Using Tensorflow to Detect Objects in an Image

Reginald Kelley, Thomas Johnson, Edsel Norwood, Taeyonn Reynolds
Jerome Mitchell :: Center of Excellence in Remote Sensing Education and Research

Abstract

The abundance of various types of satellite imagery has created a challenge with processing and analyzing data into useful information. In this project, we explored the development, implementation, and evaluation of a machine learning algorithm, specifically a neural network, to automate the detection of ships to track traffic in a desired port or region. We also used a graphical approach to computation using TensorFlow, which offers easy massive parallelization and deployment to the cloud. The final result is an algorithm, which is capable of receiving images from various sources of imagery at various resolutions and be able to identify the appropriate objects within the image.

ShipsNet is a labeled training dataset consisting of images extracted from Planet satellite imagery. It contains hundreds of 80x80 pixel RGB image chips labeled with either a "ship" or "no-ship" classification. Machine learning models can be trained against this data to classify any given input chip into either one of these classes.

With an accurately trained model, this classification process can be extended to a full ship image scene by using the sliding window technique. A 80x80 pixel window is moved across each pixel position in the image, extracted, and classified by the model. Neighboring window positions that are classified as "ship" are then clustered into a single detection. These detections are highlighted with a bounding box in a copy of the original ship scene (*shown in figure 1*).

What is Tensorflow?

TensorFlow is an open source software library for high performance numerical computation. Its flexible architecture allows easy deployment of computation across a variety of platforms. It comes with strong support for machine learning and deep learning, and the flexible numerical computation core is used across many other scientific domains.

Required Software

Language

- Python Version 3.5+

Libraries

- Numpy Version 1.13.0+mkl
- Pillow Version 4.1.1
- Scipy Version 0.19.0
- Tensorflow Version 1.2.0
- Tflearn Version 0.3.1

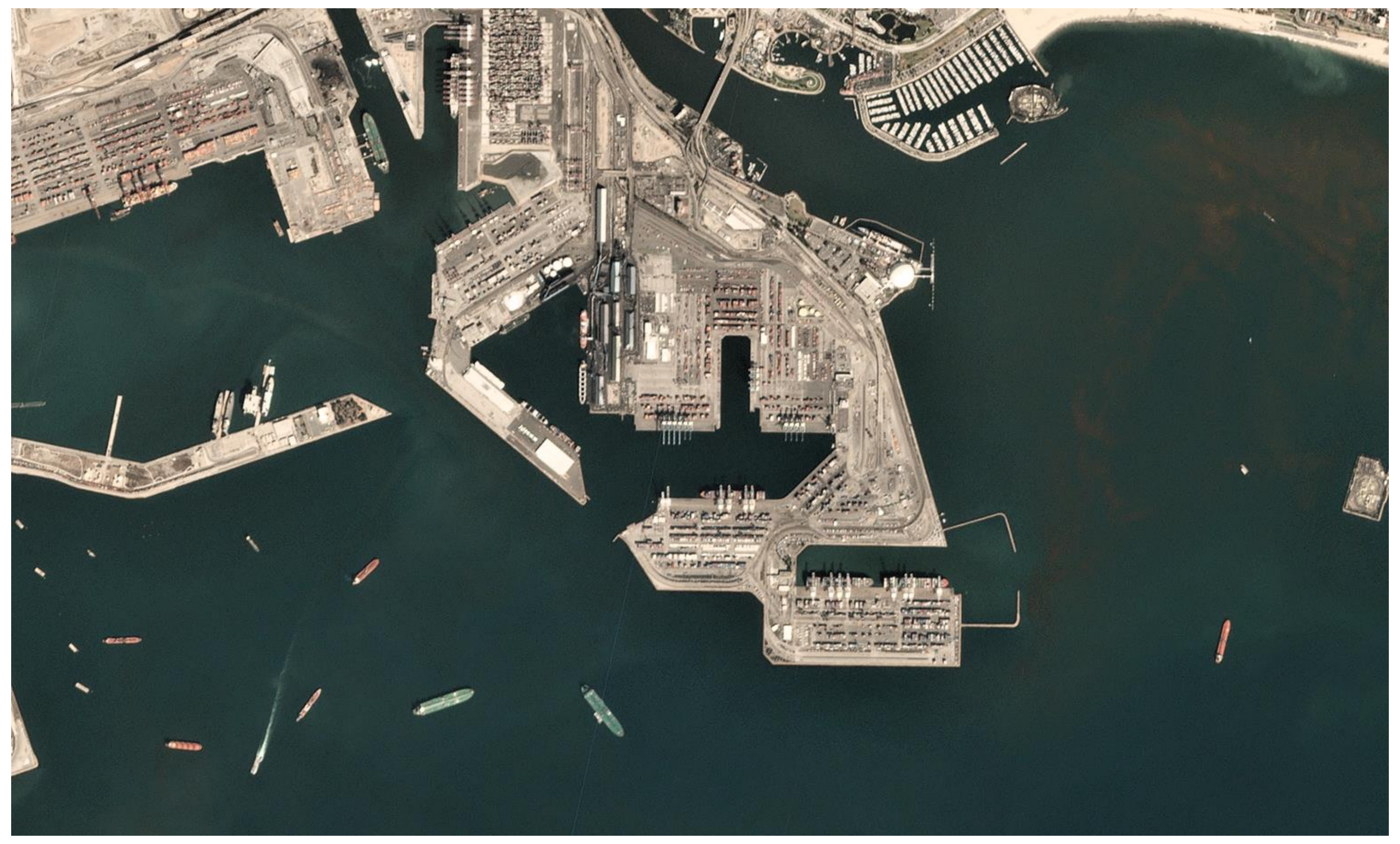
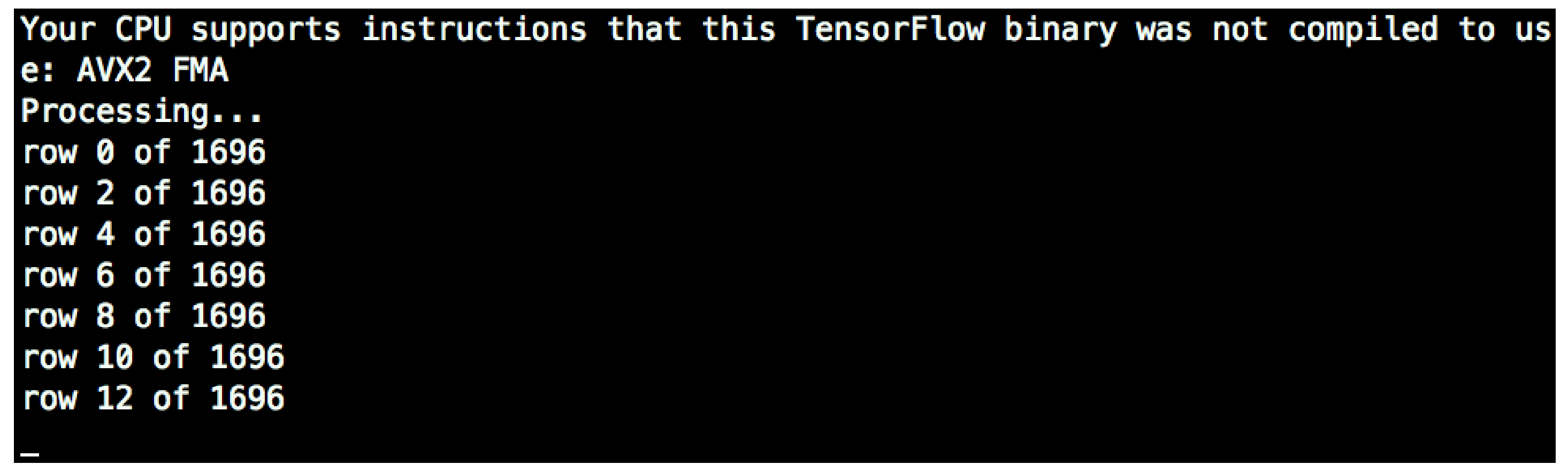


Figure 1

Training the Program

Before being able to efficiently identify objects in the provided images the program needs to be trained using a series of sample images on what it should be looking for. The command responsible for beginning this training is python3 train.py "shipsnet.json" "models/model.tfl". Training is done in separate sessions or “epochs” (*shown in figure 3*).



```
Your CPU supports instructions that this TensorFlow binary was not compiled to us
e: AVX2 FMA
Processing...
row 0 of 1696
row 2 of 1696
row 4 of 1696
row 6 of 1696
row 8 of 1696
row 10 of 1696
row 12 of 1696
_
```

Figure 2

Results

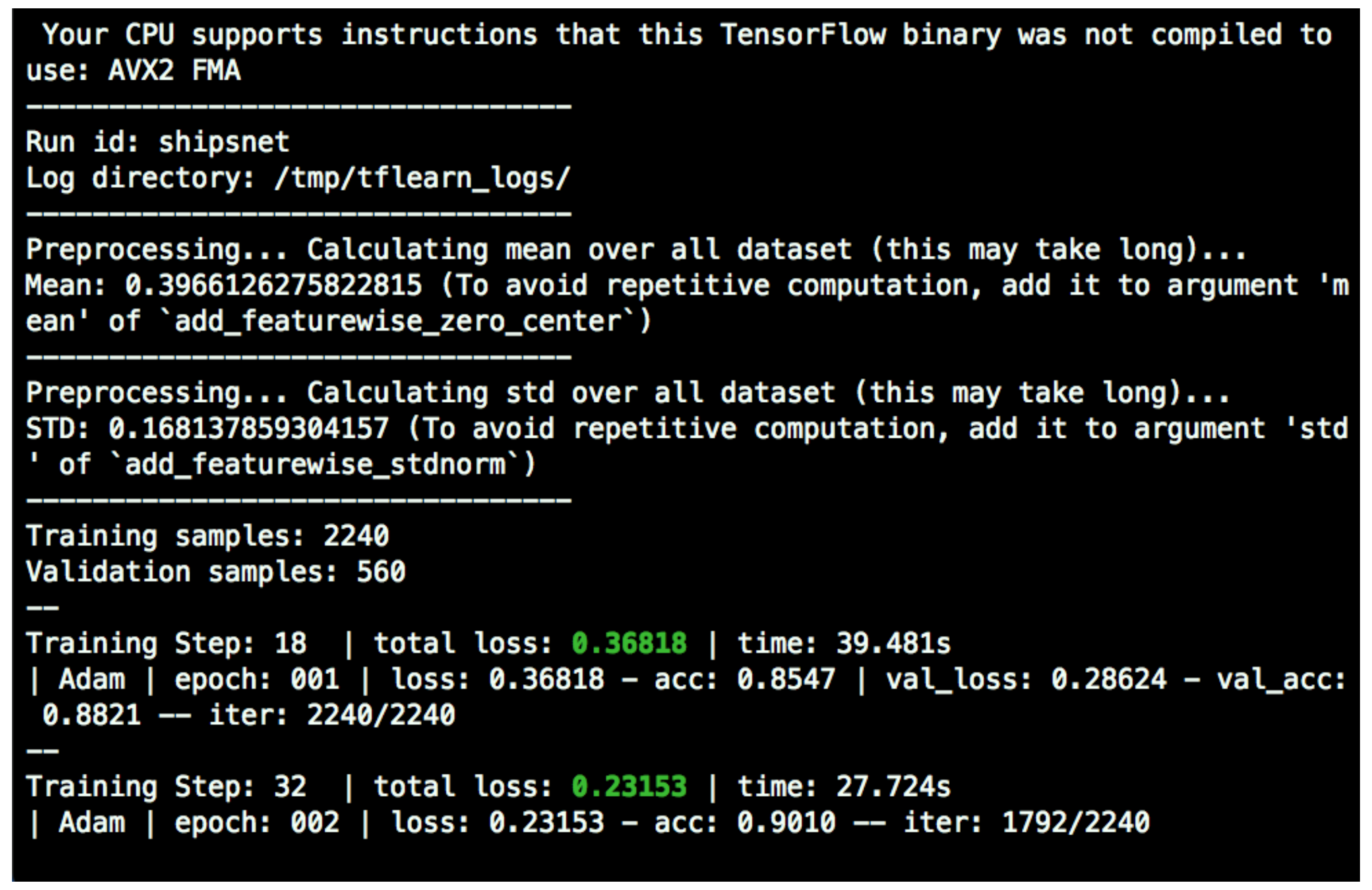
The final result was an algorithm, which is capable of receiving images from various sources of imagery at various resolutions and able to accurately detect most of the ships within the image (*shown in figure 4*)

Conclusion

In conclusion the developed algorithm was not 100% adequate, in the fact that the model had three mistakes. It missed one ship and classified two non-ship objects as ships. Although the algorithm had errors the Center of Excellence in Remote Sensing Education and Research (CERSER) program at Elizabeth City State University can benefit from the use of the algorithm. The algorithm can be modified and re-trained to detect any object the user wishes.

In the future one can improve the efficiency of the algorithm. Lessening the run time and making it easier for users to see results in a timely manner. Also modifying the code to be able to better indicate if an image displays a ship or not.

Altering the neural network model in its various details, such as the layers to see if there are significant shifts in the accuracy of the models developed in the training.



```
 Your CPU supports instructions that this TensorFlow binary was not compiled to
use: AVX2 FMA
---------------------------------
Run id: shipsnet
Log directory: /tmp/tflearn_logs/
---------------------------------
Preprocessing... Calculating mean over all dataset (this may take long)...
Mean: 0.3966126275822815 (To avoid repetitive computation, add it to argument 'm
ean' of `add_featurewise_zero_center`)
---------------------------------
Preprocessing... Calculating std over all dataset (this may take long)...
STD: 0.168137859304157 (To avoid repetitive computation, add it to argument 'std
' of `add_featurewise_stdnorm`)
---------------------------------
Training samples: 2240
Validation samples: 560
--
Training Step: 18  | total loss: 0.36818 | time: 39.481s
| Adam | epoch: 001 | loss: 0.36818 - acc: 0.8547 | val_loss: 0.28624 - val_acc:
 0.8821 -- iter: 2240/2240
--
Training Step: 32  | total loss: 0.23153 | time: 27.724s
| Adam | epoch: 002 | loss: 0.23153 - acc: 0.9010 -- iter: 1792/2240
```

Figure 3

Processing the Image

Once the training is done, one can begin to detect objects in an image using the command *python3 detector.py "models/model.tfl" "images/scene_1.png"*. Processing a image of high resolution will be a time consuming task as this requires the program to traverse through each individual row of the picture (*shown in figure 2*).

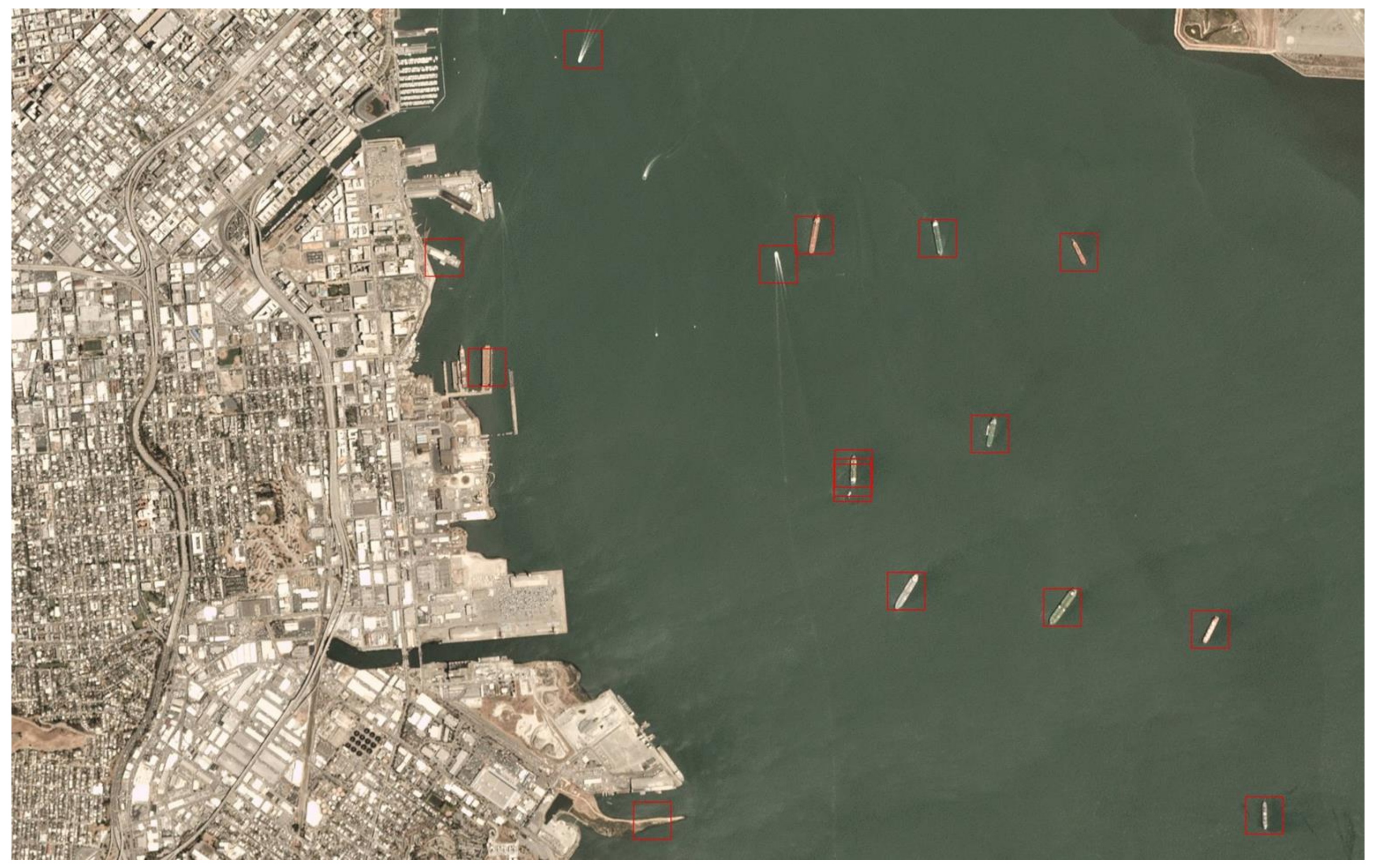


Figure 4

Acknowledgements

Jerome Mitchell for his guidance, contributions, and help with completing this research.

Dr. Linda Hayden for the research opportunity that was made possible through the CERSER program.