IJIAC
Vol. 06, No 01, pp. 46–64.
Published by TensorGate
OPEN ACCESS
International Journal of Intelligent
Automation and Computing

An AI-Driven Framework for Dynamic Resource Allocation in Software-Defined Networking to Optimize Cloud Infrastructure Performance and Scalability

Kaushik Sathupadi Staff Engineer, Google LLC, Sunnyvale, CA

01, March, 2023

Recieved: 02, Jan, 2023. | Revised: 26 Feb 2023. | Published: 01 March 2023

Abstract

The rise of cloud computing and the proliferation of highly dynamic workloads present significant challenges for efficient resource management in cloud infrastructures. Existing resource allocation methods are ill-equipped to handle the real-time demands of modern cloud services, especially in scenarios that require low latency, high throughput, and scalability. Software-Defined Networking (SDN) is a promising approach to address these challenges by decoupling the control plane from the data plane, enabling more flexible and programmable network management. However, the static nature of conventional SDN architectures limits their capacity to react dynamically to fluctuating cloud workloads. This paper proposes a novel framework integrating Artificial Intelligence (AI) with SDN for dynamic resource allocation in cloud environments. The study focuses on the development of AI-driven resource allocation algorithms, using machine learning (ML) techniques such as reinforcement learning (RL), deep learning (DL), and predictive analytics to optimize network performance, reduce latency, and improve overall service quality. The proposed AI-SDN architecture dynamically adjusts network resources based on real-time data, network state predictions, and workload analysis, enabling more agile, responsive, and efficient cloud infrastructure. This paper further explores the architectural considerations, algorithmic design, and performance metrics of AI-driven SDN systems, demonstrating the porspective of AI to transform SDN into a fully autonomous network management paradigm capable of meeting the demands of modern cloud ecosystems.

Keywords: *Artificial Intelligence, Cloud Computing, Machine Learning, Resource Allocation, Software-Defined Networking, Workload Management*

1 Introduction

1.1 Cloud Resource Allocation Challenges

The rise of cloud computing has significantly altered how computational resources are provisioned, managed, and consumed. Cloud platforms now offer a broad spectrum of services, ranging from basic storage and compute resources to sophisticated application environments. However, the inherent variability of cloud workloads, driven by factors such as fluctuating user demand, application behavior, and the diversity of service-level agreements, poses substantial challenges for effective resource management (Lin et al., 2010). This is evident in scenarios that require real-time responsiveness and adaptability.

Workload variability is a key challenge in cloud environments. These systems must contend with unpredictable and fluctuating workloads, often requiring real-time adjustments to resource allocation. Traditional approaches, which rely on static or semi-static resource provisioning, often prove insufficient in such dynamic environments. Over-provisioning leads to inefficiencies and wasted resources, while under-provisioning can result in performance degradation, negatively impacting the user experience. Effective management in such environments necessitates the development of resource allocation strategies that can dynamically adapt to changes in workload demand, ensuring both resource efficiency and system performance (Mireslami et al., 2019).

Applications with stringent latency requirements present another significant challenge. Real-time video conferencing, virtual and augmented reality, and IoT-based systems demand fast and reliable data processing and transmission. In these cases, even minor delays can lead to unacceptable performance degradation. Static resource allocation mechanisms struggle to meet the demands of such latency-sensitive applications, as they lack the capability to adjust resources in real-time to handle sudden increases in load or other unexpected conditions. To meet the needs of these applications, more adaptive resource management strategies are required, enabling the system to maintain low-latency performance under variable workloads.

Resource contention is an additional concern in multi-tenant cloud environments, where multiple applications and services may compete for the same underlying infrastructure. This competition can result in inefficient resource utilization and degraded Quality of Service. In such environments, ensuring fair and efficient resource distribution becomes increasingly complex, especially as workloads scale. Without intelligent resource management strategies, resource contention can lead to significant performance bottlenecks, undermining the advantages offered by cloud infrastructures (Liu et al., 2017).

Addressing these challenges requires a shift towards more intelligent, agile, and scalable approaches to resource management, in software-defined networking (SDN)-enabled cloud systems. While SDN provides a degree of network agility and programmability, it often lacks the autonomous decision-making capabilities necessary for real-time, dynamic resource allocation. Integrating advanced decision-making frameworks into SDN-enabled cloud environments can help overcome these limitations, allowing for the efficient handling of dynamic workloads, minimizing latency, and mitigating resource contention.

In this context, machine learning and artificial intelligence are increasingly seen as promising tools for enhancing the resource management capabilities of cloud systems. By leveraging predictive models, these systems can anticipate changes in workload demand, optimize resource allocation in real-time, and ensure that resources are used efficiently without compromising on performance. This approach not only addresses the inherent variability in cloud workloads but also enables the system to make more informed decisions about how to allocate resources in multi-tenant environments, where resource contention is a major concern.

Furthermore, the adoption of autonomous decision-making capabilities in SDN-enabled cloud systems can significantly improve the handling of latency-sensitive applications. By allowing the system to adapt to real-time changes in workload and application demands, these advanced techniques ensure that resources are allocated precisely where they are needed, minimizing latency and ensuring that performance remains consistent, even under highly variable conditions.

1.2 Software-Defined Networking (SDN)

Software-Defined Networking (SDN) represents a significant departure from traditional network architectures by introducing a framework that decouples the network's control plane from its data plane (Paul et al., 2014). This separation allows for centralized management of network devices and introduces programmability, which is critical for handling complex, large-scale networks in a more flexible and dynamic manner. The core principles of SDN—centralized control, programmability, and the decoupling of data and control planes—have enabled more efficient traffic management, simplified network configuration, and facilitated network innovation. However, while SDN has delivered significant advantages, it faces technical challenges when applied to environments with highly dynamic and unpredictable traffic patterns, such as modern cloud infrastructures.

At the heart of SDN is the centralized network control paradigm. In a traditional network, the control logic is embedded within each individual network device, which operates independently based on locally stored routing tables or policies. This distributed control plane model lacks a holistic view of the network, making it difficult to optimize traffic flow, detect bottlenecks, or respond to network faults in real-time. SDN overcomes this limitation by centralizing the control plane in an SDN controller, which maintains a global view of the entire network topology, traffic statistics, and resource status. From this centralized position, the controller can make informed decisions about how to route traffic, allocate bandwidth, and manage resources across the entire network. By dynamically programming the underlying network devices (switches and routers), the controller enables rapid, real-time reconfiguration of the network to accommodate changing demands (Li et al., 2017a).

The programmability offered by SDN is achieved through the use of well-defined APIs such as OpenFlow, which allows the SDN controller to communicate with network devices and issue instructions for modifying forwarding tables, adjusting traffic flows, or implementing security policies. This level of programmability enables network operators to automate complex tasks that would otherwise require manual intervention, such as rerouting traffic in response to congestion or implementing Quality of Service (QoS) policies for specific applications. Additionally, programmable interfaces allow for the deployment of custom traffic engineering algorithms, security policies, and resource management strategies that are tailored to specific network requirements (Bhat and Kavasseri, 2023). The flexibility offered by this programmability is crucial for environments where traffic patterns are highly variable or unpredictable, as is often the case in cloud data centers and multi-tenant infrastructures.

A critical aspect of SDN's architecture is the decoupling of the data and control planes. In traditional networks, the control plane (responsible for making routing and forwarding decisions) and the data plane (responsible for forwarding packets) are tightly integrated within each device. This integration complicates network management and limits the ability to make granular changes to traffic handling. SDN separates these two functions, delegating packet-forwarding responsibilities to the data plane while moving all control logic to the centralized SDN controller (Jain and Paul, 2013). This separation enables more agile and fine-grained control over network traffic. For example, traffic flows can be dynamically adjusted to optimize for latency, bandwidth, or specific application requirements without reconfiguring the physical hardware (Jani, 2022). Furthermore, network policies can be updated centrally and propagated across the network in real time, allowing for immediate responses to network events, such as link failures or traffic surges.

However, despite these advantages, traditional SDN architectures encounter significant limitations in environments where traffic patterns and network conditions are rapidly changing. Cloud environments, characterized by dynamic and often unpredictable workloads, require real-time resource allocation and traffic engineering that static, pre-defined policies cannot adequately support. Traditional SDN relies on rule-based decision-making, where policies and forwarding rules are manually defined or statically configured in advance. This approach becomes inefficient as network conditions evolve, leading to suboptimal resource allocation, increased latency, and reduced overall performance (Li et al., 2017b).

To address these limitations, artificial intelligence (AI) and machine learning (ML) have been proposed as solutions for enhancing the decision-making capabilities of SDN. By integrating AI into the SDN control plane, it becomes possible to make more adaptive and predictive decisions regarding resource management, traffic routing, and congestion control. AI-driven SDN systems can continuously monitor network performance data and apply predictive algorithms to forecast traffic trends, allowing the controller to proactively reconfigure network paths and allocate resources in anticipation of future demand. For example, machine learning models can analyze historical traffic patterns, user behavior, and application performance metrics to predict impending traffic congestion or resource contention, enabling the SDN controller to adjust bandwidth allocation or reroute traffic before performance degradation occurs (Baucke et al., 2013).

Furthermore, reinforcement learning (RL) algorithms can be employed within the SDN control plane to dynamically optimize network configurations based on real-time feedback. RL-based controllers learn by interacting with the network and receiving feedback on the success of different traffic engineering strategies. Over time, the controller refines its decision-making process, enabling it to handle complex and highly dynamic traffic patterns without human intervention. For instance, in a cloud environment with variable application workloads, an RL-based SDN controller could autonomously adjust routing paths, bandwidth allocations, and load balancing strategies in response to changing demand, ensuring that the network operates efficiently under all conditions.

Additionally, the integration of AI into SDN enables more advanced resource management techniques. Traditional SDN systems rely on static resource allocation policies, which are often based on conservative estimates of future traffic demands. This leads to either over-provisioning, where resources are allocated but remain underutilized, or under-provisioning, where resource shortages lead to performance bottlenecks. AI-driven SDN systems can leverage real-time traffic data and predictive analytics to allocate resources more precisely, ensuring that resources are allocated efficiently in response to actual network conditions. For example, in multi-tenant cloud environments, AI-based controllers can dynamically adjust virtual machine placement, bandwidth allocation, and storage distribution to minimize resource contention and maximize overall performance.

AI enhances SDN's ability to handle security and anomaly detection. In traditional networks, security policies are often static and predefined, leaving networks vulnerable to novel or evolving threats. AI-based SDN controllers can continuously analyze network traffic for anomalous patterns, such as distributed denial-of-service (DDoS) attacks, unauthorized access attempts, or abnormal traffic spikes, and automatically take corrective action by reconfiguring network paths, isolating compromised devices, or applying security policies in real-time. This capability is especially important in cloud environments, where security threats can propagate quickly and affect multiple tenants simultaneously (Dai et al., 2016).

Technique	**Application**	**Concepts**	**In SDN**
Reinforcement Learning (RL)	Real-Time Network Optimization	Defines state based on traffic load, latency, adjusts routing, bandwidth, and resources based on QoS feedback.	Enables dynamic, real-time adaptation, optimal resource allocation, and load balancing.
Deep Learning (DL)	Traffic Prediction and Classification	Predicts traffic demands and classifies data types, allowing for proactive resource allocation.	Prevents congestion, enhances QoS by anticipating and addressing traffic demands.
Predictive Analytics	Proactive Resource Allocation	Forecasts network congestion, workload spikes, and failures, optimizing resource distribution.	Improves network reliability, prevents service disruptions, ensures resource availability.

Table 1: AI Algorithms for Resource Allocation in SDN-Enabled Cloud Environments

2 Architectural Design of AI-Based SDN Systems

The architecture of an AI-enhanced Software-Defined Networking (SDN) controller represents a sophisticated fusion of artificial intelligence models with the SDN control plane, aimed at enabling intelligent, adaptive decision-making capabilities that extend beyond traditional static, rule-based frameworks. This architecture is composed of several interdependent layers and components, each serving a distinct function in facilitating real-time, AI-driven control of the network. At its core, the AI-SDN controller's design integrates advanced data collection mechanisms, AI processing capabilities, decision-making logic, and flexible interfaces for seamless communication with both the network infrastructure and higher-level orchestration systems.

2.1 Data Collection Layer

The data collection layer of the AI-SDN controller functions as a telemetry system that gathers real-time data from the network, which serves as input for the AI models tasked with analyzing network conditions and making decisions. This layer aggregates network traffic statistics, tracks link utilization rates, and monitors changes in network topology. The collection of these metrics allows the AI-SDN controller to maintain an accurate and current understanding of the network's state, enabling it to respond appropriately to varying conditions.

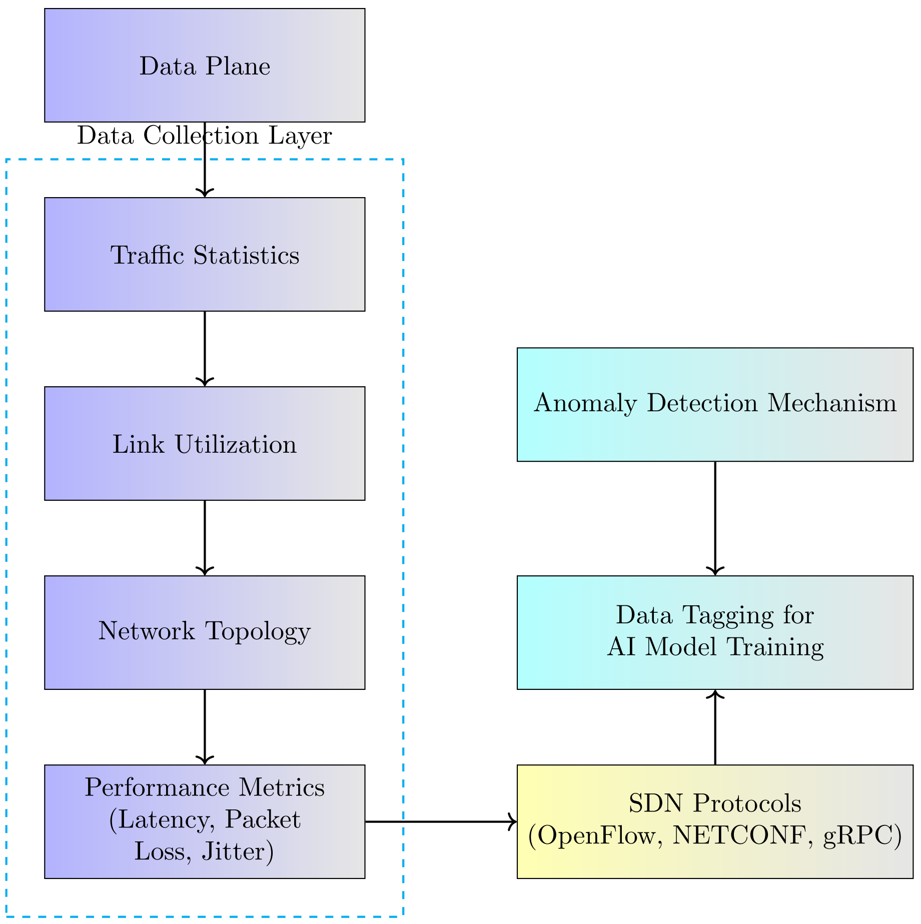
Data Plane
Data Collection Layer
Traffic Statistics
Link Utilization
Network Topology
Performance Metrics (Latency, Packet Loss, Jitter)
Anomaly Detection Mechanism
Data Tagging for AI Model Training
SDN Protocols (OpenFlow, NETCONF, gRPC)

Figure 1: Data Collection Layer Architecture in AI-SDN

One critical aspect of this layer is the monitoring of Quality of Service (QoS) metrics, which include latency, jitter, and packet loss. These metrics are essential for assessing network performance. Latency measures the time delay between sending and receiving a packet, while jitter refers to variations in packet arrival times, which can affect applications sensitive to timing. Packet loss, which measures the percentage of data packets that fail to reach their destination, can degrade network performance and service quality. The collection of such data allows the AI models to evaluate the network's health and performance more effectively and make decisions based on current conditions.

The data acquisition process is enabled by standardized protocols such as OpenFlow and NETCONF. These protocols are integral to the functionality of Software-Defined Networking (SDN) because they allow for the control and monitoring of network devices in a centralized manner. OpenFlow enables the controller to program network devices dynamically, directing how traffic should be routed through the network. NETCONF, on the other hand, provides mechanisms for configuring, monitoring, and managing network devices in a standardized way. In addition to these SDN-specific protocols, the data collection layer often utilizes modern telemetry frameworks like gRPC (Google Remote Procedure Call), which allows for efficient and scalable data exchange between devices. These protocols and frameworks collectively ensure that the data collection layer can retrieve detailed, high-fidelity data in real time, making the information available to the AI models for analysis and decision-making.

In more advanced implementations, the data collection layer may also include systems for anomaly detection. These systems continuously analyze network data to identify deviations from expected behavior. For instance, an anomaly could be an unexpected increase in packet loss or a sudden change in traffic patterns that could indicate network congestion, a potential security breach, or hardware malfunctions. Anomaly detection mechanisms can apply statistical models, rule-based detection, or machine learning algorithms to detect these irregularities. Statistical models may use historical data to define normal behavior, while machine learning techniques can identify complex patterns that are not easily detectable through traditional methods. Once an anomaly is detected, it is flagged for further investigation or immediate action by the AI-SDN controller.

The integration of anomaly detection serves multiple purposes. First, it helps maintain the stability and security of the network by identifying potential issues early. For example, if an anomaly is detected in the form of unusual traffic patterns that suggest a Distributed Denial of Service (DDoS) attack, the AI-SDN controller can quickly intervene to mitigate the attack by rerouting traffic or applying rate-limiting policies. Additionally, anomalies can point to hardware failures, such as a malfunctioning switch or router, which could degrade network performance. Detecting these issues promptly allows network operators to take corrective action before they lead to more significant problems, such as outages or service degradation.

Furthermore, anomaly detection systems contribute to the development and refinement of AI models. By continuously monitoring network activity and flagging unusual events, the system generates labeled datasets that can be used to train machine learning models. These labeled datasets are crucial for supervised learning, where the AI model learns to classify and predict future anomalies based on past data. For instance, if the system identifies a pattern that consistently leads to packet loss, that pattern can be used to train the model to recognize and predict similar events in the future. As the model encounters more data over time, it can improve its accuracy in detecting and predicting network issues, thereby enhancing the overall performance and resilience of the network.

In terms of scalability, the data collection layer must be designed to handle the growing size and complexity of modern networks. As networks expand and data volumes increase, the system must be capable of aggregating and processing vast amounts of information without becoming a bottleneck. To achieve this, distributed data collection architectures are often employed, where multiple telemetry agents are deployed across the network to collect data in parallel. These agents then forward the data to a central controller for analysis. Such an approach allows the system to scale efficiently, ensuring that even in large and complex network environments, the AI models have access to the most up-to-date and comprehensive data.

Another important consideration is the timeliness of data collection. In network environments where conditions can change rapidly, such as during a traffic surge or security incident, it is critical that the data collection layer operates with minimal latency. This means that data must be collected, processed, and made available to the AI models in near real-time. Protocols like gRPC are well-suited for this purpose because they allow for fast, efficient communication between telemetry agents and the central controller. Additionally, techniques such as edge computing, where some processing is performed closer to the data source, can be employed to reduce the time it takes to process and analyze data. By distributing processing tasks closer to the network edge, the system can respond more quickly to changes in network conditions.

In terms of security, the data collection layer plays a role in safeguarding the network by ensuring that telemetry data is transmitted securely. This involves using encryption protocols to protect data as it travels between network devices and the controller. Without proper security measures, telemetry data could be intercepted or tampered with, compromising the integrity of the AI models' decision-making process. Therefore, secure transmission protocols such as Transport Layer Security (TLS) are often implemented to protect data during transmission. Additionally, authentication mechanisms ensure that only authorized devices and entities can access the telemetry system, preventing unauthorized access to sensitive network data.

2.2 AI Processing Layer

The AI processing layer forms the computational core of the AI-SDN controller, executing sophisticated AI algorithms on data obtained from the network. This layer is tasked with addressing complex network management challenges, such as traffic classification, resource optimization, and fault prediction, by employing a range of AI models specifically designed for these tasks. Each model is tailored to solve distinct problems in real-time, enhancing the controller's ability to optimize network performance and prevent issues proactively.

For traffic classification and analysis, the AI-SDN controller utilizes Deep Learning (DL) models, such as Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks, to process network traffic data. CNNs are effective in recognizing spatial patterns within network traffic data, making them highly suitable for distinguishing different types of traffic, such as video streaming, Voice over IP (VoIP) communications, or large file transfers. The ability of CNNs to classify traffic in real time helps in the prioritization of bandwidth for critical applications, while minimizing the impact of less sensitive data flows. For example, when analyzing packet flow data, the CNN can classify a packet stream p_i as belonging to a certain class (e.g., video streaming) based on spatial features, ensuring appropriate resource allocation (Fard et al., 2020).

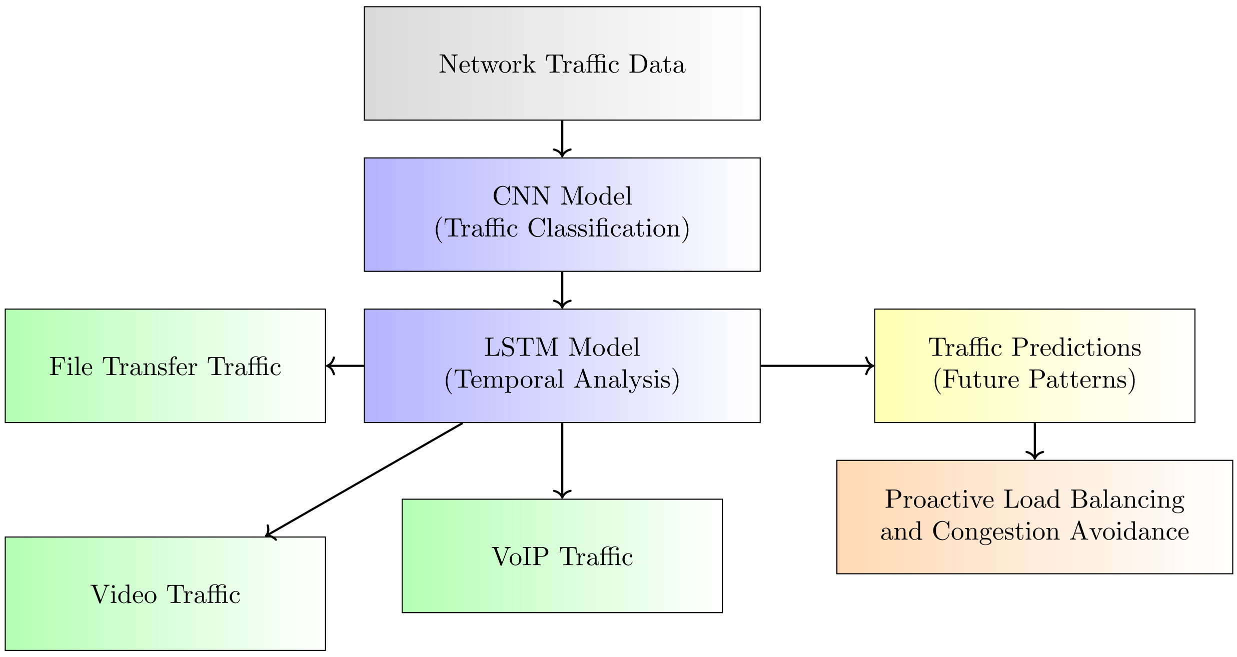
Network Traffic Data
CNN Model (Traffic Classification)
LSTM Model (Temporal Analysis)
File Transfer Traffic
Traffic Predictions (Future Patterns)
Proactive Load Balancing and Congestion Avoidance
Video Traffic
VoIP Traffic

Figure 2: Traffic Classification and Analysis Using DL Models in AI-SDN

LSTM networks, on the other hand, are designed to handle sequential and temporal data, making them ideal for analyzing and predicting time-series traffic patterns. The LSTM's capability to maintain memory of past network conditions allows it to forecast future traffic loads by learning temporal correlations. Let $T(t)$ represent the traffic load at time t, and using the historical traffic data, the LSTM model can predict future traffic loads $T(t+k)$ over a prediction horizon k. These predictions enable the controller to make proactive adjustments to network configurations, such as rerouting traffic or preemptively allocating bandwidth to prevent congestion. This predictive capability is critical in managing network resources, in dynamic and high-traffic environments, where efficient load balancing and congestion avoidance are paramount.

In the area of resource optimization, Reinforcement Learning (RL) techniques, such as Deep Q-Networks (DQNs) and Proximal Policy Optimization (PPO), are leveraged to dynamically adjust network parameters to improve performance. RL operates by modeling network management as a Markov Decision Process (MDP), where the RL agent interacts with the network environment, learns from its state, and takes actions aimed at optimizing a cumulative reward function. The state of the network at time t, denoted s_t, includes metrics such as current traffic loads, network utilization, and link performance. Based on the state s_t, the RL agent selects an action a_t, such as adjusting routing paths or reallocating bandwidth, that maximizes the expected cumulative reward over time.

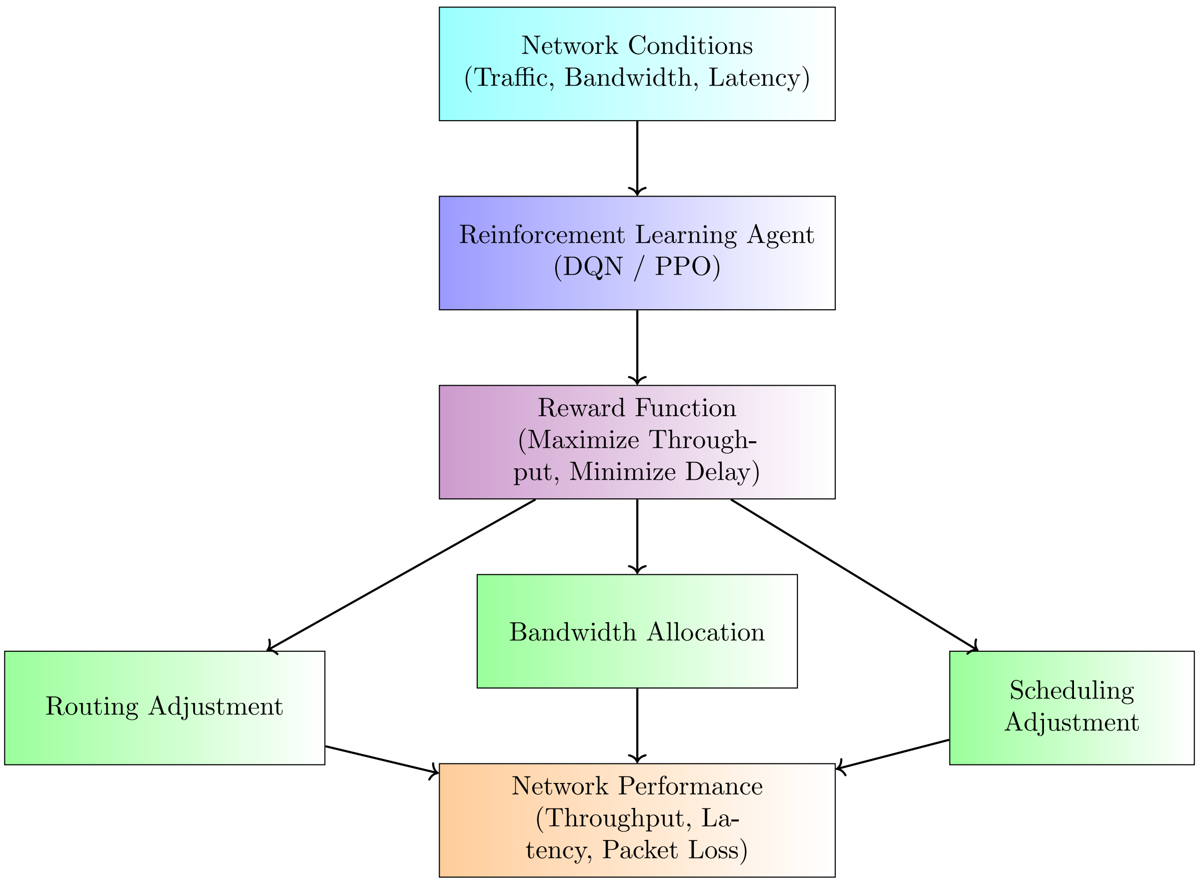
Network Conditions
(Traffic, Bandwidth, Latency)
Reinforcement Learning Agent
(DQN / PPO)
Reward Function
(Maximize Throughput, Minimize Delay)
Routing Adjustment
Bandwidth Allocation
Scheduling Adjustment
Network Performance
(Throughput, Latency, Packet Loss)

Figure 3: Resource Optimization Using RL Agents in AI-SDN

The reward function R_t at time t can be formulated to balance multiple performance objectives. For example, the reward may be defined as:

$$R_t = \alpha \cdot \mathrm{Thr}(t) - \beta \cdot L(t) - \gamma \cdot \Delta(t),$$

where $\mathrm{Thr}(t)$ is the network throughput at time t, $L(t)$ represents packet loss, $\Delta(t)$ denotes latency, and α, β, and γ are weighting factors that balance the importance of maximizing throughput, minimizing packet loss, and reducing latency. The RL agent iteratively refines its policy based on feedback from the network, allowing it to adjust network configurations such as routing paths, bandwidth allocation, and traffic scheduling in real time. Over time, the agent becomes more adept at optimizing network performance, even under varying traffic conditions and unpredictable demand patterns.

For predictive analytics and fault management, the AI-SDN controller utilizes machine learning (ML) models such as Random Forests and Gradient Boosting Machines (GBMs) to identify patterns that signal potential network issues, such as performance degradation or equipment failures (Luo and Chen, 2013). These models are trained on historical network data, including metrics such as packet loss, latency, and congestion levels, to detect anomalies or conditions that precede faults. By learning the relationships between these metrics and network failures, the models can forecast the likelihood of a failure event F.

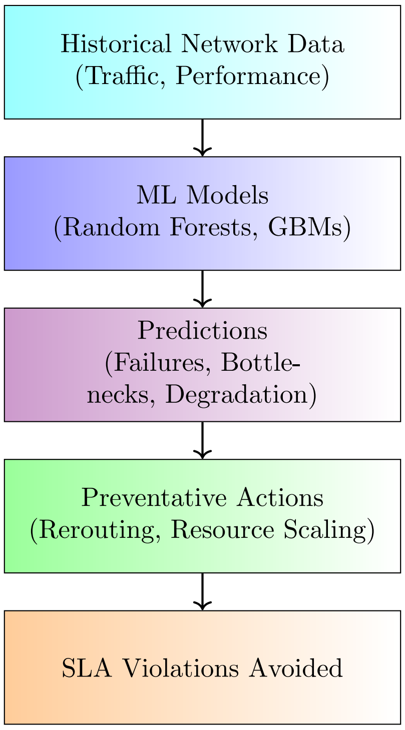

Figure 4: Predictive Analytics and Fault Management Using ML Models in AI-SDN

Let the input features to the model be $L(t)$ for packet loss, $\Delta(t)$ for latency, and $C(t)$ for network congestion at time t. The model outputs a probability $P(F)$ of a failure occurring in the near future. If $P(F)$ exceeds a predefined threshold, the AI-SDN controller can take preemptive actions, such as rerouting traffic away from potentially failing links or allocating additional resources to stabilize the network. For example, if the Random Forest model predicts a significant risk of failure on a link based on increasing congestion and packet loss, the controller might redirect traffic to alternate routes, ensuring continued network availability and minimizing the impact on service quality.

This predictive capability is critical for maintaining high network reliability and adherence to Service Level Agreements (SLAs). By forecasting issues before they materialize, the AI-SDN controller reduces the frequency and severity of network outages, enhances fault tolerance, and optimizes the overall performance of the network infrastructure. This proactive approach to fault management, combined with the controller's ability to dynamically adjust network configurations in real time, ensures that the AI-SDN system remains resilient under changing conditions (Dürr, 2012).

2.3 Decision-Making Layer

The decision-making layer in the AI-SDN architecture is integral to translating the analytical insights generated by the AI processing layer into enforceable network policies. It functions as the logical interface that bridges the predictive outputs of AI-driven models with the operational control mechanisms of the network. Once the AI processing layer provides its output—such as forecasts of impending traffic congestion, classifications of network flows, or optimized routing decisions—the decision-making layer is responsible for converting these insights into specific, executable instructions that can be implemented by SDN-enabled devices in real time.

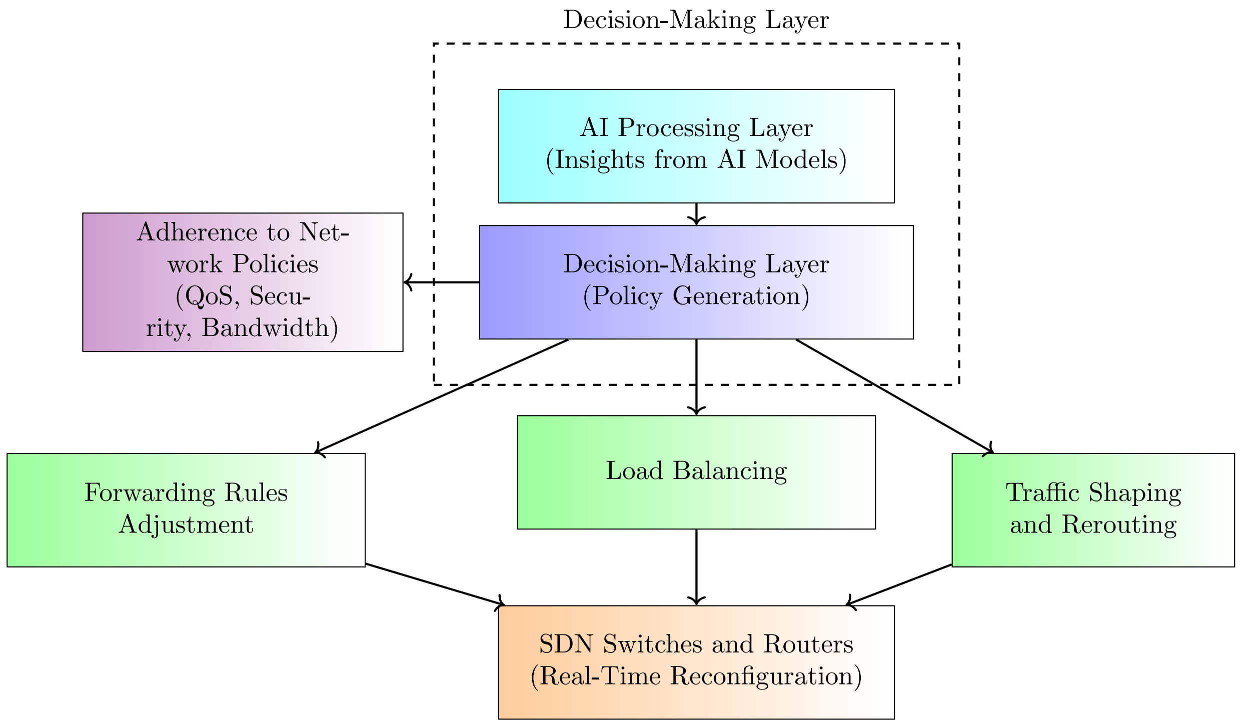
Decision-Making Layer
AI Processing Layer (Insights from AI Models)
Decision-Making Layer (Policy Generation)
Adherence to Network Policies (QoS, Security, Bandwidth)
Forwarding Rules Adjustment
Load Balancing
Traffic Shaping and Rerouting
SDN Switches and Routers (Real-Time Reconfiguration)

Figure 5: Decision-Making Layer in AI-SDN Architecture

To illustrate, consider a scenario in which the AI processing layer predicts the onset of congestion within a particular network segment. The decision-making layer must respond by issuing new routing policies to mitigate this congestion. Let f_i represent a network flow, and let $P(f_i)$ denote the path assigned to f_i through the network. If the predicted congestion is on $P(f_i)$, the decision-making layer will select an alternative path, $P'(f_i)$, which is expected to have lower congestion. This re-routing decision is made in accordance with multiple operational considerations, including Quality of Service (QoS) parameters, security policies, and bandwidth requirements. By dynamically adapting network configurations in response to real-time conditions, the decision-making layer ensures that the network remains efficient, minimizes delays, and maintains the desired performance levels.

The interaction between the decision-making layer and the physical network infrastructure is facilitated through standardized control protocols such as OpenFlow. These protocols enable the decision-making layer to modify flow tables, update forwarding rules, and manage resource allocations within SDN-enabled devices. For instance, when rerouting a flow due to predicted congestion, the decision-making layer directly issues instructions to network switches and routers to modify the forwarding paths in accordance with the new routing policy $P'(f_i)$. OpenFlow provides the flexibility to adjust the network's control plane in real time, enabling swift reconfiguration based on the AI-driven insights.

Furthermore, the decision-making layer must operate under stringent constraints to ensure that any actions taken do not violate overarching network policies related to security, QoS, or service-level agreements (SLAs). For example, in critical use cases, such as networks supporting latency-sensitive applications, the decision-making layer must ensure that rerouting decisions do not adversely affect the latency guarantees required by specific flows. Similarly, bandwidth reservation mechanisms, implemented to maintain service quality, must be considered when reallocating resources to avoid oversubscription or performance degradation in other parts of the network. Thus, the decision-making layer's actions are not solely based on the prediction of congestion but must also account for the broader operational context in which those decisions are made.

The real-time nature of this decision-making process is critical to maintaining the responsiveness and adaptability of the AI-SDN system (Latif et al., 2022). Given the dynamic nature of modern networks—where traffic patterns, resource availability, and user demands can change rapidly—the decision-making layer must process inputs and execute control actions with minimal delay. This responsiveness is essential for maintaining network performance, in scenarios involving large-scale networks with complex topologies and heterogeneous traffic types. By continually adapting to the evolving network state, the decision-making layer contributes to the overall stability and efficiency of the system.

In more advanced implementations, the decision-making layer may also incorporate multi-objective optimization techniques to balance competing demands on the network. For instance, in addition to rerouting traffic to avoid congestion, the system might simultaneously aim to minimize energy consumption by favoring paths that traverse underutilized or more energy-efficient devices. Such multi-criteria decision-making algorithms ensure that the AI-SDN controller not only addresses immediate performance concerns but also optimizes network resources holistically, considering long-term operational efficiency and sustainability.

The effectiveness of the decision-making layer is highly dependent on the quality and timeliness of the data provided by the AI processing layer, as well as the precision with which the network control commands are executed. Any inaccuracies in the prediction of traffic congestion or misalignment between the decision-making logic and the real-world network conditions could lead to suboptimal performance or, in the worst case, network instability. As such, the decision-making layer often incorporates feedback mechanisms that allow it to verify the outcomes of its actions and adjust policies if the desired effects are not achieved. These feedback loops enhance the robustness of the system, allowing it to learn from past decisions and improve its future responses.

2.4 Northbound Interface (NBI)

The northbound interface (NBI) in the AI-SDN architecture functions as an abstraction layer that enables higher-level network management and orchestration platforms to interact with the AI-SDN controller. It provides external applications, such as cloud infrastructure managers or Multi-access Edge Computing (MEC) orchestrators, with access to real-time network statistics and control mechanisms via exposed APIs, such as RESTful APIs or gRPC endpoints. These external systems can issue requests for network insights, push configuration updates, or instruct the AI-SDN controller to make routing or traffic flow decisions based on broader business objectives.

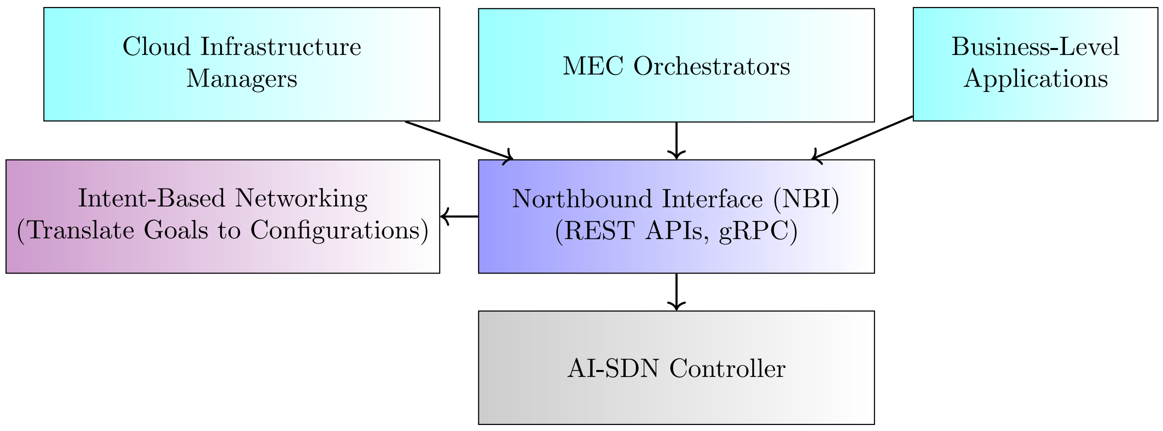
Cloud Infrastructure Managers
MEC Orchestrators
Business-Level Applications
Intent-Based Networking (Translate Goals to Configurations)
Northbound Interface (NBI) (REST APIs, gRPC)
AI-SDN Controller

Figure 6: Northbound Interface (NBI) in AI-SDN Architecture

A key feature of the NBI is its support for intent-based networking (IBN), which abstracts the low-level details of network management, allowing network administrators to specify high-level service goals without needing to define the specific configurations required to achieve those goals. In IBN, the administrator expresses intents, such as latency requirements or throughput objectives, and the AI-SDN controller autonomously translates these intents into specific network actions, such as routing decisions, resource allocations, or prioritization rules (Ibrar et al., 2020).

For example, consider a scenario in which a network administrator specifies a maximum latency requirement $\Delta_{\max}$ for a critical service. The AI-SDN controller, through the NBI, receives this intent and ensures that the latency experienced by the service does not exceed $\Delta_{\max}$. Let f_i represent a specific network flow associated with the critical service, and let $P(f_i)$ denote the path assigned to f_i through the network. The AI-SDN controller continuously monitors the latency along $P(f_i)$ and compares it against $\Delta_{\max}$. If the latency $L(P(f_i))$ along the current path exceeds $\Delta_{\max}$, i.e.,

$$L(P(f_i)) > \Delta_{\max},$$

the controller will dynamically reassign the flow to an alternative path $P'(f_i)$, which is expected to have a lower latency. This reassignment is represented as:

$$P(f_i) \to P'(f_i) \quad \text{where} \quad L(P'(f_i)) < \Delta_{\max}.$$

In this way, the NBI facilitates a closed-loop system where high-level performance goals, such as maintaining latency below $\Delta_{\max}$, are achieved without requiring manual intervention from network operators. The AI-SDN controller autonomously adjusts network configurations in real time, ensuring that the performance objectives are consistently met.

Another example involves optimizing throughput for specific flows. Let $T(f_i)$ represent the throughput of flow f_i along its designated path $P(f_i)$. Suppose the intent specifies a minimum throughput requirement $T_{\min}$ for a particular service. The AI-SDN controller, via the NBI, continuously monitors the throughput for f_i and checks if:

$$T(f_i) < T_{\min}.$$

If the throughput falls below $T_{\min}$, the controller can reallocate resources, such as bandwidth or prioritize f_i over other less-critical flows, to ensure that $T(f_i) \geq T_{\min}$. This dynamic adjustment is performed without manual input and ensures that the network adapts to changing traffic conditions while meeting the specified throughput objectives.

The interaction between the NBI and external systems relies on APIs that facilitate real-time data exchange and control. RESTful APIs, which are based on standard HTTP methods, allow external applications to query network states, request telemetry data, or issue configuration updates in a stateless manner. For performance-sensitive applications, gRPC provides a more efficient mechanism for communication, using a binary protocol that supports lower-latency interactions. Through these APIs, external systems can subscribe to network telemetry streams, such as monitoring link utilization, jitter, or packet loss, and can also push high-level intents to the AI-SDN controller.

For example, a MEC orchestrator might request real-time network statistics for a particular edge node. Let $U(l_i)$ denote the utilization of a network link l_i. The orchestrator, through the NBI, can subscribe to telemetry updates for $U(l_i)$ and receive real-time feedback. Based on this information, the MEC orchestrator might instruct the AI-SDN controller to redirect certain flows to prevent overutilization if:

$$U(l_i) > U_{\max},$$

where $U_{\max}$ is the maximum acceptable utilization threshold for the link. The AI-SDN controller would then reroute flows away from l_i to balance traffic and prevent congestion.

In intent-based networking, these decisions are driven by high-level goals rather than specific configurations. The NBI abstracts the complexity of network management, allowing external systems to focus on strategic objectives, such as ensuring application performance or adhering to SLAs, without dealing with the underlying details of path selection, traffic engineering, or resource allocation (Li et al., 2015).

The NBI's ability to abstract these lower-level details and expose higher-level control via APIs is also crucial for automating network operations. Automation platforms can use the NBI to implement intent-based policies in an orchestrated fashion. For example, in a cloud environment, the NBI can be used by an orchestration platform to dynamically adjust network configurations based on fluctuating demand, such as increasing bandwidth for virtual machines experiencing high traffic or deploying new network functions to handle increased load.

In conclusion, the northbound interface (NBI) provides an abstraction layer that connects the AI-SDN controller to higher-level network management and orchestration systems, facilitating interaction through APIs such as RESTful and gRPC endpoints. A key capability of the NBI is its support for intent-based networking, allowing administrators to define high-level objectives, such as latency or throughput requirements, which are autonomously translated by the AI-SDN controller into specific actions, such as dynamic flow rerouting or resource reallocation (Mireslami et al., 2017). By continuously monitoring network conditions and acting on high-level intents, the NBI enables real-time optimization and ensures that the network adapts to meet performance goals efficiently. The mathematical expressions underlying this process, such as latency constraints and throughput guarantees, are critical to ensuring that the AI-SDN controller's actions align with business-level objectives while maintaining network stability and performance.

2.5 Southbound Interface (SBI)

The southbound interface (SBI) plays a critical role in the AI-SDN architecture as the communication channel between the AI-SDN controller and the underlying network infrastructure, which includes switches, routers, and other forwarding devices. The SBI is responsible for translating the decisions made by the AI-SDN controller into concrete actions that can be executed by the physical network components. This interface utilizes various standardized protocols, such as OpenFlow, NETCONF, and BGP-LS, to facilitate the dynamic configuration and management of the network devices. These protocols enable the controller to modify flow tables, adjust routing paths, and configure Quality of Service (QoS) settings in real time, responding to insights derived from the AI processing layer.

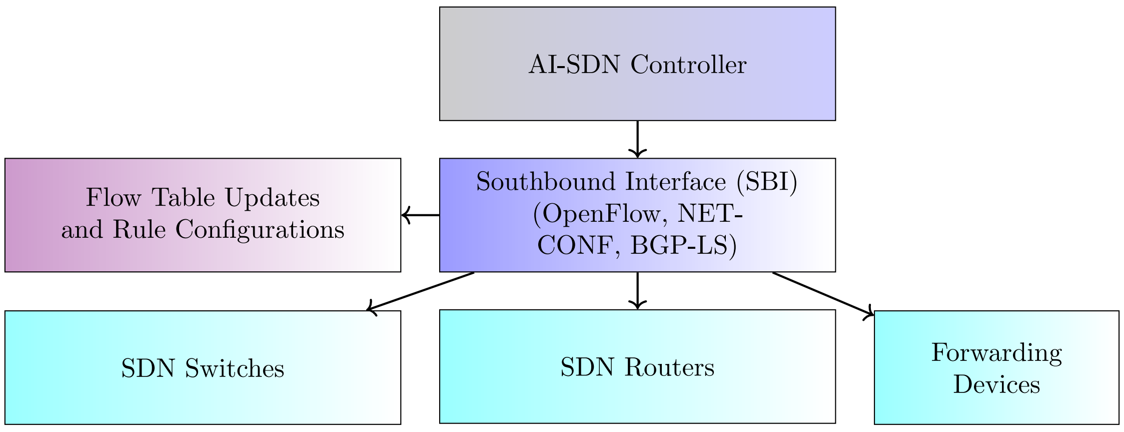
AI-SDN Controller
Southbound Interface (SBI) (OpenFlow, NETCONF, BGP-LS)
Flow Table Updates and Rule Configurations
SDN Switches
SDN Routers
Forwarding Devices

Figure 7: Southbound Interface (SBI) in AI-SDN Architecture

OpenFlow is one of the most widely used protocols in this context, providing a flexible mechanism for controlling how packets are routed through the network. It allows the AI-SDN controller to directly manipulate the flow tables of switches, defining how specific flows of packets should be handled. NETCONF, on the other hand, is a network management protocol that facilitates the configuration of network devices through remote procedure calls (RPCs), enabling the controller to manage the device configurations in a standardized manner. BGP-LS (Border Gateway Protocol - Link State) extends BGP to carry detailed link-state information, which is useful for AI-SDN controllers in obtaining an accurate view of the network topology and making informed routing decisions.

A primary function of the SBI is to ensure that the network infrastructure can dynamically adapt to the real-time decisions made by the AI-SDN controller. For example, if the AI processing layer identifies that a specific network link is approaching its capacity limit, the controller can use the SBI to offload traffic to alternative, less congested links. Mathematically, this involves selecting a new path $P'(p_i)$ for a set of packets p_i, which minimizes the expected delay $E[\Delta]$ while maintaining network performance. If the current path $P(p_i)$ is associated with a higher delay due to congestion, the decision-making process would involve reassigning the flow from $P(p_i)$ to $P'(p_i)$, where:

$$P(p_i) \to P'(p_i) \quad \text{such that} \quad E[\Delta(P'(p_i))] < E[\Delta(P(p_i))].$$

This optimization process ensures that the flow p_i follows a path with reduced delay, thus improving overall network efficiency and avoiding congestion. The SBI translates this decision into specific instructions sent to the network switches, where the flow tables are updated to reroute the packets in real time.

In practice, the AI-SDN controller may need to make these adjustments for multiple flows simultaneously, which requires efficient coordination and rapid communication between the controller and the network devices. The SBI ensures that these modifications are carried out promptly, allowing the network to adapt to changing conditions without the need for manual intervention. The ability to modify flow tables on the fly, adjust routing policies, and update QoS configurations enables the AI-SDN system to maintain optimal network performance, even in highly dynamic environments with fluctuating traffic patterns (Holik and Dolezel, 2020).

For instance, in a situation where the AI-SDN controller predicts an increase in traffic for a particular service, it may decide to prioritize certain flows to ensure that critical applications meet their QoS requirements. The controller, through the SBI, can instruct switches to allocate more bandwidth to these high-priority flows by adjusting the QoS settings. Let b_i represent the bandwidth allocated to flow f_i. If the AI-SDN controller determines that f_i requires additional bandwidth to meet its performance objectives, it can modify b_i to ensure sufficient resources are allocated:

$$b_i \to b_i' \quad \text{where} \quad b_i' > b_i.$$

This dynamic bandwidth reallocation ensures that high-priority traffic receives the necessary resources to meet its performance goals, such as low latency or high throughput, while less critical traffic is deprioritized if needed.

The SBI's role is not limited to modifying flow tables or bandwidth allocations; it also supports the reconfiguration of routing policies based on AI-driven insights. In particular, if the AI processing layer detects suboptimal routing, the controller can leverage the SBI to adjust the paths that packets take through the network. This is often necessary in scenarios where changes in traffic demand or network conditions require the rerouting of traffic to maintain service quality. Let $R(f_i)$ represent the routing policy for flow f_i. If the controller determines that the current routing policy is leading to inefficiencies, such as higher delays or packet loss, it can modify the routing policy to an optimized version $R'(f_i)$, ensuring that the flow takes a more efficient path through the network:

$$R(f_i) \to R'(f_i) \quad \text{where} \quad \text{performance metrics (e.g., delay, loss) are improved under } R'(f_i).$$

This capability allows the AI-SDN system to continuously optimize routing decisions based on real-time data, ensuring that the network adapts quickly to both predicted and unforeseen changes in traffic patterns.

Table 2: AI Model Applications in SDN

AI Model	**Application**	**Key Features**
CNN	Traffic Classification	Identifies spatial patterns in traffic data
LSTM	Traffic Prediction	Analyzes temporal data for future traffic load prediction
DQN	Resource Optimization	Optimizes routing and resource allocation via RL
Random Forest	Fault Prediction	Predicts network failures based on historical data

Table 3: Key Interfaces in AI-SDN Controller

Interface	**Functionality**
Northbound Interface (NBI)	Connects with higher-level orchestration platforms (e.g., MEC)
Southbound Interface (SBI)	Facilitates communication with network devices (e.g., routers, switches)

3 Performance Metrics and Evaluation

The performance of an AI-enhanced SDN system can be evaluated using several KPIs, which can be expressed mathematically as follows:

Latency refers to the time it takes for a packet to travel from the source to the destination. The average latency L can be expressed as:

$$L = \frac{1}{N} \sum_{i=1}^{N} \left(t_i^{\text{arrival}} - t_i^{\text{departure}} \right)$$

Where:

- N is the number of packets,
- t_i^{arrival} is the arrival time of packet i,
- $t_i^{\text{departure}}$ is the departure time of packet i.

Throughput measures the total amount of data transmitted across the network over a given time period. The throughput T is given by:

$$T = \frac{D_{\text{total}}}{t_{\text{end}} - t_{\text{start}}}$$

Where:

- D_{total} is the total amount of data transmitted (in bits or bytes),
- t_{start} and t_{end} are the start and end times of the measurement period.

Resource utilization refers to how efficiently network resources are allocated. The resource utilization RU can be expressed as:

$$RU = \frac{R_{\text{used}}}{R_{\text{total}}}$$

Where:

- R_{used} is the amount of resources currently being used,
- R_{total} is the total available resources.

Service quality encompasses metrics like jitter, packet loss, and responsiveness. Jitter J, which refers to the variation in packet delay, can be defined as:

$$J = \frac{1}{N-1} \sum_{i=1}^{N-1} \left| \left(t_{i+1}^{\text{arrival}} - t_i^{\text{arrival}} \right) \right|$$

Packet loss PL can be expressed as the percentage of lost packets during transmission:

$$PL = \frac{P_{\text{lost}}}{P_{\text{sent}}} \times 100$$

Where:

- P_{lost} is the number of lost packets,
- P_{sent} is the total number of sent packets.

The proposed AI-SDN system can be evaluated through simulation or deployment in a cloud testbed, where the impact of AI-driven decision-making on network performance is measured against traditional SDN approaches.

4 Conclusion

This research explores the integration of Artificial Intelligence (AI) with Software-Defined Networking (SDN) to address dynamic resource allocation challenges in cloud environments. Traditional SDN architectures, while flexible and programmable, are limited by static rule-based systems that struggle to meet the real-time demands of modern cloud services, especially under fluctuating workloads. By incorporating AI techniques such as reinforcement learning, deep learning, and predictive analytics, the study aims to enhance SDN's decision-making capabilities, allowing for more adaptive and autonomous network management. This AI-driven SDN framework focuses on optimizing resource allocation, improving network performance, reducing latency, and meeting the needs of latency-sensitive and high-throughput applications.

The study delves into the architectural and algorithmic design of AI-enhanced SDN systems, evaluating their effectiveness in dynamic cloud infrastructures. AI models integrated into SDN controllers can predict network states, adjust resources in real time, and proactively manage network traffic and congestion. The proposed AI-SDN framework offers a path towards autonomous network management in cloud environments, but the paper also acknowledges the complexities and challenges associated with AI integration, including computational overhead, security vulnerabilities, and model generalization issues. The application of AI to SDN in dynamic cloud environments faces several technical, operational, and practical limitations that challenge its effectiveness, scalability, and real-world deployment. The incorporation of AI techniques such as deep learning (DL), reinforcement learning (RL), and predictive analytics into SDN for dynamic resource allocation introduces significant computational demands. These AI models typically require extensive computational resources for training, as well as non-trivial processing power for inference during real-time decision-making. In large-scale cloud environments, where the network must respond dynamically to fluctuating workloads, the computational overhead associated with AI processing can introduce additional latency. This latency can undermine one of the primary goals of AI-enhanced SDN systems—namely, reducing network latency and improving real-time resource allocation efficiency.

For example, RL, which is often used in these systems to optimize decisions based on trial and error, requires substantial computational time to evaluate the network state, choose actions, and update policies. The iterative nature of RL models, combined with the need to process large amounts of real-time network data, increases the time required for the SDN controller to make informed decisions. In mission-critical applications such as real-time video conferencing, augmented reality (AR), and virtual reality (VR), any delay in decision-making could degrade the quality of service (QoS) significantly, leading to unsatisfactory user experiences.

Furthermore, deep learning models used for traffic prediction and classification demand high-performance hardware, such as Graphics Processing Units (GPUs) or specialized AI accelerators, to perform adequately in real-time scenarios. The need for such advanced hardware can increase the cost of deploying AI-SDN systems and limit their accessibility to organizations without the requisite infrastructure. Additionally, the latency associated with these computations can contradict the initial intent of optimizing resource allocation, in latency-sensitive applications. Therefore, the computational overhead is a significant barrier to the widespread adoption of AI-enhanced SDN systems. AI models, especially those leveraging machine learning (ML) and deep learning, are heavily reliant on large datasets to function effectively. These models must be trained on vast amounts of network traffic data, historical resource utilization patterns, and workload behavior to make accurate predictions or optimizations. However, gathering and labeling this data in dynamic cloud environments can be challenging, in multi-tenant environments where data privacy concerns may prevent access to sufficient training data.

Moreover, the diversity and variability of cloud workloads present challenges for AI model generalization. AI models trained on specific network configurations or traffic patterns may struggle to generalize to new, unseen network conditions. This is especially problematic in cloud environments, which are highly dynamic and often host a wide variety of applications with different performance requirements. For instance, a model trained in one cloud environment may not perform optimally in another, especially if the workload patterns, traffic types, or infrastructure configurations differ significantly.

In addition, changes in the network environment, such as the addition of new hardware, the introduction of novel traffic types, or shifts in user behavior, may necessitate retraining or fine-tuning of the AI models to maintain performance. Retraining these models is not only time-consuming but also requires continuous monitoring of model performance to ensure they adapt appropriately to the changing environment. This places additional operational burdens on network administrators and requires ongoing resource allocation for maintaining AI models, which could reduce the overall efficiency gains initially sought through automation. The integration of AI with SDN introduces new security vulnerabilities, in the form of adversarial attacks. AI models, especially those used for traffic prediction, classification, or resource allocation, are vulnerable to manipulation by adversarial inputs. Adversarial attacks involve subtly altering the input data fed to an AI model in such a way that the model makes incorrect predictions or decisions. In the context of AI-enhanced SDN, adversarial attacks could be used to trick the AI model into misallocating resources, rerouting traffic inefficiently, or creating congestion in critical parts of the network.

For instance, an attacker could generate adversarial traffic patterns that cause the AI model to incorrectly classify traffic as low-priority, leading to poor QoS for important applications. Alternatively, adversarial attacks could be used to exploit weaknesses in the AI model's decision-making process, leading to security breaches, such as unauthorized access to sensitive data or denial-of-service (DoS) attacks.

The complexity of AI models also makes it difficult to secure them effectively. Traditional security measures, such as firewalls or intrusion detection systems, may not be sufficient to detect or prevent adversarial attacks targeting the AI components of the SDN system. Securing AI-enhanced SDN systems requires specialized knowledge of AI security, as well as continuous monitoring for potential adversarial activity, which increases the operational complexity of managing such systems. AI-enhanced SDN systems are inherently more complex than traditional SDN architectures due to the introduction of machine learning models, data pipelines, and autonomous decision-making processes. This increased complexity poses several operational challenges. First, the design, deployment, and maintenance of AI-SDN systems require specialized expertise in both networking and AI, which may not be readily available in all organizations. Network administrators who are accustomed to managing traditional SDN environments may find it difficult to adapt to the new AI-driven paradigm, leading to a steep learning curve and potential misconfigurations.

Moreover, troubleshooting and maintaining AI-enhanced SDN systems can be more difficult than managing traditional SDN systems. The interaction between AI models and network components is not always transparent, and diagnosing performance issues may require in-depth knowledge of both the underlying network infrastructure and the AI algorithms in use. This complexity can also make the system more prone to errors or misconfigurations, if the AI models are not properly fine-tuned or if the network environment changes in ways that the AI models are not equipped to handle.

In addition, the dynamic nature of AI-driven decision-making can lead to unintended consequences if the AI models are not carefully designed and tested. For example, an AI model that optimizes for one performance metric, such as minimizing latency, may inadvertently cause resource contention or degrade performance in other areas, such as throughput or energy efficiency. Balancing these trade-offs requires careful tuning of the AI models, which adds to the operational complexity of managing AI-SDN systems.

Cloud environments are often heterogeneous, consisting of infrastructure from multiple vendors and spanning various platforms and technologies. Ensuring interoperability between AI-SDN systems and existing cloud infrastructure can be challenging, if the AI models are designed to operate within specific network configurations or with proprietary hardware. Many cloud environments rely on a mix of SDN controllers, network devices, and orchestration tools from different vendors, each with its own set of APIs and protocols.

The lack of standardization in AI-enhanced SDN systems further complicates interoperability. While SDN itself has benefited from the development of standardized protocols such as OpenFlow, the integration of AI introduces new challenges related to data collection, model deployment, and decision-making interfaces. Different vendors may implement AI-SDN solutions in ways that are not fully compatible with one another, leading to potential integration issues when deploying AI-enhanced SDN across heterogeneous cloud environments.

In addition, the rapid pace of innovation in both AI and SDN means that standards are constantly evolving, making it difficult for organizations to keep up with the latest developments. This lack of standardization can result in vendor lock-in, where organizations are forced to rely on proprietary AI-SDN solutions that are not easily interoperable with other systems. This, in turn, limits the flexibility and scalability of AI-SDN deployments, in large, distributed cloud environments where interoperability is critical for efficient resource management.

References

Baucke, S., Ali, R. B., Kempf, J., Mishra, R., Ferioli, F., and Carossino, A. (2013). Cloud atlas: A software defined networking abstraction for cloud to wan virtual networking. In *2013 IEEE Sixth International Conference on Cloud Computing*, pages 895–902. IEEE.

Bhat, S. and Kavasseri, A. (2023). Enhancing security for robot-assisted surgery through advanced authentication mechanisms over 5g networks. *European Journal of Engineering and Technology Research*, 8(4):1–4.

Dai, W., Qiu, L., Wu, A., and Qiu, M. (2016). Cloud infrastructure resource allocation for big data applications. *IEEE Transactions on Big Data*, 4(3):313–324.

Dürr, F. (2012). Towards cloud-assisted software-defined networking. *Stuttgart, Germany: Institute of Parallel and Distributed Systems (IPVS)*.

Fard, M. V., Sahafi, A., Rahmani, A. M., and Mashhadi, P. S. (2020). Resource allocation mechanisms in cloud computing: a systematic literature review. *IET Software*, 14(6):638–653.

Holik, F. and Dolezel, P. (2020). Industrial network protection by sdn-based ips with ai. In *Asian Conference on Intelligent Information and Database Systems*, pages 192–203. Springer.

Ibrar, M., Akbar, A., Jan, S. R. U., Jan, M. A., Wang, L., Song, H., and Shah, N. (2020). Artnet: Ai-based resource allocation and task offloading in a reconfigurable internet of vehicular networks. *IEEE Transactions on Network Science and Engineering*, 9(1):67–77.

Jain, R. and Paul, S. (2013). Network virtualization and software defined networking for cloud computing: a survey. *IEEE Communications Magazine*, 51(11):24–31.

Jani, Y. (2022). Optimizing database performance for large-scale enterprise applications. *International Journal of Science and Research (IJSR)*, 11(10):1394–1396.

Latif, S. A., Wen, F. B. X., Iwendi, C., Li-Li, F. W., Mohsin, S. M., Han, Z., and Band, S. S. (2022). Ai-empowered, blockchain and sdn integrated security architecture for iot network of cyber physical systems. *Computer Communications*, 181:274–283.

Li, F., Cao, J., Wang, X., and Sun, Y. (2017a). A qos guaranteed technique for cloud applications based on software defined networking. *IEEE access*, 5:21229–21241.

Li, F., Cao, J., Wang, X., Sun, Y., and Sahni, Y. (2017b). Enabling software defined networking with qos guarantee for cloud applications. In *2017 IEEE 10th International Conference on Cloud Computing (CLOUD)*, pages 130–137. IEEE.

Li, Y., Tang, X., and Cai, W. (2015). Dynamic bin packing for on-demand cloud resource allocation. *IEEE Transactions on Parallel and Distributed Systems*, 27(1):157–170.

Lin, W.-Y., Lin, G.-Y., and Wei, H.-Y. (2010). Dynamic auction mechanism for cloud resource allocation. In *2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, pages 591–592. IEEE.

Liu, N., Li, Z., Xu, J., Xu, Z., Lin, S., Qiu, Q., Tang, J., and Wang, Y. (2017). A hierarchical framework of cloud resource allocation and power management using deep reinforcement learning. In *2017 IEEE 37th international conference on distributed computing systems (ICDCS)*, pages 372–382. IEEE.

Luo, M.-Y. and Chen, J.-Y. (2013). Software defined networking across distributed datacenters over cloud. In *2013 IEEE 5th International Conference on Cloud Computing Technology and Science*, volume 1, pages 615–622. IEEE.

Mireslami, S., Rakai, L., Far, B. H., and Wang, M. (2017). Simultaneous cost and qos optimization for cloud resource allocation. *IEEE Transactions on Network and Service Management*, 14(3):676–689.

Mireslami, S., Rakai, L., Wang, M., and Far, B. H. (2019). Dynamic cloud resource allocation considering demand uncertainty. *IEEE Transactions on Cloud Computing*, 9(3):981–994.

Paul, S., Jain, R., Samaka, M., and Pan, J. (2014). Application delivery in multi-cloud environments using software defined networking. *Computer Networks*, 68:166–186.

AFFILIATION OF KAUSHIK SATHUPADI STAFF ENGINEER, GOOGLE LLC, SUNNYVALE, CA : Sunnyvale, CA
sathupadi.kaushik@gmail.com