AN EXTENSIBLE AND EASY-TO-USE TOOLBOX FOR DEEP LEARNING BASED ANALYSIS OF REMOTE SENSING IMAGES

Raian Vargas Maretto, Thales Sehn Körting, Leila Maria Garcia Fonseca

National Institute for Space Research (INPE, Brazil)

ABSTRACT

Deep Learning (DL) methods are currently the state-of-the-art in Machine Learning and Pattern Recognition. In recent years, DL has been successfully applied to Remote Sensing (RS) image processing for several tasks, from pre-processing to classification. This paper presents DeepGeo, a toolbox that provides state-of-the-art DL algorithms for RS image classification and analysis. DeepGeo focuses on providing easy-to-use and extensible methods, making it easier to those RS analysts without strong programming skills. It is distributed as free and open source package and is available at `https://github.com/rvmaretto/deepgeo`.

Index Terms— Deep Learning, Convolutional Neural Networks, Remote Sensing, Semantic Segmentation

1. INTRODUCTION

With the recent growing accessibility of new generation Remote Sensing (RS) sensors, a large bulk of data has become available. Having access to an incredible amount of data have brought the opportunity to widen our ability to understand the Earth. At the same time, it turned to be impracticable the traditional non-automatic analysis, increasing the focus on the ability to automatically extract valuable information from those images.

In recent years, Deep Learning (DL) has become a hotspot in the Machine Learning and Pattern Recognition communities. It is characterized by a set of Artificial Neural Networks (ANN) composed of multiple feature extraction layers, also called Deep Neural Networks (DNN). These layers are responsible for extracting features in different levels of abstraction, starting from the raw data [1]. In these feature extraction layers, each level transforms the representation of the previous ones into a more abstract model, hierarchically combining them, and then being able to model and explore intrinsic correlations in the data [2].

In RS Image Processing, DL methods have been successfully applied for many different purposes, such as pansharpening [3] and semantic segmentation (pixelwise classification) [4, 5, 6]. Therefore, these methods has been considered by [7] as crucial for the future of RS data analysis, specially in the age of big data.

Several toolboxes are available for DL development, like TensorFlow[8], Keras [9], Theano [10] and PyTorch [11]. Although very powerful, the currently available toolboxes for DL development are hard to use by analysts without strong programming skills. Due to the complexity of the concepts involved, they demand from the analyst a strong background in Computer Science to be able to implement a DNN and perform tasks like classification and analysis.

Focusing on facilitating the access to DL techniques by RS analysts with as few lines of source code as possible, this paper presents DeepGeo toolbox. It provides configurable building blocks to perform the entire cycle of DL based analysis of RS data.

2. DEEPGEO TOOLBOX

DeepGeo is a Python toolbox that provides, as configurable building blocks, tools to perform spatial and multi-temporal DL based analysis of RS images. It integrates tools to perform the following tasks: *pre-process data; generate training, evaluation and validation datasets; train predefined DNNs; easily customize and implement new DNNs; apply DL classification based on a trained DNN; and analyse and visualize results*.

DeepGeo is distributed as a free and open source software under the terms of the GNU General Public License version 3.0 or later, running on multiple platforms, e.g., Windows, Mac OS X and Linux. The system works as a package for Python programming language, which provides a high level and easy-to-use API (Application Programming Interface). DeepGeo API was developed with focus on making it easy to perform the entire DL analysis cycle with few lines of source code, taking advantages of TensorFlow parallelism to make it easily scalable to process large amounts of data, remaining flexible and easily extensible. After defined the input data and the DNN model to be used, we consider that the cycle of DL based classification and analysis of RS data, as shown in Figure 1, have the following main steps:

1. defining the input data and DNN model;

Thanks to the World Bank and to the São Paulo Research Foundation (FAPESP) for funding through the FIP (Forest Investment Program) program and the project #2017/24086-2 respectively.

2. preprocessing input data;
3. generating training dataset;
4. training the Model;
5. evaluating training results, repeating the training step, if necessary, until having satisfactory results;
6. perform the classification;
7. visualize and analyze classification results, repeating the training step if necessary.

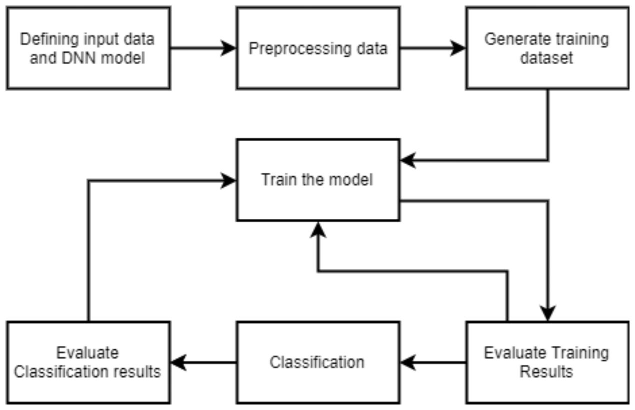
Defining input data and DNN model
Preprocessing data
Generate training dataset
Train the model
Evaluate Classification results
Classification
Evaluate Training Results

Fig. 1. Cycle for Remote Sensing image classification and analysis using Deep Learning.

In the next sub-sections, we describe, based on this cycle, the conceptual modules of DeepGeo and present some examples of how to use its functionalities.

2.1. Preprocessing Module

When dealing with RS images, having it properly prepared to input to a DNN is often a great challenge. This module provides easy ways to perform a wide range of preprocessing operations, like performing mosaics, crop images, rasterize vector layers of ground truth data and compute spectral indices.

Normalize input images is an important step for ANN, once it accelerates the training process, making the convergence faster [12]. This module also provides functions to automatically perform standardization or normalization of the input images with several strategies. The code snippet presented in Figure 2 shows the definition of a Preprocessor structure.

In RS classification tasks, the ground truth data is frequently provided as maps in vector format instead of raster. The Rasterizer type allows the user to easily convert a vector ground truth data to raster to make it possible to input it as labels to the DNN. Figure 3 presents a code snippet that rasterizes an input vector file and save it in a new raster file.

```
import deepgeo.dataset.preprocessor as prep
raster_file = "my_raster.tif"
# Define a Preprocessor for file "my_raster.tif"
preproc = prep.Preprocessor(raster_file,no_data=0)
# Compute Vegetation indices (NDVI and EVI2)
preproc.compute_indices({
    "ndvi": {"idx_b_red": 3, "idx_b_nir": 4},
    "evi2": {"idx_b_red": 3, "idx_b_nir": 4}})
# Standardize the image. Subtracts from the mean
    and then divides by the standard deviation
preproc.standardize_image("mean_std")
preproc.save_stacked_raster("output.tiff")
```

Fig. 2. Defining and using a Preprocessor.

```
import deepleeo.dataset.rasterizer as rast
# Defines input data
raster_file = "my_raster.tif"
labels_shp = "my_labels.shp"
# Defines the column in shape file containing the
    classes
class_column = "class"
# Define the classes to be rasterized
classes_of_inter = ["deforestation", "forest"]
# Defines the Rasterizer
rasterizer = rast.Rasterizer(shape_file,
                labels_shp,
                class_column=class_column,
                classes_interest=classes_of_inter)
# Rasterizes the data and save at "my_labels.tif"
rasterizer.rasterize_layer()
rasterizer.save_labeled_raster_to_gtiff("my_labels
    .tif")
```

Fig. 3. Rasterizing vector ground truth data.

2.2. Dataset Generation Module

Due to its depth and complexity, DL models are usually computationally hard to process, making it impossible to process an entire RS Image at once. Due to this limitation, it is common to split the images into smaller processing units, called chips or patches, i.e., small windows in the original image. The *Dataset Generation Module* provides a simple API to sequentially or randomly split the image and save it into a training dataset, to sequentially split images for the classification process and to reconstitute a classified image from a sequential set of classified patches.

2.3. Deep Learning Module

The *Deep Learning module* provides several DNNs models already implemented, like the U-Net [13] and the Fully Convolutional Networks (FCN) proposed by [14], and some adaptations of these networks for multi-temporal analysis. It also provides an easy way to define new models, using the predefined DeepGeo structure to train them and perform classification, without the need of large experience with TensorFlow API. The code snippet presented in Figure 4 defines a FCN and perform the training process based on a previously generated dataset.

Despite powerful, DL methods are highly prone to overfitting, being necessary a huge amount of samples and some regularization techniques to avoid it. Some RS applications, due to the difficulties to acquire huge amounts of labeled samples, are even more prone to this problem [1]. Attempting to counteract overfitting, a common regularization technique is called *data augmentation*, which artificially increase the size of the training dataset synthetically modifying existing samples. It is also important to make the model more invariant to the position of the target object in the image. This module provides operations to perform data augmentation applying on the samples different angles of rotation and flipping them, substantially increasing the number of training samples. Taking advantages of the parallelism and the structure of TensorFlow Data Input Pipeline, the data augmentation is applied while loading the images for the training process.

```
import deepgeo.networks.model_builder as mb
import deepgeo.dataset.utils as dsutils
# Datasets are stored in TFRecords
train_ds = "train_dataset.tfrecord"
test_ds = "test_dataset.tfrecord"
valid_ds = "valid_dataset.tfrecord"
model_dir = "trained_model"
test_img = "img.tif"
output = "classif.tif"
# Defines some parameters of the DNN.
params = {"epochs": 600,
          "batch_size": 20,
          "learning_rate": 0.0001,
          "data_aug_ops": ["rot90", "rot180",
                "rot270", "flip_left_right",
                "flip_up_down", "flip_transpose"]}
# Defines a FCN8s model and train it.
model = mb.ModelBuilder("fcn8s")
model.train(train_ds, test_ds, params, model_dir)
# Test in validation dataset
model.validate(valid_ds, params, model_dir)
# Perform classification
model.predict(test_img, params, model_dir, output)
```

Fig. 4. Defining and training a FCN8s model.

2.4. Visualization and Classification Analysis Module

The *Visualization and Classification Analysis module* focuses on providing tools to visualize and analyze the quality of the input dataset and the classification results. It provides several metrics to measure the classification accuracy, like pixel-wise accuracy, Receiver Operating Characteristics (ROC) curve, F1-score and cross-entropy. In addition, to make it possible to visually analyze the quality of the input dataset and the classified labels, this module provides tools to easily plot the image, ground truth labels, classified labels, histograms, confusion matrices, and patches distribution in the original image.

3. EXPERIMENTAL RESULTS: MAPPING DEFORESTED AREAS IN BRAZILIAN AMAZON

In this section we present a case study to illustrate the effective use of DeepGeo toolbox. The focus here is not to obtain a high accurate classification model, but to exemplify the use of DeepGeo in a practical application. We used the system to produce a classification of deforested areas in an small area of the Brazilian Amazon for the year 2017, taking the PRODES data [15] as ground truth to train the Fully Convolutional Network FCN8s, proposed by [14]. PRODES is a program developed by INPE [1] that provides a large database of yearly maps of deforested areas in the Brazilian Amazon since 2000.

The area to be classified corresponds to one scene of Landsat 8 OLI [2] sensor. Figure 5 shows this image in (a), the ground truth labels in (b) and the classification results in (c). Preprocessing steps were made through the source code presented in Figure 2. The ground truth data was rasterized through the source code presented in Figure 3. Based on the input raster, 2000 patches were randomly generated as the input training dataset. Data augmentation was performed rotating and flipping the generated patches, thus totaling 14000 input patches for the training process. The training process was performed through the source code shown in Figure 4.

Some advantages of using this system can be pointed out, as the facility to perform preprocessing steps in the input image, standardizing it, computing spectral indices, generating training and classification datasets, perform data augmentation and classification without much programming expertise.

4. CONCLUSIONS AND FUTURE WORKS

DeepGeo toolbox was presented in this paper. By providing DL methods, preprocessing, dataset generation and result analysis functionalities as easy-to-use building boxes, it allows to easily perform the entire cycle of DL based analysis on RS data. In this way, DeepGeo can make DL methods more accessible to those RS analysts without strong background in computer science. It also provides easy ways to extend the current functionalities by adding new strategies for each step of the analysis cycle. Besides that, taking advantages of the flexibility and expressiveness of Python programming language, DeepGeo provides easy ways to be extended and coupled to another tools.

The system can also deal with several types of geospatial data formats for raster and vector data, that can be used as ground truth. For now, it only provides tools based on convolutional encoder-decoders for semantic segmentation. Future works includes extending DeepGeo to provide more DL approaches and applications, like Recurrent Neural Networks for Time Series Analysis or Autoencoders for pansharpening.

[1]National Institute for Space Research, Brazil
[2]Operational Land Imager

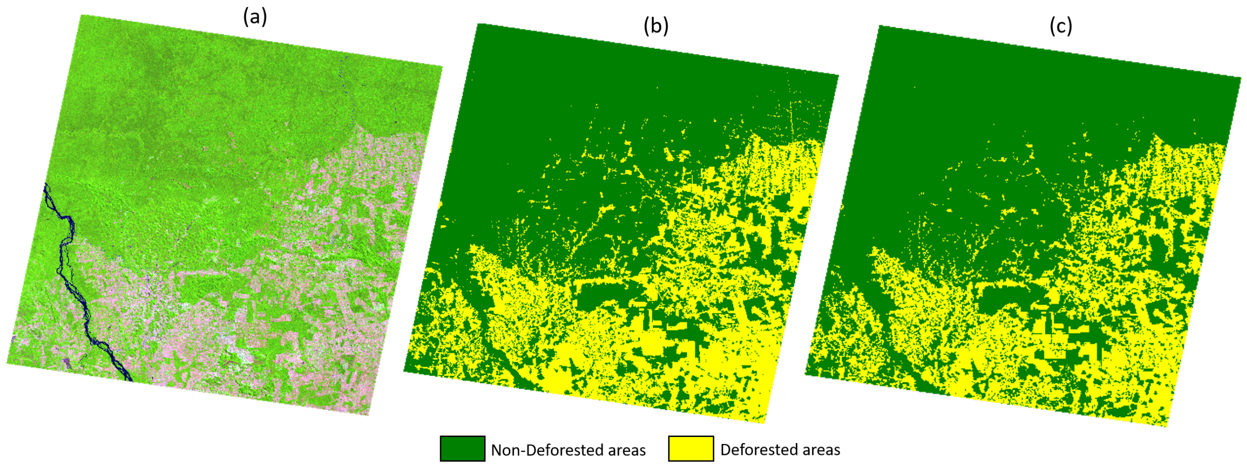

Fig. 5. Classification of deforested areas: a) Input Image; b) Ground truth; c) Classification output

5. REFERENCES

[1] X. X. Zhu, D. Tuia, L. Mou, et al., "Deep Learning in Remote Sensing: A Comprehensive Review and List of Resources," *IEEE Geoscience and Remote Sensing Magazine*, vol. 5, no. 4, pp. 8–36, 2017.

[2] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, 2015.

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

[4] G. Fu, C. Liu, R. Zhou, T. Sun, and Q. Zhang, "Classification for high resolution remote sensing imagery using a fully convolutional network," *Remote Sensing*, vol. 9, no. 5, pp. 1–21, 2017.

[5] R. Kemker, C. Salvaggio, and C. Kanan, "Algorithms for semantic segmentation of multispectral remote sensing imagery using deep learning," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 145, no. March, pp. 60–77, 2018.

[6] M. Zhai, Z. Bessinger, S. Workman, and N. Jacobs, "Predicting ground-level scene layout from aerial imagery," *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-January, pp. 4132–4140, 2017.

[7] L. Zhang, L. Zhang, and V. Kumar, "Deep learning for Remote Sensing Data: A technical tutorial on the state of the art," *IEEE Geoscience and Remote Sensing Magazine*, vol. 4, no. 2, pp. 22–40, 2016.

[8] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, et al., "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems," *arXiv preprint arXiv:1603.04467*, 2016.

[9] F. Chollet et al., "Keras," 2015.

[10] Theano Development Team, "Theano: A Python framework for fast computation of mathematical expressions," *arXiv e-prints*, vol. abs/1605.02688, May 2016.

[11] A. Paszke, S. Chintala, R. Collobert, et al., "Pytorch: Tensors and dynamic neural networks in python with strong gpu acceleration, may 2017," .

[12] Y. LeCun, L. Bottou, G. B. Orr, and K. R. Müller, "Efficient backprop," in *Neural Networks: Tricks of the Trade. Lecture Notes in Computer Science*, vol. 7700, pp. 9–48. Springer, Berlin, Heidelberg, 2012.

[13] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," *Medical Image Computing and Computer-Assisted Intervention MICCAI 2015*, vol. 9351, pp. 234–241, 2015.

[14] J. Long, E. Shelhamer, and T. Darrell, "Fully Convolutional Networks for Semantic Segmentation," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3431–3440. IEEE Xplore, 2015.

[15] INPE, *PRODES Deforestation estimates in Brazilian Amazon*, National Institute for Space Research, 2019.