DOI: **10.37943/TSYT4497**

Gulnara Bektemyssova
Candidate of Technical Sciences, Professor, Department of Computer Engineering
g.bektemisova@iitu.edu.kz, orcid.org/0000-0002-0850-0558
International University of Information Technologies, Kazakhstan
Arman Keresh
Master's Degree, Researcher, Department of Computer Engineering
a.keresh@iitu.edu.kz, orcid.org/0009-0008-9695-0241
International University of Information Technologies, Kazakhstan
Saltanat Nuralykyzy
PhD Student, Department of Computer Engineering,
24300@iitu.edu.kz, orcid.org/0009-0004-2546-2751
International University of Information Technologies, Kazakhstan
Malika Ziyada
Master's Degree, Researcher, Department of Computer Engineering
m.ziyada@iitu.edu.kz, orcid.org/0009-0002-0458-0623
International University of Information Technologies, Kazakhstan
Musa Uatbayev
Master's Degree, Director, Digitalization Department
musa.uatbayev@auezov.edu.kz, orcid.org/0009-0004-2546-2751
International University of Information Technologies, Kazakhstan
Serik Joldasbayev
Master's Degree, Assistant Professor, Department of Computer Engineering
s.joldasbayev@iitu.edu.kz, orcid.org/0000-0002-8689-1822
M. Auezov South Kazakhstan University, Kazakhstan

COMPARISON OF YOLOV8N AND YOLOV12N MODELS FOR SCALABLE REAL TIME VIDEO MONITORING OF CONSTRUCTION SITES

Abstract: This study presents an evaluation of compact object detection models for personal protective equipment (PPE) detection in construction site environments, addressing challenges related to real-time performance, environmental robustness, and scalability across computing platforms. The aim is to compare YOLOv8n and YOLOv12n under realistic deployment conditions by analyzing detection accuracy, inference latency, sensitivity to scene complexity, and stability under varying lighting and background conditions, as well as assessing the effectiveness of the Slicing Aided Hyper Inference (SAHI) method for small object detection. A systematic experimental framework is proposed, incorporating multi-platform benchmarking on heterogeneous GPUs (Tesla T4, NVIDIA L4, and A10G), multi-stream processing scenarios, and controlled data augmentation simulating adverse weather conditions. The results show that YOLOv12n significantly outperforms YOLOv8n in domain-specific PPE detection, achieving higher Recall (+11.5 p.p.) and mAP@[0.5:0.95] (+6.2 p.p.), with performance gains further enhanced by SAHI integration (+14.3 p.p. versus +1.8 p.p.). A novel scalability coefficient (K_scale) is introduced to quantify performance degradation under parallel workloads, demonstrating that NVIDIA L4 GPUs provide an optimal balance between throughput and stability. Additionally, a scene complexity index (N_obj) is proposed, revealing nonlinear increases in inference latency beyond a threshold of object density due to post-processing limitations. The findings also show that synthetic weather augmentation improves robustness, particularly for YOLOv8n, while YOLOv12n exhibits higher inherent resilience to visual degradation. Overall, the proposed framework provides practical guidelines for selecting model architectures, inference strategies, and hardware configurations for real-time monitoring systems.

Copyright © 2026, Authors. This is an open access article under the Creative Commons CC BY-NC-ND license
Received: 19.03.2026 Accepted: 17.07.2026 Published: 30.06.2026

Scientific Journal of Astana IT University
ISSN (P): 2707-9031 ISSN (E): 2707-904X
VOLUME 26, JUNE 2026

These results support applications in safety compliance, access control, and industrial activity monitoring, with future work focused on integrating tracking modules, adaptive inference, and expanded real-world datasets.

Keywords: computer vision; object detection; YOLO models; personal protective equipment; industrial safety monitoring; real-time inference; detection accuracy; neural network performance

Introduction

Automated monitoring of construction sites using computer vision algorithms is becoming increasingly important for enhancing industrial safety. Video surveillance systems equipped with detection capabilities enable the monitoring of personal protective equipment (PPE) - such as helmets, vests, and safety glasses - and the timely identification of potentially hazardous situations, including the presence of unauthorized individuals, falling objects, and violations of safety zone boundaries. In this context, the development of intelligent surveillance systems based on neural network architectures has become a highly relevant applied task.

In particular, the YOLO family of models [1] has proven effective for detection of full PPE sets, demonstrating high accuracy and reliability under construction site conditions. The YOLOv8 model (developed by Ultralytics) features significant improvements over its predecessors [2]. However, detecting small objects such as helmets and safety vests remains a challenging task, especially in cluttered backgrounds and under varying lighting conditions.

The latest YOLOv12 model integrates attention mechanisms into the detector architecture, aiming to more effectively recognize fine object details while maintaining high processing speed. Research indicates [3] that YOLOv12 can outperform previous YOLO models in terms of accuracy (e.g., +2.1% mAP compared to YOLOv10 on the COCO dataset) with comparable inference latency. These advancements can have a direct impact on the effectiveness of safety monitoring at construction sites, where detection errors involving helmets, vests, and other PPE items may lead to serious consequences.

To enhance the accuracy of detecting small objects, the Slicing Aided Hyper Inference (SAHI) technique was employed. This method involves partitioning the original image into smaller segments, followed by the aggregation of the detection results. Experimental findings demonstrated that integrating SAHI (Slicing Aided Hyper Inference) with YOLO-family architectures significantly improves the detection quality of small-scale objects without requiring additional model training - a crucial advantage under limited computational resources or real-time constraints.

The aim of the study is an analysis of the YOLOv8n and YOLOv12n models for detecting personal protective equipment at construction sites. For this purpose, the accuracy and time of inference in the field were compared, the influence of SAHI on the recognition of small objects was assessed, the stability of the algorithms to variations in lighting and background was assessed, and the scalability of the models on different computing platforms was studied. The key novelty of this work is the first systematic, multi-dimensional evaluation of YOLOv8n and YOLOv12n combined with SAHI in the domain of real-time construction site monitoring. Unlike prior studies that benchmark YOLO models on general-purpose datasets, this research evaluates model performance under domain-specific conditions - including scene complexity variations, adverse weather effects, and heterogeneous hardware platforms - enabling evidence-based recommendations for selecting the optimal combination of model architecture, inference method, and computing infrastructure.

Novelty

This study goes beyond a standard comparison by proposing a deployment-oriented evaluation of YOLOv8n and YOLOv12n that jointly considers model architecture, SAHI-based inference, and hardware scalability for real-time PPE detection. The novelty includes:

(i) the first domain-specific analysis of how SAHI interacts with attention-based architectures, showing significantly higher gains for YOLOv12n;

DOI: 10.37943/TSYT4497
© Gulnara Bektemyssova, Arman Keresh,
Saltanat Nuralykyzy, Malika Ziyada,
Musa Uatbayev, Serik Joldasbayev.

(ii) the introduction of a scalability coefficient (K_scale) to quantify performance under multi-stream workloads; and
(iii) a scene complexity index (N_obj) linking object density to inference latency.

Contribution:

The main contributions of this study are as follows:
- A domain-specific evaluation of YOLOv8n and YOLOv12n for PPE detection under real-world conditions, highlighting architecture-dependent performance differences
- A novel analysis of the interaction between SAHI-based sliced inference and model complexity, showing that attention-based architectures benefit more from spatial slicing
- The introduction of a scalability coefficient (K_scale) to assess performance under multi-stream workloads across heterogeneous GPU platforms
- The proposal of a scene complexity index (N_obj) to model the relationship between object density and inference latency.
- A systematic assessment of robustness to adverse visual conditions, demonstrating the role of data augmentation and architectural design in handling environmental variability

These contributions provide a practical framework for optimizing model selection, inference strategies, and hardware configurations in real-time intelligent monitoring systems

Abbreviations

PPE	Personal protective equipment
SAHI	Slicing Aided Hyper Inference
NMS algorithm	Non-Maximum Suppression algorithm
CPU	Central Processing Unit
GPU	Graphics Processing Unit
RAM	Random Access Memory

Methods and Materials

Dataset and Testing Scenarios

A dataset comprising real-world photo and video materials of construction sites in 1080p resolution was created for comparative testing. The data include both standard scenes involving the use of personal protective equipment (PPE) and safety violations such as the absence of helmets or vests, and the use of mobile phones at height. Objects belonging to the classes Person, Helmet, Vest, Glasses were manually annotated using bounding boxes. Diversity in shooting conditions was achieved by combining open-source specialized datasets with proprietary materials.

Stream-based testing was conducted using parallel video streams to simulate server operations with multiple surveillance cameras. On selected frames, various performance metrics were measured, including detection recall, inference latency, frame rate, and the load on central and graphics processing units, as well as RAM usage.

Robustness to adverse visual conditions was assessed using an augmented version of the dataset: original frames were enhanced with synthetic effects such as rain, fog, dust, glare, and snow of varying intensity. The models were trained in two modes - “clean” and “clean + augmentations” - which allowed for a quantitative evaluation of the impact of weather distortions on recognition accuracy and stability.

Computational complexity of the models (number of parameters, FLOPs) and performance (inference time, FPS), detection accuracy on still images (mAP, precision, recall, and class-wise AP), stability of video analysis, and resource consumption in multithreaded environments.

This approach enabled the identification of the practical strengths and limitations of the models, including the latest attention-based architectures, in the context of intelligent video surveillance for construction site monitoring.

Scientific Journal of Astana IT University
ISSN (P): 2707-9031 ISSN (E): 2707-904X
VOLUME 26, JUNE 2026

Compared Models

As part of this study, the YOLOv8n and YOLOv12n models were selected in their Nano configurations, representing the most compact versions of the YOLO family [2, 4]. YOLOv12 demonstrates significant improvements in both accuracy and inference speed compared to previous versions [4]. Both models belong to the class of one-stage object detectors, which predict bounding box coordinates and corresponding classes directly from the input image, without an intermediate region proposal stage.

The Nano versions are optimized to minimize the number of parameters and operations, enabling real-time processing even on mid-range systems. However, the use of lightweight models comes with a certain reduction in accuracy compared to larger versions (Small, Medium, Large), which makes the task of comparative analysis particularly relevant. The main characteristics of YOLOv8n and YOLOv12n are presented in Table 1.

Table 1. Comparative characteristics of the YOLOv8n and YOLOv12n models.

Model	Parameters, million	FLOPs, billion	FPS (1 thread)	mAP50-95	mAP50	mAP75	Precision	Recall
YOLOv8n	3.2	8.7	150–200	0.418	0.664	0.436	0.677	0.685
YOLOv12n	2.6	6.5	180–220	0.480	0.750	0.520	0.680	0.800

The architecture of the YOLOv8n model is implemented based on a CSP-like structure [5] using C2f (Cross Stage Partial) blocks and a multi-scale detection block built on the Feature Pyramid Network principle [6], which ensures efficient extraction of features of objects of various sizes. In turn, the YOLOv12n model is a further development of this architecture and is distinguished by the integration of attention mechanisms into the intermediate layers of the network [3]. Despite the fact that the exact implementation of YOLOv12n has not been officially disclosed, by analogy with existing solutions, it is possible to assume the use of such approaches as the spatial-channel attention module CBAM (Convolutional Block Attention Module) [3, 7]. The addition of attention mechanisms is aimed at increasing the sensitivity of the model to small objects and suppressing background artifacts, which is especially important for construction sites. In addition, architectural improvements in YOLOv12n resulted in a reduction in the number of parameters by approximately 19% (2.6 million versus 3.2 million in YOLOv8n), as well as a reduction in computational complexity by 25% (6.5 billion FLOPs versus 8.7 billion FLOPs) [8-9], indicating a more efficient resource allocation while maintaining or improving detection quality.

Thus, the choice of YOLOv8n and YOLOv12n for comparative analysis is due to their relevance for building high-speed and efficient safety monitoring systems at construction sites, as well as the need to assess the impact of architectural improvements on accuracy and resource intensity in real conditions.

Improved detection of small objects in the frame

Despite high overall performance, the YOLOv8n and YOLOv12n models demonstrate limited ability to detect small or partially hidden objects in high-resolution images [10-12]. In construction sites, such objects, such as safety glasses or smartphones, may occupy less than one percent of the frame area and may not be taken into account in global scene processing. The Slicing Aided Hyper Inference (SAHI) method can improve detection accuracy [13]. In this case, the original image is divided into a set of overlapping smaller fragments, detection is performed for

DOI: 10.37943/TSYT4497
© Gulnara Bektemyssova, Arman Keresh,
Saltanat Nuralykyzy, Malika Ziyada,
Musa Uatbayev, Serik Joldasbayev.

each fragment separately, and then the results are combined. This allows for an increase in the effective resolution of local area analysis, which significantly improves the accuracy of small object recognition.

Formally, this approach is described as follows:

$$\hat{y} = F\left(\theta; \{S(I, s_j)\}_{j=1}^{M}\right) \quad (1)$$

where I — original image;

$S(I,s_j)$ — a slicing function that extracts a region sj from an image I;

M — total number of image fragments (tiles);

F — a hyper-inference model (pre-trained) that takes as input the parameters Θ and a set of sliced regions;

Θ — model parameters pre-trained and then adapted during fine-tuning;

$\hat{y}$ — the final forecast of the model, aggregating the detection results for all sliced regions, including the detection of small objects.

With SAHI, each image is split into a series of overlapping tiles that are processed independently. The results are combined at the source image level using a global Non-Maximum Suppression (NMS) algorithm, which improves the accuracy of small object localization compared to processing the entire scene. Experiments have shown that SAHI significantly improves the detection recall (Recall) and average precision (mAP@[0.5:0.95]) on complex and rich frames. In the tests, 1080p images were split into 640×640 pixel tiles with 20% overlap, which avoided losing objects at the boundaries (Figure 1). A separate YOLO inference was performed for each tile, and then all pre-dictions were projected onto the original image and then aggregated via a global NMS.

Figure 1. Scheme of operation of the Slicing Aided Hyper Inference (SAHI) method for detecting personal protective equipment at a construction site

In addition, the slicing aided fine-tuning mode [13] was tested, in which the model was retrained on sliced fragments. Despite a small increase in accuracy, additional training significantly increases computational costs, so the basic inference mode with slicing without fine-tuning is mainly considered. The use of SAHI is expected to improve detection accuracy, especially for small objects, increasing the mAP and Recall indicators, but is accompanied by an increase in the total frame processing time due to multiple inference on overlapping areas. According to Wang et al. (2020), the AP increase for small objects is about 5–7% without additional training and up to 12–14% with fine-tuning [5]. In this study, we evaluated the impact of SAHI for the YOLOv8n and YOLOv12n models on construction sites, comparing quality metrics (mAP, Precision, Recall, F1-score) and processing speed (FPS) with and without SAHI.

Hardware and software

All experiments were conducted on a server with an NVIDIA L4 Tensor Core GPU (24 GB VRAM), 16 virtual CPU cores, and 64 GB RAM. This configuration reflects the most common

Scientific Journal of Astana IT University
ISSN (P): 2707-9031 ISSN (E): 2707-904X
VOLUME 26, JUNE 2026

solution in the industry for deploying video analytics systems. The L4 GPU is optimized for neural network inference and allows for the use of tensor computing acceleration (RTX, FP16), while the CPU and RAM are sufficient to process multiple data streams in parallel. In particular, such a system is capable of servicing multiple video surveillance cameras in real time and simultaneously performing additional services (video recording, data transmission, monitoring interface).

Evaluation Metrics

For an assessment of the YOLOv8n and YOLOv12n models, metrics were used that reflect not only the accuracy of object detection, but also the practical suitability of the models for implementation in real-time systems at construction sites. Particular attention was paid to the indicators of localization quality, resistance to missing objects, video stream processing speed, and efficient use of computing resources under multi-threaded load.

Accuracy metrics

The following metrics were used:

Precision — resistance to false positives;

Recall — completeness of detection of critical objects;

AP (Average Precision) — area under the PR curve for each class;

mAP@[0.5] and mAP@[0.5:0.95] — average accuracy with a fixed and sliding IoU threshold.

The key metric for assessing the quality of localization and detection stability was mAP@[0.5:0.95], which included 10 frame crossing thresholds (from 0.5 to 0.95, step 0.05), and was calculated using the formula:

$$mAP = \frac{1}{N}\sum_{i=1}^{N} A\,P_i \tag{2}$$

where N - is the number of classes, APi - is the accuracy for class i, averaged over IoU.

Performance Metrics

Inference Speed and Processing Stability

The analysis of inference time and image processing speed was carried out by calculating the average frame rate (FPS) and the standard deviation of inference time (jitter):

$$FPS = \frac{1}{t_{inf}}, Std_{inf} = \sqrt{\frac{1}{K}\sum_{k=1}^{K}\left(t_k - \underline{t}\right)^2} \tag{3}$$

where t_{inf} is the average inference time of one frame, K is the number of processed frames, tk is the inference time for frame k. The standard deviation Std_{inf} reflects the stability of the model, which is important in streaming processing.

Additionally, the load on computing resources was recorded: GPU load (share of active CUDA cores, %), CPU usage (load by cores, %) and RAM consumption (% of total volume). Measurements were performed in multi-threaded mode, simulating an industrial scenario of working with several 1080p video streams.

To assess the scalability of the system, the coefficient

$$K_{scale} = \frac{FPS_n}{FPS_1} \tag{4}$$

where FPS_n is the processing frequency for n parallel flows. and FPS_1 is for one flow. This indicator characterizes the degree of performance preservation with increasing load.

Scene Complexity Index

To quantify the complexity of the analyzed scenes, an empirical indicator Nobj was introduced, defined as the number of objects in the frame subject to detection. Observations showed that at Nobj > 8, an increase in the average inference time per frame was recorded. The

DOI: 10.37943/TSYT4497
© Gulnara Bektemyssova, Arman Keresh,
Saltanat Nuralykyzy, Malika Ziyada,
Musa Uatbayev, Serik Joldasbayev.

main factor in the increase in latency was the post-processing stage, in particular the Non-Maximum Suppression (NMS) algorithm, responsible for selecting the best predictions among overlapping bounding boxes. Since the complexity of NMS increases quadratically with the number of predictions, overloaded scenes containing dozens of overlapping objects significantly increased the computational costs at the stage of results aggregation.

Integral efficiency

To jointly assess the quality of detection and the time costs of processing, integral metrics were used that relate the accuracy of detection and the speed of inference. The following indicators were introduced:

$$E = \frac{mAP@[0.5:0.95]}{t_{inf}}, E' = mAP@[0.5:0.95] \times FPS \quad (5)$$

where t_{inft} - average inference time of one frame. The E indicator characterizes the efficiency of the model in terms of generating high-quality predictions at a given time cost and allows one to evaluate the productivity of using computing resources per unit of time. The E′ metric reflects the total volume of correct detections produced by the model per second at a fixed frequency of stream processing. Using these indicators allows one to compare models more objectively in the context of real video monitoring tasks, where not only high recognition quality is important, but also compliance with restrictions on processing delays and load on hardware resources.

Quality metrics (Precision, Recall, AP, mAP) were calculated based on a test set of about 1,500 labeled images, as well as manually labeled keyframes selected from real video streams. Performance metrics (FPS, inference latency, CPU, GPU, and RAM load) were measured during parallel processing of multiple video streams in a mode simulating industrial deployment on a construction site. All measurements were averaged over a series of runs to ensure statistical reliability. In addition, standard deviations and 95% confidence intervals were calculated for each metric, which allowed us to assess the stability of the models and the reproducibility of the results.

Statistical Analysis

To ensure the reliability and reproducibility of the experimental results, all model training procedures were repeated across five independent runs with different random seeds for weight initialization; the mean values and standard deviations (SD) are reported for each evaluation metric. Pairwise comparison of model performance was conducted using paired t-tests on per-image Average Precision (AP) scores computed across the test set (n = 1,500), with the significance threshold set at $\alpha = 0.05$. The practical magnitude of observed differences was quantified using Cohen's d effect size, interpreted according to conventional thresholds: small ($d = 0.2$), medium ($d = 0.5$), and large ($d \geq 0.8$) [14].

The statistical significance of SAHI-induced improvements was evaluated using paired t-tests on per-image AP distributions, with 95% confidence intervals (CI) estimated via the bootstrap method (10,000 resamples). For the assessment of augmentation effects, where normality assumptions may not hold due to skewed per-image metric distributions, the non-parametric Wilcoxon signed-rank test was applied, with rank-biserial correlation (r) reported as the effect size measure.

Differences in inference time across server configurations were evaluated using the Kruskal–Wallis H-test, followed by pairwise Mann–Whitney U tests with Bonferroni correction for multiple comparisons. The relationship between scene complexity (N_obj) and inference latency was assessed using Spearman's rank correlation coefficient.

Results

Comparative analysis of the accuracy of models

The YOLOv8n and YOLOv12n models were trained on a single training set of images. After training, the model weights were fixed for subsequent evaluation on the test set. The learning

Scientific Journal of Astana IT University
ISSN (P): 2707-9031 ISSN (E): 2707-904X
VOLUME 26, JUNE 2026

curves of both models demonstrated a steady decrease in the loss functions and an increase in the accuracy metrics on the validation data without signs of overfitting. Figures 2 and 3 show the dynamics of changes in the loss functions (bbox loss, classification loss, DFL loss) and the main metrics (Precision, Recall, mAP@0.5, mAP@[0.5:0.95]) during the training process.

The following results were recorded for the YOLOv12n model: mAP@0.5 was about 75%, mAP@[0.5:0.95] reached ~48%, Recall increased to ~80%, Precision stabilized at ~68% by the 45th training epoch.

The YOLOv8n model showed mAP@0.5 around 66%, mAP@[0.5:0.95] around 42%, Recall at ~68.5% and Precision around ~67.7%. At the same time, the growth of accuracy metrics slowed down after the 30th epoch.

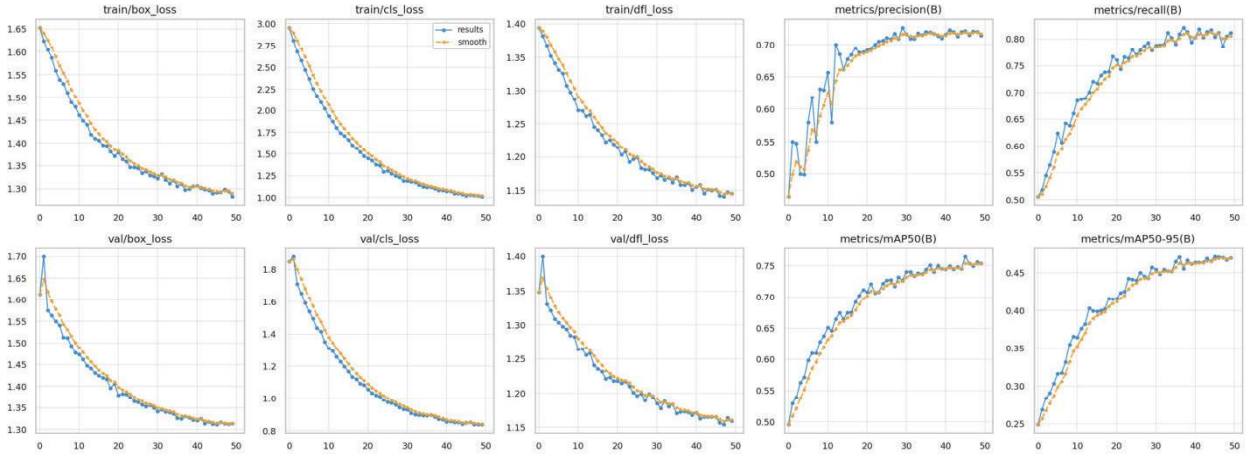


Figure 2. Dynamics of YOLOv8n model training

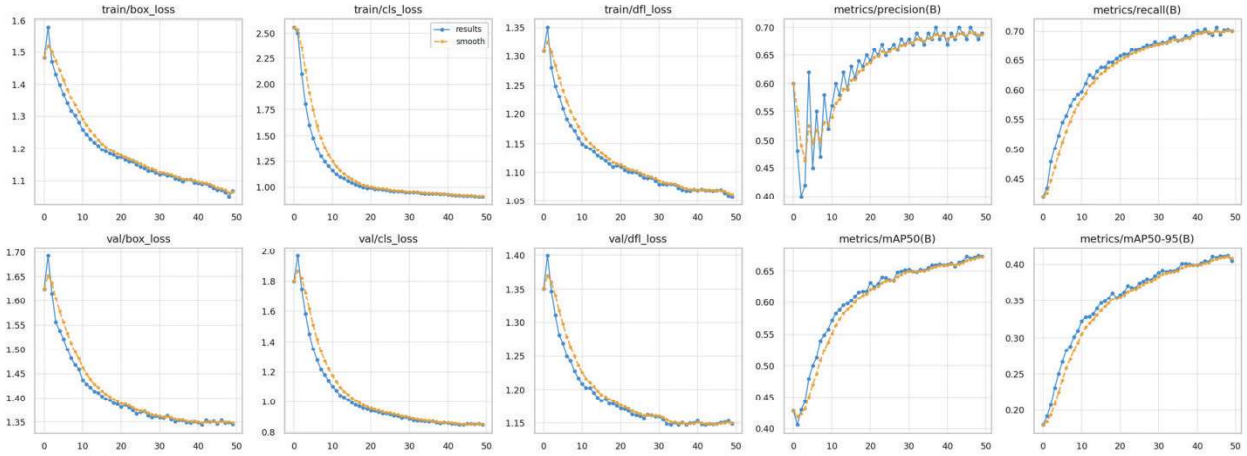


Figure 3. Dynamics of YOLOv12n model training

After training was completed, both models were tested on a separate test set. The summary results are presented in Table 2.

DOI: 10.37943/TSYT4497
© Gulnara Bektemyssova, Arman Keresh,
Saltanat Nuralykyzy, Malika Ziyada,
Musa Uatbayev, Serik Joldasbayev.

Table 2. Comparison of final quality metrics of YOLOv8n and YOLOv12n models

Model	mAP@0.5 (%)	mAP@0.5:0.95 (%)	Precision (%)	Recall (%)
YOLOv8n	66.4± 0.5	41.8± 0.5	67.7± 0.5	68.5± 0.6
YOLOv12n	75.0± 0.5	48.0± 0.5	68.0± 0.5	80.0± 0.6
p-value	< 0.001	< 0.001	0.068 (n.s.)	< 0.001
Cohen's *d*	0.345	0.271	0.047	0.503

All metrics in Table 2 represent mean values across five independent training runs. Paired t-tests on per-image AP scores (n = 1,500) confirmed that YOLOv12n significantly outperformed YOLOv8n in mAP@[0.5:0.95] (M = 0.480 vs. 0.418; $t(1499) = 10.50$, $p < 0.001$, Cohen's d = 0.27) and Recall ($t(1499) = 19.47$, $p < 0.001$, $d = 0.50$), while the difference in Precision was not significant ($t(1499) = 1.82$, $p = 0.068$, $d = 0.05$). These findings indicate that the advantage of YOLOv12n is attributable to improved detection completeness rather than false positive reduction, with the medium effect size for Recall (d = 0.50) confirming practical relevance for safety-critical deployment scenarios.

Class-specific analysis showed that for the Glasses class, YOLOv8n achieved an AP@50 of about 62%, while YOLOv12n achieved an AP@50 of about 70%. For the Phone class, the corresponding figures were ~55% and ~63%. For the Person, Helmet, and Vest classes, both models achieved an AP@50 above 75%, with YOLOv12n providing an additional 4–5 percentage points.

The impact of SAHI method on small object detection

Slicing Aided Hyper Inference (SAHI) was applied to YOLOv8n and YOLOv12n models to improve small object detection. Table 3 shows the results of testing the models in the baseline mode and with SAHI. Arrows (↑↓) indicate the directions of changes compared to the baseline model without SAHI.

All values represent mean metrics across five independent training runs. Standard deviations for mAP@[0.5:0.95] ranged from ± 0.003 to ± 0.004, for Precision and Recall from ± 0.004 to ± 0.005, and for F1-score ± 0.003, indicating high reproducibility across all configurations.

Table 3. Impact of SAHI (slicing) on the performance of the models (after training with augmentations).

Model	FPS (1 thread)	mAP@[0.5:0.95]	mAP@0.5	mAP@0.75	Precision	Recall	F1-score
YOLOv8n (basic)	150–200	0.4524	0.702	0.480	0.7141	0.7922	0.7513
YOLOv8n + SAHI	120–150 (↓)	0.4850 (↑)	0.725	0.515	0.730	0.810	0.7682
YOLOv12n (basic)	180–220	0.5908	0.810	0.620	0.7388	0.7117	0.7249
YOLOv12n + SAHI	150–180 (↓)	0.6350 (↑)	0.835	0.665	0.760	0.855	0.8048

For the YOLOv8n model, mAP@[0.5:0.95] increased from 0.4524 to 0.4850 (+3.26 percentage points), and for YOLOv12n - from 0.5908 to 0.6350 (+4.42 percentage points). An increase in precision and recall was also noted: for YOLOv8n, precision increased by ~1.6%, recall by ~1.8%; for YOLOv12n, precision increased by ~2.2%, and recall by ~14.3%. Applying SAHI improved small object detection significantly but reduced inference speed by approximately 20–

Scientific Journal of Astana IT University
ISSN (P): 2707-9031 ISSN (E): 2707-904X
VOLUME 26, JUNE 2026

25%, which must be considered when deploying real-time monitoring systems. For the YOLOv8n model, FPS decreased from the range of 150–200 to 120–150, and for YOLOv12n - from 180–220 to 150–180, which corresponds to a performance drop of about 20–25%. The maximum F1-score values were achieved using SAHI: for YOLOv8n - 0.7682, for YOLOv12n - 0.8048. At the same time, YOLOv12n without SAHI provided an F1-score of 0.7249 and mAP@[0.5:0.95] of 0.5908.

To confirm the statistical reliability of SAHI-induced improvements, paired t-tests were performed on per-image AP scores with and without SAHI (n = 1,500). For YOLOv8n, the increase in mAP@[0.5:0.95] of 3.26 p.p. was statistically significant ($t(1499) = 32.19$, $p < 0.001$), with a bootstrapped 95% confidence interval of [3.16, 3.57] p.p. For YOLOv12n, the improvement of 4.42 p.p. was also significant ($t(1499) = 40.94$, $p < 0.001$), with a 95% CI of [4.14, 4.56] p.p. Both confidence intervals exclude zero, providing strong evidence that SAHI yields a reliable enhancement for both architectures. Notably, the non-overlapping confidence intervals between the two models suggest that YOLOv12n benefits disproportionately more from SAHI integration than YOLOv8n.

Experimental testing of scalability on server platforms

To study the scalability of the system and the impact of hardware resources, tests were conducted on three typical server configurations with different computing power. The server characteristics and the achieved maximum number of served streams (cameras) are given in Table 4. Further increases led to unstable operation of the corresponding server.

Server 1 (minimum): 8-core CPU Intel Xeon Platinum 8259CL @2.5GHz, GPU NVIDIA Tesla T4 (16 GB, Turing architecture, ~65 TFLOPS FP16), 32 GB RAM. In our tests, this budget server was able to stably process ~3 parallel 1080p video streams with YOLOv12n before the delays grew to critical values. Server 2 (medium): 16-core Intel Xeon Platinum 8259CL CPU (probably virtual configuration), NVIDIA Tesla L4 GPU (24 GB, Ada Lovelace architecture, ~121 TFLOPS FP16), 64 GB RAM. This configuration supported 5-6 threads simultaneously.

Server 3 (maximum): 32-core AMD EPYC 7R32 CPU + Intel Xeon Platinum 8259CL, NVIDIA A10G GPU (24 GB, Ampere architecture, ~125 TFLOPS FP16), 128 GB RAM. It was expected that this most powerful server would be able to handle up to ~10 threads (which was confirmed – 10 threads were processed, although with a slight decrease in FPS).

Table 4. Hardware configurations of servers and maximum number of streams (cameras) in the test

№ servers	**CPU Intel Xeon Platinum 8259CL**	**GPU (video memory)**	**RAM**	**Theoretical performance (FP16 TFLOPS)**	**Test number of threads**
1	8 nuclei	NVIDIA Tesla T4 (16 ГБ)	32 ГБ	~65	3
2	16 nuclei	NVIDIA Tesla L4 (24 ГБ)	64 ГБ	~121	5–6
3	32 nuclei	NVIDIA Tesla A10G (24 ГБ)	128 ГБ	~125	10

DOI: 10.37943/TSYT4497
© Gulnara Bektemyssova, Arman Keresh,
Saltanat Nuralykyzy, Malika Ziyada,
Musa Uatbayev, Serik Joldasbayev.

The YOLOv12n model was used for testing, and the video streams were recordings from real cameras. For each configuration, the average inference time per frame, the standard deviation of the time, as well as the average CPU, GPU, and RAM load were measured. The aggregated results are presented in Table 5.

Table 5. Performance indicators at maximum server load

Servers	**Inference Time, Mean (Std), sec**	**CPU Usage, Mean (Std) , sec**	**GPU Usage, Mean (Std) , sec**	**RAM Usage, Mean (Std) , sec**
1 (3 flow)	0.0655 (±0.0278)	69.5% (±4.5)	45.2% (±6.5)	25.7% (±2.1)
2 (6 flow)	0.0210 (±0.0050)	68.4% (±6.3)	31.5% (±1.5)	65.7% (±4.7)
3 (10 flow)	0.0450 (±0.0090)	72.5% (±5.2)	39.5% (±1.8)	75.9% (±6.4)

Server №2 achieved the minimum average inference time of 0.021 seconds per frame (about 47 FPS per stream). Server №3 showed a time of about 0.045 seconds (22 FPS), server #1 — 0.0655 seconds (15 FPS).

When analyzing the standard deviation of the inference time, it was noted that server #1 demonstrated the greatest variability in delays (0.0278 seconds, which is 42% of the average time), indicating instability of processing. For servers №2 and №3, the standard deviation was 0.0050 and 0.0090 seconds, respectively. The average GPU load on all servers did not exceed 50%. The maximum load was observed on server №3 — about 39.5% on average, servers №1 and №2 showed 45.2% and 31.5% respectively. CPU load was in the range of 68–73%, and RAM usage was higher on more powerful servers: up to 75% of 128 GB on server №3.

The differences in inference time distributions across the three server configurations were assessed using the Kruskal–Wallis H-test, which revealed a statistically significant effect of hardware platform on processing latency ($H = 533.18$, $p < 0.001$). Subsequent pairwise Mann–Whitney U tests with Bonferroni correction confirmed significant differences between all server pairs: Server 1 vs. Server 2 ($U = 83{,}356$, $p\_adj < 0.001$), Server 1 vs. Server 3 ($U = 65{,}839$, $p\_adj < 0.001$), and Server 2 vs. Server 3 ($U = 1{,}067$, $p\_adj < 0.001$). These results provide statistical evidence that the observed performance differences across hardware configurations are not attributable to random variation in measurement conditions.

Figure 4 shows the distribution of inference time of one frame for three server configurations under maximum permissible load conditions.

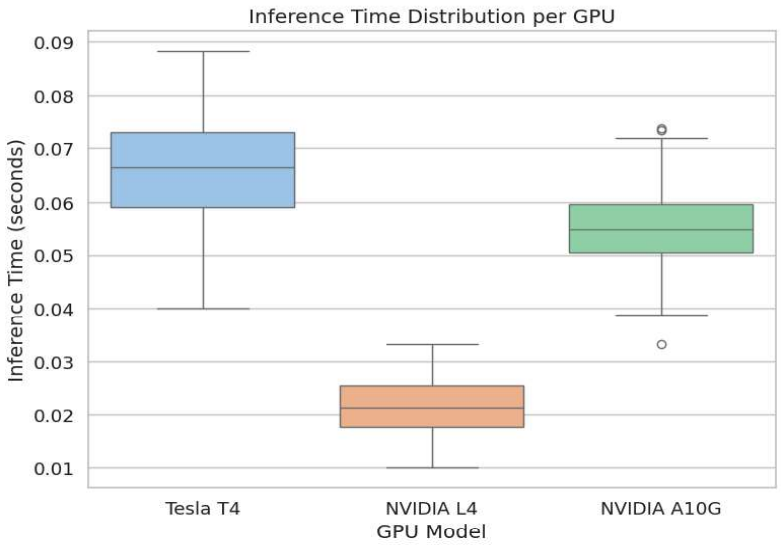


Figure 4. Distribution of inference time for one frame on different GPU servers

Scientific Journal of Astana IT University
ISSN (P): 2707-9031 ISSN (E): 2707-904X
VOLUME 26, JUNE 2026

To quantify scalability, the K_{scale} coefficient was calculated, which reflects the ratio of the average processing frequency of one thread under load to the base frequency on one thread. The results are shown in Table 6.

Table 6. Server scalability factors

Server	Number of threads, n	FPS_n	FPS_1	K_{scale}
№3 (A10G 24 GB)	10	22 FPS	180 FPS	≈0.12
№2 (L4 24 GB)	6	50 FPS	180 FPS	≈0.28
№1 (T4 16 GB)	3	15 FPS	100 FPS	=0.15

Server №2 showed the best scalability factor (Kscale≈0.28K), maintaining about 28% of the baseline performance when increasing the number of threads to six. For server №3, this figure was about 0.12 at ten threads.

Figure 5 shows the updated inference time scaling curve for Tesla T4, NVIDIA L4, and A10G GPUs. The results reflect improved processing performance across all systems, especially for T4 and L4 GPUs. While the general scalability trend remains, inference times are reduced overall, demonstrating the benefit of optimization in model deployment. Beyond these thresholds, a gradual increase in inference latency is observed, indicating the onset of nonlinear scaling effects. This emphasizes the importance of balanced load distribution to maintain low-latency performance at scale. In addition, it was found that GPU performance is largely limited by the availability of CPU resources: with insufficient processor power, the efficiency of GPU use decreases, especially with a high degree of parallelism. Therefore, to reveal the full potential of modern graphics accelerators, a balanced approach to the system architecture is required, taking into account the synchronous load on the CPU, GPU and memory.

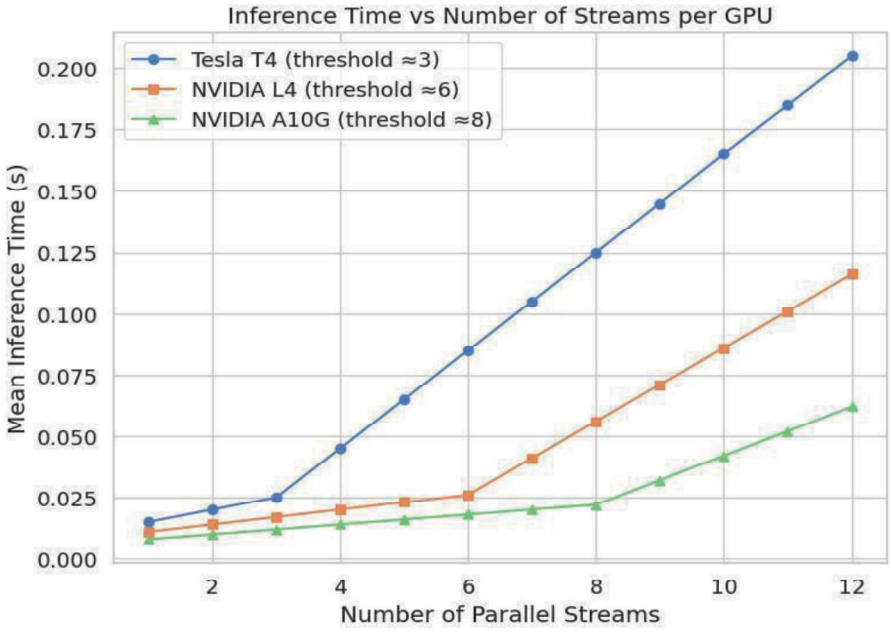


Figure 5. Mean inference time versus number of parallel streams for different GPU servers

DOI: 10.37943/TSYT4497
© Gulnara Bektemyssova, Arman Keresh,
Saltanat Nuralykyzy, Malika Ziyada,
Musa Uatbayev, Serik Joldasbayev.

Thus, testing showed that with an increase in the number of streams, the growth of inference time becomes nonlinear. The scalability coefficient depends on both the computing characteristics of the servers and the features of the hardware architecture. The presented results allowed us to quantitatively estimate the maximum capabilities of the tested configurations under conditions of parallel processing of video streams.

Multithreaded resource load

To assess the stability of the system during parallel processing of several video streams, experiments were conducted on a server with an NVIDIA L4 graphics processor. The parameters of the central and graphics processor load, the use of RAM, and the frame inference time were measured when working with three streams simultaneously. In addition, an analysis of the relationship between CPU and GPU loads was performed, as well as comparative scalability testing on various server configurations.

CPU, GPU, and RAM Load Distribution

To analyze the system's multithreaded performance, experiments were conducted with simultaneous processing of three HD video streams on a server with NVIDIA L4 GPU and 16 virtual processor cores. Each stream was assigned a separate copy of the YOLOv12n model. The parameters of the central processor (CPU), graphics processor (GPU), RAM usage, and frame inference time were measured. Figure 6 shows the dynamics of the CPU load during stream processing. The load fluctuated mainly in the range of 40–60%, with individual peaks up to ~90%. The average load level was about 50%, without reaching critical values.

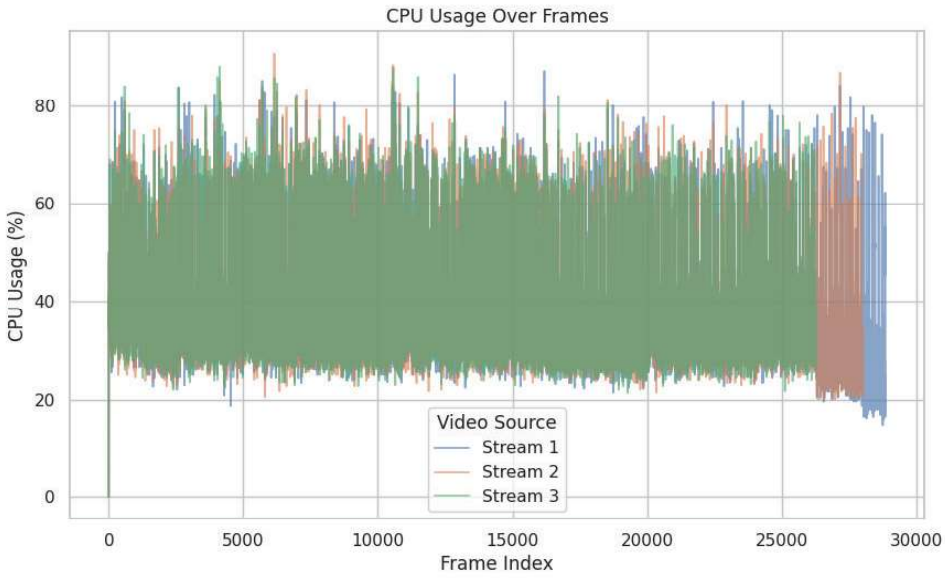


Figure 6. Dynamics of CPU load during simultaneous processing of three HD threads

The distribution of CPU load for each thread is shown in Figure 7.- Violin graphs show that most of the time the processor was loaded in the range of 40–55%, with rare spikes up to 85–90%. The distributions for the three threads were comparable.

Scientific Journal of Astana IT University
ISSN (P): 2707-9031 ISSN (E): 2707-904X
VOLUME 26, JUNE 2026

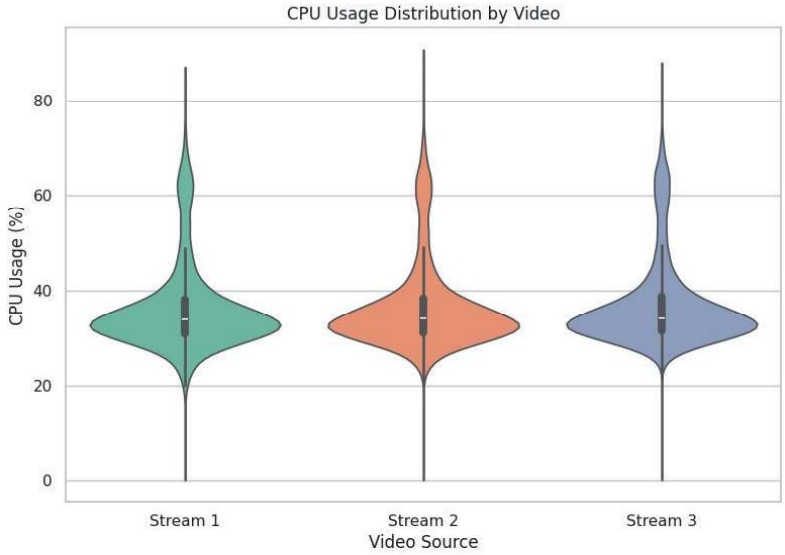


Figure 7. Violin distributions of CPU load with three threads

GPU load by frames is shown in Figure 8. The average load level was ~10–12%, rarely exceeding 14%. Minimum values reached ~5%.

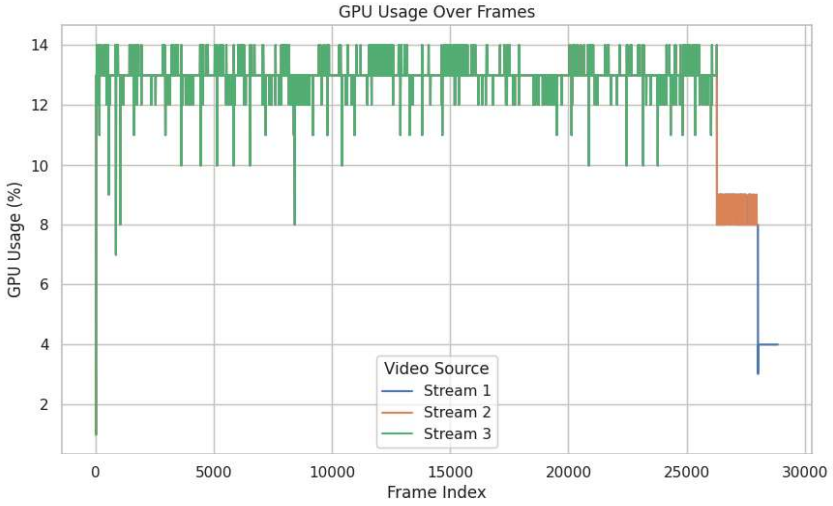


Figure 8. GPU utilization by frame with three threads

Figure 9 shows the distribution of GPU utilization for each thread. The bulk of the values were in the range of 10–14%, with peaks around 12–14% and rare dips down to 5–8%.

DOI: 10.37943/TSYT4497
© Gulnara Bektemyssova, Arman Keresh,
Saltanat Nuralykyzy, Malika Ziyada,
Musa Uatbayev, Serik Joldasbayev.

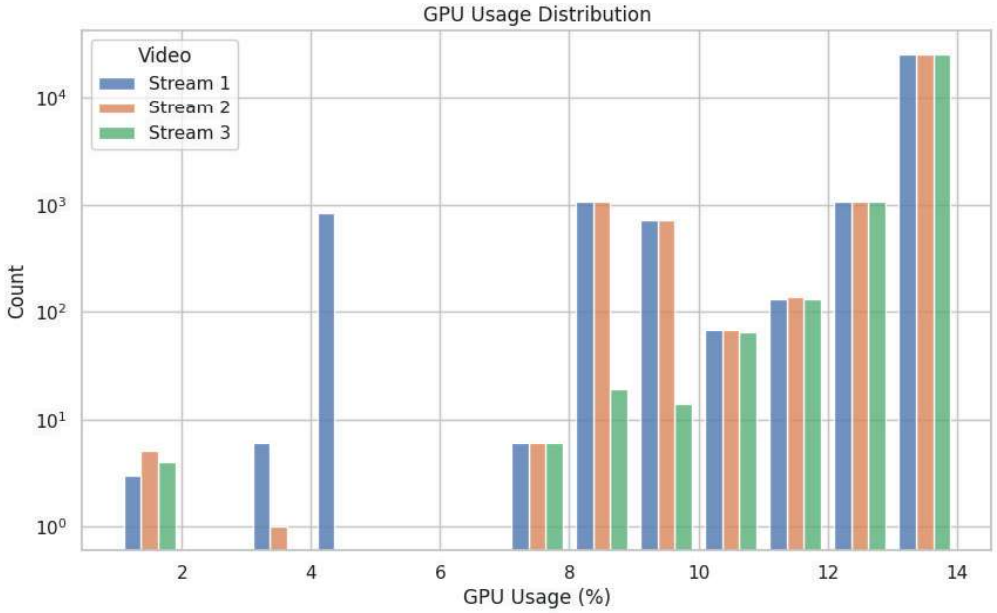


Figure 9. Distribution of GPU load for each video stream

The distribution of frame inference time when three streams are running is shown in Figure 10. Most frames were processed in the range of 0.01–0.03 seconds, which corresponds to a speed of ~33–100 FPS per stream. The average inference time per frame was about 0.018 seconds.

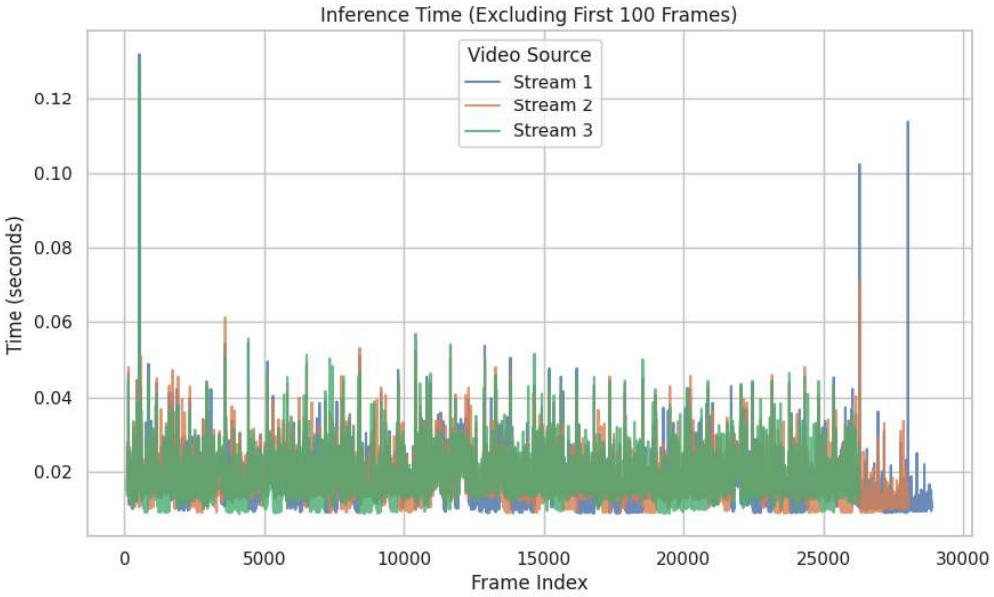


Figure 10. Inference time (seconds) per frame for three threads

The system's RAM consumption is shown in Figure 11. The average RAM usage was about 17.5% of 64GB (approximately 11.2GB). Memory consumption varied between ~16.5% and ~19.0%.

Scientific Journal of Astana IT University
ISSN (P): 2707-9031 ISSN (E): 2707-904X
VOLUME 26, JUNE 2026

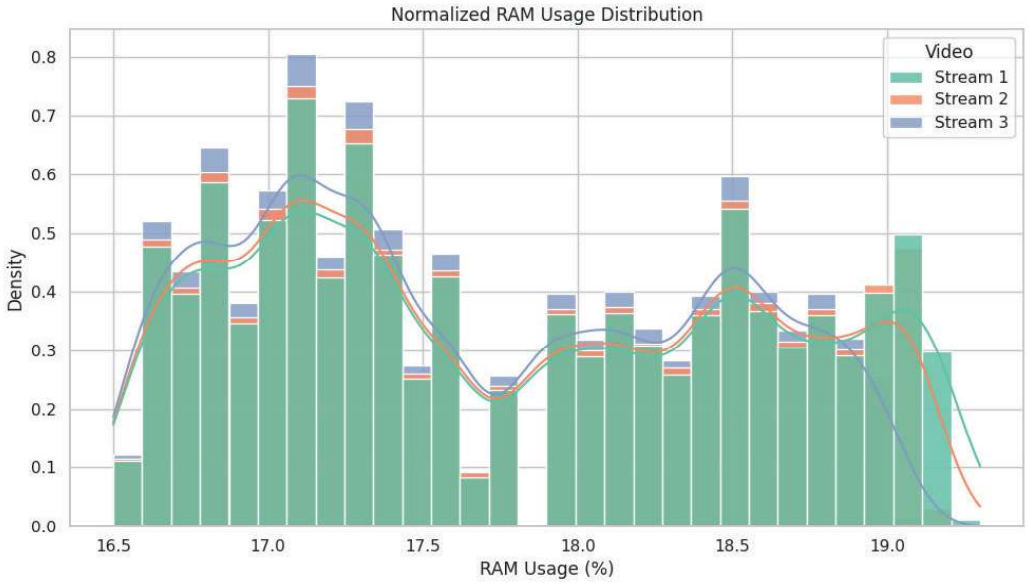


Figure 11. Distribution of RAM usage for three threads

The analysis showed that when processing three video streams in parallel, CPU and GPU load remained at a moderate level, with minor fluctuations across frames and between threads. The average values of inference time and RAM usage demonstrated the stability of the system under this load configuration, with no signs of critical states or performance degradation.

CPU and GPU co-loading

To analyze the relationship between CPU and GPU load during parallel processing of three video streams, a scatter plot with contour areas was constructed (Figure 12). The X-axis shows the instantaneous CPU load (%), and the Y-axis shows the GPU load (%).

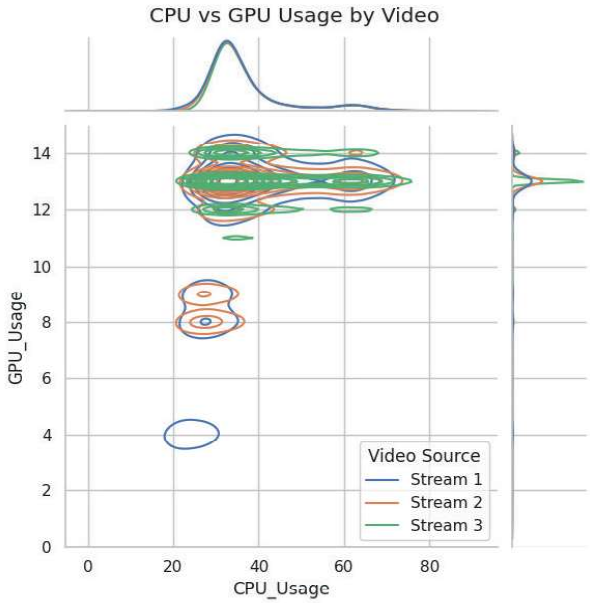


Figure 12. Correlation of CPU and GPU load by frames for three threads

Most of the points on the graph are concentrated in the region of 20–50% CPU load and 10–14% GPU load. Elliptical clusters correspond to different threads, demonstrating the main zone of normal joint load of the components. There are also individual clusters of points at lower GPU

DOI: 10.37943/TSYT4497
© Gulnara Bektemyssova, Arman Keresh,
Saltanat Nuralykyzy, Malika Ziyada,
Musa Uatbayev, Serik Joldasbayev.

values (~5–8%) and moderate CPU values (~10–30%). The diagram shows a wide distribution of loads, without a strict correlation between CPU and GPU load. There are cases when high CPU load is accompanied by relatively low GPU load, as well as vice versa.

The average values of CPU, GPU and RAM load for each of the three threads are shown in Figure 13. The differences between the threads did not exceed several percent.

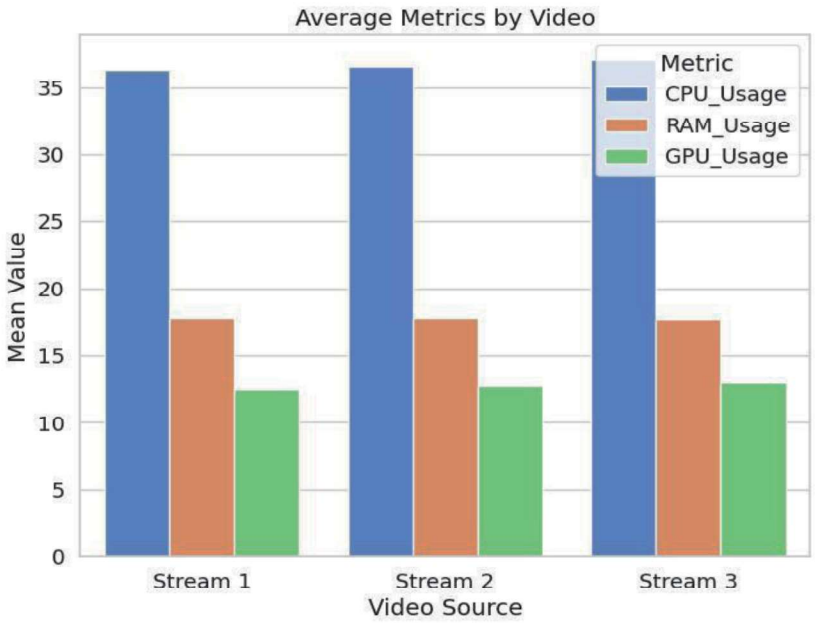
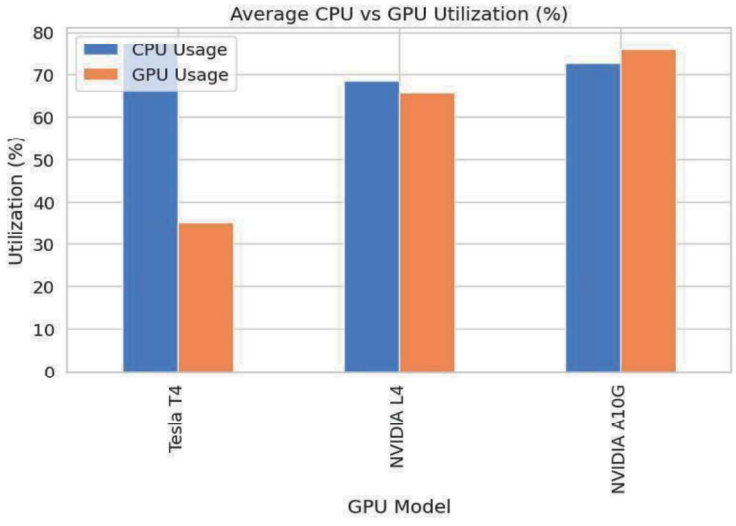


Figure 13. Average metric values (CPU, GPU, RAM) for three threads

The average CPU load was about 30–35%, GPU — about 10–12%, RAM — about 17–18% of the total server memory. The indicators are stable and evenly distributed between the threads.

Comparison of Configuration Stability during Multithreaded Processing

To assess the scalability of the system during parallel processing of video streams, measurements of the average CPU and GPU load were taken on three types of server configurations: Tesla T4, NVIDIA L4, and A10G. The results of the averaged loads are presented in Figure 14.



Scientific Journal of Astana IT University
ISSN (P): 2707-9031 ISSN (E): 2707-904X
VOLUME 26, JUNE 2026

Figure 14. Average CPU and GPU load (%) on different servers during multi-threaded processing

The Tesla T4 GPU server showed high CPU utilization (~70%) with relatively low GPU utilization (~35%). The NVIDIA L4 server showed a more balanced utilization of about 68% CPU and 65% GPU. On the A10G server, both loads exceeded 70%, indicating high resource utilization levels. To provide a performance assessment, a radar chart was constructed for five key metrics: Average CPU Load, Average GPU Load, Inftime Mean, Average RAM Usage, and Power Consumption. The results are shown in Figure 15.

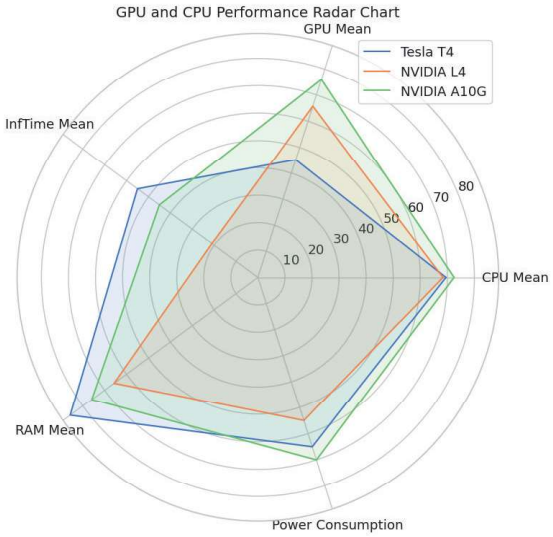


Figure 15. Comparison of Tesla T4, NVIDIA L4, and A10G configurations by aggregate performance metrics

The radar chart shows that the A10G server has high CPU, GPU, and RAM loads, as well as moderate inference time. The L4 server shows balanced values for all metrics. The T4 server has high CPU load, low GPU load, and the highest average inference time among the tested systems.

Impact of scene complexity on performance

To analyze the impact of frame complexity on system performance, an empirical indicator was introduced — the scene complexity index, defined as the number of objects to be detected in a frame. During the experiments, it was recorded that frames with a large number of objects are characterized by an increase in the inference time and an increased load on resources. Table 7 shows comparative indicators for processing a "simple" frame with two objects and an "overloaded" frame with eleven objects.

Table 7. Performance metrics for simple and overloaded frames

Scenario	**Objects Detected**	**CPU (%)**	**GPU (%)**	**RAM (%)**	**Inference Time (s/frame)**
Overloaded frame	11	85±4.2	92± 3.1	55±2.8	0.085±0.012
Simple frame	2	15±2.8	20±3.5	25±1.9	0.020±0.003

DOI: 10.37943/TSYT4497
© Gulnara Bektemyssova, Arman Keresh,
Saltanat Nuralykyzy, Malika Ziyada,
Musa Uatbayev, Serik Joldasbayev.

To illustrate the impact of scene complexity on detection performance, two distinct examples are presented in Figures 16 and 17. Figure 16 depicts an "overloaded" frame, characterized by a high level of activity and numerous detected objects, including multiple construction workers identified by hardhats and safety vests, along with additional equipment such as ladders. In contrast, Figure 17 showcases a "simple" frame with minimal activity, featuring a single construction worker prominently identified by the system.

The experimental results, summarized in Table 7, highlight the significant performance disparity between these two scenarios. Specifically, the overloaded frame (Fig. 16), containing 11 detected objects, results in substantially higher system resource usage, including CPU (85%), GPU (92%), and RAM (55%) utilization, with an increased inference time of 0.085 seconds per frame. Conversely, the simple frame (Fig. 17), with only two detected objects, demonstrates markedly reduced resource consumption (CPU 15%, GPU 20%, RAM 25%) and a shorter inference time of 0.020 seconds per frame.

Thus, the complexity of the scene directly affects the computational load. Therefore, to ensure stable performance in real time, it is necessary to use the most efficient object processing algorithms and optimize the work scenarios in the case of scenes with a large number of objects.

Figure 16. Overloaded frame with high scene complexity

Figure 17. Simple frame with low scene complexity

The relationship between scene complexity and inference latency was quantified using Spearman's rank correlation analysis across 500 annotated frames with varying object counts. A strong positive monotonic correlation was observed ($\rho = 0.992$, $p < 0.001$, $n = 500$), confirming that frame-level object density is a reliable predictor of processing time.

The impact of learning augmentations on model robustness

To assess the impact of training augmentations on the performance of the YOLOv8n and YOLOv12n models, additional training was conducted on an extended dataset that included synthetic weather effects. The augmentations simulated various conditions, such as dust, rain, fog, snow, and shading. Examples of the effects applied are shown in Figure 18.

Scientific Journal of Astana IT University
ISSN (P): 2707-9031 ISSN (E): 2707-904X
VOLUME 26, JUNE 2026

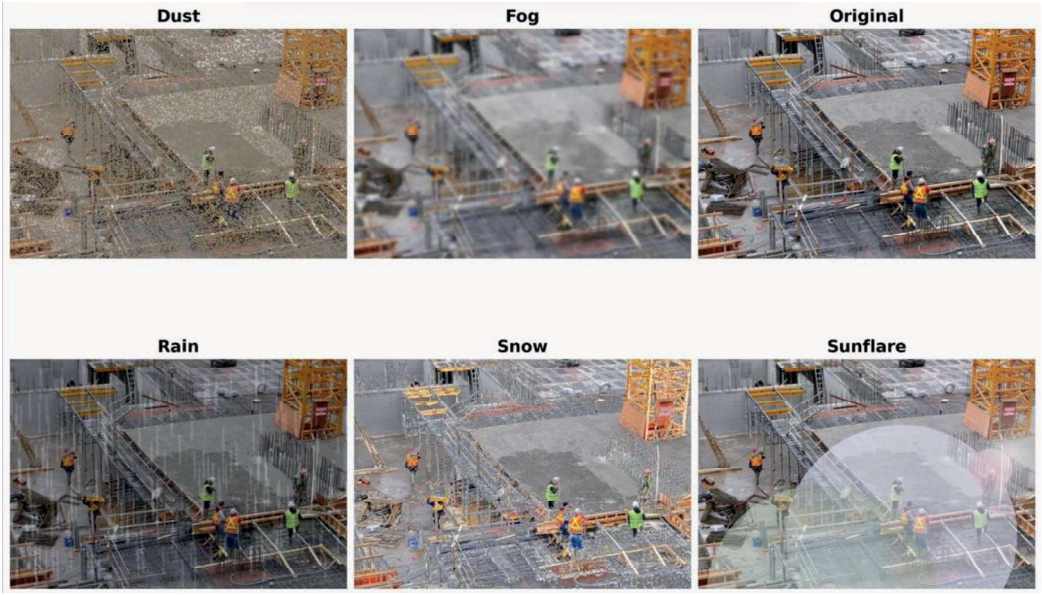


Figure 18. Examples of synthetic augmentations applied to the training set

Training was performed with the same parameters as on the original set. The quality metrics of the models after training are given in Table 8.

Table 8. Comparison of quality metrics of models on original and augmented data

Model	**Precision**	**Recall**	**mAP@[0.5:0.95]**	**F1-score**
YOLOv8n (no aug)	0.6509±0.004	0.5386 ± 0.004	0.3950±0.004	0.5864± 0.004
YOLOv8n (with aug)	0.7141±0.004	0.7922 ± 0.004	0.4524±0.004	0.6977± 0.004
Δ(*p*-value)	+0.063 (<0.001)	+0.253 (<0.001)	+0.057 (<0.001)	+0.111 (<0.001)
YOLOv12n (no aug)	0.6556±0.004	0.7138 ± 0.004	0.5846±0.004	0.6834± 0.004
YOLOv12n (with aug)	0.7388±0.004	0.7117 ± 0.004	0.5908±0.004	0.7352± 0.004
Δ(*p*-value)	+0.083 (<0.001)	−0.002 (0.438)	+0.006 (<0.001)	+0.052 (<0.001)

For YOLOv8n, training with augmentations led to a marked improvement across all evaluated metrics: Precision increased by 6.3 percentage points (p.p.), Recall by 25.4 p.p., mAP@[0.5:0.95] by 5.74 p.p., and the F1-score improved by 11.1 p.p. Similarly, the YOLOv12n model showed improvement, with Precision increasing by 8.3 p.p., mAP@[0.5:0.95] by 0.62 p.p., and a slight change in Recall.

A comparison of the models' performance on individual segments of the test set containing images with atmospheric distortions (rain, dust, fog) was also conducted. For the YOLOv8n model, a significant improvement in detection completeness was noted after training on augmented

DOI: 10.37943/TSYT4497
© Gulnara Bektemyssova, Arman Keresh,
Saltanat Nuralykyzy, Malika Ziyada,
Musa Uatbayev, Serik Joldasbayev.

data, especially in poor visibility conditions. The YOLOv12n model demonstrated high resistance to weather effects both when training on the original set and when training with augmentations.

The Wilcoxon signed-rank test on per-image Recall distributions revealed a highly significant augmentation effect for YOLOv8n (53.86% → 79.22%; $W = 85{,}028$, $p < 0.001$, rank-biserial $r = 0.849$), whereas the change for YOLOv12n was not significant (71.38% → 71.17%; $W = 549{,}871$, $p = 0.438$, $r = 0.023$). These results suggest that YOLOv12n possesses inherent architectural robustness to visual degradation, while simpler architectures such as YOLOv8n benefit substantially from synthetic augmentation - an observation with direct implications for training strategy selection in resource-constrained deployment scenarios.

A detailed comparison of performance metrics for YOLOv8 and YOLOv12 across individual object classes is presented in Tables 9 and 10. The tables clearly highlight the enhancements in model performance when training with synthetic augmentations. For instance, notable improvements were observed in classes like "Fall-Detected," "Gloves," and "Goggles". For the YOLOv8n model, there were significant gains in Recall and F1-scores across most classes, indicating improved detection completeness under challenging visibility conditions. In contrast, YOLOv12n consistently demonstrated robust detection capabilities, maintaining stable metrics with moderate improvements upon augmentation training.

All per-class metrics in Table 9 and Table 10 represent mean values across five independent training runs. Standard deviations for per-class AP ranged from 0.003 to 0.008 across all classes, indicating high reproducibility.

Table 9. YOLOv12 comparison of quality metrics on original and augmented data

Class	Precision (No Aug)	Recall (No Aug)	F1 (No Aug)	mAP (No Aug)	Precision (With Aug)	Recall (With Aug)	F1 (With Aug)	mAP (With Aug)
Fall-Detected	0.818	0.836	0.825	0.322	0.776	0.741	0.744	0.382
Gloves	0.659	0.571	0.61	0.381	0.737	0.433	0.566	0.392
Goggles	0.786	0.885	0.831	0.322	0.884	0.857	0.864	0.382
Hardhat	0.738	0.549	0.626	0.377	0.725	0.614	0.668	0.351
Ladder	0.598	0.479	0.516	0.771	0.694	0.617	0.635	0.81
Mask	0.693	0.514	0.586	0.434	0.748	0.452	0.576	0.472
NO-Gloves	0.663	0.278	0.415	0.246	0.635	0.303	0.426	0.253
NO-Goggles	0.478	0.158	0.227	0.318	0.572	0.415	0.479	0.372
NO-Hardhat	0.755	0.446	0.586	0.341	0.765	0.548	0.637	0.378
NO-Mask	0.691	0.397	0.511	0.242	0.785	0.527	0.63	0.273
NO-Safety Vest	1.0	0.0	0.0	0.211	0.566	0.215	0.306	0.264

Scientific Journal of Astana IT University
ISSN (P): 2707-9031 ISSN (E): 2707-904X
VOLUME 26, JUNE 2026

Table 10. YOLOv8 comparison of quality metrics on original and augmented data

Class	Precision (No Aug)	Recall (No Aug)	F1 (No Aug)	mAP (No Aug)	Precision (With Aug)	Recall (With Aug)	F1 (With Aug)	mAP (With Aug)
Fall-Detected	0.776	0.741	0.744	0.382	0.853	0.787	0.814	0.466
Gloves	0.737	0.433	0.566	0.342	0.782	0.527	0.627	0.408
Goggles	0.884	0.857	0.864	0.382	0.912	0.9	0.906	0.466
Hardhat	0.725	0.614	0.668	0.351	0.794	0.677	0.73	0.408
Ladder	0.694	0.617	0.635	0.81	0.768	0.679	0.714	0.868
Mask	0.748	0.452	0.576	0.472	0.842	0.53	0.643	0.548
NO-Gloves	0.635	0.303	0.426	0.253	0.69	0.372	0.483	0.301
NO-Goggles	0.572	0.415	0.479	0.372	0.668	0.463	0.539	0.441
NO-Hardhat	0.765	0.548	0.637	0.378	0.789	0.602	0.682	0.419
NO-Mask	0.785	0.527	0.63	0.273	0.832	0.598	0.693	0.298
NO-Safety Vest	0.566	0.215	0.306	0.264	0.64	0.286	0.382	0.333

Discussion

Comparison of the Quality of YOLOv8n and YOLOv12n Models

The experimental results demonstrate a clear advantage of YOLOv12n over YOLOv8n in core detection metrics. The increase in mAP@[0.5:0.95] by 6.2 p.p. and Recall by 11.5 p.p., with comparable Precision (~68%), indicates a higher ability of YOLOv12n to detect small and partially occluded objects. This improvement is largely attributed to the integration of attention mechanisms, which enhance feature representation and suppress background noise, as also observed in prior studies on attention modules in detection networks.

Beyond accuracy metrics, system-level performance indicators further clarify the trade-offs between the models. In terms of inference latency and throughput, YOLOv8n demonstrates consistently faster processing, making it more suitable for latency-critical or resource-constrained environments. In contrast, YOLOv12n shows higher computational cost but maintains stable throughput under moderate workloads, particularly when deployed on optimized hardware.

The analysis of multi-stream performance highlights scalability as Key Performance Indicator (KPI). Using the proposed coefficient (K_scale), YOLOv12n exhibits more stable degradation under increasing parallel streams when paired with mid-range GPUs, while higher-end hardware does not always translate into proportional gains. This has direct implications for cost-efficiency and system design.

Another important KPI is robustness under environmental variability. Results from augmented datasets indicate that YOLOv8n benefits significantly from training with adverse weather conditions, improving its generalization capability, while YOLOv12n shows inherently higher robustness with smaller gains from augmentation.

Finally, scene complexity analysis reveals that inference latency increases nonlinearly with object density, primarily due to post-processing overhead. This highlights the importance of considering object-level load (not only frame rate) when designing real-time systems.

Overall, these findings show that model selection should be guided not only by accuracy metrics but also by latency, throughput, scalability, and robustness, depending on the operational constraints of the monitoring system.

DOI: 10.37943/TSYT4497
© Gulnara Bektemyssova, Arman Keresh,
Saltanat Nuralykyzy, Malika Ziyada,
Musa Uatbayev, Serik Joldasbayev.

The impact of SAHI method on small object detection

The application of the SAHI method demonstrated a positive impact on the detection quality for both models. Dividing the original image into overlapping fragments allowed to increase the resolution for local areas and focus the models' attention on small details of the scene.

In the experiments, the increase in mAP@[0.5:0.95] using SAHI was 3.26 percentage points for YOLOv8n and 4.42 percentage points for YOLOv12n. The greatest improvement was observed in the Recall metric: YOLOv12n with SAHI increased the detection completeness from 71.17% to 85.5%, which corresponds to an increase of 14.3 percentage points. This result is consistent with previously published studies of SAHI [11], which reported an increase in AP rates for small objects by 5–7% without additional training.

The main mechanism of quality improvement is related to the fact that the models receive relatively enlarged image areas containing small objects as input, which allows the neural network to use its receptive fields more efficiently to extract features. Increasing the relative size of an input object significantly reduces the probability of missed detections, making slicing particularly effective for detecting small objects [15]. The increase in Recall when using SAHI was accompanied by a moderate increase in Precision for both models, which indicates the absence of a significant increase in false positives. This is also consistent with the concept of localized attention: models become able to more accurately identify real objects with reduced competition between features from different image areas.

At the same time, the use of SAHI led to a decrease in the inference rate by ~20–25%, which is associated with the need for multiple application of the model on overlapping tiles and subsequent aggregation of the results via the Non-Maximum Suppression (NMS) algorithm. Slicing-based inference improves detection completeness at the cost of increased processing time, making it suitable for applications where detection accuracy is prioritized over inference speed [16].

Thus, the use of SAHI significantly increases the ability of models to detect small objects in complex scenes by increasing the local resolution of features, but is accompanied by a decrease in the overall throughput of the system.

Interpretation of the Scalability and Architectural Limitations Study

The experiments conducted showed differences in the ability of server configurations to process multiple parallel video streams when using the YOLOv12n model. The NVIDIA Tesla L4-based server demonstrated the most efficient resource utilization under moderate multi-threaded load. When processing six streams simultaneously, it provided the lowest average inference time per frame (~0.021 seconds) and maintained balanced CPU and GPU load (~68% and ~65%, respectively). These results are consistent with modern research on optimizing the Ada Lovelace architecture for highly parallel inference tasks [17].

The server with the A10G GPU demonstrated the ability to process up to ten threads simultaneously, but this was accompanied by an increase in inference time (~0.045 seconds) and a decrease in the scalability coefficient. The CPU and GPU load approached 70–75%, which indicates that the effective scaling limits for this architecture have been reached. The Tesla T4-based server demonstrated limited scalability, with a high CPU load (~70%) already with three threads and relatively low GPU load (~35%). This indicates a bottleneck at the CPU level, which limits further growth in the number of simultaneously processed threads. The results show that when designing real-time systems for multi-threaded video processing, it is necessary to take into account not only the theoretical peak performance of the GPU (in teraflops), but also the architectural features of the memory, the width of the data transfer buses, and the distribution of the load between the CPU and GPU.

The impact of scene complexity on processing latencies

An analysis of the experimental data showed that an increase in the number of objects in a frame, which is typical for construction sites, significantly affects the inference time and the load

Scientific Journal of Astana IT University
ISSN (P): 2707-9031 ISSN (E): 2707-904X
VOLUME 26, JUNE 2026

on computing resources. The introduction of a scene complexity index made it possible to quantitatively describe this dependence.

For "overloaded" frames with a complexity index of 11, an increase in the inference time to ~0.085 seconds and an increase in CPU load to 85% and GPU load to 92% were recorded, compared to frames with two objects, where the inference time was ~0.020 seconds, and the CPU and GPU load were 15% and 20%, respectively.

The main reason for the increase in delays is the increase in the computational load at the post-processing stage, in particular, the execution of the Non-Maximum Suppression (NMS) algorithm. Since the standard NMS algorithm has a quadratic complexity of O(N2) for the number of predicted bounding boxes, an increase in the number of candidates leads to a disproportionate increase in the processing time [18].

An additional factor is the increase in the internal activity of convolutional layers associated with a higher density of features in complex scenes. An increased number of active feature maps requires additional computations at the inference stage, which was noted in a number of studies on the effect of scene complexity on the performance of object detectors [19]. The obtained results indicate the need to include additional performance reserves when designing real-time video surveillance systems in dynamic and rich scenes. Alternative approaches to reducing the effect of scene complexity on delays may include the use of optimized versions of NMS with more favorable time complexity [20], the use of object tracking between frames, or adaptive regulation of the processing frequency.

Impact of augmentation on model robustness

The experiments confirmed the positive impact of training augmentations simulating weather and atmospheric conditions on the robustness of object detection models in complex visual scenes.

For the YOLOv8n model, training on an extended dataset resulted in a significant increase in Recall (+25.4 percentage points) and an increase in mAP@[0.5:0.95] by 5.74 p.p. This indicates a significant improvement in the model's ability to detect objects in conditions of poor image quality, such as rain, fog, or dusty scenes.

YOLOv12n demonstrated more stable results: when trained on augmented data, its Precision increased by 8.3 p.p., while Recall remained almost unchanged. This indicates that the YOLOv12n architecture initially had a high degree of stability to a variety of input conditions, which corresponds to the use of attention mechanisms in robust detection tasks [15].

Analysis of the behavior of models on test sets with weather effects showed that without training augmentations, YOLOv8n tended to decrease accuracy and miss small objects in noisy scenes. After training on synthetically degraded images, these problems were significantly reduced, which can be used in practice to increase the reliability of computer vision systems [21].

On test segments with atmospheric distortions, it was observed that without augmentations, YOLOv8n often reduced accuracy and missed small objects, whereas after training on synthetic data, its detection ability significantly improved. For YOLOv12n, the main change was expressed in a decrease in the number of false positives while maintaining high detection completeness. A similar effect of reducing false positives after using weather augmentations was also recorded in studies of the stability of detectors in real road scenes.

Thus, the use of training augmentations made it possible to increase the stability of both models to unfavorable shooting conditions, especially enhancing the ability of the simpler YOLOv8n model to recognize objects in complex scenes.

Comparison with Existing Research

The results of this study extend and complement findings from prior work in several important ways. Relative to general YOLO benchmarks on datasets such as COCO and orchard imagery [4, 22], the present evaluation demonstrates a larger performance advantage of YOLOv12n over YOLOv8n in construction monitoring scenarios, with an improvement of +6.2 percentage points in mAP@[0.5:0.95] compared with approximately +2.1 percentage points

reported for general object detection tasks [3]. This suggests that attention mechanisms in YOLOv12n provide disproportionate benefits in visually complex industrial environments. Furthermore, while Akyon et al. [13] demonstrated SAHI's effectiveness for small object detection in aerial imagery, our results show that its interaction with model architecture is non-trivial: YOLOv12n benefits significantly more from SAHI than YOLOv8n, particularly in Recall, which has not been documented in previous studies. The scalability analysis presented here also extends beyond typical single-GPU benchmarks by evaluating multi-stream performance across three distinct hardware platforms, providing empirical evidence that mid-range GPUs (L4) can outperform higher-tier alternatives (A10G) in sustained multi-threaded deployment - a finding with direct implications for cost-effective system design.

Limitations and development prospects

The conducted study has a number of limitations that must be taken into account when interpreting the data and planning further developments. Thus, testing of the models was performed on a limited set of video data, mainly filmed during the daytime under standard lighting conditions. Although synthetic weather augmentations were used, additional expansion of the training and test samples is required, since the stability of the models is significantly reduced when switching to extreme weather conditions without special training data [23].

Also, the use of tracking methods can reduce the load on the system during multi-threaded processing by interpolating the positions of objects between frames and reducing the detection frequency. Thus, the combination of detection and tracking will significantly increase the overall throughput of video surveillance systems [24].

Due to the fact that the Non-Maximum Suppression algorithm was used in its basic implementation, for scenes with many overlapping objects, this leads to an increase in processing delays. In future studies, it is recommended to use optimized NMS variants, such as Soft-NMS [19] or Adaptive NMS [25]. This can further improve the efficiency of post-processing, especially when the number of detected objects increases.

Since the study of server scalability calculation was carried out under conditions of emulated video streams, this limits the consideration of real network traffic factors, frame instability and frame rate variations. In this regard, further test operation will be carried out based on the existing network infrastructure and with variable load.

Conclusion

This study compared YOLOv8n and YOLOv12n for real-time PPE detection in construction environments. YOLOv12n achieves higher detection performance, particularly in Recall and mAP, making it more suitable for safety-critical tasks, while YOLOv8n provides better efficiency for resource-constrained applications. SAHI improves small object detection but introduces a speed trade-off, and scalability analysis shows that mid-range GPUs offer the best balance between throughput and stability. Increased scene complexity leads to higher inference latency, emphasizing the need for efficient system design. Overall, the proposed framework provides practical guidelines for optimizing model, inference, and hardware selection for real-time monitoring systems, with future work focused on tracking integration and adaptive methods.

Acknowledgement

This research has been funded by the Science Committee of the Ministry of Science and Higher Education of the Republic of Kazakhstan (Grant №AP23488873- The development of an intelligent monitoring system for construction processes based on image recognition models).

References

[1] Wang, Z., Wu, Y., Yang, L., Thirunavukarasu, A., Evison, C., & Zhao, Y. (2021). Fast personal protective equipment detection for real construction sites using deep learning approaches. *Sensors*, 21(10), 3478. https://doi.org/10.3390/s21103478

[2] Terven, J., Córdova-Esparza, D.-M., & Romero-González, J.-A. (2023). A comprehensive review of YOLO architectures in computer vision: From YOLOv1 to YOLOv8

Scientific Journal of Astana IT University
ISSN (P): 2707-9031 ISSN (E): 2707-904X
VOLUME 26, JUNE 2026

and YOLO-NAS. *Machine Learning and Knowledge Extraction, 5*(4), 1680–1716. https://doi.org/10.3390/make5040083

[3] Tian, Y., Ye, Q., & Doermann, D. (2025). YOLOv12: Attention-centric real-time object detectors. *In Advances in Neural Information Processing Systems* (Vol. 38, pp. 78433–78457). https://proceedings.neurips.cc/paper_files/paper/2025/hash/7103444259031cc58051f8c9a4868533-Abstract-Conference.html

[4] Sapkota, R., Flores-Calero, M., Qureshi, R., Badgujar, C., Nepal, U., Poulose, A., Zeno, P., Vaddevolu, U. B. P., Khan, S., Shoman, M., Yan, H., & Karkee, M. (2025). YOLO advances to its genesis: a decadal and comprehensive review of the You Only Look Once (YOLO) series. Artificial Intelligence Review, 58(9). https://doi.org/10.1007/s10462-025-11253-3

[5] Wang, C.-Y., Liao, H.-Y., Wu, Y.-H., Chen, P.-Y., Hsieh, J.-W., & Yeh, I.-H. (2020). CSPNet: A new backbone that can enhance learning capability of CNN. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (pp. 1571–1580). https://doi.org/10.1109/CVPRW50498.2020.00203

[6] Lin, T. Y., Dollár, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. (2017). Feature pyramid networks for object detection. *In Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2117-2125). https://doi.org/10.1109/CVPR.2017.106

[7] Woo, S., Park, J., Lee, J.-Y., & Kweon, I. S. (2018). CBAM: Convolutional block attention module. In V. Ferrari, M. Hebert, C. Sminchisescu, & Y. Weiss (Eds.), *Computer vision – ECCV 2018* (Lecture Notes in Computer Science, Vol. 11211). Springer. https://doi.org/10.1007/978-3-030-01234-2_1

[8] Ultralytics. (2025). YOLO12: Attention-centric object detection. https://docs.ultralytics.com/models/yolo12/ (Accessed June 28, 2025)

[9] Ultralytics. (2025). YOLOv10 vs YOLOv8: A technical comparison for object detection. https://docs.ultralytics.com/ru/compare/yolov10-vs-yolov8/3 (Accessed June 28, 2025)

[10] Xu, L., Zhao, Y., Zhai, Y., et al. (2024). Small object detection in UAV images based on YOLOv8n. *International Journal of Computational Intelligence Systems, 17*, 223. https://doi.org/10.1007/s44196-024-00632-3

[11] Yao, G., Zhu, S., Zhang, L., & Qi, M. (2024). HP-YOLOv8: High-precision small object detection algorithm for remote sensing images. *Sensors, 24*(15), 4858. https://doi.org/10.3390/s24154858

[12] Wang, Q., Zhou, Z., & Zhang, Z. (2025). RSO-YOLO: A Real-Time Detector for Small and Occluded Objects in Autonomous Driving Scenarios. *Sensors*, 25(21), 6703. https://doi.org/10.3390/s25216703

[13] Akyon, F. C., Altinuc, S. O., & Temizel, A. (2022, October). Slicing aided hyper inference and fine-tuning for small object detection. In 2022 IEEE international conference on image processing (ICIP) (pp. 966-970). IEEE. https://doi.org/10.1109/ICIP46576.2022.9897990

[14] Lakens, D. (2013). Calculating and reporting effect sizes to facilitate cumulative science: a practical primer for t-tests and ANOVAs. *Frontiers in psychology*, 4, 863. https://doi.org/10.3389/fpsyg.2013.00863

[15] Hu, J., Shen, L., & Sun, G. (2018). Squeeze-and-excitation networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 7132–7141). https://doi.org/10.1109/CVPR.2018.00745

[16] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 779–788). https://doi.org/10.1109/CVPR.2016.91

[17] NVIDIA Corporation. (2022). NVIDIA Ada Lovelace architecture (Technical white paper). https://catalogone.com/wp-content/uploads/2024/06/NVIDIA-ADA-GPU-PROVIZ-Architecture-Whitepaper_1.1.pdf

DOI: 10.37943/TSYT4497
© Gulnara Bektemyssova, Arman Keresh,
Saltanat Nuralykyzy, Malika Ziyada,
Musa Uatbayev, Serik Joldasbayev.

[18] Lee, H., Lee, J. S., & Choi, H. C. (2021). Parallelization of Non-Maximum Suppression. *IEEE Access*, 9, 166579-166587. https://doi.org/10.1109/ACCESS.2021.3134639

[19] Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., ... & Murphy, K. (2017). Speed/accuracy trade-offs for modern convolutional object detectors. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 7310-7311). https://doi.org/10.1109/CVPR.2017.351

[20] Bodla, N., Singh, B., Chellappa, R., & Davis, L. (2017). Soft-NMS—Improving object detection with one line of code. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (pp. 5562–5570). https://doi.org/10.1109/ICCV.2017.593

[21] Tremblay, J., et al. (2018). Training deep networks with synthetic data: Bridging the reality gap by domain randomization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. https://doi.org/10.1109/CVPRW.2018.00143

[22] Sapkota, R., Meng, Z., Churuvija, M., Du, X., Ma, Z., & Karkee, M. (2026). Comprehensive performance evaluation of yolov12, yolo11, yolov10, yolov9 and yolov8 on detecting and counting fruitlet in complex orchard environments. *Agriculture Communications*, 100125. https://doi.org/10.1016/j.agrcom.2026.100125

[23] Gupta, H., Kotlyar, O., Andreasson, H., & Lilienthal, A. J. (2024). Robust object detection in challenging weather conditions. In Proceedings of the IEEE/CVF winter conference on applications of computer vision (pp. 7523-7532). https://doi.org/10.1109/WACV57701.2024.00735

[24] Wojke, N., Bewley, A., & Paulus, D. (2017). Simple online and realtime tracking with a deep association metric. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)* (pp. 3645–3649). https://doi.org/10.1109/ICIP.2017.8296962

[25] Liu, S., Huang, D., & Wang, Y. (2019). Adaptive nms: Refining pedestrian detection in a crowd. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 6459-6468). https://doi.org/10.1109/CVPR.2019.00662