**Вісник Тернопільського національного технічного університету**
**https://doi.org/10.33108/visnyk_tntu**
Scientific Journal of the Ternopil National Technical University
*2024, № 1 (117) https://doi.org/10.33108/visnyk_tntu2025.01*
*ISSN 2522-4433. Web:* ***visnyk.tntu.edu.ua***

**UDC 004.632**

# UTILIZING REINFORCEMENT LEARNING TO OPTIMIZE DATA ARCHIVING STRATEGY FOR APPLICATION SERVER

**Andrii Harasivka; Anatolii Lupenko**

*Ternopil Ivan Puluj National Technical University, Ternopil, Ukraine*

***Abstract.*** *Efficient data compression is a critical component of modern data backup systems, particularly in environments with diverse file types and different performance features. Backups safeguard critical application server data against unexpected failures, such as hardware malfunctions, software bugs, or cyberattacks, ensuring business continuity. Many industries maintain secure data backups to meet legal and regulatory requirements, ensuring loyalty to data protection and privacy laws. However regular backup solutions are often unable to adapt effectively to the changing data of the application server. This paper proposes a novel reinforcement learning (RL) approach using the Proximal Policy Optimization (PPO) model to dynamically optimize data archiving strategy, which is part of backup systems. The model is trained to predict the most efficient combination of compression parameters based on the attributes of files in the target file system. By learning from file-specific observations and rewards, it adapts to work in a backup-specific environment to minimize consumed disk storage and backup time. This adaptive approach enables real-time decision-making tailored to workload variations of the application server environment. The proposed solution performs backup operations across various file types and configurations for the learning phase, where the model evaluates and adjusts policy to maximize its efficiency. To evaluate the results another client should perform all possible combinations of action parameters to determine all possible observations and rewards. The rewards are compared to decisions made by the proposed solution to ensure PPO model has correct and best possible predictions during the evaluation phase. This study highlights the potential of RL in automating and optimizing data backup tasks, providing a scalable solution for high-performance systems or environments with frequent data writes. The results obtained contribute to improving the software backup systems and DevOps specialists' work and reduce disk storage consumption and time elapsed for backup tasks for the application server.*

***Key words:*** *data backup, machine learning, proximal policy optimization, backup strategy, compression solution, reinforcement learning, application server.*

*https://doi.org/10.33108/visnyk_tntu2025.01.018* *Received 13.11.2024*

## 1. INTRODUCTION

Data backup is a main element of modern environments, ensuring that critical files and systems can be restored in case of data loss. Data is exposed to numerous risks in the digital world, including cyberattacks [1], natural disasters, hardware failures, software bugs or accidental deletions. Amount of various devices (shared servers, personal PCs, or IoT devices) that store the data and are connected to public networks increases every year [2]. Without an effective backup strategy, organizations and individuals face the potential damage or complete loss of irreplaceable data, which can result in financial losses, legal consequences, and spoiled reputations. Backup provide a safeguard against such risks, enabling swift recovery and continuity in operations, whether for businesses ensuring uptime or individuals protecting personal memories and sensitive information. One of the most important parts of data backup is the corresponding archiving strategy, which should optimize storage utilization and handle a sufficient level of performance [3].

As data volumes grow and storage environments become more complex, archiving strategies must evolve to handle the diversity and scale of modern server environments. With the increasing dependence on cloud storage and hybrid architectures [4], automated and intelligent backup systems play an important role in reducing costs, improving efficiency, and ensuring data security across distributed environments.

Traditional rule-based backup solutions, like Acronis or Veritas NetBackup, rely on static heuristics [5], which are often unable to adapt effectively to diverse file types, varying

 *Corresponding author: Anatolii Lupenko; e-mail: Lupenko.san@gmail.com*

storage conditions, and evolving performance requirements of servers. However, as described and proven in [6] and [7], a well-trained machine-learning model could avoid static behaviour and dynamically adapt to various conditions of the environment [8]. Due to the various performance of the application server and the increasing amount of data produced: database, logs, cache – the new solution should know how to backup and compress different amounts of incoming files [9]. Due to the fact that the environment will continuously change – the proposed solution should also adapt via trial and error. The model of reinforcement learning type of ML will be suitable for such dynamic environments [7]. The latest developments in one of the most popular machine learning library stable-baselines3 [10] include: AC2, PPO, RecurrentPPO and TRPO, these algorithms support all types of incoming data for training. PPO was selected for its robustness and efficiency in handling continuous and discrete action spaces, as well as its proven ability to balance exploration and exploitation during training [10].

The purpose of the new solution will be to achieve the best results of compression data which will be compared to results of static compression solution.

## 2. EXPERIMENTAL SOLUTION

The use of machine learning (ML), specifically the Proximal Policy Optimization (PPO) reinforcement learning algorithm, was chosen to reduce the complexity and perform optimization of selecting best parameters for the archivation strategy for output data of the application server – to maximize efficiency, minimize storage consumption, and reduce backup time, outperforming static approaches and providing a scalable, adaptive solution for modern data backup challenges.

The environment in ML is the external system or simulation in which the agent operates and interacts. A single operation in the environment is called a step. It defines the rules of interaction, the current state, and how the agent's actions influence that state. After step completion environment provides observations to the agent and evaluates its actions by returning a reward or penalty [11]. By providing different files for an agent, the environment should emulate real-world conditions and serve as a training ground for the learning process of dynamic optimization of backup strategy.

For this experiment, the environment will be an instance of a separate application entity that will simulate a system where the agent decides how to perform a data backup based on given file attributes and system constraints. It should know all possible actions to initialize the action space of PPO model, in our case, it will be a backup type (full, incremental, or differential), compression method (Deflate, Deflate64. BZip2, LZMA, PPMd, Copy), compression level (0–9) and dictionary size (64 kB – 1536 MB). As the environment perform a single action per step, so backup type will be always full, which means the whole source (file or directory) will be backuped up to the target directory. Aside from action space, model should have observation space of predefined, fixed size – it is the range of observable values that will be used by PPO to create policy. Such range defined by environment limitations, e.g. maximum allowed time or file size, in our case it will be a box with normalized values from 0 to 1. The environment in RL should perform the next functions: reset and step.

The reset function in the environment performs the restart of the model's observation and reward to 0's if there are no files provided (e.g. due to learning). If there are files provided (e.g. due to evaluation) – reset will save file info, like file extension and file size so a model could make an actual prediction based on this file info.

The step function of the environment represents the part of the episode - single operation (in our case - backup task). The episode means a set of steps, so in our case episode will contain steps for all incoming files for learning. The behavior of step will vary depending on the agent's actions from action space: backup type and compression parameters. These actions will be passed as parameters to perform an actual archivation process by NanaZip external software. The environment will start the NanaZip command line interface in a subprocess with arguments: source, target, compression method, compression level, and dictionary size.

To evaluate a performed step environment needs to collect additional information about the step results. It will contain compression ratio, elapsed time, and success flag. The success flag indicates if the action performed successfully and no error occurred. Compression ratio is a measure of the effectiveness of a compression process, expressed as the ratio between the size of the uncompressed data and the size of the compressed data. It quantifies the amount of space saved during compression and is defined by the formula:

$$compression\ ratio = \frac{size\ of\ uncompressed\ data}{size\ of\ compressed\ data}$$

These values will be included in a reward to return as a model`s state and guide the agent's learning, balancing objectives like minimizing compression time and storage usage while maximizing compression efficiency. The reward could be defined by the formula:

$$R = S + T + C,$$

where:

- $R$: Total reward.
- $S$: Success reward or penalty, defined as: $S = \begin{cases} +1.0\ , if\ successfull \\ -1.0\ \ \ \ \ \ , if\ failed \end{cases}$
- $T$: Time penalty, defined as: $T = -3 * backup\ time\ in\ total\ seconds$
- $C$: Compression reward or penalty, defined as:

$$C = \begin{cases} 0.1 * compression\ ratio\ , if\ compression\ ratio < 1 \\ 0.3 * compression\ ratio\ , if\ compression\ ratio \geq 1 \end{cases}$$

Alongside calculation reward, the environment should provide observation from the performed operation [12]. Observation should be defined as array of values that will fit into the observation space of the model. For the proposed solution it will be: normalized file extension, normalized file size, normalized backup time, normalized compression ratio, normalized compression method, normalized compression level, and success flag (1 or 0) enabling the agent to evaluate its actions. Normalized means that the value should be in the range from 0 to 1 so it could be counted into policy calculation.

$$normalized\ compression\ ratio = \frac{compression\ ratio}{\max compression\ ratio\ observed}$$

$$normalized\ file\ size = \frac{file\ size}{\max file\ size\ observed}$$

$$normalized\ compression\ time = \frac{elapsed\ time}{\max elapsed\ time\ observed}$$

$$normalized\ file\ extension = \begin{cases} ext\_map[extension], if\ extension \in ext\_map \\ \frac{N}{N+1} \qquad\qquad\qquad , if\ extension \notin ext\_map \end{cases}$$

All scalar values and formulas used for reward and observation calculation in the environment definition are achieved experimentally by trial and error and should be adjusted during development.

The most important part of ML – the evaluation (predictions) of the model should be performed only after completing the learning phase (training). The evaluation phase contains

of reset and actual predictions made by model for actual data. Usually, data sets are split into learning and evaluation sets which are used to the corresponding phase of ML model use.

## 3. RESULTS AND DISCUSSION

The evaluation of the PPO model's performance in learning optimal parameters for backup strategy involves analyzing key metrics [13] such as elapsed time, compression ratio, and cumulative reward value. By comparing the predicted actions (e.g., compression method and compression level) with their actual outcomes, we can assess the model's ability to minimize compression time while maximizing compression efficiency.

Software development with ML algorithms is iterative and requires a lot of training and adjusting in the development process. The proposed solution was implemented using: Visual Studio Code environment; Python interpreter; Anaconda distribution, which simplifies the managing of Python references; and a set of other packages including stable-baseline3 ML package. Project architecture becomes more complex as the requirements change. It should include many components for work with FileSystem (create/delete directories, write/read files), operating system (read time to measure performance), or call another software (start subprocess with NanaZip). The flow diagram of the proposed solution`s behavior – Fig. 1. To have the ability of restore and continue work without losing progress – ML model have a feature to save model to disk on each learning step, so after restart solution could load previous state of model.

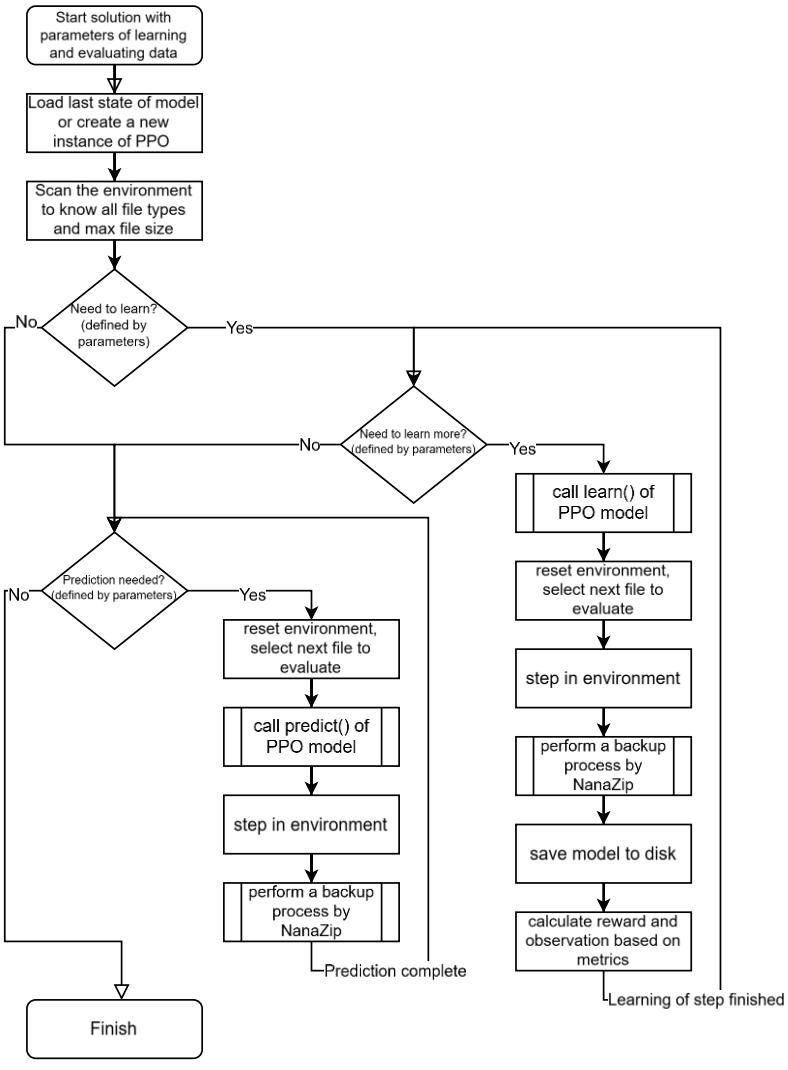


**Figure 1**. Flow diagram of behavior of proposed solution

The proposed solution was trained during ~96 hours in a simulated environment, which includes sample PC (Table 1) and training data set: 23 files (8 images, 7 PDF documents, 2 archives, 3 executable, 3 text files). As our environment will perform compressing the file (an actual backup task) – the smaller the training files will be – the faster the environment will perform a backup task. So, smallest files was selected, Fig. 1.

**Table 1**

Test machine configuration

| *Category* | *Name* | *Specifications* |
|---|---|---|
| Hardware | Main board | MSI X470 Gaming Pro |
| | CPU | AMD Ryzen 7 2700X 3.7 GHz |
| | Hard disk | Samsung SSD 970 EVO 1TB |
| | Network card | Intel (R) I211 Gigabit Network |
| | Memory | 32 GB DDR4 3200 MHz |
| | Graphics | NVIDIA GeForce GTX 1060 6GB |
| Software | Operating system | Windows 10 Pro 22H2 |
| | IDE | Visual Studio Code |
| | Interpreter | Python 3.12 |
| | Python distribution | pip 24.0<br>Anaconda 24.9.2 |
| | Packages installed | Gym 0.26.2<br>NumPy 1.26.4<br>stable-baselines3 2.3.2<br>torch 2.5.1<br>tensorboard 2.18.0 |

Progress of learning of PPO model will be tracked by TensorBoard software, which will analyse the logs produced by the model during learning, logs contains needed metrics [14]. By default, TensorBoard will show few metrics: «rollout/ep_len_mean», «rollout/ep_rew_mean», «time/fps». «ep_len_mean» metrics show the amount of steps per episode, but as there are fixed 8 files per episode and so 1 file per step there will be a straight line – Fig. 2.

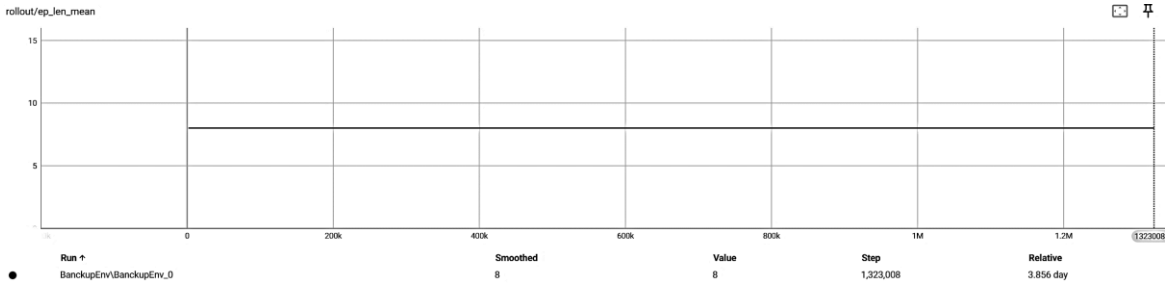


**Figure 2.** Graph of ep_len_mean – average mean of episode length during learning

The metric «ep_rew_mean» refers to the mean episode reward in reinforcement learning, calculated as the average reward obtained by the agent over all steps within an learning. It always starts from 0, when the model do not know about the environment but should increase due to learning, a stable or increasing «ep_rew_mean» signals that the agent is effectively understanding and optimizing the task of the environment. This metric provides a measure of how well the PPO model is optimizing the backup process based on

the defined reward function. On average model have a reward 24.6, Fig. 3. Changing in the environment or its behavior could lead to significant changes of models progress and reward correspondingly.

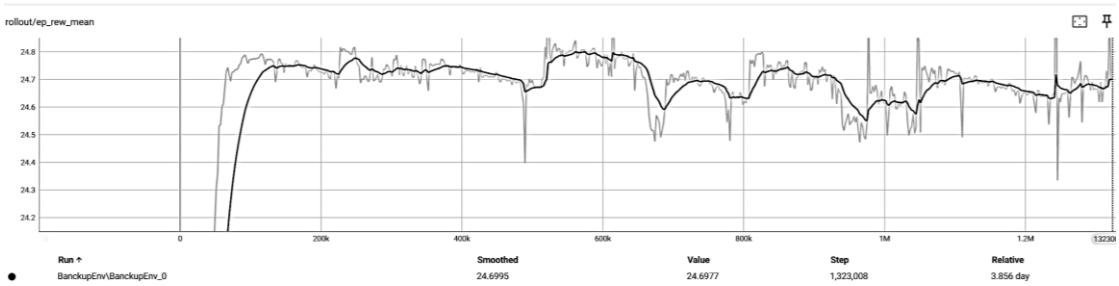


**Figure 3.** Graph of ep_rew_mean – average mean of reward during learning

The time/fps metric represents the training speed of the RL model, measured in frames per second (FPS). It indicates how quickly the agent processes and learns from interactions with the environment. Higher FPS values reflect more efficient training, which can be influenced by factors like the complexity of the environment, computational resources, and model architecture. As the environment is dependent directly on performance of a storage – time/fps have very low resolution – only 8–9 fps, Fig. 4.

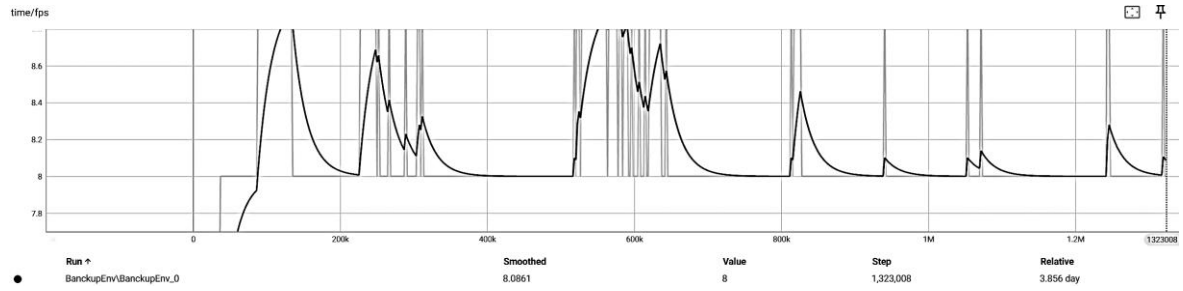


**Figure 4.** Graph of time/fps – amount of frames (actions) per second during learning

The proposed solution perform also console the output of each action made and step performed, so performance could be tracked in editors too. For evaluation were selected another files from actual production-ready server environment, one text file (96934 kB) and one photo (444 kB), Fig. 5. Actual predictions of the solution could be seen both in the console and in resulting files – Fig. 6, Fig. 7, Fig. 8.

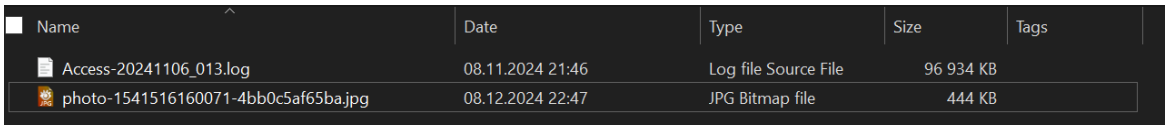


**Figure 5.** Explorer view with input data (text and photo files) for evaluation of proposed solution

```
Access-20241106_013.log           , m: PPMd, c: 8, cr: 36.594 in 3.612711s, r: 2.140, state: [0.8, 1.0, 1.0, 0.9230769230769231, 0.8888888888888888, 1.0, 1.0]
Episode E:\src\StorageAgent\input_data\evaluate\Access-20241106_013.log: Reward = 2.14009120155799
```

**Figure 6.** Prediction in console output of proposed solution about text file – PPMd compression method and 8 compression level

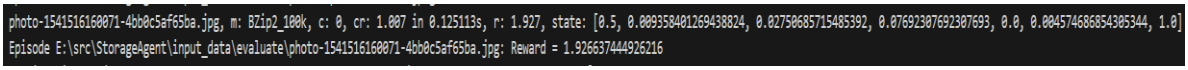


**Figure 7.** Prediction in console output of proposed solution about photo file – BZIP2 compression method with 100 kB dictionary size and 5 compression level

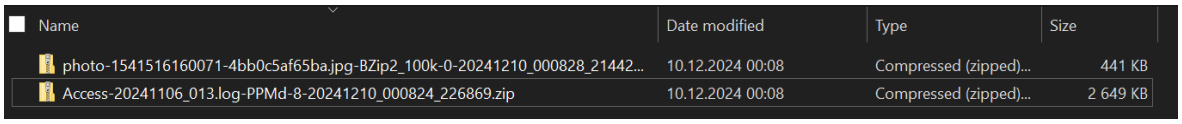


**Figure 8.** Explorer view of output of backup actions made by proposed solution text and photo files

To evaluate the results made by the proposed solution [15], another software client (agent 2) will be introduced, which will perform all possible combinations of backup parameters and write all results to find the best combination of elapsed time and compression ratio. Results made by the client will be written to a CSV file with columns «file name», «compression method», «compression level», «elapsed time» and «compression ratio». Reward column [16] will be introduced to find the best parameters, by the same formula: *2 + -3 * «elapsed_time» + (if «compression ratio» < 1; «compression ratio» * 0.1; «compression ratio»*0.3)*. The best compression parameters will have the highest score. Scatter charts were created to illustrate the result, Fig. 9 and Fig. 10.

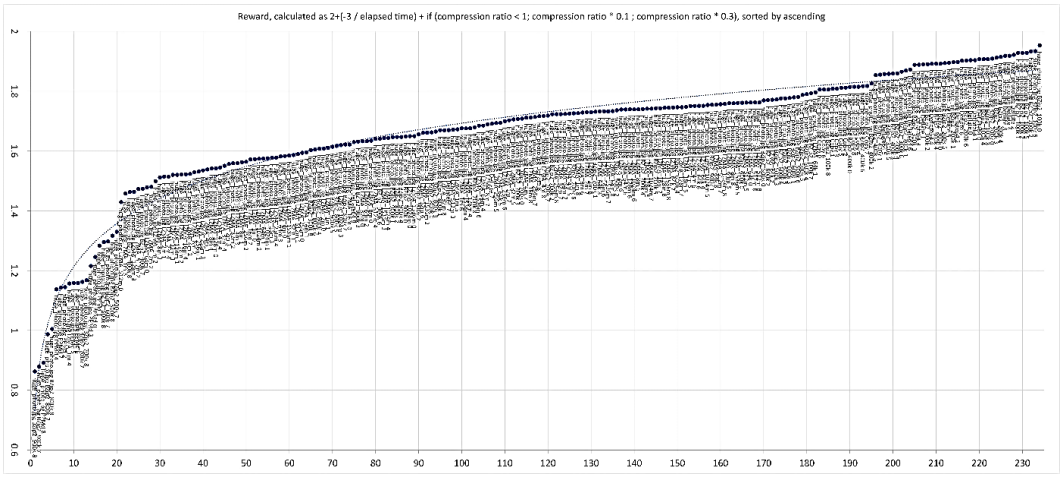


**Figure 9.** Chart of all combinations of compression parameters of photo file, sorted by reward, highest score – a combination of compression method BZIP2 100kB and compression level 0

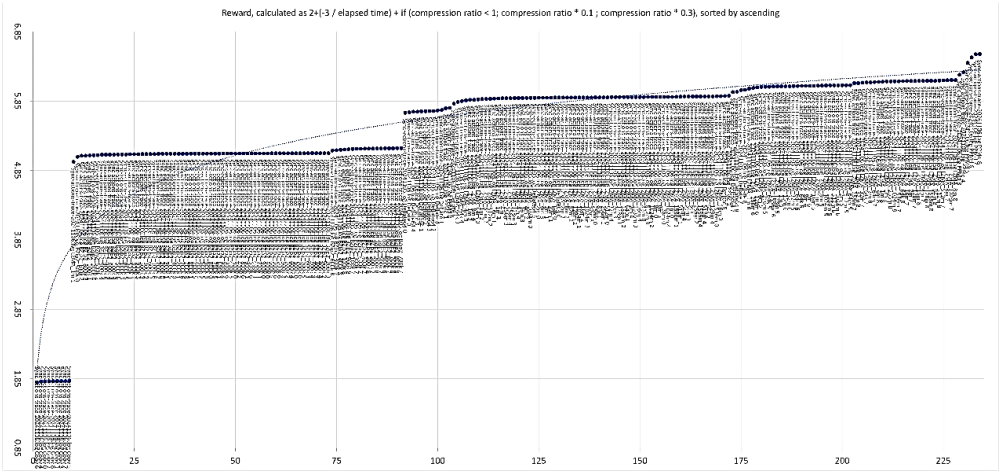


**Figure 10.** Chart of all combinations of compression parameters for text file, sorted by reward, highest score – a combination of compression method PPMd and compression level 8

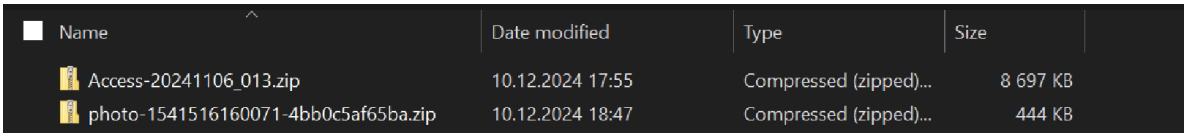


**Figure 11.** Backup tasks performed by a use of Windows Explorer "Send compressed (zipped) folder", resulting file size is 8697 kB for LOG file and 444 kB for JPG file

From Fig. 7 and 8 we can see that the proposed solution made a prediction to backup LOG file (size 95934 kB) with the next compression parameters: PPMd compression method and 8 compression level. This backup was performed in 3.612 s. with a compression ratio of 36.594 or as noted 36:1, so the prediction has the reward of 2.140. The resulting backup file had size of 2649 kB. That combination of compression parameters is the best optimal which is shown in Fig. 9. This backup consumes 3 times less storage and performed in 1.38 times faster than the backup made by the default command of Microsoft Windows – such backup with size 8697 kB finished in 5.001 s. (Fig. 11). Command used: *measure-command {powershell Compress-Archive .\photo-1541516160071-4bb0c5af65ba.jpg photo.zip}*.

Also, we can see that the proposed solution made a prediction to backup JPG file (size 444 kB) with the next compression parameters: BZIP2 compression method, 100 kB dictionary size and 0 compression level. This backup was performed in 0.119 s. with a compression ratio of 1.007 or as noted 1.007:1, so the prediction has the reward of 1.927. The resulting backup file had size of 443 kB. That combination of compression parameters is the best optimal which is shown in Fig. 9. This backup consume 1.002 times less storage and performed in 39.45 times faster than backup made by built-in option of Microsoft Windows – backup with size 443 kB finished in 4.695 s. (Fig. 11).

In summary, the PPO model could achieve successful results with enough steps in the environment (to get the first successful predictions need approximately ~50000 steps) and a sufficient amount of incoming data.

## 4. CONCLUSIONS

1. Results demonstrate that the proposed solution could find the best optimal of compression parameters, so the resulting backup will use 3 times less storage than the backup made by the regular static backup approach. It's done by achieving higher compression ratios (36:1) and reduced backup time in 1.38 times for heterogeneous files.

2. The use of a normalized observation space, including metrics like file extension, file size, and compression ratio, enables the PPO model to generalize well to a wide range of file types. This approach can be scaled to handle diverse datasets and backup scenarios without requiring manual configuration or file-specific rules.

3. The proposed solution could be used for implementing production-ready backup software by wrapping the learned model along with python scripts into executable problem, e.g. exe file for OS Microsoft Windows.

4. The proposed solution also identifies challenges, such as the need for fine-tuning reward functions to handle complex trade-offs and the potential computational overhead of training and deploying reinforcement learning models.

5. The model could be trained on a larger data set including medical, engineering or mechanical calculations and corresponding files produced by application servers.

**References**

1. Ferdous J., Islam R., Mahboubi A., Islam Z. (2023). A Review of State-of-the-Art Malware Attack Trends and Defense Mechanisms. Institute of Electrical and Electronics Engineers (IEEE) Access, pp. 121118–121141. https://doi.org/10.1109/ACCESS.2023.3328351

2. Number of Internet of Things (IoT) connections worldwide from 2022 to 2023, with forecasts from 2024 to 2033. Available at: https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/ (accessed 09.12.2024).
3. Vanness R., Chowdhury M., Rifat N. (2023). Malware: A Software for Cybercrime // Institute of Electrical and Electronics Engineers (IEEE) International Conference on Electro Information Technology (eIT), pp. 513–518. https://doi.org/10.1109/eIT53891.2022.9813970
4. Arányi G., Vathy-Fogarassy Á., Szücs V. (2024) Evaluation of a New-Concept Secure File Server Solution. Future Internet, Multidisciplinary Digital Publishing Institute (MDPI), 16 (9), p. 306. https://doi.org/10.3390/fi16090306
5. Chang D., Li L., Chang Y., Qiao Z. (2021) Cloud Computing Storage Backup and Recovery Strategy Based on Secure IoT and Spark. Mobile Information SystemsVolume 2021, issue 1, volume 6, p. 9505249. https://doi.org/10.1155/2021/9505249
6. Wang Z., Goudarzi M., Gong M., Buyya R. (2024). Deep Reinforcement Learning-based scheduling for optimizing system load and response time in edge and fog computing environments. Future Generation Computer Systems 152, pp. 55–69. https://doi.org/10.1016/j.future.2023.10.012
7. Zhou G., Tian W., Buyya R., Xue R., Song L. (2024). Deep reinforcement learning-based methods for resource scheduling in cloud computing: a review and future directions. Artificial Intelligence Review, pp. 57–124. https://doi.org/10.1007/s10462-024-10756-9
8. Lee S., Kang J., Kim J., Baek W., Yoon H. (2024) A Study on Developing a Model for Predicting the Compression Index of the South Coast Clay of Korea Using Statistical Analysis and Machine Learning Techniques. Applied Sciences (Switzerland), volume 14, issue 3, p. 952. https://doi.org/10.3390/app14030952
9. Dantas P., Sabino da Silva W., Cordeiro L., Carvalho C. (2024) A comprehensive review of model compression techniques in machine learning. Applied Intelligence, volume 54, issue 22, pp. 11804–11844. https://doi.org/10.1007/s10489-024-05747-w
10. RL Algorithms – Stable Baselines3 2.5.0a0 documentation, Available at: https://stable-baselines3.readthedocs.io/en/master/guide/algos.html#rl-algorithms/ (accessed: 09.12.2024).
11. Stefanyshyn V., Stefanyshyn I., Pastukh O., Kulikov S. (2024) Comparison of the accuracy of machine learning algorithms for brain-computer interaction based on high-performance computing technologies. Scientific Journal of TNTU (Tern.), vol. 115, no. 3, pp. 82–90. https://doi.org/10.33108/visnyk_tntu2024.03.082
12. Zhao L., Gatsis K., Papachristodoulou A.. Stable and Safe Reinforcement Learning via a Barrier-Lyapunov Actor-Critic Approach. 62nd Institute of Electrical and Electronics Engineers (IEEE) Conference on Decision and Control (CDC), 2023, pp.1320–1325. https://doi.org/10.1109/CDC49753.2023.10383742
13. Varshosaz M., Ghaffari M., Johnsen E., Wąsowski A. Formal Specification and Testing for Reinforcement Learning. 2022 Institute of Electrical and Electronics Engineers (IEEE) International Conference on Electro Information Technology (eIT), 2022, pp. 513–518.
14. Pawlicki M., Pawlicka A., Uccello F., Szelest S., D'Antonio S., Kozik R., Choraś M. (2024) Evaluating the necessity of the multiple metrics for assessing explainable AI: A critical examination. Neurocomputing, volume 602, p. 128282. https://doi.org/10.1016/j.neucom.2024.128282
15. Stefanyshyn V., Stefanyshyn I., Pastukh O., Kulikov S.. (2024) Comparison of the accuracy of machine learning algorithms for brain-computer interaction based on high-performance computing technologies // Scientific Journal of TNTU (Tern.), vol. 115, no. 3, pp. 82–90. https://doi.org/10.33108/visnyk_tntu2024.03.082
16. Abdulhameed A. S., Lupenko S. (2022) Potentials of reinforcement learning in contemporary scenarios // Scientific Journal of TNTU (Tern.), vol. 106, no. 2, pp. 92–100. https://doi.org/10.33108/visnyk_tntu2022.02.092

**УДК 004.632**

# ВИКОРИСТАННЯ НАВЧАННЯ З ПІДКРІПЛЕННЯМ ДЛЯ ОПТИМІЗАЦІЇ СТРАТЕГІЇ АРХІВУВАННЯ ДАНИХ ДЛЯ ПРОГРАМНОГО СЕРВЕРА

**Андрій Гарасівка; Анатолій Лупенко**

*Тернопільський національний технічний університет імені Івана Пулюя, Тернопіль, Україна*

***Резюме.*** *Ефективне стиснення даних є критично важливим компонентом сучасних систем резервного копіювання даних, особливо в середовищах із різними типами файлів і різними функціями*

*продуктивності. Резервне копіювання захищає критично важливі дані програмного сервера від неочікуваних збоїв, таких як апаратні збої, помилки програмного забезпечення або кібератаки, забезпечуючи безперервність роботи бізнесу. Надійні стратегії резервного копіювання необхідні для відновлення критично важливих систем і служб, забезпечення безперервності роботи в разі збоїв інфраструктури. Багато галузей підтримують безпечне резервне копіювання даних відповідно до правових і нормативних вимог щодо захисту даних та конфіденційності. На основі аналізу експериментальних даних запропоновано новий підхід до навчання з підкріпленням, використовуючи модель проксимальної політики оптимізації для динамічної оптимізації стратегії архівування даних, яка є частиною систем резервного копіювання. Модель навчена передбачати найефективнішу комбінацію параметрів стиснення на основі атрибутів файлів у цільовій файловій системі, вивчаючи стан середовища та скалярну безпосередню винагороду і час резервного копіювання даних для конкретних файлів. Цей адаптивний підхід дозволяє моделі приймати рішення в режимі реального часу, пристосовані до варіацій робочого навантаження програмного сервера. Пропонований підхід дає змогу реалізувати операції резервного копіювання різних типів файлів з певною комбінацією параметрів стиснення. При цьому модель, навчаючись, оцінює винагороду та коригує політику для максимізації своєї ефективності прогнозування. Для оцінювання результатів експерименту необхідно виконати резервне копіювання зі всіма можливими комбінаціями параметрів стиснення, щоб визначити максимальну можливу винагороду для кожної комбінації параметрів стиснення. Максимальна винагорода використовується для порівняння з винагородою моделі в процесі оцінювання моделі. Наголошено на потенціалі машинного навчання для автоматизації та оптимізації завдань резервного копіювання даних, надаючи оптимізоване рішення для високопродуктивних систем або середовищ із частим записуванням даних. Отримані результати сприятимуть покращенню програмного забезпечення та роботи DevOps спеціалістів, а також зменшать використання дискового простору та час резервного копіювання для даних програмного сервера.*

***Ключові слова:*** *резервне копіювання даних, машинне навчання, оптимізація проксимальної політики, стратегія резервного копіювання, рішення для стиснення, навчання з підкріпленням, програмний сервер.*

*https://doi.org/10.33108/visnyk_tntu2025.01.018* *Отримано 13.11.2024*