Exploring Yolo architecture for detection of Ukrainian car license plates in images

Oleh Petryliak[1, †], Svitlana Kostenko[1, *, †], Lesia Dobuliak[1, †] and Lyubomyr Chyrun[1, †]

[1] *Ivan Franko National University of Lviv, Universytetska 1, 79000, Lviv, Ukraine*

Abstract

The rapid advancement of deep learning has significantly improved object detection tasks, including license plate. This paper explores the application of deep learning for the detection of vehicle license plates in images. The study evaluates and compares various generations of the YOLO (You Only Look Once) model to determine the most effective architecture in terms of both accuracy and speed. YOLO models are widely used for real-time object detection due to their efficiency in processing images in a single forward pass through the neural network.

A systematic approach was applied to train and compare YOLOv5, YOLOv6, YOLOv8, YOLOv9, YOLOv10, and YOLOv11. Each model was trained on a dataset of vehicle images containing license plates, with annotations marking the exact location of the plates. The training process lasted 100 epochs, allowing the models to stabilize and reach optimal performance. The evaluation metrics included mean Average Precision (mAP) at different IoU thresholds, detection speed, and generalization capability on unseen data. Based on the analysis, YOLOv9 was selected as the most efficient model, balancing accuracy and processing speed.

The selected models were tested on real-world images of vehicles captured under varying conditions, including different angles, lighting, and license plate formats such as Cyrillic characters, green electric vehicle plates, and American-style layouts. The results confirmed the robustness of most YOLO models, although YOLOv11 showed lower stability in more complex scenarios.

The findings of this research highlight that careful evaluation of YOLO architectures enables the selection of an effective model for automatic license plate detection. The chosen YOLOv9 model demonstrates strong performance and is well suited for real-time applications such as traffic monitoring, automated access control, and law enforcement.

Keywords

license plate detection, deep learning, computer vision, model comparison.

1. Problem statement

Automated license plate detection is an important task in the areas of traffic control, security and traffic management. Such systems are essential for law enforcement agencies, utilities, and traffic monitoring companies, allowing them to effectively detect traffic violators, track down stolen cars, and automate access control processes at various facilities. Modern approaches based on deep learning, such as the YOLO architecture [1], demonstrate significant improvements in the speed and accuracy of object detection. However, the effectiveness of different generations of YOLO for the task of number plate detection remains insufficiently studied. It is important to determine which architecture best balances performance and accuracy in the face of varying lighting conditions, different angles, and possible object overlap.

This study focuses on Ukrainian number plates, which are distinguished by a variety of formats and styles. The main difficulty is posed by number plates of a different design used on imported cars,

MoMLeT-2025: 7th International Workshop on Modern Machine Learning Technologies, June, 14, 2025, Lviv-Shatsk, Ukraine

* Corresponding author.

† These authors contributed equally.

oleh.petryliak@gmail.com (O. Petryliak); svitlana.kostenko@lnu.edu.ua (S. Kostenko); lesia.dobuliak@lnu.edu.ua (L. Dobuliak); lyubomyr.chyrun@lnu.edu.ua (L. Chyrun)

0009-0004-4717-6617 (O. Petryliak); 0000-0003-0823-9801 (S. Kostenko); 0000-0001-8665-8783 (L. Dobuliak); 0000-0002-9448-1751 (L. Chyrun)

which are characterized by different sizes, fonts and character arrangements. Additional challenges are posed by Cyrillic characters, where some characters are visually similar to Latin characters, which can cause confusion during detection.

Particular attention is paid to the detection of number plates with non-standard color schemes, such as green number plates used for electric vehicles and red number plates typical of transit vehicles. These color variations require the model to adapt to different levels of contrast and illumination, which increases the complexity of the number plate detection task.

YOLO (You Only Look Once) is one of the most popular deep learning architectures for detecting objects in images and classifying them. The main feature of this architecture is the ability to search for objects in real time, processing images in one pass through the neural network, which ensures high speed of the model while maintaining accuracy.

The YOLO prediction principle is based on the fact that the input image is divided into a grid of size $S \times S$, where S — the number of horizontal and vertical image divisions, which is selected depending on the size of the input image and the model architecture (for example, 7×7 or 13×13). Each cell of this grid is used to detect objects whose center falls within its boundaries, so the model makes predictions for each cell:

1. The parameters of the bounding box (a rectangular area that outlines the detected object in the image): the coordinates of the upper left corner of the box (X_0, Y_0), its width W and height H.
2. Confidence score — a numerical indicator that indicates the probability that the bounding box really contains an object, as well as the accuracy of the predicted box coordinates. If the confidence value is below a certain threshold, the prediction can be rejected as unreliable.
3. Object class — the category to which the found object belongs.

The final output of the model is a matrix of size $S \times S \times (B \times 5 + C)$, where B the number of predicted frames for each cell, 5 — the number of parameters of each frame (coordinates of the upper left corner of the frame, width, height, and confidence), and C — the number of possible classes.

Next, the Non-Maximum Suppression (NMS) method [2] is used to remove unnecessary frames.

The results of the model are the parameters of the bounding boxes that most closely match the objects in the image and the classification of these objects.

The described approach is implemented by a neural network whose architecture consists of three main parts (Fig. 1): Backbone, which detects features in the input image, Neck, which combines features to improve the quality of detecting objects of different scales, and Heads, which performs the final prediction of frames and classes.

Before training the neural network and setting up the YOLO model, a training set must be prepared in a special way. This sample should consist of a set of image files with objects to be recognized and their annotations in a format supported by YOLO models. This format includes, for each image, the class identifier of the existing object, the coordinates of the center of its bounding box, and the width and height of the box normalized to the image size. This dataset is used by the model for training, allowing it to identify license plate features such as shape, proportions, and contrast relative to the background.

YOLO training is based on minimizing the Loss Function, which is the sum of three components, such as Localization Loss (mean square error (MSE) for the predicted coordinates of the bounding box), Classification Loss (cross-entropy or MSE for the object class prediction), and Objectness Loss (probability that the box contains an object).

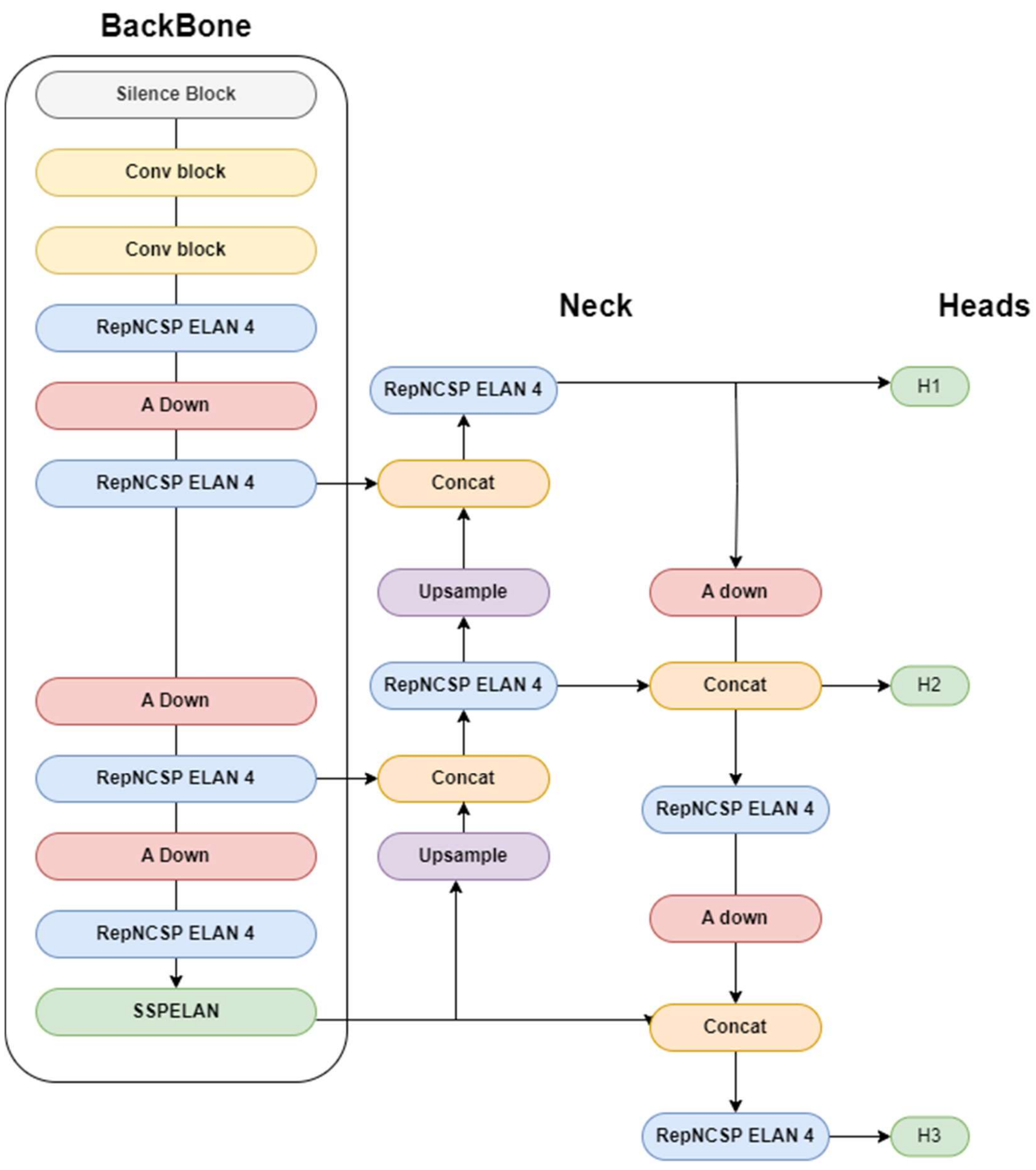


Figure 1: Architecture of the YOLOv9 model.

2. YOLO generations

During the development of the YOLO architecture, eleven generations from YOLOv1 to YOLOv11 have appeared, each of which takes into account the shortcomings of the previous one and offers certain improvements to increase the accuracy, speed and efficiency of work, as well as adaptation to new tasks:

1. YOLOv1 (2015): Introduced the concept of “You Only Look Once” by offering a one-step approach to object detection. For the first time, the image was divided into a grid, predicting the coordinates of the frames and classes of objects in each cell.
2. YOLOv2 (2016): Batch Normalization was added, Anchor Boxes and Dimension Clusters were introduced, which improved the accuracy and stability of training.
3. YOLOv3 (2018): A multi-level model architecture is used to handle objects of different sizes, as well as an improved base layer (Darknet-53), which improved speed and accuracy.
4. YOLOv4 (2020): Introduced Mosaic – data augmentation, a new Anchor-Free Detection Head and an optimized loss function. The model became more efficient on large data sets.

5. YOLOv5 (2020): Automatic optimization of hyperparameters, integration of an experiment tracking system, and the ability to export models to various formats have been provided, making it popular for commercial use.
6. YOLOv6 (2022): Introduced Bi-directional Concatenation module to improve localization signals, an anchor-aided training strategy that combines the benefits of both approaches, and improved Base and Neck to achieve the best accuracy on Microsoft's Common Objects in Context (COCO) dataset.
7. YOLOv7 (2022): Added support for Pose Estimation tasks and other improvements, such as optimization for keypoints.
8. YOLOv8 (2023): Expanded functionality to support all major computer vision tasks: detection, segmentation, classification, pose estimation. Improved accuracy and efficiency of computing resources.
9. YOLOv9 (2024) [3]: Implemented Programmable Gradient Information and Generalized Efficient Layer Aggregation Network technologies, which provides more flexible work with gradients.
10. YOLOv10 (2024): The concept of End-to-End Detection Head is integrated, instead of Non-Maximum Suppression (NMS). This has reduced real-time delays.
11. YOLOv11 (2024): Improved architecture and increased efficiency through optimization for different environments, support for different export formats, and the ability to work on different platforms.

3. Research and publication analysis

The effectiveness of using different generations of YOLO for vehicle, road sign, and license plate detection has been considered in many scientific papers. In particular, [4] analyzes the results of using YOLO for automated license plate detection. The study confirms that YOLO is an effective approach, but the main challenges remain the variability of lighting, shooting angles, and partial overlap of license plates.

Study [5] provides an overview of methods for improving YOLO performance for real-world applications, including the use of various options after processing the detection results. The authors note that to improve accuracy, it is necessary to adapt the model to specific environmental conditions and improve the balance between speed and performance.

Paper [6] investigates the impact of using pre-trained models and methods before training YOLO on new datasets. The authors emphasize that adapting models to specific scenarios significantly increases their efficiency, especially in conditions of variable lighting and low image quality.

Also worth mentioning is the VEZHA LPR system from Incoresoft [7], which is actively used for automatic recognition of Ukrainian license plates in video surveillance and traffic speed control systems. VEZHA LPR demonstrates high accuracy in difficult real-world conditions, such as fast traffic, poor lighting, difficult angles, and partial overlap of license plates.

While such systems are already in operation, there is a need for systematic academic evaluation and comparison of different deep learning architectures tailored to Ukrainian license plate formats. This research aims to fill that gap by analyzing and benchmarking various YOLO model generations, which can serve as a basis for developing flexible, open, and easily deployable solutions or for enhancing existing systems with empirically validated architectures.

4. Approaches to assessing model performance

To evaluate the performance (efficiency) of a model, YOLO model training uses a function called fitness, which is a weighted sum of key machine learning metrics calculated based on the accuracy of the predictions made for the objects in the training set [8]:

$$Fitness = w_1 * P + w_2 * R + w_3 * mAP50 + w_4 * mAP50\text{-}95, \quad (1)$$

where P – precision, R – recall, $mAP50$ – mean average precision calculated at an intersection over union (IoU) threshold of 0.50, $mAP50\text{-}95$ – average of the mean average precision calculated at varying IoU thresholds, ranging from 0.50 to 0.95. The weighting coefficients w_i are chosen heuristically, based on practical experience. By default, $w_1 = w_2 = 0$, $w_3 = 0{,}1$, $w_4 = 0{,}9$. In this study, different weights were applied, specifically $w_1 = w_2 = 0{,}1$, $w_3 = 0{,}3$, $w_4 = 0{,}5$.

Precision and Recall are calculated as follows:

$$P = \frac{TP}{TP + FP}, R = \frac{TP}{TP + FN}, \tag{2}$$

where TP (True Positive) – the number of correctly detected objects, FP (False Positive) – the number of incorrectly detected object, FN (False Negative) – the number of actual objects that were missed.

A key metric for assessing prediction accuracy in object detection tasks is the Jaccard Index (IoU), which is defined as the ratio of the area of intersection to the area of union between the ground truth bounding box and the predicted bounding box:

$$IoU = \frac{|B_p \cap B_r|}{|B_p \cup B_r|}, \tag{3}$$

where B_p – the bounding box predicted by the model, B_r – the ground truth bounding box.

Average Precision (AP) – the area under the *AUC* (*area under ROC curve)*, formed by the ROC curve, which is based on $P(R)$ – the dependence of precision on recall, i.e.

$$AP = \int_0^1 P(R)\, dR, \tag{4}$$

The average accuracy for the threshold value of the Jaccard coefficient $IoU = 0.5$ is defined as:

$$mAP50 = AP(IoU = 0.5). \tag{5}$$

The average accuracy in the range of IoU thresholds from 0.5 to 0.95 with a step of 0.05 is calculated by the formula:

$$mAP50\text{-}95 = \frac{1}{10} \sum_{t=0.5}^{0.95, \Delta=0.05} AP(IoU = t). \tag{6}$$

5. Data preparation, training, and object detection stages

For this task, we collected about 11500 images of cars from open sources using an automated parser based on the Selenium library [9]. To annotate the data, we used the CVAT (Computer Vision Annotation Tool) [10], which generates a text annotation of the required format according to the rectangular area with an object selected in the image. In our case, the objects we are looking for in the images are license plates. They will be assigned to the class with the identifier 0. That is, the model is trained exclusively on detecting license plates without dividing them into subcategories.

Figure 2 shows an example of a car image with license plate annotation (marked with the area where the license plate is located), and Figure 3 shows the corresponding text file with the required parameters.

The resulting sample was randomly divided into training and test samples. The training set contains 10000 images that were used to train the models, while the remaining 1500 images were included in the test set to evaluate the quality of license plate recognition and determine the speed of the model.

For the study, we trained the YOLOv5, YOLOv6, YOLOv8, YOLOv9, YOLOv10, and YOLOv11 models. YOLOv7 was not included in the experiments due to the lack of significant improvements in object detection compared to other versions. The obtained results allowed us to determine the most optimal architecture for the license plate detection task, taking into account the balance between accuracy, speed and stability of the model.

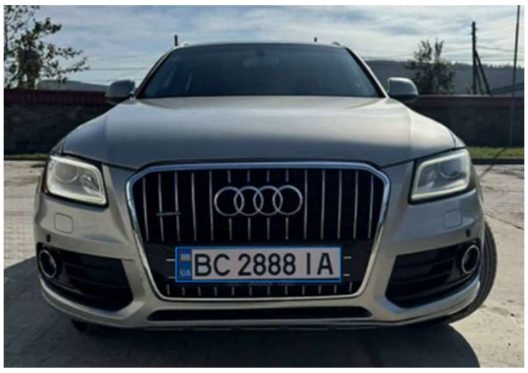
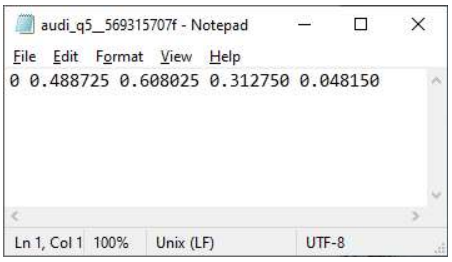


Figure 2: An example of a car image with a bounding box around the license plate

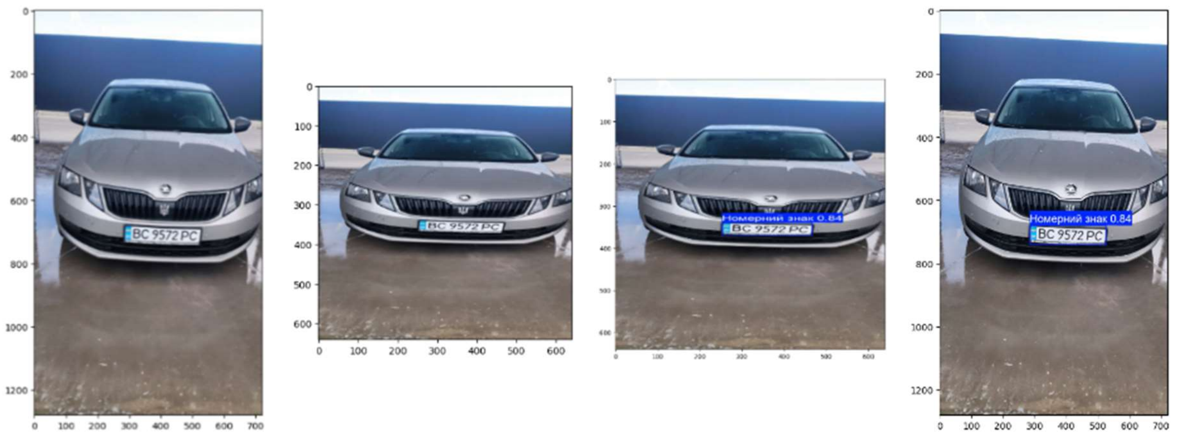

audi_q5_569315707f - Notepad
File Edit Format View Help
Ln 1, Col 1 100% Unix (LF) UTF-8

```
0 0.488725 0.608025 0.312750 0.048150
```

Figure 3: An example of a text file with an object annotation

The process of training YOLO models of different generations lasted 100 epochs using standard settings (batch size – 16, optimizer – Adam, input image size – 640 pixels, frame error weight – 7.5, class identification error weight – 0.5). The training was performed using PyTorch [11] on a system with an Intel i5-14600KF, Nvidia RTX 4070, and 32 GB of RAM. The total training time was:

- YOLOv5 – 7494,9 seconds
- YOLOv6 – 7099,8 seconds
- YOLOv8 – 6819,7 seconds
- YOLOv9 – 6584,6 seconds
- YOLOv10 – 8744,7 seconds
- YOLOv11 – 8073,9 seconds



a) b) c) d)

Figure 4: License plate prediction process using the YOLO model: a) input image; b) scaled image to the standard model size; c) model prediction on the standard model size image; d) display of the prediction on the input image

With the help of the obtained YOLO models, the search (prediction) of a license plate in a new image consists of several main stages (Fig. 4).

At the first stage, the model accepts an image of arbitrary size (Fig. 4.a), which must be scaled to a standard size in accordance with the requirements of the input layer of the neural network. In the case of YOLO, the image is scaled to a square, usually 640×640 pixels (Fig. 4.b).

After that, the model makes predictions: the neural network processes the image and finds potential objects based on the obtained characteristics. The output of the model contains the parameters of the detected bounding box, the object class, and the level of confidence in the prediction. Figure 4.c shows the result of prediction on a 640×640 pixels image.

At the last stage, the frame is scaled and applied to the original image (Fig. 4.d). This is necessary to preserve the input resolution and correctly display the detection results in real conditions, i.e., the exact positioning of the bounding box and the correspondence of the prediction to the real coordinates of the license plate.

6. Comparison of the performance of YOLO models of different generations

Using the test sample, we calculated the performance indicators of each model. Table 1 shows the results of evaluating the performance of different generations of YOLO based on four key metrics: Precision (*P*), Recall (*R*), mean average precision at *IoU = 0.5* (*mAP50*), and average of the mean average precision at *IoU* from 0.5 to 0.95 (*mAP50-95*). Based on these values, we calculated the overall fitness of the model, which allows us to assess its balance between accuracy, completeness, and overall detection quality.

Table 1
YOLO models performance metrics

Model	P	R	mAP50	mAP50-95	Fitness
YOLO5	0.975	0.964	0.973	0.768	0.87
YOLO6	0.978	0.966	0.973	0.777	0.875
YOLO8	0.975	0.959	0.973	0.776	0.874
YOLO9	0.973	0.951	0.971	0.781	0.874
YOLO10	0.973	0.949	0.969	0.766	0.866
YOLO11	0.973	0.968	0.979	0.782	0.879

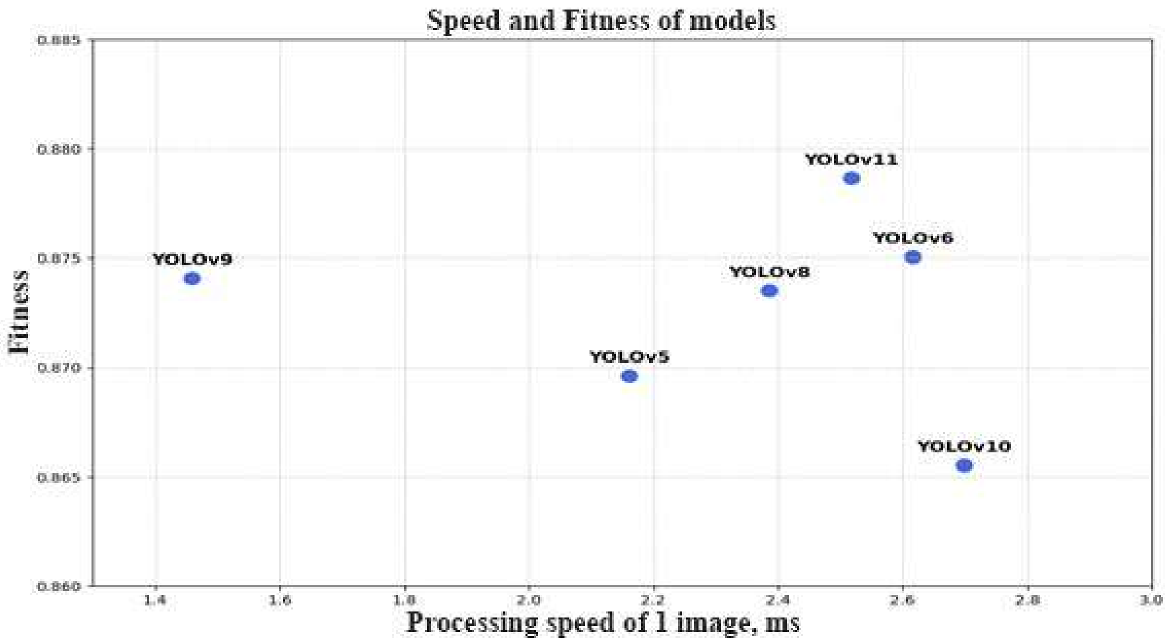


Figure 5: Speed and fitness of YOLO models of different generations

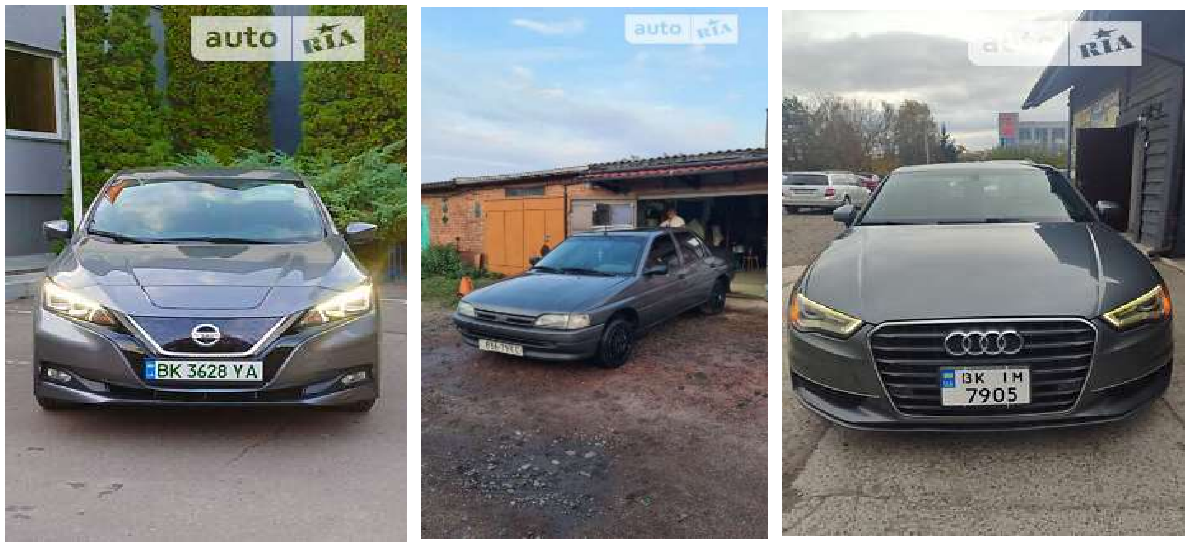
Figure 5 shows the relationship between the average processing speed of one image from the test set for object detection (on the X-axis, measured in milliseconds) and the fitness of the models (on the Y-axis). The fastest model was YOLOv9 with a processing time of about 1.45 ms, making it the best choice for applications where performance is critical. The YOLOv6, YOLOv8, and YOLOv11 all have slightly faster processing times (approximately 2.4-2.6 ms) while still providing a high level of usability, making them well-balanced options. The slowest was YOLOv10 with a time of about 2.7 ms, which, combined with its low fit, makes it the least effective model.

The fitness of the models confirms the high potential of YOLOv11, which demonstrates the best ratio of accuracy to efficiency. Despite the highest speed, YOLOv9 has a slightly lower suitability score, which may be due to the trade-off between accuracy and speed.

In summary, the YOLOv9 is the best choice as it provides the highest image processing speed with an acceptable level of fitness.

7. Comparison of prediction accuracy of YOLO models of different generation

For the first experiment on license plate detection, three images with different types of Ukrainian license plates were selected: green license plates for electric cars, Cyrillic license plates of the standard format, and American license plates.



a) b) c)

Figure 6: Varieties of Ukrainian license plates: a) license plate of an electric vehicle; b) license plate with Cyrillic characters; c) license plate of the American format

Table 2 contains the parameters of the bounding boxes with license plates detected in the images (Fig. 6). Each row of the table contains the coordinates of the upper left corner of the frame (X_0, Y_0), as well as its width(W) and height(H), calculated by a YOLO model of a specific generation for the corresponding image. For comparison, the ground truth coordinates are also provided.

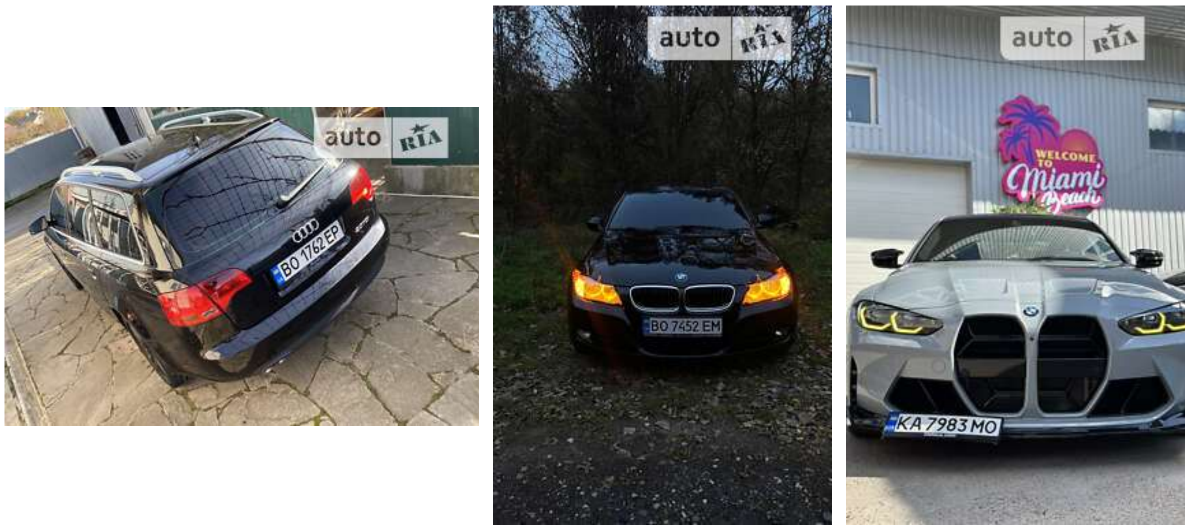
The results presented in Table 2 demonstrate the overall consistency of various YOLO generations in detecting license plates in images: the models produce nearly identical bounding box coordinates compared to the ground truth, indicating strong alignment in their performance.

Table 2
Coordinates of predicted and actual bounding boxes for license plates in test images

Figure / Model	6.a				6.b				6.c			
	X_0	Y_0	W	H	X_0	Y_0	W		X_0	Y_0	W	H
YOLOv5	121	259	72	16	102	512	57	26	124	272	55	29
YOLOv6	121	259	72	16	102	512	57	28	124	272	55	30
YOLOv8	121	259	72	16	102	512	58	27	124	272	55	30
YOLOv9	121	259	72	16	102	512	58	27	124	272	55	28
YOLOv10	121	258	72	17	102	512	58	27	124	272	55	29
YOLOv11	120	259	71	16	102	513	57	23	124	270	53	31
Ground truth	121	259	73	15	102	512	58	27	124	271	55	30

The largest deviation in license plate detection is observed in Figure 6.b, likely due to the challenging angle and specific lighting conditions. The car is positioned at an angle that alters the perspective of the license plate, potentially complicating accurate detection by YOLO models. YOLOv11 delivered the weakest result on this image. Meanwhile, other YOLO versions provided similar predictions, confirming the general robustness of the architecture to such challenges. This highlights potential directions for further optimization, particularly in improving the adaptability of models to variations in license plate orientation.

The non-standard height of the license plate in Figure 6.c contributes to differences in the predicted bounding box dimensions.



a) b) c)

Figure 7: Examples of challenging conditions for license plate detection: a) angled license plate; b) poor lighting; c) non-standard placement

For the second experiment, images with challenging conditions for automatic license plate detection were selected (Fig. 7): an angled license plate, poor lighting, and non-standard placement. Table 3 contains the parameters of the detected bounding boxes for the license plates, obtained using YOLO models of different generations.

The results of model testing on the presented images demonstrate generally high accuracy in license plate detection, with minor deviations caused by lighting and angle variations. However, in all of the given cases, the YOLOv11 model showed the weakest performance, with more significant errors in determining the bounding box position, especially in terms of height. Other models produced results close to the ground truth coordinates.

The analysis confirms that YOLO models are generally effective for license plate detection under standard conditions. However, challenging factors such as angled views, insufficient lighting, and non-standard plate placement can impact prediction accuracy, indicating potential directions for further model optimization.

Table 3
Coordinates of predicted and actual bounding boxes for license plates in challenging images

Figure / Model	7.a				7.b				7.c			
	X_0	Y_0	W	H	X_0	Y_0	W	H	X_0	Y_0	W	H
YOLOv5	239	115	57	58	139	234	58	14	72	306	85	20
YOLOv6	239	114	58	58	139	234	58	14	72	306	84	20
YOLOv8	239	115	59	58	138	234	58	14	72	307	84	20
YOLOv9	239	114	58	61	139	234	58	14	71	306	84	20
YOLOv10	239	114	58	57	139	234	58	15	72	307	85	19
YOLOv11	242	112	62	59	139	235	57	13	71	306	83	18
Ground truth	239	115	59	59	139	235	60	15	72	308	84	20

8. Conclusions

This study presents a comparative analysis of the effectiveness of various generations of YOLO models (YOLOv5, YOLOv6, YOLOv8, YOLOv9, YOLOv10, and YOLOv11) for the task of detecting Ukrainian vehicle license plates. The evaluation was based on key metrics: precision, recall, mean average precision (mAP) at different Intersection over Union (IoU) thresholds, and image processing speed. Based on the results, YOLOv9 was identified as the most effective model, providing an optimal balance between accuracy and speed.

Testing the models on images with different formats of Ukrainian license plates (green for electric vehicles, Cyrillic, and American-style) as well as under challenging conditions (angled view, poor lighting, and non-standard placement) demonstrated the overall robustness of YOLO models. However, YOLOv11 showed the lowest stability among all tested models, particularly in cases involving difficult angles and non-standard plate positioning.

Therefore, YOLOv9 is recommended as the optimal architecture for real-world applications of automated Ukrainian license plate detection due to its high accuracy, speed, and reliability. Future research may focus on improving model performance under more complex conditions and integrating the models into practical monitoring and video surveillance systems.

Declaration on Generative AI

During the preparation of this work, the authors used X-GPT-4 and Gramby in order to: Grammar and spelling check. After using these services, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] A. Aggarwal, YOLO Explained. https://medium.com/analytics-vidhya/yolo-explained-5b6f4564f31

[2] V. S. Subramanyam, Non Max Suppression (NMS). Medium. https://medium.com/ analytics-vidhya/non-max-suppression-nms-6623e6572536

[3] S. Dixit, Understanding Multi-Headed YOLO-v9 for Object Detection and Segmentation. Medium. https://medium.com/@srddev/understanding-multi-headed-yolo-v9-for-object-detection-and-segmentation-8923ee21b652

[4] S. M. Silva, C. R. Jung, License plate detection and recognition in unconstrained scenarios // 2018 European Conference on Computer Vision (ECCV), 2018, pp. 580-596. https://openaccess.thecvf.com/content_ECCV_2018/papers/Sergio_Silva_License_Plate_ Detection_ECCV_2018_paper.pdf

[5] R. Laroca, E. Severo, L. Zanlorensi et al. A robust real-time automatic license plate recognition based on the YOLO detector // Journal of Visual Communication and Image Representation, 2021, volume 74, 103013. https://www.researchgate.net/ publication/323444033_A_Robust_Real-Time_Automatic_License_Plate_ Recognition_ Based_on_the_YOLO_Detector

[6] Redmon J., Farhadi A. YOLO9000: better, faster, stronger // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 7263-7271. https://ieeexplore.ieee.org/document/8100173

[7] Incoresoft. VEZHA LPR. https://vezhavms.com/moduli-vezha/rozpiznavannya-avtonomeriv-lpr/

[8] Hyperparameter Evolution for YOLOv5. https://docs.ultralytics.com/yolov5/ tutorials/hyperparameter_evolution/

[9] Selenium Documentation. https://www.selenium.dev

[10] Computer Vision Annotation Tool. https://app.cvat.ai/tasks

[11] PyTorch Documentation. https://pytorch.org/