DOI: https://doi.org/10.15276/aait.08.2025.25
UDC 52-004.8

Research on the impact of optimization algorithms on the accuracy of YOLOv11 neural networks

Serhii M. Kovbasa[1)]
ORCID: https://orcid.org/0000-0002-2954-455X; skovbasa@ukr.net. Scopus Author ID: 55328200100
Anton O. Holosha[1)]
ORCID: https://orcid.org/0009-0003-4858-202X; kholosha.anton@gmail.com
[1)] National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", 37, Beresteiskyi Ave. Kyiv, 03056, Ukraine

ABSTRACT

Visual inspection and positioning based on image detection results is a rapidly growing component of automation systems. Machine vision is increasingly used in production lines for various technological purposes, as well as in special equipment. Improving recognition accuracy in such applications can be a difficult task, especially in conditions of possible limitations, one of which may be size and weight restrictions, which in turn limit the power of computer devices that implement image detection and recognition. A possible solution to this problem is to improve recognition accuracy by automatically tuning the hyperparameters of detection models using various optimization methods. This article presents the results of a study of the effectiveness of algorithms for automatically tuning the hyperparameters of the YOLO (You Only Look Once) image detection model, built on the basis of the SGD (stochastic gradient descent), Adam (adaptive learning rate estimation), and AdamW (improved version of Adam) optimization methods. The models were trained on the COCO 2017 dataset, limited to eight classes and balanced in terms of the number of images. For each optimization algorithm, 10 iterations of automatic selection were used, followed by training for 30 epochs. The test results showed that in tasks where detection accuracy is important and there are sufficient computational resources for hyperparameter selection during model training, it is advisable to use the stochastic gradient descent optimization algorithm, since the detection model configured with its use provides a higher probability of successful image recognition in real time under conditions of limited computational resources.

Keywords: Artificial intelligence; computer vision; convolutional neural networks; COCO dataset; hyperparameter optimization; SGD; Adam; AdamW; YOLOv11; mAP; IoU; precision; recall

***For citation*: Kovbasa S. M., Holosha A. O. "Research on the impact of optimization algorithms on the accuracy of YOLOv11 neural networks". *Applied Aspects of Information Technology.* 2025; Vol.8 No.4: 386–396. DOI: https://doi.org/10.15276/aait.08.2025.25**

INTRODUCTION

In recent years, production systems for various technological purposes have been increasingly modernized towards the use of computer vision, namely object detection and classification [1], [2], [3]. Many systems, including unmanned ones, are implemented in a portable version, which causes problems with usable space, which in turn affects the potential computing power for computer vision [4]. Therefore, increasing the accuracy of object recognition without increasing the size of the detection model (DM) or computational resources is an important and urgent task. One solution for improving detection is to optimize the existing model, which is the subject of ongoing research [5], [6], [7]. In [5], research is conducted on the Gray Wolf Optimizer (GWO), Artificial Rabbit Optimizer (ARO), and Chimpanzee Leader Selection Optimizer (CLEO) optimization methods only under poor conditions and without comparison with standard optimization methods. In turn, in [6], optimization was performed without comparing methods.

The work [7] deserves special attention, in which a modification of the YOLO architecture (YOLO-TC) is proposed for detecting small objects, such as cigarettes and mobile phones, in construction sites using tower cranes. The authors integrated the ECA-CBAM spatial-channel attention module into the backbone and expanded the HA-PANet PAN structure in the neck, as well as optimized the MPDIoU loss function. The proposed model demonstrated improved accuracy on a specially formed dataset and high resistance to real production conditions. However, the optimization was achieved by increasing the number of model parameters, which complicates its application on portable devices. A fairly common approach to solving the optimization problem without increasing the size of the model is to tune the DM hyperparameters, which allows for increased performance.

Tuning hyperparameters is a lengthy process of manually or automatically selecting the parameters that control the training of an artificial intelligence model. For object recognition models, such as YOLO (You Only Look Once), the initial hyperparameters available for tuning include batch size, loss functions, image size, and the number of training epochs, which subsequently significantly

affect recognition accuracy [8], [9]. This approach is especially useful in systems that cannot be upgraded due to the high cost of additional equipment.

1. ANALYSIS OF LITERATURE AND STATEMENT OF THE PROBLEM

Convolutional neural networks (CNNs) have become the basis for modern object detection algorithms, including YOLO, due to their ability to automatically learn from input data with labeled features [10], [11]. CNNs work by applying convolutional layers that act as filters to detect local features, and then by combining layers that reduce the dimensionality of the data while preserving the most salient features. This extraction process allows convolutional neural networks to handle complex tasks of generating, classifying, and recognizing objects in images. One of the most famous and, as a result, the most ambitious models for object detection tasks is the aforementioned YOLO neural network. Image recognition neural networks are divided into models that perform recognition in several stages and those that perform detection in a single pass of the image. The YOLO neural network treats object detection as a single regression task. Because YOLO processes images in a single pass, it provides the fastest performance among the neural networks available on the market and, at the same time, is constantly evolving. For example, YOLOv3 introduced multi-scale detection and bounding boxes, allowing the model to detect objects of different sizes in aspect ratio [12]. In the YOLOv11 version, the recognition model was further improved by increasing the amount of data and improving the calculation of the loss function, making it a highly accurate and efficient model for real-time object detection [13].

YOLO has five main pre-trained models that differ in speed, number of parameters, computing power requirements, and output accuracy, and are classified by letters: n – nano, s – small, m – medium, l – large, x – extra large.

The main parameters of each DM type and metrics on the standard COCO 2017 dataset [14] models are presented in Table 1. The number of parameters has a nonlinear effect on the size of the model, so for the most dynamic and lightweight tasks, the smallest model is used, followed by tuning of hyperparameters, the dataset, and loss calculation options. There is also a tendency for accuracy to decrease with the number of parameters, which also plays a significant role in finding options to improve accuracy without increasing the size of the model.

RESEARCH GOALS AND OBJECTIVES

Despite annual progress in YOLO model architecture, improving image recognition quality remains a challenging task. Progress in YOLO model architecture is shown in Fig. 1, which shows minimal growth between recent generations. The need to tune the model before training is still an important area of research. Tuning parameters by default is not always the best option, especially when accurate object detection is required, while manual tuning requires significant time for verification. To ensure the process of automatic hyperparameter tuning in YOLO, three algorithms based on the following optimization methods have been most widely used: SGD (stochastic gradient descent), Adam (adaptive learning rate estimation), and AdamW (an improved version of Adam). The purpose of this work is to study the effectiveness of algorithms for automatic hyperparameter tuning based on the SGD, Adam, and AdamW optimization methods to improve the accuracy of YOLO in conditions of limited computational resources.

To achieve this goal, the following tasks must be performed:

- analyze existing approaches to optimizing YOLO model hyperparameters;
- perform automatic tuning of YOLOv11n hyperparameters using SGD, Adam, and AdamW;
- compare the impact of optimization algorithms on mAP, precision, recall, and loss function metrics;
- analyze the effectiveness of optimizers in the context of limited computing resources;
- formulate recommendations for choosing an optimizer for systems with limited computing power.

2. MAIN RESULTS OF THE RESEARCH

2.1. YOLO structure

In general, the structure of YOLO has not changed since YOLOv3 and consists of the following main blocks. Each block plays its role in the process of image analysis, detection, and classification, all of which takes place in the hidden layers of the data blocks.

Backbone

This block is responsible for extracting certain features and characteristics from the image and for further identification of these features. The process is the lowest level of detection and at the same time is critically important because it converts raw image data into probability maps that capture visual information.

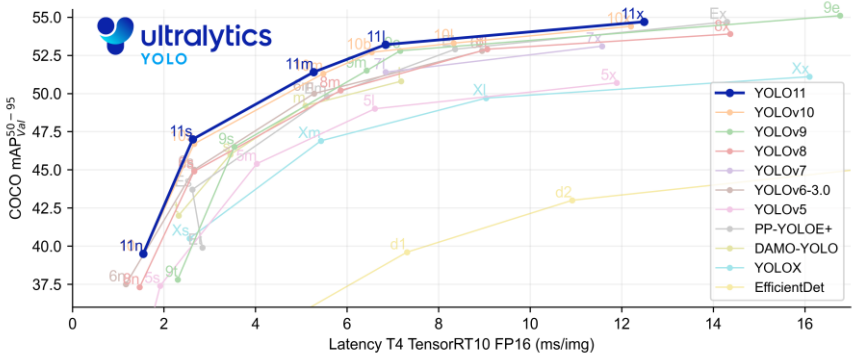

Fig. 1. **Progress of YOLO architectures**
***Source*: created by the [15]**

Table 1. **Comparative characteristics of YOLO models**

Model	size(pixels)	mAP^{val}_{50-95}	Speed CPU ONNX (ms)	Speed T4 TRT10 (ms)	params (M)	FLOPs (B)
YOLOv11n	640	39.5	56.1	1.5	2.6	6.5
YOLOv11s	640	47.0	90.0	2.5	9.4	21.5
YOLOv11m	640	51.5	183.2	4.7	20.1	68.0
YOLOv11l	640	53.4	238.6	6.2	25.3	56.9
YOLOv11x	640	54.7	462.8	11.3	56.9	194.9

***Source*: created by the [15]**

Neck

This block combines the features obtained from the previous block, forms a holistic view of the objects in the image, and passes it on to the Head detection block. The block is designed to improve the ability to recognize small objects by using layers with different resolutions. The improvement is achieved by enhancing and localizing features in small layers.

Head block

This layer is responsible for prediction, constraint generation, class prediction, and object detection. It contains mechanisms that allow the network to make multiple predictions for a given region of interest, improving detection accuracy. In addition, this part includes a simplified prediction strategy that allows for faster conclusions, which is critical for real-time applications. These improvements in the prediction layer ensure efficient and accurate model performance in dynamic and time-sensitive scenarios [16].

Fig. 2 shows a block diagram of YOLOv11, in which the base block has the following internal blocks: Conv (convolution) – a convolution block that extracts basic image features such as edges and texture corners, while using filters to reduce the image while preserving important information, preparing data for further processing by more complex layers; C3K2 – a cascade convolution block that uses multiple convolutions for even deeper feature extraction and combination for better detection. The neck block has the following internal blocks: Upsample – a block that increases the image size to improve object recognition at different levels, a critical block for small object detection; Concats – a block that combines detected features from different levels and scales; SPFF (Spatial Pyramid Pooling Fusion) – uses the concept of spatial pyramid pooling to extract features at different scales. The main block only has Detect blocks that define bounding boxes for each object in the image. It generates a grid of possible bounding boxes and the probability of each block [17].

Of particular interest in the main block are IoU (Intersection over Union) losses, which are an evaluation metric commonly used for object detection. IoU losses, which take into account the difference between the area of the actual and predicted bounding boxes, are defined as

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}, \qquad (1)$$

where Area of Overlap is the intersection area between the predicted bounding box and the actual bounding box, Fig. 3; Area of Union – is the union area, i.e., the total area covered by both bounding boxes.

If IoU = 1, this means a perfect match. Typically, an IoU value ≥ 0.5 is considered a sign of successful object detection [18].

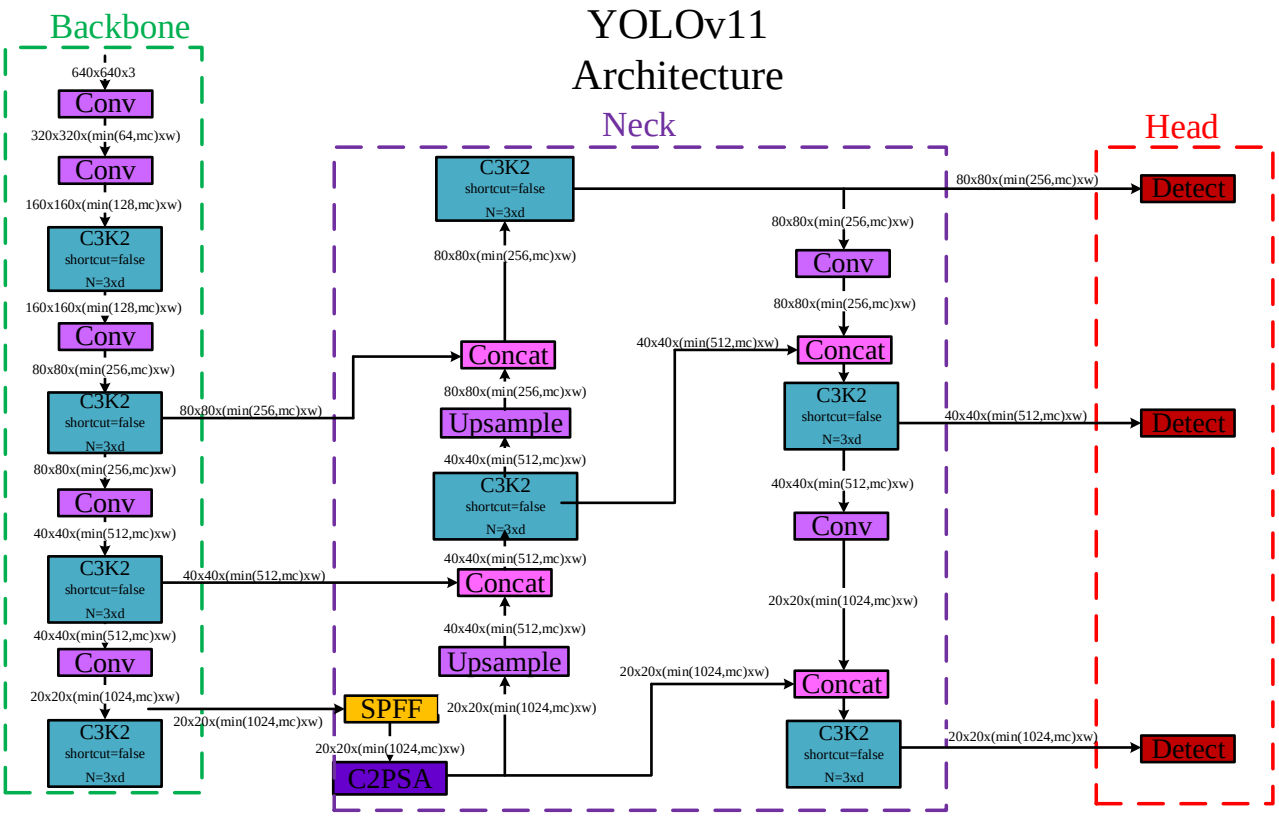

Fig. 2. **Structural diagram of YOLOv11**
***Source*: created by the [17]**

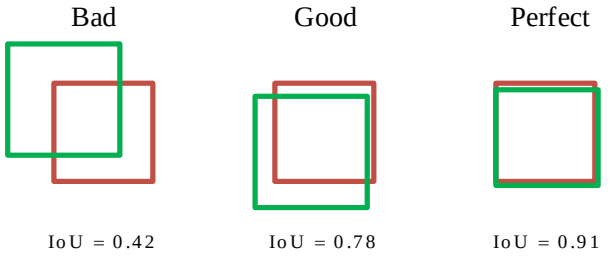

Fig. 3. **IoU intersection area options**
***Source*: created by the authors**

Fig. 4 shows a simplified example of the stages of the image detection model. The first stage of the DM's operation after receiving an image is to divide it into a grid of 80 × 80 cells, which is performed by the backbone. This grid size allows for a balance between accuracy and image processing speed, since each block is processed separately. After dividing into cells and extracting features using the above-mentioned blocks, the data is transferred by the implicitly represented neck block the main block. At this stage, multi-scale feature fusion takes place, thanks to which the model is able to recognize both large and small objects equally well. The neck block improves the detection quality of scaled and differently sized objects. In the head block, each cell predicts the number of bounding boxes, each of which has its own confidence value and probability of belonging to a certain object class. At this stage, IoU metric values are formed, which allow assessing the degree of overlap between the predicted frame and the actual one. Next, the head block transmits the data with probabilities for processing, where maximum suppression is not applied and the final prediction is generated [19], [21].

2.2. Hyperparameters

In general, hyperparameters are a set of DM configuration coefficients that are set by the user in advance and determine the behavior and effectiveness of the DM training process. They cannot be changed during training by the user or automatically. These parameters can only be changed before training begins, and the results for the set of parameters that were entered before training begins become known after the model has been fully trained, which may take some time.

The paper [22] shows that to optimize YOLO performance, several hyperparameters can be adjusted, such as input image size, number of epochs, batch size, learning rate, and initial momentum. In addition, adjusting hyperparameters such as learning rate and initial momentum can reduce training time and improve model accuracy.

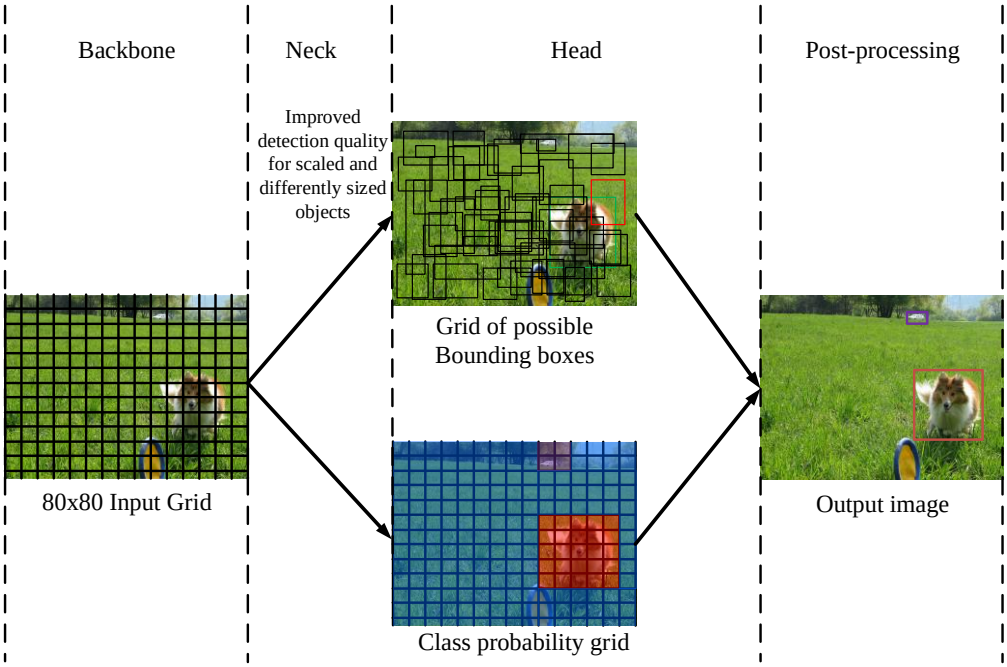

Fig. 4. **Stages of the detection model**
***Source*: created by the authors**

The choice of hyperparameters plays a significant role in the model's ability to learn, and the selection is made individually for each new dataset. From a practical point of view, incorrectly defined hyperparameters can cause a deterioration in detection quality and be accompanied by underfitting and overfitting.

Table 2 below lists the main hyperparameters for detection models.

2.3. Optimization algorithms

When training neural networks, hyperparameters can be selected either manually or using optimization algorithms. In this paper, we will consider the three most common ones: SGD, Adam, and AdamW [20].

Table 2. **Hyperparameters for YOLOv11 models**

Parameter	Type	Range	Description
lr0	float	[1e-5; 1e-1]	Initial learning rate at the beginning of training. Lower values provide more stable learning, but slower convergence.
lrf	float	[0.01; 1.0]	The final learning rate coefficient as a fraction of lr0. Controls how much the learning rate decreases during training.
mom-entum	float	[0.6; 0.98]	SGD momentum factor. Higher values help maintain a constant gradient direction and can accelerate convergence.
weight_ decay	float	[0.0; 0.001]	L2 regularization coefficient to prevent overfitting. Higher values provide stronger regularization.
warmup_ epochs	float	[0.0; 5.0]	Number of epochs for linear warmup of learning rate. Helps prevent instability in the early stages of training.
warmup_ moment	float	[0.0; 0.95]	Initial momentum during the learning warmup phase. Gradually increases to the final momentum value.
warmup_ bias_lr	float	[1e-5; 1e-3]	The learning rate for bias parameters during the warmup phase, which helps stabilize model training in the early epochs.
box	float	[1; 10]	The weight of the bounding box loss in the overall loss function. Balances bounding box regression and classification.
cls	float	[0.2; 4.0]	The weight of classification losses in the total loss function. Higher values emphasize correct class prediction.
dfl	float	[0.5; 2.5]	The weight of distribution focal loss used in some versions of YOLO for accurate classification.

***Source*: created by the [8]**

SGD is an optimization algorithm widely used in machine learning and neural network training. It is the default optimization algorithm for training detection models. SGD works by adjusting the model parameters (θ_i) to reduce the error $L(\theta_i)$ during training, where i is the iteration number. This is achieved by calculating how much the model error changes in response to small changes in its parameters . This is called the gradient, which shows the direction of movement to reduce the error.

SGD updates the model parameters by moving them in the opposite direction to this gradient.

The size of each step is controlled by a value called the learning rate (η), which determines how large or small the adjustments will be:

$$\theta_i = \theta_{i+1} - \eta \cdot \nabla_\theta L(\theta_i), \quad (2)$$

where θ_i is the model parameter; θ_{i+1} is the updated parameter after the learning step; η is the learning rate parameter; $L(\theta_i)$ is the loss function for the current model parameters; $\nabla_\theta L(\theta_i)$ is the gradient of the loss function with respect to the parameters θ.

For each training example, the model performs an update based on the error for that example. Over time, this helps the model reduce its overall error and improve its predictions.

Thus, SGD helps the model learn by gradually adjusting its internal settings based on information about how the current settings affect the overall error [21].

Adam and AdamW optimizers are based on SGD concepts but implement adaptive learning rate adjustment methods. These optimizers are used to regulate network parameters by calculating gradients, similar to SGD. The AdamW method implements parameter regularization by reducing them, which generally leads to faster convergence to the optimal result.

Adam and AdamW are extensions of SGD that include adaptive moment estimation, namely the mean value of previous gradients.

They calculate moving averages of gradients, denoted as m_i, and quadratic gradients, denoted as v_i, using the following expressions:

$$m_i = \beta_1 m_{i-1} + (1-\beta_1) g_i, \quad (3)$$

$$v_i = \beta_2 v_{i-1} + (1-\beta_2) g_i^2, \quad (4)$$

where the coefficient β_1 controls the exponential decay rate for the first moment estimate, the coefficient β_2 controls the decay rate for the second moment estimate, and g_i is the gradient computed on the current batch of data. These coefficients are used to update m_i and v_i during the learning process, which ultimately leads to improved convergence in optimization [21].

For the final parameter update formula, it is necessary to introduce moment offset corrections, which are calculated using the following formulas:

$$\hat{m}_i = \frac{m_i}{1-\beta_1^i}; \quad (5)$$

$$\hat{v}_i = \frac{v_i}{1-\beta_2^i}. \quad (6)$$

Parameter updates using adjusted moments and gradients are performed according to the expression:

$$\theta_i = \theta_{i-1} - \frac{\eta}{\sqrt{\hat{v}_i} + \varepsilon} + \hat{m}_i. \quad (7)$$

where ε is a small value to prevent division by zero**.**

2.4. Research methodology

The Google Colab tool, which provides access to server capacity for a limited period, was used to train the YOLOv11n model. Training took place on local hardware with an AMD Ryzen 5 5600 processor, NVIDIA RTX 3060 graphics accelerator, and 32 GB of DDR4 RAM.

The dataset was formed using the standard COCO 2017 [14] dataset, which was limited to the following classes: person, car, motorcycle, bus, train, truck, traffic light, and stop sign. The total number of images in these classes is 50,000. To ensure balance between the classes in the dataset, the number of images in the person class was reduced, as it significantly outweighs the others in terms of number. As a result, 27,000 images were obtained, which were then divided into three parts: 18,900 images (70 %) were used for training, 5,400 (20 %) for validation, and 2,700 (10 %) for final testing.

During training configuration, the following parameters were used, which are classified as hyperparameters but are easy to configure: batch size, which was set to 64, completely filling only the allocated video memory without involving an additional buffer; models were trained for 30 epochs, which is the average value for the COCO dataset.

The hyperparameters were configured using three optimization algorithms: AdamW, SGD, and Adam, which had 10 iterations of 30 epochs each in each run. This number of iterations and epochs is sufficient to show the learning trend and the main

differences in the performance of these optimization algorithms and the impact of hyperparameters on model accuracy, although it does not provide completely complete transition processes of accuracy metrics.

2.5. Object detection platform

Nvidia Jetson Nano [23] or Raspberry Pi 5 [24] portable computers with a Hailo-8 expansion board [25] can be used as image detection hardware on portable devices, Fig. 5.

Nvidia Jetson Nano has a power of up to 67 TOPS (Tera Operations Per Second), while Raspberry Pi 5 with the Hailo-8 expansion board has 26 TOPS, which imposes restrictions on the size and power of DM in portable devices. The dimensions of the first board are 100x80x29mm, and it consumes up to 25W. Raspberry with Hailo-8 has dimensions of 86x56x18mm and consumes up to 20W, which is a more compact and energy-efficient solution.

Fig. 5. **Raspberry Pi 5 with Hailo-8 HAT**
***Source*: created by the authors**

MODELING RESULTS

The main indicator used to evaluate the effectiveness of each model is the mean accuracy (mAP), which is crucial for object detection systems. Fig. 6 shows the iterations of the optimization algorithms, where it can be seen that the SGD optimization algorithm outperforms other algorithms in terms of the integral model quality indicator Fitess, which is characterized by the average accuracy at $IoU = 0.5$ and $0.5 \le IoU \le 0.95$.

Fig. 8 compares optimization algorithms in terms of accuracy and shows that SGD outperformed both Adam and AdamW in terms of average accuracy at the $IoU = 0.5$ (mAP50) threshold by 18.2%, the average accuracy at thresholds of $0.5 \le IoU \le 0.95$ (mAP50-95) by 13.2%, and the absolute prediction accuracy by 5%, which indicates a more accurate selection of hyperparameters,specially the learning rate, initial momentum, and decay coefficient of L2 regularization. The increase in mAP directly correlates with improved object localization and classification accuracy, which is extremely important for object detection.

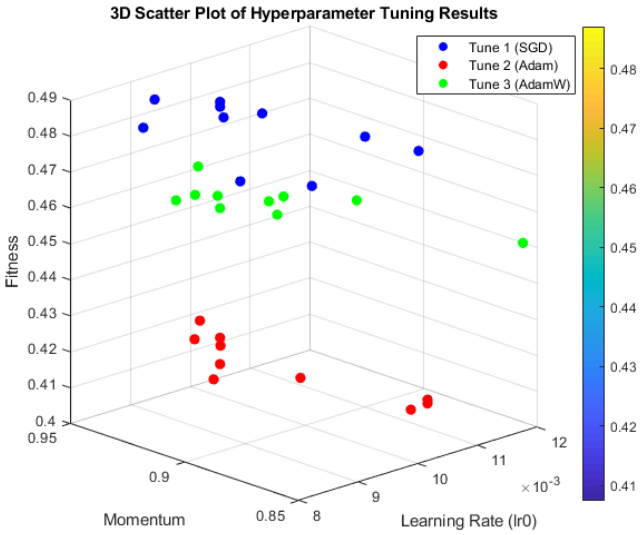

Fig. 6. **Iterations of optimization algorithms**
***Source*: created by the authors**

When configuring hyperparameters and training DM to improve its performance, a significant increase in the time spent on hyperparameter selection was observed compared to the standard training process. As a result of using optimization algorithms, the total time for parameter selection for one optimization algorithm was 65,000 seconds (about 18 hours), i.e., the time spent on training increases proportionally to the number of iterations. This time value implies the need for significant computing power to configure hyperparameters.

From Fig. 8, comparing optimization algorithms in terms of accuracy, it can be seen that an increase in mAP directly correlates with improved object localization and classification accuracy, which is extremely important for object detection. From Fig. 7, a comparison of optimization algorithms in terms of loss shows that the values of training and validation losses have a similar trend, with SGD having the lowest losses at 1.12 for training and 0.82 for validation, indicating more effective error correction during training. SGD shows the highest Precision values in the initial epochs, but in the last training period, AdamW shows similar results, with a Precision value of 0.72. The Recall value in SGD is 0.61 compared to 0.53 in Adam, which indicates a greater number of positive detections in SGD.

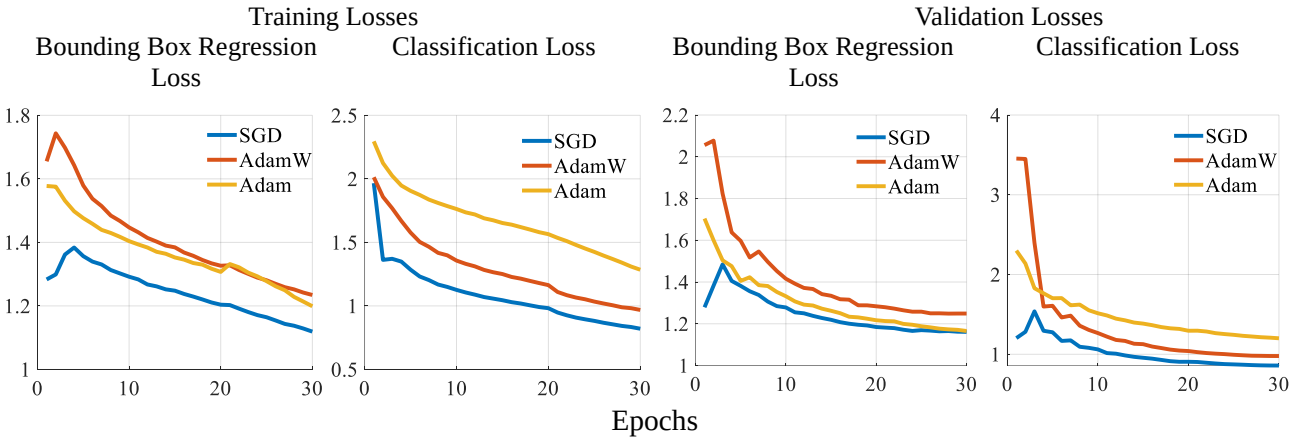

Fig. 7. **Comparison of optimization algorithms by loss indicators: a – training losses; b – validation losses**

***Source*: created by the authors**

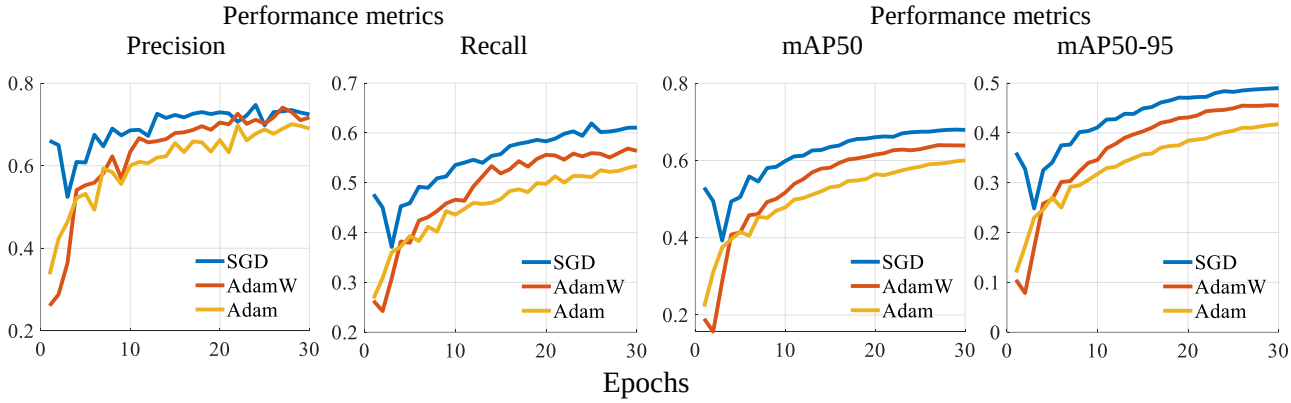

Fig. 8. **Comparison of optimization algorithms by accuracy metrics: a – Precision; b – Recall; c – mAP50; d – mAP50-95**

***Source*: created by the authors**

CONCLUSIONS

Based on the results of the study of hyperparameter tuning procedures and DM training, it was found that the SGD optimization algorithm showed the best results in terms of mAP – 0.49, demonstrating a significant advantage in accuracy over the worst mAP option – 0.41 in Adam.

Hyperparameters such as the initial learning rate (lr0) and regularization coefficient (weight_decay) have the greatest impact on detection accuracy (mAP). The significance of the impact of hyperparameters is confirmed by the fact that changing only two parameters (lr0 and weight_decay) in the best combinations led to a 13.2% increase in mAP50. Warmup parameters (warmup_epochs, warmup_momentum) ensure training stability in the initial epochs, while changing the momentum coefficient allows for faster convergence without loss of accuracy. The box, cls, and dfl parameters directly affect the loss and penalty metrics during training.

The results obtained indicate the feasibility of using the SGD optimization algorithm in tasks where detection accuracy is important, provided that sufficient computational resources are available for hyperparameter selection, i.e., for DM training. This will ensure a higher probability of successful detection by the real-time image recognition system under conditions of limited computational resources.

REFERENCES

1. “Object detection in industrial machine vision applications”. *HD Vision Systems*. – Available from: https://www.hdvisionsystems.com/en/blog/object-detection-in-industrial-machine-vision-applications. – [Accessed: 10 Jul 2025].

2. Abiodun, O. I. et al. “Comprehensive Review of artificial neural network applications to pattern recognition”. *IEEE Access*. 2019; 7: 158820–158846, https://www.scopus.com/pages/publications/85077962906?origin=resultslist. DOI: https://doi.org/10.1109/access.2019.2945545.

3. Wright, J. et al. “Sparse representation for computer vision and pattern recognition”. *Proceedings of the IEEE*. 2010; 98 (6): 1031–1044. https://www.scopus.com/pages/publications/77952717202?origin=resultslist. DOI: https://doi.org/10.1109/jproc.2010.2044470.

4. Barnell, M. et al. “Spike-Driven YOLO: ultra low-power object detection with neuromorphic computing”. *2024 IEEE High Performance Extreme Computing Conference (HPEC)*. Wakefield, USA. 2024. p. 1–5, https://www.scopus.com/pages/publications/105002718854?origin=resultslist.
DOI: https://doi.org/10.1109/hpec62836.2024.10938424.

5. Özcan, İ., Altun, Y. & Parlak, C. “Improving YOLO detection performance of autonomous vehicles in adverse weather conditions using metaheuristic algorithms”. *Applied Sciences*. 2024; 14 (13): 5841, https://www.scopus.com/pages/publications/85198479179?origin=resultslist.
DOI: https://doi.org/10.3390/app14135841.

6. Poureskandar, R. & Razzagzadeh, S. “Improving object detection performance through YOLOv8: A comprehensive training and evaluation study”. – Available from: https://www.researchgate.net/publication/391803447_Improving_Object_Detection_Performance_through_YOLOv8_A_Comprehensive_Training_and_Evaluation_Study. – [Accessed: 12 Aug 2025].

7. Ding, D., Deng, Z. & Yang, R. “YOLO-TC: An optimized detection model for monitoring safety-critical small objects in tower crane operations”. *Algorithms*. 2025; 18 (1): 27, https://www.scopus.com/pages/publications/85215972246?origin=resultslist.
DOI: https://doi.org/10.3390/a18010027.

8. “Ultralytics YOLO hyperparameter tuning guide”. *Ultralytics YOLO Docs*. – Available from: https://docs.ultralytics.com/ru/guides/hyperparameter-tuning/#default-search-space-description. – [Accessed: 14 Sep 2025].

9. Arnold, C. “The role of hyperparameters in machine learning models and how to tune them”. *Political Science Research and Methods*. 2024. p. 1–8, https://www.scopus.com/pages/publications/85185162516?origin=resultslist. DOI: https://doi.org/10.1017/psrm.2023.61.

10. LeCun, Y., Bengio, Y., Hinton, G. “Deep learning”. *Nature*. 2015; 521 (7553): 436–444, https://www.scopus.com/pages/publications/84930630277?origin=resultslist.
DOI: https://doi.org/10.1038/nature14539.

11. Pateriya, P. N. et al. “Deep residual networks for image recognition”. *International Journal of Innovative Research in Computer and Communication Engineering*. 2023; 11 (09): 10742–10747. DOI: https://doi.org/10.15680/ijircce.2023.1109026.

12. Liu, X., Gan, H. & Yan, Y. “Study on improvement of YOLOv3 Algorithm”. *Journal of Physics: Conference Series*. 2021; 1884 (1): 012031, https://www.scopus.com/pages/publications/85105430301?origin=resultslist.
DOI: https://doi.org/10.1088/1742-6596/1884/1/012031.

13. Xu J. et al. “Resture object detection and recognition based on YOLOv11”. *Applied and Computational Engineering*. 2025; 133 (1): 81–89. DOI: https://doi.org/10.54254/2755-2721/2025.20604.

14 “COCO 2017 dataset”. *Common Objects in Context*. – Available from: https://cocodataset.org/#home. – [Accessed: 14 Aug 2025].

15. “YOLO11”. *Ultralytics YOLO Docs*. – Available from: https://docs.ultralytics.com/ru/models/yolo11. – [Accessed: 11 Aug 2025].

16. Rao, S. N. “YOLOv11 explained: next-level object detection with enhanced speed and accuracy”. *Medium*. – Available from: https://medium.com/@nikhil-rao-20/yolov11-explained-next-level-object-detection-with-enhanced-speed-and-accuracy-2dbe2d376f71. – [Accessed: 14 Aug 2025].

17. “YOLOv11: An overview of the key architectural enhancements”. *arXiv.org*. – Available from: https://arxiv.org/html/2410.17725v1. – [Accessed: 16 Aug 2025].

18. Terven, Juan, Diana-Margarita Córdova-Esparza, and Julio-Alejandro Romero-González. 2023. “A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS”. *Machine Learning and Knowledge Extraction* 5, No. 4: 1680–1716,

https://www.scopus.com/pages/publications/85180465185?origin=resultslist.
DOI: https://doi.org/10.3390/make5040083.

19. “YOLO performance metrics”. *Ultralytics YOLO Docs*. – Available from: https://docs.ultralytics.com/guides/yolo-performance-metrics. – [Accessed: 20 Aug 2025].

20. Kushal, K. A. “Optimization algorithms in machine learning: a comprehensive guide to understand the concept and implementation” *Medium*. – Available from: https://medium.com/@koushikkushal95/optimization-algorithms-in-machine-learning-a-comprehensive-guide-to-understand-the-concept-and-3db1df7a2f59. – [Accessed: 20 Oct 2025].

21. Deepika, H. C., Vijayashree, R. & Srinivasan, G. N. “An overview of you only look once: unified, real-time object detection”. *International Journal for Research in Applied Science and Engineering Technology*. 2020; 8 (6): 607–609. DOI: https://doi.org/10.22214/ijraset.2020.6098.

22. Isa I. S. et al. “Optimizing the hyperparameter tuning of YOLOv5 for underwater detection”. *IEEE Access*. 2022, https://www.scopus.com/pages/publications/85131275572?origin=resultslist. DOI: https://doi.org/10.1109/access.2022.3174583.

23. “NVIDIA Jetson Nano”. *NVIDIA*. – Available from: https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-nano/product-development. – [Accessed: 20 Aug 2025].

24. “Raspberry-pi-5”. *Raspberrypi*. – Available from: https://www.raspberrypi.com/products/raspberry-pi-5. – [Accessed: 20 Aug 2025].

25. “AI processor Hailo-8 for edge devices | Up to 26 tops hardware”. *Hailo*. – Available from: https://hailo.ai/products/ai-accelerators/hailo-8-ai-accelerator. – [Accessed: 22 Aug 2025].

Conflicts of Interest: The authors declare that they have no conflict of interest regarding this study, including financial, personal, authorship or other, which could influence the research and its results presented in this article

Received 20.10.2025
Received after revision 25.11.2025
Accepted 04.12.2025

DOI: https://doi.org/10.15276/aait.08.2025.25
УДК 52-004.8

Дослідження впливу алгоритмів оптимізації на точність нейронних мереж YOLOv11

Ковбаса Сергій Миколайович[1)]
ORCID: https://orcid.org/0000-0002-2954-455X; skovbasa@ukr.net. Scopus Author ID: 55328200100
Холоша Антон Олексійович[1)]
ORCID: https://orcid.org/0009-0003-4858-202X; kholosha.anton@gmail.com
[1)] Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»
пр. Берестейський, 37. Київ, 03056, Україна

ABSTRACT

Візуальна інспекція та позиціонування по результатах детекції зображень є інтенсивно зростаючою складовою систем автоматизації. Машинний зір все ширше використовується у виробничих лініях різного технологічного призначення, а також у спеціальній техніці. Підвищення точності розпізнавання в таких застосуваннях може бути нелегкою задачею, особливо в умовах можливих обмежень, одним з яких може бути обмеження за розміром та вагою, що в свою чергу обмежує потужність комп’ютерних пристроїв, що реалізують детектування та розпізнавання зображень. Можливим рішенням цієї проблеми є підвищення точності розпізнавання за рахунок автоматичного налаштування гіперпараметрів моделей детекції з використанням різних методів оптимізації. В даній статті представлено результати дослідження ефективності алгоритмів автоматичного налаштування гіперпараметрів моделі детекції зображень YOLO (You Only Look Once), побудованих на основі методів оптимізації SGD (стохастичного градієнтного спуску), Adam (адаптивної оцінки моменту навчання), та AdamW (покращеного варіанту Adam). Навчання моделей здійснювалось на COCO 2017 датасеті, обмеженого вісьма класами та збалансованого за кількістю зображень. Для кожного алгоритму

оптимізацій використано 10 ітерацій автоматичного підбору з подальшим навчанням протягом 30 епох. За результатами тестування встановлено, що в задачах, де важливою є точність детекції за умови наявності достатніх обчислювальних ресурсів для підбору гіперпараметрів під час навчання моделі детекції, доцільно використовувати алгоритм оптимізації стохастичного градієнтного спуску, оскільки налаштована з його використанням модель детекції забезпечує більшу ймовірність успішного розпізнавання зображень в реальному часі за умов обмежених обчислювальних ресурсів.

Keywords: Штучний інтелект; комп'ютерний зір; згорткові нейронні мережі; COCO dataset; оптимізація гіперпараметрів; SGD; Adam; AdamW; YOLOv11; mAP; IoU; precision; recall

ABOUT THE AUTHORS

Serhii M. Kovbasa - Doctor of Engineering Sciences, Head of the Department of Electromechanical Systems Automation and Electrical Drives. National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", 37, Beresteiskyi Ave. Kyiv, 03056, Ukraine
ORCID: https://orcid.org/0000-0002-2954-455X; skovbasa@ukr.net. Scopus Author ID: 55328200100
***Research field*:** AC Motors Nonlinear Control Systems; Semiconductor Converters Control; DSP Based Control Systems

Ковбаса Сергій Миколайович - доктор технічних наук, завідувач кафедри Автоматизації електромеханічних систем та електроприводу. Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», пр. Берестейський, 37. Київ, 03056, Україна

Anton O. Holosha - PhD Student of Department of Electromechanical Systems Automation and Electrical Drives, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", 37, Beresteiskyi Ave. Kyiv, 03056, Ukraine
ORCID: https://orcid.org/0009-0003-4858-202X; kholosha.anton@gmail.com
***Research field*:** AC Motors Nonlinear Control Systems; Semiconductor Converters Control; DSP Based Control Systems

Холоша Антон Олексійович - аспірант кафедри Автоматизації електромеханічних систем. . Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», пр. Берестейський, 37. Київ, 03056, Україна