AMERICAN JOURNAL OF MANAGEMENT AND IOT MEDICAL COMPUTING
Peer Reviewed, Referred & Indexed Journal
E-ISSN: 3069-0110
Vol.5, No.1(2026)
www.ajmimc.com

# INTELLIGENT TRAFFIC MANAGEMENT IN SOFTWARE-DEFINED NETWORKS THROUGH DATA-DRIVEN MULTI-AGENT REINFORCEMENT LEARNING

**PERUGU GANGAIAH[1], MARU MANDHIRA[2], MALI AISHWATHA[3], KATLA VIDYA SAGAR[4], KELOTHU JAYA KALYAN[5], KARRESHIVAPRASAD[6]**

**ASSISTANT PROFESSOR[1], UG SCHOLAR[2,3,4,5&6]**

**DEPARTMENT OF CSE, NARSIMHA REDDY ENGINEERING COLLEGE (UGC- AUTONOMOUS) MAISAMMAGUDA (V), KOMPALLY, SECUNDERABAD, TELANGANA-500100**

**ABSTRACT**

Software-Defined Networks (SDNs) have revolutionized network management by separating the control and data planes, enabling centralized traffic control. In India, rapid growth in internet users over 1.2 billion mobile subscriptions and rising data traffic has led to increasing congestion and packet loss in networks. Traditional approaches struggle with dynamic traffic patterns. The objective is to enhance congestion detection and traffic control in SDNs using machine learning. It enables intelligent, adaptive network management, reducing delays and improving throughput dynamically. Manual network traffic control relies on human monitoring and static routing rules. Network administrators adjust traffic paths and priorities based on observed congestion patterns and historical usage data. Tools like SNMP and simple logging systems provide basic insights but require constant human intervention to optimize performance. Manual systems are slow, error-prone, and inflexible. They cannot respond to real-time traffic fluctuations, often resulting in packet loss, increased latency, and inefficient bandwidth utilization. Scalability is limited, making them unsuitable for large, high-speed SDN environments. The motivation is to overcome manual limitations by leveraging intelligent, adaptive methods. Machine learning models like SVM, KNN, and TaoTree address inefficiency and latency issues, providing real-time congestion detection and traffic prediction. The system improves reliability, accuracy, and scalability, supporting dynamic network management for high-traffic SDN environments. The proposed system uses data-driven machine learning models for adaptive congestion detection and traffic control in SDNs. SVM handles high-dimensional traffic data, KNN predicts traffic patterns using neighbor instances, and TaoTree with SMOTE balances multiclass datasets for accurate classification. These models analyze real-time network traffic, detect congestion proactively, and predict throughput and delay. By integrating predictions into SDN controllers, the system dynamically adjusts routing and bandwidth allocation, improving network efficiency, reducing packet loss, and enabling self-optimizing traffic management.

**1.INTRODUCTION** In modern communication networks, data plays a pivotal role, not only because of the transmitted data but also due to its utilization to optimize network performance. From traditional architectures to next-generation programmable infrastructures such as Software-Defined Networks (SDNs), data has consistently guided critical decisions in routing, load balancing, and traffic engineering. Network administrators have relied on data extracted from traffic to monitor it, identify its patterns, and determine the most appropriate routing strategies to ensure efficient flow transmission The rise of Artificial Intelligence has enabled a shift from static protocol configurations toward intelligent, data-driven network control. Across domains such as wireless networks, vehicular systems, and SDNs, data-centric methods have demonstrated clear benefits. For instance, Fortuna et al. [1] proposed a multi-level modeling strategy using real-world wireless datasets to optimize performance and manage resources. Similarly, Jiang et al. [2] introduced Data-Driven Networking (DDN), a paradigm that allows protocols to be dynamically adjusted in response to quality metrics collected in multiple sessions. Their approach emphasizes automatic control, using adaptive settings rather than a fixed one. In parallel, the adoption of machine learning in domains such as network intrusion detection has highlighted the strength of data-driven frameworks. Jiang et al. [2] proposed a hierarchical classification strategy using both supervised learning and unsupervised autoencoders to detect and classify cyber attacks, emphasizing the importance of feature representation and balanced training data for effective intrusion detection in IP-based networks. In the context of SDNs, the centralized control plane and the global view of the network offer an opportunity to integrate intelligent frameworks that can learn from the real-time state of the network. In addition, decoupling control from data planes in SDNs enable more flexible, programmable, and adaptive systems. However, the efficient detection and control of network congestion represents one of the persistent challenges in SDNs due to the high volume of traffic coming from large and dense networks such as data centers, and the complexity of the network topology that leads to reduced network performance in terms of packet loss, slow data transfer speeds, and reduced throughput. Kandula et al. [3] highlight that a server cluster with around 1500 nodes may process close to 100,000 network flows every second. In a similar context, Kandula et al. [4] report that, under peak load conditions, a network with 100 switches can experience up to 10 million flow requests per second.

Additionally, even a modest delay of 10 milliseconds in flow setup, introduced by the SDN controller, can add approximately 10% latency to the large number of short-lived flows typically present in such environments. Yet, most existing congestion detection and control approaches rely on static thresholds or heuristics, which lack adaptability in dynamic environments. To address the limitations of static and monolithic congestion control approaches, we propose a data-driven Multi-Agent Reinforcement Learning (MARL) framework designed for congestion classification and adaptive decision-making in Software-Defined Networks (SDNs). The proposed framework leverages real-time network metrics—such as packet loss, delay, jitter, and throughput—collected from live traffic to dynamically assess network state and inform control actions. Guided by these observations and feedback from a tailored reward function, the system continuously learns to adjust bandwidth allocation and traffic routing strategies. The architecture comprises two specialized and collaborative agents: (1) a Congestion Classification Agent responsible for evaluating network conditions and classifying the current state into predefined congestion levels, and (2) a Decision-Making Agent, which utilizes Deep Q-Learning (DQN or its variants) to select the optimal actions, including increasing, decreasing, or maintaining bandwidth, and rerouting or preserving current traffic paths. In contrast to previous work employing a single-agent model to jointly handle perception and control, our approach introduces a decoupled agent structure. This modular design improves interpretability, allows for targeted training of each agent, and enhances learning efficiency and robustness. The agents are initially trained offline using both synthetic and real traffic traces, including the MAWI dataset, to simulate heterogeneous environments. After the offline training phase, the trained agents are deployed for online testing in a simulated SDN environment built with Mininet and the Ryu controller. During online operation, the agents act on real-time traffic conditions without further training, enabling real-world evaluation of their generalization and adaptability.

### 1.1 Background, History, and Overview

Software-Defined Networks (SDNs) have transformed traditional networking by separating the control plane from the data plane, allowing centralized network management and flexible traffic routing. In India, internet usage has grown exponentially, with over 1.2 billion mobile subscriptions and an average daily data consumption exceeding 20 GB per user in metropolitan areas. This surge in traffic has led to frequent congestion, packet delays, and inefficient bandwidth utilization. Traditional network management techniques struggle to handle dynamic traffic patterns effectively. Data-driven and intelligent approaches, particularly machine learning, are increasingly applied to detect congestion, predict traffic trends, and dynamically optimize routing, ensuring high network performance. SDNs integrated with machine learning enable real-time monitoring, predictive congestion management, and adaptive traffic control, making them highly relevant in modern high-traffic networks.

### 1.2 Problem Definition

Traditional SDN traffic management relies on static routing rules and human monitoring, which cannot respond to real-time congestion. Networks experience frequent packet loss, high latency, and uneven bandwidth utilization. Manual adjustment of routes is slow and often inaccurate. Tools like SNMP and log analysis provide limited insights, insufficient for large-scale, high-speed networks. Dynamic traffic patterns in modern SDNs require predictive and adaptive mechanisms beyond manual intervention.

### 1.3 Research Motivation

The motivation is to develop a system that overcomes the inefficiencies of manual SDN management. Machine learning models can analyze real-time traffic data, predict congestion, and optimize routing proactively. By leveraging SVM, KNN, and TaoTree models, the system improves traffic prediction accuracy, reduces packet loss, and enhances bandwidth allocation. Intelligent network control ensures scalability and reliability for high-traffic environments. Adaptive, data-driven approaches are essential to maintain network performance under growing demand in India’s digital ecosystem.

### 1.4 Objective

The objective of this project is to implement adaptive congestion detection and traffic control in SDNs using machine learning. It aims to provide real-time network monitoring, predictive congestion analysis, and dynamic routing adjustments to improve throughput, minimize delays, and optimize bandwidth utilization.

### 1.5 Applications

1. Real-time traffic management in Internet Service Provider networks.
2. Optimizing bandwidth allocation in cloud computing and data centers.
3. Congestion detection and resolution in smart city IoT networks.
4. Adaptive routing for 5G and high-speed mobile networks.
5. Network reliability enhancement in enterprise SDN environments.

6. Predictive traffic control for streaming services and video conferencing.
7. Automated network optimization in industrial IoT and smart factories.
8. Reducing packet loss and improving Quality of Service (QoS) in large-scale deployments.

**1.6 Significance**

This project is significant because it transforms SDN traffic management from reactive, manual intervention to proactive, intelligent control. Machine learning models provide high accuracy in congestion detection and traffic prediction, enabling dynamic routing and efficient bandwidth utilization. It reduces network delays, minimizes packet loss, and enhances user experience in high-traffic networks. The system ensures scalability and reliability, supporting India's growing digital infrastructure and high data demand. Overall, it establishes a self-optimizing, data-driven framework for modern SDNs.

**2.LITERATURE SURVEY**

- In the context of congestion management in Software-Defined Networks (SDNs), several studies have focused on traffic classification and traffic pattern identification as a vital step for intelligent routing, load balancing, and stable network performance. Due to the complexity and dynamics of network environments, traditional approaches often face limitations in detecting and mitigating congestion effectively. Consequently, this has led to increasing interest in the use of machine learning and data-driven techniques to analyze real-time traffic patterns, predict congestion, and make control decisions in advance. Using historical and real-time network data collected from network observations, these approaches aim to enhance adaptability and real-time decision-making by enabling the SDN controller to learn from the state of the network and optimize performance efficiently.
- Poupart et al. [17] utilize data mining techniques to predict the size of network flows at the early stages of transmission, which is critical to maintain network performance while traffic conditions change. Their method models the flow size as a statistical distribution rather than relying on a deterministic function that applied fixed conditions. By computing the mean or mode of this distribution based on initial packet observations, the approach provides a probabilistic estimate of whether a flow will become significant. This early prediction enables the controller to differentiate between short-lived and potentially high-impact flows and allocate resources accordingly.
- While Poupart et al. [17] rely on statistical modeling for flow size prediction, Bishop et al. [18] use machine learning algorithms for traffic classification and flow size estimation. They evaluate the effectiveness of a range of machine learning algorithms by investigating the comparative accuracy of neural networks, Gaussian Process Regression (GPR) by M. Seeger [19], and Online Bayesian Moment Matching (oBMM) by F. Omar [20]. GPR provides a probabilistic model that captures uncertainty in flow prediction, while oBMM enables the online updating of flow classification based on sequential data without retraining from scratch. The results demonstrate that learning-based models can significantly outperform traditional rule-based systems, especially in dynamic network environments where traffic characteristics change frequently.
- Shafiq et al. [21] investigate application-level traffic classification using machine learning techniques to improve the identification of network application types based on flow-level features. Their study conducts a comparative analysis of multiple supervised learning algorithms, including Support Vector Machines (SVMs), BayesNet, and Naive Bayes, to evaluate their effectiveness in classifying traffic from common application protocols such as WWW, DNS, FTP, P2P, and Telnet. The classification is performed using flow features extracted from captured network flows, such as packet size, inter-arrival time, flow duration, and total bytes transferred, using the NetMate tool. This enables the models to identify the traffic pattern and make volumetric predictions. The results show that machine learning classifiers, when trained on pertinent features, can provide high accuracy and adaptability, making them suitable for real-time traffic classification in modern networks. Their work highlights the practicality of feature-based ML approaches for scalable and efficient network traffic analysis.
- For traffic prediction, Lv et al. [22] propose a deep learning framework based on Sparse Autoencoders (SAEs) to learn high-level traffic representations. Their architecture employs stacked autoencoders to extract spatial and temporal correlations embedded in network traffic patterns. The model is trained in two phases: first, using greedy layer-wise unsupervised pretraining to initialize weights effectively, and then applying supervised fine-tuning to optimize prediction

performance. By using sparse representations, the model is able to focus on the most prominent traffic features, thereby improving generalization and robustness. Experimental results demonstrate that their approach achieves higher accuracy in predicting traffic volume compared to traditional models, making it suitable for dynamic and complex network environments.

- J. Mao et al. [23] propose a flow control mechanism based on the Deep Deterministic Policy Gradient (DDPG), by T. P. Lillicrap et al. [24], a reinforcement learning algorithm designed for continuous action spaces. In this reinforcement learning framework, the agent observes network parameters such as bandwidth availability, latency, and traffic load, and learns to adjust the data sending rate in real-time. This allows for fine-grained congestion control, enabling the system to respond to minor variations in network dynamics with high precision. By leveraging the ability of DDPG to operate in continuous domains, the proposed approach achieves a more adaptive, continuous and refined control strategy compared to discrete action RL models.
- Other studies have addressed congestion from a control plane perspective. Yu et al. [25] propose DIFANE, a scalable flow management solution that minimizes switch-controller interactions by delegating most forwarding decisions to authority switches. By keeping decision logic within the data plane, DIFANE significantly reduces the communication bottleneck on the controller, thereby alleviating potential congestion in the control plane. Similarly, Curtis et al. [26] introduce DevoFlow, an architectural enhancement that further limits control traffic by transferring greater responsibility to switches. DevoFlow aims to reduce controller overhead by handling frequent or short-lived flows locally, allowing the controller to focus on high-level policies and exceptional events.
- R. Jin et al. [27] explore congestion control within SDN-based data center environments using reinforcement learning algorithms such as SARSA and Q-learning. Although their approach shows promising results, it is primarily limited to single-task scenarios and cannot be generalize across diverse network conditions. Lei et al. [28] introduce MTDQ-ER, (Multi-Task Deep Q-Learning with Experience Replay), a framework that utilizes the programmability of SDN to implement a multi-task RL agent capable of learning and applying optimal congestion control policies across multiple traffic scenarios.
- As refined Q-learning, Zhang et al. [29] present a method that dynamically adjusts transmission rates based on real-time bandwidth utilization. However, their approach employs a single-agent model with fixed-rate control logic, which lacks the adaptability needed in complex and rapidly evolving networks. This rigidity may lead to inefficiencies and even control plane congestion, as the system does not continuously monitor or respond to minor network change.
- Beyond flow detection, machine learning techniques have also been explored for intelligent traffic routing in communication networks. These routing strategies generally fall into two categories: decentralized and centralized approaches. Kato et al. [30] propose a decentralized deep reinforcement learning framework in which individual routers are modeled as neural network agents capable of learning and adapting their forwarding behavior. The system is initially trained using data collected from Open Shortest Path First (OSPF) routing, and then fine-tuned through continuous feedback from the network environment. This architecture enables each router to make autonomous, data-driven routing decisions based on real-time traffic conditions, enhancing flexibility and scalability in heterogeneous network infrastructures.
- In contrast, Rusek et al. [31] introduce RouteNet, a centralized routing framework built on Graph Neural Networks (GNNs). Unlike decentralized models, RouteNet operates at the controller level, leveraging the global view of the network provided by SDNs. By modeling the network as a graph—where nodes represent routers or switches and edges represent links—RouteNet learns to predict performance metrics such as average end-to-end delay. These predictions enable the controller to make informed, topology-aware routing decisions that adapt to changing traffic patterns, thereby improving the overall network efficiency and service quality.
- In our previous research, we explored machine learning-based techniques for elephant flow detection in SDN environments. In [32], we conducted a comparative performance evaluation of several supervised learning algorithms, including Decision Trees, Random Forests, and Support Vector Machines, to identify high-bandwidth flows contributing to network congestion. The study revealed that Random Forests provided superior detection accuracy with

minimal false positives, underscoring the viability of traditional supervised learning for proactive traffic identification. In a later study, in [33], we proposed a federated learning approach to elephant flow detection in distributed SDNs. This work emphasized bias mitigation and privacy preservation by enabling multiple domains to collaboratively train a global model without sharing raw data, effectively addressing class imbalance and data heterogeneity across controllers. While both studies achieved strong results in detection, they were focused exclusively on flow identification and did not incorporate adaptive decision-making or congestion control mechanisms.

## 3.SYSTEM ANALYSIS

**EXISTING SYSTEM** Traditional traffic management in **Software-Defined Networks (SDN)** mainly relies on **rule-based algorithms, static routing mechanisms, and conventional congestion control techniques**. In these systems, the SDN controller monitors network traffic and makes routing decisions based on predefined policies and network statistics such as bandwidth utilization, packet loss, and delay.

Most existing approaches use **heuristic algorithms or single-agent machine learning methods** to detect congestion and manage network traffic. Techniques like shortest path routing, load balancing, and threshold-based congestion detection are commonly used to control traffic flow across network switches.

However, these methods have several limitations. They often struggle to handle **dynamic and large-scale network environments**, where traffic patterns frequently change. Static or rule-based strategies cannot adapt quickly to real-time congestion conditions, leading to **network bottlenecks, increased latency, and packet loss**. Additionally, single-agent learning models may have limited scalability and cannot efficiently coordinate multiple network nodes in complex SDN infrastructures.

**PROPOSED SYSTEM**

The proposed system introduces an **Intelligent Traffic Management framework for Software-Defined Networks using Data-Driven Multi-Agent Reinforcement Learning (MARL)**. This approach enables the network to **adapt dynamically to changing traffic conditions and optimize routing decisions in real time**.

In this system, multiple reinforcement learning agents are deployed across different network components such as **SDN controllers and switches**. Each agent observes network states including traffic load, queue length, bandwidth usage, and packet delay. Based on these observations, the agents learn optimal traffic control strategies through continuous interaction with the network environment.

The **multi-agent reinforcement learning model** allows agents to cooperate and share information, enabling coordinated decision-making across the network. Using reward-based learning, the agents aim to **minimize congestion, reduce packet loss, and optimize overall network throughput**.

The system also uses **data-driven learning techniques** to analyze historical network traffic patterns and improve prediction accuracy. The SDN controller aggregates the decisions from multiple agents and dynamically updates routing policies to balance network traffic effectively.

By combining **SDN programmability with multi-agent reinforcement learning**, the proposed system provides an adaptive and scalable solution for intelligent traffic management in modern high-speed networks.

**Advantages of the Proposed System**

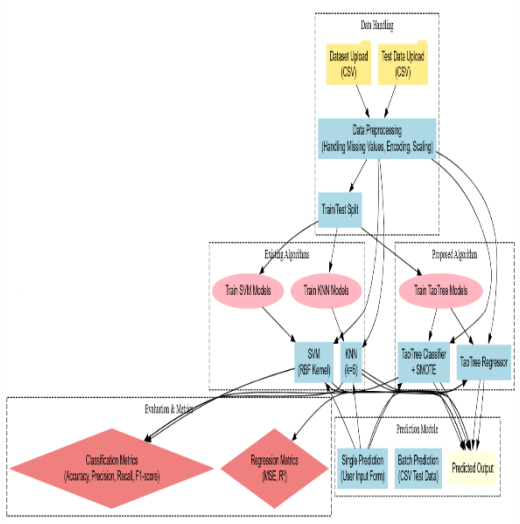
- Adaptive traffic control based on real-time network conditions
- Improved network throughput and resource utilization
- Reduced congestion, latency, and packet loss
- Scalable solution for large and complex SDN environments
- Intelligent coordination between multiple network nodes
- Better performance compared to traditional rule-based approaches

## 4.IMPLEMENTATION

### 4.1 Overview

Step 1: Dataset Collection

The first step in this research involves acquiring a suitable dataset that captures the dynamics of the network environment under study. For the purpose of adaptive congestion detection and traffic control in Software-Defined Networks (SDNs), the dataset must contain features such as packet arrival rates, queue lengths, throughput, link utilization, and other relevant network traffic metrics. These features serve as the input variables for machine learning algorithms. The dataset may be obtained either from real network traffic logs or generated synthetically using network simulation tools to emulate different congestion scenarios. Ensuring the dataset contains sufficient variability and representative examples of congestion events is critical for developing robust predictive models.




Step 2: Dataset Preprocessing

Once the dataset is collected, preprocessing is performed to prepare it for model training. This step addresses data quality issues such as missing or null values, which can significantly affect the performance of machine learning models. Null values are identified and either removed or imputed using suitable statistical techniques depending on the distribution and importance of the feature. Categorical features, such as node identifiers or traffic types, are encoded numerically using label encoding or one-hot encoding to make them compatible with machine learning algorithms. Additionally, feature scaling is applied when necessary to normalize the range of continuous variables, ensuring that algorithms sensitive to feature magnitude (e.g., SVM) perform optimally. This preprocessing step ensures that the data is clean, consistent, and ready for subsequent modeling.

Step 3: Existing Model Development

In this step, the performance of traditional or baseline machine learning algorithms is evaluated on the preprocessed dataset. Specifically, models such as Support Vector Machines (SVM) and K-Nearest Neighbors (KNN) are trained to detect network congestion. SVM, known for its robustness in high-dimensional spaces, is employed with a Radial Basis Function (RBF) kernel to capture complex nonlinear relationships in network traffic patterns. KNN, an instance-based learning algorithm with k=5 neighbors, is used to classify network states based on the similarity of incoming feature vectors to previously observed instances. Training these models on historical data provides baseline performance metrics, which serve as a reference point for assessing the effectiveness of the proposed methods.

Step 4: Proposed Model Development

To address limitations observed in existing models and improve detection and prediction accuracy, the research introduces a novel machine learning approach that combines TAO Tree classifiers with Synthetic Minority Oversampling Technique (SMOTE). The TAO Tree is a fast decision tree algorithm optimized for multiclass classification problems, making it suitable for detecting various congestion levels in SDN traffic. SMOTE is applied to balance the dataset, particularly when certain congestion classes are underrepresented, preventing bias in the model towards majority classes. The combination of TAO Tree and SMOTE ensures the model can accurately learn from all classes, enhancing its predictive capability in real-world network scenarios. Model training involves fine-tuning hyperparameters, validating using cross-validation techniques, and assessing performance against standard metrics.

Step 5: Performance Evaluation

After training, all models—both existing and proposed—are rigorously evaluated to measure their effectiveness in congestion detection and traffic control. Performance metrics include classification accuracy, precision, recall, and F1-score for multiclass classification tasks. For a more comprehensive evaluation, confusion matrices are generated to visualize the model's ability to distinguish between different congestion levels. Statistical comparisons between baseline models (SVM and KNN) and the proposed TAO Tree + SMOTE approach highlight the improvements achieved through the new methodology. This evaluation not only quantifies the predictive power of the models but also demonstrates their applicability to real-time SDN traffic management.

Step 6: Prediction on Unseen Test Data

The final step involves applying the trained models to new, unseen test data to assess their generalization capability. This step simulates real-world deployment, where network states must be predicted based on live traffic inputs that the models have not previously encountered. Single-instance predictions allow administrators to take proactive measures, while batch predictions enable comprehensive monitoring of network performance over time. Successful prediction on unseen data confirms the robustness and reliability of the proposed methodology, providing confidence that the model can be integrated into SDN controllers for adaptive congestion detection and automated traffic management.

**4.2 Data Preprocessing in this Research**

In this research, data preprocessing plays a critical role in preparing the raw dataset for machine learning modeling. The first step involves handling missing or null values, which can arise due to network logging errors or incomplete traffic records.

Missing values are carefully analyzed, and appropriate strategies, such as imputation with mean/median for continuous features or mode for categorical features, are applied to ensure data completeness without introducing bias. Following this, categorical variables are transformed into numeric representations through label encoding or one-hot encoding, allowing algorithms like SVM and KNN to process them efficiently.

Feature engineering is performed to create additional informative attributes that can improve model performance. For example, features such as average packet arrival rate, queue utilization ratio, and flow count per switch are derived from raw network statistics. This step helps the models capture the underlying patterns in network congestion more effectively. Additionally, text-based preprocessing is applied if the dataset contains categorical descriptions of traffic types, network events, or labels. This may involve tokenization, lowercasing, removal of stopwords, and vectorization to convert textual information into a numerical format compatible with machine learning models.

Finally, feature scaling is applied to normalize numerical values, ensuring that all features contribute equally during model training. This is particularly important for distance-based algorithms like KNN and margin-based algorithms like SVM, where feature magnitude directly affects performance. The preprocessing step ensures the dataset is clean, balanced, and fully prepared for subsequent exploratory data analysis, model training, and evaluation.

**4.3 Exploratory Data Analysis (EDA)**

Exploratory Data Analysis (EDA) is conducted to understand the distribution, relationships, and patterns in the dataset prior to model training. EDA involves generating descriptive statistics, visualizations such as histograms, boxplots, scatterplots, and correlation heatmaps. These visualizations help identify feature importance, outliers, and trends that may influence model performance. For instance, a high correlation between queue length and packet drop rate may indicate potential congestion events that the model should focus on detecting.

As part of the data preparation for EDA, the dataset may be divided into subsets for visualization and analysis. Additionally, techniques such as SMOTE (Synthetic Minority Oversampling Technique) are applied to address class imbalance. SMOTE generates synthetic samples for minority classes, ensuring that models do not become biased towards majority classes and that patterns in underrepresented network states are captured. Feature extraction is also considered at this stage to reduce dimensionality while retaining relevant information, allowing more efficient modeling.

EDA also guides the feature selection process by revealing redundant or irrelevant features that can be removed without sacrificing predictive power. This step improves both model efficiency and interpretability. Insights from EDA inform the configuration of training and testing data, influencing how models are validated and fine-tuned. Overall, EDA serves as both a diagnostic and preparatory stage, bridging raw data and machine learning algorithms.

**4.4 Train-Test Split**

After preprocessing and EDA, the dataset is divided into training and testing subsets to evaluate model performance in an unbiased manner. The training set is used to fit the machine learning model, allowing it to learn the underlying patterns and relationships among the features. The test set, which is kept separate, is used exclusively to assess how well the trained model generalizes to unseen data. This split prevents overfitting, where a model performs well on training data but poorly on new inputs.

The division typically follows a standard ratio (e.g., 70:30 or 80:20), ensuring sufficient data for both learning and validation. Care is taken to maintain the class distribution in both subsets, particularly in multiclass network congestion scenarios, to prevent skewed evaluation results. When combined with oversampling techniques like SMOTE, the training set becomes balanced, allowing the model to learn equally from all classes. This methodological split lays the foundation for reliable performance assessment, enabling accurate measurement of predictive accuracy, precision, recall, and other metrics.

**4.5 Model Building**

Model building in this research involves training both existing and proposed algorithms on the prepared dataset to detect and predict congestion in SDN environments. The process begins with baseline models, which establish reference performance metrics. These models are carefully trained, validated, and compared to advanced techniques to determine improvements in accuracy, robustness, and generalization. The trained models incorporate features derived from the preprocessing and EDA stages, ensuring that all relevant patterns are considered.

**4.5.1 Existing Algorithm: SVM Classifier and Regressor**

Support Vector Machines (SVM) are a class of supervised learning algorithms widely used for classification and regression tasks. In classification, SVM seeks to find the optimal hyperplane that maximally separates data points of different classes in a high-dimensional space. For regression (SVR), the algorithm estimates a function that approximates continuous output values while tolerating deviations within a defined margin. SVMs are particularly robust for high-dimensional datasets, as they can

handle complex, nonlinear relationships using kernel functions such as the Radial Basis Function (RBF).

The working principle of SVM involves mapping input features into a higher-dimensional space where a linear separator can be found. The algorithm identifies support vectors—critical data points that lie closest to the decision boundary—and constructs the hyperplane that maximizes the margin between classes. During training, the model minimizes a cost function that balances misclassification errors and margin maximization. The kernel trick allows SVM to operate in nonlinear spaces efficiently, capturing intricate patterns in the data without explicit transformation.

**Algorithmic Steps (SVM Architecture):**

1. Map input features into a high-dimensional space using a kernel function.
2. Identify support vectors that define the decision boundary.
3. Compute the hyperplane that maximizes the margin between classes (for classification) or fits within an acceptable error margin (for regression).
4. Optimize the cost function to minimize classification/regression error while maximizing the margin.
5. Use the trained hyperplane to make predictions on new data.

**Disadvantages:** Despite its strengths, SVM has limitations. It is computationally expensive for large datasets due to the need to solve quadratic optimization problems. Performance is highly sensitive to kernel choice and hyperparameters, which require careful tuning. SVMs are less interpretable compared to tree-based models, making it difficult to understand feature contributions in complex scenarios. Additionally, SVM may struggle with noisy data and overlapping classes, potentially reducing accuracy in real-world network traffic datasets.

**4.5.2 Proposed Algorithm: TAO Tree with SMOTE**

**Definition and Overview:**

The proposed algorithm combines the TAO Tree classifier with SMOTE (Synthetic Minority Oversampling Technique) to address class imbalance and enhance multiclass classification performance in network congestion detection. TAO Tree is a fast, decision-tree-based learning method optimized for handling high-dimensional and multiclass data. It builds a hierarchical tree structure by recursively partitioning the feature space to maximize class separability. SMOTE, on the other hand, addresses the issue of imbalanced classes by generating synthetic samples for underrepresented classes, ensuring that all congestion levels are adequately represented during training. This combination results in a model that is both fast and accurate, capable of capturing intricate patterns in SDN traffic while avoiding bias towards majority classes.

**How It Works:**

The algorithm first preprocesses the dataset, including missing value handling, feature scaling, and label encoding. SMOTE is then applied to the training data to create synthetic instances for minority classes. TAO Tree constructs its decision hierarchy by evaluating feature splits that maximize class separability at each node. During tree construction, the algorithm selects the optimal feature and split point to minimize impurity in child nodes, recursively growing until stopping criteria are met (e.g., maximum depth or minimum samples per leaf). Once trained, the tree can classify unseen data efficiently by traversing from root to leaf nodes based on feature values. This process enables the model to adaptively learn complex decision boundaries for multiclass congestion states.

**Algorithmic Steps**

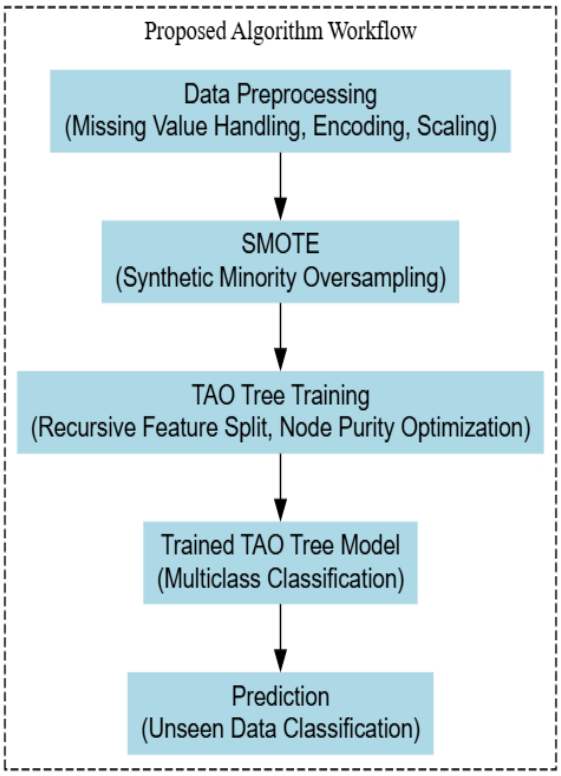
1. **Data Preprocessing:** Handle missing values, encode categorical variables, normalize features.
2. **SMOTE Application:** Oversample minority classes to balance class distribution.
3. **Tree Construction:** Recursively partition the feature space based on optimal feature splits to reduce impurity.
4. **Model Training:** Assign class labels to leaf nodes based on majority class of samples.
5. **Prediction:** Traverse the trained tree using feature values of unseen instances to predict class labels.

**Advantages of TAO Tree + SMOTE:**

- Handles multiclass problems efficiently.
- Reduces bias caused by imbalanced datasets.
- Fast training and inference due to optimized tree structure.
- Robust to high-dimensional feature spaces.
- Easy interpretability compared to black-box models.
- Improves predictive accuracy for underrepresented classes.

**Mathematical Intuition and Architecture**

The core intuition behind the algorithm is that TAO Tree partitions the feature space to maximize class purity while SMOTE ensures that minority class regions are sufficiently populated. SMOTE synthesizes new samples by interpolating between existing minority class instances, effectively "expanding" the decision boundary regions for these classes. TAO Tree then uses these balanced regions to construct a hierarchical decision structure, ensuring that each leaf node represents a well-defined class distribution. This combination allows the model to generalize better, reduce overfitting on majority classes, and capture subtle patterns in network congestion metrics




**CONCLUSION**

The study demonstrates the effectiveness of machine learning models in enhancing **SDN traffic management** through both **congestion detection** and **throughput/delay prediction**. In congestion detection, models like **SVM** and **KNN** provided decent accuracy (around 88–89%), but **TaoTree** significantly outperformed them, achieving 98.7% across all evaluation metrics, indicating its strong capability in reliably identifying congested states in the network. Similarly, for traffic control performance, SVM showed high errors and low predictive power, while KNN improved accuracy moderately. TaoTree again excelled, delivering minimal latency and throughput errors and achieving an $R^2$ of 0.969, reflecting precise prediction of network performance. These results collectively highlight that advanced ML techniques like TaoTree can significantly improve SDN efficiency by enabling proactive congestion detection and accurate traffic state prediction, which can reduce delays, optimize throughput, and improve overall network reliability.

**FUTURE SCOPE**

Future research can focus on further enhancing SDN performance using **hybrid and deep learning approaches** to handle large-scale, highly dynamic network environments. Incorporating **reinforcement learning** could allow adaptive traffic control that responds to real-time network changes more efficiently. Integration of **edge computing** and **IoT-based data sources** can provide more granular traffic state information, improving model accuracy. Additionally, exploring **ensemble learning techniques** combining multiple ML models may further enhance congestion detection and prediction robustness. Finally, the deployment of these models in **real-world SDN testbeds** or **5G network scenarios** will validate scalability, latency optimization, and practical applicability, enabling more intelligent and self-optimizing network management systems.

**REFERENCES**

1. Fortuna, C.; Poorter, E.D.; Škraba, P.; Moerman, I. Data driven wireless network design: A multi-level modeling approach. *Wirel. Pers. Commun.* **2016**, *88*, 63–77. [Google Scholar] [CrossRef]

2. Jiang, J.; Sekar, V.; Stoica, I.; Zhang, H. Unleashing the potential of data-driven networking. In Proceedings of the Communication Systems and Networks: 9th International Conference, COMSNETS 2017, Bengaluru, India, 4–8 January 2017; Revised Selected Papers and Invited Papers 9. Springer: Berlin/Heidelberg, Germany, 2017; pp. 110–126. [Google Scholar]

3. Kandula, S.; Sengupta, S.; Greenberg, A.; Patel, P.; Ronnie, C. The nature of data center traffic: Measurements & analysis. In Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement, Chicago, IL, USA, 4–6 November 2009; pp. 202–208. [Google Scholar]

4. Erickson, D. The beacon openflow controller. In Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, Hong Kong, China, 16 August 2013; pp. 13–18. [Google Scholar]

5. McKeown, N.; Anderson, T.; Balakrishnan, H.; Parulkar, G.; Peterson, L.; Rexford, J.; Shenker, S.;

Turner, J. OpenFlow: Enabling innovation in campus networks. In *ACM SIGCOMM Computer Communication Review*; Association for Computing Machinery: New York, NY, USA, 2008. [Google Scholar]

6. Adrichem, N.L.M.V.; Doerr, C.; Kuipers, F.A. OpenNetMon: Network monitoring in OpenFlow Software-Defined Networks. In Proceedings of the 2014 IEEE Network Operations and Management Symposium (NOMS), Krakow, Poland, 5–9 May 2014; pp. 1–8. [Google Scholar]
7. Shirali-Shahreza, S.; Ganjali, Y. Efficient implementation of security applications in OpenFlow controller with FleXam. In Proceedings of the 2013 IEEE 21st Annual Symposium on High-Performance Interconnects, San Jose, CA, USA, 21–23 August 2013; pp. 49–54. [Google Scholar]
8. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*, 2nd ed.; MIT Press: Cambridge, MA, USA, 2018. [Google Scholar]
9. Bellman, R. *Dynamic Programming*; Princeton University Press: Princeton, NJ, USA, 1957. [Google Scholar]
10. Howard, R.A. *Dynamic Programming and Markov Processes*; John Wiley & Sons: Hoboken, NJ, USA, 1960. [Google Scholar]
11. Bowling, M.; Veloso, M. Multiagent learning using a variable learning rate. *Artif. Intell.* **2002**, *136*, 215–250. [Google Scholar] [CrossRef]
12. Zhang, K.; Yang, Z.; Basar, T. Multi-agent reinforcement learning: A selective overview of theories and algorithms. In *Handbook of Reinforcement Learning and Control*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 321–384. [Google Scholar]
13. Zhang, J.; Liu, Z.; Zhu, Y.; Shi, E.; Xu, B.; Yuen, C.; Niyato, D.; Debbah, M.; Jin, S.; Ai, B.; et al. Multi-Agent Reinforcement Learning in Wireless Distributed Networks for 6G. *arXiv* **2024**, arXiv:2502.05812. [Google Scholar]
14. Hernandez-Leal, P.; Kartal, B.; Taylor, M.E. A survey of learning in multiagent environments: Dealing with non-stationarity. *Artif. Intell.* **2018**, *256*, 1–35. [Google Scholar]
15. Neary, C.; Xu, Z.; Wu, B.; Topcu, U. Reward machines for cooperative multi-agent reinforcement learning. In Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS), Online, 3–7 May 2021; pp. 934–942. [Google Scholar]