CS 6375 Machine Learning
Project Report
Airbus Ship Detection Challenge (Kaggle)

Shruti Agrawal, Maleeha S. Koul, Mahasweta Sarma, Abhisek Banerjee

1. Introduction

Shipping Traffic is a very common and a very dangerous issue in the sea. As the number of ships sailing in the sea increases, the number of violations and threats also increases, like accidents, drug trafficking, illegal fishing etc. This problem has lead to different organizations trying their best to efficiently have a closer watch on the ships that are sailing in the open sea. Airbus is one such organization that is trying to closely detect ships in the sea by utilizing detailed monitoring services for finding intricate details and wide coverage of the entire sea for ship detection, so that they can help the maritime industry to anticipate the threats and trigger alerts before there is any damage. Airbus has worked for a long time in the area of automatic ship detection but their results are not satisfactory. So our project is based on the challenge put by Airbus on Kaggle for creating a model that automatically detects all the ships in satellite images with best accuracy and speed.

We have used four algorithms for this project which have given us the best results, SVM, Neural Network, AdaBoost and XGBoost. The input to our algorithms was the data given by Kaggle, where there were satellite images of the sea and their encoded pixels if there was a ship(ships) in that image. We first separated the data into positive and negative examples with the help of the encoded pixels of the images. We then processed the images to filter the regions of interest. We cropped the regions of interest and stored in to extract their feature values and their class label. Finally, we used our four models to find the most accurate results.

This was a joint project and all the work showcased here are performed by Mahasweta Sarma, Shruti Agrawal, Maleeha Shabeer Koul and Abhisek Banerjee.

2. Related Work

Many different solutions to the same problem have been tried before by kagglers and data science enthusiasts. Most of them use tensorflow backend for making the predictions. The image processing has been done using keras. The results of the related work are viable and satisfactory. Our experiment has been to use a different library for image processing and testing various models for training apart from Convolutional Neural Networks which has been the most common approach to this problem before. We have used various boosting algorithms along with non linear classifiers for making predictions. Similar work has been done in real-time prediction of traffic signals where the same approach has been used. The real-time prediction of traffic signals uses various sign board images and traffic signal images to feed to the training model for learning. The true positives and false negatives were separated after extracting the regions of interest from the images. The same approach has been used in our project and has provided satisfactory results.

3. Dataset Description

To perform the needed predictions, we have taken the data set from the Kaggle competition. The data set has been provided by Airbus. Airbus offers many maritime services and manufactures and runs ships for transport and cargo. The data set is in .jpg format. The data set comprises of satellite images which may or may not contain ships. Ships within and across images may differ in size and be located in open sea, at docks, marinas etc.

The test and training data sets on which the model was trained and evaluated have been taken as a subset of the greater dataset (which is about 200,000 images). The image data was preprocessed and converted to a csv file of HOG features of the cropped image (Regions of Interest from the whole image) and the class label. The classification is linear and has two class labels: 0 and 1 where 0 corresponds to absence of ships in the image and 1 corresponds to the ship being detected.

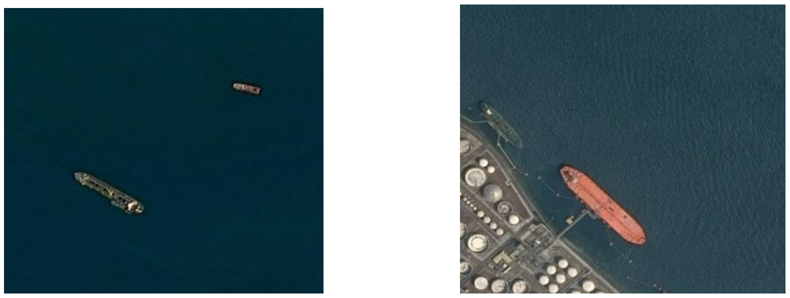
Fig 1. Sample images from the dataset

4. Pre-processing techniques

Since the data is purely image data in .jpg format, we used various image processing techniques. Image pre-processing can significantly increase the reliability of an optical inspection. Several filter operations which intensify or reduce certain image details enable an easier or faster evaluation. Normalization, converting image to different color spaces, edge filters etc. are some examples of the pre processing.
A few pre-processing techniques were applied to the images and Bilateral filter worked best for extracting the features. Images are processed using Open CV Library.

Bilateral Filter:

Bilateral filter is popular blurring technique which is highly effective in noise removal while keeping edges sharp. After applying other filters like gaussian filter, we saw that it takes the neighborhood around the pixel and find its gaussian weighted average. This gaussian filter is a function of space alone, so it blurs the edges also, which we don't want to do. Bilateral filter takes a gaussian filter in space, but also one more gaussian filter which is a function of pixel difference. Gaussian function of space make sure only nearby pixels are considered for blurring while gaussian function of intensity difference make sure only those pixels with similar intensity to central pixel is considered for blurring. So it preserves the edges since pixels at edges will have large intensity variation.

Contours:

Contours simply can be defined as a curve joining all the continuous points along the boundary. The curve is supposed to be having the same color or intensity. The contours are a useful tool for shape analysis and object detection and recognition. Since the primary goal of this project is object detection, contours from Open CV was the most useful function for cropping the regions of interests (ships in the images).

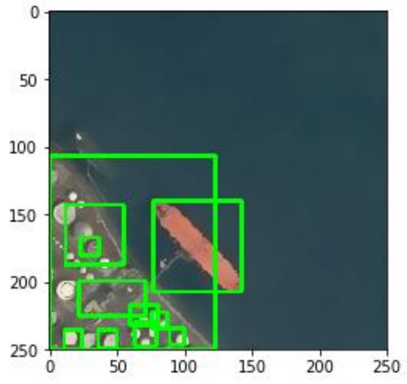

Fig 2. Image with contours before applying training model to identify regions of interest

5. Feature Extraction

For extracting the best features, we applied **HOG** on the images. HOG was applied again using Open CV.
Histogram of Oriented Gradients: The histogram of oriented gradients (HOG) is a feature descriptor used in computer vision and image processing for the purpose of object detection. The technique counts occurrences of gradient orientation in localized portions of an image. The essential thought behind the histogram of oriented gradients descriptor is that local object appearance and shape within an image can be described by the distribution of intensity gradients or edge directions. The image is divided into small connected regions called cells, and for the pixels within each cell, a histogram of gradient directions is compiled. The descriptor is then the concatenation of these histograms. 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. This normalization results in better invariance to changes in illumination and shadowing.
From the HOG features, a .csv file for the training images was created which was used for training the models.

6. Proposed Solution and Methods

Our approach first started with pre-processing the images that were separated as positive datasets. Image processing techniques like bilateral filter, contour etc. were used to extract the regions of interest (ships in images) and remove any noisy edges. For image processing, Open CV was used. A detailed description of the image processing phase can be found above. After image processing, came feature extraction, where we used HOG images and extracted around 2916 feature attributes of the images along with their class labels. Our next step was to train our models based on these features. We fed the csv file having all the extracted features and the class label (0 when ship is being detected and 1 when no ship was being detected in the image) as a data frame and used four classifiers to compute the accuracy on train and test data. Below are the four methods that were used to create out model:

I. **SVM** : SVM stands for Support Vector Machine. It is a supervised machine learning algorithm which can be used for regression as well as classification. It makes use of the kernel parameter in order to transform data and then separate the data using the most optimal decision boundary. The reason we chose SVM is since SVM is a supervised learning algorithm, it requires clean and annotated data in order to have good results. As we had already done preprocessing, feature extraction and image cropping on our dataset, we knew this is one of the classifiers that would give us a decent accuracy. In machine learning , support vector machines (SVMs, also support vector networks) are supervised learning models with associated learning algorithms that analyze data and recognize patterns, used for classification and regression analysis . An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. Intuitively, a good separation is achieved by a hyperplane that has the largest distance to the nearest training-data point of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on.

II. **Neural Network** : Neural Network is one of the complex models of machine learning, which means that for different subsets of data they have varying outputs. But since it has hidden layers, it brings out accurate results. Neural Nets are widely used in image datasets because each hidden layer can be utilized to represent one or more features. The computer can see the images in way that we as users cannot, if we transform our images into numbers.
An MLP is a network of simple neurons called perceptrons. The basic concept of a single perceptron was introduced by Rosenblatt in 1958. The perceptron computes a single output from multiple real-valued inputs by forming a linear combination according to its input weights and then possibly putting the output through some nonlinear activation function. A single perceptron is not very useful because of its limited mapping ability. No matter what activation function is used, the perceptron is only able to represent an oriented ridge-like function. The perceptrons can, however, be used as building blocks of a larger, much more practical structure. A typical multilayer perceptron (MLP) network consists of a set of source nodes forming the input

layer, one or more hidden layers of computation nodes, and an output layer of nodes. The input signal propagates through the network layer-by-layer.

III. **Adaboost** : Adaboost is the short form of adaptive boosting. It is one of the most popular boosting algorithms that goes by the concept that a number of weak classifiers can combine and form a strong classifier. Adaboost is a boosting algorithm which combines weak classifiers like decision tree of depth '1' or stump to give rise to a strong classifier. The strong classifier hence provides a combination of the weak hypothesis and gives rise to a strong hypothesis.
Since our data was pre-arranged and adaboost has a reputation of being sensitive to noisy data, we decided to use it as our third model.

IV. **XGBoost** : XGBoost is the go to algorithm if we want our model to give the best results. It stands for Extreme Gradient Boosting. It has the same concept as Adaboost but it is superfast and give better results, which is why we could not help but use it as one of our models and interestingly it did give us the best results.
The implementation of XGBoost offers several advanced features for model tuning, computing environments and algorithm enhancement. It is capable of performing the three main forms of gradient boosting (of which we have used Gradient Boosting (GB) and it is robust enough to support fine tuning and addition of regularization parameters.
This algorithm is developed with both deep consideration in terms of systems optimization and principles in machine learning**.** This algorithm is again a boosting algorithm which combines weak hypothesis to give a strong one. It works on the concept of gradient descend algorithm. It was developed by Tianqi Chen and now is part of a wider collection of open-source libraries developed by the Distributed Machine Learning Community (DMLC). XGBoost has been a winning algorithm for many Kaggle competitions and as such, it also gives us the best results for our project among the other models.

6.b. Methodology

1. First the entire data set was analyzed for deciding on the approach to the problem statement.
2. The data set downloaded from Kaggle came with a train_set segmentations file which divided the train data into classes. The two classes in this case were one with the ship in it and the other with no ships.
3. Since the size of data set was quite large (200,000 train images), the available computation was not able to handle the amount of data. For this reason, we took a subset of images from the data set for train as well as test.
3. After dividing the data into positive and negative set of images, we used image processing to process the image for extraction of the useful features (feature extraction). This was done using open cv library and the pre-processing techniques have been described above.
4. After pre-processing, we used contours on the images to identify the regions of interest.
5. The regions of interest had to be separated into positives and negatives to eliminate the false negatives and false positives.
6. After extracting the features, we created a data frame of the extracted features which came to have 2916 features. The last column contains the class labels viz., 0 for images with ships and 1 for images without ships.
7. The csv file created was used for training the models.
8. Each of the model was tuned for the best parameters using Grid Search.
9. The model was saved using pickle.
10. Each of the saved models were evaluated for accuracy using multiple model evaluation parameters like F1-score, precision, recall, accuracy score and RoC with area under curve.
11. The best model (XGBoost in our project because it gave the best accuracy) is used to make real-time predictions on any test image. The code for demo performs the final task.

7. Experimental results and analysis

I. Pre-processing experiments:

MSER (Maximally stable extremal regions) was also used on the images for pre-processing but it did not give the desired results. **Contours** worked much better in making the bounding rectangles and hence the same was used for pre-processing and extracting regions of interest.

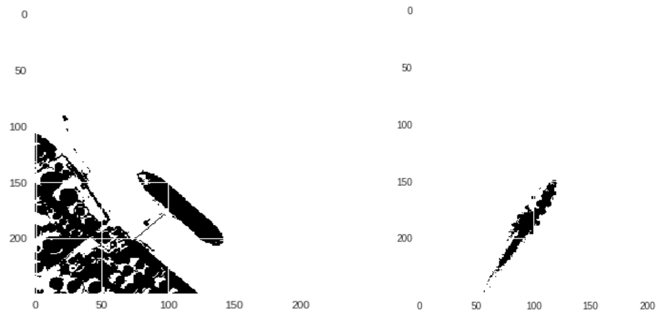

Fig 2. Samples of thresholded images

II. Model selection and evaluation:

All models were tuned over hyper parameters using Grid Search (grid search from Scikit Learn). The best parameters have been selected and evaluated using multiple evaluation metrics: precision, recall, f1-score and support.

Below are the model evaluations for the best set of parameters after grid search for each model that was trained on the training data.

Model 1: Support Vector Machine with 'rbf' kernel

Best parameters: SVC(C=10, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto', kernel='sigmoid',
  max_iter=1000, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)

```
             precision    recall  f1-score   support

          0       0.76      0.22      0.34      1405
          1       0.59      0.94      0.73      1692

avg / total       0.67      0.61      0.55      3097
```

Detailed confusion matrix:

```
[[ 305 1100]
 [  95 1597]]
```

Accuracy Score:

```
0.6141427187600904
```

Accuracy in %age: 61.4%

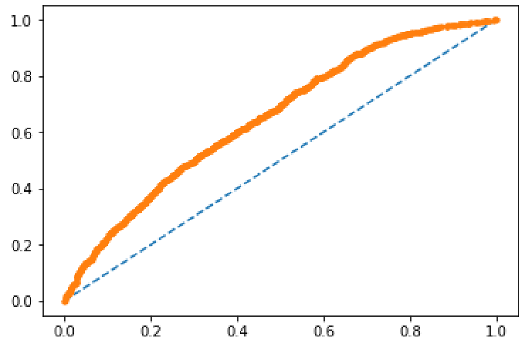

Fig 3. **ROC curve** for SVM

Area Under Curve: 0.654

Model 2: Multi-Layer Perceptron

Best parameters: MLPClassifier(activation='relu', alpha=0.001, batch_size='auto', beta_1=0.9, beta_2=0.999, early_stopping=False, epsilon=1e-08, hidden_layer_sizes=(10, 20), learning_rate='constant', learning_rate_init=0.001, max_iter=100, momentum=0.9, nesterovs_momentum=True, power_t=0.5, random_state=None, shuffle=True, solver='lbfgs', tol=0.0001, validation_fraction=0.1, verbose=False, warm_start=False)

```
             precision    recall  f1-score   support

          0       0.65      0.63      0.64      1405
          1       0.70      0.72      0.71      1692

avg / total       0.68      0.68      0.68      3097
```

Detailed confusion matrix:

```
[[ 882  523]
 [ 470 1222]]
```

Accuracy Score:

0.6793671294801421

Accuracy in %age: 67.9%

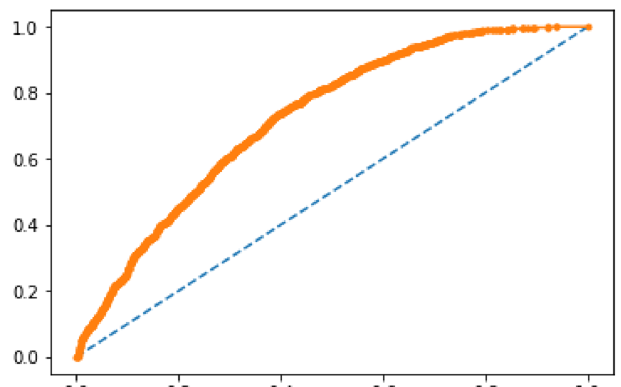

Fig 4. **ROC curve** for MLP

Area Under Curve: 0.725

Model 3: Adaboost Classifier

Best parameters: AdaBoostClassifier(algorithm='SAMME.R', base_estimator=None,
learning_rate=0.2, n_estimators=350, random_state=None)

```
             precision    recall  f1-score   support

          0       0.69      0.60      0.64      1405
          1       0.70      0.78      0.73      1692

avg / total       0.69      0.69      0.69      3097
```

Detailed confusion matrix:

```
[[ 836  569]
 [ 380 1312]]
```

Accuracy Score:

0.6935744268647078

Accuracy in %age: 69.3%

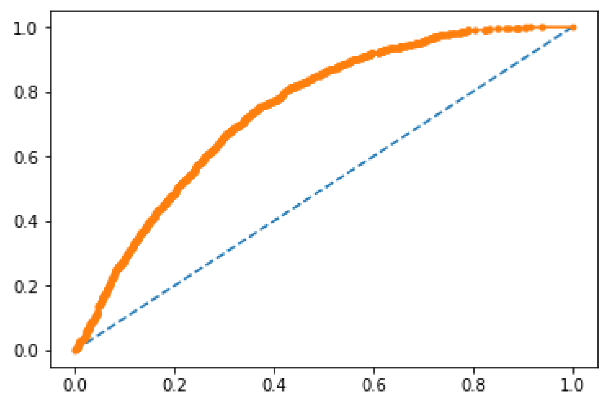

Fig 5. **ROC curve** for AdaBoost

Area Under Curve: 0.745

Model 4: XG Boost

Best parameters: XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
    colsample_bytree=1, gamma=0, learning_rate=0.09, max_delta_step=0,
    max_depth=3, min_child_weight=1, missing=nan, n_estimators=300,
    n_jobs=1, nthread=None, objective='binary:logistic', random_state=0,
    reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
    silent=True, subsample=1

XGBoost gave the best accuracy on the test data set. The results with the ROC curve has been plotted for the prediction as below:

```
             precision    recall  f1-score   support

          0       0.71      0.61      0.66      1405
          1       0.71      0.79      0.75      1692

avg / total       0.71      0.71      0.71      3097

  Detailed confusion matrix:
               [[ 862  543]
                [ 348 1344]]
  Accuracy Score:
          0.7123022279625444

  Accuracy %age: 71.2%
```

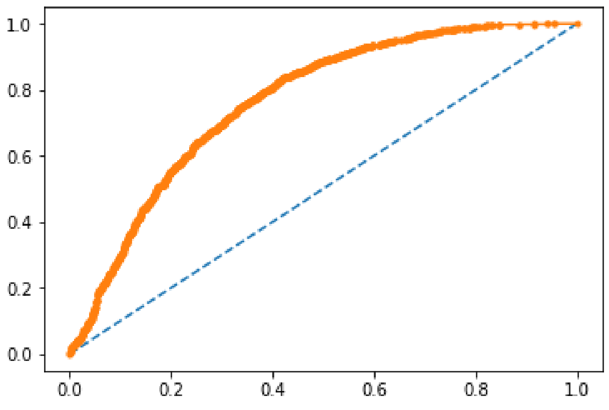

Fig 6. **ROC curve** for XGBoost
Area Under Curve: 0.766

As indicated by the area under curve for the ROC plot, the model performs significantly better than the random guessing indicated by the dashed line. Hence, XGBoost gives the best predictions on the given data set. XGBoost proves to be a powerful classifier and boosting henceforth is a very useful technique for creating powerful learners.

Final Results:

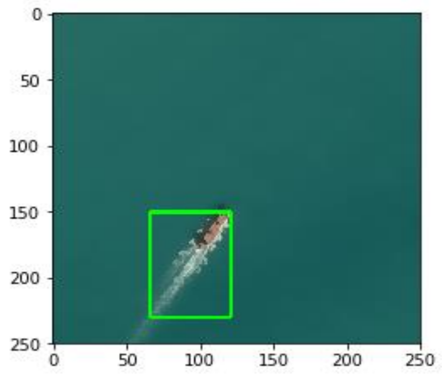

Fig 4. Predicting ship on an image with a bounding rectangle

8. Conclusion

Out of the four models that we trained, XGBoost gave us the best accuracy among others(71%). Our biggest challenge was the size of the data given. Because of the size , we could not train our models in Google Colab and therefore it took a lot of time for our models to get trained and the image features to get extracted in our local systems. Despite that, we believe we got a decent accuracy score for an image dataset this large. We wanted to work on r-CNN and cascade classifier, but due to lack of computational resources, we could not implement it within the timeframe given. If we decide to continue with our project in the future, we would definitely try to implement CNN to train our models and that would give a much higher accuracy than what we have right now.

9. Contribution of Team Members

- Shruti Agrawal
  Developed the code for Pre-processing and cropping; Converted the processed images to a csv file for training; Used the similar approach for creating the test data set; Trained the data for Adaboost classifier and evaluated the model for test data; Made real-time predictions on images

- Mahasweta Sarma
  Separating datasets into positive and negative instances; training Neural Network Model Using MLP Classifier; image cropping; documentation.

- Maleeha Shabeer Koul
  Data pre-processing using Open CV; Experimented test techniques for processing the image; Created csv file for test data set; worked with data manipulation; Trained XGBoost model and evaluated the model for test data; Report for experimental analysis and results; Documentation

- Abhisek Banerjee
  Debugging the code, Experimented with r-CNN model for training and processing, Trained SVM with non linear kernel and performed evaluation metrics, Debugged the code for real time prediction, Prepared README file for the project.

10. References

- Paul Viola and Michael Jones. "RAPID OBJECT DETECTION USING A BOOSTED CASCADE OF SIMPLE DETECTIONS" published in Computer Vision and Pattern Recognition (CVPR 2001).
- Auranuch Lorsakul and Jackrit Suthakorn, "TRAFFIC SIGN RECOGNITION FOR INTELLIGENT VEHICLE/DRIVER ASSISTANCE SYSTEM USING NEURAL NETWORK ON OPENCV" in 4th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI 2007).
- OpenCV documentation: http://docs.opencv.org/
- ROS documentation: http://wiki.ros.org/
- N. Dalal and B. Triggs, "HISTOGRAMS OF ORIENTED GRADIENTS FOR HUMAN DETECTION," in Computer Vision and Pattern Recognition (CVPR 2005).
- Wikipedia
- Scikit Documentation: https://scikit-learn.org/stable/documentation.html
- Kaggle competitions: https://www.kaggle.com/c/airbus-ship-detection