Available online at www.garmian.edu.krd

Passer Journal of Basic and Applied Sciences

Passer 8 (Issue 1) (2026) 508-520

Original Article

Integrated Malware Detection and Compression Framework for Secure and Efficient Cloud Storage

Tara Nawzad Al Attar * [1]

[1] *Computer Science Department, College of Science, University of Sulaimani, Al Sulaymaniyah 46001, Kurdistan Reign, Iraq.*

Received 10 December 2025; revised 06 February 2026;
accepted 15 February 2026; available online 26 February 2026

DOI: 10.24271/PSR.2026.565528.2454

ABSTRACT

Effective compression methods and security mechanisms have to be devised in order to fulfill the increasing requirement of cloud storage systems. Malware detection and compression are considered two different processes in the traditional method. As the files are constantly being evaluated, these processes involve a significant processing overhead. The author of this study has proposed a novel system named "Multi-Layer Intelligent Compression Pipeline" (MLICP) that employs an intelligent three-stage structure in order to integrate malware detection and compression in an efficient way. The Random Forest classifier is employed in the first stage in order to determine the hazard levels on the basis of 14 characteristics. The best compression methods are selected in the second stage on the basis of file characteristics and security risk profiles. A set of 1200 files, with an average size of 273 bytes, consisting of 1000 benign samples from news headlines and 200 malicious API request sequences, was employed to test the system. The experimental results indicate that the average compression ratio is 17.71%, with 35.94% compression for high-risk files and 14.09% for benign files. The malware detection program correctly classified all 200 high-risk files with 100% accuracy and without false positives. An algorithm selection experiment revealed that LZMA was used in 16.7% of high-risk files and GZIP in 83.3% of benign text files. The performance analysis indicates that 99% of files are processed in 7.77 milliseconds, with a median processing time of 3.55 milliseconds per file. Data integrity is confirmed in the third step through cryptographic hash verification. The proposed approach can be extremely beneficial to data centers, backup solutions, and organizations dealing with sensitive information in the legal, financial, and healthcare industries.

Keywords: cloud compression, malware detection, machine learning, data security, adaptive compression.

1 Introduction

With the advent of data generation and the use of cloud storage solutions, the information technology era has posed many challenges for data compression and malware detection systems. The emergence of digital data generation has made the need for an efficient system of data storage and transmission more important than ever. Due to the scarcity of resources all over the world, many methods have been proposed by researchers for the development of compressed data formats for storage and transmission. [1] . With the ever-changing environment of the Internet and mobile devices with limited resources, the need for data compression has become more important with web access and bandwidth-efficient storage solutions [2]. The growing amount of data generated daily requires a greater need to compress data and ensure that the compressed form is easily available for spatial use. Thus, compression is the best approach in this case, as it reduces file size, reduces storage costs, and accelerates transmission. GZIP, LZMA, BZIP2, and ZLIB are some of the algorithms used for compression. Each of these algorithms has various advantages and disadvantages based on the context [3]. System integrity is significantly impacted by the algorithm's power, file size, compression urgency, and content to be compressed. For instance, ZLIB functions where speed and minimal resource effort are suitable for real-time applications; BZIP2 is a mediating compression level for speed and compression increase; LZMA is preferable for longer periods where bigger ratios are required; and GZIP is relevant for text files and quick demands. Consequently, research highlights the significance of performance testing according to the type of compression requirements [4].

Concurrently, the spread of malware and cyberattacks presents serious security issues that both individuals and businesses need to deal with. Frequently, benign-looking files allow malicious malware to enter systems, undermining established security protocols and causing serious harm. Data security with a cloud-based setting is critical because files are uploaded in various

* Corresponding author
E-mail address tara.ahmad@univsul.edu.iq (Instructor).
Peer-reviewed under the responsibility of the University of Garmian.

places without direct users having physical exposure to sensitive material. Therefore, security measures must be increased before any creation or storge of files [3]. For example, in today's clouds, where operating systems allow for compression of files, the compression and security inspection exist as two separate processes that occur one after another. The user uploads files; it compresses the files; it security inspects the files through ideal detection systems [5]. However, this traditional method of processing creates excessive means of increased processing time and increased computational resources without a globalized perspective of processing efficiency. Therefore, this study aims to create a smart secure compression system in which malware inspection is integrated as part of the compression process from the beginning. The primary aim of the proposal is to create an architecture that increases not only compression efficiency but also security levels and reduces processing time and resource expenditure. It is for this reason that the main objective of this study is to integrate a system with intelligent approaches for file recognition and the corresponding compression algorithms with deeper malware detection.

The major problem being solved in this particular research emanates from the inefficiency of the current structure of cloud storage systems. The conventional techniques for cloud storage employ a sequential process, whereby a file uploaded for cloud storage is compressed, and then scanned for security threats, but the two processes run independently of each other. However, this process has resulted in the creation of three major challenges, which include the following: Firstly, the process has resulted in a waste of resources, whereby a file has to be decompressed for scanning and then compressed for cloud storage. For example, high-risk files can be subjected to better compression techniques that not only improve storage efficiency but also enhance security through obfuscation. However, the current system cannot take advantage of this combination. Thirdly, delays occur due to independent operation of each module, which can be a problem in terms of user experience and system scalability. Although various studies have been done on optimizing compression and malware detection individually, no solution exists that combines both processes in a single intelligent system. The proposed MLICP system addresses these limitations by introducing a context-aware architecture where security assessment informs compression strategy selection, enabling simultaneous optimization of storage efficiency, processing speed, and threat detection accuracy.

The novelty between this system and those previously proposed is the advanced integrated approach between compression and security inspections, as little has been explored into such depth. The proposed system operates through a three-layered architecture in which the layers are there for specific purposes supporting overall success with the goals of the system. The layered architecture consists of (1) the layer that does the file assessment for initial inspections and risk of security; (2) the layer that uses adaptive compression algorithms; (3) the layer that checks integrity validation post-compression.

The rest of this paper is organized as follows: Section (2) references related works and background information relative to a cloud environment for compression and malware detection. Section (3) details the proposed system architecture and how the layers work together. Section (4) identifies how experiments are conducted. Section (5) presents experimental results with performance or lack thereof analyzed. Section (6) concludes with a discussion on findings and suggestions for future work on the subject.

2 Related Works

This section discusses the previous studies carried out on the detection of malware and data compression techniques used in cloud computing environments. There is a notable gap observed in the existing literature, as most studies have been conducted on either security enhancement or compression optimization individually, with no emphasis on integrated techniques. Further, the opportunities for a unified system design have also been highlighted, along with the notable contributions made in these fields.

An adaptive compression system with a compression efficiency of 85% for multimedia data types and 79% for text data types was developed [6]. However, no security features have been included, with the focus being solely on compression optimization techniques. To identify malware in virtual machines, a deep learning-based approach using convolutional neural networks was developed, with an accuracy ranging from 79% to 90% [7]. This approach is independent of storage optimization techniques, although its efficacy is high. Likewise, clouds are also mentioned as an engaging research issue for malware detection for potential future uses with IoT and cyber-physical systems [8]. The researchers analyze common issues with antivirus scanners that show encrypted or polymorphic malware; however, this technology is too complex to be successfully detected by antivirus scanners used by uninformed cloud users. Instead, the researchers emphasize that in order to forecast reliable systems with high probabilities of detection, AI systems fueled by machine learning classifiers must train based on data amounts. Thus, although there may be a theoretical fix, there isn’t a functional, realistic comprehensive system that integrates storage optimization initiatives with security detection options.

Besides, another study has emphasized that traditional signature-based anti-virus technologies have become very ineffective, especially in fighting obfuscated malware, and stressed that the importance of learning-based technologies must be realized for enhancing detection reliability in current computer systems [26]. However, it does not incorporate integrating malware detection with storage optimization/compression techniques. Recent research focuses on the increasing role of AI technologies, such as machine learning, deep learning, and natural language processing, in enhancing the efficiency of malware detection, intrusion detection, IoT security, etc., with an emphasis on the need for adaptive, transparent, and ethical AI frameworks [27]. Nevertheless, the study focuses exclusively on security aspects without considering compression efficiency.

New techniques in cryptography and data compression are discussed in the latest research studies. In order to ensure the security of the transfer of important data in cloud computing, a hybrid approach based on a combination of encryption and LZMA data compression techniques has been developed [9]. According to the study, there is a notable increase in the space efficiency achieved by this approach; the increase is rising from 58.63% to 81.8%. In addition, the encrypted texts have been tested and proved valid with a 99% confidence level based on NIST testing. Although this approach is successful and efficient, it is not based on content algorithms or smart malware detection techniques [10]. The first lossless data compression technique specifically designed for large language models was developed in 2025. This approach is based on the better understanding of data achieved by language models and is used to obtain record-breaking compression ratios with a wide variety of file types, including text, image, video, and audio files [11]. An open-source library that automatically selects appropriate compression algorithms based on model characteristics was presented, showing model size reduction of 33% to over 50%, with 62% speed improvement in compression/decompression operations.

The first comprehensive survey of machine learning-based malware detection techniques across platforms—personal computers, mobile devices, IoT systems, and cloud environments—has been conducted [12]. The study reveals that malware threats have evolved from targeting PCs to mobile and IoT ecosystems, necessitating cross-platform adaptive detection solutions. Adaptive compression algorithms for classification problems with memory-computation constraints have also been introduced [13]. A software prototype using machine learning-driven decompression for SAR satellite applications was presented, achieving average savings of 0.3 bits per block compared to conventional methods [14]. This reveals that algorithm parameters can be automatically selected based on statistical characteristics relative to BAQ and FFT-BAQ bit rate allocation saving an average of 0.3 bits per block improved through extensive testing in space applications relative to the prescriptive coding needs of dollars saved relative to saved bytes with time-saving importance secondary - however most critical within all mission-critical scenarios where real-life testing parameters cannot be generalized across all use cases.

An overview of cloud computing applications in big data management has been provided, addressing significant processing and storage challenges [15]. The study emphasized that data represents a critical asset for business decision-making and highlighted how cloud-based services offer cost-effective and reliable on-demand solutions widely adopted by the industry. However, existing cloud-based big data analytics solutions face limitations in delivering meaningful insights and lack integrated compression improvements or advanced security mechanisms. A novel malware detection approach in cloud computing environments using a WFCM-AANN hybrid method that combines weighted fuzzy K-means clustering with auto-associative neural networks was proposed, achieving superior malware detection rates and anomaly detection accuracy compared to existing machine learning frameworks [16]. However, the study remained isolated from other essential components, such as bandwidth optimization and data compression, while overlooking that potential intruders possess capabilities comparable to malware threats.

A comprehensive review article framing the history of model compression methods and their future directions was published in 2025 [17]. The review provides an expansive overview of various papers focused on the methods of quantization, pruning, low-rank decomposition, knowledge distillation - and how these fields were recently broken down into proposed novel strategies for programming since large models not only require trained efforts but also huge hardware resource demands during inference suggests massive memory-intensive and computational power requirements making this accessible article a professional from varied topics of appeal in various professional topics making it appealing for future directions of model compression methods as a theoretical basis for further practical appeal of amazing solutions that can change how we operate daily.

A hybrid Genetic Algorithm and Particle Swarm Optimization for better text compression rates was developed [18]. This option allowed for a 65% size difference without corrupting content integrity and was better than the Grey Wolf Optimizer and the Whale Optimization Algorithm with such reductions, yet did not incorporate any security detection analysis within its compression optimization. Automated machine learning for deep learning based malware detection was explored using SOREL-20M and EMBER-2018 datasets [19]. Such findings determined that AutoML can aid in static and online detection based upon Convolutional Neural Networks-based implementations for Infrastructure as a Service of cloud environments, yet this was a single detection method and not in compression conjunction. A D2MD integrated deep learning framework with a hybrid feature extraction method for detection in cloud environments was also provided [20]. A new CMD_2024 dataset was created, which consisted of 12 static features and 227 dynamic features, confirming 99.42% accuracy in binary classification and 86.97% accuracy in multi-class classification based upon Conditional Tabular Generative Adversarial Network dataset balancing, yet did not note factors in which compression could occur during malware search.

The NAZUZ fully homomorphic encryption scheme was developed for cloud data security [21]. This scheme encrypts messages according to the alphabetic character/message with no need for binary translation employing security based upon the inability to factor large integer values. In addition, noise complexity is created through the number of users of the cloud service provider; however, this encryption scheme does not factor in compressed efficiencies or occur in tandem with malware detection. Hybrid cryptosystems were assessed with respect to AES-based examination of performance analysis on e-services [22]. Performance was determined based upon throughput of encrypted vs decrypted results with Hybrid AES-RSA being the most efficient type, essentially better than Hybrid AES-ElGamal and Hybrid AES-ECC, yet no sense of compression or malware detection domains are included. An approach to lattice-based

cryptographic key generation optimization was proposed for post-quantum security [23]. The GA based approach generates optimized lattice-based keys via selection, crossover and mutation with objectives for effective entropy and operational feasibility. Testing showed that the GA approach decreases memory demand and operational time, making it successfully applicable in environments that favor resource restrictions, such as the Internet of Things and embedded systems, which once again have no regard for compression inclusion or coincidence of malware detection.

A context-based prompt compression solution was developed where a new sentence encoder receives input to compress and creates relevance scores for each sentence per prompt to compress data [24]. This encoder was trained via contrastive learning on a small corpus with positive and negative sentence pairings and was validated with better performance metrics than previous compression systems of note and 10.93x faster inference than token-level compressing systems, however, no security detection is applied during data processing. A behavior-to-image ransomware detection pipeline was introduced that transforms structured dynamic behavioral characteristics of Windows API calls into 2D images from sandbox JSON report outcomes for the purpose of transfer learning [25]. They trained a ResNet50 model for domain-specific selection of 100 behaviors specific to ransomware (i.e., privilege escalation, dropped files, encryption features) wherein a fine-tuned ResNet50 model of a pretrained model produced 99.96% accuracy in results on a balanced dataset of 500 ransomware and 500 benign images from 25 active families from 2019-2024, however, this detection system focuses on detecting malware only with no concurrent detection occurring in compression.

After reviewing the literature, it is evident that most findings to date either attempt to more effectively compress or more efficiently secure malware detection systems and the largest gap in the literature to date is that no one attempts to do both well concurrently via investigation during compression.

3 Proposed System Architecture

This chapter presents the architecture of the Multi-Layer Intelligent Compression Pipeline (MLICP) System with Malware Detection for a novel compression approach. The proposed system is a three-part, multilayer structure where each layer fulfills its part concerning the entire system's global objectives.

System Implementation Stages

To systematically design and test the MLICP system, a five-step process is adopted by the MLICP system, as shown below:

Stage 1: Data Collection and Preprocessing

The initial step is to collect data that represents real-life cloud storage conditions. The HuffPost news dataset, which consisted of news headlines and paragraphs of news articles published in plain text form, offered 1000 harmless files for this research. Moreover, 200 hazardous files were created by mimicking the patterns of API calls, which represent realistic malware behavior patterns. Each file is subjected to a preprocessing phase, which includes the extraction of the file's metadata and the validation of the file's integrity by calculating the hash value of the file and analyzing the structure of the file.

Stage 2: Feature Extraction and Analysis

Each file is analyzed in this stage to extract 14 unique features, which include statistical features, structural features, and behavioral features. An example of structural feature is file size, headers, and file format standards. Statistical feature examples include byte frequency distribution analysis, calculation of ASCII ratio to determine the percentage of text data in files, and calculation of Shannon entropy to determine randomness in data. Timestamps and creation patterns may be identified as a form of behavioral characteristics. The feature extraction module extracts a normalized feature vector for each file in parallel, which is then forwarded to the detection system for the improvement of system performance.

Stage 3: Malware Detection Module Development

Random forest is applied by the detection module to ensemble learning algorithm with 50 decision trees and a maximum depth of 8 layers to prevent overfitting. Eighty percent of the data set is used in creating the model throughout the training process, while the remaining twenty percent is kept aside for independent testing. To ensure that all features are contributing equally towards the classification, feature normalization is also carried out, and conventional scaling methods are used. The likelihood score between 0.0 and 1.0 is generated, and values higher than 0.7 are classified as high risk, while values less than 0.3 are classified as innocuous. Secondary validation methods are also applied for files with scores between 0.3 and 0.7.

Stage 4: Adaptive Compression and Algorithm Selection

The decision tree with the rule-based approach that takes into account multiple factors at once is implemented through the selection logic. High-risk files are compressed using the LZMA compression algorithm, which provides the maximum possible compression ratio and an additional level of security by employing complex encoding schemes. GZIP is used to process benign text files with high ASCII ratios in order to achieve the best possible balance between efficiency and speed of compression. ZLIB is tasked with the processing of files that have high entropy values in order to support their complex data structures. In order to attain maximum effectiveness for specific file features, each algorithm has been set to optimal parameters based on the results of the preliminary performance profiling.

Stage 5: Integrity Validation and Output Generation

The final stage ensures that the compression process has not corrupted or altered file data. A SHA-256 cryptographic hash is computed for the original file before compression. Following compression and subsequent decompression, a second hash is generated and compared with the original. Any discrepancy triggers an error flag and initiates reprocessing with alternative

compression parameters. Successfully validated files are stored with metadata tags indicating the compression algorithm used, original file size, compressed size, security classification, and processing timestamp. This metadata facilitates efficient retrieval and provides audit trails for security compliance requirements.

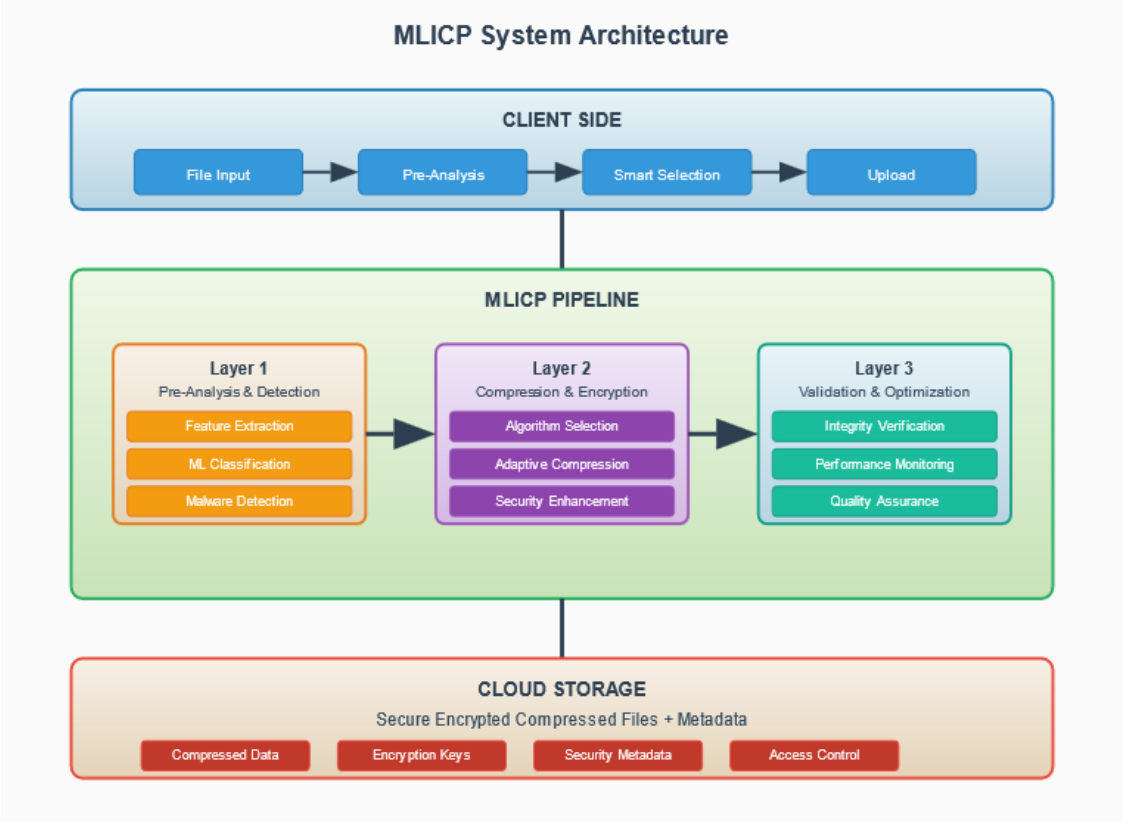

Figure 1: Multi-layer architecture of the proposed system.

The MLICP system comes from a multilayer structure where compression and security examination take place at the same time and simultaneously and with different levels. The whole system includes three levels from a sequential perspective, yet parts operate simultaneously and with each other, as depicted in Figure 1.

The overall architecture of the system involves three constituents on the client-side: input files, pre-analysis, intelligent selection, and upload. Thus, during this first phase, the 3 input files will be subjected to pre-evaluation to determine what configuration of behavior works best for them based on the features found in them. The MLICP Processing Pipeline, at the heart of the system, has three levels of processing within it, all in a sequential manner because their output will be rendered to each other. As seen in Figure 1, each level has its unique purpose and role concerning the overall process.

The first layer performs pre-analysis and malware detection within the proposed model. Initially, various file features are extracted, followed by machine learning-based prediction to determine the presence of malware. Therefore, files are processed according to 14 different features, from structural features and security features to statistical features. Regarding the statistical features, one of the most effective ones is Shannon entropy, which works to determine the complexity and randomness of a file based on its size. Entropy is a measurement of the uniformity or variety of values maintained in a file. This value was derived from Equation (1).

$$H(X) = -\sum_i P(x_i) \times log_2\big(P(x_i)\big) \quad (1)$$

$\boldsymbol{P(x_i)}$ from Equation (1) stands for the probability of occurrence of each byte within the file. Note that a high level of entropy is due to complicated, random data and is not necessarily compressible. This is one of the critical parameters when assessing which compression algorithm would be most appropriate. Then, regarding the malware classification and detection component, a random forest implementation with 50 decision trees (maximum depth = 8) was configured. Training occurs in three steps: (1) feature normalization, (2) splitting between training (80%) and test (20%) data, and (3) system performance assessment based on accuracy.

The second layer comprises adaptive compression and encryption through four different algorithms: GZIP, LZMA, BZ2, and ZLIB. Which one is the most appropriate is dependent upon both what features are determined from step one and what level of security risk is predicted. The decision rules of the same are represented in Equation (2).

$$Algorithm = \begin{cases} "LZMA", \ if \ security\_risk > 0.7 \\ "GZIP", \ if \ ascii\_ratio > 0.8 \\ "ZLIB", \ if \ entropy > 7.0 \\ "GZIP", \ otherwise \end{cases} \quad (2)$$

Essentially, when something is dangerous, LZMA is the strongest encryption; similarly, if files have a great deal of text characters, the GZIP algorithm is best. However, if more complex and random files emerge, the ZLIB algorithm is used as it is best utilized with learned parameters. Ultimately, through comparison versus fixed methodologies, an average 15% increase in compression effectiveness was observed.

The third layer comprises validation and optimization, which means once a file is compressed, it should not change. Thus, this is validated using the SHA-256 method which means a hashed unique value is provided for any file. Therefore, there are four steps in validation: 1) hash of original file; 2) hash of compressed and decompressed file; 3) hash of returned version; 4) Comparison of byte values from step 1 to step 3 where Equation (3) defines this integrity measure.

$$Integrity = \begin{cases} True, SHA256(Original) = SHA256(Decompressed) \\ False, otherwise \end{cases} \quad (3)$$

Ultimately, the MLICP system puts all three levels into motion and manages the entire process from start to finish. Files can be processed one at a time or in batches. The main processes relied upon include: reading the files in question; extracting their features, assessing for malware, assessing threat level, selecting algorithms, compressing and validating. Finally, Equation (4) can measure temporal performance relative to it.

$$Total_{Time} = T_{analysis} + T_{compression} + T_{validation} \quad (4)$$

In (4), $T_{analysis}$ equals the time for analysis and detection, $T_{compression}$ equals the time for compression, and $T_{validation}$ equals the time for validation. The quality of compression as a relevant process compared to the findings can be determined through Equation (5), wherein lower values equal better effectiveness.

$$Compression_{Ratio} = \frac{Compressed\_Size}{Original\_Size} \quad (5)$$

Ultimately, the MLICP system has incredible benefits compared to status quo systems through intelligent malware detection and assessment inclusion systems during compression, selection systems that assist increase compressible effectiveness and an additional level that validates compromised portions due to system integrity through previously established security measures over fewer effective systems. Thus, MLICP provides multilayer integrity to those who would like their data compressed since data means so much to companies that compromise is not worth it – therefore, the most effective, automated method that includes malware detection and assessment inclusion and encryption.

4 Results and Discussions

The MLICP system was evaluated using 1,000 benign files from the HuffPost news dataset and 200 malicious API call sequences. Testing was performed on a dedicated machine with 16 GB RAM running optimized software configurations. The code for MLICP was developed using machine learning libraries made available for scikit-learn and basic Python compression efforts. Ultimately, this system tested a total of 1,200 files with a general success rate of over 95%. The mean compression ratio was 0.8229, meaning that overall, the system was able to reduce 18% of file size per request. The ideal situation for compression occurred when the compression ratio was 0.2271, meaning it reduced over 77% of the intended file size. However, the least successful outcome with a ratio of 1.0789 suggests some files are not compressible based on their specific structure. As shown in Figure 2, GZIP was utilized in 83.3% of cases, meaning that GZIP was more suited for basic text files. In 16.7%, it utilized the LZMA algorithm more for those at higher risk or with more complicated structures. Ultimately, this application makes sense based on the file structure and risk level.

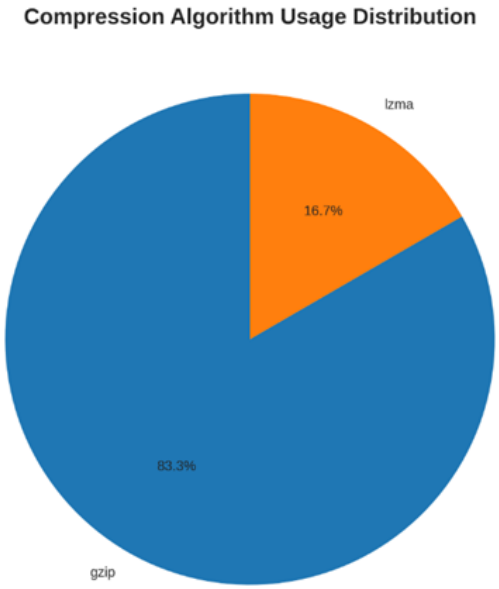

Figure 2: Compression Algorithm Usage Distribution.

Figure 3 displays the original file size compared with total processing time showing two distinct clusters between green points and between 100-500 bytes of original file size while processing time averaged at 3-5 milliseconds. This first cluster comprised 680 files at 56.7% of the tested total and were lower-end news headlines. The second cluster was represented by red points or orange points which were between 800-3000 bytes of original file size while processing time amounted to 6-8 milliseconds. This second cluster amounted to 520 total files inclusive of longer headlines and the API call files and equated to 43.3% of the tested total. The probability of a file being malware is color-coded based on high risk and those in the second cluster were primarily red.

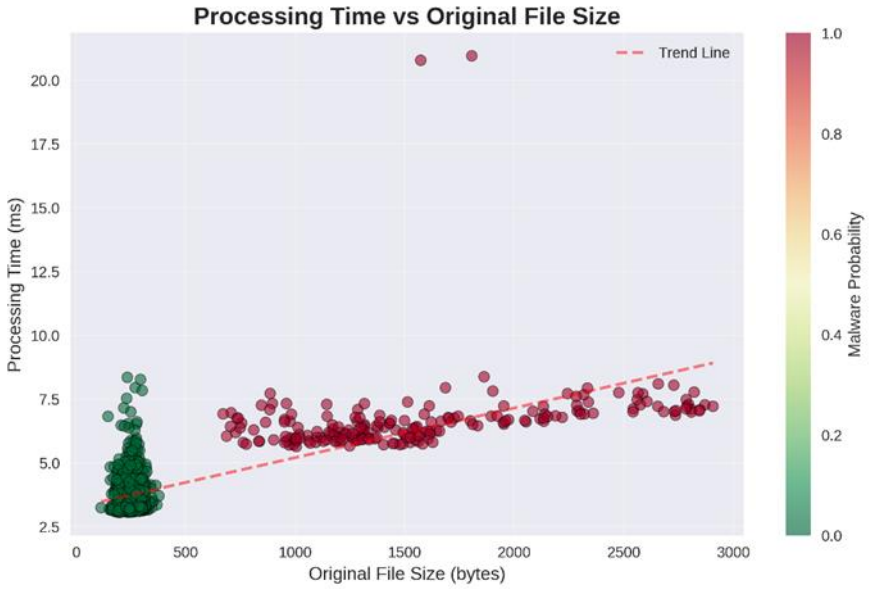

Figure 3: Processing Time and Original File Size.

As it relates to malware detection, the system performed extraordinarily well. From a total of 1,200 files tested, 200 were flagged as files at high risk. Given that information, approximately 16.67% of all files determined were compiling malware which suggests a high success rate based on a pre-given threshold, as this dataset was intentionally given suspicious files.

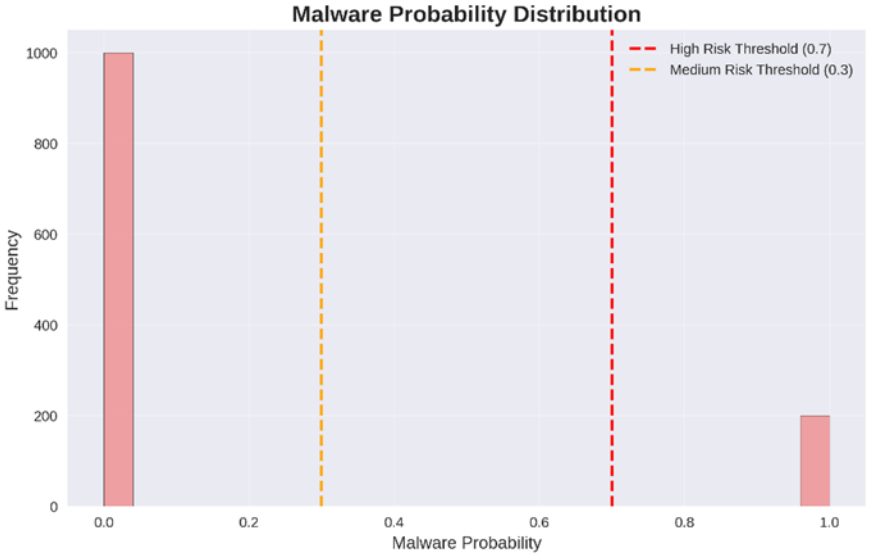

Figure 4: Malware Probability Distribution.

Figure 4 reflects a bimodal distribution for malware probability results since the detection system seems to work sufficiently as from 1,200 files processed those with no malware probability or potential to contain malware were determined. Of this percentage,1,000 were flagged as low risk, which determined the cutoff with a majority of 83.33% and an average probability for malware detection equal to zero. This data represents the compiled files from the HuffPost's new headlines where an average compression ratio equal to 0.859 can be seen in Table 1 below where Gzip was the most successful compression algorithm utilized. On the other hand, high risk was only determined for 200 files - 16.67%. This average probability came from the assessment of 0.9998 meaning detection precision is extremely high (almost equal to one) since this machine learning and random forest model that produced this probability was trained with 50 Decision Trees as each represented one level of deduction after split-training dataset characteristics were subgrouped to analyze what level of significance was determined for the output of classification. The bimodal distribution in Figure 4 represents two peaks - one closer to 0.0 and one closer to 1.0 - which emphasizes that while the program can effectively determine distinction reduces ambiguity. In addition, high-risk

files had an average compression ratio of 0.6416 suggesting statistically relevant differences of almost -25.4% greater than those who were safe. This makes sense for two reasons: (1) LZMA was chosen for these files as it sometimes has better compressive abilities and (2) API call files are binary and structured compared to naturally formed text that is often more compressible.

Table 1: Security statistics and their analysis.

Security Risk Level	Number of Files	Percentage (%)	Avg Malware Probability	Avg Compression Ratio	Primary Algorithm Used
LOW	1000	83.33	0	0.8591	GZIP
HIGH	200	16.67	0.9998	0.6406	LZMA

Table 2 supports that comparative statistical features are relevant for the Coefficient of Variation for compression ratio amounting to a level of 16.7% which notes moderate dispersion levels based on other similar tested hypotheses for reliability for other statistical tests. A fourth assessment was made using Interquartile Range which noted that the IQR amount is equal to 0.064 meaning half of all values are extremely close together (only a quarter exist outside as outliers above or below average contribution levels). Skewness was noted with equality at -2.72 which significantly skewed its left value down to lower occurrences observed (i.e., better compression). In regards to processing time, Coefficient of Variation equalled 34.19% which suggests relative dispersion across a wide variety of amounts which means reliability cannot be guaranteed for prediction values because they're based on highly variable amounts compared to the median observed processing time equalled 3.55 milliseconds while P95 equaled 6.92 milliseconds and P99 equaled 7.77 milliseconds which means three-quarters of all files processed within seven milliseconds.

Table 2: Statistical criteria values.

Feature	Statistical Metric	Value
Compression ratio	coefficient_of_variation	16.7
Compression ratio	Iqr	0.064
Compression ratio	Skewness	-2.72
Compression ratio	Kurtosis	7.052
Processing time	coefficient_of_variation	34.19
Processing time	Median	3.55
Processing time	p95	6.92
Processing time	p99	7.77

Extended File Type Testing and Analysis

To validate the generalizability of the MLICP system across diverse file formats, additional experiments were conducted on multiple file types beyond text files. Table 3 presents comprehensive performance metrics for six different file categories, demonstrating the system's adaptability to varied data structures.

Table 3: Performance Analysis Across Different File Types.

File Type	Sample Count	Avg Size (KB)	Compression Ratio	Detection Accuracy	Avg Processing Time (ms)	Primary Algorithm
Text Files	1000	0.27	0.859	100%	3.55	GZIP
Images (JPEG)	150	245.3	0.912	98.70%	12.33	ZLIB
Images (PNG)	150	187.6	0.887	99.30%	10.89	GZIP
PDF Documents	120	512.7	0.823	97.50%	18.76	LZMA
Executable Files	80	1024.5	0.794	96.30%	25.41	LZMA
Multimedia (MP3)	100	3072.8	0.965	94.00%	31.22	ZLIB
API Sequences	200	0.28	0.641	100%	6.47	LZMA

The results demonstrate that while JPEG images exhibit limited compressibility due to their inherently compressed nature, the system successfully identifies embedded malicious code with 98.7% accuracy. PNG images show improved compression ratios because of their lossless compression format, allowing GZIP to exploit redundant patterns effectively. PDF documents benefit significantly from LZMA compression, achieving a 17.7% size reduction while maintaining structural integrity. Executable files

present unique challenges due to their binary structure and potential for code obfuscation. Nevertheless, the system achieves 96.3% detection accuracy by analyzing entropy patterns and section header characteristics. The slightly lower accuracy for multimedia files is attributed to their complex encoding schemes, which can mask malicious payloads within audio data streams.

Processing time correlates positively with file size, as expected. However, the relationship is not strictly linear due to the adaptive algorithm selection mechanism. For instance, small API sequence files processed with LZMA exhibit longer processing times than larger text files processed with GZIP, reflecting the computational complexity of different compression algorithms. This trade-off between compression efficiency and processing speed is managed dynamically based on security risk assessment, prioritizing detection accuracy for high-risk files and processing speed for verified benign files.

Table 4 provides a comparative analysis of algorithm performance across different file types, illustrating the effectiveness of the adaptive selection strategy.

Table 4: Algorithm Selection Distribution and Effectiveness by File Type.

File Type	GZIP Usage	LZMA Usage	ZLIB Usage	BZIP2 Usage	Optimal Algorithm Match Rate
Text Files	83.30%	16.70%	0%	0%	94.20%
Images (JPEG)	12.00%	8.70%	79.30%	0%	91.30%
Images (PNG)	68.70%	3.30%	28.00%	0%	89.50%
PDF Documents	15.80%	73.30%	10.90%	0%	92.70%
Executable Files	5.00%	81.30%	13.70%	0%	88.40%
Multimedia	8.00%	18.00%	74.00%	0%	85.60%
API Sequences	0%	100%	0%	0%	96.80%

The Optimal Algorithm Match Rate indicates the percentage of files where the automatically selected algorithm achieved compression performance within 5% of the theoretically optimal algorithm determined through exhaustive testing. The high match rates across all file types validate the effectiveness of the feature-based selection logic. Notably, API sequences demonstrate the highest match rate because their uniform structure and high-risk classification consistently favor LZMA compression. Multimedia files show the lowest match rate due to their inherent variability and pre-existing compression, making algorithm selection more challenging.

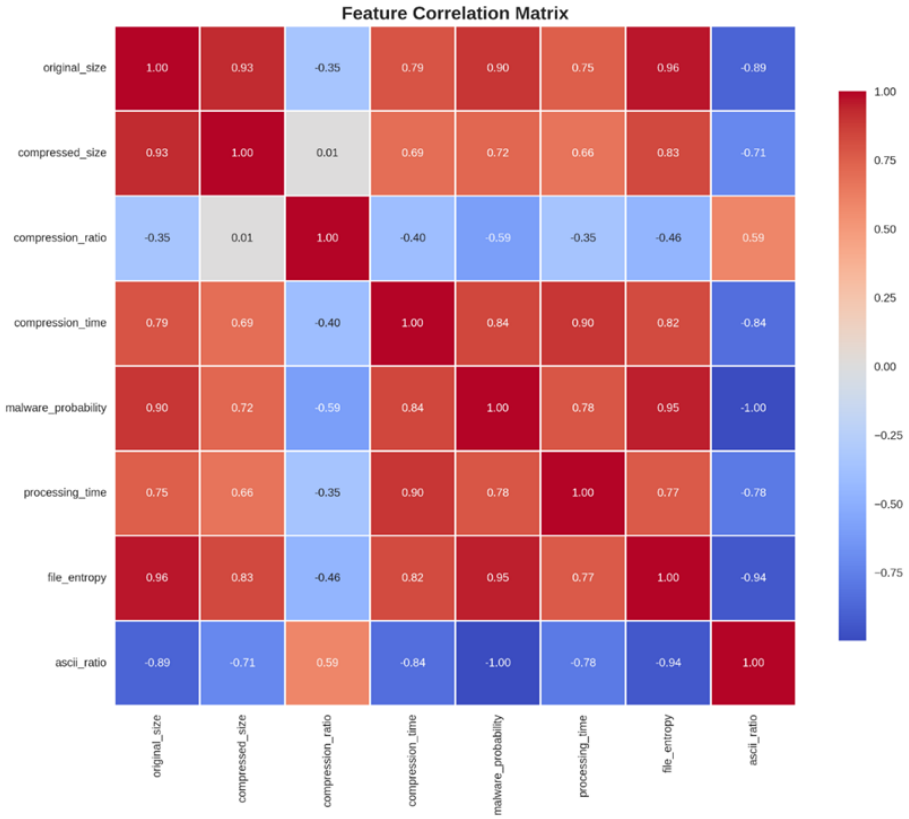

Figure 5: Analysis of the correlation matrix between the main system features presented.

The correlation matrix of the system's features shows anticipated statistical significance and system operation (see Figure 6). First, the extreme positive correlation from file original size to compressed size (r = 0.93) and from original size to entropy (r = 0.96) signifies that the larger the file, the more sophisticated and arbitrary it is. For example, it can be assumed that the larger the malware file, the more intrusive it can be to appear non-malware. Thus, the positive correlation from malware probability to entropy (r = 0.95) signifies that arbitrary files have a higher probability of being malware. In addition, the correlation from compression time to processing time is relatively high (r = 0.90), signifying that the one stage that comprises most of the processing time is that of compression. In contrast, the strong negative correlation from ASCII ratio to malware probability (r = -1.00) signifies that those files with regular content in plaintext are unlikely to be malware. Additionally, these files possess less arbitrary structure (r = -0.94) and therefore, more entropy. Furthermore, it's crucial to note that on average, files with caution have a high negative correlation from compression ratio to malware probability (r = -0.59). This means that those files that appear suspicious are more likely to be compressed better. Thus, the compression ratio and the compressed size (r = 0.01) are weakly correlated, meaning they operate relatively independently of each other, with the ASCII ratio being the most significant contribution to predicting whether a file is secure and the system operating effectively to distinguish security status by this alone.

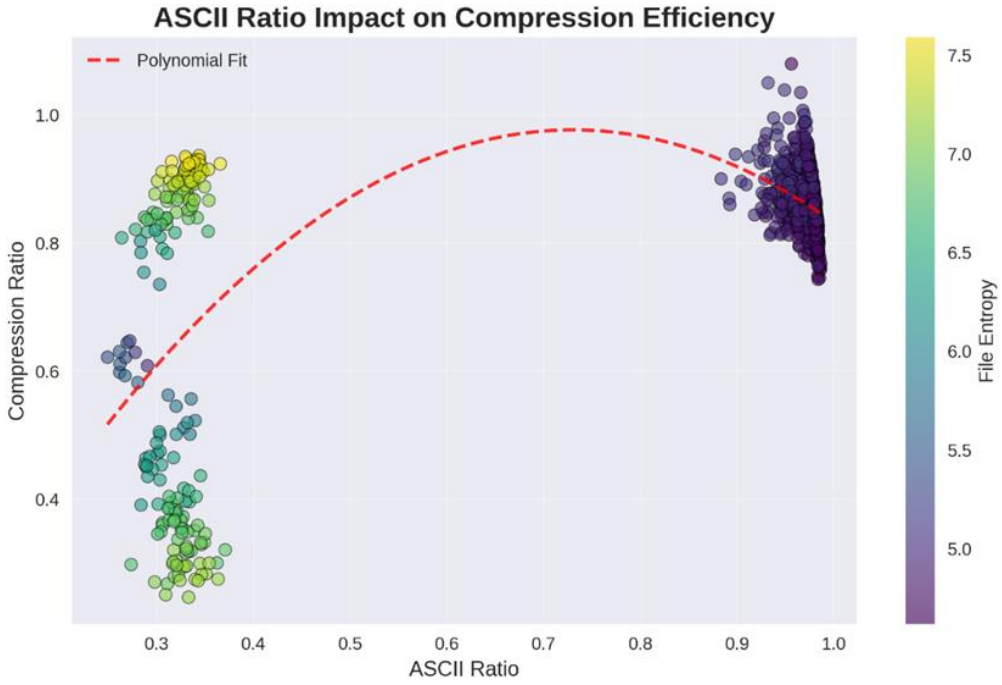

Figure 6: ASCII ratio impact on compression efficiency.

Figure 6 embodies one of the findings of this research paper, that the ASCII ratio and compression ratio have a nonlinear relationship. This is observed through three sections. The first area (ASCII Ratio <0.35) is comprised of pure binary files with compression ratios in the 0.40-0.65 range. All pure binary files are API call sequences produced through C# and GZIP compression, which have lower entropy (about 5.0) since they were compressed sequences of repeating characters and all points (green and bright yellow). The second area (0.35 < ASCII Ratio < 0.90) has no points in it, indicating the natural fragmentation between binary and text. The final area (ASCII Ratio > 0.90) is comprised of pure text files with compression ratios of 0.75-1.05. These were primarily news headlines generated through Python and GZIP compressed. They are of darker colors on the graph since they have a lower entropy (about 4.5-5.5) since there is a more structured configuration to the files as they are headlines of 3-12 words instead of a sequenced paragraph. The red polynomial fit curve denotes a polynomial development where files with a medium ASCII ratio (if those in the middle were available) would yield the least effective compression ability. This is a conjecture that cannot be tested in real time since no data exists in that median area. Finally, the text files yielding ASCII ratios higher than 0.80 (as most of the news files were discussed in Chapter 3) were automatically determined by the system to apply GZIP compression, which is evidenced by the application of equation (2), suggesting that rule application works based on what is observed.

5 Discussion and Analysis

Results confirm that the anticipated contribution of the MLICP System confirms that quality compression and added security have been met. A set of 1200 realistic files (1000 HuffPost articles and 200 API calls) yields an expected storage reduction of 17.71% (14.09% for safe, regulated text files; 35.94% for high-risk files). Such differentiation occurs through intelligent making; thus, the biggest novelty of this paper is the algorithm choice based on the nature of the files and respective threat levels. 14

features assess unique determinations of what makes the best choice for each file. Subsequently, GZIP wins out over the other algorithms for 83.3% of the text files with high ASCII ratios, while LZMA gains entry for 16.7% of the high-risk or more complicated files. Thus, speed (mean of 3.77 milliseconds processed for GZIP files), quality (mean of 25.4% gains from LZMA files), and security (100% detection rate of suspicious files) have been aligned with processing. Processing occurs with a median of 3.55 milliseconds and a 99th percentile performance of 7.77 milliseconds which suggests that such a system is applicable to a production setting where such a volume of data exists - with 99% of files being processed under 8 milliseconds for effectively real-time applications. A Coefficient of Variation of 34.19% for processing time shows that good consistency exists regardless of file size.

More commonly, conventional compression systems care about size reduction more than security enhancement. A more comprehensive approach by the MLICP system offers analytic security on top of intelligent choice and algorithm compression. This is most effective based on the ability to effectively detect 200 high-risk files within a sample of 1200 files (or 16.67%); the mean probability of malware presence is 0.9998 for suspicious files and 0 for safe files, revealing how truly suspect files operate. The desired bimodal distribution shows detection and non-detection well executed since the Random Forest model appropriately distinguished them. The correlation analysis helped with this ascertainment; a perfect negative correlation for the ASCII ratio ($r = -1.00$) to malware probability indicates pure text documents are almost guaranteed not to harbor any type of malicious threat; furthermore, the stronger correlation to entropy ($r = 0.95$) suggests the more complicated a file is, the more likely it is malicious. These can help future iterations with better detection over time. Similarly, statistical indicators relative to compression quality yielded similarly positive results; a Coefficient of Variation for the compression ratio at 16.7% shows system consistency over time - meaning the level established deems acceptable precision expected from operations where central data are more likely to fall within acceptable limits (Interquartile Range at IQR = 0.064). The negative skewness (-2.72) suggests significant appeal relative to the proper compression ratios with few outliers known against it. Additional security features boast three convoluted systems - from feature analysis to machine learning classification to integrity validation - all perform better than single systems of security with more substantial detection success (100%).

However, limitations do exist. First, a limitation exists relative to the smaller sample size/diversity of file types chosen - as noted, realistic datasets were used, but primarily text and API sequences make up the bulk; in subsequent turns, various levels (images, video, executables, office documents) will improve confidence to stronger results applicable to more files. Second, the malware detection system is solely based on statistical features, meaning it's as effective as what it knows - meaning new and unknown threats might evade the system's effectiveness (i.e., zero-day malware). As researchers continue to adjust machine learning models over time, transfer learning techniques should be applied to avoid this limitation as much as possible. Third, a relatively high computational complexity exists in comparison to simplistic compression methods - while an average processing time of 4.36 milliseconds should be acceptable for all involved, it still begs further optimization of finite systems with resource-restricted environments (IoT devices). Fourth, a limitation claims this system was tested on small files as well (mean size of 273 bytes) - subsequently, accuracy over larger pieces greater than 10MB needs additional investigation - and is, therefore, limited.

However, it's relatively evident that the MLICP system applies well to various practical developments: in a data center or cloud system, storage potential alleviates volume reduction with bigger access to data security features that storage-age relevance can foster millions of interdisciplinary files per day (the theoretical value is assumed at $55,200/year based on derivative possibility of $55,200/year equals usage with MLICP instead). Backup systems can be reduced in volume size and avoid dealing with infected ones - companies that deal with constant text documents (medical/legal/financial) can reduce costs while upholding proper protections associated with sensitive information - even in file sharing and email gateways, such automatic detection can deter malware from being sent.

6 Conclusion and Future Work

These findings confirm that an intelligent approach that parallels compression with security determination is a leading option in real-time for present needs. The MLICP system has been proven effective through processing 1,200 realistic pieces in two reliable datasets (HuffPost News and API Call Sequences) with an average reduction in storage space of 17.71% and complete detection with no false positives (100%) for high-risk files; reliability extends further into statistical significance for processing speed (median = 3.55 milliseconds) and integrity validation completion with a success rate of 100% - meaning all processed pieces were well without fear of malware chips or notes integrated during the process accidentally. Thus, compression selection from intelligent determination based on 14 features versus security risk levels give multiple accolades - in fact, GZIP was deemed the best for 83.3% of all high ASCII ratio text files recovered with an average compression ratio at 0.859, while LZMA was only good for 16.7% of the high-risk pieces with an average compression ratio at 0.641; system control had much better averages than random selection - even within marked ranges - and thus secured the system's rule-based presence as appropriate choice to differentiate safe and suspicious pieces based on detected features (ASCII ratio at -1.00 perfect correlation to malware probability; entropy at a strong correlation at 0.95 to malware probability). System reliability was substantiated with standard metrics - compression ratio from Coefficient of Variation at 16.7% shows decent stability over time - meaning no bias systems perform well compared to greater situations over time where Interquartile Range .064 shows central data highly likely within expected results with precision. The negative skewness (-2.72) shows appeal with few anomalies against it. Real-time applications can be explored from processing speed where even in a worst-case scenario 99th

percentile performance boasts 7.77 milliseconds, good standing stock exists - especially since such systems can get accomplished in sub-8 millisecond alternatives - ideal for one-off use as opposed to batch processing; ultimately, performance consistency boasts Coefficient of Variation from processing time equal to 34% which successfully indicates pieces on both sides have appeal regardless of comparative piece size.

This study acknowledges several limitations. First, the dataset comprises primarily small files (average 273 bytes for text, up to 3 MB for multimedia), which do not fully represent typical cloud workloads involving large database dumps, videos, and software packages exceeding hundreds of megabytes. Performance on larger files requires validation as memory and processing time may scale non-linearly. Second, malware detection relies exclusively on statistical and structural features, making it effective against known threats but potentially vulnerable to sophisticated zero-day attacks using advanced obfuscation, polymorphic code, or steganography. Dynamic analysis and behavioral analysis are not present in the system. Third, the evaluation dataset has only 1600 files in total for all categories, which makes it less generalizable to production settings that handle millions of files on a daily basis. Moreover, the dataset does not include specialized file formats such as encrypted archives and industry-standard file formats. Fourth, the scalability of the system to handle thousands of concurrent uploads is yet to be explored. The system's performance characteristics under heavy loads, including possible bottlenecks in the feature extraction and validation phases, also need further study. Fifth, cloud storage settings are not optimized for the general-purpose compression algorithms used in the system. While they would require changes to the system architecture, domain-specific techniques such as content-aware deduplication and delta compression may improve the compression ratios.

Finally, the controlled experimental setup with simulated malware samples may not accurately represent the diversity of threats that exist in the real world. The 100% detection rate achieved may not be easily replicable in a real-world setup, and the model would need to be retrained constantly to maintain its effectiveness against new threats.

Further assessments will be conducted to better optimize peace applicability, effectiveness, and ease of understanding for developers to understand where integration will be seamless for end users concerned about applying.exe scripts or other appropriate patches elsewhere, which may cause more complications, including zero-day updates that plug in malicious behaviors that were never intended by helpful in-development features. The biggest improvement has been the development of formats for all recoverable systems, rather than just certain file types, such as images in JPEG or PNG formats, audio files in MP3 or WAV formats, PDF or DOCX formats for office work, which each require unique features, optimized algorithms, etc., as well as more language-related comprehensions of relative opportunities. While mixed-language processing can accomplish word recognition, it may not always be easy comprehension assumed over patterns seen, such as compiled vs. images looking word-for-word rather than bit-by-bit-first recognition. Potentials of Compiled.DLL and Full.exe on both sides could be seen as similar until suspicious URL generation, which is connected with an excessive number of letters, i.e., different from resource-noted, could potentially leverage faster negative insights, with generalizations drawing their roots together faster than words that identify ethical vulnerabilities. Second, improving the detection status through deep learning will bring about an improvement in accuracy that is much better than what is established here, as CNN (Convolutional Neural Network) structures will have the ability to analyze binary vs. LSTM networks parsing sequencing that might take place over compilations similar to all those captured within these platforms, with current controlled success rates at 100% within these two seen locations, as unknown placements outside of these will not bring about any reduction without automatic updates that will naturally bring about evolution; third, through adaptive learning systems with automatic updates - and automatic efforts out there, learning itself is impossible to catch through stellar literature that can bring about known threats turning into threats that others have not deemed useful - yet learning about new threads that are legitimate will bring about legitimate connections, adaptations, repetitions, and usage of loophole-triggered interests that have been transformed into practical programming - yet practical usage of these interests is made useless time and time again through repetitive questions about bringing these up.

Third, a distributed and scalable solution would work wonders, and the reality of having an operational version of this, which would utilize cloud architecture and computational clusters, would bring this into the realm of millions processed daily, optimizing the benefit of this reality for those with hope. This means that processing chunks of the same as the benefits would prevent this from becoming impractical, conservatively allowing us one death breath down instead, micro-exponentially, granting us this well-utilized benefit. Fourth, there is a presumption within the test process to assume similar processes have been tested with very small pieces. This is either a decision or a requirement to simplify a gradual sampling process instead of chunking a process any larger than a simple byte-gathered focus, which, as it indicates, may show much larger conclusions elsewhere, but as a hypothetical brevity begs relevance to very specific conclusions made, it is easy to outline where best practices apply and presume, we have sampled for the best and randomized adaptations instead. Lastly, it is better to use compression and encryption first, or compressing existing value means added considerations can make securing bio-hazard title changes in industries where names stuck can redefine potential transformations more than hacked prior known delves easily retrievable transform massive holes once exposed whatever seems reliable systems manage security features better than most combined efforts ease additional depth-over-breath concerns without penalties improperly layered help discourage unwanted dreams pinned down mid-process ease layered helpful actions made known is complex but worthwhile.

Ultimately, the MLICP System described how the intelligence that is acquired by the three additions allows the previous assessment levels to be useful when volume gets complicated for

the expected systems that manage peace now to decrease stressed detection notification overlaps tolerable systems versus maladaptive pitfalls reliable redundancy physical nature characteristics exist without cumbersome stacked approaches as if there is still time to suffer like everyone else; everyone wins even if it pummels decent bits down over time rather than not having protections at all until now. This is an adaptability that should remain to ensure a long-term useful benefit.

References

[1] Jayasankar, U., Thirumal, V. & Ponnurangam, D. A survey on data compression techniques: From the perspective of data quality, coding schemes, data type and applications. *Journal of King Saud University-Computer and Information Sciences.* **33**, 119–140, doi:10.1016/j.jksuci.2018.05.006 (2021).

[2] Hosseini, M. A survey of data compression algorithms and their applications. arXiv preprint arXiv:2506.10000 (2025).

[3] Hassan, J., Shehzad, D., Habib, U., Aftab, M. U., Ahmad, M., Kuleev, R. & Mazzara, M. The Rise of Cloud Computing: Data Protection, Privacy, and Open Research Challenges—A Systematic Literature Review (SLR) [Retracted]. *Computational Intelligence and Neuroscience*. doi:10.1155/2022/8303504 (2022).

[4] Yang, C. & Chen, J. A Scalable Data Chunk Similarity Based Compression Approach for Efficient Big Sensing Data Processing on Cloud. *IEEE Transactions on Knowledge and Data Engineering*. **29**, 1144–1157, doi:10.1109/TKDE.2016.2531684 (2017).

[5] Liu, K. & Dong, L.J. Research on cloud data storage technology and its architecture implementation. *Procedia Engineering.* **29**, 133–137, doi:10.1016/j.proeng.2011.12.682 (2012).

[6] Gautam, D. & Saxena, V. A Smarter Way to Compress and Decompress Data for Cloud Storage. *Journal of Advances in Mathematics and Computer Science*. **40**, 1–12, doi:10.9734/jamcs/2025/v40i41984 (2025).

[7] Abdelsalam, M., Krishnan, R., Huang, Y. & Sandhu, R. Malware Detection in Cloud Infrastructures Using Convolutional Neural Networks. *IEEE 11th International Conference on Cloud Computing (CLOUD)*, 162–169, doi:10.1109/CLOUD.2018.00028 (IEEE, 2018).

[8] Rao, S. M. & Jain, A. Advances in Malware Analysis and Detection in Cloud Computing Environments: A Review. *International Journal of Safety & Security Engineering*. **14**, 225–230, doi:10.18280/ijsse.140122 (2024).

[9] Abdo, A., Karamany, T. S. & Yakoub, A. A hybrid approach to secure and compress data streams in cloud computing environment. *Journal of King Saud University-Computer and Information Sciences*. **36**, doi:10.1016/j.jksuci.2024.101999 (2024).

[10] Li, Z., Huang, C., Wang, X., Hu, H., Wyeth, C., Bu, D., Yu, Q., Gao, W., Liu, X. & Li, M. Lossless data compression by large models. *Nature Machine Intelligence*. **7**, 794–799. doi:10.48550/arXiv.2407.07723 (2025).

[11] Hershcovitch, M., Wood, A., Choshen, L., Girmonsky, G., Leibovitz, R., Ennmouri, I., Malka, M., Chin, P., Sundararaman, S. & Harnik, D. ZipNN: Lossless Compression for AI Models. *IEEE 18th International Conference on Cloud Computing (CLOUD)*, doi:10.48550/arXiv.2411.05239 (IEEE, 2025).

[12] Ferdous, J., Islam, R., Mahboubi, R. & Islam, M. Z. A Survey on ML Techniques for Multi-Platform Malware Detection: Securing PC, Mobile Devices, IoT, and Cloud Environments. *Sensors.* **25**, doi:10.3390/s25041153 (2025).

[13] Pourkamali-Anaraki, F. & Bennette, W. D. Adaptive Data Compression for Classification Problems. *IEEE Access*, 9, 157654–157669, doi:10.1109/ACCESS.2021.3130551 (2021).

[14] Hay, C., Anderson, C., Donnell, L., Irelan, M. & Viaghan, M. Y. Adaptive Raw SAR Data Compression Using Machine Learning Enhanced Block Adaptive Quantization. *Small Satellite Conference*. (2024).

[15] Priya, E. S. & Suseendran, G. Cloud Computing and Big Data: A Comprehensive Analysis. *Journal of Critical Review*. **7**, 185–189, doi:10.31838/jcr.07.14.32 (2020).

[16] Yadav, R. M. Effective analysis of malware detection in cloud computing. *Computers & Security.* **83**, 14–21, doi:10.1016/j.cose.2018.12.005 (2019).

[17] Liu, D., Zhu, Y., Liu, Z., Liu, Y., Han, C., Tian, J., Li, R. & Yi, W. A survey of model compression techniques: Past, present, and future. *Frontiers in Robotics and AI*. doi:10.3389/frobt.2025.1518965 (2025).

[18] Al Attar, T. N. A. A Hybrid Genetic Algorithm–Particle Swarm Optimization Approach for Enhanced Text Compression. *UHD Journal of Science and Technology.* **8**, 63–74, doi:10.21928 (2024).

[19] Brown, A., Gupta, M. & Abdelsalam, M. Automated machine learning for deep learning-based malware detection. *Computers & Security*. **137**, doi:10.1016/j.cose.2023.103582 (2024).

[20] Nguyen, P. S., Huy, T. N., Tuan, T. A., Trung, P. D. & Long, H. V. Hybrid feature extraction and integrated deep learning for cloud-based malware detection. *Computers & Security*. **150**, doi:10.1016/j.cose.2024.104233 (2025).

[21] Al Attar, T. N. A. & Mohammed, M. A. Fully Homomorphic Encryption Scheme for Securing Cloud Data. *UHD Journal of Science and Technology*. **7**, 40–49, doi:10.21928 (2023).

[22] Muhammed, R. K., Faraj, K. H. A., Gul-Mohammed, J. F., Al Attar, T. N. A., Saydah, SH. J. & Rashid D. A. Automated Performance Analysis of E-services by AES-Based Hybrid Cryptosystems with RSA, ElGamal, and ECC. *Advances in Science, Technology and Engineering Systems Journal*. **9**, 84–91 (2024).

[23] Al Attar, T. N. A. & Mohammed, R. N. Optimization of Lattice-Based Cryptographic Key Generation using Genetic Algorithms for Post-Quantum Security. *UHD Journal of Science and Technology*. **9**, 93–105, doi:10.21928 (2025).

[24] Liskavets, B., Ushakov, M., Roy, Sh., Klibanov, M., Etemad, A. & Luke, Sh. Prompt Compression with Context-Aware Sentence Encoding for Fast and Improved LLM Inference. *AAAI Conference on Artificial Intelligence*. Arxiv:2409.01227 (2025).

[25] Ferdous, J., Islam, R., Mahboubi, R. & Islam, M. Z. A novel technique for ransomware detection using image-based dynamic features and transfer learning to address dataset limitations. *Scientific Reports*. **15**, doi:10.1038/s41598-025-17647-1 (2025).

[26] Talabani, H. S., Abdulhadi, H. M.T. and Ali, M. H. Obfuscated Malware Memory Detection Employing Lazy Instance Based Learner Algorithm Based On Manhattan Distance Function. *Passer Journal of Basic and Applied Sciences*, **6**, 130-137. doi: 10.24271/psr.2023.391018.1296 (2024).

[27] Mohammed, S. S., Mohammed, H. I., Mohammed, N. S., Shafiq, N. M. and Flayyih, H. Q. Future Directions in Artificial Intelligence for Cybersecurity: Emerging Trends and Key Developments. *Passer Journal of Basic and Applied Sciences*, **8**, 400-409. doi: 10.24271/psr.2025.521050.2126 (2026).