Enhancing Sea Ice Segmentation Model: Exploring Neural Network Ensembles and Optimization Strategies for Improved Performance

Bruno Henrique dos Santos Marques
Centro de Informática
UFPE
Recife, Brazil
bhsm@cin.ufpe.br

José Vinicius de Santana Souza
Centro de Informática
UFPE
Recife, Brazil
jvss2@cin.ufpe.br

Victor Gabriel de Carvalho
Centro de Informática
UFPE
Recife, Brazil
vgc3@cin.ufpe.br

***Abstract*—Polar ice, covering approximately 10% of Earth's surface and holding nearly 70% of freshwater, profoundly impacts climate and socioeconomic systems. Accurate sea ice segmentation using computer vision techniques is crucial for understanding climate change implications. This project focuses on enhancing a UNet-based segmentation model using a dataset of polar ice images. Various optimization techniques including hyperparameter tuning, ensemble methods like bagging, and pre-trained models such as ResNet18 were explored. Results show ResNet18 outperforming other models in Intersection over Union (IoU), Dice Coefficient, and Pixel Accuracy. Challenges including limited GPU access on Google Colab affected model performance, particularly for optimization methods. Despite challenges, ensemble models demonstrated promising results, reducing prediction variance. This study underscores the importance of robust segmentation techniques for monitoring polar ice dynamics in a changing climate.**

***Index Terms*—segmentation models, UNet, hyperparameter tuning, ensemble methods, model optimization, image processing, machine learning, deep learning**

I. Introduction

Polar ice covers about 10% of Earth's surface [1] and represents almost 70% of all freshwater [2]. Given this, it is undeniable that the polar regions play a pivotal role in the Earth's climate and socioeconomic system. Monitoring the dynamics of sea ice is essential for understanding temperature variations and their broader implications. Knowing precisely where ice chunks are and their dimensions can provide us with a clear oversight of the consequences of climate change and water dynamics worldwide.

A segmentation task, as the name suggests, is the process of dividing an image into different regions based on the characteristics of adjacent pixels. Consequently, ice segmentation consists of the intricate task of accurately delineating sea-ice boundaries using advanced computer vision techniques. Sea ice segmentation poses a unique set of challenges due to its complex and dynamic nature. Boundaries are often indistinguishable to the human eye.

This project unfolds the journey of modeling and enhancing a sea ice segmentation model using a data set of ice chunk images taken from space. Along with ranking the best approaches and results.

II. Objectives

Within this project, our objective is to construct a UNet-based image segmentation system [4] from the ground up. This system is designed to accurately segment ice chunks observed from space, with a particular focus on distinguishing subtle ice and water interfaces.

Moreover, we aspire to enhance the model's performance by exploring and assessing various optimization techniques in conjunction with ensemble methods. The assessment will employ established segmentation metrics, including Intersection over Union (IoU), Pixel Accuracy, and Dice Coefficient. Additionally, we will provide visualizations of predictions and ground truths to further evaluate and interpret the results.

III. Justification

Due to the important role played by sea ice in both navigation and environmental science questions,' it is utterly necessary to provide precise and reliable information about sea ice's location, size and extent. This information can help prevent future problems and provide valuable data for projects related to sea ice throughout the world.

We believe that our project can both make the climate change a more traceable problem and show the impact of optimization methods in a model's results.

IV. Dataset

This paper will rely on the Leeds SciML Sea Ice Segmentation Kaggle competition dataset [3], in which its main objective is to generate a sea ice segmented image.

A. Dataset Structure

a) Data: The dataset is composed entirely by ".tiff" images. Each data sample is separated into three different image files by its titles ending:

1) Image files with titles ending with "sar.tiff", as shown in Fig. 1, are single-band SAR (synthetic

aperture radar) images, providing useful information about the ice-water interface and its characteristics [3], representing the inputs for the models. It is worth saying that those images have low absolute valued pixels, that makes them hard to see as a standard RGB picture.

2) Image files with titles ending with "vis.tiff", as shown in Fig. 2, are three-band optical images that provide visual data that will also be used as inputs as complements for the "sar.tiff" images [3]. These images will as well be used in data analysis by the team.
3) Image files with titles ending with "ref.tiff", as shown in Fig. 3, are the ground truth label image files, providing the reference information about the dominant ice-water interface, and will be used to evaluate the accuracy of the trained models [3].

b) Train and Test datasets: The dataset is separated into two different folders:

1) "train" is the folder that contains the data used for training the models and, as said before, each data sample is separated into three image files ending with: "sar.tiff" (inputs for the model), "vis.tiff" (inputs for the model and data visualization) and "ref.tiff" (ground truth for evalauating the model).
2) "test" is the folder that contains the data used for evaluate the trained models, and each data is separated into two image files ending with: "sar.tiff" (inputs for the model) and "vis.tiff" (data visualization).

As stated by the Kaggle competition description, the data has limitations such as coarse resolution and cloud cover [3] that may become a challenge for the models to extract the right features from the data.

Fig. 1: Example of "sar.tiff" image

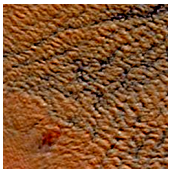


Fig. 2: Example of "vis.tiff" image

B. Exploratory Data Analysis

The dataset was loaded and converted into into a visible spectrum, Fig. 4 shows the contents of two batches from the training dataset. The training dataset has 4221 images, resulting in 1407 samples. Otherwise, the test dataset has 704 images, resulting in 352 samples, note that there is no reference image for these data.

Fig. 3: Example of "ref.tiff" image

As mentioned before, "sar.tiff" and "ref.tiff" images are single banded figures, while "vis.tiff" is a regular 3 channel satelite picture. "vis.tiff" and "ref.tiff" are both 240x240 pixels, while "sar.tiff" is 720x720 pixels.

It is noticeable how both the "sar.tiff" image and "vis.tiff" image are complementary when deciding the best segmentation. Figs 4(b) and 4(c), for example, have a cloudy "vis.tiff" picture, needing the model to rely mostly on "sar.tiff". However, Figs 4(a) and 4(d) show cases where "vis.tiff" provide a grater similarity with "ref.tiff".

Following this thought, our proposed data loader provided both "sar.tiff" and "vis.tiff" images stacked together as a four channel image.

V. Models

In order to learn from data, a UNet [4] model was used. It is one of the most popular approaches for image segmentation, combining the efficiency of convolutional neural networks and residual links. The encoder pathway captures features at multiple scales, while the decoder pathway reconstructs the segmented output with remarkable spatial resolution. Fig. 5 shows a simplified architecture for UNet, it is clear to see some main characteristics like the skip connections and its U shape.

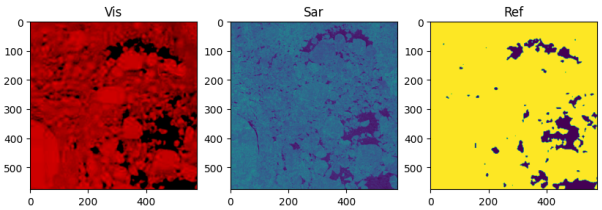
Along with our handmade UNet, we will also evaluate a pre-trained approach (consisting of a ResNet encoders trained on ImageNet dataset) and a series of optimizations and ensembles, as described bellow.

A. Optimization Techniques

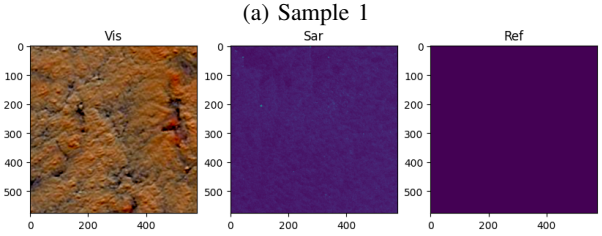
To obtain the best possible results from the UNet [4] model, a series of optimization techniques were used. We will focus on using different types of ensemble methods, and also, search for the best hyperparameters combination automatically with an hyperparameter optimization framework.

It is worth mentioning that these are state-of-the-art or already consolidated techniques, which contributed to our choice.

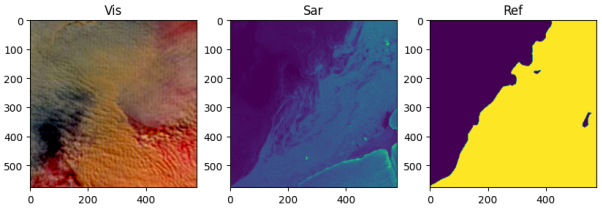
a) Hyperparameter Tuning: An important step in obtaining the best possible model, is the hyperparameter tuning phase, in which we must search for the best possible hyperparameter combination so that the model can perform with higher efficiency maintaining the best possible results. For an automatic hyperparameter tuning, the Optuna Framework [6]



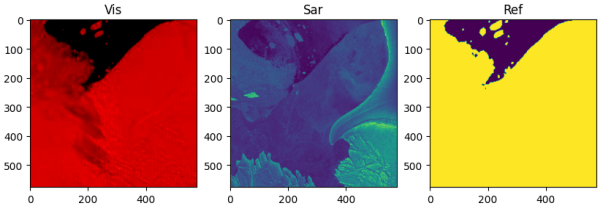
(a) Sample 1



(b) Sample 2



(c) Sample 3



(d) Sample 4

Fig. 4: Two random batches.

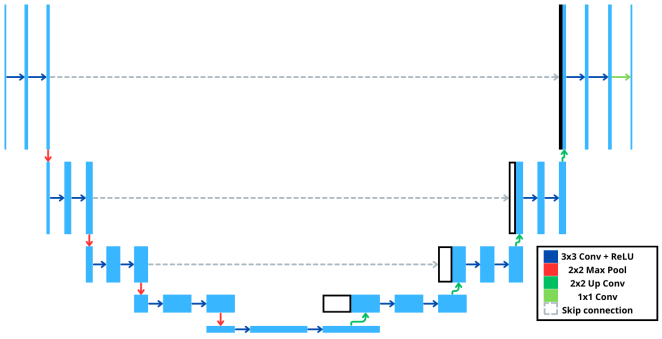


Fig. 5: Simplified diagram of UNet.

that employs a series of state of art optimization algorithms was used.

b) Bagging [5]: Is the ensemble learning method that consists in training multiple instances of the same learning algorithm on different subsets of the training data. At the end, all outputs are averaged or voted for a final result. This technique states that the combination of diverse models helps reduce variance and improve overall performance. Figure 6 shows a simplified Bagging workflow.

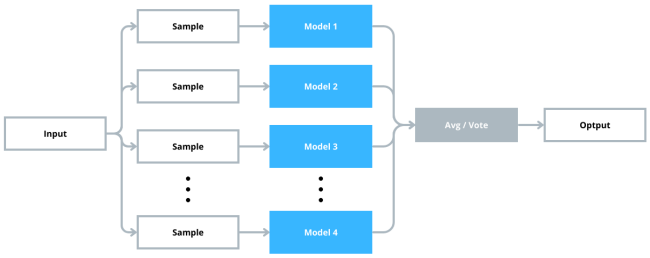


Fig. 6: Simplified diagram of Bagging technique.

B. Evaluation Metrics

To evaluate and compare our models performance, three of the most popular metrics were chosen.

a) Intersection over Union (IoU): One of the most used image segmentation metrics. It consists in quantifying how well the model can distinguish objects from their backgrounds in an image. It consists in determining how closely two areas (in our case, masks) overlap and can be defined as follows:

$$IoU = \frac{|A \cap B|}{|A| \cup |B|}$$

Note that A and B represent the conjunct of points on each related mask.

b) Dice Coefficient: It is a similarity metric that acts between two sets. In our case, it will check the similarity between our reference and our predicted image. Dice Coefficient is calculated from the precision and recall of a prediction, essentially being their harmonic mean.

$$DC = \frac{2 \times |A \cap B|}{|A| + |B|}$$

The main difference between DC and IoU is that Dice can be more robust to class imbalance. While IoU penalizes more under and over-segmentation.

c) Pixel Accuracy: Measures the overall accuracy of pixel-wise predictions. It is a really simple and straightforward approach, it is given by:

$$PA = \frac{Number\ of\ correctly\ classified\ pixels}{Total\ number\ of\ pixels}$$

d) Visualization: Model's outputs will also be visually analysed in comparison to its inputs.

C. Coding Environment

During our implementation and evaluation, we will both Google Colab and local machines. Colab is an online Python coding platform that provides us T4 GPUs, essential for training convolutional neural networks more rapidly. Additionally, Colab has the benefit of working remotely and without the need of a local powerful computer, code can also be shared and version tracked between our team.

However, Colab's GPU access is limited. So, we decided to make our final code run also locally on a GTX 970 graphics card.

VI. Experiments

The code was designed to use less than 4GB of VRAM. The images were set to a resolution of 384x384, make computations easier while keeping a good size for convolutions in series. The training dataset was divided into train and validation sets, splitting the data into 20% for validation and 80% for training. Due to the lack of reference images on the test set (Kaggle competitions do not provide those on test sets), all evaluation metrics performed on the validation set, the test was used only for results visualization.

It is worth mentioning that, due to processing limitations on Colab, our code was split into four notebooks, each one running a different optimization on the same dataset and using the same random seeds, to ensure consistence.

All stand alone models were trained for a maximum of 20 epochs and a patience criteria of 3 (that usually stopped the model in advance). The ensemble models were trained for a maximum of 10 epochs, with the same patience criteria.

First, we trained a regular and handmade UNet model, to serve as our base model for comparison with the optimizations. Followed by a a series of pre-trained UNets with ResNet encoders (ResNet50, ResNet34 and ResNet18).

A hyperparameter optimization framework, Optuna [6], was employed to enhance the efficiency of the manually crafted UNet model. Utilizing Optuna, we generated studies that enable us to suggest variable values, and subsequently employed state-of-the-art sampling techniques to identify the most valuable samples for enhancing model performance in each study trial. The suggestions were:

- Features list: manages the size of each UNet block convolutional layer, so we variate the number of features, as well as the size of each features expecting that we could reduce the training time as well as improving the results.
- Optimizer: the parameter optimization algorithm. We suggested 4 algorithms with different approaches: Adam, Adadelta, AdamW, and RMSprop.
- Learning rate: that adjusts the size of the steps that the parameter optimization algorithm takes at each update. We set an common range between 0.00001 and 0.1 with an log distribution.

For each ensemble technique, we used an approach with three classifiers (instantiated from our base UNet class), trained on random portions of the dataset.

VII. Analysis of Results

The resulting metrics for each implementation approach are shown in Table I. It is possible to see that the overall performance was acceptable, ranging from 0.6 to 1.0 in all evaluations. The highest results came from ResNet18 encoder model, followed by our Base Model and the Bagging ensemble. Figure 7 shows an output example of ResNet18.

We expected for the Optuna models to increase the Base model performance, but due to the time access limitations from the Free Google Colab GPUs, it may caused an impact at the models performances leaving their IoU scores the lowest in comparison to the others.

TABLE I: Models Results

Model	IoU ↑	Dice ↑	PA ↑	Val Loss ↓	Time ↓
Base	0.733	0.807	0.807	0.341	22
Base Optuna	0.662	0.746	0.892	0.310	16
Base Optuna 2	0.627	0.741	0.764	0.310	36
ResNet18	**0.772**	**0.832**	0.878	**0.255**	**9**
ResNet34	0.688	0.757	0.737	0.509	13
ResNet50	0.706	0.780	0.765	0.365	17
Bagging	0.708	0.756	**0.921**	-	81

Note: Times are expressed in minutes.

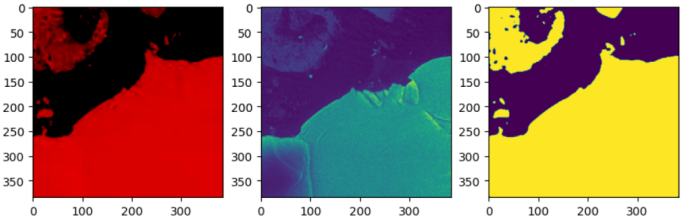


Fig. 7: Fine tuned ResNet18 output for a new input.

VIII. Conclusions and Discussions

We believe that, due to the small dataset and the small resolution images, approaches like ResNet50 and 30 tended to overfit on training data or degrade the images after many convolutions, making simpler models like ResNet18 and Base UNet output the best results.

The limited access time of the Google Colab GPUs were a problem, specially for the ensemble models and the Optuna optimizations. Because at every run, the access were removed, and the Colab stopped, leaving the models training unfinished. This can be one of the reasons why Optuna had a bad performance in comparison to the others. We placed 10 trials for the first Optuna study, and 12 trials for the second Optuna study, but the first study was forcefully stopped at the 4th trial and the second study at the 2nd trial. Since it uses the history of trials scores to decide what hyperparameters to change and how they will change, the first batch of trials usually are a baseline for the study sampler to choose the best hyperparameter modifications.

The ensemble model also had a good performance, probably form the fact that it could leverage the simplicity of the base

models and reduce the prediction variance. A down side is that it took much more time to finish its training and also takes more time to predict an output.

REFERENCES

[1] NASA, "Global Ice Viewer" climate.nasa.gov. https://climate.nasa.gov/interactives/global-ice-viewer/#/ (accessed Dec. 15, 2023)

[2] T. Bralower, D. Bice, "Distribution of Water on the Earth's Surface," e-education.psu.edu. https://www.e-education.psu.edu/earth103/node/701 (accessed Dec. 15, 2023)

[3] Eilish O'Grady, Naomi Shakespeare-Rees, spiruel. (2023). "Leeds SciML Sea Ice Segmentation," Kaggle. https://kaggle.com/competitions/leeds-sciml-sea-ice-segmentation (accessed Dec. 18, 2023)

[4] Ronneberger, O., Fischer, P., Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. In: Navab, N., Hornegger, J., Wells, W., Frangi, A. (eds) Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015. MICCAI 2015. Lecture Notes in Computer Science(), vol 9351. Springer, Cham. https://doi.org/10.1007/978-3-319-24574-4_28

[5] Breiman, L. Bagging predictors. Mach Learn 24, 123–140 (1996). https://doi.org/10.1007/BF00058655

[6] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A Next-generation Hyperparameter Optimization Framework. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '19). Association for Computing Machinery, New York, NY, USA, 2623–2631. https://doi.org/10.1145/3292500.3330701