Inform : Jurnal Ilmiah Bidang Teknologi Informasi dan Komunikasi
Vol.11 No.1, January 2026, P-ISSN : 2502-3470, E-ISSN : 2581-0367

Integration of Google Maps with the YOLO Method for Urban Greening Monitoring

Yennimar[1], Christian Vieri[2], Dandi Prasetyo[3], Amanda Febriyani[4]

[1,2,3,4]*Informatics Department, Universitas Prima Indonesia, Medan, Indonesia*

[1]yennimar@unprimdn.ac.id (*)

[2,3,4][christianvieri22032001, dandiprasetyoo18, amandafebriyani2001]@gmail.com

Received: 2025-01-18; Accepted: 2025-11-26; Published: 2026-01-05

Abstract— The development of artificial intelligence (AI) and computer vision technology has opened up new opportunities in environmental monitoring, especially green open spaces (RTH) in urban areas. This study aims to develop an automatic tree-detection system for satellite imagery using the You Only Look Once version 8 (YOLOv8) algorithm integrated with the Google Maps API. Therefore, the novelty of this study lies in integrating the YOLOv8 model with the Google Maps JavaScript API to produce an interactive spatial visualization of detection results, enabling real-time analysis of vegetation distribution to support sustainable urban policies. The methods used include collecting a 640 × 640-pixel-resolution satellite image dataset from 21 sub-districts in Medan City, object labelling using the Roboflow platform, data augmentation to increase diversity, and model training on Google Colaboratory. The resulting YOLOv8 model achieved excellent performance, with a precision of 0.864, recall of 0.788, and an mAP@0.5 of 0.938, indicating strong object detection capabilities. The detection results were then converted to geographic coordinates and visualized using the Google Maps JavaScript API to provide interactive spatial information on tree distribution in urban areas. This study not only demonstrates the technical feasibility of integrating the YOLOv8 algorithm with the Google Maps API but also provides a methodological framework for spatial AI coupling that can be replicated across other environmental monitoring domains. In addition, the study identifies several limitations, including dependence on image resolution, limited temporal coverage, and relatively lower mAP@0.5–0.95 performance in dense vegetation, which will guide future model improvement.

Keywords— Google Maps API; Green Open Space; Urban Greening Monitor; Tree Mapping, YOLOv8.

I. INTRODUCTION

The development of Artificial Intelligence (AI) and computer vision technologies has had a significant impact across various fields [1]-[3], including urban environmental monitoring. One of the most prominent approaches in object detection is the You Only Look Once (YOLO) algorithm, renowned for its ability to perform real-time detection with high accuracy [4][5]. In recent years, YOLO has undergone continuous updates [6], including the release of its latest version, YOLOv8, which offers improvements in inference speed and accuracy, making it well-suited for spatial analysis using satellite imagery [7]-[10].

On the other hand, rapid urbanization has led to a significant reduction in green open spaces (GOS) in major cities, including those in Indonesia [11][12]. Medan is one of the largest cities in Indonesia with a high level of urbanization, resulting in a decline in green open space due to accelerated development. The presence of GOS plays a vital role in maintaining air quality, reducing pollution, and enhancing public comfort and health [13]. Therefore, monitoring vegetation in urban areas has become an urgent necessity, both to support spatial planning policies and to align with the Sustainable Development Goals (SDGs), which emphasize the importance of sustainable cities [14]-[16].

Conventional methods for monitoring green open spaces (GOS), such as field surveys, often require substantial costs and extended time [17][18]. To address these limitations, the use of satellite imagery and digital mapping platforms such as Google Maps provides an effective solution by offering fast, accurate, and up-to-date access to spatial data [19][20]. The Google Maps API enables the acquisition of high-resolution imagery for detailed vegetation analysis. Integrating these images with deep learning–based detection methods creates opportunities for developing more efficient and automated monitoring systems [21]-[23].

In the context of vegetation detection modelling, the use of YOLOv8 offers significant advantages as this algorithm is designed to address visual challenges such as variations in object size, lighting conditions, and complex backgrounds in satellite imagery [9][24][25]. Applying data augmentation during model training also enhances detection robustness across diverse conditions. By integrating detection results into an interactive map using the Google Maps JavaScript API, spatial data on vegetation distribution can be visualized and used for policy analysis in urban environmental management.

Previous studies have highlighted various strengths and limitations of different YOLO variants for urban greening monitoring. YOLOv4-Lite [10][26] demonstrated efficiency with a lightweight architecture and high accuracy, yet it was less optimal for detecting densely vegetated areas and real-time, city-scale applications. YOLOv2 [26][27] offered fast performance (~45 FPS) and high accuracy (precision 0.88; recall 1.00), but its application was limited to campus areas and required computationally expensive 3D reconstruction. YOLOv5 [28][29] proved effective for detecting large-canopy trees and long-term monitoring; however, it tended to be less accurate for small trees, and its performance metrics were not

DOI : https://doi.org/10.25139/inform.v11i1.10800

always reported in detail. Meanwhile, YOLO-SegNet, based on YOLOv8 [30] achieved very high accuracy (IoU 92%) and handled complex backgrounds effectively [31], but its two-stage architecture made real-time implementation challenging.

This study adopts YOLOv8 because it offers high inference speed and improved accuracy compared to its predecessors, enabling the detection of small objects, such as trees, in high-resolution satellite imagery. In addition, YOLOv8 features an anchor-free architecture and advanced augmentation techniques, which enhance robustness against variations in object size, lighting, and complex urban backgrounds. The integration of YOLOv8 with the Google Maps API further enables practical interactive spatial visualization for vegetation distribution analysis and data-driven decision-making.

Despite numerous studies employing various YOLO versions for vegetation monitoring, most remain application-oriented with minimal methodological innovation. Previous works have rarely addressed the challenge of transforming pixel-level detection results into coordinate-based spatial intelligence. Therefore, this research fills that gap by proposing a coupling framework between YOLOv8 detection and Google Maps API to bridge image-level recognition and spatial decision-making, offering both algorithmic enhancement and applied value.

This study provides several significant contributions to the field of object detection and spatial analysis. The YOLOv8-based tree detection system developed in this research achieved an mAP@0.5 of 0.938, indicating a substantial improvement in accuracy over previous models. The detection results were then integrated into Google Maps by converting bounding boxes into geographic coordinates, enabling interactive map-based monitoring. This approach also supports the planning of green open spaces (GOS) by providing accurate data for evidence-based policy-making. Furthermore, this study presents a framework that can be further developed into a real-time vegetation monitoring dashboard to enhance the effectiveness of urban environmental management.

The technical characteristics of YOLOv8, such as its anchor-free architecture and adaptive feature aggregation, have been widely discussed in prior works. The researchers [4][32] confirm its robustness for satellite imagery–based object detection.

Based on the background outlined above, this study aims to develop an automated tree-detection system for Google Maps satellite imagery using the YOLOv8 algorithm. Specifically, this research aims to train an object detection model to accurately identify the presence of trees and integrate the detection results into a digital map using geographic coordinates to support spatial analysis and the monitoring of green open spaces.

II. RESEARCH METHODOLOGY

This study adopts a quantitative, experimental deep learning approach to detect tree objects in Google Maps satellite imagery. This approach was chosen because YOLOv8 supports direct integration with spatial data, enabling its application for automated monitoring of green open spaces.

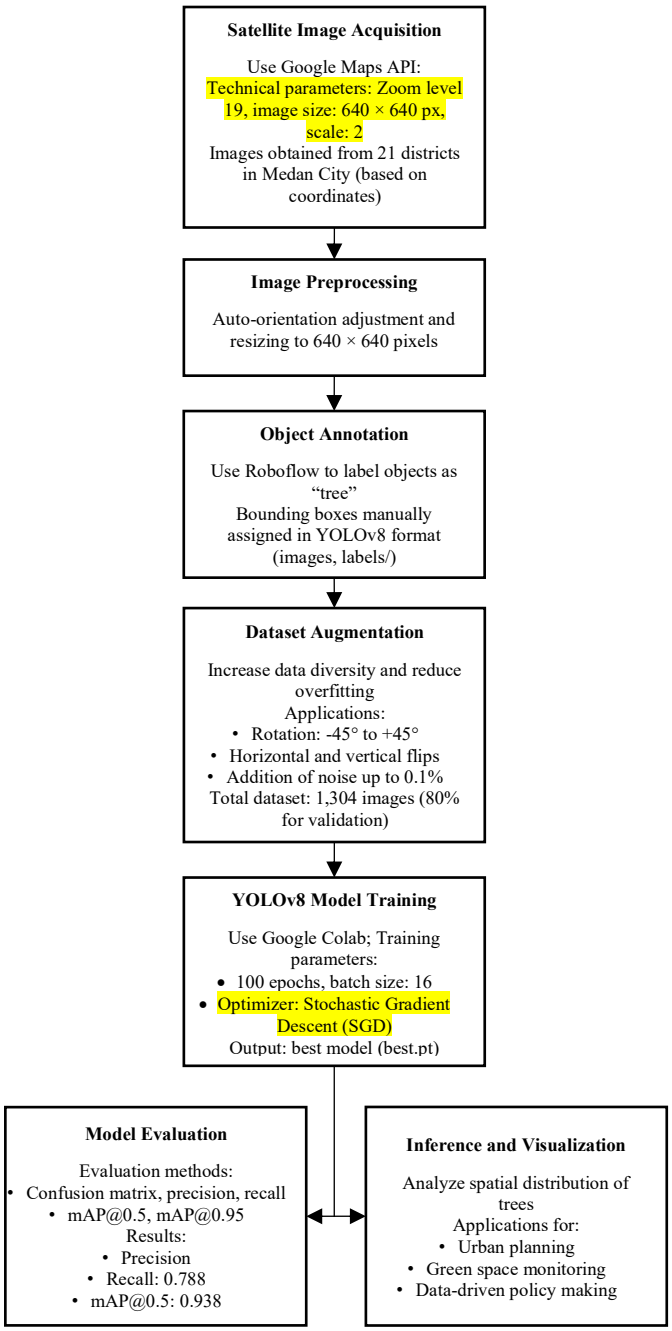

Fig.1. Flow of the YOLOv8 Method

The data acquisition process, the flow of how the YOLOv8 method works for Urban Greening Monitoring, is illustrated in Fig.1. This workflow outlines each critical step, from image pre-processing to model training, evaluation, detection, and finally, spatial visualization. The flow provides a systematic overview that supports understanding of how YOLOv8 is operationalized in this study to accurately identify tree objects

DOI : https://doi.org/10.25139/inform.v11i1.10800

and translate detection results into actionable geospatial data.

The study was conducted in Medan City, the capital of North Sumatra Province, Indonesia, one of the largest metropolitan areas outside Java Island. Medan consists of 21 administrative districts with varying topography, land use, and vegetation density. These diverse characteristics make the city an ideal case study for analyzing urban greening. The research focused on detecting trees in this urban environment using satellite imagery from the Google Maps API. Images were acquired using specific technical parameters, including a zoom level of 19, resolution of 640 × 640 pixels, and a scale factor of 2, ensuring high spatial resolution suitable for deep learning-based object detection. Image acquisition was strategically conducted at selected coordinate points to ensure comprehensive spatial coverage across the entire study area.

A. Dataset Collection

The data collection process in this study used the Google Maps API to obtain satellite imagery of the Medan City area. Data acquisition focused on 21 districts representing geographic and vegetation diversity, including densely populated areas, suburban zones, and green open spaces (GOS) such as city parks, green corridors, and urban forests. The retrieved images had a resolution of 640 × 640 pixels to ensure object details were preserved for labelling and model training. In total, 1,304 images were successfully collected. An example of the satellite imagery used in this study is presented in Fig. 2.

Fig.2. Example of Satellite Imagery

The collected dataset was then split into two main subsets: training and validation. This split was carried out proportionally, with 80% allocated for training and 20% for validation, as shown in Table I.

TABLE I
DATASET SPLIT

Label	Total Images	Training (80%)	Validation (20%)
Tree	1.304	1.043	261

Table I presents the dataset split used in this study, comprising 1,304 satellite images, each annotated with a single object class: "tree". This exclusive focus on tree objects aligns with the study's goal of monitoring urban greening by accurately detecting vegetative elements in urban satellite imagery. The dataset was divided into two subsets: 80% (1,043 images) for training and 20% (261 images) for validation. This ratio follows common best practices in deep learning, allocating most of the data to model training to maximize learning performance, while reserving a smaller portion for evaluating the model's generalization to unseen data. The labelling process ensured that each image reliably represents the presence of tree objects, enabling the YOLOv8 model to learn distinct spatial and textural features associated with vegetation. The total number of labelled instances is considered sufficient for a one-class object detection task, particularly in satellite imagery, where vegetation patterns may vary across geographic locations and environmental conditions. Moreover, the size and distribution of this dataset help reduce the risk of overfitting and support robust model training and validation across diverse spatial scenarios found in Medan City.

B. Image Annotation

The image annotation stage was carried out after the dataset collection process was completed. Satellite images from Google Maps were annotated in the Roboflow platform to detect trees. Each tree in the image was assigned a bounding box and labelled as "tree." This annotation process was performed manually to produce highly accurate labelled data, aligning with the research objective of mapping vegetation presence in urban areas. An example of the satellite imagery used in this stage is shown in Fig. 3.

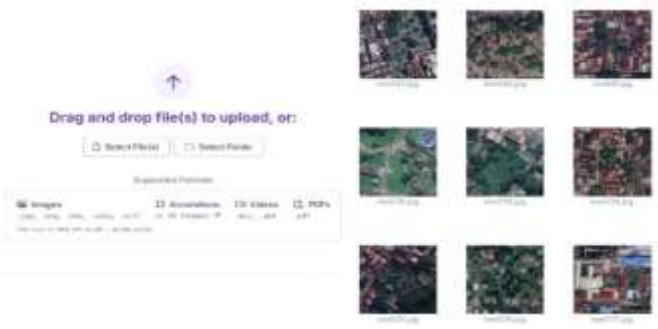

Fig.3. Example of an Annotated Satellite Image

1) Dataset Upload: The dataset was uploaded to the Roboflow platform before the annotation process. This step enabled resolution adjustment, file format configuration, and

DOI : https://doi.org/10.25139/inform.v11i1.10800

annotation setup, ensuring that dataset management was more structured and supported more efficient object detection.

2) Dataset Class Creation: Object classes were created to facilitate the classification of a dataset annotated with bounding boxes. At this stage, Roboflow automatically grouped the data into predefined categories, ensuring a more systematic workflow for model training.

3) Labelling Process: Each object in the image was annotated with a bounding box to indicate the tree's position. Once the bounding box was set, the Roboflow system displayed the category label according to the predefined class. This labelling process is illustrated in Fig. 4, which shows how bounding boxes were manually applied to maintain data quality.

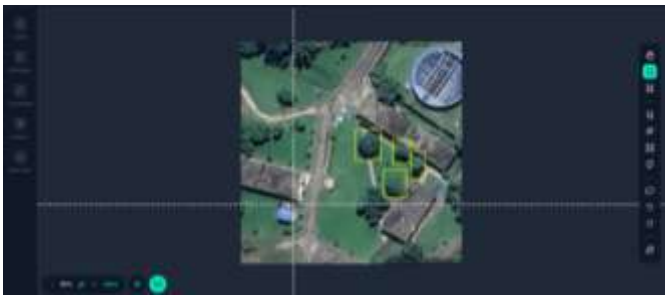

Fig.4. Labelling Process Using Roboflow

4) Dataset Splitting: After labelling was completed, the dataset of 434 original images was split into two subsets: 80% for training (347 images) and 20% for validation (87 images). This process was performed automatically by Roboflow.

To enhance data diversity and prevent overfitting, data augmentation was applied to the training set. The augmentation techniques used included horizontal and vertical flips, rotations between -45° and +45°, and the addition of 0.1% noise. The purpose of augmentation was to enable the model to recognize trees under various visual conditions and orientations. Before augmentation was applied, all images underwent a pre-processing stage, which included auto-orientation adjustment and resizing to 640 × 640 pixels, in accordance with the YOLOv8 input format. This process is illustrated in Table II.

TABLE II
PRE-PROCESSING DAN AUGMENTASI DATA

Pre-processing	Auto-Orient: Applied Resize: Stretch to 640x640
Augmentations	Outputs per training example: 3 Flip: Horizontal, vertical Rotation: Between -45° and +45° Noise: Up to 0,1% of pixels

Before training, the acquired images underwent a pre-processing stage that included auto-orientation adjustment and resizing to 640 × 640 pixels to ensure uniformity of the model's input. Subsequently, an annotation process was carried out using the Roboflow platform, during which each tree was assigned a bounding box and labelled as "tree." Annotation was performed manually to maintain high label accuracy. The dataset was then exported to the YOLOv8 format, including image and label directories.yaml configuration file.

The resulting dataset consisted of 1,304 images, split into 80% for the training set and 20% for the validation set. To increase data variability and reduce the risk of overfitting, an augmentation process was applied, including rotations ranging from -45° to +45°, horizontal and vertical flips, and the addition of noise up to 0.1%. These augmentation techniques aimed to expand the visual variability of trees, thereby enhancing the model's robustness to different conditions.

The model was trained on Google Colaboratory using a dataset formatted for the YOLOv8 architecture. The training configuration included 50 epochs, a batch size of 16, and an optimizer based on Stochastic Gradient Descent (SGD), chosen for its stability and smooth convergence during training. Upon completion of training, the system generated the best weight file (best.pt), representing the model with optimal performance. Model evaluation was conducted using a confusion matrix and evaluation metrics, including precision, recall, and mean Average Precision (mAP) at IoU thresholds of 0.5 and 0.5–0.95.

The evaluation results demonstrated excellent performance, with a precision score of 0.864, a recall of 0.788, and mAP@0.5 reaching 0.938. After training, the model was used to perform inference on test images to detect trees. Each detection result was then converted into geographic coordinates (latitude and longitude) and visualized on an interactive map using the Google Maps JavaScript API. This visualization displays the distribution of trees as green markers, which can be utilized for spatial analysis, support urban spatial planning, and enable continuous monitoring of green open spaces.

III. RESULT AND DISCUSSION

A. Training

After pre-processing and augmentation, the dataset was exported in a format compatible with the YOLOv8 architecture for model training. The dataset was organized into a folder structure with train and valid directories, each containing subfolders for images and labels. The images folder stored the image files, while the labels folder contained text files with bounding box information and class labels. In addition, a data.yaml file served as the main configuration file, specifying the dataset paths, the number of classes, and the class names. This structure was designed to ensure compatibility with the model training process in Google Colaboratory (Google Colab).

1) Exporting the Dataset from Roboflow: The annotated dataset was exported from Roboflow in YOLOv8 format, allowing it to be used directly in Google Colab. This process ensured that all files were organized correctly in accordance with the model architecture's requirements.

2) Model Training in Google Colab: The model training was conducted on the Google Colaboratory platform using the exported dataset. The dataset files were uploaded manually and utilized to train the YOLOv8 model. The training configuration leveraged the previously structured dataset, which facilitated

DOI : https://doi.org/10.25139/inform.v11i1.10800

model implementation. A sample code snippet used for the training process is presented in Code 1.

Code 1.

```
from ultralytics import YOLO

model = YOLO("yolov8n.pt")

model.train(
data=
"/content/drive/MyDrive/tree.detection.v1i.yolov8/da
ta.yaml",
    epochs=50,
    imgsz=640,
    batch=16,
    name="green_mapping_yolo",
    project="training_results",
    exist_ok=True
)
```

The training results indicate that the model achieved an accuracy of 93.83%. This value was obtained after training with 80% of the data for training and 20% for validation. The training process proceeded stably without signs of overfitting, as evidenced by a consistent decrease in loss and an improvement in accuracy metrics. Upon completion of the training, the system generated a model weight file with a .pt extension, named best.pt, which represents the model parameters with the best performance during training. This trained model was then used for inference on new images and for object detection testing in the subsequent evaluation stage.

3) Model Evaluation: The performance of the YOLOv8 model in detecting trees in Google Maps satellite imagery was evaluated using a confusion matrix and several evaluation metric curves (Fig. 5). The confusion matrix was employed to compare the model's predictions against the ground truth across two classes: "tree" and "background." Based on the analysis, the results were as follows: True Positives (TP) 15,963 tree objects correctly detected; False Positives (FP) 1,356 background objects incorrectly classified as trees; and False Negatives (FN) 1,945 tree objects not detected. These findings indicate that the model achieved approximately 92% precision and 89% recall, demonstrating strong detection performance. Although the false-positive rate was low, the 16% false-negative rate suggests that further improvements are needed to ensure the model detects all tree instances. These results are visualized in a normalized confusion matrix, which shows that 84% of tree objects were correctly classified, while the background class was identified with near-perfect accuracy.

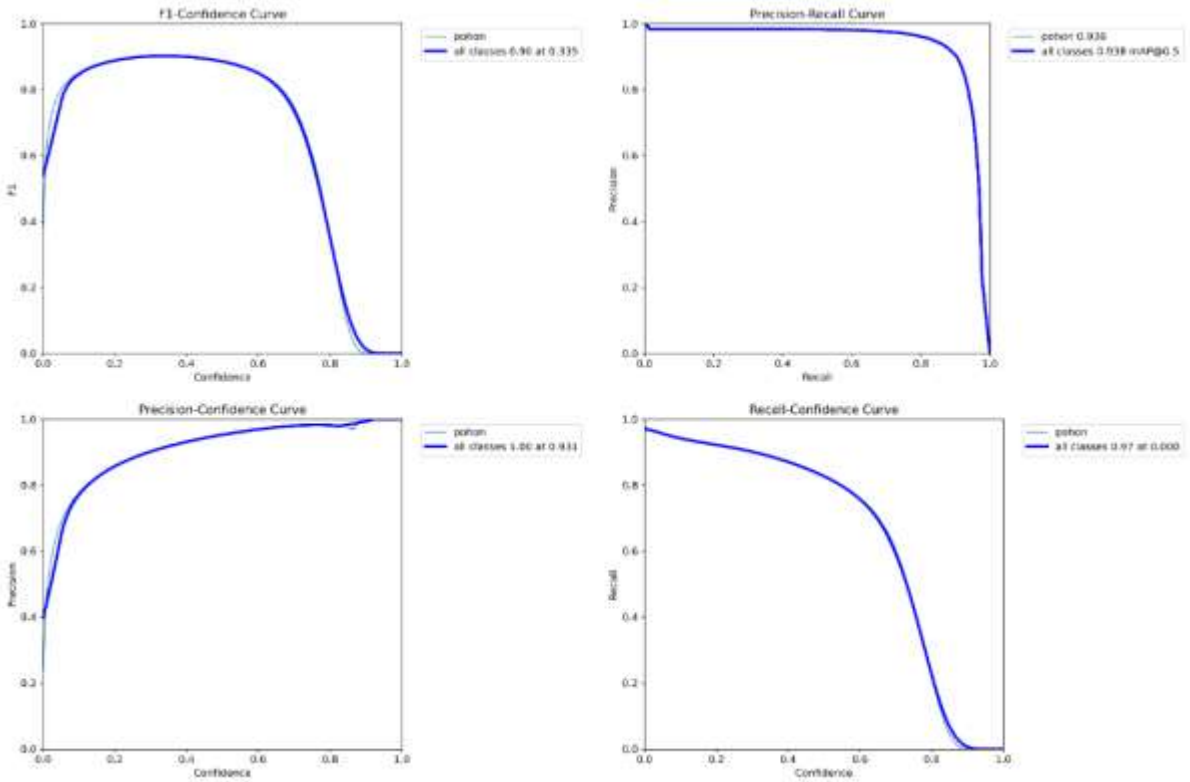

Fig.5. Model Performance

The F1-Confidence curve illustrates the relationship between the confidence threshold and the F1-score. Based on the analysis, the optimal F1-score of 0.90 was achieved at a confidence threshold of 0.335, indicating the best balance between precision and recall. Increasing the threshold beyond this point led to a significant decline in recall, as the model became overly selective, whereas lowering the threshold led to an increase in false positives.

DOI : https://doi.org/10.25139/inform.v11i1.10800

The Precision–Recall curve shows that precision remained above 0.85 even as recall exceeded 0.7, indicating that the model maintained high accuracy despite broad detection coverage. The mAP@0.5 score of 0.938 demonstrates the model's ability to accurately detect the majority of tree objects at an IoU threshold of 50%, without compromising classification precision.

The Precision-Confidence curve illustrates the relationship between confidence values and precision. Precision increases as the confidence threshold rises, reaching 100% at 0.931. However, this is accompanied by a sharp decline in recall, indicating that while the accepted results are highly accurate, the detection coverage becomes limited.

Conversely, the Recall–Confidence curve shows that the highest recall (0.97) was achieved at a threshold of 0.0, but a high number of false positives accompanied it. Once the threshold surpassed 0.6, recall dropped sharply, indicating a trade-off between detection coverage and accuracy. Based on all evaluation curves, the model's best performance occurred at low to medium thresholds, where a balance between precision and recall could be maintained. Overall, the YOLOv8 model demonstrated reliable detection performance, achieving a precision of 0.864, a recall of 0.788, mAP@0.5 of 0.938, and mAP@0.5–0.95 of 0.57.

The training curves also exhibited positive trends, characterized by consistent decreases in box loss, classification loss (cls loss), and distribution focal loss (dfl loss), along with steady improvements in both precision and recall. With 84% of tree objects correctly detected and a very low false-positive rate, the model is highly effective for tree detection, though further refinement is needed to reduce missed objects.

D. Analysis of Data Distribution and Spatial Dispersion

The analysis of data distribution and spatial dispersion was conducted to evaluate both the dataset's quality and the model's performance in detecting tree objects in satellite imagery. Fig.6 shows the distribution of bounding box centre points, evenly spread along the *x* and *y* axes within the normalized range 0 to 1. This uniform dispersion indicates that the model does not exhibit bias toward any specific location when detecting objects.

In addition, the histogram of bounding box sizes shows that most objects are small, with sizes less than 0.2 in the normalized scale, consistent with the visual characteristics of trees in high-resolution satellite imagery. The strong correlation between the width and height of bounding boxes also reflects the consistent shape of the trees recognized by the model, suggesting that the detection process is scale- and proportionate.

The distribution of the "tree" label in the dataset is visualized in Fig. 7, which presents a density map (heatmap). The tree objects appear evenly distributed across the entire image, with no excessive concentration in any specific location, indicating that the dataset does not exhibit spatial bias. The total number of label instances exceeds 60,000, demonstrating that the dataset is both abundant and representative for training the model. Most bounding boxes are concentrated in the smaller size range, which aligns with the typical appearance of trees in satellite imagery. This condition supports the model's effectiveness in detecting small-sized objects with wide spatial coverage.

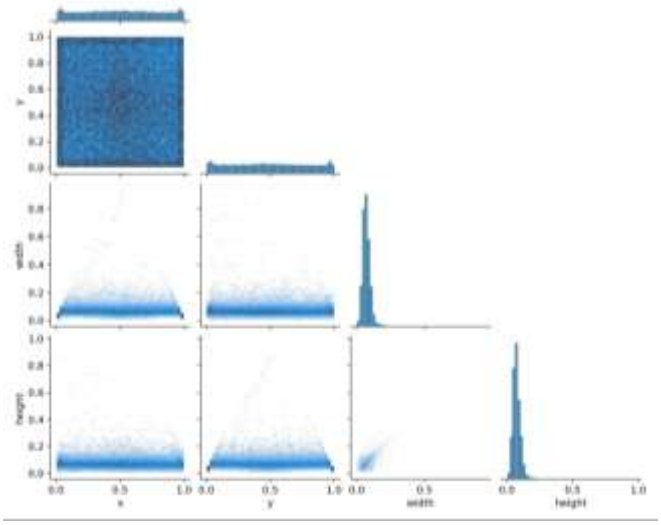

Fig.6. Bounding Box Data Distribution

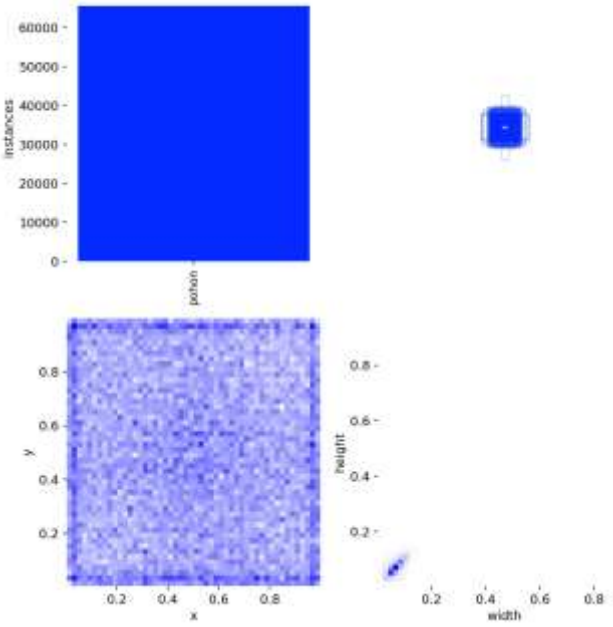

Fig.7. Tree Label Density Map

The model's training performance is illustrated in Fig. 8, which shows the training and validation curves over 50 epochs. These curves clearly exhibit a consistent and positive trend throughout the training process. Specifically, all loss components—box loss, classification loss (cls), and distribution focal loss (dfl)—showed a steady, simultaneous decrease across both the training and validation datasets. This simultaneous reduction indicates that the model was learning effectively without overfitting, as performance improved

DOI : https://doi.org/10.25139/inform.v11i1.10800

across both data subsets rather than just the training set. In parallel with the decreasing loss, the model's evaluation metrics increased notably. Precision increased significantly from approximately 0.60 in the early stages of training to 0.90 by the final epochs, reflecting the model's improved ability to identify tree objects correctly. Similarly, recall rose from 0.60 to 0.88, indicating a higher rate of correctly detected tree instances. Furthermore, the mean average precision at IoU threshold 0.5 (mAP@50) improved markedly from 0.60 to 0.94, while the more stringent metric, mAP@50 95, increased from 0.25 to 0.57. These results collectively demonstrate substantial improvements in the model's detection accuracy and generalization throughout training.

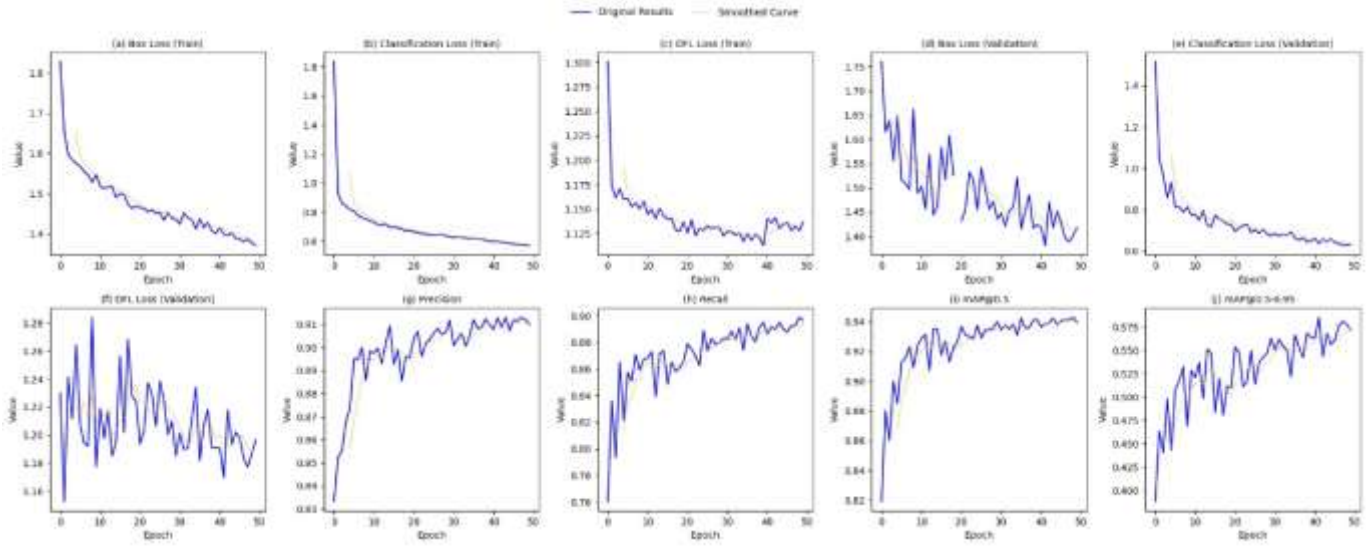

Fig.8. Model Training Performance Curves

Fig. 8 presents the YOLOv8 model training curves over 50 epochs, illustrating changes in metric values and loss for both the training and validation datasets. The graph contains ten curves representing: (a) Box Loss (Train), (b) Classification Loss (Train), (c) Distribution Focal Loss (Train), (d) Box Loss (Validation), (e) Classification Loss (Validation), (f) DFL Loss (Validation), (g) Precision, (h) Recall, (i) mAP@0.5, and (j) mAP@0.5–0.95. The blue line on each curve indicates the actual values for each epoch, while the orange line represents the smoothed trend. The X-axis shows the number of epochs (1-50), and the Y-axis shows the corresponding metric values.

The training results of the YOLOv8 model, shown in Fig. 8, demonstrate a consistent decrease in loss and improvement in performance metrics over the 50 training epochs. The box loss steadily decreased from approximately 1.8 at the beginning to 1.4 at the end of training on both the training and validation sets, indicating increased accuracy in placing bounding boxes close to ground truth. Classification loss (cls loss) also showed a significant reduction from around 1.8 to 0.6, reflecting the model's improved ability to classify objects into the "tree" class correctly. Distribution focal loss (DFL) decreased from 1.3 to 1.14, suggesting stability in the model's bounding-box distribution predictions.

From the perspective of detection performance, the precision curve rose from 0.77 to nearly 0.90, stabilizing after epoch 30, indicating a low false-positive rate. The recall curve also increased significantly from 0.76 to nearly 0.90, meaning that most tree objects were successfully detected. In terms of accuracy metrics, mAP@0.5 reached 0.94, indicating very high detection accuracy at an IoU threshold of 50%, while mAP@0.5–0.95 stabilized at 0.56, demonstrating that the model maintained good performance even at stricter IoU thresholds.

Overall, these results suggest that the model's high precision (~0.90) makes it reliable for monitoring applications by minimizing false detections, while its near-0.90 recall ensures comprehensive tree detection. A high mAP@0.5 value (0.94) confirms excellent detection accuracy, supporting spatial analysis based on Google Maps.

These findings confirm that integrating the YOLOv8 algorithm with Google Maps imagery is highly effective for detecting and mapping trees in urban areas. The model demonstrated robustness even under challenging conditions, including variable lighting, varying vegetation density, and complex image backgrounds. The detection results can be used to accurately distribute green space analysis, which is critical for urban planning and environmental conservation. Additionally, this approach opens opportunities for further development, such as tree species classification or periodic vegetation monitoring to support data-driven environmental management.

E. Testing

Before conducting the analysis, the geographic coordinates of the target area were determined in Google Maps to ensure spatial accuracy and relevance. These coordinates served as the reference points for image retrieval, enabling precise alignment between the detection results and real-world locations. Based

DOI : https://doi.org/10.25139/inform.v11i1.10800

on predefined coordinates, map imagery was systematically retrieved via the Google Maps API, using specific technical settings to ensure data quality and usability. The selected configuration included a zoom level of 19, an image resolution of 640 × 640 pixels, and a scale factor of 2. This setup was deliberately chosen to obtain high-resolution images capable of capturing fine-grained visual details necessary for accurate object detection, particularly for identifying vegetation such as trees. All retrieved images were stored in the designated directory/content/drive/MyDrive/test, serving as the image dataset for the subsequent analysis phase, which involved tree detection using the trained YOLOv8 model.

The result of this image retrieval process, based on the geographic coordinates (3.610622, 98.648552), is illustrated in Fig. 9, which presents a high-resolution visual representation of the test area that complies with the conFig.d parameters and supports the accuracy of spatial analysis in the study. This image also serves as a reference for evaluating the model's detection performance in real-world conditions. The clarity and scale of the imagery ensure that individual tree crowns can be identified, which is critical for validating detection outcomes.

Fig.9. Map Imagery Results Based on Coordinates

After the map imagery was obtained, the next step involved detecting tree objects using the YOLO version 8 (YOLOv8) model. This model used the best weight file (*best.pt*) obtained during the previous training process. The detection process was conducted to identify the presence of trees in the image, based on the class defined during training. The detection results served as the basis for vegetation distribution analysis in the test area.

An example of the detection results is shown in Fig. 10, which compares the image before and after detection. In the detection output, each identified tree is marked with a bounding box, indicating the model's success in classifying the object as the target class.

Fig.10. Example of Image Before and After Detection

The real-time detection process using the trained model is illustrated below. This process demonstrates how the model operates, using the optimized training weights to generate object predictions for new images. Thus, this test confirms that the model can accurately detect trees in high-resolution satellite imagery obtained via the Google Maps API using Code 2.

```
Code 2.
model =
YOLO("/content/drive/MyDrive/training_results/green_
mapping_yolo/weights/best.pt")

model.predict(
    source=image_dir,          # folder berisi gambar
    save=True,                 # simpan gambar hasil
deteksi
    conf=0.3,                  # confidence threshold
    save_txt=True,             # simpan hasil deteksi
ke .txt
    imgsz=640                  # ukuran input YOLO
)
print("Deteksi selesai. Hasil disimpan di
'runs/detect/'")
```

F. Integration with Google Maps

The technical workflow for integrating object detection with spatial visualization involves several key stages. The first stage is object detection using the YOLOv8 model, which produces bounding boxes in the normalized format (*x_center, y_center, width, height*) relative to the image dimensions. These detection results are stored in *.txt* files for each image, following the standard YOLO data storage format.

The second stage involves converting the bounding boxes into geographic coordinates. Since the detection output is pixel-based, a transformation to geographic coordinate systems is required to map the results to real-world locations. This process uses the Web Mercator projection, which takes the image centre coordinates (*center_lat, center_lon*) from the Google Maps API, the image dimensions (e.g., 1280 × 1280 pixels), and the map zoom level (e.g., 19). The conversion formula used is Code 3.

```
Code 3.
lat = bounds['north'] - (y_px / img_h) * lat_range
```


DOI : https://doi.org/10.25139/inform.v11i1.10800

```
lon = bounds['west'] + (x_px / img_w) * lon_range
```

While the conversion results are presented in a JSON array format, as illustrated in Code 4.

```
Code 4.
[
  { "lat": 3.610622, "lng": 98.648552 },
  { "lat": 3.611245, "lng": 98.648911 }
]
```

The third stage is visualizing the converted results on Google Maps using the Google Maps JavaScript API. Each converted coordinate is displayed as a marker on an interactive satellite map, as implemented in the core code shown below. Green markers indicate the positions of trees successfully detected by the model using Code 5.

```
Code 5.
const map =
newgoogle.maps.Map(document.getElementById("map"), {
  zoom: 22,
  center: { lat: 3.610622, lng: 98.648552 },
  mapTypeId: 'satellite'
});

pohon.forEach(p => {
  new google.maps.Marker({
    position: p,
    map: map,
    icon: {
      url:
"https://maps.google.com/mapfiles/kml/pal2/icon12.pn
g",
      scaledSize: new google.maps.Size(20, 20)
    }
  });
});
```

The fourth stage involves enabling map interactivity, including zooming and panning, so users can enlarge the map view and navigate to different areas. The green markers play a key role in providing a clear visualization of tree distribution within the analyzed region.

This integration offers several benefits for spatial analysis, including facilitating the identification of areas with low Green Open Space (GOS) coverage to support urban greening plans, enabling the incorporation of population density data to prioritize green space development, and providing a foundation for a real-time vegetation monitoring dashboard.

The integration of tree detection results with the Google Maps JavaScript API produced an interactive map (Fig.11) that visualizes vegetation distribution in the Medan City area, with each green marker representing the geographic coordinates of trees detected by the YOLOv8 model. This visualization offers added value over image outputs with bounding boxes, as it enables coordinate-based spatial analysis to support urban spatial planning. In addition, the map facilitates the identification of areas with low tree density, allowing greening interventions to be prioritized. It can also be integrated with supplementary data such as population density or air pollution levels to support evidence-based policy-making. From a technical perspective, the mapping can be extended into a heatmap layer using the Google Maps API to provide a more intuitive visualization of vegetation density.

Fig.11. Visualization of Tree Detection Results Converted to Geographic Coordinates Displayed on Google Maps

The system also supports periodic monitoring of vegetation changes by detecting differences across time intervals, which is beneficial for tracking vegetation degradation or growth. Moreover, the detected coordinate data can be exported in GeoJSON or CSV format for further analysis using Geographic Information Systems (GIS) platforms such as QGIS or ArcGIS. Future development may focus on building an interactive dashboard with area- or distance-based filters, integrating real-time monitoring via automated data update pipelines, and conducting sustainability analysis by combining spatial data with environmental indicators to evaluate the contribution of green open spaces to achieving the Sustainable Development Goals (SDGs).

The results of this study, which show strong detection performance of the YOLOv8 model when applied to Google Maps satellite imagery, are consistent with, yet distinct from, findings in several related studies. For instance, a study on orchard-tree detection across multiple observation dates found that YOLOv3 performed poorly. At the same time, Mask R-CNN and the Segment Anything Model (SAM) achieved higher precision, recall, and F1 scores. All models, including YOLO, struggled during early-season or hazy conditions, suggesting that standard YOLO architectures may be sensitive to seasonal and atmospheric variations, potentially leading to incomplete or noisy detection results when applied to real-world mapping [33]. Similarly, research based on the VHRTrees dataset reported that model performance varied considerably across geographic regions due to differences in canopy heterogeneity and image resolution. A model trained in one area often failed

DOI : https://doi.org/10.25139/inform.v11i1.10800

to generalize well to other regions without additional fine-tuning, indicating that although the YOLOv8 model performed effectively in Medan City, its transferability to other regions should be validated before large-scale deployment [34].

In a comparative analysis, it was observed that YOLO-based counting models tend to overestimate object counts relative to other detection methods, leading to a higher false-positive rate, particularly when canopies are dense or backgrounds are complex. This implies that, when integrated into Google Maps for tree-count visualization, YOLO-based detection could over-represent the actual number of trees relative to ground truth [35].

Further studies have emphasized that YOLO's baseline variants often struggle with small-object detection unless enhanced through techniques such as image tiling, SAHI (Slicing Aided Hyper Inference), negative sampling, or model variants like BiFPN-YOLOv8m or YOLOv8-MFD. These improvements help mitigate background complexity and scale variation issues. Without such enhancements, YOLO's detection quality was found to be unsatisfactory, suggesting that the success of this study's YOLOv8 implementation may be partly due to pre-processing and model optimization strategies [36].

Finally, comparative analyses between YOLOv8 and segmentation-based approaches, such as DeepForest, demonstrated that YOLOv8 was superior in speed and bounding-box localization but inferior in fine-grained crown delineation and attribute identification. Consequently, while this study's marker-based visualization on Google Maps helps map tree locations, it may not fully capture canopy boundaries or biophysical parameters that segmentation methods can [37].

Overall, these findings highlight both the strengths and limitations of YOLO-based tree detection: its robustness for fast, large-scale spatial monitoring, but reduced accuracy under varying illumination, complex backgrounds, and fine-scale canopy differentiation. Future research should therefore explore hybrid approaches combining YOLO detection with segmentation or multispectral analysis to achieve more comprehensive environmental assessments.

IV. CONCLUSION

This study successfully developed a tree detection system using the YOLOv8 algorithm, integrated with Google Maps satellite imagery, to support the monitoring of green open spaces (GOS) in Medan City. The YOLOv8 model was trained on Google Colaboratory for 50 epochs with a batch size of 16 and the Stochastic Gradient Descent (SGD) optimizer. The model demonstrated excellent performance, as evidenced by a precision score of 0.864, a recall of 0.788, a mean average precision at 0.5 (mAP@0.5) of 0.938, and a mAP@0.5–0.95 of 0.57. The training curves showed a trend of decreasing loss and increasing precision and recall, indicating the model did not overfit. Evaluation using the confusion matrix showed that 84% of tree objects were correctly detected, with a very low false-positive rate.

Furthermore, the bounding box distribution analysis confirmed an even spread and a predominance of small objects, consistent with the typical characteristics of vegetation in satellite imagery. These findings highlight the model's effectiveness in detecting small tree objects under varying lighting conditions and vegetation densities. The testing phase conducted on new satellite images demonstrated the model's real-time detection capability with high accuracy. Detected tree locations were successfully converted to geographic coordinates (latitude and longitude) and visualized using the Google Maps JavaScript API. This interactive spatial analysis enables more precise interpretation of vegetation distribution throughout Medan City and offers valuable support for evidence-based decision-making in urban spatial planning and environmental management.

REFERENCES

[1] W. Ahmad, R. Shokeen, and R. Raj, "Artificial Intelligence: Solutions in Special Education," in *Advances in Educational Technologies and Instructional Design*, A. G. Walters, Ed., IGI Global, 2024, pp. 459–520, doi: 10.4018/979-8-3693-5538-1.ch017.

[2] A. V. Abdussalam and G. A. Auladi, "Pushing Boundaries: AI and Computer Science in the Era of Technological Revolution," *TechCompInnovations*, vol. 1, no. 1, pp. 1–9, Jun. 2024, doi: 10.70063/techcompinnovations.v1i1.22.

[3] M. Yao, "Applications of Artificial Intelligence in Computer Vision and Network Fields," *GBP Proc. Ser.*, vol. 1, pp. 64–71, Jan. 2025, doi: 10.71222/e08y3a92.

[4] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," Apr. 23, 2020, *arXiv:2004.10934*, doi: 10.48550/arXiv.2004.10934.

[5] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," Apr. 8, 2018, *arXiv:1804.02767*, doi: 10.48550/arXiv.1804.02767.

[6] I. C. Wicaksana, R. A. Pramunendar, G. W. Saraswati, and G. A. Trisnapradika, "Skin Lesion Classification Using YOLOv11 on the HAM10000 Dataset," *Inform: Jurnal Ilmiah Bidang Teknologi Informasi dan Komunikasi*, vol. 10, no. 1, pp. 45–52, 2025, doi: 10.25139/inform.v10i1.9310.

[7] K. V. Chand, K. Ashwini, M. Yogesh, and G. P. Jyothi, "Object Detection in Remote Sensing Images Using YOLOv8," in *Proc. 2024 Int. Conf. Integrated Intelligence and Communication Systems (ICIICS)*, Kalaburagi, India, Nov. 2024, pp. 1–6, doi: 10.1109/ICIICS63763.2024.10859885.

[8] M. Sohan, T. S. Ram, and C. V. R. Reddy, "A Review on YOLOv8 and Its Advancements," in *Data Intelligence and Cognitive Informatics*, I. J. Jacob, S. Piramuthu, and P. Falkowski-Gilski, Eds., Algorithms for Intelligent Systems, Singapore: Springer Nature, 2024, pp. 529–545, doi: 10.1007/978-981-99-7962- 2_39.

[9] R. Varghese and S. M., "YOLOv8: A Novel Object Detection Algorithm with Enhanced Performance and Robustness," in *Proc. 2024 Int. Conf. Advances in Data Engineering and Intelligent Computing Systems (ADICS)*, Chennai, India, Apr. 2024, pp. 1–6, doi: 10.1109/ADICS58448.2024.10533619.

[10] T. Wu and Y. Dong, "YOLO-SE: Improved YOLOv8 for Remote Sensing Object Detection and Recognition," *Applied Sciences*, vol. 13, no. 24, p. 12977, Dec. 2023, doi: 10.3390/app132412977.

[11] C. Silver, "Rapid Urbanization: The Challenges and Opportunities for Planning in Indonesian Cities," in *The Indonesian Economy and the Surrounding Regions in the 21st Century*, B. P. Resosudarmo and Y. Mansury, Eds., *New Frontiers in Regional Science: Asian Perspectives*, vol. 76, Singapore: Springer Nature, 2024, pp. 35–48, doi: 10.1007/978-981-97-0122-3_3.

[12] A. G. M. Valones and U. A. Junaedi, "Urban Green Space Policy Reform in Indonesia: Breathing in the Middle of Development," *JLLR*, vol. 4, no. 2, pp. 183–210, Apr. 2023, doi: 10.15294/jllr.v4i2.66228.

[13] N. Guillerm and G. Cesari, "Fighting Ambient Air Pollution and Its Impact on Health: From Human Rights to the Right to a Clean Environment,"

DOI : https://doi.org/10.25139/inform.v11i1.10800

Int. J. Tuberc. Lung Dis., vol. 19, no. 8, pp. 887–897, Aug. 2015, doi: 10.5588/ijtld.14.0660.

[14] S. Ali, T. Ahmed, S. K. Dwivedi, A. Singh, and A. K. Shrivastava, "Impact of Urban Vegetation in Sustainable Smart Cities Development: A Comprehensive Appraisal," *Int. J. Innovative Research in Computer Science and Technology (IJIRCST)*, pp. 184–192, Mar. 2024, doi: 10.55524/CSISTW.2024.12.1.33.

[15] F. J. Tapiador, A. Navarro, J. Mezo, S. De La Llave, and J. Muñoz, "Urban Vegetation Leveraging Actions," *Sustainability*, vol. 13, no. 9, p. 4843, Apr. 2021, doi: 10.3390/su13094843.

[16] G. Zulian, F. Marando, L. Mentaschi, C. Alzetta, B. Wilk, and J. Maes, "Green Balance in Urban Areas as an Indicator for Policy Support: A Multi-Level Application," *One Ecosystem*, vol. 7, p. e72685, Mar. 2022, doi: 10.3897/oneeco.7.e72685.

[17] D. L. Garner, H. B. Underwood, and W. F. Porter, "Use of Modern Infrared Thermography for Wildlife Population Surveys," *Environmental Management*, vol. 19, no. 2, pp. 233–238, Mar. 1995, doi: 10.1007/BF02471993.

[18] J. K. Gillan, G. E. Ponce-Campos, T. L. Swetnam, A. Gorlier, P. Heilman, and M. P. McClaran, "Innovations to Expand Drone Data Collection and Analysis for Rangeland Monitoring," *Ecosphere*, vol. 12, no. 7, p. e03649, Jul. 2021, doi: 10.1002/ecs2.3649.

[19] M. Lesiv et al., "Characterizing the Spatial and Temporal Availability of Very High Resolution Satellite Imagery in Google Earth and Microsoft Bing Maps as a Source of Reference Data," *Land*, vol. 7, no. 4, p. 118, Oct. 2018, doi: 10.3390/land7040118.

[20] D. D. Wilson, G. W. Tefera, and R. L. Ray, "Application of Google Earth Engine to Monitor Greenhouse Gases: A Review," *Data*, vol. 10, no. 1, p. 8, Jan. 2025, doi: 10.3390/data10010008.

[21] R. Apsara and S. R. Harikrishnan, "Object Detection in Satellite Images Using Computer Vision Models," *Int. J. Adv. Res. Sci. Comput. Technol. (IJARSCT)*, pp. 366–372, Aug. 2024, doi: 10.48175/IJARSCT-19255.

[22] M. Lin, "Civil Aviation Satellite Navigation Integrity Monitoring with Deep Learning," *Adv. Comput. Commun. (ACC)*, vol. 4, no. 4, pp. 260–264, Sep. 2023, doi: 10.26855/acc.2023.08.008.

[23] B. S. Neyigapula, "Integrating Deep Learning Algorithms with Satellite Images: Revolutionizing Earth Observation," *In Review*, Jul. 27, 2023, doi: 10.21203/rs.3.rs-3195683/v1.

[24] B. Khalili and A. W. Smyth, "SOD-YOLOv8: Enhancing YOLOv8 for Small Object Detection in Aerial Imagery and Traffic Scenes," *Sensors*, vol. 24, no. 19, p. 6209, Sep. 2024, doi: 10.3390/s24196209.

[25] W. Zhou, C. Zhu, and D. Miao, "Object Detection Model of YOLOv8-CSD for UAV Images," in *Proc. 2024 7th Int. Conf. Pattern Recognition and Artificial Intelligence (PRAI)*, Hangzhou, China, Aug. 2024, pp. 77–83, doi: 10.1109/PRAI62207.2024.10826612.

[26] Y. Zheng and G. Wu, "YOLOv4-Lite–Based Urban Plantation Tree Detection and Positioning with High-Resolution Remote Sensing Imagery," *Front. Environ. Sci.*, vol. 9, p. 756227, Jan. 2022, doi: 10.3389/fenvs.2021.756227.

[27] K. Itakura and F. Hosoi, "Automated Tree Detection from 3D LiDAR Images Using Image Processing and Machine Learning," *Appl. Opt.*, vol. 58, no. 14, p. 3807, May 2019, doi: 10.1364/AO.58.003807.

[28] Ş. N. Topgül, E. Sertel, S. Aksoy, C. Ünsalan, and J. E. S. Fransson, "VHRTrees: A New Benchmark Dataset for Tree Detection in Satellite Imagery and Performance Evaluation with YOLO-Based Models," *Front. For. Glob. Change*, vol. 7, p. 1495544, Jan. 2025, doi: 10.3389/ffgc.2024.1495544.

[29] L. Velasquez-Camacho et al., "Monitoring Temporal Changes in Large Urban Street Trees Using Remote Sensing and Deep Learning," *PLoS One*, vol. 20, no. 6, p. e0326562, Jun. 2025, doi: 10.1371/journal.pone.0326562.

[30] T. Yang, S. Zhou, A. Xu, J. Ye, and J. Yin, "YOLO-SegNet: A Method for Individual Street Tree Segmentation Based on the Improved YOLOv8 and the SegFormer Network," *Agriculture*, vol. 14, no. 9, p. 1620, Sep. 2024, doi: 10.3390/agriculture14091620.

[31] G. F. Mumtaz, J. Zeniarja, A. Luthfiarta, and A. N. I. Muttaqin, "Optimizing Face Recognition and Emotion Detection in Student Identification Using FaceNet and YOLOv8 Models," *Inform: Jurnal Ilmiah Bidang Teknologi Informasi dan Komunikasi*, vol. 10, no. 1, pp. 34–44, 2025, doi: 10.25139/inform.v10i1.9304.

[32] *Ultralytics*, "YOLOv8 Documentation," [Online]. Available: https://docs.ultralytics.com/

[33] M. Kelly, *et al.*, "Mapping Orchard Trees from UAV Imagery Through One-Shot, Two-Shot and Zero-Shot CNNs," *Drones*, vol. 9, no. 9, Art. no. 593, 2025. doi:10.3390/2504-446X/9/9/593.

[34] Ş. N. Topgül, E. Sertel, S. Aksoy, C. Ünsalan, and J. E. S. Fransson, "VHRTrees: a new benchmark dataset for tree detection in satellite imagery and performance evaluation with YOLO-based models," *Frontiers in Forests and Global Change*, vol. 7, Art. no. 1495544, 2024. doi:10.3389/ffgc.2024.1495544.

[35] M. R. Karim, M. N. Reza, S. Ahmed, K.-H. Lee, J. Sung, and S.-O. Chung, "Detection of Trees and Objects in Apple Orchard from LiDAR Point Cloud Data Using a YOLOv5 Framework," *Electronics*, vol. 14, no. 13, Art. no. 2545, 2025. doi:10.3390/electronics14132545.

[36] A. Jiang, *et al.*, "Tree Trunk Recognition in Orchard Autonomous Systems Using Thermal Camera and Faster R-CNN," *Sensors*, vol. 22, no. 5, Art. no. 2065, 2022. doi:10.3390/s22052065.

[37] M. Pérez-Carrasco, B. Karelovic, R. Molina, R. Saavedra, P. Cerulo, and G. Cabrera-Vives, "Precision silviculture: use of UAVs and comparison of deep learning models for the identification and segmentation of tree crowns in pine crops," *International Journal of Digital Earth*, vol. 15, no. 1, pp. 2223–2238, 2023. doi:10.1080/17538947.2022.2152882.

DOI : https://doi.org/10.25139/inform.v11i1.10800