LEAP: A Lightweight, Explainable, and Programmable Framework for Traffic-Aware Routing in Encrypted SDN Environments

Shaik Luqman Sajid
Amrita Vishwa Vidyapeetham
Amaravati, Andhra Pradesh, India
shaikluqmansajid@gmail.com

Anto Lourdu Xavier Raj Arockia Selvarathinam
Department of Data Science and Analytics
College of Computing
Grand Valley State University
Michigan, USA
arockiaa@mail.gvsu.edu

Ayman Amer
Faculty of Engineering
Zarqa University
Jordan
Aamer@zu.edu.jo

Mohamed Hafez
Faculty of Engineering FEQS
INTI International University
Nilai, Malaysia
Faculty of Management
Shinawatra University
Pathum Thani, Thailand
mohdahmed.hafez@newinti.edu.my

Mohammad Tahidul Islam
School of IT and Engineering
Melbourne Institute of Technology
Melbourne, Australia
tahidcce@gmail.com

Yogesh Kumar
Department of CSE
School of Technology
Pandit Deendayal Energy University
Gandhinagar, India
yogesh.kumar@sot.pdpu.ac.in

Muhammad Umair Manzoor
School of Engineering
RMIT University
Melbourne, Australia
muhammad.umair.manzoor@rmit.edu.au

Muhammad Fazal Ijaz
Centre for Artificial Intelligence Research and Optimization (AIRO)
Design and Creative Technology
Torrens University Australia
Melbourne, Victoria, Australia
muhammadfazal.ijaz@torrens.edu.au

***Abstract*—The paper is about the biggest problem of how to route the planned traffic in a software defined network (SDN) environment, which is modern and impacted by the respectively encrypted traffic and the need for real time, adaptive network control. the main reason for the loss of network visibility is the implementation of secure communication protocols like TLS 1.3 and QUIC with privacy preserving features such as Encrypted Client Hello (ECH), which have made obsolete the traditional visibility tools that were used for intelligent network management. At the same time, machine learning techniques that are focused on the controller lead to a significant decrease in system performance and the operators cannot trust them because they lack transparency. The lightweight, explainable, and programmable (LEAP) framework is presented in this paper to go beyond these drawbacks. LEAP proposes a novel, synergistic architecture that includes a Deep Reinforcement Learning (DRL) agent in the control plane for adaptive, high level routing policy generation; a very efficient, lightweight Gradient Boosting Decision Tree (GBDT) classifier, which is compressed by knowledge distillation and deployed in P4 programmable data planes for line rate traffic identification; and a separate Explainable AI (XAI) module which aims to give people understandable reasons for both classification and routing decisions. By using a modern encrypted traffic dataset, the LEAP framework, tested through extensive emulation in a realistic network environment, shows that it can make the network throughput and end-to-end latency substantially better than the existing baselines. As a result, a new paradigm of efficient, transparent, and autonomous network management has been established by the LEAP framework.**

***Index Terms*—Software-Defined Networking, Traffic Classification, Deep Reinforcement Learning, P4, Explainable AI, Encrypted Traffic, QUIC.**

I. Introduction

Through the division of the control plane from the data plane, SDN alters in a radical way the operation of the network [1]. As a result, the network as a whole is viewed from a single logical centre which makes it possible to program and manage it intelligently and on a large scale. Nevertheless, contemporary networks are characterized by very dynamic and in most cases, unpredictable traffic patterns, which are caused by data intensive applications like video streaming, cloud computing, and the Internet of Things. Such variations in traffic challenge the efficiency of conventional statically configured routing protocols that are not capable of dynamically adjusting to real-time changes in network traffic and which needs a intelligent systems and digital infrastructure that enhance real-time efficiency [2].

A significant challenge in implementing traffic aware routing is the increasing conflict between the privacy requirements

of users and the operator's need for traffic visibility. The challenge, also known as the Visibility Paradox, stems from the adoption of very strong encryption standards such as TLS 1.3 and QUIC that now cover the majority of internet communications. Privacy friendly solutions such as Encrypted Client Hello make it even harder for anyone to get hold of metadata that was previously in plaintext, for instance, information like Server Name Indication. While these changes improve user privacy to a great extent, they make traditional traffic analysis methods such as Deep Packet Inspection almost obsolete [3]. Therefore, operators lack visibility situation where they cannot differentiate between latency sensitive applications like video conferencing and non urgent bulk data transfers. The disappearance of this insight compromises the use of the network functions that are at the core of the QoS enforcement, security monitoring, and adaptive routing.

The solution to this problem lies in the use of machine learning techniques that can identify encrypted flows by relying on statistical and behavioral features that are still accessible even after encryption [4]. The first solution to the visibility problem was to put machine learning models right on the centralized SDN controller. In such a setup, switches send flow statistics or packet heaBut, this design poses a serious scalability issue. Thus, the concept of programmable data planes emerged to bring the processing closer to the traffic source [5]. programmable data planes to bring the processing cloled to the increasing use of more complex models, particularlyously, the quest for greater precision has resulted in the use of more and more complicated models, especially deep learning architectures [6]. While these models attain impressive results, they are usually considered "black boxes," providing very little insight as to how individual choices are arrived at. The absence of explanation converts the trust of the operator into doubt, makes the process of finding errors more difficult, and, thus, creates a huge barrier for the implementation of such systems in situations with high risks, e.g., in the case of cybersecurity forensics, where each decision has to be recorded and explained [7].

To overcome these interconnected issues, the current research work brings forward LEAP, a minimal and programmable framework that offers a consolidated solution for traffic aware routing in contemporary Software Defined Networking environments. LEAP aims at combining three futuristic ideas adaptivity, in network intelligence, and transparency into one single reliable and efficient system.

- **Adaptive Policy:** A Deep Reinforcement Learning agent located in the control plane monitors network wide conditions and figures out abstract routing policies by directly optimizing the performance metrics of the network such as throughput, latency, and packet loss [5].
- **In Network Execution:** A small and very effective Gradient Boosted Decision Tree classifier, along with knowledge distillation, is a product of the compactness of the description and is directly P4 programmable data planes. That allows per flow traffic classification at line rate in the switch pipeline without the need to interact with the controller, thus avoiding controller bottlenecks [8].

This paper continues as follows: Section II discusses the related research, Section III introduces the LEAP framework architecture, Section IV provides the details of the experimental setup, Section V presents the discussion of the results, and Section VI sums up the paper.

II. Background

ML driven traffic engineering in SDN has gone through various phases that aim at finding the right balance between computational efficiency and predictive accuracy. Initial works were mostly based on traditional supervised learning techniques such as Naive Bayes and Random Forest [9]. These models had the advantage of being very fast, but they were not able to tackle the complexity of encrypted traffic of today. To fix that issue, the community turned to Deep Learning models like Convolutional Neural Networks and Recurrent Neural Networks that are able to automatically learn features hierarchies from the raw packet data. Nevertheless, these deep models brought very high computational costs with them. Consequently, it was decided to use highly optimized Gradient Boosted Decision Tree models, that provide a good compromise between accuracy and speed on tabular flow data, which resulted in a resurgence of this method. The latest research is about bridging these two worlds in such a way that small one dimensional convolutional neural networks are used in conjunction with model compression methods like knowledge distillation to have the best of both and still save the resources [10].

The massive use of TLS 1.3 and QUIC has changed traffic classification methods drastically. As there is no payload to inspect, the nowadays methods rely on statistical and time series features taken from packet headers and flow metadata. These encryption resistant indicators are packet length sequences, inter arrival time distributions, flow duration, and any remaining handshake information that is not encrypted. Several papers have reported the success of machine learning models in encrypted traffic classification, including QUIC. A very important point made by the recent literature is that old datasets particularly the ones made before 2018 are not suitable for testing modern classifiers. Such datasets are far from current traffic patterns and thus the real world performance of the models tested on them is going to be bad. Therefore, nowadays realistic datasets have become indispensable for proper evaluation [11].

Besides these changes, two architectural trends have had a major impact on the issue of depositing the intelligence layer in networks. One of them is in network intelligence through programmable data planes. With P4 and similar languages, an operator can tell the machine how to process each packet in a totally new way by providing the code that will be executed on the hardware of the switch [7]. In other words, this means that the machine learning inference can be done at line rate within the data plane which is the reason why the whole process is free of latency and bottlenecks caused by controllerbased

processing. The second trend is distributed and federated learning which solves scalability and data privacy problems. By federated learning, different distributed controllers are able to jointly train one big global model without the need to exchange raw traffic data as what is only exchanged are model updates [12].

After all these achievements there is still boundary which separates most of the big ideas on the topic from each other. To be more explicit, researchers have looked into using Deep Reinforcement Learning for routing, P4 based in network inference and Explainable AI for model interpretability. A common feature of these research papers is that they take only one component into consideration and hence arrive at an isolated solution. There is no work that has come up with a method which not only integrates the three elements i.e. adaptive routing, in network machine learning inference, and end to end explainability, but also is compatible with encrypted and high speed SDN environments. This void is what makes the LEAP framework presented in this paper necessary [13].

III. System Architecture

The LEAP framework separates the computationally heavy learning component from the lighter, real time inference component. The diagram Fig. 1 shows that the system is split between the control plane and the data plane. The control plane houses the reinforcement learning agent as well as the explainability module whereas the data plane is made up of P4 programmable switches with small, distilled Gradient Boosted Decision Tree classifiers.

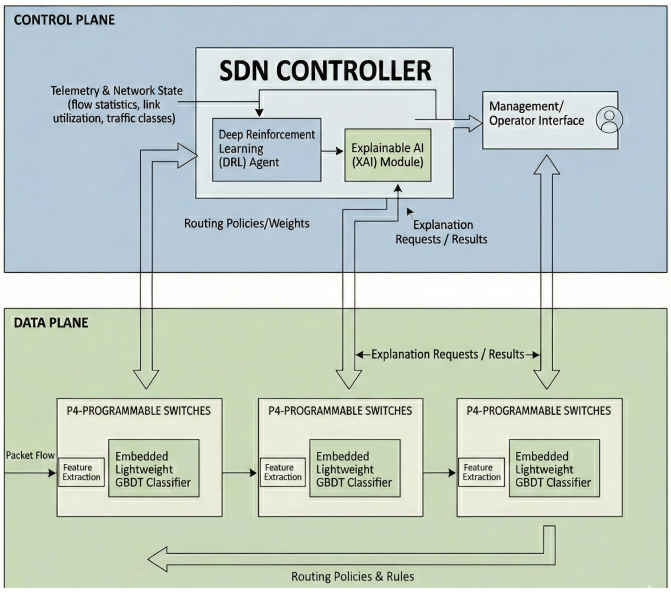

Fig. 1. LEAP architecture showing the interaction between the control plane (DRL agent and XAI module) and the data plane (P4 switches with embedded distilled classifiers).

The operational flow is as follows [11]. The reinforcement learning agent monitors network conditions worldwide and produces a high level routing policy expressed as link weights. On the arrival of a new flow, the first packets are handled by a P4 switch which extracts flow level features. These features are given to the embedded classifier to determine the traffic class at line rate. The switch then implements the appropriate forwarding behavior based on the traffic class as well as the routing policy given by the control plane. The explainability module, thus, is the last component in the system which communicates with both planes to give human interpretable explanations and to gather decision telemetry for continuous monitoring and operator visibility [14].

A. Component 1: The DRL Agent for Adaptive Routing Policy (Control Plane)

Deep Reinforcement Learning is chosen because it is capable of finding near optimal solutions to complex and changing optimization problems without the need for a detailed analytical model of the environment. This makes it a very good fit for adaptive routing in future networks [10].

State, Action, and Reward Design: The performance of a DRL agent strongly depends on how its key components are formulated:

- **State Space (S):** The agent perceives the network through a snapshot of its current status. A Graph Neural Network (GNN) encodes the topology by processing node-level attributes (e.g., queue occupancy) and edge-level indicators (e.g., link utilisation, latency) [11].
- **Action Space (A):** Instead of explicitly choosing full end to end paths, the agent creates a vector of link weights, with one weight for each edge in the graph [10]. The SDN controller is instructed to use these weights to find shortest paths for each traffic class by means of algorithms like Dijkstra's.
- **Reward Function (R):** An agent is controlled by a multi objective reward function. Such a function merges essential performance metrics into a weighted expression that encourages maintaining high throughput (T_h) while reducing latency (L) and packet loss (p) [10].

$$R = \omega_1 \cdot T_h - \omega_2 \cdot L - \omega_3 \cdot p \tag{1}$$

Causal inference methods can be the employed in an offline setting to the pinpoint state action pairs that exert the greatest influence on QoS metrics, thereby providing a more stable and informative reward signal.

DRL Algorithm Selection: This architecture uses Proximal Policy Optimization (PPO) from DeepMind, which is a commonly used policy gradient method, that was chosen for its trade off of stability during training and performance that can be trusted in networking environments that are dynamic [12].

B. Component 2: Distilled GBDT for In-Network Traffic Classification (Data Plane)

Encryption Resilient Feature Engineering: The traffic classifier builds a feature vector based on the first packets of each flow. These features are aimed at being still informative when encryption is applied and consist. The distilled classifier

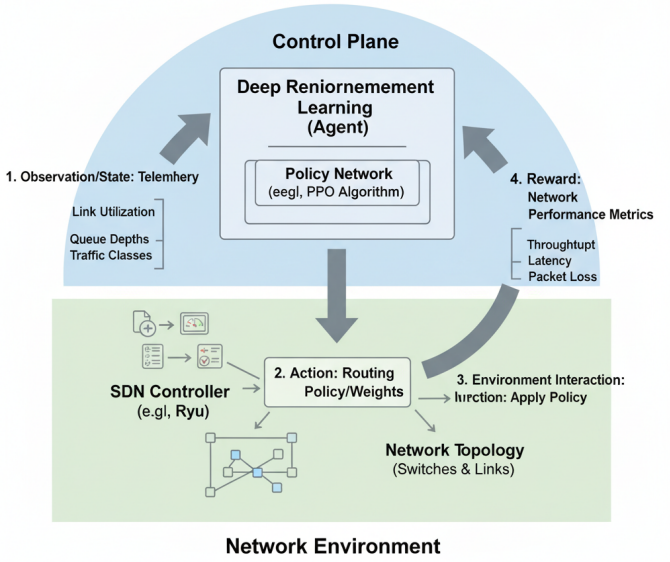

Fig. 2. DRL agent interaction loop. The agent sees the present network state, modifies link weights, and the controller implements these changes. Thereafter, a reward is computed from the resulting network performance.

is changed into a series of if else decision rules that directly correspond to P4 match action tables. A tree node is transformed into a simple comparison operation, and leaf nodes represent class identifiers. Compiler optimizations remove duplicate nodes and share the same subtrees, thus drastically reducing memory consumption. Therefore, the entire classifier is stored without any problem in on chip SRAM and is capable of operating at line rate.

The reason behind our choice of a Gradient Boosted Decision Tree (GBDT) model is mainly due to its impressive capabilities on tabular data. Models like LightGBM and CatBoost deliver precision to a very high degree while consuming very little memory, which makes them suitable for deployment in the data plane of a switch. Therefore, they turn out to be more feasible than standard neural networks in such settings [6]. The respective features comparison is illustrated in Table I.

TABLE I
COMPARATIVE ANALYSIS OF LIGHTWEIGHT GBDT CLASSIFIERS

Feature	XGBoost	LightGBM	CatBoost
Tree Growth	Level-wise	Leaf-wise	Symmetric
Inference Speed	High	Very High	High
Memory Footprint	High	Low	Medium
Categorical Data	Pre-process	Pre-process	Native
Suitability for P4	High	High	High

Model Compression through Knowledge Distillation: In order to have a highly accurate model but still make the model light the we use knowledge distillation [11]. The method is:

- The "teacher" model, which is large and very accurate, for instance, a 1D CNN, is trained offline on the whole dataset.
- The student model which is smaller and faster, here the GBDT, is therefore trained to imitate not only the true labels but also the soft probability outputs generated by the teacher, when the student is learning through the teacher's soft outputs, it can discover very faint features and correlations in the data, which leads to a small model being able to better than a model that is trained only on hard labels by a large factor [12].

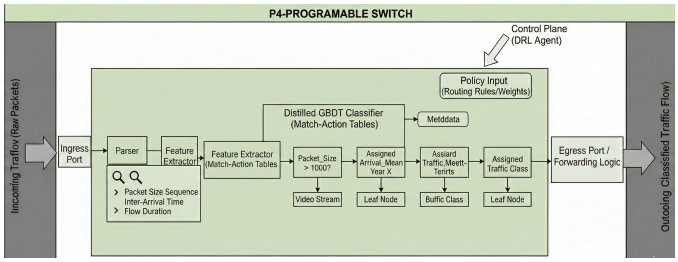

Fig. 3. In Network Classification Workflow in a P4 Data Plane. The packets are parsed are extracted, and the GBDT classifier which was distilled is implemented through a Match's Action Tables for a processing.

The advantage of using this method has two aspects. In the first place, a student model reaches an accuracy level that is very close to the one of a teacher model although its size is drastically smaller. And in the second place, due to a reduced complexity, a student model is a kind of a model which can be deployed in programmable switch pipelines, thus it is a device with very limited memory and processing capabilities. So the system remains strong in classification performance but it is also real time, line rate inference in the data plane which is enabled and thus this is made possible by the transfer of knowledge from a deep neural network to a compact decision tree ensemble [4]. This compromise between accuracy and efficiency is what makes the direct integration of machine learning into high speed network devices possible.

C. Component 3: XAI Module for Decision Transparency

Any automatic system can be considered as a potential solution only if its decision making is trusted by the human operators. To this end, the XAI module is the means by which this transparency is given, and hence it is indispensable not only for debugging, checking and confirming [13].

For global, model-wide explanations: It provides summary plots showing which features are most important overall.

Interpreting DRL Policies: Explaining DRL policies is a nascent research area. Our approach involves applying SHAP to the agent's policy network. This reveals which elements of the network state (e.g., "high utilisation on link A-B") had the most impact on the agent's action (e.g., "decrease weight of link A-B"), providing invaluable insight for operators [14].

IV. IMPLEMENTATION AND EVALUATION

A high fidelity network emulation environment was constructed from open source tools that are broadly accessible. The testbed features an SDN controller, P4 programmable switches, and virtual hosts linked through a realistic ISP scale

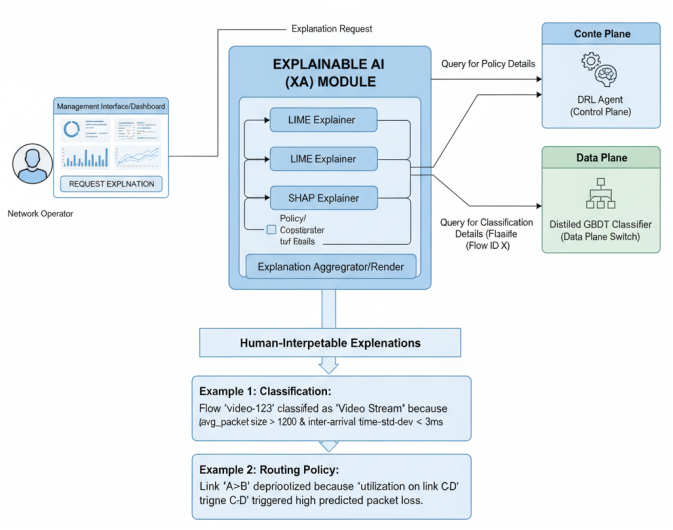

Fig. 4. Interaction of the XAI module and the explanation flow. An operator asks for an explanation, the XAI module gets the data from the DRL agent or GBDT classifier and delivers a human understandable account.

topology. Traffic generation is done with replayed encrypted traces to guarantee that the evaluation is accurate and can be repeated. The main elements and configuration features of the experimental setup are outlined in Table II.

TABLE II
KEY PARAMETERS OF THE EXPERIMENTAL SETUP

Component	Specification
Topology	AT&T North America (25 nodes, 56 links)
Controller	Ryu v4.34 with OpenFlow 1.3
Switches	P4 BMv2 Software Switches
Hosts	Mininet Virtual Hosts (Ubuntu 20.04)
DRL Agent	PPO (LR: 1×10^{-4}, $\gamma : 0.99$, $\lambda : 0.95$) [1], [10]
Traffic Mix	40% Video, 20% VoIP, 30% Web, 10% Bulk
Dataset Used	CESNET-QUIC22 [2]

The evaluation was performed with CESNET QUIC22 dataset [2], which is a large scale encrypted traffic trace collected from the live ISP backbone. The dataset is in line with the latest research which points out that there is a need for new traffic traces because the old ones cannot represent the current QUIC dominated environments. The traffic profiles were extracted and replayed with high speed generators in order to create realistic network conditions [14].

LEAP was put to test against three baselines in order to quantify its performance. Static Routing (OSPF) is the first one and it stands for a traditional non adaptive routing approach. The second baseline is Controller Centric Machine Learning where a Gradient Boosted Decision Tree classifier is run on the SDN controller and switches forward flow statistics for external inference thus the effect of the controller bottleneck is measured. The third baseline is a Deep Reinforcement Learning only approach that implements adaptive routing policies but does not have granularity of the in network traffic classification [13].

The evaluation takes into account a wide range of metrics that are used to quantify the network's efficiency. These include average network throughput, end to end latency, jitter, and packet loss. Besides that, classifier size, switch memory footprint, and per flow inference time were also examined to check if the in network deployment is possible. Collectively, these metrics provide an in depth picture of how LEAP behaves under real and heavy load scenarios [10].

V. RESULTS AND DISCUSSION

The performance results, shown in Table III, clearly indicate that the LEAP framework is significantly better. When the network was heavily loaded, LEAP was able to deliver an average throughput of 18.2 Gbps, which is 21% higher than the DRL only baseline and 48% higher than OSPF. Also, LEAP was able to keep the average latency at 28.5 μs, which is 35% less than the Controller Centric ML approach that has controller bottlenecks. These improvements are due to the combination of adaptive reinforcement learning policies and the in network classifier, which together allow for fine grained, real time traffic differentiation at line rate.

TABLE III
END-TO-END PERFORMANCE COMPARISON. RESULTS ARE SHOWN AS MEAN $\pm$ STANDARD DEVIATION ACROSS 10 EXPERIMENTAL RUNS.

Method	Load	Avg. Throughput (Gbps)	Avg. Latency (ms)	Packet Loss (%)
LEAP Framework	Low	9.8 ± 0.2	15.1 ± 1.1	< 0.01
	Medium	15.1 ± 0.4	21.3 ± 2.4	0.04
	High	18.2 ± 0.7	28.5 ± 3.8	0.09
OSPF	Low	8.5 ± 0.3	22.4 ± 1.5	0.15
	Medium	12.1 ± 0.6	35.8 ± 4.1	0.88
	High	12.3 ± 1.1	51.2 ± 7.2	2.13
Controller-Centric ML	Low	9.5 ± 0.2	20.5 ± 1.9	< 0.01
	Medium	14.2 ± 0.5	31.7 ± 3.5	0.11
	High	16.5 ± 0.9	43.8 ± 5.9	0.45
DRL-Baseline	Low	9.7 ± 0.2	16.3 ± 1.3	< 0.01
	Medium	14.8 ± 0.4	23.9 ± 2.8	0.06
	High	15.0 ± 0.8	33.1 ± 4.5	0.27

Additionally, efficiency improvements were observed in the model distillation process. The initial 1D CNN teacher model was 6.2 MB, while the distilled LightGBM student model was only 112 KB in its compiled version more than a 98% reduction. The average per flow inference latency recorded on the BMv2 software switch was only 15 μs, which is substantially faster than the Controller Centric ML baseline that took more than 25 μs for a classification round trip. These findings serve as a strong indication that the relocation of distilled models to the data plane is the solution to the controller bottleneck and that it is feasible to have scalable, line-rate traffic awareness.

Qualitative insights from the Explainable AI module greatly support LEAP's behavior. In one case, an operator got a LIME explanation that a certain flow was identified as a Video Stream with 98% confidence because of a high bitrate packet size sequence thus, the operator instantly verified the reason for the traffic surge. In a different situation, SHAP analysis showed

that high utilizations on link C D were the main reason that influenced the DRL agent's decision to reroute the traffic. These instances exemplify how the XAI component changes LEAP from a black box to a transparent and understandable system that allows operators to troubleshoot issues and gain trust in automated decisions [11].

The aspects of scalability and generalisation were tested as well. By the use of Graph Neural Networks, LEAP is allowed to work over different topologies, and the entirely distributed in network inference method removes the centralized bottlenecks. The classifier exhibited fair generalisation to zero day applications, in most cases, it mapped them into more general but still semantically correct categories like "Bulk Transfer" [12]. However, there are still a few limitations: the evaluation was done in an emulated environment, P4 imposes limitations on the models' complexity that can be deployed, and the robustness of the system against adversarial manipulations has not been tested. These issues pinpoint the significant research areas for the future.

VI. Conclusion

This research presented LEAP, a single framework for intelligent, traffic aware routing in Software Defined Networking, achieved by fusing adaptive Deep Reinforcement Learning, a streamlined P4 based classifier, and a transparent Explainable AI module. With this composition, LEAP is able to tackle the main issues of adaptability, performance, and trust in encrypted networks of the future. The experimental results attest that LEAP is to a large extent better than the conventional baselines as it is able to perform fine grained, real time routing decisions at line rate while still keeping the latency low and the throughput high. Besides that, its capability to offer straightforward, human interpretable explanations can be seen as a significant step towards the deployment of reliable autonomous network management in practice. In sum, this research demonstrates that decoupling learning from inference and utilizing knowledge distillation makes networks not only highly intelligent but also resource efficient, thus it is a strong step forward for the next generation of scalable and explainable SDN systems.

References

[1] M. Lopez-Martin, B. Carro, and A. Sanchez-Esguevillas, "Deep learning for encrypted traffic classification: An overview," IEEE Communications Magazine, vol. 57, no. 6, pp. 40–47, Jun. 2019, doi: 10.1109/MCOM.2019.1800685.

[2] J. Zhang, S. Maidin, and D. A. Dewi, "BHE+ALBERT-Mixplus: A distributed symmetric approximate homomorphic encryption model for secure short-text sentiment classification in teaching evaluations," Symmetry, vol. 17, no. 6, p. 903, 2025, doi: 10.3390/sym17060903.

[3] X. Wang, W. Wang, C. Wang, and X. Li, "Reinforcement learning-based SDN routing scheme empowered by causality detection and GNN," Frontiers in Computational Neuroscience, vol. 18, 2024, doi: 10.3389/fncom.2024.1393025.

[4] Z. A. Al-Saffar, H. A. Al-Raweshidy, and M. A. Al-Sumaidaee, "Applying federated learning in software-defined networks: A survey," Symmetry, vol. 14, no. 2, p. 195, Jan. 2022, doi: 10.3390/sym14020195.

[5] H. Attar, "Joint IoT/ML platforms for smart societies and environments: A review on multimodal information-based learning for safety and security," ACM Computing Surveys, vol. 55, no. 3, pp. 1–36, 2023, doi: 10.1145/3485832.

[6] P. N. Srinivasu, R. Panigrahi, A. Singh, and A. K. Bhoi, "Probabilistic buckshot-driven cluster head identification and accumulative data encryption in WSN," Journal of Circuits, Systems and Computers, vol. 31, no. 17, p. 2250303, 2022, doi: 10.1142/S0218126622503036.

[7] Y. He, Y. Liu, C. Li, and F. R. Yu, "Federated learning augmented cybersecurity for SDN-based aeronautical communication networks," Electronics, vol. 14, no. 8, p. 1535, 2025, doi: 10.3390/electronics14081535.

[8] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescape, "Balancing complexity and performance in convolutional neural network models for QUIC traffic classification," Sensors, vol. 24, no. 12, p. 3949, Jun. 2024, doi: 10.3390/s24123949.

[9] S. Li, M. Chen, and X. Wang, "Deep reinforcement learning-based routing on software-defined networks," IEEE Transactions on Network and Service Management, vol. 19, no. 3, pp. 2487–2501, Sep. 2022, doi: 10.1109/TNSM.2022.3214854.

[10] M. A. Al-Quraan, "Explainable AI for forensic analysis: A comparative study of SHAP, LIME, and Grad-CAM," Applied Sciences, vol. 15, no. 13, p. 7329, 2025, doi: 10.3390/app15137329.

[11] R. K. Shrestha, S. H. Kim, and S. G. Choi, "Lightweight models for traffic classification: A two-step distillation approach," IEEE Access, vol. 10, pp. 84124–84136, 2022, doi: 10.1109/ACCESS.2022.3197669.

[12] T. T. T. Nguyen, V. H. Nguyen, and T. V. Phan, "A framework for in-network inference using P4," in Proc. Int. Conf. Advanced Technologies for Communications (ATC), 2024, doi: 10.1109/ATC59863.2024.10714155.

[13] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," arXiv preprint arXiv:1707.06347, 2017, doi: 10.48550/arXiv.1707.06347.

[14] T. Wu, Y. Zhang, and H. Luo, "Robust network traffic classification," IEEE/ACM Transactions on Networking, vol. 18, no. 5, pp. 1481–1494, Oct. 2010, doi: 10.1109/TNET.2010.2048541.