Symbiosis

www.symbiosisonline.org
www.symbiosisonlinepublishing.com
ISSN Online: 2474-9257

Research Article | Journal of Computer Science Applications and Information Technology | Open Access

Classification of Vegetation Areas Using LIDAR Images and AI

Gevand Balayan and Yfantis EA*

Professor, Computer Science Department, University of Nevada, Las Vegas, NV 89154-4019, USA

Received: September 09, 2018; Accepted: September 14, 2018; Published: September 21, 2018

***Corresponding author:** Evangelos Yfantis, Professor, Computer Science Department, University of Nevada, Las Vegas, NV 89154-4019, USA, E-mail: evangelos.yfantis@unlv.edu*

Abstract

Many states have limited amount of drinking water which becomes scarcer every year due to changing climate and growth. To manage their water resources wisely they encourage their residence to replace the grass on their loans with xeriscaping. Thus, they pay the residence to take of the grass which demands a great deal of water and replace it with gravel or desert plants demanding very little watering, or no more watering than the rainfall. The amount of money paying the residence to take of the grass and other water consuming vegetation is considerable often in the order of tens of thousands of dollars. Often, property owners take of the grass, take the money, and later on they change their mind or sell the property to another person and they reinstall the grass in the property. In such a case the authorities must take the money back. In order to automate the process of who maintains a loan with grass and who does not, an unmanned aerial vehicle with LIDAR is being used to automatically recognize grass loans and vegetation areas, as well as xeriscaping for each address in a city. In this research paper we show the digital image processing and AI algorithm used on LIDAR images in order to classify a building as having a grass loan and vegetation, or having a certain percentage of vegetation, and the rest xeriscaping, or all xeriscaping.

Keywords: Machine vision; Computer vision; Machine intelligence; Image segmentation;

Introduction

One of the most important programs that the water conservation departments run is the Water Smart Landscape (WSL). It pays out people who want to get rid of their lawn and replace it with desert landscape. However there needs to be a way to constantly monitor if the customer chooses to put the grass back. In that case they need to make a payment back to Water Authority (WA), and refund the money the WA gave them. Currently it is done through manual inspections of aerial imagery or simply driving to the location and confirming that the grass hasn't been planted back. As more homes participate in this program, the backlog of homes that needs to be checked yearly grows. In addition to that the labor cost is relatively large. This cost includes inspectors walking from building to building and inspecting if they are compliant according to their contract, people inserting the data in the computer, people providing permission to use the data bases, quality control people, etc. Associated with using that large number of people is the ability for each one of these people to introduce errors at each level of this multilevel process. In order to reduce the cost and expedite the process a manned flying platform with LIDAR imagery was introduced. Although this is an improvement over the walk inspection it still has some drawbacks associated with human error. It is very difficult for the pilot to fly in a way that would not miss any building, or to avoid occlusions thus missing areas of interest. In addition to that the large video recorded has to be inspected manually by humans and that process was proven to be error prone. By developing an AI that can constantly scan aerial imagery using UAVs and look for signs of vegetation in places it shouldn't exist, it helps the WA to save time and money on mundane task of monitoring hundreds of thousands of square feet of aerial imagery. The UAV can preprogrammed to follow a route that would thoroughly examine every building, capture the video, analyze it real time insert the results in the record of the database for that building using a large capacity solid state disk on board to store, and update the data. The embedded software uses real time LIDAR video capture, followed by real time image processing and real time artificial intelligence for classification. The main objective of this research is to develop an accurate and efficient way of identifying whether an aerial image contains vegetation. We use neural networks and specifically the back propagation neural network algorithm, whose task is to work with a small subsection of an input image and provide a probability of that section containing vegetation. Once each section's probability is measured the image could be reconstructed and the square footage of how much vegetation exists on the aerial image could be calculated. The main advantages of this solution are the speed at which each section could be calculated. The back propagation neural network is simple and requires few calculations to classify each section. The process itself could be completely parallelized since each section is independent from each other. In addition to that the classification algorithm can be programmed in an FPGA which can be incorporated on a PCB board used to input the camera video and recognize real time, areas having vegetation and areas with xeriscaping. Our contribution consists of several important decisions. The first is the size of the input which included the width in pixels, and the height in pixels of

the input image. The second is the number of input neurodes. The third is the number of hidden layers, and the size of each hidden layer. The fourth is the number of neurodes of the output layer. These four decisions define the architecture of the neural network used. In addition to the architecture decisions related to the loss function that best applies to the problem needed solution, has to be made. In our case we chose the quadratic loss function. The remaining super parameters have to be defined also. That includes the optimal weight initialization, and the learning rate. Weight initialization is very important because proper initialization improves convergence and increases the probability of correct classification. The number of epochs to be used during the training is also very important. New to this research is also the sampling and experimental design. Sampling and experimental design help us decide how many input data are needed in order to properly train the neural network. This research paper is organized as follows: First is the abstract, followed by the introduction, followed by the background section, followed by advantages and drawbacks of using back propagation compared to convolutional neural networks and other statistical pattern recognition algorithms. The background section is followed by the sampling theory and experimental design needed in order to assure that enough data is used for the training set in order to train the neural network to maximize the probability of correct classification, then the conclusion and further research, and finally the reference.

Background

Four band aerial imagery is commonly used in GIS applications. Generally band 1, 2, and 3 are Red, Green, and Blue"true color" values. The 4th band is the alpha channel and for LIDAR is generally known as NIR (near infrared). It is commonly saved as a .tiff or .tif format. Neither tiff nor tif formats have a real differences in format between them. The pixel value is stored as a 32 bit integer. Where the first 8 bits are the Alpha channel, the next 8 are Blue, followed by Green and finishing with Red. The below code sample shows how to extract a band value of pixel given an input integer by using the logical and operator with 1111 1111 8 bit mask and shifting properly. The following code is used to extract the values of Red, Green and Blue channels, as well as the A which is the near infrared channel. Figure 1 shows how the LIDAR pixel data is stored.

```
public static int GetR(int pixel)
        return (pixel & 0xff);
public static int GetG(int pixel)
        return ((pixel >> 8) & 0xff);
public static int GetB(int pixel)
        return ((pixel >> 16) & 0xff);
public static int GetA(int pixel)
        return ((pixel >> 24) & 0xff);
```

Figure 1: Structure of a LIDAR pixel

Back Propagation (BP) refers to a broad family of Artificial Neural Networks (ANN), whose architecture consists of different interconnected layers [1]. The BP ANNs represents a kind of ANN, whose learning algorithm is based on the Steepest - Descent technique. If provided with an appropriate number of Hidden units, they will also be able to minimize the error of nonlinear functions of high complexity. A neural network consists of neurons [1]. They are non-linear processing elements that will sum the incoming signals in order to generate an output signal via a pre-defined non-linear function. The neurons are connected by terms with variable weights. The output of one neuron multiplied by a weight becomes the input of an adjacent neuron of the next layer [2].

The neurons are arranged into sections called layers. A neural network will always have an input layer and an output layer. In between the two layers there could be any number of middle layers known as hidden layers. The goal of training a neural network is to find weights that will result the minimization of the error signal in the output layer. The most common form of machine learning is supervised learning. Supervised learning

consists of presenting an input vector into the network's input layer. The network's output is calculated, and compared to the true value of the input layer. The error between the two is then used in the adjustment of weights until the error reaches acceptable levels. Artificial intelligence is a hot topic in the field of computer science. Technological leaders such as IBM, Intel, NVIDIA, Google, Amazon, Microsoft and Facebook are all investing heavily in building AIs that will give them advantage in the market. Since the focus of this paper is on research done in classifying aerial imagery, I chose to focus my research on topics similar to that nature. Neural networks is the part of artificial intelligence that works so well in classification and pattern recognition that is used very widely in machine learning, automation, computer vision, robot vision, machine intelligence and robot intelligence.

Vehicle detection with orientation estimation in aerial images has received widespread interest as it is important for intelligent traffic management. This is a challenging task, not only because of the complex background and relatively small size of the target, but also the various orientations of vehicles in aerial images captured from the top view [3]. The researchers chose to use a feed-forward Convolutional Neural Network (CNN) named Oriented SSD (Single Shot MultiBox Detector, SSD). The group chose to base their research on the VGG-16 convolution network .

Figure 2 shows the structure of the neural network's convolution [3]. A 512 by 512 image goes through 5 VGG16 layers, after which additional detection layers are added as layer 6 through 11. Figure 3 explains the architecture of the VGG16 lair. The main difference between the works for aerial imagery detection is the method used in classification of the images. With vegetation having the benefit of multi-band imagery allows to use a simpler form of a back propagation neural network, however since the researchers were only working with basic three band images, a convolutional layer is needed for further accuracy in classification. A future research topic could be on combining the use of convolutional neural networks proposed with the multi-band imagery in order to see if the accuracy of the results improves.

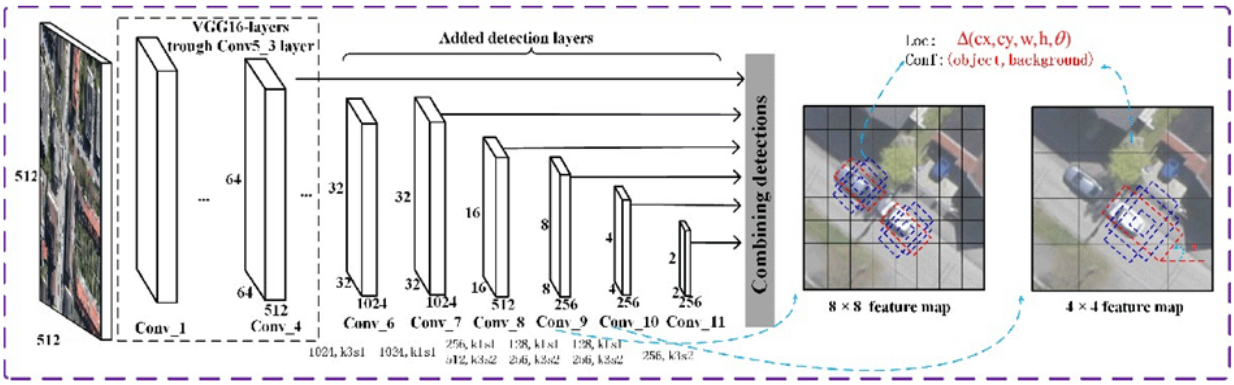

Figure 2: Proposed Convolutional Structure

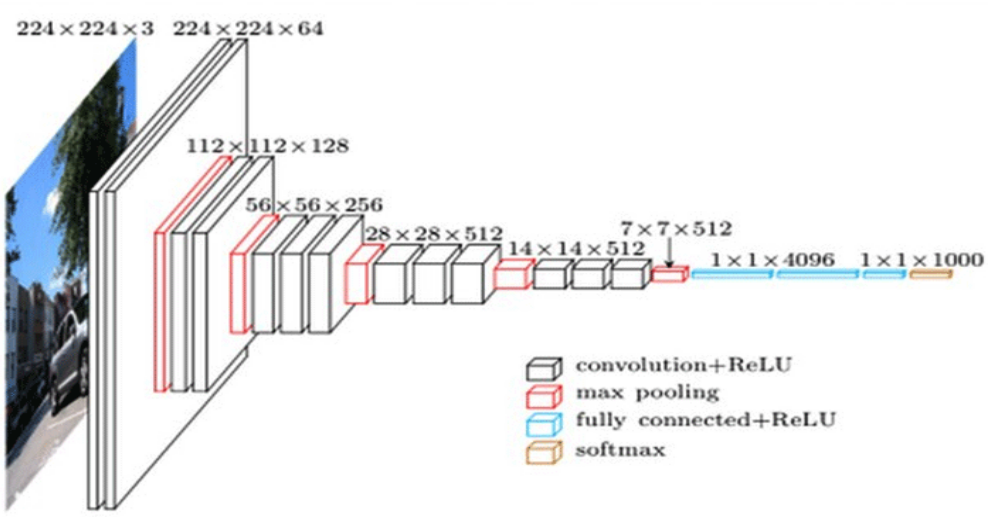

Figure 3: VGG 16

Applications of neural networks using shallow learning or deep learning architectures have increased exponentially in the recent years and include many areas especially robotics, communications, structural engineering and many others [4-12].

Sampling Theory Experimental Design and Learning Rate Control

The images obtained by the LIDAR consisted of pixels having the R, G, B, and A channels where the A channel was the near infrared. After performing digital image processing on the near infrared channel it was clear enough that the near infrared channel was sufficient to classify if a chosen segment of an image contains vegetation or not. In the right side of figure 4 we show a typical LIDAR image section when all bands are used.

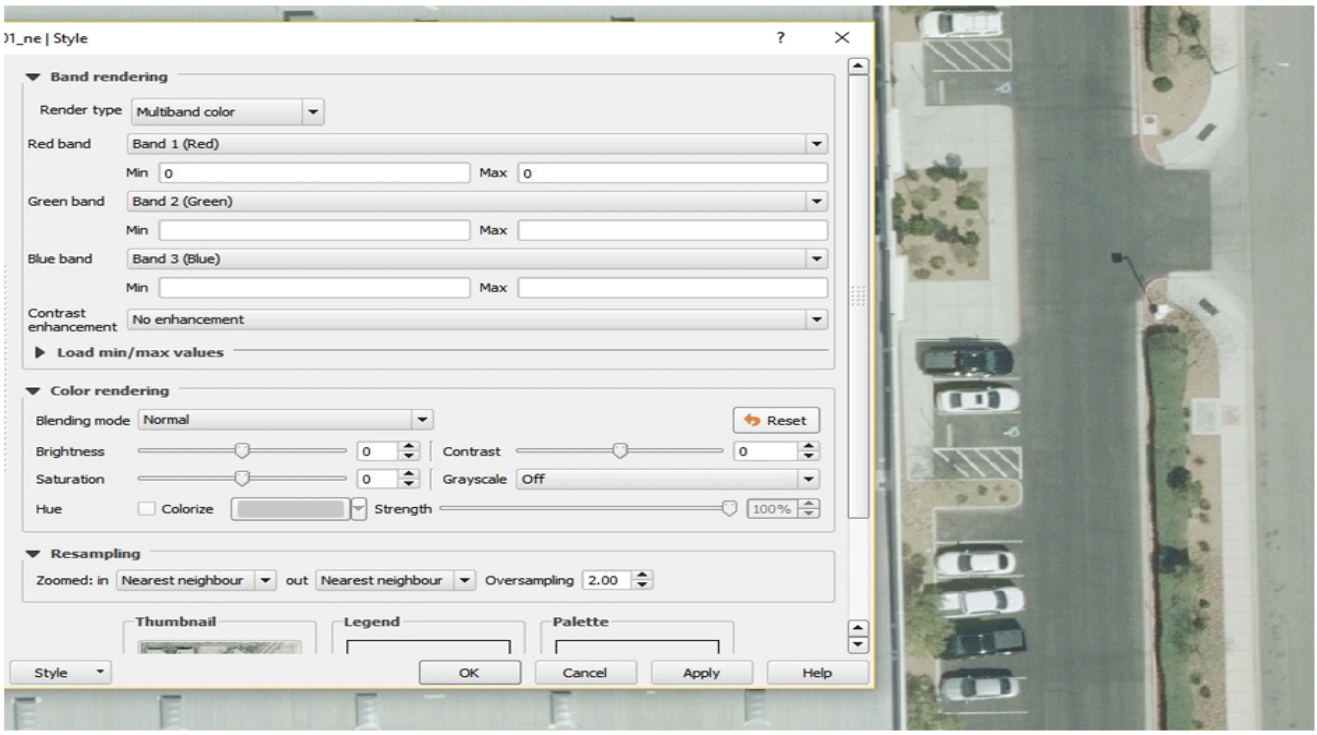

Figure 4: The right side of the figure shows a typical LIDAR image obtained by the flying platform

After creating several classifiers some using the (R, G, B) others using the (R, G, B, A) and yet others using the A (near infrared) channel only we realized that we get the same classification results using the near infrared only as when using all channels. Considering that our algorithm had to do the classification real time we opted to use the near infrared channel only. Thus we divided our image into segments of 10x10 pixels and our objective was to classify the 10x10 segment if it has 100% vegetation, or 75% vegetation, or 50% vegetation, or 25% vegetation, or less than 25% vegetation (but not 0) or no vegetation at all. For each pixel we created a neurode. Thus the input layer consisted of 100 neurodes. The process of choosing the proper number of neurodes in the hidden layer, the proper number of segments in the training set, the proper learning parameter, the proper number of epochs, are both an art and a science. In this case after trial and error, one hidden layer was chosen with 30 hidden neurodes based on trial and error. Notice that if too many neurodes are chosen in the hidden layer some weights will be zero and proper adjustments have to be made to avoid that. The output layer has six neurodes. The first neurode from the top right recognizes if the segment has 80-100% vegetation and the ideal correct output for that is the six component vector [t1 t2 t3 t4 t5 t6] = [1 0 0 0 0 0]. The second neurode from the top recognizes if the segment has 60%-80% vegetation, the ideal correct output for that is the six component vector [t1 t2 t3 t4 t5 t6] = [0 1 0 0 0 0]. The third output neurode from the top recognizes if the segment has 40% - 60% vegetation, the ideal correct output for that is the six component vector [t1 t2 t3 t4 t5 t6] = [0 0 1 0 0 0]. The fourth output neurode from the top recognizes if the segment has 20% - 40% vegetation, the ideal correct output for that is the six component vector [t1 t2 t3 t4 t5 t6] = [0 0 0 1 0 0]. The fifth output neurode from the top recognizes if the segment has 1% - 20% vegetation, the ideal correct output for that is the six component vector [t1 t2 t3 t4 t5 t6] = [0 0 0 0 1 0]. The last neurodes recognizes if the segment has no vegetation at all, and the ideal correct output for that is the six component vector [t1 t2 t3 t4 t5 t6] = [0 0 0 0 0 1]. During the operation of the neural network for a given input we obtain an output vector of six numbers. Each one of these six numbers is between 0 and 1. If the ith element of the vector (starting from the

far left) is the maximum then we classify the input that it belongs to the ith class. Thus in our case if the output is [0.9 0.3 0.1 0.0 0.2 0.0] we classify the segment as having 80%-100% vegetation, on the other hand if it is [0.2 0.4 0.8 0.5 0.1 0.0] we classify it as 40%-60% vegetation. Many activation functions were considered, including the logistic, hyperbolic tangent, and ReLU. The tanH and ReLU worked equally well for our case. The loss function used was the sum of square error. The initial value of the learning rate was 0.1, and the process was on line learning meaning that every sample from the learning set as it went thru the neural network starting from the input layer going thru the hidden layer and then thru the output layer, propagated the error backwards adjusting the weights of the output layer and the hidden layer so that the total error was reduced in every trial. If for one sample going thru the total error increased then we went back repeating the process using a reduced learning rate. The big question was how many samples we should use in order to teach the neural network to correctly classify a 10x10 segment of a LIDAR image. To answer this question one has to consider the labor cost of preparing every 10x10 segment. The preparation entails writing a computer program to count the number of vegetation pixels of a 10x10 segment at a point, pointed by the mouse and then cropping it and creating a file for it if needed. Considering that the neural network has to have the ability to recognize with high probability accuracy has higher priority than the cost of sample preparation, but still preparing many more samples than needed adds to the cost and time considerably without adding to the accuracy of the neural network considerably. In order to decide how many samples are needed, first we prepared 200 samples from each class for a total of 1,200 samples; we used this to train our network. After computing the weights using on line learning with the 1,200 samples we used 40 epochs to further adjust the weights. The reason we used 40 epochs is that after that the error was not improving and was isolating above and below the value obtained in the 40th epoch. In the next trial we repeated the process using 400 samples per class and that decreased the error considerable. So in each trial we increase the number of samples per class by 200 more than the previous trial. When the numbers of samples were 1,200 per class for a total of 7,200 samples the process stabilized, meaning that the total error reduction in the following trial was not significant. So although according to our process the correct number of samples was 1,200 samples per class we opted to overdesign the sampling process and we used 2,400 samples per class for a total of 14,400 samples. We also used 7,200 samples as a test set. Weight initialization is an important factor in overall convergence and performance of the neural network. We experimented with many weight initialization probability distribution functions. We started with the uniform [-0.5 0.5], the normal distribution with standard deviation equal to 0.8 and mean zero, and with the double exponential with mean zero and standard deviation equal to 1. We obtained the fastest convergence initializing the weights using the normal distribution with mean zero and variance 1, and also with the double exponential with mean and standard deviation 1. We also tried mini batches of size 10 using the same 14,400 samples to teach the neural network to recognize the correct class that a 10x10 segment belongs to the error obtained and the probability of correct classification was the same as in the online learning. The overall probability of correct classification was 0.983. For class six was 100% and for the other 5 classes was about 0.98. Capturing the images using a flying platform is an expensive process because renting aircraft and pilot fees are expensive. Preparing the samples for the learning set, and the testing set, was also time consuming and expensive. Designing and testing the neural network was also time consuming process. Once the neural network is operational and the correct weights for the neurodes of the hidden layer and the output layer are computed then running the network to classify new inputs is very fast. The process is also highly parallelizable. It is also not very difficult to program the algorithm in an FPGA and then use the FPGA to take the LIDAR image as input and process it real time.

Conclusion and Future Work

In this research paper we developed a back propagation NN to classify segments of LIDAR images and decide the proportion of vegetation they include. The NN network consists of 100 input neurodes, one hidden layer with 30 neurodes, and one output layer with 6 neurodes. The training set included 14,400 samples, and the test set included 7,200 samples. After initializing the neurode weights, selecting the activation function, the learning rate and developing a strategy of adjusting the learning rate during the learning process, and using 40 epochs to train the network, the overall probability of correct classification 98.3 %. Although the learning process was relatively slow, the production algorithm is very fast, parallelizable, and can be program in an FPGA for real time classification. The future work includes automation to this process. Automation will consist of a UAV design to glide for many hours with as little power as possible with secure communications with a LIDAR, and embedded system that will include the algorithm described in this paper, and additional software to automatically log the vegetation properties associated with each building.

References

1. Fu-Chuang Chen. Back-Propagation Neural Networks for Nonlinear Self-Tuning Adaptive Control. IEEE Control Systems Magazine. 1990;10:44-48.
2. Tang T, Zhou S, Deng Z, Lei L, Zou H. Arbitrary-Oriented Vehicle Detection in Aerial Imagery with Single Convolutional Neural Networks. Remote Sens. 2017;9(11);1170.
3. Simonyan K, Zisserman A. Very deep convolutional networks for Large-Scale image recognition. Computer Vision and Pattern Recognition (in Press). 2015.
4. Yfantis EA. An Intelligent Robots-Server System for Solar Panel Cleaning and Electric Power Output Optimization. Int Rob Auto J. 2017;3(5). doi: 10.15406/iratj.2017.03.00066
5. Yfantis EA, Harris SL. An Autonomous UAS with AI for Forest Fire Prevention, Detection, and Real Time Advice and Page 2 of 5 Communication To and Among Firefighters. J Comp Sci Appl Inform Technol. 2017;2(3):1-5.

6. Zamora E, Nakakuni M, Yfantis EA. Quantitative Measures to evaluate Neural Network Weight Initialization Strategies. The 7th IEEE Annual Computing and Communication Workshop and Conference. 2017. doi: 10.1109/CCWC.2017.7868389
7. Yfantis EA, Nakakuni M, Zamora E. Low Bandwidth Transmission Algorithm for Reliable Wireless Communication. The 7th IEEE Annual Computing and Communication Workshop and Conference. 2017. doi: 10.1109/CCWC.2017.7868422
8. Zamora E, Ramos M, Moutafis K, Yfantis EA. Robot-Server Architecture for Optimizing the Solar Panel Power Output. Transaction on Machine Learning and Artificial Intelligence. 2016;4(4):9-17.
9. Zamora E, Moutafis K, Ramos M, Yfantis EA. A robot architecture for detecting dust and cleaning solar panels. Proceedings of the 31st International Conference on Computers and Their Applications, CATA. 2016:393-399.
10. Zamora E, Ho S, Yfantis EA. Using Spectral Decomposition to Detect Dirty Solar Panels and Minimize Impact on Energy Production. Advances in Image and Video Processing, 2015;3(6):1-12.
11. Shrestha K, Shrestha PP, Bajracharya D, Yfantis EA. Hard-Hat Detection for Construction Safety Visualization. Journal of Construction Engineering. 2015;1-8.
12. Yfantis EA, Fayed A. A novel Digital Modulation Method foe Digital Wireless Communications. JCMCC. 2015:43-51.