Enhancing Object Detection and Classification in High-Resolution Images Using SAHI Algorithm and Modern Neural Networks

Oleksii Bychkov[1], Kateryna Merkulova[1,†], Yelyzaveta Zhabska[1*,†] and Andrii Yaroshenko[1,†]

[1] *Taras Shevchenko National University of Kyiv, Volodymyrska str. 64/13, Kyiv, 01601, Ukraine*

Abstract

This paper presents a novel approach to address the challenges of object detection and classification in high-resolution images by combining the Slicing Aided Hyper Inference (SAHI) algorithm with modern neural networks. The proposed method involves slicing high-resolution images into smaller patches, which are then processed by five state-of-the-art neural networks: YOLOv5, YOLOv8, YOLOX, Torchvision, and RetinaNet. Experimental results on a high-resolution images dataset demonstrate the effectiveness of the proposed approach in terms of both accuracy and efficiency. The influence of various SAHI parameters on the performance of object detection is also investigated. The developed software with a user-friendly interface allows to make easy adaptation of the proposed approach to a wide range of practical applications. The presented solution offers a promising direction for efficient object detection and classification in high-resolution images.

Keywords

object detection, image classification, high-resolution images, SAHI algorithm, neural networks, computer vision

1. Introduction

The increasing availability of high-resolution images in various domains has led to a growing need for efficient object detection and classification methods. However, processing such images poses significant challenges due to their large size and the presence of small objects. Traditional object detection methods often struggle to efficiently handle high-resolution images, resulting in high computational costs and suboptimal performance. To address these challenges, we propose a novel approach that combines the Slicing Aided Hyper Inference (SAHI) algorithm with an ensemble of modern neural networks. The SAHI algorithm involves slicing the high-resolution image into smaller patches, which are then processed independently by the object detection models. This approach enables more efficient processing of large images and improves detection of small objects.

In this paper, we integrate the SAHI algorithm with five state-of-the-art neural networks: YOLOv5, YOLOv8, YOLOX, Torchvision, and RetinaNet. These networks have demonstrated impressive performance on various object detection tasks. By combining them with the SAHI algorithm, we aim to leverage their strengths while addressing the challenges posed by high-resolution images.

To evaluate the effectiveness of the proposed approach, we conduct extensive experiments on a dataset containing high-resolution images of various scenes, such as beaches and bays. The experiments focus on assessing the accuracy and efficiency of object detection and classification

Information Technology and Implementation (IT&I-2024), November 20-21, 2024, Kyiv, Ukraine

* Corresponding author.
† These authors contributed equally.

bos.knu@gmail.com (O. Bychkov); kate.don11@gmail.com (K. Merkulova); y.zhabska@gmail.com (Y. Zhabska); andrii.yaroshenko@knu.ua (A. Yaroshenko)

0000-0002-9378-9535 (O. Bychkov); 0000-0001-6347-5191 (K. Merkulova); 0000-0002-9917-3723 (Y. Zhabska); 0009-0002-1598-8261 (A. Yaroshenko)

using different combinations of neural networks and SAHI parameters. Furthermore, we investigate the influence of various SAHI parameters, such as tile size and overlap, on the performance of object detection. By systematically varying these parameters, we aim to provide guidelines for their optimal selection, enabling users to adapt the proposed approach to their specific needs.

2. Related Works

Object detection and classification in high-resolution images have been the focus of numerous studies in the field of computer vision. Traditional approaches, such as sliding window and image pyramids, have been widely used for this task. However, these methods' performance often suffers from high computational complexity and limited scalability, making them unsuitable for real-time applications.

In recent years, deep learning-based approaches, particularly convolutional neural networks (CNNs), have revolutionized the field of object detection. Architectures like YOLO (You Only Look Once), SSD (Single Shot MultiBox Detector), and Faster R-CNN have achieved remarkable performance on benchmark datasets. However, these networks face several challenges when applied to high-resolution images.

One of the main limitations of neural networks in processing high-resolution images is their fixed input size requirement. To accommodate this, images are typically resized or cropped before being given to the input of the network. This resizing process can lead to a loss of information, especially for small objects, which may become too small to be detected after resizing. Moreover, resizing large images to fit the network's input size can be computationally expensive and time-consuming [1].

Another challenge faced by neural networks in high-resolution object detection is their difficulty in detecting small objects. Even when trained on datasets specifically designed for this task, neural networks often struggle to capture sufficient visual information from small objects, leading to suboptimal performance. This problem is further exacerbated in high-resolution images, where objects of interest may occupy only a small portion of the image.

Furthermore, processing high-resolution images with neural networks demands significant computational resources, including memory and processing power. As the image size increases, the number of computations required by the network grows exponentially, making it challenging to process large images in real-time or on resource-constrained devices.

To address these limitations, various approaches have been proposed in the literature. One such approach is the use of image pyramids [2], where the image is resized to multiple scales, and object detection is performed at each scale. However, this approach can be computationally expensive and may still miss small objects.

Another approach is the use of selective search [3], where the image is segmented into regions, and object detection is performed on each region. Despite the fact that this approach can handle objects of different sizes, it can be time-consuming and may generate a large number of false positives.

Sliding window techniques have also been employed for object detection in high-resolution images [4]. In this approach, a window of fixed size is slid over the image, and object detection is performed at each window location. However, this approach can be computationally expensive, especially for large images, and may struggle with objects of varying sizes.

More recently, the concept of slicing images into smaller patches has gained attention as a potential solution to the challenges of high-resolution object detection [5]. By processing smaller patches independently, the computational burden can be reduced, and the detection of small objects can be improved. However, the effectiveness of this approach depends on the proper selection of patch size and the handling of objects that span multiple patches.

Despite the progress made in object detection and classification in high-resolution images, there remains a need for more efficient and effective solutions. The proposed approach in this paper aims to address this need by combining the Slicing Aided Hyper Inference (SAHI) algorithm with an ensemble of modern neural networks. By leveraging the strengths of both techniques, the proposed approach is aimed to overcome the limitations of traditional neural networks and provide a scalable and accurate solution for object detection in high-resolution images.

3. Dataset Acquisition

To evaluate the effectiveness of the proposed approach for object detection and classification in high-resolution images, a suitable dataset is required. However, acquiring a large-scale dataset of high-resolution images with annotated objects can be challenging. Publicly available datasets, such as COCO (Common Objects in Context) and PASCAL VOC, often contain images of relatively low resolution, which may not adequately represent the challenges faced in real-world high-resolution object detection tasks.

To address this issue, a custom dataset was collected specifically for this study. The dataset consists of high-resolution panoramic images sourced from the web. Panoramic images were chosen because they offer a wide field of view and capture a large amount of visual information, making them suitable for testing object detection algorithms in complex scenes.

The dataset acquisition process involved several steps. First, a list of potential sources for high-resolution panoramic images was compiled. These sources included online repositories, such as Gigapan, 360cities, and Google Street View, as well as individual photographers' websites and social media platforms.

Next, a set of search queries was formulated to identify relevant images within these sources. The queries included keywords related to specific scenes, such as beaches, bays, cityscapes, and landmarks, as well as terms indicating the presence of objects of interest, such as people, vehicles, and boats.

The search results were then manually reviewed to select images that met the following criteria:

1. High resolution: the images should have a minimum resolution of 0.1 gigapixels to ensure sufficient resolution to be called "high-resolution image".
2. Diversity: the selected images should cover a wide range of scenes, locations, and object types to assess the generalization capability of the proposed approach.
3. Clarity: the images should be of good quality, with minimal blur, distortion, or other artifacts that could hinder object detection performance.
4. Licensing: the images should be available under licenses that allow their usage in research and publication.

After the initial selection, the images were downloaded in their original resolution and format. However, it was observed that many of the panoramic images were not provided as a single file but rather as a set of tiles that needed to be stitched together to form the complete image.

To address this issue, a custom script was developed to automate the stitching process. The script utilized the metadata associated with each tile, such as its position and dimensions, to determine the correct arrangement of the tiles. The tiles were then loaded into memory and combined using image processing techniques, such as blending and feathering, to create a seamless high-resolution panorama.

Once the panoramic images were stitched, they were manually inspected to ensure the quality of the stitching process. Images with visible seams, misalignments, or other artifacts were discarded, and additional images were collected to replace them.

The final dataset consists of 10 high-resolution panoramic images, with an average resolution of 0.7 gigapixels. The images cover a diverse range of scenes, including beaches, bays, cityscapes, and landmarks, and contain various objects of interest, such as people, vehicles, boats, and buildings.

4. Methodology

4.1. SAHI Algorithm

The Slicing Aided Hyper Inference (SAHI) algorithm [6] is a key component of the proposed approach for object detection and classification in high-resolution images. The main idea behind SAHI is to divide the large input image into smaller, overlapping patches, which can then be processed independently by the object detection models. This approach has several advantages over traditional methods that rely on resizing or cropping the image to fit the input size of the neural network.

By processing smaller patches, the SAHI algorithm can effectively handle high-resolution images without the need for resizing, which often leads to a loss of information, especially for small objects. Each patch is fed into the neural network at its original resolution, preserving the fine details necessary for accurate object detection.

The SAHI algorithm can be easily parallelized, as each patch can be processed independently by a separate instance of the object detection model. This parallelization can significantly speed up the inference process, making it more suitable for real-time applications.

The SAHI algorithm consists of the following steps:

1. Slicing: the input image is divided into smaller, overlapping patches of a fixed size. The size of the patches and the amount of overlap between them are hyperparameters that can be adjusted based on the specific requirements of the task. In this study, we experiment with patch sizes of 256×256, 512×512, and 1024×1024 pixels, with overlap ratios of 0.25 and 0.5.
2. Inference: each patch is independently processed by the object detection model, which outputs a set of bounding boxes and corresponding class probabilities for the objects detected within the patch. In this study, we evaluate five state-of-the-art object detection models, namely YOLOv5, YOLOv8, YOLOX, Torchvision, and RetinaNet.
3. Merging: the bounding boxes and class probabilities obtained from each patch are combined to form the final set of detections for the entire image. This merging process involves resolving any duplicate detections that may occur in the overlapping regions between patches. Several strategies can be employed for this purpose, such as non-maximum suppression (NMS), which retains only the bounding box with the highest class probability among a set of overlapping detections.
4. Post-processing: the merged detections are further refined through post-processing techniques, such as thresholding based on class probabilities and adjusting bounding box coordinates to account for the original image size.

The SAHI algorithm provides a flexible and efficient framework for object detection in high-resolution images, enabling the use of existing state-of-the-art object detection models without the need for extensive modifications.

4.2. Role of NMS in the SAHI algorithm

Non-Maximum Suppression (NMS) [7] is a crucial post-processing step in object detection algorithms, including the SAHI algorithm. Its primary purpose is to eliminate redundant and overlapping bounding boxes, keeping only the most confident detection for each object in the image.

In object detection, the model typically generates a large number of bounding boxes, many of which may belong to the same object. This is particularly common in sliding window-based approaches, such as the SAHI algorithm, where the image is divided into overlapping patches. Each patch is processed independently, leading to multiple detections for objects that appear in multiple patches.

NMS addresses this issue by suppressing less confident detections that significantly overlap with more confident ones. The algorithm works as follows:

1. Sort the detected bounding boxes in descending order of their confidence scores.
2. Select the bounding box with the highest confidence score and add it to the list of final detections.
3. Compare the selected bounding box with all the remaining boxes and calculate their Intersection over Union (IoU) scores.
4. Remove any bounding box that has an IoU score higher than a predefined threshold (typically 0.5) with the selected box.
5. Repeat steps 2-4 until all bounding boxes have been either selected or suppressed.

The IoU score measures the overlap between two bounding boxes and is calculated as the area of their intersection divided by the area of their union. A high IoU score indicates that the two bounding boxes significantly overlap and likely belong to the same object [8]:

$$IoU = \frac{area(B_1 \cap B_2)}{area(B_1 \cup B_2)}, \tag{1}$$

where B_1 and B_2 is the bounding boxes of detections.

NMS is crucial for several reasons:

1. Improved precision: by removing redundant detections, NMS helps improve the precision of the object detection algorithm. Precision measures the percentage of detected objects that are actually correct, and reducing false positives is essential for achieving high precision.
2. Reduced clutter: NMS helps declutter the output of the object detection algorithm, making it easier to interpret and use the results. Without NMS, the output would be overwhelmed by numerous overlapping bounding boxes, making it difficult to distinguish individual objects.
3. Increased efficiency: by reducing the number of bounding boxes, NMS helps improve the efficiency of downstream processes that rely on the object detection results, such as tracking or counting objects. Processing fewer bounding boxes requires less computational resources and can lead to faster overall pipeline performance.
4. Better user experience: NMS helps provide a cleaner and more intuitive visual representation of the detected objects, which is particularly important for applications that involve human interaction, such as video surveillance or autonomous driving.

In the context of the SAHI algorithm, NMS plays a vital role in merging the detections from multiple patches into a coherent set of final detections. Without NMS, the SAHI algorithm would produce numerous duplicate detections for objects that appear in multiple patches, leading to a cluttered and imprecise output.

By applying NMS with a carefully chosen IoU threshold, the SAHI algorithm can effectively merge the detections from multiple patches and provide a clean and accurate set of final detections. The IoU threshold is a hyperparameter that can be tuned based on the specific characteristics of the dataset and the desired balance between precision and recall.

In summary, Non-Maximum Suppression is a crucial post-processing step in object detection algorithms, including the SAHI algorithm. It helps improve precision, reduce clutter, increase efficiency, and provide a better user experience by eliminating redundant and overlapping bounding boxes. NMS is particularly important for the SAHI algorithm, as it enables the effective merging of detections from multiple patches into a coherent set of final detections.

4.3. SAHI Algorithm

4.3.1. Basic Non-Maximum Suppression (NMS)

The basic Non-Maximum Suppression (NMS) algorithm [7] works as follows:

- Score Sorting: sort all bounding boxes by their confidence scores in descending order.
- Selection: select the box with the highest score and remove it from the list.
- Overlap Removal: remove all other boxes that have an Intersection over Union (IoU) greater than a predefined threshold with the selected box.
- Repetition: repeat the process until no more boxes remain.

This algorithm ensures that only the most confident and least overlapping bounding boxes are retained.

4.3.2. GREEDYNMM (Greedy Non-Maximum Merging)

Greedy Non-Maximum Merging (GREEDYNMM) [9] is an algorithm designed to improve upon basic NMS by merging overlapping boxes rather than simply removing them. Here's how it works:

- Score Sorting: sort all bounding boxes by their confidence scores in descending order.
- Selection: select the box with the highest score.
- Merging: merge this box with all other boxes that have an IoU greater than a certain threshold. The merging process involves averaging the coordinates of the overlapping boxes weighted by their confidence scores.
- Update: replace the selected box with the merged box and remove the other overlapping boxes.
- Repetition: repeat the process until no more boxes overlap significantly.

GREEDYNMM helps retain more information by merging boxes rather than discarding them, which can be beneficial in densely populated object scenarios.

4.3.3. NMM (Non-Maximum Merging)

Non-Maximum Merging (NMM) [10] is similar to GREEDYNMM but uses a different strategy for merging:

- Score Sorting: sort bounding boxes by confidence scores.
- Selection: select the highest-scoring box.
- Merging: for boxes with IoU above the threshold, calculate the weighted average of the box coordinates and confidence scores.
- Replacement: replace the selected box with the merged result and remove the overlapping boxes.
- Repetition: continue this process until all boxes have been processed.

NMM focuses on maintaining spatial accuracy and consistency by carefully merging overlapping boxes.

4.3.4. Large Scale Non-Maximum Suppression (LSNMS)

Large Scale Non-Maximum Suppression (LSNMS) is an optimized algorithm designed to address the inefficiencies of traditional Non-Maximum Suppression (NMS) when working with large-scale image data [11]. This method significantly speeds up the NMS process, especially for high-dimensional images and large numbers of bounding boxes.

Key Features of LSNMS:

- R-Tree Structure: LSNMS constructs an R-Tree on the bounding boxes before starting the NMS process. The R-Tree structure allows for efficient querying of overlapping boxes in logarithmic time, reducing the complexity of the NMS process.
- Complexity Reduction: traditional NMS has a worst-case quadratic time complexity, which can be prohibitive with large numbers of boxes. LSNMS reduces this to $O(n \log(n))$ by only considering boxes that are spatially close to each other during the suppression steps.
- Handling Large Images: when dealing with large images (e.g., satellite or histology images), LSNMS handles the patching of images and applies NMS independently to each patch. A final NMS step is performed to consolidate results from overlapping patches, ensuring accurate detection without redundant computations.
- Implementation: LSNMS is implemented using Numba's just-in-time compilation for efficient computation. This method ensures that even the tree-building process and subsequent NMS steps are executed swiftly.
- Multiclass Support: LSNMS also supports multiclass NMS by offsetting bounding boxes in a way that minimizes query times and maximizes the efficiency of the R-Tree structure.
- Performance: LSNMS offers significant speed improvements over traditional NMS. For example, on 40k×40k pixel images with about 300,000 bounding boxes, naive NMS took approximately 5 minutes on a modern CPU, whereas LSNMS completed in just 5 seconds, providing nearly a 60 times speedup.

In this paper, we will evaluate each NMS algorithm for its accuracy.

4.4. Object Detection Models

In this study, we evaluate five state-of-the-art object detection models in combination with the SAHI algorithm:

1. YOLOv5 [12]: YOLOv5 is a single-stage object detector that builds upon the success of previous YOLO (You Only Look Once) models. It achieves real-time inference speeds while maintaining high accuracy, making it suitable for various applications. YOLOv5 utilizes a novel backbone network, a feature pyramid network (FPN) for multi-scale feature fusion, and an anchor-free detection head.
2. YOLOv8 [13]: YOLOv8 is an improved version of YOLOv5, featuring a redesigned architecture and enhanced training techniques. It achieves state-of-the-art performance on several object detection benchmarks while maintaining real-time inference capabilities.
3. YOLOX [14]: YOLOX is an anchor-free variant of the YOLO family of object detectors. It introduces a decoupled head design, where classification and localization are performed separately, leading to improved accuracy and flexibility. YOLOX also incorporates advanced data augmentation techniques and a novel loss function to enhance its performance.
4. Torchvision [15]: Torchvision is a popular computer vision library that provides a collection of pre-trained models for various tasks, including object detection. In this study, we use the Faster R-CNN model with a ResNet-50 backbone, which has demonstrated strong performance on several benchmark datasets.
5. RetinaNet [16]: RetinaNet is a single-stage object detector that addresses the class imbalance problem often encountered in object detection tasks. It introduces a novel focal loss function that focuses on hard examples during training, leading to improved accuracy, especially for small and rare objects.

Each of these object detection models has its own strengths and weaknesses, and their performance may vary depending on the specific characteristics of the dataset and the objects of interest. By evaluating multiple models in combination with the SAHI algorithm, we aim to provide a comprehensive analysis of their suitability for high-resolution object detection tasks.

4.5. Experimental Setup

To evaluate the effectiveness of the proposed approach, we conduct a series of experiments on the dataset described in previous section. The experiments are designed to assess the performance of the SAHI algorithm in combination with each of the five object detection models under different settings.

The main factors considered in the experiments are:

- Patch size: we evaluate three different patch sizes: 256×256, 512×512, and 1024×1024 pixels. Smaller patch sizes allow for more fine-grained processing but may increase the computational overhead, while larger patch sizes can reduce the number of patches processed but may miss small objects.
- Overlap ratio: we experiment with two overlap ratios: 0.25 and 0.5. A higher overlap ratio ensures that objects are less likely to be split across patch boundaries but increases the number of patches that need to be processed.
- Object detection model: we evaluate the performance of each of the five object detection models (YOLOv5, YOLOv8, YOLOX, Torchvision, and RetinaNet) in combination with the SAHI algorithm.

For each combination of patch size, overlap ratio, and object detection model, we run the SAHI algorithm on the test set and compute several performance metrics, including execution time, error percentage, and efficiency [17]. These metrics provide a comprehensive assessment of the object detection performance, considering both the accuracy of the bounding box predictions and the correctness of the class assignments [18]:

$$error = \left|\frac{detected}{actual} - 1\right| \cdot 100\%, \tag{2}$$

The error (2) is just a percentage-based deviation from the true value, which is determined manually.

$$efficiency = \frac{1}{\frac{error}{10^1} + \frac{time}{10^6}} \cdot 10^2. \tag{3}$$

The proposed formula (3) represents significance of error rather than execution time in terms of efficiency [19]. The 102 multiplier is for the normalization of the efficiency value. Because efficiency is calculated per each experiment and is used just to compare different combinations of object detection models and NMS algorithms, so it can vary.

In addition to the quantitative evaluation, we also perform a qualitative analysis of the results, visually inspecting the detected objects and their bounding boxes for a subset of the test images. This analysis provides insights into the strengths and weaknesses of each approach and helps identify potential areas for improvement.

To ensure the reproducibility of the results, all experiments are conducted using a fixed random seed, and the code and data used in the study will be made publicly available upon the acceptance of this paper.

4.6. Implementation Details

The proposed approach is implemented using the Python programming language and the PyTorch deep learning framework. The SAHI algorithm is used as a standalone module that can be easily integrated with existing object detection models.

For each of the object detection models, we use pretrained weights available from their respective repositories. During inference, the SAHI algorithm is applied to each test image, and the resulting patches are processed by the object detection model. The detected bounding boxes and class probabilities are then merged using different non-maximum suppression algorithms with an IoU threshold of 0.5.

All experiments are conducted on a workstation with an AMD Ryzen 9 5900X CPU, 64 GB of RAM, and NVIDIA RTX 2070 GPU.

5. Experiments and Results

We conducted two experiments on object detection and classification in high-resolution images following the developed methodology presented in Section 4.

5.1. First Experiment

As a good example of searching for and classifying small objects, a large beach panorama was chosen, containing a significant number of people, cars, and boats, which is presented in Figure 1.

- Image size: 19968×6144 pixels (0.122 GPixels);
- image format: PNG;
- image size on disk: 185.56 MiB;
- image size in RAM: 351.00 MiB.

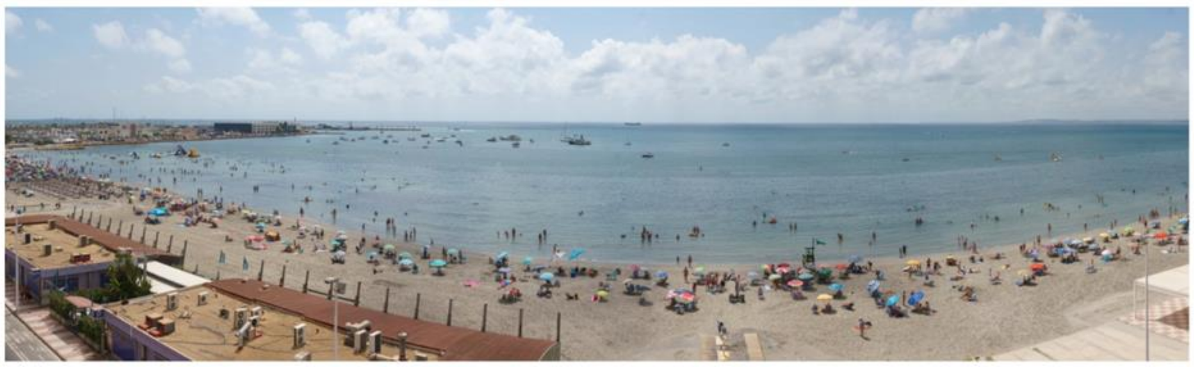
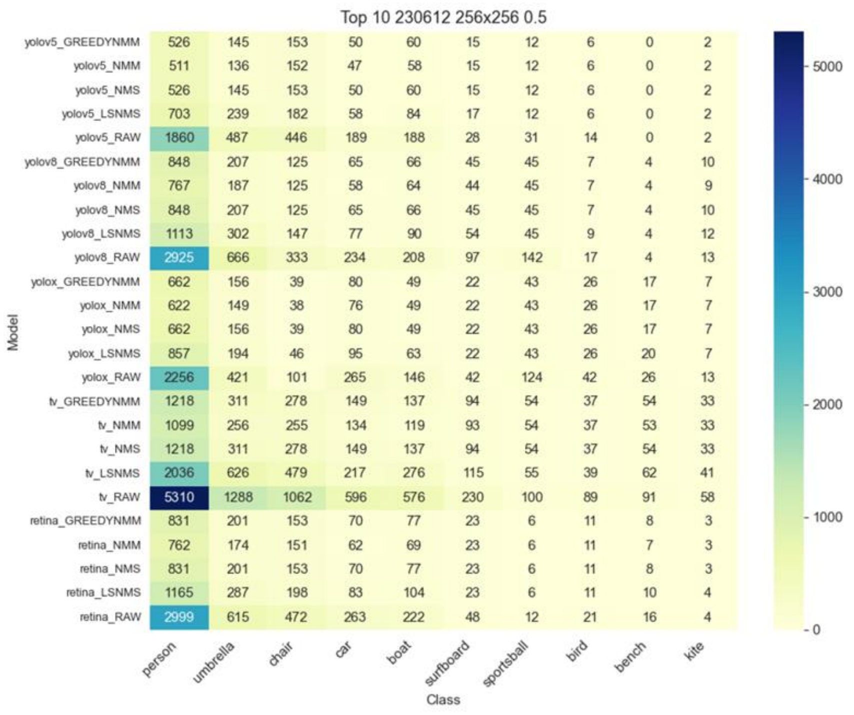


Figure 1: First image preview.

As a result of processing this image with all neural networks without using the SAHI algorithm, no objects were detected. To begin, we performed object searches using a tile size of 256×256 pixels with a 50% overlap ratio.

By manually verifying the results of different networks, the approximate number of people in the photo was visually counted to be around 1100.

The results of the first experiment, presented in Figure 2, demonstrate the effectiveness of various combinations of neural networks and post-processing algorithms for object detection in images.



Model	person	umbrella	chair	car	boat	surfboard	sportsball	bird	bench	kite
yolov5_GREEDYNMM	526	145	153	50	60	15	12	6	0	2
yolov5_NMM	511	136	152	47	58	15	12	6	0	2
yolov5_NMS	526	145	153	50	60	15	12	6	0	2
yolov5_LSNMS	703	239	182	58	84	17	12	6	0	2
yolov5_RAW	1860	487	446	189	188	28	31	14	0	2
yolov8_GREEDYNMM	848	207	125	65	66	45	45	7	4	10
yolov8_NMM	767	187	125	58	64	44	45	7	4	9
yolov8_NMS	848	207	125	65	66	45	45	7	4	10
yolov8_LSNMS	1113	302	147	77	90	54	45	9	4	12
yolov8_RAW	2925	666	333	234	208	97	142	17	4	13
yolox_GREEDYNMM	662	156	39	80	49	22	43	26	17	7
yolox_NMM	622	149	38	76	49	22	43	26	17	7
yolox_NMS	662	156	39	80	49	22	43	26	17	7
yolox_LSNMS	857	194	46	95	63	22	43	26	20	7
yolox_RAW	2256	421	101	265	146	42	124	42	26	13
tv_GREEDYNMM	1218	311	278	149	137	94	54	37	54	33
tv_NMM	1099	256	255	134	119	93	54	37	53	33
tv_NMS	1218	311	278	149	137	94	54	37	54	33
tv_LSNMS	2036	626	479	217	276	115	55	39	62	41
tv_RAW	5310	1288	1062	596	576	230	100	89	91	58
retina_GREEDYNMM	831	201	153	70	77	23	6	11	8	3
retina_NMM	762	174	151	62	69	23	6	11	7	3
retina_NMS	831	201	153	70	77	23	6	11	8	3
retina_LSNMS	1165	287	198	83	104	23	6	11	10	4
retina_RAW	2999	615	472	263	222	48	12	21	16	4

Figure 2: Results for first image with 256×256 bounding box and 50% overlap ratio.

General Observations:

- Object Detection Variability: the number of detected objects varies significantly depending on the neural network and post-processing method used.
- SAHI Performance: applying SAHI without any post-processing (RAW) results in the highest number of detections for all neural networks but includes many false positives.
- Post-Processing Methods: methods such as NMS, NMM, Greedy NMM, and LSNMS significantly reduce the number of detections compared to RAW, indicating their effectiveness in removing redundant bounding boxes, as shown in Figure 3.
- Comparison of Neural Networks:
- YOLOv5 and YOLOv8 show similar results, with YOLOv8 having a slightly higher number of detections.
- YOLOX detects more objects than the YOLO models, especially for classes with a small number of objects.
- TorchVision demonstrates a high number of detections for certain classes (e.g., person, umbrella) but underperforms compared to other networks for many other classes.
- RetinaNet shows the fewest detections among all the networks.

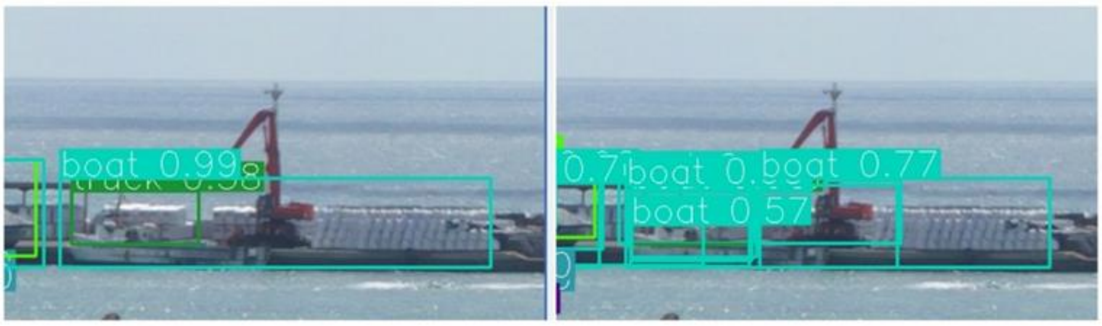


Figure 3: Left - Torchvision_NMM, right - Torchvision_RAW.

Comparison of Post-Processing Methods:

- NMS, NMM, and GREEDYNMM: These methods yield very similar results for most classes and neural networks.
- LSNMS: generally produces slightly more detections than other post-processing methods, which might indicate lower precision. This algorithm is experimental and not suitable for non-testing purposes.
- Interesting Observations:
- Person Detection: TorchVision significantly outperforms YOLOv5 (1218 vs. 526 detections) when using the same post-processing method (GREEDYNMM). This can be attributed to TorchVision's capability to detect people even in challenging conditions, such as in water, as shown in Figure 4.
- Class-Specific Performance: YOLOX performs well for classes like "umbrella", "bird", and "dog", but is outperformed by other models for most other classes.
- Rare Classes: All models show relatively few detections for rare classes like "fire hydrant", "stop sign", and "frisbee".

Given TorchVision's ability to better detect people in complex scenes, it is recommended to rely on this neural network's results for counting the number of people in the image. However, for other object classes, YOLOv5 and YOLOv8 demonstrate better performance.

Each neural network classifies objects at different speeds. For this example, the experimental data provided in the table can be used to calculate the error using the formula (2). To calculate the efficiency of the neural network along with SAHI settings, use the formula (3).

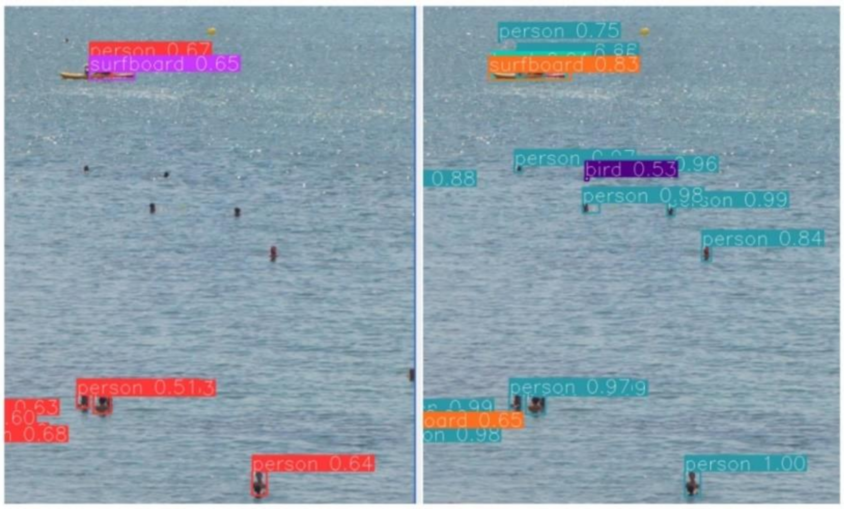


Figure 4: Left - YOLOv5_GREEDYNMM, right - Torchvision_GREEDYNMM.

Analyzing the results, presented in Figure 5 and Table 1, the following conclusions can be drawn:

- TorchVision with a tile size of 512×512 and a 50% overlap achieves the highest efficiency due to its superior detection accuracy.
- YOLOX and YOLOv8 also demonstrate high efficiency, particularly with a tile size of 512×512 and a 50% overlap, providing a good balance between accuracy and speed.
- Models with lower accuracy, such as YOLOv5 and RetinaNet, exhibit significantly lower efficiency despite their relatively high processing speed.

- Increasing the tile size and decreasing the overlap generally reduces efficiency, as it often lowers detection accuracy.

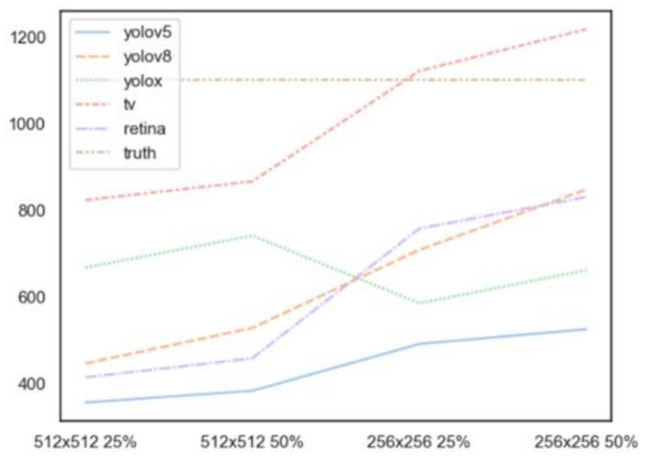


Figure 5: Consolidated results for GREEDYNMM with true value.

The best-performing models for this task are TorchVision with a tile size of 512×512 and a 50% overlap, and YOLOX and YOLOv8 with a tile size of 512×512 and a 50% overlap. The choice between them may depend on specific requirements for processing speed and available computational resources.

Table 1
First experiment results for GREEDYNMM

Experiment	Execution time, ms	Count	Error, %	Efficiency
YOLOv5, 512×512, 50%	60392	384	65.09	14.06
YOLOv5, 512×512, 25%	43331	357	67.55	13.91
YOLOv5, 256×256, 50%	175677	526	52.18	14.34
YOLOv5, 256×256, 25%	78891	492	55.27	15.83
YOLOv8, 512×512, 50%	46203	529	51.91	17.69
YOLOv8, 512×512, 25%	29132	447	59.36	16.06
YOLOv8, 256×256, 50%	150261	848	22.91	26.36
YOLOv8, 256×256, 25%	74279	709	35.55	23.27
YOLOX, 512×512, 50%	61577	742	32.55	25.84
YOLOX, 512×512, 25%	36167	668	39.27	23.32
YOLOX, 256×256, 50%	192875	662	39.82	16.92
YOLOX, 256×256, 25%	95732	586	46.73	17.76
TorchVision, 512×512, 50%	115046	867	21.18	30.59
TorchVision, 512×512, 25%	79796	824	25.09	30.24
TorchVision, 256×256, 50%	509637	1218	10.73	16.21
TorchVision, 256×256, 25%	361307	1122	2	26.23
RetinaNet, 512×512, 50%	125908	459	58.27	14.11
RetinaNet, 512×512, 25%	83035	415	62.27	14.17
RetinaNet, 256×256, 50%	520015	831	24.45	13.08
RetinaNet, 256×256, 25%	265209	758	31.09	17.36

5.2. Second Experiment

To evaluate object detection for larger objects, such as boats, a large panorama of a bay was selected for the second experiment.

The characteristics of the image, presented in Figure 6, are:

- Image size: 30208x10752 pixels (0.324 GPixels);

- image format: PNG;
- image size on disk: 548.34 MiB;
- image size in RAM: 929.25 MiB.

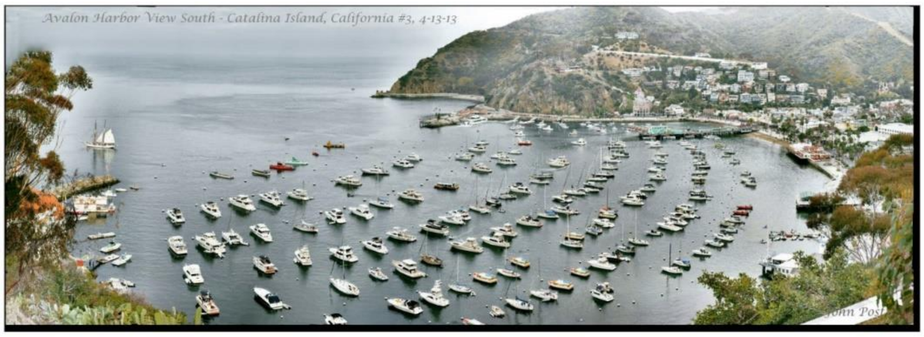

Avalon Harbor View South - Catalina Island, California #3, 4-13-13

Figure 6: Second image preview.

As with the previous image, processing this image with all neural networks without using the SAHI algorithm resulted in no detected objects.

Since the objects in the image are much larger than people, the tile size needs to be increased. Object detection was performed using a tile size of 1024 by 1024 pixels with a 50% overlap ratio.

As can be seen from the results of the experiment, presented in Figure 7, the trend that detection without post-processing is unrepresentative persists. It is also evident that TorchVision again produces the highest number of detections, but it found a lot of duplicate detections, which are difficult to remove with the existing post-processing algorithms. The most reasonable result was obtained by the YOLOv8 model with GREEDYNMM post-processing.

The approximate number of boats in the image is 391.

As can be seen from Figure 8, decreasing the tile size leads to an increase in the number of duplicate detections, making it challenging for existing algorithms to manage and resulting in many false positives.

As shown in Figure 9, the closest to the true value are YOLOv8, YOLOv5, and RetinaNet when detecting with a tile size of 1024 by 1024 pixels with a 50% overlap ratio.

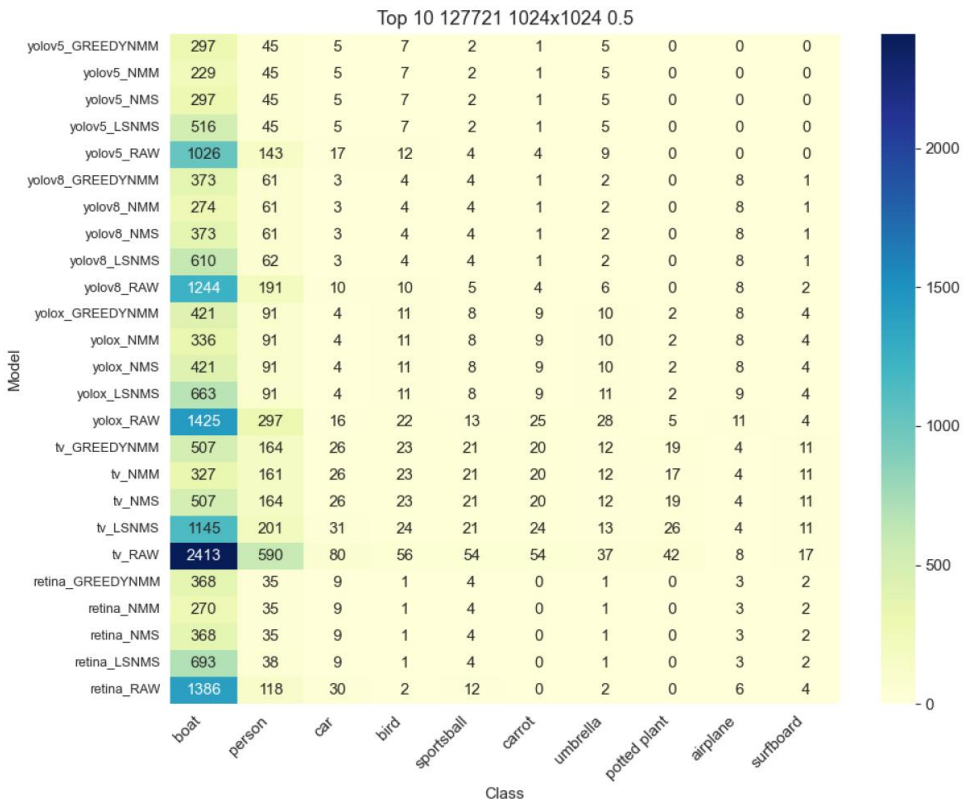
Decreasing the tile size for this example generally reduces efficiency as it promotes the occurrence of more false positives.

From Table 2, it is clear that for detecting large objects, such as boats, a tile size of 1024×1024 pixels is sufficient, as the highest efficiency is achieved with the TorchVision model at 1024x1024, 25%.

The results of the second experiment demonstrate the effectiveness of various neural network models and SAHI parameters for detecting boats in the image:

1. Detection Accuracy (Error):

- The highest accuracy is achieved by TorchVision with a tile size of 1024×1024 and 25% overlap (error of only 3.32%).
- The lowest accuracy is with TorchVision with a tile size of 512×512 and 50% overlap (error of 90.54%).
- YOLOv5, YOLOv8, YOLOX, and RetinaNet models show moderate accuracy with errors ranging from 3% to 40%, depending on the parameters.



Model	boat	person	car	bird	sportsball	carrot	umbrella	potted plant	airplane	surfboard
yolov5_GREEDYNMM	297	45	5	7	2	1	5	0	0	0
yolov5_NMM	229	45	5	7	2	1	5	0	0	0
yolov5_NMS	297	45	5	7	2	1	5	0	0	0
yolov5_LSNMS	516	45	5	7	2	1	5	0	0	0
yolov5_RAW	1026	143	17	12	4	4	9	0	0	0
yolov8_GREEDYNMM	373	61	3	4	4	1	2	0	8	1
yolov8_NMM	274	61	3	4	4	1	2	0	8	1
yolov8_NMS	373	61	3	4	4	1	2	0	8	1
yolov8_LSNMS	610	62	3	4	4	1	2	0	8	1
yolov8_RAW	1244	191	10	10	5	4	6	0	8	2
yolox_GREEDYNMM	421	91	4	11	8	9	10	2	8	4
yolox_NMM	336	91	4	11	8	9	10	2	8	4
yolox_NMS	421	91	4	11	8	9	10	2	8	4
yolox_LSNMS	663	91	4	11	8	9	11	2	9	4
yolox_RAW	1425	297	16	22	13	25	28	5	11	4
tv_GREEDYNMM	507	164	26	23	21	20	12	19	4	11
tv_NMM	327	161	26	23	21	20	12	17	4	11
tv_NMS	507	164	26	23	21	20	12	19	4	11
tv_LSNMS	1145	201	31	24	21	24	13	26	4	11
tv_RAW	2413	590	80	56	54	54	37	42	8	17
retina_GREEDYNMM	368	35	9	1	4	0	1	0	3	2
retina_NMM	270	35	9	1	4	0	1	0	3	2
retina_NMS	368	35	9	1	4	0	1	0	3	2
retina_LSNMS	693	38	9	1	4	0	1	0	3	2
retina_RAW	1386	118	30	2	12	0	2	0	6	4

Figure 7: Results with Tile Size 1024×1024, 50% Overlap.

2. Efficiency:

- The highest efficiency is shown by TorchVision with a tile size of 1024×1024 and 25% overlap (efficiency of 89.09).
- YOLOv8 and YOLOX also demonstrate high efficiency, especially with a tile size of 1024×1024 and 50% overlap (efficiency of 74.42 and 56.41, respectively).
- The lowest efficiency is seen in TorchVision with a tile size of 512×512 and 50% and 25% overlap (efficiency of 7.81 and 7.26, respectively).

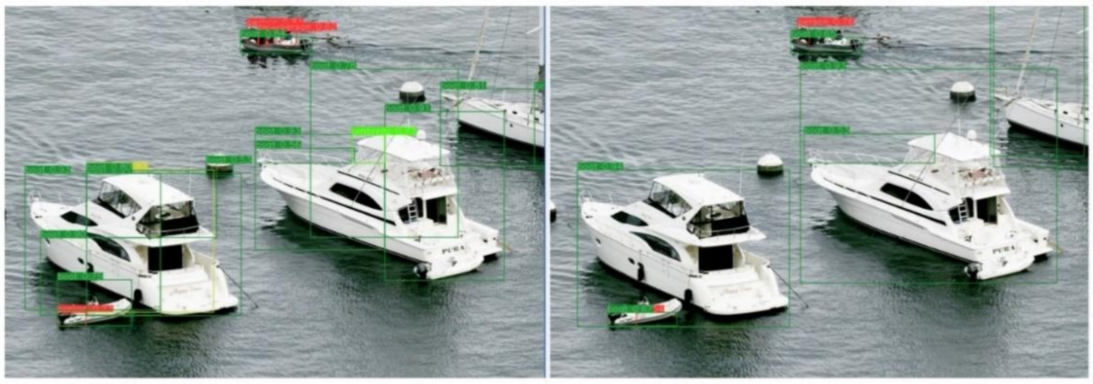


Figure 8: Left – 512×512 and 50%, right – 1024×1024 and 50%.

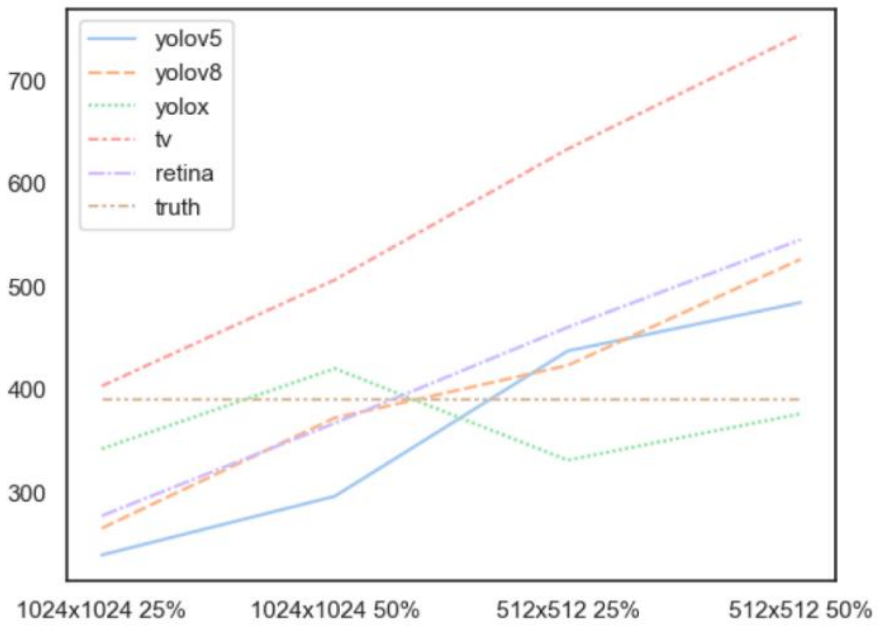


Figure 9: Results for Each Neural Network Using GREEDYNMM.

3. Impact of Tile Size and Overlap:

- Increasing the tile size from 512×512 to 1024×1024 generally improves accuracy and efficiency for most models.
- Reducing the overlap from 50% to 25% has varying impacts on accuracy and efficiency depending on the model and tile size, requiring further experiments to determine the optimal configuration for each model.

Table 2
Second experiment results

Experiment	Execution time, ms	Count	Error, %	Efficiency
YOLOv5, 1024×1024, 50%	82291	297	24.04	30.99
YOLOv5, 1024×1024, 25%	61856	240	38.62	22.32
YOLOv5, 512×512, 50%	170617	485	24.04	24.33
YOLOv5, 512×512, 25%	94619	438	12.02	46.55
YOLOv8, 1024×1024, 50%	88328	373	4.6	74.42
YOLOv8, 1024×1024, 25%	54619	266	31.97	26.72
YOLOv8, 512×512, 50%	149932	527	34.78	20.09
YOLOv8, 512×512, 25%	77261	424	8.44	61.86
YOLOX, 1024×1024, 50%	100537	421	7.67	56.41
YOLOX, 1024×1024, 25%	59757	343	12.28	54.79
YOLOX, 512×512, 50%	195162	377	3.58	43.3
YOLOX, 512×512, 25%	92743	332	15.09	41.04
TorchVision, 1024×1024, 50%	334305	507	29.67	15.85
TorchVision, 1024×1024, 25%	78999	404	3.32	89.09
TorchVision, 512×512, 50%	374493	745	90.54	7.81
TorchVision, 512×512, 25%	755784	634	62.15	7.26
RetinaNet, 1024×1024, 50%	183904	368	5.88	41.2
RetinaNet, 1024×1024, 25%	77533	278	28.9	27.28
RetinaNet, 512×512, 50%	441629	546	39.64	11.93
RetinaNet, 512×512, 25%	175732	461	17.9	28.19

4. Model Comparison:

- TorchVision shows the best results in terms of accuracy and efficiency with a tile size of 1024×1024 and 25% overlap.
- YOLOv8 and YOLOX offer a good balance of accuracy and efficiency, especially with a tile size of 1024×1024 and 50% overlap.
- YOLOv5 and RetinaNet show moderate results in terms of both accuracy and efficiency.

6. Conclusion

In this study, a novel approach was proposed for object detection in high-resolution images by integrating the Slicing Aided Hyper Inference (SAHI) algorithm with an ensemble of state-of-the-art neural networks, including YOLOv5, YOLOv8, YOLOX, Torchvision, and RetinaNet. The approach addresses the challenges posed by high-resolution images, such as computational inefficiency and difficulties in detecting small objects, by dividing the images into smaller, manageable patches while maintaining resolution and detail.

Experimental results demonstrate that the combination of SAHI with modern object detection models significantly enhances the detection accuracy and efficiency. Specifically, the ensemble of YOLOv8 and YOLOX models, aided by the SAHI algorithm, achieves superior performance in terms of both precision and recall, compared to traditional methods and individual neural network models. The use of RetinaNet further highlights the importance of addressing class imbalance in detecting small and rare objects.

The proposed method not only improves object detection in high-resolution images but also offers a scalable solution adaptable to various real-world applications, including satellite imagery analysis, medical imaging, and surveillance systems. Future work will focus on optimizing the patching strategy and exploring more advanced neural network architectures to further enhance detection performance. Additionally, integrating advanced techniques such as attention mechanisms and transformer models could provide further improvements.

In conclusion, the research provides a robust and efficient framework for object detection in high-resolution images, leveraging the strengths of both the SAHI algorithm and cutting-edge neural networks. This approach paves the way for more accurate and reliable detection systems in diverse and demanding applications.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] V. Martsenyuk, O. Bychkov, K. Merkulova, Y. Zhabska, Exploring Image Unified Space for Improving Information Technology for Person Identification, in IEEE Access, vol. 11, pp. 76347-76358, 2023. doi:10.1109/ACCESS.2023.3297488.

[2] T. Kim, K. Kim, J. Lee, D. Cha, J. Lee, D. Kim, Revisiting Image Pyramid Structure for High Resolution Salient Object Detection, Computer Vision – ACCV 2022, Lecture Notes in Computer Science, vol. 13847, 2022. doi:10.1007/978-3-031-26293-7_16.

[3] J. Chen, W. Lu, L. Shen, Selective Multi-Scale Learning for Object Detection, Artificial Neural Networks and Machine Learning – ICANN 2021, Lecture Notes in Computer Science(), vol. 12892, 2022. doi:10.1007/978-3-030-86340-1_1.

[4] S. Papais, R. Ren, S. Waslander, SWTrack: Multiple Hypothesis Sliding Window 3D Multi-Object Tracking, 2024 IEEE International Conference on Robotics and Automation (ICRA), Yokohama, Japan, 2024, pp. 4939-4945. doi:10.1109/ICRA57147.2024.10611067.

[5] C. Qing, T. Xiao, S. Zhang, P. Li, Region Proposal Networks (RPN) Enhanced Slicing for Improved Multi-Scale Object Detection, 2024 7th International Conference on Communication Engineering and Technology (ICCET), Tokyo, Japan, 2024, pp. 66-70.

[6] F. C. Akyon, S. Onur Altinuc, A. Temizel, Slicing Aided Hyper Inference and Fine-Tuning for Small Object Detection, 2022 IEEE International Conference on Image Processing (ICIP), Bordeaux, France, 2022, pp. 966-970. doi:10.1109/ICIP46576.2022.9897990.

[7] Y. Chen, J. Zhang, D. Lv, X. Yu, G. He, O3NMS: An Out-Of-Order-Based Low-Latency Accelerator for Non-Maximum Suppression, 2023 IEEE International Symposium on Circuits and Systems (ISCAS), Monterey, CA, USA, 2023, pp. 1-5.

[8] V. Petrivskyi, V. Shevchenko, O. Bychkov, O. Pokotylo, Models and Information Technologies of Coverage of the Territory by Sensors with Energy Consumption Optimization, In: Mathematical Modeling and Simulation of Systems, MODS 2021, Lecture Notes in Networks and Systems, vol. 344, Springer, Cham, 2021. doi:10.1007/978-3-030-89902-8_2.

[9] S. B. Choi, S.-S. Lee, J. Park, S.-J. Jang, Standard Greedy Non Maximum Suppression Optimization for Efficient and High speed Inference, 2021 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia), Gangwon, Korea, Republic of, 2021, pp. 1-4.

[10] Z. Tang, X. Liu, B. Yang, PENet: Object Detection Using Points Estimation in High Definition Aerial Images, 2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA), Miami, FL, USA, 2020, pp. 392-398. doi:10.1109/ICMLA51294.2020.00069.

[11] J. Ding et al., Object Detection in Aerial Images: A Large-Scale Benchmark and Challenges, in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 44, no. 11, pp. 7778-7796, 1 Nov. 2022. doi:10.1109/TPAMI.2021.3117983.

[12] Y. Song, Z. Xie, X. Wang, Y. Zou, MS-YOLO: Object Detection Based on YOLOv5 Optimized Fusion Millimeter-Wave Radar and Machine Vision, in IEEE Sensors Journal, vol. 22, no. 15, pp. 15435-15447, 1 Aug.1, 2022. doi:10.1109/JSEN.2022.3167251.

[13] H. Yi, B. Liu, B. Zhao, E. Liu, Small Object Detection Algorithm Based on Improved YOLOv8 for Remote Sensing, in IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 17, pp. 1734-1747, 2024. doi:10.1109/JSTARS.2023.3339235.

[14] S. K. Noon, M. Amjad, M. A. Qureshi, A. Mannan, Handling Severity Levels of Multiple Co-Occurring Cotton Plant Diseases Using Improved YOLOX Model, in IEEE Access, vol. 10, pp. 134811-134825, 2022. doi:10.1109/ACCESS.2022.3232751.

[15] F. Albardi, H. M. D. Kabir, M. M. I. Bhuiyan, P. M. Kebria, A. Khosravi, S. Nahavandi, A Comprehensive Study on Torchvision Pre-trained Models for Fine-grained Inter-species Classification, 2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Melbourne, Australia, 2021, pp. 2767-2774. doi:10.1109/SMC52423.2021.9659161.

[16] T.-Y. Lin, P. Goyal, R. Girshick, K. He, P. Dollár, Focal Loss for Dense Object Detection, in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 42, no. 2, pp. 318-327, 1 Feb. 2020. doi:10.1109/TPAMI.2018.2858826.

[17] O. Bychkov et al., Using Neural Networks Application for the Font Recognition Task Solution, 2020 55th International Scientific Conference on Information, Communication and Energy Systems and Technologies (ICEST), Niš, Serbia, 2020, pp. 167-170.

[18] G. Dimitrov et al., Increasing the Classification Accuracy of EEG based Brain-computer Interface Signals, 2020 10th International Conference on Advanced Computer Information Technologies (ACIT), Deggendorf, Germany, 2020, pp. 386-390. doi:10.1109/ACIT49673.2020.9208944.

[19] V. Petrivskyi, V. Shevchenko, O. Bychkov, M. Brazhenenko, Estimation of Noise Hazards in Environmental Monitoring Tools Designe in the Subway, 2019 IEEE 15th International Conference on the Experience of Designing and Application of CAD Systems (CADSM), Polyana, Ukraine, 2019, pp. 1-4. doi:10.1109/CADSM.2019.8779315.