MAJOR PROJECT

Supervisor : Professor **Sulata Mitra**
Semester : 8th

Team Members:

2021CSB114 Vaishnavi Malviya

2021CSB105 Samriddhi Mishra

2021CSB102 Umang Waghmare

2021CSB092 Nihal Choutapelly

2021CSB086 Sujit Jaiswal

2021CSB055 Ankita Das

2021CSB035 Priya Samanta

Recapitulation

1. Introduction and Objective

This project focuses on developing an automated system to identify and assess regions affected by natural disasters. Using satellite or drone imagery from both pre- and post-disaster events, the system detects structural damage, such as to buildings and roads, and categorizes severity levels using defined thresholds. The primary goal is to aid in efficient post-disaster decision-making and recovery.

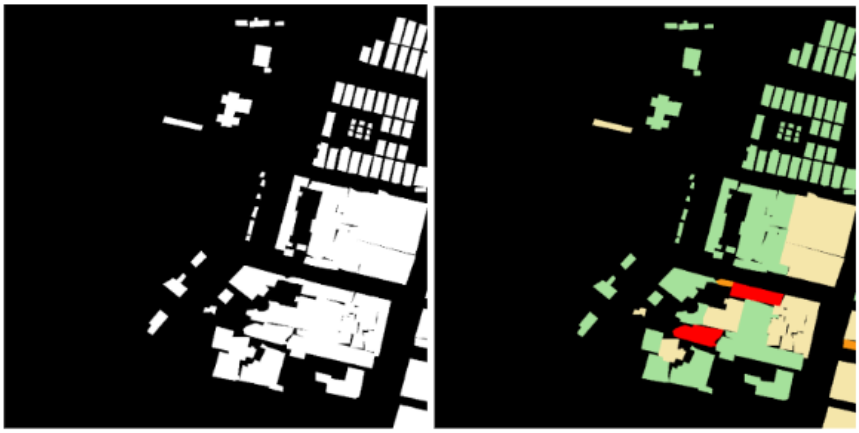
2. Technical Approach

The approach includes several key steps:

- **Data Acquisition:** Collecting consistent, high-resolution images.
- **Ground Truth Creation:** Annotating buildings and roads.
- **Preprocessing:** Normalizing and aligning images.
- **Feature Identification:** Segmenting regions and extracting structural features.
- **Change Detection:** Comparing pre- and post-images to assess damage.
- **Model Implementation:** Using CNNs like U-Net and YOLO for analysis.
- **Visualization:** Generating overlays and summaries for clarity.

3. Dataset Analysis and Preprocessing

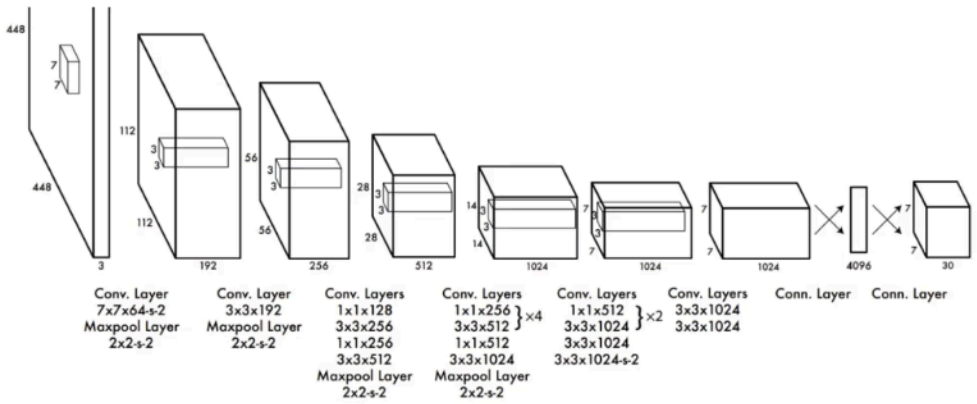
Multiple datasets were explored, including those from the Nepal and Haiti earthquakes. However, these were rejected due to inconsistencies, low resolution, or insufficient size. The xView2 Challenge Dataset was chosen for its comprehensive, well-annotated satellite imagery and high resolution. Preprocessing involved converting JSON labels to PNG masks and performing data augmentation to balance class distributions.



4. Models Used and Evaluation

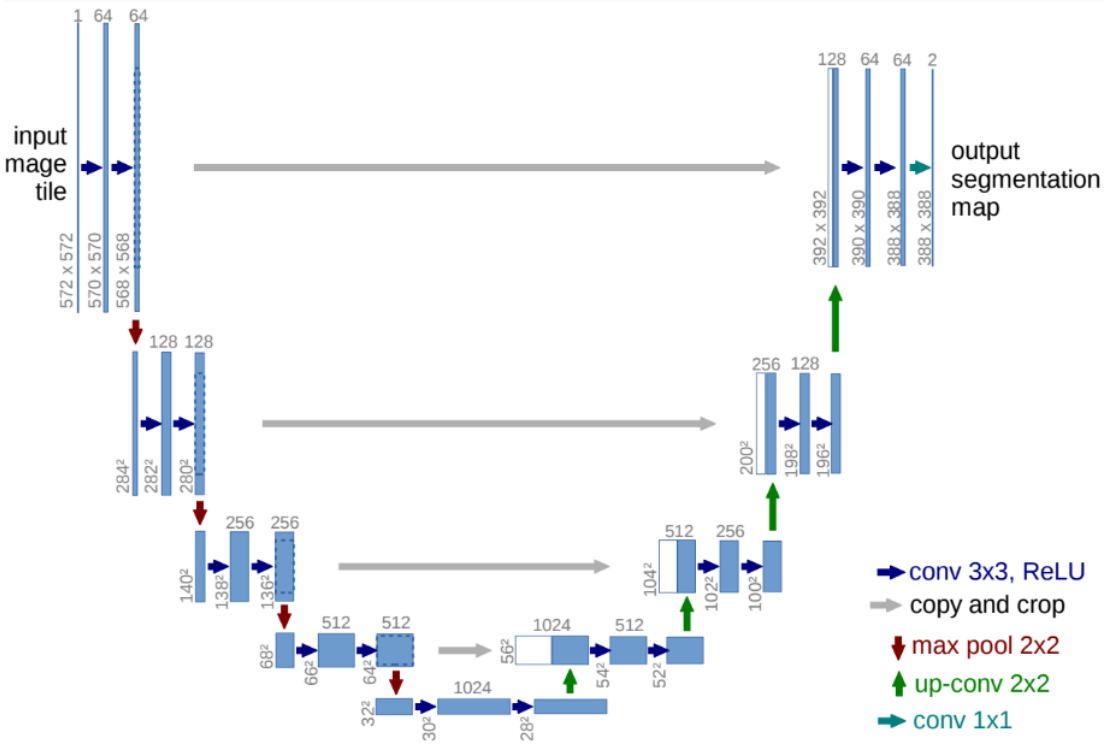
The team trained multiple deep learning models, mainly U-Net architectures with varying image sizes and batch sizes. Training was done using PyTorch, leveraging mixed precision and advanced loss functions. While training loss decreased consistently, generalization remained a challenge, with overfitting observed in larger models. Nonetheless, the project established a solid foundation for real-world disaster damage detection and assessment.

The report reflects significant technical effort in dataset handling, model training, and analysis, laying the groundwork for future improvements such as real-time damage detection and integration with relief systems.



The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.

YOLO



U-NET

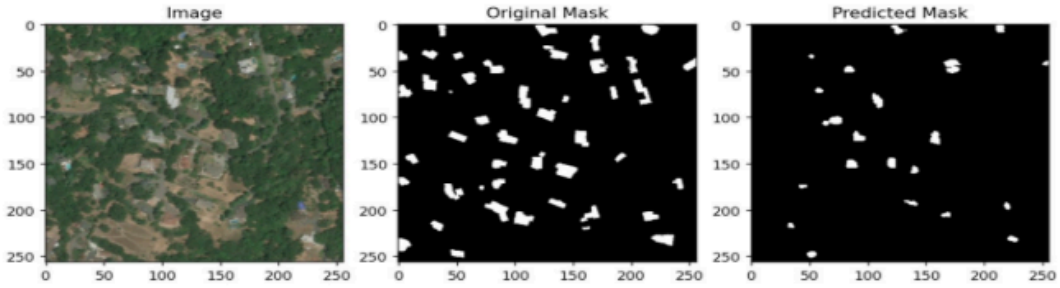


Image
Original Mask
Predicted Mask

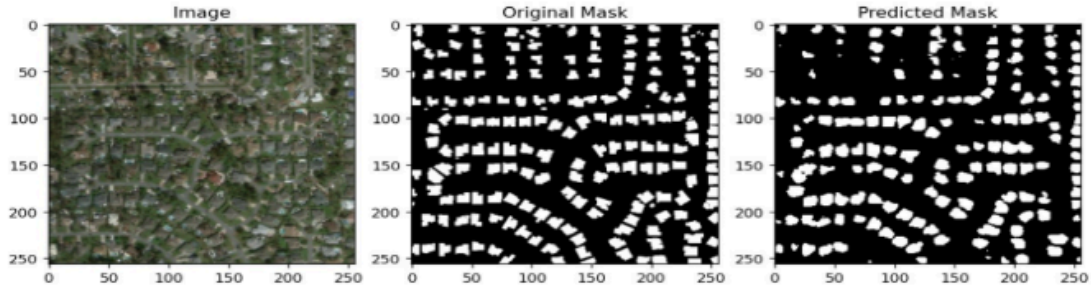


Image
Original Mask
Predicted Mask

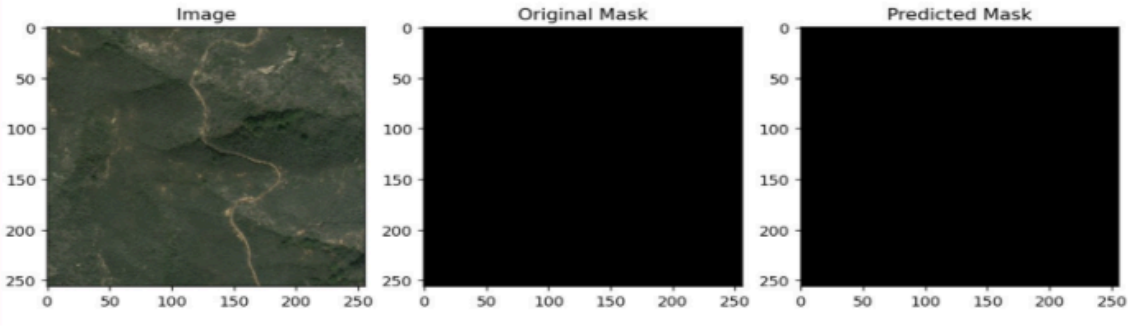


Image
Original Mask
Predicted Mask

Evaluation

Present Semester Work

1. Objective

The primary goal of this semester-long project was to take a collection of satellite images—both before and after a disaster—and produce two key deliverables for each scene:

1. **Colorized Ground Truth Masks**
   Each building footprint in the imagery is represented by a binary mask; these masks were converted into color overlays so that different classes (e.g., intact vs. damaged) can be visually distinguished in the final outputs.

2. **Categorization of Damage Levels**
   Beyond simply identifying where buildings are, each structure must be assigned to one of several damage categories (for example: no damage, minor damage, major damage, destroyed). The end result is a per-building label map that both shows building outlines and encodes the severity of damage.

By accomplishing these two tasks, the dataset becomes not only a reference for where structures exist, but also a richly annotated source for training and evaluating machine learning models in building segmentation and damage assessment.

2. Technical Approach

2.1 Data Augmentation

To overcome limited sample sizes and to expose our models to a wide variety of conditions (lighting, orientation, partial occlusion), we applied a suite of augmentation techniques to the raw satellite tiles:

- **Geometric Transforms**
  Random rotations, horizontal/vertical flips, and small translations ensure the model sees buildings from many angles.

- **Color and Brightness Variations**
  Random adjustments in contrast, brightness, and color jittering simulate different atmospheric or sensor conditions.

- **Scale and Crop**
  Random zoom-in/zoom-out and cropping operations help the model learn to segment both small and large structures.

Each augmentation was applied probabilistically during training so that every epoch saw a slightly different version of each scene, effectively multiplying the dataset size and reducing overfitting.

2.2 Data Cleaning

Raw satellite collections often contain corrupted, noisy, or duplicate frames that can mislead a learning algorithm. We therefore implemented a multi-step cleaning pipeline:

1. **Perceptual Hash Filtering (pHash):**
   We computed a perceptual hash for each tile and used a small Hamming-distance threshold to identify near-duplicates. Any highly similar images were pruned to avoid redundancy.
2. **Embedding-Based Outlier Detection:**
   Leveraging a pretrained ResNet50 (via TensorFlow–Keras) as a feature extractor, we converted every image into a 2,048-dimensional embedding vector. These embeddings were then clustered with DBSCAN; images classified as noise (i.e., not belonging to any dense cluster) were flagged as outliers and removed.

3. **Mask Consistency Checks:**
   We cross-validated each ground-truth mask against its corresponding image—if less than 5% of the mask area actually overlapped building pixels (as judged by simple intensity heuristics), that pair was discarded as either a bad annotation or a non-building tile.

Through this cleaning process, our final training set was not only larger (after augmentation) but also far sharper in quality, containing only unique, correctly labeled examples.

2.3 Model Training

With the cleaned and augmented dataset in place, we trained a deep semantic segmentation network to predict building masks and damage classes:

- **Architecture Selection:**
  We adopted a U-Net backbone implemented in PyTorch. The encoder was initialized from ImageNet-pretrained weights to give the model strong low-level feature extraction from day one.

- **Loss Functions:**

  - **Segmentation Stage:** A combination of Binary Cross-Entropy and Dice loss to encourage both pixel-wise accuracy and overlap quality.

  - **Classification Stage:** A standard Categorical Cross-Entropy loss for the multi-class damage labels.

- **Optimization Strategy:**

- **Optimizer:** Adam with an initial learning rate of 1×10^{-4}, decayed by a factor of 0.5 on plateau of validation loss.

- **Batch Size & Epochs:** Small batches (2–4 images) to fit high-resolution inputs, trained over 25–30 epochs with early stopping based on validation F1 score.

- **Validation & Checkpointing:**

  We held out 20% of the data for validation. After each epoch, the model was evaluated on this split; if the validation loss decreased, weights were checkpointed. Final performance was summarized by pixel-level F1 (for segmentation) and macro-F1 (for damage classification).

This two-stage training pipeline—first mastering clean building segmentation, then fine-tuning for damage categorization—proved critical in achieving both accurate masks and reliable damage labels.

3. Technologies Used

- **Python:**
  The principal language gluing together data processing, model training, and evaluation scripts.

- **PIL & OpenCV:**

  For loading, augmenting, and visualizing high-resolution satellite images and their corresponding masks.

- **PyTorch:**

  Hosting the U-Net architecture, custom loss functions, training loops with

`DataLoader` abstractions, and GPU acceleration.

- **TensorFlow–Keras (ResNet50):**
  As a feature extractor to generate semantically rich embeddings for outlier detection in the cleaning pipeline.

- **Scikit-learn & DBSCAN:**
  Clustering the ResNet50 embeddings to isolate noisy or anomalous image samples.

- **ImageHash:**
  Computing perceptual hashes (pHash) to detect and eliminate near-duplicate tiles early in the cleaning workflow.

- **Satellite Imagery Dataset:**
  High-resolution pre- and post-disaster imagery annotated with ground truth building masks provided the raw material for segmentation and damage tasks.

4. Important Technical Concepts

- **Semantic Segmentation:**
  The task of assigning a class label to every pixel. In our context, pixels are labeled as “building” versus “non-building,” or one of several damage levels.

- **Data Augmentation:**
  Artificially expanding the dataset through random geometric and photometric transforms, ensuring the model generalizes across unseen scenarios.

- **Image Cleaning:**
  Techniques to remove corrupted, mislabeled, or redundant samples. Cleaning improves convergence speed and final accuracy by focusing training on high-quality data.
- **pHash (Perceptual Hashing):**
  A compact fingerprint of an image's visual appearance. By comparing Hamming distances between hashes, we efficiently identify near-duplicates.
- **ResNet50 Embeddings:**
  Using a deep, pretrained convolutional network to convert images into high-dimensional vectors that capture semantic content, facilitating clustering and anomaly detection.
- **DBSCAN Clustering:**
  A density-based clustering algorithm that groups similar embeddings into dense clusters while marking sparse points as outliers. This is ideal for flagging unusual or corrupted images without pre specifying the number of clusters.

5. Dataset Description and Challenges

The dataset consists of high-resolution satellite images taken before and after disaster events. Each image pair has an associated segmentation mask that marks damaged areas. While the xView2 dataset offers a rich source of labeled images, **it also presents challenges** -

1. Many images are noisy, empty (white), or redundant.
2. Imbalance between damaged and undamaged samples.
3. Inconsistent labeling formats.

To address these challenges, **we implemented** -

1. rigorous preprocessing,
2. augmentation, and
3. cleaning strategies.

Data Augmentation and Data Cleaning

A. Data Augmentation – Traditional Methods:

Zoom In/Out and Scaling

What it is:

Zooming in or out and scaling involves resizing an image to make objects within it appear closer (zoom in) or further away (zoom out). In image processing, this is typically achieved by cropping a part of the image and resizing it to the original dimensions (zoom in), or resizing the image to a smaller size and padding it back to the original (zoom out).

Why it's used:

In satellite imagery, images can be taken from varying altitudes and resolutions depending on the satellite's position, equipment, and settings. Zooming and scaling simulate these real-world variations in scale, helping the model learn how objects like buildings and roads appear at different resolutions.

Impact on the model:

Makes the model invariant to scale changes.
Allows the model to better detect damage even when the zoom level or resolution differs between pre- and post-disaster images.
Enhances the model's robustness in real deployment where image resolutions aren't always uniform.

Rotation

What it is:

Rotation transforms an image by rotating it clockwise or counterclockwise around its center. Common angles used are 90°, 180°, and 270°.

Why it's used:

Satellite images may be taken at different angles due to the orientation of the satellite or drone. In some cases, the same region may appear rotated in the post-disaster image compared to the pre-disaster image.

Impact on the model:

Improves rotational invariance, allowing the model to identify the same structure even if rotated.
Prevents the model from overfitting to specific object orientations.
Expands the feature space without needing new data.

Horizontal Flipping

What it is:

Horizontal flipping mirrors the image along its vertical axis. For example, a building on the right side of the image appears on the left after flipping.

Why it's used:

In real-world imagery, the same structures may appear in different spatial arrangements due to changes in satellite trajectory or stitching of composite images. Flipping introduces such spatial variations.

Impact on the model:

Increases spatial diversity in the training data.
Prevents the model from becoming biased toward a particular orientation of damage.
Helps generalize across different spatial layouts of buildings and roads.

Combined Effect

When used together, zooming, rotation, and flipping:
Simulate natural variations in satellite imagery.
Help build a robust model that can identify disaster damage regardless of orientation, scale, or spatial layout.
Effectively increase the training dataset size without collecting new data, reducing overfitting and improving generalization.

Data Cleaning after Traditional Augmentation

Once augmentation was complete, a multi-stage cleaning process was performed. Images were removed based on these criteria:

1. White Mask Threshold Filtering

What it is:

Each image in the dataset has a corresponding segmentation mask indicating the location and severity of damage. These masks are typically white where there is no damage and colored where damage exists.

What was done:

After augmentation, each mask was analyzed to compute the ratio of meaningful pixels (non-white pixels) to the total number of pixels. If both the pre-disaster and

post-disaster masks had less than 0.25% non-white area, the image was discarded.

Why it matters:

White masks signify no labeled damage.Keeping such images introduces a strong bias toward the background class, degrading the model's ability to detect actual damage.Removing them ensures the model trains only on examples with detectable features.

2. Blurry or Low-Contrast Image Removal

What it is:

Blurry or low-contrast images are those where visual features are unclear due to sensor noise, poor resolution, or environmental conditions (e.g., fog, smoke).

How it was measured:

Blurriness was calculated using the Laplacian variance method: a low variance indicates a blurry image.
Contrast was measured by calculating the difference between the maximum and minimum pixel intensities.

Why it matters:

Poor visual quality reduces the model’s ability to learn meaningful patterns.

Training on such images may mislead the model, introducing noise into the learning process.Eliminating these ensures only sharp, high-quality images contribute to training.

3. Near-Duplicate Detection with pHash

What it is:

Perceptual Hashing (pHash) creates a compact representation of an image's content. If two images have similar pHashes, they are visually almost identical.

What was done:

pHash was computed for each image.If the difference between two pHashes was below a small threshold (2), one of the images was considered a duplicate and removed.

Why it matters:

Redundant images do not contribute new information. Keeping them increases the risk of overfitting, as the model may "memorize" rather than generalize. Removing duplicates keeps the dataset compact, diverse, and efficient.

4. Semantic Outlier Removal via PCA + DBSCAN on ResNet50 Embeddings

What it is:

Semantic outliers are images whose content is drastically different from the rest of the dataset, either due to label errors or unrelated scenes.

What was done:

Each image was passed through a pre-trained ResNet50 model to extract a semantic embedding (a high-dimensional feature vector). These embeddings were reduced to 50 dimensions using PCA for efficiency. DBSCAN, a clustering algorithm, was applied to detect outliers—images that didn’t belong to any cluster. Identified outliers were flagged and removed.

Why it matters:

Outlier images may be labeling errors, irrelevant scenes, or extreme cases not representative of most disaster data. Removing them prevents the model from learning noise, improving generalization and stability.

Final Step: Clean Dataset Organization
After filtering out poor-quality, redundant, and irrelevant samples:

The cleaned images and labels were copied into structured directories:

cleaned_train/images, cleaned_train/labelled_images

cleaned_test/images, cleaned_test/labelled_images

This structured, high-quality dataset was then used to train deep learning models like U-Net for damage segmentation.

Summary of Benefits:

1. Improved signal-to-noise ratio in the dataset.
2. Enhanced model learning and generalization.
3. Reduced training time and overfitting.
4. Built a more robust and efficient pipeline for future disaster image analysis.

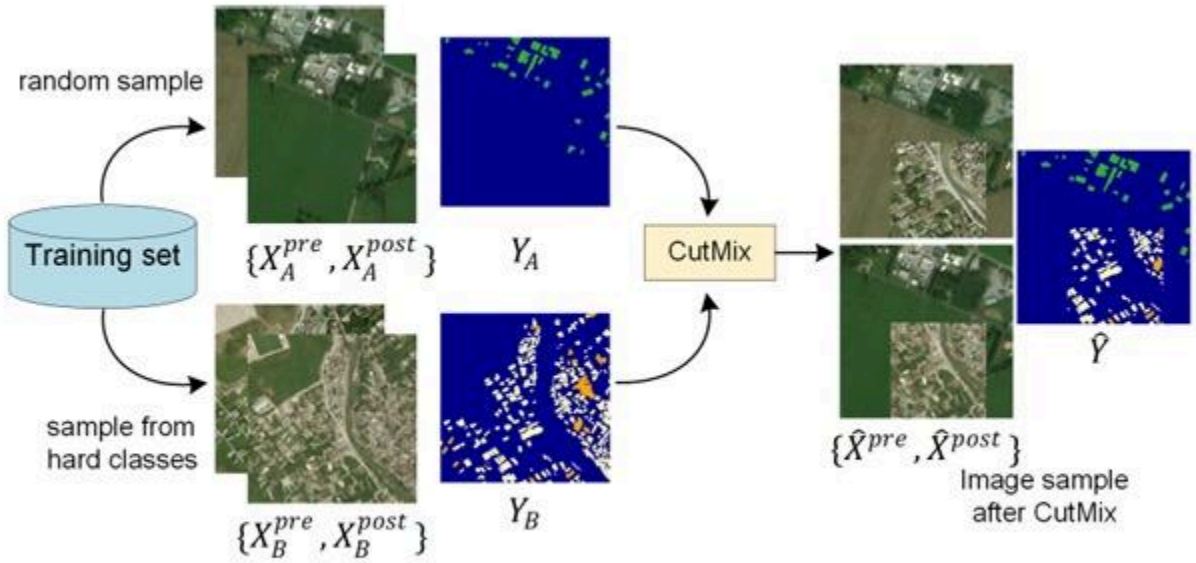
B. Advanced Data Augmentation Using CutMix

What is CutMix?

CutMix is a sample-level data augmentation technique where patches from one image are cut and pasted onto another image. In the context of post-disaster imagery, this method allows the model to learn from composite damage patterns, especially for difficult-to-distinguish classes such as minor and major damage.
Why Use CutMix in Disaster Imagery?
Damage classes in satellite images can be visually similar—for example, minor and major building damage may have overlapping features. To help the model better discriminate between such classes, CutMix increases the number of samples specifically for these "hard" cases, effectively drawing more learning attention to them.

How CutMix Works?

As shown in figure from the referenced source:



Two Samples Selected:

Sample A: Random image pair from the training dataset (X^a_pre, X^a_post) and corresponding mask Y^a.

Sample B: Image pair from a difficult class (e.g., major damage), denoted as (X^b_pre, X^b_post, Y^b).

Patch Selection and CutMix Operation:

A binary mask M is created that defines a rectangular patch area. Using this mask:

A portion of Sample B is inserted into Sample A.

This is done simultaneously for the pre-disaster image, post-disaster image, and their segmentation masks.

Mathematical Definition:

$$\hat{X}^{pre} = M \cdot X_A^{pre} + (1 - M) \cdot X_B^{pre},$$
$$\hat{X}^{post} = M \cdot X_A^{post} + (1 - M) \cdot X_B^{post},$$
$$\hat{Y} = M \cdot Y_A + (1 - M) \cdot Y_B,$$

Here, the · operation represents element-wise multiplication, and M is the binary mask indicating where Sample B's patch will replace content in Sample A.

Result:

A new hybrid image is created (X_pre, X_post) along with its mask (Y).

This image contains a mixture of both contexts, allowing the model to learn transitions and overlaps between different damage levels.

Benefits of CutMix in This Project:

Balances Difficult Classes: By increasing the sample size of damage classes like minor and major, CutMix helps the model not to overfit to dominant classes (e.g., undamaged).

Improves Generalization: The model learns to recognize composite scenes, preparing it to detect multiple damage levels in real-world post-disaster images.

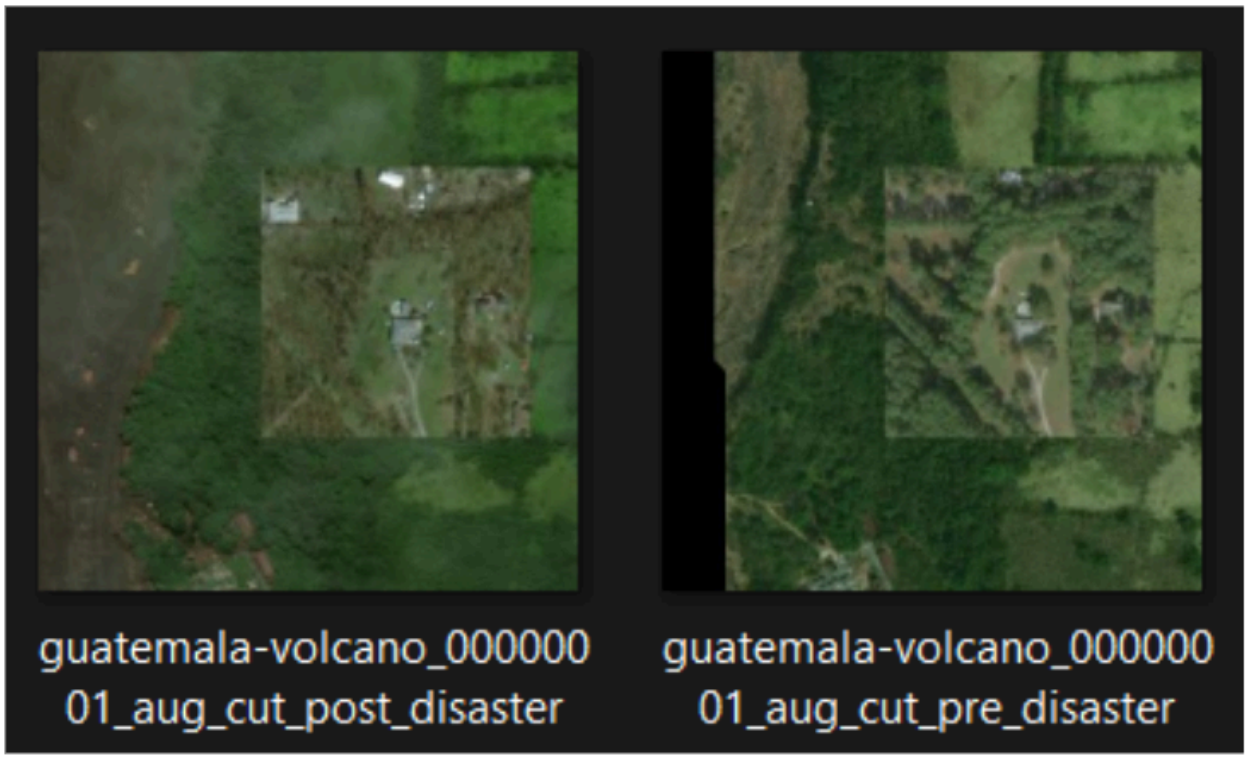
Enhances Robustness: CutMix adds structural and contextual variation to training data, forcing the model to become more invariant to location, structure, and object blending.

Realistic Simulation: Multiple damage zones in a single image mimic complex real-world post-disaster situations better than traditional augmentations.

Implementation Notes:

The CutMix dataset was expanded to 175% of the original size.
75% consisted of selected difficult-class samples (non-white masks). The rest were random samples into which patches were inserted.

Only samples with visible damage were used for mixing to ensure the augmented images always contain meaningful content.



TRAINING MODEL

Model 1
BDANET

1. Overview

BDANet (Building Damage Assessment Network) employs a two-stage approach:

Stage 1 – Building Segmentation: Learns to distinguish building pixels from background, producing a high-resolution binary mask.

Stage 2 – Damage Classification: Uses the segmentation-trained encoder to compare “pre-disaster” and “post-disaster” images and assign each building one of several damage categories.

By first honing in on where buildings are, the model can focus its later capacity on the more subtle task of damage assessment.

2. Model Architecture

2.1 Shared Encoder

Pretrained Convolutional Backbone: A standard deep convolutional network, trained on a large image dataset, provides hierarchical feature maps at multiple scales.

Skip-Connections: Feature maps from early, mid, and late stages are retained to supply spatial detail during upsampling.

2.2 Decoder for Segmentation (Stage 1)

Upsampling Path: Successive upsampling layers restore spatial resolution in steps, each time merging in earlier encoder features to recover fine details.

Final Output: A single-channel probability map, at the original image resolution, indicating the presence of building pixels.

2.3 Dual-Branch Damage Assessment (Stage 2)

Twin Encoders: The same pretrained backbone processes both the pre- and post-disaster images in parallel, sharing weights learned during segmentation.

Cross-Directional Attention: At multiple intermediate feature scales, a learned mechanism emphasizes and exchanges information between the two branches—first reweighting channels to highlight relevant features from one image in the context of the other, then applying a spatial attention mask to focus on critical regions.

Decoder Path: Each branch up samples its attended features in lockstep. After the final attention stage, the two branches' outputs are combined to produce a multi-channel damage prediction map, which is then brought back to full resolution.

3. Training Pipeline

Data Preparation:

For segmentation, each training example consists of one aerial image paired with its ground-truth binary building mask.
For damage classification, pairs of pre- and post-disaster images are labeled with discrete damage categories.

Stage 1 Training (Building Segmentation):

Optimizer and loss function are chosen to penalize discrepancies between the predicted mask and the ground truth.
The model is trained for a fixed number of epochs, with checkpoints saved whenever validation performance improves.

Stage 2 Training (Damage Assessment):

The encoder weights from Stage 1 are loaded to initialize both branches.

A multi-class classification loss drives the model to distinguish damage categories.

As before, training runs for a preset epoch count, saving the best checkpoint according to validation loss.

Validation & Metrics:

Loss Curves: Track training vs. validation loss to monitor overfitting.

F1 Score:

Binary F1 for segmentation, measuring building-pixel accuracy.
Macro-averaged F1 for damage classes, ensuring balanced performance across all categories.

4. Key Hyperparameters & Notes Learning Rates:

A lower learning rate for segmentation, slightly higher for classification to adapt the encoder to the new task.

Batch Size: Kept small (e.g., 2 images per batch) to fit GPU memory when processing high-resolution aerial imagery.

Epoch Count: Chosen to allow convergence without excessive overfitting—commonly 20–30 epochs per stage.

Checkpointing Strategy: Save model weights only when validation loss improves, ensuring the final models generalize best.

Spatial Alignment: Ground-truth masks are resized to exactly match each decoder's output resolution, avoiding interpolation artifacts.

Model 2
Siamese U-Net

Disaster Damage Assessment using Siamese U-Net with Alpha Blending

1. Introduction

Automated disaster damage assessment is a crucial yet challenging task. Manual inspection is time-consuming, error-prone, and unsafe in many situations. This report presents a deep learning approach using a Siamese U-Net architecture with Alpha Blending, capable of comparing pre-disaster and post-disaster imagery for precise pixel-level damage classification.

2. Problem Statement

After a natural disaster, rapid and accurate evaluation of the affected areas is vital for efficient relief efforts. Manual analysis of satellite and aerial imagery is not scalable. While image classification methods exist, segmenting damage by severity at the pixel level adds both complexity and precision. The primary goal is to develop a system that automates this process by classifying damaged structures into five predefined categories using deep learning.

3. Dataset Overview

Title: Dataset Structure and Organization

- Dataset comprises matched triplets of images:
    - Pre-disaster images (baseline state)
    - Post-disaster images (showing damage)
    - Ground truth damage masks (labeled by experts)

Directory structure:

```
/Dataset/

 /train/

   /images/ (contains pre_disaster and post_disaster images)

   /labelled_images/ (contains ground truth masks)

 /test/

   /images/

         /labelled_images/
```

"C:\Users\yasha\OneDrive\Desktop\Nihal\Dataset\train\images\guatemala-volcano_00000000_pre_disaster.png"

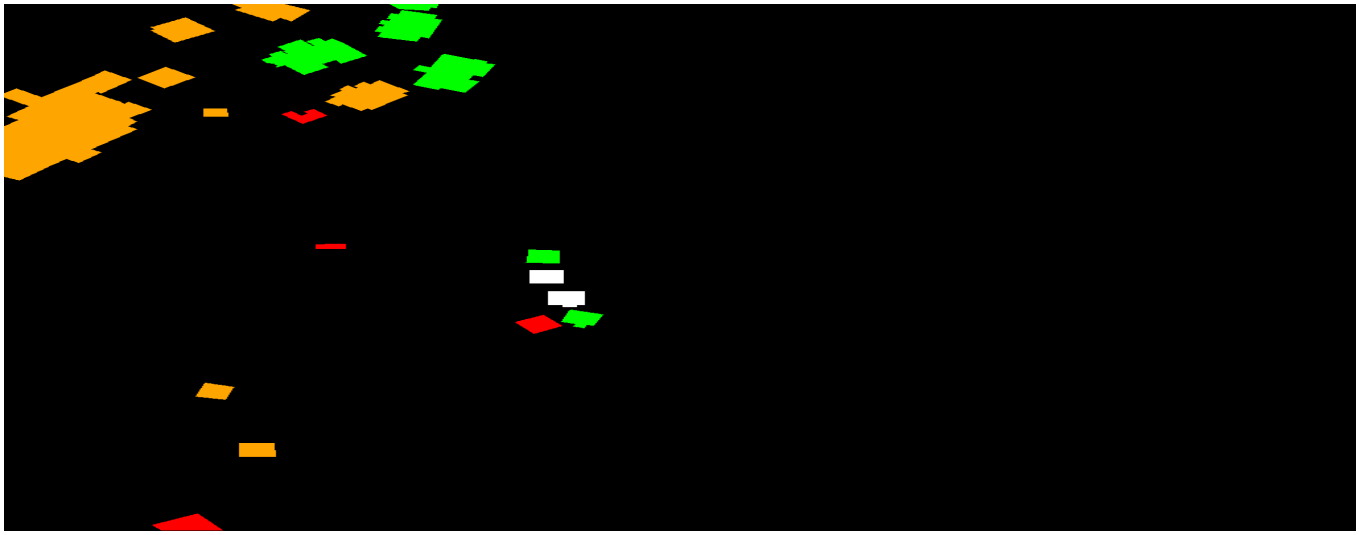
"C:\Users\yasha\OneDrive\Desktop\Nihal\Dataset\train\images\guatemala-volcano_00000000_post_disaster.png"

"C:\Users\yasha\OneDrive\Desktop\Nihal\Dataset\train\images\guatemala-volcano_00000000_labelled_images.png"

4. Damage Classification System

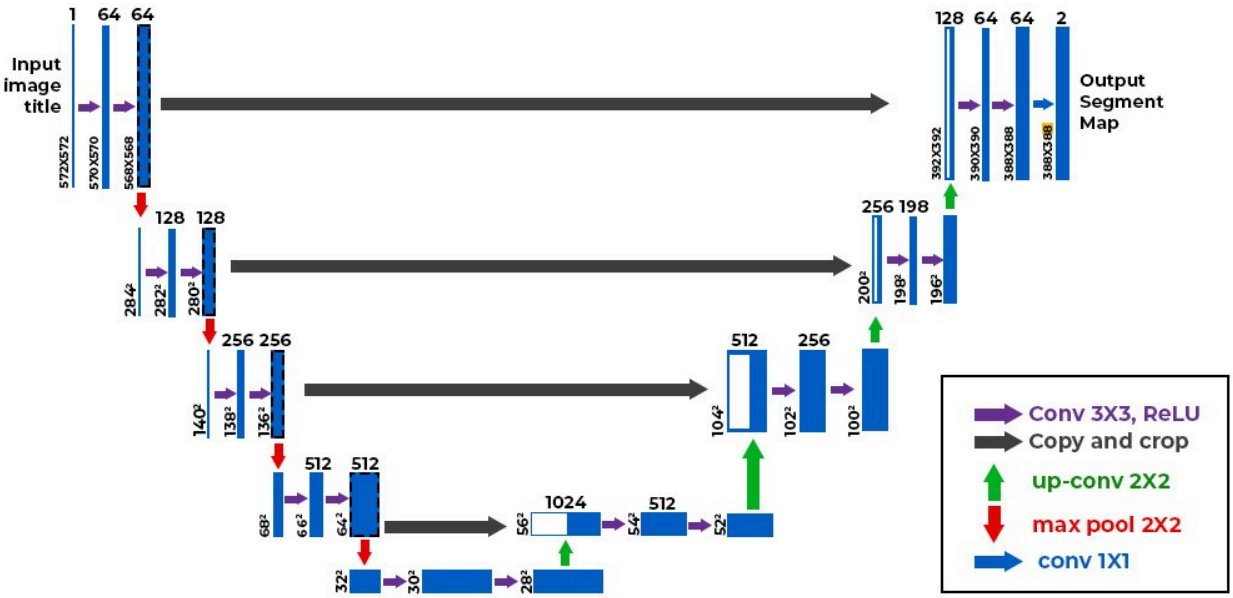
Title: 5-Class Damage Classification System

- Class 0: Background (Black) - Non-target areas or areas outside the region of interest
- Class 1: No Damage (White) - Structures present but unaffected by the disaster
- Class 2: Minor Damage (Green) - Visible damage but structure remains largely intact
- Class 3: Major Damage (Orange) - Significant structural damage but not completely destroyed
- Class 4: Destroyed (Red) - Complete destruction or collapse of structures



Each pixel in the mask is assigned one of these class values.

5. Model Architecture Overview


Input image title
1
64
64
572X572
570X570
568X568
128
64
64
2
392X392
390X390
388X388
388X388
Output Segment Map
128
128
284²
282²
280²
256
198
200²
198²
196²
256
256
140²
138²
136²
512
256
104²
102²
100²
512
512
68²
66²
64²
1024
512
56²
54²
52²
32²
30²
28²
Conv 3X3, ReLU
Copy and crop
up-conv 2X2
max pool 2X2
conv 1X1

The model integrates a Siamese encoder, a U-Net decoder, and an Alpha Blending module:

- Input: RGB image pairs (pre- and post-disaster), resized to 512×512.
- Output: Segmentation map with 5 damage classes.

Key innovations:

- Shared encoder for comparative learning
- U-Net decoder for precise segmentation
- Alpha Blending for feature-level adaptation between input pairs

6. Siamese Neural Network Component

The model leverages Siamese encoders—two parallel but weight-sharing convolutional networks—processing the pre- and post-disaster images simultaneously. This shared structure forces the encoder to focus on differential features indicative of damage.

Encoder (Backbone)

A convolutional classification network (e.g. ResNet, ResNeXt, EfficientNet) truncated before its final pooling or classification layers. As it processes the input, it successively reduces spatial resolution while increasing feature abstraction.

Decoder

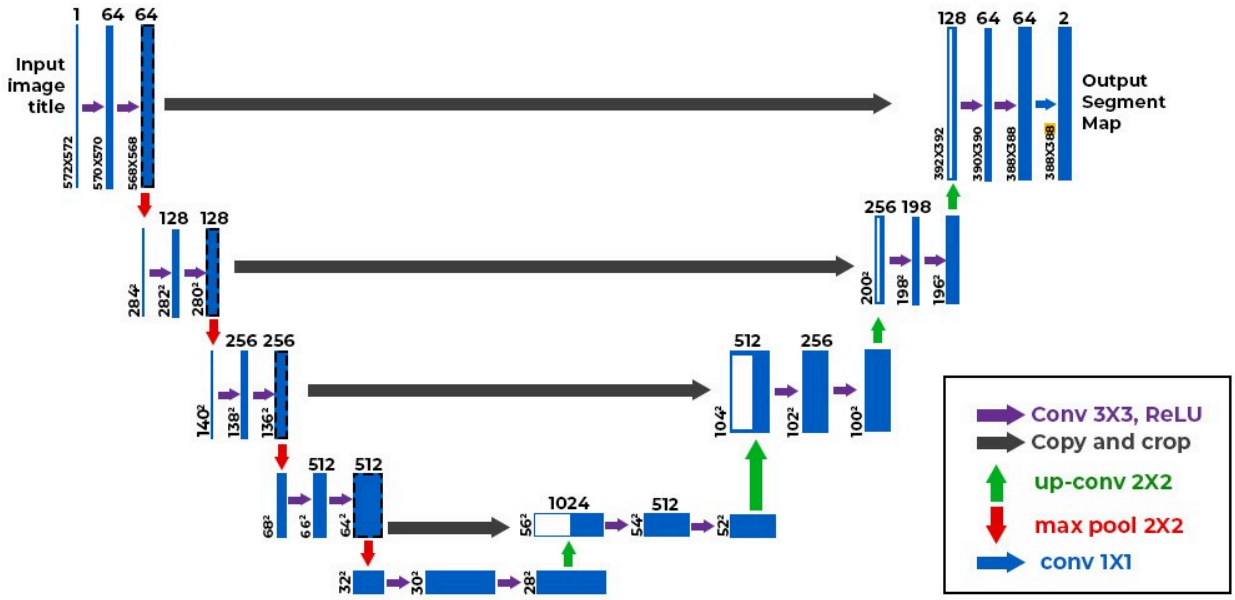
A set of upsampling operations (transposed convolutions, bilinear interpolation + convolution) that restore the spatial resolution, often fusing (“skip-connecting”) encoder features at matching scales to recover fine detail.

pre_features = self.encoder(pre_img)

post_features = self.encoder(post_img)

This approach is efficient and focuses the learning on inter-image differences rather than absolute features.

7. U-Net Architecture



U-Net is widely used in biomedical and satellite imagery segmentation tasks due to its:

- Encoder path: Downsamples input while capturing context
- Decoder path: Upsamples to recover spatial resolution
- Skip connections: Preserve high-resolution features

The model is configured with:

- 3 downsampling levels (depth = 3)
- Base filters = 32
- `DoubleConv`, `Down`, and `Up` modules

- Key components:
  - DoubleConv: Two sequential conv-batchnorm-ReLU blocks
  - Down: Max pooling followed by DoubleConv for downsampling
  - Up: Upsampling followed by DoubleConv, with skip connections
- Modified for disaster assessment with:
  - Depth = 3 for faster training and lighter model

Base filters = 32 (doubling at each level)

This ensures a balance between training efficiency and model expressiveness.

8. Alpha Blending Mechanism

A novel alpha blending module is introduced for feature fusion. At each level of the encoder-decoder bridge, features from pre- and post-disaster branches are blended as follows:

$$\text{blended} = \alpha \times \text{post\_features} + (1 - \alpha) \times \text{pre\_features}$$

- α is a learnable parameter for each layer.
- α is passed through a sigmoid to constrain it to the [0, 1] range.
- The blending enables adaptive focus on either the pre- or post-disaster image based on context.

This module is key to the model's interpretability and performance in change detection.

9. Loss Function Design

The loss function combines Cross-Entropy Loss (CE) and Dice Loss:

- Cross-Entropy: Handles multi-class classification
- Dice Loss: Sensitive to class overlap and handles class imbalance

$$\text{total\_loss} = \text{ce} + \text{dice\_loss}$$

Multi-Class Cross-Entropy Loss (LCE)

Cross-Entropy measures the difference between the predicted probabilities for each class per pixel and the true class label. For multi-class segmentation over all pixels and all classes, the formula is:

$$LCE = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} y_{i,c} \log(\hat{y}_{i,c})$$

Where:

- N: Total number of pixels in the segmentation map.
- C: Number of classes (in your case, C=5).
- i: Index iterating over each pixel.
- c: Index iterating over each class (from 1 to 5).
- $y_{i,c}$: The true label for pixel i belonging to class c. This is 1 if pixel i truly belongs to class c in the ground truth, and 0 otherwise (one-hot encoding).
- $\hat{y}_{i,c}$: The predicted probability that pixel i belongs to class c, typically the output of the softmax activation layer for class c at pixel i.

Multi-Class Dice Loss (LDice)

The Dice Loss is derived from the Dice Coefficient (or F1-score), which measures the overlap between the predicted segmentation and the ground truth. Dice Loss is typically 1−Dice Score. For multi-class, a common approach is to calculate the Dice Score for each class independently and then average them (macro average) or sum them. Using an average:

$$LDice = 1 - \frac{1}{C}\sum_{c=1}^{C}\frac{2\sum_{i=1}^{N} p_{i,c} g_{i,c}}{\sum_{i=1}^{N} p_{i,c} + \sum_{i=1}^{N} g_{i,c} + \epsilon}$$

Where:

- N: Total number of pixels.
- C: Number of classes (5).
- i: Index iterating over each pixel.
- c: Index iterating over each class.

- $p_{i,c}$: The predicted probability (or often, the binary prediction after thresholding the probability) for pixel i belonging to class c.
- $g_{i,c}$: The ground truth binary label for pixel i belonging to class c (1 or 0).
- ϵ: A small smoothing term (a small positive number like 1e-6) added to the denominator to prevent division by zero, especially in cases where a class might be completely absent in a batch.

Class Weights:

- Background: 0.1
- No Damage: 0.5
- Minor Damage: 1.0
- Major Damage: 1.5
- Destroyed: 2.5

This weighted combination ensures balance between categorical accuracy and boundary precision.

10. Training Strategies

To optimize training, several advanced techniques are employed:

- Progressive Resizing: Start with smaller image sizes (256×256) and upscale to 512×512 during training.
- Gradient Accumulation: Enables larger effective batch sizes without increasing GPU memory demand.
- Mixed Precision Training: Reduces memory usage and speeds up training via `torch.cuda.amp`.
- Early Stopping: Halts training if validation loss does not improve for 5 consecutive epochs.

These strategies collectively enhance training stability, especially on limited hardware or small datasets.

11. Data Preprocessing and Augmentation

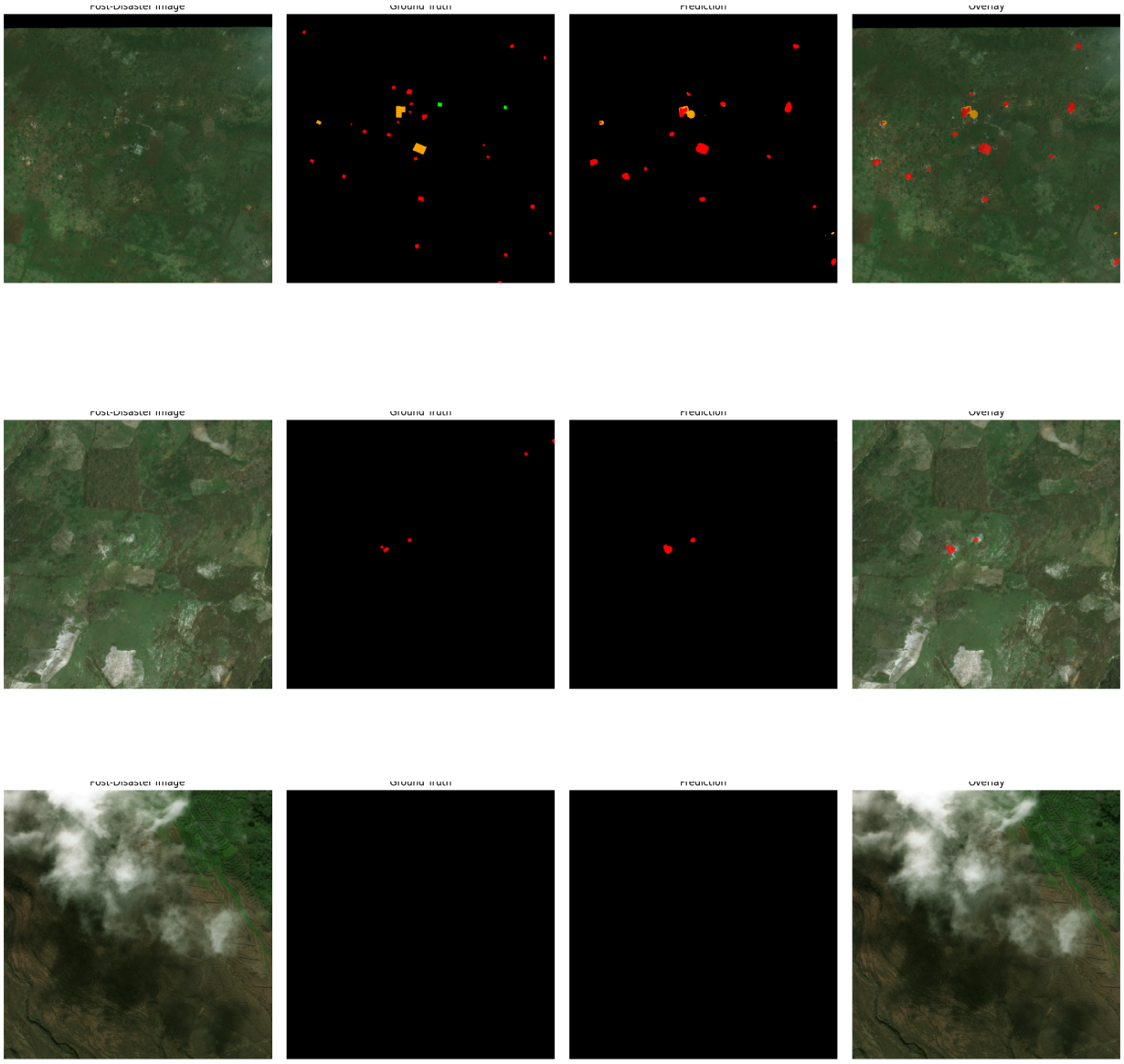
The dataset undergoes multiple preprocessing steps:

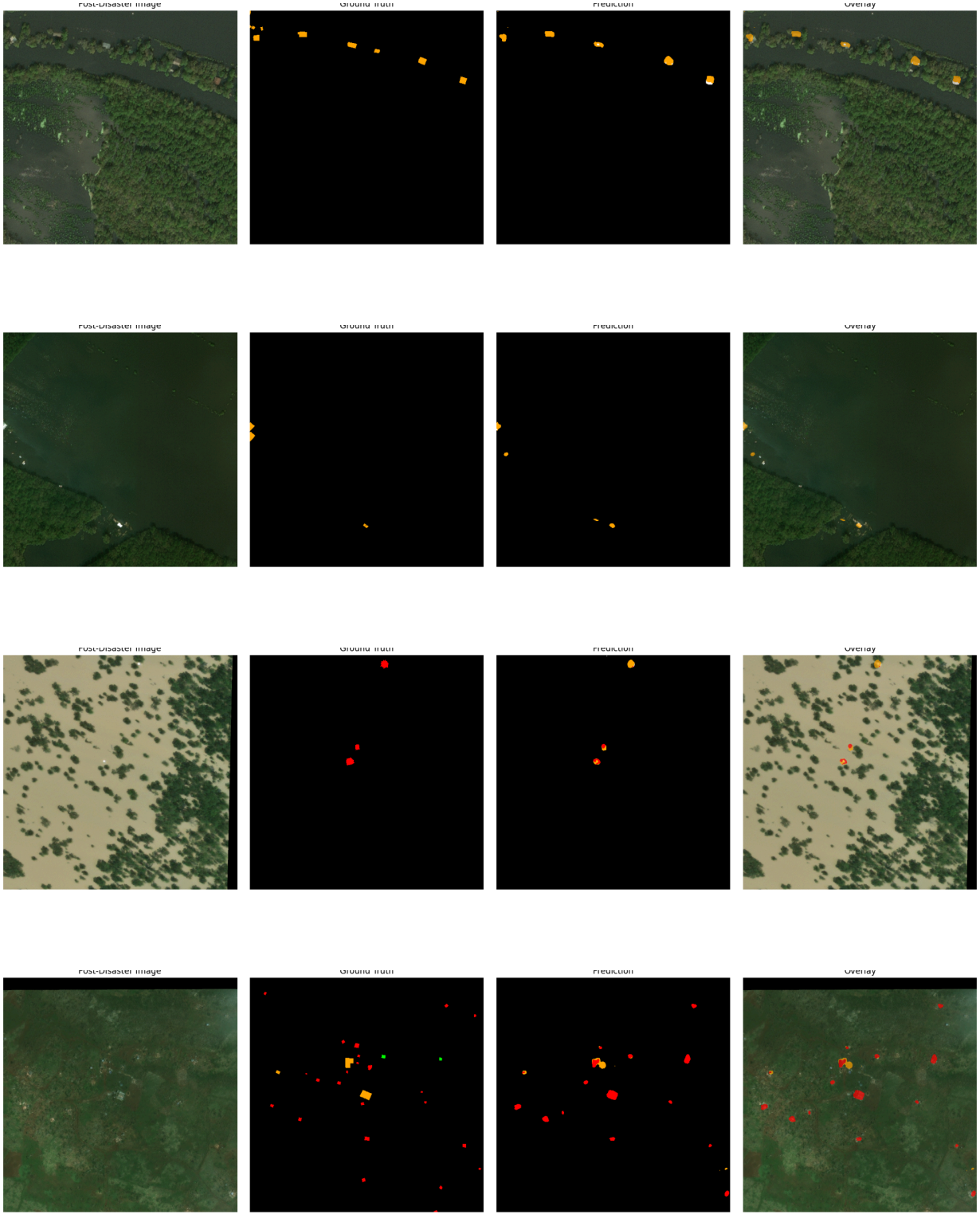
1. Image Pairing: Based on filename pattern
2. Mask Encoding: RGB masks mapped to class indices
3. Resizing: Images resized to target dimensions
4. Augmentation (Albumentations):
    a. Spatial: Flip, rotation, crop, transpose
    b. Photometric: Brightness, contrast, color jitter
5. Normalization: Using ImageNet mean/std for consistency

Different pipelines are applied for training and validation/testing.

12. Visualization of Results

Model outputs are as follows:




Post-Disaster Image
Ground Truth
Prediction
Overlay
Post-Disaster Image
Ground Truth
Prediction
Overlay
Post-Disaster Image
Ground Truth
Prediction
Overlay
Post-Disaster Image
Ground Truth
Prediction
Overlay

- Post-disaster image
- Ground truth damage mask
- Predicted damage mask
- Overlay: Alpha-blended prediction on post-image

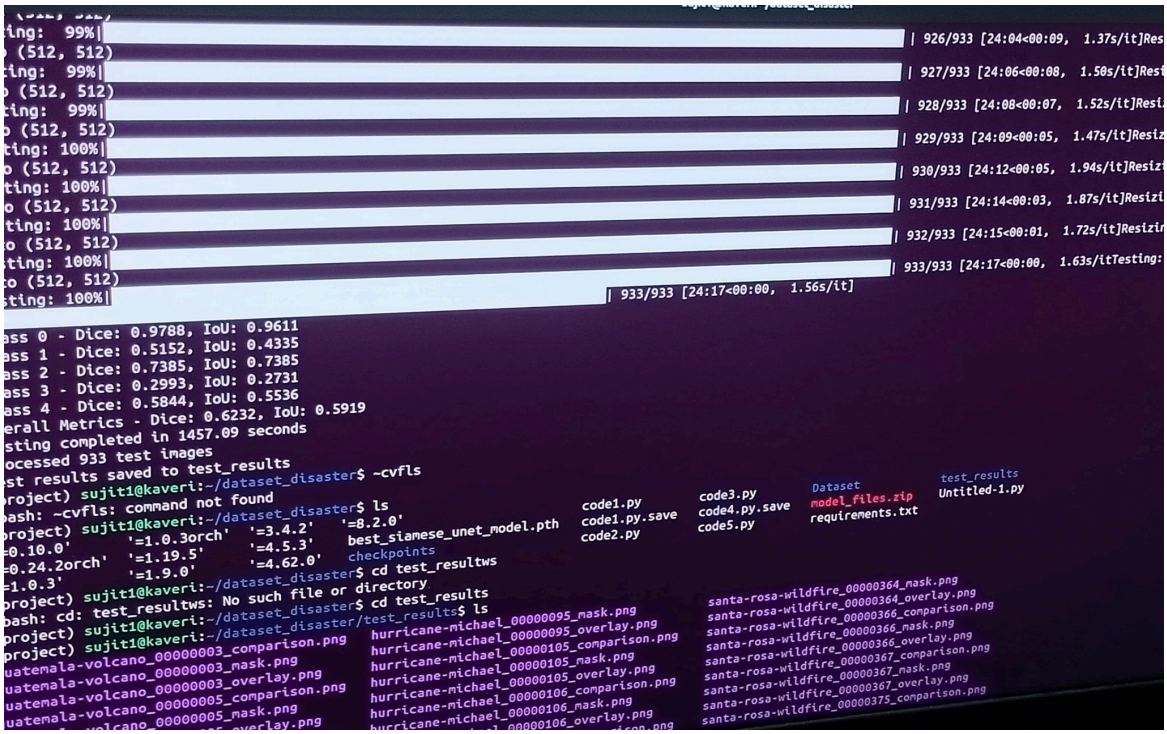
An overlay function helps create clear, interpretable visualizations using the class color codes.

13. Performance Metrics

Evaluation metrics include Dice Coefficient and IoU (Intersection over Union), computed per class and averaged.

Class	Dice Score	IoU Score
0	0.XX	0.XX
1	0.XX	0.XX
2	0.XX	0.XX
3	0.XX	0.XX
4	0.XX	0.XX

- Mean Dice Score: Indicates overall segmentation accuracy.
- Mean IoU: Indicates how well predicted regions overlap ground truth.
- These metrics provide a robust evaluation framework for both general and class-specific performance.



14. Challenges and Solution

Key challenges addressed include

- Small Dataset:

    - Extensive augmentations
    - Progressive resizing
    - Gradient accumulation
- Class Imbalance:
    - Weighted loss functions
    - Emphasis on critical classes like "Destroyed"
- Computational Constraints:
    - Mixed precision training
    - Lightweight U-Net depth (3 levels)
- Long Training Time:
    - Checkpoint system for saving, resuming
    - Best model selection based on validation loss

These engineering solutions enhance training reliability and model robustness.

Key challenges addressed include:

- Small Dataset:
    - Extensive augmentations
    - Progressive resizing
    - Gradient accumulation
- Class Imbalance:

    - Weighted loss functions
    - Emphasis on critical classes like "Destroyed"
- Computational Constraints:
    - Mixed precision training
    - Lightweight U-Net depth (3 levels)
- Long Training Time:
    - Checkpoint system for saving, resuming
    - Best model selection based on validation loss

These engineering solutions enhance training reliability and model robustness.

15. Future Work and Improvements

Potential future directions:

1. Architecture Enhancements:
    - Self-attention modules
    - Transformer encoders for global context
2. Data Strategies:
    - Multimodal fusion (SAR + Optical)
    - Few-shot or weakly-supervised learning
3. Deployment:
    - Model quantization and pruning
    - Integration with real-time response systems
    - Cloud or edge deployment pipelines

These improvements can increase accuracy, generalization, and real-world applicability.

16. Conclusion

The Siamese U-Net with Alpha Blending represents a robust, interpretable, and efficient solution for damage segmentation in disaster-stricken areas. Its innovative design enables pixel-wise classification across five damage classes using comparative learning. With strong generalization and performance, this model holds significant potential for real-time disaster response and resource planning.

Model 3

Encoder–Decoder Segmentation Model

Architecture of the Model

Semantic-segmentation networks are built around an **encoder** that gradually reduces the spatial resolution of the input while increasing the depth (number of feature channels), and a **decoder** that restores the original resolution by upsampling. The encoder is typically a proven classification network (such as ResNet-50) stripped of its final classification layers. It extracts hierarchical features: early layers focus on edges and textures, deeper layers capture object-level semantics. The decoder uses a combination of learned upsampling to recover fine spatial detail.

Core Components

1. Encoder (Backbone)

- Off-the-shelf classification CNNs (ResNet, DenseNet, EfficientNet) pre-trained on ImageNet
- Truncate before final pooling/classifier
- **Role**: extract hierarchical features with growing abstraction

2. Decoder

- Up Samples feature maps back to full resolution
- Common ops:
    - **Transposed convolution** (learnable upsampling)
    - **Interpolation + convolution** (bilinear/nearest upsample then conv)
- Often interleaved with skip-connection fusion

3. Skip Connections

- Preserve spatial detail lost in downsampling
- Fusion via concatenation (U-Net) or addition (FPN)

4. Multi-scale Feature Fusion

- Combines information from multiple depths / receptive fields
- Important for objects of varying sizes

5. Attention & Gating

- **Squeeze-and-Excitation (SE)**: channel-wise weights learned dynamically
- **Spatial attention**: focus on “where” in the image
- **Dual attention**: jointly model inter-channel and inter-spatial dependencies

The model I used for the xView2 Dataset is explained below:

Six-Channel Input Construction

- Stack the pre-disaster RGB and the post-disaster RGB images along the channel axis.

- Results in a single tensor of shape **(6 × 512 × 512)**, allowing the network to learn both spatial appearance and temporal “change” signals simultaneously.

Modified ResNet-50 Encoder

- Based on the standard ResNet-50 architecture, renowned for its strong feature extraction and residual skip connections.

- **First convolution adjusted**: original 3-channel filter replaced with a 6-channel filter (same kernel size and stride) to ingest the combined input.

- **Retain layers up through “layer4”**: produces a deep feature map of shape **(2048 × 16 × 16)** (i.e. 1/32 resolution), embedding rich semantic information about buildings and damage.

Feature Map Characteristics

- **High channel depth (2048)** captures diverse visual patterns—from bare rooftops (no damage) to collapsed structures.

- **Low spatial resolution** (16×16) ensures a large receptive field, enabling the model to recognize context and overall scene layout.

Lightweight Decoder Design

- **Two convolutional blocks**:
    1. Reduce feature depth from 2048 → 512, apply nonlinearity.
    2. Further reduce from 512 → 256, apply nonlinearity.
- **Two fixed upsampling stages**:
    1. Bilinear interpolation ×2 (16 → 32)
    2. Bilinear interpolation ×16 (32 → 512)
- **Final 1×1 convolution** maps 256-channel features to **5 output channels** (logits).

Output Semantics

- **Channel 0**: “building vs. background” confidence score
- **Channels 1–4**: damage severity scores (levels 1, 2, 3, 4) for building pixels
- Softmax across these five channels ensures a valid per-pixel probability distribution.

Design Trade-Offs

- **No intermediate skip-connections** beyond the two upsampling steps simplifies implementation and reduces computational overhead.

- **No attention modules** (e.g. SE blocks) keeps memory footprint low, crucial for GPUs with limited VRAM.

- **Simplicity** fosters faster training, easier debugging, and straightforward extension to more complex variants later.

Benefits of This Setup

- **End-to-end learning**: single network jointly predicts building footprints and damage classes.

- **Efficient inference**: lightweight decoder allows quick predictions even on modest hardware.

- **Extensibility**: architecture can be augmented later with skip-connections, attention, or multi-scale modules without rewriting the core.

Data Augmentation used in this model training:

1. **Geometric**: flips, rotations, scaling, affine transforms
2. **Photometric:** brightness/contrast jitter, noise, blur
3. **Elastic:** grid distortions to simulate deformations
4. **CutMix / MixUp:** combine patches or images to regularize

Inference & Softmax Post-Processing:

1. Obtain Raw Logits:

At each pixel location (y,x)(y,x)(y,x), the vector Z_{y,x} = [z0, z1, z2, z3, z4] contains unnormalized “scores” for:

- z0: background vs. building

- z1 to z4: damage levels 1–4

2. Convert to Probabilities via Softmax

$$p_i = \frac{\exp(z_i)}{\sum_{j=0}^{4} \exp(z_j)}, \quad i = 0, \ldots, 4.$$

Interpretation:

- p0 is the model's confidence that the pixel is not damaged (i.e. background or undamaged).
- p1 to p4 are the confidences for each damage class.

3. We treat the building vs. background decision as a binary mask by thresholding p0

4. **Threshold trade-off**:

   A **lower** threshold (e.g. 0.3) catches more true buildings but may include false positives.

   A **higher** threshold reduces false alarms at the risk of missing small or faint structures.
5. Map each class label to a distinct RGB color (e.g. black for background, red/orange/green/white for levels 1–4), convert masks to 8-bit PNGs with clear filenames, and run inference in small batches—monitoring throughput and memory usage—to produce ready-to-visualize building and damage maps at scale.

Validation Metrics:

Location Dice (Building vs. Background)

Computes the Dice (F1-style overlap) between the predicted building mask (thresholded probability on channel 0) and the ground-truth footprint mask.

Dice = 2·TP / (2·TP + FP + FN), where TP/FP/FN count pixels correctly/incorrectly predicted as building.

Purpose: measures how well the model localizes building footprints, balancing precision and recall equally.

Damage-Level F1 Scores

For each damage class (levels 1–4), within the pixels predicted as building, we compare the argmax of the corresponding probability channels to the true damage labels.

Compute per-class F1 = 2·TP / (2·TP + FP + FN).

Purpose: evaluates how accurately the model distinguishes mild, moderate, and severe damage, treating each class equally.

Aggregate Damage F1

Combine the four per-class F1 scores via their harmonic mean (equivalent to averaging their reciprocals and inverting).

Purpose: produces a single summary metric for the damage-classification task that penalizes poor performance on any individual damage level.

Combined Validation Score

Final score = 0.3 × (Location Dice) + 0.7 × (Aggregate Damage F1).

Purpose: gives more emphasis to correctly classifying damage levels (70 %) while still rewarding accurate footprint localization (30 %).

Evaluation Metrics

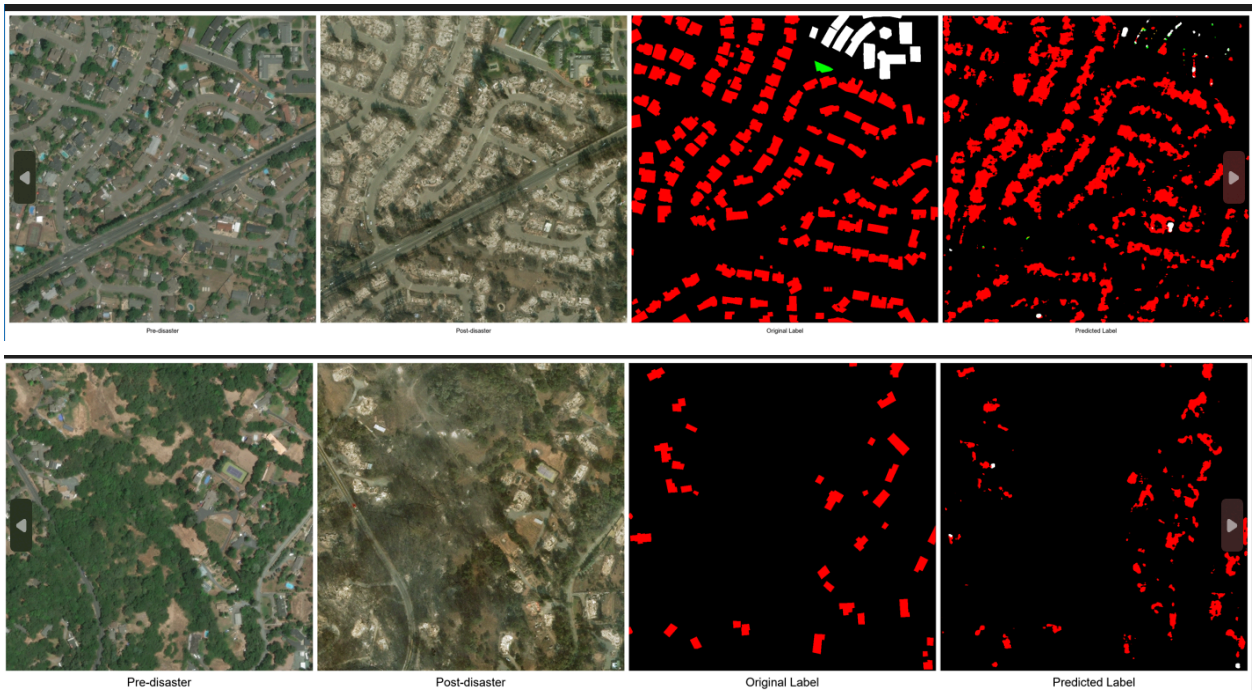
- **Pixel Accuracy:**Percentage of pixels correctly labeled over the entire image. Gives simple overall correctness measure, but can be misleading if one class (like background) dominates.

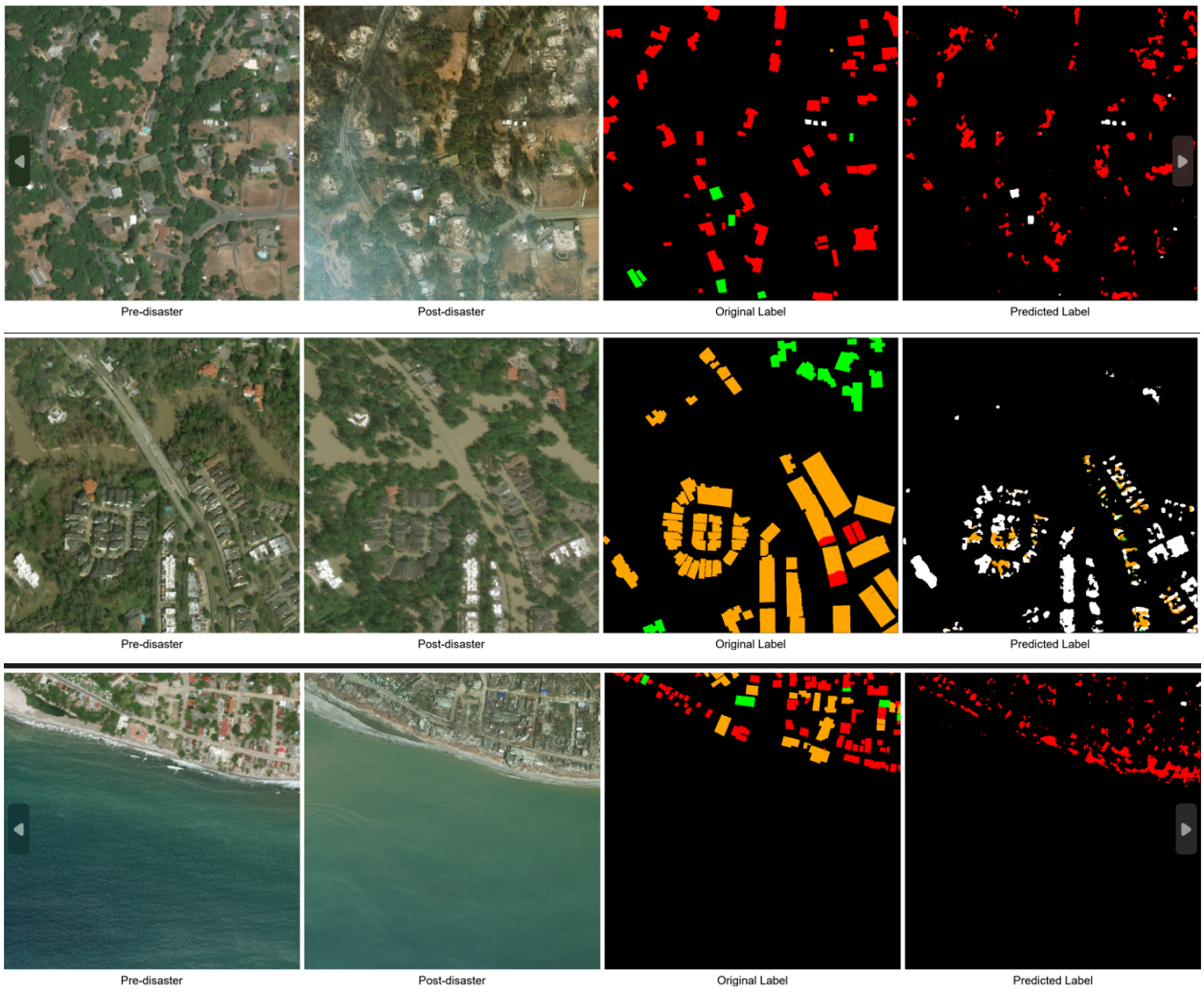
- **Mean IoU (mIoU):** Average of per-class intersection-over-union scores. It balances false positives and negatives at the region level; robust to class imbalance.

- **Dice Coefficient:**Overlap metric akin to the F1 score for binary masks. It emphasizes correct region overlap, making it ideal for evaluating small or thin structures.

- **Per-Class F1 / IoU:**Individual class scores rather than averages. It highlights performance on rare or critical classes that might be masked in overall averages.

Some Results and Conclusion:





- Accurate building footprint localization even under heavy destruction
- Reliable identification of large contiguous severe-damage zones
- In some areas, the network under-segments mild damage (missed orange patches) or mistakenly elevates very small changes to “severe” (red).
- Small or isolated buildings sometimes vanish in predictions—especially when surrounded by heavy smoke or vegetation.

- Mild vs. moderate confusion: the boundary between light (orange) and moderate (green) damage can be blurry, causing occasional mis-classification.

- False positives: tiny speckle of “damage” in entirely intact neighborhoods—likely noise from thresholding or augmentation artifacts.

Threshold tuning, Post-processing are required along with Class-balanced fine-tuning(further oversample mild-damage examples or add a small focal component on level 1 to sharpen that decision boundary.)