

Machine Learning Project

Arctic sea ice

Shelby Short || 4/25/2023

Table of Contents

[Introduction 2](#_Toc133422426)

[Background 2](#_Toc133422427)

[Dataset for Image Training and Testing 2](#_Toc133422428)

[Preprocessing in ArcPro 2](#_Toc133422429)

[set up 2](#_Toc133422430)

[Pond Digitization 4](#_Toc133422431)

[Rasterization 4](#_Toc133422432)

[Model Set Up in Colab 5](#_Toc133422433)

[Refining Mask 5](#_Toc133422434)

[Image Tile-ing 7](#_Toc133422435)

[Image Augmentation 8](#_Toc133422436)

# Introduction

## Background

An overall goal of this research was to use machine learning methods to generate new data to aid in the interpretation of arctic sea ice while interpolating between data unknowns.

To generate meaningful interpretations on arctic sea ice, a metric to assess and further evaluate arctic sea ice must be established. Melt ponds on sea ice surfaces serve as this metric because they are visually identifiable and can therefore be used to understand spatial and temporal trends. Therefore, a model to identify melt ponds will be the first machine learning task.

Machine learning methods used for this research involved the use of Convolutional Neural Networks(CNN) for image segmentation coupled with masks created from sea ice pond imagery in ArcPro. Additionally, Meta’s foundation model, SAM (Segment Anything Model) was used to improve model accuracy by generating more training data and associated masks. As SAM was released recently (04/05/2023) comparison and implementation with this model is relatively new.

## Dataset for Image Training and Testing

Choice of training and testing dataset was limited to 07/21 and 07/23 as these are the two days from the study that had significant data overlap. From here, 07/23 was selected as this date appeared to have clearer individual ponds and less cloud coverage. From here, Flight Line 0, section c (FL0\_SC) was selected as it had great overlap with FL2 as well as defined melt ponds. Specifically, images UF174\_UL006 to UF\_168\_UL0006 were used.

# Preprocessing in ArcPro

## set up

Arc Pro was used to evaluated sea ice images and generate masks for image segmentation. A mask for machine learning purposes can be thought of as a binary image in the form of an array or matrix that acts as a filter which allows certain values or features to pass through. Therefore, masks were used in this case to aid in the identification of melt pond features while simultaneously reducing the input that sea water and ice have.

To generate a mask, georeferenced hi-resolution (.tif) Chiroptera imagers were first added to arcPro and acted as a base for melt pond digitization which was later converted to a mask. To begin digitization, a feature class ‘ponds’ was created in ArcPro. Properties of the feature class are noted in table 1 and a segment of the feature class’s attributes are shown in figure 1.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Tolerance | XY Resolution | M Values | Z Values | Feature Class Type | Projected Coordinate System | Unit |
| 1mm(default) | .05 cm | yes | yes | Polygon | WGS 1984 UTM Zone 21N | Meters |

Table 1: Properties of ‘ponds’ Feature Class. Tolerance refers to the minimum distance between coordinates before they are considered equal. XY resolution reflects resolution of input imagery.

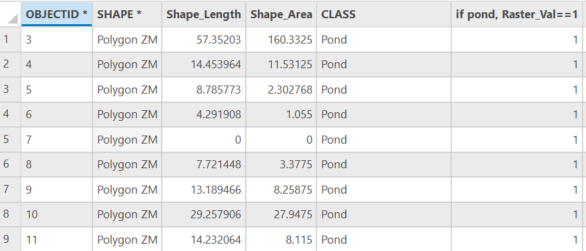


Figure 1: Screen Capture of ‘ponds’ attribute table. Shape length and Area are in units of m and m^2, respectively. All features are classified as ponds. Pond features have a value of 1 to be rasterized.

## Pond Digitization

The feature class that new features were added to for this project was the ‘ponds’ feature class. Ponds were digitized using the polygon tool. Default properties of the feature class include class set to ‘Pond’ and Raster\_Val set to ‘1’. To start, about 300 ponds were digitized.

## Rasterization

To create a mask, the ‘ponds’ feature class needed to be converted to a binary data set that spatially covered the entire input image. To do this, the feature class was exported as a shapefile using the ‘Export features’ tool in arcPro. The shapefile was then converted to a raster dataset using the ‘Polygon to Raster’ tool. The ‘Con’ tool was then used to set all pixels within pond polygons to a value of 1 and all pixels outside of the ponds, but within the extent of the original .tif file, to a value of 2. The extent of the binary raster did not exactly match with the original extent. To account for this, the raster and original image were both clipped to the same size. Additionally, the raster output was shifted using the ‘Shift’ tool in arcPro, which was necessary as there was a display issue between the full resolution(p100) image and 10% resolution(p10) image\*. While the ponds were digitized on the full resolution image, it was computationally too intensive to run a model on the full resolution image so the p10 image was used. From here, the raster and shifted p10 image were exported to be used in the machine learning process.

\*Still not sure why this issue came about, projections were the same between both full res and low res image, as well as the image extents.

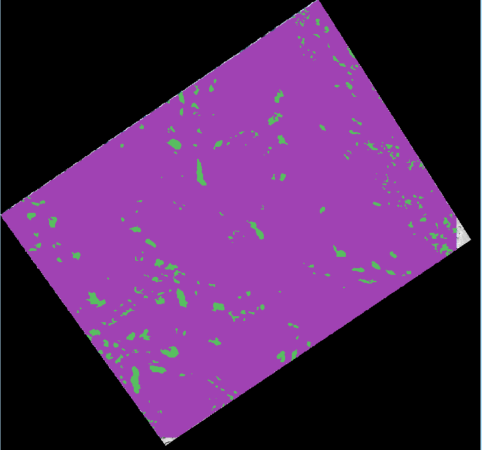


Figure 2: Screen shot of binary raster output in arcPro

# Model Set Up in Colab

Google Colab was used for this project as it; saves versions of your python code, can easily be shared and edited, has pre -installed libraries, has free GPU use, and is cloud based, making it well suited for machine learning projects.

The google colab file and dataset used for this model are available to read and download here:

## Refining Mask

Open CV was used to read the background image and the mask. The mask needed to be further edited to make it useable for our project. Since the mask was originally generated on the full resolution image, the mask shape was too large to be used with the p10 image. Therefore, the mask needed to be resized which was done using cv2.resize()

mask\_resize = cv2.resize(mask, None, fx=img.shape[0]/mask.shape[0], fy=img.shape[1]/mask.shape[1], interpolation = cv2.INTER\_NEAREST)

Figure 3: OpenCV python code that downscaled the mask to be equivalent with the original input image while maintaining the aspect ratio. Interpolation used was Nearest Neighbor method.

While the goal was to have the mask raster import into python as binary, the mask was padded with 0 and 255 values which can be though of as “null data” values. Therefore, the mask needed to be changed to a 3 channel grey mask which was accomplished by setting ‘null values’ to 0, ‘pond’ values to 255, and ‘ice’ values to 100.

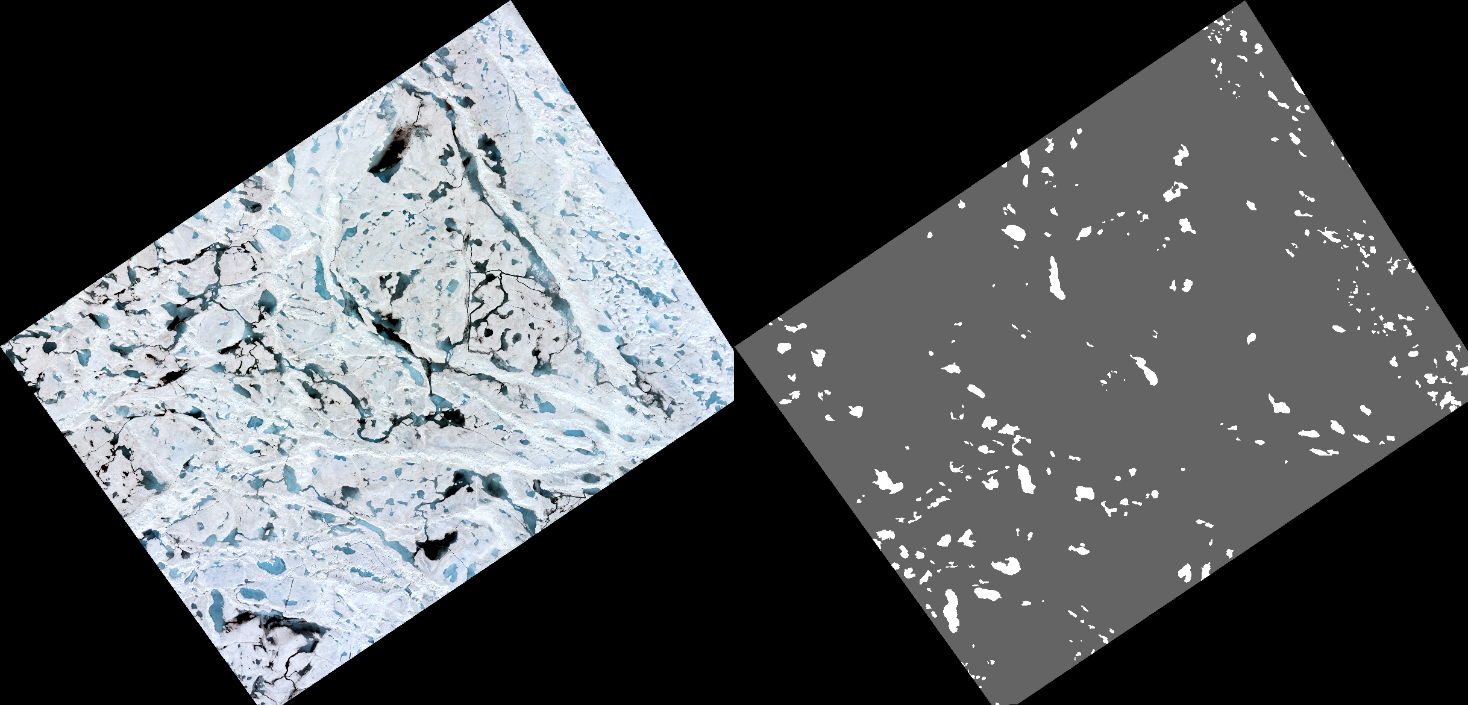


Figure4: 4 channel input image of sea ice on left. 3 channel grey mask on right.

At this point, we can blend the mask with the background image (figure 5).

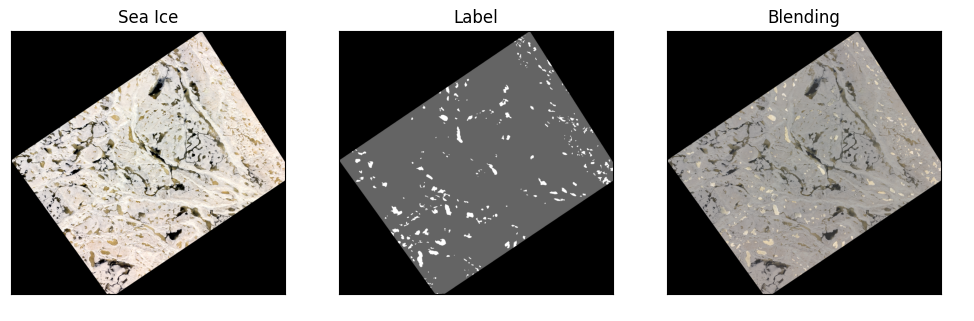


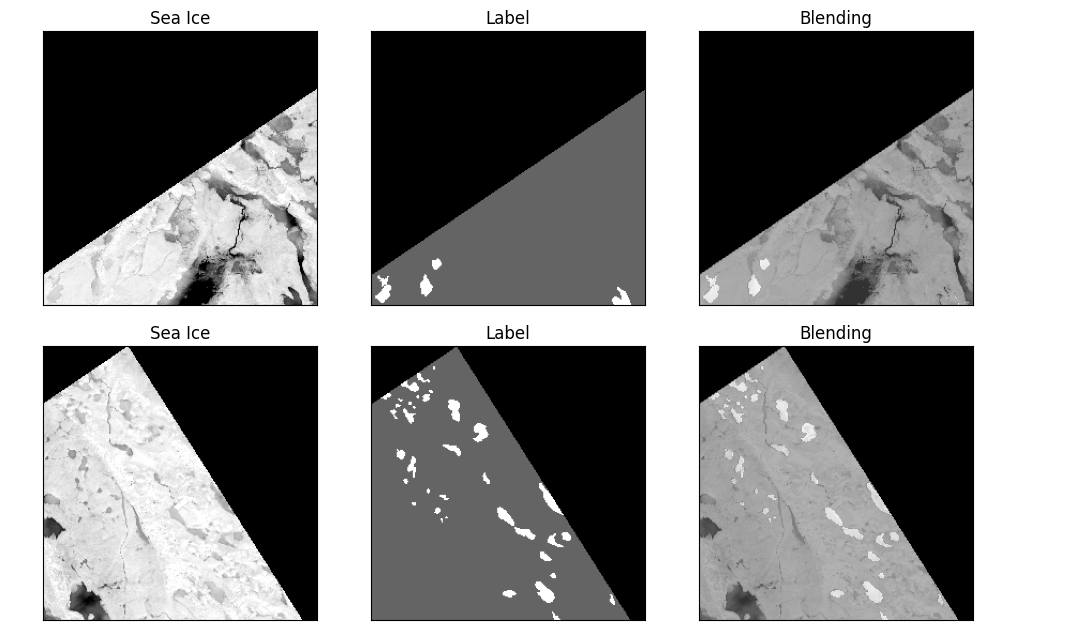
Figure 5: Sea Ice images showing the input (left), mask (middle), and blend(right).

## image Tile-ing

To increase the amount of training and testing data available, the mask(label) and sea ice input image were tiled.

## ask about the code for this, significance of 256

## why does it switch to gray?



## Image Augmentation

To artificially expand and enhance our dataset, image augmentation was used. All augmentation techniques were applied to all the tiles. Images were rotated, flipped, blurred, inverted, and equalized. The salt and pepper technique adds noise to the original images, while dropout reomves a certain percentage of the image to

We will use Segment Anything Model (SAM), an open-source segmentation tool, to identify melt ponds in the imagery. The automatic mode creates a segmentation mask for any visible object within an image and reduces duplicate images through mask post processing. Figure 31 illustrates the current work progress where the ponds are rasterized in ArcGIS Pro and a binary mask is created. SAM generated an output that requires minor modification and refining.