# Towards understanding of Deep Reinforcement Learning Agents used in Cloud Resource Management

Andrzej Małota, Paweł Koperek[0000−0003−3613−2390], and Włodzimierz Funika[0000−0003−3321−7348]

AGH University of Krakow, Faculty of Computer Science, Electronics and Telecommunication, Institute of Computer Science, al. Mickiewicza 30, 30-059, Kraków, Poland

email:malota.andrzej@gmail.com, funika@agh.edu.pl, pkoperek@gmail.com

**Abstract.** Cloud computing resource management is a critical component of the modern cloud computing platforms, aimed to manage computing resources for a given application by minimizing the cost of the infrastructure while maintaining a Quality-of-Service (QoS) conditions. This task is usually solved using rule-based policies. Due to their limitations more complex solutions, such as Deep Reinforcement Learning (DRL) agents are being researched. Unfortunately, deploying such agents in a production environment can be seen as risky because of the lack of transparency of DRL decision-making policies. There is no way to know why a certain decision is made. To foster the trust in DRL generated policies it is important to provide means of explaining why certain decisions were made given a specific input. In this paper we present a tool applying the Integrated Gradients (IG) method to Deep Neural Networks used by DRL algorithms. This allowed to obtain feature attributions that show the magnitude and direction of each feature's influence on the agent's decision. We verify the viability of the proposed solution by applying it to a number of sample use cases with different DRL agents.

**Keywords:** cloud resource management, deep reinforcement learning, explainable artificial intelligence, explainable reinforcement learning, deep neural networks

## 1 Introduction

In the recent years using cloud computing infrastructure became the dominating approach to provisioning computing resources. Thanks to virtually unlimited resources and usage-based billing, creating the cost-effective applications which automatically adjust the amount of used resources became straightforward. At the same time, the Cloud Service Provider (CSP) can increase the utilization of hardware by using it to serve multiple users in the same time. Managing resources in cloud computing infrastructures is a challenging task. The resource

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36021-3_55

management algorithms can, unfortunately, provide too much or too few resources for the services. To mitigate this issue many CSP provide tools which allow for on-demand dynamic resource allocation (auto-scaling). Such tools implement typically a *horizontal scaling* approach in which resources (e.g. Virtual Machine (VM)s) of the same type are added or released [5]. Horizontal scaling can use various methods of triggering scaling actions including rule-based methods [24] or predicting resource requirements using polynomial regression [5]. Nowadays, many methods are based on more advanced techniques such as deep learning and reinforcement learning.

Reinforcement Learning (RL) is a method of learning from interactions with an environment [23]. During a training process, a RL agent learns to map observations into actions through an iterative trial-and-error process. Upon executing an action, the agent receives a positive or negative reward signal and its objective is to maximize the sum of rewards received. Deep Reinforcement Learning (DRL) is one of the subfields of Machine Learning (ML) that combines RL and Deep Learning (DL) by employing Deep Neural Network (DNN), e.g. as various function approximators. In recent years it has attracted much attention and has been successfully applied to many complex domains: playing computer games [15], robotics [18], Natural Language Processing (NLP) in human- machine dialogue [4], traffic signal control [8], or cloud resource management [26].

Applying DNN suffers from the lack of explainability (also called interpretability). It is difficult to *explain* why a certain result has been produced by a DNN. Furthermore, the user of a machine learning algorithm may be legally obliged to provide explanations for certain decisions made with use of those algorithms. That issue can be mitigated by utilizing one of the Explainable Artificial Intelligence (XAI) techniques which aim at providing additional information on how ML algorithms produce their outputs. Whereas explainability can be considered well developed for standard ML models and neural networks [19, 21], in the case of RL there are still many issues that need to be resolved in order to enable using it in fields where it is imperative to understand its decisions [11]. Explainable Reinforcement Learning (XRL) is a recently emerged new subfield of XAI which receives a lot of attention. The goal of XRL is to explain how decisions are generated by policies employed by the DRL agents [14]. Such policies can be very complex and difficult to debug. Providing more insight into how they function allows to improve the design of the training environments, reward functions and DNN model architectures.

In this paper we extend the prior work [6, 7], where the use of the policy gradient optimization approach for automatic cloud resource provisioning has been studied. We implement a tool that allows to explain the actions of the cloud computing resource management policies trained with the use of DRL. This approach enables post hoc attributing input features to decisions made by the DRL agent which dynamically creates or deletes VMs based on an observed application requirements and resource utilization. The tool is available as an open source project [13]. To the best of authors' knowledge, this paper is the

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36021-3_55

first example of using the Integrated Gradients (IG) XAI technique to interpret DRL agents in the cloud resource management setting.

This paper has the following structure: Section 2 presents related work on the use of DRL in cloud resource management and application XAI techniques on DRL agents. Section 3 describes a cloud resource management simulation environment. In Section 4 we analyze the experimental results of the interpretation scheme applied in various scenarios. Lastly, Section 5 draws conclusions from the experiments and outlines future work.

## 2 Related Work

### 2.1 DRL in Cloud Resource Management

In recent years, the usage of DRL in cloud resource provisioning gained significant attention. In [25], authors explore the use of standard RL and DRL for cloud provisioning. They utilize a system where the users specify rewards based on cost and performance to express their goals. In such an environment they compare the use of the tabular-based Q- learning algorithm with the Deep Q-Networks (DQN) approach to achieve the objectives set by the users. The policy trained with the DQN approach achieved the best results.

In their study, authors of [2] utilized the Double Deep Q-Networks (DDQN) algorithm to reduce power consumption in CSP. The proposed DRL-based cloud resource provisioning and task scheduling method consist of two stages. The first one allocates the task to one of the server farms. The second one chooses the exact server to run the task in. The reward function is calculated using the energy cost of the performed action. The proposed *DRL-Cloud* system compared with a round-robin baseline improved the energy cost efficiency while maintaining a low average reject rate.

In [6] three DRL policy gradient optimization methods (Vanilla Policy Gradient (VPG), Proximal Policy Optimization (PPO) and Trust Region Policy Optimization (TRPO)) are used to create a policy used to control the behavior of an autonomous cloud resources management agent. The agent interacts with a simulated cloud computing environment which processes a stream of computing jobs. The environment state is represented by a set of metrics that are calculated in each step of the simulation. The reward function is set up as the negative cost of running the infrastructure with added penalties for breaching the Service-Level-Agreement (SLA) conditions. The policy which achieved the lowest cost was created using the PPO algorithm. In [7], the PPO-based autonomous management agent is compared with the traditional auto-scaling approach available in Amazon Web Services (AWS). As a sample workload an evolutionary experiment, consisting of multiple variable size phases, has been chosen. The policy training has been conducted within a simulated cloud environment. Afterwards the policy has been deployed to a real cloud infrastructure, the AWS Elastic Compute Cloud. The total cost of managed resources was slightly lower (0.7%) when a PPO-trained policy has been used, compared with a threshold-based ap-

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36021-3_55

proach. The trained policy was considered to be able to generalize well enough to be re-used across multiple similar workloads.

### 2.2 Explainable AI in Deep Reinforcement Learning

DRL has shown great success in solving various sequential decision-making problems, such as playing complicated games or controlling simulated and real-life robots. However, existing DRL agents make decisions in an opaque fashion, taking actions without accompanying explanations. This lack of transparency creates key barriers to establishing trust in an agent's policy and significantly limits the applicability of DRL techniques in critical application fields such as finance, self-driving cars, or cloud resource management [10]. So far, most of the research on the usage of XAI in DRL has been focused on finding the relationship between the agent's action and the input observation at the specific time step - detecting the features that contributed most to the agent's action at that specific time. According to the XAI taxonomies in [1], it is possible to classify all recent studies into two main categories: *post hoc explainability* and *transparent methods*.

When dealing with images as input data, one can provide explanations through saliency maps. A saliency map is a heat map that highlights pixels that hold the most relevant information and the value for each pixel shows the magnitude of its contribution to the Convolutional Neural Network (CNN)'s output. Unfortunately, saliency maps are sensitive to input variations. The authors of [9] aim to fix that with their perturbation-based saliency method applied to agents trained to play Atari games via the Asynchronous-Advantage-Actor-Critic (A3C) algorithm. They conducted a series of investigative explorations aiming to explain how agents made their decisions. First, they identified the key strategies of the three agents that exceed human baselines in their environments. Second, they visualized agents throughout training to see how their policies evolved. Third, they explored the use of saliency for detecting when an agent is earning high rewards for the wrong reasons. This includes a demonstration that the saliency approach allows non-experts to detect such situations. Fourth, they found Atari games where the trained agents performed poorly and used saliency to debug these agents by identifying the basis of their low-quality decisions. The approach presented in [9] is an example of *post hoc explainability*.

In [12] the authors applied a post hoc interpretability to the DRL agent which learned to play the video game CoinRun [3]. They used PPO [20] algorithm, with the agent's policy model being a CNN. *Attribution* shows how the neurons affect each other, usually how the input of the network influences its output. Dimensionality reduction techniques [17] have been applied to the input frames from the game and attributions were calculated using IG. IG computes the gradient of the model's prediction output to its input features. This allows to produce the feature attributions that display the feature's influence on the prediction. Furthermore, IG requires no modification to the original DNN. Authors built an interface for exploring the detected objects that shows the original image and also positive and negative attributions, to explain how the objects influence the agent's value function and policy. Their analysis consists of three

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36021-3_55

main parts: dissecting failure (trying to understand what the agent did wrong and what reasons behind it were), hallucinations (model detected a feature that was not there), model editing (manually editing the model weights so that the agent would ignore certain observations).

In [16], an interpretability analysis of the DRL agent with a recurrent attention model on the games from the ALE environment is performed. The authors observed basic attention patterns using an attention map, which is a scalar matrix representing the relative importance of layer activations at different 2D spatial locations with respect to the target task. It was also discovered that the model attends to task-relevant things in the frame - player, enemies, and score. The ALE environment is predictable, e.g. enemies appear at regular times and in regular configurations. It is important, to ensure that the model truly learns to attend to the objects of interest and act upon the information, rather than to memorize and react only to certain patterns in the game. In order to test it, they injected an enemy object into the observation at an unexpected time and in an unexpected location. They observed that the agent correctly attends to and reacts to the new object. They discovered that the model performs forward planning/scanning - it learns to scan through available paths starting from the player character, making sure there are no obstacles or enemies in the way. When the agent does see the enemy, another path is produced in order to avoid it.

## 3 Environment Setup

The simulation environment used in our research is a result of the prior work [6]. A fundamental component of the simulation process is implemented with the CloudSim Plus simulation framework [22]. The environment is wrapped with the interface provisioned by the Open AI Gym framework. Figure 1 presents the system architecture.

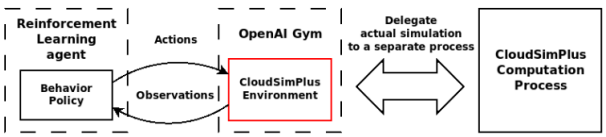



Fig. 1: Components of the system [6].

The workload used in this work is a simple evolutionary experiment which improves the architecture of a network that recognizes handwritten digits. The main objective of the agent is to allocate cloud infrastructure resources in an optimal way to the running workload. The environment in which the agent's training was conducted was simulated to avoid the high costs of provisioning a real computing infrastructure. In order to finish the experiment in a reasonable

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36021-3_55

time, simulation time was sped up and the number of steps per episode was limited.

The reward function $R$ (Equation 1) equals the negative cost of running the infrastructure added to the SLA penalty. The SLA penalty adds some cost for every second delay in task execution and was calculated by multiplying the number of seconds of the delay of task execution by the penalty value. The cost of running the infrastructure was calculated by multiplying the number of VMs by the hourly cost of using the VM.

$$R = -(\text{NumberOfVMs} * \text{HourlyCostOfVM} + \text{SlaPenalty}) \tag{1}$$

where:

$$\text{HourlyCostOfVM} = \$0.2,$$
$$\text{SlaPenaltyNumberOfSecondsOfDelay} * \text{Penalty},$$
$$\text{Penalty} = 0.00001\$$$

The environment state is represented by a vector of cloud infrastructure metrics that are being calculated at each time step of the simulation. These metrics include: the number of running virtual machines (vmAllocatedRatio), average RAM utilization (avgMemoryUtilization), 90-th percentile of RAM utilization (p90MemoryUtilization), average CPU utilization (avgCPUUtilization), 90-th percentile of CPU utilization (p90CPUUtilization), total task queue wait time (waitingJobsRatioGlobal), recent task queue wait time (waitingJobsRatioRecent). The set of available actions is limited to the following: do nothing, add or remove a small VM, add or remove a medium VM, add or remove a large VM.

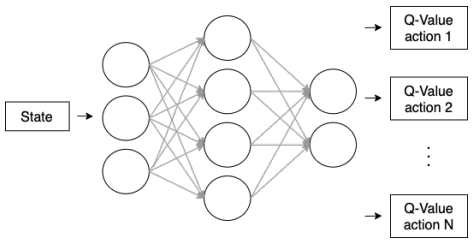
## 4 Understanding of DRL agents used in Cloud Resource Management

In this paper, we develop a tool to interpret the decision-making process of DRL agents used in cloud resource management. Identifying the relationships between the input data and the agent's output allows to understand why certain decisions were made. To achieve this we employ the IG method which shows the magnitude and direction of each feature's influence on the agent's output. It requires minimal modification of the original network and can inform on the cloud infrastructure metrics that influenced the agent's decision-making process. The attribution value for a feature can be positive or negative, indicating its contribution towards or against a certain prediction, respectively. We demonstrate the usefulness of the discussed approach in a few scenarios: attributing input metrics to the action chosen by the policy (for two DRL algorithms: DQN with an MultiLayer Perceptron (MLP) model and PPO with a CNN model), providing a policy summarization, explaining how a policy evolves during training, debugging policy decisions, removing irrelevant features.

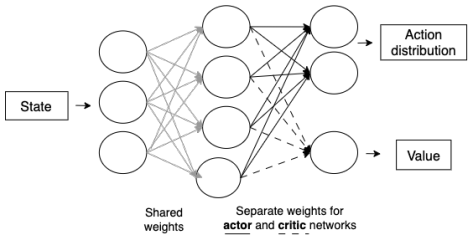
ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36021-3_55

### 4.1 Input metric attribution

The DNN model used in the DQN algorithm produces approximations of $q$-values which denote the value of taking action $a$ in state $s$. The model produces $q$-values for each possible action. The final output is chosen using a greedy approach which selects the action with the highest value. Figure 2a presents an overview of the neural network model used in DQN.




(a) Model used as a policy for the DQN algorithm




(b) Model used as a policy for the PPO algorithm.

Fig. 2: Simplified diagrams of used DNN models.

The IG can calculate feature attributions for each possible action separately, however, we focus on the action which has the highest $q$-value, which is then chosen for execution. In the DQN approach we have applied it to a MLP network. The MLP architecture is straight-forward to understand. Due to its simplicity it is also faster to calculate its output's attributions. MLP accepts a one-dimensional (1D) vector of feature observations as an input, which allows to visualize attributions as a simple to read bar-chart. Figure 3 presents the attributions for a sample action chosen by the policy. It consists of two parts. The first one (top) shows the environment state (y-axis presents metric value (0-1)). The second one (bottom) shows the attributions value for the best action (positive in green, negative in red).

The PPO policy model has an actor-critic architecture and produces two outputs. The policy (*actor*) network provides a distribution of the probabilities of the actions possible to execute in a given state. Since an action is selected by

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36021-3_55

sampling the provided distribution the actor is considered to be directly responsible for choosing the action. The *critic* network estimates the *value* function, which quantifies the expected reward if the agent follows the policy until the terminal state. Typically, to optimize the amount of computational resources required for training, the actor and critic networks are combined into a single network with two distinct outputs, what is presented in Figure 2b.

In the context of PPO the IG is used to calculate attributions for the input features in relation to the action chosen by sampling the distribution. To demonstrate the capabilities of such an approach we have applied the discussed method to a model which included CNN layers. CNN are best known for their usage in the image recognition, however they can be also used in the time series prediction. The main advantage of CNN over MLP while performing the attribution is that CNN can present feature attributions from previous $n$ time steps for the action that takes place at time step $t$. It is important because the agent's decision is not always based on the immediate state of the environment but may depend upon an observation that took place sometime in the past. Figure 4 presents the example attributions for the CNN architecture for the best action. It consists of three plots. The upper left chart shows the frame of the input to the neural network, x-axis shows time steps, from 0 - the current one down to the -14 time step (15-th time step from the past), the y-axis presents a value (0-1) for each environment metric. The upper right one shows the positive attribution values while the bottom one shows the negative attribution values for the best action.

### 4.2 Debugging the training process

The interpretability tool can be leveraged to investigate the rationale behind an agent's decision for a given observation. The aim is to discern whether the decision was adequate given the observed circumstances and if it was triggered by the expected inputs. If this is not the case, the tool should help to discover which feature values might have influenced an incorrect decision.

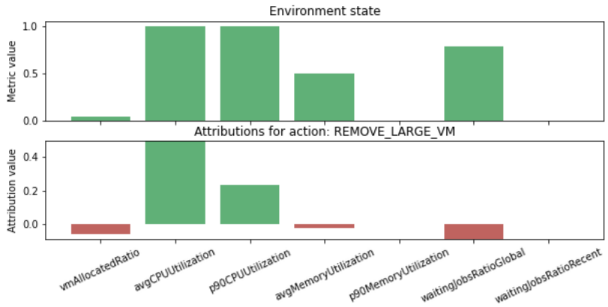



Fig. 3: Attributions for example observation - DQN agent with MLP architecture.

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36021-3_55

In Figure 3 we observe the attributions underlying the DQN-MLP agent's decision to remove a large VM. Such a result was primarily influenced by the CPU utilization features (avgCPUUtilization, p90CPUUtilization). However, given the context of high CPU utilization, long waiting times for job execution (waitingJobsRatioGlobal), and low resource allocation (vmAllocatedRatio), we can consider removing resources as an incorrect decision. A human operator would have rather avoided taking an action or increased the number of VMs.

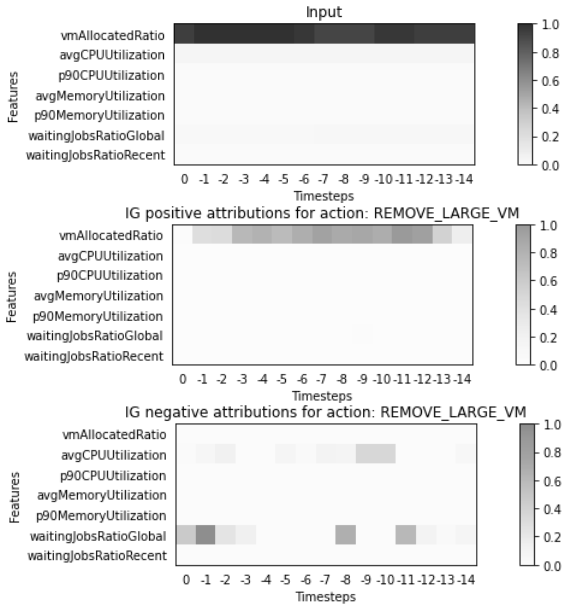



Fig. 4: Attributions for example observation - PPO agent with CNN architecture. Heatmaps are presented in grayscale to improve readability.

In Figure 4 we present the attributions underlying the PPO-CNN agent's decision to remove a large VM, which at the time was deemed to be the appropriate course of action. The top plot illustrates the state of the environment at the time of the decision. The vmAllocatedRatio feature exhibited consistently high values across all time steps, indicating that most of the available VMs were being utilized, while the values of all other features were close to zero. Examining the middle plot, which illustrates positive attributions (i.e., reasons to make the decision), we can confidently conclude that the decision to remove a large VM was primarily driven by the vmAllocatedRatio feature. The bottom chart shows negative attributions (i.e., reasons to not make the decision), with the waitingJobsRatioGlobal feature - denoting the number of jobs waiting to be executed due to insufficient resources - being assigned a small negative attribution against the decision to remove the large VM. In other words, the decision to

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36021-3_55

remove the large VM was appropriate because at the time, a substantial number of VMs were already in use, and there were very few jobs waiting in the queue to be executed.

### 4.3 Policy summarization

To identify general behavioral patterns of the agents, we must examine the policy predictions of the agent across multiple examples , in other words use a *global interpretation approach*. To accomplish this, we determine the global feature importance for the policy, which is achieved by calculating the mean absolute attributions over hundreds of predictions. Figure 5 illustrates the policy summarization for the DQN agent with the MLP architecture. It is apparent that the agent relies primarily on three features - avgCPUUtilization, vmAllocatedRatio, and p90CPUUtilization. The remaining features have little to no influence on the agent's predictions. In contrast, the recurrent PPO agent with MLP policy relies primarily on one feature (vmAllocatedRatio) when making decisions.

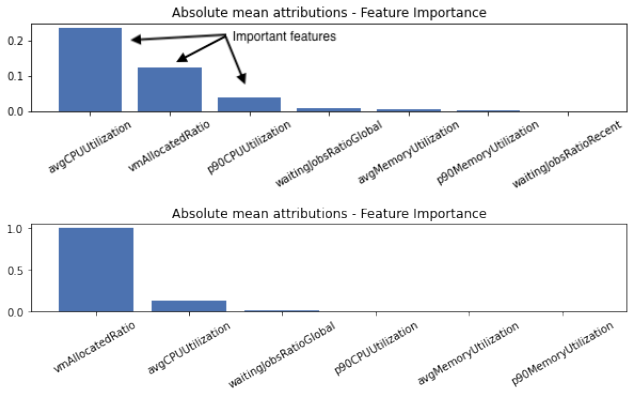



Fig. 5: Absolute mean attributions - Feature Importance - DQN-MLP (top), Recurrent PPO-MLP (bottom).

The feature importance analysis for the CNN based architectures provides more detailed information, allowing to observe the exact mean influence that each feature had at each time step. Figure 6 presents the feature importance for the agents with a CNN architecture. The agent relies primarily on three features: avgCPUUtilization, vmAllocatedRatio, and waitingJobsRatioGlobal. In contrast, the feature importance for the PPO agent with CNN architecture is more balanced, as it makes decisions based on five out of the seven available features across all time steps.

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36021-3_55

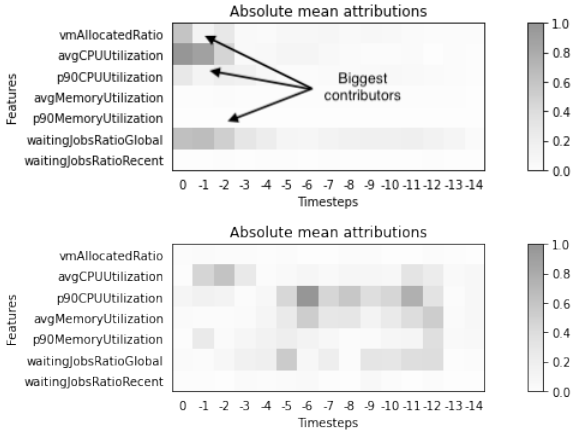



Fig. 6: Absolute mean attributions - Feature Importance - DQN-CNN (top), Recurrent PPO-CNN (bottom). Heatmaps are presented in grayscale to improve readability.

### 4.4 Evolution of policies during training

In the process of training, DRL agents begin with a random policy and adjust their weights to optimize their performance. This unfortunately means that the decisions made during the first training iterations might have a negative effect on the managed system, e.g. the amount of resources might be reduced instead of getting increased. In this study we investigate how the resource management strategy of the DRL agents evolves over time. The insights learnt from this analysis allow to understand whether a policy is making progress during training. Furthermore, in case the results are inadequate, such an analysis can be a starting point for debugging the training process. The presented analysis demonstrates that the agents training in our experiments underwent significant changes in their attributions during the training process.

Figure 7 presents changes in an absolute mean attribution in a sample experiment with training a DQN agent using a policy including CNN layers. The figure includes five plots: attribution in the initial model without any training (top), attribution while the model is being trained (middle three), attribution in the final model after training (bottom). As expected, the initial model's attributions are dispersed across various features and time steps. However, as the training progresses, the policy begins to emerge, and the number of attributed features is being reduced. By the end of the training, the attributions are only present for three features: $vmAllocatedRatio$, $avgCPUUtilization$, and $waitingJobsRatioGlobal$. Furthermore, the policy seemed to have focused solely on the three most recent time steps and began to ignore earlier time steps. These insights suggest that we could reduce the number of past steps included in observations since they are not being utilized. That in turn would result in creating a smaller model and, as a consequence, a faster training.

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36021-3_55

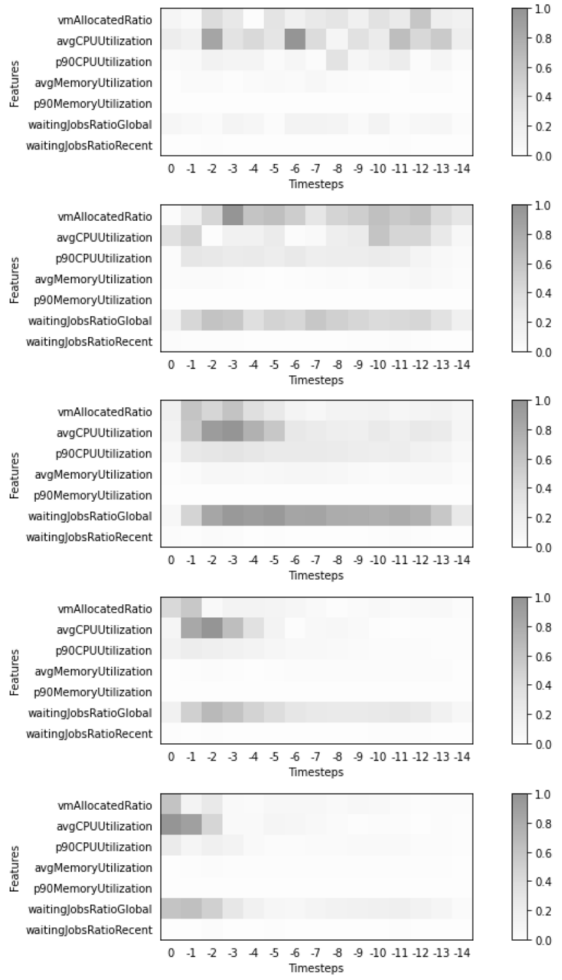



Fig. 7: Mean absolute attributions throughout the training process of the DQN agent with CNN policy. Heatmaps are presented in grayscale to improve readability.

### 4.5 Removing irrelevant features

The global feature importance can be quantified by calculating the absolute mean attributions over a given dataset, and can serve as a useful tool for selecting only the most relevant features for an agent's decision-making. There are several benefits of using a smaller feature set, including a simpler and more compact model, improved interpretability of the agent's decision-making process, and faster training times.

An example of such a situation is presented in Figure 8. The absolute mean attributions for the Recurrent PPO-MLP agent reveal that the feature *waiting−*

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36021-3_55

*JobsRatioRecent* has a negligible influence on the agent's decisions. To validate the feature attributions as a metric for the feature selection, we removed *waitingJobsRatioRecent* and retrained the agent. The bottom chart in Figure 8 presents the mean attributions of the new agent, which are nearly identical to those of the original agent trained with all the features. The comparable performance of the agent trained with and without the feature *waitingJobsRatioRecent* indicates that the IG and other XRL techniques can effectively aid in the feature selection process for DRL agents.

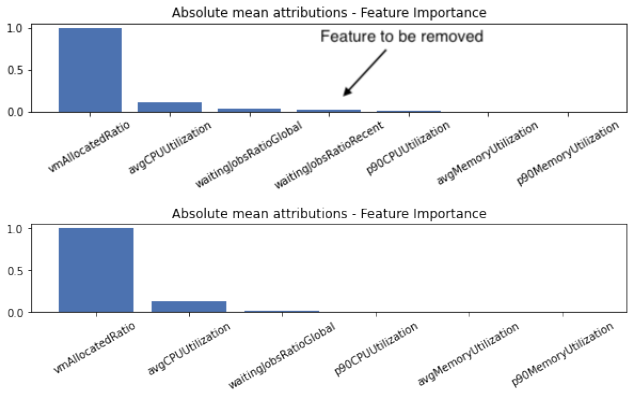



Fig. 8: Absolute mean attributions - Feature Importance - Recurrent PPO-MLP. The top chart presents attributions including the *waitingJobsRatioRecent*. Bottom one presents attributions in the case when that metric is excluded.

## 5 Conclusions

In this paper an approach allowing to explain the actions of an automated cloud computing resource management policy has been implemented. It was demonstrated how it can help understand why a neural network model returned a particular value (which can be translated, e.g. into a scaling action in the context of the resource management) by applying the IG method. To evaluate its correctness and utility, a number of experiments with training policies using DRL algorithms have been conducted. Two approaches were analyzed: Q-learning (DQN with MLP and CNN models) and policy gradient optimization (PPO with MLP, CNN, recurrent with MLP or CNN feature extractor). As the training environment a simulated cloud computing system processing a sample, dynamic workload has been chosen. The agent embedded in that system was controlled by the policy trained with the use of the mentioned algorithms and could increase or reduce the amount of used resources (virtual machines).

The presented experiments demonstrated how the described approach can be used to identify which input metrics contributed towards making a decision. It

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36021-3_55

is worth noting that such metrics were different depending on the combination of the model architecture and the training algorithm. We have demonstrated that removing a feature with a little to no influence on the predictions did not affect the quality of decisions made by the policy. Additionally we have analyzed how metric changes in time being influenced the policies' output in the case of models that use CNN. Finally, we have demonstrated how Integrated Gradients method can be used to determine whether the decisions made by the policies were taken using the right premises (e.g. if a decision to remove a VM is taken when the usage of available resources is low).

The information obtained with the use of the IG method during the agent's training provided insight into the direction of the learning process. This allowed to confirm whether the policy was in fact making progress during training and that its decisions were in fact taken in connection to the discovered relationships between metrics. We believe that ability to provide evidence of such behavior can help accelerate the adoption of automated resource management systems. Understanding how the input to the policy affects its output allows to debug situations when an improper resource scaling decision is made. Furthermore, tracking such information might be necessary to fulfill legal obligations. Without that deploying automated resource management software might not be allowed.

**Acknowledgements** The research presented in this paper was supported by the funds assigned to AGH University of Krakow by the Polish Ministry of Education and Science.

## References

1. Barredo Arrieta, A., et al.: Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. Information Fusion **58**, 82–115 (2020). https://doi.org/10.1016/j.inffus.2019.12.012
2. Cheng, M., et al.: DRL-cloud: Deep reinforcement learning-based resource provisioning and task scheduling for cloud service providers. In: 2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC). pp. 129–134 (2018)
3. Cobbe, K., et al.: Quantifying Generalization in Reinforcement Learning (2018). https://doi.org/10.48550/ARXIV.1812.02341
4. Cuayáhuitl, H.: SimpleDS: A Simple Deep Reinforcement Learning Dialogue System (2016). https://doi.org/10.48550/ARXIV.1601.04574
5. Dutta, S., et al.: SmartScale: Automatic Application Scaling in Enterprise Clouds. In: 2012 IEEE Fifth International Conference on Cloud Computing. pp. 221–228 (2012)
6. Funika, W., Koperek, P.: Evaluating the Use of Policy Gradient Optimization Approach for Automatic Cloud Resource Provisioning. In: Wyrzykowski, R., Deelman, E., Dongarra, J., Karczewski, K. (eds.) Parallel Processing and Applied Mathematics. pp. 467–478. , LNCS 12043, Springer International Publishing, Cham (2020)
7. Funika, W., Koperek, P., Kitowski, J.: Automatic Management of Cloud Applications with Use of Proximal Policy Optimization. In: Computational Science – ICCS 2020: 20th International Conference, June 3–5, 2020, Proceedings, Part I. pp. 73–87. , LNCS 12137, Springer-Verlag, Berlin, Heidelberg (2020)

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36021-3_55

8. Gregurić, M., et al.: Application of Deep Reinforcement Learning in Traffic Signal Control: An Overview and Impact of Open Traffic Data. Applied Sciences **10**(11) (2020)
9. Greydanus, S., et al.: Visualizing and Understanding Atari Agents (2017). https://doi.org/10.48550/ARXIV.1711.00138
10. Guo, W., et al.: EDGE: Explaining Deep Reinforcement Learning Policies. In: Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., Vaughan, J.W. (eds.) Advances in Neural Information Processing Systems. vol. 34, pp. 12222–12236. Curran Associates, Inc. (2021)
11. Heuillet, A., et al.: Explainability in Deep Reinforcement Learning. CoRR **abs/2008.06693** (2020)
12. Hilton, J., et al.: Understanding RL Vision. Distill (2020). https://doi.org/10.23915/distill.00029
13. Małota, A., et al.: Trainloop-driver. https://github.com/andrzejmalota/trainloop-driver/tree/master/examples (2023)
14. Milani, S., et al.: A Survey of Explainable Reinforcement Learning (2022). https://doi.org/10.48550/ARXIV.2202.08434
15. Mnih, V., et al.: Playing Atari with Deep Reinforcement Learning. In: NIPS Deep Learning Workshop (2013), http://arxiv.org/abs/1312.5602
16. Mott, A., et al.: Towards Interpretable Reinforcement Learning Using Attention Augmented Agents (2019). https://doi.org/10.48550/ARXIV.1906.02500
17. Olah, C., et al.: The Building Blocks of Interpretability. Distill (2018). https://doi.org/10.23915/distill.00010
18. OpenAI, et al.: Solving Rubik's Cube with a Robot Hand (2019). https://doi.org/10.48550/ARXIV.1910.07113
19. Ribeiro, M.T., et al.: "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. p. 1135–1144. KDD '16, Association for Computing Machinery, New York, NY, USA (2016)
20. Schulman, J., et al.: Proximal Policy Optimization Algorithms (2017). https://doi.org/10.48550/ARXIV.1707.06347
21. Selvaraju, R.R., et al.: Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. International Journal of Computer Vision **128**(2), 336–359 (2019)
22. Campos da Silva Filho, M., et al.: CloudSim Plus: A cloud computing simulation framework pursuing software engineering principles for improved modularity, extensibility and correctness. In: 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM). pp. 400–406 (2017)
23. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. The MIT Press, second edn. (2018)
24. Tighe, M., Bauer, M.: Integrating cloud application autoscaling with dynamic VM allocation. In: 2014 IEEE Network Operations and Management Symposium (NOMS). pp. 1–9 (2014)
25. Wang, Z., et al.: Automated Cloud Provisioning on AWS using Deep Reinforcement Learning (2017). https://doi.org/10.48550/ARXIV.1709.04305
26. Zhang, Y., et al.: Intelligent Cloud Resource Management with Deep Reinforcement Learning. IEEE Cloud Computing **4**(6), 60–69 (2017). https://doi.org/10.1109/MCC.2018.1081063

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36021-3_55