Bidirectional Transfer Learning of Multi-Objective Reinforcement Learning for Efficient Online VNF Profiling

Pratchaya Jaisudthi *
Computer Engineering Program, Faculty of Computer Science and Information Technology, Rambhai Barni Rajabhat University,
Tachang, Muang, Chanthaburi, 22000, Thailand
***Corresponding Author E-mail:** pratchaya.j@rbru.ac.th
Received: Jul 08, 2024; **Revised:** Sep 12, 2024; **Accepted:** Sep 28, 2024

Abstract

This research investigates the effectiveness of transfer learning combined with multi-objective reinforcement learning (RL) for profiling diverse VNFs, including Snort (in both Passive and Inline modes) and virtual firewalls. We compare the resource allocation predictions of an RL model with those of a standard machine learning approach, such as a multilayer perceptron (MLP). While MLPs can outperform RL models in certain scenarios, they lack adaptability. Unlike RL, MLPs require retraining when conditions change. To address this limitation, we propose adaptable RL profilers that dynamically allocate resources (CPU, memory, and link capacity) based on the performance needs of the VNFs. The experiments were conducted in four scenarios: two cases of transferring from Snort (Passive Mode and Inline Mode) to a virtual firewall (vFW) and two cases of transferring from vFW to Snort. Our results reveal a trade-off between computational resource utilization (CPU and memory) and link capacity. In the transfer learning scenario from Snort's Inline Mode VNF to vFW, the Q-Learning model with transfer learning (TL) achieved approximately a 20% reduction in vCPU usage compared to the MLP approach. However, it did not perform as effectively as the MLP in reducing link capacity utilization. Conversely, in the transfer learning scenario from vFW to Snort Inline Mode VNF, the Q-Learning with TL model reduced link capacity usage by 20% compared to other models, although it was less efficient in reducing CPU usage.

Keywords: Transfer Learning, Multi-Objective Reinforcement Learning, Online VNF Profiling,

1. Introduction

Driven by the need for cost reduction and greater agility, telecom providers are increasingly adopting Network Function Virtualization (NFV). This technology paves the way for Zero-Touch Service Management (ZSM), a concept gaining significant traction (as evidenced by the ZSM white paper [1]). In essence, ZSM aims to fully automate network infrastructure management, minimizing human involvement across all stages - from initial setup and maintenance to troubleshooting and optimization. This white paper details how self-configuring, self-healing, self-monitoring, and self-optimizing networks achieve this automation, enabling efficient and flexible resource allocation. However, despite these advancements, human expertise remains essential for the effective functioning of these systems.

Current network management for virtualized services relies heavily on manual programming to manage resource allocation, which lacks true autonomy. Existing solutions like ONAP [2] and OSM [3] primarily offer pre-defined functions via APIs, still requiring human intervention for basic tasks like placement or scaling.

To achieve true automation, next-generation systems must possess a deeper understanding of network demands and automatically adapt to changing conditions. This shift aligns with the trend in cloud computing towards using general-purpose resources for VNFs, simplifying service management. Telecom providers deliver these services based on Service Level Agreements (SLAs), which outline performance benchmarks [4].

This method proposes an online learning model as an alternative to static VNF profiling. It aims to dynamically adjust network resources, including CPU utilization, memory utilization and bandwidth, for optimal allocation. This allocation ensures service providers meet their pre-defined performance targets for network services running on their software-defined networks (SDNs) [5],[6].

While traditional VNF profiling using profilers is essential for NFV MANO systems in optimizing resources and deploying services efficiently [7], further research is required, particularly in online VNF profiling, which demands fast and accurate processing. Key areas for exploration include:

1) Identifying the most suitable machine learning model for VNF profiling to ensure accurate information gathering.
2) Reducing the amount of training data needed for unseen VNF instances to speed up system convergence.

Additionally, Machine Learning-based profiling can leverage Transfer Learning (TL) to tackle new VNF types or instances in unfamiliar network conditions—a key challenge in machine learning (ML) networks where data and time for training and inference are scarce. Data limitations refer to the quality of the available data (e.g., monitoring granularity over time and space), while time constraints refer to the need to quickly profile new VNFs, even in real-time or during deployment.

This research makes several key contributions to the field of transfer learning using Multi-Objective Reinforcement Learning (MORL) for VNF profiling:

1) **Bidirectional Transfer Learning:** This research makes a significant contribution by demonstrating bidirectional transfer learning in the context of MORL for online VNF profiling, comparing the transfer of knowledge from Snort to Virtual Firewall and vice versa.
2) **Dynamic Resource Allocation with RL**: A major contribution is demonstrating that RL-based models can dynamically adjust resources such as CPU, memory, and link capacity based on real-time performance demands of different VNFs. This flexibility sets RL apart from traditional MLP models, which require retraining to adapt to new conditions.

These contributions collectively advance the understanding of how MORL can be used for efficient online VNF profiling, especially in dynamic and resource-constrained environments.

This research investigates the applicability of MORL for VNF resource allocation across diverse VNF types. We propose an RL-based approach that automatically configures vCPU cores, memory, and link capacity for VNFs. This approach ensures optimal resource allocation while guaranteeing that VNFs meet predefined performance objectives or key performance indicators (KPIs). Our contributions pave the way for more efficient and responsive profiling in NFV environments, further advancing the potential of transfer learning for real-time VNF management.

2. Literature Review

Researchers are exploring ways to measure the performance of NFV components, namely VNFs, under various resource limitations. This analysis, called NFV profiling, typically involves linking KPIs to specific resource allocations. One approach [8] involves incorporating an offline profiling system into the NFV design. This system can assess how limited resources, such as memory, CPU, and storage access, affect performance metrics like the number of packets processed per second. The study shows that this method is effective for analyzing both individual VNFs and Service Function Chains (SFCs).

While evaluating every resource configuration within an NFV architecture is undeniably thorough, it can be incredibly slow and impractical. To overcome this hurdle, a time-constrained profiling model was introduced by researchers [9]. This approach leverages two key elements: a selector that strategically picks a limited set of configurations for testing, and a predictor that estimates performance for configurations that haven't been directly evaluated.

A recent system, called Novel Autonomous Profiling (NAP) [10], automates resource allocation for VNFs using ML. NAP first narrows down potential configurations by considering the number of vCPU cores. Then, it refines the allocation process with a machine learning predictor. This predictor analyzes a broader range of data, including network bandwidth, memory, and vCPU cores. By considering this richer data set, NAP improves its predictions and selects the most suitable resource configuration for the workload and performance requirements. In simpler terms, NAP acts like a smart assistant that automatically assigns the optimal number of vCPUs, memory, and network bandwidth to a VNF, while also determining its ideal workload capacity.

ML offers exciting possibilities to bridge the knowledge gaps in VNF profiling, particularly for complex scenarios with multiple objectives. Unlike traditional methods, ML can adapt to dynamic environments and uncertainty by simultaneously optimizing various resources or achieving desired outcomes. While some existing research explores ML techniques like linear regression and curve fitting for profiling, limitations have emerged. For example, regression models struggle to predict saturation points in resource allocation [11]. Additionally, supervised learning models (like ANN and kNN) and interpolation methods might not effectively capture continuously increasing or decreasing trends seen in real-world network behavior. On the other hand, curve fitting demonstrates promise in accurately predicting VNF performance, but it falls short in optimizing for multiple resource objectives concurrently. In summary, supervised learning might be suitable for static resource allocation scenarios, but it struggles to adapt to the dynamic nature of real-world networks.

This research expands on iOn-Profiler introduced in [12]. While most existing ML-based profilers focus on optimizing a single goal, iOn-Profiler breaks new ground by utilizing RL to adapt to real-world network situations in real-time. This approach tackles multi-objective optimization, dynamically adjusting to changing network conditions. Building o-n work in [11], which employed static, supervised learning models, iOn-Profiler [12] and i-Profiler [13] delves deeper into RL-based adaptive VNF profiling. It uses meticulously designed reward functions to optimize resource allocation (CPU, memory, bandwidth) for VNFs. This optimization targets desired performance metrics like CPU/memory utilization, latency, and their balanced trade-off. To achieve this, the study explores a wide range of weightings for each objective, ultimately identifying the Pareto front – the set of optimal resource allocation and VNF performance combinations.

Current VNF profiling techniques often overlook the Pareto front. Even when considered, existing methods typically rely on static models designed for specific network conditions and VNF traffic patterns, as seen in work [14]. This limitation in current profiling methods motivates our research using iOn-Profiler, which leverages RL for dynamic adaptation and automatic resource management.

Self-governing networks utilize machine learning for automated resource management, reducing human involvement. A technique called transfer learning

(TL) boosts network autonomy by leveraging pre-existing models. Similar to research [15], TL allows networks to adapt to novel situations quicker by drawing on established knowledge. This is particularly beneficial in scenarios with limited resources, ultimately enhancing the effectiveness, efficiency, and reliability of autonomous networks.

In the realm of 6G networks, a novel technique called Intelligent VNF profiling employs machine learning to identify optimal configurations for VNFs. These configurations define the ideal number of resources needed for network services. This ensures service providers meet agreed-upon KPIs outlined in SLAs, all while adhering to pre-defined performance constraints. Unlike traditional methods, Intelligent VNF profiling leverages machine learning to navigate dynamic and unpredictable network environments.

Reusing knowledge from similar tasks can significantly boost machine learning model performance, and TL is a powerful technique that achieves this [15]. In the realm of NFV, TL allows us to transfer knowledge across different network functions, applications, or tasks. This leads to more efficient, effective, and high-performing NFV systems.

For instance, the Q-TRANSFER framework utilizes Q-learning to maximize positive knowledge transfer in deep learning for network applications [16]. This framework employs a control system based on Markov Decision Processes to identify the optimal strategy for transferring knowledge. This strategy finds valuable knowledge from the source domain that benefits tasks in the target domain. Through TL, Q-TRANSFER efficiently acquires this knowledge, giving models in the target domain a strong starting point.

Another study showcases the use of Artificial Neural Network (ANN)-based TL for accurate Quality of Transmission (QoT) prediction across various networks [17]. This approach eliminates the need to train ANN models from scratch and achieves reliable Q-factor predictions for different optical systems (target domains) with less additional training data. By leveraging TL, researchers can not only evaluate system upgrade potential but also significantly reduce data collection time and effort, preventing potential disruptions or delays in system deployment.

Bi-directional Online Transfer Learning (BOTL) [18] enables the exchange of knowledge between different VNFs by transferring stable models between domains. BOTL detects concept drifts using local models and transfers only stable models, which have been used across enough instances without detecting drifts, to other VNFs to improve their predictive performance. This peer-to-peer model transfer allows knowledge to flow both ways, from one VNF to another and vice versa.

3. Methods

To optimize resource allocation for a specific VNF, we employ a multi-objective reinforcement learning approach. This method leverages a Markov decision process, essentially a map of the system's possible states, available actions in each state, and the corresponding rewards for those actions.

Imagine the system as a network of interconnected states. Each state represents a unique resource configuration, defined by factors such as CPU cores, available memory, and network bandwidth. Additionally, each state captures performance metrics like CPU utilization, memory usage, latency, and the desired output rate.

Within this system, various actions can be taken to modify resource allocation. These actions involve increasing, decreasing, or maintaining the current allocation level. Each action triggers a transition to a new resource allocation state. Notably, adjustments occur in specific increments, such as 0.2 cores for processors, 100 MB for memory, and 50 Mbps for network bandwidth. To determine the most suitable action, a specialized strategy called a Scalarized ϵ-greedy policy is employed. This policy aims to maximize a combined reward function that considers all resource types.

The system incentivizes efficient resource allocation for VNFs through a reward mechanism. Similar to a points system, a function called the "zedoid" (like an inverse sigmoid function) calculates rewards based on resource usage as Eq. (1). Using more resources leads to lower rewards, encouraging efficient setups. This formula is adjusted to ensure that rewards smoothly decrease as the resource usage (ô) increases beyond a certain point (ô = 0.5). This encourages more efficient resource use, penalizing higher resource consumption.

$$r_i = \frac{1}{1+e^{\beta(\hat{o}-0.5)}} \tag{1}$$

Within this function, "i" represents a specific resource type, "ô" indicates the amount allocated (e.g., how many CPU cores are allocated), and the Greek letter beta (β) is a steepness coefficient which acts like a dial controlling the sensitivity of rewards to resource changes. β essentially determines how sharply rewards decrease with increased resource usage.

In multi-objective Q-learning, the algorithm adapts a scalarisation function to handle multiple objectives (details in **Algorithm 1**). During action selection, Q-values for all goals are stored together as a vector.

Algorithm 1 Scalarized Greedy for Q-Learning

```
Input: $w_o$ ← The weight of each objective.
$SQlist \leftarrow \{\}$
 for each action $a \in A$ do
     $v \leftarrow \hat{Q}(s,a) = \{Q_1(s,a), Q_2(s,a), \ldots, Q_i(s,a)\}$
     $\widehat{SQ}_{linear}(s,a) \leftarrow f(v,w)$
     Append $\widehat{SQ}_{linear}(s,a)$ to $SQlist$
  end
return $\arg\max_{a'} SQlist$
```

Eq. (2) defines a scalarisation function (f) that calculates an SQ value using a vector of rewards as Eq. (1) (r_1, r_2, r_3 for vCPU, memory, and link capacity) and a weight vector (w). The weights (w_1, w_2, w_3) prioritize these rewards and always sum to 1. The algorithm then stores the calculated SQ values in a list and returns the next action (a') with the highest SQ value.

$$\widehat{SQ}_{\text{linear}}(s,a)=\sum_{o=1}^{n}\left[w_o \cdot Q_o(s,a)\right] \quad \text{where} \sum_{o=1}^{n} w_o = 1 \tag{2}$$

Figure 1 represents the steps of a Q-learning algorithm applied to online VNF profiling. The algorithm begins by initializing key Q-learning parameters learning rate (α) to determines how much new information overrides old information. Here, α = 0.1, meaning the model gives more weight to new experiences. Discount factor (γ) represents how much future rewards are considered compared to immediate rewards. γ = 0.99 means the algorithm highly values future rewards. Best reward function parameter (β) is chosen for each reward function to balance the objective of the task.

Next, creating a Q-table for each resource, which the Q-table represents the Q-values for each state-action pair for different resources, denoted as $Q_i(s,a)$ where **s** is the current state, **a** is the action (e.g. increase/decrease each resource), and *i* is the resource being managed (e.g., CPU, memory, link capacity).

Then the agent uses an ε-greedy policy to balance exploration (trying new actions) and exploitation (using known actions that yield high rewards). A random number between 0 and 1 is generated. If this random number is less than ε-greedy, the agent chooses a random action (exploration). Otherwise, it chooses the action **a** that gives the maximum Q-value.

At this point, this algorithm will take action and observe new state. Once an action **a** is taken, the algorithm observes the next state **s'** and reward $\mathbf{r_i}$ for resource **i**, which is calculated using a predefined reward equation. Based on **s'**, the algorithm also determines the next action **a'**, which comes from Algorithm 1 (likely a related decision-making step).

Subsequently, updating Q-value (Q-learning update rule), the Q-value is updated using the Q-learning formula. The equation adjusts the Q-value based on the reward received, the discounted future reward, and the difference from the current Q-value. After updating the Q-value, the system scales the VNF based on the new state **s'**. This could involve adjusting resource allocations (CPU, memory, link capacity) to ensure optimal performance based on the demand.

Finally, the algorithm checks if the VNF is using the minimum required resources. If this condition is met, it moves to the next episode, implying the completion of a training iteration. If not, the process repeats, continuing to refine the resource scaling.

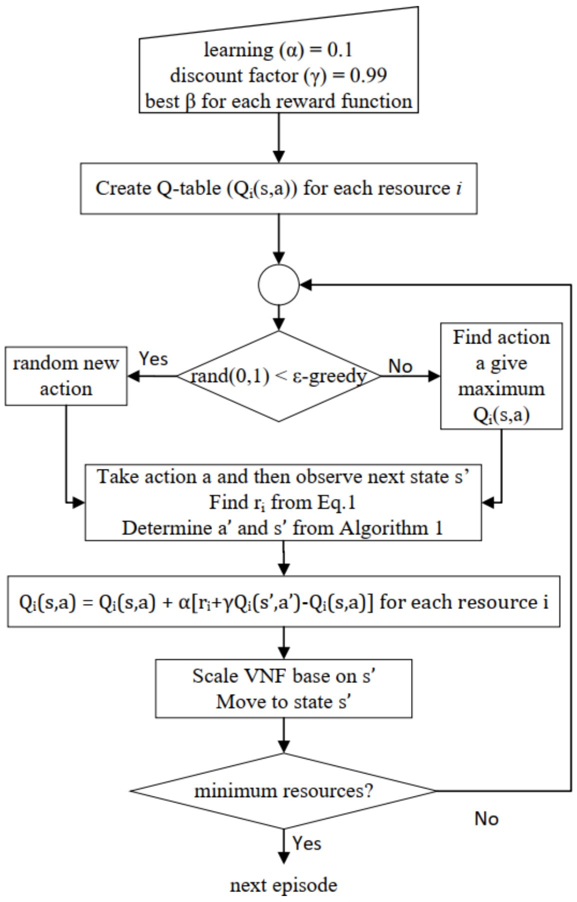

learning (α) = 0.1
discount factor (γ) = 0.99
best β for each reward function
Create Q-table ($Q_i(s,a)$) for each resource i
random new action
Yes
rand(0,1) < ε-greedy
No
Find action a give maximum $Q_i(s,a)$
Take action a and then observe next state s'
Find r_i from Eq.1
Determine a' and s' from Algorithm 1
$Q_i(s,a) = Q_i(s,a) + \alpha[r_i+\gamma Q_i(s',a')-Q_i(s,a)]$ for each resource i
Scale VNF base on s'
Move to state s'
minimum resources?
No
Yes
next episode

Figure 1 Process of ultiple-objective Q-Learning

4. Experiment Setup

In the experiment depicted in the **Figure 2**, the setup consists of three key components: a VNF under test (Snort VNF instance), a traffic generator (iPerf Client), and a traffic sink (iPerf Server), equipped with 2 vCPUs, 2 GB of memory, and 10 GB of storage. All VNFs are orchestrated using OSM MANO (Open Source MANO), which manages the resources and deployment.

The iPerf Client and Snort VNF instance are interconnected via a Linux bridge, with the same configuration between the Snort VNF and the iPerf Server. The iPerf Client generates traffic directed towards the iPerf Server, routing it through the Snort VNF for analysis and performance evaluation. The iPerf Client produced traffic patterns tailored to the iPerf Server by routing this traffic through the Snort VNF. During the training phase, resource allocation was optimized for the Snort VNF to handle iPerf traffic while maintaining optimal performance.

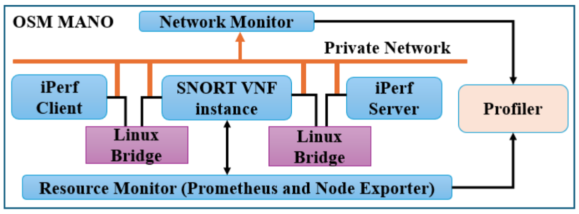


Figure 2 Experiment setup

The experiment is supported by two monitoring systems: the Network Monitor and the Resource Monitor. The Resource Monitor utilizes Prometheus and Node Exporter to track metrics such as CPU and memory usage. Simultaneously, the Network Monitor tracks traffic rates and latency over the private network. Specific performance thresholds as in **Table 1** were established, including CPU utilization between 90–100%, memory usage below 98%, and latency within 2–3 milliseconds. The offline profiling for each model lasted 48 hours, ensuring comprehensive performance insights.

Table 1 KPI Metrics

KPI Parameters	Values
vCPU utilization	95 ± 5 %
memory utilization	≤ 98
Latency	2.5 ± 0.5 ms

Our experiment implements dynamic resource adjustment (e.g., scaling vCPU, adjusting memory allocation, and link capacity) whenever the system nears its upper utilization limits (95% for vCPU, 98% for memory). This prevents over-utilization while maintaining optimal performance. Our system monitoring latency to ensure that any actions that cause spikes in vCPU or memory usage (such as traffic spikes) do not cause the latency to exceed 3 milliseconds. If latency approaches the 2.5 milliseconds upper threshold, the system should prioritize actions to reduce load, such as traffic shaping or throttling. These adjustments should ensure that the KPI thresholds for vCPU, memory, and latency are met while the online profiling continues to deliver real-time performance

We investigated the adaptability of profiler models across diverse VNFs with varying resource needs. Profilers designed for memory-intensive tasks, like packet copying, might adapt better to memory fluctuations compared to those designed for traffic interception.

To illustrate this concept, we explored two contrasting Snort use cases: Inline Mode and Passive Mode. Inline Snort acts as a network gatekeeper, inspecting all traffic before forwarding, which can potentially impact flow rates. Passive Snort, on the other hand, operates outside the main flow by copying data packets for threat detection, requiring distinct resources compared to Inline Snort.

Finally, we assessed a vFW, a VNF that controls traffic flow by restricting access to specific ports. In contrast, this research also examined the case where the vFW is used as a gateway, while Snort is the component that restricts access to network services.

This research examines how effectively models trained on one VNF can be applied to different VNFs, even without a direct connection between them. We explore this transferability with prioritizing all objectives equally in four scenarios: 1) transfer learning from Snort in Inline Mode to vFW, 2) from Snort in Passive Mode to vFW, 3) from vFW to Snort in Inline Mode and 4) from vFW to Snort in Passive Mode. To assess this, we compare online learning with Reinforcement Learning (RL) using a specifically configured Multi-Layer Perceptron (MLP) model (details in separate tables).

Although Random Forests (RF) have demonstrated strong performance in classification and regression tasks, they were not chosen for comparison in this work due to their inherent limitations when applied to dynamic and real-time environments like online VNF profiling. RF is a static, batch learning algorithm, which means that once trained, it does not adapt to changing conditions without complete retraining. This characteristic makes it less suitable for scenarios where the resource demands of VNFs evolve over time, as is the case in the continuous profiling and optimization of vCPU, memory, and link capacity in virtualized environments. MLP, on the other hand, is chosen over RF because it aligns more closely with the real-time, adaptive, and multi-objective nature of VNP Profiling. It also supports TL and integration with RL, which are central themes of this study.

5. Results

The dataset is divided into two parts: 90% is allocated for training, while the remaining 10% is reserved for testing. Prior to starting the training phase, it is worth noting that the data was normalized using Min-Max scaling. The model utilizes four KPIs as inputs and three resource-related variables as outputs. The input variables include memory usage, CPU usage, and latency, while the output variables consist of memory, link capacity, vCPU cores, and output rate. To ensure reliability, all results are reported with 95% confidence intervals based on 30 repetitions of each experiment.

Figures 3–6 depict the findings from experiments conducted under four scenarios. Each figure is further divided into subplots (a), (b), (c), and (d). The y-axis in subplots (a), (b), and (c) represents the predicted percentage of virtual CPU cores, memory allocation, and link capacity, respectively. Subplot (d) focuses on the Output Rate over link capacity, expressed as a percentage. The x-axis tracks the number of episodes completed during execution. Notably, all subplots compare the performance of MLP and RL (Q-learning) methods, with and without transfer learning applied.

Subfigures (a)–(c) in **Figures 3–4** predict resource usage for a vFW, while those in **Figures 5–6** predict resource usage for Snort in Inline Mode and Passive Mode, respectively. These predictions come from three models: a baseline using multi-layer perceptron (MLP), a reinforcement learning model, and both models applied after being trained on Snort data (Inline or Passive mode) and then adapted (via TL) to predict vFW resources. The x-axis ("training episode 0") indicates the starting point for transfer learning. This "new" episode 0 aligns with episode 200 for the original vFW model (green and blue curves, without TL), where predictions become stable at specific resource allocations. Finally, subfigures (d) in all of the following figures enable us to evaluate link utilization based on the achieved optimal output rate (OR).

All experiments (scenarios) use hyperparameters for MLP as specified in **Table 2**, while parameters for Q-learning are shown in **Table 3**, with resource weights notably set equally at 1/3. We selected the optimal β values for the reward function of each resource (CPU, MEM, LC) for each scenario. These values were determined by running the algorithm described in the Methods section. The selected β values are used consistently across all experiments.

Table 2 MLP Hyperparameters settings

Hyperparameters	Values
Activation Function in hidden layers	Selu
Activation Function in the output layer	Sigmoid
Epoch	300
Batch size	16
Optimizer	Adam
Learning rate	1e-4

Table 3 Q-Learning Parameters settings

Resources	Parameters
	Steepness coefficient (β)
Snort (Inline Mode)	
CPU	8
MEM	7
LC	7
Snort (Passive Mode)	
CPU	8
MEM	7
LC	8
vFW	
CPU	7
MEM	7
LC	9

5.1 Transfer learning from Snort Inline Mode to vFW

For this experiment, the hyperparameters of the MLP models were configured as specified in **Table 4**. **Figure 2** illustrates the transferability of the Snort VNF from Inline Mode to the vFW VNF. The Q-Learning with Transfer Learning graphs in **Figures 3(a)–(b)** demonstrate the system's ability to adapt to reduced resource utilization, namely vCPU and memory, respectively. However, **Figures 3(c)–(d)** indicate that this transfer learning scenario is unsuitable for attempts to reduce link capacity.

Table 4 Parameters settings for MLP models with transfer learning from Snort (Inline Mode)

Parameters	Values
Number of neurons in Input Layer	4
Number of neurons in output Layer	3
1st Hidden Layer	128
2nd Hidden Layer	128
3rd Hidden Layer	128

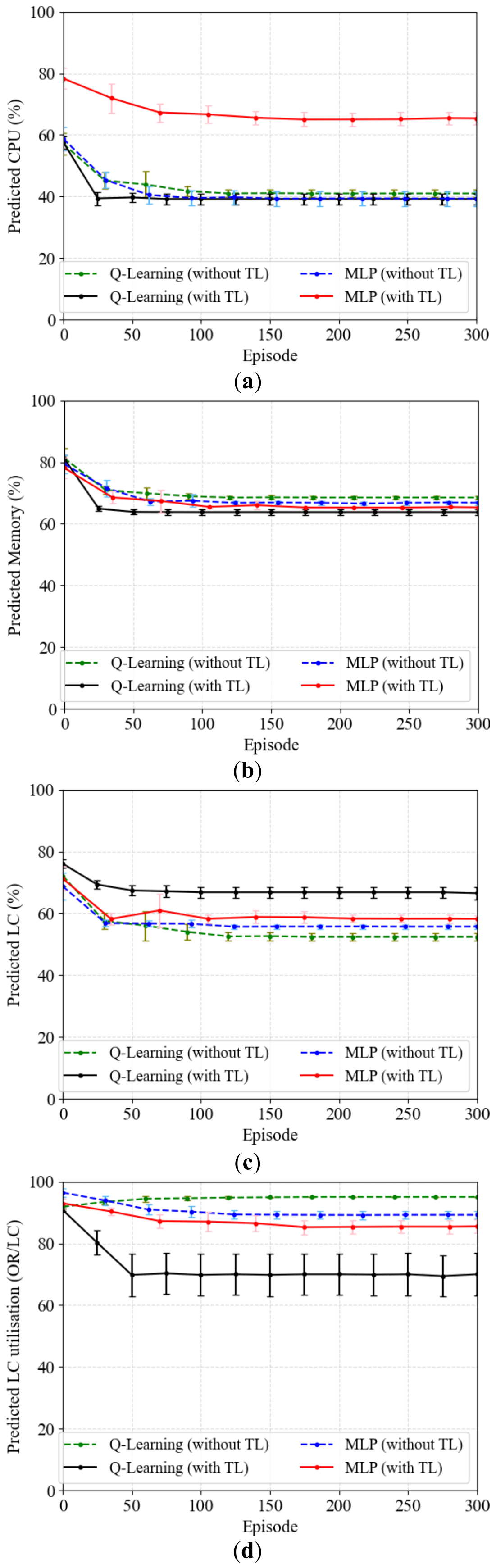


Figure 3 Percentage of vFW Predicted Resource by RL and MLP models with/without transfer learning from Snort (Inline Mode) **(a)** Predicted CPU **(b)** Predited Memory **(c)** Predicted LC **(d)** Predicted LC utilisation

Figure 3(a) demonstrates that the system was able to reduce CPU utilization to approximately 40% in the cases of Q-Learning with TL and MLP without TL. However, the Q-Learning with TL case reached convergence the fastest, around episode 25. This indicates that transfer learning from the Snort Inline mode to the vFW accelerates convergence significantly.

Figure 3(b) further illustrates that Q-Learning with TL led to the most substantial reduction in resource utilization, dropping to approximately 64%. This demonstrates that, in addition to faster convergence, transfer learning contributes significantly to resource efficiency.

In **Figure 3(c)**, Q-Learning without TL and MLP without TL stabilized at around 54% and 57%, respectively. Meanwhile, the Output Rate relative to link capacity in both cases was higher than in the transfer learning scenario, as depicted in **Figure 3(d)**. This suggests that transfer learning from Snort Inline mode to vFW has minimal impact on reducing link capacity utilization.

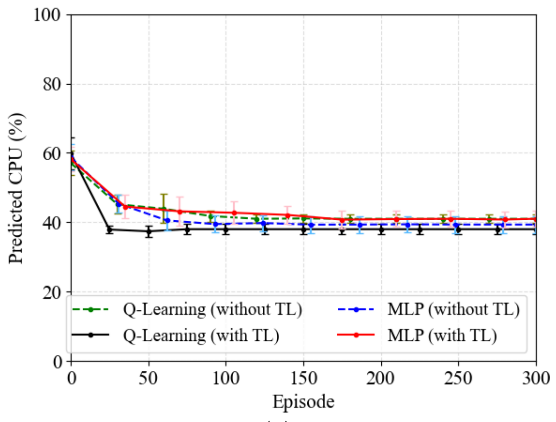
5.2 Transfer learning from Snort Passive Mode to vFW

The MLP models were constructed using the hyperparameter settings specified in **Table 5**.

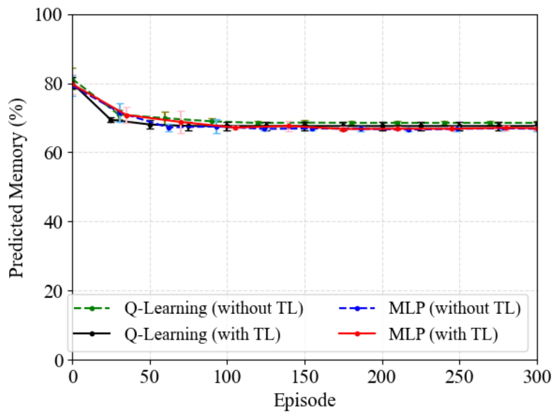
Table 5 Parameters settings for MLP models with transfer learning from Snort (Passive Mode)

Parameters	Values
Number of neurons in Input Layer	4
Number of neurons in output Layer	3
1st Hidden Layer	128
2nd Hidden Layer	128
3rd Hidden Layer	128

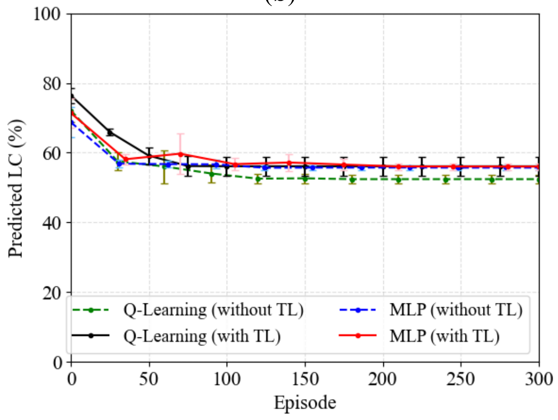
As observed in the previous cases, the vCPU utilization converges to a minimal constant value, approximately 39%. For Q-Learning with TL, this convergence occurs more rapidly than in other approaches, achieving stability around episode 25, as depicted in Figure 4(a). In contrast, as illustrated in **Figure 4(b)**, the impact of TL on memory reduction is relatively minor. Additionally, LC and OR/LC do not demonstrate significant improvements, even when TL is applied, as seen in Figures 4(c) and 4(d).



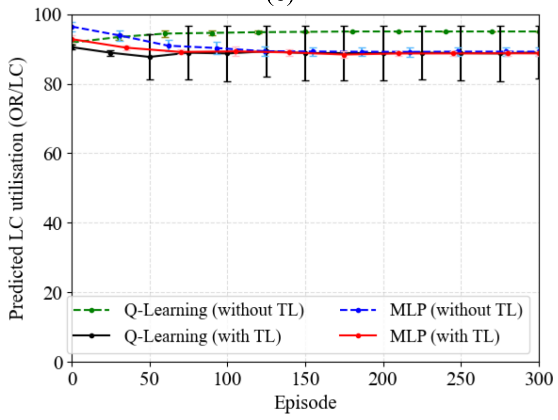
(**a**)



(**b**)



(**c**)



(**d**)

Figure 4 Percentage of vFW Predicted Resource by RL and MLP models with/without transfer learning from Snort (Passive Mode) (**a**) Predicted CPU (**b**) Predited Memory (**c**) Predicted LC (**d**) Predicted LC utilisation

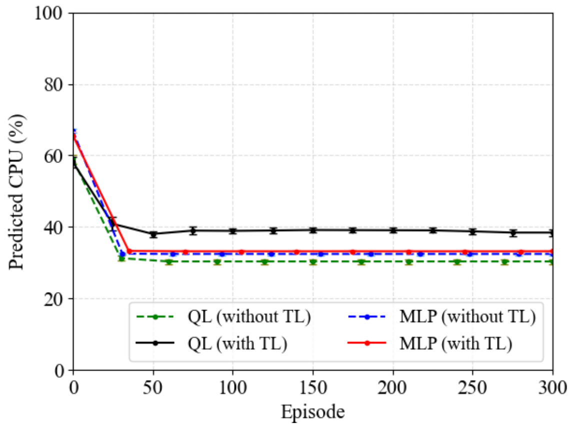
5.3 Transfer learning from vFW to Snort Inline Mode

Hyperparameter values for the MLP models were determined according to the specifications in **Table 6**.

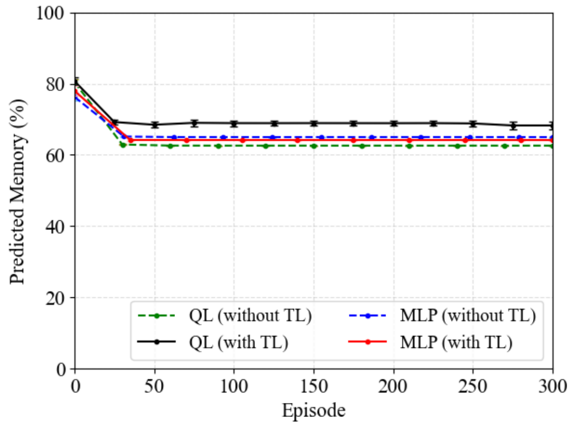
Table 6 Parameters settings for MLP models with transfer learning from vFW to Inline Mode

Parameters	Values
Number of neurons in Input Layer	4
Number of neurons in output Layer	3
1st Hidden Layer	128
2nd Hidden Layer	256
3rd Hidden Layer	128

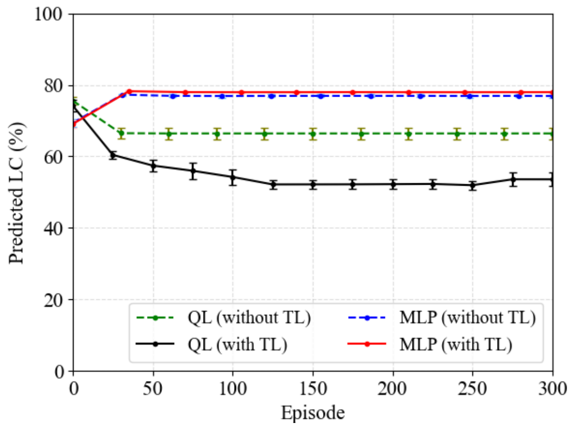
Upon examining the graphs of QL with Transfer Learning, it was observed that the system was unable to adapt to reduce vCPU and memory, as shown in **Figures 5(a)–(b)**, respectively. However, it is noteworthy in **Figures 5(c)–(d)** that this scenario prompts the system to adapt to lower link capacity while achieving higher OR/LC values compared to other lines, reaching approximately 52% and 96%, respectively.



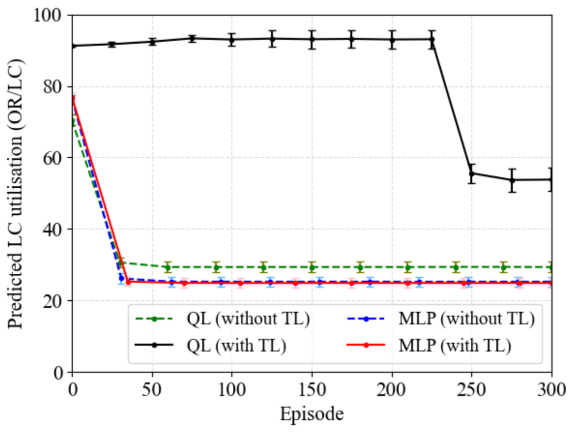
(**a**)



(**b**)



(**c**)



(**d**)

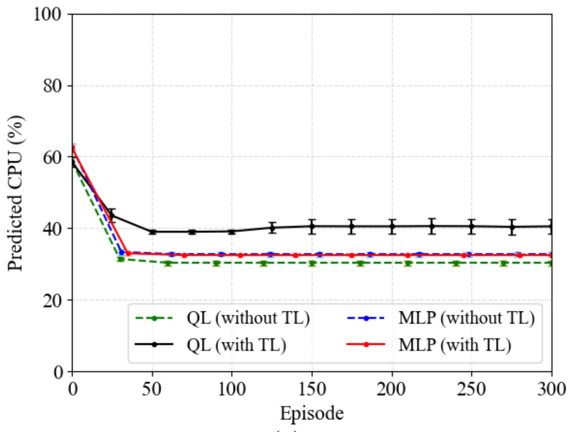
Figure 5 Percentage of Predicted Resource of Snort (Inline Mode) by RL and MLP models with/without transfer learning from vFW (**a**) Predicted CPU (**b**) Predited Memory (**c**) Predicted LC (**d**) Predicted LC utilisation

5.4 Transfer learning from vFW to Snort Passive Mode

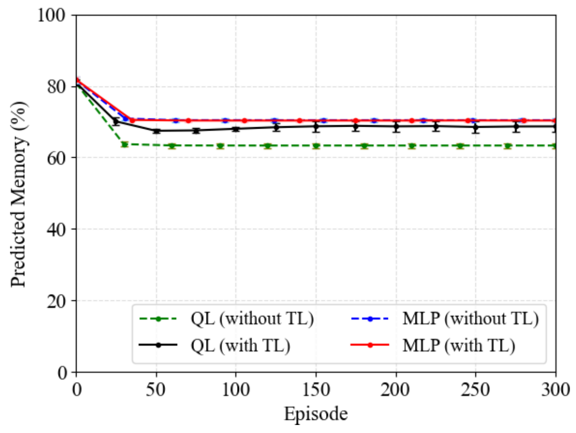
Table 7 provides the hyperparameter configurations used for the MLP models in this experiment. Similar to the previous scenario, transferring learning from vFW to Snort Passive mode in **Figure 6** demonstrates the system's attempt to minimize link capacity utilization to approximately 56% while maintaining OR/LC values at a relatively high level of around 90%.

Table 7 Parameters settings for MLP models with transfer learning from vFW to Passive Mode

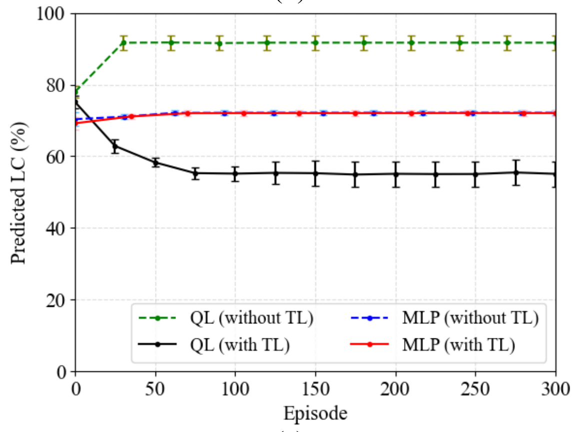
Parameters	Values
Number of neurons in Input Layer	4
Number of neurons in output Layer	3
1st Hidden Layer	128
2nd Hidden Layer	128
3rd Hidden Layer	128



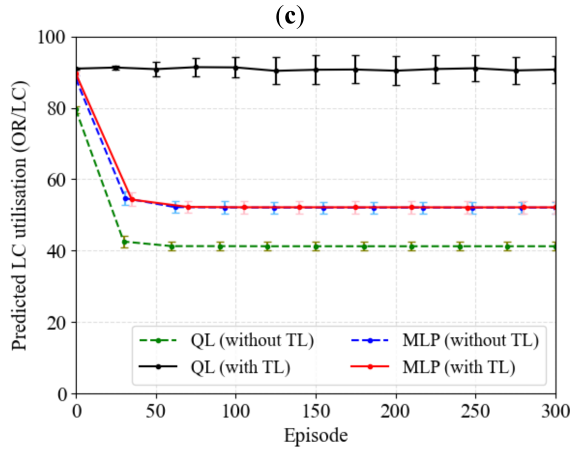
(**a**)



(b)



(c)



(d)

Figure 6 Percentage of Predicted Resource of Snort (Passive Mode) by RL and MLP models with/without transfer learning from vFW (**a**) Predicted CPU (**b**) Predited Memory (**c**) Predicted LC (**d**) Predicted LC utilisation

6. Discussion

Leveraging transferring RL models can offer significant advantages. RL shines in adapting to new environments compared to traditional models like MLP when transferred during deployment. This allows transferred RL models to quickly adjust and improve performance and resource efficiency. Additionally, RL with transfer learning converges to the minimum resource usage value faster than the case without transfer learning.

In terms of resource utilization reduction, vFW models trained with transfer learning from Snort, both in Passive Mode and Inline Mode, effectively reduced vCPU and memory usage. However, link capacity usage remained unchanged. Snort models with transfer learning from vFW, on the other hand, reduced link capacity usage but did not affect vCPU and memory consumption.

Analyzing relevant graphs, we see a trade-off between high utilization (CPU and memory) and link capacity, with higher utilization often leading to better performance. Interestingly, the original model (without TL) achieves the best results in both Snort scenarios.

Considering transferability of RL and MLP, the findings in all scenarios, where equal weight scalarization was used, suggest that RL approaches are generally preferable to MLPs when considering transferability. While MLP achieved better convergence for LC in the specific case of transfer learning from Snort (Inline), it suffered a significant performance penalty in terms of virtual CPU (vCPU) usage compared to the original model. Additionally, MLPs, like other supervised learning (SL) models, are inherently limited for real-world deployment. Unlike online learning models that can be trained and adapt during deployment, MLPs require offline training in a separate environment. This necessitates replacing a temporary model after offline training, which is cumbersome. Moreover, deploying a pre-trained SL model in real-time might result in limited or even no performance improvement.

Applying this approach in real-time is possible but comes with certain challenges and considerations:

1) Speed of Transfer Learning: Transfer learning reduces the amount of data and time required to train models, which is beneficial for real-time applications. Instead of starting from scratch, the model can apply previously learned knowledge from a similar VNF and adapt more quickly.
2) Online Learning Capabilities: If the transfer learning is implemented as an online learning mechanism, where the system continuously adapts in real-time as new data arrives, it is well-suited for real-time VNF profiling.
3) Multi-Objective Optimization: The MORL framework allows simultaneous optimization of multiple objectives (e.g., minimizing resource usage while maximizing security), which is critical in real-time VNF profiling scenarios, where trade-offs need to be balanced instantly.
4) Profile Complexity of VNFs: VNFs like Snort and virtual firewalls share similar objectives and environments, making transfer learning effective in speeding up profiling and learning.

Based on the analysis of the behaviour of Snort in inline mode and the Virtual Firewall VNF under a Transfer Learning scenario using a Q-learning model to enhance resource utilization, the unique characteristics of each VNF are observed as follows:

1) Snort Inline Mode VNF to Virtual Firewall VNF (**Figure 3**): Snort in inline mode functions by detecting and preventing intrusions in real-time, requiring intensive CPU processing due to the need for deep packet inspection (DPI). DPI involves analysing packet contents thoroughly to identify potential threats. Implementing a Q-learning model to transfer knowledge to the Virtual Firewall helps reduce the CPU load of the firewall, possibly by optimizing its operations and processes. However, since the firewall is responsible for managing access control and enforcing policies for network traffic, the link capacity may not decrease significantly. This is because the firewall still requires substantial bandwidth to handle the large volume of data traffic.
2) Virtual Firewall VNF to Snort Inline Mode VNF (**Figure 5**): The Virtual Firewall focuses on regulating the flow of network traffic, acting as a filter that manages and enforces data transmission policies. While it may not demand as much CPU processing as Snort, it places more emphasis on efficiently managing link capacity. When knowledge is transferred via a Q-learning model to Snort in inline mode, the result is improved bandwidth or link capacity management. However, due to Snort's requirement for heavy CPU usage in detecting and preventing threats, it cannot achieve the same level of CPU efficiency as the Virtual Firewall.

The distinct characteristics of Snort in inline mode and the Virtual Firewall highlight their differences in resource management. Snort demands significant CPU resources for deep packet analysis, while the Virtual Firewall primarily consumes high link capacity to control data flow in the network. Transfer Learning enhances the capabilities of each VNF but does not fully address the specific resource constraints that each VNF is most dependent on.

7. Conclusion

This study investigates the application of Reinforcement Learning (RL) models in profiling and optimizing resource allocation for various Virtual Network Functions (VNFs), including Snort (in both passive and inline modes) and virtual firewall (vFW). Our work compares the adaptability of RL-based profilers against traditional machine learning models, particularly Multi-Layer Perceptrons (MLPs). While MLPs may offer better performance in specific scenarios, they lack the flexibility of RL models, which can dynamically adjust to changing conditions without the need for retraining.

We introduced a novel approach to VNF resource allocation using transfer learning in a bi-directional manner—between Snort and vFW—demonstrating that RL models can efficiently transfer knowledge across different VNF types. Our experiments revealed significant resource optimization, with RL models reducing vCPU usage by up to 20% in some scenarios. However, trade-offs were observed, such as lower efficiency in reducing link capacity compared to MLPs, underscoring the complexity of multi-objective optimization. Our research further highlights the flexibility of RL-based profilers in dynamically adjusting CPU, memory, and link capacity allocations based on real-time VNF performance demands. The integration of transfer learning eliminates the need for offline profiling, enabling more efficient deployment of VNFs in dynamic network environments.

Looking ahead, future work will focus on scaling this approach through federated transfer learning, addressing challenges related to scalability and data privacy, particularly in 6G networks. Additionally, we aim to explore more advanced multi-objective optimization techniques, including weighted optimization, to refine resource allocation strategies for diverse VNFs.

8. References

[1] *Network transformation: (orchestration, network and service management framework)*, ETSI, Oct. 2019. [Online]. Available: https://www.etsi.org/images/files/ETSIWhitePapers/ETSI_White_Paper_Network_Transformation_2019_N32.pdf.

[2] X. Zhu and Y. Liu, “Research on the Intelligent Orchestration System of Cloud Network Based on ONAP,” in *2023 2nd International Conference on Big Data, Information and Computer Network (BDICN)*, Xishuangbanna, China, 2023, pp. 280-283, doi: 10.1109/BDICN58493.2023.00065.

[3] G. M. Yilma, Z. F. Yousaf, V. Sciancalepore and X. Costa-Perez, “Benchmarking open source NFV MANO systems: OSM and ONAP,” *Computer communications*, vol. 161, pp. 86–98, 2020, doi: 10.1016/j.comcom.2020.07.013.

[4] B. Varghese and R. Buyya, “Next generation cloud computing: New trends and research directions,” *Future Generation Computer Systems*, vol. 79, pp. 849–861, 2018, doi: 10.1016/j.future.2017.09.020.

[5] A. Mohamed, M. Hamdan, S. Khan, A. Abdelaziz, S. F. Babiker, M. Imran and M. N. Marsono, “Software-defined networks for resource allocation in cloud computing: A survey,” *Computer Networks*, vol. 195, 2021, Art. no. 108151, doi: 10.1016/j.comnet.2021.108151.

[6] A. T. Kyaw, H. T. Zaw, T. Aung, A. H. Maw and M. T. Mon, “Performance evaluation of resource allocation in software defined network,” in *2023 IEEE Conference on Computer Applications (ICCA)*, Yangon, Myanmar, Feb. 27–28, 2023, pp. 153–157, doi: 10.1109/ICCA51723.2023.10181691.

[7] N. Ferdosian, S. Moazzeni, P. Jaisudthi, Y. Ren, H. Agrawal, D. Simeonidou and R. Nejabati, “Profile-based Data-driven Approach to Analyse Virtualised Network Functions Performance,” in

2023 22nd International Symposium on Communications and Information Technologies (ISCIT), Sydney, Australia, 2023, pp. 306–311, doi: 10.1109/ISCIT57293.2023.10376096.

[8] M. Peuster and H. Karl, "Profile your chains, not functions: Automated network service profiling in devops environments," in *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, Berlin, Germany, Nov. 6–8, 2017, pp. 1–6, doi: 10.1109/NFV-SDN.2017.8169826.

[9] M. Peuster and H. Karl, "Understand your chains and keep your deadlines: Introducing time-constrained profiling for nfv," in *2018 14th International Conference on Network and Service Management (CNSM)*, Rome, Italy, Nov. 5–9, 2018, pp. 240–246.

[10] S. Moazzeni, P. Jaisudthi, A. Bravalheri, N. Uniyal, X. Vasilakos, R. Nejabati and D. Simeonidou, "A novel autonomous profiling method for the next-generation nfv orchestrators," *IEEE Transactions on Network and Service Management*, vol. 18, no. 1, pp. 642–655, 2021, doi: 10.1109/TNSM.2020.3044707.

[11] S. V. Rossem, W. Tavernier, D. Colle, M. Pickavet and P. Demeester, "Profile-based resource allocation for virtualized network functions," *IEEE Transactions on Network and Service Management*, vol. 16, no 4, pp. 1374–1388, 2019, doi: 10.1109/TNSM.2019.2943779.

[12] X. Vasilakos, S. Moazzeni, A. Bravalheri, P. Jaisudthi, R. Nejabati and D. Simeonidou, "iON-profiler: Intelligent online multi-objective vnf profiling with reinforcement learning," *IEEE Transactions on Network and Service Management*, vol. 21, no. 2, pp. 2339–2352, 2024, doi: 10.1109/TNSM.2024.3352821.

[13] P. Jaisudthi, S. Moazzeni, X. Vasilakos, R. Nejabati and D. Simeonidou, "i-profiler: Towards multi-objective autonomous vnf profiling with reinforcement learning," in *IEEE INFOCOM 2023 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, Hoboken, NJ, USA, May 20, 2023, pp. 1–6, doi: 10.1109/INFOCOMWKSHPS57453.2023.10225777.

[14] H. Phan, O. Y. Chén, P. Koch, Z. Lu, I. McLoughlin, A. Mertins and M. De Vos, "Towards more accurate automatic sleep staging via deep transfer learning," *IEEE Transactions on Biomedical Engineering*, vol. 68, no. 6, 2020, pp. 1787–1798, doi: 10.1109/TBME.2020.3020381.

[15] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong and Q. He, "A comprehensive survey on transfer learning," *Proceedings of the IEEE*, vol. 109, no. 1, pp. 43–76, 2021, doi: 10.1109/JPROC.2020.3004555.

[16] T. V. Phan, S. Sultana, T. G. Nguyen and T. Bauschert, "Q - TRANSFER: A novel framework for efficient deep transfer learning in networking," in *2020 International Conference on Artificial Intelligence in Information and Communication, ICAIIC 2020*, Fukuoka, Japan, Feb. 19–21, 2020, pp. 146–151, doi: 10.1109/ICAIIC48513.2020.9065240.

[17] W. Mo, Y. -K Huang, S. Zhang, E. Ip, D. C. Kilper, Y. Aono and T. Tajima, "ANN-Based Transfer Learning for QoT Prediction in Real-Time Mixed Line-Rate Systems," in *Optical Fiber Communications Conference and Exposition 2018*, San Diego, CA, USA, Mar. 11–15, 2018, pp. 1–3.

[18] H. McKay, N. Griffiths, P. Taylor, T. Damoulas and Z. Xu, "Bi-directional online transfer learning: a framework," *Annals of Telecommunications*, vol. 75, pp. 523–547, 2020, doi: 10.1007/s12243-020-00776-1.