UDC 004.04:623.746.519
DOI: 10.15587/2706-5448.2025.323322

**Pylyp Prystavka,**
**Olha Cholyshkina,**
**Valerii Zivakin,**
**Borys Stetsenko**

# EVALUATION OF THE PERFORMANCE OF DATA CLASSIFICATION MODELS FOR AERIAL IMAGERY UNDER RESOURCE CONSTRAINTS

*The object of the study is the process of aerial imagery data processing under limited computational resources, particularly onboard unmanned aerial vehicles (UAVs) using classification models.*

*One of the most challenging issues is the adaptation of classification models to scale variations and perspective distortions that occur during UAV maneuvers. Additionally, the high computational complexity of traditional methods, such as sliding window approaches, significantly limits their applicability on resource-constrained devices.*

*The study utilized state-of-the-art neural network classifiers, including ResNet50v2, DenseNet121, and MobileNetV2, which were fine-tuned on a specialized aerial imagery dataset.*

*An experimental evaluation of the proposed neural network classifiers was conducted on Raspberry Pi 4 Model B and OrangePi 5 Pro platforms with limited computational power, simulating the constrained resources of UAV systems. To optimize performance, a stripe-based processing approach was proposed for streaming video, ensuring a balance between processing speed and the amount of analyzed data for surveillance applications. Specific execution time evaluations were obtained for different types of classifiers running on single-board computers suitable for UAV deployment.*

*This approach enables real-time aerial imagery processing, significantly enhancing UAV system autonomy. Compared to traditional methods, the proposed solutions offer advantages such as reduced power consumption, accelerated computations, and improved classification accuracy. These results demonstrate high potential for implementation in various fields, including military operations, reconnaissance, search-and-rescue missions, and agricultural technology applications.*

***Keywords:*** *neural networks, machine learning, image processing, classification, convolutional neural networks, unmanned aerial vehicles.*

*Received: 02.12.2024*
*Received in revised form: 27.01.2025*
*Accepted: 14.02.2025*
*Published: 24.02.2025*



***How to cite***
*Prystavka, P., Cholyshkina, O., Zivakin, V., Stetsenko, B. (2025). Evaluation of the performance of data classification models for aerial imagery under resource constraints. Technology Audit and Production Reserves, 1 (2 (81)), 43–48. https://doi.org/10.15587/2706-5448.2025.323322*

## 1. Introduction

One of the key challenges in aerial imagery data processing is ensuring high-performance image processing systems under limited computational resources onboard unmanned aerial vehicles (UAVs) [1]. Traditional approaches to camera data analysis require significant processing time, making real-time applications infeasible, particularly in critical situations [2]. For instance, in military operations, delays in object detection may lead to the loss of crucial tactical advantages, while in search and rescue missions, such delays can reduce the effectiveness of locating victims [3].

This issue becomes particularly relevant under the constraints of UAV autonomy, which necessitates energy-efficient solutions. The use of efficient models for real-time object classification and segmentation can significantly reduce the need for data transmission to ground stations, thereby improving both the autonomy and functionality of the system [4].

In this work, the most widely used deep learning models for addressing the task of target object detection in UAV camera data are reviewed.

YOLO (You Only Look Once) is one of the most popular deep learning architectures for real-time object detection tasks [5, 6]. The core idea of YOLO is to simplify the detection process by simultaneously predicting object boundaries and classifications in a single pass through the network, significantly improving performance compared to traditional approaches such as R-CNN and Fast R-CNN.

YOLO demonstrates high accuracy in object detection tasks, achieving a Mean Average Precision (mAP) of 65–75 % in YOLOv8 on datasets such as COCO. However, accuracy may degrade under poor lighting conditions, high object density, or when detecting small objects that require finer details. YOLO also exhibits a low false positive rate due to its optimized loss function and a well-balanced trade-off between object localization and classification.

Furthermore, YOLO can operate on mid-range GPUs (e. g., NVIDIA GTX 1660) and even on CPUs with reduced performance. The affordability of such hardware makes it a viable option for most research and commercial projects.

Despite its significant advantages, the YOLO model has notable limitations when applied to UAV-mounted cameras, primarily due to the unique characteristics of aerial imagery and computational constraints. UAVs operate in a three-dimensional space, where the camera continuously changes its tilt angle due to maneuvers or altitude adjustments. This results in complex perspective distortions and unusual viewing angles, to which YOLO is not well adapted.

Objects are captured from varying altitudes, leading to significant scale variations within images (poor scale invariance). Although YOLO

ISSN-L 2664-9969; E-ISSN 2706-5448

supports multi-scale processing, it relies on a fixed grid-based partitioning of the image, making it less accurate for objects that significantly vary in scale. Unlike stationary cameras or vehicle-mounted cameras operating in predictable environments, UAV cameras capture images in complex and dynamic settings. These conditions may include rapid lighting changes, moving objects within the frame, and a highly variable background. YOLO does not always perform effectively in such scenarios due to its lack of context-aware adaptation mechanisms.

Furthermore, even optimized YOLO versions (e. g., Tiny-YOLO) may still be too computationally demanding for typical single-board computers such as Raspberry Pi or OrangePi, limiting the feasibility of integrating this model.

Several modern architectures are worth considering, given their specific features that may be beneficial for various computer vision tasks, particularly when processing UAV camera data.

EfficientNet is a modern model that employs automated architecture optimization to achieve a balance between performance and computational cost [6]. It provides high accuracy with significantly fewer parameters compared to other large models, making it suitable for resource-constrained systems such as UAVs. The model demonstrates efficient scaling of depth, width, and resolution, as well as relatively high detection accuracy, but requires specialized optimization for real-time operation.

Vision Transformers (ViT) [7] utilize a self-attention mechanism that effectively captures long-range dependencies between different parts of an image. They perform well in tasks requiring the analysis of complex structures and exhibit high accuracy on large datasets. However, their high computational demands limit their usability on low-power platforms, such as those used in UAVs.

NASNet (Neural Architecture Search Network) is the result of an automated architecture search optimized for classification and detection tasks [8]. Its flexibility allows for adaptation to various computational resources. The model achieves high accuracy due to automated optimization. However, its drawbacks include long training times and high resource costs for initial optimization.

Faster R-CNN is a classic model for object detection tasks. Its accuracy and flexibility in handling different data types make it a valuable tool for researchers. A key advantage of the model is its high accuracy in detection tasks, making it applicable to complex environments with high object density. However, significant computational costs hinder its real-time deployment on UAVs.

SqueezeNet is specifically designed for operation in resource-constrained environments. It features a compact architecture (~1.2 million parameters), allowing for high performance with minimal resource consumption, making it suitable for embedded systems. However, its main drawback is lower recognition accuracy compared to other modern models [9].

Thus, each of the reviewed models demonstrates distinct advantages but also has significant limitations. The primary challenges include high computational complexity, limited scale and perspective distortion invariance, and reduced performance in dynamic conditions or on resource-constrained platforms typical of UAVs. Considering these limitations, *the aim of this study* is to enhance the efficiency of aerial image processing onboard UAVs under restricted computational resources. This will be achieved through the investigation, adaptation, and optimization of modern classification models such as ResNet50v2, DenseNet121, and MobileNetV2. Additionally, let's aim to develop methods for reducing computational costs, including information filtering and linear path analysis.

## 2. Materials and Methods

In this study, let's examine the following neural network models.

ResNet50, or Residual Network-50, is one of the most well-known deep neural networks developed for image classification. This architecture was designed to address the vanishing gradient problem that arises when training very deep networks. The core idea is the introduction of residual blocks, which include direct connections or shortcut connections that bypass one or more layers, allowing gradients to propagate more easily through the network during backpropagation.

ResNet50v2 is an improved version of ResNet50 [10]. The primary differences between ResNet50 and ResNet50v2 lie in the placement of batch normalization and ReLU activation. In ResNet50v2, these operations are performed before each convolution, improving network stability and training efficiency. Additionally, in ResNet50v2, shortcut connections are added after activation, enhancing gradient flow. The placement of batch normalization before activation and convolution is a key improvement that results in better performance and faster training.

DenseNet121, or Densely Connected Convolutional Network 121 [11], is a deep neural network distinguished by its densely connected layer architecture. This design improves information flow and training efficiency compared to traditional convolutional neural networks.

The core idea of DenseNet121 is that each layer in the network receives input from all preceding layers and passes its features to all subsequent layers. This means that the network consists of dense blocks, where each layer is connected to every other layer within the block. This approach mitigates the vanishing gradient problem, enhances feature reuse, and reduces the number of parameters, as layers do not need to recompute the same features repeatedly.

MobileNetV2 is an extension of the MobileNet architecture, specifically designed for mobile and embedded systems where limited resources necessitate efficient neural networks. MobileNetV2 is based on novel concepts such as inverted residual blocks and linear bottlenecks, which enable high performance while reducing computational costs [12].

Inverted residual blocks are the key innovation of MobileNetV2. In traditional residual networks (e. g., ResNet), channel expansion is performed at the end of the block. In contrast, MobileNetV2 first increases the number of channels and then reduces them. This approach preserves more useful information throughout the network.

A comparison of the described models is presented in Table 1 [10–12].

For the deployment and testing of models, single-board computers Raspberry Pi 4 Model B and OrangePi 5 Pro were selected to evaluate their suitability for onboard integration in UAVs.

A comparison of their specifications is presented in Table 2 [13].

**Table 1**

Comparison of neural network model characteristics

| Parameter | ResNet50v2 [10] | DenseNet121 [11] | MobileNetV2 [12] |
|---|---|---|---|
| Network depth | 50 layers | 121 layers | 53 layers |
| Connection approach | Residual blocks | Dense connections | Inverted residual blocks |
| Feature growth rate | Not used | 32 | Not used |
| Number of parameters | ~23.5 M | ~8 M | ~3.4 M |
| Computational complexity | High | Medium | Low |
| Mobile optimization | No | No | Yes |
| Primary application | Large datasets | General-purpose | Mobile and embedded systems |
| Training time | Long | Medium | Fast |
| Real-time performance | Medium | High | Very high |

**Table 2**

Comparison of the specifications of single-board computers Raspberry Pi 4 Model B and OrangePi 5 Pro [13]

| Characteristic | Raspberry Pi 4 Model B | OrangePi 5 Pro |
|---|---|---|
| Processor | ARM Cortex-A72, 4 cores, 1.5 GHz | Rockchip RK3588, 8 cores (4x Cortex-A76, 4x Cortex-A55) |
| RAM | Up to 8 GB LPDDR4 | Up to 32 GB LPDDR4/4x |
| Graphics processor (GPU) | VideoCore VI (supports OpenGL ES 3.0) | Mali-G610 (high performance in graphics and AI) |
| Power consumption | 5–10 W | 10–15 W |
| Operating system | Raspberry Pi OS, Linux support | Debian, Ubuntu, Android |
| Performance | Suitable for basic computing and educational projects | High performance, suitable for AI and multimedia |
| Cost | Relatively low | Medium (higher than Raspberry Pi) |
| Advantages | Accessibility, ease of use, low cost | High performance, support for modern standards |
| Disadvantages | Limited performance for complex tasks | Higher power consumption, higher price |

For model training, software was implemented in Python using TensorFlow and Keras. The models ResNet50v2, DenseNet121, and MobileNetV2 were fine-tuned on the "Aerial Imagery" dataset [14], which contains images captured from UAVs with a resolution of 64×64 pixels. The dataset consists of more than 260,000 images and includes 43 classes (Fig. 1).

To address the issue of prolonged classification time using the sliding window method, approaches based on horizontal or vertical line traversal were proposed, along with the application of information filtering to discard non-informative fragments.

Specifically, if the UAV's navigation system includes a flight controller with an autopilot system, the flight mission planning involves defining the trajectory as a sequence of waypoints (coordinates) within the flight zone. Consequently, the UAV's flight path is essentially a polyline consisting of straight-line segments between predefined key points. Given this, processing the entire frame at each step is not necessary.

Each new video frame contains a significant portion of the information present in the previous frames, with only a relatively "narrow" strip of new data appearing due to the UAV's forward movement. Thus, for identifying objects of interest, the classifier model can be fed with a minimal amount of new information, allowing the model's attention to focus exclusively on the dynamic parts of the image. In other words, previously analyzed frame regions remain unchanged or provide minimal new information, ensuring that the model primarily processes only the varying segments of the image.

This optimization reduces the number of classification operations at each step, decreasing overall processing time and power consumption – an essential factor for real-time operation under the limited computational resources of UAVs.

Considering the arguments presented above, the horizontal line traversal method was selected for the experiment, with a line width of 64 pixels. This corresponds to the size of the input images on which the model was previously trained. Such an approach ensures consistency between the input data format and the requirements of the neural network.

All classification time measurements were performed on single-board computers Raspberry Pi 4 Model B and OrangePi 5 Pro, which simulate the constraints of onboard computational resources typical for unmanned aerial vehicle (UAV) systems. These platforms have significant limitations in processor power, RAM, and energy consumption, ensuring a realistic simulation of real-time operations.

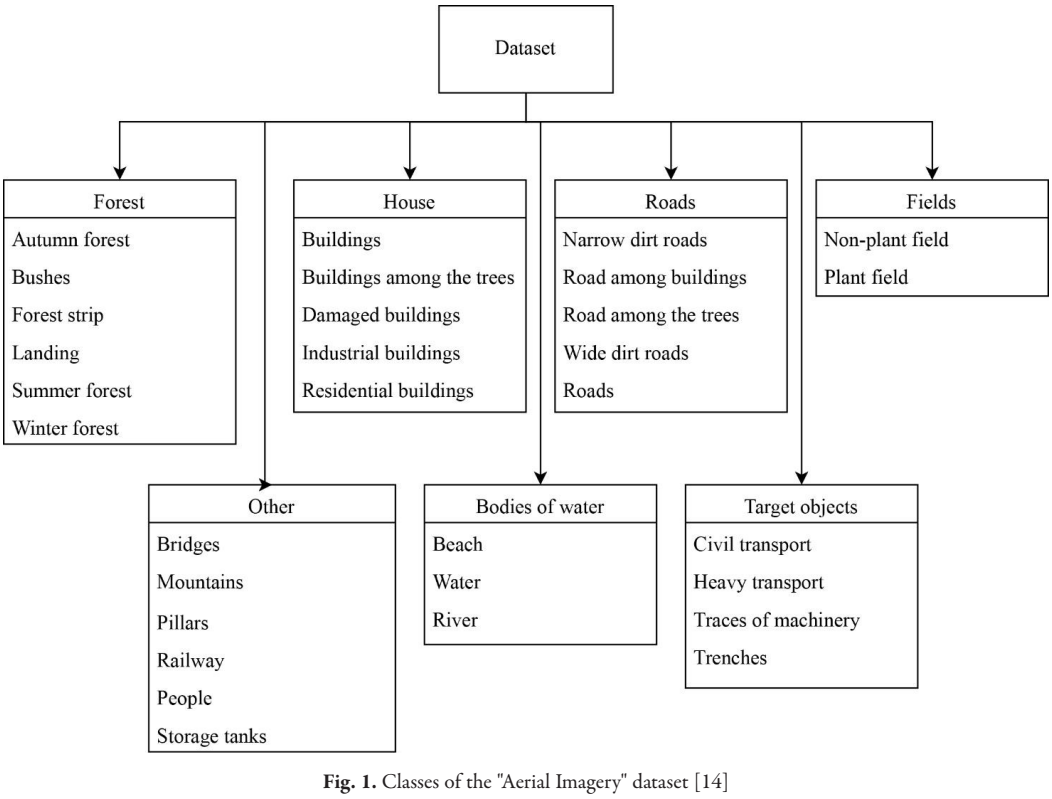


**Fig. 1.** Classes of the "Aerial Imagery" dataset [14]

ISSN-L 2664-9969; E-ISSN 2706-5448

For classification, video with a resolution of 1920×1080 pixels (Full HD) was used. The classification was performed using the horizontal line traversal method with different step sizes: 32 pixels (50 % overlap between adjacent lines), 48 pixels (approximately 67 % overlap), 64 pixels (no overlap).

At each step, image fragments were transmitted to the classifier, grouped into batches ranging from 30 to 240 images. To evaluate performance, the median classification time (in seconds) was calculated for each platform. The use of the median was justified by the fact that the classification time distribution does not follow a normal distribution.

To simulate real-time operation, the Python Threading library was used, enabling the execution of video streams in parallel processes, which improves system efficiency.

Thus, the Aerial Imagery dataset (Fig. 1) was used as training material for classifier model training (Table 2), applied for processing data from UAV onboard cameras in real-time directly on embedded computers. The core of this study is the evaluation of classification speed under different classifier application modes.

## 3. Result and Discussion

The results of the first experiment are illustrated in Tables 3–6.

**Table 3**

Results of Experiment 1 on the Raspberry Pi 4 Model B platform (horizontal traversal)

| Step | Batch | Line traversal | | | Line | | |
|---|---|---|---|---|---|---|---|
| | | DenseNet | Mobile-Netv2 | ResNet | DenseNet | Mobile-Netv2 | ResNet |
| 32 | 30 | 69.723 | 23.046 | 71.56 | 2.123 | 0.697 | 2.109 |
| | 60 | 62.466 | 18.156 | 58.877 | 3.427 | 1.069 | 3.534 |
| | 120 | 56.958 | 15.365 | 53.086 | 6.305 | 1.821 | 5.473 |
| | 240 | 55.362 | 15.459 | 57.273 | 11.713 | 3.342 | 12.182 |
| 48 | 30 | 47.857 | 15.904 | 50.099 | 2.083 | 0.699 | 2.194 |
| | 60 | 41.052 | 12.763 | 43.913 | 3.439 | 1.096 | 3.636 |
| | 120 | 43.719 | 11.317 | 43.304 | 6.357 | 1.875 | 6.586 |
| | 240 | 42.734 | 11.481 | 41.763 | 11.742 | 3.724 | 12.587 |
| 64 | 30 | 34.358 | 13.045 | 36.427 | 2.069 | 0.799 | 2.203 |
| | 60 | 29.9 | 10.234 | 32.441 | 3.447 | 1.221 | 3.657 |
| | 120 | 29.178 | 8.771 | 30.809 | 6.297 | 2.111 | 6.671 |
| | 240 | 23.73 | 7.785 | 24.716 | 11.573 | 3.715 | 12.258 |

**Table 4**

Results of Experiment 1 on Orange Pi 5 pro platform (horizontal traversal)

| Step | Batch | Line traversal | | | Line | | |
|---|---|---|---|---|---|---|---|
| | | DenseNet | Mobile-Netv2 | ResNet | DenseNet | Mobile-Netv2 | ResNet |
| 32 | 30 | 14.684 | 6.936 | 17.36 | 0.452 | 0.214 | 0.534 |
| | 60 | 10.366 | 4.638 | 11.479 | 0.642 | 0.281 | 0.71 |
| | 120 | 9.159 | 3.517 | 9.08 | 1.136 | 0.429 | 1.119 |
| | 240 | 8.616 | 2.989 | 7.859 | 2.12 | 0.728 | 1.951 |
| 48 | 30 | 10.044 | 4.682 | 11.882 | 0.452 | 0.212 | 0.532 |
| | 60 | 7.221 | 3.134 | 7.963 | 0.651 | 0.277 | 0.713 |
| | 120 | 6.088 | 2.322 | 6.237 | 1.136 | 0.398 | 1.12 |
| | 240 | 6.393 | 2.254 | 6.027 | 2.143 | 0.755 | 2.008 |
| 64 | 30 | 7.318 | 3.545 | 8.705 | 0.453 | 0.218 | 0.539 |
| | 60 | 5.255 | 2.354 | 5.75 | 0.65 | 0.287 | 0.712 |
| | 120 | 4.516 | 1.618 | 4.583 | 1.121 | 0.396 | 1.13 |
| | 240 | 4.232 | 1.528 | 3.961 | 2.105 | 0.748 | 1.965 |

As seen from the results of the experiment in Tables 3 and 4, the processing time decreases as the line step increases. This is due to the reduction in the number of image fragments transmitted to the classifier. The fastest classification time is observed for the 64-pixel step. Increasing the batch size leads to longer classification times but optimizes the utilization of computational resources for larger data volumes. However, OrangePi 5 Pro demonstrates significantly better results compared to Raspberry Pi 4 Model B. This is due to its more powerful processor and enhanced RAM capabilities, which allow for better handling of large datasets.

The objective of the second experiment was to evaluate the classification performance when changing the orientation of the analyzed line from horizontal to vertical. This approach allows for an assessment of the impact of the geometric orientation of the input data on system speed under constrained computational resources. As in Experiment 1, classification was performed on video with a resolution of 1920×1080 pixels (Full HD). The line traversal was vertical, and the number of images transmitted to the classifier per batch ranged from 30 to 240. The line step sizes were 32, 48, and 64 pixels.

As seen from the results in Tables 5 and 6, the classification time for the vertical line is generally higher than for the horizontal line across all platforms and models. This result can be explained by the increased number of line traversals required in the vertical orientation due to the frame size (1920×1080 pixels) and the limited batch size.

Regarding the comparison of models, the following conclusions can be drawn.

MobileNetV2 is the fastest model among all tested configurations. It demonstrates the shortest classification time regardless of line step size or batch size. Its lightweight architecture makes it ideal for resource-constrained platforms. It is recommended for tasks where high speed and acceptable accuracy are crucial.

ResNet has the longest classification time among the considered models, indicating its high computational complexity. Its high accuracy makes it useful for tasks where precise recognition is critical, but it requires powerful hardware. It is less suitable for real-time systems on platforms with limited resources.

DenseNet shows an intermediate classification time, balancing between MobileNetV2 and ResNet. It offers reasonable performance for tasks requiring a trade-off between speed and accuracy. It is suitable for applications with moderate computational requirements.

**Table 5**

Results of Experiment 1 on the Raspberry Pi 4 Model B platform (horizontal traversal)

| Step | Batch | Line traversal | | | Line | | |
|---|---|---|---|---|---|---|---|
| | | DenseNet | Mobile-Netv2 | ResNet | DenseNet | Mobile-Netv2 | ResNet |
| 32 | 16 | 20.447 | 7.762 | 19.613 | 0.691 | 0.262 | 0.667 |
| | 32 | 15.315 | 5.295 | 15.029 | 1.014 | 0.351 | 1.0 |
| | 64 | 12.024 | 3.882 | 11.279 | 1.529 | 0.51 | 1.597 |
| | 128 | 10.188 | 3.175 | 10.925 | 2.545 | 0.789 | 2.717 |
| | 256 | 9.62 | 2.851 | 10.451 | 4.799 | 1.407 | 5.205 |
| 48 | 16 | 11.721 | 5.148 | 12.816 | 0.605 | 0.266 | 0.664 |
| | 32 | 8.917 | 3.541 | 10.017 | 0.894 | 0.352 | 1.005 |
| | 64 | 7.537 | 2.586 | 8.002 | 1.504 | 0.512 | 1.596 |
| | 128 | 5.181 | 1.688 | 5.54 | 2.535 | 0.807 | 2.718 |
| | 256 | 4.938 | 1.523 | 5.232 | 4.854 | 1.469 | 5.181 |
| 64 | 16 | 9.433 | 4.483 | 10.05 | 0.614 | 0.295 | 0.66 |
| | 32 | 7.19 | 3.101 | 6.989 | 0.902 | 0.394 | 0.862 |
| | 64 | 6.051 | 2.296 | 5.403 | 1.501 | 0.569 | 1.347 |
| | 128 | 5.134 | 1.776 | 5.448 | 2.551 | 0.875 | 2.709 |
| | 256 | 4.864 | 1.519 | 5.209 | 4.847 | 1.506 | 5.182 |

**Table 6**

Results of Experiment 1 on Orange Pi 5 pro platform (vertical traversal)

| Step | Batch | Line traversal | | | Line | | |
|---|---|---|---|---|---|---|---|
| | | DenseNet | Mobile-Netv2 | ResNet | DenseNet | Mobile-Netv2 | ResNet |
| 32 | 16 | 20.214 | 11.078 | 25.713 | 0.335 | 0.185 | 0.425 |
| | 32 | 13.915 | 6.771 | 17.194 | 0.463 | 0.226 | 0.574 |
| | 64 | 9.999 | 4.504 | 10.911 | 0.661 | 0.294 | 0.722 |
| | 128 | 8.53 | 3.171 | 8.655 | 1.179 | 0.418 | 1.205 |
| | 256 | 9.47 | 3.268 | 8.463 | 2.389 | 0.814 | 2.116 |
| 48 | 16 | 13.484 | 7.412 | 17.228 | 0.337 | 0.187 | 0.431 |
| | 32 | 9.396 | 4.526 | 11.329 | 0.472 | 0.226 | 0.573 |
| | 64 | 6.449 | 3.078 | 7.103 | 0.651 | 0.301 | 0.707 |
| | 128 | 5.778 | 2.283 | 6.135 | 1.152 | 0.446 | 1.218 |
| | 256 | 4.972 | 1.789 | 4.431 | 2.36 | 0.783 | 2.133 |
| 64 | 16 | 10.33 | 5.712 | 12.528 | 0.336 | 0.187 | 0.405 |
| | 32 | 7.063 | 3.413 | 8.248 | 0.464 | 0.223 | 0.543 |
| | 64 | 4.959 | 2.365 | 5.41 | 0.672 | 0.267 | 0.687 |
| | 128 | 4.85 | 1.803 | 4.796 | 1.212 | 0.449 | 1.19 |
| | 256 | 4.645 | 1.657 | 4.173 | 2.317 | 0.815 | 2.069 |

When comparing hardware platforms, the following conclusions were made.

OrangePi 5 Pro demonstrated the highest performance among the tested platforms, with significantly faster classification times compared to Raspberry Pi 4 Model B, especially in complex scenarios (large batch sizes, small line steps). Thanks to its powerful processor and larger RAM capacity, the platform can efficiently process resource-intensive models such as ResNet. This platform is ideal for tasks with high-performance requirements. Its main drawback is the higher cost compared to Raspberry Pi 4 Model B. Using such a platform is justified only when performance is critical and complex models must be deployed. Therefore, OrangePi 5 Pro is recommended for tasks where performance is the key factor, such as equipping high-tech UAVs for reconnaissance or specialized missions.

Compared to OrangePi 5 Pro, Raspberry Pi 4 Model B has a significantly lower cost, making it more accessible for mass production. It can run optimized models such as MobileNetV2 with an acceptable classification time under proper configuration (64-pixel line step, batch size 64–128). Due to its low cost, this platform is an attractive choice for mass deployment, such as for a large fleet of kamikaze UAVs where cost minimization is essential. However, its drawbacks include longer classification times in complex scenarios (large batch size, small line step), which may affect real-time performance. Its limited computational resources make it difficult to use high-complexity models such as ResNet or DenseNet.

Thus, Raspberry Pi 4 Model B is the optimal choice for systems with less strict performance requirements but with a focus on cost efficiency.

It was established that the accuracy of the ResNet50v2 model on target objects was 78 % (0.78), the highest among the models considered. Other models demonstrated slightly lower results for this parameter.

On the test dataset for target objects, the false positive rate was 23 %, while the false negative rate was 22 %. These results indicate the improved effectiveness of the updated models in reducing classification and identification errors.

*Practical significance.* The findings of this study can be applied to enhance the efficiency of real-time aerial imagery processing, particularly in fields requiring rapid image analysis, such as military operations, reconnaissance, search-and-rescue missions, infrastructure inspection, agrotechnology, and logistics. The proposed approaches, including the use of the MobileNetV2 model combined with information filtering and an optimized line traversal algorithm, significantly reduce computational costs and improve the autonomy of UAV onboard systems. Furthermore, these results can be integrated into existing monitoring and data analysis platforms to enhance their performance and adaptability to dynamic conditions.

*Research limitations.* The primary limitations of this study stem from the dependence of the proposed solutions on hardware performance. The use of single-board computers such as Raspberry Pi 4 Model B and OrangePi 5 Pro imposes constraints on the complexity and depth of the models that can be integrated. Practical implementation of the results across different fields may require additional model tuning for specific datasets and operating conditions.

*Future research prospects.* Further research could focus on improving models to enhance their adaptability to dynamic environments, such as changes in scale and perspective of objects. A promising direction is the implementation of modern transformer architectures, which may improve the accuracy of analyzing complex image structures. Additionally, expanding experiments on more powerful platforms would allow for an investigation of the relationship between performance and power consumption. Another important avenue is the integration of AI-based algorithms with UAV flight path optimization technologies, enabling the development of fully autonomous systems for real-time analysis and decision-making.

## 4. Conclusions

Based on the results of the experimental studies, the following recommendations can be provided for using the analyzed models and hardware for processing Full HD video (from cameras with a resolution of 1920×1080) using the line traversal algorithm when searching for target objects.

On Raspberry Pi 4 Model B, it is recommended to use line traversal with the MobileNetV2 model, a line step of 64 pixels, and a batch size of (60; 240) if the UAV is moving at a low speed. In this case, the processing speed will be up to one second. If information filtering of non-informative fragments is applied, the MobileNetV2 model with batch-30 can be used.

On OrangePi 5 Pro, it is recommended to use line traversal with DenseNet121 and ResNet50v2 models, a line step of (32; 64) pixels, and a batch size of (60; 240) if the UAV is moving at a low speed, as well as MobileNetV2 with a line step of (32; 64) pixels and a batch size of (30; 120). If information filtering of non-informative fragments is applied, models with batch sizes of (30; 60) can be used.

The obtained research results demonstrate the possibility of improving the processing speed of aerial imagery under constrained computational resources. The proposed approach, which combines modern neural network models with an optimized line-scanning algorithm, significantly reduces computational load, thereby increasing system autonomy, particularly for UAV onboard systems.

The application of these findings is relevant in fields requiring rapid image analysis, such as military operations, reconnaissance, search-and-rescue missions, critical infrastructure inspection, agrotechnology, and logistics. Additionally, the results can be integrated into modern monitoring and data analysis platforms, enhancing their performance, adaptability to dynamic conditions, and overall efficiency in real-time operation.

## Conflict of Interest

The authors declare that they have no conflicts of interest regarding this study, including financial, personal, authorship-related, or any other factors that could influence the research and its results presented in this paper.

ISSN-L 2664-9969; E-ISSN 2706-5448

## Funding

This research was conducted without financial support.

## Data Availability

This manuscript has no associated data.

## Use of Artificial Intelligence Tools

The authors confirm that no artificial intelligence technologies were used in the creation of this work.

### References

1. Chyrkov, A., Prystavka, P.; Hu, Z., Petoukhov, S., Dychka, I., He, M. (Eds.) (2018). Suspicious Object Search in Airborne Camera Video Stream. *ICCSEEA 2018. Advances in Computer Science for Engineering and Education,* 340–348. https://doi.org/10.1007/978-3-319-91008-6_34
2. Prystavka, P., Shevchenko, A., Rokitianska, I. (2024). Comparative Analysis of Detector-Tracker Architecture for Object Tracking Based on SBC for UAV. 2024 *IEEE 7th International Conference on Actual Problems of Unmanned Aerial Vehicles Development (APUAVD),* 175–178. https://doi.org/10.1109/apuavd64488.2024.10765890
3. Prystavka, P., Chyrkov, A., Sorokopud, V., Kovtun, V. (2019). Automated Complex for Aerial Reconnaissance Tasks in Modern Armed Conflicts. *CEUR Workshop Proceedings, 2588, 57–66.*
4. Prystavka, P., Cholyshkina, O., Dolgikh, S., Karpenko, D. (2020). Automated Object Recognition System based on Convolutional Autoencoder. *2020 10th International Conference on Advanced Computer Information Technologies (ACIT),* 830–833. https://doi.org/10.1109/acit49673.2020.9208945
5. Redmon, J., Farhadi, A. (2017). YOLO9000: Better, Faster, Stronger. 2017 *IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* 6517–6525. https://doi.org/10.1109/cvpr.2017.690
6. Tian, Y., Wang, S., Li, E., Yang, G., Liang, Z., Tan, M. (2023). MD-YOLO: Multi-scale Dense YOLO for small target pest detection. *Computers and Electronics in Agriculture,* 213, 108233. https://doi.org/10.1016/j.compag.2023.108233
7. Fedus, W., Zoph, B., Shazeer, N. (2022). Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity. *Journal of Machine Learning Research, 23.* https://doi.org/10.48550/arXiv.2101.03961
8. Shaw, A., Hunter, D., Landola, F., Sidhu, S. (2019). SqueezeNAS: Fast Neural Architecture Search for Faster Semantic Segmentation. *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW),* 2014–2024. https://doi.org/10.1109/iccvw.2019.00251
9. Wu, B., Wan, A., Iandola, F., Jin, P. H., Keutzer, K. (2017). SqueezeDet: Unified, Small, Low Power Fully Convolutional Neural Networks for Real-Time Object Detection for Autonomous Driving. *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW),* 446–454. https://doi.org/10.1109/cvprw.2017.60
10. Bello, I., Fedus, W., Du, X., Cubuk, E. D., Shlens, J., Zoph, B. et al. (2021). Revisiting ResNets: Improved Training and Scaling Strategies. *Advances in* Neural. *Information Processing Systems, 27*, 22614–22627. https://doi.org/10.48550/arXiv.2103.07579
11. Tomasello, P., Sidhu, S., Shen, A., Moskewicz, M. W., Redmon, N., Joshi, G. et al. (2019). DSCnet: Replicating Lidar Point Clouds With Deep Sensor Cloning. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW),* 1317–1325. https://doi.org/10.1109/cvprw.2019.00171
12. Qin, D., Leichner, C., Delakis, M., Fornoni, M., Luo, S., Yang, F. et al. (2024). MobileNetV4: Universal Models for the Mobile Ecosystem. *Computer Vision – ECCV 2024,* 78–96. https://doi.org/10.1007/978-3-031-73661-2_5
13. *Orange Pi 5 vs Raspberry Pi 4 Model B Rev 1.2.* Available at: https://browser.geekbench.com/v5/cpu/compare/19357188?baseline=19357599&utm_source=chatgpt.com
14. Zivakin, V., Kozachuk, O., Prystavka, P., Cholyshkina, O. (2022). Training set AERIAL SURVEY for Data Recognition Systems From Aerial Surveillance Cameras. *CEUR Workshop Proceedings, 3347,* 246–255.

***Pylyp Prystavka,*** *Doctor of Technical Sciences, Professor, Head of Department of Applied Mathematics, State Non-Profit Enterprise "Kyiv Aviation Institute", Kyiv, Ukraine, ORCID: https://orcid.org/0000-0002-0360-2459*

✉***Olha Cholyshkina,*** *PhD, Associate Professor, Department of Intelligent Systems, Taras Shevchenko National University of Kyiv, Kyiv, Ukraine, ORCID: https://orcid.org/0000-0002-0681-0413, e-mail: olha.cholyshkina@knu.ua*

***Valerii Zivakin,*** *PhD, Senior Lecturer, Department of Applied Mathematics, State Non-Profit Enterprise "Kyiv Aviation Institute", Kyiv, Ukraine, ORCID: https://orcid.org/0000-0002-0420-0558*

***Borys Stetsenko,*** *Department of Applied Mathematics, State Non-Profit Enterprise "Kyiv Aviation Institute", Kyiv, Ukraine, ORCID: https://orcid.org/0009-0007-6588-9532*

✉*Corresponding author*