Sea Ice Surface Characterization via Semantic Segmentation with Convolutional Neural Networks

By

Matthew Peter Ivan King

A thesis submitted to the

School of Graduate Studies

in partial fulfillment of the requirements for the degree of

Master of Engineering

Faculty of Engineering and Applied Science

Memorial University of Newfoundland

October, 2022

St. John's Newfoundland & Labrador

Abstract

Visual data provides rich information about real-world objects. Computer vision is a substantial and growing field which seeks to distill useful information from photographic imagery. The primary focus of this work centers on the application of machine-learning based computer vision algorithms, along with minor applications of more traditional computer vision techniques. The specific task approached herein is known as semantic segmentation; the methodology by which each region of an image, at an individual pixel level, is assigned a classification from a predetermined set of possible classes. The classes considered in this work are: open water, level ice, broken ice, ridged ice, ice in flexural failure, and an 'other' class. Accurate segmentation of these classes is the primary objective of this work. The imagery utilized in this work were captured from two cameras mounted on different piers of the Confederation Bridge during the local ice season. Several hundred of the images collected have been manually labelled and split into training, validation, and testing subsets. These data have been used to train an ensemble of convolutional neural networks. Transfer learning is applied such that the encoder portions of the neural networks have been pretrained on the ImageNet dataset, providing them with the capability to produce meaningful feature maps while significantly reducing the training time required for the overall models to learn the semantic segmentation task at hand.

Acknowledgements

The author extends sincere gratitude to his academic supervisors Dr. Rocky Taylor, and to co-supervisors and National Research Council colleagues Dr. Louis Poirier, Mr. Phillippe Lamontagne, and Dr. Robert Briggs. Their supervision and feedback has been critical to the completion of this project.

Further thanks are extended to my family, without whom I could never have achieved the accomplishments and milestones which have afforded me the ability to complete my academic and personal goals.

Additional thanks to Strait Crossing Bridge Limited and Public Services Procurement Canada for access to the Confederation Bridge and the imagery used in this work.

Financial support from Memorial University of Newfoundland, the National Research Council of Canada, the Natural Sciences and Engineering Research Council (NSERC) of Canada, Hibernia Management and Development Company, Ltd. (HMDC), Terra Nova Development (Suncor Energy Inc. - Operator) and the Department of Industry, Energy and Technology are gratefully acknowledged.

Table of Contents

List of Figures

List of Tables

List of Equations

Chapter 1. Introduction

1.1 Overview

Sea ice monitoring and characterization is highly relevant to many operations in arctic and sub-arctic regions. Sea ice can pose significant hazards to offshore structures and maritime vessels.

Visual imagery has certain advantages as compared to other methods of instrumentation. Visual imagery is very information dense while also being human-interpretable. Cameras are relatively inexpensive when compared to LIDAR or radar systems. This has led to a vast quantity of images and makes monitoring equipment much more accessible.

Automatic systems to detect hazardous ice features can potentially assist to avoid catastrophic risks for vessels and may be used as input to systems for short-term load forecasting on offshore structures. Such systems could be directly implemented for real-time decision support on a vessel's bridge or could be used as an input for autonomous vessels.

In this work, an exploration of computer vision techniques for the visual characterization of sea ice conditions is conducted. Specifically, the task of semantic segmentation is employed to classify each pixel within a given image into one of a set of classes. Each class may pose its own unique risks and loads when interacting with a structure. Semantic segmentation is accomplished using convolutional neural networks trained on a dataset of images collected from cameras mounted on the Confederation Bridge and manually annotated. In this manner, a surface characterization of the ice conditions captured in each image is achieved. This system was implemented using TensorFlow [1] in Python.

Due to the data's source in a sheltered subarctic region, ice features such as icebergs and multiyear ice are not considered in this work simply due to the lack of their presence in the area of study.

1.2 Background

This work builds upon the foundation established in prior work conducted in partnership between the University of Ottawa and the National Research Council [2]. The previous work was responsible for the initial image annotations corresponding to approximately sixty percent of the training and validation datasets. Similar to this work, the previous work sought to develop a method for characterization of sea ice using convolutional neural networks, specifically the Mask R-CNN [3] architecture.

Unlike this work, the previous work utilized the approach of panoptic segmentation rather than the semantic segmentation approach employed in this work. These differences are examined in more detail in Section 2.2.1.

Furthermore, the prior work classified only four classes as compared to the six classes considered in this work. The set of ice classes considered in the prior work is depicted in Figure 1. These classes are all ice-specific and therefore exclude the classes for open-water and the 'other' class corresponding to the pier leg itself which have been added in this work. View Figure 20 from Section 3.2 for the full set of classes utilized in this work.

The previous work did not report any specific metrics of performance such as accuracy, IOU score, precision, or f1-score.

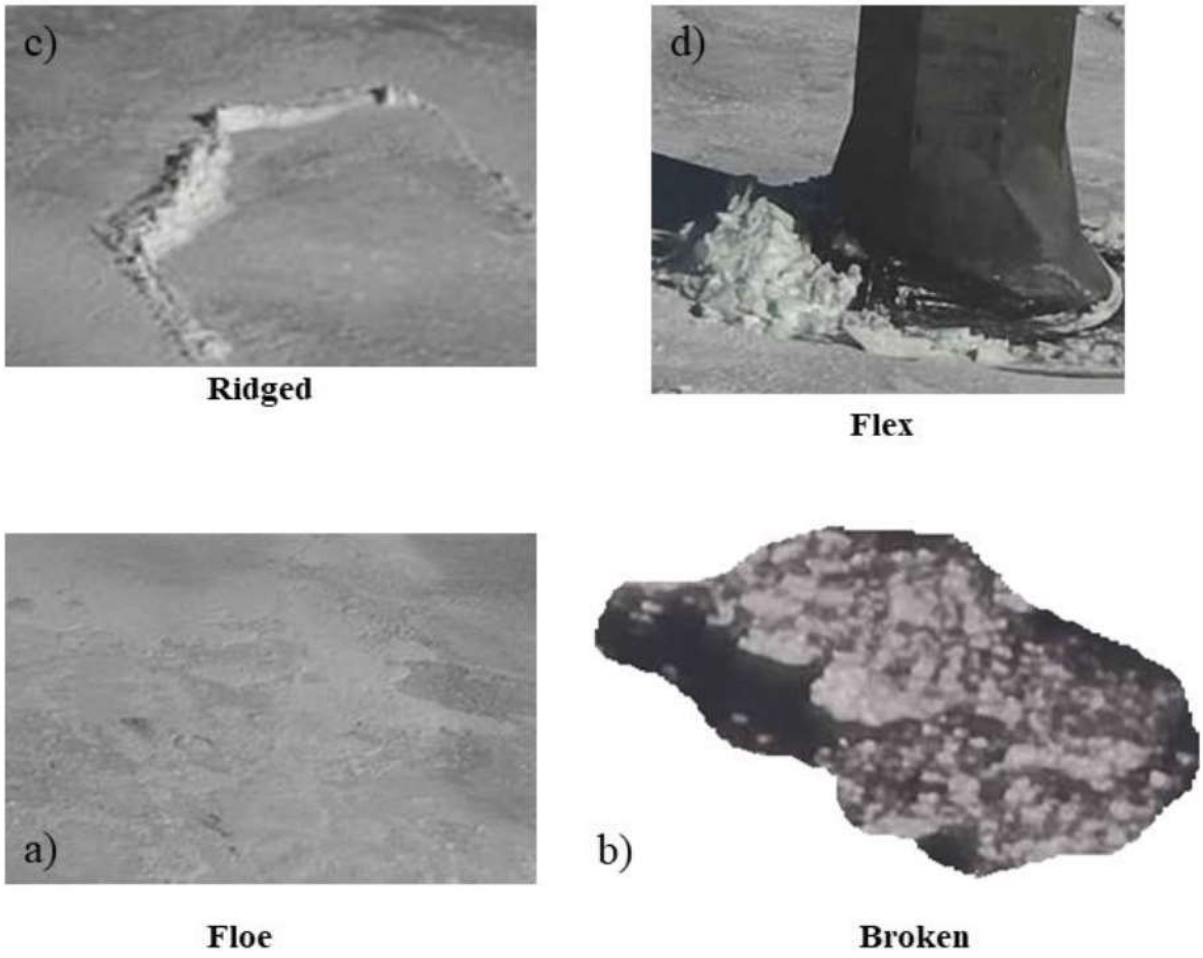


Figure 1. Ice classes of the prior work. Figure 2 from [2].

Current sea ice characterization activities are largely performed using satellite radar and human observers from aircraft or vessels. The spatial and temporal resolutions of these methods leaves gaps in information that could be critical to operations. Furthermore, human observers are liable to experience fatigue, may produce inconsistent estimates, and can only process a limited quantity of data. Visual ice characterization carried out by automatic systems do not experience fatigue, are more consistent, and can process large volumes of data.

The Confederation Bridge was selected as the imagery source for this work due to the work's relevance to the load monitoring program, as well as the availability of a large volume of high quality imagery.

1.3 Objectives

The objective of this work is the production of methods capable of producing meaningful data regarding current sea ice conditions from visual sensors. The produced data should be applicable to downstream analysis tasks.

The primary method adopted for the production of data from visual imagery is semantic segmentation. In this task, a classification is assigned to each pixel within each processed image – more in depth information is provided in Section 2.2.1. Demonstration of the usefulness of the produced data to other applications is provided in Chapter 4.

An additional objective of this work is to provide a proof-of-concept for more generalized models capable of processing sea ice imagery captured not only from the Confederation Bridge but across a broad span of arctic and sub-arctic ice conditions. Results regarding the deployment of the trained models to a previously unseen arctic domain may be viewed in Appendix C.

Chapter 2. Literature Review

2.1 Confederation Bridge

The data used for model training and evaluation was collected at the Confederation Bridge via two cameras mounted on separate bridge piers. The following section provided background information on the bridge and the bridge monitoring program.

The Confederation Bridge is an almost 13-kilometre long bridge spanning between New Brunswick and Prince Edward Island, as pictured in Figure 2 and Figure 3. The bridge was specified to have a 100-year design life [4]. This fact, in combination with its location with a history of sea ice presence, has led to significant concerns regarding the ice loads applied to the piers of the bridge and the long-term loads that may be expected across its lifetime. As such, an extensive program was carried out to monitor the bridge and the loads to which it is subjected [5, 6].

The Confederation Bridge ice load monitoring program has, since its initiation in 1997, produced a sizable and highly valuable dataset. The winters since the bridge's opening have been relatively milder compared to the period considered during the design phase of its construction. Many unique ice interactions have been closely studied and have provided valuable information regarding the nature of ice-structure interactions at the site.

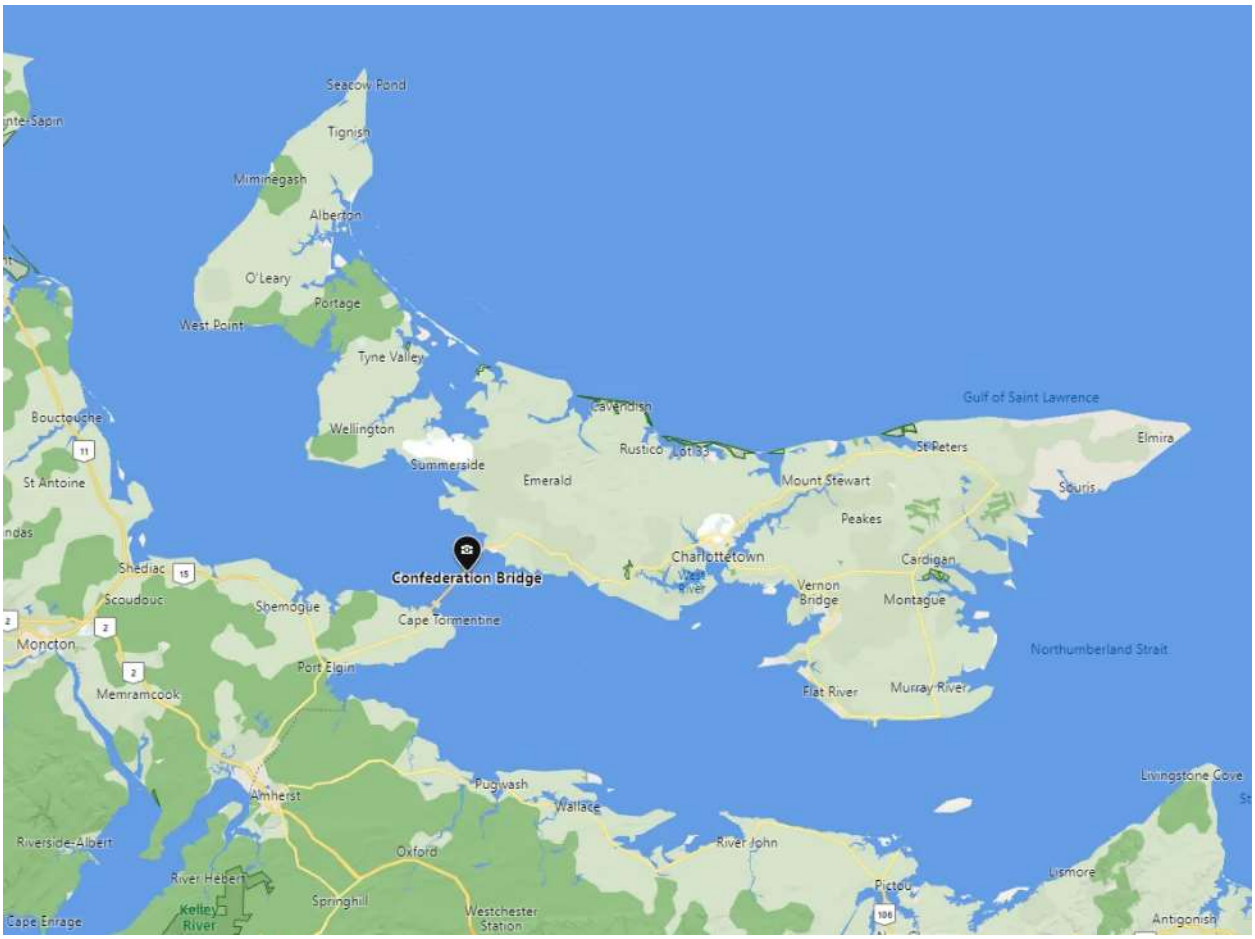


Figure 2. Location of Confederation Bridge in Northumberland Strait.

A design safety factor of 4.0 was considered with regards to the total applied load across all possible modes [4]. The primary loading factors considered were the wind, current, and ice loads. The data generated has been highly valuable and provided numerous insights into the understanding of ice-structure interactions [7, 8, 9].

Figure 3. The Confederation Bridge. Government of Canada image.

The bridge crosses the Northumberland Strait at its narrowest location for a total length of 12910 meters [4]. There are a total of 44 piers numbered P1 to P44, with the numbering starting on the PEI side of the bridge [10]. The standard pier design is presented below in Figure 4. At the center of the bridge, there is a higher section to provide vessel passage between P21 and P22. The piers at this location are taller and have been further reinforced but, as such, could not be fitted with instrumentation.

The two cameras used to collect the imagery employed in this work are located on P23 and P24. These two piers are near the navigation span and as such are taller than most of the other piers but not as tall as P21 or P22.

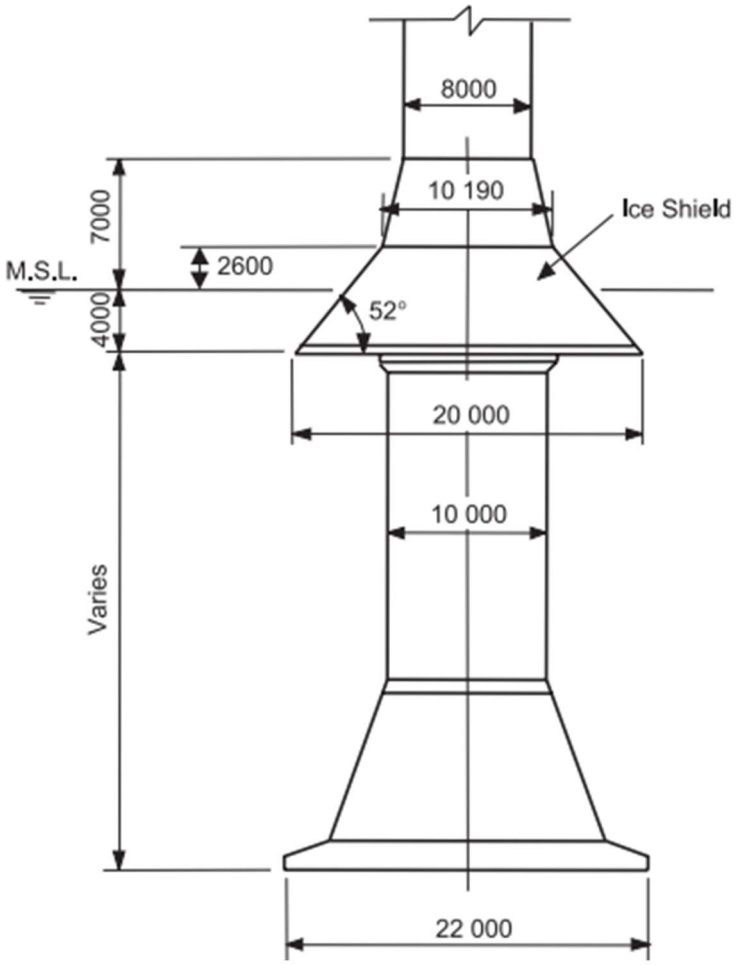


Figure 4. Standard Pier Design Figure 1 from [11]

The conical section the pier at the waterline is designed to cause ice to fail in flexure when interacting with the pier leg. This mode of ice failure is associated with lower loads. An example of the buildup of a significant amount of ice in flexural failure is presented in Figure 5. The larger ice pieces generated during flexural failure more easily clear the structure than the fine snow-like pieces which are created during a crushing failure.

Figure 5. Rubble pileup at 10:32am, April 4th, 2003. Figure 11 from [8].

2.1.1 Environmental Conditions

The Northumberland Strait's ice conditions may be categorized entirely as first-year pack ice, including both rafted and ridged ice features [6]. Significant tidal currents are observed in the strait and, in combination with predominant winds, act as the major driving force of ice movement in the region. The local ice season typically ranges from December to mid-April [10]. The most severe ice conditions typically occur towards the end of the ice season in March and April [11], and as such, many studies have focused on this period for analysis.

Over a one-hundred-year return period, the 10 minute mean wind speed has been estimated to be 26.5 m/s with a corresponding wind load of 3.7 meganewtons [4]. The typical maximum water

current is 2 m/s with a related load on a single pier of 8 meganewtons [4] in combination with the maximum expected wave heights of 4 meters.

The effect of the bridge itself on the local ice conditions has been an aspect of consideration. The ice breakup was forecasted to be delayed by an average of 0.1 days with a delay of up to 3.5 days in 1% of all scenarios [12]. The addition of near-land piers has extended the extent of land-fast ice, as has been visually confirmed in Figure 6. It was also expected that the concentration of ice would typically be lower down-range of the bridge due to fracturing and ice jams at the bridge [12].

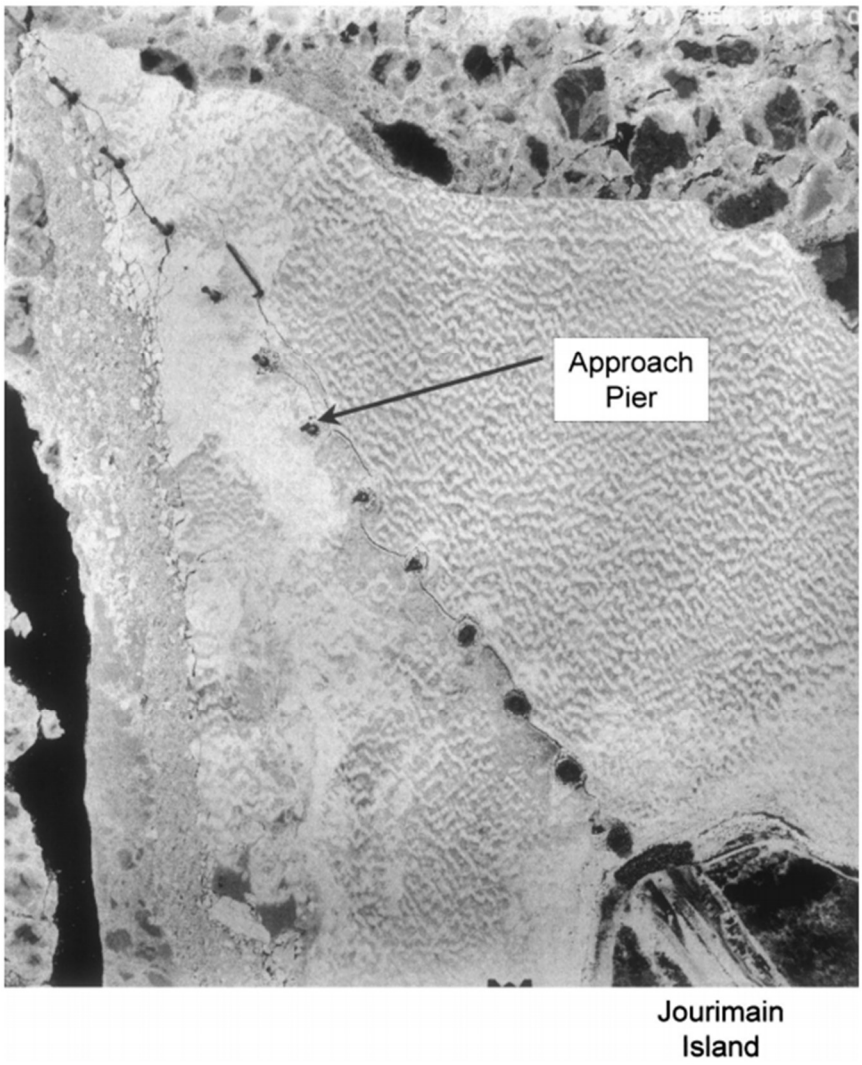


Figure 6. Land fast ice interaction with approach piers. Figure 5 from [12]

2.2 Computer Vision Tasks

This section describes the specific computer vision tasks pursued in this work. The first task is segmentation and is the primary focus. The second task is keypoint detection and is applied as a downstream application to demonstrate uses of the primary task.

2.2.1 Segmentation

Semantic segmentation is the task of assigning a classification, from a set of possible classes, to each pixel in a given image. Some well-known public segmentation datasets include Microsoft

COCO (common objects in context) [13] and Cityscape [14]. Semantic segmentation can vary across broad range of applications. Segmentation is often applied in the context of self-driving vehicles. Semantic segmentation is often applied in the medical field in order to detect anomalies such as tumors from various forms of medical imagery [15].

In the COCO dataset, and consequently much of the following works, each classification is defined as either a *stuff* category or a *things* category. Objects belonging to a *things* category represents discrete countable instances such humans, vehicles, or animals [16]. Objects belonging to a *stuff* category represent an amorphous region which is not easily discretized into individual instances, such as: ground, sky, or ocean [16]. This distinction may at first seem trivial, but in fact the consideration of these factors have a significant impact not only on problem formulation but also the selection of neural network architecture.

Tasks focusing on the segmentation of *things* is known as instance segmentation and typically employ a two-stage architecture which includes a bounding box proposal network and a secondary module to classify the pixels within each bounding box [16]. Each object instance must be assigned a unique identifier along with its classification and a binary mask denoting the object's presence within the bounding box. Instance segmentation does not guarantee the labelling of every pixel within an image.

Tasks focusing on the segmentation of *stuff* is known as semantic segmentation and typically relies on fully-convolutional networks which output a classification for every pixel within an image and does not assign unique instance identifiers [16].

Panoptic segmentation is an extension of both semantic and instance segmentation which seeks to simultaneously label every pixel in an image while also assigning unique instance identifiers to each *thing* class object identified in the image [16]. Panoptic segmentation is the most recent iteration of the segmentation problem. The difference between instance, semantic, and panoptic segmentation is visually demonstrated in Figure 7.

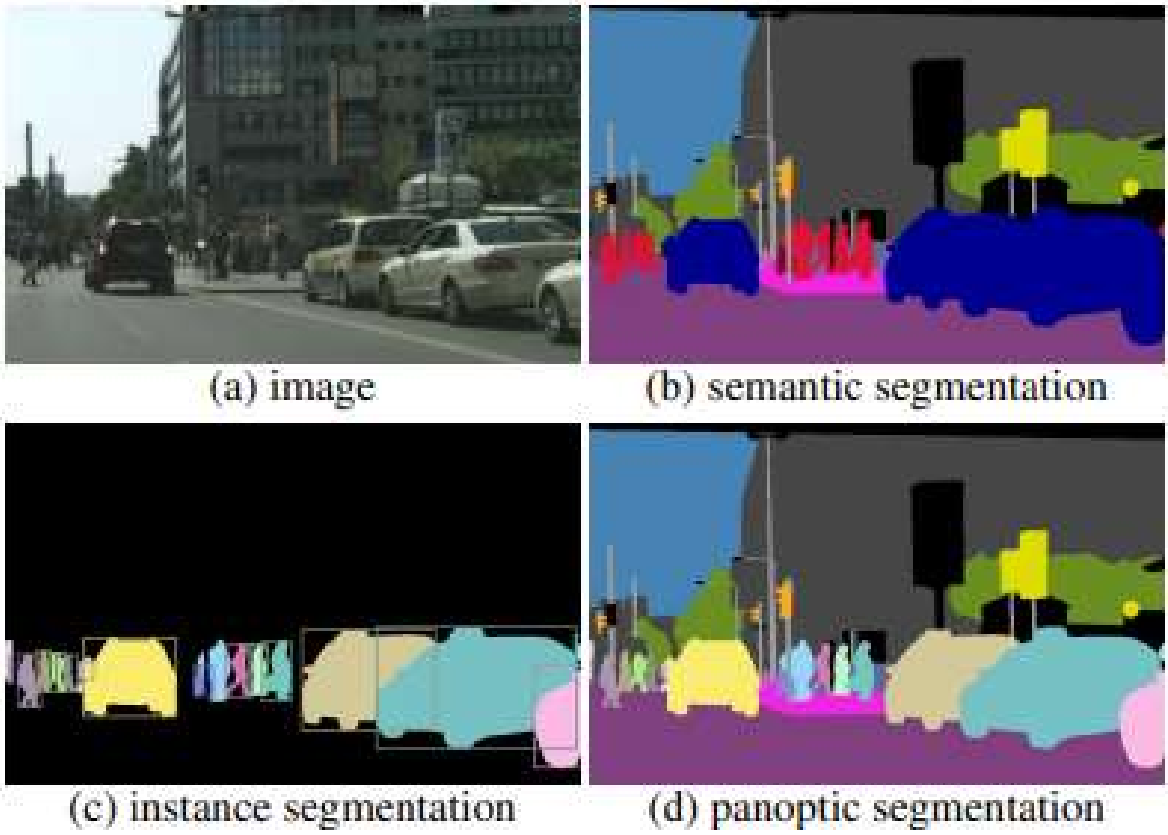


Figure 7. Semantic, instance, and panoptic segmentation. Figure 1 from [16].

2.2.2 Keypoint Detection

Keypoint detection is a traditional computer vision technique which seeks to identify points in an image which are highly distinctive. Each detected keypoint is also assigned some form of a

descriptor – a mathematical representation which encodes this keypoint in a uniquely identifiable fashion. Keypoint detection and description are most often, but not always, both performed by a single algorithm.

The task of both keypoint detection and description is to produce outputs which are invariant to rotation, translation, and scaling of the input images [17]. Furthermore, outputs should be robust to changes in illumination, changes of perspective, and noise. These useful properties lend themselves to a wide array of applications. These applications include:

- Object tracking [18]: Keypoint detection and description allows for the tracking of highly distinctive object features from one video frame to the next, so that the motion may be tracked. This is the mode in which keypoint detection is applied within this work.
- Image stitching [19]: Images taken from different angles may be merged together into larger or panoramic images. Identification and matching of keypoints present in both perspectives allows for the computation of geometric transformations that combine both images into a unified frame.
- Simultaneous localization and mapping (SLAM): Similar to image stitching, some variants of SLAM makes use of keypoints to identifying common points between frames. Unlike the previous two applications, the movement of the camera relative to the environment is considered to be non-negligible. The problem is instead reframed to track the movement of the camera itself while the surrounding environment remains static. The camera is often mounted on a robotics platform which moves through the environment and, in combination with other robotic sensors, the platform is localized at every time step and while a 2D or 3D map of the environment is constructed.

Keypoint detection has been an active field of research for many years. The largest step in this field was observed with the release of the SIFT: Scale Invariant Feature Transform [17] [20]. The SIFT algorithm works by computing histograms of the image gradients along different angles and at multiple scales. Points with gradient attributes greater than a specified threshold are considered to comprise the set of detected keypoints, and are each described by their histogram of gradients. Samples of keypoints and descriptors generated using SIFT are presented in Figure 8. Notice that the keypoints lie mainly along lines or at corners.

Following the introduction of SIFT, a number of new algorithms began to be released – each of which seeking to improve on the existing solutions in some way. Notably among those released is SURF: Speeded Up Robust Features [21]. As the name implies, SURF significantly reduced the computational complexity as compared to SIFT while still producing accurate and robust keypoints and descriptors.

One major problem with both SIFT and SURF is that their algorithms are patented and require licenses for their use in applications. In their place, a number of open source algorithms have been released. These open source alternatives include BRISK: Binary Robust Invariant Scalable Keypoints [22] and ORB: Oriented Fast and Rotated BRIEF (Binary Robust Independent Elementary Features) [23]. An example of ORB keypoints being used to match points from different perspectives is presented in Figure 9.

Figure 8. Sample of SIFT keypoint. Above: Original Image. Bottom: Detected Keypoints. Adapted from Figure 5 of [17].

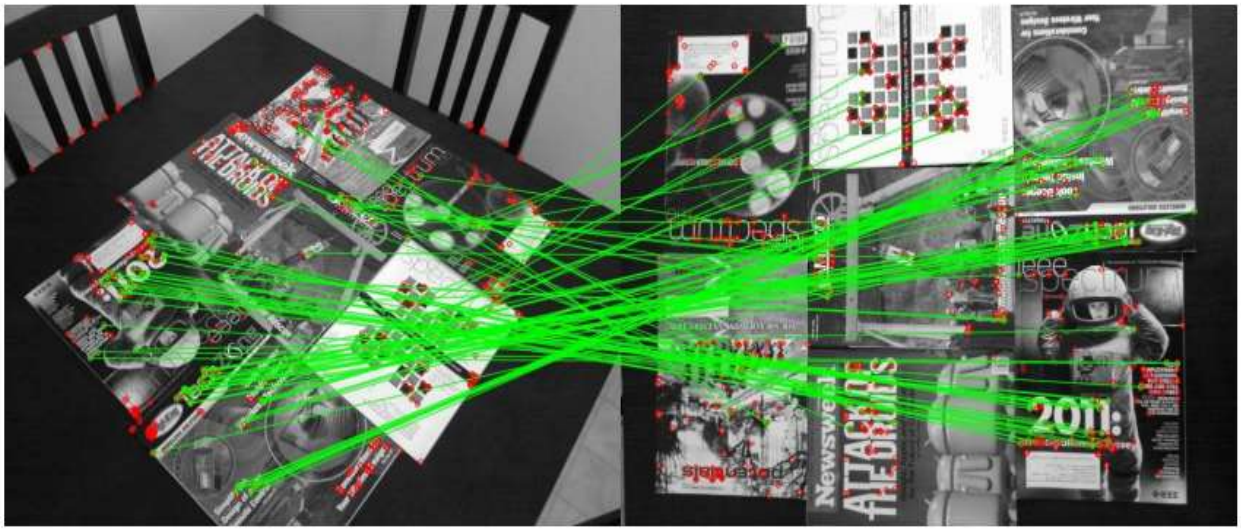


Figure 9. Keypoint matching with ORB. Valid matches in green and unmatched points in red. Figure 1 from [23].

2.3 Machine Learning Methods

Machine learning represents a broad class of algorithms which rely on data and statistical analysis in order to perform a predetermined task or to detect patterns without the manual coding of logical rules [24]. The most common forms of machine learning are regression and classification, but also includes synthesis – such as music generation, 'deep fakes', and style-transfer. Machine learning is regarded as a subclass of artificial intelligence.

Machine learning has a long history and comes in many different forms. Examples include regression, clustering, support vector machines, random forests, and many more. Deep learning, involving the training and application of artificial neural networks, represents the state-of-the-art in a number of fields including: the broad field of computer vision [25], machine translation [26], human speech audio processing [27], and a growing number of novel applications.

2.3.1 Supervised Learning

Supervised learning is the classical form of machine learning, in which a set of inputs is gathered alongside a corresponding set of targets or labels. A model is then constructed and trained in such a way so that it may map from the domain of the inputs to the domain of the outputs, ideally in such a way that the model outputs closely match those found within the provided targets. Almost all classical machine learning systems fall within this paradigm. This includes, support vector machines, random forests, linear and logistic regression, and many more.

In the instances when the computation of a global optima is not tractable, supervised learning typically begins with a model that has been randomly initialized. The new model is provided with a set of inputs and produces a set of outputs which initially will typically not reflect the true labels.

In the case of both regression and classification tasks, a loss function is implemented which computes a cost or loss based on the difference between the model's output and the true target. The closer the model's output is to the target, the lower the resulting cost will be. It is important that the cost function is differentiable with respect to the model's output. This is important because the model is trained using backpropagation, whereby the partial derivative of the loss function is identified with respect to each trainable weight within the model. The set of partial derivatives with respect to all trainable parameters is known as the gradient.

Next, the weights are updated from their initially random state along the direction which decreases the loss function as determined by the gradient. Through many repetitions of the above process, the loss function will move towards a minima and the model will become increasingly accurate at the specified task – improving the accuracy of its mapping from inputs to the desired outputs. When this routine is carried out with the gradients calculated over the full set of training examples available, the process is known as gradient descent. In practice, only a certain number of samples can be fed through a model in order to produce gradients for training. As such, a random subset of the samples available from the training dataset are selected and collectively referred to as a batch. This variant is known as stochastic gradient descent and is the cornerstone of neural network training as well as many other minima-seeking algorithms.

Numerous variants of stochastic gradient descent have been developed which seek to improve the performance via one mechanism or another, typically with the objective of decreasing the number of updates required for the model to converge to an optimal state. Popular variants include ADAM [28], NADAM [29], and RMSProp [30].

The magnitude of the step along the gradient direction is known as the learning rate λ, which is a considered a hyperparameter – a value which must be chosen manually or somehow optimized outside of the training loop. Additional hyperparameters include: batch size, the number of samples which are fed into the model and used to compute the gradient at each training step; regularization rate, the amount by which the model is penalized for building very strong weights which may lead to overfitting; and total training duration also known as the number of epochs, the total number of iterations over the complete training dataset before training is considered to be complete.

Hyperparameter optimization is a complex process by which optimal configuration of these values is determined. It is often accomplished via a grid search or random search across hyperparameter values while monitoring the change in performance against the validation set. This is further complicated by the interrelationship between various hyperparameters; they may be optimized individually, potentially leading to non-optimal combinations – or instead the parameter space may be searched jointly, potentially offering a true optima at the cost of much longer computational cost scaling exponentially with the number of hyperparameters being optimized. Formally, it can be stated that joint hyperparameter search has a time complexity of $O(c * n^k)$ where c represent the base time to evaluate one hyperparameter configuration, n being the number of discrete levels tested for each hyperparameter, and k represents the total number of hyperparameters to be optimized. Comparatively, disjointed hyperparameter optimization where only one variable is varied while the other are held constant is an $O(c * n * k)$ time complexity operation.

2.3.2 Neural Networks

Neural networks are best characterized as universal approximators [31]. In essence, this means that a sufficiently large neural network is capable of approximating any given function with an arbitrary degree of accuracy. The above postulation was originally formulated in the context of multilayer feedforward networks, also known as multilayer perceptrons. However, it has been shown that this extends to other neural network architectures such as convolutional neural networks, such that they "can be used to approximate any continuous function to an arbitrary accuracy when the depth of the neural network is large enough" [32].

The difference between convolutional neural networks and multilayer perceptrons lies in their internal connectivity structure. Both architectures utilize as their base building block neurons which, as in biology, act as computational blocks that are connected to a set of inputs via synapses.

Multilayer perceptrons consist of a set of densely connected layers where each neuron in layer N is connected to each neuron in layer N-1, as pictured in Figure 10. These layers are often referred to as a dense layer.

In a convolutional neural network, each neuron connects only to a local neighbourhood of inputs, as pictured in Figure 11, commonly a three pixel by three pixel subregion [33]. In this way, convolutional neural networks learn to make use of spatial information and have significantly less overall connections, making them far more computationally efficient than multilayer perceptrons. As such, convolutional neural networks have become the standard for many visual tasks [25, 33, 34, 35].

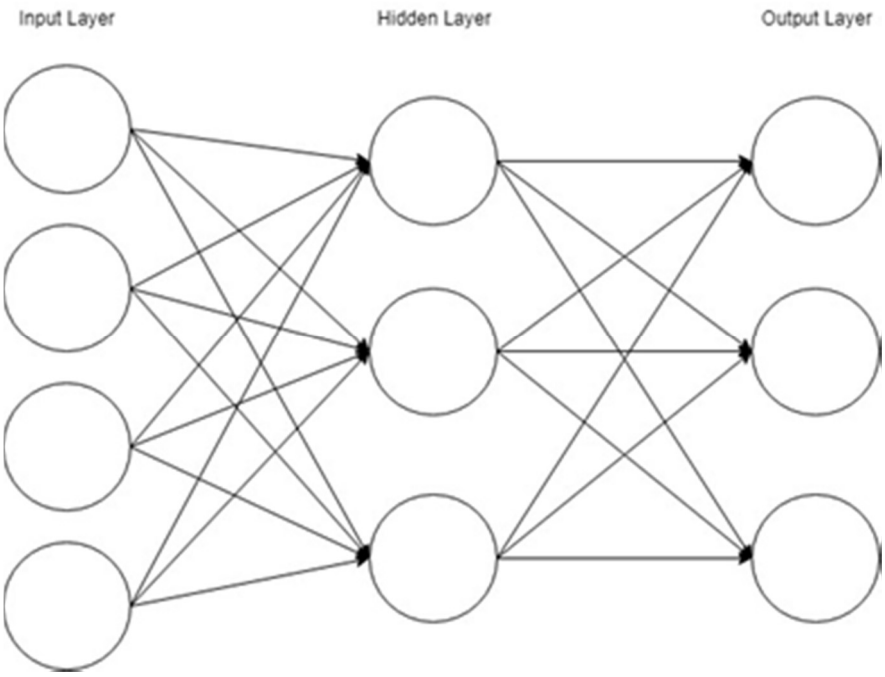


Figure 10. Multilayer Perceptron [36].

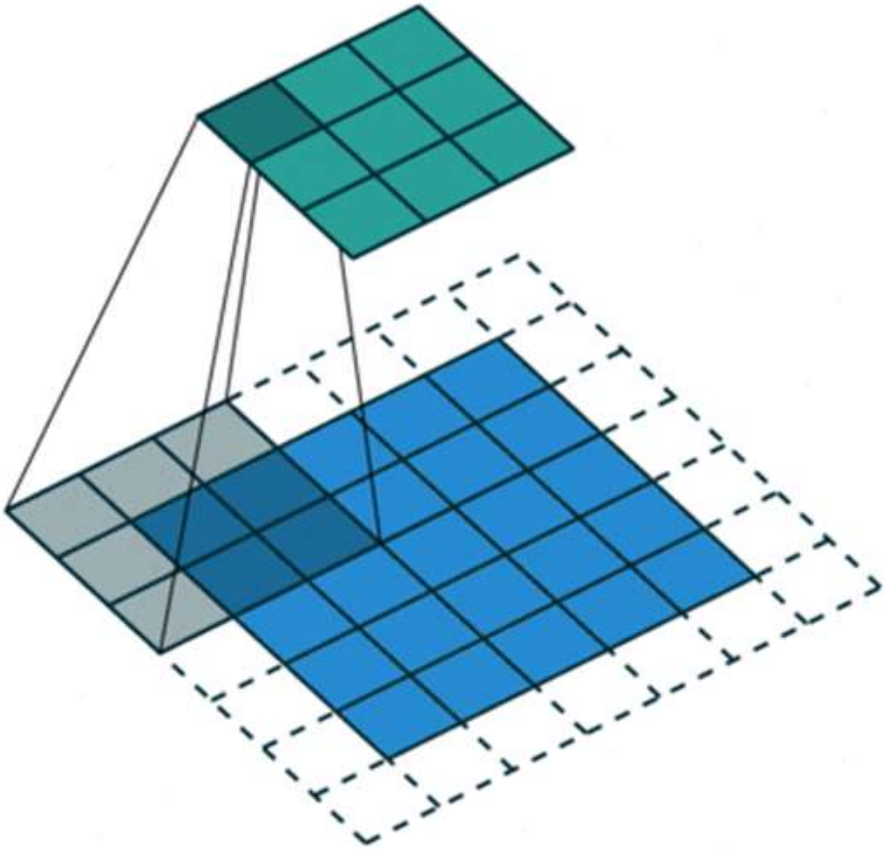


Figure 11. Convolutional neural network layer [37].

Within the domain of convolutional neural networks, there are a number of distinctive patterns that describe specific families of network designs. One such common pattern is that of Residual Networks.

Residual networks represent an architecture class in which the characteristic feature is that the output of each network layer is directly added to the input of that same layer [38]. Multiple such layers, or group of layers, are stacked together with each layer learning to correct the output of the previous layer, as pictured in Figure 12. Instead of attempting to create the perfect feature set, each layer instead is tasked to predicting the difference between the output of the previous layer and some more optimal feature space which better suits the target task.

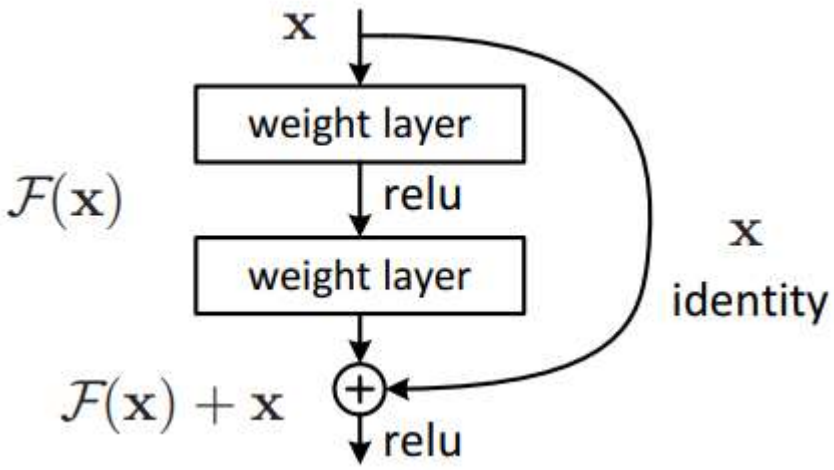


Figure 12. Residual Layer. Figure 2 from [38].

The most significant advantage of residual networks is their ability to overcome the challenge of the vanishing gradient problem [39]. The vanishing gradient problem is a mathematical

phenomenon that causes training deep neural networks to be highly difficult because the magnitude of the gradients being used to update model weights tend to shrink as they pass through more and more intermediate layers. This results in a situation where the layers close to the model output train quickly while the layers closest to the input receive very little updates. Residual networks solve this problem through their skip connections (identity mappings) which cause each layer in the network to more directly connected to the gradient source so that their weights may be more quickly and efficiently optimized.

Within the space of convolutional neural network designs, the Inception [40, 41] family of designs has been highly influential. Inception architectures arrange their layers into blocks with a highly complex and hand-designed set of layers, the outputs of which are concatenated together to create a set of features. An example of these carefully crafted blocks is presented in Figure 13.

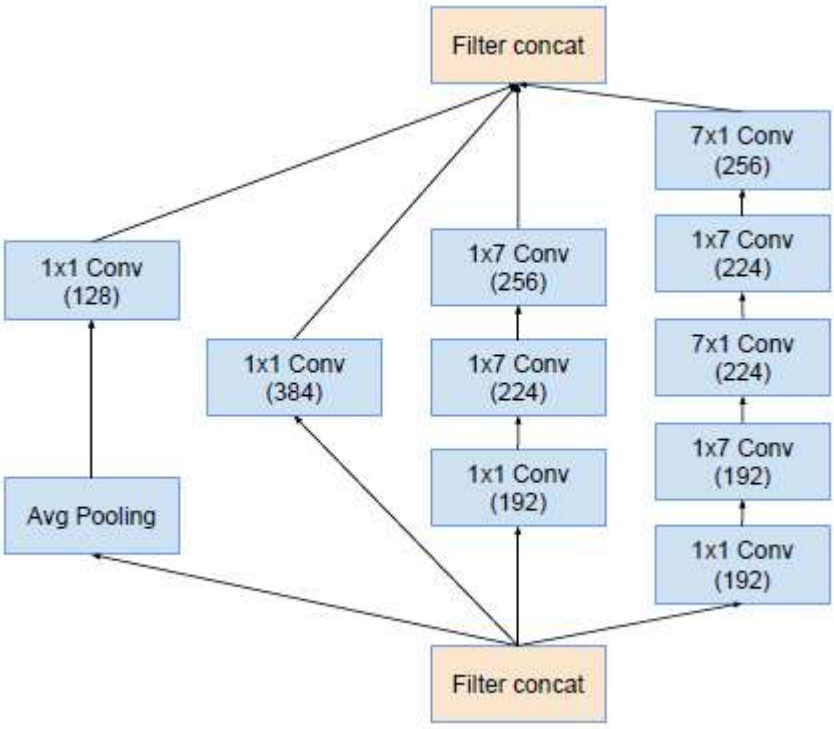


Figure 13. Sample Inception Module. Figure 5 from [40].

These hand-crafted modules were capable of outperforming many of the models which existed at the time of their publication and set many records for performance on various classification tasks. However, there are a number of disadvantages to Inception style modules such as the one pictured above. These modules must be hand-crafted by a human expert or through the expensive process of neural architecture search [42]; an iterative process which attempts to optimize the architecture of the neural network itself rather than just the model weights.

Recently, alternative architectures have begun to challenge convolutional networks in the domain of visual tasks. Attention networks [26] were first introduced in the domain of machine translation and have seen widespread adoption in various text-based domains. Attention networks make use

of the novel attention mechanism, which in the context of text processing allows a model to learn the relationships between each word or token being processed and compute an output in parallel instead of processing them sequentially using LSTMs [43] or other recurrent neural network variants. The details underlying the attention mechanism are presented in Figure 14.

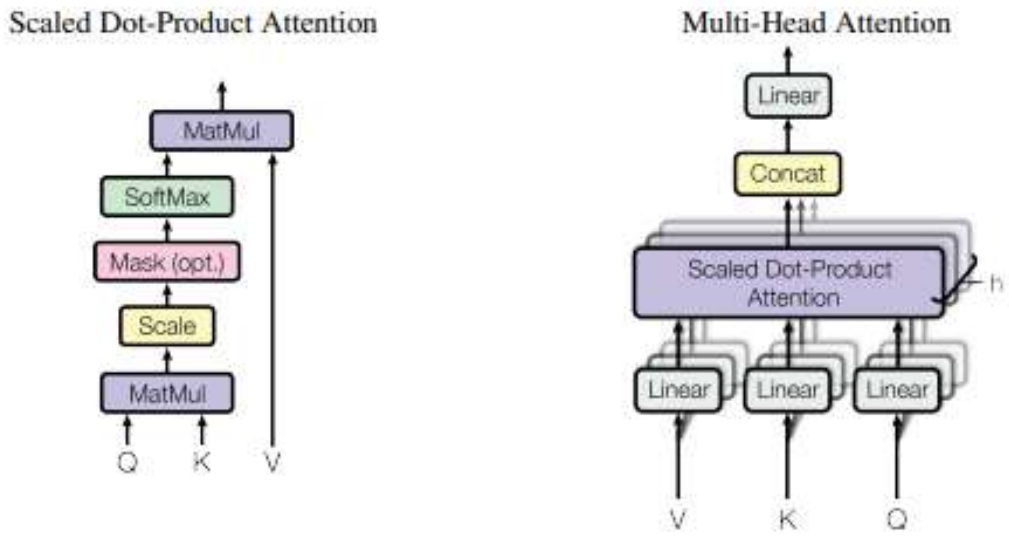


Figure 14. Dot-Production Attention Mechanism. (Figure 2 from [26]).

When this attention mechanism is extended to visual tasks, the resulting network architecture is known as vision transformers [44].These variants currently represent the latest innovation and hold the promise of significant performance improvements as novel methods are incorporated into machine vision tasks. The research on this topic is still ongoing and rapidly changing as new discoveries and methodologies are developed. Investigation and implementation of visual attention models should be considered a goal of future work. An example of a visual transformer for object detection is presented in Figure 15.

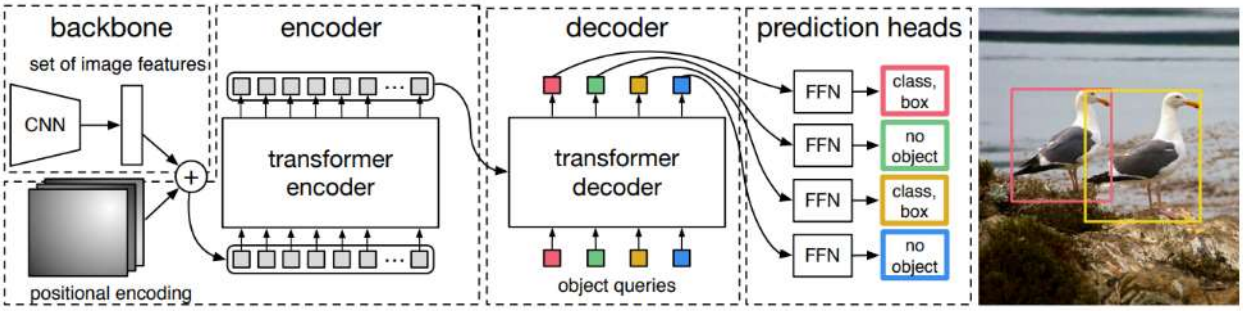


Figure 15. DETR Visual transformer architecture. Figure 2 from [45].

2.4 Related Works

Given the numerous potential applications for systems capable of performing visual ice characterization, it should come as no surprise that a number of closely related works exist which explore various methodologies in order to accomplish similar objectives. In this review, particular focus is given on related works which approach not only a similar task, but which must also accomplish their tasks via the methodology of machine learning with neural networks.

The related works discussed in this section approach sea ice characterization using multiple distinctive methodologies. The oldest work among this collection chose to approach ice characterization through multilabel image classification [46], in which a close range optical image would be fed into a convolutional neural network which would then produce a vector of the model's predictions for the probability that each ice class was present in the image. The work considered nine categories: icebergs, floebergs, floebits, floes, broken ice, level ice, brash ice, pancake ice, and deformed ice. Through a comparison of six different neural network architectures, the ResNet [38] family and particularly ResNet50 was the highest performer on the validation set;

obtaining 92% accuracy. Among the ice classes considered, the models struggled the most with the discrimination of pancake ice and brash ice.

Three of the related works approach the problem via the method of semantic segmentation, the same as is in this work. Within this group, two consider similar close range imagery captured from vessels while the third work utilizes top-down drone imagery.

The top-down drone imagery work, ICENET [47], focuses specifically on the Ningxia–Inner Mongolia reach of the Yellow River in China for the purpose of river ice monitoring. The work considered only three classes: ice, water, and others. The authors explore several custom neural network architecture choices by first constructing a network with residual connections and two parallel paths which fuse low-resolution and high-resolution information. Next, the authors add modules which enhance their models with dual attention [48]; the standard positional attention and also channel-wise attention modules. The baseline model achieved 92.9% pixel accuracy while their most sophisticated model obtained 95.9% pixel accuracy, demonstrating the capacity of attention within CNNs to enhance accuracy.

Next among the semantic segmentation group is a work which utilized imagery captured from an icebreaker on a two-month Antarctic expedition [49]. The data was processed into two separate datasets: the sea ice detection dataset and the sea ice classification dataset. The sea ice detection dataset contained 266 images which were annotated with four classes: ice, ocean, vessel, and sky. The sea ice classification dataset contained 1194 images which were annotated with seven classes: ocean, vessel, sky, lens artifact, first-year ice, new ice, and grey ice. The consideration of a lens artifact class is of particular interest and an important contribution of the work. An analysis can only be as good as the quality of its data, and as such it is vital to address the various situations in

which imagery may be partially or fully rendered unusable due to environmental and other effects. Two neural network models were tested in this work – PSPNet [50] and SegNet [51]. PSPNet was found to consistently outperform SegNet, obtaining an average 99.8% pixel-wise accuracy and 75.5% average IOU on the sea ice classification dataset as compared to SegNet's 96.7% accuracy and 53.6% IOU score on the same dataset.

The final item among the semantic segmentation group is the work which presented a multi-stage pipeline for the processing of close range optical images containing a variety of ice objects [52]. Their dataset was comprised of 338 images from online search engines plus 37 images captured during a research cruise to the Fram Strait. The dataset was annotated with a total of 14 classes: iceberg, floeberg, floebit, open water, melt pond, level ice, deformed ice, ice floe, pancake ice, underwater ice, broken ice, brash ice, shore, and sky. Similar to the steps described in Section 3.6.2, the work employed an ensemble of convolutional neural networks to perform predictions. Twelve unique neural network architectures were evaluated for use in this ensemble based on their individual performance. Through this process the architectures of PSPNet [50], PSPDenseNet, DeepLabV3+ [53], and UPerNet [54] were selected. Notably, the work makes use of convolutional conditional random field (ConvCRF) [55] as a postprocessing method which successfully demonstrated its capacity to improve ensemble outputs. Evaluation was performed using a 5-fold cross-validation scheme. PSPNet was found to have the highest individual performance. Inclusion of the ConvCRF postprocessing increased ensemble performance as measured by IOU score by approximately three percent. The full ensemble with postprocessing outperformed the baseline of PSPNet on average by a margin of two to three percent in terms of accuracy.

Chapter 3. Visual Classification of Sea Ice Features

The goal of this chapter is to detail the step-by-step process involved in the implementation of visual ice characterization using convolutional neural networks. This chapter roughly follows the seven-step machine learning process proposed by Yufeng Guo [56].

3.1 Data Collection

The data utilized in this task was collected from two Sony SNC-EB632R cameras mounted on piers of the Confederation Bridge. Each camera captures an image once every three seconds, for a total of 28,800 images per day. Not all of these images are useful, because they are also captured at night and other circumstances in which visibility may be extremely limited. The locations and approximate fields of view of the two cameras are presented in Figure 16.

The imagery was collected from the cameras via a local network installed on the bridge and stored on NRC servers. Each image was stored as an independent JPEG file, in 1920×1080 resolution. Unfortunately, the non-lossless JPEG format does lead to some instances of visible compression artifacting, as identified in Figure 17. Observe the subtle grid pattern visible at regular intervals across the region; this is a clear sign of JPEG compression artifacting.

At full scale the artifacting is barely noticeable to a human observer, only becoming evident when examining the image closely. Convolutional neural networks effectively do exactly that; examining the image at an individual pixel-level basis. As such, it is likely that this artifacting negatively affects the quality of the predictions output from the models.

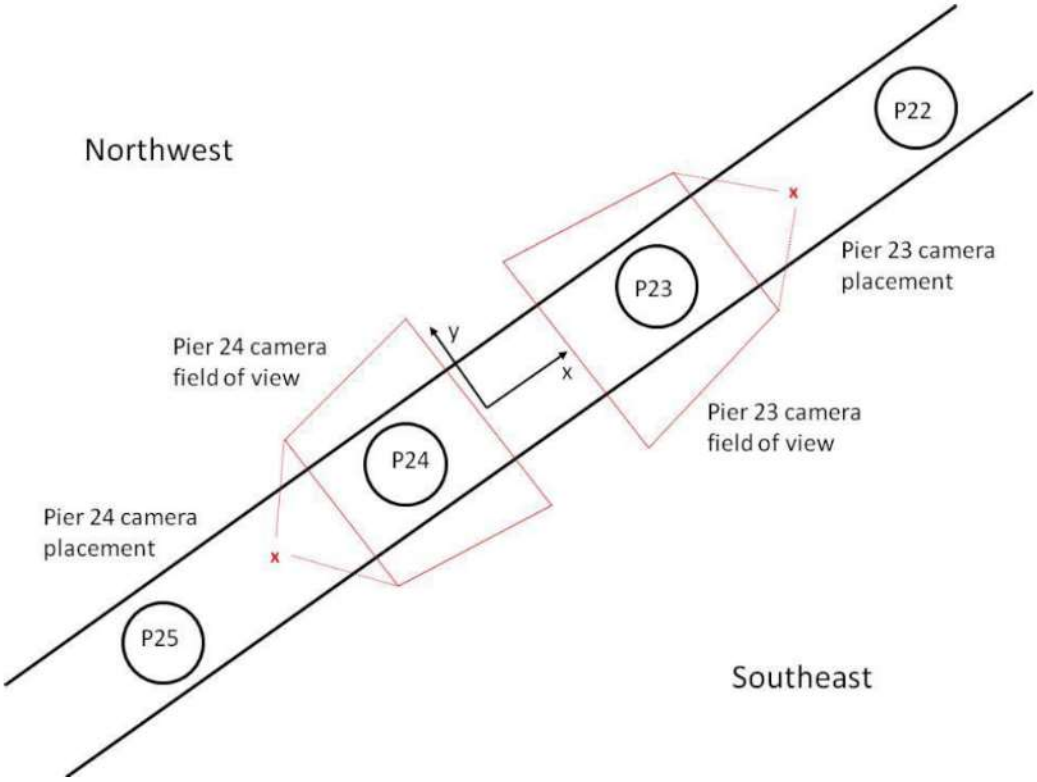


Figure 16. Approximate location and perspective of two cameras on the Confederation Bridge. Figure 22 from [57].

Figure 17. Sample 500 x 500 pixel image region showing JPEG compression artifacting.

3.2 Data Preparation

This section covers the steps required to prepare the dataset from its raw form into a format suitable for training and evaluation. The source images must be processed and the associated semantic segmentation maps produced, divided into training, validation, and test sets. Then, resizing and data augmentation may be applied.

3.2.1 Annotation

The segmentation maps were originally adapted from previous work [2]. These labels were produced using the VIA software package [58]. Annotations were performed in a polygon-centric approach, because the corresponding work applied instance segmentation rather than semantic segmentation. The data was stored in a JSON file containing a list of the polygons associated with each image. More information regarding the original annotation process is available in Appendix A. The polygon-centric data was rasterized by drawing each of the polygons onto a two-dimensional representation. Small gaps were noted between some of the polygons and these were infilled using morphological operators. This process yielded the 125 segmentation maps. Following rasterization, the images were reviewed to correct for any errors and to add all instances of the classes corresponding to open water or the bridge pier as these were not included in the first project.

An additional 130 segmentation maps were produced using the GIMP image editor software. The segmentation map was overlaid with the corresponding image in a separate layer and all pixels were labelled to match the observed conditions. These new additions were used to expand the size of the test and validation sets and to create the test set. Each segmentation map was stored as a separate PNG file.

Application of semantic segmentation in this task implicitly defines each of the classes to be of the *stuff* type, as discussed in Section 2.2.1. This distinction is not entirely clear and could in some instances be up for debate. In most images, the level ice within an image makes up a significant proportion of the image and is not easily distinguished into unique instances. However, there does exist images in which floes could be clearly defined into distinctive instances.

The colour code of the classes implemented in this work is provided in Figure 18 while an example of a particular sematic segmentation map is presented in Figure 19.

Figure 18. Semantic segmentation colour code.

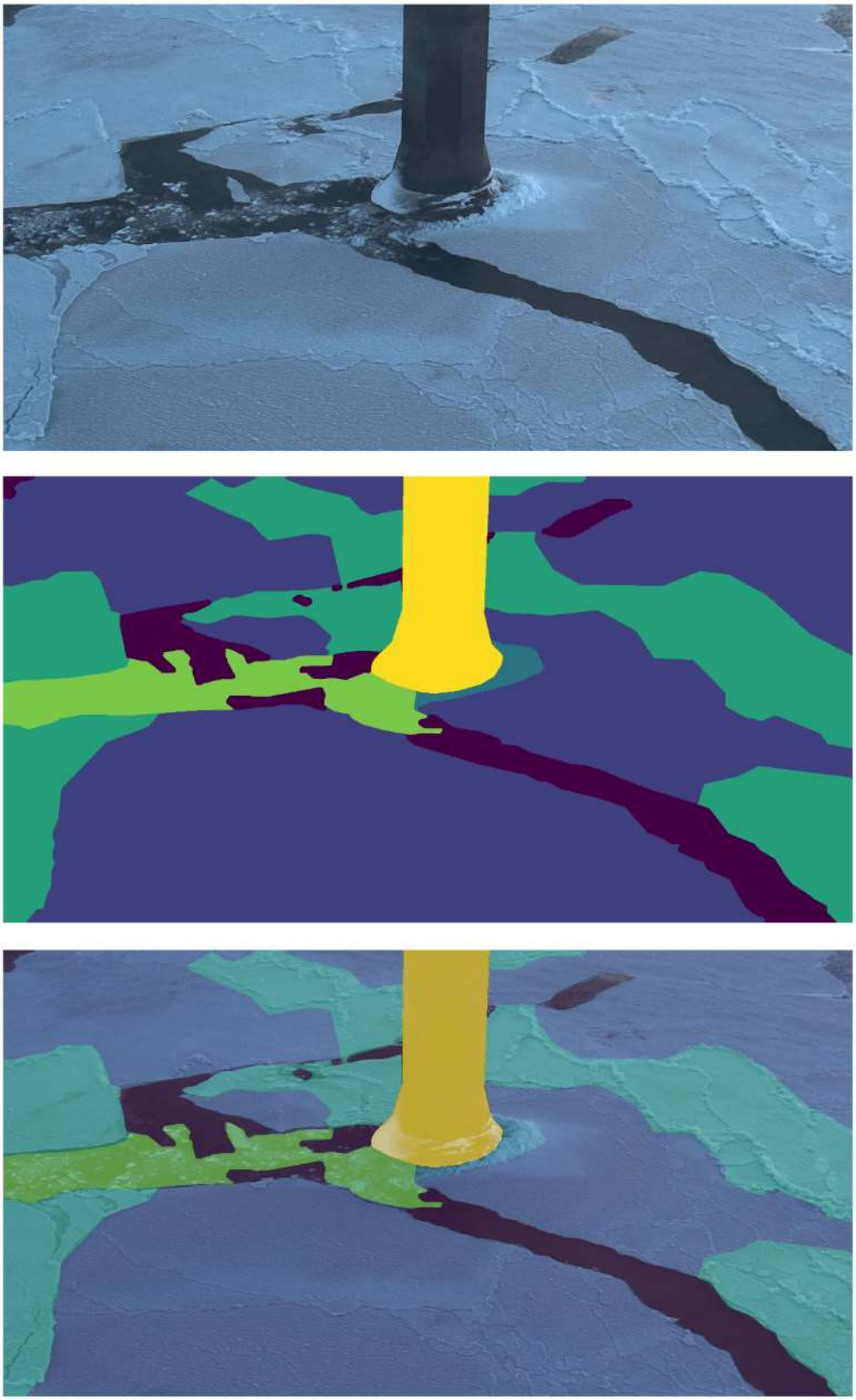


Figure 19. Sample image and associated semantic segmentation map.

3.2.2 Dataset Split

The training and validation datasets are derived from images previously selected and annotated as part of the prior work [2] and additional samples which were selected by human experts via a semi-random process. The two datasets were initially one group and precisely one fifth of this superset was removed for use in the validation set by selecting every fifth sample uniformly across the combined set.

The testing dataset consisted of samples which were selected using a fully random process which selected images from an archive of imagery which was made available to the author. Following the random process, human review removed a number of images with extremely limited visibility which were well in excess of even the most poor condition images from the training and validation sets. A number of still quite poor condition images were retained in the test set in order to evaluate the models' abilities to handle such samples and to represent real-world conditions.

The set of classes used for image annotation, training, and evaluation are presented in Figure 20.

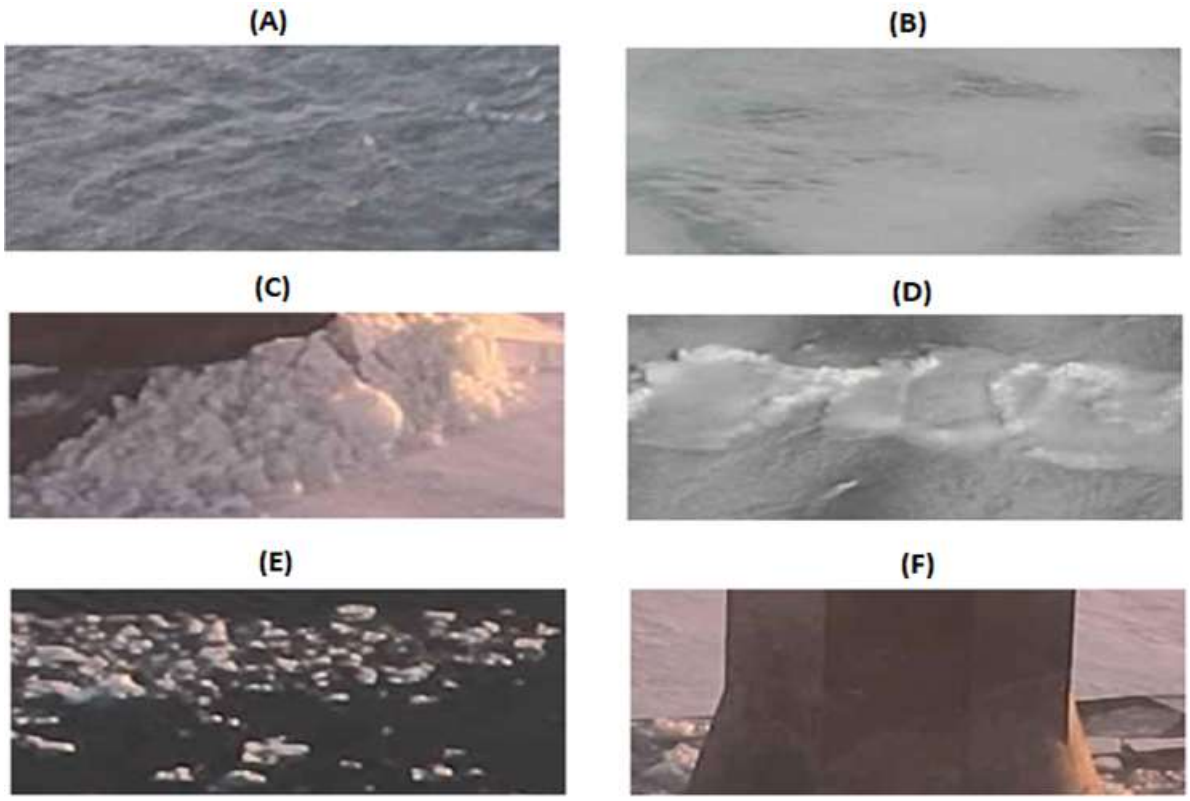


Figure 20. Dataset classifications. From top to bottom: A) open water, B) level ice, C) flexural failure, D) ridges, E) broken ice, and F) Other (pier).

In Figure 21, histograms detail the year, month, and time of day at which samples were captured for both the combined training and validation datasets and the test dataset. Certain limitations of the test set are evident. Only data from 2019 were available in the archive and no data from January or December was present. Additionally, there is an absence of samples between 6am and 8am local time.

These deficiencies of the test set are acknowledged and they may skew the final test-set evaluation metrics somewhat. Future work should ensure to expand the test set so as to rectify these deficiencies and in doing so increase the total size of the test set and correspondingly the confidence of the test-set evaluation metrics.

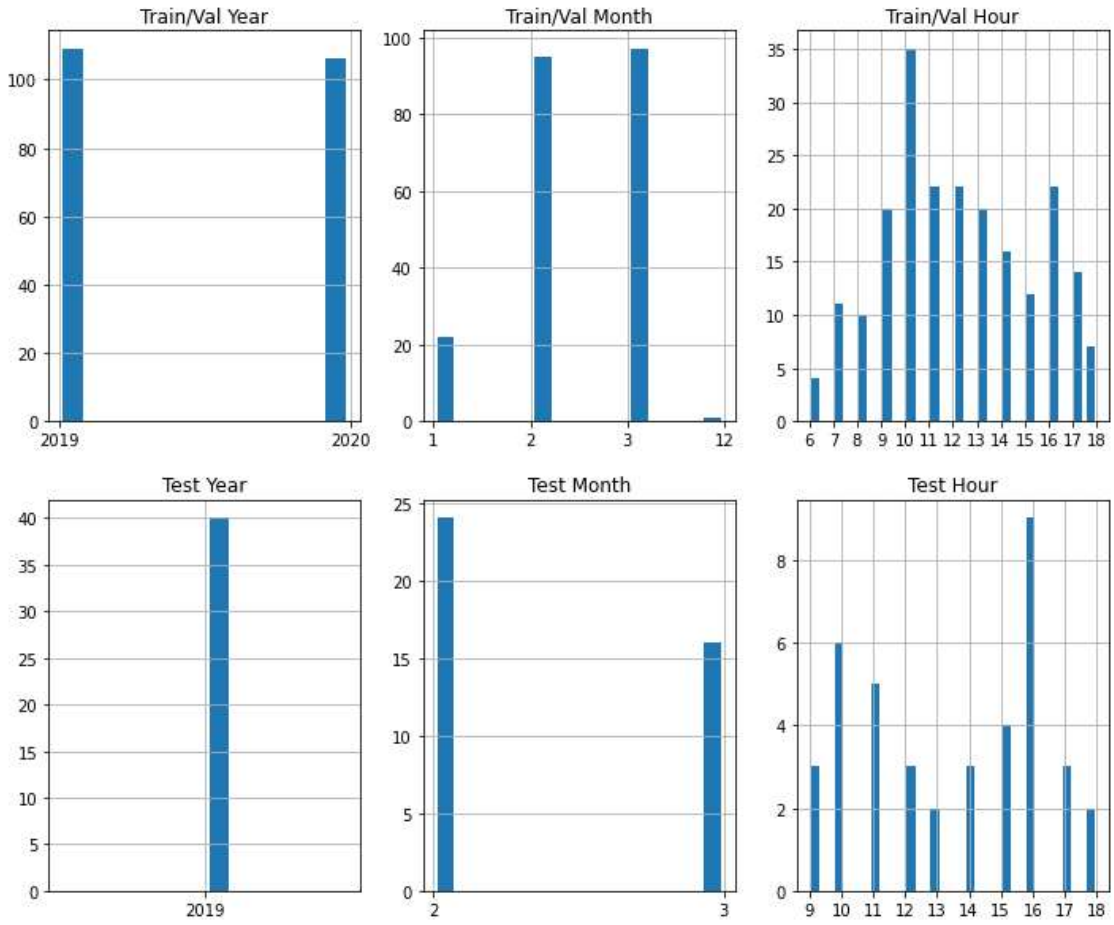


Figure 21. Train, Validation, and Test set distributions

The relative concentration of each class within both the training plus validation set and the training set are presented in Figure 22 and Figure 23 respectively. There are some differences in the distributions but they are overall quite similar. In comparison, the test set has less instances of the open water and broken ice class in favour of more level ice (floe) and ridges.

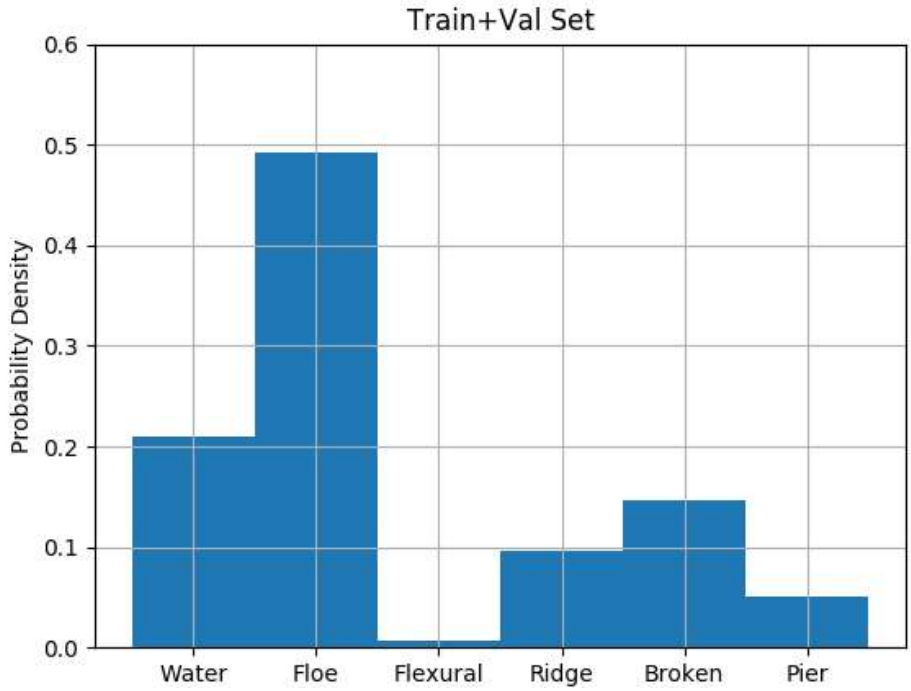


Figure 22. Training + Validation set category prevalence.

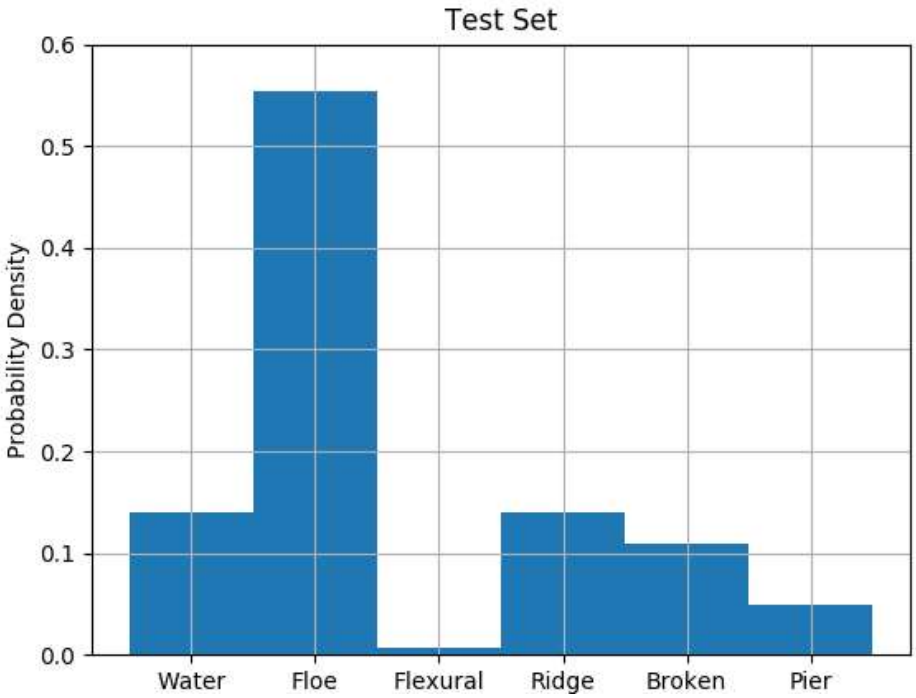


Figure 23. Test set category prevalence.

3.2.3 Preprocessing

Following this initial preparation of the datasets, the data was formatted for training and evaluation of the machine learning models. In order to remain within GPU memory constraints and to minimize training time, both the images and segmentation maps were downscaled from their original 1920×1080 resolution to 768×448 resolution. This resolution was picked because the neural network models require their inputs to be a multiple of 32 in size along both the width and height dimensions, while also approximately maintaining the aspect ratio. Following the reduction of resolution, the images and segmentation maps were collected into Numpy [59] n-dimensional arrays for quick and efficient access. The TensorFlow [1] data API was utilized to apply preprocessing and augmentations at training time.

Contrast limited adaptive histogram equalization (CLAHE) [60] was applied to all images before being passed through any of the methods described in the following sections. Previous works have used CLAHE for real world visual systems, including improvement to visibility during foggy conditions [61]. A comparison of a foggy image before and after application of CLAHE are presented in Figure 24.

Figure 24, Foggy image before (left) and after (right) application of modified CLAHE. Adapted from figures 5 and 6 of [61].

However, the anti-fog effect of CLAHE is beneficial, it was not the main motivation for the use of CLAHE in this work. Rather, CLAHE was adopted primarily as a method to reduce the effect of shadows which are cast by both the pier itself and the bridge above at various times throughout the day. The left side of Figure 25 compares an original image to the output of regular histogram equalization and CLAHE. It is clear that CLAHE performs the most optimally in both enhancing the visibility of the ice surface while also lightening the shadows, especially in comparison to the generic histogram equalization output.

The histograms represent the intensity or brightness values, derived by transforming the images from RGB to HSV format. It is evidently clear that histogram equalization favors the extrema of the distribution, while CLAHE more smoothly distributes intensity values across the whole range.

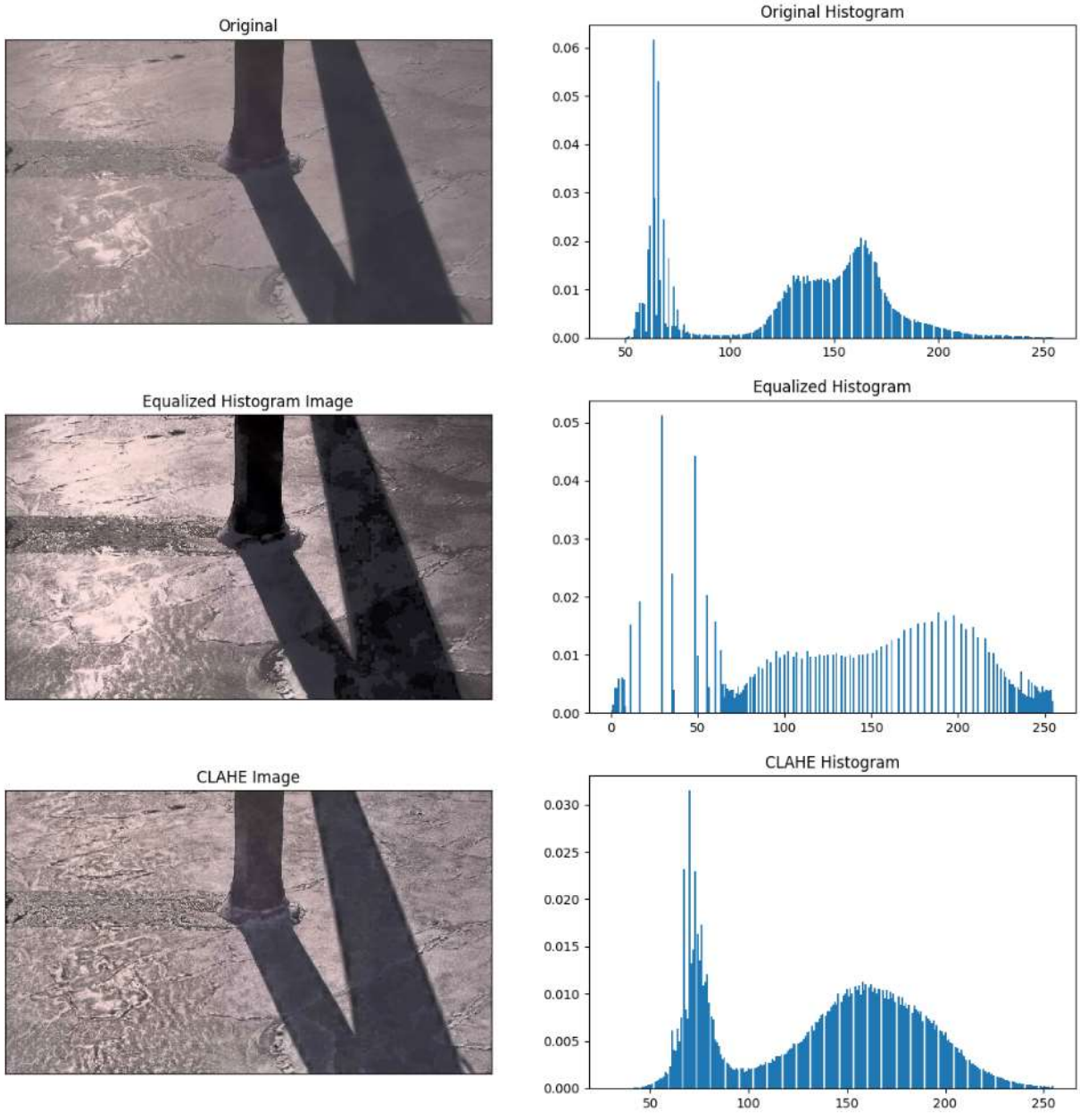


Figure 25. Demonstration of the effects of histogram equalization and CLAHE.

3.2.4 Data Augmentation

Data augmentation is a common technique in the development of deep-learning based computer vision systems which entails the modification of images at training time in order to increase the robustness of the system while reducing overfitting to the training set [62]. Data augmentation allows the number of training samples to be effectively increased while only using data available within the training set [63]. Common augmentations which are applied during model training include translation, rotation, horizontal flipping, the addition of Gaussian noise, and the modification of saturation, hue, and intensity values. Examples of an image from the Confederation Bridge dataset with various augmentations applied are presented in Figure 26.

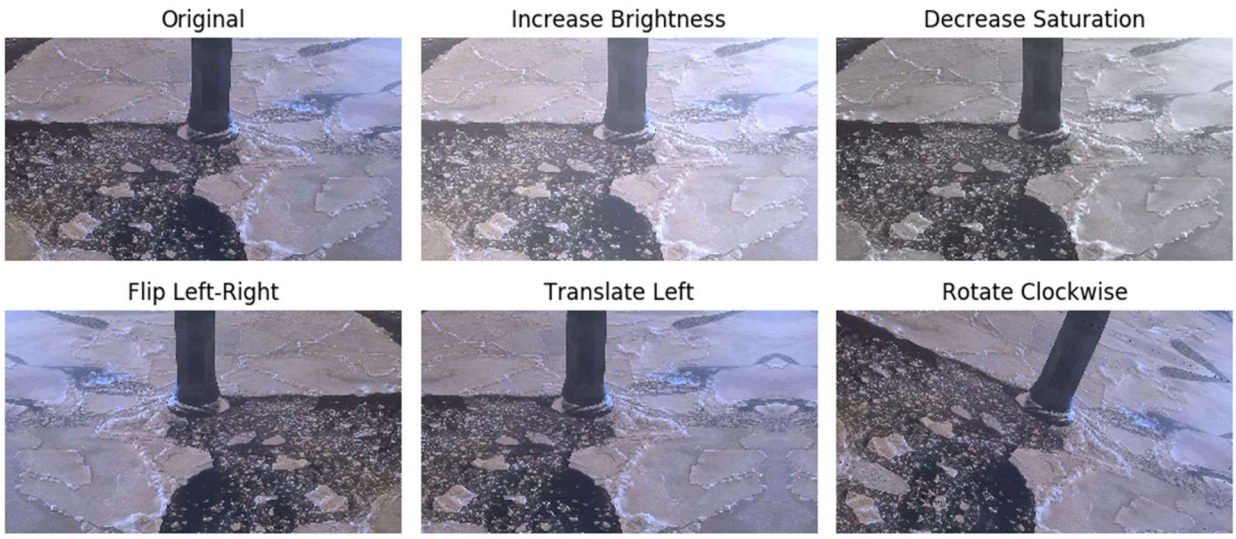


Figure 26. Sample Image Augmentations.

In addition to decreasing overfitting on the training set, data augmentation also plays an important role in further improving the model's generalization abilities. Changes in brightness and saturation are particularly useful because shifts in these overall image conditions are comparable to the

differences in images captured under different lighting conditions such as at different times of the day, including dawn and dusk.

Following the augmentations described above, the mixup [64] augmentation technique is applied. While the principle of mixup is relatively simple, it presents a number of benefits. In essence, the core technique of mixup involves the linear blending of two input alongside their corresponding targets. Let Xi and Xj represent two independent image from the training set, and let Yi and Yj represent the corresponding semantic segmentation maps. Next, mixup may be defined according to Equation 1 and Equation 2.

$$X' = \lambda X_i + (1 - \lambda)\mathrm{Xj} \qquad \text{Equation 1. Mixup input blending}$$

$$Y' = \lambda Y_i + (1 - \lambda)Y_j \qquad \text{Equation 2. Mixup target blending}$$

The resulting X' and Y' are then used in the current step of training, with X' as the current input and Y' the target output.

The variable λ is randomly drawn from the beta distribution, pictured in Figure 27. The distribution is bounded between zero and one. The beta distribution is characterized by two variables, α and β. When the two variables are equal, the distribution will be symmetric. Increasing one variable relative to the other will cause the distribution to skew towards zero or towards one. While holding the two variables equal, increasing their magnitude will increase the prevalence of values in the middle of the range. Conversely, decreasing their magnitude will cause the majority of drawn samples to be near to zero or one. Based on experimentation, the values of α=β=0.075 was selected.

This small value causes the resulting images to mainly consist of one image or the other, with only rare instances of heavily blended variables. Further exploration and optimization of this value should be considered in future work.

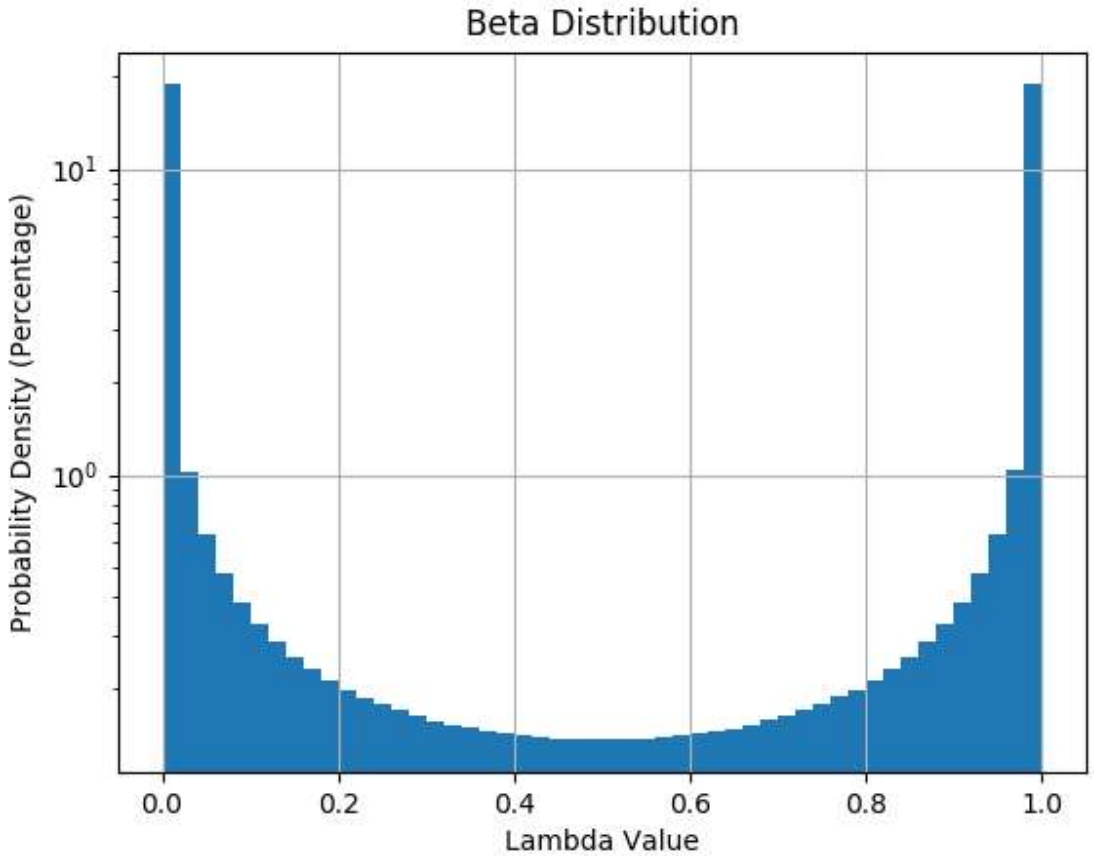


Figure 27. Beta Distribution, with α=β=0.075. Note that the Y-axis is on a logarithmic scale.

The authors of the mixup method claim a number of substantial benefits from its application. These benefits are outlined by the authors as "mixup helps to combat memorization of corrupt labels, sensitivity to adversarial examples, and instability in adversarial training.", "mixup leads to decision boundaries that transition linearly from class to class, providing a smoother estimate of

uncertainty.", and that "mixup improves the generalization of state-of-the-art neural network architectures " [64]. An example of mixup in practice is presented in Figure 28.

It is more difficult to visualize the effect of mixup on the segmentation maps. Consider a pixel in the upper left-hand corner of the resultant segmentation map. In image A, the pixel is labelled as open water while in image B, the pixel is labelled as level ice. This same pixel in the resultant segmentation map would represent open water with seventy percent confidence and level ice with thirty percent confidence – these are the target values towards which the model would be encouraged to predict. Accordingly, a pixel which is labelled as open water in both images would retain a label of open water with full confidence.

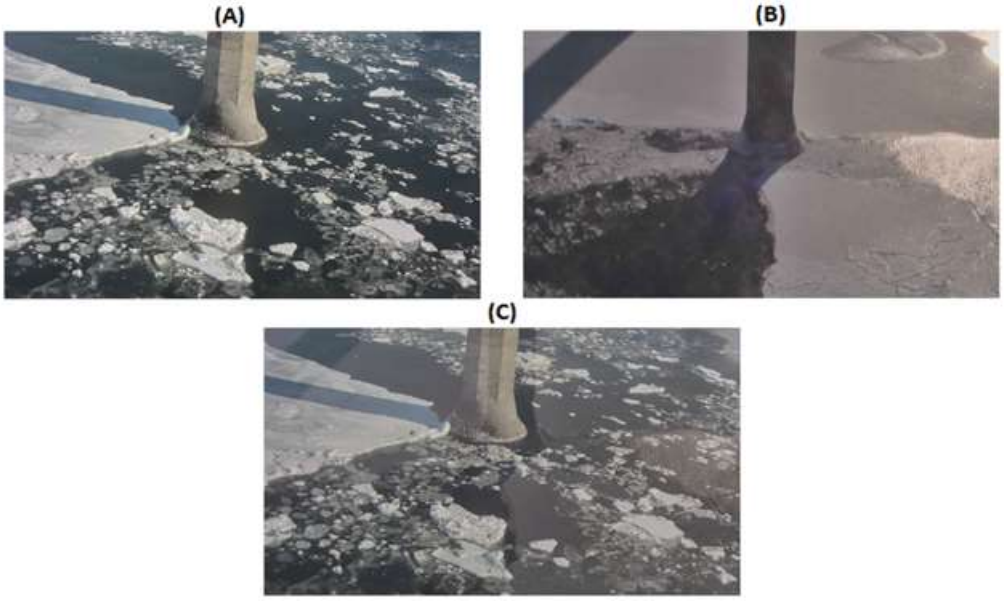


Figure 28. Mixup Example. Upper Left: Image A. Upper Right: Image B. Bottom: 70% A + 30% B.

3.3 Model Selection

A number of publically available model architectures were evaluated for use in this project. The library, “Segmentation Models” [65] was used to facilitate the use of these models contains four separate decoder architectures and ten pretrained encoder architectures, many of which include multiple versions with various degrees of computational complexity. These decoder and encoder architectures may be combined interchangeably. Three models from each of the four encoders are trained, for a total of twelve models. This conglomerate of models combine to form the system depicted in Figure 29.

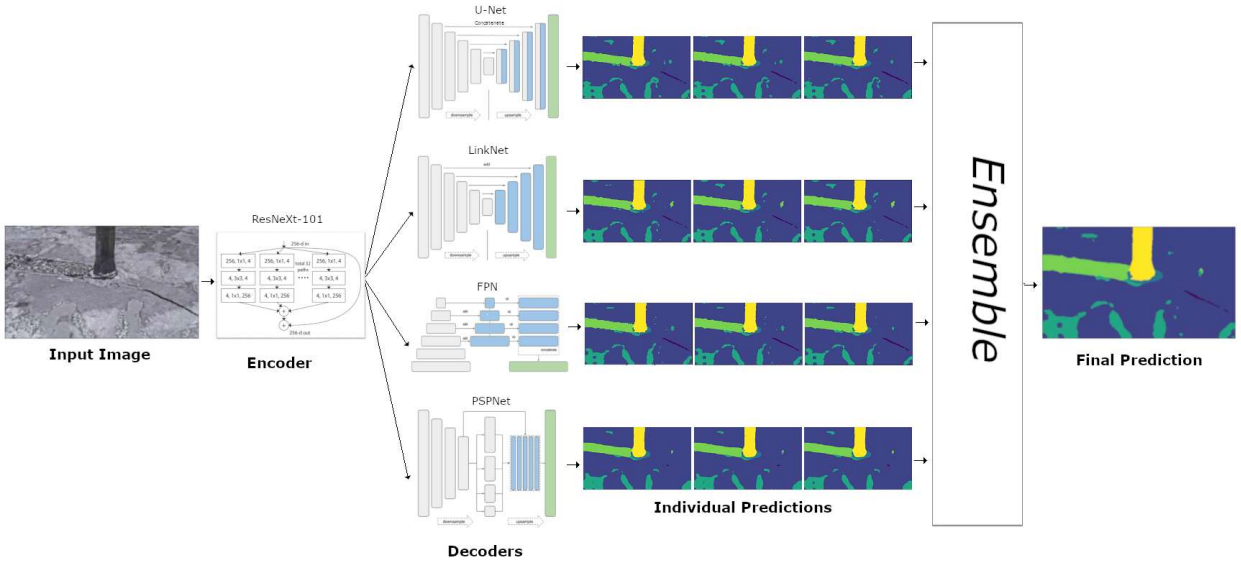


Figure 29. Full system diagram.

During preliminary stages of this work, each of the available decoders were tested in combination with each of the available encoders. The results of this process lead to the selection of the encoder and decoders discussed in the following subsections.

3.3.1 Encoder

The encoder is the portion of a neural network which processes an input image into a set of abstract features which may be then transformed into a final prediction.

Through an exhaustive search, ResNeXt [66] was identified as the encoder architecture presenting the highest performance on the validation dataset. ResNeXt is a highly performative residual [38] convolutional neural network structure originally designed for the task of image classification on the ImageNet dataset. The more computationally expensive ResNeXt-101 was selected rather than the more minimal ResNeXt-50, in order to increase learning capacity and as a calculated decision to increase model performance at the cost of greater computational requirements.

As such, each of the models employed in this work utilize a ResNeXt-101 model pretrained on the ImageNet dataset [67] as their encoder module. More details on the ResNeXt architecture are provided in the following paragraphs. The models differ only in terms of which decoder is utilized to transform the extracted features into the semantic segmentation output.

ResNeXt draws inspiration from the Inception family of architectures [40, 41]. Instead of complex designs found in Inception models, ResNeXt refactorizes the module to a more simple design. This design is shown to empirically outperform both Inception and ResNet models while presenting stronger representational power. Furthermore, ResNeXt outperforms models with even twice the computational complexity [66] – in doing so clearly presenting a more efficient strategy for visual understanding. The design of one ResNeXt block is presented in Figure 30.

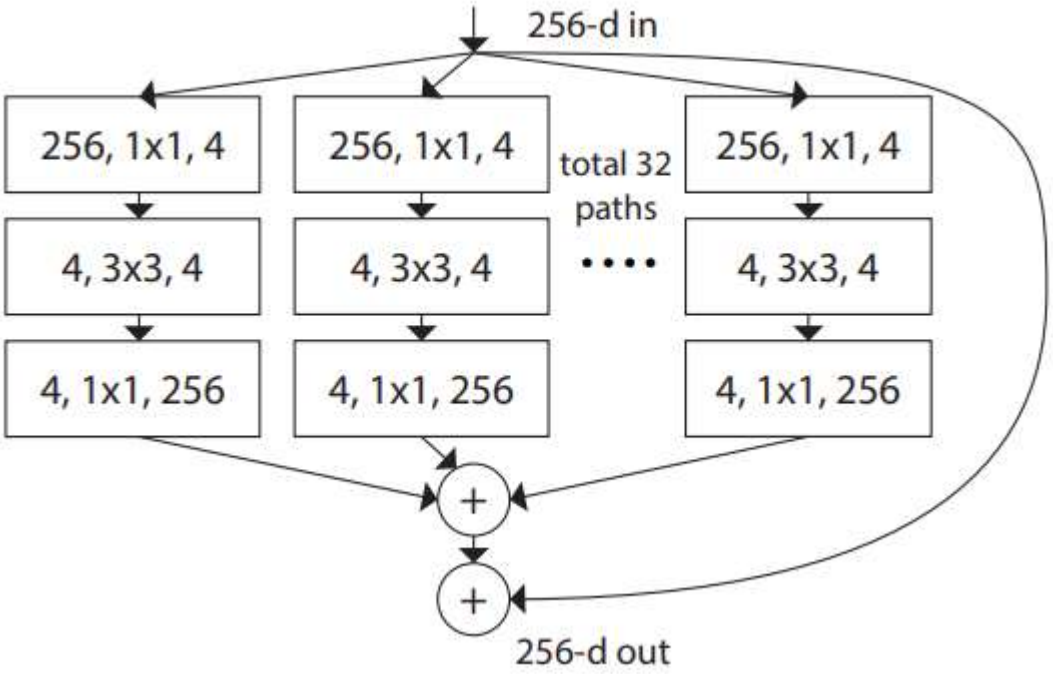


Figure 30. ResNeXt Block, Figure 1 right side from [66].

The above configuration is mathematically described in Equation 3. The function T_i represents the learned transformation computed by each of the individual paths, while *C* represents the cardinality of the block. In this case there are 32 paths in total, representing the cardinality of this block. Cardinality is a new parameter introduced by the ResNeXt authors, which they show to be a more important dimension of greater importance than the traditional parameters of width and depth.

Equation 3. ResNeXt transformation [66]

$$y = x + \sum_{i=1}^{C} T_i(x)$$

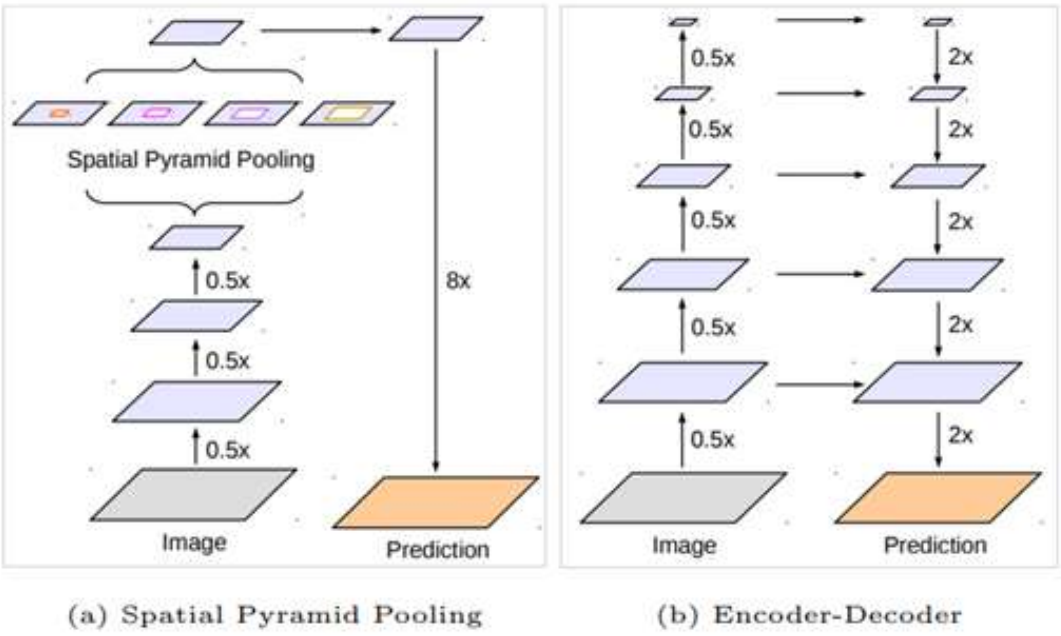
Each block first projects the high-dimensionality input into a significantly lower dimension embedding using a convolutional layer with a one by one receptive field, equivalent to a single dense layer being passed over each pixel independently. The depth of this embedding is known as the bottleneck size. Next, each path transforms the embedding using a convolutional layer with a three by three pixel receptive field, while maintaining the same depth as the initial embedding. Finally, the transformed embedding is projected back into the original dimensionality. The set of transformations created by each path are then summed together along with the original output.

The above describes the behaviour of a single ResNeXt block. The ResNeXt-101 variant used in this work had a total of 101 such blocks, as this is how the naming scheme is defined. Periodically, the resolution along the width and high dimensions are reduced by half before being input to the next block. The features just before each of these dimension reductions are those extracted for use in the decoder.

3.3.2 Decoder

The decoder is the portion of a neural network which interprets the features produced by the encoder into a final prediction. In this work, four different neural network decoder architectures are employed. The decoders in question are those found in the architecture of Feature-Pyramid-Network (FPN) [68], U-Net [69], LinkNet [70], and Pyramid-Scene-Parsing-Network (PSPNet) [50].

Pyramid-Scene-Parsing-Network may be considered a variant of (A) from Figure 31; Feature-Pyramid-Network, Linknet, and U-Net may be characterized as variants of (B) from Figure 31. The primary difference between these two architectures is based in the manner in which they combine information at different scales.



(a) Spatial Pyramid Pooling (b) Encoder-Decoder

Figure 31. Semantic segmentation architecture overview. Adapted from figure 2 of [53]

The B type architectures produce features at each resolution in order, and then starting from the smallest resolution repeatedly applies convolutional layers(s) to combine them with features produced from the next larger resolution's set of features; this is repeated until all the available feature resolutions are exhausted or the desired output resolution has been obtained.

The A type architectures utilize a much different approach for the fusion of information from different resolution features maps. The feature maps produced by the encoder at each resolution are first reduced along the features dimension, using a convolutional layer with 1x1 size receptive field. Next, each feature map is scaled to the same size, using bilinear interpolation [50], and concatenated together before being fed through one last set of convolutions to produce the final predictions. This approach has both advantages and disadvantages; the rich set of features available

to the final layer(s) allow for highly accurate predictions, but the spatial distortions introduced by the rescaling can cause the network to produce less accurate boundaries between objects [53].

More detailed architecture illustrations from the original source materials of U-Net, LinkNet, FPN, and PSPNet are presented in Figure 32, Figure 33, Figure 34, and Figure 35 respectively. One may observe that FPN and LinkNet are nearly identical in concept, but differ in the exact choice of convolutional layers which are employed for the merging of data and upsampling of resolution. Furthermore, note that only the final, largest resolution output head of FPN is utilized instead of the three different resolution output heads illustrated in the left side of Figure 34.

U-Net combines encoder and decoder features via concatenation. The encoder features of a certain resolution are concatenated with decoder features of same resolution, both of which are then passed through a convolutional layer and then upsampled to produce the next higher resolution's decoder features.

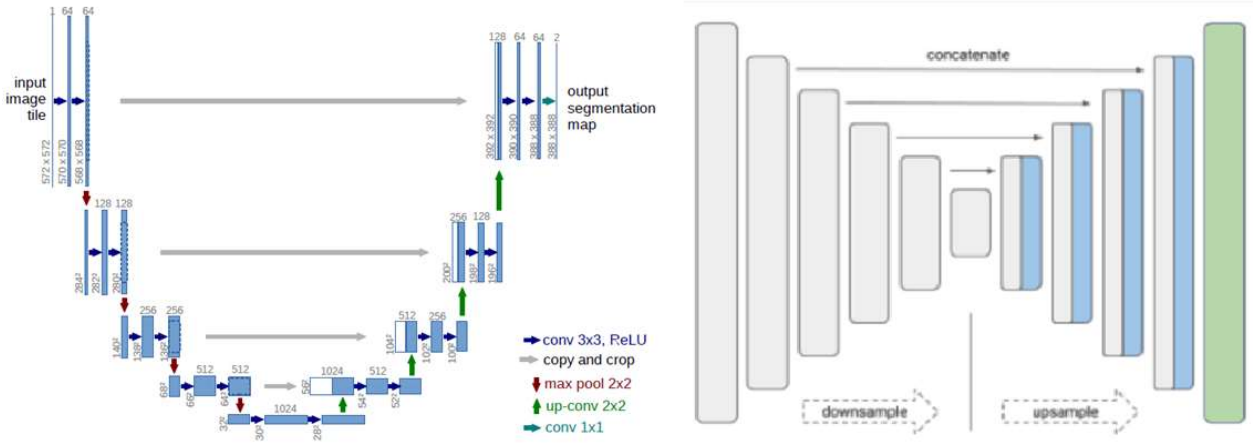


Figure 32. U-Net architecture. Left side is Figure 1 from [69], right side adapted from [65].

Similar to U-Net, LinkNet combines encoder features with the same resolution decoder features in order to produce the next higher resolution level decoder features. The critical difference is that LinkNet combines features using addition rather than through concatenation. This difference allows for LinkNet to have a slightly lower GPU memory and computational complexity requirements, but constricts the possible set of features that can be produced within the decoder.

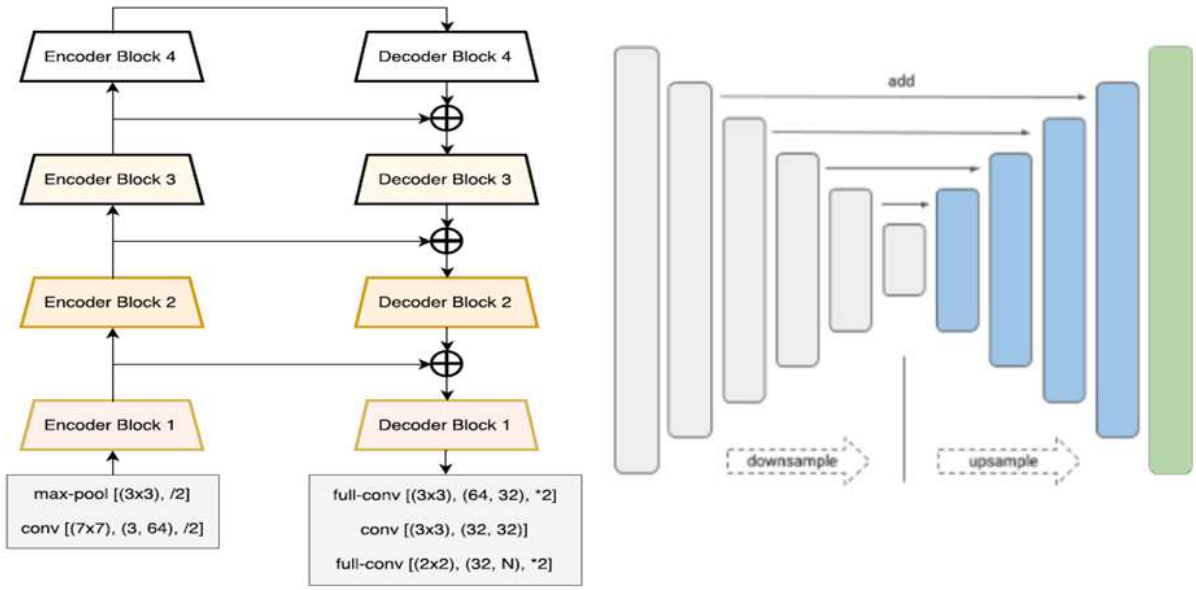


Figure 33. LinkNet architecture. Left side is Figure 1 from [70], right side adapted from [65].

The design of Feature Pyramid Network (FPN) is initially very similar to that of LinkNet, because encoder and decoder features are combined at multiple scales through the use of addition. However, FPN differs in that after processing the features at a certain scale, it then proceeds to also produce a prediction at that same scale. These multiple predictions are each upsampled to full resolution and then concatenated together before a final convolutional layer produces the output predictions.

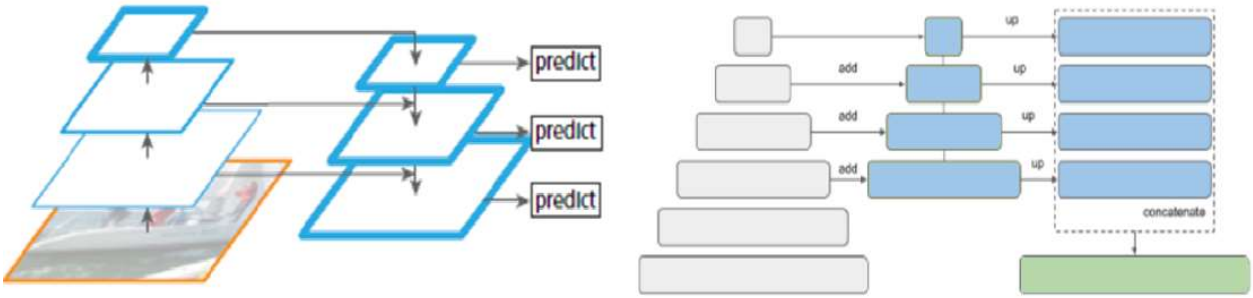


Figure 34. Feature Pyramid Network architecture. Left side is Figure 1D from [68], right side adapted from [65].

PSPNet differs the most from the other three decoder architectures, because it employs a completely different system for the combination of features produced at different resolutions. Rather than processing each resolution sequentially, each scale of encoder features are processed at the same time using the pyramid pooling module. In doing so, the pyramid pooling module produces a very rich set of output features which capture both global and local context. The outputs of the pooling module are then concatenated with the original features with the same resolution before being passed through a final convolutional layer which produces the final prediction. This final layer has access to a very rich set of information which allows for highly accurate predictions. However, excessive pooling operations can lead to the loss of fine-details such that very small class instances may not be captured in the final prediction.

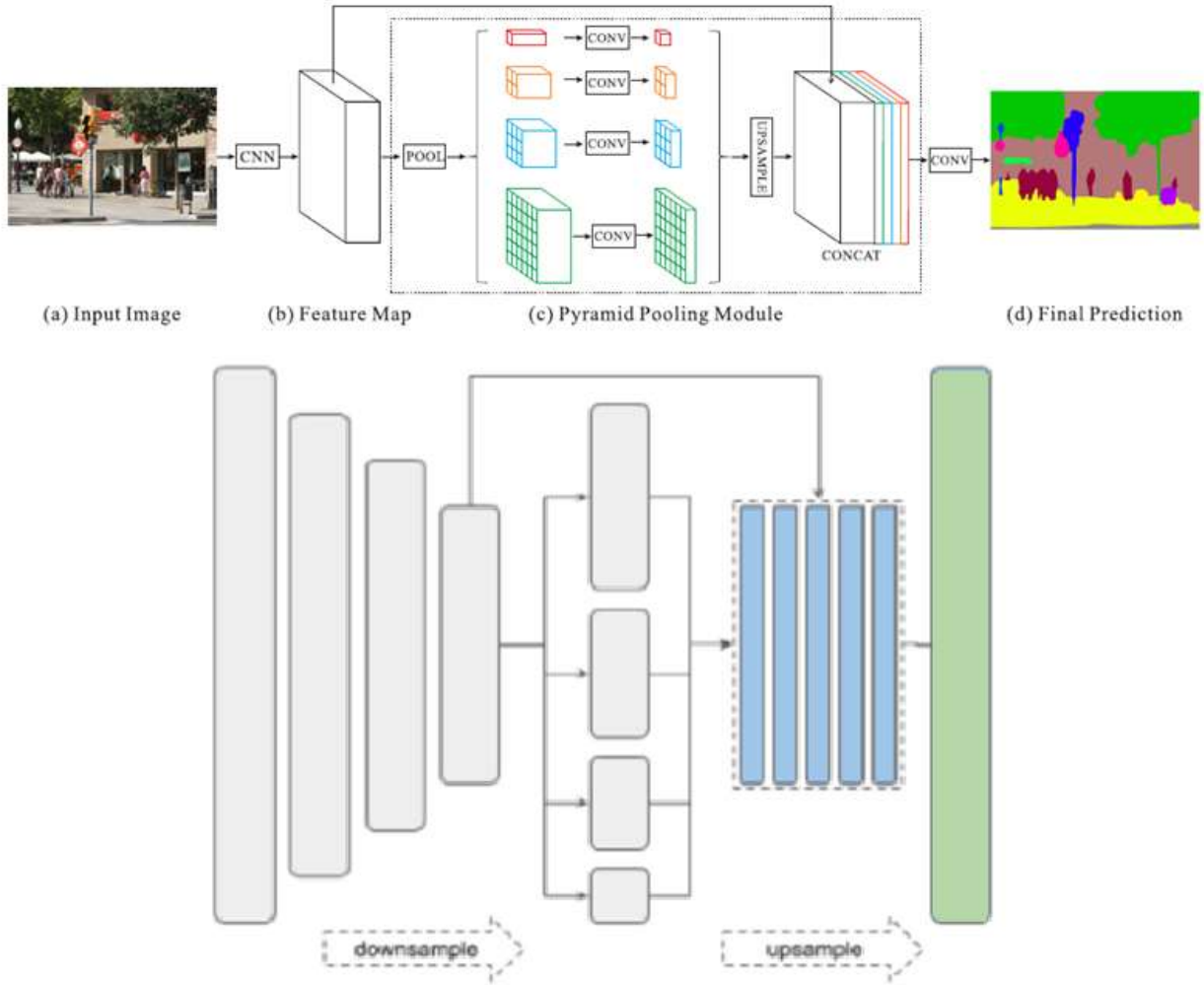


Figure 35. Pyramid Scene Parsing Network architecture. Top is Figure 3 from [50], bottom adapted from [65].

Each of the decoder architectures employed has their own unique benefits and disadvantages, and are sufficiently different that they will produce slightly different final predictions. Through the combined applications of all of these decoders, the overall system becomes more robust and accurate.

3.4 Training

This section describes the specific steps undertaken to train each of the models which will later constitute the ensemble of models. Both supervised learning and semi-supervised learning are employed.

3.4.1 Supervised Learning

Supervised learning begins with model initialization. The encoder module is ResNeXt-101 with pretrained weights sourced from training on the ImageNet dataset. The weights of the decoder module are randomly initialized. The training process then begins in full as samples are fed into the model and the model weights are updated using backpropagation of the errors between the model predictions and the true labels. A specific optimizer must be chosen to determine the fashion in which each model weight update is performed. In this work the chosen optimizer was stochastic gradient descent (SGD) with momentum [71] due to its robustness and ability to find more generalizable optima [72], despite the cost of additional training time when compared to ADAM. As such, the model weights are updated according to Equation 4 with each batch of training. The set of model weights are denoted as Θ, the learning rate is λ ; the momentum term is μ, and is set equal to 0.9 throughout. L represents the loss function, P represents the model's predictions for the current batch, and G_T represents the ground truth data of the current batch. For the initial batch of training, V_0 is set to zero. A batch size of 4 was selected.

$$V_i = \mu\, V_{i-1} + \lambda \nabla_\theta L(P; G_T)$$

Equation 4. SGD with momentum

$$\Theta_i = \theta_{i-1} - V_i$$

The loss function employed in this instance is dice loss [73] [74]. Dice loss directly optimizes the F1-score. This loss function is more appropriate than common classification losses such as binary cross-entropy due to the class imbalance within the dataset. Any loss function which does not in some way account for class imbalance will cause predictions of the most common classes, level ice floes, to be substantially more accurate at the cost all of the less common classes. Dice loss and its differentiable approximation are defined in Equation 5.

$$L_{Dice} = 1 - \frac{2 * True\ Postives}{True\ Positves + False\ Negatives}$$

Equation 5. Dice Loss Function

$$L_{Dice}(P;\ G_T) = 1 - \frac{2\ \sum_N P * G_T}{\sum_N P^2\ + \sum_N G_T^2}$$

The weights of the encoder module are frozen for the first five epochs of training so that the randomized decoder weights may learn to make use of the features already produced by the pretrained encoder. Freezing the encoder during this period prevents its already finely-tuned weights from being wildly updated by the large gradients produced initially due to the highly inaccurate predictions produced by the randomly initialized decoder. The learning rate is held at a constant value during this stage of training. Only after five epochs of training and the model has begun to converge towards steadily improving performance are the encoder weights unfrozen. Once unfrozen, the encoder may begin to learn visual features which may have been less relevant to their original task and better adapted to the ice imagery. The progression of training loss, accuracy, and f1-score during this initial training is presented in Figure 36.

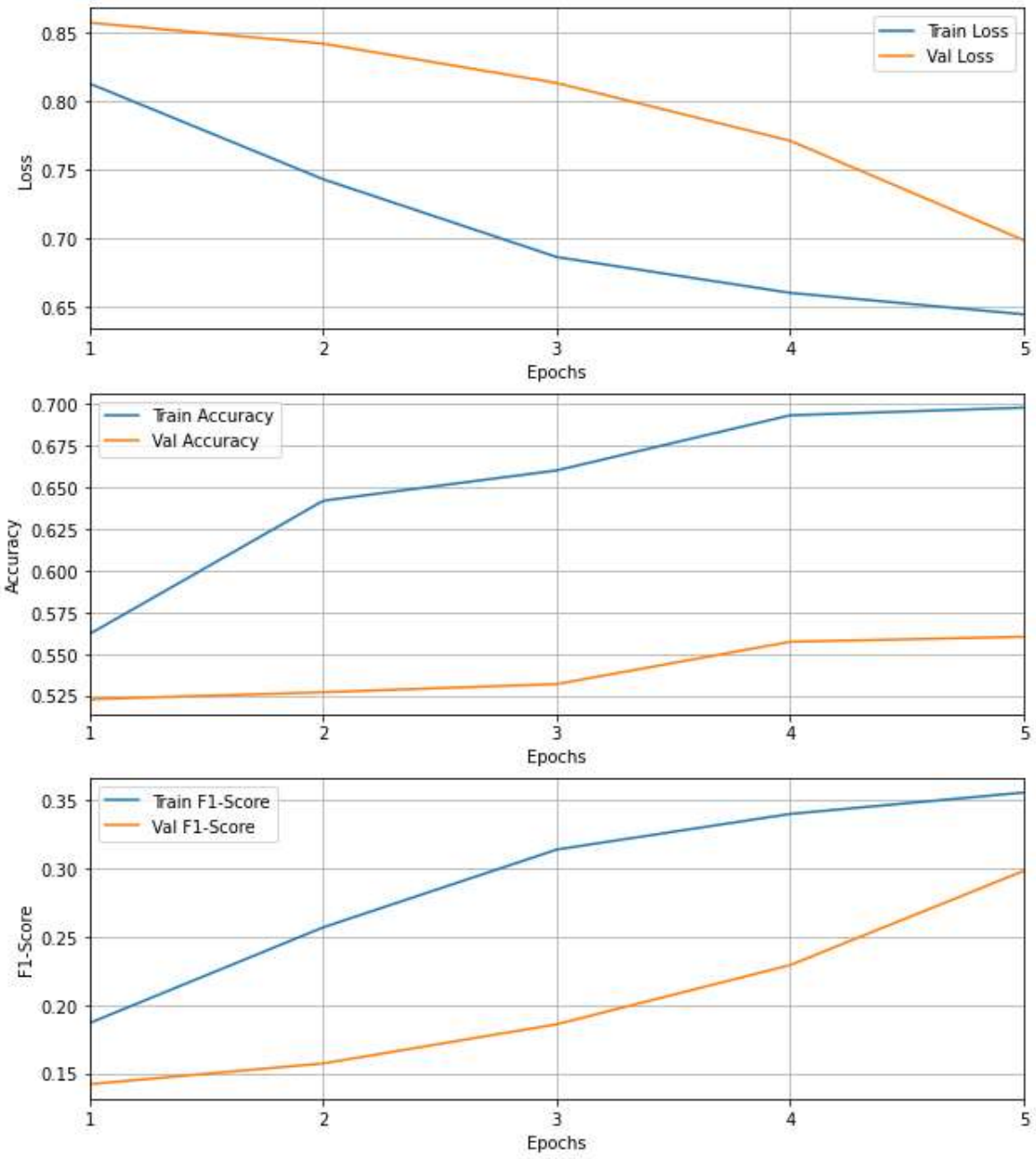


Figure 36. Progression of warmup training process.

Following the initial warmup training and the unfreezing of the encoder weights, the main training procedure may begin. The training progresses for a total of five hundred epochs. During this stage the learning rate is adjusted constantly using a cosine annealing function [75] which defines the learning rate at each epoch according to Equation 6.

The value of λ_{max} is $1.2 * 10^{-3}$, the value of λ_{min} is $2 * 10^{-4}$, T_i is the current epoch number, and the value of T_c is 50.

$$\lambda_i = \lambda_{min} + \frac{1}{2}(\lambda_{max} - \lambda_{min})\left(1 + cos\left(\frac{T_i \,\%\, T_c}{T_c}\,\pi\right)\right)$$

Equation 6. Cosine annealing function

The purpose of this cycling learning rate function is to facilitate snapshot ensembling [76]. This method allows for one training cycle to produce multiple models fit to find different optima within the learnable model parameter space [76]. The progression of the main training process, along with the corresponding learning rate at each epoch, is presented in Figure 37.

Training for 500 epochs with a cycle period of 50 epochs leads to exactly ten learning rate cycles within the main training procedure. A snapshot of the model's state is saved just before the learning rate increases, for a total of ten snapshots. Only the final three snapshots are fine-tuned and included in the final ensemble because each subsequent snapshot tends to find a better optima than was found in the previous cycle, and to maintain the computational and time complexity of the training process as well as the final system itself within acceptable confines.

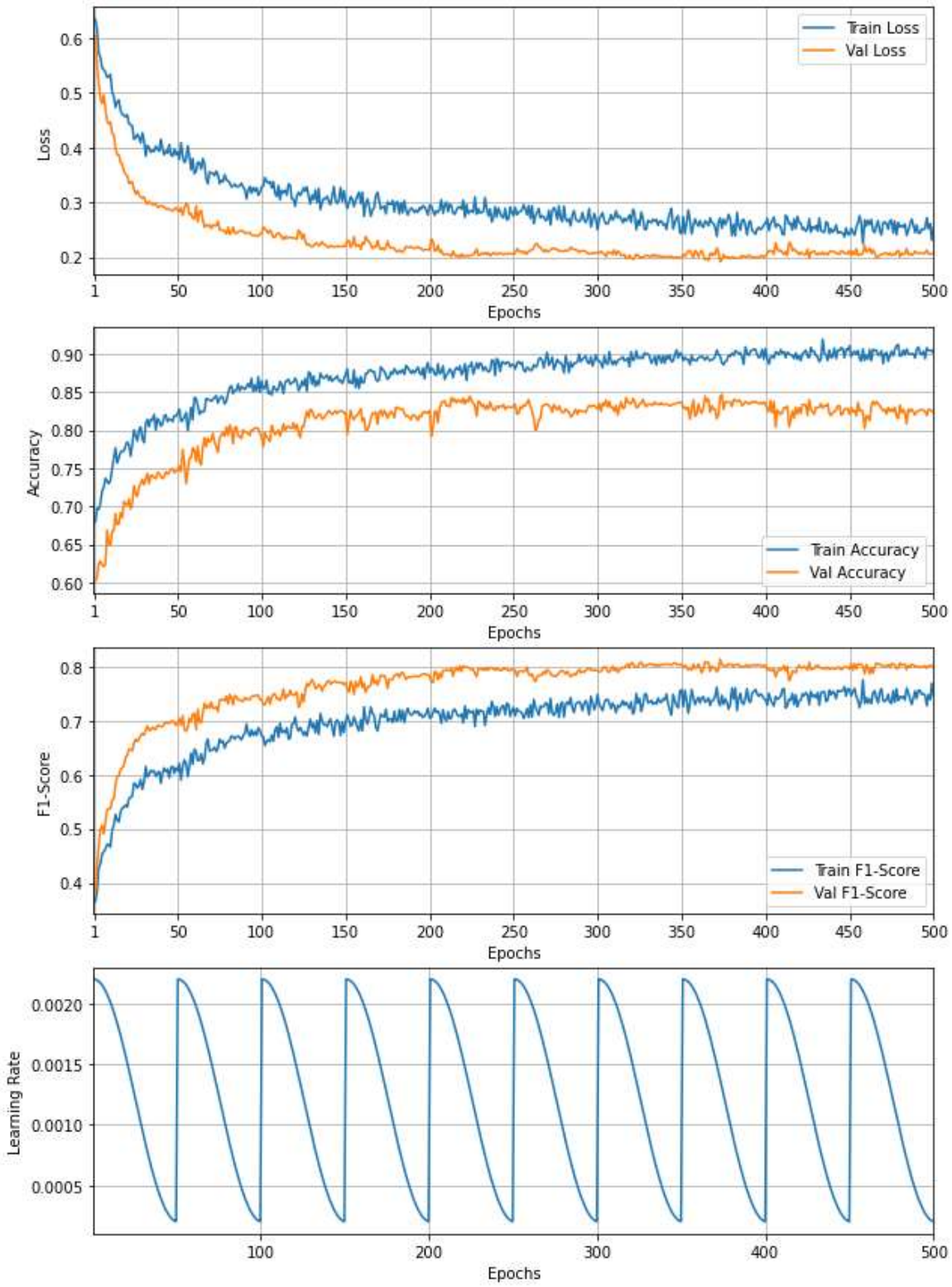


Figure 37. Progression of main training procedure.

The example in Figure 38 illustrates the learning process of a theoretical model with the learnable weights on the bottom axis and the corresponding loss on the vertical axis. The learning rate is initially high and decays until its minimal value is reached, allowing the model to converge towards a local minima similar to a standard decaying learning rate schedule. Then, the learning rate suddenly increases back to the initial high value. This causes the model to overshoot the current optima and further explore the parameter space – ideally to settle in a different optima as the learning rate again decreases.

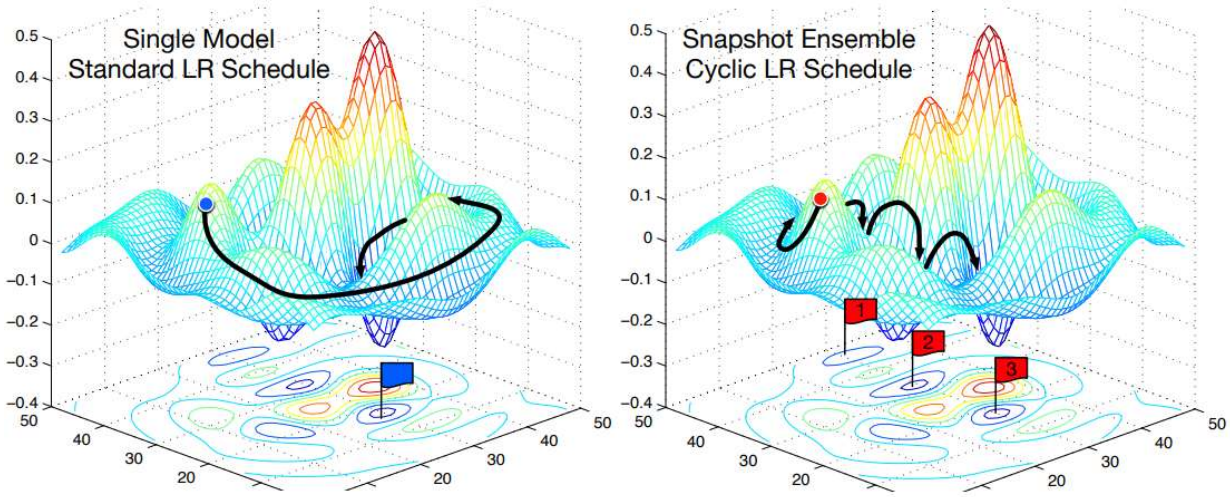


Figure 38.Comparison of standard learning rate schedule and cyclic learning rate schedule with snapshots. Figure 1 from [76].

Finally, the progression of the fine-tuning training process is presented in Figure 39. The validation metrics improve only marginally while the training metrics continue to improve throughout – suggesting a certain degree of overfitting at this stage.

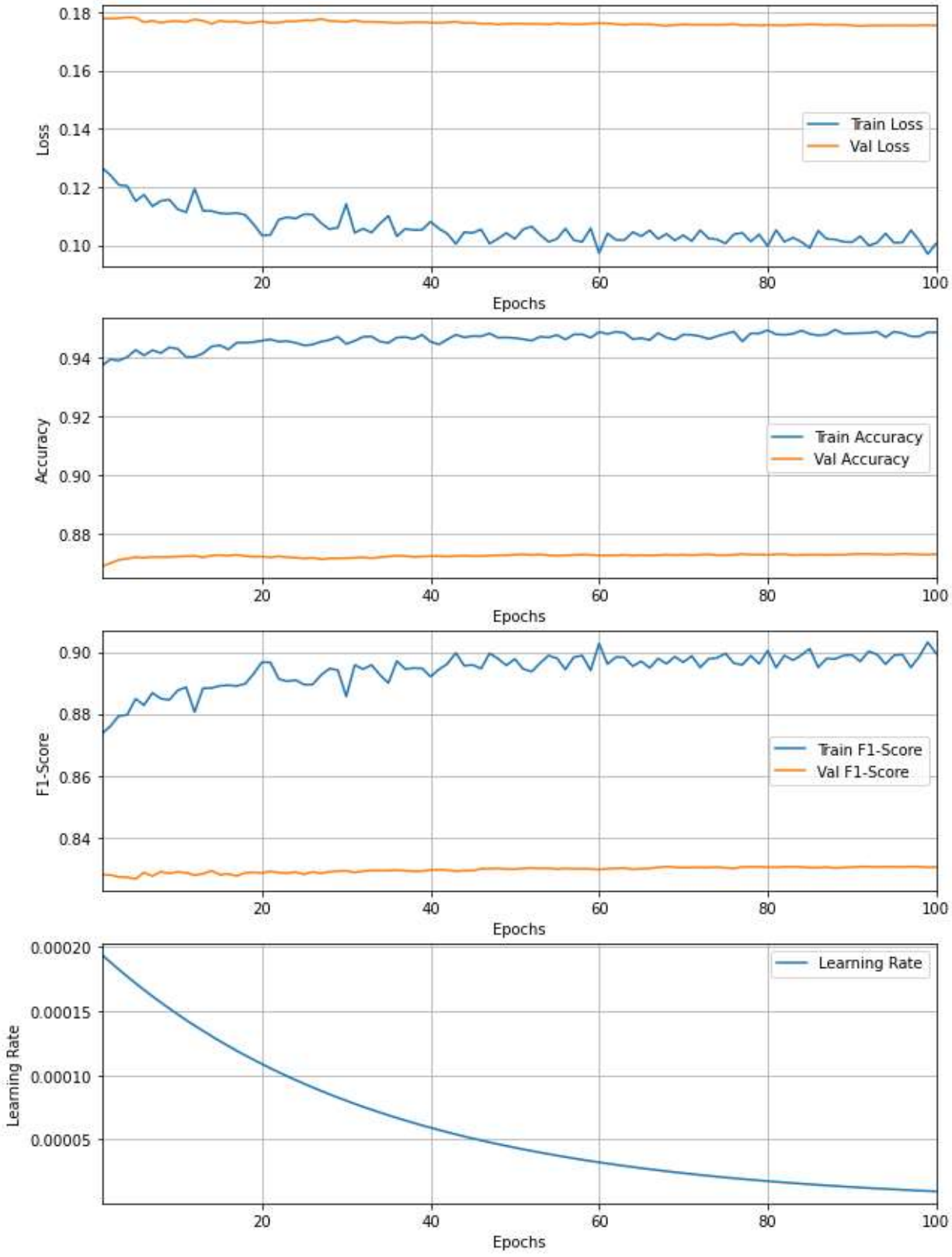


Figure 39. Progression of fine-tuning training procedure.

3.4.2 Semi-supervised Learning

Semi-supervised learning is an extension of supervised learning. It is employed in this work to allow the system to benefit from the vast amount of unlabelled data while improving system performance.

In the case of semi-supervised learning, it is assumed that a secondary set of input data is available which does not have corresponding labels. In many cases, the unlabelled dataset may be substantially larger than the original labelled dataset because the costly and time consuming process of obtaining data labels is not required. This work adopts the semi-supervised learning procedure proposed in [77].

The first step of semi-supervised learning is to proceed with a regular round of supervised learning, using the labelled training set to train a model or ensemble of models to perform the task at hand. Next, the model(s) are used to predict on each sample from the unlabelled dataset. As these predictions do not necessarily reflect the truth, they are considered to be pseudolabels.

In the next step, a new round of training is initiated. In this round, the unlabelled data and associated pseudolabels or some subset thereof are concatenated with the original training data and true labels. In the case that only a subset of the pseudolabels are utilized, they are selected based on the instances in which the model(s) predicted the pseudolabels with the greatest degree of confidence – this helps to mitigate the presence of low-quality or noisy samples. Once the labelled and pseudo-labelled data have been merged, the original training process from the initial supervised learning process is repeated from scratch, including random re-initialization of the model(s). This overall process is visualized in Figure 40.

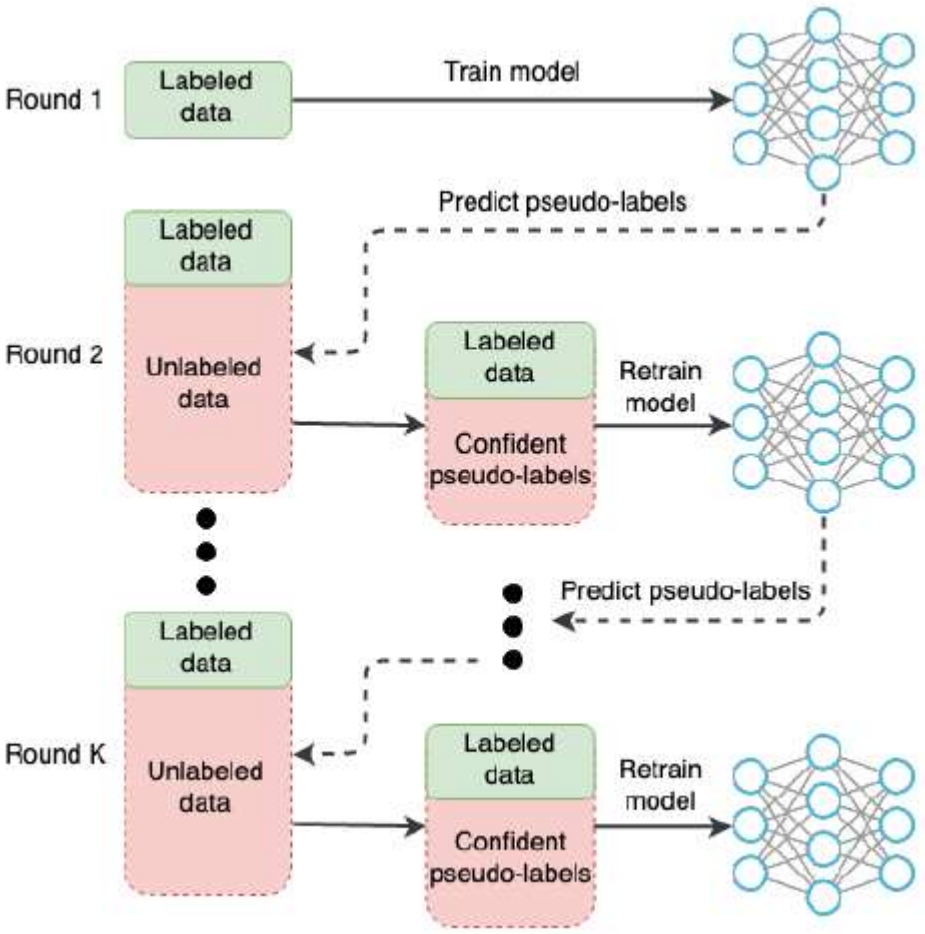


Figure 40. Semi-supervised training routine. Figure 2 from [77].

The above process may be repeated an indefinite number of times, with each round the quality of the pseudolabels is expected to increase such that after training completes the measured performance upon the validation or test set improves with each subsequent round of training. As the number of rounds of iterative training increase, it should be expected that the rate of improvement slows down as the marginal utility of each round of training decreases. This is supported by the evidence presented in Figure 41, showing diminishing returns across one round of supervised training and the two subsequent rounds of semi-supervised training that were

performed. Time and computational limitations were the driving factor regarding the choice to conduct only two rounds of semi-supervised training.

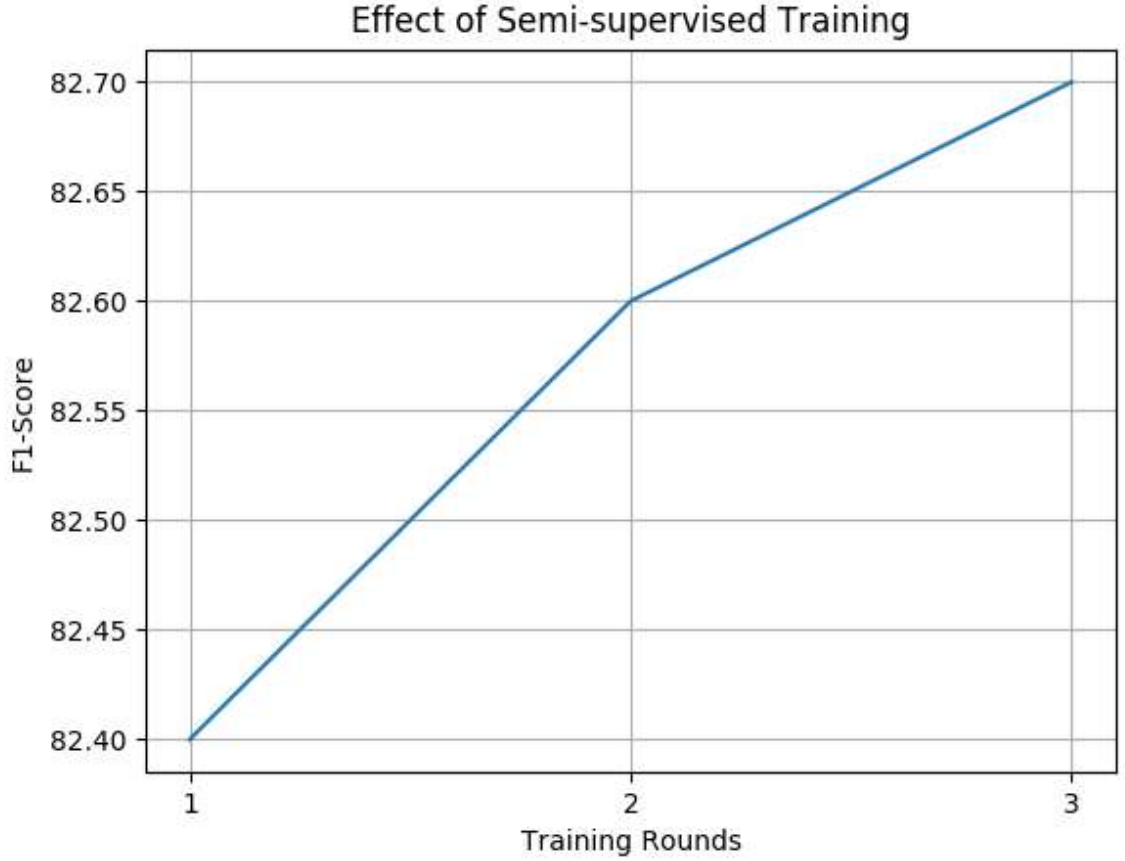


Figure 41. Effect of three rounds of semi-supervised learning on F1-score.

It stands to reason that repeating an increasingly large number of rounds of semi-supervised learning would not be able to increase accuracy to an arbitrary degree. As such, repeated rounds of semi-supervised learning would be likely to slowly increase performance and either approach some asymptote, begin to oscillate, or degrade performance due to error amplification.

3.5 Evaluation

This section details the performance of the individual models which have been trained as detailed in the previous sections. A total of twelve models are considered. The validation-set metrics of

each individual model are presented in Table 1. The bolded results highlight the best performance in each metric. An identification number is assigned to each model for the purpose of future reference to specific models. Models with ID numbers 2, 6, 9, and 10 are the models with highest validation-set performance of each architecture. These selected models are subsequently evaluated individually on the test set, with their metrics presented in Table 2.

Overall, models with the PSPNet decoder architecture have the highest performance on both the validation and test sets. However, the gap between PSPNet and the other architectures is much less on the test set in comparison to the validation set.

The metrics used to evaluate the models are: accuracy, top-2 accuracy, precision, recall, f1-score, brier score, and IOU. Accuracy is simply the percent of correctly classified pixels over all images. Top-2 accuracy is the percent of all pixels over all images for which the correct answer was among the two most likely predicted classes. Precision, recall, and f1 score are closely linked and are important in the context of datasets with high class imbalances. Precision measures the average value across all classes of the ratio of true class detections relative to all pixels predicted to be members of that class. Recall measures the average value across all classes of the ratio of true class detections relative to the true number of instances of that class. The f1-score is the geometric mean of precision and recall. The Brier score [78] is a method for measuring the accuracy of a probabilistic classifier, defined as the sum of the squared differences between the model's class probabilities output and the true classifications. Finally, the IOU [79] – also known as the Jaccard similarly coefficient, uses set notation to find the average over each class of the intersection of prediction and true class instances divided by the intersection of true and predicted class instances. In the case of all metrics except the Brier score, a higher value is more optimal.

Table 1. Individual models' validation-set metrics.

Decoder	ID	Accuracy	Top-2 Accuracy	Precision	Recall	F1	Brier	IOU
FPN	1	84.57%	95.91%	82.84%	78.90%	80.63%	0.0465	68.80%
FPN	2	84.77%	95.99%	83.21%	79.07%	80.82%	0.0460	69.21%
FPN	3	84.24%	95.80%	82.65%	78.84%	80.49%	0.0477	68.68%
LinkNet	4	85.40%	94.41%	83.55%	80.06%	81.60%	0.0460	70.20%
LinkNet	5	84.85%	94.44%	83.20%	79.49%	81.14%	0.0478	66.49%
LinkNet	6	85.34%	94.60%	83.70%	80.09%	81.68%	0.0463	70.26%
PSPNet	7	87.28%	97.25%	84.43%	81.51%	82.78%	0.0368	72.08%
PSPNet	8	87.13%	97.21%	84.27%	**81.54%**	82.72%	0.0373	72.01%
PSPNet	9	**87.32%**	**97.35%**	**84.46%**	**81.54%**	**82.81%**	**0.0366**	**72.14%**
U-Net	10	85.16%	93.78%	83.58%	79.54%	81.31%	0.0464	69.88%
U-Net	11	84.92%	93.79%	83.38%	79.66%	81.26%	0.0473	69.81%
U-Net	12	84.52%	93.43%	83.22%	78.97%	80.79%	0.0487	69.15%

Table 2. Highlighted models individual test-set metrics.

Decoder	ID	Accuracy	Top-2 Accuracy	Precision	Recall	F1	Brier	IOU
FPN	2	85.30%	96.59%	86.09%	**79.92%**	82.42%	0.0446	71.86%
LinkNet	6	84.77%	95.80%	86.39%	78.72%	81.59%	0.0486	70.82%
PSPNet	9	**86.07%**	**97.31%**	86.17%	79.45%	**82.57%**	**0.0411**	**72.30%**
U-Net	10	85.46%	95.24%	**86.47%**	79.14%	82.00%	0.0459	71.38%

Additional information regarding these models' test-set performance is found in the confusion matrices of Figure 42. Each matrix presents the percent of each class which have been classified as each of the other classes. An ideal confusion matrix would be equal to the identity matrix, with 1.0 along the diagonal and zero elsewhere.

Consider the matrix of model ID #9, and observe that the model accurately classified open water in 95% of instances, and misclassified it as level ice in only 3% of all pixels. Furthermore, observe that the model accurately classifies 74% of ridges, while misidentifying 18% of them as level ice and 7% as broken ice.

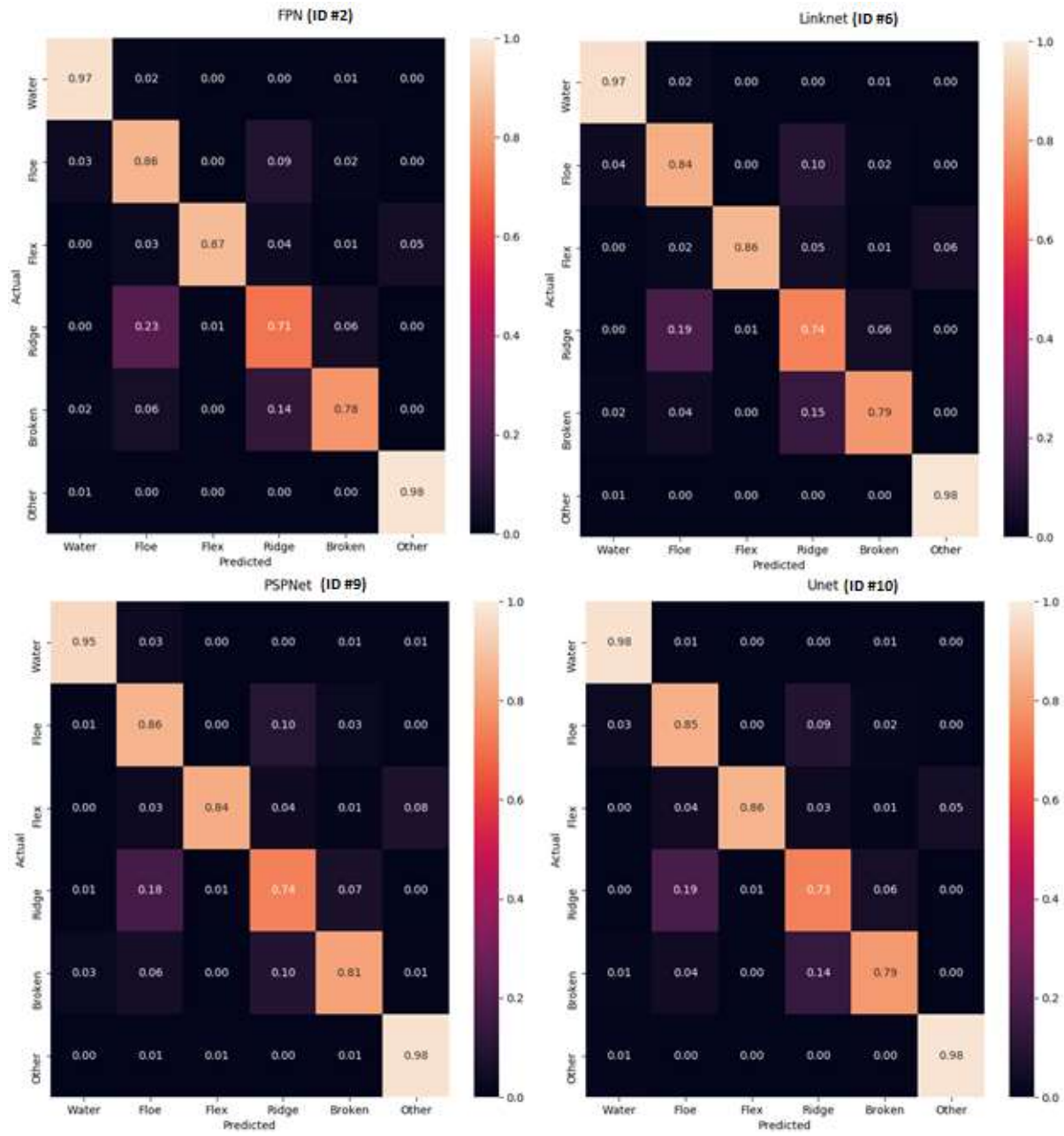


Figure 42. Highlighted models test set confusion matrices.

3.6 System Tuning

Further steps may be taken to improve the system's performance. The specific steps undertaken in this work are model calibration and multi-model ensembling.

3.6.1 Calibration

A perfectly calibrated model will yield predictions for which the confidence of predictions accurately represents the true underlying uncertainty that the particular sample is truly a member of the predicted class [80]. More simply, when a perfectly calibrated model predicts a classification with a certain percentage of confidence, it should be a correct positive classification the same percent of the time on average. There are several advantages to properly calibrated models. It has been shown that the process of model calibration should not negatively affected the classifier's accuracy [80], and furthermore it has been empirically shown that it often improves the classification accuracy [81]. This should only apply to the dataset upon which the classifier was calibrated, and as such an improvement in the validation-set accuracy is expected, while for the test set this is not guaranteed.

In this work, the models which have been trained on the training set and augmented via semi-supervised learning are then calibrated based on the validation set. The calibration process is applied independently to each model of the ensemble. First, the validation set images are fed through the model to produce its segmentation predictions. The predictions are then used to train a logistic regression to best match the model's uncertainty relative to the true validation-set segmentation. The logistic regression applies independently to the vector of probabilities produced with respect to each pixel and allows for the capture of inter-class correlations.

Finally, the outputs of the logistic regression are used to fit an isotonic regression [82]. The isotonic regression applies a non-linear monotonically increasing function to scale the predicted probabilities at each pixel to best reflect the observed uncertainty. Calibration plots for both the original and calibrated models evaluated against the training, validation, and test sets are presented

in Figure 43. Ensemble test-set performance is detailed via confusion plots of the highlighted individual models with calibration presented in Figure 44 in contrast to the uncalibrated equivalent plots in Figure 42.

The individual test-set performance of the highlighted models is presented in Table 3, directly contrasting to the results presented in Table 2. Following calibration, U-Net has displaced PSPNet for the position of top model across several of the metrics. Note that some of the individual metrics decrease while some others increase, except for the Brier score which uniformly improves across all models.

Implementations of both logistic regression and isotonic regression models are provided via Scikit-learn [83]. The above approach describes only one of many valid calibration approaches. The exploration of other calibration approaches should also be considered in future work.

Table 3.Highlighted models with calibration individual test-set metrics.

Decoder	ID	Accuracy	Top-2 Accuracy	Precision	Recall	F1	Brier	IOU
FPN	2	85.07%	95.14%	86.21%	79.28%	81.64%	0.0397	71.05%
LinkNet	6	84.82%	91.59%	86.49%	78.66%	81.47%	0.0425	70.76%
PSPNet	9	85.76%	**96.76%**	86.36%	79.45%	81.75%	**0.0375**	71.52%
U-Net	10	**86.03%**	93.56%	**86.37%**	**80.06%**	**82.40%**	0.0397	**72.11%**

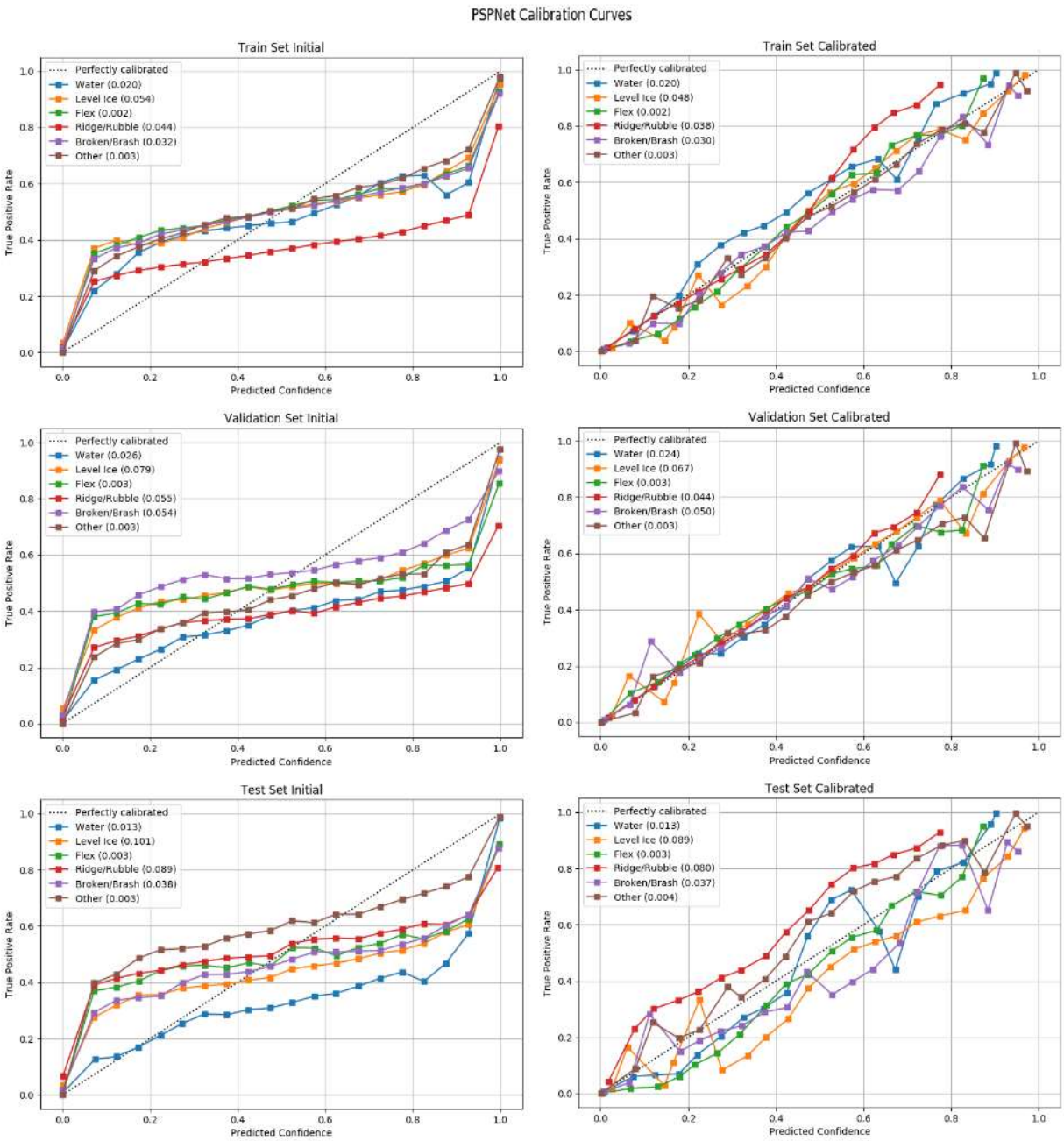


Figure 43. Calibration curves comparison, PSPNet.

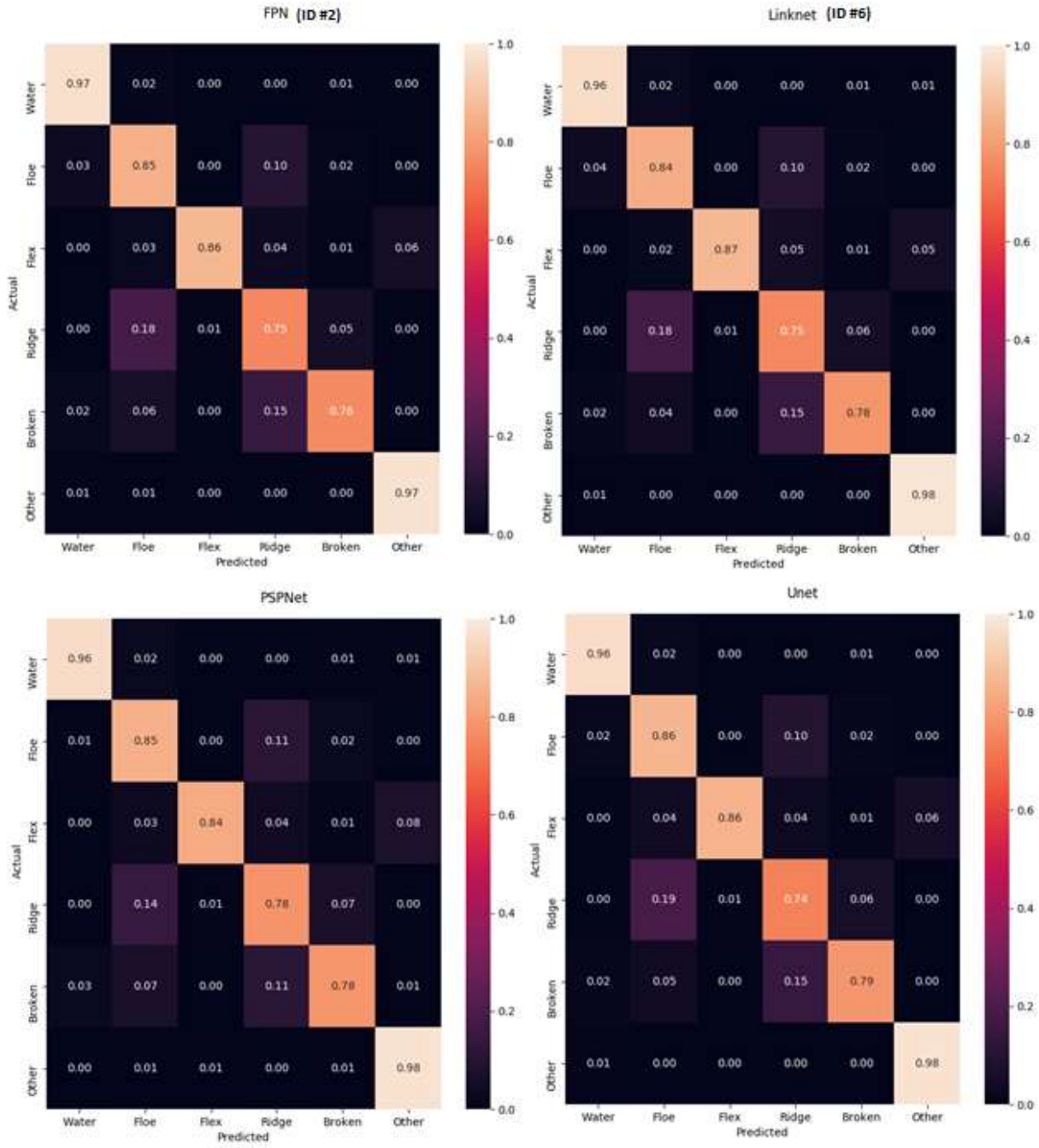


Figure 44. Highlighted models individual test-set performance with calibration.

3.6.2 Neural Network Ensemble

An ensemble is a collection of models trained to perform the same task. An ensemble of models will generally outperform each of the constituent models under the assumption of independent errors, whereby the errors of each model are assumed to be random and independent [84]. However, this assumption often does not hold in practice because models have been trained in the same manner and on the same dataset and as such their predictions may be highly correlated. Nonetheless, neural network ensembles remain an effective method to increase accuracy and decrease the vulnerability to abnormal or erratic predictions that may be exhibited by an individual model.

In order to maximize the performance improvement of the ensemble, the individual models of the ensemble should differ in some way. A total of 12 models are included in the ensemble. Four random initializations were performed, corresponding to the four neural network architectures employed. In each of the four primary training procedures, the final 3 snapshots corresponding in troughs in the cyclic learning rate are sampled and independently fine-tuned. This yields a total of 12 models to comprise the ensemble. Next, each of the 12 models are independently calibrated using the methodology discussed in Section 3.5. Following calibration, the models are complete and receive no further updates.

Following the completed training of all models in the ensemble, the next task was to merge the individual model predictions into a final ensemble-wide prediction. Table 5 summarizes the performance of the ensemble under various prediction amalgamation methods.

Next, each of methods are described. Dimensionality is an important consideration. The input of a single image into one model is represented in a three-dimensional array with shape height, width,

and the final dimension corresponding to the separate RGB components of the image. The neural network in return produces a probabilistic segmentation map, another three-dimensional array with shape height, width, and a final dimension corresponding to the probabilities that each pixel belongs to each of the 6 classes.

When an image is passed through all 12 members of the ensemble, the set of outputs is represented in four-dimensional array with shape of 12, height, width, 6. The set of outputs are represented as $\boldsymbol{X_{ihwc}}$. The processes below operate along the first of these dimensions and reduce along it in order to produce a final output with the same dimensions as if a single image was passed through a single model.

The first method employed is a simple mean of the ensemble predictions, as defined in Equation 7.

$$\text{Mean}(\boldsymbol{X}) = \frac{1}{12}\sum_{\text{i=1}}^{12}\mathbf{X_i}$$

Equation 7. Ensemble Mean

The second method is referred to as Shannon Mix which uses the mean of an output map's Shannon entropy [85] as a way to weight the final prediction more heavily towards the model outputs which were predicted with the highest degree of confidence. The mean Shannon entropy is calculated over one probabilistic segmentation map according to Equation 10. The function $norm$ refers to normalization and serves the simple purpose of dividing each element along the class-probabilities dimensions by the sum along that dimension at each location, ensuring that each pixel maintains a valid probability distribution among the classes.

$$H(X_i) = -\frac{1}{H * W * C}\sum_{h=1}^{H}\sum_{w=1}^{W}\sum_{c=1}^{C} X_{ihwc} * \ln(X_{ihwc})$$

Equation 8. Mean Shannon Entropy

$$SM(\mathbf{X}) = norm(\sum_{i=1}^{12}\frac{X_i}{H(X_i)})$$

Equation 9. Shannon Mix

The next three methods tested are variants of Borda count [86], a ranked ballot system in which choices are ranked by preference and assigned a certain weight in accordance to that rank. At each pixel in an output segmentation map, the probabilities are ranked from highest probability to lowest and then assigned a weight, now interpreted as a probability, according to Table 4.

Table 4. Borda count weightings.

Ranking / Method	1	2	3	4	5	6
Linear	0.285	0.238	0.190	0.143	0.0952	0.0476
Fibonacci	0.406	0.25	0.1562	0.0938	0.0625	0.0312
Exponential	0.634	0.2331	0.0858	0.0315	0.0116	0.00427

Table 5. Comparison of ensemble prediction amalgamation methods

Method	Accuracy	Top-2 Accuracy	Precision	Recall	F1-Score
Mean	**86.77%**	**97.15%**	**85.24%**	80.57%	**82.61%**
Shannon Mix	86.49%	94.01%	84.66%	**80.90%**	82.54%
Linear Borda	86.25%	96.00%	84.75%	80.47%	82.33%
Fibonacci Borda	86.32%	96.28%	84.78%	80.62%	82.44%
Exponential Borda	86.12%	96.44%	84.59%	80.71%	82.48%

Based on the above results, one may conclude that the mean prediction is the optimal solution from those tested. However, there still remains room for further improvement. Instead of a simple mean as tested, a weighted mean allows for further improvement in performance [87]. Weights are determined by each model's independent accuracy on the validation set. The weight assigned to each model is calculated according to Equation 10, where n=12 and $Acc(M_i)$ represents the accuracy of the model indexed.

$$W_i = \frac{Acc(M_i)}{\sum_{k=1}^{n} Acc(M_k)}$$ Equation 10. Accuracy based model weighing

The amalgamation of model predictions is then computed according to Equation 11.

$$Q(X) = \sum_{i=1}^{n} W_i * F(X;\ M_i)$$ *Equation 11. Ensemble weighted mean*

This weighted mean approach is more accurate on the validation set but this may be trivially expected because this is the set from which the weightings were derived. Comparison on the test set however also demonstrates an improvement when the weighting scheme is employed.

3.7 Complexity

This section analyzes the computation complexity and memory requirements of the system as a whole. Benchmarking is conducted on a Tesla-V100-DGXS GPU with thirty-two gigabytes of video memory. The time required to compute one image using the entire ensemble of 12 models averages between 25 and 29 seconds. Ensemble prediction with the calibration system activated increases total time by approximately two seconds.

The time to compute a batch of 10 images averages between 60 and 70 seconds. The total volume of data being processed increased by a factor of ten while the total time to compute only doubles. This demonstrates that the majority of time cost does not come from forward pass of models' predictions but rather from loading and unloading the models into GPU memory and prediction combination. This is further corroborated by the evidence presented in Figure 45, which demonstrates a base time of approximately 25 seconds and an approximate 3 second increase per image processed. Three repetitions were conducted for each batch size. The first repetition

consistently takes longer to compute than the following two due to overhead costs that are not faced in subsequent passes.

In either case, it is evident that the current implementation is not capable of real-time predictions. There are numerous optimizations that may allow for real-time predictions. Multiple GPUs each computing the prediction of one model would decrease the time cost proportional to the inverse of the number of GPUs employed. Removal of the worst performing models would also decrease the time proportionally, but possibly at slight cost of reduced accuracy. Furthermore, a concerted effort to optimize other aspects of the pipeline would likely further expedite predictions.

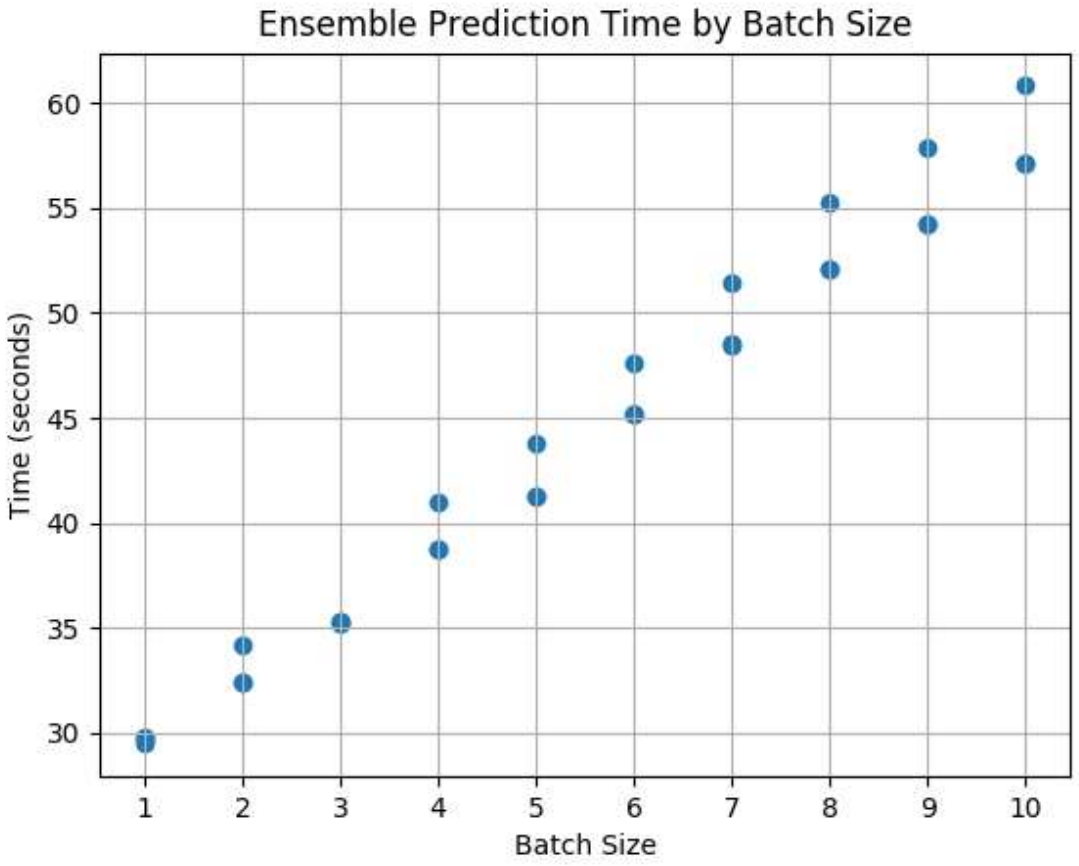


Figure 45. Ensemble prediction time by batch size.

Table 6 presents the parameter count and mean time to make a single prediction for each of the four architectures in the ensemble. Ten repeats are utilized for calculation of mean prediction time. Notably, the model with the highest validation-set accuracy also has the least parameters and the quickest prediction time. This emphasized the efficiency of PSPNet architectures and highlights them as key to further developments.

Table 6. Model type, parameters, and prediction time.

Model Type	Parameter Count	Mean Prediction Time
FPN	45, 638, 095	0.373s
LinkNet	47, 504, 623	0.349s
PSPNet	32, 453, 583	0.333s
U-Net	51, 282, 351	0.352s

3.8 Results & Discussion

The final semantic segmentations produced by the model ensemble present a high overall accuracy and strong recall on each of the individual classes. Performance on the validation set and test set is presented in Figure 46 and Figure 47 respectively. The difference in performance between the two sets is likely attributable to the relatively small size of the datasets. Performance is often

observed to degrade on the test set relative to the validation set. In this case however, one may observe that several of the metrics have improved on the test set relative to the validation set – including both accuracy and f1-score. These differences are most likely explained by the small size of both the validation and test datasets, leading to a high degree of volatility as the contribution of each individual image is comparatively large. Larger datasets would lead to more consistent results.

Hyperparameter optimization was not conducted in this work due to the high computational and time cost of this process. Hyperparameters which may be considered for optimization in future works include the minimum and maximum values of cosine annealing learning rate, the l2 regularization rate, and data augmentation parameters.

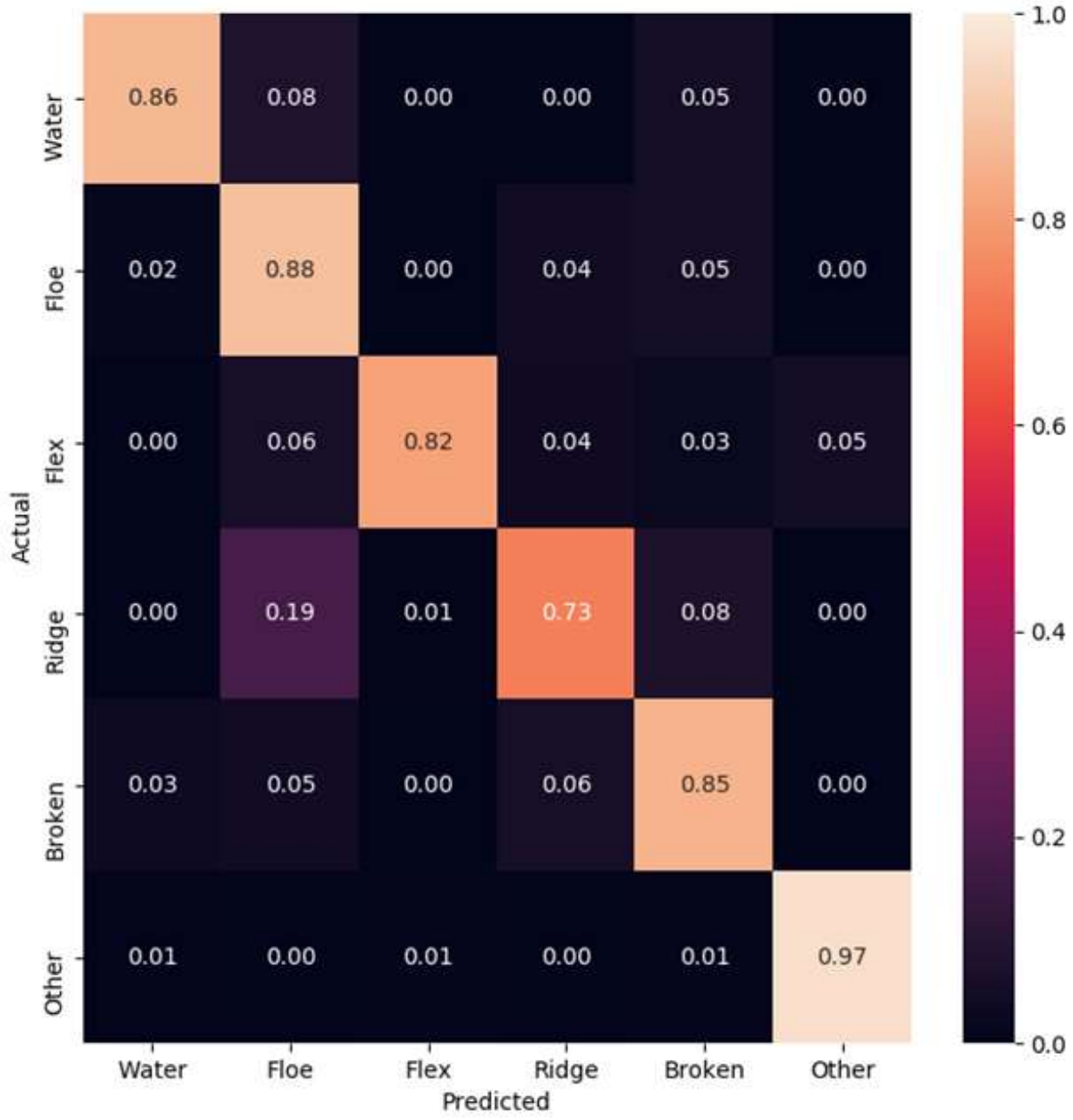


Figure 46. Ensemble validation set confusion matrix.

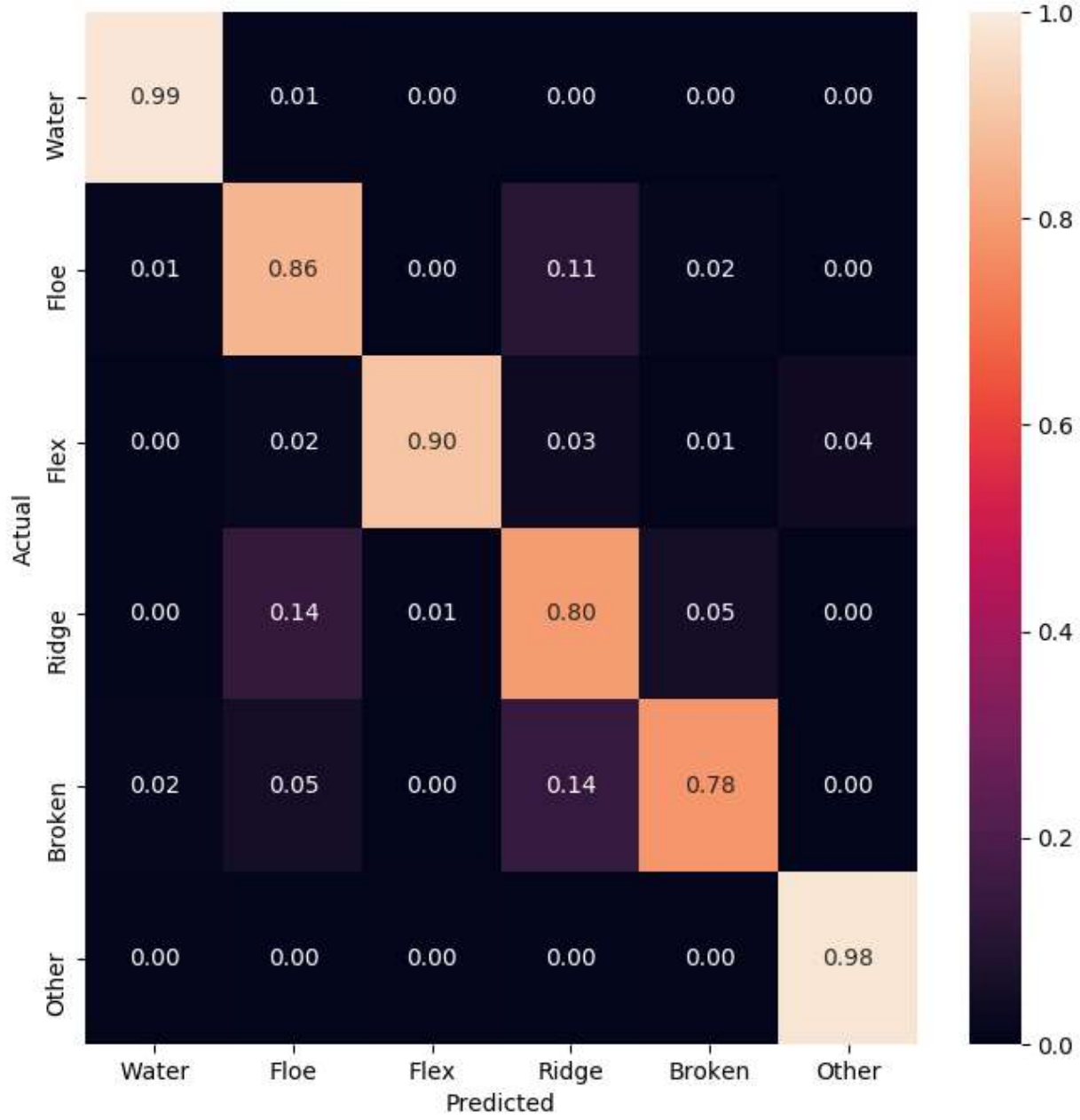


Figure 47. Ensemble test set confusion matrix.

Ensemble performance metrics on the test set are presented in Table 7. From the data found within, we may conclude that the accuracy weighted ensemble mean slightly outperforms the simple mean. Unexpectedly however, the data suggests that the calibration process actually detrimentally affects

the test-set performance. The actual calibration measure is marginally higher as reflected in the Brier score, but all of the other metrics worsened in comparison to the un-calibrated results.

Table 7. Test-set metrics comparison.

Calibration	Weighting	Accuracy	Top-2 Accuracy	Precision	Recall	F1-Score	Brier	IOU
True	None	87.05%	96.94%	88.51%	81.22%	83.57%	0.0355	74.19%
True	Accuracy	87.06%	96.96%	**88.52%**	81.23%	83.58%	**0.0354**	74.20%
False	None	87.14%	97.24%	88.24%	81.70%	84.21%	0.0360	74.71%
False	Accuracy	**87.17%**	**97.26%**	88.24%	**81.72%**	**84.23%**	0.0359	**74.74%**

Test-set performance as a function of time of day is presented in Figure 48. Overall there does not appear to be any significant trends throughout the day, with results remaining mainly consistent throughout. Similarly, test-set performance as a function of month is located in Figure 49. Once again, little overall trend is observed. The spread of metrics is wider in March than it is in February. This may be reflective of more diverse ice conditions observed in March relative to February, or may simply be a result of the small dataset size.

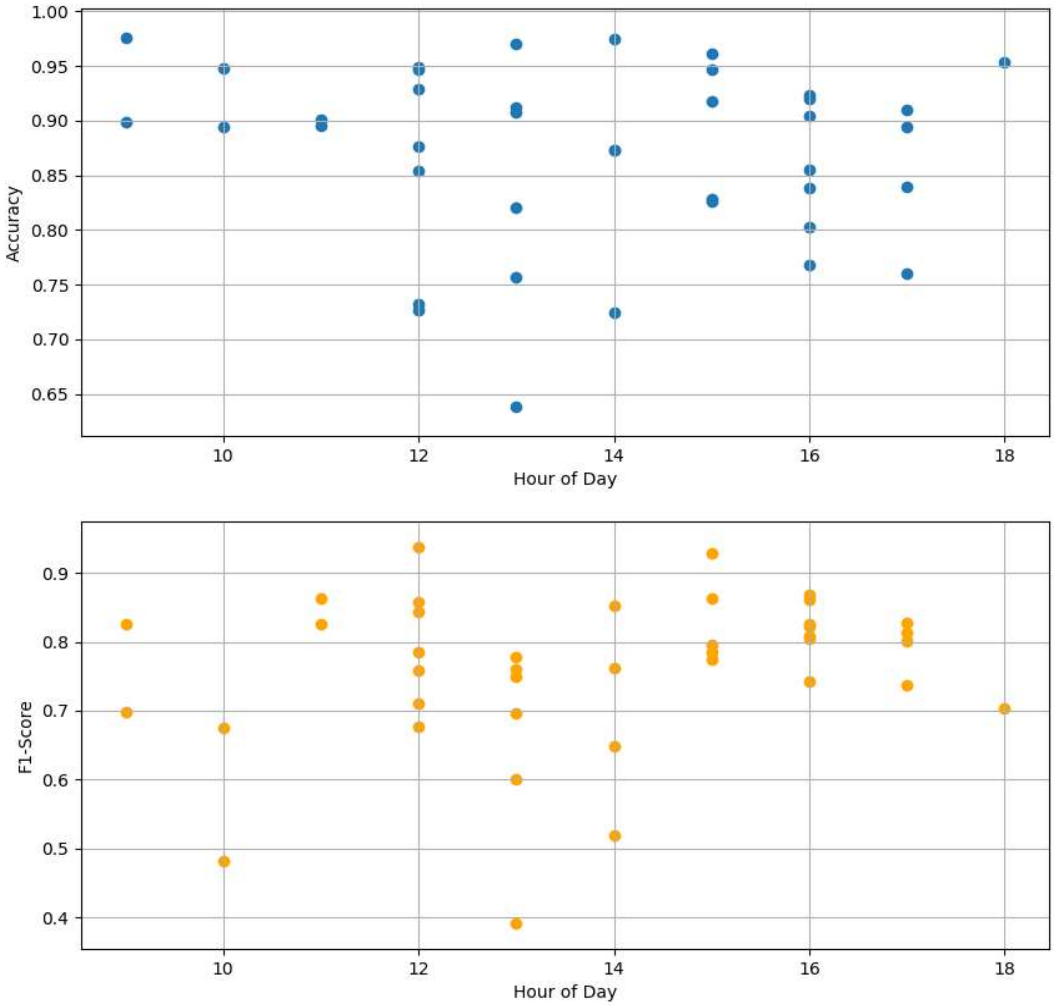


Figure 48. Test-set metrics by time of day.

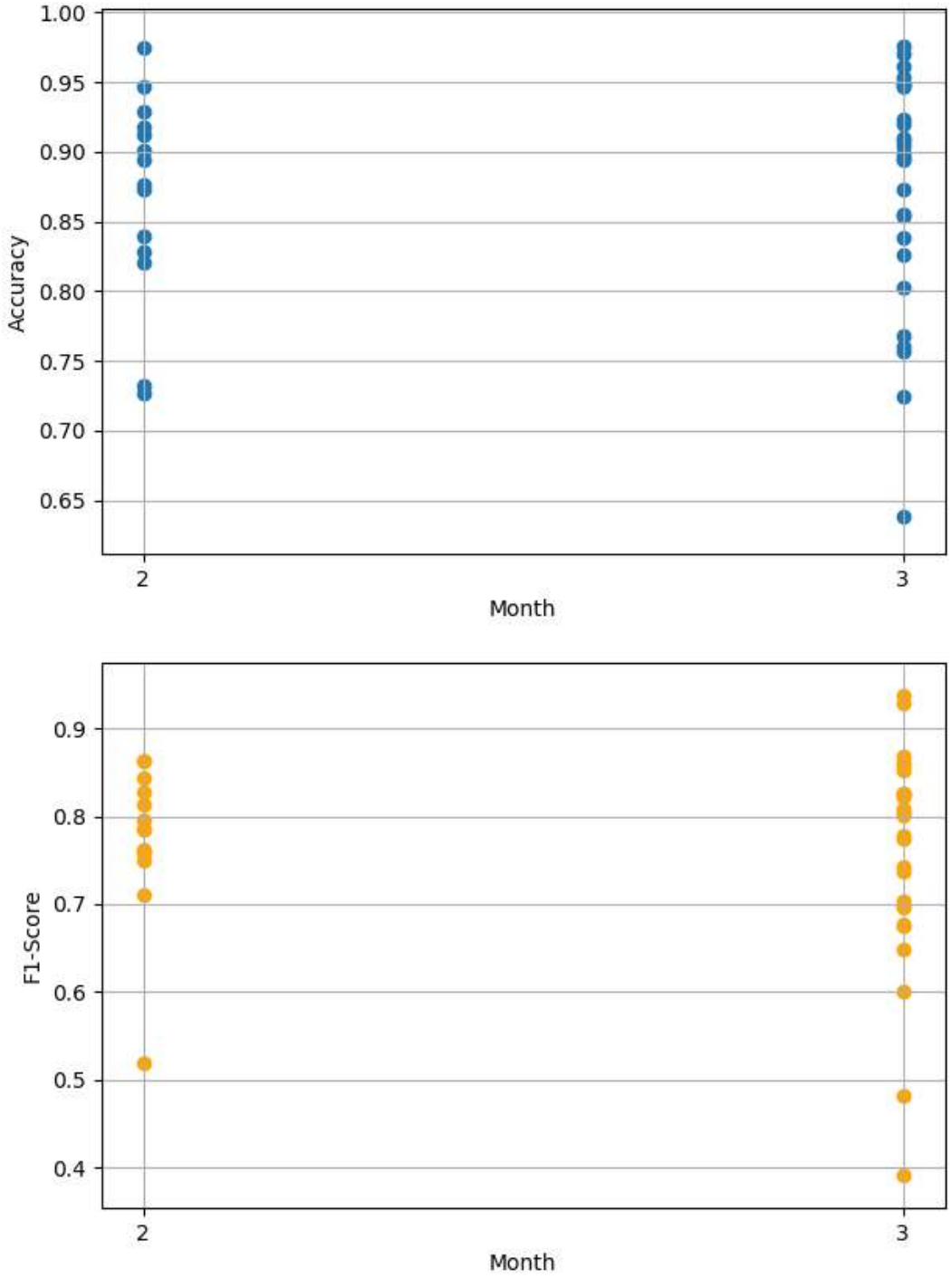


Figure 49. Test-set metrics by Month.

One test image which poses a particular challenge is pictured in Figure 50. In this image, frost has built up on the camera lens, leading to increased difficulty in accurate segmentation of the image.

There are no images in either the training of validation sets which match this level of frosting or low-level visibility. As such, it was considered to remove this image from the test set entirely. However, it was decided that this image does represent real-world conditions and should remain in the test set.

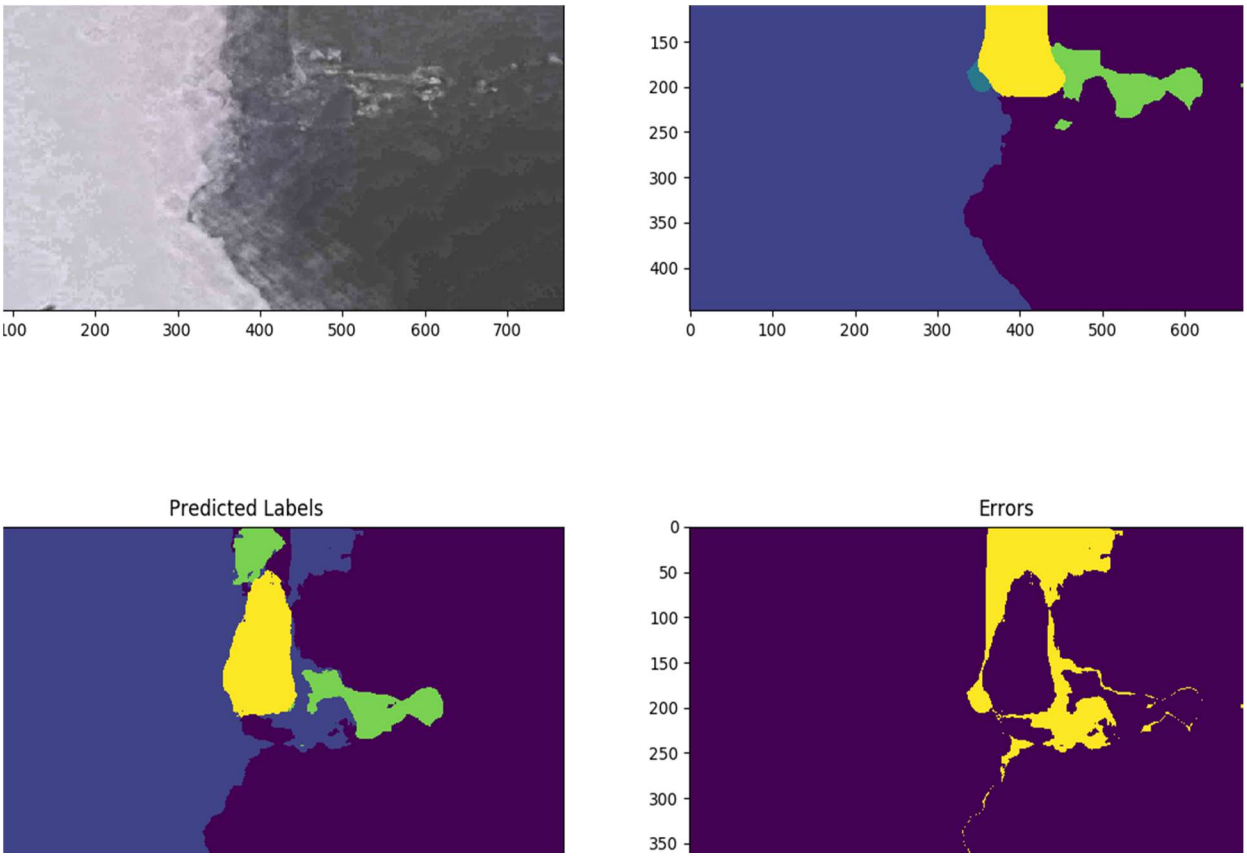


Figure 50. Test-set sample with significant lens frosting.

Shadows have been an ongoing challenge in this project. Despite attempts to mitigate the issue, the effect of shadows may still be clearly observed in many test set images. An example of this issue provided in Figure 51. The outline of the shadow may be clearly observed in the predicted segmentation map.

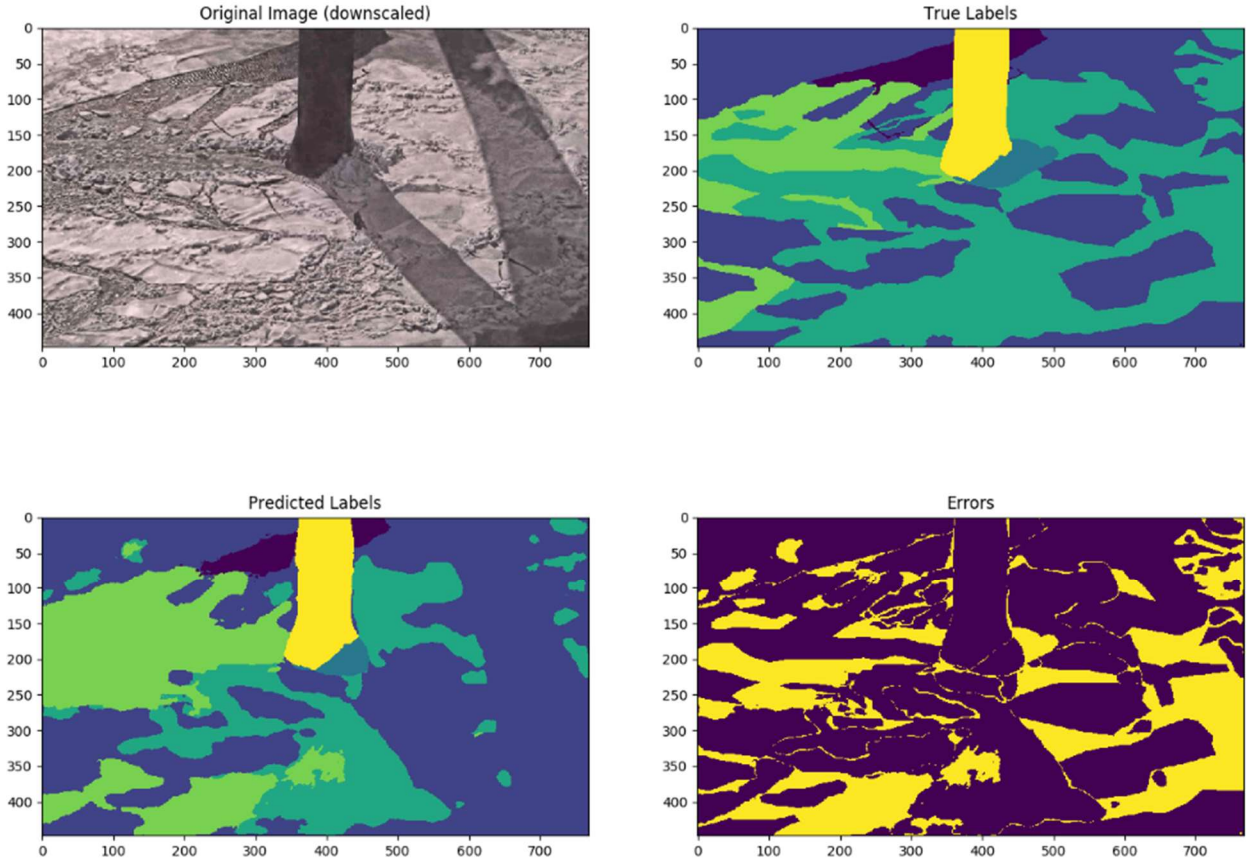


Figure 51. Sample from test set illustrating challenge of shadows.

Another issue which continues to challenge this project is the situation of fully open-water images. As illustrated in Figure 52, images consisting of only open-water tend to have significant portions of the image misclassified as level ice, particularly under wavy conditions when the wave crests catch the natural light and appear brighter than their surroundings.

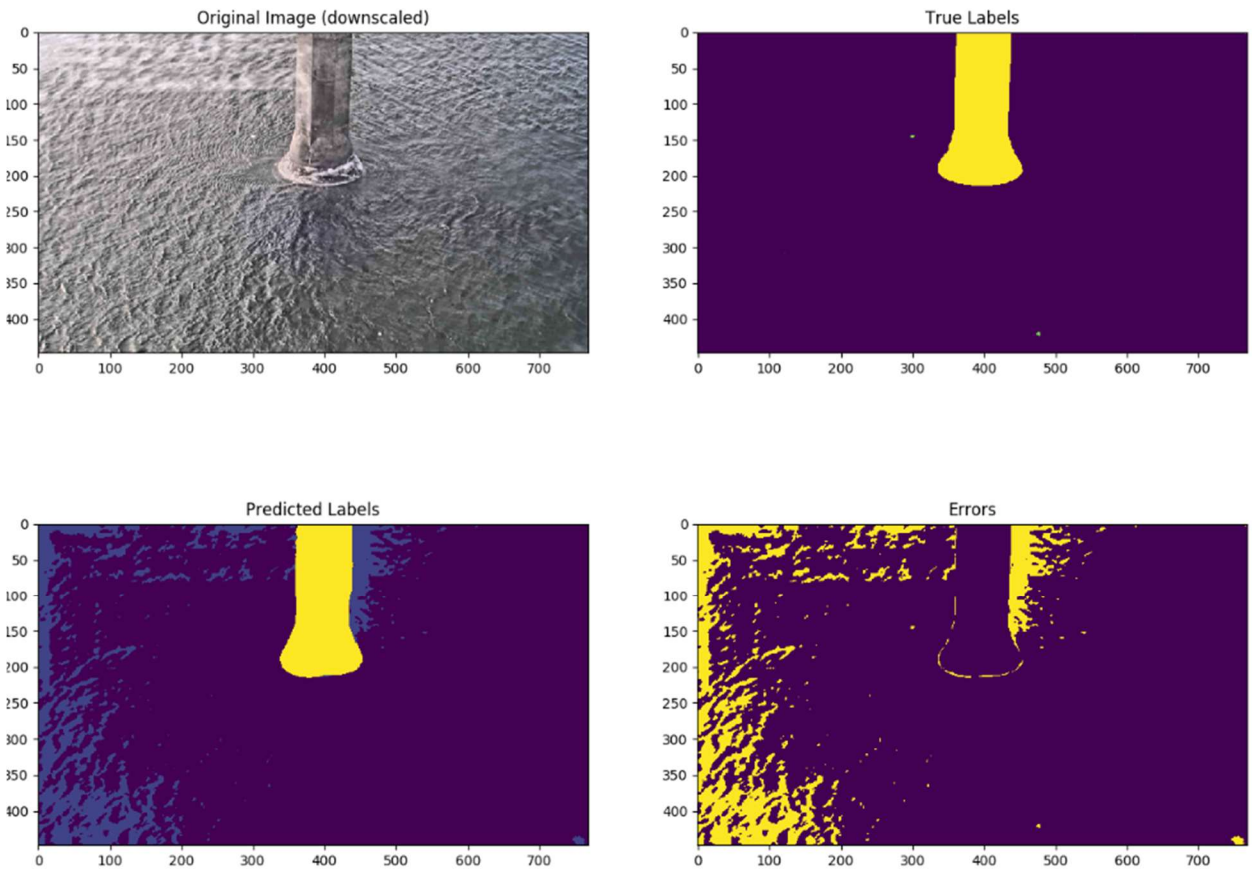


Figure 52. Sample from test set illustrating challenge of open water.

A final issue has been noted when applying the ensemble to sequences of images. Output predictions often appear inconsistent from frame to frame, with the boundaries of regions visibly flickering from frame to frame. Further work is required to improve the consistency of results.

Despite the above noted potential issues, the developed approach gave very good overall accuracy for a broad range of conditions, highlighting the utility of this approach for ice characterization using visual data.

Chapter 4. Visual Velocity Estimation of Sea Ice Floes

This chapter describes the implementation of a system using classical computer vision techniques for the purpose of estimating the velocity of ice floes within a sequence of consecutive images. Outputs from the ensemble generated in Chapter 3 are integrated and the usefulness of this segmentation data is demonstrated.

4.1 Background

Sea ice poses a significant challenge to many offshore structures and operations. Velocity of an ice floe is among the primary dominant factors in determining the loads experienced during ice-structure interactions. As such, the accurate determination of velocity is a key factor for load estimation.

Other works have attempted to produce estimates of floe velocities using various methods. One approach employed was thresholding the image based on intensity, approximate the floes in the image as circles of uniform radius and to observe the rate of change in distance between each floe [88]. Another approach, more similar to the approach employed in this section, matches Harris corner features [89] over the full extent of subsequent image pairs [90].

4.1 Methodology

In this task, each image is first split into 64×64 pixel subregions and then independently processed to increase the distinctiveness of the ice surface. Next, BRISK [22] keypoints are computed over each sub-region and matched with keypoints from the corresponding subregion of the next image

in the sequence. Furthermore, the velocity estimates are averaged with estimates produced based on the semantic segmentation map produced using the techniques described throughout this work, as applied to the original image. The production of the segmentation map does not require specific preprocessing to increase contrast, but is otherwise processed in an identical fashion. One frame from a sample image sequence, along with the corresponding segmentation map, is presented in Figure 53.

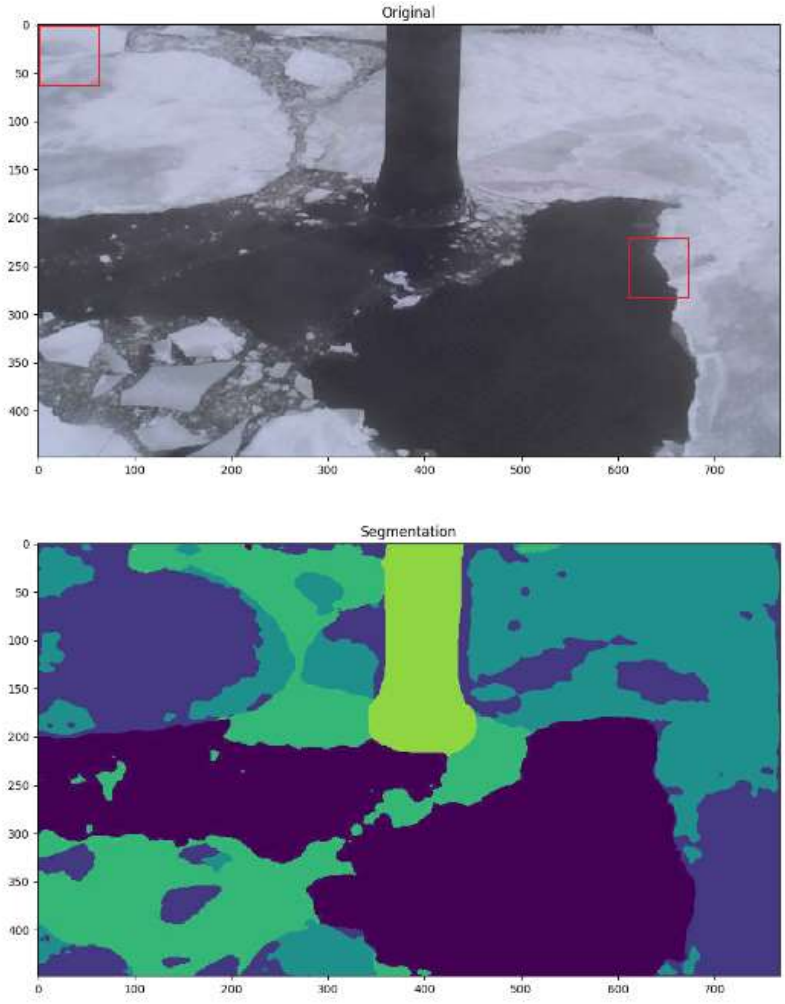


Figure 53. Sample frame and segmentation. Two outlines in red denote the regions presented in Figure 54.

For the purpose of this task, a small dataset has been collected from imagery captured by the two cameras mounted on different piers of the Confederation Bridge. A total of ten image sequences have been gathered from each camera, with each sequence having a length of five images. The cameras produce an image once every three seconds – equivalently, the cameras have a framerate of one-third frames per second.

To start, each image is downscaled from their native resolution of 1920×1080 to 768×448 using bilinear interpolation in order to match the resolution of the semantic segmentation output. In turn, this resolution was selected so as to be a multiple of 32 along each axis while roughly preserving the original aspect ratio, from 1.77 originally to 1.71 after scaling. This also has the added benefit of reducing noise present in the image similar to a light blurring. Next, each image is reduced into a set of 64×64 regions. This tiling prevents inaccurate keypoint matches from introducing potentially significant noise to the velocity estimates. However, this does raise a risk that if velocity was sufficiently high there would be no true matches within the same subregion. In practice, the highest velocity found in this dataset was less than half that which would pose this issue, and the average velocity lower by an order of magnitude.

The 64×64 regions are offset in each direction by 32 pixels, such that each region has a 50% overlap with each of its horizontal and vertical neighbors and a 25% overlap with each of its diagonal neighbors. This process was selected to ensure that ice features near the border of each region were not ignored, as the feature extractors explored were found to mainly detect features near the center of the subregion.

The majority of these subregions consist only of the surface of a particular floe, which have low contrast and make it very difficult to detect meaningful features. As such, image processing

techniques are performed in order to increase the meaningfulness of these surfaces. First, histogram equalization is applied to the subregion, dramatically improving the ability to visually distinguish unique patterns on the ice surface which may be used for feature matching. Finally, a sharpening filter is convolved over the subregion to further highlight features. The filter selected is a modified Laplacian filter [91] which detects transitions in both the horizontal, vertical, and diagonal directions. The modified filter is equivalent to passing a regular Laplacian filter over the image and then adding the output back to the original image. The exact kernel used is presented below in Table 8. Sample results of two image subject to these enhancements are pictured in Figure 54.

Table 8. Modified Laplacian Filter.

-1	-1	-1
-1	9	-1
-1	-1	-1

It is important to note that the operations described in the above section are performed on the intensity values of the image only. In order to achieve this, the original image is transformed from its original RGB representation to an HSV representation. The above operations are then applied to the V channel only, before the image is transformed back into an RGB representation. Many keypoint detection algorithms require grayscale images as input, but the selected detector requires RGB format.

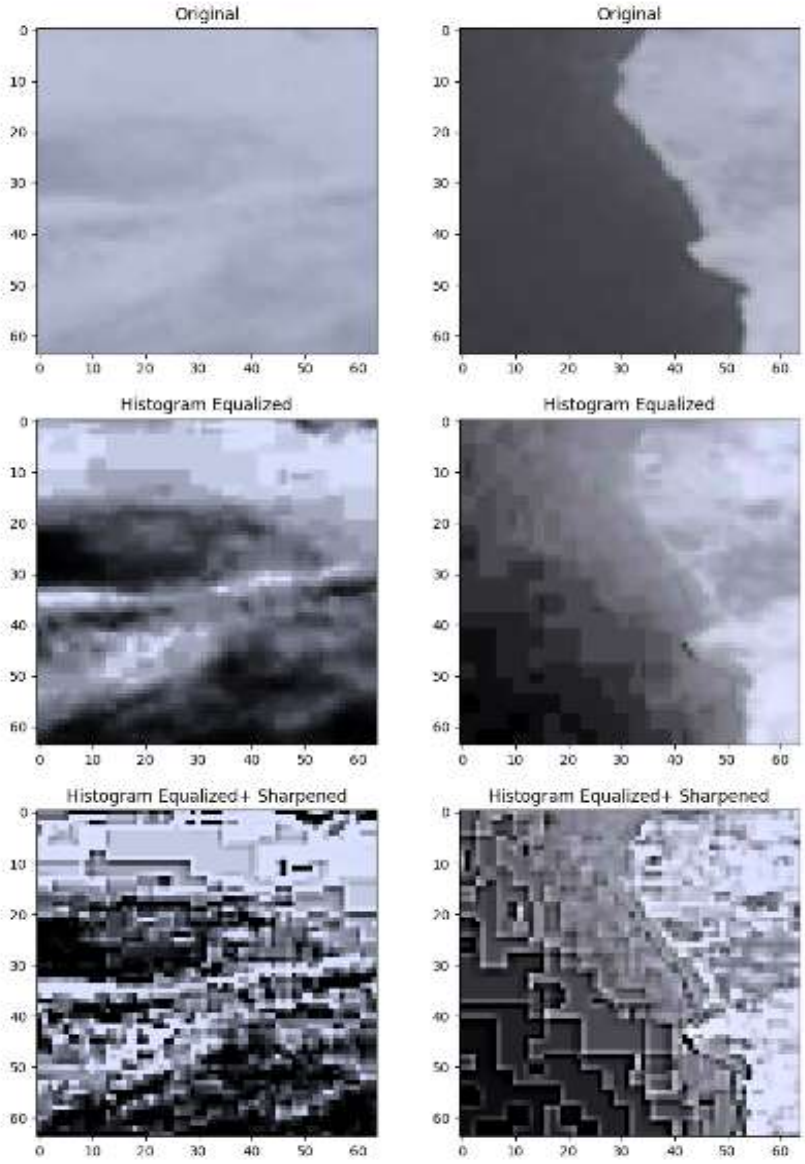


Figure 54. Region Enhancements.

A number of different feature detectors were considered for use in this work. The SIFT and SURF algorithms were excluded from consideration due to their patented status. Only open-source feature detectors with implementations provided though OpenCV [92] were considered. Though empirical examination, BRISK [22] was found to be the most reliable for detecting keypoints in this domain.

Surprisingly, ORB [23] often failed to detect any keypoints within each subregion, even following the image enhancements described above.

Finally, the detected keypoints of corresponding subregions are matched using a brute-force matcher and hamming distance metric. The brute-force match greedily loops through each keypoint in one image and returns the keypoint in the second image with the greatest degree of similarity according to their corresponding descriptors. Example keypoint matches, from the same regions presented in Figure 54, are presented in Figure 55.

For each match, the x and y component of the distance between the locations of the keypoints are recorded. Any match having an x or y component of distance greater than one standard deviation from the corresponding mean is filtered out in order to remove inaccurate matches. The mean x and y distance of the filtered matches are then returned and recorded for the value of this subregion.

The entire process is repeated for each of the 299 subregions per image. This process produces a pair of heatmaps which represent the x and y component of the estimated velocity at each location in a 23×13 grid. Any grid location having a value greater than two standard deviations from the mean are replaced by the mean value. A sample velocity grid is presented in Figure 56. It would also be possible to represent the x and y velocity components in combination as vectors overlaid onto the original image, but this would require that the vectors be projected into the cardinal directions. Uncertainty of the cameras' exact orientations and positions make this a challenge.

A second pair of grids is produced by repeating the above process, minus image enhancements, on the semantic segmentation of the original images. For cells that are fully contained within a single class and thus have no texture to track, the means of the original grids are dropped in-place.

The mean of the grids, excluding predetermined locations containing parts of the pier itself, is taken to produce the final full-image velocity estimate. Camera perspective naturally causes the velocities estimated in the far-field to be less than those in the near-field, but correcting for this distortion was decided to be outside the scope of this project. To convert the distance measured in pixels to real-world units, the known size of the narrowest part of the pier is used to create a baseline estimate of approximately 14 cm/pixel.

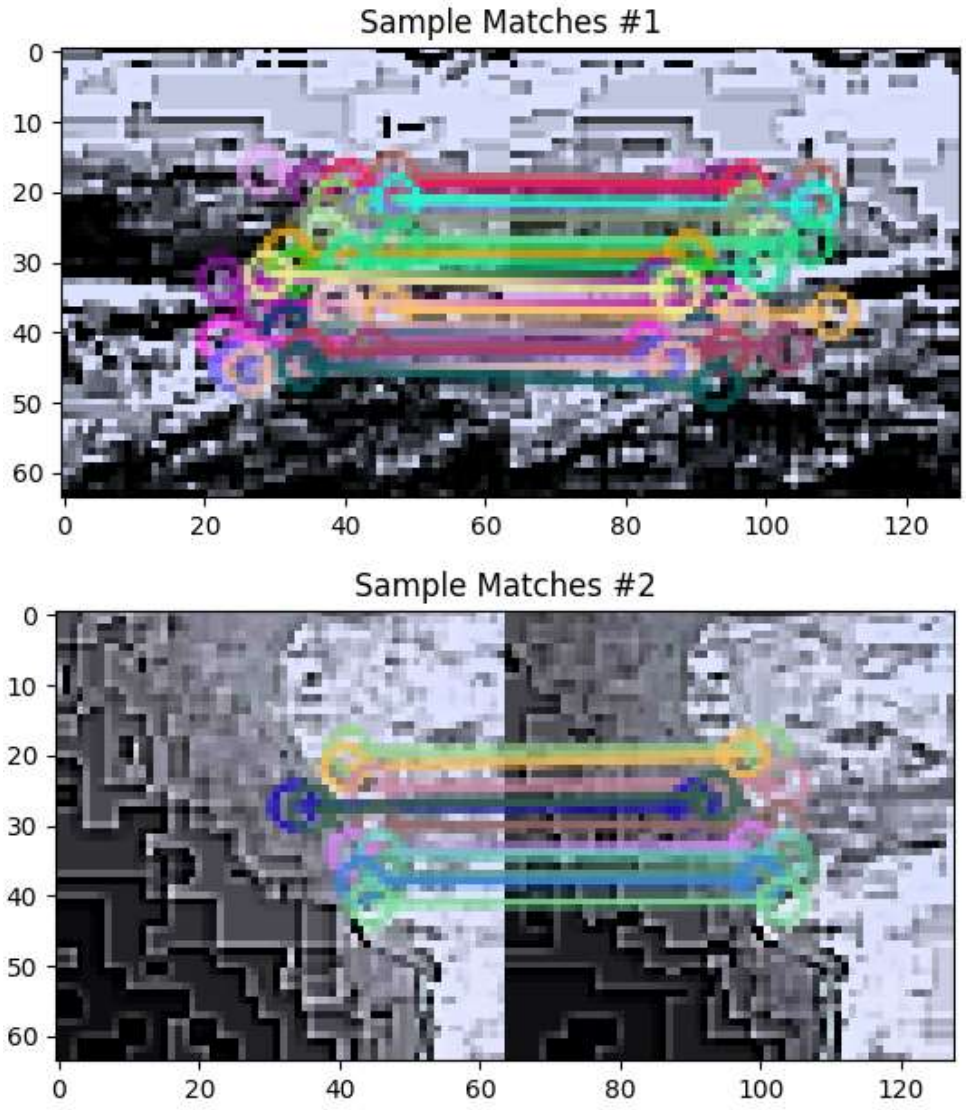


Figure 55. Sample keypoint matches.

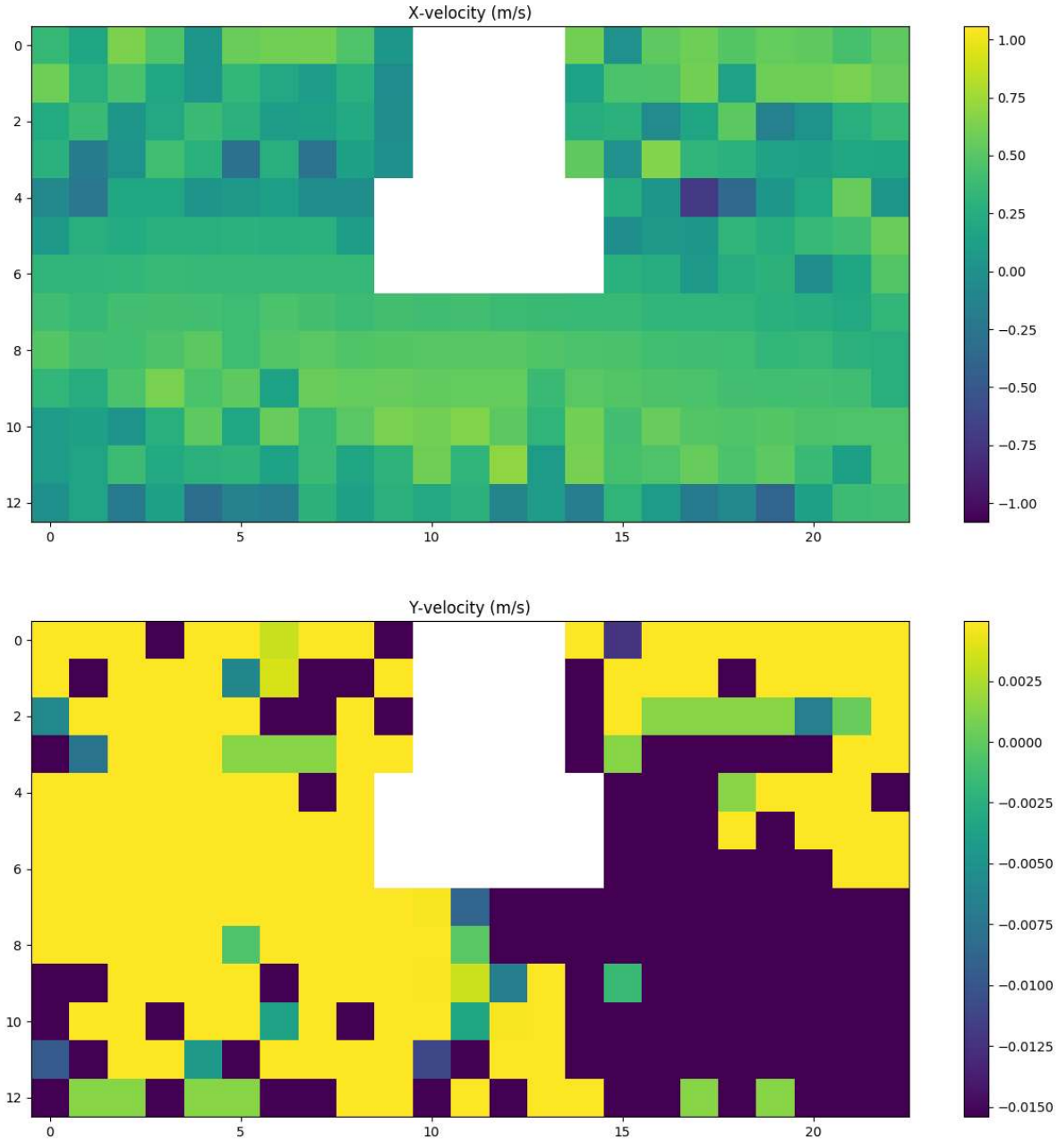


Figure 56. Heatmaps of x and y velocity components detected at each image subregion.

4.2 Results & Discussion

In the absence of directly verifiable base-truth data, a metric has been selected to evaluate the quality of these velocity estimates. The metrics of absolute error (AE) and absolute percent error (APE) are applied where the 'true' label is considered to be the mean velocity found over the duration of each short video clip.

Within the timeframes examined here, it may be assumed that acceleration is negligible and thus that the velocities should be constant. Based on this assumption, the quality of each estimate may be based on the method's ability to produce near-constant predictions over short timeframes. As such, Let *V* define the list of velocity estimates corresponding to a particular sequence, and $E(V)$ to represent the mean value over that sequence. These metric are defined in Equation 12 and Equation 13.

$$AE(V,i) = |V_i - E(V)|$$ Equation 12. Absolute error.

$$APE(V,i) = \frac{|V_i - E(V)|}{E(V)} * 100\%$$ Equation 13. Absolute percent error.

The mean of this metric across all the image-pairs within the dataset was found to be equal to 7.3 percent. The distribution of AE and APE values across the eighty samples in the dataset are presented in Figure 57. The relationship between the mean velocities predicted over the duration of a particular video clip and the AE and APE value of each frame pair is found in Figure 58.

A brief ablation study, in which one component of the system is disabled at a time and the effect observed, was conducted to assess how the various components of the proposed system interact. The following scenarios were examined: 1) visual imagery only; 2) semantic segmentation map

only; 3) median of velocity map rather than mean. In all three cases, the mean absolute percent error increased as components were modified from the original configuration. In case 2), the MAPE reached a level higher than eleven percent. These results suggest that the combination of both the visual imagery and semantic segmentation map is ideal and mutually complimentary towards the minimization of noisy predictions.

Correction for the physical location of pixels is impossible because the camera's intrinsic and extrinsic properties are unknown and because the exact angle at which the camera rests. This poses a major limitation to the current implementation.

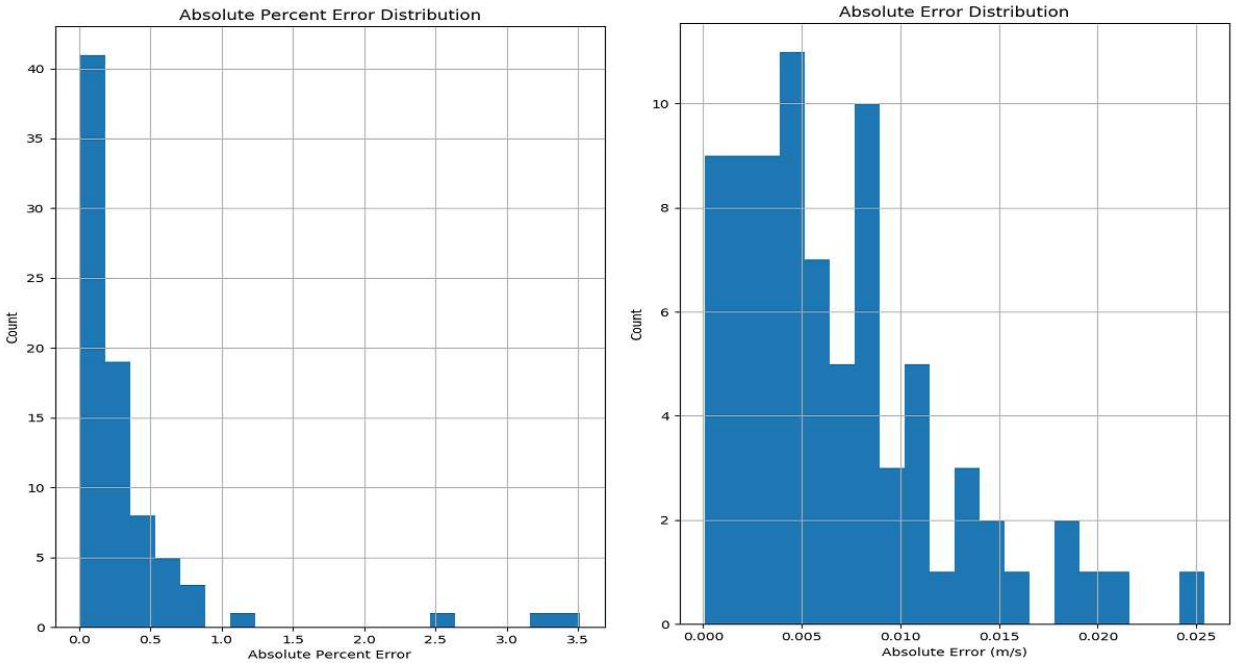


Figure 57. Velocity estimates error distribution.

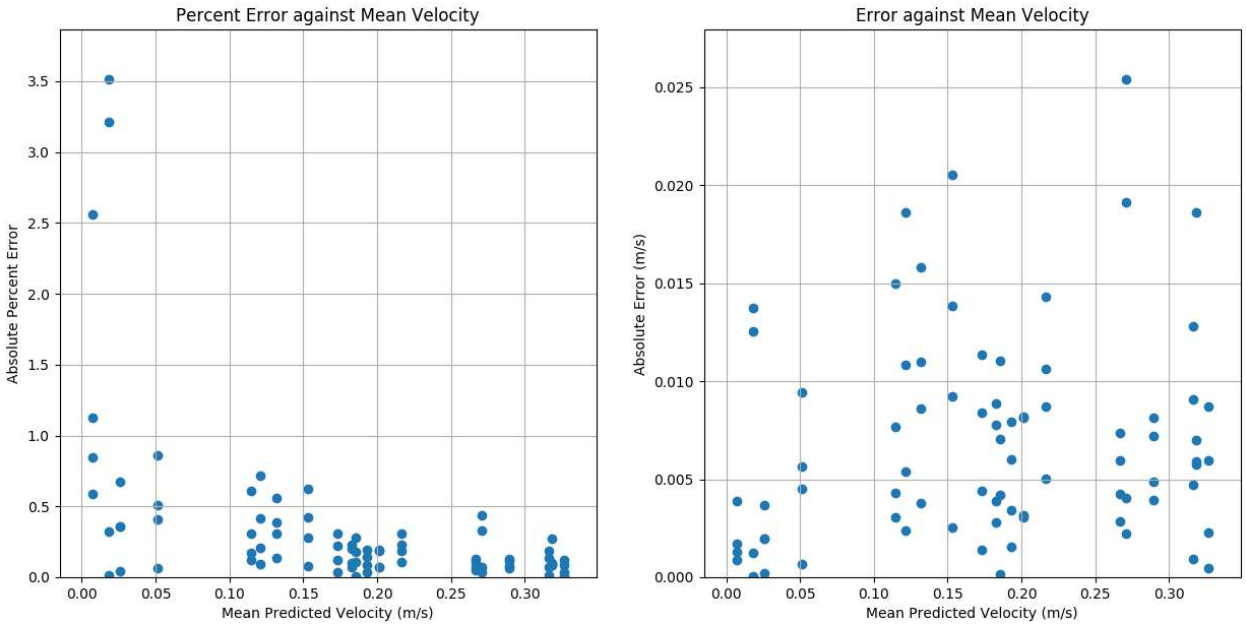


Figure 58. Absolute and percent error against mean predicted velocity of each sequence.

The above figures indicates relatively stable errors with the majority of errors less than 0.01 m/s. This leads to more noisy predictions when the overall velocity is relatively low, as is evident in the chart APE versus mean predicted velocity where the highest percent errors are all associated with the lowest overall predicted velocities.

Chapter 5. Discussion & Conclusion

This chapter provides a final summary of the work conducted along with conclusions and recommendations relevant to future work in this domain.

5.1. Summary

In this work, the development of a system for the automatic semantic segmentation of sea ice conditions was undertaken. The resultant system demonstrates high accuracy and the ability to capture each of the individual classes considered despite the high imbalance of prevalence between classes.

The individual steps involved in the training and calibration of an ensemble of models for this task are detailed in Chapter 3. Demonstration of the applicability of the produced semantic segmentation maps to downstream applications is provided in Chapter 4.

While certain issues persist, overall the project has been successful given the limitations of the small dataset of labelled images utilized. Demonstration of the project's viability has been clearly provided, and much room for further improvements remain.

One goal of this work has been to demonstrate the ability of the system to be extended to ice domains outside of just the Confederation Bridge dataset upon which it has been trained. This goal was put to the test when the models were used to process imagery collected from a Canadian Coast Guard vessel – a domain for which the models had never been trained. While the outputs were not perfect, they were subjectively of an acceptable quality which could be improved considerably with the inclusion of this domain into the training dataset.

5.2. Recommendations

This section outlines recommendations for the development of similar systems and particularly for future iterations of the current work which may seek to expand the domain beyond only imagery collected at the Confederation Bridge.

First and foremost, the collection and annotation of additional imagery should be considered a top priority of future projects. New image and annotation pairs should be added to each of the training, validation, and testing datasets.

Future work may aim to transition from the paradigm of semantic segmentation to the paradigm of panoptic segmentation. This transition would require significant efforts from selection of new model architectures, reformulation of the data processing pipeline, and re-review of the dataset to accommodate this change.

More work is required to address the issues of shadows cast by the structure of the bridge. More aggressive image augmentation may be capable of mitigating this issue – particularly a wider range of illumination level modification. Furthermore, a proposed solution may be the intentional introduction of artificial shadows during training. A random polygon within the image may be selected and all pixels within this polygon could be darkened by a uniform percent. Additionally, a greater ratio of images with open water – particularly under wavy conditions – should be included in the next iterations of the dataset. Only a small proportion of the current datasets currently demonstrate fully open water conditions, leading to degraded performance under these

circumstances. Greater inclusion of images under open water conditions, particularly into the training set, should address these concerns.

Further work is required to address the issue of inconsistent predictions when considering sequences of images. Improved datasets may ameliorate this issue somewhat, but otherwise the system may require redesign so as to directly process image sequences rather than individual frames.

Within the domain of semantic segmentation, the consideration of additional neural network architectures should be executed. Pyramid-scene-parsing-network (PSPNet) was the highest performing model of those evaluated in this work while also demonstrating the least computational complexity. As such, similar models within this domain should be of focus. One particular architecture which has been identified for future work is DeepLabv3+ [53] , which attempts to combine the best aspects of all the architectures strategies employed in this work. The DeepLabv3+ architecture is pictured in Figure 59.

An additional avenue for future development towards the improvement of ensemble performance is the application of ConvCRF [55] postprocessing as was implemented in a related work [52] discussed in Section 2.4. Conditional random field based postprocessing has been shown to improve semantic segmentation performance in the related work.

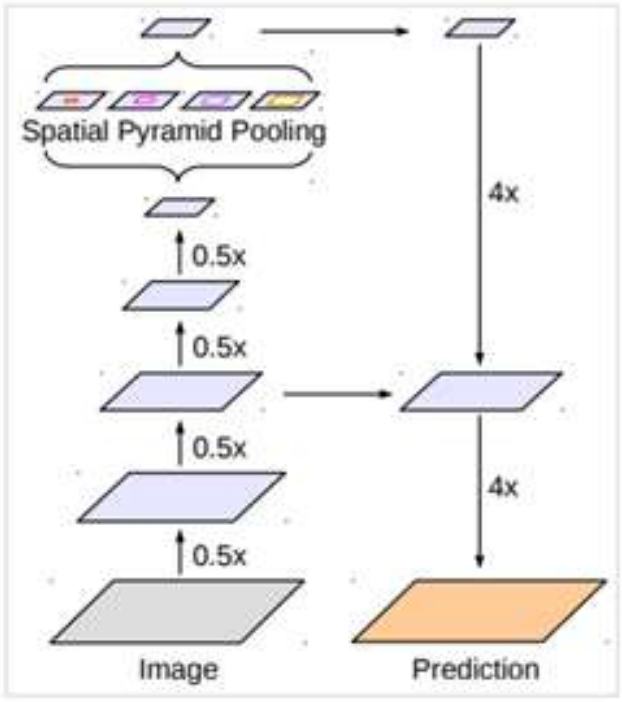


Figure 59. DeepLab neural network architecture. Adapted from figure 2 of [53].

The most important step which can be taken to address the issues that have been noted regarding the current iteration of this system is the addition of new annotated data to the dataset. The current small size of the dataset poses significant challenges to the training of robust models capable of producing high quality predictions across various ice domains.

5.3 Conclusions

Overall, this project has clearly demonstrated the capacity of convolutional neural networks to perform sea ice surface characterization using semantic segmentation. Despite the small size of the dataset employed in this work, the results are highly promising and indicate the potential for convolutional neural networks to be deployed in operational sea ice monitoring programs.

This system could be extended to support a number of applications outside the context of the Confederation Bridge. Such development would necessitate the collection and annotation of additional images sourced from alternative locations such as offshore structures and maritime vessels. Once trained, the resultant model(s) could be deployed to enable operational decision support systems and to assist automated vessels navigating through ice fields.

Fully automated systems for the detection of hazardous ice features can improve existing systems. Historically, human experts have had to manually inspect the imagery from the Confederation Bridge to visually identify moments of ice interaction with the bridge piers. Automated systems allow experts to spend less time on such tedious tasks.

Similarly, crewmembers on vessels passing though potentially dangerous waters must be constantly vigilant for any threatening ice features. Integration of automated systems similar to the one developed in this work would allow the crew to spend less time on lookout – even though it will likely be many years before these systems could be fully trusted and regulated to be responsible for this task without human supervision.

This work makes notable strides towards the goal of autonomous characterization of sea ice conditions and the detection of dangerous ice features. Further efforts can transform the current system into a product capable of many real-world applications.

Bibliography

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. David and J. Dean, "Tensorflow: A System for Large-Scale Machine Learning," in *USENIX Symposium on Operating Systems Design and Implementation*, 2016.

[2] S. Ansari, C. Rennie and L. Poirier, "Real-time classification of sea ice interacting on a bridge pier using artificial intelligence techniques," in *IAHR International Symposium on Ice*, 2020.

[3] K. He, G. Gkioxari, P. Dollar and R. Girshick, "Mask R-CNN," in *IEEE International Conference on Computer Vision*, 2017.

[4] G. Tadros., "The Confederation Bridge: An Overview," *Canadian Journal of Civil Engineering,* vol. 24, pp. 850-866, 1997.

[5] N. Shrestha and T. G. Brown, "20 years of monitoring of ice action on the Confederation Bridge piers.," *Cold Regions Science and Technology,* vol. 151, pp. 208-236, 2018.

[6] M. Cheung, G. Tadros, T. Brown, W. Dilger, A. Ghali and D. Lau, "Field monitoring and research on performance of the Confederation Bridge.," *Canadian Journal of Civil Engineering,* vol. 24, pp. 951-962, 1997.

[7] T. Brown and M. Määttänen, "Comparison of Kemi-I and Confederation Bridge cone ice load measurement results," *Cold Regions Science and Technology,* vol. 55, pp. 3-13, 2009.

[8] T. G. Brown, J. S. Tibbo, D. Tripathi and K. S. N. Obert, "Extreme ice load events on the Confederation Bridge," *Cold Regions Science and Technology,* vol. 60, pp. 1-14, 2010.

[9] K. Croasdale, T. Brown, C. Wong, N. Shrestha, G. Li, W. Spring, M. Fuglem and J. Thijssen, "Improved equations for the actions of thick level ice on sloping platforms," in *Arctic Technology Conference, Offshore Technology Conference*, 2016.

[10] I. Kubat, R. Frederking and T. Hayakawa, "Response of the Confederation Bridge to Ice Action.," in *Annual Conference of the Canadian Society for Civil Engineers*, 2000.

[11] T. Brown, I. Jordaan and K. Croasdale, "A probablistic approach to analysis of ice loads for the Confederation Bridge.," *Canadian Journal of Civil Engineering,* vol. 28, pp. 562-573, 2001.

[12] T. G. Brown, "The Confederation Bridge: analysis of environmental impacts related to ice," *Canadian Journal of Civil Engineering,* vol. 24, pp. 908-914, 1997.

[13] T.-Y. L. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár and C. L. Zitnick, "Microsoft COCO: Common Objects in Context," in *European Conference on Computer Vision (ECCV)*, 2-14.

[14] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth and B. Schiele, "The Cityscapes Dataset for Semantic Urban Scene Understanding," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[15] A. Taghanaki, k. Abhishek, J. Cohen, J. Cohen-Adad and G. Harmarneh, "Deep semantic segmentation of natural and medical images: a review," *Artificial Intelligence Review,* vol. 54, pp. 137-178, 2021.

[16] A. Kirillov, K. He, R. Girschick, C. Rother and P. Dollar, "Panoptic Segmentation," in *IEEE Conference on Computer Vision and Pattern Recogntion (CVPR)*, 2019.

[17] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision,* vol. 60, no. 2, pp. 91-110, 2004.

[18] G. Nebehay and R. Pflugfelder, "Consensus-based matching and tracking of keypoints for object tracking," in *IEEE Winter Conference on Applications of Computer Vision*, 2014.

[19] M. Brown and D. G. Lowe, "Automatic Panoramic Image Stitching using Invariant Features," *International Journal of Computer Vision,* vol. 74, no. 1, pp. 59-73, 2007.

[20] D. G. Lowe, "Object recognition from local scale-invariant features," in *International Conference on Computer Vision*, 1999.

[21] H. Bay, T. Tuytelaars and L. V. Gool, "SURF: Speeded Up Robust Features," in *European Conference on Computer Vision*, 2006.

[22] S. Leutenegger, M. Chli and R. Siegwart, "Brisk: Binary robust invariant scalable keypoints," in *IEEE International Conference on Computer Vision (ICCV)*, 2011.

[23] E. Rublee, V. Rabaud, K. Konolige and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *IEEE International Conference On Computer Vision*, 2011.

[24] K. P. Murphy, Machine Learning: A Probablistic Perspective, The MIT Press, 2012.

[25] R. Sinha, R. Pandey and R. Pattnaik, "Deep Learning For Computer Vision Tasks: A Review," in *International Conference on Intellitgent Computing and Control (I2C2)*, 2017.

[26] A. Vanswani, N. Shazeer, Parmar, N, J. Uszkoreit, L. Jones, A. Gomez, L. Kaiser and I. Polosukhin, "Attention is all you need," in *31st international Conference on Neural Information Processing Systems (NIPS)*, 2017.

[27] S. Shah and H. Dudhrejia, "Speech Recognition using Neural Networks," *International Journal of Engineering Research & Technology (IJERT),* vol. 07, no. 10, 2018.

[28] D. P. Kingma and J. L. Ba, "ADAM: A Method for Stochastic Optimization," in *International Conference for Learning Representations*, 2015.

[29] T. Dozat, "Incorporating Nesterov Momentum into Adam," 2016.

[30] G. Hinton, N. Srivastava and K. Swersky, "Neural Networks for Machine Learning. Lecture 6a: Overview of mini-batch gradient descent.," 2012. [Online]. Available: http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf.

[31] K. Hornik, M. Stinchcombe and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks,* pp. 359-366, 1989.

[32] D.-X. Zhou, "Universality of Deep Convolutional Neural Networks," arXiv 1805.10769, University of Hong Kong, 2018.

[33] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," in *ICLR*, 2015.

[34] K. Jarrett, K. Kavukcuoglu, R. Marc'Aurelio and Y. LeCun, "What is the best multi-stage architecture for object-recognition?," in *IEEE 12th International Conference on Computer Vision*, 2009.

[35] A. Krizhevsky, I. Sutskever and G. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Association for Computing Machinery,* vol. 60, no. 6, p. 84–90, 2017.

[36] DeepAI. [Online]. Available: https://deepai.org/machine-learning-glossary-and-terms/multilayer-perceptron.

[37] S. Saha, "towardsdatascience," [Online]. Available: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

[38] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[39] Y. Bengio, P. Simard and P. Frasconi, "Learning Long-Term Dependencies with Gradient Descent is Difficult," *IEEE Transactions on Neural Networks,* vol. 5, no. 2, 1994.

[40] C. Szegedy, S. Ioffe and V. Vanhoucke, "Inception-v4, inception-resnet and the impact of residual connections," in *International Conference on Learning Representations (ICLR)* , 2016.

[41] C. Szegedy and e. al, "Rethinking the Inception Architecture for Computer Vision," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[42] G. Kyriakides and K. Margaritis, "An Introduction to Neural Architecutre Search for Convolutional Neural Networks," *ArXiv 2005.11.074,* 2020.

[43] S. Hochreiter and J. Schmidhuber, Long Short-term Memory, 1995.

[44] K. Han, Y. Wang, H. Chen, X. Chen, J. Guo, Z. Liu, Y. Tang and A. Xiao, "A Survey on Vision Transformer," *arXiv e-prints arXiv:2012.12556,* 2020.

[45] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov and S. Zagoruyko, "End-to-End Object Detection with Transformers," in *European Conference on Computer Vision*, 2020.

[46] E. Kim, G. Dahiya, s. Loset and R. Skjetne, "Can a computer see what an ice expert sees? Multilabel ice objects with Convolutional," *Results in Engineering,* vol. 4, 2019.

[47] X. &. J. J. Zhang, Z. Lan, C. Li, M. Fan, Y. Wang, X. Yu and Y. Zhang, "ICENET: A Semantic Segmentation Deep Network for River Ice by Fusing Positional and Channel-Wise Attentive Features," *Remote Sensing,* vol. 12, no. 221, 2020.

[48] J. Fu, H. Liu, Y. Bao, Z. Fang and H. Lu, "Dual Attention Network for Scene Segmentation," in *Computer Vision and Pattern Recognition (CVPR)*, 2019.

[49] B. Dowden, O. De Silva, W. Huang and D. Oldford, "Sea Ice Classification via Deep Neural," *IEEE SENSORS JOURNAL,* vol. 21, no. 10, pp. 11879-11888, 2021.

[50] H. Zhao, J. Shi, X. Qi, X. Wang and J. Jia, "Pyramid Scene Parsing Network," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[51] V. Badrinarayanan, A. Kendall and R. Cipolla, "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 39, no. 12, 2017.

[52] N. Panchi, E. Kim and A. Bhattacharyya, "Supplementing remote sensing of ice: Deep learning-based image segmentation system for automatic detection and localization of sea ice formations from close-range optical images," *IEEE SENSORS JOURNAL,* vol. 21, no. 16, pp. 18004-18019,, 2021.

[53] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff and H. Adam, "Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation," in *Eurpoean Conference on Computer Vision*, 2018.

[54] T. Xiao, Y. Liu, B. Zhou, Y. Jiang and J. Sun, "nified Perceptual Parsing for Scene Understanding," *European Conference on Computer Vision (ECCV),* 2018.

[55] M. Teichmann and R. Cipolla, "Convolutional CRFs for Semantic Segmentation," in *arXiv:1805.0477*, 2018.

[56] Y. Guo, "Towards Data Science," 2017. [Online]. Available: https://towardsdatascience.com/the-7-steps-of-machine-learning-2877d7e5548e.

[57] L. Poirier and R. Frederking, "Documentation for the Ice Load Measurements at the Confederation Bridge," NRC - Ocean, Coastal, and River Engineering Research Centre, 2020.

[58] A. Dutta and A. Zisserman, "The VIA annotation software for images, audio and video," in *27th ACM international conference on multimedia*, 2019.

[59] C. R. Harris, K. J. Millman and S. J. Walt, "Array programming with Numpy," *Nature,* vol. 585, no. 7825, pp. 357-362, 2020.

[60] S. M. Pizer, E. P. Amburn, J. D. Austin, R. Cromartie, A. Geselowitz, R. B. T. H. Greer, J. B. Zimmerman and K. Zuiderveld, "Adaptive Histogram Equalization and Its Variations," *Computer Vision, Graphics, and Image Processing,* vol. 39, pp. 355-368, 1987.

[61] G. Yadav, S. Maheshwari and A. Agarwal, "Contrast limited adaptive histogram equalization based enhancement for real time video system," in *International Conference on Advances in Computing, Communcations, and Informatics (ICACCI)*, 2015.

[62] C. Shorten and T. Khoshgoftaar, "A survey on Image Data Augmentation for Deep Learning," *Journal of Big Data,* vol. 6.

[63] L. Perez and J. Wang, "The Effectiveness of Data Augmentation in Image Classification using Deep Learning," arXiv eprint 1712.04621, 2017.

[64] H. Zhang, M. Cisse, Y. Dauphin and D. Lopez-Paz, "mixup: Beyond Empirical Risk Minimization," in *International Conference on Learning Representations (ICLR)*, 2018.

[65] P. Yakubovskiy, "Segmentation Models," GitHub repository, 2019. [Online]. Available: https://github.com/qubvel/segmentation_models.

[66] S. Xie, R. Girshick, P. Dollar, Z. Tu and K. He, "Aggregated Residual Transformations for Deep Neural Networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[67] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *IEEE conference on computer vision and pattern recognition*, 2009.

[68] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan and S. Belongie, "Feature Pyramid Networks for Object Detection," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[69] O. Ronnenberger, P. Fischer and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in *18th International Conference on Medical Image Computing and Computer Assisted Interventions* , 2015.

[70] A. Chaurasia and E. Culurciello, "LinkNet: Exploiting encoder representations for efficient semantic segmentation," in *IEEE Visual Communications and Image Processing (VCIP)*, 2017.

[71] N. Quia, "On the momentum term in gradient descent learning algorithms," *Neural networks: the official journal of International Neural Network Society,* vol. 12, no. 1, pp. 145-151, 1999.

[72] N. S. Keskar and R. Socher, "Improving Generalization Performance by Switching from Adam to SGD," 2017. [Online]. Available: https://arxiv.org/abs/1712.07628.

[73] W. Crum, O. Camara and D. Hill, "Generalized Overlap Measures for Evaluation and Valdiation in Medical Image Analysis," *IEEE Transactions on Medical Imaging,* vol. 25, no. 11, 2006.

[74] C. Sudre, W. Li, T. Vercauteren, S. Ourselin and J. M. Carosos, "Generalised Dice Overlap as a Deep Learning Loss Function for Highly Unbalanced Segmentations," in *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*, 2017.

[75] I. Loshchilov and F. Hutter, "SGDR: Stochastic Gradient Descent with Warm Restarts," in *International Conference of Learning Representations*, 2017.

[76] G. Huang, Y. Li, G. Pleiss, Z. Liu, J. E. Hopcroft and K. Q. Weinberger, "Snapshot Ensembles: Train 1, Get M for Free," in *International Conference on Learning Representations*, 2017.

[77] Y. Babakhin, A. Sanakoyeu and H. Kitamura, "Semi-Supervised Segmentation of Salt Bodies in Seismic Images using an Ensemble of Convoltuional Neural Networks," in *German Conference on Pattern Recognition*, 2019.

[78] K. Rufibach, "Use of Brier score to assess binary predictions," *Journal of Clinical Epidemiology,,* vol. 63, no. 8, pp. 938-939, 2010.

[79] F. Ge, S. Wang and T. Liu, "New benchmark for image segmentation evaluation," *Journal of Electronic Imaging,* vol. 16, 2007.

[80] I. Cohen and M. Goldszmidt, "Properties and Benefits of Calibrated Classifiers," in *8th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD}*, 2004.

[81] A. Niculescu-Mizil and R. Caruana, "Predicting Good Probabilities with Supervised Learning," in *Proceedings of the 22nd International Conference on Machine Learning*, 2005.

[82] R. Barlow and H. Brunk, "The Isotonic Regression Problem and its Dual," *Journal of American Stastical Association,* vol. 67, no. 337, pp. 140-147, 1972.

[83] Pedregosa, Fabian, G. Varoquaux, A. Gramfort, V. T. B. Michel and O. Grisel, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research,* vol. 12, 2011.

[84] L. K. Hansen and P. Salamon, "Neural Network Ensembles," *IEEE Transcations on Pattern Analysis and Machine Intelligence,* vol. 12, no. 10, 1990.

[85] P. Bromiley, N. Thacker and E. Bouhova-Thacker, "Shannon Entropy, Raneyi Entropy, and Information," 2004.

[86] P. Emerson, "The original Borda count and partial voting," *Social Choice and Welfare,* vol. 40, pp. 353-358, 2013.

[87] X. Frazao and L. A. Alexandre, "Weighted Convolutional Neural Network Ensemble," in *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, 2014.

[88] M. Smith and J. Thomson, "Pancake sea ice kinematics and dynamics using shipboard stero vision," *Annals of Glaciology,* vol. 61, no. 82, pp. 1-11, 2020.

[89] C. Harris and M. Stephens, "A combined corner and edge dector," in *Alvey Vision Conferece*, 1988.

[90] S. Ji, A. Wang and Q. Yue, "Digital image techniques of sea ice field observations in the Bohai Sea," in *21st International Conference on Port and Ocean Engineering under Arctic Conditions*, 2011.

[91] X. Wang, "Laplacian Operator-Based Edge Detection," *IEEE Transcations on Pattern Analysis and Machine Intelligence,* vol. 29, no. 5, pp. 886-890, 2007.

[92] G. Bradski, "The OpenCV Library," *Journal of Software Tools,* 2000.

[93] L. Strub-Klein and D. Sudom, "A comprehensive analysis of the morphology of first-year sea ice ridges," *Cold Regions Science and Technology,* vol. Volume 82, pp. 94-109, 2012.

Appendix A – Excerpt from the Confederation Bridge Manual

Content of this section is directly quoted from [57]. Permission has been granted to include this previously unpublished content so as to add context to the work described in the main body of this text.

VIA (VGG Image Annotator) [58]*, shown in Figure A1 is an application that the University of Ottawa and the NRC have used to identify ice features observed in images from the camera systems monitoring the NRC instrumented piers at the Confederation Bridge. Ice features are currently labeled in four (4) categories (the labels are bold and underlined); Ice Floe, Broken Ice, Ridged Ice, Flexural Failure. During the first season two other labels; Open Water and Brash Ice were also used however, it was decided that identifying brash ice was not pertinent to our objectives and that ice was later categorized with broken ice and unlabeled parts of the image would be treated as open water. In the future we aim to add two more labels; Splitting Failure and Crushing Failure. The model, or a similar, one could also be used to determine level ice thickness and ridge height.*

Figure A1. VIA application.

Identification of various ice types

The ice types identified by the AI algorithm will be Ice Floe, Broken Ice, Ridged Ice, Flexural Failure and Crushing Failure. Figure A2 taken on February 27 at 16:55:13 has most of the ice types that are identified by the AI algorithm.

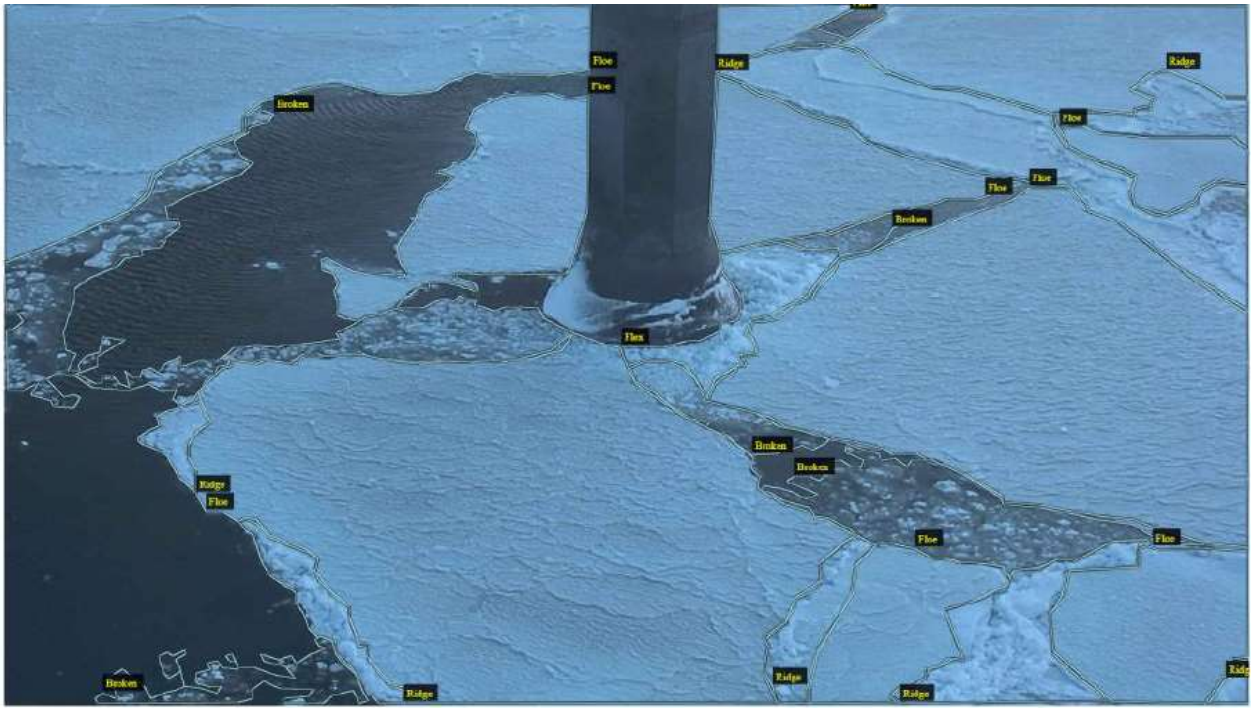


Figure A2. Image comprising most of the ice types.

For any given image the majority of the ice is likely to be categorized as an Ice Floe. The portions of Figure A2 identified as ice floes are indicated in Figure A3. These sections are generally flat ice however the flows can include deformed (ridged and rafted) ice as well. A piece of ice should be at least 5 m in diameter or about half the width of the pier to be identified as a floe.

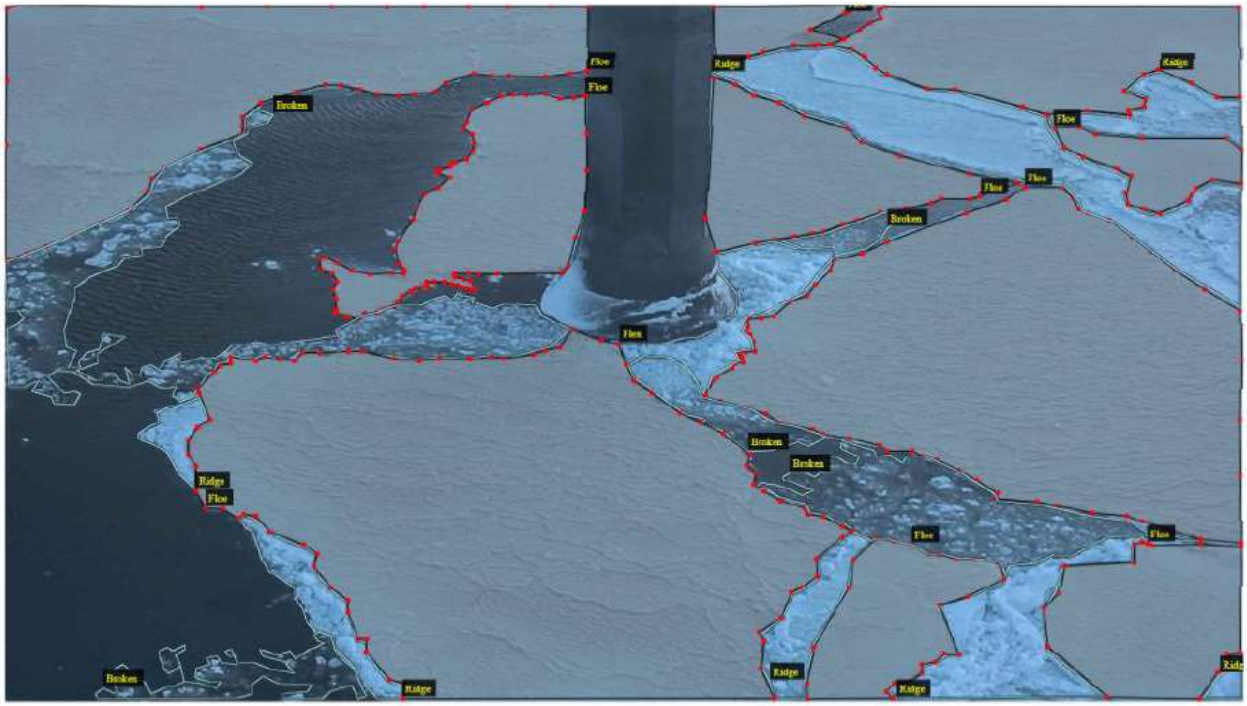


Figure A3. Example of ice floes (gray areas outlined in red) from Figure A2.

Broken ice is the least imposing ice type that will interact with a structure or vessel. You will generally find a wake of broken ice opposite where ice is failing against the pier, such as to the left of the pier in Figure A4. Because the ice is not consolidated it is unlikely to pose any significant load while clearing the bridge piers.

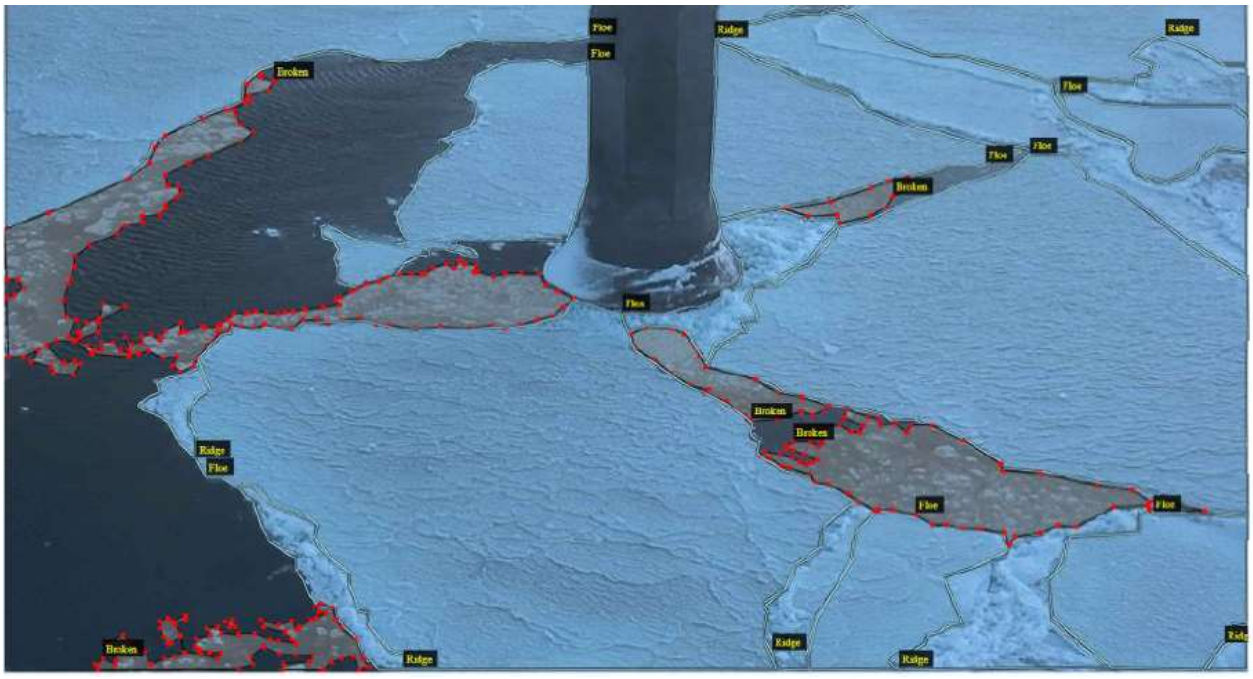


Figure A4. Example of broken ice (gray areas outlined in red) from Figure A2.

Ridged ice is the label used to identify deformed ice (ridged or rafted). The ridged sections of Figure A2 are highlighted in Figure A5. A rafted section of ice is where two level ice sheets collide and one is pushed over and the other under creating an ice thickness twice the level ice thickness. An example of rafted ice is the section extending from the right side of the pier near the top of Figure A5. The remaining examples of deformed ice in Figure A5 are ice ridges. Ridged ice is formed when an ice sheet is pushed against an object and creates debris that accumulates above and below the existing ice sheet. It regularly occurs as a result of an ice sheet colliding with another ice sheet or another object such as a bridge pier or a vessel. If a ridge is newly formed the ice will not be consolidated and despite being a greater mass of ice it may not be imposing. Once the ridge has consolidated it forms the most imposing ice observed in subarctic areas where icebergs are not found. A level ice sheet has a freeboard (above the waterline) which accounts for

10% of the ice thickness. Typical values for ice ridges have the keel (ridged ice below the waterline) as 4.2 times the thickness of the sail (ridged ice above the waterline) [93]. *The keel of a first year ridge is never entirely consolidated.*

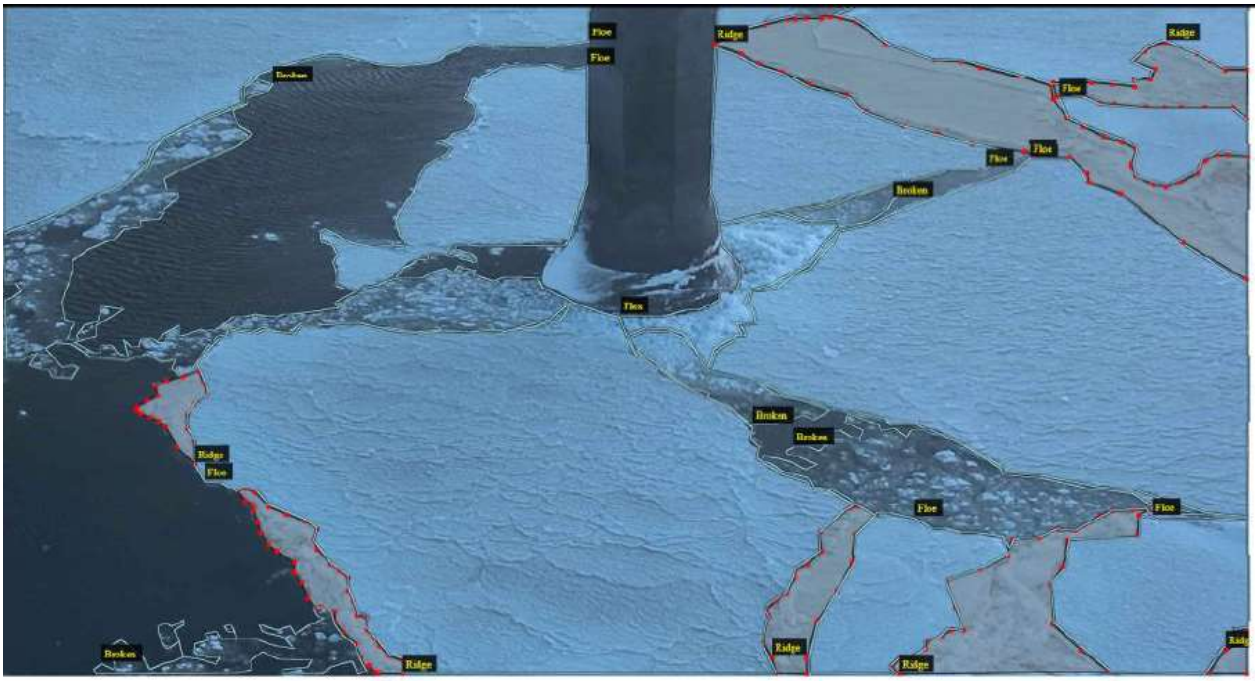


Figure A5. Example of ridged ice (gray areas outlined in red) from Figure A2.

Ice ridges located entirely in the upper approximately 10% of the image will not be identified. These ridges cannot interact with the pier and images in the far field can be difficult to resolve. The ridges from Figure A2 are relatively simple to identify. They are fresh ridges and the ice is exposed. As a ridge becomes more weathered and especially as it becomes snow covered it can become very difficult to identify. In some large loading events the only way to determine that there was interaction with a ridge is by examining the ice failing against the bridge pier. A sudden increase in ice volume failing against the bridge is an indication of a change in ice thickness, or impact with an ice ridge

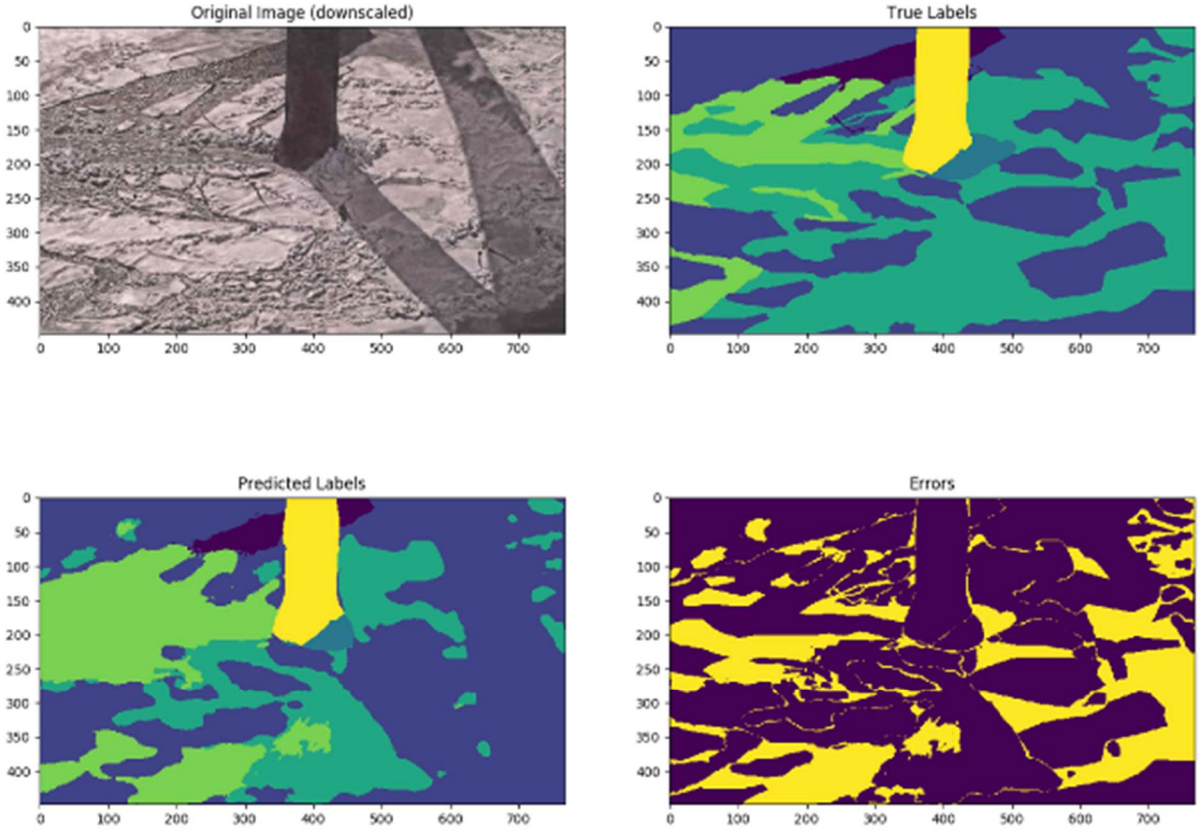
The ice failure against the bridge piers will occur around approximately half the bridge pier at the waterline as the ice fails against the structure. The ice failure can be in splitting, flexure or crushing. Splitting failures are generally short lived although occasionally they will happen regularly in succession. Failure in flexure is much more common than crushing. The conical shape of the piers at the waterline is designed to bend up an ice sheet, to encourage failure in flexure and reduce loads on the structure. It can be challenging to differentiate between flexural and crushing failure by only looking at a single image. When you examine a video during a flexural failure you will see blocks of ice break off, rise, and fall on the rubble pile. During a crushing failure instead of seeing these blocks flowing out of the top of the rubble pile you will see snow. It will appear much more fluid.

Figure A6. Example of flexural ice failure on the pier cone (gray areas outlined in red) from Figure A2.

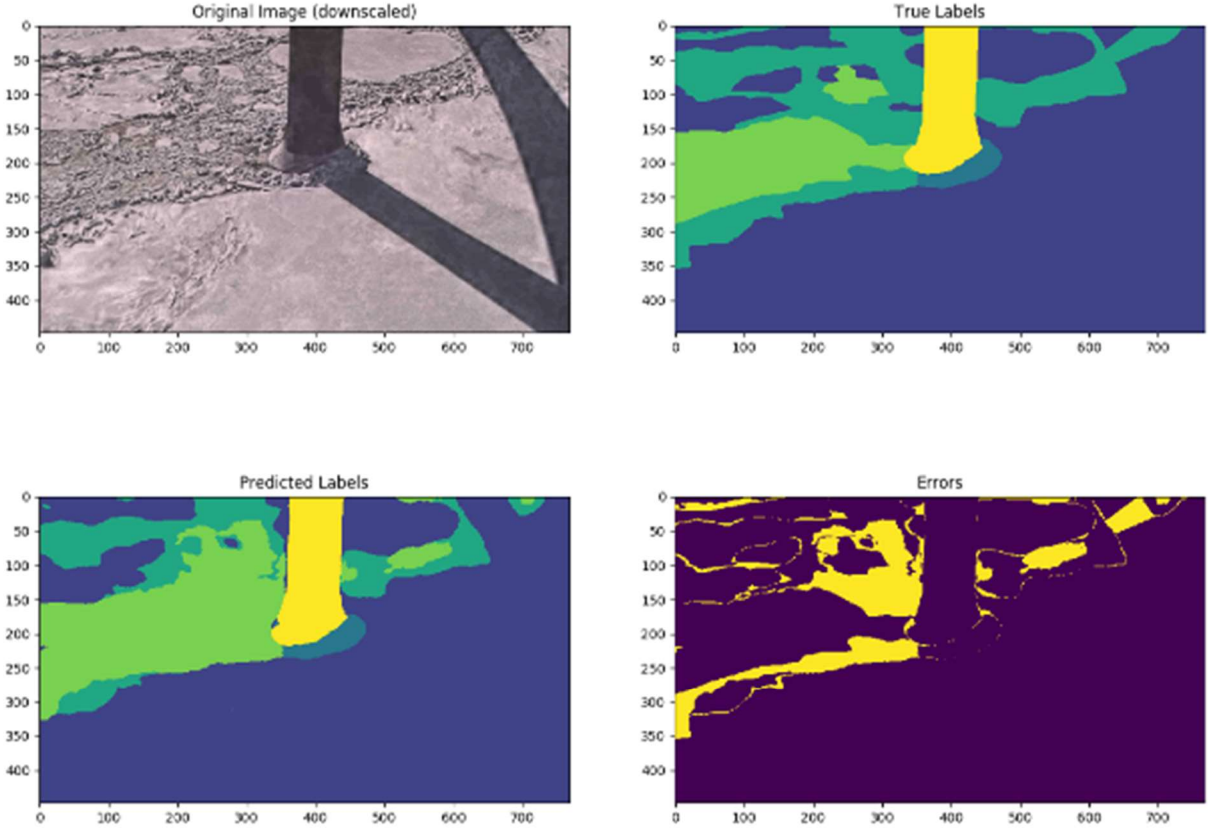
Appendix B – Test Set Results

This appendix contains all imagery from the test set, evaluated using the calibrated and accuracy-weighted ensemble. Make note of the naming scheme used to uniquely identify each image which is as follows <pier number>-(year, month, day, hour, minute, second). The labeling colour scheme is presented below for reader convenience. Errors are found in the lower-right corner of each figure and are marked in yellow to indicate each pixel for which the predicted label did not match the true label.

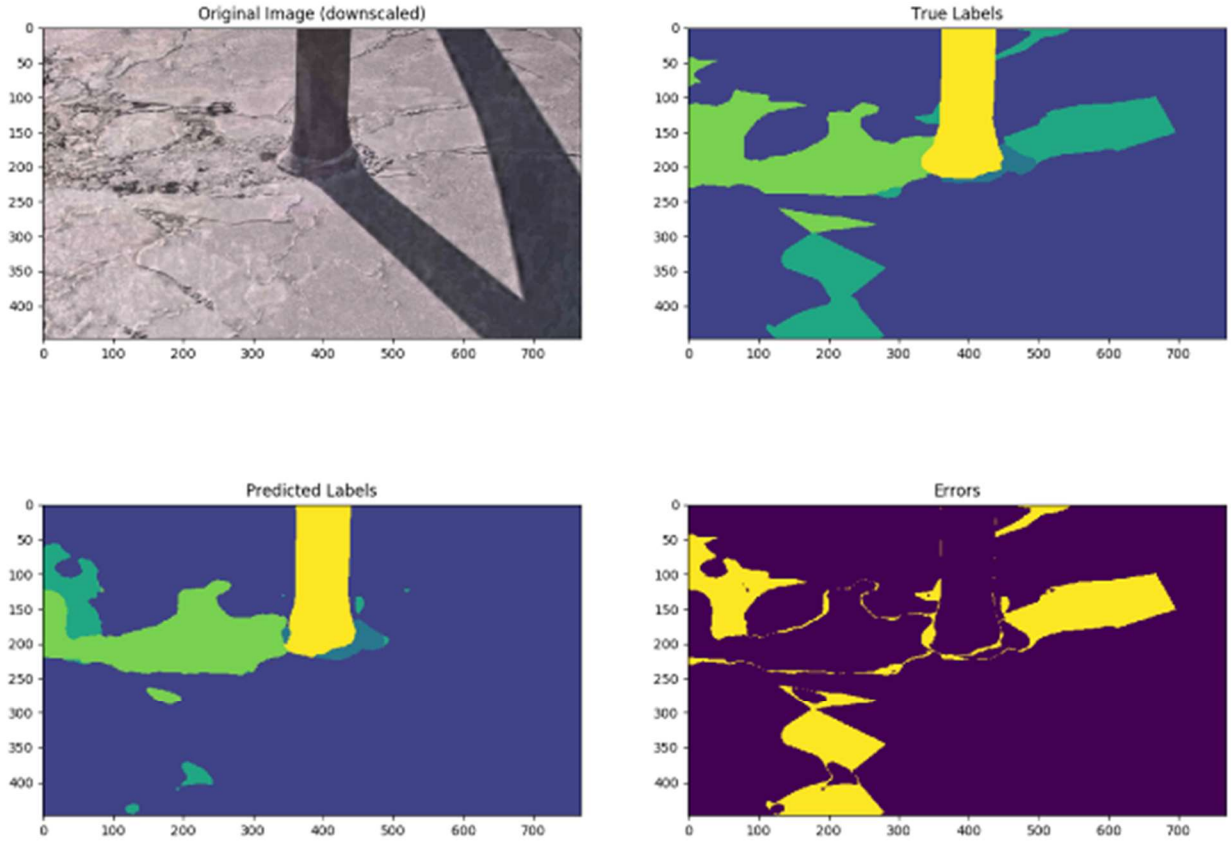
Ensemble Calibrated & Accuracy Weighted - 119-(2019, 2, 25, 12, 25, 39)



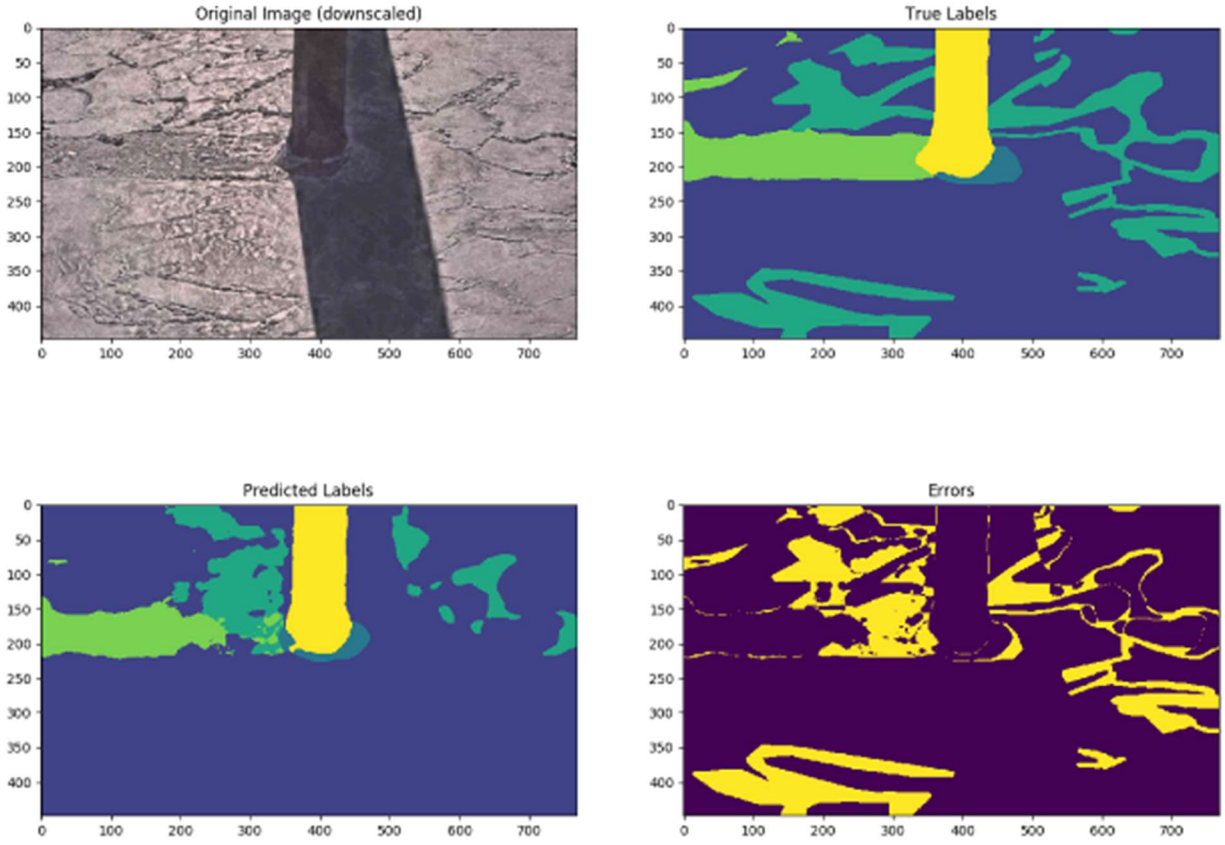
Ensemble Calibrated & Accuracy Weighted - 119-(2019, 2, 26, 11, 47, 31)



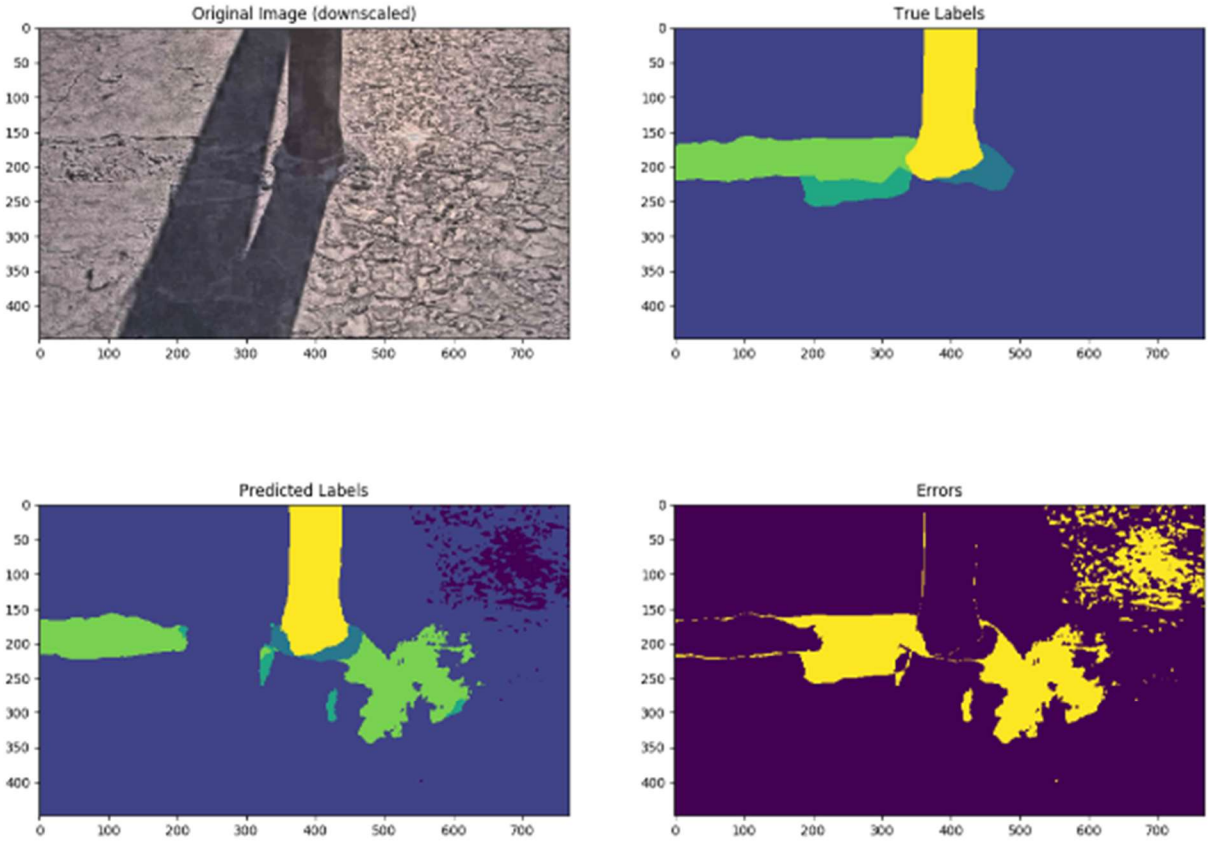
Ensemble Calibrated & Accuracy Weighted - 119-(2019, 2, 27, 12, 14, 12)



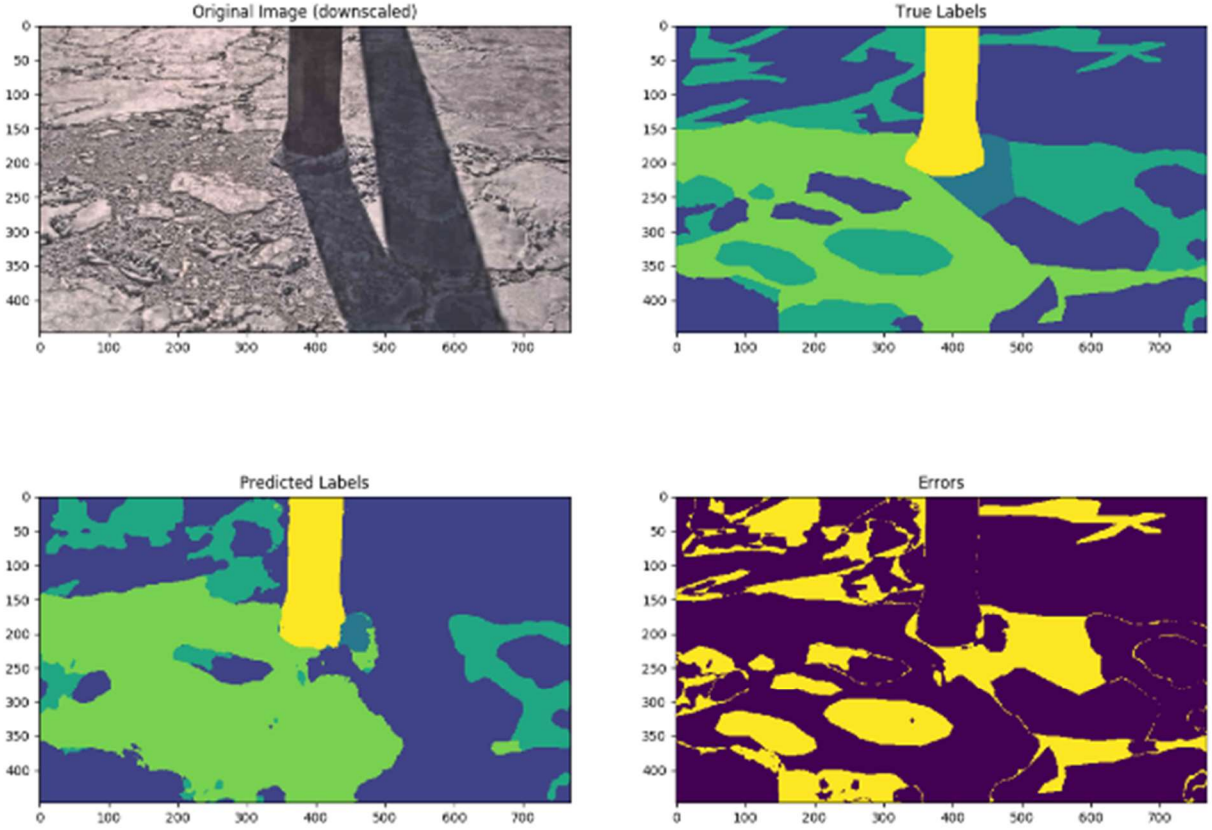
Ensemble Calibrated & Accuracy Weighted - 119-(2019, 2, 28, 13, 30, 59)



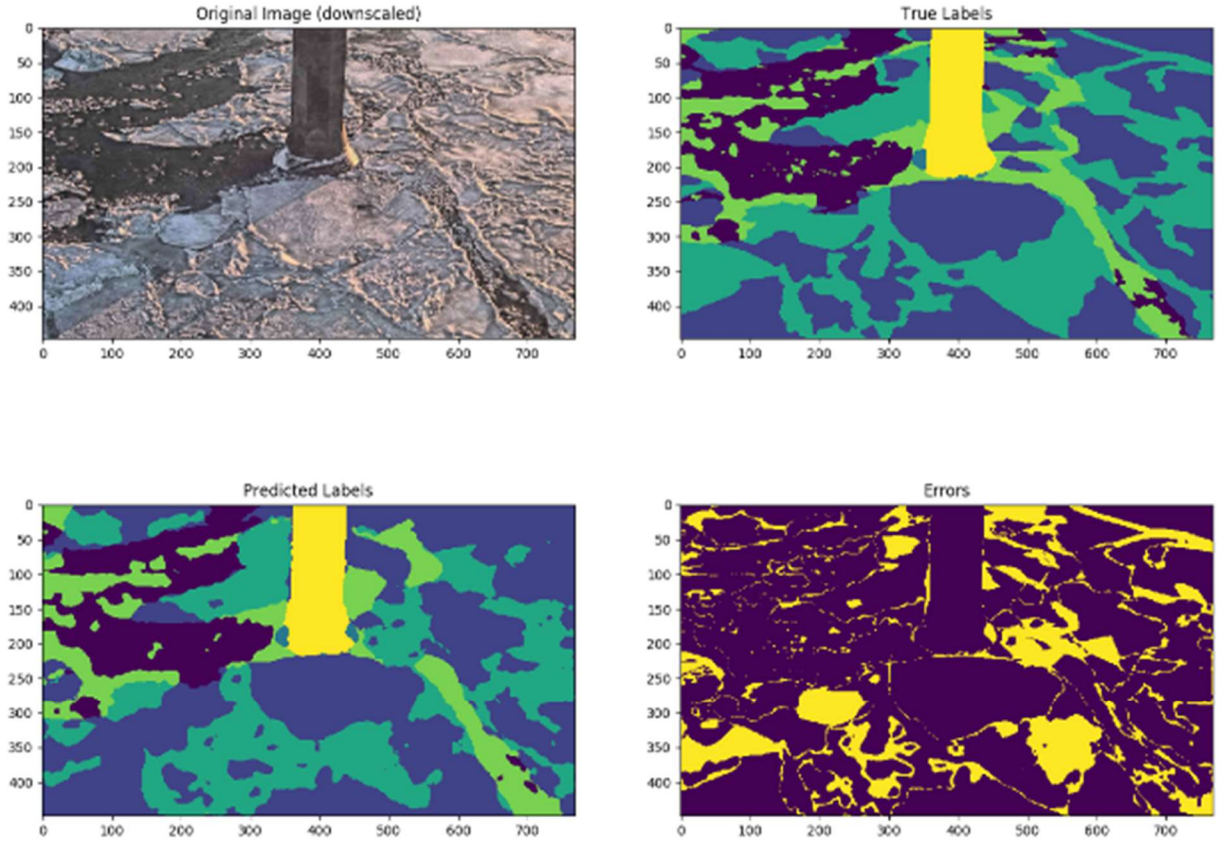
Ensemble Calibrated & Accuracy Weighted - 119-(2019, 2, 28, 14, 22, 31)



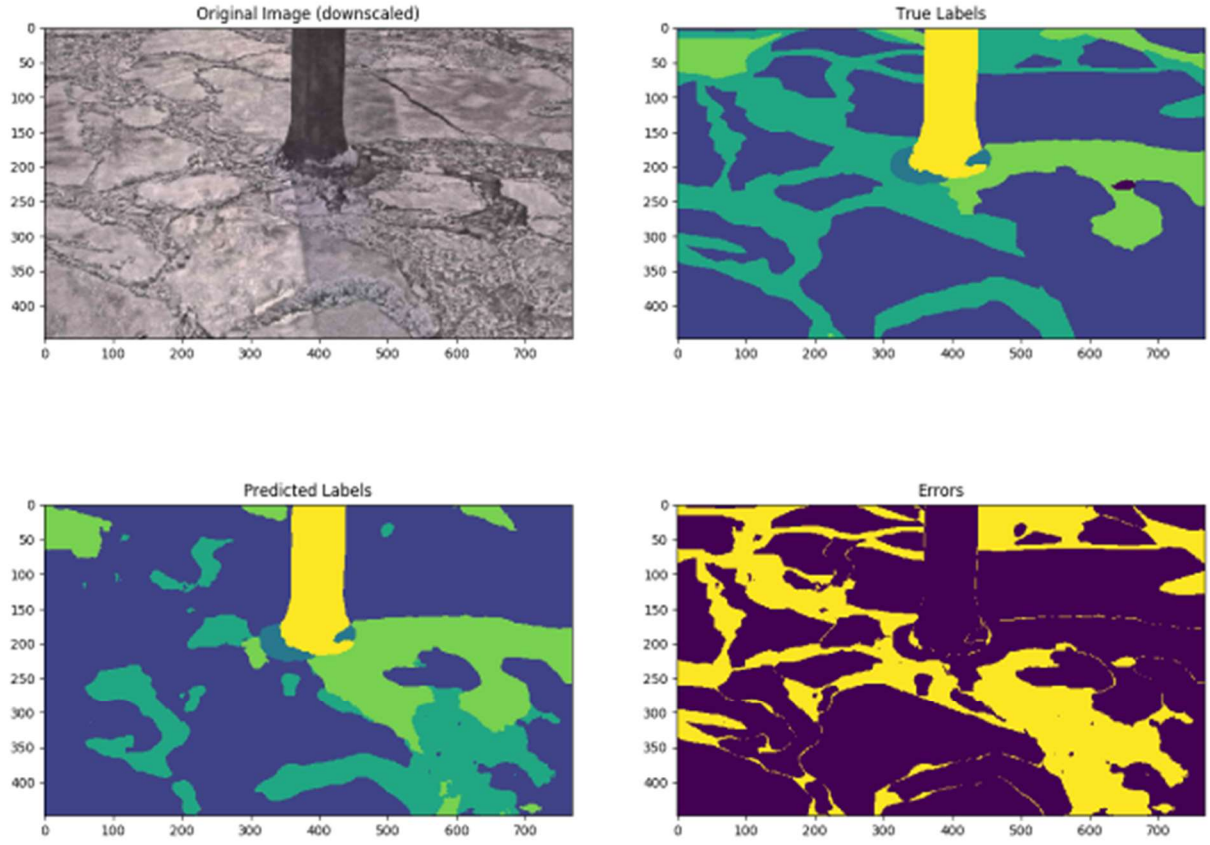
Ensemble Calibrated & Accuracy Weighted - 119-(2019, 3, 1, 13, 10, 53)



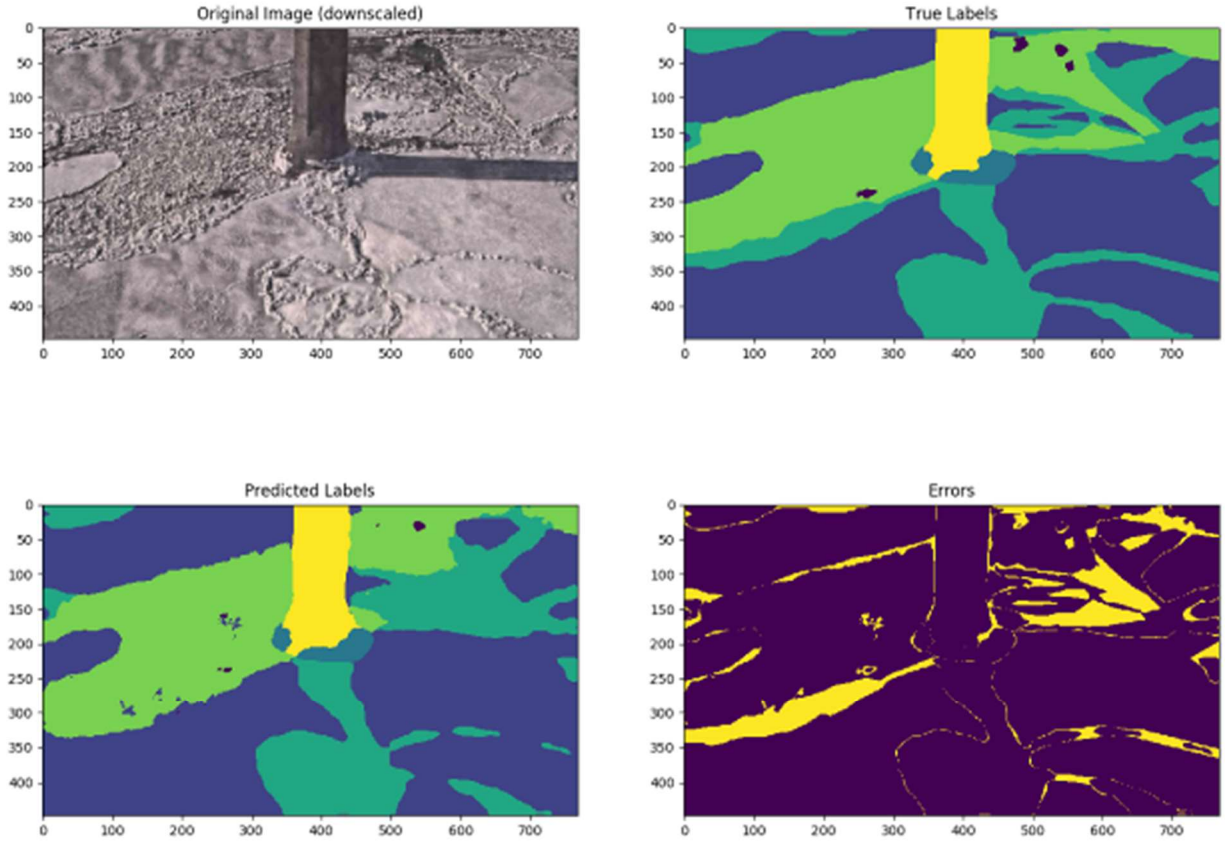
Ensemble Calibrated & Accuracy Weighted - 119-(2019, 3, 1, 16, 37, 26)



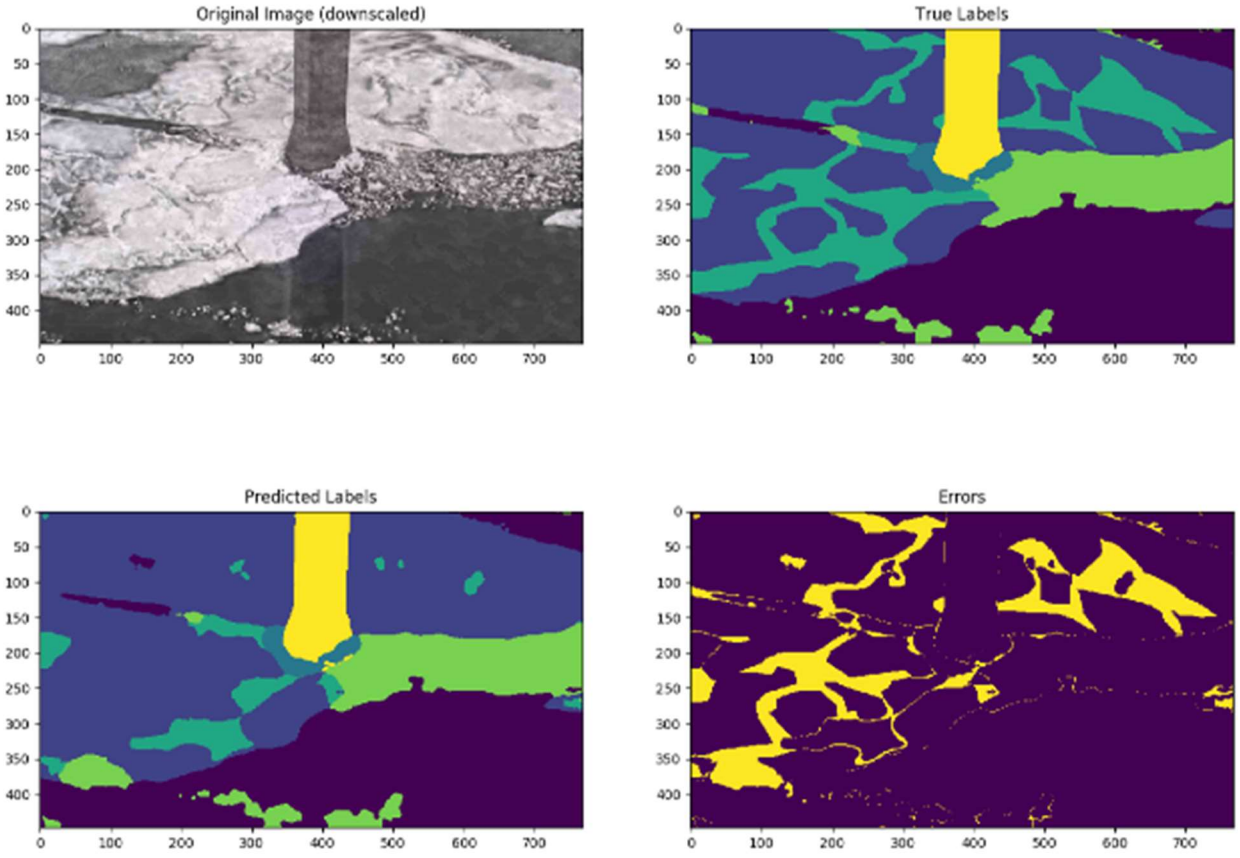
Ensemble Calibrated & Accuracy Weighted - 119-(2019, 3, 10, 14, 21, 40)



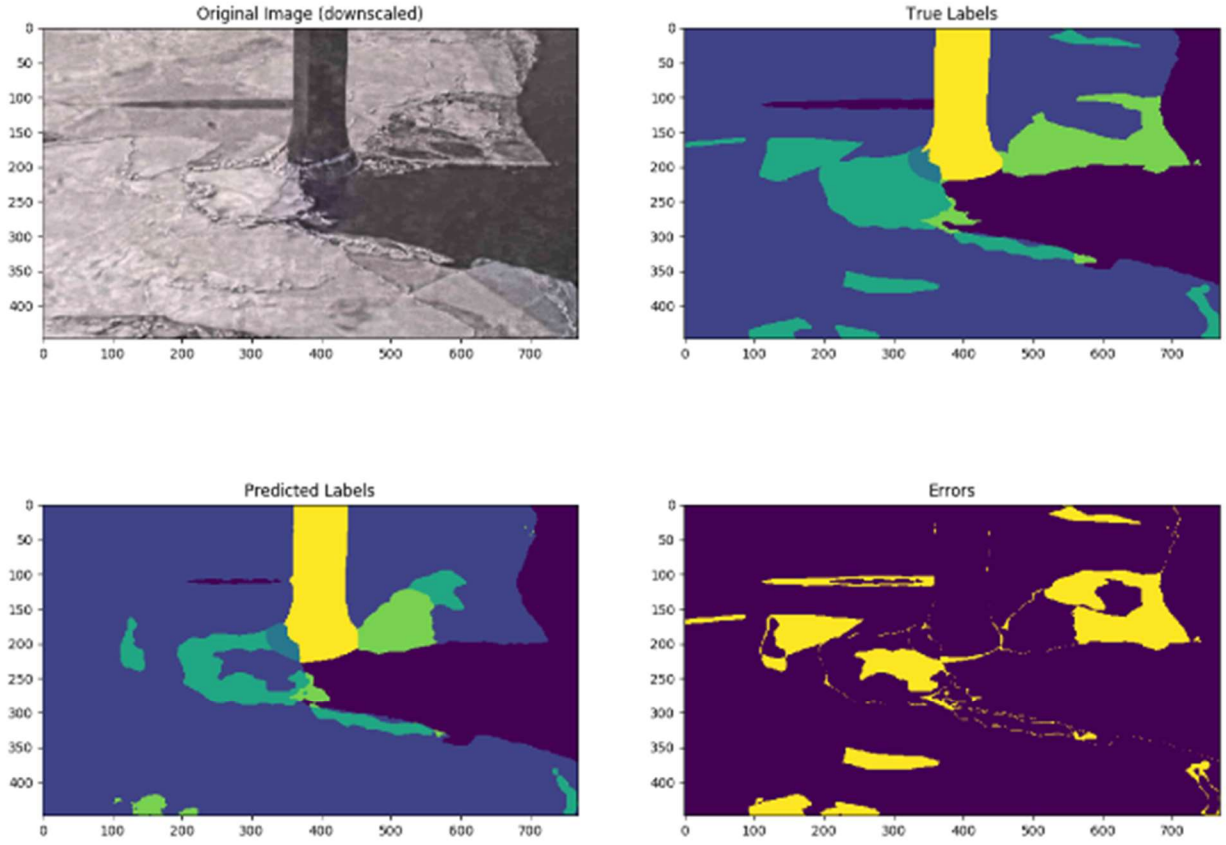
Ensemble Calibrated & Accuracy Weighted - 119-(2019, 3, 10, 9, 38, 18)



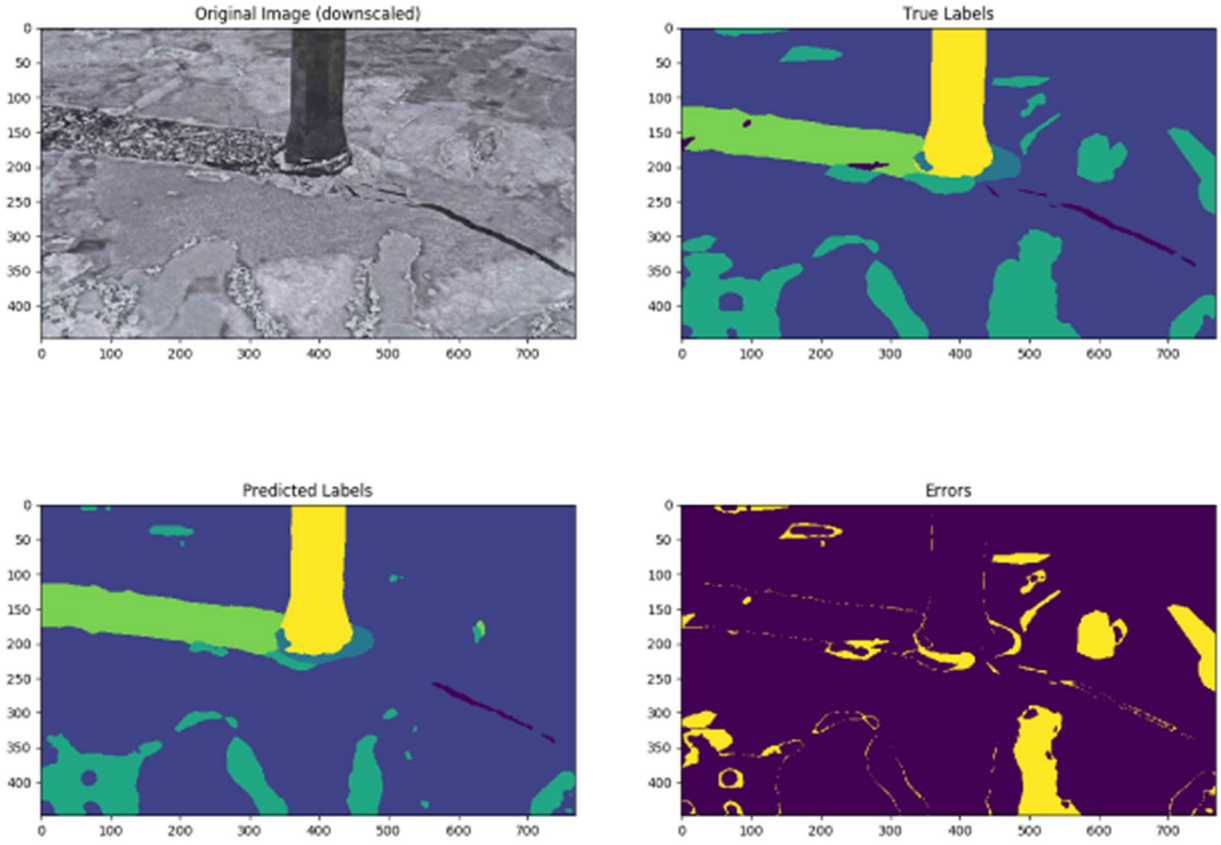
Ensemble Calibrated & Accuracy Weighted - 119-(2019, 3, 11, 14, 17, 21)



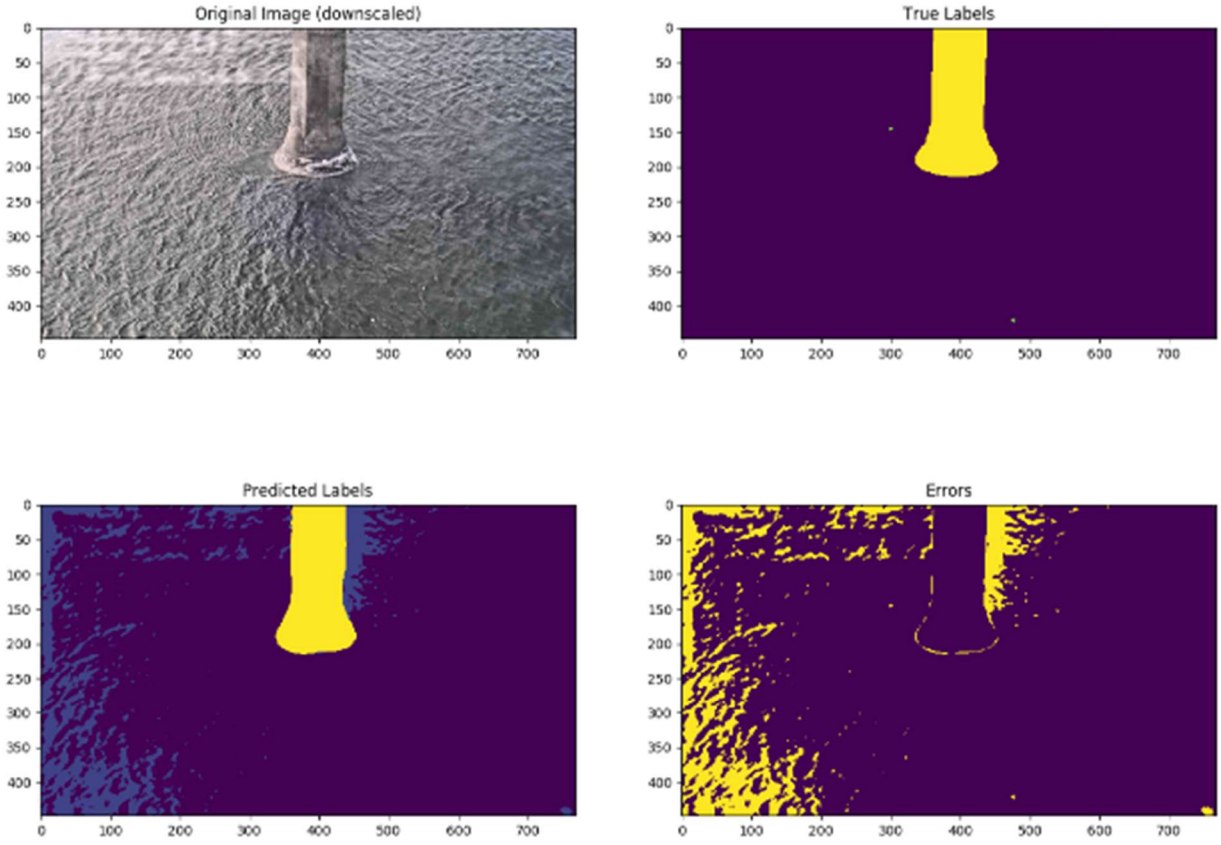
Ensemble Calibrated & Accuracy Weighted - 119-(2019, 3, 2, 11, 40, 29)



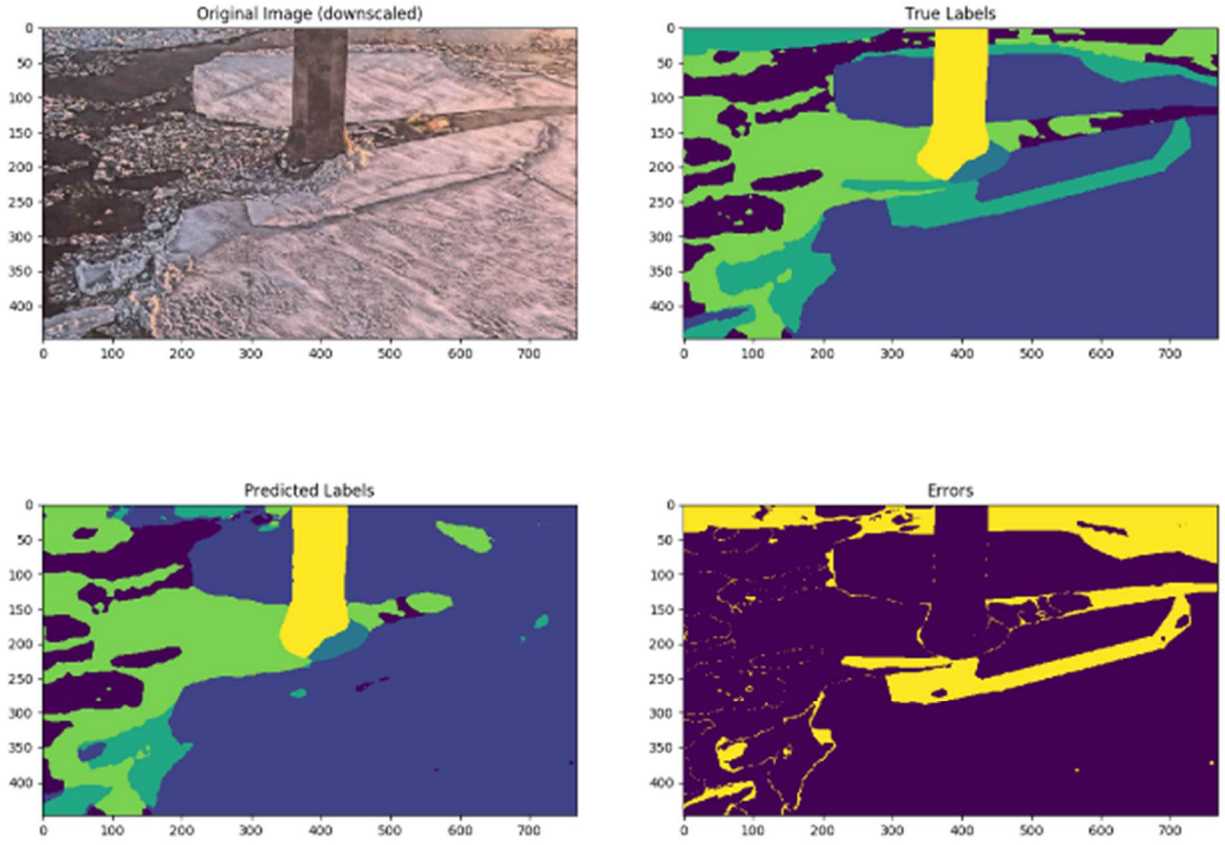
Ensemble Calibrated & Accuracy Weighted - 119-(2019, 3, 2, 16, 9, 41)

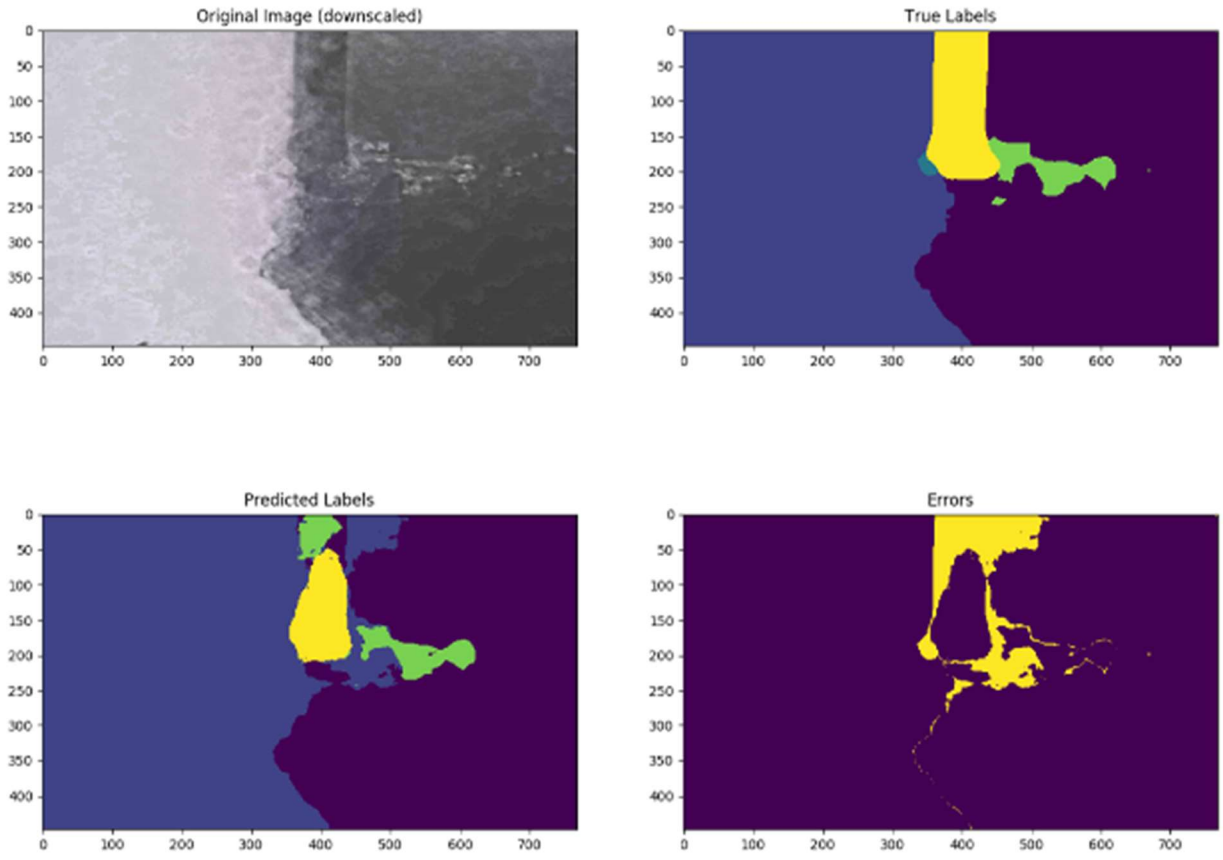


Ensemble Calibrated & Accuracy Weighted - 119-(2019, 3, 3, 10, 15, 37)

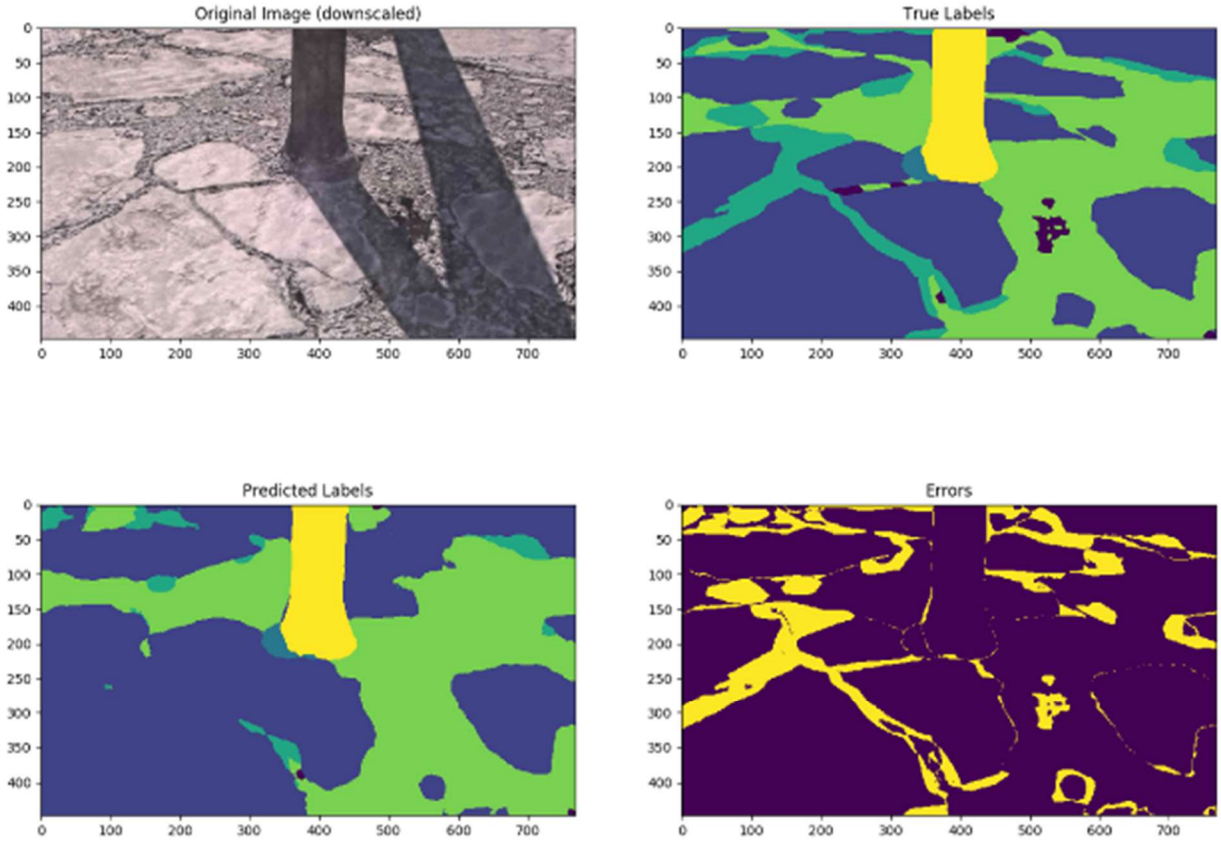


Ensemble Calibrated & Accuracy Weighted - 119-(2019, 3, 3, 16, 46, 4)

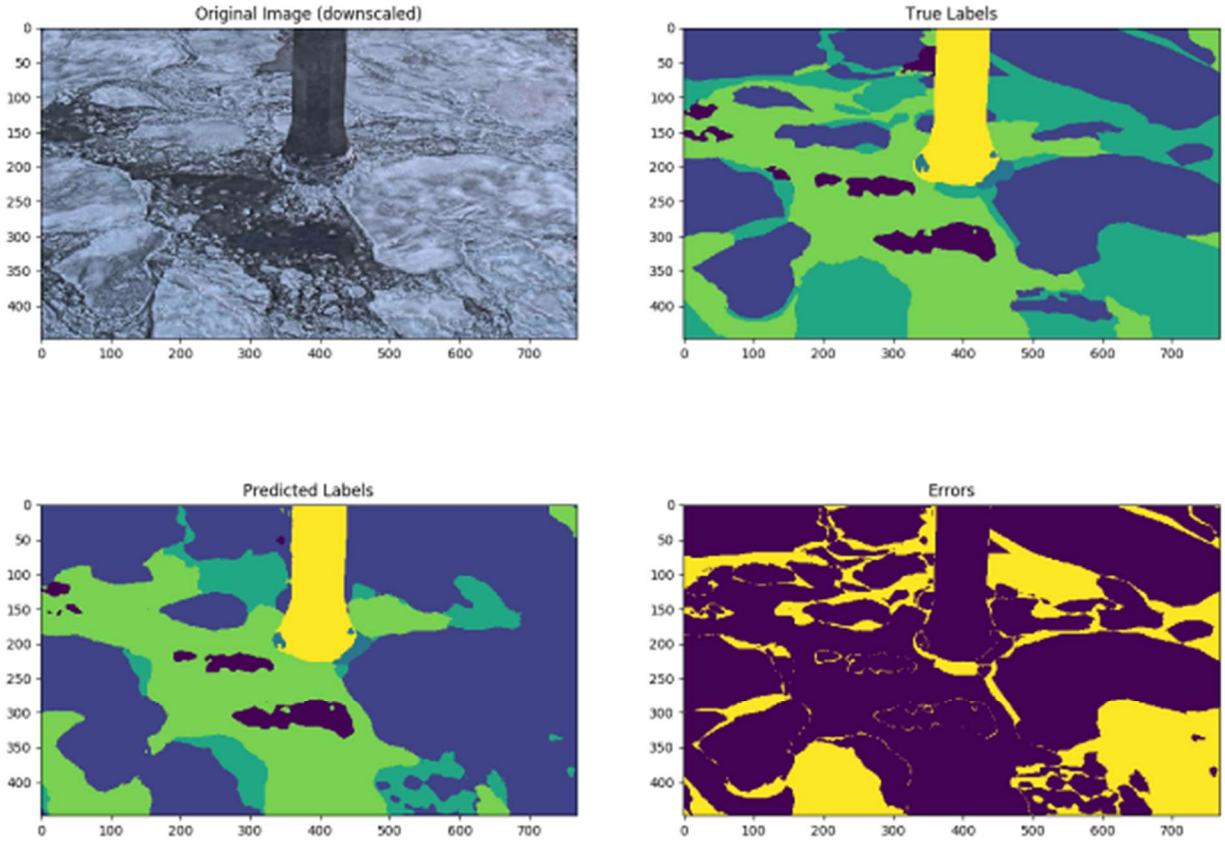



Ensemble Calibrated & Accuracy Weighted - 119-(2019, 3, 4, 10, 32, 52)
Original Image (downscaled)
True Labels
Predicted Labels
Errors

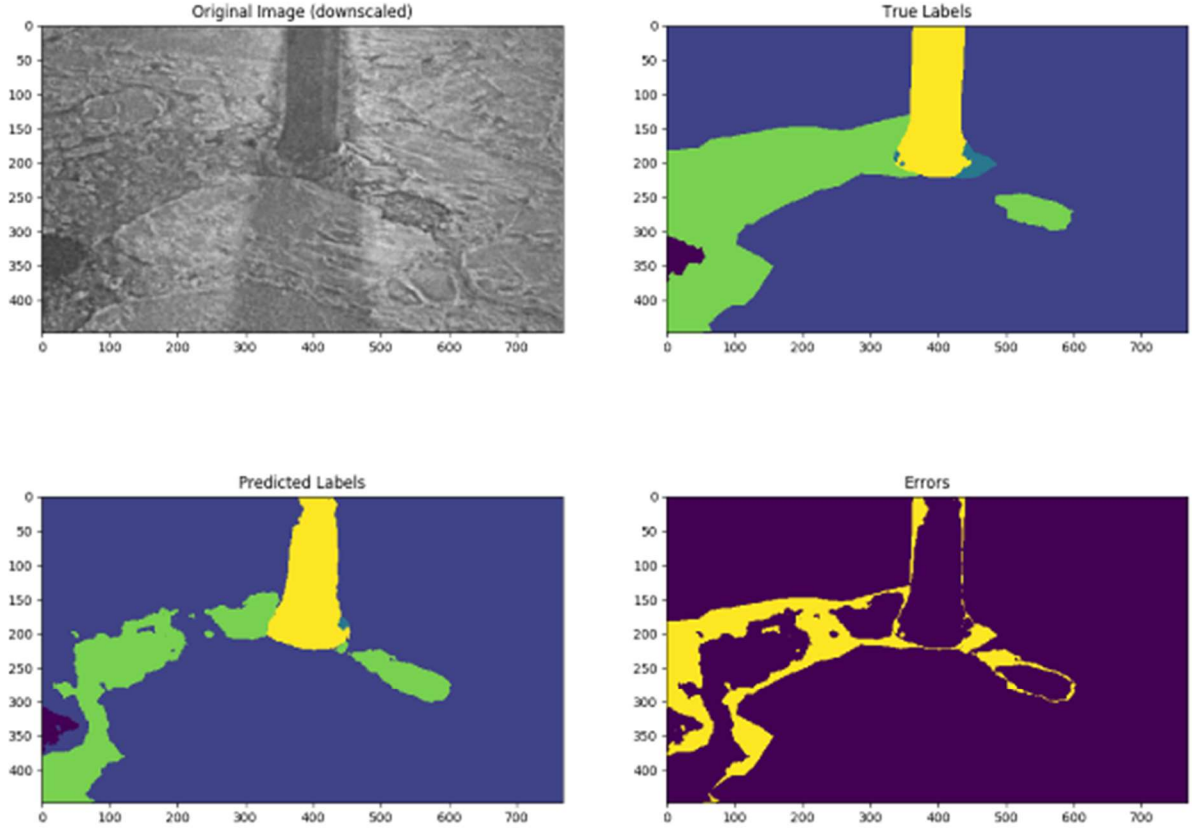
Ensemble Calibrated & Accuracy Weighted - 119-(2019, 3, 5, 12, 40, 4)



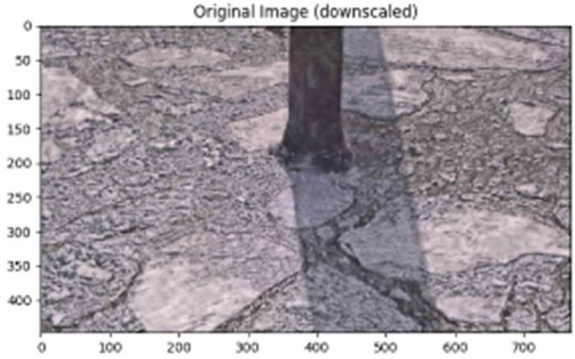
Ensemble Calibrated & Accuracy Weighted - 119-(2019, 3, 5, 17, 0, 51)

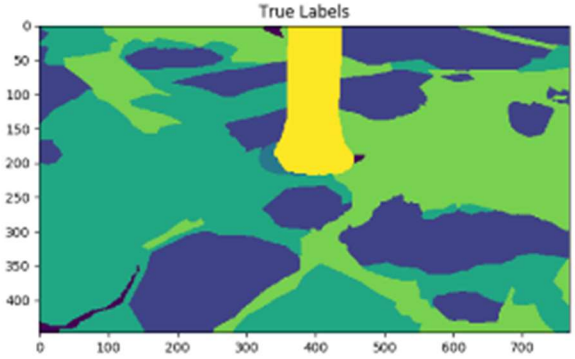


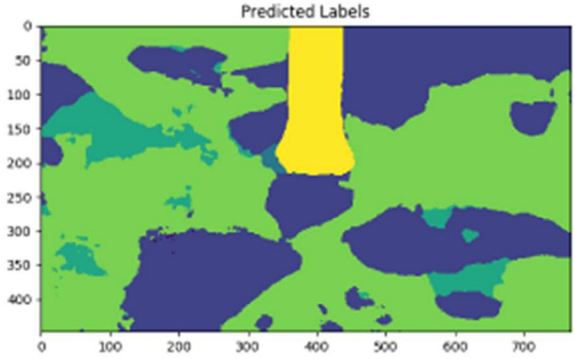
Ensemble Calibrated & Accuracy Weighted - 119-(2019, 3, 5, 17, 50, 6)

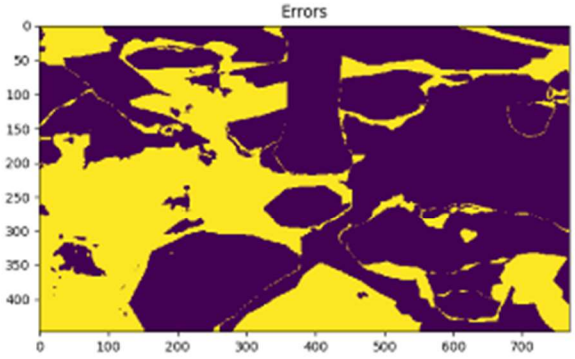


Ensemble Calibrated & Accuracy Weighted - 119-(2019, 3, 6, 13, 25, 47)

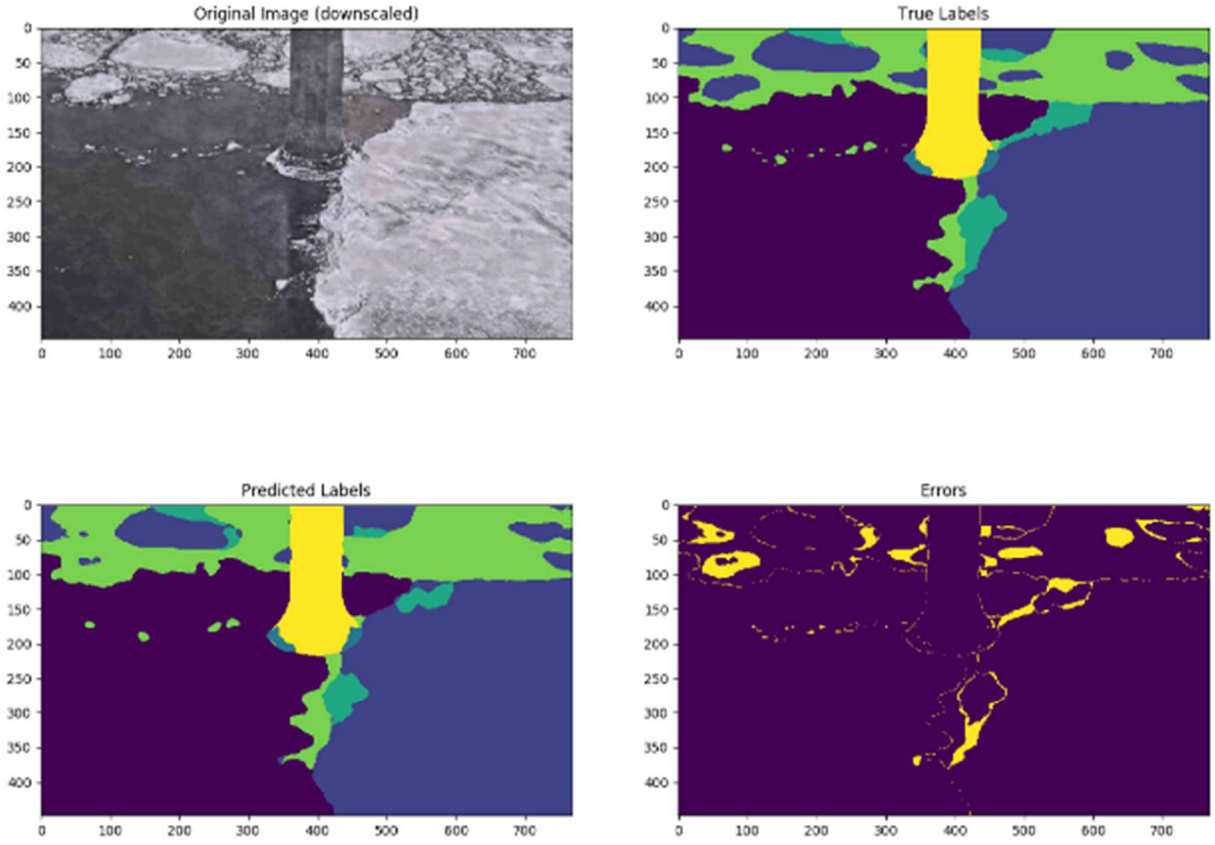




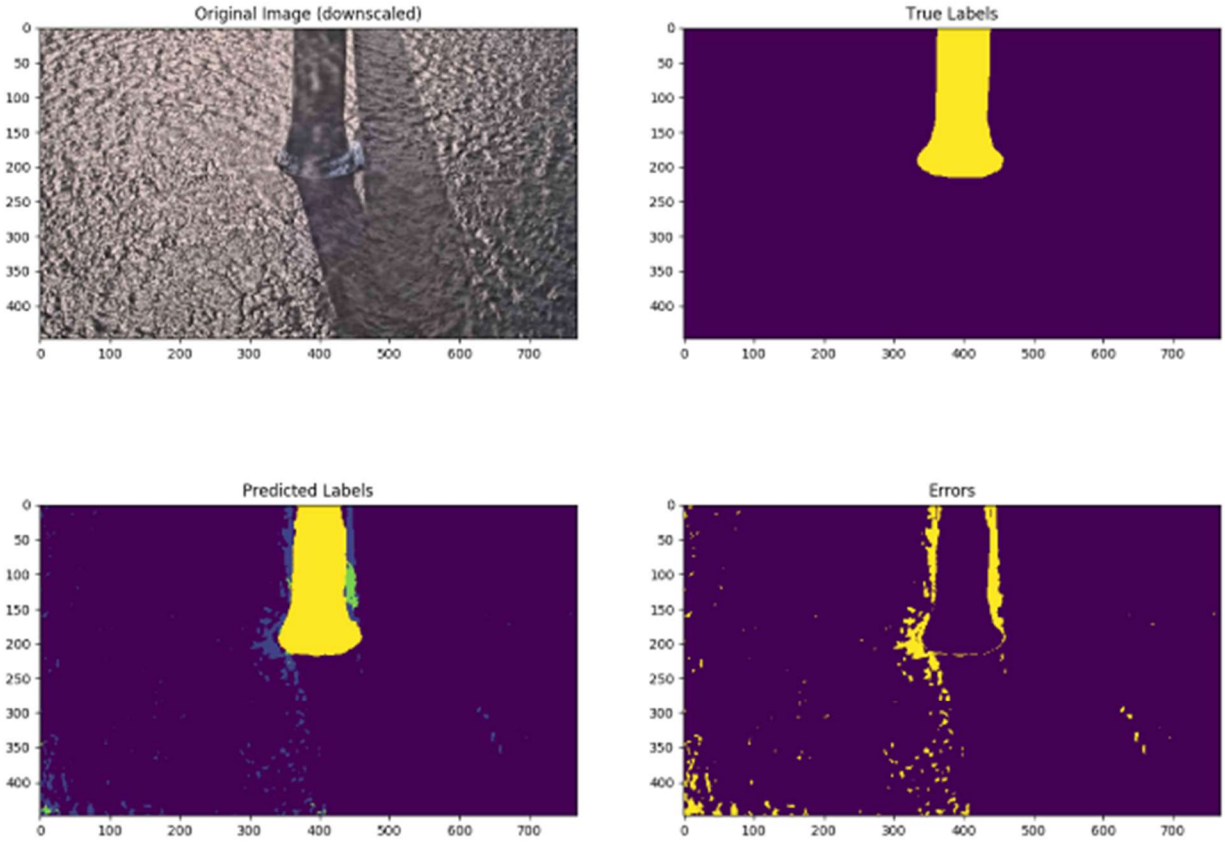




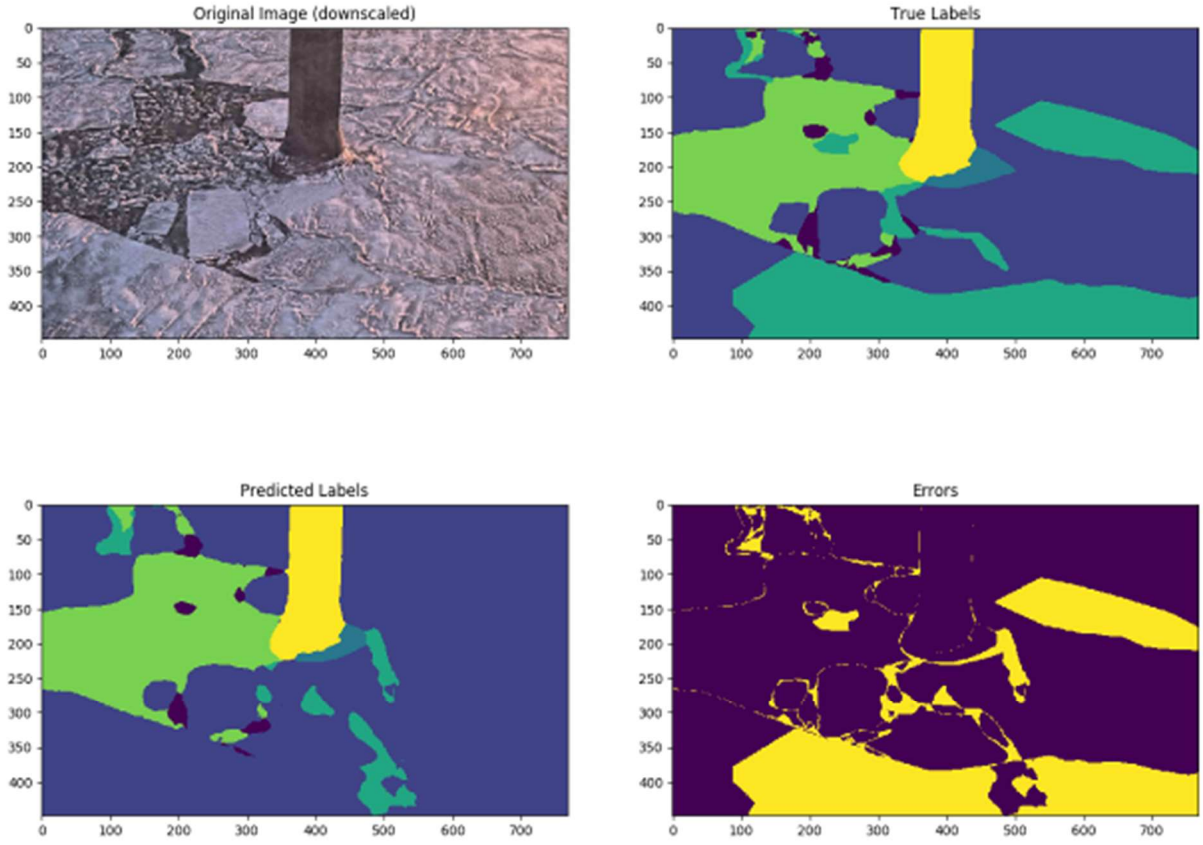
Ensemble Calibrated & Accuracy Weighted - 119-(2019, 3, 6, 15, 38, 59)



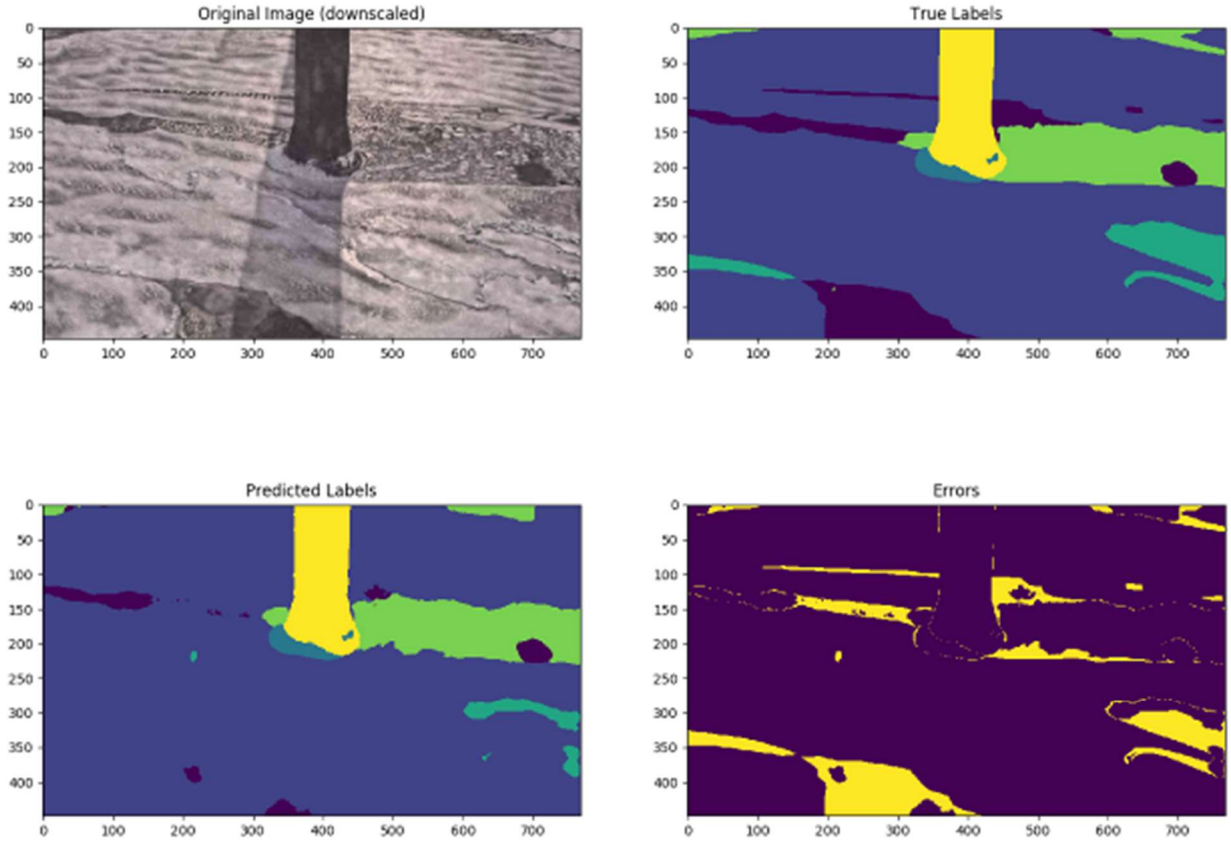
Ensemble Calibrated & Accuracy Weighted - 119-(2019, 3, 7, 13, 10, 55)



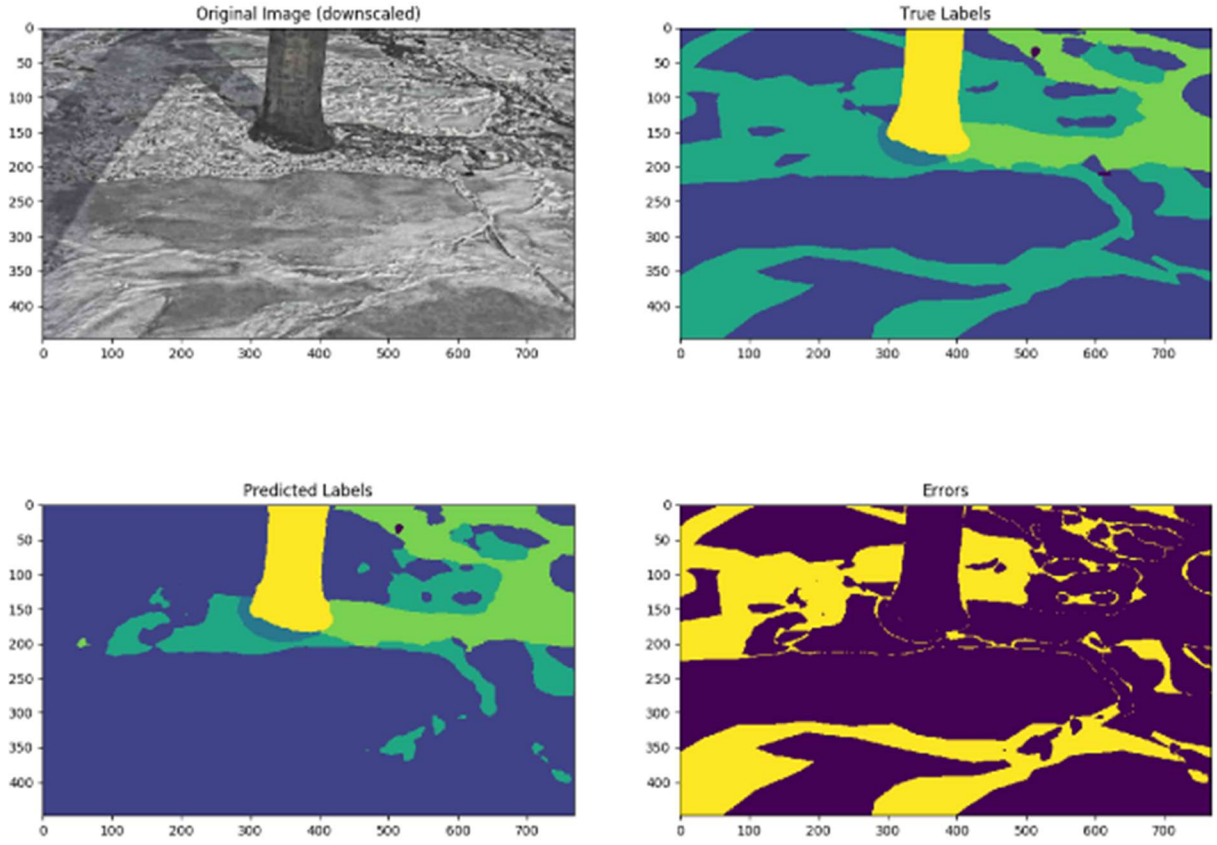
Ensemble Calibrated & Accuracy Weighted - 119-(2019, 3, 7, 16, 53, 16)



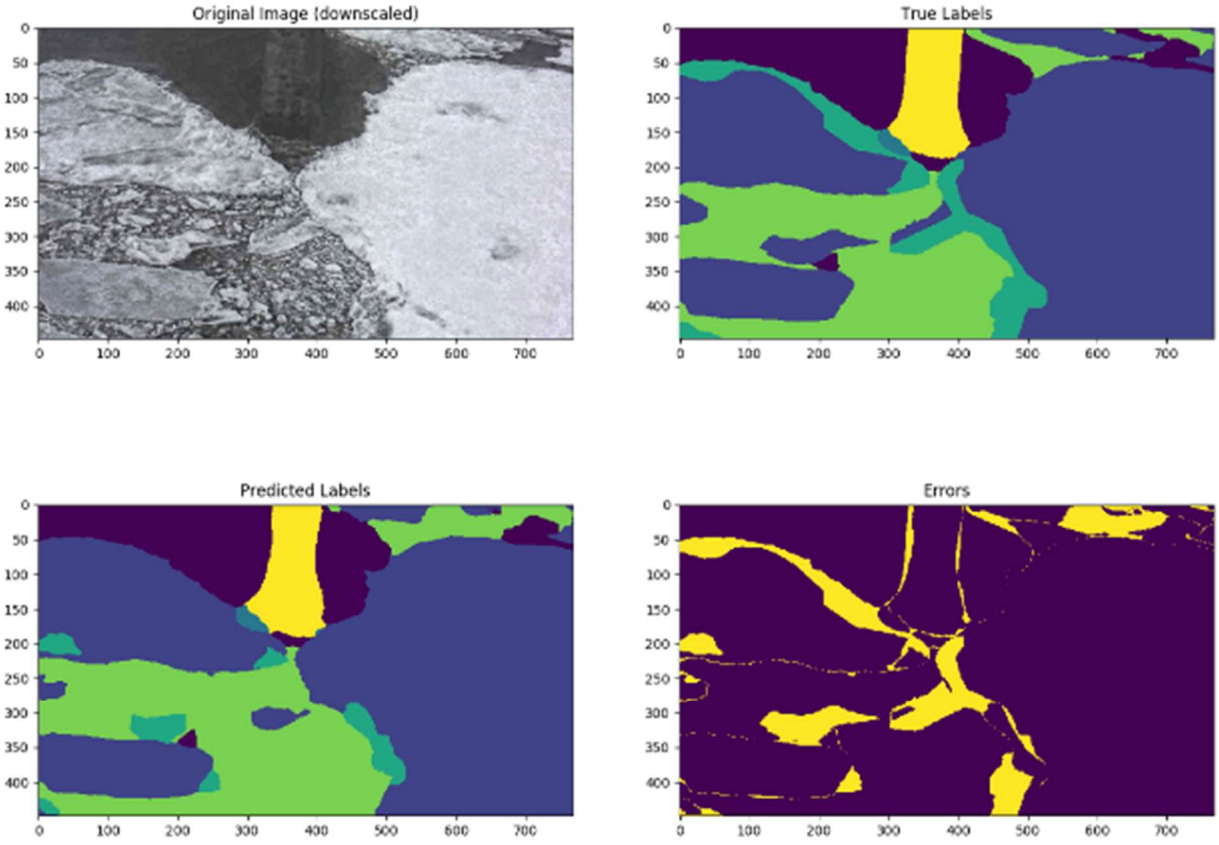
Ensemble Calibrated & Accuracy Weighted - 119-(2019, 3, 9, 13, 51, 33)



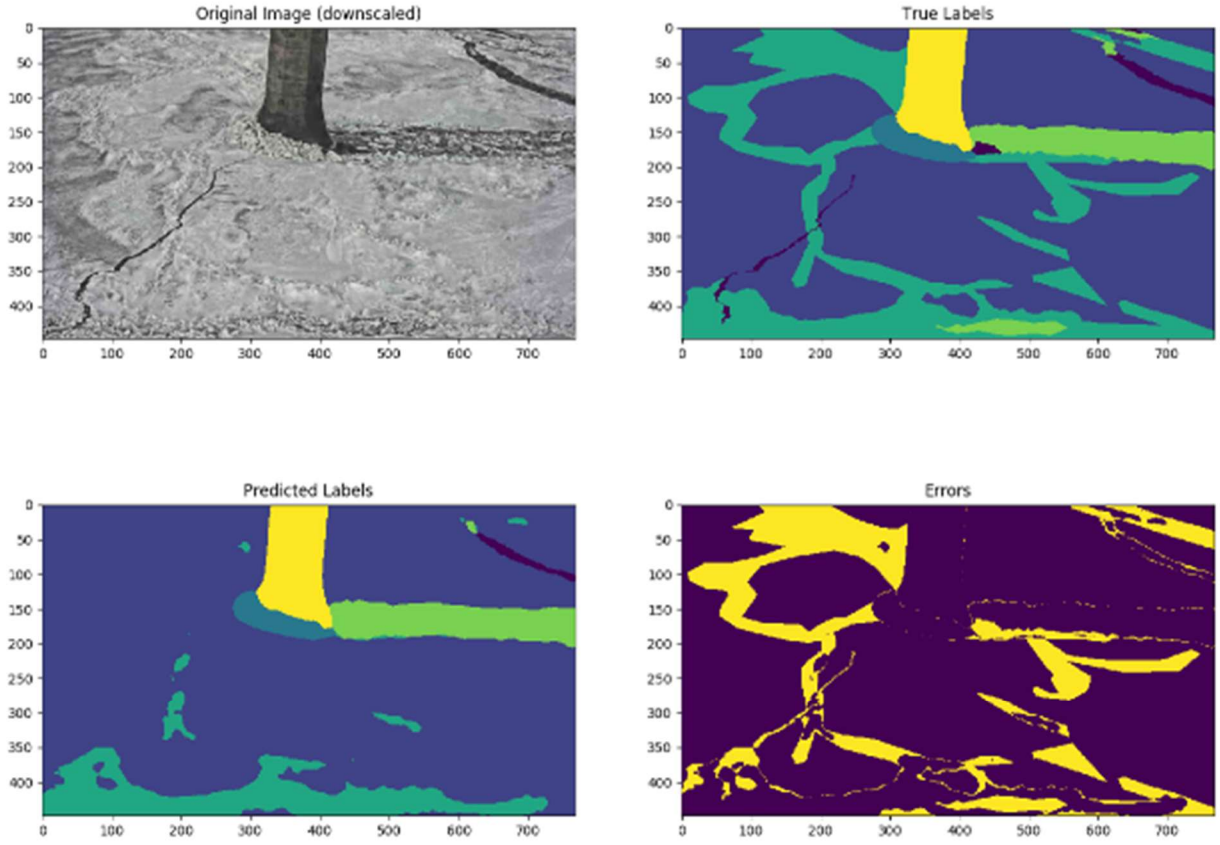
Ensemble Calibrated & Accuracy Weighted - 120-(2019, 2, 25, 12, 26, 11)



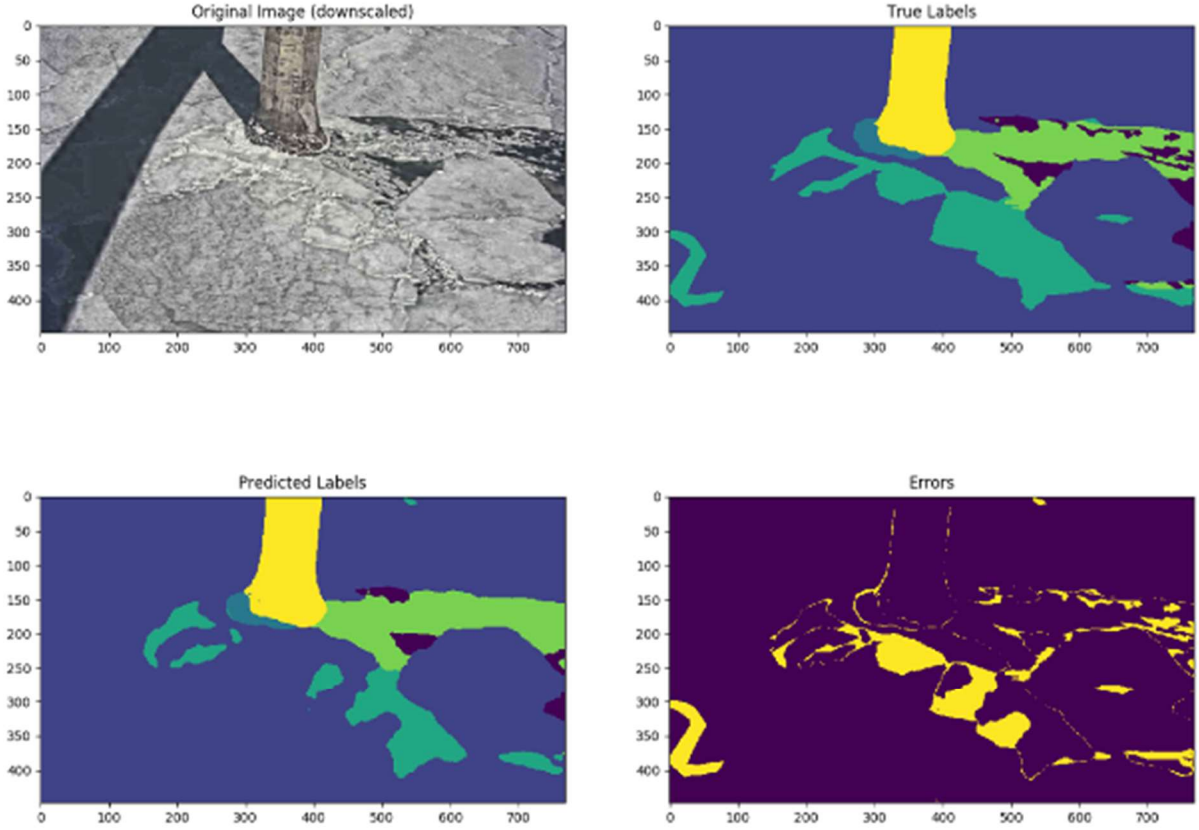
Ensemble Calibrated & Accuracy Weighted - 120-(2019, 2, 25, 17, 10, 15)



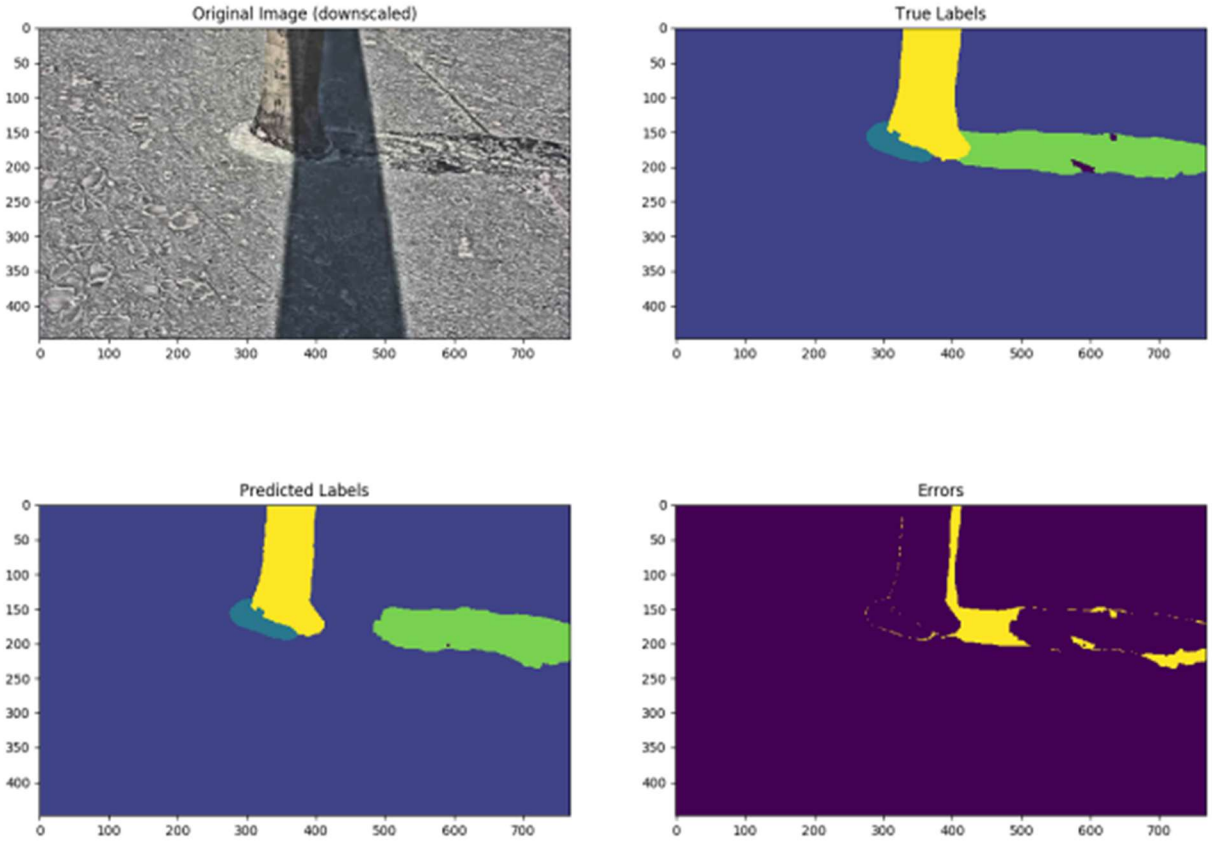
Ensemble Calibrated & Accuracy Weighted - 120-(2019, 2, 26, 15, 20, 57)



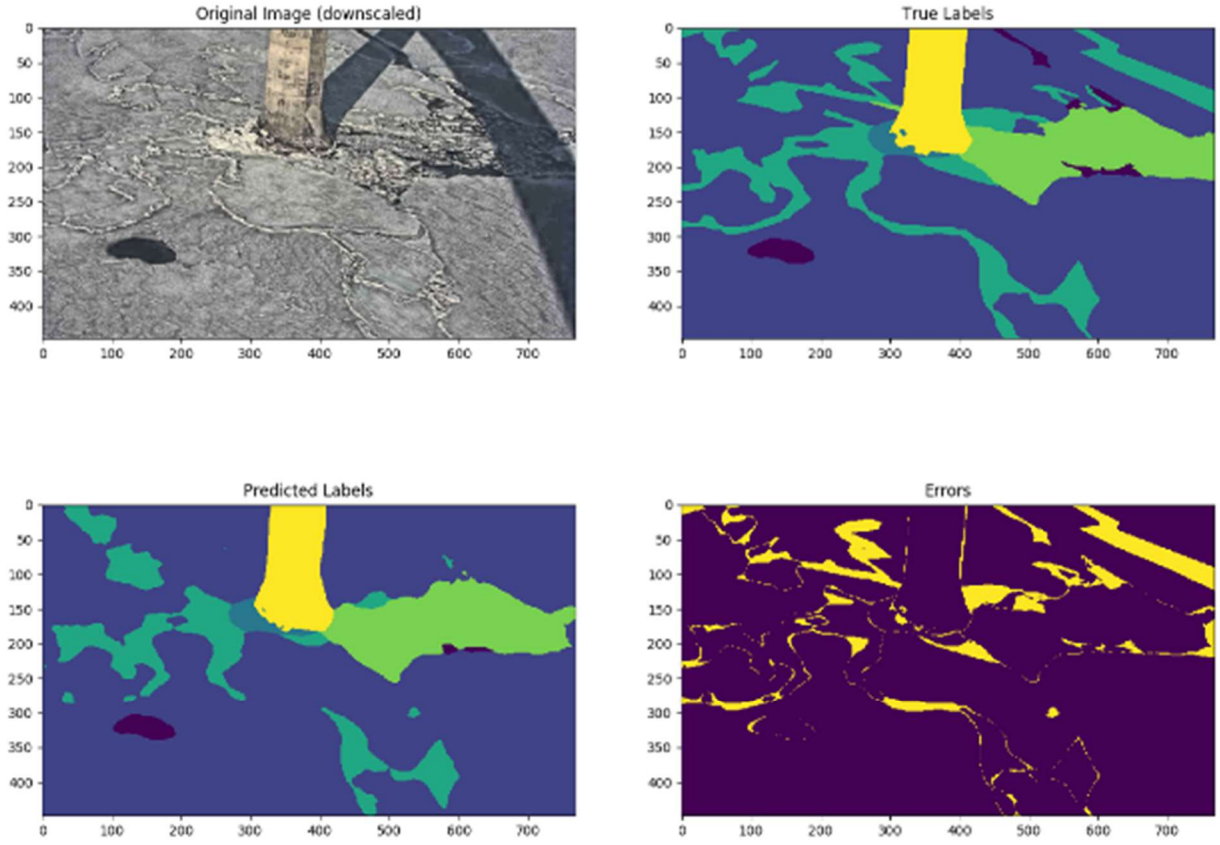
Ensemble Calibrated & Accuracy Weighted - 120-(2019, 2, 27, 12, 42, 42)



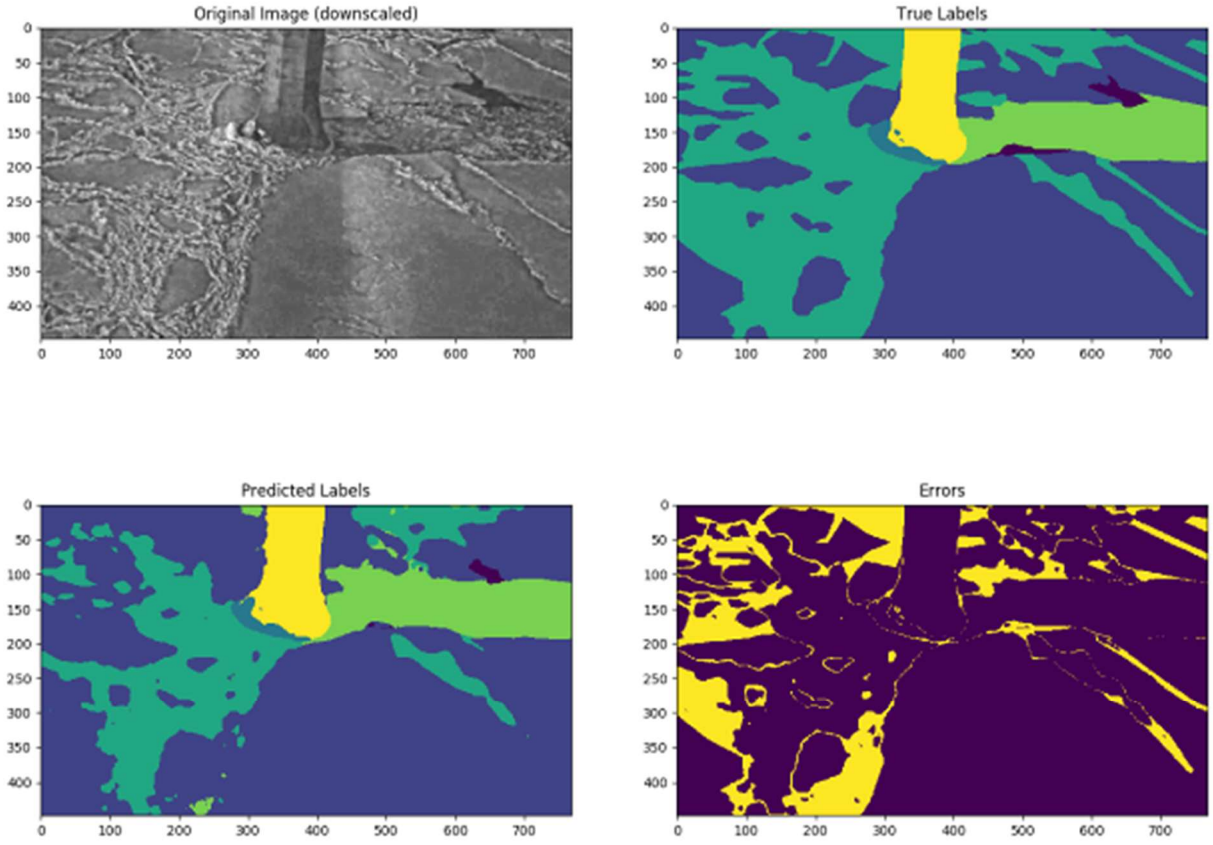
Ensemble Calibrated & Accuracy Weighted - 120-(2019, 2, 27, 14, 15, 15)



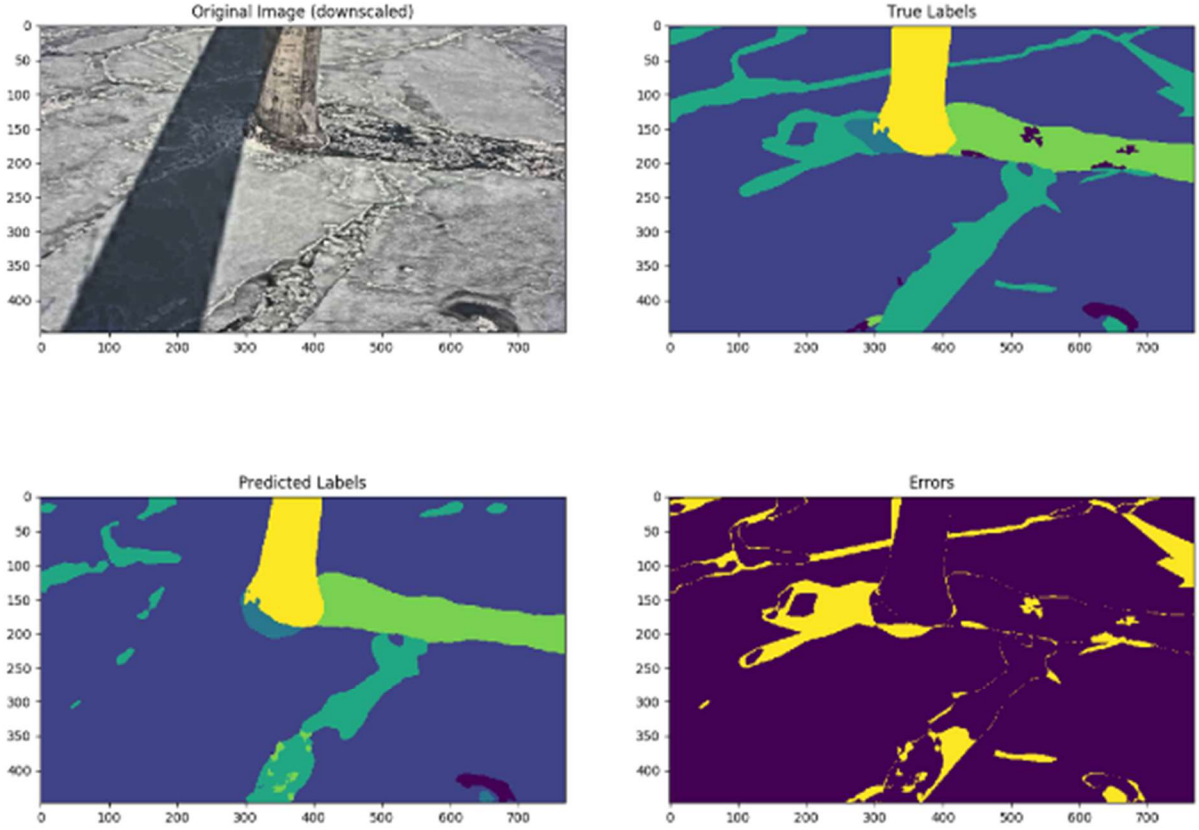
Ensemble Calibrated & Accuracy Weighted - 120-(2019, 2, 27, 15, 4, 36)



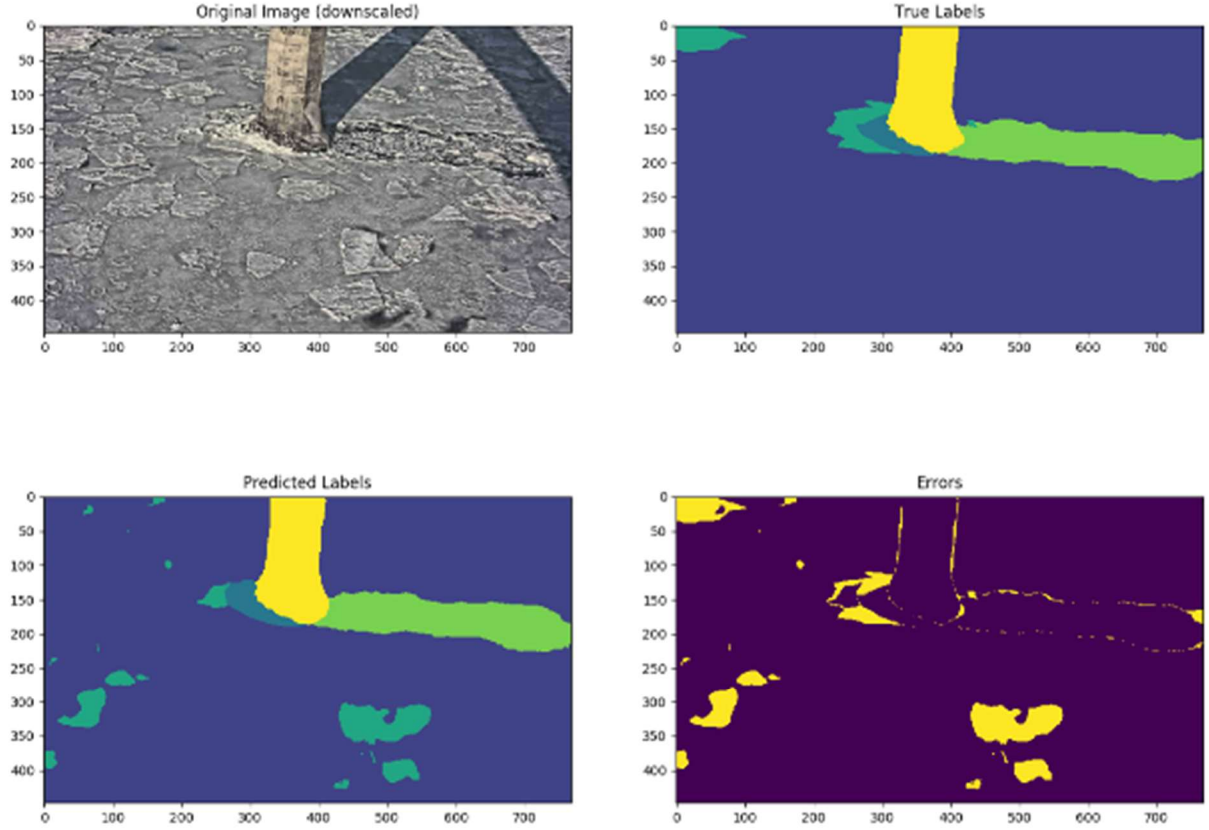
Ensemble Calibrated & Accuracy Weighted - 120-(2019, 2, 27, 17, 33, 58)



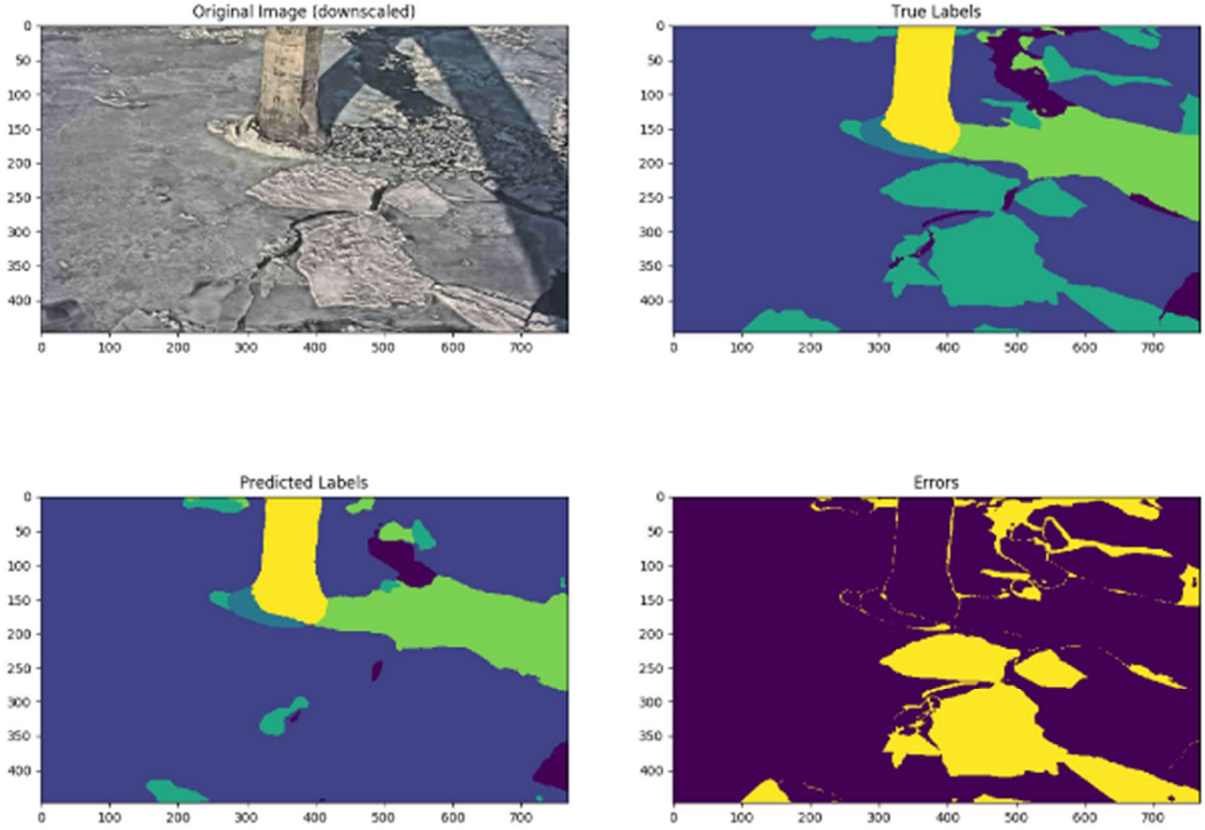
Ensemble Calibrated & Accuracy Weighted - 120-(2019, 2, 28, 13, 23, 36)



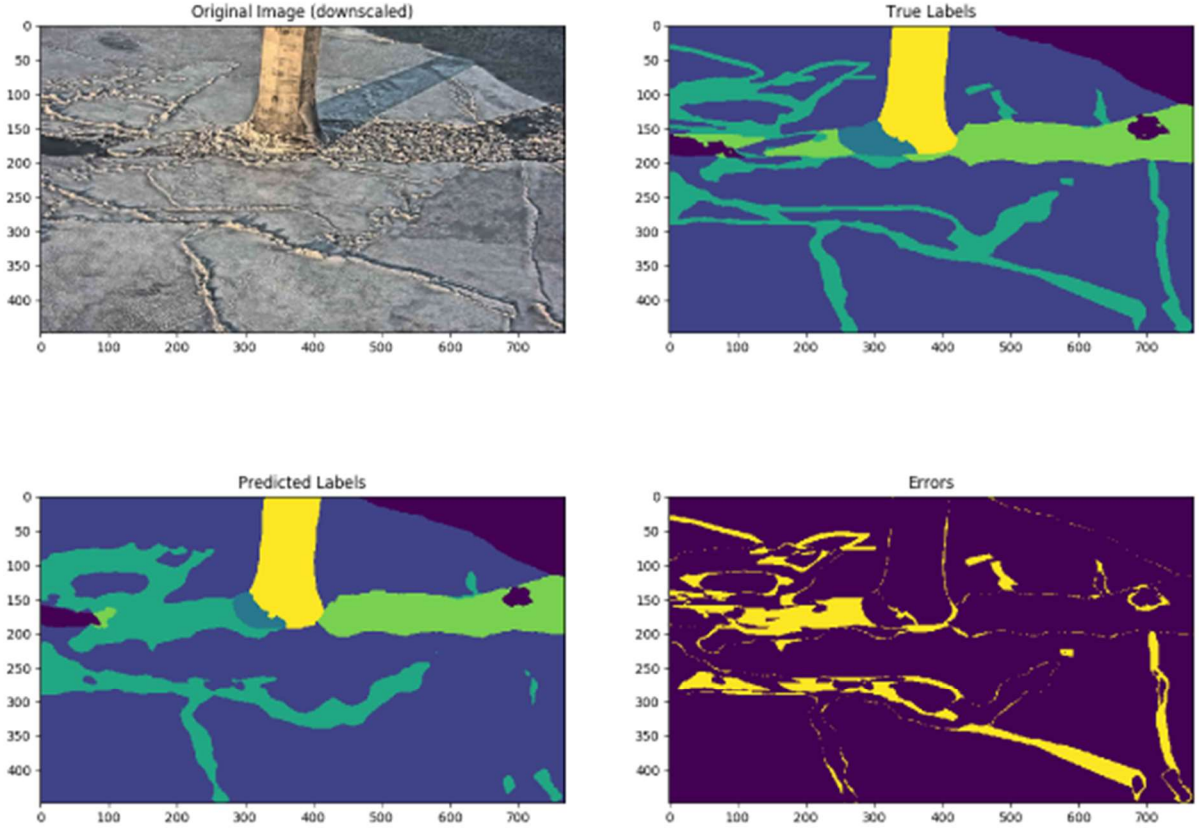
Ensemble Calibrated & Accuracy Weighted - 120-(2019, 2, 28, 15, 15, 17)



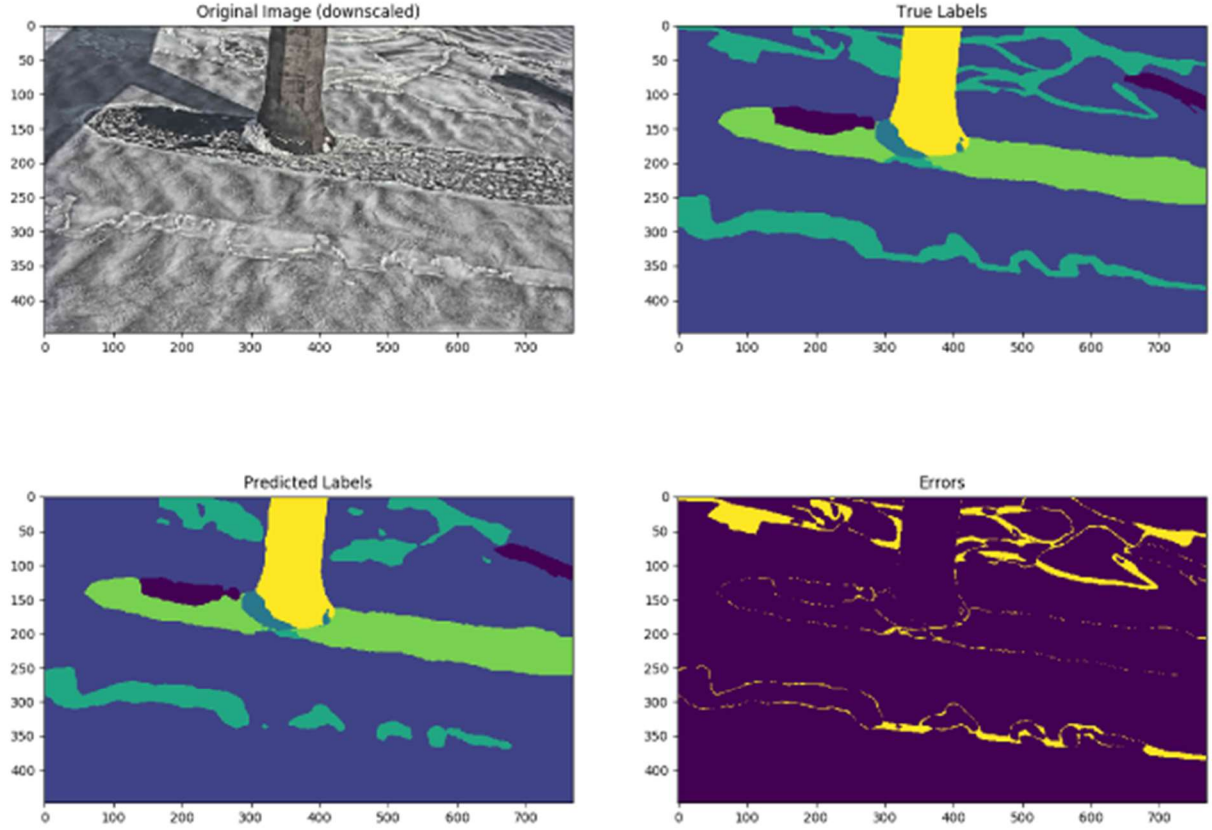
Ensemble Calibrated & Accuracy Weighted - 120-(2019, 3, 1, 15, 2, 16)



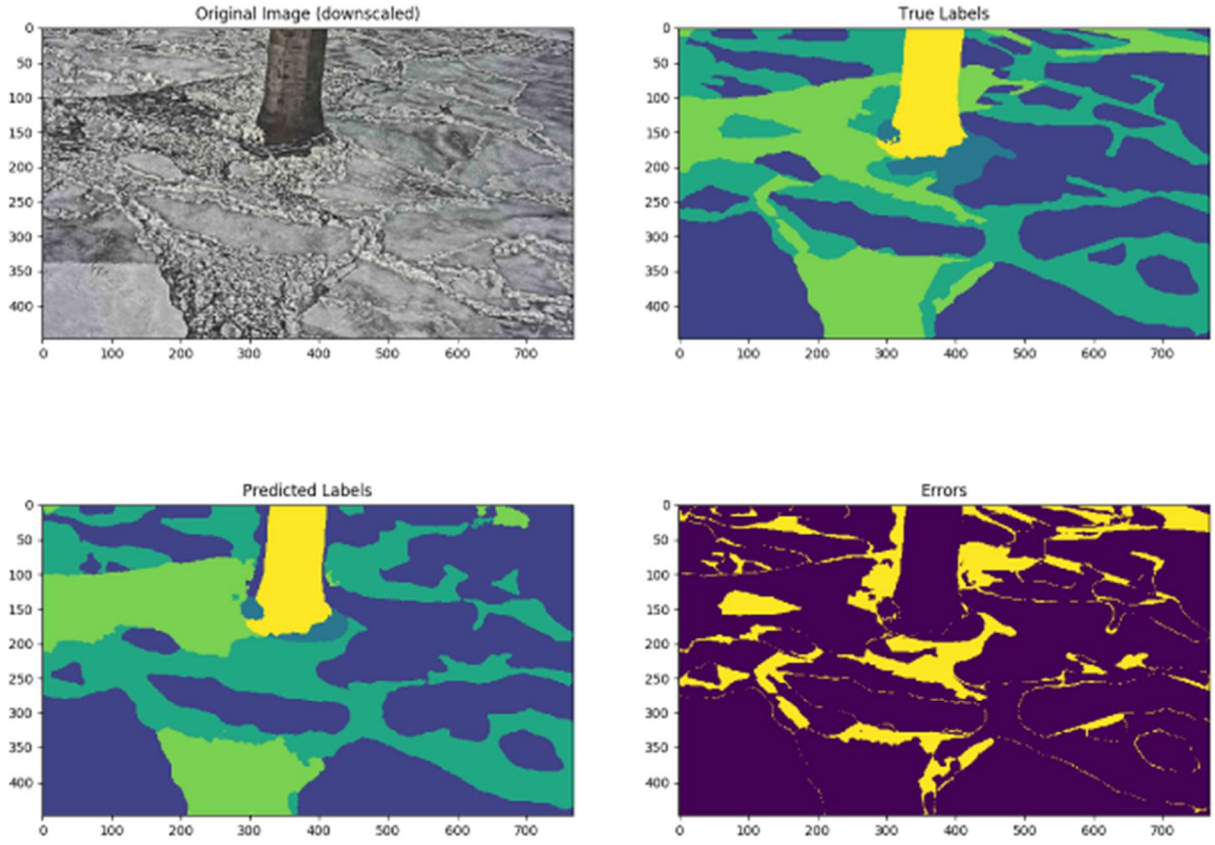
Ensemble Calibrated & Accuracy Weighted - 120-(2019, 3, 1, 16, 23, 4)



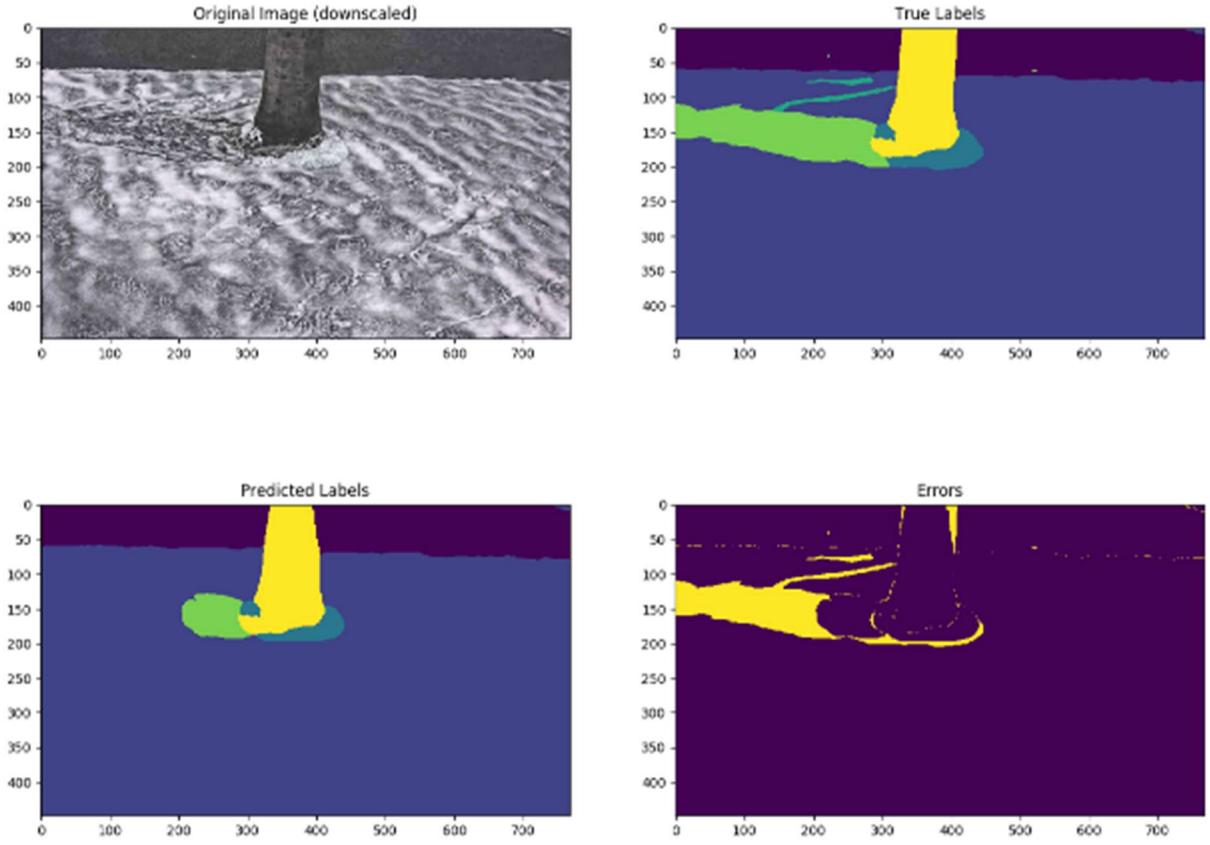
Ensemble Calibrated & Accuracy Weighted - 120-(2019, 3, 10, 12, 11, 41)



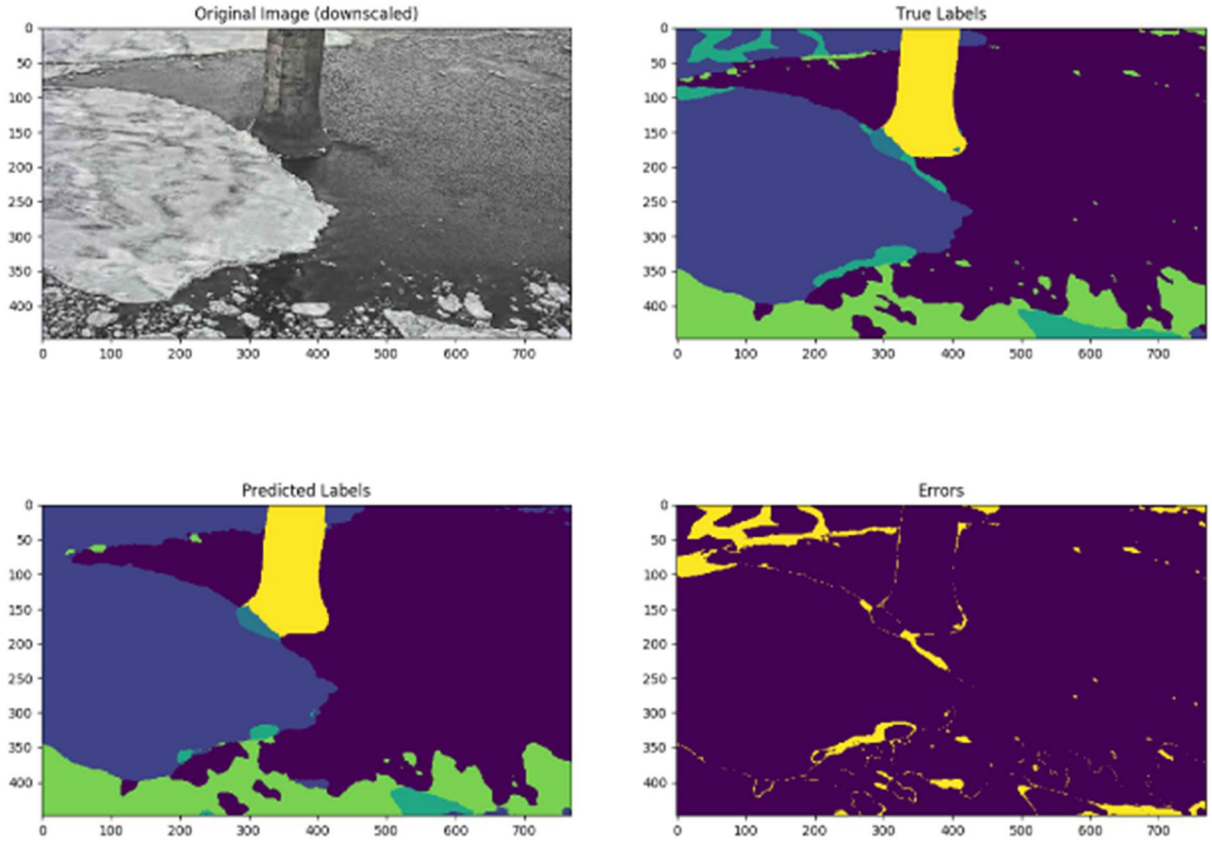
Ensemble Calibrated & Accuracy Weighted - 120-(2019, 3, 10, 16, 37, 42)



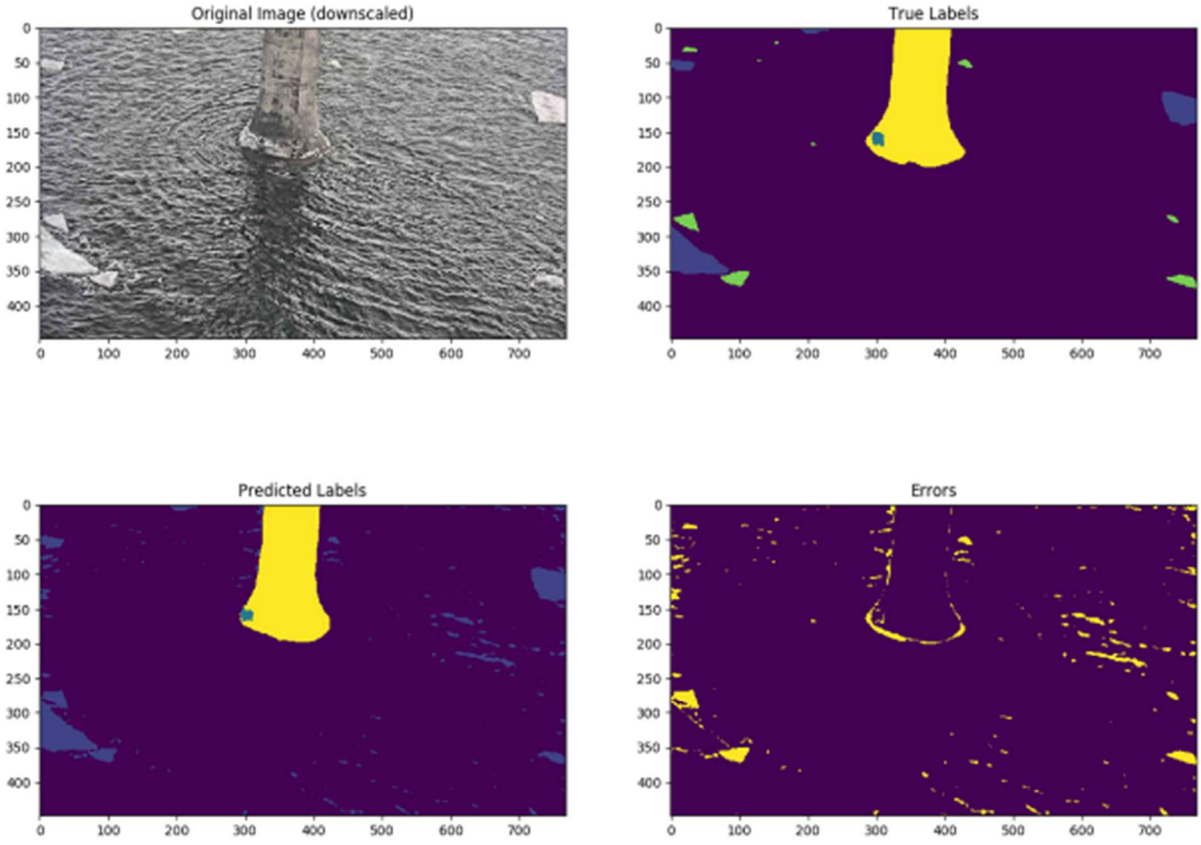
Ensemble Calibrated & Accuracy Weighted - 120-(2019, 3, 10, 18, 25, 19)



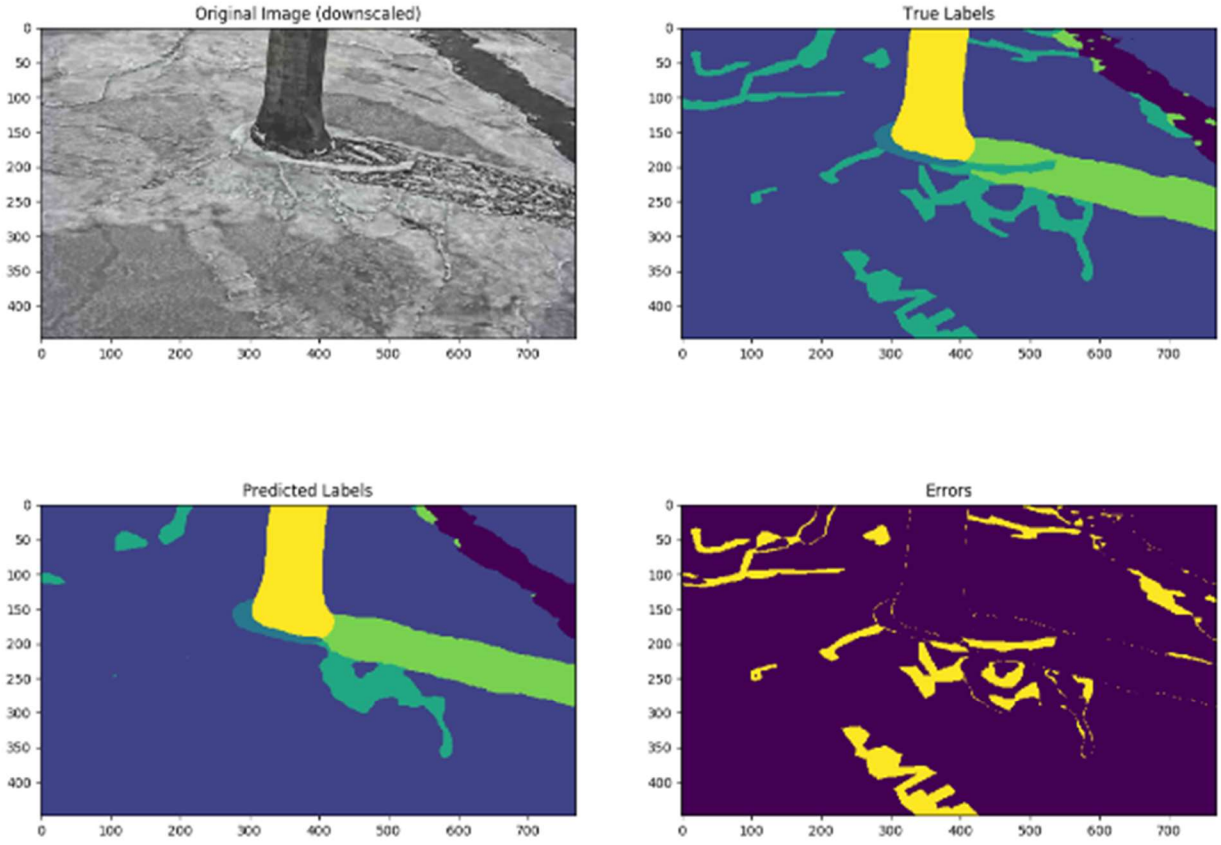
Ensemble Calibrated & Accuracy Weighted - 120-(2019, 3, 11, 12, 58, 39)



Ensemble Calibrated & Accuracy Weighted - 120-(2019, 3, 12, 9, 30, 40)



Ensemble Calibrated & Accuracy Weighted - 120-(2019, 3, 2, 16, 27, 47)



Appendix C – Out-of-domain test

This appendix presents samples outputs when the ensemble has been tasked to predict on images which are from a completely separate domain from the training set. Specifically, the imagery presented in this appendix was collected from cameras mounted on the CCGS Amundsen; a Pierre Radisson-class ice breaker and arctic research vessel.

Imagery collected from a forward facing camera and the corresponding ensemble predictions are first presented in the section ‘Front View’. Next, imagery collected from the port side and the corresponding ensemble predictions are presented first in the section ‘Side View’. Reference Figure 18 for the colour code used to present ensemble predictions.

The output predictions are not all of high quality, but still overall shows great potential when considering that the models were trained on a very limited dataset from a different camera perspective which did not move. Portions of the vessel are frequently labelled correctly as part of the ‘other’ class despite the models never having been exposed to any similar examples. The issue of open-water being misclassified as level ice remains present and is most noticeable in the side view. Turbulence and the reflection of clouds may contribute to these misclassifications.

These results could be significantly improved if images from this domain were annotated, merged with the original dataset, and used to retrain the ensemble of models.

<SORRY, IMAGES REDACTED>