Ship Detection Using Faster RCNN

Charles Young
08-Gary the Waterhog
cdy2@illinois.edu

Josh Janda
08-Gary the Waterhog
joshlj2@illinois.edu

Skyler Shi
08-Gary the Waterhog
jingtao2@illinois.edu

Ari Ben-Zeev
08-Gary the Waterhog
benzeev2@illinois.edu

I. Introduction

There is an increasing amount of sea traffic nowadays and the company Airbus wants to be able to efficiently monitor sea traffic movement. This is to provide assurance to logistics companies that their ships are not being pirated or going into illegal fishing areas. To do so, we utilize image recognition abilities on satellite images. Specifically, we will use a Keras implementation of Faster R-CNN [1]. The input will be top-down images of the ocean that may or may not contain ships. We will build bounding boxes around ships present in the training images, then return these as a set of encoded pixels to verify with the test data set.

II. Related Work

There are numerous studies relating to ship detection, and this topic has gained interest over the past 40 years at an exponential rate [1]. These studies usually involve taking images from Synthetic Aperture Radar (SAR) in order to monitor ocean activity. The images we will use come from SAR. A survey of studies relating to ship detection found that primary motivations include safety, marine traffic, monitoring piracy and smuggling, defensive systems, pollution, managing fisheries, spatial planning, and border control [1].

In order to accurately detect ships, numerous studies use region-based convolutional neural networks (R-CNN). This method involves generating region proposals for images, extracting a fixed-length feature vector from each proposal using a CNN, and then classifying each region with category-specific linear SVMs [2]. While this is a valuable method, it can be improved upon. R-CNNs are computationally expensive, which would provide difficulty for our group given our limited time and resources. A different method, called Faster R-CNN, uses very deep neural networks to achieve near real-time rates by ignoring the time spent on region proposals [3]. We will be using this Faster R-CNN method, since it is faster than the regular R-CNN method.

Some studies have tried to further improve upon the Faster R-CNN method. One such study rotated bounding boxes to create more specific detection, which allowed for more direct localization of ships in a specific area compared to Faster R-CNN (4). While this method is intriguing, we decided it would be better to go for a simpler approach. Another study we found used a squeeze and excitation rank (SER) faster R-CNN to improve performance and speed (5), but again, we decided that this would be too complex for our project.

[1]https://arxiv.org/pdf/1512.03385.pdf

III. Data

The data exists as 768x768x3 rgb images from the Airbus Kaggle competition [2]. These consist of top-down images of the ocean, some of which contain ships. These images also contain encoded pixels that represent where ships are in the images. We only train on images that have ships in them, so that brings the training set from 192,556 to 42,556 images.

We convert the encoded pixels to masks. This takes the encoded pixels and transforms into a mask, where ”ship being present” is 1 and ”no ship” is 0. This code was borrowed from a Kaggle notebook [3]. Once the mask exists, we converted them to bounding boxes using scikit.image.label and scikit.image.region_props. scikit.image.label labels where the mask is in the image, then scikit.image.region_props measures where the mask starts and ends, returning a tuple of coordinates.

Then we resize the images to fit 256x256x3 to save space, and then create an observation for each boat. There could be multiple rows for each image since each image could have more than one ship. We also convert the image ID to a file path. Finally, we flip the x and y coordinates, since the original data has the axes flipped. This returns a dataframe that has the file path, coordinates of the mask, and class name (all of which are ship.)

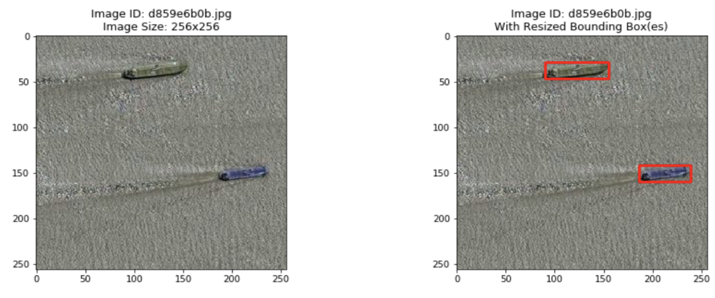

Fig. 1. An example of an image with two ships

IV. Preliminary Technical Details and Results

We implement a faster R-CNN with a base resnet50 architecture.

[2]https://www.kaggle.com/c/airbus-ship-detection/data
[3]https://www.kaggle.com/voglinio/from-masks-to-bounding-boxes

A. Faster R-CNN compared to other R-CNNs

Faster R-CNN utilizes a RPN (Region Proposal Network) to propose multiple regions of interest after the image is passed through the base resnet CNN. These proposed regions are then passed through a fully-connected network to create classification results (with softmax activation as the last layer). This makes it significantly faster to train than a normal R-CNN which passes multiple regions of interest through a CNN and fully-connected network and then their individual SVMs to classify results. This faster R-CNN passes all images through the same CNN, and all proposal regions through the same FC layers and softmax last layer. It is also faster than a fast R-CNN because a fast R-CNN uses a slower selective search algorithm to select regions of interest but the faster R-CNN uses a fast RPN to select regions of interest.

B. Resnet50

Resnet is the CNN architecture that is being utilized within the Faster R-CNN implemented. The basic idea of Resnet is that instead of directly learning a mapping of the input data to output data, a Resnet block learns the residual (difference between output and input) and sums it with the input to produce the output. This design helps us build deep networks without decay of training accuracy. Multiples of these residual blocks are combined to form a sequential architecture. We use 50 layers for our model. The Resnet is used for feature extraction of the input image. It comes before the RPN.

C. More Faster R-CNN Details

The RPN network takes anchors as inputs, which are predefined bounding boxes. These predefined bounding boxes are centered at each point of the feature map image and are of varying sizes to account for a variety of final bounding box sizes. The idea is to adjust these anchors to form the final bounding box predictions. The size of these anchors and relation of width to height are hyperparameters we can tweak. In our case, we use the default anchors sizes of and ratios of heights and widths. The RPN takes all these anchors and outputs a set of good proposal regions. It outputs the probability of the anchor being an object and the probability of it being background. It also outputs 4 numbers Δxcenter, Δycenter, Δwidth, Δheight to adjust the anchor to a final proposed region. Combining the anchors of high probability being objects with their 4 adjustment numbers, we get decent region proposals. Finally, with these decent region proposals, the faster R-CNN passes them through some FC layers and outputs a classification label and the 4 bounding box adjustment numbers again to refine the bounding-box area. To backpropogate this network, we calculate the loss function of the classification task with binary cross entropy and the loss function of the regression task with L1 loss on only the bounding boxes that denote objects (background bounding boxes excluded from regression loss).

D. Hyperparameters

Hyperparameters not mentioned below follow the default implementation values in the faster R-CNN implementation we found on github.

- Epochs
  - 1000
- Activation Functions
  - Default used in resnet
- Number of regions of interest:
  - Default from github implementation
- Batch size
  - Default from github implementation
- Loss Function
  - Classification loss uses binary cross-entropy
- Regression loss
  - Smooth L1 loss
- Number of layers for Resnet
  - 50 layers
- Number of layers for FC network that takes in proposal regions
  - Default from github implementation
- Convolution filter size, stride length
  - Default from github implementation
- Anchor definitions
  - Default from github implementation

V. Timeline

Given the image obtained from the data section, the bounding boxes seem to be accurate. We have connected to HAL, so all that is left to do is let our network train on all of the training data.

VI. Group Member Work

- Charlie Young (16.66%): Formatting IEEE document, introduction section, data section
- Josh Janda (50%): Initializing Git repository, writing data cleaning/training code, interfacing the keras faster rcnn code with our training data
- Skyler Shi (16.66%): Preliminary technical details and results section
- Ari Ben-Zeev (16.66%): Related work section

References

[1] Kanjir, Urška et al. "Vessel detection and classification from spaceborne optical images: A literature survey." Remote sensing of environment vol. 207 (2018): 1-26. doi:10.1016/j.rse.2017.12.033

[2] Girshick, Ross, et al. "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation." 2013, doi:https://arxiv.org/abs/1311.2524.

[3] Ren, Shaoqing, et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks." IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 6, Jan. 2017, pp. 1137–1149., doi:10.1109/tpami.2016.2577031.

[4] Li, Mingjie, et al. "Rotated Region Based Fully Convolutional Network for Ship Detection." IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium, 2018, doi:10.1109/igarss.2018.8519094.

[5] Lin, Zhao, et al. “Squeeze and Excitation Rank Faster R-CNN for Ship Detection in SAR Images.” IEEE Geoscience and Remote Sensing Letters, vol. 16, no. 5, 2019, pp. 751–755., doi:10.1109/lgrs.2018.2882551.