Diving into Supraglacial Lakes Detection: a Deep Semantic Segmentation Approach

Simon Walther
simon.walther@heig-vd.ch
University of Applied Sciences
Western Switzerland
(HEIG-VD/HES-SO)

Leonard Cseres
leonard.cseres@heig-vd.ch
University of Applied Sciences
Western Switzerland
(HEIG-VD/HES-SO)

Rémy Marquis
remy.marquis@heig-vd.ch
University of Applied Sciences
Western Switzerland
(HEIG-VD/HES-SO)

Bertil Chapuis
bertil.chapuis@heig-vd.ch
University of Applied Sciences
Western Switzerland
(HEIG-VD/HES-SO)

Andres Perez-Uribe
andres.perez-uribe@heig-vd.ch
University of Applied Sciences
Western Switzerland
(HEIG-VD/HES-SO)

ABSTRACT

Rising summer temperatures in Greenland have accelerated the formation of supraglacial lakes. Since these lakes play a significant role in ice sheet dynamics and bed lubrication, their continuous monitoring in a warming Arctic is becoming essential. The 31st ACM SIGSPATIAL competition (GISCUP 2023) aims to automate the detection of these lakes using satellite imagery. In this paper, we present two solutions to this problem based on image segmentation techniques: a DeepLabv3+ model that ranked first, and a U-Net-based approach that ranked fourth. We provide details about our implementations and explain the rationale behind our choices and the challenges we faced. Our results contribute to the understanding of supraglacial lake fluctuations and offer a valuable tool for ongoing environmental monitoring.

CCS CONCEPTS

• **Information systems** → **Geographic information systems**; • **Computing methodologies** → **Image processing**; **Supervised learning**.

KEYWORDS

Satellite Imagery, Computer Vision, Segmentation, Detection

ACM Reference Format:
Simon Walther, Leonard Cseres, Rémy Marquis, Bertil Chapuis, and Andres Perez-Uribe. 2023. Diving into Supraglacial Lakes Detection: a Deep Semantic Segmentation Approach. In *The 31st ACM International Conference on Advances in Geographic Information Systems (SIGSPATIAL '23), November 13–16, 2023, Hamburg, Germany*. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3589132.3629970

SIGSPATIAL '23, November 13–16, 2023, Hamburg, Germany

ACM ISBN 979-8-4007-0168-9/23/11.
https://doi.org/10.1145/3589132.3629970

1 INTRODUCTION

Over the last two decades, rising boreal summer temperatures in Greenland have accelerated the formation of high-elevation supraglacial lakes atop the ice sheet. These lakes, which rapidly grow and drain through ice cracks, significantly affect bed lubrication and ice sheet dynamics. Ongoing research is focused on understanding the influence of this phenomenon in a warming Arctic. This necessitates effective solutions for continuously monitoring these supraglacial lakes. The 31st ACM SIGSPATIAL competition (GISCUP 2023[1]) addresses the problem of automatically detecting these lakes from satellite images and tracking their existence and behavior longitudinally throughout the seasons. This paper introduces two solutions based on segmentation techniques that our team devised to address this problem. The DeepLabv3+[2] based algorithm achieved first position. The U-Net[11] based algorithm ranked fourth. In the following sections, we first describe our methodology. We then present our solutions, with a focus on the common processing stages and on the winning algorithm. Finally, we discuss our results, the challenges encountered during the competition, and explore possible avenues for future work.

The dataset provided for the challenge comprises four multi-part satellite images in the GeoTIFF format, captured during the summer 2019 melt season at different intervals. These images depict two distinct areas of Greenland, each showcasing numerous surface lakes and hydrologic features. The areas are further subdivided into six smaller regions within each image, alternately designated as test and train regions. The corresponding lake labels are represented as polygons in a geopackage file. Only the training labels were provided, while the test labels were withheld for the final evaluation and ranking.

2 METHODOLOGY

We formed a multi-disciplinary team of data scientists and software engineers. The adoption of MLOps practices and tools has significantly streamlined our model development workflow. We utilized Git for code collaboration and for sharing common preprocessing and postprocessing steps. Additionally, we integrated DVC to version our data. This enabled faster iterative experimentation through

[1] https://sigspatial2023.sigspatial.org/giscup/

cached pipeline steps, overnight automated hyperparameter tuning, and allowed for systematic evaluations and comparisons.

In our development process, we promoted collaboration and a healthy level of internal competition. We agreed to share code while also exploring multiple solutions in parallel to prevent stagnation and ensure continuous improvement. Regular meetings focused on communicating intermediate results fostered a cooperative environment and facilitated knowledge exchange. A scoreboard, updated regularly, showcased the performance of the different solutions and served as a motivating factor for achieving high levels of performance. While our initial goal was to select the best ML model for the final submission, two models excelled on the validation data, so we decided to submit both. The final solutions use a common framework, derived from insights and techniques uncovered during this process.

In addition to our internal meetings, we presented our work to two industry experts in GIS and ML, and invited them to challenge our solutions. This productive session refined our methodologies, clarified some of our ideas, and deepened our understanding of potential pitfalls.

3 SOLUTIONS OVERVIEW

This challenge focuses on generating polygons of lakes from satellite imagery. To address it, we performed semantic segmentation. We predicted binary masks, distinguishing between "non-lake" and "lake" pixels from RGB images. This process ultimately enabled us to delineate polygons based on these binary masks. In this section, we briefly introduce the semantic segmentation approach common to both of our models. In Section 4 and Section 5, we elaborate on both of our implementations relying on distinct ML algorithms.

3.1 Preprocessing

Creating a relevant training dataset is crucial to ensure that the model learns from information-rich images without introducing excessive bias towards one class or the other. Key factors in this process include the acquisition of a substantial number of images and the removal of superfluous ones.

Tiling. To increase the dataset size and obtain multiple views of the lakes, we extracted overlapping (50%) tiles across all training regions. We initially extracted these tiles at a size of 448x448 and then downsampled them to 320x320 (see Figure 1). Using a larger tile size allowed us to capture a broader context, while downsampling reduced the model's computation time and eliminated irrelevant details. This resulted in around 60, 000 tiles with their corresponding masks.

Dataset filtering. It is important to concentrate on tiles that provide essential data to the model. To accomplish this, we filtered out tiles with an excessive amount of missing data, such as tiles that were partly extending beyond the boundaries of the raster. We achieved this by examining their percentage of black pixels and setting a threshold at 60%. While we kept all tiles containing lakes in the dataset, we carefully selected non-lake tiles. Many of these tiles were predominantly white with minimal variations, offering limited information. Moreover, this could have introduced bias towards the non-lake class due to its disproportionate representation compared to the other class. To mitigate potential diversity concerns arising from the random selection of non-lake images, especially when many of them exhibit visual similarities, we filtered them out. We eventually obtained a dataset of about 15, 000 tiles with their corresponding masks.

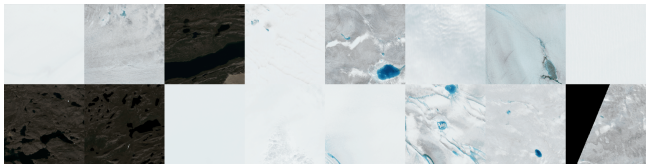

Figure 1: Tiles sample
Examples of tiles, with the last one partially extracted at the raster border.

3.2 Model

To generate segmentation masks from the image tiles, we used 320x320 RGB image tiles as input for our models to generate binary masks representing the two target classes. For model training and evaluation, we split the dataset into 80% for training and 20% for validation. We trained the final model for submission using the entire dataset. To help the model generalize, we used the Albumentations[1] library for image augmentation with different kinds of augmentations (rotation, flip, brightness, distortion, ...).

3.3 Postprocessing

To transform the model's predictions into lake polygons, we implemented a two-step postprocessing methodology. First, we generated predictions by tiling the original raster images with overlap. Subsequently, we aggregated the predicted masks and constructed polygons by identifying their contours. In the second step, we merged overlapping polygons and filtered them out based on their area.

4 DEEPLABV3+ SOLUTION

In this section, we present our DeepLabv3+[2] architecture, which stands as one of the state-of-the-art models for semantic segmentation [9]. This solution ranked first in the competition.

4.1 Preprocessing

Clustered-based undersampling. We applied a clustering algorithm and used the resulting clusters, with a probability inversely proportional to their size, to randomly select non-lake tiles. This method aims to enhance our chances of selecting tiles that contain important but less common land covers. The clusters are created using the k-means algorithm, with k set to 4 on the histograms of images' HSV color space. We chose a lower number of bins (specifically, 5 bins per channel) to reduce computation time. The decision for the value of k was driven by the intuition that a high k would lead to numerous clusters, potentially decreasing the method's efficiency in representing the rarity of land covers. Conversely, a k that is too low might capture less disparity between images and leave fewer chances to distinguish the more uncommon images.

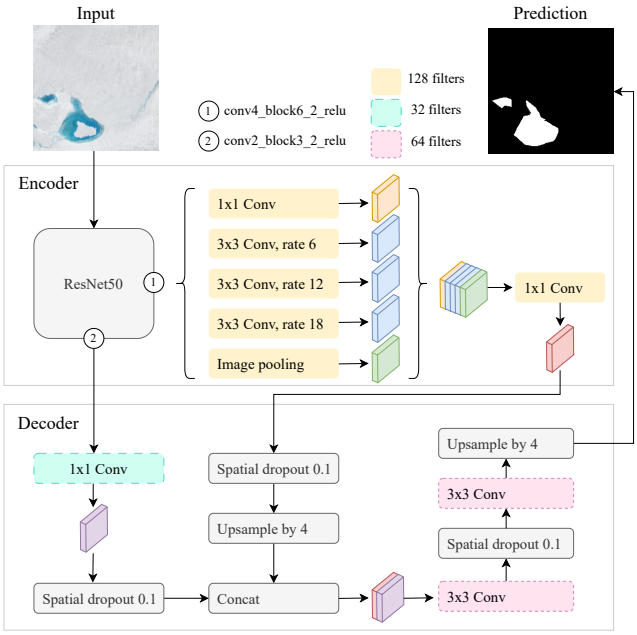

Figure 2: Modified DeepLabv3+ architecture
Changes include reduced filters and additional spatial dropout layers.

4.2 Model

Model architecture. We used the DeepLabv3+ example provided on the Keras website[10] as a basis for our solution and subsequently introduced several improvements to enhance the model's capabilities (see Figure 2). To address concerns related to overfitting and computational efficiency, we chose to reduce the number of filters. This resulted in a model having approximately 20% fewer parameters, totalling 9.5M trainable parameters, as opposed to 11.8M trainable parameters in the original implementation. Additionally, for model regularization, we incorporated three spatial dropout[12] layers. In an effort to reduce the model's sensitivity to batch size variations, we replaced batch normalization with group normalization[14]. Finally, we substituted the conventional ReLU activations with GELU activations[4], with the expectation that this change would further enhance model performance.

Training details. The loss function used is a combination of a weighted Binary Cross Entropy (BCE) and Dice loss. By combining both, we benefit from the advantages of Dice loss, notably robustness to class imbalance, while BCE mitigates its drawbacks by smoothing the loss landscape[6].

Snapshot ensembles. To achieve better performance and ensure a more robust solution, we chose to employ an ensemble of models in this approach. To reduce training time, we created this ensemble following the snapshot ensembles technique[5]. We trained our models using stochastic gradient descent with the following hyperparameters: a momentum of 0.99 and a clipnorm of 1. Additionally, we used a cosine annealing learning rate scheduler[8] with an initial learning rate set to 0.1. We determined this initial learning rate through a learning rate range test. The training process is organized into three cycles, each spanning 50 epochs. At the completion of each cycle, we saved the model weights (see Figure 3). This approach produced three distinct models that constitute the ensemble.

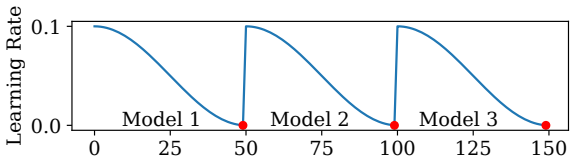

Figure 3: Learning rate scheduler for snapshot ensembling
Weight saving times are marked as red dots.

4.3 Postprocessing

Model ensemble. To diminish the impact of lake positions in images, which may lead to errors (for instance, when there is a small portion of lakes at the image's borders), and to enhance the robustness of our solution, we made predictions with a three-fold overlap. Since we used three models for predictions, we averaged nine predictions in total, except at the raster's borders where we had a minimum of three predictions.

5 U-NET SOLUTION

In this section, we present our U-Net model. This solution ranked fourth in the competition.

5.1 Preprocessing

pHash similarity based filtering. We used a perceptual hashing algorithm [7] to calculate the hashes of the generated tiles in the preprocessing step. For each image and region, we included all the tiles containing lakes and then selected the ones that did not contain lakes and did not have the same hash. This process ensures that we maintain a consistent distribution of images throughout the images and regions while eliminating similar images.

Dataset bias. Each tile is labelled as either containing a lake or not containing a lake. To create the dataset, we picked a ratio of 0.7 "lake" to "non-lake" images. As more images were available for training, this introduced a bias in the dataset, predicting fewer lakes, but enhancing the model's robustness to false positives.

5.2 Model

Model architecture. We used the U-Net[11] encoder-decoder architecture with a pre-trained ResNet50[3] encoder for predicting the masks. The weights were left trainable with a total of 32.6M parameters. We started with an implementation from NAU-Net[13] using the Pytorch Lightning framework.

Training details. We used the Adam optimizer and a custom loss function that combines, in a weighted sum, a Tversky loss ($w_{Tversky} = 0.25$) and a custom blob detection loss ($w_{blob} = 0.75$) to train the network. The Tversky loss ($alpha = 0.6$, $beta = 0.4$) is used to ensure that the model is accurate pixel-wise. The blob detection loss is used with the idea of improving the accuracy of

the number of predicted lakes. The combination of the two losses allows for more stable training.

We trained the network for 140 epochs with a batch size of 32. Through a learning rate range test, we set the learning rate to 0.00015 and multiplied it by a factor of 0.8 every 20 epochs. We incorporate the *StepLR* learning rate scheduler from PyTorch to enhance the model's convergence.

5.3 Postprocessing

Edge significance mask. To enhance the quality of the predictions near tile edges (where lakes may be truncated), we multiplied each one of them with a mask designed to diminish edge significance. We then averaged the predictions over a 50% overlap. The mask consisted in a 25% border around all edges, set at a value of 0.75, while maintaining a value of 1.0 for the inner area. The border dimensions and value were determined based on the 50% tile overlap. This ensures that overlapping tile edges have a minimum of three confident predictions for annotating border pixels.

6 RESULTS

The final evaluation of the submitted geopackage files for lake identification was based on F_1-score, which considers precision and recall. Each submitted polygon was to be classified as "true positive", "false positive", or "false negative" in comparison to the withheld test dataset. As defined by the competition rules, the criteria for a "true positive" includes: no overlap with other polygons in the submitted file, partial or full overlap with a polygon in the test dataset (the predicated area should be between 50% and 160% of the overlapping test area).

The DeepLabv3+ based model ranked first with a F_1-score of 0.712, whereas the U-Net based model ranked fourth with a F_1-score of 0.673. Using our Linux server (112 cores, 128GB RAM and four NVIDIA A40 graphic cards), computation completes in approximately 8 hours and 4 hours, respectively.

7 DISCUSSION

We encountered some noteworthy difficulties with ensuring reliable validation being a particular challenge. Since we employed a random split for validation, there was a possibility of encountering images and their overlaps in both the training and validation datasets. Additionally, it was possible that images taken at the same location but at different times could appear in both the training and validation sets.

To mitigate these issues, we implemented image augmentation. Nevertheless, we found that the metrics we used were not entirely reliable. We noticed that the training and validation Intersection over Union (IoU) were still consistently increasing without any sign of overfitting. It was only when we thoroughly examined the predictions in QGIS that we realized overfitting could still be an issue. We acknowledge that room for improvement exists, particularly in enhancing the model evaluation.

As future endeavour, we could explore the possibilities of combining the various preprocessing and postprocessing methods with the algorithms. This entails experimenting with different parameters settings. Also, we were unable to dedicate enough time to fully experiment with additional data sources, such as the near infrared channel to implement the Normalized Difference Water Index (NDWI), which has the potential to further enhance lake detection performance. However, it is worth mentioning that our experimentation with gradients computed from ArticDEM as an additional channel did not result in performance improvements. One possible reason is that the DEM model represents the mean average of each image.

These directions are however promising for enhancing the overall performance of our methods.

8 GUIDELINES TO RUN THE CODE

The code of both implementations can be found on GitHub[2] [3] under the MIT license. Refer to the related *README* files to set up the required dependencies, pull data from the remote storage, and run the ML pipeline.

ACKNOWLEDGMENTS

We would like to thank Julien Rebetez from *Picterra* and Nils Hamel from *arx iT* for their feedback and suggestions. This material is based upon work supported by the University of Applied Sciences Western Switzerland (HES-SO) and the Swiss AI Center.

REFERENCES

[1] Alexander Buslaev, Vladimir I. Iglovikov, Eugene Khvedchenya, Alex Parinov, Mikhail Druzhinin, and Alexandr A. Kalinin. 2020. Albumentations: Fast and Flexible Image Augmentations. *Information* 11, 2 (2020). https://doi.org/10.3390/info11020125

[2] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. 2018. Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation. arXiv:1802.02611 [cs.CV]

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. https://doi.org/10.1109/CVPR.2016.90

[4] Dan Hendrycks and Kevin Gimpel. 2023. Gaussian Error Linear Units (GELUs). arXiv:1606.08415 [cs.LG]

[5] Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E. Hopcroft, and Kilian Q. Weinberger. 2017. Snapshot Ensembles: Train 1, get M for free. arXiv:1704.00109 [cs.LG]

[6] Shruti Jadon. 2020. A survey of loss functions for semantic segmentation. In *2020 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*. IEEE. https://doi.org/10.1109/cibcb48159.2020.9277638

[7] Tanuj Jain, Christopher Lennan, Zubin John, and Dat Tran. 2019. Imagededup. https://github.com/idealo/imagededup.

[8] Ilya Loshchilov and Frank Hutter. 2016. SGDR: Stochastic Gradient Descent with Restarts. *CoRR* abs/1608.03983 (2016). arXiv:1608.03983 http://arxiv.org/abs/1608.03983

[9] Yujian Mo, Yan Wu, Xinneng Yang, Feilin Liu, and Yujun Liao. 2022. Review the state-of-the-art technologies of semantic segmentation based on deep learning. *Neurocomputing* 493 (2022), 626–646. https://doi.org/10.1016/j.neucom.2022.01.005

[10] Soumik Rakshit. 2023. Multiclass semantic segmentation using DeepLabV3+. https://keras.io/examples/vision/deeplabv3_plus/

[11] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. https://doi.org/10.1007/978-3-319-24574-4_28

[12] Jonathan Tompson, Ross Goroshin, Arjun Jain, Yann LeCun, and Christoph Bregler. 2015. Efficient object localization using Convolutional Networks. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 648–656. https://doi.org/10.1109/CVPR.2015.7298664

[13] Jinxiao Wang, Fang Chen, Meimei Zhang, and Bo Yu. 2022. NAU-Net: A New Deep Learning Framework in Glacial Lake Detection. *IEEE Geoscience and Remote Sensing Letters* 19 (2022), 2000905.

[14] Yuxin Wu and Kaiming He. 2018. Group Normalization. arXiv:1803.08494 [cs.CV]

[2] https://github.com/swiss-ai-center/giscup2023-deepLabV3Plus

[3] https://github.com/swiss-ai-center/giscup2023-resnet-unet