

EMPIRICAL ANALYSIS OF PARALLEL DATA ACQUISITION, STORAGE ARCHITECTURES, AND AI-DRIVEN CLASSIFICATION IN E-COMMERCE SYSTEMS

Ioan-David Pîrvan^{1*} 

¹ Faculty of Mechanical and Electrical Engineering, Petroleum-Gas University of Ploiesti, Romania

* Email (corresponding author): ioan.pirvan@student.upg-ploiesti.ro

DOI: 10.51865/JPGT.2026.02.02

ABSTRACT

In the context of evolving E-Commerce architectures and the continuous growth of internet data, cloud environments frequently encounter resource management challenges. This paper conducts a comparative analysis of image data storage methods, evaluating server-side storage with database references against Binary Large Object (BLOB) database persistence. To facilitate this study, an e-commerce simulation platform was developed, providing a controlled environment for empirical testing. Furthermore, the study explores an image classification model capable of automatically grouping PC peripherals. Such imagery is essential for machine learning (ML) services, where storage strategies directly impact performance. The developed AI application achieved a Micro-Accuracy metric of 0.8274, trained on a subset of 1,672 peripheral images sourced from the Kaggle PC Parts dataset. Regarding web scraping, this research evaluates the trade-offs between execution latency and resource consumption across varying levels of concurrency. Experimental results demonstrate that while parallel execution significantly outperforms the sequential baseline, it introduces a non-linear increase in RAM usage. A performance analysis reveals that a 3-thread configuration represents the most optimal balance, achieving a 2.27x speedup and a 56.01% reduction in operational runtime, while avoiding the efficiency degradation and memory overhead observed at higher concurrency levels. Additionally, image loading speed tests conducted over 20 iterations yielded an average retrieval time of 374.95 ms for the server-side method, compared to 600.10 ms for the BLOB alternative. The findings confirm that server-side reference storage is more reliable and resource-efficient for e-commerce platforms. The research objectives included integrating multithreading capabilities into ingestion scripts to optimize throughput and data processing efficiency, executing an empirical case study to evaluate performance differentials, latency, and resource metrics between binary database storage and local server-side hosting, as well as training a ML application to classify peripherals based on input images.

Keywords: image classification, Machine Learning, data ingestion pipeline, data storage efficiency, Selenium web scraping, parallel processing

INTRODUCTION

Modern applications use images whose quality raises issues associated with large storage volumes. The increase in the volume of multimedia data has necessitated improvements in storage methods and, implicitly, in image management. One of the most used technical

approaches is the storage of binary objects in Binary Large Object (BLOB) format in relational databases. The paper [1] notes that this storage method allows for easy information management, especially when the data is sensitive, such as in the case of medical information and associated documents. Also in the field of medical records, research has proposed various modular software solutions that include dedicated modules for BLOB storage and image processing. Such systems integrate clinical data with images and generate additional functionalities for analysis and automated processing [2]. Also in this medical field, oncological research infrastructures use databases for storing computed tomography images [3]. This paper [4] discusses the need for multimedia information management mechanisms when the data volume is extensive.

Using machine learning (ML) tools requires training on large volumes of data. For these reasons, the method of storing them is a current topic. The paper [5] discusses the issue of images used in ML, taking into account the necessity of a detailed level of content rendering. In such situations, reducing image quality is not a viable solution, which is why storage must be done without affecting quality. Another issue specific to images stored in databases is associated with their integration into artificial intelligence (AI) tools, when training is conducted, raising ethical concerns [6]. Research [7] is conducted on the SQL Server FileStream technology and shows that unstructured data, such as images, videos, and multimedia documents, raises issues related to performance, backup, restoration, and transactional consistency. Similarly, research [8] extends databases by storing PDF documents as BLOBs.

The paper [8] notes that such a task requires infrastructures that ensure access and long-term preservation of the information [9]. Also in the medical field, BLOBs are used in retinal imaging for retinal blood vessel segmentation [10], for the detection of microaneurysms [11], in cardiac segmentation [12], in pupil segmentation [13], or in dermatology for melanoma classification [14].

Performance issues for storing binary data drive the development of alternative solutions. Storing images in databases must be approached with caution, considering potential cyberattacks. Research [15] has shown that ML models can prevent over 96% of SQL Injection attacks. In the paper [16], BlobSeer is presented as a system dedicated to managing large binary objects, which involves the existence of a sharing mechanism and concurrent access to data. In parallel, research aimed at information retrieval from databases shows that storing and accessing large volumes of data requires additional optimizations to reduce computational costs and the space occupied on servers. The paper [17] reports on a parallelization and cryptography technique called ShufflePIR, which increases the throughput of access operations on large databases [18].

With the increase in data volume, researchers have paid increased attention to distributed systems, alongside Cloud architecture capable of managing images and multimedia files. The paper [19] presents an example of a system developed for managing very large satellite images [20]. This system uses PostgreSQL along with circular cache mechanisms, prefetching, multithread processing, and specific optimization, retrieval, and image display mechanisms. The conclusion of the research [19] shows that the use of parallel processing and distributed storage reduces latency by improving the user experience on the platforms. The medical field remains a benchmark for images that require the highest fidelity in rendering reality.

Cloud computing-based platforms are used in systems intended for diagnostics through distributed file systems. The authors of the paper [21] report a reduction in costs without affecting the speed of image processing. Additionally, the reduction of data redundancy is analyzed in the paper [22], where the system under review employs deduplication techniques for virtual machine images. Thus, a reduction in the occupied space of approximately 80% is achieved. The transfer time is reported in paper [22] with a 30% reduction. Similar results are reported in the paper [23], where researchers use the Interplanetary File System and distributed protocols for image storage. In this paper [23], file fragmentation and distribution are carried out across multiple nodes, which reduces costs. Distributed systems are also used in studies where the complexity of managing large-sized images requires manipulation and access mechanisms to avoid performance degradation [24].

The development of AI technologies has generated numerous research studies in the specialized literature on automatic image-level classification and analysis [25]. In the field of historical documents, a system combined BLOB storage with Deep Learning (DL) for the classification of modern Japanese characters. In this case, the obtained accuracy was 99.75% [26]. The paper [27] applies automatic fruit detection, also using BLOB storage techniques, trained with DL models. In the field of text analysis, classifiers such as Text Blob, Valence Aware Dictionary and sEntiment Reasoner (VADER), Naive Bayes, and LSTM were evaluated, achieving an accuracy of 90.64% [28].

The objective of the paper is to evaluate the methods of acquisition, storage, display, and classification of image data in the context of e-commerce platforms. From a technical standpoint, the paper conducts a comparative analysis between BLOB storage and server storage. The performance indicators target latency and resource consumption. In this regard, the paper aims to determine the optimal parallelization configuration for web scraping processes, compare the performance between storing images in BLOB format versus referencing them via server file paths, and validate an automatic classification model for PC peripherals based on ML techniques.

The main contributions of the paper include:

- The development of an e-commerce simulation platform that served as an environment for empirical testing;
- The implementation of a web scraping script with support for parallel execution, through which it was determined that a three-thread configuration represents the optimal compromise between latency and RAM consumption;
- The conduct of a comparative study on the loading times of images stored as BLOBs versus those referenced via server paths;
- The training and validation of an image classification model based on ML, using the ML.NET framework.

The paper is organized into several sections, the first being associated with the introduction, which presents a synthesis of the specialized literature. The Materials and Methods section describes the dataset used, the technologies and software architectures adopted, as well as the testing methodology. The section Performance Analysis of Data Acquisition Methods presents the experimental results regarding sequential execution versus parallel execution in the web scraping process. The section Comparative Analysis of Image Storage Architectures evaluates the differences in the proposed performance indicators. The section Development and

Validation of the Image Classification Model details the training process and the results obtained by the CNN classifier. Finally, the Conclusions section concludes the paper by presenting the main findings and proposing future research directions.

MATERIALS AND METHODS

For the training of the Image Classification application, it was used a public database extracted from Kaggle [29] comprising multiple classes, totaling at 3,279 images, at a resolution of 256x256 pixels, in color and JPG format. However, only the subsets that contained peripheral devices were selected, consisting of 7 classes with an aggregated count of 1,672 images. The choosing of this dataset was made due to the pre-processed nature of the images, rendering them suitable for the evaluation of the aimed peripherals.

The NuGet package Selenium.WebDriver (v4.43.0) was utilized for the web scraping module for the e-commerce data extraction and to automate the performance testing for the image storage study.

The development of the web application simulating an e-commerce store was implemented using the ASP.NET Core MVC (Model-View-Controller) architecture and the database population was done via data retrieval from two online platforms using the web scraping technique.

The selection of the C# programming language and .NET Framework for the creation of the scripts was motivated by their native compatibility with Microsoft SQL Server, which was used for storing the resources that will be used during the latency assessment between server hosted images and those stored within the database in BLOB format.

The integration of the web application and the database was made through Entity Framework, utilized as the Object-Relational Mapper, that facilitates the data communication and schema mapping between the two resources.

The data acquisition process for the database population presented the first temporal constraint between the retrieval methodology. As it is presented in the flowchart in Figure 1, this study will look into both the parallel and sequential data execution variants looking for the best execution time in regards of resource usage. The script was made to extract 10 sample products, with 5 products being selected from each of the selected e-commerce platforms.

The latency testing application assesses the performance of image retrieval within the e-commerce simulation, comparing the database retrieval time of images stored as BLOBs against those referenced by their respective file paths on the server. This testing is performed through a workflow comprising iterative performance measurements.

As depicted in the workflow from Figure 2, there are two separate calls made to the testing routine, one for server-side paths and one for BLOB storage. The variables serverSum, minTimeS and maxTimeS are passed during the testing of the server site storage, while blobSum, minTimeB and maxTimeB are passed for the testing of the image that is save fully within the database. It's critical to note that these variables are passed by reference within the code to allow the testing routine to modify them directly.

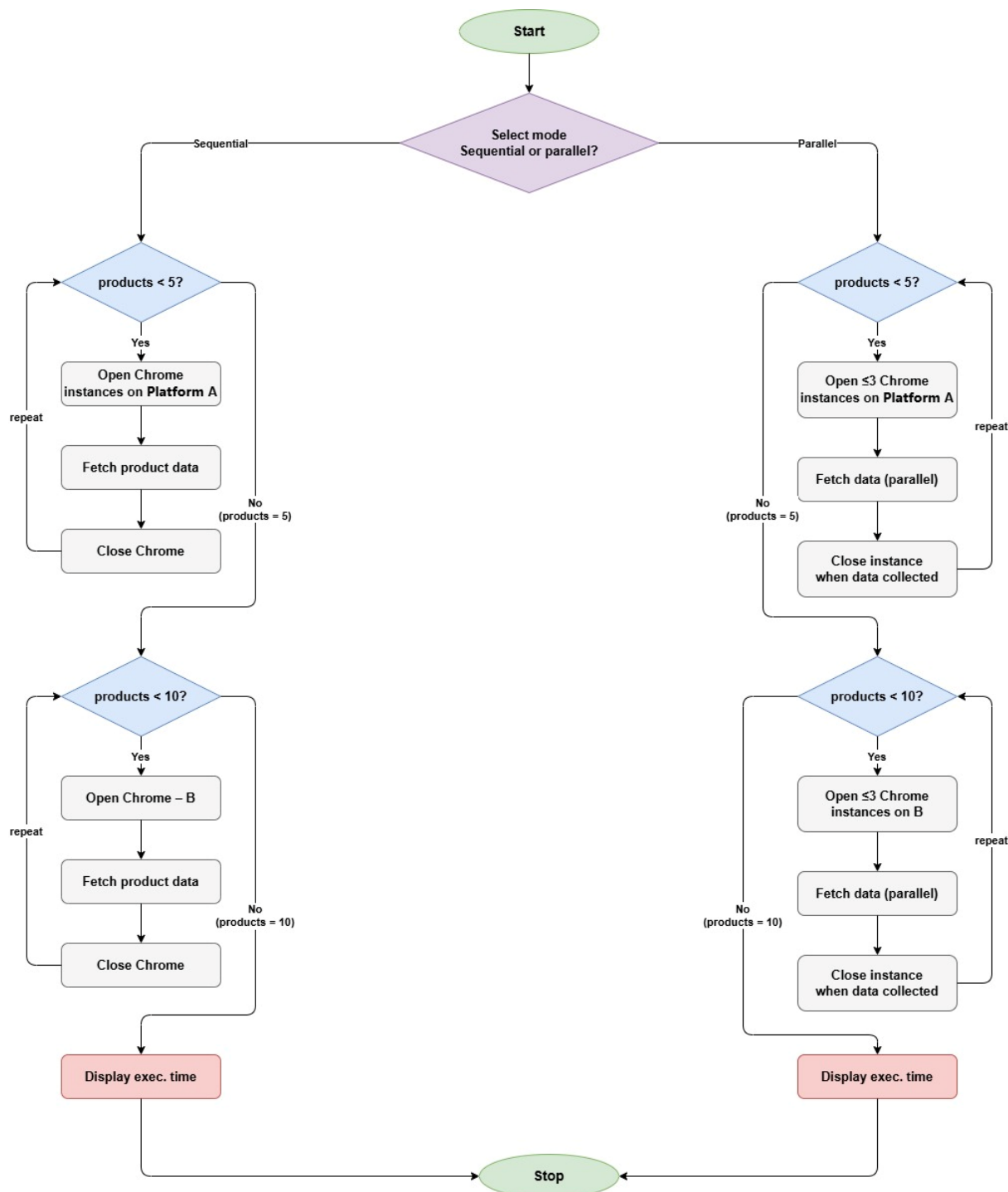


Figure 1. Logical flow for the data acquisition process: Sequential versus parallel execution.

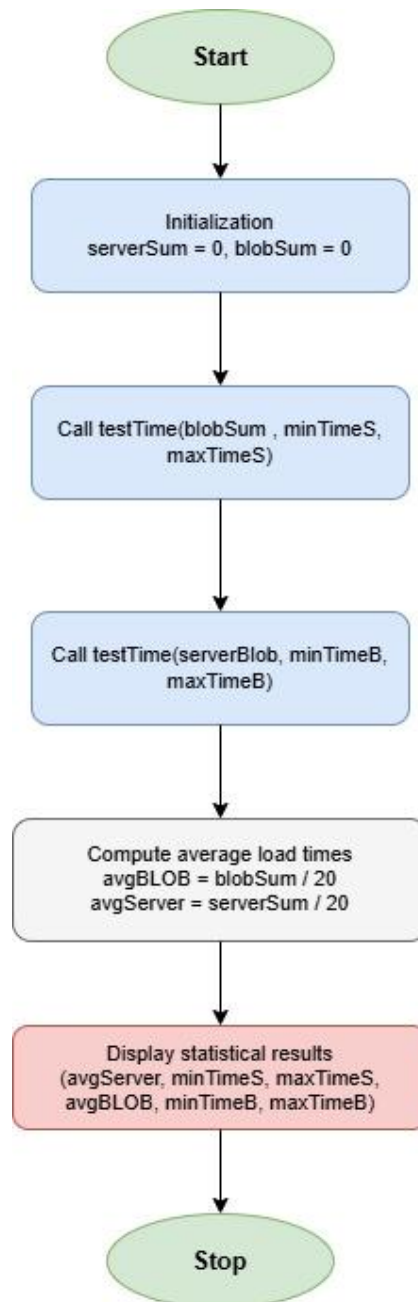


Figure 2. Logical flow of the latency assessment process.

Following the execution of the function `timeTest(...)`, the detailed logical flow of which is illustrated in Figure 3, all of the six variables will hold their respective updated values. This is achieved by measuring the image load duration with a Stopwatch instance, while iteratively launching a Chrome instance in incognito mode 20 times to perform the necessary summations and identify the minimum and maximum values.

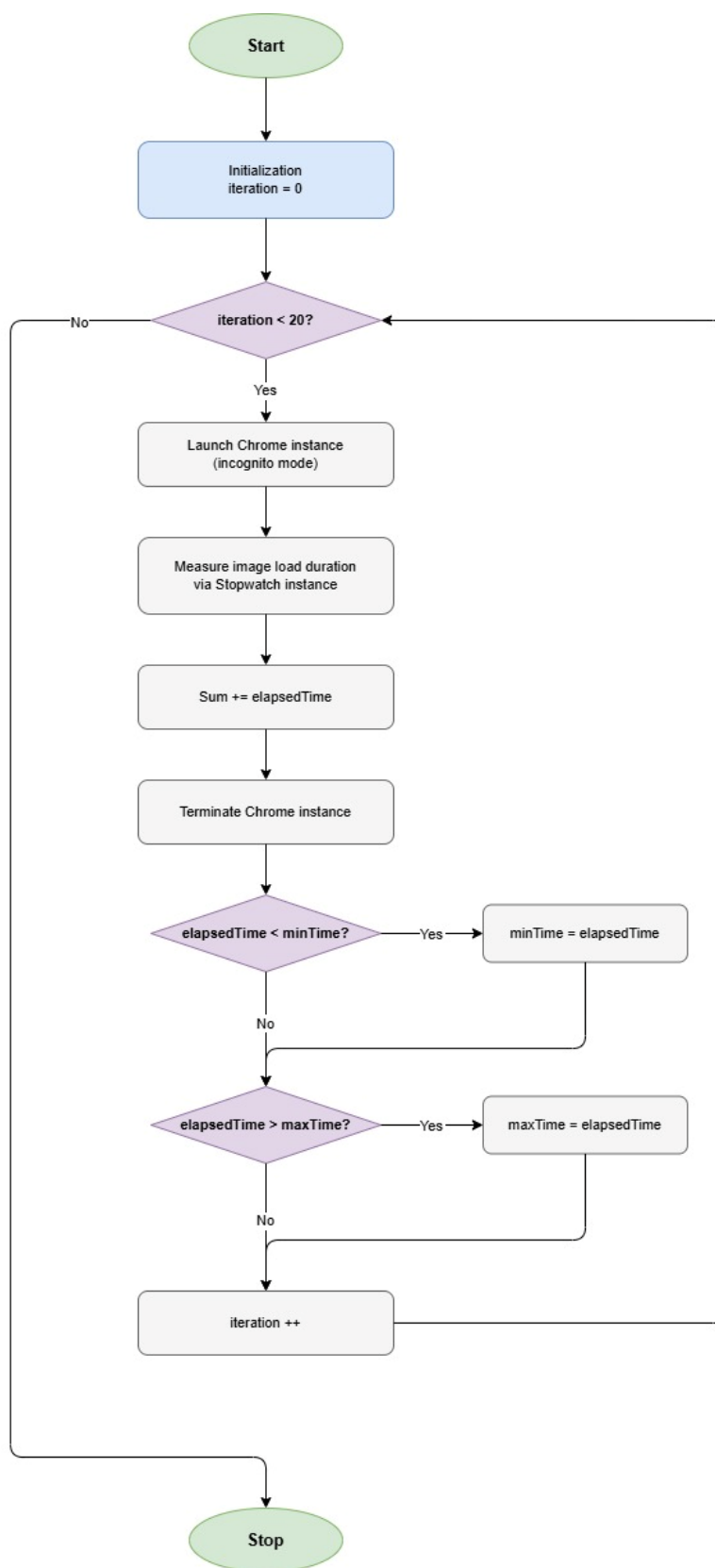


Figure 3. Logical flow of the testTime(...) function.

For the Image Classification application, the prediction system uses DL, an advanced subfield of ML. It employs a Convolutional Neural Network (CNN) that, through the processing of the 1672 labeled images, optimizes its internal parameters to identify the visual features and classify the images into the seven predefined categories. The architecture of a CNN is illustrated in Figure 4. An input image (depicted in the figure as a dog) passes through the convolutional layers, which consist of Convolution and Pooling operations, where the selection of distinctive elements occurs, enabling the AI to differentiate the image based on the unique features of every learned class. Subsequently, based on the identified features, the final classification prediction is generated via the processing during the Fully Connected Layer.

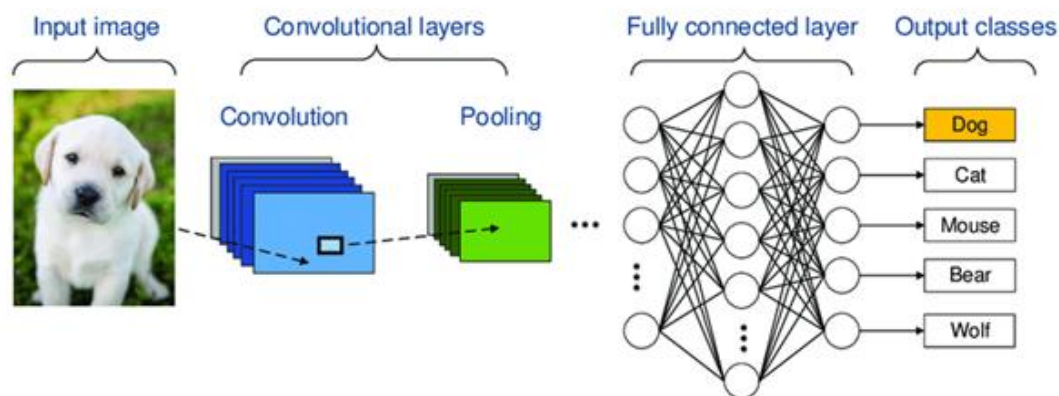


Figure 4. Conceptual diagram of a CNN for image classification [30].

The operational phases of the peripheral image classification application, the workflow of which is presented in Figure 5, follow an identification cycle that allows the processing of either individual instances or iterative batches of images. In Figures 1, 2, 3 and 5, the application's start and stop nodes are indicated in green, display blocks in red, iterative control structures in blue, and decision blocks in purple.

PERFORMANCE ANALYSIS OF DATA ACQUISITION METHODS

This paper aims to study the resource consumption and the execution latency for methods of data acquisition used in web scraping. The study was conducted using a C# script leveraging Selenium Webdriver, as outlined in the "Materials and Methods" section. Figure 6 presents the execution times recorded by the script for the sequential and parallel executions.

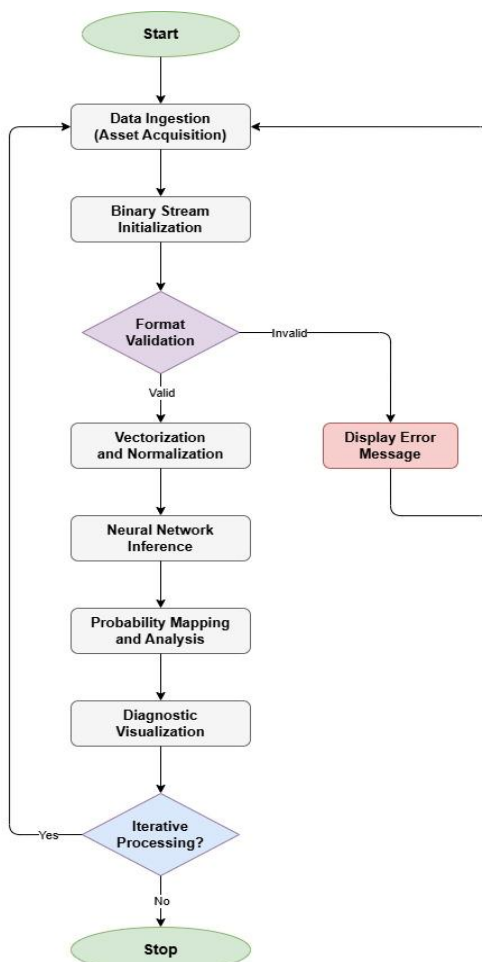


Figure 5. Logical flow of the image classification application process.

Sequential Processing Product added! Product added! Product added! Product added! Product added! Product added! Product added! Product added! Product added! Product added! Product added! Execution time: 00h 04m 01s	Parallel Processing (2) Product added! Product added! Product added! Product added! Product added! Product added! Product added! Product added! Product added! Product added! Product added! Execution time: 00h 02m 16s	Parallel Processing (3) Product added! Product added! Product added! Product added! Product added! Product added! Product added! Product added! Product added! Product added! Product added! Execution time: 00h 01m 46s
Parallel Processing (4) Product added! Product added! Product added! Product added! Product added! Product added! Product added! Product added! Product added! Execution time: 00h 01m 49s	Parallel Processing (5) Product added! Product added! Product added! Product added! Product added! Product added! Product added! Product added! Product added! Execution time: 00h 01m 27s	

Figure 6. Comparative execution times of sequential and parallel processing.

The experimental results are summarized in Table 1, which presents the sequential execution time (T_s), the parallel execution times (T_p), the calculated speedup factor (S), and the runtime reduction (ΔT). These metrics were derived using the following formulas:

$$S = \frac{T_s}{T_p} \quad (1)$$

$$\Delta T (\%) = \frac{T_s - T_p}{T_s} \times 100 \quad (2)$$

Table 1. Performance comparison: Sequential vs. Parallel execution latency.

Parallel Threads	T_s (s)	T_p (s)	S	ΔT (%)
2	241	136	1.77x	43.56%
3	241	106	2.27x	56.01%
4	241	109	2.21x	54.77%
5	241	87	2.77x	63.90%
Average	241	109.5	2.26x	54.56%

The data that was gathered within the study is depicted into the statistical analysis presented in Figure 7. It is evident that the parallel execution using five threads yields the shortest execution time, whereas the sequential baseline exhibits the longest. An interesting result in the statistic comes in the form of the performance anomaly encountered at the four threads configuration, where the T_p is 3s higher than that of the three thread counterpart.

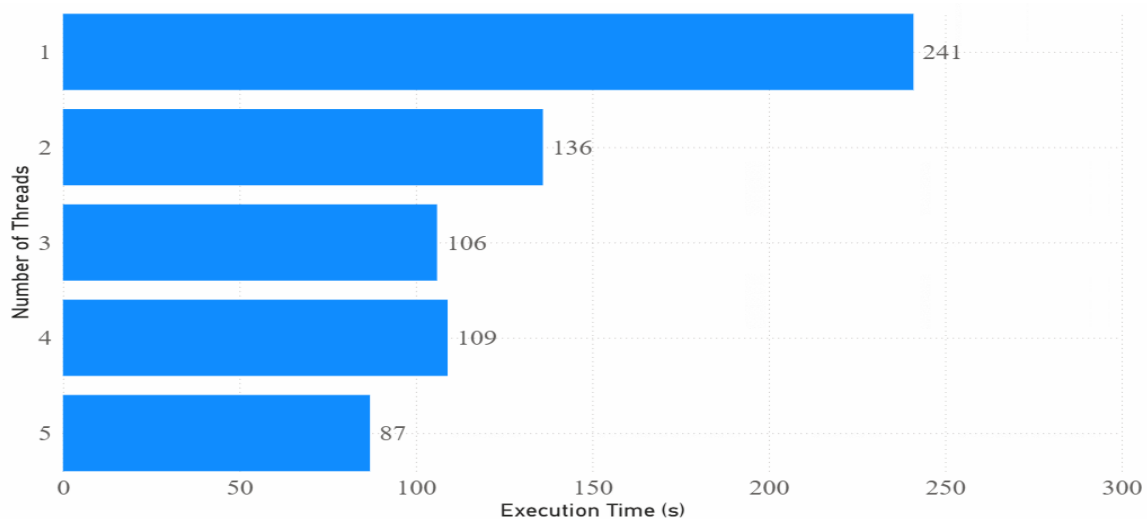


Figure 7. Bar chart visualization of execution times relative to the number of concurrent threads.

For the evaluation of the system resource usage during the application execution, memory consumption by the Chrome browser was monitored via Windows Task Manager as shown in Figure 8. Experimentally, it was determined that a single Chrome instance consumes between

280.8 MB and 321.4 MB of RAM during the whole execution time, with an additional overhead of 6.1MB for each background ChromeDriver process. The usage of RAM memory within the study is presented in Table 2.

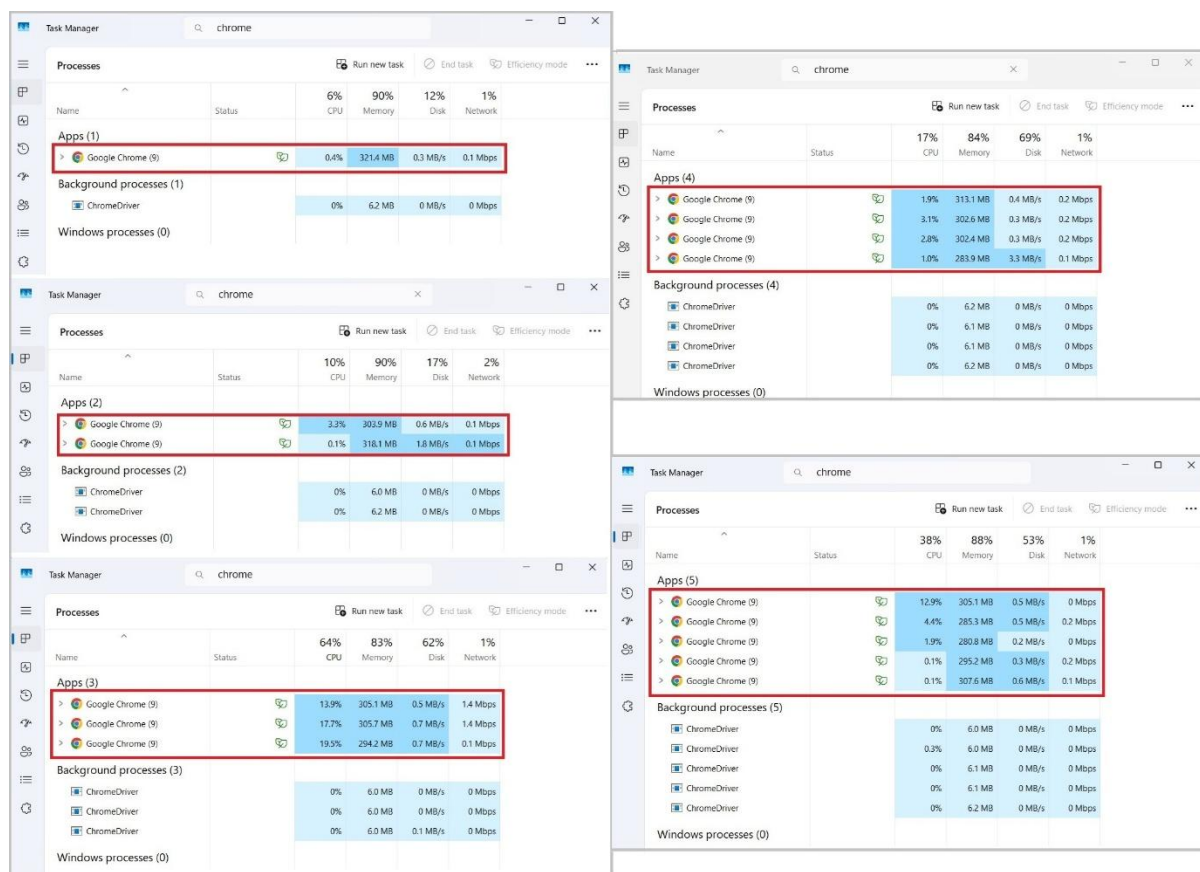


Figure 8. Real-time resource utilization monitor (RAM) per concurrent thread count, captured via Task Manager.

Table 2. Detailed analysis of RAM memory consumption based on the number of execution threads.

Number of Threads	Instance 1 (MB)	Instance 2 (MB)	Instance 3 (MB)	Instance 4 (MB)	Instance 5 (MB)	Average (MB)	Total RAM usage (MB)	Total RAM usage with Chrome Driver (MB)
1	321.4	-	-	-	-	321.4	321.40	327.50
2	303.9	318.1	-	-	-	311	622.00	634.20
3	305.1	305.7	294.2	-	-	301.66	905.00	923.30
4	313.1	302.6	302.4	283.9	-	300.5	1202.00	1226.40
5	305.1	285.3	280.8	295.2	307.6	294.8	1474.00	1504.50

By synthesizing both execution latency and RAM usage, it is concluded that the three thread configuration is the most optimal way of implementation among the five tested scenarios, of which a summary of both studied elements are presented in Table 3. For the transition from one to two threads results show a significant reduction in the execution time, having a remarkable 105 seconds improvement at a cost of approximately 300 MB of RAM. Conversely, moving from two to three threads yields a 30 seconds reduction with an additional 283 MB memory consumption. However, the transition from three to four threads proves to be the least efficient out of the five tests, as it leads to an increase in both the execution time and the RAM consumption making it the worst configuration. Furthermore, extending from three to five threads, the execution time gets reduced by just 19 seconds for a cost of aprox. 580 MB. These findings indicate that by adding more and more instances, the memory usage grows exponentially, while the execution speed diminishes rapidly.

Table 3. Summary of execution time and resource utilization (RAM) relative to the number of threads.

Number of Threads	Execution Time (s)	Total RAM usage with ChromeDriver (MB)
1	241	327.50
2	136	634.20
3	106	923.30
4	109	1226.40
5	87	1504.50

While parallel execution is superior to sequential processing in terms of latency, it necessitates a higher usage of memory resources. This study has shown that the three thread approach provides a balanced performance, maintaining an execution time under two minutes, which is closely aligned with the average ΔT values, while keeping the memory utilization below the 1 GB threshold.

COMPARATIVE ANALYSIS OF IMAGE STORAGE ARCHITECTURES: BLOB VS. FILE SYSTEM REFERENCES

For the study regarding image storage methods, a C# script was developed utilizing a Stopwatch object to measure the image rendering latency. The script uses Selenium WebDriver to access the e-commerce simulation platform. This paper aims to evaluate which method of storing the product images is more effective for an e-commerce application, particularly given that the continuous growth of internet data poses a significant resource management challenge within the cloud environments. Specifically, this research compares the loading behavior of two approaches: storing the product image directly into the database as a BLOB against storing only the image file path, which is subsequently fetched upon page load.

The simulation of an e-commerce application features a listing page containing 10 products, each of them with a name, price, star rating and image. For each iteration, a Chrome instance, controlled by Selenium, is initialized. The script starts the timer upon launch and terminates once the instance closes after all the images have fully rendered. The results that came upon the 20 iterations for both the server side and BLOB variants were compiled in Table 4, documenting their respective load times and identifying the faster approach for each cycle.

Table 4. Measured Page Load Times for Server and BLOB Storage Approaches.

Test Iteration	Server Load Time (ms)	BLOB Load Time (ms)	Better Performance
1	366	915	Server
2	351	944	Server
3	365	945	Server
4	390	942	Server
5	513	383	BLOB
6	393	1485	Server
7	383	400	Server
8	375	371	BLOB
9	357	386	Server
10	368	1352	Server
11	356	352	BLOB
12	382	374	BLOB
13	342	371	Server
14	373	382	Server
15	371	436	Server
16	354	399	Server
17	350	367	Server
18	366	403	Server
19	369	409	Server
20	375	386	Server

The mean load time for the file path approach is 374.95 ms, compared to 600.10 ms for the BLOB method, which indicates that the server side architecture is on average 1.6 times faster. Regarding peak latency, the BLOB variant displayed a significant volatility, with response times exceeding 1 second during sixth and tenth iterations, rendering it unreliable for a larger set of image datasets.

The resulting performance for the BLOB approach ranges within 352 MB to 1485 MB range making the method less predictable. Conversely, the server side variant yielded stable performance, with retrieval times confined within the 342 MB to 513 MB range confirming its reliability as illustrated in Figure 9.

From a resource management perspective, storing images in the database increases database size, which in turn increases backup and rollback execution times as the database scales. In contrast, the reference-based approach stores only the file path as a character string, offering a more efficient storage architecture. Ultimately, although the server-side approach demonstrated faster performance in 80% of the tests, it proved superior primarily due to its minimal storage overhead.

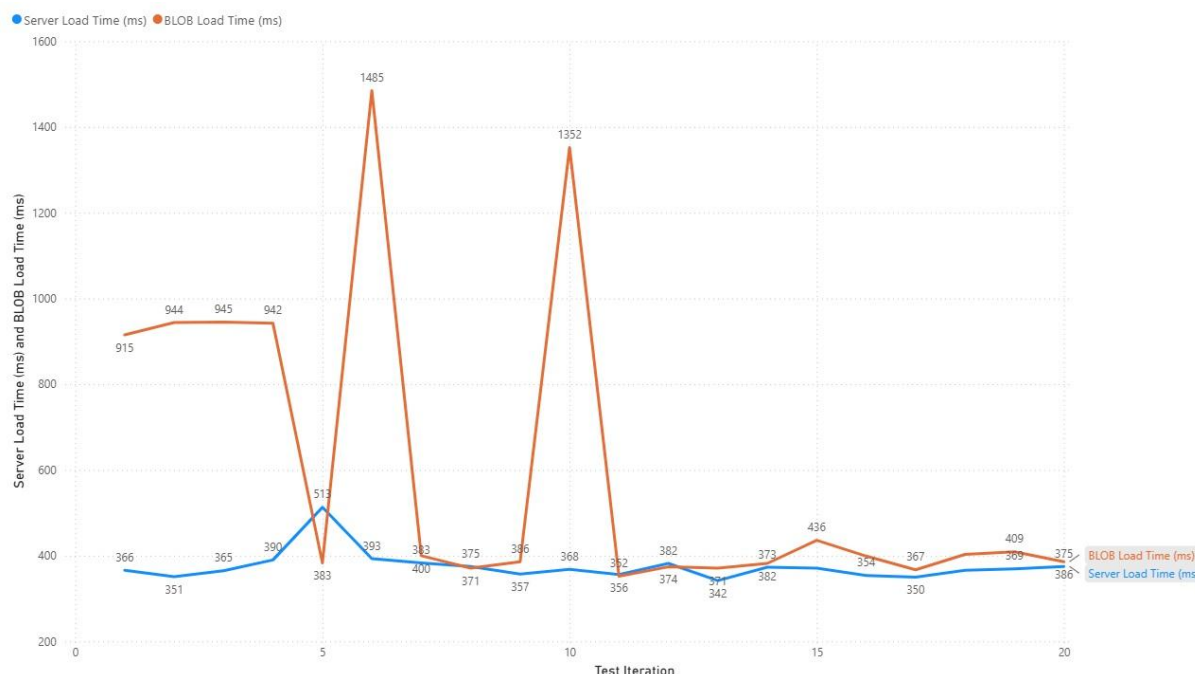


Figure 9. Comparative performance analysis: Server vs. Blob image retrieval latency across 20 iterations.

DEVELOPMENT AND VALIDATION OF THE IMAGE CLASSIFICATION MODEL

The application was developed in C# using the ML.NET ML framework and was trained using a subset of 1672 images from the Kaggle PC Parts dataset, as detailed in the "Materials and Methods" section.

For the execution of training routines on the local Central Processing Unit (CPU), the native ML.NET APIs were utilized during the pipeline development, for the optimization of the training duration time. The process was completed within 353 seconds, resulting in a model that has a micro-accuracy metric of 0.8274, which quantifies the system's performance in identifying the PC peripherals.

The effectiveness of the model was validated through a series of tests on multiple peripheral hardware, a key example of which is illustrated in Figure 10. These results highlight the accuracy of the model in identifying specific hardware components across varying environmental conditions and visual contexts.

Figure 10 presents the classification of a pair of speakers, where the model achieved a confidence score of 99.81%. The negligible distribution across the other categories indicates that the system isolates the structural features of the speakers, even when the lighting of the image is not ideal.

To further evaluate the model's versatility, additional empirical tests were conducted across other peripheral categories. In the case of a computer monitor that presented display artifacts such as vertical lines, the model achieved a confidence score of 99.84% demonstrating that the system effectively ignores these visual imperfections, by focusing instead on the primary silhouette and frame of the hardware. Similarly, a studio microphone was categorized with a

confidence of 93.43%. While a minor variance of 4.51% was attributed to the "Headphones" category due to the mounting structure and background of the picture, the primary classification remains statistically decisive. Lastly, the model successfully categorized a webcam with a confidence score of 98.69%. The successful classification of the device, despite the webcam being physically mounted on top of a computer monitor, indicates that the model prioritizes the essential features of the webcam over the surrounding environment.

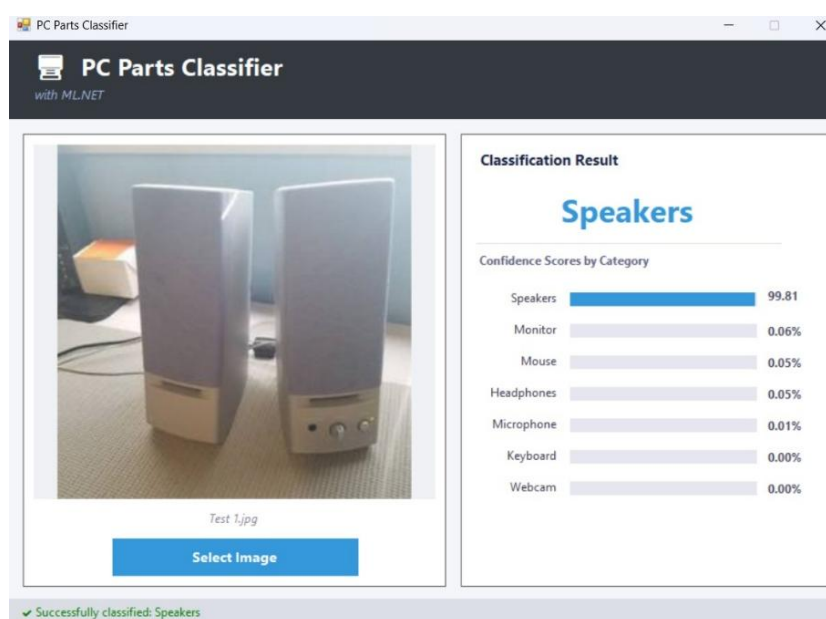


Figure 10. Classification result for desktop speakers.

Across these empirical tests, it is observed that the model consistently yields confidence scores above 93%, validating the training phase. The model successfully generalized the visual features of each class, allowing for the classification of peripherals even in contexts where lighting conditions or cluttered backgrounds would otherwise complicate the process. These results confirm the reliability of the classification system for e-commerce cataloging and inventory management tasks.

CONCLUSIONS

The improvement of the web scraper algorithms and efficient means of storing images is another cornerstone for the scalability of e-commerce websites. This paper presented a thorough review of the above-mentioned procedures, proving that utilizing three threads simultaneously constitutes the best trade-off for latency and RAM usage while ensuring the runtime remains below the 2-minute mark and keeps memory below the 1 GB mark.

Additionally, the comparative investigation of BLOBs vs. server-side file locations showed that the latter guarantees faster retrieval rates and increased stability of web applications.

As a complementary solution, the incorporation of ML.NET image classifier demonstrated excellent results with the accuracy score reaching 82.74% and reliable performance in numerous test runs.

The further development of the presented research would imply using a more diverse set of data together with additional hardware devices to improve the generality of the model. Next, using GPU computing techniques during model training will reduce the computational costs in the future. Lastly, the presented study may be improved through the introduction of dynamic load balancing strategies for cloud-based e-commerce websites.

REFERENCES

- [1] Lie W., Jiang B., Zhao W., Obstetric Imaging Diagnostic Platform Based on Cloud Computing Technology Under the Background of Smart Medical Big Data and Deep Learning, IEEE Access, vol. 8, pp 78265-78278, 2020. <https://doi.org/10.1109/access.2020.2988563>.
- [2] Deserno T.M., Haak D., Brandenburg V., Deserno V., Classen C., Specht P., Integrated Image Data and Medical Record Management for Rare Disease Registries. A General Framework and its Instantiation to the German Calciphylaxis Registry, Journal of Digital Imaging, vol. 27/issue 6, pp 702-713, 2014. <https://doi.org/10.1007/s10278-014-9698-8>.
- [3] De-Giorgio F., Benedetti B., Mancino M., Sala E., Pascali V.L., Post-Mortem imaging biobanks: Building data for reproducibility, standardization, and AI integration, European Journal of Radiology, vol. 185, pp 112011, 2025. <https://doi.org/10.1016/j.ejrad.2025.112011>.
- [4] Mougiakakou S.G., Valavanis I.K., Mouravliansky N.A., Nikita K.S., Nikita A., Diagnosis: A Telematics-Enabled System for Medical Image Archiving, Management, and Diagnosis Assistance, IEEE Transactions on Instrumentation and Measurement, vol. 58/issue 7, pp 2113-2120, 2009. <https://doi.org/10.1109/tim.2009.2015538>.
- [5] Rosca C.-M., Stancu A., A Review of Neuro-ML Breakthroughs in Addressing Neurological Disorders, Applied Sciences, vol. 15/issue 10, pp 5442, 2025. <https://doi.org/10.3390/app15105442>.
- [6] Rosca C.-M., Bold R.-A., Gereia A.-E., A Comprehensive Patient Triage Algorithm Incorporating ChatGPT API for Symptom-Based Healthcare Decision-Making, Emerging Trends and Technologies on Intelligent Systems. ETTIS 2024. Lecture Notes in Networks and Systems, Noida, India, vol. 1073, pp 167-178, 2025. https://doi.org/10.1007/978-981-97-5703-9_13.
- [7] Mhereeg M.R., Tawil A.G., Analysis and Design of a Filestream Based English Language Learning System Journal of Computers, vol. 10/issue 4, pp 268-283, 2015. <https://doi.org/10.17706/jcp.10.4.268-283>.
- [8] Zerkov V.V., Pritychenko B., Totans J., Vrapcenjak L., Rodionov A., Shulyak G.I., EXFOR-NSR PDF database: a system for nuclear knowledge preservation and data curation, Journal of Instrumentation, vol. 17/issue 03, pp P03012, 2022. <https://doi.org/10.1088/1748-0221/17/03/p03012>.
- [9] Rosca C.-M., Stancu A., Emerging Trends in AI-Based Soil Contamination Monitoring and Prevention, Agriculture, vol. 15/issue 12, pp 1280, 2025. <https://doi.org/10.3390/agriculture15121280>.
- [10] Khan M.A.U., Daud F., Akram A., Naqvi R.A., Mehmood M., Hussain D., Soomro T.A., A Scale Normalized Generalized LoG Detector Approach for Retinal Vessel Segmentation, IEEE Access, vol. 9, pp 44442-44452, 2021. <https://doi.org/10.1109/access.2021.3063292>.

- [11] Adal K.M., Sidibé D., Ali S., Chaum E., Karnowski T.P., Mériaudeau F., Automated detection of microaneurysms using scale-adapted blob analysis and semi-supervised learning, *Computer Methods and Programs in Biomedicine*, vol. 114/issue 1, pp 1-10, 2014. <https://doi.org/10.1016/j.cmpb.2013.12.009>.
- [12] Barón J.R., Bernabé G., González-Férez P., García J.M., Casas G., González-Carrillo J., Semi-Automatic Refinement of Myocardial Segmentations for Better LVNC Detection, *Journal of Clinical Medicine*, vol. 14/issue 1, pp 271, 2025. <https://doi.org/10.3390/jcm14010271>.
- [13] Malinowski K., Saeed K., An iris segmentation using harmony search algorithm and fast circle fitting with blob detection, *Biocybernetics and Biomedical Engineering*, vol. 42/issue 1, pp 391-403, 2022. <https://doi.org/10.1016/j.bbe.2022.02.010>.
- [14] Nambisan A.K., Lama N., Phan T., Swinfard S., Lama B., Smith C., Rajeh A., Patel G., Hagerty J., Stoecker W.V., Stanley R.J., Deep learning-based dot and globule segmentation with pixel and blob-based metrics for evaluation, *Intelligent Systems with Applications*, vol. 16, pp 200126, 2022. <https://doi.org/10.1016/j.iswa.2022.200126>.
- [15] Rosca C.-M., Stancu A., Popescu C., Machine Learning Models for SQL Injection Detection, *Electronics*, vol. 14/issue 17, pp 3420, 2025. <https://doi.org/10.3390/electronics14173420>.
- [16] Krishna T.L.S.R., Ragunathan T., Battula S.K., Improving performance of a distributed file system using a speculative semantics-based algorithm, *Tsinghua Science and Technology*, vol. 20/issue 6, pp 583-593, 2015. <https://doi.org/10.1109/tst.2015.7349930>.
- [17] Huang H., Guan Z., Yu B., Li X., Ge M., Ma C., Ma X., Efficient Single-Server Private Information Retrieval Based on LWE Encryption, *Mathematics*, vol. 13/issue 21, pp 3373, 2025. <https://doi.org/10.3390/math13213373>.
- [18] Voica M.C., Stancu A. Corporate Social Responsibility Reporting: Background, Evolution and Sustainability Promoter. In *Sustainable Management for Managers and Engineers*, C. F. Machado and J. P. Davim (Eds), ISTE Wiley, London, United Kingdom, 2021. <https://doi.org/10.1002/9781119804345.ch6>.
- [19] Zhu J., Zhang Z., Zhao F., Su H., Gu Z., Wang L., Efficient Management and Scheduling of Massive Remote Sensing Image Datasets, *ISPRS International Journal of Geo-Information*, vol. 12/issue 5, pp 199, 2023. <https://doi.org/10.3390/ijgi12050199>.
- [20] Rosca C.-M., Stancu A., Gortoesu I.-A., Advanced Sensor Integration and AI Architectures for Next-Generation Traffic Navigation, *Applied Sciences*, vol. 15/issue 8, pp 4301, 2025. <https://doi.org/10.3390/app15084301>.
- [21] Gorokhovatskyi V., Tvoroshenko I., Yakovleva O., Hudáková M., Image Description Compression in Classification Structural Methods, *IEEE Access*, vol. 13, pp 43631-43641, 2025. <https://doi.org/10.1109/access.2025.3548910>.
- [22] Zhang J., Han S., Wan J., Zhu B., Zhou L., Ren Y., Zhang W., IM-Dedup: An Image Management System Based on Deduplication Applied in DWSNs, *International Journal of Distributed Sensor Networks*, vol. 9/issue 7, pp 625070, 2013. <https://doi.org/10.1155/2013/625070>.
- [23] Liu F., Yang C.-Y., Yang J., Kong D.-L., Zhou A.-M., Qi J.-Y., Li Z.-B., A hybrid with distributed pooling blockchain protocol for image storage, *Scientific Reports*, vol. 12/issue 1, pp 3457, 2022. <https://doi.org/10.1038/s41598-022-07494-9>.

-
- [24] Xu W., Moon H., Kashyap S., Tseng P.-N., Kim T., Fuzzing File Systems via Two-Dimensional Input Space Exploration, IEEE Symposium on Security and Privacy, San Francisco, CA, USA, pp 818-834, 2019. <https://doi.org/10.1109/sp.2019.00035>.
 - [25] Rosca C.-M., Stancu A., Quality assessment of GPT-3.5 and Gemini 1.0 Pro for SQL syntax, Computer Standards & Interfaces, vol. 95, pp 104041, 2026. <https://doi.org/10.1016/j.csi.2025.104041>.
 - [26] Diez Y., Suzuki T., Vila M., Waki K., Automatic Processing of Historical Japanese Mathematics (Wasan) Documents, Applied Sciences, vol. 11/issue 17, pp 8050, 2021. <https://doi.org/10.3390/app11178050>.
 - [27] Arad B., Kurtser P., Barnea E., Harel B., Edan Y., Ben-Shahar O., Controlled Lighting and Illumination-Independent Target Detection for Real-Time Cost-Efficient Applications. The Case Study of Sweet Pepper Robotic Harvesting, Sensors, vol. 19/issue 6, pp 1390, 2019. <https://doi.org/10.3390/s19061390>.
 - [28] Dewi C., Dai G., Christanto H.J., Analysis of Internet Movie Database with Global Vectors for Word Representation, Vietnam Journal of Computer Science, vol. 11/issue 03, pp 343-362, 2024. <https://doi.org/10.1142/s2196888823500215>.
 - [29] Asaniczka. PC Parts Images Dataset [Classification], Kaggle, 2024. <https://doi.org/10.34740/kaggle/dsv/7565076>. (accessed on 6 May 2026).
 - [30] Shaikh Z., Pachory K. Introduction to Convolutional Neural Networks. Available online: <https://hexaware.com/blogs/introduction-to-convolutional-neural-networks/> (accessed on 10 May 2026).

Received: June 2026; Revised: July 2026; Accepted: July 2026; Published: July 2026