Received 3 June 2022, accepted 9 July 2022, date of publication 13 July 2022, date of current version 18 July 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3190640

RESEARCH ARTICLE

Dynamic Network Slice Scaling Assisted by Attention-Based Prediction in 5G Core Network

CHIEN-NGUYEN NHU[1] **AND MINHO PARK**[2], **(Member, IEEE)**
[1]Department of Information Communication Convergence Technology, Soongsil University, Seoul 156-743, South Korea
[2]School of Electronic Engineering, Soongsil University, Seoul 156-743, South Korea

Corresponding author: Minho Park (mhp@ssu.ac.kr)

This work was supported by the National Research Foundation of Korea (NRF) funded by the Korean Government [Ministry of Science and ICT (MSIT)] under Grant 2020R1F1A1076795.

ABSTRACT Network slicing is a key technology in fifth-generation (5G) networks that allows network operators to create multiple logical networks over a shared physical infrastructure to meet the requirements of diverse use cases. Among core functions to implement network slicing, resource management and scaling are difficult challenges. Network operators must ensure the Service Level Agreement (SLA) requirements for latency, bandwidth, resources, etc for each network slice while utilizing the limited resources efficiently, i.e., optimal resource assignment and dynamic resource scaling for each network slice. Existing resource scaling approaches can be classified into reactive and proactive types. The former makes a resource scaling decision when the resource usage of virtual network functions (VNFs) exceeds a predefined threshold, and the latter forecasts the future resource usage of VNFs in network slices by utilizing classical statistical models or deep learning models. However, both have a trade-off between assurance and efficiency. For instance, the lower threshold in the reactive approach or more marginal prediction in the proactive approach can meet the requirements more certainly, but it may cause unnecessary resource wastage. To overcome the trade-off, we first propose a novel and efficient proactive resource forecasting algorithm. The proposed algorithm introduces an attention-based encoder-decoder model for multivariate time series forecasting to achieve high short-term and long-term prediction accuracies. It helps network slices be scaled up and down effectively and reduces the costs of SLA violations and resource overprovisioning. Using the attention mechanism, the model attends to every hidden state of the sequential input at every time step to select the most important time steps affecting the prediction results. We also designed an automated resource configuration mechanism responsible for monitoring resources and automatically adding or removing VNF instances of network slices, which helps network operators satisfy service requirements even when the traffic of end-user requests changes dynamically. Comprehensive experiments demonstrate that our proposed solution outperforms other solutions in terms of short-term and long-term predictions while reducing the cost of SLA violations and resource overprovisioning and enhancing the delay quality of network slices.

INDEX TERMS Network slicing, auto scaling, resource prediction, deep learning, attention-based encoder-decoder.

I. INTRODUCTION

Fifth-generation (5G) networks currently cover a wide range of industrial applications such as automotive, energy, food, healthcare, so on and significantly improve network performance [1]. A wide range of industrial use cases with diverse requirements require 5G networks to be flexible, scalable, manageable, customized, and allow multi-tenancy and multi-service support [2]. Network slicing has been proposed as an essential feature of 5G networks to slice a common underlying physical network into multiple end-to-end (E2E) logical networks that are isolated, managed independently, and created on demand. A network slice is a collection of resources and network functions that are orchestrated to support a

The associate editor coordinating the review of this manuscript and approving it for publication was Tiago Cruz.

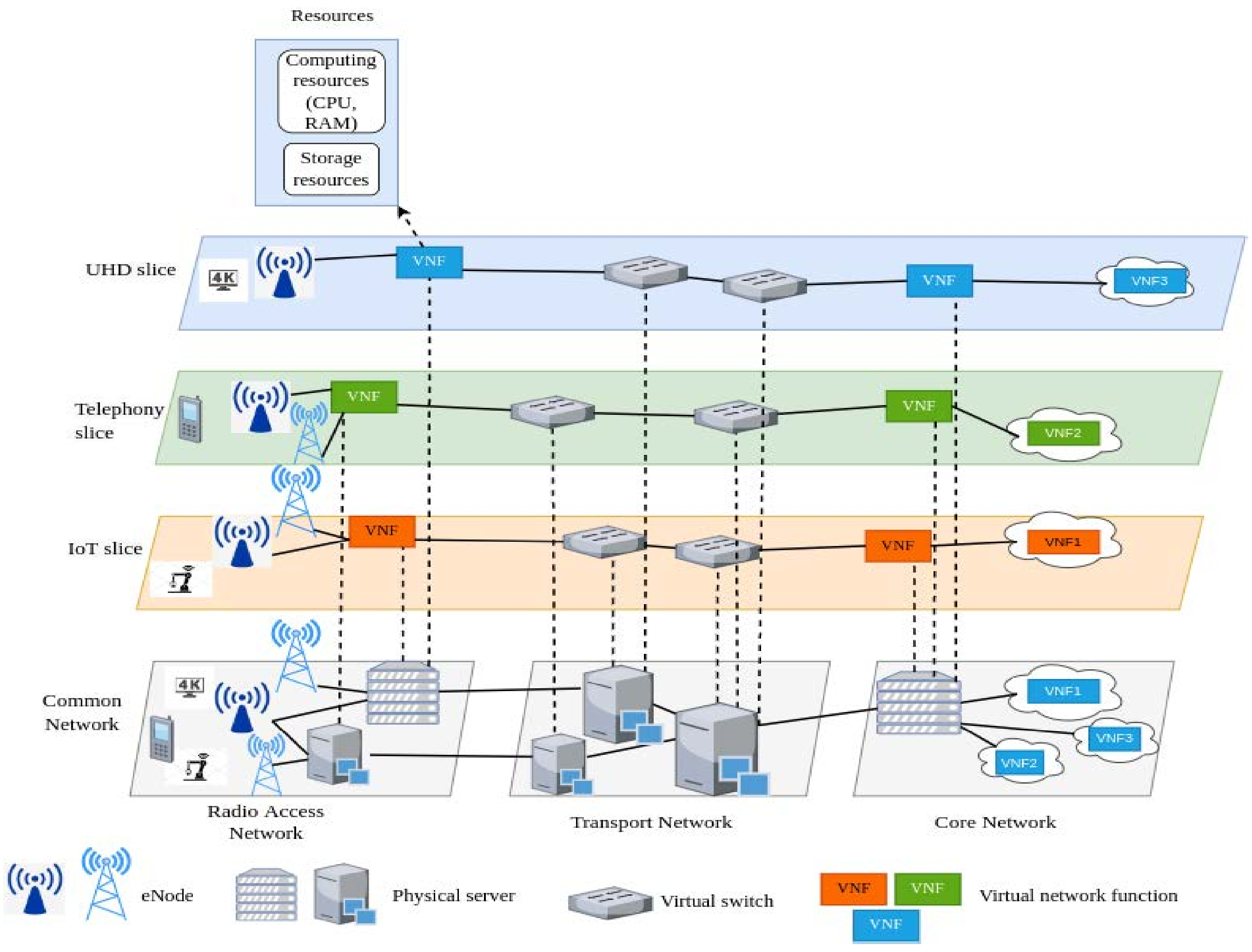


FIGURE 1. Overview of 5G network slices.

specific service [3]. The fundamental technologies to implement network slicing are network function virtualization (NFV) virtualizing network infrastructure with virtual network components and software-defined networking (SDN) enabling network automation and programmability by decoupling network control and forwarding functions [4]. Using these key technologies, network slicing divides a network into multiple virtual networks consisting of virtual network functions (VNFs) and deploy them on virtual machines (VMs) or virtual containers (e.g docker) to reduce cost and energy consumption. Each VNF can be divided into sub-functions called VNF components [5]. In this paper, for simplicity, we assume that each VNF comprises a single VNF component. Figure 1 depicts the concept of E2E network slicing in real applications.

Although network slicing technology can provide better performance and more efficient networks, it confronts a significant challenge in resource management and scaling [6]. Network operators have to ensure that the Service Level Agreement (SLA) requirements for latency, bandwidth, resource, etc for each network slice are guaranteed despite the variability of the end-user requests arrivals in each slice. A network slice must be scaled up and down according to the number of end-user requests, i.e., increasing or decreasing the number of instances for each VNF when the end-user requests increase or decrease.

Most existing resource assignment schemes are reactive and based on hysteresis thresholding [7]. They define a hard threshold for resource metrics of VMs or containers which VNFs are running on, and scale up or down the number of instances for each VNF when these resource metrics go beyond or below the threshold. However, resource instantiation or scaling is time-consuming: VM boot time in public clouds such as Amazon AWS or Microsoft Azure consistently exceeds 40 seconds, reaching 400 seconds in worst-case scenarios and modern software architectures such as Kubernetes need several seconds to execute new pods. These unavoidable time delays affect the quality of network slices using a legacy reactive approach to scale or manage resources. This unavoidable delay may result in SLA violation costs because the latency or resources requirements for the tenants are not guaranteed.

A few researchers have proposed proactive approaches using classical and statistical time series forecast algorithms such as Holt-winter [8] or autoregressive integrated moving average (ARIMA) model [9] and deep learning models. These proactive, data-driven and automated solutions forecast the future needs of network slices and reallocate resources in a timely manner depending on where and when scaling is needed. The more accurate and the longer the prediction of the forecast mechanism, the lower the cost due to SLA violation or resource waste from overprovisioning. However, no existing solutions can achieve sufficiently good results in resource forecasting for network slices for the following reasons:

- Most existing approaches focus on short-term or single-step forecasting, which is not practical for network slicing-based applications in reality because they are likely to fail the objective of resource prediction which is to help network slices reduce the time delay stemming from late VM instantiation. Thus, the network operator needs accurate long-term or multistep forecasting results to avoid affecting the delay quality of the network slices.
- These existing approaches process only univariate time series data or a single variable. In other words, they simply learn the historical information of the value they want to forecast. In a virtual machine's resource usage problem, the future usage of a resource is correlated with other resources [10]. For instance, when a VM requires more central processing unit (CPU) to process a task, it also consumes more random access memory (RAM). Thus, there are correlations between variables in multivariate time series data, and a better understanding can be obtained by modeling all related variables together than by modeling only one variable [11].
- Long-term forecasting or multistep forecasting is more complicated than single-step forecasting, because of which we must consider more conditions, e.g., accumulation of errors and degradation of forecasting performance [12]. To achieve high accuracy in multistep forecasting, the effective learning of long temporal sequence data is required. When processing a long temporal sequence as input, the methods mentioned above compress all hidden representations of the temporal sequence into a fixed-length vector, which can cause information loss from the temporal sequence and affect the forecasting ability.

Understanding the above issues, we propose a novel, efficient, and proactive resource usage prediction mechanism using a Bi-long short-term memory (Bi-LSTM)-based encoder-decoder with an attention mechanism aimed at multivariate time series forecasting. It can achieve high accuracy in both short-term and long-term predictions, which enables network slices to be scaled up and down effectively and reduces the SLA violation costs as well as resource waste from overprovisioning. The proposed attention-based encoder-decoder is applied to multivariate time series resource metric data to learn the hidden long-term dependent features and nonlinear correlation features of different resource metrics. In particular, by using the temporal attention mechanism between the encoder and decoder networks, the relevant and important sequential input's hidden states across all time-steps for multistep forecasting are selected accurately to improve the forecasting ability of our model. Our contributions are listed as follows:

- We propose a novel, efficient, and proactive resource prediction mechanism using a Bi-LSTM-based encoder-decoder with an attention mechanism to forecast the future resource usage of VNF instances in a network slice with high accuracy in both short forecasting and long-term forecasting. A network operator can scale up and down the number of VNF instances in each network slice accurately and in a timely manner to satisfy the specific requirements of delay quality and sufficient resources.
- We designed an automated resource configuration system for network slices using the proposed proactive resource prediction mechanism which is at the main core of the system. Our resource configuration system monitors the resource usage of VNF instances, predicts future resource usage, and automatically adds or removes VNF instances from each network slice. Although our proposed automated resource configuration system focuses on a core network slice, it can be applied to any network functions of network slices which are implemented by NFV technology. The system was implemented by OpensourceMano [13] as an orchestrator and Openstack framework [14] as a Virtual infrastructure Manager (VIM).
- We conducted comprehensive experiments in the designed system to compare our proposed approach with other state-of-the-art resource forecast solutions in terms of forecasting accuracy in the short-term, long-term, the delay quality of the network slices, and the costs when using different resource prediction algorithms.

The rest of the paper is organized as follows. Section II provides an overview of related works on resource forecasting. In section III, the proposed attention-based encoder-decoder forecasting model is presented. Subsequently, the resource configuration system applying the proposed attention-based encoder-decoder forecasting model to scale VNF instances automatically is described in IV. Section V presents the experimental environment used in this paper. The experimental results are analyzed in section VI. Finally, we present our conclusions and discuss future research topics.

II. RELATED WORKS

The need to design a proactive resource scaling mechanism for network slices has been emphasized in the introduction section. Several approaches to proactive resource scaling in network slicing are proposed in the literature. To address the problem of network slice scaling, the authors of [8] proposed a prediction-assisted adaptive network slice expansion algorithm to dynamically deploy network slice. They used a classical statistical time series forecasting algorithm, i.e., Holt-winter to forecast future traffic demand for network slices and then utilized a VNF adaptive scaling strategy to reasonably determine the number of VNFs and resources required for network slices to meet the requirements of SLA and avoid resource wastage. Compared with static network slice deployment which does not forecast the future traffic in network slices, the method proposed in [8] can help network operators significantly reduce the costs of resource wastage. However, classical statistical models such as Holt-winter only focus on complete data, cannot handle missing or corrupt data, and only address linear relationships of data, which reduces their forecasting ability and limited in real-world applications [15]. Over the last few decades, machine learning and deep learning models have achieved considerable success in computer vision, natural language processing and time series forecasting. A few researchers have also proposed deep-learning based forecasting algorithms to forecast the future demand of VNFs in general cloud-based environment [16]–[21] or network slicing-based applications [6], [7], [22]–[24]. LSTM and three-dimensional convolutional neural network (3D-CNN) are two state-of-the-art algorithms recently used to address the resources usage forecasting problem in network slicing. In [24], a modified LSTM model called XLSTM was presented to accurately predict the resource usage of each network slice. By using XLSTM to predict future usage, a slice broker is more adept at providing resources and reduce overprovisioning and SLA violation costs by more than 10% in comparison with the traditional LSTM and ARIMA models. The 3D-CNN model was developed from legacy CNN and 2D-CNN models by adding a time dimension to the problem and transforming the input into a 3D-tensor [7]. In [7] and [6], a 3D-CNN-based mechanism was proposed to predict the future traffic load in each network slice. These works overcome some limitations of classical statistical models to achieve high performance in short-term prediction, that is, 5 or 10 minutes ahead because these algorithms learn a sequential historical information of input data to predict the future value instead of only learning the current information as performed by other algorithms. However, short-term prediction is not sufficient in real network slicing-based applications because each VM takes time to be reallocated and resources need to be reallocated as soon as possible to meet the delay quality of the network slices. Realizing that multivariate time series data can enhance forecasting accuracy, some deep learning-based forecasting algorithms utilizing multivariate time series data have been proposed to forecast the future resource usage of VNFs in general cloud-based data centers [25]–[29] and network slicing-based applications [1], [30]. In [1], the authors designed an intent-based network slicing system that can slice and manage the core network and (radio access network) RAN resources by utilizing the forecast future resource usage of a generative adversarial neural network (GAN)-based resource prediction module. In [30], the authors proposed an LSTM-based forecasting model that utilizes multivariate time series data to forecast the future CPU usage of VMs in a network slice. These multivariate time-series-based models can improve the forecasting accuracy compared to the above univariate time series-based model because these models can learn from not only the historical information of one value but also from many different values and their correlation. However, these multivariate time series-based models compress all time steps of long sequential input data into a fixed-length vector leading to information loss, which affects the long-term prediction accuracy.

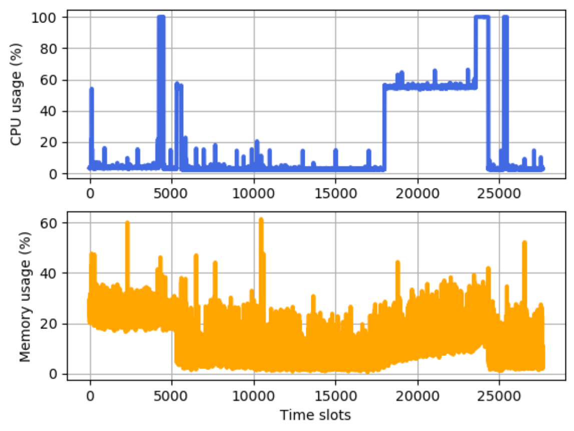
From the above survey and analysis, no existing forecasting model achieves good results in both single-step and multi-step predictions of future resource usage of network slices to help network operators avoid the cost of SLA violations and resource wastage. Therefore, we propose a novel forecasting model that can efficiently process multivariate time series data to achieve very high accuracy in both short and long-term prediction of resource usage of network slices.

III. PROPOSED ATTENTION-BASED ENCODER-DECODER FRAMEWORK

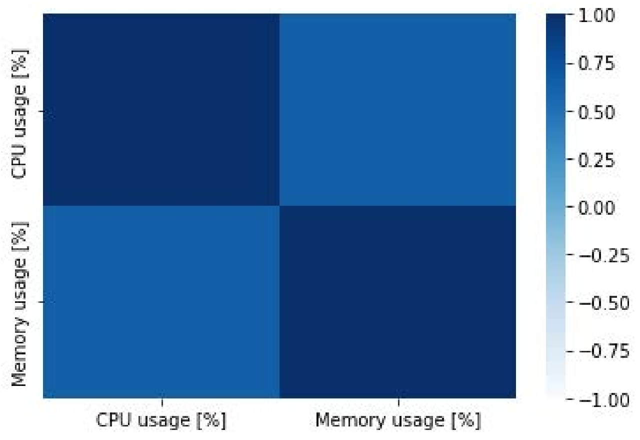
A. PROBLEM STATEMENT

In a production environment of network slicing-based applications, the usage of a resource by a VM is not independently related to that resource itself. Figure 2 shows the CPU and memory usage trace of a VM and the correlation heat map calculated by Pearson correlation coefficient [31] between these two resources in the Martena data center. In almost time slots, when the CPU usage increases, the memory usage also increases. Hence, correlations exist among resources of VMs. These potential correlations are crucial for forecasting future resource usage of VMs. In addition, the current automated resource configuration systems monitor the usage of one resource, such as CPU usage, and make a scaling decision based on that resource usage. In some cases, the memory usage of a VM reaches its maximum prior to CPU usage. If automated resource configuration systems make a scaling decision based on one resource, they cannot make a timely scaling decision and affects the quality of a network slice. Thus, it is better to know the future usage of different resources rather one resource. This is the reason we focus on the multivariate time series input-multivariate time series output forecasting problem in this study.

Moreover, every VNF running on VMs take time to instantiate and configure. Some complicated VNFs need even more times than usual to instantiate. An automated resource configuration system needs to perform a scaling decision as soon as possible to avoid affecting the delay quality of a network slice. Thus, a long-term or multistep forecasting algorithm has an important role in enhancing the delay quality of network slices. Nevertheless, the existing state-of-the-art forecasting approaches, e.g., LSTM or even the classical encoder-decoder framework which are very

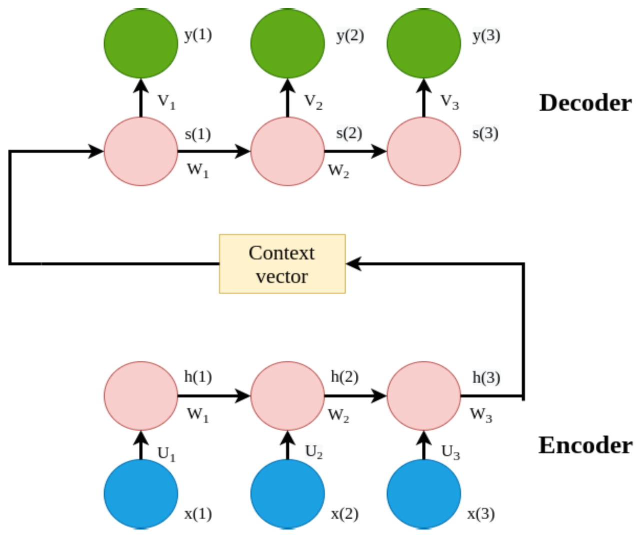


(**a**) The CPU and memory usage of a VM of Martena data center

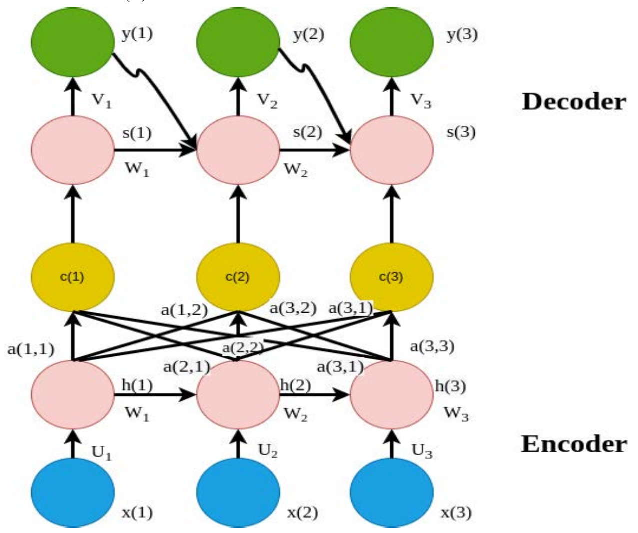


(**b**) The correlation between CPU and memory usage of the VM

FIGURE 2. The CPU and memory usage of a VM and the correlation between them.



(**a**) The classical encoder-decoder framework



(**b**) The attention-based encoder-decoder framework

FIGURE 3. The difference between classical encoder-decoder framework and attention-based encoder-decoder framework.

suitable for multivariate time series forecasting problem do not achieve good performance in multistep forecasting because they compress all hidden representations of long sequential input into a fixed-length vector which can cause information loss. Therefore, the prediction ability gradually degrades as the length of the input time series increases [32]. Based on the spirit of the research in [11], we propose an attention-based encoder-decoder forecasting algorithm that utilizes an attention mechanism to allow the decoder to selectively access encoder information during decoding. By building a different context vector for every time step of the decoder, the attention mechanism assigns different importance to the different time steps of the input sequence, gives more attention to more relevant information. Figure 3 depicts the difference between the classical encoder-decoder framework and the attention-based encoder-decoder framework when calculating the context vector.

Our research goal is to anticipate the future temporal value f_{t+1} at time $t+1$ or f_{t+p} at time $t+p$ based on the historical multivariate time series data $x = \{x_1, x_2, x_3, \ldots, x_t\}$ where $x_t \in R^M$ is the M-dimensional vector. In other words, the input data includes f_i itself and the other variables of a VNF instance. In this study, our proposed model will forecast the future CPU usage and memory usage of a VM. Thus, f_t is the two-dimensional vector or $f_t \in R^2$.

$$\begin{pmatrix} x_{t-l}^{(1)} & \ldots & x_t^{(1)} \\ \ldots & \ldots & \ldots \\ x_{t-l}^{(M)} & \ldots & x_t^{(M)} \end{pmatrix} \xrightarrow{\text{forecasting model}} \begin{pmatrix} f_{t+1}^{(1)} & \ldots & f_{t+p}^{(1)} \\ f_{t+1}^{(2)} & \ldots & f_{t+p}^{(2)} \end{pmatrix} \quad (1)$$

As shown in equation 1, the proposed forecasting model takes the history multivariate time series data $x = \{x_j^i | i = 1, 2, \ldots, M; j = t-l, \ldots, t-1, t\}$ to forecast the next p values of the VNF's resources $f = \{f_{t+j}^j | i = 1, 2; j = 1, 2, \ldots, p\}$. l is the lookup size of history data, M represents the variable number of input data, and p is the number of future steps ahead the current step.

B. OVERVIEW OF THE PROPOSED FRAMEWORK

The multivariate time series multistep forecasting framework for resource usage via attention-based encoder-decoder structure proposed in this study includes three components:

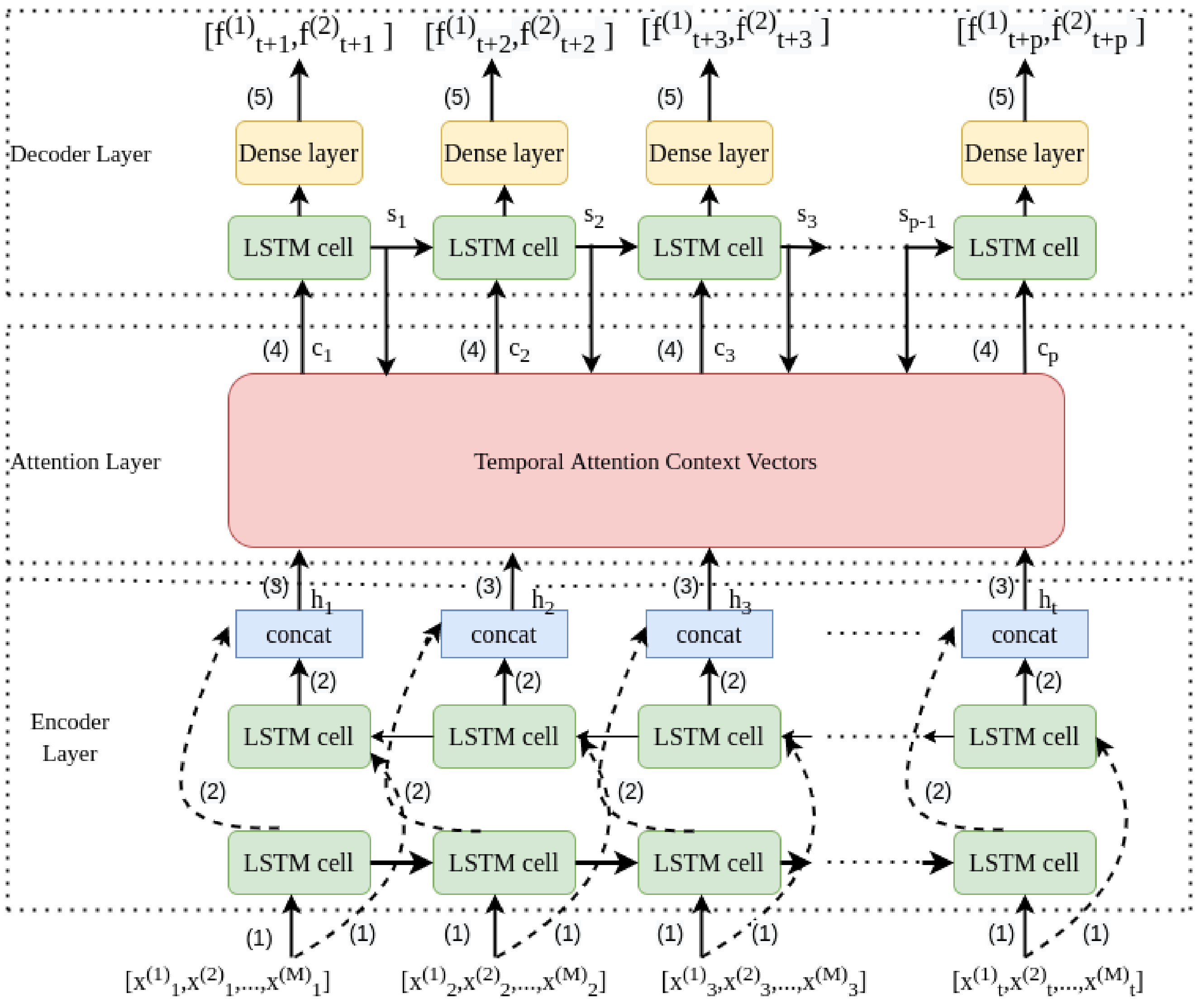


FIGURE 4. **The architecture of the proposed multivariate time series attention-based encoder-decoder model.**

Bi-LSTM as the encoder component, an LSTM as the decoder component, and a temporal attention context layer as the attention component. Figure 4 depicts the architecture and the workflow of the proposed attention-based encoder-decoder model for a multivariate time series multi-step forecasting task. Bi-LSTM learns the hidden representation of sequential input data (1) and extracts the deep temporal dependency and correlation features from the multivariate time series (2, 3). Subsequently, the sequential output of the encoder layer is fitted into the temporal attention layer with the outputs of the decoder layer to construct the attention context vectors (4). Finally, the LSTM decoder processes the attention context vectors to output future usage of resources (5). The details of each layer are presented in the following sections.

C. THE ENCODER-DECODER STRUCTURE

The encoder encodes the raw multivariate time series data $x = \{x_1, x_2, x_3, \ldots, x_t\}$ where each time step x_t in the time series is an M-dimensional vector $x_t = \{x_t^{(1)}, x_t^{(2)}, x_t^{(3)}, \ldots, x_t^{(M)}\}$ into feature representations by using an LSTM model [33]. A traditional LSTM model consists of LSTM cells calculating hidden states h_t of the input sequence at each time step t using the following equations:

$$f_t = \sigma(W_f \times [h_{t-1}, x_t] + b_f) \tag{2}$$
$$i_t = \sigma(W_i \times [h_{t-1}, x_t] + b_i) \tag{3}$$
$$\tilde{C}_t = \tanh(W_C \times [h_{t-1}, x_t] + b_C) \tag{4}$$
$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \tag{5}$$
$$o_t = \sigma(W_o \times [h_{t-1}, x_t] + b_o) \tag{6}$$
$$h_t = o_t \odot \tanh(C_t) \tag{7}$$

where i, o, f indicate the input gate, output gate, and forget gate, respectively. Here, W is a weight matrix, and b is the bias. $\tilde{C}$ and C are the candidate cell memory state and new memory state, respectively; h is the output or hidden state vector which is an encoding of input time series values until time

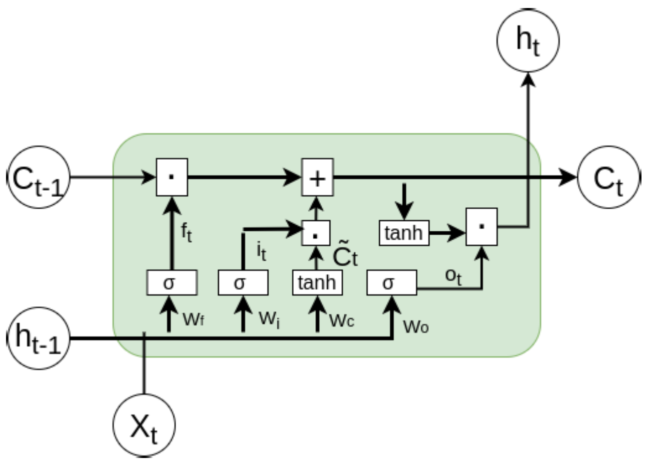


FIGURE 5. The architecture of a typical LSTM cell.

step t, x is the input vector, i is the input time, and σ denotes a sigmoid function. Figure 5 depicts the typical LSTM cell architecture.

In our proposed model, we used Bi-LSTM as the encoder to simultaneously learn sequential data in different directions through two interconnected hidden layers: a one-dimensional process uses a forward hidden layer from $t = 1$ to T, and the other direction process uses a backward hidden layer from $t = T$ to 1. The forward layer processes the sequential input at each time step t as follows:

$$\overrightarrow{f_t} = \sigma(\overrightarrow{W_f} \times [\overrightarrow{h_{t-1}}, \overrightarrow{x_t}] + \overrightarrow{b_f}) \tag{8}$$
$$\overrightarrow{i_t} = \sigma(\overrightarrow{W_i} \times [\overrightarrow{h_{t-1}}, \overrightarrow{x_t}] + \overrightarrow{b_i}) \tag{9}$$
$$\overrightarrow{\tilde{C}_t} = \tanh(\overrightarrow{W_C} \times [\overrightarrow{h_{t-1}}, \overrightarrow{x_t}] + \overrightarrow{b_C}) \tag{10}$$
$$\overrightarrow{C_t} = \overrightarrow{f_t} \odot \overrightarrow{C_{t-1}} + \overrightarrow{i_t} \odot \overrightarrow{\tilde{C}_t} \tag{11}$$
$$\overrightarrow{o_t} = \sigma(\overrightarrow{W_o} \times [\overrightarrow{h_{t-1}}, \overrightarrow{x_t}] + \overrightarrow{b_o}) \tag{12}$$
$$\overrightarrow{h_t} = \overrightarrow{o_t} \odot \tanh(\overrightarrow{C_t}) \tag{13}$$

And, the backward layer processes the sequential input at each time step t as follows:

$$\overleftarrow{f_t} = \sigma(\overleftarrow{W_f} \times [\overleftarrow{h_{t-1}}, \overleftarrow{x_t}] + \overleftarrow{b_f}) \tag{14}$$
$$\vec{i_t} = \sigma(\vec{W_i} \times [\vec{h}_{t-1}, \vec{x_t}] + \vec{b_i}) \tag{15}$$
$$\overleftarrow{\tilde{C}_t} = \tanh(\overleftarrow{W_C} \times [\overleftarrow{h_{t-1}}, \overleftarrow{x_t}] + \overleftarrow{b_C}) \tag{16}$$
$$\overleftarrow{C_t} = \overleftarrow{f_t} \odot \overleftarrow{C_{t-1}} + \overleftarrow{i_t} \odot \overleftarrow{\tilde{C}_t} \tag{17}$$
$$\overleftarrow{o_t} = \sigma(\overleftarrow{W_o} \times [\overleftarrow{h_{t-1}}, \overleftarrow{x_t}] + \overleftarrow{b_o}) \tag{18}$$
$$\overleftarrow{h_t} = \overleftarrow{o_t} \odot \tanh(\overleftarrow{C_t}) \tag{19}$$

Then, the hidden state vector h_t of input time series value until time step t is calculated by concatenating two vectors $\overrightarrow{h_t}$ and $\overleftarrow{h_t}$:

$$h_t = \overrightarrow{h_t} \oplus \overleftarrow{h_t} \tag{20}$$

By using the Bi-LSTM model, the encoder component can learn the previous context of time series data, and also the forward context of the same sequence data. Thus, the proposed model learns more features from multivariate time series data compared with the model using only the traditional LSTM model.

The decoder component utilizing an LSTM model forecasts the multi-step future usage of resources by generating target values conditioned on the previous temporal context vector and the previously generated value. The procedure for producing the target values of the decoder layer is the same as that used for a traditional LSTM layer to produce hidden states. The encoder encodes the input data for a context vector C_T. In sequence, the decoder decodes the C_T and generates the target sequence $f = \{f_{t+1}, f_{t+2}, f_{t+3}, \ldots, f_{t+p}\}$ where p is the number of time step predictions ahead of t. In the traditional encoder-decoder model, the temporal context vector C_T is a fixed-length vector, which means that a long sequential input data is encoded into a fixed-length vector. The fixed-length encoding cannot fully represent the overall information of longer input sequences. This results in the loss of key features and leads to gradient degradation of the neural network. Therefore, our proposed model uses an attention mechanism to calculate the temporal context vector to overcome this issue.

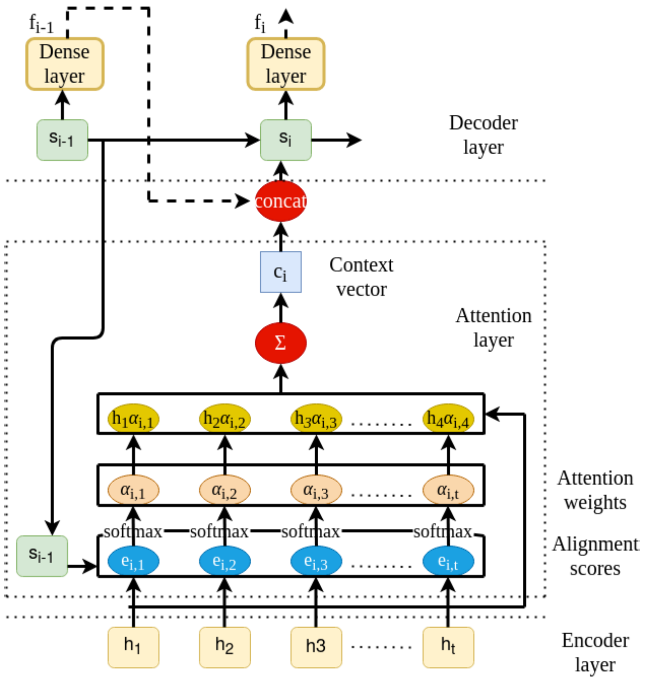


FIGURE 6. The detail architecture of the attention layer.

D. ATTENTION LAYER

In this section, we present the detailed architecture of the temporal attention layer and how the attention mechanism helps the encoder-decoder model remove the fixed-length encoding bottleneck to overcome the issue of key information loss. The main idea of the attention mechanism is to allow the decoder to selectively access encoder information during decoding. This is achieved by building a different context

Algorithm 1 The Context Vector Calculation Process of the Attention Layer

```
$h = \{h_1, h_2, h_3, \ldots, h_t\}$ ← The hidden state of the encoder layer
$s_i$ ← The hidden state of the decoder layer at time step $i$
$c_i$ ← Context vector that will be fitted into decoder's LSTM cell at time step $i$
$e_i$ ← Alignment score to calculate $c_i$
$\alpha_i$ ← Attention weight to calculate $c_i$
$p$ ← Number of time steps of the decoder layer
for $i$=1 to $p$ do
    $e_i = v_e^\top \tanh(w_e s_{i_1} + u_e h)$
    $\alpha_i = softmax(e_i)$
    $c_i = \alpha_i h$
end for
```

vector for every time step of the decoder, calculating it as a function of the previous hidden state and all the hidden states of the encoder, and assigning them trainable weights. Thus, the attention mechanism assigns different importance to the different elements of the input sequence, and gives more attention to the more relevant inputs. Our attention mechanism is adopted from the work in [32] and is also known as additive attention. The structure of the attention layer added to the encoder-decoder is shown in Figure 6. The attention layer consists of alignment score layer, attention weights, and context vector. The entire computing process of the attention layer is described in Algorithm 1. After the multivariate time series data $x = \{x_1, x_2, x_3, \ldots, x_t\}$ passes through the Bi-LSTM-based encoder layer, the hidden state h_t of each time step is used as the output of the encoder layer. The alignment score maps how well the time series inputs around time step t of the encoder and the output at time step i of the decoder match. The score is based on the hidden state of the previous decoder at time step i, i.e., s_{i-1} and the hidden state h_t of the input time series:

$$e_{i,t} = {v_e}^\top \tanh(w_e s_i + u_e h_t) \tag{21}$$

where v_e, w_e and u_e are parameters to be learned during the training process. Subsequently, a softmax function is applied to the alignment scores to obtain the attention weights:

$$\alpha_{i,t} = \frac{exp(e_{i,t})}{\sum_{k=1}^{T} exp(e_{i,k})} \tag{22}$$

The softmax function obtains the probabilities whose summary will be equal to 1. This will help to represent the weight of influence for each time step of the input sequence. The higher the attention weight of the time step, the more important the information that influences on the forecasting the future value of the time step. After the attention weights were obtained, the context vector was calculated. The context vector c_i which is decoded as the input of the decoder layer at time step i is the weighted sum of the attention weights and the encoder hidden states:

$$c_i = \sum_{t=1}^{T} \alpha_{i,t} h_t \tag{23}$$

Finally, the output of the decoder from the previous time step, i.e., f_{i-1} and the context vector are concatenated to obtain the input for the decoder at time step i.

IV. AUTOMATED RESOURCE CONFIGURATION SYSTEM DESIGN

In this section, we present the automated resource configuration system for network slices applying the proposed attention-based encoder-decoder forecasting algorithm. First, the overview of the system is described. Then, the system workflow and the system process logic are presented. Finally, the internal modules of the system are presented in detail.

A. OVERVIEW OF THE SYSTEM

To help network operators automatically control and configure resources reasonably, we propose an automated resource configuration system whose detailed architecture of the automated resource configuration system is illustrated in Figure 7. This architecture deployed on top of the ETSI NFV architecture [13] monitors the resources usage of VMs (i.e., CPU usage, RAM memory usage, disk memory,…) in each network slice, predicts future usage, and decides to scale VMs up (i.e., add new instances) or down (i.e., remove an existing instance). In this study, we deploy an automated resource configuration system to monitor and automatically scale up or down the number of instances for each VNF of the core network slice. However, this scheme can also be applied to RAN or E2E network slice. The architecture includes five different modules: the raw data processing scheme, the resource forecasting module, the scaling decision module, the management and orchestration (MANO) module and the VIM module implemented based on the OpenStack framework. The MANO module implemented based on the Open-source MANO (OSM) framework includes two sub-modules: the VNF manager (VNFM) and NFV orchestration (NFVO) module. The OSM framework provides Restful APIs to get the information of monitored resources and to create, update VNF instances. We discuss the details of each modules in the following sub-sections.

B. SYSTEM WORKFLOW

The detailed architecture and workflow of our automated resource configuration system are depicted in Figure 7. The proposed scheme includes five main modules as described in the overview system subsection. The workflow of the system is as follows. First, the NFVO sub-module of the MANO module collects the resources usage information of core network slices' VMs with the help of VIM every 5 minutes (1) and then stores it in a time series database $\overline{TSD}$ with collected data of previous observations to make a sequence of observations(2). As seen in Algorithm 2, $x_t \in R^M$ is denoted

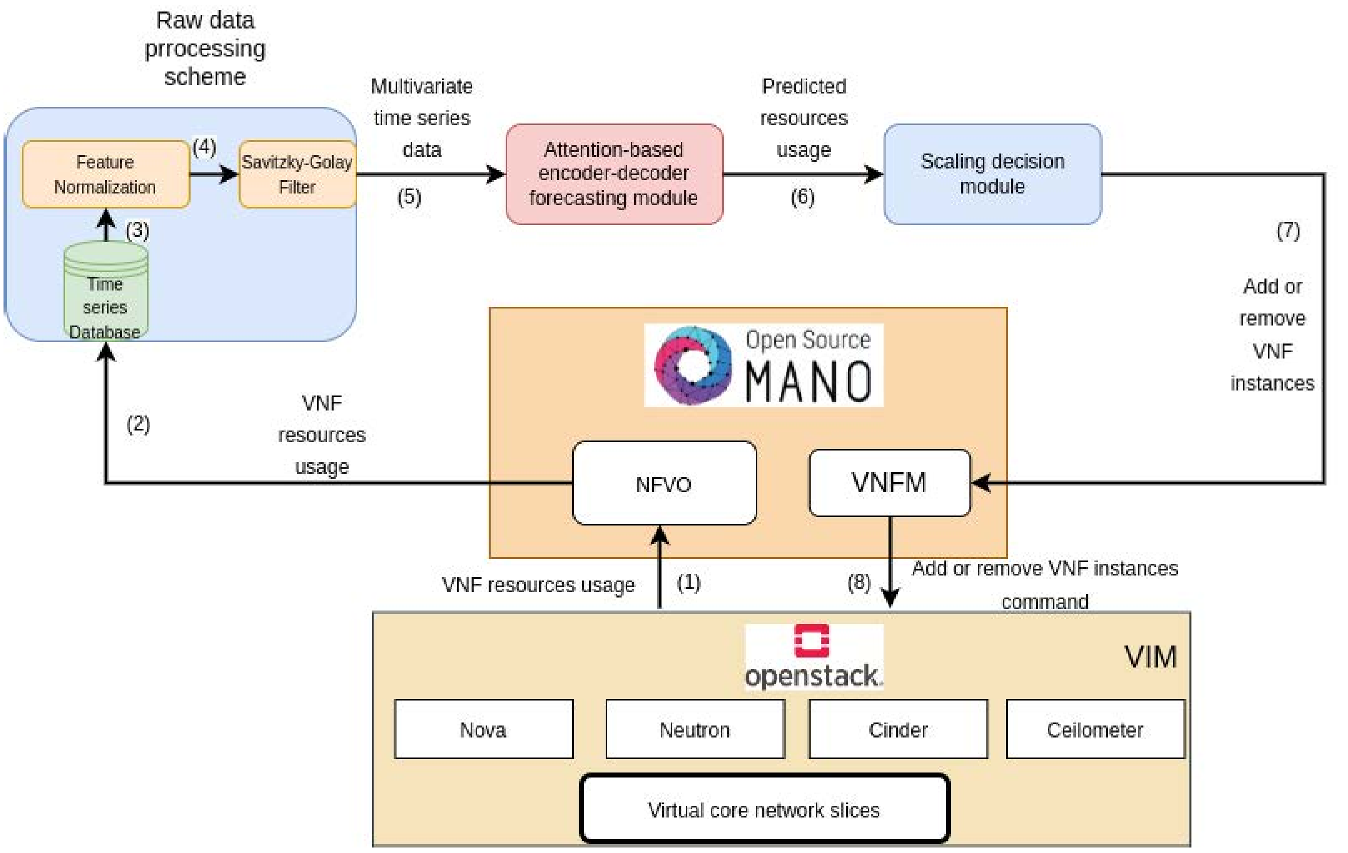

Raw data prrocessing scheme
Feature Normalization
(4)
Savitzky-Golay Filter
(3)
Time series Database
Multivariate time series data
(5)
Attention-based encoder-decoder forecasting module
Predicted resources usage
(6)
Scaling decision module
(7)
Add or remove VNF instances
Open Source MANO
(2)
VNF resources usage
NFVO
VNFM
VNF resources usage
(1)
(8)
Add or remove VNF instances command
openstack.
VIM
Nova
Neutron
Cinder
Ceilometer
Virtual core network slices

FIGURE 7. **The detail architecture of the automated resource configuration system.**

as the observation of multiple resources usage of a VNF instance at time period t and $x_{t-T:t} \in R^{M\times(T+1)}$ is denoted as a set of observations $x_{t-T}, x_{t-T+1}, \ldots, x_t$ starting from $t-T$ to t. The sequence $x_{t-T:t}$ is then normalized into the scaled data $x'_{t-T:t}$ by the feature normalization sub-module $\overline{FN}$ in the raw data processing scheme (3). Next, the normalized data are smoothed by the Savitzky-Golay filter $\overline{SGF}$ to raise the precision of the data without distorting the signal tendency (4). The preprocessed multivariate time series data $x''_{t-T:t}$ starting from $t-T$ to t are fitted into the attention-based encoder-decoder forecasting module $\overline{Forecasting}$ to forecast the future resources usage of VMs (5). In sequence, the output f of the attention-based forecasting module $\overline{Forecasting}$ is then sent to the scaling decision module which compares the future resources usage with thresholds to decide whether to add or remove a VNF instance (6).

- If $f_{t+p} \geq$ *upperThreshold* is true, then the scaling decision will send an *addInstance* request to the VNF manager module $\overline{VNFM}$ to add a new instance of the monitored VNF (7).
- If $f_{t+p} \leq$ *lowerThreshold* is true, then the scaling decision will send a *removeInstance* request to the VNF manager module $\overline{VNFM}$ to remove the monitored instance (7).

After receiving the request from the scaling decision module, the VNFM sub-module adds or deletes an instance of the monitored VNFs with the help of the VIM (8).

C. INTERNAL MODULES

Here, we present all the components of the proposed automated resource configuration scheme in order, as shown in Figure 7.

1) VIRTUAL INFRASTRUCTURE MANAGER

The VIM module implemented by the Openstack framework is a set of software tools for building and managing the cloud infrastructure. Openstack can help network operators easily deploy, monitor, and scale VMs. We deployed VNFs of core network slices in an Openstack environment. Openstack is composed of different components with different roles in managing VNFs. There are four core components:

- Nova module provides a computing service for Openstack. It is used to deploy and manage VMs. The MANO sends requests to add or remove the VNF instances to this module.
- Neutron module provides a networking service for OpenStack. This module ensures that VNF instances can communicate quickly and efficiently.
- Cinder module provides a storage service for OpenStack.
- Ceilometer module provides a telemetry service for OpenStack. It collects usage and performance measurements within an OpenStack cloud. Its primary initial targets are monitoring and metering; however, the framework is expandable to collect data for other needs. The monitoring module of the MANO obtains the

Algorithm 2 The Proposed Automated Resource Configuration System

```
$x_t = \{x_t^{(1)}, x_t^{(2)}, x_t^{(3)}, \dots, x_t^{(M)}\}$ ⟵An observation of a VNF's resources usage at time-period t
$x_{t-T:t} = \{x_{t-T}, x_{t-T+1}, \dots, x_t\}$ ⟵A multivariate time series data of a VNF's resources usage from $t-T$ to $t$
$\overline{VNFM}$ ⟵VNF Manager module
$\overline{NFVO}$ ⟵NFV module
$\overline{FN}$ ⟵Feature normalization module
$\overline{SGF}$ ⟵Savitzky-Golay filter module
$\overline{TSD}$ ⟵Time series database
$\overline{Forecasting}$ ⟵Attention-based encoder-decoder forecasting module
$upperThreshold$ ⟵The upper threshold
$lowerThreshold$ ⟵The lower threshold
$f = \{f_{t+1}, f_{t+2}, f_{t+3}, \dots, f_{t+p}\}$ ⟵Future value of a VNF's resources usage from $t+1$ to $t+p$
$addInstance$ ⟵The request of adding a new instance
$removeInstance$ ⟵The request of removing the monitored instance
loop
    Every 5 minutes $\overline{NFVO}$ collects $x_t$ and store in $\overline{TSD}$
    Send $x_{t-T:t}$ to $\overline{FN} \longrightarrow x'_{t-T:t}$
    Send $x'_{t-T:t}$ to $\overline{SGF} \longrightarrow x''_{t-T:t}$
    Send $x''_{t-T:t}$ to $\overline{Forecasting} \longrightarrow f$
    if $f_{t+p} \geq upperThreshold$ then
        Send $addInstance$ to $\overline{VNFM}$
    if $f_{t+p} \leq lowerThreshold$ then
        Send $removeInstance$ to $\overline{VNFM}$
    end if
end loop
```

resource usage information of a VNF from an API provided by this Ceilometer module.

2) MANAGEMENT AND ORCHESTRATION MODULE

The management and orchestration (MANO) module implemented by the Opensource MANO framework is responsible for creating, managing, and delivering different network services running on different network slices. The MANO module consists of two sub-modules:

- VNFM supervises a VNF or multiple VNFs and performs the life cycle management of VNF instances. Lifecycle management includes setting up, maintaining and taking down VNFs.
- NFVO manages resource and service orchestration and is responsible for the entire life-cycle management of various network services. NFVO collects information about physical and virtual resources located in network function virtualized infrastructure (NFVI) via the VIM and continuously updates its information about the available VNFs in NFVI. By doing so, NFVO initializes several network services of the network slices by chaining particular VNFs. NFVO can maintain and terminate network services.

The Opensource MANO provides Restful API to help other modules in the proposed system can interact with this module.

3) TIME SERIES DATABASE

The time series database is used to store resource usage information collected from the MANO module to make a sequence of observed data. Because the time series database stores the timestamp information of each observation, we can easily make a time series data to process in the next steps. In our implementation, we used InfluxDB [34] as the time series database.

4) FEATURE NORMALIZATION

Because our model uses multivariate time series data, there are multiple features, i.e., resource usage metrics are considered simultaneously. These features have different ranges for value and units. This significantly hinders the ability of the forecasting model. Thus, these features need to be normalized as they are shifted and rescaled so that they end up ranging a same range. The feature normalization module is responsible for normalization. Each feature x is normalized as the following formula:

$$x = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{24}$$

Here, x_{min} and x_{max} are the minimum and maximum values of the feature respectively.

5) SAVITZKY-GOLAY FILTER

The raw resource usage time series data contain significant noise caused by physical machine failures in data centers or other abnormal cases, e.g., the number of abnormal workloads and resource usage caused by unusual activities. This means that the workload and resource usage time series contain many extreme points that are considerably larger than the others. If they are not excluded from the data, the forecasting accuracy of data would be seriously affected. Thus, after normalization in the previous step, the scaled data are further smoothed to remove noise and outliers caused by a non-stationary time series. To smooth the original sequence, remove noise, and retain the peak and width of the time series, the least square polynomial smoothing method, i.e., Savitzky–Golay (SG) filter) [35] is used.

6) RESOURCE FORECASTING MODULE

The resource forecasting module is at the core of the automated resource configuration system. The module utilizes the proposed attention-based encoder-decoder algorithm to forecast the future resource usage of monitored instances. A multivariate time series $x = \{x_1, x_2, x_3, \dots, x_t\}$ where $x_t \in R^M$ (M is the number of resource metrics considered for each observation) is fitted into the proposed attention-based encoder-decoder algorithm to receive the future resource usage $f = \{f_{t+1}, f_{t+2}, f_{t+3}, \dots, f_{t+p}\}$ where p is the number

TABLE 1. Features for the attention-based encoder-decoder model.

Feature	Description
CPU capacity provisioned (MHZ)	The capacity of the CPUs in terms of MHZ
CPU usage (MHZ)	Utilization of the CPU in terms of MHZ
CPU usage (%)	Utilization of the CPU in terms of percentage
Memory provisioned (KB)	The capacity of the memory of the VM in terms of KB
Memory usage (%)	The memory actively used in terms of percentage
Memory usage (KB)	The memory actively used in terms of KB
Network in (KB/s)	Network received throughput in terms of KB/s
Network out (KB/s)	Network transmitted throughput in terms of KB/s
Disk read	Disk read throughput in terms of KB/s
Disk write	Disk write throughput in terms of KB/s

of steps ahead the current time step t and each step is 5 minutes. As analyzed in the subsection III-A, our model solves the problem of multivariate input-multivariate output. Thus, the resource forecasting model will forecast the future usage of two resources: CPU usage and memory usage. In other words, each $f_t \in R^2$.

7) SCALING DECISION MODULE

The scaling decision module receives future resource usage information from the resource forecasting module and compares two different upper thresholds and two different lower thresholds for CPU and memory usage. Subsequently, it sends the scaling decision, i.e., adding or removing a VNF instance to the MANO module through Restful APIs.

V. EXPERIMENTAL SETUP

In this section, the dataset used for training the proposed attention-based encoder-decoder model is presented. Then, the model parameters for the attention-based encoder-decoder forecasting model is presented. Finally, the implementation of the testbed is detailed.

A. DATASET DESCRIPTION

In this study, an open-source dataset [36] provided by the MARTENA data center which is a well-known service provider and IT product manufacturing organization is used for training the proposed attention-based encoder-decoder forecasting model. The dataset contains performance metrics of 54, 520, and 527 data center VMs. The dataset was collected in three traces, formatted as csv files within a period of three months (one trace for one-month timestamp). Each row of the csv file is the average usage of 12 resources of a VM and workload arriving at this VM in 5 minutes. However, not all 12 features are relevant to the future usage of resources we want to forecast. Thus, to enhance the accuracy, training speed of the model, and response time of the automated resource configuration system, a feature selection process was ran to select the most relevant features for forecasting purpose. For the feature selection process, the state-of-the-art Boruta algorithm [37] was applied to select the relevant features that contributes the most to the forecast output. The Boruta algorithm first duplicates an original dataset. Then, the z_{score} values, which were obtained from a combination of the original and the copied dataset using a random forest algorithm, were used for comparison. If the z_{score} values of the duplicated dataset were consecutively smaller than that of the original dataset, then the selected feature would be chosen. The selected features are presented in Table 1. These features were used to train the forecasting model and collected by the MANO module of the automated resource configuration system.

TABLE 2. Hyper parameter settings of the attention-based encoder-decoder model.

Parameter	Value
Number of encoder layer	1
Number of decoder layer	2
Sequence length of encoder layer	30
Node number of encoder layer and decoder-lstm layer	100
Node number of decoder-dense layer	2
Bidirectional-lstm merge mode	sum
The active function of attention layer	softmax
Training epochs	100
Dropout	0.3
Optimizer	Adagrad
Batch size	96
Loss function	mse

B. MODEL PARAMETERS FOR THE ATTENTION-BASED ENCODER-DECODER FORECASTING MODEL

After the feature selection process, the dataset with the selected features was preprocessed by the raw data processing module and then was used to train the resource forecasting model. The hyperparameters of the attention-based encoder-decoder model are listed in Table 2. The sequence length of the encoder layer was 30 and the sum function was selected as the Bi-LSTM merge mode. We built and trained three attention-based encoder-decoder models with three different output sequential lengths, i.e., 1, 5, and 10 to test the forecasting ability of the proposed model in short-term forecasting (one-step prediction) or long-term forecasting (five and ten-step prediction). The number of units in each LSTM cell of both the encoder and decoder layer was 100. The loss function was mean square error (MSE) and Adagrad function was the model optimizer. The training process was onducted over 100 epochs.

C. DETAIL IMPLEMENTATION AND TEST PREPARATION

In this work, we setup our testbed for the automated configuration system as suggested in [38]. Figure 8 depicts the

TABLE 3. Physical hardware setting in detail.

Physical node	Processor	Memory	Disk
Computer 1	Intel®i7-8700(12) 4.6000GHZ	64237 MiB	SSD 512GB
Computer 2	Intel®i7-4770(8) 3.900GHZ	15930 MiB	SSD 128GB
Computer 3	Intel®i7-8700(12) 4.6000GHZ	64237 MiB	SSD 512GB

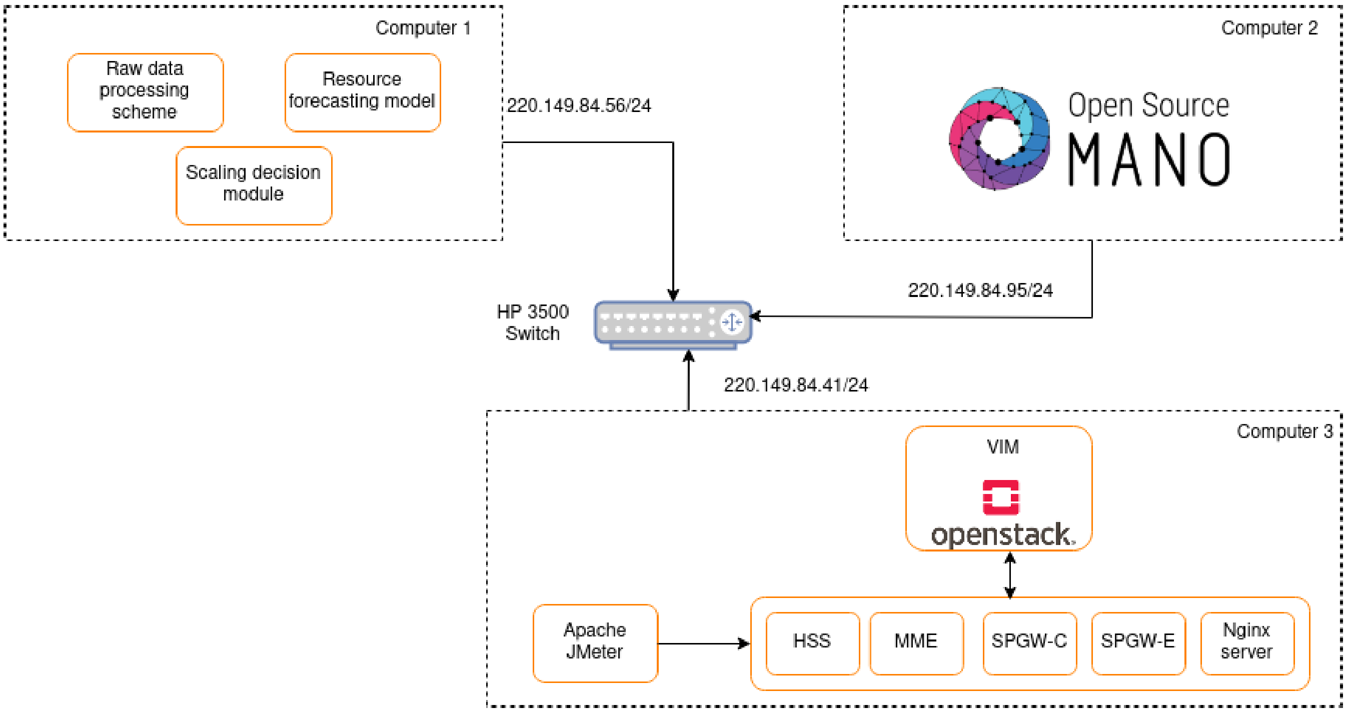


FIGURE 8. Our testbed deployment for the automated resource configuration system.

testbed environment. The automated configuration system was deployed on three physical computers. The raw data processing scheme, the resource forecasting module, and the scaling decision module were deployed on the computer 1. To run both feature normalization and Savitzky-Golay filter, *Sklearn.preprocessing* and *SciPy* python library were used. Influx DB version 1.7.9 was used as the time series database. The attention-based encoder-decoder was coded by using the Keras library [39]. The scaling decision module was a Python script. The MANO module utilizing an open-source MANO framework was deployed on the computer 2. We used the open-source MANO version 6.0. In the computer 3, we first deployed OpenStack for the VIM module and then deployed two core network slices. In our testbed, each core network slice is a service function chain of four network functions: Home Subscriber Server (HSS), Mobility Management Entity (MME), Control Plane of the Service Packet Gateway (SPGW-C) and User Plane of the Service Packet Gateway (SPGW-U). Each of these network functions is deployed on a VM using the VIM OpenStack and OSM. We used the OpenAirInterface open-source [40] to operate these core VNFs. These four VNF instances run based on Ubuntu version 18.04, and the resource configuration was vCPU 2 core, 4GB RAM, 10GB disk.

After creating the core network slices, we need to generate traffic passing through the two core network slices to test the efficiency of the proposed automated configuration system to help scale the core network slices to reduce the costs of SLA violations and overprovision. An Nginx server installed in a VM was connected to each core network slice. Additionally, we used Apache JMeter [41], which is an open-source load testing tool for web applications to generate dynamic HTTP requests to the Nginx servers. The Apache JMeter tool generated traffic for 120 minutes. The specifications of the three physical computers are listed in Table 3. These three physical computers were connected by an HP3800 switch to communicate with each other.

VI. RESULT ANALYSIS

In this section, we first evaluate the forecasting ability of the proposed attention-based encoder-decoder forecasting model for short-term and long-term prediction. In sequence, we evaluate the efficiency of the proposed automated resource configuration system in helping network operators reduce the costs of SLA violations and resource overprovisioning. Finally, we conclude this section with a general discussion.

A. COMPARISON WITH OTHER SOLUTIONS

In this study, we compared our proposed method in forecasting ability with other state-of-the-art methods in both short-term forecasting and long-term forecasting. To the best of our knowledge, the approaches in [7], [8], [24] are state-of-the-arts approaches for resource forecasting in network slicing. These methods use univariate time series data to forecast the future usage of one type of VNF resource. The approach in [8] proposes a classical time series model called Holt-winter to predict the next time-step future usage of VNF resource. The method in [24] proposes a forecasting model utilizing a modified version of LSTM called XLSTM. In [7], the authors propose 3-dimension CNN (CNN3D) to forecast the future resource usage. Moreover, to prove that our attention mechanism can enhance the forecasting ability in both short and long-term forecasting, we also compared our proposed model with two other multivariate time series-based forecasting models in [1] and [30]. The forecasting model in [1] utilizes the GAN model to forecast the multiple resources of VNF instances. The method in [30] proposes a modified version of the LSTM model utilizing multiple resources of VNF instances to forecast the short-term and long-term future usage of VNF resources. We denote this method as multi-LSTM. These forecasting models are then embedded into the automated resource configuration model to compare the efficiency of helping network operators reduce the costs of SLA violations and resource overprovisioning.

B. EVALUATION OF FORECASTING ABILITY

To evaluate the forecasting ability of our proposed forecasting model, we used four metrics as shown in the following equations:

- Mean square error (MSE) is the average squared difference between the forecast and the actual value:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \tag{25}$$

- Root mean square error (RMSE) is the square root of two of the differences between the forecast and the actual value:

$$RMSE = \sqrt{MSE} \tag{26}$$

- Mean absolute error (MAE) is the average of the absolute differences between the forecast and the actual value.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|Y_i - \hat{Y}_i| \tag{27}$$

- R^2 is the coefficient of determination which is the square of the coefficients of multiple correlations between the actual and the predicted value:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}{\sum_{i=1}^{n}(Y_i - \bar{Y}_i)^2} \tag{28}$$

where Y_i is the actual value, $\hat{Y}_i$ is the forecast value and $\bar{Y}_i$ is the mean value of Y at the observation time i. We evaluated different methods mentioned in the section VI-A in one-step forecasting, five-step forecasting, and ten-step forecasting.

First, we evaluated the short-term forecasting of the proposed model. Table 4 and Table 5 show the one-step forecasting performance, i.e., 5 minutes ahead prediction of different approaches in CPU and memory usage respectively. Each approach was applied to the same training dataset and testing dataset. For one-step forecasting, all of the approaches achieved quite good results because they are also current state-of-the-art methods for resource forecasting problem. The deep learning-based forecasting methods, i.e., XLSTM, CNN3D, GAN, and multi-LSTM still achieved better results than Holt-winter in [8] which is a classic statistical time series model. Compared to other methods using univariate time series data, the multivariate time series data-based methods which are GAN in [1], multi-LSTM in [30], and our approach achieved better performance because these multivariate time series data-based methods learn both the historical information of the forecasting resource, i.e., CPU or memory, and also the historical information of other metrics and their correlation. In particular, our proposed approach still achieved the best results in terms of MSE, MAE, RMSE, R^2 among all of the methods (MSE = 9.3, RMSE = 3.05, MAE = 1.82, R^2 = 0.97 for CPU usage forecasting and MSE = 10.3, RMSE = 3.21, MAE = 2.11 and R^2 = 0.96 for memory usage forecasting.). Using the attention mechanism, our model can select important time steps from long sequential input data for learning. To visualize the statistical distribution of absolute error calculated by the absolute difference between the predicted value and the actual value, the one-step forecasting absolute error of CPU usage and memory usage when using different methods on the test dataset are shown in Figure 9.

Then, we evaluated the long-term prediction by testing the forecasting models in five-step, i.e., 25 minutes ahead prediction and ten-step, i.e., 50 minutes ahead prediction. Table 6 and Table 7 show the five-step and ten-step prediction of CPU usage respectively. It is clear that our proposed model outperformed other methods in terms of MSE, MAE, RMSE, and R^2. For the five-step prediction, the MSE, MAE, RMSE, R^2 are 10.5, 3.01, 3.24, 0.96 respectively. For the ten-step prediction, the MSE, MAE, RMSE, R^2 are 11.42, 3.25, 3.38, 0.95. From the results presented in Table 6 and Table 7, when the prediction time steps increased, the prediction ability of all models decreased gradually, which means that the cumulative error of the multi-step forecasting model increased. This is because the longer the time step in the future, the more difficult it is to predict. Although the methods in [1], [30], and our method both learn from multivariate time

TABLE 4. CPU usage one-step prediction performance comparison of different approaches.

Model type	Holt-winter	XLSTM	CNN3D	GAN	Multi-LSTM	Our proposed model
MSE	12.5	11.3	11.1	10.7	10.5	**9.3**
RMSE	3.53	3.36	3.33	3.27	3.24	**3.05**
MAE	3.21	3.15	3.13	2.87	2.80	**1.82**
R^2	0.87	0.90	0.89	0.92	0.95	**0.97**

TABLE 5. Memory usage one-step prediction performance comparison of different approaches.

Model type	Holt-winter	XLSTM	CNN3D	GAN	Multi-LSTM	Our proposed model
MSE	20.3	17.4	16.1	13.2	13.0	**10.3**
RMSE	4.50	4.17	4.01	3.63	3.60	**3.21**
MAE	4.03	3.86	3.81	3.14	3.08	**2.11**
R^2	0.75	0.86	0.88	0.90	0.91	**0.96**

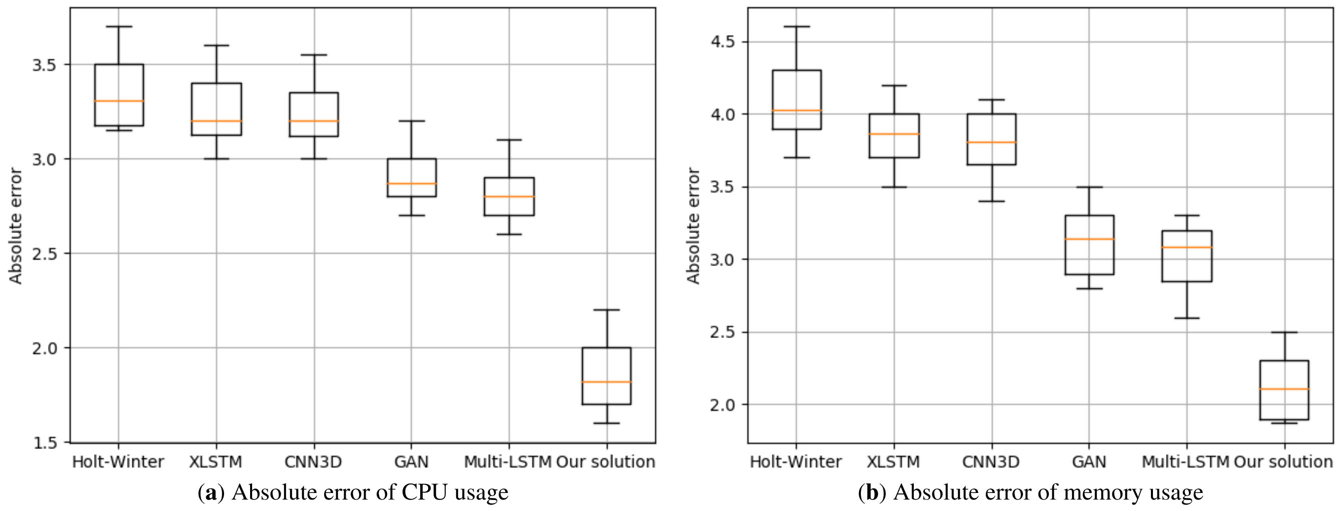


FIGURE 9. One-step forecasting absolute error of different methods.

TABLE 6. CPU usage 5-step prediction performance comparison of different approaches.

Model type	Holt-winter	XLSTM	CNN3D	GAN	Multi-LSTM	Our proposed model
MSE	19.6	17.4	17.1	15.9	15.6	**10.5**
RMSE	4.43	4.17	4.13	3.99	3.95	**3.24**
MAE	4.25	4.14	4.10	3.87	3.85	**3.01**
R^2	0.80	0.85	0.86	0.87	0.89	**0.96**

TABLE 7. CPU usage 10-step prediction performance comparison of different approaches.

Model type	Holt-winter	XLSTM	CNN3D	GAN	Multi-LSTM	Our proposed model
MSE	24.62	22.34	22.15	18.54	18.15	**11.42**
RMSE	4.96	4.73	4.71	4.31	4.26	**3.38**
MAE	4.82	4.68	4.65	4.27	4.22	**3.25**
R^2	0.77	0.81	0.82	0.84	0.86	**0.95**

series data to forecast the future usage, our method achieves considerably better results because we use a feature selection process to choose the most relevant features for future value of CPU and memory usage. In particular, by using the attention mechanism, our model can handle long sequential data better than other methods. The attention layer assigns different weights to each time step of the sequential data from the encoder layer in order to learn from the most important time steps. Achieving high accuracy in long-term prediction is beneficial to network slicing-based applications in reality because of the limitation of NFV technology in frequently reallocating resources as discussed in Section I. The results for long-step prediction of memory usage showed in Table 8 and Table 9 demonstrate that our model is superior to other

TABLE 8. Memory usage 5-step prediction performance comparison of different approaches.

Model type	Holt-winter	XLSTM	CNN3D	GAN	Multi-LSTM	Our proposed model
MSE	25.51	21.42	21.14	18.23	18.12	**12.34**
RMSE	5.05	4.63	4.60	4.27	4.26	**3.51**
MAE	4.99	4.58	4.56	4.24	4.21	**3.47**
R^2	0.70	0.81	0.83	0.84	0.86	**0.95**

TABLE 9. Memory usage 10-step prediction performance comparison of different approaches.

Model type	Holt-winter	XLSTM	CNN3D	GAN	Multi-LSTM	Our proposed model
MSE	29.53	25.51	25.48	21.27	21.14	**14.15**
RMSE	5.43	5.05	5.04	4.61	4.60	**3.76**
MAE	5.39	4.99	4.98	4.55	4.54	**3.67**
R^2	0.66	0.76	0.78	0.80	0.81	**0.95**

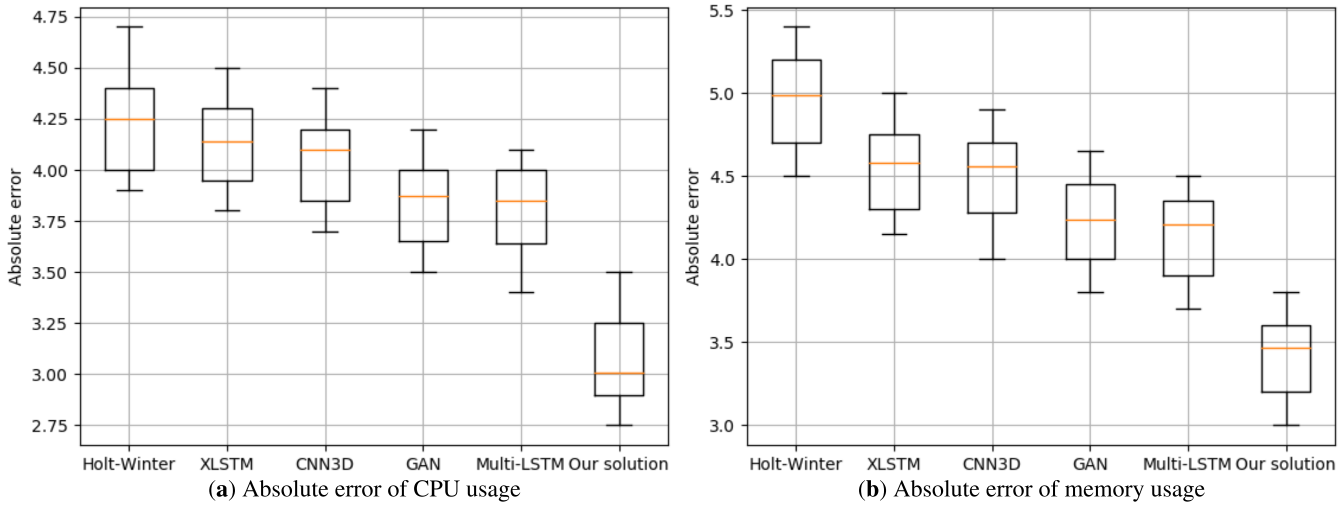


FIGURE 10. Five-step forecasting absolute error of different methods.

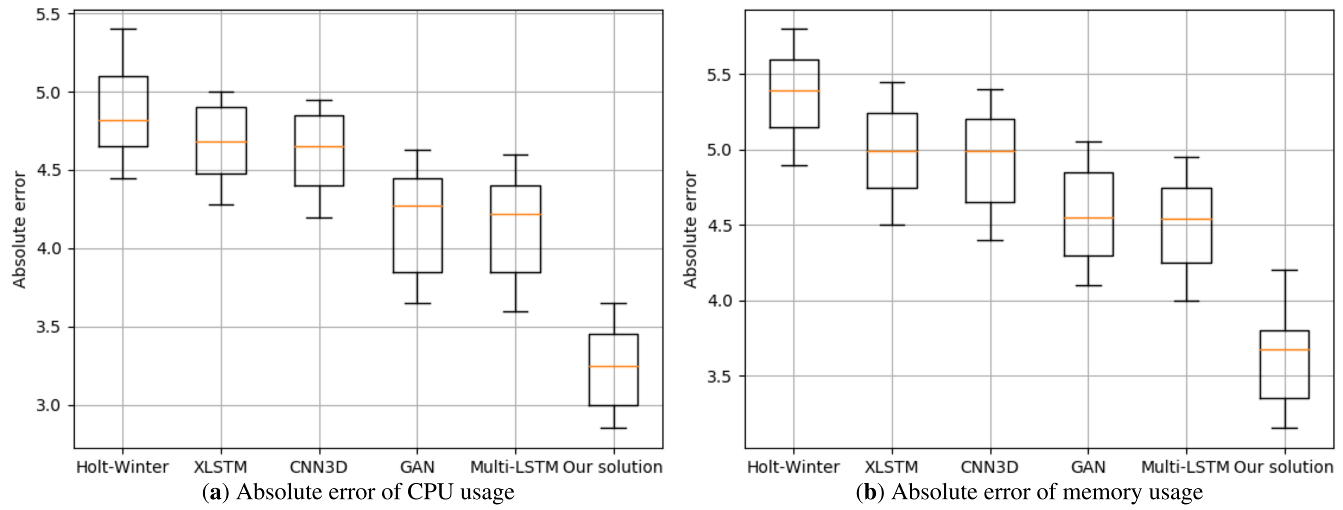


FIGURE 11. Ten-step forecasting absolute error of different methods.

state-of-the-art models in terms of both short-term and long-term prediction. To visualize the forecasting results, we show the five-step and ten-step forecasting absolute difference between the predicted value and the actual value of CPU usage and memory usage on the test dataset when using different methods in Figure 10 and Figure 11, respectively.

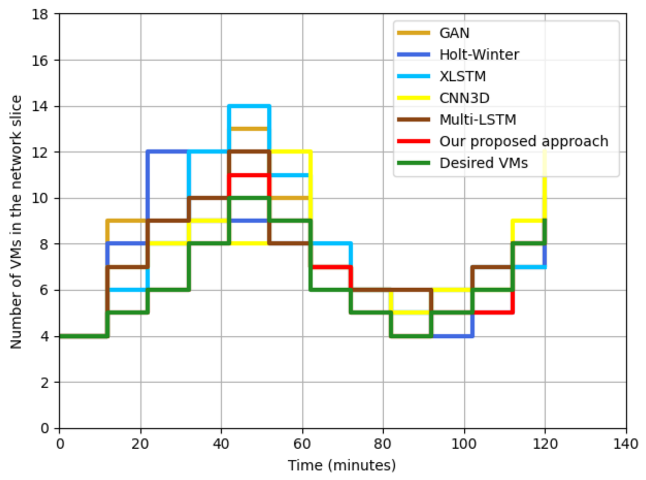


FIGURE 12. Number of VNF instances of a core network slice produced by different methods.

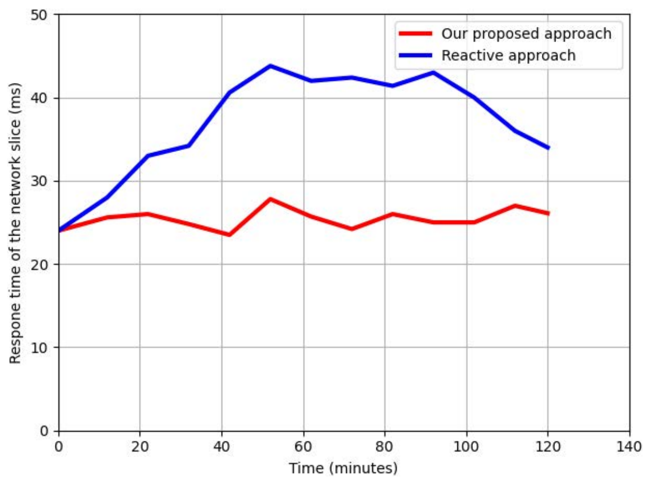


FIGURE 13. Response time of the network slices.

C. EVALUATION OF THE PROPOSED AUTOMATED RESOURCE CONFIGURATION SYSTEM

We embedded the proposed attention-based forecasting algorithm into an automated resource configuration system to evaluate how the proposed forecasting algorithm helps the network operator reduce the costs and enhance the delay quality in the network slices. The proposed automated resource configuration system adds a new VNF instance or removes the existing VNF instance base on the output, i.e., future usage of CPU and memory of the monitored VNF instance of the forecasting algorithm. Based on the results from section VI-B, we chose the five-step forecasting model for the resource forecasting module of the automated resource configuration system. The automated resource configuration system receives the next five time steps, i.e., 25 minutes of resource usage of VNF instances (i.e., CPU and memory usage) and if the future CPU usage or memory usage is greater than 80%, a new VNF instance of the monitored VNF will be added, if the future resource CPU usage or memory usage is less than 10% the monitored VNF instance will be removed. To evaluate the accuracy of the proposed forecasting algorithm, we embedded different forecasting methods into the automated resource configuration system and conducted the same experiments to compare. Figure 12 shows the number of VNF instances of the core network slices allocated during the experiment using different forecasting algorithms. Following the experiment in [42], to calculate how much the network operator has to pay for resource overprovisioning and SLA violation cost, we assume that the cost of deploying a core network slice-type VNF instance with 2vCPU is 200$ and the cost of SLA violation is 100$ per instance not allocated. To deploy each VNF instance, the network operator has to pay a deployment cost. Based on [8], the formula of VNF instances deployment cost is:

$$U_{deployment} = \lambda n \tag{29}$$

where λ represents the cost deploying a VNF instance, n denotes the number of the VNF instances in a network slice. Here, we are considering how our approach can help the network operator reduce the cost of overprovisioning and SLA violation cost. Thus, the formula 29 can be transformed to:

$$U_{over} = \lambda n_{over} \tag{30}$$

where U_{over} is the cost of overprovisioning, n_{over} is the number of VNF instances overallocated in the network slice. The SLA violation cost determines a fixed cost for the network operator at every time interval in which the tenant demand is not satisfied and is calculated as the following formula:

$$U_{violation} = \mu n_{under} \tag{31}$$

where μ denotes the cost the network operator has to pay for each time interval the tenant demand is not satisfied, n_{under} denotes the number of time interval when the network operator does not satisfy the tenant demand. Based on equations 30 and 31, we determined the operational cost of the network operator when applying different approaches as presented in Table 10. We observe that our approach yielded substantially lower costs than all other solutions. The expenses for unnecessary resource allocation when applying the proposed method was only 400$ which is less 7 times than the cost when applying multi-LSTM in [30] (2800$) which makes the lowest cost among other solutions. The fees for SLA violation when applying the proposed method are 100$ which is the lowest cost among other solutions. These results reflect the results in section VI-B. Compared with the best approach in other solutions, our approach can help the network operator reduce 85% of the cost occurred by resource overprovisioning and SLA violation.

Now, we evaluate the delay quality of network slices when applying different forecasting algorithms in the automated resource configuration system. Figure 13 shows the response time of the network slice when applying the automated resource configuration which uses the proposed attention-based encoder-decoder model and when applying a reactive approach. The reactive approach also sets a threshold as our

TABLE 10. **Operational costs comparison of different approaches.**

Model type	Holt-winter	XLSTM	CNN3D	GAN	Multi-LSTM	Our proposed model
U_{over}	3000	3800	3200	3000	2800	**400**
$U_{violation}$	500	300	200	100	100	**100**

approach but it does not predict the future resource usage. The reactive approach monitors the current resource usage of an instance and if the current resource reaches the predefined-threshold, it will add a new instance or remove the instance. According to the result, our approach can reduce the response time of the core network slice (i.e., 25.43 ms in average) compared with the reactive approach (i.e., 33.87 ms in average). The reactive approach only makes a scaling decision after the current resource usage of the VNF instance exceeds the predefined threshold. Because there will be a delay time for creating a new instance as discussed, the reactive approach makes the response time of the core network slice be higher when the traffic arriving at the core network slice increases higher and higher. In contrast, the automated resource configuration system will forecast the resource usage of the VNF instances in the next 25 minutes and make a scaling decision more accurately and ahead of time. Thus, the automated resource configuration can enhance the delay quality of the network slice.

D. GENERAL DISCUSSION

Based on the comprehensive results provided above, we summarize some of the outstanding points demonstrating the effectiveness of the proposed attention-based encoder-decoder forecasting model in automated resource configuration network slices through the experiments conducted on our practical testbed:

- Our proposed attention-based encoder-decoder forecasting model achieves the highest accuracy no matter of the condition of short-term or long-term VNF resource forecasting compared with other current state-of-the-arts methods applying for resource prediction in network slicing.
- The automated resource configuration system which applies the proposed attention-based encoder-decoder forecasting model can allocate the most appropriate number of VNF instances in network slices compared with other approaches. Therefore, the cost of SLA violation and resource overprovisioning that the network operator has to pay when uses our approach is only 17% of that when using the best-performing benchmark.
- The automated resource configuration system which applies the proposed attention-based encoder-decoder forecasting model enhances the delay quality of network slices.

VII. CONCLUSION AND FUTURE WORK

In this paper, we first proposed and evaluated a novel VNF resource forecasting algorithm which leverages the idea of multivariate time series forecasting with an encoder-decoder learning structure augmented by a temporal attention mechanism. By learning the long-term temporal dependency pattern, nonlinear correlation features of multivariate time series data, and choosing the most important information in long sequential input data, the proposed forecasting model can effectively forecast the VNF resources in network slices for both short-term and long-term prediction. Then, an automated resource configuration system using the proposed forecasting model to automatically scale in or scale out VNFs for network slices is proposed and evaluated. The proposed automated resource configuration system can help the network operator minimize the cost of SLA violation and resource overprovisioning and enhance the delay quality for network slices by allocating the appropriate number of VNF instances.

In the future, we plan to apply the proposed resource automated scheme to manage and automatically scale VNF instances in RAN slicing to build an automated resource configuration system for an E2E network slicing. In addition, the idea of the attention mechanism can be applied to other problems of network slicing, e.g., admission control problem.

REFERENCES

[1] K. Abbas, M. Afaq, T. A. Khan, A. Rafiq, and W.-C. Song, "Slicing the core network and radio access network domains through intent-based networking for 5G networks," *Electronics*, vol. 9, no. 10, p. 1710, Oct. 2020.

[2] W. Guan, X. Wen, L. Wang, Z. Lu, and Y. Shen, "A service-oriented deployment policy of end-to-end network slicing based on complex network theory," *IEEE Access*, vol. 6, pp. 19691–19701, 2018.

[3] D. Bega, M. Gramaglia, R. Perez, M. Fiore, A. Banchs, and X. Costa-Pérez, "AI-based autonomous control, management, and orchestration in 5G: From standards to algorithms," *IEEE Netw.*, vol. 34, no. 6, pp. 14–20, Nov. 2020.

[4] V. G. Nguyen, A. Brunstrom, K.-J. Grinnemo, and J. Taheri, "SDN/NFV-based mobile packet core network architectures: A survey," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1567–1602, 3rd Quart., 2017.

[5] S. D. A. Shah, M. A. Gregory, and S. Li, "Cloud-native network slicing using software defined networking based multi-access edge computing: A survey," *IEEE Access*, vol. 9, pp. 10903–10924, 2021.

[6] A. Rizwan, M. Jaber, F. Filali, A. Imran, and A. Abu-Dayya, "A zero-touch network service management approach using AI-enabled CDR analysis," *IEEE Access*, vol. 9, pp. 157699–157714, 2021.

[7] D. Bega, M. Gramaglia, M. Fiore, A. Banchs, and X. Costa-Pérez, "DeepCog: Optimizing resource provisioning in network slicing with AI-based capacity forecasting," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 2, pp. 361–376, Feb. 2020.

[8] J. Zhou, W. Zhao, and S. Chen, "Dynamic network slice scaling assisted by prediction in 5G network," *IEEE Access*, vol. 8, pp. 133700–133712, 2020.

[9] G. E. P. Box and D. A. Pierce, "Distribution of residual autocorrelations in autoregressive-integrated moving average time series models," *J. Amer. Stat. Assoc.*, vol. 65, no. 332, pp. 1509–1526, 1970.

[10] H. Xi, C. Yan, H. Li, and Y. Xiao, "An attention-based recurrent neural network for resource usage prediction in cloud data center," *J. Phys., Conf. Ser.*, vol. 2006, no. 1, Aug. 2021, Art. no. 012007.

[11] S. Du, T. Li, Y. Yang, and S.-J. Horng, "Multivariate time series forecasting via attention-based encoder–decoder framework," *Neurocomputing*, vol. 388, pp. 269–279, May 2020.

[12] Y. Bao, T. Xiong, and Z. Hu, "Multi-step-ahead time series prediction using multiple-output support vector regression," *Neurocomputing*, vol. 129, pp. 482–493, Apr. 2014.

[13] *Opensource Mano*. Accessed: 2021. [Online]. Available: https://osm.etsi.org/

[14] *Welcome to OpenStack Documentation*. Accessed: 2021. [Online]. Available: https://www.openstack.org/
[15] A. Mahmud and A. Mohammed, "A survey on deep learning for time-series forecasting," in *Machine Learning and Big Data Analytics Paradigms: Analysis, Applications and Challenges*. Cham, Switzerland: Springer, 2021, pp. 365–392.
[16] S. Rahman, T. Ahmed, M. Huynh, M. Tornatore, and B. Mukherjee, "Auto-scaling VNFs using machine learning to improve QoS and reduce cost," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2018, pp. 1–6.
[17] M. Imdoukh, I. Ahmad, and M. G. Alfailakawi, "Machine learning-based auto-scaling for containerized applications," *Neural Comput. Appl.*, vol. 32, no. 13, pp. 9745–9760, Jul. 2020.
[18] D. Janardhanan and E. Barrett, "CPU workload forecasting of machines in data centers using LSTM recurrent neural networks and ARIMA models," in *Proc. 12th Int. Conf. Internet Technol. Secured Trans. (ICITST)*, Dec. 2017, pp. 55–60.
[19] J. J. Prevost, K. Nagothu, B. Kelley, and M. Jamshidi, "Prediction of cloud data center networks loads using stochastic and neural models," in *Proc. 6th Int. Conf. Syst. Syst. Eng.*, Jun. 2011, pp. 276–281.
[20] M. Duggan, K. Mason, J. Duggan, E. Howley, and E. Barrett, "Predicting host CPU utilization in cloud computing using recurrent neural networks," in *Proc. 12th Int. Conf. Internet Technol. Secured Trans. (ICITST)*, Dec. 2017, pp. 67–72.
[21] Y. Guo and W. Yao, "Applying gated recurrent units approaches for workload prediction," in *Proc. IEEE/IFIP Netw. Oper. Manage. Symp. (NOMS)*, Apr. 2018, pp. 1–6.
[22] Q. Guo, R. Gu, Z. Wang, T. Zhao, Y. Ji, J. Kong, R. Gour, and J. P. Jue, "Proactive dynamic network slicing with deep learning based short-term traffic prediction for 5G transport network," in *Proc. Opt. Fiber Commun. Conf. (OFC)*, 2019, pp. 1–3.
[23] I. Afolabi, J. Prados-Garzon, M. Bagaa, T. Taleb, and P. Ameigeiras, "Dynamic resource provisioning of a scalable E2E network slicing orchestration system," *IEEE Trans. Mobile Comput.*, vol. 19, no. 11, pp. 2594–2608, Nov. 2020.
[24] C. Gutterman, E. Grinshpun, S. Sharma, and G. Zussman, "RAN resource usage prediction for a 5G slice broker," in *Proc. 20th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, Jul. 2019, pp. 231–240.
[25] S. Gupta and D. A. Dinesh, "Resource usage prediction of cloud workloads using deep bidirectional long short term memory networks," in *Proc. IEEE Int. Conf. Adv. Netw. Telecommun. Syst. (ANTS)*, Dec. 2017, pp. 1–6.
[26] S. Ouhame, Y. Hadi, and A. Ullah, "An efficient forecasting approach for resource utilization in cloud data center using CNN-LSTM model," *Neural Comput. Appl.*, vol. 33, no. 16, pp. 10043–10055, Aug. 2021.
[27] M. N. H. Shuvo, M. N. H. Shuvo, M. M. S. Maswood, M. M. S. Maswood, A. G. Alharbi, and A. G. Alharbi, "LSRU: A novel deep learning based hybrid method to predict the workload of virtual machines in cloud data center," in *Proc. IEEE Region 10 Symp. (TENSYMP)*, Jun. 2020, pp. 1604–1607.
[28] J. Bi, S. Li, H. Yuan, and M. Zhou, "Integrated deep learning method for workload and resource prediction in cloud systems," *Neurocomputing*, vol. 424, pp. 35–48, Feb. 2021.
[29] J. Xue, F. Yan, R. Birke, L. Y. Chen, T. Scherer, and E. Smirni, "PRACTISE: Robust prediction of data center time series," in *Proc. 11th Int. Conf. Netw. Service Manage. (CNSM)*, Nov. 2015, pp. 126–134.
[30] C. N. Nhu and M. Park, "Optimizing resource scaling in network slicing," in *Proc. Int. Conf. Inf. Netw. (ICOIN)*, Jan. 2022, pp. 413–416.
[31] J. Benesty, J. Chen, Y. Huang, and I. Cohen, "Pearson correlation coefficient," in *Noise Reduction in Speech Processing*, vol. 2. Berlin, Germany: Springer, 2009, pp. 1–4.
[32] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *CoRR*, vol. abs/1409.0473, pp. 1–15, Sep. 2015.
[33] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, "Long short term memory networks for anomaly detection in time series," in *Proc. 23rd Eur. Symp. Artif. Neural Netw., Comput. Intell. Mach. Learn.*, Bruges, Belgium, Apr. 2015, pp. 89–94.
[34] *Influx Database*. Accessed: 2021. [Online]. Available: https://www.influxdata.com/
[35] A. Savitzky and M. J. E. Golay, "Smoothing and differentiation of data by simplified least squares procedures," *Anal. Chem.*, vol. 36, no. 8, pp. 1627–1639, Jul. 1964.
[36] *Materna Dataset*. Accessed: 2021. [Online]. Available: http://gwa.ewi.tudelft.nl/datasets/gwa-t-13-materna/
[37] M. B. Kursa and W. R. Rudnicki, "Feature selection with the Boruta package," *J. Stat. Softw.*, vol. 36, no. 11, pp. 1–13, 2010.
[38] A. Esmaeily, K. Kralevska, and D. Gligoroski, "A cloud-based SDN/NFV testbed for End-to-End network slicing in 4G/5G," in *Proc. 6th IEEE Conf. Netw. Softw. (NetSoft)*, Jun. 2020, pp. 29–35.
[39] *Keras Library*. Accessed: 2021. [Online]. Available: https://keras.io/
[40] *OpenAirInterface*. Accessed: 2021. [Online]. Available: https://github.com/OPENAIRINTERFACE/openair-epc-fed/tree/master/docker-compose/magma-mme-demo
[41] *Apache JMeter Tool*. Accessed: 2021. [Online]. Available: https://jmeter.apache.org/
[42] M. Dieye, S. Ahvar, J. Sahoo, E. Ahvar, R. Glitho, H. Elbiaze, and N. Crespi, "CPVNF: Cost-efficient proactive VNF placement and chaining for value-added services in content delivery networks," *IEEE Trans. Netw. Service Manage.*, vol. 15, no. 2, pp. 774–786, Jun. 2018.

CHIEN-NGUYEN NHU received the B.S. degree in control automation engineering from the Hanoi University of Science and Technology, Vietnam, in 2018. He is currently pursuing the M.S. degree in information and communication with Soongsil University, Seoul, South Korea. His current research interests include machine learning, deep learning, network security, network functions virtualization, and cloud computing.

MINHO PARK (Member, IEEE) received the B.S. and M.S. degrees in electronics engineering from Korea University, in 2000 and 2002, respectively, and the Ph.D. degree from the School of Electrical Engineering and Computer Science, Seoul National University, Seoul, South Korea, in 2010. He is currently an Associate Professor with the School of Electronic Engineering, Soongsil University, Seoul. His current research interests include wireless networks, vehicular communication networks, network security, and cloud computing.