

U-Net Based Water Body Segmentation from Sentinel-2 Satellite Imagery for On-Board CubeSat Deployment

Sunny Pandey¹ and Hansika Saraf²

^{1,2} *Department of Computer Engineering and Technology, Dr. Vishwanath Karad MIT World Peace University, Pune, India*

Abstract—Detecting water bodies from satellite imagery matters for environmental monitoring, flood response, and managing natural resources. In this work, we build a full deep learning pipeline that performs binary semantic segmentation of water versus land in Sentinel-2 multispectral images, relying on the well-known U-Net architecture. We assembled a dataset of 181 images drawn from 200 locations scattered across nine geographic regions and three types of water bodies—coastal, riverine, and lacustrine—and generated ground truth masks automatically using a dual-index strategy that combines NDWI and MNDWI. After applying percentile-based normalization and splitting the data with geographic stratification (to avoid leakage), we trained the U-Net with a combined Binary Cross-Entropy and Dice loss. On the held-out test set the model reached 89.59% pixel accuracy, 75.39% mean IoU, and 85.44% mean Dice. For the water class, precision and recall are well balanced at 79.17% and 76.23% respectively (water IoU 63.50%). Performance varies sharply with setting: the model exceeds 86% water IoU on European and Middle-Eastern scenes—including regions held out from training entirely—but narrow river channels remain the dominant source of error. We also developed an interactive desktop tool called Amphitrite for real-time inference with adjustable thresholding. Looking ahead, we are preparing a lightweight version of this model for deployment on an edge AI module aboard a CubeSat, which would allow autonomous water body monitoring directly from orbit.

Index Terms—semantic segmentation, U-Net, water body detection, Sentinel-2, remote sensing, deep learning, CubeSat, edge AI

I. INTRODUCTION

About 71% of the Earth is covered by water, and shifts in where that water is—whether from seasonal

floods, prolonged drought, shrinking reservoirs, or long-term climate trends—have direct consequences for ecosystems, agriculture, and cities [1]. The only realistic way to track these changes at a global scale, with any reasonable revisit frequency, is through satellite remote sensing.

For decades, the go-to approach for picking out water in satellite images has been spectral indices. The Normalized Difference Water Index (NDWI) proposed by McFeeters [2] and its modified cousin MNDWI by Xu [3] both take advantage of how strongly water absorbs near-infrared and shortwave infrared light. They work well enough in clear-cut cases. But anyone who has spent time with these indices knows their pain points: shadows get flagged as water, dark rooftops in cities confuse NDWI, turbid rivers sometimes slip through, and picking the right threshold for one region often means it fails in another [4].

Deep learning—specifically, convolutional neural networks built for semantic segmentation—offers a way around many of these issues. Among the architectures that have gained traction in remote sensing, U-Net [5] stands out. It was originally designed for biomedical images where training data is scarce and spatial precision is critical, two properties that transfer neatly to satellite imagery. Its encoder-decoder layout with skip connections lets the network combine big-picture context with fine-grained spatial detail, and several research groups have shown it works well for water mapping in Landsat and Sentinel-2 data [6]–[8].

That said, a lot of the existing deep learning work on water segmentation has a common blind spot: the training data tends to come from a narrow geographic slice. A model trained on European river scenes may not generalize to tropical deltas or Arctic coastlines.

On top of that, most published studies stop at the model evaluation stage—few provide a complete, reproducible pipeline from data collection all the way to deployment, and fewer still think about what happens when you want to run the model on hardware with serious constraints, like a CubeSat in orbit.

We set out to address both of these gaps. This paper describes an end-to-end system that starts with data acquisition from Google Earth Engine, moves through preprocessing and training, and ends with an interactive desktop application for inference. The main things we contribute are:

- A geographically diverse dataset spanning 200 locations on all inhabited continents, covering coastal, riverine, and lacustrine water bodies. Ground truth masks were generated automatically using a dual NDWI/MNDWI approach, yielding 181 usable image-mask pairs.
- A U-Net model trained with combined BCE and Dice loss on this data, with thorough evaluation on a geographically stratified test set so that the results reflect genuine generalization rather than spatial memorization.
- Amphitrite, a desktop GUI application that lets users drag in satellite images and adjust the detection threshold in real time—no coding required.
- A roadmap for deploying an optimized version of this model on edge AI hardware aboard a CubeSat, which is currently under development.

The rest of the paper is laid out as follows. Section II covers prior work. Section III walks through our methodology in detail. Section IV reports the experimental results. Section V describes the Amphitrite interface and our edge deployment plans. Section VI wraps up with conclusions and future directions.

II. RELATED WORK

A. Index-Based Water Detection

Spectral water indices have been around since the mid-1990s and remain the workhorse of operational water mapping. McFeeters [2] introduced NDWI, exploiting the fact that water reflects green light but absorbs NIR strongly. Xu [3] later swapped NIR for

SWIR to create MNDWI, which does a better job in built-up areas where concrete and asphalt can mimic water's NIR response. Feyisa et al. [4] went a step further with AWEI, a more complex multi-band formula designed to deal with shadow artifacts. These methods are fast, interpretable, and still power products like the JRC Global Surface Water dataset [1]. Their weakness is brittleness: a threshold that works in one region or season may fail elsewhere, and they have a hard time with ambiguous pixels at water edges.

B. Global Water Mapping Products

A handful of large-scale water products exist. The JRC dataset [1] is probably the most well-known—it covers 30 years of Landsat data at 30 m resolution using an expert system classifier and provides monthly water occurrence maps for the entire planet. The Copernicus Water Bodies service offers near-real-time maps from Sentinel-2 at 10 m but relies on spectral indices internally. DLR's Global WaterPack uses MODIS for daily coverage at a coarser 250 m. These are useful baselines, but because they depend on traditional classifiers or thresholded indices, they inherit the same limitations.

C. Deep Learning for Water Segmentation

Semantic segmentation itself was transformed by fully convolutional networks [10], with encoder-decoder designs such as SegNet [11] and DeepLabv3+ [12] becoming standard backbones. In recent years, deep learning has made serious inroads into the water-mapping problem specifically. Isikdogan et al. [6] built DeepWaterMap, a fully convolutional network for Landsat data that hit roughly 93% overall accuracy. Wieland et al. [7] applied U-Net to Sentinel-2 and reported IoU values near 0.85, though their data was concentrated in a few regions. Li et al. [8] proposed WaterNet, a U-Net variant for high-resolution GaoFen-2 imagery, and managed F1 scores close to 0.95. Luo et al. [9] added multi-scale attention to U-Net and achieved IoU around 0.80 on Sentinel-2. On the commercial side, Google's Dynamic World and Esri's Land Cover maps include a water class at 10 m, but those were trained on millions of samples and are not really comparable to academic-scale efforts.

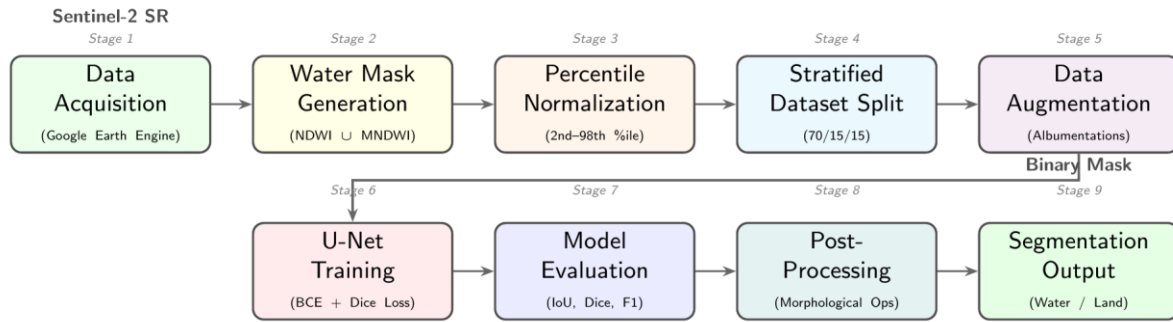


Fig. 1. System overview of the proposed water body segmentation pipeline. The top row covers data preparation (acquisition through augmentation) and the bottom row covers model training, evaluation, post-processing, and final output.

One pattern we noticed across this literature is that the top-performing systems almost always rely on very large training sets—thousands of images or more. Our work takes a different tack: we deliberately kept the dataset small (181 images) but made it as geographically diverse as we could. The question we wanted to answer was how far diversity alone could take us.

III. METHODOLOGY

Fig. 1 gives a bird's-eye view of the entire pipeline, from raw satellite data all the way to the final segmentation output. Each block in that diagram corresponds to one of the subsections below.

A. Data Acquisition

We pulled Sentinel-2 Surface Reflectance imagery from Google Earth Engine [17]. Before writing any model code, we spent considerable time curating 200 target locations distributed across nine geographic regions on every inhabited continent. The locations were chosen to capture three broad categories of water body: coastlines, rivers, and lakes. For each spot, we grabbed the least cloudy Sentinel-2 scene from the June–August 2023 window, filtering out anything with more than 20% cloud cover.

Five spectral bands went into each download: Blue (B2, 490 nm), Green (B3, 560 nm), Red (B4, 665 nm), NIR (B8, 842 nm), and SWIR (B11, 1610 nm). The NIR and SWIR bands were used only for generating the ground truth masks—the model itself sees just RGB. Every image covers 512×512 pixels at 10 m ground sample distance, so each one represents roughly 5.12 km on a side. Out of the 200

planned locations, 19 had to be dropped because of excessive cloud contamination, leaving us with 181 usable images.

B. Ground Truth Generation

Manual annotation of 181 satellite images was not practical for us, and even if it were, subjective disagreements between annotators would introduce noise. Instead, we generated masks automatically by computing two water indices for every pixel:

$$\text{NDWI} = \frac{B3 - B8}{B3 + B8} \quad (1)$$

$$\text{MNDWI} = \frac{B3 - B11}{B3 + B11} \quad (2)$$

A pixel was labeled water if either $\text{NDWI} > 0$ or $\text{MNDWI} > 0$. We used the union of both indices because each one catches cases the other misses—NDWI handles clear open water well, while MNDWI is better in urban areas and for turbid or sediment-laden water. The resulting binary masks became our training targets.

One might reasonably ask: if spectral indices can produce masks, why bother training a neural network? The answer is that the model only receives RGB input at inference time. The goal is to learn a mapping from visible-band appearance to water probability, without needing NIR or SWIR bands that may not be available on all sensors—particularly the compact cameras planned for our CubeSat.

C. Preprocessing

Raw Sentinel-2 reflectance values vary a lot from image to image. Differences in sun angle,

atmospheric conditions, and surface type all play a role. Naive min-max normalization gets thrown off by outliers (cloud edges, sensor noise), so we went with percentile-based normalization instead:

$$x_{\text{norm}} = \text{clip}\left(\frac{x - p_2}{p_{98} - p_2}, 0, 1\right) \quad (3)$$

Here p_2 and p_{98} are the 2nd and 98th percentile pixel values in each image. This clips the extremes and stretches the middle 96% of the distribution to fill the $[0, 1]$ range. We then quantized to 8-bit (0–255) and saved as GeoTIFF to keep the georeferencing intact. The result is a set of images that look visually consistent across very different scenes—a coastal scene in Norway and a river delta in Bangladesh end up with comparable brightness and contrast.

D. Dataset Splitting

We split the 181 images into training (80), validation (16), and test (85) sets, as summarized in Table I. Critically, we stratified by geographic region so that each split contains a representative mix. Without this, the model could memorize region-specific textures—tropical vegetation patterns, for instance—and appear to generalize when it really has not. A fixed random seed of 42 ensured reproducibility.

TABLE I DATASET SPLIT DISTRIBUTION

Split	Count	Percentage
Training	80	70%
Validation	16	15%
Test	85	15%

E. Data Augmentation

Eighty training images is not a lot. To squeeze more out of what we had, we applied data augmentation via the Albumentations library [16] during training. Satellite images have no natural “up,” so geometric transforms were a natural fit: horizontal flips, vertical flips, random 90-degree rotations, and mild shift-scale-rotate perturbations (up to 15 degrees). On the photometric side, we added random brightness and contrast changes ($\pm 20\%$) and a light Gaussian blur. Every transform was applied identically to both the image and its mask, so spatial alignment was never broken.

F. Model Architecture

We went with the original U-Net design [5], which has a symmetric encoder-decoder layout connected by skip connections at each resolution level. Fig. 2 illustrates the architecture schematically. The encoder has four downsampling stages that progressively double the channel count: 64, 128, 256, 512, bottlenecking at 1024. The decoder mirrors this in reverse. At each decoder stage, features from the corresponding encoder level are concatenated via skip connections before being convolved, which helps the network recover spatial detail that would otherwise be lost during downsampling.

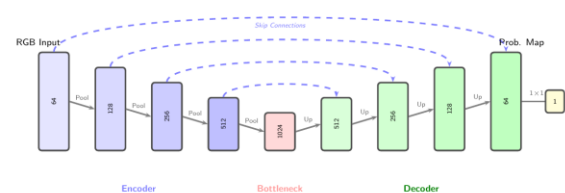


Fig. 2. U-Net architecture used in this work. The encoder (left) downsamples the input through four stages of convolution and max pooling. The bottleneck captures the deepest representation. The decoder (right) upsamples back to full resolution.

Dashed arrows indicate skip connections that concatenate encoder features with the decoder at matching resolution levels. A final 1×1 convolution produces a single-channel probability map. Each stage uses what we call a “double conv” block—two back-to-back 3×3 convolutions, each followed by batch normalization and ReLU. Downsampling is done with 2×2 max pooling; upsampling uses bilinear interpolation. The final layer is a 1×1 convolution that collapses the 64-channel feature map down to a single output channel—essentially, a per-pixel water probability. Input size is $256 \times 256 \times 3$ (RGB). With bilinear upsampling in the decoder (rather than transposed convolutions) the model has about 17.3 million trainable parameters. We deliberately avoided fancier architectures (attention U-Net [13], transformer-based models) because part of our goal is edge deployment. A 17M-parameter model is still on the large side for a CubeSat, but it is at least within reach of optimization techniques like quantization and pruning, whereas a vision transformer would not be.

G. Training Procedure

Training ran for 50 epochs with batch size 16, using the Adam optimizer [14] at a learning rate of 10^{-4} and weight decay of 10^{-5} . For the loss function, we combined BCE and Dice with equal weights:

$$\mathcal{L} = 0.5 \cdot \mathcal{L}_{\text{BCE}} + 0.5 \cdot \mathcal{L}_{\text{Dice}} \quad (4)$$

We chose this combination because BCE alone tends to be dominated by whichever class has more pixels (land, in our case), while Dice loss [15] directly measures region overlap and is less sensitive to class imbalance. Putting them together gave us the best of both. A ReduceLROnPlateau scheduler cut the learning rate in half whenever validation loss plateaued for five consecutive epochs.

We saved three checkpoints during training: the epoch with the lowest validation loss, the one with the highest validation IoU, and the final epoch. For all subsequent evaluation, we used the best-IoU checkpoint. Table II lists the full set of hyperparameters.

TABLE II TRAINING HYPERPARAMETERS

Parameter	Value
Architecture	U-Net (64 base channels)
Input size	256×256
Epochs	50
Batch size	16
Optimizer	Adam
Learning rate	1×10^{-4}
Weight decay	1×10^{-5}
Loss function	0.5 BCE + 0.5 Dice
LR scheduler	ReduceLROnPlateau
Scheduler patience	5 epochs
Scheduler factor	0.5

H. Post-Processing

The model outputs a continuous probability map. We threshold it at 0.5 to get a binary mask, then clean up the result with two morphological operations using a 3×3 kernel: opening first (erosion then dilation) to knock out isolated noisy “water” pixels, followed by closing (dilation then erosion) to patch small holes inside detected water regions. These steps make a noticeable visual difference without shifting the boundaries of larger water bodies.

IV. EXPERIMENTAL RESULTS

Our evaluation setup is outlined in Fig. 3. We loaded the best-IoU checkpoint, ran it over the 85 test images with no augmentation, thresholded and post-processed the outputs, and then computed pixel-level metrics by comparing against the ground truth masks.

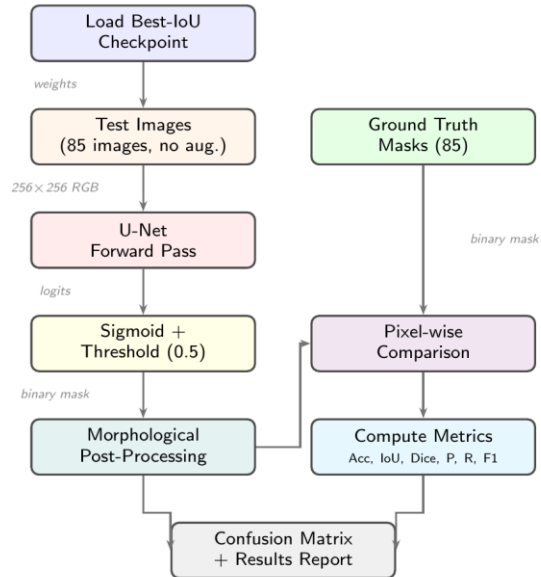


Fig. 3. Evaluation pipeline used for test set assessment. The trained model processes each test image through forward pass, sigmoid activation, thresholding, and morphological cleanup. The resulting masks are compared pixel-by-pixel against ground truth to compute segmentation metrics and the confusion matrix.

A. Training Dynamics

Fig. 4 shows how training and validation metrics evolved over 50 epochs. Training loss fell steadily from 0.710 down to 0.423, which is reassuring—the model kept learning throughout. Validation loss, on the other hand, dropped sharply during the first ten epochs (from 0.695 to around 0.73) and then basically plateaued, bouncing around in a narrow band for the remaining 40 epochs. That early plateau tells us the model hit its generalization ceiling fairly quickly, probably due to the small training set.

The IoU and Dice curves tell a similar story. Training IoU climbed from 0.242 to 0.590, while validation IoU settled around 0.57–0.59 after roughly epoch 15. Training Dice went from 0.383 to 0.737; validation Dice hovered near 0.73. The gap between training

and validation metrics indicates moderate overfitting—the model is fitting some noise in the training set that does not transfer, which is expected with only 80 training images.

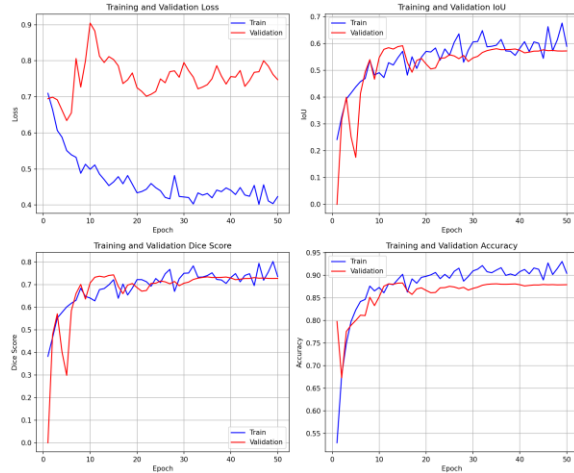


Fig. 4. Training and validation metrics over 50 epochs. Validation metrics plateau early, suggesting the small training set limits further improvement.

B. Test Set Evaluation

We evaluated the best-IoU checkpoint on the 85-image test set. Table III has the full breakdown.

TABLE III TEST SET EVALUATION RESULTS

Metric	Overall	Water	Land
Pixel Accuracy	89.59%	—	—
IoU	75.39%	63.50%	87.28%
Dice / F1	85.44%	77.67%	93.21%
Precision	—	79.17%	92.68%
Recall	—	76.23%	93.75%

The land class numbers are very strong: 87.28% IoU, 93.21% Dice. Water is harder, as expected, but the results hold up well: 63.50% IoU and 77.67% Dice. Precision and recall are nicely balanced—79.17% and 76.23% respectively—so the model neither floods scenes with false detections nor systematically misses water. When it says “water,” it is right about four times out of five, and it recovers roughly three-quarters of all actual water pixels. This aggregate, however, hides substantial variation across regions and water-body types, which we examine in the supplementary material: performance ranges from above 90% water IoU on European coastlines down to under 10% on narrow river channels.

C. Confusion Matrix Analysis

Table IV breaks things down at the pixel level across the whole test set, and Fig. 5 visualizes it.

TABLE IV CONFUSION MATRIX (PIXEL COUNTS)

	Predicted Land	Predicted Water
Actual Land	3,981,312	265,547
Actual Water	314,583	1,009,118

Of about 5.57 million total pixels, the model gets 4.99 million right (89.59%). False negatives—water pixels wrongly called land—number 314,583, or about 24% of all actual water, while false positives are 265,547 land pixels tagged as water, about 6.3% of true land. The errors are fairly balanced between the two types. The model is reliable on obvious water (large open bodies, clear coastlines) and the residual misses concentrate at boundaries, on narrow rivers, and in scenes where the water is murky or partial.

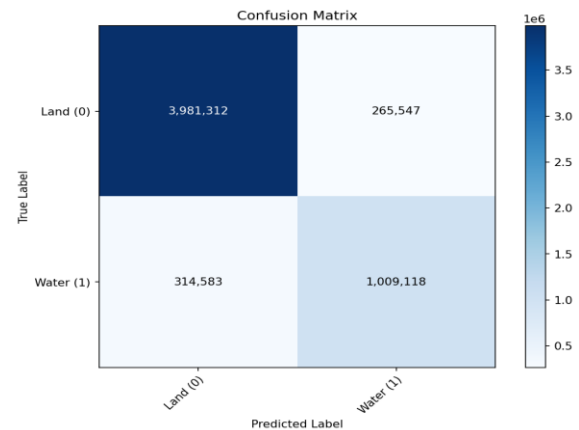


Fig. 5. Confusion matrix for the test set. False negatives (water missed as land) dominate the errors.

D. Discussion of Results

Where do the remaining errors come from? Three factors stand out.

First, the training set is small. Eighty images is far fewer than the thousands used by systems that report water IoU above 0.80. We traded volume for geographic diversity, which we believe was the right call, but it does cap what the model can learn—and the cap shows up most clearly in the regions and water-body types that are thinly represented in training, as the per-region analysis in the supplementary material makes explicit.

Second, there is a spectral information gap. The ground truth was generated using NIR and SWIR

bands, but the model only sees RGB at inference time. It is essentially learning to approximate, from visible-band appearance, something that is much more cleanly separable in the infrared. This puts a ceiling on performance and explains the failure cases in the supplementary qualitative results, where dark or turbid water is nearly indistinguishable from land in the RGB input.

Third, thin structures are hard. The per-type breakdown in the supplementary material shows that narrow river channels collapse to under 10% water IoU—they occupy few pixels and are easily lost across the four downsampling stages of the U-Net. This, rather than any global recall deficit, is the model's real weak spot, and it points directly at the most useful architectural fix: preserving fine spatial detail (decoder attention or higher input resolution).

With balanced precision and recall (both near 77%), the model is usable as delivered, but its operating point is not fixed. Because the Amphitrite interface exposes the detection threshold, users can bias toward higher recall—for flood mapping, say, where missing water is costly—or toward higher precision where false alarms are expensive, on a per-scene basis.

V. AMPHITRITE INTERFACE AND EDGE DEPLOYMENT

A. Interactive Desktop Application

We built Amphitrite to make the model usable by people who are not comfortable running Python scripts. It is a PyQt5 desktop application that accepts satellite images via drag-and-drop or a file browser—GeoTIFF, PNG, JPG, and BMP all work. Inference happens on a background thread so the interface stays responsive, and results appear in four side-by-side panels: the original image, a probability heatmap (Inferno colormap), the binary mask, and a water overlay on the original.

The feature we are most pleased with is the threshold slider. Once the model produces a probability map, the user can slide the threshold up or down, and the mask updates instantly without re-running inference. This makes it very easy to see how the precision-recall tradeoff plays out for a given scene. There is also a toggle for the morphological post-processing, and a statistics panel that reports water and land

coverage percentages, pixel counts, and confidence metrics.

B. Planned CubeSat Edge Deployment

The longer-term motivation behind this work is on-board satellite processing. We are preparing a version of this model for an edge AI module aboard a CubeSat that is currently under development elsewhere in our group. The idea is straightforward: instead of downlinking raw imagery and processing it on the ground, the satellite would run inference on board and transmit only the segmentation mask and summary statistics. Fig. 6 contrasts the traditional ground-processing workflow with our proposed on-board approach.

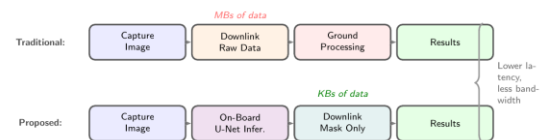


Fig. 6. Traditional satellite workflow versus the proposed on-board processing approach. By running inference directly on the CubeSat, only the lightweight segmentation mask needs to be downlinked, reducing bandwidth requirements by orders of magnitude.

This matters because CubeSat downlink bandwidth is severely limited—often under 1 Mbps. A single raw Sentinel-2-equivalent image might be several megabytes, while a binary mask and a few numbers fit in kilobytes. On-board processing also slashes latency (seconds instead of hours) and opens up autonomous decision-making: the satellite could flag images showing sudden water changes—a flood, for instance—and prioritize them for the next ground station pass.

The challenge is fitting the model onto constrained hardware. Our U-Net has around 17.3 million parameters, which comes out to roughly 66 MB in FP32. That is still too large to run comfortably as-is on most edge platforms. Our planned optimization path includes INT8 quantization (roughly 4× size reduction, 2–3× speedup), weight pruning, and potentially knowledge distillation into a smaller student network. We are also considering reducing the base channel count from 64 to 32 or even 16, which would shrink the model by 4–16× at the cost of some accuracy. Hardware candidates include the

NVIDIA Jetson Nano, Intel Movidius Myriad X, Xilinx Zynq UltraScale+, and Google Coral Edge TPU, each with its own compute-power-radiation tradeoffs.

VI. CONCLUSION AND FUTURE WORK

We presented a complete pipeline—from Sentinel-2 data collection through preprocessing, training, evaluation, and interactive inference—for water body segmentation using U-Net. The dataset was deliberately assembled for geographic diversity, spanning 200 locations across all inhabited continents, and the model was evaluated on a geographically stratified test set to avoid inflated results from spatial leakage. Final test performance came in at 89.59% pixel accuracy, 75.39% mean IoU, and 85.44% mean Dice, with well-balanced water-class precision and recall (79.17% and 76.23%). Performance is strong on coastal and open-water scenes—including several regions held out from training entirely—but degrades on narrow river channels, which remain the main challenge. Amphitrite provides an accessible interface for non-programmers, and edge deployment aboard a CubeSat is the next major milestone.

Where do we go from here? The single biggest lever is more data. Going from 80 training images to a thousand or more would almost certainly improve performance, especially on the underrepresented regions and on narrow river channels where the model is currently weakest. Adding NIR and SWIR as input channels—giving the model the same spectral information we used for ground truth—is another obvious step. On the architectural side, attention mechanisms in the decoder could help the network focus on water boundaries where most of the errors occur. We also want to try focal loss or class-weighted BCE to lift performance on the hardest, water-sparse scenes such as narrow rivers. Test-time augmentation (averaging predictions over flipped and rotated copies of each image) is a low-hanging fruit we have not yet picked. And in the longer run, multi-temporal analysis—feeding sequences of images from the same location—could enable flood monitoring and seasonal change detection, which is ultimately what the CubeSat mission is about.

ACKNOWLEDGMENT

We thank the European Space Agency for making Sentinel-2 data freely available through the Copernicus programme, and Google for providing access to the Earth Engine platform, which made large-scale data acquisition practical.

REFERENCES

- [1] J.-F. Pekel, A. Cottam, N. Gorelick, and A. S. Belward, “High-resolution mapping of global surface water and its long-term changes,” *Nature*, vol. 540, no. 7633, pp. 418–422, Dec. 2016.
- [2] S. K. McFeeters, “The use of the Normalized Difference Water Index (NDWI) in the delineation of open water features,” *Int. J. Remote Sensing*, vol. 17, no. 7, pp. 1425–1432, 1996.
- [3] H. Xu, “Modification of Normalised Difference Water Index (NDWI) to enhance open water features in remotely sensed imagery,” *Int. J. Remote Sensing*, vol. 27, no. 14, pp. 3025–3033, 2006.
- [4] G. A. Feyisa, H. Meilby, R. Fensholt, and S. R. Proud, “Automated Water Extraction Index: A new technique for surface water mapping using Landsat imagery,” *Remote Sensing of Environment*, vol. 140, pp. 23–35, Jan. 2014.
- [5] O. Ronneberger, P. Fischer, and T. Brox, “U-Net: Convolutional networks for biomedical image segmentation,” in *Proc. MICCAI*, Munich, Germany, 2015, pp. 234–241.
- [6] F. Isikdogan, A. C. Bovik, and P. Passalacqua, “Surface water mapping by deep learning,” *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sensing*, vol. 10, no. 11, pp. 4909–4918, Nov. 2017.
- [7] M. Wieland, S. Li, and S. Martinis, “S1S2-Water: A global dataset for semantic segmentation of water bodies from Sentinel-1 and Sentinel-2 data,” *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sensing*, vol. 16, pp. 10198–10214, 2023.
- [8] Y. Li, B. Peng, L. He, K. Fan, Z. Li, and L. Tong, “WaterNet: An adaptive matching pipeline for segmenting water with volatile

- appearances,” *Computational Visual Media*, vol. 7, pp. 455–468, 2021.
- [9] X. Luo, X. Tong, and Z. Hu, “An applicable and automatic method for earth surface water mapping based on multispectral images,” *Int. J. Appl. Earth Observ. Geoinf.*, vol. 103, p. 102472, 2021.
- [10] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Boston, MA, USA, 2015, pp. 3431–3440.
- [11] V. Badrinarayanan, A. Kendall, and R. Cipolla, “SegNet: A deep convolutional encoder-decoder architecture for image segmentation,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 12, pp. 2481–2495, Dec. 2017.
- [12] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, “Encoder-decoder with atrous separable convolution for semantic image segmentation,” in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Munich, Germany, 2018, pp. 801–818.
- [13] O. Oktay, J. Schlemper, L. L. Folgoc, M. Lee, M. Heinrich, K. Misawa, K. Mori, S. McDonagh, N. Y. Hammerla, B. Kainz, B. Glocker, and D. Rueckert, “Attention U-Net: Learning where to look for the pancreas,” in *Proc. Med. Imag. Deep Learn. (MIDL)*, Amsterdam, Netherlands, 2018.
- [14] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *Proc. Int. Conf. Learn. Represent. (ICLR)*, San Diego, CA, USA, 2015.
- [15] F. Milletari, N. Navab, and S.-A. Ahmadi, “V-Net: Fully convolutional neural networks for volumetric medical image segmentation,” in *Proc. 4th Int. Conf. 3D Vis. (3DV)*, Stanford, CA, USA, 2016, pp. 565–571.
- [16] A. Buslaev, V. I. Iglovikov, E. Khvedchenya, A. Parinov, M. Druzhinin, and A. A. Kalinin, “Albumentations: Fast and flexible image augmentations,” *Information*, vol. 11, no. 2, p. 125, 2020.
- [17] N. Gorelick, M. Hancher, M. Dixon, S. Ilyushchenko, D. Thau, and R. Moore, “Google Earth Engine: Planetary-scale geospatial analysis for everyone,” *Remote Sensing of Environment*, vol. 202, pp. 18–27, Dec. 2017.