Ship Detection in Remote Sensing Optical Imagery using Machine Learning

Tyler J. Morgan

The University of New South Wales at the Australian Defence Force Academy

The requirements for nations to patrol and protect their maritime borders and international waters is vital to securing economic assets, preventing illegal immigration and monitoring other countries military and commercial behavior. The large increase of military and government surveillance satellites has led to an influx of remote sensing data for which human analysis is time consuming. A recent rise of machine learning techniques such as convolutional neural networks (CNN) has allowed humans to hand over complex tasks like object detection to computer algorithms. This report aimed to conduct optimisation of a hand-built CNN as well as fine-tuning existing state of the art CNN architectures in the task of ship classification in optical remote sensing data. This report was able to achieve a maximum accuracy of 94.8% on a fine-tuned Xception model with no optimisation, outperforming a carefully optimised handmade network by approximately 5%. It was concluded that for this complex classification task an existing state-of-the-art model should be finetuned to achieve the highest accuracy.

Contents
Ship Detection in Remote Sensing Optical Imagery using Machine Learning........................................1
Contents..........................................................................................................................................................1
Nomenclature ...............................................................................................................................................2
I. Introduction ................................................................................................................................................2
   A. Aim......................................................................................................................................................2
   B. Motivation ...........................................................................................................................................2
II. Related Work ..........................................................................................................................................2
III. Methodology ..........................................................................................................................................3
   A. Dataset ...............................................................................................................................................3
      1. Size of dataset ..................................................................................................................................4
      2. Balancing of dataset ..........................................................................................................................4
      3. Size of individual images ..................................................................................................................4
   B. System Hardware.................................................................................................................................4
   C. Programming Environment ................................................................................................................4
   D. Performance Parameters ....................................................................................................................5
      1. Loss Function – Binary crossentropy .............................................................................................5
      2. True Positive Rate (Recall) ..........................................................................................................5
      3. Precision .....................................................................................................................................5
      4. Fβ Score ......................................................................................................................................5
IV. Architecture Optimisation ..................................................................................................................5

1 PLTOFF, School of Engineering & Information Technology. ZEIT4501
V. Hyperparameter Optimisation ................................................................. 6
VI. Data Augmentation and Regularization ............................................... 7
VII. Fine-Tuning High Performance Networks .......................................... 8
VIII. Discussion ....................................................................................... 9
IX. Conclusion ....................................................................................... 10
X. Recommendations ............................................................................... 10
XI. References ....................................................................................... 11

Nomenclature

\( y_t \) = true binary class label for image \( y \) (0 for no ship, 1 for ship)
\( y_p \) = predicted class label for image \( y \) (between 0 and 1)
\( t_p \) = true positive prediction
\( t_n \) = true negative prediction
\( f_p \) = false positive prediction
\( f_n \) = false negative prediction
\( SGD \) = stochastic gradient descent

I. Introduction

A. Aim

This report will outline the optimization of a convolutional neural network (CNN) which will be implemented to conduct generic binary classification of high-resolution satellite imagery. For the purpose of this report the network will be trained primarily on ship detection. The focus of this project is to outline and evaluate different methods of image preprocessing, network architectures and training protocols which result in the high performing binary classifier.

B. Motivation

Defense forces all around the world conduct large scale surveillance and monitoring of their maritime boundaries in an effort to secure their economic assets, prevent illegal immigration and defend their borders. Traditionally this monitoring is conducted by surveillance aircraft and ships which undertake lengthy patrols of the high seas generally only covering even a small percentage of their country’s boundary. The cost of conducting these operations are high and they require a significant amount of military personnel which could otherwise be assigned to other tasks. The use of military satellites has increased greatly in the last decade, especially with major economic and political powers. Fitted with remote sensing equipment these satellites provide a solution for the surveillance of maritime borders, having the ability to cover a much greater area than any of its alternatives. With the vast amount of data being generated from these satellites it is becoming increasingly important to automate the process of object detection due to the speed and accuracy that algorithms can conduct this task. This will not only allow a reallocation of manpower and military assets but also an increased detection rate of vessels in a nation’s region and beyond.

II. Related Work

Classification of ships in optical remote sensing data is a challenging task. This can be attributed to a variety of natural phenomena and features (rain, waves, coral reefs), man-made features (offshore oil platforms, harbors) and the high variability in the size and shape of the vessels being classified. Traditionally with optical remote sensing imagery the methods which have been implemented to detect ships include image thresholding, colour/texture discrimination, histogram of oriented gradient (HOG) and support vector machine (SVM) classification algorithms. With the continued development of machine learning models, especially the advances in very deep multi-class image classification networks, some work has shifted towards implementing these models in ship classification and detection algorithms.

Gallego et al. [1] proposed an CNN optimisation method which involved adjusting hyperparameters on a baseline network to find the best performing set and then using this information to help with the fine-tuning of a high performance existing network. Testing a number of high performance networks they found that Xception performed the highest in the \( F_1 \) metric with VGG-16, Inception and ResNet being close runners up. These networks saw their highest performance after about 20 epochs of training with the Xception getting an \( F_1 \) score...
of 0.997. This significantly outperformed previous ship classification methods tested in this paper with HOG + SVM achieving an F₁ score of 0.792. This approach is supported by methods incorporated the highest performing networks in the Airbus Ship Detection Challenge and as such state-of-the-art networks will be fine-tuned in this report with the consideration of hyperparameters from a baseline model.

Tajbakhsh et al. [2] found that for classification of medical images fine-tuning existing networks matched the performance of full training on the same architectures and even outperformed full training when limited data was available. This points to the usefulness of fine-tuning as not only a method to reduce the training time of networks but also to train high performance classifiers on limited data sets. With a dataset of 4000 images they found that training on even 10% of this dataset showed insignificant performance degradation. It was also found that state of the art performance can be achieved via shallow fine-tuning, which involved freezing the feature extraction layers and only allowing training of the dense classification layers. Despite this it was suggested that networks should be deeply fine-tuned, allowing training of all the network layers albeit with a significantly reduced learning rate when compared to full training. Tajbakhsh indicates in this paper that networks should be ‘deeply’ fine-tuned, allowing feature extraction layers to train at a very low learning rate. This approach will be implemented in this reports for the fine training of existing state-of-the-art networks.

Wang et al. and Ding et al [3, 4] explored the effect data preprocessing had on the performance of CNNs. Data preprocessing is generally implemented as a regularization technique to prevent overfitting and to help the network to generalize. Image transformations such as rotation, translation zoom and randomized mirroring around the horizontal and vertical axis can help randomize the dataset and increase the accuracy of the network on unseen data. Other methods such as adjusting the gamma, brightness and injecting noise or blur can also help with network generalization. Additionally, it was found that image normalization can also increase the accuracy of a network. For this report a number of networks will be trained with different combinations of image pre-processing techniques in order to find which fair best for the dataset.

Wang [3] took a trial and error approach to optimise a hand-made network. This consisted of testing a number of different network configurations, in this case between 2-7 convolutional layers and between 1-5 dense layers and selecting the highest performing combination of the two. In this process there is not a clear relationship between these parameters and the accuracy of the resulting network, this is shown in Fig. 1. This result indicates that during testing a number of network architectures should be trialed to find the highest performing network.

![Figure 1 - How changing the number of fully connected and convolutional layers effects network accuracy](image)

This trial and error approach will be taken in all architecture and hyperparameter fine-tuning as it is simple to conduct and there is little established literature on other methods of CNN optimisation. One other method will be trialed which will be by fine-tuning NASNetMobile [5] which optimises its architecture automatically during training.

### III. Methodology

#### A. Dataset

This project took advantage of the dataset provided by Airbus [6] which contains approximately 200,000 labelled optical satellite images of ships. This data is in the form of 3-channel RGB with a pixel resolution of 768 x 768 and contains over 20Gb of information. Due to its large size it was unrealistic to optimise a CNN by taking advantage of all available information therefore some data size reduction had to be implemented.
1. Size of dataset
   Training on more unique data will always result in a higher performing classifier but it is often preferable to be able to build a classifier which can achieve good accuracy on limited data; this not only decreases the time to train the network but also addresses a common problem in machine learning which is data sparsity.

   The current baseline for data required to adequately train a CNN from the ground up is 1000 images per class. This baseline is in large part due to the development of the ImageNet Competition [7] and the great success had with datasets of this size. With the focus of reducing training time but preserving network integrity a healthy medium between the total available amount of data and the minimum baseline was chosen to train on, this is outlined in Tab. 1.

<table>
<thead>
<tr>
<th>Image Label</th>
<th>Number of Images in each Set</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Training Set</td>
<td>Test Set</td>
<td>Validation Set</td>
</tr>
<tr>
<td>Ship</td>
<td>5,000</td>
<td>2,000</td>
<td>1,000</td>
</tr>
<tr>
<td>No Ship</td>
<td>5,000</td>
<td>2,000</td>
<td>1,000</td>
</tr>
</tbody>
</table>

   Due to the excess amount of data, 5000 images of each class will also be set aside for an independent model evaluation to allow a more accurate comparison of the networks.

2. Balancing of dataset
   Due to the large dataset the classes for this problem are able to be balanced, although this is often not the case. When training a network on a highly unbalanced data the user must be very careful in picking performance parameters and the loss function. To visualise this imagine if the dataset contained 90% images of class 1 and 10% images of class 0. If the network predicts all images as class 1 it would be evaluated to have an accuracy of 90%.

3. Size of individual images
   To further reduce the amount of information that the network is required to process the images themselves were resized for the prototyping phase, after some preliminary testing it was discovered that for this dataset a validation set accuracy and loss could be achieved on 250 x 250 size images which approximated these performance parameters on higher resolution data. Therefore, these sized images were used during the rapid prototyping phase due to the speed of their processing. A similar conclusion was also reached by Gallego et al.[1] who found the highest performance from 200 x 200 pixel images. It should be noted that some of the networks that were fine-tuned natively take an input of 256 x 256 or 299 x 299. From initial testing this change in resolution should not make a significant change to the detection accuracy of these algorithms, still allowing a fair comparison.

B. System Hardware
   To allow future reference of network performance, especially the time to train and classify, all work completed in this report was done so on the hardware found in Tab. 2 on a 64-bit version of Windows 10 Home operating system. None of this hardware was overclocked or changed in any way during this testing.

<table>
<thead>
<tr>
<th>CPU</th>
<th>Intel i7 8700k</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPU</td>
<td>NVIDIA GeForce GTX 1080</td>
</tr>
<tr>
<td>RAM</td>
<td>16GB DDR4</td>
</tr>
</tbody>
</table>

C. Programming Environment
   Python being one if not the most popular programming languages for development of machine learning solutions was the chosen language for this project due to its wide range of machine learning libraries and large community providing support and tutorials. The Python environment was created using the Anaconda distribution which provides simple package/environment management, it is also the easiest way to use GPU computing while training machine learning models in a Windows OS. The machine learning library used was Google’s TensorFlow [8] underneath the Keras high-level API [9]. This programming environment allowed the rapid prototyping of over one hundred different network configurations.
D. Performance Parameters

To be able to measure and compare the performance of a number of CNN models some performance parameters must be defined.

1. Loss Function – Binary crossentropy

Binary crossentropy is a loss function commonly used to evaluate the performance of a binary classification network during training. In this experiment the loss function has been chosen to be the performance parameter in which is to be optimized during training, this was chosen over the other common performance factor, accuracy, as it provides more information than just the percentage of predictions which were correct.

Binary crossentropy, like other loss functions, provides are score which is logarithmically scaled to how close the prediction was to a correct prediction. It is calculated using Eq. 1.

$$\text{loss} = -\left( y_t \log(y_p) + (1 - y_t) \log(1 - y_p) \right)$$  

(1)

2. True Positive Rate (Recall)

Recall is a measure of what percentage of ships were identified as ships, this is useful as we can see what percentage of ships are not identified by the network. This parameter also has a shortcoming, this is because if a network predicts all images as having a ship it will have a recall of 100%, this will be addressed shortly. Recall can be calculated using Eq. 2.

$$\text{True Positive Rate (TPR)} = \frac{tp}{tp + fn}$$  

(2)

3. Precision

Precision is what percentage of images which were predicted to have ships contain ships. This performance is useful when the number of false positives is desired to be reduced. Precision can be calculated using Eq. 3.

$$\text{Precision} = \frac{tp}{tp + fn}$$  

(3)

$$\text{False Positive Rate (FPR)} = 1 – \text{Precision}$$  

(4)

4. $F_\beta$ Score

$F_\beta$ Score is the main validation performance parameter which will be used to determine the overall performance of each trained model. $F_\beta$ score is a weighted average of precision and recall and will provide the means to quantitatively measure. F Score is weighted using the value beta which can be used to dictate the weight of recall to precision upon the final answer, for example $F_2$ Score gives twice the weighting to recall, $F_1$ score evenly weights precision and recall and $F_{0.5}$ score gives double the weighting to precision. $F_\beta$ score can be calculated using Eq. 5.

$$F_\beta = (1 + \beta^2) \frac{\text{precision} \times \text{TPR}}{(\beta^2 \times \text{precision}) + \text{TPR}}$$  

(5)

IV. Architecture Optimisation

<table>
<thead>
<tr>
<th>Example</th>
<th>Table 3 – LeNet Style Network</th>
</tr>
</thead>
<tbody>
<tr>
<td>Layer</td>
<td>Type</td>
</tr>
<tr>
<td>1</td>
<td>Conv1</td>
</tr>
<tr>
<td>2</td>
<td>MaxPool1</td>
</tr>
<tr>
<td>3</td>
<td>Conv2</td>
</tr>
<tr>
<td>4</td>
<td>MaxPool2</td>
</tr>
<tr>
<td>5</td>
<td>Conv3</td>
</tr>
<tr>
<td>6</td>
<td>MaxPool3</td>
</tr>
<tr>
<td>7</td>
<td>Dense1</td>
</tr>
<tr>
<td>8</td>
<td>Dense2</td>
</tr>
</tbody>
</table>

When designing a network from scratch one of the most important factors when attempting to achieve a high-performance model is to find a suitable architecture for the application. As there are an infinite number of network configurations a common technique is to choose a baseline architecture which is commonly used and has proven to be effective. The network configuration chosen for this report is like that of LeNet which implements a very commonly used CNN structure. This structure consists of stacking a number of convolution layer and pooling layer pairs which then flatten and feed into some number of dense layers. Table 3 gives an example of this network structure.

During testing 30 different network styles were tested. The networks tested were for all combinations of 2-7 convolution layers, 1-3 dense layers, and RGB/Grayscale input. The highest performing of all these trials was an RGB network with 6 convolution layers and 3 dense layers, the structure for which is shown in Tab. 4.
Table 4 – Highest Performing Network from Architecture Optimisation

<table>
<thead>
<tr>
<th>Layer</th>
<th>Type</th>
<th>Output Size</th>
<th>Activation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Input Data</td>
<td>$250 \times 250 \times 3$</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Conv_1 (3 × 3)</td>
<td>$250 \times 250 \times 16$</td>
<td>Relu</td>
</tr>
<tr>
<td>3</td>
<td>MaxPool_1 (2 × 2)</td>
<td>$125 \times 125 \times 16$</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Conv_2 (3 × 3)</td>
<td>$125 \times 125 \times 32$</td>
<td>Relu</td>
</tr>
<tr>
<td>5</td>
<td>MaxPool_2 (2 × 2)</td>
<td>$62 \times 62 \times 32$</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Conv_3 (3 × 3)</td>
<td>$62 \times 62 \times 64$</td>
<td>Relu</td>
</tr>
<tr>
<td>7</td>
<td>MaxPool_3 (2 × 2)</td>
<td>$31 \times 31 \times 64$</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Conv_4 (5 × 5)</td>
<td>$31 \times 31 \times 128$</td>
<td>Relu</td>
</tr>
<tr>
<td>9</td>
<td>MaxPool_4 (2 × 2)</td>
<td>$15 \times 15 \times 128$</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Conv_5 (7 × 7)</td>
<td>$15 \times 15 \times 256$</td>
<td>Relu</td>
</tr>
<tr>
<td>11</td>
<td>MaxPool_5 (2 × 2)</td>
<td>$7 \times 7 \times 256$</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Conv_6 (7 × 7)</td>
<td>$7 \times 7 \times 512$</td>
<td>Relu</td>
</tr>
<tr>
<td>13</td>
<td>MaxPool_6 (2 × 2)</td>
<td>$3 \times 3 \times 512$</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Dense_1</td>
<td>2048</td>
<td>Sigmoid</td>
</tr>
<tr>
<td>15</td>
<td>Dense_2</td>
<td>512</td>
<td>Sigmoid</td>
</tr>
<tr>
<td>16</td>
<td>Dense_3</td>
<td>512</td>
<td>Sigmoid</td>
</tr>
<tr>
<td>17</td>
<td>Output Layer</td>
<td>1</td>
<td>Sigmoid</td>
</tr>
</tbody>
</table>

Figure 2 - $F_p$ Scores of highest performing RGB and Grayscale networks from architecture optimisation

V. Hyperparameter Optimisation

Hyperparameter optimisation is one of the hardest evaluations to undertake as hyperparameters are not optimised automatically (e.g. via gradient descent) but are instead tested through batch or manual searching. For this reason, it is often computationally expensive to conduct optimisation for a CNN such as our highest performing RGB network. With a large amount of time and computational power the network could be trained thousands of times. The highest performing RGB network was taken from the architecture optimisation and was the model being ‘tuned’ during this phase of testing. The parameters that were adjusted during optimisation and
their highest performing values are shown in Tab.5 while this tuned models performance is compared to its predecessor model and the highest performing grayscale network in Fig. 3.

It should be noted that this test is in no way holistic but were the best results from the batch testing of approximately 60 different CNNs. A good option to avoid the manual selection of hyperparameters in the future would be to take advantage of Google Cloud’s Machine Learning Engine [10] which can conduct automatic hyperparameter tuning using cloud computing. Although the user must pay for GPU time this would be a much more effective way of optimising the networks hyperparameters for accuracy or loss.

**Table 5 - Highest Performing Hyperparameter Settings**

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Best Performing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convolutional Layer Activation Function</td>
<td>ReLu</td>
</tr>
<tr>
<td>Dense Layer Activation Function</td>
<td>Sigmoid</td>
</tr>
<tr>
<td>Convolution Layer Window Size</td>
<td>3 × 3</td>
</tr>
<tr>
<td>Max Pooling Size</td>
<td>2 × 2</td>
</tr>
<tr>
<td>Dense Layer Size</td>
<td>512 nodes for all dense layers</td>
</tr>
<tr>
<td>Optimiser</td>
<td>SGD (Learning Rate = 0.02, Momentum = 0.5)</td>
</tr>
<tr>
<td>Convolution Filters</td>
<td>Double for each convolutional layer [16,32,128,256,512,etc...]</td>
</tr>
<tr>
<td>Padding</td>
<td>“same”</td>
</tr>
<tr>
<td>Stride</td>
<td>0</td>
</tr>
</tbody>
</table>

**Figure 3 - F Scores of highest performing Optimised RGB compared to the highest performing RGB and Gray network from architecture optimisation**

**VI. Data Augmentation and Regularization**

Although the training set is large, data augmentation and regularisation techniques were implemented to evaluate the increase in network performance. Data augmentation was done through the Keras preprocessing function which provides a number of data augmentation options. Dropout regularisation was also tested and implemented to help generalise the network even further and saw promising results. This testing was conducted on the highest performing model from hyperparameter optimisation, the ‘Tuned RGB’, and its performance against this and previous models are shown in Fig. 4. The highest performing data augmentation and regularisation techniques found for this dataset are covered in Tab. 6 and these were discovered via batch testing.
### Table 6 - Highest performing data augmentation and regularisation settings

<table>
<thead>
<tr>
<th>High Performing Data Augmentation and Regularisation Techniques</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random Horizontal Flip</td>
<td>-</td>
</tr>
<tr>
<td>Random Vertical Flip</td>
<td>-</td>
</tr>
<tr>
<td>Random Rotation 90 degrees</td>
<td></td>
</tr>
<tr>
<td>Feature-wise Std Normalisation</td>
<td>-</td>
</tr>
<tr>
<td>ZCA whitening</td>
<td>-</td>
</tr>
<tr>
<td>Shear Range 5 degrees</td>
<td></td>
</tr>
<tr>
<td>Dropout Layers (between each set of dense layers)</td>
<td>50% Dropout</td>
</tr>
</tbody>
</table>

**Figure 4** - F Scores of highest performing network from data augmentation and regularisation optimisation compared to highest performing networks from previous tests.

#### VII. Fine-Tuning High Performance Networks

Due to the complexity of designing and training a network from the ground up, today's generally accepted approach to achieving a very high performing classifier is to take advantage of the thousands of man hours that have gone into designing and training some of the highest performing public networks available such as ResNet [11]. For this test the classification layers architecture of the highest performing handmade network was used in combination with the pre-trained feature extraction layers of the state-of-the-art models. This can drastically reduce the time to train and provide very high performance when compared to an optimised hand-made network. The networks which were selected for fine-tuning were the highest performing open-source ImageNet networks from the past few years of the competition. The networks chosen were Xception [12], ResNet50 [11], NasMobileNet [5] and Inception ResNet v2 [13]. Due to the large time to train these networks their hyperparameters and classification layers were unable to be tuned through iteration like the handmade network and as such these results in no way reflect the highest possible performance of these networks on this dataset but rather the highest achieved performance from two individual training protocols. The performance of these four classifiers against the highest performance handmade classifier, ‘Augmented Image Tuned RGB’, is shown in Fig. 5.
VIII. Discussion

From the above testing it can be seen that even the most refined handmade network achieved was exceeded by a fine-tuned Xception network which was trained on default settings, with a large amount of GPU hours this network could be optimised to achieve even higher accuracy with reports of up to 99% from winning teams of the Airbus Ship Detection Competition [6]. On the authors system this model could make predictions on over 220 images per second, which is suitable for real time estimation. Due to the large amount of time that goes into designing and training these state-of-the-art networks it is recommended that for complex classification tasks such as this these networks are taken full advantage of as opposed to attempting to optimise a hand-made network.

The best performing network across all F beta scores from these tests was Xception and as such some further analysis has been done on its performance as a classifier, especially in cases where it failed. The network showed great accuracy at detecting large ships both moving and stationary even when they were moored alongside.

The network sometimes fails at detecting ships which have the following characteristics:
- Very small or small and stationary
- Obstructed by clouds
- Ships that have unique shapes
- In images that are noisy/overexposed

Some cases of where the network failed to properly identify a ship are shown in Fig. 6.

![Figure 5 - F Scores of highest performing handmade network compared to four state-of-the-art models fine-tuned on the dataset.](image)

![Figure 6 - Examples of ships that have been classified as non-ships by the highest performing network](image)

Many of these problems could be resolved with fine-tuning the network on a dataset from a single sensor with correctly labelled classes. Another explanation for these mistakes is the high diversity and incorrect labels in the dataset. The dataset the network is trained on is not perfect and after the networks were trained the training set was shown to have portion of mislabeled images, resulting in

Despite these mispredictions the network did generalise well on most of the unseen data being able to achieve an accuracy of 95% which is quite impressive given the network did not undergo hyperparameter or image augmentation fine tuning like the hand-made network.
Generally, the network detected a ship if the image had one/more the following characteristics:
- Moving ship
- Large ship
- Image not under/over exposed
- Minimal environmental anomalies
- Minimal image noise/artifacts

Some examples of correct predictions by the network are shown in Fig. 7.

Figure 7 - Examples of ships that have been classified as ships by the highest performing network

Optimising the highest performing network for 3 cases, high precision, high accuracy and high recall by changing the decision threshold and testing the performance of 10,000 unseen images some baseline values for the model’s performance can be calculated. These values are presented in Tab. 7.

<table>
<thead>
<tr>
<th>Mode</th>
<th>Maximising</th>
<th>( F_\beta ) Score</th>
<th>Decision Threshold</th>
<th>tp</th>
<th>tn</th>
<th>fp</th>
<th>tn</th>
<th>Accuracy</th>
<th>Recall</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>High Precision</td>
<td>( F_{0.5} ) Score</td>
<td>0.954</td>
<td>0.94</td>
<td>4552</td>
<td>4832</td>
<td>165</td>
<td>448</td>
<td>93.87%</td>
<td>91.04%</td>
<td>96.50%</td>
</tr>
<tr>
<td>High Accuracy</td>
<td>( F_1 ) Score</td>
<td>0.948</td>
<td>0.73</td>
<td>4770</td>
<td>4710</td>
<td>290</td>
<td>230</td>
<td>94.80%</td>
<td>95.40%</td>
<td>94.27%</td>
</tr>
<tr>
<td>High Recall</td>
<td>( F_2 ) Score</td>
<td>0.962</td>
<td>0.03</td>
<td>4904</td>
<td>4407</td>
<td>593</td>
<td>96</td>
<td>93.11%</td>
<td>98.08%</td>
<td>89.21%</td>
</tr>
</tbody>
</table>

IX. Conclusion

In conclusion a number of high performing ship classification networks were developed with the highest performing of which being a fine-tuned Xception network which achieved an accuracy of approximately 95%. The performance of the un-refined Xception compared to a carefully optimised handmade network reinforces that if that computational power and time is available, a designer should focus on optimising one of the many publically available state-of-the-art networks rather than attempting to build one from the ground up. Overall the accuracy of the highest performing model did not match some of the very impressive performance in the literature, but this can be attributed in part to the highly diverse dataset and to the lack of further optimisation which with more time should be undertaken.

X. Recommendations

For future continuation of this project several recommendations have been made by the author:
- If not familiar with machine learning the author recommends taking the free online course run by Andrew Ng from Stanford University, this can be found on Coursea [14].
- The hyperparameter optimisation of state-of-the-art models such as the Xception model.
- Fine-tuning of the highest performing network in this paper on a dataset generated by the sensor in which the classifier will be implemented on, this dataset should contain greater than 500 images with at least 200 containing ships in varied conditions.
- Conduct further training via cloud computing or on a local multiple GPU machine to greatly increase speed prototyping speed.
- If another dataset can not be obtained, the model should be finetuned on a handpicked subsection of the Airbus dataset which has been ensured is labelled correctly and does not contain images with unnatural artifacts or other corruption.
With access to cloud computing NASNetLarge should be attempted to be fine-tuned, this was unable to be done on the authors machine due to using >12GB GPU VRAM on even very small batch sizes, this network shows very good promise in the literature.

XI. References