Identifying Icebergs from Space:
Remote Sensing & Deep Learning
for Commercial Shipping

Abstract:
In this project, we were tasked with “develop[ing] a machine learning algorithm that will extract intelligence from high resolution, multispectral satellite imagery”. This report details the development and training of two Convolutional Neural Networks to identify ice formations from remote sensing data, primarily for predicting where icebergs are along shipping routes, and as an extension for integration into more advanced navigational software that we propose for commercial use in international shipping.

A classification model was successfully developed, capable of identifying whether a given image contains an iceberg. We found this model to be 65% accurate, which we found to be limited by its small training set of ~100. We also demonstrated its capability to reach higher accuracy with more data. We also developed a segmentation model, which aimed to accurately label every pixel in a satellite scene as land, sea, iceberg or icesheet. On limited training, it reached 27% accuracy, insufficient for direct industrial implementation, but we have identified shortcomings and proposed next steps for improvement. With further refinement, this model could be integrated into software to predict the path of iceberg movement with respect to their shape (and other factors detected by other means) to reduce risk to vessels and open up arctic shipping routes. We establish a robust financial case for this technology’s development and implementation, with a focus on making the Northwest Passage and Northern Sea Route more viable for international shipping.

Students:
Joshua Perkins (209002082)
Jonathan Marsh (219021422)
Altamush Khawaja (219012419)
Hamza Hamed (209029699)
Ammar Alfahdi (219047887)

Dates of work:
29/09/2023 to 11/01/2024

1 – Introduction

1.1 The Project

The University of Leicester Physics department's Group Industry Project is a 3rd year module, designed to build research, industrial and team-based skills, enhancing students career development. Students undertake an original research project based around a problem being faced in industry, working towards a solution using industry standard project management frameworks and specialist technical training. The project involves presenting slides and a report to the cohort, university staff and industry partners and delivering a proposal on taking the project further to the industrial partner. The challenge set by our industrial partner is outlined below:

"In this project, teams will develop a machine learning algorithm that will extract intelligence from high resolution, multispectral satellite imagery. The team will be required to develop an in-depth understanding of ESA's Sentinel fleet and Airbus' high-resolution imagery systems (such as SPOT and Pleiades) in order to identify and obtain relevant satellite imagery from large data repositories. The teams will then use Geographic Information System (GIS) to create a collection of training data for the development and deployment of new analytical techniques."

Figure 1 A slide from our first meeting with Dr. Liam Harris of Airbus Intelligence

1.2 The Company

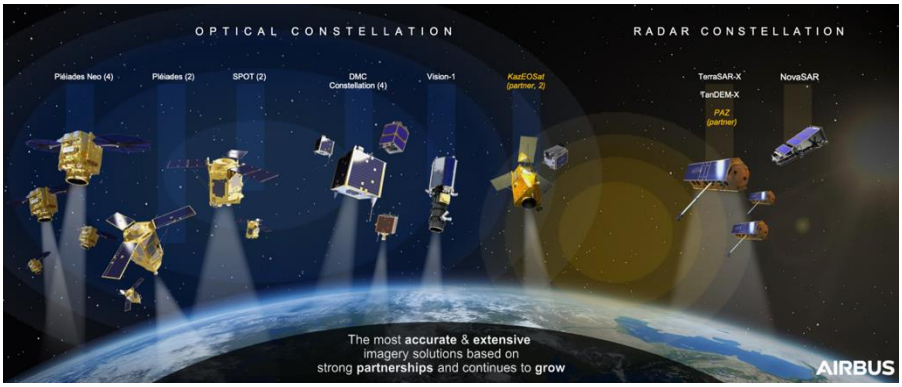


Figure 2: A slide from our first meeting with Dr. Liam Harris of Airbus Intelligence displaying the Airbus Intelligence fleet

Our partner company was Airbus Intelligence, part of the largest aerospace and defense company in Europe. Intelligence handles activities like data analysis and AI to extract useful information from satellite imagery, owning and operating their own satellite constellations (Pleiades, DMC, SPOT). The data from these satellites is then used commercially. Our external contact and mentor was Dr. Liam Harris, a geospatial applications developer with Airbus Intelligence.

1.3 The Team

Our team consisted of third year students Joshua Perkins (Physics with Space Science), interested in Earth Observation Science, Jonathan Marsh (Physics) interested in data science for social good, Hamza Hamed (Physics), interested in software development, Ammar (Physics) interested in business, and Altamush Khawaja (Physics with Space Science), also interested in data science.

1.4 Project Identification

1.4.1 Potential Ideas

The first task as a team was to brainstorm ideas on the problem we would solve. The team brainstormed some ideas:

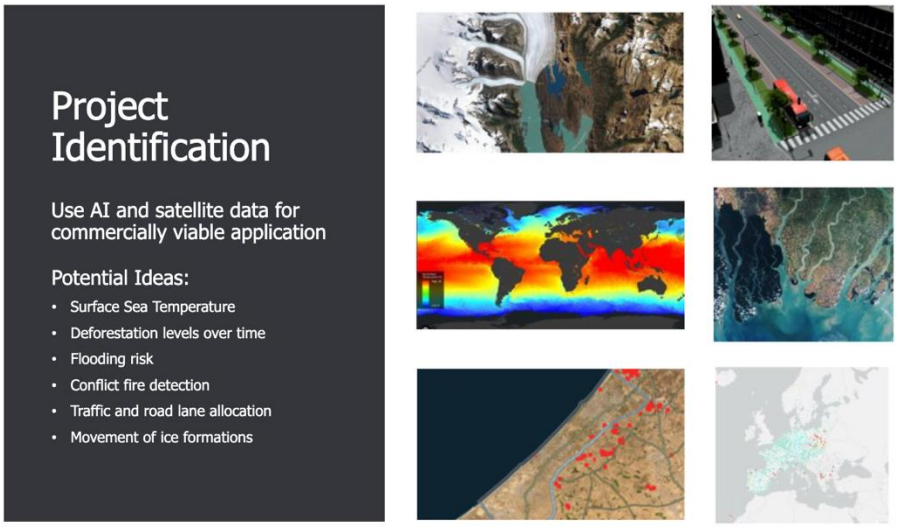


Figure 3: A slide from our final presentation detailing our potential projects

After discussion based on the data available, the immediate market situation and driving factors from Airbus, the area we chose was a neural network which would recognise and track ice. Our reasoning is discussed in the following sections.

1.4.2 The Problem

With climate change driving not only an increased break up of sea ice but also increased glacial melt, northern sea routes are becoming more accessible, but icebergs are also more present. This presents both an opportunity and a danger to global shipping. Increased access to arctic shipping allows companies to use routes much shorter than the traditional routes through Canals, saving fuel, time and money, but at greater risk. Currently assessing iceberg risk is done using arial and shore-side monitoring. For this market to expand this is unsustainable and unscalable, so the only meaningful way to build such a tool would be using satellite data and AI, namely computer vision neural networks.

1.4.2 Decision Rationale

While data for the flood risk and sea surface temperatures analysis ideas is plentiful, we decided that both ideas had too large a scope for this project, especially with limited computing resources. The conflict fires idea was limited by unbalanced and unlabeled data. Lastly, deforestation is well studied, so we felt the business case and scope for novel work was stronger for tracking icebergs. In addition, we were informed Airbus is currently interested in tracking icebergs for maritime routing; we also explored secondary use cases including climate and geomorphological research, providing further incentive. This proposal was greeted with enthusiasm, both across the team and our supervisors and external contact. Therefore, given readily available data through Sentinel and Pleiades, a manageable scope, a novel concept, multiple use cases, and a promising market, we decided our challenge would be to find icebergs in satellite data with AI for the benefit of commercial shipping.

1.5 Similar Solutions

Few similar solutions exist for the cryosphere and ocean, with very limited previous work on icebergs and icesheets. As such, looking for similar solutions that we might be able to learn from, we turned to Airbus and Dr. Liam Harris, with their previous work. A notable example was using satellite data and AI to detect oil spillages. Using semantic segmentation, along with other similar neural network applications of satellite imagery proved helpful in establishing our scope, requirements, and model architecture. Other existing models and applications we discussed were delineating farming fields, monitoring road conditions and detecting ships. The neural network picking ships out of satellite imagery was closest to our proposed solution and we found another similar solution as a submission to a challenge set by Statoil and C-Core. This challenge was to create an AI program, taking satellite data as input and identifying whether objects within the image were ships or icebergs. We found the approaches to the challenge very useful in terms of technical architecture and the input of data.

Figure 4: Examples of Airbus Intelligence Neural Networks

1.6 Proposed Solution

We propose a deep learning tool to analyse the cryosphere and ocean, searching for and predicting the probability of icebergs present. The input data would be visual satellite imagery. A Convolutional Neural Network (CNN) was to be used, either a classification model or a more complex segmentation model.

2 – Project Strategy

In this section we describe our approach to developing this solution, including the framework by which the project was managed, a full science flow down, technical requirements and the tools we used along the way.

2.1 Solving the Problem

To implement the proposed solution, we primarily needed to gather and manipulate data and then build and test a neural network to process this data and output the probability of icebergs within this data. Firstly, a project management framework needed to be formed. From this relevant research and training could be undertaken. Then data exploration and processing could begin alongside coding. To solve the problem, the main challenge we would face was needing the program to be able to distinguish between ice sheets, icebergs and snowy land. This would be done by identifying select data and significant training runs of the program.

2.2 Project Scope

The minimum output was expected to identify whether an image contained an iceberg accurately. The maximum expected output would be a segmentation model that predicts every pixel of an image accurately. From here a plan for taking the models further into marketable solutions was developed. This ensured project scope would be constrained given the limited time available for the project. It was expected many further applications may appear over the project (discussed in section 3.4). However, the potential scale and usefulness of the tool presented risks in terms of the project's scope. A well-defined project management approach and framework would be essential in ensuring the project progressed on time.

2.3 Overall Approach: ‘The Iceberg Model’

We designed our project management approach around the phases of iceberg formation (or calving). This process can be said to have three distinct stages: cracking, the shooter, then consolidation. In the first stage many forces are applied to the ice and fissures appear directing what will calve to become the iceberg. Then with little warning a chunk of ice splits off, overturns and rises sharply upwards as what's called a shooter. It then settles, with loose material falling off and an iceberg being formed.

Drawing our framework from the very natural process forming the icebergs we’d be finding; these 3 phases seemed appropriate. Phase 1 being our research and training, analogous to the stressing of the icesheet, where you do not see much happening, despite being crucial. Phase 2 represented a sudden jump in progress during implementation based on the groundwork, training and experimentation in Phase 1, and then in Phase 3 activities converged on the desired aims, outputs validated, and the whole project condensed into a presentation and report.

Figure 5: Graphic explaining the various project phases

2.4 Technical Requirements

With the initial project plan in place, the original objectives and technical requirements of this project was a computer vision Convolutional Neural Network (CNN) as follows:

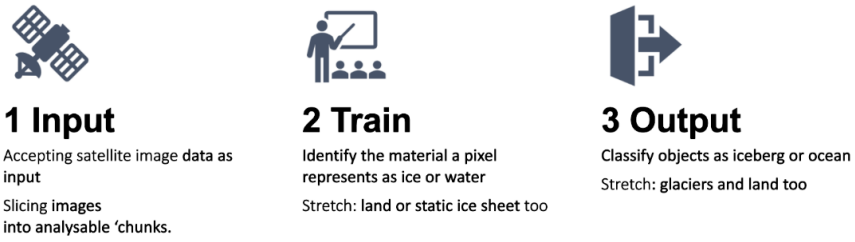


Figure 6: Original technical requirements

After significant research, towards the end of Phase 1 of the project, these requirements were refined into two outputs:

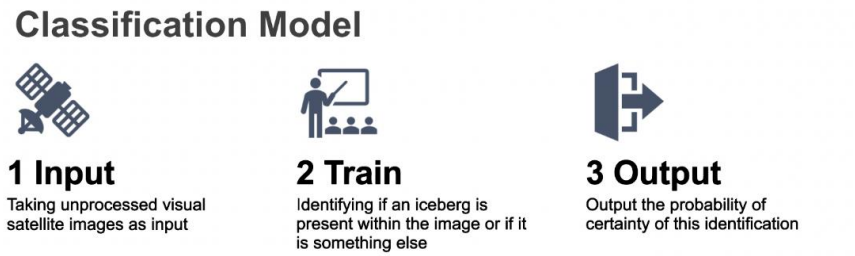


Figure 7: Technical requirements for the Classification model

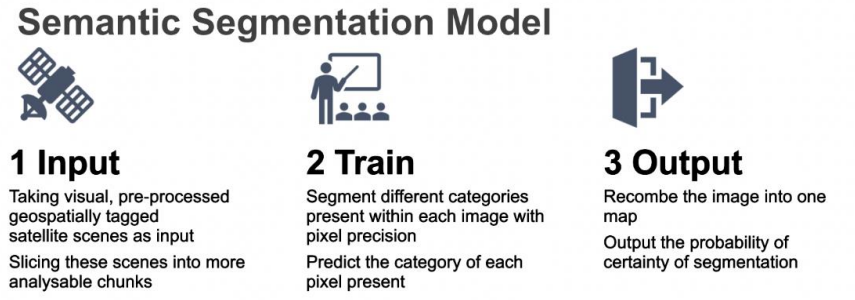


Figure 8: Technical requirements for the semantic segmentation model

These requirements then formed a full science flow down which guided our progress towards achieving them:

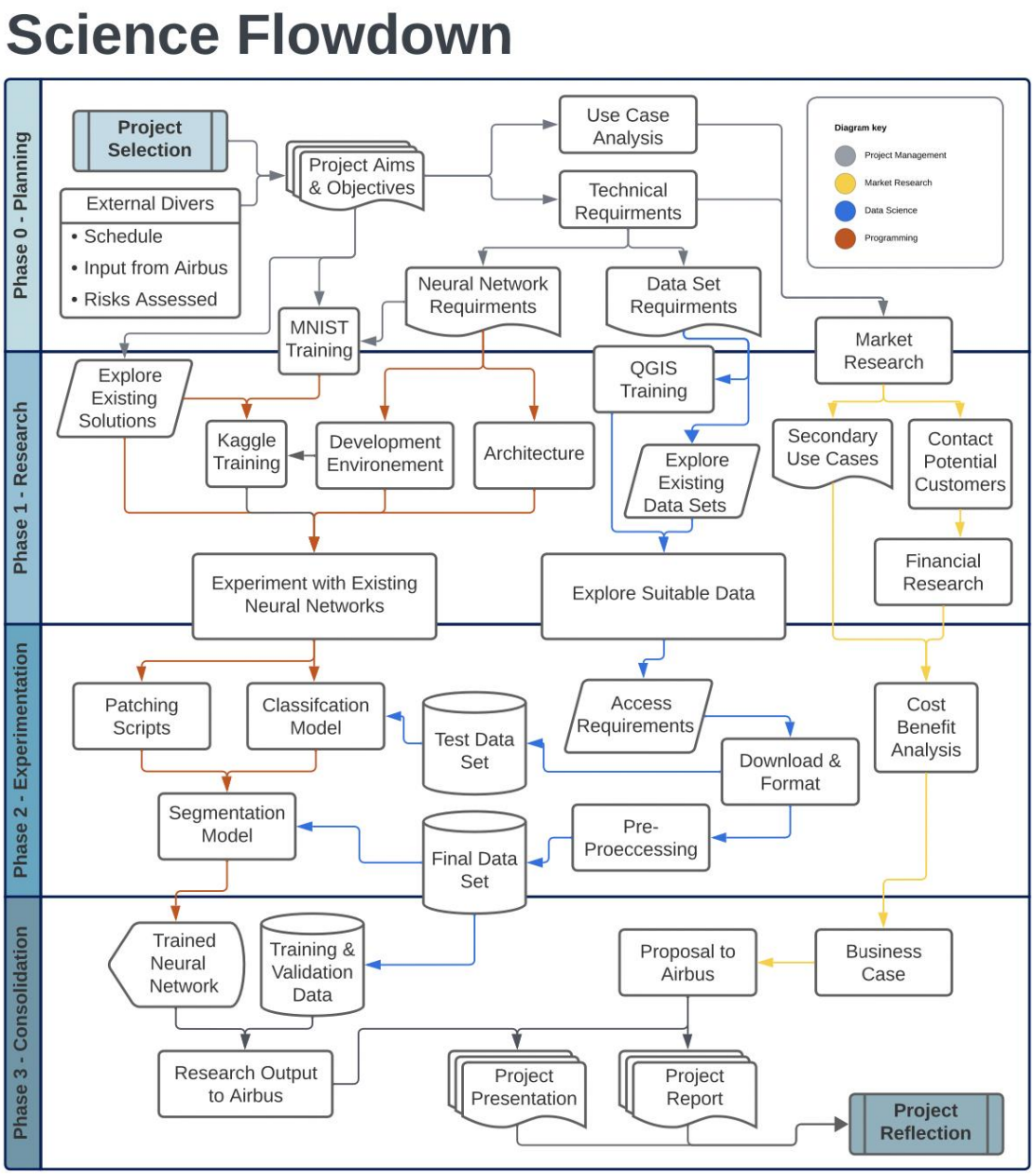


Figure 9 Science flowdown chart detailing the expected science and technical steps and decision of the project

2.5 Project Management framework

With a project structure in place, the next major objective was creating a framework for activities. Ensuring maximum collaboration, efficiency, tracking of progress and mitigating risks, to produce a successful outcome within our timeframe. This framework was made up of many induvial tools and documents, detailed below:

2.5.1 Technical Risk Assessment

There were significant technical risks present which could have affected the project's success. The major risk in this project was a lack of previous programming knowledge, which would have affected the overall progress of the project as the coding and neural network model would slow until sufficient experience was gained. This was mitigated by ensuring that all members of the team were staying up to date and understood the requirements for wider support of the project and continuous training during the project.

The second risk was project scope creep. This could occur because of a new creative use case being discovered or aiming too far to reach beyond the established objectives and not having a functional model by the end of the project. To mitigate this, objectives were narrowed, and the project's progress was constantly tracked and discussed; a record was kept of potential use cases and ideas for further development to reduce scope creep.

More expected risks and their proposed mitigations are discussed in the risk assessment below:

Technical Risk Assessment

Id No.	Risk	Potential Trigger(s)	Probability	Impact	Risk Score	Mitigation
TR1	Project scope creep	Useful functionality created outside of aims; creative end use case discovered	4	3	12	Keep to project aims and objectives during development; track and discuss project progress and keep list of potential use cases and ideas for future development.
TR2	Programming knowledge gap	Previously not understood coding techniques and requirements	4	2	8	Ensure multiple members if not the whole team understand the required programming for wider support; continuous training across the project; communicate technical issues with the External at weekly meetings.
TR3	Lack of computing resources	Limited GPU/CPU capacity on personal & Uni computers	3	3	12	Spread out computing load across members and explore potential for High Performance Computing access.
TR4	Lack of adequate data sets	More data being required to train the neural network	2	4	8	Make sure during planning stage that there is plenty of accessible data available
TR5	Data corruption	File system errors etc.	1	5	5	Create backups/ multiple copies

Personnel Risk Assessment

Id No.	Hazard	Risk	Severity before Mitigation	Mitigation measures	Severity after mitigation
PR1	Computer keyboard/mouse	Repetitive Strain Injury	2	Team members take regular breaks from coding and distribute workload evenly.	1
PR2	Computer screens	Eye Strain	2	Team members to take regular breaks from computer work.	1
PR3	Computing resources (this is technical)	Equipment Malfunction	3	Computers to be checked before use for faults and PAT tests.	1

Figure 10 Table containing our technical risk assessment

2.5.2 Meeting Minutes

We had regular meetings with the team, the supervisors and our external partner, to discuss progress and next steps, especially with so many areas of work and consistent progress throughout. In all meetings, we kept notes on discussion, key advice or information received, and all agreed upon actions. These minutes and associated actions are included in appendix C.

2.5.3 Gantt Chart

The team used a Gantt chart to track progress during the project as is shown in figure 11. Figure 11shows the Gantt chart at the start of the project. Things evolved steadily over time, and intermediate renditions of the chart are available in appendix C.

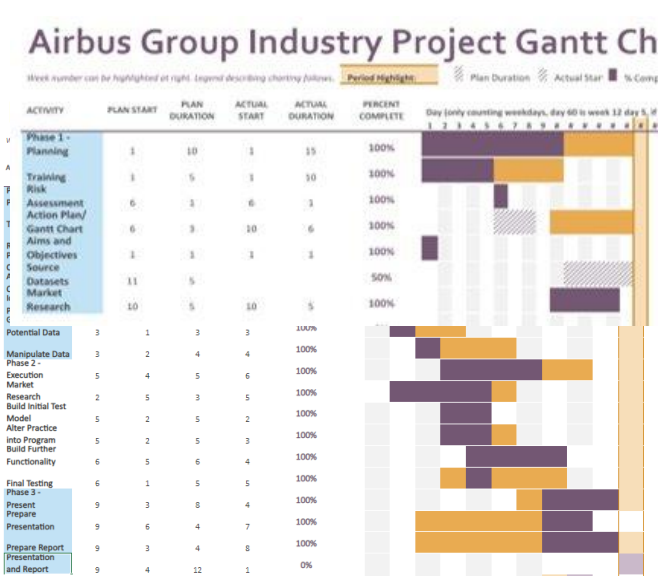


Figure 11: Gantt chart at the beginning of the project

The Gantt chart, was employed to track progress in different aspects week-on-week, showing us where we were falling behind our expectations, and this helped us develop solutions and reallocate resources to where they were needed instead of having an excess in certain areas and a lack in others. The Gantt Chart underwent alteration as the project's needs changed.

Figure 12: Gantt chart at the end of the project showing our progress each week in purple, as well as showing where we had fallen behind/gotten ahead in orange

2.5.4 Team building

Early in the project we came together and completed a team building exercise, doing a quiz together on what famous scientist we were most like, with factors ranging from work ethic to the introvert/extrovert scale and even whether we like croissants! We found the activity engaging and it, along with intentional team building across the project, helped us to take shape as a cohesive team working well together and enjoying the process.

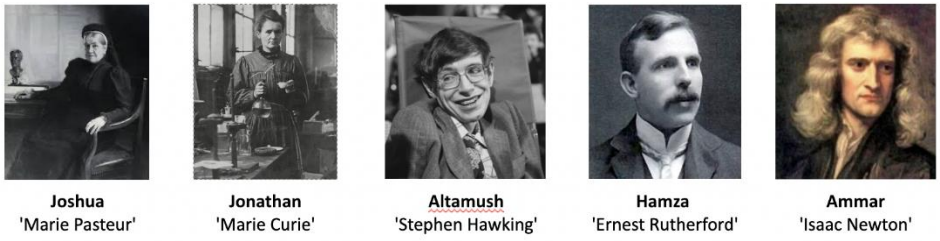


Figure 13: Results of our team building exercise

2.5.5 Training

A significant element of Phase 0 was training to build neural networks on the commonly used MNIST database[1] and learning to manipulate data using QGIS [2]. This training was crucial and informed our technical approach to data and CNN's.

2.5.6 File Sharing & Cloud Based Coding

The software we used to share and manage most of the content of the project was OneDrive, as we could share an entire folder amongst the team and all work on different aspects at the same time. This could be done while still maintaining the ability to monitor or edit the work of others where needed, which was used heavily in the production of the report as well as the presentation. We also used Google Co-lab which is a cloud-based repository, notebook and programming environment with access to GPU's and TPU's, allowing us to code together on the project simultaneously. This process of pair programming allowed us to combine our skills and expediate bug-fixing, each programmer being able to see what the other had missed. These systems created a streamlined workflow between data storage and model development.

2.6 Personal Contributions

During Phase 0, we evaluated each team member's experience, strengths and interests, then distributed responsibility accordingly.

Joshua:

- **Project Management**: Responsibility for communication between the team, our industry partner and academic supervisors; delegation of tasks across the team; and weekly progress summary presentations/feedback sessions with supervisors.
- **Data science**: Researched existing data sets and products; Discovered, manipulated and pre-processed images for the final data set.
- **Presentation & Report**: Contributed to Project Management, Research, Data, Discussions and Conclusions sections.

Altamush:

- **Project Management:** Responsible for creating and maintaining Gantt chart, as well as keeping a record of meeting minutes
- **Market Research:** Early market research, factors which would affect product value, fuel costs, ideas to find product cost through extrapolating known averages of data.
- **Presentation & Report:** Contributed to Introduction, Project Management, Market Research, and Skill Reflection sections.

Hamza:

- **Programming:** Developed the classification model. Developed the training and testing script for segmentation neural network. Managed model training, optimisation, and evaluation.
- **Presentation & Report:** Contributed to method, results and discussion for the models. Prepared and presented materials that delineated the model's development process, training strategies, and performance.

Jonathan:

- **Programming:** Researched architecture, libraries and resources, coded data processing for the segmentation model, contributed to the segmentation model.
- **Presentation & Report:** Contributed explanatory sections for data processing and architecture, assisted with accurate technical language throughout, and wrote the abstract.

Ammar:

- **Market Research:** Studied economic metrics of the market and target companies and communicated with the target companies.
- **Presentation & Report:** Market Research and Skill Reflection sections.

In the reflections section of this report, we each discuss what we have learnt and how we have developed our skillsets.

3 – Research

3.1 Target Markets and Logistics

The main target market for this product is shipping companies who wish to access arctic maritime routes. It also has academic uses in climate and geomorphological research. Tracking the location and movement of ice would allow safe and efficient routes through icy waters, opening up the Northwest Passage and Northern Sea Route. These routes are used less than equatorial routes through the Panama and Suez canals that these routes bypass.

There are several factors which would affect the value of this product. A few of these include: the cost of data, the number of applications, the scalability and the amount of competition. For example, all data may not be open source and will need to be purchased which will have an associated cost. Furthermore, if this model offers something that other similar models do not, it will become a major selling point and allow for the price of sale to be higher. Further information on how these routes bypass the canals while still traversing between relevant economic centers can be found in appendix section B.

3.2 Looking at the Numbers

~18000 ships each year pass through the Suez Canal[3], each paying tolls. This is whilst only 2994 journeys were made through the Northern Sea Route in 2022. Similarly, ~15000 ships pass through the Panama Canal each year, and pay tens of thousands for access. In comparison, between 2017 and 2019, only 21 ships passed through the Northwest Passage on average each year.

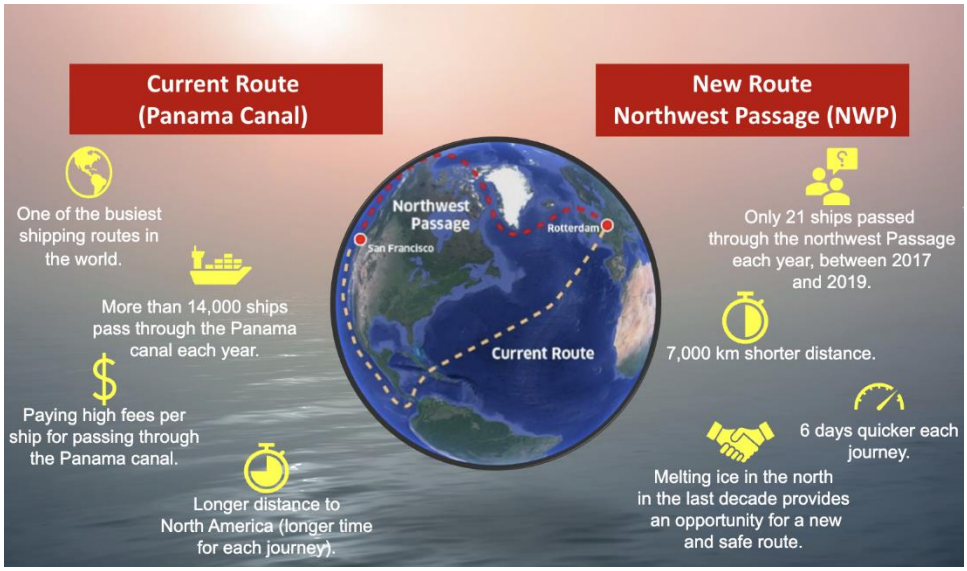
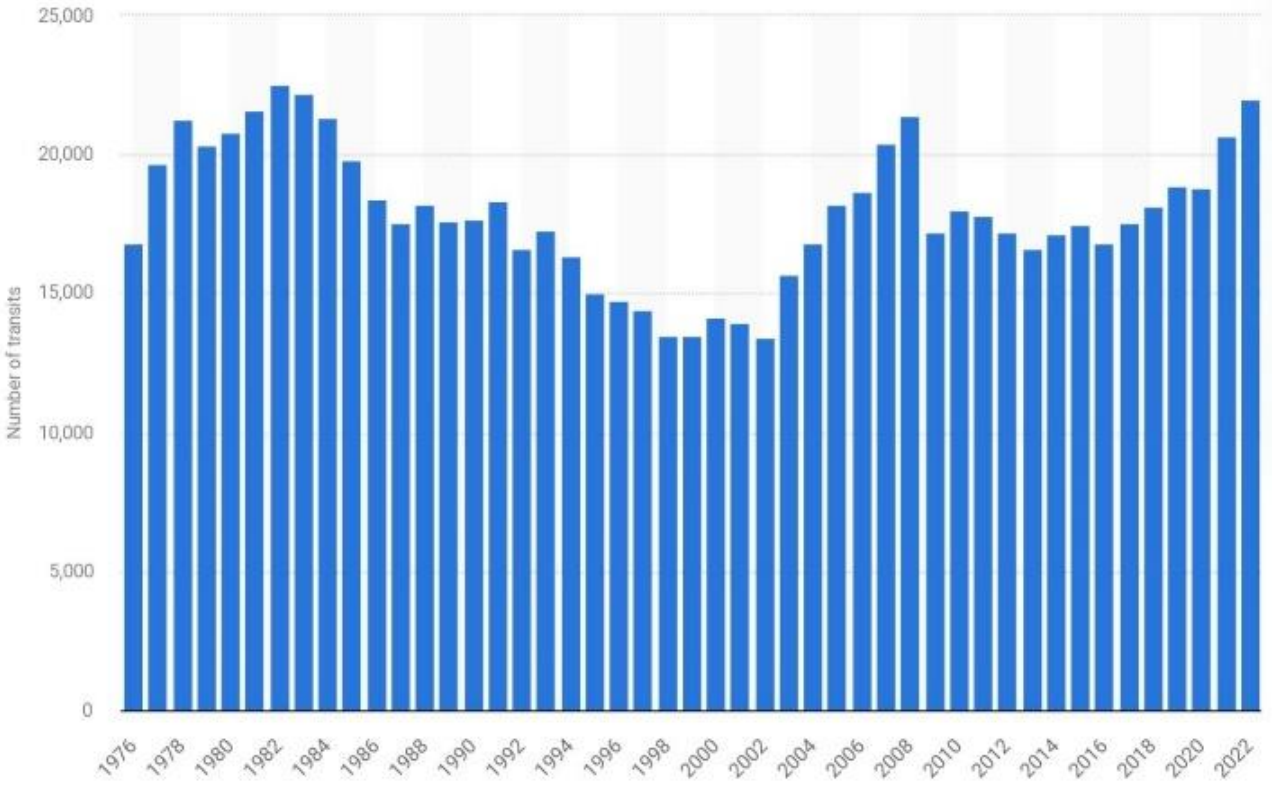


Figure 14: Graphic summarising the proposed target route vs the dominant Panamá Canal route

Figure 15: A bar chart showing the number of ships which travelled through the Suez Canal each year



since its opening in 1976 to 2022

The cost of going through the Panama Canal varies depending on several factors such as the size and type of vessel, the type of cargo, and the time and day of transit. The Panama Canal Authority uses a pricing system that considers these factors and determines the total cost.

For a commercial vessel, the cost of transiting the Panama Canal typically ranges from $3,000 to $450,000. The price is determined based on the vessel's displacement and dimensions. A Panamax vessel (294x32 meters) can cost $150,000 to $375,000 for a one-way transit. A neo-Panamax vessel, as (366x49 meters) can cost $250,000 to $500,000 for a one-way transit and would cost more for a return trip.

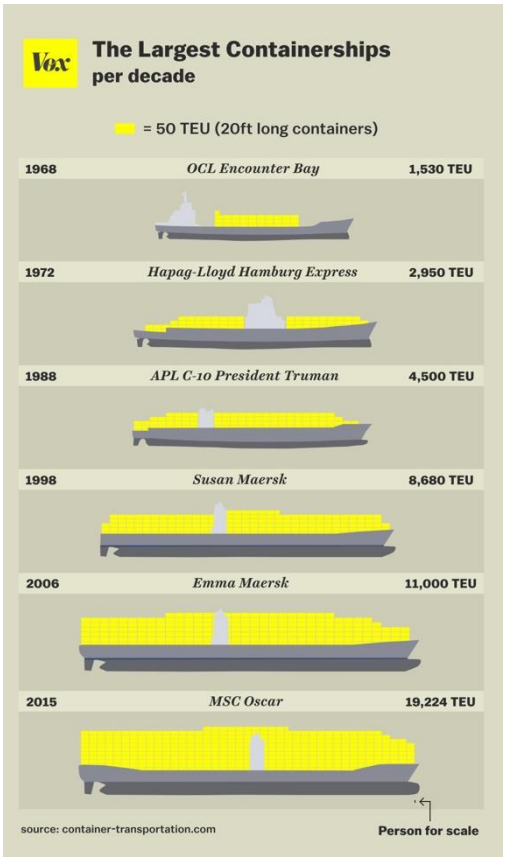


Figure 16: Graphic displaying the container-count capacity of different vessels

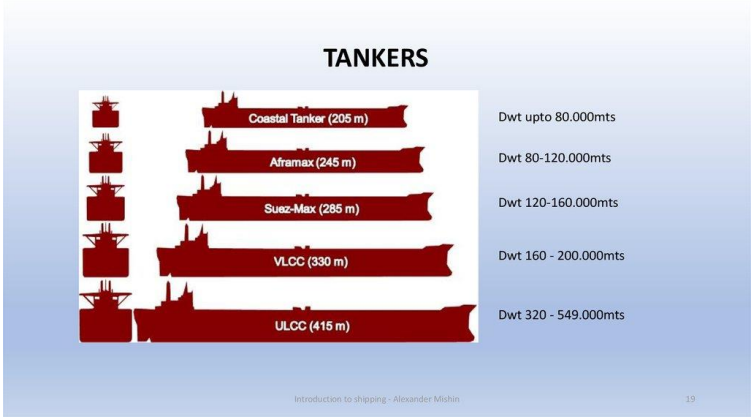


Figure 17: Graphic displaying the sizes of different vessels

This shows how large of an untapped market is available in the Northwest Passage, as companies could save huge sums from bypassing these charges alone, without even considering the fuel savings.

From this information, the market for a system which tracks ice and its movement is clearly huge, and that is without even considering the scientific use cases.

Deeper analysis allowed for the savings in the cost of fuel to be calculated from the decrease in distance which would need to be travelled by using the Northwest Passage. It is important to note that this is only an estimate. By using the Northwest Passage, instead of the Panama Canal, the saving from fuel cost would be approximately £2.12 million per journey. Further information and the calculation of this value is available in the appendix under section C.

3.3 Secondary Use Cases

While the primary customers of this tool would be shipping companies, there are several potential secondary uses. This technology could be used for monitoring ice sheet loss, observing glacial retreat, permafrost melt and other landcover applications.

4 – Experimentation

4.1 Classification model

Designed and implemented is a classification model capable of distinguishing between images of icebergs and other objects. The model is built and trained using a deep learning Convolutional Neural Network (CNN) architecture. This details the development process, including data preparation, model design, training, and evaluation.

4.1.1 Data Preparation

To create an image classification model that differentiates 'Icebergs' from 'Other' categories, we collected and stored pre-processed images in a Google Drive directory. This approach allowed for seamless access and manipulation using Google Colab. More than 100 mid to high resolution images were downloaded from ESA Earth Online Archives [4] and a portal called Terrascope [5], using Sentinel 2 colour imagery, selected with a varied mix of ocean, ice-sheet, land and icebergs. The limited preprocessing of these images involved steps to ensure they were in the proper format for the model to process. Images were read into an array and converted to the RGB colour space to standardise input. Following this, they were resized to a uniform dimension of 512x512 pixels. Finally, pixel values were normalized to fall within the 0-1 range, preparing them for efficient processing by the neural network.

4.1.2 Model Architecture

The model employed a sequential Convolutional Neural Network, leveraging TensorFlow and Keras. It begins with an input layer designed to accept 512x512 RGB images. The core of the model consists of three convolutional layers for feature extraction, each with an increasing depth of 32, 64, and 128 filters respectively, and a kernel size of 3x3. These layers are followed by a 2x2 max-pooling layer to reduce the spatial dimensionality while retaining critical information. After passing through these layers, the data is flattened into a single long feature vector.

The model then transitions into a dense layer of 512 units, utilizing a ReLU activation function to introduce non-linearity and aid in learning complex patterns. This leads to the final output layer, which is a dense layer with 2 units corresponding to the 'Icebergs' and 'Other' classes. This layer employs a SoftMax activation function to provide a probability distribution over the classes, indicating the model's predictions.

4.1.3 Training Process

With the architecture in place, the model underwent a compilation process using the Adam optimizer and a categorical cross-entropy loss function, for efficient training and convergence. The training process spanned over 20 epochs, with the model processing batches of 32 images in each iteration. Accuracy was selected as the primary performance metric for the model, providing, defined by the proportion of correctly predicted images in the test set, allowing for straightforward evaluation of the model's capabilities.

4.2 Segmentation data pre-processing

To provide the pixel-precision truth data the model needed for training our more advanced segmentation model, we needed to pre-process our data. This process consisted of four steps: identifying suitable satellite scenes, tagging the scene into different classes, converting these vectors into readable masks, then splitting or patching into smaller bitesize chunks.

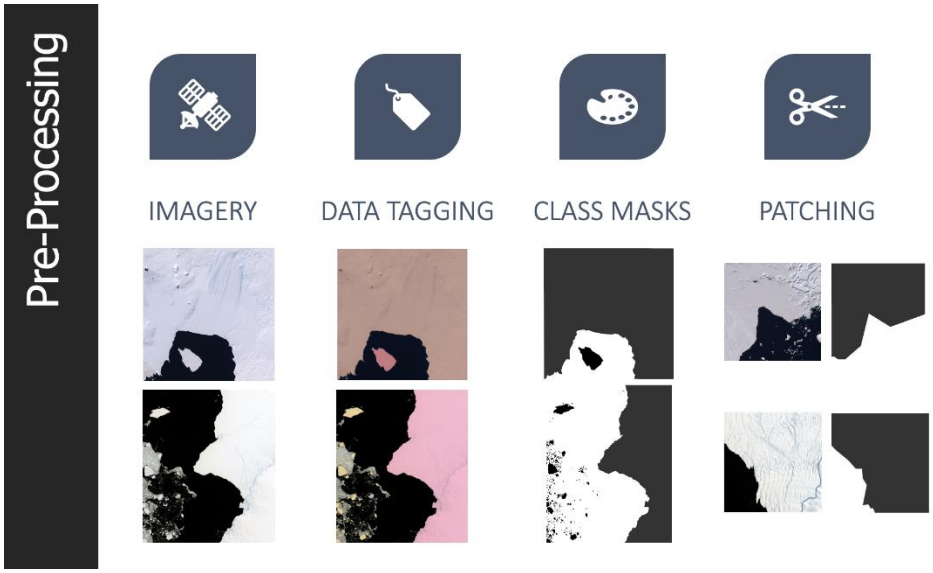


Figure 18: Graphic explaining the four pre-processing steps for the segmentation data set

4.2.1 Scene Selection

Data was taken from USGS Landsat Archive [6],we selected seven different visual spectrum images based on key considerations such as the resolution, which had to be very high; a sufficient variation of land, sea, ice and icebergs present across the images; and complexity of the satellite scenes designed to represent the variety of landscapes, ice structures and bodies of water of the arctic passages.

4.2.2 Data Tagging

These selected images are then geospatially tagged using an open-source Geographical Information System software called QGIS. This was done by creating a vector layer for each of the four categories we had decided to label:

Figure 19: Graphic explaining the different categories or classes for GIS Tagging

Using the polygon tool in QGIS, every entity that fit into these categories was captured and labelled in corresponding vector layers. These vectors (one for each class present) are then stacked upon one another to form a set for each satellite scene; this complete set was then converted to a shapefile. The tagged satellite scenes are shown below, with 11,000+ icebergs tagged, plus icesheets and land formations:

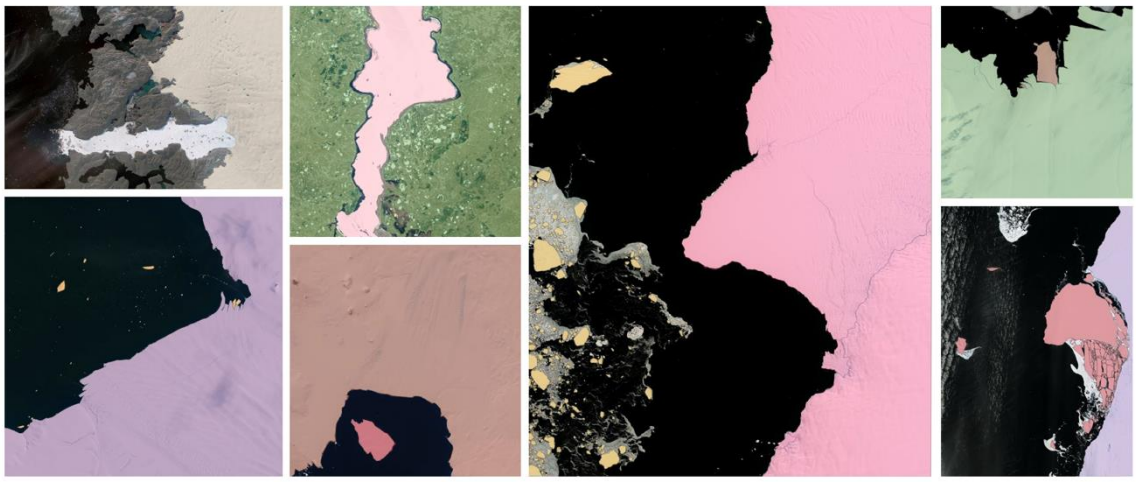


Figure 20: Collection of satellite scenes having been geospatially tagged according to classes

4.2.3 Class Masks

The shapefiles were then converted into greyscale masks - black and white PNG images where each shade of grey corresponds to a different class. This allows us to treat the class data as images, aiding us in the next step, the patching process.

This conversion to greyscale masks is done using the vector to raster function on the shapefiles, assigning uniform raster values less than 1 or equal to one for each class (such as 1 for icebergs, 0.8 for ice-sheets, 0.6 for land), using the rendering tools to make these new rasters display as black for 1 and shades of grey for other values, leaving the untagged sea as white. These are then exported as PNGs, now ready for the last step in data preparation.

4.2.4 Patching

Due to memory limitations of the computing resources available to us, our models could not process large satellite scenes all at once without downscaling and losing detail in our mask output. We also faced the issue of not having time to spare to manually tag enough images to sufficiently train the model.

The process of patching, slicing big images into smaller ones (in our case 500x500) to process them one by one, addresses both issues. In the first case, new images can be patched for the model to generate a mask for each patch, for those patch masks to then be combined into one large mask for the whole image. It addresses the second issue in that we need only tag a small handful of large scenes, and then we can patch those scenes alongside their masks into many

smaller images with corresponding masks on which to train our segmentation model. ~1000 patched images were created from the seven original scenes.

Figure 21: Diagram demonstrating the process of patching

The patching code was split into two scripts, which can be found in full in appendix D. Both scripts use the library Patchify, which patches and un-patches (re-combines) images, so our own code only needed to handle surrounding functionality.

The first script handles data pre-processing. It loads the large scenes alongside their masks using the Pillow library. It passes both through Patchify, which stores the patches of whole images in singular arrays. It then loops over every RGB and mask patch, exporting them to relevant folders. A counter, which ticks up for each patch, is used to divide the patches into train, test and validation at a 8:1:1 ratio (common best practice).

The second script patches a new image that we want to predict, puts every patch through our final model, and combines the predicted masks into one large mask. This was achieved by putting the new image through the patching function, looping over every patch, feeding the image through the model on each loop, and assigning the predicted masks to the relevant positions in an empty array of the same shape that Patchify accepts. The final array is put into Patchify's un-patching function, combining them into one mask, which is visualised next to the original image using Pyplot.

4.3 Segmentation Model Construction

The construction of the model utilised TensorFlow, Keras and the Segmentation Models libraries to establish a CNN. The U-net architecture, which was employed for our segmentation model, is structured as visualised below:

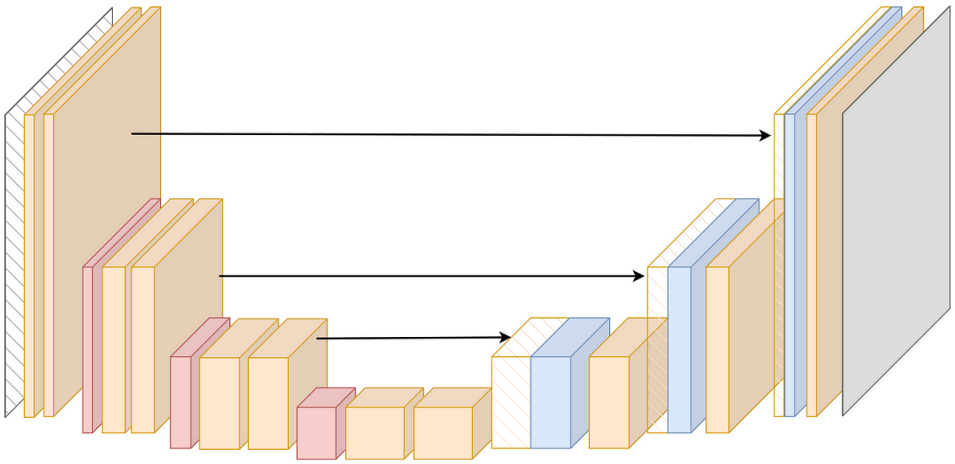


Figure 22: A visualisation of the convolutional layers of a U-net

The architecture is split into two phases, the encoding phase and the decoding phase. In the encoding phase, sets of convolutional layers extract features while periodically down-sampling, generalising a 'fuzzy' image of where each class is (in this case land, iceberg, icesheet or sea).

The decoding phase then takes this image and up-samples it, creating more precise higher resolution images from both the previous layer and information from layers on their opposite encoding phase layers of the same resolution (these 'skip-connections' are visualised by arrows in the above figure), building a pixel-perfect mask with the same resolution as the inputted image.

4.4 Segmentation Training and Validation

The training process involved presenting the network with batches of the prepared images, while systematically adjusting the internal parameters using the Adam optimizer based on the categorical cross-entropy loss function. Alongside training, a separate validation process was conducted with a distinct set of images. This step evaluated the model's accuracy and its ability to generalise beyond the training data.

Our data was enhanced by data augmentation, using the Albumentations library. Transformations such as flips, rotations, and noise were employed to bolster the data's size and variety. Pre-processing was then conducted to standardise the imagery, aligning with the EfficientNetB3 backbone's requirements. This step included normalising image data and resizing to the model's input resolution, thus optimising them for the training process.

The model's architecture utilised U-Net with EfficientNetB3, chosen for its balance of performance and computational efficiency. A softmax activation function was initialised for multi-class pixel categorisation. The model was compiled using an Adam optimizer, fine-tuned with a hybrid loss function that combines binary cross-entropy with the Jaccard index to weigh pixel-level accuracy against segmentation precision.

The model was trained over 30 epochs with a batch size of ten [7]. This combination was found to be best to use with the limitation we had with google colab. Custom Dataset and Dataloader classes streamlined data management and provision. Callbacks for model checkpointing and adaptive learning rate adjustments were deployed to capture the optimal model state and mitigate loss plateaus.

Post-training, the model underwent evaluation using a distinct test set. Predictive accuracy was gauged against ground truth masks. The results were visualised to display the original images, predicted masks, and accuracy overlays, providing a clear representation of the model's performance.

5 – Results

This section details the outcomes of the experiment, revealing the efficacy of the machine learning model through its interactions with designated training and testing datasets. The training dataset facilitated the development of the model's recognition capabilities, while the testing dataset provides a benchmark to evaluate its predictive performance.

5.1 Classification model:

In the context of neural network training, the accuracy metric serves as an indicator of the model's performance. It is calculated as the ratio of correctly predicted instances to the total instances in the dataset. The graph below visualizes the model's accuracy across the training epochs. Theoretical foundations in machine learning suggest that a model's accuracy should ideally improve incrementally with each epoch, as the model iteratively adjusts its weights to minimize error. However, practical scenarios often reveal varied learning rates and plateaus:

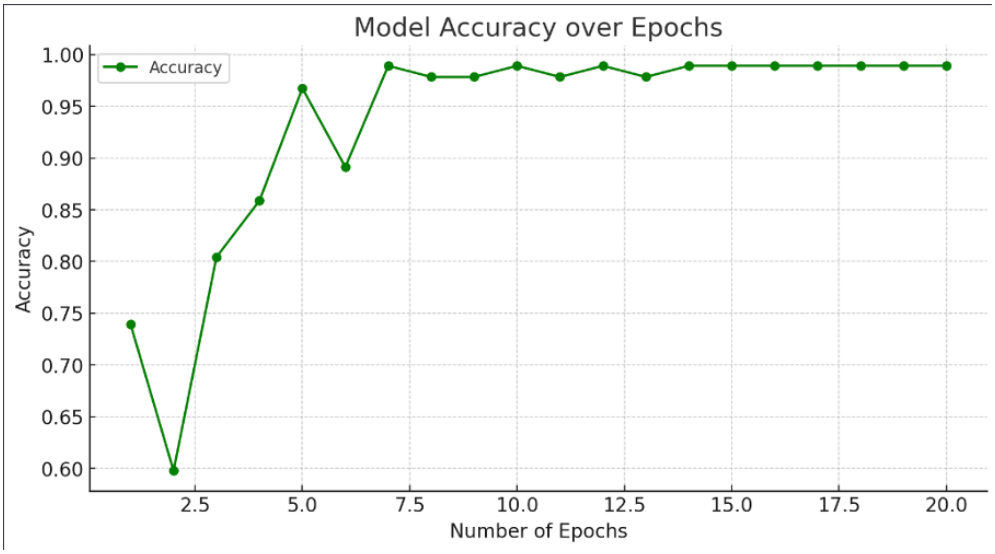


Figure 23: The training graph of the classification showing the Accuracy over epoch (using satellite data)

The training log shows that the model was trained over 20 epochs. Accuracy started at 73.91% and exhibited an improvement, reaching 98.91% by the eighth epoch and maintaining that level through to the twentieth epoch. This rapid improvement in accuracy suggests the model quickly learned the distinguishing features necessary for the classification task. However, the model also shows that it stopped learning after the seventh epoch. It indicates that further training after the tenth epoch does not add any value. This can be also observed with the loss function. In machine learning, the loss function is paramount, guiding the training of models by quantifying their prediction errors. It measures the difference between the predicted values and the actual labels. A synergy between these two metrics provides a comprehensive view of the model's learning progression.

The diagram below showcases the descent of the model's loss over successive epochs. A decline in loss is indicative of the model fine-tuning its parameters to align with the data's underlying

patterns. This graph, coupled with the accuracy, gives a holistic understanding of the performance during training:

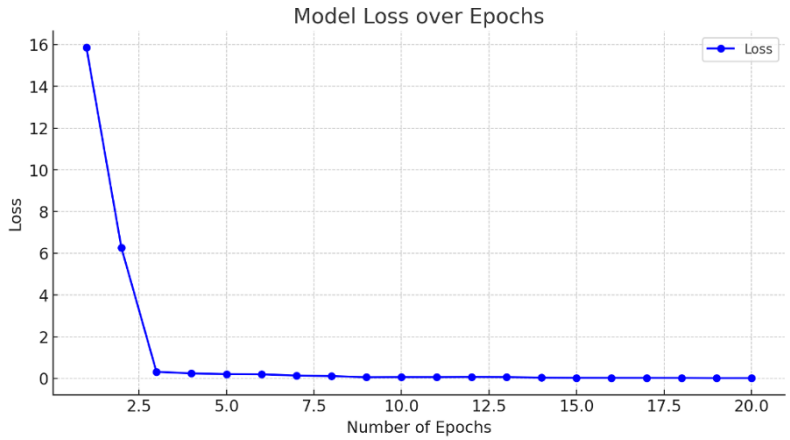


Figure 24: The training graph of the classification showing the Loss over epoch (using satellite data)

The loss decreased from 15.8588 to 0.0178 in 3 epochs. The loss function's rapid descent points to a convergence, which can be indicative of the model's high sensitivity to the training dataset's features. While rapid learning is good, it may suggest that the model is not developing a generalised understanding of the data, indicating overfitting. To test this hypothesis, we ran the model on 20 images not involved in its training process, which yielded 65% accuracy, supporting our hypothesis. We expect that overfitting occurred so rapidly due to the small dataset, having only ~100 training images. To test the model's efficacy without this issue, we tested on an alternative data set. The graph below illustrates the model's training accuracy over a series of epochs using an alternative dataset from the Dogs-Vs-Cats Kaggle challenge. This test serves to underscore the model's intrinsic capabilities when applied to a dataset with inherently simpler classification challenges compared to satellite imagery.

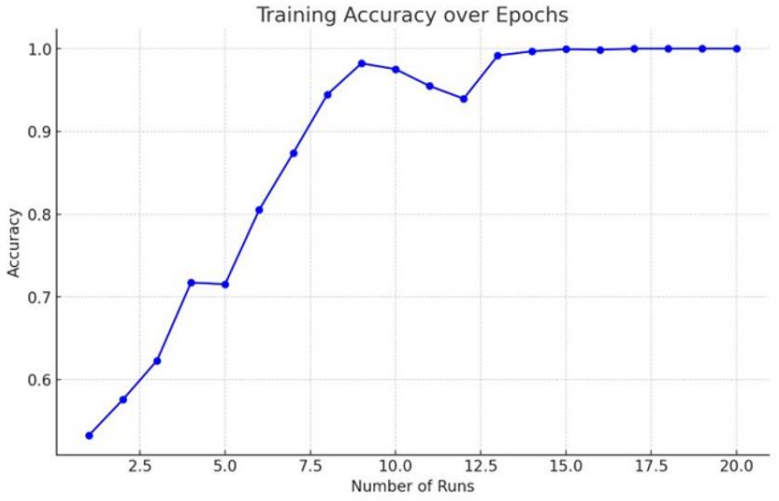


Figure 25: The training graph of the classification showing the accuracy over epoch (using Dogs-Vs-Cats data)

The graph displays the model's training progress over the epochs, which displays a noteworthy trend. The accuracy of the model increased sharply in the initial epochs and reached a stable plateau after 13 runs. This pattern of rapid learning followed by stabilization indicates that the

model is effectively extracting and learning the distinguishing features between the two categories. The plateau in accuracy indicates that the model has reached its optimal performance, which is higher than what was observed with the satellite images. This comparative analysis points to the limitations of classification model in terms of its complexity and the model's suitability for datasets with less complex images.

5.2 Segmentation model

The more complex segmentation model labels with pixel-precision rather than labeling a whole image as one class. A key metric in evaluating such models is the Intersection over Union (IoU) score, which quantifies the accuracy of the prediction.

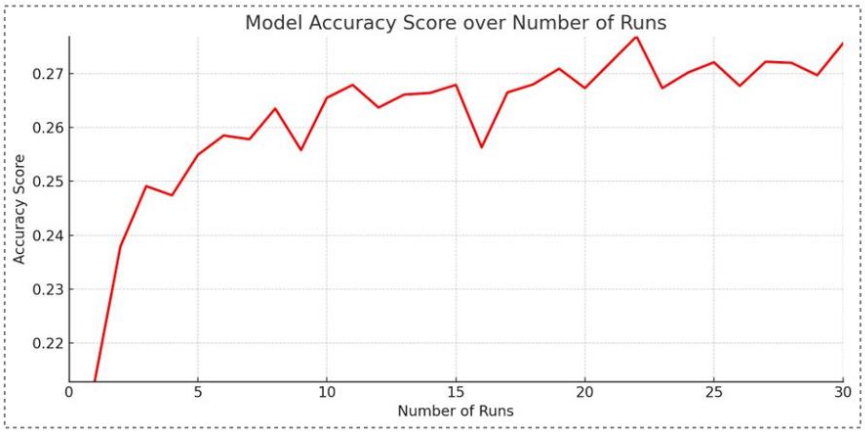


Figure 26: The training graph of the Segmentation Model

The graph visualises the model's accuracy over a series of epochs. An initial increase in the IoU score indicates successful feature learning from the data. However, the subsequent plateau at an IoU of approximately 0.27 points to a limitation in the model's ability to further refine its capabilities. This early achievement in accuracy demonstrates the model's potential, yet the levelling off suggests a need for further optimisation to enhance the performance and achieve more accurate delineation of the target classes.

In the process of developing our segmentation model, we benchmarked our training performance against an established example from ‘qubvel’ (Pavel Iakubovskii) on GitHub. The purpose of utilising Iakubovskii's work was to gain insights into the functioning of a segmentation model and set expectations for our training results. Iakubovskii's segmentation model achieved an IoU score above 70% with its respective dataset, which is considered a strong result.

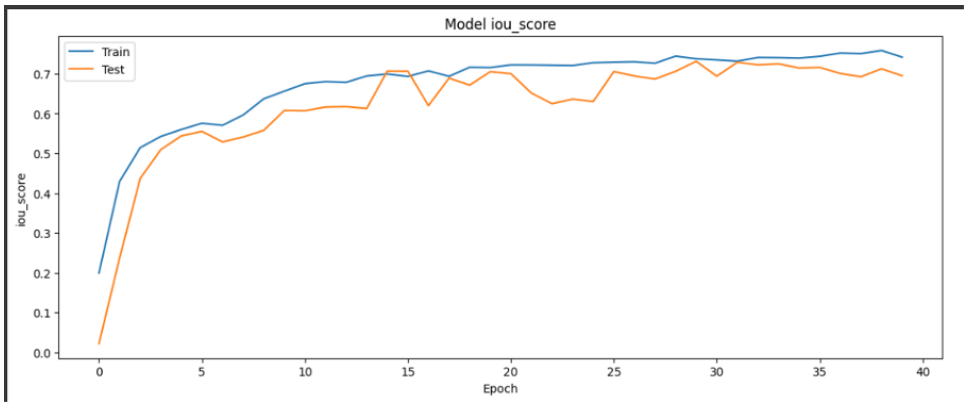


Figure 27: The training graph of a segmentation model example from GitHub

When comparing the result of qubvel's model to our model's performance (Figure 26), there are notable differences. Our model's accuracy is significantly lower than the 70% achieved in qubvel's example. Several factors contribute to this disparity, including differences in size and balance of datasets, number of augmentation techniques used, and the complexity of the segmentation task at hand.

The dataset used by qubvel's model was curated to match the specific requirements of the task, and the augmentations were tailored to enhance the model's ability to generalise across varying imagery. In contrast, our model's training data might not have been as balanced, with many patched images being entirely of one class or another, leading to a rapid convergence and a plateau in learning (a false minimum in the cost function), which is evident after the initial rise in the IoU score. This rapid convergence may have resulted in over-predicting the most common class, possibly icebergs, and failing to learn to detect multiple classes in an image, due to the imbalance in the training data.

6 – Discussion

6.1 Approaches & Project Management

Across the project, several key decisions about our approach led us to our results and a successful project completion. Initially we set out a project framework which took longer than expected, however proved essential to organising our activities and keeping us on track for almost all of the project. We also split up into specialised programming, data and business teams. Although limiting our programming capacity, this allowed a lateral approach to development whereby all three elements could be worked on simultaneously, a decision that saved much time, especially when combined with pair programming. Weekly summaries of progress also allowed us to gain greater input from supervisors and an extra avenue to our Gantt chart for keeping us accountable on the project timeline. Another key decision was deciding to pursue a classification model first and then a segmentation model, though more complicated and leaving less time to train it gave us a much larger scope for a more accurate and useful tool.

6.2 Data Science Discussions

Initially when looking for data to train and test the models we would build on, we explored existing curated data sets of or including icebergs such as NASA Earth Data GLIMS: Global Land Ice Measurements from Space [8] and the Landsat Image Mosaic of Antarctica [9]. However, these were not as useful as initially thought. Not only were they Antarctic focused (little data exists for the arctic region), but resolutions were also too low for our application, and icebergs were not the focus of the data set so we would have to manually search. As such we pivoted to explore ESA Copernicus datasets, Pleiades data archives and USGS Landsat datasets [10]. We did find one promising dataset along the way, a Kaggle repository of iceberg and ship images [11], however this was solely radar data, which fell outside the scope of our project as it's processing would have been too complex. More radar data on icebergs was subsequently found, leading to suggestions it should be usedstat for any future development. Returning to the three visual imagery data sets, the USGS Landsat data yielded the best data for iceberg identification especially for the segmentation model. There was a larger number of images that were high resolution and arctic based, compared to very few archived data from other systems. However, for the classification model a simpler set of data was downloaded from the lower resolution Sentinel 2 data. From these decisions, we chose specific images and created our own in-house dataset, as we were unable to find existing ready-to-go datasets. In future experimentation, it would be expected that radar data (for its abundance and ability to see through cloud cover) and potentially hyperspectral imagery (which could potentially see through snow cover) should be used. The decisions on pre-processing followed naturally as we knew from the beginning data for segmentation must be tagged and class masked. Developing patching scripts to create more images solved a problem we were having of not finding enough suitable images but also not having the time to tag them all, we decided to develop a script to split up a few large, tagged images to create many training images.

6.3 Architecture Discussions

From our initial guidance, supplemented by our own research, we concluded that both our classification and segmentation models should be Convolutional Neural Networks (CNNs). This

is due to their flexibility in image processing types, saving time in developing both models, their power in detail extraction, allowing for the subtleties and contextual assessment needed to differentiate between surfaces that are largely covered in snow, and the widespread documentation of their functionality.

A downside to consider before settling on a CNN (or any other deep-learning approach) for our task, particularly for the more computationally complex segmentation model, was the computational resources that deep-learning demands. Particularly in the case of segmentation, there are less intensive methods that still see industrial use. As such, we considered the viability of traditional methods, because if we only produced something that could be achieved more cheaply with less computation, our product would not be viable.

So, the following methods of segmentation were considered:

Thresholding	Pixels are divided into classes by whether they meet a threshold in a value, particularly brightness in one or all colour channels. This could certainly discern sea from land, but snow on land will have near-identical pixel values to snow on ice due to them being of the same colour and reflectance.
Region-based segmentation	From a chosen pixel, the similarity of surrounding pixels is assessed, which are then assessed against their surrounding pixels, iteratively growing regions of similarity. This would have the same issue, of snow having indistinguishable pixel values regardless of the surface that the snow is on.
Edge-based segmentation	Edges are detected using convolution, similarly to what a CNN may learn for its first layers. This establishes the shape of formations in the scene but cannot consider them to assess the class of the objects they enclose.
K-means clustering	Pixels are grouped according to their positions in, for example, RGBXY space (multidimensional space accounting for pixel location, colour and brightness). This has the same pixel value problems relating to snow and requires pre-knowledge of the number of segments in the image, making it nonviable.

Figure 28: Table explaining various non-intensive methods of segmentation

The commonality of shortcomings of these traditional methods is their reliance on the colour brightness values of the space they are assessing to be put into a class, without considering the context around it. This means that it will not be able to distinguish between snowy surfaces, which all look much the same when not considering their shape or image context which a deep-learning solution should be able to account for.

So, we considered more specifically which convolutional architectures to use:

- For our classification model, we chose a contracting CNN, wherein the first layer takes every pixel value as an input, and as the layers progress, features are extracted and down-sampled (considered as lower resolution images generated from all the information from the previous larger layer), abstracting the layers until only two values remain, the probabilities of the images belonging to each class. This was chosen as it was a well understood model with extensive online documentation, relevant Python libraries and tutorial materials, making it a reliable option for which we could use online resources to investigate potential bugs, and ensure we have a working product in time.
- For our segmentation model, the more advanced final product, we knew we needed a powerful model with both a proven industrial record on semantic segmentation capabilities and sufficient documentation and open-source projects to reference such that our small team with a limited deep-learning background could develop a new model without issue.

The U-net met these criteria perfectly. It is a flexible architecture that is an industry standard for many image processing tasks, from intelligent image sharpening to AI image generation, but it was originally developed for medical image segmentation. This original purpose is technically akin

to our task, and the explosion of interest in the U-net thanks to both AI image generation and machine vision interest for self-driving cars has meant that there is extensive documentation and tutorial material for those who seek to develop U-nets.

6.4 Classification Model Discussions

The development of our classification model was characterised by its simplicity and the low computational cost. The model's development was cost-effective, making use of free resources without the necessity of hardware typically associated with machine learning tasks. Despite the small dataset used for training, the model achieved a correct prediction rate of 65%, demonstrating the viability of this type of classification model that employed a sequential convolutional neural network.

The initial results of the model had been promising, particularly when handling simpler images. By achieving a primary aim of displaying its utility, the model has proven competent in differentiating between clearly defined categories in less complex scenes. This success lays the groundwork for its application in controlled environments where the need for intricate object recognition is minimal. Moreover, the model's architecture presents scalability, allowing for the potential expansion into multi-class classification tasks. It holds promise for fine-tuning to recognize and classify a variety of objects, which is a testament to its adaptability for simpler tasks.

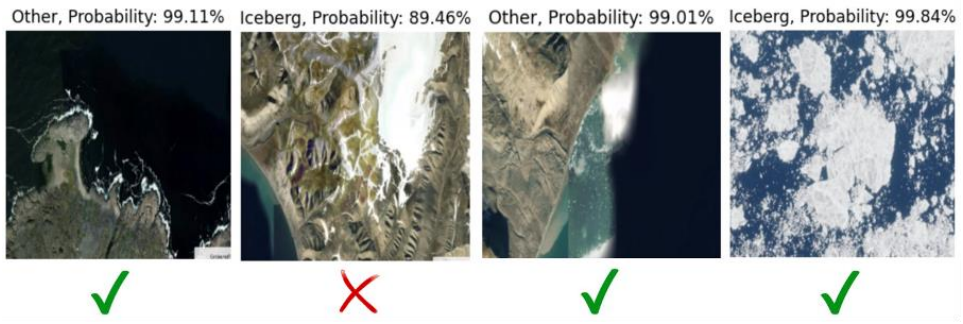


Figure 29: New test images with prediction of Classification model

The utilisation of satellite images featuring extensive and detailed scenes has posed a challenge for the binary classification model. The intricate patterns and the presence of multiple objects within these images were not always conducive to the model's learning process, which was designed to categorise into only two broad classes. This complexity, coupled with the relatively small dataset of ~100 images, contributed to overfitting. Consequently, while the model performed admirably on the training data, its performance on the test set was less impressive, with an accuracy of 65%. Though a good starting point, this is insufficient for industrial application.

This could be improved by expanding the dataset to provide the model with a more diverse array of examples, enhancing its ability to generalise and reducing the likelihood of overfitting.

To demonstrate the classification model's potential, we expanded our training and testing to incorporate an alternative dataset. We selected the Dogs-Vs-Cats dataset from the Kaggle challenge, which contains a substantial library of over 25,000 images. Given the dataset's simplicity and compatibility with our model, with a large selection of images of dogs and cats, this dataset was perfect to evaluate the model's scalability. We trained our model on a balanced subset, consisting of 1,000 images of dogs and 1,000 images of cats. This approach was inspired

by his 'approximate rule of thumb' from Pete Warden's blog—a key contributor to TensorFlow, the software framework utilised in our project.

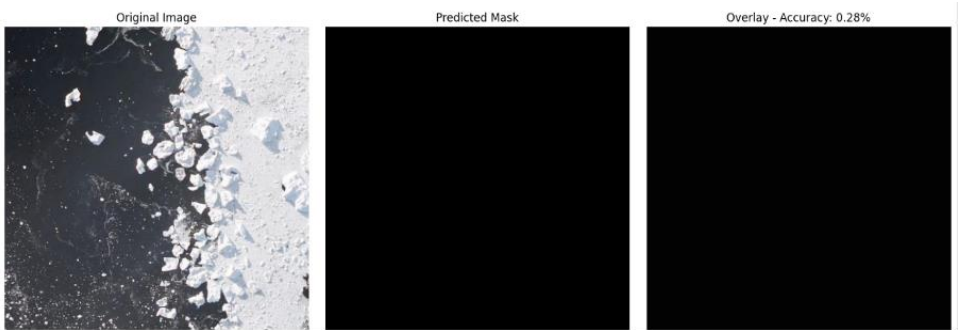
Upon completion of the training phase, the model got tested using a new set of 200 images of dogs and cats. The model correctly identified 76% of this test data, which is an improvement over the previous test accuracy of 65%. This supports the hypothesis that our model's performance benefits significantly from a larger and more balanced training dataset, though it also suggests that ~76% may be a hard limit on this model's accuracy.

Figure 30: Cat and Dog correctly identified by the classification model.

Moving forward, it is apparent that while the classification model has shown promise, there is a limit to its capabilities, especially when tasked with interpreting the complex and varied large-scene satellite imagery.

6.5 Segmentation Model Discussions

The segmentation model's training commenced with a strong learning trajectory, as indicated by the initial increase in the IoU score (Figure 26). Such an uptick was promising. However, the plateau in the IoU score beyond the initial learning phase raised concerns. The model's ability to further refine its understanding and improve its predictions didn't occur and its predictions were incorrect (example below). The images depict a stark contrast between the original satellite imagery and the predicted masks, with the model failing to accurately segment the different elements within the scenes. The overlay accuracy percentages, near 0% in some instances, are clear indicators that the model struggled to generalise its learning to new data.



5

Figure 31: The output of the testing code of the segmentation model. This shows that the model did not learn effectively from the training as all masks are dark grey (icesheet).

The development of our segmentation model encountered learning hurdles. Despite an initially promising increase in the IoU score, the model soon plateaued, a clear sign of overfitting to the predominant class (icesheets) due to an imbalanced training set. This was evidenced by the model's inability to segment new images effectively. To improve the model, addressing the data imbalance is critical. Proportionate class representation and augmentation reflective of real-world variability are necessary to enhance generalisation. Advanced regularization and hyperparameter optimization could further prevent overfitting and refine the learning process.

The fact that the model always predicted one class for the entire image also indicates that too many of our training images were similarly monotonous. This could be mitigated by filtering the dataset with thresholding methods to include only those with some variety in colour, such that those of entirely one class are usually excluded.

The model's computational demands also exceeded expectations, attributed to the complex U-Net architecture. The constraints of Google Colab's free GPU, including the 15GB ram limit, hindered our ability to iterate and optimize, especially as the deadline loomed. For future improvement, allocating more time for training and debugging would be essential.

6.6 Market Research

In the initial phase of our market research, we focused on key navigable sea paths, albeit with associated safety concerns and underutilisation despite considerable geographical advantages in terms of shorter distances and expedited transit times. Subsequent research revealed the Northern Sea Route (NSR) and the Northwest Passage (see appendix C) as pivotal maritime corridors underutilised due to ice risk, presenting lucrative prospects for shipping enterprises. We conducted methodical online research on these shipping companies looking at the composition of their fleets. Despite expecting a straightforward process, this proved challenging, with little information being publicly available. So, we attempted communication with the relevant companies via email. However, this yielded no responses after an extended waiting period. Confronted with this predicament, we consulted our supervisors, and a decision was reached to derive estimations based on publicly available annual traffic data along both routes. This pivot facilitated the acquisition of more extensive numerical insights.

The fact that the routes opened up by enhanced ice navigation bypass two major canals with publicly listed tolls also provided a source of more concrete financial values with which to calculate savings.

The shortcomings of our market research were largely a result of limited informational availability. This could be mitigated were this project to be handled directly by a well-established company like Airbus, which has the connections and influence to contact shipping companies for the relevant information, with the companies having confidence that the information is being used for a project that will be to their benefit.

6.7 Proposal for Further Development

Based on our experimentation and market research we developed a proposal to Airbus Intelligence. Given the relative success of our model given limited training, we propose continuing with the segmentation model, with some additions:

- More training data. Using Airbus's own data, we suggest optimising the CNN for Synthetic Aperture Radar (SAR) or even hyperspectral data to avoid problems with cloud and snow cover and to extract more useful information.
- Rebalance training data for more variety, using thresholding methods to filter for more diverse images.
- Modify the model to recognise scale.
- Implement a system for patching and un-patching shapefiles to save on data processing and to yield shapefile outputs.
- Produce heat map forecasts of iceberg probability as a subscription service for shipping companies.
- Given sufficiently accurate segmentation results, produce iceberg path-prediction using positional and shape information.
- Extend to a route generation tool able to integrate with google maps API, existing industry tools, or a custom solution.

We expect a team of two developers to be able to complete these tasks within 6-8 months at a total labour cost of £110-120k.

Based on our market research, we believe this to be an incredibly attractive proposition to potential customers which Airbus could expect to charge up to £250k+ annually. Spin-offs and secondary use cases are also expected. As such this would be an eminently suitable project for Airbus Intelligence who are already exploring arctic routing and iceberg tracking.

7 – Reflections

In this section we reflect on the project as a whole but also on how we developed as a team and as induvials over the course of the project activities:

7.1 Overall Development

Throughout the project the team constantly learnt new knowledge and developed new and existing skills. One of the primary skills we developed was our organisation. Evidence of this is the fact that early in the project organising our weekly meeting was incredibly difficult due to everyone in the team having different schedules. However, we implemented systems to alleviate this problem and improved communication.

We also greatly developed our communication skills. Over time, we got more comfortable with sharing our workload and delegating tasks amongst the team and completing these tasks individually before sharing our progress with each other as well as the supervisors and external partner each week. One of our targets to aim to improve on was to communicate our progress in a positive light. For example, one week, we had told the supervisors that we felt our progress was slow, however we were told the opposite and that we had clearly been very productive; it was up to us to shape the way our 'audience' received the information we gave them. Much of this was the ability to deliver our message with confidence and cohesion with the rest of the team. This is something we actively worked on by changing our style of presenting our progress in the latter half of the project, as well as rehearsing our final presentation together as well as with an audience member to give us feedback on what went well and the few things we could still refine.

We started the project, with little technical skills, and have now become competent in programming complex neural networks, manipulating satellite data, industry level market research and confidently presenting our work to professionals.

7.2 Individual Development

Altamush: This being my first long term project meant that I had no experience of project management or what the term even meant, so overseeing making the Gantt chart and keeping track of the groups progress was a skill I have developed in this project and hope to work on more. In addition, keeping a record of everything that was talked about in the meetings was something I found challenging initially but ultimately beneficial as it tested my active listening skills heavily. I wasn't as confident as I wanted to be in the presentation, so I hope to develop my direct communication skills further in the future.

Hamza: Prior to this project, having completed a short machine learning course on Udemy had helped me gain some basic knowledge. Since then and having completed this project my experience and technical skills have significantly advanced. I notably improved in programming and machine learning, particularly, by developing our classification model, devising the training scripts for our architecture, and collaboratively pair programming the segmentation model with Jonathan. Moving forward, a skill I am keen to hone is my ability to present and communicate complex concepts and ideas with clarity and confidence to large audiences.

Jonathan: Having had minimal computer science experience before coming to university, this project was an immense leap forward for me in my technical ability. I'm particularly proud of my progress in data handling, coding and understanding of convolutional architecture that I developed while working on the patching scripts, doing research for our architecture, and pair-

programming our segmentation model with Hamza. I'd like to develop my coding skills further, so that I can code complete CNNs independently.

Joshua: Having completed similar projects in the past, albeit on a much smaller scale I was able to further develop overall project organisation skills. This project's industrial nature also allowed me to improve professional communication skills, both in regular emails and meetings with supervisors and partners and in our final presentation. In addition, the technical skills I learnt in GIS, remote sensing and data science will be invaluable to my intended career path; as such I hope to further improve my research and data analysis skills.

Ammar: During this project I developed my ability to work on multiple things at the same time, which was necessitated by delays in shipping companies replying to emails. This alongside coordinating meetings with the team helped my time management skills. I hope to continue building on these organisational skills as I develop my career.

7.3 Next Steps

As a team we are looking forward to taking the skills we have learnt in this project into our individual specialist research projects as part of our 3rd year course. In particular, project organisation skills will be of great use, and we intend to more confidently portray our work going forward. These experiences will be of great benefit to our CVs, future interviews and employability.

8 – Conclusions

To conclude, we have identified a useful, sustainable and lucrative application of computer vision and built a sound financial case for our solution. The proposal aligns well with UN sustainable development goals (see appendix A) and is highly achievable for Airbus Intelligence.

Informing this proposal, we have developed a working classification model which is 65% accurate. With the industry standard success rate of 60% to 90% depending on use case, the model was a success. The main avenue for improvement would be the very small training data set, that caused the model to overfit within only a few epochs. With a larger dataset, the model would be appropriate and quickly ready for integration into commercial navigational use, only needing additional code to identify candidate images such as through thresholding and edge detecting for white regions.

We also made progress on a much more complex segmentation model, reaching 27% accuracy. We identified its shortcomings, and identified the changes required to its training dataset. A more balanced dataset with more images of different classes, achieved using thresholding for images with colour variety would drastically improve accuracy, such that a future build could be integrated into software that predicts the path of icebergs based on shape and other factors potentially extracted from hyperspectral data.

With the help of our project management framework, project aims were met across the board with all three requirements for each neural network achieved and only time limiting improvements that could be made. Such improvements have been thoroughly explored and future solutions proposed. Overall, this project has been an invaluable experience for all team members and has been thoroughly enjoyable. We have developed many skills in great demand and grown significantly as scientists while getting a taste of industry and business too.

References:

1) Brownlee, J. (2019). *How to Develop a CNN for MNIST Handwritten Digit Classification*. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/how-to-develop-a-convolutional-neural-network-from-scratch-for-mnist-handwritten-digit-classification/ [Accessed 4 Oct. 2023].
2) QGIS (n.d.). *QGIS Training Manual — QGIS Documentation documentation*. [online] docs.qgis.org. Available at: https://docs.qgis.org/3.28/en/docs/training_manual/index.html [Accessed 4 Oct. 2023].
3) Statista (n.d.). *Annual ship traffic in the Suez Canal 1976-2019*. [online] Statista. Available at: https://www.statista.com/statistics/1252568/number-of-transits-in-the-suez-cana-annually/ [Accessed 20 Oct. 2023].
4) European Space Agency (n.d.). *Search on Earth Online*. [online] earth.esa.int. Available at: https://earth.esa.int/eogateway/search?category=Data&data_type=data+description&sortby=RELEVANCE&thematic_area=sea-ice%2Cglaciers-and-ice-sheets [Accessed 21 Oct. 2023].
5) Terrascope (n.d.). *Open-source satellite images? Well spotted!* [online] terrascope.be. Available at: https://terrascope.be/en [Accessed 21 Oct. 2023].
6) United States Geological Survey (n.d.). *Landsat Collections in Earth Engine | Earth Engine Data Catalog*. [online] Google Developers. Available at: https://developers.google.com/earth-engine/datasets/catalog/landsat [Accessed 22 Oct. 2023].
7) Warden, P. (2017). *How many images do you need to train a neural network?* [online] Pete Warden's blog. Available at: https://petewarden.com/2017/12/14/how-many-images-do-you-need-to-train-a-neural-network/ [Accessed 21 Oct. 2023].
8) NASA (n.d.). *GLIMS: Global Land Ice Measurements from Space*. [online] www.glims.org. Available at: http://www.glims.org/ [Accessed 14 Oct. 2023].
9) United State Geological Survey (n.d.). *Landsat Image Mosaic Of Antarctica (LIMA): Index Page*. [online] lima.usgs.gov. Available at: https://lima.usgs.gov/ [Accessed 19 Oct. 2023].
10) United States Geological Survey (n.d.). *Landsat Collections in Earth Engine | Earth Engine Data Catalog*. [online] Google Developers. Available at: https://developers.google.com/earth-engine/datasets/catalog/landsat [Accessed 19 Dec. 2023].
11) Statoil/C-Core (n.d.). *Statoil/C-CORE Iceberg Classifier Challenge*. [online] kaggle.com. Available at: https://www.kaggle.com/c/statoil-iceberg-classifier-challenge [Accessed 10 Oct. 2023].
12) United Nations (n.d.). *How to achieve Sustainable Development Goals*. [online] The Global Goals. Available at: https://www.globalgoals.org/take-action/?gad_source=1&gclid=Cj0KCQiAnfmsBhDfARIsAM7MKi2a4Zer8OTdgOLCZBe4qWhYZKi1Mx7ILltlqX_jli1o3n09UWsGM8AaAiIDEALw_wcB [Accessed 10 Nov. 2023].
13) European Union Commission (n.d.). *Reducing emissions from the shipping sector*. [online] climate.ec.europa.eu. Available at: https://climate.ec.europa.eu/eu-action/transport/reducing-emissions-shipping-

sector_en#:~:text=In%202018%2C%20global%20shipping%20emissions [Accessed 11 Nov. 2023].
14) International Maritime Organization (2020). *Fourth Greenhouse Gas Study 2020*. [online] www.imo.org. Available at: https://www.imo.org/en/OurWork/Environment/Pages/Fourth-IMO-Greenhouse-Gas-Study-2020.aspx [Accessed 11 Nov. 2023].
15) Feingold, S. (2023). *Panama: Drought at the Panama Canal disrupts global trade | PreventionWeb*. [online] www.preventionweb.net. Available at: https://www.preventionweb.net/news/unprecedented-challenges-drought-panama-canal-disrupts-global-trade [Accessed 10 Nov. 2023].
16) González, A. and Paugam, J.-M. (2013). Climate Change and International Trade. *International Journal of Climate Change Strategies and Management*, [online] 5(3). doi:https://doi.org/10.1108/ijccsm.2013.41405caa.009.

Acknowledgements

The team would like to thank our supervisors, Prof. Ian Hutchinson, Dr. Melissa McHugh and Dr. Hannah Lerman for their support with this project, as well as our external partner from Airbus Intelligence, Dr. Liam Harris, whose advice and support has been invaluable.

Thanks also to Pavel Iakubovskii (‘qubvel’ on GitHub) for his excellent Segmentation Models library, documentation and examples, which greatly aided us in the segmentation portion of this project.

Appendices

Appendix A: Climate & Equality Considerations

Opening up trade, making Atlantic-Pacific trade more accessible is of major importance to the global trade industry but it also has important implications for the global climate crisis and socioeconomic development. These considerations are included in this appendix.

Equality of development

Panama solely owns and operates the Panama Canal, providing a significant contribution to the country's revenue. Redirecting trade routes from a significantly less well-off country to a more developed country is a potentially complex affair. The proposed solution is economically beneficial to shipping companies (the intended customers) and the global economy but as more ships move from the Panama Canal to the Northwest Passage, economic complications could arise for an already struggling Panama. As the proposal overall is already in significant alignment with UN SDG's we strongly recommend significant discussions are had on how to address issues of equality and development should this proposal be implemented.

Sustainable development goals

UN Sustainable Development Goals (SDG's) have become an important metric and consideration for any project, industrial, academic, governmental or otherwise.

Figure 32: Graphic of the UN Sustainable Development Goals [12]

The project aligns well with SDG's 8, 9, 12 and 13. The project supports economic growth mentioned in SDG 8 (although elements must be assessed in order to not endanger economic equality of growth) as well as innovation in industry (SDG 9) driving a more efficient safer way of supporting the backbone of global trade. Likewise driving down excessive fuel and even

freshwater consumption (SDG 12), all this feeding into every companies' responsibility of climate action in their operations.

In 2018 global shipping 1.076 billion Tonnes of CO2 (around 3%) of the entire world's emissions, projected to keep on increasing. Attempts to reduce fuel consumption, improve shipping efficiency and drive down emissions have been identified as paramount to reducing the transport industries contribution to the climate emergency [13].

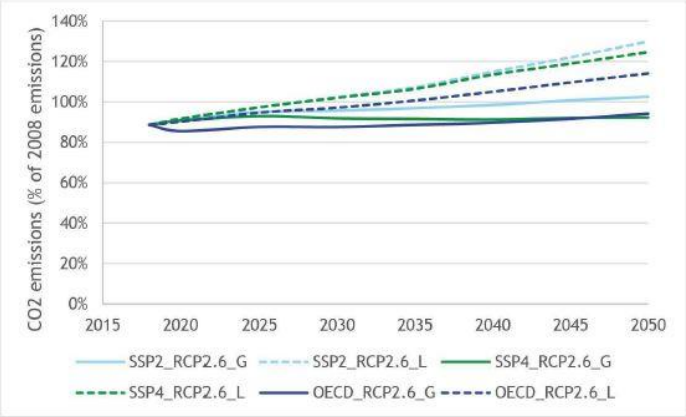


Figure 33: International Maritime Organisation graph of projected emissions due to shipping [14]

Case study: The Panama Canal

In 2023 the Panama Canal Authority (ACP) reported an environmental “crisis” with “no historical precedence”. The global climate crisis has been causing droughts in Panama, critically limiting the amount of water available to fill the locks allowing transit of the canal. Each of the twelve locks require 101,000 m^3 to fill just once. The United Nations Office for Disaster Risk Reduction (UNDRR) reports that transit wait times have increased from serveal hours to several weeks, transit fees have had to increase and shipping companies are also implementing surchages to their customers [15]. The situation is not sustainable, with few options to improve beyond engineering upgrades.

The World Trade Organisation goes further, saying “Higher temperatures, rising sea levels and more frequent extreme weather events bring the prospect of productivity losses, production shortages, damaged transport infrastructure, and supply disruptions.” [16] Our proposal to reduce these risks to companies and economies by opening up the Northern passages, as well as reduce fuel consumption and emissions will be beneficial on all these metrics.

Appendix B: Market Research

Calculations

- The amount of fuel used for 1 day or 24 hours of travelling by an average sized container ship is 63,000 gallons.
- The speed of the average ships is 46kmh, and the use of the Northwest Passage shortens the journey from the same start and end point by 7000km.
- Therefore 7000/46 = 152.17 hours of journey time saved each trip.
- 152.17/24 = 6.34 days of journey time saved each trip.
- 6.34 days x 63000 gallons = 399456.5 gallons = 1,815,965.95 Liters of fuel saved on each journey.
- Each Liter of fuel costs £1.17
- Therefore, 1,815,965.95 x 1.17 = £2,124,680.16 saved each time a journey is made through the Northwest Passage.
- Image average cost = $30 per km^2 x 2,272,613 km^2 (total covered area) = $ 68,178,390
- anama Canal Container cost = 1,450,000 (cost of one ship) / 15000 (The average of containers number per ship) = 96.4$
- Image Cost per container = $4,870 (Image cost per ship) / 15,000 (average containers per ship) = $0.33

The Northern Sea Route

The Northern Sea Route allows access to Europe from East Asia whilst bypassing the Suez Canal and thereby reducing the distance which needs to be travelled by ~8,000 km. This makes the journey shorter and therefore quicker and more affordable due to savings in fuel costs, tariffs and insurance through the Suez Canal. Much less fuel will also be used, so it will be positive environmentally.

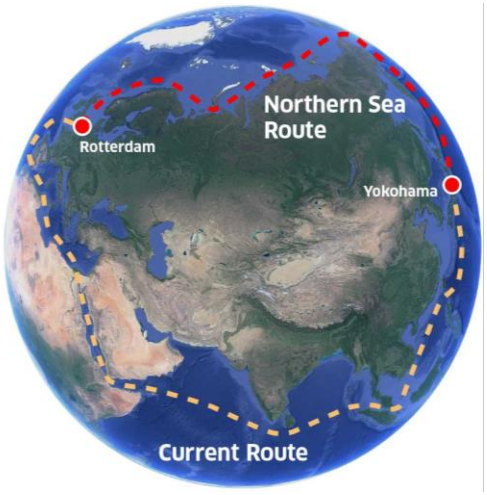


Figure 34: A diagram of the Northern Sea Route and the current route through the Indian Ocean and Suez Canal

The Northwest Passage

The Northwest Passage is a shorter route from Europe to the western coast of North America than through Panama, saving money on fuel and tolls. This passage is becoming increasingly navigable as more ice melts, opening possibilities for expanded trade across the Atlantic. This route too would use less fuel, so the environmental impact would be positive. This report focused on the Northwest Passage as the primary training ground for the CNN.

Figure 35: A diagram of the Northwest Passage in comparison to the current route taken through the Atlantic Ocean and Panama Canal.

Tolls for passage through the Panama Canal are determined by factors such as vessel type, size, and cargo. The canal plays a crucial role in global trade and is a key conduit for international maritime transportation.

Appendix C: Project Management

Gantt Charts (Intermediate Iterations)

Meeting Minutes

Supervisor Meeting - 06/10/23

People: All

Notes: Training discussed, given specific sections of training

- Neural Network Training
  - Do you understand how the model is working?
  - If you give it your own handwriting how would that work?
  - If we gave it satellite data, how would we need to change it?
- QGIS
  - 2.1 and 2.2 Overview & layers
  - 3.1 Vector attribute data
  - 5.1 New vector data set
  - 7.1 Raster data

Actions:

- Everyone reviews training material before next supervisor meeting where we'll discuss progress

Team Meeting - 10/10/23

People: JP, AK, AA, HH

Notes:

- Discussed training so far, consensus was QGIS was straightforward but having some issues getting the python algorithm to work.
- Discussed potential problem areas.
    - Flooding risk
    - Air pollution tracking
    - Others such as ice levels, rockfalls, etc.)

  Consensus was on flooding due to large scale, widespread issue and large data sets.

- Discussed roles and how to split up tasks. Waiting until we have a clearer idea of deadlines and tasks before specific allocations.

Actions:

- Make sure python and QGIS is downloaded. Everyone continues Training.
- Discuss TensorFlow issue with Melissa
- Discuss potential problems with Melissa and External
- Clarify expected deadlines at supervisor meeting
    - Once this is done, assign tasks, discuss roles further and finish planning.

Supervisor Meeting - 10/10/23

People: JP, AK, JM, HH

Notes:

- Discussed expectations and assessment
    - Overall progress mark is assessed in the supervisor and external meetings only. These meetings are very important, and we need to be well prepared.
    - We should set up a clear agenda for each meeting weekly and organise with supervisors.
    - During the meeting we should present our progress, ask any questions, take minutes and record any answers. Meetings don't have to be for a whole hour, we can meet before as well.
    - Recommended to do a team debrief afterwards without the supervisors.
    - External meetings should also take place every week.
- Organising a meeting with external this week
- Discussed roles and ideas
- HH & JM suggested a team meeting tomorrow

Actions:

- Everyone work out when we're free, come up with three options for Thurs/Fri
- JP send a draft email for External to Melissa by end of day
- Roles assigned once we have a better idea of the task at hand
- JM, AK & JP add slide on ideas to initial presentation
- HH organise room for meeting tomorrow

Team Meeting - 11/10/23

People: All

Notes:

- Email sent to External, waiting on response
- Discussed more ideas on project aims/objectives. Noted on the PowerPoint to be presented to External
- Discussed deadlines (see actions)
- All went through coding software: HH & JP got tutorial working on VS Code; AK & JM to try on Uni computers; HH & JP tried helping AA but weren't able to solve the problem. AK, JM, AA will try once more and if not will ask Mellissa for help.
- Discussed risk assessment, set up documents for Aims/Objectives & Action Plan. All to be finished for Monday ahead of Supervisor meeting.
- Discussed weekly meeting patterns, factoring in part time work and Uni commitments.

Actions:

- AA finish adding personal bio to PowerPoint [by Thursday]
- JP create agenda for next supervisor meeting [by Monday]
- JP book rooms for External meeting [Today (Wed)]
- AA organise weekly team meeting and find times for supervisor meeting [by Monday]
- Everyone finishes training where possible [by Friday]
- AA, AK, JM try setting up python with TensorFlow, if not organise time with Mellissa for help [Today (Wed)]
- Aims/Objectives & Action Plan [by Monday]
- AK finish section on Ideas PowerPoint [by Thursday]

External Meeting – 12/10/23

People: All

Notes:

- Did introductions
- Went over understanding and making sense of satellite data
- Majority of project time will be using data to train the AI
- Deep learning will be prevalent in the project – supervised learning
- https://scihub.copernicus.eu/

- Kaggle can also be used for training, as well as the Tensorflow website
- Discussed project ideas
- RsAA – may not have access to adequate data
- FRA – scope may be a barrier
- APT – what do you want the AI to do with that data?
- AIL – useful but huge – ice detector? Tell the difference between water/ice/rock
- SST - difficult
- Use existing data sets to train the AI to learn what the data 'means'
- Find the limits of what can be assessed with the tools and data available, 'test the waters'
- Extrapolate to see how much data will be needed from small scale to large

Actions:

- Finish trainings (ASAP), try Kaggle and TensorFlow training if Python didn't work
- Contact Liam/Melissa about Python difficulties
- Decide on project plan/ideas

Team Meeting – 13/10/23

People: All

Notes:

- HH sent out 'Dogs vs Cats' training challenge.
- Discussed all ideas put forward to External.
    - Air pollution: Major issue but unclear objective
    - Flood risk: Viable proposal but risk of scope becoming too large
    - Conflict fires: Interesting and potentially useful but limited use cases and limited useful data available
    - Road space allocation: Could be a very useful tool but training data would be hard to find, requires very high resolution currently inaccessible.
    - Glacial/Berg Ice tracking: Large datasets readily available, manageable scope and useful end product.
- Decided to undertake Ice identification problem, using a neural network to analyses images and categorise between ice, water and rock. Discussed further use cases i.e.. Tracking glacial retreat and uses in shipping and navigation as discussed with External.
- Training coding issues are still encountered, and solutions discussed.
- Action plan, project aims and objectives, plus risk assessment discussed and work assigned.

Actions:

- Everyone look at Kaggle training
- Everyone finish GIS training if not already [by Friday]
- Set up meetings for next week (Team meeting, supervisor meeting, external meeting)
- AA set up action plan document, everyone populates with tasks, AA collate
- JP set up Project aims document, everyone populates with thoughts, JP collate
- AK and JM finish setting up MNIST environment

- All continue training

Team Meeting – 16/10/23

People: All

Notes:

- Aims and objectives were checked by all
- Action plan was populated and organised.
- Risk assessment
- Ammar taking over organising meetings, found times for next week's meetings.
- Main idea was to initially allow the system to recognise black/blue and white i.e., ice and water, and to build from there in simple steps

Actions:

- Look at potential data sets
- Assess resolution of imagery required

Supervisor meeting 20/10/23

Notes:

- Covered progress summary (this is very good)
- Went over ideas
- Think about the market – selling the product – financial value??
- Think about things as a recruiter – over deliver enthusiasm – recap in meetings – building relationships
- Storyboard the presentation, how you want it to go
- Measure the success of the meeting – consider external as part of the team, not too formal
- TLDR, Take it easy
- Problem, tools, usefulness, MARKET SURVEY/business analysis, project management, technical vs transferrable skills
- Presenting and communicating are important skills

Action:

- Ask Liam relevant questions later
- Make an active Gantt chart
- Glacial research [whoever wants to]
- Market research [altamush and Ammar]
- Look at data availability [hamza]
- Potentially contact interested parties[wwt]
- Plan next week meeting -

External meeting 20/10/23

Notes:

- Kaggle is good
- Went over ideas on how to begin project – classification vs segmentation
- Different satellites, different applications
- Data augmentation – rotating images, changing saturations, blurring etc.
- Data sets from everywhere – different parts of the world
- Model is an iterative process – may need to collect data specifically to teach the model what NOT to do
- Dataset very important – make sure data doesn't lead to dead end – don't only have water in dataset
- Use separate training data and test data

Actions:

- PLAN MEETING FOR NEXT WEEK

Supervisor meeting 24/10/23

Notes:

- Project review/ progress report
- Technical info, computing resources, data sets
- Northwest passage, northern sea route – uses of the product/ market value factors
- Potential customers/contacts – mostly academic, not so much industrial
- CNN – segmentation and technical approaches: semantic vs panoptic
- Don't depend too heavily on outside contacts, may be unreliable + not entirely necessary
- Quantify things visually i.e., green, amber, red
- Don't aim for perfection, aim to know what you're doing
- Learn from any mistakes, keep records for future reference – keep log of why certain decisions were made
- Use these notes as points in the presentation /report
- Realtime, presentation and report – skill development reflection

Actions:

- Gantt chart (Altamush, Ammar)
- Find out data requirements as a quantitative value (if possible)

External meeting 26/10/23

Notes:

- Went over technical information – google colab, Kaggle, how long it takes for program to run on GPU and TPU, storage use etc., data set research recap

- Shapefiles in QGIS to train algorithm – will need to be converted to be compatible with python – coco…
- Rasterising the shapefile – turning it into another image, feeding an image and a label into the program
- Keep test and train data completely separate from each other – validation data is separate from train data but still a apart of it
- Northwest Passage, and Northern Sea Route – potential uses and customers (shipping companies) https://www.sciencedirect.com/science/article/pii/S0165232X18304269
- Factors that affect market value + potential academic contacts
- Different types of segmentation – Panoptic vs Semantic
- Semantic is widely used, including at airbus – try find more about Panoptic models, work in a step wise way – simpler, more traditional machine learning, edge detection etc.
- U-net architecture as a segmentation model – how will it apply to this particular use case - initially made for medical use – looks for rare objects in data
- Flexible, use in automating vehicles
- Land cover, sentinel data, worth testing if it will translate well or not
- Take about 20% of total data as test data
- Test program in the area you want to sell to i.e., Northwest Passage – set the conditions under which the model will work, Antarctic data may not be as useful Arctic data in this case

Action:

- Gantt chart (altamush)
- Find out more about Panoptic models ( )
- Continue deeper market research (Ammar)
- Pursue u net algorithm further for end goal ( )
- Explore data sets ()
- Plan next week meetings

Supervisor Meeting 31/10/23

Notes:

- Went over project progress
- New data access / computing resources– google engine and google colab, Kaggle, ESA, google drive
- Neural network stuff – non deep learning segmentation, edge detection, thresholding, region based, clustering based
- Issue is that snow looks the same, therefore confident that deep learning is required to continue – semantic segmentation, extra add on's if things finish sooner than expected
- Recap of U-net architecture
- Gantt chart
- Train, test and validation data
- Market Research update – number of ships in five largest shipping companies connecting America with Asia and Europe

- Run any other external contacts by Liam first
- Went through next steps
- Which parts are running behind/ ahead of schedule, how far behind or ahead, how do we reach our goals – link progress to Gantt chart, identify problems and priorities
- Reflect on presentations and skills and external meetings

Actions:

- Go through engines and see access to data
- Wait for Google engine response
- Research how to get data into the neural network
- Further market research – number of ships, quantitative value of trade/cost, also look at southern route, comparisons
- Companies using the Russia route between Asia and Europe vs the Suez Canal
- Contact researched companies – maybe response but probably not :'(- Run this by Liam first
- Begin drafting report and final presentation

External Meeting 07/11/23

Notes:

- Market Research email – Fill in whatever he said about it
- Whatever else was covered in the first 10 mins
- Data Sets - Google Earth Engine access doesn't work with Uni or personal email addresses- Can we use Airbus data?
- Pleiades archive data through ESA – cant get anything useful out of it – depends on the archive used as polar regions are not as heavily covered
- Labeling our own data will likely be what we have to do but is very time consuming
- Testing functionality and how the model actually works
- Most people aren't interested in polar regions as much as cities/coastlines so there's not s much data available
- Drip feeding components and steady progression are good for project management
- Trying to create the AI using other data e.g., land use as it is easier to access, just to see if it functions and it can be understood how it is trained since its generally the same neural network
- Synthetic Aperture Radar Data on Kaggle – Ships and ice- this uses radar and is therefore weather proof and will have coverage despite conditions
- We are doing a good job so just keep at it B)
- Pair coding – may be a useful technique – working on your own can lead to hitting a wall and not making progress
- Don't worry about end of project just try to focus on what is happening, like learning to manually label data, small progress is better than none
- Political issues around Russia for the Northern Sea Route - everything is fine
- We will need deep learning to do this effectively
- Future applications for the scenario – hyperspectral

Actions:

- Ask IT services for help – may be able to get a useful email addresses
- Have a look at Sentinel 2 data set and Copernicus dataset – imagery, not tagged data
- See for data released by public organizations – British Antarctic Survey
- NASA Landsat – Mostly through Google Earth Engine
- Quantify effort i.e., took 3 people x time to do y, to estimate person hours for scaling
- Continue market research and data research, as well as final report and presentation drafts
- Check SAR data
- Next meeting?

Supervisor Meeting 7/11/23

Notes:

- Having a look to structure of the report from blackboard.
- Showing the report to different people and see if they understand the report (make the report clear)
- Going through the skills developed for everyone with some examples from our work.
- Market research: having a plan b or another strategy for getting the information from the company in case if we did not get a response from them through the emails.
- Focusing on the frame of the presentation and how to reach the audience to give them a good understanding as they are a people not a part in our team.
- Focusing in the achievements have been done. Present in a positive light.
- Making a deadline for the goal and a timeline for the process and the progress.
- Consider a presentation as telling a story, just to engage audience
- Basically were the best ☺

Action:

- Having a look to structure of the report from blackboard.
- Making a plan B or another strategy for getting the information from the company in case if we did not get a response from them through the emails.
- Focusing in the weekly achievements have been done and Present in a positive light.
- Continue market research and data research, as well as final report and presentation drafts
- Working and make a deadline

Supervisor meeting 14/11/23

Notes:

- Progress summary, data stuff and project documentation

- Play ourselves up instead of pushing ourselves down, in terms of language – difficult vs challenging, etc. – big focus on this
- Team building exercise – flesh out in report in terms of usefulness
- Data sets progress – Pleiades, Copernicus, Landsat, sentinel, terrascope WE FINALLY HAVE DATA
- We also have a working model
- SAR data as a Airbus proposal, as it will likely be more successful, but we don't have the resources to do that
- Think about why you are choosing what you are presenting, why do we need to share that data was challenging to get – makes the tone more defensive and makes it seem as if you've not done well
- How do we feel about where we are in the project?
- Try and create a sense of confidence, and get rid of uncertainty – think about problems along the way and how you will solve them
- Think about slides in terms of sharing the most important key data
- Progress isn't expected to be huge, just to be made – realistic expectations will lead to more confidence
- Technical progress review – take the existing material to see if the next part is feasible – the working model does 20 runs and has been able to recognise iceberg vs not an iceberg, however data set is not big enough for instant classification – try to quantify using numerical measures and show confidence using statistics – reference other models to show how far we are (1000 images but we only have 100)
- Focus on positives
- Market Research update – try to contact again, don't stop at the first hurdle – we are approaching the end
- We are particularly good
- Strengths – we are organised, good at delivering as a collective in effective ways- good communication -progress is being made in bringing more energy – skills, knowledge and confidence has all developed
- Remaining skill development – deliver with confidence as a whole – reflect on journey made as a team – 'selling' what we have done – show that you are proud of what you have achieved

Actions:

- Flesh out plan b – detail in report, also do basic research on the secondary use cases
- Once that's done, start on the final presentation too
- Tutorial segmentation algorithm to see if colab has enough computing power
- Change existing model as an upgrade

External Meeting 16/11/23

Notes:

- Market Research – no response, successful plan B – evidence of value
- How much should the project sell for
- Basic instance image classification model, ice vs not ice

- Don’t need to train the model each time, issues with extracting test images
- Look up Tensorflow, how to save and load models
- Tensorflow will automatically save to a folder and will capture everything including parameter changes and model architecture
- Training data shouldn’t need to be used every time, just the new test image
- Separate test script vs training and validation script, should be able to automate training, shouldn’t need to be done manually every time
- Whole image in classifier – split image into smaller tiles (python code is a pain), new images to label each new small image or pixel as ice or not ice
- Google colab struggling with computing time
- Creating a U-net segmentation instead of image classification – just to see how it does
- Resolution agnostic vs satellite resolution – down sampling each image to the lowest possible resolution
- Giving the model more context may improve the accuracy of the program, lower resolution but more stuff in the image may allow the model to understand what is happening
- Not much visual data on Icebergs
- Using Qgis to draw boxes and polygons around the ice and stuff, to label manually, and chop up the image instead of downscaling
- The fact that there is little data, means that no one has done it, radar is likely more widely used, however harder, but it is not affected by weather
- Can split each image into multiple/ making our own data
- Automated tools ?

Actions:

- Secondary use case research
- Savings in fuel costs research
- Maybe research the cost of computing resources, as well as data cost
- Try splitting images into smaller tiles
- Try to automate training so it doesn’t need to be done manually

Supervisor Meeting 21/11/23

Notes:

- Presentation as a story – 25% set the scene(who we are, what we did, approach), 50% specific mechanics(what we could do, market research, coding), 25%(performance and skills developed, final conclusions/proposal)
- Report same – start, middle, end
- Presentation best practice – style, what character are we playing – technical, serious academic, but have some fun, SHOW ENTHUSIASM.
- Rehearse things like delivery – enjoy what you are doing
- Be professional without being serious (try)
- Discussed challenges – computing time, technical problems, and solutions

- Skills review, show good evidence what we did to develop our skills – report. Just bullet point in report, not too much detail required
- Industry standards – flexibility (agile)
- Use colours to highlight key points
- Use fun slides to break up the seriousness of the other stuff
- Data set research, tagging, overall technical progress – classification, now saved the training so doesn't need to be rerun each time, we are now trying to improve accuracy of the model as it overpredicts icebergs in snowy images
- Segmentation, used outside code to run on google colab, and it looks promising, we can use colab as a viable model library, it may struggle with larger resolutions, but our scale is a bit smaller. 20 mins training 40 epochs using GPU.
- What is the purpose of the slide – be specific, and recognize your audience doesn't have any context, how does the audience know that our model is a good one and not a rubbish one, explain what things mean – keep the objectives clear, and don't do things for the sake of it, make sure it is useful
- Imply things as more positive than they necessarily are

External Meeting 23/11/23

Notes:

- Challenges with segmentation -computing limitations
- Basically a a bunch of the same stuff as the supervisor meeting
- Data set research and tagging
- Technical progress - classification model upgrades
- Try to prove that lack of data is the issue – outside code, or try training the model with different amounts of data and show the effects
- Change the batch size when using VRAM
- Resolution – on the ground distance, changing resolutions might help
- Splitting image and accompanying shapefile
- Probability map?
- Shapefiles are good
- Market research

Supervisor Meeting 29/11/23

Notes:

- Video is good
- Maybe include that clip into the presentation
- Progress overview
- Be more specific in skills development examples - show understanding of skills, e.g., AGILE
- Market research very good

- Panama canal running out of water, consolidates the market reason and is another reason to find another solution
- Rehearsal of presentation – use key information to get the point across, time may be an issue
- Good handover in presentation is important
- Challenges faced in GIS – I think this would be really good to write about in the report as it is also a limit of human ability (maybe image resolution)
- "lack of programming knowledge" = "playing to the teams strength"
- Classification output – show that it is right – make it obvious, assume the audience is clueless - dumb it down
- Segmentation code – preparation, processing, training, testing – flow chart, use the visual cues
- Try to simplify it for someone who has no clue what you are talking about – personify the model where you can – dumb it down
- Pros and cons – potential enhancements – I think this is really good for the discussion section of the report – dumb it down – say what things mean without context
- "icebergs look like snow and they look like clouds so we need it to say iceberg, not iceberg, not white or not white, as that doesn't help at all" – use analogies
- Next steps for segmentation model
- Data processing – cat
- What does airbus need as OUR customer
- Try to make the context accessible for anyone without any context or any knowledge on the subject

Actions:

- Plan rehearsal
- Continue the rest of the stuff as has been discussed

External Meeting 30/11/23

Notes:

- Market Research – cost, price of offer, and profits
- In reality airbus will know exactly the image cost
- What they don't know is the time it will take so maybe 1 slide on time cost, and a small section in the report
- Gut feel about how much someone will spend on the product – we have a per container price, so how much will it cost as a one off purchase or an annual subscription?
- UN news report is good, maybe include this into the presentation and report, probably doesn't need too much detail, but its just to further confirm the idea that this is necessary as the current formula isn't sustainable.
- Keep declaring the assumptions made as there may eventually be the point where you can make improvements
- Classification model output - improvement made from 50% to 64% correctness, from 14 to 20 images to train on, small number of images but still correct, indicates that with more data this could be really accurate and successful

- Look at famous open source classification data sets – MS coco, Youtube-8M, ADE20K, VisualQA
- As you add more epochs of training, eventually the performance will begin to drop, highly unlikely to happen to us
- Segmentation model, u-net architecture, satellite images with corresponding masks and shapefiles
- Backbone – pretrained encoding phase
- Pros and cons - same as last supervisor meeting
- Data processing, testing the segmentation model on cats, basically, splits big images then sticks them back together after analysis.
- Greyscale masks
- Explain each time you make a decision and try to do it as simply as possible – don't try to make it sound complicated
- What happens next – what would happen if we carried on the project for another semester, how would it develop
- Put minutes example in appendix
- Show first and last Gantt chart, rest in between can go into appendix

Supervisor Meeting 05/12/23

Notes :

- Cool clip shown iceberg stuff
- Progress summary
- Data set – 1000 original images split into train, test and validation, 400 tagged icebergs
- More data set stuff, greyscale masks, and
- Technical progress, segmentation model, still in early stages and trained for 3 epochs small batch size, slow learning rate, challenge of low accuracy
- Very high loss value for a segmentation model, but if there are more epochs we think that this will decrease and the model will improve
- Sick transition
- Details on improvements on the segmentation models, projected improvement rate improvements as epochs increase
- UN sustainable goals development. Our project fits well with 2 of these goals. Responsible fuel consumption. Panama considerations. Also, climate and socioeconomic benefits.
- Show better enthusiasm in the real presentation, consider it a performance, and act like it. Way over the top. A bit cringe but the games the game.
- Presentation stuff

External meeting 07/12/23

Notes:

- Progress summary
- Handover of source code github, kaggle – github is good for jobs

- Segmentation model going well – 1100 tagged icebergs
- Colour stuff, high contrast
- Data set
- Segmentation model progress, same as previous meeting – challenges and improvements and achievements
- We have high loss and low ioU. Increase training epochs and see what happens, filters and blurs, but don't do too much augmentation, as it can make the final product worse
- Can augment on disk, standalone instead of the training file/code
- More issues with the dataset. Masks not found. Files loading incorrectly. Data access permissions.
- Run a new validation set every time, this is just an idea to think about.
- To measure the loss and iou, it must be using a validation data set of some sort, running on that data, and giving a value. And uses this process to learn and improve itself.
- Long disk validation data set needs to bee coded slightly differently – file manipulation, standard python stuff but may take a while to learn.
- Training more epochs is suggested next step, as it will show whether the number of epochs is actually the issue or not
- The improvement will likely be more exponential, slow at first. As the model doesn't know 'how to learn' yet
- Changing the learning rate can also be useful,
- Reducing batch size can help use less memory or reducing the number of pixels, or the physical size of the images you use, can be affected by bad code, the images will also need to be closed, depending on which library
- The data being rubbish could also be the reason that it doesn't work well, this is unlikely but still a possibility
- Incorporating stops into the code to check and debug
- Putting a . somewhere in the filename can cause issue. The script thinks that anything after the . is the file ending
- Try doing things separately to see where the issue is in each instance – debugging
- UN sustainable development goals. Pros and cons.
- How much technical info in pres? Get across what been doing, but make it accessible for everyone, avoid jargon, but if you need to use it, then explain it – don't be fancy, but try to put as much important info as possible.
- Useful for interview, breaking down a specific problem instead of a general statement on problem solving
- Questions will be generic, but the answer should be as specific as possible, structured answer, logical
- Doesn't matter if they don't understand AI, but things like asking for help, breaking the problem into manageable chunks, solving each problem individually to see where the imnprovemtn was necessary
- Debugging is applicable to multiple situations – just general problem solving

Appendix D: Code

Classification Model

```
# Training Code
from google.colab import drive
drive.mount('/content/drive')

import cv2
import os
import numpy as np
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense

# Constants
IMG_SIZE = 512
DATA_DIR = "/content/drive/My Drive/Training/Icebergs/Train_Data"
LABELS = {"Icebergs": 0, "Other": 1}
NUM_CLASSES = 2

# Initialize arrays for data and labels
data = []
labels = []

for label, num in LABELS.items():
    class_dir = os.path.join(DATA_DIR, label)  # Adjusted to correct
folder names
    if not os.path.exists(class_dir):
        print(f"Directory does not exist: {class_dir}")
        continue
    for img_name in os.listdir(class_dir):
        img_path = os.path.join(class_dir, img_name)
        img = cv2.imread(img_path)
        if img is None:
            print(f"Failed to read image: {img_path}")
            continue
        try:
            img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
            img = cv2.resize(img, (IMG_SIZE, IMG_SIZE))
            data.append(img)
            labels.append(num)
        except Exception as e:
            print(f"Error processing image {img_path}: {e}")
```

```
data = np.array(data) / 255.0
labels = to_categorical(labels, NUM_CLASSES)

# Model architecture
model = Sequential([
    Conv2D(32, (3, 3), activation="relu", input_shape=(IMG_SIZE, IMG_SIZE,
3)),
    MaxPooling2D(2, 2),
    Conv2D(64, (3, 3), activation="relu"),
    MaxPooling2D(2, 2),
    Conv2D(128, (3, 3), activation="relu"),
    MaxPooling2D(2, 2),
    Flatten(),
    Dense(512, activation="relu"),
    Dense(NUM_CLASSES, activation="softmax")
])

model.compile(optimizer="adam", loss="categorical_crossentropy",
metrics=["accuracy"])

# Train the model
model.fit(data, labels, batch_size=32, epochs=20)  # Adjusted batch size

# Save the trained model
model.save("/content/drive/My Drive/Trained_Models/iceberg_classifier.h5")
```

```
# Testing Code
from google.colab import drive
drive.mount('/content/drive')

import cv2
import os
import numpy as np
from tensorflow.keras.models import load_model
import matplotlib.pyplot as plt

# Constants
IMG_SIZE = 512
MODEL_PATH = '/content/drive/My
Drive/Trained_Models/iceberg_classifier.h5'
```

```
TEST_DATA_DIR = '/content/drive/My Drive/Training/Icebergs/Test_Data'  # 
Update this path to your test data directory

# Load the trained model
model = load_model(MODEL_PATH)

def predict_image(image_path):
    img = cv2.imread(image_path)
    if img is None:
        raise ValueError(f"The image at path {image_path} could not be 
read. Check the file path and permissions.")
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    img = cv2.resize(img, (IMG_SIZE, IMG_SIZE))
    img = np.array(img) / 255.0
    img = np.expand_dims(img, axis=0)

    prediction = model.predict(img)
    predicted_label = "Iceberg" if np.argmax(prediction) == 0 else "Other"
    probability = np.max(prediction) * 100

    return predicted_label, probability, img[0]

# Iterate over each image in the test data directory, predict and display
for img_name in os.listdir(TEST_DATA_DIR):
    img_path = os.path.join(TEST_DATA_DIR, img_name)
    try:
        label, probability, img_array = predict_image(img_path)

        # Display the image with the predicted label and probability
        plt.figure(figsize=(3, 3))
        plt.imshow(img_array)
        plt.title(f"{label}, Probability: {probability:.2f}%")
        plt.axis('off')  # Hide the axis
        plt.show()

    except ValueError as e:
        print(e)
```

Classification Model: Dogs-Vs-Cats classifier

```
# Training Code
from google.colab import drive
drive.mount('/content/drive')
```

```
import os
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Constants
IMG_SIZE = 256  # Reduced image size for memory efficiency
DATA_DIR = "/content/drive/My Drive/Training/D&Ctrain"  # Adjusted to your
dataset path
NUM_CLASSES = 2
BATCH_SIZE = 32

# Data Preprocessing
datagen = ImageDataGenerator(
    rescale=1./255,
    validation_split=0.2  # Splitting data for validation
)

# Training and Validation Generators
train_generator = datagen.flow_from_directory(
    DATA_DIR,
    target_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    subset='training'
)

validation_generator = datagen.flow_from_directory(
    DATA_DIR,
    target_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    subset='validation'
)

# Model architecture
model = Sequential([
    Conv2D(32, (3, 3), activation="relu", input_shape=(IMG_SIZE, IMG_SIZE,
3)),
    MaxPooling2D(2, 2),
    Conv2D(64, (3, 3), activation="relu"),
    MaxPooling2D(2, 2),
    Conv2D(128, (3, 3), activation="relu"),
    MaxPooling2D(2, 2),
```

```
    Flatten(),
    Dense(512, activation="relu"),
    Dense(NUM_CLASSES, activation="softmax")
])

model.compile(optimizer="adam", loss="categorical_crossentropy",
metrics=["accuracy"])

# Train the model
model.fit(
    train_generator,
    steps_per_epoch=train_generator.samples // BATCH_SIZE,
    validation_data=validation_generator,
    validation_steps=validation_generator.samples // BATCH_SIZE,
    epochs=20
)

# Save the trained model
model.save("/content/drive/My Drive/Trained_Models/DC_classifier.keras")
# Save in .keras format
```

```
# Testing Code
from google.colab import drive
drive.mount('/content/drive')

import cv2
import os
import numpy as np
from tensorflow.keras.models import load_model

# Constants
IMG_SIZE = 128
MODEL_PATH = '/content/drive/My Drive/Trained_Models/DC_classifier.keras'
TEST_DATA_DIR = '/content/drive/My Drive/Training/D&Ctest'  # Directory
containing the test data

# Load the trained model
model = load_model(MODEL_PATH)

def predict_image(image_path):
    img = cv2.imread(image_path)
    if img is None:
```

```
        raise ValueError(f"The image at path {image_path} could not be
read. Check the file path and permissions.")
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    img = cv2.resize(img, (IMG_SIZE, IMG_SIZE))
    img = np.array(img) / 255.0
    img = np.expand_dims(img, axis=0)

    prediction = model.predict(img)
    predicted_label = "Dog" if np.argmax(prediction) == 0 else "cat"
    probability = np.max(prediction) * 100

    return predicted_label, probability

# Iterate over each image in the test data directory and predict
for img_name in os.listdir(TEST_DATA_DIR):
    img_path = os.path.join(TEST_DATA_DIR, img_name)
    try:
        label, probability = predict_image(img_path)
        print("image : Predicted Label : Probability")
        print(f"{img_name}, {label}, {probability:.2f}%")
    except ValueError as e:
        print(e)
```

Segmentation Model

```
# Segmentation model V3.0
from google.colab import drive
drive.mount('/content/drive')

import os
import numpy as np
import cv2
import matplotlib.pyplot as plt
import keras
import albumentations as A
import segmentation_models as sm

# Set the directories for the training, validation, and test sets
DATA_DIR = '/content/drive/My Drive/Training/DATA3'

x_train_dir = '/content/drive/My Drive/Training/DATA2/RGBtrain'  # RGB
images folder for training
y_train_dir = '/content/drive/My Drive/Training/DATA2/masktrain'  # Masks
folder for training
x_valid_dir = os.path.join(DATA_DIR, 'RGBval')  # RGB images folder for
validation
y_valid_dir = os.path.join(DATA_DIR, 'maskval')  # Masks folder for
validation

# Define the classes for segmentation
CLASSES = ['iceberg', 'icesheet', 'land']

# Define augmentation pipelines
# Note: Adjust augmentations as needed for satellite images
def get_training_augmentation():
    train_transform = [
        # Resize images to 512x512
        A.Resize(512, 512, interpolation=cv2.INTER_NEAREST),
        A.HorizontalFlip(p=0.5),
        A.RandomRotate90(p=0.5),
        A.Transpose(p=0.5),
        A.ShiftScaleRotate(scale_limit=0.1, rotate_limit=0,
shift_limit=0.1, p=0.5, border_mode=0),
        A.GridDistortion(p=0.5),
        A.PadIfNeeded(min_height=320, min_width=320, always_apply=True,
border_mode=0),
        A.RandomCrop(height=320, width=320, always_apply=True),
```

```
        A.IAAAdditiveGaussianNoise(p=0.2),
        A.OneOf(
            [
                A.CLAHE(p=1),
                A.RandomBrightnessContrast(p=1),
            ],
            p=0.9,
        ),
        A.OneOf(
            [
                A.IAASharpen(p=1),
                A.Blur(blur_limit=3, p=1),
            ],
            p=0.9,
        ),
    ]
    return A.Compose(train_transform)

def get_validation_augmentation():
    # Add paddings to make image shape divisible by 32 for the U-Net model
    test_transform = [
        A.PadIfNeeded(384, 480)
    ]
    # Validation augmentation, just resize
    return A.Compose([
        A.Resize(512, 512, interpolation=cv2.INTER_NEAREST),
    ])
    return A.Compose(test_transform)

def get_preprocessing(preprocessing_fn):
    # Preprocessing function specific for the neural network backbone
    _transform = [
        A.Lambda(image=preprocessing_fn),
    ]
    return A.Compose(_transform)

# Define the model
BACKBONE = 'efficientnetb3'
BATCH_SIZE = 10
LR = 0.0002
EPOCHS = 30

preprocess_input = sm.get_preprocessing(BACKBONE)

n_classes = len(CLASSES) + 1  # Number of classes + 1 for the background
```

```
activation = 'softmax'

model = sm.Unet(BACKBONE, classes=n_classes, activation=activation,
encoder_weights='imagenet')

# Define dataset class for loading images
class Dataset:
    """Load images and masks for training and validation."""

    def __init__(self, images_dir, masks_dir, classes=None,
augmentation=None, preprocessing=None):
        self.ids = os.listdir(images_dir)
        self.images_fps = [os.path.join(images_dir, image_id) for image_id
in self.ids]
        # Adjust the line below if masks have a different naming pattern
        self.masks_fps = [os.path.join(masks_dir, image_id.replace('RGB',
'mask')) for image_id in self.ids]
        # convert str names to class values on masks
        self.class_values = [classes.index(cls.lower()) for cls in
classes]

        self.augmentation = augmentation
        self.preprocessing = preprocessing

    def __getitem__(self, i):
        # read data
        image_path = self.images_fps[i]
        mask_path = self.masks_fps[i]

        image = cv2.imread(image_path)
        mask = cv2.imread(mask_path, 0)

        # Check if the images were loaded successfully
        if image is None:
            raise FileNotFoundError(f"The following image file could not
be loaded: {image_path}")
        if mask is None:
            raise FileNotFoundError(f"The following mask file could not be
loaded: {mask_path}")

        assert image.shape[:2] == mask.shape[:2], f"Image and mask {i}
should have the same dimensions, but are {image.shape[:2]} and
{mask.shape[:2]}."

        # Resize images and masks to 512x512
```

```
        image = cv2.resize(image, (512, 512))
        mask = cv2.resize(mask, (512, 512),
interpolation=cv2.INTER_NEAREST)
        # one hot encode the mask
        masks = [(mask == v) for v in self.class_values]
        mask = np.stack(masks, axis=-1).astype('float')

        # add background if mask is not binary
        if mask.shape[-1] != 1:
            background = 1 - mask.sum(axis=-1, keepdims=True)
            mask = np.concatenate((mask, background), axis=-1)

        # apply augmentations
        if self.augmentation:
            sample = self.augmentation(image=image, mask=mask)
            image, mask = sample['image'], sample['mask']

        # apply preprocessing
        if self.preprocessing:
            sample = self.preprocessing(image=image, mask=mask)
            image, mask = sample['image'], sample['mask']

        return image, mask

    def __len__(self):
        return len(self.ids)

# Define DataLoader class for creating data batches
class Dataloder(keras.utils.Sequence):
    """Load data from dataset and form batches."""

    def __init__(self, dataset, batch_size=1, shuffle=False):
        self.dataset = dataset
        self.batch_size = batch_size
        self.shuffle = shuffle
        self.indexes = np.arange(len(dataset))

        self.on_epoch_end()

    def __getitem__(self, i):
        # Generate indexes of the batch
        start = i * self.batch_size
        stop = min((i + 1) * self.batch_size, len(self.indexes))

        # Get the data samples
```

```
        data = []
        for j in range(start, stop):
            data.append(self.dataset[j])

        # Transpose list of lists
        batch = [np.stack(samples, axis=0) for samples in zip(*data)]

        return batch

    def __len__(self):
        """Denotes the number of batches per epoch"""
        return len(self.indexes) // self.batch_size

    def on_epoch_end(self):
        """Updates indexes after each epoch"""
        if self.shuffle:
            np.random.shuffle(self.indexes)
# Dataset for train images

train_dataset = Dataset(
    x_train_dir,
    y_train_dir,
    classes=CLASSES,
    augmentation=get_training_augmentation(),
    preprocessing=get_preprocessing(preprocess_input),
)

# Dataset for validation images
valid_dataset = Dataset(
    x_valid_dir,
    y_valid_dir,
    classes=CLASSES,
    augmentation=get_validation_augmentation(),
    preprocessing=get_preprocessing(preprocess_input),
)

train_dataloader = Dataloder(train_dataset, batch_size=BATCH_SIZE,
shuffle=True)
valid_dataloader = Dataloder(valid_dataset, batch_size=1, shuffle=False)

# Define callbacks for learning rate adjustment and best model
checkpointing
callbacks = [
    keras.callbacks.ModelCheckpoint('./best_model.h5',
save_weights_only=True, save_best_only=True, mode='min'),
```

```
    keras.callbacks.ReduceLROnPlateau(),
]

# Compile the model
model.compile('Adam', loss=sm.losses.bce_jaccard_loss,
metrics=[sm.metrics.iou_score])

# Train the model
history = model.fit(
    train_dataloader,
    steps_per_epoch=len(train_dataloader),
    epochs=EPOCHS,
    callbacks=callbacks,
    validation_data=valid_dataloader,
    validation_steps=len(valid_dataloader),
)

# Visualization of training process can be added here if needed
# Plot the training history
def plot_training_history(history):
    plt.figure(figsize=(16, 4))

    plt.subplot(1, 2, 1)
    plt.plot(history.history['iou_score'], label='Training IoU Score')
    plt.plot(history.history['val_iou_score'], label='Validation IoU
Score')
    plt.title('IoU Score over epochs')
    plt.xlabel('Epochs')
    plt.ylabel('IoU Score')
    plt.legend()

    plt.subplot(1, 2, 2)
    plt.plot(history.history['loss'], label='Training Loss')
    plt.plot(history.history['val_loss'], label='Validation Loss')
    plt.title('Loss over epochs')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.legend()

    plt.show()

# After training the model, call this function to plot the history
plot_training_history(history)

# After training
```

```
model.save('my_model.h1')  # This saves the model architecture and weights
together


# Testing Code
from google.colab import drive
drive.mount('/content/drive')

import os
import cv2
import numpy as np
import matplotlib.pyplot as plt
import keras
import segmentation_models as sm
from sklearn.metrics import accuracy_score

# Load the trained model
model_path = '/content/drive/My
Drive/Trained_Models/iceberg_segmentation_final.h5'
model = keras.models.load_model(model_path, compile=False)

BACKBONE = 'efficientnetb3'
preprocess_input = sm.get_preprocessing(BACKBONE)

def calculate_accuracy(pred_mask, true_mask):
    # Flatten the arrays to compute accuracy
    pred_flat = pred_mask.flatten()
    true_flat = true_mask.flatten()
    return accuracy_score(true_flat, pred_flat)

def predict_and_visualize(image_path, mask_path, model, preprocessing_fn):
    # Load and preprocess the image
    image = cv2.imread(image_path)
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    desired_size = (512, 512)  # Replace with your model's expected input
size
    image = cv2.resize(image, desired_size)
    preprocessed = preprocessing_fn(image)

    # Load and resize the true mask
    true_mask = cv2.imread(mask_path, 0)
    true_mask = cv2.resize(true_mask, desired_size,
interpolation=cv2.INTER_NEAREST)

    # Predict and process the prediction
```

```
    prediction = model.predict(np.expand_dims(preprocessed,
axis=0)).squeeze()
    processed_prediction = np.argmax(prediction, axis=-1)

    # Ensure the processed prediction is 3-channel (RGB) for overlay
    processed_prediction_rgb =
cv2.cvtColor(processed_prediction.astype('uint8'), cv2.COLOR_GRAY2RGB)

    # Calculate accuracy
    accuracy = calculate_accuracy(processed_prediction, true_mask)

    return image, processed_prediction, processed_prediction_rgb, accuracy

def visualize_results(results):
    fig, axes = plt.subplots(len(results), 3, figsize=(15, 5 *
len(results)))
    if len(results) == 1:
        axes = [axes]

    for i, (image, pred_mask, overlay, accuracy) in enumerate(results):
        if len(results) > 1:
            ax1, ax2, ax3 = axes[i]
        else:
            ax1, ax2, ax3 = axes

        ax1.imshow(image)
        ax1.set_title('Original Image')
        ax1.axis('off')

        ax2.imshow(pred_mask, cmap='gray')
        ax2.set_title('Predicted Mask')
        ax2.axis('off')

        ax3.imshow(overlay)
        ax3.set_title(f'Overlay - Accuracy: {accuracy:.2%}')
        ax3.axis('off')

    plt.tight_layout()
    plt.show()

# Process all images in a folder
def process_folder(images_dir, masks_dir, model, preprocessing_fn):
    image_files = [f for f in os.listdir(images_dir) if
f.endswith('.jpg')]
    results = []
```

```
    for image_file in image_files:
        image_path = os.path.join(images_dir, image_file)
        mask_path = os.path.join(masks_dir, image_file.replace('RGB',
'mask'))

        if os.path.exists(mask_path):
            result = predict_and_visualize(image_path, mask_path, model,
preprocessing_fn)
            results.append(result)

    return results

# Example usage
images_dir = '/content/drive/My Drive/Training/DATA2/RGBtest'
masks_dir = '/content/drive/My Drive/Training/DATA2/masktest'
results = process_folder(images_dir, masks_dir, model, preprocess_input)
visualize_results(results)
```

Patching code

Preprocessing

```
#TRAINING DATA PREPROCESSING


#This is intended to be run once per satellite scene. For industrial
implementation, this should be modified into a function for scalability, though
it suffices for small numbers of scenes as was the case in this project.


#Loads the big images

BigImageRGB = Image.open(BigImageRGBlocation).convert('RGB')

BigImageMask = Image.open(BigImageMaskLocation).convert('RGB')


#splits the big images into many smaller 500x500 images ('patches')

RGBpatches = pat.patchify(np.asarray(BigImageRGB), (500, 500, 3), step = 500)

MaskPatches = pat.patchify(np.asarray(BigImageMask), (500, 500, 3), step = 500)
```

```
#Exporting patches

c = 0 #counter to split into train test and validation

#RGB patch exporting

for i in range(RGBpatches.shape[0]): #Loops over every patch of the inputted
image

  for j in range(RGBpatches.shape[1]):

    if c%10 == 0:

      filename = "/content/DATA2/test/RGBtest/RGBimage1_{}_{}.jpg" #IMPORTANT:
change image1 to image2 etc for every large image processed

      im = Image.fromarray(RGBpatches[i, j, 0]).convert('RGB')

      im.save(filename.format(i, j))

      c += 1

    elif (c + 1)%10 == 0:

      filename = "/content/DATA2/val/RGBval/RGBimage1_{}_{}.jpg" #IMPORTANT:
change image1 to image2 etc for every large image processed

      im = Image.fromarray(RGBpatches[i, j, 0]).convert('RGB')

      im.save(filename.format(i, j))

      c += 1

    else:

      filename = "/content/DATA2/train/RGBtrain/RGBimage1_{}_{}.jpg" #IMPORTANT:
change image1 to image2 etc for every large image processed

      im = Image.fromarray(RGBpatches[i, j, 0]).convert('RGB')

      im.save(filename.format(i, j))

      c += 1

c = 0

#Mask patch exporting

for i in range(MaskPatches.shape[0]): #Loops over every patch of the inputted
image

  for j in range(MaskPatches.shape[1]):

    if c%10 == 0:
```

```
      filename = "/content/DATA2/test/masktest/maskimage1_{}_{}.jpg" #IMPORTANT:
change image1 to image2 etc for every large image processed

      im = Image.fromarray(MaskPatches[i, j, 0]).convert('RGB')

      im.save(filename.format(i, j))

      c += 1

    elif (c + 1)%10 == 0:

      filename = "/content/DATA2/val/maskval/maskimage1_{}_{}.jpg" #IMPORTANT:
change image1 to image2 etc for every large image processed

      im = Image.fromarray(MaskPatches[i, j, 0]).convert('RGB')

      im.save(filename.format(i, j))

      c += 1

    else:

      filename = "/content/DATA2/train/masktrain/maskimage1_{}_{}.jpg"
#IMPORTANT: change image1 to image2 etc for every large image processed

      im = Image.fromarray(MaskPatches[i, j, 0]).convert('RGB')

      im.save(filename.format(i, j))

      c += 1
```

Prediction processing

```
#loads the new image we are segmenting

InputtedBigImage = Image.open(InputtedBigImageLocation).convert('RGB')

#slices the new image into patches

InputtedImagePatches = pat.patchify(np.asarray(InputtedBigImage), (500, 500, 3),
step = 500)

#creating a tuple recording the size and number of channels of the original image
for use in unpatching the masks

sizetuple = InputtedBigImage.size

sizelist = list(InputtedBigImage.size)

sizelist.append(3) #adding the channel dimension

TrueSizeTuple = tuple(sizelist)

#Generating and storing new masks for every patch
```

```
GeneratedMaskPatches = np.empty_like(InputtedImagePatches) #Creates an object to store the generated masks of every patch

for i in range(InputtedImagePatches.shape[0]): #Loops over every patch of the inputted image

  for j in range(InputtedImagePatches.shape[1]):

    GeneratedMaskPatches[i, j] = PredictionCode(InputtedImagePatches[i, j, 0, :, :, :]) #Uses your prediction function. Either that or put the entirety of that code in here


#Putting together our generated masks into one big mask for the whole scene

FinalGeneratedMask = pat.unpatchify(GeneratedMaskPatches, TrueSizeTuple)


#Visualisation

fig = plt.figure(figsize=(10, 7))

fig.add_subplot(1, 2, 1)

plt.imshow(InputtedBigImage)

plt.axis('off')

plt.title("Original Image")


fig.add_subplot(1, 2, 2)

plt.imshow(FinalGeneratedMask)

plt.axis('off')

plt.title("Segmentation Mask")


#Exporting

filename = "/content/finalmaskexport/generatedmask_{}.jpg"

im = Image.fromarray(RGBpatches[i, j, 0]).convert('RGB')

im.save(filename.format(random.choice(range(9999))))
```