MUSA 650 Final Report - Singapore Ship Detection

Eugene Chong, Madhura Gurav, and John Michael LaSalle

Introduction

In this project, we create a model for detecting large ships in Sentinel-2 imagery based on a training set of labeled PlanetScope images. Commercial ships use automatic identification systems (AIS) to broadcast their locations, but detecting ships in satellite imagery is useful for when ships disable their transponders to hide their locations. Identifying ships is also useful for tracking the impacts of disruptions to maritime shipping.

There are a growing number of satellite imagery providers, including private companies like Planet or DigitalGlobe, and public programs like Sentinel-2 and LandSat, which collect imagery at different spatial and temporal resolutions. This has led to greater availability of data for researchers, but labeled training sets for machine learning do not exist for all of these providers, and creating them can be a time- and labor-intensive process. For our project, we wanted to investigate how well machine learning models, including support vector machines and convolutional neural networks, trained on imagery from one of these providers could generalize to imagery from another provider with a different spatial resolution. Through this work, we aimed to better understand how straightforward it would be for researchers to mix and match satellite imagery from various sources in machine learning models.

Data

We started with Shipsnet, a publicly available dataset of labeled ship images captured with Planet PlanetScope satellites created by Bob Hammell. We accessed Shipsnet using the Kaggle API. Shipsnet has 4,000 patches, comprising 1,000 ship patches and 3,000 non-ship or partial ship patches.

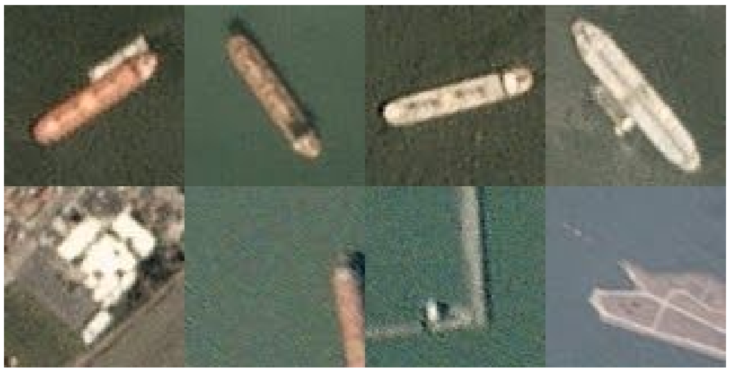

Figure 1. Shipsnet patches, ships above, not-ships below.

Shipsnet consists of 3-channel (RGB) images with a ground-sampling-distance of approximately 3 meters, meaning each 80x80 patch covers 57,600 square meters. The data are stored in a JSON file consisting of a flattened vector of 19,200 pixel values for each image and each image's label. All the images come from the Port of Long Beach or the San Francisco Bay.

The data for our test set come from the European Space Agency's Sentinel-2 program, which provides free access to 10-meter resolution imagery for the whole planet with frequent return intervals. Data is provided in 100 km by 100 km scenes with 13 bands. We chose Singapore for this project because it is covered by a single tile and is a major port with many ships.

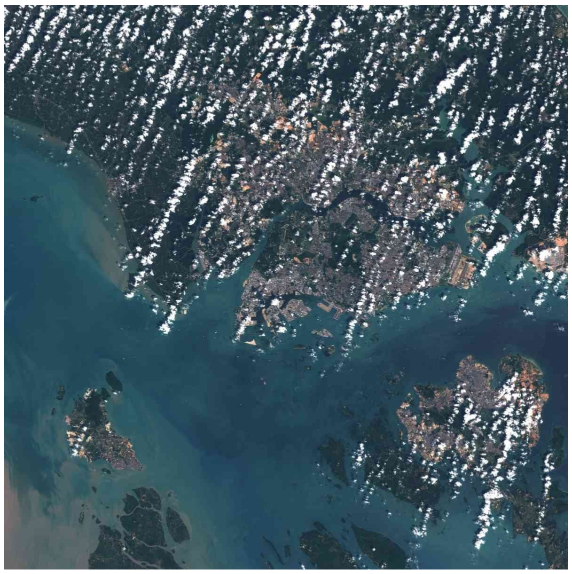

Figure 2. Sentinel-2 scene of tile 48NUG Showing Singapore.

We accessed Sentinel-2 using the Copernicus Open Access Hub API. We filtered the Sentinel-2 scenes down to those with less than 6% cloud cover that are available online, rather than in the Long-Term-Archive that has severe rate limits. This left us with three scenes of Singapore in total. The Sentinel-2 data is downloaded in a SAFE format, which contains a separate JPEG2000 file for each of the 13 bands and a 3-band RGB image. We extracted just the RGB images to match the Shipsnet dataset. We then created a validation dataset for the Sentinel-2 images by manually marking ship locations in QGIS and storing the points in a separate GeoJSON file for each scene. The number of marked ships is shown in the table below.

Capture Date	**Marked Ships**
2019-12-27	425
2019-07-05	310
2019-04-06	282

Table 1. Manually identified ships.

Methods

All operations except for manually creating the ship points were done in a Jupyter Notebook run in Google Colab, with data stored in a Google Drive directory. Our process is shown in the chart below:

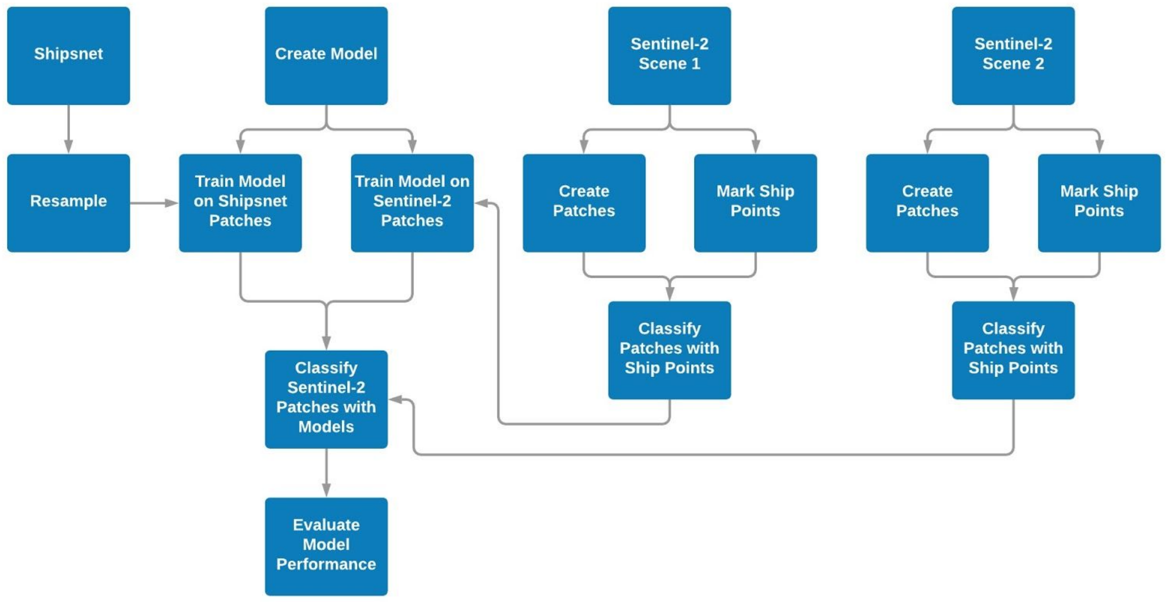
Shipsnet
Create Model
Sentinel-2 Scene 1
Sentinel-2 Scene 2
Resample
Train Model on Shipsnet Patches
Train Model on Sentinel-2 Patches
Create Patches
Mark Ship Points
Create Patches
Mark Ship Points
Classify Patches with Ship Points
Classify Patches with Ship Points
Classify Sentinel-2 Patches with Models
Evaluate Model Performance

Figure 3. Process Flowchart.

We resampled the Shipsnet patches to match the GSD of the Sentinel-2 imagery, creating patches of 32px by 32px. An exact conversion would have been a patch of 29.6px by 29.6px, but we chose 32 because it was the smallest size patch supported by the pre-trained models that we wanted to test for transfer learning.

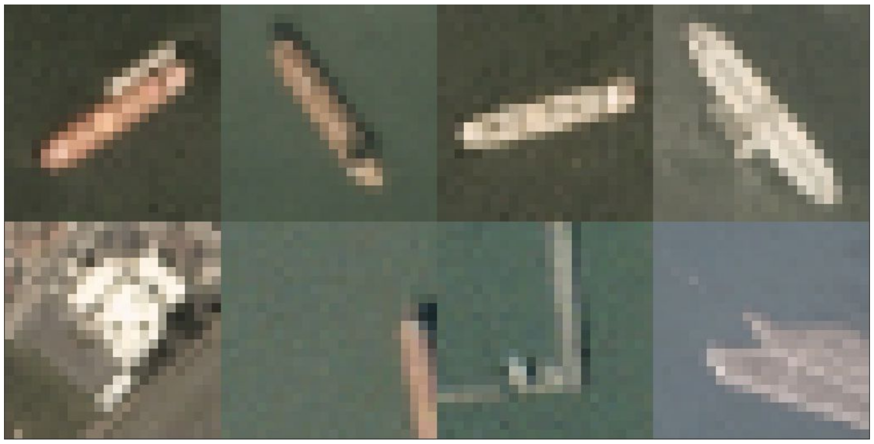

Figure 4. Resampled Shipsnet Patches

We created Sentinel-2 patches matching the size of the Shipsnet patches, using an iterative approach that created 471,969 patches of 32px by 32px, with an overlap of 16px. We then used a spatial intersection to find the patches that intersect with the manually marked ship points, creating "ground-truth" labels to use to evaluate our model's performance. In order to avoid cases where the ship point was on the edge of the patch, creating an intersection when the ship was not visible in the patch, we created a buffer from the centroid of the patch and used that to intersect to avoid literal edge cases.

Name	**Patches**	**"Ships"**	**"Not-Ships"**
Shipsnet	4 000	1 000	3 000
Sentinel Scene 1 (2019-12-27)	469 225	581	468 644
Sentinel Scene 2 (2019-07-05)	469 265	448	468 777

Table 2. Data summary.

We created 3 different deep learning models architectures: a shallow convolutional network, a deep, more complex convolutional network, and a pre-trained model using transfer learning, as well as a Support Vector Machines model. For each of these models, we trained on a 60% stratified subset of the shipsnet patches, and tested on the remainder, going through several iterations to improve performance for that architecture. We augmented data for the deep learning model using rotation and zoom. The SVM model's hyperparameters were optimized using grid search cross validation.

Model	**Description**	**Shipsnet test Accuracy**
1. SVM	● Trained on flattened RGB vectors. ● Hyperparameters optimized using nested cross-validation and GridSearchCV	Average of 94.5% across folds

2. Shallow CNN	• Trained on 32x32x3 tensors • Only one hidden convolutional layer • Some data augmentation (shearing, zooming, flipping)	97.4%
3. Deep CNN	• Several convolutional/pooling layers and a hidden dense layer • EarlyStopping and ReduceLRonPlateau callbacks • More aggressive data augmentation (shifting, rotation, more shearing and zooming)	96.2%
4. CNN with Transfer Learning	• Tried several pre-trained models for the convolutional base. DenseNet121 had the best performance • Overall, performance was much worse than simpler models	78.7%

Table 3. Model summaries.

Once we had created the models, we also trained an identical model on a balanced subset of the Sentinel-2 data from one scene to create a benchmark for performance. Then we used both the Shipsnet trained model and Sentinel-2 trained model to predict on a new Sentinel-2 scene.

Results

Based on general experience with modeling approaches we would have expected SVM to perform the worst, followed by the shallow CNN, then deep CNN, with the CNN with transfer learning performing the best. This intuition did not hold true, as we saw the CNN with transfer learning performed the worst. While all the models had high accuracy due to the overwhelming prevalence of “not-ship” patches, this was a reflection of the overwhelming number of patches that are not ships.

The baseline models trained on Sentinel-2 data had lower accuracy and specificity than the Shipsnet trained models but much higher sensitivity. Specificity for all models was higher when trained with Sentinel-2 data, meaning that the model correctly classified ships as ships, but accuracy was lower because they also classified more “not-ships” as ships. The CNN with transfer learning had higher sensitivity with Shipsnet training data relative to the other models, but lower sensitivity when trained with Sentinel-2 data.

Model	**Accuracy**	**Specificity**	**Sensitivity**	**Precision**
1. SVM (Shipsnet)	0.9944	0.9955	0.0862	0.0230
1. SVM (Baseline)	0.9075	0.9076	0.8000	0.0082
2. Shallow CNN (Shipsnet)	0.9994	0.9994	0.0517	0.0938

2. Shallow CNN (Baseline)	0.9657	0.9658	0.8889	0.0243
3. Deep CNN (Shipsnet)	0.9964	0.9975	0.1379	0.0635
3. Deep CNN (Baseline)	0.9394	0.9394	0.9556	0.0149
4. CNN with Transfer Learning (Shipsnet)	0.8781	0.8788	0.2931	0.003
4. CNN with Transfer Learning (Baseline)	0.8562	0.8566	0.4222	0.0028

Table 4. Model Performance.

Discussion

Overall, it is not easy to combine data sources and there are limits to an approach of transfer learning without any training on the new data. The Shipsnet data was not representative of all ships because it only included large cargo ships, limiting how effective the models could be, even with data augmentation. Downsampling images reduces model performance by obscuring features. Using only RGB bands makes differentiating between water and forest difficult. False positives frequently had rectangular features and sharp lines that are visually similar to the Shipsnet ship patches.

Shallower models seemed to work better. Features learned in deeper models don't translate well to new data. Overfitting to the Shipsnet data reduced model performance when predicting Sentinel-2 patches.

Mechanically, our methods would be straightforward to implement in an applied setting. However, the model performance showed that applying a model to a completely novel dataset is challenging, and we were not able to get results that would be acceptable for most applications.

Code

The code and this report are available at
https://github.com/e-chong/Singapore-Ship-Detection