FloodScan: Flood Mapping Semantic Segmentation Using a U-Net Model with Feature Representations of Sentinel-1 and Sentinel-2 Data

Yu Liu, Max Romanelli, Neelima Srivastava, Rémi Toutin
Georgia Institute of Technology
Atlanta, GA 30332
{yliu3577, mromanelli6, nsrivastava44, rtoutin3}@gatech.edu

Abstract

Recent flood events around the world have brought the importance of understanding flood behavior to the forefront. Synthetic aperture radar (SAR) or multispectral (MS) imagery is commonly used to perform flood extent segmentation. The Sen1Floods11 dataset provides Sentinel-1 (S1) and Sentinel-2 (S2) imagery which are hand-labeled for water bodies. Recent papers have approached this problem differently, applying machine learning (ML) or fully convolutional neural networks (FCNNs) to this problem. However, they are not directly comparable because of differences in evaluation metrics.

We selected four feature sets based on state-of-the-art performances from these papers. We then train U-Net models from scratch using grid-search hyperparameter optimization and compare the performance of the four feature sets with each other. We were able to achieve Intersection over Union (IoU) results without tuning on the test data set that exceeded the baseline presented in the original Sen1Floods11 paper, which relies solely on raw S1 and S2 bands.

Our findings showed that while FCNNs are unable to outperform traditional ML models with an engineered feature space, engineered features do improve the performance of FCNNs. The results highlight the importance of leveraging domain knowledge of spectroscopy in satellite flood segmentation problems.

1. Introduction/Background/Motivation

Flooding is a widespread threat around the world, and it is becoming more prevalent due to climate change. In 2024, there were numerous historically destructive flooding events, such as Hurricanes Beryl, Helene, and Milton in the United States and devastating floods in Afghanistan, Pakistan, Spain, Brazil, and several other regions. With flooding events becoming more frequent and more intense, it is crucial to understand which areas are affected by floods; this can help first responders, insurers, scientists, and others prioritize specific areas to use limited time and resources during a disaster.

Currently, there is satellite surveillance over regions that are susceptible to flooding; satellite images can cover wide areas of floodwaters. However, there are often difficulties in determining from satellite images whether a given area is flooded. For example, there may be clouds that obscure a satellite's view of an area, or it may be difficult to distinguish floodwaters from a permanent body of water. This is where deep learning can help.

We built and trained a U-Net model to build a segmentation map from a satellite image to quickly identify flooded areas. We aim to achieve or improve upon previous floodwater segmentation work in [1], [3], and [4], each of which used FCNNs or ML models.

The data used for training came from Sen1Floods11 [1]. Sen1Floods11, created by Cloud to Street to train and test deep learning algorithms on flooding segmentation for S1 satellite images, is a georeferenced dataset of images taken by the ESA's Sentinel-1 and Sentinel-2 satellites for 11 flooding events around the world in 2016-19 (collected from the Dartmouth Flood Observatory). The 512x512 nonoverlapping image chips depict both flooded and unflooded areas; they often include permanent bodies of water. In addition to raw S1 and S2 imagery, some of the chips are hand-labeled to indicate surface water (ground truth) using permanent water data labels from the JRC's Landsat satellite. The dataset contains three layers of GeoTIFF files (QC for hand-labeled chips, S1 for raw S1 SAR sensor data with two bands, and S2 for raw S2 optical data across 13 channels). The dataset is imbalanced, with only 446 hand-labeled chips and 4385 weakly-labeled chips; the weakly-labeled chips only have S1 bands as their input spectrum. The dataset is accessible through a Google Cloud storage bucket. The dataset's authors also trained and tested several FCNN models using this data and published their training and validation code on GitHub. Both the dataset and the

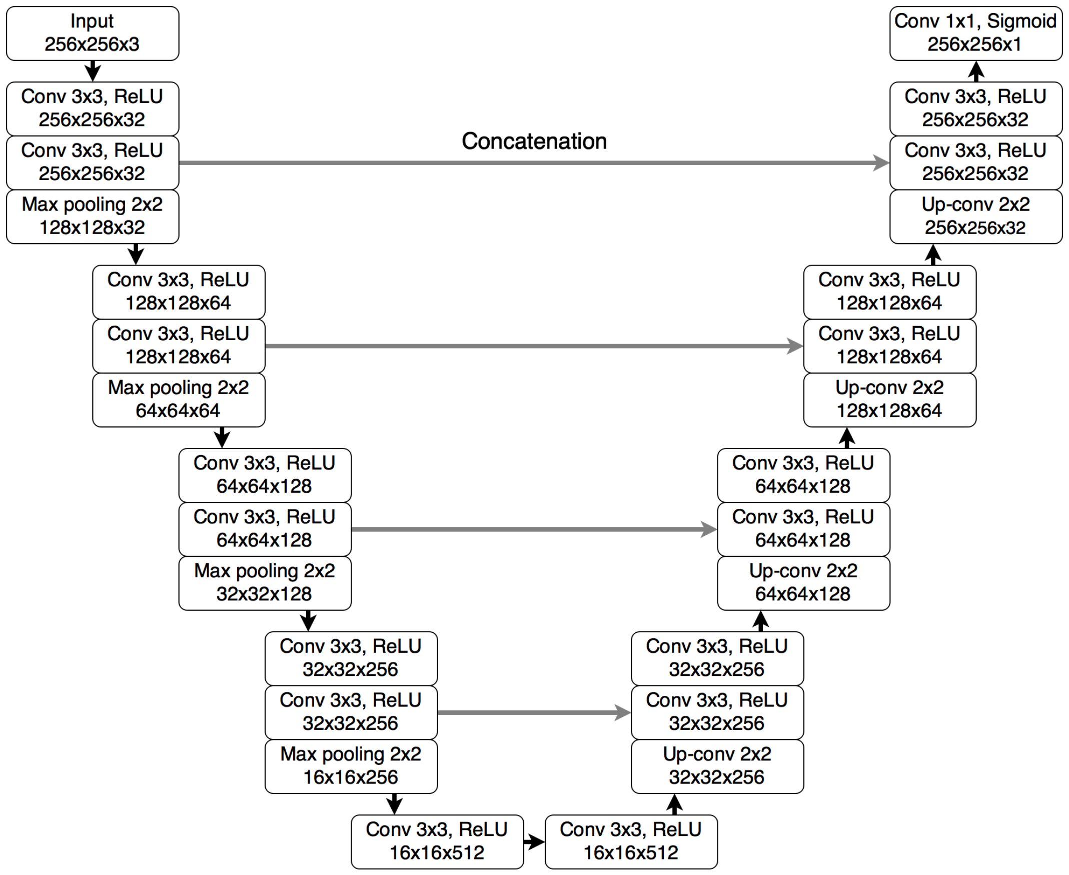

Figure 1. U-Net architecture using the brain-segmentation PyTorch model (Source: Ronneberger, 2015) [5]

code were valuable resources for building and training our own U-Net model.

2. Approach

In this project, we utilize hand labeled flood data and enhance it with handcrafted features derived from Sentinel-1 and Sentinel-2 bands. We then train a U-Net model for segmenting flooded regions. Our objective is to integrate techniques proposed in three different research papers to develop a model that achieves superior performance, particularly in terms of Intersection over Union (IoU). IoU is a critical metric for flood detection, as it balances distinguishing between flooded and unflooded regions, which makes this metric effective for imbalanced datasets.

The first paper we referenced is Sen1Floods11 [1], which introduced a publicly available georeferenced flood inundation dataset. This dataset is ideal for high-quality flood inundation training, validation, and testing, as it includes data from 11 distinct global flood events and comprises 446 manually labeled image chips. These chips are divided into training, validation, and testing sets using a random 60-20-20 split. The dataset provides information for both Sentinel-1 (S1) and Sentinel-2 (S2) bands.

The second paper that informed our project is by Konapala et al. [3], which emphasizes that handcrafted features derived from raw Sentinel-1 (S1) and Sentinel-2 (S2) band values provide more robust results for flood inundation mapping, particularly under environmental challenges such as cloud cover.

The third paper, by Iselborn et al. [4], highlights the significance of feature representation for flood inundation mapping using conventional machine learning algorithms. It also introduces formulas for calculating three novel handcrafted features to enhance training. We have used the following three formulas.

Normalized Difference Water Index (NDWI) uses the NIR and green bands from S2.

$$\text{NDWI} = \frac{\text{GREEN} - \text{NIR}}{\text{GREEN} + \text{NIR}} \tag{1}$$

Automated Water Extraction Index (AWEI) uses the SWIR band and the green band from S2 data.

$$\begin{aligned}\text{AWEI} &= 4 \times (\text{GREEN} - \text{SWIR-1}) \\ &\quad - \frac{1}{4} \times (\text{NIR} + 11 \times \text{SWIR-2})\end{aligned} \tag{2}$$

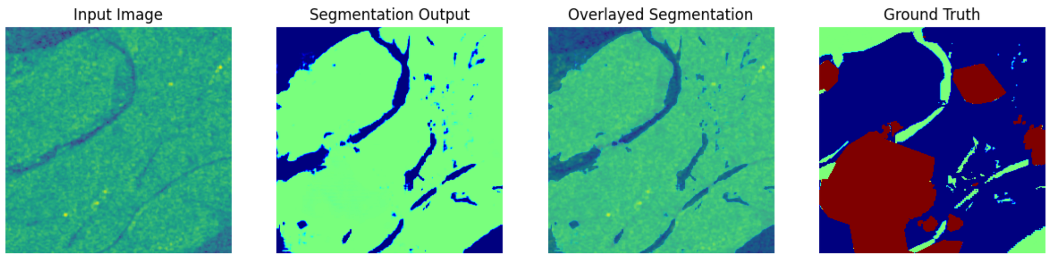

Figure 2. Visualization of learning procedure steps - from left to right: Sentinel-2 Channel B1 (Coastal) raw image in ToA (top of atmosphere) reflectance units x 10,000; Model segmentation output (green='not water', blue='water'); model output overlaid on raw satellite image; hand-labeled ground truth chip (green='water', blue='not water', red='not labeled')

Hue, Saturation, and Value (HSV) color space values are calculated from S2 band values through multiple steps: SWIR2, NIR, red values are assigned to Red, Green, and Blue (RGB) values respectively.

$$V = max(R, G, B) \tag{3}$$

$$S = V - min(R, G, B) \tag{4}$$

$$\begin{cases} 0, \text{if } V = \min(R,G,B) \\ \left(60^\circ \cdot \frac{G-B}{V-\min(R,G,B)} + 360^\circ\right) \bmod 360^\circ, \text{if } V = R \\ 60^\circ \cdot \frac{B-R}{V-\min(R,G,B)} + 120^\circ, \text{if } V = G \\ 60^\circ \cdot \frac{R-G}{V-\min(R,G,B)} + 240^\circ, \text{if } V = B \end{cases} \tag{5}$$

2.1. Data and Features Used

For our experiments, we utilized all 446 manually labeled chips from the Sen1Floods11 [1] dataset. Handcrafted features were added to this dataset, and the model was trained using various combinations of these feature sets: Sentinel-1 (S1) only, HSV only, a combination of S1 + AWEI + NDWI + HSV, and a combination of S1 + AWEI + NDWI.

2.2. U-Net Model

We replaced the ResNet50 model from the Sen1Floods11 [1] project code with a U-Net model, anticipating that U-Net, a fully convolutional neural network (FCNN) explicitly designed for image segmentation, would deliver superior performance in this context. To implement U-Net, we incorporated the version from Buda's 2021 [2] brain-segmentation-pytorch repository into the starter code, modifying it to operate within the existing training and validation loops.

Our U-Net architecture (Figure 1) includes four encoder blocks (encoder1, encoder2, encoder3, encoder4), each containing two convolutional layers followed by batch normalization. Downsampling is applied after each block using max-pooling layers. A bottleneck layer acts as a feature extractor, bridging the encoder and decoder sections. The decoder is symmetric to the encoder, consisting of four decoder blocks (decoder1, decoder2, decoder3, decoder4). Upsampling is performed using transposed convolutions, while skip connections concatenate the upsampled features with their corresponding encoder features to preserve spatial details. The output is passed through a 1x1 convolutional layer to reduce the feature maps to the desired number of channels, with a sigmoid activation function applied to generate pixel-wise probabilities.

2.3. Code Changes

We used the starter code provided in the Sen1Floods11 [1] paper, replacing the original model with the U-Net model described in Section 2.3. Additionally, we implemented code to calculate handcrafted features from Sentinel-2 (S2) data. The training code was modified to handle the feature combinations outlined in Section 2.2.

We also updated the code to support multiple channels and adjusted the processing, augmentation, and loading functions to adapt to the number of input channels based on the feature sets. This included applying normalization based on the means and standard deviations of the channels.

Furthermore, we added code to estimate tunable weight parameters for weighted precision, recall, and F1 metrics in the training, validation, and test loops, as well as incorporating hyperparameter tuning and visualizations.

2.4. Problems Encountered

We anticipated challenges with getting the model to learn, due to large differences in the model architecture and number of parameters. We did initially encounter issues with the model not learning, and attempted multiple fixes that did not work. The actual fix required was for a logical error in the code that handles multiple channels.

We also expected challenges with tuning the hyperparameters related to class imbalance. This is due to the possibility of differences between training and validation data.

Features	our model	Konapala [3]	Iselborn [4]
AWEI+NDWI+HSV	0.7661	-	0.9394
S1+HSV	0.6745	0.9	-
S1+AWEI+NDWI	0.8086	0.88	-
S1+AWEI+NDWI+HSV	0.8150	0.9	0.941

Table 1. F1-score comparison

Features	max valid baseline	last valid baseline	test baseline	max valid tuning	max valid tuned	test tuned
AWEI+NDWI+HSV	0.7292	0.6555	0.6451	0.7038	0.4100	0.3760
S1+HSV	0.5754	0.4840	0.5204	0.5488	0.2900	0.2730
S1+AWEI+NDWI	0.7444	0.7036	0.6945	0.7922	0.7275	0.6757
S1+AWEI+NDWI+HSV	0.7219	0.6621	0.6973	0.6306	0.6171	0.6060

Table 2. IoU (Intersection Over Union) results summary

However, we came into a bigger problem of cross-validated runs not being replicated in the training runs, even after setting the seed. This was addressed by reducing the number of hyperparameters tuned in each run.

3. Experiments and Results

3.1. Hypothesis

We hypothesize that our U-Net models, tuned or otherwise, will perform better than the mean IoU results presented in [1]. The primary objective of [1] is to present a new dataset, therefore achieving higher performance than [1] will serve as a sanity check.

Our objective is to recreate the U-Net used by [3] (based on the U-Net designed for biomedical image segmentation in [5]), evaluate its baseline efficacy on each of four chosen feature representations (two from [4], and two from [3]), and compare our results.

We do not have significant cause to predict whether our repurposed U-Net model will perform better or worse than the models in [4] and [3], but hypothesize that hyperparameter tuning will succeed in improving our models beyond our baseline performance runs.

[4] concludes that a gradient boosting model can outperform neural network models in this task. [4] tries to compare with [3] due to similar feature sets, but had difficulties due to [3] not using the typical IoU metric. We aim to verify this claim.

3.2. Metrics

One of the primary motivations for our approach was the discrepancy between the evaluation metrics and presented results of the supporting works. In the original Sen1Floods11 dataset source paper ([1]), the authors use mean accuracy, mean IoU (intersection over union), and omission/commission error metrics. [4] evaluates their results using mean/total accuracy, mean/total IoU, precision, recall, and F1-score. Results from [3] specifically use "Modified K-fold based median performance metrics" [3], which cannot be directly compared to the mean- and total-based metrics presented by [1] and [4].

We evaluated our models primarily using IoU. Accuracy, precision, recall, and F1-score were also calculated. F1-scores are included for secondary comparison with the results in [3]. It is important to note that accuracy is misleading due to the label imbalance.

3.3. Experimentation

One driver of the design of our hyperparameter tuning grid was the need to address the large class imbalance inherent in the dataset. Our default loss function was the cross-entropy function, and our implementation allowed for passing a weight parameter which would turn the loss function into weighted cross-entropy. The original biomedical U-Net paper ([5]) calculates this weight parameter using the inverse of the frequency of each class in the training data. The total class frequency ratio (0 to 1, or "not water" to "water") was 9.5201, and the mean class frequency ratio was 1521.6. Approximate inverses of these values were taken as weight parameters, so that, for example, a weight parameter of [1, 1000] would indicate that a prediction of class 1 ("water") would be weighted 1000 times higher than class 0 ("not water").

Another big factor in choosing hyperparameters was the need to avoid overfitting of the training data, especially considering the very small size of the hand-labeled Sen1Floods11 dataset (446 'chips' of which only 60% were designated as training data). The weight decay parameter of our AdamW optimizer, adjustments to batch size, and the conversion of all batch normalization of the module to

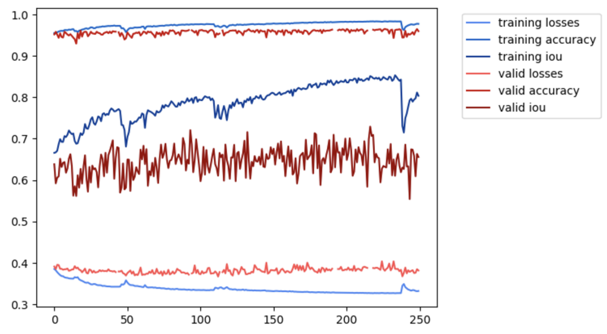

Figure 3. Baseline AWEI+NDWI+HSV model validation and training curves.

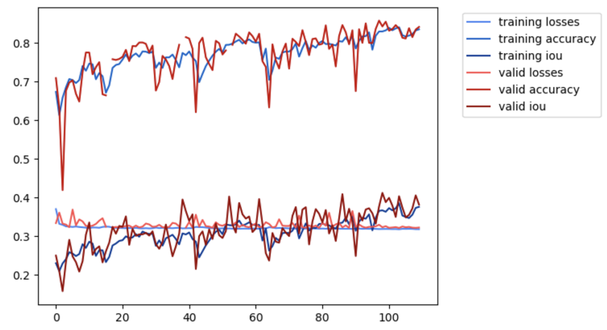

Figure 4. Tuned AWEI+NDWI+HSV model validation and training curves.

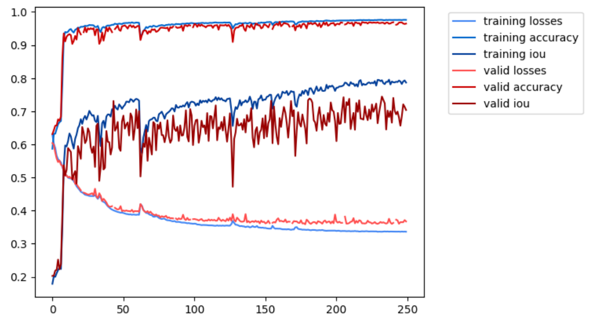

Figure 5. Baseline S1+AWEI+NDWI model validation and training curves.

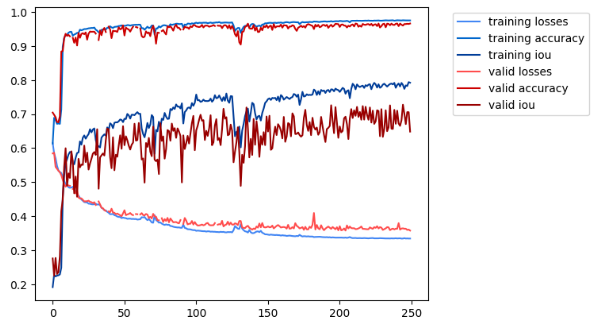

Figure 6. Tuned S1+AWEI+NDWI model validation and training curves with gradual hyper parameter change.

group normalization, were used to increase regularization. Other model specifications of the U-Net were kept constant to maintain hardware stability for training, due to limited GPU RAM.

3.4. Results

According to the visualizations of the segmentation outputs and the ground truth, as seen in Figure 2, the U-Net model performed fairly well at determining the boundaries of water in most cases. The segmentation output often resembled patterns in the raw satellite images and in the ground truth images. However, differences between classified flood extent boundaries to the label cap our model performance.

Results in [4] showed a large variance in model performance based on the choice of test dataset. While our F1-score results in Table 1 are worse than [3], this is within the range of variation for the metric. Our model also sets cross-entropy loss as the criterion, whereas [3] optimizes for F1-score. This comparison also shows the importance of the choice of performance metric to optimize for.

What we found in hyperparameter tuning was that the models were very sensitive to starting seeds and initialization values. This can be seen in Table 2, where tuned models struggle to outperform the baseline models. Difficulties in tuning are also shown in Figures 3 and 4. 3 started with good model performance, while 4 had poor initialization and model performance that were not overcome through training. Impact of initialization eclipses hyperparameters in the current setup.

Our primary experimental method of hyperparameter tuning with a focus on addressing class imbalance and avoiding overfitting was not successful. After this, our approach shifted to making longer training runs. The results of this strategy can be seen in Figures 5 and 6 where we only adjusted the weight hyperparameter for cross-entropy loss. This improved the validation IoU 0.7036 of the base run to 0.7275 for the tuned run. As long as the learning rate was set reasonably conservatively, training on larger number of epochs was beneficial to model performance.

The gradual adjustment of hyperparameters was met with incremental success, but it did not resolve the erratic and wide-ranging fluctuations of the validation IoU curves. This behavior, which can be seen in the validation curves of Figures 3, 4, 5, and 6, indicates that the optimizer starts to overfit after a small number of epochs. The current setup uses AdamW as the optimizer, but it is not typically recommended for small datasets. SGD (stochastic gradient descent) with momentum was tested as an alternative. The resulting training and validation curves are shown in 7. We can see that, compared to Figures 3, 4, 5, and 6, the validation curve fluctuation in 7 is much smaller, which indicates

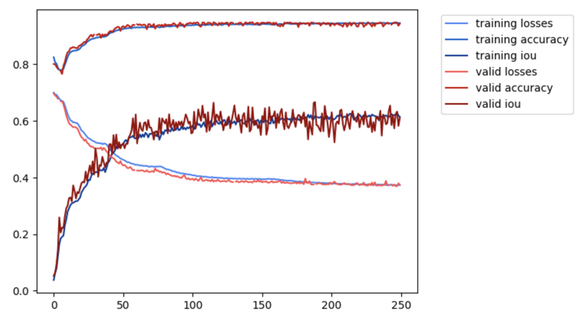

Figure 7. Baseline AWEI+NDWI+HSV model, with optimizer changed to SGD with momentum, validation and training curves

less training and validation volatility. A smaller gap between training and validation IoU suggests less overfitting, which indicates that SGD is the more suitable optimizer for future runs.

3.5. Conclusion

Our first hypothesis was that our models would outperform the results from [1]. [1] present mean IoU results of 0.4084 for Sentinel-2 channels and 0.3871 for Sentinel-1 channels. Our test scores for base models, for all four combinations of feature representations, outperformed the results from [1], confirming that our first hypothesis and our sanity check were correct and satisfied.

Regarding our second hypothesis, which was that our models would improve after hyperparameter tuning, our experimentation was mostly unsuccessful. As discussed in previous subsections, volatility and sensitivity factors played a large role in model variance between runs. This meant that performance improvements expected from model tuning were challenging to fully replicate in the final runs which involved longer training times from larger number of epochs.

Our results did not match the performance of the gradient-boosted model in [4]. [4] presents a distribution of IoU values, with a mean of 0.7998, a minimum of 0.5468, and a maximum of 0.9713. Our results show a peak test IoU performance of 0.6973. Our best model IoU performs within the range expected by [4], but still falls short of their average. Our F1-score of 0.815 also fell short of the 0.9 from [3], suggesting greater room for improvement when it comes to model tuning and architecture.

Our primary findings confirm the findings of [4]: feature representations improve the base performance of models versus inputting raw Sentinel-1 and Sentinel-2 channels. An important secondary finding is that, for a small dataset should use an optimizer like SGD with momentum instead of AdamW to instability during training. Based on comparing our results so far with [4], we can conclude that in the task of flood classification with this dataset, it is plausible that a neural network-based approach could potentially outperform classical classifiers.

4. Future Work

Future work in flood inundation modeling using a small hand-labeled dataset could benefit from adopting optimizers like Stochastic Gradient Descent (SGD) as it generalizes well with limited datasets. SGD could be paired with fine-tuned learning rates and momentum parameters to better handle the limited data and batch sizes.

A gradual hyperparameter tuning strategy of adjusting key hyperparameters like learning rate, weight decay and network depth could ensure steady progress towards achieving a higher IoU while ensuring sudden performance degradation is minimized. Employing plot-based hyperparameter search techniques to monitor the gap between training and validation scores could facilitate the early detection of overfitting or underfitting, optimizing model performance.

Future work can also leverage weakly-labeled datasets to access larger volumes of data, addressing the limitations posed by the scarcity of flood dataset with labels. This would improve model generalizability. Another way to increase the data volume is to leverage flood extents created by catastrophe model vendors and attach S1 and S2 data to them. These commercially created labels are of comparable quality as the hand-generated labels in the Sen1Floods11 dataset.

Additionally, exploring the use of wider and shallower network architectures with varying batch and group sizes presents an opportunity to enhance modeling approaches. The type of FCNNs chosen as the encoder and decoder are also expected to impact model performance. Additional hyperparameter search can be conducted on model structure, model size and batch/group sizes to search for the most optimal model structure suited for the flood segmentation problem.

5. Work Division

As shown in Table 3, our team dynamics were well-structured and balanced, with each member contributing specialized expertise to ensure comprehensive project development. Liu worked on feature set creation, additional feature calculation, and hyperparameter tuning. Romanelli managed functionality setup, U-Net modifications for multi-channel inputs, class imbalance, and optimization issues. Srivastava focused on hyperparameter tuning, grid search, and fine-tuning models. Toutin contributed through visualization and training.

Student Name	Contributed Aspects	Details
Liu	Create feature sets and experimentation, Abstract, General polish and proofreading	Load data from both S1 and S2 to calculate and combine additional features from S2 bands and group feature sets Hyperparameter tuning on one of the feature sets (S1+AWEI+NDWI+HSV). Adding test metrics to runs.
Romanelli	Initial Functionality Setup, Experiment Hypotheses and Hyperparameter Tuning, Analysis, Results Interpretation, Section 3	Incorporated U-Net code from [2], modified all net outputs in train, valid, and test loops; Added necessary functionality for notebook to work with more than just 2 input channels, including modifying the data augmentation and processing classes/functions, and calculating channel means and standard deviations of training dataset for a dynamic number of channels, for normalization; Proposed parameters to address class imbalance, estimated tunable weight parameters for weighted cross-entropy loss function by calculating inverse of class label frequencies in training data; Added precision, recall, and F1 metrics to outputs of training, validation, test loops; added option for changing default initialization of Conv2d weights in UNet model; resolved a code issue that was plaguing our performance for a significant portion of the project duration; in the last 24 hours before submission, discovered that the SGD with momentum optimizer resolves a major ongoing volatility problem - implemented an experiment, tested, gathered the evidence/measurements, and incorporated the results into an extension/continuation of a greater gradual hyperparameter adjustment strategy (AWEI+NDWI+HSV)
Srivastava	Hyperparameter tuning and Experimentation, Sections 2 and 4	Implemented code for hyperparameter grid search to optimize the model and report the best combination of hyperparameters; Modified U-Net code for experimenting with the additional dropout layers as a way to mitigate overfitting of model showing high training IOU; Trained and Fine Tuned the model for feature set (S1+AWEI+NDWI). Experimented with gradual changes in hyper parameter values and achieved a tuned parameter run performance slightly better than the base parameter run.
Toutin	Visualization, Experimentation, Section 1, Table 1, Appendix, proofreading	Visualized segmentation results. Trained and fine-tuned the model on a feature set (S1+HSV).

Table 3. Contributions of team members.

References

[1] Tellman B. Anderson T. Issenberg E. Bonafilia, D. Sen1floods11: a georeferenced dataset to train and test deep learning flood algorithms for sentinel-1., 2020. 1, 2, 3, 4, 6

[2] Mateusz Buda. brain-segmentation-pytorch. *GitHub repository*, 2021. 3, 7

[3] Shahryar Khalique Ahmad Goutam Konapala, Sujay V. Kumar. Exploring sentinel-1 and sentinel-2 diversity for flood inundation mapping using deep learning. *Exploring Sentinel-1 and Sentinel-2 diversity for flood inundation mapping using deep learning*, 180:163–173, 2021. 1, 2, 4, 5, 6

[4] Takashi Miyamoto Marlon Nuske Andreas Dengel Kevin Iselborn, Marco Stricker. On the importance of feature representation for flood mapping using classical machine learning approaches, 2023. 1, 2, 4, 5, 6

[5] Thomas Brox Olaf Ronneberger, Philipp Fischer. U-net: Convolutional networks for biomedical image segmentation. *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2015. 2, 4

Appendix

Additional materials are presented in this section.

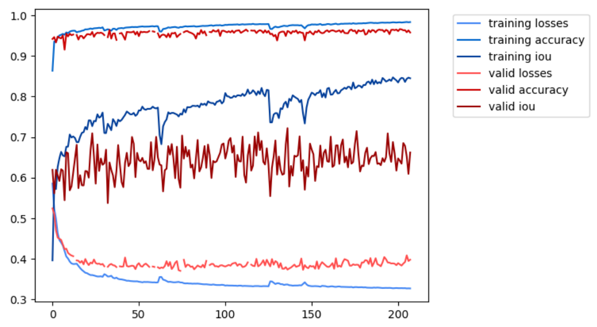

Figure 8. Baseline S1+AWEI+NDWI+HSV model validation and training curves.

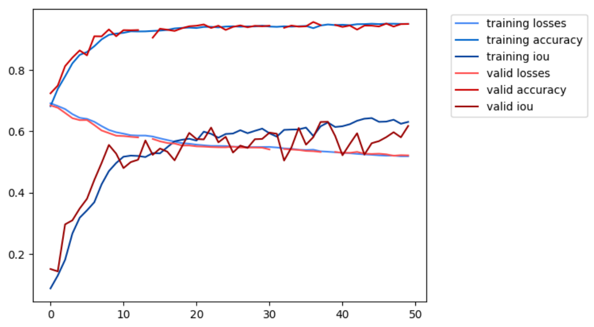

Figure 9. Tuned S1+AWEI+NDWI+HSV model validation and training curves.

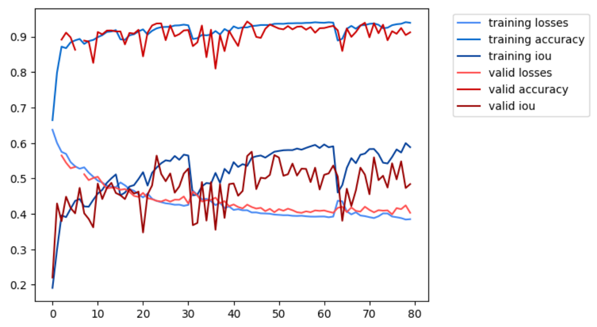

Figure 10. Baseline S1+HSV model validation and training curves.

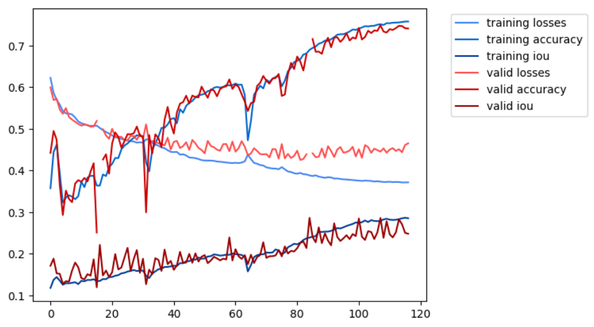

Figure 11. Tuned S1+HSV model validation and training curves.