Object Detection by means of K Nearest Neighbors Classification

Gayatri Behera

Abstract— **The primary objective of machine learning is to be able to analyze data and derive patterns from it. But the main hindrance as always remains that the quantity of data is not only vast but it can also be of varied formats. In this paper, a system to perform object detection from high resolution imagery has been discussed. K Nearest Neighbors (KNN) algorithm has been chosen to perform detection and classification of the objects from its surroundings. Finally, this paper suggests bottlenecks and possible improvements to the system so as to obtain better results and improved accuracy.**

***Keywords*— k nearest neighbors; object detection.**

I. Introduction

MACHINE learning enables identification of patterns from a given sequence of data. The data being presented can belong to any multimedia format such as text, audio, visual data. Object detection involves identifying objects in an image based on certain features that it possesses that would distinguish it from the rest of the imagery. Efficient and accurate object detection has far-reaching implications in that it can find applications in fields such as geology, security, terrain-assessment etc. Here, the focus is on obtaining insight from satellite imagery visuals, that presents a bird's eye view of terrain having highway intersections with cars plying along it. The aim being, to assess the number of red cars in the image, first the image was pre-processed to focus on a specific area. To ultimately perform the classification of the imagery, KNN algorithm was employed and its performance was assessed.

The rest of the paper is organized as follows. The implementation details are provided in section II. Section III provides a brief discussion on the various approaches employed to solve the problem. The conclusion is presented in section IV.

II. Implementation

As part of the implementation of this project, two large images having dimensions (6250, 6250, 3) were presented. The data chosen being images, it was found to be easier to derive insight from it, if it were converted into another format. Python's Numpy library was chosen to read the input files and to convert it into Numpy-based arrays. In this format it is found that such large image-based files are easier for further processing and to extract insight from. For the analysis, a training image, a testing image and some ground truth values were presented. Analysis of the training or test image showed a high-resolution satellite imagery having plots of land with highway intersections having cars plying on them. The goal was to fetch the location and number of red-colored cars from such an image. Further analysis of the ground truth data showed that it held information about the location of the cars in the training image. The location was in the form of x, y coordinate indicating position of the car.

To perform object detection of the red cars, it was required that ideally a training image be identified that would be populated with enough cars so as to train the model. The testing data here being unlabeled needed to be accurately predicted. Labels were obtained from the training data and ground truth to accurately predict the classes of the unlabeled testing data. Among the various classification methods available such as Naïve Bayes Classifier, KNN classifier etc., KNN was chosen as it is thought to be simpler to implement and having a better performance over naïve bayes classifier. KNN though is regarded to be an instance-based learning algorithm having a lazy-learning approach, in that the classifier model is ultimately built during the classification process and not during the actual training process [2]. This escalates the computation time during actual classification process.

For implementation, the default KNN library provided by Python's scikit-learn module was used. It performs classification of objects by grouping closely related objects together, after taking various features under consideration. The default implementation makes use of Euclidean distance between feature points.

The major blocks in implementation of this system include -

i. Preprocessing
ii. Training the model
iii. Validating
iv. Testing

A major hindrance observed for these kinds of systems was that computation time and speed became a major bottleneck for implementing a decent learning system. In this case, as the image size was this huge, preprocessing was required to ensure that it could fit into the model. The preprocessing employed here was cropping the main image into smaller ones to ease computation while training the model. Only after this was accomplished the rest of the implementation could be smoothly performed.

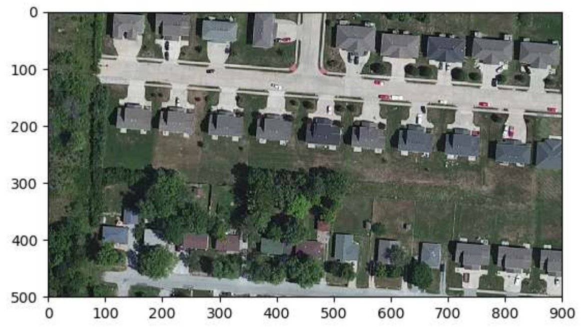

Fig. 1. Cropped image showing presence and location of red cars from original satellite image.

III. EXPERIMENTS

For deciding on the training, validation and testing datasets the original image was cropped such that while performing the actual computation we only focused on one small aspect of the image. Preprocessing the training set to smaller sub-images helped greatly reduce the classification time. Various such regions were identified in the image that had a sizeable number of red cars and that could be utilized for training, validating and testing the model. As an example, the region chosen for training the model belonged between the ranges (4700, 5000) for the x co-ordinate and ranges (900,1050) for the y co-ordinate.

On observing the ground_truth.npy by making use of the numpy library, it was determined that it was a 2D numpy array having dimensions [28,3] i.e. it was composed of 28 instances having three features each. The first two features described the x, y co-ordinates of the red cars. The last feature merely acknowledged that a red car was present at that particular co-ordinate. From this, it could be deduced that by applying these co-ordinate values to the training image by taking sub setting approach offered by Python, enabled one to extract features for training the model.

On observing the train.npy file by making use of the numpy library, it was deduced to be a 3D numpy array having dimensions [6250, 6250, 3] i.e. it was composed of three features or attributes that could be considered to being RGB values of that particular pixel. These would act as the feature set for the purpose of training and gaining insight from the image. The objective would be to classify the points on the image as either a red car or not a red car.

For the purpose of creating and training a classification model to segregate the red cars from the surrounding terrain and rest of the images K Nearest Neighbors (KNN) approach to classification was chosen. This being primarily an instance-based learning methodology keeps stock of the instances of the training data. It is classified by a majority vote, with ties broken arbitrarily. The main challenge with the KNN algorithm is to find the proper value of k which represents the number of neighbors. The best accuracy percentage was observed when the value of k lay between 5 and 20.

A. Equations

For K Neighbors Classifier the most common metric chosen to ascertain the distance between training and testing instance is the Euclidean distance, the formula for which is given by below.

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

It was found that, the feature-extraction step to ascertain the RGB values took the greatest amount of time. This was because although the x,y co-ordinates of the ground imagery could be determined beforehand, it was still required to assess the training or testing image for the pixel values surrounding the target pixel to cover the car in its entirety.

After the model was trained and made ready, few more sample images were generated, that would act as the validation or test dataset. These were then fed to the classifier to gauge the accuracy. The newly generated co-ordinates with the original co-ordinate system was compared to gauge the accuracy. The result obtained was visually represented by plotting the 'predicted' locations of cars on the original image file. This helped to gain an estimate of the accuracy of the model, at a glance.

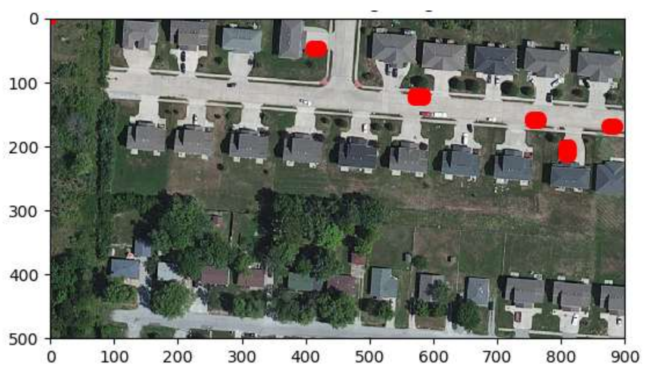

Fig. 2. Tested image showing presence and location of red cars highlighted in red, from original satellite image.

IV. CONCLUSION

After performing the analysis of the data sets by means of KNN classifier, a trained model was generated that was able to identify red cars in the test image and determine their location, with certain degree of success. An obvious drawback of the system was that as it relied on the RGB values of chosen pixels to train itself, it could be easily misled into making a faulty assumption by mistaking some part of the terrain for the car or fail to correctly identify a red car. Another drawback was that though the images were sufficiently cropped, the computation time was still found to be higher than acceptable as at certain stages each images pixel value had to be individually assessed.

To overcome these shortcomings, various methods can be considered to improve accuracy of classifying test instance:
1) Find best neighborhood size which produces more accurate results;
2) More accurate class probability estimation method can be used instead of simple voting [6].

Also, During the training period, KNN just simply stores the training instances and postpones most computations to classifying period, which leads to tremendous computational cost. But that can be overcome by moving some computational work from the classifying period to the training period, which could greatly reduce the computational cost as this part of computations will be reused when classifying new instances [4].

To improve the accuracy, few of the edge detection algorithms available as part of Python's OpenCV could also be utilized to accurately determine the boundaries of the object being studied. This would eliminate to a great extent the possibility of false positives coming up in the image due to objects sharing similar features with the targeted object. Combination of KNN algorithm with other classifiers such as Naive Bayes classifier and Support Vector Machines (SVM) could have demonstrated better utility and feasibility. KNN was the preferred approach here, owing to its simplicity [7].

For the system described, the accuracy varied between 65-85%. The variation it showed depended on factors such as if the ground image being tested or validated had any other pixel region having reddish hues.

References

[1] Aiman Moldagulova and Rosnafisah Bte. Sulaiman, (2017). Using KNN Algorithm for classification of textual documents, International Conference on Information Technology (ICIT), 8(1).

[2] Krit Inthajaki, Cattleya Duanggate, Bunyarit, Uyyanonvara, Stanislav S. Makhanov, Sarah Barman, (2011). Medical Image Blob Detection with feature stability and KNN claasification, Eighth International Joint Conference on Computer Science and Software Engineering (JCSSE)

[3] Adrian Calma, Tobias Reitmaier and Bernhard Sick, (2016). Resp-*k*NN: A probabilistic *k*-nearest neighbor classifier for sparsely labeled data, International Joint Conference on Neural Networks (IJCNN)

[4] Sun Bo Du Junping Gao Tian, (2009). Study on the Improvement of K-Nearest-Neighbor Algorithm, International Conference on Artificial Intelligence and Computational Intelligence

[5] Shiliang Sun Rongqing Huang, (2010). An Adaptive *k*-Nearest Neighbor Algorithm, International Conference on Fuzzy Systems and Knowledge Discovery, 7(1)

[6] Shweta Taneja, Charu Gupta, Kratika Goyal, Dharna Gureja, (2014). An Enhanced K-Nearest Neighbor Algorithm Using Information Gain and Clustering, International Conference on Advanced Computing & Communication Technologies, 4(1).

[7] Luis Tobıas, Aurelien Ducournau, Francois Rousseauy, Gregoire Merciery, Ronan Fablet (2016). Convolutional Neural Networks for Object Recognition on Mobile Devices: a Case Study, International Conference on Pattern Recognition (ICPR).