A machine learning approach to detect red cars in high resolution satellite images

Sai Swetha Kondubhatla
University of Florida
Gainesville, Florida
skondubhatla@ufl.edu

***Abstract*— Detecting objects in satellite imagery is very challenging, due to its cluttered background. The algorithm might mistake irrelevant objects to be target objects. This paper aims to detect all red cars in a high-resolution satellite imagery. In this paper I propose an effective method of object detection using KNN classifier. To deal with the huge training data efficiently, in the preprocessing step, red cars are extracted by building a small square around the red car in the training image. Features for the classification are the RGB values for every pixel in the extracted images. The KNN classifier is modelled with these RGB values. The results for the test image are the predicted coordinates of red cars in the test image. Comprehensive experiments on this proposed preprocessing step have demonstrated high accuracy.**

Keywords*— *Object Detection, KNN classifier, Satellite Images, red cars

I. INTRODUCTION

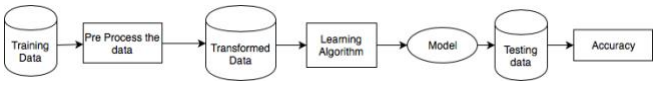
Object detection in images is considered to be an important application in the field of computer vision and machine learning. Satellite images are taken to be one of the most important data sources, due to their extensive coverage of land and objects. With new technology developments in the field of satellite sensors, high resolution satellite image data sets can be obtained and used in various fields. The purpose of object detection is to identify if the image contains the object of an interest type and determine the position information (x, y coordinates) of the predicted object in the image. In past, since 1980's object detection for satellite images is done by manually detecting the features by hand and then training the classifier. The disadvantage with this method is that, even if there is a small change in the surroundings, it might not predict the objects. These disadvantages and the increasing need of object detection in various fields led researchers in the field looking for more robust and effective approach. Detecting objects in high resolution satellite images have various applications in the field of homeland [1] and military [2] surveillance, intelligent traffic guiding systems [2] and many more, but yet remains a challenging task. The accuracy of object detection depends on the types of features extracted from the object. Various papers propose various algorithms for extracting objects from the satellite images. Zhao [2] proposed an algorithm to detect cars using its geometric boundary, front windshield and shadow of car as features into a Bayesian network. Liang et al [3] proposed an algorithm with histogram-oriented gradients as features to a multiple kernel Support vector machines. They revealed that gradient based features provide superior performance but increase the computational time

In this paper, I propose a simple preprocessing step to extract RGB features from a high-resolution satellite image with more than 30 million pixels to detect red cars. K nearest neighbors algorithm is adopted to learn the RGB values of the red cars. This paper contains the following sections. Section II describes some theoretical concepts used in this paper. Section III describes the implementation details of the algorithm. Section IV gives an overview of how experiment is carried out and the results of the experiment. Paper concludes in Section V.

II. THEORITICAL BACKGROUND

A. Machine learning

Machine learning algorithms helps deal with complex problems with the help of computer programs. It uses statistical techniques to learn the data without explicitly programming it. It can identify patterns and make decisions without human intervention. Depending on how algorithms are implemented, machine learning can be classified into 2 approaches: Supervised learning, unsupervised learning. Supervised learning is the task of mapping an input to an output, given sample input - output pairs. The two major tasks in supervised learning are Classification ad Regression. In unsupervised learning, we tend to learn the labels without having any prior knowledge. The major tasks in this are clustering and representation learning.



B. K nearest neighbors algorithm

K nearest neighbors is a simple classification technique which is used to classify the objects on the basis of most similar or K closest training samples in the feature space. The distance for each testing sample from all the training samples is calculated, and the majority vote of neighbors is used to classify the class of the object. The distance metric used for KNN classifier is Euclidean distance metric which is defined as follows:

$$d(x,y) = \sqrt{(x-y)^T \sum^{-1} (x-y)}$$

Where x and y are two instances and Σ^{-1} is the covariance matrix for x and y.

In this problem of detecting red cars in satellite images, we use supervised learning and the task is classification. We classify each pixel into one of the two classes – red, not red. We first preprocess the image, extract the features i.e. the RGB values for the transformed image, train the model with the extracted features, and then validate the model by cross validating the data for various folds and deciding the hyper parameters for the model (here, the hyper parameter is k) and then find the red cards for the test image.

III. DESIGN AND IMPLEMENTATION

The training image used is shown in Figure 1and is of size 6250 X 6250. Training the classifier model with the RGB

values of every pixel would provide incorrect results due to the sparsity of the red cars and density of not red objects such as trees, bushes, buildings, etc. In order to identify the red cars accurately, we need to do some preprocessing to the data, so that the red cards would be identified. I tried 2 different preprocessing methods. We are doing a preprocessing step since training 30 million pixels would increase the computational time and give incorrect results.

First approach is, train the data by picking $1/5^{th}$ of the image (160 X 160) randomly. This method is unreliable since, we cannot assure that the randomly picked image would contain any red cars. In that case, the model will be trained on all pixels which are not red, and it would not be able to detect the red pixels at all. To overcome this disadvantage, I propose a second approach for preprocessing. Let's define two classes. Red pixels have class 1 and not red pixels have class 0. As we know the ground truth values, all the values corresponding to ground truth coordinates will be of class 1 and rest all pixels will have a class 0. Now, build a small square of size p X p around this point and train just the small squares for every ground truth value. So, now the model will just learn the red pixels and some other pixels present in the square. Whenever the classifier sees a red pixel in the test image, it classifies as 1 which is finding the red cars in the test image. The experiment results shown in section IV will describe the appropriate k and p values.

Figure 1

IV. EXPERIMENTATION

As discussed above, for the second approach of preprocessing, we would be constructing a square of size p X p. Images for p = 3, p = 10, p = 50, p = 100 are shown in Figure 2. I performed various experiments to determine the hyper spectral parameters p, k using cross validation. Here, accuracy will not give the complete picture about the correctness of the model since, the red data is sparse, all the not red data would be identified and thereby giving an accuracy of 99%. The number of pixels classified as class 1 (red label) in the predicted label set would be the right metric to determine if at all there is any red car.

So, first we will construct a small square of size 10 X 10 around every ground truth coordinate, train the model with every pixel of all the squares with k neighbors and then validate the model by splitting the data in 70:30 ratio. We then perform cross validation of the 70% training data with number of folds as 10. The results after varying the k values are shown in **Table 1.**

Figure 2

A. Determining the k value for KNN algorithm.

We need to determine which k value would fit the data appropriately to give the best result during cross validation. Below Table 1, shows us the number of pixels which are actually of class 1 (red label) and number of predicted values with class 1. From the table we can say that, for $2 \leq k < 5$ the number of predicted pixels with class 1 match with the number of actual pixels with class 1. This is because, if k value is 1, the granularity is very small, so we might over fit the model since we have just one nearest neighbor. The data will therefore have high variance and low bias. For any $k \geq 5$, there won't be any change, since the red pixels are very sparse, the not red pixels will be more and the number of

votes for not red increase thereby classifying a red pixel as not red. The data has low variance and high bias. We will be underfitting the model for higher k values.

No of neighbors	# of pixels which are red	# of predicted pixels as red
1	4	8
2	16	11
3	8	6
4	11	1
5	11	0
10	8	0

Table 1

B. Determining the size of the constructed square

From the above Figure 2, we can see that, higher the p value, a greater number of not red pixels will be included. So, we will be training our model with a greater number of not red pixels and a smaller number of red pixels. So, model will classify the not red pixels very accurately and gets confused with a red pixel comes up because it hasn't seen the red pixel data yet.

Now, lower the p value, we will be training our model only will red values, so if any other value closest to red for example, a grey pixel comes up, it tends to classify it has red as it hasn't seen any other pixel except red. We are not giving enough data to train the model.

After performing various experiments, we found the best results when p values are between 10 and 50. For number of neighbors in the range 2 to 5 and p value in the range of 10 to 50, the average accuracy of the model is 98.083%

C. Choice of model

We had a choice of regression model, Probabilistic generative classifier and K nearest neighbors. The reason for not choosing regression model is that, since we have more than 30 million pixels, calculating the weight of every pixel would increase the computational time. The testing would be easier, but the training will still take a huge amount of time. Even after performing the preprocessing step, the minimum pixels to trained are 1000 for each ground truth, which is still very large.

The probabilistic generative classifier on the other hand, calculates the mean and variance of all the data. Since the red pixels are very less, the mean and variance will be calculated mostly on the not red pixels and will be biased. Therefore, classification of a red pixel might be difficult.

K nearest neighbors algorithm, intuitively calculates the neighbors based on the extracted features i.e. the RGB values and then classifies every pixel of the test image using the votes of k nearest neighbors. With a diverse value of red and not red in a training image, this classifier will work better than other two with minimal computational time. This algorithm is robust to noisy data and will work better when training data is large. The disadvantage of using this algorithm is that, even for classifying one pixel value, the KNN algorithm finds the distance between every training pixel and thereby increases the computational time. In other methods such as regression, we create a function and check which class it belongs to which takes linear time. For KNN algorithm it takes $O(N^2)$ time. Considering the computational time, size of dataset, and sparsity of classes, I chose K nearest neighbors algorithm.

Figure 3

The test results are shown in Figure 3. The coordinates of red cars detected in the test image, are the coordinates of every red pixel. If 20 pixels represent a red car, all 20 pixels would be there in the result. Post processing of combining all the nearest pixels and finding a metric to calculate the location of red car, like midpoint of all the pixels or drawing a boundary around the car can be done for more exact location of the car in test image.

V. Conclusion

In this paper, a preprocessing step has been presented to reduce the computational time for training a large dataset using K nearest neighbors. The problems associated with choosing another classifier other than K nearest neighbors has been addressed. Future work can be directed to use any other version of KNN algorithm such as KNN with a different distance metric, fuzzy KNN algorithm [4], probabilistic KNN algorithm [5], KNN with spatial constraints etc.

References

[1] L. Eikvil, L. Aurdal, H. Koren, "Classification-based vehicle detection in high-resolution satellite images", *J. Photogramm. Remote Sens.*, vol. 64, no. 1, pp. 65-72, Jan. 2009.

[2] T. Zhao, R. Nevatia, "Car detection in low resolution aerial images", *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 1, pp. 710-717, 2001.

[3] P. Liang, G. Teodoro, H. Ling, E. Blasch, G. Chen, L. Bai, "Multiple Kernel Learning for vehicle detection in wide area motion imagery", *Proc. 15th Int. Conf. Inf. Fusion*, pp. 1629-1636, 2012.

[4] J. M. Keller, M. R. Gray and J. A. Givens, "A fuzzy K-nearest neighbor algorithm," in IEEE Trans- actions on Systems, Man, and Cybernetics, vol. SMC-15, no. 4, pp. 580-585, July-Aug. 1985. doi: 10.1109/TSMC.1985.6313426

[5] H. Frigui and P. Gader, "Detection and Discrimination of Land Mines in Ground-Penetrating Radar Based on Edge Histogram Descriptors and a PossibilisticK-Nearest Neighbor Classifier," in IEEE Transactions on Fuzzy Systems, vol. 17, no. 1, pp. 185-199, Feb. 2009. doi: 10.1109/TFUZZ.2008.2005249