Security of Infocommunication Systems and Internet of Things

2026 Vol 4, No 1

https://doi.org/10.31861/sisiot2026.1.01012

Received 27 April 2026; revised 03 June 2026; accepted 15 June 2026; published 30 June 2026

Comparative Analysis of Detection and Segmentation Models for Building Damage Assessment from UAV Imagery

Illia Kravets*, Dmytro Uhryn and Yurii Ushenko

Yuriy Fedkovych Chernivtsi National University, Chernivtsi, Ukraine

*Corresponding author (E-mail: kravets.illia@chnu.edu.ua)

ABSTRACT This paper presents a systematic comparison of object detection models (Faster R-CNN, RetinaNet, YOLOv8) and semantic segmentation models (U-Net, DeepLabV3+, SegFormer) for automated building damage assessment from unmanned aerial vehicle (UAV) imagery. The comparison covers both two-stage (Faster R-CNN) and single-stage (RetinaNet, YOLOv8) approaches to detection, as well as convolutional (U-Net, DeepLabV3+) and transformer-based (SegFormer) segmentation architectures. The RescueNet dataset, a high-resolution post-disaster semantic segmentation benchmark, was adapted for detection by extracting bounding boxes from semantic masks via connected component analysis. Dataset adaptation was performed through separate pipelines for each annotation format (YOLO txt, torchvision JSON), and for segmentation, individual building crops were used, simulating the real-world cascaded pipeline scenario. At the segmentation stage, the task is formulated as binary building-footprint delineation within detected crops; per-building damage-level grading is performed by the subsequent classification stage of the cascade and is beyond the scope of this study. To ensure fair comparison, all models within each task were trained with identical hyperparameters, loss functions, and equivalent pretrained weights (COCO for detection, ImageNet for segmentation). Detection models were evaluated using standard COCO metrics (mAP@0.5, mAP@0.5:0.95) computed via pycocotools, while segmentation models were assessed using globally accumulated IoU, Dice, Precision, and Recall. Inference latency was additionally measured for all models. The results provide an evidence-based foundation for selecting architectures in a cascaded Detection-Segmentation-Classification pipeline for emergency response scenarios. The practical significance of the study lies in the unified evaluation framework that minimizes confounding factors such as different pretraining levels or optimizer configurations, enabling a substantially more direct comparison of architectural properties. This work is part of a broader project on developing an automated UAV-based building damage monitoring system.

KEYWORDS object detection, semantic segmentation, damage assessment, UAV, deep learning.

I. INTRODUCTION

Automated building damage assessment after natural disasters is a critically important task for emergency services, as it enables rapid determination of destruction scale and prioritization of rescue operations. The traditional approach based on ground inspection is slow, resource-intensive, and often dangerous for inspectors. In contrast, the use of unmanned aerial vehicles (UAVs) combined with deep learning methods opens the possibility of automatic analysis of high-resolution aerial photographs in near real-time [1, 2].

The damage assessment task is multi-stage: the first step requires localizing buildings in the image (detection), and the second step involves determining the degree of damage to each building (classification or segmentation). In previous work [3], a cascaded architecture "YOLOv8 - U-Net - EfficientNet" was proposed, which sequentially performs detection, segmentation of damaged areas, and classification of destruction degree. However, the selection of specific models for each cascade stage requires experimental justification on real data.

In this work, a systematic comparison of three detection models (Faster R-CNN [4], RetinaNet [5], YOLOv8 [6]) and three semantic segmentation models (U-Net [7], DeepLabV3+ [8], SegFormer [9]) is performed on the RescueNet dataset [1], adapted for detection by converting semantic masks to bounding boxes. Within the evaluated two stages of the cascade, segmentation is formulated as binary building-footprint delineation on per-building crops; assessment of distinct damage levels is performed by the third (classification) stage and is not evaluated in this work. The key requirement for comparison is ensuring identical starting conditions for all models within each task: an equivalent level of pretraining (COCO weights for all detectors; ImageNet-only encoder pretraining with randomly initialized decoders for all segmentation models), identical loss functions, optimizers, and metrics.

II. RELATED WORK

The task of automated building damage assessment is actively studied in the context of remote sensing analysis and computer vision. The xBD dataset [2], containing over 850,000 building annotations on satellite images before and after natural disasters, has become a benchmark for change detection tasks. However, satellite images have limited resolution, which complicates detection of minor damage. The RescueNet dataset [1], created from UAV images after Hurricane Michael, provides significantly higher resolution and pixel-wise annotation of 10 classes, including 4 building classes with different damage levels.

In object detection, the Faster R-CNN architecture [4] remains the foundational two-stage model that uses a Region Proposal Network (RPN) for hypothesis generation and ROI pooling for refinement. The architecture consists of two main components: RPN scans the feature map obtained from the backbone (ResNet-50) using a set of anchor boxes of different scales and proportions, predicting object probability and refining box coordinates for each anchor. In the second stage, proposed regions pass through ROI Align, after which they are classified and refined again. The key advantage of Faster R-CNN is high detection accuracy, especially for medium and large objects [4, 10]. Among the disadvantages is relatively low inference speed compared to single-stage detectors, as the two-stage process is computationally expensive [5]

$$L_{RPN} = \frac{1}{N_{cls}}\sum L_{cls}(p_i, p_{i*}) + \lambda \cdot \frac{1}{N_{reg}}\sum p_{i*} \cdot L_{reg}(t_i, t_{i*}), \quad (1)$$

where p_i is the predicted probability of anchor i being an object; p_{i*} is the ground truth label; t_i are the predicted bounding box parameters; L_{cls} is binary cross-entropy loss; L_{reg} is smooth L1 regression loss; λ is the balancing weight.

RetinaNet [5] is a single-stage detector introduced by Lin et al. that addresses the fundamental problem of single-stage architectures - extreme imbalance between positive and negative samples. The main innovation of RetinaNet is Focal Loss, a modified cross-entropy loss function that dynamically reduces the contribution of easily classified negative samples (background) and focuses training on hard examples. The architecture uses Feature Pyramid Network (FPN) [11] for building a multi-scale feature pyramid. Advantages include significantly higher inference speed compared to two-stage detectors while maintaining competitive accuracy [5], and effective performance under strong class imbalance typical for aerial image detection [10]. Among the limitations is the use of predefined anchor boxes requiring careful tuning for specific domains

$$\mathrm{FL}(p_t) = -\alpha_t(1-p_t)^{\gamma}\log(p_t), \quad (2)$$

where p_t is the estimated probability for the correct class; γ is the focusing parameter (γ=2); α_t is the class balancing factor.

YOLOv8 [6] represents the latest generation of the You Only Look Once family, one of the most popular approaches to real-time object detection. Unlike previous YOLO versions, YOLOv8 uses an anchor-free approach, eliminating the need for manual anchor box tuning. The architecture consists of three main blocks: CSPDarknet backbone for feature extraction, Path Aggregation Network (PAN) neck for multi-scale feature aggregation, and decoupled head for independent class and box coordinate prediction. Key advantages include the highest inference speed among compared models, making it suitable for deployment on edge devices (UAVs, Jetson Nano) [6, 12]. Among the limitations is somewhat lower accuracy on small objects compared to two-stage detectors [13].

$$L = \lambda_{box} \cdot L_{CIoU} + \lambda_{cls} \cdot L_{BCE} + \lambda_{dfl} \cdot L_{DFL}, \quad (3)$$

where L_{CIoU} is Complete IoU loss for box regression; L_{BCE} is binary cross-entropy for classification; L_{DFL} is Distribution Focal Loss for localization.

In semantic segmentation, U-Net [7] is a classic encoder-decoder architecture with skip connections introduced by Ronneberger et al. The encoder sequentially reduces spatial resolution through convolutional blocks and max pooling, extracting hierarchical features. The decoder restores spatial resolution through transposed convolutions. Skip connections transfer feature maps from corresponding encoder levels directly to the decoder via concatenation, enabling use of both high-level semantic features and low-level spatial details [7]. Advantages include relative architectural simplicity and low computational requirements. Among the disadvantages is a limited receptive field [8]

$$x^l_{\mathrm{dec}} = \mathrm{Conv}\left(\mathrm{Concat}\left(\mathrm{Up}\left(x^{l+1}_{\mathrm{dec}}\right), x^l_{\mathrm{enc}}\right)\right), \quad (4)$$

where x_{enc} is the encoder feature map at level l; x_{dec} is the decoder feature map; l is the network depth level.

DeepLabV3+ [8], introduced by Chen et al., features the Atrous Spatial Pyramid Pooling (ASPP) module that applies several parallel atrous (dilated) convolutions with different dilation rates. Atrous convolution increases the receptive field without reducing spatial resolution and without increasing parameter count. Unlike DeepLabV3, the V3+ version adds a specialized decoder that gradually restores spatial resolution. Advantages include effective multi-scale context capture through ASPP [14] and precise object boundary localization. Among the limitations are higher computational requirements compared to U-Net

$$y = Conv_{1\times1}(x) \oplus \oplus_{i=1}^{3} AConv_{r_i}(x) \oplus \mathrm{GAP}(x), \quad (5)$$

where r_i are dilation rates {6, 12, 18}.

SegFormer [9], introduced by Xie et al. (2021), represents a fundamentally different approach based on transformers. The encoder is built on hierarchical Mix Transformer (MiT), which generates multi-scale features without positional encoding, using Efficient Self-Attention with sequence reduction. The decoder is an extremely simple All-MLP module. Advantages include a global receptive field through self-attention [9] and architecture scalability through B0-B5 variants. The B0 variant (3.7M parameters) is the lightest, suitable for edge deployment. Among the limitations is the need for large data volumes for effective transformer training [13], and outputs have stride 4, requiring interpolation

$$\mathrm{Attention}(Q,K,V) = \mathrm{Softmax}\left(\frac{Q\cdot K^T}{\sqrt{d_{head}}}\right)\cdot V, \quad (6)$$

where Q is the query matrix; K is the key matrix; V is the value matrix; d_{head} is the attention head dimension.

Most existing comparative studies have methodological limitations: models are compared using different pretraining levels, optimizers, learning rate schedules, and datasets, making it impossible to isolate architectural differences from confounding factors. This work addresses this limitation by implementing a unified evaluation framework where all models within each task use identical hyperparameters, loss functions, and pretraining levels.

III. PROBLEM STATEMENT

The goal of this research is a comparative analysis of the effectiveness of detection and segmentation models for automated building damage assessment from UAV imagery. The specific research objectives are:

- Adapt the RescueNet dataset for the detection task by converting semantic masks to bounding boxes using connected component analysis;
- Ensure unified training conditions for all models within each task (identical pretrained weights, loss functions, hyperparameters, and optimization strategies);
- Conduct training and evaluation of three detection models (Faster R-CNN, RetinaNet, YOLOv8) using standard COCO metrics (mAP@0.5, mAP@0.5:0.95);
- Conduct training and evaluation of three segmentation models (U-Net, DeepLabV3+, SegFormer) using unified metrics (IoU, Dice coefficient, Precision, Recall);
- Measure inference latency and justify the selection of models for the cascaded architecture.

IV. METHODOLOGY

A. Dataset and data preparation. The RescueNet dataset [1] is one of the most comprehensive publicly available post-disaster aerial image collections and a high-resolution semantic segmentation benchmark specifically designed for post-disaster scene understanding from UAV imagery. The dataset was collected after Hurricane Harvey (2017) in the Texas coastal region using small unmanned aerial vehicles at low altitudes (approximately 50 – 120 m), providing ground sampling distances of approximately 3 – 6 cm per pixel. Each image has a resolution of 4000 × 3000 pixels, capturing detailed structural features of buildings, roads, and vegetation.

RescueNet provides dense pixel-level annotations across 10 semantic classes: background (0), water (1), building with no damage (2), building with minor damage (3), building with major damage (4), building with total destruction (5), vehicle (6), clear road (7), blocked road (8), and tree (9). This multi-class annotation scheme enables both object-level detection and pixel-level segmentation tasks. The four building damage classes (2-5) are of primary interest for the cascaded damage assessment pipeline investigated in this work. Fig. 1 illustrates representative examples from the dataset showing UAV imagery alongside damage class overlays highlighting affected buildings. These visualizations reveal the spatial heterogeneity of damage patterns within individual scenes, where buildings at varying damage levels frequently co-occur in close proximity, necessitating models capable of fine-grained discrimination between damage severity categories at the individual building level.

Several alternative datasets exist for building damage assessment, including xBD [2] (satellite imagery, pre/post-disaster pairs) and ISBDA (aerial imagery). RescueNet was selected for this study due to three key advantages: (1) UAV-level resolution provides significantly finer spatial detail compared to satellite-based datasets, enabling detection of subtle structural damage; (2) the multi-class damage taxonomy (no damage, minor, major, total destruction) directly supports graduated damage assessment; (3) the semantic segmentation format allows flexible adaptation to both detection and segmentation tasks through the mask conversion methodology described below.

FIG. 1. Representative samples from RescueNet dataset: UAV RGB imagery (left) and corresponding building damage overlays (right) highlighting three damage classes directly on the imagery.

Adapting a semantic segmentation dataset for object detection requires converting pixel-wise masks into instance-level bounding boxes. Two principal approaches were investigated: (a) direct contour extraction using morphological operations, and (b) connected component analysis with per-class labeling. Preliminary experiments revealed that approach (a) produced fragmented detections for adjacent buildings sharing walls, while approach (b) with per-class component extraction correctly separated buildings of different damage levels even when spatially adjacent. Consequently, per-class connected component analysis was adopted as the primary conversion methodology.

The conversion pipeline operates as follows: for each semantic mask, binary masks are extracted for each building damage class (2-5) independently. Connected component analysis (8-connectivity) is applied to each binary mask, identifying individual building instances. For each connected component, the minimum enclosing bounding rectangle is computed, yielding coordinates in [x_min, y_min, x_max, y_max] format. Components with area below 100 pixels are filtered as annotation noise or partial building fragments. Fig. 2 demonstrates this process: from the original UAV image, through the semantic mask, to extracted bounding boxes with damage class labels.

FIG. 2. Bounding box extraction process: per-class connected component analysis identifies individual buildings and generates bounding rectangles with damage class labels.

Separate annotation format converters were implemented to accommodate the different input requirements of detection frameworks: torchvision-based models (Faster R-CNN, RetinaNet) require JSON annotations with absolute pixel coordinates, while Ultralytics YOLOv8 expects normalized coordinates in [class, x_center, y_center, width, height] format. Identical filtering parameters (minimum component area, aspect ratio thresholds) were applied across both pipelines to ensure equivalent dataset composition regardless of annotation format.

For segmentation model training, individual building crops were extracted from original images using the computed bounding boxes, with a 10-pixel contextual margin. Corresponding binary segmentation masks were generated by isolating building pixels within each crop region. This approach emulates the operational cascade where detection model outputs serve as inputs for per-building segmentation. Fig. 3 presents examples of extracted building crops with their corresponding binary segmentation masks. The crop extraction procedure preserves the original spatial resolution, ensuring that fine structural details such as roof edges, wall boundaries, and partial collapse indicators remain discernible for the segmentation stage. Binary masks encode building footprint pixels as foreground against a non-building background, enabling pixel-wise damage boundary delineation.

The adapted detection dataset comprises 764 training images containing approximately 8,500 building instances and 191 validation images with approximately 1,200 instances. Class distribution analysis revealed: buildings with no damage approximately 35%, minor damage approximately 25%, major damage approximately 25%, and total destruction approximately 15%. This moderate class imbalance motivated the inclusion of RetinaNet with Focal Loss in the model comparison, as it was specifically designed for addressing class imbalance in dense detection tasks. The segmentation dataset contains 4,766 training and 688 validation building crop-mask pairs derived from the same image set. Fig. 4 presents the overall proposed

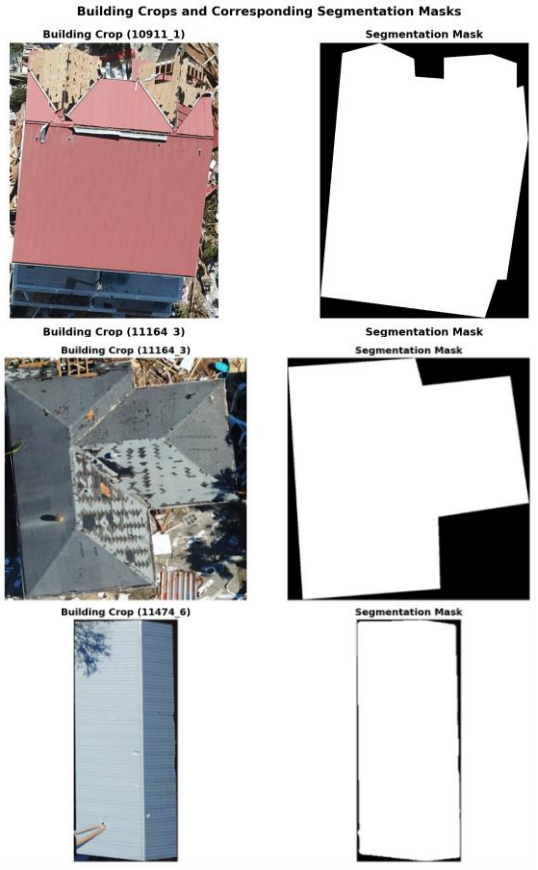

FIG. 3. Individual building crops (left) extracted from UAV imagery with corresponding binary segmentation masks (right) used for segmentation model training.

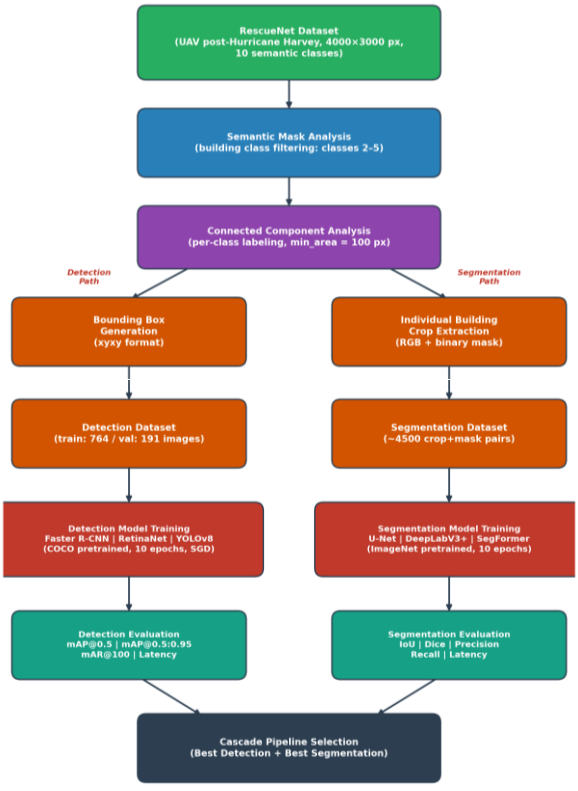

FIG. 4. Proposed end-to-end methodology pipeline for building damage assessment from RescueNet dataset through preprocessing to model evaluation and cascade selection.

methodology pipeline, which follows a systematic three-stage workflow: data preparation, model training with evaluation, and cascaded inference. The pipeline is designed to be architecture-agnostic, allowing straightforward substitution of detection and segmentation backbones for comparative analysis under identical preprocessing and evaluation conditions.

B. Detection models training. For training Faster R-CNN and RetinaNet, the torchvision library implementation was used with pretrained ResNet-50 backbone initialized with COCO weights. Faster R-CNN utilized a two-stage architecture with RPN and region-wise classification head. RetinaNet used the single-stage FPN-based architecture with Focal Loss (alpha=0.25, gamma=2.0). For both detectors, identical SGD optimizer (learning rate=5e-3, momentum=0.9, weight_decay=5e-4), StepLR scheduler (step_size=7, gamma=0.1), 20 epochs, batch_size=2, seed=42 were used.

YOLOv8n (nano variant with ~3.2M parameters) from Ultralytics was initialized with COCO pretrained weights. The architecture uses anchor-free approach with decoupled detection head. Training employed identical hyperparameters: SGD optimizer (learning rate=5e-3, momentum=0.9, weight_decay=5e-4), StepLR scheduler (step_size=7, gamma=0.1), 20 epochs, batch_size=2, seed=42. All training scripts utilize the shared common_utils.py module that fixes the seed for PyTorch, NumPy, and CUDA to ensure reproducibility.

C. Segmentation models training. U-Net and DeepLabV3+ were implemented through the segmentation_models_pytorch (smp) library [15] with pretrained ResNet-50 encoder (ImageNet). SegFormer-B0 was initialized from the nvidia/mit-b0 checkpoint, whose hierarchical Mix Transformer encoder is pretrained exclusively on ImageNet-1k image classification, with a randomly initialized decoder head producing 1 output channel. This ensures an equivalent level of pretraining across all three models: every encoder carries only ImageNet classification knowledge, and every decoder is initialized randomly, so no model benefits from prior exposure to a semantic segmentation task. SegFormer outputs have stride 4, so logits are interpolated to input size via bilinear interpolation before loss computation.

For all three segmentation models, identical BCE+Dice loss function (bce_weight=0.5, dice_weight=0.5), AdamW optimizer (learning rate=1e-4, weight_decay=1e-4), batch_size=4, seed=42, image_size=256, and ImageNet normalization were used. Training was run for a maximum of 100 epochs under a unified stopping protocol: a ReduceLROnPlateau scheduler (factor 0.5, patience 5, monitoring validation Dice) combined with early stopping after 15 epochs without improvement of validation Dice. Identical augmentations (horizontal and vertical flips with p=0.5, random rotation within ±10°, and random brightness/contrast adjustments) were applied during training. All segmentation models were trained and benchmarked in the Google Colab environment on an NVIDIA Tesla T4 GPU (16 GB VRAM), Intel Xeon CPU @ 2.00 GHz, 12 GB RAM, PyTorch 2.11 with CUDA 12.8. Metrics are computed globally through the SegmentationMetrics class that accumulates TP, FP, and FN across the entire validation set, rather than averaging per batch.

D. Evaluation metrics. For detection, standard COCO metrics computed via pycocotools were used: mAP@0.5, mAP@0.5:0.95, and mAR@100. For segmentation: IoU, Dice coefficient, Precision, Recall, and F1-score. Additionally, inference latency (ms/image) and throughput (FPS) were measured for all models

$$\text{Dice} = \frac{2 \cdot TP}{2 \cdot TP + FP + FN}, \text{IoU} = \frac{TP}{TP + FP + FN}, \quad (7)$$

where TP is true positives; FP is false positives; FN is false negatives.

V. EXPERIMENTAL WORKFLOW

A. Dataset adaptation for detection. The RescueNet semantic segmentation masks were processed to extract building instances. The connected components algorithm identified individual buildings in each mask, computing bounding rectangles for each component. Filtering removed components smaller than 100 pixels (noise) and boxes with extreme aspect ratios. For YOLO format, coordinates were normalized; for torchvision format, absolute pixel coordinates were preserved. The same filtering criteria were applied to both formats to ensure consistent dataset composition.

Statistics from the adapted dataset: training set contained approximately 8,500 building instances, validation set contained approximately 1,200 instances. Distribution across building damage classes: no damage (class 2) ~35%, minor damage (class 3) ~25%, major damage (class 4) ~25%, total destruction (class 5) ~15%. This class imbalance motivated the use of RetinaNet with Focal Loss, which is specifically designed for imbalanced datasets.

B. Dataset preparation for segmentation. For segmentation model training, the detected building bounding boxes were used to extract individual crops from the original images. Each crop included the building area plus 10-pixel margin for context. Crops were resized to 256x256 pixels using bilinear interpolation. Corresponding segmentation masks were also cropped and resized. This approach simulates the real cascaded pipeline where a pre-trained detection model provides bounding boxes that are then passed to segmentation models.

The resulting training dataset for segmentation consisted of 4,766 image crops with corresponding binary masks; the validation set contained 688 crops. Binary masks were created by combining all building classes (2, 3, 4, 5) into a single positive class. The segmentation task is therefore formulated as binary building-footprint delineation within each detected crop; grading of individual damage levels is not performed at this stage and is delegated to the subsequent classification stage of the cascaded pipeline.

C. Detection model training. All three detection models (Faster R-CNN, RetinaNet, YOLOv8n) were trained on the adapted RescueNet detection dataset using identical hyperparameters to ensure fair comparison. Training was conducted in the Google Colab environment on an NVIDIA Tesla T4 GPU (16 GB VRAM) with

batch_size=2 due to memory constraints; all detection latency measurements reported below were obtained on the same hardware. Learning rate schedule (StepLR with step_size=7, gamma=0.1) was designed to reduce learning rate after 7 epochs, allowing models to converge and refine learned features. Training was conducted for a fixed budget of 20 epochs, identical for all three detectors.

Validation was performed every epoch on the validation set using COCO evaluation protocol. Detection results were filtered by confidence threshold (0.5) before metric computation. This approach simulates practical deployment conditions where only high-confidence detections are processed by downstream stages.

D. Segmentation Model Training. Segmentation models were trained on the extracted building crops using identical hyperparameters: BCE+Dice loss, AdamW optimizer, batch_size=4, learning_rate=1e-4, with a maximum budget of 100 epochs governed by the unified early stopping protocol described in Section IV. Training images were augmented using standard techniques: horizontal and vertical flips (p=0.5 each), random rotation (±10 degrees), and random brightness/contrast adjustments. Validation was performed every epoch with metrics accumulated globally across the validation set.

The unified stopping protocol ensured that every model was trained to convergence: U-Net reached its best validation Dice at epoch 41 (training stopped at epoch 56), DeepLabV3+ at epoch 53 (stopped at 68), and SegFormer-B0 at epoch 79 (stopped at 94). The corresponding training and validation curves are presented in Fig. 11, confirming that no model was terminated prematurely. The global metrics accumulation approach (TP/FP/FN pooled across validation set) better represents real-world performance than per-batch metric averaging, as building-level metrics require aggregation of predictions across the entire validation set.

VI. EXPERIMENTAL RESULTS

Table 1 presents detection model comparison results on the RescueNet validation set.

TABLE 1. Comparison of detection models on RescueNet val.

Model	mAP@0.5	mAP@.5:.95	mAR@100	ms/img
Faster R-CNN	86.4	62.3	71.4	305.4
RetinaNet	83.8	62.8	75.5	214.5
YOLOv8n	90.9	73.8	85.9	~7

The detection results reveal clear performance differences among the three models. YOLOv8n achieves the highest mAP@0.5 of 90.9%, followed by Faster R-CNN at 86.4% and RetinaNet at 83.8%. For the stricter mAP@0.5:0.95 metric, YOLOv8n leads with 73.8%, Faster R-CNN achieves 62.3%, and RetinaNet 62.8%. Faster R-CNN and RetinaNet show comparable mAR@100 (71.4% and 75.5%, respectively), while YOLOv8n leads with 85.9%. The most striking difference lies in inference latency: YOLOv8n processes images in approximately 7 ms, whereas Faster R-CNN requires 305.4 ms and RetinaNet 214.5 ms. This makes YOLOv8n approximately 43x faster than Faster R-CNN and 30x faster than RetinaNet. Despite Faster R-CNN achieving competitive accuracy metrics, its two-stage architecture introduces significant computational overhead unsuitable for real-time deployment. These results strongly support selecting YOLOv8n as the detection component in the cascaded pipeline, especially for deployment on edge devices where real-time processing is critical. Fig. 5 presents a visual comparison of detection model metrics. Fig. 6 further illustrates qualitative detection outputs from YOLOv8n on validation images, demonstrating tight bounding box alignment with ground-truth annotations across buildings of varying sizes and damage categories. The predicted confidence scores consistently exceed 0.7 for correctly localized instances, corroborating the high mAP@0.5 of 90.9%.

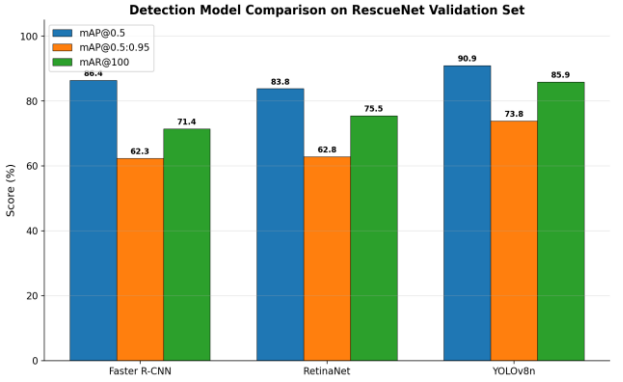

FIG. 5. Comparison of detection model metrics on RescueNet validation set.

FIG. 6. YOLOv8n detection predictions on RescueNet validation set with confidence scores.

TABLE 2. Comparison of segmentation models on RescueNet val.

Model	IoU	Dice	Prec.	Recall	F1	ms/img
U-Net	92.5	96.1	95.5	96.7	96.1	8.5
DeepLabV3+	93.0	96.4	95.6	97.1	96.4	10.9
SegFormer-B0	93.2	96.5	96.1	97.0	96.5	8.8

The segmentation results show that under the unified training protocol all three models converge to closely comparable quality. SegFormer-B0 achieves the best overall performance with IoU of 93.2%, Dice of 96.5%, and balanced Precision/Recall (96.1%/97.0%), while requiring approximately 8.7x fewer parameters (3.7M) than U-Net (32.5M) and 7.2x fewer than DeepLabV3+ (26.7M). DeepLabV3+ follows closely with IoU of 93.0% and Dice of 96.4% (Precision 95.6%, Recall 97.1%), and U-Net achieves IoU of 92.5% and Dice of 96.1% (Precision 95.5%,

Recall 96.7%). The learning curves in Fig. 11 confirm that all three models were trained to convergence, with SegFormer-B0 requiring the longest schedule (best epoch 79) and U-Net the shortest (best epoch 41). Inference latencies, measured on the same Tesla T4 GPU as the detection experiments, are comparable across models: U-Net 8.5 ms, SegFormer-B0 8.8 ms, and DeepLabV3+ 10.9 ms per crop, all compatible with real-time operation. Given the essentially equal accuracy of the three architectures, the substantially smaller parameter count and memory footprint of SegFormer-B0 make it the preferred choice for edge deployment. Fig. 7 and Fig. 8 present visual comparisons of segmentation model metrics. Fig. 10 provides qualitative segmentation results from SegFormer-B0, contrasting worst-case and best-case predictions. Even in the worst cases, SegFormer-B0 captures the overall building footprint with only minor boundary inaccuracies, while best-case predictions exhibit near-perfect overlap with ground-truth masks, confirming the quantitative IoU of 93.2%.

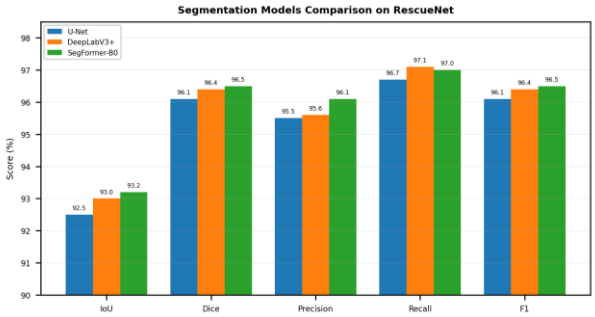

FIG. 7. Comparison of segmentation model metrics on RescueNet validation set

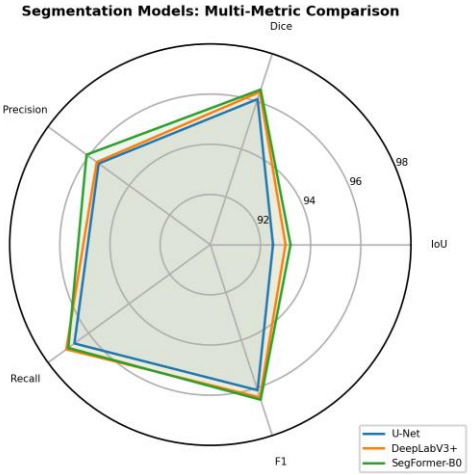

FIG. 8. Multi-metric radar comparison of segmentation models.

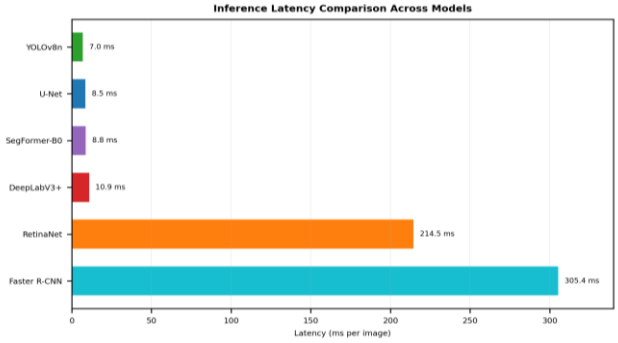

FIG. 9. Inference latency comparison across models (ms per image).

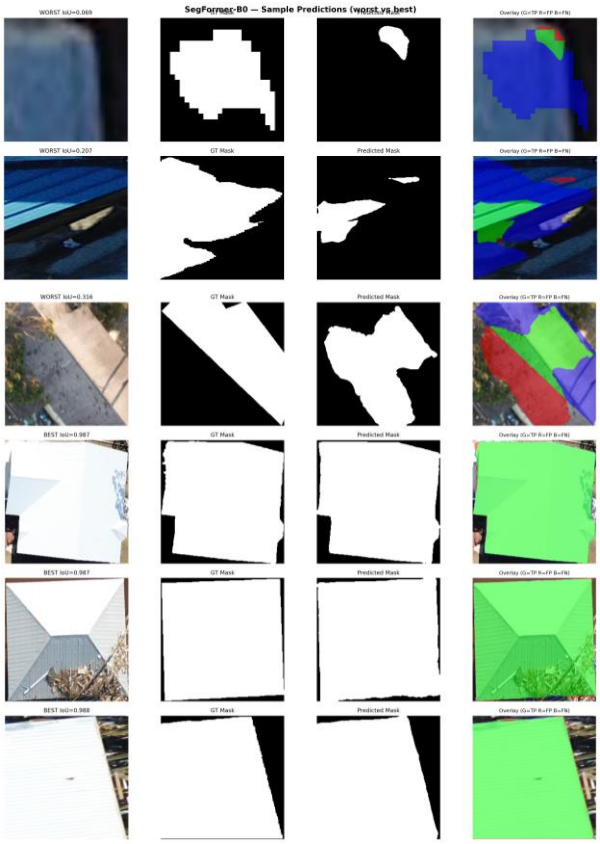

FIG. 10. SegFormer-B0 segmentation predictions: worst cases (top) and best cases (bottom). Columns: input crop, ground truth mask, predicted mask, overlay.

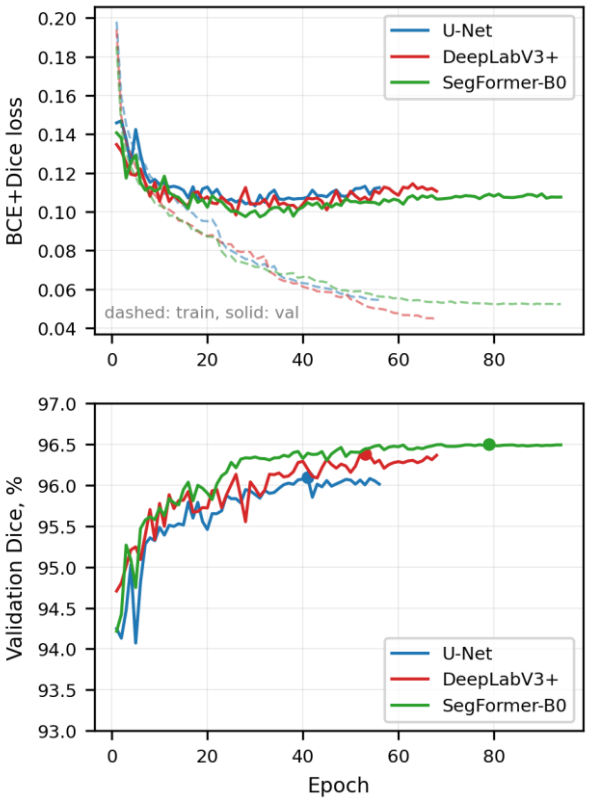

FIG. 11. Training dynamics of segmentation models: BCE+Dice loss (top; dashed – training, solid – validation) and validation Dice (bottom; markers denote best epochs). All models were trained until early stopping (patience of 15 epochs on validation Dice).

TABLE 3. Summary comparison for cascade component selection.

Stage	Model	Metric	Latency
Detection	YOLOv8n	mAP@0.5 = 90.9%	~7 ms
Segmentation	SegFormer-B0	Dice = 96.5%	8.8 ms

Edge Device Suitability. For deployment on edge devices such as NVIDIA Jetson Nano or Google Coral, inference latency is the primary constraint (Fig. 9 presents the latency comparison across all evaluated models). All three segmentation models offer comparable latency on the Tesla T4 (8.5 – 10.9 ms/image), so latency alone is no longer the deciding factor. SegFormer-B0 combines the highest accuracy (Dice = 96.5%) with a parameter count roughly an order of magnitude smaller (3.7M vs. 26.7 – 32.5M), which directly translates into a smaller memory footprint – a decisive property for memory-constrained edge accelerators. Combined with YOLOv8n for detection (~7 ms), the total pipeline latency remains around 16 ms per building, enabling real-time processing at over 60 instances/second. The YOLOv8n + SegFormer-B0 combination is therefore recommended for edge deployment in emergency response scenarios.

VII. CONCLUSION

In this work, a systematic comparison of six deep learning models for the task of building damage assessment from UAV imagery was performed: three detectors (Faster R-CNN, RetinaNet, YOLOv8) and three segmentation models (U-Net, DeepLabV3+, SegFormer). The key methodological distinction of this study is ensuring unified training conditions: within each task, all models start from an equivalent level of pretraining (COCO for detection; ImageNet-only encoders with randomly initialized decoders for segmentation) and use identical loss functions, optimizers, stopping criteria, and metrics, enabling correct comparison of architectural differences.

For adapting the RescueNet dataset to the detection task, automatic conversion of semantic masks to bounding boxes via the connected components algorithm was implemented. Separate conversion pipelines were developed for YOLO and torchvision annotation formats with identical filtering parameters, guaranteeing fair comparison. For segmentation, individual building crops were extracted based on bounding boxes, modeling the real cascaded architecture workflow.

The unified evaluation framework implemented in this work minimizes confounding factors and enables a substantially more direct comparison of model architectures. By standardizing hyperparameters, loss functions, pretrained weights, and evaluation metrics, we reduce the extent to which observed performance differences can be attributed to implementation or training artifacts rather than architectural properties, although training-related factors cannot be ruled out entirely.

Based on the experimental results, YOLOv8n demonstrated the best detection performance with mAP@0.5 of 90.9% and inference latency of approximately 7 ms, outperforming both Faster R-CNN (mAP@0.5 = 86.4%, latency 305.4 ms) and RetinaNet (mAP@0.5 = 83.8%, latency 214.5 ms) by a significant margin. For segmentation, all three models trained to convergence under the unified protocol reached closely comparable quality: SegFormer-B0 achieved the best results (IoU = 93.2%, Dice = 96.5%), marginally ahead of DeepLabV3+ (IoU = 93.0%, Dice = 96.4%) and U-Net (IoU = 92.5%, Dice = 96.1%), while using approximately 8x fewer parameters (3.7M) than either convolutional competitor. This near-parity of properly converged architectures underscores the importance of a unified training protocol: the dramatic differences that appear when training budgets are insufficient largely vanish when every model is allowed to converge. The recommended combination for the cascaded architecture is YOLOv8n for detection and SegFormer-B0 for segmentation, preferred for its parameter efficiency and highest accuracy at competitive latency (8.8 ms on Tesla T4), with a total inference time of approximately 16 ms per building instance. This combination is particularly well-suited for edge device deployment (e.g., NVIDIA Jetson, Raspberry Pi with accelerator) where both accuracy and real-time processing are required for emergency response scenarios.

Several limitations of the presented comparison should be acknowledged. First, all reported metrics correspond to a single experimental run with a fixed random seed (42), chosen to ensure exact reproducibility; confidence intervals or variance estimates across repeated runs with different initializations are not provided, and moderate run-to-run variability typical of deep learning models cannot be excluded. Second, detection models were trained with a batch size of 2 due to GPU memory constraints; although such settings are common in object detection, small batch sizes may influence optimization stability and convergence speed differently across architectures, which constitutes an additional training-related factor alongside purely architectural differences. Accordingly, the comparison presented here minimizes, but does not completely eliminate, implementation- and training-related confounding factors.

Future research directions include: (1) extending the comparison to the full cascaded architecture Detection-Segmentation-Classification with end-to-end evaluation on multi-class damage levels; (2) expanding the training dataset by incorporating additional post-disaster imagery from different geographic regions and disaster types (earthquakes, floods, wildfires); (3) reporting variance estimates across repeated runs with different random seeds to quantify the statistical reliability of the comparison; (4) experimenting with larger batch sizes and alternative learning-rate regimes, including re-benchmarking of the detection models under the same extended protocol; (5) evaluating additional architectures such as Mask R-CNN for instance segmentation, Swin Transformer for detection, and SAM (Segment Anything Model) for zero-shot segmentation; (6) performing comprehensive benchmarking on edge devices (NVIDIA Jetson Nano/Xavier, Raspberry Pi 5 with Hailo-8) with model quantization (INT8/FP16) and TensorRT optimization; (7) investigating the impact of image resolution on model performance by testing at multiple scales (256, 512, 1024 pixels); (8) developing an end-to-end real-time monitoring system with drone integration for automated post-disaster building assessment.

AUTHOR CONTRIBUTIONS

I.K. – conceptualization, methodology, software, investigation, writing-original draft preparation, writing-review and editing, visualization; D.U. – supervision, writing-review and editing; Yu.U. – supervision, methodology, writing-review and editing.

COMPETING INTERESTS

The authors declare no competing interests.

REFERENCES

[1] M. Rahnemoonfar, T. Chowdhury, and R. Murphy, "RescueNet: A High Resolution UAV Semantic Segmentation Dataset for Natural Disaster Damage Assessment," *Scientific Data*, vol. 10, Art. no. 913, 2023, doi: 10.1038/s41597-023-02799-4.

[2] S. Hafner, S. Gerard, J. Sullivan, and Y. Ban, "DisasterAdaptiveNet: A robust network for multi-hazard building damage detection from very-high-resolution satellite imagery," *International Journal of Applied Earth Observation and Geoinformation*, vol. 143, Art. no. 104756, 2025, doi: 10.1016/j.jag.2025.104756.

[3] S. Balovsyak and S. Stets, "Preprocessing of Object Images Before Their Detection Using YOLO Neural Network," *Security of Infocommunication Systems and Internet of Things*, vol. 3, no. 2, Art. no. 02002, Dec. 2025, doi: 10.31861/sisiot2025.2.02002.

[4] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in *Proc. Advances in Neural Information Processing Systems (NIPS)*, 2015, pp. 91–99.

[5] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal Loss for Dense Object Detection," in *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2980–2988.

[6] G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics YOLO," 2023. [Online]. Available: https://github.com/ultralytics/ultralytics

[7] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in *Proc. Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2015, pp. 234–241.

[8] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation," in *Proc. European Conference on Computer Vision (ECCV)*, 2018, pp. 801–818.

[9] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J. M. Alvarez, and P. Luo, "SegFormer: Simple and Efficient Design for Semantic Segmentation with Transformers," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2021, pp. 12077–12090.

[10] A. M. Braik and M. Koliou, "Automated building damage assessment and large-scale mapping by integrating satellite imagery, GIS, and deep learning," *Computer-Aided Civil and Infrastructure Engineering*, vol. 39, no. 15, pp. 2389–2404, 2024, doi: 10.1111/mice.13197.

[11] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature Pyramid Networks for Object Detection," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2117–2125.

[12] T. Adli, D. M. Bujaković, B. P. Bondžulić, M. Z. Laidouni, and M. S. Andrić, "Robustness of YOLO models for object detection in remote sensing images," *Journal of Electrical Engineering*, vol. 76, no. 5, pp. 429–442, 2025, doi: 10.2478/jee-2025-0045.

[13] G. Cheng, X. Yuan, X. Yao, K. Yan, Q. Zeng, X. Xie, and J. Han, "Towards Large-Scale Small Object Detection: Survey and Benchmarks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 11, pp. 13467–13488, 2023, doi: 10.1109/TPAMI.2023.3290594.

[14] H. Xia, J. Wu, J. Yao, H. Zhu, A. Gong, J. Yang, L. Hu, and F. Mo, "A Deep Learning Application for Building Damage Assessment Using Ultra-High-Resolution Remote Sensing Imagery in Turkey Earthquake," *International Journal of Disaster Risk Science*, vol. 14, no. 6, pp. 947–962, 2023, doi: 10.1007/s13753-023-00526-6.

[15] P. Iakubovskii, "Segmentation Models Pytorch," 2019. [Online]. Available: https://github.com/qubvel-org/segmentation_models.pytorch

Illia Kravets

PhD student at the Department of Software Engineering for Computer Systems, Yuriy Fedkovych Chernivtsi National University. MSc in Computer Science. Research interests include artificial intelligence, decision support systems, and Python-based ML solutions.

ORCID ID: 0009-0001-8467-391X

Dmytro Uhryn

Doctor of Technical Sciences, Professor at the Department of Computer Science, Yuriy Fedkovych Chernivtsi National University. Research interests include decision support information technologies, swarm intelligence systems, and industry-specific geographic information systems.

ORCID ID: 0000-0003-4858-4511

Yurii Ushenko

Doctor of Physical and Mathematical Sciences, Professor, Head of the Department of Computer Sciences, Yuriy Fedkovych Chernivtsi National University. Research interests include design of information systems, intelligent data analysis, pattern recognition and digital image processing, artificial neural networks, laser polarimetry and interferometry.

ORCID ID: 0000-0003-1767-1882

Порівняльний аналіз моделей детекції та сегментації для оцінки пошкоджень будівель на зображеннях БПЛА

Ілля Кравець*, Дмитро Угрин, Юрій Ушенко

Чернівецький національний університет імені Юрія Федьковича, Чернівці, Україна

*Автор-кореспондент (Електронна адреса: kravets.illia@chnu.edu.ua)

АНОТАЦІЯ Праця присвячена порівняльному аналізу моделей детекції об'єктів (Faster R-CNN, RetinaNet, YOLOv8) та семантичної сегментації (U-Net, DeepLabV3+, SegFormer) для задачі автоматизованої оцінки пошкоджень будівель на зображеннях безпілотних літальних апаратів. Порівняння охоплює як двоетапні (Faster R-CNN), так і одноетапні (RetinaNet, YOLOv8) підходи до детекції, а також згорткові (U-Net, DeepLabV3+) та трансформерні (SegFormer) архітектури сегментації. Як експериментальний набір даних використано RescueNet — високороздільний датасет семантичної сегментації постстихійних сцен, з якого методом компонентного аналізу масок було отримано обмежувальні рамки будівель для навчання моделей детекції. Адаптація датасету виконувалась окремими конвеєрами для кожного формату анотацій (YOLO txt, torchvision JSON), а для сегментації використано кропи окремих будівель, що моделює реальний сценарій роботи каскаду. Для забезпечення коректного порівняння всі моделі в межах кожної задачі навчались з однаковими гіперпараметрами, функціями втрат та pretrained вагами відповідного рівня (COCO для детекції, ImageNet для сегментації). Ефективність детекторів оцінювалась за метриками mAP@0.5 та mAP@0.5:0.95 через уніфіковане обчислення на базі pycocotools, а сегментаційних моделей — за глобальними IoU, Dice, Precision та Recall з акумуляцією TP/FP/FN. Додатково виміряно латентність інференсу для всіх моделей. На етапі сегментації задача формулюється як бінарне виділення контуру будівлі в межах виявленого кропу; класифікація ступеня пошкодження окремих будівель виконується на третьому етапі каскаду та виходить за межі цього дослідження. Результати дослідження дозволяють обґрунтовано обрати архітектуру для практичного розгортання каскадної системи «Детекція → Сегментація → Класифікація» в умовах реагування на надзвичайні ситуації, зокрема для розгортання на пристроях периферійних обчислень із обмеженими ресурсами. Робота є частиною ширшого проєкту з розробки автоматизованої системи моніторингу руйнувань на базі БПЛА.

КЛЮЧОВІ СЛОВА детекція об'єктів, семантична сегментація, оцінка пошкоджень, БПЛА, глибинне навчання.