LSTM-Based VNF Resource Usage Forecasting Leveraging SFC Resource Attribute Interdependencies

by

Cédric ST-ONGE

THESIS PRESENTED TO ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
IN PARTIAL FULFILLMENT FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
Ph.D.

MONTREAL, FEBRUARY 3, 2023

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

BOARD OF EXAMINERS

THIS THESIS HAS BEEN EVALUATED

BY THE FOLLOWING BOARD OF EXAMINERS

Mrs. Nadjia Kara, Thesis supervisor
Department of Software and IT Engineering, École de Technologie Supérieure

Mrs. Christine Tremblay, Chair, Board of Examiners
Department of Electrical Engineering, École de Technologie Supérieure

Mr. Abdelouahed Gherbi, Member of the Jury
Department of Software and IT Engineering, École de Technologie Supérieure

Mrs. Manar Jammal, External Independent Examiner
School of Information Technology, York University

THIS THESIS WAS PRESENTED AND DEFENDED

IN THE PRESENCE OF A BOARD OF EXAMINERS AND THE PUBLIC

ON JANUARY 12, 2023

AT ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

FOREWORD

> I fully realize that I have not succeeded in answering all of your questions... Indeed, I feel I have not answered any of them completely. The answers I have found only serve to raise a whole new set of questions, which only lead to more problems, some of which we weren't even aware were problems. To sum it all up... In some ways I feel we are confused as ever, but I believe we are confused on a higher level, and about more important things.
>
> *-Earl C. Kelley*

The topic of this work is virtual network function (VNF) resource usage forecasting leveraging service function chain (SFC) interdependencies. In the following pages, we present three main research contributions: a taxonomy of VNF dependencies, an LSTM-based VNF resource usage forecasting architecture assisted by an attention mechanism combined with four input feature engineering techniques using graph neural networks (GNNs), Pearson, Spearman's rank and Kendall's Tau correlation coefficients, and a multivariate outlier filtering mechanism based on Adjusted Outlyingness (AO).

ACKNOWLEDGEMENTS

I would like to thank the faculty and staff of the Software and Information Technology Engineering Department at ÉTS. This pandemic has been a trying one for all of us, but without your exceptional gifts of kindness toward your students, your courage and your resilience, I would not be writing these lines today. Patrice, Samuël, Alain, Stéphane, Patrick... you have been invaluable to me.

I also want to thank our partners at Ericsson Canada (Claes Edström) and Rogers Communications (Jean-Yves Bernard) for their financial support and important input, especially in terms of objectives, perspectives, issues, competition and constraints of an industrial research project. This knowledge enriched this project enormously and made me grow as a researcher.

I can't forget the support of my family during all these years. A special mention to my nephew (William) and nieces (Olivia, Florence) who have been a constant inspiration throughout this journey. They have always been there to encourage me and cheer me up, in good times and in bad.

I would also add my peers at the LASI lab with whom I share many wonderful memories. I enjoyed every moment spent with you!

I want to thank Professor Nadjia Kara for her advice, her rigour, the vast experience she has shared throughout the years and her quality of managing large-scale research projects in collaboration with the industry.

Finally, I dedicate this work to Emilie and my daughter Alice. My darlings, all the love I have for you is in this document. You are my motivation, my inspiration, my sunshine... I could not have achieved this without you and your constant support. I love you.

Prédiction d'utilisation de ressources VNF renforcée par l'interdépendance entre attributs de ressources d'une SFC

Cédric ST-ONGE

RÉSUMÉ

Depuis peu, la virtualisation des fonctions réseau (NFV) est devenue un sujet de recherche qui suscite beaucoup d'intérêt en raison des nombreux avantages qu'elle présente pour les infrastructures de réseaux de nouvelle génération (NGN), comme la 5G, la 6G et l'IoT. D'intenses recherches sont actuellement menées pour concevoir des algorithmes robustes capables de prédire avec précision l'utilisation des ressources de diverses entités dans les infrastructures NFV (NFVI), telles que les fonctions de réseau virtualisées (VNF), les chaînes de fonctions de service (SFC), les nœuds de serveur et les grappes de serveurs, dans le but de faire évoluer ou de migrer correctement ces VNF ou SFC. À ce jour, cependant, très peu d'équipes de recherche ont concentré leurs efforts sur la recherche de dépendances et de relations utiles que ces entités pourraient avoir entre elles. Par exemple, la mise à l'échelle d'une VNF spécifique peut avoir un impact sur l'utilisation des ressources CPU et le trafic réseau d'autres VNF dans le même SFC. Dans d'autres cas, la migration d'une SFC entière d'une grappe de serveurs à une autre peut avoir un impact négatif sur le trafic réseau de certaines tranches de réseau dédiées aux fonctions de contrôle ou de cryptage de cette SFC et d'autres types de SFC similaires. Pour résoudre le problème de l'identification des dépendances de VNF, l'apprentissage profond (DL) et, plus particulièrement, les réseaux neuronaux récurrents (RNN) pourraient s'avérer extrêmement utiles en raison de leur capacité avérée à déchiffrer les relations profondes et cachées entre de multiples caractéristiques dans les séries chronologiques. Par conséquent, l'objectif de ce projet de recherche sera de trouver des moyens d'identifier les dépendances VNF et SFC dans les NFVI et les environnements infonuagiques, et de tirer parti des modèles de dépendance VNF et SFC pour améliorer les méthodes de prévision dynamique de l'utilisation des ressources VNF dans les environnements multi-VNF. Pour ce faire, notre objectif sera 1) d'identifier et de classer les différentes dépendances de VNF, 2) de construire des mécanismes ETL (extraction, transformation, instanciation) pour adapter l'historique séquentiel de l'utilisation des ressources de VNF à partir de multiples caractéristiques pour DL, et enfin, 3) de concevoir un mécanisme de prédiction d'utilisation de ressources VNF en exploitant les interdépendances des attributs de ressources dans une SFC.

Mots-clés: Adaptation de ressources VNF, Prédiction d'utilisation de ressources VNF, inter-dépendances VNF, LSTM, Coefficients de corrélation, Réseaux neuronal en graphe, filtrage de valeurs aberrantes multivariées

LSTM-Based VNF Resource Usage Forecasting Leveraging SFC Resource Attribute Interdependencies

Cédric ST-ONGE

ABSTRACT

As of late, Network Function Virtualization (NFV) has emerged as a research topic garnering a lot of interest, thanks to its many purported benefits to Next-Generation Network (NGN) infrastructures such as 5G, 6G and IoT. Intense research is currently being conducted to design robust algorithms that can accurately predict resource usage of various entities in NFV infrastructures (NFVIs), such as Virtualized Network Functions (VNFs), Service Function Chains (SFCs), server nodes and server clusters, with the aim to scale or migrate those VNFs or SFCs properly. To date, however, few research teams have focused their efforts on finding useful dependencies and relationships that these entities might have with each other. For example, scaling a specific VNF might impact the CPU resource usage and network traffic of other VNFs in the same SFC. In another case, migrating a whole SFC from one server cluster to another might negatively impact the network traffic of some network slices dedicated to control or encryption functions of that SFC and other similar SFC types. To tackle the issue of identifying VNF dependencies, Deep Learning (DL) and, more particularly, Recurrent Neural Networks (RNNs) might prove extremely useful due to their proven ability to decipher deep, hidden relationships between multiple features in time series. Hence, the aim of this research project will be to find means of identifying VNF and SFC dependencies in NFVIs and cloud computing environments and to leverage VNF and SFC dependency models to enhance dynamic VNF resource usage forecasting methods in multi-VNF environments. To do so, our goal will be 1) to identify and classify different VNF dependencies, 2) build ETL (Extract, Transform, Load) mechanisms to adapt sequential VNF resource usage history from multiple features for DL, and finally, 3) design a VNF resource usage forecasting mechanism leveraging resource attribute interdependencies in an SFC.

Keywords: VNF Resource Adaptation, VNF Resource Usage Forecasting, VNF interdependencies, LSTM, Correlation Coefficients, Graph Neural Networks, Adjusted Outlyingness

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

LIST OF ALGORITHMS

Page

LIST OF ABBREVIATIONS

AA	Affinity Aggregate
AIC	Aikake Information Criterion
AO	Adjusted Outlyingness
AR	Auto-regression
ARIMA	Auto-Regressive Integrated Moving Average
BIC	Bayesian Information Criterion
CAPEX	Capital Expenditures
CSP	Cloud Service Provider
CAT-LSTM	Context and Aspect Embedded Attentive Target Dependent LSTM
DC	Datacenter
DL	Deep Learning
DT	Decision Tree
FC	Fully Connected Neural Network
GNN	Graph Neural Network
IMS	IP Multimedia Subsystem
IPS	Intrusion Prevention System
LOF	Local Outlier Factor
LSTM	Long Short-Term Memory
MA	Moving Average

MAE	Mean Absolute Error
MD	Mahalanobis Distance
ML	Machine Learning
NF	Network Function
NFV	Network Function Virtualization
NFVI	Network Function Virtualization Infrastructure
NGN	Next-Generation Network
NMdSE	Normalized Median Squared Error
NMSE	Normalized Mean Squared Error
MAE	Mean Absolute Error
MANO	Management and Orchestration
MAPE	Mean Absolute Percentage Error
MdAPE	Median Absolute Percentage Error
MDP	Markov Decision Process
MSE	Mean Squared Error
OPEX	Operational Expenditures
QoS	Quality of Service
RMSE	Root Mean Squared Error
RNN	Recurrent Neural Network
SFC	Service Function Chain

SLA	Service Level Agreement
SLO	Service Level Objective
SVM	Support Vector Machine
SVR	Support Vector Regression
TSP	Telecom Service Provider
VNF	Virtual Network Function
VNFC	Virtual Network Function Component
VNF-FG	Virtual Network Function Forwarding Graph
VNF-RA	Virtual Network Function Resource Allocation
WA	Weighted Average

LIST OF SYMBOLS AND UNITS OF MEASUREMENTS

$c(t_y)$	Attention context vector
$c^{\langle t \rangle}$	LSTM long-term state
$e(t_x, t_y)$	Alignment scores
$i^{\langle t \rangle}$	LSTM input state i
f_n	Resource attribute feature f from VNF n. I.e.: CPU_1
$f^{\langle t \rangle}$	LSTM forget state
F_x	Number of input features
F_y	Number of output features
$g^{\langle t \rangle}$	LSTM input state g
h_n	GNN node hidden state
$h(t_x)$	Attention final hidden state
$h^{\langle t \rangle}$	Attention short-term state
mbps	Megabits per second
o_n	GNN output label
$o^{\langle t \rangle}$	LSTM output state
r_s	Spearman's rank correlation coefficient
$s(t_y)$	Attention current hidden state
T_x	Number of input timesteps
T_y	Number of output timesteps

V_{θ}^{ρ}	Data point matrix
$x(i)$	Input sample matrix
$y(i)$	Output label matrix
$y^{\langle t \rangle}$	LSTM cell output
$\hat{y}_{[F_y]}^{\langle T_y \rangle}$	Resource usage prediction matrix
$\hat{y}^{\langle t_y \rangle}$	Softmax function
$\alpha(t_x, t_y)$	Attention weights
ρ	A VNF
$\rho_{X,Y}$	Pearson correlation coefficient
σ	Sigmoid activation function
θ	A resource attribute
τ	Kendall's rank correlation coefficient

INTRODUCTION

0.1 Context and Motivation

Network Function Virtualization (NFV) is a promising technology seeking to transform the way network operators architect networks by evolving standard IT virtualization technology, contributing to consolidating many network equipment types onto industry-standard high-volume servers, switches and storage (ETSI, 2013). In the years to come, NFV will provide new benefits unheard of in legacy Network Function (NF) hardware such as flexible on-demand scaling, redundancy and lower total costs of operation (Gupta *et al.*, 2017). These benefits are desirable for Cloud Service Providers (CSPs) and Telecommunications Service Providers (TSPs) because adopting this technology will enable competitive pricing, better points of presence, the flexibility of scaling and avoiding single points of failure by tapping on NFV's ability to scale and migrate VNFs on-demand across an NFV Infrastructure (NFVI). However, in order to trigger those expected results, the research community also needs to address key issues of NFV like adaptation to resource capacity (Blanco *et al.*, 2017) and performance like latency and throughput (Gupta *et al.*, 2017; Fahmy & Saxena, 2017). A sound option to mitigate these issues is by designing VNF resource adaptation mechanisms meeting service providers' management policies such as CAPEX, OPEX, Service Level Objectives (SLO), Service Level Agreements (SLA) and Quality of Service (QoS) requirements as well as adapting to fluctuating demand on their cloud environment's physical resources (e.g., performance, reliability, availability, etc.). Hence, researchers are currently scrambling to find new ways to automate VNF resource adaptation mechanisms adhering to those all-the-more complex optimization objectives.

However, following those management policies and automating VNF resource adaptation in a dynamic cloud environment is indeed a challenging task since, for instance, a VNF can either be deployed as a single entity or can be connected to other VNFs in a Service Function Chain (SFC). Because of this chaining, there are a tremendous amount of interdependencies between

the resources, jobs, tasks and services operating inside a cloud environment, an NFVI, an SFC and even a VNF that could influence VNF scaling or migration decisions.

Untangling all those interdependencies looks like an excellent starting point to enable context awareness and therefore optimize resource adaptation mechanisms in an infrastructure. Much like knowing how many concurrent end-users using a network resource can help us predict the future load of that resource, it would be interesting to know if the resource usage history of certain resource attributes of an SFC could help us respect management policies more efficiently by more accurately predicting how resource needs will shift over time. Therefore, a thorough analysis of those interdependencies is relevant to understand better what is at stake when dynamically adapting VNF resources.

In the same vein, Deep Learning (DL) has demonstrated in the last decade how it can leverage deep relationships in several input features and provide accurate predictions for complex problems. VNF resource adaptation mechanisms leveraging interdependencies from several resource attributes of an SFC could therefore significantly benefit from DL's advantages over traditional, linear machine learning (ML) techniques like auto-regression (AR), moving average (MA) and auto-regressive integrated moving average (ARIMA).

Therefore, the motivation in doing this work is to make a thorough investigation of low-level to high-level VNF interdependencies, find how those interdependencies can help enable context awareness of VNF adaptation mechanisms by anticipating future actions (scaling and migrating VNFs) to be orchestrated in the NFVI, and to investigate how DL architectures can enhance those mechanisms by leveraging VNF interdependencies.

0.2 Problem statement

Developing DL models to any applied domain is already extremely complex due to the high expertise required in both DL and the specific domain of application, but applying DL to

cutting-edge, highly dynamic, and business-critical infrastructures such as carrier-grade telecom and cloud infrastructures is an extremely challenging task and an endeavour that is not to be taken lightly, to say the least.

First, DL model training requires huge quantities and a variety of data. Hence, it requires the ability to collect live data from the infrastructure from which we aim to deploy those models. In NFV and cloud infrastructures, that data-collection ability is highly dependent on monitoring tools with custom-made configurations whose purpose is primarily to monitor the state of the infrastructure to either find any sign of system failure or to improve systems performance. The metrics collected are therefore typically aimed solely at advising telecom service operators (TSPs) and cloud service operators (CSPs) of a system failure. TSPs and CSPs then act manually to apply the correct course of action to repair, adapt to or dismiss that system failure. DL practitioners thus need to convince TSPs and CSPs to adapt their monitoring tools for more invasive, continuous monitoring and data collection, and explain to them the value of doing so. This is especially hard to accomplish when DL practitioners ask TSPs and CSPs to significantly increase the traffic overhead in the network with nothing to show in terms of network operational management enhancements, OPEX reduction, QoS improvement, or any tangible value whatsoever.

Another major aspect to consider for aspiring DL-based solution providers is data availability and data quality. Time-stamped features are the holy grail of machine learning (ML)-based solutions operating in real-time environments. Yet, those features are resource-heavy to collect and will take a considerable amount of storage over time when it comes to data archival to train new models. Unfortunately, time-stamped features are hard to come by in NFVIs, and collection of that type of data must be carefully thought out to ensure that it does not strain the storage capacity, processing power and bandwidth of the control plane. Hence, we must strike a fine balance to ensure data quality and quantity. Data collection should only be triggered in

very specific instances, in specific areas of the network and with only the most useful features available to make an optimal decision with DL-based prediction models. A final constraint is to ensure that each feature is properly time-stamped and that those time stamps are synchronized throughout the nodes and links of the infrastructure.

A third important aspect is considering the best ways to measure the accuracy and efficiency of the predictions from DL models. DL models are notorious for their low explainability, which is a major setback for artificial intelligence (AI) adoption in many industries (e.g., in fintech, emergency response, autonomous driving), since there is no easy way to address mistakes, biases, etc. In VNF resource adaptation, for instance, we should be able to explain why a model proposes to scale or migrate resources, and why this is the best course of action. To do so, most approaches will rely on the F1 score. However, it is hard to properly evaluate the consequences of false positives and true negatives on the infrastructure with such metrics. In this regard, regression problems rather than classification problems should be privileged, because it is not only easier to explain prediction accuracy by using metrics such as MSE, RMSE and MAPE, but we can also easily adjust automated resource adaptation mechanisms based on those metrics.

Finally, tying it all up is the aspect that DL approaches are data-driven. This explains why it is extremely difficult for DL to make a breakthrough in the NFV industry, for the reason that it requires that DL practitioners have vast expertise in NFV, that they deeply understand what can be achieved with the provided data, what relationships can be made between the different features, correlations to be made in the data, how to filter the data, etc. Data-driven design is actually the most challenging aspect of applied AI in any industry since it is the data that is at the very core and that will determine the whole trajectory of a research project.

That is why it is of utmost importance, in any research project involving applied AI, to have a deep understanding of the research domain where it is to be applied and also have a deep knowledge of what we can extrapolate from potential features in the data that can be collected.

Therefore, investigating VNF resource usage interdependencies sounds like an excellent starting point. Afterwards, it will make the task of deciphering the data provided by industrial partners much easier before we commit to a course of action and propose a DL-based VNF resource usage adaptation mechanism.

0.3 Objectives

The ultimate challenge of this work is to validate the hypothesis that VNF interdependencies improve VNF resource usage prediction. To date, very few applied state-of-the-art approaches have tackled resource usage prediction from multivariate data and none have tackled that challenge from the angle of interdependencies since this requires deep knowledge in both the applied research domain and in multivariate, many-to-many prediction-based DL techniques. As a matter of fact, current DL-based resource usage forecasting techniques rely on one-to-one mapping of features, i.e., the forecast output features rely on the usage history of those same features. Hence, it is difficult to confirm that interdependencies have a beneficial influence on prediction accuracy.

Henceforth, we aim to investigate, design, test and validate VNF resource usage prediction mechanisms based on recurrent neural networks (RNNs) and assess their relevance against state-of-the-art approaches. This will be an iterative process where we will seek to improve upon different accuracy measures of the previous models until we reach the point where we have the most polished and elegant approach possible, using the latest techniques available.

Since this project is made in collaboration with and received precious input from global Information and Communication Technology (ICT) partners, we seek to provide telecom service providers (TSPs) and cloud service providers (CSPs) with a generic, self-contained RNN design framework for predicting VNF and SFC resource utilization in a virtualized infrastructure. RNN was chosen because of its demonstrated ability to improve prediction accuracy by exploiting

deep relationships (or, as we call them in this work, interdependencies) between several highly correlated resource attributes such as CPU utilization rate, RAM utilization, disk I/O, I/O bandwidth utilization between VNF nodes, etc.

Simply put, the objectives of this work can be summarized as follows:

1. Investigate, analyze and define a complete taxonomy of dependencies and, more accurately, of interdependencies, affinities and anti-affinities tied to all physical, virtual and network stacks (i.e., TCP/IP protocol stack, jobs, tasks, services and resource attributes of the physical and virtual infrastructure) of cloud computing, microservice and NFV infrastructures.
2. Investigate, design, test and validate novel pre-processing techniques to properly filter and adapt multivariate resource usage data history from large datasets prior to its utilization for training and validation of Deep Learning (DL) models.
3. Investigate, design, test and validate a cutting-edge RNN-based resource usage prediction architecture leveraging interdependencies in cloud computing environments.

Moreover, those mechanisms will have to meet the following functional requirements:

- *Generic areas of applications:* The proposed architecture can be ported to different cloud computing areas (e.g., heterogeneous cloud environments, microservice architecture, NFV, carrier-grade infrastructures, IoT environments).
- *Generic inputs and outputs:* The proposed architecture can use and forecast any type of numeric data (e.g., CPU resource usage, memory usage, disk I/O, bandwidth usage, latency, write/read per second). Models can also be set to take any number of input features and time steps, and to predict independently any number of output features and time steps.
- *Self-contained:* The models are autonomous and require no human intervention to trigger either the resource usage history collection and storage, or the resource usage prediction mechanism.

- *Fast execution:* Execution time is key in next-generation networks (NGN) such as 5G and up. This requirement is set to mitigate bottleneck risks in the control plane.
- *Small footprint:* Prediction models must have a small storage size and use minimal bandwidth in the control plane of the infrastructure to optimize power, storage and bandwidth usage to the bare minimum.

0.4 Methodology

All contributions proposed in this work followed the processes depicted in our methodology framework, in Figure 0.1. The progress for each contribution was iterative and we regularly moved back and forth from one objective to another to ensure that the packaged solution was cohesive, robust and well-tied together. The process consisted of succinctly conducting a literature review, and a problem formulation followed by its modelling by mathematical, conceptual or architectural means before being solved using various techniques (input feature engineering, outlier filtering, grid search, and testing with different DL architectures). The validation of the proposed solutions was carried out using different scenarios and compared with the closest state-of-the-art techniques. The evaluation was conducted following the same settings similar to most state-of-the-art methods and the inputs provided by our industrial partners with synthetic tools (e.g., autoregression, moving average, ARIMA forecasting using libraries in Python 3.x).

Foremost, the first objective we tackled was to study VNF dependencies and make a draft of the taxonomy of VNF interdependencies. That first step was imperative to ensure that we got a good picture of what type of features to expect in the data collected and identify which features could potentially have some value in terms of interdependencies and affinity. It was also crucial to orient our research early on and set our expectations by helping us get a strong sense of what could be or could not be achieved in that particular research domain.

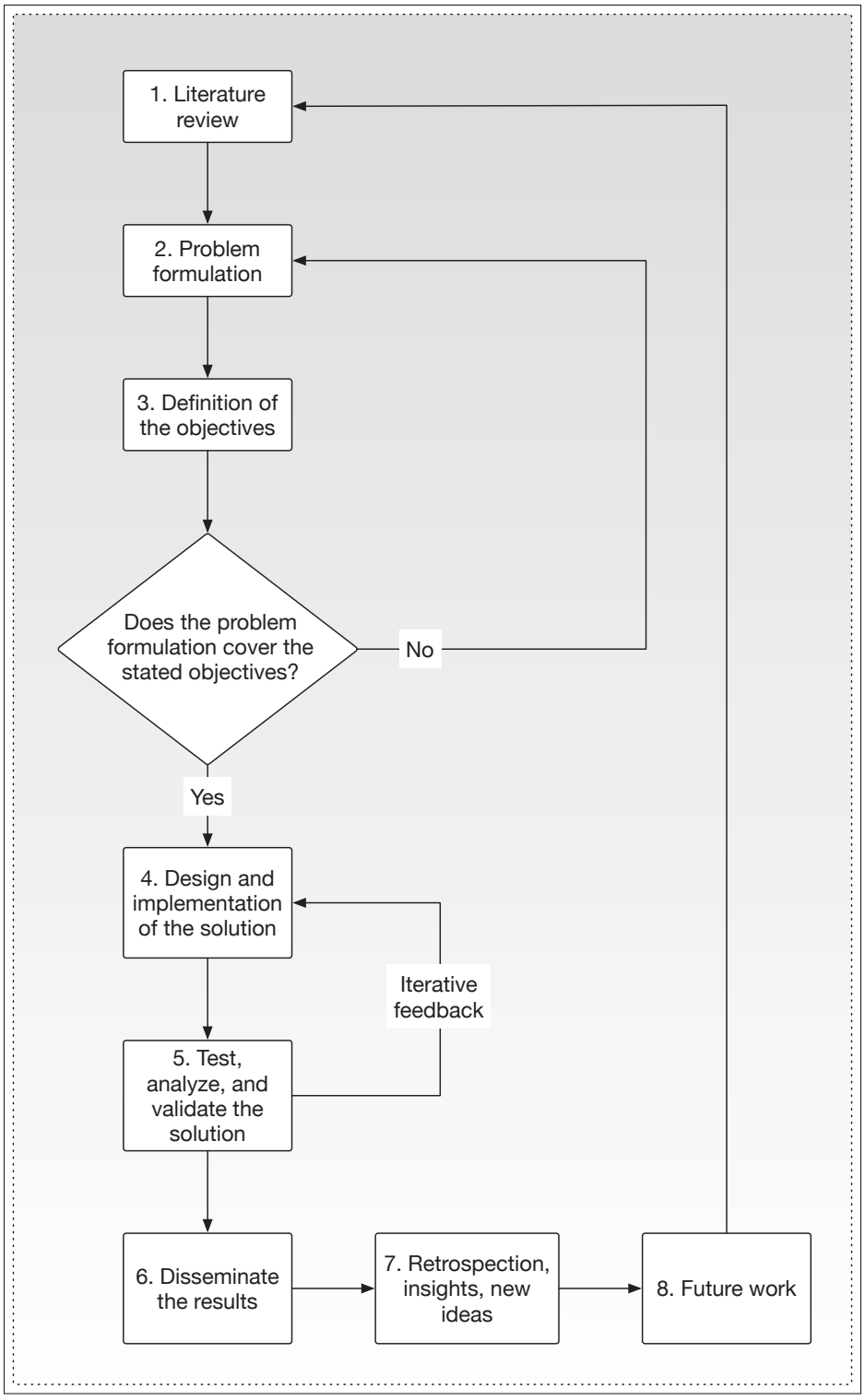


Figure 0.1 Our methodology framework

After obtaining the needed data sets from our industrial partners at Ericsson Canada (see Appendix I) to conduct our experiments, we proceeded with the other objectives by following the same processes. Interestingly, our experiments using DL techniques greatly contributed to cementing our claims in our taxonomy of VNF dependencies, which led to continuous refinements and improvements in the resulting contributions.

0.5 Contributions

Our motivation for doing this work was to demonstrate that state-of-the-art AI solutions can realistically be deployed in carrier-grade telecom infrastructures. In order to prove this claim, we designed NFVLearn, a complete solution addressing today's biggest challenges in the emergent NFV research domain. An overview of our proposed solution is shown on Figure 0.2.

NFVLearn is a software solution intended to be developed by software and telecom vendors, but could be operated by scientists working at the said vendor (provided the service provider - the customer - provides the relevant resource usage data) or by system administrators, scientists working for the service provider (provided the customer has the appropriate on-site machine learning infrastructure). It is completely VNF and SFC agnostic, meaning that it can be applied to any VNF, without prior knowledge of the function, software, vendor, or underlying physical infrastructure in which it operates.

The resource utilization forecasts of these VNFs (which in this work are predicted in the medium to long term) can then be used by the resource adaptation mechanisms in the infrastructure to adapt and/or migrate the VNFs in order to "balance" the load of the physical resources in a data center, a point of presence (PoP), an edge network, etc.

This work is therefore a contribution that proposes a new technique to build LSTM-based resource utilization prediction models for cloud and telecom service providers. The medium- and long-term resource utilization predictions from our models can then be used by VNF

resource adaptation approaches, which then benefit from accurate, multivariate, and multi-enterprise medium- and long-term resource utilization prediction enhanced by resource utilization interdependencies.

Our main contributions can be summarized as follows:

1. A taxonomy of cloud computing, VNF and microservice dependencies.
2. NFVLearn, a multivariate, many-to-many LSTM-based VNF resource usage forecasting approach leveraging VNF resource attribute interdependencies in an SFC, packaged with:
    a. Four novel input feature selection mechanisms leveraging the following techniques:
        1. Graph Neural Network
        2. Pearson correlation coefficient
        3. Spearman's rank correlation coefficient
        4. Kendall's tau correlation coefficient
    b. An extra attention mechanism that significantly enhances forecasting accuracy and model training time. The enhanced approach is known as A-NFVLearn.
3. A novel outlier filtering mechanism using the Adjusted Outlyingness (AO) technique

Aside from the taxonomy of dependencies in cloud environments presented in Chapter 2, two peer-reviewed journal papers were accepted for publication in high-ranking journals. The following publications appear in the chronological order of their submission:

- St-Onge, C., Kara, N., & Edstrom, C. (2022). *"NFVLearn: A Multi-Resource, LSTM-Based VNF Resource Usage Prediction Architecture"*, accepted for publication in "Journal of Software: Practice and Experience, Wiley", October 2022.
- St-Onge, C., Kara, N., & Edstrom, C. (2022). *"Multivariate outlier filtering for A-NFVLearn: an Advanced Deep VNF Resource Usage Forecasting Technique"*, accepted with revisions at "The Journal of Supercomputing, Springer", November 2022.

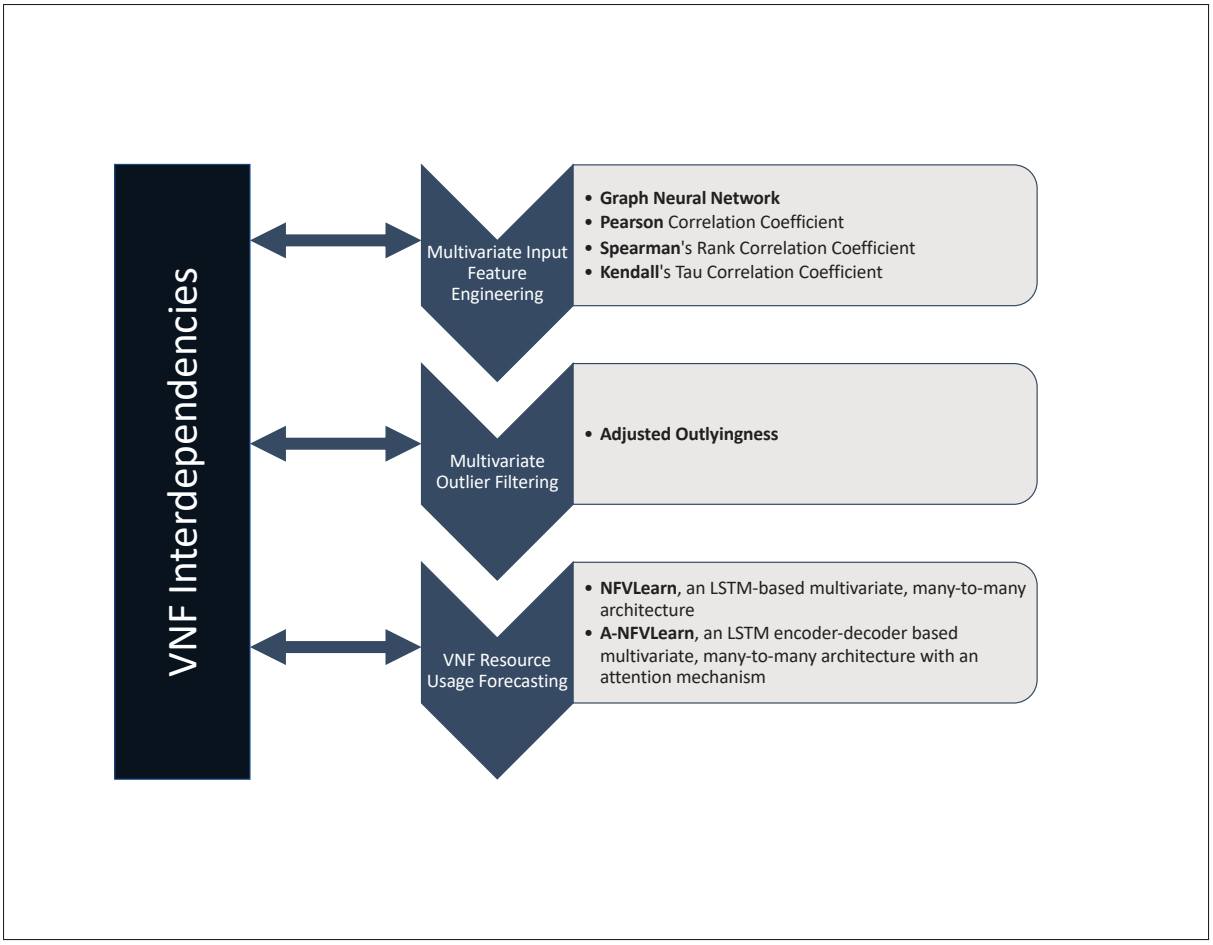


Figure 0.2 NFVLearn: an end-to-end solution

0.6 Thesis Organization

The thesis is organized as follows. We present a general literature review in Chapter 1. Chapters 2, 3, and 4 detail our contributions; Chapter 2 presents our taxonomy of VNF dependencies, Chapter 3 is dedicated to NFVLearn and our input feature selection algorithms, and Chapter 4 discusses A-NFVLearn and our outlier filtering mechanism based on AO. In Chapter 5, we offer a discussion about our journey and share some insights about applied AI in the industry of Information and Communication Technology (ICT). Lastly, we conclude the thesis and outline future directions from the remaining open research opportunities that this work opened up.

CHAPTER 1

LITERATURE REVIEW

This chapter presents a literature review covering workload forecasting methods in cloud computing, automated input feature selection techniques and outlier filtering techniques to establish the perspective of our contributions. This chapter is divided in three sections centering on specific contributions presented in chapters 3 and 4 of this thesis. Section 1.1 focuses on studies relevant to NFVLearn and our input feature selection techniques presented in chapter 3. Next, section 1.2 presents work related to A-NFVLearn and our multivariate outlier filtering technique presented in chapter 4. Then, section 1.3 comments broadly on state-of-the-art AI- and ML-based univariate and multivariate forecasting methods in cloud computing and, finally, section 1.4 positions our work inside the state-of-the-art.

1.1 Work related to NFVLearn and Input Feature Selection

This section presents the literature review relevant to two aspects of this study: ML-based VNF resource usage prediction and automated input feature selection techniques.

1.1.1 ML-based VNF Resource Usage Prediction

Proposal of numerous NFV and mobile networking management mechanisms leveraging state-of-the-art machine learning techniques has garnered huge interest in recent years throughout the research community (Chakraborty, Corici & Magedanz, 2020; Cho, Jang & Pack, 2020; Eramo, Lavacca, Catena & Salazar, 2021, 2020; Hirayama, Jibiki & Kafle, 2020; Hirayama, Miyazawa, Jibiki & Kafle, 2021; Kim *et al.*, 2019; Lange *et al.*, 2021; Mijumbi *et al.*, 2014, 2017; Moradi, Ahmadi & Nikbazm, 2022; Patel, Verma & Misra, 2019; Wang, Bi, Wu, Vasilakos & Fan, 2019). Despite this enthusiasm, deep knowledge of several aspects of NFV as a whole (MANO, services, overhead reduction, latency minimization, etc.) and of the potential, but also of the limitations of machine learning techniques are crucial to designing robust, simple and valuable solutions to the administrators of NFVIs (Boutaba, Shahriar, Salahuddin & Limam, 2021; Zhang,

Patras & Haddadi, 2019a). Our multivariate, many-to-many LSTM-based VNF resource load prediction approach stands out from other solutions by collecting many time steps of resource usage history of different resource attributes (CPU, memory, I/O bandwidth), from various VNFs in an SFC, in order to leverage their interdependencies. Therefore, the resulting resource usage forecasts of many time steps of different resource attributes from an SFC provide high-accuracy, high-fidelity predictions since they benefit from variations in resource loads of multiple resource attributes that are highly correlated to those targeted resource attributes that we aim to forecast.

For instance, some traditional time series-based prediction approaches like Auto-Regression Integrated Moving Average (ARIMA) enhanced with trend and seasonality filtering (Chakraborty *et al.*, 2020), Decision Tree (DT) and Support Vector Regression (SVR) (Moradi *et al.*, 2022), Markov Decision Process (MDP) (Wang *et al.*, 2019), and softmax regression (Mijumbi *et al.*, 2014) are still popular because of their effectiveness, swiftness and overall simplicity at predicting future resource usage of non-stationary, univariate resource attributes like CPU, memory, disk I/O, or network traffic. However, since these approaches leverage univariate (i.e., take historical data from a single resource attribute to predict a resource usage horizon of that same attribute) resource usage history to forecast a resource usage horizon, none of them can reinforce their predictions from resource usage interdependencies from multiple correlated sources.

Further, with the massive increase in data availability and the rise of DL and its ability to decipher complex relationships between different features of the data, more elaborate solutions are gradually appearing in the literature. Approaches leveraging RNNs (Hirayama *et al.*, 2020, 2021; Zhang, Wang & Luo, 2020), LSTM (Cho *et al.*, 2020; Kim *et al.*, 2019; Eramo *et al.*, 2020, 2021; Patel *et al.*, 2019; Shah, Yuan, Lu & Zerfos, 2017), and Context and Aspect Embedded Attentive Target Dependent LSTM (CAT-LSTM) (Cho *et al.*, 2020; Kim *et al.*, 2019; Lange *et al.*, 2021) have been particularly appealing in recent years.

Among RNN-based approaches, the authors in (Hirayama *et al.*, 2020, 2021) propose a traffic prediction framework that uses RNN, Random Forest and Elastic Net univariate, many-to-many mechanisms executed on the controller and resource managers to predict vCPU usage from a

VNF. They evaluate the fidelity of their predictions to the observed values with R^2 score. In (Zhang *et al.*, 2020), the univariate solution generates a non-stationary sequence prediction from inputs filtered using moving windows with exponential decay. The model evaluation is based on several criteria: Normalized Mean Squared Error (NMSE), Mean Absolute Percentage Error (MAPE), Normalized Median Squared Error (NMdSE) and Median Absolute Percentage Error (MdAPE). Again, these ML techniques (Hirayama *et al.*, 2020, 2021; Zhang *et al.*, 2020) that depend on univariate resource usage history have the same limitations as those stated previously: the predictions of these models are based on same-to-same resource usage attributes. Thus, those techniques do not benefit from historical resource usage interdependencies of multiple highly correlated resource usage attributes.

Next, among the LSTM and CAT-LSTM solutions, many of those use multivariate, many-to-one mechanisms (Cho *et al.*, 2020; Kim *et al.*, 2019; Eramo *et al.*, 2020, 2021; Lange *et al.*, 2021; Patel *et al.*, 2019). In (Cho *et al.*, 2020), the approach takes historical resource utilization of VNFs to predict future CPU loads of a VNF and its neighbours and the authors use RMSE as a loss function to evaluate results. On the other hand, the approach proposed in (Kim *et al.*, 2019) compares RSME loss of prediction models that maximize the benefits of using an SFC. In this work, the authors first propose a method to combine VNFs in a VNF forwarding graph (VNF-FG) into a directed graph, then a method to build models that predict one time step of a VNF CPU resource usage from the input of multiple CPU resource usage time steps of multiple VNFs. Other solutions (Eramo *et al.*, 2020, 2021) use LSTM with a novel asymmetric Mean Squared Error (MSE) loss function, which weighs the positive and negative prediction errors and the corresponding over-provisioning and under-provisioning costs. These approaches take bandwidth usage from all links of a VNF-FG to predict future bandwidth needs from that VNF-FG for VNF resource allocation (VNF-RA) purposes. The authors in (Lange *et al.*, 2021) propose an approach in the context of NFV network management where they use 20 seconds of historical data sampled every 5 seconds of the CPU utilization and TX / RX byte counters of all VNFs to forecast the CPU utilization of an Intrusion Prevention System (IPS) instance. The performance of their approach is evaluated with four different criteria: MSE, Mean Absolute

Error (MAE), MAPE and RMSE. In (Patel *et al.*, 2019), Patel et al. propose a proactive approach for predicting the auto-scaling of VNFs ahead of time in response to dynamic traffic variations. The approach takes different traffic features as input, then predicts the CPU resources utilized by upcoming VNF instances, and the authors use Probability Density Functions of the errors to evaluate their approach. While these approaches benefit from multivariate resource usage inputs such as CPU from multiple VNFs (Cho *et al.*, 2020; Kim *et al.*, 2019), bandwidth usage of VNF links (Eramo *et al.*, 2020, 2021), CPU and TX / RX byte counters of all VNFs (Lange *et al.*, 2021), and traffic features of VNF links (Patel *et al.*, 2019), all of them are either regression or classification problems involving a single output. Although these approaches are valuable for quick, proactive decision-making, their generated decisions have limitations compared to the depth provided by an approach with multiple output time steps such as NFVLearn. A multi-output time step approach enables flexible decision-making that can take into account a broader range of scaling decisions that would otherwise require more seconds to activate (such as migration of a VNF), and that could be more beneficial in terms of system stability, robustness and load balancing in the NFVI in the long term, for instance.

Finally, two solutions (Shah *et al.*, 2017; Mijumbi *et al.*, 2017) propose some of the few multivariate, many-to-many solutions to date. Shah et al.'s (Shah *et al.*, 2017) innovative LSTM-based approach uses a feature learning function to generate features that represent dependencies in the data. It ingests performance measurements from monitoring systems collected at various layers of the cloud stack (infrastructure/platform/application), and the output of the network is the value forecast of target variables for which the user wants to discover dependencies (CPU, packets received, packets transmitted, latency, throughput, etc.). The authors evaluate their models with MSE and MAPE. This approach is of great interest since it analyzes the interdependencies of several resource attributes of a cloud infrastructure to reinforce resource usage forecasting of LSTM models. Its scope, however, is somewhat limited compared to the aims of NFVLearn, which is not only to discover interdependencies between several correlated resource attributes but also to forecast the future resource usage of several attributes, all in a single mechanism. In Mijumbi et al. (Mijumbi *et al.*, 2017), the authors solve the VNF-RA

problem with a Feed-Forward Neural Network (FNN) and a GNN that collect 20 time steps of CPU, memory latency and call drops from an SFC in order to predict 20 time steps of future resource requirements. Their approach takes advantage of the topology of the VNFs to predict the resource requirements of each virtual network function component (VNFC) and to use this prediction to scale in/out in time to have the resource available when they are needed. The evaluation criterion of their approach is MAPE. Their approach aims to predict resource usage of a single resource attribute (a resource requirement), while NFVLearn aims to predict many resource attributes, which may be of different resource types, of a VNF over several time steps.

1.1.2 Automated Input Feature Selection Techniques

Some useful techniques to select input data features prior to ML model training and then remove uncorrelated data while avoiding reducing the models' prediction accuracy can be found in the literature. One of those is applying a GNN to a directed graph (Li, Zemel, Brockschmidt & Tarlow, 2016; Mijumbi *et al.*, 2017; Moradi *et al.*, 2022; Zhou *et al.*, 2020) and the other is feature pruning using correlation coefficients (Zhang, Wang & Gao, 2019b; Betken, Dehling, Nüßgen & Schnurr, 2021; Antwi, Gyamfi, Kyei, Gill & Adam, 2021; Zhang *et al.*, 2018).

GNN theory (Zhou *et al.*, 2020) is a popular choice among teams designing ML-based solutions in which directed graphs are involved (Li *et al.*, 2016). This is especially true for the NFV research area, since concepts of nodes (VNFs), features (resource load), and links between those nodes through a directed graph (SFC) are a direct translation of the underlying concepts of the NFVI. For instance, authors in (Mijumbi *et al.*, 2017; Moradi *et al.*, 2022) apply this theory to harness resource load relationships between a VNF and its neighbours ($k = 1$ hop) to enhance their predictions.

The other approach of interest involves applying correlation coefficients (Pearson correlation coefficient, Spearman rank correlation and Kendall rank correlation coefficient) to the input features of an ML model, then eliminating the input features with the lowest correlation

coefficients since uncorrelated features will not improve the prediction accuracy of said models. Researchers in (Antwi *et al.*, 2021) apply this concept to commodity price component determination, while the solution in (Betken *et al.*, 2021) applies in the context of a multivariate dependence framework. In an approach closer to our research area, the authors in (Zhang *et al.*, 2018) use correlation coefficients to select highly correlated input features similar to those of the observed feature to train their LSTM mechanism. Their approach analyzes time series data from industrial IoT equipment and forecasts its operation status.

1.2 Work Related to A-NFVLearn and Outlier Filtering

This section presents a literature review relevant to two aspects of this study: LSTM-based resource usage prediction and outlier filtering at pre-processing of RNN-based models.

1.2.1 Automated Learning and VNF Resource Usage Forecasting

Resource usage forecasting garners huge interest from the mobile, edge, cloud and NFV research communities. For instance, authors in (Zhang *et al.*, 2019a; Boutaba *et al.*, 2021) note how challenging it is for prospective research teams to design robust, simple and practical resource usage forecasting mechanisms for cloud computing and NFVIs. The main reasons are twofold: the first being that it requires a vast knowledge of several particularities and objectives of NFV and cloud computing resource usage prediction (delays in VNF placement, resource adaptation in the server nodes, QoS, SLAs, overhead reduction in the control plane, latency minimization, etc.), and the second one being that it requires a deep understanding of the potential and limitations of LSTM learning techniques. Research teams, therefore, require to gather cross-cutting skills in two large research areas. However, several state-of-the-art LSTM architectures have been proposed as of late that successfully enable automated NFV, cloud and mobile networking management and orchestration (Zhang *et al.*, 2018; Eramo *et al.*, 2020, 2021; Patel *et al.*, 2019; Cho *et al.*, 2020; Lange *et al.*, 2021; Kim *et al.*, 2019).

Several approaches leverage LSTM (Hochreiter & Schmidhuber, 1997). For instance, the authors in (Zhang *et al.*, 2018) propose a method for analyzing the time series correlation of IoT equipment working conditions based on univariate sensor data that leverages an LSTM-based prediction model for working status forecasting. Results showed that their approach achieves less RMSE than that of an ARIMA model. In another example, Patel & al. (Patel *et al.*, 2019) propose a proactive approach using LSTM-based deep learning for predicting the auto-scaling of VNFs ahead of time in response to dynamic traffic variations. Their LSTM model inputs CPU and bandwidth capacity as inputs to then forecast CPU resources utilized by upcoming VNF instances. It then uses a distributed VNF provisioning algorithm to produce scaling decisions ahead of time. Approaches in (Eramo *et al.*, 2020, 2021) propose VNF-FG bandwidth forecasting algorithms for the allocation of resources in NFV environments that can differently weigh the over-provisioning and under-provisioning costs through an asymmetric loss function. Their proposed solutions are interesting since their LSTM models input multivariate network traffic data from all links of a VNF-FG or an SFC to predict future resource needs and help plan resource allocation through the VNF-FG.

A new trend has also emerged in the last few years, where research teams combine LSTM cells with an attention layer (Bahdanau, Cho & Bengio, 2015) with beneficial results. Several approaches in the NFV research domain successfully applied this combination for resource usage forecasting (Kim *et al.*, 2019; Cho *et al.*, 2020; Lange *et al.*, 2021). For instance, authors in (Kim *et al.*, 2019) introduce a VNF resource prediction based on a Content and Aspect Embedded Attentive Target Dependent Long Short Term Memory (CAT-LSTM) model that maximizes the benefits of using an SFC through a directed graph. Their model inputs CPU usage of multiple VNFs over several time steps to predict CPU usage over one time step of a VNF. In (Cho *et al.*, 2020), authors also use CAT-LSTM to predict future CPU resource loads of a VNF by inputting historical CPU resource utilization of that same VNF and its neighbours. Finally, the approach proposed in (Lange *et al.*, 2021) goes further by taking the CPU, memory, input and output bandwidth of all VMs to forecast through a CAT-LSTM model the CPU utilization of an intrusion prevention system (IPS) instance. They use up to 20 seconds of multivariate data

sampled every 5 seconds to predict a resource usage horizon of 5 seconds, thus 1 time step of that IPS.

1.2.2 Outlier Filtering for Data Pre-Processing

Outlier detection and filtering of large data sets are well-established and well-documented. This is an important step at the pre-processing stage, prior to deep learning model training.

However, several of the proposed time series-based outlier filtering techniques are designed for univariate data. This is understandable by the fact that typical time series prediction techniques such as AR, MA, WA), ARIMA, etc., don't leverage multivariate data inter-dependencies and are usually "same-to-same" feature predictions. Among common outlier filtering techniques for univariate data, we find z-score filtering (Khalid *et al.*, 2020), one-class Support Vector Machine (SVM) (Mulerikkal, Thandassery, Rejathalal & Kunnamkody, 2022; Vos *et al.*, 2022; Yu, Wu & Xiong, 2022) and Local Outlier Factor (LOF) (Zhang, Hu & Yang, 2022; Zhang & Zhou, 2022).

Authors in (Khalid *et al.*, 2020) use z-score to pre-process and clean missing data and outliers in an approach that forecasts electricity load and price using Jaya-LSTM in smart grids. Although this approach can predict two different sets of features (electricity load and electricity price), data filtering focuses on univariate data in order to improve prediction accuracy on a single feature. The authors then proceed by comparing the prediction accuracy of a single output feature per model with SVM and univariate LSTM approaches. Z-score is an easy way to estimate outliers in a normal distribution by locating the mean and standard deviation of univariate values of a population.

Approaches in (Mulerikkal *et al.*, 2022; Vos *et al.*, 2022; Yu *et al.*, 2022) use one-class SVM for outlier data detection, then elimination because this approach is simple, efficient and can guarantee the authenticity of data to a certain extent. One-class SVM can greatly improve the precision of anomaly detection in the case of small samples, unbalanced sample classification, and supposes no assumptions about data distribution. For example, the authors in (Mulerikkal

et al., 2022) do so to reduce irregular patterns in metro passenger flow forecasting from an LSTM network. Next, the authors in (Vos *et al.*, 2022) introduce a novel two-step LSTM configuration for removing deterministic components associated with dominant gear signals in the first step (with LSTM regression) and removing the 'residual deterministic' components associated with varying gear signals in the second step (with one-class SVM). They present different LSTM architectures combined with a one-class SVM to separate abnormal data from normal vibration signals collected from non-consecutive time series of helicopter test flight data. They show that LSTM regression is not advantageous and that better performance can be achieved by a one-class SVM outlier detection based on statistical features. Finally, authors in (Yu *et al.*, 2022) present a data analysis system combining ARIMA and LSTM for persistent organic pollutants concentration prediction. ARIMA is used to capture linear components while LSTM is used to capture non-linear components. They use the one-class SVM method to detect the anomalies of POPs concentration values, then use the average sampling concentration in the previous month to replace this abnormal value.

LOF is another commonly applied technique to be efficient at finding anomalous data points by measuring the local deviation of a given data point with respect to its neighbours. In (Zhang *et al.*, 2022), authors apply LOF and adaptive K-means to an abnormal data recognition algorithm to implement data preprocessing and noise extraction on wind turbine data. Univariate data is then re-combined and processed into an LSTM-based stacked denoising autoencoder (LSTM-SDAE) model to obtain nonlinear temporal relationships among multivariate variables. In another approach (Zhang & Zhou, 2022), LOF is applied to enhance detection efficiency in a two-stage virtual machine abnormal behaviour-based anomaly detection mechanism. Authors demonstrate that LOF can significantly reduce computational complexity and meet the needs of real-time performance.

Lastly, a common outlier detection technique frequently found in the literature is the Mahalanobis Distance (MD). It measures the distance between a point P and a distribution D. It is a multi-dimensional generalization of measuring how many standard deviations away P is from the mean of D. In (Zhang *et al.*, 2022), MD is calculated based on reconstruction errors, and an

alarm mechanism based on the sliding window technique was set up to detect abnormalities in real time. Finally, authors in (Zhu, Wu, Wu & Liu, 2022) used MD to denoise multidimensional sensor data, flag the outliers, and enhance the robustness of the denoising process.

While these approaches prove to be extremely efficient when applied to specific problems involving univariate data, none offers a way to precisely detect outliers from multivariate data in time series. This point is particularly of high value for an approach such as A-NFVLearn, which heavily relies on interdependencies of several resource attributes, hence, of multiple interrelated variables.

1.3 State-of-the-art AI- and ML-based Univariate and Multivariate Workload Forecasting Methods

Workload forecasting methods in cloud computing is a topic that has gained renewed attention with the emergence of machine learning and deep learning techniques. Several articles raise how ML- and DL-based mechanisms can be successfully applied to several areas of cloud computing, next-generation network (NGN) and NFV infrastructures (Ray, 2021; Boutaba *et al.*, 2021). In the article by Partha Pratim Ray, the author gives a comprehensive future roadmap on the challenges, enablers, requirements and technology for 6G adoption, and dedicates a full section to Deep Learning for 6G telecom networks. Ray notes that "self-organizing architecture may be associated with the dynamic resource sharing of 6G data to improve fault, configuration, account, performance and security (FCAPS)" (Ray, 2021). Moreover, workload forecasting is often cited as a key step in order to anticipate proper virtual network embedding (VNE), VNF placement, VNF scaling, network slicing and resource allocation decisions (Boutaba *et al.*, 2021).

Undoubtedly, ML's and DL's ability to learn from deep relationships to make accurate predictions is a natural combination with complex, dynamically evolving, and large-scale infrastructures' (i.e., cloud computing and NFV) aim to self-organize resources without human intervention. Narrowing down to the specific research area of cloud computing workload forecasting, several state-of-the-art solutions were recently proposed to tackle the issues inherent to this domain. While DL techniques have gained significant popularity among practitioners lately (Lu,

Panneerselvam, Liu & Wu, 2016; Kumar, Saxena, Singh & Mohan, 2020; Kumar, Singh & Buyya, 2021a; Saxena & Singh, 2022; Zhu, Zhang, Chen & Gao, 2019; Saxena & Singh, 2021), traditional ML techniques such as auto-regression integrated moving average (ARIMA), Bayesian Ridge Regression (BRR), Support Vector Regression (SVR), Support Vector Machine (SVM), linear regression (LR) and K-Nearest Neighbors (KNN) still garner a lot of interest from the research community thanks to its algorithmic simplicity and performance (Gao, Wang & Shen, 2020; Kumar, Rao, Bulla & Venkateswarulu, 2021b).

Further, those approaches can be classified into very specific frameworks with a combination of either univariate or multivariate data (i.e., a single input feature or several input features) giving one-to-one (i.e., one input time step predicting one output time step), many-to-one (i.e., many input time steps predicting one output time step), one-to-many (i.e., one input time step predicting many output time steps) or many-to-many (i.e., many input time steps predicting many output time steps) architectures.

1.3.1 Univariate techniques for Workload Forecasting in Cloud Computing

Among the latest univariate, many-to-many ML-based approaches currently available, two stand out by their innovative way of using traditional ML techniques and combining them with novel techniques to improve their predictions (Gao *et al.*, 2020; Kumar *et al.*, 2021b).

First, in the article by Gao, Wang & Shen (Gao *et al.*, 2020), the authors compare the performance of representative state-of-the-art workload prediction methods. They also suggest a method to conduct the prediction a certain time before the predicted time point in order to allow sufficient time for task scheduling based on the predicted workload. To further improve the prediction accuracy of their method, they introduce a clustering-based workload prediction technique, which first clusters all the tasks into several categories and then trains an ML-based prediction model for each category respectively. Each prediction model is made of either a Prototype-based Clustering Method (PCM) or a Density-based Clustering Method (DCM) combined to one of three ML methods (ARIMA, BRR or LSTM).

Then, in the article by Kumar, Gangadhara, Bulla & Venkateswarulu (Kumar *et al.*, 2021b), their proposed prediction model conducts a comparative study that has been applied using ML univariate, many-to-many methods (LR, KNN, SVM, ARMA, ARIMA, and SVR) for web applications to select the suitable algorithm as per workload features.

Next, among DL-based univariate techniques, we find several approaches using a many-to-one design (Lu *et al.*, 2016; Kumar *et al.*, 2020, 2021a; Saxena & Singh, 2022).

For instance, the authors in Lu, Panneerselvam, Lui & Wu (Lu *et al.*, 2016) mention that scaling the level of active server resources in accordance with the predicted incoming workloads is one possible way of reducing the undesirable energy consumption of the active resources without affecting the performance quality. To this end, they analyze the dynamic characteristics of cloud computing workloads and define a hierarchy for the latency sensitivity levels of those Cloud workloads. The authors propose RVLBPNN, a Cloud workload forecasting model based on a back propagation neural network (BPNN). RVLBPNN takes many time steps of either CPU or memory usage from CPU, memory or latency-intensive workloads and predicts one time step of the same input feature. Their approach shows improved accuracy compared to Hidden Markov Model (HMM) and Naive Bayes Classifier. Their work was an early endeavour in DL-based resource usage prediction. As such, it lacks the more intricate features provided by LSTM, which is to leverage deep relationships between time steps of the input resource usage features. Moreover, RVLBPNN can only predict one time step from a single output feature, making it a univariate, many-to-one architecture.

The article by authors Kumar, Saxena, Singh & Mohan (Kumar *et al.*, 2020) presents a workload prediction framework based on a neural network and an adaptive evolutionary learning algorithm. Instead of using gradient descent or a backpropagation-based approach, their work introduces BiPhase, an adaptive differential evolution (BaDE) learning algorithm to improve the network learning process. BiPhase implements mapping between multiple inputs and a single output (hence, a univariate, many-to-one architecture), where the model extracts behavioural patterns of the actual workload from training input data and analyzes t previous workload values to forecast

the workload about to arrive at next $t + 1^{th}$ instance of time as output at the data center. The authors mention that it could be extended to forecast multivariate workload traces.

In the paper by authors Kumar, Singh & Buyya (Kumar *et al.*, 2021a), the authors present a self-directed workload forecasting method (SDWF) that captures the forecasting error trend by computing the deviation in recent forecasts and applies it to enhance the accuracy of further predictions. The proposed univariate, many-to-one framework is capable of learning from its past forecasts to improve future estimates.

Finally, Saxena & Singh (Saxena & Singh, 2022) present a workload prediction adaptive neural network model to forecast average workload over consecutive prediction intervals. Their univariate, many-to-one prediction model adaptively learns traces of workload for a particular prediction interval from historical data by applying a novel Auto Adaptive Differential Evo-lution (AADE) algorithm, which is enabled with three dimensions adaptation and developed and applied to train a feed-forward neural network for workload prediction.

One univariate DL-based framework proposes a many-to-many design. To improve workload prediction accuracy, authors Zhu, Zhang, Chen & Gao (Zhu *et al.*, 2019) propose an approach using LSTM encoder-decoder network with an attention mechanism for mixed workload prediction in cloud computing environments. This univariate, many-to-many approach can take a workload history of either 12, 18, 24, 36, 42 or 48 input time steps to predict a horizon fixed at 12 time steps.

1.3.2 Multivariate techniques for Workload Forecasting in Cloud Computing

The work proposed by Saxena & Singh (Saxena & Singh, 2021) is an energy-efficient resource provisioning and allocation framework aiming to meet the dynamic demands of future applications. The proposed framework built upon an Online Multi-Resource Feed-forward Neural Network (OM-FNN) addresses these challenges by matching the application's predicted CPU and memory resource requirement of tasks with the resource capacity of VMs precisely and thereby consolidating entire load on the minimum number of energy-efficient physical machines

(PMs). CPU and memory utilization of a task in a particular VM are extracted and aggregated per 5 minutes unit of time to forecast the resource usage information of a task during the next prediction interval.

1.4 Positioning of our approach

Our unique multivariate, many-to-many approach stands out from the state-of-the-art in many aspects. First, our aim with NFVLearn was to design models that could take several input features in order to forecast several other output features (i.e., resource attributes of a VNF, an SFC). This aspect was essential since we saw that we could gain precious processing time and storage on the system hosting those prediction models by proceeding this way: fewer models predicting more output features means less processing and less storage space. Hence, designing a framework with multivariate inputs and outputs was key. The next aspect was to design models that could take a resource usage history of several time steps and forecast a resource usage horizon of several time steps. The forecasting horizon was greatly important to us, since predicting several time steps in advance ensured that NFVLearn models could be called on sparse, critical occasions and only when necessary. Those models would not require to be called, for example, t consecutive times in order to forecast t time steps. Moreover, predicting several time steps in advance gives an edge to VNF scaling mechanisms, since getting a forecast resource usage pattern opens more beneficial scaling opportunities (e.g., knowing the resource usage trends well in advance opens the options to scale horizontally, scale vertically and migrate whereas not knowing the trends would compel us to use a more prudent approach and simply scale vertically).

In conclusion, none of the current state-of-the-art approaches proposes such an intricate mechanism. To the best of our knowledge, NFVLearn is the only multivariate, many-to-many approach in the cloud computing research domain. To this day, we are confident in saying that NFVLearn is several years ahead in terms of applicability, efficiency, performance and reliability compared to other workload forecasting techniques available so far.

CHAPTER 2

TAXONOMY OF DEPENDENCIES FOR RESOURCE ADAPTATION IN CLOUD ENVIRONMENTS AND EXPLORATION OF DEEP LEARNING TECHNIQUES APPLIED TO NFV INFRASTRUCTURES

This section presents a review of the literature surrounding the field of VNF dependencies to establish the perspective of our contributions. In addition, to give the reader a broader picture of the recently established paradigm shift in the design of VNFs and its impact on VNF dependencies, this chapter also covers microservices in the context of NFV, known as Virtual Network Function Components (VNFCs). Finally, we will explore recent deep learning (DL) techniques applied to NFV and cloud computing, and how our contributions, aimed at proposing mechanisms to observe, learn and predict VNF resource usage based on the interdependency between resource attributes in a multi-VNF environment, could benefit from such learning techniques.

2.1 VNF Dependencies and Resource Adaptation

For CSPs and TSPs, the main advantage of NFV is that it provides virtualized, standardized, and open interfaces. This means that NFs, their network links and management entities are now completely decoupled from the physical infrastructure that once hosted them. These VNFs can then be deployed, migrated or scaled on demand to any point on the vendor's cloud infrastructure. This provides great flexibility in allocating VNFs, allowing for more efficient fulfillment of requirements such as SLAs, SLOs, and then Quality of Service (QoS) negotiated between providers and their customers. However, respecting these policies while leaving the process of assigning VNFs and adapting physical resources in an NFVI in the hands of an automated mechanism is a challenging task. It is indeed very complex for any algorithms or mechanisms to anticipate the resource requirements of service function chains (SFCs) in dynamic environments with large workloads and workflow fluctuations. Thus, research areas such as VNF resource usage prediction, VNF resource adaptation, and physical and virtualized resource allocation address this automation challenge and have thus become hot topics in recent years. The lack

of efficient automated algorithms and mechanisms is still a major obstacle in achieving the objectives of 5G, B5G and IoT where the goal is to serve a multitude of devices running a huge amount of services and making a large number of requests, constantly entering and leaving the network, moving from one cell to another, while providing exemplary performance and quality of service with a total latency below 10 ms for latency-sensitive applications and less than 1 ms for very sensitive applications (AR/VR based applications).

To summarize, NFV resource adaptation is a set of automated mechanisms in an NFVI enabling on-demand horizontal and vertical scaling, as well as migration mechanisms, to adapt physical and virtual resources to accommodate fluctuating workloads. Vertical scaling involves scaling physical resources up or down, such as loading additional CPU cores to support increasing demand in CPU utilization, thus avoiding a VNF failure and incurring an SLA violation. Horizontal scaling involves scaling a Virtual Machine (VM) or container hosting a VNF in or out, effectively loading up or shutting down VMs to support application workload demand. Migration, on the other hand, involves taking a "snapshot" of a running VM hosting a VNF and allocating that snapshot to a different physical resource (ex: moving a VM from one host to another) to help balance resource load or reduce latency.

The benefits of NFV resource adaptation are numerous. For example, reduced power usage can be achieved by migrating workloads and powering down unused hardware (ETSI, 2013; Marotta, Zola, D'Andreagiovanni & Kassler, 2017b; Marotta, D'Andreagiovanni, Kassler & Zola, 2017a). In other cases, researchers tackle issues relating to VNF placement considering latency optimization, since virtualization may lead to abnormal latency variations and significant throughput instability irrespective of utilization (Gupta *et al.*, 2017). A common objective can be outlined throughout VNF resource adaptation cases: to balance the resource load incurred by VNFs along available physical resources, following certain policies and requirements of the service provider (e.g., to reduce energy costs, reduce resource loads such as CPU utilization, disk I/O and network traffic load, reduce latency, etc.).

For the many benefits of VNF resource adaptation, however, researchers must also tackle new challenges. Designing automated resource adaptation mechanisms for dynamic environments such as an NFVI is indeed no easy task, as you must consider a series of dependencies to make optimal scaling decisions in a matter of milliseconds. It is why, for example, considerations like "end-to-end latency and processing delays need to be continually monitored and managed whether the VNF placement is static or dynamic" (Gupta *et al.*, 2017). Fahmy & Saxena also mention some cases where VNF adaptation can be counter-productive. One of these examples is that currently, a natural solution to control latency with NFV is to instantiate service elements within an SFC onto physical machines in close proximity. They note that this ensures that congestion in other parts of the Data Center (DC), as well as latency due to inter-DC communication, can be avoided. However, such a placement policy will force cloud providers to pre-allocate VNF resources in designated sections of the DC, ultimately undermining their ability to maximize infrastructure resource utilization (Fahmy & Saxena, 2017).

To date, only a specific area of VNF dependencies is being investigated in the research community: SFC dependencies and its related field called VNF Resource Allocation (VNF-RA) (Herrera & Botero, 2016), itself referring to concepts such as VNF Forwarding Graph Embedding (VNF-FGE) (Beck & Botero, 2017; Blanco *et al.*, 2017) and SFC composition. This can be attributed to the fact that this area of research is relatively new. ETSI already foresaw that the absence of a definition of relationships between VNFs would eventually be problematic for further development of service creation in the early stages of the NFV Group Specification (GS) (ETSI, 2013).

Apart from SFC dependencies, many mentions are made in the literature about factors involved in the decision-making process to properly adapt VNFs, especially on the functional level. We thereby propose a roadmap to categorize such considerations as VNF dependencies under a common nomenclature. This proposal falls in line with ETSI's own suggestion to define and identify categories of relationships between VNFs.

To do so, we propose to narrow down VNF functional dependencies into two main classes, the "upstream dependencies" and the "downstream dependencies".

- **Upstream dependencies:** By upstream dependencies, we mean functional dependencies that, under certain management policies set in the NFVI, may "trigger", and thus precede, decision-making processes to adapt VNFs. Different dependencies, or combinations of dependencies, may enable different scaling decisions (scale up/down physical or virtual resources, scale VMs in/out, migrate VMs) depending on the context and policies set by the service provider.
  - Physical and virtual resource attributes (CPU utilization, disk I/O, network traffic, latency, throughput, etc.) to monitor in the NFVI.
  - Management policies; when a certain monitored dependency or a group of monitored dependencies reach a defined threshold, a VNF adaptation process is enabled.
- **Downstream dependencies:** Downstream dependencies include all factors impacting and following the decision-making process of VNF adaptation. These are dependencies that must imperatively be considered to ensure that VNF adaption doesn't impact the system's resiliency, performance and security, that it doesn't breach SLAs and SLOs and that it maintains QoS requirements. Following that definition, VNF placement in an SFC may be considered as a good example of a downstream dependency.
  - What physical or virtual resource, and what scaling action has to be taken by the VNF adaptation process? VNF Resource Allocation (VNF-RA) and VNF Function Graph Embedding (VNF-FGE) are good examples of this definition. This category is usually the last stage of VNF adaptation decision-making, thus includes some of the most challenging dependencies to implement and is subject to some of the most intense research activity as of this writing. These downstream dependencies help influence the scaling actions based on the context awareness of the NFVI, other VNFs in the environment, previous and ongoing actions in the system, etc. It might lead, for example, to a system to migrate a VNF to another location in the cloud infrastructure to reduce latency, instead of scaling up the bandwidth of that same VNF, which would be useless if our aim was to improve QoS for a service chain.

- Dependencies relating to VNF loading times during scaling and how long before that VNF could be available to handle requests. This is a relatively simple type of downstream dependency, but which could have a huge impact on VNF scaling actions and on the resiliency of the NFVI.
- Location; it refers to scaling actions specific to certain areas of the NFVI. Depending on the actual function of a VNF and the NFVI setup of a service provider, some VNFs may only be deployed in the core/edge/cloud/fog areas of a network. Depending on the area, physical resources, SLAs, QoS management policies, VNF design, network slicing, etc. may also change.

2.2 Microservice Dependencies and Resource Adaptation

Microservice Architecture (MSA) is a new paradigm entering the scene in Cloud infrastructures, thus NFV is also gradually adopting MSA as a means to design autonomous and lightweight VNF Components (VNFCs) to handle all or some parts of network functions. This section is included to 1) introduce the concept of MSA and 2) explain how microservice dependencies influence resource adaptation in a containerized environment. A more specific description of how VNFC (VNFs as microservices) dependencies can influence resource adaptation in an NFVI can be found in section 2.3.

MSA is an architectural style aiming at increasing a development team's capacity to build and maintain large applications in corporate environments. It facilitates an application's maintenance and troubleshooting by decoupling a monolithic application into several small, decoupled services. "Small" means that the service must have a single responsibility. "Decoupled" means that the service is an independently deployable entity (no shared state and can be updated, redeployed, or replaced). While monolithic applications use the same infrastructure for all functionalities, microservices have a complete separation of their entire technology stack (state and communication contracts, and paths), running in separate processes and/or environments so that communication via HTTP/HTTPS is the only link between the microservices (e Martins, Filho, Júnior, Giozza & da Costa, 2017).

Much like VNFs, a microservice elasticity is made by adding (scaling out) new instances of a service, whereas when there is a reduction in demand, underutilized instances are removed by scaling them in. Scaling up/down resources to support demand and migrating microservices is also possible. As such, MSA is particularly suitable for Cloud infrastructures, as it greatly benefits from the elasticity and rapid provisioning of resources (e Martins *et al.*, 2017; Francesco, Malavolta & Lago, 2017). Enthusiasm for MSA is especially notable in the fields of Internet-of-Things (IoT) and Service-Oriented Computing and Applications (SOCA) deployments, where the sensing and actuation capabilities of devices are exposed through microservices (Truong, Narendra & Lin, 2018). Then, higher-level services are composed of the basic microservices for various data processing and decision-making functions closer to the applications. This naturally fits well with the extensive use of Cloud resources as services, leading to a natural way of integrating IoT and Cloud for SOCA (Truong *et al.*, 2018).

Architecting microservices, however, is not an easy task as it requires managing a distributed architecture and its challenges (e.g., network latency and unreliability, fault tolerance, data consistency and transaction management, communication layers, load balancing) (Francesco *et al.*, 2017). Ghofrani & Lübke state in their survey that "Security, Performance, and Response Time are mentioned as very important to optimize in MSA" (Ghofrani & Lübke, 2018). Moreover, if on the one hand microservices can help in achieving a good level of flexibility (e.g., by promoting low services coupling, and higher maintainability), on the other hand adopting MSAs may bring higher complexity, mainly because they imply a high number of distributed services to operate (Francesco *et al.*, 2017).

In retrospect, we may claim that categories like upstream and downstream dependencies also apply to MSA, with the main difference being that a network function built using MSA may be composed of a higher number of network sub-functions called microservices. Hence, if not properly orchestrated, MSA may lead to a higher overhead in HTTP/HTTPS requests due to the higher granularity of a network function's composition, defeating the purpose and benefits of using an MSA. Close proximity of each microservice composing a network function must

therefore be included as a major microservice dependency, lest risking dramatically increasing latency and overhead in a Cloud environment.

2.3 VNFC Dependencies and Resource Adaptation

Mention of VNFCs has recently begun to appear in the literature (Fahmy & Saxena, 2017; Mijumbi *et al.*, 2017; Marotta *et al.*, 2017a,b). A VNFC can be seen as the definition of a microservice in NFV. Figure 2.1 shows a good example of different VNF compositions using VNFCs in an SFC. In such compositions, a VNF can be made of multiple instances of the same VNFC (VNF1, Figure 2.1), a cluster-tree of VNFCs, where some components only have one interconnection to the main component (as would be the case for a database access component) (VNF2, Figure 2.1), a VNF composed of a single VNFC (VNF3, Figure 2.1) and a cluster of VNFCs all interconnected (VNF4, Figure 2.1).

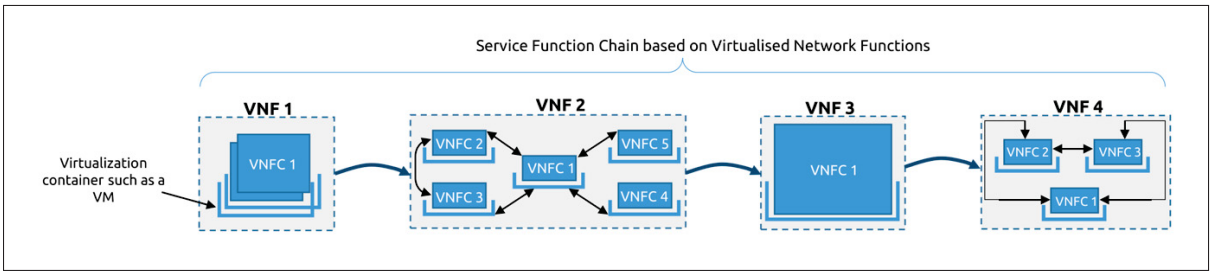


Figure 2.1 NFV service function chain with different VNF composition of VNFCs. VNFCs may be horizontally or vertically scaled
Taken from Mijumbi, Hasija et al. (2017)

Using VNFCs to compose VNFs offers a lot of advantages. For example, the ability to split VNFs into smaller VNFCs gives service providers flexibility in regard to a number of factors, like the prioritization of performance, scalability, reliability, security and other non-functional goals (Mijumbi *et al.*, 2017). Moreover, the smaller resource footprint of virtualization technologies like containers enables microservice bundles of VNFs with high affinity to be placed near each other (Fahmy & Saxena, 2017). It also allows the integration of components from other VNF providers, which could potentially free service providers even further from vendor lock-in of

VNF solutions, given VNFC providers follow similar MSA and design patterns, which is far from the case as of this writing. In fact, since VNF providers have yet to agree to definitive design rules for VNFC bundle architecture, compatibility between VNFCs composing similar VNFs from different VNF providers is non-existent. At the current state of VNFC design, different VNF providers might have a different number and topology of constituent VNFCs working together to provide the required functionality of the VNF (Mijumbi *et al.*, 2017).

Much like microservices, this issue could hurt the adoption of VNFCs, but approaches like Contain-ed (Fahmy & Saxena, 2017) tackle the lack of standards adoption by creating network microservice bundles called Affinity Aggregates (AAs). In Fahmy & Saxena's work, AAs are bundles of network services comprising VNFs that have message exchange affinity towards each other. AAs are instantiated as a single logical entity of microservices using lightweight containers, thus solving VNFC proximity dependency issues (see Section 2.2: microservice proximity). Each AA is configured to handle a predetermined transaction type and only consists of VNFs/VNFCs involved in processing this transaction type.

Aside from VNF composition, other issues pertaining to VNFCs are outlined in the literature like in Marotta, Zola et al.'s work (Marotta *et al.*, 2017b), where it is mentioned that "as VNFs are composed of a set of VNFCs that need to exchange data over the network under capacity and latency constraints, the networking also plays an important part". These factors in mind, therefore, lead us to believe that VNF composition (AA) as well as network capacity and latency constraints are key VNFC dependencies that could dramatically influence VNF adaptation decision-making.

2.4 Types of VNF Dependencies

In this section, we further narrow down upstream and downstream dependencies categories into a list of VNF dependency types. We will first discuss dependencies by resource attributes which all fall into the category of upstream dependencies. Next, we will discuss about four dependency types that fall into the category of downstream dependencies. The first one is the most common

and widely covered VNF dependency type found in the literature: dependencies by SFC. We will then discuss dependencies by Resource Allocation (VNF-RA), which is a sub-section of SFC dependencies but also a research area of its own, with different sub-categories as stated in Beck & Botero: "This challenge is commonly known as the NFV resource allocation problem, that is divided in two problem stages: 1) service chain composition and 2) service chain embedding" (Beck & Botero, 2017). Next, we will talk about dependencies by VNF localization and, finally, we will discuss other dependency types related to VNFCs.

2.4.1 Dependencies by Resource Attributes

VNF dependencies by resource attributes are dependencies based on the state of physical or virtual resource attributes (e.g., CPU, RAM and disk I/O utilization, network traffic, latency, etc.). Any dynamic resource metric monitored in the NFVI, that may enable VNF adaptation after reaching a certain threshold, fall into this type of dependency. These thresholds will often be based on configuration or management policies set by the service provider.

While resource attributes are not specifically mentioned as dependencies in the literature, many authors mention them as an important VNF adaptation factor. For example, ETSI mentions about virtual resources that "the service provider can scale the NFVI resources allocated to the VNF instance in response to increasing usage of the VNF" (ETSI, 2013). As such, most research on VNF adaptation relies on Integer Linear Programming (ILP) by using monitored resource attributes (e.g., power consumption, CPU and RAM utilization) and cost functions to help in scaling decision-making. In Gupta Samaka, et al., for example, their cost function for placing an SFC is based on the choice of Clouds, and the amount of computing, storage and networking resources consumed (Gupta *et al.*, 2017). Their approach also monitors latency between network nodes in a multi-Cloud environment. It is interesting to note that latency, being a major upstream dependency, is also dependent on a number of factors like "the state of compute, networking and storage resources, installed and used capacities of the servers and the links, traffic patterns on the link, the types of functions sharing the servers and distance between Clouds" (Gupta *et al.*, 2017). Finally, Truong, Narendra et al. propose to gather different sets

of upstream dependencies as ensembles. Ensembles consist of resources from IoT, network functions, and cloud services that establish a set of resources utilized for the application in a specific context (with time, performance, cost, etc.). Resources in an ensemble work together, clearly establishing the so-called resource slice (Truong *et al.*, 2018).

To summarize, upstream dependencies are much more complex than one can imagine. While they are based on resource attributes, which individually give a snapshot of the state of only one piece of a puzzle, VNF adaptation processes will often use a combination (sets, ensembles) of resource attributes along multiple physical and virtual resources of the NFVI to better estimate optimal scaling decisions. That is a good explanation of why VNF adaptation is an NP-Hard problem and why research teams aim to get near-optimal VNF placement solutions instead of trying to get the best solution, which is impossible considering that scaling decisions must be made within a matter of milliseconds.

2.4.2 Dependencies by SFC

Service Function Chaining (SFC), also known as Service Function Graphs (SFG) and VNF Forwarding Graph (VNF-FG), is the most commonly known VNF dependency and the most widely discussed in the literature. We refer to it as part of the downstream dependencies presented earlier in this document. It is the virtual definition of a classic Network Function (NF) Forwarding Graph, which defines the sequence of NFs that packets traverse in a network (ETSI, 2013). In other words, "a VNF Forwarding Graph provides the logical connectivity between virtual appliances (i.e. VNFs)" (ETSI, 2013) and is thus a set of VNFs or services with a well-defined sequence for the packets to travel (Gupta *et al.*, 2017). Usually, one SFC consists of a series of VNFs, with each VNF representing a specific functionality. One SFC or network service then contains one source and one destination node to communicate with other SFCs (Sun, Li, Li, Liao & Chang, 2018). For the reasons cited above, an abstract Network Service based on VNFs should include identifying the types of VNFs involved and the interconnection (forwarding) topology along with related management and dependency relationships (ETSI, 2013). All these identifications will play a significant role in the VNF scaling decision process,

as scaling one VNF along an SFC may have, depending on its function, a major influence on the resiliency and performance of the entirety or some parts of the service chain.

An approach rising in popularity to tackle the issues of VNF adaptation in the context of SFC dependencies is to place service chains as a unit rather than individual functions separately. The idea behind this approach is that each VNF composing an SFC has a set of traffic demands and maximum tolerable latency. In particular, the traffic demands specify how much traffic the first component in a service chain sends to the second one, which forwards it after some processing to the third one and so on. The latency of an SFC is therefore the sum of the experienced delays on the used paths, on which all the demands of the service chain are forwarded. It includes the propagation latency and the latency due to queuing (Marotta *et al.*, 2017b). Work proposed by Gupta, Samaka et al. also shows that placing a full chain yields better results to achieve global minima for the parameter being optimized than placing single VNFs in the SFC. The outcome is that at the infrastructure level, inter-VNF communication overheads can be reduced (Gupta *et al.*, 2017). Another work presented by Gupta, Farhan et al. also takes this approach to help network operators satisfy the service requirements for all the traffic flows in their network using minimal network resources. They model the problem as an optimization problem where the objective is to minimize the network-resource consumption by optimally placing VNF service chains across physical resources (Gupta *et al.*, 2018).

A good way to tackle the SFC dependencies problem for VNF adaptation is therefore to approach the scaling decisions based on SFCs as units, also known as a VNF Chain Composition (VNF-CC) and analyze the functional dependencies among VNFs composing an SFC, instead of basing VNF adaptation solely on single VNFs composing an SFC. On one hand, this may lead to better – and safer - scaling decisions in order to maintain the expected performance and reliability of the whole service chain. On the other hand, this may lead to a more complex evaluation of the different upstream resource attribute dependencies (e.g., the state of each VNF in the SFC, the state of physical and virtual resources), thus more computing time, to make decisions impacting a service chain. This will also have an impact on other downstream dependencies

such as resource allocation, since the optimal scaling decision proposed in some cases, may not be to scale a single VNF scarce on resources, but to scale some parts or even a whole SFC.

2.4.3 Dependencies by Resource Allocation

VNF Resource Allocation (VNF-RA) (a downstream dependency) is currently one of the hottest topics in the field of VNF adaptation. It is an offspring of the SFC research area which has grown to become a whole research area of its own. A comprehensive survey was made about VNF-RA in 2016 which encompasses new concepts tightly tied to SFC like VNF Forwarding Graph Embedding (VNF-FGE), VNF Chain Composition (VNF-CC) and VNF Scheduling (VNF-SCH) (Herrera & Botero, 2016). This type of downstream dependency, however, contrasts from SCF dependencies by the following: NFV-RA's input is a network service request composed of a set of VNFs with precedence constraints and resource demands that can be denoted by several VNF-FGs; the task of the first stage of the problem (VNFs-CC) is to efficiently build a suitable VNF-FG with regard to the operator's goals (Herrera & Botero, 2016). That means that a VNF-RA dependency is in fact a combination of several SFC dependencies (SFC dependencies also referring to SFG and VNF-FG dependencies) along with the resource demand (e.g., number of CPUs, RAM quantity, disk size, VM loading time, etc.) for each SFC. Other definitions of VNF-RA also appear in the literature, like this one in Beck & Botero's work: "The objective of an NFV-RA algorithm is to embed a set of VNF embedding requests (VNFRs) on top of a shared substrate network infrastructure in an efficient way. The algorithm has to consider placement constraints and dependencies between VNFs" (Beck & Botero, 2017). This approach to VNF-RA is similar to the previous one by Gil Herrera & Botero, with the exception that they don't clearly state what placement constraints and VNF dependencies to consider in a VNF-RA algorithm. In this context, we find that resource demand and SFC dependencies by VNF-CC make a suitable definition of VNF-RA dependencies.

The VNF-RA and adaptation problem relies on different configuration and management policies that need to be carefully weighted and balanced to get optimal results. For instance, from the tenant's point of view, the placement problem may boil down to placing network functions to

meet criteria like cost, jitter, packet loss, latency, throughput in services like voice and video calls, content delivery, broadband services, carrier backbone or a combination of these. The cloud service provider, on the other hand, would like to minimize the use of resources while meeting the tenant's requirements (Gupta *et al.*, 2017). In this context, a natural solution to control latency in an NFVI would be to instantiate service elements within an SFC onto physical machines in close proximity. This may ensure that congestion in other parts of the DC, as well as latency due to inter-DC communication, can be avoided. However, such a placement policy would force CSPs to pre-allocate VNF resources in designated sections of the DC, ultimately undermining their ability to maximize infrastructure resource utilization (Fahmy & Saxena, 2017).

Such dilemmas have pushed several research teams to find an optimal balance between a tenant's and a service provider's goals. In their work, Gupta, Samaka et al. discuss a strategy that would primarily optimize cost and at the same time keep end-to-end latency below the threshold prescribed by the carrier (Gupta *et al.*, 2017). Others like Gupta, Farhan Habib et al. determine the routing of traffic flows and placement of VNFs that minimizes network-resource (bandwidth) consumption given a network topology, link capacity, a set of DC locations, a set of network nodes with virtualization support (NFV-capable nodes), traffic flows between source-destination pairs, the bandwidth requirement of the traffic flows between the source-destination pairs, K shortest paths between source-destination pairs, and set of network functions required and a service chain (Gupta *et al.*, 2018). Other approaches take a different view on VNF-RA, as they aim to optimize an NFVI power consumption instead of optimizing network resources. Marotta, Zola et al.'s works, for example, propose a fast three-phase heuristic to tackle the problem of designing a power-efficient Virtual Network Infrastructure under uncertainty of resource demands (Marotta *et al.*, 2017a,b). Their approaches aim to help TSPs in the planning decision-making by finding a balance between protection from demand uncertainty of the VNFs and a higher cost in terms of additional energy consumption required due to more servers and network elements needed to protect from uncertainty. Their latest work (Marotta *et al.*, 2017a) is an improvement of their previous approach, where interestingly, input to the problem is not

known precisely, but rather resource demands are allowed to deviate within bounds. This allows their model and heuristic to make tradeoffs between energy efficiency and robustness under uncertainty constraints.

As previously mentioned, VNF-RA's inputs are sets of SFCs and resource demands. That means that VNF-RA optimization algorithms need to scale according to the problem size, which up to recently, was not the case (Beck & Botero, 2017). Among the main factors explaining this shortcoming is that algorithms scaling to the problem size tend to have longer runtimes without significant improvements in acceptance ratios. This explains why researchers are increasingly relying on machine learning to propose optimal solutions based on the machine's previous experience.

The use of machine learning, and AI, at the step of VNF-RA is a proper consideration for anyone willing to make an efficient end-to-end VNF adaptation solution. VNF-RA is prone to dynamic changes in policies, VNF resource requirements and traffic. It is also the last step in the chain of downstream dependencies, considering all the shifting layers of dependencies from SFCs, VNF localization and VNFCs. This makes it nearly impossible to propose optimal solutions in this context, given all parameters at stake. It is therefore crucial to consider building models benefiting from past experience of the system to help in the decision-making while keeping its runtime as low as possible.

2.4.4 Dependencies by VNF Localization

Dependencies by VNF localization are a special case of downstream dependencies. Because although the physical location of the infrastructure is largely irrelevant for cloud computing services, many network services have some degree of location dependency (ETSI, 2013). This is especially the case for TSPs with 5G and IoT deployments, as network functions in an SFC may require specific locations in the network (core, edge, CPE). The idea of a computing platform located at the mobile network edge is not new and is carried out inside ETSI by the Mobile-Edge Computing (MEC) ISG, whose activity started in December 2014 (ETSI, 2016). It follows

the current trends towards Cloud-based architectures operating in an IT environment but with the peculiarity of being located at the edge of the mobile network, within the RAN and in close proximity to the mobile subscribers. Distinctive features of the MEC architecture are low latency, proximity, location awareness, high bandwidth, and real-time insight into radio network information. This facilitates accelerated content delivery services and applications at the edge of the mobile network, closer to the end-users. The mobile subscriber's experience can be significantly improved through more efficient network and service operations, enhanced service quality, minimized data transit costs and reduced network congestion (Blanco *et al.*, 2017). Affinity and anti-affinity constraint enforcement between VNFs (Jacobs *et al.*, 2018) can also be considered a localization dependency.

Another case of VNF dependencies by localization could be where a certain service provider may be interested in running a VNF instance on the NFVI of another service provider to comply with regulatory requirements since some regulatory authorities place geographic restrictions on the location of storage and processing of certain kinds of consumer information (ETSI, 2013). Finally, a service provider may also be interested in running a VNF instance on the NFVI of another service provider to improve customer experience by reducing latency, since latency can be reduced by placing selected network functions close to the consumer of that network service. For example, a CDN service can reduce latency (and reduce cost) for content consumers by caching that content closer to the customer. In another example, certain Evolved Packet Core (EPC) functions may reduce latency and improve throughput for mobile consumers if they can be located closer to the Radio Access Network (RAN) (ETSI, 2013).

VNF adaptation by localization dependencies is currently the subject of active research, mainly in the areas of 5G and IoT, since these technologies are in full-scale deployment. For example, teams like Sun, Li et al. are dedicated to finding a network service placement scheme with minimal latency in the network with edge computing capabilities by studying the application of workflow-like requests in network environments such as wireless virtual networks or NFV and studying the application of time management and application in the process of big data processing (Sun *et al.*, 2018). In Truong, Narendra et al., the authors propose the notion of an autonomic

middleware for IoT-based systems (Truong *et al.*, 2018). The key aspect of this middleware is the use of MAPE-K concepts to facilitate context-aware adaptation of IoT service compositions. This also requires the automatic generation of these compositions, especially in two cases: (1) the possible large latency of computation required to generate the compositions, and (2) the composition generation depends upon specific formalisms to specify preconditions/effects and the planning algorithms. They also present the concept of a multi-layered context model —application layer, environment layer, device layer— so as to facilitate contextual adaptation. Truong, Narendra et al. also raise a tricky problem about localization dependencies. Generally, execution policy enforcement should be carried out across IoT, network functions and clouds. The execution of the policy on the IoT side is different from that for Clouds and networks, such as ones based on software-defined machines profiles and policies and blockchain-based permission control. They also point out that if a solution provider develops an ensemble and deploys the ensemble as a service for its customers, how does this solution provider work with underlying IoT, network functions and cloud providers with regard to execution policy? This problem still remains unclear (Truong *et al.*, 2018).

That last statement summarizes well the challenges involving localization dependencies in VNF adaptation. As localization dependencies may not be provided by VNF solution providers, it is currently left up to the researchers' perception and inner knowledge of VNF deployment to supervise these specifics in their approaches. Incidentally, that may make the task much more difficult to propose generic approaches, fitting any context of VNF adaptation. Someone willing to propose solutions for localization-specific VNF adaptation may have to focus more deeply on localization dependencies for each network function in an SFC, asking for complete localization specifications to the solutions providers.

2.4.5 Additional Dependencies Related to VNFCs

The last type of identified downstream dependencies is additional dependencies related to the architecture of VNFs through VNFCs. This is a new concept that is already being studied by

some research teams trying to mitigate, notably, latency and performance issues that could appear if VNFCs shaping a VNF are not properly instantiated.

For instance, Marotta, D'Andreagiovanni et al. have found that although deploying each VNFC on a different server may result in lower SLA violation due to CPU contention, it will also increase the energy and resource costs due to more active resources and additional traffic exchanged, leading to a higher router and link utilization, network contention and increased energy cost for the network (Marotta *et al.*, 2017a). Others, like Mijumbi, Hasija et al. discovered that the resource requirements of each VNF change over time with changes in traffic, which calls for ways of increasing and reducing resources allocated to the VNFCs as needed (Mijumbi *et al.*, 2017). This has led research teams to mitigate those issues by, for example, dynamically creating and managing affinity aggregates (AAs), thus allowing VNFCs in an AA to be placed in close proximity, hence bounding end-to-end latency (Fahmy & Saxena, 2017). Through their research, they have found that AAs can effectively bind SFC delays and significantly reduce protocol errors and service disruptions. Others, like Martins, Filho et al., use the API Gateway design pattern as a means for consumers to communicate to a single point of contact to middleware microservices (e Martins *et al.*, 2017), which turns out to become increasingly popular in MSA and Edge network IoT application design (Di Francesco, Malavolta et al., 2017). The main challenge of this implementation ecosystem is that several point-of-contact instances must be made available and unavailable simultaneously so that it becomes impracticable for the client to directly access any of them (e Martins *et al.*, 2017).

Although the concept of VNF design through VNFCs and microservices is relatively new, one must not disregard the critical aspects of their dependencies on the scale of a whole NFVI. This not only impacts expected VNF resource demand, which could greatly fluctuate depending on its VNFC design and the scaling capabilities of each component, but also whole SFCs. This is a major point to take into consideration when thinking about VNF-RA since the level of granularity that VNFCs induce in the process of estimating resource demand for different sets of SFCs could explode, or at the very least increase the margins of expected resource demand for each. It is interesting to note, however, that design patterns such as API Gateway mitigate the

expansion of point-to-point communication between consumers and Edge network entry points, thus helping reduce the number of dependencies to a bare minimum on that side of the network, by giving the consumers access to the Edge network resources only through a single point of contact (SPOC).

2.5 Deep Learning and NFV

Deep learning is becoming a hot topic in the area of system automation and optimization, and NFV isn't being let down, as both the European Telecommunication Standards Institute (ETSI) and the Internet Engineering Task Force (IETF) have recently dedicated an Industry Specification Group (ISG) and a Working Group (WG) on the matter. ETSI's Experiential Networked Intelligence (ENI) ISG suggests that by leveraging Artificial Intelligence (AI) and appropriate policies, network operators will be able to predict potentially hazardous situations where two or more network slices are competing for the same resources and employ preventive measures, e.g. by using resource reservation. In other cases, more specifically even when it is not possible to predict a certain scenario in advance, actions still need to be made at runtime autonomously, e.g. by increasing the priority of a given network slice over neighbouring slices (ETSI, 2019). Therefore, the aim of ETSI's ENI system is to automatically collect network status and associated metrics, faults, and errors, and then use AI to ensure network performance and quality of service are met at the highest possible efficiency (e.g. with the minimum required resources). An ENI system can also be used to find bottlenecks of service and/or failure of the network. Both of these benefits are done on-demand, in response to changing contextual information (ETSI, 2013). The initial scope of IETF's ANIMA working group's effort is much more modest, in comparison. ANIMA's aim is currently to make the specifications of a minimum set of specific reusable infrastructure components to support autonomic interactions between devices and to specify the application of these components to one or two elementary use cases of general value (IETF, 2020). Practically, these components should be capable of providing the following services to those distributed functions: a common way to identify nodes, a common security model, a discovery mechanism, a negotiation mechanism to enable closed-loop

interaction, a secure and logically separated communications channel and a consistent autonomic management model. We should note that any researcher willing to propose AI models and approaches for NFV applicable to the industry should closely follow both groups' efforts and stick to any future specifications that may come out in the future.

Although this research area is very recent, there are already several research teams focusing on the topic of Deep Learning in NFV and closely-tied fields such as 5G/B5G and IoT.

In Truong, Narendra et al., for example, the authors mention that adaptation needs to be context-aware and driven by stored knowledge about the system and earlier adaptation implementations. However, current adaptation techniques and implementations are only for specific systems. They thus recommend exploiting end-to-end context-aware adaptation based on issues triggered from the IoT side with also resources in the Edge and Cloud (Truong *et al.*, 2018). Mijumbi, Hasija et al. also propose an interesting approach based on Graph Neural Networks (GNN) which is comprised of four main components: (1) SFC features, (2) VNFC states, (3) state computation, and (4) output computation (Mijumbi *et al.*, 2017). The authors note that since neighbouring VNFCs will usually be part of the same SFC, resource fluctuations at one VNFC are expected to influence resource requirements at its neighbours as traffic flows from one VNFC to the other. This dependency of VNFCs on their neighbourhood makes the connectionist approach derived from the GNN model an interesting fit for managing resources in NFV. They also add that their input models have shortcomings due to the encoding networks' complexity since the backpropagation through time algorithm used for training the SFC encoding network requires storing the states of each parametric function. Therefore, if the SFC is large, this might require a considerable amount of memory.

Another work by Mijumbi, Gorricho et al. proposes an approach based on Reinforcement Learning (RL), a technique from AI in which an agent placed in an environment performs actions from which it gets numerical rewards (Mijumbi *et al.*, 2014). For each learning episode, the agent perceives the current state of the environment and takes action. The action leads to a change in the state of the environment, and the desirability of this change is communicated to

the agent through a scalar reward. The agent's task is to maximize the overall reward it achieves throughout the learning period. It can then learn to do this over time by systematic trial and error, guided by a wide variety of learning algorithms. The contribution of their work is two-fold: 1) a distributed learning algorithm that allocates resources to virtual nodes and links dynamically and 2) an initialization scheme that biases the learning policy to improve the rate of convergence. In their work, they also show through simulation that their proposal improves the acceptance ratio of virtual networks, which would directly translate into revenue for the substrate network providers, while ensuring that, after the agents have learnt an allocation policy, the quality of service to the virtual networks is not negatively affected. However, they note that implementing their proposed algorithm in real networks could pose more questions, e.g., the ease of having distributed network loading information, whether a dedicated network would be needed for communication between the agents, etc.

In Cao, Fahmy et al., the authors introduce the concept of resource flexing for NFV, which is the notion of allocating resources on-demand as the workload changes (Cao, Fahmy, Sharma & Zhe, 2018). To tackle this issue, they present an "Elastic resource flexing system for Network function VIrtualization" (ENVI) that leverages a combination of VNF-level features and infrastructure-level features to construct a neural network-based scaling decision engine for generating timely scaling decisions. ENVI's design is based on a fully-connected neural network with a classification output. The model thus takes a fixed size of VNF features as an input, then evaluates the proper scaling decision to undertake as an output (don't scale, scale up or scale out). This paper offers excellent insight into high-level network attributes and their direct influence on VNF virtual resource attribute dependencies (e.g., HTTP workload with a 30KB response size has a larger CPU impact on a VNF than those with 100KB). Their approach is based on an FC network instead of RNN, however, lacks essential mechanisms such as LSTM and an attention model, which could learn complex dependencies between different resource attributes in a dynamic environment. Moreover, since ENVI's models are trained with supervised scaling output labels, that approach could not be considered flexible. A change in system policies (i.e.,

scale a VNF when it reaches 65% CPU load instead of 60%) means we would need to re-train a new model from scratch and adapt the output labels accordingly.

In another work, Blenk, Kalmbach et al propose an admission control based on an RNN to improve a system's overall performance for the online virtual network embedding (VNE) problem (Blenk, Kalmbach, Smagt & Kellerer, 2017). Before running a VNE algorithm to embed a virtual network request (VNR), the RNN predicts whether the request will be accepted by the VNE algorithm based on the current state of the substrate and the VNR. If a solution exists for the VNR, that request is then sent to the VNE algorithm for further processing. If not, the VNR is simply rejected. More specifically, the VNRs and substrate networks are represented as graphs of network features, similar to the approach proposed in (Mijumbi *et al.*, 2017). The goal of the RNN, then, is to classify the probability that a VNR graph can be mapped to the underlying substrate network graph. A graph-based approach for solving Cloud and NFV problems, like the ones proposed in those papers, is recurrent in the literature and was applied in our own approach, as it enables the detection of complex relationships between different sequential features. The approach proposed in Blenk, Kalmbach et al., however, is solely aimed at easing the processing required for a VNE algorithm to find a near-optimal solution and, therefore, doesn't fully tap on the power of an RNN's ability to find complex solutions for complex problems. It would be interesting to work further on this approach to embed a VNE algorithm in its design (Blenk *et al.*, 2017).

Another interesting approach, while not directly related to NFV, could be applied to upstream dependencies such as resource attributes. In Prats, Berral et al., the team uses conditional restricted Boltzmann machines (CRBMs) to model time series containing resource traces measurements like CPU, memory, and IO (Prats, Berral & Carrera, 2018). In the principal case for their workload traces, they are able to verify that the proposed phases capture different properties from the workloads, consistently characterizing executions by resource consumption for different kinds of applications. In our view, this could be an interesting approach to identifying properties of VNF adaptation by characterizing scaling actions by resource consumption.

In Shah, Yuan et al. (Shah *et al.*, 2017), an approach based on a Recurrent Neural Network (RNN) using a Long Sort-Term Memory mechanism is used to analyze dependencies of Cloud applications for performance monitoring. This paper proposes an extension to the LSTM model with the introduction of a "probe matrix" to the update gate, which proves to be very useful in selecting scanning areas in time series to let the NN more easily decipher relationships between data segments.

In Jacobs, Pfitscher et al. (Jacobs *et al.*, 2018), the authors use a GNN with a single hidden layer to predict affinity or anti-affinity between VNF pairs. This mechanism, while not a typical neural network (NN) in the sense that it has only one hidden layer, provides good insight into how to organize input data in order to feed a NN.

Lastly, a promising encoder-decoder architecture called Graph2Seq is proposed in Xu, Wu et al. (Xu *et al.*, 2018). Their approach proposes a general end-to-end graph-to-sequence neural encoder-decoder architecture that maps an input graph to a sequence of vectors and uses an attention-based LSTM method to decode the target sequence from these vectors. Results obtained in shortest path task scenarios are impressive and show that, if properly tuned for our own objectives, this approach could very well fit for tasks such as proposing scaling decisions for VNF nodes in a VNF-FG, for instance.

To summarize, different families of Deep Learning algorithms could prove to be useful to help build adaptable models based on tracelogs of live NFVIs. While these algorithms require large amounts of data to make efficient models, such models, when correctly tuned, will lead to accurate VNF adaptation decision-making in dynamic environments. The main challenges for researchers at this stage are to find such large amounts of tracelogs and then to find methods to properly extract, transform and load that data for useful information. Neural Networks (NNs) such as RNN and GNN should be further investigated, as they seem to be the most suitable approaches to decipher useful information from time series-based tracelogs like those provided in NFVI.

CHAPTER 3

NFVLEARN: A MULTI-RESOURCE, LSTM-BASED VNF RESOURCE USAGE PREDICTION ARCHITECTURE

The material presented in this chapter is part of the published journal article cited below:

Cédric St-Onge[1] , Nadjia Kara[1] , Claes Edstrom[2]

[1] Department of Software and IT Engineering, École de Technologie Supérieure,
1100 Notre-Dame Ouest, Montréal, Québec, Canada H3C 1K3
[2] Ericsson Canada,
8275 Rte Transcanadienne, Saint-Laurent, QC H4S 0B6

Published in "Journal of Software: Practice and Experience, Wiley", in October 2022.

3.1 Introduction

NFV technology is a keystone in deploying next-generation telecommunication networks and cloud infrastructures. It is a concept that aims at dematerializing legacy, immutable network functions (NF) originally embedded on physical hardware and converting them into lightweight and scalable VNFs deployable on-demand anywhere in a cloud infrastructure or network function virtualization infrastructure (NFVI). VNF resource scaling allows automated deployment of NFs initially anchored in a physical SFC at various points of a telecom service provider (TSP) or a cloud service provider (CSP) infrastructure such as the core, cloud or edge network. Among its many benefits, it ensures that TSPs and CSPs efficiently meet their customers' fluctuating resource demands and Service Level Objectives (SLO) (e.g., delay and availability of highly sensitive applications), surpass service level agreement (SLA), CAPEX, OPEX, and quality of service (QoS) objectives, and significantly reduce latency in an SFC (Boutaba *et al.*, 2021).

However, to achieve these goals and bring NFV technology to maturity, new mechanisms for automated VNF deployment must be designed. To this end, many research teams have been looking at ML techniques to predict future SFC resource usage efficiently. In turn, these mechanisms will facilitate decision-making to automatically adapt available physical resources

to VNF instances by triggering VNF horizontal scaling, vertical scaling or migration requests by the NFV manager.

Nevertheless, strengthening ML-based NFV resource usage prediction mechanisms is no small task. Not only does the research community need to have access to massive amounts of data from operating NFVIs, but they also need to define techniques for mining and filtering this data in conjunction with the know-how of seasoned NFV experts as well as skilled ML and deep learning (DL) practitioners with a vast experience in ML/DL algorithms development such as time series prediction and recurrent neural networks (RNNs).

One of these challenges we aim to solve in this chapter lies in analyzing and pruning less correlated features from these datasets. This pruning allows LSTM model performance optimization by decreasing the number of input features required without reducing prediction accuracy. In the context of VNF resource usage prediction, it summarizes by taking advantage of highly correlated resource load interdependencies between the resource attributes aimed at prediction and other VNFs resource attributes within an SFC. In other words, the challenge lies in highlighting the inter-dependencies between certain resource loads (RAM, CPU, network traffic) of a VNF we aim to predict on other VNF resources of an SFC in an attempt to obtain the most accurate predictions with fewer input features.

This chapter is organized as follows. We describe the problem description and motivation in Section 3.2 and present the related work in Section 1.1. A description of correlation coefficients used in this chapter and of the NFVLearn design are presented in Sections 3.3 and 3.4, respectively. Experiments and evaluation of our proposed techniques are described in Section 3.5. Finally, a discussion and a conclusion are presented in Sections 3.6 and 3.7 respectively.

3.2 Problem Description and Motivation

Multivariate, many-to-many VNF resource usage prediction mechanisms require a resource load history from multiple sources in order to truly benefit from resource attribute interdependencies

of an SFC, thus providing the most accurate predictions. Hence, it should not be neglected that a considerable amount of historical resource usage data from VNFs may travel at any time throughout the control plane of an NFVI, therefore increasing the traffic load overhead of the infrastructure. Although this is an unavoidable limitation of data-hungry DL implementations, it can, fortunately, be mitigated when thought out at an early stage of the design. So far, two different implementation types appear in the literature to mitigate this limitation. One is to run the VNF resource usage prediction model on the NFV management and orchestration (MANO). It may be a good option if we want to offload resource usage prediction tasks from the VNFs (e.g., when there are VNF processing, memory, power and disk size constraints, a low number of SFCs to manage in the NFVI, low network latency, large network bandwidth, a small number of network hops from the VNFs to the MANO, etc.) (Mijumbi *et al.*, 2017). Another implementation type is to run the VNF prediction models locally on a VNF, which is a better option in large-scale NFVIs with a large number of managed SFCs and VNFs in remote data centers where outgoing resource usage data in the control plane could eventually cause network delays and bottlenecks (Cho *et al.*, 2020; Duggan, Shaw, Duggan, Howley & Barrett, 2019).

In its current state, NFVLearn's flexibility (i.e., selection of the number of input and output time steps and their granularity, selection of input and output features) makes it design agnostic, making it fit for different types of implementation designs. However, a gap remains in the state-of-the-art. The literature lacks an autonomous framework that could allow the selection and evaluation of different sets of highly correlated input features from many available VNF resource attribute data to build our prediction models. For instance, each set could be generated from different input feature selection setups. In turn, those sets could then be evaluated with select metrics to demonstrate that a specific model using a specific setup meets certain user-defined conditions such as prediction accuracy, prediction fidelity, the number of input features, and the number of hops required for the resource metrics to reach the prediction model.

In order to reach this goal, the main contributions of this chapter are the following:

- Proposition of a novel multivariate, many-to-many LSTM architecture tailored specifically for multi-scale resource usage prediction of multiple, different resource attributes of a VNF,

enhanced by inter-dependencies in resource usage history of multi-scale resource attributes of an SFC.

- The study and investigation of four novel input feature selection setups for a multivariate, many-to-many LSTM resource usage prediction model named NFVLearn. Those solutions leverage Graph Neural Networks (GNNs), Pearson correlation coefficients, Spearman rank correlation coefficients and Kendall rank correlation coefficients theories, respectively.
- The study and investigation of a combination of two metrics to better infer the validity of NFVLearn's output predictions: one to evaluate the prediction accuracy (RMSE) and one to evaluate the prediction fidelity (coefficient of determination).
  - RMSE is a measure of accuracy that compares individual resource load forecasting errors of the different setups for a particular dataset.
  - Coefficient of determination (R^2 score) is a measure of how well predictions are replicated by the model, based on the proportion of total variation of outcomes explained by the model.

3.3 Correlation Coefficients

This section discusses the main correlation coefficients used to analyze VNF resource attribute interdependencies.

3.3.1 Pearson Correlation Coefficient

Pearson's correlation coefficient (eq. 3.1) is the covariance of two variables divided by the product of their standard deviations. It is a measure of the linear correlation between two variables X and Y. We used that approach to identify highly correlated resource attributes from an SFC. Next, we proposed a new training setup using pruned input features from these resource attributes.

$$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \cdot \sigma_Y} \tag{3.1}$$

Where,

$cov(X, Y)$	Covariance
σ_X	Standard deviation of X
σ_Y	Standard deviation of Y

3.3.2 Spearman's Rank Correlation Coefficient

Spearman's Rank Correlation Coefficient (eq. 3.2) is a nonparametric measure of rank correlation. It assesses how well the relationship between two variables can be described using a monotonic function. The Spearman correlation between two variables equals the Pearson correlation between the rank values of those two variables.

$$r_s = \rho_{rg_X, rg_Y} = \frac{cov(rg_X, rg_Y)}{\sigma_{rg_X} \cdot \sigma_{rg_Y}} \tag{3.2}$$

Where,

ρ	Denotes the usual Pearson correlation coefficient, but applied to the rank variables
$cov(rg_X, rg_Y)$	Covariance of the rank variables
σ_{rg_X} and σ_{rg_Y}	Standard deviations of the rank variables

3.3.3 Kendall's Rank Correlation Coefficient

Kendall Rank Correlation Coefficient (eq. 3.3), or Kendall's τ coefficient, is a statistic used to measure the ordinal association between two measured quantities. It is a measure of rank correlation: the similarity of the orderings of the data when ranked by each of the quantities.

$$\tau = \frac{2}{n \cdot (n-1)} \sum_{i<j} sgn(x_i - x_j) \cdot sgn(y_i - y_j) \tag{3.3}$$

Where,

$(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$	A set of observations

3.4 NFVLearn Design

This section describes NFVLearn's design and covers the novel interdependency-based mechanisms enhancing Virtual Network Function (*VNF*) resource usage prediction implemented within. First, we present the data structure of the inputs and outputs of our models. Next, we show two different frameworks, or setups, automating the process of selecting optimal input features from a data set which helps improve the resource usage prediction accuracy of selected output resource attributes. The first setup is based on a GNN while the second setup is based on a pruning process following one of three correlation coefficient functions described in section 3.3: *Pearson*, *Spearman's rank* and *Kendall rank* correlation coefficients. Finally, we discuss NFVLearn's proposed LSTM-based architecture leveraging interdependencies in the resource usage history of multiple VNFs and different resource attributes.

3.4.1 Data Structure

NFVLearn's models are built using supervised learning. It first requires gathering timestamped resource usage historical data collected from simulations at regular intervals from Virtual Network Functions (*VNFs*) of an SFC. Resource usage can be from any resource attribute of a VNF: CPU usage, memory usage, input bandwidth, output bandwidth, etc. This data is then filtered and organized such that each data point is represented as V_θ^ρ, namely, the resource usage of resource attribute ρ from VNF V at time step θ in a matrix of dimension $[\theta, \rho]$. Data reorganized in this manner will later simplify data pre-processing as required by our LSTM architecture; for instance, it allows us to:

1. select a given number of input features F_x and output features F_y from any of the resource attribute columns ρ of the dataset and,
2. split the total number of time step rows θ of the dataset for selected input features F_x and output features F_y into training examples of T_x (the resource usage history) input time steps and T_y output time steps (the resource usage horizon).

This process results in the generation of a total of $\theta - (T_x + T_y)$ training examples which can further be split into training and validation sets, with each training example being a set of two matrices: the input matrix and the output labels matrix.
The input matrix of a training example (eq. 3.4), of dimension $[T_x, F_x]$, is denoted as follows:

$$x_{[F_x]}^{\langle T_x \rangle} \tag{3.4}$$

Where,

T_x Number of input time steps
F_x Number of input features

The output label matrix of a training example (eq. 3.5), of dimension $[T_y, F_y]$, is denoted as follows:

$$y_{[F_y]}^{\langle T_y \rangle} \tag{3.5}$$

Where,

T_y Number of output time steps
F_y Number of output features

The selection of T_x and T_y is at the user's will, based on factors such as prediction accuracy, the length between input/output time steps, interdependency reinforcement, etc. However, the selection of F_x and F_y is entirely automated and depends on the choice of one of the proposed data structure setups described in subsections 3.4.3 and 3.4.4 below. A data frame $[X, Y]$ of $\theta - (T_x + T_y)$ training and validation examples is therefore denoted as a set of input samples (eq. 3.6) and output labels (eq. 3.7):

$$x(i)_{[F_x]}^{\langle i, i+T_x \rangle} \forall i \in \left\{1, 2, ..., \theta - (T_x + T_y)\right\} \tag{3.6}$$

$$y(i)_{[F_y]}^{\langle i+T_x+1, i+T_x+T_y\rangle} \forall i \in \{T_x+1, T_x+2, ..., \theta - T_y\} \tag{3.7}$$

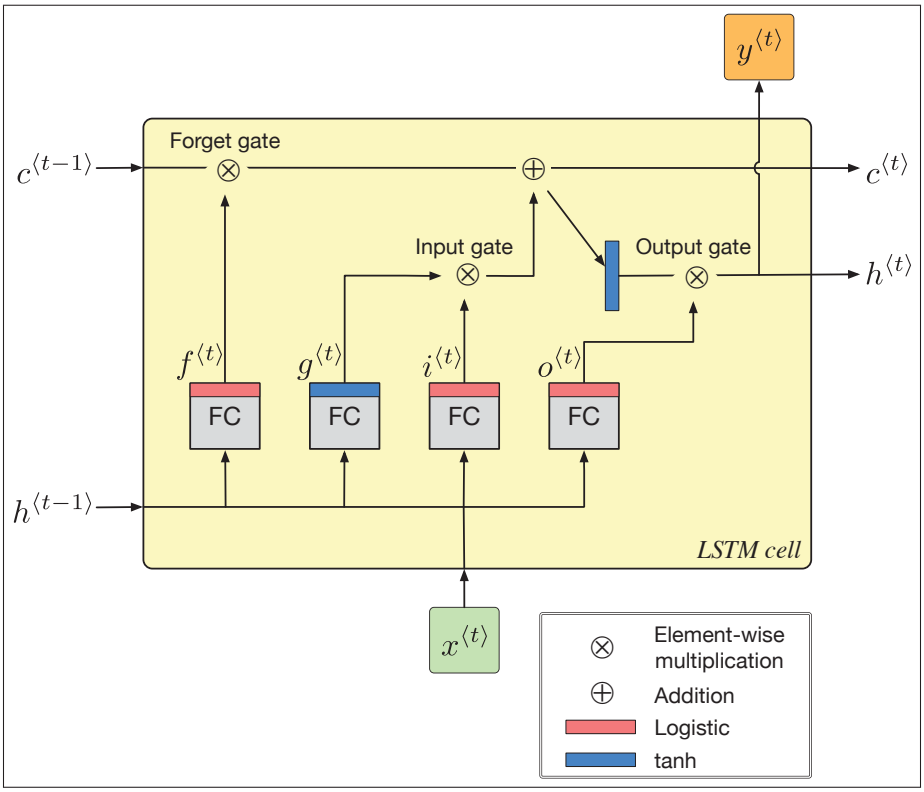


Figure 3.1 LSTM cell with fully connected (FC) neural network cells

3.4.2 NFVLearn Architecture

Long Short-Term Memory (*LSTM*) (Hochreiter & Schmidhuber, 1997) is widely used in the field of time series prediction due to its inherent capability of learning long-term dependencies in historical data. As depicted in Fig. 3.1, the key idea is that the network can learn what to store in the long-term state, what to throw away, and what to read from it. As the long-term state c at time step $t-1$ ($c^{\langle t-1\rangle}$) traverses the network, it first goes through a *forget gate*, dropping some memories, and then it adds some new memories via the *addition* operation (which adds the memories that were selected by an *input gate*). The result $c^{\langle t\rangle}$ is then sent out of the cell without any further transformation. So, at each time step, some memories are dropped, and

some memories are added. Moreover, after the *addition* operation, the long-term state is copied and passed through the *tanh* function, and then the result is filtered by the *output gate*. This produces the short-term state $h^{\langle t \rangle}$ (which is equal to the cell's output for this time step, $y^{\langle t \rangle}$).

As shown in Fig. 3.2, NFVLearn is based on a multivariate, many-to-many LSTM architecture. Note that in Figs. 3.1, 3.2 and 3.3, LSTM cells are depicted as yellow "boxes", while the inputs are in green and the outputs, in orange.

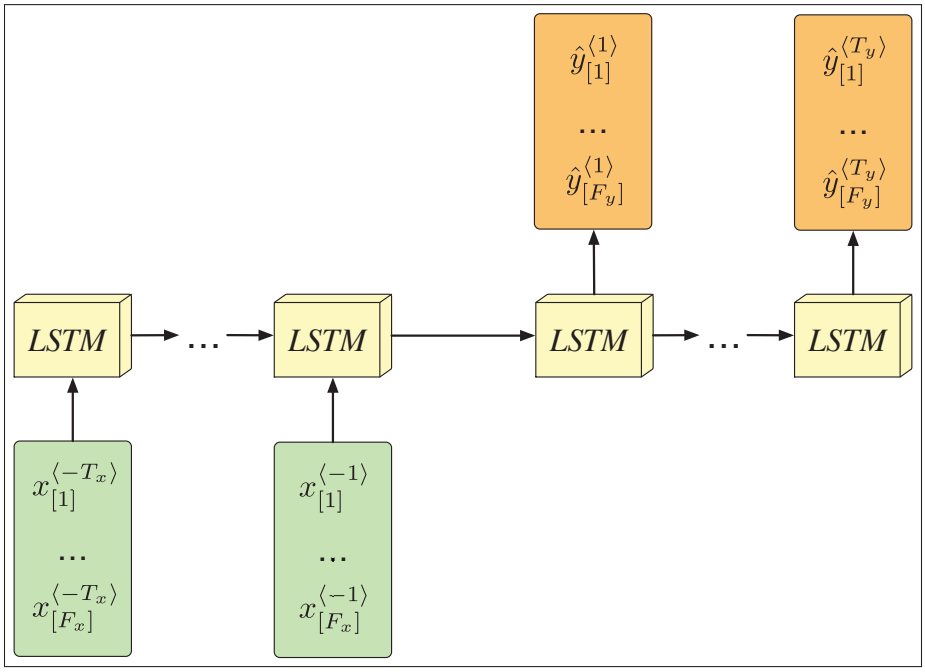


Figure 3.2 Proposed LSTM Architecture

The novelty of NFVLearn's architecture is that it draws a sample of resource usage history $x_{[F_x]}^{\langle T_x \rangle}$ of T_x time steps from a selected number of input VNF resource attributes F_x (eq. 3.4) of widely different scales to forecast a resource usage horizon $\hat{y}_{[F_y]}^{\langle T_y \rangle}$ of T_y time steps from a selected number of output VNF resource attributes F_y (eq. 3.5) which can be totally unrelated to the input features, both in scale and in the type of resource attributes targeted. To do so, an extra step needed to be applied in order to generalize predictions from several output features accurately: output labels are separately normalized using MinMax [0,1] (fit-transform in training and transform in validation), then those weights are stored to ensure that the predicted resource

usage features can be later “un-normalized”. To date, no reference to this extra step has been referenced in the literature. This makes NFVLearn a generic, state-of-the-art approach that could well be ported to other types of multivariate time series prediction architectures with minor adjustments.

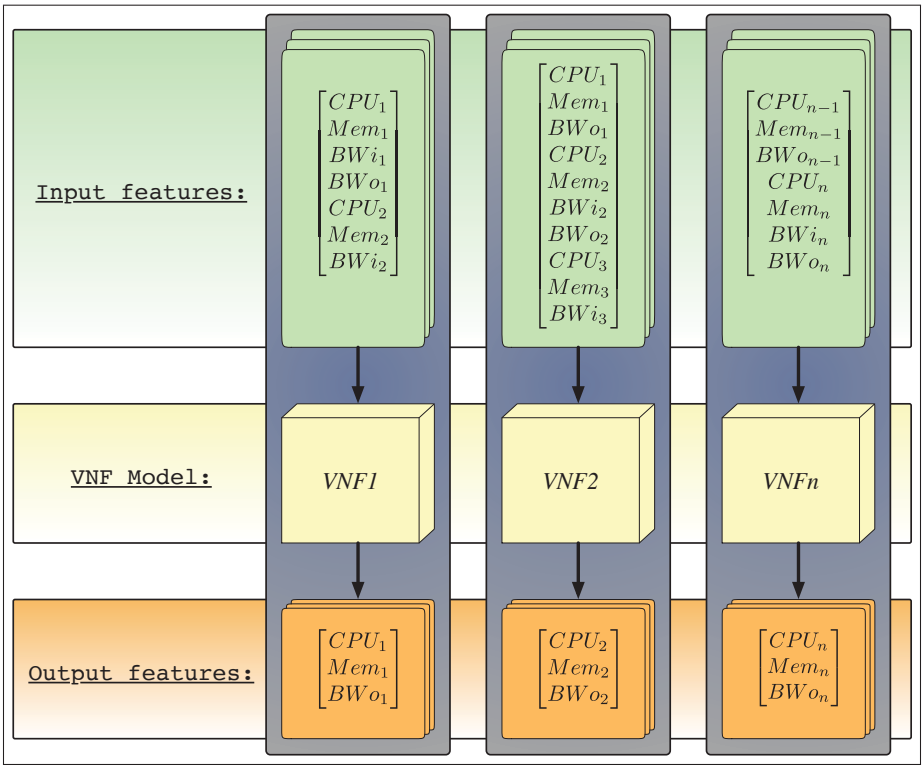


Figure 3.3 GNN feature selection by VNF location in an SFC directed graph

3.4.3 Graph Neural Networks

NFVLearn’s GNN setup follows the principles of the GNN theory. The concept of GNNs extends existing neural networks for processing the data represented in graph domains. In a graph, each node is naturally defined by its features f_n (i.e., the VNF’s own resource attributes) and those of k neighbouring nodes f_m (i.e., the resource attributes of the VNF’s neighbours). In this article, we have set the number of neighbouring nodes to $k = 1$. The aim is to learn a hidden state embedding h_n, which contains the neighbourhood information for each node n. The

hidden state embedding can then be used to produce an output label o_n for the same node. Let ϕ be a parametric function called the *local transition function* that is shared among all nodes and updates the node state according to the input neighbourhood, and let γ be the *local output function* that describes how the output is produced. Then, h_n and o_n are defined as follows:

$$h_n = \sum_{m \in k} \phi(f_n, f_m, h_m) \forall n \tag{3.8}$$

$$o_n = \gamma(h_n, f_n) \forall n \tag{3.9}$$

Equations 3.8 and 3.9 represent the activity of a network consisting of units which compute h_n and o_n for a number of time steps for each node. Those equations represent the main idea of GNNs; an information diffusion mechanism in which a graph is processed by a set of units, each corresponding to a node in the graph, linked according to the graph connectivity. Since NFVLearn benefits from an LSTM architecture, the update of the nodes' hidden states h_n and output labels o_n at each time step is entirely automated. This simply leaves the task of defining F_x input features f_{in} and F_y output features f_{out} for each model, or VNF node, as shown in Fig. 3.3, such that:

$$f_{in} = [f_n, f_m], \forall m \in [-k, k] \tag{3.10}$$

$$f_{out} = [f_n] \tag{3.11}$$

Given the resource attributes available for a VNF node in this article (*CPU, memory, input bandwidth, output bandwidth*), the number of input features for a model predicting resource usage for an edge node will always be $F_x = 7$ and for an inner node, $F_x = 10$. On the other hand, the number of output features for a model to predict will generally be $F_y = 3$, except for some

instances where the VNF end node of the SFC does not transmit network traffic (i.e., *output bandwidth*). In that case, $F_y = 2$.

Algorithm 3.1 Automated Input Feature Selection

```
1  Algorithm: InFeatureSelect(FM[], outFeaturesList[])
   Input: FM[]:= Resource usage data frame of dimension [timeStep, VNFresourceAttribute],
          outFeaturesList[]:= List of output resource usage attribute column names.
   Output: inFeaturesList:= list of input resource usage attribute column names.

2  Initialize inFeaturesList[]
3  foreach output feature o ∈ outFeaturesList[] do
4      Initialize correlation coefficient results array cor[]
5      Initialize inFeaturesPerNode[]
6      foreach feature f ∈ list(FM.columns) do
7          Compute correlation coefficients c using equation 3.1, 3.2, or 3.3
8          Compute p if using equation 3.2 (p-value) or equation 3.3 (τ)
9          cor_{o+1} ⟵ f, c, p
10     end foreach
11     Sort cor[] by highest to lowest |c|
12     while numFeatures ∈ inFeaturesPerNode < 2 do
13         if |c| >= 0.3 AND p <= 0.1 then
14             append f in inFeaturesPerNode[]
15         end if
16     end while
17     append inFeaturesPerNode in inFeaturesList[]
18 end foreach
19 return inFeaturesList[]
```

3.4.4 Pruning Setups

The pruning setup is relatively similar to the GNN setup described in section 3.4.3, with the difference that input features f_{in} are selected by algorithm 1. This process is applied to each of the three pruning setups to select the best input features from an SFC and provide accurate resource usage prediction for any desired output features.

As described in Algorithm 1, the first step (lines 3 to 10) is to apply the associated equation for each of our pruning setup options described in Section 3.3. The second step is to select the best input features (lines 11 to 16).

3.5 Experiment

This section presents the main performed experiments pertaining to NFVLearn and its input feature engineering mechanisms, and obtained results.

3.5.1 Data

Big instances of VNFs, such as an IP multimedia subsystem (IMS), may be composed of smaller instances of software components, called virtual network function components (VNFCs). A VNFC is an internal component of a VNF which provides a defined sub-set of that VNF's functionality, with the main characteristic that a single instance of this component maps 1:1 against a single virtualization container. This means that the VNFCs in a VNF are linked to each other by a combination of directed and undirected links and work together to provide the required functionality of the VNF. This matter irrevocably poses a challenge for someone aiming to collect resource usage metrics from those VNFCs since some ad hoc measures have to be taken to filter the internal network traffic between multiple VNFCs of the same VNF. For this experiment, we have chosen to combine the resource usage metrics of all VNFCs into single VNFs and redefined SFCs into directed graphs as shown in Fig. 3.4 and a complete description of the composition of the SFCs used in this work can be found below and in Fig. 3.5.

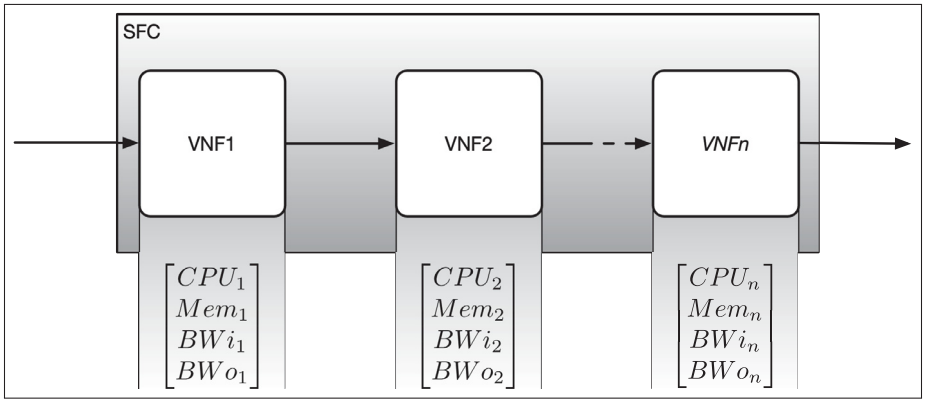


Figure 3.4 Synthesized SFC Directed Graph

The proposed approach was tested on one IMS and one Web scenario, both built upon a bare-metal Kubernetes testbed. The setups for both IMS and Web SFCs are shown in Fig. 3.5.

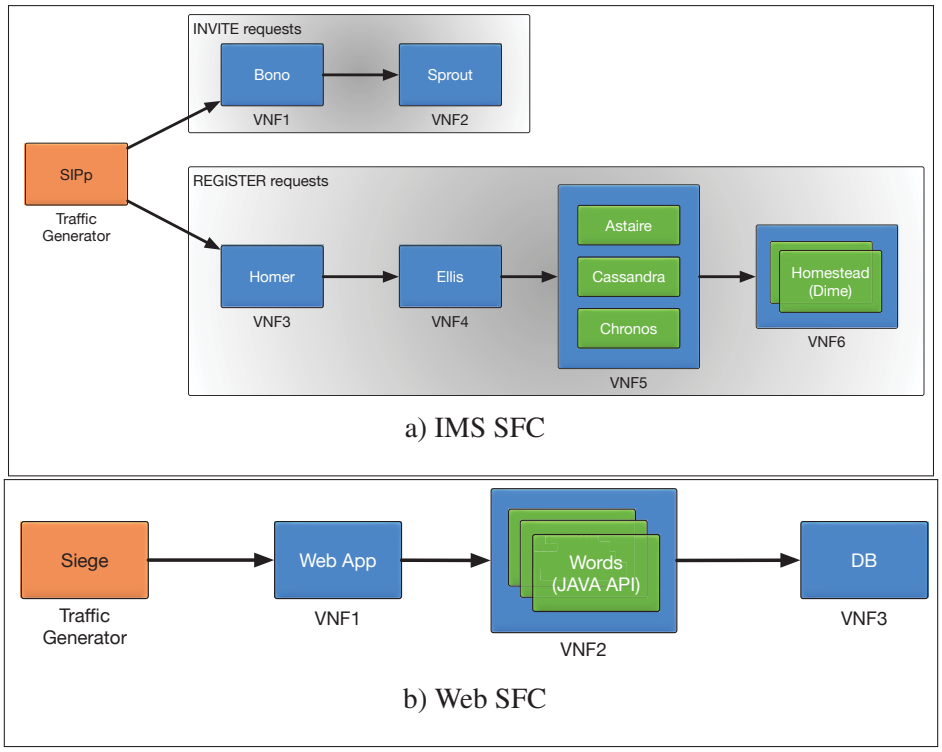


Figure 3.5 Proposed IMS and Web Designs

For instance, the IMS SFC deployment comprises a Clearwater Core IMS virtual environment organized in a VNF-FG split into INVITE and REGISTER requests. VNF instances and components have been labelled according to their position in the forwarding graph (VNF1 and VNF2 in the INVITE request FG, and VNF3, VNF4, VNF5 and VNF6 in the REGISTER request FG). A SIPp traffic generator handles the generation of both request types to the SFC. SIPp is an open-source traffic generator and test tool for the SIP protocol. We collected CPU, memory, input bandwidth and output bandwidth resource usage data from VNF1 to VNF5 and all but output bandwidth resource usage data from VNF6 for a total of 23 resource attributes.

The Web SFC deployment is a virtualized web server environment comprised of a Go web application (VNF1) attached to a virtualized switch serving as a load balancer. That load balancer is then tied to three Java REST API instances (VNF2), which serve words read from a

Postgres SQL database (VNF3). The network traffic originates from a Siege traffic generator. Siege is an open-source regression test and benchmark utility. It can stress test a single URL with a defined number of simulated users. We collected CPU, memory, input bandwidth and output bandwidth resource usage data from VNF1, and all but output bandwidth resource usage data from VNF2 and VNF3, for a total of 10 resource attributes. In this scenario, we could not collect output bandwidth resource usage data from VNF2 because the network monitor was incorrectly scanning the network traffic on the load balancer between VNF1 and VNF2 instead of the exit node of VNF2.

Each pod in the IMS and web scenarios has three vCPUs, 2 GB of RAM, and 10 Mbps links. Workload generated by SIPp and Siege both varies linearly, with sharp increase/decrease occurring around 10% of the time with VNF resource usage metrics (vCPU, vRAM, i/o bandwidth) collected every 20 seconds in a span of two weeks, giving over 80.000 timestamped raw/unfiltered data points which were used to train and validate our LSTM models.

The authors understand from extensive experience that modelling seasonal and cyclic workload trends, patterns, and bursts while considering outlier filtering is a monumental task. Moreover, we must consider the highly heterogeneous nature of the underlying infrastructure, including but not limited to VNF software (vendor, architecture, version), physical hardware, as well as NFVI (microservice architecture, virtualization, NFV-MANO).

Since VNF resource usage prediction using Deep Learning methods is fundamentally tied to the provided resource usage data (i.e., our approach is “data-driven”), and the data provided by our industrial partners were sequentially collected every 20 seconds, the scope of this experiment is to demonstrate the effectiveness of NFVLearn at forecasting mid to long-term resource usage of VNFs in an SFC (e.g., long-term periodic cycles, seasonal trends, etc.).

In this regard, while we acknowledge that this experiment does not cover short-term bursts, nor that it demonstrates its effectiveness in accurate time series-based prediction of varying time lengths, the results presented in this chapter are aimed to be presented in an angle showing that the proposed approach will perform well under real-world traffic and under any circumstances,

with a strong emphasis at showing that those predictions are reinforced by leveraging correlated input data interdependencies.

3.5.2 Tools and Model Training

Training of the models was performed on a computer with a 6-core Intel Core i7 10710U CPU with each core clocked at 1.61 GHz, 32 GB of RAM and an eGPU module hosting an Nvidia GeForce RTX 2080 Ti GPU with 12 GB of NVRAM. The learning environment was set up natively on Ubuntu Linux OS 18.04.5 LTS with Nvidia's CUDA 10.1 GPU drivers.

NFVLearn was developed using the Python 3.8 programming language on a TensorFlow 2.2 platform with the TensorFlow-GPU library and Keras 2.4.2. Several common libraries in machine learning were used, such as NumPy 1.19.0, SciPy 1.4.1, Pandas 1.0.5 and Sci-Kit-Learn 0.23.1. MatPlotLib was used for data visualization.

Table 3.1 Hyperparameters

Hyperparameter	**Value**
# input timesteps T_x	[5, 10]
# output timesteps T_y	$[1-6]$
Normalization	MinMax [0,1]
# hidden layers	1
# units per layer	50
Optimizer	ADAM
Regularizer	L1 L2 regularization
Learning rate	0.001
β_1	0.9
β_2	0.999
ϵ	1e-07
Batch size	512
Loss function	MSE
Validation set size	5% of the dataset
Early stopper	10

All trained models used the same hyperparameters shown in Table 3.1 for comparison purposes. The number of hidden layers, the number of units per layer, the regularization process as well

as its sub-parameters (ADAM optimizer's learning rate, β_1, β_2 and ϵ) and the early stopper value were selected through a grid search process that yielded the best overall MSE validation loss. For the normalization process, NFVLearn uses an unorthodox technique: before training our models, we separately fit and transform both F_x input features and F_y output features by using MinMax normalization with a range of $[0, 1]$. The key idea behind this technique is that we use different input/output features of widely different ranges ($[0.00, 3.00]$ CPU usage ratio, $[0, 4.000.000.000]$ Bytes of RAM usage and $[0, 10.000.000]$ bps of I/O bandwidth usage). Hence, we aim to uniformly scale down the values of all output features to then uniformly derive the weights of the neural nodes through gradient descent during back-propagation. The fit weights of the MinMax normalization for input and output features are then stored to "un-normalize" predicted values later on when using our trained models for prediction.

Moreover, two model subsets, one of $T_x = 5$ input time steps and another of $T_x = 10$ input time steps were created for evaluation purposes in this chapter. Both subsets were trained to predict a varying resource usage forecasting horizon of $T_y = [1 - 6]$ time steps. Our intuition was to assess the RMSE forecast degradation per time step given a different number of resource usage history. Since resource usage metrics were collected every 20 seconds, we estimated that resource usage forecasting between 20 to 100 seconds was adequate for mid-term resource usage prediction given a good RMSE score in an NFVI, giving ample time to predict and then re-allocate VNF resources by using another mechanism if need be.

3.5.3 Pre-processing: Input Feature Selection and Output Features

Input feature selection differs from one setup to another. This section provides information about the selection process for each setup proposed in this chapter.

Output features, however, remain the same for each model, no matter what setup we use. Tables II-3 and II-4 provided in Appendix II show the output features of each model from the IMS and Web SFCs.

3.5.3.1 Default setup

The Default setup is designed to incorporate all available resource attributes from an SFC as input features for every VNF model. This method is the simplest from a design perspective since all the inputs are the same for every model. Only the output features change to match the desired VNF resource attributes we wish to predict.

3.5.3.2 GNN setup

The input feature selection from the GNN setup follows the procedure described in Section 3.4.3 and as depicted in Figs. 3.3 and 3.4. Hence, the IMS SFC is composed of two directed graphs: the INVITE requests graph and the Register requests graphs (Fig. 3.5a). On the other hand, the Web SFC is composed of one directed graph (Fig. 3.5b).

3.5.3.3 Pruning setups

Automated input feature selection from any of the three proposed pruning approaches (Pearson correlation coefficient, Spearman's and Kendall's rank correlation coefficients), as described in Section 3.3, follows the steps detailed in algorithm 1. For convenience, we have provided the selected input features from each approach in Tables II-1 and II-2 described in Appendix II. Since the input feature selection process is the same for those three methods, a user can conveniently discover the best set of input features from each pruning setup. He can then select the best set by comparing each model's accuracy by using Aikake Information Criterion (AIC) or Bayesian Information Criterion (BIC), as shown in Section 3.5.5.

Observations for both Tables II-1 and II-2 highlight distinct trends. For instance, IMS SFC I/O bandwidth resource attributes for all VNFs show a very high correlation between each other and CPU resource usage, showing that this SFC is highly network-oriented. Hence, strong inter-dependencies of those resource attributes may reinforce resource usage prediction of most VNF resource usage attributes of the IMS SFC, even if those VNFs are not directly connected in the VNF-FG. On the other hand, the web SFC's VNF memory resource attributes show similar,

albeit inversely correlated trends between each other and CPU resource usage. Those results show that the web SFC is more data-oriented, with memory buffering occurring across the SFC, demonstrating the data transfer from the web server database to the end user.

3.5.4 Comparisons

The proposed approach was compared with the five input selection setup alternatives (Default, GNN, Pearson, Spearman, Kendall) described in Section 3.5.3. The main difference between these setups given in this article is the choice and number of input features F_x used in the models to predict the same output features F_y. In order to compare the prediction accuracy of each setup, the RMSE defined in eq. 3.12 and the coefficient of determination R^2 defined in eq. 3.13 were used. The principle behind this selection of metrics is that RMSE offers a good measure of the differences between values, while R^2 provides a measure of how well observed outcomes are replicated by the model, based on the proportion of total variation of outcomes explained by the model.

$$RMSE(y, \hat{y}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{3.12}$$

Where,

- y is the observed value
- $\hat{y}$ is the estimated value
- n a set of values to evaluate

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \overline{y}_i)^2} \tag{3.13}$$

Where,

- y is the observed value
- $\hat{y}$ is the estimated value
- $\overline{y}$ $= \frac{1}{n}\sum_{i=1}^{n}(y_i)$
- n a set of values to evaluate

Moreover, since we propose four novel input feature selection techniques, we also propose non-linear comparison techniques based on Aikake Information Criterion (AIC) and Bayesian Information Criterion (BIC) that can be used to quickly and efficiently select the best model among each technique, for each VNF.

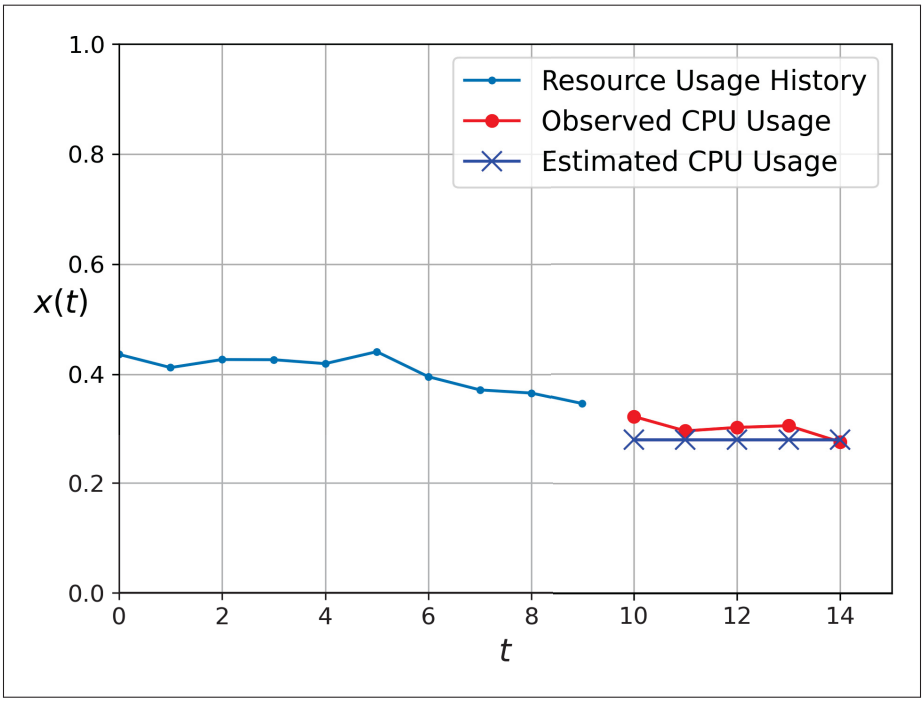


Figure 3.6 Web VNF3 CPU resource usage prediction example using a Spearman pruning setup

3.5.5 Results

The results of our evaluations are shown in Figs. 3.6-3.12 and Tables 3.2 and 3.4. Fig. 3.6 shows an example from a Web VNF3 model prediction with a Spearman setup that depicts how our architecture estimates a resource usage horizon. In this instance, a resource usage history of 10 time steps from the resource attributes CPU_3, Mem_3 and CPU_1 are used to predict 5 time steps of the resource attributes CPU_3 and Mem_3. Only CPU_3's resource usage history, observed resource usage horizon and estimated resource usage horizon are displayed for convenience.

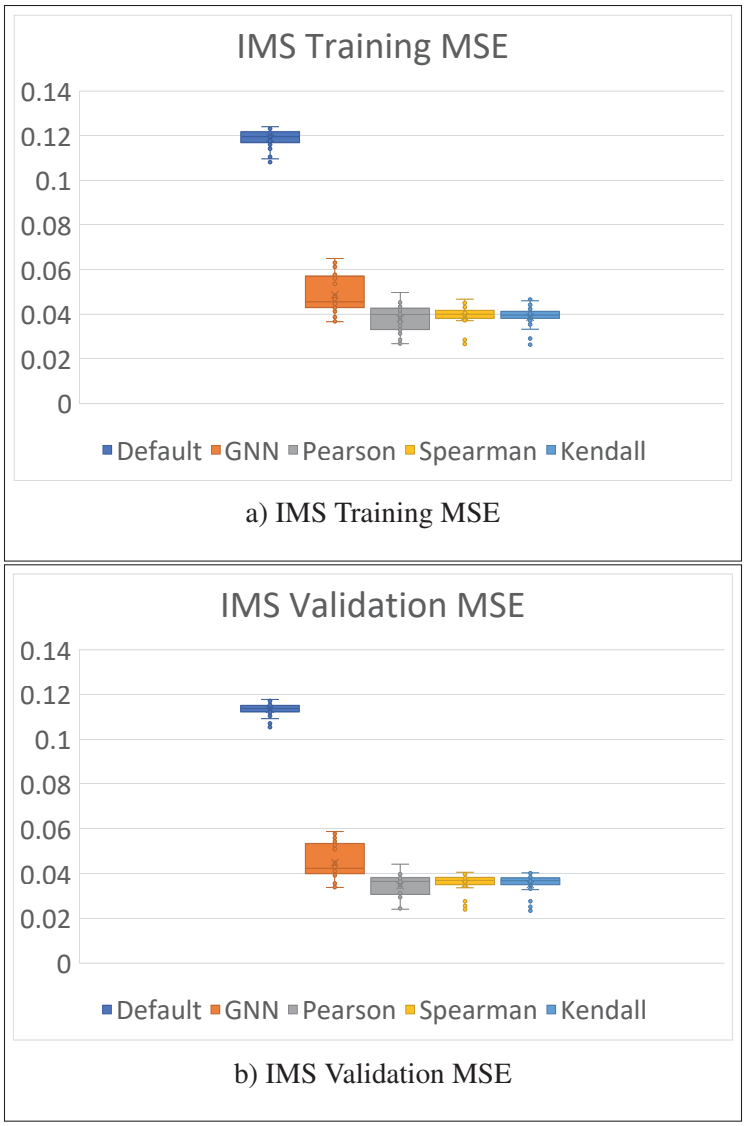
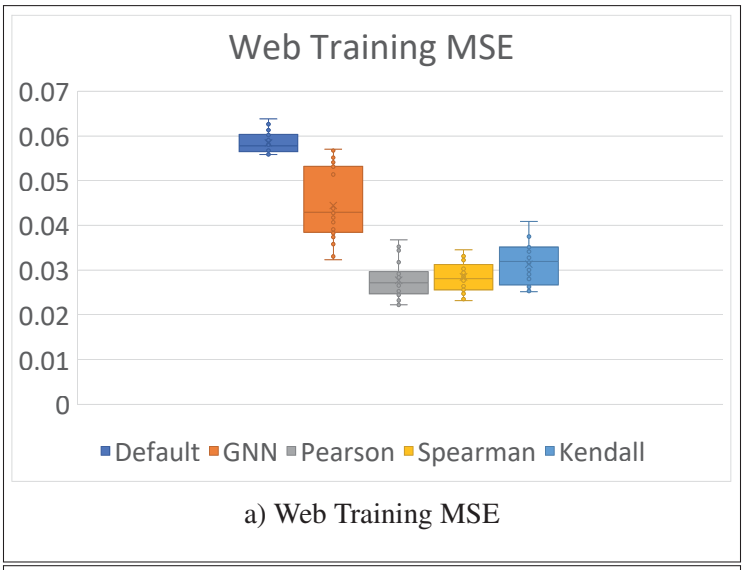


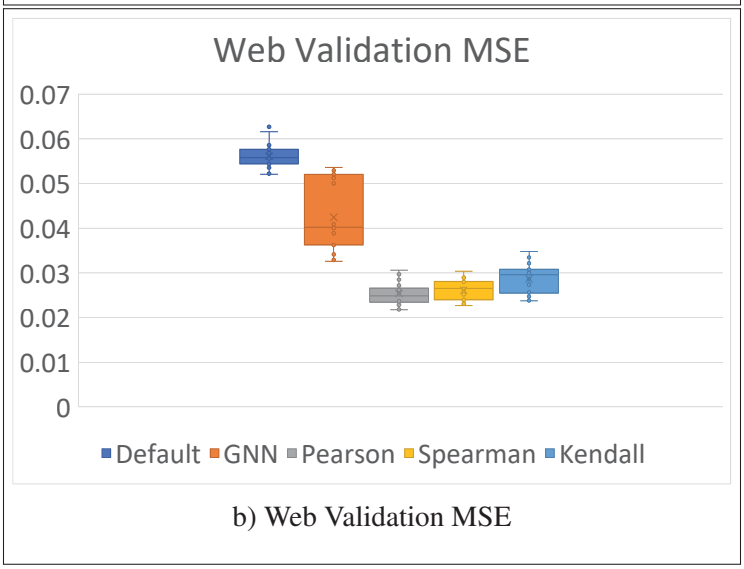
Figure 3.7 Whisker boxes of IMS training and validation MSE

Figs. 3.7, 3.8 and Tables 3.2 and 3.3 show validation and training MSE loss from both our IMS and Web models after repeated training of each VNF model, for each proposed setup, in order to get a clearer overview of the general accuracy of the setup mechanisms. That gives us 300 trained models for the IMS testbed and 150 models for the Web testbed to enhance the mean, median, variance, and the quartiles of the four displayed whisker boxes of Figs. 3.7 and 3.8. Figs. 3.9, 3.10 and 3.11 depict the R^2 and RMSE scores for each output feature from all IMS

and Web VNF models. Lastly, Fig. 3.12 shows RMSE over several forecasted time steps that depicts degrading accuracy over time.



a) Web Training MSE



b) Web Validation MSE

Figure 3.8 Whisker boxes of Web training and validation MSE

Fig. 3.6 shows that estimated resource usage is close to the observed values and that those estimated values follow a regression curve rather than random dots. This is due to the large number of training samples used for the VNF models, which significantly reduces the variance of the predictions. A smaller training sample size tends to increase the variance of the predictions.

In Figs. 3.7 and 3.8, results show that training (Figs. 3.7a and 3.8a) and validation (Figs. 3.7b and 3.8b) MSE loss from the three proposed pruning setups slightly outperform the GNN setup, and significantly outperform the default setup. This is explainable because models based on pruning setups possess less but more strongly correlated input features than the other proposed approaches. That observation implies that more input features overall lead to weaker MSE loss of our estimations, both on training and validation sets and for both IMS and Web testbeds.

Another observation is that MSE loss mean values from the pruning setups are extremely close to one another, with the GNN setup mean values of the IMS testbed not far behind. These results are partly explainable since, in most instances, few (if any) input features change from one setup to another for most models. We note, however, that the Pearson setup whisker box is favourably skewed towards lower (and more accurate) MSE loss results in all cases. Tables 3.2 and 3.3 give the accurate mean MSE losses for each setup.

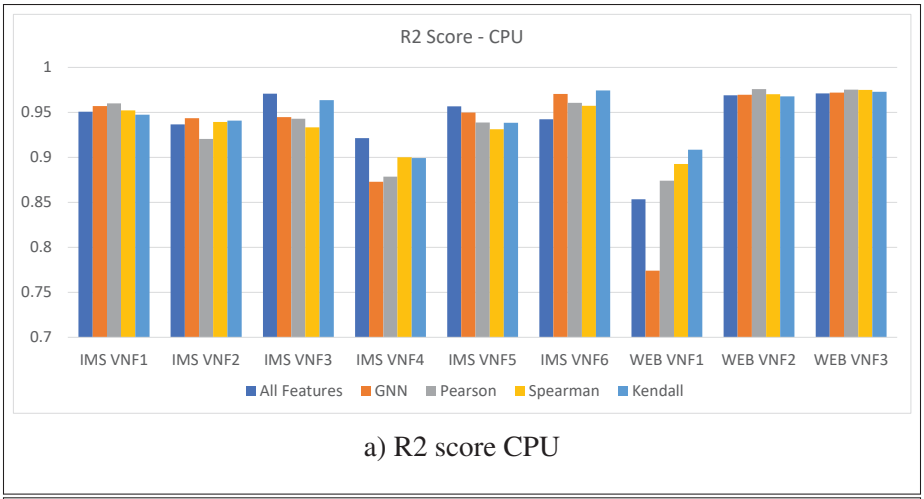
Table 3.2 IMS Input Feature Setups

Setup	**Training Loss (MSE)**	**Validation Loss (MSE)**
GNN	0.04492	0.04868
Pearson	0.03456	0.03801
Spearman	0.03528	0.03883
Kendall	0.03509	0.03870

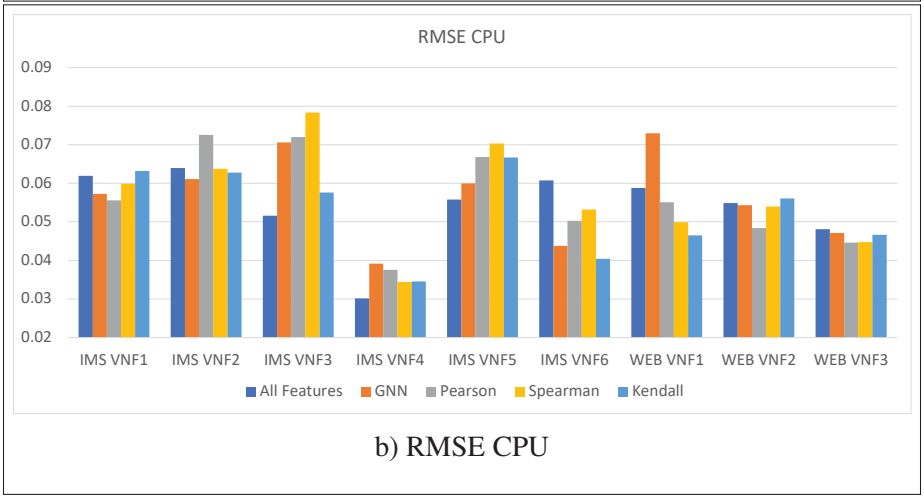
Table 3.3 Web Input Feature Setups

Setup	**Training Loss (MSE)**	**Validation Loss (MSE)**
GNN	0.04243	0.04443
Pearson	0.02535	0.02774
Spearman	0.02610	0.02849
Kendall	0.02867	0.03135

Next, Figs. 3.9, 3.10 and 3.11 depict R_2 scores and RMSE per VNF model for each targeted output feature. For instance, Figs. 3.9a-3.9b show results for each IMS and Web VNF model for the CPU output feature, Figs. 3.10a-3.10b for the memory output features and Figs. 3.11a-3.11b for the BWo output features.



a) R2 score CPU



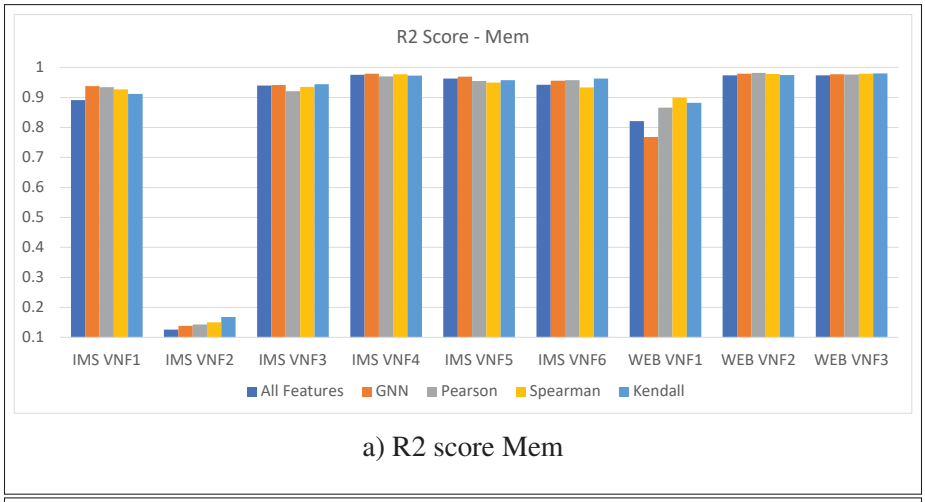
b) RMSE CPU

Figure 3.9 R2 score and RMSE per VNF CPU

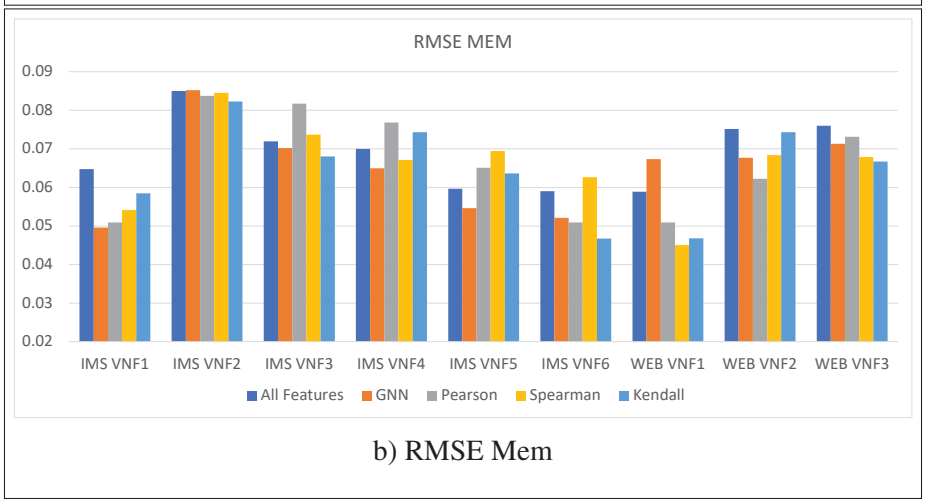
Observations of the R^2 scores shown in Figs. 3.9a, 3.10a and 3.11a demonstrate that predictions are very well replicated by our models (R^2 >= 0.90) with only a few notable exceptions. For instance, Web VNF2 models built using the GNN setup show R^2 scores below 0.80, meaning that CPU, memory and output bandwidth predictions for this VNF do not provide a reliable level of fidelity to the expected outputs.

Next, observations of the R^2 scores of the IMS VNF2 models built using the five proposed setups make us draw a similar conclusion. In this case, R^2 scores are extremely low (below 0.20), meaning that the observed outcomes do not replicate the IMS VNF2 models. Another important observation leads us to believe that models built using any of the three pruning setups

(Pearson, Spearman and Kendall) provide better R^2 scores overall compared to the GNN and Default setups.



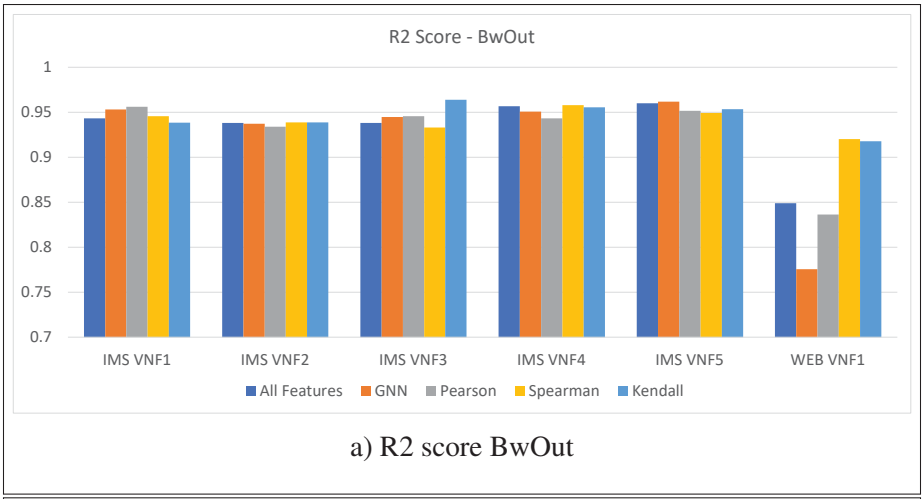
a) R2 score Mem



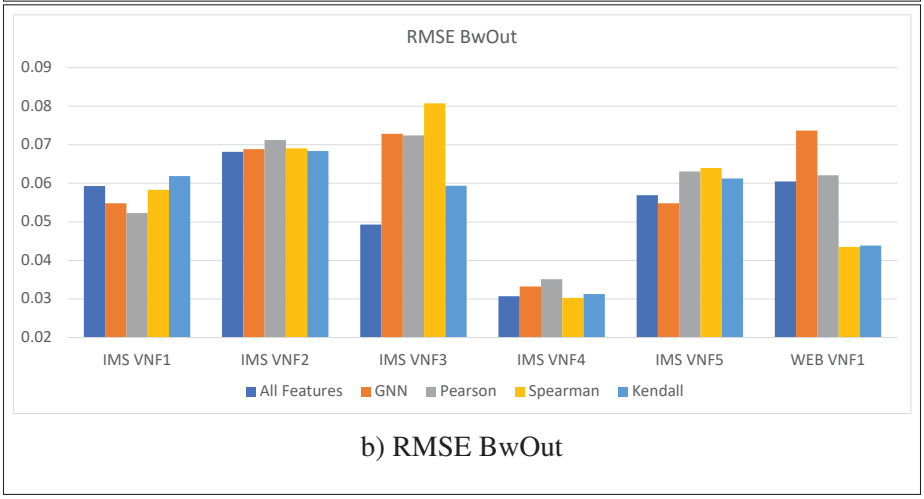
b) RMSE Mem

Figure 3.10 R2 score and RMSE per VNF Memory

RMSE scores depicted in Figs. 3.9b, 3.10b and 3.11b show a low error magnitude for all predicted resource usage attributes and for any type of input feature setup. However, by combining the R^2 scores of the different models for some predicted resource usage attributes (i.e., CPU, memory and Bw_o usage of Web VNF1), we denote that models having the lowest level of fidelity (low R^2 scores) also replicate poorer prediction accuracy (higher RMSE).



a) R2 score BwOut

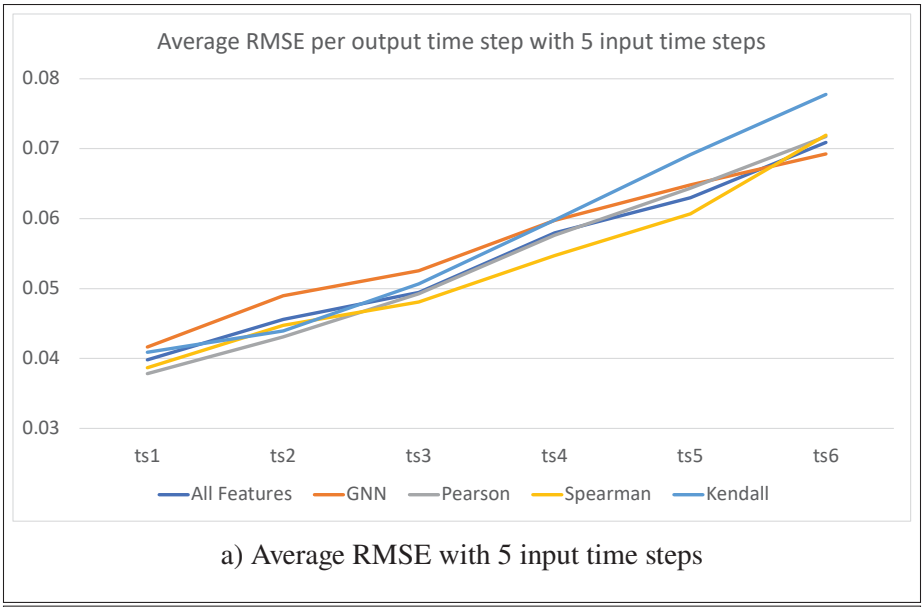


b) RMSE BwOut

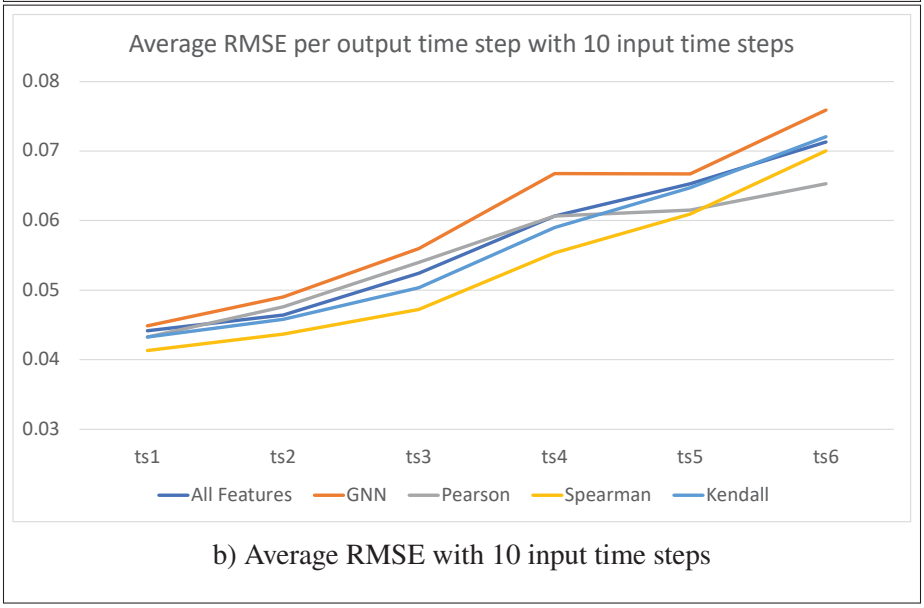
Figure 3.11 R2 score and RMSE per VNF BWo

Fig. 3.12 depicts the average RMSE per up to 6 predicted time steps for all output features from every model. Fig. 3.12a shows results given a history of 5 time steps from input features, while Fig. 3.12b shows results given a history of 10 time steps from input features.

Results show no benefits from taking a resource usage history of 10 time steps instead of 5 time steps to forecast future resource usage with NFVLearn. Moreover, results indicate that pruning setups using less, highly correlated input features (most notably Spearman) carry better accuracy over multiple time steps.



a) Average RMSE with 5 input time steps



b) Average RMSE with 10 input time steps

Figure 3.12 Average RMSE per time step

Overall results show that NFVLearn is an efficient LSTM-based mechanism that can forecast resource usage of several concurrent resource attributes of an SFC with great accuracy and fidelity over multiple time steps independently of the input features given to a model. Moreover, the proposed input feature selection mechanisms show that it is possible to reduce NFVLearn's overhead in the control plane, significantly reducing the number of needed input features and

resource usage history, all the while not negatively impacting the accuracy and fidelity of the resource usage predictions.

Finally, Table 3.4 shows each model's AIC and BIC results for each input feature selection setup. This analysis is a quick and easy way to sort each input feature selection setup from the best (lowest score) to the worst (highest score). Results demonstrate that the default and GNN setups are generally less efficient than any pruning setups in most cases, while the Pearson pruning setup shows to be the most efficient 6 out of 9 times. In all cases, either Pearson, Spearman or Kendall setups are the best-performing models in both AIC and BIC evaluations. Those results demonstrate the efficiency of the input feature selection setups based on highly correlated data.

Table 3.4 AIC and BIC for each model

Model	Default		GNN		Pearson		Spearman		Kendall	
	AIC	BIC	AIC	BIC	AIC	BIC	AIC	BIC	AIC	BIC
IMS VNF1	18014	18109	10543	10637	**8986**	**9080**	10946	11040	10432	10526
IMS VNF2	15562	15656	9951	10045	**5576**	**5670**	8338	8432	7062	7156
IMS VNF3	14510	14604	5850	5944	9512	9606	7748	7842	**−1029**	**−935**
IMS VNF4	16199	16293	11575	11669	**4680**	**4774**	6614	6708	5515	5609
IMS VNF5	13394	13488	10286	10380	**1425**	**1519**	2573	2667	3286	3380
IMS VNF6	22105	22168	16192	16255	9642	9705	**7558**	**7621**	15354	15417
WEB VNF1	16414	16508	13716	13811	2824	2918	2840	2934	**2796**	**2890**
WEB VNF2	13095	13153	12755	12816	**−8024**	**−7963**	−7773	−7714	−3772	−3712
WEB VNF3	13110	13173	10029	10092	**−7922**	**−7859**	2287	2350	−3208	−3145

3.6 Discussion

The aim of this work was to provide CSPs and TSPs with a generic and autonomous LSTM design framework to forecast the resource usage of VNFs and SFCs in a virtualized infrastructure. LSTM was selected thanks to its demonstrated ability to enhance prediction accuracy by leveraging deep relationships – or by what we call it in this work, interdependencies- between several highly correlated resource attributes like CPU, RAM, disk I/O, I/O network bandwidth between VNF nodes, etc.

Throughout this work, we have found evidence that SFCs have specific low-level resource usage patterns; we could see several correlated patterns between seemingly unrelated resource usage

features. For instance, IMS INVITE requests are highly CPU and bandwidth-intensive on some of its VNFs, whereas IMS REGISTER requests highly rely on disk I/O and memory on some other VNFs. It thus became apparent that we could improve the forecasting accuracy of certain resource usage features of a VNF by selecting some of those resource attributes from remote VNFs (i.e., output bandwidth of a VNF servicing large data packets may help to predict disk I/O of a remote database server in the same VNF forwarding graph). Hence came the idea to generate an automated input feature selection mechanism that could provide the best resource usage attributes that improve the forecasting accuracy of the resource usage from resource attributes selected by a user. To date, we haven't found any instance of resource usage forecasting mechanisms that leveraged correlated features in this manner in cloud or NFV environments.

To summarize, NFVLearn is a software solution meant to be developed by telecom and software vendors, but that could be operated by data scientists working at the said vendor (granted that the service provider -the customer- provides the relevant resource usage data) or by systems administrators, data scientists working for the service provider (granted that the customer has the proper on-premise machine learning infrastructure). It is totally VNF and SFC agnostic, meaning that it can be applied on any VNF, without prior knowledge of the function, software, vendor, or the underlying physical infrastructure in which it operates.

The resource usage forecasts of those VNFs (aimed in this work to be mid-to-long-term) can then be used by resource adaptation mechanisms in the infrastructure to properly scale and migrate VNFs to adequately "balance" the physical resource load in a data center, a point-of-presence (PoP), an edge network, etc.

Hence, our work is a contribution that proposes a novel technique to build LSTM-based resource usage forecasting models for cloud and telecom service providers. The mid-to-long-term forecast resource usage from our models can then be used by VNF resource adaptation approaches, which then benefit from an accurate, multivariate, many-to-many, mid-to-long-term resource usage prediction enhanced by resource usage interdependencies.

Results demonstrate that NFVLearn efficiently enables accurate prediction of concurrent resource attribute consumption from VNFs in an SFC. It allows the selection of the needed number of input features and resource usage historical data to reduce NFVLearn processing overhead in the control plane, which, concurrently, enables NFVLearn to be integrated within various proactive decision-making processes. The usefulness of this approach can be demonstrated in many concrete scenarios.

NFVLearn deployment in constrained Edge environments is such a scenario. In resource-constrained environments, NFVLearn can be used to predict the load of multi-attributes resource-intensive VNFs (e.g., n out of m deployed VNFs with high CPU & I/O BW loads) and supports a proactive resource orchestration mechanism in taking the appropriate actions (e.g., migrating VNFs to other edge nodes) to meet the SFC's performance requirements. It can accurately predict the targeted resource attributes (e.g., CPU & I/O BW of n VNFs) for a short forecast horizon.

Another example is NFVLearn deployment in cloud environments with VNFs with affinity and anti-affinity features. In this scenario, NFVLearn can accurately predict VNF resource attributes with affinity and anti-affinity features (e.g., VNF1 is I/O BW and CPU intensive, and VNF2 is I/O BW and Memory intensive). It can help a proactive resource orchestration mechanism identify VNFs that should be kept together (e.g., VNF1 and VNF2 placed on the same host) or separate (VNF1 and VNF2 placed in separate hosts).

One last notable scenario could be within a cloud environment with VNFs using ephemeral resources: NFVLearn can predict ephemeral resources (e.g., disks) to help resource orchestration mechanisms manage time bounds for the usage of these resources and release them when they are no longer needed.

Throughout the different phases of our experiments (exploration, filtering and pre-processing of the data, hyperparameter tuning, analysis), much thought has been put into the value of NFVLearn in the domain of NFV and, most importantly, to the TSPs that require a precise resource usage prediction mechanism in their infrastructure. Among our concerns was reducing

human intervention to the bare minimum, the biggest challenge in modern LSTM techniques. To achieve this goal, a deep knowledge of the data and NFV as a whole is of utmost importance, but also of the benefits, limitations, deep intuition and understanding of LSTM and RNNs beyond the "bells and whistles." In this regard, we viewed NFVLearn's design on the axis of finding a solution to a problem and not building a solution in search of a problem. This viewpoint is crucial if the research community wishes to push deep learning to its full potential.

For instance, validating a model with an R^2 score and another metric such as MAPE, MSE, or RMSE is key in determining if it is even remotely possible to use NFVLearn for resource usage prediction of some attributes. Memory usage prediction of IMS VNF2 (Fig. 3.10a) validates that statement. In this example, interdependency-based resource usage prediction is not advised since memory usage of IMS VNF2 is not tied or bound to resource usage history from any other resource attribute of the SFC. A solution to this problem might be to use a more traditional method to predict future memory usage, like ARIMA, or consider discarding that resource attribute from the pool of legitimate resource attributes that can be estimated.

Moreover, other evaluation metrics beyond the scope of this work could help determine which specific input feature selection scheme to use. The frequency at which we collect VNF resource usage history and the resulting exponential growth of the traffic load in the control plane can be a significant concern in large NFVIs. The size of the infrastructure raises other concerns since it influences the number of hops that the resource usage metrics need to travel throughout the network, thus influencing latency. It is essential to precisely know our aim when selecting a resource usage history or a forecast horizon, leading us to ask crucial questions. What is the influence of the number of input/output time steps and the size per time step required for our models? What influence will it have on prediction accuracy and other thresholds of other ML-based mechanisms such as those used in VNF-RA? Those concerns open up new research tracks that may expand our contributions in future work.

3.7 Conclusion

This chapter proposed NFVLearn, a novel multivariate many-to-many LSTM-based resource load prediction model for NFV environments and four automated input feature selection setups. Based on GNN, Pearson correlation coefficient, Spearman rank correlation coefficient, and Kendall rank correlation coefficient theories. We evaluated those input feature selection setups by comparing each individual forecasted VNF resource load outputs for RMSE and coefficients of determination (R^2 score). Results show that leveraging VNF resource attribute interdependencies in an SFC helps build NFVLearn models with a lesser number of highly correlated input features while keeping high RMSE accuracy and high R^2 fidelity to the observed outputs.

CHAPTER 4

MULTIVARIATE OUTLIER FILTERING FOR A-NFVLEARN: AN ADVANCED DEEP VNF RESOURCE USAGE FORECASTING TECHNIQUE

The material presented in this chapter is part of the journal article accepted for publication cited below:

Cédric St-Onge[1] , Nadjia Kara[1] , Claes Edstrom[2]

[1] Department of Software and IT Engineering, École de Technologie Supérieure,
1100 Notre-Dame Ouest, Montréal, Québec, Canada H3C 1K3
[2] Ericsson Canada,
8275 Rte Transcanadienne, Saint-Laurent, QC H4S 0B6

Accepted for publication with revision in "The Journal of Supercomputing, Springer",
November 2022.

4.1 Introduction

Enhancing LSTM-based resource usage forecasting mechanisms is not a trivial task. Researchers not only need to acquire massive amounts of data from operating NFVIs, but they also need to devise ad-hoc mining and filtering techniques for this data with the help of both seasoned experts of the NFV domain as well as skilled Deep Learning practitioners with a vast experience in time series forecasting.

Hence, such large amounts of data require adequate filtering mechanisms in order to remove outliers efficiently. Doing so with the proper know-how brings huge performance benefits at the learning stage of LSTM models thanks to the reduced quantity of training examples. When carefully executed, it also makes trained models less prone to over-fitting without impacting the accuracy of the predicted outputs.

Therefore, it makes outlier filtering an important challenge for budding resource usage forecasting experts and as such, tackling this issue is the main challenge we want to tackle in this chapter. The next challenge is to build upon NFVLearn and improve its prediction accuracy.

The chapter is organized as follows. Problem description and motivation are described in Section 4.2. A description of data-driven resource usage prediction, correlation coefficients used in this work and the A-NFVLearn design, an enhanced version of NFVLearn with an attention mechanism, are presented in Section 4.3. Experiments and evaluation of our proposed techniques are described in Section 4.4. Finally, a discussion and a conclusion are presented in Sections 4.5 and 4.6 respectively.

4.2 Problem Description and Motivation

Resource usage forecasting based on Deep Learning techniques such as NFVLearn requires considerable amounts of historical resource usage data from VNFs. This data often travels intermittently throughout an NFVI's control plane, which puts pressure on the traffic load overhead in the substrate network. Although this is an unavoidable limitation of data-hungry DL implementations, it can, fortunately, be mitigated by carefully selecting the proper model architecture. As previously stated in chapter 3, two different types of implementation appear in the literature to mitigate this limitation. The first one is to run a VNF resource usage forecasting model on the NFV-MANO, which is a good option when an administrator wants to offload resource usage forecasting tasks away from the VNFs (e.g., when there are VNF processing, memory, power and disk size constraints, a low number of SFCs to manage in the NFVI, low network latency, large network bandwidth, a small number of network hops from the VNFs to the MANO, etc.) (Mijumbi *et al.*, 2017). The second implementation type is to run a VNF forecasting model locally on a VNF, which is a better option in large-scale NFVIs and remote data centers with large numbers of managed SFCs and VNFs, where outgoing resource usage data in the control plane could eventually cause network delays and bottlenecks (Cho *et al.*, 2020).

NFVLearn's current agnostic design (i.e., selection of the number of input and output time steps and their granularity, selection of input and output features) makes it extremely flexible in this regard and therefore makes it fit for different types of implementations. However, its LSTM-based architecture could benefit from the latest advancements in the DL field. For

instance, its ability to decipher interdependencies over several time steps of multiple resource usage features of historical data makes it prone to great forecasting accuracy improvements from an attention mechanism.

Moreover, in order to provide the most accurate predictions, VNF resource usage forecasting mechanisms typically require resource load history from multiple sources in order to truly benefit from resource attribute inter-dependencies of an SFC. Those mechanisms based on DL, however, are very sensitive to noise in the collected resource usage history required to forecast future resource load, especially for online prediction. That is why data pre-processing, outlier filtering and outlier removal are key in order to reduce variance in the training set, thus reducing under-fitting. This is especially critical in cloud computing and NFV since those environments host a wide variety of heterogeneous applications, services, tasks and processes, and SFCs can run along several server hosts and clusters at various points of the core and edge networks.

In this context, multivariate outlier filtering becomes a key mechanism to DL-based VNF resource usage forecasting techniques. It discards unhelpful training samples from highly correlated workload behaviour of several input features. Those unhelpful training samples, or outliers, seldom depict normal workload behaviour under stress and negatively influence prediction accuracy. Moreover, outliers cause underfitting of trained models.

Our aim in this chapter is to tackle these challenges. To reach this goal, our main contributions are the following:

- The study and investigation of an added attention mechanism to NFVLearn's architecture. We will then compare the training performance and resource usage prediction accuracy of this new design, called A-NFVLearn, with the results provided by NFVLearn's models.
- The study and investigation of AO, a novel multivariate outlier filtering technique, and its application at A-NFVLearn's pre-processing stage. We will compare the training performance and resource usage prediction accuracy of A-NFVLearn's models with and without AO.

- Ensure that this added outlier filtering technique is entirely automated and generalizes well with resource usage data from different types of SFCs and scenarios (IMS and web virtualized instances).

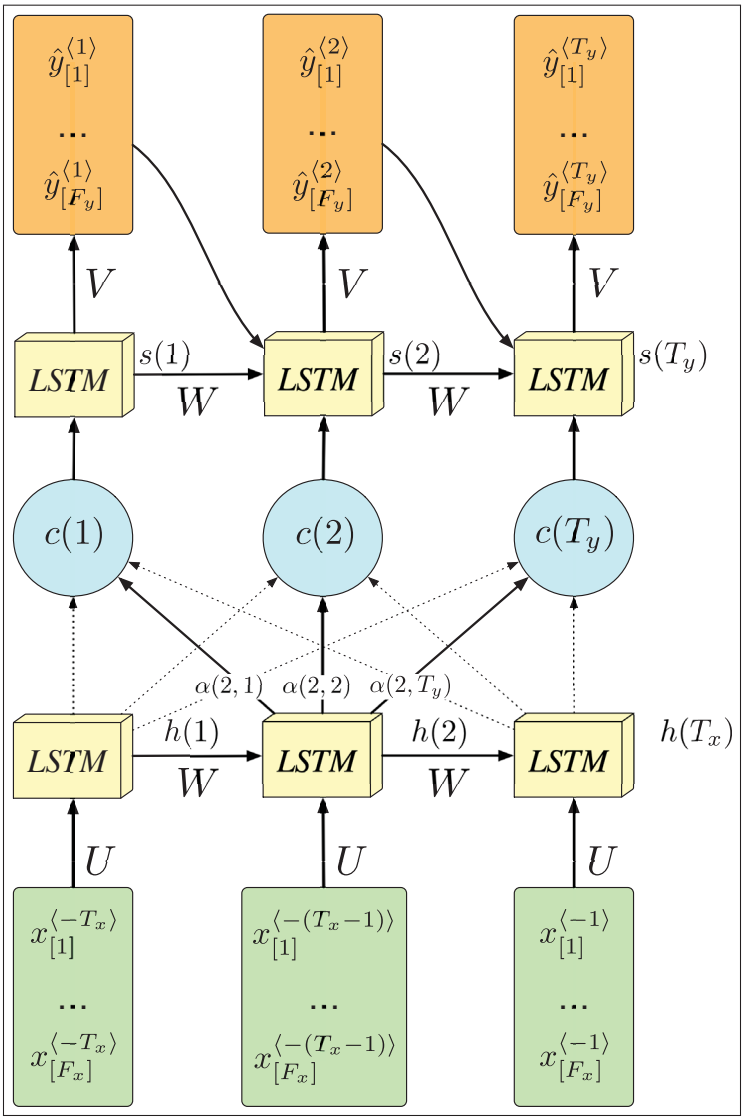


Figure 4.1 A-NFVLearn's Architecture and Data Structure

4.3 A-NFVLearn Design

In this section, we first discuss about the data structure of the collected resource usage data used for our experimentation. Next, we give detailed information about the attention mechanism enhancing A-NFVLearn's resource usage predictions. Lastly, we present the AO approach used to remove outliers from multivariate data. For reference, A-NFVLearn's data structure and architecture are depicted in Fig. 4.1.

4.3.1 Data Structure

Timestamped resource usage historical data is required in order to train A-NFVLearn models through supervised learning. This VNF Component (VNFC) and/or VNF resource usage data is collected from simulations at regular intervals and can be from any resource attribute: CPU, memory, input bandwidth, output bandwidth, etc. This stored data is then filtered, scaled and organized in a way that a column represents the resource attribute of a VNF and a row represents a time stamp at which the resource usage occurred. Data reorganized in this manner will later simplify data pre-processing as required by A-NFVLearn's architecture; for instance, it allows us to:

1. label F_x input features and F_y output features from any of the resource attribute columns of the dataset and,
2. for each labelled input features and output features, create resource usage sliding windows of size T_x input time steps and of size T_y output time steps.

This process results in the generation of training examples which can further be split into training and validation sets, with each training example being a set of two matrices: the input matrix and the output labels matrix.

The input matrix of a training example (eq. 3.4), of dimension $[T_x, F_x]$, and the output label matrix of a training example (eq. 3.5), of dimension $[T_y, F_y]$, are shown in chapter 3.

The choice of values for T_x and T_y is at the will of the user prior to model training. The number of input/output features and time steps is based on a number of factors such as prediction accuracy, the length between input/output time steps, inter-dependency reinforcement, etc. as we will later see in Section 4.4.4. A filtered data frame is therefore denoted as a set of input samples (eq. 3.6) and output labels (eq. 3.7) as follows shown in chapter 3.

4.3.2 Attention Mechanism

A-NFVLearn's attention mechanism (Bahdanau *et al.*, 2015) is developed in order to improve the performance on long input sequences. The main idea is to allow the decoder to selectively access encoder information during decoding. This is achieved by building a different "context vector" for every time step of the decoder, calculating it in the function of the previous hidden state and of all the hidden states of the encoder, assigning them trainable weights.

In this way, the attention mechanism assigns different importance to the different features of the input sequence and gives more attention to the more relevant inputs.

4.3.2.1 Encoder

At each time step, the representation of each input sequence $t_x \in \{1, 2, ..., T_x\}$ is computed as a function of the hidden state $h(t_x)$ of the previous time step and of the current input. The final hidden state $h(T_x)$ contains all the encoded information from the previously hidden representations and the previous inputs.

$$h(t_x) = f(W \cdot h(t_x - 1) + U \cdot x^{\langle t_x \rangle}) \tag{4.1}$$

4.3.2.2 Context Vector

A different context vector $c(t_y)$ is computed for every time step $t_y \in \{1, 2, ..., T_y\}$ of the decoder. In order to calculate the context vector $c(t_y)$ for time step t_y we proceed as follows. First of

all, for every combination of time step t_x of the encoder and time step t_y of the decoder, the so-called alignment scores $e(t_x, t_y)$ are computed with the following weighted sum:

$$e(t_x, t_y) = V_a \cdot tanh(U_a \cdot s(t_y - 1) + W_a \cdot h(t_x)) \tag{4.2}$$

In this equation, W_a, U_a and V_a are trainable attention weights. The weights W_a are associated to the hidden states $h(t_x) \in \{1, 2, ..., T_x\}$ of the encoder, the weights U_a are associated to the hidden states $s(t_y) \in \{1, 2, ..., T_y\}$ of a decoder, and the weights V_a define the function that calculates the alignment score.

For every time step t_y, the scores $e(t_x, t_y)$ are normalized using a softmax activation function over the encoder time steps t_x, obtaining the attention weights $\alpha(t_x, t_y)$:

$$\alpha(t_x, t_y) = \frac{exp(e(t_x, t_y))}{\sum_{t_x \in T_x} exp(e(t_x, t_y))} \tag{4.3}$$

The attention weights $\alpha(t_x, t_y)$ capture the importance of the input of time step t_x for decoding the output of time step t_y. The context vector $c(t_y)$ is calculated as the weighted sum of all the hidden values $h(t_x)$ of the encoder according to the attention weights:

$$c(t_y) = \sum_{t_x \in T_x} \alpha(t_x, t_y) \cdot h(t_x) \tag{4.4}$$

Hence, this context vector allows paying more attention to the more relevant inputs in the input sequence.

4.3.2.3 Decoder

Now the context vector $c(t_y)$ is passed to the decoder, which computes the probability distribution of the next possible output. This operation of decoding goes for all the time steps present in the input.

Then the current hidden state $s(t_y)$ is computed according to an LSTM function, taking as input the context vector $c(t_y)$, the hidden state $s(t_y - 1)$ and output $\hat{y}^{\langle t_y - 1 \rangle}$ of the previous time step:

$$s(t_y) = f(s(t_y - 1), \hat{y}^{\langle t_y - 1 \rangle}, c(t_y)) \tag{4.5}$$

Using this mechanism, the model is able to find the correlations between different parts of the input sequence and corresponding parts of the output sequence.

For each time step, the output of the decoder is calculated by applying the softmax function to the weighted hidden state $s(t_y)$:

$$\hat{y}^{\langle t_y \rangle} = tanh(V \cdot s(t_y)) \tag{4.6}$$

4.3.3 AO implementation

AO is an outlier detection mechanism allowing skewness in multivariate data presented in (Hubert & Van Der Veeken, 2008). The method is based on the adjusted boxplot for skewed data (Hubert & Vandervieren, 2008) and essentially defines for univariate data a different scale on each side of the median. This scale is obtained by means of a robust measure of skewness (Brys, Hubert & Struyf, 2004).

4.3.3.1 AO for univariate skewed data

The outlyingness of a univariate data point tells us how far the observation lies from the center of the data, standardized by means of a robust scale. In this definition, it does not matter whether the data point is smaller or larger than the median. However, when the distribution is skewed, authors in (Hubert & Van Der Veeken, 2008) apply a different scale on each side of the median. The AO for univariate data is then defined as:

$$AO_i = AO^{(1)}(x_i, X_n) = \begin{cases} \frac{x_i - med(X_n)}{w_2 - med(X_n)} \text{ if } x_i > med(X_n) \\ \frac{med(X_n) - x_i}{med(X_n) - w_1} \text{ if } x_i < med(X_n) \end{cases} \tag{4.7}$$

With w_1 and w_2 the lower and upper whisker of the adjusted boxplot are applied to the data set X_n.

Note that $AO^{(1)}$ is location and scale invariant, hence it is affected by changing the center and/or the scale of the data. As the AO is based on robust measures of location, scale and skewness, it is resistant to outliers. In theory, resistance up to 25% of outliers can be achieved, although medcouple often has a substantial bias when the contamination is more than 10%.

4.3.3.2 AO for multivariate skewed data

Consider now a p-dimensional sample $X_n = (x^1, ..., x^n)^T$ with $x_i = (x_{i1}, ..., x_{ip})^T$. AO outlier detection for multivariate data is then defined as:

$$AO_i = AO(x_i, X_n) = sup_{\alpha \in \mathbb{R}^p} \, AO^{(1)}(\alpha^T x_i, X_n \alpha) \tag{4.8}$$

Note that in practice, AO cannot be computed by projecting the observations on *all* univariate vectors α. Hence, we should restrict ourselves to a finite set of random directions. Our simulations

have shown that considering $m = 250p$ directions yields a good balance between 'efficiency' and computation time. Random directions are generated as the directions perpendicular to the subspace spanned by p observations, randomly drawn from the data set. As such, AO is invariant to affine transformations of the data. Moreover, in our implementation we always take $||\alpha|| = 1$, although this is not required as $AO^{(1)}$ is invariant.

Once AO is computed for every observation, we can use this information to decide whether an observation is outlying or not. Unless for normal distributions for which the AOs are asymptotically distributed, the AO distribution is in general unknown (but typically right-skewed as they are bounded by zero). Hence, we compute the adjusted boxplot of the AO-values and declare a multivariate observation outlying if its AO_i exceeds the upper whisker of the adjusted boxplot. More precisely, the outlier cutoff value equals:

$$\text{cutoff} = Q_3 + 1.5e^{3MC}IQR \tag{4.9}$$

Where Q_3 is the third quartile of the AO, and similarly for IQR and MC. Here, MC stands for MedCouple which is a robust measure of skewness (Brys *et al.*, 2004). It is defined as:

$$MC(X_n) = med_{x_i<med_n<x_j}\ h(x_i, x_j) \tag{4.10}$$

With med_n the sample median, and:

$$h(x_i, x_j) = \frac{(x_j - med_n) - (med_n - x_i)}{x_j - x_i} \tag{4.11}$$

4.4 Experiment

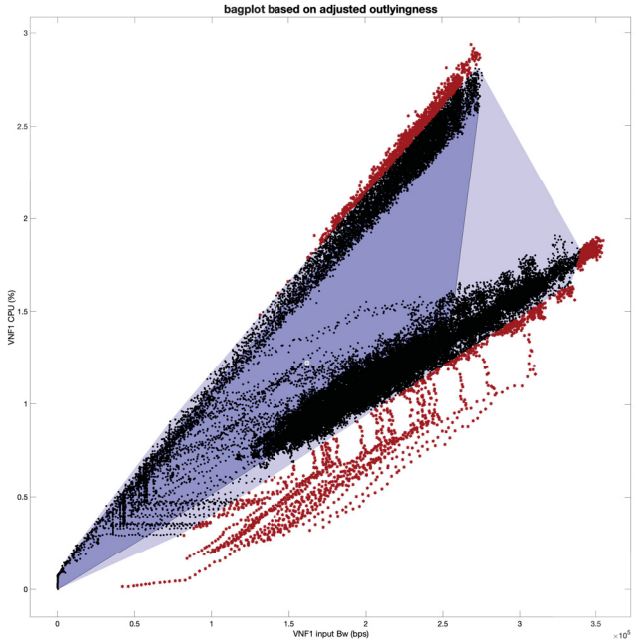
This section presents the main performed experiments and obtained results.

4.4.1 Data

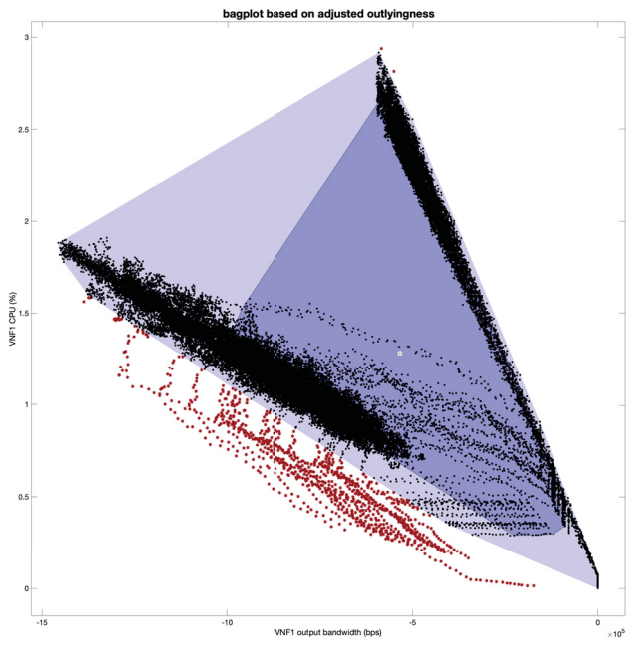
Big instances of VNFs such as an IP multimedia subsystem (IMS) may be composed of smaller instances of software components, called VNFCs. A VNFC is an internal component of a VNF which provides a defined sub-set of that VNF's functionality, with the main characteristic that a single instance of this component maps 1:1 against a single virtualization container. This means that the VNFCs in a VNF are linked to each other by a combination of directed and undirected links, and work together to provide the required functionality of the VNF. This irrevocably poses a challenge for someone aiming to collect resource usage metrics from those VNFCs since some ad hoc measures have to be taken in order to filter the internal network traffic between multiple VNFCs of the same VNF. For the purpose of this experiment, we have chosen to combine the resource usage metrics of all VNFCs into single VNFs and redefined SFCs into directed graphs as shown in Fig. 3.4 and a complete description of the composition of the SFCs used in this work can be found in Section 3.5.1 and in Fig. 3.5.

Table 4.1 Hyperparameters

Hyperparameter	**Value**
# input time steps T_x	[5, 10]
# output time steps T_y	[3, 5, 7]
Normalization	MinMax [0,1]
# hidden layers	1
# units per layer	50
Optimizer	ADAM
Regularizer	L1 L2 regularization
Learning rate	0.001
β_1	0.9
β_2	0.999
ϵ	1e-07
Batch size	512
Loss function	MSE
Validation set size	5% of the dataset
Early stopper	10



a) BWi (x-axis) versus CPU (y-axis)



b) BWo (x-axis) versus CPU (y-axis)

Figure 4.2 Examples of detected AO bivariate outliers (in red) from VNF1

4.4.2 Tools and Model Training

Training of the models was performed on a computer with a 6-core Intel Core i7 10710U CPU with each core clocked at 1.61 GHz, 32 GB of RAM and an eGPU module hosting an Nvidia GeForce RTX 2080 Ti GPU with 12 GB of NVRAM. The learning environment was set up natively on Ubuntu Linux OS 18.04.5 LTS with Nvidia's CUDA 10.1 GPU drivers.

A-NFVLearn was developed using the Python 3.8 programming language on a TensorFlow 2.2 platform with the TensorFlow-GPU library and Keras 2.4.2. Several libraries commonly used in machine learning were used such as NumPy 1.19.0, SciPy 1.4.1, Pandas 1.0.5 and Sci-Kit-Learn 0.23.1. MatPlotLib was used for data visualization.

AO filtering was made in Matlab 2016a using LIBRA[1], developed at Robust@Leuven. The outliers were flagged and stored in a .csv file, then filtered at the data pre-processing stage of model training. Examples of flagged outliers (in red) in a bagplot using LIBRA for bivariate data are depicted in Fig. 4.2.

Table 4.2 IMS SFC inputs per correlation coefficients

Model	**Input features**
VNF1	CPU_1, Mem_1, BWo_1, BWi_1, BWo_5, BWi_2
VNF2	CPU_2, Mem_2, BWo_2, BWi_2, BWi_3, BWi_6
VNF3	CPU_3, Mem_3, BWo_3, BWi_3, Mem_6, Mem_5
VNF4	CPU_4, Mem_4, BWo_4, BWi_4, Mem_3
VNF5	CPU_5, Mem_5, BWo_5, BWi_5, BWo_3, BWi_2
VNF6	CPU_6, Mem_6, BWi_6, Mem_3

Table 4.3 WEB SFC inputs per correlation coefficients

Model	**Input features**
VNF1	CPU_1, Mem_1, BWo_1, BWi_1
VNF2	CPU_2, Mem_2, CPU_3
VNF3	CPU_3, Mem_3, CPU_2

[1] Available at https://wis.kuleuven.be/statdatascience/robust/LIBRA/

All trained models used the same hyperparameters shown in Table 4.1 for comparison purposes. The number of hidden layers, the number of units per layer, the regularization process as well as its sub-parameters (ADAM optimizer's learning rate, β_1, β_2 and ϵ) and the early stopper value were selected through a grid search process that yielded the best overall MSE validation loss. For the normalization process, NFVLearn uses an unorthodox technique: before training our models, we separately fit and transform both input features and output features by using MinMax normalization with a range of $[0, 1]$. The key idea behind this technique is that we use different input/output features of widely different ranges ($[0.00, 3.00]$ usage ratio from 3 vCPUs, $[0, 4.000.000.000]$ Bytes of RAM usage and $[0, 10.000.000]$ bps of I/O bandwidth usage). Hence, our aim is to uniformly scale down the values of any and all output features, to then uniformly derive the weights of the neural nodes through gradient descent during backpropagation. The fit weights of the MinMax normalization for both input and output features are then stored in order to "un-normalize" predicted values later on, when using our trained models for prediction.

Table 4.4 IMS Output Features

Model	**Output features**
VNF1	CPU_1, Mem_1, BWo_1
VNF2	CPU_2, Mem_2, BWo_2
VNF3	CPU_3, Mem_3, BWo_3
VNF4	CPU_4, Mem_4, BWo_4
VNF5	CPU_5, Mem_5, BWo_5
VNF6	CPU_6, Mem_6

Table 4.5 Web Output Features

Model	**Output features**
VNF1	CPU_1, Mem_1, BWo_1
VNF2	CPU_2, Mem_2
VNF3	CPU_3, Mem_3

Moreover, we have selected specific input features (Tables 4.2 and 4.3) based on the highest correlation coefficient scores matching one or several output features (Tables 4.4 and 4.5) for each model. We made those choices to ensure the highest RMSE scores for each predicted

resource usage feature by leveraging LSTM's ability to reinforce predictions based on hidden inter-dependencies between each selected resource attribute.

Finally, we have trained models with different sets $T_x = [5, 10]$ and $T_y = [3, 5, 7]$ in order to evaluate the prediction accuracy of different numbers of output time steps given different numbers of input time steps.

4.4.3 Comparisons

Different LSTM-based combinations are tested in Section 4.4.4. For instance, we designed models with NFVLearn and A-NFVLearn, and combined them either with or without AO filtering, for a total of four different combinations. Then, we trained those models with variations in the numbers of input time steps ($[5, 10]$) and predicted output time steps ($[3, 5, 7]$). In order to compare the prediction accuracy of each setup, the RMSE defined in eq. 3.12 and the coefficient of determination R^2 defined in eq. 3.13 were used. The principle behind this selection of metrics is that RMSE offers a good measure of the differences between values, while R^2 provides a measure of how well observed outcomes are replicated by the model, based on the proportion of total variation of outcomes explained by the model.

Table 4.6 Number of detected outliers

Filtered model	Number of outliers
IMS VNF1	142
IMS VNF2	5515
IMS VNF3	1090
IMS VNF4	2682
IMS VNF5	6127
IMS VNF6	583
WEB VNF1	2316
WEB VNF2	841
WEB VNF3	12618

4.4.4 Results

The results of our experiments and our observations are explained in this section. Tables 4.6-4.10 and Figs. 4.3-4.10 depict our results and analyses.

Table 4.7 IMS Validation RMSE to baseline model ratio

	w/ AO	**w/o AO**
w/ Attention	0.990160181	0.846047724
w/o Attention	1.132801209	1

Table 4.8 IMS Performance to baseline model ratio

	w/ AO	**w/o AO**
w/ Attention	0.711271062	0.716189919
w/o Attention	0.931850079	1

Table 4.9 Web Validation RMSE to baseline model ratio

	w/ AO	**w/o AO**
w/ Attention	1.094133979	0.978894549
w/o Attention	1.092430993	1

Table 4.10 Web Performance to baseline model ratio

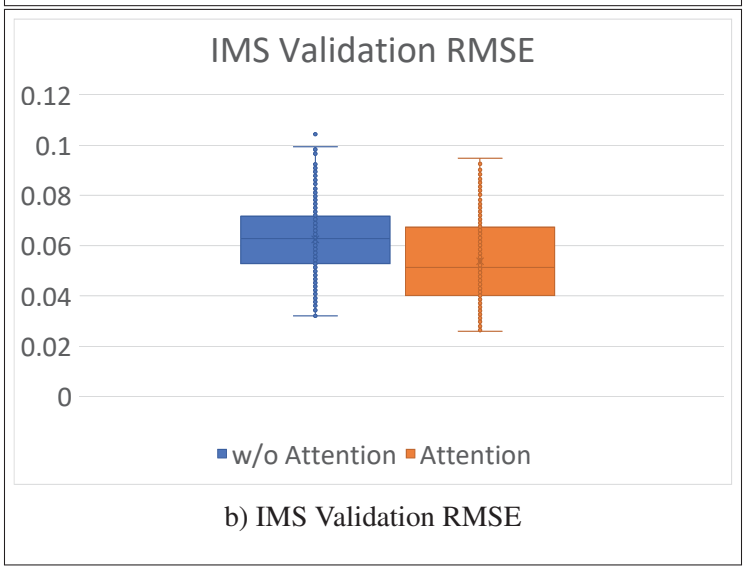
	w/ AO	**w/o AO**
w/ Attention	0.610178012	0.68154401
w/o Attention	0.846191554	1

For instance, Table 4.6 gives the number of flagged AO outliers for each model at pre-processing. Those flagged outliers were then processed out of the data set for each training instance along with their neighbouring data in a time window of length T_x.

We notice that several models (i.e., *IMS VNF1*, *IMS VNF3*, *IMS VNF6*, *WEB VNF1*, and *WEB VNF2*) have a relatively low number of detected outliers that amount to less than 5% of their respective data sets. On the other side of the spectrum, however, other models (i.e., *IMS VNF5*

WEB VNF2) get a high number of filtered outliers that amount to over 10% of their respective data sets.

a) IMS Training RMSE



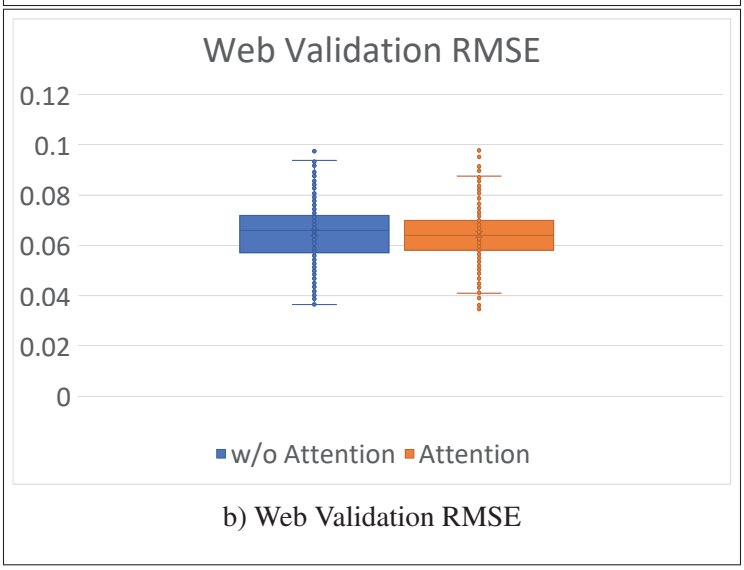
b) IMS Validation RMSE

Figure 4.3 Whisker boxes of IMS training and validation RMSE - Models with and without attention

Tables 4.7 and 4.8 show the RMSE ratio and performance ratio of each combination of attention-based and AO-filtered IMS models, while Tables 4.9 and 4.10 do the same for attention-based and AO-filtered web models, respectively. For each table, the average RMSE and performance are compared to the average "without attention/without AO" baseline models.

In Table 4.7, we observe that RMSE scores of IMS attention-based models outperform those without an attention mechanism. Furthermore, observations in Table 4.8 show that IMS models with AO filtering outperform models without AO filtering during the training of the models. Moreover, we notice that IMS models with an attention-based mechanism clearly have the edge over models without attention performance-wise.

a) Web Training RMSE

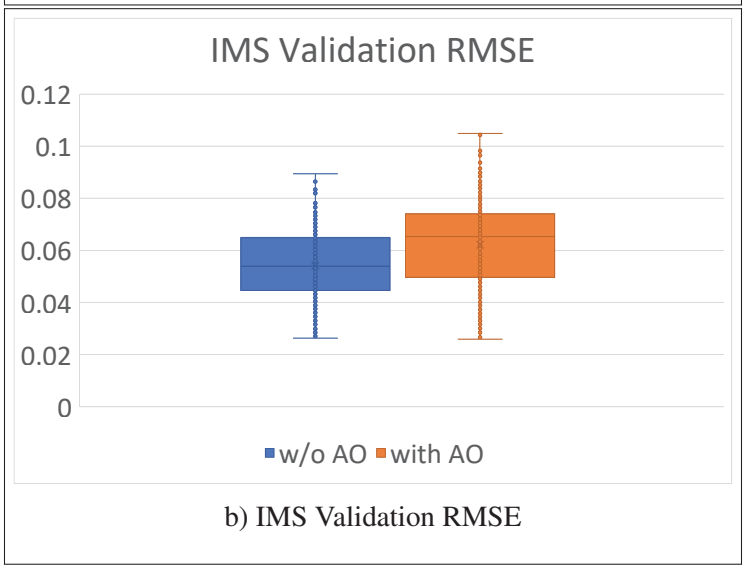


b) Web Validation RMSE

Figure 4.4 Whisker boxes of Web training and validation RMSE - Models with and without attention

These observations demonstrate that IMS models using an attention mechanism and AO filtering not only improve RMSE accuracy of the resource usage predictions but also significantly improve IMS model training performance.

a) IMS Training RMSE



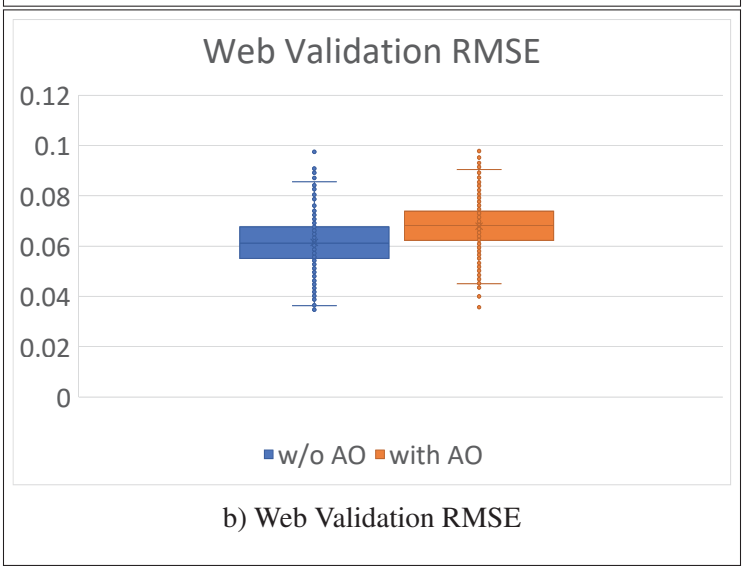
b) IMS Validation RMSE

Figure 4.5 Whisker boxes of IMS training and validation RMSE - Models with and without AO

Observations made from Table 4.9 show that RMSE scores of web models without AO filtering outmatch those with AO filtering. In this case, models with AO filtering yield over 9% higher

RMSE scores than the baseline web model which is still at a tolerable level since the baseline web model's average RMSE score is 0.0621.

a) Web Training RMSE

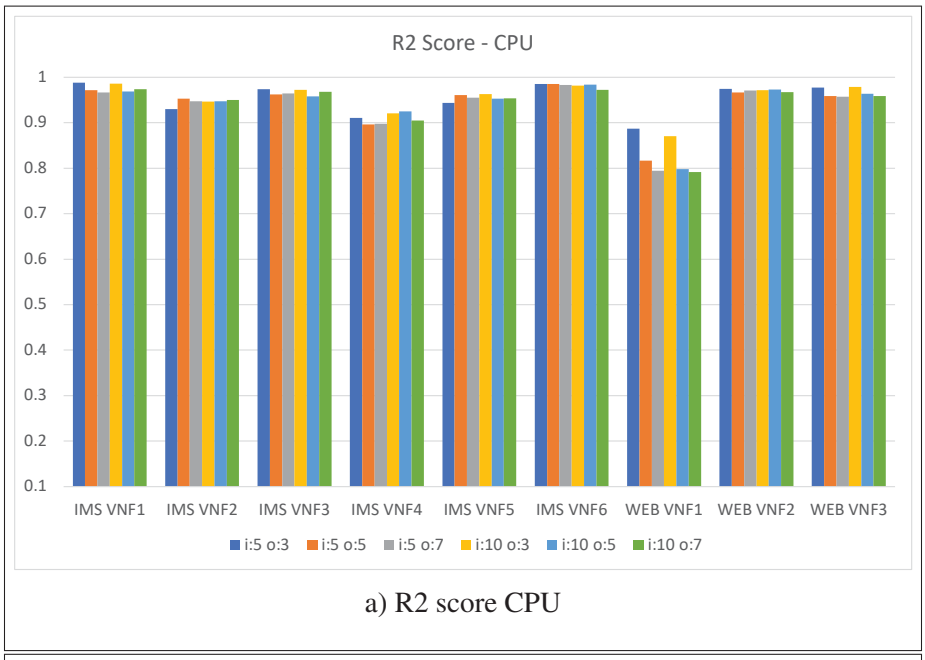


b) Web Validation RMSE

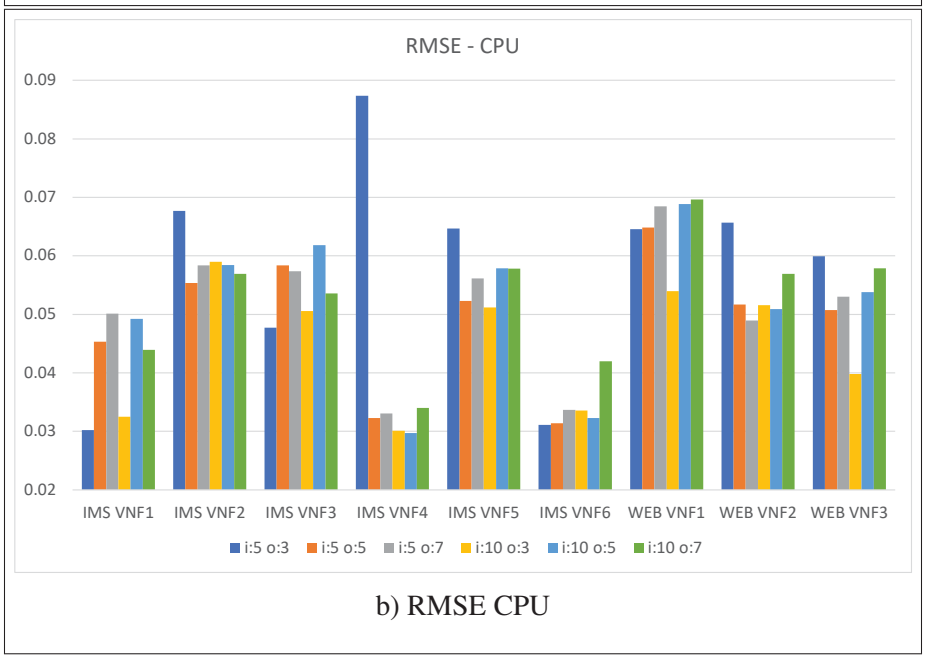
Figure 4.6 Whisker boxes of Web training and validation RMSE - Models with and without AO

Next, we observe in Table 4.10 that attention-based web models clearly outperform web models without an attention mechanism, with an edge for models which also use AO-filtering with a 39% performance gain during training. These observations show that attention-based, AO-filtered web

models benefit from an outstanding training performance gain without significantly impacting the overall RMSE scores of the models.



a) R2 score CPU

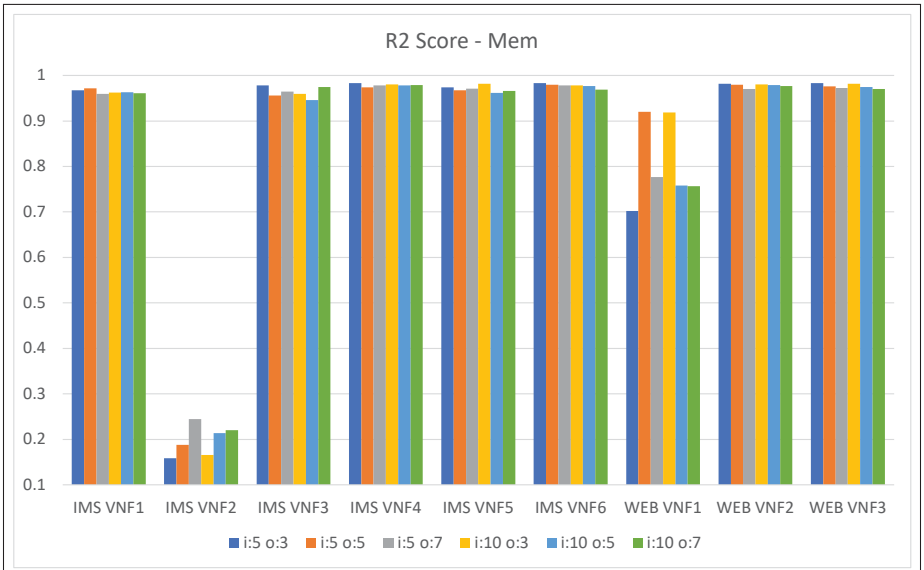


b) RMSE CPU

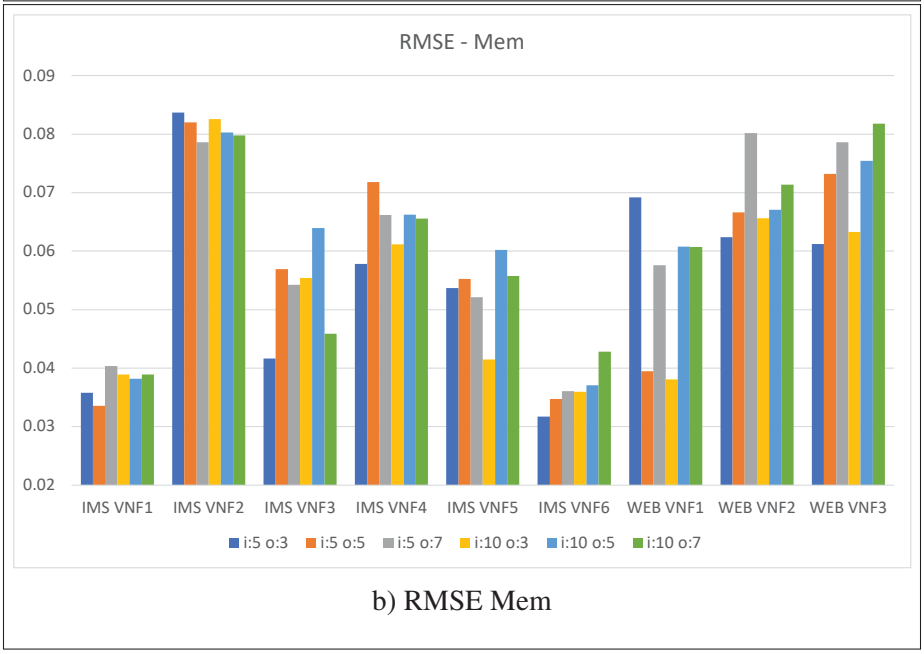
Figure 4.7 R2 score and RMSE per VNF CPU

Figs. 4.3-4.6 depict training and validation results of all IMS and web models assembled in two sets: the first compares models with and without an attention mechanism (Figs. 4.3, 4.4) and the

second compares models with and without AO filtering (Figs. 4.5, 4.6). In Figs. 4.3 and 4.4, all subfigures indicate that models with an attention mechanism achieve better RMSE results than those without, both for IMS and web models, as well as in training and validation.

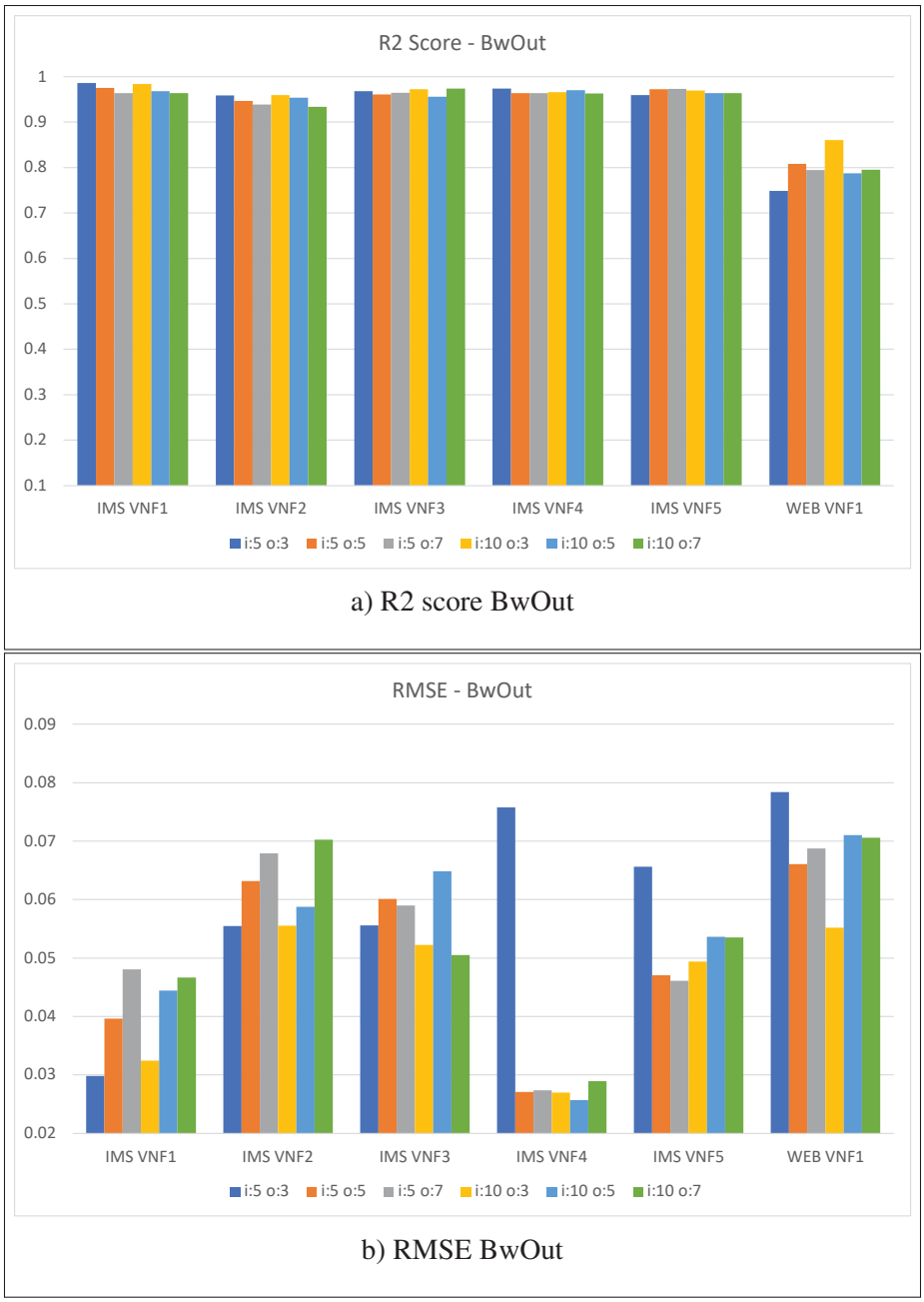


a) R2 score Mem



b) RMSE Mem

Figure 4.8 R2 score and RMSE per VNF memory



a) R2 score BwOut

b) RMSE BwOut

Figure 4.9 R2 score and RMSE per VNF BwOut

Figs. 4.7-4.9 show R^2 scores and RMSE of individual predicted resource usage from each model built using input time step sets of $T_x = [5, 10]$ and output time step set of $T_y = [3, 5, 7]$. For instance, subfigures 4.7a and 4.7b depict histograms of R^2 scores and RMSE from predicted CPU resource usage, respectively. Next, subfigures 4.8a and 4.8b show R^2 scores and RMSE

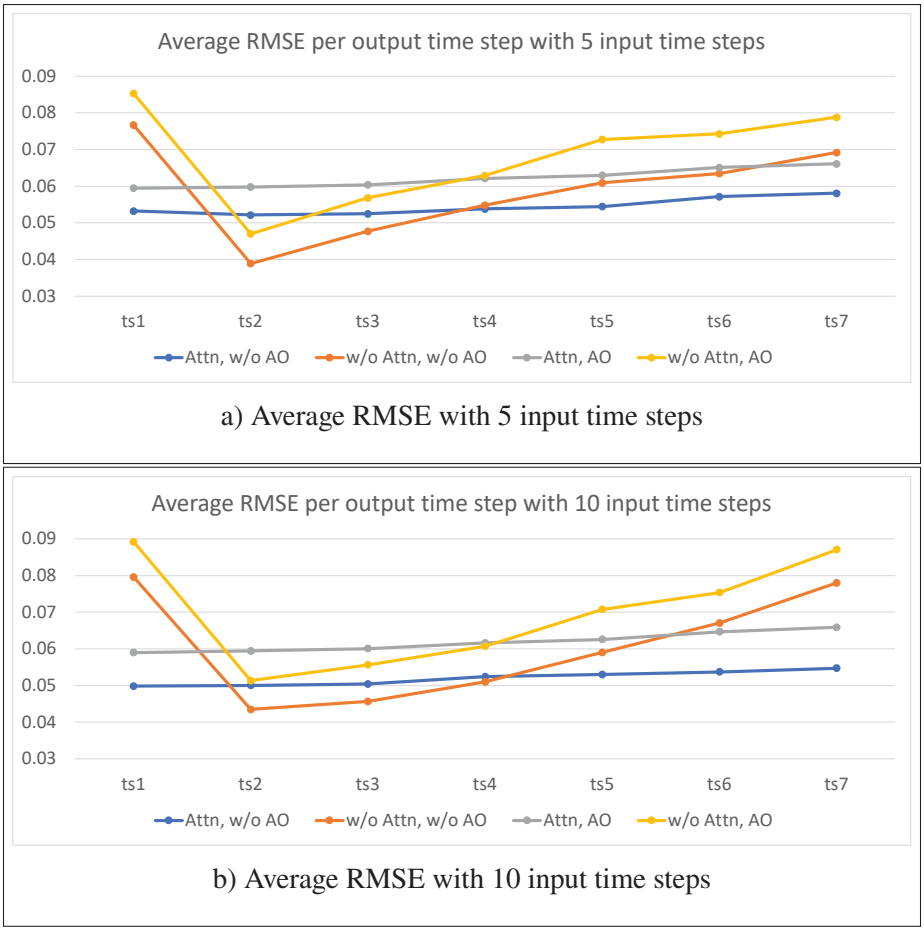
of predicted memory usage. Lastly, subfigures 4.9a and 4.9b show results of predicted output bandwidth resource usage.

General observation of R^2 scores throughout shows that the majority are > 0.90, which indicates a high fidelity of the predictions compared to the observed results at the validation step. Only WEB VNF1 CPU (Fig. 4.7a, memory (Fig. 4.8a and output bandwidth (Fig. 4.9a) which R^2 scores are near 0.8, and IMS VNF2 memory (Fig. 4.8a, near or below 0.2) show R^2 scores that requires further investigation.

In WEB VNF1's case, we understand that this is an instance of a web server receiving HTTP requests from end users and as such, it makes it difficult to forecast resource usage of any kind given prior resource usage history from that same VNF or its neighbours. Furthermore, IMS VNF2's memory usage forecasting is difficult to predict since that VNF is the end point of IMS "INVITE" requests and, as such, its main function is, therefore, more network traffic oriented and less reliant on random access memory than, for example, a VNF hosting a database manager.

Observations of RMSE scores in Figs. 4.7b, 4.8b and 4.9b show that models built using $T_x = 5$ and $T_y = 3$ (dark blue) are proportionally the less accurate models overall, and those built using $T_x = 10$ and $T_y = 3$ being the most accurate. Interestingly, the RMSE accuracy of the models has stronger ties to the number T_x of input time steps than to those of the number T_y of output time steps, showing that LSTM does indeed learn relationships over many time steps and enhances output predictions accordingly.

Finally, Fig. 4.10 yields a lot of useful information. First, comparing sub-figures 4.10a and 4.10b shows interesting trends. First, we notice that models without attention yield better RMSE accuracy when using a lesser number of input time steps (models with 5 input time steps are more accurate than those with 10 input time steps), while models with an attention mechanism show the opposite. Moreover, we notice that the gap in RMSE accuracy becomes larger between models with and without attention when the model uses more input time steps. This observation demonstrates an attention mechanism's ability to decipher complex relationships between input features over a larger number of input time steps than models simply using LSTM.



a) Average RMSE with 5 input time steps

b) Average RMSE with 10 input time steps

Figure 4.10 Average RMSE per time step

Next, when looking at both sub-figures of Fig. 4.10, other trends emerge. First, we notice a significantly poorer RMSE accuracy for the predicted outputs at time step 1 from models without attention, followed by excellent RMSE accuracy results at output time steps $[2-4]$. Further investigation should be made to figure out why output predictions at time step 1 fail to generalize. Results also show that RMSE accuracy degrades more steadily and with a lower slope when using models with an attention mechanism. This is especially beneficial at predicted output time steps $[4-7]$, where models with attention clearly outperform those without attention. Lastly, we observe that models using AO show slightly lower RMSE accuracy than those without AO. However, it is worth noting that models with attention and AO outperform those without AO

(both with and without attention) when predicting resource usage values at output time steps $]5-7]$.

Results in Fig. 4.10 highlight A-NFVLearn's efficiency with both an attention mechanism and AO filtering. The attention mechanism ensures a smooth, steady prediction accuracy over a large horizon window while AO filtering provides outstanding training performance. When used together, models trained with attention and AO outmatch models without attention and that advantage grows with the availability of a large resource usage history.

4.5 Discussion

First of all, a very clear trend emerges from the results in Section 4.4.4: A-NFVLearn has significant gains in prediction accuracy and training performance over NFVLearn for both IMS and Web models. This observation is very interesting from a design perspective since it confirms our initial hypothesis that NFVLearn's multivariate, many-to-many architecture was suitable for an attention mechanism upgrade. This enhancement demonstrates that attention mechanisms are perfectly fit for resource usage forecasting using interdependency-based resource usage history in cloud and NFV environments. A-NFVLearn not only benefits from the attention mechanism's deeper ability to decipher relationships between several input features, but it also improves model generalization and does so with fewer training iterations than NFVLearn's standard LSTM architecture.

The next highlight that is drawn out from the results is that AO filtering systematically improves the training performance of the VNF resource usage forecasting models. Pairing it with an attention mechanism further improves performance results while mitigating a decrease in RMSE accuracy. This technique proves to be a good option to A-NFVLearn's pre-processing stage since an administrator gets the option to tradeoff minor loss in RMSE accuracy with improved prediction under-fitting reduction and training performance. Applied to large datasets and NFV environments hosting large numbers of VNFs, AO filtering will significantly reduce the overall training time of all the VNF models, which will also reduce the power required to train all

those models. As mentioned, the decision to apply AO filtering or not is left entirely under the systems administrator according to model training performance, accuracy and prediction variance objectives.

Another comment is that the AO filtering process is extremely user-friendly. Throughout the course of the experiments, we have kept AO's outlier detection parameters to default settings as provided in the implementation in (Hubert & Van Der Veeken, 2008). The flagged multivariate outlier data points are automatically filtered out along with their neighbours from the inputs and outputs of the training set without user intervention. A list of the flagged outliers is then saved for review.

Now, a more general view of the applicability of an approach such as A-NFVLearn in online NFVIs is that although the results shown thereof are promising, the research team is conscious that LSTM-based architectures are heavily data-centric and that broader analysis must be conducted of this approach in live environments. RMSE accuracy and generalization of the models rely heavily on ad-hoc concepts of SFC digital twins. Such concepts evolve according to VNF and VNFC software architecture which varies from vendor to vendor, software versions, underlying NFVI hosts' architecture, the configuration of the server clusters in the edge and core networks, network links and overall heterogeneity of the infrastructure. In this regard, although we have demonstrated that A-NFVLearn adapts well to different IMS and web SFC scenarios, the research team should extend its experiments to a larger NFVI when possible.

Moreover, further research should be conducted in regard to the granularity of resource usage history, which was set to 20 seconds per time step for those experiments. VNF services, jobs and tasks behave differently seen from different time scales at which the resource usage data was collected. Therefore, a thorough investigation of those behaviours must be conducted in order to align properly with VNF resource adaptation objectives.

Addressing the two aforementioned observations will ensure that deploying A-NFVLearn in commercial carrier-grade solutions becomes realistic in the near future.

4.6 Conclusion

In this chapter, we've presented A-NFVLearn and a novel outlier filtering technique for multivariate data based on AO. A-NFVLearn is a novel, attention-enhanced architecture based on NFVLearn's multivariate, many-to-many LSTM-based architecture designed for resource usage forecasting of multiple resource attributes of a VNF in an SFC. We compared training performance, RMSE prediction accuracy and R^2 prediction fidelity of A-NFVLearn and NFVLearn with and without AO's pre-processing outlier filtering technique. Results show that leveraging A-NFVLearn's attention mechanism significantly improves the VNF model's training time performance and RMSE prediction accuracy and that AO multivariate outlier filtering also improves training time performance and reduces variance in the forecasted resource usage of CPU, memory and network bandwidth resource attributes of VNF from IMS and web SFCs.

CHAPTER 5

DISCUSSION

This thesis is the culmination of a long journey rich in findings and discoveries. The path through this journey has been ripe with unexpected challenges, thought-provoking dilemmas and opportunities that have, in the end, crystallized into innovative solutions and, moreover, into a strong knowledge about how to design complex applied AI solutions for industries operating in Information and Communication Technology (ICT).

The challenges that we have faced have truly been a great learning experience. Chiefly among those was to set the bar for our expectations the moment we received useful data for our DL models. Prior to getting the data, we were on a "model-driven" mindset, thinking about which architectures and designs would be more suited to our tastes, or whether we should choose to build classification or regression models. Getting the data has been a humbling experience, because we then realized that a lot of time would have to be spent on data analysis, pre-processing and modelling.

We understood at that point that our approach would rather have to be "data-driven", that is, to understand what type of features were available, what could be accomplished with the data, how the data could be properly collected in a live environment, how easily that data collection process could be replicated in industrial implementations, how that data collection could impact network traffic overhead in the infrastructure, and, finally, at what frequency that data would be collected. When those considerations were figured out, only then were we able to analyze what type of machine learning technique would be more suited to our needs. Therefore, picking LSTM was not the result of a basic design and implementation decision, but a very long process of trial and error, thorough analysis and comparisons with other approaches. More so, NFVLearn's implementation of LSTM is not a trivial one and was the result of several tweaks for it to fit the intended requirements of multivariate, many-to-many VNF resource usage prediction perfectly.

We also faced dilemmas and difficult decisions that had to be dealt with. For instance, we were on track with designing a solution built on a classification problem (i.e., make VNF scaling decisions based on the resource needs of an SFC). However, after lengthy discussions with industrial partners, it appeared clear that the industry was leaning towards applied machine learning solutions which could provide measurable predictions. In other words, classification problems that rely on F1 scores or similar measures were to be avoided because the notion of false positives and true negatives did not compel the full picture of a scaling decision. Stakeholders from the industry instead sought measurable outcomes such as future physical CPU usage, memory usage, bandwidth usage, power usage and disk I/O so that it would be easier to predict the impact on resource provisioning of physical hosts, which is especially crucial on the edge network, where physical resources are limited in 5G infrastructures and over-provisioning is out of the question.

Finally, we encountered opportunities that extended our work in unexpected manners. In our research, we did not expect to create abstractions of VNFs from raw data, which became a key part of our contributions even though we barely make any reference to this highlight in the previous chapters of this thesis. Since our data was generated from a virtual Clearwater Core IMS infrastructure, tremendous work was done to define VNFCs and VNFs from the generated data of several virtual machines, design a VNF-FG for each request type, and finally define the SFCs. We had to repeat the experiment with a virtual web server as well for our second testbed.

Looking back, we realize that the scope of this work is far beyond what we candidly expected early onto the project. Those challenges, dilemmas and opportunities propelled our work to a complete, complex, yet elegant solution with each objective tightly integrated together. Our satisfaction with this work equates to the interest it has garnered over the years from great minds at our industrial partners.

A-NFVLearn incorporates all the contributions made over the course of our research. Most notably:

1. It leverages deep VNF interdependencies based on resource usage history from different resource attributes of an SFC, thanks to its innovative LSTM architecture assisted by an attention mechanism.
2. It comes packaged with an elaborate data modelling and pre-processing mechanism that:
    a. Creates abstract VNFs linked through a VNF-FG, filters the features from those abstract VNFs, then
    b. Leverages a novel input feature engineering mechanism that only selects highly correlated features in our models based on either GNN, Pearson, Spearman's rank or Kendall's Tau correlation coefficients.
3. A novel multivariate outlier filtering mechanism based on Adjusted Outlyingness that maintains high prediction accuracy while reducing underfitting and significantly improving the training performance time of our models.

Now, more specifically on the topic of our contributions, there are some important observations that we want to share with the reader. As previously stated, this work has been supported and benefited from the input, knowledge and insights from research peers and engineers from Ericsson and Rogers Communications. Over the many opportunities we had to share ideas, we came up with several requirements that NFVLearn had to meet to become a convincing solution for carrier-grade cloud and NFV infrastructures.

First, although comparisons with other state-of-the-art ML approaches cannot be found in these pages, thorough testing and analysis was conducted in the early stages of development to ensure that multivariate inputs were indeed improving forecasting accuracy of NFVLearn models. To do so, we compared three different configurations of NFVLearn with baseline RNN and LSTM models: one using a univariate, many-to-many architecture, another using multivariate, many-to-many architecture with all available resource attributes of a VNF as input features, and a last one using a multivariate, many-to-many architecture with all available resource attributes of an SFC as input features. The last two configurations (the ones using multivariate, many-to-many architectures) systematically and clearly outperformed baseline RNN and LSTM architectures

in RMSE accuracy and R^2 fidelity, confirming our hypothesis that leveraging resource usage interdependencies indeed improved forecasting accuracy.

Next, those experiments were conducted using our filtered datasets from real-world testbeds as described in Appendix I. RMSE, R^2 score, AIC and BIC results displayed in Sections 3.5.5 and 4.4.4 lead us to assert with confidence that NFVLearn can be used in a large number of real-world scenarios, be it in cloud and NFV environments, to forecast workload of a VNF or a SFC from different function types.

Finally, another important software requirement that we have highlighted (e.g., Chapter 2) was to meet very low latency (below 10 ms) required by several functions and services in 5G, B5G and 6G carrier-grade infrastructures (e.g.: V2X, telemedicine). To meet this requirement, we had to ensure that a model execution time was below that mark to avoid SLA, SLO and QoS breaches. The explanation being that forecasting and proposing scaling decisions beyond the 10 ms mark would become irrelevant and arrive too late in environments where workload evolves at such speed. Hence, the average execution time of 17 μs of NFVLearn models meets this requirement. A word of caution, though, is to highlight that those results were obtained on an Intel i7 10700k processor. Results may of course vary in real-time environments, depending on the location where the model is located in the NFVI (i.e., the NFV-MANO, a VNF), the processor type and the number of tasks and jobs already being executed on the host.

CONCLUSION AND RECOMMENDATIONS

In this work, we have proposed three main contributions to the field of interdependency-based resource usage forecasting in cloud environments. We have first elaborated a comprehensive taxonomy of interdependencies in cloud, NFV and microservice environments. The knowledge gained throughout this enterprise then inspired us to design NFVLearn for our second contribution. NFVLearn is a novel VNF resource usage forecasting mechanism based on cutting-edge DL techniques that leverages LSTM, an enhanced RNN architecture, which can find hidden relationships between input features over many time steps. The main feature of this approach is that a model can take inputs of several resource usage time steps from several resource attributes of any number of VNFs in an SFC. That model can then forecast several resource usage time steps from many other resource attributes of one VNF. Alongside NFVLearn, we proposed input feature engineering techniques based on GNNs and Pearson, Spearman's rank and Kendall's rank correlation coefficients. Those techniques aim to improve RMSE prediction accuracy and R^2 prediction fidelity of NFVLearn's models while also reducing their number of input features. Those input feature engineering techniques do so by leveraging inter-dependencies between highly-correlated resource attributes of an SFC. Later in our research project, NFVLearn evolved into A-NFVLearn, which features a significant improvement over its predecessor: an attention-based architecture. A-NFVLearn models yield significantly better RMSE prediction accuracy over a longer number of predicted time steps, all with a lower training time. Finally, our third and last main contribution is an outlier filtering mechanism leveraging adjusted outlyingness (AO). AO filtering reduces the number of training samples required to train an A-NFVLearn model by removing outliers in the training set, thus reducing variance in the remaining training samples. VNF resource usage forecasting made using AO filtering reduces underfitting, and keeps high RMSE accuracy and R^2 fidelity while also reducing the training time of A-NFVLearn models.

As recommended future work, we first wish to investigate multivariate interdependencies further and use clustering techniques to label correlations in resource usage of multiple resource attributes from SFCs and VNFs. We expect that classifying those correlated multivariate behaviours will considerably enhance ML- and DL-based prediction mechanisms by correctly leading to a proper prediction model given the resource usage pattern history of an SFC, a VNF, a service, a function or a task in the infrastructure. Next, we will also work towards enhancing NFVLearn with a real-time training mechanism that could further improve the prediction accuracy of the models while running online. We predict that re-training those models with small batches of training samples could easily be achieved with minimal processing power directly on the host server of an NFV-MANO.

To close this final chapter in out Ph.D. journey, there are a few recommendations we want to share with the reader, especially those willing to design applied DL-based forecasting solutions in the area of cloud computing. Before taking your first step into this adventure, you must ensure that you have a positive answer to all of the following questions:

- Do you have access to or have the skills to design a platform, a testbed or a live infrastructure that generates data that is useful for your models? Getting access to such a platform, getting the required skills and setting up your own platform is definitely not a trivial task.
- Do you have advanced knowledge in AI and the applied field of research? Lacking in any of the two will require a willingness to take the necessary measures to fill the gap. AI and applied research are easy to learn, but hard to master. And you need to be master of both if you want to design practical industrial solutions.
- What will be the granularity of the input and output time steps of your models? The workload behaviour of a system radically changes with time length. Trends, patterns and seasonality of the time series will invariably vary as a result.
- What goal are you trying to achieve with workload forecasting? The forecasting granularity and number of time steps will influence the downstream decision-making process at the end

of the pipeline of your mechanism. E.g., migrating a virtual machine takes up to 20 seconds to proceed before it can handle requests. Forecasting workload in a horizon of 5 seconds with a time length of 1 second per time step will not provide a reliable insight whether it is a good decision or not.

In conclusion, we would like to thank Claes Edstrom at Ericsson Canada and Jean-Yves Bernard at Rogers Communications for their precious input which significantly enhanced the quality of this work. Through them, we have benefited from a unique opportunity to share our findings and ideas with top researchers and engineers from the telecom industry. That experience made us grow as researchers and taught us how to design practical solutions and conduct applied industrial research with global industry leaders in a competitive environment.

APPENDIX I

DATASETS

Included herein is a description of the datasets from our IMS and web testbeds. Both testbeds were designed from freely available software and configured on a barebone Kubernetes platform provided by our partners at Ericsson Canada.

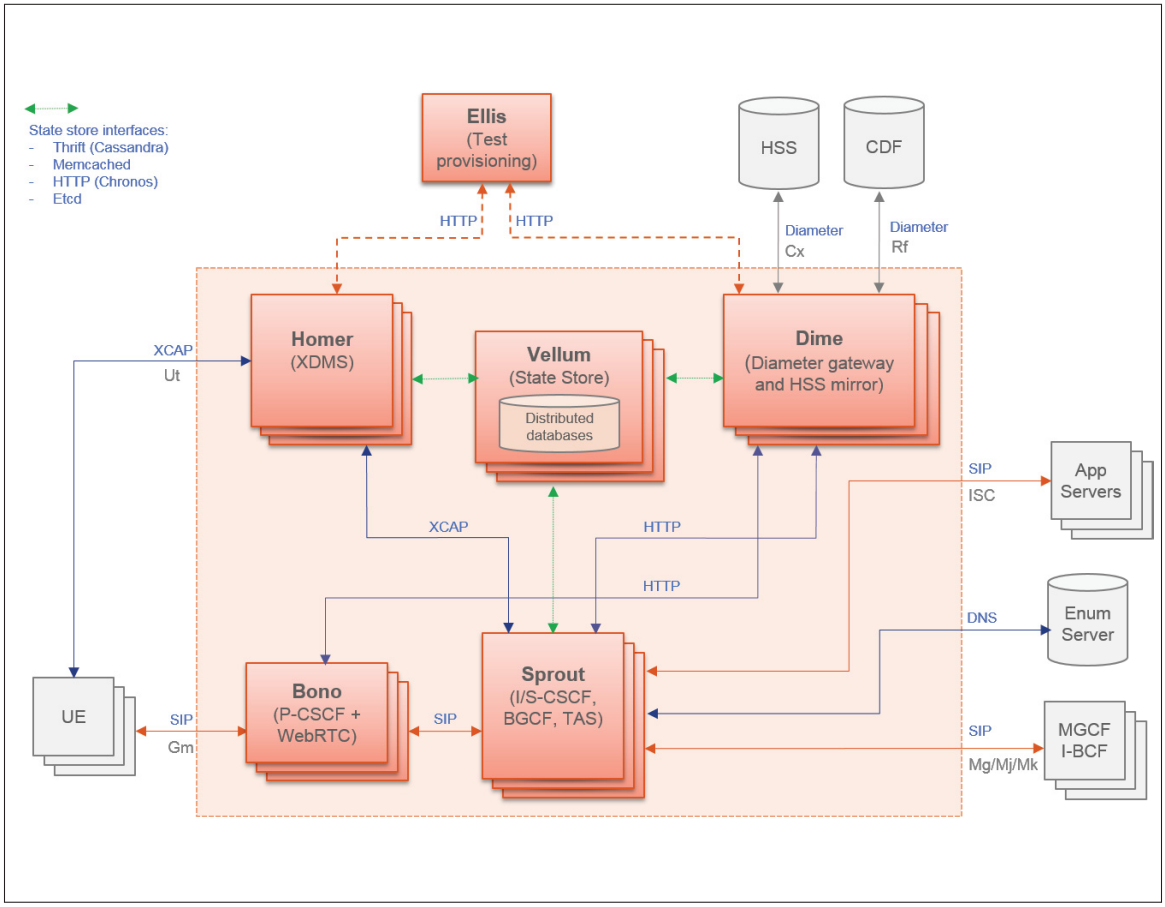


Figure-A I-1 Clearwater IMS Core Testbed
Taken from: https://clearwater.readthedocs.io/en/stable/Clearwater_Architecture.html
(accessed Jan. 31 2023)

Data collection from both testbeds was synchronized to ensure that resource usage metrics from all IMS and web VNFCs were properly time-stamped, and three different simulations were run over a course of two weeks (14 days) each, with resource usage metrics collected every 20

seconds. That gave roughly over 180,000 time-stamped samples of four resource usage metrics (CPU, memory, input bandwidth, output bandwidth) from 14 different pods (containers). Under all accounts, that gave us a total of over 10,000,000 raw, unfiltered data points.

The next step was then to combine VNFCs into their respective VNF in a SFC. More details about the IMS SFCs and web SFC are found in sections 1 and 2 below.

After proper pre-processing of the raw data, we obtained less than 80,000 time-stamped samples of four resource usage metrics, from 9 different VNFs, for a total of about 3,000,000 filtered data points. That number of samples was rich and diversified enough to ensure that we could confidently build NFVLearn models that could faithfully forecast resource usage from VNFs of our IMS and web testbeds.

1. IMS Datasets

The IMS testbed was designed from Metaswitch's Clearwater Core IMS platform. The complete VNF-FG composition (VNFCs, links) is shown in Figure I-1.

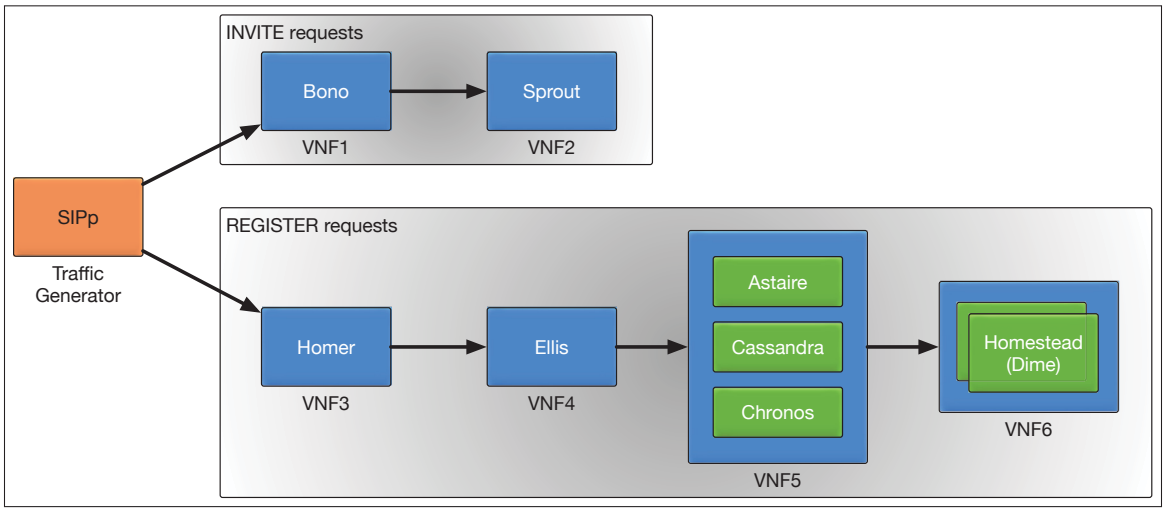


Figure-A I-2 IMS SFC Architecture

From the Clearwater platform, resource usage metrics (CPU, memory, I/O bandwidth) were collected from several VNFCs: Bono, Sprout Homer, Ellis, Vellum (Astaire, Cassandra and

Chronos) and two instances of Dime. We have combined VNFC resource usage metrics into their respective function (into VNFs), then separated the testbed VNF-FG into two separate SFCs: one for IMS INVITE requests and another for IMS REGISTER requests. The resulting architecture used for our experiment is shown in Figure I-2.

2. Web Datasets

The web testbed was designed from a custom Go web application which calls an API and builds words into sentences. The VNF-FG is composed of a Go web application, a load balancer that directs requests to three VNF instances of a REST API, which then serve words read from a database, and a Postgres database which stores words. The web testbed is depicted in Figure I-3.

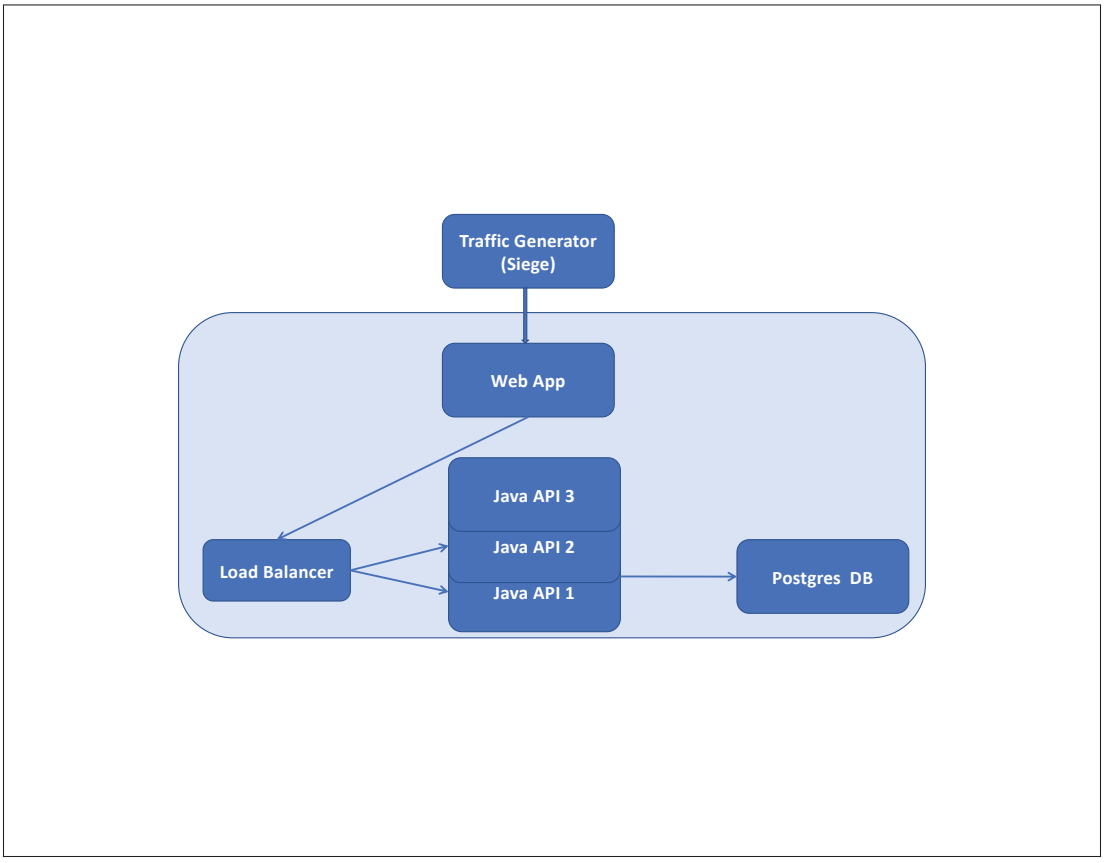


Figure-A I-3 Web Testbed

Resource usage metrics (CPU, memory, I/O bandwidth) was collected from each VNFC, then combined into their respective function as shown in Figure I-4.

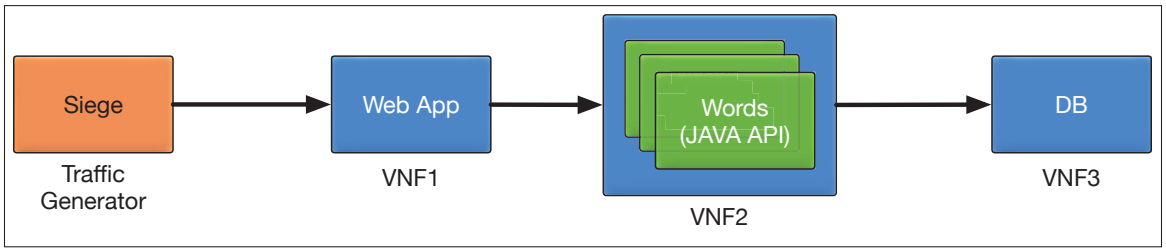


Figure-A I-4 Web SFC Architecture

APPENDIX II

NFVLEARN INPUT AND OUTPUT FEATURES

Tables II-1 and II-2 are supplied here for a reference regarding input features selected by the automated Pearson, Spearman and Kendall pruning setup options. The same applies to Tables II-3 and II-4, which reference the selected outputs for every model.

Table-A II-1 IMS SFC inputs per correlation coefficients

Model	Pearson	Spearman	Kendall
VNF1	CPU_1, Mem_1, BWo_1, BWi_1, BWo_5, BWi_2	CPU_1, Mem_1, BWo_1, BWi_1, BWi_2, BWi_5	CPU_1, Mem_1, BWo_1, BWi_1, Mem_4, BWo_5
VNF2	CPU_2, Mem_2, BWo_2, BWi_2, BWi_3, BWi_6	CPU_2, Mem_2, BWo_2, BWi_2, Mem_3, BWi_1	CPU_2, Mem_2, BWo_2, BWi_2, Mem_3, BWi_6
VNF3	CPU_3, Mem_3, BWo_3, BWi_3, Mem_6, Mem_5	CPU_3, Mem_3, BWo_3, BWi_3, Mem_2, Mem_5	CPU_3, Mem_3, BWo_3, BWi_3, CPU_5, BWo_5
VNF4	CPU_4, Mem_4, BWo_4, BWi_4, Mem_3	CPU_4, Mem_4, BWo_4, BWi_4, Mem_3, BWi_2	CPU_4, Mem_4, BWo_4, BWi_4, Mem_1, BWi_2
VNF5	CPU_5, Mem_5, BWo_5, BWi_5, BWo_3, BWi_2	CPU_5, Mem_5, BWo_5, BWi_5, BWo_3, BWo_1	CPU_5, Mem_5, BWo_5, BWi_5, BWi_6, BWo_2
VNF6	CPU_6, Mem_6, BWi_6, Mem_3	CPU_6, Mem_6, BWi_6, CPU_2	CPU_6, Mem_6, BWi_6, BWo_2

Table-A II-2 WEB SFC inputs per correlation coefficients

Model	Pearson	Spearman	Kendall
VNF1	CPU_1, Mem_1, BWo_1, BWi_1	CPU_1, Mem_1, BWo_1, BWi_1	CPU_1, Mem_1, BWo_1, BWi_1
VNF2	CPU_2, Mem_2, CPU_3	CPU_2, Mem_2, CPU_3	CPU_2, Mem_2, CPU_3, Mem_3
VNF3	CPU_3, Mem_3, CPU_2	CPU_3, Mem_3, CPU_1	CPU_3, Mem_3, CPU_2, Mem_2

Table-A II-3 IMS Output Features

Model	Output features
VNF1	CPU_1, Mem_1, BWo_1
VNF2	CPU_2, Mem_2, BWo_2
VNF3	CPU_3, Mem_3, BWo_3
VNF4	CPU_4, Mem_4, BWo_4
VNF5	CPU_5, Mem_5, BWo_5
VNF6	CPU_6, Mem_6

Table-A II-4 Web Output Features

Model	**Output features**
VNF1	CPU_1, Mem_1, BWo_1
VNF2	CPU_2, Mem_2
VNF3	CPU_3, Mem_3

BIBLIOGRAPHY

Antwi, E., Gyamfi, E. N., Kyei, K., Gill, R. & Adam, A. M. (2021). Determinants of Commodity Futures Prices: Decomposition Approach. *Mathematical Problems in Engineering*, 2021. doi: 10.1155/2021/6032325.

Bahdanau, D., Cho, K. & Bengio, Y. (2015). Neural Machine Translation by Jointly Learning to Align and Translate. *ICLR*, 1–15.

Beck, M. T. & Botero, J. F. (2017). Scalable and coordinated allocation of service function chains. *Computer Communications*, 102, 78-88. doi: 10.1016/j.comcom.2016.09.010.

Betken, A., Dehling, H., Nüßgen, I. & Schnurr, A. (2021). Ordinal pattern dependence as a multivariate dependence measure. *Journal of Multivariate Analysis*, 186, 104798. doi: 10.1016/j.jmva.2021.104798.

Blanco, B., Fajardo, J. O., Giannoulakis, I., Kafetzakis, E., Peng, S., Pérez-Romero, J., Trajkovska, I., Khodashenas, P. S., Goratti, L., Paolino, M., Sfakianakis, E., Liberal, F. & Xilouris, G. (2017). Technology pillars in the architecture of future 5G mobile networks: NFV, MEC and SDN. *Computer Standards and Interfaces*, 54, 216-228. doi: 10.1016/j.csi.2016.12.007.

Blenk, A., Kalmbach, P., Smagt, P. V. D. & Kellerer, W. (2017). Boost online virtual network embedding: Using neural networks for admission control. *2016 12th International Conference on Network and Service Management, CNSM 2016 and Workshops, 3rd International Workshop on Management of SDN and NFV, ManSDN/NFV 2016, and International Workshop on Green ICT and Smart Networking, GISN 2016*, 10-18. doi: 10.1109/CNSM.2016.7818395.

Boutaba, R., Shahriar, N., Salahuddin, M. A. & Limam, N. (2021). Managing Virtualized Networks and Services with Machine Learning. *Communication Networks and Service Management in the Era of Artificial Intelligence and Machine Learning*, 33–68. doi: 10.1002/9781119675525.ch3.

Brys, G., Hubert, M. & Struyf, A. (2004). A robust measure of skewness. *Journal of Computational and Graphical Statistics*, 13(4), 996–1017. doi: 10.1198/106186004X12632.

Cao, L., Fahmy, S., Sharma, P. & Zhe, S. (2018). Data-driven resource flexing for network functions visualization. pp. 111-124. doi: 10.1145/3230718.3230725.

Chakraborty, P., Corici, M. & Magedanz, T. (2020). A comparative study for Time Series Forecasting within software 5G networks. *2020 14th International Conference on Signal Processing and Communication Systems, ICSPCS 2020 - Proceedings*. doi: 10.1109/ICSPCS50536.2020.9310033.

Cho, Y., Jang, S. & Pack, S. (2020). On Performance VNF Load Prediction Models in Service Function Chaining. *International Conference on ICT Convergence*, 2020-Octob, 344–346. doi: 10.1109/ICTC49870.2020.9289275.

Duggan, M., Shaw, R., Duggan, J., Howley, E. & Barrett, E. (2019). A multitime-steps-ahead prediction approach for scheduling live migration in cloud data centers. *Software - Practice and Experience*, 49(4), 617–639. doi: 10.1002/spe.2635.

e Martins, L. M. C., Filho, F. L. D. C., Júnior, R. T. D. S., Giozza, W. F. & da Costa, J. P. C. (2017). Increasing the Dependability of IoT Middleware with Cloud Computing and Microservices. *Companion Proceedings of the 10th International Conference on Utility and Cloud Computing - UCC '17 Companion*, 203-208. doi: 10.1145/3147234.3148092.

Eramo, V., Lavacca, F. G., Catena, T. & Salazar, P. J. P. (2020). Proposal and investigation of an artificial intelligence (Ai)-based cloud resource allocation algorithm in network function virtualization architectures. *Future Internet*, 12(11), 1–13. doi: 10.3390/fi12110196.

Eramo, V., Lavacca, F. G., Catena, T. & Salazar, P. J. P. (2021). Application of a Long Short Term Memory neural predictor with asymmetric loss function for the resource allocation in NFV network architectures. *Computer Networks*, 193(September 2020), 108104. doi: 10.1016/j.comnet.2021.108104.

ETSI. (2013). GS NFV 001 - V1.1.1 - Network Function Virtualization (NFV); Technical Requirements. 1, 1-50. Retrieved from: http://www.etsi.org/standards-search.

ETSI. (2016). GS MEC 002 - V1.1.1 - Mobile Edge Computing (MEC); Technical Requirements. 1, 1-7. Retrieved from: http://www.etsi.org/standards-search.

ETSI. (2019). GS ENI 001 - V2.1.1 - Experiential Networked Intelligence (ENI); ENI use cases. Retrieved from: https://portal.etsi.org/TB/ETSIDeliverableStatus.aspx.

Fahmy, S. & Saxena, V. (2017). Contain-ed : An NFV Micro-Service System for Containing e2e Latency. *SIGCOMM '17 workshop*, 47, 12-17. doi: 10.1145/3094405.3094408.

Francesco, P. D., Malavolta, I. & Lago, P. (2017). Research on Architecting Microservices: Trends, Focus, and Potential for Industrial Adoption. *Proceedings - 2017 IEEE International Conference on Software Architecture, ICSA 2017*, 21-30. doi: 10.1109/ICSA.2017.24.

Gao, J., Wang, H. & Shen, H. (2020). Machine Learning Based Workload Prediction in Cloud Computing. *2020 29th International Conference on Computer Communications and Networks (ICCCN)*, pp. 1-9. doi: 10.1109/ICCCN49398.2020.9209730.

Ghofrani, J. & Lübke, D. (2018). Challenges of Microservices Architecture : A Survey on the State of the Practice. *ZEUS 2018*, 10th ZEUS.

Gupta, A., Habib, M. F., Mandal, U., Chowdhury, P., Tornatore, M. & Mukherjee, B. (2018). On service-chaining strategies using Virtual Network Functions in operator networks. *Computer Networks*, 133, 1-16. doi: 10.1016/j.comnet.2018.01.028.

Gupta, L., Samaka, M., Jain, R., Erbad, A., Bhamare, D. & Metz, C. (2017). COLAP: A predictive framework for service function chain placement in a multi-cloud environment. *2017 IEEE 7th Annual Computing and Communication Workshop and Conference, CCWC 2017*. doi: 10.1109/CCWC.2017.7868377.

Herrera, J. G. & Botero, J. F. (2016). Resource Allocation in NFV: A Comprehensive Survey. *IEEE Transactions on Network and Service Management*, 13, 518-532. doi: 10.1109/TNSM.2016.2598420.

Hirayama, T., Jibiki, M. & Kafle, V. P. (2020). Regressor relearning architecture adapting to traffic trend changes in NFV platforms. *Proceedings of the 2020 IEEE Conference on Network Softwarization: Bridging the Gap Between AI and Network Softwarization, NetSoft 2020*, 272–276. doi: 10.1109/NetSoft48620.2020.9165449.

Hirayama, T., Miyazawa, T., Jibiki, M. & Kafle, V. P. (2021). Sparse regression model-based relearning architecture for shortening learning time in traffic prediction. *IEICE Transactions on Information and Systems*, E104.D(5), 606–616. doi: 10.1587/transinf.2020NTP0010.

Hochreiter, S. & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8), 1735–1780. doi: 10.1162/neco.1997.9.8.1735.

Hubert, M. & Vandervieren, E. (2008). An adjusted boxplot for skewed distributions. *Computational Statistics and Data Analysis*, 52(12), 5186–5201. doi: 10.1016/j.csda.2007.11.008.

Hubert, M. & Van Der Veeken, S. (2008). Outlier detection for skewed data. *Journal of Chemometrics*, 22(3-4), 235–246. doi: 10.1002/cem.1123.

IETF. (2020). Autonomic Networking Integrated Model and Approach (ANIMA). Retrieved from: https://datatracker.ietf.org/wg/anima/charter/.

Jacobs, A. S., Pfitscher, R. J., Santos, R. L. D., Franco, M. F., Scheid, E. J. & Granville, L. Z. (2018). Artificial neural network model to predict affinity for virtual network functions. *IEEE/IFIP Network Operations and Management Symposium: Cognitive Management in a Cyber World, NOMS 2018*, 1-9. doi: 10.1109/NOMS.2018.8406253.

Khalid, R., Javaid, N., Al-zahrani, F. A., Aurangzeb, K., Qazi, E. U. H. & Ashfaq, T. (2020). Electricity load and price forecasting using jaya-long short term memory (JLSTM) in smart grids. *Entropy*, 22(1), 10. doi: 10.3390/e22010010.

Kim, H. G., Jeong, S. Y., Lee, D. Y., Choi, H., Yoo, J. H. & Hong, J. W. K. (2019). A Deep Learning Approach to VNF Resource Prediction using Correlation between VNFs. *Proceedings of the 2019 IEEE Conference on Network Softwarization: Unleashing the Power of Network Softwarization, NetSoft 2019*, 444–449. doi: 10.1109/NETSOFT.2019.8806620.

Kumar, J., Saxena, D., Singh, A. K. & Mohan, A. (2020). BiPhase adaptive learning-based neural network model for cloud datacenter workload forecasting. *Soft Computing*, 24, 14593-14610. doi: 10.1007/s00500-020-04808-9.

Kumar, J., Singh, A. K. & Buyya, R. (2021a). Self directed learning based workload forecasting model for cloud resource management. *Information Sciences*, 543, 345-366. doi: 10.1016/j.ins.2020.07.012.

Kumar, K., Rao, K. G., Bulla, S. & Venkateswarulu, D. (2021b). Forecasting of Cloud Computing Services Workload using Machine Learning.

Lange, S., Tu, N. V., Jeong, S. Y., Lee, D. Y., Kim, H. G., Hong, J., Yoo, J. H. & Hong, J. W. K. (2021). A Network Intelligence Architecture for Efficient VNF Lifecycle Management. *IEEE Transactions on Network and Service Management*, 18(2), 1476–1490. doi: 10.1109/TNSM.2020.3015244.

Li, Y., Zemel, R., Brockschmidt, M. & Tarlow, D. (2016). Gated graph sequence neural networks. *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, (1), 1–20. Retrieved from: http://arxiv.org/abs/1511.05493.

Lu, Y., Panneerselvam, J., Liu, L. & Wu, Y. (2016). RVLBPNN: A workload forecasting model for smart cloud computing. *Scientific Programming*, 2016. doi: 10.1155/2016/5635673.

Marotta, A., D'Andreagiovanni, F., Kassler, A. & Zola, E. (2017a). On the energy cost of robustness for green virtual network function placement in 5G virtualized infrastructures. *Computer Networks*, 125, 64-75. doi: 10.1016/j.comnet.2017.04.045.

Marotta, A., Zola, E., D'Andreagiovanni, F. & Kassler, A. (2017b). A fast robust optimization-based heuristic for the deployment of green virtual network functions. *Journal of Network and Computer Applications*, 95, 42-53. doi: 10.1016/j.jnca.2017.07.014.

Mijumbi, R., Gorricho, J. L., Serrat, J., Claeys, M., De Turck, F. & Latré, S. (2014). Design and evaluation of learning algorithms for dynamic resource management in virtual networks. *IEEE/IFIP NOMS 2014 - IEEE/IFIP Network Operations and Management Symposium: Management in a Software Defined World*, 1–9. doi: 10.1109/NOMS.2014.6838258.

Mijumbi, R., Hasija, S., Davy, S., Davy, A., Jennings, B. & Boutaba, R. (2017). Topology-Aware Prediction of Virtual Network Function Resource Requirements. *IEEE Transactions on Network and Service Management*, 14(1), 106–120. doi: 10.1109/TNSM.2017.2666781.

Moradi, M., Ahmadi, M. & Nikbazm, R. (2022). Comparison of Machine Learning Techniques for VNF Resource Requirements Prediction in NFV. *Journal of Network and Systems Management*, 30(1). doi: 10.1007/s10922-021-09629-1.

Mulerikkal, J., Thandassery, S., Rejathalal, V. & Kunnamkody, D. M. D. (2022). Performance improvement for metro passenger flow forecast using spatio-temporal deep neural network. *Neural Computing and Applications*, 34(2), 983–994. doi: 10.1007/s00521-021-06522-5.

Patel, Y. S., Verma, D. & Misra, R. (2019). Deep Learning Based Resource Allocation for Auto-Scaling VNFs. *International Symposium on Advanced Networks and Telecommunication Systems, ANTS*, 2019-Decem, 1–6. doi: 10.1109/ANTS47819.2019.9118065.

Prats, D. B., Berral, J. L. & Carrera, D. (2018). Unsupervised Learning Pipelines. 15, 142-155.

Ray, P. P. (2021). A perspective on 6G: Requirement, technology, enablers, challenges and future road map. *Journal of Systems Architecture*, 118. doi: 10.1016/j.sysarc.2021.102180.

Saxena, D. & Singh, A. K. (2021). A proactive autoscaling and energy-efficient VM allocation framework using online multi-resource neural network for cloud data center. *Neurocomputing*, 426, 248-264. doi: 10.1016/j.neucom.2020.08.076.

Saxena, D. & Singh, A. K. (2022). Auto-adaptive learning-based workload forecasting in dynamic cloud environment. *International Journal of Computers and Applications*, 44, 541-551. doi: 10.1080/1206212X.2020.1830245.

Shah, S. Y., Yuan, Z., Lu, S. & Zerfos, P. (2017). Dependency analysis of cloud applications for performance monitoring using recurrent neural networks. *Proceedings - 2017 IEEE International Conference on Big Data, Big Data 2017*, 1534–1543. doi: 10.1109/BigData.2017.8258087.

Sun, G., Li, Y., Li, Y., Liao, D. & Chang, V. (2018). Low-latency orchestration for workflow-oriented service function chain in edge computing. *Future Generation Computer Systems*, 85, 116-128. doi: 10.1016/j.future.2018.03.018.

Truong, H. L., Narendra, N. C. & Lin, K. J. (2018). Notes on ensembles of IoT, network functions and clouds for service-oriented computing and applications. *Service Oriented Computing and Applications*, 12, 1-10. doi: 10.1007/s11761-018-0228-2.

Vos, K., Peng, Z., Jenkins, C., Shahriar, M. R., Borghesani, P. & Wang, W. (2022). Vibration-based anomaly detection using LSTM/SVM approaches. *Mechanical Systems and Signal Processing*, 169(December 2021), 108752. doi: 10.1016/j.ymssp.2021.108752.

Wang, S., Bi, J., Wu, J., Vasilakos, A. V. & Fan, Q. (2019). VNE-TD: A virtual network embedding algorithm based on temporal-difference learning. *Computer Networks*, 161, 251–263. doi: 10.1016/j.comnet.2019.05.004.

Xu, K., Wu, L., Wang, Z., Feng, Y., Witbrock, M. & Sheinin, V. (2018). Graph2Seq: Graph to Sequence Learning with Attention-based Neural Networks. 1-14. Retrieved from: http://arxiv.org/abs/1804.00823.

Yu, L., Wu, C. & Xiong, N. N. (2022). An Intelligent Data Analysis System Combining ARIMA and LSTM for Persistent Organic Pollutants Concentration Prediction. *Electronics*, 11(4), 652. doi: 10.3390/electronics11040652.

Zhang, C., Patras, P. & Haddadi, H. (2019a). Deep Learning in Mobile and Wireless Networking: A Survey. *IEEE Communications Surveys and Tutorials*, 21(3), 2224–2287. doi: 10.1109/COMST.2019.2904897.

Zhang, C., Hu, D. & Yang, T. (2022). Anomaly Detection and Diagnosis for Wind Turbines Using Long Short-term Memory-based Stacked Denoising Autoencoders and XGBoost. *Reliability Engineering System Safety*, 222(March), 108445. doi: 10.1016/j.ress.2022.108445.

Zhang, H. & Zhou, W. (2022). A two-stage virtual machine abnormal behavior-based anomaly detection mechanism. *Cluster Computing*, 25(1), 203–214. doi: 10.1007/s10586-021-03385-2.

Zhang, W., Guo, W., Liu, X., Liu, Y., Zhou, J., Li, B., Lu, Q. & Yang, S. (2018). LSTM-Based Analysis of Industrial IoT Equipment. *IEEE Access*, 6, 23551–23560. doi: 10.1109/ACCESS.2018.2825538.

Zhang, X., Wang, J. & Gao, Y. (2019b). A hybrid short-term electricity price forecasting framework: Cuckoo search-based feature selection with singular spectrum analysis and SVM. *Energy Economics*, 81, 899–913. doi: 10.1016/j.eneco.2019.05.026.

Zhang, Y., Wang, Y. & Luo, G. (2020). A new optimization algorithm for non-stationary time series prediction based on recurrent neural networks. *Future Generation Computer Systems*, 102, 738–745. doi: 10.1016/j.future.2019.09.018.

Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C. & Sun, M. (2020). Graph neural networks: A review of methods and applications. *AI Open*, 1, 57–81. doi: 10.1016/j.aiopen.2021.01.001.

Zhu, Y., Zhang, W., Chen, Y. & Gao, H. (2019). A novel approach to workload prediction using attention-based LSTM encoder-decoder network in cloud environment. *Eurasip Journal on Wireless Communications and Networking*, 2019. doi: 10.1186/s13638-019-1605-z.

Zhu, Y., Wu, J., Wu, J. & Liu, S. (2022). Dimensionality reduce-based for remaining useful life prediction of machining tools with multisensor fusion. *Reliability Engineering and System Safety*, 218(PB), 108179. doi: 10.1016/j.ress.2021.108179.