VICTORIA UNIVERSITY OF WELLINGTON
Te Whare Wānanga o te Ūpoko o te Ika a Māui

School of Engineering and Computer Science
Te Kura Mātai Pūkaha, Pūrorohiko

PO Box 600
Wellington
New Zealand

Tel: +64 4 463 5341
Fax: +64 4 463 5045
Internet: office@ecs.vuw.ac.nz

Automatic Assessment of Image Quality from At-Sea Monitoring Systems

Matt Rothwell

Supervisors:
Andrew Lensen
Finlay Thompson (Industry)

Submitted in partial fulfilment of the requirements for
Bachelor of Engineering with Honours.

Abstract

This project explores if machine learning tools can be used to accurately identify issues with remote imaging systems on board fishing vessels. With an increasing numbers of cameras being used at sea, issues with degraded image quality pose challenges for reviewers. Footage is collected with the goal of replacing the traditional role of sea-going fisheries observers, which would lower costs. Fisheries observers collect data aboard commercial fishing vessels about fish being caught, efforts to catch fish and environmental interactions (eg, catching protected species). However, when images are of poor quality, it makes this task harder or impossible in some cases. This project implemented and evaluated a machine learning system that can detect and classify image quality issues from snapshots taken aboard fishing vessels. Despite the effects of a small and unbalanced dataset, classifiers created achieved up to 79.6% accuracy on unseen data. Further work incorporating additional data would allow better performance overall.

Contents

Chapter 1

Introduction

The use of video monitoring systems on fishing vessels is becoming increasingly common, as it poses an alternative to having a fisheries observer on board at all times for law enforcement. Fisheries observers report on fish caught, efforts to catch fish and other environmental interactions during their time aboard commercial fishing vessels [2]. The commercial fishing vessels equipped with camera systems studied are operating in, and around the Hauraki Gulf of Auckland. Of particular interest to researchers at Dragonfly Data Science, the industry stakeholder for this project, is the critically endangered Petrel. Fishing vessels on occasion have been responsible for the deaths of these birds, often becoming caught in lines being set. Thus, it is important to monitor fishing boats for interactions involving these birds.

Figure 1.1: A grey faced petrel [1].

As a result of the harsh marine environment, cameras on these fishing vessels can become obscured and dirty quickly, decreasing the quality of images collected. This can prevent reviewers that view the footage from making accurate observations, thus defeating the purpose of the cameras. Image quality from the cameras can become degraded as a result of sea spray and other water, glare, condensation, lack of light, obstructions in front of the lens or even failure of the camera completely. Currently, image quality analysis requires a human checking the footage manually, or simply inspecting the camera lens on-board the vessel. This can mean that cameras can go for long periods of time recording degraded quality or completely unusable footage. Thus, it would be beneficial to automate the assessment of image quality with these camera systems, allowing quality issues to be resolved faster.

1.1 Goals

The overall goal of this project is to create an image classification system capable of automatically labelling the quality of images collected from fishing vessels in an accurate manner. Fulfilling a number of objectives will aid this goal being achieved:

1. Perform exploratory data analysis to gain an understanding of the data, including trends and abnormalities.
2. Analyse and select feature extraction methods beneficial for classification.
3. Design, implement and refine classifiers required.
4. Perform thorough evaluation of classifiers produced with regard to overall goal.

Chapter 2

Background and Related Work

2.1 Problem Description

Machine learning tools allow the production of self-improving algorithms that improve over time without being explicitly programmed [3]. We aim to produce a system that can evaluate the quality of the images automatically, with high accuracy when compared with the labels given to data by human reviewers, through the use of machine learning.

As the vessels are at sea where cellular internet coverage is patchy, the footage is recorded to a device on the vessel. When cellular internet is available, individual still images or 'snapshots' are transmitted at regular intervals back to researchers at Dragonfly Data Science. These individual snapshots can then be assessed, labelled, and used for research purposes.

An automated system assessing these snapshots would allow faster identification of image quality issues with cameras. For example, obstructions, dirt and water on the lens, and other failures could be detected. This allows the crew to be notified to resolve the issue, therefore maintaining footage quality for reviewers.

Image classification utilizing machine learning algorithms is the standard way of tackling this type of problem [4]. While there are many different algorithms and tools that could be used, it is beneficial to understand the data we have available for use first. This allows early identification of patterns, interesting relations and any anomalies within the data.

2.1.1 Data

Data in this project is sensitive in nature because it includes images of people, vessels and locations that aren't intended to exist within the public domain. Therefore, in accordance with the data access agreement signed (which can be found in the appendix), imagery shown in this report is limited to images where people, vessel names and other identifying features are not shown or redacted.

For each image within the dataset, there are 5 labels we need to predict to assess image quality. This will allow the labelling of snapshots in a similar manner to human reviewers. These are reviewability, clarity, light level, activity and position.

Overall, with regard to the problem posed by this project, reviewability is the most important label for classification. It is the overarching measure of whether or not action needs to be taken to correct image quality problems with a camera.

Reviewability

A measure of how useful the snapshot is for making observations. This attribute has several self-describing, ordinal values that describe the reviewability: 'unusable', 'poor', 'acceptable', 'good'.

Clarity

Clarity is a self-describing, categorical attribute that describes the current condition of the camera's visibility with regard to transparency. Possible values are 'obscured', 'dirty', 'condensation', 'splashed' and 'good'.

Light Level

Light level is a categorical attribute that describes the lighting conditions present within the image. This is an important attribute that can influence the overall reviewability of an image. Possible values include:

- 'day' for when the lighting conditions appear to be during daylight hours.
- 'glare' for when there is significant glare within the image, which can degrade quality.
- 'deck lights' for when the image is illuminated by lights onboard the vessel.
- 'dark' for when the lighting conditions appear to be at night without deck lighting.

Activity

Activity is a categorical attribute that describes the current operation the ship is undertaking. Possible values include:

- 'port' for when the vessel is berthed in port.
- 'steaming' for when the vessel is underway under engine power.
- 'setting' for when lines with baited hooks are being set at sea.
- 'hauling' for when the lines are being hauled aboard the vessel.
- 'anchor' for when the ship is anchored at sea.
- 'other' for any other miscellaneous activity.

Position

Position is a boolean attribute that describes whether the camera is deployed into the operational position above the side of the vessel, or in the stowed position.

2.2 Related Work

This project is essentially an image classification problem, which is widely documented through academic literature. Thus, before beginning design and implementation work, I first conducted a brief literature review. This looked at solutions for detecting objects within images, similar to what I believed was needed for this project.

2.2.1 Droplet Detection

As the vessels within the dataset are at sea, droplets of water are common on cameras. In a paper exploring the identification and removal of droplets from images taken by cameras on mining equipment [5], a Mask Attention Network was used to detect the characteristics of water droplets from their surroundings. This could be used to generate an attention map to detect the droplets so they could generate a new image without the droplets. This network design could be useful for the identification of droplets within the images within the context of this project. It may also be possible to use a similar approach for detecting dirt on the lens.

Cameras in cars are becoming especially common with modern cars incorporating camera-based safety systems and autonomous driving systems. There are a number of pieces of research that look into algorithms for detecting rain droplets in images taken from an on-board camera. These systems have been implemented using Convolutional Neural Networks (CNN) with varying success [6][7]. The techniques used in these papers may be useful to aid droplet detection within the context of this project.

2.2.2 Glare Detection

Within the images collected, quality is severely impacted by glare. This is extremely common because the water is extremely reflective and the cameras are mounted off the side of the boat often directly above the water. Luckily in existing literature, glare and extremely bright spots are easily detected with tools with a high confidence. Machine learning tools will likely not be required for low-level detection, however, with the use of ML tools, this process can be made faster. Researchers found that it was possible to detect glare in images with a simple Naive Bayes algorithm in Tensorflow, trained on luminance features within document images with high accuracy [8].

Another study compared classical image processing techniques and Convolutional Neural Networks for detecting glare within images from automotive cameras [9]. It found a classical method worked at a similar rate to a CNN solution on some examples but noted their CNN needed further evolution as it frequently incorrectly detects sun glare in a few pixels. However, they do consider their CNN model a baseline and discuss combining it with some classical techniques to make a more accurate algorithm.

2.3 Evaluating Classifiers

For evaluation, the dataset of images will be split into a test set and a training set. This project will use 30% of the data for the test set, and 70% for the training set. This allows the model to be trained on most of the dataset to produce a generalised solution, while still ensuring there is a significant amount of unseen data for testing of the model. This aids the identification of models overfitted to the training set. Overfitting occurs when a model becomes overly tuned to deliver good results upon a specific dataset. This means that real-world performance on unseen data will be poor, as the model is not generalised enough.

There are a number of different metrics that will be used for evaluating models produced for the system:

- Accuracy, which is the portion of correct classifications produced by the model upon a test set.

- Precision, which is a measure of what predictions for a certain class, are actually correct.
- Recall, which is the portion of expected instances that are successfully classified.
- F-score, which is the weighted average between precision and recall. This will be particularly useful as it is more useful than overall accuracy for unbalanced datasets.

Through the combined analysis of these metrics, we can evaluate the performance of the overall system and understand whether its fit for purpose.

Chapter 3

Design

3.1 Exploratory Data Analysis

Exploratory Data Analysis (EDA) is a statistical process of performing preliminary investigations into a data set to find patterns, features, test theories and check assumptions made [10].

As of writing, the dataset supplied by Dragonfly Data Science has 163 labelled, and 1117 unlabelled instances. The data consists of CSV files listing the image file name, and its labels (if labelled). This dataset is being actively collected and is continually getting larger as more snapshots are transmitted back from active vessels. The data is being collected as part of an ongoing study into whether cameras can replace the traditional role of seagoing fisheries observers on commercial fishing vessels.

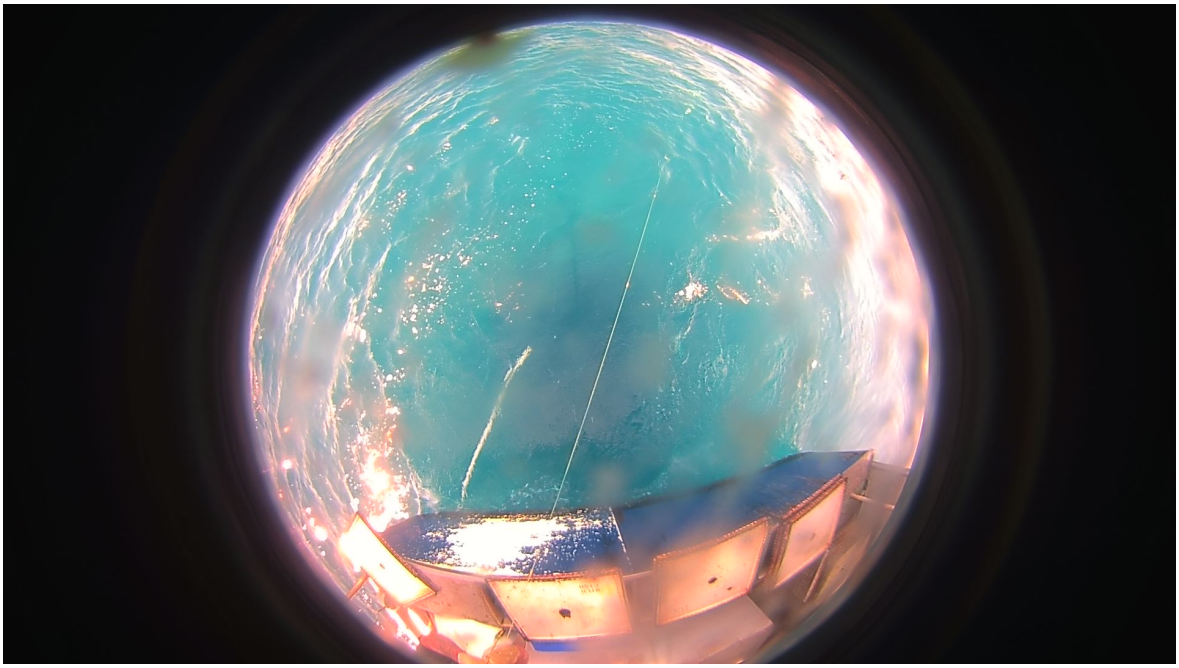


Figure 3.1: An example image snapshot from the dataset.

Image snapshots are in 1920 x 1080 resolution, but they are taken through a fish-eye lens and therefore heavily distorted as seen in Figure 3.1. As a result of the distortion, a large amount of definition is lost around the edges of the image due to vignetting. While the distorted data will be impossible to rebuild given the lossy nature of images, it may be beneficial to manipulate the images to best remove the distortion before analysis.

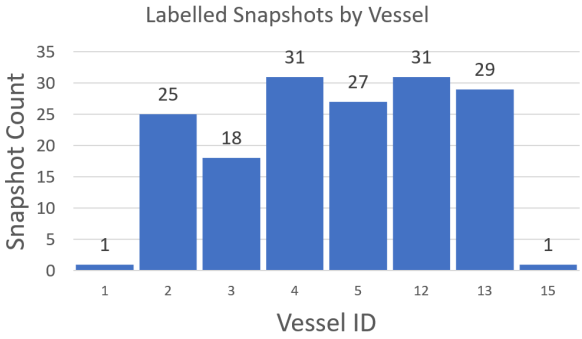


Figure 3.2: The amount of labelled snapshots within the dataset by vessel id.

Currently, the dataset has a good spread of labelled data from the vessels being monitored, with the majority having 18-31 snapshots each. However, two outlier vessels only have one labelled snapshot each, as seen in Figure 3.2. It would be a reason for concern if the majority of the snapshots were from one vessel. However, as there is a good distribution of data between the remaining 6 vessels, overfitting of a model to a specific vessel should not be an issue.

From initial impressions of the data, It would appear that the reviewability attribute is dependant on the other attributes, such as light level and clarity. When light levels are darker, and clarity is anything other than 'good', reviewability becomes worse. Therefore, the logical focus becomes extracting features relevant to light level and clarity, as opposed to directly trying to classify reviewability from the images. Once a model can predict light level and clarity with high accuracy, it will enable another model to use them are predictors for estimating overall reviewability. Reviewability is more statistically likely to be 'good' or 'acceptable', as most of the instances in the dataset fall into these groups. This is shown in Figure 3.3.

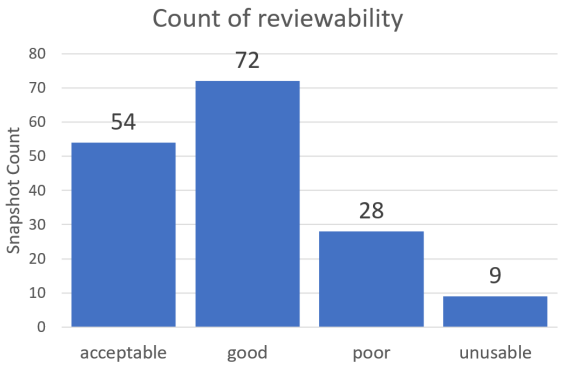


Figure 3.3: Snapshot count by reviewability within the dataset.

The amount of labelled data is of concern, as we may need a larger dataset to produce a highly accurate model. Models trained to recognise rain on vehicle windscreens, as discussed when conducting background research, often required hundreds of thousands of images during training to achieve a high accuracy [6][7].

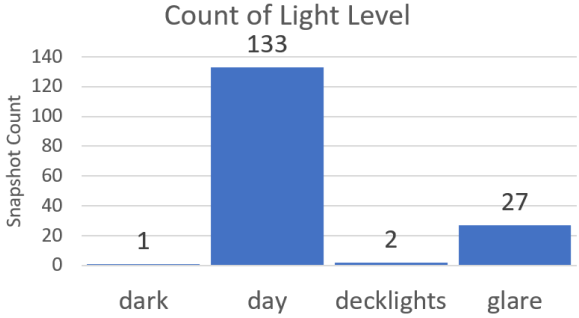


Figure 3.4: Snapshot count by light level within the dataset.

Also of concern is the class distribution for 'light level', where there are very few in-

stances that are 'dark' or 'deck lights' in the labelled data, as seen in Figure 3.4. This means that a model tuned for this data could be biased towards classifying images as 'day', but it may appear highly accurate, as the majority of a test set may be at 'day'. If this were to occur, it is highly likely that more labelled data would be required. This would involve manually labelling some of the 1117 unlabelled instances. While the industry stakeholder at Dragonfly Data Science may be able to do this, it would likely take time.

A similar pattern of class imbalance can be seen for other features, where one specific class tends to be more common for a feature than the others.

As seen in Figure 3.5 for 'clarity', classes 'dirty' and 'good' are dominant compared to the other classes. While this may be statistically representative of the real-world distribution of image clarity, it may be difficult for us to produce an unbiased model.

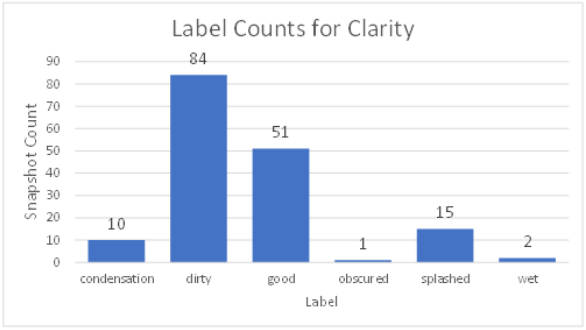
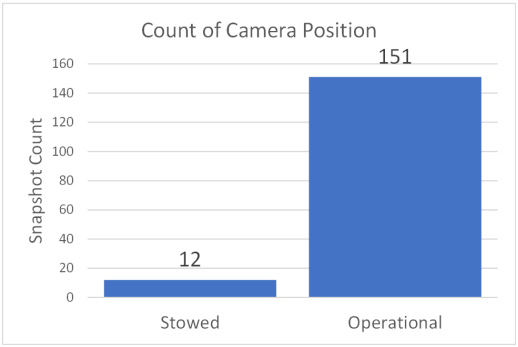


Figure 3.5: Snapshot count by clarity within the dataset.

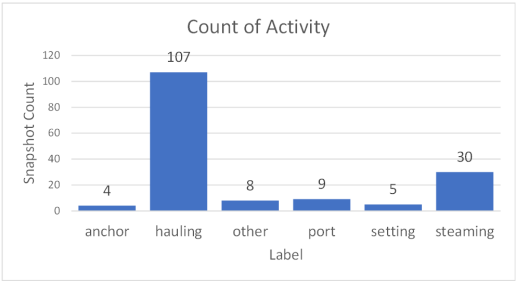
Also of concern, seen in Figure 3.6a is the 'position' feature, which only contains 12 instances of the camera in the 'stowed' position, compared to the 151 'operational' position instances.

Another example of unbalanced classes is the 'Activity' feature, where the vast majority of instances are 'hauling', with only 'steaming' the only other label exceeding 9 examples within the dataset. This can be seen in Figure 3.6b.

Class balancing may also be an option, however, this may prove difficult as the maximum amount of images within the dataset with a value for some attributes is one. Therefore, all classes would have to have one image to be balanced, which is very unlikely to produce a useful model.



(a) Snapshot count by camera position within the dataset.



(b) Snapshot count by activity within the dataset.

Figure 3.6

3.2 Classifier Selection

As discussed in Section 3.1, the amount of labelled data available and the generally unbalanced class distribution increase the difficulty of this task significantly. During a discussion with the industry stakeholders at Dragonfly Data Science, they expressed interest in pursuing deep learning with convolutional neural networks, using established tools like Tensorflow. Unfortunately, this requires a significant amount more labelled data than the 163 instances available for this project, with studies stating around 1,000 images per class can be considered a small dataset [11].

As a result of these factors, the classifier to be used was chosen to be a Random Forest (RF) classifier. The RF is capable of handling high dimensional data easily in multi-class classification problems [12], which is important because images contain large amounts of data. RF classifiers also have robust methods for dealing with noise in the data and class imbalance when compared to other classifiers like Support Vector Machines (SVM) [13]. As the dataset for this project suffers from class imbalance, this will be beneficial to our use case.

Scikit-learn is a machine learning library that offers a wide range of features, including an implementation of the RF Classifier [14] and other useful functions for handling data and evaluating classifiers. Using the Scikit-learn library will save time instead of reproducing many machine learning algorithms from scratch for use in this project. This will likely be integrated with other methods of extracting useful information from the images. The process of feature extraction and some useful tools for this is explained further below.

3.3 Feature Extraction

Feature extraction is the process of decreasing the dimensionality of data, grouping specific relevant information useful to a problem [15]. This grouped data can then be used by a machine learning model for training and making predictions. Performing feature extraction upon images is extremely common and existing literature provides some insight into relevant tools and techniques that may apply to this project. There are several different features we wish to be able to extract from the images assessed by the system. These include water droplets on the lens, dirt and salt, glare and condensation. We also wish to be able to detect the lighting conditions of the image.

3.3.1 Feature Extraction Tools

There are several existing machine learning libraries and tools that exist for the purpose of image processing. Using one or more of these will speed up the modelling process and will allow us to build a system faster. They can provide standardised algorithms that can act as building blocks for our system instead of implementing our own from scratch.

Scikit-image

Scikit-image is an open-source collection of algorithms for image processing in the Python programming language [16]. It provides an implementation of many different image manipulation modules, including filtering, geometric transformations and edge finding, as well as feature detection.

Tensorflow

Tensorflow by Google also provides a computer vision framework for object detection [17]. Overall however the options for further computer vision development in Tensorflow are rather limited due to its nature as a generic neural network framework.

OpenCV

OpenCV (Open Source Computer Vision Library) is another already existing library that provides a vast collection of machine learning and computer vision algorithms [18]. The general consensus from online literature is that OpenCV is a more versatile and feature-complete library than competitors. OpenCV also has interfaces for C++, Java, MATLAB and Python. It is however limited to two-dimensional imagery [16], which is not an issue for this project.

Summary

Using one or more of these libraries will allow us to iterate on a project-specific implementation at a faster rate than if we were writing these algorithms from scratch repeatedly. OpenCV appears to offer the best set of computer vision algorithms to start out with basic feature extraction. It also offers a much more advanced feature set for computer vision problems if we require them for an advanced project-specific implementation. It also appears to be the industry standard for machine learning systems that involve computer vision. Scikit-image also provides a useful toolkit for extracting features from images and has plenty of available documentation. Each library has some tools that the other does not, but if required, both OpenCV and Scikit-image are interoperable.

3.4 Narrowing of Scope

Through the data analysis process, it became apparent that several of the labels would be extremely difficult to produce a model for. These are the 'activity' and the camera 'position' labels. This is largely due to their class imbalance within the dataset. Therefore it would make sense to exclude these, especially as 'activity' does not contribute to the goal of this project, which is analysing the **quality** of images collected. As discussed with the industry stakeholders, this information would be nice to have automatically labelled, but not necessary.

Therefore, the labels we will attempt to predict will be limited to:

- Reviewability
- Clarity
- Light Level

3.5 Reporting and Presentation of Results

The industry stakeholders requested an easy to understand report that would be automatically generated after the code was run. Ideally, this would have statistics on the classifier being trained, its accuracy on a test set, and other information about the data in general. This will also generate useful graphics that can be included in this report when evaluating the model(s) produced.

3.6 System Architecture

Due to the industry stakeholders' data privacy concerns, bulk image data was not to be stored locally on devices. Data is stored in an Amazon Web Services (AWS) s3 storage bucket, and can be downloaded into a Docker image running on Kahawai, an internal system used by Dragonfly Data Science that runs software on AWS instances. As shown in Figure 3.7 below, when git commits are made to the project git repository, a Docker image runs the contents of the git repository created by executing a shell script build file.

This instance can be configured with varying amounts of memory, CPU's and GPU's if required. By design, this meant that no compute was done locally, and no image data was put at risk as it never left Dragonfly Data Science's internal environment.

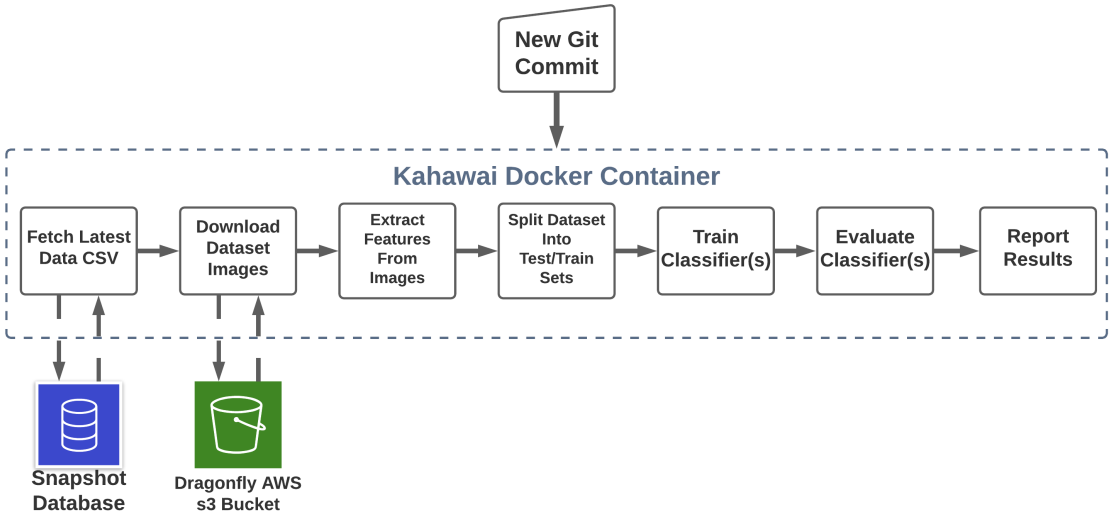


Figure 3.7: Simplified outline of how the pipeline should function when operational.

Python was chosen for the creation of this system, as it is compatible with most of the tools identified in Section 3.3.1. Python is also widely used in an image classification context, and an existing knowledge base is extremely useful for efficiently solving issues.

A standard pipeline design will be used for the architecture of the system being created. As seen in Figure 3.7, data will first be retrieved and stored. Then feature extraction can be performed, which allows the retrieval of useful information from the images within the dataset. Following this, the data will be split into a training set and a test set. The training set will be used for training the classifier(s) and the test set will be used for testing the accuracy of the classifier(s), which can then be used when reporting results for analysis.

Chapter 4

Implementation

The implementation of a system was delayed significantly due to several unforeseen factors. The reasons for these delays were explained in the preliminary report for this project, which can be found in the appendix. As mentioned in Chapter 3, our industry stakeholders required all work to run internally on their systems to minimise the risk of data leakage and mitigate legal concerns. Thus, a significant portion of time was consumed learning how their Kahawai system operates and solving issues that arose.

4.1 Kahawai Integration

As previously mentioned, Kahawai is a platform used internally at Dragonfly Data Science that can be described as an extension to a git repository, that allows execution similar to a CI/CD pipeline. Within a git project that integrates with Kahawai, there is a build script that executes after each commit. This can be used to run any code you have in the repository, output results, and produce other reporting.

The biggest challenge using Kahawai was the learning curve and its limitations. It ultimately slowed down the development of the initial system for quite some time. Numerous compatibility issues were encountered with outdated software and ultimately OpenCV and Tensorflow would not work (At this stage, the use of convolutional neural networks had not been ruled out). After a significant amount of time debugging, Scikit-image and Scikit-learn were installed and functioned as expected when a Docker image was run by Kahawai, allowing work to move on to fetching data. Scikit-image was not the first choice in feature extraction tools, but it has enough variety that it will work well enough for this project.

4.2 Gathering Data

Every time the program is run, multiple CSV data files are fetched. These contain information about each image, like metadata including date of capture, labels and data needed to fetch it from the AWS s3 bucket. A script was created that downloads all the images in the labelled dataset to the Docker image. It also produces a merged CSV file containing all the information and metadata that may be needed for creating a classification model, like the labels for each image, and where they are stored locally to the instance.

4.3 Early Implementations

The very first implementation created was very basic and only classified 'reviewability'. No feature extraction was performed upon the images. They were converted to greyscale, and their two-dimensional array form was flattened to become one large one dimension array per image. A RF classifier was then trained using a 30% test set, 70% training set split.

This performed poorly, only getting 50% accuracy upon the test set. It also gained F-scores of 48% for 'acceptable' images, 58% for 'good' images, and 25% for 'poor' images. While this is still statistically better than picking the majority class, it's not fit for purpose with regard to the overall goal of this project. 'Good' had a better F-score and captured most of the actual images labelled with 'Good', it's also the majority class, as seen in Section 3.1.

Following this, merging some of the 'reviewability' classes into two classes was tested, simply 'good' and 'bad'. The idea being this difference was an acceptable compromise for simplicity's sake, as essentially 'bad' was what we wanted to detect. This would simplify the classifier and increase the amount of data for each of the individual classes. Having different levels of 'bad' and 'good' added unnecessary complexity to the classifier as they were needlessly specific.

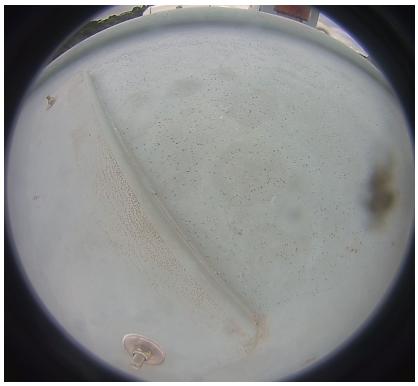


Figure 4.1: An example of an obscured, but good quality image.

Issues with this idea arose as the dataset contained images that were of good quality, but as a result of the camera lens being physically obstructed, were classed as 'unusable' by reviewers. An example of an image like this is seen in Figure 4.1. Thus, the classifier would incorrectly label these images as 'good', as we can assume it would think they are of decent quality. They should be classed as 'bad' or 'unusable' if using the standard labels. This is because reviewers are unable to make observations as they would expect from the cameras in 'good' conditions.

This simplified 'good'/'bad' model achieved an overall accuracy of 76% upon the test set. However, it was biased towards picking 'good' over 'bad'. Recall and F-score for 'bad' was also poor, at 18% and 27% respectively. This was unlike 'good', where recall was 94%, meaning it correctly classified nearly all of the images that were 'good'.

The most valuable information learned from these initial models is that producing multiple classifiers to inform reviewability was likely to benefit accuracy. The complexity of the problem is not as simple as it seems, as shown by the instances described earlier, where the image is of good quality, but obscured, making it unusable for reviewers.

4.4 Evolving the Implementation

Taking the learnings from the implementation described above, the system evolved into three different classifiers. These utilised methods like feature extraction to gain more useful and specific data from the images to improve our overall performance. A few different

forms of feature extraction were tested upon the dataset of images, for each of the various classifiers we aim to create and tune.

In addition to this, during pre-processing, all images were cropped vertically to remove excess dark pixels. These pixels provided no useful information as this area was almost identical visually across all images. This was done to reduce data dimensionality. It also had the bonus effect of slightly decreasing the amount of resources required for feature extraction and training.

4.4.1 Clarity

For predicting the clarity label for images, algorithmic feature extraction methods were used. The use of image segmentation was also considered. Image segmentation is the process of splitting an image into multiple segments, typically used for locating objects and their edges in an image [19]. This was decided against because the items we wish to detect aren't contiguous and often have elements of transparency.

Dirt, condensation, and water typically distort or blur images as they are closer to the camera than their minimum focal distance. Thus, they can be stated to be creating patterns and distortion upon a scene in an image, rather than appearing as a physical object. Therefore, image segmentation would be extremely challenging because it is difficult to place hard boundaries on where these clarity-effecting features appear. Because of this, several different methods of feature extraction were trialled for detecting this distortion. These are detailed below.

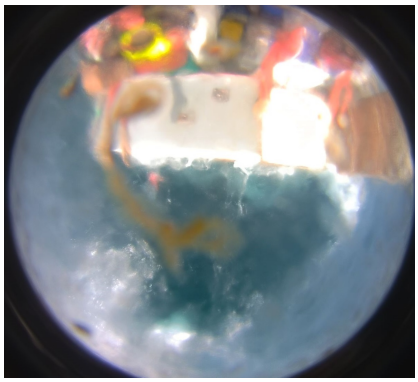


Figure 4.2: A 'dirty' snapshot that appears wet.

Removal of Smaller Classes

Due to the lack of data for several classes within clarity, and their very specific nature, several classes were excluded from the clarity model. Class distribution for clarity can be seen in Figure 3.5. Some snapshots also contained visual elements that could place them in two separate classes within clarity. The snapshot shown in Figure 4.2 is labelled as 'dirty', however, it could be labelled as both 'dirty' and 'wet'. If we included 'wet' and other smaller classes in the dataset for this classifier, the model would struggle given the small amount of data available. Unfortunately, this decreases the model's effectiveness towards achieving the goals of this project, but it is required until more labelled data becomes available. The classes 'dirty' and 'good' both had usable amounts of data, so were left in the model.

Feature Extraction Methods

First, edge detectors were tested upon the dataset. Edge detection algorithms use mathematical methods for identifying visual edges within images [20]. Images with degraded quality tend to have a weaker definition of edges, thus these features are likely useful to this classifier. The Roberts, Sobel and Canny edge detectors were trialled. An example of an image with the Canny edge detector applied is seen in Figure 4.3. All edge detectors had good accuracy overall, but poor results at the class level, deeming them unsuitable. The model

created with the Canny edge detector had the highest accuracy of the edge detectors tested, gaining 73% accuracy overall. However, this was because most of the instances predicted were of class 'dirty', which made up the majority of the dataset. The recall score for 'good' was only 13%. Precision for 'dirty' was also poor, further indicating the heavy bias of the model.

Second, the use of Local Binary Patterns (LBP) [21] for feature extraction was trialled. LBP algorithms can be used to produce a descriptor for texture by performing thresholding operations on neighbouring pixels in an image. This allows for the extraction of useful texture information relating to dirt and other objects on the lens in images. LBP was implemented using a radius of 1 and 8 points surrounding the central point, and the resulting information was used to train the classifier. This achieved an overall accuracy of 63%, but it predicted all but two instances as 'dirty'. This resulted in 'good' having a recall score of 12%. This poor result showed that the parameters for LBP needed tuning or LBP needed to be used in combination with other feature extractors like Histogram of Oriented Gradients (HOG) [22].

Figure 4.3: A snapshot with the Canny edge detector applied.

HOG takes localised portions of an image and counts occurrences of gradient orientation within that portion. It is well used in machine learning for object detection. HOG can be described as a way of reducing the dimensionality of the data while still retaining a large amount of the information needed for a classifier. As the images in the dataset were high resolution, HOG was set to use 16x16 pixels in each cell as this fitted the cropped image resolution without excess. 8 orientations were also used within each cell when examining localised gradient. An example of how a portion of an image would be represented using HOG is shown in Figure 4.4 below.

Performance

When feature vectors produced by HOG are included in our image classification pipeline for clarity, HOG produced some interesting results. The classifier achieved 59% overall accuracy, with a recall score for 'good' instances achieving 30%. This was better than what was achieved with LBP. As shown in Figure 4.5, 16 images were misclassified as 'dirty' instead of 'good', thus overall precision for 'dirty' was only 52%. F-scores for 'dirty' and 'good' were 67% and 45% respectively.

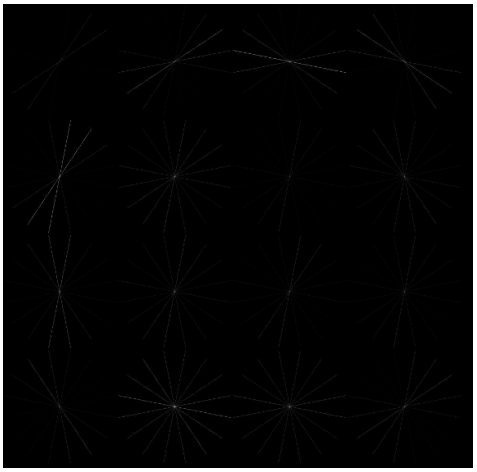


Figure 4.4: How a portion of a snapshot is represented with HOG applied.

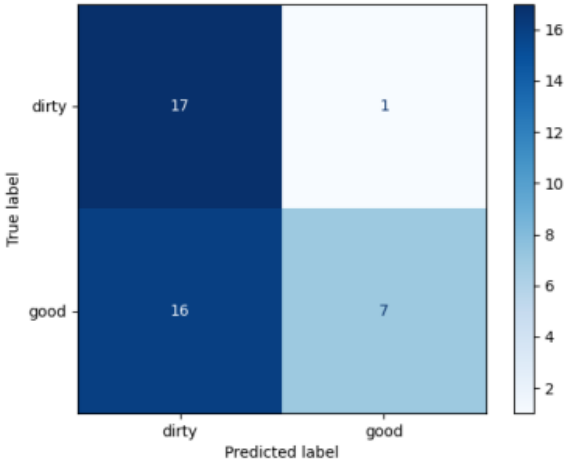


Figure 4.5: Confusion matrix for clarity classifier using HOG.

While this is the classifier with the lowest overall accuracy compared to other methods of feature extraction, it provided the best accuracy when we look at the classification rate of the minority class. However, it did tend to bias towards predicting 'dirty' which was the most popular class in the dataset. An example of an incorrect classification is seen in Figure 4.6. This snapshot was classified as 'dirty' but its actual label is 'good'. As we only have 135 instances, after the removal of smaller classes, producing a classifier is extremely difficult, especially given the high dimensionality of the data that is attempting to be used. Extracting better features may not offer any improvement to the model if there is simply not enough data to allow appropriate tuning.

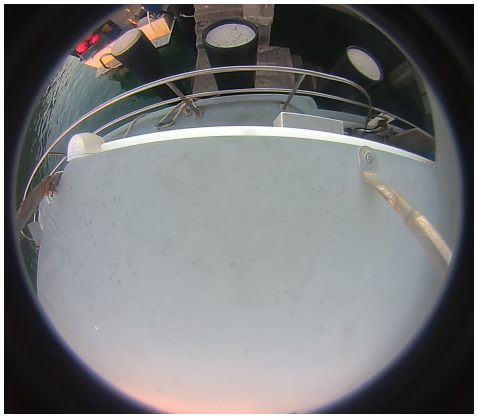


Figure 4.6: A 'good' snapshot that was predicted to be 'dirty' by the classifier.

4.4.2 Light Level

For predicting the light level label for images, statistical feature extraction of colour information was used as opposed to extracting image features using algorithms described above in Section 4.4.1. This was chosen because light controls the contrast and brightness of an image, which influences how other objects will appear, but it is not a physical object or recognisable pattern.

The system extracted the following statistics features about each image:

- The proportion and amount of very dark pixels (Greyscale pixel value < 30) in the image.
- The proportion and amount of very bright pixels (Greyscale pixel value > 230) in an image.
- The proportion and amount of pixels not overly bright or dark (midrange values, between 30 and 230).
- The average brightness of the image.

- A value representing the amount of colour in an image, as described in [23].

These statistics are designed to represent the main characteristics of the four types of light level stated in Section 2.1.1. The amount of bright pixels is useful for detecting glare in snapshots, as these will have lots of extremely bright pixels. Similarly, the average brightness and amount of dark pixels can inform whether or not the image was taken in dark conditions. Differentiating between the various light levels is possible when we look at the images calculated statistical values as mentioned above. As light decreases, the amount of colour captured in snapshots also decreases as there is less physical light to reflect off coloured surfaces. Images at night had little colour and generally contained many dark pixels. Thus it was beneficial to introduce a simple measure of colour along with the other statistics. The colour score was implemented to represent this statistically.

To calculate the colour score, the simple opponent colour model is used [24]. This colour model is derived from how human vision interprets colour by processing signals from cone and rod cells in the eye [25]. Three components make up the opponent colour space, the luminance component, the red-green channel, and the blue-yellow channel. First, the red, green and blue channels of an RGB image are separated into relevant 2d arrays, to be known as R, G and B respectively. Opponent colour theory is used to calculate rg, the red-green component, and yb, the yellow-blue component of the image, using the formulas below [23].

$$rg = R - G$$

$$yb = \frac{1}{2}(R + G) - B$$

From these values, the standard deviation and mean are calculated using the formulas below [23].

$$\sigma_{rgyb} = \sqrt{\sigma_{rg}^2 + \sigma_{yb}^2}$$

$$\mu_{rgyb} = \sqrt{\mu_{rg}^2 + \mu_{yb}^2}$$

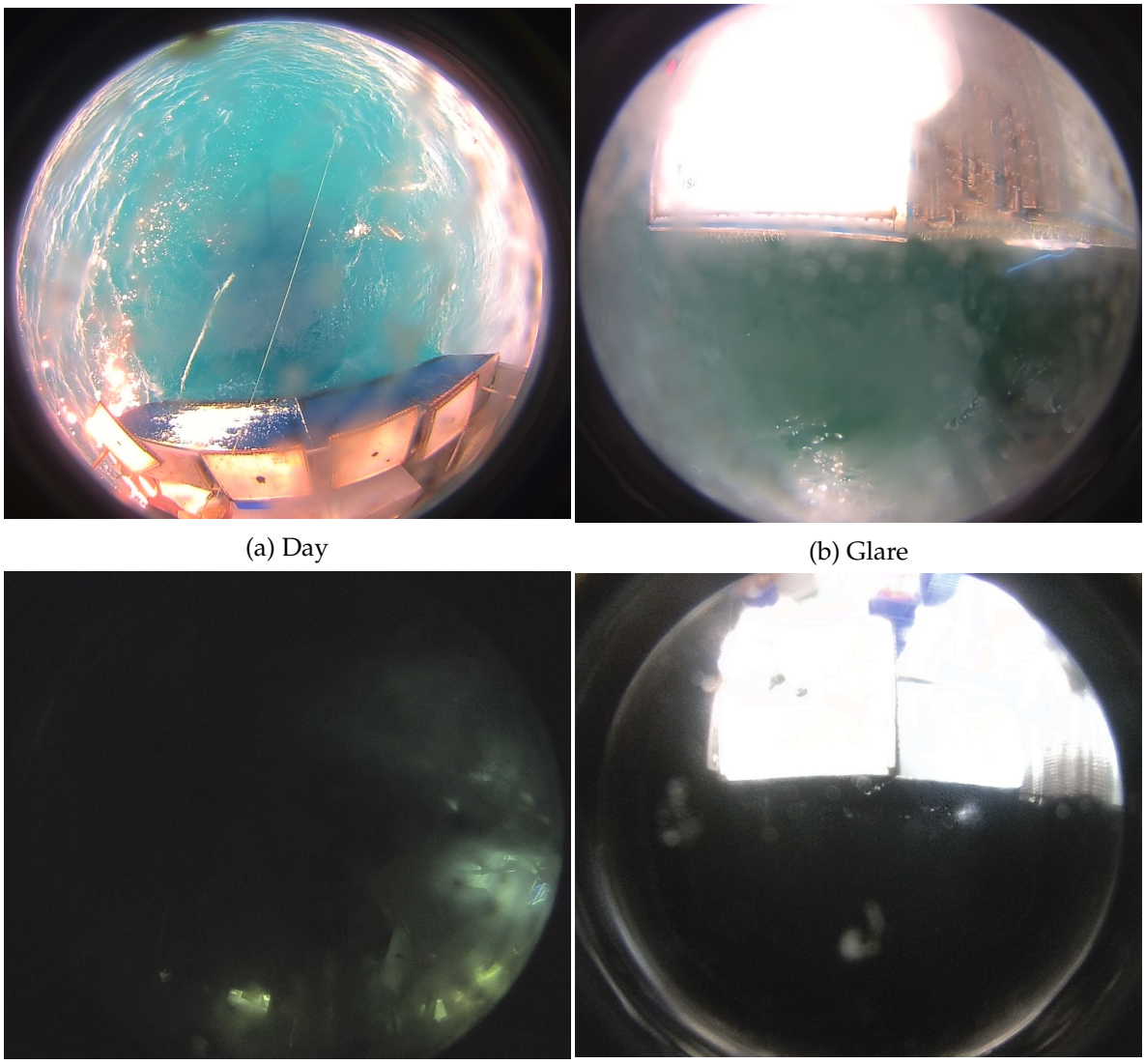
Then the colour score, C, is calculated using the formula below [23]. This gives us an overall colour score for the image.

$$C = \sigma_{rgyb} + 0.3 * \mu_{rgyb}$$

This statistical method of feature extraction can be seen in the images in Figure 4.7 and the related statistics featured in Table 4.1. There is a clear split observable in the data, especially concerning average pixel brightness and the colour score. The 'day' instance in Figure 4.7a can also be observed to have high average pixel brightness and a high colour score. The 'glare' instance in Figure 4.7b also has a high average pixel brightness, high amount of bright pixels, but a lower colour score as more of the image is dominated by glare. 'dark' instances like Figure 4.7c tend to feature higher proportions of darker pixels and lower proportions of brighter pixels. They also had lower average brightness and colour score. The extreme contrast between the darkness and the illumination from the vessel lighting in 'deck lights' instance Figure 4.7d is also observable as it has a high amount of both bright pixels and dark pixels. In addition, the average brightness value remained high and the colour score was low.

Feature	Fig 4.7a	Fig 4.7b	Fig 4.7c	Fig 4.7d
Average Pixel Brightness	88.8	85.1	32.8	71.4
Dark Pixel %	45.9	45	71.6	47.9
Bright Pixel %	4.5	11.1	0	15.2
Mid Range %	49.6	44	28.4	36.9
Colour Score	44.5	14.8	6.2	6.6

Table 4.1: A selection of statistical features extracted from images in Figure 4.7.



(a) Day

(b) Glare

(c) Dark

(d) Deck Lights

Figure 4.7: Examples of captured images with varying light levels.

Performance

Upon the test set, the light level classifier achieved 79.6% accuracy. It was not able to successfully classify either of the 'deck lights' or 'dark' instances as no 'dark' instances, and only one 'deck lights' instance was in the training set. Thus, 'deck lights' and 'dark' has precision, recall and F-scores of 0. Of the classes that have been predicted as day, 84% of them

are correct. Recall is extremely good for 'day' instances achieving 95% accuracy. F-scores for 'day' and 'glare' is 59% and 40% respectively. Only 30% of the 'glare' instances in the test set were successfully classified as 'glare', the rest were classified as 'day', as seen in the confusion matrix shown in Figure 4.8. This would indicate that the model is biased towards labelling images as 'day' at this stage, as 'day' is the majority class, and the dataset is not very large.

There were two instances of 'day' incorrectly classified as 'glare'. One instance, shown in Figure 4.9, does include a significant amount of glare. Upon further inspection, this instance may be an example of incorrect labelling, and the classifier is correct. Understandably, instances of 'deck lights' and 'dark' were not correctly predicted by the classifier. To improve the accuracy for these two classes, more data is required such that the model can learn their characteristics appropriately.

Interestingly, during a separate trial where the test and training set was composed differently, the model was successfully able to classify the 'dark' image as 'deck lights'. The training set, in this case, contained the instance seen in Figure 4.7d, it then classified the instance seen in Figure 4.7c as 'deck lights' during testing. This was indicative that the statistical feature extraction being performed was working well for this use case.

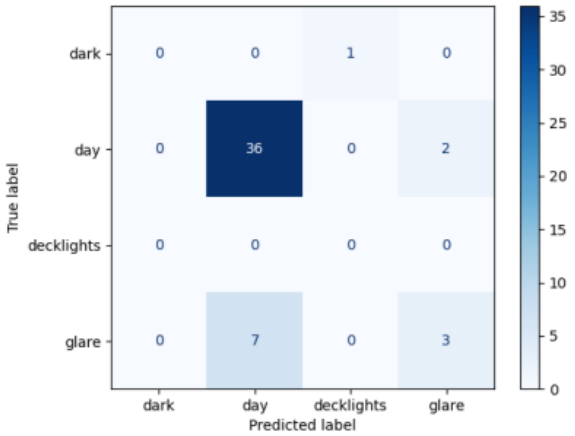


Figure 4.8: Confusion matrix for light level.

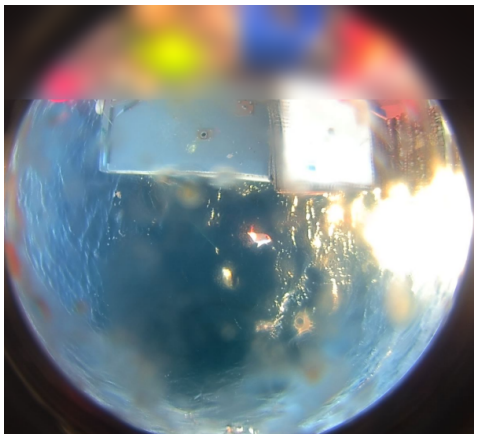


Figure 4.9: A snapshot labelled 'day' that was classified as 'glare' by the classifier. Note: Image is altered for data security reasons.

While this model functions well for daylight images, it needs further refinement and tuning to improve its performance for non-day classes. This is extremely difficult given the amount of data available, especially with the extremely unbalanced classes present. Potential labelling inconsistencies also add complications to this classifier. As the images were labelled by humans, there will naturally be errors present. It is also unknown what the threshold for an image being labelled 'glare' is, and therefore the classifier must work this out and generalize based on the data it's trained on.

4.4.3 Reviewability

The reviewability classifier does not use image data for making predictions. Instead, this classifier uses the labels produced by the clarity and light level classifiers for making predictions. As explained in Section 4.3, reviewability needs to take into account factors like

clarity when making predictions. Without this, it struggles to label abnormal images, like examples where the camera is obscured.

To train and test a classifier for reviewability, feature sets are created that contain an instances' actual light level and clarity. This data, along with the appropriate labels for reviewability, is then split into a test and training set. A RF classifier is then trained and evaluated per the standard project pipeline.

Performance

When the trained model was evaluated with a test set, an overall accuracy of 63% was achieved. Prediction accuracy for 'good' was very good. Precision was high at 89%, and recall of 76% giving it an F-score of 82%. Lots of instances were labelled as 'acceptable' by the system, as seen in Figure 4.10, but many were not correct resulting in precision being 50%. Recall is better at 69% giving a resulting F-score of 58%. Classification of 'poor' instances generally performed poorly, only achieving 44% precision. Recall was marginally better at 57%, and the F-score was 50%.

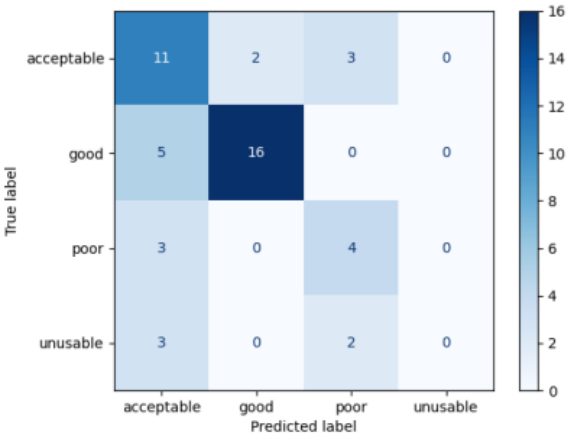


Figure 4.10: Confusion matrix for the reviewability classifier.

As the training data does not rely on feature extraction, this low result suggests that there are intricacies in the relationship between light level, clarity, and reviewability that cannot be learned with this small amount of data. This is evident in Figure 4.10, where 3 instances were predicted to be of 'acceptable' reviewability but should have been labelled as 'unusable'. This can also be observed for the 2 instances predicted to be 'poor' that should have been labelled as 'unusable'. These instances are highly likely to have 'obscured' as their label for clarity. There are not many instances like this in the dataset, but it would deem an images reviewability to be 'unusable' without the need to consider 'light level'. It is highly likely the model didn't encounter enough training examples to make this association, thus they were incorrectly predicted to be 'acceptable' during testing. It is also possible that if data has been labelled inconsistently by reviewers that the model is struggling to create an accurate generalisation.

4.5 Current System

4.5.1 Generating Reports

At the end of each successful pipeline execution, a report is generated for all of the classifiers. These reports are designed to present information about model performance in a readable and easy to understand manner. Before these reports were produced, results were reported in log files, or through printing to the Docker output. The reports contain the following details:

- Date and time of report generation.

- Time taken to train the model.
- General information, usually describing what features are used for the model.
- Information about the dataset, including classes, dataset size, class distribution and test/train set split.
- Performance metrics in the form of a classification report containing precision, recall, F-score for all classes.
- Confusion matrices.
- Test set gallery, with snapshots from the test set, with their actual label and their predicted label.

An example report for the reviewability classifier can be found in the appendix. However, it is missing the test set gallery to comply with the data access agreement with the industry stakeholders. These reports are discussed more later on in Section 5.2 and 6.1.

4.5.2 Data Augmentation

The limiting factor in this project has become a lack of data. Thus alternate measures of creating data were considered, as more data could likely not be labelled in the time available. Data augmentation can be used to increase the accuracy of classifiers using smaller datasets [26]. More images for the dataset can be augmented through several means. Common means of data augmentation on images include the use of random rotations, cropping, flipping and shifting of colours.

The only classifier that can benefit from dataset augmentation is clarity. The statistics extracted for light level would not change as a result of simple transformations. Reviewability only has two features, and attempting to augment data here would be the equivalent of adding duplicates to the dataset.

The first attempt at performing dataset augmentation was including images multiple times each with a random rotation in the dataset. Performance after adding this form of data augmentation was questionable. Recall was only 17% for 'good' instances, but it was 100% precise. For 'dirty' instances, 100% recall was achieved, but only 66% precision. F-scores of 80% and 29% were achieved for 'dirty' and 'good' respectively. This achieved 68% accuracy but was extremely biased towards labelling instances as 'dirty'.

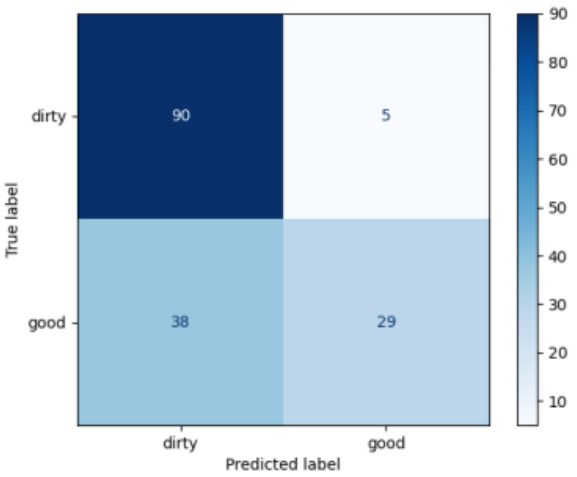


Figure 4.11: Confusion matrix for clarity after data augmentation.

With the second attempt to perform data augmentation for clarity, each image added to the classifier dataset is transformed through a series of flips, allowing each image to be used four times. This ensured that no areas of the image were lost outside of the frame, which occurred with random rotations. Through this, the images in the clarity dataset were increased from 135 to 540. When trialled, this performed better than the previous clarity model, increasing overall accuracy from 59% to 73.5%. However, the model

still suffers from a bias towards classifying instances as 'dirty', as seen in Figure 4.11. As seen in Table 4.2, recall for 'good' increased 2%, from 30% to 32%, but precision decreased slightly from 88% to 80%. Classification precision for 'dirty' instances increased with data augmentation, offering a better F-score.

	No Data Augmentation			**Data Augmentation**		
	Precision	**Recall**	**F-Score**	**Precision**	**Recall**	**F-Score**
Dirty	52%	94%	67%	69%	95%	80%
Good	88%	30%	45%	80%	32%	46%

Table 4.2: The performance difference between the clarity model in section 4.4.1 and the current model featuring data augmentation.

4.5.3 Class Balancing

As discussed at length in Section 3.1, all of the classes we are attempting to predict with models suffer from high class imbalance. Data augmentation in the clarity classifier simply multiplied the amount of data by four. This meant the class imbalance remained. The RF classifier in Scikit-learn has an option that automatically manipulates the model's weights inversely proportional to their frequencies within the dataset [27]. These weights influence how a classification model trains by penalising the incorrect classifications of minority classes more so than the majority classes. When this is trialled, it improved overall accuracy for clarity and reviewability but did not improve light level for any of the performance metrics.

	No Class Balancing			**Class Balancing**		
	Precision	**Recall**	**F-Score**	**Precision**	**Recall**	**F-Score**
Dirty	69%	95%	80%	71%	96%	82%
Good	80%	32%	46%	88%	45%	59%

Table 4.3: The performance difference between the clarity model with augmented data 4.5.2 and the latest model with class balancing.

As seen in Table 4.3, with class balancing all performance metrics for classification of clarity increased, clearly benefiting the model. While accuracy only increased 1.2%, the gains in precision and recall prove that this is a better model than the prior one. The largest improvement was in recall of 'good' instances, going from 32% to 45%. Increasing recall for 'good' instances in clarity means that the bias towards classifying instances as 'dirty' is being reduced. This is because more instances that should be classified as 'good' are correctly being classified as 'good'.

	No Class Balancing			**Class Balancing**		
	Precision	**Recall**	**F-Score**	**Precision**	**Recall**	**F-Score**
Acceptable	50%	69%	58%	52%	81%	63%
Good	89%	76%	82%	100%	71%	83%
Poor	44%	57%	50%	44%	57%	50%
Unusable	0%	0%	0%	0%	0%	0%

Table 4.4: The performance difference between the reviewability model in 4.4.3 and the latest model with class balancing.

Reviewability also benefited from the classifier allowing for class balance. Overall accuracy increased from 63.3% to 65.5%. As seen in Table 4.4, all performance metrics improved or stayed the same except for recall for 'good' instances. This decreased from 76% to 71%, but precision increased to 100%. The model was still unable to classify any 'unusable' instances, which can be attributed to the lack of training examples. Recall and precision for 'acceptable' instance classification also improved with a 12% increase in recall and a 2% increase in precision. The 'poor' class' performance did not benefit from the balancing changes. As discussed previously, some errors in this model may result from inconsistent labelling of the dataset. However, overall this model is a small improvement over the prior model that did not account for class imbalance.

Chapter 5

Evaluation

5.1 Fitness of Solution

The results of the various classifiers are interesting given the unbalanced and limited nature of the data. Further work and refinement is needed before the models produced could be used in a real-world system. For this system to be deemed fit for purpose, several metrics need further improvement and tuning for each of the classifiers produced.

Overall, reviewability needs to become more accurate, and the false positive rate especially needs to be minimised. If a snapshot was evaluated by the system, and the reviewability was estimated to be poor or unusable, this could be used to notify the vessel's crew to clean the lens of the camera. If false positives generated by the system repeatedly request that the camera be cleaned, this would likely annoy the crew and cause them to disregard the system.

The light level model is the most useful classifier produced currently, achieving 79.6% accuracy on unseen data. Evaluation and performance metrics have shown the statistical features selected for light level do allow for successful recognition of the different light levels. The performance of predicting lower light level classes, such as 'dark' and 'deck lights' needs to be improved, but more data is required.

The clarity model achieved 74.7% overall accuracy on unseen data, however with a reduced scope. Further work is required to add back in the other smaller classes to this classifier as these will be required for informing the overall reviewability of an image. Reviewability achieved 65.3% accuracy on unseen instances when informed by the 'clarity' and 'light level' features. There is room for further refinement, and the potential to include more features like 'activity' and 'camera position' to further inform the classifier. A fully functional system must also be able to classify 'activity' and 'camera position' too. Production of a model for these was not attempted. This was because assessing clarity, light level and reviewability was deemed to be a good starting point to test the viability of creating accurate classifiers with the limited dataset. A system used for classifying these attributes would require more data. Deep learning would be beneficial for classifying these, as a model that can handle high dimensionality would be required for such a classification task.

5.2 Feedback from Stakeholders

Feedback from the industry stakeholders was limited over the duration of the project as other commitments were of a higher priority to them. However, they liked the automatically generated reports for each classifier as they were good summaries of current progress.

5.3 Deliverables

Over the course of this project, a well organised and effective pipeline has been created that allows for easy prototyping and reporting. As stated above in Section 5.2, the reports generated by the system are tidy, concise and informative. They allow easy viewing of key performance metrics and comparison of labels upon actual instances from the test set. The report generation code is easily extensible to allow for additional content to be included in the future.

Further evolution and addition of classifiers are easily achieved with the system because of the component-based pipeline. As shown in Figure 3.7, steps can be added, removed or modified as required. In the future, more classifiers can be added, utilising the same data retrieval, processing and reporting systems.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

This project suffered from a slow start due to unforeseen circumstances that were described in the Preliminary Report, which can be found in the appendix. However, I believe the results produced were promising, and they highlighted the challenges of working with small datasets featuring extreme class imbalance. As a result, the high accuracy of the classifiers produced can be attributed to biases within the models trained. This is a direct result of the class imbalance present within the data supplied for this project. Several different methods of extracting features were trialled and evaluated. Data augmentation was explored with some success. Parameter tuning of the classifiers to account for class imbalance offered marginal improvements, but unfortunately the amount of data available ultimately limited the ability to produce an effective set of models that would solve the problem posed.

This project created several deliverables that have the potential for future use in the development and refinement of models. The pipeline created over the course of this project does not require any changes to accept new data that would likely improve the overall performance of the classifiers. The reports generated by the pipeline produced also allow a quick and detailed look at the performance of the individual classifiers, and as stated in Section 3.5, this success was something our industry stakeholders expressed an interest in early on.

6.2 Future Work

The most beneficial thing for future work on this project would be the addition of more data. The number of instances just simply isn't enough for producing a classifier as complex as required for the problem posed. This problem was made worse with the imbalance within the various classes. If more data were available for use, it would also potentially allow for the use of a deep learning approach in the future, instead of the traditional feature extraction and classifier approach explored. Deep learning and the use of convolutional neural networks was originally desired for this project, but not possible with the dataset size. To overcome the issues with dataset size, the use of pre-trained models may increase accuracy. A pre-trained model is a model created by somebody else for a similar problem and allows the learning from that model to be potentially transferred or used as a starting point for improving another [28]. A model that detects dirt or water on cameras for automotive purposes may be a good starting point for this application. This would potentially allow a model of high accuracy to be created, without having to label or augment more data.

Bibliography

[1] S. Sunbird, "Grey-faced petrel." [Online]. Available: https://commons.wikimedia.org/wiki/File:Grey-faced_petrel.JPG

[2] Ministry for Primary Industries, "Fisheries observers," Sep 2020. [Online]. Available: https://www.mpi.govt.nz/about-mpi/careers/working-mpi/roles-at-mpi/fisheries-observers/

[3] I. El Naqa and M. J. Murphy, *What Is Machine Learning?* Springer International Publishing, 2015, pp. 2–3. [Online]. Available: https://doi.org/10.1007/978-3-319-18305-3_1

[4] Z. Zou, Z. Shi, Y. Guo, and J. Ye, "Object detection in 20 years: A survey," *arXiv preprint arXiv:1905.05055*, 2019. [Online]. Available: https://arxiv.org/pdf/1905.05055.pdf

[5] X. Li, Z. Liu, B. Li, X. Feng, X. Liu, and D. Zhou, "A novel attentive generative adversarial network for waterdrop detection and removal of rubber conveyor belt image," *Mathematical Problems in Engineering*, vol. 2020, 2020.

[6] C.-C. Lai and C.-H. G. Li, "Video-based windshield rain detection and wiper control using holistic-view deep learning," in *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*, 2019, pp. 1060–1065.

[7] M.-J. Zheng, S.-C. Hsia, and S.-H. Wang, "Raining detection with deep learning method for vehicle system," in *2021 IEEE International Conference on Consumer Electronics (ICCE)*. IEEE, 2021, pp. 1–4.

[8] D. Rodin and N. Orlov, "Fast glare detection in document images," in *2019 International Conference on Document Analysis and Recognition Workshops (ICDARW)*, vol. 7, 2019, pp. 6–9.

[9] L. Yahiaoui, M. Uřičář, A. Das, and S. Yogamani, "Let the sunshine in: Sun glare detection on automotive surround-view cameras," *Electronic Imaging*, vol. 2020, no. 16, pp. 80–1, 2020.

[10] J. T. Behrens, "Principles and procedures of exploratory data analysis." *Psychological Methods*, vol. 2, no. 2, p. 131, 1997.

[11] G. Litjens, T. Kooi, B. E. Bejnordi, A. A. A. Setio, F. Ciompi, M. Ghafoorian, J. A. Van Der Laak, B. Van Ginneken, and C. I. Sánchez, "A survey on deep learning in medical image analysis," *Medical image analysis*, vol. 42, pp. 60–88, 2017.

[12] M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim, "Do we need hundreds of classifiers to solve real world classification problems?" *The journal of machine learning research*, vol. 15, no. 1, pp. 3133–3181, 2014.

[13] L. Chen and X. Cheng, "Classification of high-resolution remotely sensed images based on random forests," *J. Softw. Eng*, vol. 10, pp. 318–327, 2016.

[14] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[15] D. ping Tian *et al.*, "A review on image feature extraction and representation techniques," *International Journal of Multimedia and Ubiquitous Engineering*, vol. 8, no. 4, pp. 385–396, 2013.

[16] S. van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, and T. Yu, "scikit-image: image processing in python," *PeerJ*, vol. 2, 2014.

[17] Google Brain Team, "Tensorflow: An end-to-end open source machine learning platform." [Online]. Available: https://www.tensorflow.org/

[18] OpenCV, "Opencv: Open source computer vision library." [Online]. Available: https://opencv.org/

[19] N. M. Zaitoun and M. J. Aqel, "Survey on image segmentation techniques," *Procedia Computer Science*, vol. 65, pp. 797–806, 2015.

[20] R. Maini and H. Aggarwal, "Study and comparison of various image edge detection techniques," *International journal of image processing (IJIP)*, vol. 3, no. 1, pp. 1–11, 2009.

[21] Z. Guo, L. Zhang, and D. Zhang, "A completed modeling of local binary pattern operator for texture classification," *IEEE transactions on image processing*, vol. 19, no. 6, pp. 1657–1663, 2010.

[22] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE computer society conference on computer vision and pattern recognition*, vol. 1. Ieee, 2005, pp. 886–893.

[23] D. Hasler and S. E. Suesstrunk, "Measuring colorfulness in natural images," in *Human vision and electronic imaging VIII*, vol. 5007. International Society for Optics and Photonics, 2003, pp. 87–95.

[24] C. A. Bouman, "Opponent color spaces," 2021. [Online]. Available: https://engineering.purdue.edu/~bouman/ece637/notes/pdf/Opponent.pdf

[25] L. M. Hurvich and D. Jameson, "An opponent-process theory of color vision." *Psychological review*, vol. 64, no. 6p1, p. 384, 1957.

[26] J. Wang, L. Perez *et al.*, "The effectiveness of data augmentation in image classification using deep learning," *Convolutional Neural Networks Vis. Recognit*, vol. 11, pp. 1–8, 2017.

[27] Scikit-learn Developers, "Random forest classifier." [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[28] P. Marcelino, "Transfer learning from pre-trained models," *Towards Data Science*, 2018.