SAGEO 2018

Spatial Analysis and GEOmatics

# Actes de l'atelier

# « Deep Learning pour la Télédétection »

Mardi 6 novembre 2018

## Organisateurs :

- Jonathan Weber, MCF, Université de Haute-Alsace, IRIMAS
- Camille Kurtz, MCF, Université Paris Descartes, LIPADE
- Germain Forestier, PU, Université de Haute-Alsace, IRIMAS
- Pierre Gançarski, PU, Université de Strasbourg, ICUBE

## Comité scientifique :

- Renaud Allioux, Directeur Technique, EarthCube
- Mauro Dalla Mura, MCF, Grenoble INP, GIPSA-lab
- Germain Forestier, PU, Université de Haute-Alsace, IRIMAS
- Pierre Gançarski, PU, Université de Strasbourg, ICUBE
- Cécile Gomez, CR IRD, UMR LISAH
- Dino Ienco, Chercheur, IRSTEA
- Jordi Inglada, Chercheur, CESBIO
- Camille Kurtz, MCF, Université Paris Descartes, LIPADE
- Sébastien Lefèvre, PU, Université de Bretagne-Sud, IRISA
- Charlotte Pelletier, Research Fellow, Monash University
- François Petitjean, Senior Research Fellow, Monash University
- Jonathan Weber, MCF, Université de Haute-Alsace, IRIMAS
- Cédric Wemmert, PU, Université de Strasbourg, ICUBE

# Table des matières

# A two-branch Deep Learning architecture for land cover classification of PAN and MS imagery

**R. Gaetano** [3], **D. Ienco** [2], **K. Ose** [4], **R. Cresson** [4]

1. *UMR-TETIS, CIRAD, Univ. of Montpellier, Montpellier, France*
   *raffaele.gaetano@cirad.fr*
2. *UMR-TETIS, IRSTEA, Univ. of Montpellier and LIRMM, Montpellier, France*
   *dino.ienco@irstea.fr*
3. *UMR-TETIS, IRSTEA, Univ. of Montpellier, Montpellier, France*
   *kenji.ose@irstea.fr*
4. *UMR-TETIS, IRSTEA, Univ. of Montpellier, Montpellier, France*
   *remi.cresson@irstea.fr*

*RÉSUMÉ.*

*ABSTRACT. The use of Very High Spatial Resolution (VHSR) imagery in remote sensing applications is nowadays a current practice whenever fine scale monitoring of earth surface is concerned. Modern VHSR sensors provide data at multiple spatial and spectral resolutions, most commonly as a couple of a higher resolution single-band panchromatic (PAN) and a coarser multispectral (MS) imagery. In the typical land cover classification workflow, the multi-resolution input is preprocessed to generate a single multispectral image at the highest resolution available by means of a pansharpening process. Recently, deep learning approaches have shown the advantages of avoiding data preprocessing by letting the machine learning algorithms automatically transform input data to best fit the classification task. Following this rationale, we here propose a new deep learning architecture to jointly use PAN and MS imagery for a direct classification without any prior image sharpening or resampling process. Experiments carried out on a real world scenario underline the quality of our method w.r.t. state of the art competitors.*

*MOTS-CLÉS :*

*KEYWORDS: Land Cover Mapping, Deep Learning, Satellite Image Time series, Very High Spatial Resolution.*

SAGEO'2018 - Montpellier, 6-9 novembre 2018

## 1. Introduction

The production of precise and timely Land Use/Land Cover (LULC) maps for monitoring human and physical environment is nowadays a matter of fact. Their use in a multitude of different domains, ranging from ecology, agriculture, mobility to health, risk monitoring and management policies is by now a consolidated practice (Bégué *et al*., 2018). The range of LULC maps applications has even increased since the large scale availability of Very High Spatial Resolution (VHSR) imagery, particularly suitable to capture fine-scale thematic information over territories (Georganos *et al*., 2018) and proving necessary in many real-world contexts (urban monitoring, road network updating, cadastral abuses, environmental police, etc.).The use of VHSR imagery have been raising specific challenges in remote sensing image analysis, mostly due to the fact that the majority of VHSR optical sensors provide data at different spectral and spatial resolution. More precisely, users generally dispose of a multispectral (MS) and a panchromatic (PAN) image acquired simultaneously and covering the same geographical area, with the spatial resolution of PAN images higher than that of MS images (Liu *et al*., 2018). Examples are the IKONOS, Quickbird and GeoEye sensors (4 m MS and 1 m PAN images), Pléiades (2 m MS and 0.5 m PAN images) and SPOT6/7 (6 m MS and 1.5 m PAN images). Common techniques to deal with multi-resolution information coming from the same sensor are related to the use of pansharpening (Fasbender *et al*., 2008 ; Colditz *et al*., 2006). The pansharpening process aims to "sharpen" a multispectral image using a panchromatic (single band) image. More generally, the common classification pipeline of multi-resolution VHSR images involves three main steps: 1) produce a single resolution dataset by means of a downsampling/upsampling or pansharpening procedure starting from the multi-resolution sources (Colditz *et al*., 2006), 2) extract spatio-spectral features in a hand-crafted fashion and 3) classify the resulting feature set by means of machine learning techniques (Regniers *et al*., 2016). A notable example is given in (Mura *et al*., 2010), where the authors propose to extract hand-crafted spatio-spectral features (attribute and morphological profiles) directly from the pansharpened image as a new representation of the input data. Successively, a Random Forest classifier is feeded with the extracted features to perform the final classification. With respect to the first step of the aforementioned pipeline, performances can be affected by artifacts or noise introduced upstream by the pansharpening process (Colditz *et al*., 2006). So far, only few techniques were proposed to directly manage multi-resolution classification avoiding the image fusion step in order to limit the impact of radiometric errors and spatial artifacts produced at this stage (Wemmert *et al*., 2009 ; Storvik *et al*., 2005 ; Liu *et al*., 2018). Concerning the two last steps, an emerging trend in remote sensing is to leverage Deep Learning to encompass feature extraction and classification in a unique optimization framework (Zhang, Du, 2016). Our work also follows this approach, focusing especially on the possibility to enclose the whole pipeline in a Deep Learning solution, including the pre-processing of the multi-resolution source (step 1 of the common pipeline). In particular, the overall contributions of this paper can be summarized as follows: i) provide a Deep Learning architecture for the supervised classification of MS and PAN sources from VHSR imagery which avoids any prior

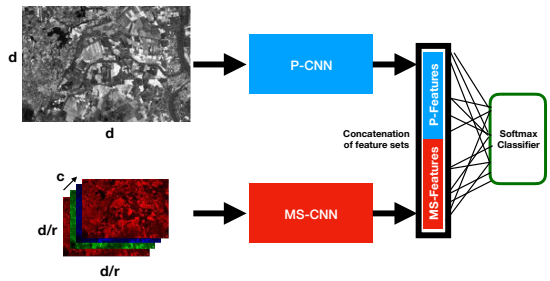
error prone pre-processing such as pansharpening; ii) leverage Convolutional Neural Networks (CNNs) to exploit spatial and spectral information at both available spatial resolutions and evaluate their ability as feature extractors and iii) deliver an end-to-end solution for Land Cover classification of VHSR images which is suitable for real world scenarios characterized by large areas as well as spatially sparse and limited reference data.

The rest of the article is organized as follows: Section 2 introduces the proposed deep learning architecture for the joint classification of PAN and MS imagery. The study sites and the associated data are presented in Section 3 while, the experimental setting and the evaluations are carried out and discussed in Section 4. Finally, Section 5 draws conclusions.

## 2. Method

In this Section we describe the proposed classification framework. Figure 1 supplies a global overview of our proposal.

*Figure 1. General Overview of $MultiResoLCC$. The model is based on a two-branch CNN architecture to deal with PAN and MS information sources at their native resolution.*




Our objective is to provide a per-pixel classification of the source dataset at the PAN spatial resolution. In order to exploit the spatio-spectral contextual information coming from both sources, in our workflow each pixel is represented by means of a pair of patches, extracted respectively from PAN and MS images and covering the same geographical area. To this purpose, supposing that the spatial resolution ratio between the PAN and MS image is equal to $r$, we set the size of the PAN patch equals to ($d$ x $d$), hence a patch size of $d/r$ x $d/r$ for the MS image. To ensure the best spatial correspondence between PAN and MS patches, $d$ is here chosen as a multiple of $r$, hence producing even-sized patches at PAN resolution since $r$ is typically a even number (e.g. $r = 4$ for most VHSR imagery). By convention, each pair of patches is associated to the pixel in position $(d/2, d/2)$ on the PAN patch, as well as the land cover class label associated to that pixel.

Based on this sampling strategy, our deep learning architecture is composed of two parallel branches that process PAN and MS patches through a dedicated Convolutio-

nal Neural Network (CNN) module. Each CNN module transforms the input patch, hence two feature sets are produced (one for the PAN and one for the MS source) that summarizes the spatial and spectral joint information. We name $P$-$CNN$ (resp. $MS$-$CNN$) the CNN working on the PAN image (resp. the MS image). Successively, the two feature sets are combined together by means of a simple concatenation and the whole set of features is directly used to perform the final classification via a SoftMax (Zhang, Du, 2016) classifier. The model is trained end-to-end from scratch.

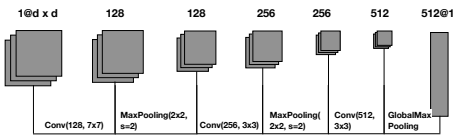
The $P$-$CNN$ branch takes as input a tensor of size $d$ x $d$ x 1 (since in the general case we only dispose of a single panchromatic band) where the parameter $d$ is used to define the patch size. Conversely, the branch associated to the $MS$-$CNN$ takes as input a tensor of size ($d/r$ x $d/r$ x $c$) where $c$ is the number of channels contained in the MS image.

Coherently to the model training, at inference time, the PAN grid is scanned, and for each pixel the PAN and MS patches are extracted and processed to provide the final class for that pixel, eventually producing a full PAN resolution land cover map. We remind that, conversely to (Liu *et al.*, 2018) in which the MS image was upsampled inducing bias related to interpolation as well as increasing the amount of data to manage, $MultiResoLCC$ directly deals with the different spatial resolutions avoiding additional interpolation biases and limiting the quantity of data to process. The prediction of $MultiResoLCC$ is performed at the same resolution of the PAN image. This means that our approach can be employed to produce LULC maps at the finest spatial resolution among those of the input sources. In the rest of this section we describe the Convolution Neural Networks ($MS$-$CNN$ and $P$-$CNN$) that are the core components of our framework. We also describe the training strategy we adopt to learn the parameters of our architecture.

### *2.1. CNN architectures for the panchromatic and the multispectral information*

Both branches of our model are inspired from the VGG model (Simonyan, Zisserman, 2014), one of the most well-known network architectures usually adopted to tackle with standard computer vision tasks. More in detail, for both branches we constantly increase the number of filters along the network until we reach a reasonable size of the feature maps.

*Figure 2. $P$-$CNN$: Dedicated CNN Structure to manage PAN imagery.*

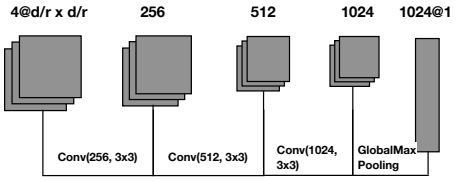



Considering the $P$-$CNN$ module (see Figure 2), we perform a max pooling operation after each convolution to reduce the number of features to process and to force the network to focus on the most important part of the input signal.All max pooling

operations are performed with a pooling size (window on which the max pooling is applied) of 2×2 with a stride equals to 2.

The first convolution has a kernel of 7×7 and it produces 128 feature maps. The second and the third convolutions have a kernel of 3×3 and they produce respectively, 256 and 512 feature maps. At the end of the process, a global max pooling is applied in order to extract 512 features. The global max pooling extract one feature for each feature maps obtained after the last convolution.

*Figure 3. MS-CNN: Dedicated CNN Structure to manage MS imagery.*




For the $MS\text{-}CNN$ module (Figure 3), no max pooling operation between two successive convolution stages is performed. This is done to preserve as much as possible all the spectral information along the processing flow. For the same reason, in each convolutional layer (three total layers as for $P\text{-}CNN$) the size of the kernel is limited to 3×3. Moreover, each of these layers produces respectively 256, 512 and 1024 feature maps (doubled w.r.t. the corresponding $P\text{-}CNN$ layers) to deal with the richness of the spectral information with the aim to better exploit the correlations among the original MS bands. The final set of features summarizing the MS information is derived similarly to the $P\text{-}CNN$ model. Also in this case, we apply a global max pooling extracting 1024 features (one feature for each feature maps obtained after the last convolution).

The features extracted by each branch of the architecture are successively merged by concatenation. Such set of concatenated features, 512 (resp. 1024) from the PAN branch (resp. MS branch) supplies a total of 1536 features that are directly fully connected to the Softmax classifier to perform the final classification. The SoftMax layer (Zhang, Du, 2016) produces a kind of probability distribution over the class labels. The model weights are learned by back-propagation.

In both branches, each convolution is associated with a linear filter, followed by a Rectifier Linear Unit (ReLU) activation function (Nair, Hinton, 2010) to induce non-linearity and a batch normalization step (Ioffe, Szegedy, 2015). The ReLU activation function is defined on the positive part of the linear transformation. The choice of ReLU nonlinearities is motivated by two factors: i) the good convergence properties it guarantees and ii) the low computational complexity it provides (Nair, Hinton, 2010). Furthermore, batch normalization (Ioffe, Szegedy, 2015) accelerates deep network training convergence by reducing the internal covariate shift.

### *2.2. Network Training Strategy*

Due to the network architecture peculiarity (two branches, multi-scale input, different number of channels for each branch) we learn the network weights end-to-end from scratch since we cannot reuse any existing available pre-trained architecture. The cost function associated to our model is :

$$LOSS = L([PAN_{feat}, MS_{feat}], W, b) \quad (1)$$

where

$$L(feat, W, b) = L(Y, SoftMax(feat \cdot W + b))$$

with $Y$ being the true value of the class variable. The cost function is modeled through categorical cross entropy, a typical choice for multi-class supervised classification tasks (Ienco *et al.*, 2017).

Although the number of parameters of our architecture is not prohibitive, training of such models might be difficult and the final model can suffer by overfitting (Dahl *et al.*, 2013). To avoid such phenomena, following common practice for the training of deep learning architecture we leverage dropout (Dahl *et al.*, 2013) and data augmentation (Perez, Wang, 2017).

Dropout has been proposed to avoid co-adaptation of neurons during training (Dahl *et al.*, 2013). Dropout randomly "turning off" a given percentage of neurons (dropout rate hyperparameter) and their connections, corresponds to train a different, less correlated, model at every epoch. At inference time the neuron contribution is weighted by the dropout rate. In our architecture we decide to apply dropout (rate equals to 0.4) on the feature sets extracted by the two branches of $MultiResoLCC$ just before the concatenation operation. This strategy will avoid to extract co-adapted features among the set of features successively employed to make the final decision.

Data augmentation (Perez, Wang, 2017) is a common strategy to further increase the size of the training set and achieve higher model generalization. It consists in creating new synthetic training examples from those already available, by applying label preserving (random) transformations. In our case the (random) transformations are sampled from standard data augmentation techniques (90 degree rotation, vertical/horizontal flips and transpose). For each example, each technique is simultaneously performed on both the PAN and the corresponding MS patch. On average, the final training set has a size around three times the original training set.

## 3. GARD Data

The SPOT6 image, acquired on March 12th 2016 consists of a 1.5 m Panchromatic band and 4 Multi Spectral bands (Blue, Green, Red and Near Infrared) at 6 m resolution *Top of Atmosphere* reflectance. The Panchromatic image has a size of 24 110 $\times$ 33 740 while the Multi Spectral image has a size of 6 028 $\times$ 8 435. The field database, constituting the ground truth, was built from various sources: (i) the *Registre*

*parcellaire graphique* (RPG) [1] reference data of 2016 and (ii) photo interpretation of the VHSR image conducted by an expert, with knowledge of the territory, for distinguishing between natural and urban areas.

Ground truth comes in GIS vector file format containing a collection of polygons each attributed with a unique land cover class label. To ensure a precise spatial matching with image data, all geometries have been suitably corrected by hand using the corresponding VHSR image as a reference. Successively, the GIS vector file containing the polygon information has been converted in raster format at the spatial resolution of the PAN image (1.5m in our case). The final ground truth are constituted of 400 970 pixels distributed over 8 classes for the *Gard* benchmark (Table **??**). We remind that the ground truth was collected over large areas. Due to the practical constraints associated to such missions (time consumption, costs and human effort), reference data are spatially sparse, noisy and limited with respect to the study area on which land cover classification is performed.

TABLE *1. Per Class ground truth statistics of the Gard Dataset .*

| **Class** | Label | **# Polygons** | **# Pixels** |
|---|---|---|---|
| 1 | *Cereal Crops* | 167 | 50100 |
| 2 | *Other Crops* | 167 | 50098 |
| 3 | *Tree Crops* | 167 | 50027 |
| 4 | *Meadows* | 167 | 49997 |
| 5 | *Vineyard* | 167 | 50100 |
| 6 | *Forest* | 172 | 50273 |
| 7 | *Urban areas* | 222 | 50275 |
| 8 | *Water Surfaces* | 167 | 50100 |

## 4. Experiments

In this section, we present and discuss the experimental results obtained on the study site introduced in Section 3.

### *4.1. Competitors*

With the purpose to compare our approach ($MultiResoLCC$) to techniques tailored for the classification of VHSR images, different competitors are involved in the analysis. Firstly, we compare $MultiResoLCC$ with other deep learning approaches and then, similarly to what is proposed in (Ienco *et al.*, 2017), we investigate the possibility to use the deep learning methods as feature extractor to obtain new data representation for the classification task. In this context, we compare the deep learning features to spatio-spectral representations usually employed for the land cover classification of VHSR images (Mura *et al.*, 2010; Volpi, Tuia, 2017). Regarding deep

1. RPG is part of the European Land Parcel Identification System (LPIS), provided by the French Agency for services and payment

learning competitors, the first method we consider is a CNN classifier that has the same architecture structure of the $P\text{-}CNN$ module with 256, 512 and 1024 as number of filters for each layer respectively. The number of filters is augmented w.r.t. the $P\text{-}CNN$ architecture due to the amount of radiometric information (four bands) as input of such CNN. We refer to this competitor as $CNN_{PS}$. The second deep learning competitor we consider is the model proposed in (Liu *et al.*, 2018), named $DMIL$, since it can be trained from sparsely annotated data to produce classification at pixel-level from PAN and MS images. Regarding the feature extraction analysis, we compare the features extracted by $MultiResoLCC$ with respect to the features extracted by the other deep learning approaches as well as hand-crafted features obtained by common spatio-spectral methods. To this end, we feed a random forest classifier (with a number of trees equals to 400) with the features extracted by each of the different deep learning methods. To refer to this setting, we use the notation $RF(\cdot)$. For instance, $RF$(MultiResoLCC) indicates the random forest trained on the representation (features) learned by $MultiResoLCC$. With the objective to supply a more complete evaluation scenario, we consider two other competitors based on spatio-spectral features (Mura *et al.*, 2010). The first one involves a random forest classifier trained on the data patches extracted from the pansharpened image. The final feature set is composed of 4 096 features ($32 \times 32 \times 4$). We refer to this method as $RF(PATCH)$. For the second one, similarly to what is proposed in (Volpi, Tuia, 2017), we extract a spatio-spectral hand-crafted features from the pansharpened image. More precisely, for each raw band of the pansharpened image (Red, Blue, Green and NIR) we extract four mathematical morphology operators (opening, closing, dilation and erosion) as well as a texture statistic (entropy). Each filter is operated in three window size (7, 11, 15 pixels). The final feature set is composed of 60 spatio-spectral features. We named this representation $MRSSF$ (Multi-Resolution Spatial Spectral Features) and the related classification method $RF(MRSSF)$. The Pansharpened image, derived from the combination of the panchromatic and multi-spectral sources, is obtained using the Bayesian Data Fusion technique (Fasbender *et al.*, 2008) implemented by the Orfeo ToolBox (G. *et al.*, 2017).

### *4.2. Experimental Setting*

All the deep learning methods are implemented using the Python Tensorflow library. We adopt the Adam optimizer (Kingma, Ba, 2014) (Adaptive Moment Estimation) that is commonly employed in the parameter optimization of both CNN and Recurrent Neural Networks. Adam is an optimization algorithm, based on adaptive estimates of lower-order moments, that can be used instead of the classical stochastic gradient descent procedure to iteratively update network weights based on training data. We set the learning rate equal to $2 \cdot 10^{-4}$. The training process is conducted over 250 epochs with a batch size equals to 64. The model that reaches the lowest value of the cost function (at training time) is used in the test phase. The dataset consists in pairs of patches (PAN, MS) associated to land cover class labels. We set the value of $d$, the PAN patch size, to 32. The patches sizes are respectively (32 x 32 x 1) and (8

x 8 x 4) for the PAN and MS images, as the pixel spacing ratio between SPOT6 PAN and MS is 4, and because the MS image has 4 spectral bands. Coherently to what explained in section 2, each pair of patches is associated to the pixel in position (16,16) of the PAN patch and its associated land cover class label. Prior to patch extraction, each spectral band is normalized in the interval $[0, 1]$. Considering the $CNN_{PS}$ and $RF(PATCH)$ approaches, these methods take as input a patch of size ($32 \times 32 \times 4$) coming from the pansharpened image and, also in this case, the label information refers to the pixel in position (16,16).

We divide the dataset into two parts, one for learning and the other one for testing the performance of the supervised classification methods. We used 30% of the objects for the training phase while the remaining 70% are employed for the test phase, in order to force a relative parsimony in the training stage with respect to the available reference data while ensuring a more robust validation. We impose that pixels of the same object belong exclusively to the training or to the test set to avoid spatial bias in the evaluation procedure (Inglada *et al.*, 2017).

Table 2 reports the training time of each deep learning method on a workstation with an Intel (R) Xeon (R) CPU E5-2667 v4@3.20Ghz with 256 GB of RAM and TITAN X GPU. $CNN_{PS}$ is the approach that demands more time. $DMIL$ and $MultiResoLCC$ consume very similar training time. The difference among the different methods is due to the fact that $CNN_{PS}$, for a fixed geographical area, needs to manage more information as input.

| Dataset | $CNN_{PS}$ | $DMIL$ | $MultiResoLCC$ |
|---|---|---|---|
| *Gard* | 9h30m | 7h20m | 7h30m |

TABLE 2. *Training time of the different deep learning approaches on the Gard study site.*

The assessment of the classification performances is done considering global precision (*Accuracy*), *F-Measure* (Ienco *et al.*, 2017) and *Kappa* measures. It is known that, depending on the split of the data, the performances of the different methods may vary as simpler or more difficult examples are involved in the training or test set. To alleviate this issue, for each dataset and for each evaluation metric, we report results averaged over ten different random splits performed with the strategy previously presented.

### *4.3. General Classification Results*

Table 3 summarizes the results obtained by the different methods on the *Gard* study site. The upper part of the table shows the performances of the deep learning competing methods ($CNN_{PS}$, $DMIL$ and $MultiResoLCC$) while the lower part summarizes the results of the random forest classifier trained on the features learned by the different deep learning architectures as well as the spatio-spectral representation obtained from the Pansharpened image.

Considering the results of the main competing approaches, we can observe that $MultiResoLCC$ always outperforms the other approaches for all the three evaluation metrics. Systematically, $MultiResoLCC$ obtains the best performances followed by $CNN_{PS}$ and $DMIL$ respectively. We can also underline that the difference between the best and the second best result is always higher than four points for the *Accuracy* measure. Similar behavior is exhibited considering the other evaluation metrics. These experimental findings support our intuition that, when using a CNN-based deep learning approach for land cover classification, letting the architecture exploits sources at their native resolution (considering both spatial and spectral information) is more adequate than performing a prior pansharpening.

Regarding the use of the different deep learning approaches to extract features that are successively injected as input to standard machine learning method, we can note that this practice does not degrade the classification performances while, most of the time, it results in an improvement of the classification performances (w.r.t. the deep learning classification counterparts) of one or two points considering the whole set of evaluation metrics. We can also note that the $RF$ classifier combined with a patch based input ($RF(PATCH)$) and with multi-resolution spatio-spectral features ($RF(MRSSF)$) supplies results that are competitive w.r.t. $CNN_{PS}$ and $DMIL$ on the *Gard* study site.

TABLE *3. Accuracy, F-Measure and Kappa results achieved on the GARD study site with different competing methods. Best results are reported in bold.*

| | *Accuracy* | *F-Measure* | *Kappa* |
|---|---|---|---|
| $CNN_{PS}$ | 66.14 ± 0.78 | 65.80± 0.77 | 0.6131 ± 0.0089 |
| $DMIL$ | 61.96 ± 1.00 | 61.76 ± 1.01 | 0.5652 ± 0.0115 |
| $MultiResoLCC$ | **70.48** ± 0.55 | **70.19** ± 0.67 | **0.6627** ± 0.0063 |
| RF(PATCH) | 69.93 ± 0.76 | 69.55 ± 0.77 | 0.6564 ± 0.0087 |
| $RF(MRSSF)$ | 68.30 ± 0.33 | 68.18 ± 00.51 | 0.6377 ± 0.0038 |
| RF($CNN_{PS}$) | 68.04 ± 0.82 | 67.72 ± 0.84 | 0.6348 ± 0.0093 |
| RF(DMIL) | 64.79 ± 0.73 | 64.43 ± 0.82 | 0.5976 ± 0.0084 |
| RF($MultiResoLCC$) | **71.98** ± 0.58 | **71.73** ± 0.54 | **0.6797** ± 0.0066 |

### *4.4. Per Class Classification Results*

Table 4 depicts the per class F-Measure results for the *Gard* study site. We differentiate between the main competing methods ($CNN_{PS}$, $DMIL$ and $MultiResoLCC$) and experiments with random forest classifier learned on hand-crafted or deep learning features.

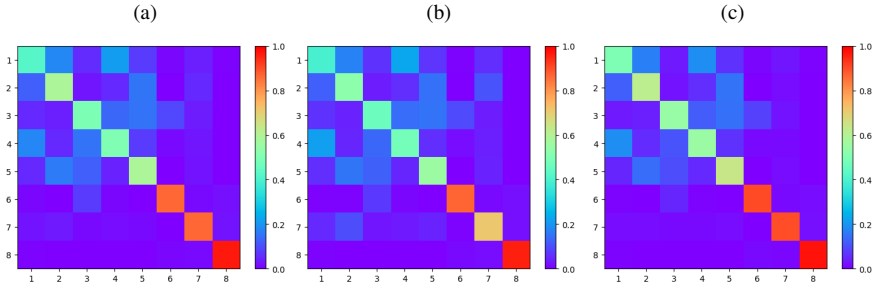
$MultiResoLCC$ obtains better or very similar per class *F-Measure* w.r.t. the others competing approaches. The analysis shows improvement for certain classes: (1) and (3) (*Cereal Crops* and *Tree Crops*) with an average gain of 7 points of *Accuracy*.

TABLE *4. Per Class F-Measure results achieved on the Gard study site with different competing methods. Best results are reported in bold.*

| **Method** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| $CNN_{PS}$ | 44.67 | 57.04 | 50.63 | 51.0 | 57.14 | 86.75 | 83.37 | 95.59 |
| $DMIL$ | 40.78 | 51.1 | 47.87 | 47.63 | 54.85 | 86.16 | 70.01 | 95.52 |
| $MultiResoLCC$ | **51.74** | **60.2** | **58.51** | **55.36** | **61.15** | **89.07** | **88.94** | **96.3** |
| RF(PATCH) | 48.28 | 62.56 | 56.54 | 56.17 | 62.13 | 87.71 | 85.25 | **97.46** |
| RF($MRSSF$) | 47.65 | 58.77 | 54.82 | 53.27 | 61.56 | 87.97 | 83.98 | 97.39 |
| RF($CNN_{PS}$) | 45.86 | 60.07 | 54.9 | 53.56 | 58.33 | 88.33 | 84.47 | 96.06 |
| RF($DMIL$) | 41.17 | 54.89 | 52.02 | 51.83 | 59.14 | 86.97 | 73.52 | 95.76 |
| RF($MultiResoLCC$) | **52.53** | **64.13** | **60.49** | **58.14** | **63.85** | **89.1** | **88.95** | 96.46 |

We can also note that the random forest approach coupled with the features learned by the different methods provides systematic improvement w.r.t. almost all the land cover classes compared to the pure deep learning classification approaches.

*Figure 4. Confusion matrices of the deep learning approaches on the Gard dataset ( $CNN_{PS}$ (a), $DMIL$ (b) and $MultiResoLCC$ (c) ).*




To further advance the understanding of our method, we report in Figure 4 the confusion matrices associated to the $CNN_{PS}$, $DMIL$ and $MultiResoLCC$ methods respectively.

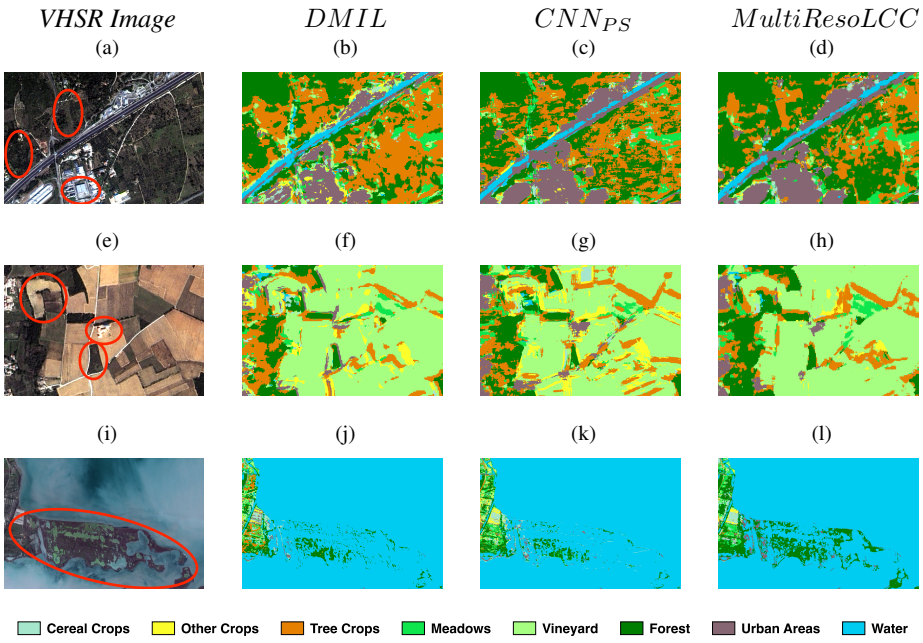
Figure 4a, Figure 4b and Figure 4c represent the confusion matrices of $CNN_{PS}$, $DMIL$ and $MultiResoLCC$, respectively, on the *Gard* study site. Here, we can observe a slightly more suitable behavior exhibited by $MultiResoLCC$: i) a slightly darker diagonal on both strong and weaker classes and ii) a generally less intense "noise" outside the diagonal compared to the competitors.

### *4.5. Qualitative Inspection of Land Cover Map*

In Figure 5 we report some representative map classification details considering the $DMIL$, $CNN_{PS}$ and $MultiResoLCC$, respectively.

The first example (Figures 5a, 5b, 5c and 5d) depicts an area mainly characterized by tree crops, urban area and forest. Here, we highlight three representative zones

*Figure 5. Qualitative investigation of Land Cover Map details produced on the Gard study site by $DMIL$, $CNN_{PS}$ and $MultiResoLCC$ on three different zones (from the top to the bottom): i) mixed area (tree crops, urban settlement and forest); ii) rural area and iii) wetland area.*




on which classification differences are more evident. From the top to the bottom, the first two circles point out a field characterized by tree crops and forest zone respectively. On these two zones, we can observe that both $DMIL$ and $CNN_{PS}$ present confusion between these two classes and do not preserve the geometry of the scene. Conversely, we can observe that $MultiResoLCC$ supplies a better (and more homogeneous) characterization of the two zones reducing confusion between the two classes and more correctly detecting parcel borders. The first zone highlighted in this example also involves an urban area. We can note that, $MultiResoLCC$ provides a more homogeneous classification of this zone with respect to the other two approaches that make some confusion between *urban areas* and *other crops* classes. The second example (Figures 5e, 5f, 5g and 5h) represents a rural area mainly characterized by different crop types. Also in this case we highlight three zones to pinpoint the differences among the deep learning approaches. From the top to the bottom, the first focus is on a vineyard field. $DMIL$ and $CNN_{PS}$ have some issues to correctly assign the *vineyard* class to the entire field making confusion among *Tree crops* and *Other Crops*. This is not the case for $MultiResoLCC$ that provides a more correct delimitation. The other two zones pointed out in this example involve an urban area and a forest field. We can observe that, also in this case, $MultiResoLCC$ shows better performance on both

*Urban Areas* and *Forest* classes than the other approaches. The third example (Figures 5i, 5j, 5k and 5l) involves a wetland area. Here, we can clearly observe that the first two approaches ($DMIL$ and $CNN_{PS}$) have serious issues to recognize non water area and they tend to overestimate the water class. Conversely, $MultiResoLCC$ achieves better performance to discriminate between *water* and other classes. On this study area, $MultiResoLCC$ seems to be more effective on some particular classes like *Tree Crops*, *Forest* and *Urban areas*. These results are consistent with those reported in Table 4. Considering a more fine visual inspection of the land cover maps, we can observe that the land cover map produced by $CNN_{PS}$ shows some horizontal strip artifact evident on the *Tree Crops* class (orange color). $CNN_{PS}$ exhibits similar artifacts also on the second example. This behavior is not shared by the other approaches, which probably mean that such artifacts are due to some slight radiometric inconsistency of the pansharpened source.

## 5. Conclusion

In this paper, a novel Deep Learning architecture to leverage PAN and MS imagery for land cover classification has been proposed. The approach, named $MultiResoLCC$, exploits multi-spectral and panchromatic information at their native resolutions. The architecture is composed of two branches, one for the PAN and one for the MS source. The final land cover classification is achieved by concatenating the features extracted by each branch. The framework is learned end-to-end from scratch. The evaluation on A real-world study site has shown that $MultiResoLCC$ achieves better quantitative and qualitative results than recent classification methods for optical VHSR images.

## 6. Acknowledgements

This work was supported by the French National Research Agency under the Investments for the Future Program, referred as ANR-16-CONV-0004 (DigitAg) and the GEOSUD project with reference ANR-10-EQPX-20.

## Bibliographie

Bégué A., Arvor D., Bellón B., Betbeder J., Abelleyra D. de, Ferraz R. P. D. *et al.* (2018). Remote sensing and cropping practices: A review. *Remote Sensing*, vol. 10, n° 1, p. 99.

Colditz R. R., Wehrmann T., Bachmann M., Steinnocher K., Schmidt M., Strunz G. *et al.* (2006). Influence of image fusion approaches on classification accuracy: a case study. *International Journal of Remote Sensing*, vol. 27, n° 15, p. 3311-3335.

Dahl G. E., Sainath T. N., Hinton G. E. (2013). Improving deep neural networks for LVCSR using rectified linear units and dropout. In *Icassp*, p. 8609–8613.

Fasbender D., Radoux J., Bogaert P. (2008). Bayesian data fusion for adaptable image pansharpening. *IEEE Transactions on Geoscience and Remote Sensing*, vol. 46, n° 6, p. 1847-1857.

G. M., M. J., P. V., I. J., S. M., C. R. (2017, 29 Jun). Orfeo toolbox: open source processing of remote sensing images. *Open Geospatial Data, Software and Standards*, vol. 2, n° 1, p. 15.

Georganos S., Grippa T., Vanhuysse S., Lennert M., Shimoni M., Wolff E. (2018). Very high resolution object-based land use-land cover urban classification using extreme gradient boosting. *IEEE Geosci. Remote Sensing Lett.*, vol. 15, n° 4, p. 607–611.

Ienco D., Gaetano R., Dupaquier C., Maurel P. (2017). Land cover classification via multitemporal spatial data by deep recurrent neural networks. *IEEE GRSL*, vol. 14, n° 10, p. 1685–1689.

Inglada J., Vincent A., Arias M., Tardy B., Morin D., Rodes I. (2017). Operational high resolution land cover map production at the country scale using satellite image time series. *Remote Sensing*, vol. 9, n° 1, p. 95.

Ioffe S., Szegedy C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Icml*, p. 448–456.

Kingma D. P., Ba J. (2014). Adam: A method for stochastic optimization. *CoRR*, vol. abs/1412.6980. Consulté sur http://arxiv.org/abs/1412.6980

Liu X., Jiao L., Zhao J., Zhao J., Zhang D., Liu F. *et al.* (2018). Deep multiple instance learning-based spatial-spectral classification for PAN and MS imagery. *IEEE TGRS*, vol. 56, n° 1, p. 461–473.

Mura M. D., Benediktsson J. A., Waske B., Bruzzone L. (2010). Morphological attribute profiles for the analysis of very high resolution images. *IEEE Trans. Geoscience and Remote Sensing*, vol. 48, n° 10, p. 3747–3762.

Nair V., Hinton G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Icml10*, p. 807–814.

Perez L., Wang J. (2017). The effectiveness of data augmentation in image classification using deep learning. *CoRR*, vol. abs/1712.04621. Consulté sur http://arxiv.org/abs/1712.04621

Regniers O., Bombrun L., Lafon V., Germain C. (2016). Supervised classification of very high resolution optical images using wavelet-based textural features. *IEEE Trans. Geoscience and Remote Sensing*, vol. 54, n° 6, p. 3722–3735.

Simonyan K., Zisserman A. (2014). Very deep convolutional networks for large-scale image recognition. *CoRR*, vol. abs/1409.1556. Consulté sur http://arxiv.org/abs/1409.1556

Storvik G., Fjørtoft R., Solberg A. H. S. (2005). A bayesian approach to classification of multiresolution remote sensing data. *IEEE Trans. Geoscience and Remote Sensing*, vol. 43, n° 3, p. 539–547.

Volpi M., Tuia D. (2017). Dense semantic labeling of subdecimeter resolution images with convolutional neural networks. *IEEE Trans. Geoscience and Remote Sensing*, vol. 55, n° 2, p. 881–893.

Wemmert C., Puissant A., Forestier G., Gançarski P. (2009). Multiresolution remote sensing image clustering. *IEEE Geosci. Remote Sensing Lett.*, vol. 6, n° 3, p. 533–537.

Zhang L., Du B. (2016). Deep learning for remote sensing data: A technical tutorial on the state of the art. *IEEE Geoscience and Remote Sensing Magazine*, vol. 4, p. 22–40.

# $M^3Fusion$: A Deep Learning Architecture for Multi-{Scale/Modal/Temporal} satellite data fusion

**P. Benedetti** [1], **D. Ienco** [2], **R. Gaetano** [3], **K. Ose** [4], **R. Pensa** [5], **S. Dupuy** [6]

1. *UMR-TETIS, IRSTEA, Univ. of Montpellier, Montpellier, France*
   *paola.benedetti@irstea.fr*
2. *UMR-TETIS, IRSTEA, Univ. of Montpellier and LIRMM, Montpellier, France*
   *dino.ienco@irstea.fr*
3. *UMR-TETIS, CIRAD, Univ. of Montpellier, Montpellier, France*
   *raffaele.gaetano@cirad.fr*
4. *UMR-TETIS, IRSTEA, Univ. of Montpellier, Montpellier, France*
   *kenji.ose@irstea.fr*
5. *Univ. of Turin, Computer Science Department,Turin, Italy*
   *ruggero.pensa@unito.it*
6. *UMR-TETIS, CIRAD, Univ. of Montpellier, Saint-Pierre and Montpellier, France*
   *stephane.dupuy@cirad.fr*

*RÉSUMÉ.*

*ABSTRACT. Modern Earth Observation systems provide remote sensing data at different temporal and spatial resolutions. Among all the available spatial mission, today the Sentinel-2 program supplies high temporal (every 5 days) and high spatial resolution (10m) images that can be useful to monitor land cover dynamics. On the other hand, Very High Spatial Resolution (VHSR) imagery is still essential to figure out land cover mapping characterized by fine spatial patterns. Understanding how to jointly leverage these complementary sources when dealing with land cover mapping is challenging. With the aim of providing land cover mapping through the fusion of multi-temporal High Spatial Resolution and VHSR satellite images, we propose a suitable end-to-end Deep Learning framework, namely $M^3Fusion$. Experiments carried out on the Reunion Island study area confirm the quality of our proposal considering both quantitative and qualitative aspects.*

*MOTS-CLÉS :*

*KEYWORDS: Land Cover Mapping, Deep Learning, Satellite Image Time series, Very High Spatial Resolution.*

SAGEO'2018 - Montpellier, 6-9 novembre 2018

## 1. Introduction

Modern Earth Observation (EO) systems produce huge volumes of data every day. Earth Observation programs (e.g., Copernicus) supply image acquisition at high spatial resolution (10m) with high temporal revisit period (every 5 days). This information can be organized into time series of high-resolution satellite imagery (SITS) that are particular useful for area monitoring over time. Other Earth Observation programs are able to provide image information at finer spatial resolution (between 0.5 to 2m) but with a low revisiting frequency. An example of EO program that supplies this kind of information is the SPOT6/7 mission that produces images with a spatial resolution of 1.5m. Such kind of images supply Very High Spatial Resolution (VHRS) information and they are extremely useful to characterize land use or land cover by means of their spatial structure (Maggiori *et al*., 2017). In the context of land use and land cover classification, employing high spatial resolution (HSR) time series, instead of a single image of the same resolution, can be useful to distinguish land use classes according to their temporal profile or evolution (Abade *et al*., 2015). On the other hand, the use of fine spatial information (VHSR images) helps to differentiate other kind of classes that need spatial context information at a finer scale (Schmitt, Zhu, 2016). Due to the diverse, although complementary, information carried out by each of these different Earth Observation sources, how to intelligently combine satellite image time series and VHSR images via a dedicated fusion process, for a particular task at hand, constitutes an important challenge in the field of remote sensing (Karpatne *et al*., 2016 ; Schmitt, Zhu, 2016).

Recently, the deep learning revolution (Zhang, Du, 2016) has shown that neural network models are well adapted tools for automatically managing and classifying remote sensing data. The main characteristic of this type of model is the ability to simultaneously extract features optimized for image classification as well as for the associated classifier. This advantage is fundamental in a data fusion process such as the one involving high resolution time series (i. e. Sentinel-2) and VHSR data (i. e. Spot6/7). CNN are well suited to model the spatial autocorrelation available in an image. RNN networks, instead, are specifically tailored to manage time dependencies (Ienco *et al*., 2017 ; Minh *et al*., 2018a ; Geng *et al*., 2018) from multidimensional time series. In this article, we propose to leverage both CNN and RNN to fuse together HSR time series of Sentinel-2 images and a single VHSR scene (SPOT6/7), covering the same study area, with the goal of performing land use mapping.

The method we propose, named $M^3Fusion$ (Multi-Scale/Modal/Temporal Fusion), is a deep learning architecture that integrates both a CNN module (to integrate VHSR information) and an RNN module (to manage HSR time series information) in an end-to-end learning process. Each information source is integrated through its dedicated module and the extracted descriptors are successively concatenated to per-

form the final classification. All the non-linear transformations are learned together resulting in an architecture that is able to manage, simultaneously, multi-temporal and multi-scale information, thus enabling the extraction of complementary and diversely features for land use mapping.

To validate our approach, we conduct experiments on a data set describing the *Reunion Island* site. This site is a French Overseas Department located in the Indian Ocean (east of Madagascar) and it will be described in Section 2. The rest of the article is organized as follows: Section 3 introduces the $M^3Fusion$ Deep Learning Architecture for the multi-source classification process. The experimental setting and the findings are discussed in Section 4. Finally, conclusions are drawn in Section 5.

## 2. Data

The study was carried out on the Reunion Island, a French overseas department located in the Indian Ocean. The dataset consists of a time series of 34 Sentinel-2 (S2) images acquired between April 2016 and May 2017, as well as a very high spatial resolution (VHSR) SPOT6/7 image acquired in April 2016 and covering the whole island. The S2 images used are those provided at level 2A by the THEIA pole [1], where the bands at 20m resolution were resampled at 10m via bicubic interpolation. A pre-processing was performed to fill cloudy observations through a linear multi-temporal interpolation over each band (cfr. *Temporal Gapfilling*, (Inglada *et al*., 2017)), and six radiometric indices were calculated for each date: NDVI, NDWI, brightness index (BI), NDVI and NDWI of infrared means (MNDVI and MNDWI), and vegetation index Red-Edge (RNDVI) (Inglada *et al*., 2017 ; Lebourgeois *et al*., 2017). A total of 16 variables (10 surface reflectances plus 6 indices) are considered for each pixel of each image in the time series.

The SPOT6/7 image, acquired on April 6th 2016 and originally consisting of a 1.5 m panchromatic band and 4 multispectral bands (blue, green, red and near infrared) at 6 m resolution, was pansharpened to produce a single multispectral image at 1.5 m resolution and then resampled at 2 m via bicubic interpolation because of the network architecture learning requirements [2]. Its final size is 33 280 $\times$ 29 565 pixels on bands (4 *Top of Atmosphere* reflectance plus the NDVI). This image was also used as a reference to realign the different images in the time series by searching and mapping anchor points, in order to improve the spatial coherence between the different sources.

The field database was built from various sources: (i) the *Registre parcellaire graphique* (RPG) reference data of 2014, (ii) GPS records from June 2017 and (iii) photo interpretation of the VHSR image conducted by an expert, with knowledge of the territory, for distinguishing between natural and urban spaces. All polygon contours have

1. Data are available via http://theia.cnes.fr, preprocessed in surface reflectance via the *MACCS-ATCOR Joint Algorithm* (Hagolle *et al*., 2015) developed by the National Centre for Space Studies (CNES).

2. This was done to ensure a direct and non-overlapping correspondence between the time series pixels (10 m) and a block of VHSR pixels (5 $\times$ 5).

been resumed using the VHSR image as a reference. The final dataset is composed of a total of 322 748 pixels (2 656 objects) distributed over 13 classes, as indicated in Table 1.

TABLE *1. Characteristics of the Reunion Dataset*

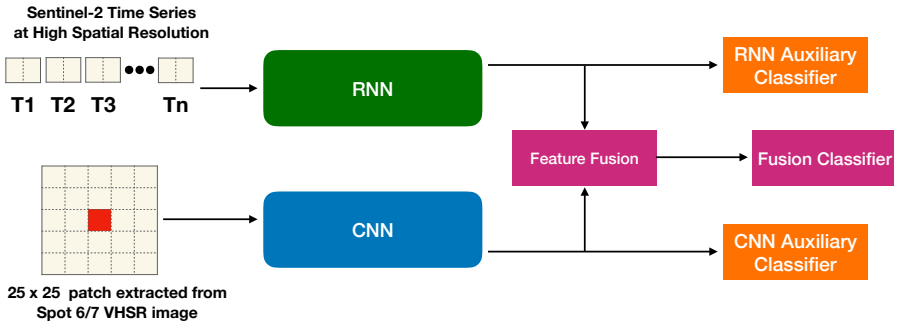
| Class | Label | # Objects | # Pixels |
|---|---|---|---|
| 1 | *Crop Cultivations* | 380 | 12090 |
| 2 | *Sugar cane* | 496 | 84136 |
| 3 | *Orchards* | 299 | 15477 |
| 4 | *Forest plantations* | 67 | 9783 |
| 5 | *Meadow* | 257 | 50596 |
| 6 | *Forest* | 292 | 55108 |
| 7 | *Shrubby savannah* | 371 | 20287 |
| 8 | *Herbaceous savannah* | 78 | 5978 |
| 9 | *Bare rocks* | 107 | 18659 |
| 10 | *Urbanized areas* | 125 | 36178 |
| 11 | *Greenhouse crops* | 50 | 1877 |
| 12 | *Water Surfaces* | 96 | 7349 |
| 13 | *Shadows* | 38 | 5230 |

## 3. Contributions

### *3.1.* $M^3Fusion$ *model overview*

Figure 1 visually describes the Multi-Scale/Modal/Temporal Fusion ($M^3Fusion$) approach proposed in this work. First of all, we define the input data for our deep learning model. $M^3Fusion$ takes as input a dataset $\{(x_i, y_i)\}_{i=1}^{M}$ where each example is associated with a class value $y_i \in 1, ..., C$. An example $x_i$ is defined as a pair $x_i = (ts_i, patch_i)$ such that $ts_i$ is the (multidimensional) time series of a Sentinel-2 pixel (10 m resolution) and $patch_i$ is a subset of the image Spot6/7 (at 2 m resolution) centered around the corresponding Sentinel-2 pixel. Note here that every example is purposely built to encompass two different acquisition modes at two different scales for a given sample area: a pixel-based spectral dynamic via multi-temporal HSR data, and a patch-based fine-scale spatial/contextual information via the single date VHSR scene. For every $patch_i$, we fix the window size to 25×25 pixels on the Spot6/7 (which corresponds to a window size 5x5 on a Sentinel-2 image) centered around a Sentinel-2 pixel described by the corresponding $ts_i$. In order to merge the temporal information (Sentinel-2) and the VHSR information (Spot 6/7), we designed a deep learning architecture which has two parallel branches, one for each of the two modes (spatial/temporal). For the Sentinel-2 pixel-based time series we use a Recurrent Neural Network (RNN) architecture. In particular, we used a Gated Recurrent Unit (GRU) introduced in (Cho *et al*., 2014) which has already demonstrated its effectiveness in the remote sensing field (Mou *et al*., 2017 ; Minh *et al*., 2018b). On the other hand, the spatial information supplied by the VHSR image is integrated in the pipeline via the use of a Convolutional Neural Network, a more suitable family of models for spatial/contextual feature extraction (Maggiori *et al*., 2017). The two branches of analysis learn complementary features that are successively combined for the land cover mapping, performed at the scale of the Sentinel-2 pixel. A third classifier, working on the fusion (by concatenation) of the two sets of features, produce the final land use classification.

FIGURE *1. Visual representation of $M^3Fusion$.*




### *3.2. Integration of HSR time series information*

Recurrent Neural Networks are well established machine learning techniques that demonstrate their quality in different domains such as speech recognition, signal processing, and natural language processing (Soma *et al.*, 2015 ; Linzen *et al.*, 2016). Unlike standard feed forward networks (e.g., Convolutional Neural Networks – CNNs), RNNs explicitly manage temporal data dependencies since the output of the neuron at time t-1 is used, together with the next input, to feed the neuron itself at time t. Recently, recurrent neural network (RNN) approaches have demonstrated their quality in the remote sensing field to produce land use mapping using time series of optical images (Ienco *et al.*, 2017) and recognize vegetation cover status using Sentinel-1 radar time series (Minh *et al.*, 2018b). Motivated by these recent research results, we introduce an RNN module to integrate information from the Sentinel-2 time series into our fusion process. In our model, we choose the GRU unit (Gated Recurrent Unit) introduced by (Cho *et al.*, 2014) since it has a moderate number of parameters to learn and it has already demonstrated its effectiveness in the field of remote sensing (Ienco *et al.*, 2017 ; Mou *et al.*, 2017). We coupled the Gated Recurrent Unit with an *attention* mechanism (Britz *et al.*, 2017).

The input of a RNN unit is a sequence of variables $(x_{t_1},..., x_{t_N})$ where a generic element $x_{t_i}$ is a feature vector and $t_i$ refers to the corresponding time stamp. In the context of HSR satellite image time series, $x_{t_i}$ is a vector with as many components as the number of spectral bands (including raw bands and indexes) carried by each satellite image. Equations 1, 2 and 3 formally describes the *GRU* neuron.

$$z_{t_i} = \sigma(W_{zx}x_{t_i} + W_{zh}h_{t_{i-1}} + b_z) \quad (1)$$

$$r_{t_i} = \sigma(W_{rx}x_{t_i} + W_{rh}h_{t_{i-1}} + b_r) \quad (2)$$

$$h_{t_i} = z_t \odot h_{t-1} + \quad (3)$$
$$(1 - z_{t_i}) \odot \tanh(W_{hx}x_t + W_{hr}(r_t \odot h_{t_{i-1}}) + b_h)$$

The $\odot$ symbol indicates an element-wise multiplication while $\sigma$ and $\tanh$ represent Sigmoid and Hyperbolic Tangent function, respectively.

The *GRU* unit has two gates, update ($z_t$) and reset ($r_t$), and one cell state, i.e., the hidden state ($h_t$). Moreover, the two gates combine the current input ($x_t$) with the information coming from the previous timestamp ($h_{t-1}$). The update gate effectively controls the trade off between how much information from the previous hidden state will carry over to the current hidden state and how much information of the current timestamp needs to be kept. On the other hand, the reset gate monitors how much information of the previous timestamps needs to be integrated with current information. As all hidden units have separate reset and update gates, they are able to capture dependencies over different time scales. Units more prone to capturing short-term dependencies will tend to have a frequently activated reset gate, but those that capture longer-term dependencies will have update gates that remain mostly active (Cho *et al.*, 2014). This behavior enables the GRU unit to remember long-term information.

Attention mechanisms (Britz *et al.*, 2017) are widely used in automatic signal processing (language or 1D signal) and they allow to gather together the information extracted by the GRU model at the different timestamps. The output returned by the GRU model is a sequence of feature vectors learned for each date: ($h_{t_1}$,..., $h_{t_N}$) where each $h_{t_i}$ has the same dimension $d$. Their matrix representation $H \in \mathbb{R}^{N,d}$ is obtained vertically stacking the set of vectors. The attention mechanism allows us to combine together these different vectors $h_{t_i}$, in a single one $rnn_{feat}$, to attentively combine the information returned by the GRU unit at each of the different timestamps. The attention formulation we use, starting from a sequence of vectors encoding the learned descriptors ($h_{t_1}$,..., $h_{t_N}$), is the following one:

$$v_a = tanh(H \cdot W_a + b_a) \tag{4}$$

$$\lambda = SoftMax(v_a \cdot u_a) \tag{5}$$

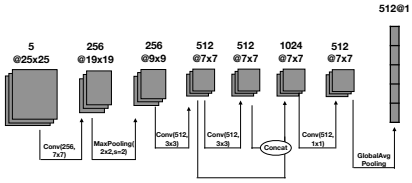
$$rnn_{feat} = \sum_{i=1}^{N} \lambda_i \cdot h_{t_i} \tag{6}$$

where matrix $W_a \in \mathbb{R}^{d,d}$ and vectors $b_a, u_a \in \mathbb{R}^d$ are parameters learned during the process. These parameters allow to combine the vectors contained in matrix $H$. The purpose of this procedure is to learn a set of weights ($\lambda_{t_1}$,..., $\lambda_{t_N}$) that allows the contribution of each time stamp to be weighted by $h_{t_i}$ through a linear combination. The $SoftMax(\cdot)$ (Ienco *et al.*, 2017) function is used to normalize weights $\lambda$ so that their sum is equal to 1. The output of the RNN module is the feature vector $rnn_{feat}$: they encode temporal information related to $ts_i$ for the pixel $i$.

### *3.3. Integration of VHSR information*

The VHSR information is integrated in $M^3Fusion$ through a CNN module. Computer vision literature offers several convolutional architectures (He *et al.*, 2016 ; Huang *et al.*, 2017) aimed at classifying images. Most of these networks are designed to process RGB images (three channels) having size higher than 200x200 pixels. Such networks are composed by multiple (tens or hundreds) layers.

FIGURE 2. *Convolutional Neural Network Structure*




In our scenario, the image patch has a size of 25x25 pixels and it involves five channels. In order to adopt a CNN module that well fits our scenario and remains computational affordable parameter-wise, we design the CNN module depicted in Figure 2. Our CNN network applies a first 7×7 kernel to the five-channel patch in order to produce 256 feature maps. Then, a *max pooling* layer is used to reduce the size and the number of parameters. Two successive convolution operations, with a 3×3 kernel, extract 512 feature maps each, which in their turn, are concatenated and reduced again by a convolution 1×1 kernel. The final size is then 512×7×7. Finally, a Global Average Pooling operation enables the construction of a feature vector of size 512. Each convolution is associated with a linear filter, followed by a Rectifier Linear Unit (ReLU) activation function (Nair, Hinton, 2010) to introduce non-linearity and a batch normalization step (Ioffe, Szegedy, 2015). The key points of our proposal are twofold: a) a higher number of filters in the first step and b) the concatenation of feature maps at different resolutions. The first point is related to the higher amount of spectral information (five channels) in input of our model compared to RGB images. The second point concerns the concatenation of feature maps. With the goal of exploiting information at different resolutions we adopt a philosophy similar to (Huang *et al.*, 2017) in which feature maps, at different level of the Deep architecture, are concatenated together. The output of this module is a vector of dimensions 512 ($cnn_{feat}$) which summarizes the spatial context ($patch_i$) associated to the $i$-th Sentinel-2 pixel.

### *3.4. The End-To-End Fusion process*

One of the advantages of deep learning, compared to standard machine learning methods, is the ability to link, in a single pipeline, the feature extraction step and the associated classifier (Zhang, Du, 2016). This capability is particularly important in a multi-source, multi-scale and multi-temporal fusion process, such as the one represented by our scenario. $M^3Fusion$ leverages this characteristic to extract complementary knowledge from two data sources that describe the same information from different points of view. To further strengthen the complementarity as well as the discriminative power of the learned features for each information branch, we adapt the technique proposed in (Hou *et al.*, 2017) to our problem. In (Hou *et al.*, 2017), the authors propose to learn two complementary representations (using two convolutional networks) from the same image. The discriminative power is enhanced by two auxiliary classifiers, linked to each group of features, in addition to the classifier that uses the merged

information. The complementary is enforced by alternating the optimization of the parameters of the two branches. In our case, we have two complementary sources of information (sentinel-2 time series and VHSR data) to which two auxiliary classifiers are connected to independently increase their ability to discriminate among land cover classes.

In detail, the classifier that exploits the full set of features is fed by concatenating the output features of both CNN ($cnn_{feat}$) and RNN ($rnn_{feat}$) modules together. Empirically, we have observed that the RNN module overfits the data. To alleviate this problem, we add a Dropout layer (Dahl *et al.*, 2013) on $rnn_{feat}$ with a drop-rate equals to 0.4. The learning process involves the optimization of three classifiers at the same time, one specific to $rnn_{feat}$, a second one related to $cnn_{feat}$ and the third one that considers $[rnn_{feat}, cnn_{feat}]$.

The cost function associated to our model is :

$$\begin{aligned} L_{total} &= \alpha_1 * L_1(rnn_{feat}, W_1, b_1) + \\ &= \alpha_2 * L_2(cnn_{feat}, W_2, b_2) + \\ &= L_{fus}([cnn_{feat}, rnn_{feat}], W_3, b_3) \end{aligned} \tag{7}$$

where

$$L_i(feat, W_i, b_i) = L_i(Y, SoftMax(feat \cdot W_i + b_i))$$

with $Y$ being the true value of the class variable. $L_1(rnn_{feat}, W_1, b_1)$ (resp. $L_2(cnn_{feat}, W_2, b_2)$) is the cost function of the first (resp. the second) auxiliary classifier that takes as input the set of descriptors returned by the RNN module (resp. CNN module) and the parameters $W_1, b_1$ (resp. $W_2, b_2$) to make the prediction. $L_{fus}(cnn_{feat}, rnn_{feat}, W_3, b_3)$ is the cost function of the classifier that uses the combined set of features ($[cnn_{feat}, rnn_{feat}]$). This last cost function is parameterized through $W_3$ et $b_3$. Each cost function is modeled through categorical cross entropy, a typical choice for multi-class supervised classification tasks (Ienco *et al.*, 2017).

$L_{total}$ is optimized end-to-end. Once the network has been trained, the prediction is carried out only by means of the classifier involving $W_3$ and $b_3$, which uses all the features learned by the two branches.

## 4. Experiments

In this section, we present and discuss the experimental results obtained on the study site introduced in Section 2.

In the evaluation, we investigate several points deeply related to a more clear understanding of our framework. Firstly, we provide a comparison between the ability of $M^3Fusion$ and a standard machine learning classifier (*Random Forest*) to deal with land cover mapping, secondly, we assess the effectiveness of the fusion method with respect to the use of each information source alone and finally, we perform a qualitative study considering the maps obtained by the competing methods. This evaluation supplies some examples that support the quality and effectiveness of our framework.

### *4.1. Experimental Settings*

For the *RF* model, we fix the number of generated random trees to 200. We use the publicly available Python implementation supplied by the *scikit-learn* library (Pedregosa *et al.*, 2011). In order to fairly compare the two methods, we supply the same input data set both to *RF* and to $M^3Fusion$. Each example of the data set provided to the Random Forest approach has a size of 3 669, corresponding to $25 \times 25 \times 5$ ($patch_i$) plus $34 \times 16$ ($ts_i$).

In our model, we choose the value $d$ (number of hidden units for the RNN module) equals to 1 024. We empirically fix $\alpha_1$ and $\alpha_2$ to 0.3. During the learning phase, we use the Adam method (Kingma, Ba, 2014) to learn the model parameters with a learning rate equals to $2 \cdot 10^{-4}$. The training process is conducted over 400 epochs. The model that reaches the lowest value of the cost function (at training time) is used in the test phase. $M^3Fusion$ is implemented using the Python Tensorflow library. The learning phase takes about 15 hours while the classification on the test data takes about one minute on a workstation with an Intel (R) Xeon (R) CPU CPU E5-2667 CPU v4@3.20Ghz with 256 GB of RAM and TITAN X GPU.

The data are prepared as follows. We divide the dataset into two parts, one for learning and the other one to test the performance of the supervised classification methods. We used 30% of the objects for the training phase (97 110 pixels - 797 objects) while the remaining 70% are used for the test phase (225 638 pixels - 1 859 objects). We impose that pixels of the same object belong exclusively to the training or to the test set (Inglada *et al.*, 2017). The values are normalized, per spectral band, in the interval $[0, 1]$. Finally, the assessment of the classification performances is done considering global precision (*Accuracy*), *F-Measure* (Ienco *et al.*, 2017) and Kappa.

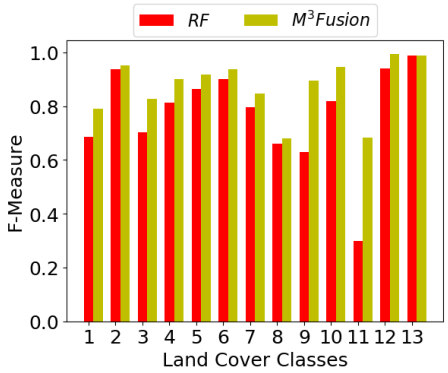
### *4.2. Comparative Analysis*

Figure 3 provides the results of a comparative analysis between our model and Random Forest (RF), an ensemble learning method that is commonly employed in the field of Remote Sensing for dealing with the land cover mapping task.

We observe that $M^3Fusion$ reaches higher performance indices than Random Forest on all the land cover classes. The highest gains are related to classes (1), (3), (9), (10) and (11) (resp. *Crop Cultivation*, *Orchards*, *Bare rocks*, *Urbanized areas* and *Greenhouse crops*). Considering the characteristics of such classes, the different gains are the results of the effectiveness of $M^3Fusion$ to combine temporal and fine spatial information together leveraging the complementary of the two sources of information.

As further comparative analysis, in Table 2 we report a summary of the results obtained by applying the two approaches ($M^3Fusion$ and RF) on the fusion of the two information sources (VHSR and SITS) as well as on each single source of information individually. In the latter case, the approaches are named as follows: $RF_{ts}$ and $M^3Fusion_{ts}$ stand for RF and $M^3Fusion$ applied on time series data only; $RF_{vhsr}$ and $M^3Fusion_{vhsr}$ stand for RF and $M^3Fusion$ applied on VHSR data only. We com-

FIGURE *3. Per class F-Measure results of Random Forest and $M^3Fusion$ methods.*




pare the different methods by means of Accuracy, (average) F-Measure and Kappa. We can observe that also for the Random Forest approach the use of multiple sources results in a general improvement of land cover mapping performances. This behavior points out that the two sources of information carrying out complementary knowledge and the joint use of temporal and fine spatial information positively influence the land cover classification task.

After a closer look at the values of all the evaluation metrics, we can state that, on the *Reunion Island* dataset, the data fusion process implemented by $M^3Fusion$ is more effective than the one carried out by the Random Forest classifier. The Deep Learning data fusion approach smartly leverages the complementary information reaching a gain of more than 0.06 Accuracy point with respect to the best individual source application scenario ($M^3Fusion_{vhsr}$) while, in the case of Random Forest, this gain is limited to less than 0.02 Accuracy points compared to its best individual source result ($RF_{ts}$).

TABLE *2. Accuracy, F-Measure, Kappa of different methods considering the fusion process as well as one source at time.*

| | *Accuracy* | *F-Measure* | *Kappa* |
|---|---|---|---|
| $RF_{ts}$ | 0.8543 | 0.8519 | 0.8258 |
| $M^3Fusion_{ts}$ | 0.8319 | 0.8325 | 0.8033 |
| $RF_{vhsr}$ | 0.8237 | 0.8140 | 0.7908 |
| $M^3Fusion_{vhsr}$ | 0.8369 | 0.8364 | 0.8677 |
| $RF$ | 0.8716 | 0.8681 | 0.8491 |
| $M^3Fusion$ | **0.9149** | **0.9148** | **0.9000** |

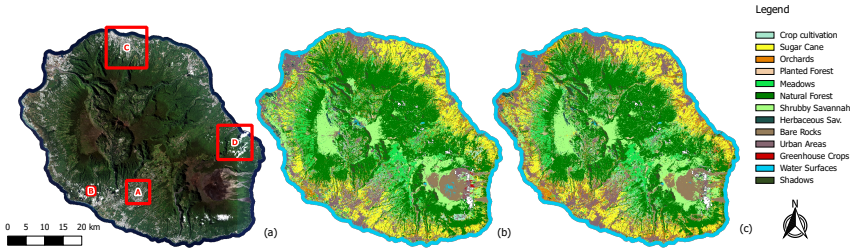
### *4.3. Map Comparison*

In addition to the evaluations reported in the previous sections, we also propose a first visual qualitative evaluation of the produced maps. The maps obtained by

$M^3Fusion$ (resp. Random Forest) is shown in Figure 4b (resp. Figure 4a) for a qualitative overview. When we visually analyze the map issued by $M^3Fusion$, we observe that the detection of the majority classes, i.e., the areas cultivated with sugar cane on the coast, as well as the various natural areas (grasslands, savannas and forests) and the urban areas are well recognized with less salt and pepper error than the map produced by the *Random Forest* classifier (Figure 4a).

Some comparisons between the two maps are provided at the scale of some remarkable details in Figure 3: in the first column, a fragment of urban areas is displayed, where the presence of noise is particularly marked for the $RF$'s map (in the middle). This phenomenon is highlighted by the transition zones between buildings, which are often interpreted as crops. This effect is less present on the $M^3Fusion$'s map (bottom). A particular interesting effect concerns the artifacts of the $RF$ map due to the presence of clouds or shadows (detail on second column) on the VHSR image, which are definitely mitigated in the map produced by $M^3Fusion$ for this example. This effect is also visible on larger cloudy areas (last column), where some errors persist in $M^3Fusion$'s map but most of the affected area is correctly retrieved. A possible explanation for these aberrations could be a biased prediction behavior of $RF$ in favor of information coming from VHSR data. Notice that this situation does not occur when the same data are processed by $M^3Fusion$. This behavior can be explained by the way the *Random Forest* works and the feature cardinality of each data source. Due to the random nature of *Random Forest*, each time it samples a random set of features to build the trees belonging to the forest. Since the number of features available from the VHSR source (3 125) is bigger than the number of features coming from the time series data (544); *Random Forest* tends to exploit more frequently features coming from the former source than from the latter one. This fact probably bias the *Random Forest* to overuse VHSR information. A last example showing map improvements is on the third column of Fig. 3, where the dense urban area has reduced noise in $M^3Fusion$'s map with respect to $RF$'s one, and some clear errors are corrected on vegetated areas (e.g. grass among airport lanes is erroneously classified as sugar cane using $RF$, while $M^3Fusion$mostly detects the meadows class).

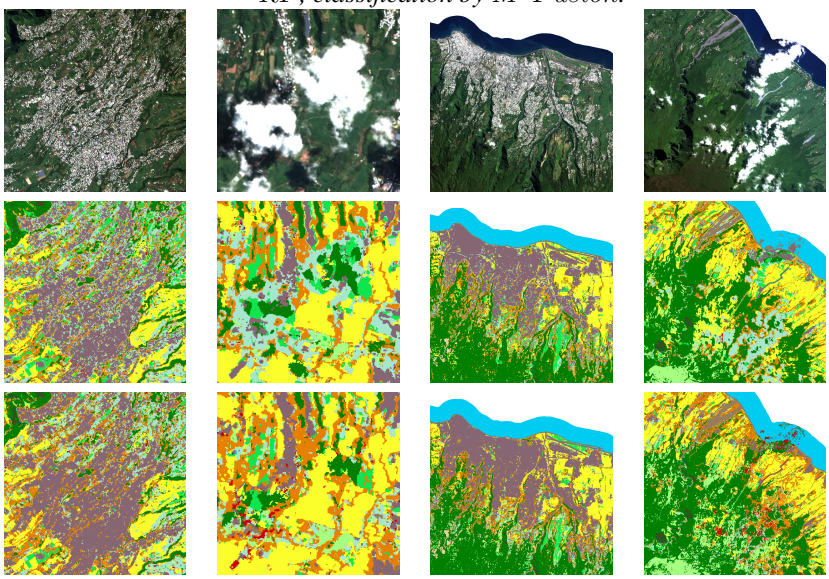
FIGURE *4. Source VHRS scene (a) (see Fig. 3 for details in red boxes), maps produced by $RF$ (a) and $M^3Fusion$ (b).*



## 5. Conclusions

In this article, we have proposed a new deep learning architecture for the fusion of satellite data at high temporal/spatial resolution with an image at very high spatial resolution (VHSR) to perform land cover mapping. Experiments carried out on a study site have validated the quality and effectiveness of our approach compared to a common machine learning approach usually employed in the field of remote sensing. In the future, we plan to investigate several extensions of our architecture to integrate other complementary data sources.

TABLE *3. Classification results obtained with $RF$ and $M^3Fusion$. Top to bottom: excerpts from SPOT6/7 imagery (respectively A,B,C,D from Fig. 4), classification by $RF$, classification by $M^3Fusion$.*



## 6. Acknowledgements

This work was supported by the French National Research Agency under the Investments for the Future Program ANR-16-CONV-0004 (DigitAg), ANR-10-EQPX-20 (GEOSUD). This work also used an image acquired under the CNES Kalideos scheme (La Réunion site) and the Programme National de Télédétection Spatiale (PNTS, http://www.insu.cnrs.fr/pnts ), grant $n^o$PNTS-2018-5.

## Bibliographie

Abade N. A., Júnior O. A. d. C., Guimarães R. F., Oliveira S. N. de. (2015). Comparative analysis of modis time-series classification using support vector machines and methods

based upon distance and similarity measures in the brazilian cerrado-caatinga boundary. *Remote Sensing*, vol. 7, nº 9, p. 12160–12191.

Britz D., Guan M. Y., Luong M. (2017). Efficient attention using a fixed-size memory representation. In *Emnlp*, p. 392–400.

Cho K., Merrienboer B. van, Gülçehre Ç., Bahdanau D., Bougares F., Schwenk H. *et al*. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Emnlp*, p. 1724–1734.

Dahl G. E., Sainath T. N., Hinton G. E. (2013). Improving deep neural networks for LVCSR using rectified linear units and dropout. In *Icassp*, p. 8609–8613.

Geng J., Wang H., Fan J., Ma X. (2018). SAR image classification via deep recurrent encoding neural networks. *IEEE Trans. Geoscience and Remote Sensing*, vol. 56, nº 4, p. 2255–2269.

Hagolle O., Huc M., Villa Pascual D., Dedieu G. (2015). A Multi-Temporal and Multi-Spectral Method to Estimate Aerosol Optical Thickness over Land, for the Atmospheric Correction of FormoSat-2, LandSat, VEN$\mu$S and Sentinel-2 Images. *Remote Sensing*, vol. 7, nº 3, p. 2668–2691.

He K., Zhang X., Ren S., Sun J. (2016). Deep residual learning for image recognition. In *Cvpr*, p. 770–778.

Hou S., Liu X., Wang Z. (2017). Dualnet: Learn complementary features for image recognition. In *IEEE iccv*, p. 502–510.

Huang G., Liu Z., Maaten L. van der, Weinberger K. Q. (2017). Densely connected convolutional networks. In *Cvpr*, p. 2261–2269.

Ienco D., Gaetano R., Dupaquier C., Maurel P. (2017). Land cover classification via multitemporal spatial data by deep recurrent neural networks. *IEEE GRSL*, vol. 14, nº 10, p. 1685–1689.

Inglada J., Vincent A., Arias M., Tardy B., Morin D., Rodes I. (2017). Operational high resolution land cover map production at the country scale using satellite image time series. *Remote Sensing*, vol. 9, nº 1, p. 95.

Ioffe S., Szegedy C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Icml*, p. 448–456.

Karpatne A., Jiang Z., Vatsavai R. R., Shekhar S., Kumar V. (2016). Monitoring land-cover changes: A machine-learning perspective. *IEEE Geoscience and Remote Sensing Magazine*, vol. 4, p. 8–21.

Kingma D. P., Ba J. (2014). Adam: A method for stochastic optimization. *CoRR*, vol. abs/1412.6980.

Lebourgeois V., Dupuy S., Vintrou E., Ameline M., Butler S., Bégué A. (2017). A combined random forest and OBIA classification scheme for mapping smallholder agriculture at different nomenclature levels using multisource data (simulated sentinel-2 time series, VHRS and DEM). *Remote Sensing*, vol. 9, nº 3, p. 259.

Linzen T., Dupoux E., Goldberg Y. (2016). Assessing the ability of lstms to learn syntax-sensitive dependencies. *TACL*, vol. 4, p. 521–535.

Maggiori E., Tarabalka Y., Charpiat G., Alliez P. (2017). Convolutional neural networks for large-scale remote-sensing image classification. *IEEE TGRS*, vol. 55, nº 2, p. 645–657.

Minh D. H. T., Ienco D., Gaetano R., Lalande N., Ndikumana E., Osman F. *et al*. (2018a). Deep recurrent neural networks for winter vegetation quality mapping via multitemporal SAR sentinel-1. *IEEE Geosci. Remote Sensing Lett.*, vol. 15, n° 3, p. 464–468.

Minh D. H. T., Ienco D., Gaetano R., Lalande N., Ndikumana E., Osman F. *et al*. (2018b). Deep recurrent neural networks for winter vegetation quality mapping via multitemporal sar sentinel-1. *IEEE GRSL*, vol. Preprint, n° -, p. -.

Mou L., Ghamisi P., Zhu X. X. (2017). Deep recurrent neural networks for hyperspectral image classification. *IEEE TGRS*, vol. 55, n° 7, p. 3639–3655.

Nair V., Hinton G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Icml10*, p. 807–814.

Pedregosa F., Varoquaux G., Gramfort A., Michel V., Thirion B., Grisel O. *et al*. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, vol. 12, p. 2825–2830.

Schmitt M., Zhu X. X. (2016). Data fusion and remote sensing: An ever-growing relationship. *IEEE Geoscience and Remote Sensing Magazine*, vol. 4, n° 4, p. 6–23.

Soma K., Mori R., Sato R., Furumai N., Nara S. (2015). Simultaneous multichannel signal transfers via chaos in a recurrent neural network. *Neural Computation*, vol. 27, n° 5, p. 1083–1101.

Zhang L., Du B. (2016). Deep learning for remote sensing data: A technical tutorial on the state of the art. *IEEE Geoscience and Remote Sensing Magazine*, vol. 4, p. 22–40.

# Classification d'objets urbains à partir de données LiDAR 3D terrestre par Deep-Learning

**Younes Zegaoui**[1, 2], **Marc Chaumont**[1, 3], **Gérard Subsol**[1], **Philippe Borianne**[4], **Mustafa Derras**[2]

1. *LIRMM, Univ. Montpellier, CNRS, France*
   *zegaoui@lirmm.fr,subsol@lirmm.fr*
2. *Berger-Levrault, Toulouse, France*
   *mustafa.derras@berger-levrault.com*
3. *Univ. Nîmes, France*
   *marc.chaumont@lirmm.fr*
4. *AMAP, Univ. Montpellier, CIRAD, CNRS, INRA, IRD, Montpellier, France*
   *philippe.borianne@cirad.fr*

RÉSUMÉ. *La détection automatique d'objets urbains reste un défi pour les gestionnaires de ville. Les approches existantes en télédétection comprennent l'utilisation d'imagerie aérienne ou de LiDAR pour cartographier une scène. Nous voulons mettre à l'épreuve les méthodes 3D Deep Learning pour aborder le problème de la détection d'objets urbains. Dans cet article, nous présentons les résultats de plusieurs expériences sur la classification des objets urbains avec le réseau PointNet.*

ABSTRACT. *Automatic urban object detection remains a challenge for city management. Existing approaches in remote sensing include using aerial images or LiDAR to map a scene. We then wanted to make use of the rise of 3D Deep-Learning methods to tackle the issue of urban object detection with the help of terrestrial LiDAR acquisition. In this paper we present result of several experiments on urban object classification with the PointNet network.*

MOTS-CLÉS : *LiDAR, deep-learning, nuages de points, objets urbains*

KEYWORDS: *LiDAR, deep-learning, 3D points clouds, urban objects*

## 1. Présentation et contexte

La reconnaissance de formes est entrée dans une nouvelle ère avec le développement des algorithmes d'apprentissage en profondeur, ou Deep-Learning, dans la communauté universitaire comme dans le monde industriel, grâce aux avancées majeures réalisées ces 5 dernières années essentiellement dans le traitement des images 2D. Les méthodes d'apprentissage en profondeur s'étendent maintenant à de très nombreux domaines scientifiques ou industriels,qui pourraient potentiellement tous bénéficier des progrès récents de cette technologie LeCun *et al.* (2015).

Actuellement, la plupart des recherches sur l'apprentissage en profondeur proviennent de l'imagerie 2D où elles ont permis des gains de performances énormes dans la classification d'images, visibles en particulier dans le challenge ILSVRC Russakovsky *et al.* (2014). Cette augmentation soudaine des performances a permis à des algorithmes de détection d'objets robustes en temps réel tel que Faster R-CNN Ren *et al.* (2015) d'émerger. Cependant, leur généralisation à des données 3D n'est pas une tâche simple. Cela est particulièrement vrai dans le cas des nuages de points 3D où les informations ne sont pas structurées comme dans les maillages 3D.

Les capteurs LiDAR terrestres, qui sont de nos jours généralement monté sur des appareils mobiles tels que des voitures, peuvent être utilisés pour effectuer une acquisition dynamique d'une scène entière telle qu'une ville ou une agglomération. Cela peut facilement être réalisé avec une voiture équipée de LiDAR traversant le réseau routier d'une ville, générant un nuage de points 3D correspondant à la ville dans son entièreté Anguelov *et al.* (2010). Par rapport à un simple enregistrement vidéo, une telle acquisition donne une information contextuelle 3D et fournit des mesures précises de profondeur.

A partir d'une acquisition LiDAR, il serait possible de détecter les objets d'une scène 3D et d'utiliser cette information dans la gestion d'une agglomération. Par exemple, connaître précisément le nombre d'arbres ou de poteaux présents et leurs emplacements aiderait grandement à mettre à jour les bases de données d'objets urbains, les localisant précisément et en gardant une trace de leur statut. D'autre part, il est particulièrement intéressant de suivre des objets en constante évolution, l'exemple le plus notable étant les arbres. La plupart des appareils LiDAR mobiles sont aussi équipés d'émetteurs GNSS (Global Navigation Satellite System) qui permettent un géoréférencement des données lors de l'acquisition. Ainsi, tout objet urbain détecté dans le nuage de points peut être projeté sur un SIG (Système d'Information Géographique) existant. Néanmoins, pour que des algorithmes de localisation basé Deep-Learning fonctionnent, nous devons au préalable nous assurer que leur contreparties en classification fournissent des résultats acceptables.

Dans cet article, nous proposons d'évaluer un réseau de Deep-Learning 3D récent et performant sur une tâche de classification de nuages de points. Après

un bref état de l'art du Deep-Learning en 3D dans la section 2, nous présentons notre méthodologie de classification dans la section 3. Dans la section 4, nous donnons les résultats expérimentaux et évaluons comment le réseau neuronal reconnaît un nuage de points 3D comme un objet urbain.

## 2. Deep-Learning en 3D

Comme précédemment indiqué, il n'existe pas de moyen simple de généraliser les méthodes de classification des images 2D aux nuages de points 3D. Ceci est dû au fait qu'un nuage de points 3D est une structure de données non ordonnée contrairement à une image où les pixels sont ordonnés. Il n'y a pas de corrélation entre l'ordre dans lequel les points sont classés dans le nuage, qui est simplement une liste de sommets (X, Y, Z), et les informations qu'ils encodent. Le nuage de points reste le même quelle que soient les permutations appliquées à la liste, alors que dans les images 2D, les valeurs des pixels voisins sont plus ou moins corrélées entre elles. Afin de contourner ces problèmes, les premiers algorithmes de classification 3D ont utilisé des structures de données intermédiaires pour représenter les nuages de points.

Les méthodes existantes peuvent être divisées en trois sous-catégories selon la structure de données intermédiaire qu'elles utilisent :

– Les méthodes basées sur les voxels ont été les premières à fournir des résultats significatifs. Elles utilisent des algorithmes de voxellisation pour transformer les nuages de points en images volumiques, comme celles utilisées dans l'imagerie médicale. Les convolutions spatiales peuvent alors être appliquées de la même manière que pour les images 2D avec des noyaux définis sur des voxels au lieu des pixels. Il est donc possible d'utiliser une même architecture de réseau neuronal convolutif (CNN) sur les données voxellisées en adaptant légèrement les couches afin qu'elles puissent traiter correctement la troisième dimension. VoxNet Maturana, Scherer (2015) utilise une grille d'occupation pour l'étape de voxellisation et un CNN à 3 couches pour la tâche de classification. Les auteurs de ORION "ORION" (s. d.) montrent que l'ajout d'une prédiction d'orientation à la tâche de classification aide le réseau à obtenir de meilleurs résultats. OctNet Riegler *et al.* (2017) propose une méthode de voxellisation des nuages en octree pour résoudre un des problèmes majeurs de ces méthodes, à savoir la consommation de mémoire graphique (VRAM) due à la taille en mémoire nécessaire pour avoir une résolution volumique suffisante. À compter de 2018, les performances des méthodes basées sur les voxels sont globalement inférieures aux deux autres.

– Les méthodes multi-vues génèrent plusieurs angles de vue autour de l'objet 3D avec une caméra virtuelle et synthétisent pour chaque angle une image 2D de l'objet. Les images 2D peuvent être en niveaux de gris, en RGB classique, une carte de profondeur ou une silhouette (image binaire). Les images 2D sont ensuite traitées par un CNN classique pour obtenir la classe. Le réseau

PairWise (Johns *et al.*, 2016) propose un réseau capable de prédire le meilleur point de vue suivant selon une vue initiale. En termes de précision, ces méthodes donnent les meilleurs résultats. Par exemple RotationNet Kanezaki *et al.* (2018) définit actuellement un nouvel état de l'art sur le jeu de données de Princeton ModelNet40. Cependant, elles supposent qu'il est possible de synthétiser des images 2D à partir de tous les angles possibles autour de l'objet. Cela est généralement vrai pour des maillages CAO qui sont triangulés et complets. Par contre les LiDAR fournissent des nuages de points qu'il faut trianguler, ce qui n'est pas facile du fait des fortes variations de densité des points et des nombreuses occultations.

– Enfin, les approches basées uniquement sur les points ne nécessitent pas de structure de données intermédiaire. Apprendre directement sur les coordonnées des points est une option qui a réellement démarré en 2017 avec l'apparition de PointNet Charles *et al.* (2017). Sa particularité est de ne pas utiliser de structure intermédiaire entre le nuage de points et le réseau : les points sont directement passés au réseau. Il y a deux problèmes principaux à aborder lors de l'apprentissage direct à partir des coordonnées des points : le réseau doit être invariant par rapport à l'ordre des points et au repère géométrique des nuages. Vous trouverez plus de détails sur la manière dont PointNet gère ces problèmes dans la section suivante. PointNet ++ Qi *et al.* (2017) et Engelmann *et al.* (2017) ont proposé d'ajouter des couches supplémentaires pour permettre à PointNet d'utiliser les informations de contexte spatial. KD-net Klokov, Lempitsky (2017) propose d'utiliser un arbre KD qui est généralement utilisé pour réduire les coûts de calcul, pour ajouter une information de contexte tout en apprenant à partir des coordonnées des points.

Le fait que PointNet n'ait pas besoin d'une structure de données intermédiaire et qu'il fournisse de très bons résultats malgré une architecture relativement simple comparée aux autres réseaux basés sur VGG ou ResNet, en fait le candidat le plus approprié pour nos expériences de classification.

## 3. Description du réseau PointNet

Le but de l'architecture PointNet Charles *et al.* (2017) est de calculer une fonction symétrique afin que les résultats produits par le réseau soient invariants par rapport à l'ordre des points dans le nuage. Il comprend également un mini-réseau appelé T-Net pour gérer le problème de normalisation des coordonnées.

Le T-Net est essentiellement une version miniaturisée de PointNet dont l'objectif est de prédire les coefficients d'une matrice 3x3. Ces coefficients correspondent à une transformation affine qui est appliquée au nuage de points en entrée afin d'aligner les données sur un espace canonique.

Une fois que les points sont "alignés" par le T-Net, ils passent par le module d'extraction de caractéristiques. Ce module consiste en une série de couches

MLP (Multi Layer Perceptron) où des vecteurs de caractéristiques sont extraits de chaque point, suivis par une opération d'agrégation regroupant ces caractéristiques en un vecteur global. Ce qui correspond au calcul d'un espace d'entité invariant à l'ordre des points. Le vecteur global passe ensuite dans un classificateur qui permet de prédire la classe du nuage de points.

## 4. Expériences de classification

Nous avons décidé d'évaluer les performances de la *classification* d'objets urbains, numérisés par acquisition LiDAR, avec l'utilisation du réseau PointNet. Chaque nuage de points 3D, c.-à-d. une liste de coordonnées (X, Y, Z), contient un seul objet urbain. Le but de ces expériences est d'évaluer la capacité du réseau à prédire la classe d'un objet en traitant uniquement ses coordonnées. Nous allons décrire brièvement les expériences dans la section suivante.

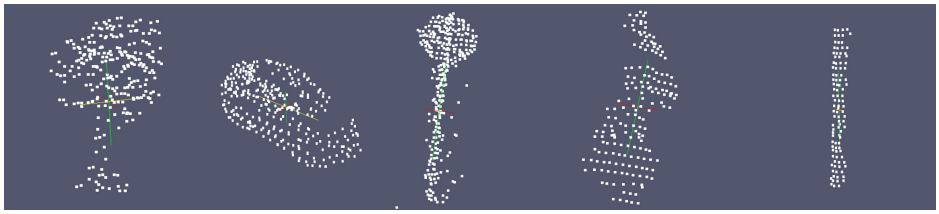


FIGURE 1 – BaseA : de gauche à droite, un arbre, une voiture, un panneau de signalisation, un piéton, un poteau.

### *4.1. Jeu de données utilisé*

En raison du manque de base de données de référence dans le domaine de la classification de nuages de points 3D, nous avons décidé de rassembler des données à partir de plusieurs bases de données annotées contenant des nuages de points d'objets urbains. Nous répartissons les objets urbains en 8 classes différentes: arbre, voiture, feu et panneau de signalisation, poteau, piéton, bâtiment, bruit et artefact. Quelques exemples sont présentés dans la Fig. 1. Nous avons trouvé trois ensembles de données accessibles au public correspondaient à ces critère à notre connaissance : Lai, Fox (2010), Serna *et al.* (2014) et Quadros *et al.* (2012). Cela fait un total de 724 objets dont 80% utilisés pour l'apprentissage et 20% utilisés pour la validation. Dans le reste de l'article, nous ferons référence à cet ensemble de données sous le nom de BaseA.

De plus, nous avons réalisé une acquisition LiDAR terrestre à l'aide d'un sac à dos Leica Pegasus [1]. Cette acquisition a été réalisée en milieu urbain par une personne équipée du sac sur 200 mètres à pied. Une vue globale de la scène urbaine peut être trouvée dans la Fig. 2. Nous avons extrait manuellement 174

1. Nous aimerions remercier LeicaGeosystems d'avoir réalisé l'acquisition LiDAR.

objets urbains de cette acquisition et les avons annotés. La répartition de cette ensemble de données, appelé BaseB dans la suite de l'article, est la suivante: 75 arbres, 39 voitures, 8 feux et panneaux de signalisation, 23 poteaux et 29 piétons.

La taille des nuages de points est comprise entre 200 et 3 000 points pour la BaseA et entre 1000 et 300 000 pour la BaseB. Nous avons ensuite ré-échantillonnés chaque objets à 512 points avec un filtre "voxel-grid" afin qu'ils puissent être utilisés comme entrée pour le réseau PointNet. Dans la Fig. 2, nous présentons une illustration de cette étape pour un arbre de la BaseB. Les deux bases A et B sont toutes deux accessibles : http://www.lirmm.fr/~chaumont/DemoAndSources.html

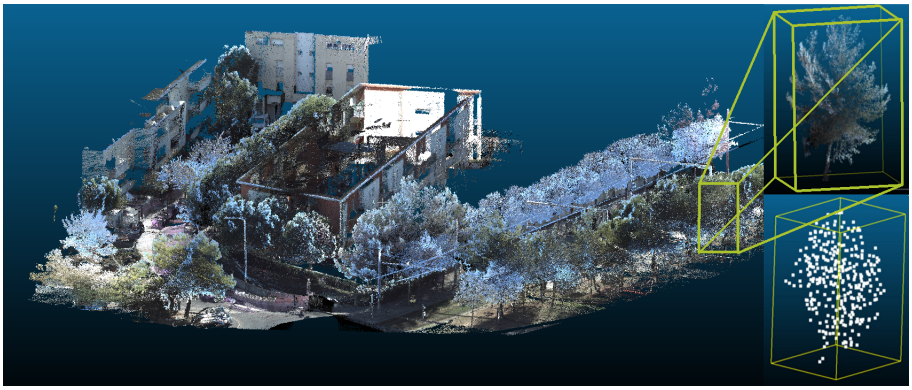


FIGURE 2 – Vue globale de notre acquisition LiDAR terrestre. On peut y distinguer des arbres, un arrêt de tramway ainsi qu'une petite résidence. A droite, un arbre est isolé puis sous échantillonné en un nuage de 512 points

### *4.2. Expérience témoin*

Dans cette section, nous rappelons l'expérience de Zegaoui *et al.* (2018). Nous utilisons l'implémentation de PointNet [2] sous Tenserflow pour entraîner le réseau avec la BaseA et le tester sur la BaseB. Cette implémentation utilise l'augmentation des données avec des rotations aléatoires sur l'axe vertical, un bruit gaussien sur les coordonnées, l'optimisation Adam et la normalisation des lots (batch normalization). Les résultats sont reportés dans le tableau 1.

Nous avons constaté que la F mesure global était de 0.744. Les scores F1 pour la classe arbre : 0.896, voiture : 0.904 et piéton : 0.828, sont satisfaisants. Cependant, pour les feux et panneaux de signalisation ainsi que les poteaux, les scores sont beaucoup plus faibles : 0.267 pour les panneaux de signalisation et 0.074 pour les poteaux.

2. https://github.com/charlesq34/pointnet

Nous remarquons que le réseau confond ces deux dernières classes. Le réseau prédit que 12 des 23 poteaux sont des panneaux de signalisation et 3 des 4 panneaux de signalisation, des poteaux. Dans l'ensemble, les résultats sont vraiment encourageants compte tenu du peu de données utilisées.

| | | **Vérité terrain (annotations)** | | | | |
|---|---|---|---|---|---|---|
| | | ***arbre (75)*** | ***voiture (39)*** | ***signalisation (8)*** | ***poteau (23)*** | ***piéton (15)*** |
| **Classification** | ***"arbre"*** | 69 | 2 | 0 | 8 | 0 |
| | ***"voiture"*** | 1 | 33 | 0 | 0 | 0 |
| | ***"signalisation"*** | 4 | 0 | 4 | 12 | 2 |
| | ***"poteau"*** | 0 | 0 | 3 | 1 | 0 |
| | ***"piéton"*** | 1 | 0 | 1 | 0 | 12 |
| | ***"bâtiment"*** | 0 | 0 | 0 | 2 | 0 |
| | ***"bruit"*** | 0 | 4 | 0 | 0 | 1 |
| | **F mesure** | 0.896 | 0.904 | 0.267 | 0.074 | 0.828 |

TABLE 1 – Matrice de confusion pour l'expérience témoin. F mesure globale : 0.744

### *4.3. Expériences supplémentaires*

Dans cette section, toutes les expériences reposent sur la même méthodologie que l'expérience témoin, sauf si précisé autrement.

#### *4.3.1. Rajout de base de données*

Dans cette expérience, nous enrichissons la BaseA avec certains des objets disponibles dans le très récent jeu de données Roynard *et al.* (2018).

À partir de ce jeu de données, nous extrayons plus de 900 objets qui correspondent à l'une de nos 8 classes et nous les ajoutons à notre BaseA. Nous avons ensuite effectué une autre expérience de classification avec la BaseA nouvellement augmenté en tant qu'ensemble d'entraînement et avec la BaseB comme ensemble de test. La taille de notre base d'entraînement augmente ainsi de 230% (1668 objets au total), avec la répartition suivante 349 arbres, 382 voitures, 279 feux et panneaux de signalisation, 292 poteaux, 164 piétons, 61 exemples de bruits et 141 bâtiments.

Les résultats, dans le tableau 2 montrent une augmentation du score F1 global : de 0.744 à 0.857 (+11.3%). L'augmentation de la F mesure affecte toutes les classes à l'exception de la classe piéton qui passe de 0.828 à 0.815 (-1.3 %). L'augmentation est la plus prononcée pour la classe poteau avec un gain de 0.074 à 0.629 (+55.5%). La classe voiture atteint également un score parfait de 1.000 (+9.6 %).

Nous observons une amélioration notable des performances provenant de l'enrichissement de notre ensemble d'entraînement. Cela n'est pas surprenant étant donné qu'il est généralement admis dans le domaine de l'apprentissage en profondeur que le moyen le plus efficace d'améliorer la précision d'un réseau est de l'alimenter avec davantage de données.

| | | Vérité terrain (annotations) | | | | |
|---|---|---|---|---|---|---|
| | | *arbre (75)* | *voiture (39)* | *signalisation (8)* | *poteau (23)* | *piéton (15)* |
| Classification | *"arbre"* | 70 | 0 | 0 | 2 | 1 |
| | *"voiture"* | 2 | 39 | 0 | 0 | 1 |
| | *"signalisation"* | 2 | 0 | 7 | 10 | 2 |
| | *"poteau"* | 0 | 0 | 1 | 11 | 0 |
| | *"piéton"* | 0 | 0 | 1 | 0 | 11 |
| | *"bâtiment"* | 1 | 0 | 0 | 0 | 0 |
| | *"bruit"* | 0 | 0 | 0 | 0 | 0 |
| | **F mesure** | 0.946 | 1.000 | 0.467 | 0.629 | 0.815 |

TABLE 2 – Matrice de confusion pour l'expérience avec ensemble d'entraînement enrichi. F mesure global : 0.857

#### *4.3.2. Fusion de classes*

Une des raisons des faibles performances à la fois pour les classes poteau et feu/panneau de signalisation est la confusion du réseau entre celles-ci. Ce qui est compréhensible étant donné qu'elles décrivent des objets géométriquement proches.

Nous les avons donc fusionnés en une seule classe TSLP (Traffic Sign/Light + Pole) et exécuter à nouveau l'expérience de classification avec les mêmes ensembles de données que dans notre expérience de base, mais avec seulement 6 classes au lieu de 7.

Les résultats dans le tableau 3 montrent un score F1 significativement plus élevé pour la classe TSLP, 0.849, que dans le tableau 1 pour la classe signalisation 0.267, et la classe poteau 0.074. Si nous combinons les résultats de l'expérience témoin pour ces deux classes, nous trouvons que 20 fois sur 31, le réseau a "correctement" prédit soit un poteau, soit un feu ou panneau de signalisation. Cela nous donne un score F1 de 0.702 qui reste inférieur à celui de la classe TSLP, 0.849(+14.7%). Il y a également une augmentation du score F1 global de 0.744 à 0.831 (+8.7%) ainsi qu'une légère augmentation pour la classe des arbres, de 0.896 à 0.920 (+ 2.4%). Cependant, nous pouvons également voir une diminution pour les classes voiture et piéton, respectivement de 0.904 à 0.806 (-9.8%) et de 0.828 à 0.750 (-7.8%).

Le réseau apprend correctement à différencier les feux et panneaux de signalisation ainsi que les poteaux du reste, mais il n'arrive pas à faire la différence entre les deux. Néanmoins, la fusion de ces classes ne peut pas être une stratégie à long terme car, comme l'indiquent nos résultats, le consensus général en matière d'apprentissage en profondeur est que plus il y a de classes utilisées lors de l'entraînement, meilleurs sont les résultats globaux. Cela vient du fait que moins de classes signifie moins de contre-exemples pour chacune des autres classes. C'est pourquoi la baisse de précision des classes piéton et voiture n'est pas surprenante.

| | | Vérité terrain (annotations) | | | |
|---|---|---|---|---|---|
| | | *arbre (75)* | *voiture (39)* | *TSLP(31)* | *piéton (15)* |
| Classification | *"arbre"* | 69 | 3 | 3 | 0 |
| | *"voiture"* | 1 | 27 | 0 | 0 |
| | *"TSLP"* | 2 | 0 | 28 | 5 |
| | *"piéton"* | 0 | 0 | 0 | 9 |
| | *"bâtiment"* | 1 | 0 | 0 | 0 |
| | *"bruit"* | 2 | 9 | 0 | 1 |
| | **F mesure** | 0.920 | 0.806 | 0.849 | 0.750 |

TABLE 3 – Matrice de confusion avec fusion des classes signalisation et poteaux.F mesure globale : 0.831

### *4.3.3. Augmentation virtuelle de la base*

Dans cette expérience, nous voulons tester une nouvelle manière de faire de l'augmentation de données, qui consiste à simuler des occultations. Pour chaque nuage de notre BaseA, nous en générons plusieurs versions "occultée". Un moyen simple d'y parvenir est de générer des plans aléatoires pour découper les nuages de points. Les plans sont définis par l'axe des z, voir Fig. 3 pour la visualisation des coordonnées, un angle aléatoire, et passent par l'isobarycentre du nuage. Le nuage est alors divisé en deux et nous ne conservons que la plus grande partie. Cette opération est répétée k fois pour chaque nuage.

Les résultats sont reportés dans les tableaux 4, 5 et 6 pour chaque k. Avec k = 5, le score F1 global est de 0.669 (-7.5%), pour k = 10, il est de 0.444 (-30%) et pour k = 50, il est de 0.578 (-16.6%). Le score de la classe poteau augmente de 0.074 dans l'expérience de base à 0,080 avec k = 5 (+0.6%) et 0.083 (+0.9%) avec k = 10 et k = 50. Le score F1 de la classe signalisation augmente de 0.267 à 0.370 (+10.3%) avec k = 10 et à 0.387 (+12%) avec k = 50. Toutes les autres classes voient leurs scores chuter parfois de manière drastique, par exemple le score de la classe de voiture passe de 0.904 à 0.453 (-45.1%) pour k = 10.

L'enrichissement de notre ensemble d'entraînement cause plus de diminution que d'augmentation au niveau des scores. Cela a tendance à être plus remarquable pour les valeurs plus élevées de k. Une explication de ce phénomène serait que plus on passe d'objets «coupés» au réseau, plus il apprend à reconnaître des plans géométrique et devient ainsi moins efficace lorsqu'il est testé sur un objet «entier».

### *4.3.4. Rotations*

Notre ensemble de test provient d'une acquisition LiDAR différente de notre ensemble d'entraînement, ce qui signifie que les nuages de points peuvent être orientés différemment, ce qui pourrait entraîner une baisse des performances si le réseau n'est pas invariant à la rotation. Notre expérience consiste à appliquer des rotations aléatoires à notre ensemble de test selon un axe à la fois. Les angles de rotations sont choisis au hasard entre 0 et $2\pi$.

Pour la rotation selon l'axe X, tableau 7, le score F1 global diminue de 0.744 à 0,669 (-7.5%). De plus, le réseau n'arrive plus à reconnaître les poteaux et les

| | | Vérité terrain (annotations) | | | | |
|---|---|---|---|---|---|---|
| | | *arbre (75)* | *voiture (39)* | *signalisation (8)* | *poteau (23)* | *piéton (15)* |
| **Classification** | *"arbre"* | 58 | 2 | 0 | 4 | 0 |
| | *"voiture"* | 3 | 34 | 0 | 0 | 0 |
| | *"signalisation"* | 1 | 0 | 4 | 18 | 2 |
| | *"poteau"* | 0 | 0 | 1 | 1 | 0 |
| | *"piéton"* | 0 | 0 | 3 | 0 | 10 |
| | *"bâtiment"* | 6 | 3 | 0 | 0 | 3 |
| | *"bruit"* | 7 | 0 | 0 | 0 | 0 |
| | **F mesure** | 0.835 | 0.895 | 0.242 | 0.080 | 0.714 |

TABLE 4 – Matrice de confusion pour l'augmentation virtuelle avec k = 5. F mesure globale : 0.669

| | | Vérité terrain (annotations) | | | | |
|---|---|---|---|---|---|---|
| | | *arbre (75)* | *voiture (39)* | *signalisation (8)* | *poteau (23)* | *piéton (15)* |
| **Classification** | *"arbre"* | 45 | 1 | 1 | 2 | 3 |
| | *"voiture"* | 0 | 12 | 0 | 1 | 1 |
| | *"signalisation"* | 0 | 0 | 5 | 14 | 0 |
| | *"poteau"* | 0 | 0 | 0 | 1 | 0 |
| | *"piéton"* | 0 | 0 | 2 | 0 | 8 |
| | *"bâtiment"* | 1 | 0 | 0 | 1 | 1 |
| | *"bruit"* | 29 | 26 | 0 | 4 | 2 |
| | **F mesure** | 0.709 | 0.453 | 0.370 | 0.083 | 0.640 |

TABLE 5 – Matrice de confusion pour l'augmentation virtuelle avec k = 10. F mesure globale : 0.444

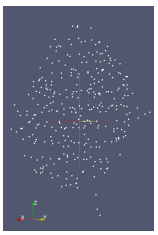


FIGURE 3 – Repère géométrique utilisé pour les nuages de points

piétons, les deux classes ayant un score F1 nul. On peut également noter une légère augmentation pour les classes voiture et signalisation, respectivement de 0.904 à 0.947 (+4.3%) et de 0.267 à 0.314 (+4.7%). La rotation selon l'axe Y, tableau 8 entraîne une légère diminution du score F1 global, de 0.744 à 0.731 (-1.3%), identique pour les classes voiture et arbre, respectivement de 0.896 à 0.890 (-0.6%) et 0.904 à 0.873 (-3.1%). La diminution est plus nette pour la classe piéton qui passe de 0.828 à 0.667 (-16.1%). Il y a également une augmentation pour les classes poteaux et signalisation, respectivement de 0.074 à 0.149 (+7.5%) et de 0.267 à 0.343 (+7.6%). Pour la rotation selon Z, tableau 9, on observe que le score F1 global passe de 0.744 à 0.706 (-3.8%). Les classes impactées par la diminution sont les classes voiture, arbre et piéton qui descendent respectivement à: 0.842 (de 0.904 : -6.2%), 0.730 (de 0.896 : -16.6%) et 0.800 (de 0.828 : -2.8%). Il y a également une augmentation pour les classes

| | | Vérité terrain (annotations) | | | | |
|---|---|---|---|---|---|---|
| | | *abre (75)* | *voiture (39)* | *signalisation (8)* | *poteau (23)* | *piéton (15)* |
| Classification | *"arbre"* | 66 | 7 | 0 | 1 | 2 |
| | *"voiture"* | 0 | 12 | 0 | 0 | 1 |
| | *"signalisation"* | 1 | 0 | 6 | 16 | 0 |
| | *"poteau"* | 0 | 0 | 0 | 1 | 0 |
| | *"piéton"* | 0 | 0 | 2 | 0 | 10 |
| | *"bâtiment"* | 1 | 0 | 0 | 0 | 2 |
| | *"bruit"* | 7 | 20 | 0 | 5 | 0 |
| | **F mesure** | 0.874 | 0.462 | 0.387 | 0.083 | 0.740 |

TABLE 6 – Matrice de confusion pour l'augmentation virtuelle avec k = 50. F mesure globale : 0.578

poteau et signalisation, respectivement de 0.074 à 0.286 (+21.2%) et de 0.267 à 0.412 (+14.5%).

Nous pouvons déduire des résultats que le réseau n'est pas invariant par rapport à l'orientation des objets. Nous supposons que l'architecture PointNet prend en compte l'orientation des formes lors de leur apprentissage. Néanmoins, nous ne savons pas à quel point PointNet est robuste aux petites rotations.

| | | Vérité terrain (annotations) | | | | |
|---|---|---|---|---|---|---|
| | | *arbre (75)* | *voiture (39)* | *signalisation (8)* | *poteau (23)* | *piéton (15)* |
| Classification | *"arbre"* | 63 | 1 | 0 | 2 | 0 |
| | *"voiture"* | 1 | 36 | 0 | 0 | 0 |
| | *"signalisation"* | 2 | 0 | 8 | 19 | 13 |
| | *"poteau"* | 0 | 0 | 0 | 0 | 0 |
| | *"piéton"* | 1 | 0 | 0 | 0 | 0 |
| | *"bâtiment"* | 4 | 0 | 0 | 2 | 0 |
| | *"bruit"* | 4 | 2 | 0 | 0 | 2 |
| | **F mesure** | 0.894 | 0.947 | 0.314 | 0.000 | 0.000 |

TABLE 7 – Matrice de confusion pour une rotation selon X. F mesure globale : 0.669

| | | Vérité terrain (annotations | | | | |
|---|---|---|---|---|---|---|
| | | *arbre (75)* | *voiture (39)* | *signalisation (8)* | *poteau (23)* | *piéton (15)* |
| Classification | *"arbre"* | 69 | 5 | 0 | 6 | 0 |
| | *"voiture"* | 1 | 31 | 0 | 0 | 0 |
| | *"signalisation"* | 2 | 0 | 6 | 14 | 5 |
| | *"poteau"* | 0 | 0 | 1 | 2 | 0 |
| | *"piéton"* | 2 | 0 | 1 | 0 | 9 |
| | *"bâtiment"* | 1 | 0 | 0 | 1 | 0 |
| | *"bruit"* | 0 | 3 | 0 | 0 | 1 |
| | **F mesure** | 0.890 | 0.873 | 0.343 | 0.149 | 0.667 |

TABLE 8 – Matrice de confusion pour une rotation selon Y. F mesure globale : 0.731

#### *4.3.5. Taille des nuages de points*

Dans l'expérience témoin, nous avons échantillonné toutes nos données à 512 points, en les réduisant parfois de 300 000 à 512 points (ce qui était le cas pour certains arbres) et parfois de 200 à 512 points (cela a été fait pour certains piétons). Afin d'évaluer l'influence du nombre de points, nous avons fait le même échantillonnage mais en remplaçant la taille final par 2048 points au lieu de 512.

| | | Vérité terrain (annotations) | | | | |
|---|---|---|---|---|---|---|
| | | *arbre (75)* | *voiture (39)* | *signalisation (8)* | *poteau (23)* | *piéton (15)* |
| Classification | *"arbre"* | 69 | 13 | 0 | 7 | 0 |
| | *"voiture"* | 1 | 23 | 0 | 0 | 0 |
| | *"signalisation"* | 4 | 0 | 7 | 11 | 4 |
| | *"poteau"* | 0 | 0 | 1 | 4 | 0 |
| | *"piéton"* | 0 | 0 | 0 | 0 | 10 |
| | *"bâtiment"* | 1 | 1 | 0 | 1 | 0 |
| | *"bruit"* | 0 | 2 | 0 | 0 | 1 |
| | **F mesure** | 0.842 | 0.730 | 0.412 | 0.286 | 0.800 |

TABLE 9 – Matrice de confusion pour une rotation selon Z. F mesure globale : 0.706

Nous avons donc fixé la taille des nuages à 2048 points (le maximum recommandé par les auteurs de PointNet) pour nos deux jeux de données et réalisé une fois de plus l'expérience de classification.

Les résultats, dans le tableau 10, montre que le score F1 global augmentant légèrement de 0.744 à 0.763 (+1.9%). Certains scores montent comme pour les classes arbre : 0.935 (à partir de 0.896: +3.9%), signalisation : 0.385 (à partir de 0.267: +11.8%) et poteau : 0.080 (à partir de 0.074 : +0.6%). D'autres descendent comme pour les classes voiture : 0.886 (à partir de 0.904 : -1.8%) et piéton : 0.813 (à partir de 0.828: -1.8%). Cependant, ces variations restent faibles.

L'interprétation globale de ces résultats est que le nombre de points a peu d'impact sur les prédictions du réseau. C'est la même conclusion que les auteurs de Charles *et al.* (2017) ont tiré dans leurs expériences. Une suite intéressante à cette expérience serait de déterminer à quel point il est possible de sous échantillonner les nuages de points avant de constater une diminution significative des scores F1.

| | | Vérité terrain (annotations) | | | | |
|---|---|---|---|---|---|---|
| | | *arbre (75)* | *voiture (39)* | *signalisation (8)* | *poteau (23)* | *piéton (15)* |
| Classification | *"arbre"* | 72 | 3 | 0 | 4 | 0 |
| | *"voiture"* | 0 | 31 | 0 | 0 | 0 |
| | *"signalisation"* | 1 | 0 | 5 | 10 | 2 |
| | *"poteau"* | 0 | 0 | 1 | 1 | 0 |
| | *"piéton"* | 2 | 0 | 2 | 0 | 13 |
| | *"bâtiment"* | 0 | 4 | 0 | 2 | 0 |
| | *"bruit"* | 0 | 1 | 0 | 6 | 0 |
| | **F mesure** | 0.935 | 0.886 | 0.385 | 0.080 | 0.813 |

TABLE 10 – Matrice de confusion avec une taille de 2048 points. F mesure globale : 0.763

### *4.3.6. Discussion*

Nous pouvons conclure de nos expériences que la classification d'objets urbains à partir de nuages de points LiDAR, donnent des résultats satisfaisants qui ne peuvent que s'améliorer à mesure que nous obtenons plus de données pour entraîner les réseaux. Même si d'autres pistes peuvent être explorées afin

améliorer davantage les performances, telle une architecture plus complexe ou une augmentation virtuelle fonctionnelle, le meilleur moyen reste encore d'ajouter encore plus de données à l'ensemble d'entraînement.

Une autre remarque que l'on peut faire est que dans certains exemples de nuages de points où le réseau fait une erreur, différencier les versions sous-échantillonnées des nuages de points semble très difficile. Nous supposons donc que certaines des erreurs pourraient provenir de la dégradation des nuages de points d'origine par l'algorithme "voxel-grid" plutôt que de la capacité du réseau à reconnaître les formes 3D.

### *4.4. Conclusion*

Dans cet article, nous avons présenté une série d'expériences sur la classification des objets urbains en 3D à l'aide de PointNet. Ces expériences ont montré que le meilleur moyen d'améliorer le score F1 est d'avoir un ensemble d'entraînement plus large. Nous sommes également arrivés à la conclusion que le réseau n'est pas invariant par rapport à l'orientation des objets et que le réseau confond les panneaux de signalisation et les poteaux entre eux, tout en arrivant à les différencier des autres classes.

Nous avons également souligné que l'un des facteurs limitant ces expériences pourrait être la dégradation des nuages de points par le filtre "voxel-grid". Un moyen d'améliorer le réseau de classification serait alors de lui permettre de prendre en entrée les nuages de points LiDAR d'origine. Cette amélioration est également liée à notre premier objectif qui est la détection des objets dans des grandes scènes 3D.

À l'avenir, nous prévoyons d'expérimenter avec des nuages de points synthétisés représentant des objets urbains simples tels que des poteaux ou des arbres et voir si cela a le même effet que l'ajout de données numérisées réelles. Nous souhaitons également aborder la question de la détection d'objets urbains dans les grands nuages de points 3D.

## Bibliographie

(s. d.).

Anguelov D., Dulong C., Filip D., Früh C., Lafon S., Lyon R. *et al.* (2010). Google street view: Capturing the world at street level. *Computer*, vol. 43, p. 32-38.

Charles R. Q., Su H., Kaichun M., Guibas L. J. (2017, July). PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *2017 ieee conference on computer vision and pattern recognition (cvpr)*, p. 77-85.

Engelmann F., Kontogianni T., Hermans A., Leibe B. (2017). Exploring spatial context for 3d semantic segmentation of point clouds. *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, p. 716-724.

Johns E., Leutenegger S., Davison A. J. (2016). Pairwise decomposition of image sequences for active multi-view recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, p. 3813-3822.

Kanezaki A., Matsushita Y., Nishida Y. (2018). Rotationnet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints. In *Proceedings of ieee international conference on computer vision and pattern recognition (cvpr).*

Klokov R., Lempitsky V. S. (2017). Escape from cells: Deep kd-networks for the recognition of 3d point cloud models. *2017 IEEE International Conference on Computer Vision (ICCV)*, p. 863-872.

Lai K., Fox D. (2010, July). Object Recognition in 3D Point Clouds Using Web Data and Domain Adaptation. *International Journal of Robotics Research*, vol. 29, n° 8, p. 1019–1037.

LeCun Y., Bengio Y., Hinton G. (2015, mai). Deep learning. *Nature*, vol. 521, n° 7553, p. 436–444.

Maturana D., Scherer S. (2015, Sept). VoxNet: A 3D Convolutional Neural Network for real-time object recognition. In *2015 ieee/rsj international conference on intelligent robots and systems (iros)*, p. 922-928.

Qi C. R., Yi L., Su H., Guibas L. J. (2017). Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Nips.*

Quadros A., Underwood J., Douillard B. (2012, May 14-18). An Occlusion-aware Feature for Range Images. In *Robotics and automation, 2012. icra'12. ieee international conference on.*

Ren S., He K., Girshick R. B., Sun J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, p. 1137-1149.

Riegler G., Ulusoy A. O., Geiger A. (2017). Octnet: Learning deep 3d representations at high resolutions. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, p. 6620-6629.

Roynard X., Deschaud J.-E., Goulette F. (2018). Paris-lille-3d: A large and high-quality ground-truth urban point cloud dataset for automatic segmentation and classification. *The International Journal of Robotics Research*, vol. 37, n° 6, p. 545-557.

Russakovsky O., Deng J., Su H., Krause J., Satheesh S., Ma S. *et al.* (2014). Imagenet large scale visual recognition challenge. *CoRR*, vol. abs/1409.0575.

Serna A., Marcotegui B., Goulette F., Deschaud J.-E. (2014). Paris-rue-Madame Database - A 3D Mobile Laser Scanner Dataset for Benchmarking Urban Detection, Segmentation and Classification Methods. In *Icpram.*

Zegaoui Y., Chaumont M., Subsol G., Borianne P., Derras M. (2018). First experiments of deep-learning on lidar point clouds for classification of urban object. *CFPT*.

# Télédétection et Deep Learning : détection automatisée des modes d'occupation des sols en Nouvelle-Calédonie

**Guillaume Rousset** [1], **Morgan Mangeas** [1], **Dominique Simpelaere** [2], **Marc Despinoy** [1]

1. *ESPACE-DEV, Institut de Recherche pour le Développement (IRD) Nouméa, Nouvelle-Calédonie*

   *guillaume.rousset@ird.fr*

2. *ISEA, Université de Nouvelle-Calédonie, Nouméa, Nouvelle-Calédonie*

RÉSUMÉ. *Le MOS (Mode d'Occupation des Sols), un projet initié par l'OEIL (Observatoire de l'Environnement de Nouvelle-Calédonie), utilise des données optiques à très haute résolution spatiale pour créer une cartographie complète de l'occupation du sol par photo-interprétation sur l'ensemble de la Province Sud de Nouvelle-Calédonie. Il s'agit ici d'adapter les techniques du Deep Learning à des problématiques de télédétection pour l'analyse des changements de modes d'occupation du sol.*

ABSTRACT. *MOS (Mode d'Occupation des Sols) A project initiated by OEIL (Observatoire de l'Environnement de Nouvelle-Calédonie), uses very high resolution optical data to create a complete mapping of land use by photo-interpretation over the entire Southern Province of New Caledonia. The aim is to adapt Deep Learning techniques to remote sensing problems for the analysis of changes in land use patterns.*

MOTS-CLÉS : *Deep Learning, télédétection, occupation du sol, indices, réseau de neurones convolutionnels, classification, multi-label*

KEYWORDS: *Deep Learning, remote sensing, land cover, indices, convolutional neural network, classification, multi-label*

SAGEO'2018 - Montpellier, 6 novembre 2018

## 1. Introduction

Le contexte environnemental de la Nouvelle-Calédonie (NC) est unique du fait de son endémicité et son exceptionnelle biodiversité (Morat, 1993). La protection de l'environnement en NC représente un enjeu considérable en considérant les effets du changement climatique (intensité des cyclones, fréquence et intensité des pluies, élévation du niveau de la mer...) et des pressions anthropiques (urbanisation, incendies...). Le suivi automatique de son écosystème devient de plus en plus une priorité pour le pays. Dans le même temps, les données satellitaires sont de plus en plus accessibles à moindre coût et offrent des informations permettant la caractérisation fine de l'évolution des milieux. Ces séries temporelles enrichissent des bases de données qu'il convient de traiter avec des méthodes adaptées.
Le Deep Learning, apprentissage profond en français, est une technique à la croisée de l'informatique et de la modélisation mathématique (LeCun *et al.*, 1998) qui se situe actuellement au plus haut dans les classements des compétitions dans le domaine de la reconnaissance d'objets, avec un taux de reconnaissance proche des performances humaines (Schmidhuber, 2014).
En tenant compte des contraintes liées au traitement de l'imagerie satellitaire et des problèmes liés aux algorithmes d'apprentissage ; l'objectif est d'adapter les techniques du Deep Learning à des problématiques de télédétection pour à terme, surveiller automatiquement et efficacement l'évolution de l'environnement d'une île tropicale haute possédant de nombreux milieux hétérogènes fragmentés (Lambin *et al.*, 2003). Ces méthodes, une fois développées, seront aisément adaptables à de nombreux domaines comme la détection du corail, l'érosion des sol et les effets de l'urbanisme sur l'environnement.

Dans cette communication nous nous focalisons sur la détection des Modes d'Occupation des Sols (MOS). L'idée est d'exploiter des données obtenues par des moyens semi-automatiques et par validation via photo-interprétation, pour alimenter des réseaux de neurones denses et apprendre à classifier les différentes classes du MOS d'une nomenclature établie (Table 1). Cette tâche est relativement complexe dans le cadre de milieux hétérogènes rencontrés dans les zones tropicales et les techniques classiques de machine learning comme SVM ou XGboost ne fournissent pas de résultats satisfaisants.
Ces travaux s'inscrivent dans le cadre d'une thèse de 3 ans, financée par une bourse de la Province Sud de Nouvelle-Calédonie. L'encadrement est porté par l'Université de Nouvelle-Calédonie par le biais de l'école doctorale du Pacifique, et par l'Institut de Recherche pour le Développement de Nouméa, où l'UMR Espace-DEV est le laboratoire d'accueil.

## 2. Données et Méthodes

### *2.1. Le MOS et les données satellites*

L'Observatoire de l'Environnement de Nouvelle-Calédonie (OEIL) a initié il y a quelques années un projet visant à fournir une cartographie du MOS en exploitant des données optiques à très haute résolution disponible sur l'ensemble de la Province Sud de la Nouvelle-Calédonie ($\sim$ 7000 $km^2$). Le MOS est une base de données géographique représentant de façon détaillée à l'aide d'une nomenclature hiérarchisée la couverture biophysique du territoire. Une telle nomenclature est mise en place pour décrire au mieux chaque classe d'occupation du sol (ex : zones artificialisées, zones agricoles, zones humides, forêts, etc.). Plusieurs classes d'environnement et d'aménagement du territoire sont extraites pour créer des indicateurs directement dérivés du MOS tels que le degré d'artificialisation, la fragmentation des habitats, ou indirectement dérivés tels que la sensibilité des sols à l'érosion, les risque de départ de feu...
Une première classification du sol de la Province Sud de Nouvelle-Calédonie a été réalisée à partir d'images RapidEye de 2010, et une deuxième cartographie d'occupation du sol à partir d'images SPOT6 de 2014. Les MOS de 2010 et 2014 ont été produits avec pour objectif une précision géographique de 1 hectare.

TABLE *1. Nomenclature hiérarchisées du Mode d'Occupation du Sol de Nouvelle-Calédonie*

| L1 | L2 | Description | C2 |
|---|---|---|---|
| 1 | | Territoires artificialisés | |
| | 11 | Zones urbanisées | 1 |
| | 12 | Zones industrielles ou commerciales et équipements | 2 |
| | 13 | Décharges, chantiers, extraction de matériaux, mines | 3 |
| | 14 | Réseaux de communication | 4 |
| 2 | | Formations naturelles et agricoles | |
| | 21 | Strate arborée | 5 |
| | 22 | Strate arbustive | 6 |
| | 23 | Espaces ouverts, sans ou avec peu de végétation | 7 |
| | 24 | Roches | 8 |
| | 25 | Zones incendiées | 9 |
| 3 | | Zones humides | 10 |
| 4 | | Surfaces en eau | 11 |

Nous disposons, pour alimenter les architectures de neurones denses, des données satellites citées précédemment :

– Le capteur RapidEye pour les images de 2010

  - 5 canaux sont utilisés (bleu, vert, rouge, red-egde et proche infrarouge)
  - résolution de 5 mètres
  - surface sur l'ensemble de la Province Sud
- Le capteur SPOT6 pour les images de 2014
  - 4 canaux sont utilisés (bleu, vert, rouge et proche infrarouge)
  - résolution de 1,5 mètres
  - surface sur l'ensemble de la Province Sud

### *2.2. Méthode classique de détection*

La méthode utilisée pour fournir des résultats de référence consiste à calculer des produits dérivés des canaux brutes des données de télédétection et d'appliquer une méthode de classification non linéaire. Avec G représentant le canal vert ("Green"), R le canal rouge et PIR le proche infrarouge, les indices de végétation utilisés sont les suivants : Le NDVI (Normalized Difference Vegetation Index), le MNDWI (Modified Normalized Difference Water Index) (Han-Qui, 2005) et le MSAVI (Modified Soil-Adjusted Vegetation index) (Qi *et al.*, 1994).

$$NDVI = \frac{(PIR - R)}{(PIR + R)} \tag{1}$$

$$MNDWI = \frac{(G - PIR)}{(G + PIR)} \tag{2}$$

$$MSAVI = \frac{2 \times PIR + 1 - \sqrt{(2 \times PIR + 1)^2 - 8 \times (PIR - R)}}{2} \tag{3}$$

Ces trois indices de végétation sont robustes aux variations radiométriques (différence entre les dates d'acquisition, les angles d'acquisition, ...). Ils permettent à eux trois de caractériser les différents états de la végétation : la senescence, les stress notamment hydriques, et les rapports de signaux sol/végétation.

Pour chacun de ces indices, des calculs de textures d'haralick ont été appliqués. Les filtres de textures représentent l'arrangement spatial des objets dans l'image (Majdoulayne, 2009). Les filtres les plus souvent utilisés sont dérivés de la matrice de co-occurrence qui dénombre les paires de pixels jouissant d'une même propriété spectrale, d'une périodicité ainsi que d'une directivité des textures (M. *et al.*, 1973 ; Haralick, 1979). Les textures utilisées ici sont : "homogeneity", "contrast", "dissimilarity", "entropy" et "correlation", en utilisant une fenêtre glissante de taille 31x31.
L'ensemble des données sont ensuite fournies en entrée d'un modèle du type xgboost (Chen, Guestrin, 2016). Le modèle xgboost a été préféré à des méthodes du type SVM (Cortes, Vapnik, 1995) car elle nécessite peu de ressources de calcul pour converger et permet de traiter un grand volume de données en un

temps de calcul raisonnable. Le modèle a été developpé en utilisant le langage R (R Core Team, 2018) et la librairie xgboost (Chen *et al.*, 2018).

### *2.3. Architecture de Deep Learning*

De nombreuses architectures Deep Learning existent (Cheng *et al.*, 2017 ; Zhang *et al.*, 2016) et sont applicables à la problématique d'occupation du sol. Une en particulière a été utilisée, l'architecture SegNet (Badrinarayanan *et al.*, 2015) présentée dans la figure 1. Développée par l'Université de Cambridge, cette architecture du type "encodeur-décodeur" classifie automatiquement les pixels d'une image en plusieurs classes. Utilisée principalement pour la compréhension d'images de trafic automobile, distinguant la route, les voitures, le marquage au sol et son environnement, l'architecture a été réadaptée pour la télédétection de l'occupation du sol. Cette architecture a été choisie pour sa capacité à segmenter une image et classifier chaque pixel la constituant selon plusieurs classes. De plus, la problématique spatiale des pixels voisins est gérée entièrement par l'utilisation de réseaux de neurones convolutionnels (LeCun *et al.*, 1998).

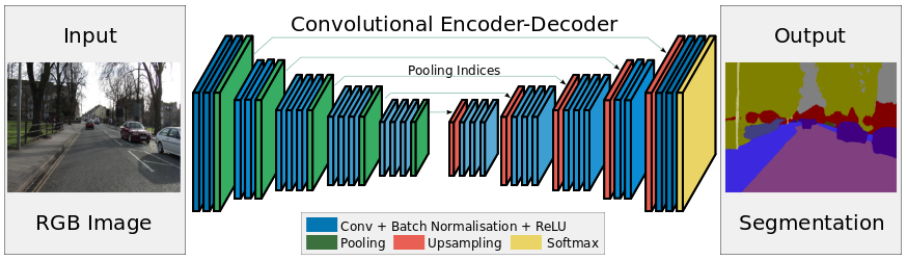



FIGURE *1. L'architecture Segnet. source : http://mi.eng.cam.ac.uk/projects/segnet/*

## 3. Résultats et discussions

L'architecture SegNet et la méthode xgboost ont été appliquées à l'occupation du sol (Table 2) dont on peut voir une application directe, pour l'architecture SegNet, à un secteur minier en Nouvelle-Calédonie dans la figure 2. La couleur des pixels utilisés est visible dans la nomenclature présentée précédemment (Table 1).
Avant d'obtenir ce résultat, un soin particulier a été pris pour la constitution de la base de données. La création du jeu de données est une étape importante qui influe grandement sur les résultats finaux de l'apprentissage des architectures Deep Learning. En effet, que le jeu de données soit équitablement réparti

ou non, la résolution des images, leur découpage, et la sur-représentation/sous-représentation des données ont un impact (Buda *et al.*, 2017).

Même si l'architecture SegNet arrive à labelliser correctement la majorité des pixels, achevant un meilleur résultat que la méthode xgboost, des différences importantes sont notées. Figure 2, La masse d'eau trouvée au cœur des mines et identifiée convenablement par notre architecture malgré la mauvaise labellisation du jeu de données initiale. L'intérieur des classes est plus précis, par exemple avec la détection d'étendues de végétation au sein d'un groupe de pixels appartenant aux mines dans le MOS. Ces différences s'expliquent par les règles initiales du MOS négligeant les espaces inférieurs à un hectare les intégrant directement dans le groupe adjacent majoritaire.
Nous observons aussi les problèmes de l'architecture à prédire les zones couvertes par des nuages ou des ombres car non pris en compte lors de l'apprentissage en lui-même. Enfin nous observons la faible détection des routes par l'architecture de Deep Learning. Ce problème est lié directement au jeu de données lui-même. En effet il a été remarqué que la plupart des routes ne sont pas référencés fidèlement par la classification par photo-interprétation. Cette classification se base sur une donnée exogène pour intégrer le réseau de routier de Nouvelle-Calédonie qui ne représente que les axes routiers principaux et ainsi labellisant la majorité des routes de Nouvelle-Calédonie dans d'autres classes. De plus à cela s'ajoute un problème de décalage entre les données du MOS et l'image satellite SPOT6 utilisée. Pour une classe aussi précise que les routes, un décalage de 2 ou 3 pixels perturbe considérablement l'apprentissage de la classe route.

TABLE *2. Résultats des différentes méthodes de machine learning sur le jeu de validation (en %) par rapport au différentes classes de la nomenclature*

| Classe | xgboost | SegNet |
|---|---|---|
| Zones urbanisées | 69,30 | 79,58 |
| Zones industrielles | 52,71 | 81,70 |
| Décharges, chantiers | 87,32 | 93,22 |
| Réseaux de communication | 16,84 | 20,46 |
| Strate arborée | 69,39 | 72,17 |
| Strate arbustive | 61,51 | 64,54 |
| Espaces ouverts | 38,31 | 58,37 |
| Roches | 52,21 | 81,08 |
| Zones incendiées | 59,40 | 78,63 |
| Zones humides | 86,22 | 93,89 |
| Surfaces en eau | 80,65 | 82,06 |
| Global | 61,71 | 73,24 |

## 4. Conclusion

Certains problèmes comme la prise en compte des ombres et des nuages peuvent être résolu par la mise en point d'un filtre automatique masquant ces


FIGURE *2. Résultat de l'architecture SegNet (en bas) après apprentissage sur l'occupation du sol initial (en haut) sur une zone minière de Nouvelle-Calédonie.*

zones ou en incluant ces classes dans la classification faites par notre architecture. Pour résoudre les problèmes de décalage pour les données à haute résolution, une orthonormalisation des données satellites à partir d'ortho-photos,

c'est-à-dire un recalage automatique d'un pixel par rapport au pixel d'une image de façon automatique, est envisagé à l'aide d'une architecture Deep Learning (Han *et al.*, 2015 ; Altwaijry *et al.*, 2016). Cette solution s'inspire directement de l'algorithme SIFT (Lowe, 2004) pouvant reconnaître un objet et sa disposition d'une image à une autre. Cette implémentation est un problème complexe en télédétection à cause des changements d'illumination au cours du temps et des différences liées à l'utilisation de différent capteurs. L'intégration dans une même architecture de capteurs de différentes natures (optique, radar ou lidar) et/ou de résolutions différentes (problématique multi-capteurs/multi-résolutions) représente une piste de recherche très intéressante pour améliorer les performances de détection.
Dans le cadre de cette thèse, une application d'autres architectures de Deep Learning, plus simples, est en cours d'implémentation. La création automatique et systématique à partir de données libres et gratuites, avec les données Sentinel-2, qui est l'un des objectifs principaux de cette thèse est aussi en cours de travail. D'autres applications sont également envisagées telles que la détection du palmier Babaçu via des images simulées par le logiciel DART (Gastellu-Etchegorry *et al.*, 2015) ou encore la détection de typologies urbaines au Brésil.

## Bibliographie

Altwaijry H., Veit A., Belongie S. (2016). Learning to Detect and Match Keypoints with Deep Architectures. *British Machine Vision Conference (BMVC)*, p. 12.

Badrinarayanan V., Kendall A., Cipolla R. (2015). SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *Cvpr 2015*, p. 5. Consulté sur http://mi.eng.cam.ac.uk/projects/segnet/

Buda M., Maki A., Mazurowski M. A. (2017). A systematic study of the class imbalance problem in convolutional neural networks. , p. 1–23.

Chen T., Guestrin C. (2016). Xgboost: A scalable tree boosting system. *CoRR*, vol. abs/1603.02754. Consulté sur http://arxiv.org/abs/1603.02754

Chen T., He T., Benesty M., Khotilovich V., Tang Y., Cho H. *et al.* (2018). xgboost: Extreme gradient boosting Manuel de logiciel. Consulté sur https://CRAN.R-project.org/package=xgboost (R package version 0.71.2)

Cheng G., Han J., Lu X. (2017). Remote Sensing Image Scene Classification: Benchmark and State of the Art. *CoRR*, vol. abs/1703.0.

Cortes C., Vapnik V. (1995, 01 Sep). Support-vector networks. *Machine Learning*, vol. 20, n° 3, p. 273–297. Consulté sur https://doi.org/10.1007/BF00994018

Gastellu-Etchegorry J.-P., Yin T., Lauret N., Cajgfinger T., Gregoire T., Grau E. *et al.* (2015). Discrete anisotropic radiative transfer (dart 5) for modeling airborne and satellite spectroradiometer and lidar acquisitions of natural and urban landscapes. *Remote Sensing*, vol. 7, n° 2, p. 1667–1701.

Han X., Leung T., Jia Y., Sukthankar R., Berg A. C. (2015). MatchNet: Unifying feature and metric learning for patch-based matching. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 07-12-June, p. 3279–3286.

Han-Qui X. (2005). A study on information extraction of water body with the modified normalized difference water index (mndwi). *Journal of Remote Sensing*, vol. 9, n° 5, p. 589-595.

Haralick R. M. (1979). Statistical and structural approachs to texture. , vol. 67, n° 5, p. 786-804.

Lambin E. F., Geist H. J., Lepers E. (2003). Dynamics of land-use and land-cover change in tropical regions. *Annual Review of Environment and Resources*, vol. 28, n° 1, p. 205-241.

LeCun Y., Bottou L., Bengio Y., Haffner P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, vol. 86, n° 11, p. 2278–2323.

Lowe D. G. (2004). Distinctive image features from scale invariant keypoints. *International Journal of Computer Vision*, vol. 60, p. 91–11020042.

M. H. R., K. S., I. D. (1973). Textural features for image classification. *IEEE Transactions on Systems, Man and Cybernetics*, vol. 3, n° 6, p. 610-621.

Majdoulayne H. (2009). *Extraction de caractéristiques de texture pour la classification d'image satellite*. Thèse de doctorat non publiée, Université de Toulouse.

Morat P. (1993). Our knowledge of the flora of new caledonia: Endemism and diversity in relation to vegetation types and substrates. *Biodiversity Letters*, vol. 1, n° 3/4, p. 72–81.

Qi J., Chehbouni A., Huete A., Kerr Y., Sorooshian S. (1994). A modified soil adjusted vegetation index. *Remote Sensing of Environment*, vol. 48, n° 2, p. 119 - 126. Consulté sur http://www.sciencedirect.com/science/article/pii/0034425794901341

R Core Team. (2018). R: A language and environment for statistical computing Manuel de logiciel. Vienna, Austria. Consulté sur https://www.R-project.org/

Schmidhuber J. (2014). Deep Learning in Neural Networks: An Overview. *CoRR*, vol. abs/1404.7.

Zhang L., Zhang L., Kumar V. (2016). Deep learning for Remote Sensing Data. *IEEE Geoscience and Remote Sensing Magazine*, vol. 4, n° 2, p. 22–40.