MA429 Summative Project
Automatic Ship Classification On Satellite Imagery

May 2021

Contents

1 Executive Summary

Cheap availability of high-resolution satellite imagery and advances in high-performing data mining techniques in recent years have contributed to enormous excitement over a wide range of potential commercial and strategic use cases. Models that can perform automatic ship classification on satellite imagery have the potential to be used for studying and monitoring illegal fishing activity, sea smuggling routes, high traffic shipping corridors, nefarious activity in contested maritime border regions etc. In this project, we experimented with data mining methods focused on the classification of ships in satellite imagery. We trained classification models on the Ships in Satellite Imagery dataset consisting of Planet satellite imagery collected over the San Francisco Bay area, as well as on the Airbus Ship Detection Challenge dataset. On the San Francisco Bay dataset of 80×80 pixel satellite images of closely cropped ships, all of our models achieved predictive accuracy of over 90%, with CNN models achieving the strongest performance. On the binary classification task of predicting whether Airbus 768×768 pixel satellite images contain no ships at all, or contain exactly one ship, our strongest models attained true positive rates and true negative rates north of 90%. On the multi-class classification task of predicting whether Airbus satellite images contain no ships, one ship, or at least two ships, our strongest models successfully recognised the presence of multiple ships when given satellite images of multiple ships. However, they had a harder time distinguishing between one and multiple ships in satellite images containing just one ship, particularly in satellite images featuring significant noise from extraneous environmental features. Finally, we took the model trained on the San Francisco dataset, applied several generalisation techniques, and successfully predicted the presence of ships on unseen satellite images from the Suez Canal during the March 2021 canal blockage.

2 Introduction

In this project we experiment with data mining methods focused on the classification of ships in satellite imagery. The use case that we have in mind concern automatic detection systems capable of monitoring vast expanses of water to detect anomalous or interesting activity. Models that can accurately predict the presence of ships in satellite imagery can be used to monitor sea smuggling routes, illegal fishing activity, high traffic shipping corridors etc., conferring a compelling value proposition to many organisations.

We first attempt to train a series of models that are able to classify 80×80 pixel satellite images in the San Francisco Bay dataset of ships in satellite imagery. We use multiple methods for this problem, and we find that convolutional neural networks (CNNs) are able to very successfully learn this task and predict whether a given satellite image includes a ship or not. We then extend our work to classification tasks on the Airbus dataset of medium-sized 768×768 pixel satellite images. First we train and evaluate CNN binary classification models that predict whether satellite images contain no ships at all, or contain exactly one ship. This is equivalent to the classification task carried out earlier on the San Francisco dataset. We then extend our binary classification models to the multi-class classification setting, where we train on labeled data to predict whether satellite image contain no ships, exactly one ship, or multiple ships. Strong performance on both tasks suggest a broad range of possibilities for real-world use.

Having found promising results with our CNN models, we then attempt to deploy our trained models in a potential real world application by attempting to identify ships on an unseen data set. Between March 23 - 29, 2021, the Suez Canal was obstructed by the massive 400 metre Ever Given cargo vessel, clogging a vital artery of international trade and creating a serious disruption to global supply chains[1][2][3]. Hundreds of ships were backed up, with delivery for sea freight estimated to be delayed by 10+ weeks as a result[4]. This gave us fresh impetus to determine whether or not our trained classification model might be able to detect ships in satellite images of the Suez Canal region during the crisis in an attempt to evaluate the impact of the blockage on backed up shipping traffic in the Suez region.

In an attempt to solve this problem, we realised that this became more of an object detection, or counting,

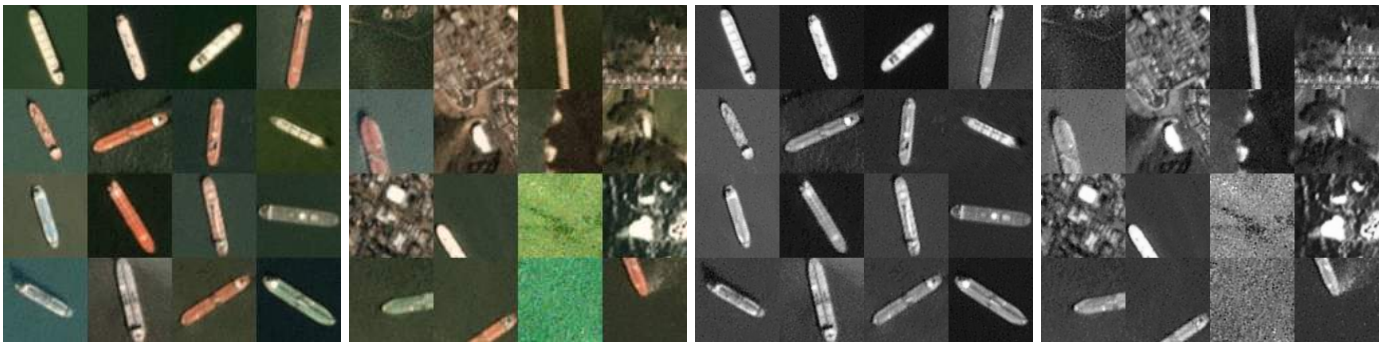
problem rather than a classification problem. In order to count ships in the Suez Canal dataset, we used a cutout approach to counting ships in the Suez region. By applying various generalisation methods to the models trained on the San Francisco Bay dataset, we were able to split up the Suez Canal satellite images into smaller chunks and predict whether these small chunks included ships or not, and finally summed up the count of ships over all image chunks to determine how many ships were in the original Suez satellite image. We conclude by demonstrating that this naive approach does have moderate success, as most ships in our Suez satellite images are correctly classified by the trained model.

3 Datasets & Methodology

3.1 Dataset 1: Ships in Satellite Imagery

In this project, we first try to classify ships using the Ships in Satellite Imagery dataset from Kaggle (available from: https://www.kaggle.com/rhammell/ships-in-satellite-imagery). This dataset consists of image chips extracted from Planet satellite imagery collected over the San Francisco Bay and San Pedro Bay areas of California. The Image chips are orthorectified to a 3-meter pixel size. There are 4,000 80×80 pixels RGB images in this dataset, with 1,000 labelled as "ship" and 3,000 labelled as "no-ship". The "ship" images are near-centred on the body of a single ship. Ships of different colours, shapes, sizes and orientations are included. The "no-ship" class includes images of the water, coasts, buildings, bridges etc. Most importantly, the partial ships are classified as the non-ships in this dataset. Sample images of each classes are shown in Figure 1a and Figure 1b.

Figure 1: Examples of Colourful and grayscale images of the two classes.



(a) Colourful ships (b) Colourful no-ships (c) Grayscale ships (d) Grayscale no-ships

We didn't perform much pre-processing on the dataset since the ships are already centred and scaled. As the data for a single 80×80 RGB image is of $3 \times 80 \times 80 = 19,200$ dimensions, it would be hard for us to apply some of the machine learning methods. Therefore, we created a black and white version of the dataset so that the dimension for an instance is now 6,400. Each attribute stands for the grayscale value of a fixed pixel. As the grayscale data range from 0 to 255, we normalise the data by dividing each column by 255. The same but grayscale sample images are shown in Figure 1c and Figure 1d.

We randomly split the dataset into a training set with 3,200 instances and a test set with 800 instances. The distribution of the ships and non-ships in the original dataset and each subset are shown in Table 1.

3.2 Dataset 2: Airbus Ship Detection

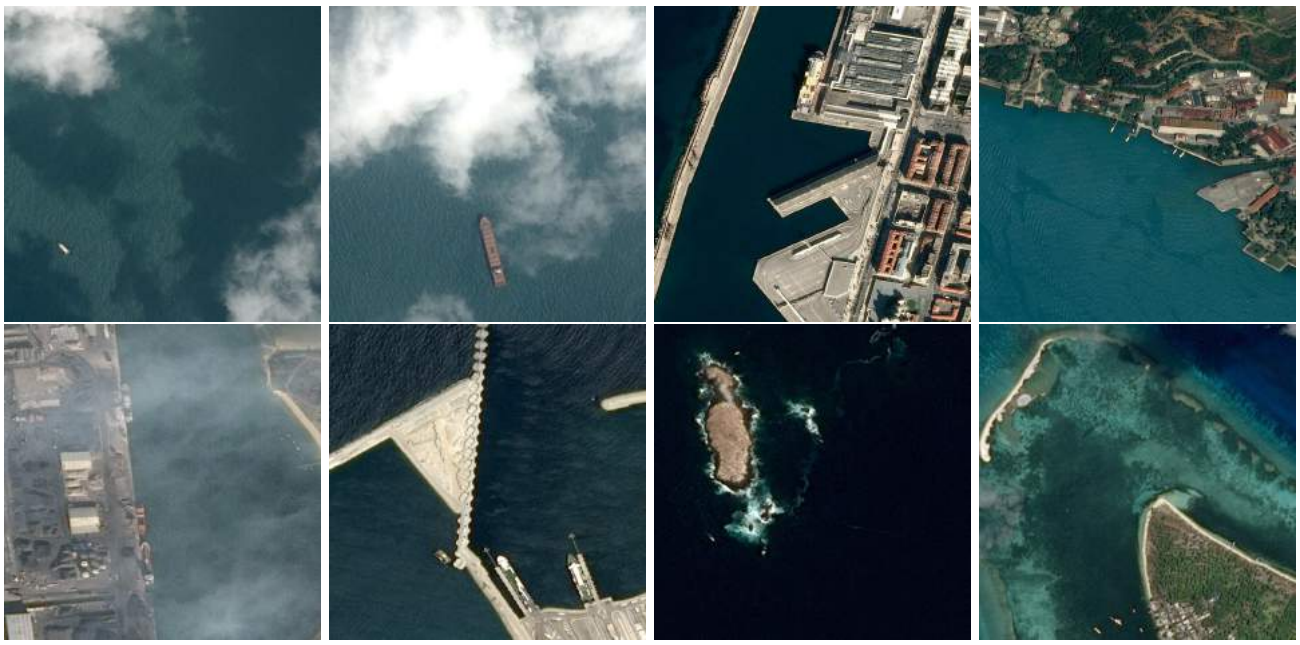
The Airbus dataset (available from: https://www.kaggle.com/c/airbus-ship-detection/overview) contains 192,556 labeled RGB satellite images, with each image 768×768 pixels, at a resolution of approximately 1.5 meters per pixel, a higher resolution than San Francisco Bay dataset. A significant proportion of images do not contain ships. For the most part, the ships in this dataset are quite small in the satellite images, making it much more

Table 1: Number of the ships and non-ships classes in each dataset.

Dataset	Class		Total
	Ship	No-ship	
Original data	1000	3000	4000
Training data	818	2382	3200
Test data	182	618	800

challenging to predict the presence of ships in images, especially compared to the Ships in Satellite Imagery 80×80 images of centered ships. Ships across different images can differ in size, and can be found in open sea, at marinas, seen leaving a dock etc. The satellite images also exhibit broad diversity in visible environmental features, from thick cloud cover and fog, to breaking waves and island clusters etc. Sample images of ships found in different environments are shown in Figure 2.

Figure 2: Examples of satellite images with ships found in different environments.



In total, the images in this dataset take up ∼27.3 GB, making it infeasible for us to train on the entire dataset given our hardware constraints for this project. Instead, we trained, validated, and tested our models on a smaller subset of this dataset, using approximately $19,000$ images across training, validation, and test sets. Due to the skewed class distribution of this dataset, we sampled uniformly at random to attain equal class distribution in the training data, while the validation and test sets retained their original class distributions. Table 20 in the Appendix shows the class distribution in the original dataset, and in the training, validation, and test sets for the binary and multi-class classification tasks. Mask segmentation information in run length encoding (RLE) format is provided for each image in this dataset in the `train_ship_segmentation_v2.csv` file. Each row in this file denotes either no ships in the image, or the location of one ship. Converting this information into boolean format, then performing a groupby operation with sum aggregation gets us the ship counts per image.

3.3 Dataset 3: Suez Canal Satellite Images

In order to try to make predictions about ships during the Suez Canal crisis, we first had to find a data set that would allow us to analyse recent satellite images from whichever region we wanted. As the original San

Francisco dataset came from Planet's[1] satellite imagery database, that was the first place we looked. We applied for and were granted an educational research account to download up to 5,000 sq. km. of data per month, with no geographical restrictions.

Planet satellites produce new images every day, but the quality of them can vary tremendously. For example, certain days have high levels of cloud cover and make it near impossible to see the surface. When selecting certain test images to run our trained models on, we narrowed our search to only include satellite images with no cloud cover and with high visibility metrics in order to give our models the best chance of detecting ships.

After taking the time to understand how to download and analyze Planet data, we referred back to the original San Francisco Bay dataset to make sure we were downloading the exact same type of image, with similar resolution and coloring. We found that the Planet data set was based on satellite images taken at a resolution of 3 meters per pixel, and had been colour-corrected. These types of satellite images are referred to as "Planet Scope OrthoTile" image types by Planet, and these are the types of images we downloaded to test our trained models. We selected dates that had the clearest satellite images, with minimal cloud cover. The only date during the Suez Blockage Crisis that we were able to analyze was March 29, the last day it was blocked. We also downloaded satellite image data from February 21 2021, March 11 2021, March 17 2021, and April 4 2021.

3.4 Evaluation Metrics

To evaluate the performance of our classification models, we primarily focused our analysis on the Accuracy, F1-score, Precision, Recall (TPR), and Selectivity (TNR) metrics of our models. Classification accuracy was used to give a basic, holistic view of how effective our models are at accurately classifying the presence of ships. To get a more detailed understanding of model performance, we also paid careful attention to Precision, Recall, and Selectivity, while reporting confusion matrices where appropriate. For our binary classification models, we didn't see an inherent benefit between wanting to minimize missed ships (Recall) or wanting to minimize the amount of times we incorrectly classify a shoreline, island, or cloud as a ship (Precision), and so focused on Accuracy and F-1 scores to gauge how well the models generally performed. For multi-class classification, we reported Precision, Recall, and Selectivity for each class, and computed both Macro (simple average) and Weighted (by the # of samples in each class) averages of these metrics across all classes. Finally, we also report Precision-Recall AUC, ROC-AUC, and the Kappa statistic where appropriate.

4 Image Classification

4.1 K-nearest Neighbours

Since a binary label (ship or no-ship) is assigned to each image in the dataset, the image classification task can be also viewed as a binary classification problem which we can solve using normal machine learning techniques. Firstly, we applied the K-nearest neighbour method to the grayscale images. To determine the optimal value of K (the number of neighbours), we apply the 5-fold cross-validation and search for the K that gives the highest accuracy. The cross-validation accuracy of models using $K = 1, 2, \ldots, 8$ are shown in Figure 3 where we see that $K = 1$ is the best choice.

As all the ships are centred and the images are changed into grayscale, the images labelled as ships are quite similar. Therefore, it is reasonable to predict new images by using the most similar image in the training set. We retrained the full model and performed predictions on the test data. The KNN model with $K = 1$ gives an accuracy of 93.00% and a recall of 93.95%, which means that it is good at finding most of the ships in the test set.

[1] See planet.com for more information.

Figure 3: Performance of KNN models using different K.

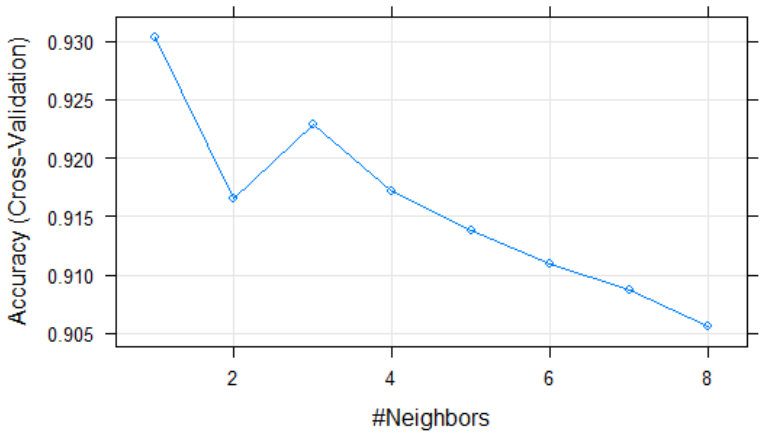


Table 2: Confusion Matrix for the KNN model

		Prediction	
		No-ship	Ship
Actual	No-Ship	71.62%	5.62%
	Ship	1.38%	21.38%

4.2 Support Vector Machine

Support Vector Machine is also a widely used method for image classification tasks. It also works efficiently on the high dimensional data, where the number of features is larger than the number of observations. This is exactly the case we are dealing with now: there are only 3,200 images in the training set but each of the instances has 6,400 features (6,400 pixels). In this section, we tried the SVM models with different kernels and applied the 5-fold cross-validation for parameter tuning.

4.2.1 Linear SVM

Linear SVM aims to find a hyperplane that separates the two classes. The parameter *cost* stands for the cost of constraints violation and controls the bias-variance tradeoff. As we can see from Figure 4, $cost = 0.001$ gives the highest accuracy of 89.80% through the cross validation. Therefore, we set $cost = 0.001$ and it took about 4 minutes to obtain the final linear SVM model. Using this linear SVM model, we are able to correctly predict 90.12% of the test images and we can evaluate its performance through the confusion matrix shown in Table 3.

Figure 4: Performance of linear SVMs with different costs.

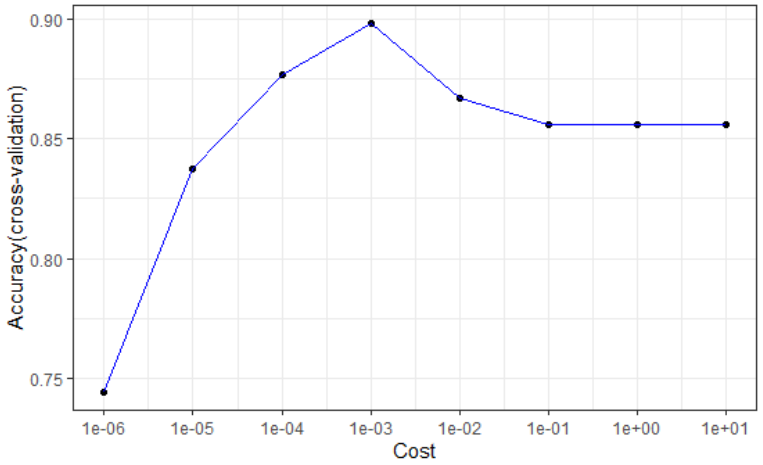


Table 3: Confusion Matrix for the final linear SVM model.

		Prediction	
		Non-ships	Ships
Actual	Non-ships	72.00%	5.25%
	Ships	4.63%	18.12%

4.2.2 Non-linear SVMs

We then further explored the classification with non-linear decision boundaries and fit the SVMs with polynomial and radial kernels.

SVMs with Polynomial kernel The polynomial kernel is $k(x, x') = (\gamma\langle x, x'\rangle + \text{coef}_0)^{\text{degree}}$. In our experiment, we kept the $\text{coef}_0 = 0$ (the default value in the 'kernlab' r package) and tuned the value of *degree*, γ and *cost* to find the model giving the best performance through cross validation. Firstly, we need to decide the degree of the polynomial kernel. We specified $cost = 1$, $\gamma = 0.001$ and then fitted the models with degrees 1, 2, 3 and 4. From Table 4, we can see that degree 2 gives the highest accuracy as well as the Kappa statistic.

Table 4: Performance of polynomial SVM models with different degrees.

Degree	Accuracy	Kappa
1	89.59%	72.49%
2	95.22%	87.74%
3	93.22%	82.92%
4	93.66%	83.85%

Then we fixed $degree = 2$ and tried models with different pairs of parameters where γ equals 10^{-5},10^{-4},10^{-3} or 10^{-2}, and *cost* equals 10^{-4},10^{-3},10^{-2},10^{-1},1,10 or 100. We can infer from Figure 5 that the model with $cost = 0.1$ and $\gamma = 10^{-3}$ has the best performance over all the 28 models.

Figure 5: Performance of polynomial SVM models using different costs and γ.

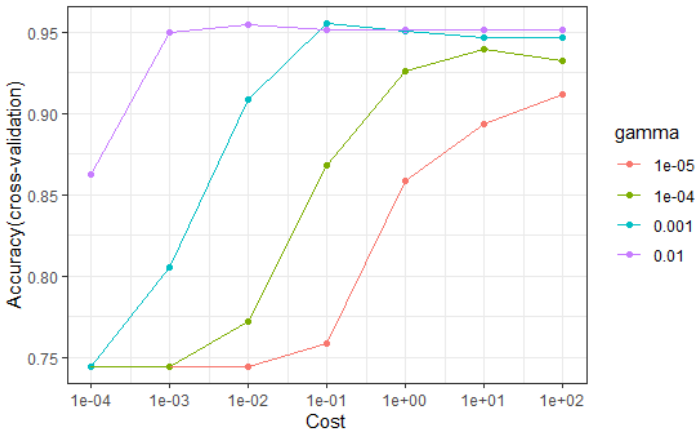


Figure 6: Performance of radial SVM models using different costs and γ.

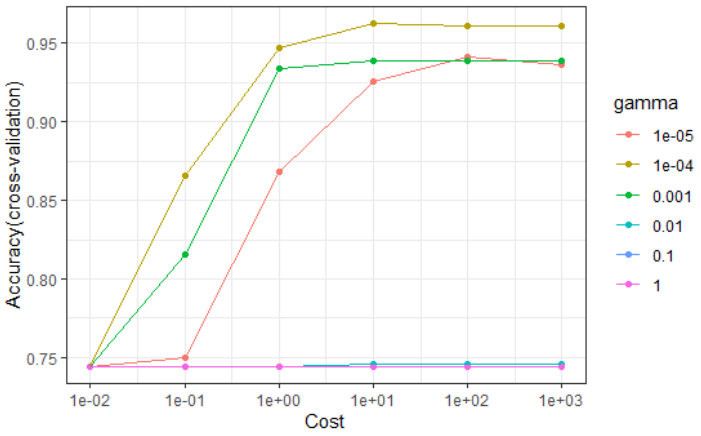


Table 5: Confusion Matrix for the polynomial SVM Model.

		Prediction	
		No-ship	Ship
Actual	No-ship	75.88%	1.37%
	Ship	4.50%	18.25%

Table 6: Confusion Matrix for the radial SVM Model.

		Prediction	
		No-ship	Ship
Actual	No-ship	75.75%	1.50%
	Ship	2.12%	20.63%

The parameters we use for the final polynomial SVM model are : $degree = 2$, $\gamma = 0.001$ and $cost = 0.1$. We implemented the model on the test set and the confusion matrix is shown above in Table 5. The model gives an accuracy of 94.13% which is about 4% higher than the previous linear SVM model.

SVM with radial kernel We then move on with the SVMs with radial kernel. The Gaussian RBF kernel can be written as $k(x, x') = exp(-\gamma||x - x'||^2)$. Here we need to decide the value of gamma and the cost

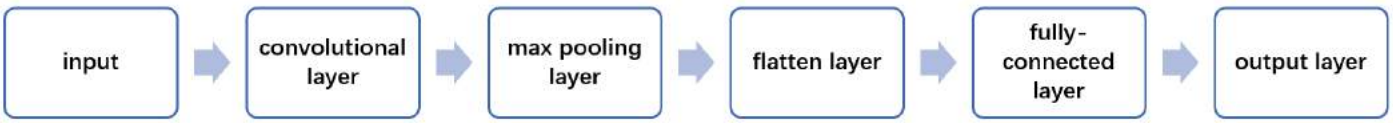
for training. We tried various models with $\gamma = 10^{-5}, 10^{-4}, \ldots, 1, 10$ and $cost = 10^{-2}, 10^{-1}, \ldots, 10^{3}$. We can conclude from Image 6 that the model using $\gamma = 10^{-4}$ and $cost = 10$ gives the highest accuracy among the 36 models and therefore we adopt this pair of parameters to train the final SVM with radial kernel model. The confusion matrix we get from the preditions made on the test data is shown in Table 6. We can see that all the three models with different kernels can correctly predict over 90% images in the test dataset. The SVM with radial kernel performs the best as its accuracy reaches 96.38 % and the F1-score reaches 91.92%.

4.3 Convolutional Neural Network

Convolutional neural networks have demonstrated outstanding performance on image classification tasks. Here, we attempt to build a simple CNN model using TensorFlow for R. To make the training process more efficient, we also use Google Colab for the access of GPU. Convolutional neural networks typically comprise of three types of layers, namely convolutional layers, pooling layers and fully-connected layers. The convolutional layer applies different filters on the previous layer and creates the output feature map. The pooling layer simplifies the information from the output from the convolutional layers and also reduces the dimensions of the feature maps. The fully-connected layer connects all the neurons in the previous layer to every single neuron of current layer.

We first begin with a very simple model which consists of only one convolutional layer, one max pooling, one fully connected layer with 64 neurons and an output layer. We apply softmax activation function in the output layer because it usually generates a well-performed probability distribution of the outputs for classification problems[5]. For the other layers of the network, we choose the ReLU activation function when needed. We used 64 3×3 filters in the convolutional layer and the 2×2 blocks in the max pooling layer. As the CNN models are able to deal with the colourful images, we implement the model on the colourful image data and thus the shape of an input instance is $80 \times 80 \times 3$. The data flow of this model is shown in figure. The flatten layer just converts the pooled feature map into an array which is then passed to the fully connected layer.

Figure 7: Data Flow Diagram of the very simple CNN model.



From this very simple model, we gradually increase the number of the layers and expect that as the network becomes deeper, the model can extract and learn the features and patterns of the input image better. For each step, we add exactly one convolutional layer and one max pooling right before the flatten layer. To evaluate the performance of these models, we reserve a validation set from the original training dataset and thus we now use 2560 images for training and 640 images for validation. The accuracy and loss of the different models on the validation set are shown in the following table

Table 7: Accuracy of the model on the validation set for each step.

Step	Number of convolutional layers	Number of max pooling layers	Accuracy
1	1	1	97.50%
2	2	2	98.75%
3	3	3	99.37%

The accuracy on the validation set reaches over 99% at the third step so we stop increasing the number of the layers and apply this final model to the test data. The details of the model are shown in the Table 8.

Table 8: Details of the final CNN model.

Layer	Layer Details
Convolutional Layer 1	64 Filters, 3x3 Kernel Size, ReLu Activation Function
Max Pooling Layer 1	2x2 Pooling Size, Stride of 2
Convolutional Layer 2	64 Filters, 3x3 Kernel Size, ReLu Activation Function
Max Pooling Layer 2	2x2 Pooling Size, Stride of 2
Convolutional Layer 3	64 Filters, 3x3 Kernel Size, ReLu Activation Function
Max Pooling Layer 3	2x2 Pooling Size, Stride of 2
Flatten layer	Flatten Size - 4096
Fully Connected Layer	Size - 64, ReLU Activation Function
Output	Softmax activation Function
Total Parameters: 337,986	

We retrain this final CNN model on the full training set (with 3200 images) and implemented the model on the test data. We also train a model with the same structure on the grayscale data and performed the prediction on test data. The confusion matrices of both models are shown in the following tables. The accuracy of the CNN trained on the grayscale images is 99.125% and it is slightly higher than the accuracy of the CNN trained on the colourful images, which equals 98.875%.

Table 9: Confusion Matrix for the CNN trained on colourful images.

		Prediction	
		No-ship	Ship
Actual	No-ship	75.87%	1.38%
	Ship	4.50%	18.25%

Table 10: Confusion Matrix for the CNN trained on grayscale images.

		Prediction	
		No-ship	Ship
Actual	No-ship	75.75%	1.50%
	Ship	2.12%	20.63%

4.4 Summary of the Experiments

Table 11 summarizes the performance measurement of all the different models we have tried. All the 6 models perform well as they can correctly predict over 90% images in the test data and the F1-scores are also relatively high. One possible explanation is that the image data set we use are pretty clean and all the ships are well centred and scaled. Not surprisingly, the convolutional neural networks have the overall best performance. As we train the CNN models on a GPU, the training process is very efficient and it only takes about 30 seconds to train a model. Therefore, we will then move on to explore the real life application of the image classification problem using our CNN model.

Table 11: Performance of all models.

Model	Accuracy	F1-score	Recall	Precision
KNN	93.00%	85.92%	93.96%	79.17%
Linear SVM	90.12%	78.59%	79.67%	77.54%
Polynomial SVM	94.12%	86.13%	80.22%	93.00%
Gaussian RBF SVM	96.37%	91.92%	90.66%	93.22%
CNN on colourful data	98.87%	97.52%	97.25%	97.79%
CNN on grayscale data	99.12%	98.09%	98.90%	97.30%

5 Airbus Experiment

5.1 Introduction

The automatic classification of ships in satellite imagery is a challenging task that has attracted a considerable amount of research over the years. While we were able to demonstrate in the previous section strong, robust predictive performance on small, centred, zoomed-in 80×80 cutouts of ships in the San Francisco Bay dataset, in practice most datasets of satellite imagery cover much larger geographic areas, with objects of interest that are much smaller relative to the size of the images. Systems that are trained on too narrow scenes of satellite imagery lose out on the ability to infer patterns and incorporate decision-useful context from broader scenes to make high fidelity predictions in real-world settings. Moreover, to monitor sea smuggling routes, illegal fishing activity, high traffic shipping corridors etc., high-calibre detection systems often need to be able to monitor vast expanses of water.

In light of this, we propose two related tasks on the Airbus dataset of medium-sized 768×768 pixel satellite images, which we will refer to as Tasks A1 and A2 going forward. First, in Task A1, we train and evaluate CNN binary classification models that predict whether satellite images contain no ships at all, or contain exactly one ship. This is equivalent to the classification task carried out earlier on the San Francisco Bay dataset. In Task A2, we extend our binary classification models to the multi-class classification setting, where we train on labeled data to predict the following three classes: "No Ship", "One Ship", and "Multiple Ships" (i.e. at least two ships). As a natural extension to the binary classification model, Task A2 broadens the range of possibilities in real-world use. For example, to monitor large areas, part of the challenge for detection systems is to be able to quickly zero in on potential areas of concern or interest. Once flagged, proposal regions can then be cataloged for further examination by other methods, in more granular detail (e.g. by a model that has been trained to recognise ships on smaller cutouts of 80×80 pixels). Such a solution requires the ability to recognise not just one ship in a medium-sized satellite image, but also the possibility of multiple ships. Moreover, a model that can quickly distinguish between settings with exactly one ship and settings with more than two ships could potentially yield interesting insights into localised pockets of suspicious activity, such as nefarious activity in contested maritime border regions.

5.2 Experiment

5.2.1 CNN Classification Models

We consider six convolutional neural network configurations for our tasks on the Airbus dataset, which we will hereby refer to by their names (A-F). These models draw inspiration from the landmark CNN architecture VGG-16 proposed by the University of Oxford's Visual Geometry Group as the basis for their submission to the ImageNet Challenge (Simonyan and Zisserman, 2015). In Table 12, we see the layers and network parameters of the six CNNs. Note that Conv3-32 denotes a convolutional layer with 3×3 filter and 32 channels.

Each network starts with fewer channels in its earlier layers, before gradually increasing the number of channels in successive layers (i.e. we see this from top-to-bottom in the table). The idea is for earlier layers to learn lower-level features, such as edges, while higher layers gradually learn to recognize more abstract features (Gu, 2017). Going from left-to-right in the table, we increase the number of layers in each architecture, starting from the shallowest CNN-A on the left. By comparing the performance of CNN-A through CNN-F, we can assess how increasing network depth affects performance and model generalizability. The choice of max pooling layer with filter size 2×2 is motivated by our prediction task and dataset: larger filters aggressively downsample intermediate feature maps, making it harder for our networks to recognise small-size objects in large satellite images. Similar reasoning motivates us to try three to four successive convolutional layers without max pooling layers early in the network in CNN-E and CNN-F. A dropout layer follows each max pooling layer to help reduce network overfitting.

A global max pooling layer follows the last convolutional layer with max pooling and dropout. This is fol-

Table 12: Proposed CNN architectures.

CNN Architectures					
A	B	C	D	E	F
Conv3-32	Conv3-32 Conv3-32	Conv3-32 Conv3-32	Conv3-32 Conv3-32	Conv3-32 Conv3-32 Conv3-32	Conv3-32 Conv3-32 Conv3-32 Conv3-32
MaxPool (2×2)					
Dropout					
Conv3-64	Conv3-64	Conv3-64 Conv3-64	Conv3-64 Conv3-64	Conv3-64 Conv3-64	Conv3-64 Conv3-64
MaxPool (2×2)					
Dropout					
			Conv3-128	Conv3-128	Conv3-128
			MaxPool (2×2)		
			Dropout		
GlobalMaxPool					
Dense-128					
Dense-1 (Sigmoid) / Dense-3 (Softmax)					

lowed by a dense fully connected layer, which is in turn connected to a single output node with sigmoid (for Task A1) or softmax (for Task A2) activation to predict the class outcome.

5.2.2 Training

All convolutional layers use the 'tanh' activation function, which have the benefit of mitigating the 'dying ReLU' problem sometimes encountered in training deeper networks[6]. We use the Adam optimizer with decaying learning rate, which tends to converge faster and gives more stable performance across epochs compared to stochastic gradient descent[7]. A batch size of 128 was chosen to moderate the tradeoff between performance and training efficiency (smaller batch sizes tend to perform better but surrender GPU training speedup advantages from vectorized matrix operations). Binary cross-entropy loss and categorical cross-entropy loss were used to train models in Task A1 and A2 respectively.

Training was achieved on the following hardware: NVIDIA GeForce GTX 1080 Ti GPU, Intel Core i7-8700 CPU, and 32GB DRAM. Since the dataset was too big to fit into memory, we streamed data from disk into the network. As a result, one potential bottleneck during training was the CPU bottleneck of feeding images into the network in between batches. However, this was effectively dealt with by increasing the number of parallel workers in our training routine.

5.2.3 Results

Task A1: Predicting Ship/No Ship
In Table 13 we have the number of parameters and running time of models A-F.

Table 13: Number of parameters and running time of models A-F.

CNN Architectures	A	B	C	D	E	F
# of Model Parameters	27,841	37,089	74,017	156,065	165,313	174,561
Running Time	8 min 46 s	11 min 48 s	13 min 29 s	15 min 42 s	17 min 12 s	20 min 14 s

In Figure 17 in the Appendix we see the training history of network CNN-F. The models' loss history, accuracy,

precision, and recall are plotted on the training and validation sets. We see that by epochs 15-20, model performance on the validation set has stabilised across all four metrics, with minimal improvement to be gained from further epochs. Notice that while recall on training and validation sets achieve ∼90% in earlier epochs, precision on the validation set consistently lags behind precision on the training set by a fairly wide margin. This is understandable given the significant class imbalance in the validation set[2]. Even if the true negative rate (aka selectivity) is high, and equivalently the false positive rate is low[3], even a small proportion of the negative class samples wrongly predicted as positive will heavily skew Precision $= \frac{TP}{TP+FP}$ lower.

Table 14 summarises the performance of CNN models A-F. Consistently high performance on both the true negative rate (aka selectivity) and the true positive rate (aka recall), provides strong evidence that despite the imbalanced class distribution in the test set, models A-F are able to make strong ship vs no ship classification decisions. Across most metrics, model performance shows strong improvement as more layers are added to preceding architectures. Notably, the monotonic improvement in model performance first from CNN-D to CNN-E, then to CNN-F, lend support to our earlier hypothesis that attempting successive convolutional layers without max pooling layers early in the network makes it easier for the network architecture to recognise small-size objects in wider resolution images. Starting from CNN-A, through to CNN-F, accuracy, F1-score, precision, and selectivity all show strong performance. Precision, in particular, improves dramatically from a low baseline of 53.2% to 72.9%!

Table 14: Performance of all models on Task A1 test set.

Model	Accuracy	F1-score	Precision	Recall	Selectivity	PR-AUC	ROC-AUC
CNN-A	86.4%	65.8%	53.2%	86.4%	86.4%	76.4%	94.0%
CNN-B	88.8%	71.3%	58.3%	91.8%	88.3%	85.3%	96.5%
CNN-C	90.9%	74.2%	65.1%	86.3%	91.7%	85.2%	96.1%
CNN-D	88.6%	69.6%	58.3%	86.4%	89.0%	83.7%	95.2%
CNN-E	91.6%	75.3%	68.5%	83.8%	93.1%	85.4%	95.9%
CNN-F	92.6%	77.3%	72.9%	82.3%	94.5%	86.2%	95.4%

Table 15: Confusion matrices of models CNN-B and CNN-F.

(a) Model CNN-B

		Prediction	
		No Ship	One Ship
Actual	No Ship	74.90%	9.94%
	One Ship	1.24%	13.92%

(b) Model CNN-F

		Prediction	
		No Ship	One Ship
Actual	No Ship	80.21%	4.63%
	One Ship	2.69%	12.47%

While the deepest model CNN-F outperforms the other five models across nearly all metrics, on model recall it shows worse performance, especially compared to model CNN-B. Table 15 shows the confusion matrices of these two models evaluated on the test set. Compared to the model CNN-B, the percentage of false positives generated by CNN-F more than halve, from 9.94% to 4.63%, accounting for the dramatic improvement in model precision. On the other hand, the percentage of false negatives generated by CNN-F increases, from 1.24% to 2.69%, accounting for the decrease in model recall.

Task A2: Detecting Multiple Ships

Table 16 summarises the performance of CNN models A-F on Task A2. Similar to Task A1, models CNN-B and CNN-C demonstrate stronger performance on model recall (macro and weighted), compared to the deeper models CNN-E and CNN-F. In addition, CNN-C also outperforms both CNN-E and CNN-F on model precision

[2]The validation set has 3,781 images with no ships vs 647 images with one ship, i.e. over 85% of the validation set consists of negative class images.

[3]Where TNR $= \frac{TN}{TN+FP} = 1 - FPR$.

(macro and weighted) and on accuracy too, suggesting a possible saturation point in the usefulness of increasing network depth past a certain number of layers on this task.

Table 16: Performance of all models on Task A2 test set.

Model	Accuracy	Precision (Macro)	Precision (Weighted)	Recall (Macro)	Recall (Weighted)	Selectivity (Macro)	Selectivity (Weighted)
CNN-A	76.9%	58.8%	84.9%	68.7%	76.9%	89.9%	93.1%
CNN-B	81.3%	63.0%	86.4%	72.9%	81.3%	91.7%	93.9%
CNN-C	84.5%	67.2%	86.8%	72.5%	84.5%	90.9%	88.3%
CNN-D	82.8%	65.0%	86.6%	73.3%	82.8%	91.1%	90.6%
CNN-E	83.2%	65.0%	86.0%	71.6%	83.2%	90.5%	88.3%
CNN-F	83.2%	64.1%	84.5%	68.3%	83.2%	89.1%	84.1%

In Table 17, we have the multi-class confusion matrices of models CNN-C and CNN-F evaluated on the test set. In the bottom right corner of both confusion matrices, we see that when given images with multiple ships, both models do a comparatively strong job of recognising the presence of multiple ships (as opposed to just one ship). On the other hand, when given images with just one ship, the models do a relatively poor job of distinguishing between one ship or multiple ships. As a result, while recall on the 'Multiple Ships' class is 79.0% and 67.7% for models CNN-C and CNN-F respectively, recall on the 'One Ship' class is just 46.5% and 45.5% on these two models. Table 18 summarises the precision and recall metrics for each class.

Table 17: Confusion matrices of models CNN-C and CNN-F.

(a) Model CNN-C

		Prediction		
		No Ship	One Ship	Multiple Ships
Actual	No Ship	71.36%	2.80%	3.27%
	One Ship	2.57%	6.68%	5.13%
	Multiple Ships	0.43%	1.29%	6.46%

(b) Model CNN-F

		Prediction		
		No Ship	One Ship	Multiple Ships
Actual	No Ship	71.07%	3.59%	2.77%
	One Ship	3.48%	6.55%	4.36%
	Multiple Ships	0.70%	1.94%	5.53%

Table 18: Precision, recall, and selectivity on each class for CNN-C and CNN-F.

	CNN-C			CNN-F			
Class	Precision	Recall	Selectivity	Precision	Recall	Selectivity	# of Samples
No Ship	96.0%	92.2%	86.7%	94.4%	91.8%	81.5%	3,429
One Ship	62.1%	46.5%	95.2%	54.2%	45.5%	93.5%	637
Multiple Ships	43.5%	79.0%	90.9%	43.7%	67.7%	92.2%	362
Macro Average	67.2%	72.5%	90.9%	64.1%	68.3%	89.1%	4,428
Weighted Average	86.8%	84.5%	88.3%	84.5%	83.2%	84.1%	4,428

Examples of satellite images correctly predicted as having multiple ships by both the CNN-C and CNN-F models are shown in Figures 8 and 9. In the first row of images, we see a sample of images with 2, 4, 6, and 6 ships (l-r). Across many satellite images with low visibility, both models are able to successfully make out the presence of multiple ships. In the second row of images, we see a sample of images with multiple ships set against a variety environmental features: (from l-r) a shipping terminal, a significant cluster of high density housing, a private marina, and a neighbourhood of large beachside mansions. Given satellite images with multiple ships of different sizes, often set against a diverse array of terrain and environmental features, both models are able to accurately recognise the presence of multiple ships.

On the other hand, when the models are shown satellite images containing just one ship, they often find it difficult to sort through the noise of extraneous environmental features. This is clearly evident in the images in Figure 10. Rectangular shaped protruding features on the shorelines of the first two images give the false impression that there may be multiple ships. In the last two images, what appears to be ships at different resolutions turn out to be something else entirely.

Figure 8: Examples of satellite images correctly predicted as having multiple ships by both models C & F.

Figure 9: Examples of satellite images correctly predicted as having multiple ships by both models C & F.

Figure 10: Examples of satellite images with one ship incorrectly predicted as having multiple ships.

6 Counting Ships in Suez Canal Satellite Images

As we saw in the previous section, our trained multi-ship classification CNN model was unable to "count" the number of ships in image, as it only differentiated between classes of images with no ship, 1 ship, and 2+ ships. Therefore, we had to try to take a different approach when attempting to count the number of ships in a satellite image from the Suez Canal satellite image data set.

One potential naive approach that seemed to be worth attempting was to cut up the large Suez Canal image into smaller individual images, use one of our trained ship classification models to predict the existence of a ship in the cut up Suez Canal images one at a time, and then sum up all of the positive predictions across the entire set of cut up images. This sliding window approach is very similar to the approach that was taken in Gallego et. al [8], where they produced very promising results in identifying ships in larger satellite images.

Recall that the Suez Canal dataset has a resolution of 3 meters per pixel, similar to the resolution of the San Francisco Bay dataset.

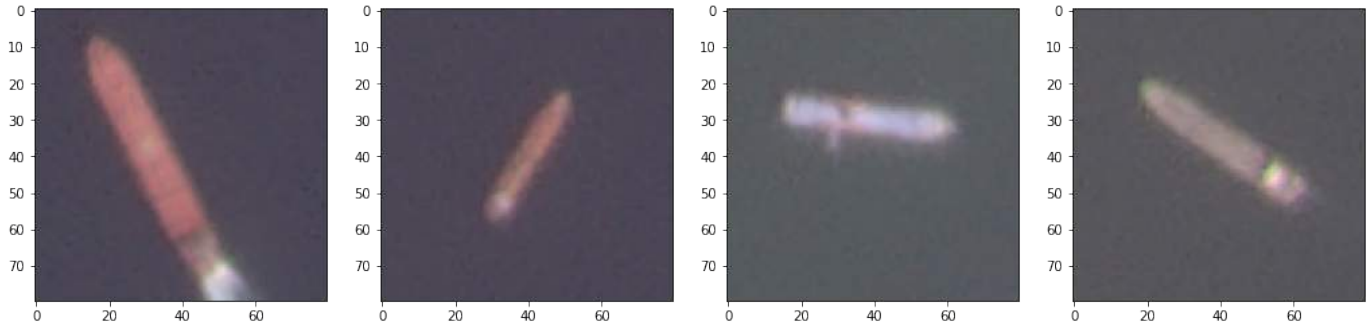
Meanwhile, the Airbus dataset had an image resolution of approximately 1.5 meters per pixel. The size of the images that were used for training on the first model were 80×80 pixels, while the Airbus data set images were all 768×768 pixels. Essentially, we could decide to cut up the Suez Canal satellite image ($4000+ \times 4000+$ pixels) into images of 80×80 and make predictions using the first CNN model we trained on the San Francisco data set. Alternatively, we could cut the Suez image into 768×768 pieces and make predictions using the second model, trained on the Airbus dataset. Due to the fact that the 80×80 images, covering an area 240 meters by 240 meters, are much less likely to have multiple ships in them, and that the resolutions of the San Francisco dataset match those of the Suez images, we decided to use the model trained on the San Francisco Bay dataset (in Section 4 of this report) to make predictions on images from the Suez satellite images. The Suez satellite image from March 29, 2021, the last day that the canal was blocked by the beached cargo ship, is shown in Figure 11.

Figure 11: Original Suez Canal Satellite Image - March 29 2021

6.1 Application of Trained CNN Model to Suez Canal Examples

Unfortunately, the Suez data set is unlabelled, so we can't use traditional loss metrics to gauge how well our model will generalize to this new unseen data set. To get a base understanding of how well our pre-trained model performed on the Suez images, we started by manually identifying some 80×80 pixel cut-outs centred around ships, as similar as possible to the original training set data, expecting our model to be able to easily recognize the ships in these images. Figure 12 below shows these test cutouts we used, as well as the output predictions made from our pre-trained model.

Figure 12: Initial Tests on Suez Image Cut-Outs



(a) Prediction - 0.01% Ship (b) Prediction - 3.74% Ship (c) Prediction - 0.01% Ship (d) Prediction - 99.48% Ship

As shown in Figure 12 above, our pre-trained model does a poor job of identifying ships in the Suez data set, even when they are nicely centered, as it only identifies the last image as a ship. Without a labelled test set for the Suez classification problem, it's hard to perfectly diagnose the issue, but we saw these results as potentially being a sign of over-fitting to certain features in the original data set. It's possible that the types of cargo ships present in the San Francisco Bay data set are a bit different than those usually found in the Suez Canal

satellite images, and if our model is over-fitting to only recognise ships in the San Francisco dataset it while have a hard time generalising to other ships. The next step that we took was to attempt to try to re-train the original CNN model in a way that might allow us to generalise to other image data sets a bit better.

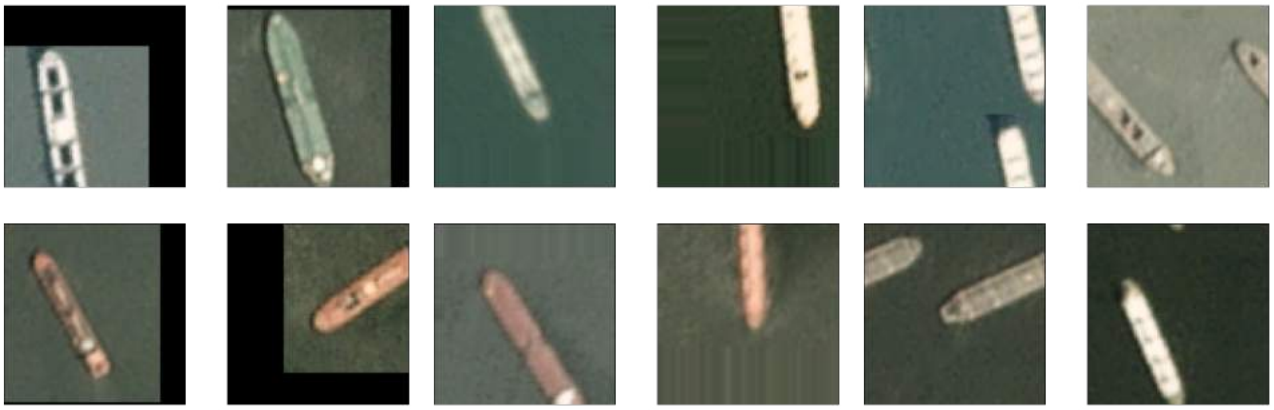
6.2 Network Architecture and Generalisation Methods

The two methods we found to be most effective in allowing our model to generalise better to the Suez dataset were drop-out and data augmentation.

Drop-out is applied by randomly eliminating nodes in the neural network at each training step, which allows us to aggregate the performance of different model architectures, similar to an ensemble model.[9] Drop out creates "thinner" networks that have less nodes, and then essentially averages all of these trained thin networks when it comes time to make predictions on a test set to avoid over-fitting. After tinkering a bit with the hyper-parameters for the neural network architecture, we found that the architecture in Table 21 in the Appendix to be most effective at being able to generalise to the Suez dataset. Note that the only significant difference between this architecture and the one that was originally trained on the San Francisco dataset is the inclusion of dropout layers.

Data augmentation is a way of generating more training data, and applying different types of noise or alterations to the data to avoid the model over-fitting to the original image training set.[10] The biggest issue that we had with the original San Francisco Bay training set was that the images labelled as having ships all had the ships very well centred, as these were manually labelled by the creator of the original data set. The dataset did also have examples of partial ships, where for example only half of a ship was present in the image, but these examples were all labelled in the no-ship class. When deploying this model on unseen data, we would like our model to be able to recognise a partial ship, otherwise our model would only be able to recognize ships that were properly centered on the image. Therefore, we originally focused our data augmentation efforts on this issue of partial ships, and attempting to augment the data in a way that would allow the model to recognize partial ships as belonging to the ship class. Since the class of no ship images was collected randomly, it was not necessary to apply similar data augmentation methods to those images.

Figure 13: Examples of Image Shifting Methods



(a) Black Border Method (b) Nearest Pixel Method (c) Wrapping Method

Our first approach at data augmenting was focused on shifting the images horizontally and vertically, and doing so quite drastically, shifting images enough so that the previously centered ship images would now look like partial ships. In order to shift images, there are a couple of different approaches we could take when filling the empty space. First, we could shift images and fill the empty space with a black border. We could also fill the empty space with the color of the nearest pixel to the edge. Finally, we could also fill the empty space by "wrapping" the image, so that the portion of the image that had been shifted out of picture was now replaced

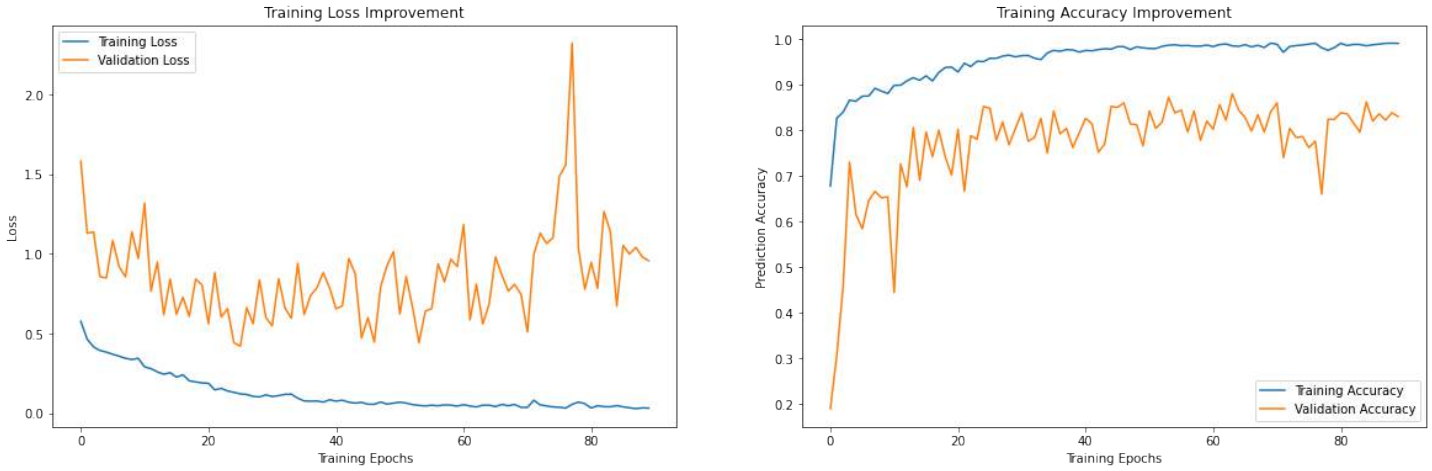
on the border. These three potential approaches are illustrated below.

Initially, we found that the nearest pixel method was giving us the best results when trying to identify partial ships in the cut-up Suez satellite images. However, as explained later in this section, we then changed our prediction approach slightly to avoid having to identify partial ships, as we believe that the existence of partial ships in the original data set labelled as "no ship" made it very hard for our model to identify the partial ships with a high level of accuracy. For this modified approach, we found that we were getting better results with the wrap method, which is the method used in the final version of the model. Since our new approach wasn't so dependent on being able to classify partial ships, we shifted our focus of the data augmentation to be about adding a wide variety of training examples of ships to the original data set. In addition to shifting images, we also randomly applied horizontal reflections, vertical reflections and angled rotations to the original images, and combined these augmented images with the original data set.

6.3 Training and Testing the Generalised Model

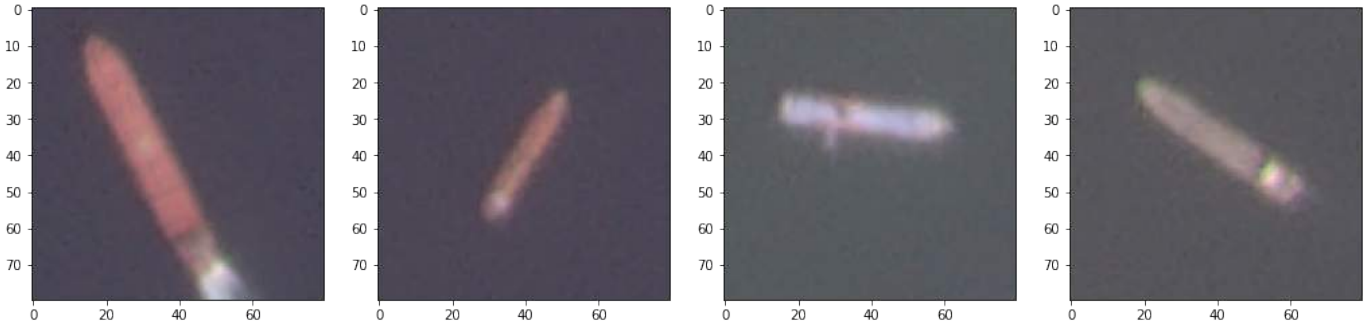
Since we weren't using the new model to make predictions on the images in the San Francisco data set, we decided to use the entire original data set, along with the augmented images of the ship class. So we ended up with a total training data set of 5,000 images, consisting of the original 4,000 images in the data set, along with an additional 1,000 augmented images of the positive class that we created. When training this new model architecture with the dropout regularization method, we also used a longer training time to give the model more chance to converge to the optimal baseline. We trained the model for 90 epochs, and Figure 14 shows the training performance on both the training and hold-out validation sets, where the validation set was a random 10% subset of the 5,000 image data set.

Figure 14: Training Performance of the Dropout Model on Augmented Data



As shown in the figure above, we see very good loss and accuracy improvement on the training data set, with worse results on the holdout validation set. Although this is potentially a worrisome sign of over-fitting, this is to be expected with a model that is trained using dropout and augmented data, and final accuracy rates of around 85% on the validation set are promising for the model's ability to generalize to unseen satellite images. Since we don't have a labelled test set from the Suez Canal satellite images, we were unable to evaluate the performance of this generalized model in a rigorous manner. In order to gauge whether this new methodology has had an improvement in it's ability to potentially generalize to the Suez data set, we again run a limited number of test predictions on the four ship images that we've manually cut out from the large satellite image. These updated results are illustrated in Figure 15.

Figure 15: Testing the Generalized Model on Suez Image Cut-Outs



(a) Prediction - 8.31% Ship (b) Prediction - 98.99% Ship (c) Prediction - 98.50% Ship (d) Prediction - 84.92% Ship

As shown in the figure above, we now see much more promising results in our model's ability to generalize to the Suez data set, as our model now correctly classifies 3 of the images as ships.

6.4 Approaches for Counting Ships in the Suez Canal Image

The final step is now to attempt to run the generalized model over the entire Suez satellite image. As alluded to above, we approached this problem in a couple of different ways. At first, we simply cut up the original image into non-overlapping 80×80 pixel chunks and ran our model on those chunks to determine whether it included a ship or not. This approach led to most of the ships in the Suez image chunks being partially cut off, and while our model trained on augmented data with heavily shifted images did correctly classify some of these, it was still missing the majority of the ships in the satellite image.

We then decided to attempt to cut up the Suez satellite image into overlapping chunks, so as to maximize the number of predictions we could make on images with ships that were relatively well centered. To do this, we moved our 80 x 80 pixel window 20 pixels at a time, similar to how a CNN model cycles through an image with a certain stride, allowing for overlap of analyzed image segments. Unfortunately, this approach does exponentially increase the amount of time it takes to make a prediction on the entire satellite image, and making predictions on the large satellite images took over 20 minutes, compared to a run-time under 2 minutes when not using the overlapping image chunk approach. With this new approach, we were able to correctly identify many more ships, and now the only issue was to find a way to avoid double-counting ships as they could now show up in multiple overlapping image chunks. To solve this, every time we made a "ship" prediction on an image chunk, we saved the location of the center of the image, and once we finished making predictions on the entire image we removed predictions that were within 80 pixels of another positive prediction. This method is far from perfect, as we can see in the test image that this distance might not be enough to avoid overlapping counts for larger ships. However, increasing this minimum distance would have resulted in missing ships that were close together. Since the vast majority of cargo ships are between 100 meters and 400 meters in length, we felt that using a distance of around 240 meters (80 pixels x 3 meters per pixel) resulted in a good trade-off between double counting ships and missing unique ships. The final set of non-overlapping ship predictions for the March 29 satellite image is shown in Figure 16, with ship predictions marked by the red squares in the image. Additional prediction outputs from other dates within the February-April 2021 period are provided in the Appendix.

As shown in Figure 16, our generalized model does an impressive job of identifying most of the ships in the satellite image. However, it's far from being able to translate the 99% accuracy rates that we see in training to making such accurate predictions on an unseen data set. In this case, out of the 79 unique ship predictions that the model makes, 15 of them appear to be either false positives or instances of double counting, as it has classified shorelines and small islands as ships, while the model has also missed 7 additional ships that appear to be in the satellite image. As shown in Table 19, our model correctly identifies that there is a drastic increase

in the number of ships in the region starting March 29, the last day of the blockage.

Figure 16: Generalized Model Making Ship Predictions on the Suez Canal Satellite Image

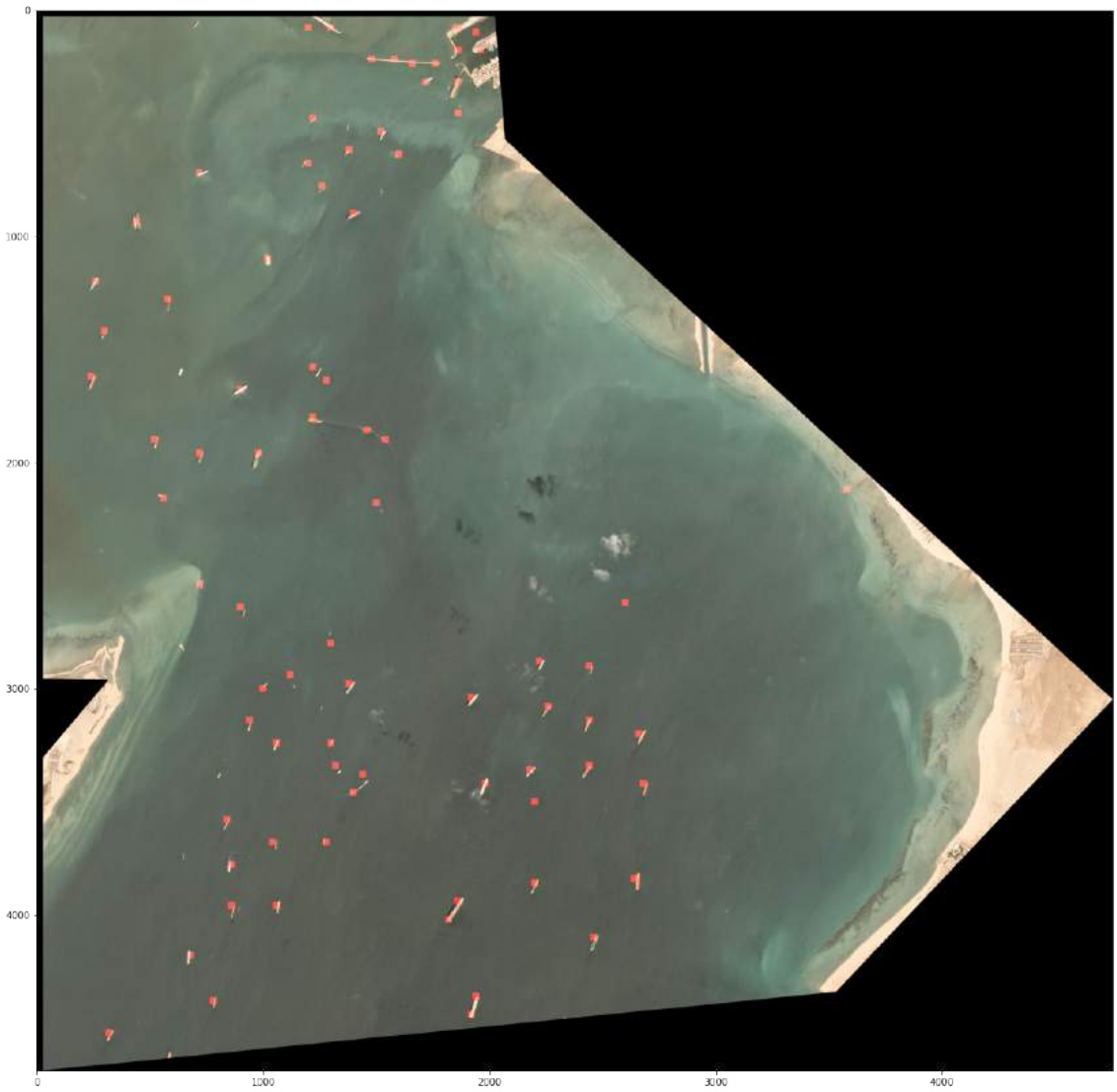


Table 19: Predicted Ships in the Suez Canal Region Over the Analyzed Period

Date	Predicted Ships
February 21, 2021	29 Ships
March 11, 2021	30 Ships
March 17, 2021	28 Ships
March 29, 2021	79 Ships
April 4, 2021	42 Ships

Once again, it is hard to rigourously evaluate the performance on a data set like the Suez Canal images as the data isn't labelled, but we believe that the methods we attempted show that at the very least this sort of approach to counting ships in satellite images could be moderately successful for high-level monitoring of maritime traffic patterns.

7 Conclusion

7.1 Summary of Results

In this project we experimented with data mining methods focused on the classification of ships in satellite imagery. We first explored the performance of KNN, SVM, and convolutional neural network methods on the simpler San Francisco Bay dataset of 80×80 pixel satellite images of nicely centred ships. The results from this experiment are shown in Table 11. All the models we tried can give the accuracy of over 90% and the CNN models perform the best in this image classification task.

We reasoned that the closely cropped, zoomed-in nature of this dataset - where the objects of interest are extremely large relative to the size of the image - made it almost trivial for a range of methods to successfully predict whether an image contained one ship or no ships at all. In real-world settings however, high-calibre detection systems often need to be able to monitor vast expanses of water, with objects of interest that are much smaller relative to the size of the images. As a result, we next explored the performance of CNN classification models on the Airbus dataset of medium-sized 768×768 pixel satellite images. Ships in these satellite images are much smaller in size relative to the size of the satellite images. Moreover, these ships are set against an enormous variety of background environmental and terrain features, which extend the trained models' ability to make predictions in a vast array of real-world settings. This comes at the expense of adding considerable challenge to the prediction task however.

On the binary classification task of predicting whether Airbus satellite images contain no ships at all, or contain exactly one ship, the deepest network (CNN-F) with seven convolutional layers demonstrated the strongest performance across most metrics. The test set results from this task are shown in Table 14. Performance by models CNN-A to CNN-F confirmed our initial hypothesis that attempting successive convolutional layers without max pooling layers early in the network makes it easier for the network architecture to recognise small-size objects in wider resolution images. Given a satellite image containing one ship, the CNN classification models are able to accurately predict the presence of a ship, consistently achieving true positive rates of over 80% across all models, peaking at 91.8% with model CNN-B. Given a satellite image containing no ships, the best performing model (CNN-F) consistently recognises the absence of ships, predicting 'No Ship' 94.5% of the time. All in all, these results demonstrate strong performance on a much more challenging dataset.

On the multi-class classification task of predicting whether Airbus satellite images contain no ships, one ship, or at least two ships, the shallower network CNN-C with four convolutional layers, each with a moderate number of channels (i.e. 32 or 64 channels, as opposed to a 128 channel layer), performed the best across most metrics. The precision, recall, and selectivity on each class for the CNN-C model are shown in Table 18. Unsurprisingly, when given a satellite image containing no ships, the model demonstrates comparable performance to the best performing model on the binary classification task, successfully predicting 'No Ship' 92.2% of the time. Given a satellite image containing at least two ships, the model is able to successfully predict 'Multiple Ships' 79.0% of the time, a strong outcome. On the other hand, given a satellite image containing just one ship, the model seems to be uncertain about whether there is 'One Ship' or 'Multiple Ships'. A representative sample of satellite images with one ship incorrectly predicted as having multiple ships was shown in Figure 10. Often the noise from extraneous environmental features (e.g. rectangular shaped protruding features on the shoreline) made it difficult for the trained model to distinguish between one and multiple ships. In a use case where interesting or concerning tracts of satellite images are first flagged for further examination by other methods (in more granular detail), it is better that such a system errs on the side of caution and predicts 'Multiple Ships' rather than 'No Ship' in the event that it does misclassify a satellite image containing just one ship (i.e. it is better to have a false positive on the 'Multiple Ships' class than to default to predicting 'No Ship' in this case). Since this is indeed what happens in our multi-classification model, we were pleased with its overall promise for real-world use in an automatic detection system.

In recent years, one promising line of research inquiry has focused on training predictive models on small/medium-

sized satellite images, which can then be generalised to much larger satellite scenes (e.g. 5000×5000 pixels satellite images) via cutouts of smaller resolution patches[4]. Note that this method has a natural synergy with the automatic detection systems described earlier. To monitor smuggling routes, busy shipping corridors, and contested maritime border regions, high-calibre detection systems often need to monitor vast expanses of water, while having the capability to examine smaller proposal regions in more granular detail. The cutout method similarly leverages models trained on smaller-sized satellite images in order to operate with a more expansive field of view. We were thus motivated to try a method along this vein to analyse expansive satellite imagery taken over the Suez Canal in late March as an interesting empirical application of our classification models. We decided to use an augmented version of the model trained on the San Francisco dataset to make predictions on images from the Suez Canal satellite images. Using generalisation methods such as random dropout and data augmentation, we were able to train a relatively simple convolutional neural network that was moderately successful in identifying ships in the Suez satellite images, and which also confirmed the significant impact that the Suez Canal blockage of March 2021 had on maritime transport in the region.

7.2 Ethical Implications

While deep learning has provided significant advancements in our ability to extract information and insights from satellite images, there are still some important ethical questions to be asked relating to privacy and security. Satellite images are largely being collected and analyzed without the consent of those present in the images. To the extent that certain policy decisions are now being made from analyzing satellite images, this is a potential issue that AI practitioners need to consider. Researchers also need to be aware of the fact that state of the art models are bound to be implemented for military and security reasons by governments and surveillance agencies around the world. If image recognition technology gets to a point where we can accurately identify and observe individuals from satellites in space without their knowledge, is that technology that should be kept in the hands of a few, or should we make steps to make sure that all advancements in these technologies are accessible to all?

7.3 Further Work

Building off of what attempted with the Suez Canal empirical application, as a next step, we could have leveraged the multi-class classification model trained in Section 5 to aid in the Suez counting problem. For example, we could have first fed the Suez images through this multi-class model to separate images into classes of no ship, 1 ship, and 2+ ships, and then used the 80×80 sliding window approach on the images in the 2+ ships class. However, given the resolution difference between the Airbus data set and the Suez data set, it's unclear how well that model would have generalised to the new data, but it would have been interesting to see nonetheless.

In Computer Vision research, this task of counting items in images is known as object detection. While image classification models tend to look for certain features within images to determine whether or not that image is of a certain type, object detection models attempt to identify objects and generate borders (i.e. bounding boxes) around these objects to determine where in the image a certain object is located. The Kaggle competition that provided the Airbus data set was an object detection competition, where borders around ships were provided as part of the training data set to allow models to learn how to generate these borders when making object detection predictions. Although we were able to come up with a moderately successful approach by cutting the Suez image into smaller chunks and making predictions on those one at a time, state of the art object detection models would have experienced much higher accuracy rates, and would have allowed us to make accurate predictions on hundreds of Suez images and to then generate high-level analytics on ship movements in the region over time[11].

[4](Lin, 2018) for example proposes cut out patches of 321×321 pixels each.

8 References

[1] "Lex: Suez Canal/supply chains: stuck ship sinks trade flows". In: *Financial Times* (Mar. 2021). URL: https://www.ft.com/content/273433b1-6677-431a-8037-1fd9c62adf84.

[2] David Sheppard and Heba Saleh. "Suez Canal clearance could take 'weeks', says salvage company". In: *Financial Times* (Mar. 2021). URL: https://www.ft.com/content/31dbe2b9-3ff2-4bcc-a3e3-813600caeb49.

[3] Vivian Yee. *Ship Is Freed After a Costly Lesson in the Vulnerabilities of Sea Trade*. Mar. 2021. URL: https://www.nytimes.com/2021/03/29/world/middleeast/suez-canal-ever-given.html.

[4] Harry Dempsey and Stefania Palma. "Global ports grapple with backlog from Suez Canal blockage". In: *Financial Times* (Apr. 2021). URL: https://www.ft.com/content/da6c709d-29c3-4f8b-b37d-a3480c6938e7.

[5] Guo et al. "Simple convolutional neural network on image classification". In: *2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA)*. 2017, pp. 721–724. DOI: 10.1109/ICBDA.2017.8078730.

[6] Lu et al. "Dying ReLU and Initialization: Theory and Numerical Examples". In: *Communications in Computational Physics* 28.5 (June 2020), pp. 1671–1706. ISSN: 1991-7120. DOI: 10.4208/cicp.oa-2020-0165. URL: http://dx.doi.org/10.4208/cicp.OA-2020-0165.

[7] Kingma and Ba. "Adam: A Method for Stochastic Optimization". In: (2017). arXiv: 1412.6980 [cs.LG].

[8] Gallego et al. "Automatic Ship Classification from Optical Aerial Images with Convolutional Neural Networks". In: *Remote Sensing* 10.4 (2018). ISSN: 2072-4292. DOI: 10.3390/rs10040511. URL: https://www.mdpi.com/2072-4292/10/4/511.

[9] Srivastava et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958. URL: http://jmlr.org/papers/v15/srivastava14a.html.

[10] Perez and Wang. "The Effectiveness of Data Augmentation in Image Classification using Deep Learning". In: (2017). arXiv: 1712.04621 [cs.CV].

[11] Schöller et al. "Assessing Deep-learning Methods for Object Detection at Sea from LWIR Images". In: *IFAC-PapersOnLine* 52.21 (2019). 12th IFAC Conference on Control Applications in Marine Systems, Robotics, and Vehicles CAMS 2019, pp. 64–71. ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2019.12.284. URL: https://www.sciencedirect.com/science/article/pii/S240589631932169X.

[12] Simonyan and Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: (2015). arXiv: 1409.1556 [cs.CV].

[13] Gu et al. "Recent Advances in Convolutional Neural Networks". In: (2017). arXiv: 1512.07108 [cs.CV].

[14] Bentes et al. "Ship Classification in TerraSAR-X Images With Convolutional Neural Networks". In: *IEEE Journal of Oceanic Engineering* 43.1 (2018), pp. 258–266. DOI: 10.1109/JOE.2017.2767106.

[15] Feng et al. "Towards Automated Ship Detection and Category Recognition from High-Resolution Aerial Images". In: *Remote Sensing* 11.16 (2019). ISSN: 2072-4292. DOI: 10.3390/rs11161901. URL: https://www.mdpi.com/2072-4292/11/16/1901.

[16] Hu et al. "Salient Ship Detection via Background Prior and Foreground Constraint in Remote Sensing Images". In: *Remote Sensing* 12.20 (2020). ISSN: 2072-4292. DOI: 10.3390/rs12203370. URL: https://www.mdpi.com/2072-4292/12/20/3370.

[17] Li et al. "Ship detection and classification from optical remote sensing images: A survey". In: *Chinese Journal of Aeronautics* 34.3 (2021), pp. 145–163. ISSN: 1000-9361. DOI: https://doi.org/10.1016/j.cja.2020.09.022. URL: https://www.sciencedirect.com/science/article/pii/S1000936120304544.

[18] Lin et al. "Fully Convolutional Network With Task Partitioning for Inshore Ship Detection in Optical Remote Sensing Images". In: *IEEE Geoscience and Remote Sensing Letters* 14.10 (2017), pp. 1665–1669. DOI: 10.1109/LGRS.2017.2727515.

9 Appendix

Figure 17: Training history of CNN-F (clockwise: loss, accuracy, precision, and recall).

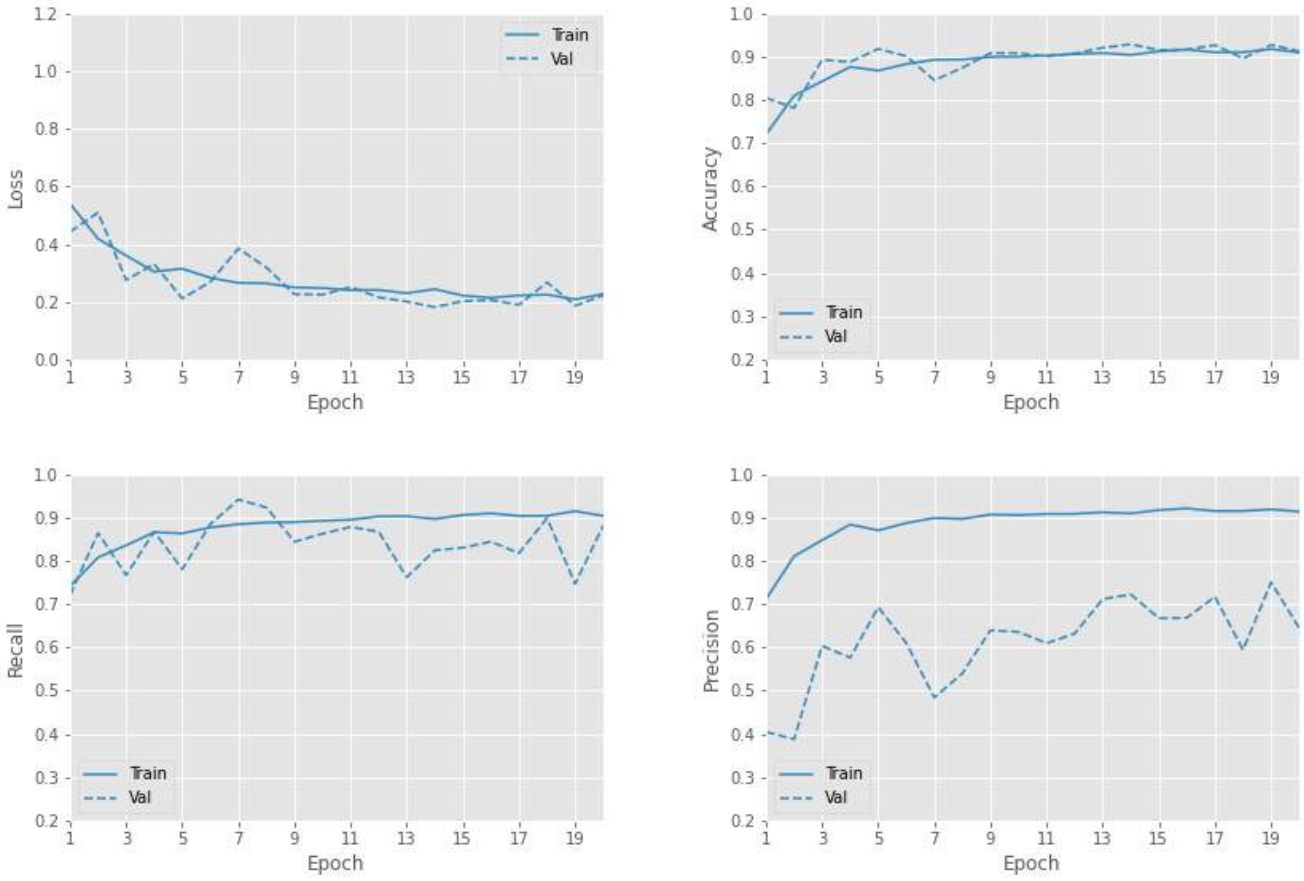


Table 20: Airbus dataset class distributions.

Dataset	Class			Total
	No Ship	One Ship	Multiple Ships	
Full Dataset	150,000	27,104	15,452	192,556
Binary Classification (Training)	5,390	5,390	-	10,780
Binary Classification (Validation)	3,781	647	-	4,428
Binary Classification (Test)	3,756	671	-	4,427
Multi-class Classification (Training)	3,595	3,595	3,595	10,785
Multi-class Classification (Validation)	3,415	649	364	4,428
Multi-class Classification (Test)	3,429	637	362	4,428

Table 21: Details of the Generalised CNN model with Dropout.

Layer	Layer Details
Convolutional Layer	64 Filters, 3x3 Kernel Size, ReLu Activation Function
Max Pooling Layer	2x2 Pooling Size, Stride of 2
Dropout Layer	20% Dropout Rate
Convolutional Layer	64 Filters, 3x3 Kernel Size, ReLu Activation Function
Max Pooling Layer	2x2 Pooling Size, Stride of 2
Dropout Layer	20% Dropout Rate
Convolutional Layer	64 Filters, 3x3 Kernel Size, ReLu Activation Function
Max Pooling Layer	2x2 Pooling Size
Dropout Layer	20% Dropout Rate
Flatten Layer	Flattened Size - 4,096
Fully Connected Dense Layer	Size - 32 Nodes, ReLu Activation Function
Dropout Layer	40% Dropout Rate
Output Layer	Sigmoid Activation Function
Total Trainable Parameters: 280,513	

Figure 18: Suez Canal Ship Predictions - February 21 2021

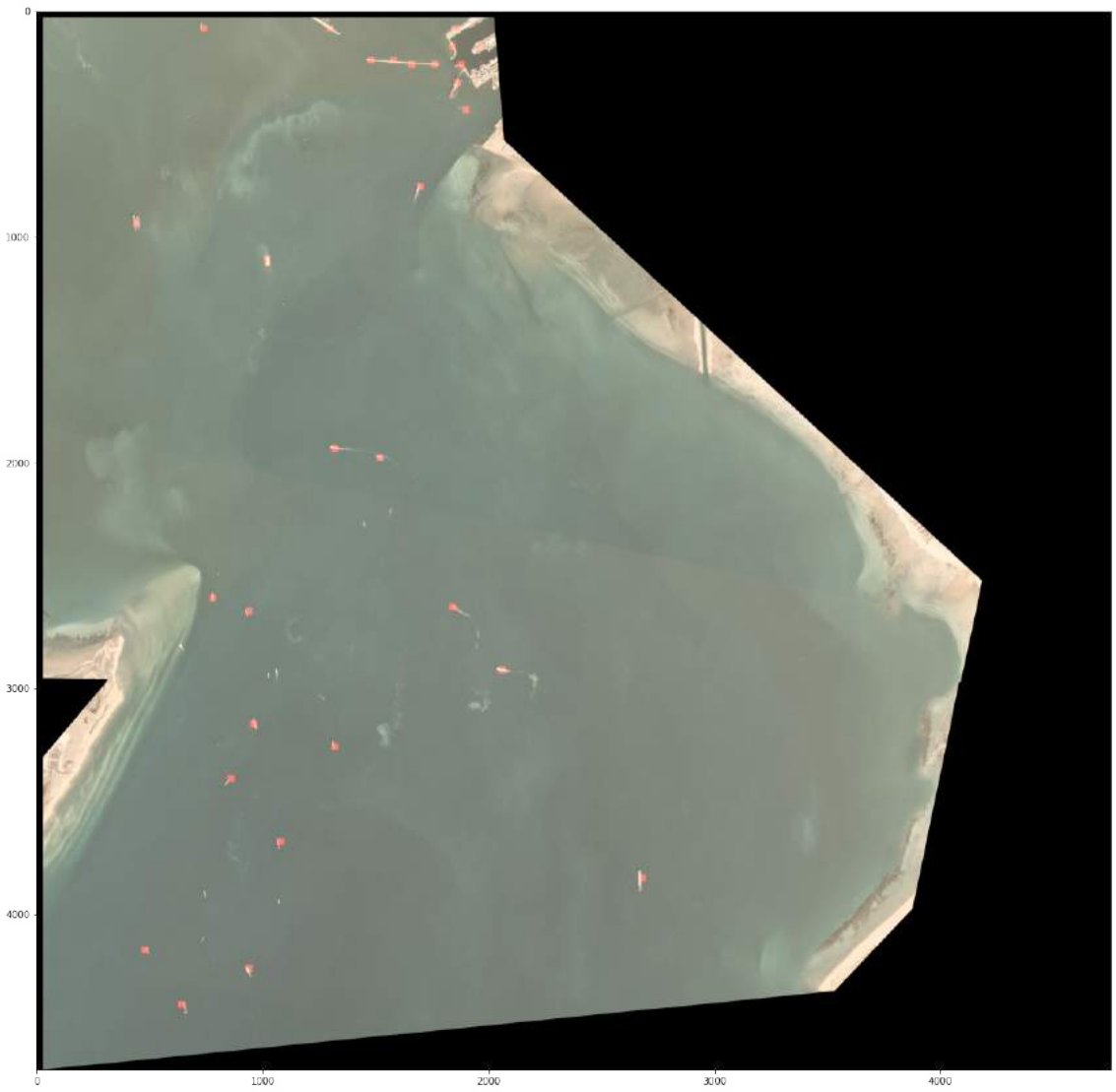


Figure 19: Suez Canal Ship Predictions - March 11 2021

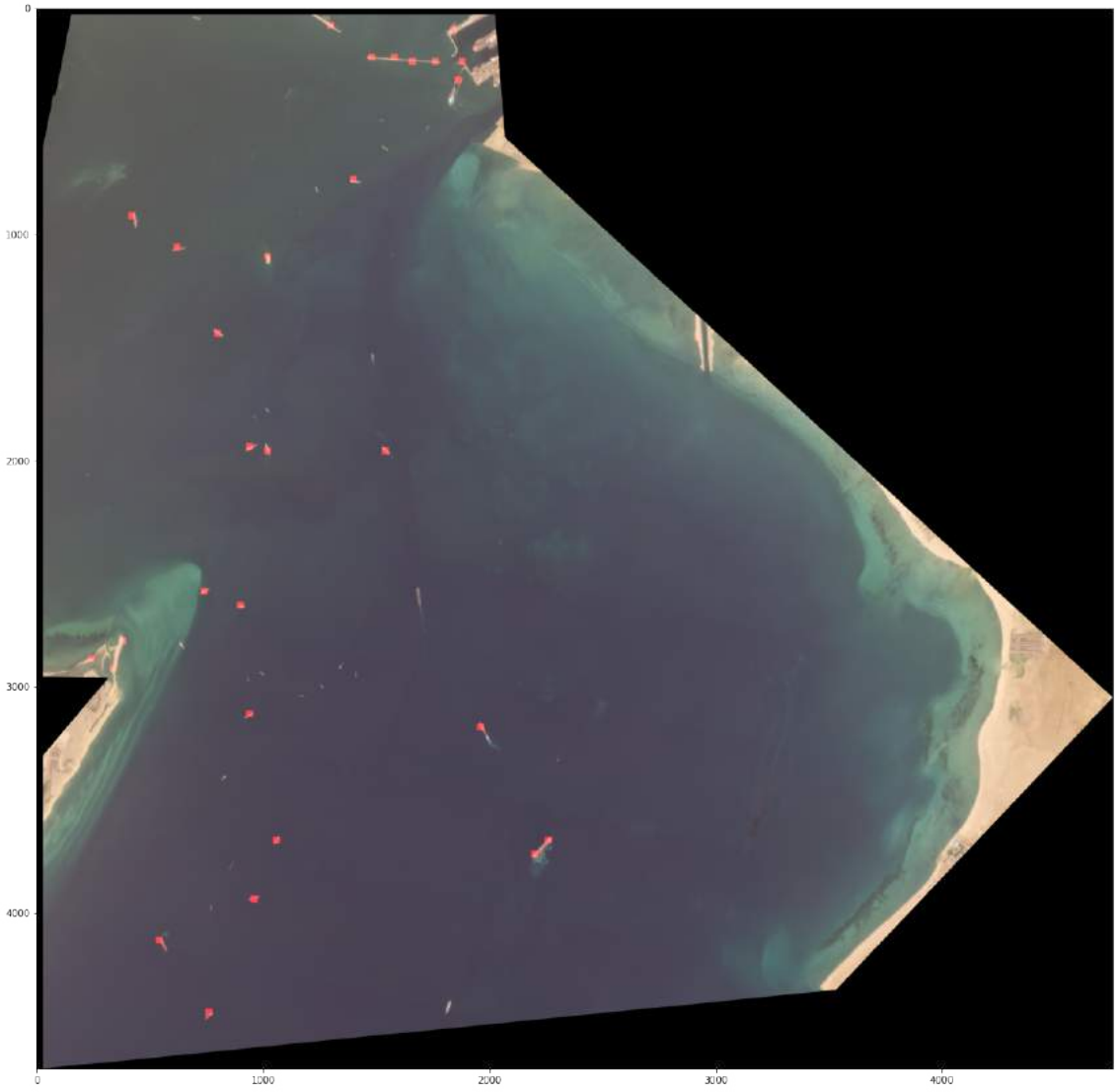
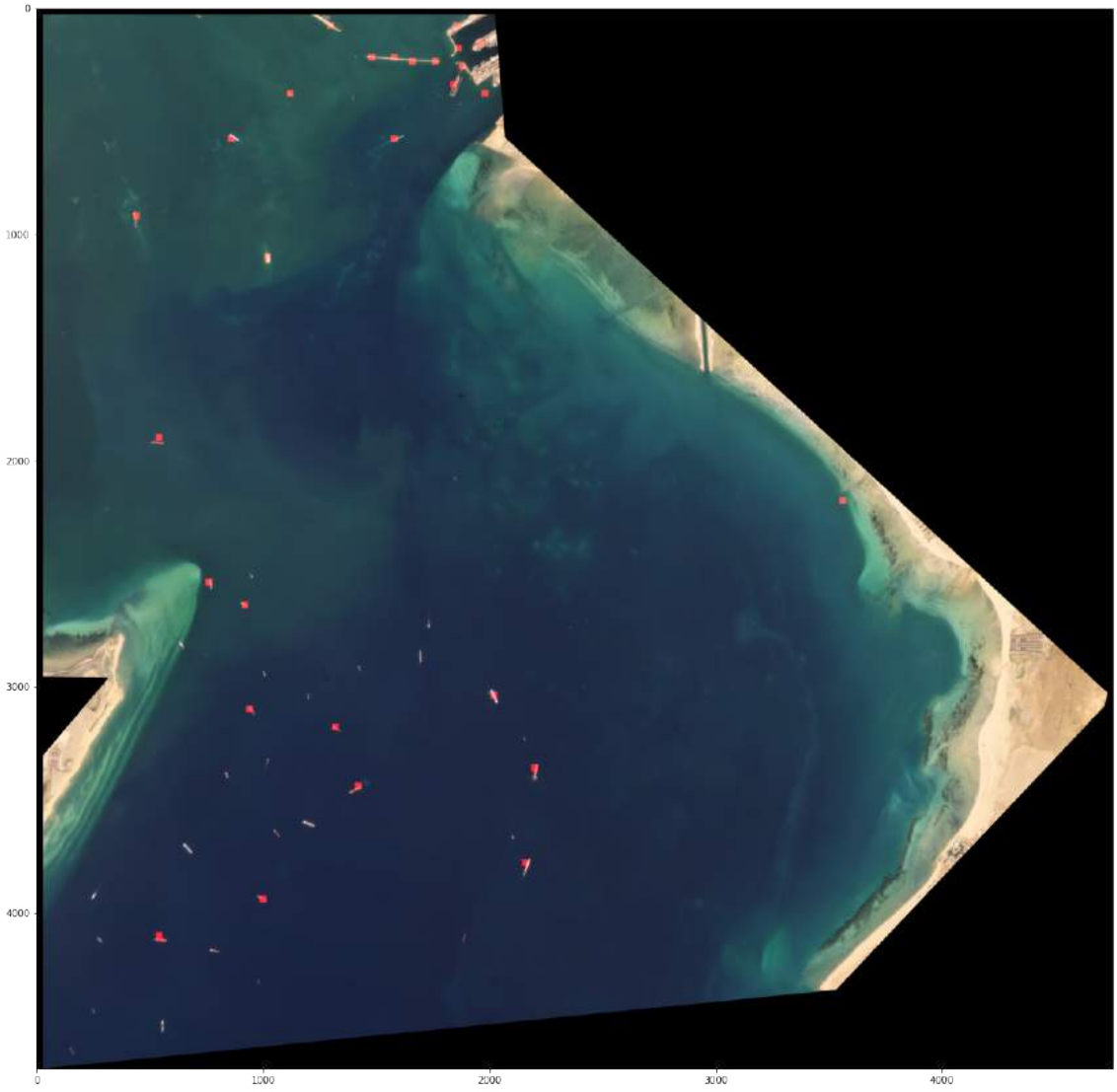


Figure 20: Suez Canal Ship Predictions - March 17 2021



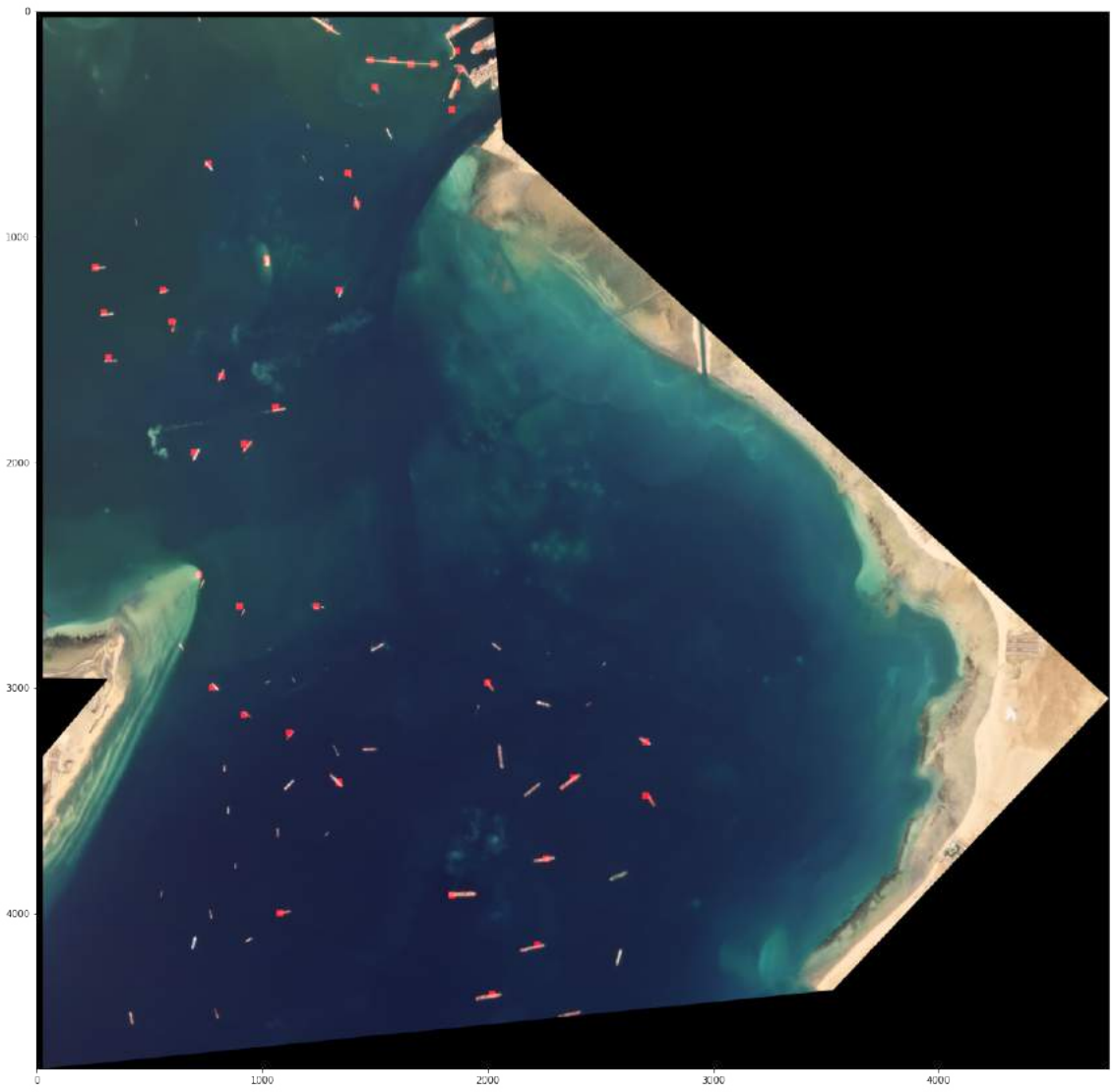


Figure 21: Suez Canal Ship Predictions - April 4 2021