A Proactive Performance Prediction Framework for Virtual Network Functions in 5G Networks

Milad Rafiee*, Andres F. Ocampo†, Amir Taherkordi* and Özgü Alay*‡

*Department of Informatics - University of Oslo, Oslo, Norway
†SimulaMet – Simula Metropolitan Center for Digital Engineering, Oslo, Norway
‡Computer Science Department - Karlstad University, Karlstad, Sweden

***Abstract*—Virtual Network Functions (VNFs) are essential components in modern networking that decouple network functions from dedicated hardware, thereby enhancing flexibility, scalability, and cost-efficiency. The inherently dynamic nature and fluctuating loads of networks, combined with evolving network demands, often lead to the *over-allocation* or *under-provisioning* of VNF resources, posing a significant challenge in optimal VNF resource usage. This issue becomes even more challenging when VNFs are connected in a specific sequence to fulfill a functional need—Service Function Chain (SFC). A promising solution to this challenge is proactive resource management—predicting resource demands of VNFs in advance. In this paper, we present a performance metrics prediction framework designed to anticipate multi-step resource demands for VNFs. The framework employs machine learning models, including RNN, LSTM, GRU, and Transformer models, within a robust meta-learner to capture complex usage patterns, accounting for both short-term and long-term dependencies in VNF resource consumption. This approach allows for precise prediction of critical key performance indicators (KPIs), including CPU usage, memory usage, processing latency, and traffic load for each VNF within an SFC. The evaluation results show that the proposed framework achieves a 75% reduction in mean absolute error (MAE) compared to the Transformer model and over 84% compared to RNN, LSTM, and GRU models. These results demonstrate the framework's substantial improvement over state-of-the-art approaches.**

***Index Terms*—5G Networks, Virtual Network Function (VNF), Cloud-Native Network Function (CNF), Service Function Chain (SFC), Performance Metrics Prediction**

I. Introduction

Resource management within Network Function Virtualization (NFV) refers to the efficient resource allocation and orchestration of VNFs [1]. Managing these resources is essentially challenging due to the inherent dynamic nature of the network environment and workload demands. This may lead to over-allocation or under-provisioning of VNF resources. For example, over-provisioning may cause inefficient resource utilization, while under-provisioning can lead to degraded performance and reduced service quality.

To overcome these issues, an approach is to continuously monitor and predict the resource demands of VNFs, enabling the system to proactively react and adapt to changing conditions [2]. This helps prevent bottlenecks, ensure smooth operation, and meet service level agreements (SLAs) [3]. Resource prediction becomes even more important in light of the SFC, which defines how VNFs are connected to fulfill specific functional needs. The reason is that the performance of the entire SFC is tightly coupled with the efficiency and availability of resources across all VNFs in the chain.

Previous studies have addressed VNF resource demand prediction [4]–[8], but most rely on single algorithms that capture either short- or long-term dependencies, falling short of reflecting the full complexity of real network environments. Analyzing long-term trends reveals extended load and resource fluctuations, while short-term volatility analysis, enables rapid response to immediate variations [9]. Moreover, these studies predict a subset of key KPIs, such as CPU usage, memory usage, processing latency, and traffic load. Forecasting all these KPIs is crucial for a comprehensive understanding of VNFs resource and performance requirements.

To address these issues, this paper presents a prediction framework that combines multiple machine learning algorithms to predict all the VNF KPIs mentioned above. This synergistic method would harness the strengths of different algorithms, allowing for better capture of complex patterns and dependencies within VNF resource usage data.

The framework is designed to collect KPIs from all VNFs within an SFC and predict their values for future multi-step intervals. The framework integrates two primary components: a *KPIs Collection Component* for real-time acquisition of critical metrics and a *Prediction Component* that leverages advanced machine learning techniques. The proposed prediction component utilizes ensemble learning to integrate multiple machine learning algorithms, including RNN, LSTM, GRU, and Transformer models, into a robust meta-learner.

The implementation of the proposed prediction framework demonstrates a significant improvement in prediction accuracy when compared to single models. For example, our prediction framework achieved a 75% reduction in MAE when compared to the Transformer model. Additionally, it outperforms other models, including RNN, LSTM, and GRU, with a reduction of over 84% in MAE.

This paper is organized as follows. Section II reviews related work on VNF resources prediction. Section III presents the system model. Section IV introduces the proposed prediction framework, and Section V outlines the experimental setup. Section VI discusses the evaluation results. Finally, Section VII concludes the paper and highlights future directions.

II. Related Work

Extensive research, summarized in Table I has been dedicated to addressing the resource prediction challenges for

978-3-903176-75-1

VNFs. Works focusing on short-term dependencies include [4], [6] and [7]. Luo et al. [4] proposed a framework built on deep learning to enhance the scalability of geo-distributed VNF chains. They employed a traffic model based on a RNN to predict upcoming flow rates, and a DRL agent to determine the optimal placement of VNFs in a chain. Mijubi et al. [6] presented a graph neural network (GNN) based algorithm that uses VNF forwarding graph topology information to predict future resource requirements for each VNF component (VNFC). The topological information of every VNFC is obtained by aggregating its historical resource usage with the predicted impact on it from neighboring VNFCs.

Another work focusing on short-term dependencies is presented in [7], where a deep learning model combining GNNs with multi-task learning (MTL) is used to predict VNF resource demands. The process involves offline and online learning. Offline learning computes initial neural network weights for input, intermediate, and output layers. Online learning then predicts VNF delay, CPU, and storage requirements. Pandey et al. [8] by considering long-term dependencies proposed an automated scaling and placement method for SFCs using GRU. The autoscaling module sets up the SFC for incoming requests by predicting the VNFs KPIs. Similarly, Wu et al. [5] captured long-term dependencies by implementing a multilayer perceptron (MLP) network, where the prediction module consists of an input layer, multiple hidden layers, and an output layer to predict traffic load of VNF.

TABLE I: Related Works

Work	Prediction KPI					Dependencies	
	CPU	Memory	Traffic	Precessing	Latency	Short	Long
Luo et al. [4]			✓			✓	
Mijumbi et al. [6]	✓	✓				✓	
Zhang et al. [7]	✓	✓		✓		✓	
Pandey et al. [8]	✓	✓	✓				✓
Wu et al. [5]			✓				✓
This paper	✓	✓	✓	✓		✓	✓

Existing research relies on single machine learning algorithms, capturing either short-term or long-term data dependencies, limiting their ability to address real-world network complexity. These studies typically predict a subset of KPIs, such as CPU usage, memory usage, traffic load, and processing latency. Comprehensive prediction of all KPIs is essential for understanding VNF resource and performance requirements.

This study presents a prediction framework for VNFs in an SFC, forecasting CPU usage, memory usage, traffic load, and processing latency. It integrates short-term and long-term temporal features to capture immediate variations and extended trends in time series data, enabling responsive scaling and proactive resource planning. It can further support resource and service orchestration [10] in edge and fog networks.

III. System Model

This section provides an overview of the system model, depicted in Figure 1, which is used to evaluate the proposed performance metrics prediction framework for VNF resource management in 5G networks. The *Data Generation Layer* comprises a distributed network of software IoT devices communicating via software gateways, emulating large-scale IoT ecosystems with thousands of interconnected devices [11]. Sensors send messages to their gateways, which buffer and batch them by size to reduce transmission overhead before forwarding to the VNFs for processing.

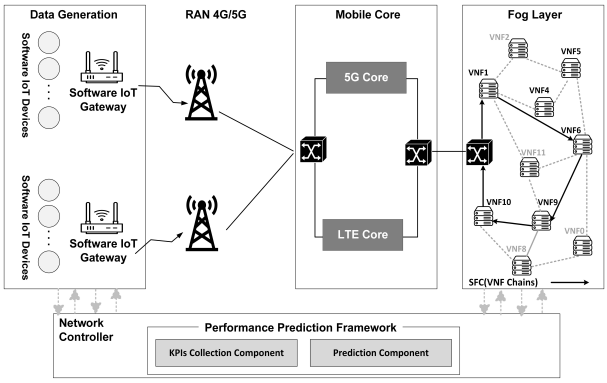


Fig. 1: System Model

The Radio Access Network (RAN) facilitates the transmission of data from IoT gateways to the Mobile Core over a wireless communication network. Depending on the deployment requirements, the network supports both LTE (4G) and 5G technologies, ensuring compatibility with diverse IoT environments. The Mobile Core then routes the data to the Fog Layer, where computational tasks and service function execution take place.

In the Fog Layer, the data is processed through VNFs, which are organized into an SFC. The SFC can be tailored to apply various VNFs based on specific use-case requirements as shown in the Figure 1. In this scenario, the SFC comprises a sequence of security and optimization functions designed to meet application-specific demands, such as Firewall, Deep Packet Inspection(DPI), data encryption, and traffic compression. In this SFC, the sequence ensures that security is applied at multiple layers, FW for basic protection and DPI for advanced inspection. Data confidentiality is maintained through Enc and network efficiency is optimized by compressing the data. The Network Controller depicted in the system model is a central component that acts as a decision-making entity, interacting with various layers to ensure efficient utilization of available resources. Based on its responsibility for dynamically scaling the resources of VNFs, the Network Controller leverages the proposed prediction framework to enhance the efficiency of this process.

The proposed prediction framework consists of two main components: the KPIs Collection Component and the Prediction Component. The KPIs Collection Component monitors VNFs and gathers several KPIs for each VNF. The Prediction Component leverages machine learning algorithms to predict these KPIs for subsequent time steps of each VNF.

IV. Performance Prediction Framework

The proposed prediction framework comprises a *KPIs Collection Component* and a *Prediction Component*, detailed below for their roles within the framework.

A. KPIs Collection Component

The framework's efficacy depends on the accuracy of its predictive model, which is driven by the quality and timeliness of training data. By continuously collecting up-to-date VNF resource and traffic telemetry for retraining, it adapts to evolving network conditions and maintains effectiveness in real deployments. The KPIs Collection Component comprises modules that gather specific VNF performance metrics (cf. Figure 2). These modules continuously monitor VNFs and store the data in a centralized database for predictive analysis.

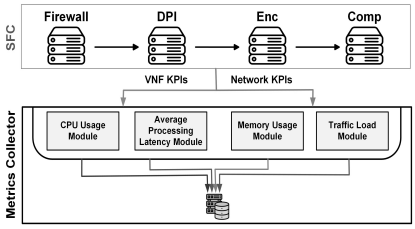


Fig. 2: KPIs Collection Component

The collected metrics are categorized into two groups: VNF-specific metrics and network-related metrics.

VNF-specific metrics: These metrics focus on the computational and memory resource consumption of VNFs. The following are the key metrics in this category:

- **CPU Usage:** Monitoring and predicting CPU usage allows for proactive scaling of resources, preventing over or under utilization while maintaining optimal performance.
- **Memory Usage:** Like CPU usage, accurate predictions of memory usage ensure efficient resource allocation, avoiding potential shortages or excessive provisioning.
- **Average Processing Latency:** This metric is a critical indicator of system responsiveness and efficiency, helping to minimize delays by dynamically adjusting resources.

Network Metrics: These metrics are tied to network performance and resource demands:

- **Traffic Load:** Measures the total amount of data transmitted through the VNF, typically in megabits per second (Mbps). Predicting traffic load provides insights into resource requirements, allowing the framework to prepare for fluctuations and maintain uninterrupted performance.

B. Prediction Component

The proposed Prediction Component uses ensemble learning to integrate multiple machine learning algorithms into a robust meta-learner, enhancing prediction accuracy by combining the strengths of diverse models. Ensemble methods are widely recognized for their ability to improve overall performance by aggregating the predictions of individual models [12].

A notable technique employed is stacked ensemble learning, which combines the output from multiple base models to improve predictive performance. This process begins with training various base models on the same dataset using cross-validation. The base models, which may utilize different underlying algorithms, are trained independently [13].

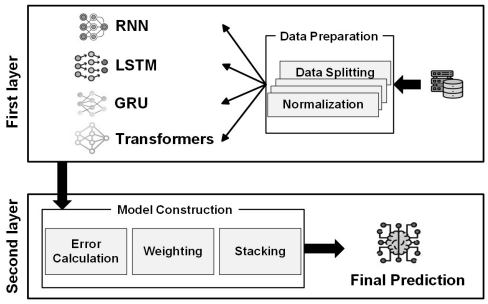


Fig. 3: Prediction Component

Figure 3 illustrates the hierarchical Prediction Component, designed to estimate critical performance metrics for VNFs in an SFC. The Prediction Component consists of two layers that work together to ensure accurate and reliable predictions for resource management. The first layer focuses on the preparation of the data and the generation of preliminary predictions using advanced neural network architectures. To ensure consistency, the input data undergoes a normalization process that scales the feature values to a standard range between 0 and 1 using the min-max normalization method. In addition to normalization, the data preparation phase employs a sliding window approach to generate input-output pairs. Using a fixed number of past observations as input to predict a sequence of future values, the system captures temporal dependencies in the data.

The prepared data is then processed by four distinct neural network architectures, each chosen for its ability to handle sequential data and capture varying dependencies. These neural networks are trained independently, producing separate prediction models for collected metrics across VNFs.

Recurrent Neural Networks (RNNs) serve as a baseline model for identifying short-term dependencies, while Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRUs) are utilized for their efficiency in capturing long-term dependencies, with GRUs offering a more computationally efficient alternative. Transformers complement these models by employing self-attention mechanisms to identify global patterns and complex dependencies within the data. Leveraging the unique capabilities of RNN, LSTM, GRU, and Transformer models, which offer diverse architectures and the ability to capture both short-term and long-term dependencies, the first layer generates accurate predictions for key metrics while enhancing overall prediction accuracy.

The second layer integrates the outputs of the base models to produce a unified prediction. During this phase, the MAE of each base model is calculated and the weights are assigned inversely proportional to the MAE, ensuring that models with lower errors have a greater influence on the final prediction. A stacking approach is then employed, where the weighted outputs are combined and fed into a higher-level meta-model trained to synthesize these predictions into a single, accurate predict. Finally, the stacked ensemble model combines the outputs of the base models, trained on the entire dataset, using a meta-model that is specifically trained to integrate

these outputs into a unified and accurate final prediction. This approach generates precise forecasts for all key metrics across VNFs in the SFC.

V. Experimental Setup

This section presents the experimental setup using a real testbed with real data to emulate dynamic network environments. As shown in Figure 4, the testbed consists of an SFC deployed on four fog servers, each running a Kubernetes-orchestrated VNF. The setup also incorporates software IoT devices and gateways. The testbed emulates 4G and 5G network behaviors using data from a real-world dataset collected from monitoring the performance of the main Mobile Network Operator (MNO) in Norway.

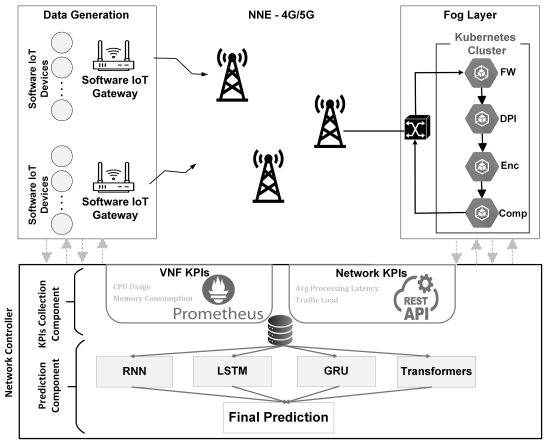


Fig. 4: Experimental Testbed

A. Data Generation

The performance evaluation was based on four IoT deployment scenarios small, medium, large, and very large varying in device types, data volumes, and key parameters such as number of IoT devices, IoT gateway, and sensor rate, to accurately model diverse real-world environments.

Our experiments use Software IoT Devices and Gateways to emulate real hardware. Devices generate sensor data, while gateways aggregate and forward it to the SFC. This software-based approach enables scalable, flexible testing of diverse IoT scenarios without physical hardware [14].

The scenarios in Table II reflect real-world IoT use cases. The small scenario models a residential setting with one gateway and 1,000 devices (e.g., appliances, security, sensors) [15]. The medium scenario reflects commercial deployments with 10 gateways and 10,000 devices. This setup models retail stores, offices, or public spaces, where IoT solutions are used for inventory management, customer analytics, and facility optimization [16]. The large scenario simulates industrial environments with 25 gateways and 25,000 devices in factories or logistics hubs [17]. The very large scenario, representing a smart city deployment, scales up to 50 gateways and 50,000 devices, incorporating IoT applications for urban planning, transportation management, and environmental monitoring.

IoT sensor data streams were modeled as JSON-encoded measurements containing fields such as sensor ID, name, type, unit, value, and timestamp. Each sensor generates data at predefined rate ranging from 4 to 50 messages per second. These configurations align with those used in prior research [11], designed to simulate diverse IoT environments with progressively increasing volumes of sensor measurements.

TABLE II: Summary of the IoT experimental settings

Parameter	Scenario			
	Small	**Medium**	**Large**	**Very Large**
Software IoT Gateway (g)	1	10	25	50
Software IoT Devices (d)	1000	10000	25000	50000
Sensor Rate (s_r)	4, 12, 25, 50			
Data batch size (b_s)	4000			

B. Data Transmission Emulation Using NNE Dataset

To ensure the experiments reflect realistic network conditions, we utilized a specific dataset derived from the NortNet-Edge (NNE) measurement platform, deployed across Norway [18]. Collected between November 2023 and February 2024, this dataset provides detailed performance metrics from 4G and 5G networks.

The dataset focuses on bandwidth-related KPIs, specifically downlink and uplink data rates aggregated from speed tests to represent the available bandwidth. In our experiments, the traffic rate—which determines the volume of data transmitted to the mobile network over time—was configured based on the upload speed metrics from the dataset. These metrics were instrumental in shaping the test traffic to emulate realistic network conditions, ensuring the experiments accurately mirrored the dynamic behaviors of 4G and 5G environments.

This dataset provides a robust basis for evaluating resource prediction and placement strategies, as it captures diverse network conditions and KPIs observed in real-world deployments.

C. Service Function Chaining

In this setup, the network functions (NFs) are implemented as CNFs, leveraging containerized applications for enhanced flexibility, scalability, and portability. Each network function is deployed as a container that encapsulates its software and dependencies. The orchestration and lifecycle management of CNFs in the SFC are handled by Kubernetes. The VNFs are deployed across a cluster of four servers, each hosting one container. Collectively, these servers operate as a fog computing environment.

The testbed includes four specific VNFs, each performing a distinct role within the SFC. The firewall VNF is implemented as a module that inspects the headers of incoming packets by sniffing input traffic at the network interface. This VNF uses a JSON-based configuration file to define blocking rules based on the 5-tuple information of packets.The DPI VNF, implemented using Snort3 (version 3.1.43.0) [19], examines the payloads of packets traversing the network. This VNF enhances security by analyzing network traffic for malicious content, policy violations, and abnormal patterns.

The encryption VNF utilizes Advanced Encryption Standard – Galois/Counter Mode (AES-GCM) to secure network traffic [20]. AES-GCM provides both encryption and authentication, ensuring that the confidentiality and integrity of packet contents are preserved. The compression VNF reduces the size of

transmitted data using the Brotli compression algorithm [21]. After compression, the data is encoded in base64 format to ensure compatibility with text-based transmission protocols.

Each VNF operates as an individual server, within the testbed, with its application containerized using Docker. This configuration supports a flexible and efficient evaluation of the framework in a cloud-native 5G environment.

D. Metrics collector

The KPIs of the VNFs were collected using Prometheus, a widely adopted open-source monitoring tool. The component queries each VNF at regular intervals, retrieving their KPIs and storing them in a centralized database for further analysis. For network KPIs, a scalable Representational State Transfer (REST) API was used, providing a standardized interface for nodes to transmit their KPIs. Each VNF periodically sends its KPIs to a server listening for incoming requests on predefined endpoints. These metrics are then stored in the same centralized database.

The experimental setup was run continuously for approximately 12 hours, with each scenario lasting 15 minutes. During this period, KPIs were collected at one-second intervals, generating a comprehensive dataset. This dataset was subsequently divided into two subsets: 80% for training the models and 20% for testing and evaluation purposes.

VI. Evaluation Results

This section evaluates the proposed prediction framework, focusing on computational efficiency, accuracy, error analysis, and multi-step prediction. Its performance is benchmarked against baseline models: RNN, LSTM, GRU, and Transformer.

A. Evaluation Metrics

To evaluate the accuracy and robustness of the proposed framework, we use MAE, Mean Squared Error (MSE), and Root Mean Squared Error (RMSE) metrics. These metrics assess different aspects of prediction performance. Together, these metrics provide a balanced evaluation of predictive accuracy, with MAE highlighting average errors, and MSE and RMSE emphasizing sensitivity to large deviations.

B. Computational Complexity

The time complexity of the proposed framework and baseline models was evaluated via their training and testing times (Table III). The RNN trains in 220.82 s and tests in 2.31 s, while LSTM and GRU take longer to train (529.83 s and 581.30 s, respectively) but GRU tests faster. The Transformer is the most resource-intensive, requiring 9.264 s to train and 7.39 s to test due to its attention mechanisms.

In the proposed framework, base models are trained independently, with the meta-learner adding 520.78 s to fit on their concatenated outputs. During inference, base models run in parallel, and the meta-learner then combines their outputs. The inference time for the entire test set is 7.74 s. The proposed framework adds a mere 0.35 s to the slowest model's latency while delivering a level of robustness and accuracy far beyond any single model.

TABLE III: The execution time during training and testing phases (in seconds)

Model	Training Time (sec)	Testing Time (sec)
RNN	220.82	2.31
LSTM	529.83	5.39
GRU	581.30	2.89
Transformer	9264.06	7.39
Meta-learner	520.78	0.35

C. Prediction Accuracy and Error Analysis

The accuracy of the proposed framework and error characteristics are evaluated using two distinct analyses: residual error range and MAE. These analyses highlight the framework's ability to deliver accurate predictions and minimize deviations across various models.

1) Residual Error Analysis: Figure 5 provides a comparative visualization of residual errors across RNN, LSTM, GRU, Transformer, and the proposed framework. The residual range for the proposed framework is significantly smaller (-0.1 to 0.3) compared to other models like RNN and LSTM (-0.9 to 0.9). This 77.8% reduction in the error range demonstrates the framework's improved prediction accuracy and reduced variance. Furthermore, the median residual error for the proposed framework is near zero, emphasizing its robustness in minimizing prediction deviations.

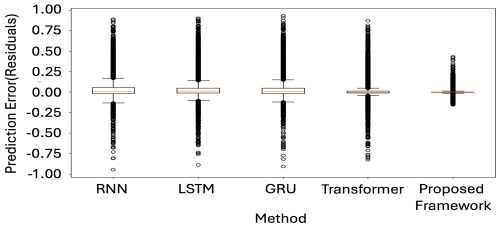


Fig. 5: Residual Analysis of VNF Resource Demand Prediction Methods

2) MAE Comparison: Table IV shows the MAE for different methods in predicting the KPIs of VNFs. The proposed framework achieves an MAE of 0.0081, representing a 75% reduction compared to the Transformer model (0.0318) and over an 84% reduction relative to RNN (0.0527), LSTM (0.0494), and GRU (0.0483). These results showcase the capability of the framework to produce predictions with substantially lower average errors than baseline models, further emphasizing its superiority. Together, these analyses highlight the proposed framework's strengths in delivering highly accurate predictions with minimal residual errors and average deviations.

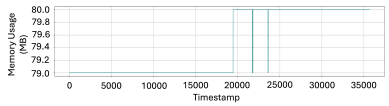
TABLE IV: Mean Absolute Error (MAE) Across Prediction Methods

Model	Mean Absolute Error (MAE)
RNN	0.0527
LSTM	0.0494
GRU	0.0483
Transformer	0.0318
Proposed Framework	0.0081

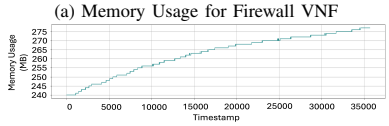
D. Prediction Accuracy Across KPIs

This section evaluates the prediction accuracy of the proposed framework across various KPIs, including memory usage, CPU usage, traffic load, and processing latency. The performance of the framework is benchmarked against traditional models such as RNN, LSTM, GRU, and Transformer, highlighting its superior accuracy and efficiency.

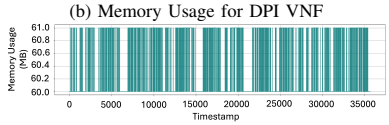
1) Memory Usage Prediction: Figure 6 shows the memory usage patterns of three VNFs: Firewall, DPI, and Compression. The memory usage for the Firewall was recorded between 79 MB and 80 MB, while the Enc and Comp exhibited stable consumption levels at 60 MB and between 60 MB and 61 MB, respectively. These predictable and stable patterns of memory usage rendered prediction unnecessary for these VNFs.



(a) Memory Usage for Firewall VNF



(b) Memory Usage for DPI VNF



(c) Memory Usage for Compression VNF

Fig. 6: Memory Usage Across Different VNFs

In contrast, the DPI VNF presented a more dynamic memory usage profile, warranting prediction analysis. The results of this analysis, detailed in Table V, reveal that the proposed prediction framework significantly outperforms traditional models across all error metrics. Specifically, the framework achieved reductions of 99.16% in MAE, 99.99% in MSE, and 97.60% in RMSE compared to baseline models. These outcomes highlight the framework's capability to accurately predict memory demands for the DPI VNF, underscoring its potential to enhance VNF resource allocation efficiency and minimize the risk of service disruptions caused by resource constraints.

TABLE V: Memory Usage Performance Metrics (DPI)

Model	MAE	MSE	RMSE
RNN	0.027537	0.001732	0.041621
LSTM	0.026278	0.002083	0.045640
GRU	0.025049	0.001204	0.034698
Transformer	0.002731	0.000098	0.009914
Proposed Framework	0.000233	0.000001	0.001081

2) CPU Usage Prediction: The proposed framework demonstrates remarkable accuracy in CPU usage prediction, achieving superior results for the four VNFs considered in our evaluation: Firewall, DPI, Enc, and Compression. Table VI presents the performance metrics—MAE, MSE, and RMSE—for CPU usage prediction across these VNFs.

The proposed framework consistently exhibits the lowest error rates among all models, highlighting its ability to accurately predict CPU demands. For instance, in the Firewall VNF, the framework achieves an MAE of 0.001570, reflecting an 80.4% reduction compared to the Transformer model, the best-performing baseline. Similar improvements are observed for the other VNFs, further underscoring the framework's robustness in CPU usage prediction.

3) Traffic Load Prediction: Accurate traffic load prediction is essential for optimizing VNF performance and reducing network congestion. Table VII shows that the proposed framework outperforms baseline models across all error metrics—MAE, MSE, and RMSE. For the Enc VNF, the framework achieves remarkable improvements in RMSE, reducing the error by 37.0% compared to the LSTM model, 45.0% compared to the RNN, 39.4% compared to the GRU, and 31.8% relative to the Transformer model.

4) Processing Latency Prediction: Processing latency is vital for evaluating VNF responsiveness in dynamic networks. Accurate prediction supports proactive resource management and adherence to service level agreements. Table VIII presents the processing latency prediction errors across the four VNFs. The proposed framework demonstrates exceptional performance in this task, achieving an MAE of 0.022 a significant 61.5% reduction compared to the best baseline model.

E. Multi-Step Prediction

The framework's robustness was also evaluated via multi-step prediction, estimating each VNF's resource demands over the next τ time slots using the last σ observed values.

Multi-step predictions were performed for horizons of 2, 15, 30, and 60 seconds, as shown in Table IX. For the shorter horizons (2, 15, and 30 seconds), the most recent 120 time slots were utilized as inputs, while the 60-second forecast relied on the last 240 time slots to enhance accuracy.

The MAE results in Table IX highlight the superior performance of the proposed framework across all time horizons. At shorter intervals, such as 2 and 15 steps, the framework achieved significantly lower MAE values of 0.007503 and 0.006495, respectively, outperforming the closest competitor (Transformer) by a considerable margin. This trend continued for the 30-step forecast, where the MAE remained exceptionally low at 0.006255. Even for the longest horizon (60 steps), the proposed framework demonstrated remarkable stability, with an MAE of 0.007324.

VII. Conclusion

This research introduces a performance metrics prediction framework designed to monitor and predict resource demands in VNFs, addressing the critical challenge of efficient resource allocation in next-generation telecommunications infrastructure. The framework consists of two main components: the KPI Collection Component and the Prediction Component. The KPI Collection Component facilitates real-time monitoring and acquisition of critical VNFs metrics and network performance data. The Prediction Component employs a robust meta-learning approach that integrates multiple neural network architectures to capture both short-term and long-term dependencies in the resource usage of each VNF in an SFC. Experimental evaluations demonstrate substantial reductions in prediction errors across multiple performance indicators, including CPU usage, memory usage, processing latency, and traffic load. Compared to traditional machine learning models such as RNN, LSTM, GRU, and Transformer, the proposed approach achieves remarkable improvements, with up to a 75% reduction in MAE relative to the Transformer model.

TABLE VI: CPU Usage Prediction Error for Different Models

Model	Firewall			DPI			Encryption			Compression		
	MAE	MSE	RMSE	MAE	MSE	RMSE	MAE	MSE	RMSE	MAE	MSE	RMSE
RNN	0.023654	0.001679	0.040970	0.023551	0.002038	0.045146	0.027241	0.001828	0.042753	0.034698	0.002982	0.054612
LSTM	0.031628	0.002593	0.050918	0.028540	0.003350	0.057882	0.025329	0.002333	0.048303	0.026903	0.002940	0.054221
GRU	0.023396	0.001641	0.040504	0.020647	0.002055	0.045334	0.021181	0.001532	0.039139	0.023189	0.001947	0.0441247
Transformer	0.008015	0.000786	0.028038	0.005707	0.000723	0.026890	0.004979	0.000596	0.024411	0.007956	0.000849	0.029139
This Work	0.001570	0.000009	0.003044	0.001701	0.000011	0.003349	0.002015	0.000016	0.004017	0.001721	0.000011	0.003355

TABLE VII: Traffic Load Prediction Error for Different Models

Model	Firewall			DPI			Encryption			Compression		
	MAE	MSE	RMSE	MAE	MSE	RMSE	MAE	MSE	RMSE	MAE	MSE	RMSE
RNN	0.053706	0.005785	0.076058	0.041144	0.003934	0.062723	0.067434	0.008319	0.091207	0.053743	0.005803	0.076175
LSTM	0.038630	0.003623	0.060193	0.039218	0.004099	0.064020	0.045058	0.004702	0.068569	0.037764	0.003650	0.060411
GRU	0.037530	0.003628	0.060233	0.038614	0.003847	0.062021	0.041197	0.004181	0.064661	0.039727	0.003508	0.059232
Transformer	0.031754	0.002811	0.053020	0.033549	0.002837	0.053262	0.031504	0.002829	0.053185	0.031449	0.002823	0.053127
This Work	0.010401	0.000241	0.015519	0.011533	0.000297	0.017221	0.010826	0.000261	0.016149	0.011174	0.000278	0.016670

TABLE VIII: Processing Latency Prediction Error for Different Models

Model	Firewall			DPI			Encryption			Compression		
	MAE	MSE	RMSE	MAE	MSE	RMSE	MAE	MSE	RMSE	MAE	MSE	RMSE
RNN	0.073391	0.011722	0.108268	0.115501	0.022741	0.150802	0.082568	0.010147	0.100732	0.060672	0.005747	0.075811
LSTM	0.087132	0.015695	0.125278	0.140657	0.030257	0.173946	0.067524	0.007742	0.087988	0.048014	0.003507	0.059223
GRU	0.088189	0.015462	0.124346	0.131733	0.027020	0.164377	0.083070	0.010052	0.100258	0.054391	0.003886	0.062338
Transformer	0.057184	0.008483	0.092104	0.085373	0.014402	0.120010	0.062943	0.006674	0.081695	0.050608	0.003773	0.061423
This Work	0.021988	0.000942	0.030698	0.025556	0.001277	0.035728	0.007065	0.000108	0.010387	0.002461	0.000012	0.003480

TABLE IX: MAE for Multi-Step Prediction

Model	2 Steps	15 Steps	30 Steps	60 Steps
RNN	0.054061	0.054736	0.057387	0.051665
LSTM	0.052174	0.056408	0.058931	0.059402
GRU	0.048055	0.052483	0.055795	0.059153
Transformer	0.034303	0.044075	0.047421	0.051731
Proposed Framework	0.007503	0.006495	0.006255	0.007324

Future work will explore applying the proposed framework to complex networks and varying traffic patterns, and extend it to dynamically scaling VNF resources within an SFC for efficient and adaptive resource allocation based on real-time traffic demands and network conditions.

ACKNOWLEDGEMENT

This work was supported by the Norwegian Research Council under Grant 322473 (AirQMan project).

REFERENCES

[1] R. Mijumbi, J. Serrat, and et al, "Network function virtualization: State-of-the-art and research challenges," *IEEE Communications surveys & tutorials*, vol. 18, no. 1, pp. 236–262, 2015.

[2] A. Sarah, G. Nencioni, and M. M. I. Khan, "Resource allocation in multi-access edge computing for 5g-and-beyond networks," *Computer Networks*, vol. 227, p. 109720, 2023.

[3] V. R. Chintapalli, M. Adeppady, B. R. Tamma, *et al.*, "Restrain: A dynamic and cost-efficient resource management scheme for addressing performance interference in nfv-based systems," *Journal of Network and Computer Applications*, vol. 201, p. 103312, 2022.

[4] Z. Luo, Z. Li, and W. Zhou, "Scaling geo-distributed network function chains: A prediction and learning framework," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 8, pp. 1838–1850, 2019.

[5] Y. Wu, J. Liu, C. Wang, and Q. Yang, "Graph attention lstm for load prediction of fine-grained vnfs in sfc," in *ICC 2023-IEEE International Conference on Communications*, pp. 4899–4904, IEEE, 2023.

[6] R. Mijumbi, S. Hasija, S. Davy, A. Davy, B. Jennings, and R. Boutaba, "Topology-aware prediction of virtual network function resource requirements," *IEEE Transactions on Network and Service Management*, vol. 14, no. 1, pp. 106–120, 2017.

[7] J. Zhang, Y. Liu, Z. Li, and Y. Lu, "Forecast-assisted service function chain dynamic deployment for sdn/nfv-enabled cloud management systems," *IEEE Systems Journal*, 2023.

[8] S. Pandey, M. Choi, J.-H. Yoo, and J. W.-K. Hong, "Rnn-edgeql: An auto-scaling and placement approach for sfc," *International Journal of Network Management*, vol. 33, no. 4, p. e2213, 2023.

[9] M. Ye, J. Luo, C. Xiao, and F. Ma, "Lsan: Modeling long-term dependencies and short-term correlations with hierarchical attention for risk prediction," in *Proceedings of the 29th ACM international conference on information & knowledge management*, pp. 1753–1762, 2020.

[10] M. Rafiee, A. Taherkordi, and Ö. Alay, "Cross network layer cognitive service orchestration in edge computing systems," in *2024 IEEE International Conference on Edge Computing and Communications (EDGE)*, pp. 12–21, IEEE, 2024.

[11] F. Tusa, S. Clayman, A. Buzachis, and M. Fazio, "Microservices and serverless functions—lifecycle, performance, and resource utilisation of edge based real-time iot analytics," *Future Generation Computer Systems*, vol. 155, pp. 204–218, 2024.

[12] M. A. Ganaie, M. Hu, A. K. Malik, M. Tanveer, and P. N. Suganthan, "Ensemble deep learning: A review," *Engineering Applications of Artificial Intelligence*, vol. 115, p. 105151, 2022.

[13] Z.-H. Zhou, *Ensemble methods: foundations and algorithms*. CRC press, 2012.

[14] M. Salama, Y. Elkhatib, and G. Blair, "Iotnetsim: A modelling and simulation platform for end-to-end iot services and networking," in *Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing*, (New York, NY, USA), p. 251–261, 2019.

[15] J. Bokstaller, J. Schneider, and J. vom Brocke, "Estimating soc, soh, or rul of rechargeable batteries via iot: A review," *IEEE Internet of Things Journal*, 2023.

[16] P. Suresh, J. V. Daniel, V. Parthasarathy, and R. Aswathy, "A state of the art review on the internet of things (iot) history, technology and fields of deployment," in *2014 International conference on science engineering and management research (ICSEMR)*, pp. 1–8, IEEE, 2014.

[17] M. H. ur Rehman, I. Yaqoob, K. Salah, M. Imran, P. P. Jayaraman, and C. Perera, "The role of big data analytics in industrial internet of things," *Future Generation Computer Systems*, vol. 99, pp. 247–259, 2019.

[18] A. F. Ocampo and J. P. Pereira dos Santos, "Reinforcement learning-driven service placement in 6g networks across the compute continuum," in *CNSM2024, the 20th International Conference on Network and Service Management*, 2024.

[19] M. Roesch, "Snort - lightweight intrusion detection for networks," in *Proceedings of the 13th USENIX Conference on System Administration (LISA)*, 1999.

[20] M. Dworkin, "Recommendation for block cipher modes of operation: Galois/counter mode (gcm) and gmac," Tech. Rep. SP 800-38D, National Institute of Standards and Technology (NIST), 2007.

[21] J. Alakuijala and Z. Szabadka, "Brotli compressed data format," 2016.