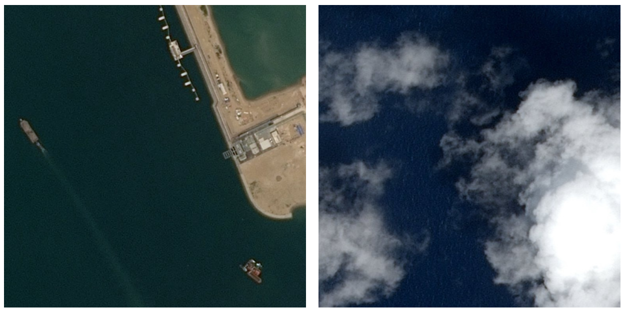
Airbus Ship Detection - Traditional v.s. Convolutional Neural Network Approach

Ying Chen [*1] **Junwen Zheng** [*2] **Zhengqing Zhou** [*3]

Abstract

We compared traditional machine learning methods (naive Bayes, linear discriminant analysis, k-nearest neighbors, random forest, support vector machine) and deep learning methods (convolutional neural networks) in ship detection of satellite images. We found that among all traditional methods we have tried, random forest gave the best performance (93% accuracy). Among deep learning approaches, the simple train from scratch CNN model achieve 94 % accuracy, which outperforms the pre-trained CNN model using transfer learning.



(a) Image labeled by "Ships" (b) Image labeled by "No Ships"

Figure 1. Example images from dataset (Kaggle, 2018)

1. Introduction

The fast growing Shipping traffic increases the chances of infractions at sea, such as environmentally devastating ship accidents, piracy, illegal fishing, drug trafficking, and illegal cargo movement. Comprehensive maritime monitoring services helps support the maritime industry to increase knowledge, anticipate threats, trigger alerts, and improve efficiency at sea. This challenge origins partly from the *Airbus Ship Detection Challenge* on Kaggle. We plan to come up with a solution to efficiently detect ship in satellite images. This classification is challenging because boats are really small in the satellite images. Various scenes including open water, wharf, buildings and clouds appear in the dataset (Kaggle, 2018). Most of the images do not contain any ships and the fact that many images are covered by clouds and fog also increases the difficulty of detection.

In this paper, we compared traditional methods and deep learning methods in solving the classification problem. For traditional methods, we experimented with naive Bayes, linear discriminant analysis, k-nearest neighbors, random forest and support vector machine model. Before training these models, we did image features extraction by finding global feature descriptors of every image, which includes color histogram, Hu Moments, Haralick Texture and Histogram of Oriented Gradients (HOG). We found that feature engineering significantly improved the performance of traditional models. We also noticed that for some model, certain combination of feature descriptors give the best performance, indicating that having more features doesn't necessary improve performance but can also lower performance.

For deep learning method, we used a pre-trained network (DenseNet-169 of ImageNet) architecture as the baseline network (referred as TL-CNN). We then designed a simple CNN model consisting of 4 convolutional layers and 4 max-pooling layers (referred as SIM-CNN) without using transfer learning approach. We observed that the simple train from scratch CNN model worked better than the pre-trained CNN model.

2. Related Works

Recently, machine learning and artificial intelligence have attracted increasing attention and achieved great success in different areas including Computer Vision (Karpathy & Li, 2014), (Sermanet et al., 2013) and Natrual Language Processing (Collobert & Weston, 2008), (Graves et al., 2013). With rapid progress in machine learning, especially in deep learning, many significant breakthroughs have been made in the area of image classification, such as AlexNet (Krizhevsky et al., 2012), ResNet (He et al., 2015) and DenseNet (Huang et al., 2016).

When narrow down to the problem of ship detection, there are no research papers studying this problem since it is a recent *Kaggle Challenge* problem. However, there are several attempts made by some Kaggle users. For example, Kevin

Mader (Mader, 2018) used a transfer learning technique to tackle this ship detection problem. We reproduced his work in our baseline model (see Section 4 for more details).

3. Data Set and Features

3.1. Data Description

We obtained a public dataset provided on the *Airbus Ship Detection Challenge* website (Kaggle, 2018). The dataset contain more than 100k 768×768 satellite images with a total size exceeding 30 Gb, and is actually quite imbalance in the sense that only $\approx 1/4$ of the data images have ships. Along with the images is a CSV file that lists all the images ids and their corresponding pixels coordinates. These coordinates represent segmentation boxes of ships. Not having pixel coordinates means the image doesn't have any ships. However, due to computational limits, we can only handle a dataset of size $\approx$10k. Although one can manually balance the data when training on a subset of full data, since imbalance is the nature of the data, we want our subset to inherit the imbalance property and we may use other method such as threshold tuning to tackle the problem. If our method turns out to make good prediction under this setting, we may have more confidence that the model would work when train on the whole dataset (where imbalance is inevitable). Hence we sample uniformly at random from the full dataset and selected 7k images as training set, 3k images as development set and 5k images as test set.

3.2. Data Preprocessing

First of all, we re-sized the original image to the size of $256 \times 256 \times 3$, then we applied different data preprocessing techniques for traditional machine learning algorithms and deep learning algorithms.

3.2.1. Feature Extraction For Traditional Machine Learning Method

We used hand engineering features extraction methods (Gogul09, 2017) to obtain three different global features for traditional ML algorithms. The images were converted to grayscale for Hu and Ha, and to HSV color space for His before extraction, as shown in Figure 2. (Hu, Ha and His are types of features explained below.)

Color Histogram (His) : Features are applied to quantified the **color** information. A color histogram focuses only on the proportion of the number of different types of colors, regardless of the spatial location of the colors. They show the statistical distribution of colors and the essential tone of an image.

Haralick Textures (Ha) : Features are extracted to described the **texture**. The Haralick Texture, or Image texture, is a quantification of the spatial variation of grey tone values. Haralick et suggested the use of gray level co-occurrence matrices (GLCM). This method is based on the joint probability distributions of pairs of pixels. GLCM shows how often each gray level occurs at a pixel located at a fixed geometric position relative to other pixels, as a function of the gray level.

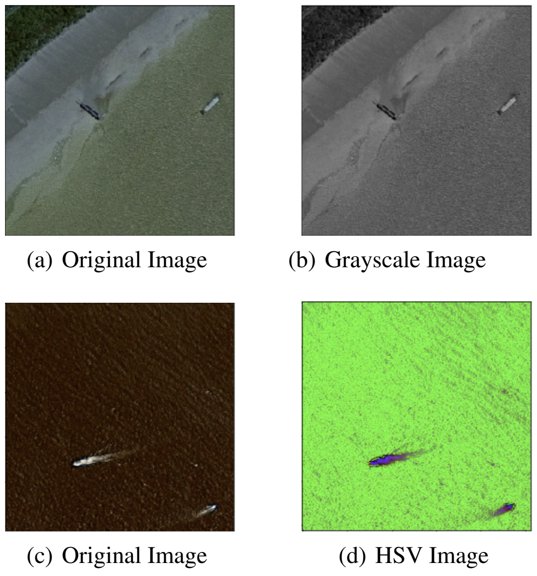


Figure 2. Feature Engineering Examples

Hu Moments (Hu): Features are used to captured the general **shape** information. Hu moment, or image moment is a certain particular weighted average (moment) of the image pixels' intensities. Simple properties of the image which are found via image moments include area (or total intensity), centroid, and information about its orientation.

3.2.2. Data Augmentation for CNN Model

To enhance the robustness of our CNN model, for all the images in the training set, we implemented data augmentation method, such as rotation, shifting, adjusting brightness, shearing intensity, zooming and flipping. Data augmentation can improve the models ability to generalize and correctly label images with some sort of distortion, which can be regarded as adding noise to the data to reduce variance.

4. Methods

To classify whether an image contains ships or not, several standard machine learning algorithms and deep learning algorithms were implemented. We compared different approaches to evaluate how different model performed for this specific task. For all the algorithms, the feature vector is $x = (x_1, \ldots, x_n)$ representing n features (from the flattened original image or from feature engineering).

4.1. Machine Learning Algorithms

4.1.1. Linear Discriminant Analysis (LDA)

The algorithm finds a linear combination of features that characterizes and separates two classes. It assumes that the conditional probability density functions $p(x|y = 0)$ and $p(x|y = 0)$ are both normally distributed with mean and co-variance parameters $(\vec{\mu}_0, \Sigma_0)$ and $(\vec{\mu}_1, \Sigma_1)$. LDA makes simplifying assumption that the co-variances are identical $(\Sigma_0 = \Sigma_1)$ and the variances have full rank. After training on data to estimate mean and co-variance, Bayes' theorem is applied to predict the probabilities.

4.1.2. K-Nearest Neighbors (KNN)

The algorithm classifies an object by a majority vote of its neighbors, with the object being assigned to the class that is most common among its 5 nearest neighbors. The distance metric uses standard Euclidean distance as

$$d(x^{(1)}, x^{(2)}) = \sqrt{\sum_{i=1}^{n}(x_i^{(1)} - x_i^{(2)})^2}$$

4.1.3. Naive Bayes (NB)

Naive Bayes is a conditional probability model. To determine whether there is a ship in the image, given a feature vector $x = (x_1, \ldots, x_n)$, the probability of C_0 "No ships" and C_1 "has ships" is $p(C_k|x_1, \ldots, x_n)$. Using [[Bayes' theorem]], the conditional probability can be decomposed as $p(C_k|x) = \frac{p(C_k)\ p(x|C_k)}{p(x)}$. With the "naive" conditional independent assumption, the joint model can be expressed as $p(C_k|x_1, \ldots, x_n) = p(C_k)\prod_{i=1}^{n} p(x_i|C_k)$

4.1.4. Random Forest (RF)

The algorithm is an ensemble learning method. Bootstrap samples are selected from the training data, and then the model learns classification trees using only some subset of the features at random instead of examining all possible feature-splits. After training, prediction is made by taking the majority vote of the learned classification trees. The depth of tree is limited by 5 and the number of trees is 10.

4.1.5. Support Vector Machine (SVM)

The algorithm finds the maximum margin between different classes by determining the weights and bias of the separating hyperplane. The soft-margin SVM classifier minimizes the loss as

$$\mathcal{L} = \left[\frac{1}{n}\sum_{i=1}^{n}\max\left(0, 1 - y_i(w \cdot x_i - b)\right)\right] + \lambda\|w\|^2$$

The fit time complexity is more than quadratic with the number of samples.

4.2. Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN), which is prevailing in the area of computer vision, is proved to be extremely powerful in learning effective feature representations from a large number of data. It is capable of extracting the underlying structure features of the data, which produce better representation than hand-crafted features since the learned features adapt better to the tasks at hand. In our project, we experimented two different CNN models.

4.2.1. Transfer Learning CNN (TL-CNN)

The motivation of using a transfer learning technique is because CNNs are very good feature extractors, this means that you can extract useful attributes from an already trained CNN with its trained weights. Hence a CNN with pre-trained network can provide a reasonable baseline result.

As mentioned in Section 2, we reproduced the so-called "Transfer Learning CNN" (TL-CNN) as our baseline model. We transferred the "Dense169" (Huang et al., 2016) (which was pre-trained on the ImageNet (Deng et al., 2009)) to our task. More precisely, for each $256 \times 256 \times 3$ image, fed it into the "Dense169" network with pre-trained weights, and then applied a Batch Normalization layer, a Dropout layer, a Max Pooling layer and two Fully Connected layers, we finally sent the output to a Sigmoid function to produce a prediction probability of whether the input image contains ships. See Figure 3 for more details.

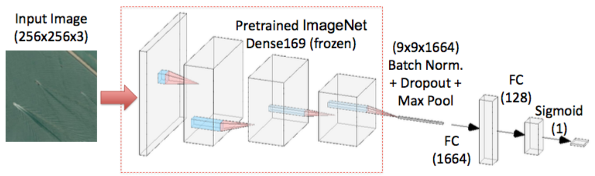


Figure 3. Transferred Learning CNN (TL-CNN) Model Framework

4.2.2. Simple CNN (SIM-CNN)

Instead of using a pre-trained CNN model, we decided to construct a simple CNN model (named "SIM-CNN") with a few layers and trained it from scratch. One might hope that this could improve the performance of TL-CNN since its CNN weights were trained specifically by our Dataset (Kaggle, 2018), hence it was more specific to our task.

The Figure 4 in below shows the structure of SIM-CNN. The model took images as input and predicted probabilities as output. It started with 4 Convolutional layers and 4 Max

Pooling layers, with each Max Pooling layers following a Convolutional layer, and ended in 2 Fully Connected Layers as well as a sigmoid function.

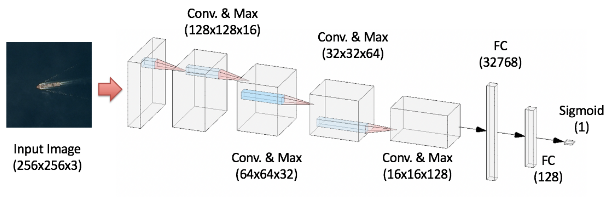


Figure 4. Simple CNN (SIM-CNN) Model Framework

4.3. Training

For the traditional machine learning algorithms discussed in Section , we applied `sklearn` package in `python` to realize each algorithms.

For the deep learning approach, the two CNN models were trained in Microsoft Azure Cloud with GPUs. Cross-entropy loss was used. We trained the model for 30 epochs using mini-batch (size 64) with batch normalization. We applied Adam optimizer as well as decaying learning rate to facilitate model training. The weights of model would not be updated, if the loss of development set did not improve after training an epoch.

5. Results & Discussion

5.1. Result Analysis for Machine Learning Algorithms

5.1.1. Effects of Feature Engineering

Alg	His	Ha	Hu	his + Ha	Ha + Hu	Hu + His	w/all	w/o
LDA	0.83	0.86	0.83	0.87	0.86	0.83	**0.87**	0.74
KNN	0.82	0.79	0.82	0.79	0.79	**0.82**	0.79	0.78
NB	0.49	**0.84**	0.49	0.75	0.84	0.49	0.75	0.42
RF	0.92	0.90	0.92	0.93	0.89	0.92	**0.93**	0.85
SVM	0.85	0.84	0.85	0.83	0.84	**0.85**	0.83	0.65

Table 1. Test Accuracy of Machine Learning Algorithms with different feature extraction techniques, “His” stands for Color Histogram, “Ha”means Haralick Textures and “Hu” ia abbreviation of Hu Moments

Comparing performance with and without feature engineering, we found that feature engineering significantly improved performance in general. Instead of directly training on image pixels information, the information extracted from feature engineering amplified the signal of whether an image containing ships, especially for NB and SVM approaches. However, since 0.84 of the images in test set are labeled as “No ships”, among those traditional machine learning method only LDA and RF outperformed the accuracy of blindly guesting “No ships”. In addition, We learned that some algorithms give much better performance when working with only certain combination of features. It is suggested that Haralick Textures information of image is of great importance for NB method (improving from 0.42 to 0.75).

5.1.2. Test Accuracy

We applied 10-fold Cross Validation to each machine learning algorithms and create the corresponding box plot in Figure 5. As we can see, RF achieve the highest mean test accuracy and the smallest variance among all the machine learning algorithms. Though 10-fold Cross Validation only reduced the training data by 10%, the other methods except RF fluctuated a lot about 5%. They were not as stable as RF. For LDA, the reason why it outperformed blindly guesting might just due to good luck of reaching the upper bound.

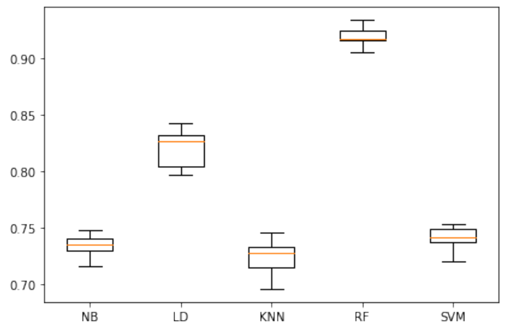


Figure 5. Box Plot for Test Accuracy of Machine Learning Algorithms

5.2. Results Analysis for CNN models

5.2.1. Train Accuracy

For each CNN Model, we plotted the training loss curve as well as the training accuracy curve for 30 epoch (see Figure 6). We observed that both training processes converged within 30 epoch. Additionally, the SIM-CNN model achieve both higher training accuracy and lower training loss compared to TL-CNN.

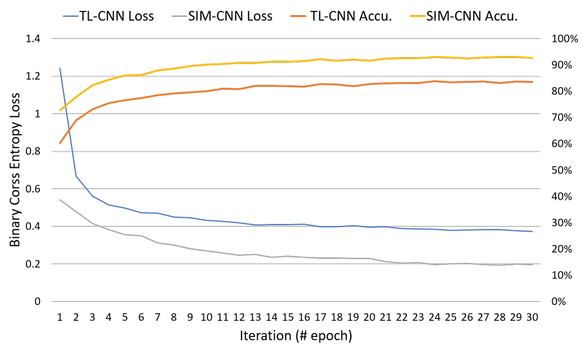


Figure 6. Training Loss and Accuracy of CNN Models

5.2.2. Threshold Scanning

Consider our dataset being imbalanced, instead of using a threshold of 0.5, we wanted to find one that would take imbalance into consideration as suggested by (Collell et al., 2016). We scanned thresholds from 0.1 to 0.9 and plot the accuracy of the validation data (see Figure 7). We used 0.1 for TL-CNN and 0.6 for SIM-CNN to make prediction on the test data. With these thresholds, TL-CNN gave 0.90 test accuracy while SIM-CNN gave 0.94 test accuracy. The result is reasonable because the images of our dataset were different from those in the pre-trained ImageNet dataset, and TL-CNN was not specialized for this project. Besides, there were much more trainable parameters in SIM-CNN $(4, 292, 001)$ than in TL-CNN $(216, 577)$. Therefore, SIM-CNN could better capture the information from the image than TL-CNN.

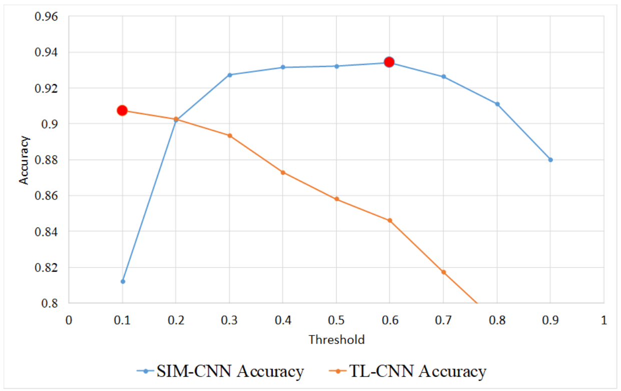
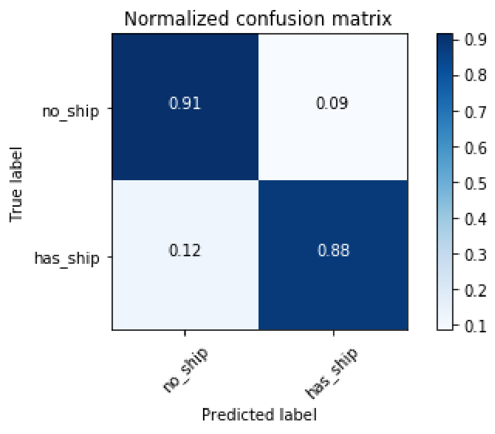
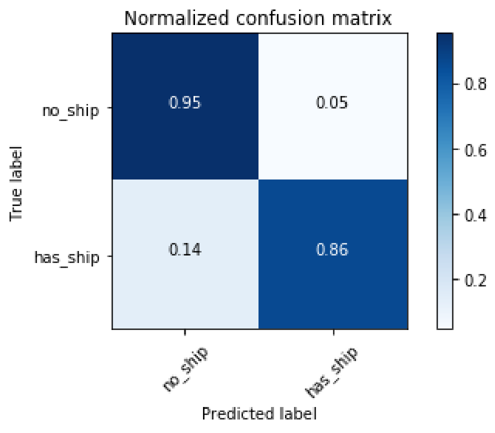


Figure 7. Threshold Scanning of TL-CNN and SIM-CNN



(a) TL-CNN



(b) SIM-CNN

Figure 8. Normalized Confusion Matrix

5.2.3. Confusion Matrix

In Figure 8 there are normalized confusion matrices of TL-CNN and SIM-CNN methods. From the confusion matrix, TL-CNN has a precision of 0.65 with recall equals to 0.88 and specificity equals to 0.92. To compare, SIM-CNN has a precision of 0.76 with recall equals to 0.86 and specificity equals to 0.95 . On one hand, we found that SIM-CNN did better job at classifying “no ship” images. However, on the other hand TL-CNN outperforms SIM-CNN when classifying “has ship” images. Notice that by Section 5.2.2 that SIM-CNN has higher test accuracy, we suspected that this might be caused by the imbalance nature of our dataset.

6. Conclusoin & Future Works

Among all traditional methods, Random Forest gave the best result (0.93 accuracy) with feature engineering. As for the CNN model, our train from scratch SIM-CNN model outperforms the baseline TL-CNN model based on pre-trained DenseNet 169 network.

In the future, for traditional Machine Learning algorithms, we plan to improve feature engineering by extracting global features along with local features such as SIFT, SURF or DENSE, which could be used along with Bag of Visual Words (BOVW) technique. For Deep Learning algorithms, to achieve better performance, we will try implementing different networks (e.g., deeper network) to train the classifier. Last but not least, it is more challenging but also more interesting to try applying segmentation technique to identify the locations of all ships in a image.

7. Team Contribution & Project Code

All three members of this team work together and contribute equally to this project in data prepossessing, algorithm designing, model designing, model training and report writing.

Please follow project code or https://github.com/cs229ShipDetection/CS229project-Airbus-Ship-Detection for the project code.

References

Collell, Guillem, Prelec, Drazen, and Patil, Kaustubh R. Reviving threshold-moving: a simple plug-in bagging ensemble for binary and multiclass imbalanced data. *CoRR*, abs/1606.08698, 2016. URL http://arxiv.org/abs/1606.08698.

Collobert, Ronan and Weston, Jason. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pp. 160–167. ACM, 2008.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

Gogul09. Image classification using python and machine learning. https://github.com/Gogul09/image-classification-python, 2017.

Graves, Alex, Mohamed, Abdel-rahman, and Hinton, Geoffrey. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pp. 6645–6649. IEEE, 2013.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL http://arxiv.org/abs/1512.03385.

Huang, Gao, Liu, Zhuang, and Weinberger, Kilian Q. Densely connected convolutional networks. *CoRR*, abs/1608.06993, 2016. URL http://arxiv.org/abs/1608.06993.

Kaggle. Dataset For Airbus Ship Dectection Challenge. https://www.kaggle.com/c/airbus-ship-detection/data, 2018.

Karpathy, Andrej and Li, Fei-Fei. Deep visual-semantic alignments for generating image descriptions. *CoRR*, abs/1412.2306, 2014. URL http://arxiv.org/abs/1412.2306.

Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pp. 1097–1105, USA, 2012. Curran Associates Inc. URL http://dl.acm.org/citation.cfm?id=2999134.2999257.

Mader, Kevin. Transfer Learning For Boat or No−Boat. https://www.kaggle.com/kmader/transfer-learning-for-boat-or-no-boat, 2018.

Sermanet, Pierre, Kavukcuoglu, Koray, Chintala, Soumith, and LeCun, Yann. Pedestrian detection with unsupervised multi-stage feature learning. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pp. 3626–3633. IEEE, 2013.