A COMPARATIVE STUDY OF OBJECT DETECTION ALGORITHMS FOR SECURITY APPLICATIONS

Roman Kriuchkov,
[0009-0005-9880-3614]
Timea Bezdan*
[0000-0001-6938-6974]

Singidunum University,
Belgrade, Srebia

Abstract:

Currently, there are many different computer vision models available, and each has its unique characteristics. Selecting the most suitable and, importantly, well-performing model can be challenging for companies and researchers who plan to use artificial intelligence to solve their problems. This study aims to evaluate the performance of four prominent computer vision models: YOLOv5, Faster R-CNN, SSD 300, and RetinaNet. The models were assessed on their ability to detect and classify weapons in images. The primary metrics used for evaluating their performance are mAP@50 and mAP@50-95. The dataset used for testing these models is taken from the well-known dataset platform Kaggle and consists of images of various types of weapons sorted by class. This circumstance also makes it possible to associate this research with the field of security and its automation. Experimental results identified YOLOv5 as the best-performing model among the four. The overall performance was constrained by the dataset's limited size and image quality, with the highest mAP@50 reaching 0.8. The findings of this study offer practical insights for companies seeking effective computer vision solutions, as well as for researchers examining the development and comparative performance of object detection models.

Keywords:

Artificial Intelligence, Computer Vision, Classification, Object Detection, Security Applications.

INTRODUCTION

The field of artificial intelligence, specifically computer vision, has become one of the most active and impactful areas of research in recent years. Computer vision technologies enable the automation and simplification of numerous tasks across various domains[1]. This study focuses on a key capability of this technology, object detection in images, a fundamental task in computer vision. In addition to detecting objects, the models are also evaluated on their ability to classify them into relevant dataset categories. Various types of weapons, including both firearms and edged weapons, served as the objects to be identified.

Correspondence:
Timea Bezdan

e-mail:
tbezdan@singidunum.ac.rs

DOI: 10.15308/Sinteza-2025-524-530

The ability to promptly recognise dangerous objects is both critical and highly valuable[2]. There are numerous environments where enhanced security is essential, including airports, schools, hospitals, and ports[3]. In most of these locations, security is managed by humans, which inherently introduces risks associated with human factors such as fatigue, distraction, or even negligence[4]. Computer vision-based tools in this area mainly aim to enhance security, reduce the likelihood of human errors, and improve monitoring capabilities for video and image data obtained from surveillance cameras or other information transmitters[5].

Successfully performing all the tasks outlined above requires artificial intelligence systems to be trained on a properly constructed dataset and based on a high-quality model capable of effective training and demonstrating high performance in accomplishing the given tasks. Consequently, conducting a comparative analysis of state-of-the-art models is essential to determine the most suitable candidates for such applications. Despite the availability of numerous object detection models, their comparative performance in real-world security contexts, especially with limited-quality datasets, remains underexplored. This raises the key research question: Which object detection model performs best in identifying and classifying weapons in images under constrained data conditions? This research compares four popular and widely used computer vision models, YOLOv5 [6], R-CNN [7], SSD 300 [8], and RetinaNet [9], to identify the most efficient one, defined as the model demonstrating the highest accuracy in object detection tasks. The models' performance is evaluated based on two key metrics mAP@50 [10] and mAP@50-95 [11]. Additionally, the study outlines the dataset preparation process for model training, emphasizing its critical role in the successful application of artificial intelligence in computer vision.

This work contributes by providing a practical comparison of these models on a real-world dataset involving weapon detection, offering insights into their suitability for security applications under constrained data conditions.

The rest of the paper is organised as follows: Section 2 describes the methodology. Section 3 presents the experimental setup, the dataset used, and the results along with their analysis. Section 4 concludes the paper by summarising the key findings.

2. METHODOLOGY

The computer vision models used in this research have different architectures. For instance, YOLOv5, SSD 300, and RetinaNet are single-stage detectors, whereas Faster R-CNN is a two-stage detector. The primary difference between these two approaches lies in the number of processing stages required for detecting and classifying objects within an image: a single-stage detector needs just one pass, while a two-stage detector requires two. During the first stage, region proposal networks (RPN) are generated, and region of interest (ROI) pooling is performed to extract features for each candidate region. The second stage in this architecture is responsible for classifying detected candidates and refining their bounding boxes. As a result, the two-stage detector requires more processing time per image compared to single-stage models, which can be a critical consideration for real-time or resource-constrained applications.

Single-stage object detectors generally follow a modular architecture comprising three main components: (i) the backbone, (ii) the neck, and (iii) the head. While they share this overall structure, the specific implementations can vary significantly across models at both high and low abstraction levels. For models such as YOLOv5, SSD 300, and RetinaNet, the backbone is the main convolutional neural network responsible for feature extraction from the original image. As a result, the backbone identifies both high-level and low-level features. High-level features include shapes and objects, while low-level features correspond to edges and textures [12].

To combine low-level and high-level features, a component known as the neck is used. The neck is an additional set of layers, which acts as a bridge between the backbone and the head, aggregating and refining feature maps from different levels to enhance detection performance. This component creates multi-scale image representations and is implemented in YOLOv5 and RetinaNet. However, YOLOv5 uses two algorithms for its neck, Spatial Pyramid Pooling – Fast (SPPF) [13] and Path Aggregation Network (PANet) [14], whereas RetinaNet employs the Feature Pyramid Network (FPN) [15]. The SSD 300 model does not have a neck component as such, instead, it adds extra convolutional layers after the backbone. These additional layers create multi-scale feature maps, enabling the detection of objects of varying sizes.

The final component in these models is the head, which generates predictions based on the feature maps received from the neck. It is responsible for producing the model's outputs, including object classifications and

bounding box coordinates. Although the implementation of the head varies across models, they all incorporate a multi-scale prediction mechanism, which allows the model to make predictions at different feature levels. This approach enhances the detection of objects of varying sizes by leveraging both fine and coarse spatial information.

Another distinguishing feature among the four models is the resolution of the input images they process. YOLOv5 uses a default resolution of 640×640, while SSD 300 operates on 300×300 pixel inputs. RetinaNet and Faster R-CNN use variable input sizes, typically with the shorter side resized to 800 pixels and the longer side not exceeding 1333 pixels. Further details regarding each model's configuration are provided in Section 3.1, Experimental Setup.

As mentioned earlier, the metrics mAP@50 (Mean Average Precision at 50% IoU) and mAP@50–95 were used to assess the performance of each model. These metrics evaluate how accurately a model detects and localises objects based on the overlap between predicted and ground-truth bounding boxes. mAP@50 measures the average precision when the Intersection over Union (IoU) threshold is fixed at 0.50. In contrast, mAP@50–95 is a more comprehensive metric that calculates the mean of average precision scores across ten IoU thresholds, ranging from 0.50 to 0.95 in increments of 0.05, thereby offering a more rigorous assessment of model performance.

Additionally, precision and recall metrics are reported for each model in this study. Precision indicates the proportion of correctly detected objects among all detected instances, while recall measures the proportion of correctly detected objects relative to the total number of ground-truth objects in the dataset.

3. EXPERIMENTAL RESULTS AND DISCUSSION

The experiment conducted in this study aimed to evaluate and compare the performance of each model on the available dataset. All experiments were implemented in PyTorch and executed using an NVIDIA T4 GPU.

3.1. DATASET PREPARATION

The dataset used in this research, titled "Weapon Detection Dataset", was sourced from Kaggle and contains 714 images of weapons belonging to nine distinct classes: Automatic Rifle, Bazooka, Handgun, Knife, Grenade Launcher, Shotgun, SMG, Sniper, and Sword. In addition to the images, the dataset includes a description file with metadata and YOLO-format annotations stored as plain text files. The dataset was initially divided into training and validation sets in an 80/20 ratio.

During dataset preparation, extensive work was performed to identify and remove duplicate images and to prevent data leakage from the training set to the validation set. Additionally, many classes in the annotations were modified, as they did not match the classes described in the accompanying file and incorrectly represented the objects shown in the images. The final step in dataset preparation involved creating COCO annotations based on the corrected YOLO annotations. This step was necessary to train the SSD 300, RetinaNet, and Faster R-CNN models since they cannot work directly with YOLO annotations.

Figure 1 shows one example of the images taken from the dataset, displaying bounding boxes and class IDs after completing all the data preprocessing steps

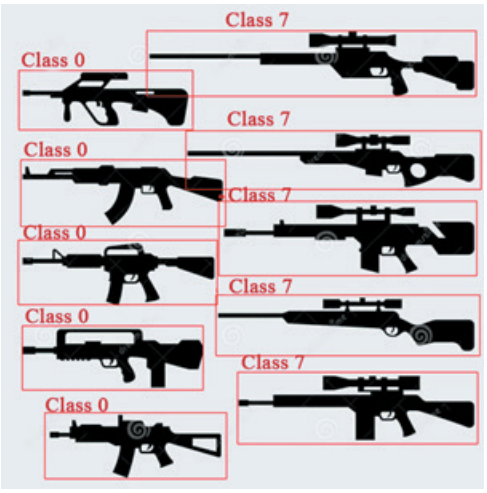

Figure 1. Sample image with bounding boxes from the dataset

described above. As a result, 500 images remained in the dataset, with 88% allocated for training and 12% for validation.

3.2. EXPERIMENTAL SETUP

Each of the four models was trained for 75 epochs. The batch size and number of data loading workers were kept consistent across all models, while the learning rate and input image resolution varied depending on the model architecture and its requirements. Table 1 summarises the key training parameters, including the image sizes that produced the best mAP@50 and mAP@50–95 results for each model.

It is also worth noting the significant difference in configuring SSD 300, Faster R-CNN, and RetinaNet compared to YOLOv5. The latter offers a much simpler and more compact configuration process.

3.3. EXPERIMENTAL RESULTS AND DISCUSSION

This subsection describes the results of experiments conducted using all four computer vision models on a dataset containing images of various types of weapons. Table 2 presents the comparative analysis.

Based on the comparative analysis results, the model that demonstrated the best performance was YOLOv5. The mAP@50 metric was equal to 0.8, and mAP@50-95 reached 0.57, indicating strong object localisation and classification performance. Precision of 75% and recall of 77% suggest that the model occasionally makes mistakes in identifying the desired objects in images. However, these metrics are very close to each other, indicating that YOLOv5 has indeed learned to reliably detect target objects. Given that recall is at 77%, the quality of predictions remains high, meaning the number of false positives is relatively low. This is further supported by the small difference between precision and recall, only 2%. It is worth noting that the model ultimately manages to detect certain types of weapons in images quite successfully, thus fulfilling its intended purpose. Additionally, the model could be further fine-tuned using additional data to improve its results.

Figure 2 illustrates the confidence points in the YOLOv5 model's predictions, as well as boundary boxes. As can be seen in the image, the model can quite reliably detect classes such as SMG, sword, and handgun, with confidence scores ranging from 0.8 to 0.9. However, YOLOv5 still struggles to accurately identify classes like shotgun and knife, with confidence points ranging from 0.6 to 0.7.

The second-best performing model was Faster R-CNN. This model demonstrated results close to YOLOv5 with metrics mAP@50 and mAP@50-95 equal to 0.75 and 0.53, respectively, which are satisfactory outcomes. However, the model's precision (53%) and recall (60%) were considerably lower than those of YOLOv5. Due to these low precision and recall scores, the model is likely to perform poorly in detecting weapons within images, and it may be less reliable in real-world weapon detection tasks.

Table 1. Parameter configuration

Model	Batch size	Learning rate	Workers	Image size (pixels)
YOLOv5	4	1e-2	2	640 (long side)
SSD 300	4	1e-4	2	300x300
RetinaNet	4	1e-4	2	512x512
Faster R-CNN	4	1e-4	2	800 (short side), max 1333 (long side), aspect ratio preserved

Table 2. Comparative analysis

Model	mAP@50	mAP@50-95	Precision	Recall
YOLOv5	0.80	0.57	75%	77%
SSD 300	0.57	0.36	36%	52%
RetinaNet	0.38	0.20	20%	48%
Faster R-CNN	0.75	0.53	53%	60%

Figure 2. YOLOv5 predictions for validation dataset

Table 3. Per-class evaluation metrics for YOLOv5

Class	mAP@50	mAP@50-95
Automatic Rifle	0.65	0.53
Bazooka	0.70	0.37
Handgun	0.90	0.73
Shotgun	0.90	0.66

SSD 300 ranks third, with a mAP@50 of 0.57 and mAP@50-95 of 0.36. Such low metrics indicate that the model struggles to accurately identify object boundaries. In addition to this issue, SSD 300 demonstrated low accuracy in correct predictions: specifically, a recall of 52% combined with a high number of false positives, as shown by a low precision of 36%. Likely, due to generating many predictions, including incorrect ones, the model captures some true positives, thereby increasing recall. However, this also means that the model did not effectively learn to recognise the desired objects in images.

RetinaNet showed the worst results. Its mAP@50 and mAP@50-95 metrics were just 0.38 and 0.20, respectively. This clearly indicates that the model does not effectively detect correct object boundaries. Precision was only 20% yet recall reached an unexpectedly high 48%. Based on these figures, many false positives were observed, reflecting low model selectivity. RetinaNet essentially failed to learn how to properly detect various types of weapons in images.

The results described above can be attributed, in part, to limitations in the dataset. For YOLOv5 and Faster R-CNN, dataset limitations were the main obstacle to achieving more accurate predictions. The dataset has several systemic issues that collectively prevented these models from performing better. Specifically, the dataset contains 9 classes of various weapons but only 439 training images, which is insufficient to teach the model to correctly detect each class. Indirect confirmation of this issue is shown in Table 3 below.

As seen from the table, mAP@50 and mAP@50-95 scores differ significantly across weapon types, indicating that the model particularly struggled to identify automatic rifles and bazookas in images. At the same time, this issue was not due to an insufficient number of epochs. YOLOv5 and Faster R-CNN extracted the maximum amount of information available from the dataset used. This conclusion can be drawn from the graphs illustrating the growth of mAP@50 and mAP@50-95 over epochs.

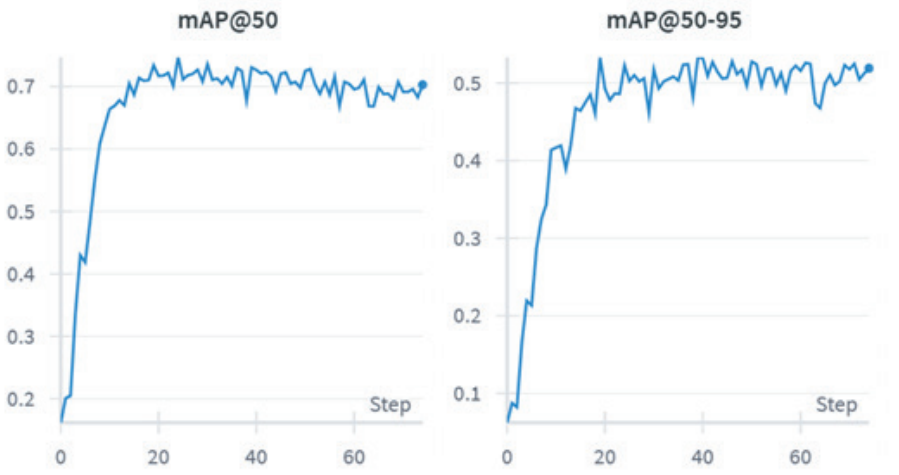

Figure 3. The evolution of accuracy (mAP) metrics on the validation set for Faster R-CNN

As evident from the graphs above, both metrics plateaued around the 20th epoch, indicating that further increases in epoch count would be pointless. Expanding the original dataset would yield a greater improvement.

Regarding SSD 300, the most probable reason for its poor results was the limitation of input image size. The dataset contains images of various dimensions, and resizing larger images down to 300x300 inevitably results in losing some low-level features. This leads to degradation in the results for mAP@50 and mAP@50-95 metrics for this model.

For RetinaNet, which demonstrated the lowest performance, various input resolutions were tested, but the most effective was found to be 512x512. Despite this, RetinaNet exhibited a similar plateau on the accuracy graph as shown in Figure 3, likely indicating that RetinaNet's internal algorithms are less effective compared to the other tested models [16].

4. CONCLUSION

This research addressed the core research question by comparatively evaluating four object detection models, YOLOv5, SSD 300, RetinaNet, and Faster R-CNN, on a security-relevant dataset. Despite certain systematic issues with the dataset, two models achieved good results according to the metrics mAP@50 and mAP@50–95. YOLOv5 demonstrated the best performance across all parameters. The mAP@50 for this model reached 0.80, which is a respectable result, though it could be further improved by expanding and refining the dataset. This work contributes to a clearer understanding of the strengths and limitations of these popular object detection models in the context of weapon detection, potentially assisting researchers and industry practitioners in selecting and adapting models for real-time security applications. Additionally, the study highlighted potential shortcomings in SSD 300 and RetinaNet, offering insights for future improvements.

Future research should focus on evaluating these models on more diverse and larger datasets, exploring fine-tuning strategies, and testing in real-world surveillance scenarios to assess their robustness and generalisation capabilities.

REFERENCES

[1] S. Akcay, M. E. Kundegorski, C. G. Willcocks, and T. P. Breckon, "Using Deep Convolutional Neural Network Architectures for Object Classification and Detection within X-ray Baggage Security Imagery", Accessed: Mar. 30, 2025. [Online]. Available: https://durham-repository.worktribe.com/output/1338347/

[2] A. Egiazarov, F. M. Zennaro, and V. Mavroeidis, "Firearm Detection via Convolutional Neural Networks: Comparing a Semantic Segmentation Model Against End-to-End Solutions," Dec. 17, 2020, *arXiv*: arXiv:2012.09662. doi: 10.48550/arXiv.2012.09662.

[3] G. Batsis, I. Mademlis, and G. T. Papadopoulos, "Illicit item detection in X-ray images for security applications," May 03, 2023, *arXiv*: arXiv:2305.01936. doi: 10.48550/arXiv.2305.01936.

[4] K. J. Liang et al., "Toward Automatic Threat Recognition for Airport X-ray Baggage Screening with Deep Convolutional Object Detection," Dec. 13, 2019, *arXiv*: arXiv:1912.06329. doi: 10.48550/arXiv.1912.06329.

[5] S. Yellapragada et al., "CCTV-Gun: Benchmarking Handgun Detection in CCTV Images," Jul. 11, 2023, *arXiv*: arXiv:2303.10703. doi: 10.48550/arXiv.2303.10703.

[6] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *Proc. CVPR*, May 2016, pp. 779–788. doi: 10.48550/arXiv.1506.02640.

[7] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," Jan. 06, 2016, *arXiv*: arXiv:1506.01497. doi: 10.48550/arXiv.1506.01497.

[8] W. Liu et al., "SSD: Single Shot MultiBox Detector," vol. 9905, 2016, pp. 21–37. doi: 10.1007/978-3-319-46448-0_2.

[9] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal Loss for Dense Object Detection," Feb. 07, 2018, *arXiv*: arXiv:1708.02002. doi: 10.48550/arXiv.1708.02002.

[10] "(PDF) The Pascal Visual Object Classes (VOC) challenge." Accessed: Mar. 27, 2025. [Online]. Available: https://www.researchgate.net/publication/220659463_The_Pascal_Visual_Object_Classes_VOC_challenge

[11] T.-Y. Lin et al., "Microsoft COCO: Common Objects in Context," Feb. 21, 2015, *arXiv*: arXiv:1405.0312. doi: 10.48550/arXiv.1405.0312.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 770–778. doi: 10.1109/CVPR.2016.90.

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition," vol. 8691, 2014, pp. 346–361. doi: 10.1007/978-3-319-10578-9_23.

[14] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path Aggregation Network for Instance Segmentation," Sep. 18, 2018, *arXiv*: arXiv:1803.01534. doi: 10.48550/arXiv.1803.01534.

[15] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature Pyramid Networks for Object Detection," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017, pp. 936–944. doi: 10.1109/CVPR.2017.106.

[16] Z.-Q. Zhao, P. Zheng, S. Xu, and X. Wu, "Object Detection with Deep Learning: A Review," Apr. 16, 2019, *arXiv*: arXiv:1807.05511. doi: 10.48550/arXiv.1807.05511.