# xView3 competition third place solution

## 1. Introduction

The competition focused on the ability to detect targets at sea and the ability to classify vessels (including vessel size and whether they were fishing or not). Therefore, we decompose the task into two parts, detection and classification, respectively. In the detection model, we use HRNet-based[1] heat map regression model to detect maritime targets, including fixed facilities and vessels. In the classification model, we crop 224*224 size image blocks centered on the detected targets and use ResNet101[2] model for prediction, including three dimensions: target category, whether fishing, and vessel length.

The source code can be found in xView3_3rd_place_solution.

## 2. Data preprocessing

A total of 554 scenes of training data and 50 scenes of validation data were provided in this competition. After a simple analysis of the data, we found that the training data had a large amount of noise, mainly some missed annotations. Therefore, the training data is not suitable for training detection models, but can be used to train classification models.

### 1) Detection training data preprocessing

When training the detection model, we only used the provided validation data. Specifically, for the 50 scenes of validation data, we first transformed the raw SAR data to the 8-bit form, range 0-255. Then for each scene of data, we used a block size of 1024 and a step size of 896 to generate small cropped images. For each small cropped image, if there is a target in it, we use a Gaussian kernel to generate a heat map, which is divided into maritime non-vessel targets and vessel targets. Cropped images are divided into two categories based on the presence or absence of a target in the image. In total, there are 30,560 cropped images, of which 3986 images have targets on them.

See data_preprocess/detection.py for reference.

The processed detection training data sample is shown in the following figure:

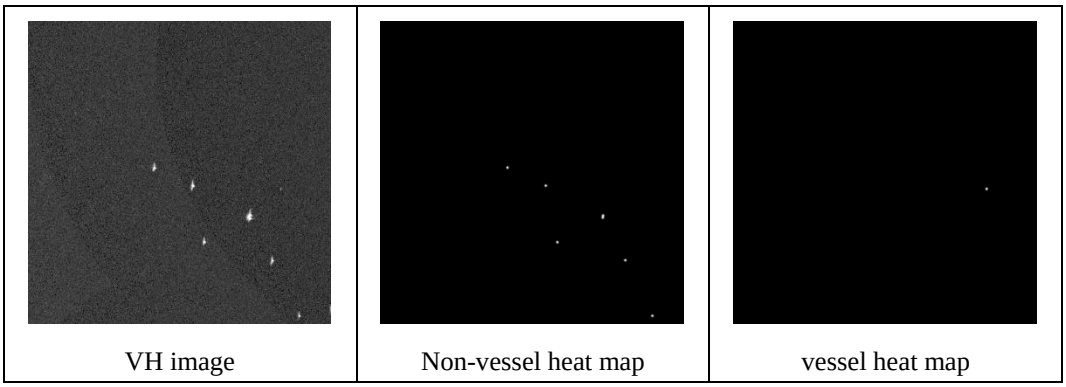


Figure 1: detection training data sample.

**2) Classification training data preprocessing**

When training the classification model, we used both the training and validation data. Specifically, for each object in the csv file whose confidence level is not LOW, we crop an image block of 224*224 size centered on it and use it as a sample for classification. The classification labels consist of three parts, whether it is a vessel, whether it is fishing, and the length of the vessel. A total of 53067 images can be cropped from the training data and 15790 images from the validation data.

See data_preprocess/classification.py for reference.

The processed classification training data sample is shown in the following figure:

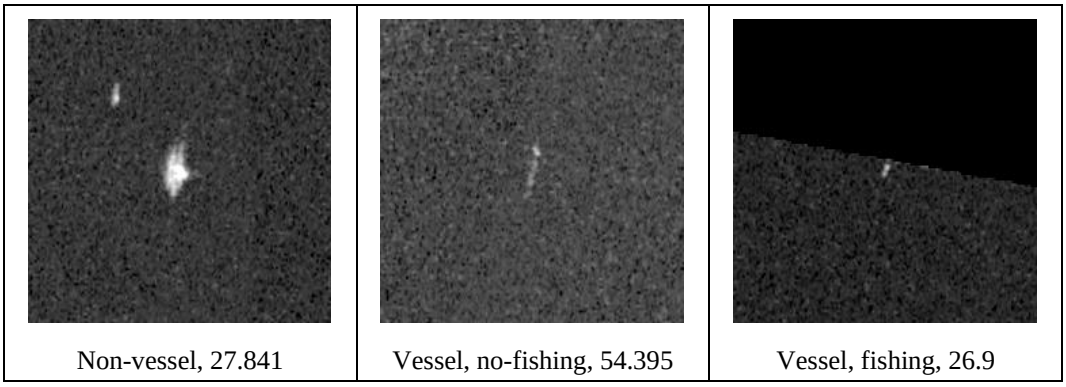


Figure 2: classification training data sample.

## 3. Methodology

**Detection model**

For the detection model, we used the HRNet-w48 model with two channels as output, which are responsible for predicting the heat maps of vessels and non-vessels, respectively. We divided the 50-scene validation data into 5 groups equally, and took one group at a time as the validation data, and trained a total of 5 detection models.

For each training round, we first trained the model using small chunks of images containing the maritime object to enable the model to converge quickly, and then used the full data to fine-tune the model to make it more generalizable. The loss function for training is the MSE loss function. The training parameters are shown in the following table:

| Optimizer | Learning Rate | Batch Size | Epoch Num | Crop Size |
|---|---|---|---|---|
| Adam | 0.0001 | 8 | 20 | 512*512 |

See src/train_detect_split.py for reference.

For testing, we use sliding window strategy to get the heat map prediction of the whole scene image. The window size and stride are set 2560 and 2240. We average the heat maps predicted by the five models to get the average heat map for each window. We then do non-maximum suppression on the heat maps with max-pooling operation to get the maritime object detection results. Since the provided images have coordinate information, we can store the geographic locations of non-vessel targets from the validation data. When testing the test images, the accuracy of the detection model is improved by determining whether targets are still present at these locations based on the prediction results of the detection model.

See src/inference/inference.py for reference.

**Classification model**

For the classification model, we used a ResNet101 model with 3 channels as output, which are responsible for predicting whether it is a vessel, whether it is fishing, and the length of the vessel, respectively. We divided the 554-scene training data into 10 groups equally and the 50-scene validation data into 5

groups equally, and took one group of the training data and one group of the validation data as the validation data each time, and trained a total of 10 classification models.

The loss function for the first two classification tasks was the Cross Entropy loss function, and the loss function for the vessel length regression was the normalized L1 loss function. The training parameters are shown in the following table:

| Optimizer | Learning Rate | Batch Size | Epoch Num | Crop Size |
|---|---|---|---|---|
| Adam | 0.01 | 128 | 100 | 224*224 |

See src/train_class_split.py for reference.

For testing, we crop small blocks of 224×224 size in a large image centered on the maritime object as input, and average 10 predictions of the classification model as the final classification result.

See src/inference/inference.py for details.

## 4. Conclusion and Acknowledgments

In this competition, we decomposed the task into two subtasks and used two classical methods to solve the two subtasks respectively, which achieved good results and met the requirements in terms of processing speed.

However, we did not get a good use of the large amount of training data provided. If the training data is used for reasonable semi-supervised training, the performance of the model may be better. In addition, we did not do a targeted treatment for the close-to-shore object in the evaluation index, and it might have been effective if a separate model was trained for the close-to-shore object.

Finally, we would like to thank the xView team for providing us with the opportunity to apply the machine learning model on real data.

## Reference

[1]. Jingdong Wang, et al. "Deep high-resolution representation learning for visual recognition." IEEE transactions on pattern analysis and machine intelligence (2020).

[2]. Kaiming He, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.