Deep Learning for Flood Damage Detection Using Satellite and Real-World Imagery

Team Members:
Teja Reddy Mandadi, Visesh Bentula, Umapathi Konduri

1. Dataset Update and Preprocessing

In the original proposal, two datasets were selected for flood detection experiments:

- Flooding Image Dataset (hhrclemson/flooding-image-dataset) – containing ~9,000 RGB images labeled as flooded or non-flooded for classification tasks.
- SEN12-FLOOD Flood Detection Dataset (rhythmroy/sen12flood-flood-detection-dataset) – a multispectral Sentinel-1/2 dataset intended for satellite-based flood segmentation.

During implementation on the TAMU HPRC cluster, both datasets presented practical issues:

- The Flooding Image Dataset extracted only a single "flooded" folder, leaving missing or mis-organized non-flooded images, making balanced training impossible.
- The SEN12-FLOOD dataset was over 12 GB and consisted of numerous multispectral .tif files from Sentinel-1 and Sentinel-2 sensors. These required complex geospatial preprocessing and band alignment steps, and initial parsing attempts returned zero valid image–mask pairs due to inconsistent folder structures.

To ensure meaningful progress, the project adopted the Flood Area Segmentation Dataset by Faizal Karim et al. (https://www.kaggle.com/datasets/faizalkarim/flood-area-segmentation/data) . This dataset contains ~300 paired RGB images and binary masks of flooded regions already aligned and labeled. It offered a manageable size and structure compatible with PyTorch, allowing seamless training on the cluster without administrative access.

After manual upload and verification, 291 images and 290 corresponding masks were successfully recognized. All samples were resized to 224 × 224 pixels, normalized to ImageNet statistics, and split 80 % / 20 % into training (232) and validation (59) sets.

This dataset supports the same research objective — flood area detection and segmentation from imagery — while being computationally feasible for this project phase.

2. Exploratory Analysis and Model Setup

The training images and masks were inspected for alignment and pixel intensity distribution. Visualizations of sample image–mask pairs confirmed that the dataset clearly differentiates between flooded and dry regions.

A custom U-Net Convolutional Neural Network was implemented in PyTorch for semantic segmentation. The network consists of four encoder–decoder blocks with 3×3 convolutions, batch normalization, ReLU activation, and 2×2 max-pooling, followed by transposed convolutions for upsampling.

Training Configuration:

Loss Function: Binary Cross-Entropy (BCE)

Optimizer: Adam (lr = 1e-4)

Input Size: 224×224

Epochs: 10

Batch Size: 4

Device: NVIDIA GPU (HPRC JupyterLab)

3. Initial Training Results

Epoch	Train Loss	Val Loss
1	0.456	0.454
5	0.351	0.403
10	0.309	0.398

Both training and validation losses decreased consistently, demonstrating stable learning and good generalization on the limited dataset.

Mean IoU ≈ 0.62 and Mean Dice ≈ 0.75 were achieved on the validation set.

Loss Curve: Training loss shows a steady decline while validation loss stabilizes near 0.39

(Figure 1: Training vs Validation Loss Curve — see GitHub repository)

4. Challenges and Next Steps

Challenges

- Cluster restrictions (no sudo, unzip, or GUI access) required manual data upload and Python-based extraction.
- Unbalanced and small dataset size (~300 samples) limited model generalization.
- Minor file mismatches between images and masks caused initial index errors, resolved via custom data loader.

Next Steps

1. Add data augmentation (flip, rotate, brightness jitter) to increase sample diversity.
2. Experiment with transfer learning (ResNet-34 encoder U-Net).
3. Incorporate evaluation metrics like Precision, Recall, F1, and confusion matrix.
4. Extend training to 25 epochs and add visualizations of predicted flood areas for final report.

Summary

Despite dataset and environment limitations, the U-Net model successfully segmented flooded regions with reasonable accuracy and stable loss behavior. This update demonstrates a working baseline for flood segmentation on limited data. Future work will focus on transfer learning, augmentation, and metric optimization to improve performance and robustness.

GitHub Repository: https://github.com/tejareddy615/MLDL_FloodDetection/