Investigations on Combinational Approach for Processing Remote Sensing Images Using Deep Learning Techniques

Ramya T E[#1], Marikkannan M[*2]

[#1]*M.E Computer Science and engineering*
Institute of Road & Transport Technology,
Vasavi College Post, ErodeTamilnadu-638316, India.
[*2]*Assistant Professor (Senior),*
Department of Computer Science and Engineering,
Institute of Road and Transport Technology (I.R.T.T),
Erode Tamilnadu- 638 316,India.

Abstract — *Deep learning (DL) techniques are becoming important to solve a number many of image processing tasks. Among common algorithms, the convolutional neural network and recurrent neural network-based systems achieves the state-of-the- art results on satellite and aerial imagery in many applications. While these approaches are subjected to the scientific interest, there is currently a no operational and generic implementation available at the user level for the remote sensing (RS) community.*

In this paper, we propose a framework whichenablesthe use of DL techniques with RS images and geospatial data. The results takes roots in two extensively used open-source libraries namely, the RS image processing library Orfeo ToolBox and the high-performance numerical computation library TensorFlow. Though ,it can be capable to apply deep nets without restriction on image size and is found computationally efficient, regardless of hardware configuration.

Keywords – *Aerial images, deep learning (DL), neural networks, Orfeo Toolbox (OTB), remote sensing (RS),Tensor Flow (TF).*

I. INTRODUCTION

The volume of earth over observing the data had never stopped to increase in recent years. On one hand, the use of very- high-spatial-resolution satellite (VHRS) images is booming, and sensors are sharper in each generation .There are also more sensors with the close temporal acquisition, such as the Sentinel constellation. Though, the community-based geographic information are gathering systems which is expanding each year in situ geospatial databases like Open Street Map. Deep learning (DL) is enabling a growing trend towards their access in the big data analysis and had a breakthrough impact on the past few years on such diverse domains had involved their impact in streams as computer vision, speech recognition, image processing, and remote sensing (RS). Convolutional neural networks (CNNs) are designed to extract features in images, enabling image recognition, object detection, and semantic segmentation. The Recurrent neural networks (RNNs) are found to be well suited for the sequential data analysis, such as speech recognition and the action recognition tasks. The availability of data and computing resources [1]. Many typical RS problems have been successfully addressed withsynthetic aperture radar interpretation with target recognition, classification from time series (TS) [2],parameter inversion [3], hyperspectral image analysis with classification ,anomaly detection [4], VHRS images interpretation with scene classification [5], [6], object detection [7], image retrieval [8], and classification from time series [9]. DL has addressed other issues in RS, such as data fusion, e.g., multimodal classification, pan sharpening [1], and 3-D reconstruction.

DL allows the researchers in RS stream to move beyond usual approaches and enables to tackle a number of problems with solid results. However, there are still challenges remaining ahead entering into DL requires familiarizing with a framework. Several popular open-source frameworks exist, such as Caffe, Torch, Theano, and TensorFlow (TF), but implementing new methods requires an extensive programing background and DL knowledge, holding back DL democratization in RS community.

Conversely, many studies rely exclusively on preprocessed, densely annotated public data sets like UC Merced or Postdam [11] because data extraction requires knowledge of geospatial standards and tools like geographic information systems and RS image processing software. Second, the need for tools, which scale enough to the huge size of real-world RS images and data sets, rises as a major computing challenge.

A generic framework to enable DL for handling RS images, whose characteristics are computationally efficient and allows the processing of very large RS images with deep nets in machine learning algorithms like support vector machines (SVM) or random forests, offers state-of-the-art results [12]–[13] as,

- open source and cross platform
- enable the users without programming knowledge to use deep nets on RS images
- integrated in an existing rich machine learning framework for RS images

II. PROPOSED SYSTEM

The RS image processing library, Orfeo Toolbox (OTB) [14], and rely on the high-performance numerical computation library for training and inference on deep neural networks, TF [15]. While RS image's processing software has internal mechanisms to deal with the large stacks of images and this problem is out of DL framework's scope.

However, even if RS image processing software implements generally a machine learning framework, there is no existing one enabling the combination of DL algorithms with already implemented well-known machine learning algorithms.

The existing machine learning framework of OTB includes tools with many strategies and parameters fine tuning and extend this framework using TF to enable the processing of RS images as,

- RS data selection,
- sampling,
- training,
- classification, and
- regression,

The goal was to provide a generic DL framework for RS images processing. We propose a solution that takes roots in the OTB library. The original machine learning framework was enriched to embed TF for training and using deep neural networks. We introduce a new component enforcing the OTB workflow, which the developers can implement in pipelines.

A. RS image processing

RS image's processing software has internal mechanisms to deal with the large stacks of images, this problem is out of DL framework's scope. And it often involves serving one unique purpose. Therefore, coding is not generic and each new algorithm implementation requires new effort and expertise. Though, it involves the use of use various frameworks as OTB and TF for its effective accession.

a) *OTB*

The OTB is a library for RS image processing, built on the top of an application development framework widely used in medical image processing, the Insight Toolkit (ITK) . The machine learning framework of OTB is able to process large data sets at continental scale for land mapping and benefits from high-performance computing (HPC) architectures like clusters [16].

b) TF

TF is a library for dataflow programming. It is a symbolic math library and is intensively used for machine learning applications such as deep nets. It can also operate at a large scale in HPC architectures like GPU. We aim to extend the existing rich machine learning framework of OTB with TF. Implementation of the state-of-the-art deep nets should be enabled with the minimum effort, and the opportunity to apply them on RS images must be granted to non-developers, namely users. The existing user-oriented machine learning framework of OTB must be preserved by involving the applications of deep learning mechanisms. The implementation of a component, can be used in a transparent way inside any OTB pipeline, to enable the combination of already implemented approaches with DL.

B. Library paradigms:

The library paradigms enables the involvement of processing logics of various workflows and the following paradigms explains the processing logics of OTB and TF shown in fig 1.

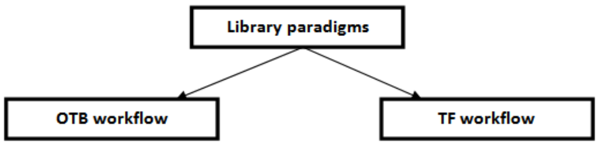

Fig. 1.Processing logic of library paradigms

a) OTB Workflow:

A *pipeline* refers to a directed graph of process objects that can be either sources (initiating the pipeline), filters (processing the data), or mappers (typically, write a result on disk). Sources and filters are found effective in generating one or multiple data objects (e.g. Image and vector). The architecture of OTB inherits from ITK and hides the complexity of internal mechanisms for pipeline execution, which involve several steps. A detailed description of the pipeline mechanism can be found in including figures and sequence diagram.

The execution of a pipeline starts from a mapper, triggering its upstream filter(s).When a filter is triggered, the information about mandatory input data [i.e., information about the output data of the upstream process object(s)] is also requested upstream. In this way, it is propagated back through the pipeline from mappers to sources via filters. Once this request reaches sources, data object's information is generated (e.g., image size and pixel spacing) then propagated downstream to mappers. It should be noted that filters can potentially modify this information, according to the process they implement (e.g., changing image size) which is the case with image re-sampling for instance. Finally, they reach the mapper, initiating the data processing. Information regarding the size of the image that will be produced is then used by the mapper to choose a splitting strategy. Then, the mapper requests its input image to upstream filter(s) sequentially, region after region. This mechanism, named *streaming*, enables the processing of very large images regardless of the memory consumption of the process objects composing the pipeline.

b) TF Workflow:

TF uses symbolic programming, which distinguishes the definition of computations from their execution. In TF, tensors are abstraction objects of the operations and are objects in the memory, simplifying manipulation regardless of the computing environment (e.g., CPU or GPU). A TF model consists in operations arranged into a graph of nodes. Each node can be viewed as an operation taking zero or more tensors as inputs and producing a tensor.

This data flow graph defines the operations (e.g., linear algebra operators), but the computations are performed within the so-called *session*. A high-level API enables the construction of TF graphs, and the session runs the graph in delegating calculations to low-level, highly optimized routines. Among tensors, we can distinguish concepts such as *placeholders*, *constants*, and *variables*. A placeholder is a symbol which involves hosting the input data, e.g., a set of images. As its name indicates, constants are tensors with constant values, and variables hold non-persistent values, e.g., parameters to estimate during training. Variables must be explicitly initialized and can be saved or restored during a session along with the graph. A number of tools enable the design of TF models, like the TF Python API or user-oriented software developed by the TF community.

C. *Pipeline Flow*

The integration of a process object that runs TF session for generic deep nets in an OTB pipeline with RS images as data objects. In the pipeline workflow, the generation of image information and the requested region computation by process objects are crucial steps and the *spacing* is denotedas the physical size of a single RS image pixel. The generation of output image information includes origin, spacing, size, and additional RS metadata, e.g., projection reference. Process objects must also propagate the requested regions to input images. Regarding deep nets implementation and particularly CNNs, this process must be carefully handled. CNNs usually involve several operations, mostly including a succession of convolutions, pooling, and nonlinear functions. There are also a number of deriving operators, e.g., transposed convolution.

Most of these operators modify the size and spacing of the result. For example, convolution can change the output image size, depending on its kernel size and input padding. It can also change the spacing of the output if performed with strides, that is, the step of which is shifted the filter at each computation. Pooling operators are also a common kind of operator, which modify the output size and spacing, depending on the stride and the size of the sliding window and the number of other operators which changes the size and scale the spacing of the output should be noted.

D. Running TF session in process object:

A new OTB process object that internally invokes the TF engine and enforces the pipeline execution and hence can seamlessly process large images. It takes one or multiple input images and produces zero or multiple output images. The input placeholders of the TF model which are found to be fed with the input images of the process object.

After the session run, computed tensors are assigned to the process object outputs. Placeholders corresponding to input images, as well as tensors corresponding to output images are both specified as named in the TF model. The process object uses the RF of inputs, as well as EF and scaling factor of outputs, to propagate requested input image regions and generate information of output images. Finally, two processing modes are currently supported shown in fig 2.

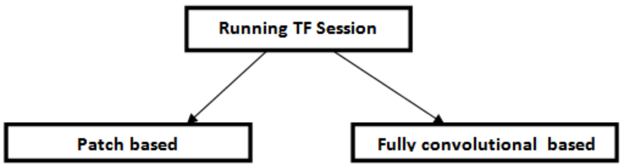

Fig. 2. Processing modes in Running TF session

a) Patch Based processing:

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

b) Fully convolutional processing:

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

E. The OTB Applications :

OTB applications generally implement pipelines composed of several process objects connected together and that provides a new application dedicated to RS images sampling suited for DL. Then, using our new filter provides the new applications for the TF model training and serving finally, involves some new applications consisting in an assembly of multiple OTB applications.

The set of new applications which involves the users without involving the programming knowledge can operate: patches sampling, model, training and serving, and also training and classification applications performing on features of any deep net exported as a TF model. This approach is successfully applied to common RS images with two representative state-of-the-art deep nets. Processing time measurements have shown a great scalability and also our applications can run deep nets without restriction on data set size and regardless hardware configuration. All these new applications are integrated seamlessly into existing machine learning framework of OTB which is shown in fig 3.

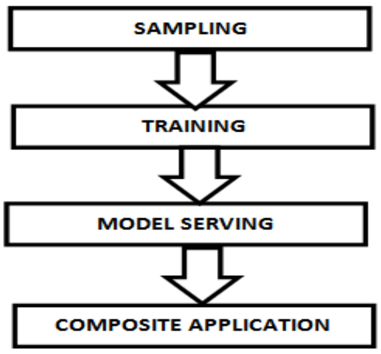

Fig. 3.OTB Application Framework

Then, the new filter provides the new applications for the TF model training and serving. Finally, the introduction of some new applications consisting an access in an assembly of multiple OTB applications. All these new applications are integrated seamlessly into the existing machine learning framework of OTB and defines the flowchart for application framework which is shown in fig 4.

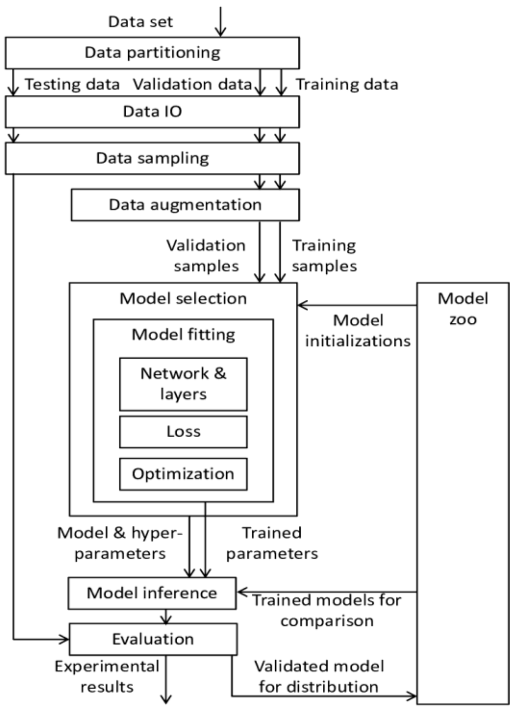

Fig.4.Flowchart forOTB Application Framework

III. EXPERIMENTAL SETUP

A. Sampling:

The existing machine learning framework of OTB includes sample selection and extraction applications suited for geospatial data like vector layers and RS images. Multiple sampling strategies can be chosen with fine control ofparameters which is shown in fig 5. However, DL on images usually involves reference data made of patches, but the existing application performs pixel-wise sample extraction.

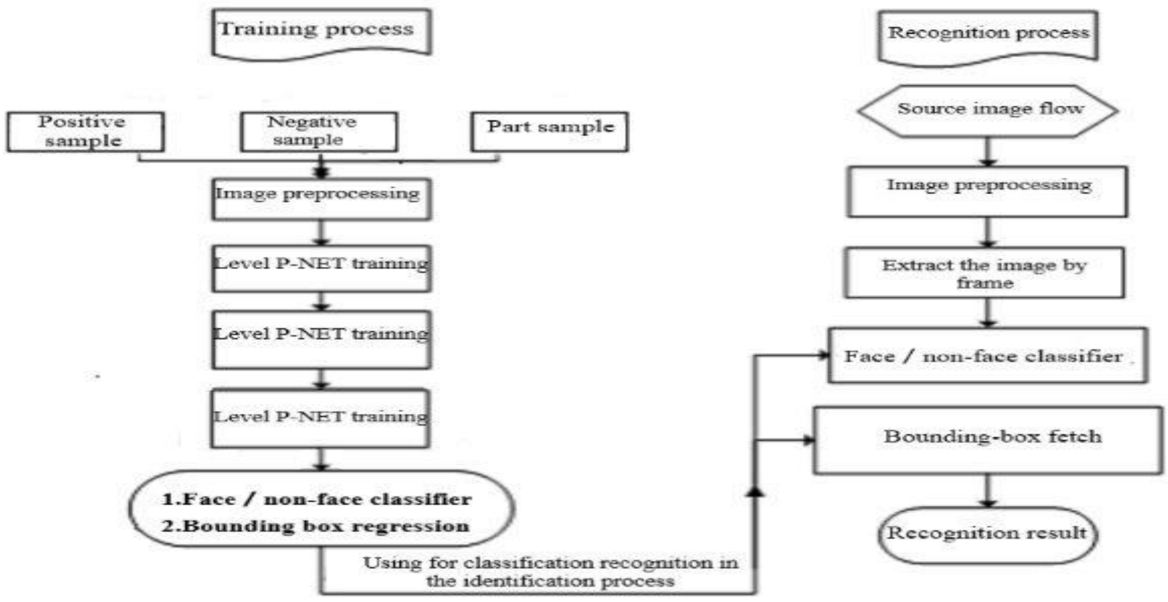

Fig. 5. Flowchart for Sampling

B. Training:

Since the high-level TF API is Python, a light Python environment is provided to allow developers to build and train their models from patches images produced using our sampling application described in Section II-E1. As there are already a number of available TF models and the existing user-oriented open-source applications to create the TF models. Our contribution for RS image deep net training is a user-oriented OTB application dedicated for training the existing TF models. The application can import any existing TF model. It can restore from file model variables before the training or save them to disk after training. Thus, this application can be used either to train a particular model from scratch or perform fine tuning depending on the variables restored. The training can be performed over one or multiple inputs given their corresponding TF placeholder names and provide usual evaluation metrics.

C. Model Serving

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

D. Composite Application

The Composite applications are the OTB applications connected together and the recent studies have suggested that deep nets' features can be used as input features of algorithms like classification, leading state-of-the art results. Regarding the RS image classification, OTB already implements a number of algorithms in its classification application, e.g., SVM, random forests, boost classifier, decision tree classifier, gradient-boosted tree classifier, and normal Bayes classifier. Though the implementation of a composite application that reuses our model serving application as the input of two existing OTB applications: the Train Images Classifier and Images Classifierapplications, respectively, dedicated to train a classifier and perform an image classification.

IV. CONCLUSIONS

The processing of RS images using DL techniques from the user perspective is found to be effective towards its processing rate of access. In generic framework, the OTB and TF libraries are successfully applied that the existing state-of-the-art deep nets on common RS images using our framework and shown a good computational efficiency, without restriction on image sizes and regardless hardware configuration and this approach enables users without coding skills to use deep nets in their RS applications, and developers to create operational RS processing pipelines are enabling the benefit from the development framework of the OTB library.

In the future development, those parameters could be extracted from the serialized TF graph and the memory footprint is computed within the OTB pipeline. The further research could focus on improving the automatic retrieval ofnet parameters and memory footprint and such retrieval of parameters increases the performance of accessing by the use of libraries OTB and TF which enables to improve the performance rate of accession for the future accession and the exposed framework will be proposed as an official contribution in the forthcoming releases of OTB.

REFERENCES

[1] X. X. Zhu et al. (2017). “Deep learning in remote sensing: A review.” [Online]. Available: https://arxiv.org/abs/1710.03959

[2] D. H. T. Minhet al., “Deep recurrent neural networks for winter vegetation quality mapping via multitemporal SAR sentinel-1,” IEEE Geosci. Remote Sens. Lett., vol. 15, no. 3, pp. 464–468, Mar. 2018.

[3] L. Wang, K. A. Scott, L. Xu, and D. A. Clausi, “Sea ice concentration estimation during melt from dual-pol sar scenes using deep convolutional neural networks: A case study,” IEEE Trans. Geosci. Remote Sens., vol. 54, no. 8, pp. 4524–4533, Aug. 2016.

[4] W. Li, G. Wu, and Q. Du, “Transferred deep learning for anomaly detection in hyper-spectral imagery,” IEEE Geosci. Remote Sens. Lett., vol. 14, no. 5, pp. 597–601, May 2017.

[5] M. Volpi and D. Tuia, “Dense semantic labeling of subdecimeter resolution images with convolutional neural networks,” IEEE Trans. Geosci. Remote Sens., vol. 55, no. 2, pp. 881–893, Feb. 2016.

[6] E. Maggiori, Y. Tarabalka, G. Charpiat, and P. Alliez, “Convolutional neural networks for large-scale remote-sensing image classification,” IEEE Trans. Geosci. Remote Sens., vol. 55, no. 2, pp.645–657, Feb. 2017.

[7] G. Cheng and J. Han, “A survey on object detection in optical remote sensing images,” ISPRS J. Photogramm. Remote Sens., vol. 117, pp. 11–28, Jul. 2016.

[8] P. Napoletano, “Visual descriptors for content-based retrieval of remote- sensing images,” Int. mote Sens., vol. 39, no. 5, pp. 1343–1376, 2018.

[9] D. Ienco, R. Gaetano, C. Dupaquier, and P. Maurel, “Land cover classification via multitemporal spatial data by deep recurrent neural networks,” IEEE Geosci. Remote Sens. Lett., vol. 14, no. 10, pp. 1685–1689, Oct. 2017.

[10] G. Masi, D. Cozzolino, L. Verdoliva, and G. Scarpa, “Pansharpening by convolutional neural networks,” Remote Sens., vol. 8, no. 7, p. 594, Jul. 2016.

[11] K. Nogueira, O. A. B. Penatti, and J. A. dos Santos, “Towards better exploiting convolutional neural networks for remote sensing scene classification,” Pattern Recognit., vol. 61, pp. 539–556, Jan. 2017.

[12] D. Marmanis, M. Datcu, T. Esch, and U. Stilla, “Deep learning Earth observation classification using Image Net pretrained networks,” IEEE Geosci. Remote Sens. Lett., vol. 13, no. 1, pp. 105–109, Jan. 2016.

[13] M. Grizonnet, J. Michel, V. Poughon, J. Inglada, M. Savinaud, and R. Cresson, “Orfeo ToolBox: Open source processing of remote sensing images,” Open Geospatial Data, Softw. Standards, vol. 2, no. 1, p. 15, 2017.

[14] M. Abadi et al., “TensorFlow: A system for large-scale machine learning,” in Proc. OSDI, vol.16. 2016, pp. 265–283.

[15] J. Inglada, A. Vincent, M. Arias, B. Tardy, D. Morin, and I. Rodes, “Operational high resolution land cover map production at the country scale using satellite image time series,” Remote Sens., vol. 9, no. 1, p. 95, 2017.

[16] Potsdam Dataset. Accessed:Apr.30,2018. [Online]. Available: www2. isprs.org/commissions/comm3/wg4/2d-sem-label-potsdam.html

[17] OTBTensorflow Remote Module for Orfeo Toolbox. Accessed: May 11, 2018. [Online]. Available: https://github.com/remicres/otbtf