



# A framework for remote sensing images processing using deep learning techniques

Romain Cresson

## ► To cite this version:

Romain Cresson. A framework for remote sensing images processing using deep learning techniques. IEEE Geoscience and Remote Sensing Letters, 2019, 16 (1), pp.25-29. 10.1109/LGRS.2018.2867949 . hal-02607952

**HAL Id: hal-02607952**

**<https://hal.inrae.fr/hal-02607952>**

Submitted on 16 May 2020

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# A framework for remote sensing images processing using deep learning techniques

Rémi Cresson\*

Deep learning techniques are becoming increasingly important to solve a number of image processing tasks. Among common algorithms, Convolutional Neural Networks and Recurrent Neural Networks based systems achieve state of the art results on satellite and aerial imagery in many applications. While these approaches are subject to scientific interest, there is currently no operational and generic implementation available at user-level for the remote sensing community. In this paper, we presents a framework enabling the use of deep learning techniques with remote sensing images and geospatial data. Our solution takes roots in two extensively used open-source libraries, the remote sensing image processing library Orfeo ToolBox, and the high performance numerical computation library TensorFlow. It can apply deep nets without restriction on images size and is computationally efficient, regardless hardware configuration.

## I. INTRODUCTION

The volume of earth observation data never stopped to increase in recent years. In one hand, the use of very high spatial resolution satellites (VHRS) images is booming, and sensors are each generation sharper [1]. There is also more and more sensors with close temporal acquisition, such as the Sentinel constellation. In the other hand, community-based geographic information gathering systems are expanding each years in-situ geospatial data bases like OpenStreetMap [2]. Deep learning (DL) is a growing trend in big data analysis, and had a breakthrough impact in the last few years on such diverse domains like Computer Vision, Speech Recognition, Image Processing, and Remote Sensing (RS). Convolutional Neural Networks (CNN) are designed to extract features in images, enabling image recognition, object detection, semantic segmentation. Recurrent Neural Networks (RNNs) are suited for sequential data analysis, such as speech recognition and action recognition tasks.

In recent years, a number of studies have shown that RS benefits strongly from these new approaches, thanks to the availability of data and computing resources [3]. Many typical RS problems have been successfully addressed with these architectures: synthetic aperture radar interpretation with target recognition [4], classification from time series [5], and parameter inversion [6], hyperspectral image analysis with classification [7], anomaly detection [8], VHRS images interpretation with scene classification [9, 10], object detection [11], image retrieval [12], classification from time-series [13]. DL has addressed other issues in RS, like data fusion (see [14] for a review) e.g. multimodal classification [15], pansharpening [16], and 3D reconstruction.

DL allows researchers in RS to move beyond usual approaches and tackle a number of problems with solid results. However, there is still challenges remaining ahead in operational context. First, one crucial point is software implementation. Entering into

DL requires to familiarize with a framework. Several popular open-source frameworks exist, such as Caffe, Torch, Theano, and TensorFlow, but implementing new methods requires an extensive programming background and DL knowledge, holding back DL democratization in RS community. Conversely, many studies rely exclusively on pre-processed, densely annotated public datasets like UC Merced [17] or Potsdam [18] because data extraction requires knowledge of geospatial standards and tools like Geographic Information Systems and RS image processing software. Secondly, the need of tools that scale enough to the huge size of real-world RS images and datasets, raises as a major computing challenge. While RS images processing software have internal mechanisms to deal with the large stacks of images, this problematic is out of DL frameworks scope. Another point is that algorithms presented in the literature are often serving one unique purpose. Therefore, coding is not generic and each new algorithm implementation require new effort and expertise. Finally, recent studies have pointed that using the features computed with deep nets in machine learning algorithms like Support Vector Machines (SVM) or Random Forests, offers state of the art results [19–21]. However, even if RS image processing software implement generally a machine learning framework, there is no existing one enabling the combination of DL algorithms with already implemented well-known machine learning algorithms.

Regarding these matters, we propose a generic framework to enable DL for RS images, which (i) is open-source and cross-platform, (ii) enables users without programming knowledge to use deeps nets on RS images, (iii) is integrated in an existing rich machine learning framework for RS images, (iv) is computationally efficient and allows the processing of very large RS images. Our approach is based on the RS image processing library, Orfeo Toolbox (OTB) [22] and rely on the high performance numerical computation library for training and inference on deep neural networks, TensorFlow (TF) [23]. The existing machine learning framework of OTB includes tools for RS data selection, sampling, training, classification, regression, with many strategies and parameters fine tuning. We extend this framework using TF to enable the processing of RS images with deep nets. We first give in

\* R. Cresson is with the IRSTEA, UMR TETIS, BP 5095, Montpellier, France. [remi-dot-cresson-at-irstea.fr](mailto:remi-dot-cresson-at-irstea.fr)

Section II a detailed description of our method. We then analyze the scalability of our approach with state of the art deep nets and Spot-7 and Sentinel-2 images (Section III). We finally discuss the main advantages and limits of our approach in Section IV.

## II. METHOD

### A. Overview

The Orfeo ToolBox (OTB) is a library for RS image processing, built on top of an application development framework widely used in medical image processing, the Insight Toolkit (ITK) [24]. The machine learning framework of OTB is able to process large datasets at continental scale for land mapping [25] and benefits from High Performance Computing (HPC) architectures like clusters [26]. TensorFlow (TF) is a library for dataflow programming. It is a symbolic math library, and is intensively used for machine learning applications such as deep nets. It can also operate at large scale in HPC architectures like GPU. We aim to extend the existing rich machine learning framework of OTB with TF. Implementation of state of the art deep nets should be enabled with the minimum effort, and the opportunity to apply them on RS images must be granted to non-developers, namely users. The existing user-oriented machine learning framework of OTB must be preserved. We also sought the implementation of a component which can be used in a transparent way inside any OTB pipeline, to enable the combination of already implemented approaches with DL. In the following sections, we provide description of the libraries processing frameworks (Sections IIB and IIC). Then we introduce our low-level OTB compliant component in Section IID and our new user-oriented OTB applications in Section IIE. Finally, we analyze performances of our approach with state of the art RS deep nets in Section III.

### B. Libraries paradigms

This section describes the processing logic of OTB and TF.

#### 1. OTB workflow

A *pipeline* refers to a directed graph of process objects, that can be either sources (initiating the pipeline), filters (processing the data) or mappers (typically, write a result on disk). Sources and filters can generate one or multiple data objects (e.g. image, vector). In the same way, filters and mappers consume one or multiple data objects. The architecture of OTB inherits from ITK and hides the complexity of internal mechanisms for pipeline execution, which involve several steps. Detailed description of

the pipeline mechanism can be found in [27] including figures and sequence diagram. The execution of a pipeline starts from a mapper, triggering its upstream filter(s). When a filter is triggered, information about mandatory input data (i.e. information about output data of upstream process object(s)) is also requested upstream. In this way, it is propagated back through the pipeline, from mappers to sources via filters. Once this request reach sources, data objects information are generated (e.g. image size, pixel spacing) then propagated downstream to mappers. It should be noted that filters can potentially modify these information, according to the process they implement (e.g. changing image size) which is the case with image re-sampling for instance. Finally, they reach the mapper, initiating the data processing. Information regarding the size of the image that will be produced, is then used by the mapper to choose a splitting strategy. The default splitting scheme is based on the system memory specification. Other strategies can be chosen, e.g. striped or tiled regions with fixed dimensions. Then, the mapper requests its input image to upstream filter(s) sequentially, region after region. The data request and generation is handled through the pipeline in the same way as for the information: once the request reaches the sources, initiating the pipeline, the requested region is produced, then processed through filters, to finally end in the mapper. The pipeline execution continues with the next image region until the entire output is generated. This mechanism, named *streaming*, enables the processing of very large images regardless the memory consumption of the process objects composing the pipeline.

#### 2. TF workflow

TF uses symbolic programming, which distinguish definition of computations from their execution. In TF, tensors are abstraction objects of the operations and objects in the memory, simplifying manipulation regardless the computing environment (e.g. CPU or GPU). A TF model consists in operations arranged into a graph of nodes. Each node can viewed as an operation taking zero or more tensors as inputs, and producing a tensor. This data flow graph defines the operations (e.g. linear algebra operators), but the computations are performed within the so-called *session*. A high-level API enables the construction of TF graphs, and the session runs the graph in delegating calculations to low level, highly-optimized routines. Among tensors, we can distinguish concepts such as *Placeholders*, *Constants* and *Variables*. A Placeholder is a symbol hosting input data, e.g. a set of images. As its name indicates, Constants are tensors with constant values, and Variables hold non persistent values, e.g. parameters to estimate during training. Variables must be explicitly initialized, and can be saved or restored during a session along with the graph. A number of tools enable design of TF models, like the TF Python API or user-oriented software developed

by the TF community.

### C. Flowing the pipeline

In this section, we present prerequisites for the integration of a process object that runs TF session for generic deep nets, in a OTB pipeline with RS images as data objects. In the pipeline workflow, the generation of images information and the requested regions computation by process objects, are crucial steps. We denote the *spacing* the physical size of a single RS image pixel. The generation of output images information includes origin, spacing, size, and additional RS metadata e.g. projection reference. Process objects must also propagate the requested regions to input images. Regarding deep nets implementation, and particularly CNNs, this process must be carefully handled. CNNs usually involve several operations, mostly including a succession of convolutions, pooling, and non-linear functions. There is also a number of deriving operators e.g. transposed convolution. Most of these operators modify the size and spacing of the result. For example, convolution can change the output image size, depending its kernel size and input padding. It can also change the spacing of the output if performed with strides, that is, the step of which is shifted the filter at each computation. Pooling operators are also a common kind of operator that modify the output size and spacing, depending of the stride and the size of the sliding window. We should note that a number of other operators change size and scale the spacing of the output.

Considering a process object implementing such operations, it must propagate requested regions of images to its inputs. In the following, we introduce a generic description of size and spacing modifications that a deep net induces in processing RS images. We name the *scaling factor* of a particular output of the net, the ratio between the output spacing and a reference spacing (typically one input image feeding the deep net). This parameter enables the description of any change of physical size of the pixel introduced by operators such as pooling or convolution involving non-unitary strides. In addition, each input has its own *receptive field*, the input space that a particular output of the net is affected by. In the same way, each output has its own *expression field*, the output space resulting from the receptive field. Images regions and spacing modifications induced by a TF graph are thus defined with the receptive field of inputs, and the scaling factor and expression field of outputs.

### D. Running TF session in process object

We introduce a new OTB process object which internally invokes the TF engine and enforces the pipeline execution described in Section II C and hence, can seamlessly process large images. It takes one or

multiple input images, and produces zero or multiple output images. The input placeholders of the TF model are fed with the input images of the process object. After the session run, computed tensors are assigned to the process object outputs. Placeholders corresponding to input images, as well as tensors corresponding to output images, are both specified as named in the TF model. The process object uses receptive field of inputs, as well as expression field and scaling factor of outputs, to propagate requested input images regions and generate information of output images. Finally, two processing modes are currently supported:

#### 1. Patch-based

Extract and process patches independently at regular intervals. Patches sizes are equal to the receptive field sizes of inputs. For each input, a tensor with a number of elements equal to the number of patches feeds the TF model.

#### 2. Fully-convolutional

Unlike patch-based mode, it allows the processing of an entire requested region. For each input, a tensor composed of one single element, corresponding to the input requested region, is fed to the TF model. This mode requires that receptive fields, expression fields and scale factors are consistent with operators implemented in the TF model, input images physical spacing and alignment. Blocking artifact is avoided by computing input images regions aligned to the expression field sizes of the model, and keeping only the subset of the output corresponding to the requested output region.

### E. New OTB applications

OTB applications generally implement pipelines composed of several process objects connected together. We first provide a new application dedicated to RS images sampling suited for DL. Then, using our new filter described in Section II D, we provide new applications for TF model training and serving. Finally, we introduce some new applications consisting in an assembly of multiple OTB applications. All these new applications are integrated seamlessly in the existing machine learning framework of OTB.

#### 1. Sampling

The existing machine learning framework of OTB includes sample selection and extraction applications suited for geospatial data like vector layers and RS images. Multiple sampling strategies can be chosen,

with fine control of parameters. However, DL on images usually involves reference data made of patches, but the existing application performs pixel-wise sample extraction. We introduce a new application that performs patches extraction in input images, from samples positions resulting from the existing sample selection application of OTB. As we want to stick to RS images files formats, samples are concatenated in rows to form one unique big image of patches. Typically, considering a number  $n$  of sampled patches of size  $[S_{row}, S_{col}, S_{ch}]$ , the resulting image have a size of  $[n \cdot S_{row}, S_{col}, S_{ch}]$ . The main advantage of this technique is that pixel interleaving is unchanged (typically *row*, *column*, *channel*), guaranteeing the efficient copy and processing.

## 2. Training

Since the high level TF API is Python, we provide a light Python environment to allow developers to build and train their models from patches images produced using our sampling application described in Section II E 1. As there is already a number of available TF models and existing user-oriented opens-source applications to create TF models, we chose not to focus it.

Our contribution for RS images deep net training is a user-oriented OTB application dedicated for training existing TF models. The application can import any existing TF model. It can restore from file model variables before the training, or save them to disk after training. Thus this application can be used either to train a particular model from scratch, or perform fine tuning depending on the variables restored. The training can be performed over one or multiple inputs given their corresponding TF placeholders names and provide usual evaluation metrics.

## 3. Model serving

We introduce a single OTB application dedicated to TF model serving, that implements a pipeline with the filter described in Section II D. It produces one output image resulting from TF model computations. As the entire pipeline implements the streaming mechanism described in Section II C, it can process one or multiple images of arbitrary sizes. The user can adjust the produced images blocks sizes, thanks to the internal OTB application architecture. This application provides a generic support for the operational deployment of trained deep nets on RS images.

## 4. Composite applications

Composite applications are OTB applications connected together. Recent studies have suggested that deep nets features can be used as input features of

**Table 1.** Models parameters

	Maggiori FC17	Ienco M3
RF	80x80 ( <i>VHRS</i> )	1x1 ( <i>TS</i> ), 25x25 ( <i>VHRS</i> )
EF	16x16	1x1
Mode	Fully-conv	Patch-based
Reference	<i>VHRS</i>	<i>TS</i>

**Table 2.** Characteristics of the dataset

Id	Product type	Size ( $col \times row \times band$ )	Pixel spacing
<i>VHRS</i>	Spot-7 PXS	24110 $\times$ 33740 $\times$ 4	1.5 meters
<i>TS</i>	S2 Time series	3616 $\times$ 5060 $\times$ 592	10 meters

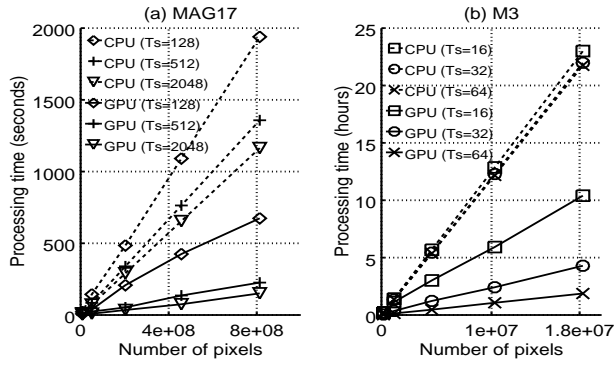
algorithms like classification, leading state of the art results. Regarding RS image classification, OTB already implement a number of algorithms in its classification application, e.g. SVM, Random Forests, boost classifier, decision tree classifier, gradient boosted tree classifier, normal Bayes classifier. In the sake of demonstrating the operational aspect of our approach, we implement a composite application that reuse our model serving application (described in II E 3) as input of two existing OTB applications: the *TrainImagesClassifier* and *ImagesClassifier* applications respectively dedicated to train a classifier and perform an image classification.

## III. EXPERIMENTS

In this section, we analyze performances of our TF model serving application. We conduct a series of experiments on two representative state of the art deep nets for RS images, the Maggiori et al. fully convolutional model [10] and the Ienco M3 data fusion model [13]. Both perform land cover classification, but are different in terms of architecture as well as implementation. The model of Maggiori is a single-input CNN composed exclusively of convolutions, pooling and activation functions, allowing to process one image region with a single convolution. The M3 model is an hybrid CNN-RNN model that inputs time series (TS) and one VHRS image. Unlike the Maggiori model, it consumes patches of the VHRS image and one pixel stack of time series, to perform a single estimation at one only location. Table 1 summarizes models parameters: Expression Field (EF), Receptive Fields (RF), mode and input reference image. Both models have an unitary scale factor. The used images are presented in Table 2: one VHRS Spot-7 image acquired within the GEOSUD project [28] and one S2 TS provided by the THEIA Land Data Center [29]. The TS consists in spectral bands resampled at 10m and additional radiometric indices. The Maggiori net was trained to estimate two classes and M3 eight classes.

All experiments are conducted on two computers. The first is a basic workstation without GPU support (Intel Xeon CPU E5-2609 @ 2.50GHz and 16 GB of





**Figure 1.** Measured processing times for different images sizes for (a) the Maggiori model (MAG17) and (b) lenco *M3* model (*M3*)

RAM). The second is a server with GPU support (Intel (R) Xeon (R) CPU E5-2667 v4@3.20Ghz with 256 GB of RAM and TITAN X GPU). Libraries versions are OTB 6.7 and TF 1.7. We measure run times of our applications using the two models on both configurations, for different computed images sizes and various tile size. We presents separately the results for both models because their processing time differs largely due to the fact that *M3* is more complex than the Maggiori model. Figure 1 shows that processing time is linear to the number of pixels of the produced image, regardless parameters. Very large input datasets are entirely processed thank to the *streaming* mechanism, even on the basic workstation with only 16Gb RAM. TF computations are benefiting strongly from the GPU support especially for image convolution, but CPU workstation processing time is reasonable with less than 24 hours for *M3* (Fig. 1b) and less than 1 hour for the Maggiori model (Fig. 1a).

#### IV. DISCUSSION

##### A. Operational deep learning framework for RS images

Our first goal was to provide a generic DL framework for RS images processing. We propose a solution which takes roots in the OTB library. The original machine learning framework was enriched to embed TF for training and using deep neural networks. We introduce a new component enforcing the OTB workflow, that developers can implement in pipelines. We present a set of new applications that users without programming knowledge can operate: patches sampling, model training and serving, and also training and classification applications performing on features of any deep net exported as a TF model. Our approach is successfully applied to common RS images with two representative state of the art deep nets. Processing times measurements have shown a great

scalability and also that our applications can run deep nets without restriction on dataset size and regardless hardware configuration.

##### B. Limits and further research

It should be noted that the presented method also has limits. The user has responsibility in providing crucial parameters to applications. In particular, receptive field, expression field and scale factor of deep nets. In future development, those parameters could be extracted from the serialized TF graph. In the same way, the memory footprint is computed within the OTB pipeline, but does not takes in account the memory consumed internally by TF during the execution of the computational graph: the user is thus responsible of fine tuning using the OTB application engine parameters.

#### V. CONCLUSION

This work was carried out with a view toward processing remote sensing (RS) images using deep learning (DL) techniques from the user perspective. We propose a generic framework based on Orfeo Toolbox (OTB) and TensorFlow (TF) libraries. We have successfully applied existing state of the art deep nets on common RS images using our framework and shown a good computational efficiency, without restriction on images sizes and regardless hardware configuration. Our approach enables users without coding skills to use deep nets in their RS applications, and developers to create operational RS processing pipelines benefiting from the development framework of OTB library. Our approach allows the combination of the existing OTB machine learning framework with deep nets. The integration of DL processes in high performance computing architectures is enabled thank to the heterogeneous devices supported by the used libraries. Further research could focus on improving the automatic retrieval of nets parameters and memory footprint. The source code and documentation corresponding to the implementation presented in this paper is available at [30], and the exposed framework will be proposed as official contribution in the forthcoming releases of OTB.

#### ACKNOWLEDGMENTS

The author thanks the OTB team, Dino Ienco and Emmanuel Maggiori. This work was supported by public funds received through GEOSUD, a project (ANR-10-EQPX-20) of the *Investissements d'Avenir* program managed by the French National Research Agency. The author would also like to thank the reviewers for their valuable suggestions.

- [1] J. A. Benediktsson, J. Chanussot, and W. M. Moon. Advances in very-high-resolution remote sensing. *Proceedings of the IEEE*, 101, 2013.
- [2] Mordechai Haklay and Patrick Weber. Openstreetmap: User-generated street maps. *IEEE Pervasive Computing*, 7(4):12–18, 2008.
- [3] Xiao Xiang Zhu, Devis Tuia, Lichao Mou, Gui-Song Xia, Liangpei Zhang, Feng Xu, and Friedrich Fraundorfer. Deep learning in remote sensing: a review. *arXiv preprint arXiv:1710.03959*, 2017.
- [4] Sizhe Chen and Haipeng Wang. Sar target recognition based on deep learning. In *Data Science and Advanced Analytics (DSAA), 2014 International Conference on*, pages 541–547. IEEE, 2014.
- [5] Dinh Ho Tong Minh, Dino Ienco, Raffaele Gaetano, Nathalie Lalande, Emile Ndikumana, Faycal Osman, and Pierre Maurel. Deep recurrent neural networks for winter vegetation quality mapping via multitemporal sar sentinel-1. *IEEE Geoscience and Remote Sensing Letters*, 15(3):464–468, 2018.
- [6] Lei Wang, K Andrea Scott, Linlin Xu, and David A Clausi. Sea ice concentration estimation during melt from dual-pol sar scenes using deep convolutional neural networks: A case study. *IEEE Transactions on Geoscience and Remote Sensing*, 54(8):4524–4533, 2016.
- [7] Y. Chen, Z. Lin, X. Zhao, G. Wang, and Y. Gu. Deep learning-based classification of hyperspectral data. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 7(6):2094–2107, June 2014.
- [8] W. Li, G. Wu, and Q. Du. Transferred deep learning for anomaly detection in hyperspectral imagery. *IEEE Geoscience and Remote Sensing Letters*, 14(5):597–601, May 2017.
- [9] Michele Volpi and Devis Tuia. Dense semantic labeling of subdecimeter resolution images with convolutional neural networks. *IEEE Transactions on Geoscience and Remote Sensing*, 55(2):881–893, 2017.
- [10] Emmanuel Maggiori, Yuliya Tarabalka, Guillaume Charpiat, and Pierre Alliez. High-resolution aerial image labeling with convolutional neural networks. *IEEE Transactions on Geoscience and Remote Sensing*, 55(12):7092–7103, 2017.
- [11] Gong Cheng and Junwei Han. A survey on object detection in optical remote sensing images. *ISPRS Journal of Photogrammetry and Remote Sensing*, 117:11–28, 2016.
- [12] Paolo Napoletano. Visual descriptors for content-based retrieval of remote-sensing images. *International Journal of Remote Sensing*, 39(5):1343–1376, 2018.
- [13] Dino Ienco, Raffaele Gaetano, Claire Dupaquier, and Pierre Maurel. Land cover classification via multitemporal spatial data by deep recurrent neural networks. *IEEE Geoscience and Remote Sensing Letters*, 14(10):1685–1689, 2017.
- [14] Luis Gomez-Chova, Devis Tuia, Gabriele Moser, and Gustau Camps-Valls. Multimodal classification of remote sensing images: A review and future directions. *Proceedings of the IEEE*, 103(9):1560–1584, 2015.
- [15] P Benedetti, D Ienco, R Gaetano, K Osé, R Pensa, and S Dupuy. M3fusion: A deep learning architecture for multi-{Scale/Modal/Temporal} satellite data fusion. *arXiv preprint arXiv:1803.01945*, 2018.
- [16] Giuseppe Masi, Davide Cozzolino, Luisa Verdoliva, and Giuseppe Scarpa. Pansharpening by convolutional neural networks. *Remote Sensing*, 8(7):594, 2016.
- [17] Yi Yang and Shawn Newsam. Bag-of-visual-words and spatial extensions for land-use classification. In *Proceedings of the 18th SIGSPATIAL international conference on advances in geographic information systems*, pages 270–279. ACM, 2010.
- [18] Potsdam dataset. [www2.isprs.org/commissions/comm3/wg4/2d-sem-label-potsdam.html](http://www2.isprs.org/commissions/comm3/wg4/2d-sem-label-potsdam.html). Accessed: 2018-04-30.
- [19] Keiller Nogueira, Otávio AB Penatti, and Jefersson A dos Santos. Towards better exploiting convolutional neural networks for remote sensing scene classification. *Pattern Recognition*, 61:539–556, 2017.
- [20] Dimitrios Marmanis, Mihai Datcu, Thomas Esch, and Uwe Stilla. Deep learning earth observation classification using imagenet pretrained networks. *IEEE Geoscience and Remote Sensing Letters*, 13(1):105–109, 2016.
- [21] Fan Hu, Gui-Song Xia, Jingwen Hu, and Liangpei Zhang. Transferring deep convolutional neural networks for the scene classification of high-resolution remote sensing imagery. *Remote Sensing*, 7(11):14680–14707, 2015.
- [22] Manuel Grizonnet, Julien Michel, Victor Poughon, Jordi Inglada, Mickaël Savinaud, and Rémi Cresson. Orfeo toolbox: open source processing of remote sensing images. *Open Geospatial Data, Software and Standards*, 2(1):15, 2017.
- [23] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.
- [24] T. S. Yoo, M. J. Ackerman, W. E. Laorensen, W. Schroeder, V. Chalana, S. Aylward, D. Metaxas, and R. Whitaker. Engineering and algorithm design for an image processing API: A technical report on ITK - The Insight Toolkit. In IOS Press Amsterdam J. Westwood, ed., editor, *Proceedings of Medicine Meets Virtual Reality*, 2002.
- [25] Jordi Inglada, Arthur Vincent, Marcela Arias, Benjamin Tardy, David Morin, and Isabel Rodes. Operational high resolution land cover map production at the country scale using satellite image time series. *Remote Sensing*, 9(1):95, 2017.
- [26] Rémi Cresson and Gabriel Hauteux. A generic framework for the development of geospatial processing pipelines on clusters. *IEEE Geoscience and Remote Sensing Letters*, 13(11):1706–1710, 2016.
- [27] Luis Ibez and Brad King. ITK (<http://www.aosabook.org/en/itk.html>). Accessed: 2016-05-24.
- [28] [www.equipex-geosud.fr](http://www.equipex-geosud.fr).
- [29] <http://www.theia-land.fr/en>.
- [30] OTBTensorflow remote module for orfeo toolbox. <https://github.com/remicres/otbtf>. Accessed: 2018-09-05.