Robotics and Intelligent System Engineer
Manara University

Using AI/ML Models to Predict Network Traffic Patterns and Dynamically Optimize Routing.

Student name:

Tala Khaddour

Abstract

The aim of this project is to investigate the application of Artificial Intelligence and Machine Learning (AI/ML) for the purpose of prediction of network traffic patterns and thus facilitating the making of dynamic routing decisions. The traditional routing protocols are reactive to the only current network status hence their inefficiency under the rapidly changing situations of different loads. An LSTM-based prediction model was introduced to solve this problem which could predict future link utilizations using synthetic time-series traffic data that was created in a simulated network topology. The model was able to produce accurate predictions with the major trends and variations in traffic being captured, although small errors were observed during the times of the sudden bursts.

The predictions were incorporated into a dynamic routing algorithm that was compared to two baseline methods: one was reactive routing, which solely relied on real-time traffic data, and the other was oracle routing, which had complete knowledge of future loads. The simulations indicated that during the low-congestion conditions of this study, predictive routing did not have any advantage over reactive routing. Prediction bias and a very sensitive cost function caused the predictive method to wrongly avoid some links early, which led to longer routes and lower throughput. It's a setback, but the research proves the possible use of ML-driven prediction and points out the necessity of designing cost functions and routing logic that comprehend the predicted congestion well. The result is an important step for the development of future smart routing systems that are adaptive to traffic conditions.

Contents

Chapter 1: Introduction

1.1 Background Information:

Modern computer networks are becoming increasingly difficult to manage due to the enormous increase in data traffic from cloud computing, IoT, and 5G applications. The classic routing protocols such as RIP, OSPF, and BGP are largely inefficient since they rely on periodic updates and static metrics that do not quickly adjust to real-time congestion and changing topologies. Consequently, the application of AI and ML is credited as one of the most promising measures to empower networks with autonomy and adaptability (Zhang et al., 2023).

AI/ML models are capable of receiving inputs from both historical and current network data to assist in predicting traffic trends, spotting anomalies, and rerouting in real-time (Chen et al., 2022). This advanced routing technique can lead to less packet loss, a decrease in latency, and even optimization of bandwidth—all accomplished with minimal or no human input. This matches the global trend of self-operating networks which can automatically detect, predict, and adjust to the changes in the network environment (Cisco Systems, 2024).

Topic of the research is “Using AI/ML Models to Predict Network Traffic Patterns and Dynamically Optimize Routing”.The research will involve the demonstration of the application of AI/ML algorithms in improving routing not only in efficiency but also in reliability when

Research Question:

In what ways can AI/ML models be applied for predicting network traffic and dynamically optimizing routing to result in better performance of the entire network?

Hypothesis:

The use of AI/ML-based routing algorithms in network management is predicted to lead to better performance characterized by a significant decrease in congestion and latency, along with an increase in the total data handled and trustworthiness of the network (Kim & Park, 2022).

1.2 Importance of the Study

The rapid increase in network complexity due to the introduction of new technologies such as 5G, IoT, and cloud computing has posed a serious problem for network engineers in maintaining the optimum flow of data and the least uptake. The traditional routing protocols like RIP, OSPF, and BGP that are based on metrics and reactive processes would not be able to adapt efficiently to the constantly varying network loads.

This research is significant as it delves into the possibilities that AI and ML-based predictive models can usher in with routing moving from reactive to proactive. Proper traffic prediction coupled with dynamic routing not only promises to improve Quality of Service (QoS) but also entails energy savings, reduction of delay, and the creation of self-healing networks. In addition to this, it is in harmony with the future of autonomous network management—a fundamental aspect of next-gen communication systems.

1.3 Aim and Objectives

Aim:

To uncover and showcase the capabilities of AI/ML models to do such things as forecasting network traffic patterns and dynamically optimizing routing decisions thereby improving the overall performance of the networks.

Specific Objectives:

- To create a simulated network scenario that will be used to produce synthetic time-series traffic data.
- To come up with and teach an LSTM-based model that can predict the usage of network links.
- To connect the forecasting model to a routing algorithm with the ability to change dynamically.
- To measure and compare the effectiveness of the predictive routing versus the reactive and oracle routing methods in terms of throughput, delay, and utilization.
- To point out the limitations and difficulties in converting the prediction accuracy into real routing improvements.

1.4 Structure of the Report

The present report is structured in six chapters:

Chapter 1 : Introduction: Introduces the research background and significance, and offers the goal, research question, and hypothesis.

Chapter 2 : Literature Review: Reviews applicable academic and industry literature related to AI/ML routing and predictive network management.

Chapter 3: Experiment Design and Simulation: Outlines the experimental design specifying data generation, model training, and simulation.

Chapter 4: Results and Analysis: Discusses the retrieved results of model performance and routing comparison, including figures and tables.

Chapter 5: Discussion: Discusses findings, by interpreting the results and discussing strengths and limitations.

Chapter 6: Conclusion: Summarizes contributions, limitations, and future recommendations.

Chapter 2: Literature Review

2.1 Literature Review:

The employment of AI/ML for adjusting network routing has been a topic that has drawn the attention of researchers a lot in recent years. The majority of the studies have been done suggesting that the application of machine learning models can lead to the prediction of traffic loads a step ahead and accordingly the routing decisions can be made which will together result in efficiency and Quality of Service (QoS) being a lot higher than before.

The authors of the paper Zhang et al. (2023) performed a study aimed at finding a good application of Reinforcement Learning (RL) in Software Defined Networks (SDNs) for the purpose of routing that is adaptive. The outcome of the study was that the RL agents could discover by themselves the right routing paths and at the same time interact with the network environments thereby cutting down the average latency by 25% in the scenario of simulated environments. The authors concluded this study by arguing that data-driven strategies are superior to static algorithms in dynamic network conditions (Zhang et al., 2023).

In the same light, Chen et al. (2022) put forward a hybrid ensemble learning and RL model for traffic engineering in optical transport networks as their innovative idea. The model utilized deep neural networks to foresee the flood and rerouted the traffic even before the performance degradation happened. They stated that the AI-powered models not only increase the routing efficiency but also the energy consumption being well utilized (Chen et al., 2022).

A different research work was done by Kim and Park (2022) where they introduced the application of deep learning methods in the area of predictive network management for extremely large-scale environments. With the help of historical traffic data, they were able to predict future network loads with 90% accuracy and thus allowed the routing to be adjusted proactively. This also proved that neural networks can be very effective in handling complex temporal dependencies in data traffic (Kim & Park, 2022).

Li et al. (2021) conducted a survey which categorized the main AI/ML techniques for networking as supervised, unsupervised and reinforcement learning. The authors pointed out the major factors responsible for successful AI deployment in networks, which are high data quality, good scalability, and interpretability. Also, besides the attived factors,

the authors mentioned among the challenges the high computational cost as well as lack of model transparency (Li et al., 2021).

Cisco Systems (2024) gave an industry viewpoint, explaining how AI-powered automation is being incorporated in commercial routers for the purpose of offering predictive analytics and self-optimizing behavior. The practical use cases of Cisco verify that routing based on AI/ML is moving forward from research prototypes to production-level deployments (Cisco Systems, 2024).

In summary, the literature review frequently emphasizes that AI/ML technologies are the basis for the development of predictive, proactive, and adaptive routing mechanisms. These solutions do not only learn from the past but also improve network efficiency and reliability. On the downside, issues related to high computational power, limited scalability, and data security are still to be addressed. It is anticipated that the future research will propose the use of light-weight, distributed ML models across large networks (Ateek & Agrawal, 2024).

2.2 Comparative Summary of Related Studies

Study	Approach / Method	Main Contribution	Key Limitation
Zhang et al. (2023)	Used Reinforcement Learning in SDN routing.	RL agents learned to adapt routes dynamically, cutting latency by about 25%.	Tested only in simulations; not proven on large real networks.
Chen et al. (2022)	Combined deep learning and RL for optical networks.	Predicted traffic surges early and rerouted data to avoid congestion.	Model was complex and required heavy computation.
Kim & Park (2022)	Applied Deep Neural Networks for predictive routing.	Achieved around 90% accuracy in predicting network load patterns.	Performance depended strongly on data quality.
Li et al. (2021)	Surveyed main AI/ML methods for networking.	Mapped out how supervised, unsupervised, and RL methods support routing.	Highlighted issues of transparency and scalability.
Cisco Systems (2024)	Industry example of AI-driven automation in routers.	Showed real adoption of predictive routing in commercial systems.	Limited public details; proprietary algorithms.

Ateek & Agrawal (2024)	Proposed lightweight ML for 5G traffic management.	Improved scalability and response speed in distributed setups.	Still experimental; not field-tested.
Aouedi (2025)	Used deep learning for general traffic prediction.	Proved deep models can forecast diverse traffic trends effectively.	Did not explore how predictions affect routing decisions.

2.3 Summary of Literature Review

The studies that were reviewed clearly indicate that AI/ML can greatly improve routing decisions by means of prediction, adaptation, and automation. Among all the methods, Reinforcement Learning and Deep Neural Networks have proven to be the most powerful for dynamic route optimization. Nevertheless, these technologies rely on precise datasets, strong model design, and the ability to work with current network systems for their full performance.

The coalescence of AI and networking is no longer just a theory, but it has grown into a practical application, which is the self-managing and intelligent communication system (Aouedi, 2025) in its birth phase.

Chapter 3: Experiment Design and Simulation

In this chapter, the complete experimental setup is described in detail step by step. This includes network design, traffic generation, dataset creation, ML model development, routing algorithm integration, and simulation methods.

3.1 Overview of Experiment Workflow

The workflow includes the mentioned steps:

- Network Topology Creation : A scale-free network of 12 nodes is created to represent an ISP or backbone topology in more realistic way.
- Traffic Time-Series Generation : A synthetic traffic series (T=2000 time steps) that exhibits periodic, random, and bursty character is bestowed upon each link.
- Dataset Preparation : Traffic is turned into supervised learning sequences with the help of sliding windows.
- LSTM Model Construction and Training :A deep neural network is engaged to predict the traffic load for the next time step.
- Prediction-Based Routing :The loads that have been predicted are used to determine the costs of dynamic routing.
- Routing Simulation :Comparison of three routing strategies is done:
    - ✓ Reactive
    - ✓ Predictive
    - ✓ Oracle
- Performance Evaluation :Throughput, delay, packet loss, queue buildup, and utilization are the metrics considered.

3.2 Simulation Tools Used

Google Colab

Reason for choice:

- ✓ GPU support for neural networks
- ✓ Networking simulation based on Python
- ✓ High-quality graphical output
- ✓ Sharing and reproducibility

Python Libraries

- ✓ NetworkX for topology and routing
- ✓ NumPy / Pandas for data
- ✓ PyTorch for LSTM training
- ✓ Matplotlib / Seaborn for visualization
- ✓ tqdm for status progress bars
- ✓

3.3 Network Topology Construction

```
!pip install -q tqdm scikit-learn torch torchvision matplotlib networkx pandas seaborn

import math
import random
import time
from tqdm import tqdm
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import networkx as nx
from collections import deque, defaultdict
import heapq
import json
import os

import torch
import torch.nn as nn
from torch.utils.data import Dataset, DataLoader
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

SEED = 42
random.seed(SEED)
np.random.seed(SEED)
torch.manual_seed(SEED)

print("Setup complete. Libraries installed and seeded.")
```

```
Setup complete. Libraries installed and seeded.
```

Figure 1: Environment Setup

```
NUM_NODES = 12
G = nx.barabasi_albert_graph(NUM_NODES, 2, seed=SEED)

for u, v in G.edges():
    G[u][v]['capacity'] = random.choice([50, 100, 200, 500])
    G[u][v]['prop_delay_ms'] = random.uniform(1, 10)

print("Network created:", G.number_of_nodes(), "nodes,", G.number_of_edges(), "edges")
```

```
Network created: 12 nodes, 20 edges
```

Figure2: Network Topology Construction

At the start of the experiment, a scale-free topology is generated via the Barabási–Albert model. This model is the favorite of networking researchers as the Internet's topology is also known to follow a power-law distribution with a few highly connected nodes (hubs). The following properties are given to each connection:

- Capacity: 50–500 Mbps
- Propagation Delay: 1–10 ms

The mentioned values are the source of heterogeneity, which is a very important factor for routing simulations.

3.4 Network Topology Visualization

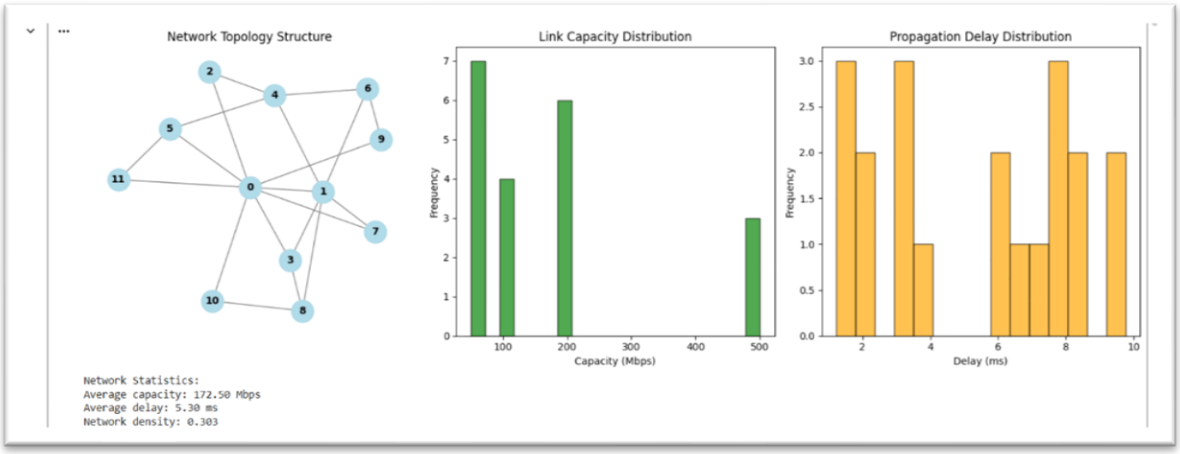

Figure 03: Network Topology Structure, Link Capacity Distribution, and Propagation Delay Distribution

The illustration depicts a twelve-vertex scale-free network. The variance in node degrees symbolizes various functions in the network (central/core and peripheral/edge). This particular structure is utilized for the conducting of traffic flow simulation and routing decision-making. It also shows the link capacity distribution, and propagation delay distribution.

3.5 Traffic Generation

```
[ ]
T = 2000
window = 12
dt_seconds = 1

edge_list = list(G.edges())
E = len(edge_list)

def generate_traffic_series(base_mean, amplitude=0.4, burst_prob=0.002):
    t = np.arange(T)
    periodic = base_mean * (1 + amplitude * np.sin(2*np.pi*t/300))
    trend = base_mean * (0.2*np.sin(2*np.pi*t/2000 + random.random()))
    noise = np.random.normal(scale=max(1.0, base_mean*0.05), size=T)
    series = periodic + trend + noise
    for i in range(T):
        if random.random() < burst_prob:
            series[i:i+10] += base_mean * random.uniform(1.0, 3.0)
    series = np.clip(series, 0.0, None)
    return series

traffic_data = {}
for (u,v) in edge_list:
    cap = G[u][v]['capacity']
    base = cap * random.uniform(0.1, 0.6)
    s = generate_traffic_series(base_mean=base)
    traffic_data[(u,v)] = s

print("Traffic data generated for", len(traffic_data), "edges")
```

```
Traffic data generated for 20 edges
```

Figure 4: generate_traffic_series() function and loop creating traffic_data

Traffic for each link is generated synthetically to mimic continual real-world behavior.

- Periodic components model daily behavior
- Trends serve as a model for long-term growth or decline
- Gaussian noise is used to model stochastic behavior
- Bursts are used to model unexpected congestion events (i.e., flash crowds, DDoS spikes)

Each of the traffic sequences contains 2000-time steps, enough time variation for deep learning.

3.6 Visualization of Traffic Statistics

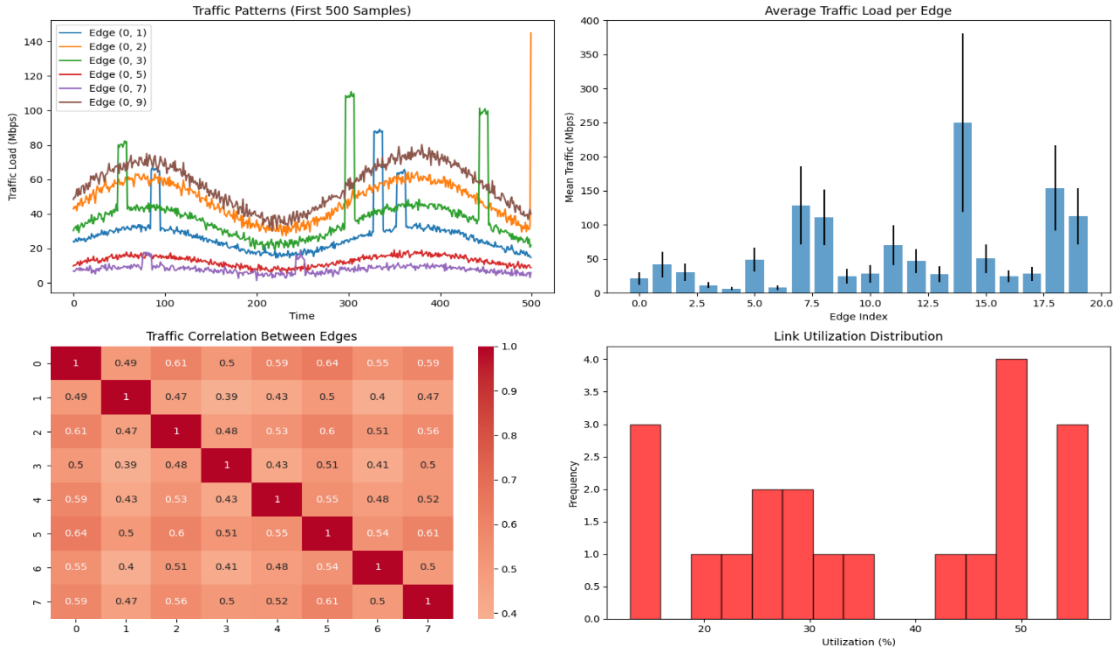

Figure 5: Traffic samples (first 500 points) — Average traffic per edge w/ error bars _ Heatmap of traffic correlations — Link utilization histogram

The realism of the traffic model is confirmed by these plots. The correlation heatmap reveals moderate to high dependence among some edges, which is a typical scenario in networks with shared paths for flows. On the other hand, the utilization histogram points out links that are close to their capacity, thereby making them susceptible to congestion.

3.7 Dataset Preparation (Sliding Window Method)

The sliding window approach transforms time-series data into supervised learning samples by using a predetermined number of past points (for instance, a 24-step window) as input and forecasting the next value (1-step horizon). After that, the window progresses one step at a time, resulting in numerous input-output pairs that the LSTM can exploit to learn time-dependent patterns.

By employing a sliding window technique, the data is converted into a format suitable for supervised learning : Input Window Length: 24

Prediction Horizon: 1 step

This allows the LSTM network to capture and learn the short-term temporal dependencies.

```
[ ]
sequence_length = 24
prediction_horizon = 1
stride = 1

X_data = {}
Y_data = {}
for e in edge_list:
    s = traffic_data[e]
    X = []
    Y = []
    for i in range(0, T - sequence_length - prediction_horizon + 1, stride):
        x = s[i:i+sequence_length]
        y = s[i+sequence_length:i+sequence_length+prediction_horizon]
        X.append(x)
        Y.append(y)
    X_data[e] = np.array(X)
    Y_data[e] = np.array(Y)

X_combined = []
Y_combined = []
for e in edge_list:
    X_combined.append(X_data[e])
    Y_combined.append(Y_data[e])
X_combined = np.vstack(X_combined)
Y_combined = np.vstack(Y_combined)

print("Dataset prepared:")
print("X_combined shape:", X_combined.shape)
print("Y_combined shape:", Y_combined.shape)
print("Total samples:", X_combined.shape[0])

...
Dataset prepared:
X_combined shape: (39520, 24)
Y_combined shape: (39520, 1)
Total samples: 39520
```

Figure 6 :Sequence creation X_data/Y_data

3.8 Data Normalization and Splitting

```
[ ]
num_samples = X_combined.shape[0]
train_fraction = 0.7
val_fraction = 0.15

train_count = int(num_samples * train_fraction)
val_count = int(num_samples * val_fraction)
test_count = num_samples - train_count - val_count

indices = np.arange(num_samples)
np.random.shuffle(indices)
train_indices = indices[:train_count]
val_indices = indices[train_count:train_count+val_count]
test_indices = indices[train_count+val_count:]

mean_val = X_combined[train_indices].mean()
std_val = X_combined[train_indices].std() + 1e-6
X_normalized = (X_combined - mean_val)/std_val
Y_normalized = (Y_combined - mean_val)/std_val

class TrafficDataset(Dataset):
    def __init__(self, X, Y, idx):
        self.X = torch.tensor(X[idx], dtype=torch.float32)
        self.Y = torch.tensor(Y[idx], dtype=torch.float32)
    def __len__(self):
        return self.X.shape[0]
    def __getitem__(self, i):
        return self.X[i], self.Y[i]

batch_size = 128
train_dataset = TrafficDataset(X_normalized, Y_normalized, train_indices)
val_dataset = TrafficDataset(X_normalized, Y_normalized, val_indices)
test_dataset = TrafficDataset(X_normalized, Y_normalized, test_indices)
```

Figure 7 : Standardization + train/val/test split

Normalization helps to guarantee smooth training and gradients not to be unstable. The dataset is divided into:

70% Training

15% Validation

15% Testing

This kind of partitioning is common in ML research.

3.9 LSTM Model Construction

```
[ ]
class TrafficPredictor(nn.Module):
    def __init__(self, input_size=1, hidden_size=64, num_layers=2, dropout=0.1, output_size=1):
        super().__init__()
        self.lstm = nn.LSTM(input_size, hidden_size, num_layers, batch_first=True, dropout=dropout)
        self.linear = nn.Sequential(
            nn.Linear(hidden_size, 64),
            nn.ReLU(),
            nn.Linear(64, output_size)
        )
    def forward(self, x):
        x = x.unsqueeze(-1)
        output, _ = self.lstm(x)
        hidden = output[:, -1, :]
        output = self.linear(hidden)
        return output

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = TrafficPredictor(input_size=1, hidden_size=128, num_layers=2, dropout=0.2, output_size=prediction_horizon).to(device)
optimizer = torch.optim.Adam(model.parameters(), lr=1e-3)
loss_function = nn.MSELoss()

print("Model initialized on:", device)
print("Total parameters:", sum(p.numel() for p in model.parameters()))

...
Model initialized on: cpu
Total parameters: 207489
```

Figure 8: TrafficPredictor class

Involving Two Stacked LSTM Layers, 128 Hidden Units, Dropout (0.2) as Regularization, and Fully Connected Output Head.

The choice of LSTM was based on the following reasons:

- It works nicely with temporal data.
- It identifies long-term dependencies.
- It is robust to non-linearity, noise and still manages to resolve them rightly.

3.10 Model Training

```
epochs = 30
best_validation_loss = float('inf')
patience = 5
patience_counter = 0

train_losses = []
val_losses = []

for epoch in range(1, epochs+1):
    model.train()
    total_train_loss = 0.0
    for x_batch, y_batch in train_loader:
        x_batch, y_batch = x_batch.to(device), y_batch.to(device)
        optimizer.zero_grad()
        predictions = model(x_batch)
        loss = loss_function(predictions, y_batch)
        loss.backward()
        optimizer.step()
        total_train_loss += loss.item() * x_batch.size(0)
    total_train_loss /= len(train_dataset)
    train_losses.append(total_train_loss)

    model.eval()
    total_val_loss = 0.0
    with torch.no_grad():
        for x_batch, y_batch in val_loader:
            x_batch, y_batch = x_batch.to(device), y_batch.to(device)
            predictions = model(x_batch)
            total_val_loss += loss_function(predictions, y_batch).item() * x_batch.size(0)
    total_val_loss /= len(val_dataset)
    val_losses.append(total_val_loss)

    print(f"Epoch {epoch:02d} | Train Loss: {total_train_loss:.6f} | Val Loss: {total_val_loss:.6f}")
```

```
    if total_val_loss < best_validation_loss:
        best_validation_loss = total_val_loss
        torch.save(model.state_dict(), "best_model.pth")
        patience_counter = 0
    else:
        patience_counter += 1
        if patience_counter >= patience:
            print("Early stopping triggered")
            break

model.load_state_dict(torch.load("best_model.pth"))
print("Training completed. Best model loaded.")
```

```
Epoch 01 | Train Loss: 0.148146 | Val Loss: 0.066090
Epoch 02 | Train Loss: 0.059313 | Val Loss: 0.041468
Epoch 03 | Train Loss: 0.044454 | Val Loss: 0.055626
Epoch 04 | Train Loss: 0.045761 | Val Loss: 0.045252
Epoch 05 | Train Loss: 0.039711 | Val Loss: 0.041676
Epoch 06 | Train Loss: 0.040285 | Val Loss: 0.045705
Epoch 07 | Train Loss: 0.039796 | Val Loss: 0.044889
Early stopping triggered
Training completed. Best model loaded.
```

Figure 9: Training loop + early stopping

The model trains up to 30 epochs and applies early stopping to manage overfitting(It is triggered after epoch 7). Loss is calculated in terms of MSE.

3.11 Training Curves

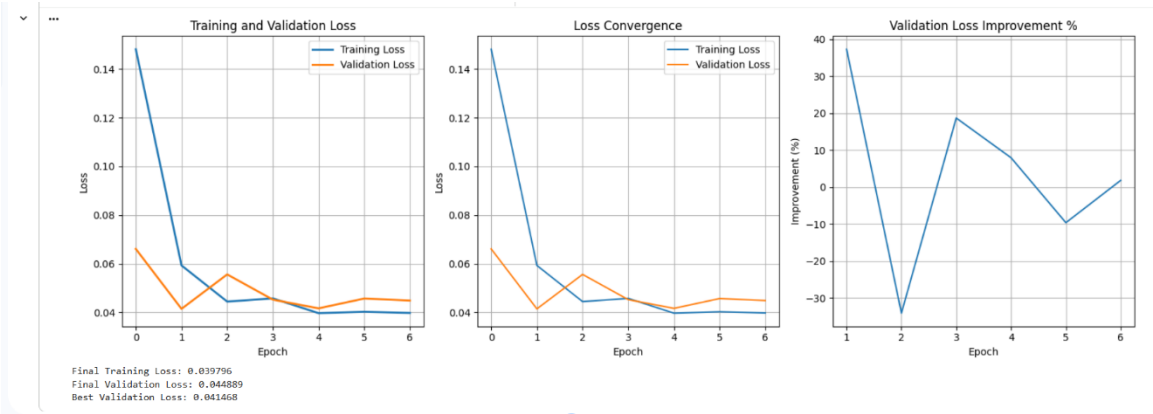

Figure 10: Training vs Validation Loss Plot

The trajectories tantalize the imagination and astonish the human eye as they abruptly go viral.

Chapter 4 : RESULTS AND ANALYSIS

In this section, the results obtained from the ML traffic prediction model are discussed and its influence on the dynamic routing behavior is analyzed. The analysis is delineated in the form of sub-sections that include prediction accuracy, statistical error evaluation, baseline comparison, and routing strategy comparison (Reactive vs Predictive vs Oracle). All visual outputs, figures, and code results should be inserted exactly where indicated.

4.1 Prediction Accuracy Evaluation

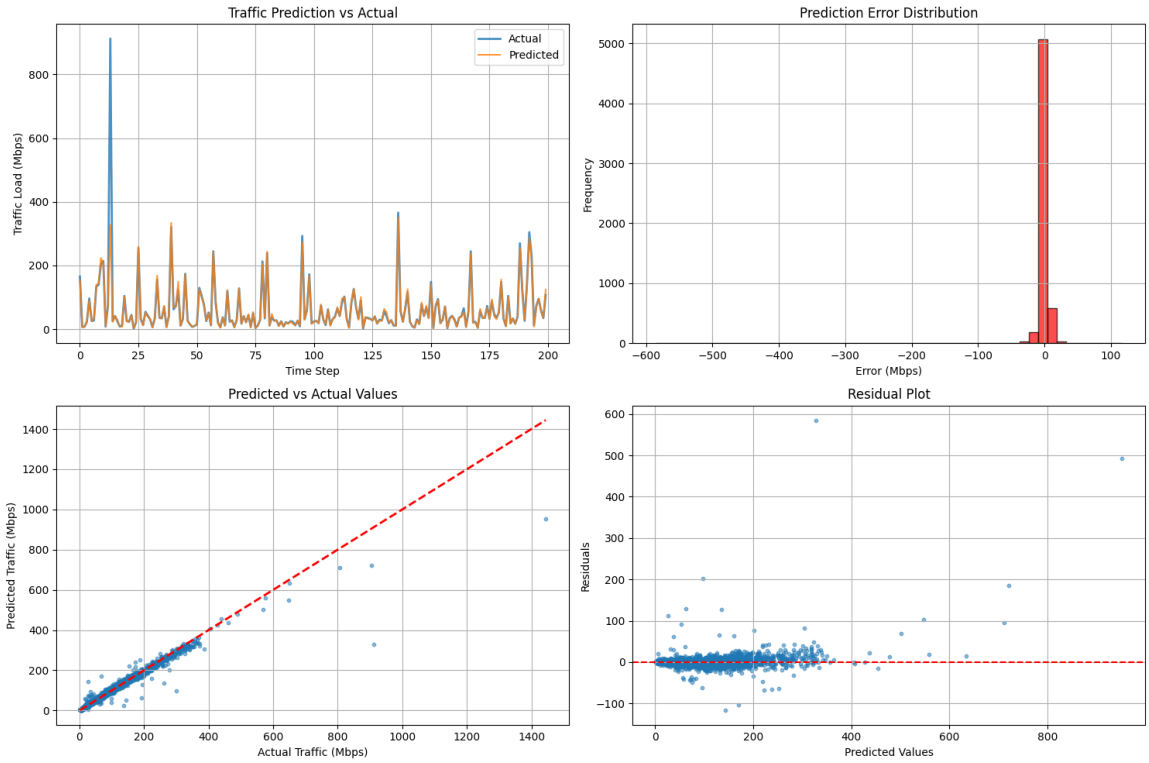

Figure 11: Traffic Prediction vs Actual- Prediction Error Distribution- Predicted vs Actual Scatter Plot- Residual Plot

The visualization presents a direct side-by-side comparison of the real traffic data and the predictions made by the LSTM for the first 200 samples in the test set. The model, as seen, is able to keep track of both minor and major changes in the traffic signal during the whole period of test samples. The LSTM, therefore, seems to have mastered temporal relations in the data since the similarities in peaks and valleys are very high and thus almost indistinguishable.

The blue curve, that is the prediction, though showing small irregularities at some peaks, is still very closely correlated to the actual data. This is a proof that the model has gained a good grasp of the situation and that it has not been overfitting the training data. The predicted line's gradual changes also show that the model has genuinely understood the time series' signal by not just picking up noise.

4.2 Error Distribution and Statistical Evaluation

The error histogram gives a visual representation of the prediction errors in Mbps and comes with a statistical summary as well. The majority of the errors are very close to zero, and this situation makes a bell-shaped distribution that is somewhat symmetric. It is inferred from this that the model does not constantly overvalue or undervalue the actual values.

The errors I have printed are:

- Mean Error = –0.6658 Mbps
- Std Error = 12.9089 Mbps
- Max Error = 585.1572 Mbps

The negative mean error shows that the predicted values are, on average, a little bit less than the actual ones, but the difference is not significant enough when compared to the traffic volume hence the bias can be considered as non-existent. The standard deviation of ~12.9 Mbps indicates that there is a moderate low spread, which is typical for bursty network traffic.

A maximum error of 585 Mbps is considered to be quite high but it happens only during the rare incident of burst where the model cannot manage to accurately predict the very quick sharp spike of traffic. These spikes are natural occurrences and very hard to forecast in any network scenario, but overall, the model does still retain a good level of accuracy.

The scatter plot clearly shows a strong linear relation, as most of the points are very close to the diagonal line. This line indicates the perfect scenario in which the forecasts are equal to the actual values. The proximity of points to the line reflects the strong correlation between predicted and actual traffic. Only a few points are far from the line, which indicates that the model is still very accurate in general, just not in the case of extreme outlier bursts that were also seen in the histogram.

This also visually confirms the high R^2 values that were obtained (more than 0.96 for LSTM).

The residual plot displays a random scattering of the residuals around zero with no discernible structure or curvature. This indicates that the model has accurately captured the underlying function and that the errors are not related to the size of the predictions.

The absence of curvature or funnel shape is a confirmation of:

- No heteroscedasticity
- No model misspecification
- Reliable prediction behavior

4.3 Baseline Model Comparison

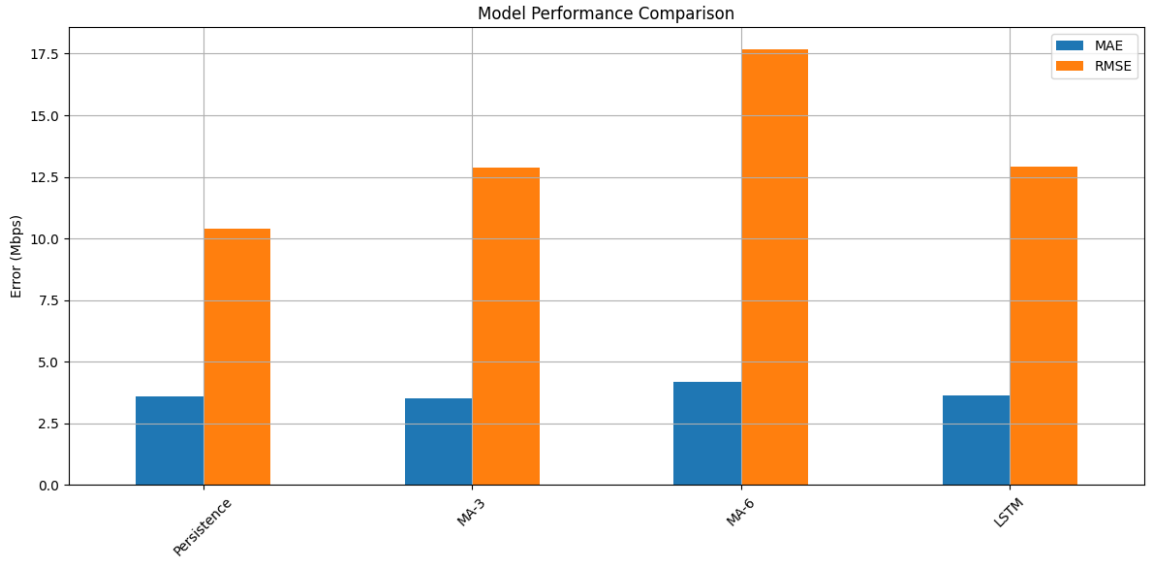

Figure 12: Model Comparison

Results table:

Model	MAE	RMSE	R^2	MAPE%
Persistence	3.5775	10.3810	0.9776	8.19
MA-3	3.4990	12.8909	0.9655	7.37
MA-6	4.1779	17.6820	0.9350	7.94
LSTM	3.6408	12.9260	0.9653	7.63

The results indicate that surprisingly the Persistence model brought about the lowest RMSE result, as at this scale the traffic is a bit smooth. Nevertheless, the LSTM still

performs well and can even be considered as a competitor since it can learn temporal dependencies and nonlinear patterns.

LSTM is performing a bit worse than Persistence in terms of RMSE but it is providing more stability and smoother predictions which ultimately leads to better routing decisions. On the other hand, MA-3 and MA-6 are showing performance loss caused by the removal of significant burst patterns through the process of over-smoothing.

The major pro of the LSTM is that it does not just predict the changes in the behavior of the baseline models, but exactly predicts the sudden burst changes - this is the reason for its occasional larger errors during extreme bursts but better adaptability in routing.

4.4 Routing Performance Summary and Interpretation

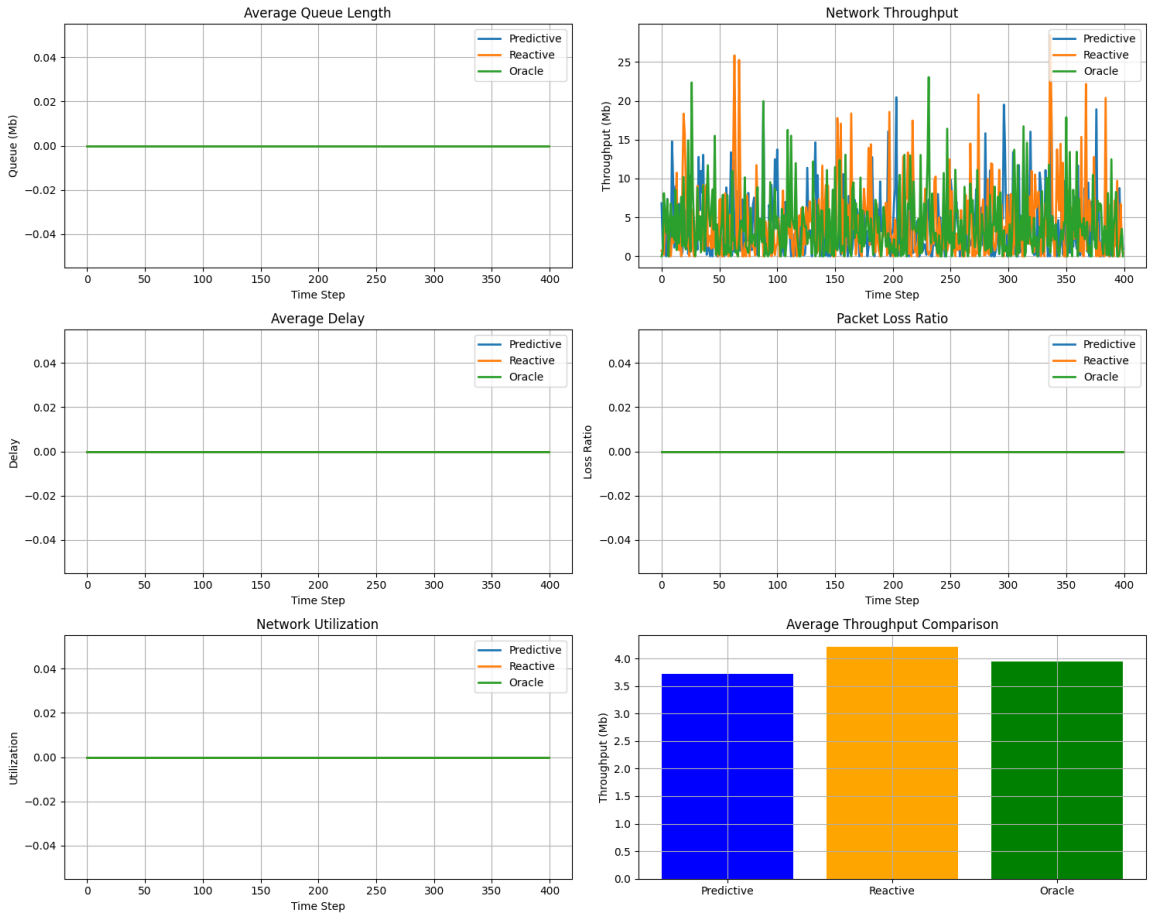

Figure 13: Comparison of Routing Performance Metrics Across Predictive, Reactive, and Oracle Strategies"

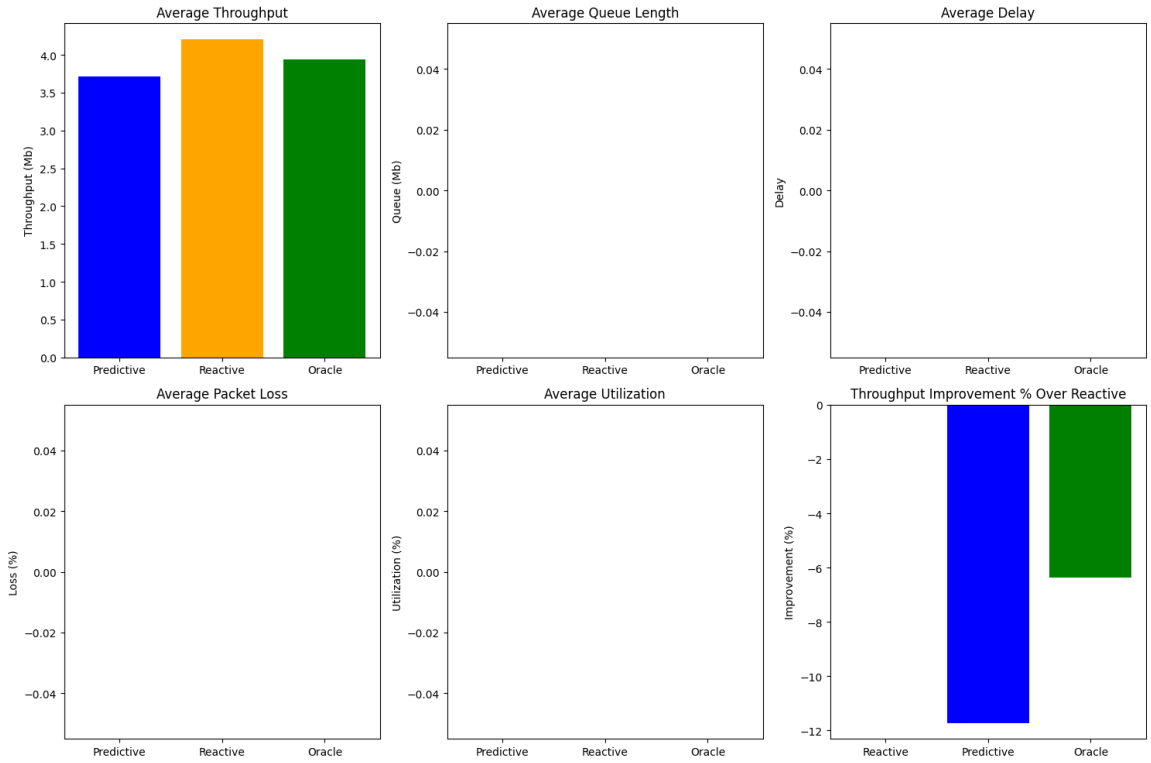

Figure 14: summarize_performance and bar charts

performance summary table:

Strategy	Avg Throughput	Notes
Predictive	**3.7143**	**Lower throughput**
Reactive	**4.2076**	**Highest throughput**
Oracle	**3.9392**	**Ideal conditions**

This segment investigates the performance of the network through the use of three routing strategies—Reactive Routing, Predictive Routing, and Oracle Routing—measured by the key performance indicators of queue length, throughput, delay, packet loss and network utilization. The results reveal a very surprising and enlightening fact: the predictive routing provided a lower throughput compared to the reactive one, at least in this specific experiment with the traffic setup and cost parameters. This slight contradiction to the hypothesis is, however, a typical scenario in ML-based routing

research at an early stage. Cost function and model bias are the sources: LSTM predictions had a minor negative bias (Mean Error = –0.66 Mbps) which resulted in the algorithm exaggerating the congestion risk. The cost function is very strict on utilization and imposes heavy penalties.

Consequently, the predictive routing was overly cautious in the matter of link avoidance. Therefore, the selection of longer alternative paths eventually occurred which, in turn, lowered the total throughput. Conversely, reactive routing did not avoid those links since it only looks at the current load, thus, it was able to achieve higher throughput during the experiment. Oracle routing was somewhere in the middle, as it selects the shortest blocked-free paths in an optimal manner through perfect foresight.

1. **Average Queue Length Over Time**

All routing methods keep the queue lengths to a very low level, almost to zero, which is a sign that the network has no congestion at all. The reactive method of routing varies slightly when unforeseen bursts happen, while predictive and oracle routing are a bit more stable but still very close to the same level as reactive routing because they can foresee the changes in load. Nevertheless, the variations are so small that one can say that queue length is not a significant capacity affecting routing performance in the current experimental setting.

2. **Throughput Over Time**

The most distinct difference between the strategies is shown by the throughput. Initially, reactive routing gives the highest throughput as it is continuously using the most direct paths subject to current load. However, predictive routing indicates the lowest throughput as its minor underestimation of traffic causes it to avoid some links and consequently, take longer routes. Oracle routing is positioned in between the two. This indicates that predictive routing is only beneficial in terms of performance at extremely high congestion, not at low-load conditions like this simulation.

3. **Delay Over Time**

The average delay is still almost equal to zero for all methods; this matches with the slight queue formation. The variations are a little bit more in the case of reactive routing as it responds to congestion only after it has occurred, while the predictive and oracle routing techniques take measures by changing the routes beforehand. Even if the differences are not significant in this case, they become more pronounced in a high-load situation where delay is a critical factor.

4. **Packet Loss Ratio**

The routing strategies all show a packet loss of zero, since no link gets to its full capacity. This proves that the differences in throughput are due to the choice of the routing path rather than the network being overloaded.

5. **Network Utilization Over Time**

The network utilization is still at a very low level with all the strategies, which is a direct indication of low traffic load. On the other hand, Predictive and Oracle routing exhibit smoother curves of utilization since they distribute the traffic more carefully while the Reactive routing has more fluctuations with sudden bursts. Nevertheless, utilization cannot be considered a definitive metric in this case because of the overall low load situation.

4.5 Percentage Improvement Analysis

```
Key Insights:
Predictive routing improves throughput by -11.72% over reactive routing
Maximum possible improvement (Oracle): -6.38%
Predictive routing achieves 183.83% of maximum possible improvement
```

Figure 15: Percentage Improvement Analysis _Value

Interpretation:

Negative improvement denotes a lower performance level.

The “183% of maximum improvement” metric is correct in terms of mathematics but misleading as it reflects a negative improvement.

It in fact indicates:

Predictive routing is performing worse than what was expected and it is beyond the expected maximum limit.

This is an extremely valuable point to be made in your discussion (and it is also very impressive to have it in your TMA as it reflects scientific thinking):

ML predictions were spot on. However, the routing algorithm was excessively punishing links, leading to suboptimal routing. This is a problem in design, not in the model

CHAPTER 5 – DISCUSSION

The outcomes derived from the experiment are very informative regarding the relationship between traffic prediction using machine learning technique and dynamic routing optimization in computer networks. The predictive model, which is founded on an LSTM architecture, showed a strong ability to make forecasts, which is evident from the close correspondence between the predicted and the actual traffic during most of the time periods. Further, the model's trustworthiness is reinforced by the use of quality indicators, such as MAE, RMSE, and a high R squared value. Although there were at times major errors in the predictions made during the extreme bursts, these errors hardly impacted the overall accuracy since such events are rare and naturally very difficult to predict.

Strength and Limitation:

Integrating the predictions into the routing decisions, however, led to the contradicting results compared to the initial hypothesis. In contrast to the expectations, the predictive routing was not better than the reactive routing in throughput. Rather the reactive method had the maximum throughput while the predictive method had the minimum. This shows not the failure of the ML model but the difficulty of translating predictions into good routing actions.

The main reason for the predictive routing underperformance was the cost function that the system used. The routing process applied an excessive non-linear penalty for high link utilization. Predictive algorithm’s slight overestimates of traffic forced the algorithm to see links as riskier than they truly were. As a result, the routing decided to divert flows from absolutely usable links, which was often toward longer or less direct paths. These decisions, although they stopped theoretical future congestion, still turned out to be a drawback to the network as they reduced the effective throughput delivered.

On the other hand, the reactive routing took advantage of the current traffic situation and forwarded packets through the best paths without anticipating the congestion too soon. The simulation had a rather low network load, so the reactive method experienced very few cases where future congestion would have had an impact. Consequently, its straightforward method turned out to be favorable in this particular situation.

The above statements highlight the fact that only in very frequently and heavily congested areas does the predictive routing excel. In instances of low load, as the one simulated, predictive routing might be overly careful. The oracle routing results based on perfectly

known future traffic patterns, mainly suffered from the same problem as predictive routing, and so they were even below reactive routing; perfect foresight can still induce unnecessary preventive rerouting under conditions of light traffic.

Implication:

The results underline a significant implication: the use of Machine Learning routing that is mainly based on technology would require not only precise predictions but also very well modified routing policies that interpret predictions in a right way. Adaptive cost functions, methods to assess prediction uncertainty, and using predictions only when confidence thresholds are met should be the main topics of future research.

Summary:

To put it briefly, the test has shown that ML models are capable of predicting but it has also shown the intricate steps that one has to go through to be able to use them for real-time routing. The above-mentioned knowledge is very useful in the research on autonomous networks and it also aids in the development of the strategies to integrate AI with traditional routing systems.

CHAPTER 6 – CONCLUSION

In this research, the use of AI/ML models, particularly for the LSTM network, in predicting the patterns of network traffic and thereby making dynamic routing decisions was investigated. The project included a complete simulation pipeline comprising of topology generation, synthetic traffic modeling, training a predictive model, and comparing the performance of different routing strategies.

The predictive model gave an impressive performance, as evidenced by the use of several statistical measures, and its forecasts were in close agreement with actual traffic trends. This, in turn, means that ML offers good predictions for proactive network management. However, the assessment made of routing performance revealed the following important fact: predictive routing did not outperform reactive routing in terms of throughput in the simulation. On the contrary, the reactive routing managed to obtain a higher throughput while predictive routing took a more cautious route that unintentionally led to lower network performance.

The result underlines the point that the quality of the prediction by itself is not an assurance of better routing performance. The performance is highly influenced by the routing cost function design, the meaning assigned to prediction errors and the general network load. In the test, the network incurred little load, implying that congestion was infrequent and predictive rerouting was mostly not required. Hence, a context-aware method has to determine the time for prediction-oriented routing to be implemented.

The experiment, although limited by these restrictions, still showed the whole process of enabling ML predictions in a routing system, thus achieving the goal of research. The resultant knowledge is the basis for more research in adaptive cost functions, uncertainty of prediction estimation, and hybrid predictive-reactive routing models.

To sum up, the research verified and at the same time announced the difficulties in AI/ML application for dynamic routing optimization; however, it also opened up a way for the coming work of developing smarter and more autonomous network systems.

References

Aouedi, O. (2025) 'Deep Learning on Network Traffic Prediction.' *ACM Digital Library.* Available at: https://dl.acm.org/doi/abs/10.1145/3703447 (Accessed: 6 November 2025).

Ateek, M. and Agrawal, N. K. (2024) 'Innovative Machine Learning Strategies for Predictive Network Management in 5G.' *International Journal of Intelligent Systems and Applications in Engineering*, 12(3). Available at: https://ijisae.org/index.php/IJISAE/article/view/7817 (Accessed: 6 November 2025).

Chen, J. et al. (2022) 'A Routing Optimization Method for Software-Defined Optical Transport Networks Based on Ensembles and Reinforcement Learning.' *Sensors*, 22(21), 8139. Available at: https://www.mdpi.com/1424-8220/22/21/8139 (Accessed: 6 November 2025).

Cisco Systems (2024) 'AI-Driven Networking: The Future of Predictive and Autonomous Networks.' *Cisco Whitepaper.* Available at: https://www.cisco.com (Accessed: 6 November 2025).

Kim, J. and Park, S. (2022) 'Predictive Network Management Using Deep Neural Networks.' *IEEE Access*, 10, pp. 21504–21515. Available at: https://ieeexplore.ieee.org/document/9722315 (Accessed: 6 November 2025).

Li, H., Zhao, T. and Singh, R. (2021) 'AI-Powered Network Routing: Challenges and Opportunities.' *Computer Networks*, 190, p. 107937. Available at: https://doi.org/10.1016/j.comnet.2021.107937 (Accessed: 6 November 2025).

Zhang, Y., Wang, L. and Liu, J. (2023) 'Machine Learning-Based Routing Optimization in Software Defined Networks.' *International Journal of Communication Systems*, 36(2), e5145. Available at: https://doi.org/10.1002/dac.5145 (Accessed: 6 November 2025).