Recognizing terrain features on terrestrial surface using a deep learning model -- An example with crater detection

Wenwen Li
School of Geographical Sciences and Urban Planning, Arizona State University
Tempe, AZ 85287-5302
Wenwen@asu.edu

Bin Zhou
Brisky Inc.
Beijing, China
senosy@gmail.com

Chia-Yu Hsu
Ira A. Fulton Schools of Engineering, Arizona State University
Tempe, AZ
chsu53@asu.edu

Yixing Li
Ira A. Fulton Schools of Engineering, Arizona State University
Tempe, AZ
yixingli@asu.edu

Fengbo Ren
Ira A. Fulton Schools of Engineering, Arizona State University
Tempe, AZ
renfengbo@asu.edu

ABSTRACT

This paper exploits the use of a popular deep learning model -- the faster-RCNN -- to support automatic terrain feature detection and classification using a mixed set of optimal remote sensing and natural images. Crater detection is used as the case study in this research since this geomorphological feature provides important information about surface aging. Craters, such as impact craters, also effect global changes in many aspects, such as geography, topography, mineral and hydrocarbon production, etc. The collected data were labeled and the network was trained through a GPU server. Experimental results show that the faster-RCNN model coupled with a widely used convolutional network ZF-net performs well in detecting craters on the terrestrial surface.

CCS CONCEPTS

• **Computing methods** → Machine learning; *Machine learning approaches*; Neural networks

KEYWORDS

Terrain feature recognition, deep learning, crater, region proposal network

ACM Reference Format:

Wenwen Li, Bin Zhou, Chia-Yu Hsu, Yixing Li and Fengbo Ren. 2017. Recognizing terrain features on terrestrial surface using a deep learning model -- An example with crater detection. In *First ACM SIGSPATIAL Workshop on Articial Intelligence and Deep Learning for Geographic Knowledge Discovery*, November 7–10, 2017, Los Angeles Area, CA, USA. 4 pages. https://doi.org/10.1145/3149808.3149814.

GeoAI'17, November 7–10, 2017, Los Angeles Area, CA, USA

ACM ISBN 978-1-4503-5498-1/17/11…$15.00

1 INTRODUCTION

Terrain feature recognition and classification has become a key research theme in the GIS and Geology community as evidenced by the increasing need to understand the composition of different landscapes and their geological processes [1]. Motivation for such research includes the need for disaster locationing and response [2], precise land management [3], and scaling up site-specific analysis onto local, regional, and national scales [4].

The ability to recognize different geomorphological features from satellite or natural images is also a key measure of success with respect to machine intelligence, which is widely employed to support geospatial intelligence projects[5], (natural) scene comprehension [6], navigation [7], and place-based studies [8].

Current popular approaches for terrain feature (or object) detection on aerial photographs or other optimal images always involves the use of segmentation and classification based on low to mid-level features, such as color, texture, and/or shape [9]. Some machine learning (ML) approaches, such as Support Vector Machines (SVM), or Random Forest, are always incorporated into the classification process. These conventional approaches work well for problems with smaller datasets and few outliers and/or less noise within the data. However, they always suffer from performance challenges when dealing with large datasets and complex detection or classification tasks.

Recently, deep learning techniques have become the state-of-the-art in image processing and many other big data analytics tasks [10]. In comparison to conventional ML techniques, deep learning models (i.e. deep convolutional neural networks, DCNN) hold a deep structure with multiple interconnected layers and have the ability to self-learn low-level features from the raw data and composite them into mid-level to high-level semantics through information propagation and composition in the DNN model.

Differing from image classification [11], terrain feature recognition requires localization of an image object in addition to classification for its category. Szegedy [12] developed a DNN-based regression model to learn both the geometry information and object class. This method was soon beaten by a R-CNN (Region-based CNN) model [13] which provided an almost 30%

increase in the mean average precision (mAP) on the same testing dataset. The model was then modified to reduce computation cost by convolution sharing across proposals [14]. The model then was additionally refined into the Faster-RCNN, which combines object detection network with RPN (Region Proposal Network) to further speed up the training process [15]. A recent benchmark on object detector across Faster-RCNN and other DNN models shows that Faster-RCNN achieves high accuracy as well as outstanding speed/accuracy tradeoff on various datasets [16].

Therefore, in this paper, we employ Faster-RCNN to support the terrain feature recognition task using a combination of remote sensing and aerial images. We start with a binary classification problem aiming at detecting the existence and location of craters, an important geomorphological feature that effects global changes in term of geography, topography, mineral and hydrocarbon production etc., as well as providing knowledge on surface aging and the Earth's history [17]. Although craters seem to be an obvious feature to recognize using the naked eye, tools that apply machine vision for automatic crater detection have not been very successful due to: (1) craters, especially impact craters, often cluster together, therefore, it is difficult to separate one crater from its neighbor; (2) some other terrain features, such as volcanoes or valleys have similar characteristics as craters [18].

To address these challenges, the power of deep learning is harnessed. Section 2 introduces the architecture of the Faster RCNN model coupled with the ZF Net [19]. Section 3 describes the collection of the training data, the training process, the visualization of filters and feature maps for a testing image, as well as some preliminary results and analysis. Section 4 summarizes the work and discusses future research directions.

2 Method

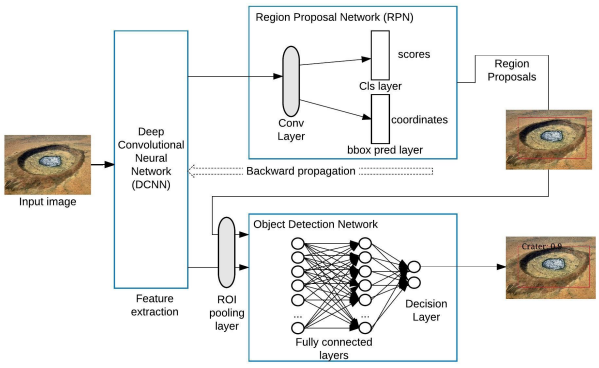

Figure 1: Architecture of the Faster-RCNN based deep learning model

Fig. 1 demonstrates the architecture of the Faster-RCNN based deep learning model, which contains three major components: the deep convolutional neural network (DCNN), the region proposal network (RPN) and the object detection network (ODN). The DCNN is responsible for extracting representative features (or attributes) that characterize different types of objects through a chain of stacked convolutional layers. Popular network models include the ZF-net or VGG [20]. Since ZF-net is used in our implementation, we use it to explain the structure of a DCNN.

The main element of a DCNN is a convolution layer, the input of which is the raw image or a feature map, and the output is another feature map. A convolution operation is defined as applying a group of trainable filters *f* of certain size (i.e. 7*7) to all sub-regions *sr* of the same size in the input data using different moving steps. If we consider the values in a filter as weights, each cell in the resultant feature map will be the sum of the dot product of all elements in *f* and *sr*, and a bias parameter *b*. The output feature map from a convolution layer is often applied with an activation function to decide the signal of an element in the feature map is activated or not. Then the feature map is further processed by a pooling layer for down-sampling purposes. That is, the feature map is divided into non-overlapping smaller regions, and the maximal value is selected to represent each sub-region. The pooling operation generates an abstraction of representations of the feature map. This combined convolution, activation, and pooling layer can be defined as a full convolutional unit (we call it "conv" for short). When multiple such units are stacked together, it becomes a DCNN. In some cases, a convolution unit may only consist of convolution and activation without the pooling operation. We call such unit a "p-conv". Take ZF-net as an example, it is composed by two conv layers and three p-conv layers.

After the convolution, the DCNN is connected with a RPN for generating region proposals. The RPN network starts with a p-conv layer and then two separate convolutional layers (see the RPN box in Fig. 1). The *bbox prediction layer* is to output the predicted bounding box information for a detected object, while the *cls layer* is to output the score indicating whether or not there contains an object, regardless of its category, in the predicted bounding box. The loss function will be a weighted sum from the loss of both the *cls layer* and the *bbox prediction layers*. Although a variety of loss functions can be employed, they all follow the rule that the more significant the difference between the predicted value and the ground truth value is, the higher value the loss function will output. Therefore, minimizing the loss function is the major objective during the RPN training process.

The trained RPN will output the bounding box (BBOX) for detected object, next goal is to feed this information into the ODN to predict which category an object belongs to. The ODN layer can also be called a classification layer. Since the detected BBOX may be in different sizes, a ROI (Region of Interest) Pooling layer is applied to reproject feature map output from the DCNN and cropped by the BBOX into equal-dimension feature maps, then feed this data into two fully connected layers for classification (see ODN box in Fig. 1). The decision layer in this case includes two nodes, representing crater and non-crater categories, both have a value of 0-1, 1 is true for the category, 0 is not, values between 0 and 1 can be considered a possibility of the object belonging to a category. The loss function in the ODN measures the difference between predicted and ground-truth values.

3 Experiments and Results

3.1 Training dataset

To prepare for the training dataset, we manually collected and labeled 300 images (both remote sensing and aerial images) containing craters and 100 images of other categories, such as mountains, geysers, etc. Among the dataset, we randomly selected 80% to use as training data and 20% to be used as testing data. Fig. 2 lists a few example training images. As shown, the raw input images can be of different sizes. Once entering the network for training, they will be resized into an image with the same dimension (224*224), as required by the ZF-net used in our DCNN implementation using CAFFE (https://github.com/BVLC/caffe).

Figure 2: A list of training samples (both craters and non-craters)

3.2 Model training

The training process includes four stages: in stage 1, the RPN model is trained. The parameters, say W_0, in the DCNN network is initiated by a pre-trained model using large databases, such as ImageNet, to reduce training time. The parameters in the RPN model will start with some random values. In stage 2, the ODN will be trained with DCNN reinitiated using W_0 and randomly generated values for parameters in the fully connected layers. The trained parameters in the DCNN are denoted by W_1. In Stage 3, parameters with RPN will be fine-tuned using DCNN with W_1. In Stage 4, ODN is further trained using the newly adjusted BBOX obtained through Stage 3 towards a higher classification accuracy.

Through preliminary experiments, we found that when the iterations are set to be {40k, 20k, 40k, 20k} for stage 1-4, the network achieves a good balance between training and testing mAPs. Note, too many iterations may lead to a higher mAP for training data but it may also cause an overfitting problem. Therefore, this optimal iteration setting was used in the training process.

3.3 Visualization of the feature maps

Fig. 3 demonstrates what the filters and the feature maps look like in the first and second layer of the DCNN module after the network is well trained. For ZF-Net, there are 96 filters in layer 1 and 256 filters in layer 2. Note that each filter should result in a different feature map. In our visualization, only the first 36 feature maps are visualized for a clear view. Boxes in red show a correspondence of a filter with a resultant feature map.

It can be observed that the trained filter in layer 1 includes different texture and color patterns, some of which help to identify the edge information of a crater. In layer 2, since each filter has 96 channels (bands), and only three bands can be used to generate the feature map image, it is difficult to distinguish their patterns. However, from the feature maps one can tell that the area and shape (circular) characteristics were captured in this layer. This visualization result showcases how a DNN model derives knowledge through hierarchical composition of semantic information from shallower to deeper layers.

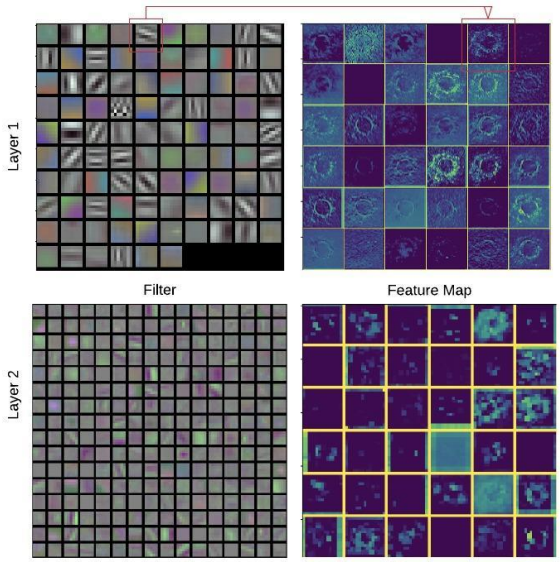

Figure 3: Visualization of filters and feature maps at the first and second layer of DCNN.

3.4 Results and Analysis

The experiments were run on a GPU server using a Geforce GTX 980 chip with 6GB memory, the total training time was 233 minutes for the proposed iteration setting and a learning rate (0.001). The mAP can reach over 97% for training data and 90% for testing data. Fig. 4 illustrates detection results for a few sample images. The results show that the deep learning model is capable of detecting multiple craters coexisting in the same scene (Fig. 4(a)), very small instances (in Fig. 4(b)), and those adjacent to each other (Fig. 4(c)). This is a benefit from the outstanding self-learning ability of the DCNN with an optimized training

process. In addition, the results also show that it is possible to transfer the knowledge learned from an aerial image to make predictions for craters in the remote sensing image, the labeled training data of which is less available than the labeled aerial images. However, we also observed some false positive results. As shown in Fig. 4(d), there exist two connected craters and their shared border was erosion. Although they were detected, but only at a low accuracy and detected as one instance rather than two. The network should be better tuned to handle such complexity in the future.

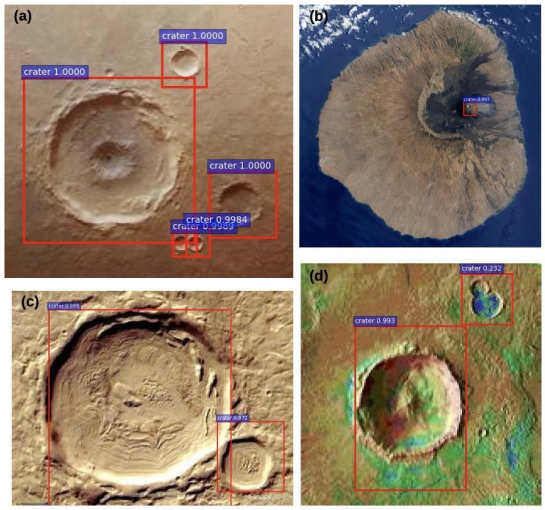

Figure 4: Detection results from trained Faster-RCNN model

4 CONCLUSIONS

In summary, we implemented a Faster-RCNN based deep learning model for recognizing terrain features automatically. Although DNN models have been exploited in object detection tasks in the literature, few works have focused on characterizing natural terrain objects. This is partially because such objects do not have a clear edge as other manmade object does. Training data for this task, especially from remote sensing images are also rare. In this work, we take a combination of natural scene and remote sensing images as the training data in the hope that the knowledge learned about terrain objects in a natural scene image can also be transferred to detect and categorize those in a remote sensing image. In this work, we verified the feasibility of using natural images to compliment the scarcity of labeled remote sensing training images for crater detection.

In the future, several directions are worth immediate investigation. First, we are extending to model to support the multi-class terrain feature classification and recognition problem. Although the network creates very high accuracy on the binary classification problem, we expect its performance (accuracy, training time) to be affected when being extended to classify multiple terrain objects simultaneously. Second, to further increase the prediction accuracy, we are working on extending the Faster-RCNN model by a set of ensemble approaches. We expect this work to contribute significantly to the attempt of using new and AI (Artificial Intelligence) methods for exploiting terrain objects not only on the Earth surface but also on other planet, such as Mars, to support a variety of discovery with regard to geological, geographical, and geophysical processes.

ACKNOWLEDGMENTS

This work was partially supported by U.S. Geological Survey under Grant G15AC00085.

REFERENCES

[1] Conrad, O., Bechtel, B., Bock, M., Dietrich, H., Fischer, E., Gerlitz, L., ... & Böhner, J. 2015. System for automated geoscientific analyses (SAGA) v. 2.1. 4. *Geoscientific Model Development*, *8*(7), 1991-2007.

[2] Blaschke, T., Feizizadeh, B., & Hölbling, D. 2014. Object-based image analysis and digital terrain analysis for locating landslides in the Urmia Lake Basin, Iran. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, *7*(12), 4806-4817.

[3] Cirencester, U. K. 2008. Optimal mapping of site-specific multivariate soil properties. *Precision Agriculture: Spatial and Temporal Variability of Environmental Quality*, *790*, 208.

[4] MacMillan, R. A., Pettapiece, W. W., Nolan, S. C., & Goddard, T. W. 2000. A generic procedure for automatically segmenting landforms into landform elements using DEMs, heuristic rules and fuzzy logic. *Fuzzy sets and Systems*, *113*(1), 81-109.

[5] Miller, M. H. 2009. *Geospatial Intelligence School Enhances Iraqi Army Effectiveness*. ARMY ENGINEER SCHOOL FORT LEONARD WOOD MO.

[6] Huajun, L., Jingyu, Y., & Chunxia, Z. 2004, December. A generic approach to rugged terrain analysis based on fuzzy inference. In *Control, Automation, Robotics and Vision Conference, 2004. ICARCV 2004 8th* (Vol. 2, pp. 1108-1113). IEEE.

[7] Williams, S., Dissanayake, G., & Durrant-Whyte, H. 2001. Towards terrain-aided navigation for underwater robotics. *Advanced Robotics*, *15*(5), 533-549.

[8] Hu, Y., Gao, S., Janowicz, K., Yu, B., Li, W., & Prasad, S. 2015. Extracting and understanding urban areas of interest using geotagged photos. *Computers, Environment and Urban Systems*, *54*, 240-254.

[9] Zhou, X., & Li, W. 2017. A Geographic Object-Based Approach for Land Classification Using LiDAR Elevation and Intensity. *IEEE Geoscience and Remote Sensing Letters*, *14*(5), 669-673.

[10] LeCun, Y., Bengio, Y., & Hinton, G. 2015. Deep learning. *Nature*, *521*(7553), 436-444.

[11] Hu, F., Xia, G. S., Hu, J., & Zhang, L. (2015). Transferring deep convolutional neural networks for the scene classification of high-resolution remote sensing imagery. *Remote Sensing*, *7*(11), 14680-14707.

[12] Szegedy, C., Toshev, A., & Erhan, D. 2013. Deep neural networks for object detection. In *Advances in neural information processing systems* (pp. 2553-2561).

[13] Girshick, R., Donahue, J., Darrell, T., & Malik, J. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 580-587).

[14] Girshick, R. 2015. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision* (pp. 1440-1448).

[15] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2017. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Trans. Pattern Anal. Mach. Intell.* 39, 6 (June 2017), 1137-1149. DOI: https://doi.org/10.1109/TPAMI.2016.2577031

[16] Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., ... & Murphy, K. 2016. Speed/accuracy trade-offs for modern convolutional object detectors. *arXiv preprint arXiv:1611.10012*.

[17] Robbins, S. J., & Hynek, B. M. 2012. A new global database of Mars impact craters≥ 1 km: 1. Database creation, properties, and parameters. *Journal of Geophysical Research: Planets*, *117*(E5).

[18] Kim, J. R., Muller, J. P., van Gasselt, S., Morley, J. G., & Neukum, G. 2005. Automated crater detection, a new tool for Mars cartography and chronology. *Photogrammetric Engineering & Remote Sensing*, *71*(10), 1205-1217.

[19] Zeiler, M. D., & Fergus, R. 2014, September. Visualizing and understanding convolutional networks. In *European conference on computer vision* (pp. 818-833). Springer, Cham.

[20] Simonyan, K., & Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.