Machine Learning Methods for Ship Detection in Satellite Images

Yifan Li yil150@ucsd.edu
Huadong Zhang huz095@ucsd.edu
Qianfeng Guo qig020@ucsd.edu
Xiaoshi Li xil758@ucsd.edu

***Abstract*— In this project, we compared the performances of several machine learning methods on binary classification task, then the results were improved by HOG feature extraction in data preprocessing. Furthermore, we implemented convolutional neural network (CNN) and reached an accuracy of 99%. Based on the CNN model, we accurately detected all ships in satellite images of San Francisco Bay Area with bounding boxes using sliding windows detection algorithm.**

Keywords: image classification, object detection, HOG feature extraction, convolutional neural network, sliding windows detection

I. INTRODUCTION

Satellite imagery is the scanning of the earth by satellites, which uses different kinds of sensors to collect electromagnetic radiation reflected from the earth. When satellite images are made, the invisible types of light perceived by satellite sensors are assigned a visible color. Satellite images are useful because different surfaces and objects can be identified by the way they react to radiation. Also, objects react differently to different wavelengths of radiation. This flood of new imagery is outgrowing the ability for organizations to manually look at each image that gets captured, so there is a need for machine learning and computer vision algorithms to help automate the analysis process[1].

Though CNN has been proven to be a powerful tool in image classification task, given the relatively small dataset in this project, it is still reasonable to tentatively try some traditional machine learning methods since they tend to run faster than CNN. It is much feasible if they could achieve similar accuracy with shorter runtime compared with CNN. Therefore, we firstly compared 4 different machine learning methods on binary classification task, then implemented CNN and made further comparison. Since CNN finally outperformed with an high accuracy of 99%, based on the model, we used sliding windows detection to accurately detect all ships in satellite images of San Francisco Bay Area.

II. DATA

The dataset consists of image chips extracted from Planet satellite imagery collected over the San Francisco Bay Area. It includes 2800 80×80 RGB images, 700 of which are ship images labeled as "ship", while the 2100 left are no-ship images labeled as "no ship". Image chips were derived from PlanetScope full-frame visual scene products, which are orthorectified to a 3-meter pixel size[2].

A. Preprocessing

- Images: each image in this dataset is represented as a 19200×1 vector. We transformed each vector into a 3×80×80 RGB vector to make the image dataset a 2800×3×80×80 4-dimension numpy array.
- Labels: labels were transformed to a 2800×1 1-dimension numpy array with 700 1's associated with ship images and 2100 0's associated with no-ship images.

B. Visualization

As an example, we show the digital information of the first image in 3 different channels separately.

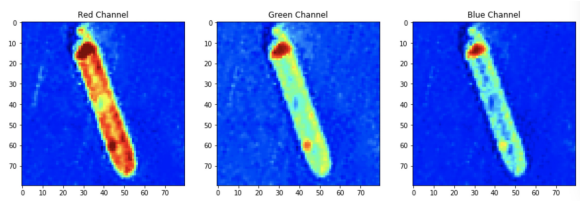


Fig. 1: Data Visualization

C. Histogram of Oriented Gradient (HOG)

The essential idea behind the HOG feature extraction is that local object appearance and shape within an image can be described by the distribution of intensity gradients or edge directions. The implementation of these descriptors can be achieved by dividing the image into small connected regions called cells, and for each cell compiling a histogram of gradient directions or edge orientations for the pixels within the cell. The combination of these histograms then represents the descriptor. For improved accuracy, the local histograms can be contrast-normalized by calculating a measure of the intensity across a larger region of the image called a block, and then using this value to normalize all cells within the block[3].

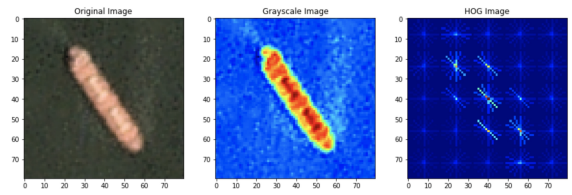


Fig. 2: HOG Feature Extraction

III. Physical and Mathematical Framework

A. Machine Learning Methods

We firstly compared the performances of the following 4 methods using Scikit-learn framework.

1) Logistic Regression (LR): For binary classification problem, LR is a classic algorithm. It is named for the function used at the core of the method, logistic function:

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}} \tag{1}$$

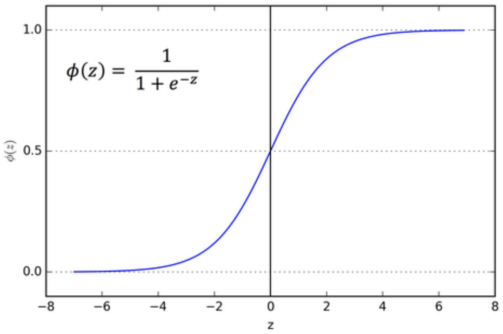


Fig. 3: Logistic(Sigmoid) Function

Basically, it learns a parameter θ to separate the data points using 0.5 as a threshold by log-loss function:

$$J(\theta) = \sum_{i=1}^{m} y^i log(h_\theta(x)) + (1 - y^i) log(1 - h_\theta(x)) \tag{2}$$

where y^i indicates the i^{th} label in the training data. When $h_\theta(x) \geq 0.5$, the model should predict 1 as a positive sample, otherwise 0 as a negative one.

2) Random Forest (RF): RF is a popular machine learning method which consists of Classification and Regression Trees (CART) proposed by Breiman et al. [4]. In this project, obviously, RF was used as a classification method.

It is straight-forward to implement the CART. As described in [5], the input dataset D which consists of N samples of d-dimension is first considered as the root of the classification tree. Then, it loops through all features to find the best feature to split the dataset into 2 subsets. There are two metrics that are mostly used to determine which specific feature to split the dataset: $Gini\ Impurity$ and $Information\ Gain$. Gini impurity measures the probability of the data, which is randomly chosen, being misclassified if the label is also randomly assigned:

$$Gini(f) = \sum_{i=1}^{J} f_i(1 - f_i), \quad i = 1, 2, ..., J \tag{3}$$

where J is the number of classes and f_i is the portion of data which belongs to class i. Information gain is calculated based on the concept of entropy which gives similar results as Gini impurity:

$$Entropy = -\sum_{i=1}^{J} p_i log_2 p_i, \quad i = 1, 2, ..., J \tag{4}$$

where p_i is the probability of class i and the sum of p_i should be added up to 1. The information gain is calculated as the difference between the entropy of the node and its children:

$$IG = Entropy(T) - Entropy(T|a) \tag{5}$$

The dataset keeps splitting based on either of these two metrics until all the nodes cannot be splitted anymore or a specific threshold is reached[6].

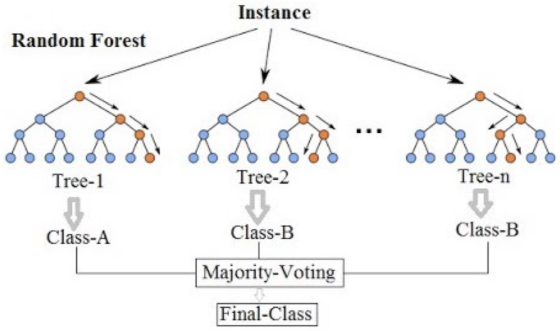


Fig. 4: Illustration of Random Forest

3) k-Nearest Neighbors (k-NN): k-NN is a non-parametric lazy learning algorithm, which means that it does not make any assumptions on the underlying data distribution, also, it does not use the training data points to do any generalization. At its most basic level, it is essentially classification by finding the most similar data points in the training data, and making an educated guess based on their classifications.

Given data with N unique features, the feature vector would be a vector of length N, where entry I of the vector represents that data points value for feature I. Each feature vector can thus be thought of as a point in R^N.

Once we have formed our training data-set, which is represented as an M×N matrix where M is the number of data points and N is the number of features, we can now begin classifying. For each classification query, the gist of k-NN is to:

- compute a distance value between the item to be classified and every item in the training data-set
- pick the k closest data points (the items with the k lowest distances)
- conduct a majority vote among those data points the dominating classification in that pool is decided as the final classification

After doing all of the above and deciding on a metric, the result of the k-NN algorithm is a decision boundary that partitions R^N into sections. Each section represents a class in the classification problem. The boundaries need not be formed with actual training examples, instead, they are

calculated using the distance metric and the available training points[7].

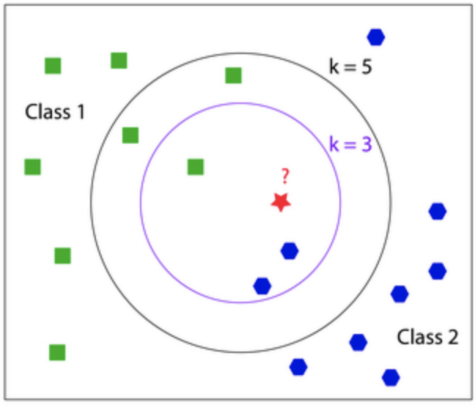


Fig. 5: Illustration of k-Nearest Neighbors

4) Support Vector Machine (SVM): SVM is a common binary classifier used in supervised machine learning. The algorithm is based on SV learning which is a method to find near optimal of functions without knowing its statistical distribution[8]. This method enables predictions only depend on training data. Since support vector machines are binary non-probability classifiers, the idea of SVM is to find a separating hyperplane between two classes as illustrated in Fig.1. Assuming there exists such a separating hyperplane expressed as:

$$y = w^T + b \tag{6}$$

with classes labeled by +1 for $y \geq 0$ and 1 for $y < 0$.

The distance between two classes can be represented by:

$$\gamma = \frac{|w^T + b|}{||w||^2} \tag{7}$$

Given that the aim of SVMs is to find the separating hyperplane with largest margin, the optimal solution for SVM is:

$$\arg\max \frac{\gamma}{2} \tag{8}$$

which equals to:

$$\arg\min \frac{||w||^2}{2} \tag{9}$$

Here the hyperplane is assumed to be a canonical hyperplane with property:

$$t_n \cdot (w^T + b) \geq 1 \tag{10}$$

where t_n is the label corresponding to each data point.

In many cases, samples are not linearly separable. Then kernel trick is needed to transform data into higher dimentional feature spaces to find separating hyperplane.

In SVMs algorithm, only data points at boundary, which are called support vectors, have influence on the hyperplane choosing. By only taking these vectors into consideration, computing complexity is greatly reduced[9].

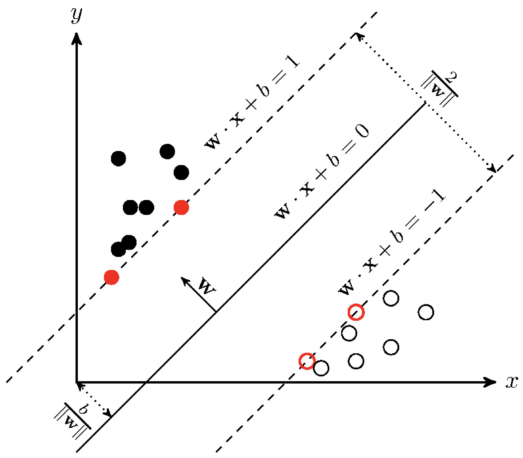


Fig. 6: A Linear Hyperplane Learned by Training SVM in 2-dimension[10]

IV. Convolutional Neural Network (CNN)

After HOG feature extraction, former methods achieved pretty good results. Since deep learning approaches have been proven the superiority in image classification problem, we further reached a better accuracy by implementing CNN.

A. Model Design

Given this relatively small dataset and binary classification task that is not too complex, our network has 4 hidden layers, each of which contains:

- Convolutional layer: to extract input features by adding elementwise product between input regions and multiple filters;
- ReLU activation: a non-linear transformation of feature space to solve linearly inseparable problems;
- Pooling layer: 2*2 maxpooling to enhance invariance and decrease parameters in spatial dimensions;
- Dropout: to avoid overfitting.

and two fully-connected layers to learn a function from the feature combinations of former layers and further do classification. We used cross-entropy as loss function, which is defined as:

$$L = \frac{1}{n}\sum_{n=1}^{N}[y_n log\hat{y_n} + (1 - y_n)log(1 - \hat{y_n})] \tag{11}$$

where N represents the number of samples, $\hat{y_n}$ is the predicted label of sample n, y_n is the target label of the sample.

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 80, 80, 32)	896
max_pooling2d_1 (MaxPooling2	(None, 40, 40, 32)	0
dropout_1 (Dropout)	(None, 40, 40, 32)	0
conv2d_2 (Conv2D)	(None, 40, 40, 32)	9248
max_pooling2d_2 (MaxPooling2	(None, 20, 20, 32)	0
dropout_2 (Dropout)	(None, 20, 20, 32)	0
conv2d_3 (Conv2D)	(None, 20, 20, 32)	9248
max_pooling2d_3 (MaxPooling2	(None, 10, 10, 32)	0
dropout_3 (Dropout)	(None, 10, 10, 32)	0
conv2d_4 (Conv2D)	(None, 10, 10, 32)	102432
max_pooling2d_4 (MaxPooling2	(None, 5, 5, 32)	0
dropout_4 (Dropout)	(None, 5, 5, 32)	0
flatten_1 (Flatten)	(None, 800)	0
dense_1 (Dense)	(None, 512)	410112
dropout_5 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 2)	1026

Total params: 532,962
Trainable params: 532,962
Non-trainable params: 0

Fig. 7: Architecture of CNN Model

B. Optimization Setup

1) Optimizer: We chose AdamOptimizer which is a combination of momentum and RMSprop. In addition to storing an exponentially decaying average of past squared gradients v_t like Adadelta and RMSprop, Adam also keeps an exponentially decaying average of past gradients m_t, similar to momentum. We compute the decaying averages of past and past squared gradients m_t and v_t respectively as follows:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{12}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) {g_t}^2 \tag{13}$$

where g_t is partial derivative of the objective function, m_t and v_t are estimates of the first moment (the mean) and the second moment (the uncentered variance) of the gradients respectively, hence the name of the method. Then bias-corrected first and second moment estimates are computed as:

$$\hat{m_t} = \frac{m_t}{1 - \beta_1^t} \tag{14}$$

$$\hat{v_t} = \frac{v_t}{1 - \beta_2^t} \tag{15}$$

Then the parameters are updated using Adam rule:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v_t}} + \epsilon} \hat{m_t} \tag{16}$$

Usually, default values are proposed of 0.9 for β_1, 0.999 for β_2, and 10^{-8} for ϵ[11].

2) Mini-batch: We used mini-batch and set the size to be 32, which could much faster gradient descent.

V. Sliding Windows Detection

The last part of this project was to detect all ships and their corresponding positions based on CNN model using sliding windows detection, which is an effective tool to localize exactly the position of target object. The principle is scanning an image step by step with a predetermined window. For each of these windows, we would normally take the window region and apply an aforementioned image classifier to determine if the window has an object. Combining with image pyramids, it is possible to recognize objects at varying scales and locations in the image. There are two parameters needed to be controlled, window size and stride. Window size can be predetermined by the knowledge or varying scales to fit different objects. Stride will affect the accuracy of the objects position. The smaller stride is, the more windows needed to be examined and the better accuracy.

A. Parameters Setup

In this project, ships have almost uniform size. Therefore, we set the windows size to be 80×80 pixels, which is the same as the images in training set. As for the stride, considering the trade-off between efficiency and accuracy, we set it to be 10 pixels. If the predicted value of a window, which indicates the possibility of a positive sample, is larger than 0.9, then we consider an existence of ship.

B. Non-Maxima Suppression (NMS)

In the procedure, there are many windows indicating a same target. However, we just want to find a box with best fit. So, we applied NMS method to remove the overlap windows. NMS method makes a trade-off on overlap by calculating the intersection over union (IoU) and can be expressed as[12]:

$$overlap = \frac{area(window_1 \cap window_2)}{area(window_1 \cup window_2)} > threshold \tag{17}$$

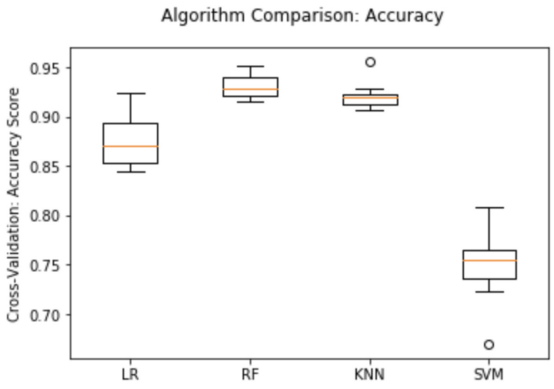
If the result is larger than threshold, then $window_1$ and $window_2$ have a same object. In our project, we set 0.2 as the threshold to avoid overlaps.

VI. Results

A. Machine Learning Methods Comparison

1) Preliminary Result: In our first attempt, we directly used image vectors as input. However, as shown below, the result was not satisfying. Then we figured out that image vectors were not good representation of image features. Thus, in the next, we implemented HOG feature extraction in data preprocessing to improve the result.

```
Compare Multiple Classifiers:

K-Fold Cross-Validation Accuracy:

LR: 0.875893 (0.025707)
RF: 0.930804 (0.011512)
KNN: 0.921429 (0.012969)
SVM: 0.750893 (0.035926)
```

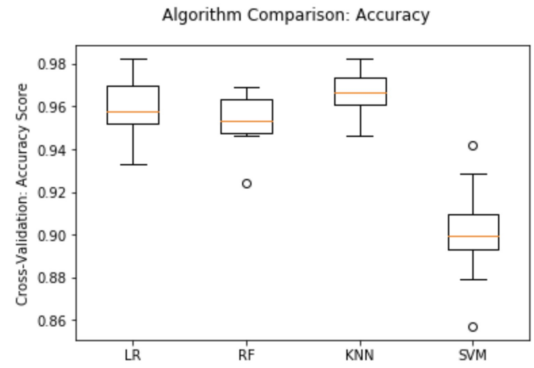



```
LR = Logistic Regression
RF = Random Forest
KNN = K-Nearest Neighbors
SVM = Support Vector Machine
Runtime: 1865.1310410499573s
```

Fig. 8: Preliminary Result

2) Improvement by HOG Feature Extraction: After transforming image vectors into HOG features as input, we significantly improved not only the accuracy, but runtime as well.

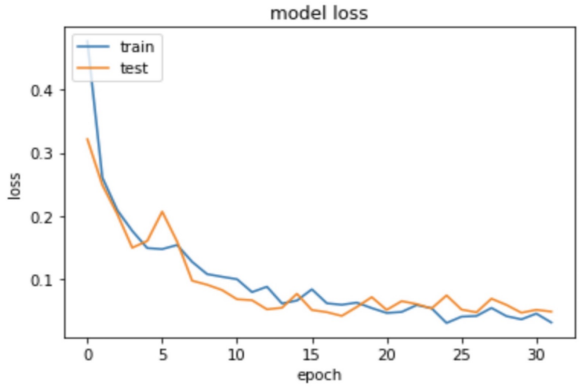
```
Compare Multiple Classifiers:

K-Fold Cross-Validation Accuracy:

LR: 0.958929 (0.014644)
RF: 0.953571 (0.012658)
KNN: 0.966518 (0.010229)
SVM: 0.900893 (0.022746)
```



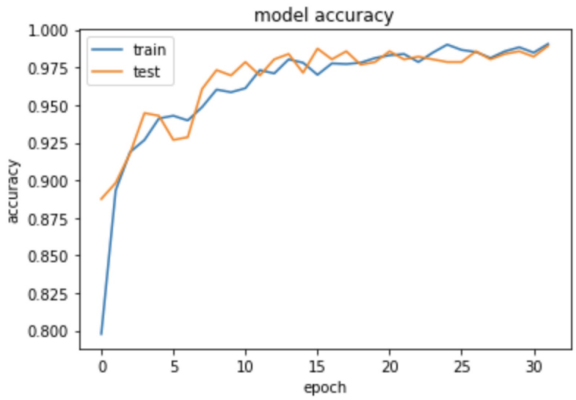

```
LR = Logistic Regression
RF = Random Forest
KNN = K-Nearest Neighbors
SVM = Support Vector Machine
Runtime: 27.46461319923401s
```

Fig. 9: Improvement after HOG Extraction

3) Convolutional Neural Network: After 32 epochs, our model could reach a 99% accuracy.



(a)



(b)

Fig. 10: (a) Model Loss (b) Model Accuracy

4) Sliding Windows Detection: Based on the CNN model, the final result after applying NMS method is shown below.

Fig. 11: Ship Detection with Bounding Boxes

VII. Conclusion

In this project, we compared several machine learning methods on binary classification task then improved their accuracy and also runtime by HOG feature extraction. Furthermore, the CNN model we built outperformed with a 99%

accuracy after 32 epochs, which again proved the superiority of CNN on image classification task. Then based on the model, we used sliding windows detection to capture all ships in satellite images with bounding boxes.

VIII. Future Work

Though sliding windows detection was quite feasible in this project, if given images with large scale, we still need a well-labeled dataset for more advanced detection algorithms like YOLO, since sliding windows detection tends to be less efficient with longer runtime in situations like this.

References

[1] Available from: *http://globalforestlink.com/tutorials/how-does-satellite-imaging-work*.

[2] Available from: *https://www.kaggle.com/rhammell/ships-in-satellite-imagery*.

[3] Luo, H.-b., et al. *A method for real-time implementation of HOG feature extraction. in International Symposium on Photoelectronic Detection and Imaging 2011: Advances in Infrared Imaging and Applications*. International Society for Optics and Photonics. 2011.

[4] Breiman, L., *Machine learning*, 45, pp. 5-32. Kluwer Academic Publishers. 2001.

[5] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and regression trees*. Monterey, CA, 1984.

[6] Chen, Y., et al., *Reimplementation of Source Localization in an Ocean Waveguide using Supervised Machine Learning*. 2017.

[7] Available from: *https://towardsdatascience.com/introduction-to-k-nearest-neighbors-3b534bb11d26*.

[8] Steinwart, I. and A. Christmann, *Support vector machines*. Springer Science and Business Media. 2008.

[9] Zhang, Z., et al., *Source localization in an ocean waveguide using supervised machine learning*. 2017.

[10] Lowery, A., et al., *MicroArray Technology-Expression Profiling of MRNA and MicroRNA in Breast Cancer*. INTECH Open Access Publisher. 2011.

[11] Diederik P. Kingma and Jimmy Ba. Adam, *A Method for Stochastic Optimization*. ICLR. 2014.

[12] Available from: *http://www.cs.utoronto.ca/~fidler/slides/CSC420/lecture17.pdf*.