Building footprint extraction from aerial images using an edge-aware YOLO-v8 network

Ziyu Song [1, 2], Weixi Wang [1,2], Zhenyu Hao[1,2], Xiaoming Li [1,2], Shengjun Tang [1,2], Linfu Xie [1,2*]
[1] Stale Key Laboratory of Subtropical Building and Urban Science, School of Architecture and Urban Planning, Shenzhen University, Shenzhen, PR. China
[2] Research Institute for Smart Cities, Shenzhen University, Shenzhen, PR. China

Keywords: Building footprint extraction, Remote sensing images, YOLO-v8, Instance segmentation, Prewitt model

Abstract

Building footprint extraction is a critical indicator for assessing urban infrastructure, and extracting building footprints from remote sensing imagery can have significant practical applications. However, achieving rapid and accurate extraction of building footprints remains highly challenging, especially in scenarios with complex scenes, dense building distributions, and small targets. The instance segmentation models of the YOLO series offer strong real-time performance, reducing considerable time and effort in practical applications. Therefore, we propose building footprint extraction based on an enhance YOLO-v8 network. This study focuses on three enhancements to the YOLO-v8 network to improve extraction accuracy. Building upon the YOLO-v8 framework, we have incorporated the Feature Pyramid Network (FPN) module into feature maps at all scales to efficiently propagate high-level semantic information. Additionally, we introduce the Triple Feature Encoder (TFE) module, which integrates spatial detail information from feature maps at three different scales to enhance the network's ability to extract multi-scale information. Finally, we explore the integration of the Prewitt model, a conventional edge detection operator, to assist in extracting edge features in target regions of feature maps. This integration aims to reduce the jagged edges frequently seen in the outcomes of the original YOLO-v8. Furthermore, the Prewitt operator's noise suppression capability helps mitigate the influence of non-target areas in the feature maps. The proposed framework achieves an instance segmentation accuracy of mAP_{50} is 84.6% and $mAP_{50:95}$ is 51.4% on public datasets, outperforming the original YOLO-v8 network.

1. Instructions

For applications such as urban information updating, urban planning, and urban 3D reconstruction, accurately and rapidly extracting building footprint information from remote sensing imagery is an important yet highly challenging task. With the development of deep learning theories, this task has garnered significant exploration in recent years. Currently, these studies can be broadly categorized into two types. The first type is pixel-wise semantic labeling tasks, such as PolyBuilding (Hu, Wang et al. 2023) and weighted U-Net (Gui and Qin 2021), which can generate raster building polygons. The second type is vertex-based object segmentation methods, exemplified by PolyWorld(Zorzi, Bazrafkan et al. 2022), which primarily extract building vertices to form contours and generate vector-building polygons. However, vertex-based object segmentation methods tend to produce redundant vertices and exhibit poor performance in extracting complex contours. Therefore, this paper utilizes an instance segmentation-based approach for building footprint extraction.

In the field of instance segmentation, researchers have proposed various methods, including two different architectures: two-stage and one-stage. Two-stage methods typically consist of two main stages: region proposal generation and instance segmentation. In the region proposal generation stage, the model utilizes a Region Proposal Network (RPN) (Ren, He et al. 2015) or other region generation methods to propose candidate regions that may contain targets. Subsequently, in the instance segmentation stage, further processing is performed on each candidate region to obtain accurate instance segmentation results. Representative models of this approach include Mask R-CNN (He, Gkioxari et al. 2017) and Cascade R-CNN (Cai and Vasconcelos 2018), which have achieved remarkable results in instance segmentation tasks. However, due to their multi-stage processing, these methods incur high computational costs and relatively slower speeds.

In contrast to two-stage methods, one-stage methods integrate region proposal generation and instance segmentation into a single stage to simplify the model structure and improve inference speed. These methods typically achieve instance segmentation by directly performing classification and bounding box regression at each pixel. Representative models of this approach include the YOLO series, which have higher inference speeds and lower computational costs, making them suitable for real-time application scenarios. Therefore, the experiments in this paper employ the real-time YOLOv8 (Ultralytics 2023) network.

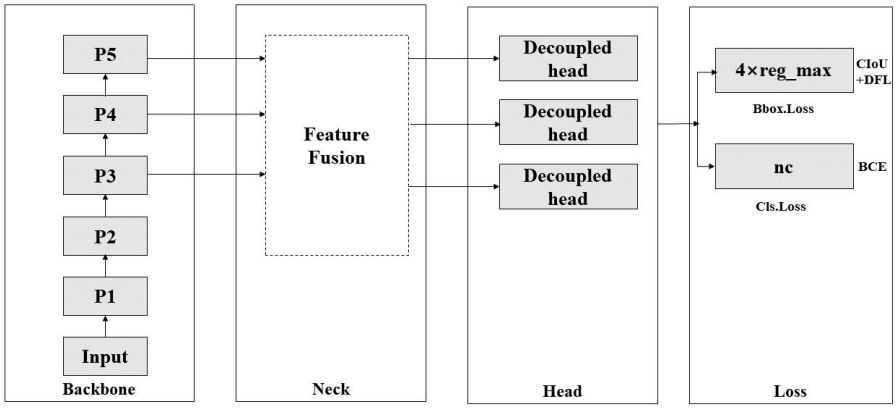

Figure 1: An abridged general view of the framework of YOLO-v8

The typical YOLO framework consists of four main components: Backbone, Neck, Head, and Loss, as illustrated in Figure 1. The Backbone is a convolutional neural network responsible for extracting image features at different granularities, resulting in five feature maps of different scales. The Backbone is composed of multiple ConvBNSiLU modules, C2f (CSP bottleneck including 2 convolutional layers with shortcut), and the final SPFF module. YOLO-v8 differs from YOLO-v5(Ultralytics 2023) by replacing the C3 module (CSP bottleneck including 3 convolutional layers) with the C2f module, and employing the Decoupled head instead of the Coupled head in the Head section. In addition to object detection and instance segmentation, YOLO-v8 also incorporates the idea of YOLACT(Bolya, Zhou et al. 2019). The Neck section primarily performs feature fusion, where the effective feature branches P3, P4, and P5 from the Backbone are input into the Feature Pyramid Network (FPN)(Lin, Dollár et al. 2017) for multi-scale fusion. The Head is responsible for decoding the fused features. During the decoding process, three heads of different sizes corresponding to the effective feature branches of the Backbone are used for object bounding box prediction. After upsampling the P3 features, pixel-by-pixel decoding is performed for target segmentation mask prediction, completing instance segmentation. In the segmentation head, three scales of features output three different anchor boxes, and the mask proto module outputs prototype masks, which are processed to obtain detection boxes and segmentation masks for instance segmentation tasks.

In this paper, we propose a one-stage instance segmentation model for building footprint extraction. The model primarily focuses on improving the Neck section of YOLO-v8 to achieve better multi-dimensional feature fusion. The main contributions of this work are as follows:

1) Addressing the multi-scale problem in building object detection and instance segmentation, we enhance the entire feature pyramid by implementing the FPN concept from the Neck section of YOLO-v8 across all five feature layers. The main modification involves adding the high-resolution feature maps P1 and P2 to the FPN. This facilitates the propagation of high-level semantic information to lower levels, thereby enhancing the entire pyramid.

2) We introduce a Triple Feature Encoder (TFE) (Kang, Ting et al. 2023) module to propagate spatial detail features from lower levels. By aggregating low-level feature maps (with high resolution but weak semantic information) and high-level feature maps (with low resolution and rich semantic information), we can further enhance the representation capability of multi-scale features.

3) We designed a feature map-based edge-guided module—the Prewitt model(Prewitt 1970). This module utilizes Prewitt operators to extract building boundary contours from lower-level feature maps that have already focused on building boundaries. The obtained edge information is then used as weight values added to the feature maps for the final instance segmentation.

2. Methods

2.1 Overall Architecture

Figure 2 shows the overall network architecture used in this paper. The framework combines multi-scale semantic and spatial information for building footprint extraction in remote sensing imagery. (1) The FPN Model, concatenates the five feature maps obtained from the backbone sequentially downward, facilitating the propagation of strong semantic features. (2) The TFE module, integrates high-level semantic information from feature maps with different sizes in the spatial dimension. The P3, P2, and P1 from lower levels are normalized to the same size through upsampling and downsampling, and then concatenated along the channel dimension as inputs to the detection head. (3) The Prewitt model, utilizes Prewitt operators to compute boundary information in each feature map along the channel dimension. This information is then added as weights to the original feature maps to highlight target boundary information, thereby improving network detection and segmentation accuracy.

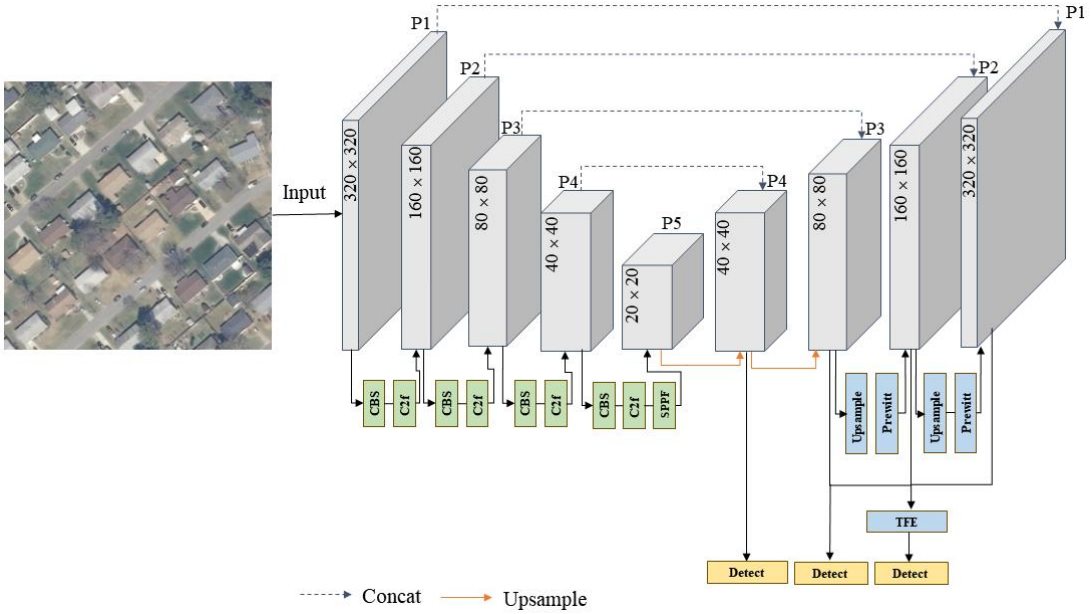

Figure 2: The overview of the proposed enhanced YOLO-v8 network

2.2 Feature Pyramid Network Model

FPN exploits the inherent multi-scale, pyramidal hierarchy of deep convolutional networks to construct feature pyramids with marginal extra cost. A top-down architecture with lateral connections is developed to generate high-level semantic feature maps at all scales. Although high-level feature maps have lower resolution, they contain rich semantic information after multiple convolutions. As illustrated in Figure 3, FPN propagates high-level semantic features from top to bottom, enhancing the semantic information of the entire feature pyramid. In the normal YOLO-v8 model, the feature maps of levels P3, P4, and P5 are concatenated and inputted into the detector.

The modification in this paper involves concatenating all five levels of feature maps with the corresponding feature maps in the backbone, with the addition of feature maps P1 and P2. The feature maps at lower levels undergo fewer convolution operations and focus more on image textures, geometric shapes, and other details, preserving most of the spatial information from the original image. Additionally, these two feature maps have higher spatial resolutions, which are more advantageous for recognizing and detecting small objects.

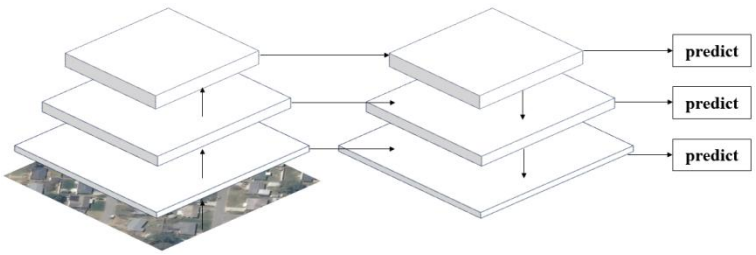

Figure 3: The overview of the FPN Model

2.3 Triple Feature Encoder Model

The traditional FPN feature fusion mechanism only performs upsampling on high-level small-sized feature maps and then adds them to the previous layer feature maps. Although this method transfers semantic information, it ignores the rich spatial details contained in the low-level features. Therefore, we adopt a Triple Feature Encoder Model, which incorporates large-sized feature maps to enhance detailed feature information. The main operations of the TFE module include the following:

1) Downsampling the large-sized feature map using the AvgPooling module to reduce its size by half. Using AvgPooling helps maintain high-resolution features to prevent the loss of information about small targets. Then, the 1 × 1ConvBNSiLU module is applied for convolutional operations to enhance the model's non-linearity and complexity.

2) Performing a 1 × 1 convolutional operation on the medium-sized feature map to maintain model complexity.

3) Upsampling the small-sized feature map using nearest neighbor interpolation(Rukundo and Cao 2012) to double its size. This helps maintain the richness of local features in low-resolution images. Then, a 1 × 1 convolutional operation is performed.

4) Finally, the three obtained feature maps are concatenated along the spatial dimension and input to the detector for object detection and instance segmentation.

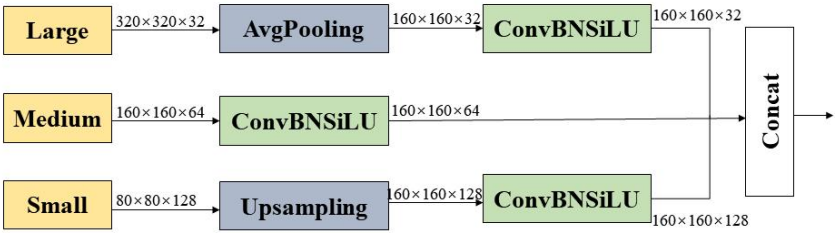

Figure 3: The overview of the TFE Model

2.4 Prewitt Model

By observing the feature maps, it can be noticed that after multiple convolutional operations and feature fusion, the P2 feature map has already extracted most of the features, with some channels already focusing on the target objects. To extract the features that have been attended to in the feature map and define non-target areas as negative samples, we attempted to use traditional edge detection operators to extract the target objects in the feature map. By quantifying the edge features and adding them as weight values into the original feature map, a method similar to the attention mechanism was employed to make the network pay more attention to the target regions.

Common image edge detection algorithms include the Prewitt operator, Laplacian operator, Sobel operator (Sobel and Feldman 1968), and Canny operator (Canny 1986). Through experiments on edge detection using various operators on the feature maps, it was found that the Prewitt operator achieves the best performance in extracting targets from the feature maps and does not introduce significant noise. The experimental results are shown in Figure 5. Therefore, the Prewitt operator was chosen as the fundamental edge detection algorithm in this framework.

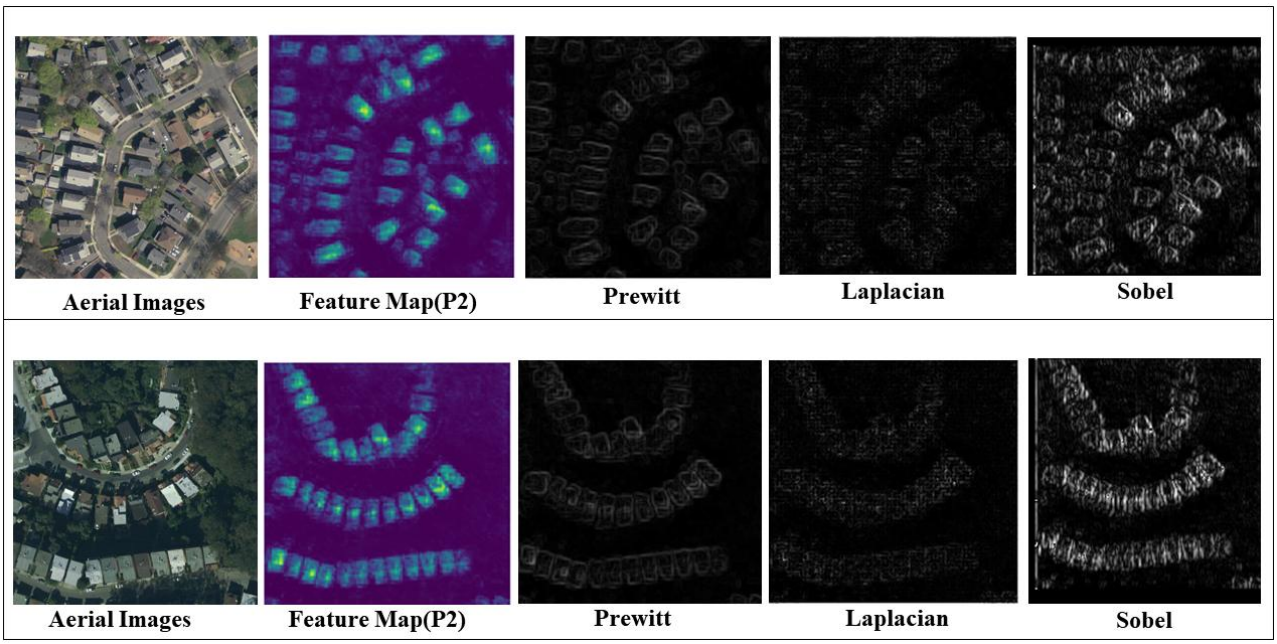

Figure 5: Comparison of several traditional edge detection methods

The Prewitt operator is an edge detection method based on first-order differentiation. It detects edges by identifying extreme values of the difference in grayscale values between neighboring pixels in the vertical and horizontal directions. This operator can eliminate some false edges and has a smoothing effect on noise. The principle involves using two directional templates in the image space to perform neighborhood convolution with the image. One template detects horizontal edges, and the other detects vertical edges.

The principle of edge detection using the Prewitt operator is as follows:

1) Define the horizontal convolution kernel (PrewittX):

$$\begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix} \quad (1)$$

2) Define the vertical convolution kernel (PrewittY):

$$\begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix} \quad (2)$$

For pixels $I(x, y)$ in the image, the horizontal gradient (G_x) and vertical gradient (G_y) can be computed by applying the convolution kernels to local regions of the image.

3) Compute the horizontal gradient(G_x):

$$G_x = \mathrm{PrewittX} * I(x,y) \quad (3)$$

4) Compute the vertical gradient(G_y):

$$G_y = \mathrm{PrewittY} * I(x,y) \quad (4)$$

5) Compute the gradient magnitude for each pixel using the following formula:

$$G = \sqrt{{G_x}^2 + {G_y}^2} \quad (5)$$

Where G represents the gradient magnitude for each pixel, based on the calculated gradient magnitude, pixels with values greater than the threshold are considered as edge pixels, thus achieving edge detection. This edge detection method is simple to compute and has a fast calculation speed.

The flowchart of the Prewitt Model is illustrated in Figure 6, which can be primarily divided into five main steps:

1) Using the Prewitt operator, extract the edge features E_c from the spatial dimensions of the feature map, where c represents the channel dimension. Simultaneously, initialize a learnable parameter α as a coefficient, which learns the weight assigned to each E_c during network training.

$$E = \alpha * E_c \quad (6)$$

2) Concatenate c feature maps along the channel dimension, then sum them up along the channel dimension to obtain the weight values of the edge features across the entire image.

3) Apply the Sigmoid function to nonlinearize the weight values and constrain them within the range [0,1].

4) Perform a 1×1 convolution on the original feature maps. The main purpose is to increase the model's complexity and learnability, avoiding the occurrence of gradient disappearance or overfitting.

5) Finally, multiply the obtained weight values by the convolved feature values.

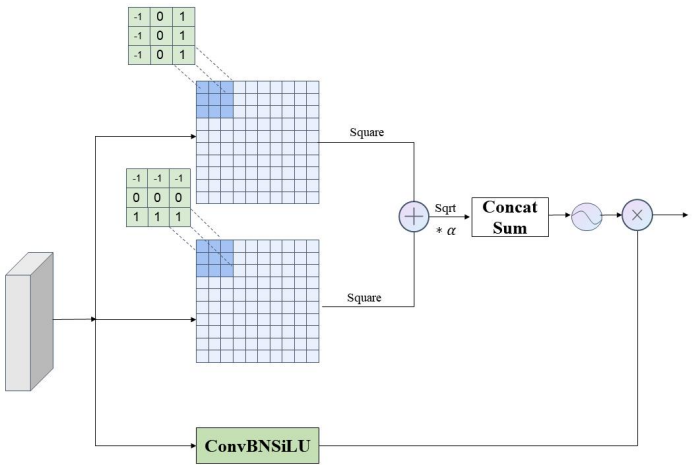

Figure 6: The overview of the Prewitt Model

3. Experiments and Preliminary Results

3.1 Datasets

The experimental data used in this study was obtained from a publicly available dataset compiled by the 2016 Data Plus Energy Analytics Group (https://doi.org/10.6084/m9.figshare.3504413.v1). The dataset consists of high-resolution aerial orthoimages with spatial resolutions ranging from 0.15 m to 0.3 m, acquired from the United States Geological Survey (USGS), and corresponding building footprints downloaded from OpenStreetMap (OSM). For the experiments, a total of 2500 orthoimages with a size of 640×640 pixels, covering five different cities, were selected as the experimental dataset. The dataset was randomly split into training set, validation set, and test set in the ratio of 8:1:1, with specific quantities of 1966, 267, and 267 images.

3.2 Training Implementation

The experiments are implemented on the NVIDIA A100-SXM4-40GB GPU and Pytorch 2.1.1, Python 3.8, and CUDA 12.1 dependencies. We did not utilize pre-trained weights from the COCO dataset. The input image size is 640×640. The batch size of the training data quantity is 16. The training process lasts 250 epochs. We use Stochastic Gradient Descent (SGD) as an optimization function to train the model. The hyperparameters of SDG are set to 0.937 of the momentum, 0.01 of the initial learning rate, and 0.0005 of the weight decay.

3.3 Quantitative Results

Table 1 shows the improvement in accuracy achieved by our method on two base network models, YOLOv8n-seg and YOLOv8n-p2-seg. A comparative analysis reveals that our approach enhances the Mask mAP_{50} and Mask $mAP_{50:95}$ metrics by 2.7% and 2.9%, over the original YOLO-v8 model. Additionally, there is an increase in accuracy for the YOLOv8n-p2-seg network, which is specifically designed for small object detection within the YOLO series.

3.4 Qualitative Results

Figure 7 illustrates a visual comparison of three methods for detecting and segmenting buildings in satellite imagery. By comparing the yellow-boxed areas in the image, it can be

observed that our method outperforms the other two methods in detecting areas with complex scenes and dense building distributions. Comparing the green areas reveals that our method exhibits better regularity and smoothness along edges compared to YOLO-v8. Comparing the blue areas demonstrates that our method achieves better overall target extraction performance than YOLOv8-p2-seg. Our approach combines the advantages of the other two methods while also compensating for their respective shortcomings.

Model	Box		Mask	
	mAP_{50}	$mAP_{50:95}$	mAP_{50}	$mAP_{50:95}$
YOLOv8n-seg	0.839	0.562	0.819	0.485
YOLOv8n-p2-seg	0.84	0.563	0.817	0.478
ours	**0.855**	**0.574**	**0.846**	**0.514**

Table 1: Performance comparison of different models for building instance segmentation on the dataset. The best results are in bold.

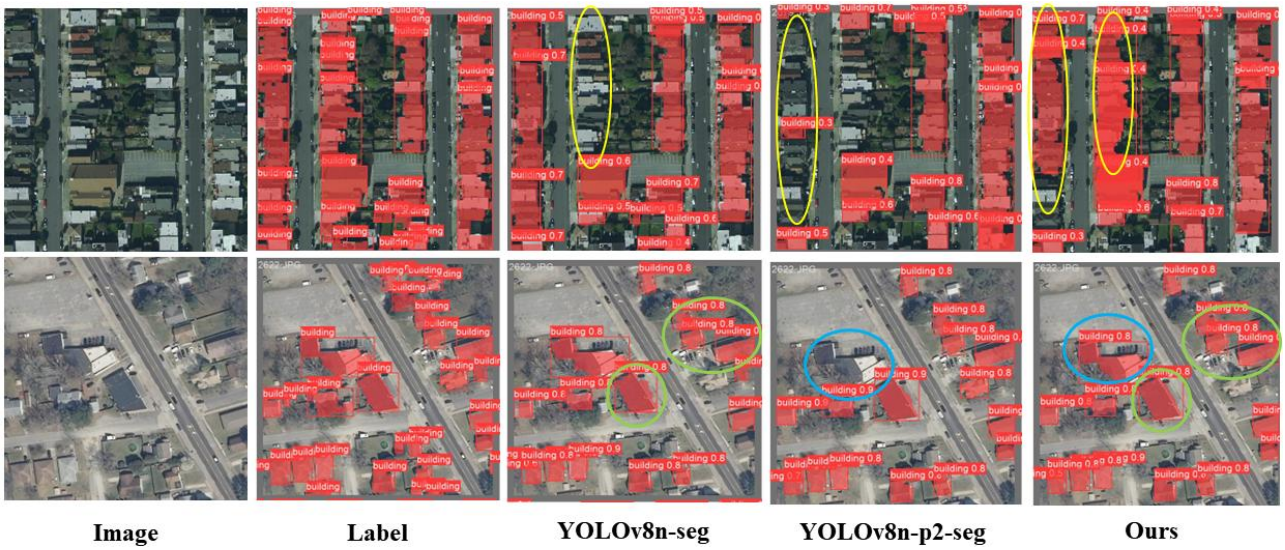

Figure 7: Qualitative comparison of different instance segmentation models

Figure 8 visualizes the difference in the large-scale feature map P2 before and after passing through the Prewitt Model. It is evident from the image that after overall weighting, regions with higher attention in the feature map exhibit more pronounced edge information. Regions with lower attention, such as roads, are assigned smaller weight values, making them easier to recognize as negative samples and subsequently filtered out during the subsequent object detection process. Additionally, due to the strong noise suppression capabilities of the Prewitt operator, a significant amount of noise in the resulting Prewitt Map is filtered out.

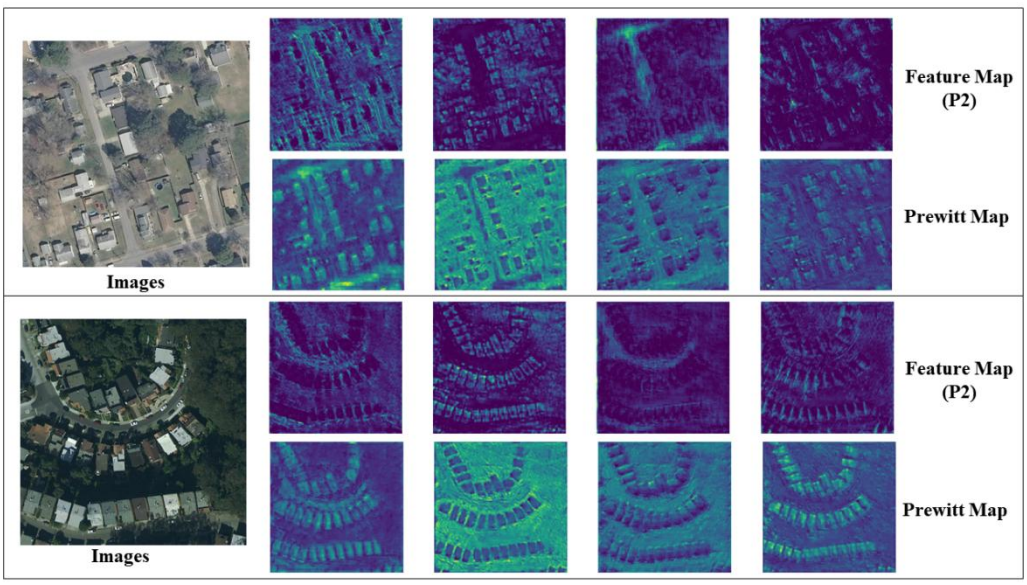

Figure 8: Comparison of feature maps before and after applying the Prewitt operator.

3.5 Ablation Experiments

Table 2 compares the contributions of three different modules to instance segmentation performance. The most significant improvement is observed when incorporating the TFE Model, which integrates rich spatial detail information from lower-level feature maps into higher-level semantic information, thereby enhancing the detection performance of small objects. The

primary contribution of the Prewitt Model lies in improving the smoothness of building edges. FPN effectively propagates high-level semantic information to the lowest-level feature maps. The combined use of these three different modules elevates the performance of YOLO-v8 in both object detection and instance segmentation to a satisfactory level of accuracy.

Method			Box		Mask	
FPN	TFE	Prewitt	mAP_{50}	mAP_{50}	mAP_{50}	mAP_{50}
			0.839	0.562	0.819	0.485
√			0.844	0.567	0.826	0.491
√	√		0.848	0.571	0.838	0.51
√		√	0.849	0.571	0.835	0.503
√	√	√	**0.855**	**0.574**	**0.846**	**0.514**

Table 2: Adding accuracy comparison of different modules

4. Conclusion

We have improved YOLO-v8 to develop a precise and fast instance segmentation model for building remote sensing imagery. This model effectively propagates rich semantic information from high-level layers while integrating both semantic and spatial detail information for shape-aware building detection and segmentation. We made improvements to three aspects of YOLO-v8: the fusion of multi-scale feature information through FPN and TFE models enhances segmentation performance for multi-scale and small object instances. The Prewitt Model guides the network by computing target edges to enhance focus on target instances while suppressing noise effects on results, achieving accurate extraction of building edges. Although our model performs well in densely distributed and complex scenes, the detection performance for buildings obscured by trees remains suboptimal. Further research is needed to address this specific scenario.

5. References

Bolya, D., et al. (2019). Yolact: Real-time instance segmentation. Proceedings of the IEEE/CVF international conference on computer vision.

Cai, Z. and N. Vasconcelos (2018). Cascade r-cnn: Delving into high-quality object detection. Proceedings of the IEEE conference on computer vision and pattern recognition.

Canny, J. (1986). "A computational approach to edge detection." IEEE Transactions on pattern analysis and machine intelligence(6): 679-698.

Gui, S. and R. Qin (2021). "Automated LoD-2 model reconstruction from very-high-resolution satellite-derived digital surface model and orthophoto." ISPRS Journal of Photogrammetry and Remote Sensing **181**: 1-19.

He, K., et al. (2017). Mask r-cnn. Proceedings of the IEEE international conference on computer vision.

Hu, Y., et al. (2023). "PolyBuilding: Polygon transformer for building extraction." ISPRS Journal of Photogrammetry and Remote Sensing **199**: 15-27.

Kang, M., et al. (2023). "ASF-YOLO: A Novel YOLO Model with Attentional Scale Sequence Fusion for Cell Instance Segmentation." arXiv preprint arXiv:2312.06458.

Li, W., et al. (2023). "Joint semantic–geometric learning for polygonal building segmentation from high-resolution remote sensing images." ISPRS Journal of Photogrammetry and Remote Sensing **201**: 26-37.

Lin, T.-Y., et al. (2017). Feature pyramid networks for object detection. Proceedings of the IEEE conference on computer vision and pattern recognition.

Prewitt, J. M. (1970). "Object enhancement and extraction." Picture processing and Psychopictorics **10**(1): 15-19.

Ren, S., et al. (2015). "Faster r-cnn: Towards real-time object detection with region proposal networks." Advances in neural information processing systems **28**.

Rukundo, O. and H. Cao (2012). "Nearest neighbor value interpolation." arXiv preprint arXiv:1211.1768.

Sobel, I. and G. Feldman (1968). "A 3x3 isotropic gradient operator for image processing." a talk at the Stanford Artificial Project in **1968**: 271-272.

Ultralytics (2023). "NEW - YOLOv8 in PyTorch." from https://github.com/ultralytics/ultralytics.

Ultralytics (2023). "Ultralytics YOLOv5 architecture." from https://github.com/ultralytics/yolov5.

Zorzi, S., et al. (2022). Polyworld: Polygonal building extraction with graph neural networks in satellite images. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.