.1.3 Training . . . . .	28
3.2 Mathematical Morphology . . . . .	29
<b>4 General Experimental Setup</b>	<b>35</b>

4.1	Datasets . . . . .	35
4.1.1	Scene Classification Datasets . . . . .	35
4.1.2	Pixel Classification Datasets . . . . .	37
4.2	Protocols . . . . .	40
4.3	Measures . . . . .	41
<b>5</b>	<b>ConvNet-Based Scene Classification</b>	<b>49</b>
5.1	Strategies for Exploiting ConvNets . . . . .	51
5.1.1	Fully-trained Network . . . . .	51
5.1.2	Fine-tuned Network . . . . .	52
5.1.3	ConvNet as a Feature Extractor . . . . .	53
5.2	Specific Experimental Setup . . . . .	55
5.2.1	Classical Feature Extraction Strategies . . . . .	55
5.2.2	ConvNets . . . . .	58
5.2.3	Protocol . . . . .	62
5.3	Results and Discussion . . . . .	63
5.3.1	Generalization Power Evaluation . . . . .	63
5.3.2	Comparison of ConvNets Strategies . . . . .	66
5.3.3	Comparison with Baselines . . . . .	70
5.4	Conclusions . . . . .	72
<b>6</b>	<b>ConvNet-Based Pixel Classification</b>	<b>75</b>
6.1	Pixel Classification Approach . . . . .	79
6.1.1	Dynamic Multi-Context Algorithm . . . . .	79
6.1.2	Architectures . . . . .	81
6.2	Specific Experimental Setup . . . . .	83
6.2.1	Baselines . . . . .	83
6.2.2	Protocol . . . . .	84
6.3	Results and Discussion . . . . .	85
6.3.1	Patch Distribution Analysis . . . . .	85
6.3.2	Score Function Analysis . . . . .	86
6.3.3	Range Analysis . . . . .	87
6.3.4	Convolution Operation Analysis . . . . .	88
6.3.5	Convergence Analysis . . . . .	90
6.3.6	Performance Analysis . . . . .	91
6.3.7	State-of-the-art Comparison . . . . .	93
6.4	Conclusions . . . . .	98

<b>7</b>	<b>An Introduction to Deep Morphological Networks</b>	<b>103</b>
7.1	Deep Morphological Networks . . . . .	105
7.1.1	Basic Morphological Framework . . . . .	107
7.1.2	Morphological Processing Units . . . . .	110
7.1.3	Morphological Layer . . . . .	114
7.1.4	Optimization . . . . .	116
7.1.5	DeepMorphNet Architecture . . . . .	118
7.2	Experimental Setup . . . . .	119
7.2.1	Specific Datasets . . . . .	120
7.2.2	Baselines . . . . .	121
7.2.3	Experimental Protocol . . . . .	122
7.3	Results and Discussion . . . . .	122
7.3.1	Synthetic Datasets . . . . .	122
7.3.2	Image Classification Datasets . . . . .	125
7.4	Conclusion . . . . .	127
<b>8</b>	<b>Conclusions and Future Work</b>	<b>131</b>
8.1	Future Work . . . . .	132
	<b>Appendix A List of Publications</b>	<b>135</b>
	<b>Bibliography</b>	<b>139</b>



# Chapter 1

## Introduction

Earth's Planet is constantly being modified due to natural and human interference, including hurricanes, earthquakes, new residential and agricultural areas, landfills, etc. It is costly and almost impractical to understand all these changes and developments via on-the-ground observations. Thus, a lot of effort has been employed for obtaining images from the Earth's surface, i.e., aerial ones. Although a laborious task, it can be justified first by the amount of information that may be extracted from these images and second by the potential usage of this data in several tasks (such as classification and segmentation) assisting in the understanding of a myriad of events. Based on this argument, new technologies have been proposed toward acquiring aerial images with improved quality, resulting in more advanced satellites launched to observe the Earth, as well as, more recently, in drones and unmanned aerial vehicles. Nowadays, new sensor technologies give us the capacity of obtaining high spatial resolution images, with bands beyond the visible ones. These top-notch Remote Sensing Images (RSIs) may provide useful information that could be employed in several Earth Observation applications, including urban planning [Tayyebi et al., 2011; Taylor and Lovell, 2012; Volpi and Ferrari, 2015; Volpi and Tuia, 2017], crop and forest management [dos Santos et al., 2012; Nogueira et al., 2015b], disaster relief [Fustes et al., 2014; Nogueira et al., 2018], phenological studies [Nogueira et al., 2016b, 2017a, 2019b], oil spill [Fingas and Brown, 2014], poverty mapping [Xie et al., 2016], etc.

In general, all the information distilled by these applications are highly dependent on the creation of high quality thematic maps (to establish precise inventories about land cover use [Jensen and Lulla, 1987]) as well as on detection and monitoring of events. However, the development of both tasks, by manual efforts (e.g., using edition tools), is slow and costly, being unfeasible, given the large amount of data. Therefore, automatic methods appear as an appealing alternative for the community.

Traditionally, such automatic methods [Huang et al., 2011; Avramović and Risojević, 2014] perform these tasks by using machine-learning based approaches over features encoded by some visual description technique. Therefore, efficiently encoding of these features is one of the most important steps in almost any computer vision problem (as well as in the remote sensing field) since it is the main key to generate discriminative models. Given this, through years, substantial efforts have been dedicated to develop automatic and discriminative feature techniques [Kumar and Bhatia, 2014], commonly called (hand-crafted) descriptors. Some of them [Swain and Ballard, 1991; Tao and Dickinson, 2000; Huang et al., 1997; Lowe, 2004; de O. Stehling et al., 2002] were originally proposed and successfully employed in the computer vision scenario and then, experimented into the remote sensing domain, while others [Benediktsson et al., 2005; Dalla Mura et al., 2010; Yu et al., 2016; Hu et al., 2016] were specifically designed for Earth Observation applications.

In the first case, successful descriptors proposed to handle everyday pictures, i.e., RGB images typically acquired by ordinary cameras and commonly used in the computer vision domain, may not have the same favorable outcome for RSIs given the distinct characteristics between these images, which include:

- (i) Perspective. This comes from the fact that traditional images typically have a concept of depth and, consequently, a clear definition of fore and background. Although this concept somehow exists in aerial scenes, it is not useful for remote sensing applications, being usually handled as noise, which is corrected with a process called orthorectification [Leprince et al., 2007]. This difference introduces nuances when encoding the features, since in aerial images all the pixels should be managed with the same attention independently of its elementary composition (color, texture, etc);
- (ii) Context. Given that everyday pictures have a specific notion of context related to the scene (for instance, a car passing far from the region of interest is not part of the scene) while, in aerial images, there is no such context but there is the geographic context (a car can not appear in middle of the ocean). This fact should be leveraged by descriptors in order to improve its feature representation;
- (iii) Elementary properties. Given that traditional images usually have more complex and rich scenes than aerial ones, mainly when considering low-resolution images. This implies in more information (such as gradient, texture, color, etc) that could be extracted by hand-crafted descriptors in the former images but that may not



be encoded in the latter ones [Jensen and Lulla, 1987; Campbell and Wynne, 2011];

- (iv) Channels (or bands). While traditional images usually encode only visible information, RSIs may have hundreds or even thousands of bands (including non-visible ones). Furthermore, visual description techniques proposed for computer vision images usually rely on a specific color space (RGB, HSI, CieLab, etc) or in a transformation between these spaces, which also prevents its use in the remote sensing domain, which may have images with more bands that do not fit in such color spaces. All of these points should be taken into account to create a robust and efficient feature representation.

Thus, as introduced, based on these specificities and differences, many of these techniques, originally proposed and successfully applied for computer vision applications [Chen et al., 2010], have not the same success in the remote sensing domain [dos Santos et al., 2010].

In the second case, though successfully proposed and employed into the remote sensing domain, each descriptor technique is highly dependent of the intrinsic properties of the image, such as gradient, edges, colors, etc [Jensen and Lulla, 1987; Campbell and Wynne, 2011]. For instance, a novel descriptor proposed specifically for land-use scenes may not be a good choice for agricultural images. Thus, the development of algorithms for spatial extraction information is still a hot research topic in the remote sensing community [Benediktsson et al., 2013; Ma et al., 2015; Ball et al., 2017].

Besides all this, in a typical scenario, different descriptors may produce distinct results depending on the data. Therefore, it is imperative to design and evaluate many descriptor algorithms in order to find the most suitable ones for each application [dos Santos et al., 2014]. This process is also expensive and, likewise, does not guarantee an efficient and discriminative representation.

Overcoming aforementioned limitations, a resurgent technique, renamed deep learning, has become the state-of-the-art solution for pattern recognition. Although deep learning was initially applied to recognition problems in the early 1990's [LeCun et al., 1995, 1998], it has become a very hot research topic since 2012, with [Krizhevsky et al., 2012] winning the ImageNet classification challenge [Deng et al., 2009]. Given its success, deep learning has been intensively used in several distinct tasks of different domains [Goodfellow et al., 2016; Bengio, 2009], including remote sensing [Chen et al., 2014; Firat et al., 2014; Penatti et al., 2015; Xie et al., 2016; Hu et al., 2015; Nogueira et al., 2016b; Lin, 2016; Cheng et al., 2016; Nogueira et al., 2017a; Volpi and Tuia, 2017]. In fact, the use of deep learning techniques in the remote sensing domain is growing

very quickly, since it has a natural ability to effectively encode spectral and spatial information based mainly on the data itself. Methods based on deep learning have obtained state-of-the-art results in many different Earth Observation applications, such as image classification [Chen et al., 2014; Firat et al., 2014; Zhang et al., 2015; Guan et al., 2015; Penatti et al., 2015; Lin, 2016], semantic segmentation [Paisitkriangkrai et al., 2015; Nogueira et al., 2015b; Yue et al., 2015; Paisitkriangkrai et al., 2015; Marmanis et al., 2018; Sherrah, 2016; Chen et al., 2016; Volpi and Tuia, 2017; Nogueira et al., 2019a], poverty mapping [Xie et al., 2016], phenological studies [Nogueira et al., 2016b, 2017a, 2019b], and so on.

Specifically, deep learning [Goodfellow et al., 2016; Bengio, 2009] is a branch of machine learning that refers to multi-layered interconnected neural networks that can learn features and classifiers at once, i.e., a unique network may be able to learn features and classifiers (in different layers) and adjust the parameters, at running time, based on accuracy, giving more importance to one layer than another depending on the problem. End-to-end feature learning (e.g., from image pixels to semantic labels) is the great advantage of deep learning when compared to previously state-of-the-art methods [LeCun et al., 2015], such as mid-level (Bag of Visual Words (BoVW) [Sivic and Zisserman, 2003]) and global low-level color and texture descriptors. Among all deep learning-based networks, a specific type, called Convolutional (Neural) Networks [Goodfellow et al., 2016; Bengio, 2009], *ConvNets* or *CNNs*, is the most popular for learning features in computer vision applications, as well as in remote sensing domain. This sort of network relies on the natural stationary property of an image, i.e., the statistics of one part of the image are the same as any other part and information extracted at one part of the image can also be employed to other parts. Furthermore, *ConvNets* usually obtain different levels of abstraction for the data, ranging from local low-level information in the initial layers (e.g., corners and edges), to more semantic descriptors, mid-level information (e.g., object parts) in intermediate layers and high level information (e.g., whole objects) in the final layers. As stated, overall, this feature learning step allows the network to extract distinct features in different level of semantics in a data-driven process, which is the great advantage of deep learning.

## 1.1 Research challenges

There are several computational research challenges related to pattern recognition applications in the remote sensing domain. In this thesis, the investigated research challenges may be arranged into three main components.

First, hand-crafted descriptors are created for a specific domain and may fail when applied in another one. For instance, descriptors created to handle everyday pictures in the computer vision domain may fail to encode features of aerial images as well as descriptor techniques conceived to deal with urban aerial scenes may encounter problems in handling agricultural images. Therefore, a data-driven feature learning step, as in [ConvNets](#), is essential to extract all feasible information from the data and create discriminative models. However, [ConvNets](#) are hard to train, because of its high number of parameters, requiring a really large amount of annotated data. In fact, [\[Bengio, 2009\]](#) suggests that, to efficiently train a new neural network from scratch, it is necessary, at least, 1,000 annotated images per class. Going in the other direction, remote sensing domain has huge amount of data but with only few annotations. Even using data augmentation techniques [\[Goodfellow et al., 2016; Bengio, 2009\]](#), the total amount of images may be still small to allow the convergence of a deep neural network. This discrepancy creates several challenges, including:

- (i) Usage of deep learning-based techniques in scenarios with very small amount of annotated samples. These comes from the fact that these scenarios, with small sets of annotated images, are very common in Earth Observation applications mainly because of the difficulty to validate and annotate aerial images [\[Tuia et al., 2011; dos Santos et al., 2013; Almeida et al., 2014\]](#), which are highly dependent of specialists (such as biologists, agronomists, etc) that may need to access difficult areas of study (in the middle of the jungle or far away from civilization, for instance). Thus, strategies capable of extracting all feasible information from the few available samples are essential to exploit these techniques in such difficult scenarios.
- (ii) Better exploit the feature representation, learned from deep networks on huge computer vision datasets, into the remote sensing domain. Although remote sensing domain does not have datasets with lots of annotations, the computer vision area has several datasets (composed of thousands, even millions, of pictures [\[Deng et al., 2009\]](#)) that fulfill this requirement being better choices when training deep learning networks. However, as introduced, computer vision and remote sensing are distinct domains with their images having different properties, which creates other challenge. In fact, this challenge comes from the notion that deep learning techniques may converge better with more data (just as in computer vision datasets) [\[Bengio, 2009; Goodfellow et al., 2016\]](#) but, when converging, they learn feature representation based specifically on the input data. These features may not be suitable when used into another domain given that the generalization is

not guaranteed because of the intrinsic difference between the images. Therefore, it is fundamental to investigate the possible generalization of these features and how to exploit them in order to alleviate the problem of small amount of annotated data for the remote sensing scenario.

- (iii) Adjust feature representation learned in huge computer vision datasets for the remote sensing domain. This also comes from the fact that large datasets, such as the ones in the computer vision domain [Deng et al., 2009], are more propitious to train deep learning models. However, the feature representation learned on this data may fail when encoding information in a different domain (such as the remote sensing one), mainly because of the differences between the images. So, it is very important to efficiently understand this gap between different source of data (with distinct elementary properties) in order to adjust the feature representation to fill such breach and improve the final performance of the method.

Second, although a lot of attention has been given to scene classification, one of the most important application in the remote sensing community is the creation of thematic maps, which may help in understanding events over a specific region, such as new urban areas and deforestation. Essentially, this application is modeled in a supervised manner which should have, as outcome, a class for each and every pixel of the input image. Based on its outcome, this task is commonly called pixel classification (also known, in the computer vision field, as semantic segmentation) [Wilkinson, 2005; Nogueira et al., 2016a; Volpi and Tuia, 2017; Nogueira et al., 2019a]. Though important, pixel classification is a hard task given that its basic element (the pixel) has not enough information to allow its classification. This creates several and important research challenges, such as:

- (i) Difficulty of performing pixel classification with only the pixel itself. Traditionally, methods use the value of the pixel in each band as a signature to perform its classification [Wilkinson, 2005], a process that is only possible because the reflectance encoded by the pixels along the spectrum (bands) is usually associated to the physical properties of the surface materials. However, as aforementioned, the pixel (signature) itself may not have enough information hindering such methods. Thus, it might be interesting to explore the context of the pixel in its classification in order to create robust and efficient methods improving the final thematic maps.
- (ii) Efficiently exploit and classify the pixels of the input images. This is because the classification of each pixel independently demands a lot of computational

resources. Therefore, it should be interesting to research an efficient technique that could perform the classification of multiple pixels of the input images (or patches) at once. This strategy should be able to efficiently use the context of the pixels to improve the classification while reducing the time and computational complexity.

- (iii) Better exploitation of the context of the pixels. This is because, first, the size of this context must be somehow defined, whether based on prior knowledge or experimentally. However, this definition is of vital importance, given that small contexts could not bring enough information to allow the understanding of the scene while larger contexts could lead to semantically mixed information. Therefore, it should be extremely useful to research a method that could analyze and define the best context. Second, depending on the image, a unique size of context may not be suitable for all cases. For these scenarios, it is important to study new strategies to better exploit and extract information from varying contexts (multi-context scenario).

Third, as introduced, different types of images have distinct elementary properties, such as borders, corners, colors, etc. These properties are essentially encoded by the *ConvNets* using linear filters, i.e., despite the network employs non-linear functions and pooling layers to bring some non-linearity to the learning process, the filters learned through convolution layers are fundamentally based on linear operations. However, it is known in the literature that non-linear operations are able to cope with some image properties better than the linear ones, being preferable in some applications [Xu et al., 2016; Liu et al., 2017; Wang et al., 2018a]. In fact, supported by this property, some non-linear operations, such as the morphological ones, are still very popular being considered state-of-the-art in some remote sensing scenarios [Dalla Mura et al., 2010; Xia et al., 2015]. This divergence generates several research challenges, including:

- (i) Combine and exploit morphological operations with deep learning. As introduced, non-linear transformations, such as the morphological ones, are able to cope better with some properties of the images, such as borders, corners, and so on. Therefore, it should be extremely important to research for a deep learning-based technique capable of performing morphological operations.
- (ii) Optimize the morphological operations. The network is usually optimized using partial derivatives. These are commonly obtained using the backpropagation algorithm [Goodfellow et al., 2016], which is strongly supported by the fact that all operations of the network are easily differentiable, including the convolution one.

However, this process can not be directly applied to non-linear operations, such as the presented morphological ones, because those operations do not have easy, integrable derivatives. Therefore, it is essential to study and analyze a method capable of performing the feature learning step, by optimizing the structuring elements of the morphological operations, in order to extract all feasible information from the input images and improve its performance.

The work developed in this thesis addresses these important research challenges.

## 1.2 Hypothesis, objectives, and contributions

In this thesis, we focus on the data representation and feature extraction problems with the objective of developing effective solutions for classification and semantic segmentation of remote sensing images. We pursue this objective based on the three validation hypotheses below:

- (i) It is possible to exploit a previously learned representation to improve the classification of small remote sensing scene datasets.
- (ii) Without increasing the complexity of the model, it is possible to aggregate multi-context information and improve the final pixel classification.
- (iii) If optimized, morphological operations are able to cope better with distinct image properties when compared to [ConvNets](#).

The first hypothesis concerns the use of deep learning-based techniques in the remote sensing scene classification task. Fundamentally, it comes from the fact that deep learning techniques are usually trained using a large amount of data but, in the remote sensing domain, commonly, there are only a few annotated labels. The contributions concerning the first hypothesis are related to the proposal of a novel [ConvNet](#) architecture as well as in the analysis of different strategies to exploit [ConvNets](#), both considering the remote sensing scene classification task. In the first, based on recently published works and after a systematic set of experiments, we proposed a novel network architecture fully trained specifically for the remote sensing domain. This novel architecture has fewer layers and parameters, being able to converge using a small quantity of data, demonstrating the effectiveness of deep learning methods to encode features even for remote sensing domain. In the second, we proposed to evaluate different strategies to exploit [ConvNets](#) for the remote sensing domain. Three strategies were evaluated in three different datasets. Experiments demonstrate that fine-tuning a

network (pre-trained on huge computer vision datasets, such as ILSVRC [Deng et al., 2009]) into the specific domain is the best strategy to exploit *ConvNets*. The novel network architecture was published in the Conference on Graphics, Patterns and Images (SIBGRAPI) [Nogueira et al., 2015a]. The analysis of strategies to exploit *ConvNets* was published in the Pattern Recognition journal (PR) [Nogueira et al., 2017c]. All aforementioned contributions are presented in Chapter 5.

The second hypothesis is related to the different possibilities of exploiting the context of the pixel to improve the pixel classification and create enhanced thematic maps, that may help in the understanding of land cover use, agricultural production, etc. Essentially, this comes from the fact that the pixel itself has not enough information to allow its classification, but its context could bring useful knowledge capable of filling this gap of information. However, there are several conditions related to the exploitation of this context, including the size, the dependence of the size of objects in the scene, etc. The contributions related to the second hypothesis are a new *ConvNet* architecture (based on context windows) as well as a technique that exploits a multi-context paradigm while defining adaptively the best context size for the inference stage. In the first, the proposed network aggregates the context of the pixel by using overlapping windows, centered on each pixel, that carry the context of the pixel and help understand the spatial patterns around them. This strategy allows the network to efficiently understand the context around the pixel and correctly classify it. In the second, we proposed a novel technique to perform semantic segmentation of remote sensing images that aggregates information from contexts of multiple sizes (without increasing the number of parameters) while defining the best context size for the testing phase. This multi-context strategy allows the network to capture distinct information of the context of the objects, allowing a better understanding of the scene. Experiments showed that the proposed method is capable of aggregating multi-context information and improve the pixel classification, mainly of small objects. The novel network was published in the International Conference on Pattern Recognition (ICPR) [Nogueira et al., 2016a]. A preliminary version of this network, trained specifically for coffee recognition, was firstly published in the Iberoamerican Congress on Pattern Recognition (CIARP) [Nogueira et al., 2015b]. The multi-context technique was published in the IEEE Transactions on Geoscience and Remote Sensing (TGRS) [Nogueira et al., 2019a]. Furthermore, this multi-context approach was exploited and considered the winner [Nogueira et al., 2018] of the 2017 Multimedia Satellite Contest [Bischke et al., 2017], which was part of the traditional MediaEval Benchmark. Details of these contributions are presented in Chapter 6.

The third hypothesis is related to the exploitation of morphological filters with

deep learning. Specifically, although **ConvNets** use non-linear functions and pooling layers to bring some non-linearity to the learning process, the filters learned through its convolution layers are essentially based on linear operations. However, non-linear (morphological) operations are able to cope with some visual properties better than the linear ones, being preferable in some domains, such as the remote sensing one. Therefore, this hypothesis comes from the fact that a combination of the morphological operations and the feature learning strategy (performed by deep learning-based methods) should produce a method capable of better handle certain types of image properties. The contribution related to the third hypothesis is a novel method for deep feature learning, called Deep Morphological Network (**DeepMorphNet**), which is capable of doing non-linear morphological operations while performing the feature learning step (by optimizing the structuring elements). This new approach, strongly based on the **ConvNets** (because of the similarity between the operation performed by the morphological transformations and convolutional layers), would aggregate the benefits of morphological operations while learning the structuring element during the training process. Experiments demonstrated that the method is able to better preserve some characteristics of the images (such as borders and corners) when compared to the **ConvNets**. Details of this contribution are introduced in Chapter 7.

### 1.3 Organization of the text

This thesis is organized following the hypotheses. It has eight chapters, including this introduction. Chapter 2 presents the works related to this thesis. Chapter 3 introduces some background concepts of the convolutional network, given that this technique was employed throughout all this work. In Chapter 4, we present remote sensing image datasets used in the experiments, including some proposed during this work.

The following three chapters are directly related to the hypotheses and have, aside the proposed technique, also an overview of the works related to the respective task, the experimental protocol used, baselines, obtained results and conclusions. Following this, Chapter 5 presents our study of remote sensing image classification based on **ConvNets**. In Chapter 6, we present our proposed **ConvNet**-based approach to perform pixel classification of remote sensing images. Chapter 7 introduces the proposed network capable of doing non-linear morphological operations while performing the feature learning step (by optimizing the structuring elements).

Finally, Chapter 8 presents our conclusions and future perspectives.



# Chapter 2

## Related Work

In this chapter, we present related work for scene and pixel classification for remote sensing applications. Section 2.1 presents the related work for scene classification. Section 2.2 gives a literature review of works related to remote sensing pixel classification. Finally, Section 2.3 presents works combining morphological operations and deep learning for remote sensing image scene classification.

### 2.1 Scene Classification

Considerable efforts have been dedicated to the development of suitable feature descriptors for remote sensing applications [Kumar and Bhatia, 2014]. Although several of these descriptors have been proposed or successfully used for remote sensing image processing [Yang and Newsam, 2008; dos Santos et al., 2010; Bouchiha and Besbes, 2013], as stated in Chapter 1, there are many applications that demand more specific techniques mainly due to the intrinsic properties of such images, including presence of non-visible bands, different perspective and context, etc. Towards a robust and efficient feature representation, ConvNets involve machine learning in the process of obtaining the best features for a given problem. Despite interesting, deep ConvNets have several drawbacks, such as impossibility to confirm convergence, “black box” nature, high computational cost, proneness to overfitting and empirical nature of model development [Goodfellow et al., 2016; Bengio, 2009]. In an attempt to alleviate these effects, three strategies can be used to better exploit existing ConvNets, which are: (i) full-trained ConvNets, (ii) fine-tuned ConvNets, and (iii) pre-trained ConvNets used as feature extractors. As introduced, the objective of this Chapter is to evaluate these strategies to explore existing deep neural networks. Thus, this section focus on review and analyze existing works that exploit any of these strategies for the remote sensing

scenario.

Train a (new or existing) network from scratch is preferable since it tends to give specific feature representation for the dataset. Also, this strategy gives full control of the architecture and parameters, which tends to yield a more robust and specific network. However, it requires a considerable amount of data [Goodfellow et al., 2016; Bengio, 2009]. During the years, successful ConvNets were the ones fully trained (from scratch) in large amount of data, such as ImageNet dataset [Deng et al., 2009], which has been used to train several famous architectures [Krizhevsky et al., 2012; Simonyan and Zisserman, 2014; Szegedy et al., 2015; He et al., 2016; Huang et al., 2017]. AlexNet, proposed by [Krizhevsky et al., 2012], was the winner of ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [Deng et al., 2009] in 2012. This work is the mainly responsible for the recent popularity of neural networks, since it introduced several important concepts to the field, including non-saturating neurons, dropout [Srivastava et al., 2014], normalization, etc. VGG ConvNets, presented in [Simonyan and Zisserman, 2014], won the ILSVRC-2014 competition (localization and classification) while GoogLeNet, presented in [Szegedy et al., 2015], is the ConvNet architecture that won the classification and detection tracks of the same competition. The former network is strongly based on the AlexNet architecture being a deeper version of it. The latter is based on the new concept of “Inception” module, which is a set of layers trained apart and used to composed the final network architecture (that is consistent of several “Inception” modules). More recently, some architectures were proposed to allow the deepening of the network while avoiding gradient problems, such as the degradation problem [He et al., 2016]. Precisely, the idea behind those approaches is to use skip connections [Goodfellow et al., 2016], that allow the gradients to flow more directly without much interference. [He et al., 2016] proposed the Residual Networks (ResNets) which make improvements on the final representation by combining (via skip connections) the input of a layer with the feature maps learned by this layer. Such networks have won the ILSVRC-2015 competition (classification track) and are one of the most advanced techniques in terms of ConvNets. [Huang et al., 2017] proposed the Densely Connected Networks (DenseNets), an improved version of the ResNet architectures. The DenseNets allow a layer to have access to the feature maps learned from all previous layers. Those maps are usually concatenated (via skip connections) and provided to that layer in order to generate enhanced outcomes and avoid gradient problems.

Although huge annotated remote sensing data are unusual, there are many works, usually employing reasonable datasets (more than 2,000 images), that achieved promising results by proposing the full training of new ConvNets [Makantasis et al., 2015; Yu et al., 2017; Cheng et al., 2018]. [Makantasis et al., 2015] classified hyperspectral images

through a fully-trained new ConvNet with only two convolution layers achieving state-of-the-art in four datasets. In [Yu et al., 2017], the authors used different techniques to perform data augmentation in order to improve the feature representation learned by ConvNets. They fully trained a network in three different datasets and showed that distinctive data augmentation techniques may really improve representation. [Cheng et al., 2018] proposed a ConvNet-based technique that is fully trained using a metric learning concept in order to perform remote sensing scene classification. Aside this, a myriad of deep learning-based methods (including Restricted Boltzmann Machines, Autoencoders, Generative Adversarial Networks, etc) were fully-trained for the remote sensing domain [Chen et al., 2014; Firat et al., 2014; Zhang et al., 2015; Guan et al., 2015; Lin, 2016; Lin et al., 2017; Xu et al., 2018].

As introduced, in general, ConvNets have a peculiar property: they all tend to learn first-layer features that resemble either Gabor filters, edge detectors or color blobs. Supported by this characteristic, another strategy to exploit ConvNets is to perform fine-tuning of its parameters using the new data. Several works [Cheng et al., 2016; Li et al., 2017; Zhang et al., 2019], in the remote sensing community, exploit the benefits of fine-tuning pre-trained ConvNets. [Cheng et al., 2016] fine-tuned three existing networks [Krizhevsky et al., 2012; Simonyan and Zisserman, 2014; Szegedy et al., 2015] to perform remote sensing scene classification. After the tuning process, they extracted features of each one and used machine learning techniques to perform the final classification yielding state-of-the-art in the UCMerced Land-use Dataset. In [Li et al., 2017], the authors propose a two-stage neural network to perform remote sensing scene classification. The first stage is composed of a fine-tuned AlexNet [Krizhevsky et al., 2012], which results in features that are used as input for a Restricted Boltzmann Machine network (the second stage), that is responsible to learn a better feature representation and classify the scenes. [Zhang et al., 2019] combined and fine-tuned several pre-trained ConvNets with Capsule Networks [Sabour et al., 2017]. Their final architecture is able to encode the properties and spatial information of features in an image to achieve equivariance.

Based on aforementioned characteristics, ConvNets can also be exploited as a feature extractor. Specifically, these features, usually called deep features, are obtained by removing the last classification layer and considering the output of previous layer (or layers). In some recent studies [Penatti et al., 2015; Hu et al., 2015; Cheng et al., 2016, 2017; Li et al., 2019], ConvNets have shown to perform well even in datasets with different characteristics from the ones they were trained with, without performing any adjustment, using them as feature extractors only and using the features according to the application (e.g., classification, retrieval, etc). In remote sensing

domains, [Penatti et al., 2015] evaluated the use of different **ConvNets** as feature extractors, achieving state-of-the-art results in two remote sensing datasets, outperforming well-known descriptors. [Hu et al., 2015] combined features extracted from several pre-trained **ConvNets** to perform classification of high-resolution remote sensing imagery. [Cheng et al., 2016] extracted features from pre-trained **ConvNets** and use them to create a machine learning model capable of perform classification of land-use scene images. In [Cheng et al., 2017], the authors used features extracted from pre-trained **ConvNets** to create a **BoVW** and, then, classify the remote sensing images. [Li et al., 2019] combined pre-trained **ConvNets** with Attention Maps in order to produce a representation focused on the salient parts of the scene.

Concerning the evaluation of different practices to exploit deep neural networks, [Jarrett et al., 2009] analyzed the best architecture and training protocol to explore deep neural networks, including unsupervised and supervised feature learning as well as supervised and fine-tuned classification of the instances. Also, [Larochelle et al., 2009] studied the best way to train neural networks, including greedy layer-wise and fine-tuning strategies. Finally, [Donahue et al., 2014] evaluated the generalization power of deep features extracted from hidden layers of pre-trained **ConvNets**, but they did not investigate any other strategy of exploiting **ConvNets**, such as fine-tuning or fully-training. Specifically, they analyze if features extracted from pre-trained deep convolutional network can be repurposed to novel generic tasks, which may differ significantly from the originally trained tasks and do not have sufficient labeled data to conventionally train or adapt a deep architecture to the new tasks.

Contributions related to scene classification (and presented in Chapter 5) distinguish from the literature, given that, while some aforementioned works just exploited distinct training strategies (such as full-training, fine-tuning, and feature extractor), we evaluate these practices. Furthermore, we investigate multiple **ConvNets** (six) and datasets (three), differing from other works that evaluate only one or a few **ConvNets** and datasets.

## 2.2 Pixel Classification

As introduced, deep learning has made its way into the remote sensing community, mainly due to its success in several computer vision tasks [Krizhevsky et al., 2012; Long et al., 2015; Zhang et al., 2015, 2016; Du et al., 2017]. Towards a better understanding of the Earth’s surface, a myriad of techniques [Paisitkriangkrai et al., 2015; Nogueira et al., 2015b; Marcu and Leordeanu, 2016; Audebert et al., 2016; Maggiori et al., 2017;

Sherrah, 2016; Marmanis et al., 2018; Paisitkriangkrai et al., 2016; Wang et al., 2017; Peng et al., 2019; Nogueira et al., 2019b] have been proposed to perform semantic segmentation of RSIs.

In order to aggregate context information about the pixels (helping in their classification) and to avoid computational problems (possibly caused by the huge size of the RSIs), all of these methods process the input images using overlapping patches (that should be superimposed to retrieve the final thematic map). Some of them [Paisitkriangkrai et al., 2015; Nogueira et al., 2015b; Paisitkriangkrai et al., 2016; Nogueira et al., 2019b] consider those patches as context windows (i.e., overlapping images, in which each one is centered on a specific pixel helping to understand the spatial patterns around that pixel) and perform the classification of each pixel independently. Other works [Marcu and Leordeanu, 2016; Audebert et al., 2016; Sherrah, 2016; Maggiori et al., 2017; Wang et al., 2017; Marmanis et al., 2018; Liu et al., 2018; Peng et al., 2019] classify the whole input patch producing a dense outcome with all pixels classified. Although somehow successful, the definition of the best input patch size is of vital importance for the methods, given that small patches could not bring enough information to allow the understanding of the scene while larger contexts could lead to semantically mixed information. Towards a better exploitation of the context and in an attempt to alleviate such dependence of the patch size, several works [Nogueira et al., 2015b; Audebert et al., 2016; Sherrah, 2016; Marcu and Leordeanu, 2016; Paisitkriangkrai et al., 2016; Maggiori et al., 2017; Wang et al., 2017; Marmanis et al., 2018; Peng et al., 2019] have proposed to aggregate multi-context information to their learning process. This procedure allows the methods to extract and capture patterns of varying granularities, helping the method to aggregate more useful information.

In [Nogueira et al., 2015b], the authors proposed two multi-context techniques based on ConvNet to perform semantic segmentation. The first method is based on cascading of networks while the second is established in an iterative process of ConvNet. They achieve remarkable results in agricultural datasets composed of coffee and non-coffee crops. In [Audebert et al., 2016], the authors fine-tuned a deconvolutional network (based on SegNet [Badrinarayanan et al., 2017]) using  $256 \times 256$  fixed size patches. To incorporate multi-context knowledge into the learning process, they proposed a multi-kernel technique at the last convolutional layer. Specifically, the last layer is decomposed into three branches. Each branch processes the same feature maps but using distinct filter sizes generating different outputs which are combined into the final dense prediction. They argue that these different scales smooth the final predictions due to the combination of distinct fields of view and spatial context.

Sherrah [Sherrah, 2016] proposed methods based on fully convolutional net-

works [Long et al., 2015]. The first architecture was purely based on the fully convolutional paradigm, i.e., the network has several downsampling layers (generating a coarse map) and a final bilinear interpolation layer, which is responsible to restore the coarse map into a dense prediction. In the second strategy, the previous network was adapted by replacing the downsampling layers with dilated convolutions, allowing the network to maintain the full resolution of the image. Finally, the last strategy evaluated by the authors was to fine-tune pre-trained networks over the remote sensing datasets. None of the aforementioned strategies exploit the benefits of the multi-context paradigm. Furthermore, these techniques were evaluated using several input patch sizes with final architectures processing patches with  $128 \times 128$  or  $256 \times 256$  pixels depending on the dataset.

[Marcu and Leordeanu, 2016] combined the outputs of a dual-stream network in order to aggregate multi-context information for semantic segmentation. Specifically, each network processes the image using patches of distinct size, i.e., one network process  $256 \times 256$  patches (in which the global context is considered) while the other processes  $64 \times 64$  patches (where local context is taken into account). The outputs of these architectures are combined in a later stage using another network. Although they can train the network jointly, in an end-to-end process, the number of parameters is really huge allowing them to use only small values of batch size (10 patches per batch). In [Paisitkriangkrai et al., 2016], the authors proposed a multi-context semantic segmentation by combining ConvNets, hand-crafted descriptors, and Conditional Random Fields [Lafferty et al., 2001; Krähenbühl and Koltun, 2011]. Specifically, they trained three ConvNets, each one with a different patch size ( $16 \times 16$ ,  $32 \times 32$  and  $64 \times 64$  pixels). Features extracted from these networks are combined with hand-crafted ones and classified using random forest classifier. Finally, Conditional Random Fields [Lafferty et al., 2001; Krähenbühl and Koltun, 2011] are used as a post-processing method in an attempt to improve the final results.

[Maggiori et al., 2017] proposed a multi-context method that performs labeling segmentation based on upsampled and concatenated features extracted from distinct layers of a fully convolutional network [Long et al., 2015]. Specifically, the network, that receives as input patches of  $256 \times 256$  or  $512 \times 512$  pixels (depending on the dataset), is composed of several convolutional and pooling layers, which downsample the input image. Downsampled feature maps extracted from several layers are, then, upsampled, concatenated and finally classified by another convolutional layer. This proposed strategy resembles somehow the DenseNets [Huang et al., 2017], with the final layer having connections to the previous ones. [Wang et al., 2017] proposed to extract features from distinct layers of the network to capture multi-context low- and high-level

information. They fine-tuned a ResNet-101 [He et al., 2016] to extract salient features from  $600 \times 600$  patches. Feature maps are then extracted from intermediate layers, combined with entropy maps, and upsampled to generate the final dense prediction.

In [Marmanis et al., 2018], the authors proposed multi-context methods that combine boundary detection with deconvolution networks (specifically, based on SegNet [Badrinarayanan et al., 2017]). The main contribution of this work is the Class-Boundary (CB) network, which is responsible to help the proposed methods to give more attention to the boundaries. Based on this CB network, they proposed several methods. The first uses three networks that receive as input the same image but with different resolutions (as well as the output of the corresponding CB network) and output the label predictions, which are aggregated, in a subsequent fusion stage, generating the final label. They also experimented fully convolutional architectures [Long et al., 2015] (with several skip layers in order to aggregate multi-context information) and an ensemble of several architectures. All aforementioned networks initially receive  $256 \times 256$  fixed size patches. In [Peng et al., 2019], the authors combined the concepts of skip and dense connections [He et al., 2016; Huang et al., 2017]. Precisely, they proposed a basic block composed of several convolutional layers with skip connections between them. Then, this block is stacked to create a deep architecture that has dense connections [Huang et al., 2017] between the blocks. In order to aggregate multi-context information, this network processes the input image using three branches, each one using different kernel sizes.

Contributions related to pixel classification (and presented in Chapter 6) distinguish from the literature in several points. First, as presented, although some works have already exploited fully convolutional networks, the proposed models explore such paradigm to the maximum by never downsampling the input data (a common process performed in the literature works [Long et al., 2015; Yu and Koltun, 2015; Marmanis et al., 2018]). Second, our proposed technique is capable of exploiting the multi-context strategy without any modification of the network or combination of several architectures (or layers), a very common approach to exploit multi-context information in the literature. Finally, our proposed approach is also able to determine the best patch size adaptively (in training time), instead of evaluating possible patch sizes (to find the best one) or to use a patch size determined by network constraints (which could not be the best one), common strategies adopted in the literature.

## 2.3 Morphological-Based Feature Extraction

As introduced, non-linear operations, such as the morphological ones [Serra and Soille, 2012], have the ability to preserve borders and corners, which may be essential properties in some scenarios, such as the remote sensing one. Supported by such advantages, since their introduction [Pesaresi and Benediktsson, 2001], morphological operations have been extensively applied to several Earth Observation applications [Xia et al., 2015; Mirzapour and Ghassemian, 2015; Gu et al., 2016; Aptoula et al., 2016; Xu et al., 2016; Liu et al., 2017; Wang et al., 2018a].

Some of these techniques [Xia et al., 2015; Mirzapour and Ghassemian, 2015; Gu et al., 2016; Liu et al., 2017] use the advantages of morphology to extract robust features that are employed as input to standard machine learning techniques [Bishop, 2006] (such as Support Vector Machines (SVM), decision trees, and extreme learning machine) in order to perform image classification and segmentation. [Xia et al., 2015] proposed a method based on random subspace ensembles and advanced morphological operations. The final image classification is performed using several standard machine learning techniques, including decision trees and extreme learning machine [Bishop, 2006]. They achieved state-of-the-art results in two hyperspectral datasets composed of more than a hundred bands. [Mirzapour and Ghassemian, 2015] combined texture, shape and spectral features in order to reduce overall classification error. Particularly, they used morphological, gabor and grey-level co-occurrence matrices (GLCM) [Haralick et al., 1973] features and achieved state-of-the-art in three hyperspectral datasets. [Gu et al., 2016] proposed a framework based on multiple kernel learning based on spatial and spectral features extracted by morphological operations. They achieved remarkable results, when compared with several state-of-the-art algorithms, in three hyperspectral datasets. In [Xu et al., 2016], the authors described the images using morphological components based on distinct textural features, that are classified using multinomial logistic regression. Results indicated that their proposed approach can lead to very good classification performances in two different types of data (hyperspectral and Synthetic Aperture Radar). [Liu et al., 2017] proposed to perform different morphological operations using structuring elements of varying size to extract distinct features. These are segmented using the Simple Linear Iterative Clustering (SLIC) method [Achanta et al., 2012] and then classified using an SVM.

More recently, ConvNets [Goodfellow et al., 2016] started achieving outstanding results, mainly in applications related to images. Therefore, it would be more than natural for researchers to propose works combining the benefits of ConvNets and morphological operations. In fact, several works [Masci et al., 2013; Mellouli et al., 2017;



[Aptoula et al., 2016; Wang et al., 2018a] tried to combine these techniques in order to create a more robust model. Some works [Aptoula et al., 2016; Wang et al., 2018a] employed morphological operations as a pre-processing step in order to extract the first set of discriminative features. Specifically, in [Aptoula et al., 2016], the authors used attribute profiles [Dalla Mura et al., 2010] to extract discriminative features that are used as input for a *ConvNet*. This network classifies each pixel based on the profiles creating a thematic map for hyperspectral images. [Wang et al., 2018a] employed several morphological operations as pre-processing to generate more features, which are then used as input for a *ConvNet* responsible to perform the classification. In both works, the structuring elements of the morphological operations are not learned and pre-defined structuring elements are employed. Based on the fact that morphology generates interesting features that are not captured by the convolutional networks, such works achieved outstanding results on pixel classification.

Other works [Masci et al., 2013; Mellouli et al., 2017] combined morphological operations and neural networks, creating a framework in which the structuring elements are optimized. [Masci et al., 2013] proposed a convolutional network that aggregates morphological operations, such as pseudo-erosion, pseudo-dilation, pseudo-opening, pseudo-closing, and pseudo-top-hats. Specifically, their proposed network uses the counter-harmonic mean [Bullen, 2013], which allows the convolutional layer to perform its traditional linear process, or approximations of morphological operations. They show that the approach produces outcomes very similar to traditional (not approximate) morphological operations. [Mellouli et al., 2017] performed a more extensive validation of the previous method, proposing different deeper networks that are used to perform image classification. In their experiments, the proposed network achieved promising results for two datasets of digit recognition.

Contributions related to our method that combines morphological operations and deep learning are presented in Chapter 7. Such contributions distinguish from the literature in two main points: (i) differently from [Masci et al., 2013; Mellouli et al., 2017], the proposed technique really carries out morphological operations without any approximation (except for the geodesic reconstruction), and (ii) to the best of our knowledge, this is the first approach to implement (approximate) morphological geodesic reconstruction within deep learning-based models.



# Chapter 3

## Background Concepts

This chapter presents essential techniques that we employed in this work. Specifically, Section 3.1 presents the fundamental concepts of Convolutional Networks, including processing units, layers, and training. Then, Section 3.2 reviews the basics of mathematical morphology.

### 3.1 Convolutional Networks

This section formally presents some background concepts of Convolutional (Neural) Networks [Goodfellow et al., 2016; Bengio, 2009], or simply *ConvNets* or *CNNs*, a specific type of deep learning method. These networks are generally presented as systems of interconnected processing units (neurons) which can compute values from inputs leading to an output that may be used on further units. These neurons work in agreement to solve a specific problem, learning by example, i.e., a network is created for a specific application, such as pattern recognition or data classification, through a supervised learning process. As introduced, *ConvNets* were initially proposed to work over images, since they try to take leverage from the natural stationary property of an image, i.e., information extracted in one part of the image can also be applied to another region. Furthermore, *ConvNets* present several other advantages: (i) they automatically learn local feature extractors, (ii) they are invariant to small translations and distortions in the input pattern, and (iii) they implement the principle of weight sharing which drastically reduces the number of free parameters and increases their generalization capacity. Next, we present some concept employed in *ConvNets*.

### 3.1.1 Processing Units

Although deep learning-based techniques have been developed through the years with different models emerging, the basic component of every model remains the same. In other words, all networks are basically composed of artificial neurons, which try to simulate the biological ones in a limited way. Formally, artificial neurons are basically processing units that compute some operation over several input variables and, usually, have one output calculated through the activation function. Mathematically, an artificial neuron has a weight vector  $W = (w_1, w_2, \dots, w_n)$ , some input variables  $y = (y_1, y_2, \dots, y_n)$ , a threshold or bias  $b$  and an activation function  $f(\cdot)$ . These four main components are processed as stated in Equation 3.1. Making an analogy, the weights represent the strength of the biological dendritic connections while bias is considered the value that must be surpassed by the inputs before an artificial neuron becomes active (i.e., greater than zero).

$$z = f \left( \sum_i^n x_i \times W_i + b \right) \quad (3.1)$$

where  $z$ ,  $y$ ,  $w$ ,  $b$ , and  $n$  represent output, input, weights, bias, and total size, respectively.  $f(\cdot) : \mathfrak{R} \rightarrow \mathfrak{R}$  denotes an activation function.

Conventionally, a non-linear function is provided in  $f(\cdot)$ . There are a lot of alternatives for  $f(\cdot)$ , such as sigmoid, hyperbolic, and rectified linear function. The latter function is currently the most used in the literature. Neurons with this configuration have several advantages when compared to others [Nair and Hinton, 2010; Goodfellow et al., 2016]: (i) they work better to avoid saturation during the learning process, (ii) they induce the sparsity in the hidden units, and (iii) they do not face the gradient vanishing problem<sup>1</sup> as with sigmoid and tanh function. The processing unit that uses the rectifier as activation function is called Rectified Linear Unit (ReLU) [Nair and Hinton, 2010]. The first step of the activation function of a ReLU is presented in Equation 3.1 while the second one is introduced in Equation 3.2.

$$a = \begin{cases} z, & \text{if } z > 0 \\ 0, & \text{otherwise} \end{cases} \quad \Leftrightarrow \quad a = f(z) = \max(0, z) \quad (3.2)$$

---

<sup>1</sup>The gradient vanishing problem occurs when the propagated errors become too small and the gradient calculated for the back propagation step vanishes, making it impossible to update the weights of the layers and to achieve a good solution.

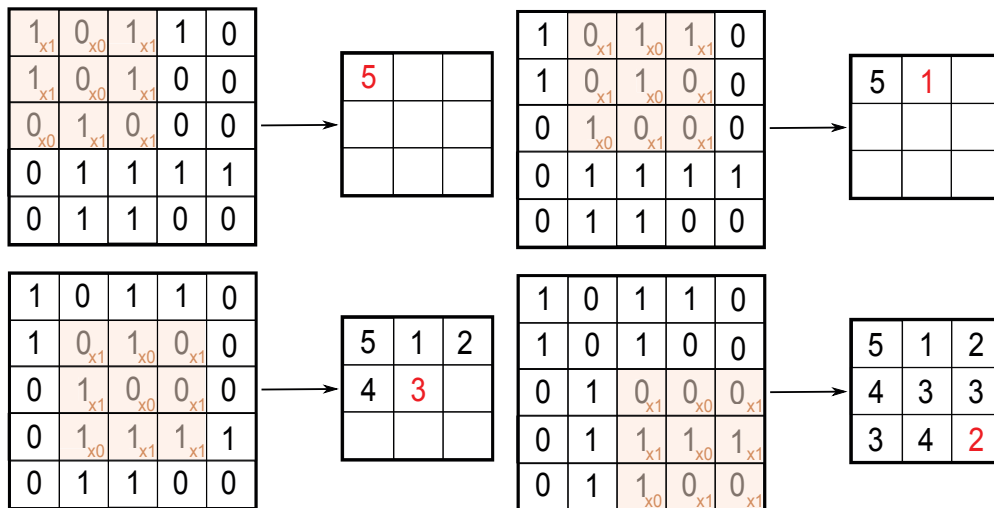
### 3.1.2 Layers

The aforementioned processing units are grouped forming a component commonly called layer. There are several types of layers, such as convolutional and variations, fully-connected, etc. Some of these layers, basis of the [ConvNets](#) and extensively exploited in this work, are presented next.

#### 3.1.2.1 Convolutional Layer

Among the different types of layers, the convolutional one is the responsible for capturing the features from the images. In a deep network, the first convolutional layers usually extract low-level characteristics (like edges, and lines) while the others capture high-level features (such as structures, objects and shapes). The process of feature extraction performed by this type of layer is strongly based on the convolution process. Formally, a 2-D convolution procedure, commonly employed in this type of layer, receives a two-dimension input  $y$  and a 2-D weight vector  $W$  (in this case, with size  $n \times n$ ) and processes them according to (the linear) Equation 3.3.

$$z[k, l] = \sum_{i=1}^n \sum_{j=1}^n y(k+i-1, l+j-1)W(i, j) \quad (3.3)$$



**Figure 3.1:** Some steps of a  $3 \times 3$  window of a convolutional layer extracting the features from an image. Figure adapted from [Ng et al., 2011a].

Intuitively, this process can be decomposed into two phases: (i) the moving step, where a fixed-size window (in this case, the kernel) runs over the image (with some stride<sup>2</sup>) defining a region of interest, and (ii) the processing step, that multiplies the

<sup>2</sup>Stride is the distance between the centers of each window considering two steps.

pixel values inside that region with the filter weights of the neurons, finally extracting the features from that region and producing the output. As presented in Figure 3.1, the process performed by this layer results in a new image (commonly called feature map), generally smaller than the original one, with the visual features extracted.

### 3.1.2.2 Dilated Convolutional Layer

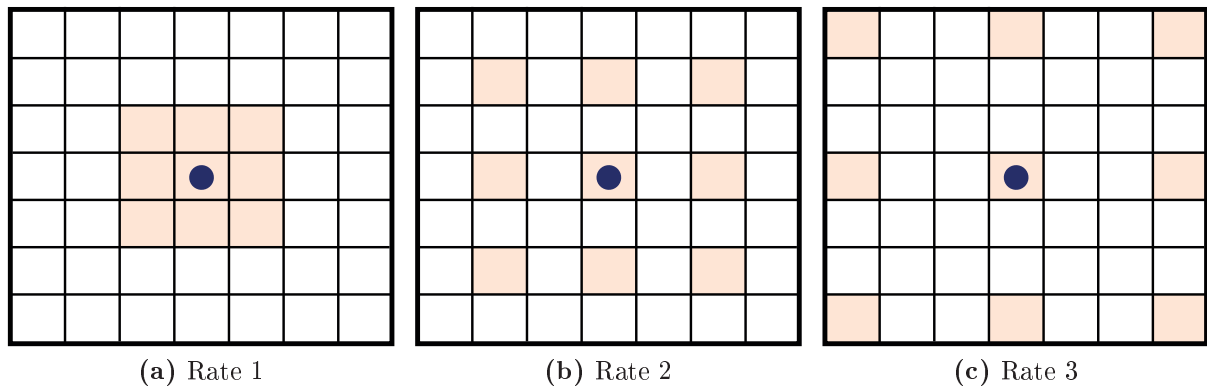
Although the previous layer has been extensively employed in ConvNet architectures, several convolutional variant layers have been proposed trying to improve somehow the process performed by standard layer. An important variant layer is called dilated convolution [Yu and Koltun, 2015] and has been successfully applied to several applications [Yu and Koltun, 2015; Chen et al., 2018; Sherrah, 2016; Wang et al., 2018b; Hamaguchi et al., 2018] and was also exploited in this work. In this type of layers, filter weights are employed differently when compared to standard convolutions. Specifically, filters of this layer do not have to be contiguous and may have gaps (or “holes”) between the parameters. These gaps, inserted according to the dilation rate  $r \in \mathbb{N}$ , enlarge the convolutional kernel but preserve the number of trainable parameters since the inserted holes are not considered in the convolution process. Therefore, this dilation rate  $r$  can be seen as a parameter responsible to define the final alignment of the kernel weights, i.e., it corresponds to the stride with which the input signal is sampled.

Formally a 2-D dilated convolution receives a two-dimension input  $y$ , a dilation rate  $r$ , and a 2-D weight vector  $W$  (in this case, with size  $n \times n$ ) and processes them according to Equation 3.4. Note the differences between the dilated and the standard convolution (presented in Equation 3.3).

$$z[k, l] = \sum_{i=1}^n \sum_{j=1}^n y(k + r \times i, l + r \times j) W(i, j) \quad (3.4)$$

The effect of different dilation rate  $r$  are presented in Figure 3.2. As can be seen smaller rates result in a more clustered filter (in fact, rate 1 generates a kernel identical to the standard convolution) while larger rates make an expansion of the filter, producing a larger kernel with several gaps. Since this whole filter dilation process is independent of the input data, changing the dilation rate does not impact in the resolution of the outcome, i.e., in a dilated convolution, independent of the rate, input and output have the same resolution (considering appropriate stride and padding).

By enlarging the filter (with such gaps), the network expands its receptive field (since the weights will be arranged in a more sparse shape) but preserves the resolution and no downsampling in the data is performed. Hence, this process has several advan-



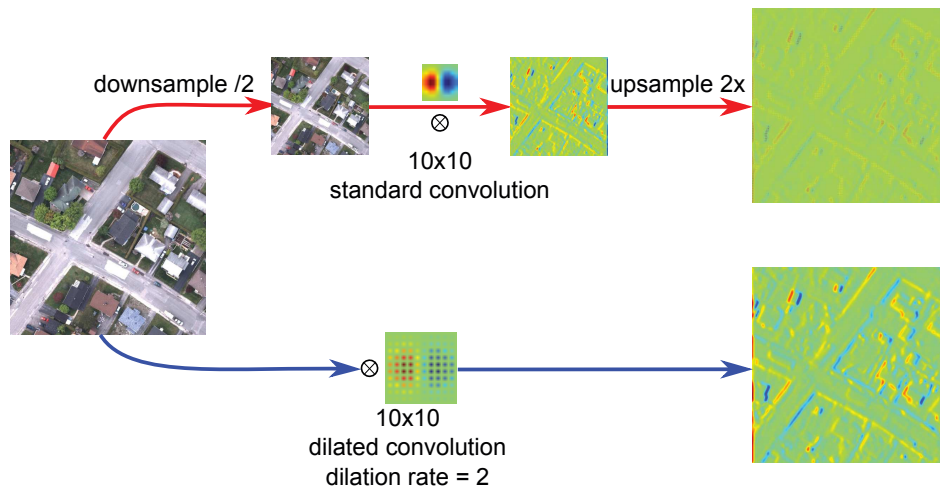
**Figure 3.2:** Example of dilated convolutions. Dilation supports expansion of the receptive field without loss of resolution or coverage of the input. The blue dot represents the pixel that is being considered as reference during the process.

tages, such as: (i) supports the expansion of the receptive field without increasing the number of trainable parameters per layer [Yu and Koltun, 2015], which reduces the computational burden, and (ii) preserves the feature map resolution, which may help the network to extract even more useful information from the data, mainly of small objects.

To better understand the aforementioned advantage, a comparison between dilated and standard convolution is presented in Figure 3.3. Given an image, the first network (in red) performs a downsampling operation (that reduces the resolution by a factor of 2) and a convolution, using horizontal Gaussian derivative as the kernel. The obtained low-resolution feature map is then enlarged by an upsampling operation (with a factor of 2) that restores the original resolution but not the information lost during the downsampling process. The second network (blue) computes the response of a dilated convolution on the original image. In this case, the same kernel was used but rearranged with dilation rate  $r = 2$ , making both networks have the same receptive field. Although the filter size increases, only non-zero values are taken into account when performing the convolution. Therefore, the number of filter parameters and of operations per position stay constant. Furthermore, it is possible to observe that salient features are better represented by the dilated model since no downsampling is performed over the input data.

### 3.1.2.3 Pooling Layer

Many of the features extracted by the convolutional layers are very similar, since each convolution window may have common pixels, generating redundant information. Typically, after the convolutional layers, there are pooling layers that were created in order



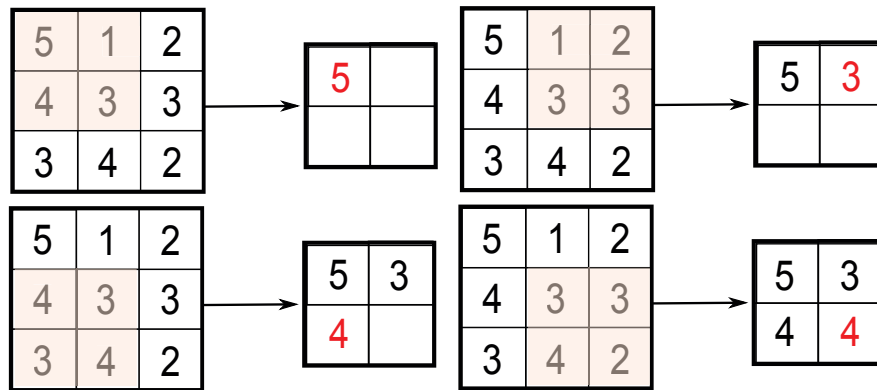
**Figure 3.3:** Comparison between dilated and standard convolutions. Top (red) row presents the feature extraction process using a standard convolution over a downsampled image and then an upsample in order to recover the input resolution (a common procedure performed in ConvNets). Bottom (blue) row presents the feature extraction process using dilated convolution with rate  $r = 2$  applied directly to the input (without downsample). The outcomes clearly show the benefits of dilated convolutions over standard ones.

to reduce the variance of features by computing some operation of a particular feature over a region of the image. Specifically, a fixed-size window runs over the features extracted by the convolutional layer and, at each step, an operation is realized to minimize the redundancy and optimize the gain of the features. Typically, two operations may be realized on the pooling layers: the max or average operation, which select the maximum or mean value over the feature region, respectively. Figure 3.4 presents an example of a pooling layer using max operation over the features. This process ensures that the same result can be obtained, even when image features have small translations or rotations, being very important for object classification and detection. Thus, the pooling layer is responsible for sampling the output of the convolutional one preserving the spatial location of the image, as well as selecting the most useful features for the next layers.

#### 3.1.2.4 Fully-connected Layer

After several convolutional and pooling layers, the architecture may have fully-connected ones, which are responsible to generate a general representation of data creating the feature vectors. Such layers take all neurons in the previous layer and connect them to every single neuron in its layer. The previous layers can be convolutional, pooling or fully-connected, however the next ones must be fully-connected until the classifier layer, because the spatial notion of the image is lost in this layer. Since a fully-connected layer occupies most of the parameters, overfitting can easily happen. To prevent this, the dropout method [Srivastava et al., 2014] can be employed. This





**Figure 3.4:** A pooling layer selecting the max value between the features inside a window of size  $2 \times 2$ . Figure adapted from [Ng et al., 2011b].

technique randomly drops several neuron outputs, which do not contribute to the forward pass and back propagation anymore. These drops are equivalent to decreasing the number of neurons of the network, improving the speed of training and making model combination practical, even for deep networks. Although this method creates networks with different architectures, those networks share the same weights, permitting model combination and allowing that only one network is needed at test time.

The last layer of a **ConvNet** architecture is usually a fully-connected layer with a specific configuration. Specifically, this layer is actually a fully-connected one but with number of neurons directly connected to the problem and with specific activation function (usually depending on the design of the network). A classifier layer may be used to calculate the class probability of each instance. The most common classifier layer is the softmax one [Bengio, 2009], based on the namesake function. The softmax function, or normalized exponential, is a generalization of the multinomial logistic function that generates a  $K$ -dimensional vector of real values in the range  $(0, 1)$  which represents a categorical probability distribution. Equation 3.5 shows how softmax function predicts the probability for the  $j$  class over a sample vector  $X$ .

$$h_{W,b}(X) = P(y = j|X; W, b) = \frac{\exp^{X^T W_j}}{\sum_{k=1}^K \exp^{X^T W_k}} \quad (3.5)$$

where  $j$  is the current class being evaluated,  $X$  is the input vector, and  $W$  represent the weights.

In addition to all the aforementioned layers, the network can also have several normalization ones, such as Local Response [Krizhevsky et al., 2012] and Batch Normalization [Ioffe and Szegedy, 2015]. These layers are used in specific cases, such as to preserve certain high-frequency features or to reduce the covariate shift of the internal layers.

### 3.1.3 Training

The **ConvNet** needs to adjust its weights and bias usually basing itself on the loss generated by the output layer, given that these models are based on a supervised procedure. Specifically, the training process of a network is strongly based upon a differentiable cost function  $\mathcal{J}(W, b)$  that is created to minimize the output error (Equation 3.6) making the **ConvNet** to converge to a (local or global) minimum (in terms of cost). Amongst several functions, the log loss one has become more pervasive because of exciting results achieved in some problems [Krizhevsky et al., 2012]. Equation 3.7 presents a general log loss function, without any regularization (or weight decay) term, which is used to prevent overfitting.

$$\underset{W, b}{\operatorname{argmin}}[\mathcal{J}(W, b)] \quad (3.6)$$

$$\mathcal{J}(W, b) = -\frac{1}{N} \sum_{i=1}^N (y^{(i)} \times \log h_{W, b}(x^{(i)}) + (1 - y^{(i)}) \times \log(1 - h_{W, b}(x^{(i)}))) \quad (3.7)$$

where  $y$  represents a possible class,  $x$  is the data of an instance,  $W$  the weights,  $i$  is an specific instance,  $h$  is the activation of the last layer, and  $N$  represents the total number of instances.

With the cost function defined, the **ConvNet** can be trained in order to minimize the loss by using some optimization algorithm, such as Stochastic Gradient Descent (SGD), to gradually update the weights and bias in search of the optimal solution:

$$W_{ij}^{(l)} = W_{ij}^{(l)} - \alpha \frac{\partial \mathcal{J}(W, b)}{\partial W_{ij}^{(l)}} \quad (3.8)$$

$$b_i^{(l)} = b_i^{(l)} - \alpha \frac{\partial \mathcal{J}(W, b)}{\partial b_i^{(l)}} \quad (3.9)$$

where  $\alpha$  denotes the learning rate, a parameter that determines how much an updating step influences the current value of the weights, i.e., how much the model learns in each step.

However, as presented, the partial derivatives of the cost function, for the weights and bias, are needed. To obtain these derivatives, the back propagation algorithm is used. Specifically, it must calculate how the error changes as each weight is increased or decreased slightly. The algorithm computes each error derivative by first computing the rate at which the error ( $E$ ) changes as the activity level of a unit is changed. For classifier layers, this error is calculated considering the predicted and desired output. For other layers, this error is propagated by considering the weights between each pair of layers and the error generated in the most advanced layer.

The training step of a ConvNet occurs in two steps: (i) the feed-forward one, that passes the information through all the network layers, from the first until the classifier one, usually with high batch size<sup>3</sup>, and (ii) the back propagation one, which calculates the error ( $E$ ) generated by the ConvNet and propagates this error through all the layers, from the classifier until the first one. As presented, this step also uses the errors to calculate the partial derivatives of each layers for the weights and bias.

## 3.2 Mathematical Morphology

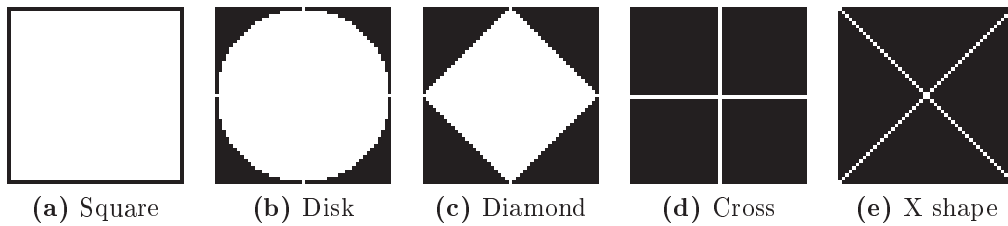
Mathematical morphology, a combination of set and lattice theories, is the foundation of the morphological operations, commonly employed in the image processing area. Since its introduction to the remote sensing domain [Pesaresi and Benediktsson, 2001], these morphological operations have been generalized from the analysis of a single band image to hyperspectral images made up of hundreds of spectral channels and has become one of the state-of-the-art techniques for a wide range of applications [Serra and Soille, 2012]. This study area includes several different operations (such as erosion, dilation, opening, closing, top-hats, and reconstruction), which can be applied to binary and grayscale images in any number of dimensions [Serra and Soille, 2012].

Formally, let us consider a grayscale 2D image  $I(\cdot)$  as a mapping from the coordinates ( $\mathbb{Z}^2$ ) to the pixel-value domain ( $\mathbb{Z}$ ). Most morphological transformations process this input image  $I$  using a structuring element (SE) (usually defined prior to the operation). A SE  $B(\cdot)$  can be defined as a function that returns the set of neighbors of a pixel  $(i, j)$ . This neighborhood of pixels is taken into account during the morphological operation, i.e., while probing the image  $I$ . Normally, a SE is defined by two components: (i) shape, which is usually a discrete representation of continuous shapes, such as square, circle, (ii) center, that identifies the pixel on which the SE is superposed when probing the image. Figure 3.5 presents some examples of common SEs employed in the literature. As introduced, the definition of the SE is of vital importance for the process to extract relevant features. However, in literature [Gu et al., 2016; Liu et al., 2017], this definition is performed experimentally, an expensive process that does not guarantee a good descriptive representation.

After its definition, the SE can be then employed in several morphological processes. Most of these operations are usually supported by two basic morphological transformations: *erosion*  $\mathcal{E}(\cdot)$  and *dilation*  $\delta(\cdot)$ . Such operations receive basically the same input: an image  $I$  and the structuring element  $B$ . While erosion transformations

---

<sup>3</sup>Batch size is a parameter that determines the number of images that goes through the network before the weights and bias are updated.



**Figure 3.5:** Examples of common structuring elements employed in the literature. These SEs can be seen as sets that define the activated pixels.

process each pixel  $(i, j)$  using the supremum (the smallest upper bound,  $\wedge$ ) function, as formally denoted in Equation 3.10, the dilation operations process the pixels using the infimum (the greatest lower bound,  $\vee$ ) function, as presented in Equation 3.11. Intuitively, these two operations probe an input image using the SE, i.e., they position the structuring element at all possible locations in the image and analyze the neighborhood pixels. This process, very similar to the convolution procedure, outputs another image with regions compressed or expanded (depending on the operation). Some examples of erosion and dilation are presented in Figure 3.6, in which it is possible to notice the behavior of each operation. Precisely, erosion affects brighter structures while dilation influences darker ones (with respect to the neighborhood defined by the SE).

$$\mathcal{E}(B, I)_{(i,j)} = \{\wedge I((i, j)') | (i, j)' \in B(i, j) \cup I(i, j)\} \quad (3.10)$$

$$\delta(B, I)_{(i,j)} = \{\vee I((i, j)') | (i, j)' \in B(i, j) \cup I(i, j)\} \quad (3.11)$$

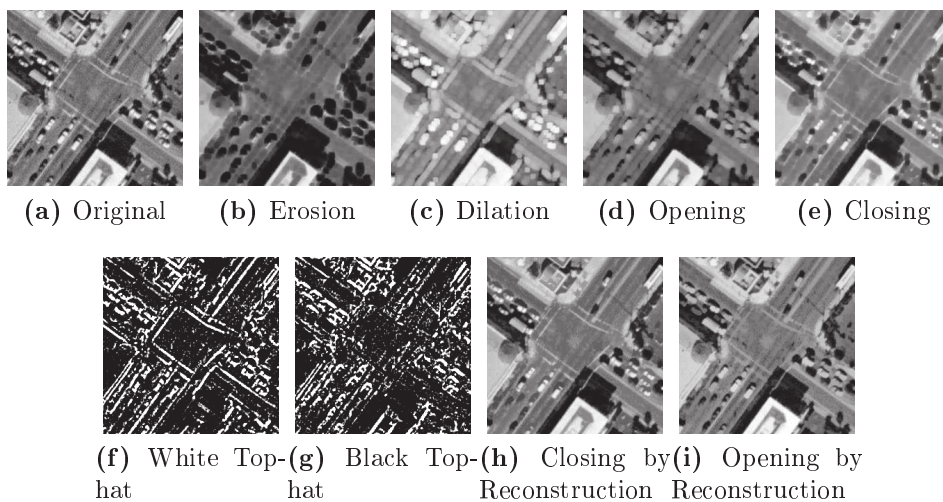
If we have an ordered set, then the erosion and dilation operations can be simplified. This is because the infimum and the supremum are respectively equivalent to the minimum and maximum functions when dealing with ordered sets. In this case, erosion and dilation can be defined as presented in Equations 3.12 and 3.13, respectively.

$$\mathcal{E}(B, I)_{(i,j)} = \left\{ \min_{(i,j)' \in B(i,j)} (I((i, j)')) \right\} \quad (3.12)$$

$$\delta(B, I)_{(i,j)} = \left\{ \max_{(i,j)' \in B(i,j)} (I((i, j)')) \right\} \quad (3.13)$$

Based on these two fundamental transformations, other more complex morphological operations may be computed. The morphological *opening*, denoted as  $\gamma(\cdot)$  and defined in Equation 3.14, is simply an erosion operation followed by the dilation (using the same structuring element) of the eroded output. In contrast, a morphological *closing*  $\varphi(\cdot)$  of an image, formally defined in Equation 3.15, is a dilation followed by the erosion (using the same SE) of the dilated output. Intuitively, while an erosion would affect all brighter structures, an opening flattens bright objects that are smaller than the size of the SE and, because of dilation, mostly preserves the bright large areas.

A similar conclusion can be drawn for darker structures when closing is performed. Examples of this behavior can be seen in Figure 3.6. It is important to highlight that by using erosion and dilation transformations, opening and closing perform geodesic reconstruction in the image. Operations based on this paradigm belongs to the class of filters that operate only on connected components (flat regions) and cannot introduce any new edge to the image. Furthermore, if a segment (or component) in the image is larger than the  $SE$  then it will be unaffected, otherwise, it will be merged to a brighter or darker adjacent region depending upon whether a closing or opening is applied. This is crucial because avoids the generation of distorted structures, which is obviously an undesirable effect.



**Figure 3.6:** Examples of morphological images generated for the UC Merced Land-use Dataset. All these images were processed using a  $5 \times 5$  square as structuring element.

$$\gamma(B, I) = \delta(B, \mathcal{E}(B, I)) \quad (3.14)$$

$$\varphi(B, I) = \mathcal{E}(B, \delta(B, I)) \quad (3.15)$$

Other important morphological operations are the *top-hats*. Top-hat transform is an operation that extracts small elements and details from given images. There are two types of top-hat transformations: (i) the white one  $\mathcal{T}^w(\cdot)$ , defined in Equation 3.16, in which the difference between the input image and its opening is calculated, and (ii) the black one, denoted as  $\mathcal{T}^b(\cdot)$  and defined in Equation 3.17, in which the difference between the closing and the input image is performed. White top-hat operation preserves elements of the input image brighter than their surroundings but smaller than the  $SE$ . On the other hand, black top-hat maintains objects smaller than the  $SE$  with brighter surroundings. Examples of these two operations can be seen in Figure 3.6.

$$\mathcal{T}^w(B, I) = I - \gamma(B, I) \quad (3.16)$$

$$\mathcal{T}^b(B, I) = \varphi(B, I) - I \quad (3.17)$$

Another important morphological operation based on erosions and dilations is the *geodesic reconstruction*. There are two types of geodesic reconstruction: by erosion and by dilation. For simplicity, only the former one is formally detailed here, however, the latter one can be obtained, by duality, using the same reasoning. The geodesic reconstruction by erosion  $\rho^{\mathcal{E}}(\cdot)$ , mathematically defined in Equation 3.18, receives two parameters as input: an image  $I$  and a SE  $B$ . The image  $I$  (also referenced in this operation as mask image) is dilated by the SE  $B$  ( $\delta(B, I)$ ) creating the marker image  $Y$  ( $Y \in I$ ), responsible for delimiting which objects will be reconstructed during the process. A SE  $B'$  (usually with any elementary composition [Serra and Soille, 2012]) and the marker image  $Y$  are provided for the reconstruction operation  $\mathcal{R}_I^{\mathcal{E}}(\cdot)$ . This transformation, defined in Equation 3.19, reconstructs the marker image  $Y$  (with respect to the mask image  $I$ ) by recursively employing geodesic erosion (with the elementary SE  $B'$ ) until idempotence is reached (i.e.,  $\mathcal{E}_I^{(n)}(\cdot) = \mathcal{E}_I^{(n+1)}(\cdot)$ ). In this case, a geodesic erosion  $\mathcal{E}_I^{(1)}(\cdot)$ , defined in Equation 3.20, consists of a pixel-wise maximum operation between an eroded (with elementary SE  $B'$ ) marker image  $Y$  and the mask image  $I$ . As aforementioned, by duality, a geodesic reconstruction by dilation can be defined, as presented in Equation 3.21. These two crucial operations try to preserve all large (than the SE) objects of the image removing bright and dark small areas, such as noises. Some examples of these operations can be seen in Figure 3.6.

$$\rho^{\mathcal{E}}(B, I) = \mathcal{R}_I^{\mathcal{E}}(B', Y) = \mathcal{R}_I^{\mathcal{E}}(B', \delta(B, I)) \quad (3.18)$$

$$\begin{aligned} \mathcal{R}_I^{\mathcal{E}}(B', Y) &= \mathcal{E}_I^{(n)}(B', Y) = \\ &= \underbrace{\mathcal{E}_I^{(1)}\left(B', \mathcal{E}_I^{(1)}\left(B', \dots \mathcal{E}_I^{(1)}\left(B', \mathcal{E}_I^{(1)}(B', Y)\right)\right)\right)}_{\text{n times}} \end{aligned} \quad (3.19)$$

$$\mathcal{E}_I^{(1)}(B', Y) = \max\{\mathcal{E}(B', Y), I\} \quad (3.20)$$

$$\rho^{\delta}(B, I) = \mathcal{R}_I^{\delta}(B', Y) = \mathcal{R}_I^{\delta}(B', \mathcal{E}(B, I)) \quad (3.21)$$

Note that geodesic reconstruction operations require an iterative process until the convergence. This procedure can be expensive, mainly when working with a large number of images (a common scenario when training neural networks [Goodfellow et al., 2016]). An approximation of such operations, presented in Equations 3.22 and 3.23 can

be achieved by performing just **one** transformation over the marker image with a large (than the **SE** used to create the marker image) structuring element. In other words, suppose that  $B$  is the **SE** used to create the marker image, then  $B'$ , the **SE** used in the reconstruction step, should be larger than  $B$ , i.e.,  $B \subset B'$ . This process is faster since only one iteration is required, but may lead to worse results, given that the use of a large filter can make the reconstruction join objects that are close in the scene (a phenomenon known as *leaking* [Serra and Soille, 2012]).

$$\tilde{\rho}^{\mathcal{E}}(B, I) = \mathcal{E}_I(B', \delta(B, I)) \quad (3.22)$$

$$\tilde{\rho}^{\delta}(B, I) = \delta_I(B', \mathcal{E}(B, I)) \quad (3.23)$$

Although all previously defined morphological operations used a grayscale image  $I$ , they could have employed a binary image or even an image with several channels. In this case, morphological operations would be applied to each input channel independently and separately, generating an outcome with the same number of input channels.





# Chapter 4

## General Experimental Setup

This chapter describes the distinct datasets, protocol, and measures employed to validate the techniques proposed in this work. Section 4.1 presents several datasets used in both image and pixel classification. Section 4.2 defines the experimental protocols employed in this work. Finally, Section 4.3 introduces the measures used to evaluate the results obtained in the performed experiments.

### 4.1 Datasets

In order to better analyze the effectiveness and robustness of each proposed technique, we have selected several remote sensing datasets with very distinctive properties. Next, we present the datasets used in the scene classification task followed by the ones used in the semantic segmentation task.

#### 4.1.1 Scene Classification Datasets

As aforementioned, datasets with different visual properties were selected. The first one, presented in Section 4.1.1.1, is a multi-class land-use dataset that contains aerial high-resolution scenes in the visible spectrum. Section 4.1.1.2 introduces the second dataset, which is a multi-class high-resolution dataset with images collected from different regions all around the world. Finally, the last one, presented in Section 4.1.1.3, has multispectral high-resolution scenes of coffee crops and non-coffee areas. An overview of these datasets is shown in Table 4.1. The datasets are described in details in the following sections.

**Table 4.1:** Overview statistics for the three classification datasets employed in this work.

	UC-Merced	WHU-RS19	Coffee Scenes
<b>Type</b>	Land-use	Land-use	Agricultural
<b>Total Size</b>	2,100	1,005	2,876
<b>Image Dimensions (px.)</b>	$256 \times 256$	$600 \times 600$	$64 \times 64$
<b>Band Composition</b>	R-G-B	R-G-B	NIR-R-G
<b>Spatial Resolution</b>	0.3m	0.5m	2.5m
<b>Number of Classes</b>	21	19	2
<b>Average Distribution per Class</b>	100	52	1,438

#### 4.1.1.1 UCMerced Land-use Dataset

This publicly available dataset [Yang and Newsam, 2010] is composed of 2,100 aerial scene images each one with a resolution of  $256 \times 256$  pixels and 0.3-meter resolution per pixel. These images, composed of bands in the visible spectrum and obtained from different United States locations (via the US Geological Survey (USGS) National Map), were manually classified into 21 classes: agricultural, airplane, baseball diamond, beach, buildings, chaparral, dense residential, forest, freeway, golf course, harbor, intersection, medium density residential, mobile home park, overpass, parking lot, river, runway, sparse residential, storage tanks, and tennis courts. Samples of all classes are presented in Figure 4.1. It is remarkable the overlapping of some classes, such as “dense residential”, “medium residential” and “sparse residential”, which mainly differ in the density of structures.

#### 4.1.1.2 WHU-RS19 Dataset

This dataset [Xia et al., 2010] contains 1,005 high-spatial resolution images (composed of bands in the visible spectrum) with  $600 \times 600$  pixels divided into 19 classes, with approximately 50 images per class. Exported from Google Earth, which provides high-resolution satellite images up to half a meter (0.5m), this dataset has samples collected from different regions all around the world, which increases its diversity but creates challenges due to the changes in resolution, scale, orientation and illumination of the images. There are 19 classes, including: airport, beach, bridge, river, forest, meadow, pond, parking, port, viaduct, residential area, industrial area, commercial area, desert, farmland, football field, mountain, park, and railway station. Figure 4.2 presents examples of these classes.

**Table 4.2:** Overview statistics for the two pixel classification datasets employed in this work.

	Coffee	GRSS Data Fusion	Vaihingen	Potsdam
<b>Terrain</b>	Mountainous	Plain	Plain	Plain
<b>Image Class</b>	Agriculture	Urban	Urban	Urban
<b>Number of Images</b>	5	2 (train/test)	33 (16 with ground-truth)	38 (24 with ground-truth)
<b>Image Dimensions (px.)</b>	$500 \times 500$	$2830 \times 3989 / 3769 \times 4386$	$2494 \times 2064$ (average)	$6000 \times 6000$
<b>Band Composition</b>	NIR-R-G	R-G-B	NIR-R-G	NIR-R-G-B
<b>Spatial Resolution</b>	2.5m	0.2m	0.9m	0.5m
<b>Number of Classes</b>	2	7	6	6

#### 4.1.1.3 Brazilian Coffee Scenes

This public dataset [Penatti et al., 2015] is composed of 2,876 multi-spectral tiles extracted from four counties in the State of Minas Gerais, Brazil (Arceburgo, Guaranésia, Guaxupé, and Monte Santo) and equally divided into 2 classes: coffee and non-coffee. Each non-overlapping tile, with a spatial resolution of  $64 \times 64$  pixels and pixel resolution of  $2.5m$ , is composed of near-infrared, red and green bands (in this order), which are the most useful and representative ones for discriminating vegetation areas. Some samples of this dataset are presented in Figure 4.3.

It is important to emphasize that this dataset has its own particular challenges, such as: (i) high intraclass variance, since distinct techniques are used to manage coffee crops, (ii) high interclass variance, given that coffee is an evergreen culture and there are plants with different ages, and (iii) images with spectral distortions caused by shadows, since the South of Minas Gerais is a mountainous region.

### 4.1.2 Pixel Classification Datasets

As for the classification task, datasets with different properties were selected in order to evaluate the creation of thematic maps. The first one, named Coffee Dataset and presented in Section 4.1.2.1, is composed of multispectral high-resolution scenes of coffee crops and non-coffee areas. Section 4.1.2.2 presents the second dataset, referred here as GRSS Data Fusion Dataset, which is a very high spatial resolution in the visible spectrum with the objective of mapping urban targets, such as roads and buildings. The last two datasets related to the remote sensing pixel classification task are presented in Section 4.1.2.3 and 4.1.2.4. These datasets, referenced hereafter as Vaihingen and Potsdam, are composed of multispectral high-resolution urban images. Some statistical about these datasets are presented in Table 4.2. More details about each dataset are given in the following sections.

**Table 4.3:** Number of pixels per class for the GRSS Data Fusion Dataset.

Classes	Train		Test	
	#Pixels	%	#Pixels	%
<b>Road</b>	112,457	19,83	808,490	55,77
<b>Trees</b>	27,700	4,89	100,528	6,93
<b>Red roof</b>	45,739	8,05	136,323	9,40
<b>Grey roof</b>	53,520	9,44	142,710	9,84
<b>Concrete roof</b>	97,821	17,25	109,423	7,55
<b>Vegetation</b>	185,242	32,65	102,948	7,10
<b>Bare soil</b>	44,738	7,89	49,212	3,41
<b>Total</b>	567,217	100,00	1,449,634	100,00

#### 4.1.2.1 Coffee Dataset

This dataset is a composition of five images taken by the SPOT sensor in 2005 over Monte Santo, a coffee grower county in the State of Minas Gerais, Brazil. Each image has  $500 \times 500$  pixels with near-infrared, red and green bands (in this order), which are the most useful and representative ones for discriminating vegetation areas. More specifically, the dataset has 1,250,000 pixels with 637,544 (51%) coffee pixels and 612,456 (49%) non-coffee pixels annotated by specialists. Figure 4.4 shows each image and the respective ground-truths.

This is a very challenging dataset since intraclass variance is high due to different crop management techniques. Furthermore, coffee is an evergreen culture and the South of Minas Gerais is a mountainous region, which means that this dataset includes scenes with plants at different stages of growth and/or with spectral distortions caused by shadows.

#### 4.1.2.2 GRSS Data Fusion Dataset

Proposed in the IEEE GRSS Data Fusion Contest 2014, this dataset is composed of a fine-resolution visible image that covers an urban area near Thetford Mines in Quebec, Canada, containing seven different classes: trees, vegetation, road, bare soil, red roof, gray roof, and concrete roof. In this work, we do not consider the hyperspectral data available in this dataset. Table 4.3 shows pixel class distribution in this dataset. It is important to highlight that not all pixels are classified into one of these categories, with some pixels considered as uncategorized or unclassified.

Both training and testing images have 0.2 meter of spatial resolution, with the former having  $2830 \times 3989$  pixels of resolution while the latter has  $3769 \times 4386$  pixels. These images, as well as the respective ground-truths, are presented in Figure 4.5.

**Table 4.4:** Number of pixels per class for ISPRS datasets, i.e., Vaihingen and Potsdam.

Classes	Vaihingen		Potsdam	
	#Pixels	%	#Pixels	%
<b>Impervious Surfaces</b>	21,815,349	27.94	245,930,445	28.46
<b>Building</b>	20,417,332	26.15	230,875,852	26.72
<b>Low Vegetation</b>	16,272,917	20.84	203,358,663	23.54
<b>Tree</b>	18,110,438	23.19	126,352,970	14.62
<b>Car</b>	945,687	1.21	14,597,667	1.69
<b>Clutter/Background</b>	526,083	0.67	42,884,403	4.96
<b>Total</b>	78,087,806	100.00	864,000,000	100.00

#### 4.1.2.3 Vaihingen Dataset

This dataset [ISPRS, 2018a] was released for the 2D semantic labeling contest of the International Society for Photogrammetry and Remote Sensing (ISPRS). It is composed by a total of 33 image tiles (with an average size of  $2494 \times 2064$  pixels), that are densely classified into six possible labels: impervious surfaces, building, low vegetation, tree, car, clutter/background. Sixteen of these images have ground-truth available while the remaining ones, considered the test set, do not have available annotation, requiring submission of the prediction maps in order to be evaluated. The pixel distribution for the labeled images can be seen in Table 4.4.

Each image of this dataset is composed of near-infrared, red and green channels (in this order) and has a spatial resolution of 0.9 meter. A Digital Surface Model (DSM) coregistered to the image data was also provided, allowing the creation of a normalized DSM (nDSM) [Gerke, 2015]. In this work, we use the spectral information (NIR-R-G) and the nDSM, i.e., the input data has 4 dimensions: NIR-R-G and nDSM. Examples of the Vaihingen Dataset can be seen in Figure 4.6.

#### 4.1.2.4 Potsdam Dataset

Also proposed for the 2D semantic labeling contest, this dataset [ISPRS, 2018b] has 38 tiles of the same size ( $6000 \times 6000$  pixels), with a spatial resolution of 0.5 meter. From the available patches, 24 are densely annotated (with same classes as for the Vaihingen dataset), in which the pixel distribution is presented in Table 4.4. Analogously to the Vaihingen dataset, the remaining images are considered the test set and do not have available annotation, requiring submission of the prediction maps in order to be evaluated. This dataset consists of 4-channel images (near-infrared, red, green and blue), Digital Surface Model (DSM), and normalized DSM (nDSM). In this work, all

spectral channels plus the nDSM are used as input, resulting in a 5-dimensional input data. Some samples of these images are presented in Figure 4.7.

## 4.2 Protocols

Some of the experiments presented in this thesis were carried out using a traditional  $k$ -fold cross-validation protocol while others were performed using the simple training/test protocol.

In the former protocol, a dataset is randomly split into  $k$  mutually exclusive subsets (folds) of almost the same size. Usually,  $k - 2$  subsets are chosen as the training set, one fold is used as test-set, and the remaining one is the validation-set. However, sometimes, the validation set is not necessary (when there is no parameter to tune or evaluate, etc). In these cases,  $k - 1$  subsets are chosen as the training set, and the remaining one is the test set. In order to work with the whole dataset, the cross-validation process is repeated  $k$  times, and at each time a subset is chosen to be the test set (and another one is selected as validation set when needed) without repetition. In the end, results are reported using an average of different measures in terms of the  $k$  runs.

Specifically, this protocol was used, considering  $k = 5$ , in all classification datasets and in the Coffee dataset. For the UCMerced and WHU-RS19 datasets, each fold has an equal size (i.e., 420 and 201 images per fold, respectively) and is fully balanced in terms of the number of classes per fold. For the Brazilian Coffee Scenes dataset, 4 folds have 600 images each and the 5<sup>th</sup> has 476 images. In this case, all folds are balanced with coffee and non-coffee samples (50% each). For the Coffee dataset, each image is considered as a unique fold. So, there are 5 images/folds, which are combined as aforesaid.

In the training/test protocol, the dataset is divided into train and test. This is the most simple protocol, in which the training subset is used to train the model which is then evaluated in the test subset. When necessary, this protocol may become a training/validation/test one, in which the dataset is divided into three distinct sets: the first is used to train the model, the second to tune or evaluate the parameters, and the last one to analyze the generalization power of the trained model. In both cases, the final results are reported based on the performance on the test set. This protocol was employed in three remaining pixel classification datasets. Precisely, the GRSS Data Fusion dataset is experimented using this protocol, given its clear definition of training and test subsets. For Vaihingen [ISPRS, 2018a] and Potsdam [ISPRS, 2018b] datasets,

**Table 4.5:** Confusion matrix in which each  $x_{ij}$  represents the number of scenes or pixels (depending on the task) in the classified (observed) image category  $i$  and the ground truth (reference) cover category  $j$ . Adapted from [Liu et al., 2007].

		Reference				
		1	2	...	c	Total
Observed	1	$x_{11}$	$x_{12}$	...	$x_{1c}$	$x_{1+}$
	2	$x_{21}$	$x_{22}$	...	$x_{2c}$	$x_{2+}$
	⋮	⋮	⋮	⋮	⋮	⋮
	c	$x_{c1}$	$x_{c2}$	...	$x_{cc}$	$x_{c+}$
	Total	$x_{+1}$	$x_{+2}$	...	$x_{+c}$	

the training/validation/test protocol was employed based on the literature [Marmanis et al., 2018; Volpi and Tuia, 2017]. In this case, for the Vaihingen dataset, 11 out of the 16 annotated images were used as training set. The 5 remaining images (with IDs 11, 15, 28, 30, 34) were employed to validate the model. For the Potsdam dataset, 18 (out of 24) annotated images were used as training set while the remaining 6 (with IDs 02\_12, 03\_12, 04\_12, 05\_12, 06\_12, 07\_12) were employed to validate the method. In both cases, a separated test set is used to evaluate the trained model.

### 4.3 Measures

In our experiments, results are reported using measures in terms of values stored in confusion matrices [Liu et al., 2007]. Table 4.5 presents a confusion matrix for  $c$  classes constructed with both reference and the classified data, which may be pixels or image scenes. Based on this matrix, four evaluation measures were employed throughout this thesis: overall and average accuracy, kappa index ( $\kappa$ ), and F1 score. These metrics were selected based on their diversity: overall accuracy and kappa are biased toward classes with more samples (relevance of classes with small amount of samples are canceled out by those with large amount) while average accuracy and F1 are calculated specifically for each class and, therefore, are independent of class size [Volpi and Tuia, 2017]. A comparison of these, and other measures, can be found in [Ferri et al., 2009].

The first metric, overall accuracy [Congalton, 1991], is the most popular accuracy measure. This metric, formally presented in Equation 4.1, considers the global aspects of the classification by averaging all correctly classified pixels indistinctly. As aforementioned, this process can lead to several problems if the dataset is unbalanced, i.e., if the number of instances per class is significantly different. In these cases, the overall accuracy may give a distorted impression of the results, because the class with

more examples may dominate the statistic.

$$OA = \frac{\sum_{i=1}^c x_{ii}}{N} \times 100 \quad (4.1)$$

where  $x_{ii}$  is the number of observations in row  $i$  and column  $i$ ; and  $N$  is the total number of observations.

Overcoming this problem, the average accuracy [Congalton, 1991] reports the average (per-class) ratio of correctly classified samples, i.e., it outputs an average of the accuracy of each class. Though not so common than overall accuracy, this measure is more robust mainly when the dataset is unbalanced, given that each class has the average calculated separately. In general, when there are different numbers of samples per class, the average accuracy, presented in Equation 4.2, will be different from the overall accuracy.

$$AA = \frac{\sum_{i=1}^c \frac{x_{ii}}{x_{+i}}}{c} \times 100 \quad (4.2)$$

where  $x_{ii}$  is the number of observations in row  $i$  and column  $i$ ; and  $x_{+i}$  are the marginal totals of column  $i$ .

Kappa index ( $\kappa$ ) [Congalton, 1991] measures the agreement between the reference data and the classified result. This measure, computed as in Equation 4.3, has values that varies from  $-1$  to  $1$ . In general, negative Kappa means that there is no agreement at all between classified data and reference data. Kappa value closer to one means a better result, with  $\kappa = 1$  being “perfect agreement”. Experiments in different areas show that Kappa could have various interpretations and these guidelines could be different depending on the application. Similar to the overall accuracy, this metric can be biased to classes with more samples [Volpi and Tuia, 2017].

$$\kappa = \frac{N \sum_{i=1}^c x_{ii} - \sum_{i=1}^c (x_{i+} \times x_{+i})}{N^2 - \sum_{i=1}^c (x_{i+} \times x_{+i})} \quad (4.3)$$

where  $x_{ii}$  is the number of observations in row  $i$  and column  $i$ ;  $x_{i+}$  and  $x_{+i}$  are the marginal totals of row  $i$  and column  $i$ , respectively; and  $N$  is the total number of observations.

The last measure employed in this work is the F1 score [Ferri et al., 2009]. This metric, formally presented in Section 4.4, is defined as the harmonic mean of precision and recall. Note that, similarly to the average accuracy, the F1 score calculates an average of the harmonic mean of each class. Hence, again, this metric is more robust



to imbalanced datasets.

$$F1 = \frac{1}{c} \sum_{i=1}^c \left( 2 \cdot \frac{\frac{x_{ii}}{x_{i+}} \cdot \frac{x_{ii}}{x_{+i}}}{\frac{x_{ii}}{x_{i+}} + \frac{x_{ii}}{x_{+i}}} \right) \quad (4.4)$$

where  $x_{ii}$  is the number of observations in row  $i$  and column  $i$ ;  $x_{i+}$  and  $x_{+i}$  are the marginal totals of row  $i$  and column  $i$ , respectively.

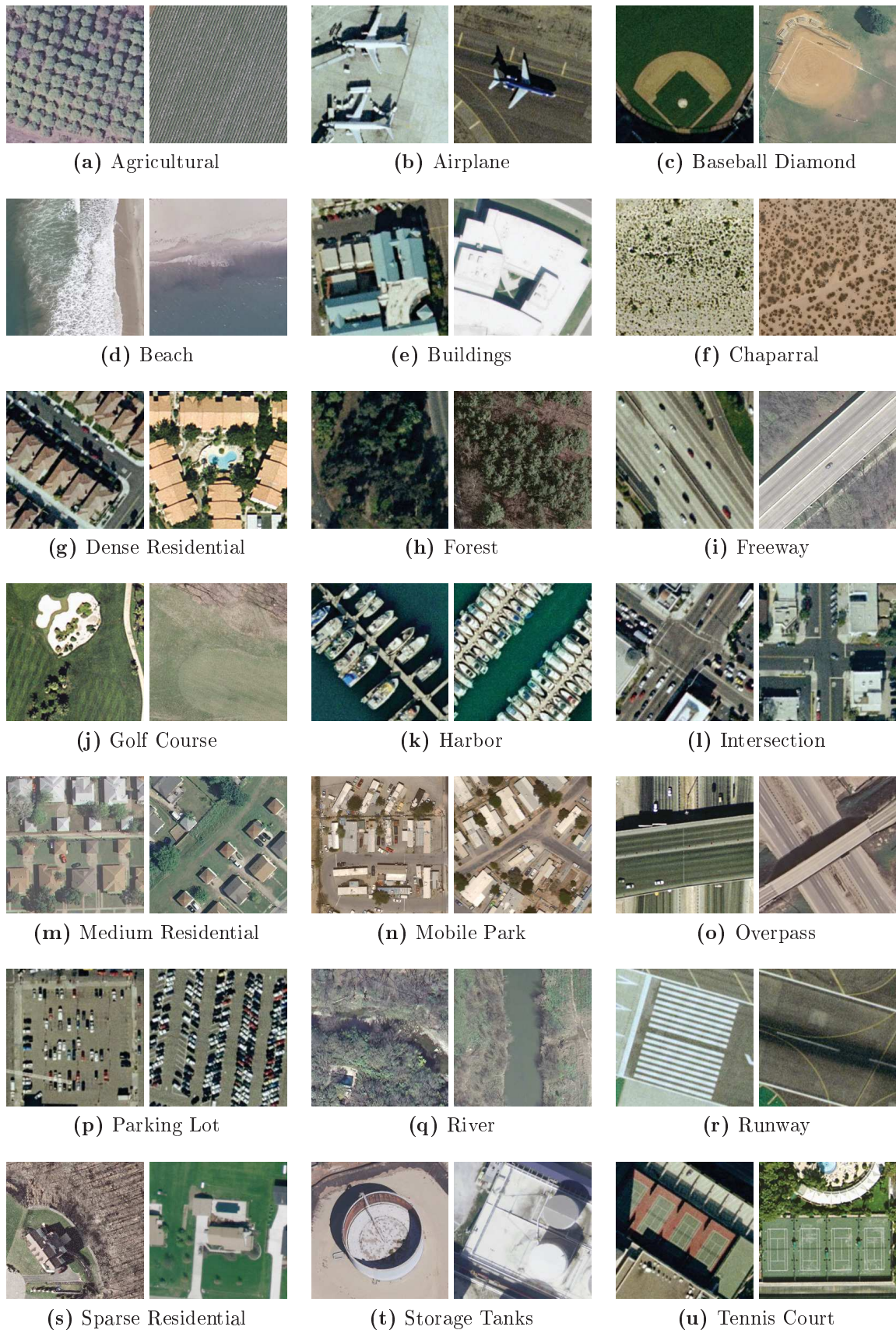
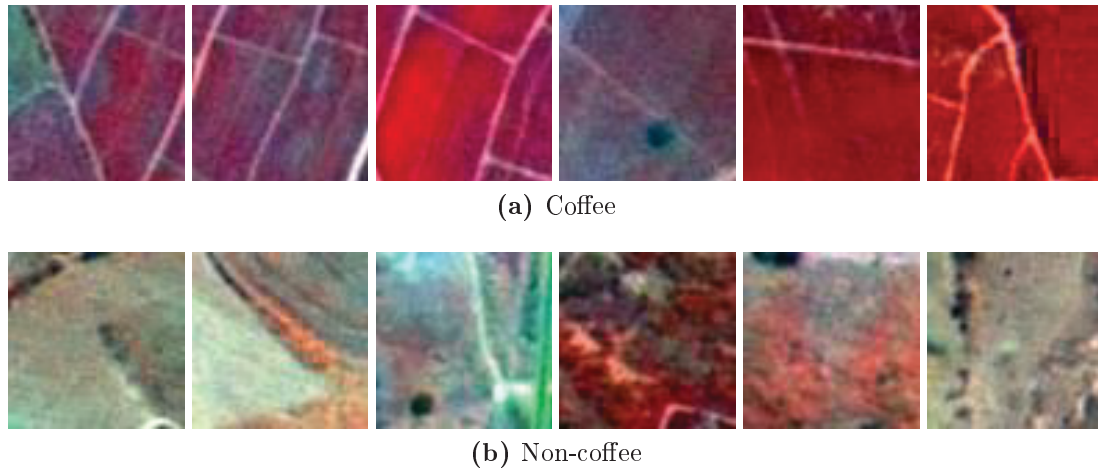


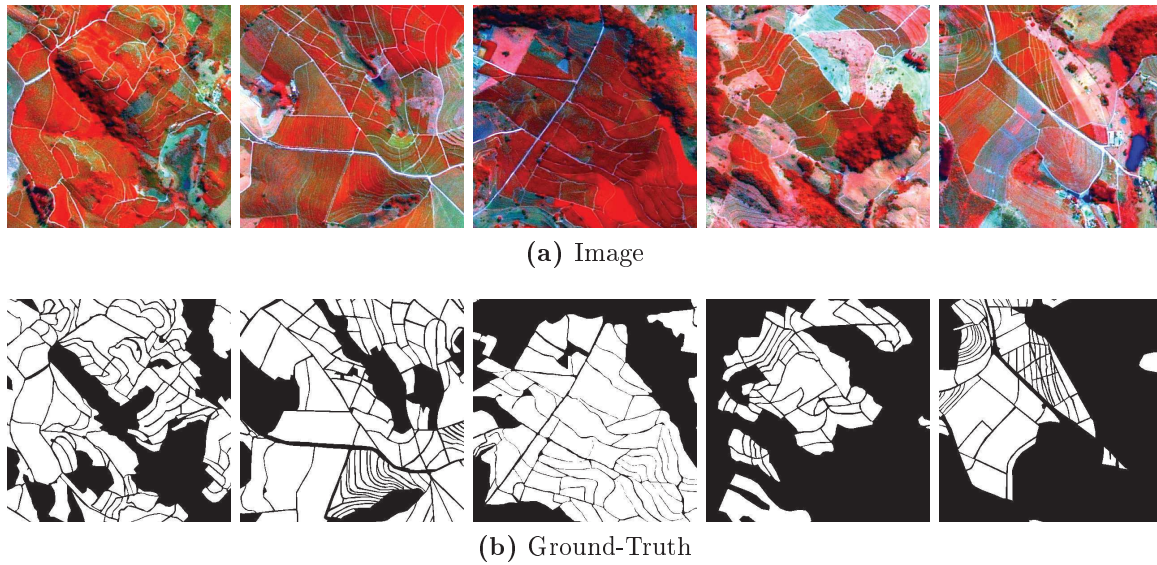
Figure 4.1: Examples of the UCMerced Land Use Dataset.



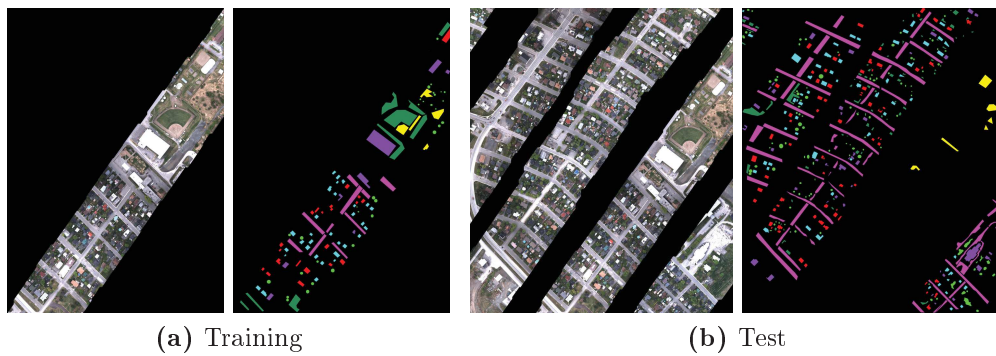
Figure 4.2: Examples of the WHU-RS19 Dataset.



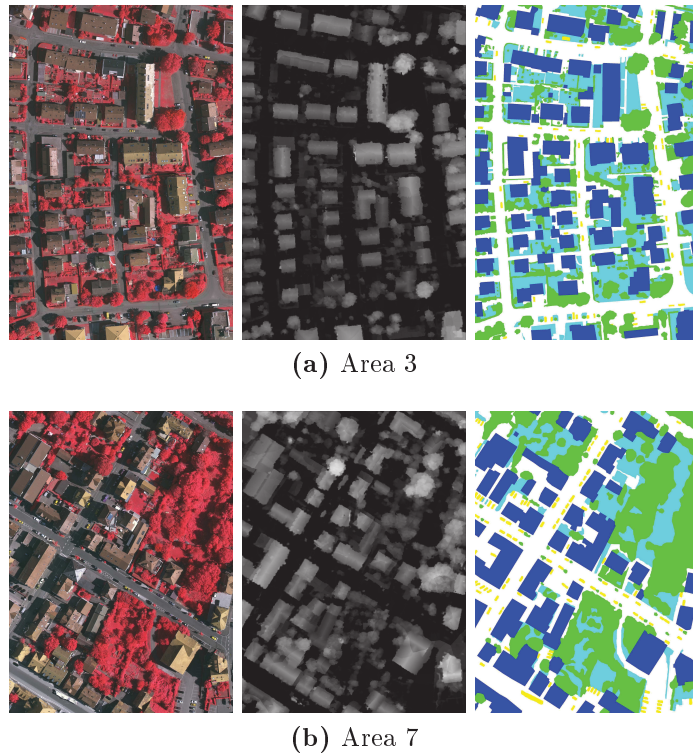
**Figure 4.3:** Examples of coffee and non-coffee samples in the Brazilian Coffee Scenes dataset. The similarity among samples of opposite classes is notorious. The intraclass variance is also perceptible.



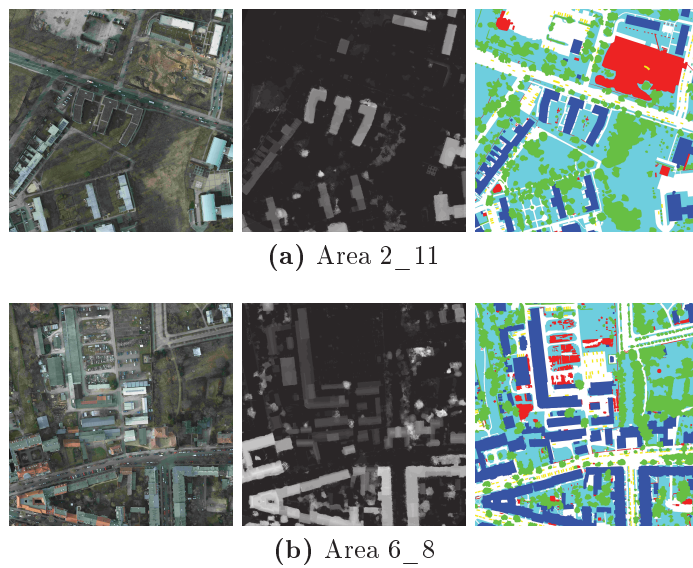
**Figure 4.4:** The Coffee Dataset. Multispectral images and ground-truths. Legend – White: Coffee areas. Black: Non Coffee areas.



**Figure 4.5:** The GRSS Data Fusion Dataset. Training and test data and their respective ground-truths. Legend – Black: unclassified. Light purple: road. Light green: trees. Red: red roof. Cyan: gray roof. Dark purple: concrete roof. Dark green: vegetation. Yellow: bare soil.



**Figure 4.6:** Examples of the Vaihingen Dataset. Legend – White: impervious surfaces. Blue: buildings. Cyan: low vegetation. Green: trees. Yellow: cars. Red: clutter, background.



**Figure 4.7:** Examples of the Potsdam Dataset. Legend – White: impervious surfaces. Blue: buildings. Cyan: low vegetation. Green: trees. Yellow: cars. Red: clutter, background.



# Chapter 5

## ConvNet-Based Scene Classification

ConvNets are the most propitious network to be used with images (including RSIs), given its natural process of learning features with different levels of abstraction directly from the input data, exploring its stationary property [Goodfellow et al., 2016]. Although powerful, the exploration of the potential of deep ConvNets can be a complex task because of several challenges [Bengio, 2009; Goodfellow et al., 2016], such as: (i) high computational burden, since it needs a lot of data and computational resources, (ii) complex tuning, since convergence cannot be totally confirmed, (iii) “black box” nature, since there is no mathematical model that explains the whole network, (iv) proneness to overfitting, given the high number of parameters, and (v) empirical nature of model development, which is over parameterized. However, through years, researchers have developed strategies to explore the potential of deep ConvNets in different domains and scenarios. In this chapter, we evaluate and analyze possible strategies of exploiting ConvNets for remote sensing scene classification.

**Definition 1. Scene Classification.** Let  $\mathcal{D}$  and  $\mathcal{T}$  be the training and testing sets, respectively. The training set  $\mathcal{D}$  consists of tuples of the form  $\{I_i, c_i\}$ , where  $I_i$  is an image and  $c_i$  its respective class. Note that  $i \in \{1, \dots, |\mathcal{D}|\}$  and  $c_i \in \mathcal{C}$ , where  $\mathcal{C} = \{C_1, C_2, \dots, C_k\}$  is the set of the  $k$  possible classes. Distinctly, the testing set  $\mathcal{T}$  is only composed of images  $I_j$  (with  $j \in \{1, \dots, |\mathcal{T}|\}$ ).

$\mathcal{D}$  is used to perform a supervised training of a function  $\mathcal{F}$  that receives as input an image  $I_i$  and outputs a membership probability  $p(c|I_i)$  for each class  $c \in \mathcal{C}$ . This probability is used to define the final predicted class (the one with higher score), which is employed, with the ground-truth  $c_i$ , to estimate the error and optimize  $\mathcal{F}$ . After the optimization process, we use  $\mathcal{F}$  to predict classes for images in  $\mathcal{T}$ . Therefore, the final goal of the scene classification task is to predict, as accurately as possible, a class for

each image of  $\mathcal{T}$  using the trained function  $\mathcal{F}$ .

In fact, the objective of the proposed analysis is to elucidate and discuss some aspects of **ConvNets** which are necessary to take the most advantage of their benefits. Specifically, considering the remote sensing scene classification task, we carried out a systematic set of experiments to evaluate three different strategies: (i) fully-train **ConvNets**, (ii) fine-tuning **ConvNets**, and (iii) use pre-trained **ConvNets** as feature extractors.

In the first strategy, a (new or existing) network is trained from scratch obtaining specific features for the dataset. This approach is preferable since it gives full control of the architecture and parameters, which tends to yield a more robust and efficient network. However, as aforesaid, it requires a considerable amount of data [Bengio, 2009; Goodfellow et al., 2016], given its high number of parameters and proneness to overfitting. These drawbacks make almost impracticable to fully design and train efficient networks from scratch for most remote sensing problems, since large datasets in this domain are unusual given that training data may require high costs and complex logistics [Tuia et al., 2011; dos Santos et al., 2013; Almeida et al., 2014]. In an attempt to overcome the issue of few labeled data to train the network, data augmentation<sup>1</sup> techniques [Krizhevsky et al., 2012] can be employed. However, for small datasets even data augmentation is not enough to avoid overfitting [Bengio, 2009; Goodfellow et al., 2016].

Although remote sensing datasets have few labeled data, the computer vision area has several datasets composed of thousands, even millions, of pictures [Deng et al., 2009], being an intuitive choice when working with such techniques. Thus, the other two strategies rely on using pre-trained **ConvNets** on huge computer vision datasets to encode the features of remote sensing scenes. In spite of the aforementioned gap between this and the remote sensing domain (given its different image properties), these strategies benefit from the property that initial layers of **ConvNets** tend to be generic filters, like edge or color blob detectors, which are less dependent on the application and could be used in a myriad of tasks. In fact, [Razavian et al., 2014] suggest that features obtained from deep learning should be the primary candidate in most recognition tasks. Therefore, the second strategy uses a pre-trained **ConvNet** and performs fine-tuning of its parameters (filter weights) using the remote sensing data of interest. Particularly, this transfer-learning strategy tries to efficiently understand the difference between domains in order to adjust the feature representation to fill such gap and improve

---

<sup>1</sup>Data augmentation is a technique that artificially enlarges the training set with the addition of replicas of the training samples under certain types of transformations that preserve the class labels.



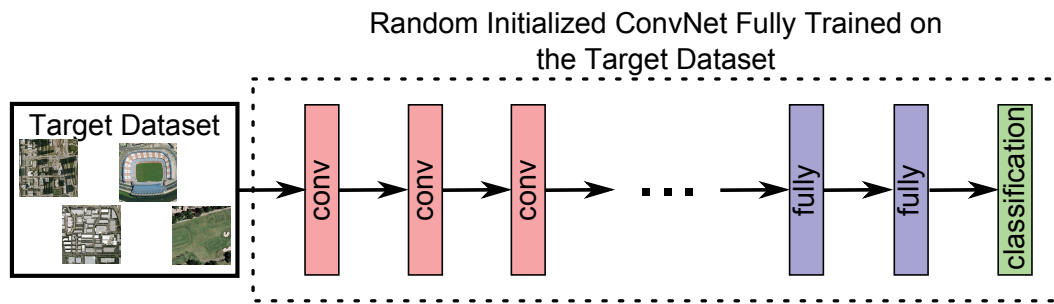
the final performance of the method. Commonly, in this case, the earlier layers are preserved, as they encode more generic features, and final layers are adjusted to encode specific features of the data of interest [Hinton et al., 2006; Larochelle et al., 2009]. The third strategy simply uses a pre-trained ConvNet as a feature extractor, by removing the last classification layer and considering its previous layer (or layers) as feature vector of the input data. In this case, an external machine learning algorithm (such as Support Vector Machine – SVM) is trained over these features and used in the classification process. In this strategy, it is assumed that the gap between domains is not so relevant and that the learned feature representation generalize for both domains, which is not guaranteed in the case of computer vision and remote sensing field.

## 5.1 Strategies for Exploiting ConvNets

This section aims at explaining the most common strategies of employing existing ConvNets in different scenarios from the ones they were trained for. As introduced, training a deep network from scratch requires a considerable amount of data as well as a lot of computational power. In many problems, few labeled data is available, therefore training a new network is a challenging task. Hence, it is common to use a pre-trained network either as a fixed feature extractor for the task of interest or as an initialization for fine-tuning the parameters. We describe these strategies, including their advantages and disadvantages, in the next subsections. Specifically, Section 5.1.1 describes about the full training of a new network while Section 5.1.2 presents the fine-tuning process. Finally, Section 5.1.3 explains the use of deep ConvNets as a fixed feature extractor.

### 5.1.1 Fully-trained Network

The strategy to train a network from scratch (with random initialization of the filter weights), which is illustrated in Figure 5.1, is the most common one, mainly in the computer vision domain. This may be justified by the fact that this process is useful when the dataset is large enough to make a network converge, presenting several advantages, including: (i) extractors tuned specifically for the dataset, which tend to generate more accurate features, and (ii) full control of the architecture. Therefore, fully training a ConvNet is expected to achieve better results when compared to other strategies, since the deep network learns specific features for the dataset of interest. However, this strategy requires a lot of computational and data resources [Bengio, 2009; Goodfellow et al., 2016], given its high number of parameters and convergence pruned



**Figure 5.1:** Illustrative example of a **ConvNet** being fully trained. Weights from the whole network are randomly initialized and then trained for the target dataset.

to overfitting. Also, the convergence of a **ConvNet** can not be totally confirmed (given its highly non-convex property) making tuning not so trivial.

In spite of these drawbacks, fully training is the first strategy to be considered when training **ConvNets**. There are basically two options for training a deep network that we can fit in the case of fully training. The first one is by fully designing and training a new architecture, including the number of layers, neurons, and type of activations, the number of iterations, weight decay, learning rate, etc. The other option is by using an existing architecture and fully training its weights to the target dataset. In this last case, the architecture and parameters (weight decay, learning rate, etc) are not modified at all. In this thesis, we evaluated both options. Specifically, we proposed and fully-trained a novel **ConvNet** architecture (PatreoNet) as well as consider the fully training of existing architectures in new (remote sensing) datasets.

## 5.1.2 Fine-tuned Network

When the new dataset is reasonably large, but not enough to fully train a new network, fine-tuning is a good option to extract the maximum effectiveness from pre-trained deep **ConvNets**, since it can significantly improve the performance of the final classifier. Fine-tuning is based on a curious property of modern deep neural network: they all tend to learn first-layer features that resemble either Gabor filters, edge or color blob detectors, independently of the training data. More specifically, earlier layers of a network contain generic features that should be useful to many tasks, but later layers become progressively more specific to the details of the classes contained in the original dataset (i.e., the dataset in which the deep **ConvNet** was originally trained). Supported by this property, the initial layers can be preserved while the final ones should be adjusted to suit the dataset of interest.

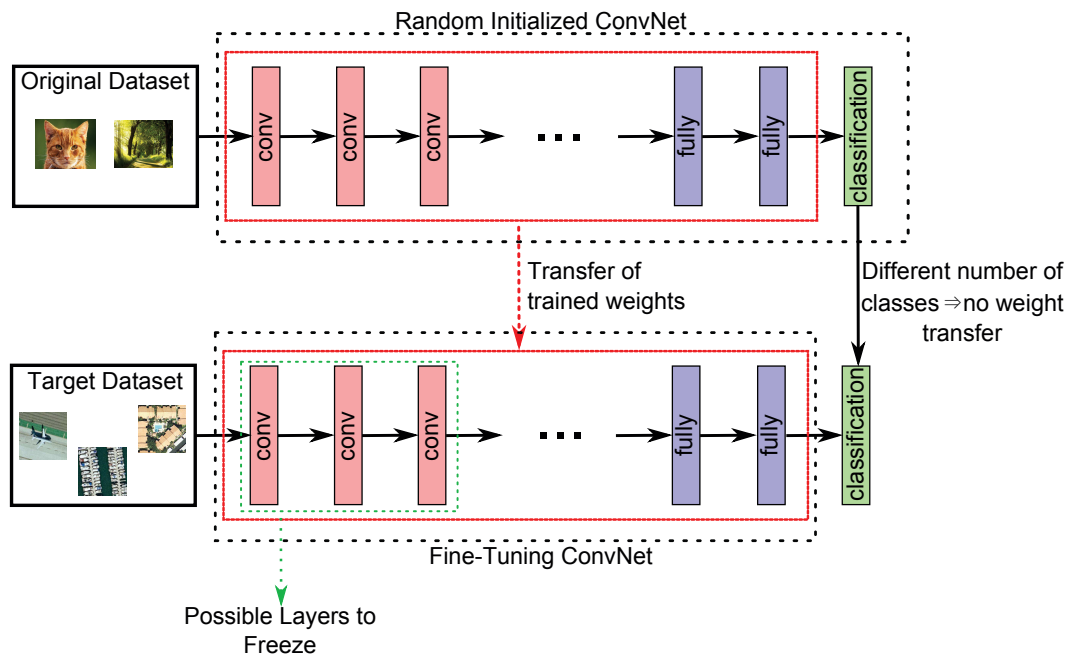
Fine-tuning is a transfer-learning technique that performs a fine adjustment in the parameters of a pre-trained network by resuming the training of the network from

a current set of parameters but considering a new dataset, aiming at accuracy improvements. In other words, fine-tuning uses the parameters learned from a previous training of the network on a specific dataset and, then, adjusts the parameters from the current state for the new dataset, improving the performance of the final classifier. It is expected that this adjustment process efficiently understands the difference between the dataset domains making the initial feature representation more close to the new dataset, filling the gap naturally created by different types of images.

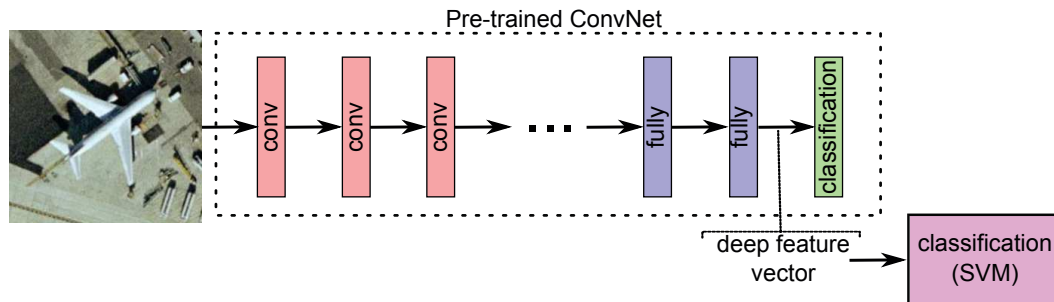
Based on the aforementioned characteristics, there are two possible approaches of performing fine-tuning in a pre-trained network, both exploited in this work: (i) fine-tune all layers, and (ii) fine-tune only higher-level layers keeping some of the earlier ones fixed (due to overfitting concerns). It is important to emphasize that in both scenarios, the search space is bounded to just small variations in each step since the learning rate is initialized with reduced value. Specifically, in the first case, some layers (usually the final ones, such as the classification layer, since the number of classes tends to be different) have weights ignored, being randomly initialized. These layers have the learning rate increased, so they can learn faster and converge, while the other layers may also change weights by very small variations since they use the reduced value of the learning rate without any augmentation. By doing this, the first layers can use the information previously learned with just a few adjustments to the dataset of interest, and at the same time, the final layers can really learn based only on the new dataset. In the second case, the initial layers are frozen to keep the generic features already learned, while the final layers are adjusted using the increased value of the learning rate. Both strategies of fine-tuning are analyzed and evaluated in this proposal considering the remote sensing datasets. These two options of fine-tuning are illustrated in Figure 5.2.

### 5.1.3 ConvNet as a Feature Extractor

A pre-trained network can be used as a feature extractor for any image, as illustrated in Figure 5.3. This is only possible since the generic features (learned in earlier layers) are less dependent on the final application and could be used in a myriad of tasks. Specifically, features (usually, called deep features) can be extracted from any layer of a pre-trained network and then used in a given task. Deep features trained on ImageNet [Deng et al., 2009] (a dataset of everyday objects) have already shown remarkable results in applications such as flower categorization [Sunderhauf et al., 2014], human attribute detection [Hara et al., 2016], bird sub-categorization [Ge et al., 2015], image scene retrieval [Babenko et al., 2014], and many others [Chatfield et al., 2014; Girshick et al., 2014], including remote sensing [Penatti et al., 2015; Hu et al., 2015; Cheng



**Figure 5.2:** Illustrative example of two options for the fine-tuning process. In one of them (highlighted in red), all layers are fine-tuned according to the target dataset, but final layers have increased learning rates. In the other option (highlighted in green), weights of initial layers can be frozen and only final layers are tuned.



**Figure 5.3:** Illustrative example of the use of a *ConvNet* as feature extractor. The final classification layer is ignored and one should only choose from which layer to consider the features. The figure shows the use of the features from the last layer before the classification layer, which is commonly used in the literature.

et al., 2016]. Though successful, the performance of deep features is highly dependent on its generalization as well as the proximity between the (trained and final) domains.

The strategy of using pre-trained *ConvNets* as feature extractors is very useful given its simplicity since no retraining or tuning is necessary. Moreover, one only needs to select the layer to be used, extract the deep features and use them combined with some machine learning technique, in case of a classification setup. According to previous works [Penatti et al., 2015; Razavian et al., 2014; Chatfield et al., 2014; Cheng et al., 2016], deep features can be extracted from the last layer before the classification layer (usually, a fully-connected one) and, then, used to train a classifier, which is the strategy employed in this project.

## 5.2 Specific Experimental Setup

In this section, we present the details about the experiments conducted to evaluate and analyze the different strategies to exploit *ConvNets*. Section 5.2.1 describes the low-level (global) and mid-level (*BoVW*) descriptors used while Section 5.2.2 presents the evaluated *ConvNets*. Finally, Section 5.2.3 presents details about the used experimental protocol.

### 5.2.1 Classical Feature Extraction Strategies

Several previously state-of-the-art descriptors have been selected to be evaluated, based on preceding successful works [Yang and Newsam, 2008; dos Santos et al., 2010; van de Sande et al., 2010; Penatti et al., 2012; Yang and Newsam, 2013; dos Santos et al., 2014], in which they were evaluated for remote sensing image classification, texture and color image retrieval/classification, and web image retrieval. Our selection includes simple global low-level descriptors, like descriptors based on color histograms and variations, and also descriptors based on Bags of Visual Words (*BoVW*) [Sivic and Zisserman, 2003].

#### 5.2.1.1 Low-Level descriptors

There is a myriad of descriptors available in the literature [Penatti et al., 2012] that can be used to represent image elements. Clearly, different descriptors may provide distinct information about images producing contrastive results. Thus, we tested a diverse set of 7 descriptors based on color, texture, and gradient properties, in order to extract visual features from each image and evaluate the potential of deep features when compared them. The global low-level descriptors considered are:

- Auto-Correlogram Color (*ACC*) [Huang et al., 1997], a color descriptor that aims to encode spatial color distribution in the image. Such information is extracted by computing the probabilities of having two pixels of color  $c_i$  in a distance  $d$  from each other.
- Border/Interior Pixel Classification (*BIC*) [de O. Stehling et al., 2002], a color descriptor that computes two color histograms for an image: one for border pixels and other for interior pixels. All pixels in an image are classified as border or interior, depending on the color of their neighbors: interior pixels have all of its four neighbors with the same color; if any of its neighbors are different, the pixel is classified as a border. One color histogram is computed for each type of pixel.

- Local Color Histogram (LCH) [Swain and Ballard, 1991], a simple color descriptor computed by dividing an image into tiles. An independent histogram is computed for each tile and then, they are concatenated to form the image feature vector.
- Local Activity Spectrum (LAS) [Tao and Dickinson, 2000], which captures texture spatial activity in four different directions separately: horizontal, vertical, diagonal, and anti-diagonal. The four activity measures are computed for a specific pixel by considering the values of neighboring in the four directions.
- Statistical Analysis of Structural Information (SASI) [Çarkacıoğlu and Yarman-Vural, 2003], a texture descriptor based on second order statistics of clique autocorrelation coefficients, which are the autocorrelation coefficients over a set of moving windows. These windows are covered in different ways varying size and shape, defining a neighborhood system.
- Histogram of Oriented Gradients (HOG) [Dalal and Triggs, 2005], which computes histograms of gradient orientations in each position of a sliding window, i.e., counts occurrences of gradient orientation in localized portions of an image.
- GIST [Oliva and Torralba, 2001] provides a global holistic description representing the dominant spatial structure of a scene. GIST is popularly used for scene representation [Douze et al., 2009].

The implementations of ACC, BIC, LCH, SASI, and LAS descriptors follow the specifications of [Penatti et al., 2012]. GIST implementation is the one used in [Douze et al., 2009] with the parameters discussed therein<sup>2</sup>. The implementation of HOG was obtained from the VLFeat framework [Vedaldi and Fulkerson, 2010]. We used HOG in different configurations, varying the cell size in  $14 \times 14$ ,  $20 \times 20$ ,  $40 \times 40$  and  $80 \times 80$  pixels, but keeping the orientation binning in 9 bins.

### 5.2.1.2 Mid-Level descriptors

Although simple to compute and good at providing a general idea of the image content, low-level global descriptors have some problems to encode details and low effectiveness in some precise applications, like recognition tasks for cluttered images [Tuytelaars and Mikolajczyk, 2007]. These have motivated the research community to develop more efficient and robust extraction algorithms. Towards this goal, researchers have developed Bag of Visual Words (BoVW), a mid-level feature extraction technique that

---

<sup>2</sup><http://lear.inrialpes.fr/software> (as of July 24th, 2017).

aims at transforming low-level local descriptors (such as SIFT [Lowe, 2004]) into a global and richer image representation of intermediate complexity [Boureau et al., 2010].

Specifically, a mid-level representation uses local features built upon low-level ones creating a new representation for an image, without looking for understanding its high-level features. According to [Boureau et al., 2010], in order to get the mid-level representation, the standard processing follows three steps: (i) low-level local feature extraction, (ii) coding, which, using a codebook of patches, performs a transformation of the descriptors into a representation better adapted to the task and (iii) pooling, which summarizes the coded features. Classification algorithms are then trained on the obtained mid-level vectors.

In fact, BoVW and their variations [Sivic and Zisserman, 2003; Lazebnik et al., 2006; Boureau et al., 2010; Perronnin et al., 2010; van Gemert et al., 2010; Penatti et al., 2014; Avila et al., 2013] are considered mid-level representations because of the aforementioned codebook of visual discriminating patches (visual words) that is employed to compute statistics about the visual word occurrences in the test images. This intermediate process transforms low-level local features into a more high-level and better-adapted representation, capable of extracting enhanced information about the images. BoVW descriptors have been state-of-the-art for several years in the computer vision community and are still important candidates to perform well in many tasks.

We tested BoVW in several different configurations:

- Sampling: sparse (Harris-Laplace detector [Mikolajczyk and Schmid, 2005]) or dense (grid of circles with 6 pixels of radius)
- Low-level descriptor: SIFT [Lowe, 2004] and OpponentSIFT [van de Sande et al., 2010]
- Visual codebooks of size: 100, 1000, 5000, and 10000
- Coding: hard or soft (with  $\sigma = 90$  or 150)
- Pooling: average, max pooling or WSA [Penatti et al., 2014]

To differentiate them in the experiments, we used the following naming:  $BX_{cp}^w$ , where  $X$  is S (sparse sampling) or D (dense sampling);  $w$  is the codebook size;  $c$  refers to the coding scheme used, h (hard), s (soft);  $p$  refers to the pooling technique used, a (average), m (max), or W (WSA).

The low-level feature extraction of BoVW descriptors was based on the implementation of [van de Sande et al., 2011]. For BoVW, in UCMerced and RS19 datasets,

**Table 5.1:** Some statistics about the deep networks evaluated in this work.

Networks	#parameters (millions)	#connections (millions)
OverFeat <sub>S</sub>	145	2,810
OverFeat <sub>L</sub>	144	5,369
AlexNet	60	630
CaffeNet	60	630
GoogLeNet	5	>10,000
VGG <sub>16</sub>	138	4,096
PatreoNet	15	200

we used SIFT [Lowe, 2004] to describe each patch, but in the Brazilian Coffee dataset, we used OpponentSIFT [van de Sande et al., 2010], as color should provide more discriminating power.

## 5.2.2 ConvNets

We selected some of the most popular *ConvNets* available nowadays to be evaluated in order to define a better strategy to exploit existing deep networks. All networks presented in this section<sup>3</sup>, except for the OverFeat ones, were implemented in Convolutional Architecture for Fast Feature Embedding [Jia et al., 2014], or simply Caffe, a fully open-source framework that affords clear and easy implementations of deep architectures. The OverFeat *ConvNets* were proposed and implemented by the namesake framework [Sermanet et al., 2014]. Furthermore, as aforesaid, all networks exploited in this work, for fine-tuning and feature extractors, were pre-trained on the ImageNet training set [Deng et al., 2009].

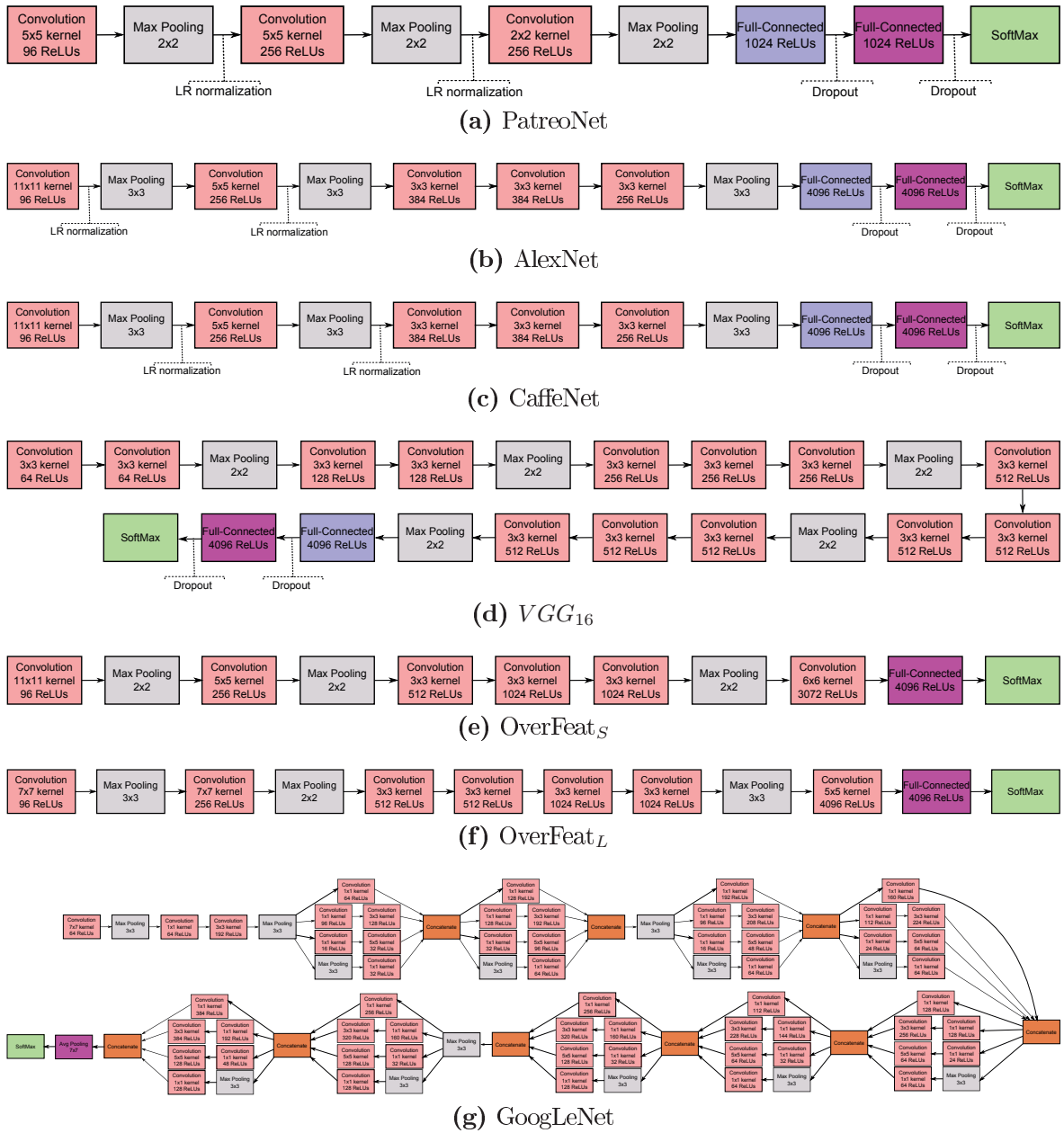
Some information about the networks evaluated in this work is presented in Table 5.1. GoogLeNet [Szegedy et al., 2015] is the biggest network, with higher number of connections, followed OverFeat<sub>L</sub> and VGG<sub>16</sub>. In fact, these networks, OverFeat and VGG<sub>16</sub>, are also the ones with a higher number of parameters, which require more memory during the training process. We describe some properties of each *ConvNet* in the following subsections.

### 5.2.2.1 PatreoNet

PatreoNet is a *ConvNet* that we proposed specifically for the remote sensing domain. Created based on recently published works and after a systematic set of experiments,

<sup>3</sup>Part of the code employed in this work has been made publicly available at <https://github.com/keillernogueira/PatreoNet>





**Figure 5.4:** Architectures of different the ConvNets evaluated in this work. Purple boxes indicate the layers from where features were extracted in the case of using the ConvNets as feature extractors.

this network has fewer layers and parameters being able to converge and learn specific spatial features using a small quantity of data, demonstrating the effectiveness of deep learning methods to encode features even for remote sensing scenario. This network, which architecture can be seen in Figure 5.4a, has 3 convolutional layers, 3 pooling ones and 3 fully-connected ones (considering the SoftMax), with (Local Response) normalization following the first and second pooling layers. All layers are composed of ReLUs, which were introduced in Section 3.1. PatreoNet was only used in fully-training strategy since it has no model pre-trained in large datasets.

### 5.2.2.2 AlexNet

**AlexNet**, proposed by [Krizhevsky et al., 2012], was the winner of ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [Deng et al., 2009] in 2012. This **ConvNet**, that has 60 million parameters and 650,000 neurons, consists of five convolutional layers, some of which are followed by max-pooling layers, and three fully-connected layers with a final softmax. Its final architecture can be seen in Figure 5.4b. It was a breakthrough work since it was the first to employ non-saturating neurons, GPU implementation of the convolution operation and dropout to prevent overfitting.

In the experiments, AlexNet was used as a feature extractor network by extracting the features from the last fully-connected layer (purple one in Figure 5.4b), which results in a feature vector of 4,096 dimensions. AlexNet was also fine-tuned for all the datasets used in the experiments. For UCMerced and RS19 datasets, the network was fine-tuned by giving more importance to the final softmax layer (without freezing any layer), while for the Coffee Dataset, the first three layers were frozen and the final ones participate normally in the learning process. This is due to the similarities of the former datasets and the difference of the latter dataset when compared to the ImageNet dataset [Deng et al., 2009]. Finally, AlexNet was fully trained from scratch for all datasets under the same configurations of the original model [Krizhevsky et al., 2012].

### 5.2.2.3 CaffeNet

**CaffeNet** [Jia et al., 2014] is almost a replication of AlexNet [Krizhevsky et al., 2012] with some important differences: (i) training has no relighting data-augmentation, and (ii) the order of pooling and normalization layers is switched (in CaffeNet, pooling is done before normalization). Thus, this network, which architecture can be seen in Figure 5.4c, has the same number of parameters, neurons, and layers of the AlexNet. Given its similarity to the AlexNet architecture, in the experiments, CaffeNet were exploited in the same way of the aforementioned network.

### 5.2.2.4 GoogLeNet

**GoogLeNet**, presented in [Szegedy et al., 2015], is the **ConvNet** architecture that won the ILSVRC-2014 competition (classification and detection tracks). Its main peculiarity is the use of “Inception” modules, which reduce the complexity of the expensive filters of traditional architectures allowing multiple filters, with different resolutions, to be used in parallel. GoogLeNet has two main advantages: (i) utilization of filters

of different sizes at the same layer, which maintains more spatial information, and (ii) reduction of the number of parameters of the network, making it less prone to overfitting and allowing it to be deeper. In fact, GoogLeNet has 12 times fewer parameters than AlexNet, i.e., approximately 5 millions of parameters. Specifically, the 22-layer GoogLeNet architecture, which can be seen in Figure 5.4g, has more than 50 convolutional layers distributed inside the “Inception” modules.

In our experiments, GoogLeNet was used as a feature extractor network by extracting the features from the last pooling layer (purple one in Figure 5.4g), which results in a feature vector of 1,024 dimensions. GoogLeNet was fine-tuned for all datasets just like AlexNet, with exactly the same strategies for each dataset. Finally, GoogLeNet was fully trained from scratch for all datasets under the same configurations of the original model [Szegedy et al., 2015].

### 5.2.2.5 VGG ConvNets

**VGG ConvNets**, presented in [Simonyan and Zisserman, 2014], won the localization and classification tracks of the ILSVRC-2014 competition. Several networks have been proposed in this work, but two have become more successful:  $VGG_{16}$  and  $VGG_{19}$ . Giving the similarity of both networks, we choose to work with the former one because of its simpler architecture and slightly better results. However, similar results obtained by  $VGG_{16}$  should be also yielded by  $VGG_{19}$ . This network, which architecture can be seen in Figure 5.4d, has 13 convolutional layers, 5 pooling ones and 3 fully-connected ones (considering the softmax).

$VGG_{16}$  was used as a feature extractor network by extracting the features from the last fully-connected layer (purple one in Figure 5.4d), which results in a feature vector of 4,096 dimensions.  $VGG_{16}$  was also fine-tuned in the UCMerced and RS19 datasets by giving more importance to the final softmax layer, without freezing any layer. However, this network could not be fine-tuned for the Brazilian Coffee dataset (the most different one) as well as could not be fully trained from scratch for any dataset. This problem is due to the large amount of memory required by this network, as presented in Table 5.1, which allows only small values of batch size to be used during the training process. Since larger values of batch size, combined with other parameters (weight decay, learning rate), help the convergence of a ConvNet in the training process [Goodfellow et al., 2016; Bengio, 2009], there was no convergence in the aforementioned scenarios.

### 5.2.2.6 OverFeat ConvNets

**OverFeat** [Sermanet et al., 2014], a deep learning framework focused on ConvNets and winner of the detection track of ILSVRC 2013, has two ConvNet models available, which can be used to extract features and/or to classify images. There is a small (*fast* – OverFeat<sub>S</sub>) and a larger network (*accurate* – OverFeat<sub>L</sub>), both having similarities with AlexNet [Krizhevsky et al., 2012]. The main differences are: (i) no response normalization, and (ii) non-overlapping pooling regions. OverFeat<sub>L</sub>, which architecture can be seen in Figure 5.4f, has more differences including: (i) one more convolutional layer and, (ii) the number and size of feature maps, since a different number of kernels and stride were used for the convolutional and the pooling layers. In the other way around, OverFeat<sub>S</sub>, which architecture can be seen in Figure 5.4e, is more similar to the AlexNet, differing only in the number and size of feature maps. The main differences between the two OverFeat networks are the stride of the first convolution, the number of stages and the number of feature maps [Sermanet et al., 2014].

These ConvNets are only used as feature extractor since no model was provided in order to perform fine-tuning or fully-training of the networks. Considering this strategy, a feature vector of 4,096 dimensions is obtained from the last fully-connected layer, which are illustrated as purple layers in Figures 5.4e and 5.4f, for OverFeat<sub>S</sub> and OverFeat<sub>L</sub>, respectively.

## 5.2.3 Protocol

We carried out all experiments with a 5-fold cross-validation protocol, as introduced in Section 4.2, while results are reported in terms of average accuracy and standard deviation among the 5 folds, as presented in Section 4.3.

When performing fine-tuning or training a network for scratch, at each run, three folds are used as training set, one as validation (used to tune the parameters of the network) and the remaining one is used as the test set. It is important to mention that the full training or fine-tuning of a network starts from the beginning for every new iteration of the 5-fold cross-validation strategy. Five different networks are obtained, one for each step of the 5-fold cross-validation process. That is, there is no contamination of the training set with testing data. When using the ConvNets as feature extractors, four sets are used as training while the last is the test set. Still considering this strategy, we always used linear SVM as the final classifier.

When fine-tuning or full training a network, we basically preserve the parameters of the original author, varying only two according to Table 5.2. It is important to

**Table 5.2:** Parameters utilized in fine-tuning and full-training strategies.

Strategy	#iterations	Learning Rate
<b>Fine-tuning</b>	20,000	0.001
<b>Full-training</b>	50,000	0.01

highlight that there is no training when using a pre-trained **ConvNet** (without fine-tuning) as feature extractor, thus there are no parameters to analyze.

All computational experiments were performed on a 64 bits Intel i7 4960X machine with 3.6GHz of clock and 64GB of RAM memory. Two GPUs were used: a GeForce GTX770 with 4GB of internal memory and a GeForce GTX Titan X with 12GB of memory, both under a 7.5 CUDA version. Ubuntu version 14.04.3 LTS was used as operating system.

## 5.3 Results and Discussion

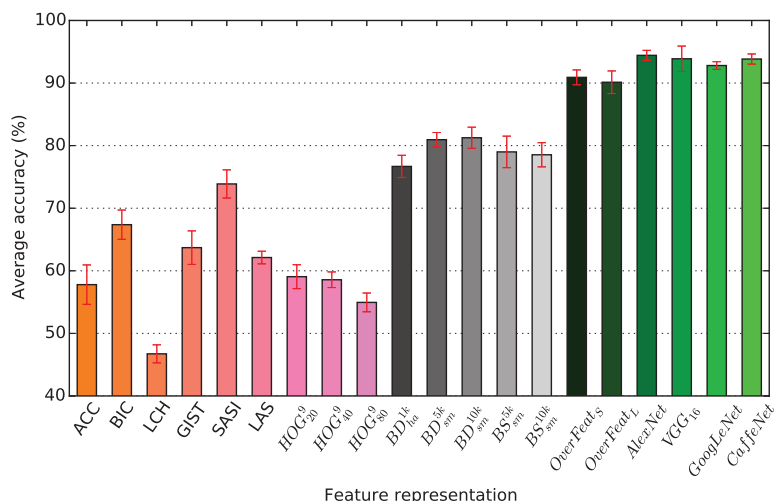
In this section, we present and discuss the experimental results related to the use of different strategies to exploit **ConvNets** in the remote sensing scenario. Firstly, in Section 5.3.1, we discuss the power of generalization of **ConvNets** as feature descriptors and compare them with low-level and mid-level representations. Then, we compare the performance of the three different strategies for exploiting the existing **ConvNets** in Section 5.3.2. Finally, Section 5.3.3 compare the most accurate **ConvNets** against some of the state-of-the-art methods for each dataset.

### 5.3.1 Generalization Power Evaluation

In this first section, we compare six pre-trained **ConvNets** used as descriptors against low-level and mid-level representations for aerial and remote sensing scene classification. We conducted several experiments in order to evaluate the best **BoVW** configurations, but only the top-5 best were reported. It is also important to highlight that the **ConvNets** results in this section refer to their use as feature extractors, by considering the features of the last layer before softmax as input for a linear SVM. So, the original network was not used to classify the samples. The objective is to observe how deep features perform in datasets from different domains they were trained, i.e., how good is the generalization power of this learned representation. As introduced, this is very important analysis because, since remote sensing domain does not have lots of

labeled data available, a feature representation, learned on huge but distinct sort of datasets, may be useful.

In Figure 5.5, we show the average accuracy of each descriptor and *ConvNet* for the UCMerced dataset. We can notice that *ConvNet* features achieve the highest accuracy rates ( $\geq 90\%$ ). CaffeNet, AlexNet, and *VGG*<sub>16</sub> yield the highest average accuracies (more than 93%). GoogLeNet achieves  $92.80\% \pm 0.61$  and OverFeat achieves  $90.91\% \pm 1.19$  for the small and  $90.13\% \pm 1.81$  for the large network. SASI is the best global descriptor ( $73.88\% \pm 2.25$ ), while the best BoVW configurations are based on dense sampling, 5 or 10 thousand visual words and soft assignment with max pooling ( $\sim 81\%$ ).

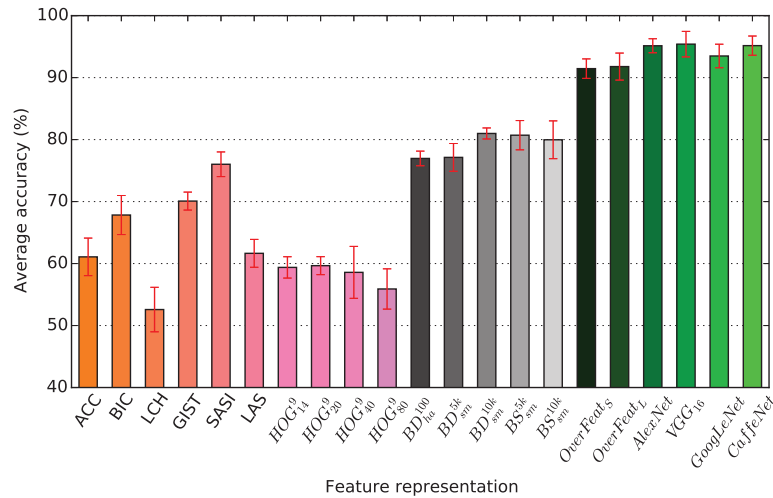


**Figure 5.5:** Average accuracy of pre-trained *ConvNets* used as feature extractors and low- and mid-level descriptors for the UCMerced Land-use Dataset.

In Figure 5.6, we show the average accuracies for the RS19 dataset. *ConvNets* again achieved the best results ( $\geq 90\%$ ). The best global descriptor was again SASI and the best BoVW configurations have 5 or 10 thousand visual words with soft assignment and max-pooling, but in this dataset, sparse sampling also achieved similar accuracies to dense sampling.

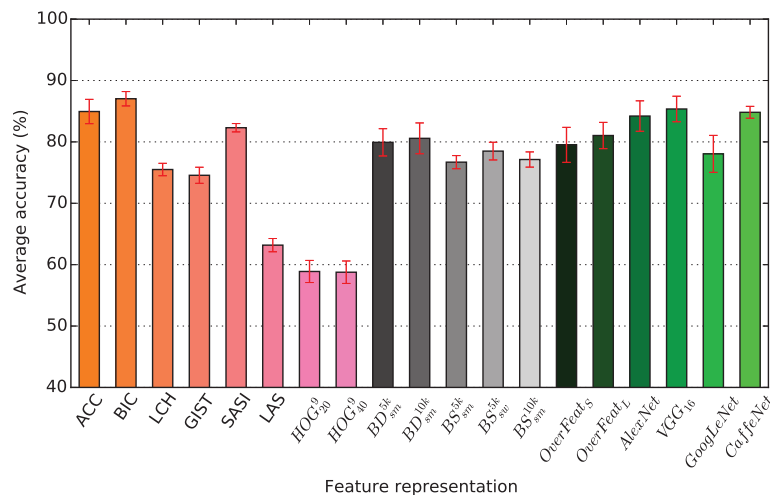
The results with UCMerced and RS19 datasets illustrate the capacity of *ConvNet* features to generalize from everyday pictures to the aerial domain.

In Figure 5.7, we show the average accuracies for the Brazilian Coffee Scenes dataset. In this dataset, the results are different than in the other two datasets already mentioned. We can see that, although most of the *ConvNets* achieve accuracy rates above 80%, with *VGG*<sub>16</sub> achieving  $85.36\% \pm 2.08$ , BIC and ACC also achieved high accuracies. BIC achieved the highest average accuracy ( $87.03\% \pm 1.17$ ) outperforming all the descriptors including the *ConvNets* in this dataset. The BIC algorithm for



**Figure 5.6:** Average accuracy of pre-trained *ConvNets* used as feature extractors and low- and mid-level descriptors for the RS19 Dataset.

classifying pixels in border or interior basically separates the images into homogeneous and textured regions. Then, a color histogram is computed for each type of pixel. As for the Brazilian Coffee Scenes dataset, the differences between classes may be not only in texture but also in color properties, *BIC* could encode well such differences. The best *BoVW* configurations are again based on dense sampling, 5 or 10 thousand visual words and soft assignment with max pooling, and they have comparable results to *OverFeat*.



**Figure 5.7:** Average accuracy of pre-trained *ConvNets* used as feature extractor and low- and mid-level descriptors for the Brazilian Coffee Scenes Dataset.

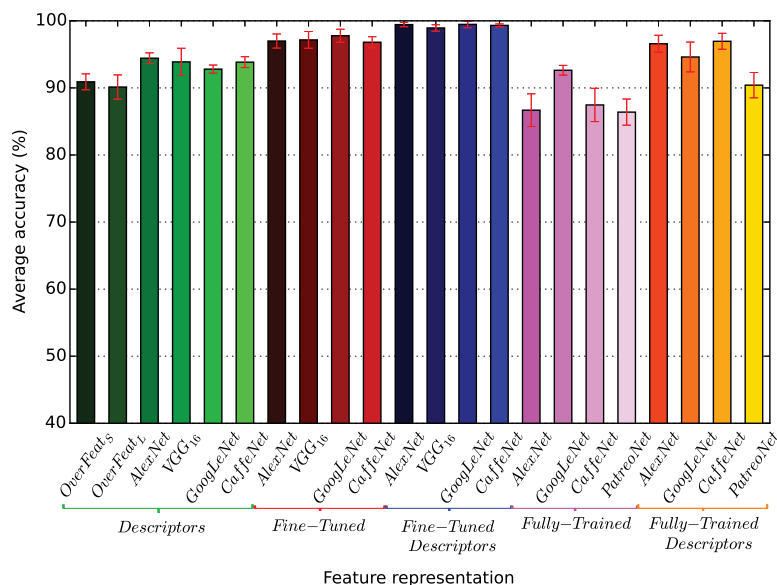
A possible reason for the deep features to perform better in the aerial dataset than in the agricultural one is due to the particular intrinsic properties of each dataset. The aerial datasets have more complex scenes, composed of a lot of small objects (e.g.,

buildings, cars, airplanes). Many of these objects are composed of similar patterns in comparison with the ones found in the dataset used to train the *ConvNets*.

Concerning the Brazilian Coffee Scenes dataset, it is composed of finer and more homogeneous textures where the patterns are much more overlapping visually and more different than everyday objects. The color/spectral properties are also important in this dataset, which fit with results reported in other works [dos Santos et al., 2014; Faria et al., 2014].

### 5.3.2 Comparison of ConvNets Strategies

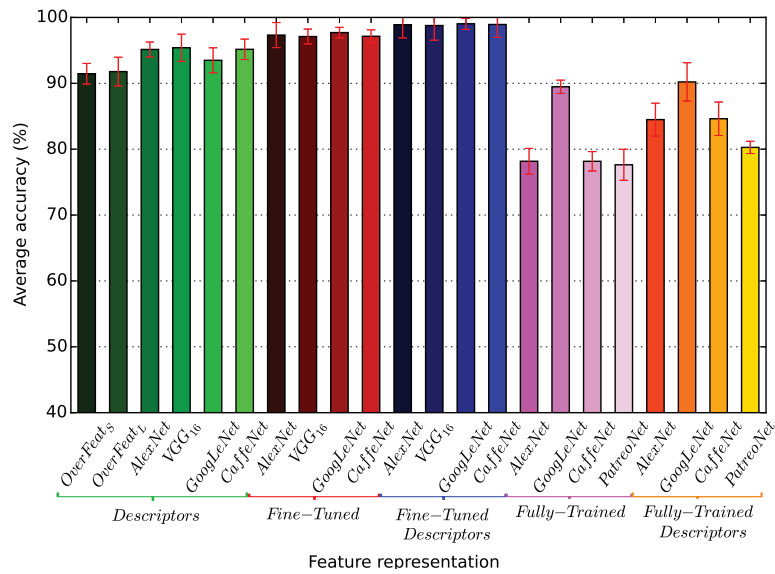
In this section, we compare the performance of the three different strategies for exploiting the existing *ConvNets*: full training, fine-tuning, and using as feature extractors. Figures 5.8 to 5.10 show the comparison of the strategies in terms of average classification accuracy. In such figures, the suffix “Descriptors” refers to when we use the features from the last layer before the softmax layer as input for another classifier, which was a linear SVM in our case. However, it is worth mentioning that SVM was used only after the fine-tuning or the fully-training process, not during the training process.



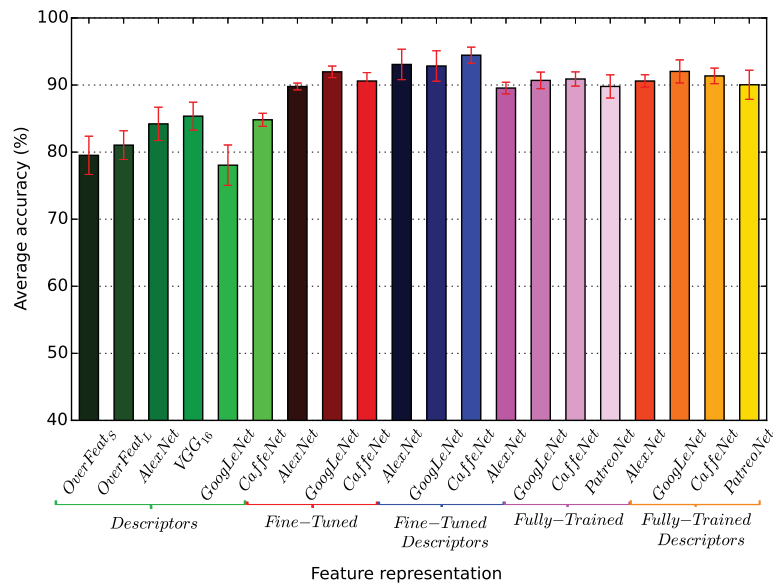
**Figure 5.8:** Average accuracy considering all possible strategies to exploit *ConvNets* for the UCMerced Land-use Dataset.

There are several interesting aspects illustrated in the graphs. The first one is that fine-tuning (red and blue bars) is usually the best strategy, outperforming the other two in all the datasets. The difference was higher for UCMerced and RS19 datasets





**Figure 5.9:** Average accuracy considering all possible strategies to exploit ConvNets for the RS19 Dataset.

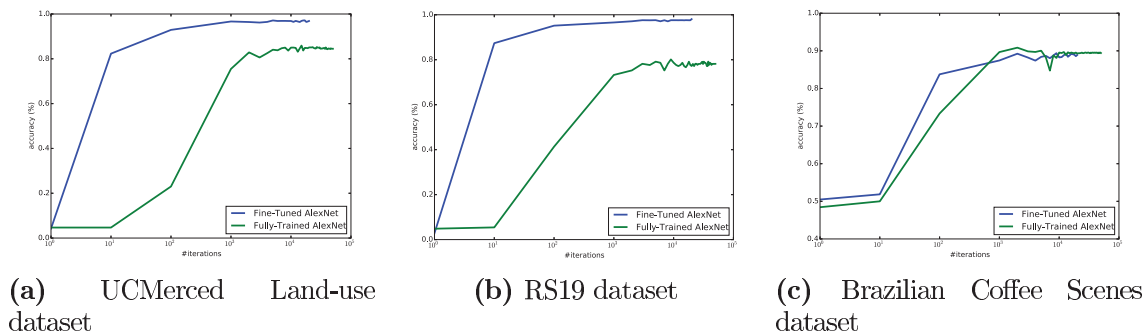


**Figure 5.10:** Average accuracy considering all possible strategies to exploit ConvNets for the Brazilian Coffee Scenes Dataset.

(Figures 5.8 and 5.9). For the Coffee Scenes dataset, this difference is small, however, the Fine-Tuned Descriptors (blue bars) were slightly superior.

This advantage of fine-tuned networks, when compared to the fully-trained ones, is maybe due to a better initialization in the search space. This can be noticed in Figure 5.11, where even with more iterations, fully-trained networks stick in worse local minimum than the fine-tuned ones, what demonstrates that a better initialization of the filter weights tends to provide better results. Another way to explain these results is based on the fact this strategy uses the initialization to really understand and fill

the existing gap between the different domains (computer vision and remote sensing ones). Specifically, this strategy can use a specific initialization to understand how the current representation is distinct from the requested one (which is based on the new dataset), adjusting the representation and improving the final accuracy.



**Figure 5.11:** Examples of convergence of AlexNet for all datasets, considering Fold 1.

Other aspect to highlight is that full training (orange bars) was not a good strategy for datasets UCMerced and RS19. For RS19 dataset especially, there was a drop in accuracy in relation even to the original feature extractors (green bars), which were trained on ImageNet. This may be justified by the small amount of labeled data available in these datasets, which makes the networks incapable of learning good feature representation. However, for Coffee Scenes dataset, the fully-training strategy improved results in comparison with other ones. Though this dataset also has a limited amount of labeled data, results may be explained by the fact that it has very distinct properties from the original dataset (ImageNet) used to pre-train the networks. Thus, in this case, training a ConvNet from scratch is a better strategy to learn the feature representation.

Another aspect of the results is that replacing the last softmax layer by SVM on almost every ConvNet was a better solution. The Fine-Tuned Descriptors (blue bars) and Fully-Trained Descriptors (orange bars) were usually superior than their counterparts with the softmax layer (red and pink bars, respectively). In the Coffee Scenes, however, this difference was smaller.

Comparing the different ConvNets, we can see that their results are very similar in all the datasets. GoogLeNet seems to be less affected by the full training process, as their results decreased less than the other ConvNets when comparing the Fine-Tuned and the Fully-Trained versions. One possible reason is that GoogLeNet has fewer parameters to be learned and, as the datasets used are very small considering the requirements of the full training process, GoogLeNet was less affected.

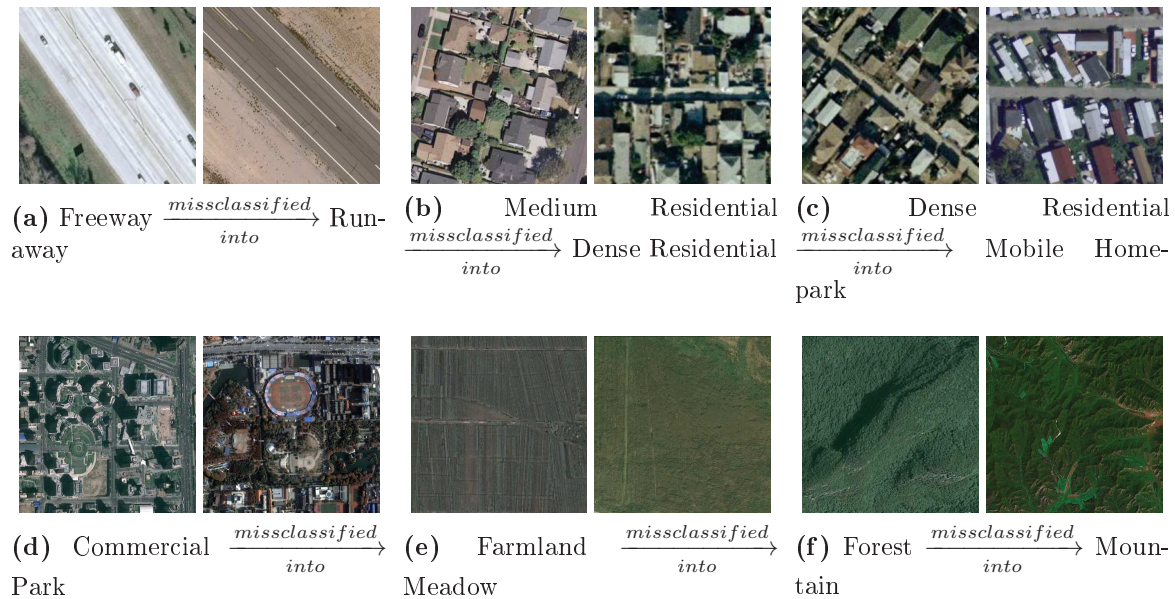
Comparing the results of the **ConvNets** as feature extractors (green bars) in relation to fine-tuning (red and blue bars), we can see that the original feature extractors, trained on ImageNet, are not too worse than the fine-tuned version of the **ConvNets**, especially for the UCMerced and RS19 datasets. The reason is that in such datasets, the edges and local structures of the images are more similar to everyday objects than in the Coffee Scenes dataset, in which the difference was higher in favor of fine-tuned **ConvNets**. In the Coffee Scenes dataset, the textures and local structures are very different than everyday objects.

Comparing the results among the three datasets, we can see that the Coffee Scenes dataset has different behavior for the **ConvNets**. Fully-trained networks achieve better accuracy in this dataset than in the others. This maybe motivated by the huge difference between the datasets, since UCMerced and RS19 datasets are aerial ones while Coffee Scenes is a multi-spectral one.

Considering the computational load, using a pre-trained **ConvNet** as feature extractor is the most efficient strategy since no training over the network is required. The fine-tuning strategy is less efficient since requires a little training over the parameters of the network, but this method, generally, yields better results. The strategy that needs more computational requirements is full training since a network is trained from scratch (with randomly initialized weights and bias), requiring more time and resources to be trained. Specifically for these two last strategies, in our experiments, five **ConvNets** are trained (one for each fold), which makes the process time-consuming. Considering the fine-tuning technique, the whole process takes around 5 hours to be completed, while the fully-trained strategy takes, approximately, 9 hours. In both cases, we consider an average training time of all three datasets on the GeForce GTX Titan X.

As a summary, we can recommend fine-tuning as the strategy that tends to be more promising in different situations. In addition, fine-tuning is less costly than full training, which can represent another advantage when efficiency is a constraint. On top of that, we can also recommend the use of the features extracted from the last layer of the fine-tuned network and then using SVM for the classification task, instead of using the softmax layer.

As shown in the experimental results, the best **ConvNet** configurations classify almost 100% of the aerial images (UCMerced and RS19). Notwithstanding, the wrong classified images are really difficult, as can be noted in the examples shown in Figure 5.12. Notice how these misclassified samples are quite similar visually.



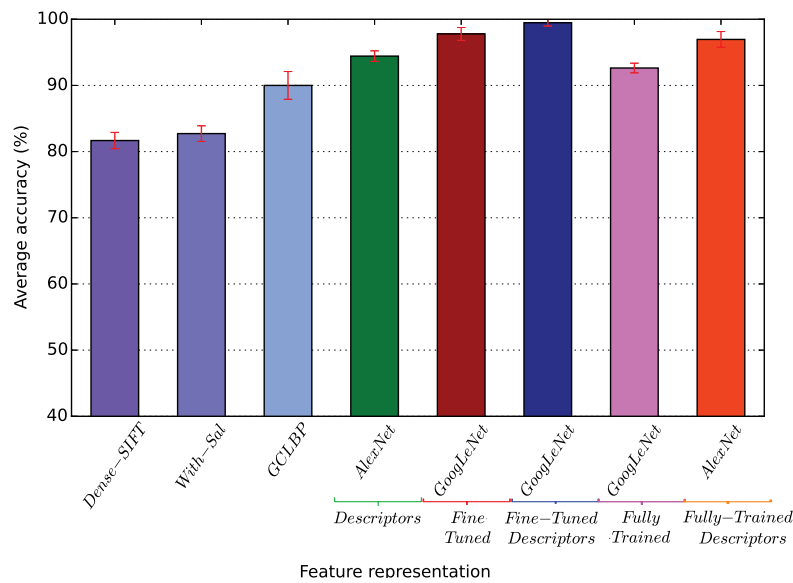
**Figure 5.12:** Three examples of wrong predictions of each aerial dataset, UCMerced and RS19, for a fine-tuned AlexNet. (a)-(f) The first image is the misclassified one, while the second is a sample of the predicted class. Notice the similarity between the classes.

### 5.3.3 Comparison with Baselines

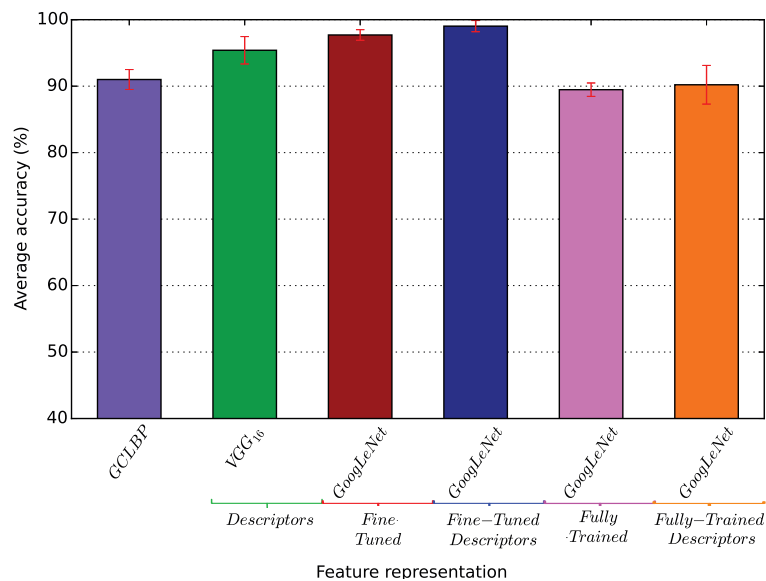
In this section, we compare the performance of the best results of each strategy for exploiting the existing *ConvNets* and state-of-the-art baselines. Figures 5.13 to 5.15 show the comparison in terms of average accuracy. As in the previous section, the suffix “Descriptors” specify when *ConvNets* were used as feature extractor, being the deep features classified with linear SVM.

For the UCMerced dataset, we select three state-of-the-art baselines: (i) GCLBP [Chen et al., 2015], (ii) With-Sal [Zhang et al., 2015], and (iii) DenseSIFT [Cheriyadat, 2014]. The results presented in Figure 5.13 show that the baselines were outperformed by all strategies, except the full training one. Furthermore, the classification of deep features extracted from the fine-tuned GoogLeNet with linear SVM achieve the best result of all ( $99.47\% \pm 0.50$ ), being closely followed by the fine-tuned GoogLeNet, which yielded  $97.78\% \pm 0.97$ , in terms of average accuracy.

For the RS19 dataset, we compare the best results of each strategy to the GCLBP [Chen et al., 2015] approach, which yielded  $91.0\% \pm 1.5$ . The outcomes presented in Figure 5.14 confirm the results obtained in the UCMerced dataset since these two datasets are very similar: using linear SVM to classify deep features extracted from a fine-tuned GoogLeNet yielded the best result. Furthermore, analogous to the previous dataset, the fully-trained network did not outperform the baseline, being statistically similar.



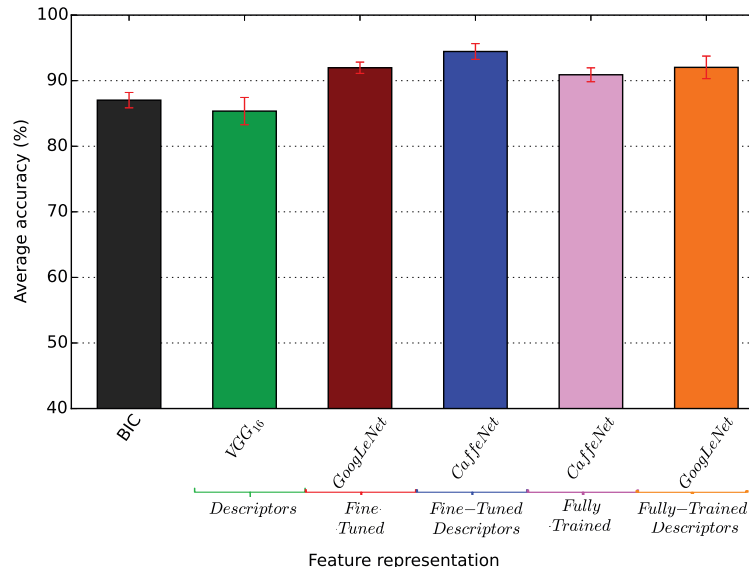
**Figure 5.13:** Comparison between state-of-the-art baselines and the best results of each strategy to exploit *ConvNets* for the UCMerced Land-use Dataset. The Fine-Tuned Descriptors extracted by GoogLeNet achieved the highest accuracy rates.



**Figure 5.14:** Comparison between state-of-the-art baselines and the best results of each strategy to exploit *ConvNets* for the RS19 Dataset. The Fine-Tuned Descriptors extracted by GoogLeNet achieved the highest accuracy rates.

For the Brazilian Coffee Scenes dataset, the only state-of-the-art result available is the one which was released with the dataset in our previous work [Penatti et al., 2015], using the *BIC* descriptor, that we also present here in Section 5.3.1. Now, the best result for this dataset and current state-of-the-art is achieved by extracting deep features from the fine-tuned CaffeNet ( $94.45\% \pm 1.20$ ), as presented in Figure 5.15. Note that although *BIC* outperforms the *ConvNet* used as a descriptor, it is not true for

the Fully-Trained and Fine-Tuned. It means that we can adjust the domain by using a fully-trained *ConvNet*. However, by using parameters obtained in other domain is useful to yield even better results.



**Figure 5.15:** Comparison between state-of-the-art baselines and the best results of each strategy to exploit *ConvNets* for the Brazilian Coffee Scenes Dataset. The Fine-Tuned Descriptors extracted by CaffeNet achieved the highest accuracy rates.

## 5.4 Conclusions

We evaluated three strategies for exploiting existing *ConvNets* in different scenarios from the ones they were trained. All this evaluation has the following objectives: (i) to verify the generalization power of deep features, (ii) to analyze how well fine-tuning strategy benefits from a better initialization of the filter weights, how it understands the difference of domains, and how it adjusts its feature representation, and (iii) to understand the best way to obtain the most benefits from these state-of-the-art deep learning approaches in existing applications, in which there are few labeled data. We performed experiments evaluating the following strategies for exploiting the *ConvNets*: full training, fine-tuning, and using as feature extractors. The experiments considered six popular *ConvNets* (OverFeat networks [Sermanet et al., 2014], AlexNet [Krizhevsky et al., 2012], CaffeNet [Jia et al., 2014], GoogLeNet [Szegedy et al., 2015], VGG<sub>16</sub> [Simonyan and Zisserman, 2014], and PatreNet) in three remote sensing datasets. It is important to emphasize that the Brazilian Coffee Scenes dataset was created specifically for this analysis [Penatti et al., 2015], since, in general, there is a lack of agriculture dataset.

The results pointed out that deep features of the existing pre-trained **ConvNets** generalize to other domains, mainly when there are similarities between them (case of UCMerced and RS19 datasets). Also, results demonstrated that fine-tuning strategy can use a better initialization to improve the final performance of the networks. In other words, this strategy, somehow, learns the difference of domains and adjusts its feature representation into one closer to the target dataset. Furthermore, fine tuning tends to be the best strategy in different situations as demonstrated by the experimental results. Specially, using the features of the fine-tuned network with an external classifier, linear SVM in our case, provides the best results. It is important to note that, although some of the conclusions above seem obvious nowadays, they were not so clear when this pioneer work was conducted.

As additional contributions, we can point the evaluation of different **ConvNets** in each strategy mentioned in three remote sensing datasets, comparing their results with traditional low- and mid-level descriptors, as well as with state-of-the-art baselines of each dataset. And finally, we obtained state-of-the-art results in the three datasets used (UCMerced land use, RS19, and Brazilian Coffee Scenes).





## Chapter 6

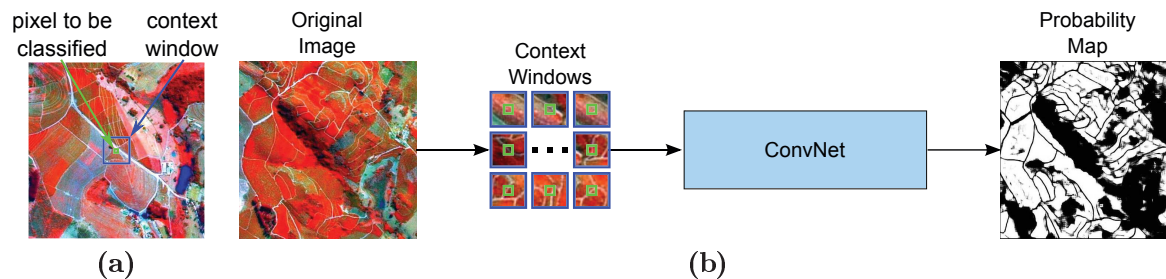
# ConvNet-Based Pixel Classification

Fully scene understanding is a primary task in a wide range of applications and one of the most important in the remote sensing community. This task is strongly based on the creation of thematic maps which, in turn, are essentially modeled in a supervised manner that should have, as outcome, a class for each and every pixel of the input image. Based on its outcome, this process is commonly called pixel classification (also known, in the computer vision field, as semantic segmentation) [Wilkinson, 2005; Nogueira et al., 2016a; Volpi and Tuia, 2017].

**Definition 2. Pixel Classification.** *Let  $\mathcal{D}$  and  $\mathcal{T}$  be the training and testing sets. The training set  $\mathcal{D}$  consists of tuples of the form  $\{p_i, c_i\}$ , where  $p_i$  is a **pixel** and  $c_i$  its respective class. Note that  $i \in \{1, \dots, |\mathcal{D}|\}$  and  $c_i \in \mathcal{C}$ , where  $\mathcal{C} = \{C_1, C_2, \dots, C_k\}$  is the set of the  $k$  possible classes. Distinctly, the testing set  $\mathcal{T}$  is only composed of pixels  $p_j$  (with  $j \in \{1, \dots, |\mathcal{T}|\}$ ).*

*$\mathcal{D}$  is used to perform a supervised training of a function  $\mathcal{F}$  that receives as input a pixel  $p_i$  and outputs a membership probability  $p(c|p_i)$  for each class  $c \in \mathcal{C}$ . This probability is used to define the final predicted class (the one with higher score), which is employed, with the ground-truth  $c_i$ , to estimate the error and optimize  $\mathcal{F}$ . After the optimization process, we used  $\mathcal{F}$  to predict classes for pixels in  $\mathcal{T}$ . Therefore, the final goal of the pixel classification task is to predict, as accurately as possible, a class for each pixel of  $\mathcal{T}$  using the trained function  $\mathcal{F}$ , allowing the creation of the prediction (thematic) map.*

This process produces essential and useful information capable of assisting in the decision making of a wide range of areas, including crop and forest management [dos Santos et al., 2012; Nogueira et al., 2015b; Santana et al., 2017], disaster relief [Fustes



**Figure 6.1:** (a) Example of a context window. The pattern is represented by a large window that is centered on the pixel of interest in order to include the context of its neighborhood. (b) The process performed by a pixelwise network. A set of context windows are generated for each pixel and then classified by the network. The predicted class for the context window is actually the label of the centered pixel.

et al., 2014; Nogueira et al., 2018], urban planning [Volpi and Ferrari, 2015; Volpi and Tuia, 2017], phenological studies [Nogueira et al., 2019b], and so on.

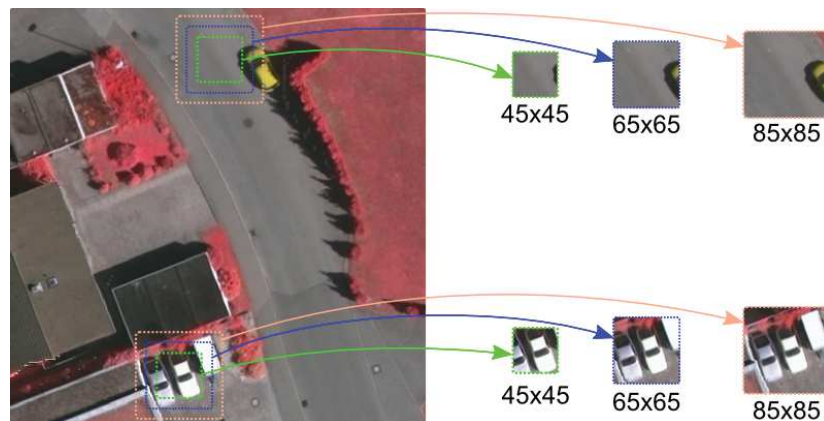
Because of its usefulness and relevance, the development of pixel classification algorithms is still considered an open and hot research topic in the remote sensing community [Benediktsson et al., 2013; Ma et al., 2015; Ball et al., 2017]. In fact, several strategies have been proposed over the years to perform pixel classification. Among all approaches, deep learning [Bengio, 2009; Goodfellow et al., 2016] is the current state-of-the-art [Long et al., 2015; Noh et al., 2015; Badrinarayanan et al., 2017; Wang et al., 2017; Peng et al., 2019; Nogueira et al., 2019b] for this task, mainly due to its feature learning concept. There are two basic techniques based on deep learning for pixel classification. The first one receives as input context windows (Figure 6.1a) and outputs a class (that is, in fact, associated with the centered pixel). This approach, presented in Figure 6.1b, is computationally inefficient, given that each pixel must be classified independently. The second technique improves the first strategy by solving this computational problem. Particularly, it solves this issue by adapting the *ConvNets* to output a dense prediction, i.e., to produce another image (usually with the same resolution of the input) that has each pixel associated to a semantic class. In this case, the network receives as input an image and outputs another image (with the same resolution of the input) with all pixels classified. Differently from aforesaid approaches that exploit the pixel signature to perform the classification, such deep learning-based approaches try to improve the performance by exploiting the pixel context, which is, in these cases, directly connected to the input image.

This input (and, consequently, the context) is actually represented by a patch, in the remote sensing domain. This is due to the fact that *RSIs* have a huge size (when compared to everyday photos) and therefore can not be processed directly (due to computational constraints). Because of this, as aforementioned, the *RSIs* are usually

divided into fixed-size overlapping patches, which delimit the context that may be exploited by the algorithms. Due to this, the definition of the best input patch size is of vital importance for the network, given that patches of small size (or context) could not bring enough information to allow the network to capture the patterns while, larger patches could lead to semantically mixed information, which could affect the performance of the *ConvNet*. In the literature, the definition of this patch size is usually performed using two strategies: (i) empirically [Sherrah, 2016; Volpi and Tuia, 2017], by evaluating several sizes and selecting the best one, which is a very expensive process, given that, for each size, a new network must be trained (without any guarantee for the best patch configuration), and (ii) imposed [Audebert et al., 2016; Marmanis et al., 2018], in which the patch size is defined by network constraints (i.e., changing the patch size implies modifying the architecture). This could be a potentially serious limitation given that the patch size required by the network could be not even close to the optimal one. Hence, it is clear that both current strategies suffer from drawbacks and could not lead to the best patch size.

An attempt to alleviate such dependence of the patch size is to aggregate multi-context information. Multi-context paradigm has been proven to be essential for segmentation methods [dos Santos et al., 2012; Sherrah, 2016], given that it allows the model to extract and capture patterns of varying granularities, helping the method to aggregate more useful information. Precisely, as presented and explained in (caption of) Figure 6.2, smaller contexts may be preferable in some situations while larger ones can be useful in other scenarios. Therefore, several works [Marcu and Leordeanu, 2016; Audebert et al., 2016; Maggiori et al., 2017; Marmanis et al., 2018; Paisitkriangkrai et al., 2016; Wang et al., 2017] incorporate the benefits of the multi-context paradigm in their architectures using different approaches. Some of them [Marcu and Leordeanu, 2016; Paisitkriangkrai et al., 2016; Audebert et al., 2016] train several distinct layers or networks, one for each context, and combine them for the final prediction. Others [Marmanis et al., 2018; Maggiori et al., 2017; Wang et al., 2017] extract and merge features from distinct layers in order to aggregate multi-context information. Independently of the approach, to aggregate multi-context information, more parameters are included in the final model, resulting in a more complex learning process [Goodfellow et al., 2016].

In this chapter, we propose a novel technique to perform semantic segmentation of RSIs that exploits the multi-context paradigm without increasing the number of parameters while defining adaptively the best patch size for the inference stage. Specifically, this technique is based upon an architecture composed exclusively on dilated convolutions [Yu and Koltun, 2015], which are capable of processing input patch



**Figure 6.2:** Example showing the importance of multi-context information. In the top case, while smaller contexts may not provide enough information for the understanding of the scene, a large context brings more information that may help the model to identify that it is a road with a car on it. In the bottom scenario, smaller contexts bring enough information for the identification of cars, while a large context may confuse the network and lead it to misclassify a different object as a car.

of varying sizes without distinction, given that they learn the patterns without down-sampling the input. In fact, the multi-context information is aggregated to the model by allowing it to be trained using patches of varying sizes (and contexts), a process that increases scale-invariance and reduces over-fitting [He et al., 2015]. This procedure allows the extraction of multi-context information without any combination of distinct networks or layers (a common process of deep learning-based multi-context approaches), resulting in a method with fewer parameters and easier to train. Moreover, during the training stage, the network gives a score (based on accuracy or loss) for each patch size. Then, in the prediction phase, the process selects the patch size with the highest score to perform the segmentation. Therefore, differently from empirically selecting the best patch size which requires a new network trained for each evaluated patch (increasing the computational complexity and training time), the proposed technique evaluates several patches during the training stage and selects the best one for the inference phase doing only a unique training procedure. Aside from the aforementioned advantages, the proposed networks can be fine-tuned for any semantic segmentation application, since they do not depend on the patch size to process the data. This allows other applications to benefit from the patterns extracted by our models, a very relevant feature specially when working with small amounts of labeled data [Nogueira et al., 2017c].

## 6.1 Pixel Classification Approach

In the next sections, we describe the basic ideas of the proposed approach to perform pixel classification in *RSIs*. The core (dynamic multi-context) algorithm is presented in Section 6.1.1. Then, the dilated *ConvNet* architectures proposed to perform pixel classification are presented in Section 6.1.2.

### 6.1.1 Dynamic Multi-Context Algorithm

We propose a novel method to perform semantic segmentation of *RSIs* that: (i) exploits the multi-context paradigm without increasing the number of trainable parameters of the network, and (ii) defines, in training time, the best patch size that should be exploited by the network in the **test phase**.

As presented in Algorithm 1, the **training process** receives as input: (i) the data  $\mathcal{D}$ , where the images and their reference labels come from, (ii) a patch size distribution  $\mathcal{P}$ , that represents the probability function (kept the same during all the training procedure) from which the patch sizes will come from, (iii) the patch scores  $\mathcal{S}$  (initialized with zeros), which will be used during the training procedure to accumulate the score of the patch sizes produced by the network, (iv) the network  $\mathcal{N}(\cdot)$ , which can be seen as a function that processes the input batch  $(\mathcal{X}_{\lambda \times \lambda}, \mathcal{Y}_{\lambda \times \lambda}) \in \mathcal{D}$  (a tuple of patches and reference semantic labels with the same resolution  $\lambda \times \lambda$ ) with respect to the current weights  $\mathcal{W}$ , updating them, and outputting a score for the batch  $v$ , that can be seen, somehow, as a quality assessment of the patch size relative to the current network, (v) the number of iterations or epochs  $n$ .

The first step (line 2) of each iteration (comprised in the loop from line 1 to 6) of training procedure is to randomly select a patch size  $\lambda_k$  from the distribution  $\mathcal{P}$ , which may be any valid distribution, such as uniform or multinomial. Then, in the line 3 of the algorithm, this patch size  $\lambda_k$  is used to create a new batch  $(\mathcal{X}_{\lambda_k \times \lambda_k}, \mathcal{Y}_{\lambda_k \times \lambda_k}) \in \mathcal{D}$ . Observe that, at each iteration of the algorithm, a new patch size is selected and a new random batch (using different sites) is sampled based on this size. This batch is then employed to train the network  $\mathcal{N}$ , i.e., to update its weights  $\mathcal{W}$  (line 4). It is important to emphasize that this training process (performed by the sampled batch) represents only a single step (iteration) of the mini-batch optimization strategy [Goodfellow et al., 2016] (and not the full train) that processes one whole batch to then update the network weights  $\mathcal{W}$ . As aforementioned, for each step of the mini-batch training algorithm, the network  $\mathcal{N}$  outputs a score for the current batch  $v$ , which can be any metric (such as a loss or accuracy) that estimates the performance of the network based on the

current batch. This generated score  $v$  is used to update the patch scores  $\mathcal{S}$  (line 5 of the algorithm), which accumulate, throughout the training procedure, the scores of the patch sizes and are employed in the selection of the best patch size during the inference stage. Note the difference between the patch distribution  $\mathcal{P}$  and the scores  $\mathcal{S}$ , i.e., while the former is a distribution employed during the whole training procedure the latter accumulates the scores of the patch sizes given by the network to be employed in the prediction phase. Hence, there is no connection between patch distribution  $\mathcal{P}$  and the scores  $\mathcal{S}$ , and an update in  $\mathcal{S}$  has no impact on  $\mathcal{P}$ , which is kept fixed throughout the training process.

All the aforementioned steps are repeated during the training process until the number of iterations  $n$  is reached. As it can be noticed, the multi-context information is aggregated to the model by allowing the network to be trained using batches composed of patches of multiple sizes. This process allows the network to capture and extract features by considering distinct context regions, a very important process as presented and explained in (caption of) Figure 6.2.

When the training phase is finalized, the algorithm outputs the updated network  $\mathcal{N}$  (i.e., its weights  $\mathcal{W}$ ) and the updated patch scores  $\mathcal{S}$ . The second benefit of the proposed method is almost a direct application of the patch scores  $\mathcal{S}$  created during the training phase. Precisely, in the **prediction phase**, scores  $\mathcal{S}$  over the patch sizes are averaged and analyzed. The **best patch** size  $\lambda^*$  (which corresponds to the highest or lowest score, for accuracy and loss, respectively) is then selected and used to create patches. The network processes these patches (of  $\lambda^* \times \lambda^*$  pixels) outputting the prediction maps, but no updates in the patch scores  $\mathcal{S}$  are performed. It is important to highlight that the proposed technique can only choose the best patch size within all possible sizes determined by the patch distribution  $\mathcal{P}$ , since only the patches within  $\mathcal{P}$  are evaluated by the algorithm.

#### ALGORITHM 1

Process of dynamic training a Convolutional Networks.

**Require:** data  $\mathcal{D}$ , network  $\mathcal{N}$  with its weights  $\mathcal{W}$ , number of iterations  $n$ , patch distribution  $\mathcal{P}$ , and patch scores  $\mathcal{S}$  (initialized with zeros).

**Ensure:** updated of the network weights  $\mathcal{W}$ , and patch scores  $\mathcal{S}$ .

- 1: **para**  $t=1$  to  $n$  **faça**
- 2:    $\lambda_k = \mathcal{P}(k)$  {Randomly select current patch size}
- 3:    $(\mathcal{X}_{\lambda_k \times \lambda_k}, \mathcal{Y}_{\lambda_k \times \lambda_k}) \in \mathcal{D}$  {Create new batch}
- 4:    $v_{\lambda_k} = \mathcal{N}(\mathcal{X}_{\lambda_k \times \lambda_k}, \mathcal{Y}_{\lambda_k \times \lambda_k}; \mathcal{W})$  {Continue training}
- 5:    $\mathcal{S}_{\lambda_k} = \mathcal{S}_{\lambda_k} + v_{\lambda_k}$  {Update scores}
- 6: **fim para**

### 6.1.2 Architectures

As presented in Section 3.1.2.2, the properties of the dilated convolutions [Yu and Koltun, 2015] make them fit perfectly into the proposed multi-context methodology, given that a network composed of such layers is capable of processing an input of any size without downsampling it. This creates the possibility of processing patches of any size without constraints. Although these layers have the advantage of computing feature responses at the original image resolution, a network composed uniquely of dilated convolutions would be costly to train especially when processing entire (large) scenes. However, as previously mentioned, processing an entire RSI is not possible (because of its huge size) and, therefore splitting the image into small patches is already necessary, which naturally alleviates the training process.

Though, in this work, we explore networks composed of dilated convolutions, other types of ConvNets could be used, such as fully convolutions [Long et al., 2015] and deconvolutions [Noh et al., 2015; Badrinarayanan et al., 2017]. These networks can also process patches of varying size, but they have restrictions related to a high variation of the patch size. Specifically, these networks need to receive a patch larger enough to allow the generation of a coarse map, that is upsampled to the original size. If the input patch is too small, the network could reach a situation where it is not possible to create the coarse map and, consequently, the final upsampled map. Such problem is overcome by dilated convolutions [Yu and Koltun, 2015], which are allowed to process patches of any size, without distinction, always outputting results with the same resolution of the input data (given proper configurations, such as stride and padding). Such concept is essential to allow the variance of patch sizes, from very small values (such as  $7 \times 7$ ) to larger ones (for instance,  $256 \times 256$ ).

Considering this, a full set of experiments (guided by [Bengio, 2012]) was performed in order to define the best architectures. After the experiments, four networks, illustrated in Figure 6.3, have been selected (based on the average accuracy) and extensively evaluated in this work. The first network, presented in Figure 6.3a, is composed of seven layers: six dilated convolutions (that are responsible to capture the patterns of the input images) and a final  $1 \times 1$  convolution layer, which is responsible to generate the dense predictions. There is no pooling or normalization in this network, and all layers have stride 1. Specifically, the first two convolutions have  $5 \times 5$  filters with dilation rate  $r$  1 and 2, respectively. The following two convolutions have  $4 \times 4$  filters but rate 3 and 4 while the last two convolutions have smaller filters ( $3 \times 3$ ) but 5 and 6 as dilation rate. Because this network has 6 layers responsible for the feature extraction, it will be referenced as **Dilated6**. The second network (Figure 6.3b) is based on

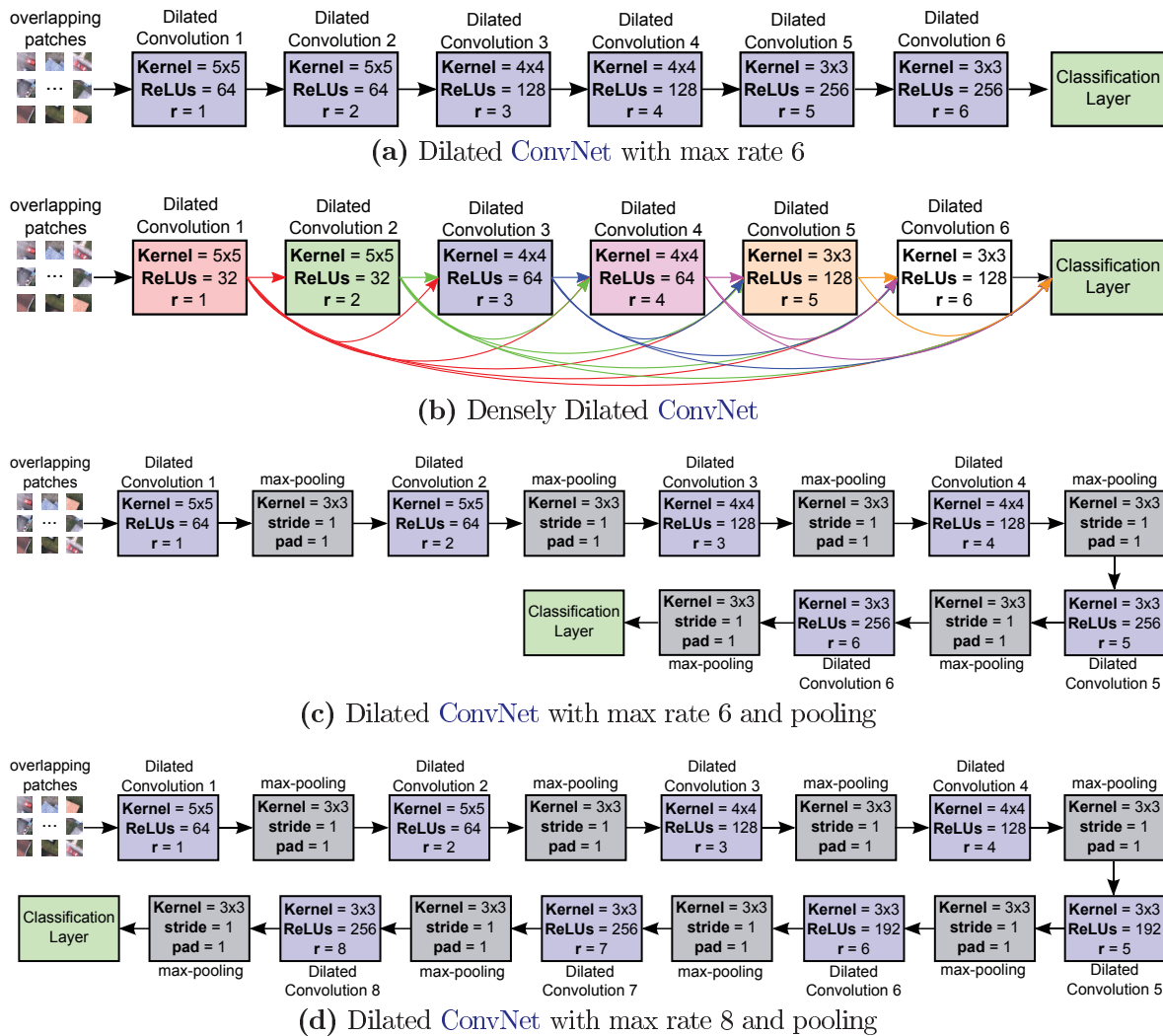


Figure 6.3: Dilated Convolutional Network architectures.

densely connected networks [Huang et al., 2017], which recently achieved outstanding results on the image classification task. This network is very similar to the first one having the same number of layers and configuration. The main difference between these networks is that a layer receives as input feature maps of all preceding layers. Hence, the last layer has access to all feature maps generated by all other layers of the network. This process allows the network to combine different feature maps with distinct level of abstraction, supporting the capture and learning of a wide range of feature combination. Because this network has 6 layers responsible for the feature extraction and is densely connected, it will be referenced in this work as **DenseDilated6**. The third network, presented in Figure 6.3c, has the same configuration of the Dilated6, but with pooling layers between each convolutional one. Given a specific combination of stride and padding, no downsampling is performed over the inputs in these pooling



layers. Because of the number of layers and the pooling layers, this network will be referenced hereafter as **Dilated6Pooling**. The last network (Figure 6.3d) is an extension of the previous one, having 8 dilated convolutions instead of only 6. The last two convolutional layers have smaller filters ( $3 \times 3$ ) but 7 and 8 as dilation rate. There are pooling layers between all convolutional ones. Given that this network has 8 dilated convolutional and pooling layers, it will be referenced hereafter as **Dilated8Pooling**. Although only this network with 8 layers is explored in this work, other variant networks (such as Dilated8 and DenseDilated8) were initially considered but not retained for further experiments due to the similar initial performance and longer training time when compared to the Dilated6 variant networks.

## 6.2 Specific Experimental Setup

In this section, we present the details about the experiments conducted to evaluate the proposed method. Section 6.2.1 describes the baselines while Section 6.2.2 presents details about the used experimental protocol, such as the framework.

### 6.2.1 Baselines

For the Coffee Dataset, we employed the Cascaded Convolutional Neural Network (CCNN) [Nogueira et al., 2015b] as baseline. This method, a contribution of this thesis, employs a multi-context strategy by aggregating several **ConvNets** in order to perform the classification of fixed size tiles towards the final segmentation of the image. For the GRSS Data Fusion Dataset, we employed, as baseline, the method proposed by [Santana et al., 2017]. Their algorithm extracts features with many levels of context by exploiting different layers of a pre-trained convolutional network, which are then combined in order to aggregate multi-context information.

Aside from this, for both aforementioned datasets, we also considered as baseline a precursor method [Nogueira et al., 2016a] proposed during the development of this thesis. This basic approach, referenced hereafter as **pixelwise**, is essentially based on context windows and, therefore, classifies each pixel independently. Also, for these two datasets, we considered as baselines: (i) Fully Convolutional Networks (FCNs) [Long et al., 2015]. In this case, the precursor pixelwise architectures were converted into a fully convolutional network and exploited as a baseline. (ii) Deconvolutional networks [Badrinarayanan et al., 2017; Noh et al., 2015]. Again, the precursor pixel architectures were converted into a deconvolutional network (based on the well-known SegNet [Badrinarayanan et al., 2017] architecture) and exploited as a baseline

in this work. (iii) dilated network [Yu and Koltun, 2015], which is, in this case, the Dilated6Pooling (Figure 6.3c). All these networks were trained traditionally using patches of constant size defined according to a previous set of experiments. Precisely, patches of  $7 \times 7$  and  $25 \times 25$  were used for the Coffee and GRSS Data Fusion datasets, respectively.

For the remaining datasets (Vaihingen and Potsdam), we refer to the official results published on the challenge website<sup>1</sup> as baselines for the proposed work.

## 6.2.2 Protocol

For the Coffee Dataset, we conducted a 5-fold cross-validation to assess the performance of the proposed algorithm. In this case, the reported results are the average metric of the five runs followed by its corresponding standard deviation. For the remaining datasets, we employed the training/test protocol, as presented in Section 4.2. In these experiments, the final results are reported based on the performance on the test set. As introduced, the reported results are always some combination of the metrics described in Section 4.2, in order to provide enough information about the effectiveness of the proposed method.

The proposed method and network<sup>2</sup> were implemented using TensorFlow [Abadi et al., 2015], a framework conceived to allow efficient exploitation of deep learning with Graphics Processing Units (GPUs). All experiments were performed on a 64 bits Intel i7 4960X machine with 3.6GHz of clock and 64GB of RAM memory. Four GeForce GTX Titan X with 12GB of memory, under an 8.0 CUDA version, were employed in this work. Note, however, that each GPU was used independently and that all networks proposed here can be trained using only one GPU. Ubuntu version 16.04.3 LTS was used as operating system.

As previously stated, a set of experiments (guided by [Bengio, 2012]) was executed to define the hyperparameters. After all the setup experiments, the best values for hyperparameters, presented in Table 5.2, were determined for each dataset. The number of iterations increases with the complexity of the dataset in order to ensure convergence. In the proposed models, the learning rate, responsible to determine how much an updating step influences the current value of the network weights, starts with a

---

<sup>1</sup><http://www2.isprs.org/commissions/comm2/wg4/vaihingen-2d-semantic-labeling-contest.html> and <http://www2.isprs.org/commissions/comm2/wg4/potsdam-2d-semantic-labeling.html>.

<sup>2</sup>The code has been made publicly available at <https://github.com/keillernogueira/dynamic-rs-segmentation>

**Table 6.1:** Hyperparameters employed in each dataset.

Datasets	Learning Rate	Weight Decay	Iterations	Exponential Decay (decay/steps)
Coffee Dataset	0.01	0.001	150,000	0.5/50,000
GRSS Data Fusion Dataset	0.01	0.005	200,000	0.5/50,000
Vaihingen Dataset	0.01	0.01	500,000	0.5/50,000
Potsdam Dataset	0.01	0.01	500,000	0.5/50,000

high value and is reduced during the training phase using the exponential decay [Abadi et al., 2015] with parameters defined according to the last column of Table 6.1.

## 6.3 Results and Discussion

In this section, we present and discuss the obtained results. Specifically, we first analyze the parameters of the proposed technique: Section 6.3.1 presents the results achieved using different patch distributions, Section 6.3.2 analyzes distinct functions to update the patch size score, and Section 6.3.3 evaluates different ranges for the patch size. Then, a comparison between the dilated and standard convolution is presented in Section 6.3.4. A convergence analysis of the proposed technique is performed in Section 6.3.5 while a comparison between networks trained with the proposed and standard training techniques is presented in Section 6.3.6. Finally, a comparison with the state-of-the-art is reported in Section 6.3.7.

### 6.3.1 Patch Distribution Analysis

As explained at the beginning of Section 6.1.1, the algorithm receives as input a list of possible patch sizes and a correspondent distribution. In fact, any distribution could be used, including uniform or multinomial. Given the influence of this distribution over the proposed algorithm, experiments have been conducted to determine the most appropriate distribution. Towards this, we selected and compared three distinct distributions. First is the **uniform** distribution over a range of values, i.e., given two extreme points, all intermediate values (extremes included) inside this range should have the same probability of being selected. Second is the uniform distribution but over selected values (and not a range). In this case, referenced as **uniform fixed**, the probability distribution is equally divided into the given values (the remaining intermediate points have no probability of being selected). The last distribution evaluated is the **multinomial**. In this case, ordinary values inside a range have the same proba-

**Table 6.2:** Results over different distributions.

	Overall Accuracy	Kappa	Average Accuracy	F1 Score
<b>Uniform</b>	86.13±2.39	69.39±3.48	84.81±1.65	84.58±1.90
<b>Uniform Fixed</b>	86.27±1.44	69.41±2.01	84.85±1.66	84.62±1.06
<b>Multinomial</b>	86.06±1.68	68.94±2.94	84.56±2.00	84.39±1.51

bility but several given points have twice the chance of being selected. In those last two distribution, the relevant points are chosen based on previous works of the literature.

The main difference between the evaluated distributions is related to the prior knowledge of the application. In the uniform distribution, no prior knowledge is assumed, and all patch sizes from the input range have the same probability, taking more time to converge the model. The uniform fixed distribution assumes a good knowledge of the application and only pre-defined patch sizes can be (equally) selected and evaluated, taking less time to converge the model. The multinomial distribution tries to blend previous ideas. Assuming a certain prior knowledge of the application, the multinomial distribution weighs the probabilities allowing the network to give more attention to specific pre-defined patch sizes but without discarding the others. If prior intuition is confirmed, these pre-defined patch sizes are randomly selected more often and the network should converge faster. Otherwise, the proposed process is still able to use other (non-pre-defined) patch sizes and converge the network anyway.

Results of this analysis can be seen in Table 6.2. Note that all experiments were performed using the Coffee Dataset [Nogueira et al., 2016a], Dilated6 network (Figure 6.3a), accuracy as score function, and hyperparameters presented in Table 6.1. In these experiments, patches size varied from  $25 \times 25$  to  $50 \times 50$ . Specifically, for the uniform distribution, any value between 25 and 50 has the same probability of being selected, while for the multinomial distribution, all values have some chance to be selected, but these two points have twice the probability. For the uniform fixed, these two patch sizes split the total probability and each one has 50% of being selected. Overall, the variation of the distribution has no serious impact on the final outcome, since results are all very similar. However, given its simplicity and faster convergence, for the remaining of this work, results will be reported using the **uniform fixed** distribution.

### 6.3.2 Score Function Analysis

As introduced in Section 6.1, at each training iteration an update is performed in the score of patch sizes, which are used in the selection of the best patch size during the testing stage. In this work, we evaluated two possible score functions that could be

**Table 6.3:** Results over different score functions.

	<b>Overall Accuracy</b>	<b>Kappa</b>	<b>Average Accuracy</b>	<b>F1 Score</b>
<b>Accuracy</b>	86.27±1.44	69.41±2.01	84.85±1.66	84.62±1.06
<b>Loss</b>	86.15±1.96	69.16±3.41	84.68±2.02	84.49±1.76

employed in this step: the loss and the accuracy. In the first case, the loss is a measure (obtained using cross entropy [Goodfellow et al., 2016], in this case) that represents the error generated in terms of the ground-truths and the network predictions. In the second case, the score is represented by the pixel classification accuracy [Congalton and Green, 2008] of the images.

To analyze the most appropriate score function, experiments were performed varying only this particular parameter and maintaining the remaining ones. Specifically, these experiments were conducted using: the Coffee Dataset [Nogueira et al., 2016a], Dilated6 network (Figure 6.3a), uniform fixed distribution (over  $25 \times 25$  and  $50 \times 50$ ), and same hyperparameters presented in Table 6.1. Results can be seen in Table 6.3. Through the table, it is possible to see that both score functions achieved similar results. However, since **accuracy score** is more intuitive, for the remaining of this work, results will be reported using this function.

### 6.3.3 Range Analysis

Although the presented approach is proposed to select automatically the best patch size, in training time, avoiding lots of experiments to adjust such size (as done in several works [Nogueira et al., 2016a; Paisitkriangkrai et al., 2016; Volpi and Tuia, 2017]), in this section, the patch size range is analyzed in order to examine the robustness of the method.

This range is evaluated on all datasets, except Potsdam. Such dataset is very similar to Vaihingen one and, therefore, analysis and decisions made over the latter dataset are also applicable to the Potsdam one. Furthermore, in order to evaluate such dataset, a validation set, created according to [Volpi and Tuia, 2017], was employed. Experiments were conducted varying only the patch size range but maintaining the remaining configurations. Particularly, the experiments employed the same hyperparameters (presented in Table 6.1), Dilated6 network (Figure 6.3a), and uniform fixed distribution.

Table 6.4 presents the obtained results. Each dataset was evaluated over several ranges, selected based on previous works [Nogueira et al., 2016a; Volpi and Tuia, 2017]. Specifically, each dataset was evaluated in a large range (comprising from small to large

sizes) and subsets of such range. Table 6.4 also presents the most selected patch size (for the testing phase) for each experiment, giving some insights about how the proposed method behaves during such step.

For the Coffee Dataset [Nogueira et al., 2016a], obtained results are all very similar making it difficult to define a better or worse range. Hence, any patch size range could be selected for further experiments, showing the robustness of the proposed algorithm which yielded similar results independently of the patch size range. Because of processing time (smaller patches are processed faster), in this case, patch size range 25, 50 was selected and used in all further experiments.

For remaining datasets, a specific range achieved the best result. For the GRSS Data Fusion Dataset [Liao et al., 2015], the best result was obtained when considering the largest range (7, 14, 21, 28, 35, 42, 49, 56, 63, 70), i.e., the range varying from small to large patch sizes. For Vaihingen [ISPRS, 2018a], the intermediate range (45, 55, 65, 75, 85) achieved the best result. Therefore, in these cases, such ranges were selected and used in the remaining experiments of this work. However, as can be seen through Table 6.4, other ranges also produce competitive results and could be selected and used without significant loss of performance, which confirms the robustness of the proposed method in relation to the patch size range, allowing it to process images without the need of experimentally searching for the best patch size configuration.

In terms of patch size selection (during the inference phase), the algorithm really varies depending on the experiment. For the Coffee Dataset, the most selected patch sizes were 50 and 75, showing a trend towards such interval. For the remaining datasets, larger patches were favored in our experiments. This may be justified by the fact that urban areas have complex interactions and larger patches allow the network to capture more information about the context. Though the best patch size is really dependent on the experiment, current results showed that the proposed approach is able to learn and select the best patch size in processing time producing interesting outcomes when compared to state-of-the-art works, a fact reconfirmed in Section 6.3.7.

### 6.3.4 Convolution Operation Analysis

Although the proposed networks use dilated convolutions, it is possible to recreate such architectures using standard convolution operations. As introduced in Section 3.1.2.2, the only difference between these convolution operations is the possibility to have gaps in the filter weights, a special characteristic of the dilated convolutions [Yu and Koltun, 2015]. Such aspect makes all the difference since dilated convolution can expand the exploited context (by enlarging the filter weights) without increasing the number of

**Table 6.4:** Results of the proposed approach when varying the input range of patch sizes. For Vaihingen, a validation set (created according [Volpi and Tuia, 2017]) is employed. Bold patch size ranges were selected for all further experiments.

Datasets	Patch Size Range	Most Selected Size	Overall Accuracy	Average Accuracy
Coffee	<b>25,50</b>	50	86.27±1.44	84.85±1.66
	50,75	50	87.32±1.82	85.59±1.59
	75,100	75	86.07±1.95	85.91±1.68
	25,50,75,100,125	75	87.11±1.74	85.17±1.52
GRSS	7,14,21,28,35	35	87.93	85.87
	28,35,42,49,56	49	87.71	85.26
	42,49,56,63,70	70	88.33	88.04
	<b>7,14,21,28,35,42,49,56,63,70</b>	70	90.10	90.13
Vaihingen	25,45,55,65	65	86.60	71.03
	<b>45,55,65,75,85</b>	85	88.66	71.96
	25,45,55,65,85,95,100	95	87.44	71.30

parameters, while standard convolutions are not able to do this since the filters are always grouped (without gaps). This is a great advantage since a deeper network composed of standard convolution operations (without any downsample or upsample operation) would require more layers in order to aggregate a large context, while a network composed of dilated convolutions can expand the context without increasing the number of parameters, requiring fewer layers.

In order to demonstrate this advantage of dilated convolutions over standard ones, we performed experiments comparing two networks that have exactly the same architecture (Dilated6 – Figure 6.3a) but differ in the convolution operation: while one network uses dilated convolutions, the second architecture employs the standard operation. Since the Dilated6 network does not have pooling layers, the comparison between these networks is totally fair, given that the only difference is the convolution operation type. All datasets were used in this experiment, except Potsdam. This is because the Vaihingen and Potsdam Datasets are very similar and analysis performed over one can also be extended to the other. A validation set, created according to [Volpi and Tuia, 2017], was used to evaluate the Vaihingen Dataset. Experiments were executed preserving all configurations and varying only the convolution type. Particularly, the configuration was defined taken into account previous experiments, i.e., it uses uniform fixed distribution, patch ranging according to Section 6.3.3, accuracy as score function, and hyperparameters presented in Table 6.1.

Results can be seen in Table 6.5. Overall, architectures based on dilated convolution outperformed the networks that employ the standard operation. Since the only

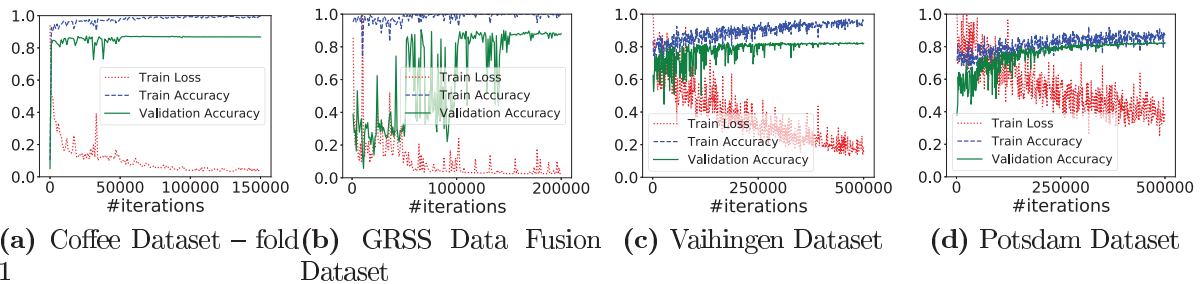
**Table 6.5:** Results of the Dilated6 network trained using distinct convolution types.

Datasets	Convolution Type	Overall Accuracy	Average Accuracy
Coffee	Standard	84.13±1.28	82.97±0.48
	Dilated	86.27±1.44	84.85±1.66
GRSS	Standard	85.70	85.31
	Dilated	90.10	90.13
Vaihingen	Standard	86.13	69.65
	Dilated	88.66	71.96

difference between the networks is the convolution (and, consequently, the exploited context), these results show the advantage of the dilated operations over the standard one.

### 6.3.5 Convergence Analysis

In this section, we analyze the convergence of the proposed technique. Figure 6.4 presents the convergence of the datasets using the Dilated6 network, accuracy as score function, uniform fixed distribution, and hyperparameters presented in Table 6.1. According to the figure, the loss and accuracy vary significantly at the beginning of the process but, with the reduction of the learning rate, the networks converge independently of the use of distinct patch sizes during the training. Moreover, the test/validation accuracy (green line) converges and stabilizes showing that the networks can learn to extract features from patches of multiple sizes.



**Figure 6.4:** Convergence of Dilated6 network for all datasets. For the Coffee Dataset, only the fold 1 is reported. For Vaihingen and Potsdam Datasets, the validation set (created according [Volpi and Tuia, 2017]) is reported.



### 6.3.6 Performance Analysis

To analyze the efficiency, in terms of performance and processing time, of the proposed algorithm, several experiments were conducted comparing the same network trained using two distinct methods: (i) the **traditional** training process [Goodfellow et al., 2016], in which the network is trained using patches of constant size, without any variation. This method is the standard one when it comes to neural networks and is the most exploited in the literature for training deep learning-based techniques. Also, this is the approach that is used to empirically select the best patch size, which is traditionally done by training several networks, one for each considered patch. (ii) the proposed **dynamic** training process, in which the network is trained with patches of varying size.

Two datasets were selected to be evaluated using these training strategies: (i) the GRSS Data Fusion Dataset, which has the largest patch size range (according to Section 6.3.3) allowing a better comparison between the training strategies, and (ii) Vaihingen Dataset, which is very similar to Potsdam one and, therefore, allows the conclusions to be applied to this one. To evaluate this dataset, a validation set, created according [Volpi and Tuia, 2017], was employed.

Specifically, in these experiments, Dilated6 network (Figure 6.3a) is trained using both strategies. For the proposed dynamic training process, previous experiments were taken into account, i.e., it uses uniform fixed distribution, patch ranging according to Section 6.3.3, accuracy as score function, and hyperparameters presented in Table 6.1. Concerning the traditional training process, several networks (with same architecture) were trained using each of the possible patch sizes.

Results of these experiments are presented in Table 6.6. For both datasets, networks trained with the proposed approach outperform the models trained with the traditional training process (independently of the patch size), showing the ability of the proposed method to capture multi-context information from patches of distinct size which improve the performance of the final model. Also, on average, the processing time of the proposed method is lower than the traditional training process, in which the computational time increases with the increase of the patch size, an expected behavior given that the convolution process using large inputs takes more time than using smaller ones.

Specifically, for the GRSS Data Fusion Dataset, considering only models trained with the traditional method, the best result is achieved by the network using patches of  $70 \times 70$  pixels. This **ConvNet** took around 160 hours to train using 200,000 iterations and achieved 86.93% of average accuracy. However, the model trained with

**Table 6.6:** Comparison between the dilated network trained using the proposed and the traditional method.

Dataset	Training Process	Patch Size	Training Time (hours)	Average Accuracy
GRSS Data Fusion	Traditional	7	35	83.33
		28	59	85.48
		49	86	85.94
		70	160	86.93
	Dynamic	7,14,21, 28,35,42,49, 56,63,70	81	90.13
Vaihingen	Traditional	45	125	66.29
		55	160	66.77
		65	200	66.84
		75	260	66.65
		85	325	66.96
	Dynamic	45,55,65,75,85	220	71.96

the proposed dynamic process outperforms this result while taking less time to train. Particularly, Dilated6 network trained using the dynamic process produced 90.13% of average accuracy while taking around 81 hours to train. This improvement in the performance is due to the exploitation of distinct contexts provided by different patch sizes during the training procedure. This process of using distinct patch sizes also speeds up the training, given that small patches (which are processed faster) are also used together with large ones.

The same conclusions hold for the Vaihingen Dataset. Precisely, the best result using the traditional method is achieved by the network trained with patches of  $85 \times 85$  pixels. This `ConvNet` took around 325 hours to train using 500,000 iterations and achieved 66.96% of average accuracy. However, this result was outperformed by the network trained using the proposed dynamic strategy, while taking less training time. Such model produced 71.96% of average accuracy while taking around 220 hours to train.

Moreover, the proposed dynamic strategy has another advantage: while the empirical method would require training several networks in order to select the best patch size, resulting in a greater computational time, the proposed strategy combines all patch sizes during the training stage while selecting the best size for the inference phase, requiring only one full procedure to achieve its final result. Hence, overall, the proposed method requires less training time than the empirical approach, while achieving better results.

## 6.3.7 State-of-the-art Comparison

### 6.3.7.1 Coffee Dataset

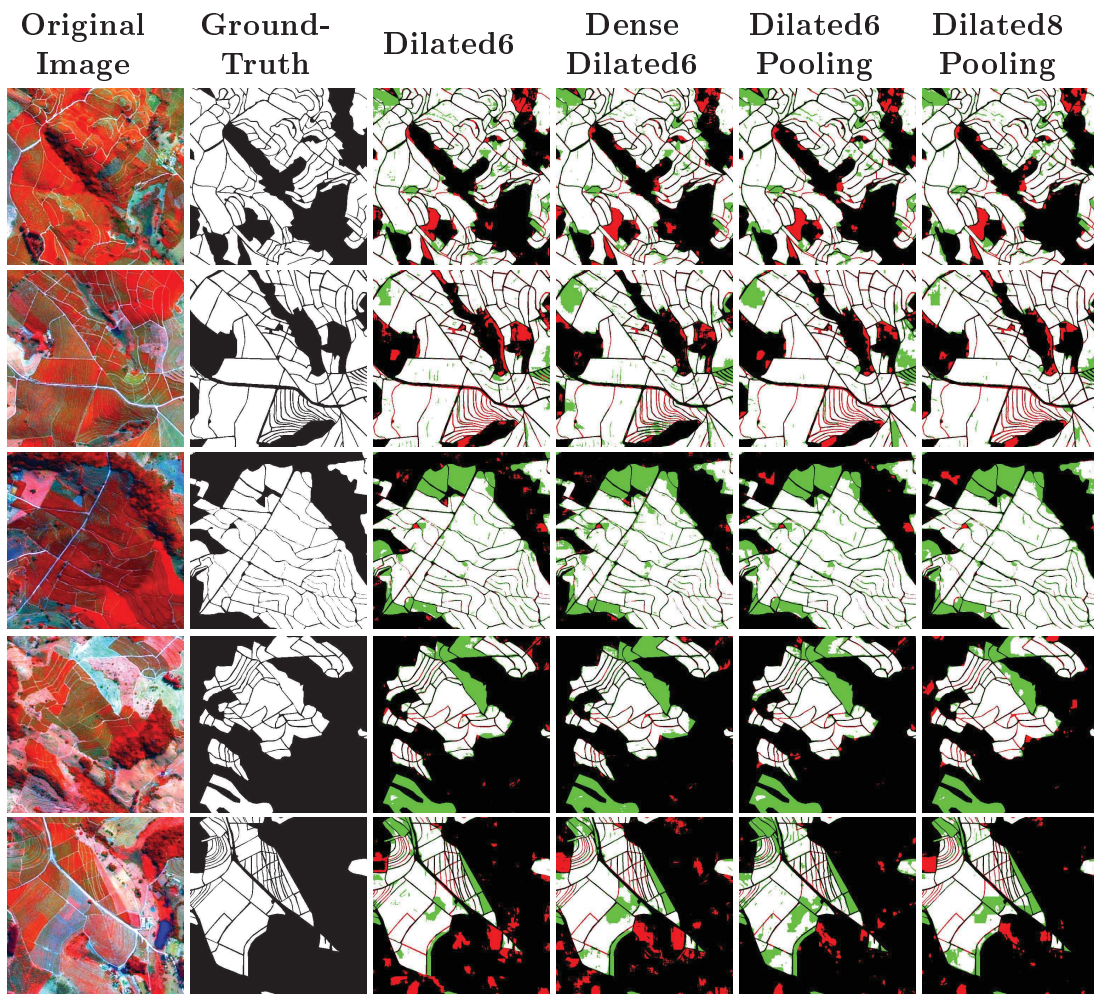
Using analysis performed on previous sections, we have conducted several experiments over the Coffee Dataset. Results for the proposed method, as well as, for the state-of-the-art baselines are presented in Table 6.7. In order to allow a visual comparison, relevance maps (an image that facilitates the observation of the right and wrong predictions) for the Coffee Dataset using different networks trained with the proposed method are presented in Figure 6.5.

Overall, all baselines produced similar results. While the pixelwise network yielded a slightly worse result with a higher standard deviation, all other baselines reached basically the same level of performance, with a smaller standard deviation. This may be justified by the fact that the pixelwise network does not learn much information about the pixel interaction (since each pixel is processed independently), while the other methods process and classify a set of pixels simultaneously. Because of the similar results, all baselines are comparable.

This same behavior may be seen among the networks trained with the proposed methodology. Although these networks achieved comparable results, such models outperformed the baselines. Furthermore, the Dilated6Pooling trained with the proposed dynamic method produced better results than the same network trained with traditional training process (mainly in the Kappa Index). These results show the effectiveness of the proposed technique that produces state-of-the-art outcomes by capturing multi-context information while selecting the best patch size, two great advantages when compared to the traditional training process.

**Table 6.7:** Results for the Coffee dataset.

Training Process	Network	Average Accuracy	Kappa
<b>Traditional</b>	Pixelwise [Nogueira et al., 2016a]	81.72±2.38	62.75±7.42
	CCNN [Nogueira et al., 2015b]	82.80±2.30	64.60±4.34
	FCN	83.25±2.47	66.00±3.55
	Deconvolution Network	82.61±2.05	65.56±3.47
	Dilated network (Dilated6Pooling)	82.52±1.14	66.14±2.27
<b>Dynamic</b>	Dilated6	84.85±1.66	69.41±2.01
	DenseDilated6	85.88±2.34	71.51±2.74
	Dilated6Pooling	85.77±1.74	72.27±1.38
	Dilated8Pooling	86.67±1.39	73.78±1.87



**Figure 6.5:** Images of the Coffee Dataset, their respective ground-truths, and the relevance maps generated by the proposed algorithm. Legend – White: True Positive. Black: True Negative. Red: False Positive. Green: False Negative.

### 6.3.7.2 GRSS Data Fusion Dataset

We also performed several experiments on the GRSS Data Fusion Contest Dataset [Liao et al., 2015] considering all analysis carried out in previous sections. Experimental results, as well as baselines, are presented in Table 6.8. The prediction maps obtained for the test set are presented in Figure 6.6.

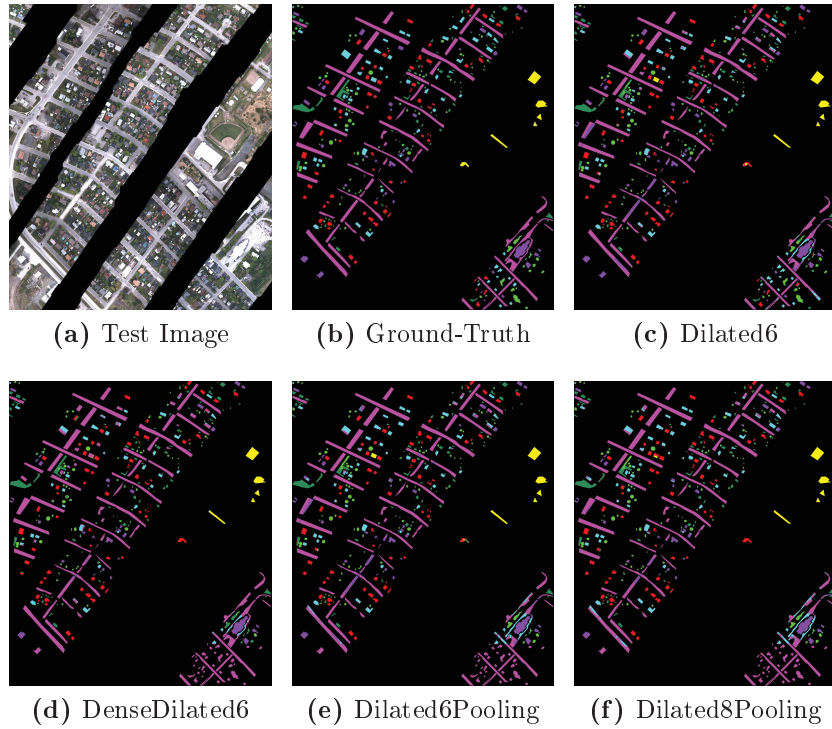
Overall, Dilated6 produced the best result among all approaches. In general, networks trained with the proposed method outperformed the baselines. Moreover, the Dilated6Pooling trained with the proposed dynamic technique outperformed the baseline composed of the same network trained using traditional training process, corroborating with previous conclusions.

Among the baseline methods, although all of them achieved comparable results,

the best outcome was yielded by the Deep Contextual [Santana et al., 2017]. This method also leverages from multi-context information, since it combines features extracted from distinct layers of pre-trained ConvNets. When comparing this method with the best result of the proposed technique (Dilated6), one can clearly observe the advantage of the proposed approach, which improves the results for all metrics when compared to the Deep Contextual [Santana et al., 2017] approach. This reaffirms the effectiveness of the proposed dynamic method, corroborating with previous conclusions.

**Table 6.8:** Results for the GRSS Data Fusion Dataset.

Training Process	Network	Overall Accuracy	Average Accuracy	Kappa Index
Traditional	Pixelwise [Nogueira et al., 2016a]	85.04	86.52	78.18
	FCN	83.27	87.45	76.10
	Deconvolution Network	82.15	86.24	75.04
	Dilated network (Dilated6Pooling)	83.96	83.83	76.12
	Deep Contextual [Santana et al., 2017]	85.45	88.33	79.01
Dynamic	Dilated6	90.10	90.13	85.22
	DenseDilated6	88.66	80.62	81.80
	Dilated6Pooling	88.05	86.12	81.81
	Dilated8Pooling	89.03	85.31	83.08



**Figure 6.6:** The GRSS Data Fusion test image, the respective ground-truth, and the prediction maps generated by the proposed algorithm. Legend – Black: unclassified. Light purple: road. Light green: trees. Red: red roof. Cyan: gray roof. Dark purple: concrete roof. Dark green: vegetation. Yellow: bare soil.

### 6.3.7.3 Vaihingen Dataset

As introduced in Section 6.2.1, official results for the Vaihingen Dataset are reported only by the challenge organization that held some images that are used for testing the submitted algorithms. Therefore, one must submit the outcomes of the proposed algorithm to have them evaluated. In our case, following previous analysis, we submitted five approaches: the first four are related to each network presented in Section 6.1 trained with the 6 classes (which are represented in the official results as UFMG\_1 to 4), and the fifth one, represented in the official results as UFMG\_5, is the Dilated8 network (Figure 6.3d) trained with only 5 classes, i.e., all labels except the clutter/background one. This last submission is due to the lack of training data for that class which corresponds to only 0.67% of the dataset (as stated in Table 4.4). It is important to note that all submissions related to the proposed work do not use any post-processing, such as Conditional Random Fields (CRF) [Lafferty et al., 2001].

Some official results reported by the organization are summarized in Table 6.9. In addition to our results, this table also compiles the **best results of each work** with enough information to make a fair comparison, i.e., in which the proposed approach is minimally explained. In order to allow a visual comparison, examples of the proposed method, for the validation and test sets, are presented in Figures 6.7 and 6.8, respectively.

It is possible to notice that the proposed work yielded competitive results. The best result, in terms of overall accuracy, was 90.3% achieved by DLR\_9 [Marmanis et al., 2018] and GSN3 [Wang et al., 2017]. Our best result (UFMG\_4) appears in fifth place by yielding 89.4% of overall accuracy, outperforming several methods, such as ADL\_3 [Paisitkriangkrai et al., 2015] and RIT\_L8 [Liu et al., 2017], that also tried to aggregate multi-context information. However, as can be seen in Table 6.9 and Figure 6.9a, while the other approaches have a larger number of trainable parameters, our network has only 2 millions, which makes it less prone to overfitting and, consequently, easier to train, showing that the proposed method really helps in extracting all feasible information of the data even if using limited architectures (in terms of parameters). In fact, the number of parameters of the network is so relevant that authors of DLR\_9 submission [Marmanis et al., 2018], one of the best results but with a higher number of parameters, do not recommend their proposed method for practical use because of the memory consumption and expensive training phase. Furthermore, the obtained results, that do not have any post-processing, are better than others, such as DST\_2 [Sherrah, 2016], that employ CRF as post-processing method, which shows the potential of dilated convolutions in aggregate refined information.

**Table 6.9:** Official results for the Vaihingen Dataset.

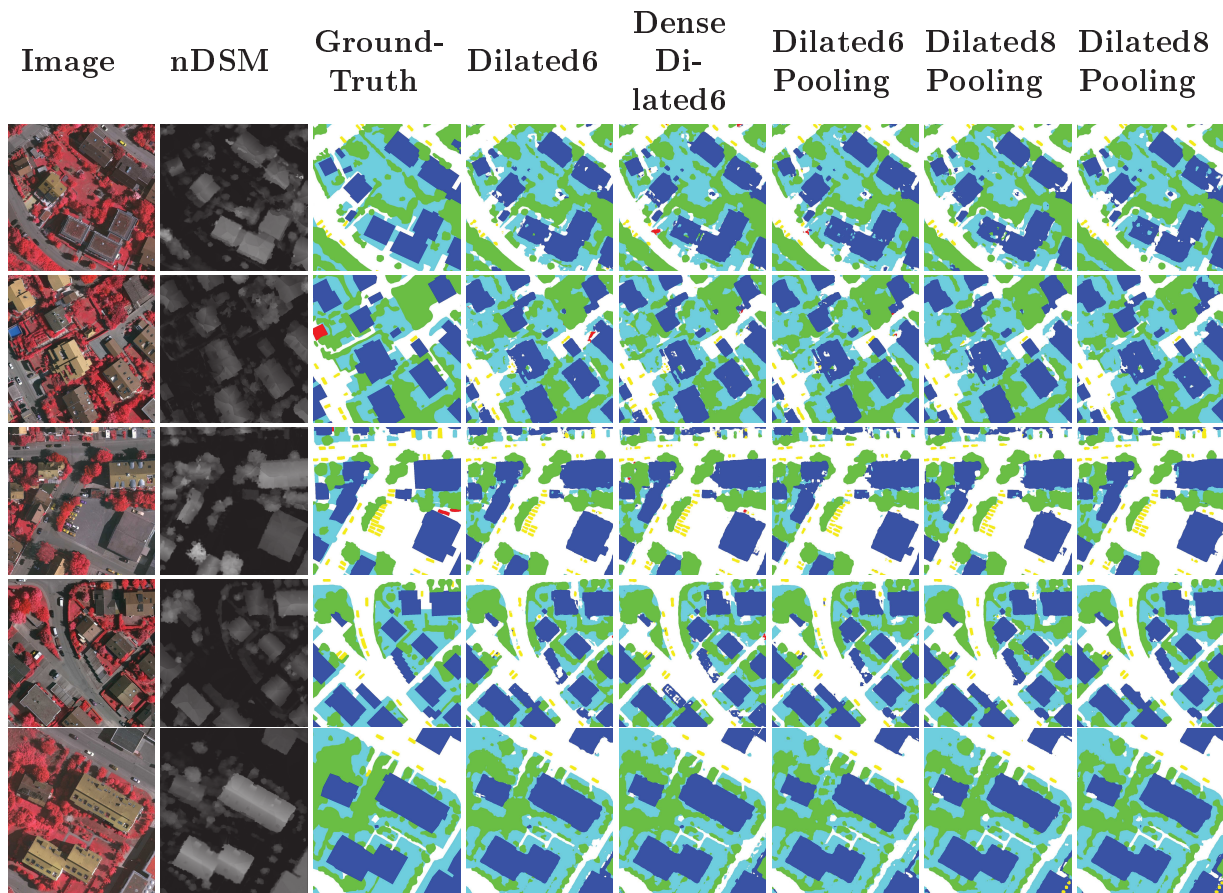
Method	#Parameters	F1 Score					Overall Accuracy
		Impervious Surface	Building	Low Vegetation	Tree	Car	
DLR_9 [Marmanis et al., 2018]	$806 \cdot 10^6$	92.4	95.2	83.9	89.9	81.2	90.3
GSN3 [Wang et al., 2017]	$44 \cdot 10^6$	92.3	95.2	84.1	90.0	79.3	90.3
ONE_7 [Audebert et al., 2016]	$28 \cdot 10^6$	91.0	94.5	84.4	89.9	77.8	89.8
INR [Maggiori et al., 2017]	$4 \cdot 10^6$	91.1	94.7	83.4	89.3	71.2	89.5
UFMG_4	$2 \cdot 10^6$	91.1	94.5	82.9	88.8	81.3	89.4
UFMG_5	$2 \cdot 10^6$	91.0	94.6	82.7	88.9	82.5	89.3
UFMG_1	$1.3 \cdot 10^6$	90.5	94.1	82.5	89.0	78.5	89.1
DST_2 [Sherrah, 2016]	$3.5 \cdot 10^6$	90.5	93.7	83.4	89.2	72.6	89.1
UFMG_2	$0.8 \cdot 10^6$	90.7	94.3	82.5	88.5	77.4	89.0
UFMG_3	$1.3 \cdot 10^6$	90.6	93.4	82.4	88.5	79.8	88.8
ADL_3 [Paisitkriangkrai et al., 2015]	$0.5 \cdot 10^6$	89.5	93.2	82.3	88.2	63.3	88.0
RIT_2 [Piramanayagam et al., 2016]	$138 \cdot 10^6$	90.0	92.6	81.4	88.4	61.1	88.0
RIT_L8 [Liu et al., 2017]	$134 \cdot 10^6$	89.6	92.2	81.6	88.6	76.0	87.8
UZ_1 [Volpi and Tuia, 2017]	$2.5 \cdot 10^6$	89.2	92.5	81.6	86.9	57.3	87.3

Aside from this, the proposed work (UFMG\_5) achieved the best result (82.5% of F1 Score) in the car class, which is one of the most difficult classes (of this dataset) when compared to others (such as building) because of its composition (small objects) and its high intraclass variance (caused by a great variety of models and colors). This may be justified by the fact that the proposed network does not downsample the input image preserving important details for such classes composed of small objects. However, this submission ignores the clutter/background class, which could be considered as an advantage, making the comparison unfair. But, there are other works doing the same training protocol (i.e., ignoring the clutter/background class), such as INR [Maggiori et al., 2017]. Yet such works have not achieved good accuracy in the car class as the proposed work. Furthermore, still considering the car class, the second best result (81.3% of F1 Score) is also yielded by our proposed work (UFMG\_4), which employs all classes during the training phase, which shows the effectiveness and robustness of our work mainly for classes related to small objects.

#### 6.3.7.4 Potsdam Dataset

As for the Vaihingen Dataset, official results for the Potsdam dataset are reported only by the challenge organization. For this dataset, we have four submissions, one for each network presented in Section 6.1 trained with the 6 classes (which are represented, in the official results, as UFMG\_1 to 4). In this dataset, there is no need to disregard the clutter/background class, since it has a sufficient amount of samples (4.96%). As before, all submissions related to the proposed work does not use any post-processing.

Table 6.10 summarizes some results reported by the organizers. Again, besides our results, the table also compiles the **best results of each work** with enough information to make a fair comparison. Visual examples of the proposed method, for



**Figure 6.7:** Example predictions for the validation set of the Vaihingen Dataset. Legend – White: impervious surfaces. Blue: buildings. Cyan: low vegetation. Green: trees. Yellow: cars. Red: clutter, background.

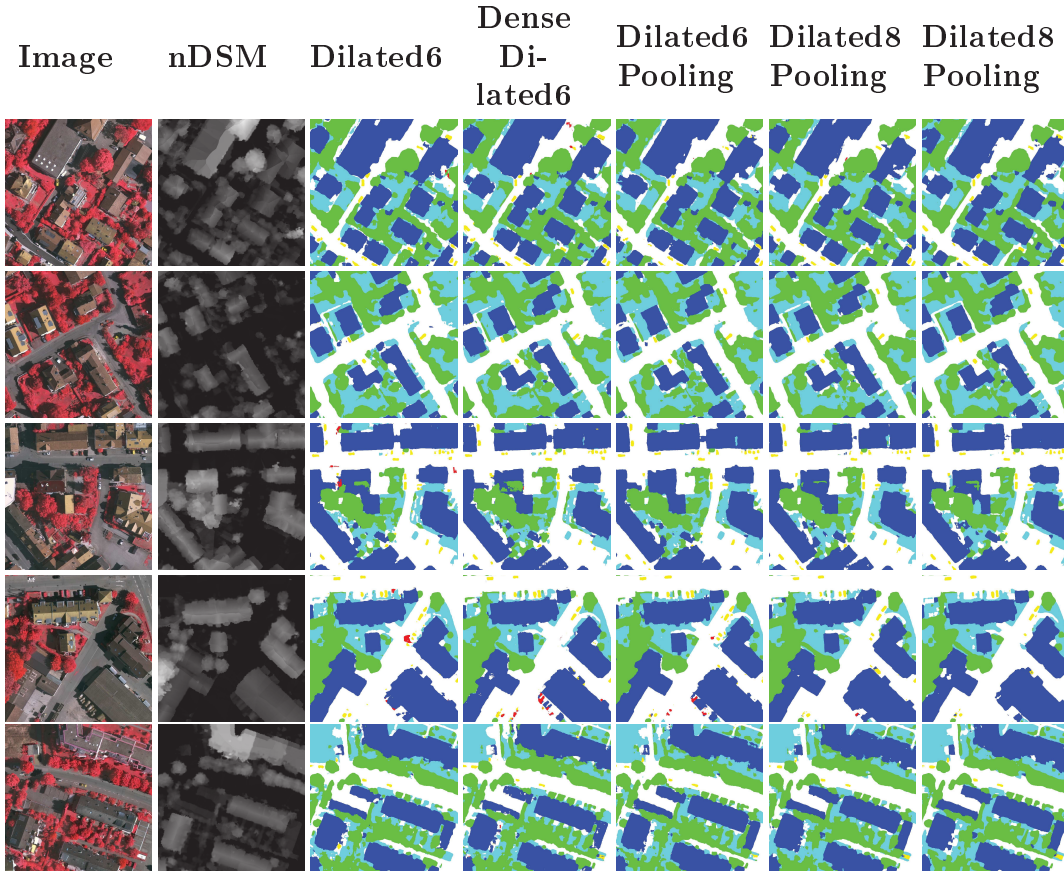
the validation and test sets, are presented in Figures 6.10 and 6.11, respectively.

The proposed work achieved competitive results, appearing in third place according to the overall accuracy. DST\_5 [Sherrah, 2016] and RIT\_L7 [Liu et al., 2017] are the best result in terms of overall accuracy. However, they have a larger number of trainable parameters when compared to our proposed networks, as seen in Figure 6.9b. This outcome corroborates with previous results, reaffirming obtained conclusions.

## 6.4 Conclusions

In this chapter, we propose a novel approach based on Convolutional Networks to perform pixel classification of remote sensing scenes. The method exploits networks composed uniquely of dilated convolution layers that do not downsample the input. Based on these networks and their no downsampling property, the proposed approach: (i) employs, in the training phase, patches of different sizes, allowing the networks to





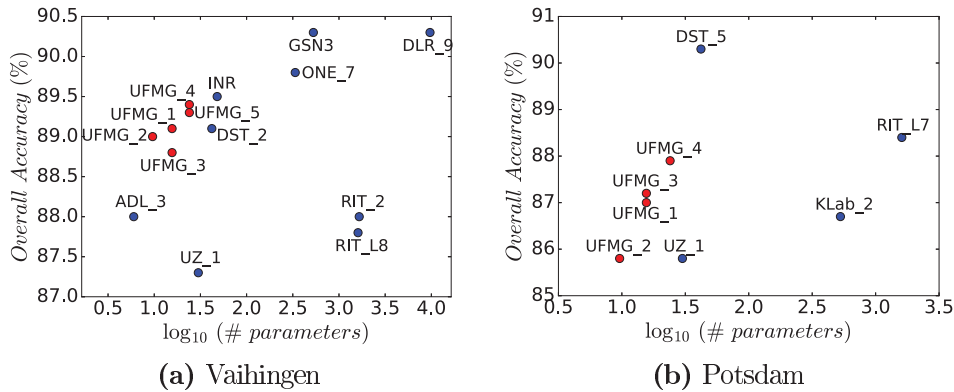
**Figure 6.8:** Example predictions for the test set of the Vaihingen Dataset. Legend – White: impervious surfaces. Blue: buildings. Cyan: low vegetation. Green: trees. Yellow: cars. Red: clutter, background.

**Table 6.10:** Official results for the Potsdam Dataset.

Method	#Parameters	F1 Score					Overall Accuracy
		Impervious Surface	Building	Low Vegetation	Tree	Car	
DST_5 [Sherrah, 2016]	$3.5 \cdot 10^6$	92.5	96.4	86.7	88.0	94.7	90.3
RIT_L7 [Liu et al., 2017]	$134 \cdot 10^6$	91.2	94.6	85.1	85.1	92.8	88.4
UFMG_4	$2 \cdot 10^6$	90.8	95.6	84.4	84.3	92.4	87.9
UFMG_3	$1.3 \cdot 10^6$	90.5	95.6	83.3	82.6	90.8	87.2
UFMG_1	$1.3 \cdot 10^6$	90.1	95.6	83.7	82.4	91.3	87.0
KLab_2 [Kemker et al., 2018]	$44 \cdot 10^6$	89.7	92.7	83.7	84.0	92.1	86.7
UFMG_2	$0.8 \cdot 10^6$	88.7	95.3	83.1	80.8	90.8	85.8
UZ_1 [Volpi and Tuia, 2017]	$2.5 \cdot 10^6$	89.3	95.4	81.8	80.5	86.5	85.8

capture multi-context characteristics given the distinct context size, and (ii) updates a score for each of these patch sizes in order to select the best one during the testing phase.

We performed experiments on four high-resolution remote sensing datasets with very distinct properties. Experimental results have showed that our method is effective and robust. It achieved state-of-the-art results in two datasets (Coffee and GRSS Data

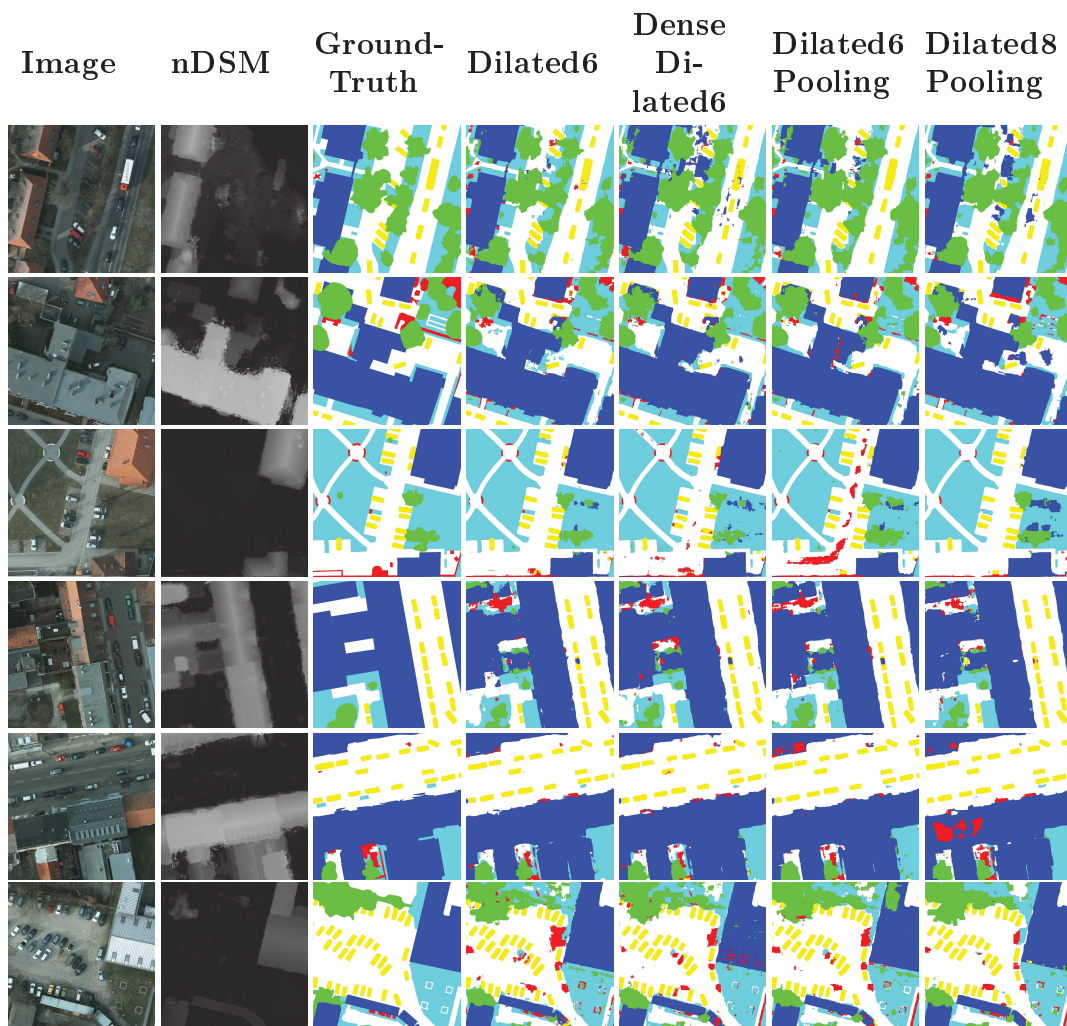


**Figure 6.9:** Comparison, in terms of overall accuracy and number of trainable parameters, between proposed and existing networks for Vaihingen and Potsdam Datasets. Ideal architectures should be in the top left corner, with fewer parameters but higher accuracy. Since the x axis is logarithmic, a change of only 0.3 in this axis is equivalent to more than 1 million new parameters in the model.

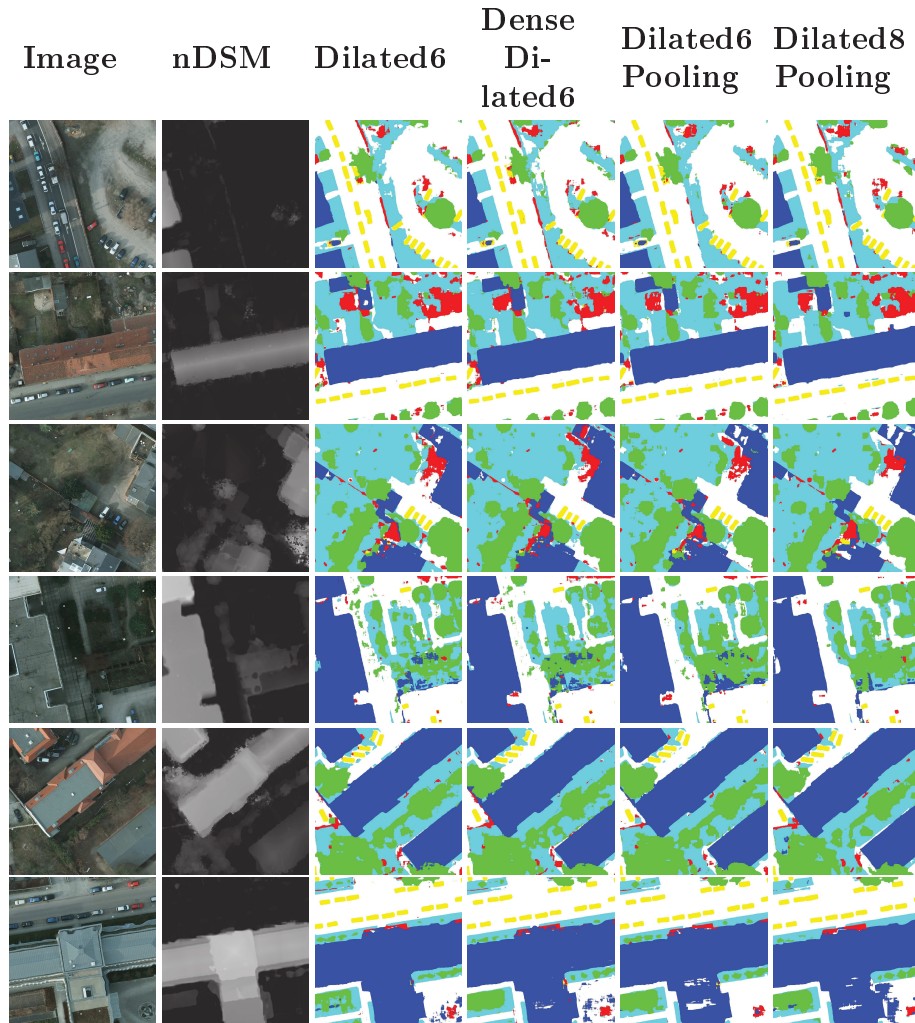
Fusion datasets) outperforming several techniques (such as Fully Convolutional [Long et al., 2015] and deconvolutional networks [Badrinarayanan et al., 2017]) that also exploit the multi-context paradigm. This shows the potential of the proposed method in learning multi-context information using patches of multiple sizes.

For the other datasets (Vaihingen and Potsdam), although the proposed technique did not achieve state-of-the-art, it yielded competitive results. In fact, our approach outperformed some relevant baselines that exploit post-processing techniques (although we did not employ any) and other multi-context strategies. Among all methods, the proposed one has the least number of parameters and is, therefore, less prone to overfitting and, consequently, easier to train. At the same time, it produces one of the highest accuracies, which shows the effectiveness of the proposed technique in extracting all feasible information from the data using limited (in terms of parameters) architectures. Furthermore, the proposed technique achieved one of the best results for the car class, which is one of the most difficult classes of these datasets because of its composition (small objects). This demonstrates the benefits of processing the input image without downsampling it, a process that preserves important details for classes that are composed of small objects.

Aside from this, the proposed networks can be fine-tuned for any semantic segmentation application, since they do not depend on the patch size to process the data. This allows other applications to benefit from the patterns extracted by our models, a very important process mainly when working with small amounts of labeled data [Nogueira et al., 2017c].



**Figure 6.10:** Example predictions for the validation set of the Potsdam Dataset. Legend – White: impervious surfaces. Blue: buildings. Cyan: low vegetation. Green: trees. Yellow: cars. Red: clutter, background.



**Figure 6.11:** Example predictions for the test set of the Potsdam Dataset. Legend – White: impervious surfaces. Blue: buildings. Cyan: low vegetation. Green: trees. Yellow: cars. Red: clutter, background.

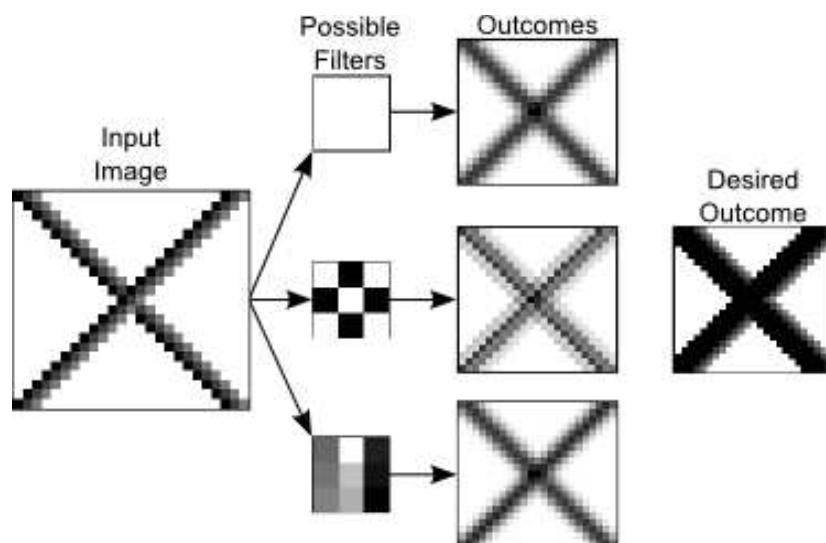
## Chapter 7

# An Introduction to Deep Morphological Networks

ConvNets [Goodfellow et al., 2016] achieved state-of-the-art in several applications, including image classification [Krizhevsky et al., 2012; Penatti et al., 2015], object and scene recognition [Long et al., 2015; Nogueira et al., 2015b; Noh et al., 2015; Badrinarayanan et al., 2017; Yu and Koltun, 2015; Nogueira et al., 2019b], and many others. This network is essentially composed of convolutional layers [Goodfellow et al., 2016] that process the input using optimizable filters, which are actually responsible to extract features from the data.

Despite using non-linear activation functions and pooling layers to bring some non-linearity to the learning process, those convolutional layers only perform linear operations over the input data, ignoring non-linear processes and the relevant information captured by them. For instance, one desires a basic network composed of just one neuron that always extracts the minimum of the neighborhood defined by its learnable filter. As presented in Figure 7.1, despite having a myriad of possible configurations, in this case, the ConvNet is not able to perform such non-linear operation and, therefore, may not be able to produce the expected output.

Concerning the image characteristics, non-linear operations are able to cope with some properties better than the linear ones, being preferable in some applications. Precisely, in some scenarios, such as the remote sensing one, images do not have a clear concept of perspective (i.e., fore and background) with all objects (or pixels) having equivalent importance. In these cases, borders and corners can be considered salient and fundamental features that should be preserved in order to help distinguish objects (mainly small ones). However, linear transformations (as performed by the convolutional layers) weight the pixels (with respect to the neighborhood) blurring



**Figure 7.1:** Illustration showing an input image (representing what can be seen as an x-shaped border), some ConvNet filters and the produced outputs generated by these filters. Note that none of the network outcomes is similar to the desired output, which was generated using a morphological erosion with a  $3 \times 3$  filter/structuring element, equal to the first one employed by the ConvNet. This shows that regardless of the filter, the Convolutional Network is not able to produce an output as desired due to its fully linear operations.

edges and losing this notion of borders and corners. Hence, in these applications, edge-aware filters, such as non-linear operations, can be considered a better option than linear ones [Tanizaki, 2013], since they are able to preserve those relevant features.

Supported by this property, some non-linear operations are still very popular being considered state-of-the-art in some applications, including in remote sensing ones. A successful non-linear filter are the morphological operations [Serra and Soille, 2012]. Such processes are considered effective tools to automatic extract features but preserving essential characteristics (such as corners and borders), being still widely employed and current state-of-the-art in several applications in which such properties are considered fundamental [Dalla Mura et al., 2010; Xia et al., 2015; Kimori et al., 2016; Seo et al., 2018]. Although successful in some scenarios, morphological operations have a relevant drawback: a structuring element (filter used to define the neighborhood that must be taken into account during the processing) must be defined and provided for the process. In typical scenarios, since different structuring elements may produce distinct results depending on the data, it is imperative to design and evaluate many structuring elements in order to find the most suitable ones for each application, an expensive process that does not guarantee a good descriptive representation.

Encouraged by the current scenario, in this chapter, we propose a novel method for deep feature learning, called Deep Morphological Network (DeepMorphNet), which is capable of doing non-linear morphological operations while performing the feature learning step (by optimizing the structuring elements). For simplicity, in this work, only

morphological operations based on binary SEs that operate over one single channel at a time were considered. This new approach, strongly based on the ConvNets (because of the similarity between the operation performed by the morphological transformations and convolutional layers), would aggregate the benefits of morphological operations while overcoming the aforementioned drawback by learning the structuring element during the training process. Particularly, the processing of each layer of the proposed technique can be divided into three steps: (i) the first one employs depthwise convolutions [Chollet, 2017] to rearrange the input pixels according to the binary filters (that represent the structuring elements), (ii) the second one uses depthwise pooling to select the pixel (based on erosion and dilation operations) and generate an eroded or dilated outcome, and (iii) the last one employs pointwise convolutions [Chollet, 2017] to combine the generated maps in order to produce one final morphological map (per neuron). This whole process resembles the depthwise separable convolutions [Chollet, 2017] but using binary filters and one more step (depthwise pooling) between the convolutions.

## 7.1 Deep Morphological Networks

In this section, we present the proposed network, called Deep Morphological Networks (or simply DeepMorphNets), capable of doing morphological operations while optimizing the structuring elements. Technically, this new network is strongly based on ConvNets mainly because of the similarity between the morphological operations and convolutional layers, since a significant analogy can be made between the processing operations performed by these transformations (i.e., both probe the image with a sliding window that defines the neighborhood of interest). Therefore, this new network seeks to efficiently combine morphological operations and deep learning, aggregating the ability to learn certain important types of image properties (such as borders and corners) of the former and the feature learning step of the latter. Such combination would bring advantages that could assist several applications in which borders and shape are considered essential. However, there are several challenges in fully integrating morphological operations and deep learning-based methods, especially convolutional neural networks.

In this specific case, a first challenge is due to the convolutional layers and their operations. Precisely, such layers, the basis of ConvNets, extract features from the input data using an optimizable filter by performing only linear operations not supporting non-linear ones. Formally, let us assume a 3D input  $y(\cdot)$  of a convolutional layer as a mapping from coordinates ( $\mathbb{Z}^3$ ) to the pixel-value domain ( $\mathbb{Z}$  or  $\mathbb{R}$ ). Anal-

ogously, the trainable filter (or weight)  $W(\cdot)$  of such layer can be seen as a mapping from 3D coordinates ( $\mathbb{Z}^3$ ) to the real-valued weights ( $\mathbb{R}$ ). A standard convolutional layer performs a convolution (denoted here as  $*$ ) of the filter  $W(\cdot)$  over the input  $y(\cdot)$ , according to Equation 7.1. Note that the output of this operation is the summation of the linear combination between input and filter (across both space and depth). Also, observe the difference between this operation and the morphological ones stated in Section 3.2. This shows that the integration between morphological operations and convolutional layers is not straightforward.

$$S(W, y)_{(i,j)} = (W * y)(i, j) = \sum_m \sum_n \sum_l W(m, n, l) y(i + m, j + n, l) \quad (7.1)$$

Another important challenge is due to the optimization of non-linear operations by the network. Technically, in **ConvNets**, a loss function  $\mathcal{L}$  is defined to allow the evaluation of the network's current state and its optimization towards a better state. Nevertheless, the objective of any network is to minimize this loss function by adjusting the trainable parameters (or filters)  $W$ . Such optimization is traditionally based on the derivatives of the loss function  $\mathcal{L}$  w.r.t. the weights  $W$ . For instance, suppose Stochastic Gradient Descent (SGD) [Goodfellow et al., 2016] is used to optimize a **ConvNet**. As presented in Equation 7.2, the optimization of the filters (towards a better state) depends directly on the partial derivatives of the loss function  $\mathcal{L}$  w.r.t. the weights  $W$  (employed with a pre-defined learning rate  $\alpha$ ). Those partial derivatives are usually obtained using the backpropagation algorithm [Goodfellow et al., 2016], which is strongly supported by the fact that all operations of the network are easily differentiable (w.r.t. the filters), including the convolution presented in Equation 7.1 (since, it is only a linear combination). However, this algorithm can not be directly applied to non-linear operations, such as the presented morphological ones, because those operations do not have easy derivatives.

$$W = W - \alpha \frac{\partial \mathcal{L}}{\partial W} \quad (7.2)$$

Overcoming such challenges, we propose a novel network, based on **ConvNets**, that employs depthwise and pointwise convolution with depthwise pooling to recreate and optimize morphological operations, from basic to complex ones. First, Section 7.1.1 introduce the basic concepts used as a foundation for the proposed Deep Morphological Network. Section 7.1.2 presents the proposed neurons responsible to perform morphological operations. The proposed morphological layer, composed of the proposed neurons, is presented in Section 7.1.3. The optimization of the filters



(also called structuring elements) of such layers is explained in Section 7.1.4. Finally, the proposed [DeepMorphNet](#) architectures are introduced in Section 7.1.5.

### 7.1.1 Basic Morphological Framework

The combination of morphological operations and deep learning is subject to an essential condition: the new technique should be capable of conserving the end-to-end learning strategy, i.e., it should be able to blend with the current training procedure. The reason for this condition is two-folded: (i) to extract the benefits of the feature learning step (i.e., optimization of the filters) from deep learning, and (ii) to allow the combination of morphological operations with any other existing operation explored by deep learning-based approaches. Towards such objective, we have proposed a new framework, capable of performing morphological erosion and dilation, based on operations that meet this condition, i.e., neurons based on this framework can be easily integrated into the standard training process. The processing of this framework can be separated into two steps. The **first** one employs depthwise convolution [[Chollet, 2017](#)] to perform a delimitation of features, based on the neighborhood (or filter). As defined in Equation 7.3, this type of convolution differs from standard ones since it handles the input depth independently, using the same filter  $W$  to every input channel. In other words, suppose that a layer performing depthwise convolution has  $k$  filters and receives an input with  $l$  channels, then the processed outcome would be an image of  $k \times l$  channels, since each  $k$ -th filter would be applied to each  $l$ -th input channel. The use of depthwise convolution simplifies the introduction of morphological operations into the deep network since the linear combination performed by this convolution does not consider the depth (as in standard convolutions presented in Equation 7.1). This process is fundamental for the recreation of the introduced morphological operations since such transformations can only process one single channel at a time (as aforementioned in Section 3.2). Although there are morphological operations capable of handling more than one channel at a time [[Chevallier et al., 2015](#)], in this work, as introduced, only operations based on binary SEs that operate over one single channel at a time were considered.

$$S_l(W, y)_{(i,j)} = \sum_m \sum_n W(m, n) y(i + m, j + n, l) \quad (7.3)$$

However, just using this type of convolution does not allow the reproduction of morphological transformations, given that a spatial linear combination is still performed by this convolutional operation. To overcome this, all filters  $W$  are first con-

verted into binary and then used in the depthwise convolution operation. Precisely, this binarization process, referenced hereafter as max-binarize, activates only the highest value of the filter, i.e., only the highest value is considered active, while all others are deactivated. Formally, the max-binarize  $b(\cdot)$  is a function that receives as input the real-valued weights  $W$  and processes them according to Equation 7.4, where  $\mathbb{1}\{condition\}$  is the indicator function, that returns 1 if the *condition* is true and 0 otherwise. This process outputs a binary version of the weights, denoted here as  $W^b$ , in which **only** the highest value (in  $W$ ) is activated (in  $W^b$ ). By using this binary filter  $W^b$ , the linear combination performed by depthwise convolution can be seen as a simple operation that preserves the exact value of the single pixel activated by this binary filter.

$$W_{(i,j)}^b = b(W(i,j)) = \mathbb{1}\{max_{k,j}(W(k,j)) = W(i,j)\} \quad (7.4)$$

However, only preserving one pixel w.r.t. the binary filter is not enough to reproduce the morphological operations, since they usually operate over a neighborhood (defined by the SE  $B$ ). In order to reproduce this neighborhood concept in the depthwise convolution operation, we decompose each filter  $W$  into several ones, that when superimposed retrieve the final SE  $B$ . More specifically, suppose a filter  $W$  with size  $s \times s$ . Since only one position can be activated at a time (because of the aforementioned binarization process), this filter has a total of  $s^2$  possible activation variations. Suppose also a SE with size  $s \times s$ . As explained in Section 3.2, such SE defines the pixel neighborhood and can have any feasible configuration. Considering each position of this SE independently, each one can be considered activated (when that position of the neighborhood should be taken into account) or deactivated (when the neighboring position should not be taken into account). Therefore, a SE of size  $s \times s$  has  $s^2$  possible configurations when considering each position separately. Based on all this, a set of  $s^2$  max-binary filters with size  $s \times s$  is able to cover all possible configurations of a SE with the same size, i.e., with this set, it is possible to recreate any feasible configuration of a  $s \times s$  SE. Precisely, a set of  $s^2$  filters with size  $s \times s$  can be seen as a decomposed representation of the neighborhood concept (or of the SE) given that those  $s^2$  filters (with only a single activated position) can be superimposed in order to retrieve any possible  $s \times s$  neighborhood configuration defined by the SE. Supported by this idea, any  $s \times s$  SE can be decomposed into  $s^2$  filters, each one with size  $s \times s$  and only one activated value. By doing this, the concept of neighborhood introduced by the SE can be exploited in depthwise convolution. Particularly, a  $s^2$  set of  $s \times s$  filters  $W$  can be converted into binary weights  $W^b$  (via the aforementioned max-binarize function  $b(\cdot)$ ) and then, used to process the input data. When exploited by Equation 7.3, each of

these  $s^2$  binary filter  $W^b$  will preserve only one pixel which is directly related to one specific position of the neighborhood. Thus, technically, this first step recreates, in depth, the neighborhood of a pixel delimited by a  $s \times s$  SE  $B$ , which is essentially represented by  $s^2$  binary filters  $W^b$  of size  $s \times s$ .

$$B_{(i,j)} = \mathbf{1}\left\{\sum_l W^b(i,j,l) \geq 1\right\} \quad (7.5)$$

Since the SE  $B$  was decomposed in depth, in order to retrieve it, a depthwise operation, presented in Equation 7.5, must be performed over the  $s^2$  binary filters  $W^b$ . Analogously, a depthwise operation is also required to retrieve the final outcome, i.e., the eroded or dilated image. This is the **second** step of this proposed framework, which is responsible to extract the relevant information based on the depthwise neighborhood. In this step, an operation, called depthwise pooling  $P(\cdot)$ , performs a pixel and depthwise process over the  $s^2$  outcomes (of the decomposed filters), producing the final morphological outcome. This pooling operation is able to actually output the morphological erosion and dilation by using pixel and depthwise minimum and maximum functions, as presented in Equations 7.6 and 7.7, respectively. Note that the reproduction of morphological operations using minimum and maximum functions is only possible because the set created with each pixel position along the channels can be considered an ordered set (similar to the definition presented in Section 3.2). The outcome of this second step is the final (eroded or dilated) feature map that will be exploited by any subsequent process.

$$P^{\mathcal{E}}(y)_{(i,j)} = \min_l y(i,j,l) \quad (7.6)$$

$$P^{\delta}(y)_{(i,j)} = \max_l y(i,j,l) \quad (7.7)$$

Equations 7.8 and 7.9 compile the two steps performed by the proposed framework for morphological erosion and dilation, respectively. This operation, denoted here as  $M(\cdot)$ , performs a depthwise convolution (first step), which uses (binary) filters that decompose the representation of the neighborhood concept introduced by SEs, followed by a pixel and depthwise pooling operation (second step), outputting the final morphological (eroded or dilated) feature maps. Note the similarity between these functions and Equations 3.12 and 3.13 presented in Section 3.2. The main difference between these equations is in the neighborhood definition. While in the standard morphology, the neighborhood of a pixel is defined spatially (via SE  $B$ ), in the proposed framework, the neighborhood is defined along the channels due to the decomposition of the SE  $B$  into several filters and, therefore, minimum and maximum operations also operate over

the channels.

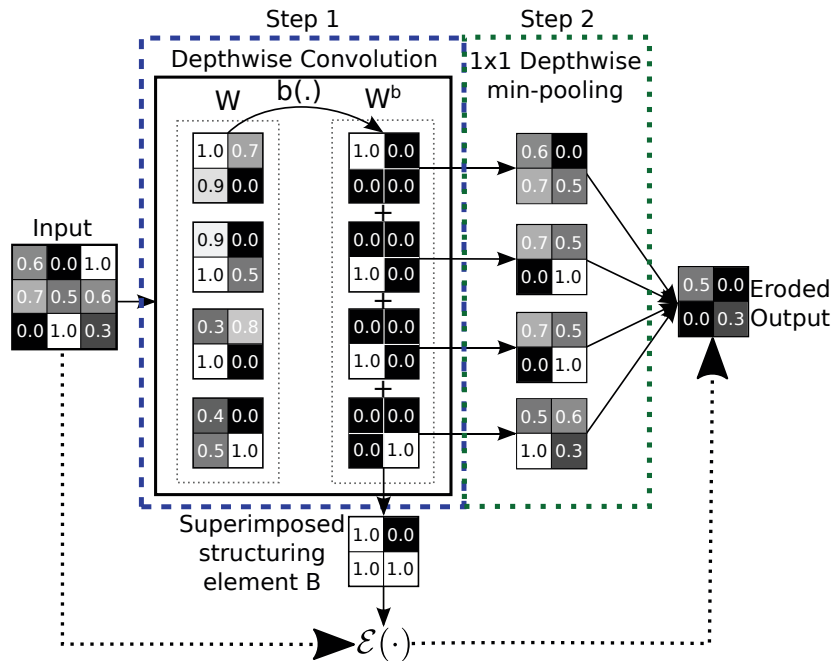
$$M^{\mathcal{E}}(W, y)_{(i,j)} = P^{\mathcal{E}}(S_l(W, y))_{(i,j)} = \min_l S_l(W, y)_{(i,j)} \quad (7.8)$$

$$M^{\delta}(W, y)_{(i,j)} = P^{\delta}(S_l(W, y))_{(i,j)} = \max_l S_l(W, y)_{(i,j)} \quad (7.9)$$

A visual example of the proposed framework being used for morphological erosion is presented in Figure 7.2. In this example, the depthwise convolution has 4 filters  $W$  with size  $4 \times 4$  which actually represent a unique  $4 \times 4$  SE. The filters  $W$  are first converted into binary using the max-binarize function  $b(\cdot)$ , presented in Equation 7.4. Then, each binary filter  $W^b$  is used to process (step 1, blue dashed rectangle) each input channel (which, for simplicity, is only one in this example) using Equation 7.3. In this process, each binary filter  $W^b$  outputs an image in which each pixel has a direct connection to the one position activated in that filter (that, actually, represents a neighborhood position activated in the SE  $B$ ). The output is then processed (step 2, green dotted rectangle) via a pixel and depthwise min-pooling  $P(\cdot)^{\mathcal{E}}$  (according to Equation 7.6) to produce the final eroded output. Note that the binary filters  $W^b$ , when superimposed (using Equation 7.5), retrieve the final SE  $B$ . The dotted line shows that the processing of the input with the superimposed SE  $B$  using the standard morphological erosion ( $\mathcal{E}(\cdot)$  presented in Equation 3.12) results in the same eroded output image produced by the proposed morphological erosion.

### 7.1.2 Morphological Processing Units

The proposed framework is the foundation of all proposed morphological processing units (or neurons). However, although the proposed framework is able to reproduce morphological erosion and dilation, it has an important drawback: since it employs depthwise convolution, the number of outcomes can grow polynomially, given that, as previously explained, each input channel is processed independently by each processing unit. Thus, in order to overcome this issue and make the proposed technique more scalable, we propose to use a pointwise convolution [Chollet, 2017] to force each processing unit to output only one image (or feature map). Particularly, any neuron proposed in this work has the same design with two parts: (i) the core operation (fundamentally based on the proposed framework), in which the processing unit performs its morphological transformation outputting multiple outcomes, and (ii) the pointwise convolution [Chollet, 2017], which performs a pixel and depthwise (linear) combination of the outputs producing only one outcome. Observe that though the pointwise convolution performs a depthwise combination of the multiple outcomes, it does not

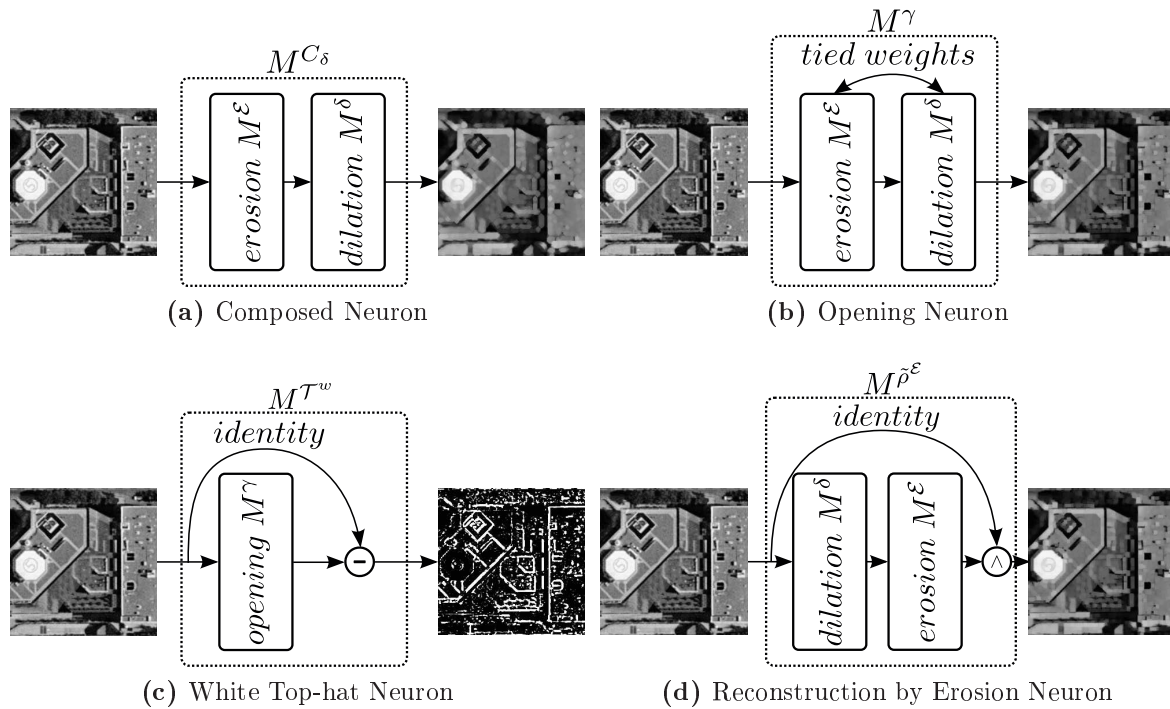


**Figure 7.2:** Example of a morphological erosion based on the proposed framework. The 4 filters  $W$  (with size  $4 \times 4$ ) actually represent a unique  $4 \times 4$  SE. Each filter  $W$  is first converted to binary  $W^b$ , and then used to process each input channel (step 1, blue dashed rectangle). The output is then processed via a pixel and depthwise min-pooling to produce the final eroded output (step 2, green dotted rectangle). Note that the binary filters  $W^b$ , when superimposed, retrieve the final SE  $B$ . The dotted line shows that the processing of the input with the superimposed SE  $B$  using the standard morphological erosion results in the same eroded output image produced by the proposed morphological erosion.

learn any spatial feature, since it employs a pixelwise (or pointwise) operation, managing each pixel separately. This design allows the morphological neuron to have the exact same input and output of a standard existing processing unit, i.e., it receives as input an image with any number of bands and outputs a single new representation. It is interesting to notice that this processing unit design employs depthwise and pointwise convolution [Chollet, 2017], resembling very much the depthwise separable convolutions [Chollet, 2017], but with extra steps and binary decomposed filters. Next Sections explain the **core operation** of all proposed morphological processing units. Note that these neurons were conceived to be equivalent in terms of operations. Therefore, all of them have, exactly, two operations (based on the proposed framework). Also observe that, although not mentioned in the next Sections, the pointwise convolution is present in all processing units proposed in this work.

### 7.1.2.1 Composed Processing Units

The newly introduced framework allows a deep network to perform erosion and dilation, the two basic operations of morphology. However, instead of using such operations independently, the first proposed processing unit is based on both morphological trans-



**Figure 7.3:** Definition of morphological neurons based on the proposed framework. In this case, just one type of each neuron is represented. Some neurons have tied weights to force the network to use the same filters (i.e. SE) in both operations. Other neurons have skip connections in order to allow a new operation using the original input and the processed data.

formations. The so-called composed processing units, which are totally based on the proposed framework, have in their core a morphological erosion followed by a dilation (or vice-versa), without any constraint on the weights (i.e., on the SE). The motivation behind the composed processing unit is based on the potential of the learned representation. While erosion and dilation can learn simple representations, the combination of these operations is able to capture more complex information. Formally, Equations 7.10 and 7.11 present the two possible configurations of the morphological composed neurons. It is important to notice that the weights ( $W_1$  and  $W_2$ ) of each operation of this neuron are independent. Aside from this, a visual representation of one type of composed neuron can be seen in Figure 7.3a.

$$M^{C_\delta}(W, y) = M^\delta(W_2, M^\epsilon(W_1, y)) \quad (7.10)$$

$$M^{C_\epsilon}(W, y) = M^\epsilon(W_2, M^\delta(W_1, y)) \quad (7.11)$$

### 7.1.2.2 Opening and Closing Processing Units

Aside from implementing morphological erosion and dilation, the proposed framework is also able to support the implementation of other, more complex, morphological

operations (or their approximations). The most intuitive and simple transformations to be implemented are the opening and closing. As stated in Section 3.2, an opening is simply an erosion operation followed by a dilation of the eroded output (Equation 3.14), while closing is the reverse operation (Equation 3.15). In both cases, the two basic operations (erosion and dilation or vice-versa) use the same SE  $B$ . Based on this, the implementation of the opening and closing processing units, using the proposed framework, is straightforward. Precisely, the core of such neurons is very similar to that of the composed processing units, except that in this case a tie on the filters of the two basic morphological operations is required in order to make them use the same weights, i.e., the same SE  $B$ . A visual representation of the proposed opening neuron, presented in Figure 7.3b, allows a better view of the operation. Formally, Equations 7.12 and 7.13 define the opening and closing morphological neurons, respectively. Note the similarity between these functions and Equations 3.14 and 3.15.

$$M^\gamma(W, y) = M^\delta(W, M^\mathcal{E}(W, y)) \quad (7.12)$$

$$M^\varphi(W, y) = M^\mathcal{E}(W, M^\delta(W, y)) \quad (7.13)$$

### 7.1.2.3 Top-hat Processing Units

The implementation of other, more complex, morphological operations is a little more tricky. This is the case of the top-hat operations, which require both the input and processed data to generate the final outcome. Therefore, for such operations, a skip connection [Goodfellow et al., 2016; He et al., 2016] (based on the identity mapping) is employed to support the forwarding of the input data, allowing it to be further processed. The core of the top-hat processing units is composed of three parts: (i) an opening or closing morphological processing unit (depending on the type of the top-hat), (ii) a skip connection, that allows the forwarding of the input data, and (iii) a subtraction function that operates over the data of both previous parts, generating the final outcome. A visual concept of the white top-hat neuron is presented in Figure 7.3c. Such operation and its counterpart (the black top-hat) are formally defined in Equations 7.14 and 7.15, respectively.

$$M^{\mathcal{T}^w}(W, y) = y - M^\gamma(W, y) \quad (7.14)$$

$$M^{\mathcal{T}^b}(W, y) = M^\varphi(W, y) - y \quad (7.15)$$

### 7.1.2.4 Geodesic Reconstruction Processing Units

Similarly to the previous processing units, the geodesic reconstruction also requires the input and processed data in order to produce the final outcome. Hence, the implementation of this important operation is also based on skip connections. Aside from this, as presented in Section 3.2, reconstruction operations require an iterative process. Although this procedure is capable of producing better outcomes, its introduction in a deep network is not straightforward (given that each process can take a different number of iterations). Supported by this, the reconstruction processing units proposed in this work are an approximation, in which just one transformation over the marker image is performed (and not several iterations). Particularly, the input is processed by two basic morphological operations (without any iteration) and an elementwise max- or min-operation (depending on the reconstruction) is performed over the input and processed images. Such concept is formally presented in Equations 7.16 and 7.17 for reconstruction by erosion and dilation, respectively. A visual representation of the processing unit for reconstruction by erosion is presented in Figure 7.3d. Note that the SE used in the reconstruction of the marker image (denoted in Section 3.2 by  $B'$ ) is a dilated version of the SE employed to create such image, i.e., the SE exploited in the second morphological operation is a dilated version of the SE employed in the first transformation.

$$M^{\tilde{\rho}^{\mathcal{E}}}(W, y) = M_y^{\mathcal{E}}(W, M^{\delta}(W, y)) \quad (7.16)$$

$$M^{\tilde{\rho}^{\delta}}(W, y) = M_y^{\delta}(W, M^{\mathcal{E}}(W, y)) \quad (7.17)$$

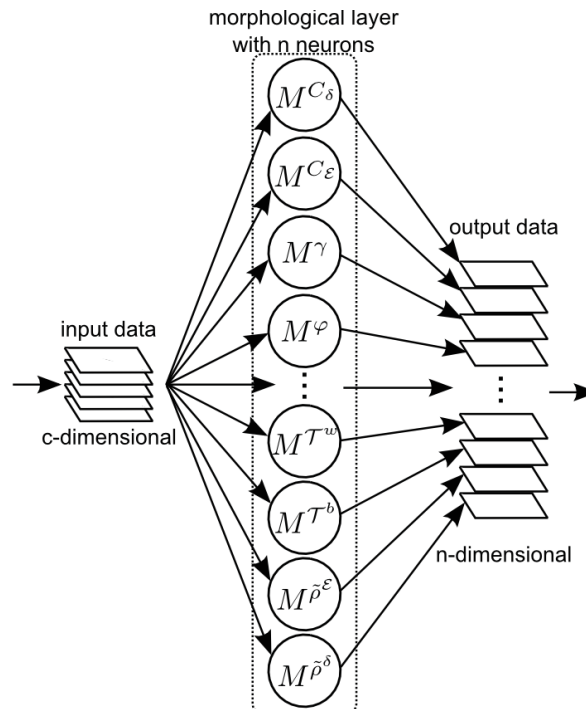
## 7.1.3 Morphological Layer

After defining the processing units, we are able to formally specify the morphological layers, which provide the essential tools for the creation of the DeepMorphNets. Similar to the standard convolutional layer, this one is composed of several processing units. However, the proposed morphological layer has two main differences when conceptually compared to the standard one. The first one is related to the neurons that compose the layers. Precisely, in convolutional layers, the neurons are able to perform the convolution operation. Though the filter of each neuron can be different, the operation performed by each processing unit in a convolutional layer is a simple convolution. On the other hand, there are several types of morphological processing units, from opening and closing to geodesic reconstruction. Therefore, a single morphological layer can be composed of several neurons that may be performing different



operations. This process allows the layer to produce distinct (and possibly complementary) outputs, increasing the heterogeneity of the network and, consequently, the generalization capacity. The second difference is the absence of activation functions. More specifically, in modern architectures, convolutional layers are usually composed of a convolution operation followed by an activation function (such as ReLU [Nair and Hinton, 2010]), that explicitly maps the data into a non-linear space. In morphological layers, there are only processing units and no activation function is employed. This is due to the fact that such functions may change drastically the values of the pixels, an undesired effect in the mathematical morphology.

Figure 7.4 presents the concept of a single morphological layer. Observe that each neuron is performing a specific operation and outputting only one feature map. Also, note that, although the input has  $c$  channels, supported by the pointwise convolution, each neuron outputs only one feature map. Hence, the number of outputted maps is directly connected to the number of neurons in that layer. In Figure 7.4, the layer has  $n$  neurons that, consequently, produce  $n$  feature maps.



**Figure 7.4:** Concept of a morphological layer. Note that a single morphological layer can have neurons performing different operations. This process is able to aggregate heterogeneous and complementary information.

### 7.1.4 Optimization

Aside from defining the morphological layer, as introduced, we must optimize its parameters, i.e., the filters  $W$ . Since the proposed morphological layer uses common (derivable) operations already employed in other existing deep learning-based methods, the optimization of the filters is straightforward. In fact, the same traditional existing techniques employed in the training of any deep learning-based approach, such feed-forward, backpropagation and Stochastic Gradient Descent (SGD) [Goodfellow et al., 2016], can also be used for optimizing a network composed of morphological layers.

The whole training procedure is detailed in Algorithm 2. Given the training data  $(y_0, y^*)$ , the **first step** is the feedforward, comprised in the loop from line 2 to 12. In line 4, the weights of the first depthwise convolution are converted into binary (according to Equation 7.4). Then, in line 5, the first depthwise convolution is performed, while the first depthwise pooling is executed in line 6. The same operations are repeated in line 8 to 10 for the second depthwise convolution and pooling. Finally, in line 11, the pointwise convolution is carried out. After the forward propagation, the total error of the network can be estimated. With this error, the gradients of the last layer can be directly estimated (line 14). These gradients can be used by the backpropagation algorithm to calculate the gradients of the inner layers. In fact, this is the process performed in the **second training step**, comprised in the loop from line 15 to 22. It is important to highlight that during the backpropagation process, the gradients are calculated normally, using real-valued numbers (and not binary). Precisely, lines 16 and 17 are responsible for the optimization of the pointwise convolution. Line 16 propagates the error of a specific pointwise convolution to the previous operation, while in line 17 calculates the error of that specific pointwise convolution operation. The same process is repeated for the second and then for the first depthwise convolutions (lines 18-19 and 20-21, respectively). Note that during the backpropagation, the depthwise pooling is not optimized since this operation has no parameters and only passes the gradients to the previous layer (similar to the backpropagation employed in the max-pooling layers commonly explored in ConvNets). The **third and last step** of the training process is the update of the weights and optimization of the network. This process is comprised in the loop from line 24 to 28. Observe that, for simplicity, Algorithm 2 uses SGD to optimize the network, however, any other optimization algorithm could be exploited. For a specific layer, line 25 updates the weights of the pointwise convolution while lines 26 and 27 update the parameters of the first and second depthwise convolutions, respectively.

**ALGORITHM 2**

Training a Deep Morphological Network with L layers.

**Require:** a minibatch of inputs and targets  $(y_0, y^*)$ , previous weights  $W$ , and previous learning rate  $\alpha$ .**Ensure:** updated weights  $W$ .

- 
- 1: 1. Forward propagation:
  - 2: **para** k=1 to L **faça**
  - 3: {First Processing Unit Operation}
  - 4:  $W_k^{b(1)} \leftarrow b(W_k^{(1)})$  {Binarization}
  - 5:  $s_k^{(1)} \leftarrow y_{k-1} * W_k^{b(1)}$  {Depthwise Convolution}
  - 6:  $y_k^{(1)} \leftarrow P(s_k^{(1)})$  {Depthwise Pooling}
  - 7: {Second Processing Unit Operation}
  - 8:  $W_k^{b(2)} \leftarrow b(W_k^{(2)})$  {Binarization}
  - 9:  $s_k^{(2)} \leftarrow y_k^{(1)} * W_k^{b(2)}$  {Depthwise Convolution}
  - 10:  $y_k^{(2)} \leftarrow P(s_k^{(2)})$  {Depthwise Pooling}
  - 11:  $y_k \leftarrow y_k^{(2)} * W_k^{(1 \times 1)}$  {Pointwise Convolution}
  - 12: **fim para**
  - 13: 2. Backpropagation: {Gradients are not binary.}
  - 14: Compute  $g_{y_L^{(1)}} = \frac{\partial \mathcal{L}}{\partial y_L}$  knowing  $y_L$  and  $y^*$
  - 15: **para** k=L to 1 **faça**
  - 16:  $g_{y_{k-1}} \leftarrow g_{y_k^{(1)}} W_{k-1}^{(1 \times 1)}$
  - 17:  $g_{W_{k-1}^{(1 \times 1)}} \leftarrow g_{y_k^{(1)}}^\top y_{k-1}$
  - 18:  $g_{y_{k-1}^{(2)}} \leftarrow g_{y_{k-1}} W_{k-1}^{b(2)}$
  - 19:  $g_{W_{k-1}^{b(2)}} \leftarrow g_{y_{k-1}^{(2)}}^\top y_{k-1}^{(2)}$
  - 20:  $g_{y_{k-1}^{(1)}} \leftarrow g_{y_{k-1}^{(2)}} W_{k-1}^{b(1)}$
  - 21:  $g_{W_{k-1}^{b(1)}} \leftarrow g_{y_{k-1}^{(1)}}^\top y_{k-1}^{(1)}$
  - 22: **fim para**
  - 23: 3. Update the weights (via SGD):
  - 24: **para** k=1 to L **faça**
  - 25:  $W_k^{(1 \times 1)} \leftarrow W_k^{(1 \times 1)} - \alpha g_{W_k^{(1 \times 1)}}$
  - 26:  $W_k^{(1)} \leftarrow W_k^{(1)} - \alpha g_{W_k^{b(1)}}$
  - 27:  $W_k^{(2)} \leftarrow W_k^{(2)} - \alpha g_{W_k^{b(2)}}$
  - 28: **fim para**
-

### 7.1.5 DeepMorphNet Architecture

With all the fundamentals defined, we can finally specify the **DeepMorphNet** architectures exploited in this work. Particularly, three networks, composed of morphological and fully connected layers, were proposed for image classification. Although such architectures have distinct designs, the pointwise convolutions exploited in the morphological layers have always the same configuration: kernel  $1 \times 1$ , stride 1, and no padding. This is due to the fact this configuration does not allow the layer to learn and extract spatial information (what should be performed by the morphological layers) but only a pixel-level combination. Furthermore, all networks use cross-entropy as loss function and Stochastic Gradient Descent as optimization algorithm [Goodfellow et al., 2016].

The first network, presented in Figure 7.5a, is the simplest one, having just a unique layer composed of one morphological opening  $M^\gamma$  (with kernel size of  $11 \times 11$ , stride 1 and padding 5 for both depthwise convolutions). In fact, this architecture was only designed to be used with the proposed synthetic datasets (as introduced in Section 7.2.1.1). Because of this, such network is referenced hereafter as **DeepMorphSynNet**. Technically, this network was only conceived to validate the learning process of the proposed framework as explained in Section 7.3.1.

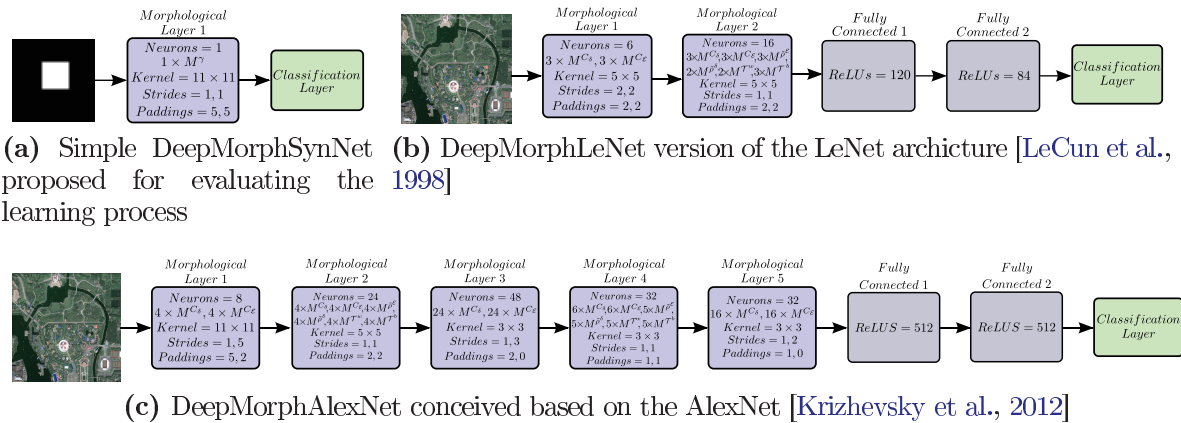
The second proposed network, presented in Figure 7.5b, is a morphological version of the famous LeNet architecture [LeCun et al., 1998]. In virtue of this, such network is called here as **DeepMorphLeNet**. Formally, this architecture is composed of two morphological and three fully connected layers. The first layer has 6 neurons equally divided into the two types of composed processing units ( $M^{C^\varepsilon}$  and  $M^{C^\delta}$ ). The processing units of this first layer have kernels of size  $5 \times 5$ , and stride and padding equal 2 (for both depthwise convolutions). The second morphological layer has 16 neurons: 3 of the first type of composed processing units  $M^{C^\varepsilon}$ , 3 of the second type of composed neurons  $M^{C^\delta}$ , 3 reconstruction by erosion  $M^{\tilde{\rho}^\varepsilon}$  and 2 by dilation  $M^{\tilde{\rho}^\delta}$ , 2 white  $M^{T^w}$  and 3 black top hats  $M^{T^b}$ . This layer has kernel filters of size  $5 \times 5$ , with stride 1, and padding 2 for both depthwise convolutions. After the morphological layers, three fully connected ones are responsible to learn high-level features and perform the final classification. The first layer has 120 neurons while the second has 84 processing units. Both layers use ReLUs [Nair and Hinton, 2010] as the activation function. Finally, the last fully connected has the number of neurons equal the number of class of training dataset and uses softmax as the activation function.

To analyze the effectiveness of the technique in a more complex scenario, we proposed a larger network based on the famous AlexNet [Krizhevsky et al., 2012] architecture. However, in order to have more control of the trainable parameters, the

proposed morphological version of the AlexNet architecture [Krizhevsky et al., 2012], called **DeepMorphAlexNet** and presented in Figure 7.5c, has the same number of layers but less neurons in each layer. The morphological first layer has 8 processing units: 4 related to the first type of composed neurons  $M^{C\varepsilon}$  and 4 related to the second type of composed neurons  $M^{C\delta}$ . Furthermore, in this layer, the kernels have size of  $11 \times 11$  (for both depthwise convolutions) and stride of 1 and 5, and padding of 5 and 2, for the first and second depthwise convolutions, respectively. The second layer has 24 neurons, with 4 for each of the following operations: first ( $M^{C\varepsilon}$ ) and second ( $M^{C\delta}$ ) types of composed processing units, reconstruction by erosion  $M^{\tilde{\rho}^\varepsilon}$  and by dilation  $M^{\tilde{\rho}^\delta}$ , white  $M^{\mathcal{T}^w}$  and black  $M^{\mathcal{T}^b}$  top hats. In this layer, the kernels have size of  $5 \times 5$ , with stride 1, and padding 2 for both depthwise convolutions. The third morphological layer has 48 neurons equally divided into the two types of composed processing units ( $M^{C\varepsilon}$  and  $M^{C\delta}$ ). This layer has the kernels of size  $3 \times 3$  (for both convolutions) and stride 1 and 3, and padding 1 and 0, for the first and second depthwise convolutions, respectively. The fourth layer has 32 neurons: 6 of the first ( $M^{C\varepsilon}$ ) and second ( $M^{C\delta}$ ) types of composed processing units, 5 reconstruction by erosion  $M^{\tilde{\rho}^\varepsilon}$  and 5 by dilation  $M^{\tilde{\rho}^\delta}$ , 5 white  $M^{\mathcal{T}^w}$  and 5 black  $M^{\mathcal{T}^b}$  top hats. Moreover, this layer has, for both convolutions, the following parameter configuration: kernel filters of size  $3 \times 3$ , and stride and padding equal 1. The fifth (and last) morphological layer has 32 neurons also equally divided into the two types of composed processing units ( $M^{C\varepsilon}$  and  $M^{C\delta}$ ). This layer has kernel of size  $3 \times 3$  (for both convolutions), and stride of 1 and 2, and padding of 1 and 0, for the first and second depthwise convolutions, respectively. In the end, three fully connected layers are responsible to learn high-level features and perform the final classification. Similarly to the DeepMorphLeNet, the first two layers have 512 neurons (with ReLUs [Nair and Hinton, 2010] as the activation function) while the last one has the number of neurons equal the number of classes of the training dataset (with softmax as activation function).

## 7.2 Experimental Setup

In this section, we present the experimental setup. Section 7.2.1 presents specific datasets employed to validate the proposed technique. Baselines are described in Section 7.2.2 while the experimental protocol is introduced in Section 7.2.3.



**Figure 7.5:** The three Deep Morphological Network (DeepMorphNet) architectures proposed and exploited in this work. Note that the number of morphological neurons of each type in each layer is codified using the Equations of Section 7.1.2. Also, observe that the two depthwise convolutions of a same layer share the kernel size, sometimes differing only in the stride and padding. Moreover, the padding and stride of each layer are presented as follows: the value related to the first depthwise convolution is reported separated by a comma of the value related to the second depthwise convolution. Though not visually represented, the pointwise convolutions explored in the morphological layers always use the same configuration: kernel  $1 \times 1$ , stride 1, and no padding.

## 7.2.1 Specific Datasets

Four datasets were employed to validate the proposed DeepMorphNets. Two of them (UCMerced Land-use and WHU-RS19 datasets) have already been presented in Section 4.1.1. The other two are synthetic datasets that were exclusively designed to check the feature learning of the proposed technique.

### 7.2.1.1 Synthetic Datasets

As introduced, two simple image classification datasets were designed in this work to validate the feature learning process of the proposed DeepMorphNets. In order to allow such validation, these datasets were created so that it is possible to define, a priori, the optimal structuring element (i.e., the SE that would produce the best results) for a classification scenario. Hence, in this case, the validation would be performed by comparing the learned structuring element with the optimal one, i.e., if both SEs are similar, then the proposed technique is able to perform well the feature learning step.

Specifically, both datasets are composed of 1,000 grayscale images with a resolution of  $224 \times 224$  pixels (a common image size employed in famous architecture such as AlexNet [Krizhevsky et al., 2012]) equally divided into two classes.

The **first dataset** has two classes of squares: (i) a small one, with  $5 \times 5$  pixels, and (ii) a large one, with  $9 \times 9$  pixels. In this case, a simple opening with a square structuring element larger than  $5 \times 5$  but smaller than  $9 \times 9$  should erode the small

squares while keeping the larger ones, allowing the network to perfectly classify the dataset.

More difficult, the **second synthetic dataset** has two classes of rectangles. The first class has shapes of  $7 \times 3$  pixels while the other one has rectangles of  $3 \times 7$ . This case is a little more complicated because the network should learn a structuring element based on the orientation of one the rectangles. Particularly, it is possible to perfectly classify this dataset using a single opening operation with one of the following types of SEs: (i) a rectangle of at least 7 pixels of width and height larger than 3 but smaller than 7 pixels, which would erode the first class of rectangles and preserve the second one, or (ii) a rectangle with a width larger than 3 but smaller than 7 pixels and height larger than 7 pixels, which would erode the second class of rectangle while keeping the first one.

## 7.2.2 Baselines

For all datasets, two baselines were used. The first baseline is a standard convolutional version of the proposed [DeepMorphNet](#). Precisely, this baseline recreates the exact morphological architecture using the traditional convolutional layer (instead of depthwise and pointwise convolutions) but preserving all remaining configurations (such as filter sizes, padding, stride, etc). Furthermore, differently from the morphological networks, this baseline makes use of max-pooling layers between the convolutions, which makes it very similar to the traditional architectures of the literature [[LeCun et al., 1998](#); [Krizhevsky et al., 2012](#)]. The second baseline, referenced hereafter with the prefix “Depth”, is exactly the [DeepMorphNet](#) architecture but without using binary weights and depthwise pooling. This baseline reproduces the same [DeepMorphNet](#) architecture using only depthwise and pointwise convolutions (i.e., depthwise separable convolution [[Chollet, 2017](#)]) without binary weights.

Differently from the morphological networks, both baselines use Rectified Linear Units (ReLUs) [[Nair and Hinton, 2010](#)] as activation functions and batch normalization [[Ioffe and Szegedy, 2015](#)] (after each convolution). It is important to highlight that, despite the differences, both baselines have the exact same number of layers and feature maps of the base [DeepMorphNet](#). We believe that this conservation allows a fair comparison between the models given that the potential representation of the networks is somehow the same.

### 7.2.3 Experimental Protocol

For synthetic datasets, the train/test protocol, presented in Section 4.2, was employed. Particularly, in this case, the whole dataset is randomly divided into three sets: training (composed of 60% of the instances), validation and test (each one composed of 20% of the dataset samples). Once determined, these sets are used throughout the experiments for all networks and baselines. Results of this protocol are reported in terms of the average accuracy (presented in Section 4.3) of the test set.

For the other datasets, a 5-fold cross-validation was conducted to assess the accuracy of the proposed algorithm. In this case, the final results are the mean of the average accuracy (for the test set) of the five runs followed by its corresponding standard deviation.

All networks proposed in this work were implemented using Torch<sup>1</sup>, a scientific computing framework with wide support for machine learning algorithms. This framework is more suitable due to its support to parallel programming using CUDA, an NVidia parallel programming based on Graphics Processing Units. All experiments were performed on a 64 bit Intel i7 5930K machine with 3.5GHz of clock, 64GB of RAM memory and a GeForce GTX Titan X Pascal with 12GB of memory under a 9.0 CUDA version. Ubuntu version 18.04.1 LTS was used as operating system.

## 7.3 Results and Discussion

In this section, we present and discuss the obtained outcomes. Section 7.3.1 presents the results of the synthetic datasets while Section 7.3.2 discusses about the experiments over the image classification datasets.

### 7.3.1 Synthetic Datasets

As explained in Section 7.2.1.1, two synthetic datasets were proposed in this work to validate the feature learning of the deep morphological networks. Furthermore, as introduced, both datasets can be perfectly classified using one opening with specific structuring elements. Supported by this, the basic DeepMorphSynNet (Figure 7.5a), composed of one opening neuron, can be used to validate the feature learning process of the proposed technique, given that this network has the capacity of perfectly classifying the datasets as long as it successfully learns the SE.

---

<sup>1</sup><http://torch.ch/> (as of May 2019).



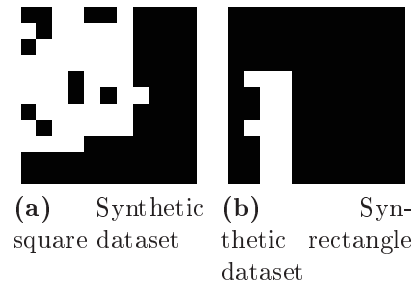
**Table 7.1:** Results, in terms of average accuracy, of the proposed method and the baselines for the square synthetic dataset (Section 7.2.1.1).

Method	Average Accuracy (%)
Classification Layer	86.50
ConvNet	58.00
Depth-ConvNet	60.00
DeepMorphSynNet	<b>100.00</b>

Given the simplicity of these datasets, aside from the methods describe in Section 6.2.1, we also employed as baseline a basic architecture composed uniquely of a classification layer. Specifically, this network has one layer that receives a linearized version of the input data and outputs the classification. The proposed morphological network, as well as the baselines, were tested for both synthetic datasets using the same configuration, i.e., learning rate, weight decay, momentum, and number of epochs of 0.01, 0.0005, 0.9, and 10, respectively.

Results for the synthetic square dataset are presented in Table 7.1. Among the baselines, the worst results were generated by the ConvNets while the best outcome was produced by the network composed of a single classification layer (86.50%). A reason for this is that the proposed dataset does not have much visual information to be extracted by the convolution layers. Hence, in this case, the raw pixels themselves are able to provide relevant information (such as the total amount of pixels of the square) for the classification. However, the result yielded by the best baseline (the network composed of the classification layer) was worse than the result generated by the proposed morphological network. Precisely, the DeepMorphSynNet yielded a 100% of average accuracy, perfectly classifying the whole test set of this synthetic dataset. As introduced in Section 7.2.1.1, in order to achieve this perfect classification, the opening would require a square structuring element larger than  $5 \times 5$  but smaller than  $9 \times 9$  pixels. As presented in Figure 7.6a, this was exactly the structuring element learned by the network. Moreover, as introduced, with this SE, the opening would erode the small  $5 \times 5$  squares while keeping the larger  $9 \times 9$  ones. This was the exact outcome of the morphological network, as presented in Figure 7.7.

Results for the synthetic rectangle dataset are presented in Table 7.2. Differently from the synthetic square dataset, the network composed of a single classification layer produced the worst outcome while the convolutional architectures yielded perfect results (100%). This difference may be justified by the fact that this is a more complex dataset that has two classes with equal shapes but differing in other relevant properties (such as the orientation) that may be extracted by the convolution layers. Also



**Figure 7.6:** Learned structuring elements for the synthetic datasets. In both cases, the learned SEs are exactly as expected, i.e., a square SE larger than  $5 \times 5$  but smaller than  $9 \times 9$  for the square synthetic dataset, and a rectangle larger than the rectangles of one class and in the same orientation for the rectangle synthetic dataset.



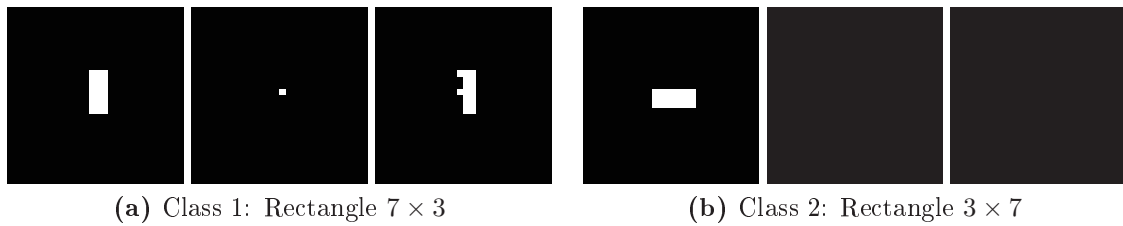
**Figure 7.7:** Examples of the output of the opening neuron for the square synthetic dataset. The first column represents the input image, the second one is the output of the erosion, and the last one is the output of the dilation. Since erosion and dilation have tied weights (i.e., the same structuring element), they implement an opening.

**Table 7.2:** Results, in terms of average accuracy, of the proposed method and the baselines for the synthetic rectangle dataset (Section 7.2.1.1).

Method	Average Accuracy (%)
Classification Layer	55.00
<b>ConvNet</b>	<b>100.00</b>
<b>Depth-ConvNet</b>	<b>100.00</b>
<b>DeepMorphSynNet</b>	<b>100.00</b>

different from previous outcomes, in this case, the proposed DeepMorphSynNet and best baselines produced the same results (100% of average accuracy), perfectly classifying this synthetic dataset. As introduced in Section 7.2.1.1, to perform this perfect classification, the opening operation (of the DeepMorphSynNet) would require a specific SE that should have the same orientation of one of the rectangles. As presented in Figure 7.6b, this is the SE learned by the morphological network. With such filter, the opening operation would erode the one type of rectangles while keeping the other, the exact outcome presented in Figure 7.8.

Results obtained with the synthetic datasets show that the proposed morphological networks are able to optimize and learn interesting structuring elements. Furthermore, in some scenarios, such as those simulated by the synthetic datasets,



**Figure 7.8:** Examples of the output of the opening neuron for the synthetic rectangle dataset. The first column represents the input image, the second one is the output of the erosion, and the last one is the output of the dilation. Since erosion and dilation have tied weights (i.e., the same structuring element), they implement an opening.

the *DeepMorphNets* have achieved promising results. A better analysis of the potential of the proposed technique is performed in the next section using real (not synthetic) datasets.

### 7.3.2 Image Classification Datasets

For the image classification datasets, aside from the methods describe in Section 6.2.1, we also employed as baseline an approach, called hereafter as Static *SEs*, that reproduces the exactly morphological architectures but with static (non-optimized) structuring elements. In this case, each neuron has the configuration based on the 5 most common structuring elements (presented in Figure 3.5). The features extracted by these static neurons are classified by a Support Vector Machine (SVM). The idea behind this baseline is to have a lower bound for the morphological network, given that this proposed approach should be able to learn better structuring elements and, consequently, produce superior results.

Aside from this, for both datasets, all networks were tested using essentially the same configuration, i.e., bath size, learning rate, weight decay, and momentum of 16, 0.01, 0.0005, and 0.9, respectively. The only difference in the experiments is related to the number of epochs. Precisely, the LeNet-based architectures were trained using 500 epochs while for the AlexNet-based networks, 2,000 epochs were employed.

#### 7.3.2.1 UCMerced Land-use Dataset

Results for the UCMerced Land-use dataset are reported in Table 7.3. In this case, all networks outperformed their respective lower bounds (generated by the Static *SEs*), an expected outcome given the feature learning step performed by the deep learning-based approaches. Considering the LeNet-based networks, the best result, among the baselines, was produced by the architecture based on depthwise separable convolutions [Chollet, 2017], i.e., the Depth-LeNet. The proposed *DeepMorphLeNet* produced

**Table 7.3:** Results, in terms of accuracy, of the proposed method and the baselines for the UCMerced Land-use Dataset.

Method	Average Accuracy (%)	Number of Parameters (millions)	Training Time (hours per fold)
Static SEs	19.46±2.06	-	-
LeNet [LeCun et al., 1998]	53.29±0.86	4.42	1.2
Depth-LeNet	54.81±1.25	5.04	6.2
DeepMorphLeNet (ours)	56.52±1.74	6.04	7.8
Static SEs	28.21±2.64	-	-
AlexNet-based [Krizhevsky et al., 2012]	72.62±1.05	6.50	8.5
Depth-AlexNet-based	73.14±1.43	7.47	101.1
DeepMorphAlexNet (ours)	76.86±1.97	10.50	209.9

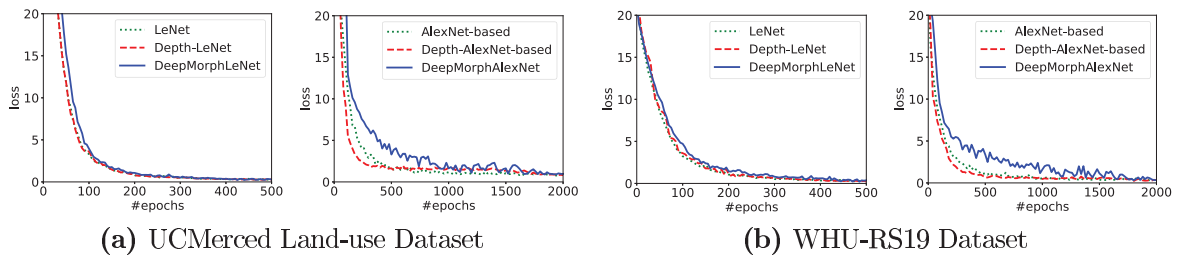
similar results when compared to this baseline, which shows the potential of the proposed technique that optimizes the morphological filters to extract salient and relevant features.

The exact same conclusions can be drawn from the AlexNet-based networks. Specifically, the best baseline was the Depth-AlexNet-based, that produces 73.14±1.43% of average accuracy. However, again, the proposed DeepMorphAlexNet yielded competitive results when compared to this baseline (76.86±1.97% of average accuracy). These results show that morphological operations are able to learn useful features. Some of these feature maps of the AlexNet-based architectures are presented in Figure 7.10. Note the difference between the characteristics learned by the distinct networks. In general, the morphological network is able to capture different features when compared to the ConvNets.

In order to better evaluate the proposed morphological network, a convergence analysis for the UCMerced Land-use dataset is presented in Figure 7.9a. Note that the DeepMorphNets are slower to converge compared to the other networks. A reason for that is the number of trainable parameters. As presented in Table 7.3, the DeepMorphNets have more parameters and, therefore, are more complex to train. However, given enough training time, all networks converge very similarly, which confirms that the proposed DeepMorphNets are able to effectively learn interesting SEs and converge to a suitable solution.

### 7.3.2.2 WHU-RS19 Dataset

Table 7.4 presents the results related to the WHU-RS19 dataset. Again, as expected, all architectures outperformed their respective lower bounds (generated by the Static SEs). Considering the LeNet-based networks, the best result, among the baselines, was pro-



**Figure 7.9:** Convergence of the proposed morphological networks and the baselines for both datasets. Note that only fold 1 is reported.

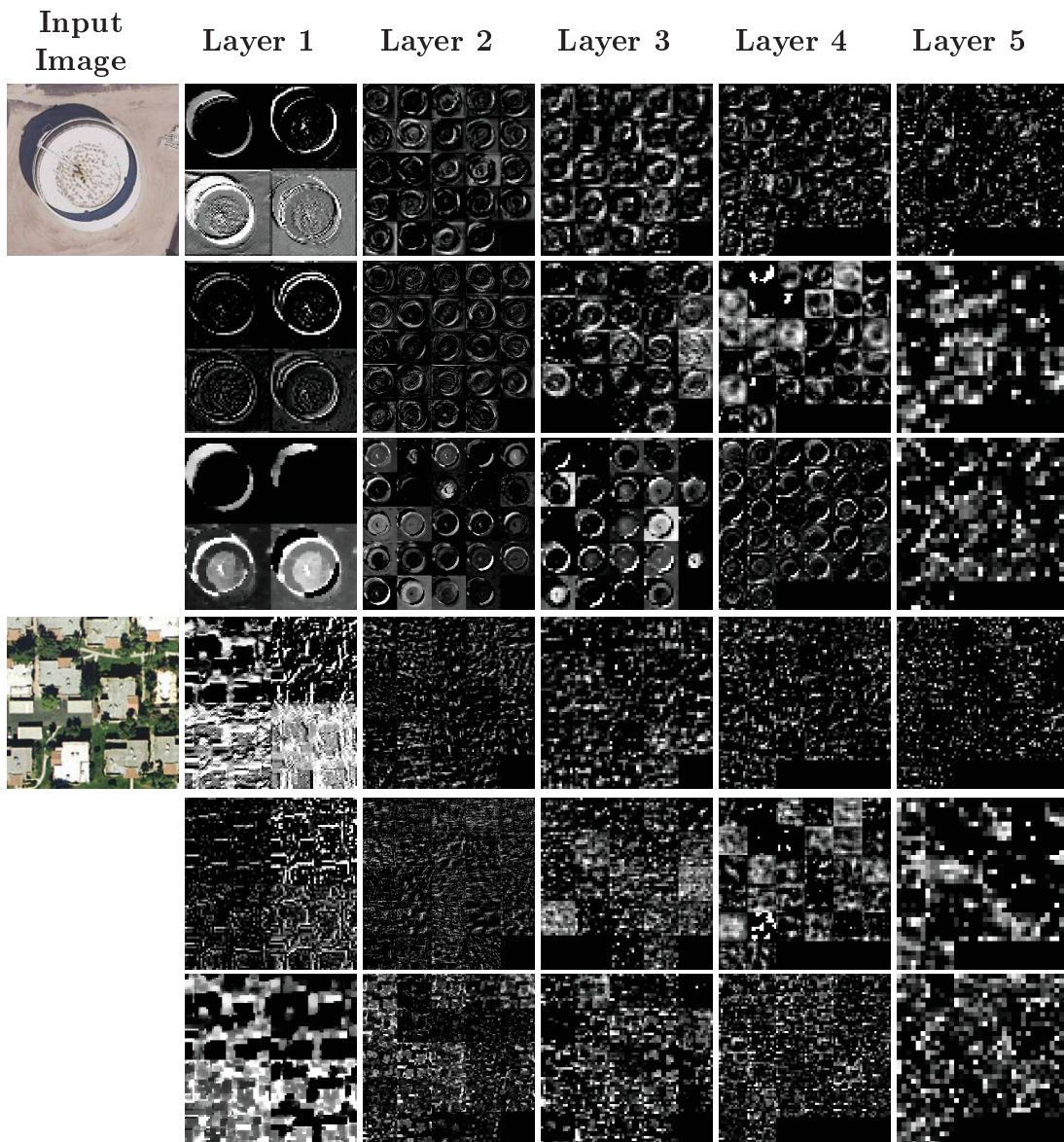
duced by the LeNet architecture [LeCun et al., 1998]. The proposed DeepMorphLeNet generated competitive results when compared to this baseline, corroborating with previous conclusions about the ability of the proposed technique to optimize the morphological filters and extract important features.

As for the UCMerced Land-use dataset, the exact same conclusions can be drawn from the AlexNet-based networks. In this case, the best baseline was the AlexNet-based network, that produces  $64.38 \pm 2.93\%$  of average accuracy. However, the proposed DeepMorphAlexNet produced similar results when compared to this baseline ( $68.20 \pm 2.75\%$  of average accuracy). These results reaffirm the previous conclusions related to the ability of the morphological networks to capture interesting information. Figure 7.11 presents some feature maps learned by the AlexNet-based networks for the WHU-RS19 Dataset. Again, it is remarkable the difference between the features extracted by the different architectures. Overall, the DeepMorphNet is capable of learning specific and distinct features when compared to the ConvNets.

As for the previous dataset, a convergence analysis for the WHU-RS19 dataset is presented in Figure 7.9b. Again, although the DeepMorphNets are slower to converge (mainly due to the number trainable parameters, as presented in Table 7.4), they are able to achieve similar results if enough time is provided for training the model, corroborating with previous conclusions.

## 7.4 Conclusion

In this chapter, we proposed a novel method for deep feature learning, called Deep Morphological Network (DeepMorphNet), that is able to do morphological operations while optimizing their structuring elements toward a better solution. This proposed DeepMorphNet is composed of morphological layers, which are strongly based on a framework that is basically consisted of depthwise convolutions (that process the input with decomposed binary weights) and pooling layers. Such framework provides



**Figure 7.10:** Input images and some produced (upsampled) feature maps extracted from all layers of the AlexNet-based networks for the UCMerced Land-use Dataset. For each input image: the first row presents features from the AlexNet-based network, the second row presents the feature maps learned by the Depth-AlexNet-based architecture, and the last row presents the features of the proposed morphological network.

support for the creation of the basic morphological layers that perform erosion and dilation. This basic layer, in turn, allows the creation of other more complex layers, which may perform opening, dilation, (black and white) top-hat, and (an approximation of) geodesic reconstruction (by erosion or dilation). The proposed approach is trained end-to-end using standard algorithms employed in deep learning-based networks, including backpropagation and Stochastic Gradient Descent (SGD) [Goodfellow et al., 2016].

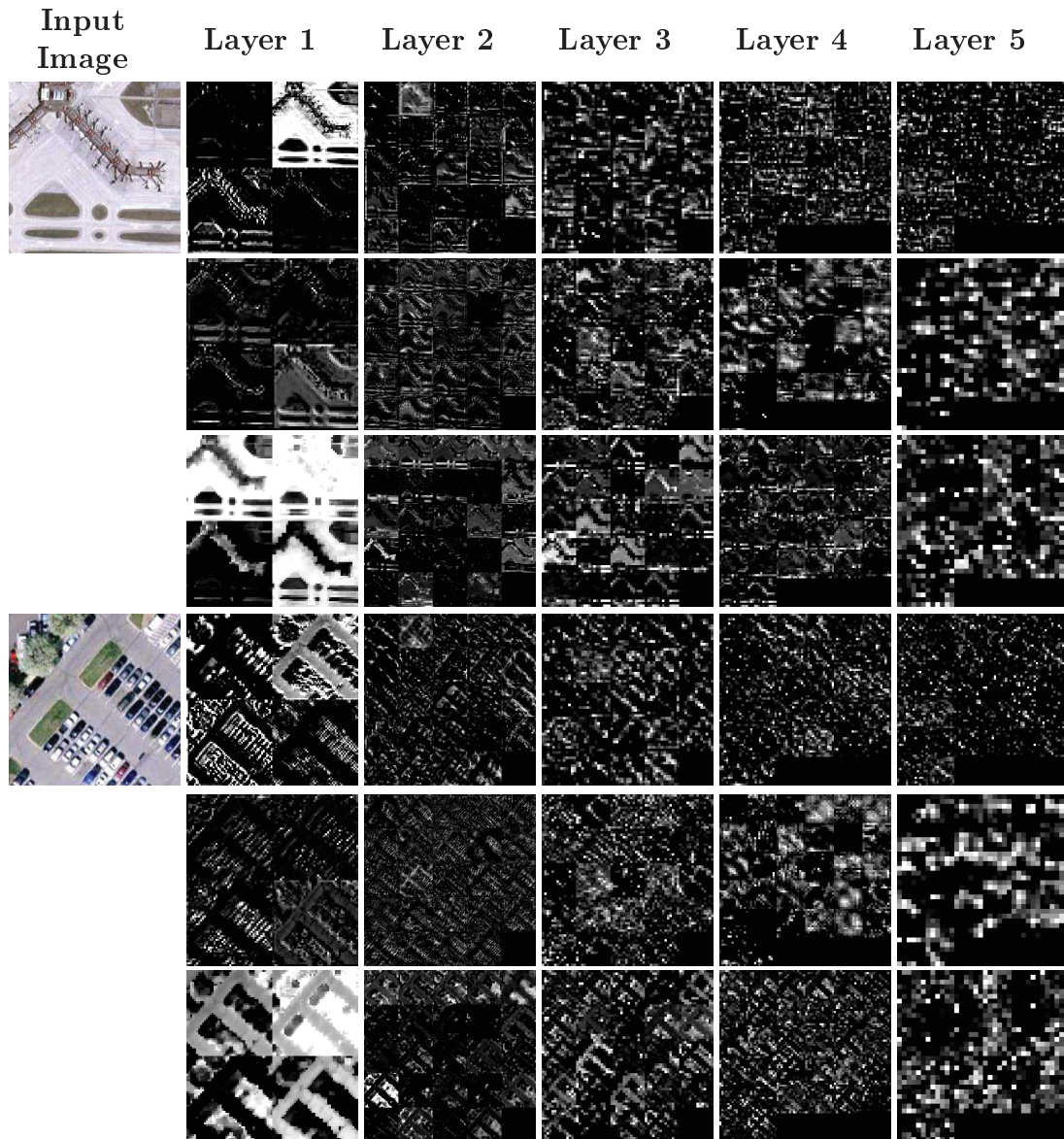
Experiments were first conducted on two synthetic datasets in order to analyze

**Table 7.4:** Results, in terms of average accuracy, of the proposed **DeepMorphNets** and baselines for the WHU-RS19 dataset.

Method	Average Accuracy (%)	Number of Parameters (millions)	Training Time (hours per fold)
<b>Static SEs</b>	15.17±2.83	-	-
<b>LeNet</b> [LeCun et al., 1998]	48.26±2.01	4.42	0.6
<b>Depth-LeNet</b>	47.19±2.43	5.04	3.1
<b>DeepMorphLeNet (ours)</b>	52.91±2.60	6.04	3.7
<b>Static SEs</b>	25.33±2.95	-	-
<b>AlexNet-based</b> [Krizhevsky et al., 2012]	64.38±2.93	6.50	4.7
<b>Depth-AlexNet-based</b>	63.27±2.14	7.47	44.7
<b>DeepMorphAlexNet (ours)</b>	68.20±2.75	10.50	99.8

the feature learning of the proposed technique as well as its efficiency. Results over these datasets have shown that the proposed **DeepMorphNets** are able to learn relevant structuring elements. In fact, the method could learn the expect (a priori) structuring element. Furthermore, the proposed approach learned a perfect classification of both datasets outperforming or producing equal results when compared to the ConvNets. This result shows the potential of **DeepMorphNets**, which are able to learn important filters that are also different from those learned by the ConvNets.

After this first analysis, the **DeepMorphNet** was analyzed using two remote sensing image classification datasets. In both scenarios, the **DeepMorphNets** produced competitive results when compared to the convolutional networks. This outcome corroborates with the previous conclusion, that the morphological networks are capable of learning relevant filters.



**Figure 7.11:** Input images and some produced (upsampled) feature maps extracted from all layers of the AlexNet-based networks for the WHU-RS19 Dataset. For each input image: the first row presents features from the AlexNet-based network, the second row presents the feature maps learned by the Depth-AlexNet-based architecture, and the last row presents the features of the proposed morphological network.



## Chapter 8

# Conclusions and Future Work

This thesis addresses remote sensing scene and pixel classification challenges. Many of them are directly related to peculiarities of remote sensing images, including data availability, context exploitation, and so on. Considering this, we have presented contributions in several main research topics that concerns those remote sensing scene and pixel classification challenges.

In Chapter 5, we evaluated three strategies for exploiting existing *ConvNets* in different scenarios from the ones they were trained. The objective was to understand the best way to obtain the most benefits from these state-of-the-art deep learning approaches in problems that usually are not suitable for the design and creation of new *ConvNets* from scratch. Such scenarios reflect many existing applications, in which there is few labeled data. We performed experiments evaluating the three strategies for exploiting the *ConvNets* (full training, fine tuning, and using as feature extractors) over six popular *ConvNets* in three remote sensing datasets. Experimental results showed that fine tuning tends to be the best strategy in different situations. Furthermore, results pointed out that deep features of the existing pre-trained *ConvNets* generalize to other domains, mainly when there are similarities between them (case of UCMerced and RS19 datasets).

In Chapter 6, we focus on pixel classification of remote sensing images. Since the pixel may not have enough information to allow its classification, we proposed a new technique that exploits the context to improve the classification results. Given that the context size is of vital importance for the model, the proposed approach, based on dilated *ConvNets*, alleviates this problem by exploiting the multi-context paradigm (without increasing the number of parameters). Furthermore, this technique can define adaptively the best patch size for the inference stage. Experimental results pointed out that the proposed method can effectively exploit the benefits of the pixel

context by aggregating multi-context information, achieving state-of-the-art results in two datasets and competitive outcome in other two remote sensing datasets.

In Chapter 7, we proposed a novel method for deep feature learning, called Deep Morphological Network (*DeepMorphNet*), that is able to do morphological operations while optimizing their structuring elements toward a better solution. The intuition behind this method comes from the fact that morphological operations are able to cope better with some visual characteristics when compared to the filters employed in *ConvNets*. Experimental results showed that the proposed method are really able to extract distinct image properties when compared to *ConvNets*. Moreover, the proposed approach yielded competitive results in two remote sensing image classification datasets.

## 8.1 Future Work

The work presented in this thesis open new opportunities towards a better spectral-spatial feature representation, which is still needed for remote sensing applications, such as agriculture or environmental monitoring.

In general, some possible research venues are:

- *Analyze the effect of each type of morphological neuron.* This topic involves understanding the benefits of each type of morphological neuron and analyzing which ones are the best for each scenario. This is an interesting research topic given that each type of neuron produces distinct outcomes. Therefore, this analysis would allow the definition of which neurons are most suitable for each application.
- *Combine *ConvNets* and *DeepMorphNets*.* This is a captivating research topic given that it focuses on extracting and combining the benefits of convolutional and morphological networks. As introduced in this thesis, these techniques are able to capture distinct features. Hence, a combination of these approaches should be able to create a better representation, mainly because of the diversity of the extracted features.
- *Adapt the deep morphological networks to perform pixel classification.* This is a direct application of the proposed *DeepMorphNets*. In this thesis, such network has been evaluated only for remote sensing scene classification. Therefore, it should be natural to apply the proposed networks for remote sensing pixel classification and analyze the benefits brought by this method.

- *Implement modern layers and architectures based on the deep morphological networks.* There are several modern layers (such as the ones with dilated filters) and architectures (including ResNets [He et al., 2016] and DenseNets [Huang et al., 2017]). It is an interesting research topic to analyze if it is feasible and logical to adapt the deep morphological networks to recreate these techniques.



# Appendix A

## List of Publications

This thesis has generated publications directly and indirectly related to its context.

List of journal papers:

1. *Dynamic Multi-Context Segmentation of Remote Sensing Images based on Convolutional Networks* [Nogueira et al., 2019a]. K. Nogueira, M. Dalla Mura, J. Chanussot, W. R. Schwartz, J. A. dos Santos. IEEE Transactions on Geoscience & Remote Sensing. (to appear).
2. *Spatio-Temporal Vegetation Pixel Classification By Using Convolutional Networks* [Nogueira et al., 2019b]. K. Nogueira, J. A. dos Santos, N. Menini, T. S. F. Silva, L. P. C. Morellato, R. S. Torres. IEEE Geoscience and Remote Sensing Letters (to appear).
3. *Exploiting ConvNet Diversity for Flooding Identification* [Nogueira et al., 2018]. K. Nogueira, S. G. Fadel, . C. Dourado, R. O. Werneck, J. A. V. Muoz, O. A. B. Penatti, R. T. Calumby, L. T. Li, J. A. dos Santos, R. S. Torres. IEEE Geoscience and Remote Sensing Letters, 2018.
4. *Towards Better Exploiting Convolutional Neural Networks for Remote Sensing Scene Classification* [Nogueira et al., 2017c]. K. Nogueira, O. A. B. Penatti, J. A. dos Santos. Pattern Recognition, 2017.

List of conference papers:

1. *Deep contextual description of superpixels for aerial urban scenes classification* [Santana et al., 2017]. T. M. Santana, K. Nogueira, A. M. Machado, J. A. dos Santos. IEEE International Geoscience and Remote Sensing Symposium (IGARSS), 2017.

2. *Semantic segmentation of vegetation images acquired by unmanned aerial vehicles using an ensemble of ConvNets* [Nogueira et al., 2017a]. K. Nogueira, J. A. dos Santos, L. Cancian, B. D. Borges, T. S. Silva, L. P. Morellato, R. S. Torres. IEEE International Geoscience and Remote Sensing Symposium (IGARSS), 2017.
3. *Learning Deep Features on Multiple Scales for Coffee Crop Recognition* [Baeta et al., 2017]. R. Baeta, K. Nogueira, D. Menotti, J. A. dos Santos. Conference on Graphics, Patterns and Images (SIBGRAPI), 2017.
4. *Data-Driven Flood Detection using Neural Networks* [Nogueira et al., 2017b]. K. Nogueira, S. G. Fadel, Í. C. Dourado, R. Werneck, J. A. V. Muoz, O. A. B. Penatti, R. T. Calumby, L. T. Li, J. A. dos Santos, R. S. Torres. MediaEval Multimedia Benchmark Workshop, 2017.
5. *Towards Vegetation Species Discrimination by using Data-driven Descriptors* [Nogueira et al., 2016b]. K. Nogueira, J. A. dos Santos, T. Fornazari, T. S. F. Silva, L. P. Morellato, R. S. Torres. Pattern Recognition in Remote Sensing (ICPR Workshop), 2016.
6. *Learning to Semantically Segment High-Resolution Remote Sensing Images* [Nogueira et al., 2016a]. K. Nogueira, M. Dalla Mura, J. Chanussot, W. R. Schwartz, J. A. dos Santos. International Conference on Pattern Recognition (ICPR), 2016.
7. *A Ranking Fusion Approach for Geographic-Location Prediction of Multimedia Objects* [Muñoz et al., 2016]. J. A. V. Muoz, L. T. Li, Í. C. Dourado, K. Nogueira, S. G. Fadel, O. A. B. Penatti, J. Almeida, L. A. M. Pereira, R. T. Calumby, J. A. dos Santos, R. S. Torres. MediaEval Multimedia Benchmark Workshop, 2016.
8. *Improving Spatial Feature Representation from Aerial Scenes by Using Convolutional Networks* [Nogueira et al., 2015a]. K. Nogueira, W. O. Miranda, J. A. dos Santos. Conference on Graphics, Patterns and Images (SIBGRAPI), 2015.
9. *Coffee Crop Recognition Using Multi-scale Convolutional Neural Networks* [Nogueira et al., 2015b]. K. Nogueira, W. R. Schwartz, J. A. dos Santos. Iberoamerican Congress on Pattern Recognition (CIARP), 2015.
10. *Do Deep Features Generalize from Everyday Objects to Remote Sensing and Aerial Scenes Domains?* [Penatti et al., 2015] O. A. B. Penatti, K. Nogueira, J. A. dos Santos. EarthVision (IEEE Conference on Computer Vision and Pattern Recognition Workshop – CVPRW), 2015.

11. *RECOD @ Placing Task of MediaEval 2015* [Li et al., 2015]. L. T. Li, J. A. V. Muoz, J. Almeida, R. T. Calumby, O. A. B. Penatti, I. C. Dourado, K. Nogueira, P. R. Mendes-Junior, L. A. M. Pereira, D. C. G. Pedronette, J. A. dos Santos, M. A. Goncalves, R. da S. Torres. MediaEval Multimedia Benchmark Workshop, 2015.





# Bibliography

- Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.
- Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., and Süsstrunk, S. (2012). Slic superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282.
- Almeida, J., dos Santos, J. A., Alberton, B., Torres, R. d. S., and Morellato, L. P. C. (2014). Applying machine learning based on multiscale classifiers to detect remote phenology patterns in cerrado savanna trees. *Ecological Informatics*, 23:49–61.
- Aptoula, E., Ozdemir, M. C., and Yanikoglu, B. (2016). Deep learning with attribute profiles for hyperspectral image classification. *IEEE Geoscience and Remote Sensing Letters*, 13(12):1970–1974.
- Audebert, N., Le Saux, B., and Lefèvre, S. (2016). Semantic segmentation of earth observation data using multimodal and multi-scale deep networks. In *Asian Conference on Computer Vision*, pages 180–196. Springer.
- Avila, S., Thome, N., Cord, M., Valle, E., and de A. Araújo, A. (2013). Pooling in image representation: the visual codeword point of view. *Computer Vision and Image Understanding*, 117(5):453–465.

- Avramović, A. and Risojević, V. (2014). Block-based semantic classification of high-resolution multispectral aerial images. *Signal, Image and Video Processing*, pages 1--10.
- Babenko, A., Slesarev, A., Chigorin, A., and Lempitsky, V. (2014). Neural codes for image retrieval. In *Computer Vision—ECCV 2014*, pages 584--599. Springer.
- Badrinarayanan, V., Kendall, A., and Cipolla, R. (2017). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *Transactions on Pattern Analysis and Machine Intelligence*, 39(12):2481--2495.
- Baeta, R., Nogueira, K., Menotti, D., and dos Santos, J. A. (2017). Learning deep features on multiple scales for coffee crop recognition. In *Conference on Graphics, Patterns and Images (SIBGRAPI)*, pages 262--268.
- Ball, J. E., Anderson, D. T., and Chan, C. S. (2017). Comprehensive survey of deep learning in remote sensing: theories, tools, and challenges for the community. *Journal of Applied Remote Sensing*, 11(4):042609.
- Benediktsson, J., Chanussot, J., and Moon, W. (2013). Advances in very-high-resolution remote sensing [scanning the issue]. *Proceedings of the IEEE*, 101(3):566--569.
- Benediktsson, J. A., Palmason, J. A., and Sveinsson, J. R. (2005). Classification of hyperspectral data from urban areas based on extended morphological profiles. *IEEE Transactions on Geoscience and Remote Sensing*, 43(3):480--491.
- Bengio, Y. (2009). Learning deep architectures for ai. *Foundations and trends in Machine Learning*, 2(1):1--127.
- Bengio, Y. (2012). Practical recommendations for gradient-based training of deep architectures. In *Neural Networks: Tricks of the Trade*, pages 437--478. Springer.
- Bischke, B., Helber, P., Schulze, C., Venkat, S., Dengel, A., and Borth, D. (2017). The multimedia satellite task at mediaeval 2017: Emergence response for flooding events. In *Proceedings of the MediaEval 2017 Workshop*, Ireland.
- Bishop, C. M. (2006). *Pattern recognition and machine learning*. springer.
- Bouchiha, R. and Besbes, K. (2013). Comparison of local descriptors for automatic remote sensing image registration. *Signal, Image and Video Processing*, 9(2):463--469.

- Boureau, Y.-L., Bach, F., LeCun, Y., and Ponce, J. (2010). Learning mid-level features for recognition. In *Conference on Computer Vision and Pattern Recognition*, pages 2559--2566.
- Bullen, P. S. (2013). *Handbook of means and their inequalities*, volume 560. Springer Science & Business Media.
- Campbell, J. B. and Wynne, R. H. (2011). *Introduction to remote sensing*. Guilford Press.
- Çarkacıoğlu, A. and Yarman-Vural, F. (2003). Sasi: a generic texture descriptor for image retrieval. *Pattern Recognition*, 36(11):2615 – 2633.
- Chatfield, K., Simonyan, K., Vedaldi, A., and Zisserman, A. (2014). Return of the devil in the details: Delving deep into convolutional nets. *arXiv preprint arXiv:1405.3531*.
- Chen, C., Zhou, L., Guo, J., Li, W., Su, H., and Guo, F. (2015). Gabor-filtering-based completed local binary patterns for land-use scene classification. In *IEEE International Conference on Multimedia Big Data (BigMM)*, pages 324--329. IEEE.
- Chen, C.-h., Pau, L.-F., and Wang, P. S.-p. (2010). *Handbook of pattern recognition and computer vision*, volume 27. World Scientific.
- Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. (2018). Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *Transactions on Pattern Analysis and Machine Intelligence*, 40(4):834–848.
- Chen, Y., Jiang, H., Li, C., Jia, X., and Ghamisi, P. (2016). Deep feature extraction and classification of hyperspectral images based on convolutional neural networks. *IEEE Transactions on Geoscience and Remote Sensing*, 54(10):6232--6251.
- Chen, Y., Lin, Z., Zhao, X., Wang, G., and Gu, Y. (2014). Deep learning-based classification of hyperspectral data. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 7(6):2094--2107.
- Cheng, G., Li, Z., Yao, X., Guo, L., and Wei, Z. (2017). Remote sensing image scene classification using bag of convolutional features. *IEEE Geoscience and Remote Sensing Letters*, 14(10):1735--1739.
- Cheng, G., Ma, C., Zhou, P., Yao, X., and Han, J. (2016). Scene classification of high resolution remote sensing images using convolutional neural networks. In *IEEE International Geoscience & Remote Sensing Symposium*, pages 767--770. IEEE.

- Cheng, G., Yang, C., Yao, X., Guo, L., and Han, J. (2018). When deep learning meets metric learning: Remote sensing image scene classification via learning discriminative cnns. *IEEE Transactions on Geoscience and Remote Sensing*, 56(5):2811--2821.
- Cheriyadat, A. M. (2014). Unsupervised feature learning for aerial scene classification. *IEEE Transactions on Geoscience and Remote Sensing*, 52(1):439--451.
- Chevallier, E., Chevallier, A., and Angulo, J. (2015). N-ary mathematical morphology. In *International Symposium on Mathematical Morphology and Its Applications to Signal and Image Processing*, pages 339--350. Springer.
- Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In *Conference on Computer Vision and Pattern Recognition*, pages 1800--1807.
- Congalton, R. G. (1991). A review of assessing the accuracy of classifications of remotely sensed data. *Remote sensing of environment*, 37(1):35--46.
- Congalton, R. G. and Green, K. (2008). *Assessing the accuracy of remotely sensed data: principles and practices*. CRC press.
- Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *Conference on Computer Vision and Pattern Recognition*, pages 886--893.
- Dalla Mura, M., Benediktsson, J. A., Waske, B., and Bruzzone, L. (2010). Morphological attribute profiles for the analysis of very high resolution images. *IEEE Transactions on Geoscience and Remote Sensing*, 48(10):3747--3762.
- de O. Stehling, R., Nascimento, M. A., and Falcao, A. X. (2002). A compact and efficient image retrieval approach based on border/interior pixel classification. In *International Conference on Information and Knowledge Management*, pages 102--109.
- Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *Conference on Computer Vision and Pattern Recognition*, pages 248--255. IEEE.
- Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., and Darrell, T. (2014). Decaf: A deep convolutional activation feature for generic visual recognition. In *International Conference on Machine Learning*, pages 647--655.
- dos Santos, J., Penatti, O., Gosselin, P., Falcao, A., Philipp-Foliguet, S., and Torres, R. (2014). Efficient and effective hierarchical feature propagation. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, PP(99):1--12.

- dos Santos, J. A., Gosselin, P.-H., Philipp-Foliguet, S., Torres, R. d. S., and Falcao, A. X. (2013). Interactive multiscale classification of high-resolution remote sensing images. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 6(4):2020--2034.
- dos Santos, J. A., Penatti, O. A. B., and da Silva Torres, R. (2010). Evaluating the potential of texture and color descriptors for remote sensing image retrieval and classification. In *International Conference on Computer Vision Theory and Applications*, pages 203--208.
- dos Santos, J. A. d., Gosselin, P.-H., Philipp-Foliguet, S., Torres, R. d. S., and Falcao, A. X. (2012). Multiscale classification of remote sensing images. *IEEE Transactions on Geoscience and Remote Sensing*, 50(10):3764--3775.
- Douze, M., Jégou, H., Sandhawalia, H., Amsaleg, L., and Schmid, C. (2009). Evaluation of gist descriptors for web-scale image search. In *International Conference on Image and Video Retrieval*, pages 19:1--19:8.
- Du, B., Xiong, W., Wu, J., Zhang, L., Zhang, L., and Tao, D. (2017). Stacked convolutional denoising auto-encoders for feature representation. *IEEE Transactions on Cybernetics*, 47(4):1017--1027.
- Faria, F., Pedronette, D., dos Santos, J., Rocha, A., and Torres, R. (2014). Rank aggregation for pattern classifier selection in remote sensing images. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 7(4):1103--1115.
- Ferri, C., Hernández-Orallo, J., and Modroi, R. (2009). An experimental comparison of performance measures for classification. *Pattern Recognition Letters*, 30(1):27--38.
- Fingas, M. and Brown, C. (2014). Review of oil spill remote sensing. *Marine pollution bulletin*, 83(1):9--23.
- Firat, O., Can, G., and Yarman Vural, F. (2014). Representation learning for contextual object and region detection in remote sensing. In *International Conference on Pattern Recognition*, pages 3708--3713.
- Fustes, D., Cantorna, D., Dafonte, C., Arcay, B., Iglesias, A., and Manteiga, M. (2014). A cloud-integrated web platform for marine monitoring using gis and remote sensing. *Future Generation Computer Systems*, 34:155--160.

- Ge, Z., McCool, C., Sanderson, C., Bewley, A., Chen, Z., and Corke, P. (2015). Fine-grained bird species recognition via hierarchical subset learning. In *International Conference on Image Processing*, pages 561--565. IEEE.
- Gerke, M. (2015). Use of the stair vision library within the isprs 2d semantic labeling benchmark (vaihingen). In *ITC, University of Twente, Technical Report*.
- Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Conference on Computer Vision and Pattern Recognition*, pages 580--587. IEEE.
- Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. <http://www.deeplearningbook.org>.
- Gu, Y., Liu, T., Jia, X., Benediktsson, J. A., and Chanussot, J. (2016). Nonlinear multiple kernel learning with multiple-structure-element extended morphological profiles for hyperspectral image classification. *IEEE Transactions on Geoscience and Remote Sensing*, 54(6):3235--3247.
- Guan, H., Yu, Y., Ji, Z., Li, J., and Zhang, Q. (2015). Deep learning-based tree classification using mobile lidar data. *Remote Sensing Letters*, 6(11):864--873.
- Hamaguchi, R., Fujita, A., Nemoto, K., Imaizumi, T., and Hikosaka, S. (2018). Effective use of dilated convolutions for segmenting small object instances in remote sensing imagery. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1442--1450. IEEE.
- Hara, K., Jagadeesh, V., and Piramuthu, R. (2016). Fashion apparel detection: the role of deep convolutional neural network and pose-dependent priors. In *Applications of Computer Vision (WACV), 2016 IEEE Winter Conference on*, pages 1--9. IEEE.
- Haralick, R. M., Shanmugam, K., et al. (1973). Textural features for image classification. *IEEE Transactions on systems, man, and cybernetics*, (6):610--621.
- He, K., Zhang, X., Ren, S., and Sun, J. (2015). Spatial pyramid pooling in deep convolutional networks for visual recognition. *Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1904--1916.
- He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Conference on Computer Vision and Pattern Recognition*, pages 770--778.

- Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527--1554.
- Hu, F., Xia, G.-S., Hu, J., and Zhang, L. (2015). Transferring deep convolutional neural networks for the scene classification of high-resolution remote sensing imagery. *Remote Sensing*, 7(11):14680--14707.
- Hu, F., Xia, G.-S., Hu, J., Zhong, Y., and Xu, K. (2016). Fast binary coding for the scene classification of high-resolution remote sensing imagery. *Remote Sensing*, 8(7):555.
- Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Conference on Computer Vision and Pattern Recognition*, pages 2261--2269.
- Huang, J., Kumar, S. R., Mitra, M., Zhu, W., and Zabih, R. (1997). Image indexing using color correlograms. In *Conference on Computer Vision and Pattern Recognition*, pages 762--768.
- Huang, X., Zhang, L., and Gong, W. (2011). Information fusion of aerial images and lidar data in urban areas: vector-stacking, re-classification and post-processing approaches. *International Journal of Remote Sensing*, 32(1):69--84.
- Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.
- ISPRS (2018a). International society for photogrammetry and remote sensing (isprs). <http://www2.isprs.org/commissions/comm3/wg4/2d-sem-label-vaihingen.html>. Accessed: 2018-06-18.
- ISPRS (2018b). International society for photogrammetry and remote sensing (isprs). <http://www2.isprs.org/commissions/comm3/wg4/2d-sem-label-potsdam.html>. Accessed: 2018-06-18.
- Jarrett, K., Kavukcuoglu, K., Ranzato, M., and LeCun, Y. (2009). What is the best multi-stage architecture for object recognition? In *International Conference on Computer Vision*, pages 2146--2153. IEEE.
- Jensen, J. R. and Lulla, K. (1987). Introductory digital image processing: a remote sensing perspective.

- Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., and Darrell, T. (2014). Caffe: Convolutional architecture for fast feature embedding. In *ACM International Conference on Multimedia*, pages 675–678. ACM.
- Kemker, R., Salvaggio, C., and Kanan, C. (2018). Algorithms for semantic segmentation of multispectral remote sensing imagery using deep learning. *ISPRS Journal of Photogrammetry and Remote Sensing*, 145:60–77.
- Kimori, Y., Hikino, K., Nishimura, M., and Mano, S. (2016). Quantifying morphological features of actin cytoskeletal filaments in plant cells based on mathematical morphology. *Journal of theoretical biology*, 389:123–131.
- Krähenbühl, P. and Koltun, V. (2011). Efficient inference in fully connected crfs with gaussian edge potentials. In *Neural Information Processing Systems*, pages 109–117.
- Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Neural Information Processing Systems*, pages 1106–1114.
- Kumar, G. and Bhatia, P. K. (2014). A detailed review of feature extraction in image processing systems. In *Advanced Computing & Communication Technologies*, pages 5–12. IEEE.
- Lafferty, J. D., McCallum, A., and Pereira, F. C. N. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *International Conference on Machine Learning*, pages 282–289.
- Larochelle, H., Bengio, Y., Louradour, J., and Lamblin, P. (2009). Exploring strategies for training deep neural networks. *The Journal of Machine Learning Research*, 10:1–40.
- Lazebnik, S., Schmid, C., and Ponce, J. (2006). Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Conference on Computer Vision and Pattern Recognition*, pages 2169–2178.
- LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.
- LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.



- LeCun, Y., Jackel, L., Bottou, L., Brunot, A., Cortes, C., Denker, J., Drucker, H., Guyon, I., Muller, U., Sackinger, E., et al. (1995). Comparison of learning algorithms for handwritten digit recognition. In *International conference on artificial neural networks*, volume 60, pages 53--60.
- Leprince, S., Barbot, S., Ayoub, F., and Avouac, J.-P. (2007). Automatic and precise orthorectification, coregistration, and subpixel correlation of satellite images, application to ground deformation measurements. *IEEE Transactions on Geoscience and Remote Sensing*, 45(6):1529--1558.
- Li, H., Fu, K., Xu, G., Zheng, X., Ren, W., and Sun, X. (2017). Scene classification in remote sensing images using a two-stage neural network ensemble model. *Remote Sensing Letters*, 8(6):557--566.
- Li, J., Lin, D., Wang, Y., Xu, G., and Ding, C. (2019). Deep discriminative representation learning with attention map for scene classification. *arXiv preprint arXiv:1902.07967*.
- Li, L. T., Muñoz, J. A. V., Almeida, J., Calumby, R. T., Penatti, O. A. B., Dourado, I. C., Nogueira, K., Mendes Júnior, P. R., Pereira, A. M. L., Pedronette, D. C. G., Gonçalves, M. A., dos Santos, J. A., and Torres, R. d. S. (2015). Recod @ placing task of mediaeval 2015. In *Working Notes Proc. MediaEval Workshop*, page 2.
- Liao, W., Huang, X., Van Coillie, F., Gautama, S., Pižurica, A., Philips, W., Liu, H., Zhu, T., Shimoni, M., Moser, G., and Tuia, D. (2015). Processing of multiresolution thermal hyperspectral and digital color data: Outcome of the 2014 ieee grss data fusion contest. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 8(6):2984--2996.
- Lin, D. (2016). Deep unsupervised representation learning for remote sensing images. *arXiv preprint arXiv:1612.08879*.
- Lin, D., Fu, K., Wang, Y., Xu, G., and Sun, X. (2017). Marta gans: Unsupervised representation learning for remote sensing image classification. *IEEE Geoscience and Remote Sensing Letters*, 14(11):2092--2096.
- Liu, C., Frazier, P., and Kumar, L. (2007). Comparative assessment of the measures of thematic classification accuracy. *Remote sensing of environment*, 107(4):606--616.
- Liu, T., Gu, Y., Chanussot, J., and Dalla Mura, M. (2017). Multimorphological superpixel model for hyperspectral image classification. *IEEE Transactions on Geoscience and Remote Sensing*, 55(12):6950--6963.

- Liu, Y., Fan, B., Wang, L., Bai, J., Xiang, S., and Pan, C. (2018). Semantic labeling in very high resolution images via a self-cascaded convolutional neural network. *ISPRS Journal of Photogrammetry and Remote Sensing*, 145:78--95.
- Liu, Y., Piramanayagam, S., Monteiro, S. T., and Saber, E. (2017). Dense semantic labeling of very-high-resolution aerial imagery and lidar with fullyconvolutional neural networks and higher-order crfs. In *Conference on Computer Vision and Pattern Recognition Workshop*, pages 1561--1570.
- Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Conference on Computer Vision and Pattern Recognition*, pages 3431--3440.
- Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91--110.
- Ma, Y., Wu, H., Wang, L., Huang, B., Ranjan, R., Zomaya, A., and Jie, W. (2015). Remote sensing big data computing: Challenges and opportunities. *Future Generation Computer Systems*, 51:47--60.
- Maggiori, E., Tarabalka, Y., Charpiat, G., and Alliez, P. (2017). High-resolution aerial image labeling with convolutional neural networks. *IEEE Transactions on Geoscience and Remote Sensing*, 55(12):7092--7103.
- Makantasis, K., Karantzalos, K., Doulamis, A., and Doulamis, N. (2015). Deep supervised learning for hyperspectral data classification through convolutional neural networks. In *IEEE International Geoscience & Remote Sensing Symposium*, pages 4959--4962. IEEE.
- Marcu, A. and Leordeanu, M. (2016). Dual local-global contextual pathways for recognition in aerial imagery. *arXiv preprint arXiv:1605.05462*.
- Marmanis, D., Schindler, K., Wegner, J., Galliani, S., Datcu, M., and Stilla, U. (2018). Classification with an edge: Improving semantic image segmentation with boundary detection. *ISPRS Journal of Photogrammetry and Remote Sensing*, 135:158--172.
- Masci, J., Angulo, J., and Schmidhuber, J. (2013). A learning framework for morphological operators using counter-harmonic mean. In *International Symposium on Mathematical Morphology and Its Applications to Signal and Image Processing*, pages 329--340. Springer.

- Mellouli, D., Hamdani, T. M., Ayed, M. B., and Alimi, A. M. (2017). Morph-cnn: A morphological convolutional neural network for image classification. In *International Conference on Neural Information Processing*, pages 110--117. Springer.
- Mikolajczyk, K. and Schmid, C. (2005). A performance evaluation of local descriptors. *IEEE transactions on pattern analysis and machine intelligence*, 27(10):1615--1630.
- Mirzapour, F. and Ghassemian, H. (2015). Improving hyperspectral image classification by combining spectral, texture, and shape features. *International Journal of Remote Sensing*, 36(4):1070--1096.
- Muñoz, J. A. V., Li, L. T., Dourado, I. C., Nogueira, K., Fadel, S. G., Penatti, O. A. B., Almeida, J., Pereira, L. A. M., Calumby, R. T., dos Santos, J. A., and Torres, R. d. S. (2016). A ranking fusion approach for geographic-location prediction of multimedia objects. In *Working Notes Proc. MediaEval Workshop*, page 2.
- Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807--814.
- Ng, A., Ngiam, J., Foo, C. Y., Mai, Y., and Suen, C. (2011a). Feature extraction using convolution. [http://ufldl.stanford.edu/wiki/index.php/Feature\\_extraction\\_using\\_convolution/](http://ufldl.stanford.edu/wiki/index.php/Feature_extraction_using_convolution/). Accessed: 2017-08-02.
- Ng, A., Ngiam, J., Foo, C. Y., Mai, Y., and Suen, C. (2011b). Pooling. <http://ufldl.stanford.edu/wiki/index.php/Pooling/>. Accessed: 2017-02-08.
- Nogueira, K., Dalla Mura, M., Chanussot, J., Schwartz, W. R., and dos Santos, J. A. (2016a). Learning to semantically segment high-resolution remote sensing images. In *International Conference on Pattern Recognition*, pages 3566--3571. IEEE.
- Nogueira, K., Dalla Mura, M., Chanussot, J., Schwartz, W. R., and dos Santos, J. A. (2019a). Dynamic multi-context segmentation of remote sensing images based on convolutional networks. *IEEE Transactions on Geoscience and Remote Sensing*.
- Nogueira, K., dos Santos, J. A., Cancian, L., Borges, B. D., Silva, T. S. F., Morellato, L. P., and Torres, R. S. (2017a). Semantic segmentation of vegetation images acquired by unmanned aerial vehicles using an ensemble of convnets. *IEEE International Geoscience & Remote Sensing Symposium*.

- Nogueira, K., Dos Santos, J. A., Fornazari, T., Silva, T. S. F., Morellato, L. P., and Torres, R. d. S. (2016b). Towards vegetation species discrimination by using data-driven descriptors. In *Pattern Recognition in Remote Sensing (PRRS), 2016 9th IAPR Workshop on*, pages 1--6. IEEE.
- Nogueira, K., dos Santos, J. A., Menini, N., Silva, T. S. F., Morellato, L. P. C., and Torres, R. S. (2019b). Spatio-temporal vegetation pixel classification by using convolutional networks. *IEEE Geoscience and Remote Sensing Letters*.
- Nogueira, K., Fadel, S. G., Dourado, I. C., Werneck, R. d. O., Muñoz, J. A. V., Penatti, O. A. B., Calumby, R. T., Li, L. T., dos Santos, J. A., and Torres, R. d. S. (2017b). Data-driven flood detection using neural networks. In *Working Notes Proc. MediaEval Workshop*, page 2.
- Nogueira, K., Fadel, S. G., Dourado, Í. C., Werneck, R. O., Muñoz, J. A. V., Penatti, O. A., Calumby, R. T., Li, L. T., dos Santos, J. A., and Torres, R. S. (2018). Exploiting convnet diversity for flooding identification. *IEEE Geoscience and Remote Sensing Letters*, 15(9):1446--1450.
- Nogueira, K., Miranda, W. O., and Dos Santos, J. A. (2015a). Improving spatial feature representation from aerial scenes by using convolutional networks. In *Conference on Graphics, Patterns and Images (SIBGRAPI)*, pages 289--296. IEEE.
- Nogueira, K., Penatti, O. A., and dos Santos, J. A. (2017c). Towards better exploiting convolutional neural networks for remote sensing scene classification. *Pattern Recognition*, 61:539--556.
- Nogueira, K., Schwartz, W. R., and dos Santos, J. A. (2015b). Coffee crop recognition using multi-scale convolutional neural networks. In *Iberoamerican Congress on Pattern Recognition*, pages 67--74. Springer.
- Noh, H., Hong, S., and Han, B. (2015). Learning deconvolution network for semantic segmentation. In *International Conference on Computer Vision*, pages 1520--1528.
- Oliva, A. and Torralba, A. (2001). Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision*, 42(3):145--175. ISSN 0920-5691.
- Paisitkriangkrai, S., Sherrah, J., Janney, P., and Hengel, A. (2015). Effective semantic pixel labelling with convolutional networks and conditional random fields. In *Conference on Computer Vision and Pattern Recognition Workshop*, pages 36--43.

- Paisitkriangkrai, S., Sherrah, J., Janney, P., and van den Hengel, A. (2016). Semantic labeling of aerial and satellite imagery. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 9(7):2868--2881.
- Penatti, O. A., Nogueira, K., and dos Santos, J. A. (2015). Do deep features generalize from everyday objects to remote sensing and aerial scenes domains? In *Conference on Computer Vision and Pattern Recognition Workshop*, pages 44--51.
- Penatti, O. A. B., Silva, F. B., Valle, E., Gouet-Brunet, V., and da S. Torres, R. (2014). Visual word spatial arrangement for image retrieval and classification. *Pattern Recognition*, 47(2):705--720. ISSN 0031-3203.
- Penatti, O. A. B., Valle, E., and da S. Torres, R. (2012). Comparative study of global color and texture descriptors for web image retrieval. *Journal of Visual Communication and Image Representation*, 23(2):359--380.
- Peng, C., Li, Y., Jiao, L., Chen, Y., and Shang, R. (2019). Densely based multi-scale and multi-modal fully convolutional networks for high-resolution remote-sensing image semantic segmentation. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*.
- Perronnin, F., Sánchez, J., and Mensink, T. (2010). Improving the Fisher Kernel for Large-Scale Image Classification. In *European Conference on Computer Vision*, pages 143--156.
- Pesaresi, M. and Benediktsson, J. A. (2001). A new approach for the morphological segmentation of high-resolution satellite imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 39(2):309--320.
- Piramanayagam, S., Schwartzkopf, W., Koehler, F., and Saber, E. (2016). Classification of remote sensed images using random forests and deep learning framework. In *Image and Signal Processing for Remote Sensing XXII*, volume 10004, page 100040L. International Society for Optics and Photonics.
- Razavian, A. S., Azizpour, H., Sullivan, J., and Carlsson, S. (2014). Cnn features off-the-shelf: An astounding baseline for recognition. In *Conference on Computer Vision and Pattern Recognition Workshop*, pages 512--519.
- Sabour, S., Frosst, N., and Hinton, G. E. (2017). Dynamic routing between capsules. In *Neural Information Processing Systems*, pages 3856--3866.

- Santana, T. M., Nogueira, K., Machado, A. M., and dos Santos, J. A. (2017). Deep contextual description of superpixels for aerial urban scenes classification. In *IEEE International Geoscience & Remote Sensing Symposium*, pages 3027–3031.
- Seo, Y., Park, B., Yoon, S.-C., Lawrence, K. C., and Gamble, G. R. (2018). Morphological image analysis for foodborne bacteria classification. *Transactions of the American Society of Agricultural and Biological Engineers*, 61:5–13.
- Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., and LeCun, Y. (2014). Overfeat: Integrated recognition, localization and detection using convolutional networks. In *International Conference on Learning Representations*. CBLS.
- Serra, J. and Soille, P. (2012). *Mathematical morphology and its applications to image processing*, volume 2. Springer Science & Business Media.
- Sherrah, J. (2016). Fully convolutional networks for dense semantic labelling of high-resolution aerial imagery. *arXiv preprint arXiv:1606.02585*.
- Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.
- Sivic, J. and Zisserman, A. (2003). Video google: a text retrieval approach to object matching in videos. In *International Conference on Computer Vision*, volume 2, pages 1470--1477.
- Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929--1958.
- Sunderhauf, N., McCool, C., Upcroft, B., and Tristan, P. (2014). Fine-grained plant classification using convolutional neural networks for feature extraction. In *Working notes of CLEF 2014 conference*.
- Swain, M. J. and Ballard, D. H. (1991). Color indexing. *International Journal of Computer Vision*, 7(1):11--32. ISSN 0920-5691.
- Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Conference on Computer Vision and Pattern Recognition*, pages 1--9.
- Tanizaki, H. (2013). *Nonlinear filters: estimation and applications*. Springer Science & Business Media.

- Tao, B. and Dickinson, B. W. (2000). Texture recognition and image retrieval using gradient indexing. *Journal of Visual Communication and Image Representation*, 11(3):327--342. ISSN 1047-3203.
- Taylor, J. R. and Lovell, S. T. (2012). Mapping public and private spaces of urban agriculture in chicago through the analysis of high-resolution aerial images in google earth. *Landscape and Urban Planning*, 108(1):57--70.
- Tayyebi, A., Pijanowski, B. C., and Tayyebi, A. H. (2011). An urban growth boundary model using neural networks, gis and radial parameterization: An application to tehran, iran. *Landscape and Urban Planning*, 100(1):35--44.
- Tuia, D., Volpi, M., Copa, L., Kanevski, M., and Munoz-Mari, J. (2011). A survey of active learning algorithms for supervised remote sensing image classification. *IEEE Journal of Selected Topics in Signal Processing*, 5(3):606--617.
- Tuytelaars, T. and Mikolajczyk, K. (2007). Local invariant feature detectors: A survey. *Foundations and Trends in Computer Graphics and Vision*, 3(3):177--280.
- van de Sande, K. E. A., Gevers, T., and Snoek, C. G. M. (2010). Evaluating color descriptors for object and scene recognition. *Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1582--1596.
- van de Sande, K. E. A., Gevers, T., and Snoek, C. G. M. (2011). Empowering visual categorization with the gpu. *Transactions on Multimedia*, 13(1):60--70.
- van Gemert, J. C., Veenman, C. J., Smeulders, A. W. M., and Geusebroek, J.-M. (2010). Visual word ambiguity. *Transactions on Pattern Analysis and Machine Intelligence*, 32:1271--1283.
- Vedaldi, A. and Fulkerson, B. (2010). Vlfeat: An open and portable library of computer vision algorithms. In *ACM International Conference on Multimedia*, pages 1469--1472. ACM.
- Volpi, M. and Ferrari, V. (2015). Semantic segmentation of urban scenes by learning local class interactions. In *Conference on Computer Vision and Pattern Recognition Workshop*, pages 1--9.
- Volpi, M. and Tuia, D. (2017). Dense semantic labeling of subdecimeter resolution images with convolutional neural networks. *IEEE Transactions on Geoscience and Remote Sensing*, 55(2):881--893.

- Wang, A., He, X., Ghamisi, P., and Chen, Y. (2018a). Lidar data classification using morphological profiles and convolutional neural networks. *IEEE Geoscience and Remote Sensing Letters*, 15(5):774--778.
- Wang, H., Wang, Y., Zhang, Q., Xiang, S., and Pan, C. (2017). Gated convolutional neural network for semantic segmentation in high-resolution images. *Remote Sensing*, 9(5):446.
- Wang, P., Chen, P., Yuan, Y., Liu, D., Huang, Z., Hou, X., and Cottrell, G. (2018b). Understanding convolution for semantic segmentation. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1451--1460. IEEE.
- Wilkinson, G. G. (2005). Results and implications of a study of fifteen years of satellite image classification experiments. *IEEE Transactions on Geoscience and Remote Sensing*, 43(3):433--440.
- Xia, G.-S., Yang, W., Delon, J., Gousseau, Y., Sun, H., and Maitre, H. (2010). Structural high-resolution satellite image indexing. In *ISPRS TC VII Symposium-100 Years ISPRS*, volume 38, pages 298--303.
- Xia, J., Dalla Mura, M., Chanussot, J., Du, P., and He, X. (2015). Random subspace ensembles for hyperspectral image classification with extended morphological attribute profiles. *IEEE Transactions on Geoscience and Remote Sensing*, 53(9):4768--4786.
- Xie, M., Jean, N., Burke, M., Lobell, D., and Ermon, S. (2016). Transfer learning from deep features for remote sensing and poverty mapping. In *AAAI Conference on Artificial Intelligence*, pages 3929--3935.
- Xu, S., Mu, X., Chai, D., and Zhang, X. (2018). Remote sensing image scene classification based on generative adversarial networks. *Remote sensing letters*, 9(7):617--626.
- Xu, X., Li, J., Huang, X., Dalla Mura, M., and Plaza, A. (2016). Multiple morphological component analysis based decomposition for remote sensing image classification. *IEEE Transactions on Geoscience and Remote Sensing*, 54(5):3083--3102.
- Yang, Y. and Newsam, S. (2008). Comparing sift descriptors and gabor texture features for classification of remote sensed imagery. In *International Conference on Image Processing*, pages 1852--1855.
- Yang, Y. and Newsam, S. (2010). Bag-of-visual-words and spatial extensions for land-use classification. *ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*.



- Yang, Y. and Newsam, S. (2013). Geographic image retrieval using local invariant features. *IEEE Transactions on Geoscience and Remote Sensing*, 51(2):818–832.
- Yu, F. and Koltun, V. (2015). Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*.
- Yu, H., Yang, W., Xia, G.-S., and Liu, G. (2016). A color-texture-structure descriptor for high-resolution satellite image classification. *Remote Sensing*, 8(3):259.
- Yu, X., Wu, X., Luo, C., and Ren, P. (2017). Deep learning in remote sensing scene classification: a data augmentation enhanced convolutional neural network framework. *GIScience & Remote Sensing*, pages 1--18.
- Yue, J., Zhao, W., Mao, S., and Liu, H. (2015). Spectral–spatial classification of hyperspectral images using deep convolutional neural networks. *Remote Sensing Letters*, 6(6):468--477.
- Zhang, F., Du, B., and Zhang, L. (2015). Saliency-guided unsupervised feature learning for scene classification. *IEEE Transactions on Geoscience and Remote Sensing*, 53(4):2175–2184.
- Zhang, F., Du, B., and Zhang, L. (2016). Scene classification via a gradient boosting random convolutional network framework. *IEEE Transactions on Geoscience and Remote Sensing*, 54(3):1793--1802.
- Zhang, W., Tang, P., and Zhao, L. (2019). Remote sensing image scene classification using cnn-capsnet. *Remote Sensing*, 11(5):494.